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

Tianxiang Dai
Fraunhofer SIT
Germany

Philipp Jeitner
Fraunhofer SIT
TU Darmstadt
Germany

Haya Shulman
Fraunhofer SIT
Germany

Michael Waidner
Fraunhofer SIT
TU Darmstadt
Germany

ABSTRACT

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

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

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

CCS CONCEPTS

• **Security and privacy** → **Network security**.

KEYWORDS

DNS Cache Poisoning, Fragmentation, BGP hijacking, Side Channels

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

1 INTRODUCTION

Domain Name System (DNS), [RFC1034, RFC1035] [59, 60], plays a central role in the Internet. Designed and standardised in the 80s to provide lookup services DNS has evolved into a complex infrastructure and is being increasingly used to support a wide variety of existing and future applications and security mechanisms. Given the large dependency of the Internet on DNS it also became a lucrative target for attacks.

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

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

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

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

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

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

What applications are at risk due to cache poisoning? How can an attacker exploit cache poisoning to attack applications? What is *the fraction of vulnerable applications in the Internet? What are the challenges and what cache poisoning methodologies are more suitable?*

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

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

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

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

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

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

Disclosure and ethics. Our attacks were tested against remote networks reliably, yet were ethically compliant. We measured and evaluated vulnerabilities in the DNS caches of the subjects of our study and measured which services use the caches but did not hijack their traffic nor Internet resources and neither did we place incorrect DNS records for Internet domains that are not under our control in the caches of our test subjects. Specifically, to avoid harming Internet customers and domains, we set up a victim AS and victim domains as well as adversarial AS and adversarial hosts on that AS, which were used by us for carrying out the attacks against the victims. Our measurement study for evaluating the vulnerabilities was performed using our victim domains, which ensured that the targets of our study would not use the spoofed records for any "real" purpose.

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

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

• We implement three methodologies for launching off-path DNS cache poisoning attacks: (1) BGP prefix hijacking, (2) side-channels and (3) fragmentation. We perform the first Internet-scale evaluation of these methodologies against DNS resolvers and compare them for applicability, stealthiness and success of cache poisoning.

• We apply these methodologies to launch *cross-layer attacks* against widely used applications and services (see taxonomy in Table 1). Our study shows that cache poisoning can be used to bypass security mechanisms, to cause DoS attacks, or for impersonation attacks.

• We provide recommendations for countermeasures for DNS caches against cache poisoning attacks and for applications against cross-layer attacks even when using poisoned caches.

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

2 DNS CACHE POISONING OVERVIEW

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

2.1 Defences Against Poisoning

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

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

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

2.2 History of DNS Cache Poisoning

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

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

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

2.3 DNS Cache Poisoning in the Wild

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

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

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

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

3 TAXONOMY OF POISONING METHODS

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

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

3.1 Intercepting DNS with BGP Hijacking

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

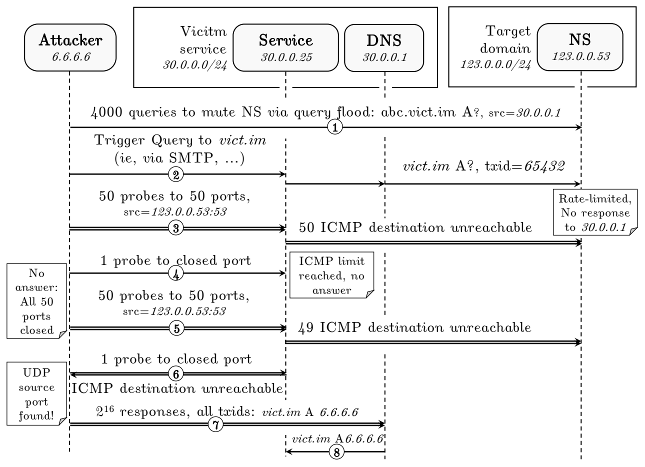

Attacker 6.6.6.6
Vicitm service 30.0.0.0/24
Service 30.0.0.25
DNS 30.0.0.1
Target domain 123.0.0.0/24
NS 123.0.0.53
4000 queries to mute NS via query flood: abc.vict.im A?, src=30.0.0.1
Trigger Query to vict.im (ie, via SMTP, ...)
vict.im A?, txid=65432
50 probes to 50 ports, src=123.0.0.53:53
Rate-limited, No response to 30.0.0.1
50 ICMP destination unreachable
1 probe to closed port
ICMP limit reached, no answer
No answer: All 50 ports closed
50 probes to 50 ports, src=123.0.0.53:53
49 ICMP destination unreachable
1 probe to closed port
ICMP destination unreachable
UDP source port found!
2^16 responses, all txids: vict.im A 6.6.6.6
vict.im A 6.6.6.6

Figure 1: DNS Poisoning with side-channel.

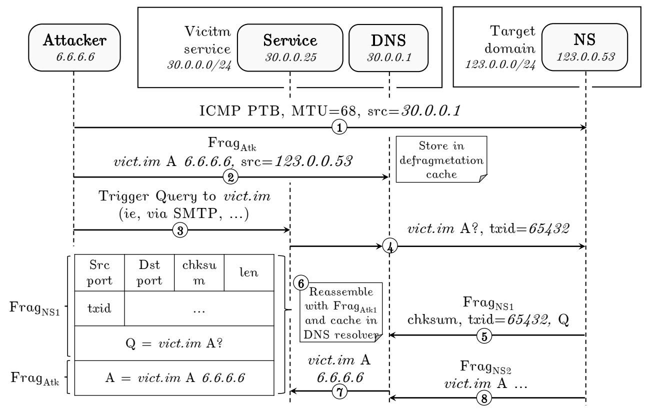

Attacker 6.6.6.6
Vicitm service 30.0.0.0/24
Service 30.0.0.25
DNS 30.0.0.1
Target domain 123.0.0.0/24
NS 123.0.0.53
ICMP PTB, MTU=68, src=30.0.0.1
FragAtk vict.im A 6.6.6.6, src=123.0.0.53
Store in defragmetation cache
Trigger Query to vict.im (ie, via SMTP, ...)
vict.im A?, txid=65432
Src port
Dst port
chksum
len
txid
...
Q = vict.im A?
A = vict.im A 6.6.6.6
FragNS1
FragAtk
Reassemble with FragAtk1 and cache in DNS resolver
FragNS1 chksum, txid=65432, Q
vict.im A 6.6.6.6
FragNS2 vict.im A ...

Figure 2: Fragmentation-based DNS poisoning.

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

3.2 Guessing Challenges with Side-channel

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

Once the open port is identified the attacker sends multiple spoofed DNS responses from a spoofed IP address (of the nameserver) to that open UDP port of the resolver, for each possible TXID value, total of 2^{16} spoofed responses; e.g., [37, 45, 57]. A packet with the correct TXID is accepted by the DNS resolver. The attack is illustrated in Figure 1. The attack applies to only about 18% of the domains with nameservers that use rate-limiting. The rate limiting allows the adversary to delay the response from the genuine nameserver and hence to win the race against it. Additionally, the attack applies only against resolvers with a global (un-patched) ICMP rate limit.

3.3 Injecting Records via IP Fragmentation

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

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

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

4 EXPLOITING DNS POISONING FOR CROSS-LAYER ATTACKS

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

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

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

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

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

4.1 Methodology for Selecting Applications

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

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

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

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

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

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

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

Some applications enable the attacker to send arbitrary queries, e.g., in systems which use DNS for peer discovery in federated systems like Radius, XMPP and SMTP. This is because in these systems, the queried domains are part of the user's ID. This user ID can be controlled by the attacker to trigger a query to a domain of its choice. The same applies to all (sub-)systems used as part of web browsing, like HTTP, DANE and OCSP, since the attacker can establish direct connection from the victim client to arbitrary web servers which will trigger a DNS lookup that way. Setting the domain name is not always possible, e.g., in NTP the query is selected by the resolver based on the hostname that it receives from the local NTP server.

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

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

4.2 Methodology for Attacking Applications

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

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

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

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

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

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

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

4.3 Methodologies for Triggering Queries

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

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

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

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

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

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

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

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

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

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

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

4.4 Applicability to Applications

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

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

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

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

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

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

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

4.5 Exploiting Poisoned Caches for Attacks

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

Downgrade attacks. In downgrade attacks the attacker makes the security mechanism not available, as a result, causing the processing of the data to be performed without the additional information provided by the security mechanism. For instance, by poisoning the responses to queries for SPF or DKIM records the attacker can trick the victim Email server into accepting phishing Emails or Emails with malicious attachments. Similarly, by causing the RPKI validation to fail, the adversary can make a network, that filters bogus BGP announcements with route origin validation, to accept hijacked prefixes as authentic. This is due to the fact that RPKI validation will result in status 'unknown' and hence will not be used.

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

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

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

5 INTERNET MEASUREMENTS

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

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

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

5.1 Vulnerabilities in Resolvers

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

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

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

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

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

*Sub-prefix BGP hijacks (*HijackDNS*).* Since many networks filter BGP advertisements with prefixes more specific than /24, we consider an IP address hijackable if it lies inside a network block whose advertised size is larger than /24. We therefore map all the resolvers' IP addresses to network blocks and consider those vulnerable to sub-prefix hijacks whose advertisement is larger than /24, since an advertisement with a smaller prefix will always take precedence over a bigger one. For the remaining addresses, a BGP-hijack may still be possible using same-prefix hijacks. To infer the scope of DNS platforms potentially vulnerable to cache poisoning via BGP sub-prefix hijack attacks we perform Internet measurements checking for DNS platforms on prefixes less than /24. We collect information on the state of the global BGP table in the Internet with Routeview [71] and RIPE RIS [63] collectors. We analyse the BGP announcements seen in public collectors for identifying networks vulnerable to sub-prefix hijacks by studying the advertised prefixes sizes. The measurements of resolvers vulnerable to BGP sub-prefix hijacks are listed in Table 3 and plotted in Figure 3.

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

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

For the open resolver dataset we measured a vulnerability rate of 12%, a notable reduction from the 35% vulnerability rate of the original paper [57]. This difference could be influenced by various factors including the fact that our dataset contained more resolvers than [57]. The crucial difference is likely that our study was conducted after the vulnerability which allowed the global rate limit to be exploited was patched in many systems. For example, all updated versions of Ubuntu should have been patched by the time we carried out our evaluations[1].

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

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

[1]https://ubuntu.com/security/CVE-2020-25705

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

Table 3: Vulnerable resolvers.

5.2 Vulnerabilities in Domains

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

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

Table 4: Vulnerable domains.

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

(1) Eduroam institution lists from United Kingdom [46], Germany [30] and Austria [15]. (2) Alexa Top 1 Million domains, including subsets of domains which have (3) MX and (4) SRV (XMPP) records. Domains from account email addresses from whois databases of (5) RIRs and (6) Registrars. (7) Well-known NTP server domains. (8) Well-known cryptocurrency domains. (9) Well-known RPKI validator database domains. (10) Domains of IKE and OpenVPN servers' certificates. This resulted in 904,555 domains hosted on 200,086 nameservers in 24,353 ASes associated with 60,511 prefixes.

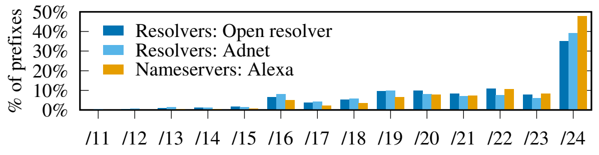


Figure 3: Announced prefixes.

5.2.2 Measuring cache poisoning vulnerabilities in nameservers.

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

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

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

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

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

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

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

Table 5: ANY caching results of popular resolvers.

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

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

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

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

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

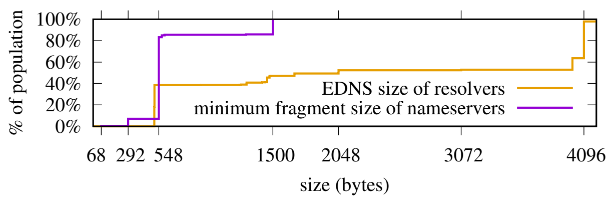
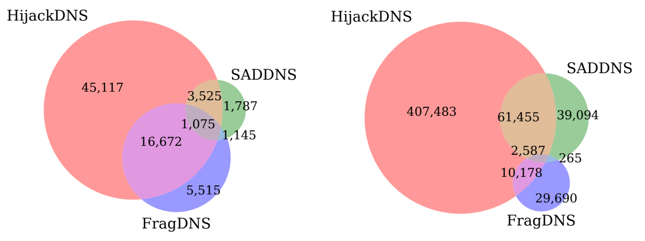


Figure 4: CDF of resolver EDNS UDP size vs. minimum fragment size emitted by nameservers.



(a) Resolver **(b) Domain**

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

5.3 Comparative Analysis

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

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

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

Table 6: Comparison of the cache poisoning methods.

5.3.1 *Applicability.*

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

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

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

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

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

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

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

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

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

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

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

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

6 COUNTERMEASURES

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

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

6.1 DNS servers

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

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

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

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

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

Securing BGP. Full deployment of RPKI (together with BGPSec) would prevent the HijackDNS attack. However, because of several deployment barriers, most of the prefixes are not protected by RPKI and most ASes do not enforce Route Origin Validation (ROV) [36, 40, 62]. We refer to [39] for a comprehensive discussion of the deployment issues.

6.2 Applications

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

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

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

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

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

7 CONCLUSIONS

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

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

ACKNOWLEDGEMENTS

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

REFERENCES

[1] [n.d.]. Best Infrastructure as a Service (IaaS) Providers. https://www.g2.com/categories/infrastructure-as-a-service-iaas. Accessed: 2020-10-09.

[2] [n.d.]. Market Share Analysis: IaaS and IUS, Worldwide, 2018. https://www.gartner.com/en/newsroom/press-releases/2019-07-29-gartner-says-worldwide-iaas-public-cloud-services-market-grew-31point3-percent-in-2018. Accessed: 2020-10-09.

[3] [n.d.]. Market share trends for SSL certificate authorities. https://w3techs.com/technologies/history_overview/ssl_certificate. Accessed: 2020-10-09.

[4] [n.d.]. Quarterly Cloud Spending Blows Past $30B; Incremental Growth Continues to Rise. https://www.srgresearch.com/articles/quarterly-cloud-spending-blows-past-30b-incremental-growth-continues-rise. Accessed: 2020-10-09.

[5] [n.d.]. Top IaaS Providers: 42 Leading Infrastructure-as-a-Service Providers to Streamline Your Operations. https://stackify.com/top-iaas-providers/. Accessed: 2020-10-09.

[6] 2011. The CAIDA AS Relationships Dataset, 2011. http://www.caida.org/data/active/as-relationships/.

[7] 2015. Hacked or Spoofed: Digging into the Malaysia Airlines Website Incident. https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/hacked-or-spoofed-digging-into-the-malaysia-airlines-website-compromise. Accessed: 2021-1-19.

[8] 2015. Webnic Registrar Blamed for Hijack of Lenovo, Google Domains. https://krebsonsecurity.com/2015/02/webnic-registrar-blamed-for-hijack-of-lenovo-google-domains/. Accessed: 2021-1-19.

[9] 2018. DNSpionage Campaign Targets Middle East. https://blog.talosintelligence.com/2018/11/dnspionage-campaign-targets-middle-east.html. Accessed: 2021-01-19.

[10] 2019. Global DNS Hijacking Campaign: DNS Record Manipulation at Scale. https://www.fireeye.com/blog/threat-research/2019/01/global-dns-hijacking-campaign-dns-record-manipulation-at-scale.html. Accessed: 2021-1-19.

[11] 2019. Sea Turtle keeps on swimming, finds new victims, DNS hijacking techniques. https://blog.talosintelligence.com/2019/07/sea-turtle-keeps-on-swimming.html. Accessed: 2021-01-19.

[12] 2019. 'Unprecedented' DNS Hijacking Attacks Linked to Iran. https://threatpost.com/unprecedented-dns-hijacking-attacks-linked-to-iran/140737/

[13] 2020. Security Incident on November 13, 2020. https://blog.liquid.com/security-incident-november-13-2020. Accessed: 2021-01-19.

[14] 2021. The Hijackers Guide To The Galaxy: Off-Path Taking Over Internet Resources. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association. https://www.usenix.org/conference/usenixsecurity21/presentation/dai

[15] aconet. [n.d.]. eduroam-Teilnehmer in Österreich. https://www.aco.net/eduroam_teilnehmer.html. Accessed: 2020-12-02.

[16] F. Alharbi, J. Chang, Y. Zhou, F. Qian, Z. Qian, and N. Abu-Ghazaleh. 2019. Collaborative Client-Side DNS Cache Poisoning Attack. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 1153–1161. https://doi.org/10.1109/INFOCOM.2019.8737514

[17] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. 2017. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 375–392.

[18] Dan J. Bernstein. 2002. DNS Forgery. Internet publication at http://cr.yp.to/djbdns/forgery.html.

[19] Robert Beverly and Steven Bauer. 2005. The Spoofer project: Inferring the extent of source address filtering on the Internet. In *Usenix Sruti*, Vol. 5. 53–59.

[20] Robert Beverly, Arthur Berger, Young Hyun, and K Claffy. 2009. Understanding the efficacy of deployed internet source address validation filtering. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. 356–369.

[21] Robert Beverly, Ryan Koga, and KC Claffy. 2013. Initial longitudinal analysis of IP source spoofing capability on the Internet. *Internet Society* (2013), 313.

[22] Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. 2018. Bamboozling Certificate Authorities with BGP. In *27th USENIX Security Symposium (USENIX Security 18)*. 833–849.

[23] Peter Boothe, James Hiebert, and Randy Bush. 2006. Short-lived prefix hijacking on the Internet. *In Proc. of the NANOG* 36 (2006).

[24] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. 2018. Domain Validation++ For MitM-Resilient PKI. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2060–2076.

[25] R Bush and R Austein. 2013. RFC 6810: The Resource Public Key Infrastructure (RPKI) to Router Protocol.

[26] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. A Longitudinal, End-to-End View of the DNSSEC Ecosystem. In *26th USENIX Security Symposium (USENIX Security 17)*. 1307–1322.

[27] Taejoong Chung, Roland van Rijswijk-Deij, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. Understanding the role of registrars in DNSSEC deployment. In *Proceedings of the 2017 Internet Measurement Conference*. 369–383.

[28] D. Madory. 2018. Recent Routing Incidents: Using BGP to Hijack DNS and more. https://www.lacnic.net/innovaportal/file/3207/1/dougmadory_lacnic_30_rosario.pdf

[29] David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. 2008. Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries. In *ACM Conference on Computer and Communications Security*, Peng Ning, Paul F. Syverson, and Somesh Jha (Eds.). ACM, 211–222. http://doi.acm.org/10.1145/1455770.1455798

[30] DFN-Verein. [n.d.]. Karte der aktuellen eduroam Standorte in Deutschland. https://www.dfn.de/dienstleistungen/eduroam/. Accessed: 2020-12-02.

[31] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. *Tor: The second-generation onion router.* Technical Report. Naval Research Lab Washington DC.

[32] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. 2015. A Search Engine Backed by Internet-Wide Scanning. In *22nd ACM Conference on Computer and Communications Security.*

[33] Tony Finch, Evan Hunt, Peter van Dijk, Anthony Eden, and Willem Mekking. 2019. *Address-specific DNS aliases (ANAME).* Internet-Draft draft-ietf-dnsop-aname-03. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-ietf-dnsop-aname-03.txt.

[34] Pedro Franco. 2014. *Understanding bitcoin.* Wiley Online Library.

[35] Lixin Gao and Jennifer Rexford. 2001. Stable Internet routing without global coordination. *IEEE/ACM Transactions on networking* 9, 6 (2001), 681–692.

[36] Yossi Gilad, Avichai Cohen, Amir Herzberg, Michael Schapira, and Haya Shulman. 2017. Are We There Yet? On RPKI's Deployment and Security.. In *NDSS.*

[37] Amir Herzberg and Haya Shulman. 2012. Security of Patched DNS. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings.* 271–288.

[38] Amir Herzberg and Haya Shulman. 2013. Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In *IEEE CNS 2013. The Conference on Communications and Network Security, Washington, D.C., U.S.* IEEE.

[39] Tomas Hlavacek, Italo Cunha, Yossi Gilad, Amir Herzberg, Ethan Katz-Bassett, Michael Schapira, and Haya Shulman. 2020. DISCO: Sidestepping RPKI's Deployment Barriers. In *Network and Distributed System Security Symposium (NDSS).*

[40] Tomas Hlavacek, Amir Herzberg, Haya Shulman, and Michael Waidner. 2018. Practical Experience: Methodologies for Measuring Route Origin Validation. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018.* 634–641. https://doi.org/10.1109/DSN.2018.00070

[41] P Hoffman and P McManus. 2018. RFC 8484: DNS Queries over HTTPS (DoH).

[42] Z Hu, L Zhu, J Heidemann, A Mankin, D Wessels, and P Hoffman. 2016. RFC 7858-Specification for DNS over Transport Layer Security (TLS).

[43] A. Hubert and R. van Mook. 2009. *Measures for Making DNS More Resilient against Forged Answers.* RFC 5452. RFC Editor.

[44] Philipp Jeitner and Haya Shulman. 2021. Injection Attacks Reloaded: Tunnelling Malicious Payloads over DNS. In *30th USENIX Security Symposium (USENIX Security 21).*

[45] Philipp Jeitner, Haya Shulman, and Michael Waidner. 2020. The Impact of DNS Insecurity on Time. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).* IEEE, 266–277.

[46] Jisc. [n.d.]. Organisations participating in eduroam in the UK. https://www.jisc.ac.uk/eduroam/participating-organisations. Accessed: 2020-12-02.

[47] Dan Kaminsky. 2008. It's the End of the Cache As We Know It. Presentation at Blackhat Briefings.

[48] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. 2008. Autonomous security for autonomous systems. *Computer Networks* 52, 15 (2008), 2908–2923.

[49] Varun Khare, Qing Ju, and Beichuan Zhang. 2012. Concurrent prefix hijacks: Occurrence and impacts. In *Proceedings of the 2012 Internet Measurement Conference.* ACM, 29–36.

[50] Amit Klein. 2007. BIND 9 DNS cache poisoning. *Report, Trusteer, Ltd* 3 (2007).

[51] Amit Klein. 2007. Windows DNS Server Cache Poisoning,".

[52] Amit Klein. 2020. Cross Layer Attacks and How to Use Them (for DNS Cache Poisoning, Device Tracking and More). *arXiv preprint arXiv:2012.07432* (2020).

[53] Amit Klein, Haya Shulman, and Michael Waidner. 2017. Internet-wide study of DNS cache injections. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications.* IEEE, 1–9.

[54] Matt Lepinski, Stephen Kent, and Derrick Kong. 2012. A profile for route origin authorizations (ROAs). *IETF, RFC* 6482 (2012).

[55] Qasim Lone, Matthew Luckie, Maciej Korczyński, Hadi Asghari, Mobin Javed, and Michel van Eeten. 2018. Using Crowdsourcing Marketplaces for Network Measurements: The Case of Spoofer. In *2018 Network Traffic Measurement and Analysis Conference (TMA).* IEEE, 1–8.

[56] Matthew Luckie, Robert Beverly, Ryan Koga, Ken Keys, Joshua A Kroll, and k claffy. 2019. Network Hygiene, Incentives, and Regulation: Deployment of Source Address Validation in the Internet. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.* 465–480.

[57] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. 2020. DNS Cache Poisoning Attack Reloaded: Revolutions with

Side Channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1337–1350. https://doi.org/10.1145/3372297.3417280

[58] Jared Mauch. 2013. Open resolver project. In *Presentation, DNS-OARC Spring 2013 Workshop (Dublin)*.

[59] P. Mockapetris. 1987. *Domain names - concepts and facilities*. STD 13. RFC Editor. http://www.rfc-editor.org/rfc/rfc1034.txt http://www.rfc-editor.org/rfc/rfc1034.txt.

[60] P. Mockapetris. 1987. *Domain names - implementation and specification*. STD 13. RFC Editor. http://www.rfc-editor.org/rfc/rfc1035.txt http://www.rfc-editor.org/rfc/rfc1035.txt.

[61] Pradosh Mohapatra, John Scudder, David Ward, Randy Bush, and Rob Austein. 2013. BGP prefix origin validation. In *IETF RFC 6811*.

[62] Andreas Reuter, Randy Bush, Ítalo Cunha, Ethan Katz-Bassett, Thomas C. Schmidt, and Matthias Wählisch. 2017. Towards a Rigorous Methodology for Measuring Adoption of RPKI Route Validation and Filtering. *CoRR* abs/1706.04263 (2017). arXiv:1706.04263 http://arxiv.org/abs/1706.04263

[63] RIPE NCC. 2021. RIS Raw Data. https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-raw-data

[64] S. Goldberg. 2018. The myetherwallet.com hijack and why it's risky to hold cryptocurrency in a webapp. https://medium.com/@goldbe/the-myetherwallet-com-hijack-and-why-its-risky-to-hold-cryptocurrency-in-a-webapp-261131fad278

[65] Haya Shulman. 2014. Pretty bad privacy: Pitfalls of DNS encryption. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. 191–200.

[66] Haya Shulman and Michael Waidner. 2015. Towards security of internet naming infrastructure. In *European Symposium on Research in Computer Security*. Springer, 3–22.

[67] Haya Shulman and Michael Waidner. 2017. One Key to Sign Them All Considered Vulnerable: Evaluation of DNSSEC in the Internet.. In *NSDI*. 131–144.

[68] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. 2019. Encrypted DNS–> Privacy? A Traffic Analysis Perspective. *arXiv preprint arXiv:1906.09682* (2019).

[69] Jonathan Spring. [n.d.]. Probable Cache Poisoning of Mail Handling Domains. https://insights.sei.cmu.edu/cert/2014/09/-probable-cache-poisoning-of-mail-handling-domains.html.

[70] Yixin Sun, Maria Apostolaki, Henry Birge-Lee, Laurent Vanbever, Jennifer Rexford, Mung Chiang, and Prateek Mittal. 2021. Securing internet applications from routing attacks. *Commun. ACM* 64, 6 (2021), 86–96.

[71] University of Oregon. 2012. Route Views Project. http://bgplay.routeviews.org/.

[72] Paul Vixie. 1995. DNS and BIND Security Issues. In *Proceedings of the 5th Symposium on UNIX Security*. USENIX Association, Berkeley, CA, USA, 209–216.

[73] S Weiler and D Blacka. 2013. RFC 6840: Clarifications and Implementation Notes for DNS Security (DNSSEC). *IETF Standard* (2013).

[74] Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan, and Zhiyun Qian. 2020. Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices. In *29th USENIX Security Symposium (USENIX Security 20)*. 577–593.