
Internet-Wide Evaluations of Security and Vulnerabilities

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Genehmigte Dissertation von Tianxiang Dai aus China
Tag der Einreichung: 21/10/2022, Tag der Prüfung: 12/12/2022

1. Gutachten: Prof. Dr. Michael Waidner
2. Gutachten: Prof. Dr. Haya Shulman
3. Gutachten: Prof. Dr. Christian Rossow
Darmstadt



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Computer Science
Department

Technische Universität
Darmstadt

Security in Information
Technology (SIT) Research
Group

Internet-Wide Evaluations of Security and Vulnerabilities

Accepted doctoral thesis by Tianxiang Dai

Date of submission: 21/10/2022

Date of thesis defense: 12/12/2022

Darmstadt

Bitte zitieren Sie dieses Dokument als:

URN: urn:nbn:de:tuda-tuprints-234122

URL: <http://tuprints.ulb.tu-darmstadt.de/23412>

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de

Urheberrechtlich geschützt / In Copyright

<https://rightsstatements.org/page/InC/1.0/>

Darmstadt, Technische Universität Darmstadt

Jahr der Veröffentlichung der Dissertation auf TUpriints: 2023

For Yuanzao

Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 21/10/2022

T. Dai

Abstract

The Internet significantly impacts the world culture. Since the beginning, it is a multi-layered system, which is even gaining more protocols year by year. At its core, remains the Internet protocol suite, where the fundamental protocols such as IPv4, TCP/UDP, DNS are initially introduced. Recently, more and more cross-layer attacks involving features in multiple layers are reported. To better understand these attacks, e.g. how practical they are and how many users are vulnerable, Internet-wide evaluations are essential.

In this cumulative thesis, we summarise our findings from various Internet-wide evaluations in recent years, with a main focus on DNS. Our evaluations showed that IP fragmentation poses a practical threat to DNS security, regardless of the transport protocol (TCP or UDP). Defense mechanisms such as DNS Response Rate Limiting could facilitate attacks on DNS, even if they are designed to protect DNS. We also extended the evaluations to a fundamental system which heavily relies on DNS, the web PKI. We found that Certificate Authorities suffered a lot from previous DNS vulnerabilities. We demonstrated that off-path attackers could hijack accounts at major CAs and manipulate resources there, with various DNS cache poisoning attacks. The Domain Validation procedure faces similar vulnerabilities. Even the latest Multiple-Validation-Agent DV could be downgraded and poisoned.

On the other side, we also performed Internet-wide evaluations of two important defence mechanisms. One is the cryptographic protocol for DNS security, called DNSSEC. We found that only less than 2% of popular domains were signed, among which about 20% were misconfigured. This is another example showing how poorly deployed defence mechanisms worsen the security. The other is ingress filtering, which stops spoofed traffic from entering a network. We presented the most completed Internet-wide evaluations of ingress filtering, which covered over 90% of all Autonomous Systems. We found that over 80% of them were vulnerable to inbound spoofing.

This cumulative thesis includes contents from following papers and posters:

- [Dai16] T. Dai, H. Shulman, and M. Waidner. “DNSSEC Misconfigurations in Popular Domains”. In: *Cryptology and Network Security*. CANS ’16. 2016. CORE Rank B. Appendix A.1.

-
-
- [Dai18] M. Brandt et al. “Domain Validation++ For MitM-Resilient PKI”. in: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. 2018. CORE Rank A*. Appendix A.2.
- [Dai18p] T. Dai, H. Shulman, and M. Waidner. “Poster: Off-Path Attacks Against PKI”. in: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. 2018. CORE Rank A*. Appendix A.3.
- [Dai21i1] T. Dai, H. Shulman, and M. Waidner. “Poster: Fragmentation Attacks on DNS over TCP”. in: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. 2021. CORE Rank A. Appendix A.4.
- [Dai21a] T. Dai, H. Shulman, and M. Waidner. “DNS-over-TCP Considered Vulnerable”. In: *Proceedings of the Applied Networking Research Workshop*. ANRW ’21. 2021. Appendix A.5.
- [Dai21s] T. Dai et al. “From IP to Transport and beyond: Cross-Layer Attacks against Applications”. In: *Proceedings of the 2021 ACM SIGCOMM Conference*. SIGCOMM ’21. 2021. CORE Rank A*. Appendix A.6.
- [Dai21u] T. Dai et al. “The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021. CORE Rank A*. Appendix A.7.
- [Dai21c] T. Dai, H. Shulman, and M. Waidner. “Let’s Downgrade Let’s Encrypt”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’21. 2021. CORE Rank A*. Appendix A.8.
- [Dai21ac] T. Dai and H. Shulman. “SMap: Internet-Wide Scanning for Spoofing”. In: *Annual Computer Security Applications Conference*. ACSAC. 2021. CORE Rank A. Appendix A.9.

Contributions

The cumulative thesis includes contents of multiple papers and posters in recent years. Most works benefit from discussions and collaborations with other thoughtful researchers. In some cases, the contributions are so closely integrated that it is difficult to identify them individually, even though I try to declare my own contributions clearly. That said, I would only use first-person singular personal pronouns in this section.


This cumulative thesis includes contents of seven papers and two posters. All of them are cited with a prefix **Dai**. I am the only PhD co-author of four papers [Dai16; Dai21a; Dai21c; Dai21ac] and two posters [Dai18p; Dai21i1], with the valuable support from my supervisors Prof. Dr. Haya Shulman and Prof. Dr. Michael Waidner. The other three papers, [Dai18; Dai21u; Dai21s], were co-authored with other PhD candidates. [Dai18] is a joint work with Markus Brandt, who is another doctoral student at the Technical University of Darmstadt. A detailed declaration of contributions is attached in Appendix A.2. [Dai21u; Dai21s] are joint works with Philipp Jeitner, who is also a doctoral student at the Technical University of Darmstadt. Detailed declaration of contributions are attached in Appendix A.6 and A.7 accordingly.

Industrial Contributions

Apart from the academic publications included in this thesis, there are also some related industrial contributions I would like to mention.

Following the disclosure of our work [Dai18], major CAs blocked fragments to mitigate fragmentation-based vulnerabilities over UDP. The DNS community discussed the impact of fragmentation-based attacks and proposed to rely on TCP to avoid fragmentation [Com]. However, our work [Dai21a] pointed out that TCP was also vulnerable. That paper was presented at the ACM/IRTF Applied Networking Research Workshop 2021 (ANRW'21), which was hosted at IRTF meeting. The topic attracted various researchers and network operators. I was later invited for a guest post at the APNIC Blog about that, [Dai21ap1].

Following the disclosure of our work [Dai21u], I was invited for talks at NANOG 83 and RIPE 83, during which many network operators showed concerns about the work. RIRs



such as ARIN and RIPE stated that they would implement the countermeasures suggested by us. I was also invited for a guest post about that at the APNIC Blog, [Dai22ap1].

Acknowledgements

It has been almost seven years since I started working on research that has culminated in this thesis. Through all the years, I received a lot of support and help from many people. I am sincerely grateful to anyone who has ever helped me, even though I cannot mention all of them here.

I thank my supervisors Prof. Dr. Haya Shulman and Prof. Dr. Michael Waidner for all the guidance and support during my PhD. Special thanks to Prof. Dr. Christian Rossow for being my third referee. Thanks to Prof. Dr. Marc Fischlin, Prof. Dr. Max Mühlhäuser, and Prof. Dr. Zsolt Istvan for taking the time to serve on the committee.

Most research projects included in the thesis were not completed solely by myself. I thank all my collaborators - Markus Brandt, Philipp Jeitner, Haya Shulman, Michael Waidner. Also thanks to Birgit Blume, Karina Köhres and Michael Kreutzer for helping with many of the administrative tasks. Besides, I thank the IT team and the MPR team of Fraunhofer SIT for their professional technical support.

It is a pleasant experience to be involved in the Collaborative Research Center CROSSING. I thank Johannes Braun, Jacqueline Brendel, Marc Fischlin and Stefanie Kettler for keeping me in the loop and always inviting me to CROSSING events. I enjoyed all of them, especially all the CROSSING retreats. I am also grateful to CROSSING for offering the German courses, which make my life in Germany much easier.

I would like to give special thanks to Ranim Chakra, Daniel Senf, Kris Shrishak and Shujie Zhao. They are not only my colleagues, but also my friends in real life. I cannot imagine a PhD without them.

Last but not least, I thank all my families and friends who have shared time with me, wherever they are. Above all things, thanks to my wife, Yuanzao Zhu, for her unconditional support through all the years.

Tianxiang Dai
Leipzig, October 2022

Contents

Abstract	vii
Contributions	ix
Acknowledgements	xi
Contents	xiii
1. Introduction	1
1.1. Thesis Outline	2
1.2. Contributions	2
1.3. Papers and Posters	3
2. Evaluations on DNS Vulnerabilities	7
2.1. IP Fragmentation	7
2.1.1. Related Works	8
2.1.2. Contributions	10
2.2. Response Rate Limiting	10
2.2.1. Related Works	11
2.2.2. Contributions	11
2.3. DNSSEC	12
2.3.1. Related Works	12
2.3.2. Contributions	13
3. Evaluations on PKI Security	15
3.1. Related Works	16
3.2. Contributions	17
4. Evaluations on Ingress Filtering	19
4.1. Related Works	19
4.2. Contributions	21

5. Summary and Future Work	23
5.1. Summary	23
5.2. Future Work	24
Bibliography	27
Appendix	39
A. Papers and Posters	39
A.1. DNSSEC Misconfigurations in Popular Domains	39
A.2. Domain Validation++ For MitM-Resilient PKI	50
A.3. Poster: Off-path Attacks Against PKI	68
A.4. Poster: Fragmentation Attacks on DNS over TCP	72
A.5. DNS-over-TCP Considered Vulnerable	75
A.6. From IP to Transport and Beyond: Cross-Layer Attacks Against Applications	82
A.7. The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources	97
A.8. Let's Downgrade Let's Encrypt	117
A.9. SMap: Internet-wide Scanning for Spoofing	138

1. Introduction

Ever since the standardisation in the 1980s and the commercialisation in the 1990s, the Internet has significantly influenced the world culture and commerce. Nowadays, the majority of communications are carried over the Internet, in the form of instant messaging, email, telephony, video call, etc. So are the contents distributed to the whole world through the World Wide Web (WWW), via different services such as blogs, forums, social networks, news websites, etc. More and more activities are hosted in the Internet, too. To name a few, working, shopping, voting, teaching, etc. Statistics shows that in 2021, over 5 billions people used the Internet. That is about 66% of the whole world population [Sta]. And the number keeps increasing.

Even though there are various ways to access the Internet, e.g. wirelessly from mobile devices or wiredly from desktops, the core of the Internet remains the same, which is the Internet protocol suite [RFC1122; RFC1123]. It includes a suite of protocols classified into four conceptual layers. At the bottom is the link layer, which connects nodes on the local network segment (link). Above that is the internet layer, which enables hosts to identify and locate each other, as well as transport packets between each other. Among them, one of the most important protocols is the Internet Protocol (IP), which introduces IP addresses for host identification and location. In this thesis, we only mean IPv4 [RFC791]. Transport layer connects different hosts with a logical channel. Two typical examples are Transmission Control Protocol (TCP) [RFC793] for reliable delivery and User Datagram Protocol (UDP) [RFC768] for unreliable datagram service. On top of them is the application layer. The Domain Name System (DNS) [RFC1034; RFC1035] belongs here, which resolves human-friendly domain names to the numerical IP addresses. Another example is the Hypertext Transfer Protocol (HTTP) [RFC1945], the foundation of the World Wide Web. With the rising concerns on security and privacy, cryptographic protocols such as the Transport Layer Security (TLS) [RFC8446] get more adopted recently, which is also an application layer protocol.

Conceptual separation of functions makes it easier for protocol design and implementation. However, it also introduces interoperational issues in practice. Recently, more and more cross-layer attacks involving features in different layers are reported. One example is to use IP-fragmentation-based DNS cache poisoning to generate fraudulent certificate

for TLS. The basic idea is to make use of specific mechanism of IP, to mislead the DNS resolvers, which could then trick the authorities to issue fraudulent certificate. More details are in our paper [Dai18] and poster [Dai18p], also in Appendix A.2 and A.3.

To better understand these cross-layer attacks, e.g. how practical they are and how many users are vulnerable, Internet-wide evaluations are essential.

1.1. Thesis Outline

We performed many different Internet-wide evaluations from various perspectives in recent years, with a main focus on DNS. We would summarise our findings in this thesis. We start with our evaluations on DNS vulnerabilities in Chapter 2. More specifically, IP-fragmentation-related vulnerabilities in Section 2.1 and Response-Rate-Limiting-related vulnerabilities in Section 2.2. Additionally, we also evaluated the adoption of DNS Security Extensions (DNSSEC) [RFC4033; RFC4034; RFC4035] and misconfigurations in signed domains, which is presented in Section 2.3. In Chapter 3, we show our evaluations on the web Public Key Infrastructure (PKI) system, focusing on the Certificate Authorities (CA), concerning previous DNS vulnerabilities. Apart from those DNS-related topics, we present our evaluations on ingress filtering in Chapter 4. Considering that the ability of spoofing is required for all previously mentioned cross-layer attacks, the evaluations on ingress filtering are indeed relevant. We conclude the thesis in Chapter 5.

1.2. Contributions

In this thesis, the focus is on Internet-wide evaluations of vulnerabilities and secure mechanisms. We summarise our evaluations through the following contributions:

- **IP fragmentation poses a practical threat to DNS security.** We present two proof-of-concept attacks with IP fragmentation. One uses IP fragmentation to bypass security verification in DNS responses, which leads to cache poisoning and hijacking. The other makes use of fragment mis-association, which results in Denial of Service (DoS) and downgrade attacks. We show that DNS traffic over both TCP and UDP are vulnerable to IP-fragmentation-based attacks. Our latest evaluation indicated that over 55K domains were vulnerable.
- **DNS Response Rate Limiting facilitates attacks on DNS,** even if it is designed to protect DNS. DNS RRL could be abused to mute nameservers. Besides the recent side-channel-based cache poisoning which uses DNS RRL to extend the attack window,

we present a new downgrade attack with DNS RRL. The new attack makes the latest Multiple-Validation-Agents (Multi-VA) system vulnerable. Our latest evaluation showed that at least 20% of popular domains were vulnerable.

- **The low adoption and high misconfiguration of DNSSEC could not secure DNS.** At the time of evaluation, we found that only less than 2% of popular domains were signed. Among those signed domains, about 20% could not establish a chain of trust to the root zone. That is to say, 20% of signed domains might be inaccessible due to DNSSEC.
- **The current practices of CA operations are insecure.** We demonstrated that off-path attackers could hijack accounts at major CAs and manipulate resources there, with various DNS cache poisoning attacks. The Domain Validation (DV) procedure, a prerequisite for issuing a certificate, also faces similar vulnerabilities. We also present a new downgrade attack using IP-fragmentation and DNS RRL. Even the latest Multi-VA DV can be downgraded and poisoned. Our evaluation showed that all tested popular CAs were vulnerable to the new downgrade attack.
- **Ingress filtering is poorly adopted.** We performed Internet-wide evaluations of ingress filtering using standard protocols such as IPv4, PMTUD and DNS. Thanks to the availability of these standard protocol in almost all networks, our evaluations covered over 90% of all ASes, much more than all other related studies. We found that over 80% of tested ASes were vulnerable to inbound spoofing.

1.3. Papers and Posters

In this thesis, contents from the following papers and posters are included. The original versions are also attached in Appendix A.

- [Dai16] Tianxiang Dai, Haya Shulman, and Michael Waidner. “DNSSEC Misconfigurations in Popular Domains”. In: *Cryptology and Network Security*. Ed. by Sara Foresti and Giuseppe Persiano. Cham: Springer International Publishing, 2016, pp. 651–660. ISBN: 978-3-319-48965-0. CORE Rank B. Appendix A.1.

-
- [Dai18] Markus Brandt et al. “Domain Validation++ For MitM-Resilient PKI”. in: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 2060–2076. ISBN: 9781450356930. DOI: 10.1145/3243734.3243790. URL: <https://doi.org/10.1145/3243734.3243790>. CORE Rank A*. Appendix A.2.
- [Dai18p] Tianxiang Dai, Haya Shulman, and Michael Waidner. “Poster: Off-Path Attacks Against PKI”. in: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 2213–2215. ISBN: 9781450356930. DOI: 10.1145/3243734.3278516. URL: <https://doi.org/10.1145/3243734.3278516>. CORE Rank A*. Appendix A.3.
- [Dai21i1] Tianxiang Dai, Haya Shulman, and Michael Waidner. “Poster: Fragmentation Attacks on DNS over TCP”. in: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. 2021, pp. 1124–1125. DOI: 10.1109/ICDCS51616.2021.00118. CORE Rank A. Appendix A.4.
- [Dai21a] Tianxiang Dai, Haya Shulman, and Michael Waidner. “DNS-over-TCP Considered Vulnerable”. In: *Proceedings of the Applied Networking Research Workshop*. ANRW ’21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 76–81. ISBN: 9781450386180. DOI: 10.1145/3472305.3472884. URL: <https://doi.org/10.1145/3472305.3472884>. Appendix A.5.
- [Dai21s] Tianxiang Dai et al. “From IP to Transport and beyond: Cross-Layer Attacks against Applications”. In: *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. SIGCOMM ’21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 836–849. ISBN: 9781450383837. DOI: 10.1145/3452296.3472933. URL: <https://doi.org/10.1145/3452296.3472933>. CORE Rank A*. Appendix A.6.
- [Dai21u] Tianxiang Dai et al. “The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3147–3164. ISBN: 978-1-939133-24-3. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/dai>. CORE Rank A*. Appendix A.7.

-
- [Dai21c] Tianxiang Dai, Haya Shulman, and Michael Waidner. “Let’s Downgrade Let’s Encrypt”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 1421–1440. ISBN: 9781450384544. DOI: 10.1145/3460120.3484815. URL: <https://doi.org/10.1145/3460120.3484815>. CORE Rank A*. Appendix A.8.
- [Dai21ac] Tianxiang Dai and Haya Shulman. “SMap: Internet-Wide Scanning for Spoofing”. In: *Annual Computer Security Applications Conference*. ACSAC. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 1039–1050. ISBN: 9781450385794. DOI: 10.1145/3485832.3485917. URL: <https://doi.org/10.1145/3485832.3485917>. CORE Rank A. Appendix A.9.

2. Evaluations on DNS Vulnerabilities

Domain Name System (DNS), [RFC1034; RFC1035], was initially designed to translate human-friendly domain names to numerical IP addresses. It plays a key role in the Internet. Almost all services using domain names rely on DNS, e.g. web and email. With the rapid development of the Internet, DNS has evolved into an extremely complex ecosystem. The complexity of DNS ecosystem is continuously growing, as its usage and client base keep increasing.

Considering the rising utilisation of DNS, attacks against DNS also increase, most notably, DNS cache poisoning, [HS13a; HS13b; HS13c; Kam08; SW14; Ste03]. In a DNS cache poisoning attack, the attacker injects malicious DNS records into the victim's resolver, so as to redirect the victim to an attacker-controlled malicious host for further exploits. DNS cache poisoning attacks are not only practiced by cyber criminals, for credential theft or malware distribution, but also performed by governments, for censorship [And12] or surveillance [Hu15].

In this chapter, we evaluate DNS vulnerabilities in the Internet. More specifically, IP-fragmentation-related evaluations in Section 2.1 and response-rate-limiting-related evaluations in Section 2.2. Besides, in Section 2.3, we measure adoption and misconfigurations of a well-standardised security mechanism for DNS, Domain Name System Security Extensions (DNSSEC) [RFC4033; RFC4034; RFC4035].

2.1. IP Fragmentation

IP fragmentation is a mechanism defined in Internet Protocol (IP) [RFC791]. By saying IP fragmentation, we mean only IPv4 fragmentation. IPv6 fragmentation is a different topic, which we would not cover. With fragmentation, large packets are broken into smaller pieces (fragments), so that they can be transmitted over network links with smaller Maximum Transmission Unit (MTU) than the original packets. The fragments will get reassembled later by the receiver. Both the sender and the routers can fragment packets, while only the receiver can reassemble them. Fields in the IPv4 header including source address, destination address, protocol number as well as identification number (IPID) are

used to verify if the fragments belong to the same packet [RFC815].

DNS is transmitted over IP and is vulnerable to fragmentation-based attacks. Assume that a server sends a fragmented packet to a client. By sending a spoofed fragment to the victim client, which will get reassembled with other genuine fragments from the server, the attacker could launch different attacks. If the reassembled packet is invalid and get dropped, this results in a Denial of Service (DoS) attack against the server [KM95]. If the reassembled packet is valid but contains malicious payload from the spoofed fragment, this results in an injection attack, which can be used for DNS cache poisoning [HS13a]. With Internet Control Message Protocol (ICMP) [RFC792] fragmentation needed error message (type: 3, code: 4), the attacker could even trigger source fragmentation.

In this section, we will briefly introduce related works about DNS cache poisoning and fragmentation-based attacks. Then present our contributions in this field. This section includes contents in our following papers and posters:

- [Dai18], also in Appendix A.2
- [Dai18p], also in Appendix A.3
- [Dai21i1], also in Appendix A.4
- [Dai21a], also in Appendix A.5
- [Dai21s], also in Appendix A.6
- [Dai21u], also in Appendix A.7
- [Dai21c], also in Appendix A.8

2.1.1. Related Works

History of DNS Cache Poisoning

Back in 1995, Vixie [Vix95] revealed the cache poisoning vulnerability and proposed to randomise User Datagram Protocol (UDP) source ports in DNS requests. In 2002, Bernstein [Ber] also warned that only randomising the Transaction ID (TXID) field could not provide enough entropy, which made DNS vulnerable. In 2007, Klein found vulnerabilities in Bind9 [Kle07a] and in Windows resolver [Kle07b], which allowed off-path attackers to guess the TXID more easily. In 2008, Kaminsky [Kam08] presented a practical DNS cache poisoning attack, even against fully randomised TXID. To mitigate that, source port randomisation was standardised in [RFC5452].

However, the safety didn't last long. New attacks to bypass both source port and TXID randomisation came up. In 2012, Herzberg et al. [HS12] showed that source port could be inferred with side channels. More studies on side channels followed. Man et al. [Man+20] used ICMP side channel and Klein [Kle21] used Pseudo-Random Number Generator (PRNG) side channel, to predict UDP source port. In 2013, Herzberg et al. [HS13a] proposed a new DNS cache poisoning attack using IPv4 fragmentation. The IPv4-fragmentation-based cache poisoning was further used to attack PKI [Dai18] and Network Time Protocol (NTP) [JSW20]. In 2015, Shulman et al. [SW15] showed that it was possible to attack resolvers behind upstream forwarders. This attack was further extended for stub resolvers [Alh+19] and forwarding devices [Zhe+20]. In 2018, Birge et al. [Bir+18] presented another DNS cache poisoning attack via short-lived Border Gateway Protocol (BGP) prefix hijacking.

Fragmentation-based Attacks

Back in 1995, Kent et al. [KM95] warned IP fragmentation could possibly lead to Denial of Service (DoS). Gilad et al. [GH11] further explored this kind of DoS attack under the network setups of Network Address Translation (NAT) or tunnelling. In 2004, Zalewski [Zal] mentioned that it was possible to inject meaningful data into Transmission Control Protocol (TCP) sessions with IP fragmentation. In 2013, Herzberg et al. [HS13a; HS13c] found that it was possible to launch DNS cache poisoning attack with IP fragmentation, bypassing both UDP source port and TXID randomisation. Shulman et al. [SW14; JSW20; SW15] evaluated this vulnerability in the Internet and presented more practical scenarios, such as NTP. In 2016, Malhotra et al. [Mal+16] noticed different implementations of IP reassembly when fragments overlapped and demonstrated how to use that to shift time in NTP.

Despite of those attacks, not many defensive mechanisms are proposed. In 2003, Kaufman et al. [KPS03] argued that most protocols running on top of TCP could avoid the need for fragmentation. In 2015, Zhu et al. [Zhu+15] proposed to use TCP and Transport Layer Security (TLS) for DNS so as to improve privacy and security. Afterwards, DNS over TCP was updated in [RFC7766], addressing the concern on fragmentation. DNS over TLS (DoT) was later standardised as [RFC7858]. More recently, best practices [RFC8900; FV21] considered TCP with Path Maximum Transmission Unit Discovery (PMTUD) as a solution to IP fragmentation. Besides, Let's Encrypt announced to reduce Extension Mechanisms for DNS (EDNS) buffer size from 4,096 to 512 [Enca]. DNS operation community recommended to reduce default EDNS buffer size from 4,096 to 1,232 [Com]. Both tried to send large DNS responses over TCP, in order to avoid fragmentation.

2.1.2. Contributions

We evaluated DNS vulnerabilities in the Internet concerning IP fragmentation. Our contributions are:

- We show that off-path fragmentation-based DNS cache poisoning is a practical threat. In our paper [Dai18] and poster [Dai18p], we demonstrated how to issue fraudulent certificates with fragmentation-based cache poisoning and verified that at least 7 Certificate Authorities (CAs) were vulnerable. In our paper [Dai21u], we found about 17% of 94,997 accounts at different providers were vulnerable to fragmentation-based cache poisoning. Their resources could be hijacked due to that. We extended the evaluation to 904,555 domains in our paper [Dai21s], of which about 4% were vulnerable to fragmentation-based cache poisoning. Later, in our paper [Dai21c], we evaluated 1,858,165 domains and identified that 3% of the domains were vulnerable.
- We show that IP fragmentation attacks also apply to servers that communicate over TCP. In our paper [Dai21a] and poster [Dai21i1], we evaluated nameservers in the Alexa Top-100K domains [Ama]. We found 496 domains whose nameservers could be forced to fragment DNS responses over TCP. Among them, 393 domains were only vulnerable when transmitting over TCP, instead of UDP. Moreover, 76 of the 393 domains' nameservers had TrunCation (TC) bit set when responding over UDP. They followed the best practices to avoid fragmentation with TCP, which actually made them vulnerable.
- We show that IP fragmentation can be used to eliminate nameservers so as to manipulate server selection at resolvers, which could facilitate attacks on DNS. This is part of a new downgrade attack against the latest Multiple-Validation-Agents (Multi-VA) system [Encb], as presented in our paper [Dai21c].
- We provide a free online tool to test if a user's resolvers are vulnerable to fragmentation-based DNS cache poisoning. It can be accessed at <https://crosslayerattacks.sit.fraunhofer.de>.

2.2. Response Rate Limiting

DNS Response Rate Limiting (DNS RRL) [VS12] is a method to limit the rate of responses by a DNS nameserver, in order to mitigate DNS reflection and amplification attacks. Such attacks send a large quantity of malicious DNS requests to a DNS nameserver with spoofed

source IPs. By configuring the limit of responses per-IP, per-prefix or even globally, the nameserver can either drop the responses or truncate them, once the limit is reached. All modern DNS software such as BIND, NSD and PowerDNS, support this security feature. DNS hosting service providers such as Cloudflare also offer similar feature.

However, this feature can be abused to mute a nameserver, if the attacker can spoof DNS requests to the nameserver with the victim resolver's IP as source address. Once the rate is high enough to trigger RRL for the victim resolver, the resolver can hardly receive any response from the nameserver.

In this section, we will briefly introduce related works about DNS RRL. Then present our contributions in this field. This section includes contents in our following papers:

- [Dai21s], also in Appendix A.6
- [Dai21u], also in Appendix A.7
- [Dai21c], also in Appendix A.8

2.2.1. Related Works

There are not so many works on RRL. In 2012, Vixie et al. [VS12] proposed RRL to blunt the impact of DNS reflection and amplification attacks. Soon the method was implemented in BIND. Afterwards, it was well acknowledged as a good mitigation of Distributed DoS (DDoS) attacks [RMK13; CIS]. However, MacFarland et al. [MSK15] found that only 2.69% of the studied domains employed rate limiting in 2015, indicating the approach was not widely used in practice. They evaluated again in 2017 [MSK17], and found that 10.23% of nameservers employed the protective measure. In 2019, Deccio et al. [DAD19] found that about 16% of authoritative DNS nameservers employed some sort of rate limiting.

On the other side, Man et al. [Man+20] revealed that DNS RRL could be leveraged maliciously to mute a nameserver. They used it to extend the attack window so as to facilitate their DNS cache poisoning attack with ICMP side channel. Their measurement showed that 18% of Alexa Top-100K domains had RRL enabled, thus were vulnerable to the attack.

2.2.2. Contributions

We evaluated DNS vulnerabilities in the Internet concerning DNS RRL. Our contributions are:

-
- We present up-to-date Internet-wide measurements on DNS RRL, concerning its involvement in side-channel-based DNS cache poisoning. In our paper [Dai21u], we found about 10% of 94,997 accounts at different providers were vulnerable to side-channel-based cache poisoning. Their resources could be hijacked due to that. We extended the evaluation to 904,555 domains in our paper [Dai21s], of which about 12% were vulnerable to side-channel-based cache poisoning. Afterwards, in our paper [Dai21c], we evaluated 1,858,165 domains and identified that about 20% of the domains had DNS RRL enabled.
 - We show that DNS RRL can be used to eliminate nameservers so as to manipulate server selection at resolvers, which could facilitate attacks on DNS. This is part of a new downgrade attack against the latest Multi-VA system, as presented in our paper [Dai21c].

2.3. DNSSEC

Domain Name System Security Extensions (DNSSEC) [RFC4033; RFC4034; RFC4035], was designed and standardised to protect DNS against DNS cache poisoning. It introduces signatures for any authoritative DNS records, in order to provide data authentication and data integrity. Besides, it also offers authenticated denial of existence. When a resolver receives a signed DNS response, it could verify the data with the signature and the public key, which is similar to the web Public Key Infrastructure (PKI). Even though DNSSEC was officially standardised in 2005, it is still not widely adopted and deployed [HS14].

In this section, we will briefly introduce related works about DNSSEC evaluations and describe our contributions. For more details, please check our paper [Dai16], also in Appendix A.1.

2.3.1. Related Works

Measurements on DNSSEC show that DNSSEC deployment is rare but increasing. In 2011, Yang et al. [Yan+10] estimated that there were only about 10,523 DNSSEC enabled zones operating in the wild. In 2014, Herzberg et al. [HS14] found that only 2% of the top 300,000 domains from Alexa [Ama] support DNSSEC. In 2017, Chung et al. [Chu+17a] observed between 0.6% (.com) and 1.0% (.org) of domains have DNSKEY records published.

On the resolver side, in 2013, Fukuda et al. [FSM13] revealed that less than 50% of the potential DNSSEC validators were validating caching resolvers in the wild. Lian et al. [Lia+13] showed that the fraction of clients which actually validate DNSSEC-signed

records was less than 3%. In 2017, Chung et al. [Chu+17a] found only 12% of the DNSSEC-aware resolvers correctly validated DNSSEC responses. Measurements from Asia-Pacific Network Information Centre (APNIC) show that currently about 26% of the DNS resolvers validate DNSSEC [APN].

Besides the low adoption, DNSSEC deployments suffer from misconfigurations. Herzberg et al. [HS14] found only three out of 14 US popular domains adopted DNSSEC correctly. Shulman et al. [SW17] showed that 35% of signed domains shared moduli with others and 66% used weak keys. Chung et al. [Chu+17a] identified about one thousand keys shared by multiple domains. And one key was even shared by over 132,000 domains! They also studied how registrars make things worse [Chu+17b].

There are already some tools to study DNSSEC. OARC's DNS Reply Size Test Server [OAR] is an online service for testing response size of DNS. Users can evaluate the maximum response size that their network can support, which is critical for DNSSEC adoption, as DNSSEC-related responses normally exceed the default maximum size of 512 bytes. Both DNSViz [San] and DNSSEC Analyzer [Ver] analyse all the keys the domain has and the signatures over those keys. They also check if it is possible to establish a chain of trust from the root to the target domain. SecSpider [Uni] provides metrics for DNSSEC global deployment, by collecting DNSSEC-relevant records from the zones.

2.3.2. Contributions

Our paper [Dai16], also in Appendix A.1 has following contributions:

- We designed a DNSSEC evaluation tool. It collects DNSSEC-relevant records from the Internet, analyses the misconfigurations and generates statistics automatically. The reports quantify two types of vulnerabilities in signed domains: cryptographic failures (e.g. broken chain of trust or vulnerable DNSSEC keys) and transport failures (e.g. lack of TCP or EDNS support).
- We found that 90% of Top Level Domains (TLDs) and 1.66% of Alexa domains were signed. Among signed domains, 0.89% TLDs and 19.46% Alexa domains could not establish a chain of trust to the root zone. 12.88% of Alexa domains had nameservers which could not serve DNS responses over TCP.
- Different from existing tools, our tool allows to study insecurity and misconfigurations on a given domain, as well as historical within a given time period. It can be accessed at <https://dnssec.cad.sit.fraunhofer.de>.

3. Evaluations on PKI Security

Public Key Infrastructure (PKI) is a system to create, store, and distribute digital certificates, which can be used to prove that a particular public key belongs to a certain entity. PKI has been used in different applications for security purpose, e.g. web, email and code signing. In this thesis, we mainly discuss web PKI, which is used by Hypertext Transfer Protocol Secure (HTTPS), for secure and private web browsing.

In web PKI system, if a certificate would like to be accepted by a browser automatically, it must be chained up to a trusted root certificate. The trusted root certificate list is managed by either the operating system or by the browser. In either case, a website could not generate a certificate on its own and have it trusted by anyone else in the Internet. To achieve that, the owner or administrator has to visit a trusted Certificate Authority (CA), or its reseller, to issue a trusted certificate. During the issuance procedure, one of the most important steps is Domain Validation (DV), which validates if the applicant actually controls the domain or not.

With the increasing concerns on security and privacy, HTTPS is used by more and more websites and web users. BuiltWith shows that over 60% of popular websites in the Internet redirect traffic to an HTTPS version by default [Bui]. Google announces that 95% of its services are served over HTTPS and over 80% of pages loaded in Chrome browser are over HTTPS [Gooc]. Firefox shows similar statistics [Moza]. As the fundament of HTTPS, the PKI system, more specifically, the procedure of certificate issuance and management are of great importance.

Considering that a certificate binds a public key to one or more domain names, the web PKI system has a strong connection to DNS. Therefore, PKI security is associated with DNS vulnerabilities, which are mentioned in Chapter 2.

In this chapter, we will present our evaluations on PKI security, which are introduced by the DNS vulnerabilities mentioned in Chapter 2. We start with related works on PKI security in Section 3.1. Then show our contributions in Section 3.2. This chapter includes contents in our following papers and posters:

- [Dai18], also in Appendix A.2
- [Dai18p], also in Appendix A.3

-
- [Dai21s], also in Appendix A.6
 - [Dai21u], also in Appendix A.7
 - [Dai21c], also in Appendix A.8

3.1. Related Works

CA Compromises

A certificate signed by any trusted CA is accepted by all the browsers. As a consequence, any vulnerable or compromised CA could subvert the security of any domain in the Internet. In 2011, one regional CA, Dutch DigiNotar, issued many fraudulent certificates against more than 20 different domains, including `*.google.com`, due to a security breach. Later DigiNotar was removed from root list of all browsers [Wik]. In 2013, another regional CA, TURKTRUST, disclosed to have mistakenly issued two intermediate CA certificates, which were later detected to issue fraudulent certificate for `*.google.com` [Goob]. In 2016, Chinese CA WoSign reported a bug which allowed to issue a certificate for the base domain if the applicant were able to prove control of any subdomain. For example, a user of `someone.github.com` could get a certificate of `github.com`. Considering the severe breaches, WoSign was also removed from the root list [Mozc]. Recently, one of the largest CAs, Symantec, was distrusted by all browsers due to various issues [Mozb].

Domain Validation

Domain Validation (DV) is a process to verify if an applicant has control over a domain. The most classic way is to send an email with verification code to some reserved email addresses under that domain, e.g. `webmaster@domain` and `admin@domain`. However, it was reported that in rare cases, the reserved email addresses could be registered by normal users [Gooa]. Other DV methods such as DNS DV and HTTP DV, require the applicant to modify either the DNS zone or the web server. In any case, DV heavily relies on DNS, thus suffers from DNS hijacking. Many attacks were reported recently, [18; 19b; 19c; 19a]. During these attacks, the attackers gained access to victim DNS zones and hijacked DNS. Then they could easily issue fraudulent certificates and establish phishing websites. Most of the time, social engineering was involved to compromise the victim accounts. In 2018, Birge et al. [Bir+18] evaluated the impact of BGP prefix hijacks on domain validation. They also mentioned that with BGP prefix hijacking, an attacker could redirect DNS traffic and hijack a whole domain. A similar attack was involved in the `myetherwallet.com` hijack [S G18].

Distributed Domain Validation

Following the studies in 2018 [Bir+18; Dai18], Let's Encrypt deployed domain validation from multiple vantage points [Bir+21; Encb]. The concept of this kind of distributed validation is not new. In 2004, Park et al. [Par+04] proposed CoDNS to achieve low-latency, low-overhead and high-reliability name resolution. Poole et al. [PP06] further extended CoDNS with multi-site agreement and per-site lookup histories, for better security. In 2008, Wendlandt et al. [WAP08] proposed Perspectives, a notary service involving multiple network vantage points, to improve host authentication. Instead of deploying servers in different locations, Alicherry et al. [AK09] used the Tor [DMS04] network to achieve similar goal. However, except for Let's Encrypt's Multi-VA [Encb], none of those proposals are deployed and adopted by the community, due to additional modifications to the existing infrastructure and the lack of motivation.

3.2. Contributions

We evaluated PKI security in the Internet concerning DNS-related vulnerabilities. Our contributions are:

Resource Management

- We demonstrate that the current practices of Internet resource management are insecure. In our paper [Dai21u], we demonstrated that off-path attackers could hijack accounts at major CAs and manipulate resources there. We found that all the five tested major CAs were vulnerable to fragmentation-based DNS cache poisoning, as showed in Section 2.1. Two of them did not even validate DNSSEC. All the five CAs did not use CAPTCHA, which made it possible to repeat the password recovery for hijacking. When controlling an account at a CA, the attacker could easily revoke or reissue existing certificates, sometimes even without additional domain validation. Those new certificates could be further used for phishing or other cyber crimes.

Domain Validation

- We show that DV is vulnerable to off-path DNS cache poisoning attacks. In our paper [Dai18] and poster [Dai18p], we demonstrated how to issue fraudulent certificates with fragmentation-based cache poisoning and verified that at least 7 CAs were vulnerable. In our recent work [Dai21s], we found that even though DV of five tested

CAs were not vulnerable to fragmentation-based (Section 2.1) or side-channel-based (Section 2.2) DNS cache poisoning, most of them were vulnerable to BGP hijacking.

- We show that even latest Multi-VA is vulnerable to DNS-related downgrade attacks. In our paper [Dai21c], we developed off-path downgrade attacks to reduce the domain validation to be performed against a single, attacker-selected nameserver. The attacks involved either fragmentation (Section 2.1) or RRL (Section 2.2). Our evaluation showed that all tested popular CAs, including Let's Encrypt with Multi-VA, were vulnerable to the attacks.

4. Evaluations on Ingress Filtering

IP spoofing allows attackers to send packets with a false source IP address so as to impersonate other hosts in the Internet, which could avoid attack source detection or filtering. This is weaponised for reflecting traffic during Distributed Denial of Service (DDoS) attacks [Ros14]. Recent DNS cache poisoning attacks also make use of IP spoofing [Dai18; Man+20].

The recommended method to mitigate IP spoofing is to deploy Source Address Validation (SAV) on all packets. There are two types of SAV. One is ingress filtering, which inspects inbound traffic at the receiving side. The other is egress filtering, which filters outbound packets at the sending side. A specific Best Current Practice (BCP) addressing that was standardised in 2000 as BCP38 [RFC2827]. BCP38 proposed to use ingress traffic filtering to prohibit DoS attacks with IP spoofing, which was further updated in BCP84 [RFC3704].

In this chapter, we will briefly introduce related works about measurements on ingress filtering and describe our contributions. For more details, please check our paper [Dai21ac], also in Appendix A.9.

4.1. Related Works

Egress vs. Ingress

Most SAV-related studies focus on egress filtering, while only a few discuss about ingress filtering. Even fewer studies compare them. In 2019, Luckie et al. [Luc+19] showed that at least a quarter of tested ASes did not deploy egress filtering while at least two thirds of tested ASes did not deploy ingress filtering, even though ingress filtering protects their own network. In 2020, Jonglez et al. [JD20] looked into 559 /24 networks. They found 298 networks without ingress filtering only and 15 without egress only, which indicates that ingress filtering is less deployed than egress. Later they [Kor+20b] showed that out of 515 ASes within the Mutually Agreed Norms for Routing Security (MANRS), 81 ASes were vulnerable to outbound spoofing, while as many as 114 and 207 ASes were fully and partially vulnerable to inbound spoofing. The correlation between egress and ingress

filtering in these studies show that the measurements of networks without ingress filtering could provide a lower bound on the number of networks without egress filtering.

Vantage Points

The Spoofer Project [CAI; BB05] first presented measurements of networks without egress filtering in 2005. The idea is to craft specific packets with spoofed source IP addresses and send them from vantage points, or volunteers, to several servers in different locations. The disadvantage of this approach is that it requires the volunteers to install and run a software with administrative privileges, which brings trust and permission issues. Even though promoted for eight years, Spoofer only covered 1,586 ASes in 2013 [BKC13], less than 5% of all ASes. In 2015, Huz et al. [Huz+15] addressed the concern of coverage and proposed to use crowdsourcing for vantage-point-based measurements. Afterwards, Lone et al. [Lon+18] reported the results with crowdsourcing from the Spoofer Project. They discovered 342 new ASes, a 15% increase than previous year. In the latest publication of the Spoofer Project [Luc+19], 5,178 ASes were included, among which 31.5% were spoofable. Despite a big increase, it only covered less than 10% of all ASes. And the coverage across networks and geolocations was not uniform.

Network Traces

Alternative approaches using network traces to infer spoofing abilities are proposed to overcome the dependency on vantage points. In 2017, Lone et al. [Lon+17] used loops in traceroute to detect absence of ingress filtering on provider ASes. They identified loops in 1,780 ASes, 3.2% of all the ASes, and 703 of the ASes were spoofable. However, this technique has significant limitations. It requires support for traceroute, and misconfigurations which create loops. Both can be challenging in practice. Lichtblau et al. [Lic+17] proposed and evaluated a method to passively detect spoofed packets in the traffic exchanged at a large European Internet Exchange Point (IXP). They covered over 700 networks, among which over 50% were vulnerable. Apparently, this approach requires to have access to the data from the IXP and can only include networks sending traffic through the IXP.

Remote Scanning

With the help of certain reflective features or side channels, it is possible to detect SAV from remote. Kühner et al. [Küh+14] made use of a broken resolver implementation to detect absence of egress filtering. Even though this approach only applies to the networks

with such misconfigured DNS resolvers, it reported 2,692 ASes without egress filtering, even more than the Spoofer Project. This shows the power of Internet-wide scans. The Closed Resolver Project [KN] uses a different method of remote scanning. They send DNS queries for their own domains to DNS resolvers all over the Internet, with spoofed source IP. With the receipt of DNS queries at their own nameserver, they could confirm if the remote networks have received the spoofed queries, which means absence of ingress filtering. In their latest publication [Kor+20a], they reported 32,673 IPv4 ASes to be vulnerable to spoofing of inbound traffic, which is a big increase, compared to all previous studies.

4.2. Contributions

Our paper [Dai21ac], also in Appendix A.9 has following contributions:

- We present the Spoofing Mapper (SMap), a tool to perform Internet-wide studies of ingress filtering. SMap evaluates spoofability using standard protocols such as IPv4, PMTUD and DNS, which are present in almost any networks. SMap allows fully remote scanning without cooperation of remote networks. It is easily reproducible and scalable.
- SMap covers the most networks among all SAV-related studies. Recent scans covered 63,522 ASes, over 90% of all ASes. Among them, 51,046 were vulnerable to inbound spoofing.
- We publish the statistics online and share the data on demand. Network administrators could also use our online form to perform remote tests of their own networks. SMap can be accessed at <https://smap.cad.sit.fraunhofer.de>.

5. Summary and Future Work

In this chapter, we conclude the thesis by summarising our main contributions in Section 5.1 and proposing potential future work directions in Section 5.2.

5.1. Summary

This thesis brings significant contributions on Internet-wide evaluations of DNS-related vulnerabilities and security mechanisms. We summarise our findings below.

DNS Vulnerabilities

In Chapter 2, we present Internet-wide evaluations of DNS vulnerabilities. Section 2.1 includes evaluations of IP fragmentation from our papers [Dai18; Dai21a; Dai21s; Dai21u; Dai21c] and posters [Dai18p; Dai21i1]. We show that DNS traffic over both TCP and UDP are vulnerable to IP-fragmentation-based attacks, which could result in hijacking or DoS. Our latest evaluation indicated that over 55K domains were in danger. Section 2.2 contains evaluations of DNS RRL from our papers [Dai21s; Dai21u; Dai21c]. We warn that even if RRL is designed to protect DNS, it actually facilitates attacks on DNS. We show how RRL can be abuse to mute nameservers, so as to downgrade the latest Multi-VA system for DNS cache poisoning. Our latest evaluation showed that at least 20% of popular domains were vulnerable. Section 2.3 summarises evaluations of DNSSEC from our paper [Dai16]. We found that only less than 2% of popular domains were signed, while about 20% of those signed domains were misconfigured.

PKI Security

In Chapter 3, we present evaluations of PKI security, focusing on the CA side. We demonstrated that off-path attackers could hijack accounts at major CAs and manipulate resources there, as shown in our paper [Dai21u]. The DV procedure of major CAs faces similar vulnerabilities. Our papers [Dai18; Dai21s] and poster [Dai18p] found off-path DNS

cache poisoning attacks as a practical threat to DV. Even the latest Multi-VA DV can be downgraded and attacked via cache poisoning, as presented in our paper [Dai21c].

Ingress Filtering

In Chapter 4, we present Internet-wide evaluations of ingress filtering. We only use standard protocols such as IPv4, PMTUD and DNS to perform remote test without any cooperation from the target network. Our evaluations covered the most ASes among all related studies, over 90%. And we found absence of ingress filtering in over 80% of tested ASes, as stated in our paper [Dai21ac].

5.2. Future Work

DNS Security

Chapter 2 presents our evaluations of DNS vulnerabilities. It also points out a severe issue of existing defensive mechanisms. That said, even if they are designed to protect DNS, they indeed make it worse, under specific circumstances. RRL could reduce the possibilities of DDoS, while it could be abused to facilitate attacks, as shown in Section 2.2. DNSSEC increases the size of DNS responses, which makes them easier to fragment, resulting in fragmentation-related vulnerabilities mentioned in Section 2.1. The community [Com] suggested to move on to TCP to mitigate that. However, the evaluations in our paper [Dai21a] and poster [Dai21i1] revealed that even DNS over TCP was vulnerable. More researches are needed in the field of design and deployment of these defensive mechanisms, so that they could protect the Internet without bringing new vulnerabilities.

There are also other security protocols for DNS. For example, DNS over TLS (DoT) [RFC7858] and DNS over HTTPS (DoH) [RFC8484]. Both of them transmit DNS traffic over encrypted channels to avoid eavesdropping and data manipulation, so as to increase user privacy and security. However, they only secure the path between the clients and the resolvers. The traffic between the resolvers and the nameservers are still unprotected, which is left for DNSSEC. Moreover, the DNS traffic aggregation at the resolver providers brings huge debates on privacy issues [Cor]. As there are only few DNS providers offering DoT or DoH services, people are indeed worried that those giant providers such as Cloudflare and Google would get too much data. Researches on security-enhancing as well as better privacy-preserving mechanisms are also promising.

PKI Security

Following the attacks against DV [Bir+18; Dai18; Dai18p], Let's Encrypt deployed DV from multiple vantage points (Multi-VA) [Encb]. Even if secure against previously mentioned attacks [Bir+21], we found it vulnerable to downgrade attacks, as showed in our paper [Dai21c]. We pointed out that the main pitfall of Let's Encrypt's Multi-VA deployment was the limited selection of VAs, which allowed the attacker to investigate and predict in advance. Better distribution and selection of VAs should have made Multi-VA even secure against strong MitM adversaries. Studies on this are yet missing.

Another important direction of PKI security is about monitoring and revoking. Certificate Transparency (CT) [Lau14] allows to detect fraudulent certificates quickly so that CAs could revoke them once alarmed. However, CT monitors are sometimes unreliable [Li+19] and it might take a few hours to detect those fraudulent certificates [Dai21c]. During that undetected time, the damage of the attacks would have already taken place [S G18]. Better automation of certificate monitoring and revoking should also get more attentions.

Ingress Filtering

Chapter 4 summarises the findings on ingress filtering in our paper [Dai21ac]. Surprisingly, even if ingress filtering protects the network itself, it is much less deployed. Lichtblau et al. [Lic+17] mentioned two main reasons for not filtering traffic. One is that it might falsely drop legitimate traffic from customers. The other is that filtering required a lot of planning, configuration and maintenance, which was too difficult and too much work. These bring a new question for ingress filtering. How to do it more accurately and easily? More researches are expected.

Bibliography

- [18] *DNSpionage Campaign Targets Middle East*. <https://blog.talosintelligence.com/2018/11/dnspionage-campaign-targets-middle-east.html>. Accessed: 2021-01-19. 2018.
- [19a] *'Unprecedented' DNS Hijacking Attacks Linked to Iran*. Accessed: 2021-1-19. 2019. URL: <https://threatpost.com/unprecedented-dns-hijacking-attacks-linked-to-iran/140737/>.
- [19b] *Global DNS Hijacking Campaign: DNS Record Manipulation at Scale*. <https://www.fireeye.com/blog/threat-research/2019/01/global-dns-hijacking-campaign-dns-record-manipulation-at-scale.html>. Accessed: 2021-1-19. 2019.
- [19c] *Sea Turtle keeps on swimming, finds new victims, DNS hijacking techniques*. <https://blog.talosintelligence.com/2019/07/sea-turtle-keeps-on-swimming.html>. Accessed: 2021-01-19. 2019.
- [AK09] Mansoor Alicherry and Angelos D Keromytis. "Doublecheck: Multi-path verification against man-in-the-middle attacks". In: *2009 IEEE Symposium on Computers and Communications*. IEEE. 2009, pp. 557–563.
- [Alh+19] Fatemah Alharbi et al. "Collaborative client-side DNS cache poisoning attack". In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE. 2019, pp. 1153–1161.
- [Ama] Amazon. *ALEXA INTERNET*. URL: <https://www.alexa.com/> (visited on 09/30/2021).
- [And12] Daniel Anderson. "Splinternet Behind the Great Firewall of China: Once China opened its door to the world, it could not close it again." In: *Queue* 10.11 (2012), pp. 40–49.
- [APN] APNIC. *DNSSEC Validation Rate*. URL: <https://stats.labs.apnic.net/dnssec> (visited on 09/30/2021).

-
- [BB05] Robert Beverly and Steven Bauer. “The Spoofer project: Inferring the extent of source address filtering on the Internet”. In: *Usenix Sruti*. Vol. 5. 2005, pp. 53–59.
- [Ber] Dan J. Bernstein. *DNS Forgery*. URL: <http://cr.yp.to/djbdns/forgery.html> (visited on 09/30/2021).
- [Bir+18] Henry Birge-Lee et al. “Bamboozling certificate authorities with {BGP}”. In: *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 2018, pp. 833–849.
- [Bir+21] Henry Birge-Lee et al. “Experiences deploying multi-vantage-point domain validation at Let’s Encrypt”. In: *30th {USENIX} Security Symposium ({USENIX} Security 21)*. 2021.
- [BKC13] Robert Beverly, Ryan Koga, and KC Claffy. “Initial longitudinal analysis of IP source spoofing capability on the Internet”. In: (2013).
- [Bra+18] M. Brandt et al. “Domain Validation+ + For MitM-Resilient PKI”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. CCS ’18*. 2018.
- [Bui] BuiltWith. *SSL by Default Usage Statistics*. URL: <https://trends.builtwith.com/ssl/SSL-by-Default> (visited on 09/30/2021).
- [CAI] CAIDA. *Spoofers*. URL: <https://www.caida.org/projects/spoofers/> (visited on 09/30/2021).
- [Chu+17a] Taejoong Chung et al. “A longitudinal, end-to-end view of the {DNSSEC} ecosystem”. In: *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 2017, pp. 1307–1322.
- [Chu+17b] Taejoong Chung et al. “Understanding the role of registrars in DNSSEC deployment”. In: *Proceedings of the 2017 Internet Measurement Conference*. 2017, pp. 369–383.
- [CIS] CISA-NCAS. *Alert (TA13-088A) - DNS Amplification Attacks*. URL: <https://www.cisa.gov/uscert/ncas/alerts/TA13-088A> (visited on 09/30/2021).
- [Com] Community. *DNS flag day 2020*. URL: <https://dnsflagday.net/2020/> (visited on 09/30/2021).

-
- [Cor] Gareth Corfield. *DoH! Mozilla assures UK minister that DNS-over-HTTPS won't be default in Firefox for Britons*. URL: https://www.theregister.com/2019/09/24/mozilla_backtracks_doh_for_uk_users/ (visited on 09/30/2021).
- [DAD19] Casey Deccio, Derek Argueta, and Jonathan Demke. "A Quantitative Study of the Deployment of DNS Rate Limiting". In: *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE. 2019, pp. 442–447.
- [Dai+21a] T. Dai et al. "From IP to Transport and beyond: Cross-Layer Attacks against Applications". In: *Proceedings of the 2021 ACM SIGCOMM Conference*. SIGCOMM '21. 2021.
- [Dai+21b] T. Dai et al. "The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources". In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021.
- [Dai16] Tianxiang Dai, Haya Shulman, and Michael Waidner. "DNSSEC Misconfigurations in Popular Domains". In: *Cryptology and Network Security*. Ed. by Sara Foresti and Giuseppe Persiano. Cham: Springer International Publishing, 2016, pp. 651–660. ISBN: 978-3-319-48965-0.
- [Dai18] Markus Brandt et al. "Domain Validation++ For MitM-Resilient PKI". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 2060–2076. ISBN: 9781450356930. DOI: 10.1145/3243734.3243790. URL: <https://doi.org/10.1145/3243734.3243790>.
- [Dai18p] Tianxiang Dai, Haya Shulman, and Michael Waidner. "Poster: Off-Path Attacks Against PKI". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 2213–2215. ISBN: 9781450356930. DOI: 10.1145/3243734.3278516. URL: <https://doi.org/10.1145/3243734.3278516>.
- [Dai21a] Tianxiang Dai, Haya Shulman, and Michael Waidner. "DNS-over-TCP Considered Vulnerable". In: *Proceedings of the Applied Networking Research Workshop*. ANRW '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 76–81. ISBN: 9781450386180. DOI: 10.1145/3472305.3472884. URL: <https://doi.org/10.1145/3472305.3472884>.

-
-
- [Dai21ac] Tianxiang Dai and Haya Shulman. “SMap: Internet-Wide Scanning for Spoofing”. In: *Annual Computer Security Applications Conference*. ACSAC. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 1039–1050. ISBN: 9781450385794. DOI: 10.1145/3485832.3485917. URL: <https://doi.org/10.1145/3485832.3485917>.
- [Dai21ap1] Tianxiang Dai. *DNS-over-TCP considered vulnerable*. URL: <https://blog.apnic.net/2021/11/09/dns-over-tcp-considered-vulnerable/> (visited on 11/30/2021).
- [Dai21c] Tianxiang Dai, Haya Shulman, and Michael Waidner. “Let’s Downgrade Let’s Encrypt”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 1421–1440. ISBN: 9781450384544. DOI: 10.1145/3460120.3484815. URL: <https://doi.org/10.1145/3460120.3484815>.
- [Dai21i1] Tianxiang Dai, Haya Shulman, and Michael Waidner. “Poster: Fragmentation Attacks on DNS over TCP”. In: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. 2021, pp. 1124–1125. DOI: 10.1109/ICDCS51616.2021.00118.
- [Dai21s] Tianxiang Dai et al. “From IP to Transport and beyond: Cross-Layer Attacks against Applications”. In: *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. SIGCOMM ’21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 836–849. ISBN: 9781450383837. DOI: 10.1145/3452296.3472933. URL: <https://doi.org/10.1145/3452296.3472933>.
- [Dai21u] Tianxiang Dai et al. “The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3147–3164. ISBN: 978-1-939133-24-3. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/dai>.
- [Dai22ap1] Tianxiang Dai. *The Hijackers Guide to the Galaxy: Off-path taking over Internet resources*. URL: <https://blog.apnic.net/2022/04/21/off-path-taking-over-internet-resources/> (visited on 04/30/2022).
- [DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The second-generation onion router*. Tech. rep. Naval Research Lab Washington DC, 2004.

-
-
- [DS21] T. Dai and H. Shulman. “SMap: Internet-Wide Scanning for Spoofing”. In: *Annual Computer Security Applications Conference. ACSAC*. 2021.
- [DSW16] T. Dai, H. Shulman, and M. Waidner. “DNSSEC Misconfigurations in Popular Domains”. In: *Cryptology and Network Security. CANS ’16*. 2016.
- [DSW18] T. Dai, H. Shulman, and M. Waidner. “Poster: Off-Path Attacks Against PKI”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. CCS ’18*. 2018.
- [DSW21a] T. Dai, H. Shulman, and M. Waidner. “DNS-over-TCP Considered Vulnerable”. In: *Proceedings of the Applied Networking Research Workshop. ANRW ’21*. 2021.
- [DSW21b] T. Dai, H. Shulman, and M. Waidner. “Let’s Downgrade Let’s Encrypt”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. CCS ’21*. 2021.
- [DSW21c] T. Dai, H. Shulman, and M. Waidner. “Poster: Fragmentation Attacks on DNS over TCP”. In: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. 2021.
- [Enca] Let’s Encrypt. *Mitigating DNS fragmentation attack*. URL: <https://community.letsencrypt.org/t/mitigating-dns-fragmentation-attack/74838> (visited on 09/30/2021).
- [Encb] Let’s Encrypt. *Multi-Perspective Validation Improves Domain Validation Security*. URL: <https://letsencrypt.org/2020/02/19/multi-perspective-validation.html> (visited on 09/30/2021).
- [FSM13] Kensuke Fukuda, Shinta Sato, and Takeshi Mitamura. “A technique for counting dnssec validators”. In: *2013 Proceedings IEEE INFOCOM*. IEEE. 2013, pp. 80–84.
- [FV21] Kazunori Fujiwara and Paul A. Vixie. *Fragmentation Avoidance in DNS*. Internet-Draft draft-ietf-dnsop-avoid-fragmentation-06. Work in Progress. Internet Engineering Task Force, Dec. 2021. 12 pp. URL: <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-avoid-fragmentation-06>.
- [GH11] Yossi Gilad and Amir Herzberg. “Fragmentation considered vulnerable: blindly intercepting and discarding fragments”. In: *Proceedings of the 5th USENIX conference on Offensive technologies*. 2011, pp. 2–2.

-
- [Gooa] Dan Goodin. *Bogus SSL certificate for Windows Live could allow man-in-the-middle hacks*. URL: <https://arstechnica.com/information-technology/2015/03/bogus-ssl-certificate-for-windows-live-could-allow-man-in-the-middle-hacks/> (visited on 09/30/2021).
- [Goob] Google. *Enhancing digital certificate security*. URL: <https://security.googleblog.com/2013/01/enhancing-digital-certificate-security.html> (visited on 09/30/2021).
- [Gooc] Google. *HTTPS encryption on the web*. URL: <https://transparencyreport.google.com/https/overview?hl=en> (visited on 09/30/2021).
- [HS12] Amir Herzberg and Haya Shulman. “Security of patched DNS”. In: *European Symposium on Research in Computer Security*. Springer. 2012, pp. 271–288.
- [HS13a] Amir Herzberg and Haya Shulman. “Fragmentation considered poisonous, or: One-domain-to-rule-them-all. org”. In: *2013 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2013, pp. 224–232.
- [HS13b] Amir Herzberg and Haya Shulman. “Socket overloading for fun and cache-poisoning”. In: *Proceedings of the 29th Annual Computer Security Applications Conference*. 2013, pp. 189–198.
- [HS13c] Amir Herzberg and Haya Shulman. “Vulnerable delegation of DNS resolution”. In: *European Symposium on Research in Computer Security*. Springer. 2013, pp. 219–236.
- [HS14] Amir Herzberg and Haya Shulman. “Retrofitting security into network protocols: The case of dnssec”. In: *IEEE Internet Computing* 18.1 (2014), pp. 66–71.
- [Hu15] Margaret Hu. “Taxonomy of the snowden disclosures”. In: *Wash. & Lee L. Rev.* 72 (2015), p. 1679.
- [Huz+15] Gokay Huz et al. “Experience in using mturk for network measurement”. In: *Proceedings of the 2015 ACM SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data*. 2015, pp. 27–32.
- [JD20] Baptiste Jonglez and Andrzej Duda. “Don’t Forget to Lock the Front Door! Inferring the Deployment of Source Address Validation of Inbound Traffic”. In: *Passive and Active Measurement: 21st International Conference, PAM 2020, Eugene, Oregon, USA, March 30–31, 2020, Proceedings*. Vol. 12048. Springer Nature. 2020, p. 107.

-
-
- [JSW20] Philipp Jeitner, Haya Shulman, and Michael Waidner. “The impact of dns insecurity on time”. In: *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE. 2020, pp. 266–277.
- [Kam08] Dan Kaminsky. “Black ops 2008: It’s the end of the cache as we know it”. In: *Black Hat USA 2 (2008)*.
- [Kle07a] Amit Klein. “BIND 9 DNS cache poisoning”. In: *Report, Trusteer, Ltd 3 (2007)*.
- [Kle07b] Amit Klein. *Windows DNS Server Cache Poisoning*. 2007.
- [Kle21] Amit Klein. “Cross layer attacks and how to use them (for dns cache poisoning, device tracking and more)”. In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 1179–1196.
- [KM95] Christopher A Kent and Jeffrey C Mogul. “Fragmentation considered harmful”. In: *ACM SIGCOMM Computer Communication Review 25.1 (1995)*, pp. 75–87.
- [KN] Maciej Korczyński and Yevheniya Nosyk. *The Closed Resolver Project*. URL: <https://closedresolver.korlabs.io/> (visited on 09/30/2021).
- [Kor+20a] Maciej Korczyński et al. “Inferring the Deployment of Inbound Source Address Validation Using DNS Resolvers”. In: *Proceedings of the Applied Networking Research Workshop. ANRW ’20*. Association for Computing Machinery, 2020, pp. 9–11. DOI: 10.1145/3404868.3406668.
- [Kor+20b] Maciej Korczyński et al. “The Closed Resolver Project: Measuring the Deployment of Source Address Validation of Inbound Traffic”. In: *arXiv preprint arXiv:2006.05277 (2020)*.
- [KPS03] Charlie Kaufman, Radia Perlman, and Bill Sommerfeld. “DoS protection for UDP-based protocols”. In: *Proceedings of the 10th ACM conference on Computer and communications security*. 2003, pp. 2–7.
- [Küh+14] Marc Kühler et al. “Exit from Hell? Reducing the Impact of {Amplification}{DDoS} Attacks”. In: *23rd USENIX Security Symposium (USENIX Security 14)*. 2014, pp. 111–125.
- [Lau14] Ben Laurie. “Certificate transparency”. In: *Communications of the ACM 57.10 (2014)*, pp. 40–46.
- [Li+19] Bingyu Li et al. “Certificate transparency in the wild: Exploring the reliability of monitors”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 2505–2520.

-
-
- [Lia+13] Wilson Lian et al. “Measuring the Practical Impact of {DNSSEC} Deployment”. In: *22nd {USENIX} Security Symposium ({USENIX} Security 13)*. 2013, pp. 573–588.
- [Lic+17] Franziska Lichtblau et al. “Detection, classification, and analysis of inter-domain traffic with spoofed source IP addresses”. In: *Proceedings of the 2017 Internet Measurement Conference*. 2017, pp. 86–99.
- [Lon+17] Qasim Lone et al. “Using loops observed in traceroute to infer the ability to spoof”. In: *International Conference on Passive and Active Network Measurement*. Springer. 2017, pp. 229–241.
- [Lon+18] Qasim Lone et al. “Using crowdsourcing marketplaces for network measurements: The case of spoofer”. In: *2018 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE. 2018, pp. 1–8.
- [Luc+19] Matthew Luckie et al. “Network hygiene, incentives, and regulation: deployment of source address validation in the Internet”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 465–480.
- [Mal+16] Aanchal Malhotra et al. “Attacking the Network Time Protocol”. In: *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.
- [Man+20] Keyu Man et al. “DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 1337–1350.
- [Moza] Mozilla. *Percentage of Web Pages Loaded by Firefox Using HTTPS*. URL: <https://letsencrypt.org/stats/#percent-pageloads> (visited on 09/30/2021).
- [Mozb] Mozilla. *Symantec Issues*. URL: https://wiki.mozilla.org/CA/Symantec_Issues (visited on 09/30/2021).
- [Mozc] Mozilla. *WoSign Issues*. URL: https://wiki.mozilla.org/CA/WoSign_Issues (visited on 09/30/2021).
- [MSK15] Douglas C MacFarland, Craig A Shue, and Andrew J Kalafut. “Characterizing optimal DNS amplification attacks and effective mitigation”. In: *International Conference on Passive and Active Network Measurement*. Springer. 2015, pp. 15–27.

-
-
- [MSK17] Douglas C MacFarland, Craig A Shue, and Andrew J Kalafut. “The best bang for the byte: Characterizing the potential of DNS amplification attacks”. In: *Computer Networks* 116 (2017), pp. 12–21.
- [OAR] OARC. *OARC’s DNS Reply Size Test Server*. URL: <https://www.dns-oarc.net/oarc/services/replysizetest> (visited on 09/30/2021).
- [Par+04] KyoungSoo Park et al. “CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups.” In: *OSDI*. Vol. 4. 2004, pp. 14–14.
- [PP06] Lindsey Poole and Vivek S Pai. “ConfiDNS: Leveraging Scale and History to Improve DNS Security.” In: *WORLDS*. 2006.
- [RFC1034] P.V. Mockapetris. *Domain names - concepts and facilities*. RFC 1034. IETF, Nov. 1987. URL: <http://tools.ietf.org/rfc/rfc1034.txt>.
- [RFC1035] P.V. Mockapetris. *Domain names - implementation and specification*. RFC 1035. IETF, Nov. 1987. URL: <http://tools.ietf.org/rfc/rfc1035.txt>.
- [RFC1122] R. Braden. *Requirements for Internet Hosts - Communication Layers*. RFC 1122. IETF, Oct. 1989. URL: <http://tools.ietf.org/rfc/rfc1122.txt>.
- [RFC1123] R. Braden. *Requirements for Internet Hosts - Application and Support*. RFC 1123. IETF, Oct. 1989. URL: <http://tools.ietf.org/rfc/rfc1123.txt>.
- [RFC1945] T. Berners-Lee, R. Fielding, and H. Frystyk. *Hypertext Transfer Protocol – HTTP/1.0*. RFC 1945. IETF, May 1996. URL: <http://tools.ietf.org/rfc/rfc1945.txt>.
- [RFC2827] P. Ferguson and D. Senie. *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. RFC 2827. IETF, May 2000. URL: <http://tools.ietf.org/rfc/rfc2827.txt>.
- [RFC3704] F. Baker and P. Savola. *Ingress Filtering for Multihomed Networks*. RFC 3704. IETF, Mar. 2004. URL: <http://tools.ietf.org/rfc/rfc3704.txt>.
- [RFC4033] R. Arends et al. *DNS Security Introduction and Requirements*. RFC 4033. IETF, Mar. 2005. URL: <http://tools.ietf.org/rfc/rfc4033.txt>.
- [RFC4034] R. Arends et al. *Resource Records for the DNS Security Extensions*. RFC 4034. IETF, Mar. 2005. URL: <http://tools.ietf.org/rfc/rfc4034.txt>.
- [RFC4035] R. Arends et al. *Protocol Modifications for the DNS Security Extensions*. RFC 4035. IETF, Mar. 2005. URL: <http://tools.ietf.org/rfc/rfc4035.txt>.

-
-
- [RFC5452] A. Hubert and R. van Mook. *Measures for Making DNS More Resilient against Forged Answers*. RFC 5452. IETF, Jan. 2009. URL: <http://tools.ietf.org/rfc/rfc5452.txt>.
- [RFC768] J. Postel. *User Datagram Protocol*. RFC 768. IETF, Aug. 1980. URL: <http://tools.ietf.org/rfc/rfc0768.txt>.
- [RFC7766] J. Dickinson et al. *DNS Transport over TCP - Implementation Requirements*. RFC 7766. IETF, Mar. 2016. URL: <http://tools.ietf.org/rfc/rfc7766.txt>.
- [RFC7858] Z. Hu et al. *Specification for DNS over Transport Layer Security (TLS)*. RFC 7858. IETF, May 2016. URL: <http://tools.ietf.org/rfc/rfc7858.txt>.
- [RFC791] J. Postel. *Internet Protocol*. RFC 791. IETF, Sept. 1981. URL: <http://tools.ietf.org/rfc/rfc0791.txt>.
- [RFC792] J. Postel. *Internet Control Message Protocol*. RFC 792. IETF, Sept. 1981. URL: <http://tools.ietf.org/rfc/rfc0792.txt>.
- [RFC793] J. Postel. *Transmission Control Protocol*. RFC 793. IETF, Sept. 1981. URL: <http://tools.ietf.org/rfc/rfc0793.txt>.
- [RFC815] D.D. Clark. *IP datagram reassembly algorithms*. RFC 815. IETF, July 1982. URL: <http://tools.ietf.org/rfc/rfc0815.txt>.
- [RFC8446] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. IETF, Aug. 2018. URL: <http://tools.ietf.org/rfc/rfc8446.txt>.
- [RFC8484] P. Hoffman and P. McManus. *DNS Queries over HTTPS (DoH)*. RFC 8484. IETF, Oct. 2018. URL: <http://tools.ietf.org/rfc/rfc8484.txt>.
- [RFC8900] R. Bonica et al. *IP Fragmentation Considered Fragile*. RFC 8900. IETF, Sept. 2020. URL: <http://tools.ietf.org/rfc/rfc8900.txt>.
- [RMK13] Thijs Rozebrans, Matthijs Mekking, and Javy de Koning. "Defending against DNS reflection amplification attacks". In: *University of Amsterdam System & Network Engineering RP1* (2013).
- [Ros14] Christian Rossow. "Amplification Hell: Revisiting Network Protocols for DDoS Abuse". In: *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.

-
- [S G18] S. Goldberg. *The myetherwallet.com hijack and why it's risky to hold cryptocurrency in a webapp*. Accessed: 2021-1-19. 2018. URL: <https://medium.com/@goldbe/the-myetherwallet-com-hijack-and-why-its-risky-to-hold-cryptocurrency-in-a-webapp-261131fad278>.
- [San] Sandia. *DNSViz*. URL: <https://dnsviz.net/> (visited on 09/30/2021).
- [Sta] Internet World Stats. *World Internet Users Statistics*. URL: <https://internetworldstats.com/stats.htm> (visited on 09/30/2021).
- [Ste03] Joe Stewart. *DNS cache poisoning-the next generation*. 2003.
- [SW14] Haya Shulman and Michael Waidner. "Fragmentation considered leaking: port inference for dns poisoning". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2014, pp. 531–548.
- [SW15] Haya Shulman and Michael Waidner. "Towards security of internet naming infrastructure". In: *European Symposium on Research in Computer Security*. Springer. 2015, pp. 3–22.
- [SW17] Haya Shulman and Michael Waidner. "One key to sign them all considered vulnerable: Evaluation of {DNSSEC} in the internet". In: *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 2017, pp. 131–144.
- [Uni] George Mason University. *SecSpider*. URL: <http://secspider.net/> (visited on 09/30/2021).
- [Ver] VeriSign. *DNSSEC Analyzer*. URL: <https://dnssec-analyzer.verisignlabs.com/> (visited on 09/30/2021).
- [Vix95] Paul Vixie. "DNS and BIND Security Issues". In: *Proceedings of the 5th USENIX Security Symposium, Salt Lake City, Utah, USA, June 5-7, 1995*. Ed. by Frederick M. Avolio and Steven M. Bellovin. USENIX Association, 1995.
- [VS12] Paul Vixie and Vernon Schryver. *DNS Response Rate Limiting (DNS RRL)*. TN 2012-1. ISC, 2012. URL: <https://www.isc.org/pubs/tn/isc-tn-2012-1.txt>.
- [WAP08] D Wendlandt, D Andersen, and A Perrigo Perspectives. "Improving SSH-style Host Authentication with Multi-path Network Probing". In: *USENIX Annual Technical Conference*. 2008.
- [Wik] Wikipedia. *DigiNotar*. URL: <https://en.wikipedia.org/wiki/DigiNotar> (visited on 09/30/2021).

-
- [Yan+10] Hao Yang et al. “Deploying cryptography in Internet-scale systems: A case study on DNSSEC”. In: *IEEE Transactions on Dependable and Secure Computing* 8.5 (2010), pp. 656–669.
- [Zal] Michal Zalewski. *A new TCP/IP blind data injection technique?* URL: <https://bugtraq.securityfocus.narkive.com/rm25I7e1/a-new-tcp-ip-blind-data-injection-technique> (visited on 09/30/2021).
- [Zhe+20] Xiaofeng Zheng et al. “Poison over troubled forwarders: A cache poisoning attack targeting {DNS} forwarding devices”. In: *29th USENIX Security Symposium (USENIX Security 20)*. 2020, pp. 577–593.
- [Zhu+15] Liang Zhu et al. “Connection-oriented DNS to improve privacy and security”. In: *2015 IEEE symposium on security and privacy*. IEEE. 2015, pp. 171–186.

A. Papers and Posters

A.1. DNSSEC Misconfigurations in Popular Domains

[Dai16]

Tianxiang Dai, Haya Shulman, and Michael Waidner. “DNSSEC Misconfigurations in Popular Domains”. In: *Cryptology and Network Security*. Ed. by Sara Foresti and Giuseppe Persiano. Cham: Springer International Publishing, 2016, pp. 651–660. ISBN: 978-3-319-48965-0

DNSSEC Misconfigurations in Popular Domains

Tianxiang Dai^(✉), Haya Shulman, and Michael Waidner

Fraunhofer Institute for Secure Information Technology SIT, Darmstadt, Germany
{tianxiang.dai,haya.shulman,michael.waidner}@sit.fraunhofer.de

Abstract. DNSSEC was designed to protect the Domain Name System (DNS) against DNS cache poisoning and domain hijacking. When widely adopted, DNSSEC is expected to facilitate a multitude of future applications and systems, as well as security mechanisms, that would use the DNS for distribution of security tokens, such as, certificates, IP prefix authentication for routing security, anti-spam mechanisms. Multiple efforts are invested in adopting DNSSEC and in evaluating challenges towards its deployment.

In this work we perform a study of errors and misconfigurations in signed domains. To that end, we develop a DNSSEC framework and a webpage for reporting the most up to date statistics and provide reports with vulnerabilities and misconfigurations. Our tool also supports retrieval of historical data and enables to perform long-term studies and observations of changes in the security landscape of DNS. We make our tool and the collected data available via an online webservice.

1 Introduction

Domain Name System (DNS), [RFC1034, RFC1035], has a key role in the Internet. The correctness and availability of DNS are critical to the security and functionality of the Internet. Initially designed to translate domain names to IP addresses, the DNS infrastructure has evolved into a complex ecosystem, and the complexity of the DNS infrastructure is continuously growing with the increasing range of purposes and client base. DNS is increasingly utilised to facilitate a wide range of applications and constitutes an important building block in the design of scalable network infrastructures.

There is a long history of attacks against DNS, most notably, DNS cache poisoning, [5–7, 12, 14, 17]. DNS cache poisoning attacks are known to be practiced by governments, e.g., for censorship [1] or for surveillance [11], as well as by cyber criminals. In the course of a DNS cache poisoning attack, the attacker provides spoofed records in DNS responses, in order to redirect the victims to incorrect hosts for credential theft, malware distribution, censorship and more.

To mitigate the threat from the DNS cache poisoning attacks, the IETF designed and standardised Domain Name System Security Extensions (DNSSEC) [RFC4033-RFC4035]. Unfortunately DNSSEC requires significant changes to the DNS infrastructure as well as to the protocol, and although proposed and standardised already in 1997, it is still not widely deployed. Studies show that less than 1% of the domains are signed with DNSSEC, [9, 19] and

about 3% of the DNS resolvers validate DNSSEC records, [3, 13]. However, the situation is improving and following the recent ICANN regulation, [15], the registrars are turning domain signing into an automated task, as the procedures for automated domain signing by the registrars and hosting providers are becoming widely supported. Now that the DNSSEC is taking off, tools for evaluating problems with signed domains are critical, since they can alert the domain owners as well as clients of the potential pitfalls. Although tools for studying DNSSEC exist, and we compare them with our tool in Related Work, Sect. 2, our tool detects and reports misconfigurations and cryptographic vulnerabilities which were not performed prior to our work.

In this work we perform a study of miconfigurations among DNSSEC-signed domains. We first collect a list of popular signed domains, and then measure the different misconfigurations and problems among them. We provide access to our tool through a webpage, which can be accessed at: <https://dnssec.cad.sit.fraunhofer.de>.

Contributions. We designed and implemented a framework, *DNSSEC misconfiguration validation engine*, which collects signed domains from multiple sources, analyses the misconfigurations among them, and processes them into reports. Our reports quantify two types of vulnerabilities in signed domains: *cryptographic failures* (those preventing a DNS resolver from establishing a chain of trust or domains using vulnerable DNSSEC keys) and *transport failures* (e.g., lack of support of TCP or EDNS). We use our engine to perform Internet-wide collection of 1349 Top-Level Domains (TLDs) and top-1M Alexa (www.alexacom) domains.

We collected statistics between March and September 2016 with our tool, and report on the current status as well as improvements that we detected over time. Our study indicates that 90% of TLDs and 1.66% of Alexa domains are signed. Among signed domains, 0.89% TLDs and 19.46% Alexa domains cannot establish a chain of trust to the root zone; among those Alexa domains, 85.5% are Second-Level Domains (SLDs). We also checked for the presence of DNSSEC keys in domains with a broken chain of trust, in other repositories for DNSSEC keys distribution. Of the 19.46% of the Alexa domains, only 51 have a DLV resource record in dlv.isc.org. Namely, majority of the signed domains do not provide any benefit by signing their records, since the clients anyway cannot validate the signatures. We find domains with vulnerable DNSSEC keys, using even RSA modulus. In contrast to February 2016, where 3% of TLDs did not have support for TCP, all TLDs currently support TCP. However, 12.88% of Alexa domains have nameservers which still cannot serve DNS responses over TCP.

The reports and statistics can be accessed at <https://dnssec.cad.sit.fraunhofer.de>.

Organisation. In Sect. 2 we compare our research to related work. In Sect. 3 we describe our DNSSEC configuration validation engine, its components and the data collection that we performed with. In Sect. 4 we perform a measurement of

signed domains and characterise causes for the misconfigured signed domains. We conclude this work in Sect. 5.

2 Related Work

The research and operational communities invested significant efforts in generating online services for studying DNS. We review some of the central services.

OARC's DNS Reply Size Test Server is an online service for testing responses size of DNS. The clients can use the tool to evaluate the maximum response size that their network can support. This test is especially critical for adoption of DNSSEC, since DNSSEC enabled responses typically exceed the standard size of 512 bytes.

Multiple online services were designed for evaluating the security of port selection algorithms, most notably `porttest.dns-oarc.net`; see survey and analysis in [6]. The tools study the randomness in ports selected by the DNS resolver.

Recently multiple tools were proposed for checking DNSSEC adoption on zones. For instance, `DNSViz`, given a domain name, visualises all the keys the domain has and signatures over DNS records. It also checks that it is possible to establish a chain of trust from the root to the target domain. `SecSpider` provides overall statistics for DNSSEC deployment on zones, by collecting signed DNS records and keys from the zones.

Our tool complements the existing tools by allowing to study insecurity or misconfigurations on a given domain, as well as analysing statistics of the misconfigurations over a given time period, and for a set of domains. In contrast to existing tools which provide an analysis for a given domain that they receive in an input, our tool is invoked periodically over the datasets that it uses, analyses the data and produces reports with statistics. The reports contain misconfigurations on the transport layer, such as support of TCP, as well as on the cryptographic aspects, such as vulnerable keys and lack of chain of trust. Our tool provides important insights to clients accessing domains as well as for domain owners, and allows researchers to study changes in security and configurations of domains over time.

Prior studies measuring adoption of DNSSEC, investigated validation on the DNS resolvers' side, [13], showing that a large fraction of DNS resolvers do not perform correct validation of DNSSEC signatures. Other works investigated obstacles towards adoption of DNSSEC, suggesting mitigations and alternative mechanisms, [8–10].

Our tool provides insights on the status of adoption of DNSSEC among zones and on misconfigurations within signed domains in DNS hierarchy, as well as on the failures on nameservers, such as failures to serve responses over TCP.

3 DNSSEC Adoption/Configuration Framework

In this section we present our framework for collecting and processing domains, illustrated in Fig. 1. In the rest of this section we explain the components of our

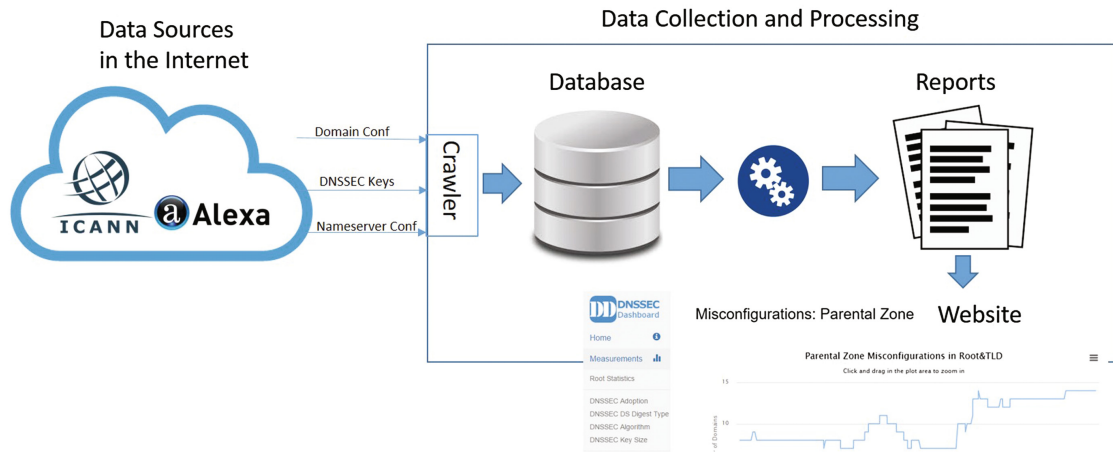


Fig. 1. DNSSEC adoption and configuration evaluation framework.

DNSSEC validation engine, including data sources and data collection, and the analysis of the data and processing into reports and online web page.

Domains Crawler. We developed a crawler to collect and store DNSSEC-signed domains.

Data Sources. We collected sources of DNSSEC signed zones that we feed to the database as ‘crawling seeds’:

- (1) the root and Top Level Domain (TLD) zone files – we obtained the root and TLD zone files (e.g., for `com`, `net`, `org`, `info`) from the Internet Corporation for Assigned Names and Numbers (ICANN). In total we study 1301 TLDs.
- (2) we scanned the top-1M popular domains according to Alexa www.alexa.com.

4 Evaluating Vulnerabilities in DNSSEC Adoption

In this section we provide our measurement of adoption of DNSSEC among the domains in our dataset, i.e., the Top Level Domains (TLDs) and Second Level Domains (SLDs) (based on the data sources in Sect. 3), and report on misconfigurations and vulnerabilities.

Quantifying Signed Domains. We define DNSSEC-signed domains as those with `DNSKEY` and `RRSIG` records. To check for the fraction of signed domains, we checked for existence of `DNSKEY` and `RRSIG` records in our dataset. Our results show that 90% of the TLDs and 1.66% of the SLDs are signed.

In Fig. 2 we plot the results we collected between March and September 2016. The upper line indicates the total number of TLDs/SLDs, while the lower line indicates the number of DNSSEC-signed TLDs/SLDs. In that time interval the number of new TLDs increased by 250 and we observe roughly the same increase in the number of signed TLDs. The graph also shows a growth in a number of new

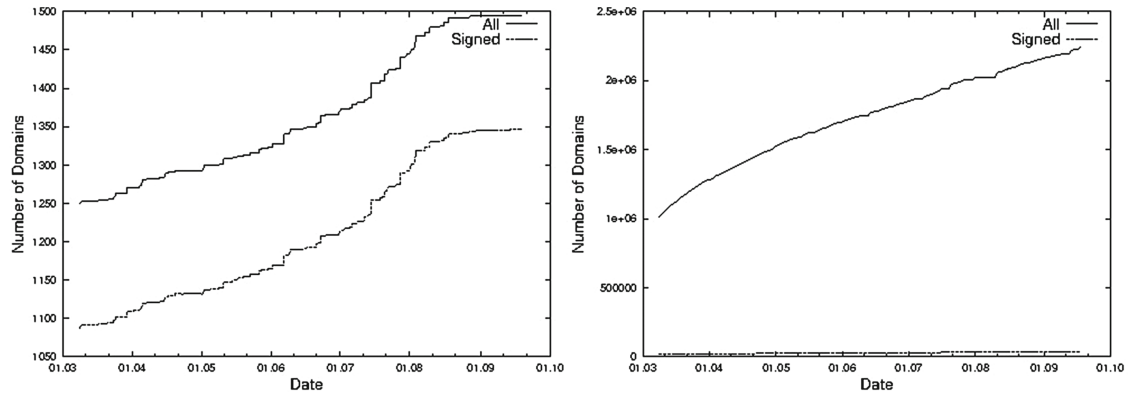


Fig. 2. All TLDs vs. signed TLDs (left). All SLDs vs. signed SLDs (right).

SLDs. However, in contrast to the steady increase in signed TLDs, the results indicate a negligible increase in newly signed SLDs. The significant and constant growth in the number of signed TLDs indicates that there is an increased awareness to DNSSEC adoption. One of the main reason for lack of increase in SLDs is that many registrars still do not support automated procedures for DNSSEC.

Crypto-Algorithms in Signed Domains. The signed zones can use an arbitrary number of DNSSEC-standardised algorithms¹. In addition, [RFC4641,RFC6781] list mandatory support for RSA and recommend avoiding large keys (specifying a range of 512–2048 bits for (ZSK) key size and recommending a default value of 1024 bits); in order to avoid fragmentation, communication and computation overhead and other problems with large keys and signatures. In particular, [RFC6781] states “it is estimated that most zones can safely use 1024-bit keys for at least the next ten years”.

We analysed our dataset of signed domains, and plot the results in Fig. 3. For TLDs, the upper two lines are RSA-SHA256 and RSA-SHA1-NSEC3 correspondingly. The two lines in the bottom are RSA-SHA512 and RSA-SHA1. For SLDs, the upper four lines correspond to RSA-SHA256, RSA-SHA1-NSEC3, RSA-SHA1 and ECDSA-P256-SHA256. DSA, RSA-SHA512 and ECDSA-P384-SHA384 are in the bottom.

Our measurement shows that there is hardly any support for other cryptographic algorithms, e.g., those that produce short signatures, such as ECC, since the motivation to add more overhead to the transmitted data is low. Indeed, most domains adopt different versions of RSA, which produces larger keys and signatures.

RSA, with different digest implementations (SHA1, SHA256, SHA512), dominates among the signed TLDs, and there is no support for other algorithms among the TLDs, Fig. 4. In contrast, there is some, albeit still limited, attempt to adopt also other cryptographic algorithms, such as DSA and EC in SLDs, see Fig. 3. Indeed, ECDSA-P256 is ranked third among the cryptographic

¹ <http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml>.

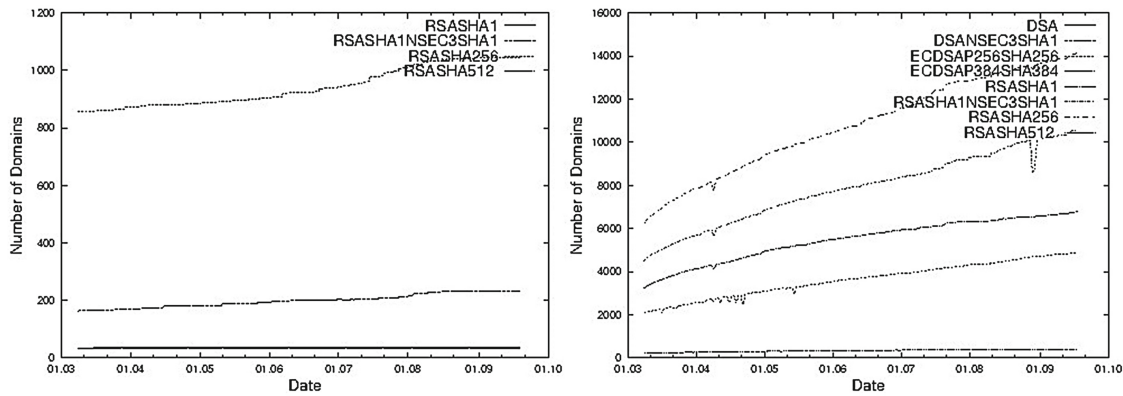


Fig. 3. DNSSEC algorithms adoption between March-September 2016 in signed TLDs (left) and signed SLDs (right).

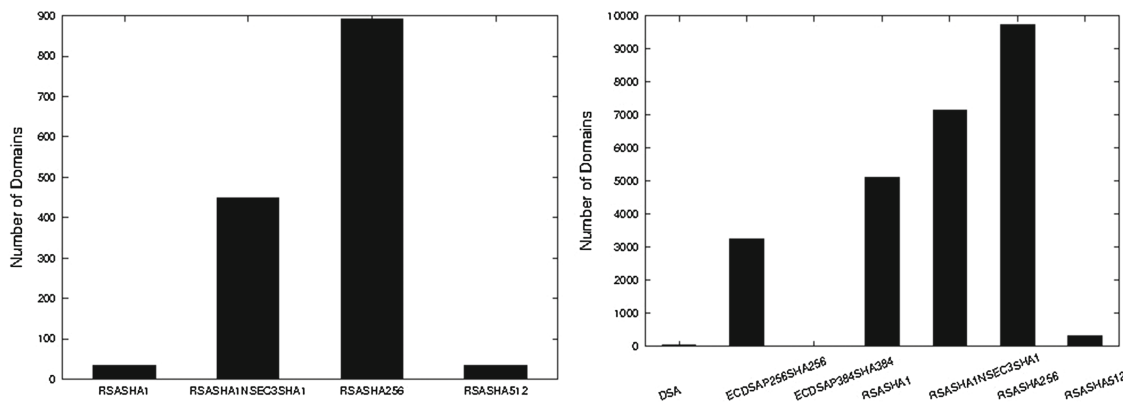


Fig. 4. DNSSEC algorithms in signed TLDs (left), in signed SLDs (right).

algorithms, just behind RSA-SHA1 (including RSA-SHA1 and RSA-SHA1-NSEC3) and RSA-SHA256. ECDSA-P256 is gaining more popularity and grows steadily. This also shows that more and more admins are adopting new algorithms to improve DNSSEC performance.

We measured the key sizes in use by the different variations of RSA algorithms, we plot our results in Fig. 5 on the right. It’s a CDF of key size in signed domains. We can see about 34 % of TLDs and 52 % of SLDs are still using keys shorter than or equal to 1024 bits. As for keys only, almost 1.4M keys are below 1024 bits, and 10 K keys are 512 bits long. These are really vulnerable. [18] showed that factoring 512 bit keys on a cloud is a practical task. For updated statistics on keys and DNSSEC algorithms see our webpage.

We also checked the digest algorithms used in DNSSEC. There are mainly three digest algorithms employed by DNSSEC, SHA1, SHA256 and SHA512. SHA1 has been known to be considerably weak. [16] showed a collision attack against SHA1. Google also announced that they would completely block SHA1 certificates in 2017 [2]. Digests are used in three ways in DNSSEC. First, in digital signature RRSIG along with RSA or ECC. Second, for authenticated proof of the non-existence, in NSEC3. Third, as anchor for Key Signing Key, in DS.

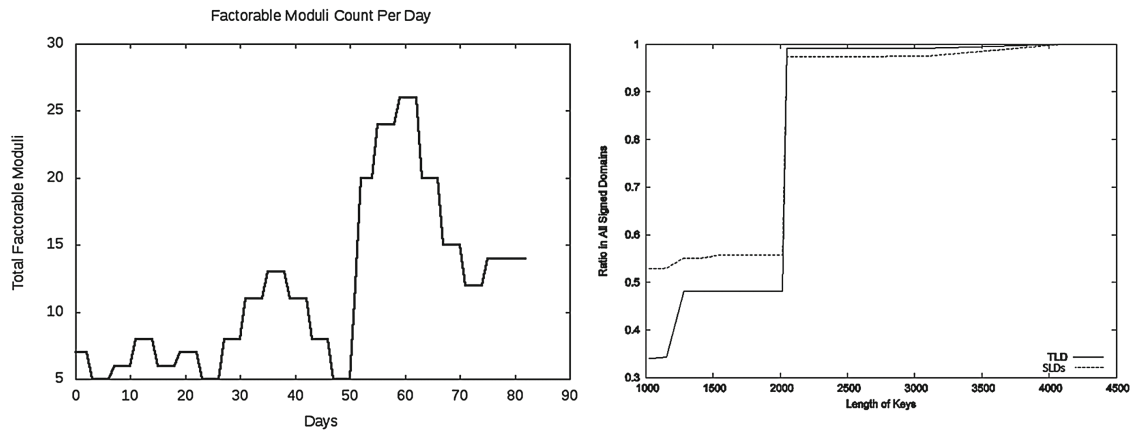


Fig. 5. Keys with even RSA moduli (left) and key sizes in TLDs and SLDs (right).

For digests in signature RRSIG as can be seen in Fig. 3 on the left for TLDs, SHA256 is still the most popular and grows faster than the others. The growth in adoption of SHA1 slows down. And there’s almost no increase in SHA512. This indicates that there is an increased awareness to sunset SHA1 and promote SHA256, while SHA512 is still not essential. When we look at the SLDs on the right, SHA1 (including RSA-SHA1 and RSA-SHA1-NSEC3, 2nd and 3rd lines) has almost the same share as SHA256 (including RSA-SHA256 and ECDSA-P256, 1st and 4th lines). The good point is that SHA256 is growing faster than SHA1. But it still needs time to move from SHA1 to SHA256.

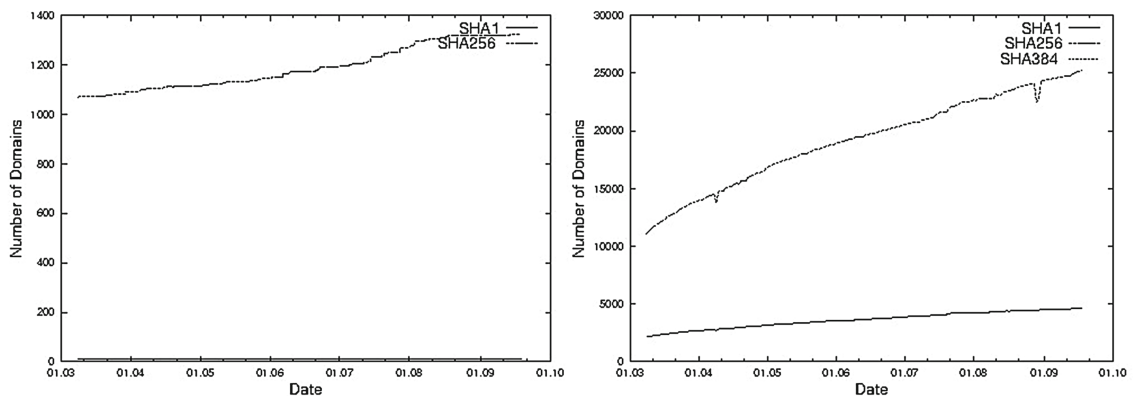


Fig. 6. DNSSEC DS digest algorithms between March-September 2016 in signed TLDs (left), in signed SLDs (right).

For digests in DS this is even more important, since a DS RR is the entry point of a zone. As can be seen in Fig. 6, SHA256 overwhelms SHA1 in TLDs. Among SLDs, number of domains using DS with SHA256 grows much faster than that using SHA1 only. This indicates the increased awareness of vulnerability of SHA1.

Broken Chain of Trust. Finally we evaluate whether the DNS resolvers can establish a chain of trust from the root to the signed domains (i.e., those

with DNSKEY and RRSIG records). We perform this measurement for TLDs and SLDs and report the results in Fig. 7. We use the terminology of [RFC3090], where *locally signed* means that a chain of trust cannot be established from the root (and the keys are also not present in external repositories, such as DLV dlv.isc.org). The problems include wrong (or missing) DS records in parent domain, incorrect (or missing) signatures, expired keys, DNSKEY and DS do not match and more. There are 0.89 % of domains among TLDs and 19.46 % among the SLDs to which we could not establish a chain of trust from the root, nor could we locate their keys in DNSKEY repositories.

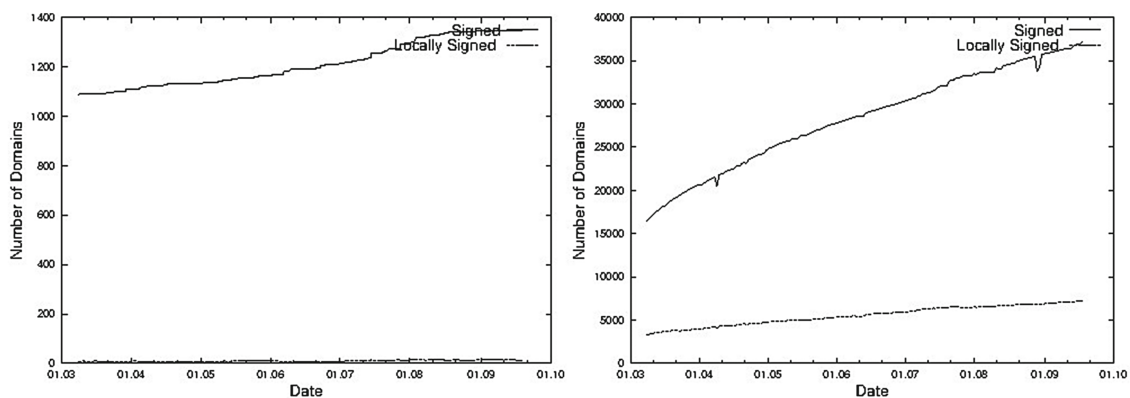


Fig. 7. TLDs with broken chain of trust vs. secure (left). SLDs with broken chain of trust vs. secure (right).

In both domain types there is an increase in the number of signed domains that cannot be validated. The increase is aligned with the increase in newly signed domains.

We checked for the factors behind the large fraction of signed domains with a broken chain of trust. The most common case of broken chain of trust is an existence of DNSKEY but no DS in parent. This may happen when a domain owner wants to enable DNSSEC but his registrar does not support DNSSEC, which is common. Alternately, the same obstacle occurs when the registrar does not support DNSSEC for a TLD under which the domain is registered, e.g., GoDaddy supports DNSSEC only for 10 TLDs. Other common cause is a faulty DS record. This may happen when the domain operator transfers/updates the domain/key or changes the name servers.

To fix these problems, it is recommended to move domains to the registrars that support DNSSEC. If the TLD is not supported by the registrar, the DLV service should be utilised. To track for misconfigurations, we provide our tool for a public use, which can be accessed at: <https://dnssec.cad.sit.fraunhofer.de>.

RSA Keys with Even Moduli. Distinct moduli that share a prime factor will result in public keys that appear different but whose private keys are efficiently computable by calculating the greatest common divisor (GCD). For calculation of GCD of every pair of keys we followed the approach in [4] and used the *fast*

pairwise GCD quasilinear-time algorithm for factoring a collection of integers into coprimes; we compiled and used the source code (<https://factorable.net/resources.html>) provided by [4].

After calculating group-GCD on all the DNSKEY records, we found 16 even RSA moduli.

The keys with even RSA moduli belonged to domains hosted or registered by known registrars, such as Network Solutions, GoDaddy, OnlineNic. In Fig. 5 we plot our measurements of factorable RSA keys, collected over a period of March-September 2016.

5 Conclusion

In this work we measured adoption of DNSSEC among TLDs and SLDs, and then studied the security of the signed domains. To that end, we designed and developed a tool that periodically collects data from signed domains, analyses it and produces reports with statistics. Our data collection indicates that a large fraction of signed domains have cryptographic misconfigurations, leading to insecurity. The misconfigurations are either due to a broken chain of trust, preventing the DNS resolver from validating the supplied DNS records, or due to vulnerable cryptographic keys.

We developed an online service for providing updated reports and statistics on adoption of DNSSEC, vulnerabilities and misconfigurations: <https://dnssec.cad.sit.fraunhofer.de>.

Acknowledgments. The research reported in this paper has been supported by the German Federal Ministry of Education and Research (BMBF) and by the Hessian Ministry of Science and the Arts within CRISP www.crisp-da.de/.

References

1. Anderson, D.: Splinternet behind the great firewall of china. *Queue* **10**(11), 40 (2012)
2. Google Online Security Blog: An Update on SHA-1 Certificates in Chrome (2015). <https://security.googleblog.com/2015/12/an-update-on-sha-1-certificates-in.html>
3. Fukuda, K., Sato, S., Mitamura, T.: A technique for counting DNSSEC validators. In: 2013 Proceedings IEEE INFOCOM, pp. 80–84. IEEE (2013)
4. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your PS, QS: detection of widespread weak keys in network devices. In: Presented as part of the 21st USENIX Security Symposium (USENIX Security 12), pp. 205–220 (2012)
5. Herzberg, A., Shulman, H.: Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In: The Conference on Communications and Network Security IEEE CNS 2013, Washington, D.C., U.S. IEEE (2013)
6. Herzberg, A., Shulman, H.: Socket overloading for fun and cache poisoning. In: C.N.P. Jr. (ed.) ACM Annual Computer Security Applications Conference (ACM ACSAC), New Orleans, Louisiana, U.S, December 2013

7. Herzberg, A., Shulman, H.: Vulnerable delegation of DNS resolution. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 219–236. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-40203-6_13](https://doi.org/10.1007/978-3-642-40203-6_13)
8. Herzberg, A., Shulman, H.: Negotiating DNSSEC algorithms over legacy proxies. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 111–126. Springer, Heidelberg (2014). doi:[10.1007/978-3-319-12280-9_8](https://doi.org/10.1007/978-3-319-12280-9_8)
9. Herzberg, A., Shulman, H.: Retrofitting security into network protocols: the case of DNSSEC. *Internet Comput.* **18**(1), 66–71 (2014). IEEE
10. Herzberg, A., Shulman, H., Crispo, B.: Less is more: cipher-suite negotiation for DNSSEC. In: Computer Security Applications Conference, ACSAC 2014. Annual. IEEE (2014)
11. Hu, M.: Taxonomy of the snowden disclosures. *Wash Lee L. Rev.* **72**, 1679–1989 (2015)
12. Kaminsky, D.: It's the End of the Cache As We Know It. In Black Hat conference, August 2008. <http://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf>
13. Lian, W., Rescorla, E., Shacham, H., Savage, S.: Measuring the practical impact of DNSSEC deployment. In: Proceedings of USENIX Security (2013)
14. Shulman, H., Waidner, M.: Fragmentation considered leaking: port inference for DNS poisoning. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 531–548. Springer, Heidelberg (2014). doi:[10.1007/978-3-319-07536-5_31](https://doi.org/10.1007/978-3-319-07536-5_31)
15. Internet Society: ICANNs 2013 RAA Requires Domain Name Registrars To Support DNSSEC (2013)
16. Stevens, M., Karpman, P., Peyrin, T.: Freestart collision for full sha-1. *Cryptology ePrint Archive, Report 2015/967* (2015). <http://eprint.iacr.org/2015/967>
17. Stewart, J.: DNS cache poisoning—the next generation (2003)
18. Valenta, L., Cohny, S., Liao, A., Fried, J., Bodduluri, S., Heninger, N.: Factoring as a service
19. Yang, H., Osterweil, E., Massey, D., Lu, S., Zhang, L.: Deploying cryptography in internet-scale systems: a case study on DNSSEC. *IEEE Trans. Dependable Secur. Comput.* **8**(5), 656–669 (2011)

A.2. Domain Validation++ For MitM-Resilient PKI

[Dai18]

Markus Brandt et al. "Domain Validation++ For MitM-Resilient PKI". in: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 2060–2076. ISBN: 9781450356930. DOI: 10.1145/3243734.3243790. URL: <https://doi.org/10.1145/3243734.3243790>

Declaration of Contributions

The paper "Domain Validation++ For MitM-Resilient PKI" was published as a full research paper at the "2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)". It constitutes a joint work of Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman and Michael Waidner.

Haya Shulman proposed initial concept and structured the paper. Haya Shulman also wrote Introduction, Related Work and Conclusions. Tianxiang Dai wrote the offence part including CA selection (Section 2) and attacks against CAs (Section 3), except for Subsection 3.3 about cache overwriting, which was written by Haya Shulman. Tianxiang Dai also wrote the Appendix. Markus Brandt wrote the defence part about DV++ (Section 4). Tianxiang Dai designed, implemented and analysed the evaluations on CAs and popular domains. More specifically, on the CA and resolver side, Tianxiang Dai setup the testing infrastructure and evaluated all CA resolvers' vulnerabilities, except for the cache overwriting test, which was performed by Haya Shulman. On the domain and nameserver side, Tianxiang Dai performed all the evaluations and analysed them. Markus Brandt designed, implemented and analysed the DV++ system. Tianxiang Dai also implemented a proof-of-concept attack and demonstrated it against a chosen CA. Besides, Tianxiang Dai developed the project website. Michael Waidner was a general advisor of this work and contributed with continuous feedback during all phases of the paper writing process. The paper was presented at the conference by Markus Brandt.

All authors agree with the use of their joint paper as part of Tianxiang Dai's cumulative dissertation, considering a contribution of 50% from Tianxiang Dai.

Domain Validation++ For MitM-Resilient PKI

Markus Brandt
Fraunhofer SIT
TU Darmstadt

Tianxiang Dai
Fraunhofer SIT

Amit Klein
Fraunhofer SIT

Haya Shulman
Fraunhofer SIT
TU Darmstadt

Michael Waidner
Fraunhofer SIT
TU Darmstadt

ABSTRACT

The security of Internet-based applications fundamentally relies on the trustworthiness of Certificate Authorities (CAs). We practically demonstrate for the first time that even a weak *off-path* attacker can effectively subvert the trustworthiness of popular commercially used CAs. Our attack targets CAs which use Domain Validation (DV) for authenticating domain ownership; collectively these CAs control 99% of the certificates market. The attack utilises DNS Cache poisoning and tricks the CA into issuing fraudulent certificates for domains the attacker does not legitimately own – namely certificates binding the attacker’s public key to a victim domain.

We discuss short and long term defences, but argue that they fall short of securing DV. To mitigate the threats we propose Domain Validation++ (DV++). DV++ replaces the need in cryptography through assumptions in distributed systems. While retaining the benefits of DV (automation, efficiency and low costs) DV++ is secure even against Man-in-the-Middle (MitM) attackers. Deployment of DV++ is simple and does not require changing the existing infrastructure nor systems of the CAs. We demonstrate security of DV++ under realistic assumptions and provide open source access to DV++ implementation.

CCS CONCEPTS

• **Security and privacy** → **Security protocols**; *Public key encryption*; *Distributed systems security*;

KEYWORDS

PKI security, DNS cache poisoning, Certificates, CA attacks

ACM Reference Format:

Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. 2018. Domain Validation++ For MitM-Resilient PKI. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*, October 15–19, 2018, Toronto, ON, Canada. ACM, New York, NY, USA, 17 pages. <https://doi.org/10.1145/3243734.3243790>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CCS '18, October 15–19, 2018, Toronto, ON, Canada

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-5693-0/18/10...\$15.00

<https://doi.org/10.1145/3243734.3243790>

1 INTRODUCTION

Stability and security of web ecosystem rely on Certificate Authorities (CAs) to ensure that services are trusted and communication to them is secure. CAs vouch for trustworthiness of a service by issuing a digital certificate that binds a domain name to a public key of the service. Upon receiving the request for a certificate for domain, say `vict.im`, a CA validates that the server issuing the request owns and controls the domain for which it requests the certificate. After a successful validation of the ownership of the domain, CA issues the certificate. The certificate contains, among others, the public key of the requesting server and the requested domain. Domain name within the certificate is a key element on which trust can be built. The certificate is signed by the private key of the CA. The server then uses this certificate to prove its identity to clients in the Internet. The clients use the server’s public key in the certificate to establish a secure (encrypted and authenticated) connection to the server.

Browsers have hundreds of registered CAs, and a valid certificate for any domain signed by any of these trusted CAs is accepted by the browsers. Hence, correctly verifying ownership of the domain during the certificate issuance is critical to ensure the security of the clients and services. There are a number of approaches that CAs can use to establish ownership of domains: Domain Validation (DV), Organisation Validation (OV) and Extended Validation (EV). DV provides a number of techniques (e.g., using Email or Domain Name System (DNS)) that allow to prove in an automated way that the applicant owns a given domain name. The idea behind DV is that only the owner of the domain can receive the communication sent to the services in that domain and can respond to them. EV and OV are meant to ensure more stringent certification and are carried out with some human interaction.

Although EV and OV provide a higher assurance of ownership, they are cumbersome and inefficient since they require manual processes for establishing the identity of the applicant, e.g., communication with the customer that requests the certificate, phone calls to the company, additional documents, such as personal identification card, or impose checks against official government sources. In addition, manual verification during certificates generation is more lengthy and results in high costs, e.g., certificates’ prices can exceed 1000 USD. In contrast, the automation offered by DV enabled to reduce the certificates’ prices while improving efficiency of the certificates’ issuance process and ultimately increasing the market share of DV supporting CAs to 99%.

In this work we explore the security of the DV procedure used by the CAs to establish ownership of domains. We identify vulnerabilities and show that even a weak off-path attackers can trick the

DV process and issue certificates for domains they do not own. We evaluate the attack against CAs and show that vulnerabilities exist in popular and large CAs. Our attack exploits vulnerabilities in DNS, which allow us to inject incorrect mappings into the caches of DNS resolution platforms of CAs. These mappings map the target domain to attacker's controlled IP addresses. As a result, the CAs perform the DV process against the hosts that are controlled by the attacker and not against the real owner of the domain. We discuss obstacles and challenges and show how to overcome them and launch a successful off-path attack during the certificate issuance. Our results demonstrate that *Public Key Infrastructure (PKI)*¹ which is meant to provide security against strong Man-in-the-Middle (MitM) attackers, is relying on a weak building block that can be circumvented by an off-path attacker.

We discuss short term patches but show that they do not mitigate the vulnerability. A cryptographic defence for DNS, DNSSEC [RFC4033-RFC4035], would prevent the attacks, but it would take long until the domains are protected with DNSSEC and DNS resolvers perform validation (we explain this in Section 3.6.2). We build upon the ideas of replacing cryptography through assumptions in distributed systems, [19], and design and implement Domain Validation++ (DV++) - a distributed mechanism for authenticating ownership of Internet domains. We show that DV++ is resilient to MitM attackers, while retaining the benefits of DV.

PKI Security

A large research effort is focused on evaluating and improving security of PKI; see Section A for background on PKI. There are works that evaluate security of keys used to establish a secure communication [29], others showed how to exploit side channels to recover plaintext from encrypted communication [6], or launch downgrade attacks for recovering the encrypted communication [9]. There is also a history of attacks against PKI, typically by compromising a CA (we review these in Related Work, Section 5). However, no attention was given to the authentication of ownership used by the CAs. Correctly authenticating ownership of resources is a key element in certification and essential for building trust in the cryptographic material used for securing the communication.

The vulnerabilities along with the need to secure PKI motivated research on security mechanisms, most of which propose alternative models for PKI. Some proposals attempt to identify compromised certificates by using log servers which monitor CAs behaviour, for instance, Certificate Transparency [42], Sovereign Keys [21], Accountable Key Infrastructure [38], Attack-Resilient Public Key Infrastructure [12].

The proposals provide a good starting point and promising directions for design of future PKI. However, most are not adopted due to their prohibitive complexity and performance as well as the changes that they require to the existing infrastructure and the deployment overhead (e.g., introduction of multiple interacting entities). In this work we propose improvements to the existing PKI without changing the infrastructure or introducing new actors - we design and implement DV++ which replaces DV without requiring further modifications to the PKI. In contrast to DV, which we show

is vulnerable even to off-path attacks, DV++ provides resilience against the strong (on-path) Man-in-the-Middle (MitM) attackers. Security against MitM attackers is essential since PKI needs to operate over untrusted networks. Hence it is prudent to assume that the attackers can eavesdrop, modify and inject messages. A critical property of DV++ is that it requires no changes to the existing CAs ecosystem. The interface and communication with the CA and with the verified domain are identical to DV. The difference is in the verification process which DV++ applies - which is transparent to the other actors in the CA ecosystem. We explain this in Section 4.

Attacker Model

For our attacks we use the weakest off-path attacker, which does not have access to the communication of legitimate parties. The attacker can generate packets and spoof source IP addresses. Notice that often off-path attackers can gain MitM capabilities, e.g., by launching BGP prefix hijacking attacks. Indeed, attackers are becoming more sophisticated and increasingly leverage BGP hijacking for DNS cache poisoning [44]. Such attackers obtain MitM capabilities for a short period of time and can efficiently launch the attacks described in this work. We demonstrate however, that even weaker (complete off-path) attackers, can subvert the security of DV validation in PKI.

Disclosure and Ethics

Our research shows that even weak off-path attackers can exploit vulnerabilities to issue fraudulent certificates. This puts at risk not only the vulnerable CAs but the entire PKI ecosystem, with services and clients. Nevertheless, we believe that this research is important: since the vulnerabilities exist, they may be exploited by attackers for malicious purposes without notifying the affected entities. Our goal is to expose and mitigate these issues. We are disclosing the vulnerabilities and are in contact with the affected CAs.

In the attacks that we demonstrate in this work we leverage DNS cache poisoning for injecting spoofed records into DNS caches. Hence to mitigate our attacks the immediate short term countermeasure is to fix the vulnerabilities in DNS that allow injection of records into caches. We notified the affected DNS vendors, DNS operators and service providers of the vulnerabilities.

Our attacks did not target any existing Internet clients and domains. Evaluation of our attacks against CAs' DNS resolvers were carried out using a domain that we own (for simplicity in this work we use domain `vict.im`). This ensured that the CA would not use the spoofed records for any "real" purpose. We setup a set of attacking hosts, which were issuing certificates for resources in domain `vict.im`. Our techniques can be applied to attack other domains (we survey the attack surface of popular domains that can be potential victims) hence adoption of mitigations is critical.

Our attacks were carried out ensuring that the normal CAs functionality is not affected. Part of our study was inferring different cache overwriting vulnerabilities, which we tested against the DV-supporting CAs. The study of caches introduces a larger volume of traffic, than say, merely running an attack with the goal of exploiting a single vulnerability. To avoid any potential load on the CAs infrastructure we distributed the study over a long period of time, with waiting intervals between the requests for issuing certificates.

¹PKI is a set of roles, policies, procedures and entities for creating and managing certificates and public-key encryption.

Hence, guaranteeing that we do not generate an excessive traffic volume on the CAs.

Contributions

In this work we show that the DV mechanism, applied by the CAs to authenticate ownership of domains, can be circumvented by off-path attackers. In our attacks we leverage DNS cache poisoning for injecting spoofed records into caching DNS resolvers, mapping the resources in the target domain to attacker-controlled hosts. As a result, the DV performed by the CA is run against attacker's hosts (in this work these are our machines), which allows the off-path attacker to successfully pass the validation, and to receive fraudulent certificates (signed by a CA) for domains that the attacker does not own. We demonstrate the attack and evaluate it against DV supporting CAs. We successfully launched the complete attack against 7 of the DV-supporting CAs. Hence at least 7 CAs are vulnerable to our attack, but potentially more DV-supporting CAs are vulnerable; we explain the conditions for successful attack in Section 3. Unfortunately, even a single vulnerable CA is sufficient for subverting the security of PKI. This is due to the fact that the security of PKI is based on the security of the weakest link in the infrastructure - compromising a single CA allows attackers to issue fraudulent certificates for any domain, which would then be accepted by any operating system and any browser which have that vulnerable CA on a list of trusted CAs.

Our work is the first to weaponise off-path DNS cache poisoning for attacking a complex system, such as the certificates generation. Prior to our work, such off-path attacks were considered anecdotal and rather on a theoretical spectrum. DNS cache poisoning attacks are known to be launched in the wild, but these are done with a MitM attacker, which observes the requests and can efficiently craft malicious DNS responses, e.g., DNS cache poisoning for email hijacking [18] or for stealing digital cash [44]. In this work we provide the first demonstration of exploitation of off-path DNS cache poisoning for attacking a critical system - the certificate generation in PKI.

We discuss possible mitigations but argue that they do not solve the problem. Cryptographic protection of DNS would prevent the attacks but it is not clear when DNSSEC is going to be fully deployed. We follow the ideas of [19] for replacing cryptography through assumptions in distributed systems. We propose DV++, a modification to DV, which maintains the benefits of DV (it is efficient, automated, fits within the existing business model) while providing resilience even against MitM attackers. We discuss how DV++ can be useful also in other settings where mechanisms rely on correct and secure DNS functionality. We make the code of DV publicly available.

Organisation

In Section 2 we discuss the CAs ecosystem and explain which CAs we focus on in this work. In Section 3 we present the different modules in our attack and then show how to apply them to trick CAs to issue certificates for domains that the applicant does not own. We also survey the attack surface of domains that are potential victims. We provide recommendations and discuss their impact on clients. In Section 4 we propose DV++, and provide its implementation

and experimental evaluations. In Section 5 we review related work. Finally, in Section 6 we conclude this work. Appendix, Section A provides background on DNS and PKI.

2 DV-SUPPORTING CA LIST

2.1 Selecting CAs

Although there are 122 root CAs, the 51 DV supporting CAs control more than 99% of certificates market share^{2,3}. The other CAs are resellers which use root CAs, device based CAs, e.g., for ID card or hardware, or country-specific CAs, which accept only specific country code for issuing certificates.

In our study we focus on CAs supporting DV, with which we could register and issue certificates. We list them in Figure 1. Root certificate programs can be extracted from the browsers and the Operating Systems (OS). We extracted CAs from the following OS: Internet Explorer with Windows, Apple with OS X/iOS, Mozilla with Linux.

2.2 Issuing Certificates

To issue a certificate an applicant should fill out a form called Certificate Signing Request (CSR) on CAs websites. The CSR contains information that is included in the certificate, such as organisation name, domain name, country, public key and more. A CA then uses the submitted CSR to authenticate the domain ownership by the applicant and subsequently to issue the certificate. When submitting the CSR the applicant should also specify which DV method it would like the CA to use for authenticating ownership of the domain. In what follows we describe the DV procedures that are supported by the CAs.

2.3 DV Methods

There are a number of methods for performing DV. We list them in Figure 1. Some CAs support more than one DV method. When a CA supports more than one method, the applicant can specify which method it wishes to use. All the methods rely on DNS, and can be subverted via DNS cache poisoning. We summarise which CA supports what DV methods in Figure 1, and below explain how the validation is performed:

2.3.1 Email-Based DV. Upon filling out a CSR an email is issued to the administrative contact of the domain selected by the applicant out of email addresses registered for that domain in Whois. The email typically contains validation code and link, which the recipient has to click and enter the code to prove control over the domain. If the correct code is entered the code proceeds with the certificate issuance.

2.3.2 WHOIS-Based DV. Similar to email-based DV, except that the client cannot select which email (out of those registered as administrative for the domain) will be used in the validation. During the DV procedure the CA selects itself the email address and can use any Admin, Registrant, Tech or Zone contact email address that appears in the domain's WHOIS record.

²https://w3techs.com/technologies/overview/ssl_certificate/all

³<https://www.netcraft.com/internet-data-mining/ssl-survey/>

2.3.3 *DNS-Based DV.* Upon submitting a CSR, a hash value is provided which has to be entered as a DNS CNAME Resource Record (RR) for the domain in the zonefile. For example, assume that applicant's domain is `vict.im` and CA's domain is `ca-domain.com`. The CNAME record would be:

hash1.www.vict.im. CNAME hash2.ca-domain.com.

The DNS resolver of the CA queries the domain of the applicant and checks the presence of the CNAME record. If the correct record is present, the CA proceeds to issue the requested certificate.

2.3.4 *HTTP/S-Based DV.* Upon the submission of a CSR, a hash value is returned to the client. A file should be created and placed at the root of the web server with the hash value as its name, as follows: `http://www.vict.im/hash-value1.txt`.

The content of the file should contain `hash-value2` and the domain `ca-domain.com`.

The CA makes an HTTP/HTTPS request to retrieve the file. If correct, the CA proceeds to issue the certificate.

		Email	DNS	HTTP/S	WHOIS
1	COMODO	✓	✓	✓	
2	DigiCert				✓
3	Entrust				✓
4	GeoTrust	✓	✓	✓	
5	GlobalSign	✓	✓	✓	
6	GoDaddy	✓	✓	✓	
7	NetworkSolutions	✓			
8	SSL.com	✓	✓	✓	
9	SwissSign	✓			
10	Thawte	✓			
11	Trustwave				✓
12	Symantec				✓
13	StartCom	✓			
14	Let'sEncrypt		✓	✓	
15	Unizeto	✓	✓		
16	NETLOCK	✓		✓	
17	IdenTrust				✓
18	RapidSSL	✓	✓	✓	
19	StartSSL	✓			
20	Certum	✓	✓	✓	
21	InstantSSL	✓	✓	✓	

Figure 1: List of CAs and the supported DV methods.

3 OFF-PATH ATTACKS AGAINST DV

In this section we show that an off-path attacker can impersonate a victim domain to a CA and cause the CA to issue a spoofed (fraudulent) certificate binding the public key of an off-path attacker to a victim domain. The main ingredient in impersonating a victim domain is a DNS cache poisoning attack against a caching resolver of the CA. During the attack we inject a spoofed DNS record mapping the CA to an attacker controlled email and DNS server. As a result the DV process is performed against the attacker-controlled host, to which the attacker can respond, impersonating a victim domain.

The attack consists of a number of components. First we show how to cause the *victim* CA to issue a DNS request to the name-server controlled by the attacker. Next we show how to match the challenge response authentication parameters in the *spoofed* DNS response, such that the response is accepted by the receiving

caching DNS resolver of the CA. Finally, we explain how to construct the records in the spoofed DNS response, so that they are cached by the receiving caching resolver. We then present the attack and provide a measurement study of the CAs and the potential victim domains (from popular Alexa `www.alexacom` domains) to estimate the number of clients and servers in the Internet vulnerable to our attack.

3.1 Triggering DNS Request

To initiate our study of cache poisoning vulnerabilities in CAs we need to trigger a DNS request from the *victim* DNS caching resolver to our nameservers. The problem is that the resolvers are configured not to respond to external requests, and since we are not on the network of the CA triggering a DNS request from the outside is a challenge. We trigger a query indirectly. To trigger a DNS request we upload a CSR – this initiates the DV procedure against the target domain (which we provide in the CSR). The “target domain” is the victim domain whose records the attacker wishes to poison in CA’s DNS resolver. As a target domain in this work we use `vict.im`.

In the rest of this section, our study and attacks are performed in a black-box manner, using the requests that are sent from the caching DNS resolver of the CA to our nameservers. In response to the DNS request of the caching resolver we generate responses that cause the resolver to follow-up with DNS requests. This is achieved with, e.g., referral type responses, as well with responses with CNAME records. These interactions allow us to characterise the caches, identify vulnerabilities and eventually launch the attacks.

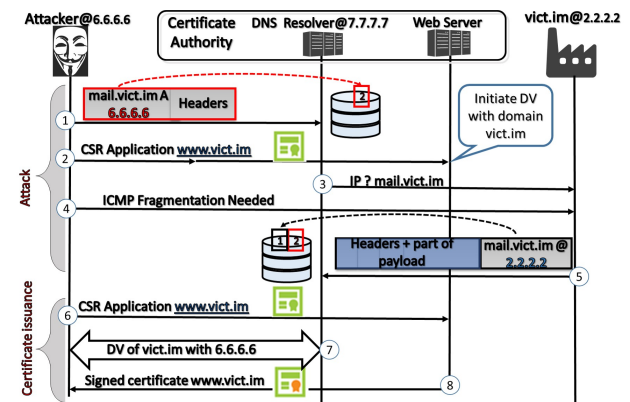


Figure 2: Defragmentation cache poisoning for subverting DV.

3.2 Defragmentation Cache Poisoning

The goal of our attacker is to spoof a DNS response packet from the nameserver to the CA’s DNS resolver, which the resolver accepts as valid.

Once a DNS request is issued the time window for DNS cache poisoning attack is initiated. The window ends either when a timeout event happens or when a correct response arrives. To craft a correct response, the attacker has to guess the challenge-response authentication parameters in the DNS request. These are different values, most notably, source port and DNS Transaction Identifier (TXID),

which are randomised by the DNS resolver in the request, and are validated in the response [RFC5452]; we provide background on DNS and security against cache poisoning in Appendix, Section A. Figure 3 illustrates a structure of a DNS request from DNS resolver at IP address 7.7.7.7 to a nameserver at IP address 2.2.2.2 sent from UDP source port 12345 and with TXID of 76543.

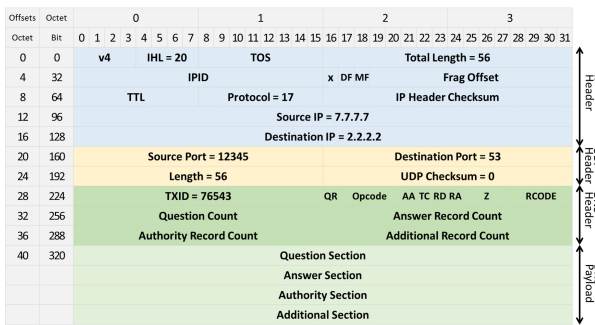


Figure 3: DNS request packet from DNS resolver at 7.7.7.7 to a nameserver at 7.7.7.7

Since source port and TXID are both 16 bits, they generate together a range of 2^{32} possible values. This is a lot of entropy hence it seems to make DNS robust to off-path attacks. Specifically, the attacker must spoof DNS responses with the correct source port and TXID which is about 2^{32} bits of entropy. In this section we show that this is not the case. We show how an off-path attacker can hijack the communication between the resolver and the nameserver. The key ingredient is overlapping IPv4 packet fragments. Our attacker does not attempt to guess the source port and TXID. Instead we use IPv4 packet fragmentation to overwrite the relevant fields in a real DNS response from the nameserver, with malicious values. In what follows, we explain fragmentation and Maximum Transmission Unit (MTU), we discuss how to force a server to send a fragmented response, and then describe how to combine these for attacking a defragmentation cache to inject a spoofed record into the caching DNS resolver of the CA.

3.2.1 Maximum Transmission Unit. The Maximum Transmission Unit (MTU) is the largest number of bytes that can be transmitted over a link in one datagram. The Path MTU between two end points is limited by the lowest MTU on the path [RFC791]. When packets exceed the path MTU, they will not be received at the destination. To cope with this, the Internet Protocol (IP) provides the possibility to fragment packets into smaller fragmented packets. Most networks support currently MTU of 1492 bytes [RFC2516]. According to [RFC791] the minimum MTU in the Internet is 68 bytes. The DNS responses typically do not exceed 1500 bytes, unless they contain a cryptographic material due to the use of DNSSEC [RFC4033-RFC4035]. Hence, most DNS responses would not fragment.

The idea of the attack is that the attacker “convinces” the nameserver to fragment responses to a specific destination. As a result, the nameserver responds with a fragmented packet, such that the second fragment contains either the additional or also the authority

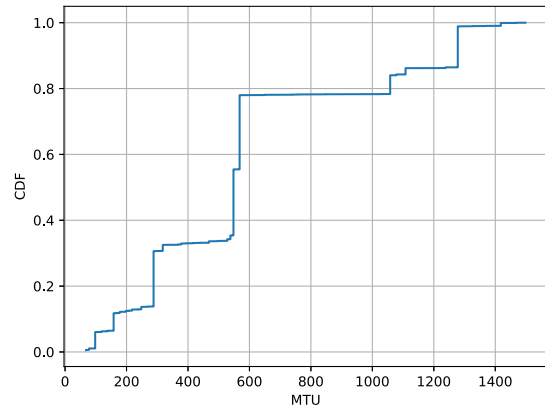


Figure 4: CDF of the packets' sizes in response to an ICMP fragmentation needed error message for servers that reduced their responses sizes.

sections of a DNS response; see Appendix A for details on DNS packet structure.

3.2.2 ICMP Fragmentation Needed. To reduce fragmentation load on the routers, [RFC1191] proposes a mechanism to discover the MTU between two hosts. To do so, hosts make use of the Do Not Fragment (DF) bit in the IP header to instruct the routers along the path to not fragment the packet in case the packet exceeds the MTU of the next hop. Instead, intermediate hosts will discard the packet and issue an *ICMP Destination Unreachable* error message (type 3) to the originator with the code *Fragmentation Needed and DF set* (code 4). The information in the ICMP error message is stored by the receiving OS, e.g., 10 minutes by default on Linux 3.13. The ICMP error message can be originated by any Internet node on the path between the sender and the receiver and the receiver of the ICMP error message is not expected to know the IP addresses of the nodes on the path. Hence, we use an off-path attacker to issue an ICMP fragmentation needed packet to the nameserver, indicating that it should reduce the MTU when sending packets to the *victim* resolver.

We measured the fraction of servers among 5K-top Alexa that reduce the MTU size following ICMP fragmentation needed packet: 33, 4% of the servers reduce the packet size up to 296 bytes, and 11% reduce the fragment size to below 296 bytes. Figure 4 shows the CDF of packet sizes that were received in response to an ICMP fragmentation needed packet. While less than 15% of servers reduce their packet size below 296 bytes, almost 80% of servers are willing to reduce the packet size below 600 bytes (the two steps in Figure 4 represent 552 bytes, which is the default minimal value in the Linux kernel and 576, which is the suggested minimum MTU by the RFC).

The spoofed ICMP error message does not need to be sent from a spoofed source IP address - ICMP error message can originate from *any* node on the path. In contrast to ICMP with TCP headers, the OSes typically do not apply any checks on the received ICMP error

messages with UDP headers, e.g., Linux 3.13. This is due to the fact that UDP is stateless. Hence, crafting a spoofed ICMP fragmentation needed error message is a simple task.

Figure 5 shows an example ICMP fragmentation needed error sent by an attacker at IP 6.6.6.6 to nameserver at IP 2.2.2.2 telling the nameserver to reduce the MTU to 100 bytes for all packets it sends to the DNS resolver at 7.7.7.7. The payload of the ICMP packet is the IP header and the first eight bytes of the original packet (that triggered the ICMP error message). The nameserver stores this information and uses it for limiting the size of IP packets to destination IP (7.7.7.7) and protocol (UDP).

Offsets	Octet	0	1	2	3
Octet	Bit	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	
0	0	v4	IHL = 20	TOS	Total Length = 56
4	32	IPID		x DF MF	Frag Offset
8	64	TTL	Protocol = 1	IP Header Checksum	
12	96	Source IP = 6.6.6.6			
16	128	Destination IP = 2.2.2.2			
20	160	Type = 3	Code = 4	ICMP Checksum	
24	192	Unused		MTU = 100	
28	224	v4	IHL = 20	TOS	Total Length = 76
32	256	IPID		x DF MF	Frag Offset
36	288	TTL	Protocol = 17	IP Header Checksum	
40	320	Source IP = 2.2.2.2			
44	352	Destination IP = 7.7.7.7			
48	384	Source Port = 53		Destination Port = 12345	
52	416	Length = 56		UDP Checksum = 0	

Figure 5: ICMP fragmentation needed packet from attacker at 6.6.6.6 to nameserver at 2.2.2.2 indicating an MTU of 100 bytes for resolver at 7.7.7.7.

3.2.3 IPv4 Fragmentation and Reassembly. Upon arrival to the receiver the fragments of an IP packet are stored in an IP defragmentation cache, by default for 30 seconds. The receiver uses the IP ID value in the fragments to identify all the fragments of the same original IP packet (they all have the same IP ID value). Then uses the offset field to reassemble their payload together. The receiver knows that all the fragments of the original IP packet have arrived by checking that the fragment with the lowest offset has a More Fragments (MF) value of zero.

Offsets	Octet	0	1	2	3	
Octet	Bit	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31		
0	0	v4	IHL = 20	TOS	Total Length = 85	
4	32	IPID = 23456		x DF MF	Frag Offset = 48	
8	64	TTL	Protocol = 17	IP Header Checksum		
12	96	Source IP = 2.2.2.2				
16	128	Destination IP = 7.7.7.7				
20	160	Data Length = 4	IPv4 Address			
24	192	= 2.2.2.2			Name = 0	Type
28	224	= OPT	UDP Payload Size = 4096	EXTENDED-RCODE = 0		
32	256	Version = 0	DO	Z	Data Length	
36	288	= 0				

Figure 6: Malicious second fragment sent by attacker at 6.6.6.6 from a spoofed IP address 2.2.2.2 to the DNS resolver at 7.7.7.7, assuming MTU of 68 bytes.

Offsets	Octet	0	1	2	3
Octet	Bit	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	
0	0	v4	IHL = 20	TOS	Total Length = 85
4	32	IPID = 23456		x DF MF	Frag Offset = 0
8	64	TTL	Protocol = 17	IP Header Checksum	
12	96	Source IP = 2.2.2.2			
16	128	Destination IP = 7.7.7.7			
20	160	Source Port = 53		Destination Port = 12345	
24	192	Length = 65		UDP Checksum = 0x14de	
28	224	TXID = 76543	QR Opcode = 0	AA TC RD RA	Z RCODE = 0
32	256	Question Count = 1		Answer Record Count = 1	
36	288	Authority Record Count = 0		Additional Record Count = 1	
40	320	4	m	a	i
44	352	l	4	v	i
48	384	c	t	2	i
52	416	m		Type = A	
56	448	Class = IN		Name (Pointer)	
60	480	Type = A		Class = IN	
64	512	TTL			

Figure 7: First fragment sent by the nameserver at 2.2.2.2 to the DNS resolver at 7.7.7.7, assuming MTU of 68 bytes.

3.2.4 Exploiting Fragmentation for Defrag. Cache Poisoning. In this section we show how we exploit fragmentation for injecting a spoofed payload into a DNS response from the real nameserver. For our attack we exploit fragmentation, and trick the receiving resolver into reassembling the first fragment from the real response from the nameserver with the second fragment generated by the attacker. This allows us to bypass the challenge-response authentication fields used by the DNS resolvers, since they are echoed by the nameserver in the first fragment. In Section 3.2.2 we showed how we ensure that the response from the target nameserver is fragmented.

Assume that the attacker wishes to impersonate vict.im, and get a certificate for vict.im with a mapping to an attacker controlled IP address. The attack is initiated with an off-line preprocessing phase, during which the attacker needs to perform a measurement and a calculation, which it will use to set the values in the ICMP fragmentation needed error message in step (4) (Figure 2) and based on which it will construct the spoofed second fragment in step (1) (Figure 2). The attacker measures the responses' sizes from the nameserver of vict.im and calculates the offset where the fragmentation should occur. The MTU in the ICMP fragmentation needed error message is set accordingly. The goal is to ensure that the records, which it will replace with spoofed records, are in second fragment.

The attack proceeds as illustrated in Figure 2. In step (1) the attacker sends to the victim DNS resolver a spoofed second fragment (see spoofed second fragment illustrated in Figure 6). The fragment is cached by the receiving resolver in the IP defragmentation cache waiting for the remaining fragments to arrive. In step (2) the attacker uploads a CSR form requesting certification for domain vict.im. The attacker selects an email based DV, hence in step (3) the DNS resolver at the CA issues a DNS request to the nameserver of vict.im domain asking for an IP address of the email server mail.vict.im. The attacker issues an ICMP fragmentation needed packet in step (4) to ensure that the response is fragmented; this ICMP error message can be issued also before step (3). In step (5) the nameserver of vict.im sends a fragmented DNS response. The first fragment contains the entropy (resolver's source port and TXID are copied by the nameserver to the response) and some of the DNS payload, such as the question section, answer section, and

probably part of the authoritative section (see first fragment in Figure 7). Some of the records from the authoritative section (or the complete authoritative section) as well as the additional section are in the second fragment. The first fragment is then reassembled with the spoofed second fragment that was already in the cache of the DNS resolver; they both leave the cache, and are passed on to the UDP layer. The legitimate second fragment will not have any first fragment to reassemble with, since when it arrives the legitimate first fragment have already been reassembled with the spoofed second fragment and left the buffer. The remaining steps (6)-(8) complete the DV process.

Notice that in step (1) when sending the spoofed fragment the attacker needs to ensure the correctness of UDP checksum and IP ID. In what follows we describe both challenges and explain how we ensure them.

3.2.5 UDP Checksum. The method to calculate UDP checksum is defined in [RFC768]. The UDP checksum is calculated as the 16-bit one's complement of the one's complement sum of the fields in IP header, the UDP header and the UDP payload. The UDP checksum value is stored in the UDP header. Upon receipt of a UDP packet the receiver calculates the UDP checksum (using the same procedure as the sender did) and then compares whether it is the same value as in UDP header. If the value differs - the UDP datagram is discarded.

If the checksum in the second spoofed fragment is not adjusted, When the spoofed second fragment is reassembled with the first real fragment, the overall value of the UDP checksum of both fragments is altered, and will differ from the UDP checksum value in the UDP header. Specifically, the spoofed second fragment contains different records than the real second fragment. Hence, the attacker needs to adjust the UDP checksum of the second fragment such that the computation of the UDP checksum using the payload in both fragments remains similar to the original value of UDP checksum in the UDP header.

The attacker can ensure this as follows: the attacker sends a request to the nameserver and receives a response in two fragments (possibly using the ICMP fragmentation needed packet to enforce fragmentation). The attacker calculates the UDP checksum value of the second fragment, then alters the second fragment and calculates the checksum of the altered second fragment. The difference value in the checksum of both second fragments the attacker needs to add (or remove) by altering two bytes (in even location) in the altered second fragment. This is a simple computation since the attacker knows the original value, and knows what bytes were modified, it can efficiently compute the difference and add (or remove) it to the value of UDP checksum of second fragment.

To adjust the value of two bytes the attacker can either modify two bytes in any record in the second altered fragment, or can add the two bytes at the end, after the EDNS [RFC6891] record (in this case the DNS software will ignore these two bytes but they will be included in the UDP checksum computation).

3.2.6 IP ID. The spoofed second fragment should contain the correct IP ID value - the same value as the original IP packet sent by the nameserver. The attacker must predict this value correctly, otherwise, when the first fragment and the spoofed second fragment have different IP ID values, the receiving OS will not reassemble them together.

Predicting the IP ID value can typically be done efficiently. There are three IP ID assignment algorithms: sequentially incrementing (the OS increments IP ID value by one for every sent packet), per-destination sequentially incrementing (the OS increments IP ID by one for every packet sent to a given destination), and random [24, 41]. We describe the three cases below.

Sequentially Incrementing IP ID. More than 60% of 10K-top Alexa domains use sequentially incrementing IP ID values assignment. Windows operating systems use a sequentially incrementing IP ID.

The attacker samples the IP ID value from the nameserver and samples the IP ID increase rate. Then the attacker calculates the IP ID value that will be assigned at the time the nameserver sends the DNS response to the victim DNS resolver. The attacker uses this value in the second spoofed fragment. The attacker can also send multiple fragments with different IP ID values to increase his chances to hit the correct IP ID value.

Per-Destination IP ID. A bit less than 40% of the nameservers use a per-destination incrementing IP ID. Linux versions are using a per-destination incrementing IP ID assignment. Our attacker uses the techniques presented in [41] for predicting the IP ID values. Since this is not the focus of our work, we do not describe the algorithm here and refer an interested reader to [24, 41] for details.

Random IP ID. Very few servers use random IP ID values, less than 1%. The reason is the overhead that it introduces on the server. Specifically, instead of maintaining a single counter (as in the case of globally incrementing) or one counter per destination (in the case of per destination incrementing IP ID), in this case the server has to maintain multiple counters, and to continually check for collisions, i.e., that it does not select an IP ID that was already allocated.

The success probability of the attack in case of a single fragment is $\frac{1}{2^{16}}$. The attacker can increase its success probability by sending multiple fragments, each with a different IP ID value. Assuming an infinite size defragmentation cache, if the attacker sends 2^{16} fragments, its success probability is 1. The existing OSes however limit the number of fragments that can be sent, e.g., Windows to 100 fragments, recent Linux versions to 64 and older allow several thousand fragments [37]. In Linux, the limit is set via `ip_frag_max_dist` parameter.

3.3 Overwriting Cached Records

Defragmentation cache poisoning allows to bypass the challenge response authentication as the entropy is in the first fragment and the attacker injects payload into the second fragment. A successful defragmentation cache poisoning allows to bypass the validation of the OS and the packet is transferred on to the DNS software. Now, the task is to ensure that the records (sent in the spoofed DNS response) are cached and served in responses to applications and clients. The problem is that often the records will already be present in the cache. For instance, records of domains of interest or popular domains, such as banks and social networks, are typically present in caches. The resolvers will not necessarily overwrite the already cached values, in fact, often they will silently ignore records for which there are cached copies but with other values. This poses a challenge - how to overwrite the already cached records with new values?

The tricky part here is that different DNS software assign different trust ranks to DNS records and apply different logic when overwriting the cached records with new values. The higher the rank the more difficult it is to overwrite the cached record with a new value. The ranking of DNS records is discussed in [RFC2181], and these recommendations are interpreted and implemented in each software differently. Since the records' overwriting behaviour is so different between the different DNS resolver software, it can be used to fingerprint the software of the DNS cache.

To understand how to overwrite cached records with new values we did a characterisation of DNS caches of CAs and modelled under which conditions the caches replace copies of cached records with new values. Our study builds on [40] which evaluated different approaches for overwriting cached records. We next describe the setup, the study and the results.

3.3.1 Setup. Based on a lab model of the caches' overwriting behaviour, we applied our study over the resolvers of CAs. We used the `vict.im` domain and its subdomains for our study. The zone file was configured with 6 nameservers each with an A record (IPv4 address) and AAAA record (IPv6 address), 3 MX records (incoming email servers), and other records, such as DNS specific records (e.g., SOA) and anti-spam records (e.g., SPF). The study proceeded in a black box manner - we cause the CAs to issue DNS requests for records within `vict.im` and monitor requests that arrive to our nameservers. We generate responses dynamically on the nameserver with program we built based on `Stanford::DNSserver` perl library. We then use the monitored queries to characterise the caches and to understand the overwriting behaviour.

In Appendix, in Figure 12 we recap a number of selected payloads (see full list in [40]) of the DNS responses and the cached records that they can overwrite, using records that are returned in answer, authority and/or additional sections. Our measurement of minimum MTU values in popular Alexa servers shows that the responses can be often reduced to even 68 bytes. Furthermore, fragmentation at the boundaries of authority or additional sections is common and can often be enforced.

In Figure 13 we recap the in-lab evaluation of selected payloads (from [40]) against popular DNS resolvers' software, appliances and public services. The indexes in the leftmost column correspond to indexes of the payloads in Figure 12. The values in cells indicate whether the cache in column j is vulnerable to overwriting by payload in row k : 0 means not vulnerable and 1 means vulnerable. The evaluation indicates that all the resolvers that were tested, except Unbound in hardened mode (which is resilient to all the payloads), are vulnerable to at least one type of cache overwriting payloads.

Caches' Overwriting Study. For each CA, our study proceeds for each payload p_i in Figure 12 in three phases: (a) seed the honey record, (b) overwrite the value of the honey record with a new value, (c) probe the value of the honey record.

We first plant the honey record in the cache of the DNS resolver of the CA. The honey record simulates a real value of our test domain `vict.im`. During phase (b) we attempt to overwrite the honey record with a new value. During phase (c) we send a DNS request for the honey record and check its value. If the value was modified

following step (b), we mark the cache as vulnerable to payload p_i . Next iteration, increment i and evaluate with new payload.

Overwriting Vulnerabilities in CAs. We next list the CAs that were found vulnerable to overwriting attacks. Within our study we also identified CAs that share the same infrastructure (Email server and caching DNS resolver) - we grouped those CAs below.

- COMODO, InstantSSL, NetworkSolutions, SSL.com: these CAs use the same MX email server `mcm111.mcr.colorado.comodo.net` which uses the same caching DNS resolver. The results from our cache overwriting methods indicates that the DNS resolver software is New BIND 9.x with DNSSEC-validation.
- Thawte, GeoTrust, RapidSSL: use the same MX server and resolution platform.
- StartCom⁴, StartSSL: both use the same email server and the same DNS resolver.
- SwissSign: uses New BIND 9.x.

Caches Overwriting Attacker. The study of caches overwriting is performed with a man-in-the-middle (MitM) type attacker - in this step we do not attempt to guess ports or TXID, just to characterise the caches of the CAs. In particular, we observe the requests from the caching DNS resolvers of the CAs and generate the responses dynamically "on the fly" as a function of the requests and the payloads in Figure 12. After characterising the caches of the CAs in this model, we run the complete attack in an off-path attacker model in Section 3.4.

3.4 Evaluation of Attack Against CAs

3.4.1 Setup. We evaluated the attack (with the components in Sections 3.1, 3.2 and 3.3) against the DNS resolvers of CAs (in Figure 1) using our own test domain `vict.im`. The nameservers hosting the test domain were located on one network (belonging to Internet Service Provider (ISP) A), while the attacking hosts were located on a different network (belonging to ISP B). Our nameservers running `vict.im` as well as its subdomains were set up with a globally sequential IP ID allocation. The attacking hosts are configured with a mapping of cache overwrite vulnerabilities per each CA on the list. Namely, when running an attack against CA ω , the attacker selects one of the payloads from the list 12 to which the caching DNS resolver of ω is vulnerable, and uses it to generate the payload of the spoofed second fragment. The attacking host queries the nameserver for IP ID value $ipid$ and uses it to calculate the IP ID value that will be assigned to the response that we will be attacking with a spoofed second fragment.

3.4.2 Issuing Spoofed Certificate. The attacking host initiates the CSR process for victim domain `vict.im` (and its subdomains). After submitting the CSR the attacker selects the DV method that it wishes to pass (CAs typically support a number of DV methods, see Figure 1). Since the attacker cannot control the timing when the request is sent, it must periodically transmit spoofed second fragment, until the validation is initiated. Finally, if the attacker receives a signed certificate for the resource that it requested (i.e., a domain `vict.im` or its subdomains) we mark the CA as vulnerable.

⁴StartCom stopped issuing new certificates in January 2018.

3.4.3 Which Record To Attack? All the different DV methods (in Section 2) generate NS (nameserver) and A (IP address) type requests. In addition, DV with Email (or with WHOIS) also generate MX (email server exchanger) type requests. Responses to these requests are suitable attack targets. Specifically, the attacker hijacks the entire `vict.im` domain by injecting into a response packet a spoofed NS or A records or hijacks an email server by injecting an MX record (with the corresponding A record). Once a CA caches a record mapping the nameserver of `vict.im` to attacker's host, all the subsequent requests (e.g., for email server or webserver) will go to the attacker. The attacker passes any DV verification option listed in Section 2.

3.4.4 Constructing the Second Fragment. Essentially for many domains the attacker can control even the answer section of a DNS response, if fragmentation is at around 200 bytes. For instance, consider fragmentation at 68 bytes. Given a nameserver whose responses fragment so that the first fragment is 68 bytes long, the attacker can control the whole answer section. The headers take 40 bytes: IP header is 20 bytes, UDP header is 8 bytes, DNS header is 12 bytes. The query part (which is the first part in the response) copies the QNAME ($n + 1$ bytes, for QNAME of length n) together with the CLASS (2 bytes) and type (2 bytes). So a QNAME of length 23 or more bytes results in an Answer section getting fragmented before the answer part. Even with first fragment of size 200 bytes, the attacker controls the answer part using QNAME of 160 or more bytes (DNS names can be up to 253 characters). In other cases, where fragmentation occurs significantly above 200 bytes, the attacker controls the authority and additional sections. Depending on the DNS resolver software and the fragmentation restriction the attacker selects the suitable payload for cache poisoning attacks (selected payloads are listed in Figure 12).

3.4.5 Measuring the CAs (Off-Path) Attack Surface. All the CAs we measured use unpredictable source ports and TXIDs. Hence, we use the defragmentation cache poisoning to bypass the entropy in first fragment. As it turns out, many CAs block fragmented IP packets. We evaluated the complete attack against CAs that allow fragmented responses. These include COMODO, InstantSSL, NetworkSolutions, Let's Encrypt, SSL.com, CERTUM, NETLOCK.

3.5 Challenges and Conditions for Success

Our attack does not apply against all the DV-supporting CAs. In this section we explain why, and list the conditions that are required for our attack to succeed. We also discuss the challenges and hurdles in such off-path attack and how we overcame them.

3.5.1 Conditions for Success. Our attack succeeds against a DNS resolver of a given CA, for a specific target (victim) domain `vict.im` if the following conditions hold:

- The authentication of the client requesting a certificate is done via DV. The list of DV-supporting CAs that we could test is in Figure 1.
- The DNS resolver and the network of the CA allow fragmented responses. Some CAs block fragmented traffic (see list in Section 3.4.5).
- The domain `vict.im` does not filter ICMP fragmentation needed error messages and reduces the MTU based on the MTU

indicated in ICMP fragmentation needed packet. The fraction of domains that do not filter ICMP fragmentation needed messages were measured by [27], see Figure 4.

- The caching DNS resolver is vulnerable to at least one of the overwriting attacks. As we show in Section 3.3 all tested resolvers except one (Unbound in hardened mode) are vulnerable to at least one overwriting payload.

3.5.2 Challenges and Hurdles. The attacks we launched are hard due to the following factors:

Putting it All Together. Putting all the modules into a working attack is a challenge. First the attacker has to synchronise all the modules: triggering the query, measuring the IP ID, crafting the spoofed second fragment with the correct payload (that will overwrite the value of a cached record with a new value), injecting the spoofed second fragment into the defragmentation cache. Carrying these steps out correctly requires off-line preprocessing prior to the attack.

Second, some servers fragment not exactly at the location specified in the MTU in ICMP fragmentation needed error message. For instance, when signalling MTU of 512 bytes, some servers may fragment at 500 bytes. To cope with this, depending on the situation, the attacker may need to add padding, or use records' names that fit within the fragment. The attacker can also play with compression in DNS (defined in [RFC1035]) and increase or decrease the size of the injected record. The compression uses pointers to locations where a given string already appeared, hence by using combinations of names that are already in the first fragment the attacker can reduce the size, or with strings that did not appear before, increase the size.

Off-path Attacker. Off-path attackers cannot see the communication between the DNS resolver and the nameservers. This introduces challenges with timing and synchronisation. To overcome this, we did a preprocessing phase, during which we performed a careful study of the certificate issuance procedure for a given CA: the time at which DNS queries are triggered, when validation is performed. The attack is then tailored for each CA based on the observed behaviour of the CA during the preprocessing phase.

Indirect Study. A key ingredient in our attacks is understanding the caching behaviour in the DNS resolver of the CA to identify and evaluate cache overwriting vulnerabilities. The challenge is that we do not have a direct communication to the resolver, and cannot trigger the queries ourselves. We trigger the queries indirectly, using the CSR submission form. This indirect communication introduces noise and prevents us from knowing when the queries will be sent.

Victim Domain. In this work we attacked our own domain. This allowed us to anticipate the traffic volume to the domain and the IP ID assignment method. In a real life scenario the attacker does not have visibility into the nameservers of some other domain, and hence has to learn this information using side channels. To identify the IP ID assignment the attacker sending queries to the nameserver from different hosts and checks the IP ID values in responses. Similarly, the attacker has to measure the IP ID increments by probing the value in intervals. If a random IP ID assignment is used, the attack becomes much more difficult since it needs to be repeated

until the correct IP ID value is hit. Fortunately for the attackers (and unfortunately for security), random IP ID assignment is rare.

3.6 Mitigations and Recommendations

Our evaluations indicate that off-path attacks against CAs are feasible using DNS cache poisoning. We found a few vulnerable CAs, but even a single trusted CA that is attacked exposes to devastating consequences. We next discuss a few possible short and long term mitigations.

3.6.1 Blocking Fragmentation. Perhaps the simplest way to protect against defragmentation cache poisoning (and hence against issuance of spoofed certificates) is by blocking fragmented traffic; which many CAs indeed do. We considered this as a possible simple fix to the problem. However, as we show, MTU reductions are still common in the Internet, e.g., because of PPPoE on DSL lines. How common MTU values less than 1500 bytes in the Internet?

Is Fragmentation Common? We answer this question empirically, using the data captures that Center for Applied Internet Data Analysis (CAIDA) [14] offers. CAIDA operates two passive Internet monitors in the Equinix datacenters in Chicago and San Jose. The monitors capture packets, anonymise them, and provide them for download as *gzipped pcap* files, with the content limited to network and transport-layer protocol headers. Each dataset corresponds to one hour (between 12:59 and 14:01 UTC), split into chunks of one minute (between approx. 150 MB - 1.5 GB). In total, we analysed 14484 traces in 121 datasets, representing data from 9 years (2008 - 2016).

We observe 2.9 million ICMP fragmentation needed packets in total which is approximately one in 135k packets. On average, there are 24k ICMP fragmentation needed packets per dataset. Our measurement study indicates that blocking fragmented traffic would block access to many Internet clients.

The Issue Is More Significant. Nevertheless, even blocking fragments would not block the vulnerability. Off-path attackers may come up with other approaches to bypass the randomisation. Furthermore, there is an increasing tendency to launch short-lived Border Gateway Protocol (BGP) prefix hijacks, for redirecting DNS requests via attacker's network for launching DNS cache poisoning attacks. This allows the attacker to become a MitM (on-path) attacker for a short period of time, sufficient to inspect the DNS request, and to craft a correct DNS response (with valid challenge-response parameters), e.g., see [44]. In this recent attack, the attackers leveraged BGP prefix hijacking for launching DNS cache poisoning to redirect myetherwallet clients to attacker controlled servers.

Hence, the solution should be robust even against stronger MitM attackers, and should not rely on patches against off-path attackers.

3.6.2 DNSSEC. To mitigate DNS cache poisoning attacks even by MitM attackers, the IETF designed and standardised Domain Name System Security Extensions (DNSSEC) [RFC4033-RFC4035]. With DNSSEC, nameservers respond with signed DNS records and keys and the DNS resolvers validate the records prior to caching them. Fully deployed DNSSEC would prevent DNS cache poisoning attacks.

Nevertheless, although proposed and standardised more than two decades ago, DNSSEC is still not widely deployed. Measurements show that currently about 25% of the DNS resolvers validate DNSSEC (e.g., see `stats.labs.apnic.net/dnssec`) and only a bit more than 1% of domains are signed with DNSSEC, [51]. This means that DNSSEC validating DNS resolver gets no security benefit since most domains are not signed. More significantly, recent works found problems with DNSSEC keys generation and management procedures exposing a large fraction of signed domains (more than 35%) to attacks, [16, 47]. It is not clear when the problems are expected to be resolved since they involve practices used by many large registrars and DNS hosting providers.

Therefore, it is important to improve security of PKI independently of other defences. In the next section we introduce our proposal DV++ for a MitM-resilient PKI.

4 DOMAIN VALIDATION++

Our goal is to design a defence that preserves the benefits of DV while providing resilience against MitM attackers. We aim to ensure that the integration of the new mechanism would not require any changes to the CA infrastructure and functionality and that it should be easy to deploy. These properties ensure that the mechanism will have better changes to be used by the CAs.

In this section, we present the design, implementation, and simulations of our proposal.

4.1 Setup and Attacker Model

The main aspect of our proposal is to utilise distributed nodes which perform DV from multiple vantage points. The security against MitM attackers is achieved by placing the nodes in different networks, which do not traverse overlapping paths. In Figure 8 we provide an illustration of the attacker model, the CAs, and the DV++ agents.

In contrast to a cryptographic eavesdropping attacker, which is a global eavesdropper, a realistic MitM attacker can be present only on some networks but does not control the entire Internet. This serves as a premise for our design of DV++. The attacker is a malicious ISP, that passively collects the traffic that traverses its networks. The attacker can also actively try to attract traffic from other networks by performing BGP prefix hijacking.

4.2 Design

DV++ is a decentralised mechanism, that utilises the existing Internet infrastructure to validate claims of domain ownership. In contrast to the centralised validation performed by the CAs with DV, DV++ is based on a flat hierarchy with several equally trusted certification agents. To pass a DV++ validation domain owners must prove their ownership to a majority of the agents in a fully automated manner by responding to queries sent by the agents for the resource records in the domain. The agents are synchronised with an orchestrator module. The orchestrator is located on the CA network. The components of DV++ with the messages exchange are illustrated in Figure 9.

The orchestrator and the agents use HTTPS for their communication. During the domain validation process, all the agents receive from the orchestrator the domain and the record that need to be

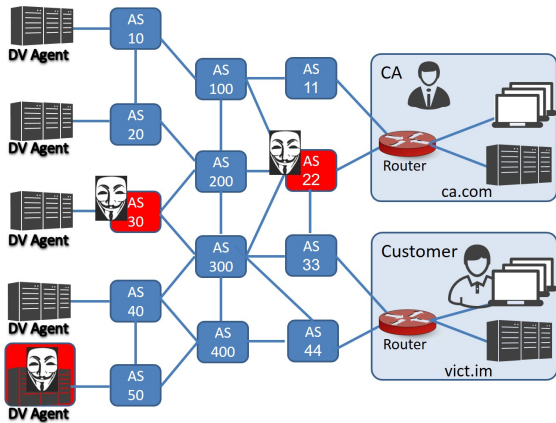


Figure 8: DV++ setup and attacker model.

queried. The agents send DNS requests to the nameservers in the domain requesting the record. As soon as the response arrives, the agent sends the response to the orchestrator. When more than 50% of the responses arrive, and they match, the orchestrator returns *success* otherwise *failure*. The number of the correct responses from the agents is a parameter that the CA can configure.

When sending a DNS request, each agent selects independently a source port at random as well as a random TXID. To launch a successful attack the attacker has to spoof correct responses to more than 50% of the agents. This is an impossible task even for strong, nation state attackers.

The agents are configured on different physical availability networks of AWS cloud. The selection of the cloud networks is done so that the agents are located in different networks, whose routes do not intersect. For selecting the networks to place the agents we use the empirically derived CAIDA AS-level graph [4] from 2016.

Similarly to DV, DV++ authentication is initiated by the CA following a submission of a CSR by the applicant. During this process, queries are sent to the agents, that perform look-ups in the target domain. Once majority of the responses are received by the orchestrator, they are processed.

The core idea of DV+ is that, even if the attacker can corrupt some agents, controls some networks or path, it cannot control all the paths and all the networks in the Internet and cannot corrupt all the agents. For instance, even strong nation state attackers, such as security agencies, do not control all Internet networks and paths.

4.3 Implementation

The orchestrator and the agents are written in Go. This ensures good performance, easy configuration and cross-compilation of static executables. Static executables allow easy deployment without the need to install any runtimes or libraries. Integrating DV++ in CAs current infrastructure does not require replacing software or hardware nor any other modifications: DV++ uses the same interface and interaction as DV. To use DV++ a CA should merely configure the system to make a call to DV++ library instead of DV.

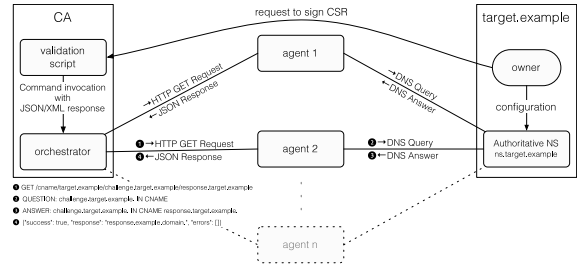


Figure 9: DV++ components and messages exchange.

The code, as well as configuration and execution instructions, are provided on GitHub <https://github.com/dvpp/dvpp>.

Using Our Setup. We have set up a pilot installation for evaluating our DV++ implementation. This can be used by the CAs, systems or users in the Internet, for verifying DNS responses and authenticating domain ownerships. We prepared a zip file containing everything needed to perform a DNS-based domain validation with a CNAME record. The zip file is available at <http://45.76.90.74/orchestrator.zip>, it contains a configuration file to use the local orchestrator with our agents. To get started, first extract the zip file and enter the extracted orchestrator directory with your terminal. Three parameters should be provided: (1) the domain requesting the certificate, (2) the CNAME to look up and (3) the expected response. For instance, to request a certificate for domain `example.com` with verification of `abcdef.example.com` that resolves to `ghijkl.example.com` the user should invoke (in one line without a linebreak):

```
./orchestrator_<platform>_<architecture> cname
example.com. abcdef.example.com. ghijkl.example.com
```

where `platform` is either Windows, Linux, or Darwin and `architecture` is either 386 or amd64. The command line tool will then return the status of the verification as a JSON object. If XML output is preferred a `-x` can be written in front of the CNAME.

4.4 Security Evaluations

We consider an attacker that tries to pass the authentication of DV++ to issue a fraudulent certificate. To succeed the attacker must provide correct spoofed responses for the majority of the DNS requests issued by the DV++ agents.

If the attacker is located on a network of the victim nameserver, it can hijack requests on the network of the nameserver and send spoofed responses to the DV++ agents. Luckily domains have more than one nameserver and the nameservers are placed in different networks; this is following best practices to avoid a single point of failure for domains. Our measurements of 1M-top Alexa domains show that an average number of nameservers per domain is 5, and minimal is 2. Furthermore, the nameservers belonging to the same domain are hosted in different networks. This ensures that the attacker cannot hijack and replace responses from all the nameservers.

We next evaluate attacks by passive MitM attacker, that controls a large ISP, and by an active attacker, that also additionally attempts to attract traffic from other networks. We run simulations

using a different number of agents, and demonstrate that even 3 agents in networks of top-tier ISPs suffice to provide strong security guarantees against MitM attackers.

4.4.1 MitM Attacker. We quantify the ability of an on-path attacker to intercept majority of the DNS requests sent from the agents to the victim domain. We run simulations using Alexa nameservers in 1000-top domains and quantify over different attacker-agents pairs and the fraction of nameservers whose BGP routes to the victim traverse the attacker. We run simulations with the BGP route computations presented in [25, 26] to the empirically derived CAIDA AS-level graph [4] from 2016. The graph is annotated with bilateral business relationships. We average our measurements over 10^6 combinations of attacker-victim AS pairs, selecting them at random. To identify ASes that host the nameservers we mapped the IP addresses of the nameservers to AS numbers using RIPE <https://stat.ripe.net>.

The simulations evaluate all the possibilities for an on-path attacker to cover almost all the routes between the victim domain and the DV++ agents. We used the dataset from January 2018, which contained 60006 ASes. We categorise ASes into four classes: large ASes with more than 250 customers, medium ASes that have between 25 and 250 customers, small ASes with 1 to 25 customers and stub ASes that have no customers. AS graph has 56 large ASes, 615 medium ASes, 8329 small ASes and 50995 stub ASes. Our graph had in total 60006 nodes and 261340 edges.

In our simulation for each combination of attacker-victim ASes pairs, we measure the fraction of attacker-victim pairs in which the attacker can capture traffic from more than 50% of the DV++ agents. This would be a very strong attacker, nevertheless, we show that DV++ when using sufficient amount of agents ensures security. In practice, the attackers are of course much weaker. The simulations show that only 0.1% of the attackers can be on the path between the victim nameservers and 50% or more of the agents, and hence capture the traffic between the victim and the agents. No attacker can capture 50% of the traffic to the agents when one or more agents are in a large AS, and attackers that are small ISPs or stubs cannot launch successful attacks. We summarise the result of our simulation in Table 1, and plot the results in Figure 10. Figure 10 shows that even with agents in 3 vantage points located in either of top 10 ISPs, the probability of the attacker to launch a successful hijack of more than 50% of the agents drops to almost 0.

victim/attacker	Large	Medium	Small	Stub
Large	0.00	0.00	0.00	0.00
Medium	0.21	0.02	0.00	0.00
Small	0.34	0.12	0.00	0.00
Stub	0.52	0.07	0.00	0.00

Table 1: Evaluation of on-path attacker. % of attackers capturing traffic from $\geq 50\%$ agents

4.4.2 Hijacking Attacker. The Border Gateway Protocol (BGP) is vulnerable to prefix hijack attacks [1-3, 10]. In prefix hijacks, the attacker hijacks all the traffic of a victim network. We evaluate

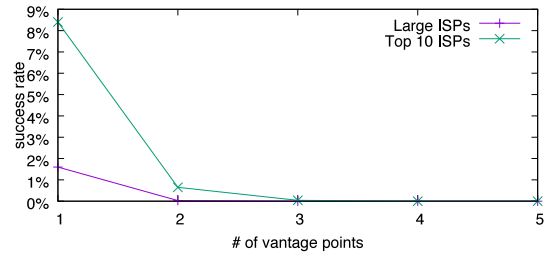


Figure 10: Passive Attacker. Large ISP with > 250 customers, and top ISP is tier 1 ISP.

attacker ability to exploit the insecurity in inter-domain routing protocol (BGP) to hijack traffic between the victim nameservers and the DV++ agents. In this case, both the victim nameservers and the attacker announce victims' BGP prefixes. We evaluate the fraction of agents that the attacker can attract. The simulation result in Figure 11 shows that probability that the attacker attracts more than 50% of the agents is 2%. The simulation also shows that the attackers that can hijack traffic from 50% of the agents, would disconnect the victim from the rest of the Internet. Only 0.20% of the attackers can successfully launch the attack while maintaining their route to the victim in order to relay packets between the victim nameservers and the rest of the Internet while avoiding detection. This is due to the fact that the fraction of agent nodes that the attacker hijacks is close to the fraction of the ASes in the Internet that the attacker attracts when announcing the victim's prefix.

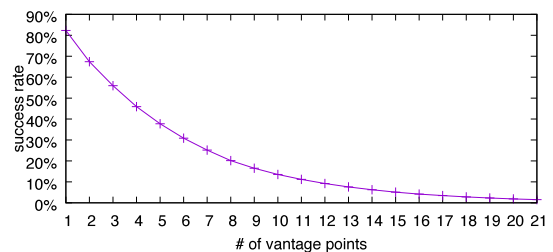


Figure 11: Simulations with active BGP prefix hijacking attacker.

4.5 Other Applications of DV++

DV++ can be used to bootstrap other mechanisms with security. For instance, our attacks apply against password recovery procedures in popular web services. In password recovery procedure the password or a link to reset the password is sent back to the email that initiated the password recovery. If the DNS resolver on the network of the service is attacked, and caches an incorrect record (mapping the email of the victim to attacker's controlled IP address) the password will be sent to the attacker.

DV++ can be used by the web services, to validate the DNS record of the email requesting the password recovery, hence blocking malicious requests that do not pass the verification of DV++.

5 RELATED WORK

5.1 DNS Security

Although researchers warned already in late 90s against the potential vulnerabilities to cache poisoning, DNS cache poisoning was first demonstrated in 2008 [36] against DNS resolvers that used fixed or incrementing source ports. Subsequently, [RFC5452] provided recommendations for patches, which included randomising the source ports in DNS requests, selecting the nameservers to which the requests are sent at random, and other patches. Nevertheless, shortly after attacks were found allowing to circumvent the [RFC5452] patches. Initially, [30, 32] showed that despite randomisation the source ports can be often predicted using side channels, such as timing. Followup works applied fragmentation for bypassing the recommendations in [RFC5452], [31, 33, 45].

However, predicting the source ports and other defences in [RFC5452] does not suffice for guaranteeing successful cache poisoning attacks. In particular, when the records are already in the cache, the DNS resolvers would often ignore and discard the DNS records which contain new values. To gain insights into caches behaviour [40, 48] performed a study of Which resolvers overwrite cached records with new values and under what conditions. Often, DNS resolution platforms consist of multiple actors and caches, these were studied in [39, 43] and multiple forwarders in DNS resolution chains which also expose to cache poisoning attacks [46].

DNSSEC [RFC4033-RFC4035] would prevent the cache poisoning attacks, however, recently vulnerabilities and misconfigurations were found in DNSSEC keys generation and management procedures exposing a large fraction of signed domains (more than 35%) to attacks, [16, 17, 47]. Furthermore, as [28] showed DNSSEC does not prevent attacks which replay DNSSEC-signed DNS response to redirect clients to incorrect servers in Content Distribution Networks (CDNs).

5.2 CA Compromises

A valid certificate signed by any of the trusted CAs in the browser, is accepted by the browsers. This property has spurred a proliferating market for certificates, it also has devastating implications for security: *any vulnerable or compromised CA can subvert the security of any domain.*

Indeed, there is a long history of attacks against the CAs. The compromise of small Dutch CA DigiNotar CA in 2011 was a critical point in history of PKI. For the first time a CA was removed from browser root stores because of poor infrastructure control and issuance of spoofed certificates. Subsequent works documented loopholes in PKI ecosystem, [7, 20, 34]. Other large CAs such as Comodo and Verisign have experienced breaches as well. In 2016 Chinese CA Wosign 'mistakenly' gave out certificates for GitHub domains due to a loophole in domain validation process. Specifically, if a user could prove ownership of a subdomain the CA would also issue certificate for the parent domain without applying DV to prove ownership of the base domain. In 2015 Comodo issued a certificate to an unauthorised party for live.fi – domain used by Microsoft to provide free email. This would have caused users to leak their credentials. Recently a Egyptian ISP (MCS holdings) obtained a certificate that was signed by CNNIC, that was included in root

stores. Namely any certificate issued by MCS would be accepted as valid (even for domains MCS did not own). In 2008, a bug in Debian OpenSSL caused thousands of certificates to be issued for keys with only 15-17 bits of entropy [22]. In addition, many browsers accept (non-root) certificates for 1024-bit RSA keys, which are believed to be broken by nation state attackers [11].

The research community also pointed out that Border Gateway Protocol (BGP) prefix hijacking can be leveraged to bypass domain validation (DV), see poster [13].

5.3 PKI Defences and Alternative Proposals

Following CA compromises and the risk of MitM attacks, new security mechanisms have been added to SSL/TLS, and the PKI. These include certificate transparency (CT), HSTS and HPKP headers, SCSV for protection against downgrade attacks. A taxonomy in [12] lists the proposals for improving the security of PKI infrastructure. We briefly recap here for completeness.

There are attempts to create a new alternative PKI and proposals to use additional entities for storing and checking certificates. Some proposals are aimed at making compromises visible using log servers to monitor CAs behaviour, for instance, Certificate Transparency (CT) [42], Sovereign Keys [21], Accountable Key Infrastructure [38], Attack-Resilient Public Key Infrastructure (ARPKI) [12], DTKI [5, 15, 49, 50]. DNS-Based Authentication of Named Entities (DANE) [RFC7671] uses DNSSEC [RFC4033-RFC4035] to list trusted CAs for issuing domains certificates. Similarly DNS Certificate Authority Authentication (CAA) Resource Record [RFC6844] uses DNS to list acceptable CAs. The goal of these proposals is to replace or complement the existing certificates issued by CAs trusted by the browsers. The proposals provide a good starting point and promising options for design of future PKI. However, due to their complexity and efficiency overhead as well as the changes that they require to the existing infrastructure and the deployment overhead (e.g., introduce multiple interacting entities), most are not adopted. Since March 2016, the Chrome Browser uses CT and ceased displaying the green bar for EV certificates that are not registered in a log server.

After making our DV++ available in March 2017, a parallel similar direction was proposed by LetsEncrypt, called `multi-VA`. The difference is that in contrast to DV++, `multi-VA` uses fixed nodes (currently three). Which it uses to perform the validation. By corrupting the nodes, the attacker can subvert the security of `multi-VA` mechanism. DV++ selects the nodes at random from a large set. Furthermore, we ensure that the nodes' placement guarantees that the nodes are not all located in the same autonomous system (AS) and the paths between the nodes and that the validated domain servers do not overlap.

6 CONCLUSIONS

Automated, efficient and easy-to-use procedures pave success for many mechanisms and DV is no exception to it. Unfortunately, the benefits also come with reduced security due to reliance on an insecure DNS. This makes DV vulnerable to off-path attacks, allowing even very weak attackers to issue fraudulent certificates for domains they do not own. We demonstrate such attacks.

Since cryptography for DNS is far from being sufficiently deployed, we need another way to bootstrap security. The only alternative is utilising the distributed nature of the Internet and making structural assumptions about the attacker's capabilities. Based on this observation we design and implement DV++: an innovative distributed mechanism which comes with the same benefits as DV while providing resilience even against strong MitM attackers.

7 ACKNOWLEDGEMENTS

The research reported in this paper was supported in part by the German Federal Ministry of Education and Research (BMBF), by the Hessian Ministry of Science and the Arts within CRISP (www.crisp-da.de/) and co-funded by the DFG as part of project S3 within the CRC 1119 CROSSING.

REFERENCES

- [1] [n. d.]. Hijack Event Today by Indosat. <http://www.bgpmo.net/hijack-event-today-by-indosat>. ([n. d.]).
- [2] [n. d.]. New Threat: Targeted Internet Traffic Misdirection. <http://www.renysys.com/2013/11/mitm-internet-hijacking>. ([n. d.]).
- [3] 2008. Renesys Blog - Pakistan Hijacks YouTube. http://www.renysys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtml. (Feb. 2008).
- [4] 2011. The CAIDA AS Relationships Dataset, 2011. <http://www.caida.org/data/active/as-relationships/>. (2011).
- [5] Martin Abadi, Andrew Birrell, Ilya Mironov, Ted Wobber, and Yinglian Xie. 2013. Global Authentication in an Untrustworthy World. In *HotOS*.
- [6] Nadhem J Al Fardan and Kenneth G Paterson. 2013. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 526–540.
- [7] Bernhard Amann, Matthias Vallentini, Seth Hall, and Robin Sommer. 2012. Extracting certificates from live traffic: A near real-time SSL notary service. *Technical Report TR-12-014* (2012).
- [8] Daniel Anderson. 2012. Splinternet Behind the Great Firewall of China. *Queue* 10, 11 (2012), 40.
- [9] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J Alex Halderman, Viktor Dukhovni, et al. 2016. DROWN: Breaking TLS Using SSLv2.. In *USENIX Security Symposium*. 689–706.
- [10] Hitesh Ballani, Paul Francis, and Xinyang Zhang. 2007. A Study of Prefix Hijacking and Interception in the Internet. (2007), 265–276 pages. <https://doi.org/10.1145/1282380.1282411>
- [11] E. Barker and A. Roginsky. 2011. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. NIST Special Publication. (2011).
- [12] David Basin, Cas Cremers, Tiffany Hyuni-jin, Adrian Perrig, Ralf Sasse, and Pawel Szalachowski. 2016. Design, Analysis, and Implementation of ARPKI: an Attack-Resilient Public-Key Infrastructure. *IEEE Transactions on Dependable and Secure Computing* (2016).
- [13] Henry Birge-Lee, Yixin Sun, Annie Edmundson, Jennifer Rexford, and Prateek Mittal. 2017. Using BGP to acquire bogus TLS certificates. *HotPETS&AZ17* (2017).
- [14] CAIDA. [n. d.]. Anonymized Internet Traces Dataset. ([n. d.]).
- [15] Vincent Cheval, Mark Ryan, and Jiangshan Yu. 2014. DTKI: a new formalized PKI with no trusted parties. *arXiv preprint arXiv:1408.1023* (2014).
- [16] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. A longitudinal, end-to-end view of the DNSSEC ecosystem. In *USENIX Security*.
- [17] Tianxiang Dai, Haya Shulman, and Michael Waidner. 2016. DNSSEC Misconfigurations in Popular Domains. In *International Conference on Cryptology and Network Security*. Springer, 651–660.
- [18] Deploy260. 2014. Email Hijacking. <https://www.internetsociety.org/blog/2014/09/email-hijacking-new-research-shows-why-we-need-dnssec-now/>. (2014).
- [19] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. 1993. Perfectly Secure Message Transmission. *J. ACM* 40, 1 (Jan. 1993), 17–47. <https://doi.org/10.1145/138027.138036>
- [20] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2013. ZMap: Fast Internet-wide Scanning and Its Security Applications.. In *USENIX Security Symposium*, Vol. 8. 47–53.
- [21] Peter Eckersley. 2011. Sovereign keys: A proposal to make https and email more secure. *Electronic Frontier Foundation* 18 (2011).
- [22] P. Eckersley and J. Burns. 2010. An observatory for the SSLiverse. *DEFCON'18*. (2010).
- [23] Hongyu Gao, Vinod Yegneswaran, Yan Chen, Phillip Porras, Shalini Ghosh, Jian Jiang, and Haixin Duan. 2013. An empirical reexamination of global DNS behavior. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 267–278.
- [24] Yossi Gilad and Amir Herzberg. 2013. Fragmentation Considered Vulnerable. *ACM Transactions on Information and System Security (TISSEC)* 15, 4 (April 2013), 16:1–16:31. A preliminary version appeared in WOOT 2011.
- [25] Phillipa Gill, Michael Schapira, and Sharon Goldberg. 2011. Let the market drive deployment: a strategy for transitioning to BGP security. In *SIGCOMM*, Srinivasan Keshav, Jörg Liebeherr, John W. Byers, and Jeffrey C. Mogul (Eds.). ACM, 14–25. <http://doi.acm.org/10.1145/2018436.2018439>
- [26] Phillipa Gill, Michael Schapira, and Sharon Goldberg. 2012. Modeling on quicksand: Dealing with the scarcity of ground truth in interdomain routing data. *ACM SIGCOMM Computer Communication Review* 42, 1 (2012), 40–46.
- [27] Matthias Gohring, Haya Shulman, and Michael Waidner. 2018. Path MTU Discovery Considered Harmful. In *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*. 866–874.
- [28] Shuai Hao, Yubao Zhang, Haining Wang, and Angelos Stavrou. 2018. End-Users Get Maneuvered: Empirical Analysis of Redirection Hijacking in Content Delivery Networks. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association.
- [29] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2012. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*. 205–220.
- [30] Amir Herzberg and Haya Shulman. 2012. Security of Patched DNS. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*. 271–288.
- [31] Amir Herzberg and Haya Shulman. 2013. Fragmentation considered poisonous, or: One-domain-to-rule-them-all. org. In *Communications and Network Security (CNS), 2013 IEEE Conference on*. IEEE, 224–232.
- [32] Amir Herzberg and Haya Shulman. 2013. Socket Overloading for Fun and Cache Poisoning. In *ACM Annual Computer Security Applications Conference (ACM ACSAC), New Orleans, Louisiana, U.S., Charles N. Payne Jr. (Ed.)*.
- [33] Amir Herzberg and Haya Shulman. 2013. Vulnerable Delegation of DNS Resolution. In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*. 219–236. https://doi.org/10.1007/978-3-642-40203-6_13
- [34] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. 2011. The SSL landscape: a thorough analysis of the x. 509 PKI using active and passive measurements. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 427–444.
- [35] Margaret Hu. 2015. Taxonomy of the Snowden Disclosures. *Wash & Lee L. Rev.* 72 (2015), 1679–1989.
- [36] Dan Kaminsky. 2008. It's the End of the Cache As We Know It. In *Black Hat conference*. <http://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf>.
- [37] Kernel.org. 2011. Linux Kernel Documentation. <http://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt>. (2011).
- [38] Tiffany Hyun-Jin Kim, Lin-Shung Huang, Adrian Perrig, Collin Jackson, and Virgil Gligor. 2013. Accountable key infrastructure (AKI): a proposal for a public-key validation infrastructure. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 679–690.
- [39] Amit Klein, Haya Shulman, and Michael Waidner. 2017. Counting in the Dark: Caches Discovery and Enumeration in the Internet. In *The 47th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.
- [40] Amit Klein, Haya Shulman, and Michael Waidner. 2017. Internet-Wide Study of DNS Cache Injections. In *INFOCOM*.
- [41] Jeffrey Knockel and Jedidiah R Crandall. 2014. Counting Packets Sent Between Arbitrary Internet Hosts. In *FOCI*.
- [42] Ben Laurie, Adam Langley, and Emilia Kasper. 2013. *Certificate transparency*. Technical Report.
- [43] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. 2013. On measuring the client-side DNS infrastructure. In *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 77–90.
- [44] Sharon Goldberg. 2018. The myetherwallet.com hijack and why it's risky to hold cryptocurrency in a webapp. <https://medium.com/@goldbe/the-myetherwallet-com-hijack-and-why-its-risky-to-hold-cryptocurrency-in-a-webapp-261131fad278>. (2018).
- [45] Haya Shulman and Michael Waidner. 2014. Fragmentation Considered Leaking: Port Inference for DNS Poisoning. In *Applied Cryptography and Network Security (ACNS), Lausanne, Switzerland*. Springer.
- [46] Haya Shulman and Michael Waidner. 2015. Towards Security of Internet Naming Infrastructure. In *European Symposium on Research in Computer*

- Security*. Springer, 3–22.
- [47] Haya Shulman and Michael Waidner. 2017. One Key to Sign Them All Considered Vulnerable: Evaluation of DNSSEC in the Internet. In *NSDI*. 131–144.
- [48] Soel Son and Vitaly Shmatikov. 2010. The hitchhiker's guide to DNS cache poisoning. In *Security and Privacy in Communication Networks*. Springer, 466–483.
- [49] Pawel Szalachowski, Stephanos Matsumoto, and Adrian Perrig. 2014. PoliCert: Secure and flexible TLS certificate management. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 406–417.
- [50] Dan Wendlandt, David G Andersen, and Adrian Perrig. 2008. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In *USENIX Annual Technical Conference*, Vol. 8. 321–334.
- [51] Hao Yang, Eric Osterweil, Dan Massey, Songwu Lu, and Lixia Zhang. 2011. Deploying cryptography in Internet-scale systems: A case study on DNSSEC. *Dependable and Secure Computing, IEEE Transactions on* 8, 5 (2011), 656–669.

A PKI AND DNS SECURITY

A.1 Public Key Infrastructure

Public Key Infrastructure (PKI) is a comprehensive system to provide public-key encryption and digital signature services. It includes a set of roles, policies and procedures to manage digital certificates and public-key encryption. It is used when there is a strict requirement to confirm the identity of parties involved in the communication or to validate the information transferred. The basic concept behind PKI is to bind public keys with respective identities of entities such as persons or organizations. To establish the binding, an entity should go through a process of registration and verification at and by a certificate authority (CA). Since there are different levels of binding, the process may vary, which can be fully automated or under human supervision. With a complete process of registration and verification, the CA would conclude the binding by issuing a certificate, which confirms the identity of the entity bound with the public key. Registration Authority (RA) accepts requests for digital certificates and authenticates the requesting entity. However, they do not actually sign the certificate that is issued. They would forward the certificate issuing to a CA after assuring valid and correct registration. After the issuance of a certificate, CA should publish it so that applications can retrieve it on behalf of users. Directory systems that are LDAP (Lightweight Directory Access Protocol)-compliant are believed to be the best technology for certificate repositories. Directories may be made publicly available or they may be private to a specific organization. Prior to each use of a certificate, the revocation status of a certificate must be checked. As a result, a PKI must incorporate a scalable certificate revocation system. The CA must securely publish the status of each certificate, while application software must verify the revocation information prior to each use of a certificate. Certificate Revocation List (CRL) is a popular way to maintain revocation information, which contains a list of digital certificates that have been revoked. CRLs can be published to a directory system by CAs. An alternative to CRLs is the certificate validation protocol, Online Certificate Status Protocol (OCSP), where a responder answers queries about the revocation information of a requested certificate.

A.2 Domain Name System (DNS)

Domain Name System (DNS), [RFC1034, RFC1035], is a distributed database containing mappings for resources (also called *resource records (RRs)*), from *domain names* to different values. The most

popular and widely used mappings, [23], are for IP addresses, represented by A type RRs, that map a domain name to its IPv4 address, and name servers, represented by NS type RRs, that map a name server to domain name. The resource records in DNS correspond to the different services run by the organisations and networks, e.g., hosts, servers, network blocks.

DNS is a client-server protocol, used by the resolvers to retrieve RRs stored in the zone files maintained by the name servers. The resolvers communicate to the name servers using a simple request-response protocol (typically over UDP); for instance, (abstracting out details) to translate `www.foo.bar` resolvers locate the name server `ns.foo.bar`, authoritative for `foo.bar`, and obtain the IP address of the machine hosting the web server of the website `www.foo.bar`. Resolvers store the DNS records, returned in responses, in their caches for the duration indicated in the Time To Live (TTL) field of each record set.

The zones are structured hierarchically, with the root zone at the first level, Top Level Domains (TLDs) at the second level, and millions of Second Level Domains (SLDs) at the third level. The IP addresses of the 13 root servers are provided via the *hints* file, or compiled into DNS resolvers software and when a resolver's cache is empty, every resolution process starts at the root. According to the query in the DNS request, the root name server redirects the resolver, via a referral response type, to a corresponding TLD, under which the requested resource is located. There are a number of TLDs types, most notably: *country code TLD (ccTLD)*, which domains are (typically) assigned to countries, e.g., `us`, `il`, `de`, and *generic TLD (gTLD)*, whose domains are used by organisations, e.g., `com`, `org`, and also by US government and military, e.g., `gov`, `mil`. Domains in SLDs can also be used to further delegate subdomains to other entities, or can be directly managed by the organisations, e.g., as in the case of `ibm.com`, `google.com`.

A.3 DNS Cache Poisoning and Defences

In the course of a DNS cache poisoning attack, the attacker sends spoofed DNS responses impersonating a real nameserver. The responses contain malicious DNS records, pointing legitimate services at incorrect addresses or names. If a victim DNS resolver accepts and caches the responses, it will redirect the services or clients using it to incorrect hosts. DNS cache poisoning expose to malware distribution, credentials theft, and can be leveraged for censorship [8] or for surveillance [35], as well as for financial gain by cyber criminals. The poisoned DNS responses are sent from a spoofed IP address (impersonating a legitimate nameserver) to the victim DNS resolver. Prior to accepting poisoned responses, the DNS resolvers validate that the responses contain the same source port and transaction identifier (TXID) as were in the corresponding DNS request. In accordance with the standard best practices [RFC5452] the DNS resolvers should select source ports and TXIDs at random, hence spoofing the correct responses, for attackers that do not see the DNS request packets, is not a simple challenge – the attacker has a $\frac{1}{2^{32}}$ success chance.

Following Kaminsky's cache poisoning attack, DNS resolvers were quickly patched to support challenge-response defences against cache poisoning. Most existing challenge-response mechanisms, [RFC5452], are 'patches', randomising and validating existing fields in the TCP/IP protocols. We next review standardised and

Test	DNS Values in DNS fields in spoofed response Fiel	Overrides cached record
1	Q A?two.test-ns0.cache.example An two.test-ns0.cache.example A Au test-ns0.cache.example NS ns2.test-ns0.cache.example Ad ns2.test-ns0.cache.example A 6.6.6.6	test-ns0.cache.example NS ns.test-ns0.cache.example
2	Q A? two.test-ns0-auth.cache.example An two.test-ns0-auth.cache.example A 2.2.2.2 Au test-ns0-auth.cache.example NS ns2.test-ns0-auth.cache.example Ad ns2.test-ns0-auth.cache.example A 6.6.6.6	test-ns0-auth.cache.example NS ns.test-ns0-auth.cache.example
3	Q A? two.test-ns2.cache.example An two.test-ns2.cache.example A 2.2.2.2 Au test-ns2.cache.example NS ns2.test-ns2-sub.cache.example Ad —	test-ns2.cache.example NS ns.test-ns2.cache.example
4	Q A? two.test-ns2-auth.cache.example An two.test-ns2-auth.cache.example A 2.2.2.2 Au test-ns2-auth.cache.example NS ns2.test-ns2-auth-sub.cache.example Ad —	test-ns2-auth.cache.example NS ns.test-ns2-auth.cache.example
5	Q A? two.test-b4.cache.example An — Au sub.test-b4.cache.example NS ns.test-b4.cache.example Ad ns.test-b4.cache.example A 6.6.6.6	ns.test-b4.cache.example A 2.2.2.2
6	Q A? two.test-u1-auth.cache.example An — Au test-u1-auth.cache.example NS ns2.test-u1-auth.cache.example Ad ns2.test-u1-auth.cache.example A 6.6.6.6	test-u1-auth.cache.example NS ns.test-u1-auth.cache.example
7	Q A? two.test-u3-3.cache.example An — Au test-u3-3.cache.example NS ns.test-u3-3.cache.example Ad ns.test-u3-3.cache.example A 6.6.6.6	ns.test-u3-3.cache.example A 2.2.2.2
8	Q A? two.sub.test-u3-4.cache.example An — Au sub.test-u3-4.cache.example NS ns.test-u3-4.cache.example Ad ns.test-u3-4.cache.example A 6.6.6.6	ns.test-u3-4.cache.example A 2.2.2.2
9	Q A? zwei.one1.test-w11.cache.example An — Au one1.test-w11.cache.example NS ns2.test-w11-sub.cache.example Ad —	one1.test-w11.cache.example NS ns.one1.test-w11.cache.example
10	Q A? zwei.one1.test-w11bis.cache.example An — Au one1.test-w11bis.cache.example NS ns2.test-w11bis.cache.example ns2.test-w11bis.cache.example A 6.6.6.6	one1.test-w11bis.cache.example NS ns.one1.test-w11bis.cache.example
11	Q A? two.test-u3-2.cache.example An two.test-u3-2.cache.example A 2.2.2.2 Au test-u3-2.cache.example NS ns.test-u3-2.cache.example Ad ns.test-u3-2.cache.example A 6.6.6.6	ns.test-u3-2.cache.example A 2.2.2.2

Figure 12: Selected payloads for cache overwriting; see full list in [40]. Spoofed DNS records are marked in red.

Payloads	BIND 9.10.2-P2	BIND 9.4.1	Unbound 1.5.4	MaraDNS 3.2.07 Deadwood	PowerDNS 3.7.3	MS DNS 6.1 Win Server'08 R2 6.1.7601	MS DNS 6.2 Win Server'12 6.2.9200	MS DNS 6.3 Win Server'12 R2 6.3.9600	Google Public DNS	Open DNS	BIND 9.10.2-P2 w/DNSSEC	Nominum Vantio CacheServe v5	Nominum Vantio CacheServe v7
1	0	1	1	0	1	1	1	1	0	1	0	0	0
2	0	1	1	0	1	0	0	0	0	0	0	0	0
3	1	1	1	0	1	1	1	1	0	1	0	0	0
4	1	1	1	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	1	1	1	1	0	1	0	0	0
6	0	0	1	0	1	0	0	0	0	0	0	0	0
7	0	0	1	0	1	1	1	1	0	1	0	0	0
8	0	0	1	0	1	1	1	1	0	0	0	0	0
9	1	1	1	0	1	1	1	1	0	1	1	0	0
10	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 13: Evaluation of selected payloads against popular DNS software, appliances and public services; see full list in [40].

most commonly used challenge-response authentication mechanisms.

DNS uses a random 16-bit TXID (transaction identifier) field that associates a DNS response with its corresponding request. DNS implementations additionally support a random selection of name servers each time they send a request. The main defence, that makes poisoning impractical is a (16-bit) source port randomisation recommended in [RFC5452], which together with a TXID result in a search space containing 232 possible values; a source port identifies the client side application in requests, and is echoed (as a destination port) in responses. Specific recommendations for port randomisation algorithms were recently provided in [RFC6056]. Due to the significance of port randomisation for preventing off-path attacks, e.g., cache poisoning and injections into TCP, multiple studies were conducted to measure support of port randomisation in the Internet,

and it seems that many resolvers adopted port randomisation methods that were recommended in [RFC6056]. Currently the security of most DNS resolvers relies on these challenge-response mechanisms. However, relying only on TXID and source port randomisation is not believed to be sufficient against cache poisoning.

To mitigate the DNS cache poisoning attacks, the IETF designed and standardised Domain Name System Security Extensions (DNSSEC) [RFC4033-RFC4035]. Unfortunately DNSSEC requires significant changes to the DNS infrastructure as well as to the protocol, and although proposed and standardised already in 1997, it is still not widely deployed. The low adoption of DNSSEC in tandem with the recent wave of cache poisoning vulnerabilities and evidence for DNS injections in the Internet stimulated efforts within the operational and research communities to standardise alternative *easy to adopt* defences.

A.3. Poster: Off-path Attacks Against PKI

[Dai18p]

Tianxiang Dai, Haya Shulman, and Michael Waidner. “Poster: Off-Path Attacks Against PKI”. in: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 2213–2215. ISBN: 9781450356930. DOI: 10.1145/3243734.3278516. URL: <https://doi.org/10.1145/3243734.3278516>

Poster: Off-path Attacks Against PKI

Tianxiang Dai
Fraunhofer SIT

Haya Shulman
Fraunhofer SIT
TU Darmstadt

Michael Waidner
Fraunhofer SIT
TU Darmstadt

ABSTRACT

The security of Internet-based applications fundamentally relies on the trustworthiness of Certificate Authorities (CAs). We practically demonstrate for the first time that even a very weak attacker, namely, an *off-path* attacker, can effectively subvert the trustworthiness of popular commercially used CAs. We demonstrate an attack against one popular CA which uses Domain Validation (DV) for authenticating domain ownership. The attack exploits DNS Cache Poisoning and tricks the CA into issuing fraudulent certificates for domains the attacker does not legitimately own – namely certificates binding the attacker's public key to a victim domain.

1 INTRODUCTION

Public Key Infrastructure (PKI) is a fundamental mechanism that facilitates Internet security and bootstraps cryptographic mechanisms. In a nutshell PKI provides procedures and technology for creating and managing digital certificates. Certificates bind entities, such as digital resources, to cryptographic keys. These keys can then be used to establish security, e.g., authentication or encryption, with Internet destinations.

Over the years various PKI models were proposed, designed and standardised. The main difference between the models is in the entities that can be certified, currently these include: clients, devices and digital resources. A challenge common to all the different PKIs is the authentication of the entity that is to be certified.

In this work, we focus on the most widely deployed and used PKI in the Internet: the web PKI, which certifies ownership over domain names. We explore the security of the authentication technique used by popular Certificate Authorities (CAs) to establish ownership over domain names during the certificates issuance procedure. Correctly establishing ownership over domain names is critical for ensuring that services are trusted and communication to them is secure. Ultimately CAs vouch for trustworthiness of a service by issuing a digital certificate that binds a domain name to a public key of the service. The certificate contains, among others, the public key of the requesting server and the requested domain. Domain name within the certificate is a key element on which trust can be built. The certificate is signed by the private key of the CA. The server then uses this certificate to prove its identity to clients in the Internet. The clients use the key in the certificate to establish a secure (encrypted and authenticated) connection to the server.

To validate that the server issuing the request owns and controls the domain for which it requests the certificate most CAs use Domain Validation (DV). DV allows to prove in an automated way that the applicant owns a given domain name. The idea behind DV is that only the owner of the domain can receive the communication sent to the services in that domain and can respond to them.

In this work we build on the attack presented in [1] and prepare a real life demonstration of the attack against a popular and widely used CA with a victim domain controlled by us. Our demo is the first to weaponise DNS cache poisoning by off-path attackers in practice and to demonstrate it against a critical system such as the web PKI. Prior to our work, off-path DNS cache poisoning was considered a theoretical threat.

Our demo shows that although PKI is supposed to provide security against the strong Man-in-the-Middle (MitM) attackers, the widely supported DV mechanism, which underlies the security of certificates issued within the PKI, is vulnerable to even weak off-path attackers. We show that off-path attackers can trick the DV process and issue fraudulent certificates for domains they do not own.

Our attack uses DNS cache poisoning as a building block to inject incorrect mappings into the caches of CAs. These mappings map the target domain to attacker's controlled IP addresses. As a result, the CAs perform the DV process against attacker controlled hosts and not against the real owner of the domain. We show how to overcome all the challenges with an off-path attacker during the certificate issuance. Our results show that *Public Key Infrastructure (PKI)*¹ which is meant to provide security against strong Man-in-the-Middle (MitM) attackers, is relying on a weak building block that can be circumvented by an off-path attacker – and in this work we demonstrate this.

CA Compromises

There is a long history of attacks against the CAs. The compromise of small Dutch CA DigiNotar CA in 2011 was a critical point in history of PKI. Subsequent works documented loopholes in PKI ecosystem, [4]. In 2016 Chinese CA Wosign 'mistakenly' gave out certificates for GitHub domains due to a loophole in domain validation process. Recently a Egyptian ISP (MCS holdings) obtained a certificate that was signed by CNNIC, that was included in root stores. Namely any certificate issued by MCS would be accepted as valid (even for domains MCS did not own). In 2008, a bug in Debian OpenSSL caused thousands of certificates to be issued for keys with only 15-17 bits of entropy, [5].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CCS '18, October 15-19, 2018, Toronto, ON, Canada

© 2018 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5693-0/18/10.

<https://doi.org/10.1145/3243734.3278516>

¹PKI is a set of roles, policies, procedures and entities for creating and managing certificates and public-key encryption.

DNS Cache Poisoning

DNS cache poisoning was first demonstrated in 2008 [10] against DNS resolvers that used fixed or incrementing source ports. Subsequently, [RFC5452] provided recommendations for patches. Nevertheless, shortly after attacks were found allowing to circumvent the [RFC5452] patches. Initially, [6, 8] showed that despite randomisation the source ports can be often predicted using side channels, such as timing. Followup works applied fragmentation for bypassing the recommendations in [RFC5452], [7, 9, 11-13].

DNSSEC [RFC4033-RFC4035] would prevent the cache poisoning attacks, however, recently vulnerabilities and misconfigurations were found in DNSSEC keys generation and management procedures exposing a large fraction of signed domains (more than 35%) to attacks, [2, 3, 14].

2 WEAPONISING CACHE POISONING TO SUBVERT DV

In this section we demonstrate an off-path attack that allows an attacker to impersonate a victim domain to a CA and cause the CA to issue a spoofed (fraudulent) certificate binding the public key of the attacker to a victim domain. Our demonstration is based on the attack presented in [1], and describes the technical and practical challenges.

The attack leverages off-path DNS cache poisoning against a DNS resolver of the CA. During the attack we inject a spoofed DNS record mapping the victim domain to attacker controlled hosts, so that the DV checks, that the CA runs, are performed against the attacker's hosts.

For our demo we set up a victim domain at `victim-org.info`. We will inject spoofed records redirecting a nameserver and email server in this domain to attacker controlled IP addresses. The attack is initiated with a DNS request, which the CA should send to the victim. The attacker then injects a DNS response with malicious records from a spoofed IP address (impersonating a victim nameserver). If the CA's cache does not have the records of the victim domain in its DNS cache - injecting the DNS records is simple. Often, the victim's records would be present in the cache. In this case the cached records need to be overwritten with new values supplied by the attacker. We present demonstration of both cases (when the target records are cached and when they are not).

2.1 DNS Request

To prevent attacks against DNS resolvers the networks are configured to block external requests from the Internet and to serve only the requests that are originated by the internal clients or services. Since the off-path attacker is not located on the same network with the attacked CA, we need to use alternative techniques to initiate communication with the CA's DNS resolver.

We trigger DNS requests by utilising the CSR uploading form. This causes the CA to initiate a DV process, which triggers DNS requests for the domain that was provided in the CSR - this is the target domain.

The attack is initiated with a DNS request. To succeed in the attack, the attacker has to craft a correct DNS response before the authentic response from the real nameserver arrives. The attack leverages defragmentation cache poisoning, with a spoofed second

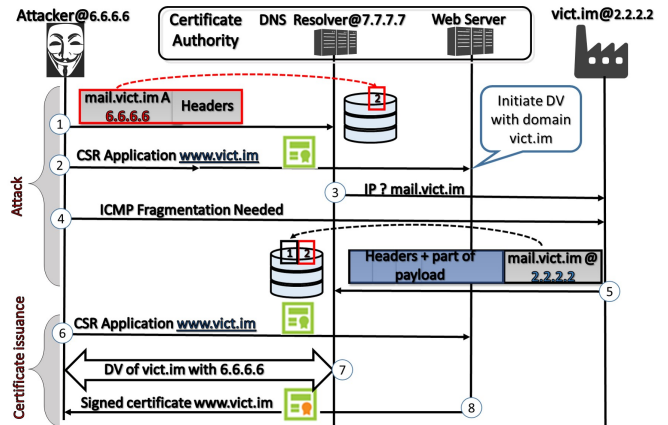


Figure 1: Defragmentation cache poisoning for subverting DV; in the illustration we use `vict.im` as victim domain.

fragment that contains malicious payload. Once the malicious second fragment is reassembled with the authentic first fragment, the payload is passed on to the DNS software. The software examines the records, and based on its caching logic determines whether to cache the records or to ignore them. We show how we launched the attack with the steps above against a real CA, which uses a patched Bind9 resolver: New BIND 9.x with DNSSEC.

2.2 Defragmentation Cache Poisoning

The idea of the attack is that the attacker "convinces" the (victim) nameserver to fragment responses to a specific destination. To cause fragmentation the attacker has to reduce the Maximum Transmission Unit (MTU) from the server to the CA. To do this our off-path attacker issues an ICMP fragmentation needed packet to the nameserver, indicating that it should reduce the MTU when sending packets to the *victim* resolver (of the CA).

After reducing the MTU we trick the receiving resolver into reassembling the first fragment of the real response from the nameserver with the second fragment generated by the attacker. This allows us to bypass the challenge-response authentication fields used by the DNS resolvers, since they are echoed by the nameserver in the *first fragment*.

We first perform a measurement and a calculation, needed for setting the values in the ICMP fragmentation needed error message in step (4) (Figure 1) and based on which the attacker constructs the spoofed second fragment in step (1) (Figure 1). The attacker measures the responses' sizes from the nameserver of `vict.im` and calculates the offset where the fragmentation should occur. The MTU in the ICMP fragmentation needed error message is set accordingly. The goal is to ensure that the records, which it will replace with spoofed records, are in the second fragment.

The attack proceeds as illustrated in Figure 1. In step (1) the attacker sends to the victim DNS resolver a spoofed second fragment. The fragment is cached by the receiving resolver in the IP defragmentation cache waiting for the remaining fragments to arrive. In step (2) the attacker uploads a CSR form requesting certification for victim domain. The attacker selects an email based DV, hence

in step (3) the DNS resolver at the CA issues a DNS request to the nameserver of victim domain asking for an IP address of the email server. The attacker issues an ICMP fragmentation needed packet in step (4) to ensure that the response is fragmented; this ICMP error message can be issued also before step (1). In step (5) the nameserver of the victim sends a fragmented DNS response. The first fragment contains the entropy (resolver's source port and TXID are copied by the nameserver to the response) and some of the DNS payload. The question, answer, authoritative and additional sections are split between the first and the second fragment. The first fragment is then reassembled with the spoofed second fragment that was already in the cache of the DNS resolver; they both leave the cache, and are passed on to the UDP layer. The legitimate second fragment will not have any first fragment to reassemble with, since when it arrives the legitimate first fragment has already been reassembled with the spoofed second fragment and left the buffer. The remaining steps (6)-(8) complete the DV process.

Notice that in step (1) when sending the spoofed fragment the attacker needs to ensure the correctness of UDP checksum and IP ID.

2.3 Overwriting Cached Records

Next, the task is to ensure that the records (sent in the spoofed DNS response) are cached and served in responses to applications and clients. Often the records will already be present in the cache. We use the techniques in [12] to overwrite the cached record with new values. The attack in our demo is launched against a patched new BIND 9.x with DNSSEC, hence we focus on the caching logic of BIND.

```
// CASE - NOT CACHED //
Query: MX? victim-org.info
Answer: victim-org.info MX exchanger0.victim-org.info
```

```
Query: A? exchanger0.victim-org.info
Answer: exchanger0.victim-org.info A 198.22.162.189
// Attacker changes this IP to 199.244.49.220
```

```
// CASE - CACHED //
Query: MX? victim-org.info
Answer: victim-org.info MX exchanger0.victim-org.info
```

```
Query: A? exchanger0.victim-org.info
Answer: exchanger0.victim-org.info A 198.22.162.189
Authority: ns0.victim-org.info NS ns0.victim-org.info
Additional: ns0.victim-org.info A 198.22.162.189
// Attacker changes last two records
// victim-org.info NS ns000.attacker.info
// ns000.attacker.info A 199.244.49.220
```

2.4 Validating Success

How can we know if the attack succeeded? An off-path attacker cannot measure a state change in a remote cache. We measure the success of the attack indirectly, by inspecting the records arriving at the attacker's host. Specifically, once the record with malicious mappings is injected into the cache of the CA's DNS resolver, the remainder of DV will be performed against attacker's IP addresses and not against the victim domain. Hence, by inspecting the queries at the attacker's host at IP 199.244.49.220 we identify when the attack is successful. Subsequently, after successfully passing the DV, the attacker will also receive the fraudulent certificate to its own email address.

3 CONCLUSIONS

We provide a real life proof of concept attack against PKI. Our attack is launched with the weak off-path attacker and utilises DNS cache poisoning to issue fraudulent certificates which the attacker does not own. After deployment of [RFC5452] recommendations in 2008 DNS cache poisoning was believed to be only a theoretical threat. This also reduced the motivation to adopt more systematic cryptographic protection with DNSSEC.

Our work is the first since 2008 to weaponise DNS cache poisoning with an off-path attacker, and to leverage cache poisoning against an application of interest - the web PKI. Our demonstration runs an attack against a large and popular CA. Although more popular CAs are vulnerable, e.g., see [1], we argue that even a single vulnerable CA (trusted by the browsers and operating systems) is sufficient for subverting web security with devastating consequences for Internet clients and services.

ACKNOWLEDGEMENTS

The research reported in this paper was supported in part by the German Federal Ministry of Education and Research (BMBF), by the Hessian Ministry of Science and the Arts within CRISP (www.crisp-da.de/) and co-funded by the DFG as part of project S3 within the CRC 1119 CROSSING.

REFERENCES

- [1] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. 2018. Domain Validation++ For MitM-Resilient PKI. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*.
- [2] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. A longitudinal, end-to-end view of the DNSSEC ecosystem. In *USENIX Security*.
- [3] Tianxiang Dai, Haya Shulman, and Michael Waidner. 2016. DNSSEC Misconfigurations in Popular Domains. In *International Conference on Cryptology and Network Security*. Springer, 651–660.
- [4] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2013. ZMap: Fast Internet-wide Scanning and Its Security Applications.. In *USENIX Security Symposium*, Vol. 8, 47–53.
- [5] P. Eckersley and J. Burns. 2010. An observatory for the SSLiverse. DEFCON'18. (2010).
- [6] Amir Herzberg and Haya Shulman. 2012. Security of Patched DNS. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*. 271–288.
- [7] Amir Herzberg and Haya Shulman. 2013. Fragmentation considered poisonous, or: One-domain-to-rule-them-all. org. In *Communications and Network Security (CNS), 2013 IEEE Conference on*. IEEE, 224–232.
- [8] Amir Herzberg and Haya Shulman. 2013. Socket Overloading for Fun and Cache Poisoning. In *ACM Annual Computer Security Applications Conference (ACM ACSAC), New Orleans, Louisiana, U.S., Charles N. Payne Jr. (Ed.)*.
- [9] Amir Herzberg and Haya Shulman. 2013. Vulnerable Delegation of DNS Resolution. In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*. 219–236. https://doi.org/10.1007/978-3-642-40203-6_13
- [10] Dan Kaminsky. 2008. It's the End of the Cache As We Know It. In *Black Hat conference*. <http://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf>.
- [11] Amit Klein, Haya Shulman, and Michael Waidner. 2017. Counting in the Dark: Caches Discovery and Enumeration in the Internet. In *The 47th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.
- [12] Amit Klein, Haya Shulman, and Michael Waidner. 2017. Internet-Wide Study of DNS Cache Injections. In *INFOCOM*.
- [13] Haya Shulman and Michael Waidner. 2014. Fragmentation Considered Leaking: Port Inference for DNS Poisoning. In *Applied Cryptography and Network Security (ACNS), Lausanne, Switzerland*. Springer.
- [14] Haya Shulman and Michael Waidner. 2017. One Key to Sign Them All Considered Vulnerable: Evaluation of DNSSEC in the Internet.. In *NSDI*. 131–144.

A.4. Poster: Fragmentation Attacks on DNS over TCP

[Dai21i1]

Tianxiang Dai, Haya Shulman, and Michael Waidner. “Poster: Fragmentation Attacks on DNS over TCP”. in: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. 2021, pp. 1124–1125. doi: 10.1109/ICDCS51616.2021.00118

Poster: Fragmentation Attacks on DNS over TCP

Tianxiang Dai*, Haya Shulman*, Michael Waidner*[†]

*Fraunhofer Institute for Secure Information Technology SIT, [†]Technical University of Darmstadt

Abstract—The research and operational community believe that TCP provides protection against IP fragmentation based attacks and recommend that servers avoid sending responses over UDP and use TCP instead.

In this work we show for the first time that IP fragmentation attacks may also apply to communication over TCP. We perform a study of the nameservers in the 100K-top Alexa domains and find that 454 domains are vulnerable to IP fragmentation attacks. Of these domains, we find 366 additional domains that are vulnerable only to IP fragmentation attacks on communication with TCP. We also find that the servers vulnerable to TCP fragmentation can be forced to fragment packets to much smaller sizes (of less than 292 bytes) than servers vulnerable to UDP fragmentation (not below 548 bytes). This makes the impact of the attacks against servers vulnerable to fragmentation of TCP segments much more detrimental. Our study not only shows that the recommendation to use TCP and avoid UDP is risky but it also shows that the attack surface due to fragmentation is larger than was previously believed.

We evaluate known IP fragmentation-based DNS cache poisoning attacks against DNS responses over TCP.

I. INTRODUCTION

IP fragmentation allows to adjust packets to the size supported by the networks which the packets traverse. Given a too large packet, the source or the routers fragment packets into smaller fragments. The receiver reassembles the fragments back into the original IP packets. To identify the fragments during reassembly that belong to the same IP packet the receiver uses a 16 bit IP identifier (IP ID) in the IP header: the operating system at the sender assigns an IP ID to every outbound IP packet.

Fragmentation-based attacks. There is a long history of attacks which exploit fragmentation to launch Denial of Service (DoS) [1], [2] or DNS cache poisoning attacks [3]–[5], [5]–[8]. The idea underlying the attacks is to send a spoofed fragment to the victim client, which when reassembled with the genuine fragment from the server, results either in an incorrect fragment which is discarded by the client (causing DoS attack against the target service) or results in a correct fragment which contains malicious payload that was carried in the spoofed fragment. Such attacks are typically launched against services that run over UDP, such as most of DNS traffic, or directly over IP, such as tunnelling mechanisms. Since most UDP traffic is not fragmented, the attackers can trigger source fragmentation by sending a spoofed ICMP fragmentation needed error message (type: 3, code: 4) to the target server.

TCP uses path MTU discovery. Communication over TCP avoids fragmentation since TCP performs path MTU discovery (PMTUD) and accordingly adjusts the Maximum Segment Size (MSS). To discover the MTU the sender transmits IP packets with a `do not fragment` (DF) bit set in the IP header. When a router with a smaller MTU receives such a packet it discards the packet and returns to the sender an ICMP fragmentation

needed error message. This causes the TCP at the sender to reduce the MSS to the value indicated in the MTU (minus 20 bytes) and resend the packet. The process iterates until the packet is received by the destination. As a result of PMTUD, services running over TCP are not subject to fragmentation and hence should not be vulnerable to fragmentation based attacks. Indeed, there are many recommendations to use TCP when possible in order to avoid (DNS cache poisoning and DoS) fragmentation based attacks [9], [10]. Many DNS servers are configured to avoid fragmentation by setting the TC bit on DNS responses, signalling to the client to resend the DNS request over TCP.

We show fragmentation-based attacks against TCP. In this work we show for the first time that off-path attackers can force the servers to fragment communication over TCP. We perform a measurement study of the vulnerable servers. In contrast to common belief that the TCP provides sufficient protection against off-path network adversaries we show that this is not so. We reveal that fragmentation based attacks against services over TCP allow to circumvent the entropy in TCP segments, including Sequence Number (SN) and source port. We explain the challenges that the network adversaries need to address to inject spoofed payload into TCP segments and demonstrate how forcing the servers to fragment traffic over TCP can be exploited for DNS cache poisoning attacks.

II. MEASURING TCP FRAGMENTATION

We perform a study of the Alexa Top-100K domains to infer the fraction of domains with nameservers that can be forced to fragment responses over TCP. For this, we send to the nameservers ICMP fragmentation needed packet with an TCP packet embedded, a UDP packet embedded and an echo reply. We then issue a DNS request over TCP to the nameserver, and check if the response arrives fragmented, and the minimum fragment size that we observed. The results of our measurements are listed in Table I.

Out of 100K domains, we collected nameservers in 97,493 domains (column ‘Total’ in Table I) by querying for NS record. ‘N/A’ indicates that we received no response. The valid responses are marked as ‘Checked’. If we are able to cause nameservers in a domain to fragment responses, we mark the domain as ‘Fragmented’. Column ‘TC/NEW’ has different meanings for UDP and TCP rows. For the UDP row, it says 13.43% domains have nameservers responding with TC (Truncated) bit set. For the TCP row, it shows the number of new domains that were identified as ‘Fragmented’, which do not fragment DNS responses over UDP. Our results show that 9752 domains are vulnerable only to fragmentation attacks against UDP and 366 domains are vulnerable only to fragmentation attacks against TCP. More importantly, among the servers that are vulnerable to our IP fragmentation attacks over TCP, we find servers

which actively avoid UDP by responding with a TC bit set and requesting that the DNS resolver re-sends the DNS request over TCP. Indeed, there are 74 of the 366 new domains which respond with TC bit but are vulnerable to our attacks. Namely, these servers implement the recommendations to move to TCP in order to avoid fragmentation based attacks to which communication over UDP is known to be vulnerable!

	Fragmented	Checked	N/A	TC/NEW	Total
UDP	10.09% 9840	87.51% 85320	12.49% 12173	13.43% 13093	100% 97493
TCP	0.47% 454	90.94% 88663	9.06% 8830	0.38% 366	100% 97493

TABLE I
FRAGMENTATION OF UDP AND TCP IN 100K-TOP ALEXA DOMAINS.

To infer the fragment sizes that the vulnerable nameservers are willing to reduce to, we evaluated ICMP error messages with varying MTU sizes. Surprisingly, we find that domains supporting fragmentation over TCP allow much smaller fragments than UDP. About 50% of the domains that allow TCP fragmentation, can be reduced to MTU of 292 bytes or smaller, while 90% of the domains vulnerable to UDP fragmentation limit the minimum fragment size to at least 548 bytes. Our findings are inline with the findings in [11], that analysed the CAIDA datasets between years 2009 and 2016 and found a large fraction of small fragments with TCP segments and UDP datagrams.

III. ATTACKS EXPLOITING TCP FRAGMENTATION

IP fragmentation attacks on TCP allow to bypass the randomisation with the sequence number and the source ports of TCP. To inject a spoofed fragment the attacker has to compute IP ID value in the original response from the nameserver.

Predicting IP ID. Our measurements of the 100K-top Alexa domains show that 76,267 of the nameservers use predictable IP ID values in IP packets that contain TCP segments. In contrast, when communicating over UDP only 12,540 of the servers use predictable IP ID values in outbound packets.

Attack evaluation. DoS attacks by fragment mis-association are simple: the attacker sends fragments that result in an invalid TCP segment that is discarded by the operating system. We evaluate the traditional DNS cache poisoning attacks that were previously shown to apply to UDP communication, against the servers that we found vulnerable to fragmentation attacks on TCP. The idea of the attacks is to inject a spoofed fragment (with a source IP address of the victim nameserver) that contains a malicious payload. We implement the DNS cache poisoning attack from [3] and inject DNS records in a second spoofed fragment, which is reassembled with the genuine first fragment from the nameserver. The resulting DNS response contains malicious records injected from the second spoofed fragment.

We simulate the success rate of the attack using an updated DNS resolver with an IP defragmentation cache size of 64 packets, for different IPID increment rates Figure 1. The hitrate reaches 30% even at very high traffic rates of 1K packet per second.

IV. CONCLUSION: BUG OR FEATURE?

We show that the attacks which were believed to apply only to connectionless communication, such as UDP or tunnelling,

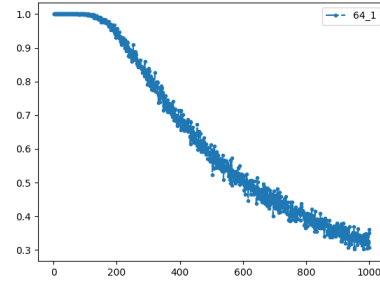


Fig. 1. Attack hit rate for different traffic volumes.

also apply to IP communication with TCP. Our work demonstrates that the recommendation to move to TCP to avoid the fragmentation attacks, does not solve the problem since TCP faces similar exploits as UDP. We recommend that the ICMP messages that trigger fragmentation should be filtered, e.g., in firewalls.

ACKNOWLEDGEMENTS

This work has been co-funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) SFB 1119.

REFERENCES

- [1] C. A. Kent and J. C. Mogul, "Fragmentation considered harmful," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 75–87, 1995.
- [2] Y. Gilad and A. Herzberg, "Fragmentation considered vulnerable: blindly intercepting and discarding fragments," in *Proceedings of the 5th USENIX conference on Offensive technologies*, 2011, pp. 2–2.
- [3] A. Herzberg and H. Shulman, "Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org," in *2013 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2013, pp. 224–232.
- [4] —, "Vulnerable delegation of dns resolution," in *European Symposium on Research in Computer Security*. Springer, 2013, pp. 219–236.
- [5] H. Shulman and M. Waidner, "Fragmentation considered leaking: port inference for dns poisoning," in *International Conference on Applied Cryptography and Network Security*. Springer, 2014, pp. 531–548.
- [6] M. Brandt, T. Dai, A. Klein, H. Shulman, and M. Waidner, "Domain validation++ for mitm-resilient pki," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2060–2076.
- [7] P. Jeitner, H. Shulman, and M. Waidner, "The impact of dns insecurity on time," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 266–277.
- [8] H. Shulman and M. Waidner, "Towards security of internet naming infrastructure," in *European Symposium on Research in Computer Security*. Springer, 2015, pp. 3–22.
- [9] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya, "Connection-oriented dns to improve privacy and security," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 171–186.
- [10] LetsEncrypt, "Mitigating DNS Fragmentation Attack," <https://community.letsencrypt.org/t/mitigating-dns-fragmentation-attack/74838>, 2018.
- [11] M. Göhring, H. Shulman, and M. Waidner, "Path mtu discovery considered harmful," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 866–874.

A.5. DNS-over-TCP Considered Vulnerable

[Dai21a]

Tianxiang Dai, Haya Shulman, and Michael Waidner. “DNS-over-TCP Considered Vulnerable”. In: *Proceedings of the Applied Networking Research Workshop. ANRW '21*. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 76–81. ISBN: 9781450386180. DOI: 10.1145/3472305.3472884. URL: <https://doi.org/10.1145/3472305.3472884>

DNS-over-TCP Considered Vulnerable

Tianxiang Dai
Fraunhofer SIT
Germany

Haya Shulman
Fraunhofer SIT
Germany

Michael Waidner
Fraunhofer SIT
TU Darmstadt
Germany

ABSTRACT

The research and operational communities believe that TCP provides protection against IP fragmentation attacks and recommend that servers avoid sending DNS responses over UDP but use TCP instead.

In this work we show that IP fragmentation attacks also apply to servers that communicate over TCP. Our measurements indicate that in the 100K-top Alexa domains there are 393 additional domains whose nameservers can be forced to (source) fragment IP packets that contain TCP segments. In contrast, responses from these domains cannot be forced to fragment when sent over UDP.

Our study not only shows that the recommendation to use TCP instead of UDP in order to avoid attacks that exploit fragmentation is risky, but it also unveils that the attack surface due to fragmentation is larger than was previously believed. We evaluate IP fragmentation-based DNS cache poisoning attacks against DNS responses over TCP.

CCS CONCEPTS

• Security and privacy → Network security.

KEYWORDS

DNS Cache Poisoning, IP Fragmentation, TCP

ACM Reference Format:

Tianxiang Dai, Haya Shulman, and Michael Waidner. 2021. DNS-over-TCP Considered Vulnerable. In *Applied Networking Research Workshop (ANRW '21)*, July 24–30, 2021, Virtual Event, USA. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3472305.3472884>

1 INTRODUCTION

IP fragmentation allows to adjust packets to the size supported by the networks which the packets traverse. Given

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ANRW '21, July 24–30, 2021, Virtual Event, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8618-0/21/07...\$15.00

<https://doi.org/10.1145/3472305.3472884>

a too large packet, the source or the routers fragment packets into smaller fragments. The receiver reassembles the fragments back into the original IP packets. To identify the fragments that belong to the same IP packet the receiver uses a 16 bit IP identifier (IP ID) in the IP header during reassembly. The operating system at the sender assigns an IP ID to every outbound IP packet. If a packet is fragmented by the source or by the intermediate devices, each IP fragment receives the same IP ID value as the original IP packet. The receiving host identifies fragments with the same IP ID and reassembles them into original IP packet.

Computing the IP ID value. Some operating systems use a globally incremental counter to generate IP ID values: after a packet is sent, the IP ID counter is incremented and the next packet receives the next value. For instance, Windows (e.g., Windows 8 and 10), Android 5.0.1 and FreeBSD use a global counter [20]. Other operating systems, such as new Linux versions, use unpredictable IP ID assignment. For instance, Linux and MacOS to use local counters and OpenBSD use pseudo-random IP ID assignment [21]. In this work we focus on servers with global incremental IP ID counters.

Fragmentation-based attacks. The idea underlying the attacks that exploit fragmentation is to interfere in the IP reassembly process with fragment mis-association by sending a spoofed fragment to the victim client, which when reassembled with the genuine fragment from the server, results either in an incorrect fragment which is discarded by the client (causing DoS attack against the target service, such as [16]) or results in a correct fragment which contains malicious payload that was carried in the spoofed fragment [10]. Such attacks are typically launched against services that run over UDP, such as DNS, where the majority of the requests/responses are exchanged over UDP. Since most UDP traffic is not fragmented, the attackers can trigger source fragmentation by sending a spoofed ICMP fragmentation needed error message (type: 3, code: 4) to the target server. The ICMP packet should contain the original IP header and first 8 bytes of the payload that triggered the error message as well as the MTU of the router that sent the ICMP error, [RFC792] [22].

TCP uses path MTU discovery. To avoid fragmentation TCP performs path MTU discovery (PMTUD) and accordingly adjusts the Maximum Segment Size (MSS). To discover the MTU the sender transmits IP packets with a do not

fragment (DF) bit set in the IP header. When a router with a smaller MTU receives such a packet it discards the packet and returns to the sender an ICMP fragmentation needed error message, illustrated in Figure 1. This causes the TCP at the sender to reduce the MSS to the value indicated in the MTU (minus 20 bytes) and resend the packet. The process iterates until the packet is received by the destination. As a result of PMTUD, services running over TCP are not subject to fragmentation and hence should not be vulnerable to fragmentation based attacks. Indeed, there are recommendations to use TCP when possible in order to avoid (DNS cache poisoning and DoS) fragmentation based attacks [2, 7, 18, 27]. Many DNS servers are configured to avoid fragmentation by lowering default EDNS buffer size and setting the TC bit on oversized DNS responses, signalling to the client that it should resend the DNS request over TCP.

Offsets	Octet	0				1				2				3																			
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	v4				IHL = 20				TOS				Total Length = 56																			
4	32					IPID				x DF MF				Frag Offset																			
8	64	TTL				Protocol = ICMPv4				IP Header Checksum																							
12	96	Source IP																															
16	128	Destination IP																															
20	160	Type = 3				Code = 4				ICMP Checksum				MTU = 68																			
24	192	Unused																															
28	224	v4				IHL = 20				TOS				Total Length = 100																			
32	256					IPID				x DF MF				Frag Offset																			
36	288	TTL				Protocol = ICMPv4				IP Header Checksum																							
40	320	Source IP = 2.2.2.2																															
44	352	Destination IP = 7.7.7.7																															
48	384	Type = 0				Code = 0				ICMP Checksum																							
52	416	Identifier								Sequence Number																							

Figure 1: ICMP Packet too big with EchoReply.

Fragmentation attacks against TCP. In this work we show that off-path attackers can force the servers to fragment communication over TCP. We perform a measurement study of the vulnerable servers. In contrast to common belief that TCP provides sufficient protection against off-path network adversaries we show that this is not so. We reveal that fragmentation based attacks against services over TCP allow to circumvent the entropy in TCP segments, including Sequence Number (SN) and source port. We explain the challenges that the network adversaries need to address to inject spoofed payload into TCP segments and demonstrate how forcing the servers to fragment traffic over TCP can be exploited for DoS and DNS cache poisoning attacks. In addition, fragmentation is also performed by the intermediate routers, when these have small next-hope MTU values. An analysis of CAIDA data-traces between 2008 and 2016 showed at least 1K intermediate routers in the Internet generate ICMP fragmentation needed packets with next MTU values even below 576 bytes, and fragment IP packets with TCP, despite path MTU discovery of the source, [9].

Organisation. We review related work in Section 2. In Section 3 we provide measurements of servers that can be

forced to fragment TCP traffic. In Section 4 we show measurements of IPID allocation in the Internet. In Section 5 we evaluate DNS cache poisoning attacks against DNS responses served over TCP. We conclude in Section 6.

2 RELATED WORK

IP fragmentation of UDP communication has a long history of attacks, including DoS (Denial of Service) attacks on IP defragmentation caches, [15, 16], and more recently IP fragments mis-association was exploited for injecting malicious content into NTP and DNS packets for shifting time and for launching DNS cache poisoning attacks respectively, [3, 10, 11, 13, 14, 19, 24, 25].

To reduce the threats introduced by fragmented IP packets that carry UDP datagrams best practices recommend to avoid fragmentation [1, 2, 5, 7, 18, 27]. Indeed, most servers were patched to avoid fragmentation, and use TCP for transmitting responses that are too large to fit in a single IP packet.

Is TCP really secure against IP fragmentation attacks? The possibility of fragmentation attacks against TCP was discussed in non academic community, starting with Zalewski who in 2004 in a bugtraq post¹ discussed a possibility for an attack that spoofs a non-first fragment of a TCP segment. There was however not a proof of concept of this attack and it was not validated in practice. In this work we demonstrate attacks that exploit IP fragmentation for injecting malicious records into DNS responses sent over TCP.

3 MEASURING TCP FRAGMENTATION

We perform a study of the Alexa Top-100K domains to infer the fraction of domains with nameservers that can be forced to fragment responses over TCP. For this, we first create a TCP connection to remote nameserver and send a DNS request over the connection. Once we receive the response, without acking, we send to the nameservers ICMP fragmentation needed packet, a.k.a. Packet Too Big (PTB, type: 3 and code: 4). Then we wait for retransmission and check if the response arrives fragmented, and record the minimum fragment size that we observe.

We evaluate reaction to three types of payloads embedded in ICMP PTB messages: TCP header, UDP header and ICMP echo reply. One example of ICMP PTB packet with ICMP echo reply as payload is showed in Figure 1. Considering that TCP is stateful, we also test two variations of TCP headers. One with wrong sequence number embedded in ICMP PTB (the TCP_Wrong row). The other with random port, thus for unexisting TCP connection (the TCP_New row). The results of our measurements are listed in Table 1.

¹<https://bugtraq.securityfocus.narkive.com/rm25I7e1/a-new-tcp-ip-blind-data-injection-technique>

Out of 100K domains, we collected nameservers in 97,493 domains (column ‘Total’ in Table 1) by querying for NS record. ‘N/A’ indicates that we received no response. The valid responses are marked as ‘Checked’. If we are able to cause nameservers in a domain to fragment responses, we mark the domain as ‘Fragmented’. Column ‘TC/NEW’ has different meanings for UDP and TCP rows. For the UDP row, it says 13.77% domains have nameservers responding with TC (Truncated) bit set. For the TCP row, it shows the number of new domains that were identified as ‘Fragmented’, which do not fragment DNS responses over UDP. Our results show that 9,751 domains are vulnerable only to fragmentation attacks against UDP and 393 domains are vulnerable only to fragmentation attacks against TCP. More importantly, among the servers that are vulnerable to our IP fragmentation attacks over TCP, we find servers which actively avoid UDP by responding with a TC bit set and requesting that the DNS resolver re-sends the DNS request over TCP. Indeed, there are 76 of the 393 new domains which respond with TC bit but are vulnerable to our attacks. Namely, these servers implement the recommendations to move to TCP in order to avoid fragmentation based attacks to which communication over UDP is known to be vulnerable!

	Fragmented	Checked	N/A	TC/NEW	Total
UDP	10.11%	87.62%	12.38%	13.77%	100%
PTB_UDP	9,854	85,428	12,065	13,429	97,493
TCP	0.46%	90.30%	9.70%	0.37%	100%
PTB_TCP_Right	445	88,037	9,456	356	97,493
TCP	0.43%	89.42%	10.58%	0.34%	100%
PTB_TCP_Wrong	420	87,174	10,319	334	97,493
TCP	0.13%	89.51%	10.49%	0.07%	100%
PTB_TCP_New	130	87,263	10,230	72	97,493
TCP	0.15%	90.29%	9.71%	0.12%	100%
PTB_UDP	149	88,031	9,462	121	97,493
TCP	0.10%	89.99%	10.01%	0.10%	100%
PTB_EchoReply	95	87,732	9,761	94	97,493
TCP	0.51%	91.43%	8.57%	0.40%	100%
ALL	496	89,135	8,358	393	97,493

Table 1: Fragmentation of UDP and TCP in 100K-top Alexa domains.

To infer the fragment sizes that the vulnerable nameservers are willing to reduce to, we evaluated ICMP error messages with varying MTU sizes. We plot the distribution of the minimum fragment sizes in Figure 2. We find that domains supporting fragmentation over TCP allow much smaller fragments than UDP. About 40% of the domains that allow TCP fragmentation, can be reduced to MTU of 292 bytes or smaller, while 90% of the domains vulnerable to UDP fragmentation limit the minimum fragment size to at least 548 bytes.

We also measure the ratio of vulnerable nameservers within a domain. The results are plotted in Figure 3. Among

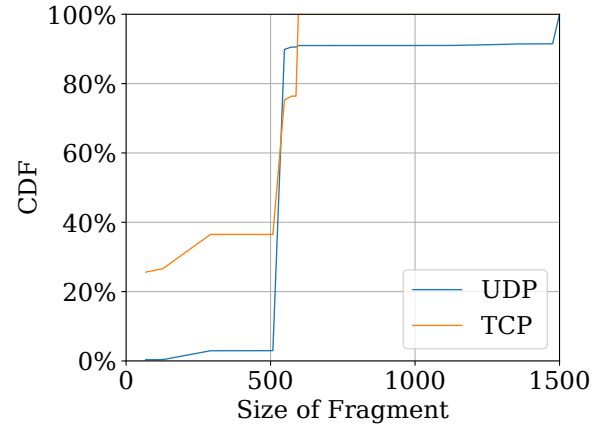


Figure 2: CDF of TCP and UDP fragment sizes.

the 496 domains with at least one nameserver vulnerable to fragmentation over TCP, 354 (71%) of them have more than half of their nameservers vulnerable. More specifically, all of the nameservers in 170 (34%) domains are vulnerable. Corresponding numbers for UDP are 9,065 (92%) and 7,706 (78%).

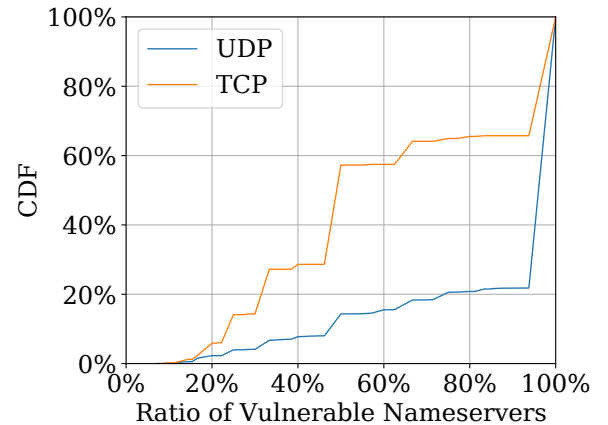


Figure 3: CDF of ratio of vulnerable nameservers for all vulnerable domains.

4 IP IDENTIFIER MEASUREMENTS

In this section we describe the IP ID allocation methods and report on the IP ID results we collected from the popular nameservers.

To identify the value of the IP ID we send packets from two hosts (with different IP addresses) to a nameserver.

IP Identifier. The 16 bit IP Identifier (IP ID) field in the IP header is used to identify fragments that belong to the same original IP packet [RFC791] [23]. The fragments are then reassembled by the recipient according to source and destination IP addresses, IP ID value and protocol field (e.g., TCP).

Global counter. Initially most operating systems used a globally incremental IP ID assignment which is easy to implement and has little requirement to keep state: just a single counter which is incremented with every packet that is sent. Global counters however were shown to be vulnerable to off-path attacks, [10]. A global counter is still popular in the Internet. Our study shows that 5.53% nameservers use global counter for UDP datagrams and 2.30% nameservers global counters for IP packets with TCP, see details in Table 2. To prevent the attacks some operating systems were patched to randomise their IP ID assignment.

In our work we focus on servers that implement globally incremental IP ID counters.

	Per-Host	Global	Zero	Random and other	N/A	Total
UDP	52.60%	5.53%	7.34%	33.40%	1.14%	100%
	51,281	5,388	7,152	32,560	1,112	97,493
TCP	14.43%	2.30%	75.92%	1.30%	6.04%	100%
	14,072	2,247	74,020	1,266	5,888	97,493

Table 2: IP ID allocation of in 100K-top Alexa.

5 OFF-PATH DNS POISONING OVER TCP

In this section we demonstrate cache poisoning attacks against DNS responses sent over TCP by injecting DNS records into fragmented IP packets. This shows that IP fragmentation attacks on TCP allow off-path attackers to bypass the randomisation with the sequence number and the source ports of TCP.

5.1 Attack Steps

In the first step of the attack, the adversary sends an ICMP echo reply message indicating a lower MTU. In the second step, the adversary finds out the IP ID value that will be assigned to the DNS response that the nameserver will send to the target victim. Next, the adversary crafts a spoofed second fragment with a TCP segment that contains a DNS record, and with the correct IP ID value, that matches the IP ID of the fragments sent by the nameserver. This spoofed fragment is stored in the IP defragmentation cache of the DNS resolver for 30 seconds. Note that the spoofed fragments are placed in advance. Thus there is no race condition. Subsequently, the adversary triggers a DNS request to the target domain. When the first genuine fragment of the DNS response arrives it is reassembled with the spoofed second fragment of the adversary and is moved on to TCP. If the checksum

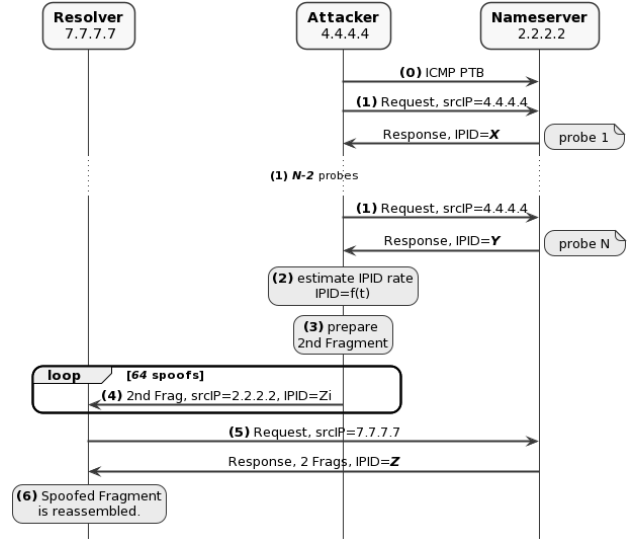


Figure 4: Traffic flow of the attack.

is correct, the ACK is in window, segment is accepted, and passed to DNS software. The ACK is in the window since it is the same sequence number (SN) value that was sent by the nameserver. Computing the TCP checksum is similar to computing the UDP checksum and is easy for an off-path attacker which knows the content of the original second fragment and knows which bytes it changed. Guessing the correct IP ID value depends on the rate at which the nameserver receives DNS requests. We explain next how to fix the TCP checksum and to extrapolate the IP ID value.

5.1.1 Fixing the checksum. Since the spoofed fragment modifies parts of the payload of the genuine IP packet sent by the nameserver it also changes the checksum. Hence the adversary needs to adjust the checksum to ensure the checksum matches the one in the original IP packet, this is simple and done similarly to UDP [10].

5.1.2 Predicting the IP ID value. We show our measurements of the IP ID algorithms in nameservers and then explain how we extrapolate the IP ID value in responses sent by busy nameservers.

IP ID measurements. Our measurements of the 100K-top Alexa domains show that 76,267 of the nameservers use predictable IP ID values in IP packets that contain TCP segments. In contrast, when communicating over UDP only 12,540 of the servers use predictable IP ID values in outbound packets.

Extrapolating IP ID. We first measure the rate at which the packets arrive at the nameserver. We do this by probing the nameserver’s IP ID value at stable intervals. Let this rate be M packets per second. We next do an extrapolation of the IP ID value assuming that the server receives M packets per

second. We use the following components: our prober and a nameserver that uses globally incremental IP ID. We use linear regression with Ordinary Least Square (OLS) method to estimate the relation between IP ID and timestamp t . Since IP ID is incremental, we assume:

$$IPID = a * t + b + \epsilon, \epsilon \sim N(0, \sigma^2)$$

We send N probes to 2.2.2.2 (in step (1)). With N probes, we can estimate a , b and σ using OLS method in step (2).

We implement this extrapolation algorithm for probing the IP ID values prior to beginning of the attack.

5.2 Attack Evaluation

We set up an Unbound DNS resolver, and run the attack evaluations against the nameservers with global IP ID counters in 100K-top Alexa domains. Recent as well as more older approaches on IP ID prediction in old operating systems [6, 8, 17] demonstrate that our attacks have a much wider scope. We next show how to construct a second IP fragment with a malicious DNS record inside it, so that it is reassembled with the first fragment sent by the nameserver and results in a valid TCP segment.

Figure 5 and Figure 6 are examples of two fragments of a DNS response over TCP. As we can see, all DNS related random challenges are in the first fragment, including TCP port and DNS Transaction ID (TXID). The only challenge the attacker needs to solve is to match the IPID. As explained in Section 5.1.2, it can be accomplished by probing and extrapolating. Afterwards, the attacker modifies the second fragment to include some malicious payload. To make the checksum pass, he can change few unused bytes in the packets, just as in [10]. Then he can spoof those second fragments to the resolver.

Offsets	Octet	0	1	2	3
Octet	Bit	0	1	2	3
0	0	v4	IHL = 20	TOS	Total Length
4	32	IPID		x DF MF	Frag Offset
8	64	TTL	Protocol=TCP	IP Header Checksum	
12	96	Source IP			
16	128	Destination IP			
20	160	Source Port		Destination Port	
24	192	Sequence Number			
28	224	Acknowledgement Number			
32	256	Data Offset	Flags	Window Size	
36	288	Checksum	Urgent Pointer		
40	320	DNS Length		TXID	
44	352	DNS Flags		Question Count	
48	384	Answer Count		Authority Count	
52	416	Additional Count		4	v
56	448	i	c	t	3
60	480	c	o	m	0
64	512	Type = A		Class = IN	

Figure 5: First fragment, assuming MTU = 68

We evaluate the DNS cache poisoning attack that were previously shown to apply to UDP communication [10], against the servers that we found vulnerable to fragmentation attacks on TCP. The idea of the attacks is to inject a spoofed

Offsets	Octet	0	1	2	3
Octet	Bit	0	1	2	3
0	0	v4	IHL = 20	TOS	Total Length
4	32	IPID		x DF MF	Frag Offset
8	64	TTL	Protocol=TCP	IP Header Checksum	
12	96	Source IP			
16	128	Destination IP			
20	160	Name Pointer		Type = A	
24	192	Class = IN		TTL	
28	224	TTL		Data Length = 4	
32	256	IPv4 Address = 4.4.4.4			
36	288				

Figure 6: Spoofed second fragment, with DNS payload modified

fragment (with a source IP address of the victim nameserver) that contains a malicious payload. The spoofed fragment of the attacker is reassembled with the genuine first fragment from the nameserver. The resulting DNS response contains malicious records injected from the second spoofed fragment.

We simulate the success rate of the attacks using our DNS resolver with an IP defragmentation cache size of 64 packets, against nameservers with different IP ID increment rates Figure 7. The hitrate reaches 30% even at very high traffic rates of 1K packet per second.

The amount of packets sent during the whole attack is low: N probes to estimate server’s IP ID rate and 64 spoofed second fragments, as showed in Fig. 4. Normally, fewer than 100 packets are required. Besides, the attacker does not need to repeat the probing phase for every attack. This makes the attack difficult to be detected.

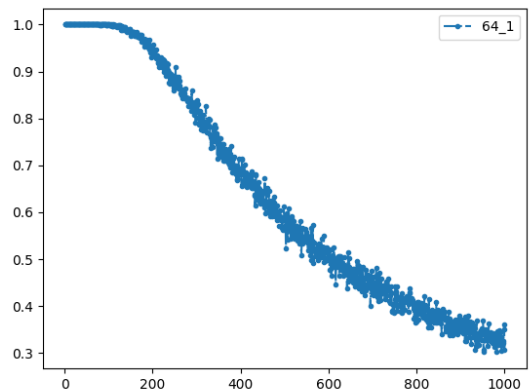


Figure 7: Attack hit rate for different traffic volumes. X-axis is the IP ID increment rate at nameserver (packet per second), Y-axis is success rate.

6 CONCLUSION: BUG OR FEATURE?

We show that the attacks which were believed to apply only to connectionless communication, such as UDP or tunnelling, also apply to IP communication with TCP. Our work shows that the recommendations to move to TCP to avoid the fragmentation attacks that apply to UDP, do not solve the problem.

The core issue is that the attacker can exploit ICMP messages with UDP datagrams as well as ICMP messages with echo reply packets to trigger fragmentation on TCP. Since these protocols allow fragmentation and do not trigger ICMP fragmentation needed error messages, the reaction of the operating systems to ICMP messages in such scenarios should be clarified in the standard. Our recommendation is that such ICMP messages are filtered in firewalls or at the servers.

To avoid DNS cache poisoning it is important to sign the zone files with DNSSEC [RFC4033-RFC4035]. Nevertheless, care should be taken as recent research demonstrated that some cache injections cannot be prevented even with DNSSEC, [12]. In addition, DNSSEC-signed domains can be vulnerable when the cryptographic material is weak or misconfigured, e.g., due to reuse of the shared modulus, [4, 26].

ACKNOWLEDGEMENTS

This work has been co-funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) SFB 1119.

REFERENCES

- [1] 2020. DNS Flag Day 2020. <https://dnsflagday.net/2020/>
- [2] Ron Bonica, Fred Baker, Geoff Huston, Bob Hinden, Ole Troan, and Fernando Gont. 2019. *IP fragmentation considered fragile*. Technical Report. IETF Internet-Draft (draft-ietf-intarea-frag-fragile), work in progress
- [3] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. 2018. Domain validation++ for mitm-resilient pki. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2060–2076.
- [4] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. A longitudinal, end-to-end view of the {DNSSEC} ecosystem. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 1307–1322.
- [5] J Dickinson, S Dickinson, R Bellis, A Mankin, and D Wessels. 2016. RFC7766: DNS transport over TCP-implementation requirements.
- [6] Xuewei Feng, Chuanpu Fu, Qi Li, Kun Sun, and Ke Xu. 2020. Off-Path TCP Exploits of the Mixed IPID Assignment. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1323–1335.
- [7] K Fujiwara and P Vixie. 2020. *Fragmentation Avoidance in DNS*. Technical Report. IETF Internet-Draft (draft-fujiwara-dnsop-avoid-fragmentation-03), work in progress
- [8] Yossi Gilad and Amir Herzberg. 2011. Fragmentation considered vulnerable: blindly intercepting and discarding fragments. In *Proceedings of the 5th USENIX conference on Offensive technologies*. 2–2.
- [9] Matthias Göhring, Haya Shulman, and Michael Waidner. 2018. Path MTU Discovery Considered Harmful. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 866–874.
- [10] Amir Herzberg and Haya Shulman. 2013. Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org. In *2013 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 224–232.
- [11] Amir Herzberg and Haya Shulman. 2013. Vulnerable delegation of DNS resolution. In *European Symposium on Research in Computer Security*. Springer, 219–236.
- [12] Philipp Jeitner and Haya Shulman. 2021. Injection Attacks Reloaded: Tunnelling Malicious Payloads over {DNS}. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.
- [13] Philipp Jeitner, Haya Shulman, and Michael Waidner. 2020. The Impact of DNS Insecurity on Time. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 266–277.
- [14] Philipp Jeitner, Haya Shulman, and Michael Waidner. 2020. Pitfalls of Provably Secure Systems in Internet The Case of Chronos-NTP. In *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 49–50.
- [15] Charlie Kaufman, Radia Perlman, and Bill Sommerfeld. 2003. DoS protection for UDP-based protocols. In *Proceedings of the 10th ACM conference on Computer and communications security*. 2–7.
- [16] Christopher A Kent and Jeffrey C Mogul. 1987. *Fragmentation considered harmful*. Vol. 17.
- [17] Amit Klein and Benny Pinkas. 2019. From IP ID to Device ID and KASLR Bypass. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1063–1080.
- [18] LetsEncrypt. 2018. Mitigating DNS Fragmentation Attack. <https://community.letsencrypt.org/t/mitigating-dns-fragmentation-attack/74838>.
- [19] Aanchal Malhotra, Isaac E Cohen, Erik Brakke, and Sharon Goldberg. 2016. Attacking the Network Time Protocol.. In *NDSS*.
- [20] Sophon Mongkolluksamee, Kensuke Fukuda, and Panita Pongpaibool. 2012. Counting NATted hosts by observing TCP/IP field behaviors. In *2012 IEEE International Conference on Communications (ICC)*. IEEE, 1265–1270.
- [21] Liran Orevi, Amir Herzberg, and Haim Zlatokrivlov. 2018. DNS-DNS: DNS-based de-nat scheme. In *International Conference on Cryptology and Network Security*. Springer, 69–88.
- [22] Jon Postel. 1981. Internet Control Message Protocol darpa internet program protocol specification. *RFC 792* (1981).
- [23] Jon Postel. 1981. Internet protocol—DARPA internet program protocol specification, rfc 791. (1981).
- [24] Haya Shulman and Michael Waidner. 2014. Fragmentation considered leaking: port inference for dns poisoning. In *International Conference on Applied Cryptography and Network Security*. Springer, 531–548.
- [25] Haya Shulman and Michael Waidner. 2015. Towards security of internet naming infrastructure. In *European Symposium on Research in Computer Security*. Springer, 3–22.
- [26] Haya Shulman and Michael Waidner. 2017. One key to sign them all considered vulnerable: Evaluation of {DNSSEC} in the internet. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 131–144.
- [27] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. 2015. Connection-oriented DNS to improve privacy and security. In *2015 IEEE symposium on security and privacy*. IEEE, 171–186.

A.6. From IP to Transport and Beyond: Cross-Layer Attacks Against Applications

[Dai21s]

Tianxiang Dai et al. “From IP to Transport and beyond: Cross-Layer Attacks against Applications”. In: *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. SIGCOMM '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 836–849. ISBN: 9781450383837. DOI: 10.1145/3452296.3472933. URL: <https://doi.org/10.1145/3452296.3472933>

Declaration of Contributions

The paper "From IP to Transport and beyond: Cross-Layer Attacks against Applications" was published as a full research paper at the "ACM SIGCOMM 2021 Conference". It constitutes a joint work of Tianxiang Dai, Philipp Jeitner, Haya Shulman and Michael Waidner.

Haya Shulman proposed the initial concept and wrote Introduction, Background and Conclusions, while Philipp Jeitner wrote the rest. After the initial reviews, Haya Shulman and Philipp Jeitner decided on what to change about the paper and to add new attack methodologies. Tianxiang Dai and Philipp Jeitner performed the analysis of applications, more specifically Tianxiang Dai did this for XMPP, Email and PKI and Philipp Jeitner for RADIUS, NTP, Bitcoin, Tunneling and Intermediate Devices. Tianxiang Dai and Philipp Jeitner designed, implemented and analysed the measurements together. More specifically, on the resolver side, Tianxiang Dai developed the testing infrastructure and performed the SADDNS measurement for all. Tianxiang Dai did the resolver tests for CAs, VoIP, Email, and Open Resolvers, while Philipp Jeitner did the resolver tests on Eduroam, CDNs, AdNet and NTP. On the nameserver side, Tianxiang Dai performed all the measurements. The data analysis was done by Tianxiang Dai and Philipp Jeitner together, where Philipp Jeitner did most of the analysis. Tianxiang Dai also built the online tool which allows everyone in the Internet to test their own resolvers. Michael Waidner was a general advisor of this work and contributed with continuous feedback during all phases of the paper writing process. The paper was presented at the conference by Philipp Jeitner.

All authors agree with the use of their joint paper as part of Philipp Jeitner’s and Tianxiang Dai’s cumulative dissertation, considering a contribution of 70% from Philipp Jeitner and 30% from Tianxiang Dai.

From IP to Transport and Beyond: Cross-Layer Attacks Against Applications

Tianxiang Dai
Fraunhofer SIT
Germany

Philipp Jeitner
Fraunhofer SIT
TU Darmstadt
Germany

Haya Shulman
Fraunhofer SIT
Germany

Michael Waidner
Fraunhofer SIT
TU Darmstadt
Germany

ABSTRACT

We perform the first analysis of methodologies for launching DNS cache poisoning: manipulation at the IP layer, hijack of the inter-domain routing and probing open ports via side channels. We evaluate these methodologies against DNS resolvers in the Internet and compare them with respect to effectiveness, applicability and stealth. Our study shows that DNS cache poisoning is a practical and pervasive threat.

We then demonstrate cross-layer attacks that leverage DNS cache poisoning for attacking popular systems, ranging from security mechanisms, such as RPKI, to applications, such as VoIP. In addition to more traditional adversarial goals, most notably impersonation and Denial of Service, we show for the first time that DNS cache poisoning can even enable adversaries to bypass cryptographic defences: we demonstrate how DNS cache poisoning can facilitate BGP prefix hijacking of networks protected with RPKI even when all the other networks apply route origin validation to filter invalid BGP announcements. Our study shows that DNS plays a much more central role in the Internet security than previously assumed.

We recommend mitigations for securing the applications and for preventing cache poisoning.

CCS CONCEPTS

• Security and privacy → Network security.

KEYWORDS

DNS Cache Poisoning, Fragmentation, BGP hijacking, Side Channels

ACM Reference Format:

Tianxiang Dai, Philipp Jeitner, Haya Shulman, and Michael Waidner. 2021. From IP to Transport and Beyond: Cross-Layer Attacks Against Applications. In *ACM SIGCOMM 2021 Conference (SIGCOMM '21), August 23–28, 2021, Virtual Event, USA*. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3452296.3472933>

1 INTRODUCTION

Domain Name System (DNS), [RFC1034, RFC1035] [59, 60], plays a central role in the Internet. Designed and standardised in the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGCOMM '21, August 23–28, 2021, Virtual Event, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8383-7/21/08...\$15.00

<https://doi.org/10.1145/3452296.3472933>

80s to provide lookup services DNS has evolved into a complex infrastructure and is being increasingly used to support a wide variety of existing and future applications and security mechanisms. Given the large dependency of the Internet on DNS it also became a lucrative target for attacks.

DNS cache poisoning. In a cache poisoning attack an adversary injects malicious DNS records into the cache of a victim DNS resolver. Poisoning the cache enables the adversary to redirect the victims using that DNS resolver to malicious hosts instead of the genuine servers of the target domain. As a result, the adversary intercepts all the services in the target domain.

In this work we explore how practical off-path DNS cache poisoning attacks are and how such attacks can be exploited to launch cross-layer attacks against applications.

Taxonomy of cache poisoning methodologies. As we explain in Section 2, off-path DNS cache poisoning is challenging to launch in practice. Nevertheless, there are methodologies that, depending on different conditions, can result in practical attacks. In this work we evaluate such methodologies for launching cache poisoning attacks: (1) BGP prefix hijacking, (2) transport layer side channels and (3) injections into IP defragmentation cache. These methodologies were previously used for issuing fraudulent certificates, [22] or for hijacking bitcoins [17]. Attacks for issuing fraudulent certificates were also carried out by [24] using IP fragmentation; a method initially proposed in [38]. [57] combined ICMP error messages and rate limiting of nameservers to create a side channel for guessing the source port in DNS requests, but have not evaluated this attack against real Internet systems.

Which of the methods is more effective? Which has higher applicability? Which is stealthier and does not trigger alerts?

To answer these questions we perform the first comparative analysis of the methodologies for cache poisoning attacks. In addition, in order to gain a deeper understanding of the methodologies and their impact on the Internet applications, we also extend the evaluations in the previous work [22, 24, 57] for Internet scale measurements of applicability and effectiveness of these methodologies against multiple Internet networks.

Taxonomy of vulnerable applications. The implications of cache poisoning for the other Internet services and applications has not been explored. There is evidence of cache poisoning in the wild, mostly for redirecting victims to impersonating websites [64]. Cache poisoning was also demonstrated in research against the certificate authorities [22, 24]. But there is no comprehensive study of exploits of cache poisoning against Internet clients and services.

What applications are at risk due to cache poisoning? How can an attacker exploit cache poisoning to attack applications? What is

the fraction of vulnerable applications in the Internet? What are the challenges and what cache poisoning methodologies are more suitable?

We answer these questions by evaluating the cache poisoning methodologies against a range of popular applications. We defined nine categories of applications, ranging from security mechanisms, to VoIP, email and intermediate devices; see Table 1. We provide the first systematic study of cache poisoning against a collection of popular applications and security mechanisms.

Poisoning is a threat to applications. Our results demonstrate that, *although challenging to launch, off-path DNS cache poisoning poses a realistic threat for many Internet applications.* Surprisingly, we show that DNS cache poisoning can be applied for downgrade attacks against security mechanisms causing the victims not to perform validation, e.g., RPKI or domain-based anti-spam validation. Taking RPKI as an example, we developed an attack that by redirecting the RPKI cache [RFC6810] [25] to a wrong repository via DNS cache poisoning, the attacker can cause the RPKI validation to result in status unknown (instead of invalid). As a result the RPKI cache will not validate correctness of the BGP announcements that it receives. Suppressing RPKI validation allows the adversary to perform BGP prefix hijacks even of ASes which have the corresponding RPKI material (Route Origin Authorization and resource certificates [54]) in the public repositories and hijack even the senders which enforce route origin validation [61].

Another example is malware distribution by causing the anti-spam validation to fail via cache poisoning.

This is the first demonstration of the devastating power of DNS cache poisoning, which shows that in addition to traditional threats, such as impersonation, DNS cache poisoning can facilitate much stronger attacks which were otherwise not possible. We also show that DNS cache poisoning can be used to inflict Denial of Service (DoS) on applications and their clients.

In our experimental evaluation against the applications we exploit DNS cache poisoning to subvert correctness and security of basic Internet functions, enabling the attackers to take over IP addresses, to hijack telephony, to de-synchronise local time, and even prevent victims from connecting to the correct VPN tunnel.

Off-path attacks. Our study is performed with off-path attackers. This is the weakest attacker model in the Internet, it can merely send packets from spoofed IP addresses, which is a realistic assumption since around 30% of the Internet networks do not enforce egress filtering [19–21, 55, 56, 58]. Essentially any adversary in the Internet has off-path capabilities and can select networks which allow it to send packets with spoofed source IP addresses. Stronger attackers, most notably the on-path Man-in-the-Middle (MitM), can do more devastating attacks. Nevertheless, MitM attackers are more rare and even such attackers have limitations: the strong government sponsored attackers can be on-path only to some of the Internet victims depending on the paths that they control but even they do not control all of the networks. Therefore, it is critical to understand the threat that an off-path attacker poses to applications.

Disclosure and ethics. Our attacks were tested against remote networks reliably, yet were ethically compliant. We measured and evaluated vulnerabilities in the DNS caches of the subjects of our study and measured which services use the caches but did not hijack their traffic nor Internet resources and neither did we place incorrect DNS records for Internet domains that are not under

our control in the caches of our test subjects. Specifically, to avoid harming Internet customers and domains, we set up a victim AS and victim domains as well as adversarial AS and adversarial hosts on that AS, which were used by us for carrying out the attacks against the victims. Our measurement study for evaluating the vulnerabilities was performed using our victim domains, which ensured that the targets of our study would not use the spoofed records for any “real” purpose.

We believe that in addition to disclosing the vulnerabilities to the affected entities it is critical to raise awareness to the extent and the scope of the vulnerabilities.

Contributions. We present the first comprehensive study of the attack surface that off-path DNS cache poisoning introduces on the Internet ecosystem.

- We implement three methodologies for launching off-path DNS cache poisoning attacks: (1) BGP prefix hijacking, (2) side-channels and (3) fragmentation. We perform the first Internet-scale evaluation of these methodologies against DNS resolvers and compare them for applicability, stealthiness and success of cache poisoning.
- We apply these methodologies to launch *cross-layer attacks* against widely used applications and services (see taxonomy in Table 1). Our study shows that cache poisoning can be used to bypass security mechanisms, to cause DoS attacks, or for impersonation attacks.
- We provide recommendations for countermeasures for DNS caches against cache poisoning attacks and for applications against cross-layer attacks even when using poisoned caches.

Organisation. We review DNS cache poisoning and related work in Section 2. In Section 3 we present the DNS cache poisoning methodologies that we use throughout our work. In Section 4 we demonstrate cross-layer attacks against applications using DNS cache poisoning. We provide results of our measurements in Section 5 and recommend mitigations in Section 6. We conclude this work in Section 7.

2 DNS CACHE POISONING OVERVIEW

Domain Name System (DNS) [60] cache poisoning allows an attacker to redirect victims to attacker controlled hosts. Typically the attackers targets recursive DNS resolvers whose caches serve multiple clients. A single injection of a malicious DNS record propagates to all the hosts that use that resolver. The attacker can then intercept the traffic between the services (such as web, email, FTP) in the victim domain and the hosts that use the poisoned cache. DNS resolvers use defences to make launching successful cache poisoning attacks difficult.

2.1 Defences Against Poisoning

The DNS resolvers are required to randomise certain fields in DNS requests sent to the nameservers, [RFC5452] [43]. These include a random 16 bit UDP source port and the 16 bit DNS transaction identifier (TXID); additional defences include nameserver randomisation [43] and 0x20 encoding [29]. The nameservers copy these fields from the DNS request to the DNS response. DNS resolvers accept the first DNS response with the correctly echoed challenge values and ignore any responses with incorrect values.

To launch a successful cache poisoning attack, the attacker needs to guess the correct challenge values and make sure that his spoofed response arrives before the genuine response from the real nameserver. This is easy for an on-path (man-in-the-middle) attacker, which can simply copy the values from the request to the response. Cryptographic signatures with DNSSEC [RFC6840] [73] could prevent on-path attacks, however, DNSSEC is not widely deployed. Less than 1% of the second level domains (e.g., 1M-top Alexa) are signed, and most resolvers do not validate DNSSEC signatures, e.g., [26] found only 12% in 2017. Our measurements indicate that less than 5% of the domains we studied are signed. There is however an increase in the resolvers validating DNSSEC: we found 28.6% validating resolvers via our ad-network study. Deploying DNSSEC was shown to be cumbersome and error-prone [27]. Even when widely deployed DNSSEC may not always provide security: a few research projects identified vulnerabilities and misconfigurations in DNSSEC deployments in popular registrars [44, 67].

Recent proposals for encryption of DNS traffic, such as DNS over HTTPS [41] and DNS over TLS [42], although vulnerable to traffic analysis [65, 68], may also enhance resilience to cache poisoning. These mechanisms are not yet in use by the nameservers in the domains that we tested. Nevertheless, even if they become adopted, they were not designed to protect the entire resolution path, but only the link between the client and the recursive resolver, and hence will not prevent DNS cache poisoning attacks.

2.2 History of DNS Cache Poisoning

In 2007 Klein identified vulnerability in Bind9 DNS resolvers [50] and in Windows DNS resolvers [51] allowing off-path attackers to reduce the entropy introduced by the TXID randomisation. In 2008 Kaminsky [47] presented a practical cache poisoning attack even against truly randomised TXID. Vixie suggested to randomise the UDP source ports already in 1995 [72], subsequently in 2002 Bernstein warned that relying on randomising TXID alone is vulnerable [18]. Following Kaminsky attack DNS resolvers were patched against cache poisoning [43], and most randomised the UDP source ports in queries.

Nevertheless, shortly after new approaches were developed allowing cache poisoning attacks. In 2012 [37] showed that off-path attackers can use side-channels to infer the source ports in DNS requests. In 2015 [66] showed how to attack resolvers behind upstream forwarders. This work was subsequently extended by [74] with poisoning the forwarding devices. A followup work demonstrated such cache poisoning attacks also against stub resolvers [16]. [57] showed how to use ICMP errors to infer the UDP source ports selected by DNS resolvers. Recently [52] showed how to use side channels to predict the ports due to vulnerable PRNG in Linux kernel. In 2013 [38] provided the first feasibility result for launching cache poisoning by exploiting IPv4 fragmentation.

For the evaluations in this work we selected three generic cache poisoning methodologies developed in [22, 38, 57], which are not specific to implementation or setup and do not result due to bugs in randomness generation, such as [52]. We perform Internet-wide measurements of these methodologies testing experimentally DNS cache poisoning against DNS resolution platforms. We then exploit

these poisoned caches to attack applications that use the poisoned records we injected.

2.3 DNS Cache Poisoning in the Wild

There is numerous evidence of DNS cache poisoning attempts in the wild, [7–13, 28, 64, 69], which were predominantly launched via short-lived BGP (Border Gateway Protocol) prefix hijacks or by compromising a registrar or a nameserver of the domain.

We consider only attacks done by network attackers by manipulating the protocols remotely but without compromising services or networks. Hence compromises of registrars or servers is not in our scope and in the review of works we focus only on BGP prefix hijacks, side channels and fragmentation attacks.

In 2017 [17] simulated the effects of BGP prefix hijacks on bitcoin without experimentally evaluating it in the wild. In 2018, [22] experimentally evaluated the impact of BGP prefix hijacks on domain validation and [24] evaluated the impact of DNS cache poisoning on domain validation. In 2020 a recent research project [70] evaluated BGP prefix hijacks for cross-layer attacks on Tor (the onion routing) [31] users, domain validation and bitcoin [34].

Except for fragmentation based DNS cache poisoning against domain validation [24] there were no studies of cache poisoning using different methodologies and their evaluation against applications. In this work we perform the first comprehensive study of DNS cache poisoning against different applications, and using different methodologies.

3 TAXONOMY OF POISONING METHODS

In our evaluations in subsequent sections we use three methodologies for poisoning DNS caches, which were shown to be practical in previous research: (1) intercepting DNS requests with BGP prefix hijacking [70], (2) guessing challenge values in DNS requests via side-channel [57] or (3) injecting content into IP defragmentation cache [38]. In this section we describe these attack methodologies, their unique properties and explain what attacker capabilities they assume. We compare effectiveness and stealthiness of each of these methods for carrying out cache poisoning attacks.

Setup. To test our attacks experimentally in the Internet we setup a victim AS and associate a /22 prefix with our AS. We register victim domains and setup nameservers and a DNS resolver.

3.1 Intercepting DNS with BGP Hijacking

A malicious Autonomous System (AS) can exploit vulnerabilities in BGP to hijack packets of some victim AS. A route hijack happens when an attacker announces an incorrect prefix belonging to a different AS. The attacker hijacks the prefix or a sub-prefix which has the IP address of a DNS nameserver or a resolver. If the hijack succeeds, the ASes that accepted the hijack will send all their traffic destined to the victim prefix instead to the attacker. The goal of the attacker is to intercept a single DNS packet, either a query sent by the resolver or a corresponding response of the nameserver. For simplicity in this discussion we focus on sub-prefix hijacks and assume that the attacker attempts to hijack the DNS query; see [22] for a taxonomy of BGP prefix hijack attacks. The attacker intercepts the DNS query and crafts a spoofed DNS response with malicious records and the correct challenge values, and sends it

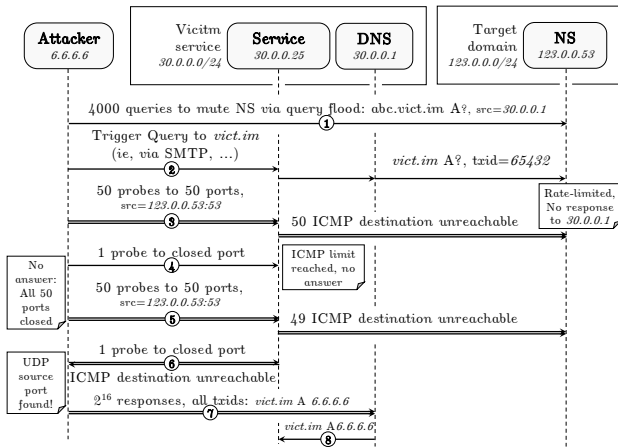


Figure 1: DNS Poisoning with side-channel.

to the victim DNS resolver. Additionally, to avoid detection due to blackholing, the attacker should relay all the traffic to the legitimate destination, except for the DNS query which it intercepted (to avoid race condition with the response from the genuine nameserver). We call this DNS cache poisoning attack method HijackDNS.

3.2 Guessing Challenges with Side-channel

The SadDNS off-path attack [57] uses an ICMP side channel to guess the UDP source port selected by the victim resolver in its query to the target nameserver. This is done via a side-channel present in most modern operating systems which allows the attacker to test if a given UDP port open or not. The operating systems have a constant, global limit of how many ICMP port unreachable messages they will return when packets are received at closed UDP ports (50 in the case of linux). The attacker splits the range of ports to sets of N ports and for every set performs the following: the attacker sends 50 probes with a spoofed source IP address of the nameserver to a range of UDP ports at the resolver. If the probes arrived at closed ports only, the returned ICMP error messages reach the global limit, and further messages will not be issued. The attacker sends a single probe from the IP address of the attacker to a known-closed port. If all of the previously probed 50 ports were closed the attacker will not receive an ICMP message in response to his own message. However, if one of the 50 probed ports was open, the limit was not reached, the attacker will receive an ICMP port unreachable message. The attacker repeats this process until a set containing an open port is found. Once a set with an open port is found, the attacker applies divide and conquer with the technique above dividing the ports until a single open port is isolated. This reduces the entropy of the challenge-response parameters unknown to the attacker from 32 bit (DNS TXID + UDP port number) to 16 bit.

Once the open port is identified the attacker sends multiple spoofed DNS responses from a spoofed IP address (of the nameserver) to that open UDP port of the resolver, for each possible TXID value, total of 2^{16} spoofed responses; e.g., [37, 45, 57]. A packet with the correct TXID is accepted by the DNS resolver. The attack is illustrated in Figure 1. The attack applies to only about 18% of the

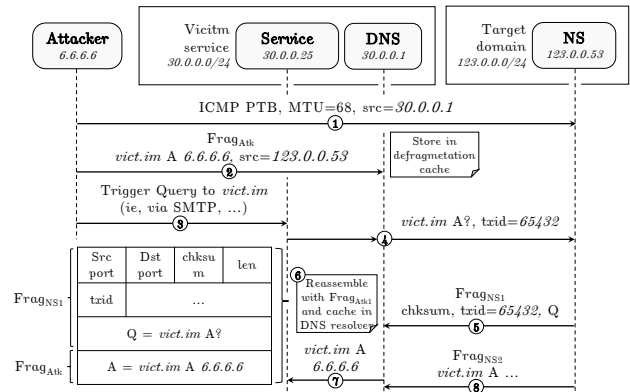


Figure 2: Fragmentation-based DNS poisoning.

domains with nameservers that use rate-limiting. The rate limiting allows the adversary to delay the response from the genuine nameserver and hence to win the race against it. Additionally, the attack applies only against resolvers with a global (un-patched) ICMP rate limit.

3.3 Injecting Records via IP Fragmentation

In this section we describe an attack which exploits IP fragmentation to inject spoofed fragments into the IP defragmentation cache on the victim system. The spoofed fragments contain malicious content, which when reassembled with the genuine fragments, manipulate the payload of the original IP packet without having to guess the values in the challenge-response parameters, [38].

We assume that the response from the nameserver is fragmented and arrives in at least two fragments. The fragment sent by the attacker is reassembled with the first fragment sent by the nameserver. The attacker replaces the second fragment of the nameserver with its malicious fragment, which overwrites part of the payload of the genuine DNS response from the nameserver, with malicious values. Since the challenge-response values (port, TXID) are in the first fragment, they remain unchanged. The illustration of the attack is in Figure 2.

To cause the nameserver to fragment a DNS response the attacker sends to the nameserver a *ICMP Destination Unreachable Fragmentation Needed* error message (type 3, code 4) with a DF bit set, signalling to the nameserver that the Maximum Transmission Unit (MTU) to the destination is smaller than the packet's length. The nameserver reduces the size of the packet accordingly by fragmenting the IP packet to smaller fragments.

4 EXPLOITING DNS POISONING FOR CROSS-LAYER ATTACKS

In this section we demonstrate how DNS cache poisoning can be used to launch cross-layer attacks against popular applications. In Section 4.1 we explain our methodology for selecting the applications. We list the categories according to which we selected the applications in Table 1. Our analysis of the applications is performed according to the key properties related to cache poisoning: (1) control over the query, (2) which records can be injected, (3)

Category	Protocol	Use Case	query name known	query trigger method	Record Type	DNS used for loc. fed. auth.	Methodologies Hijack SadDNS Frag	Cache Poisoning impact
Authentication	Radius	Peer discovery	target ✓ ¹	direct	NAPTR, SRV, A	✓ ✓	✓ ✓ ✓	DoS: no network access
Online Chat	XMPP	Chat+VoIP	target ✓ ¹	bounce	A, SRV	✓ ✓	✓ ✓ ✓	Hijack: eavesdropping
Email	SMTP SPF,DMARC DKIM	Mail	target ✓ ¹	direct/bounce	A, MX	✓ ✓	✓ ✓ ✓	Hijack: eavesdropping
		Anti-Spam	target ✓ ¹	authentication	TXT	✓	✓ ✓ ✓	Downgrade: spoofing
		Integrity Checking	target ✓ ¹	direct/bounce	TXT	✓	✓ ✓ ✓	Downgrade: spoofing
Web	HTTP SMTP	Web sites	target ✓ ¹	direct	A	✓	✓ ✓ ✓	Hijack: eavesdropping
		Password recovery	target ✓ ¹	direct	A, MX, TXT	✓	✓ ✓ ✓	Hijack: account hijack
Sync	NTP	Time synchronisation	known ✓	connection DoS	A	✓	✓ X ✓ ²	Hijack: change time
Crypto-currency	Bitcoin	Peer discovery	known ✓	waiting	A	✓	✓ X X	Hijack: fake blockchain
Tunnelling	OpenVPN	VPN	config X	connection DoS	A	✓	✓ ✓ ² ✓ ²	DoS: no VPN access
	IKE	VPN	config X	connection DoS	A	✓	✓ ✓ ² ✓ ²	DoS: no VPN access
	IKE	Opportunistic Enc.	target ✓ ¹	bounce	IPSECKEY	✓ ✓	✓ ✓ ² ✓ ²	Hijack: eavesdropping
PKI	DV	Domain Validation	target ✓ ¹	authentication	A, MX, TXT	✓ ✓	✓ X X	Hijack: fraud. certificate
	OCSP	Revocation checking	target ✓ ¹	direct	A	✓	✓ ✓ ✓	Downgrade: no check
	RPKI	Repository sync.	known ✓	waiting	A	✓	✓ X X	Downgrade: no ROV
Intermediate devices	-	Firewall filters	config X	waiting	A	✓	✓ ✓ ² ✓ ²	Downgrade: no filters
	HTTP/...	Loadbalancers	config X	on-demand	A	✓	✓ ✓ ² ✓ ²	Hijack: eavesdropping
	HTTP	CDN's	config X	on-demand	A	✓	✓ X ✓ ²	Hijack: eavesdropping
	DNS	ANAME/ALIAS[33]	config X	on-demand	A	✓	✓ ✓ ² ✓ ²	Hijack: eavesdropping
	HTTP/Socks	Proxies	target ✓ ¹	direct	A	✓	✓ ✓ ² ✓ ²	Hijack: eavesdropping

¹: Depends on the attack scenario. ²: Requires a third-party application to trigger queries.

Table 1: Evaluation of attacks against popular systems leveraging a poisoned DNS cache.

how the application uses the injected records, and (4) the outcome of the attack.

4.1 Methodology for Selecting Applications

We select the applications according to the following considerations: application category, usage of DNS by the application and the impact of DNS cache poisoning on the application.

4.1.1 Category. We categorise the applications to groups, covering most of the popular applications and security mechanisms in the Internet (left most column in Table 1). Within each category we selected a few representative protocols and systems for that category, see column ‘Protocol’ in Table 1.

4.1.2 Usage of DNS. One of the considerations for selecting the applications is how the application uses DNS: how the queries are sent by the application to the DNS resolver and how the results from the lookups are processed. The column ‘Use-Case’ in Table 1 describes the usage scenarios of the DNS by the application. We defined the following types:

Location (loc): DNS is used to locate a direct communication partner, typically in form of a hostname-to-ip (A, AAAA) mapping.

Federation (fed): DNS is used to locate a user’s home-server based on the domain part of a user address of the form user@domain.

Authorisation (auth): DNS is used to authorise a certain action or host in the name of the domain’s owner.

4.1.3 Queries. Applications differ in flexibility in allowing external entities to trigger queries. Our selection of applications aims to cover the variety of options for triggering queries. To initiate the attack, our adversary needs to cause the victim resolver to issue the target query or to predict when the query will have been issued.

Some applications enable the attacker to send arbitrary queries, e.g., in systems which use DNS for peer discovery in federated systems like Radius, XMPP and SMTP. This is because in these systems, the queried domains are part of the user’s ID. This user

ID can be controlled by the attacker to trigger a query to a domain of its choice. The same applies to all (sub-)systems used as part of web browsing, like HTTP, DANE and OCSP, since the attacker can establish direct connection from the victim client to arbitrary web servers which will trigger a DNS lookup that way. Setting the domain name is not always possible, e.g., in NTP the query is selected by the resolver based on the hostname that it receives from the local NTP server.

We evaluated popular appliances and systems for their query triggering behaviour. We list some selected systems in Table 2. As can be seen, some allow external adversaries to trigger queries (indicated with “on-demand” in column Trigger query). Other devices use timers for issuing queries. Hence the adversaries can often predict when the query is issued.

4.1.4 Impact of poisoning on applications. We select applications to demonstrate the impact that cache poisoning on applications can create: DoS (Denial of Service), downgrade of security or interception attacks.

4.2 Methodology for Attacking Applications

We developed cross-layer attacks that leverage DNS cache poisoning to attack applications listed in Table 1. The steps underlying all our cross-layer attacks against applications are:

(1) Use the application to send to the victim DNS resolver a query. In addition to the traditional ways of triggering queries, such as with a script or Email, we also developed new ways to trigger queries which were not known prior to our work. Some of these techniques are specific to appliances and platforms, see Table 2, while others are application-independent methodologies for triggering queries. We explain our methodologies for triggering queries in Section 4.3.

(2) Inject malicious records to poison the cache of the victim DNS resolver. We use the methodologies in Section 3 for injecting malicious records into the cache of the victim DNS resolver. In

Type	Provider	Trigger query	Caching time	Websites in Alexa 100K
Firewall	pfSense	timer	500s	-
	Sophos UTM	timer	240s	-
Load balancer	Kemp Technologies	timer	1h	-
	F5 Networks	timer	1h	-
CDN	Stackpath	on-demand	TTL	79
	Fastly	timer	TTL	1,143
	AWS	on-demand	TTL	11,057
	Cloudflare	on-demand	TTL	17,393
Managed DNS (ALLIAS)	DNSimple	on-demand	TTL	248
	DNS Made Easy	timer	~35min	1,192
	Oracle Cloud	on-demand	TTL	1,382
	Cloudflare	on-demand	TTL	20,027

Table 2: Query triggering behaviour at middleboxes. Last column shows the number of websites in 100K-top Alexa which use that provider.

Table 1 we summarise the applicability of the cache poisoning methodologies for cross-layer attacks against each application, and explain this in Section 4.4.

(3) Exploit the poisoned records to cause a victim application to divert from the expected behaviour. The outcomes of our cross-layer attacks against applications range from downgrading security to denial of service attacks and to more traditional impersonation attacks, explained in Section 4.5.

4.3 Methodologies for Triggering Queries

4.3.1 Common ways for triggering queries. The most challenging aspect of cross layer attacks that use DNS cache poisoning is the ability to trigger or predict DNS requests. Typically an external adversary does not have access to internal services, such as the DNS resolver, and hence should not be able to cause the DNS resolver to issue arbitrary DNS requests. Adversaries can trigger queries via bounce. For instance, by sending an Email to a non existing recipient in the target domain the adversary will cause the Email server to return an error message with Delivery Status Notification. To send the error the Email server requires the IP address and hostname of the MX server in the domain that sent the Email message which triggered the error. This causes queries to the domain specified by the attacker.

The adversary can also set up a web server and lure clients to access it, this is a direct query triggering. The clients download the web objects from the adversary’s domain, and send DNS requests to the DNS resolvers on their networks. When resolvers receive DNS requests from servers or clients on their networks they initiate DNS resolution. However, these approaches are limited. For instance, only about 18% of the Email servers trigger DNS requests when receiving Emails, [53]. The limitation with web clients is that the adversary must wait until the target client visits the malicious web page. Furthermore, web clients cannot be used to poison resolvers that are used only by servers, such as Email or NTP. In this section we develop new approaches for triggering queries.

4.3.2 Cross-applications DNS caches. The adversary may be able to use one application to trigger queries to inject a record that is meant

to be used for cross-layer attack against a different application that uses the same DNS cache. For instance, when an adversary cannot trigger queries via an application that it wishes to attack, it may often be able to trigger queries via a different application, that uses the same DNS cache. The adversary may also choose to inject into such cross-applications caches an application agnostic records; for instance, a malicious NS record, mapping the nameserver of a domain of the target application to the attacker’s IP address, is an example of an application agnostic record.

Such cross-applications DNS cache scenario is not uncommon. The DNS resolvers often serve multiple applications and the networks use the caching of the resolver to reduce traffic and latency for all the applications. We use open resolvers to check how common cross-applications DNS caches are. We perform our measurements against a list of open resolvers from censys [32] and probe their caches for the well-known domain(s) used by the applications on our list in Table 1, e.g., pool.ntp.org for NTP. For each application for which the records are in the cache we consider that the resolver is used by that application or by the clients of that application. We found that 69% of the open resolvers are shared between two or more of the applications on our list.

A recent study [45] analysed how an attacker can find third-party SMTP servers to trigger queries at typically closed forwarders used by web clients. By scanning the /24 network block of the resolver’s outbound IP address, the study found that an adversary could find an SMTP server which allows triggering queries from the same resolver in 11.3% of the cases. Additionally 2.3% of the resolvers were open resolvers in the first place.

4.3.3 Triggering queries via forwarders. In this section we show how to trigger queries with resolvers when this is not possible from the target application.

DNS forwarders make up the majority of open resolvers in the internet. Finding an open forwarder which forwards to the resolver of choice whose cache the adversary wishes to poison is not difficult. We explore the prevalence of forwarders through which one can force a given recursive DNS resolver to trigger a query. We perform a two step measurement: we first collect the forwarders used by open DNS resolvers and then which of these forwarders are used by random clients in the Internet.

In our measurement we use the list of all open DNS resolvers from Censys [32] (which performs a full IPv4 scan for open resolvers). We query all the resolvers for a custom query with a randomised subdomain under a domain which we control. This allows us upon the arrival of the DNS requests to our nameservers to map the open resolver’s IP address to the recursive forwarder that it uses. This forwarder is determined by the outbound IP address in the DNS query that arrives at our nameserver.

In the second step we run a web ad-based study against random clients in the Internet that download our object. We trigger DNS requests via those clients to our own domain. We use a random subdomain associated with each client. Per client, we then obtain the list of recursive resolvers’ IP addresses that arrived at our nameserver. We search them in the list of recursive resolvers IP addresses from our dataset of open resolvers.

Our results are as follows: focusing only on the IP addresses of the recursive resolvers, we find 4146 addresses out of which 3275

(79%) addresses are in the open resolver database. Consequently, assuming that an adversary targets a DNS resolver used by a typical web client (represented by the ad-net clients in our study), there is a high probability (79%) that it can find an open forwarder which can be used to poison the cache of the target victim recursive resolver used by that web client.

4.4 Applicability to Applications

In this section we explore which cache poisoning methodology is applicable to which of the applications listed in Table 1.

For all methodologies, the attacker requires the knowledge of the domain which is queried. In cases where the domain is pre-configured in the applications configuration ("config" in Table 1), this information needs to be fetched out of band.

4.4.1 HijackDNS. The adversary can hijack a sub-prefix or same-prefix of the victim AS. We explain the success probability of cache poisoning through both methods.

Sub-prefix hijack. The attacker can advertise a sub-prefix of the victim. The routers prefer more specific IP prefixes over less specific ones, hence this announcement will redirect all traffic for that sub-prefix to the attacker.

Same-prefix hijack. Same-prefix hijack occurs when the attacker hijacks a route to an existing IP prefix of the victim. The attacker can advertise the same prefix as the victim AS and depending on the local preferences of the ASes will intercept traffic from all the ASes that have less hops (shorter AS-PATH) to the attacker than to the victim AS. The success of the hijack depends on the topological relationship between the attacking AS and the domain and the victim resolver.

4.4.2 SadDNS. The attack is probabilistic since it depends on the ability of the adversary to win the race, by correctly guessing the randomised TXID before the timeout event. A prerequisite to a successful attack is the ability to trigger a large volume of queries. Typically, this is the case when the query domain can be set by the attacker ("target" in Table 1, Column "query name") or when a third party application is used to trigger the queries (marked with ✓² in Table 1, see Section 4.3.3).

4.4.3 FragDNS. FragDNS is also a probabilistic attack since its success depends on correctly guessing the IP ID value in the spoofed IP fragment. This is easy when systems have large IP defragmentation buffers, such as old linux versions which allows the adversary to send multiple fragments with different IP ID values, or when systems use incremental IP ID counters which can be predicted. A successful poisoning with FragDNS typically requires more packets than with prefix hijacks but less than with SadDNS attack.

4.5 Exploiting Poisoned Caches for Attacks

Applications that use DNS resolvers with poisoned caches are exposed to a range of attacks. In this section we explain the possible outcomes of the attacks via DNS cache poisoning.

Downgrade attacks. In downgrade attacks the attacker makes the security mechanism not available, as a result, causing the processing of the data to be performed without the additional information provided by the security mechanism. For instance, by poisoning the responses to queries for SPF or DKIM records the attacker can

trick the victim Email server into accepting phishing Emails or Emails with malicious attachments. Similarly, by causing the RPKI validation to fail, the adversary can make a network, that filters bogus BGP announcements with route origin validation, to accept hijacked prefixes as authentic. This is due to the fact that RPKI validation will result in status 'unknown' and hence will not be used.

The attacker can also trick a security mechanism via DNS cache poisoning. For instance, the attacker can bypass domain validation, by redirecting the validation to run against attacker's host [24], and hence can issue fraudulent certificates.

Hijack attacks. In hijack attacks the victims are redirected to attacker's host which impersonates a genuine service in the Internet. Network adversaries can hijack traffic to take over Internet resources, such as SSO accounts at public providers. For instance, the adversaries can take over the SSO accounts at Regional Internet Registries (RIRs), by exploiting a combination of DNS cache poisoning with password recovery [14]. The idea is to poison the cache of the RIR, and to inject a record that maps the victim LIR to the host of the attacker. Running a password recovery procedure causes the password for the victim SSO account to be sent to the attacker instead of the victim. As a result, the attacker can hijack the digital resources, such as IP addresses and domains, that belong to the victim LIR.

DoS attacks. The attacker can block connectivity, e.g., for radius clients or access to services, such as secure tunnels. The idea is that if the attacker cannot forge cryptographic material, such as a certificate to authenticate a radius client, it can redirect the client to the wrong host via cache poisoning, preventing the client from connecting to the genuine target service. The adversary will not be able to provide authenticated material which will result in a failure, and lack of service for the victim client.

5 INTERNET MEASUREMENTS

In this section, we analyse the fraction of the vulnerable resolvers and nameservers with respect to each DNS poisoning method. We evaluate properties which influence the success of the cross-layer attacks against applications. Our measurements in this section show that the vulnerabilities do not significantly differ for most of the application-specific datasets. The outliers can be summarised as follows:

- Vulnerabilities to BGP sub-prefix hijacking are exceptionally high for eduroam and low for RPKI domains. The cause may be inherent in networks' sizes (large in case of universities and small for RPKI repository operators) and accordingly use BGP announcements which are larger than /24 for large networks or equal to /24 for small networks.
- Vulnerabilities to fragmentation cache poisoning among open resolvers is low compared to other resolvers in our dataset. This may be due to the fact that the distribution of the open resolvers is skewed towards poorly configured devices which cannot handle fragmentation.
- Domains with MX, SRV, NAPTR (eduroam) records are more often vulnerable to fragmentation based cache poisoning than the

domains in the 1M-top Alexa dataset. One reason is that the responses to ANY queries result in much larger packets, which often exceed the minimum MTU limit.

5.1 Vulnerabilities in Resolvers

We test the DNS resolvers for vulnerabilities to the three cache poisoning methods (Section 3) for different applications. The results of our evaluations for all datasets and all poisoning methods are summarised in Table 3.

5.1.1 Dataset. For each application from Section 4, we gather datasets of resolvers used by the front-end systems (i.e., Web clients, Alexa MX records, etc.) of that application. To achieve this, we first look for an appropriate dataset of front-end systems and then trigger queries through those front-end systems. This allows us to discover and test the corresponding resolver.

For front-end systems, we use the following datasets, listed in Table 3: (1) Our local university eduroam service. (2) Password recovery of popular infrastructure service providers, consisting of: All 5 Regional Internet Registries, popular domain registrars used by Alexa Top 100K domains and popular cloud providers [1, 2, 4, 5]. (3) Domain validation of most popular Certificate authorities [3]. (4) Popular CDNs in Alexa Top 100K (by mapping A record to ASN). (5,6) SMTP and XMPP servers of Alexa Top 1M domains. (7) Web clients gathered via an Ad-network. (8) Open resolvers from Censys [32] and (9) a subset of those open resolvers who cache pool. ntp.org. This resulted in a dataset of 89,924 resolvers (back-end IP addresses) in 13,804 ASes associated with 33,418 prefixes.

We report the dataset size in terms of front-end systems (i.e., number of SMTP servers or number of open resolver front-end IP addresses) in column "Dataset size" of Table 3. For vulnerability, we report the percentage of vulnerable front-end systems which was measured as described in Section 5.1.2. When a front-end system uses multiple resolvers, we consider it vulnerable if any of the resolvers it uses is vulnerable.

5.1.2 Measuring cache poisoning vulnerabilities in resolvers. The results of our measurements and evaluations against resolvers for different poisoning methodology are summarised in Table 3. In the following sections we explain the measurements we carried out of each attack methodology against the resolvers in our dataset.

Sub-prefix BGP hijacks (HijackDNS). Since many networks filter BGP advertisements with prefixes more specific than /24, we consider an IP address hijackable if it lies inside a network block whose advertised size is larger than /24. We therefore map all the resolvers' IP addresses to network blocks and consider those vulnerable to sub-prefix hijacks whose advertisement is larger than /24, since an advertisement with a smaller prefix will always take precedence over a bigger one. For the remaining addresses, a BGP-hijack may still be possible using same-prefix hijacks. To infer the scope of DNS platforms potentially vulnerable to cache poisoning via BGP sub-prefix hijack attacks we perform Internet measurements checking for DNS platforms on prefixes less than /24. We collect information on the state of the global BGP table in the Internet with Routeview [71] and RIPE RIS [63] collectors. We analyse the BGP announcements seen in public collectors for identifying networks vulnerable to sub-prefix hijacks by studying the advertised prefixes

sizes. The measurements of resolvers vulnerable to BGP sub-prefix hijacks are listed in Table 3 and plotted in Figure 3.

Same-prefix BGP hijacks (HijackDNS). We perform simulations of same-prefix BGP hijacks using a set of randomly selected attacker and victim AS pairs using a simulator developed in [39] and Internet AS level topology downloaded from CAIDA [6]. The simulator selects Gao-Rexford policy compliant paths [35], and considers prefix lengths and AS-relationship (provider, customer and peer) and sizes (stub, small, medium and tier one). The attackers are randomly selected from all the ASes whereby the victim ASes are selected from our dataset of DNS resolvers and 1M-top Alexa domains. For each (attacker, victim)-pair we perform a simulation of same-prefix hijack that the attacker AS launches against a victim AS. If the attacking AS is closer to the victim, the attack succeeds. The simulation shows that the attacking AS was capable of hijacking the traffic in 80% of the evaluations.

SadDNS. To test resolvers vulnerable to SadDNS, we test the resolvers back-end IP addresses for a global ICMP message limit which allows to use the side-channel identified by [57]. To limit our dataset to functional resolvers which are still reachable, we furthermore send an ICMP echo-response ('ping') packet to the resolver first. This is especially important for the open resolvers dataset, since this dataset tends to include resolvers operating from dynamic IP addresses, which may have changed since the dataset was collected.

For the open resolver dataset we measured a vulnerability rate of 12%, a notable reduction from the 35% vulnerability rate of the original paper [57]. This difference could be influenced by various factors including the fact that our dataset contained more resolvers than [57]. The crucial difference is likely that our study was conducted after the vulnerability which allowed the global rate limit to be exploited was patched in many systems. For example, all updated versions of Ubuntu should have been patched by the time we carried out our evaluations¹.

FragDNS. To test vulnerability to fragmentation-based DNS cache poisoning, we use a custom nameserver application which will always emit fragmented responses padded to a certain size to reach the tested fragment size limit. The nameserver is configured to only send CNAME responses in the first fragmented response. This means that if the resolver receives a fragmented response, it needs to re-query for the CNAME-alias. This allows us to verify that the answer arrived at the resolver and thus, that the resolver is vulnerable to this type of attack.

Using this setup, we test all resolvers by triggering queries to our nameservers and observe if the fragmented responses are accepted. In our bigger datasets, vulnerability rates range between 31% for Open resolvers and 91% for Ad-net resolvers. For the smaller datasets, we still observe many vulnerable services. However, all certificate authorities' resolvers in our dataset rejected our fragmented responses, maybe attributed to the fact that this attack method was already evaluated and disclosed to CAs previously [24]. We report results for all datasets and all poisoning methods in Table 3.

¹<https://ubuntu.com/security/CVE-2020-25705>

Dataset	Protocol	Vulnerable against			Dataset size
		BGP hijack sub-prefix	Sad-DNS	Fragment	
(1) Local university	Radius	100%	0%	100%	1
(2) Popular services	PW-recovery	93%	16%	90%	29
(3) Popular CAs	DV	75%	0%	0%	5
(4) Popular CDNs	CDN	100%	0%	25%	4
(5) Alexa 1M SRV	XMPP	73%	1%	57%	476
(6) Alexa 1M MX	SMTP SPF DKIM DMARC	79%	9%	56%	61,036
(7) Ad-net study	HTTP DANE OCSP	70%	11%	91%	5,847
(8) Open resolvers	All	74%	12%	31%	1,583,045
(9) Cache test	NTP	79%	9%	32%	448,521

Table 3: Vulnerable resolvers.

5.2 Vulnerabilities in Domains

In this section we perform measurements of the vulnerabilities in domains to our cache poisoning methodologies for different applications. We collect lists of the domains associated with these applications and test all the nameservers serving each domain according to the properties required for each cache poisoning method (from Section 3). The results of our evaluations and measurements for all the tested datasets and poisoning methods are summarised in Table 4.

Dataset	Protocol	Vulnerable against				DNS SEC	Total
		BGP hijack sub-prefix	Sad-DNS	Fragment Any	Fragment Global		
(1) Eduroam list	Radius	96%	11%	44%	18%	10%	1,152
(2) Alexa 1M	HTTP DANE DV	53%	12%	4%	1%	2%	877,071
(3) Alexa 1M MX	SMTP SPF DKIM DMARC	44%	6%	7%	1%	3%	63,726
(4) Alexa 1M SRV	XMPP	44%	4%	29%	5%	7%	2,025
(5) RIR whois	PW-	59%	9%	14%	4%	4%	58,742
(6) Registrar whois	recovery	51%	10%	23%	5%	6%	4,628
(7) Well-known	NTP	25%	0%	25%	25%	25%	9
(8) Well-known	Cryptocurrency	28%	17%	21%	3%	21%	32
(9) Well-known	RPKI	14%	0%	0%	0%	67%	8
(10) Cert. Scan	IKE OpenVPN	51%	11%	5%	1%	7%	307

Table 4: Vulnerable domains.

5.2.1 *Dataset.* For each application in Section 4, we collect datasets of typical domains looked up by clients (or servers) of that application. We collect such domains from the following data sources, listed in Table 4:

(1) Eduroam institution lists from United Kingdom [46], Germany [30] and Austria [15]. (2) Alexa Top 1 Million domains, including

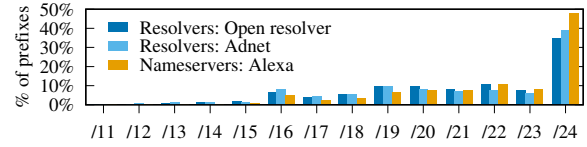


Figure 3: Announced prefixes.

subsets of domains which have (3) MX and (4) SRV (XMPP) records. Domains from account email addresses from whois databases of (5) RIRs and (6) Registrars. (7) Well-known NTP server domains. (8) Well-known cryptocurrency domains. (9) Well-known RPKI validator database domains. (10) Domains of IKE and OpenVPN servers’ certificates. This resulted in 904,555 domains hosted on 200,086 nameservers in 24,353 ASes associated with 60,511 prefixes.

5.2.2 Measuring cache poisoning vulnerabilities in nameservers.

HijackDNS. We perform a similar analysis as in Section 5.1.2, to check the vulnerabilities to BGP prefix hijacks. The results are plotted in Figure 3. The differences between the fractions of nameservers in 1M-top Alexa domains that can be sub-prefix hijacked are not significantly different than those of the resolvers.

The resilience of the DNS infrastructure to BGP hijack attacks is also a function of the distribution and the topological location of the nameservers in the Internet. We measured the characteristics of the nameservers from the Internet routing perspective. Our findings show that the nameservers are concentrated in just a few ASes. Our measurements show that 80% of the ASes host less than 10% of the nameservers, and the rest of the nameservers are concentrated on the remaining ASes. This concentration of the nameservers on a few ASes, typically CDNs, makes it easier to intercept traffic of multiple nameservers with a single prefix hijack.

SadDNS. For a nameserver to be vulnerable to side-channel attack (Section 3.2), the attacker must be able to ‘mute’ the nameserver to extend the time-window for the attack. This is achieved by abusing rate-limiting in nameservers. To find out if a nameserver supports rate-limiting, we use the following methodology: we send to the nameserver a burst of 4000 queries in one second, and see if this stops (or reduces) the subsequent responses received from this server. We consider a nameserver to be vulnerable if we can measure a reduction in responses after the burst.

Fragmentation. We evaluate the vulnerability to fragmentation-based poisoning in nameservers and domains by testing three properties required to create a sufficiently large fragment in order to inject malicious records into it: (1) support of IMCP fragmentation needed, (2) record types for optimising response size, (3) by bloating the queried domain and (4) fitting the response into the limitation of EDNS.

PMTUD. We first check for the support of path MTU discovery (PMTUD) with ICMP fragmentation needed: we send to the nameserver an ICMP fragmentation needed packet, which indicates that the nameserver should fragment packets sent to our test host. Then we send queries of different type to that domain. We consider a nameserver vulnerable if the responses return fragmented.

Record types. We evaluated fragmentation with three record types: ANY, A and MX. We use DNS requests of type ANY to increase

Implementation	Vulnerable	Note
BIND 9.14.0	yes	cached
Unbound 1.9.1	no	doesn't support ANY at all
PowerDNS Recursor 4.3.0	yes	cached
systemd resolved 245	yes	cached
dnsmasq-2.79	no	not cached

Table 5: ANY caching results of popular resolvers.

the response size above the fragmentation limit of the nameserver. We find that for 19.50% of domains in 1M-top Alexa there is at least one nameserver which emits fragmented DNS responses, which can be used for cache poisoning attacks via injection of IP fragments. We plot the minimum fragmenting size emitted by those nameservers in Figure 4, which shows that most affected nameservers (83.2%) fragment DNS responses down to a size of 548 bytes and 7.05% even down to 292 bytes. We tested ANY response caching in 5 of the most popular resolver implementations and found that 3 out of 5 use the contents of an ANY response, to answer subsequent A queries, without issuing further queries (See Table 5). Namely, the adversaries can often launch cache poisoning attacks by issuing queries for ANY record type in the domain.

However, only open resolvers (or forwarders) allow the attacker to trigger ANY queries. We repeat the same study using queries for A record type and then for MX record type, which are the query types typically triggered using the other query-triggering methods, such as via email or a script in a browser. We get vulnerability rates of 0.29% and 0.44% respectively due to the smaller response sizes which are often not sufficiently large to reach the nameserver's minimum fragment size. However, these numbers represent the lower bound.

Bloat query. The attacker can bloat the queries by concatenating multiple subdomains which increases the responses sizes. The maximum increase is up to 255 characters. The labels are limited to max 63 characters (+1 for the label delimiter) and the attacker can concatenate four subdomains: 4*64 (minus the parent domain). This increases the vulnerable resolvers to above 10%.

Fitting into response. Additionally to the requirement that the DNS response size must be big enough to trigger fragmentation on the nameserver side, it must also be small enough to fit in the resolvers maximum response size advertised in EDNS.

To evaluate this, we measure the EDNS UDP size of more than 1.5K open resolvers collected from Censys [32] IPv4 Internet scans. We query each resolver by triggering a query to our own nameserver and measure the EDNS UDP payload size advertised in the query. The results are shown in Figure 4. Approximately 40% of the resolvers support UDP payload sizes of up to 512 bytes, while 50% of the resolvers advertise a payload size equal or larger than 4000 bytes. The remaining 10% are between 1232 and 2048 bytes. Given the minimum MTU size measurement of the nameservers in 1M-top Alexa domains in Figure 4, this means that the resolver population is essentially portioned in two groups: one group (40%) which is vulnerable to poisoning attacks with 7% of all vulnerable domains and one group (50-60%) which is vulnerable to poisoning attack with all the vulnerable domains.

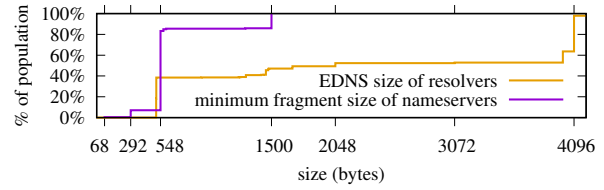


Figure 4: CDF of resolver EDNS UDP size vs. minimum fragment size emitted by nameservers.

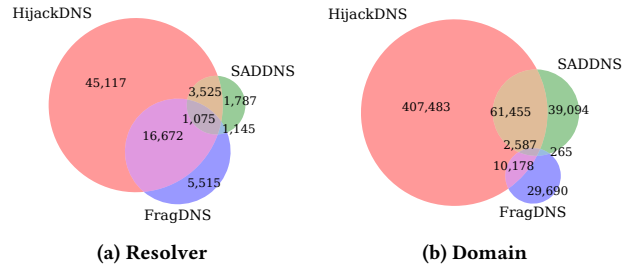


Figure 5: Venn diagram of all vulnerable resolvers (by number of back-end addresses) and domains.

5.3 Comparative Analysis

Our measurements show that the methodologies for DNS cache poisoning can often result in practical attacks, depending on the setup, network conditions and server configurations. In this section we compare the DNS cache poisoning methodologies with respect to stealthiness, effectiveness and applicability.

The main insights of the experimental measurements that we performed using each of the methods in Section 3 are summarised in Table 6. The columns in Table 6 correspond to the attacks we carried out against the domains and resolvers in our dataset (see Section 5).

	BGP Hijack		SadDNS	Fragmentation	
	sub-	same-		any IPID	global IPID
Applicability					
Vuln. resolvers	70%	80%	11%	and 91%	and 1%
Vuln. domains	or 53%	or 70%	and 12%	4%	
Effectiveness					
Hitrate	100%	0.2%	0.1%	20%	
Queries needed	1	497	1024	5	
Total traffic (pkts)	2	987K	65K	325	
Stealthiness					
Visibility	very visible	visible	stealthy, but locally detectable (Packet flood)	very stealthy	
Additional requirements					
Additional requirements	none	none	max(resolver EDNS size) < min(nameserver MTU)		

Table 6: Comparison of the cache poisoning methods.

5.3.1 Applicability. A method is applicable against a resolver for some domain if it results in a practical DNS cache poisoning attack. The applicability for each method for resolvers and domains is listed in Table 6.

To compare the applicability of the methodologies we use the results of our internet measurements (Tables 3 and 4) and take the numbers for the ad-Net resolvers and 1M-top Alexa domains datasets. We also show the absolute number of all vulnerable resolvers (according to a back-end address) and domains in all our datasets in Figure 5. This figure shows that the number of resolvers and domains vulnerable to HijackDNS is by far the highest, while SadDNS has more vulnerable domains and FragDNS has more vulnerable resolvers. The overlaps between the vulnerable domains and resolvers can be seen as expected for a distribution of unrelated properties, i.e., SadDNS and FragDNS have a significant overlap with HijackDNS, which is due to the fact that 53-70% of the systems we measured are vulnerable to HijackDNS, while SadDNS and FragDNS only have a small overlap compared to number of vulnerable systems in each category. Only 11% of the DNS resolvers and 12% of the domains are vulnerable to SadDNS attack. Many more resolvers are vulnerable to injection of content via IPv4 fragments, hence FragDNS attack is more applicable than SadDNS. In addition, due to its large size, the open resolver dataset dominates the results in our comparison.

5.3.2 Effectiveness. Attack effectiveness is demonstrated with the traffic volume needed for a successful attack, which is a function of the number of queries that should be triggered for a successful attack. The larger the attack volume, the less stealthy the attack is. We define hitrate as the probability to poison the target DNS cache with a single query and calculate the expected number of queries for each of the poisoning methods by inversion of the hitrate. We estimate the expected number of packets sent to the resolver by multiplying this with the traffic volume generated per query. For SadDNS where the amount of traffic during the attack is not stable, we analyse the experimental data for the amount of traffic needed.

HijackDNS. If an AS prefers a malicious BGP announcement of the adversary to the announcement of the victim AS, then the attack is effective, requiring only a single packet to send a malicious BGP announcement and then another packet to send a spoofed DNS response with malicious DNS records.

SadDNS. Using our implementation of SadDNS attack from Section 3.2 we find that the DNS cache poisoning with SadDNS succeeds after an average of 471 seconds (min: 39 seconds, max: 779 seconds). This is inline with the results in [57] which report an average of 504 seconds. To achieve a successful attack we needed to run 497 iterations on average. This is correlated with the attack duration since we do not trigger more than two queries per second. When more queries within one attack iteration are triggered, the resolvers respond with `servfail`. By inverting this number we get a hitrate of 0.2%. Notably however, since most of the queries do not result in attack windows of meaningful length, an attacker should be able to optimise the attack by analysing the exact back-off strategies used by the target resolver, and adjusting the queries according to this.

Using the results from our SadDNS experiment, we also obtain statistics for how many packets are sent to the target resolver. On average, our implementation sent 986,828 packets or 88MB of traffic, which is again, comparable to the original attack (69MB in [57]).

FragDNS. Only about 1% of the domains allow deterministic fragmentation-based cache poisoning attacks thanks to slowly incremental global IPID counter in nameservers. More than 4% of the domains are vulnerable to probabilistic attacks by attempting to hit an unpredictable IPID counter and to match the UDP checksum. When the IPID values are not predictable, the probability to hit the correct value is roughly 0.1%. To match the UDP checksum, the attacker needs to predict the partial UDP checksum of the second fragment of response sent by the nameserver. This means that the probability to match the UDP checksum is the inverse of the number of possible second fragments emitted by the nameserver (assuming equal distribution).

To calculate the per-nameserver hitrate of FragDNS attack for each domain we calculate the product of both probabilities, matching the IPID as well as matching the UDP checksum. We take the average of these per-nameserver hitrates to calculate a per domain hitrate. The results of our evaluation are: when the nameservers use a single global counter for IPID, depending on the rate at which queries arrive at the nameserver, the median hitrate over all vulnerable domains (for different rates of queries from other sources) is 20%. When the nameserver selects IPID values pseudorandomly, the median hitrate is 0.1% which is the probability to correctly guess the IPID, as most servers do not randomise the records in DNS responses.

FragDNS attack also requires large traffic volumes with 1024 packets median computed over vulnerable domains with 65K packets for an unpredictable IPID, and with only 325 packets on average against a predictable IPID against high load servers, such as the servers of top-level domains.

In the worst case, the attack requires 64 packets to fill the resolver IP-defragmentation buffer and another packet to trigger the query. Combined with a 0.1% success rate, this translates to an average of 65,000 packets.

5.3.3 Stealthiness. In BGP prefix hijacks malicious BGP announcements manipulate the control plane and a single BGP announcement suffices to change the forwarding information in the routers. BGP prefix hijacks generate lower traffic volume when performing the hijack but may be more visible in the Internet since the attack impact is more global. The more networks are affected as a result of the BGP hijack the higher the chance is that such attacks may be detected. Same-prefix hijack is more stealthy in control plane than sub-prefix hijack since it does not affect the global routing BGP table in the Internet, but causes manipulations only locally at the ASes that accept the malicious announcement. Furthermore, as we already mentioned, short-lived BGP hijacks typically are ignored and do not trigger alerts [23, 48, 49]. In contrast, guessing the source port with SadDNS method (Section 3.2) or injecting malicious payload via IPv4 fragmentation (Section 3.3) generate more traffic than BGP hijacks, but only locally on the network of the victim DNS resolver or the target nameserver. In contrast to BGP hijacks the attack is performed on the data plane, and is hence not visible in the global BGP routing table in the Internet.

SadDNS attack creates a large traffic volume and hence may be detected by the affected networks. FragDNS attacks against domains that uses a global sequentially incremental IPID counter are the stealthiest.

6 COUNTERMEASURES

Almost all Internet systems, applications and even security mechanisms use DNS. As we showed, a vulnerable DNS introduces not only threats to systems using it but also to security mechanisms, such as PKI. We provide recommendations to mitigate that threat.

We also set up a tool at <https://crosslayerattacks.sit.fraunhofer.de> to allow clients to check if their networks are operating DNS platforms vulnerable to the cache poisoning attacks evaluated in our work. In the rest of this section we separately explain our recommendations for DNS servers to prevent cache poisoning attacks and then for applications to prevent cross-layer attacks.

6.1 DNS servers

In addition to recommendations and best practices for patching DNS servers, such as those in [RFC5452] [43], we recommend a new countermeasure we call **security by obscurity**. Our experience of cache poisoning evaluation in the Internet showed that the less information the adversary has, the more hard it becomes to launch the attacks in practice. Security by obscurity proves effective not only against off-path but also against on-path MitM attacks. Although it is a known bad practice in cryptography it turns out useful in practice. Specifically, for launching the attacks the attackers need to collect intelligence about the target victims, such as which caching policies are used, which IP addresses are assigned to the resolver - randomising or blocking this information, will make a successful attack harder. The network administrators can deploy countermeasures to make such information difficult to leak, e.g., DNS resolvers should use multiple caches with different DNS software on each, resolvers should not send ICMP errors, nameservers should randomise records in responses.

Preventing queries. Server operators might choose to configure systems to do less (or no) DNS lookups, ie. in the case of email servers. This reduces the chance an attacker can trigger a query to start the poisoning.

Blocking fragmentation. Resolver operators can block fragmented responses in firewalls to reduce the applicability of FragDNS attacks. Some operators only implement filtering of small fragments (i.e., Google's 8.8.8.8) which can prevent the attack since the attacker might not be able to cause a nameserver response of the size needed to reach the filtering limit.

Randomise DNS responses. Randomising nameserver responses complicates the FragDNS attack as the attacker needs to predict the UDP checksum of the original nameserver's response.

0x20 Encoding. 0x20 Encoding adds entropy to the DNS query which must be matched by the response. This complicates the SadDNS attack to a point where it is no longer viable (ie. adding 0x20 Encoding to a domain with 16 alphanumeric characters adds 16 bits of entropy to the query). Since this randomness is only contained in the question section of the DNS packet, it cannot prevent the FragDNS attack as it will be in the first fragment along with the TXID.

Securing BGP. Full deployment of RPKI (together with BGPsec) would prevent the HijackDNS attack. However, because of several deployment barriers, most of the prefixes are not protected by RPKI and most ASes do not enforce Route Origin Validation (ROV)

[36, 40, 62]. We refer to [39] for a comprehensive discussion of the deployment issues.

6.2 Applications

In the rest of this section we provide recommendations for preventing cross-layer attacks that use DNS cache poisoning.

Separate resolvers and caches. It is common in networks to use one DNS resolver for multiple services and servers. Our attacks exploit that. We recommend using different DNS resolvers (each with a distinct cache) for each system.

Third party authentication (TLS). Third party authentication, like TLS, can mitigate the attacks against all DNS use-cases which aim to locate a server (i.e.m federation and address lookup use-cases). However, even such mechanisms can only reduce the harm of DNS poisoning, but not completely mitigate it, e.g., adversaries can use DNS cache poisoning to subvert the security of DV during certificates issuance. Furthermore, an attacker can still use cache-poisoning for DoS attacks.

Two factor authentication. Should be enabled by default (and not optional as it is now). This would prevent the attacker from getting access to the account even if it has acquired the login credentials for the victim.

Secure fallback. Instead of allowing a transaction when no information about its authorisation state can be gathered (like done currently in SPF and RPKI) a security-mechanism could decide to not allow it. This however would mean that attacking the availability of DNS for a certain domain would allow DoS attacks instead, preventing a resolver from looking up a domain's SPF records would prevent that domain from sending any emails to the servers using this resolver.

7 CONCLUSIONS

We evaluated methodologies for launching practical DNS cache poisoning attacks and derived insights on the applicability, effectiveness and stealth of these attacks. We then applied the methodologies for a systematic evaluation of cross-layer attacks against popular applications.

Our work demonstrates the significant role that DNS plays in the Internet for ensuring security and stability of the applications and clients. If DNS is vulnerable, our work shows that in addition to traditional attacks, such as redirection to adversarial hosts, weak off-path adversaries can even downgrade protection of security mechanisms, such as RPKI or DV. We provide recommendations for mitigations and developed a public tool at <https://crosslayerattacks.sit.fraunhofer.de> to enable clients to identify vulnerabilities in DNS platforms on their networks.

ACKNOWLEDGEMENTS

We thank the anonymous referees for thoughtful feedback on our work. This work has been co-funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) SFB 1119.

REFERENCES

- [1] [n.d.]. Best Infrastructure as a Service (IaaS) Providers. <https://www.g2.com/categories/infrastructure-as-a-service-iaas>. Accessed: 2020-10-09.
- [2] [n.d.]. Market Share Analysis: IaaS and IUS, Worldwide, 2018. <https://www.gartner.com/en/newsroom/press-releases/2019-07-29-gartner-says-worldwide-iaas-public-cloud-services-market-grew-31point3-percent-in-2018>. Accessed: 2020-10-09.
- [3] [n.d.]. Market share trends for SSL certificate authorities. https://w3techs.com/technologies/history_overview/ssl_certificate. Accessed: 2020-10-09.
- [4] [n.d.]. Quarterly Cloud Spending Blows Past \$30B; Incremental Growth Continues to Rise. <https://www.srgresearch.com/articles/quarterly-cloud-spending-blows-past-30b-incremental-growth-continues-rise>. Accessed: 2020-10-09.
- [5] [n.d.]. Top IaaS Providers: 42 Leading Infrastructure-as-a-Service Providers to Streamline Your Operations. <https://stackify.com/top-iaas-providers/>. Accessed: 2020-10-09.
- [6] 2011. The CAIDA AS Relationships Dataset, 2011. <http://www.caida.org/data/active/as-relationships/>.
- [7] 2015. Hacked or Spoofed: Digging into the Malaysia Airlines Website Incident. <https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/hacked-or-spoofed-digging-into-the-malaysia-airlines-website-compromise>. Accessed: 2021-1-19.
- [8] 2015. Webnic Registrar Blamed for Hijack of Lenovo, Google Domains. <https://krebsonsecurity.com/2015/02/webnic-registrar-blamed-for-hijack-of-lenovo-google-domains/>. Accessed: 2021-1-19.
- [9] 2018. DNS Spionage Campaign Targets Middle East. <https://blog.talosintelligence.com/2018/11/dnsponage-campaign-targets-middle-east.html>. Accessed: 2021-01-19.
- [10] 2019. Global DNS Hijacking Campaign: DNS Record Manipulation at Scale. <https://www.fireeye.com/blog/threat-research/2019/01/global-dns-hijacking-campaign-dns-record-manipulation-at-scale.html>. Accessed: 2021-1-19.
- [11] 2019. Sea Turtle keeps on swimming, finds new victims, DNS hijacking techniques. <https://blog.talosintelligence.com/2019/07/sea-turtle-keeps-on-swimming.html>. Accessed: 2021-01-19.
- [12] 2019. "Unprecedented" DNS Hijacking Attacks Linked to Iran. <https://threatpost.com/unprecedented-dns-hijacking-attacks-linked-to-iran/140737/>
- [13] 2020. Security Incident on November 13, 2020. <https://blog.liquid.com/security-incident-november-13-2020>. Accessed: 2021-01-19.
- [14] 2021. The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association. <https://www.usenix.org/conference/useenixsecurity21/presentation/dai-aconet>. [n.d.]. eduoam-Teilnehmer in Österreich. https://www.aco.net/eduoam_teilnehmer.html. Accessed: 2020-12-02.
- [15] F. Alharbi, J. Chang, Y. Zhou, F. Qian, Z. Qian, and N. Abu-Ghazaleh. 2019. Collaborative Client-Side DNS Cache Poisoning Attack. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 1153–1161. <https://doi.org/10.1109/INFOCOM.2019.8737514>
- [17] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. 2017. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 375–392.
- [18] Dan J. Bernstein. 2002. DNS Forgery. Internet publication at <http://cr.yyp.to/djb/dns/forgery.html>.
- [19] Robert Beverly and Steven Bauer. 2005. The Spoofing project: Inferring the extent of source address filtering on the Internet. In *Usenix Sruti*, Vol. 5. 53–59.
- [20] Robert Beverly, Arthur Berger, Young Hyun, and K Claffy. 2009. Understanding the efficacy of deployed internet source address validation filtering. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. 356–369.
- [21] Robert Beverly, Ryan Koga, and KC Claffy. 2013. Initial longitudinal analysis of IP source spoofing capability on the Internet. *Internet Society* (2013), 313.
- [22] Henry Birge-Lee, Yixin Sun, Anne Edmondson, Jennifer Rexford, and Prateek Mittal. 2018. Bambooing Certificate Authorities with BGP. In *27th USENIX Security Symposium (USENIX Security 18)*. 833–849.
- [23] Peter Boothe, James Hiebert, and Randy Bush. 2006. Short-lived prefix hijacking on the Internet. In *Proc. of the NANOG 36* (2006).
- [24] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. 2018. Domain Validation++ For MitM-Resilient PKI. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2060–2076.
- [25] R Bush and R Austein. 2013. RFC 6810: The Resource Public Key Infrastructure (RPKI) to Router Protocol.
- [26] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. A Longitudinal, End-to-End View of the DNSSEC Ecosystem. In *26th USENIX Security Symposium (USENIX Security 17)*. 1307–1322.
- [27] Taejoong Chung, Roland van Rijswijk-Deij, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. Understanding the role of registrars in DNSSEC deployment. In *Proceedings of the 2017 Internet Measurement Conference*. 369–383.
- [28] D. Madory. 2018. Recent Routing Incidents: Using BGP to Hijack DNS and more. https://www.lacnic.net/innovaportal/file/3207/1/dougmadory_lacnic_30_rosario.pdf
- [29] David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. 2008. Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries. In *ACM Conference on Computer and Communications Security*, Peng Ning, Paul F. Syverson, and Somesh Jha (Eds.). ACM, 211–222. <http://doi.acm.org/10.1145/1455770.1455798>
- [30] DFN-Verein. [n.d.]. Karte der aktuellen eduoam Standorte in Deutschland. <https://www.dfn.de/dienstleistungen/eduoam/>. Accessed: 2020-12-02.
- [31] Roger Dingleline, Nick Mathewson, and Paul Syverson. 2004. *Tor: The second-generation onion router*. Technical Report. Naval Research Lab Washington DC.
- [32] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. 2015. A Search Engine Backed by Internet-Wide Scanning. In *22nd ACM Conference on Computer and Communications Security*.
- [33] Tony Finch, Evan Hunt, Peter van Dijk, Anthony Eden, and Willem Mekking. 2019. *Address-specific DNS aliases (ANAME)*. Internet-Draft draft-ietf-dnsop-aname-03. IETF Secretariat. <http://www.ietf.org/internet-drafts/draft-ietf-dnsop-aname-03.txt>.
- [34] Pedro Franco. 2014. *Understanding bitcoin*. Wiley Online Library.
- [35] Lixin Gao and Jennifer Rexford. 2001. Stable Internet routing without global coordination. *IEEE/ACM Transactions on networking* 9, 6 (2001), 681–692.
- [36] Yossi Gilad, Avichai Cohen, Amir Herzberg, Michael Schapira, and Haya Shulman. 2017. Are We There Yet? On RPKI's Deployment and Security. In *NDSS*.
- [37] Amir Herzberg and Haya Shulman. 2012. Security of Patched DNS. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*. 271–288.
- [38] Amir Herzberg and Haya Shulman. 2013. Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In *IEEE CNS 2013. The Conference on Communications and Network Security, Washington, D.C., U.S. IEEE*.
- [39] Tomas Hlavacek, Italo Cunha, Yossi Gilad, Amir Herzberg, Ethan Katz-Bassett, Michael Schapira, and Haya Shulman. 2020. DISCO: Sidestepping RPKI's Deployment Barriers. In *Network and Distributed System Security Symposium (NDSS)*.
- [40] Tomas Hlavacek, Amir Herzberg, Haya Shulman, and Michael Waidner. 2018. Practical Experience: Methodologies for Measuring Route Origin Validation. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018*. 634–641. <https://doi.org/10.1109/DSN.2018.00070>
- [41] P Hoffman and P McManus. 2018. RFC 8484: DNS Queries over HTTPS (DoH).
- [42] Z Hu, L Zhu, J Heidemann, A Mankin, D Wessels, and P Hoffman. 2016. RFC 7858-Specification for DNS over Transport Layer Security (TLS).
- [43] A. Hubert and R. van Mook. 2009. *Measures for Making DNS More Resilient against Forged Answers*. RFC 5452. RFC Editor.
- [44] Philipp Jeitner and Haya Shulman. 2021. Injection Attacks Reloaded: Tunnelling Malicious Payloads over DNS. In *30th USENIX Security Symposium (USENIX Security 21)*.
- [45] Philipp Jeitner, Haya Shulman, and Michael Waidner. 2020. The Impact of DNS Insecurity on Time. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 266–277.
- [46] Jisc. [n.d.]. Organisations participating in eduoam in the UK. <https://www.jisc.ac.uk/eduoam/participating-organisations>. Accessed: 2020-12-02.
- [47] Dan Kaminsky. 2008. It's the End of the Cache As We Know It. Presentation at Blackhat Briefings.
- [48] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. 2008. Autonomous security for autonomous systems. *Computer Networks* 52, 15 (2008), 2908–2923.
- [49] Varun Khare, Qing Ju, and Beichuan Zhang. 2012. Concurrent prefix hijacks: Occurrence and impacts. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, 29–36.
- [50] Amit Klein. 2007. BIND 9 DNS cache poisoning. *Report, Trustee, Ltd* 3 (2007).
- [51] Amit Klein. 2007. Windows DNS Server Cache Poisoning.
- [52] Amit Klein. 2020. Cross Layer Attacks and How to Use Them (for DNS Cache Poisoning, Device Tracking and More). *arXiv preprint arXiv:2012.07432* (2020).
- [53] Amit Klein, Haya Shulman, and Michael Waidner. 2017. Internet-wide study of DNS cache injections. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 1–9.
- [54] Matt Lepinski, Stephen Kent, and Derrick Kong. 2012. A profile for route origin authorizations (ROAs). *IETF, RFC 6482* (2012).
- [55] Qasim Lone, Matthew Luckie, Maciej Korczykński, Hadi Asghari, Mobin Javed, and Michel van Eeten. 2018. Using Crowdsourcing Marketplaces for Network Measurements: The Case of Spoofing. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 1–8.
- [56] Matthew Luckie, Robert Beverly, Ryan Koga, Ken Keys, Joshua A Kroll, and k claffy. 2019. Network Hygiene, Incentives, and Regulation: Deployment of Source Address Validation in the Internet. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 465–480.
- [57] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. 2020. DNS Cache Poisoning Attack Reloaded: Revolutions with

- Side Channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (Virtual Event, USA) (CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1337–1350. <https://doi.org/10.1145/3372297.3417280>
- [58] Jared Mauch. 2013. Open resolver project. In *Presentation, DNS-OARC Spring 2013 Workshop (Dublin)*.
- [59] P. Mockapetris. 1987. *Domain names - concepts and facilities*. STD 13. RFC Editor. <http://www.rfc-editor.org/rfc/rfc1034.txt> <http://www.rfc-editor.org/rfc/rfc1034.txt>.
- [60] P. Mockapetris. 1987. *Domain names - implementation and specification*. STD 13. RFC Editor. <http://www.rfc-editor.org/rfc/rfc1035.txt> <http://www.rfc-editor.org/rfc/rfc1035.txt>.
- [61] Pradosh Mohapatra, John Scudder, David Ward, Randy Bush, and Rob Austein. 2013. BGP prefix origin validation. In *IETF RFC 6811*.
- [62] Andreas Reuter, Randy Bush, Ítalo Cunha, Ethan Katz-Bassett, Thomas C. Schmidt, and Matthias Wählisch. 2017. Towards a Rigorous Methodology for Measuring Adoption of RPKI Route Validation and Filtering. *CoRR abs/1706.04263* (2017). [arXiv:1706.04263](https://arxiv.org/abs/1706.04263) [http://arxiv.org/abs/1706.04263](https://arxiv.org/abs/1706.04263)
- [63] RIPE NCC. 2021. RIS Raw Data. <https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-raw-data>
- [64] S. Goldberg. 2018. The myetherwallet.com hijack and why it's risky to hold cryptocurrency in a webapp. <https://medium.com/@goldbe/the-myetherwallet-com-hijack-and-why-its-risky-to-hold-cryptocurrency-in-a-webapp-261131fad278>
- [65] Haya Shulman. 2014. Pretty bad privacy: Pitfalls of DNS encryption. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. 191–200.
- [66] Haya Shulman and Michael Waidner. 2015. Towards security of internet naming infrastructure. In *European Symposium on Research in Computer Security*. Springer, 3–22.
- [67] Haya Shulman and Michael Waidner. 2017. One Key to Sign Them All Considered Vulnerable: Evaluation of DNSSEC in the Internet. In *NSDI*. 131–144.
- [68] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. 2019. Encrypted DNS -> Privacy? A Traffic Analysis Perspective. *arXiv preprint arXiv:1906.09682* (2019).
- [69] Jonathan Spring. [n.d.]. Probable Cache Poisoning of Mail Handling Domains. <https://insights.sei.cmu.edu/cert/2014/09/-probable-cache-poisoning-of-mail-handling-domains.html>.
- [70] Yixin Sun, Maria Apostolaki, Henry Birge-Lee, Laurent Vanbever, Jennifer Rexford, Mung Chiang, and Prateek Mittal. 2021. Securing internet applications from routing attacks. *Commun. ACM* 64, 6 (2021), 86–96.
- [71] University of Oregon. 2012. Route Views Project. <http://bgplay.routeviews.org/>.
- [72] Paul Vixie. 1995. DNS and BIND Security Issues. In *Proceedings of the 5th Symposium on UNIX Security*. USENIX Association, Berkeley, CA, USA, 209–216.
- [73] S Weiler and D Blacka. 2013. RFC 6840: Clarifications and Implementation Notes for DNS Security (DNSSEC). *IETF Standard* (2013).
- [74] Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan, and Zhiyun Qian. 2020. Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices. In *29th USENIX Security Symposium (USENIX Security 20)*. 577–593.

A.7. The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources

[Dai21u]

Tianxiang Dai et al. “The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3147–3164. ISBN: 978-1-939133-24-3. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/dai>

Declaration of Contributions

The paper "The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources" was published as a full research paper at the "30th USENIX Security Symposium (USENIX Security 21)". It constitutes a joint work of Tianxiang Dai, Philipp Jeitner, Haya Shulman and Michael Waidner.

Haya Shulman proposed initial concept and structured the paper. Haya Shulman also wrote Introduction, Background and Conclusions, while Philipp Jeitner wrote the rest. After the initial round of reviews, Haya Shulman and Philipp Jeitner decided on what to change about the paper and to add new attack methodologies and other types of Internet resource providers. Tianxiang Dai and Philipp Jeitner designed, implemented and analysed the measurements together. More specifically, on the provider and resolver side, Tianxiang Dai developed the testing infrastructure and performed the SADDNS measurement for all. He did the resolver tests on RIRs and CAs, while Philipp Jeitner did the resolver tests on Registrars and IaaS providers. On the customer and nameserver side, Tianxiang Dai performed all the measurements. The data analysis was done by Tianxiang Dai and Philipp Jeitner together, where Philipp Jeitner did most of the analysis. Philipp Jeitner contributed the list of countermeasures. Tianxiang Dai also implemented a proof-of-concept attack and demonstrated it on one test account of a chosen RIR. Michael Waidner was a general advisor of this work and contributed with continuous feedback during all phases of the paper writing process. The paper was presented at the conference by Philipp Jeitner. The work was later presented at community event NANOG 83 and RIPE 83 by Tianxiang Dai, also as a guest post at APNIC Blog.

All authors agree with the use of their joint paper as part of Philipp Jeitner’s and Tianxiang Dai’s cumulative dissertation, considering a contribution of 60% from Philipp Jeitner and 40% from Tianxiang Dai.



The *Hijackers* Guide To The Galaxy: *Off-Path* Taking Over Internet Resources

Tianxiang Dai, *Fraunhofer Institute for Secure Information Technology SIT*;
Philipp Jeitner, *Fraunhofer Institute for Secure Information Technology SIT*,
Technical University of Darmstadt; Haya Shulman, *Fraunhofer Institute for
Secure Information Technology SIT*; Michael Waidner, *Fraunhofer Institute for
Secure Information Technology SIT, Technical University of Darmstadt*

<https://www.usenix.org/conference/usenixsecurity21/presentation/dai>

**This paper is included in the Proceedings of the
30th USENIX Security Symposium.**

August 11-13, 2021

978-1-939133-24-3

**Open access to the Proceedings of the
30th USENIX Security Symposium
is sponsored by USENIX.**

The *Hijackers* Guide To The Galaxy: *Off-Path* Taking Over Internet Resources

Tianxiang Dai^{*}, Philipp Jeitner^{*†}, Haya Shulman^{*} and Michael Waidner^{*†}

^{*}Fraunhofer Institute for Secure Information Technology SIT

[†]Technical University of Darmstadt

Abstract

Internet resources form the basic fabric of the digital society. They provide the fundamental platform for digital services and assets, e.g., for critical infrastructures, financial services, government. Whoever controls that fabric effectively controls the digital society.

In this work we demonstrate that the current practices of Internet resources management, of IP addresses, domains, certificates and virtual platforms are insecure. Over long periods of time adversaries can maintain control over Internet resources which they do not own and perform stealthy manipulations, leading to devastating attacks. We show that network adversaries can take over and manipulate at least 68% of the assigned IPv4 address space as well as 31% of the top Alexa domains. We demonstrate such attacks by hijacking the accounts associated with the digital resources.

For hijacking the accounts we launch off-path DNS cache poisoning attacks, to redirect the password recovery link to the adversarial hosts. We then demonstrate that the adversaries can manipulate the resources associated with these accounts. We find all the tested providers vulnerable to our attacks.

We recommend mitigations for blocking the attacks that we present in this work. Nevertheless, the countermeasures cannot solve the fundamental problem - the management of the Internet resources should be revised to ensure that applying transactions cannot be done so easily and stealthily as is currently possible.

1 Introduction

Internet resources form the cornerstone of modern societies. The daily activities and services are increasingly digitalised, from critical infrastructures to medical services and child care. The society relies on the control over its Internet resources for availability and stability of digital services and assets. Due to their importance, Internet resources pose a lucrative target for adversaries.

Internet resources are at risk. In this work we explore the security of the Internet management systems of basic

digital assets: IP addresses management with Regional Internet Registries (RIRs) [RFC7020], domains with domain registrars, virtual machine resources with infrastructure as a service (IaaS) providers and certification with Certificate Authorities (CAs), see the list in Table 1. These providers manage the allocation, registration and operation of the Internet resources for their customers. We study how easy it is for network adversaries to take over the accounts of these resource providers and then exploit the resources associated with the compromised accounts.

We show that the current practices of Internet resources management are insecure. Adversaries can take control over digital assets of customers and maintain control over them for long periods of time without being detected. Although such attacks are appealing for strong nation state adversaries and security agencies, we demonstrate that even weak *off-path* network adversaries can, through a series of protocol manipulations, take over the accounts of customers and thereby control the Internet resources associated with them.

Adversaries can hijack accounts. The idea behind our attacks is the following: the adversary poisons the cache of the DNS resolver of a resource provider, injecting a malicious record mapping the Email server of the victim customer to an adversarial host. The adversary invokes password recovery procedure. The Email server of the provider sends the password reset link to the IP address of the adversary. Adversary resets the password and hijacks the victim account. We demonstrate how the adversary can perform manipulations over the resources associated with the hijacked accounts.

Manipulation of the digital resources. The SSO (Single Sign On) accounts of the RIRs pose the highest risk: hijacking an SSO account allows a weak adversary to take over ASes and IP blocks allocated to the victim. Furthermore, through the hijacked account the adversary can make manipulations not only in the control plane of the Internet infrastructure but also in the Internet Routing Registries (IRR) and in Internet Addressing Resource Registries. Such modifications in the IRR can among others also facilitate extremely effective BGP prefix hijacks. Specifically, IRR records are prerequisite for

BGP hijack attacks - without proper records in the IRR the attacker cannot convince benign upstream providers to accept and propagate the fraudulent BGP announcements in the input filters on the BGP sessions. Adversaries without the ability to modify the IRR, have to use less vigilant and generally poorly managed networks as upstream providers or have to utilise path manipulation attacks [35] - both restricting the success rate and the stealthiness of the attack. Our adversary can, by modifying the records in the IRR, cause well managed and reputed upstream providers to unwittingly propagate the malicious BGP announcements. Hence making BGP prefix hijacks more effective than the typical control plane BGP prefix hijacks while at the same time more difficult to identify. To maintain control over the victim Local Internet Registries (LIRs) resources over long periods of time the adversary implants itself in the system with elevated privileges.

We also show that hijacking an account under a CA allows an adversary to revoke certificates, renew a certificate or issue new certificates for domains registered under the hijacked account. Renewal of certificates allows to associate a new key-pair with the certificate. Nevertheless some CAs do not perform validation of certificate renewal requests issued from registered accounts.

By hijacking the accounts of domain registrars, the adversary can manipulate records in victim domains, e.g., to launch phishing attacks. Finally, hijacking accounts of IaaS providers enables the attackers to take over virtual machines and the resources that run on those virtual machines, including databases, applications and computations.

Disclosure and ethics. Our attacks were tested against providers and customers reliably, yet were ethically compliant. To avoid harming Internet users, we set up victim domains and registered victim accounts which were used by us for carrying out the attacks and for evaluating the vulnerabilities. This ensured that the providers would not use the spoofed records for any “real” purpose. In addition to evaluating the attacks with the “victim” accounts that we set up, we also evaluated our exploits of hijacked accounts against one large ISP under RIPE NCC and attacked the real domain of that ISP in coordination with that ISP. We are disclosing the results of our work to the providers.

Contributions. We provide the first demonstration of off-path attacks exploiting hijacked accounts under popular providers and show that adversaries can perform manipulations in the resources assigned to the accounts over long time periods without being detected.

Organisation. In Section 2 we review DNS cache poisoning and related work. In Section 3 we provide an overview of our study. In Section 4 we list methodologies for off-path DNS cache poisoning attacks. In Section 5 we evaluate the cache poisoning methodologies for taking over customers accounts in different providers in our dataset. Then, in Section 6, we demonstrate how the adversaries can manipulate digital resources assigned to the accounts they control. In

Section 7 we explain the fraction of the digital resources (IP address’ blocks and domains) that are at immediate risk due to being associated with vulnerable accounts. We recommend countermeasures in Section 8 and conclude in Section 9.

2 DNS Cache Poisoning Overview

DNS. Domain Name System (DNS), [RFC1035], performs lookup of services in the Internet. Recursive caching DNS resolvers receive DNS requests for services in different domains and send queries to the nameservers authoritative for those domains. The nameservers respond with the corresponding DNS records. The DNS records in responses are cached by the DNS resolvers and are provided to clients and servers which use that resolver. Subsequent requests for that domain are responded from the cache. For instance, to send an Email to `alice@example.info` the Email server of Bob will ask the DNS resolver for the IP address of the Email exchanger in domain `example.info`. The resolver asks the nameservers in domain `example.info` for an IP address and a hostname (A and MX records) of the Email exchanger and receives:

```
example.info IN MX mail.example.info
mail.example.info A 1.2.3.4
```

The resolver will send to the Email server of Bob the IP address of the Email exchanger of Alice and will also cache the records from the response for answering future queries for MX and A in `example.info` domain.

DNS Cache Poisoning. In a DNS cache poisoning attack the goal of the adversary is to redirect clients of some resolver to an IP address of the adversary for queries in a target victim domain. To do that, the adversary sends a DNS response from a spoofed source IP address, which belongs to the nameserver of the victim domain, with malicious DNS records mapping the victim domain to an IP address of the adversary. For instance, to intercept the Emails sent to Alice the adversary injects a DNS record mapping the Email exchanger of Alice to an adversarial host. If the resolver accepts and caches the malicious record, its cache becomes poisoned.

```
example.info IN MX mail.example.info
mail.example.info A 6.6.6.6
```

The added value of DNS cache poisoning attacks is that they have a local impact, affecting not the entire Internet but only the victim network and hence allow for extremely stealthy attacks, which can go undetected over long time periods. There is more and more evidence of DNS cache poisoning in the wild and the attacks are becoming increasingly sophisticated. In the recent cache poisoning attacks in the wild the adversaries attempt to intercept DNS packets by launching short-lived BGP (Border Gateway Protocol) prefix hijacks [34]. In such attacks, the adversary advertises a BGP announcement hijacking the prefix of a victim for a short time

only to hijack the target DNS packet and then releases the hijack [15]. This allows the attacker to poison the DNS cache of a victim resolver and then intercept all the communication between the victim resolver and the target domain. Recent research projects showed that the CAs (Certificate Authorities) and the bitcoin infrastructures were not resilient to prefix hijacks [6, 8, 9].

History of DNS Cache Poisoning. Launching cache poisoning in practice is however hard. We explain the evolution of cache poisoning attacks and the mitigations. In 1995 Vixie pointed out to the cache poisoning vulnerability and suggested to randomise the UDP source ports in DNS requests [45]. In 2002 Bernstein also warned that relying on randomising Transaction ID (TXID) alone is vulnerable [7]. Indeed, in 2007 [29] identified a vulnerability in Bind9 and in Windows DNS resolvers [30] allowing off-path attackers to reduce the entropy introduced by the TXID randomisation. In 2008 Kaminsky [26] presented a practical cache poisoning attack even against truly randomised TXID. Following Kaminsky attack DNS resolvers were patched against cache poisoning [RFC5452] by randomising the UDP source ports in queries. Nevertheless, shortly after different approaches were developed to bypass the source port and the TXID randomisation for launching off-path cache poisoning attacks. In 2012 [17] showed that off-path adversaries can use side-channels to infer the source ports in DNS requests. In 2015 [41] showed how to attack resolvers behind upstream forwarders. This work was subsequently extended by [47] with poisoning the forwarding devices. A followup work demonstrated such cache poisoning attacks also against stub resolvers [5]. [33] showed how to use ICMP errors to infer the UDP source ports selected by DNS resolvers. Recently [31] showed how to use side channels to predict the ports due to vulnerable PRNG in Linux kernel. In 2013 [18] provided the first feasibility result for launching cache poisoning by exploiting IPv4 fragmentation. IPv4 fragmentation based attacks were applied to shift time on NTP servers [11, 23, 32], these attacks are not practical anymore since the nameservers in NTP domains were patched to avoid fragmentation. The study in [11] used fragmentation based cache poisoning for bypassing domain validation with CAs. However, most CAs patched the vulnerabilities which [11] exploited to attack domain validation, e.g., Let'sEncrypt blocked fragmentation. Let'sEncrypt also deployed domain validation from multiple vantage points [9, 25], which makes the previous off-path attacks [8, 11] impractical.

In addition to other attacks in this work, we also show another way to attack the CAs, by taking over customers' accounts with the CAs and not by bypassing domain validation. As we show this allows even more effective attacks that were presented in [11]: (1) when controlling a compromised account the adversary can renew existing certificates to use a new key-pair. Since some CAs do not apply domain validation during certificates' renewal this attack allows to issue fraudulent certificates without the need to attack DV.

Furthermore, in our work we use a number of off-path DNS cache methodologies from [14] to take over accounts with providers.

Cache poisoning attacks could be prevented with DNSSEC [RFC6840] [46] which uses cryptographic signatures to authenticate the records in DNS responses. However, DNSSEC is not widely deployed. Less than 1% of the second level domains (e.g., 1M-top Alexa) domains are signed, and most resolvers do not validate DNSSEC signatures, e.g., [12] found only 12% in 2017. Our measurements show that the DNSSEC deployment in our datasets is not better: the resolvers of 19 out of 35 tested providers do not validate DNSSEC signatures (see Table 2) and less than 5% of the customers' domains are signed. Deploying DNSSEC was shown to be cumbersome and error-prone [13]. Even when widely deployed DNSSEC may not always provide security: a few research projects identified vulnerabilities and misconfigurations in DNSSEC deployments in popular registrars [12, 42]. However, even correctly deployed DNSSEC does not prevent recent cache poisoning attacks [24]. The idea behind these attacks is to encode injections into the DNS records in DNS responses. When the resolver parses the records, a misinterpretation occurs, such that when the record is stored a different domain name is used. Since DNSSEC validation is applied prior to the misinterpretation, the validation of DNSSEC succeeds, and the DNS cache poisoning occurs afterwards. Preventing these attacks requires validating or escaping records from DNS lookups.

Recent proposals for encryption of DNS traffic, such as DNS over HTTPS [21] and DNS over TLS [22], although vulnerable to traffic analysis [40, 43], may also enhance resilience to cache poisoning. These mechanisms are not yet in use by the nameservers in the domains that we tested. Nevertheless, even if they become adopted, they will not protect the entire resolution path, but only one link on which the transport is protected and hence will not completely prevent DNS cache poisoning attacks.

3 Attack Overview

In our study we explore the security of the services which provide access to and management of the key digital assets in the Internet: domains, IP prefixes and ASes, virtual machines and certificates. In Table 1 we list the resources, as well as the public service providers of these resources, that we studied in this work. Access and management of these digital resources is performed with the accounts that the providers offer to the customers via their web portals. In this section we provide an overview of our study from the perspective of the adversary for hijacking the accounts of the customers under different resource providers.

Find the target. Assume for example that the adversary wishes to hijack the DNS servers hosted on a victim prefix 205.251.192.0/18 – this was a real attack launched against

an LIR Amazon route53 in April 2018. First, the adversary needs to find an account to which these resources are assigned and through which these resources can be managed. Then, the adversary needs to find the username associated with that account. In Section 5.2 we show how to find the needed information: the owner, the public service provider, the Email which is associated with the account through which the digital resources can be managed. In the case of our example, the prefix is allocated by ARIN to an LIR with OrgId AMAZON-4, aka Amazon.com, Inc. and has 3 origin ASNs (Autonomous System Numbers) registered: AS16509, AS39111 and AS7224. We thereby learn that the responsible RIR for Amazon is ARIN and that Amazon has an LIR agreement with ARIN. We also find the Email address `ipmanagement@amazon.com` used by Amazon for managing its resources via the SSO account with ARIN.

Poison DNS of public service provider. The adversary uses one of the methodologies in Section 4 to launch off-path DNS cache poisoning attack against the DNS resolver of the service provider ARIN. During the attack the adversary injects a malicious DNS record mapping the Email server of domain `amazon.com` to the IP addresses controlled by the adversary (step ①, Figure 1). As a result, the Emails sent by ARIN to Amazon will be received by the adversary.

Hijack victim account. The adversary triggers password recovery for Email `ipmanagement@amazon.com`. This Email is associated with the SSO account at ARIN. In order to send the password recovery link, the Email server at ARIN needs the IP address of the Email server of Amazon. The resolver at ARIN already has a corresponding record in the cache, which it provides to the Email server. This IP address was injected by the adversary earlier in step ①. ARIN sends the Email with password recovery instructions to the adversary (step ②, Figure 1). The attacker resets the password and takes control over the account. We experimentally evaluate such attacks against the providers and their customers in our dataset in Section 5 for details.

Manipulate the resources. The adversary manipulates the resources assigned to the victim account, say of Amazon, and can sell the IP prefixes and ASes owned by Amazon (step ③, Figure 1). In Section 6 we describe the exploits we evaluated against the resources assigned to our victim accounts. We show that among others, the attacker can create additional accounts for itself with arbitrary privileges, and hence even if the real owner resets the password back, the attacker still maintains control over the resources. In some cases these manipulations generate notification Emails to the Email address associated with the resources. This Email address is however hijacked by the adversary, hence the adversary receives the notifications. As a result the attack will not be detected and can stay under the radar over a long period of time.

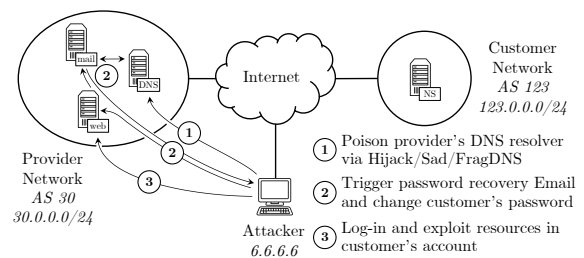


Figure 1: Attack overview

4 Off-Path DNS Cache Poisoning

The key contribution in our work is to show that once an adversary controls an account with a resource provider, it can in an easy and stealthy way manipulate the digital resources associated with that account. But, how easy is it to take over accounts? We show how to take over accounts by injecting a poisoned DNS record into the caches in DNS resolvers of providers. When the adversary triggers the password recovery procedure for the victim account, the reset email is sent to the adversarial host at the IP address in the injected DNS record.

How easy is it to launch off-path DNS cache poisoning? In this section we use methodologies from [14] to launch off-path DNS cache poisoning attacks: BGP prefix hijacks [8], side-channels [33], and IPv4 defragmentation cache poisoning [18]. We do not consider attack methodologies which are effective only against specific operating systems, say due to poor random number generators. We implement cache poisoning attacks using these methodologies and evaluate them against the providers and the customers in our dataset. We describe the experimental setup in Section 4.1. We explain our study methodology in Section 4.2. Then in Sections 4.3, 4.4 and 4.5 we present the DNS cache poisoning methodologies and the experimental evaluations against the targets in our dataset.

4.1 Setup

To test our attacks experimentally in the Internet we setup a victim AS. To purchase the victim AS we registered a secondary LIR account with RIPE NCC for our organisation (which has a primary account with RIPE NCC). We purchased a /22 prefix for our AS for 20,000USD. We connected our AS to DE-CIX internet exchange point in Frankfurt. This AS hosts the servers which we use for our evaluation of the attacks.

We set up an Unbound 1.6.7 DNS resolver on Linux 4.14.11, whose cache we poison with the records of the customer domains. We registered a victim domain and set up two nameservers in our domain and an Email server. We use our victim domain to register accounts with the services that we test in this work. We call this domain the victim customer domain. We also set up a border router which represents our attacker. The attacker's BGP router issues bogus BGP an-

nouncements that claim the prefix assigned to our victim AS. This allows us to evaluate the viability of attacks with BGP prefix hijacks against our domains hosted on our victim AS without affecting services and domains not under our control and without affecting the global BGP routing table in the Internet.

To evaluate cache poisoning attacks with side-channels we configure the nameservers in our domain to support rate-limiting and the DNS resolver to issue ICMP errors. To evaluate fragmentation based cache poisoning attacks we configure nameservers in our domain to reduce the MTU according to the value in ICMP fragmentation needed messages. The nameservers in our victim domain use a globally incremental IPID counter.

4.2 Study Methodology

Our experimental evaluation of the attacks is performed reliably yet without disrupting the functionality of the customers. To achieve this we evaluate the attacks in two steps: (1) We evaluate vulnerabilities to cache poisoning in providers. For this we set up victim domains and register victim accounts with the providers. We experimentally test the attack methodologies against providers by poisoning their DNS caches with malicious records mapping the Email server in our victim domain to the adversarial hosts that we control. We then hijack our victim accounts by triggering the password recovery procedures and changing their passwords. This enables us to validate vulnerabilities to cache poisoning yet without risking that the providers use poisoned records for genuine customers. The ability to take over the accounts of the real customers depends not only on vulnerabilities in providers' infrastructure but also on properties in customers' domains. (2) Hence, in this step we set up a victim DNS resolver and poison its cache with malicious records mapping the genuine customer domains to our adversarial hosts. The combination of both evaluations against the providers and against the customers enables us to estimate the extent of the vulnerable accounts that can be hijacked.

4.3 BGP Prefix Hijack

BGP (Border Gateway Protocol) allows ASes to compute the paths to any Internet destination. Since BGP is currently not protected, adversaries can send bogus BGP announcements to hijack victim prefixes, hence intercept the communication of victim ASes that accept the malicious BGP announcements. In our attacks we hijack the prefix of our AS: once in the evaluation against providers to intercept the responses from our nameservers sent to the DNS resolvers of the providers and then again during the evaluation of the customers, to intercept requests from our DNS resolver to the customers' domains. After our AS accepts the bogus BGP announcement, all the communication between the servers on our AS and the

servers of the targets in our dataset traverse our adversarial BGP router.

We launch short-lived hijacks. Such hijacks are common [37] and allow the attacker to stay below the radar [4, 16]. It is believed that short-lived traffic shifts are caused by the configuration errors (that are quickly caught and fixed) and since they do not have impact on network load or connectivity, they are largely ignored [10, 27, 28]. We evaluate our attacks using short-lived same prefix hijacks and sub-prefix of the victim prefix.

Our experimental evaluation reflects a common BGP hijacking attacker: the attacker controls a BGP router or an AS, and issues BGP announcements hijacking the same-prefix or a sub-prefix of a victim AS in the Internet.

4.3.1 Attack evaluation against providers

The adversary announces to our victim AS a prefix of the network of the provider where the target DNS resolver is located. The bogus BGP announcement is sent *only* on the interface that is connected to our AS and is not sent to other destinations in the Internet. As a result, the responses from the nameservers of our victim domain are sent to the adversarial host instead of the DNS resolver of the provider. The adversary initiates password recovery procedure for an account of our victim customer domain. This triggers a DNS request to our victim domain. The corresponding nameserver sends a response, which is instead redirected to the adversary's host. The adversary manipulates the response, and injects a DNS record that maps the Email server of our victim domain to the IP address of the adversary. The response is then sent to the provider and the BGP hijack is released. The DNS resolver caches the response and returns it to the Email server, which sends the password recovery link to the IP address of our adversary. The adversary resets the password and takes control over the account.

4.3.2 Attack evaluation against customers

The adversary announces to our victim AS prefixes of the networks that host the nameservers in the target customers' domain. The bogus BGP announcements are sent *only* on the interface that is connected to our AS and not to other destinations in the Internet. As a result, the DNS requests from the DNS resolver on our victim AS are sent to the adversarial host instead of the nameservers of the customer' domain. The attacker releases the hijacked prefix, and additionally crafts a spoofed DNS response to our DNS resolver mapping the IP address of the adversary to the Email server of the victim customer's domain. The records from the DNS response are cached by our resolver.

4.4 Side-channel Port Inference

SadDNS off-path attack [33] uses an ICMP side channel to guess the UDP source port used by the victim resolver in the query to the target nameserver. This reduces the entropy in a DNS request from 32 bit (DNS TXID & UDP port) to 16 bit. The adversary then uses brute-force to match the TXID by sending spoofed packets for each possible TXID value to the resolver.

4.4.1 Attack evaluation against providers

We verify the existence of the ICMP global-rate limit: we send a single UDP probe to the resolver to verify that it emits ICMP port unreachable messages. Then, we send a burst of 50 spoofed UDP packets to closed ports at the resolver and follow up with a single non-spoofed UDP packet and observe if an ICMP port unreachable message is received by our sender. If the ICMP global rate-limit is present no message will be received because the global rate-limit is already reached.

The adversary initiates password recovery procedure with a provider for an account of our victim customer domain. This triggers a DNS request to our victim domain. The adversary mutes the nameservers on our victim AS, to prevent the response from being sent to the resolver of the provider, then runs the procedure for inferring the source port in the DNS request. Once the source port is found, it sends 2^{16} spoofed responses for each possible TXID values with malicious DNS records in payload. The records map the nameservers of the victim domain to attacker controlled IP addresses. If the response is accepted by the resolver of the public service, it is cached and used by the service for sending an Email with the password or the reset link. The attacker now controls the account.

4.4.2 Attack evaluation against customers

We configure our DNS resolver to send ICMP errors on closed ports. We use our own implementation of the SadDNS port scanning application with binary search and attempt to poison the resolver with a malicious record pointing the domain of the customer to our adversarial host. Due to a high failure rate, evaluation of each tuple (resolver, domain) takes up to 30 minutes, hence evaluating SadDNS on all the domains in our dataset is not practical. We therefore perform the measurement on a dozen randomly selected customers in our dataset. Our implementation performs the complete attack from triggering the queries to muting the nameservers and scanning the ports (using the ICMP side-channel) and in the last step sending the spoofed DNS responses with malicious records.

The high failure rate of the SadDNS attack is due to the fact that most of the queries do not generate a useful attack window, since the resolver times out after less than a second. The attacker can further improve this via manual attack by analysing the back-off strategies of the target resolver. The

timeout of the resolver is implementation dependent, e.g., the timeout value of Unbound is a dynamically computed value based on RTT to the nameserver, while Bind uses 0.8 seconds. The DNS software increases the timeout value after each retransmission.

4.5 Injection into IP-Defragmentation Cache

The off-path adversary uses a spoofed IPv4 fragment to manipulate the fragmented response from the nameserver, [18]. The idea is to send a spoofed fragment which is reassembled with the first genuine fragment of the nameserver. The adversary replaces the second fragment of the nameserver with its malicious fragment, hence overwriting some parts of payload of a DNS response with new (attacker's injected) content. As a result, the reassembled IP packet contains the legitimate DNS records sent by the genuine nameserver with the malicious records from the fragment sent by the adversary. Since the challenge values (port, TXID) are in the first fragment of the response from the nameserver, they remain intact, and hence correct.

4.5.1 Attack evaluation against providers

We evaluate FragDNS attack against the resolvers of the providers with our victim domain. For our nameservers we use a custom application that we developed, which always emits fragmented responses padded to a certain size to reach the tested fragment size limit. The nameservers are configured to send CNAME records in the first fragmented response. As a result, when the resolver of the provider receives a fragmented response and reassembles it, the DNS software will issue a subsequent query for the CNAME-alias. This allows us to verify that the spoofed fragment arrived at the resolver and was reassembled correctly and cached, which is an indicator that the cache poisoning via fragmentation attack succeeded. Throughout the attack we use our adversarial host to trigger password recovery procedures and to inject malicious DNS records into the caches of the providers, mapping the Email servers in our domains to the IP addresses allocated to our adversary.

The adversary sends two spoofed fragments (for each nameserver's IP address) to the resolver of the public service. The fragments are identical except for the source IP addresses: one is sent with a source IP address of one nameserver and the other is with the source IP address of the other nameserver. The fragments are constructed so that they match the first fragment in the response that will have been sent by our nameserver. In the payload the fragments contain malicious DNS records mapping the Email server to the IP address of the adversary. The adversary initiates password recovery for our victim account. This triggers a DNS request to one of the nameservers in our victim domain. We do not know in advance which nameserver that will be, and hence initially send

two spoofed fragments (for each nameserver). The response from the nameserver is sent in two fragments. Once the first fragment reaches the IP layer at the resolver of the provider it is reassembled with one of the second fragments of the adversary (it is already waiting in IP defragmentation cache). The reassembled packet is checked for UDP checksum and if valid, passed on to the DNS software on the application layer. If the records from the DNS packet are cached by the resolver of the public service, the password recovery link will be sent to the host controlled by the attacker.

When to send the spoofed ‘second’ fragment? The standard [RFC791] recommends caching the IP fragments in IP defragmentation cache for 15 seconds. If there is no matching fragment after 15 seconds, the fragment is removed from IP defragmentation cache. The actual caching time exceeds 15 seconds in most implementations. For instance in Linux `/proc/sys/net/ipv4/ipfrag_time` is set to 30 seconds. For attack to succeed the total time between the moment that the fragment enters the IP defragmentation cache at the provider’s resolver and the moment at which the first fragment from the genuine nameserver arrives should not exceed 15 seconds (to ensure that the attack is effective not only against resolvers running on Linux but also against standard compliant operating systems). We show that in practice 15 seconds suffice to launch the attack. We measure the latency from the moment that we trigger the password recovery procedure via the web interface of the provider and the moment that a DNS request from the resolver of the provider arrives at our nameservers. The measurements for different providers are plotted in Figure 2. As can be seen, except for two providers, all the latencies are below 30 seconds. The results for the attack window across all the providers, plotted in Figure 2 show that the latencies are stable, and are within the interval which provides for successful attacks. For instance, for AFRINIC RIR the attacker learns that after issuing the password recovery procedure the DNS query will be sent to the victim nameserver at a predictable time interval (between 0.1 and 0.2 seconds).

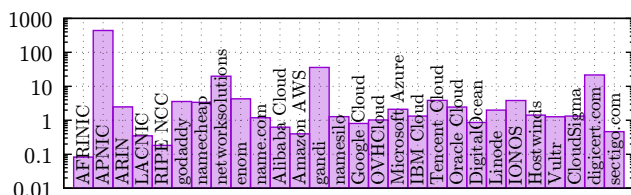


Figure 2: Avg. latency (in seconds) between registration and resolver query, excluding outliers outside $\pm 1\sigma$.

4.5.2 Attack evaluation against customers

Our evaluation is performed with the domain of a victim customer against our DNS resolver. The DNS resolver is configured to allow fragmentation. We look-up the nameservers in the domain of the customer and check if we can force

them to fragment responses: (1) for each nameserver, our DNS resolver sends requests to the nameserver and receives responses. (2) From the adversarial host we send to these nameservers *ICMP fragmentation needed* errors indicating Packet Too Big (PTB) for the source IP address of our DNS resolver. (3) We send DNS requests from our resolver and check if the responses arrive fragmented according to the MTU indicated in the ICMP errors.

We then run FragDNS attack against the nameservers that fragment DNS responses following our ICMP PTB errors: (4) The adversarial host crafts spoofed second fragments, one for each nameserver in customer’s domain. Since the adversary does not know to which nameserver the resolver will send a DNS request (the nameserver selection depends on DNS resolver software) it will send spoofed second fragments for each of the nameserver in that domain. Each fragment contains an identical payload: a malicious DNS record that maps the Email server of the customer domain to the IP address of our adversary. Each fragment has a different spoofed source IP address corresponding to each of the nameserver in the target domain. The adversary sends all these fragments to our DNS resolver. (5) The adversary causes our DNS resolver to issue a DNS request for a MX record in victim customer’s domain. The nameserver which received the request responds with a fragmented DNS packet. The first fragment is reassembled with the matching second fragment that is waiting in the IP defragmentation cache. (6) The adversary receives a DNS response from our resolver. If the Email server in the response is mapped to the IP address of our adversary, then the attack succeeded.

5 Hijacking Accounts

In this section we evaluate DNS poisoning attacks against the providers and the customers using the methodologies in Section 4.

After collecting the target providers and their customers in Section 5.1, we analyse the password recovery mechanism at each provider in Section 5.2. Then we collect the DNS resolvers at those providers in Section 5.3.1. We evaluate off-path cache poisoning attacks against the DNS resolvers of providers in Section 5.3.2. Finally we measure the percentage of vulnerable customers of those providers in Section 5.4.

5.1 Datasets

In our measurements and attacks’ evaluations we use two datasets: of providers and of their customers.

Providers. The providers that we study are listed in Table 1. For each class of resource providers (RIRs, Registrars, IaaS providers, CAs) we select a set of most popular examples. Our methodology for selecting the providers is: (1) all the five RIRs, (2) we scan the `whois` data of 100K-top Alexa domains and select the top 15 registrars according to the number of

domains each registrar is managing, (3) to select the IaaS providers, we use market share data and supplement it with additional selected providers¹, (5) we select the top 5 CAs which cover 97% of the market share², all other CAs have less than 1% market share.

For registrars and IaaS providers these datasets include providers which we could not test, because they do not allow creation of user accounts. For example, publicdomainregistry does not offer accounts to end-users directly, but only manages domain registration for webhosters. Providers where we could not register accounts are: Tucows.com, publicdomainregistry, cseglobal, markmonitor, RackSpace Cloud, CenturyLink Cloud and Joyent Triton.

We obtain a list of 32 resource providers which use 1,006 resolvers for sending Email (back-end IP addresses) on 44 ASes associated with 130 prefixes. Some resource providers use only a small amount of Email servers and resolvers on their own networks, while other providers use large pools of Email servers and resolvers provided by third-party Email services like Mailchimp and Sendgrid. We list this technical information in Table 2.

Customers. We extract account information for customers of RIRs and domain registrars from `whois` databases. We parse the Email addresses in `whois` records to extract the domains of the customers and query the nameservers responsible for those domains.

Because of data protection settings, not all `whois` records contain Email addresses, or only contain masked Email addresses which point to a registrar's Email proxy. We were able to find Email addresses for 74.62% of the ASes from RIR `whois` databases and for 10.60% of the domains owners in 100K-top Alexa list from domain registrar `whois` databases (see Table 3). We collected 94,997 user accounts hosted in 59,322 domains and 69,935 nameservers.

We were not able to retrieve user account information for IaaS accounts and CAs as this is not possible ethically in an automated way. An adversary can obtain this information, e.g., by enumerating usernames as described in Section 5.2.

Our dataset of domain registrars is also representative for other types of resources hosted under that domain. Organisations which own domains also own cloud resources at IaaS providers and certificates at CAs and use the same domain for their Email addresses and therefore are vulnerable to the same attack at those providers.

¹<https://www.srgresearch.com/articles/quarterly-cloud-spending-blows-past-30b-incremental-growth-continues-rise>, <https://stackify.com/top-iaas-providers/>, <https://www.g2.com/categories/infrastructure-as-a-service-iaas>

²https://w3techs.com/technologies/history_overview/ssl_certificate: this market share data lists most of Let'sEncrypt certificates as issued by IdenTrust as Let'sEncrypt certificates are cross signed by IdenTrust. We do not test Let'sEncrypt itself because it does not offer traditional user accounts and therefore does not support password recovery.

5.2 Collecting Accounts' Information

The first step in our attack is to trigger the password recovery procedure at the provider. This step requires collecting information of the target customer whose account the attacker attempts to hijack, such as the Email account required to log into the target account, a username or a handle. We study for each service provider which information is needed for password recovery and how to collect that information for our targets; the data is summarised in Table 1. We found that the customers' Email addresses can often be retrieved from the public `whois` records. We were able to extract the Email addresses associated with the accounts at the providers for 41% of the customers in our study. For instance, the Emails for the SSO accounts of 74.62% of the LIRs (i.e., the customers of RIRs) can be retrieved via `whois`.

For victim customers whose details cannot be publicly accessible via `whois` we find the required information with manual research and dictionary attack. To carry out the dictionary attack we used the observations we derived from our data collection from public sources: the data we collected through `whois` shows that more than 24% of the Email addresses use one of ten well-known username parts, like `domains@email.info`, `hostmaster@email.info`, etc., which enables an informed attacker to find the Email addresses in less than ten attempts when these details are not publicly available through `whois`. We apply dictionary attack to also recover other details: for example, our study shows that about 1 in 10 LIRs (customers of RIRs) use usernames that are identical to the Email address that is registered in the `whois` records; e.g., `username operator` is associated with Email address `operator@email.info`.

5.3 Attacking Providers

The adversary needs to poison the DNS cache of the provider, by injecting a record into the resolver's cache that maps the domain of the provider to the adversarial IP addresses. We therefore collect the IP addresses of the DNS resolvers of the providers.

5.3.1 Identify the target DNS resolvers

In order to poison the DNS cache of the provider the adversary needs to find the IP addresses of the DNS resolvers which are used for looking up the Email servers of the customers during requests for password recovery.

We register accounts with the providers via the web portal of each provider. For our evaluation we register 20 accounts with each provider, each account is associated with a unique domain that we registered for that purpose. We use these registered accounts to learn about the infrastructure of the provider. We trigger the password recovery procedure for our registered accounts. To stay under the radar we limit the amount of password recovery requests to ten for each account.

Type	Provider	Details needed for PW recovery	Public-known	Capcha	Fragment	SadDNS	BGP hijack
RIRs	AFRINIC	NIC-handle	✓	✗	✓	✗	✓
	APNIC	Email	✓	✗	✓	✗	✗
	ARIN	Email, Username	✗	✓	✓	-	✗
	LACNIC	Username	✓	✗	✓	✗	✓
	RIPE NCC	Email	✓	✓	✓	✗	✓
Domain registrars	godaddy	Email, Domain name	✓	✗	✓	-	✓
	namecheap	Email	✓	✓	✓	✗	✓
	networksolutions	Email	✓	✗	✓	✗	✓
	enom.com	Login ID, Sec. question	✗	✓	✓	✗	✓
	name.com	Username ¹	✓	✗	✓	-	✓
	Alibaba Cloud	Username, 2-FA	✗	✓	✓	✗	✓
	Amazon AWS	Email	✓	✓	✓	✗	✓
	gandi.net	Email	✗	✗	✓	✗	✓
	namesilo.com	Email, Sec. question	✗	✓	✓	✗	✓
	Google Cloud	Last password, 2-FA	✗	✓	✓	✓	✓
ovh.com	Email	✓	✗	✓	✗	✓	
Cloud management (IaaS)	Amazon AWS	Email	✓	✓	✗	✗	✓
	Microsoft Azure	Email	✗	✗	✗	✗	✓
	Alibaba Cloud	Username, 2-FA	✗	✓	✗	✗	✓
	Google Cloud	Last password, 2-FA	✗	✓	✗	✗	✓
	IBM Cloud	Email ('id')	✗	✓	✓	✓	✓
	Tencent Cloud	Email	✓	✓	✗	✗	✓
	Oracle Cloud	Email	✗	✗	✗	✗	✓
	DigitalOcean	Email	✗	✓	✗	✗	✓
	Linode	Username ¹	✗	✗	✓	✓	✓
	IONOS	Email, id or domain	✗	✓	✓	✓	✓
	Hostwinds	Email	✗	✗	✗	✗	✓
	OVHcloud	Email	✓	✓	✗	✓	✓
	Vultr	Email	✓	✓	✓	✓	✓
CloudSigma	Email	✗	✓	-	-	✓	
CAs	IdenTrust	Account number	✗	✗	✓	-	✓
	DigiCert	Username ¹	✗	✗	✓	✗	✓
	Sectigo	Email	✗	✗	✓	✗	✓
	GoDaddy	Username ¹ , customer No. ¹	✗	✗	✓	-	✓
	GlobalSign	Username	✗	✗	✓	-	✓

-: No response. ¹: Can be retrieved using domain name/Email.

Table 1: Password recovery at each provider.

The Email server of the provider requests the DNS resolver to look up the MX and A records for our Email exchanger - this is required in order to send the password, or the link to reset the password. We monitor the requests arriving at the nameservers of our domains and collect the IP addresses which sent DNS requests for records in domains under which we registered our accounts. These IP addresses belong to the DNS resolvers used by the providers. We repeat this for each provider on our list in Table 2.

For every provider, we list in Table 2 the service providers of the Email servers and the DNS resolvers (by mapping the observed IP addresses to ASNs). Additionally, we also performed measurements if the resolvers of the providers in our dataset support DNSSEC and the default EDNS size in DNS requests.

5.3.2 Poison providers' DNS caches

To understand the vulnerabilities to cache poisoning across the providers, we evaluate the DNS cache poisoning methodologies against the DNS resolvers of the providers in our dataset. Our evaluations are done as described in Section 4 using the victim domains that we set up and the accounts that we registered. During the evaluations the adversary triggers password recovery procedure and applies the DNS cache poisoning methodologies (one during each test) to inject into the DNS cache of the provider malicious records mapping the Email servers of our victim domains to the hosts controlled by the adversary. In this section we report on the results of our evaluations and the extent of the vulnerabilities among the providers.

HijackDNS. To infer the scope of providers vulnerable to the attack in Section 4.3 we perform Internet measurements checking for vulnerabilities that allow sub-prefix hijacks. Since many networks filter BGP advertisements with prefixes more specific than /24, we consider an IP address vulnerable if it lies inside a network block whose advertised size is larger than /24. We therefore map all resolvers' IP addresses to network blocks. Then, to obtain insights about the sizes of the announced BGP prefixes for providers' network blocks with resolvers we use the BGPStream of CAIDA [2] and retrieve the BGP updates and the routing data from the global BGP routing table from RIPE RIS [38] and the RouteViews collectors [44]. We analyse the BGP announcements seen in public collectors for identifying networks vulnerable to sub-prefix hijacks by studying the advertised prefix sizes. The dataset used for the analysis of the vulnerable sub-prefixes was collected by us in January 2021. Our analysis in Table 2 shows that the networks of 29 providers are vulnerable to sub-prefix hijacks.

To understand the viability of same-prefix hijack attacks we perform experimental simulations using the target providers and customers in our dataset. For creating the topological map of the AS-relationship dataset of the customer domains and the providers in our dataset we use CAIDA [3]. We simulate the attacks using a simulator developed in [19]. We evaluate HijackDNS attack for each provider with respect to customer domains of the corresponding provider and an AS level adversary on an Internet topology. In our simulations we consider attacks from 1000 randomly selected ASes against the domains of the customers and providers. The adversary can succeed at the attack against 80% of the Alexa customer domains with 60% success probability. One of the reasons for the high success probability is the concentration of the nameservers in few ASes: 10% of the ASes host 80% of the nameservers in Alexa domains and 1% of ASes host 80% of the domains. The customers of the LIRs are slightly more resilient since they mostly use at least two nameservers on different prefixes. This means that to succeed the attacker would need to hijack both. Furthermore, the distribution of

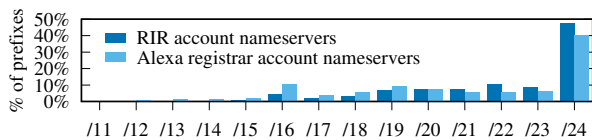


Figure 3: IP prefix distribution of customer accounts' nameservers

the nameservers across ASes is more uniform in contrast to Alexa domains.

SadDNS & FragDNS. To test vulnerabilities in the providers to SadDNS and FragDNS we perform the evaluations in Sections 4.4 and 4.5. Out of 31 tested providers, 28 (90%) are vulnerable to FragDNS attack and four of the providers are vulnerable to SadDNS attack. Vulnerabilities for each provider are listed in Table 1.

5.4 Measurements of Vulnerable Customers

The success of the attack against a specific victim customer depends not only on the vulnerabilities in the DNS resolver of the provider but also on the properties in the domain of the customer. For instance, say a DNS resolver of some provider is vulnerable to FragDNS attack but the nameservers of the customer's domain do not fragment UDP packets and packets that are too large are transmitted over TCP. In that case, the FragDNS attack is not effective against that customer. To understand the extent of the vulnerabilities in customers we evaluate the attack methodologies in Section 4 against the DNS resolvers that we own and control using the responses from the domains of the customers in our dataset. Using results from this evaluation we can reliably determine if the attack methodology is effective against a customer or not.

The results of our experimental evaluations of attacks in Section 4 and measurements of the customers' domains and their nameservers are summarised in Table 3.

HijackDNS. We analyse the prefixes of the customers similarly to Section 5.3.2. The results are plotted in Figure 3. Our findings are that more than 60% of the domains have all their nameservers on prefixes less than /24. Furthermore, above 20% of the domains host all the nameservers of each domain on a single prefix, as a result, by hijacking one prefix the adversary immediately hijacks the entire domain. Out of these, 17% host all the nameservers on a prefix that is less than /24. 10% of the domains have a single nameserver in the domain. To conclude: more than 40% of the domains are vulnerable to HijackDNS attack via sub-prefix hijack.

SadDNS. Based on the implementation in Section 4.4 we develop an automated simulation of the SadDNS attack, and run it on our dataset of customer domains to compute the success probability of SadDNS attack against our victim DNS resolver. When running the attack for domains that have the required properties (e.g., support rate limiting), poisoning succeeds after an average of 471s (min 39s, max 779s) which is

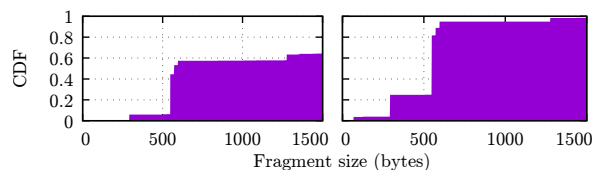


Figure 4: Cumulative distribution of lowest fragment size of nameservers (left) and domains (right) after sending ICMP PTB.

comparable to the original SadDNS results (avg 504s, min 13s, max 1404s, [33]). Our test implementation triggers 497 queries on average for each domain, which is strongly correlated with the attack duration due to the fact that we do not trigger more than two queries per second; the resolvers return SRVFAIL when receiving more than two queries per attack iteration. By inverting this number we get a hitrate of 0.2%.

Our results of an automated evaluation of SadDNS show that 8,469 accounts from the RIRs dataset and 11% of the accounts from the Alexa dataset could be hijacked via the SadDNS method.

FragDNS. We measured the victim customers' nameservers for support of ICMP errors and fragmentation. We send a DNS request to the nameserver for ANY type DNS record. After a DNS response, we follow with an ICMP PTB error, that indicates different MTU values, and repeat the request. We check if the response arrived in fragments according to the MTU value indicated in the ICMP error message. We performed evaluations with the following MTU values of 1280, 576, 296 and 68 bytes.

Figure 4 (Left) shows cumulative distribution of fragmented packet size we received after sending ICMP PTB. Right side shows the percentage of the domains where at least one nameserver supported a MTU smaller than the plotted size. As can be seen, for more than 90% of the domains with PMTUD-configured nameservers, at least one nameserver is willing to reduce the fragment size (in response to ICMP PTB) to almost 548 bytes, for roughly 35% domains to 296 bytes and for 10% to 68 bytes. This essentially allows to inflict fragmentation to any size needed. Our evaluations in Section 4.5 indicate that 11964 RIR customer accounts (13.6%) and 2352 Alexa domain holder accounts (22.2%) are vulnerable to the FragDNS attack.

We analyse the attacker's success probability of crafting the spoofed second fragment with the correct UDP checksum and the correct IPID value. To compute the success rate to hit the correct UDP checksum we performed the following evaluation. For each customer domain in our dataset, we query the nameservers of each domain multiple times sending the same DNS request (with the domain of the customer's email and type MX), and check if the DNS responses from the nameservers contain the same DNS records and the same order of DNS records, during each iteration. The computation of the UDP checksum for each domain is described in pseudocode in Algorithm 1. Our evaluation shows that for 1748 domains

	Rank	Provider	Mail service provided by	Resolver		Seen Via		Accept Fragment					BGP	DNSSEC		EDNS size	
				#	service provided by	Signup	PW Rec.	1500	1280	576	292	68	prefix-size	do	validate		
RIRs	-	AFRINIC	Self	3	Self	✓	✓	✓	✓	✓	✓	✓	/23	✓	✓	4096	
	-	APNIC	Self	1	Self	✓	✓	✓	✓	✓	✓	✓	/24	✓	✓	4096	
	-	ARIN	Self	4	Self	✓	✓	✓	✓	✓	✓	✓	/24	✓	✓	4096	
	-	LACNIC	Self	1	Self	✓	✓	✓	✓	✓	✓	✓	/22	✓	✓	1280	
	-	RIPE NCC	Self	3	Self	✓	✓	✓	✓	✓	✓	✓	/12-23	✓	✓	4096	
Registrars	1	godaddy	Self	3	Self	✓	✓	✓	✓	✓	✓	✓	/19-21	✓	✗	4096	
	2	namecheap	SendGrid	64	SendGrid	✓	✓	✗	✗	✓	✓	✓	/12-23	✓	✗	1232	
	3	networksolutions	Self	1	Self	(3)	✓	-	-	-	✓	✓	/20	✗	(1)	512 (2)	
	6	enom	Self	17	Self, Google	✓	✓	✓	✓	✓	✓	✓	/20	✓	✗	4096	
	9	name.com	Self (AWS)	8	Self (AWS)	✓	✓	✓	✓	✓	✓	✓	/12	✓	✗	4096	
	10	Alibaba cloud	Self	11	Self	✓	✓	✓	✓	✓	✓	✓	/16-21	✓	✗	4096	
	11	AWS	Self	46	Self	✓	✓	✓	✓	✓	✓	✓	/12-21	✓	✗	4096	
	12	gandi	Self	3	Self	✓	✓	✓	✓	✓	✓	✓	/23	✓	✓	4096	
	13	namesilo	Self	2	Self	✓	(1)	-	-	-	✓	✓	/16-19	✗	✗	512 (2)	
	14	Google Cloud	Self	120	Self	✓	(1)	-	✓	✓	✗	✗	/16-22	✗	✗	1232	
	15	OVHcloud	Self	4	Self	✓	✓	✓	✓	✓	✓	✓	/18-24	✓	✓	4096	
	IaaS Providers	1	Amazon AWS	Self	46	Self	✓	✓	✓	✓	✓	✓	✓	/12-21	✓	✗	4096
		2	Microsoft Azure	outlook.com	373	outlook.com	✓	✓	-	-	-	✗	✗	/13-19	✗	✗	512 (2)
		3	Alibaba Cloud	Self	11	Self	✓	✓	✓	✓	✓	✓	✓	/16-21	✓	✗	4096
		4	Google Cloud	Self	120	Self	✓	(1)	-	✓	✓	✗	✗	/16-22	✗	✗	1232
5		IBM Cloud	SendGrid	51	SendGrid	✓	(1)	✗	✗	✓	✓	✓	/12-23	✓	✗	1232	
(7)		Tencent Cloud	Self	13	Self	✓	✓	✓	✓	✓	✓	✓	/12-19	✓	✗	4096	
(8)		Oracle Cloud	Self	9	Self	✓	(1)	✗	✗	✗	✗	✗	/17-23	✓	✓	1372	
-		DigitalOcean	Mailchimp	8	Mailchimp	✓	✓	✓	✓	✓	✓	✓	/17-22	✓	✗	4096	
-		Linode	Self	2	Self	✓	✓	✓	✓	✓	✓	✓	/17	✓	✓	4096	
-		IONOS	Self	2	Self	✓	(1)	-	-	-	✓	✓	/16	✓	✓	1220	
-		Hostwinds	Postmark	15	OpenDNS	(3)	✓	✗	✗	✗	✗	✗	/19-21	✓	✓	1410	
-		OVHcloud	Self	4	Self	✓	✓	✓	✓	✓	✓	✓	/18-24	✓	✓	4096	
-		Vultr	Self	8	Self	✓	✓	✓	✓	✓	✓	✓	/18-20	✓	✓	4096	
-		CloudSigma	Mailchimp	6	Mailchimp	✓	✓	✓	✓	✓	✓	✓	/17-22	✓	✗	4096	
CAS		1	IdenTrust	Trend Micro	114	Trend Micro	✓	(1)	✓	✓	✓	✓	✓	/15	✓	(1)	4096
	2	digicert.com	Self	137	Self	✓	✓	✓	✓	✓	✓	✓	/16-22	✓	(1)	4096	
	3	sectigo.com	SendGrid	10	SendGrid	(3)	✓	✗	✗	✓	✓	✓	/12-23	✓	✗	1232	
	4	godaddy	Self	3	Self	✓	✓	✓	✓	✓	✓	✓	/19-21	✓	✗	4096	
	5	globalsign.com	(1)	35	Google	✓	(1)	✓	✓	✓	✗	✗	/20	✓	✓	4096	

Table 2: Measurement study of provider’s DNS resolvers and Email servers. (1): Could not test. (2): No EDNS. (3): No Email after sign-up. -: Does not apply.

Account Provider	# Resources			Vulnerable to				
	Total	found e-mail	# Accounts	BGP sub same	Sad-DNS	FragDNS any global		
Scanned resources				Vulnerable Accounts				
RIRs	92,857	69,287 75%	87,547	47,840 56%	n/a n/a	8,469 11%	14,136 17%	1,193 1.5%
Registrars	100,000	10,597 11%	7,450	3,308 45%	n/a n/a	666 10%	1,560 21%	85 1.2%
Both	192,857	79,884 41%	94,997	51,148 56%	n/a 80%	9,135 11%	15,696 17%	1,278 1.4%
Vulnerable resources								
		IP Addresses		81%	n/a	30%	51%	21%
		AS Numbers		60%	n/a	12%	20%	3%
		Domains		47%	n/a	10%	27%	1%
Attack success probability								
		Success probability		100%	60%	0.2%	0.1%	20%

Table 3: Customer-side vulnerability data

(62%), nameservers always return the same DNS response (with the same records and sorted in the same order); see

Algorithm 1: Predictability of records in responses.

```

for each (domain, nameserver) do
  initialise set of different DNS responses as empty;
  for batch = 1, 2, ..., 25 do
    for iteration = 1, 2, 3, 4 do
      send the same DNS request;
      if new response arrived then
        add the new response to the response set;
      end
    end
    if no new responses in last batch then
      break;
    end
  end
end
record number of different DNS responses;
end

```

Figure 5. For our measurement of the IPID allocation methods supported by the nameservers of the customers we use the following methodology. We issue queries from two hosts (with different IP addresses). Data per nameserver is listed in

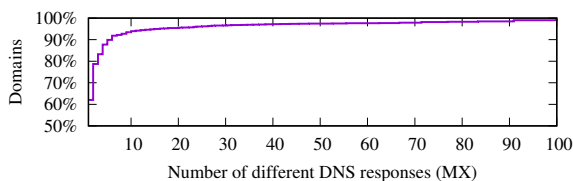


Figure 5: CDF of number of observed DNS MX responses per customer email address domain (each nameserver was queried 100 times).

Table 4. Our measurements show that 290 vulnerable nameservers (4.88%) use a globally incremental IPID assignment. The computation of the IPID allocation for each domain are described in pseudocode in Algorithm 2.

Algorithm 2: IPID allocation in nameservers.

```

for each (domain, nameserver) do
  for batch = 1, 2, 3, 4 do
    send DNS request from Prober1;
    record IPID in DNS response as IPID2i-1;
    send DNS request from Prober2;
    record IPID in DNS response as IPID2i;
  end
  if IPIDi, i = 1, 2, ..., 8 is incrementing then
    globally incrementing;
  end
  if IPIDi, i = 1, 3, 5, 7 or IPIDi, i = 2, 4, 6, 8 is incrementing then
    per-dest incrementing;
  end
  if IPIDi == 0, i = 1, 2, ..., 8 then
    zero;
  else
    random and other;
  end
end

```

	Per-Dest	Global	Zero	Random and other	N/A	Total
All	64.58% 45308	8.31% 5829	4.89% 3434	11.92% 8364	10.30% 7223	100% 70158
Frag	53.96% 3206	4.88% 290	13.75% 817	23.67% 1406	3.74% 222	100% 5941

Table 4: IPID allocation of all nameservers and of fragmenting nameservers.

We automate the attack in Section 4.5 and execute the entire FragDNS attack against all the vulnerable customer domains, by injecting malicious records mapping Email servers of customers to an IP address of our adversarial host. Our evaluation combines the data we collected on DNS records in responses (randomisation of the DNS records or of their order in responses) and the IPID allocation of the nameservers. We also used Algorithm 2 to estimate the IPID increment rate, by recording the timestamp of each response and calculating the average increment rate of IPID value. We then extrapolate the value of IPID and calculate the probability of our adversary to correctly place at least one out of 64 fragments³ with the matching IPID in the resolver’s defragmentation cache. We use different values for IPID increment rate and delay

³64 fragments is the minimal size of the IP-defragmentation cache.

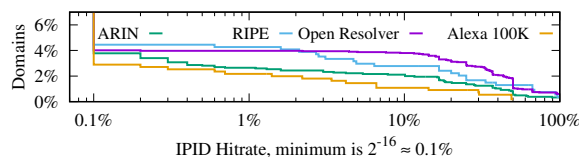


Figure 6: Reverse CDF to correctly guess the IPID for all customers’ domains.

between the query, which probes the IPID value, and the IPID value that was de-facto assigned to the DNS response by the nameserver. Results are plotted in Figure 6. For example, the IPID prediction success rate is over 10% for roughly 3% of RIPE, 2% of ARIN and 1% of 100K-top Alexa customers. Success rates for ARIN and 100K-top Alexa customers are lower mostly because of the higher latencies of those, see Figure 2. For nameservers which do not use globally incremental IPID, we assume a hitrate of $64/2^{16}$ which is achieved by just randomly guessing the IPID.

The probability to compute the correct checksum is capped at a minimum of $1/2^{16}$ in case of nameservers which generate responses with different records or with random ordering of records. Finally the probabilities to correctly compute both, the IPID value and the order of records to get the correct UDP checksum, are multiplied resulting in the combined hitrate. Our automated attack against all the customers shows that around 2% of the domains (5 for RIPE, 17 for ARIN) have a success probability higher than 10%. Furthermore, for about 20% of the domains, success probability is over 0.1% which is a consequence of non-predictable IPID allocation and the stable DNS records in responses generated by these domains. When the DNS response can be predicted, even with a random IPID allocation method, an attacker has a hitrate of about $64/2^{16} \approx 0.1\%$. At this hitrate, when the attacker performs the attack multiple times, the probability to conduct the attack successfully at least once is 50% at around 700 repetitions.

Our automated evaluation provides a lower bound for successful attacks against a randomly chosen domain – this is a worst case analysis since it also considers domains which are much more difficult to attack, e.g., since they use servers with random IPID allocation, servers with high traffic rate, and servers which return different number and order of records in responses. Adjusting the attack parameters manually against a given victim customer domain results in a much higher attack rate. Furthermore, against many customer domains with low traffic volume, incremental IPID values and fixed number and order of DNS records, the attacker can reach above 90% success rate.

6 Manipulation of Digital Resources

In this section we demonstrate exploits that the adversary can perform when controlling an account of a (victim) customer. Most of the actions are similar across the providers, even providers of different infrastructure, such as RIRs and the

Additional Validation	Attack		RIRs Registrars IaaS CAs	Outcome / Attacker use
RIRs	Account transfer/delegation		✓ ✓ ✓ ✗	permanent control
No	Changing the account details		✓ ✓ ✓ ✓	permanent control
RIRs	Close the account permanently		✓ ✓ ✓ ✓	DoS
No	Disabling Email alerts		✓ ✓* ✗ ✓*	remain stealthy
RIRs	Resource transfer		✓ ✓ ✓ ✗	permanent control
			✓ ✓ ✗ ✗	sell resources
No	Resource return / deletion		✓ ✓ ✓ ✓	DoS
CAs	Purchase new resources		✓ ✓ ✓ ✓	financial Damage
			✓ ✓ ✓ ✓	anonymous usage
No	Control / Modify Resources	Whois DB	✓ ✓ ✗ ✗	facilitates hijacking
		VMs	✗ ✗ ✓ ✗	various
		NS records	✗ ✓ ✗ ✗	traffic hijacking
No	Create new ROAs/certificates		✓ ✗ ✗ ✓	facilitates hijacking
No	Create invalid ROAs		✓ ✗ ✗ ✗	DoS
No	Revoke certificates		✗ ✗ ✗ ✓	DoS

Table 5: Actions an attacker can carry out after hijacking a customer’s account. *The Email address where the alert is sent can be changed. Additional validation requires either that additional documents are sent, or in case of issuing a new TLS certificate, that a domain validation must be passed.

domain registrars. Hence, we in details explain our demonstration of the exploits by taking over a victim LIR account, and then briefly describe the exploits we evaluated by taking over our victim accounts with the other providers. For our demonstration we select RIPE NCC RIR, GoDaddy domain registrar, Microsoft Azure IaaS provider and DigiCert CA. In order to evaluate the exploits using an account of a network operator, we cooperate with a large customer under RIPE NCC. We cooperate with that LIR and use a real account that has an *operator/administrator* role⁴. For domain registrars, IaaS and CAs, we used our own accounts which were used to buy test resources to test the possibilities of the account. We summarise the exploits for different providers in Table 5.

For our evaluation of the exploits, we first carry out our attacks in Section 4 to take over the victim accounts, and then carry out the exploits. We do this in order to understand what notifications are sent to the genuine account owners during such attacks, what actions can be performed, and which are prevented.

6.1 Regional Internet Registries

We show that adversary, controlling an SSO account of a victim LIR, can manipulate all the Internet resources associated with that LIR, e.g., the IPv4 and IPv6 addresses, ASNs, reverse DNS names to IP addresses mappings. The amount of resources managed by LIRs can vary enormously. There are small LIRs that manage just own AS, one IPv6 prefix and one

⁴RIPE NCC Single Sign-On (SSO) accounts are general authentication mechanism for all web-based services provided by RIPE NCC that include customer portal and other harmless services - for example RIPE Meeting facilitation.

or a very few IPv4 prefixes of minimal allocation size. There are also large LIRs managing vast PA address pools (Provider Aggregatable Addresses) for multiple clients. For instance, RIPE NCC has cumulative allocation of 587 202 560 IPv4 addresses.

The adversary impersonating an administrator with the RIPE NCC SSO account holder can initiate different actions that lead to disruption or degradation of the services that are tied to the IP resources managed by the victim LIR. The adversary can even initiate transfer of IPv4 addresses that belong to the victim LIR to obtain direct financial benefit from that process. Our experimental evaluation with an SSO account under RIPE NCC RIR shows that the actions of the attacker do not trigger alerts and can be detected when the LIRs realises that its digital resources are gone. The access to RIPE NCC SSO account with *operator* or *administrator* roles for the victim’s LIR opens to a range of possible exploits. We explain selected exploits with an example victim LIR under RIPE NCC below (also summarised in Table 5); the attacks similarly apply to the other RIRs.

RPKI administration. Attacker creates/deletes/modifies Route Origin Authorizations (ROAs) in hosted RPKI system. This has two purposes: (1) to disrupt the propagation of the legitimate BGP updates of the resources managed by the victim LIR and (2) to facilitate BGP hijacking by authorising attacker’s ASN to originate any subset of IP prefixes that are managed by the victim LIR. Networks which have deployed RPKI and perform filtering of BGP announcements with ROV will not trigger any alerts when the attacker issues a BGP announcement for a sub-prefix with a valid (yet fraudulent) ROA. We consider creation of ROA with origin set to ASN0 ("always drop" as per [RFC7607]) for a specific prefix within the resource pool managed by the victim LIR to be a special case of the malicious ROA intended to disrupt routing and cause DoS for the services tied to the IP addresses in question. Our measurements found that currently the Route-Origin Validation (ROV) is far from being universally deployed, with only 2190 ASes filtering invalid BGP announcements. Nevertheless, this is an increase in contrast to measurements from a few years ago, which found 71 ASes to validate ROV [20,36]. Even with 2000 validating ASes, this type of attack is likely to cause only minor disruption in service availability and will remain unnoticed for extended period of time.

RIPE DB modifications. Attacker manipulates records in RIPE DB - the Internet addressing resource registry of the region and Internet Routing Registry (IRR) in one converged database. Modification of records in resource registry allows impersonation of the victim LIR’s representatives in order to transfer resources from the victim LIR to unsuspecting recipient.

IRR records are prerequisite for BGP hijacking attacks, because without proper records in IRR the attacker would not be able to persuade any well-managed upstream provider that is consistent with AS operation best practices to accept the

fraudulent BGP announcements in the input filters on the BGP sessions. Attackers without the ability to modify IRR have to use less vigilant and generally poorly managed networks as upstream providers or have to utilise path manipulation attacks - both restrict success rate and stealthiness of the attack.

Creating the IRR records contradictory to the state in BGP is a way to partially disrupt route propagation and thus traffic. It can also de-stabilise network and significantly complicate network operation for the legitimate administrators. Route servers in majority of Internet Exchanges and major networks use IRR data in automatically generated filters that are applied on incoming BGP announcements. As a result of the contradicting IRR records these networks will drop or de-prefer the announcements from the legitimate resource holder. Moreover, well-managed networks keep manually generated import filters on small-scale BGP sessions for both peering and downstream customers. When a new session is set up or when a new prefix is about to be propagated from the neighbouring AS that is subject to filtering, the administrators manually check the IRR and resource registry to verify that the announcement is legitimate. Failing to have the proper records in IRR and in resource registry leads to refused BGP peerings, excessively strict BGP filters and therefore to dropped routes and overall degradation of the Internet connectivity for the victim network.

Initiating IPv4 addressing resource transfer. Attacker sells the resources managed by the victim LIR. The potential gain from successfully completed attacks of this type is determined by the amount of the addresses managed by the LIR and the expected monetary value of IPv4 addressing resources. After the IPv4 regular pool depletion on 4 September 2012 each LIR is eligible for allocation of a single /22 (1024 IPv4 addresses) prefix as per current IPv4 Address Allocation and Assignment Policies for the RIPE NCC Service Region (ripe-720) [1]. We performed IPv4 /22 prefix transfer of an LIR under RIPE NCC which has not triggered alerts. This is not surprising since the IPv4 addresses' transfer is performed regularly by the RIRs, e.g., RIPE NCC and ARIN perform thousands of transfers per year, see statistics on IP transfer (PI and PA) we collected from the RIRs in Figure 7.

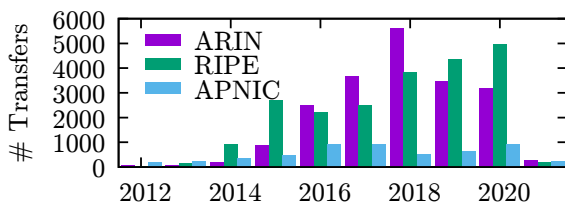


Figure 7: IPv4 resource transfers per year (PI & PA).

To sell the IPv4 addresses belonging to the victim LIR, we needed to perform the following:

(1) modify the relevant LIR contacts (Emails, phone numbers) in the RIR database and the IRR to receive the commu-

nication from the RIR intended to the genuine LIR, the buyer as well as the other parties relevant to the IP resource transfer and prevent the victim to learn about the resource transfer process. This is performed via the victim SSO account which the attacker took over.

(2) find the buyer for the resources and to impersonate the victim representatives to successfully close an IPv4 resource transfer contract; we used the compromised SSO account to sell and transfer the resources to an LIR that we set up for that purpose. In practice, the attacker can also collude with a malicious adversary, which will perform the transaction and afterwards will legally own the resources. The victim will need to prove to not have authorised the payment. In fact the resolution outcome of such a case is not clear since the RIRs have not faced this attack before.

(3) release the IPv4 prefix from the BGP routing table. This needs to be done so that the buyer believes that the resources are free and are being legitimately sold. This is done by sending an Email (via the Email address that the attacker modified in IRR and registry DB) to the upstream provider of the victim. The list of the upstream providers is available and can be obtained from the IRR. In the Email the attacker instructs the upstream provider to update the input filters and drop the network (that the attacker wishes to sell). Such requests are common, and do not require authentication or verification of requester's identity (e.g., with PGP/PKI) and the *sole source of truth that is checked prior applying the filters is RIPE DB* which, as we mentioned, the attacker can update via the SSO account.

Notice that if the buyer is colluding with the attacker - this step is not needed.

(4) Email a scanned IPv4 resource transfer contract to RIPE NCC. The contract has to match the company details of the victim and contain all formalities and certifications appropriate to the legal system in which the contract has been made. Moreover the contract has to be supplemented by extract from chamber of commerce or appropriate commercial registry that makes it possible to establish that the contract is signed by the eligible persons on both sides. This is simple to forge - the attacker can find and copy the signatures of the owner LIR online and paste them into the scanned contract document that the attacker prepares.

Extract from chamber of commerce is also simple to forge - for instance, in Czech Republic (CZE) the attacker has to go to any post office, pay 1 EUR and get either paper version with a stamp or PDF with PKI signature of state-operated CA. In any case, since the attacker only needs to send a scanned version of the document to RIPE, the attacker can get the document for any company and adjust it using Photoshop and it is accepted.

Finally, notice that the RIRs have limited personnel which, depending on the RIR, may need to deal with tens of transfers per day, see Figure 7. As a result, the adversary may often manage to sell the resources without raising alarms even

when not satisfying these simple four steps. For instance, RIPE NCC has just 24 employees responsible for IP address distribution⁵ and there are more than 20 transfers per day, see Figure 7.

User and role management. Attacker that controls an account with *administrator* role assigns other newly created users either *operator* or *administrator* roles for the victim LIR. This effectively hides the activities of the attacker for long periods of time even though the legitimate holder is actively operating the LIR.

Modification of the LIR contacts and details. There are two sets of Email and postal addresses and phone numbers related to the LIR - the first set is published in IRR and RIPE DB and it is tied to the resources and published to facilitate operation of the network and solving technical problems and those contacts are also used by the RIPE DB software itself for generating notifications about changes in RIPE DB. The second set contains the designated LIR contact information, namely the primary point of contact for the LIR and a contact for billing-related matters. Moreover, there is a postal address, that may differ from legal address of the LIR company and it can be modified in LIR portal. The attacker can redirect or change the LIR contact information to avoid detection by the victim LIR staff when activities that result in notifications or follow-up Emails are to be executed. Modifying LIR contacts will also make any attempt to rectify damages caused by the attack, when detected, harder.

Termination of LIR membership. Attacker initiates termination of LIR membership with the RIPE NCC by submitting a forged termination request via a written notice sent by Email. Forgery cases are not new and have already been seen in the past⁶.

Modification of LIR organisation name, legal address and VAT number. The attacker steals the LIR and all its IP addressing resources by pretending a transfer of ownership to other company. A scan of (the forged) contract of company acquisition has to be attached to the request in electronic form.

Requesting new or voluntarily returning IP addressing resources. The attacker requests or returns IP addressing resources from or to RIPE NCC. If the LIR is eligible for allocation of any scarce resources, the attacker obtains a new IP prefix that is not used in default-free zone (DFZ) and thus fulfils the prerequisite for transfer of not being announced. However, according to the current policy the newly obtained allocation of scarce resources (IPv4 addresses, 16-bit ASN) can not be transferred within 24 months from the allocation. The attacker can nonetheless hijack the resources for own purposes immediately and attempt to sell the resources after the grace period.

⁵<https://www.ripe.net/about-us/staff/structure/registration-services>

⁶<https://mailman.nanog.org/pipermail/nanog/2011-August/039379.html>

6.2 Domain Registrars

Domain registrars handle the ownership of domains in name of the customer. We map the 100K-top Alexa domains to registrars with whom these domains are registered in Table 1. We demonstrate exploits that the attacker can perform when taking over an account with GoDaddy⁷. The adversary can change the nameservers' IP addresses, which allows it to hijack the victim customer domain. This can be exploited to redirect clients to phishing websites. The adversary can delegate account access to itself or perform intra-and inter-registrar⁸ domain transfer. The adversary can change the Email forwarding settings of the account which would allow it to hijack the Emails forwarded to the owner of the Email address. The adversary can also delete the domains associated with the compromised account and even close the account. The account owners can enable two-factor authentication for manipulation of resources associated with their account. However, this is not enabled by default, and is up to each customer to enable it.

6.3 Infrastructure as a Service

We evaluated the exploits that the adversary can carry out on resources associated with accounts at cloud providers. The adversary can manipulate virtual resources associated with the account, such as virtual machines, network interfaces, disk. The adversary can also exploit these resources to carry out attacks against victims in the Internet or victims located on the same cloud platform, e.g., via side channels [39]. In addition the adversary can create new accounts with owner privileges or transfer subscription to another Azure account.

6.4 Certificate Authorities

When controlling an account of a customer with a CA the adversary can revoke certificates and reissue existing certificates that were issued under that account. This allows to change the key-pair associated with a certificate. Nevertheless, some CAs, do not enforce any validation on reissuing certificates, in contrast to validation (domain validation, organisation validation or extended validation) that is enforced when requesting to issue a certificate for the first time. Therefore, instead of attacking domain validation procedure of the CAs it is more profitable to hijack a victim customer account and change the keys associated with the certificates for domains that the adversary targets. DigiCert and GoDaddy do not perform additional validation on requests to reissue certificates. Sectigo, GlobalSign, IdenTrust validate all requests to reissue certificates.

⁷The other domain registrars allow similar actions.

⁸It takes 60 days for an inter-registrar transfer to finalise.

7 Vulnerable Digital Resources

The large fraction of the accounts under different providers that can be hijacked is alarming. Even more disturbing are the exploits that the adversaries can do with the resources assigned to the accounts. *What is the extent of the Internet resources that are at immediate risk due to the vulnerable providers and vulnerable customers?*

To answer this question we perform a correlation between the accounts that our study found to be vulnerable to hijacks via any of the attack methodologies in Section 4 and the digital resources (domain names, IP addresses and ASNs) that the attacker can take over as a result of hijacking that account. In our analysis we consider only the domain registrars and the RIRs and their customers. Since there is no public database of customers of cloud providers and certificate authorities we exclude them from this analysis⁹. We list the correlation between the vulnerable resources and the vulnerable accounts in Table 6.

IP resources. We compute the fraction of the assigned AS Numbers (ASNs) as well as assigned IPv4 address space which could be taken over by hijacking the vulnerable accounts of customers of RIRs. For this purpose, we combine IP-to-ASN and ASN-to-LIR mappings with our customer vulnerability data, which allows us to evaluate vulnerabilities in 73% of the assigned IPv4 address space (for 27% we could not extract LIR account information). Our results show that using any of the attack methodology in Section 4 the adversary could take over 68% of the IPv4 address space. This constitutes 93% of the address space assigned to the accounts in our dataset. Even the weaker attack methodologies (FragDNS and SadDNS), which do not require controlling a BGP router, allow the adversaries to take over 59% of the address space. Similarly, 74% of the ASNs are associated with the accounts that can be hijacked via any of the DNS cache poisoning attacks in Section 4 and 30% with the SadDNS or FragDNS attack. The difference between the vulnerability volume for IP addresses and ASNs is due to the fact that large parts of the IPv4 address space is owned by a small number of ASes, e.g., 21% of the assigned IPv4 address space is attributed to the top 10 LIR accounts.

Domain resources. We use our domain-to-account mapping to determine user accounts at registrars. This includes 11% of the accounts for which we were able to extract customer account information. We believe however that the fraction of the vulnerable accounts is representative of all the 1M-top Alexa domains since the vulnerabilities only depend on the nameservers of the customers' domains. Our study shows that 65% of the domains could be hijacked via any of HijackDNS, SadDNS or FragDNS, while 35% could be hijacked via SadDNS or FragDNS.

⁹These can be collected via a dictionary attack against the provider: the adversary inputs usernames and checks for error messages. Such a study however creates a significant load on the infrastructure of the provider.

	HijackDNS	SadDNS	FragDNS	Any	SadDNS or FragDNS
IP addresses	81%	30%	51%	93%	59%
Domains	47%	10%	27%	65%	35%

Table 6: Vulnerable resources mapped to accounts in our dataset.

Countermeasure	Layer	Provider- / Customer-side	FragDNS	SadDNS	HijackDNS
			✓	✓	✓
2-FA TAN with out-of-band notif.	Web portal	both ¹	✓	✓	✓
2-FA login	Web portal	provider	✓	✓	✓
IP-level account access restrictions	Web portal	both	✓	✓	✓
DNSSEC signing and validation	DNS	both	✓	✓	✓
Disable/Patch ICMP rate-limit	IP	provider	✗	✓	✗
Disable NS rate-limit	DNS	customer	✗	✓	✗
Disable PMTUD	IP	customer	✓	✗	✗
Blocking Fragments	IP	provider	✓	✗	✗
MTA-STX [RFC8461]	Email	both	✓	✓	✓
Hide public account details	General	both	✓	✓	✓
Request rate-limiting	Web portal	provider	✓	✓	✗
Captchas	Web portal	provider	✓	✓	✗
Separate systems	Web portal	provider	✓	✓	✗
Resolver hardening	DNS	provider	✓	✓	✗
Non-predictable IPID increment	IP	customer	✓	✗	✗
Out-of-band notifications	Web portal	provider	✓	✓	✓

Table 7: Countermeasures against different types of attackers. ¹: requires the user to verify the out-of-band delivered transaction details before entering the TAN.

8 Recommendations for Countermeasures

The fundamental problem that our attacks outline is the stealthiness and ease with which the adversaries can apply changes and manipulations over the Internet resources of providers of digital resources. Since Internet resources form the foundations for the stability and security of democratic societies, our work calls for a revision of the current practices of resource management and development of techniques that would secure the transactions over the Internet resources. For instance, selling Internet blocks should not happen immediately, and should require more than merely a scanned document over Email (which is easy to fake). In addition to the standard recommendations for hardening the DNS caches or blocking ICMP error messages, which we summarised in Table 7, we also provide recommendations for best practices for providers and customers.

Separate system for high privileged users. Currently, any user can create an account with most of the providers. The accounts can be used for managing Internet resources (high privileged) as well as for registering for events or mailing lists (low privileged). Low privileged accounts in the user management system have access to the same infrastructure (Email servers, DNS resolvers, etc) as the high privileged accounts, such as those of network operators. This enables adversaries to open low privileged accounts and use them to collect information about the infrastructure of the provider. The providers

should use separate user management systems and a separate set of servers for users which own digital resources vs. users that, e.g., are registered to mailing lists or events.

Two-factor authentication. Two factor authentication (2-FA) systems must be enabled by default. The two authentication factors must be independent of each other and an attacker should not be able to compromise both factors within a single attack. This for instance, rules out Email-based 2-FA for password recovery which is available at some of the providers we tested.

Deploy captchas. Our study shows that most providers do not use captchas, e.g., three out of five RIRs do not use captchas. Although captchas do not prevent the attack, they force the attacker to run manual tests making the attack more expensive to launch. Resolving the captchas is tedious and burdensome for the attacker (as well as for the researchers) to carry out in contrast to automated study of the victims. For instance, for studying vulnerabilities in DNS caches and for performing cache poisoning attacks we needed to run multiple password recovery procedures for triggering DNS requests to our domain. This study could not be automated in RIRs that use captchas.

Notifications of modifications. Changes performed over the resources of providers either do not generate any notifications or generate notifications to the Email configured in the compromised account. First, the Email notifications will be received by our adversary, since it hijacked the victim domains in the resolver of the provider, and second, the adversary can change the contact Email in the account, and even disable notifications. The accounts with providers should be associated with contact Email which cannot be changed through the account and which is different than the one used to access the account.

Email address masking. The Email addresses in the `whois` records of the domains should be masked. Some of the domain registrars are already following this practice.

Account level IP address access restriction. The registrars should restrict account access to only few static IP addresses belonging to the domain's owner.

Deploy DNSSEC. DNSSEC ([RFC4033] to [RFC4035]) would essentially make the attack methodologies in Section 4 practically impossible. Unfortunately, only 3.78% domains of customers of RIRs and 5.88% domains of customers of registrars are correctly signed. For instance, out of 1832 LIR domains under AFRINIC, only 58 are signed, and 27 of these domains are still vulnerable since the DNS resolvers cannot establish a chain of trust to them from the root anchor. Further, 12 use weak cryptographic keys (below 512 bits) and 12 use weak (vulnerable) hash functions. The remaining 95 domains out of those 1832 domains were not responsive. Unfortunately, even when the domain is signed and the resolver validates DNSSEC, as long as the human factor is in the loop, there is risk for vulnerabilities and misconfigurations, [12, 42]. Hence we recommend that the providers and customers de-

ploy additional measures that we list in this section to harden their infrastructure.

9 Conclusions

Each provider maintains a database that defines which customer owns which Internet resources and offers tools for the customers to manage their resources. We showed that these databases are poorly protected - the adversaries can take over the accounts for managing the Internet resources and can manipulate the databases, e.g., creating new or removing existing objects - stealthily and causing immediate changes to the customers' resources.

For our attacks we used different DNS cache poisoning methodologies and compared their applicability and effectiveness for taking over accounts. Our work shows that while challenging, our attacks are practical and can be applied against infrastructure of a large fraction of the resource providers to hijack accounts. Our results demonstrate feasibility even with weak off-path adversaries. Certainly, accounts associated with Internet resources are an attractive target also for stronger Man-in-the-Middle adversaries, such as cyber-criminal groups or nation state attackers.

We described countermeasures for mitigating our off-path attacks for taking over the accounts of customers. Addressing the fundamental problem - easy manipulation of the Internet resources - requires creating policies and revising the Internet management infrastructure as well as techniques for securing the transactions applied over Internet resources.

Acknowledgements

This work has been co-funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) SFB 1119.

References

- [1] ripe-720: Ipv4 address allocation and assignment policies for the ripe ncc service region. <https://www.ripe.net/publications/docs/ripe-720>. Accessed: 2019-6-13.
- [2] BGPStream by CAIDA. <https://bgpstream.caida.org/>, July 2016.
- [3] The CAIDA AS Relationships Dataset. <http://www.caida.org/data/as-relationships/>, January 2016.
- [4] Louis Poinsignon. BGP leaks and cryptocurrencies, 2018.
- [5] F. Alharbi, J. Chang, Y. Zhou, F. Qian, Z. Qian, and N. Abu-Ghazaleh. Collaborative client-side dns cache poisoning attack. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019.
- [6] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 375–392. IEEE, 2017.
- [7] Dan J. Bernstein. DNS Forgery. <http://cr.yp.to/djb dns/forgery.html>, November 2002.

- [8] Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. Bamboozling certificate authorities with BGP. In *27th USENIX Security Symposium (USENIX Security 18)*, 2018.
- [9] Henry Birge-Lee, Liang Wang, Daniel McCartney, Roland Shoemaker, Jennifer Rexford, and Prateek Mittal. Experiences deploying multi-vantage-point domain validation at let's encrypt. December 2020.
- [10] Peter Boothe, James Hiebert, and Randy Bush. Short-lived prefix hijacking on the internet. In *Proc. of the NANOG*, 36, 2006.
- [11] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. Domain Validation++ For MitM-Resilient PKI. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2060–2076. ACM, 2018.
- [12] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. A longitudinal, end-to-end view of the dnssec ecosystem. In *USENIX Security*, 2017.
- [13] Taejoong Chung, Roland van Rijswijk-Deij, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. Understanding the role of registrars in dnssec deployment. In *Proceedings of the 2017 Internet Measurement Conference*, pages 369–383, 2017.
- [14] Tianxiang Dai, Philipp Jeitner, Haya Shulman, and Michael Waidner. From IP to Transport and Beyond: Cross-Layer Attacks Against Applications. In *SIGCOMM '21: Proceedings of the 2021 Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication, Virtual Event, USA, August, 2021*. ACM, 2021.
- [15] Chris C Demchak and Yuval Shavitt. China's maxim—leave no access point unexploited: The hidden story of china telecom's bgp hijacking. *Military Cyber Affairs*, 3(1):7, 2018.
- [16] Doug Madory. Recent Routing Incidents: Using BGP to Hijack DNS and more, 2018.
- [17] Amir Herzberg and Haya Shulman. Security of Patched DNS. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, pages 271–288, 2012.
- [18] Amir Herzberg and Haya Shulman. Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In *IEEE CNS 2013. The Conference on Communications and Network Security, Washington, D.C., U.S. IEEE*, October 2013.
- [19] Tomas Hlavacek, Italo Cunha, Yossi Gilad, Amir Herzberg, Ethan Katz-Bassett, Michael Schapira, and Haya Shulman. Disco: Sidestepping rpki's deployment barriers. In *Network and Distributed System Security Symposium (NDSS)*, 2020.
- [20] Tomas Hlavacek, Amir Herzberg, Haya Shulman, and Michael Waidner. Practical experience: Methodologies for measuring route origin validation. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018*, pages 634–641, 2018.
- [21] P Hoffman and P McManus. Rfc 8484: Dns queries over https (doh), 2018.
- [22] Z Hu, L Zhu, J Heidemann, A Mankin, D Wessels, and P Hoffman. Rfc 7858-specification for dns over transport layer security (tls), 2016.
- [23] P. Jeitner, H. Shulman, and M. Waidner. the impact of dns insecurity on time. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.
- [24] Philipp Jeitner and Haya Shulman. Injection Attacks Reloaded: Tunneling Malicious Payloads over DNS. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, August 2021.
- [25] Josh Aas and Daniel McCartney and and Roland Shoemaker. Multi-Perspective Validation Improves Domain Validation Security, 2020.
- [26] Dan Kaminsky. It's the End of the Cache As We Know It. Presentation at Blackhat Briefings, 2008.
- [27] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. Autonomous security for autonomous systems. *Computer Networks*, 52(15), 2008.
- [28] Varun Khare, Qing Ju, and Beichuan Zhang. Concurrent prefix hijacks: Occurrence and impacts. In *Proceedings of the 2012 Internet Measurement Conference*, pages 29–36. ACM, 2012.
- [29] Amit Klein. Bind 9 dns cache poisoning. *Report, Trusteer, Ltd*, 3, 2007.
- [30] Amit Klein. Windows dns server cache poisoning," 2007.
- [31] Amit Klein. Cross layer attacks and how to use them (for dns cache poisoning, device tracking and more). *arXiv:2012.07432*, 2020.
- [32] Aanchal Malhotra and Sharon Goldberg. Attacking NTP's Authenticated Broadcast Mode. *ACM SIGCOMM Computer Communication Review*, 46(1):12–17, May 2016.
- [33] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.
- [34] Carolyn Duffy Marsan. Six worst internet routing attacks, 2009.
- [35] Asya Mitseva, Andriy Panchenko, and Thomas Engel. The state of affairs in bgp security: A survey of attacks and defenses. *Computer Communications*, 124:45–60, 2018.
- [36] Andreas Reuter, Randy Bush, Ítalo Cunha, Ethan Katz-Bassett, Thomas C. Schmidt, and Matthias Wählisch. Towards a rigorous methodology for measuring adoption of RPKI route validation and filtering. *CoRR*, abs/1706.04263, 2017.
- [37] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang. Bgp routing stability of popular destinations. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, pages 197–202. ACM, 2002.
- [38] RIPE NCC. RIS Raw Data, 2021.
- [39] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212, 2009.
- [40] Haya Shulman. Pretty bad privacy: Pitfalls of dns encryption. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 191–200, 2014.
- [41] Haya Shulman and Michael Waidner. Towards security of internet naming infrastructure. In *European Symposium on Research in Computer Security*, pages 3–22. Springer, 2015.
- [42] Haya Shulman and Michael Waidner. One key to sign them all considered vulnerable: Evaluation of dnssec in the internet. In *NSDI*, 2017.
- [43] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. Encrypted dns-> privacy? a traffic analysis perspective. *arXiv preprint arXiv:1906.09682*, 2019.
- [44] University of Oregon. Route views project. <http://bgplay.routeviews.org/>, 2012.
- [45] Paul Vixie. DNS and BIND security issues. In *Proceedings of the 5th Symposium on UNIX Security*, pages 209–216, Berkeley, CA, USA, jun 1995. USENIX Association.
- [46] S Weiler and D Blacka. Rfc 6840: Clarifications and implementation notes for dns security (dnssec). *IETF Standard*, 2013.
- [47] Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan, and Zhiyun Qian. Poison over troubled forwarders: A cache poisoning attack targeting DNS forwarding devices. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 577–593, 2020.

A.8. Let's Downgrade Let's Encrypt

[Dai21c]

Tianxiang Dai, Haya Shulman, and Michael Waidner. “Let's Downgrade Let's Encrypt”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 1421–1440. ISBN: 9781450384544. DOI: 10.1145/3460120.3484815. URL: <https://doi.org/10.1145/3460120.3484815>

Let's Downgrade Let's Encrypt

Tianxiang Dai
ATHENE Center
Fraunhofer SIT
Germany

Haya Shulman
ATHENE Center
Fraunhofer SIT
Germany

Michael Waidner
ATHENE Center
TU Darmstadt & Fraunhofer SIT
Germany

ABSTRACT

Following the recent off-path attacks against PKI, Let's Encrypt deployed in 2020 domain validation from multiple vantage points to ensure security even against the stronger on-path MitM adversaries. The idea behind such distributed domain validation is that even if the adversary can hijack traffic of some vantage points, it will not be able to intercept traffic of all the vantage points to all the nameservers in a domain.

In this work we show that two central design issues of the distributed domain validation of Let's Encrypt make it vulnerable to downgrade attacks: (1) the vantage points are selected from a small fixed set of vantage points, and (2) the way the vantage points select the nameservers in target domains can be manipulated by a remote adversary. We develop off-path methodologies, based on these observations, to launch downgrade attacks against Let's Encrypt. The downgrade attacks reduce the validation with '*multiple vantage points to multiple nameservers*', to validation with '*multiple vantage points to a single attacker-selected nameserver*'. Through experimental evaluations with Let's Encrypt and the 1M-Let's Encrypt-certified domains, we find that our off-path attacker can successfully launch downgrade attacks against more than 24.53% of the domains, rendering Let's Encrypt to use a single nameserver for validation with them.

We then develop an automated off-path attack against the 'single-server'-domain validation for these 24.53% domains, to obtain fraudulent certificates for more than 107K domains, which constitute 10% of the 1M domains in our dataset.

We also evaluate our attacks against other major CAs and compare the security and efforts needed to launch the attacks, to those needed to launch the attacks against Let's Encrypt. We provide recommendations for mitigations against our attacks.

CCS CONCEPTS

• Security and privacy → Network security.

KEYWORDS

PKI, BGP hijacks, DNS Cache Poisoning, Server Selection

ACM Reference Format:

Tianxiang Dai, Haya Shulman, and Michael Waidner. 2021. Let's Downgrade Let's Encrypt. In *Proceedings of the 2021 ACM SIGSAC Conference*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8454-4/21/11...\$15.00

<https://doi.org/10.1145/3460120.3484815>

on *Computer and Communications Security (CCS '21)*, November 15–19, 2021, Virtual Event, Republic of Korea. ACM, New York, NY, USA, 20 pages. <https://doi.org/10.1145/3460120.3484815>

1 INTRODUCTION

Identifying the legitimate owner of a domain plays a central role in the security of Public Key Infrastructure (PKI). It prevents criminals from obtaining fraudulent certificates for domains that they do not own. Prior to issuing certificates the Certificate Authorities (CAs) run domain validation (DV) against services in a domain that is to be certified, to verify that the domain owner de-facto controls the domain. To verify control a CA generates a challenge which the domain owner should integrate into the selected service in a domain, e.g., add the challenge in a TXT record to the zonefile of the domain or add the challenge to a directory of the website in the domain. The CA then checks the presence of the challenge by querying the selected service in the target domain. Since the challenge was sent to the domain, a genuine owner can receive it and hence can respond correctly. In contrast, an off-path adversary that does not control the domain, cannot receive the challenge and therefore should not be able to respond correctly.

Domain validation from single vantage point is vulnerable. Recently [14] showed an off-path attack against domain validation of popular CAs: the attacker hijacks the challenge sent by the CA to the domain during the validation of control over the domain. This allows the attacker to respond with the correct challenge and demonstrate control over a domain that it does not legitimately own. The significance of PKI for Internet security, coupled with the risks that the attacks introduced, triggered efforts to improve the security of domain validation.

Man-in-the-Middle secure distributed domain validation. Let's Encrypt was the first CA to react quickly to the disclosed vulnerabilities. It initiated efforts to enhance the security of DV even against on-path Man-in-the-Middle (MitM) adversaries, standardising a mechanism called ACME in 2019, [RFC8555] [13], and in 2020 it deployed in production environment a mechanism called multiVA [36] - domain validation with multiple Validation Authorities (VAs). Initially Let's Encrypt set up four VAs, each running a DNS resolver software for looking up resources in domains and for validating control over domains. Upon request for a certificate, the VAs perform lookup of the target domain by sending queries to the nameservers and then concurrently validate control over the domain. Each VA receives the set of nameservers and their IP addresses from the parent domain. The VA then randomly selects a nameserver to which the query is sent. If the majority of the VAs receive the same results, DV succeeds, and the certificate is issued. Otherwise, the request fails. Let's Encrypt shows that their setup with multiVA provides security for DV even against MitM adversaries: the intuition is that realistic MitM adversaries are limited in

their power, and can control or hijack some, but not many of the Internet networks. Recently [15] performed simulations to show that the diverse vantage points of multiVA allow to detect 94% of the BGP prefix hijack attacks during DV, making more than 90% of the ASes in the Internet topologically incapable of launching BGP attacks against the majority of domains. This is in contrast to the previous deployment of Let's Encrypt, where most domains were vulnerable to prefix hijacks during DV.

What about the multiple nameservers? In this work we show that in addition to considering the vantage points as in [14, 15], it is important to also consider the *domain side of domain validation*. The analysis in [15] used a single IP address for each domain. Nevertheless, instead of intercepting the query from the vantage point, the adversary can also intercept the response from the domain. This appears to expose domains to practical attacks during DV. In practice, however, hijacking the domain is challenging: domains have multiple nameservers, in fact, some domains have even more than 30 nameservers, see Figure 1 in Section 2. Hence the situation becomes very complex even for a MitM adversary. To demonstrate control over a target domain the attacker would need to hijack multiple challenges, sent by the vantage points. To complicate the situation further, these challenges are not sent all to the same nameserver, but each vantage point selects the nameserver, to which the challenge is sent, uniformly at random. If the attacker cannot anticipate which vantage point sends a query to which nameserver, to beat the domain validation it would have to craft multiple different responses. That indeed should make the attack against all the vantage points for all the nameservers impractical, even for strong on-path adversaries.

'The Downgrade' attack. In this work we develop a downgrade attack that reduces the multiVA validation against real domains that have multiple nameservers, to a validation against domains with a *single* nameserver. Our attack is based on two observations: (1) a functionality in VAs, designed to enhance security and performance, can be manipulated by network adversaries remotely and (2) Let's Encrypt uses a small and fixed set of VAs. The former manipulates the server selection by the VAs, causing the multiVA to execute against a single nameserver, one which all the VAs select for validation and lookups. The latter allows launching targeted efficient attacks against the VAs in advance, as a preprocessing step, before initiating the attack to obtain fraudulent certificates.

We show that combining our two observations the network adversaries can eliminate the *multiple VAs to multiple nameservers* effect, creating a 'multiple VAs to single nameserver' situation... which is *no longer secure against MitM adversaries*. In the course of the attack we cause the VAs to eliminate the nameservers from the list of usable servers, leaving only a single available nameserver. Worse, we show that the attacker can not only reduce the validation to one arbitrary nameserver, but force all the VAs to query a specific nameserver of attacker's choice, one which has a vulnerability that can be exploited by the attacker, e.g., server with unpatched software or server that can be attacked with side channels or fragmentation, [18, 46]. In this work, as an example, we select servers whose BGP prefix the attacker can hijack via sub-prefix hijack attacks.

Off-path attacks against Let's Encrypt. The core issues which expose domains to downgrade attack are a side effort of server

selection functionality of Let's Encrypt. To exploiting them against a specific victim domain the adversary needs to introduce a pattern into the responses from the nameservers. When the VAs receive a certain pattern of missing responses they block the nameservers. We explain that there are different ways to exploit this vulnerability and introduce a pattern into the responses, e.g., with a compromised router which selectively drops or manipulates some specific packets. We show how to exploit this vulnerability even with an *off-path adversary*.

We develop 'server-elimination' methodologies to introduce losses according to specific intervals, causing all the VAs to query just one nameserver, selected by the attacker. Some of our methodologies assume specific properties in domains, such as rate limiting, and hence can be launched only against the domains which have these properties, e.g., 24.53% of Let's Encrypt-certified domains, see Section 3. We also developed a generic server-elimination methodology, which applies to all the domains. This method however requires generating much more traffic than the other methods. Furthermore, as we mentioned the vulnerability in the CAs that allows downgrading the number of nameservers in a domain can also be exploited with stronger adversaries.

Fraudulent Let's Encrypt certificates. After downgrading validation with domains to a single nameserver, we launch attacks to prove control over domains that off-path adversaries do not own and obtain fraudulent certificates for these domains.

We compare the security of Let's Encrypt to that of other popular CAs and show that the downgrade attack eliminates the security benefits introduced by multiVA. In fact, we found all the CAs equally vulnerable to our attacks. This implies that the validation of all the CAs in our dataset can be downgraded to a single server in any Internet domain. We run a complete attack against the domains in our dataset that have properties which allow our off-path server-elimination, and force the validation to run against a single nameserver, which sub-prefix can be hijacked. This constitutes 10.6% of our 1M dataset. We proceed to obtain fraudulent certificates for these 108K domains.

Ethical considerations. Our attacks, evaluations and measurements were ethically carried out against CAs and domains in our dataset. We notified Let's Encrypt about the downgrade attacks.

Contributions. We make the following technical contributions:

- We develop a taxonomy of nameserver elimination methodologies which force the VAs of Let's Encrypt to query a nameserver of attacker's choice. One methodology is generic, it uses low-rate bursts to cause packet loss and applies to *any nameserver in any domain*. We did not evaluate this methodology in the Internet since it adversely affects communication from other sources with the nameservers. The other two methodologies require that the nameservers apply rate-limiting or fragment responses, and generate less traffic. We evaluate them on our dataset of domains to show that more than 20% of 1M-top Alexa domains¹ and 24.5% Let's Encrypt domains are vulnerable. We show that our methodologies, with slight modifications, apply also to other popular CAs. Our server elimination methodologies potentially have a wider application scope. For instance, they can be applied to redirect clients to the

¹Of 1M-top Alexa domains, 857K-top domains were responsive, without errors.

wrong server, introducing traffic fluctuations to the load balancing that the CDNs and cloud platforms use.

- Our server-elimination methodologies exploit properties in nameserver selection of DNS implementations. We perform analysis of nameserver selection in Unbound, ‘reverse engineer’ its behaviour and show that it can be remotely manipulated to cause DNS resolvers to block nameservers.

- To evaluate our attacks ethically we develop a two-sided methodology. In contrast to prior work which performed simulations or evaluated attacks only in a lab setup, our evaluation methodology allows to launch and validate real attacks in two steps. We first attack the target CA with a victim domain that we own. Our adversarial host, located on a different network than the victim domain, obtains a fraudulent certificate for the victim domain. This allows us to evaluate the vulnerability and applicability of the attack against Let’s Encrypt, yet without issuing fraudulent certificates for real victim domains. In a second step, we reproduce the setup of Let’s Encrypt on our networks, with all the relevant components, and launch automated attacks against our dataset of Let’s Encrypt-certified victim domains, issuing fraudulent certificates for these domains with a CA controlled by us. This second step allows us to identify victim domains to which our attacks apply. If the attack applies in both steps, it also applies when launched against the CA and the victim domain in real life. Our evaluation methodology has wider applicability, it can enable ethical evaluations of other attacks yet without causing damage to real victims. For instance, it can be used to evaluate different types of Denial of Service (DoS) attacks, such as fragmentation based DoS attacks.

- Our work shows that validation from multiple locations, although the right way to go, is not trivial, and requires care to avoid pitfalls. We provide recommendations for preventing our attacks.

Organisation. In Section 2 we develop our downgrade attack against Let’s Encrypt. We develop and evaluate nameserver elimination methodologies in Section 3. In Section 4 we demonstrate attacks against Let’s Encrypt to issue fraudulent certificates and evaluate them against a dataset of 1M domains certified by Let’s Encrypt. We provide recommendations for countermeasures in Section 5. Comparison to related work is in Section 6. We conclude this work in Section 7.

2 THE DOWNGRADE ATTACK

We develop a downgrade attack against Let’s Encrypt to reduce the ‘multiple VAs to multiple nameservers’ validation to ‘multiple VAs to attacker selected nameserver’ validation. Our attack is based on an observation that a functionality in VAs, which is used to increase security and performance, can be manipulated by a remote adversary. Specifically, the DNS software at each VA selects uniformly at random the nameserver to which queries are sent. This is required in order to distribute the load from all the VAs evenly among all the nameserver as well as to create unpredictable selection of nameservers by the VAs, and finally, to ensure good performance by avoiding poorly performing nameservers.

The fact that the VAs are selected from a small and a fixed set of nodes, which is known to the attacker, allows the attacker to manipulate the server selection mechanism in advance, prior to requesting a fraudulent certificate for domain that it does not control. As a

result, the validation of control over the victim domain, during the certificate issuance, is performed against a single attacker-selected nameserver.

In this section we explain the server selection mechanism (Section 2.1), and its implementation in the VAs of Let’s Encrypt (Section 2.2). Surprisingly, we show that off-path adversaries can influence the server selection function at the VAs. To manipulate the server selection we develop a server-elimination attack, forcing all the VAs of Let’s Encrypt to query a nameserver of attacker’s choice (Section 2.3). Server-elimination attack not only reduces the entropy from server selection, but also forces all the VAs to communicate with a server of attacker’s choice.

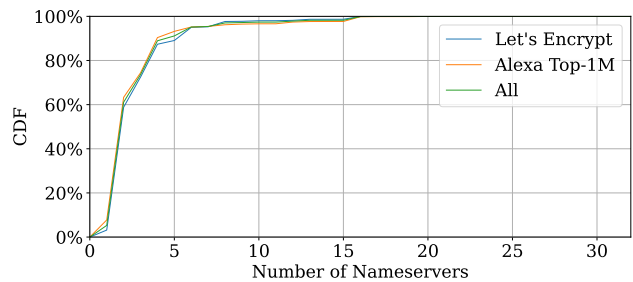


Figure 1: Number of nameservers per domain.

2.1 Server Selection

Traditionally, there were up to 13 nameservers in each domain, to fit DNS responses in 512 bytes UDP packet. After the adoption of EDNS [RFC6891] [25] the limit on number of nameservers per domain was removed, allowing each domain to configure arbitrary number of nameservers. Our measurements show that on average domains have more than 3 unique IP addresses and that there are domains with more than 30 nameservers, Figure 1.

To ensure performance as well as to balance the load of queries among the nameservers, the DNS resolver implementations use different logic for selecting the nameservers in the target domain. The implementations typically prefer most available servers with low latency. To select the server the DNS resolver monitors the performance of each nameserver in a domain and applies a computation over the responsiveness of individual nameservers as well as the latency.

A number of studies explore the impact of DNS server selection on load distribution [59, 61] and attempt to optimise performance by only selecting fast nameservers and quickly reacting to changes in nameserver performance [26]. Server selection also has implications for security of DNS, making it more difficult to launch cache poisoning attacks since an off-path adversary cannot anticipate which nameserver a target resolver will query [RFC5452] [34].

2.2 Analysis of Let’s Encrypt Server Selection

We perform an analysis of server selection behaviour of the VAs by triggering queries to our domain and record the query behaviour of the VAs. We then reproduce the same experiment in a lab environment using popular DNS software and compare produced DNS requests pattern to the one exhibited by the DNS software on the

VAs of Let's Encrypt. We then can determine the software used on the VAs.

2.2.1 Experiment with Let's Encrypt. We describe the setup, our evaluation and the results.

Setup. In this experiment we use 20 domains that we registered. We set up 5 nameservers, and configure each domain with these 5 nameservers. Each nameserver has 20 zonefiles, one for each domain. The nameservers are placed in different regions: NS1 on our AS 1, registered under RIPE NCC, NS2 on region USA west (Oregon), NS3 on region USA west (north California), NS4 on region Canada (central) and NS5 on region USA east (Ohio). The latencies between the VAs of Let's Encrypt to our nameservers ranges between 50ms and 200ms. We set the TTL (Time to Live) of our nameservers to 10 seconds.

Evaluation. We use the Certbot to request certificates for our 20 domains and monitor the DNS requests received on our nameservers. This causes the four VAs² of Let's Encrypt to issue DNS lookups to our nameservers and then to perform validation against our domains with DNS TXT/CAA. We repeat the evaluation 20 times, one iteration for each domain, and continually monitor the requests from the VAs on our nameservers. The evaluation is carried out in two phases. In the first phase we evaluate server selection during normal conditions. In the second phase we introduce losses and additional latency (between 300ms and 500ms) to responses of some of the nameservers. We monitor the DNS requests from the VAs on the nameservers. After the 20 iterations are concluded we analyse the queries sent by each of the VAs to the nameservers in our domains.

Results. Our findings are that the queries are distributed among the nameservers independent of the geo-location and of the network block on which the nameservers are placed. During the first phase, each VA sends a query to each of the nameservers with equal probability, and each of the nameservers receives roughly an equivalent portion of the queries from each VA. During the second phase, we observe that the VAs distribute the queries among the nameservers which have latency below 400ms uniformly at random. VAs avoid querying poor performing nameservers (with latency above 400ms) as well as nameservers from which a VA experienced two-three consecutive packet losses. These nameservers are avoided for more than 10 minutes. Afterwards, the VAs probe the nameserver again to see if its performance improved. We also find that the DNS software on the VAs of Let's Encrypt imposes an upper bound of 60 seconds on the cached records, irrespective of the TTL on the DNS records that the nameservers return. This, however, does not impact the time that the DNS software avoids querying poorly performing nameservers, since this information is stored in a different cache, called the infrastructure cache, as we explain below.

2.2.2 Analysis on Experimental Platform. In this section we compare the queries pattern in our experiment with Let's Encrypt to patterns generated by popular DNS software, to identify the software used by Let's Encrypt. We reproduce our experiments against

²Let's Encrypt currently requires responses to only three out of four VAs for a DV and lookup to be successful.

Let's Encrypt described in Section 2.2.1 in a controlled environment using the DNS maze³ open-source platform, which offers a reproducible test environment for DNS servers. We set up the nameservers with the same zonefiles as we used in our experiment with Let's Encrypt. We also set up 4 DNS resolvers (that correspond to the 4 VAs of Let's Encrypt). We use network emulator⁴ to introduce latencies and losses to responses from the nameservers (identical to our experimental evaluation with Let's Encrypt). During the executions we run the same set of queries as we did against Let's Encrypt.

We execute the tests in an automated way, each time using a different DNS resolver software on the VAs (using Knot, Bind, Unbound, PowerDNS and MS DNS). The results are listed in Table 1. The query distribution, the blocking time, and the distribution of queries to poorly performing nameservers provides a distinct fingerprint allowing to identify the DNS resolver software. We found that the Unbound DNS had the exact same pattern of queries and server selection as those exhibited by the VAs of Let's Encrypt.

DNS Software	Query distribution to servers	Block (min)	% queries to t.o. servers
Unbound	queries all n servers with <400ms with probability $1/n$	15	1%
Knot	>35% queries to fastest server & 10% to others	10	5%
Bind	>95% queries to fastest server & 1% to others	30	1%
PowerDNS	>97% queries to fastest server & 1% to others	3	1%
Windows DNS	uniform query distribution to available servers	<1	1%

Table 1: Server selection in popular DNS implementations.

2.2.3 Code Analysis of Unbound DNS. The server selection procedure of Unbound DNS software is defined in function `iter_server_selection` of `iter_util.c`. Unbound implements timeout management with exponential backoff and keeps track of average and variance of the response times. For selecting a nameserver, Unbound implements an algorithm in [RFC2988]: it randomly selects any server whose smoothed RTT is between the lowest one and the lowest one + 400ms. If a nameserver becomes unresponsive, a probing phase is performed where a couple of queries probe that nameserver. If timeout occurs, the nameserver is blocked for 900 seconds (`infra-ttl`) and re-probed with one query after that time interval. We provide a more detailed explanation of server selection in Appendix, Section D.1, Figure 14.

2.3 Downgrade by Elimination

Our downgrade attack is carried out by reducing the number of available servers each VA of Let's Encrypt can query, leaving just a single nameserver.

The attacker uses Certbot to request a certificate. This triggers lookups from the DNS resolvers at the four VAs of Let's Encrypt to the nameservers in the target domain. The attacker causes the requests to all the nameservers except one nameserver to timeout - we explain how to do this in next Section. Following a timeout

³<https://gitlab.nic.cz/knot/maze/>

⁴NetEm `tc qdisc`.

the VAs go into exponential backoff, and the DNS requests are retransmitted after RTO, i.e., 376ms. The attacker repeats the attack every 376ms. After 2 consecutive losses the nameserver is moved to `infra_cache` and its `infra_ttl` is set to 900sec. The attacker causes the VAs to block the $n - 1$ nameservers, and to only send the queries to the one nameserver of attacker's choice.

Challenge: how to hit the correct nameserver? Each time a VA sends or resends a query the attacker does not know to which nameserver the query is sent. Hence, the attacker needs to cause the queries to $n - 1$ nameservers to timeout, except the queries sent to the one nameserver that the attacker wants the VAs to be forced to select. After experiencing a timeout the VAs go into exponential backoff, and will resend the queries after RTO^5 ($2 \cdot 376ms$ in the case of Let's Encrypt); for detailed explanation of RTO see Appendix, Section D.2. The strategy of the attacker is therefore to launch the attack every RTO, in order to cause the queries to timeout every $RTO=376ms$. This strategy always 'hits' the queries from all the VAs, both from VAs that are in exponential backoff as well as from VAs that are sending queries for the first time to a nameserver and not as a result of a retry attempt.

Challenge: how many attack iterations required? How many times should the attack be repeated to block $n - 1$ servers and how many queries are required until all the $n - 1$ nameservers are removed from the list of usable servers at all the VAs? To answer these questions we analyse the query retransmission behaviour in Unbound, see Appendix, Section D.2, Figure 15. We find that with a single query the attacker can generate up to 32 timeouts, which result in 32 retries by the DNS software, and can be used to block 6 nameservers in a domain. Since 95% of the domains have up to 6 nameservers, a single query suffices to block nameservers of most domains. In addition, since each VA sends at least two DNS requests (for TXT and CAA records) during each certificate request invocation⁶, with a single certificate request the attacker can block 12-13 nameservers per domain. To block domains with more nameservers the attacker can submit more certificate requests.

Challenge: how to cause responses to timeout? In the next section we develop methodologies that enable even weak off-path attackers to eliminate nameservers in domains during validation with Let's Encrypt. The idea is to make it appear as if the target server has poor connectivity. In one methodology we use IP fragment reassembly to cause mis-association of IP fragments [32, 55]. The resulting (reassembled) UDP packet is discarded by the target resolver itself. Nevertheless, this event is perceived as packet loss by the resolver. In another methodology we use the rate-limiting of the nameservers, to cause the query from the resolver to be filtered. We find both these properties (fragmented DNS responses and rate limiting) in 24.53% of Let's Encrypt-certified domains. We also develop a generic methodology, which does not assume any properties in the nameservers nor domains. The idea is to send low rate bursts to

⁵The RTO is the timeout including the exponential backoff. It is used for server selection and as a timeout for the transmitted request.

⁶Each VA can also send more queries and the exact upper bound of queries depends on the responses from the nameservers. For instance, if the nameserver sends a response with the NS type records with hostnames of the nameservers but without the A type records. The resolver will issue a subsequent request for the A records with the IP addresses of the nameservers.

cause packet loss at the router which requests of the target resolver traverse.

3 SERVER-ELIMINATION METHODOLOGIES

Our key contribution in this section is a taxonomy of methodologies that we develop for off-path server elimination. These methodologies introduce packet losses on the communication between the nameservers and the VAs. The lost packets signal to the DNS software at the VA connectivity problems at the nameserver. The nameserver is then blocked by the VA for 900 seconds. We use these methodologies to launch downgrade attacks against Let's Encrypt.

One methodology is generic and applies to any domain and all nameservers without assuming any properties. The idea is to send bursts to the router that connects the network of the nameserver to the Internet. The traffic bursts never reach the nameserver network, so the attack is stealthy and cannot be detected. We evaluated this methodology ethically in a controlled environment that we set up. The other two methodologies require less traffic but assume that the nameservers in a domain have specific properties. One methodology requires that the nameserver enforces rate limiting on the inbound DNS requests. The other assumes that the responses of the nameserver can be fragmented. We experimentally evaluated these two methodologies against our dataset of domains and found that they apply to 24% of Let's Encrypt-certified domains and 20% of 857K-top Alexa domains.

Since the evaluation is carried out against a large set of almost 2M domains, we automate it. This automated evaluation provides a lower bound on the number of vulnerable domains, since it misses out potentially vulnerable domains; we explain this in Section 3.6 below.

3.1 Dataset

Our dataset contains domains certified with Let's Encrypt as well as 1M-top Alexa domains; the dataset is listed in Table 2. Out of 1M-top Alexa domains only 857K domains were valid with responsive nameservers. We use these 875K-top Alexa domains in the rest of our work. In our study we use domains with Let's Encrypt certificates to infer the fraction of vulnerable customers of Let's Encrypt. We use the popular Alexa domains to infer the overall attack surface of vulnerable domains. The Let's Encrypt and Alexa domains have only a small overlap of 12K domains.

	#Domains	#Nameservers	#ASes	Vuln.
Let's Encrypt	1,014,056	98,502	8,205	24.53%
Alexa	856,887	171,656	15,899	20.92%
Total	1,858,165	227,734	17,864	22.76%

Table 2: Dataset of domains.

3.2 Elimination via Fragmentation

IP fragmentation allows routers to adjust packets to the maximum size that the networks support. Packets that exceed the maximum transmission unit (MTU) are fragmented into smaller fragments by the source or by the routers enroute. The receiver reassembles the fragments back into the original IP packets. To identify the fragments that belong to the same IP packet the receiver uses a 16 bit IP identifier (IP ID) in the IP header of the fragments.

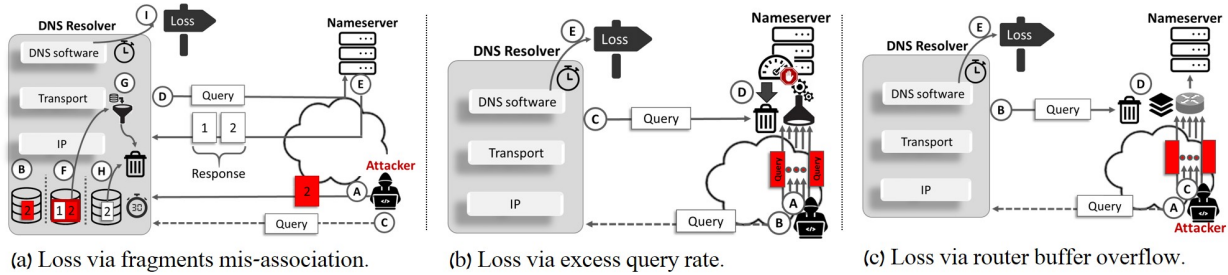


Figure 2: Nameserver elimination via (a) fragmentation, (b) rate-limiting, (c) low-rate bursts.

3.2.1 *Attack methodology.* The idea is to create fragments-misassembly: the attacker injects a spoofed fragment which the IP layer at the VA reassembles with a genuine fragment, sent by the nameserver. The attacker ensures that the resulting IP packet is *invalid* and hence is discarded by the VA. This can be done by violating the resulting length or the transport-layer checksum. The genuine second fragment from the nameserver does not have a matching first fragment, and hence is discarded after a timeout of 30 seconds. This causes the pending query to timeout, and is perceived by the DNS software on the VA as a loss event.

Elimination via fragmentation is illustrated in Figure 2 (a). In step (A) we send a fragment to the VA, from a spoofed IP address of the nameserver. This fragment can be even one byte long. We set the offset of this fragment so that it fits as a second fragment in the sequence of fragments sent by the nameserver. In step (B) this fragment is stored in IP defragmentation cache and stays there for 30 seconds (the default value supported by popular operating systems, such as Linux, FreeBSD and Windows). In step (C) we send a request for a certificate for the target domain. This causes the VA to initiate DNS lookup requests in step (D). For simplicity assume that the nameserver returns a response in two fragments, in step (E). In step (F) the first genuine fragment enters the IP defragmentation cache and is reassembled with the second fragment from the adversary that was waiting in the IP defragmentation cache. For both fragments to be reassembled the spoofed fragment needs to contain the correct IP ID value. The transport-layer processing and checks on the reassembled packet. Since our spoofed second fragment has a different payload than the genuine second fragment, it alters the transport-layer checksum of the packet, which results in an invalid value. The packet is discarded in step (G). In step (H) the second genuine fragment enters the cache; after 30 seconds it is evicted if no matching first fragment arrives. In step (I) timeout is triggered and a loss is registered. The query is resent.

Servers that fragment responses. To cause the nameservers to send fragmented responses we use an ICMP fragmentation needed packet (type: 3, code: 4) indicating that the path to the DNS resolver has a smaller MTU. The nameserver then fragments the response according to the MTU in the ICMP error message and returns it in smaller fragments. Using ICMP error messages with UDP header and ICMP echo reply, we identified that 3% of the domains in our dataset can be forced to fragment responses.

Servers with predictable IP ID. We find 13% of the nameservers with predictable IP ID allocation. For these nameservers the attacker can predict the value of IP ID that the nameservers

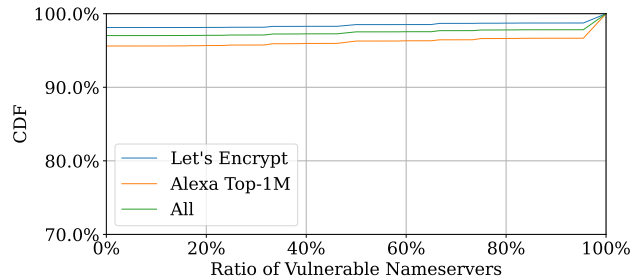


Figure 3: Nameservers per domain vulnerable to frag.

assign to the responses, and use it in the spoofed fragments. We explain how we match the IP ID in the spoofed fragment and how we measured IP ID assignment algorithms in servers in Appendix, Section E.

3.2.2 *Measurements.* We find that 1.88% of Let's Encrypt-certified domains, and 4.39% of 857K-top Alexa domains fragment responses; the results are plotted in Figure 3. The x axis plots the fraction of nameservers per domain that are vulnerable to elimination via fragmentation.

3.3 Elimination via Rate Limiting

Nameservers enforce rate limiting on queries to reduce load and to make it not attractive to abuse them in reflection attacks: after inbound queries⁷ exceed a predefined threshold the nameserver starts dropping packets.

Attack methodology. We devise a methodology that uses 'rate limiting nameservers' to cause the nameserver to filter requests from the victim DNS resolver. The victim DNS resolver perceives the lack of responses as an indication of poor performance of the nameserver and avoids querying it. Elimination via rate limiting is illustrated in Figure 2 (b). The attacker sends multiple requests to the target nameserver using a spoofed IP address of the victim resolver in step (A). The nameserver starts filtering the requests from the DNS resolver. In step (B) the attacker requests a certificate of the target domain. In step (C) the DNS resolver sends DNS query to the nameserver.

⁷Nameservers can apply rate limiting per IP address or overall independent of the IP address.

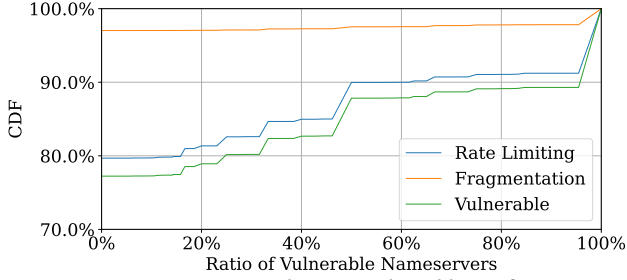


Figure 4: Nameservers per domain vulnerable to frag or rate-limiting.

The nameserver filters that request in step ④. After a timeout is reached, the loss event is registered in step ⑤, and the resolver re-transmits the query. After three consecutive losses, the nameserver is blocked for 15 minutes and will not be queried.

We conduct a study of the nameservers in Alexa domains that limit the rate at which the clients can send DNS requests. We explain our measurement methodology and then report the results.

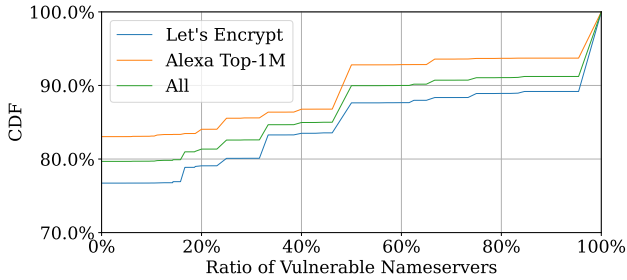


Figure 5: Nameservers per domain vulnerable to rate-limiting.

3.3.1 Measurement of rate limiting. To identify servers that apply rate limiting we use the same setup as described in [46] and perform a similar experiment. Since response rate limiting (RRL) is applied per query per /24 block we capture all the nameservers that start filtering traffic from the victim IP address. In our experiment we send requests with the same query for an A record in the domain of the nameserver concatenated with a non-existent subdomain. Using the same query name reduces the processing overhead imposed on the nameserver and does not cause the nameserver to block other clients sending queries to that domain. We send to each nameserver 4K queries distributed over a time period of a second. We use 4K packet per second (pps), which is roughly 2.5Mbps, to reduce the imposed load on the servers. Recent measurements of traffic rate to servers show that 4K pps is an ethical traffic rate that does not affect the operation of nameservers [46]. We use the overall packet loss as an indicator for rate limit, setting the threshold at 66%, which suffices to cause the nameserver to filter queries from a victim resolver.

We find that the rate limit is typically reached within a second and is enforced in the following 15 seconds. Even with this modest rate of 4K pps, we find that more than 24% of the nameservers in TLDs (Top Level Domains), as well as 23% of the nameservers in Let's Encrypt-certified domains and 17% of the nameservers in

857K-Alexa domains, are vulnerable to elimination via rate limiting attack. We plot the results for our dataset of domains in Figure 5.

3.4 Elimination via Low Rate Bursts

If the packets arrive at the router faster than they can be transmitted, they are buffered. Routers are configured for best-effort packet forwarding and typically the packets are processed using first come first served model. A packet loss in networks occurs due to queuing of packets in routers and overflowing routers' buffers.

3.4.1 Attack methodology. The idea is to cause packet loss on a router that connects the nameserver to the Internet, slightly before the arrival of the DNS request at the nameserver. We create loss by sending low-rate bursts to the router that connects the nameserver to the Internet. Targeting the router allows that attacker to avoid detection. To identify the target router the attacker runs traceroute to the nameservers. Nameserver elimination via low rate bursts is illustrated in Figure 2 (c). After requesting a certificate for the victim domain, step ④, the attacker sends a burst of packets to the target router at the estimated time that the request from the DNS resolver is sent to the nameserver in step ⑤ and ⑥. The burst causes the request to be discarded.

3.4.2 Synchronising the bursts with the queries. A crucial aspect of the accuracy of this methodology is to compute the exact time point when the burst should be sent. We measure the latencies between the attacker and the VA (Δ_{A-VA}) by pinging the services of Let's Encrypt, and the attacker and the target nameserver (Δ_{A-NS}) by querying the nameserver. We need to infer the processing delays at Let's Encrypt.

Inferring processing delay at Let's Encrypt. The time between the submission of the request with Certbot and the time when the queries from the VAs arrive to the nameserver of the attacker is: $\Delta_{A-VA-NS_A}$. Since this is the nameserver of the attacker, it holds: $\Delta_{A-VA} = \Delta_{VA-NS_A}$. Since the attacker knows Δ_{A-VA} and Δ_{VA-NS_A} , the attacker can estimate the processing delays incurred at Let's Encrypt: $t_{delay} = \Delta_{A-VA-NS_A} - \Delta_{A-VA} - \Delta_{VA-NS_A} = \epsilon$.

When to send the burst. Next, the attacker measures the latency to the nameserver in a target domain $2\Delta_{A-VA-NS}$. The time at which the attacker needs to send the burst is: $\Delta_{A-VA-NS} = \frac{2 \cdot \Delta_{A-VA-NS} - \epsilon}{2} = \Delta_{A-VA-NS} - \frac{\epsilon}{2}$, which the attacker can compute since it knows ϵ and $2 \cdot \Delta_{A-VA-NS}$.

Let $x = \Delta_{A-VA-NS} - \Delta_{A-NS}$. If $x < 0$, the attacker waits x ms and then sends the burst. Alternately, if $x > 0$ attacker sends the burst x ms at $\Delta_{A-VA-NS} - x$.

3.4.3 Measuring burst size. The burst size is a function of the buffer size on the router as well as communication from other sources that traverses the router. Since we carry out ethical experiments we do not send bursts to the routers in the Internet. Our evaluation is performed in a controlled environment on a platform that we set up, using default buffer sizes on popular routers. These measurements provide a worst-case analysis. In practice the other communication that goes through the router will keep the buffer on the router also occupied, which means that even a smaller burst can achieve a similar effect.

We compare the effectiveness of bursts when sent from one host, from two hosts and from three hosts. We also evaluate the impact of packet sizes on the loss rate.

Setup. Our setup is illustrated in Figure 6. For our experiment we set up a platform, with five end hosts, each on a different network, connected to the Internet via a router. One host is a DNS resolver that sends DNS requests, the other is a DNS nameserver. The three remaining hosts are used to generate traffic bursts. We set up a router which connects all the clients and servers. This router simulates the Internet and is connected with 100Gbps links, all the other devices are connected through 10Gbps output links. Since the transmission rates on the router, that simulates the Internet, are ten times higher than the transmission rates on the routers that connect the end devices (i.e., 100Gbps vs 10Gbps), it will not experience packet loss. This ensures that the only packet loss can occur on the routers that connect the end hosts to the Internet. All the routers are configured to add latency to every packet on outbound interface. The latency is selected at random in the range between 30ms and 50ms. This results in an RTT (round trip time) between 60ms and 100ms; similar to the typical RTT in the Internet. To add latency we use NetEm `tc qdisc`.

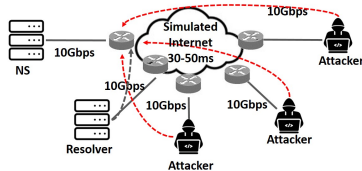


Figure 6: Simulated evaluation setup.

Experiments. We measure the optimal burst size that causes the arriving packets to be discarded. We also aim to infer the maximal sequence of packets that will be discarded after a given burst. What burst size and characteristics will result in the largest sequence of packets to be discarded. Our evaluations are performed using different buffer sizes, listed in Table 3. The timing of the attack bursts are illustrated in Figure 7.

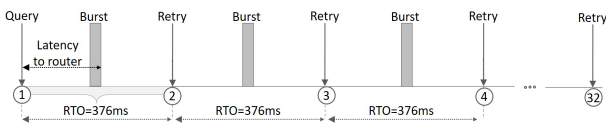


Figure 7: RTO timeline with low-rate bursts.

We test sending the burst from one vs two vs three hosts. Additionally, we create bursts using packets of: (1) identical size of 500 bytes, (2) randomly selected sizes between 68 and 1500 bytes, and (3) packets of two sizes 68 or 1500 bytes, both sizes are selected with equal probability.

The DNS resolver is sending a set of queries to the nameserver and the nameserver responds. To generate traffic bursts from the attacking hosts we use `iperf3`. During the experiment the three attacker hosts synchronise and send a burst of packets to the nameserver. The loss rate depends on the buffer sizes that are on the routers as well as the additional traffic from other sources. Since in our experiment there is no additional traffic from other clients,

our evaluation provides a lower bound. In practice in the Internet the burst would be much more effective due to traffic from other sources which also traverses the router.

Our experiment showed that the higher the latency variance is between the packets, the more overhead the burst introduces on the processing, resulting in higher loss ratio. We also find that (3) resulted in the largest sequence of packets dropped one after another, it is 7 times as large as the sequence of packets lost in experiments with bursts (1) and (2). Furthermore, bursts from multiple clients result in a packet loss more effectively. In fact our evaluations show that the load on the system and the period of time during which additionally arriving packets will be dropped is proportional to the number of attacking hosts that send the burst. Namely, the same burst volume split among multiple end hosts is more effective than when sent from a single client. This is due to the fact that when sent concurrently from different sources the inter-packet delay in a burst is reduced.

Routers	Buffer sizes	Burst size	Loss rate
Brocade MLXe	1MB	>1550 packets	100%
Cisco Nexus 3064X	9MB	>10 ⁴ packets	100%
Juniper EX4600	12MB	>15 · 10 ³ packets	92%
Cisco 6704	16MB	18 · 10 ³ packets	89%

Table 3: Burst evaluation on popular routers.

Our results are listed in Table 3. For effective packet loss the bursts can be even smaller in volume than the buffer size - packets are nevertheless discarded.

Buffer sizes. Typically routers with large buffers are used in the core of the Internet where cross traffic can cause large queues, but routers that connect networks to the Internet have smaller buffers, sufficient for a 10 Gbps traffic rates. The reason for avoiding large buffers is ‘bufferbloat’ which is too high latency that results due to network devices buffering too much data, leading to link under-utilisation. Typical buffer sizes is megabytes of buffer per 10Gbps port and for 10Gbps links, 10Mb of buffers, [11]. In our experiment we evaluate bursts on popular routers with default buffers’ sizes that are set by the vendors, these of course can be resized to smaller sizes by the operators. Our set of routers covers typical routers that connect networks to the Internet as well as large routers at the Internet core.

3.5 Applicability of Frag. & Rate-Limit

We find that 22.76% of the domains in our dataset are vulnerable to either fragmentation or rate-limiting nameserver-elimination methodologies, see Table 2. We also find that in 15% of the domains more than 50% of the nameservers enforce rate limiting or return fragmented responses, and hence are vulnerable to either elimination via rate limiting or via IP defragmentation cache poisoning; the results are plotted in Figure 4.

3.6 No False Positives, Some False Negatives

Our automated evaluation provides a lower bound on the number of vulnerable domains since it may miss out potentially vulnerable domains. This introduces *false negatives*, namely, domains which are vulnerable to our off-path server-elimination methodologies, but we have not detected this. The reason is that automated evaluation

is not sensitive to slightly different behaviours or implementations. For instance, by manually adjusting the IP ID value prediction (see more details in Appendix, Section E) the attacker has a higher success rate to hit the correct IP ID value. An automated evaluation may not predict the IP ID correctly, due to, say sudden change in outbound traffic rate from the nameserver. Similarly, nameservers employ different methodologies for limiting the rate of incoming queries, e.g., per query, or per source IP, or may simply require a slightly higher rate of incoming packets. Furthermore, our evaluation against each domain is performed once, to avoid interfering with the normal functionality of the domain. Our attacker host perceives any losses or noise as a failed evaluation, without repeating it again.

We do not have *false positives*. We only mark a successful downgrade attack in the event when all the VAs in our setup are querying a single nameserver which our attacker selected.

4 ATTACKS AGAINST LET'S ENCRYPT

In this section we combine the off-path downgrade attack with BGP same- and sub-prefix hijacks, to obtain fraudulent certificates of Let's Encrypt for victim domains. To launch our attacks against real Internet targets we develop an ethical 'two-sided' evaluation methodology. In Section 4.1 we introduce our experimental setup. Then, in Section 4.2, we launch our combined attack to issue fraudulent Let's Encrypt certificates for our own victim domains (which we registered for that purpose). This demonstrates the vulnerabilities in Let's Encrypt. Second, in Section 4.3, we evaluate our combined attack against our dataset of domains (Section 3.1). Since these are real domains, in order to evaluate the attack against them ethically we reproduce the exact setup of Let's Encrypt in a controlled experimental environment set up by us. We create our own CA, issue certificates for the domains in our dataset with our CAs, and then launch the combined attack to obtain fraudulent certificates signed by our CA for those domains. This enables us to identify domains with which the VAs of Let's Encrypt can be forced to query a server of attacker's choice, such that, that nameserver can either be same- or sub-prefix hijacked by the attacker; because it is hosted on a network block that can be sub-prefix hijacked, or because of a topological proximity between the attacker and the target nameserver. The hijacking BGP announcements are sent locally only to the router that connects the network with our CA to the Internet, and are not distributed in the Internet, to avoid impacting the global routing.

We extend our automated experimental evaluation in Section 3, and as a next step to the evaluations in 3, we also evaluate prefix hijack attacks of the nameservers that the VAs query, to obtain fraudulent certificates, signed by our CA for the victim domains.

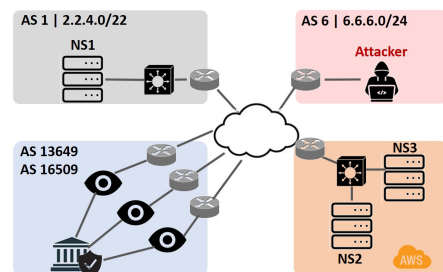
We then compare the security of Let's Encrypt to other popular CAs in the PKI ecosystem, in Section 4.4. We show that the downgrade attacks apply to all other CAs. Although, in contrast to Let's Encrypt, the other CAs do not guarantee security against MitM adversaries, our attack nevertheless makes it easier to attack them even for off-path adversaries.

In Appendix, Section G we show how to exploit fraudulent certificates signed by our own CA to launch attacks against Email servers and Web clients.

4.1 Control Setup

We prepare a control plane setup for experimental evaluation of all our attacks in this section. Our setup of the control plane with the relevant entities and components is illustrated in Figure 8. We purchase under RIPE NCC two ASes: AS 1 and AS 6. AS 1 is assigned prefix 2.2.4.0/22 and AS 6 is assigned 6.6.6.0/24⁸. AS 6 is the network controlled by our attacker, which we use for hijacking the prefix of the network on which the nameserver of our victim domain is installed. The victim domain has three nameservers, two nameservers, NS2 and NS3, are on AWS cloud and one nameserver NS1 is hosted on 2.2.4.0/22. We also set up an Unbound 1.6.7 DNS resolver on Linux 4.14.11 on 2.2.4.0/22.

From layer 3 point of view AS 6 is connected with a BGP router to DE-CIX routeserver in Frankfurt through which we have peering with many (mostly) small partners. AS 1 is connected via a different upstream provider to the Internet. We configured the BGP routers on both AS 1 and AS 6 as follows: the BGP router is a Dell running Ubuntu OS. The router is setup to handle 10Gbps traffic, the NICs are prepared for XDP (eXpress Data Path), which enables it to process tens of millions of packets per second per core with commodity hardware. We installed Bird 2 on both BGP servers since it is configurable and provides MRT files (BGP message dumps) that are easy to dump. We set up BGP sessions, such that the router for AS 1 announces 2.2.4.0/22 and the router for AS 6 is announcing the attacker's prefix 6.6.0.0/24. The Validation Authorities (VAs) of Let's Encrypt are located on different network prefixes assigned to two ASes: AS 16509 and AS 13649. Without the prefix hijack, the traffic from AS 1 flows to Let's Encrypt (AS 16509).



Issuing fraudulent Let's Encrypt certs for our victim domains.
Figure 8: Experimental setup.

4.2 Fraudulent Let's Encrypt Certificate for Our Victim Domain

In this section we launch attacks against Let's Encrypt using our victim domains.

4.2.1 Setup. We setup a victim domain with three nameservers: two nameservers NS2 and NS3 are on AWS cloud and one nameserver NS1 is hosted on 2.2.4.0/22, see Figure 8.

4.2.2 Attack. The attack proceeds in three steps. We illustrate the conceptual components of the attack in Figure 13 in Appendix, Section B. In step (A) the adversary applies methodologies in Section 3 to force all the VAs of Let's Encrypt to perform lookups and domain

⁸The network prefixes used in the paper are anonymised.

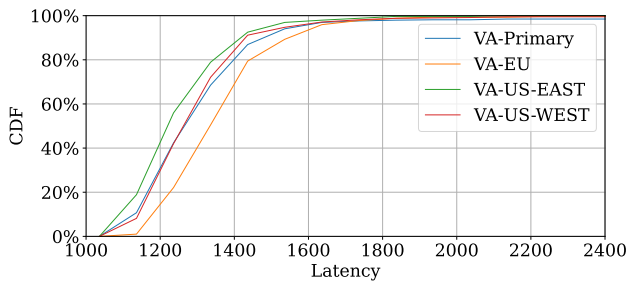


Figure 9: Obtaining fraudulent certs with each VA in Let's Encrypt (ms).

validation against a nameserver of its choice. In our evaluation we select NS 1, on prefix 2.2.0.0/16 (this is the network which we own and control). In second step (B) the adversary uses Certbot to submit request for a certificate for our victim domain. Notice that step (A) is also initiated with a certificate request for the victim domain. However, since Figure 13 illustrates logical steps of the attack, we omit this from the illustration; these steps are described in detail in Section 3. In step (C) the attacker launches BGP prefix hijacks to redirect the DNS packets to the attacker's network (AS 6). The attacker concludes the validation and receives a fraudulent certificate for our victim domain.

4.2.3 Evaluation. We ran multiple executions of the attack against our victim domains. Our plot of the duration of the attack in Figure 9 shows that in 99% of the evaluations the attack completes within 2 seconds. The plot measures the time from the issuance of a hijacking BGP announcement and until the fraudulent certificate is received. The attack, starting with a certificate request submission with Certbot (after the attacker eliminated the nameservers from the list of usable nameservers at the VAs) and until the certificate is received is automated. The duration of the attack is dominated by the propagation of the malicious BGP announcement and the convergence delays.

To understand the delays involved in propagation of BGP updates and routing convergence and their contribution to the overall attack duration, in addition to evaluations in the wild, we also evaluate convergence of BGP updates on common routing platforms in a controlled environment in Section 4.5.

4.2.4 Measurements. All the VAs of Let's Encrypt are located on prefixes smaller than /24, which makes them vulnerable to sub-prefix hijack attacks. We plot the CAs and the domains vulnerable to sub-prefix hijacks in Figure 10; 'LE VAs', in legend, refer to VAs of Let's Encrypt, 'Other CAs' refer to CAs we evaluated in Section 4.4, 'LE Domains' refer to Let's Encrypt-certified domains, and 'Alexa Domains' refer to our list of 857K-top Alexa domains. Sub-prefix hijacks succeed deterministically irrespective of the location of the attacker, however, since they affect the Internet globally they are more visible. Nevertheless, such attacks often stay under the radar over long time periods [2, 28]. Since our hijacks are short-lived, their risk of exposure is significantly reduced. For details on sub-prefix hijacks see Appendix, Section A.

4.3 Attacking Let's Encrypt-Certified Domains

In previous section we executed attacks against Let's Encrypt and issued fraudulent certificates for our own victim domains. During the evaluation we showed that off-path adversaries can bypass the validation of the multiVA of Let's Encrypt, by eliminating nameservers in a victim domain and forcing all the VAs to perform the lookup and validation against the attacker-selected nameserver. In this section we ask *do our attacks apply against customer domains of Let's Encrypt?* In particular, do the domains have the properties needed for our attacks?

To answer these questions we develop an automated attack to assess the attack surface of the domains that have certificates with Let's Encrypt. We execute our automated attack to perform the first large scale evaluation of the domains for which off-path adversaries can issue fraudulent certificates with Let's Encrypt using our attack methodologies. The challenge is, however, to develop and evaluate real attacks, yet ethically, without issuing fraudulent Let's Encrypt certificates for real customer domains of Let's Encrypt. To perform a realistic execution of our attacks yet consistent with the ethics we reproduce the deployment of Let's Encrypt using only the components that are relevant to validation and issuance of certificates. In that setup we configure DNS resolvers on the VAs which we control. We use these DNS resolvers to execute attacks against real domains in an ethical way. We explain the setup below.

4.3.1 Setup. On the three⁹ VAs we set up an Unbound 1.6.7 DNS resolver on Linux 4.14.11. The VAs are placed on three distinct prefixes, that belong to AS 1. We setup an open-source Boulder ACME (Automated Certificate Management Environment) implementation [13] used by Let's Encrypt. ACME is used by Let's Encrypt to automate certificate issuance and management. The components of Boulder relevant for our evaluation are Registration Authority (RA), Validation Authority (VA) and Certificate Authority (CA). During certificate issuance the client (we use Certbot [29]) submits a request for a certificate. The RA forwards the request to VAs. The VAs perform validation and return the result to the RA. If validation succeeds, RA requests the CA to sign the certificate. The RA returns either a failure, if validation did not succeed, or a signed certificate to the client that sent the request.

To simulate Let's Encrypt and issue our own certificates to real domains we need to set up a CA. We do this with `step-ca`¹⁰, which is an online CA supported by ACME. This enables us to use ACME APIs to issue certificates from our own private CA. To set up the ACME client we configure the URL and our root certificate. The certificate issuance is similar to Let's Encrypt: ACME client creates an account with ACME server, and uses Certbot to request a certificate. Our client uses Certbot to send a certificate request to the RA. The RA performs the domain validation using our three VAs.

4.3.2 Dataset. We search domains that have certificates of Let's Encrypt in CT (Certificate Transparency) with `crt.sh`¹¹, checking for CA commonName: R3. We only collect certificates issued in a single day, by limiting the search to `ValidityNotBefore >= 01.04.2021`

⁹Let's Encrypt uses one primary and three remote VAs and validation succeeds when correct responses are received at three VAs. Hence, in our evaluation of attacks three VAs reflect the success of the attacks against the setup of Let's Encrypt.

¹⁰<https://github.com/smallstep/certificates>

¹¹<https://crt.sh/>

00:00:00 and ValidityNotBefore < 02.04.2021 00:00:00. This resulted in 1,014,056 domains issued by Let’s Encrypt on a single day in April. We then extract the commonNames in the certificates and lookup the nameservers for each commonName. For each nameserver we map its IP address to the IP prefix and origin AS, using the BGP updates in BGPStream of CAIDA [20] on 1 April.

4.3.3 Attack. The adversary receives a list of domains with nameservers in an input. For each nameserver in each domain we include information to which attacks (from Section 3) the nameserver is vulnerable, the latency to each nameserver, and if the nameserver is vulnerable to sub-prefix hijack attacks.

For each domain the adversary executes the following attack steps: (1) submits a request for a certificate, (2) performs nameserver elimination against the nameservers in the victim domain, (3) hijack the DNS packet, (4) conclude DV, (5) obtain fraudulent certificate for a real victim domain signed by our CA.

4.3.4 Measurements of attack surface of vulnerable domains. To obtain insights about the sizes of the announced BGP prefixes in the Internet we use the BGPStream of CAIDA [20] and retrieve the BGP updates and the routing data from the global BGP routing table from RIPE RIS [53] and the RouteViews collectors [58]. The dataset used for the analysis of vulnerable sub-prefixes was collected by us in April 2021.

There are currently 911,916 announced prefixes in the Internet. From these prefixes we extracted all the announcements with prefixes of ASes which host nameservers of the domains in our dataset (Table 2). Then, we take the domains that we found vulnerable to frag. and rate-limit server elimination attacks (Table 2 column “#Vuln.”), and check which domains have nameservers that are on network blocks smaller than /24 or networks which are topologically closer to the attacker than to the VAs of Let’s Encrypt. The former set contains domains on networks that can be hijacked via a more specific BGP announcement which makes them vulnerable to sub-prefix hijacks. We obtain 10.6% of the Let’s Encrypt certified domains and 11.75% Alexa domains. Namely, against these domains our off-path attacker can force Let’s Encrypt to query a nameserver of its choice, which can be sub-prefix hijacked since it is on a network block less than /24, see Figure 10; the legend is explained in Section 4.2.4.

The latter contains domains on networks which can be intercepted via same-prefix BGP hijacks. We find that our attacker can intercept the prefixes from above 30% of the ASes with victim domains, causing the network with the VAs in our setup to accept the hijacking BGP announcements of the attacker and as a result send DNS packets through the attacker.

4.4 Comparison to Other Popular CAs

We evaluated our attack methodologies also with other CAs that control more than 95% of the certificates market, listed in Table 4. Our evaluation was performed against the popular 857K-top Alexa domains. The results are listed in Table 4. For success, only Let’s Encrypt requires that multiple vantage points receive the same responses. In contrast, other CAs, even when selecting an IP address from a large prefix, such as Certum-Google, perform the validation with a single IP address.

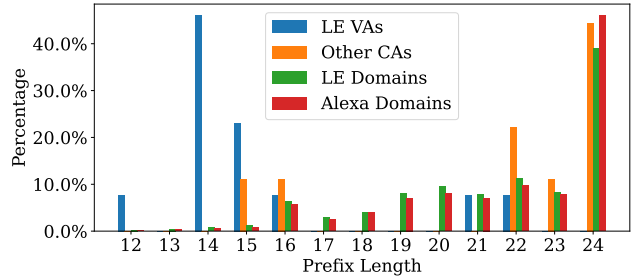


Figure 10: Network prefixes of CAs' resolvers and of domains' name-servers vulnerable to sub-prefix hijacks.

CA	#Vantage Points	Sub-prefix attack	#Time outs	Block (min)	MultiVA
Digicert	1	✗	1	5	✗
Sectigo	1	✗	2	10+	✗
GoDaddy	1	✓	10	10+	✗
GlobalSign	1	✓	4	10+	✗
Certum-Google	20+	✓	2	10+	✗
Certum-Cloudflare	1	✗	16	10+	✗
Let's Encrypt	4	✓	2	15	✓
Actalis	1	✓	2	10+	✗

Table 4: Infrastructure of popular CAs and our evaluations.

All our blocking methodologies apply to other CAs as well. We only needed to apply slight modifications according to the behaviour of the DNS software at each CA. For instance, the number of the required timeout differs (column ‘#Timeouts’), as well as the length of the blocking interval (column ‘Block’), during which the DNS software avoids querying the blocked nameserver. We conclude that all the CAs are vulnerable to nameserver-elimination, which exposes them to extremely effective off-path attacks.

Similarly to the analysis in Section 4.3.4, we obtain that 11.75% of the 857K-Alexa domains, for which the CAs (in Table 4) can be forced to query a specific nameserver, that is on a network vulnerable to sub-prefix hijack.

4.5 How Fast is Short-Lived Hijack?

The goal of our attacker is to do a short-lived hijack to avoid detection. It is believed that short-lived traffic shifts are caused by the configuration errors (that are quickly caught and fixed) and since they do not have impact on network load or connectivity, they are largely ignored [17, 37, 38].

There are a number of factors which contribute to the overall latency of the attack, nevertheless, how fast the attacker can carry out such attack depends predominantly on the speed at which the malicious BGP announcements propagate to the forwarding plane at the target victim AS. The main role in this latency play the updates of the received BGP announcements at the Autonomous System Boundary Router (ASBR), which is used to exchange routing information with the ASes. There is a rule that a BGP announcement should be delayed until the local route is installed in BGP Forwarding Table (FIB), so the announcement is not causing temporary blackholing. For instance, if the receiving router is much faster to pick up on the new announcement and starts sending the traffic before the FIB is fully converged. That is the main factor of the propagation delay and it is a delay introduced by a BGP

timer, which limits the rate at which routing announcements are transmitted.

In this experiment we evaluate the ability to process route changes on popular routing platforms, in response to a new BGP announcement, measuring the time it takes for the new route to be installed and used in the FIB on the line card. During the experiment we populate the routing and forwarding tables with data, generate and send BGP announcements and initiate the measurements. We measure convergence delay, i.e., the latency between the time that the BGP announcement is sent by the originator until the propagation of the information into the forwarding plane at ASBR with each box. In our evaluations we send BGP announcements with 300,000 IPv4 prefixes, hence measuring convergence between control and data plane at high load. The resulting latency is in the order of tens of seconds, and is device dependent. The slowest was CISCO 7600 platform with above 30 seconds and was representative among old platforms we tested. Newer platforms, such as Cisco ASR 9901 and Juniper MX204, are much faster with overall time being between 5 and 10 seconds, even when routers have multiple full BGP feeds. The fastest results were obtained with Arista 7280R3, which had an almost instant propagation time of a second. Our findings show that the convergence delay depends on CPU power and efficiency of the implementation and our evaluations results demonstrate high variance (between one second and tens of seconds). For instance, the Cisco 7600 platform tested by us was released in 2000s with control-plane module Sup720 with just 2x 600MHz MIPS CPUs. Modern platforms, like Junipers MX204, have 8-core Xeon-D @2.2GHz and they run soft-real time Linux kernel and on top of that are two virtualised instances of JunOS control-plane in QEMU/KVM, resulting in much faster processing times, not only because the box has more CPU power but also because of the more efficient software stack. The evaluation results are summarised in Table 5.

Year	BGP router	FIB Convergence
2001	CISCO 7600	>30sec
2005	CISCO IOS XR 9000	500ms
2006	Juniper MX204	5-10sec
2008	CISCO ASR 9001	5-10sec
2009	Alcatel Lucent 7750SR	3sec
2010	Arista 7280R3	1sec

Table 5: BGP convergence on popular routing platforms.

5 COUNTERMEASURES

Unpredictable VAs selection. Our attacks used the fact that the adversary knows in advance the set of VAs that perform the validation: the network blocks of the VAs are publicly known. The network blocks and the set of the VAs is small. This allowed our downgrade attack to be carried out against each VA, forcing the VAs to query attacker-selected nameserver for the next 15 minutes. This provides the adversary sufficient time to obtain fraudulent certificates. If the VAs are selected from a large set of network blocks at random, such that the adversary cannot efficiently attack them in advance, the downgrade attack would be much more challenging to launch in practice. This would enhance the security of DV with multiVA even against MitM adversaries.

Resilient nameserver-selection. The nameserver selection algorithms of the CAs should be made robust by selecting the

nameservers uniformly at random, even those with poor performance. If a nameserver does not respond, the query is resent to another nameserver after a timeout. This would prevent our off-path server-elimination methodologies. The additional latency to the certification would not be significant.

Turning off caches. Turning off caches does not prevent the cache poisoning attack [35] but makes it more difficult to launch. Caches allow to inject a malicious mapping between the victim domain and the IP address of the attacker, which is subsequently used for running domain validation against the target domain. Although Let's Encrypt limits the caching duration to 60 seconds, it still suffices for attacking the lookup phase to redirect to attacker's hosts. The validation is then run against the hosts of the attacker. The entire attack concludes within two minutes. If there are no caches the attacker has to keep the hijacked prefixes over longer time periods, which may make the attack more visible.

DNSSEC against domain hijacks. DNSSEC [RFC4033-4035] could prevent the attacks, however, recent works showed that more than 35% of signed domains are vulnerable to key recovery attacks [21, 56].

Preventing BGP hijacks with RPKI. If fully deployed RPKI [RFC6480] would prevent prefix hijack attacks. Our measurements show that most networks do not filter hijacking BGP announcements with Route Origin Validation (ROV). The ASes of Let's Encrypt do not apply ROV, hence even if the domains have a valid ROA, it does not prevent the hijacks. In addition, only 86 out of 17,864 ASes on our dataset of domains (2M Alexa and Let's Encrypt domains, Table 2) apply ROV. Worse, 57% ASes have ROAs, out of which 32.4% ROAs are invalid, and hence can be hijacked. A full adoption of RPKI (both the prefix certificates with ROAs and validation with ROV) although would not prevent all the possible attacks against DV, it would prevent the prefix hijack attacks.

Detecting Fraudulent Certificates with CT. A Certificate Transparency (CT) log [42] could expose a fraudulent certificate and allow a CA to quickly revoke it. We measured the rate at which our fraudulent certificates with Let's Encrypt appear in the logs of CT monitors. We registered with the notification third party services which continuously monitor CT logs and notify via email when a certificate for required domain is issued. We also registered with search services which provide API for retrieving logged certificates by domain name.

We observed that it took some monitors a few hours to fetch our fraudulent certificates. Furthermore, some of the monitors exhibited failures and did not detect the fraudulent certificates. Our results are aligned with the recent study which found that the monitors provide unreliable service [45]. The damage of our attack would have been done by the time the attack is detected. Indeed, the damage of such attacks is the highest in the first hour, e.g., [54].

6 RELATED WORK

Domain validation. Domain validation plays a central role in the PKI ecosystem and in Internet security. Flaws in DV can be exploited to obtain fraudulent certificates. Some CAs were shown to use buggy domain validation, e.g., to establish control over a domain WoSign¹² tested control of any TCP port at the target

¹²https://wiki.mozilla.org/CA:WoSign_Issues#Issue_L:_Any_Port_.28Jan_-Apr_.2015.29

domain, in contrast to requiring control over services, such as Email, HTTP, or TLS. In other cases, CAs were validating control over domains by sending email verification to addresses belonging to ordinary users instead of domain administrators¹³. There are also design specific flaws, which were exploited to bypass DV and issue fraudulent certificates [14, 18]. Following these attacks Let's Encrypt standardised domain validation [RFC8555] and deployed a production grade validation with multiVA. Followup works [9, 15, 36] on Let's Encrypt evaluated performance and demonstrated security of validation from multiple locations with multiVA.

Distributed domain validation. Distributed validation is a known concept. In 2004 [50] proposed CoDNS to improve the availability and performance of DNS lookups. ConfiDNS [52] extends CoDNS with peer agreements and majority vote. Perspectives' [60] verifies a server's identity by using a new infrastructure of notary servers. DoubleCheck [10] aims to prevent attacks against clients that retrieve a certificate of the target service for the first time. However, in contrast to multiVA of Let's Encrypt these proposals are not deployed due to the modifications required to the existing infrastructure and the lack of specific use cases motivating adoption. We explore the security of multiVA since it is deployed by one of the largest and rapidly growing CAs.

BGP prefix hijacks. [23, 24, 57] evaluated applicability of BGP prefix hijacks against different applications in the Internet. There is numerous evidence of DNS cache poisoning attempts in the wild [1–7, 22, 54], which are predominantly launched via short-lived BGP prefix hijacks or by compromising a registrar or a nameserver of the domain. In this work we apply BGP prefix hijacks for intercepting the DNS communication between the VA and the nameserver, in order to create a spoofed DNS response with records that map the nameservers in the victim domain to attacker's IP addresses. Once cached, these records poison the caches of the DNS resolvers at the VAs. Notice that there are also other attacks on BGP, such as poisoning AS paths [16], defences against these are path security with BGPsec proposals, [12, 44], none of which are deployed. In our work we show that even merely applying origin hijacking already leads to devastating attacks against a large fraction of Internet domains and their clients, and our evaluations show that the attacks are a practical threat.

Countermeasures against BGP hijacks. To mitigate prefix hijacks, the IETF designed and standardised Resource Public Key Infrastructure (RPKI) [RFC6810] [19]. RPKI uses Route Origin Authorizations (ROAs) to bind Autonomous Systems (ASes) to the network prefixes that they own via cryptographic signatures. In order for this binding to deliver on its security promise, the ownership over prefixes has to be correctly validated and configured in the resource certificates (RCs) and Route Origin Authorizations (ROAs), which are then placed in global RPKI repositories managed by five Regional Internet Registries (RIRs). These ROAs are fetched and validated, e.g., using the implementation of RIPE NCC validator [51]. The RPKI validator fetches the ROAs from the global RPKI repositories and applies Route Origin Validation (ROV) to create a local validated cache. This cache is then provided to a BGP-speaking routers via the RPKI to Router (RTR) protocol. Recent research [33, 40] showed that about 600 networks apply ROV. In

our measurements of the ASes with domains in our dataset, we find that very few ASes apply ROV, only 86 out of 17,864!

There are also proposals for detection of hijacks based on changes in the origin, [41], which is not yet in use in the Internet. SCION [62] proposes to replace BGP with a new routing architecture and is already deployed in production of a number of ISPs, but is not used by the vast majority of the Internet and none of the ASes which host the domains in our datasets or the CAs.

7 CONCLUSION

Domain validation is essential for bootstrapping cryptography on the Internet. After validating control over a domain, a CA generates a certificate which can be used to establish cryptographic material and protect communication between the clients and the corresponding server. In contrast to other means for verifying control over a domain, domain validation is automated, and hence is fast and cheap (or even free, e.g., as in the case of Let's Encrypt). These benefits are the reason why the CAs that offer domain validation collectively control more than 99% of the certificates market.

Unfortunately, the benefits of domain validation are coupled with insecurity. The attacks in 2018 [14, 18] showed that the domain validation used by many CAs was vulnerable. Let's Encrypt was the first CA to deploy in production mode validation from multiple vantage points, to provide security even against strong MitM adversaries [36]. Followup security analysis and simulations showed that MitM adversaries cannot attack multiple VAs of Let's Encrypt concurrently [15].

In this work we developed off-path downgrade attacks to reduce the domain validation to be performed against a single, attacker-selected nameserver. The experimental evaluation that we carried out found Let's Encrypt vulnerable to our downgrade attack. After forcing the VAs of Let's Encrypt to query a single nameserver that resides on a network vulnerable to sub-prefix hijacks we carried out ethical attacks and successfully issued fraudulent certificates for 10.60% of the domains in our dataset. They demonstrate that Let's Encrypt is not only insecure against MitM adversaries but also against off-path adversaries.

We showed that other CAs were also vulnerable to downgrade attacks, and off-path attackers can launch efficient and effective attacks to obtain fraudulent certificates with them.

Our work demonstrates that domain validation, although seemingly simple, is not a resolved problem. An interesting and an important question that we leave for future research is under what conditions and assumptions (on topology, adversary capabilities, etc.) can domain validation be made secure.

ACKNOWLEDGEMENTS

We are grateful to Jennifer Rexford for her helpful comments on our work. This work has been co-funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) SFB 1119.

¹³https://bugzilla.mozilla.org/show_bug.cgi?id=556468

REFERENCES

- [1] 2015. Hacked or Spoofed: Digging into the Malaysia Airlines Website Incident. <https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/hacked-or-spoofed-digging-into-the-malaysia-airlines-website-compromise>. Accessed: 2021-1-19.
- [2] 2015. Webnic Registrar Blamed for Hijack of Lenovo, Google Domains. <https://krebsonsecurity.com/2015/02/webnic-registrar-blamed-for-hijack-of-lenovo-google-domains/>. Accessed: 2021-1-19.
- [3] 2018. DNSspionage Campaign Targets Middle East. <https://blog.talosintelligence.com/2018/11/dnsponage-campaign-targets-middle-east.html>. Accessed: 2021-01-19.
- [4] 2019. Global DNS Hijacking Campaign: DNS Record Manipulation at Scale. <https://www.fireeye.com/blog/threat-research/2019/01/global-dns-hijacking-campaign-dns-record-manipulation-at-scale.html>. Accessed: 2021-1-19.
- [5] 2019. Sea Turtle keeps on swimming, finds new victims, DNS hijacking techniques. <https://blog.talosintelligence.com/2019/07/sea-turtle-keeps-on-swimming.html>. Accessed: 2021-01-19.
- [6] 2019. "Unprecedented" DNS Hijacking Attacks Linked to Iran. <https://threatpost.com/unprecedented-dns-hijacking-attacks-linked-to-iran/140737/>
- [7] 2020. Security Incident on November 13, 2020. <https://blog.liquid.com/security-incident-november-13-2020>. Accessed: 2021-01-19.
- [8] Louis Poinsignon. 2018. BGP leaks and cryptocurrencies. <https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies/>
- [9] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, et al. 2019. Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2473–2487.
- [10] Mansoor Alicherry and Angelos D Keromytis. 2009. Doublecheck: Multi-path verification against man-in-the-middle attacks. In *2009 IEEE Symposium on Computers and Communications*. IEEE, 557–563.
- [11] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. 2004. Sizing router buffers. *ACM SIGCOMM Computer Communication Review* 34, 4 (2004), 281–292.
- [12] Rob Austein, Steven Bellovin, Russ Housley, Stephen Kent, Warren Kumari, Doug Montgomery, Chris Morrow, Sandy Murphy, Keyur Patel, John Scudder, et al. 2017. RFC 8205-BGPsec Protocol Specification. (2017).
- [13] R Barnes, J Hoffman-Andrews, D McCarney, and J Kasten. [n.d.]. RFC 8555: Automatic Certificate Management Environment (ACME), Mar. 2019. *Proposed Standard* ([n. d.]).
- [14] Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. 2018. Bamboozling certificate authorities with {BGP}. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 833–849.
- [15] Henry Birge-Lee, Liang Wang, Daniel McCarney, Roland Shoemaker, Jennifer Rexford, and Prateek Mittal. 2021. Experiences Deploying Multi-Vantage-Point Domain Validation at Let's Encrypt. *USENIX Security* (December 2021).
- [16] Henry Birge-Lee, Liang Wang, Jennifer Rexford, and Prateek Mittal. 2019. Sico: Surgical interception attacks by manipulating bgp communities. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 431–448.
- [17] Peter Boothe, James Hiebert, and Randy Bush. 2006. Short-lived prefix hijacking on the Internet. NANOG.
- [18] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. 2018. Domain Validation++ For MitM-Resilient PKI. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2060–2076.
- [19] Randy Bush and Rob Austein. 2013. The resource public key infrastructure (RPKI) to router protocol.
- [20] CAIDA. 2021. BGP Stream. <https://bgpstream.caida.org/>
- [21] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. A Longitudinal, End-to-End View of the {DNSSEC} Ecosystem. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 1307–1322.
- [22] D. Madory. 2018. Recent Routing Incidents: Using BGP to Hijack DNS and more. https://www.lacnic.net/innovaportal/file/3207/1/dougmadory_lacnic_30_rosario.pdf
- [23] Tianxiang Dai, Philipp Jeitner, Haya Shulman, and Michael Waidner. 2021. From IP to transport and beyond: cross-layer attacks against applications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 836–849.
- [24] Tianxiang Dai, Philipp Jeitner, Haya Shulman, and Michael Waidner. 2021. The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*. 3147–3164.
- [25] Joao Damas, Michael Graff, and Paul Vixie. 2013. Extension mechanisms for DNS (EDNS (0)). *IETF RFC6891, April* (2013).
- [26] Supratim Deb, Anand Srinivasan, and Sreenivasa Kuppili Pavan. 2008. An improved DNS server selection algorithm for faster lookups. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE, 288–295.
- [27] Chris C Demchak and Yuval Shavitt. 2018. China's Maxim-Leave No Access Point Unexploited: The Hidden Story of China Telecom's BGP Hijacking. *Military Cyber Affairs* 3, 1 (2018), 7.
- [28] Frank Denis. 2013. The GOOGLE.RW Hijack. <http://labs.umbrella.com/2013/10/25/google-rw-hijack-nobody-else-noticed/>.
- [29] EFF, the Electronic Frontier Foundation. [n.d.]. Certbot. <https://certbot.eff.org/>
- [30] Xuewei Feng, Chuanpu Fu, Qi Li, Kun Sun, and Ke Xu. 2020. Off-Path TCP Exploits of the Mixed IPID Assignment. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1323–1335.
- [31] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2016. Rowhammer.js: A remote software-induced fault attack in javascript. In *International conference on detection of intrusions and malware, and vulnerability assessment*. Springer, 300–321.
- [32] Amir Herzberg and Haya Shulman. 2013. Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In *IEEE CNS 2013. The Conference on Communications and Network Security, Washington, D.C., U.S. IEEE*.
- [33] Tomas Hlavacek, Amir Herzberg, Haya Shulman, and Michael Waidner. 2018. Practical experience: Methodologies for measuring route origin validation. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 634–641.
- [34] A Hubert and R Van Mook. 2009. Measures for making DNS more resilient against forged answers. In *RFC 5452*. RFC.
- [35] Philipp Jeitner and Haya Shulman. 2021. Injection Attacks Reloaded: Tunnelling Malicious Payloads over DNS. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*. 3165–3182.
- [36] Josh Aas and Daniel McCarney and Roland Shoemaker. 2020. Multi-Perspective Validation Improves Domain Validation Security. <https://letsencrypt.org/2020/02/19/multi-perspective-validation.html>
- [37] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. 2008. Autonomous security for autonomous systems. *Computer Networks* 52, 15 (2008), 2908–2923.
- [38] Varun Khare, Qing Ju, and Beichuan Zhang. 2012. Concurrent prefix hijacks: Occurrence and impacts. In *Proceedings of the 2012 Internet Measurement Conference*. 29–36.
- [39] Amit Klein and Benny Pinkas. 2019. From {IP} {ID} to Device {ID} and {KASLR} Bypass. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1063–1080.
- [40] John Kristoff, Randy Bush, Chris Kanich, George Michaelson, Amreesh Phokeer, Thomas C Schmidt, and Matthias Wählisch. 2020. On Measuring RPKI Relying Parties. In *Proceedings of the ACM Internet Measurement Conference*. 484–491.
- [41] Mohit Lad, Daniel Massey, Dan Pei, Yiguo Wu, Beichuan Zhang, and Lixia Zhang. 2006. PHAS: A Prefix Hijack Alert System.. In *USENIX Security symposium*, Vol. 1. 3.
- [42] Ben Laurie. 2014. Certificate transparency. *Commun. ACM* 57, 10 (2014), 40–46.
- [43] Francois Le Faucheur, A Vedrenne, P Merckx, and T Telkamp. 2004. Use of Interior Gateway Protocol (IGP) metric as a second MPLS traffic engineering metric. *IETF Request for Comments RFC3785* (2004).
- [44] Matt Lepinski and Kotikalapudi Sriram. 2017. BGPSEC protocol specification. *Internet Engineering Task Force (IETF)* (2017).
- [45] Bingyu Li, Jingqiang Lin, Fengjun Li, Qiongxiao Wang, Qi Li, Jiwu Jing, and Congli Wang. 2019. Certificate Transparency in the wild: Exploring the reliability of monitors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2505–2520.
- [46] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. 2020. DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (Virtual Event, USA) (CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1337–1350. <https://doi.org/10.1145/3372297.3417280>
- [47] D McPherson, V Gill, D Walton, and A Retana. 2002. RFC3345: Border Gateway Protocol (BGP) Persistent Route Oscillation Condition.
- [48] Lucas Noack and Tobias Reichert. 2018. Exploiting Speculative Execution (SpecRet) via JavaScript. *Advanced Microkernel Operating Systems* (2018), 11.
- [49] Yossef Oren, Vasileios P Kemerlis, Simha Sethumadhavan, and Angelos D Keromytis. 2015. The spy in the sandbox: Practical cache attacks in javascript and their implications. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1406–1418.
- [50] KyoungSoo Park, Vivek S Pai, Larry L Peterson, and Zhe Wang. 2004. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. In *OSDI*, Vol. 4. 14–14.
- [51] Tashi Phuntsho. 2019. How to Install an RPKI Validator. https://labs.ripe.net/Members/tashi_phuntsho_3/how-to-install-an-rpki-validator
- [52] Lindsey Poole and Vivek S Pai. 2006. ConfIDNS: Leveraging Scale and History to Improve DNS Security.. In *WORLDS*.
- [53] RIPE NCC. 2021. RIS Raw Data. <https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/raw-data>
- [54] S. Goldberg. 2018. The myetherwallet.com hijack and why it's risky to hold cryptocurrency in a webapp. <https://medium.com/@goldbe/the-myetherwallet-com>

hijack-and-why-its-risky-to-hold-cryptocurrency-in-a-webapp-261131fad278

[55] Haya Shulman and Michael Waidner. 2014. Fragmentation considered leaking: port inference for dns poisoning. In *International Conference on Applied Cryptography and Network Security*. Springer, 531–548.

[56] Haya Shulman and Michael Waidner. 2017. One key to sign them all considered vulnerable: Evaluation of DNSSEC in the internet. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 131–144.

[57] Yixin Sun, Maria Apostolaki, Henry Birge-Lee, Laurent Vanbever, Jennifer Rexford, Mung Chiang, and Prateek Mittal. 2020. Securing Internet Applications from Routing Attacks. *arXiv preprint arXiv:2004.09063* (2020).

[58] University of Oregon Route Views Project. 2021. Route Views Project. <http://www.routeviews.org/routeviews/>

[59] Zheng Wang, Xin Wang, and Xiaodong Lee. 2010. Analyzing BIND DNS server selection algorithm. *International Journal of Innovative Computing, Information and Control* 6, 11 (2010), 5131–5142.

[60] D Wendlandt, D Andersen, and A Perrigo Perspectives. 2008. Improving SSH-style Host Authentication with Multi-path Network Probing. In *USENIX Annual Technical Conference*.

[61] Yingdi Yu, Duane Wessels, Matt Larson, and Lixia Zhang. 2012. Authority server selection in DNS caching resolvers. *ACM SIGCOMM Computer Communication Review* 42, 2 (2012), 80–86.

[62] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G Andersen. 2011. SCION: Scalability, control, and isolation on next-generation networks. In *2011 IEEE Symposium on Security and Privacy*. IEEE, 212–227.

A HIJACKING DOMAIN VALIDATION

In this section we show BGP prefix hijack attacks against a single VA of Let’s Encrypt that was used until 2020. We start by explaining and evaluating same-prefix hijacks and sub-prefix hijacks using a victim domain that we setup for this purpose. We evaluate attacks and their effectiveness and efficiency on popular router boxes. We explain why the attacks become challenging against the multiVA DV (domain validation) of Let’s Encrypt.

A.1 Setup and Attacker Model

A.1.1 Attacker Model. Most attackers are not located on the path between the nameservers of a victim domain and the VAs of Let’s Encrypt but are off-path to the victims that they wish to attack. BGP prefix hijacks enable off-path attackers to become on-path for the communication exchanged with the hijacked prefix. In this section we show how to apply BGP prefix hijacks for intercepting the communication between the nameservers and the VAs when the attacker is off-path and is not located on the communication path. Our BGP prefix hijacks are *short-lived* and are aimed at hijacking the DNS packets exchanged between the DNS resolver software at the VAs and the nameservers of victim domains. Short-lived BGP hijacks are common attack in the Internet [8, 22, 27].

Our attacker model is the same as the one against which Let’s Encrypt guarantees security [15]. This is also the attacker that was used in [14] to demonstrate insecurity of a single node DV. The attacker controls a BGP router and issues BGP announcements hijacking the same-prefix or a sub-prefix of a victim AS in the Internet. We show how the attacker can perform same-prefix and sub-prefix BGP hijacks of the VAs and the nameservers, explain the differences, and the implications of both these hijacks on the traffic that is intercepted during the interaction with Let’s Encrypt.

The attacker sets up a malicious DNS nameserver with a zonefile, that corresponds to the resources in the victim domain all mapped to the IP addresses of the attacker.

The victim AS is either a network hosting one or more of the nameservers of the target domain or the network hosting the VA. The target domain is the domain for which the attacker wishes to

issue a fraudulent certificate. The target domain has nameservers which serve records from the DNS zone of that domain. The goal is to hijack the same-prefix or a sub-prefix either of one or more nameservers of the target domain or to hijack the VAs. If the hijack succeeds, the attacker will receive all the traffic destined to the victim AS. To avoid blackholing the attacker should relay the traffic to the destination. In our attack against Let’s Encrypt the goal of the attacker is to intercept the DNS queries sent by the VAs, or the DNS responses sent by the nameservers. The attacker configures a filter for matching the IP addresses of the hijacked AS, in order to catch the target DNS packet. The attacker forwards all the remaining traffic to the legitimate destination.

We next explain how our attacker launches sub-prefix hijacks in Figure 12 and same-prefix hijacks in Figure 11 and how it impacts the traffic flow.

A.2 Same-Prefix vs. Sub-Prefix BGP Hijack

A.2.1 Same-prefix BGP hijack. The attacker advertises the same prefix as the victim AS and as a result can intercept traffic from all the ASes that have less hops (shorter AS-PATH) to the attacker than to the victim AS. Example same-prefix hijack attacks, for intercepting a DNS request or a DNS response is illustrated in Figure 11.

How effective are same-prefix attacks? The limitation of the same-prefix hijack is that it only affects the traffic of the ASes that prefer the attacker’s announcement, and does not propagate to all parts of the Internet. Hence, the effectiveness of the same-prefix hijack attacks depends on the local preferences of the ASes and the location of the attacker’s AS. In particular, the same-prefix attack only attracts traffic from ASes that have shorter path (i.e., less hops) to the attacker. Namely, the closer the attacker is to the victim (i.e., the shorter the AS-PATH is), the more effective the attack is. Hence the success of our hijack attack against the multiVA based DV of Let’s Encrypt depends on the topological relationship between the attacking AS, the target domain and the victim resolver. If the AS prefers the path announced by the attacker to the nameserver, then the hijack succeeds.

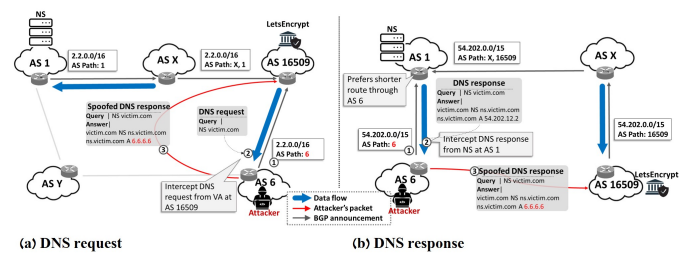


Figure 11: Same-prefix hijack: (a) Request and (b) Response.

A.2.2 Sub-prefix BGP hijack. The attack is illustrated in Figure 12 (a). The attacker can advertise a subprefix 2.2.2.0/24 of the victim AS 1. The routers prefer more specific IP prefixes over less specific ones, hence the longest-matching prefix (/24) gets chosen over the less-specific prefix (/16). Nevertheless, the adversary cannot advertise arbitrary long prefixes, e.g., (/32), since BGP routers typically discard prefixes which are more specific than 24 bits to reduce the

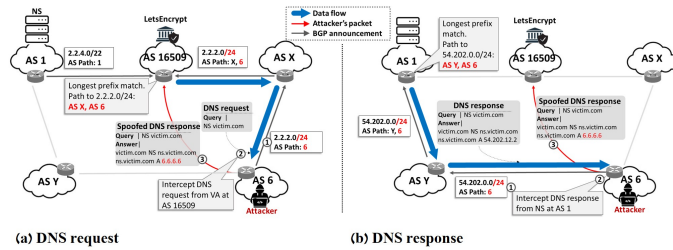


Figure 12: Sub-prefix hijack: (a) Request and (b) Response.

size of the internal routing tables. Therefore, only prefixes with less than 24 bits are vulnerable to sub-prefix hijacks. Once an AS accepts the hijacking announcement it sends all the traffic for that sub-prefix to the attacker.

How effective are sub-prefix attacks? Sub-prefix attack is highly effective since in contrast to same-prefix hijacks, all the traffic from any Internet AS globally is sent to the attacker, irrespective of the location of the attacking AS. To support these huge traffic volumes the attacker needs to set up a large infrastructure to relay traffic to the real destination ASes in the Internet. Otherwise, the attack will result in a blackhole and the attacker risks detection.

The effectiveness and applicability of the attack depends on the victim prefix size, a subset of which the attacker wishes to hijack. Our measurement evaluations of the networks of the VAs and of the nameservers showing vulnerabilities to sub-prefix hijack attacks are in Figure 10.

A.3 DNS Response vs. Request Interception

Our attacker makes malicious BGP announcements for a same-prefix or a sub-prefix containing the victim domain, for intercepting the DNS request sent by the CA, or containing the prefix of the victim resolver for intercepting the DNS response sent by the nameserver in the domain. In Section A.5 we explain that hijacking one direction, say communication sent from the CA to the domain, does not imply hijacking the other direction, since the routing paths in the Internet are asymmetric. In our attack, it suffices to hijack either the requests or the responses.

A.3.1 DNS request interception. The same-prefix hijack attack for intercepting a DNS request is illustrated in Figure 11 (a). The victim network announces its prefix 2.2.0.0/16. In step ① the attacker starts by originating a malicious BGP announcement which maps prefix 2.2.0.0/16 to AS 6. We wait between 1 to 3 minutes for the announcement to propagate; see our evaluation on the convergence duration in Section 4.5. When the announcement reaches AS 16509 its border router applies preferences to decide if to accept the announcement. In our example illustration in Figure 11 since AS 16509 has less hops to 2.2.0.0/16 through AS 6 than through AS 1, it decides to route the packets for IP addresses in prefix 2.2.0.0/16 to AS 6. To avoid blackholing the attacker sets up forwarding to relay all the packets to AS 1. Our attacker configures rules to intercept DNS packets sent to port 53 to an IP address in block 2.2.0.0/16 (i.e., DNS requests). Once the attacker captures the target DNS request, in step ②, Figure 11), it creates a corresponding DNS response with the malicious DNS records that map the nameservers in the victim

domain to the IP addresses controlled by the attacker. In step ③ the attacker sends the DNS response to the VA from a spoofed IP address (of one of the nameservers in the victim domain).

When launching a sub-prefix hijack attack, Figure 12 (a), the difference is that the attacker announces a more specific prefix of the nameservers of the victim domain than the victim AS.

Successful cache poisoning occurs once a DNS resolver at the VA accepts and caches the malicious records from the spoofed DNS response. The VA with the poisoned DNS resolver performs the domain validation against the hosts controlled by the attacker. In addition, all the subsequent DNS records will be queried from the hosts controlled by the attacker, including the services (e.g., HTTP, Email) against which the domain validation is performed.

A.3.2 DNS response interception. In a symmetric attack, illustrated in Figure 11 (b) in order to intercept the DNS response sent by the nameserver the attacker hijacks the traffic sent by the nameserver to the VA. In step ① the attacker announces the prefix 54.202.0.0/15 on which the VA is hosted. ASes that are closer to the attacker than to AS 16509 start routing the traffic for IP addresses in prefix 54.202.0.0/15 to AS 6. In Figure 11 (b) this includes AS 1 where the nameserver is hosted. The attacker configures forwarding rules, to relay all the traffic to 54.202.0.0/15 to AS 16509. The attacker also sets filtering rules to capture DNS responses from AS 1 sent to 54.202.0.0/15. Notice that in contrast to previous attack, the DNS request from the VA reaches the nameserver and the attacker cannot intercept it. The nameserver issues a response following the request. In step ② the attacker intercepts the response, changes the value of the DNS record to point at the IP addresses controlled by the attacker, and sends the modified response to the VA.

A.4 Attacks Against Single Point DV

Previous work [14] demonstrated BGP hijack attacks against single point DV: the attacker used fraudulent BGP announcements, mapping the prefix of the victim domain to the AS number of the attacker. If the network of the VA accepted that BGP announcement, the DNS lookup requests as well as DV, were performed against the hosts controlled by the attacker. The evaluations in [14] used a domain with a *single* nameserver and the hijacks were aimed at intercepting only the communication with that nameserver. For instance, the sub-prefix hijack attack was aimed at intercepting the sub-prefix with the victim nameserver, while the same-prefix hijacks used the fact that the attacker was located topologically closer to the VA than the victim domain. In reality, domains have multiple nameservers, and the DNS resolvers select a nameserver to which they send a query in an unpredictable fashion. Our measurements show that there are an average of more than 3 nameservers per domain and that there are even domains with more than 30 nameservers. Furthermore, following best practices for resilience typically each nameserver in a domain is located on a different network. For attack in [14] to be practical against realistic domains in the Internet it needs to be extended: since the attacker does not know which nameserver the VA will select, it has to hijack the communication channels between the VA to *all* the nameservers. Therefore, the attacker needs to issue multiple hijacking BGP announcements, per prefix of each nameserver.

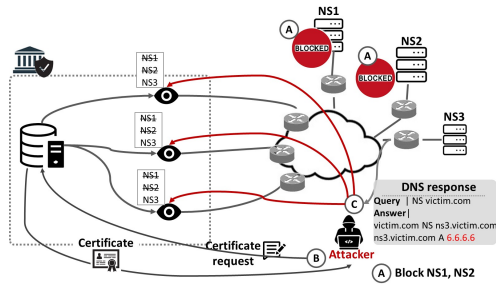


Figure 13: Attacking DV with MultiVA.

Following [14], Let’s Encrypt was the first to deploy multiple domain validation with four VAs (called multiVA), which since February 2020 runs in production mode. Recently [15] demonstrated the security of multiVA of Let’s Encrypt and showed that it significantly raises the bar for attackers, making attacks against DV impractical. The reason is that every VA selects the nameserver to which it sends the query at random and independently of other VAs. The lookup and the validation succeed, if at least three of the responses arrived, and they are identical. Since the attacker does not know to which nameserver each VA sends its query, it has to attack the communication from every VA to any nameserver. For instance, given a domain with 3 nameservers, each VA can send its query to any of the 3 nameservers. Therefore, for a successful hijack of a query the attacker would have to make the network of every VA accept a fraudulent BGP announcement for a prefix of every nameserver, and for a successful hijack of the response, the attacker would have to make the network of every nameserver accept a fraudulent BGP announcement mapping the prefix of the VA to the AS of the attacker. Such attacks are not practical even with very strong attackers.

A.5 Asymmetric Routing Paths

A.5.1 Forward and backward paths. Let the path, that the requests from the VA to the nameserver take, be the forward path, and let the path that the responses from the nameserver to the VA take be the backward path. Both forward and backward paths are computed individually by each BGP router along the path and inserted into the routing tables along the paths. Each AS in the forward and backward paths may have different local preferences and can use various communities and filter configurations for selecting the routes based on the received BGP announcements. This computation often results in asymmetric forward and backward paths.

A.5.2 Reasons for asymmetric routing paths. There are many factors for differences in forward and backward routing paths. During our measurements we identified the following reasons for asymmetric routing:

- Manually configured preference of paths going through cheaper links: typically smaller networks prefer paths to peerings with higher local preference. The best path selection algorithm of BGP assigns higher priority to the local preference than to the “select shortest path” rule. We explain this with the following example: assume we have two possible paths between AS A and AS B: A-X-Y-B and A-Z-B. If sending traffic from AS A to AS X is much cheaper (for AS A) than sending it to AS Z, it could and likely would configure

the local preference of AS A to override the shortest path. Hence, the path from AS A to AS B would be A-X-Y-B. In contrast, if AS Y and AS Z have the same price from the perspective of AS B, it is likely that AS B would not change local preference and would therefore use the shortest path B-Z-A for sending traffic to AS A. Every AS can apply local preference to set precedence of incoming routes and therefore direct outgoing traffic. But the opposite direction for directing incoming traffic is much less controllable, and can be done by coordinating between multiple ASes a special configuration (using `always-compare-med`¹⁴, [RFC3345] [47] or

¹⁴Configures the device always to compare the Multi-Exit Discriminators (MEDs), regardless of the autonomous system (AS) information in the paths. on arranging communities that all the ASes on the path would accept and interpret as external triggers for local preference).

- The best path selection algorithm of BGP uses end-rules that decide according to Interior Gateway Protocol (IGP) metric [RFC3785] [43] or router ID¹⁵ if all important metrics (local preference, MED, AS-path length) are equal. From an external perspective it is not possible to know which parameters an AS uses for computing the best path.

Implications of asymmetric routing on our attack. Which network the attacker will hijack in a real attack in the Internet depends on the location of the attacker and on the topological location of the victim. If the network of the nameserver accepts a bogus BGP announcement of the attacker claiming to originate the prefix of the VA, the responses from the nameserver will be sent through the attacker.

Launching symmetric hijacks. Such scenarios are quite easy to achieve in practice and it is quite a common form of “business intelligence gathering”: A network X wants to eavesdrop on traffic between networks A and B. Network A has peerings in one IXP (say London) and network B in another IXP (for example Amsterdam). The legitimate path from A to B goes through upstream providers that both networks A and B prefer less than their peerings. Network X has peerings with both A and B in the proper IXPs (London and Amsterdam). So the only thing that X needs to do to intercept traffic between A and B is to propagate routes from A to B and vice-versa. It means that the traffic between A and B starts flowing over X (since A and B prefer peerings over upstreams), so X can eavesdrop on it. However, X has to carry the traffic between Amsterdam and London for free (since it was just peering on both sides) and therefore both A and B were benefiting from the redirection by saving some money on transit connectivity.

B ATTACK COMPONENTS

We illustrate the conceptual components of the attack in Figure 13. The attacker first eliminates the nameservers by removing them from the VAs’ list of usable servers. The attacker then launches the prefix hijacks against a network of NS3, and injects a malicious DNS records into a spoofed DNS response.

C VULNERABLE DOMAINS

Through experimental evaluations we found 23.27% Let’s Encrypt-certified domains with nameservers that apply rate limiting, and

¹⁵A router ID is a 32-bit IP address that uniquely identifies a router in an AS.

```

UNKNOWN_SERVER_NICENESS = 376ms
RTT_BAND                 = 400ms

rtt_lost(s){
  s.RTO *=2
}

iter_fill_rtt(){
  for each server s in servers:
    RTO = infra_get_lame_rtt(s)
    If s is new:
      RTO = UNKNOWN_SERVER_NICENESS
    FastestRTO = compute_fastest_rto()
}

iter_filter_order(){
  for each server s in servers:
    if (RTO - FastestRTO <= RTT_BAND)
      move s to front
}

send_query(){
  s = randomly_choose_server(servers)
  if (query(s) = timeout)
    s.TO +=1 // increase timeouts for s
    s.RTO = s.RTO*2 // double s RTO
    if (s.RTO == 12 sec && s.TO = 2)
      // enter probing regime
      // not more than 1 query per RTO
      set_probing_regime(s)
    if (s.RTO > 120sec)
      // enter blocking regime
      // block 900sec until s expires from infra_cache
      infra-host-ttl = 900sec
      infra_cache <- s
}

```

Figure 14: Server selection algorithm of Unbound.

2% of Alexa domains that fragment responses. These are domains with nameservers that the VAs of Let's Encrypt can be forced to query. As an example case study in our work we count nameservers vulnerable to sub-prefix hijacks. Out of 35% nameservers 10.60% are vulnerable to sub-prefix hijack attacks. Alternately, the adversary may select nameservers with some other vulnerability, e.g., depending on the topological location of the attacker, it can also select nameservers that can be same-prefix hijacked. We list vulnerable domains in our dataset in Table 6.

D ANALYSIS OF UNBOUND

D.1 Server Selection

In the first step (function `iter_fill_rtt`) the DNS software uses function `infra_get_lame_rtt` to read the Round Trip Time (RTT) information for each nameserver from the infrastructure cache, called `infra_cache` (this is where the information about the servers is cached). If this is a new nameserver for which Unbound does not have information about RTT, its RTO is set to 376ms. The fastest Round Trip Timeout (RTO) is then marked. The RTO is the timeout including the exponential backoff, it is used for server selection and as a timeout for the transmitted request. The exponential backoff is implemented in function `rtt_lost` in file `rtt.c`.

In the second step Unbound rearranges the list of servers, moving all the servers that satisfy $(RTO - \text{FastestRTO}) \leq 400\text{ms}$ to the front of the server list, this is implemented in function `iter_filter_order`.

In the next step Unbound randomly chooses a nameserver from the list created in second step, and sends a query to it. If the response times-out, the RTO of that server is doubled. When the RTO exceeds 12 seconds after 2 consecutive time-outs, the server enters a 'probing regime'. This allows not more than a single query to that nameserver per RTO period. If the RTO further exceeds

120 seconds, it enters the 'blocking regime'. This means that the nameserver is moved to `infra_cache` for 900 sec (15 minutes) and will not be queried during that time period.

The pseudocode for server selection mechanism in Unbound is described in Figure 14.

D.2 Query Retransmissions

Unbound has two parameters for limiting the number of times that a DNS resolver will retry to resend the query for which no response arrived. Both parameters are defined in `iterator.h` configuration file. The `MAX_SENT_COUNT` parameter is the limit on maximal number of queries per DNS request, which is set to 32. The other is the number of retries per nameserver, defined with `OUTBOUND_MSG_RETRY`, and set to 5. The values of both parameters are hardcoded and cannot be modified.

When the Unbound DNS resolver does not have RTO (retransmission time-out) information about the nameservers in a domain to which it needs to send a query, it sets the RTO of all the nameservers to 376ms and selects a server at random. If any server was queried previously and the response arrived, the RTO reflects the previous RTT value; see server selection analysis in Section 2.2.3. A nameserver is selected at random among all the servers with RTO below 400ms. If the fastest nameserver is 400ms faster than any other server, it is the only one that can be selected.

If the response arrived, the resolution is done, the query is removed from pending queue. If the response does not arrive, the timeout is triggered after the RTO period. The RTT value for that server is updated and the `attempt_count` parameter for that server is incremented. If `OUTBOUND_MSG_RETRY` is reached, remove the server from the list of usable servers. Increment the `total_sent_count` for that query. Once `MAX_SENT_COUNT` is reached, return server fail. Return to step 1. We provide the pseudocode of Unbound retransmission behaviour in Figure 15.

```

MAX_SENT_COUNT = 32
OUTBOUND_MSG_RETRY = 5

SET request_sent_count = 0
WHILE request_sent_count < MAX_SENT_COUNT
  select server by calling iter_server_selection()
  get server_timeout from infra_cache
  send query to selected server
  wait for server_timeout period
  IF success THEN
    update SRTT info in infra_cache
    return
  ELSE
    server_timeout *= 2
    update SRTT info in infra_cache
    server_attempts++
    IF server_attempts >= OUTBOUND_MSG_RETRY
      remove this server from usable server list
  ENDF
  request_sent_count++
  IF request_sent_count >= MAX_SENT_COUNT
    return SERVFAIL
  ENDF
ENDIF
ENDWHILE

```

Figure 15: Query retransmission behaviour in Unbound.

E HITTING IP ID

In this section we describe the IP ID allocation methods and report on the IP ID results we collected from the popular nameservers.

	#Rate Limit	#Fragmentation	#Frag. or Rate-Limit	#Vuln. to sub-prefix hijack	#Total
Let's Encrypt	235,991 23.27%	19,060 1.88%	248,763 24.53%	107,517 10.60%	1,014,056
Alexa	145,280 16.95%	37,624 4.39%	179,242 20.92%	100,709 11.75%	856,887
Total	377,540 20.32%	55,229 2.97%	423,004 22.76%	207,393 11.16%	1,858,165

Table 6: Server-elimination attacks and attacks to obtain fraudulent certificates against domains in our dataset.

To identify the value of the IP ID we send packets from two hosts (with different IP addresses) to a nameserver.

IP Identifier. The 16 bit IP Identifier (IP ID) field in the IP header is used to identify fragments that belong to the same original IP packet [RFC791]. The fragments are then reassembled by the recipient according to source and destination IP addresses, IP ID value and protocol field (e.g., TCP).

Global counter. Initially most operating systems used a globally incremental IP ID assignment which is easy to implement and has little requirement to keep state: just a single counter which is incremented with every packet that is sent. Global counters however were shown to be vulnerable to off-path attacks, [32]. A global counter is still popular in the Internet. Our study shows that 5.53% nameservers use global counter for UDP datagrams and 2.30% nameservers global counters for IP packets with TCP, see details in Table 7. To prevent the attacks some operating systems were patched to randomise their IP ID assignment.

Counter-based bucket. One of the popular algorithms that was also standardised in [RFC7739] is the counter based bucket algorithms. The idea is that an index computed by hash function over the source and destination IP addresses and key, is mapped to an entry in a table. The IP ID value is calculated by choosing a counter pointed to by the hash function. Observing some IP ID values for a pair of source and destination IP addresses does not reveal anything about the IP ID values of the pairs in the other buckets. This algorithm, implemented into recent versions of Windows, Linux and Android. Recently [39] reverse engineered parts of tcpip.sys driver of 64-bit Windows RedStone 4, which allowed breaking this IP ID assignment algorithm. The attack requires the attacker to control i IP addresses in the same class B prefix. The goal of the attacker is to receive the keys used by the IP ID generation algorithm: a 320 bit vector, with two keys K_1 and K_2 that are 32 bits each. During the offline preprocessing phase the attacker uses Gaussian elimination to calculate a matrix using the IP addresses:

$$Z \in GF(2)^{15(i-1) \times 15(i-1)}$$

subsequently, the attacker sends packets to the target server from the i IP addresses that it controls and obtains the IP ID values from the i response packets. The attacker applies the computation to recover the values of the keys, which can be used to predict the IP ID values in Windows 8 and above versions.

The Linux versions 3.0 and above use separate IP ID allocation algorithms for UDP and TCP communication. For TCP the IP ID value is computed per connection, while for UDP [39] demonstrated an attack for predicting the IP ID value similar to Windows. The evaluations demonstrated practical attack times of up to 1.5 minutes at most, see also [39]. Furthermore, in a study which included 69 networks the IP ID values, of the servers that used counter-based

bucket algorithm for IP ID values calculations, could be predicted. In a subsequent work, [30] demonstrated approaches for recovering the IP ID value computed for the TCP communication. Their evaluation also demonstrated practical attacks, which apply to 20% of 100K-top Alexa domains.

Random. Another algorithm selects random IP ID values from a pool of least recently used IP ID values. This algorithm requires maintaining a lot of state, corresponding to the pool of the used IP IDs, however ensures unpredictability of IP ID selection. This approach is implemented in iOS and MacOS.

	Per-Host	Global	Zero	Random and other	N/A	Total
UDP	52.60% 51281	5.53% 5388	7.34% 7152	33.40% 32560	1.14% 1112	100% 97493
TCP	14.43% 14072	2.30% 2247	75.92% 74020	1.30% 1266	6.04% 5888	100% 97493

Table 7: IP ID allocation of in 100K-top Alexa.

F OVERVIEW OF DOMAIN VALIDATION

Validating ownership over domains plays a central role in PKI security. It enables CAs to ensure that a certificate is issued to a real domain owner and prevents attackers from issuing fraudulent certificates for domains that they do not control. Prior to issuing certificates the CAs validate that the entity requesting a certificate for a domain de-facto controls the domain by running domain validation (DV) procedure. This is done by sending challenges to the domain and verifying that the domain correctly echos the challenges. The methods for verifying challenges all depend on DNS, and can be based on: email, whois, zonefile and HTTP/S.

HTTP/S: the user adds to the root directory of the website running at the domain a challenge provided by the CA during DV. **email:** an email is sent to an administrator's email address at the domain, requiring the administrator to visit a challenge URL.

The idea underlying these methods is that the owner of the domain adds to the domain a challenge he receives from the CA after submitting the request for a certificate. The CA can then verify the presence of the correct challenge by sending a query to the domain. An attacker that does not see the challenge and does not control the domain, should not be able to add the correct challenge value to the domain's zonefile or web server, nor will it be able to echo the challenge via email.

F.1 Single Node DV

The idea behind single node DV is that the validation is performed from a single node, which sends queries to one of the nameservers of the target domain. For instance, the CA, at domain `ca.com`, receives

a CSR for domain `example.info`, and creates a challenge `$` value which has to be entered as a DNS CNAME record to the zonefile of the domain, e.g., `$www.example.info. CNAME $.ca.com`. The resolver of the CA queries the domain of the applicant, and checks the presence of the CNAME record.

F.2 Multiple Location DV with multiVA

Following the attacks against DV Let's Encrypt deployed a multiVA mechanism, i.e., performing validation from multiple nodes. The nodes are called Validation Authorities (VAs). MultiVA uses four Validation Authorities (VAs) for validating control over domains. Each VA uses DNS resolver library for looking up resources in the target domains and for validating control over domains during DV. All the VAs are located on an AWS cloud. The CA sends a domain to validate or to lookup over an encrypted connection to the VAs. The VAs perform validation and return the result of the validation over an encrypted channel to the CA. If the validation is successful, the CA issues the requested certificate. For DV to succeed at least three of the four VAs must succeed. The VAs of Let's Encrypt are set up on four network blocks:

- set 1: two IP addresses owned by Flexential Colorado Corp. on AS13649.
- set 2: five IP addresses located on AWS us-east-2 data center.
- set 3: five IP addresses, on AWS eu-central-1 data center.
- set 4: five IP addresses, on AWS us-west-2 data center.

During each lookup or during DV, multiVA selects one IP from each set. The process of multiVA concludes successfully if at least three of four VAs return identical responses. Responses are cached. The DNS software of the Let's Encrypt caps the TTL at 60 seconds.

G DECRYPTING ENCRYPTED TRAFFIC

We perform a MitM (man-in-the-middle) attack on two types of applications: web browser and SMTP MX server, where the attacker functions as a proxy and relays packets between the genuine target server and the victim client application. The attacker first launches a DNS cache poisoning attack or a BGP prefix hijack attack against a victim, to redirect it to the attacker's host for the target domain. We launch DNS cache poisoning attack by intercepting a DNS request from the client via a short lived BGP prefix hijack. We return to

the victim resolver a DNS response mapping the target domain to the nameservers controlled by the attacker. The attacker then responds to subsequent lookup requests for services in the target domain with malicious records. In particular, the attacker maps the web server and the email exchanger (MX) in the target domain to attacker's IP addresses. We then evaluate two attacks: (1) we use our client to access the webserver and (2) we send an email to the email exchanger in the target domain.

In the first attack the attacker functions similarly to a web proxy, and relays every packet it receives to the target webserver. Depending on the webserver, the attacker may leave the source IP address of the real client intact (e.g., if this information is reflected in the objects returned by the webserver, e.g., printed on the page and is visible to the client). The attacker establishes a TLS channel with the client, using the fraudulent certificate to impersonate the target webserver. The attacker poses as a client and establishes a TLS channel with the target server. Within these connections it relays the HTTP objects between the client to the server. In our evaluation we experienced timeouts since the client has to wait until the request from the attacker reaches the real server and then the client. To avoid timeouts we introduce latency to every response we send to the client that is proportional to the time required to send the request from the client to the server and receive a response. We add latency to every packet starting with the TCP SYN ACK. This causes the RTO in TCP of the client to be much longer, and not timeout.

Using this attack, the attacker can not only read and intercept all the exchanged communication but it can also modify the returned objects to inject scripts that will be persistently running in a sandbox on the client and can execute a range of attacks, such as Rowhammer attacks against RAM exploiting charges leak of memory cells via privilege escalation, [31] or Spectre attack [48, 49] against the CPU cache via timing side channels to read data in the cache.

In the latter attack we setup an SMTP server that relays packets between an outbound SMTP server and an MX exchanger (an inbound SMTP server) in the target domain. The latency introduced by this attack is not significant since the victim client does not directly experience the latency introduced by our attacking proxy.

A.9. SMap: Internet-wide Scanning for Spoofing

[Dai21ac]

Tianxiang Dai and Haya Shulman. “SMap: Internet-Wide Scanning for Spoofing”. In: *Annual Computer Security Applications Conference*. ACSAC. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 1039–1050. ISBN: 9781450385794. DOI: 10.1145/3485832.3485917. URL: <https://doi.org/10.1145/3485832.3485917>

SMap: Internet-wide Scanning for Spoofing

Tianxiang Dai
ATHENE Center, Germany
Fraunhofer SIT, Germany

Haya Shulman
ATHENE Center, Germany
Fraunhofer SIT, Germany

ABSTRACT

To protect themselves from attacks, networks need to enforce *ingress filtering*, i.e., block inbound packets sent from spoofed IP addresses. Although this is a widely known best practice, it is still not clear how many networks do not block spoofed packets. Inferring the extent of spoofability at Internet scale is challenging and despite multiple efforts the existing studies currently cover only a limited set of the Internet networks: they can either measure networks that operate servers with faulty network-stack implementations, or require installation of the measurement software on volunteer networks, or assume specific properties, like traceroute loops. Improving coverage of the spoofing measurements is critical.

In this work we present the **Spoofing Mapper (SMap)**: the first scanner for performing *Internet-wide* studies of ingress filtering. SMap evaluates spoofability of networks utilising standard protocols that are present in almost any Internet network. We applied SMap for Internet-wide measurements of ingress filtering: we found that 69.8% of all the Autonomous Systems (ASes) in the Internet do not filter spoofed packets and found 46880 new spoofable ASes which were not identified in prior studies. Our measurements with SMap provide the first comprehensive view of ingress filtering deployment in the Internet as well as remediation in filtering spoofed packets over a period of two years until May 2021.

We set up a web service at <https://smap.cad.sit.fraunhofer.de> to perform continual Internet-wide data collection with SMap and display statistics from spoofing evaluation. We make our datasets as well as the SMap (implementation and the source code) publicly available to enable researchers to reproduce and validate our results, as well as to continually keep track of changes in filtering spoofed packets in the Internet.

CCS CONCEPTS

• Security and privacy → Network security.

KEYWORDS

Ingress Filtering, Spoofing, PMTUD, IPID, DNS

ACM Reference Format:

Tianxiang Dai and Haya Shulman. 2021. SMap: Internet-wide Scanning for Spoofing. In *Annual Computer Security Applications Conference (ACSAC '21)*, December 6–10, 2021, Virtual Event, USA. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3485832.3485917>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACSAC '21, December 6–10, 2021, Virtual Event, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8579-4/21/12... \$15.00

<https://doi.org/10.1145/3485832.3485917>

1 INTRODUCTION

Source IP address spoofing allows attackers to generate and send packets with a false source IP address impersonating other Internet hosts, e.g., to avoid detection and filtering of attack sources, to reflect traffic during Distributed Denial of Service (DDoS) attacks, to launch DNS cache poisoning, for spoofed management access to networking equipment and even to trigger services which can only be accessible to internal users [8, 11, 13, 32, 39]. The best way to prevent IP spoofing is by enforcing Source Address Validation (SAV) on packets, a practice standardised in 2000 as BCP38 [19]: *ingress filtering* for blocking inbound packets and *egress filtering* for blocking outbound packets sent from spoofed IP source addresses. In contrast to egress filtering which has been extensively measured in the last 15 years, only a couple of recent studies provided measurements on the extent of ingress filtering.

Ingress filtering. To enforce ingress filtering the networks should check the source address of an inbound packet against a set of permitted addresses before letting it into the network. Otherwise, the attackers using spoofed IP addresses belonging to the network can trigger and exploit internal services and launch attacks. For instance, by spoofing internal source IP addresses the attackers can obtain access to services, such as RPC, or spoofed management access to networking equipment [RFC3704], the attackers can cause DoS amplification by triggering the ICMP error messages from the attacked hosts to other internal hosts whose IP addresses the attacker spoofed. Enforcing ingress filtering is therefore critical for protecting the networks and the internal hosts against attacks. Nevertheless, despite efforts to prevent IP spoofing, it is still a significant problem. Attacks utilising IP spoofing remain widespread [8, 10, 18, 35, 38, 41].

How widespread is the ability to spoof? There are significant research and operational efforts to understand the extent and the scope of (ingress and egress)-filtering enforcement and to characterise the networks which do not filter spoofed packets; we discuss these in Related Work, Section 2. Although the existing studies and tools, such as the Open Resolver [34] and the Spoofer [5–7, 28, 30] projects, provide a valuable contribution for inferring networks which do not enforce spoofing, they are nevertheless insufficient: they provide a meager (often non-uniform) coverage of the Internet networks and are limited in their applicability as well as effectiveness.

SMap (The Spoofing Mapper). In this work we present the first Internet-wide scanner for networks that filter spoofed inbound packets, we call the Spoofing Mapper (SMap). We apply SMap for scanning ingress-filtering in more than 90% of the Autonomous Systems (ASes) in the Internet. The measurements with SMap show that more than 80% of the tested ASes do not enforce ingress filtering (i.e., 72.4% of all the ASes in the routing system), in contrast to 2.4% identified by the latest measurement of the Spoofer Project

[30]. The reason for this significant difference is the limitation of the previous studies of ingress filtering to a small set of networks.

Limitations of filtering studies. The measurement community provided indispensable studies for assessing “spoofability” in the Internet, and has had success in detecting the ability to spoof in some individual networks using active measurements, e.g., via agents installed on those networks [28, 34], or by identifying spoofed packets using offline analysis of traffic, e.g., [29, 30]. The need to install agents on networks or the ability to obtain traces only from some networks limits the studies to non-uniform coverage of the Internet. Therefore it is not clear how representative these statistics are. Unfortunately, this limitation to a small set of networks creates a bias in the assessments of the overall number of spoofable networks. The extrapolation from the small set of networks to the entire Internet typically result in assessment that at least 30% of the Internet networks do not filter spoofed packets [30, 32]. As we show, the number of spoofable networks is above 72% which is significantly higher than what was previously believed.

Requirements on Internet studies. The key requirements for conducting Internet studies upon which conclusions can be drawn include scalable measurement infrastructure, good coverage of the Internet and a representative selection of measurement’s vantage points. We summarise the limitations of the previous studies below and in Table 1, and compare to SMap.

- *Limited coverage.* Previous studies infer spoofability based on measurements of a limited set of networks, e.g., those that operate servers with faulty network stack [26] or networks with volunteers that execute the measurement software [5–7, 28, 30, 34], or networks that agree to cooperate and volunteer their traffic logs for offline analysis, e.g., [30]. In contrast, the measurements with SMap use standard protocols supported by almost any network with Internet connectivity, for the first time providing studies of ingress filtering that cover the entire IPv4 space.

- *Limited scalability.* Previous approaches require installing agents, need to reproduce loops in traceroutes, or use misconfigurations in networks which limits their scalability. SMap is more scalable than any previous approach, since it merely exchanges requests/responses with networks using a fixed infrastructure of probers. The measurement infrastructure of SMap is not a function of the measured networks, hence adding more networks to the study does not require extending the measurement infrastructure.

- *Limited representativeness.* Volunteer or crowd-sourcing studies, such as the Spoofer Project [28], are inherently limited due to bias introduced by the participants. These measurements are performed using a limited number of vantage points, which are set up in specific networks, and hence are often not representative of the entire Internet. Increasing the coverage and selecting the networks more uniformly is imperative for collecting representative data; [22] showed that the measured network significantly influences the resulting data as well as the derived conclusions. Since SMap measures almost all the IPv4 networks the results are representative of the entire Internet.

- *Limited stability.* Current measurement studies use unstable infrastructures: volunteers running agents can reinstall computers or move to other networks [34]; misconfigured servers [28] (e.g., with open resolution or with faulty network stack) can be updated – all causing the network to “disappear from the radar” although

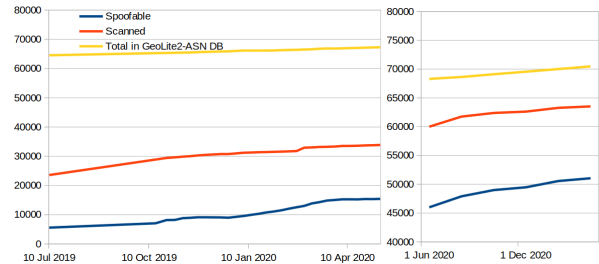


Figure 1: SMap measurements between July’19 and May’21. Domain-based (left) and IPv4-based (right).

it may still be spoofable. Hence, longitudinal studies, such as the Spoofer Project, are biased by the stability of the vantage points, and cannot accurately track deployment of ingress filtering in individual networks. A few works [34] pointed out that the instability of the infrastructure creates discrepancy in the statistics. In particular, repeating the measurements a few weeks later generates other different results.

What SMap improves. The infrastructure of SMap is more stable than those used in previous studies, e.g., we do not risk volunteers moving to other networks. Our measurements do not rely on misconfigurations in services which can be patched, blocking the measurements. The higher stability also allows for more accurate reproduction and validation of our datasets and results, and enables to perform reliable longitudinal studies. We ran ingress filtering measurements with SMap every week over a period of two years (between 10 July 2019 and 10 May 2021). Our results plotted in Figure 1 demonstrate that the number of spoofable ASes is stable and proportionally increases with the growth in the overall number of ASes in the Internet. This is in contrast to previous studies, e.g., [27–29], in which a repeated evaluation even a week later provided different statistics. Our two year long measurements between 2019 and 2021 of more than 90% of Internet’s ASes we found 50,023 new ASes that do not enforce ingress filtering, which were not known before, and confirmed all the other ASes that were found spoofable in prior studies.

Ethical Considerations. Internet-wide scans are important for security research [16, 31] and have proven valuable in improving the security landscape of the Internet, including exposing new vulnerabilities, tracking adoption of defences. Nevertheless, Internet-wide scans introduce also ethical challenges. We communicated with network operators to understand and consider the ethical implications of Internet-wide scans. We identified two issues as particularly important for our measurements: *traffic load* and *consent*.

- *Traffic load.* Network scans, such as [16, 26, 31], require exchanging packets with a large number of Internet networks as well as IP addresses inside the networks. To avoid scanning the Internet we periodically download a dataset of a full scan of the Internet done by Sonar.

- *Consent of the scanned.* It is often impossible to request permission from owners of all the tested networks in advance, this challenge similarly applies to other Internet-wide studies [15, 16, 26, 31]. Like the other studies, [15, 16], we provide an option to opt out of our scans. To opt out the network has to provide either its network

block (in CIDR notation), domain or ASN through the contact page at <https://smap.cad.sit.fraunhofer.de>. Performing security scans is important - the networks that do not enforce filtering of spoofed packets pose a hazard not only to their operators but also to their users, customers and services, as well as other networks. Due to the importance of identifying such networks, in their recent study [30] even make public the (“name-and-shame”) lists of providers with missing or misconfigured filtering of spoofed packets; [30] also discuss stronger measures against spoofable networks, including liability for damages, and various types of regulation. Inevitably, due to the risks that such networks pose to the Internet ecosystem, it is of public interest to know who those networks are. We do not make the identity of the networks, that do not filter spoofed packets, publicly available, but inform the general public on the fraction of such networks and provide their characterisation (i.e., size, geo-location, business type) in Section 5.

Undoubtedly, filtering spoofed packets is critical and networks have to deploy best practices, such as BCP38 [19] and BCP84 [3], to ensure security of the Internet ecosystem. Understanding the extent of filtering is also significant for devising future policies, defence mechanisms or estimating threats and risks to attacks.

Organisation. Our work is organised as follows: we compare our study and SMap to related work in Section 2. In Section 3 we present the design and the implementation of SMap and the measurement techniques that it uses. In Section 4 we report on the data collected with SMap and the statistics that we derived from it. We characterise the networks which we found not to enforce ingress filtering in Section 5. We conclude this work in Section 6.

2 OVERVIEW OF SPOOFING STUDIES

2.1 Egress vs. Ingress

Although there are a few studies of ingress filtering, most studies of spoofing focus on egress filtering. What can be inferred from egress filtering on ingress filtering and vice versa?

In their recent measurement of ingress and egress filtering [30] conclude that filtering of inbound spoofed packets is less deployed than filtering of outbound packets, despite the fact that spoofed inbound packets pose a threat to the receiving network. [25] analysed the networks from Spoofer and open resolver projects and found that 74% of the networks that do not filter outbound spoofed packets, do not filter inbound spoofed packets. A more recent study [24] of 515 ASes found that ingress filtering of inbound spoofed packets is more widely deployed than egress filtering of outbound packets.

The correlation between egress and ingress filtering in previous work shows that the measurements of ingress filtering also provide a lower bound on the number of networks that enforce egress filtering of spoofed outbound packets. Therefore our results on networks that do not enforce ingress filtering imply that at least as many networks do not perform egress filtering.

2.2 Measurements of Spoofability

Measurements of networks that filter spoofed packets in the Internet was previously done using *network traces* or using *vantage points*. We summarise the results of the previous studies in Table 1, and briefly explain them below.

Vantage Points. Measurement of networks which do not perform egress filtering of packets with spoofed IP addresses was first presented by the Spoofer Project in 2005 [5]. The idea behind the Spoofer Project is to craft packets with spoofed IP addresses and check receipt thereof on the vantage points operated by the *volunteers*, i.e., participants who run a “spoofer” software provided by the authors. Based on the data collected by the Spoofer Project many reports were published providing statistics on the deployment of egress filtering in the Internet [6, 7, 28, 30]; we list the statistics in Table 1.

The downside of this approach is that the Spoofer Project requires users to download, compile and execute a software - which also needs administrative privileges to run - once per measurement. This requires not only technically knowledgeable volunteers that agree to run untrusted code, but also networks which agree to operate such vantage points on their premises. [22] argues that extending the limited coverage of the Spoofer Project is difficult: the operators are unlikely to volunteer or conduct measurements that could leak a negative security posture of their networks, including lack of support of BCP38 [19]. Hence, [22] propose that the most viable method to measure filtering of spoofed packets in more networks is by crowd-sourcing. In 2018 [28] performed a one-time study of the Spoofer Project by renting a 2,000 EUR crowd-sourcing platforms with workers that executed the Spoofer software over a 6 weeks period. Their measurements included additional 342 ASes which were not covered by the Spoofer Project previously. Crowd-sourcing studies, in addition to being expensive, are also limited by the networks in which workers are present and do not provide longitudinal and repetitive studies that can be validated and reproduced.

In a recent longitudinal data analysis by the Spoofer Project [30] the authors observed that despite increase in the coverage of ASes that do not perform ingress filtering in the Internet, the test coverage across networks and geo-locations is still non-uniform.

Closely related to *volunteers* is the vantage points measurements with *faulty or misconfigured servers*. [34] noticed that some DNS resolvers do not change the source IP addresses of the DNS requests that they forward to upstream resolvers and return the DNS responses using the IP addresses of the upstream resolvers - a problem which the authors trace to broken networking implementations. [26] used this observation to measure egress filtering in networks that operate such misconfigured DNS resolvers. Such measurements are limited only to networks which operate DNS servers with broken networking implementations: out of 225,888 networks that [26] measured, they could find such DNS servers only in 870 networks.

Since the Open Resolver and the Spoofer Projects are the only two infrastructures providing vantage points for measuring spoofing - their importance is immense as they facilitated many research works analysing the spoofability of networks based on the datasets collected by these infrastructures. Nevertheless, the studies using these infrastructure, e.g., [22, 30], point out the problems with the representativeness of the collected data of the larger Internet. Both projects (the Spoofer and the Open Resolver) acknowledged the need to increase the coverage of the measurements, as well as the challenges for obtaining better coverage and stable vantage points.

Network Traces. To overcome the dependency on vantage points for running the tests, researchers explored alternatives for

Study	Coverage (scanned ASes)	Spoofable ASes	Type	Year	Longitudinal	Reproducible	Scalable
Spoofers Project [5]	202 of 18,000 (1.1%)	52	Egress	2005	✓	X	X
Spoofers Project [7]	1,586 of 44,000 (3.6%)	390	Egress	2013	✓	X	X
Misconfigured servers [26]	2,692 of 48,000 (5.6%)	870	Ingress	2014	X	✓	✓
Traceroute [29]	1,780 of 56,000 (3.2%)	703	Ingress	2017	X	X	(✓)
IXP traces [27]	700 of 56,000 (1.3%)	393	In & Eg	2017	X	X	X
Amazon Turk Spoofers Project [28]	784 of 56,000 (1.4%)	48	Egress	6w. in 2017	✓	X	X
Spoofers Project [30]	5,178 of 66,000 (7.8%)	1,631	In & Eg	2019	✓	X	X
SMap	63,522 of 70,468 (90%)	51,046	Ingress	2019-21	✓	✓	✓

Table 1: Comparison between SMap and other studies.

inferring filtering of spoofed packets. A recent work used loops in traceroute to infer ability to send packets from spoofed IP addresses, [29]. This method detects lack of ingress filtering only on provider ASes (i.e., spoofable customer ASes cannot be detected). The study in [29] identified loops in 1,780 ASes, which is 3.2% of all the ASes, and 703 of the ASes were found spoofable. Although a valuable complementary technique for active probes with vantage points, this approach has significant limitations: in the absence of loops ingress filtering cannot be inferred, alternately a forwarding loop in traceroute does not imply absence of filtering at the edge, since a loop resulting from a transient misconfiguration or routing update can occur anywhere in the network. Therefore, to identify a lack of ingress filtering reliably one needs to detect a border router and, more importantly, the traceroute loops need to be reproduced - a difficult problem in practice. Furthermore, reproducing or validating the dataset after some time is virtually impossible as the odds for failures rapidly increase. Running traceroutes is also challenging: black-holes in traceroutes, whereby the routers do not respond to probes or when routers have a limit for ICMP responses, are common in Internet [33].

[27] developed a methodology to passively detect spoofed packets in traces recorded at a European IXP connecting 700 networks. The limitation of this approach is that it requires cooperation of the IXP to perform the analysis over the traffic and applies only to networks connected to the IXP. Allowing to identify spoofing that defacto took place, the approach proposed in [27] misses out on the networks which do not enforce filtering but which did not receive packets from spoofed IP addresses (at least during the time frame in which the traces were collected).

A range of studies analysed network traces for ingress filtering using IP address characteristics [4, 9, 10, 14, 36], or by inspecting on-path network equipment reaction to unwanted traffic, [44]. In addition to a limited coverage, the studies do not support longitudinal and repeating data collection and analysis, and cannot be reproduced as they do not make the datasets of their studies public.

3 SCANNING FOR SPOOFABLE NETWORKS

3.1 Dataset

SMap architecture consists of two parts: dataset scan and ingress filtering scan. The dataset scan collects the popular services using two methods: domain-based scan and IPv4 based scan. In IPv4 scan to locate the services SMap probes every IP, checking for open

ports that correspond to the services that we need; for instance, port 25 for Email, 53 for DNS, 80/443 for Web. To reduce the traffic volume of the scan, instead of probing each IP address for target ports, SMap enables also query of the input domains for services. For every domain, it queries the IP and hostname of the services, e.g., (A, MX) for Email server, A for Web server, (A, NS) for name server.

3.2 Methodology

The measurement methodology underlying SMap uses active probes, some sent from spoofed as well as from real source IP addresses to popular services on the tested networks. The spoofed source IP addresses belong to the tested networks (similarly to the Spoofers Project [5]). The idea behind our methodology is that if the packets with spoofed addresses reach the services in the tested networks, they trigger a certain action. This action can be measured remotely. If the action was not triggered, we conclude that spoofed packets did not reach the service.

We develop three techniques to detect if networks filter spoofed traffic based on our methodology: DNS lookup, IPID and PMTUD based. Using popular services ensures that our measurements apply to as many Internet networks as possible.

SMap consists of the orchestrator which coordinates and synchronises the prober hosts. The prober hosts receive the dataset of networks to be scanned for spoofability from the orchestrator. The probes then run IPID, PMTUD and DNS lookup tests against the services on the dataset list. SMap applies one test at a time for each AS in the dataset. Each successful test indicates that packets from a spoofed IP address reached the destination on the target network, implying that the target AS does not filter spoofed packets. On the other hand, a failed test may indicate that one of the ASes on the path between the probes and the service on the target AS may be filtering spoofed packets.

The results from the tests are stored in the backend database. The GUI displays the results of the measurements at <https://smap.cad.sit.fraunhofer.de>. We next explain each measurement technique. In our measurements in Section 4 we compare the success and applicability of each technique.

3.3 IPID

Each IP packet contains an IP Identifier (IPID) field, which allows the recipient to identify fragments of the same original IP packet.

The IPID field is 16 bits in IPv4, and for each packet the Operating System (OS) at the sender assigns a new IPID value. There are different IPID assignment algorithms which can be categorised as: random and predictable. Predictable category uses either a global counter or multiple counters per designation IP address, such that the counter is incremented in predictable quotas. Random category selects each IPID value at random from a pool of values.

Recent work showed that even TCP traffic gets fragmented under certain conditions [12]. Fragmentation has long history of attacks which affect both the UDP and TCP traffic [21, 23, 40].

Methodology. We use services that assign globally incremental IPID values. The idea is that globally incremental IPID [RFC6864] [42] values leak traffic volume arriving at the service and can be measured by any Internet host. Given a server with a globally incremental IPID on the tested network, we sample the IPID value (send a packet to the server and receive a response) from the IP addresses controlled by us. We then generate a set of packets to the server from spoofed IP addresses, belonging to the tested network. We probe the IPID value again, by sending packets from our real IP address. If the spoofed packets reached the server, they incremented the IPID counter on the server - an event which we infer when probing the value from our real IP address the second time.

The challenge here is to accurately probe the increments rate of the IPID value (caused by the packets from other sources not controlled by us), in order to be able to extrapolate the value that will have been assigned to our second probe from a real source IP. This allows us to infer if the spoofed packets incremented the IPID counter.

Identifying servers with global IPID counters. We send packets from two hosts (with different IP addresses) to a server on a tested network. We implemented probing over TCP SYN, ping and using requests/responses to Name servers and we apply the suitable test depending on the server that we identify on the tested network. If the responses contain globally incremental IPID values - we use the service for ingress filtering measurement with IPID technique. We located globally incremental IPID in 63.27% of the measured networks. There are certainly more hosts on networks that support globally incremental IPID values, yet our goal was to validate our measurement techniques while keeping the measurement traffic low - hence we avoided scanning the networks for additional hosts and only checked for Web, Email or Name servers with globally incremental IPID counters via queries to the tested domain.

Statistics of IPID values distribution among tested servers are plotted in Figure 2. When ICMP is filtered, it results in ERROR, when run with TCP, the IPID values are often zero (i.e., ZERO IPID in graph) in Figure 2. To improve coverage of the IPID technique we merge the ICMP&TCP and ICMP&UDP results for each server in our measurements.

Measuring IPID increment rate. The traffic to the servers is stable and hence can be predicted, [43]. We validate this by sampling the IPID value at the servers which we use for running the test. One example evaluation of IPID sampling on one of the busiest servers is plotted in Figure 3. In this evaluation we issued queries to a Name server at 69.13.54.XXX during three minutes, and plot the IPID values received in responses in Figure 3 - the identical patterns demonstrate predictable increment rates. Which means that the traffic to the server arrives at a stable rate.

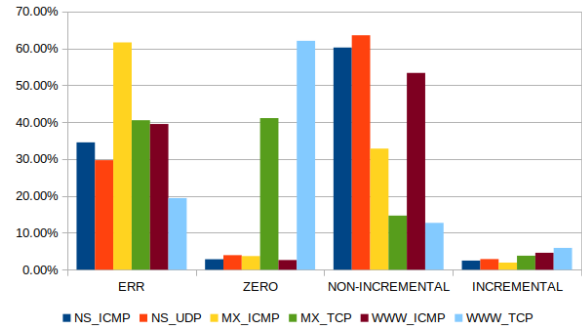


Figure 2: IPIDs on servers in dataset.

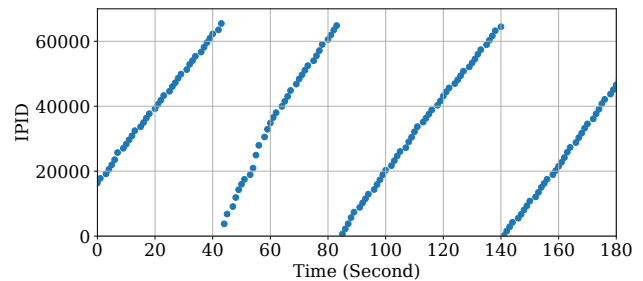


Figure 3: IPID of Name server 69.13.54.XXX during 180sec.

Accuracy of IPID measurements. The IPID techniques are known to be difficult to leverage, requiring significant statistical analyses to ensure correctness. Recently, [17, 37] developed statistical methods for measuring IPID. However, in contrast to our work, the goal in [17, 37] is different - they use IPID to measure censorship and have additional sources of inaccuracy, which do not apply to our measurements: (1) the measurements are applied against client hosts, which results in significantly higher noise than our measurements against servers - the clients move between networks, change IP addresses, the clients are located behind intermediate devices, such as Network Address translators (NAT) and firewalls - which also prevents direct measurements; (2) inaccuracies in geolocation tools, which do not apply to our study since we do not need to know the location to measure ingress filtering, (2) additional network mechanisms (anycast, rerouting, traffic shaping, transient network failures). All these can only cause us to classify the server as not 'testable', but do not impact 'spoofable' outcomes. Furthermore, the IPID measurement methods in prior work use TCP-RST packets to increment IPID, which are often blocked in firewalls. In contrast, we use packets which are not blocked such as DNS queries or TCP-SYN.

Inferring spoofing. We use the following components: the prober at IP address 7.7.7.7 and a server at IP address 1.2.3.7 that uses globally incremental IPID, illustrated in Figure 4. Using the prober at 7.7.7.7, we measure the value of the IPID and the rate at which IPID increments. We use linear regression with

Ordinary Least Square (OLS) method to estimate the relation between IPID and timestamp t . Since IPID is incremental, it holds: $IPID = a * t + b + \epsilon, \epsilon \sim N(0, \sigma^2)$

We send N probes to 7.7.7.7 (in step (1)). With N probes, we can estimate a, b and σ using OLS method in step (2). In step (3) in Figure 4 we send a set of $M = 6 * \sigma$ packets from a spoofed source IP address 1.2.3.6 (belonging to the probed network). In step (4) at time T_{M+N+1} we sample the IPID value $Z = IPID_{M+N+1}^{real}$ from the server from the prober's real IP address 7.7.7.7 - this is needed in order to receive the response. We check the IPID value Z in step (5) in Figure 4. Taking the linear regression model into consideration, we can calculate $IPID_{M+N+1}^{esti}$ at time T_{M+N+1} . If the M spoofed packets are filtered, according to 3-sigma rule, there is a 99.73% possibility that: $IPID_{M+N+1}^{esti} - 3 * \sigma \leq Z \leq IPID_{M+N+1}^{esti} + 3 * \sigma$. However, if the spoofed packets are not blocked, a.k.a. there is no ingress filtering, the IPID counter should have an additional increment of M . Thus $Z > IPID_{M+N+1}^{esti} + 3 * \sigma$, which is also $Z > IPID_{M+N+1}^{esti} + M/2$.

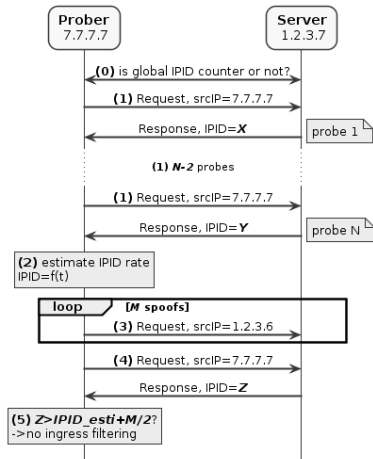


Figure 4: Sequence diagram for IPID technique.

We define outcomes of a test with IPID technique as *spoofable, applicable, non-applicable, N/A*; see Table 2. The IPID technique is not applicable if the IPID counter is constant zero or if the IPID counter is not globally incremental.

Category	IPID	PMTUD	DNS
Spoofable	no filtering	no filtering	no filtering
Applicable	server w/globally incremental IPID	host supports PMTUD	has DNS server
Non-applicable	random IPID or per-dest IPID or IPID=0	(DF=0 & MF=0) or (DF=1 or MF=1) & no change	no DNS server found
N/A	host unreachable or firewall or packet loss or load balancer	host unreachable or misconfigured service or firewall or packet loss	

Table 2: Outcomes of tests.

3.4 PMTUD

Path Maximum Transmission Unit Discovery (PMTUD) determines the MTU size on the network path between two IP hosts. The process starts by setting the Don't Fragment (DF) bit in IP headers. Any router along the path whose MTU is smaller than the packet will drop the packet, and send back an ICMP Fragmentation Needed / Packet Too Big (PTB). The payload of the ICMP packet contains the IP header and the first 8 bytes of the original packet that triggered the error as well as the MTU of the router that sent the ICMP message. After receiving an ICMP PTB message, the source host should either reduce its path MTU appropriately or unset the DF bit.

A study of CAIDA datasets in 2017 found 3M ICMP fragmentation needed packets sent by routers in the Internet, with about 1K routers sending ICMP error message with next hop MTU of less than 500 Bytes [20].

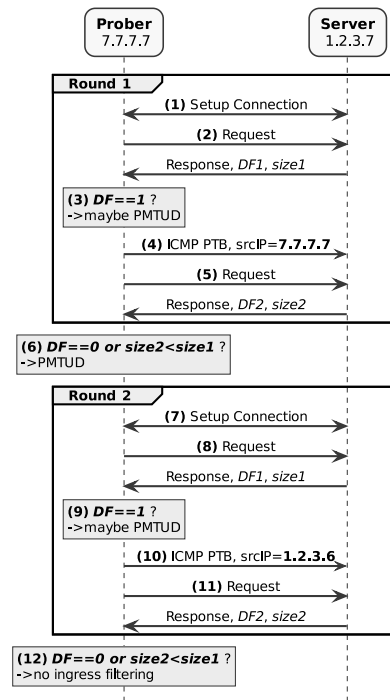


Figure 5: Sequence diagram for PMTUD technique.

Methodology. The core idea of the Path MTU Discovery (PMTUD) based tool is to send the ICMP Packet too Big (PTB) message from a spoofed source IP address, belonging to the tested network, and in the 8 bytes payload of the ICMP to insert the real IP address belonging to the prober. If the network does not enforce ingress filtering, the server will receive the PMTUD message and will reduce the MTU to the IP address specified in the first 8 bytes of the ICMP payload. We first probe the MTU to a service on the tested network, then send ICMP PTB from a spoofed IP address. If the packet arrives at the service, it will reduce the MTU to our prober, and we will identify this event in the next packet from the service -

this event implies that the tested network does not apply ingress filtering.

Identifying servers that support PMTUD. We measured networks that support PMTUD (i.e., do not filter ICMP Fragmentation Needed (Type 3, Code 4) messages), and found that 85.92% of the tested networks support PMTUD.

Inferring spoofing. The PMTUD test is illustrated in Figure 5. We establish a TCP connection to a server on the tested network. Then we send Request1 and receive Response1. If DF bit is not set, the server does not support PMTUD. Otherwise, we send an ICMP PTB with smaller MTU. Following that, we request again and get Response2. If $DF_1 == 1$ and ($DF_2 == 0$ or $size_2 \leq size_1$), the server supports PMTUD. Now we can proceed to test if ingress filtering is enforced. We spoof an ICMP PTB with smallest MTU, using server's neighbour IP as source IP address. Once that is done, we make another request. The server is not protected by ingress filtering if following condition applies: $size_3 \leq size_2$ or ($DF_2 == 1$ and $DF_3 == 0$).

We define outcomes of a test with PMTUD technique as *spoofable*, *applicable*, *non-applicable*, *N/A*; see rightmost column in Table 2.

3.5 DNS Lookup

DNS provides lookup services to networks. Upon receiving a DNS request, the resolver performs the lookup of the requested domain name and returns the response with the requested record.

Methodology. We send a DNS request to the tested network from a spoofed IP address belonging to the tested network. If the network does not enforce ingress filtering, the request will arrive at the DNS resolver on that network. A query from a spoofed source IP address will cause the response to be sent to the IP address from which the request was sent, i.e., the spoofed IP address. Since we do not control the spoofed IP address, we will not be able to observe this event and hence will not be able to infer if the DNS resolver received our request or if the request was filtered due to spoofing. To obtain insights into the traffic arriving at the resolver in the tested network we utilise the payload of the DNS request: the query contains the domain which we own, set up on Name servers that we control. Namely, eventhough the response from the DNS resolver will be returned to the spoofed IP address and will not be received by us, the DNS request will be issued to our Name servers, which is an indication that the DNS resolver on the tested network received our DNS request, sent from spoofed IP address.

Identifying DNS resolvers. The main challenge here is to locate the DNS resolvers within a domain/network and to trigger a DNS request to our Name servers. We use Email service in the target networks (retrieved via the MX type request in the target domain) to find the DNS resolvers. We send an email to target domain's Email server from one of our unique subdomains with a non-existing recipient set in the destination. This causes the Email server on the tested network to generate a Delivery Status Notification (DSN) error message [RFC3464] to our Email server. To be able to send us the DSN, the Email server will request the resolver on the tested network, to provide it the MX and A/AAAA records of our Email exchanger. At the same time, it may also trigger anti-spam checking, which requests (SPF/TXT, PTR, DKIM, DMARC)-type records in domains under our control. By monitoring the DNS queries at our

Name servers, we collect the IP addresses of the resolvers. Using this methodology we identified 49,252 DNS resolvers in 7,141 networks. However, in our regular IPv4 scan, to reduce Email traffic in the Internet, we use the list of servers with UDP port 53 open from Project Sonar as input.

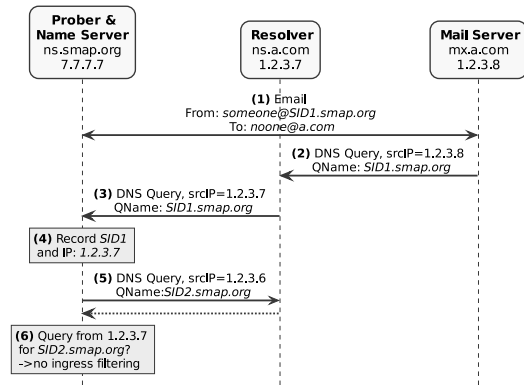


Figure 6: Sequence diagram for DNS lookup technique.

Inferring spoofing. Given a DNS resolver at IP 1.2.3.7, we send a DNS query to 1.2.3.7 port 53 asking for a record in domain under our control. The query is sent from a spoofed source IP address belonging to the tested network. We monitor for DNS requests arriving at our Name server. If a query for the requested record arrives from 1.2.3.7, we mark the network as not enforcing ingress filtering. The process is illustrated in Figure 6, steps (1-4) locate the IP address of the DNS resolver, and steps (5,6) test for ingress filtering on that network.

4 INTERNET MEASUREMENTS

In this section we report on our Internet-wide measurement of ingress filtering with SMap. Our dataset collection with SMap has been initiated on July 2019 continually collected data over a period of one year, of over 6M domains and an entire IPv4 address block.

4.1 Dataset

SMap first collects the dataset of services. Our dataset is constructed as follows: we periodically download the entire IPv4 scan from Sonar Project [2]. We use the scan results on UDP port 53 as input for Name servers and DNS resolvers, scan data on TCP port 25 for Mail servers and scan results on TCP port 80 for Web servers. Besides, we also make use of forward DNS responses and reverse DNS responses from Sonar Project to help find hostnames of servers. In the latest dataset from Sonar, we have services hosted in 63,522 ASes (Table 3) with 4,256,598 DNS servers in 38,838 ASes, 16,478,938 Email servers in 38,937 ASes, and 62,455,254 Web servers in 61,535 ASes; see Table 4.

4.2 Ingress Filtering Results

Domain-scan and IPv4-scan both show that the number of spoofable ASes grows with the overall number of the ASes in the Internet, see Figure 1. Furthermore, there is a correlation between fraction

Technique_Service	Spoofable		Applicable		Non-Applicable	N/A	Total ASes
IPID_NS	8,752	23.07%	12,056	31.78%	25,881	25,585	63,522
IPID_MX	4,355	21.48%	6,861	33.84%	13,416	43,245	63,522
IPID_WWW	30,963	51.83%	39,370	63.27%	22,891	2,608	63,522
IPID_ANY	32,248	56.25%	41,199	67.52%	22,853	1,299	63,522
PMTUD_NS	9,054	24.16%	11,592	30.93%	25,885	26,045	63,522
PMTUD_MX	23,078	68.69%	27,127	80.74%	6,471	29,924	63,522
PMTUD_WWW	41,959	76.91%	47,524	87.11%	7,034	8,964	63,522
PMTUD_ANY	43,473	75.98%	49,161	85.92%	8,053	6,308	63,522
DNS lookup	25,407	40.00%	44,577	70.18%	-	-	63,522
ANY	51,046	80.90%	58,432	92.61%	4,662	428	63,522

Table 3: Collected data and analysis per AS view.

	Name Server	Email Server	Web Server
#IPs	4,256,598	16,478,938	62,455,254
#Blocks	697,851	748,406	3,207,393
#Prefixes	229,981	217,334	542,983
#ASes	38,838	38,937	61,535

Table 4: Servers in tested networks.

of scanned domains and ASes. Essentially the more domains are scanned, the more ASes are covered, and more spoofable ASes are discovered; see Figure 7. This result is of independent interest as it implies that one can avoid scanning the IPv4 and instead opt for domains-scan, obtaining a good enough approximation. This not only reduces the volume of traffic needed to carry out studies but also makes the study much more efficient.

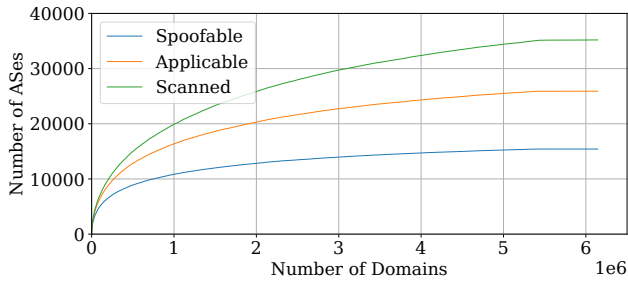


Figure 7: As we scan more domains, we cover more ASes and discover more spoofable ASes.

Further, to avoid single point of failure it is recommended that the Name servers of a domain are hosted in multiple networks. This is also our observation when correlating between domains and ASes. Essentially we find that when testing one domain for each server we can obtain different results, depending on the AS that the server is hosted on.

The results of the ingress filtering measurements with SMap are summarised in Table 3. The techniques that we integrated into SMap (IPID, PMTUD, DNS lookup) were found applicable to more than 92% of the measured ASes. Using SMap we identified 80%

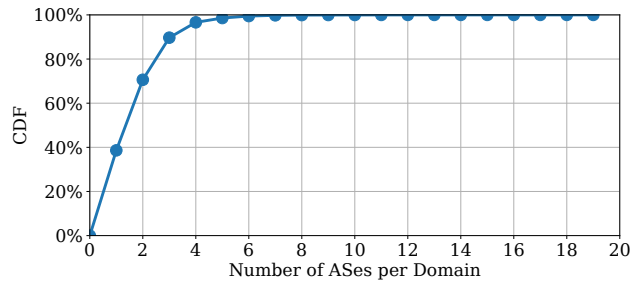


Figure 8: Fraction of domains hosted in multiple ASes. We check how many ASes host services of one domain: 70% of the domains are hosted in one or two ASes.

of the ASes that do not enforce ingress filtering. In what follows we compare the effectiveness of the techniques, explain causes for false negatives and failures. In the rest of this section we explain and analyse the applicability of our results and the success of the different techniques, discuss errors and compare to the results in previous studies.

4.3 Applicability and Success

As can be seen in Table 3 the most applicable technique is PMTUD against Web servers, which applies to a bit more than 87% of the ASes, yielded the highest fraction of spoofable ASes. This is not surprising, since the number of web servers is much larger than the others and it is recommended not to block ICMP to Web servers to allow for path MTU discovery.

We next compare the success and applicability of tests with PMTUD and IPID techniques against Email, Name and Web servers. In order to compare the effectiveness of the PMTUD and IPID measurement techniques as well as their applicability, we define the spoofable and applicable rates, as follows:

$$Rate_{spoofable} = \frac{N_{spoofable}}{N_{total} - N_{NA}}, Rate_{applicable} = \frac{N_{applicable}}{N_{total} - N_{NA}}$$

The spoofable rate reflects the fraction of the networks found not to apply ingress filtering and the applicable rate means applicability of the test technique. The coverage of each of the three techniques for different types of servers (Web, Name, and Email) is plotted in Figure 9.

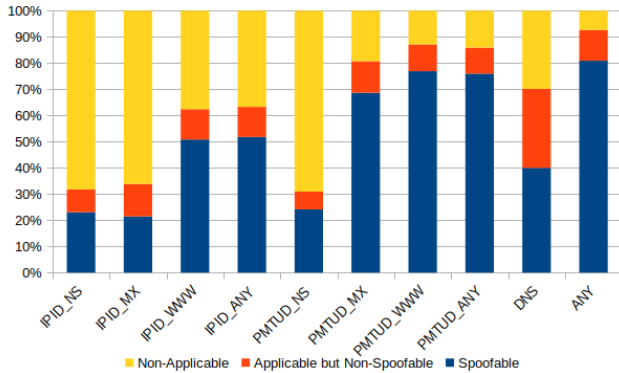


Figure 9: Coverage of the measurement techniques.

Figure 9 shows that PMTUD technique (listed as “PMTUD_ANY” in Figure 9) has a better test rate than either of the IPID and DNS tests, which indicates that PMTUD is still widely supported. Between the other two, DNS test has a slightly higher applicability than IPID test, which shows that globally sequential IPID is less supported now. In Figure 11 we similarly see that the fraction of spoofable networks that can be found through IPID and PMTUD is higher than when measured with the other methodologies; Figure 11 plots the networks found spoofable via IPID vs PMTUD excluding “N/A” networks.

In general, tests against Web servers have a higher applicability rate than the tests with Email or DNS servers, regardless of which technique was used (IPID or PMTUD). The number of Web servers is much larger than the others. It is much easier to setup a Web server than Email server or DNS server. Considering that DNS servers and Email servers are more likely to be hosted by providers, they also have higher probability to get new system updates. Furthermore, we find that when a Web server is not available (“N/A”), both Email and DNS servers cannot be tested, either. This also results in much higher N/A outcomes for tests against Email and DNS servers as opposed to Web servers.

The higher applicability of the tests against web servers also correlates with a higher number of spoofable networks. In Figure 10, we show the relationships between the applicability of SMap measurement techniques to different services and the overlap between them.

4.4 Errors

We define the result of SMap evaluation successful (i.e., true positive) if at least one of the three tests outputs that the tested network does not filter spoofed packets: either the IPID value on the server in the tested network was incremented as expected (IPID test) or we receive a query at our domain (DNS test) or the server on the tested network reduced the MTU of the packets sent to us (PMTUD test). When either of the three techniques provides a positive result, we mark the network as *not filtering*.

SMap does not make mistakes when reporting a network as not filtering. However, it can have false negatives: when the scan does not report network as not filtering when a network does not filter spoofed packets.

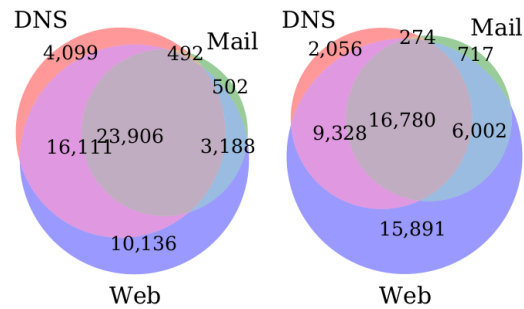


Figure 10: Number of Applicable (left) and Spoofable (right) ASes according to service type.

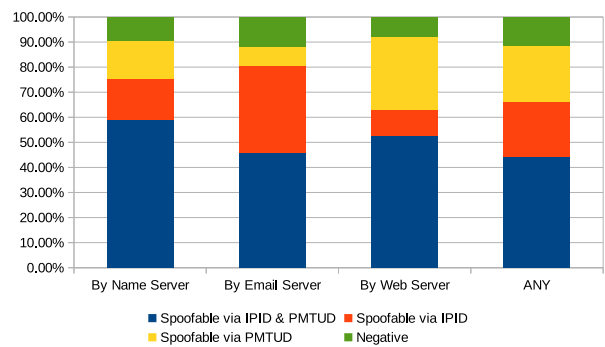


Figure 11: Comparison of spoofability via IPID and PMTUD.

4.4.1 *No False Positives.* Our techniques are not susceptible to false positives, that is, classification of the tested network as filtering spoofed packets when in fact it does not do so. This is a side effect of our methodology - only when spoofing is not filtered will the “test action” be triggered.

IPID technique. When spoofing is not filtered the counter on the server will be incremented - which is the test action. At the probing phase the counter’s value will equal or large than the expected value after the increment phase. The repeated measurements ensure that we do not accidentally interpret noise (i.e., packets from other sources to the same server) as lack of ingress filtering.

DNS technique. When spoofing is not filtered the DNS resolver on the tested network will receive a DNS request from a spoofed IP address to our domain. Hence a query at our domain is the test action that spoofed packets are not filtered.

PMTUD technique. Reduction of the MTU of the packets sent from the test server to our network is the action which indicates that spoofing filtering is not enforced.

4.4.2 *False Negatives.* False negatives in our measurements mean that a network that does not perform filtering of spoofed packets is not marked as such. We next list the causes of false negatives for each of our three techniques. Essentially the false negatives cannot be resolved, and therefore our measurement results of networks that enforce ingress filtering introduce a lower bound. The networks that we classify as those that do not apply ingress filtering -

definitely allow packets from spoofed IP addresses into the network. The networks which were not classified as “not enforcing ingress filtering”, could still be “not enforcing ingress filtering”, but this cannot be determined using our techniques.

IPID technique. Load balancing can introduce a challenge in identifying whether a given network enforces ingress filtering. As a result of load balancing our packets will be split between multiple instances of the server, hence resulting in low IPID counter values. There are different approaches for distributing the load to different instances, e.g., random or round robin, which makes it impossible to identify whether a “load-balanced-server” is on a network which applies ingress filtering or not.

Anycasted server instances can also introduce a challenge in inferring ingress filtering enforcement. We identified such cases by performing traceroutes to the server.

DNS technique. Firewalls, blocking incoming packets on port 53, would as a result generate a similar effect as ingress filtering on our servers: we would not receive any DNS requests to our domain. However, such a setting does not indicate that the tested network actually performs ingress filtering.

PMTUD technique. Firewalls are often configured to block ICMP packets. In such case the evaluation result is similar as when a tested network does not enforce ingress filtering: our PMTUD packets will be blocked by the firewall, but not because they originate from an IP address that belongs to the tested network but because the firewall blocks ICMP packets. This case can be identified by sending ICMP PMTUD packets from an IP address that does not belong to the network. If the ICMP packets are not blocked (but were blocked when the packets were sent from a spoofed IP address) then the network does not block ICMP packets and does enforce IP spoofing filtering. On the other hand if the packets are blocked then one cannot determine if the blocking is done because of ICMP or because of filtering of spoofed IP addresses.

4.5 Comparison with Other Measurements

To understand the effectiveness of our methodologies we compare the results of our measurements with the active measurements of ingress filtering performed by the CAIDA Spoofer Project. These include two types of measurements: *using traceroute* and *using agents*. The spoofer project is the only measurement study that makes the datasets from their scans available online. The traceroute approach and the agents approach are the only two other active measurements of enforcement of ingress filtering (see Related Work Section 2). We crawled all the 217,917 session reports in 2019 of CAIDA Spoofer Project. These included 2,867 ASes with Spoofer Project agents, and 2,500 ASes with Spoofer Project traceroute loops (total of 5,367 ASes). Using our methodologies we measured 63,522 ASes, which is substantially more than the previous studies all together. We compare between our results and the other two methodologies below.

Traceroute Active Measurements. We analyse the datasets from the traceroute measurements performed by the CAIDA Spoofer Project within the last year 2019, [29]. The measurements identified 2,500 unique loops, of these 703 were provider ASes, and 1,780 customer ASes. The dataset found 688 ASes that do not enforce ingress filtering. Out of 688 ASes found with traceroutes by the

Spoofer Project, we could not test 4 ASes (none of our tests applied) and 36 ASes were not included in our tests (those ASes could not be located from domain names - due to our attempt to reduce traffic and not to scan IPv4 but to collect the services via domain names). The rest of the ASes agree with our measurement results.

Agents Active Measurements. Agents with active probes found 608 ASes that were found not to be enforcing ingress filtering using the agents approach of the Spoofer Project (these include duplicates with the traceroute loops measurements). Those contain some of the duplicates from traceroute measurements: together both approaches of the Spoofer Project found 1,113 ASes to be spoofable. Apart from 57 ASes not included in our tests, we could not test 9 ASes, the rest were also identified by our tests.

Although the agents provide the optimal setup for testing filtering, with control over the packets that can be crafted and sent from both sides, as we explain in Related Work Section 2, this approach is limited only to networks that deploy agents on their networks. In contrast, SMap provides better coverage since it is potentially applicable to every network that has one of the services that are required in our tests.

In total, our results identified 51,046 ASes to be spoofable, which is more than 80% of the ASes that we tested. This is also 50,023 ASes more than that both the traceroute and the agents approaches found.

These findings show that SMap offers benefits over the existing methods, providing better coverage of the ASes in the Internet and not requiring agents or conditions for obtaining traceroute loops, hence improving visibility of networks not enforcing ingress filtering.

5 NETWORKS ANALYSIS

In order to understand if there are differences in enforcement of ingress filtering between different network types and different countries, we perform characterisation of the networks that we found to not be filtering spoofed packets. Specifically, we ask the following questions: *Does business type of networks or geo-location of networks influence filtering of spoofed packets?*

To derive the geo-location of ASes we used MaxMind GeoLite2 GeoIP database [1]. The results are listed in Table 5. The tested ASes are distributed across different countries, with most ASes being in large countries, like US and Russia. The ration of spoofable ASes ranges between 67% and 84%, with Ukraine leading with the fraction of spoofable networks, with 84%. Surprisingly the ratio between the geolocation and spoofed packets is similar across different countries, with USA and Russia leading with 32% of the networks and 33% of the networks respectively, that do not filter spoofed packets.

We also want to understand the types of networks that we could test via domains-wide scans. To derive the business types we use the PeeringDB. We classify the ASes according to the following business types: content, enterprise, Network Service Provider (NSP), Cable/DSL/ISP, non-profit, educational/research, route server at Internet Exchange Point (IXP)¹ We plot the networks that do not enforce ingress filtering according to business types in Figure 12.

¹A route server directs traffic among Border Gateway Protocol (BGP) routers.

Country	Tested ASes	Spoofable ASes	Spoofable Ratio
US	16,138	12,385	76.74%
BR	7,692	6,447	83.81%
RU	4,906	4,221	86.04%
PL	2,092	1,739	83.13%
DE	2,171	1,677	77.25%
GB	2,231	1,648	73.87%
UA	1,776	1,547	87.11%
IN	1,970	1,480	75.13%
ID	1,412	1,236	87.54%
AU	1,625	1,234	75.94%
CA	1,484	1,184	79.78%
FR	1,310	1,036	79.08%
NL	1,308	1,026	78.44%
IT	1,013	850	83.91%
ES	1,001	783	78.22%
AR	918	733	79.85%
RO	962	720	74.84%
JP	782	606	77.49%
HK	743	565	76.04%
CZ	673	560	83.21%

Table 5: Top-20 Countries with most tested ASes.

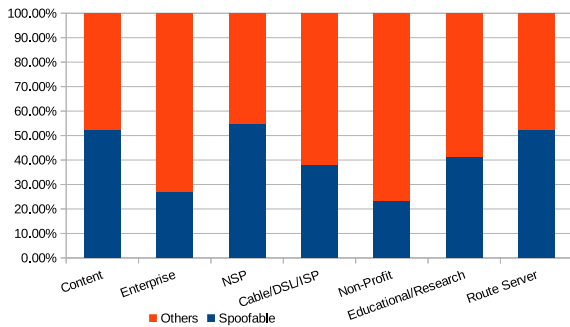


Figure 12: Spoofable ratio across ASes' types. AS type is queried from PeeringDB.

According to our study enterprise and non-profit networks enforce ingress filtering more than other networks. In contrast, NSPs contain the most networks that do not enforce ingress filtering.

There is a strong correlation between the AS size and the enforcement of spoofing, see Figure 13. Essentially, the larger the AS, the higher the probability that our tools identify that it does not filter spoofed packets. The reason can be directly related to our methodologies and the design of our study: the larger the network the more services it hosts. This means that we have more possibilities to test if spoofing is possible: for instance, we can identify a higher fraction of servers with a globally incremental IPID counters, which are not "load balanced". In Figure 14 we plot the statistics of the tested networks according to their size and type. The results show a correlation between the size of the network and its type.

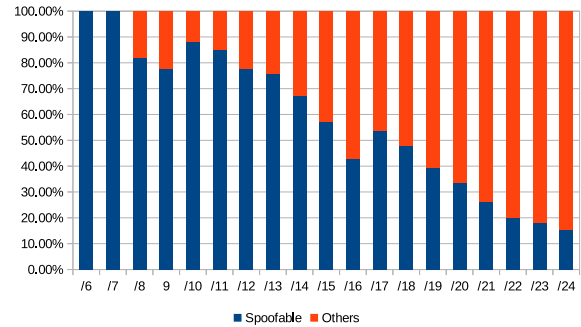


Figure 13: Spoofable ratio according to networks' sizes. Network size is calculated from GeoLite2-ASN database.

For instance, most NSP networks are large, with CIDR/6. This is aligned with our finding that among NSP networks there was the highest number of spoofable networks.

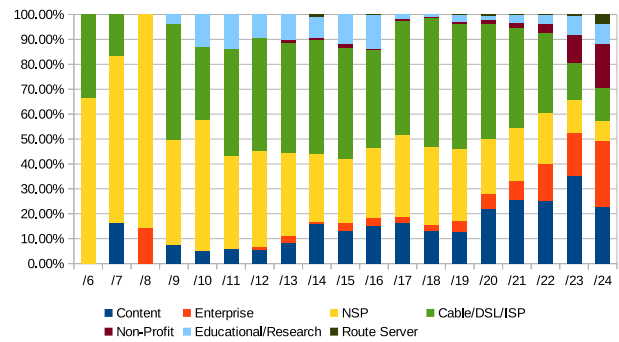


Figure 14: Distribution of networks' sizes vs types.

6 CONCLUSIONS

Much effort is invested to understand the extent of spoofability in the Internet. However, current measurement studies have limited applicability, providing results that apply to a small set of Internet networks.

Our work provides the first comprehensive view of ingress filtering in the Internet. We showed how to improve the coverage of the Internet in ingress filtering measurements to include many more ASes that were previously not studied. Our techniques allow to cover more than 90% of the Internet ASes, in contrast to best coverage so far of 7.5% of the ASes performed by the Spoofing Project. This coverage can be further extended to include 100% of the Internet's ASes by scanning the IPv4 range instead of opting for the dataset of [2], that we used in our study.

The most significant aspect of our methodologies is that they do not require coordination with the scanned networks. SMap can measure spoofability in any TCP/IP network with standard and widely supported services, such as Email and web. We integrated into SMap three techniques for testing ingress filtering: DNS-based,

IPID-based and PMTUD-based. Our experimental comparison of the effectiveness of the techniques demonstrated that DNS-based technique has a wider applicability rate on networks that operate DNS resolvers than the other two techniques, while the detection of the spoofability of networks is more accurate with PMTUD.

We set up SMap as a public service for continuous collection and analysis of the ingress filtering in the Internet at <https://smap.cad.sit.fraunhofer.de>.

ACKNOWLEDGMENTS

This work has been co-funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) SFB 1119.

REFERENCES

- [1] [n. d.]. MaxMind GeoLite2 Database. <https://dev.maxmind.com/geoip/geoip2/geoite2/>
- [2] [n. d.]. Rapid7 Labs Open Data. <https://opendata.rapid7.com/>
- [3] F. Baker and P. Savola. 2004. Ingress Filtering for Multihomed Networks. <http://tools.ietf.org/rfc/rfc3704.txt> RFC3704.
- [4] Paul Barford, Rob Nowak, Rebecca Willett, and Vinod Yegneswaran. 2006. Toward a model for source addresses of internet background radiation. In *Proc. of the Passive and Active Measurement Conference*.
- [5] Robert Beverly and Steven Bauer. 2005. The Spoofer project: Inferring the extent of source address filtering on the Internet. In *Usenix Sruti*, Vol. 5. 53–59.
- [6] Robert Beverly, Arthur Berger, Young Hyun, and K Claffy. 2009. Understanding the efficacy of deployed internet source address validation filtering. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. 356–369.
- [7] Robert Beverly, Ryan Koga, and KC Claffy. 2013. Initial longitudinal analysis of IP source spoofing capability on the Internet. *Internet Society* (2013), 313.
- [8] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. 2018. Domain validation++ for MitM-resilient PKI. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2060–2076.
- [9] Zesheng Chen, Chuanyi Ji, and Paul Barford. 2008. Spatial-temporal characteristics of internet malicious sources. In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE, 2306–2314.
- [10] Jakub Czyz, Michael Kallitsis, Manaf Gharaibeh, Christos Papadopoulos, Michael Bailey, and Manish Karir. 2014. Taming the 800 pound gorilla: The rise and decline of NTP DDoS attacks. In *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 435–448.
- [11] Tianxiang Dai, Philipp Jeltner, Haya Shulman, and Michael Waidner. 2021. The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources. In *30th USENIX Security Symposium (USENIX Security 21)*. 3147–3164.
- [12] Tianxiang Dai, Haya Shulman, and Michael Waidner. 2021. DNS-over-TCP considered vulnerable. In *ANRW '21: Applied Networking Research Workshop, Virtual Event, USA, July 24-30, 2021*. ACM, 76–81.
- [13] Tianxiang Dai, Haya Shulman, and Michael Waidner. 2021. Let's Downgrade Let's Encrypt. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. ACM.
- [14] Alberto Dainotti, Karyn Benson, Alistair King, KC Claffy, Michael Kallitsis, Eduard Glatz, and Xenofontas Dimitropoulos. 2013. Estimating internet address space usage through passive measurements. *ACM SIGCOMM Computer Communication Review* 44, 1 (2013), 42–49.
- [15] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, et al. 2014. The matter of heartbleed. In *Proceedings of the 2014 conference on internet measurement conference*. 475–488.
- [16] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2013. ZMap: Fast Internet-wide scanning and its security applications. In *Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13)*. 605–620.
- [17] Roya Ensafi, Jeffrey Knockel, Geoffrey Alexander, and Jedidiah R Crandall. 2014. Detecting intentional packet drops on the Internet via TCP/IP side channels. In *International Conference on Passive and Active Network Measurement*. Springer, 109–118.
- [18] Paul Ferguson. 2000. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. (2000).
- [19] P. Ferguson and D. Senie. 2000. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. <http://tools.ietf.org/rfc/rfc2827.txt> RFC2827.
- [20] Matthias Göhring, Haya Shulman, and Michael Waidner. 2018. Path MTU Discovery Considered Harmful. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 866–874.
- [21] Amir Herzberg and Haya Shulman. 2013. Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In *IEEE CNS 2013. The Conference on Communications and Network Security, Washington, D.C., U.S. IEEE*.
- [22] Gokay Huz, Steven Bauer, KC Claffy, and Robert Beverly. 2015. Experience in using mturk for network measurement. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Crowdsourcing and Crowdfunding of Big (Internet) Data*. 27–32.
- [23] Christopher A Kent and Jeffrey C Mogul. 1987. Fragmentation considered harmful. Vol. 17.
- [24] Maciej Korczyński, Yevheniya Nosyk, Qasim Lone, Marcin Skwarek, Baptiste Jonglez, and Andrzej Duda. 2020. The Closed Resolver Project: Measuring the Deployment of Source Address Validation of Inbound Traffic. *arXiv preprint arXiv:2006.05277* (2020).
- [25] Maciej Korczyński, Yevheniya Nosyk, Qasim Lone, Marcin Skwarek, Baptiste Jonglez, and Andrzej Duda. 2020. Don't forget to lock the front door! inferring the deployment of source address validation of inbound traffic. In *International Conference on Passive and Active Network Measurement*. Springer, 107–121.
- [26] Marc Kühner, Thomas Hupperich, Christian Rossow, and Thorsten Holz. 2014. Exit from Hell? Reducing the Impact of Amplification DDoS Attacks. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 111–125.
- [27] Franziska Lichtblau, Florian Streibelt, Thorben Krüger, Philipp Richter, and Anja Feldmann. 2017. Detection, classification, and analysis of inter-domain traffic with spoofed source IP addresses. In *Proceedings of the 2017 Internet Measurement Conference*. ACM, 86–99.
- [28] Qasim Lone, Matthew Luckie, Maciej Korczyński, Hadi Asghari, Mobin Javed, and Michel van Eeten. 2018. Using Crowdsourcing Marketplaces for Network Measurements: The Case of Spoofer. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 1–8.
- [29] Qasim Lone, Matthew Luckie, Maciej Korczyński, and Michel van Eeten. 2017. Using loops observed in traceroute to infer the ability to spoof. In *International Conference on Passive and Active Network Measurement*. Springer, 229–241.
- [30] Matthew Luckie, Robert Beverly, Ryan Koga, Ken Keys, Joshua A Kroll, and K claffy. 2019. Network Hygiene, Incentives, and Regulation: Deployment of Source Address Validation in the Internet. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 465–480.
- [31] Gordon Fyodor Lyon. 2009. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure.
- [32] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. 2020. DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS*. ACM.
- [33] Pietro Marchetta, Antonio Montieri, Valerio Persico, Antonio Pescapé, Ítalo Cunha, and Ethan Katz-Bassett. 2016. How and how much traceroute confuses our understanding of network paths. In *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 1–7.
- [34] Jared Mauch. 2013. Open resolver project. In *Presentation, DNS-OARC Spring 2013 Workshop (Dublin)*.
- [35] Rui Miao, Rahul Potharaju, Minlan Yu, and Navendu Jain. 2015. The dark menace: Characterizing network-based attacks in the cloud. In *Proceedings of the 2015 Internet Measurement Conference*. ACM, 169–182.
- [36] David Moore, Colleen Shannon, Douglas J Brown, Geoffrey M Voelker, and Stefan Savage. 2006. Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems (TOCS)* 24, 2 (2006), 115–139.
- [37] Paul Pearce, Roya Ensafi, Frank Li, Nick Feamster, and Vern Paxson. 2017. Augur: Internet-wide detection of connectivity disruptions. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 427–443.
- [38] Terrance A. Roebuck. 2005. Network security: DoS vs DDoS attacks. <http://www.crime-research.org/articles/network-security-dos-ddos-attacks/5>.
- [39] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse.. In *NDSS*.
- [40] Haya Shulman and Michael Waidner. 2014. Fragmentation considered leaking: port inference for dns poisoning. In *International Conference on Applied Cryptography and Network Security*. Springer, 531–548.
- [41] Stephen M Specht and Ruby B Lee. 2004. Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures.. In *ISCA PDCS*. 543–550.
- [42] J. Touch. 2013. Updated Specification of the IPv4 ID Field. <http://tools.ietf.org/rfc/rfc6864.txt> RFC6864.
- [43] Duane Wessels, Marina Fomenkov, et al. 2003. Wow, that's a lot of packets. In *Proceedings of Passive and Active Measurement Workshop (PAM)*.
- [44] Guang Yao, Jun Bi, and Athanasios V Vasilakos. 2014. Passive IP traceback: Disclosing the locations of IP spoofer from path backscatter. *IEEE Transactions on Information Forensics and Security* 10, 3 (2014), 471–484.