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 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

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

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 domains[1] 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

[1] Of 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.

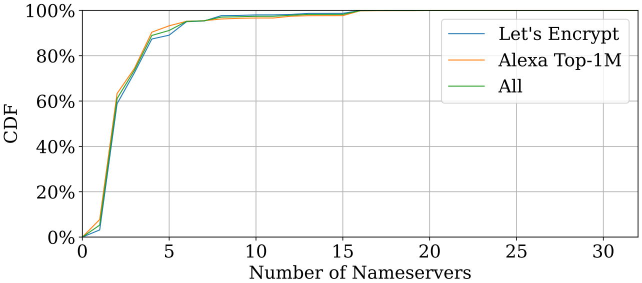


Figure 1: Number of nameservers per domain.

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 VAs[2] 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 Let's Encrypt described in Section 2.2.1 in a controlled environment using the DNS maze[3] 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 emulator[4] 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

[2] Let's Encrypt currently requires responses to only three out of four VAs for a DV and lookup to be successful.

[3] https://gitlab.nic.cz/knot/maze/

[4] NetEm `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] ($2 \cdot 376$ms 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 invocation[6], 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 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%

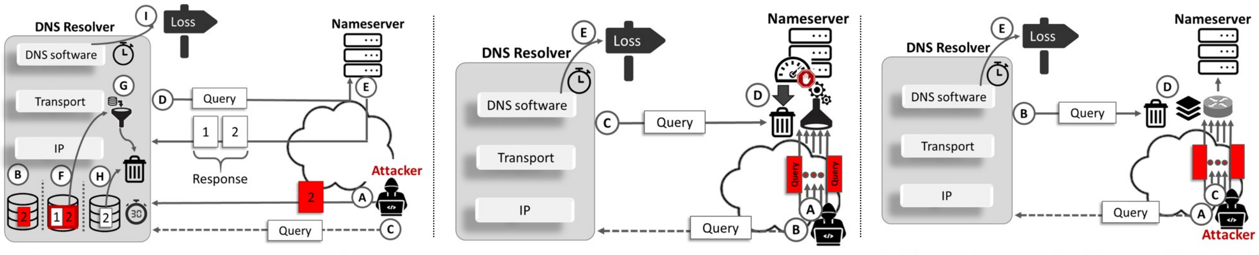
Table 2: Dataset of domains.

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.

[5] The RTO is the timeout including the exponential backoff. It is used for server selection and as a timeout for the transmitted request.

[6] Each 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.



(a) Loss via fragments mis-association.

(b) Loss via excess query rate.

(c) Loss via router buffer overflow.

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 Ⓐ 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 Ⓑ 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 Ⓒ we send a request for a certificate for the target domain. This causes the VA to initiate DNS lookup requests in step Ⓓ. For simplicity assume that the nameserver returns a response in two fragments, in step Ⓔ. In step Ⓕ 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 Ⓖ. In step Ⓗ the second genuine fragment enters the cache; after 30 seconds it is evicted if no matching first fragment arrives. In step Ⓘ 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 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.

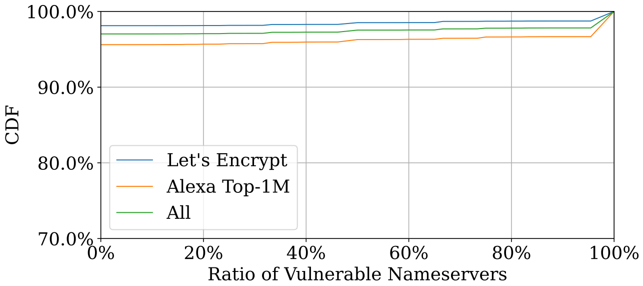


Figure 3: Nameservers per domain vulnerable to frag.

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 queries[7] 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 Ⓐ. The nameserver starts filtering the requests from the DNS resolver. In step Ⓑ the attacker requests a certificate of the target domain. In step Ⓒ the DNS resolver sends DNS query to the nameserver.

[7] Nameservers can apply rate limiting per IP address or overall independent of the IP address.

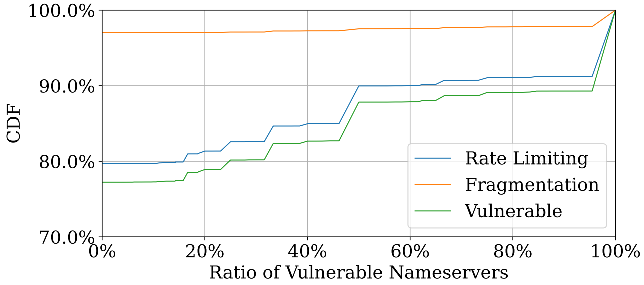


Figure 4: Nameservers per domain vulnerable to frag or rate-limiting.

The nameserver filters that request in step Ⓓ. After a timeout is reached, the loss event is registered in step Ⓔ, and the resolver retransmits 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.

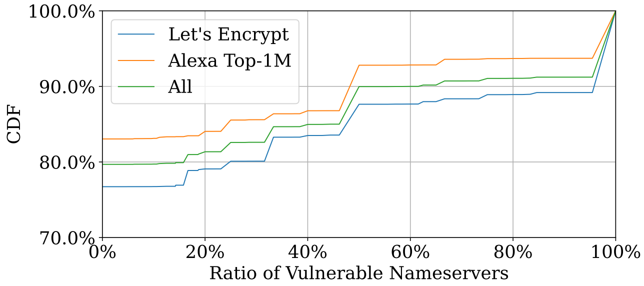


Figure 5: Nameservers per domain vulnerable to rate-limiting.

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 Ⓐ, 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 Ⓑ and Ⓒ. 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: $\Delta_{A-VA-NS} = \frac{2 \cdot \Delta_{A-VA-NS} - \epsilon}{2} = \Delta_{A-VA-NS} - \epsilon$, 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`.

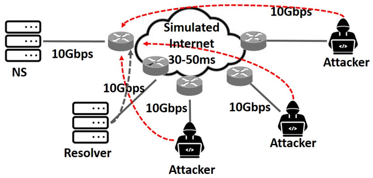


Figure 6: Simulated evaluation setup.

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.

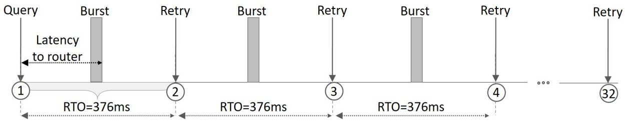


Figure 7: RTO timeline with low-rate bursts.

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	>$15 \cdot 10^3$ packets	92%
Cisco 6704	16MB	$18 \cdot 10^3$ 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 underutilisation. 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.

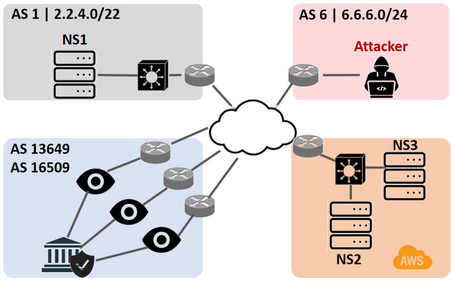
We then compare the security of Let's Encrypt to other popular CAs in the PKI ecosystem, in Section 4.4. We show that the downgrade attacks apply to all other CAs. Although, in contrast to Let's Encrypt, the other CAs do not guarantee security against MitM adversaries, our attack nevertheless makes it 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 6.6.6.0/24[8]. 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

[8] The network prefixes used in the paper are anonymised.

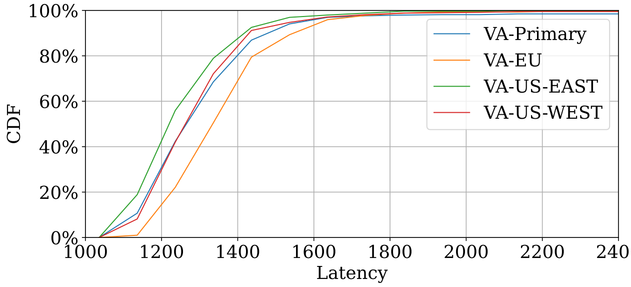


Figure 9: Obtaining fraudulent certs with each VA in Let's Encrypt **(ms).**

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 three[9] 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

[9] Let'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.

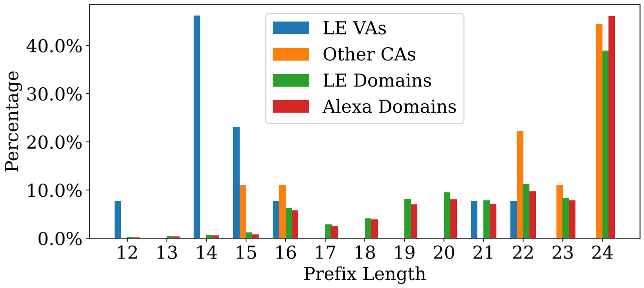


Figure 10: Network prefixes of CAs' resolvers and of domains' nameservers vulnerable to sub-prefix hijacks.

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 WoSign[12] tested control of any TCP port at the target

[12] https://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 administrators[13]. 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 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.

[13] https://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. DNSpionage Campaign Targets Middle East. https://blog.talosintelligence.com/2018/11/dnspionage-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 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 (Spectre) 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/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.

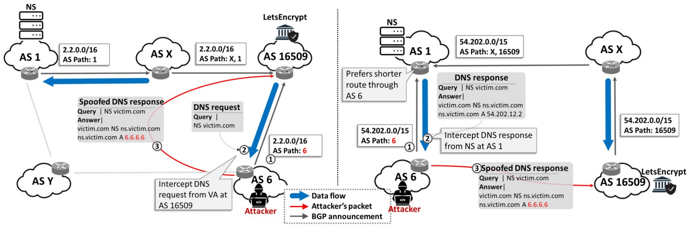
The victim AS is either a network hosting one or more of the nameservers of the target domain or the network hosting the VA. The target domain is the domain for which the attacker wishes to issue a fraudulent certificate. The target domain has nameservers which serve records from the DNS zone of that 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.

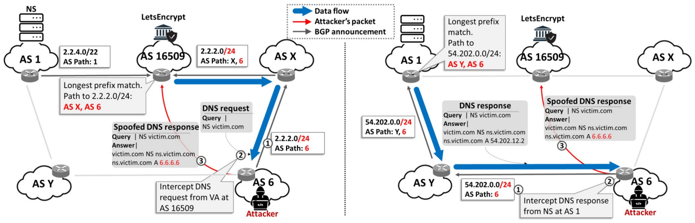
How effective are same-prefix attacks? The limitation of the same-prefix hijack is that it only affects the traffic of the ASes that prefer the attacker's announcement, and does not propagate to all parts of the Internet. Hence, the effectiveness of the same-prefix hijack attacks depends on the local 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.



(a) DNS request (b) DNS response

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



(a) DNS request

(b) DNS response

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.

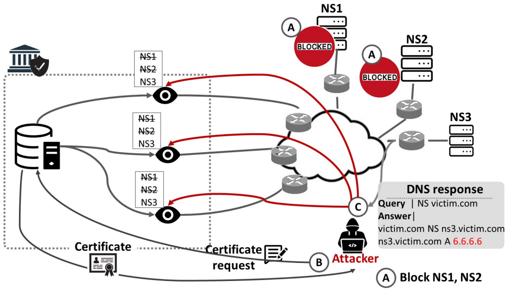


Figure 13: Attacking DV with MultiVA.

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 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 ID[15] 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

[14] Configures the device always to compare the Multi-Exit Discriminators (MEDs), regardless of the autonomous system (AS) information in the paths.

[15] A 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 Encryptcan 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-FastestRTO` <= 400ms) 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
     ENDIF
     request_sent_count++
     IF  request_sent_count >= MAX_SENT_COUNT
         return SERVFAIL
     ENDIF
 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	19,060	248,763	107,517	1,014,056
	23.27%	1.88%	24.53%	10.60%	
Alexa	145,280	37,624	179,242	100,709	856,887
	16.95%	4.39%	20.92%	11.75%	
Total	377,540	55,229	423,004	207,393	1,858,165
	20.32%	2.97%	22.76%	11.16%	

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%	5.53%	7.34%	33.40%	1.14%	100%
	51281	5388	7152	32560	1112	97493
TCP	14.43%	2.30%	75.92%	1.30%	6.04%	100%
	14072	2247	74020	1266	5888	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 the responses from the real server reach the attacker 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.