TuDoor Attack:

Systematically Exploring and Exploiting Logic Vulnerabilities in DNS Response Pre-processing with Malformed Packets

[Published at IEEE S&P 2024]

Presenter: **Xiang Li**

Tsinghua University

Attack Impact

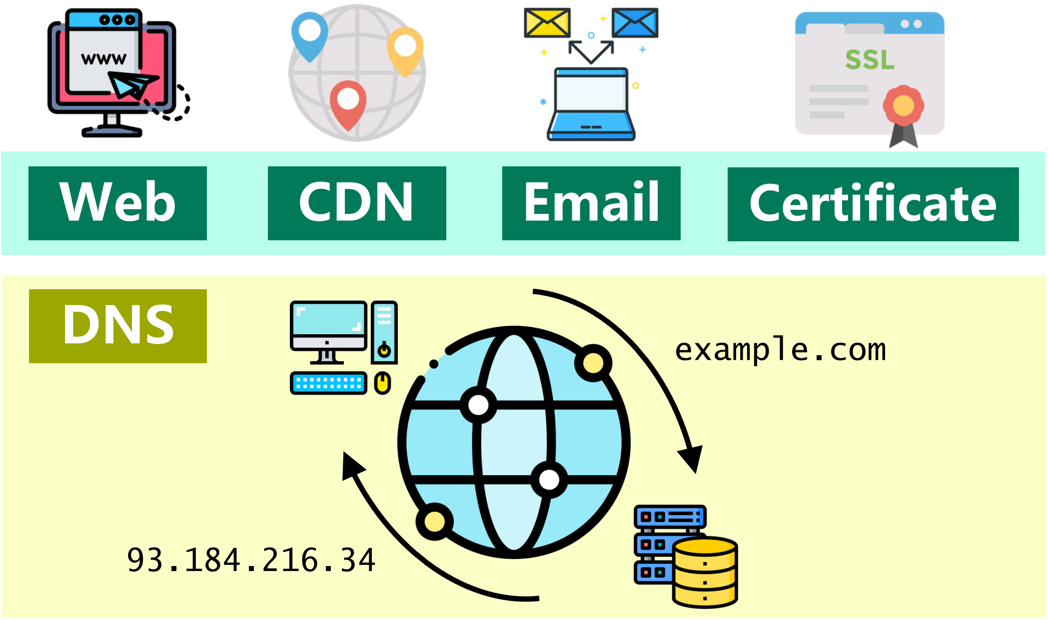
Our TuDoor attack could poison arbitrary domains, e.g., .com and .net.

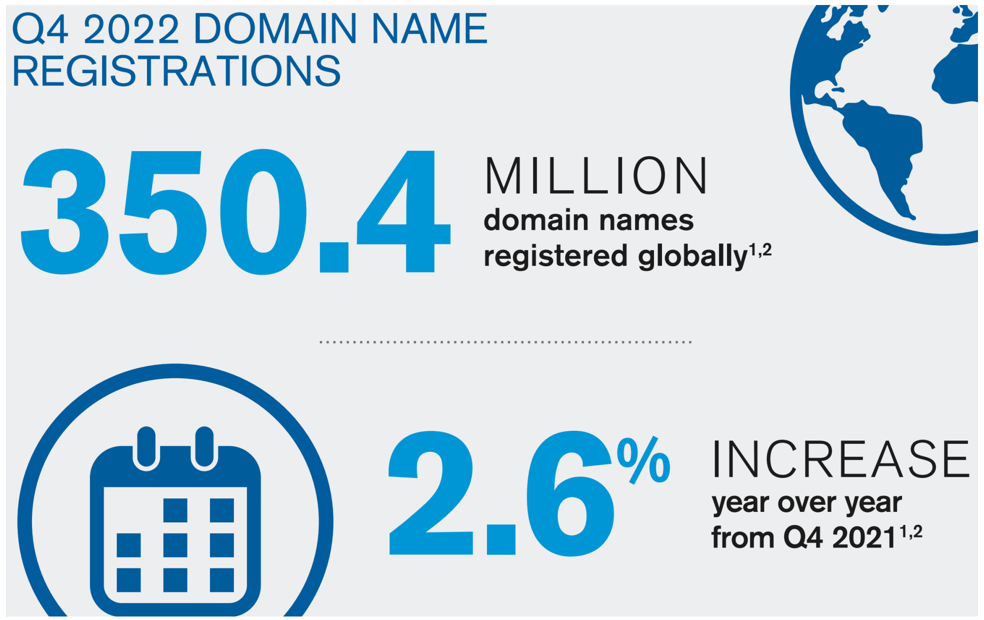
Poisoning vulnerable resolvers' cache within just one second.

Domain Name System (DNS)

➢ DNS Overview

- ❑ Translating domain names to IP addresses
- ❑ Entry point of many Internet activities
- ❑ Domain names are widely registered

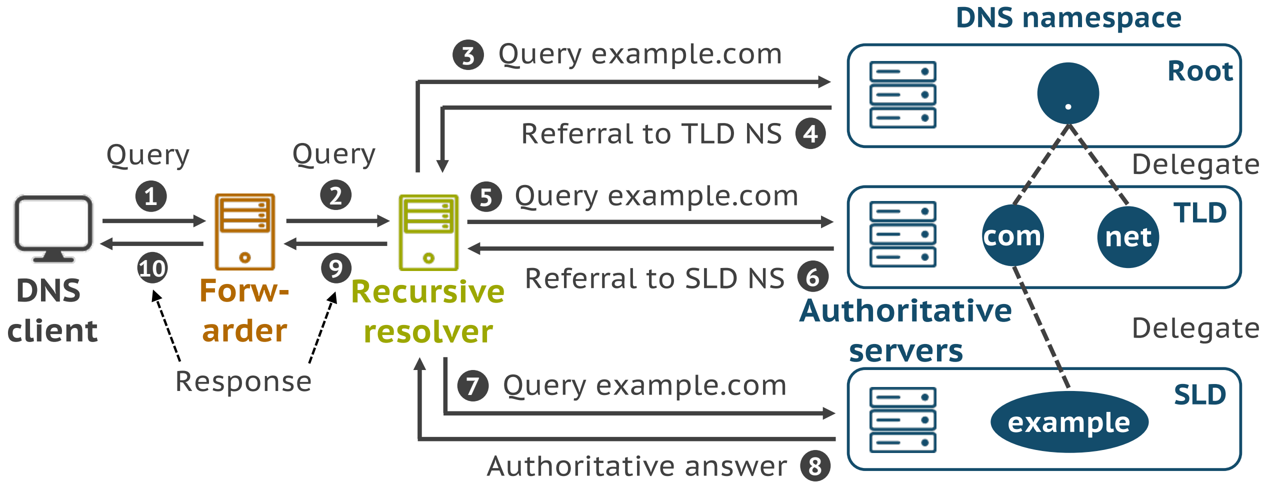




verisign.com/dnib

Domain Name System (DNS)

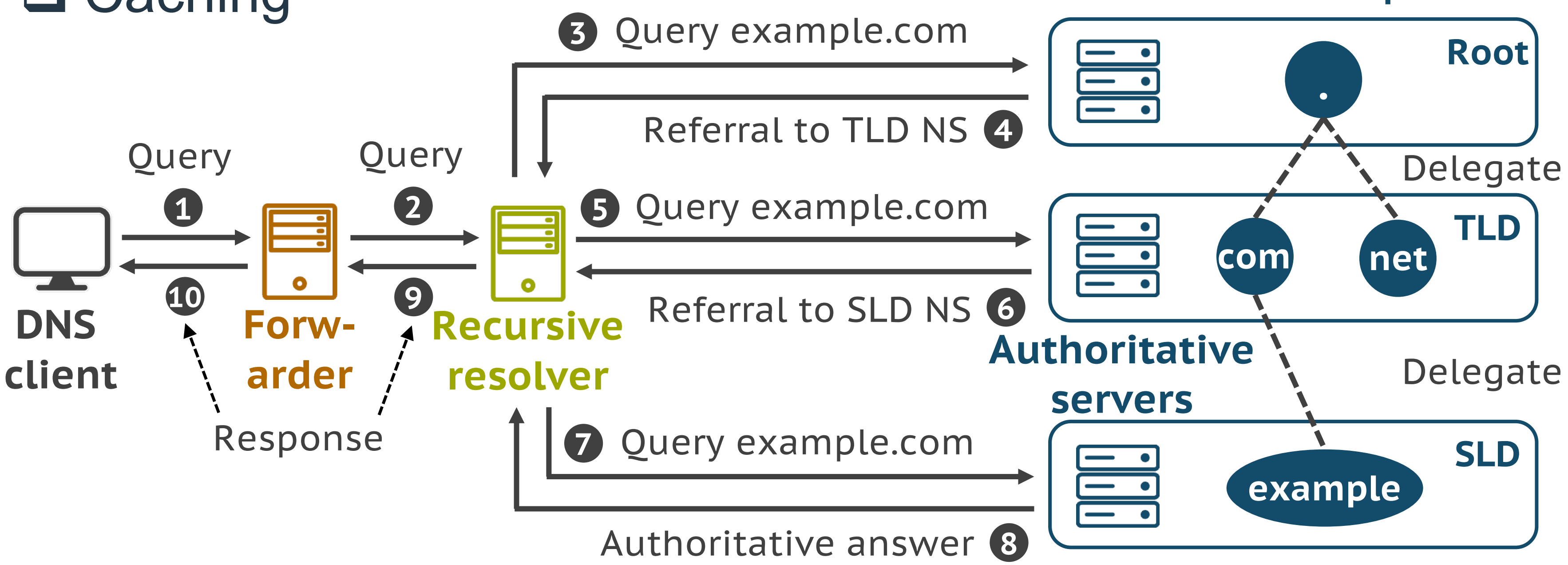
- **Hierarchical Name Space**
  - Authoritative zones: root, TLD, SLD → DNS records
  - Domain delegation → Domain registration
- **Multiple Resolver Roles**
  - Client, forwarder, recursive, authoritative
  - Caching
- **Iterative Resolution Process**
  - Client-server style

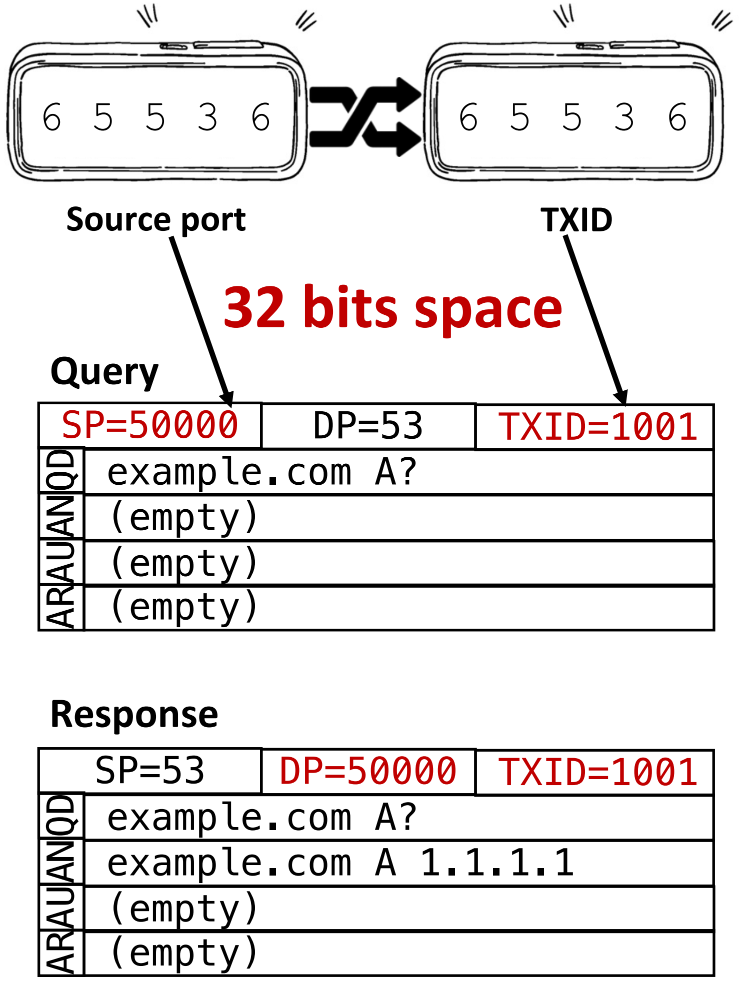


Domain Name System (DNS)

➢ DNS Resolution Process

- ❑ Primarily over UDP
- ❑ Iterative and recursive
- ❑ Caching





Takeaway

Since DNS is the cornerstone of the Internet, enabling multiple critical services and applications,

Attackers have long been trying to manipulate its response for hijacking via **cache poisoning attacks**.

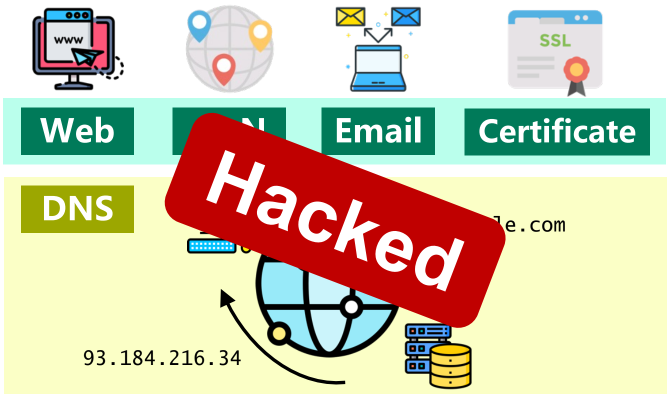
Question

What is DNS cache poisoning?

Since DNS is primarily over UDP, attackers want to **inject forged answers into resolvers' cache**.

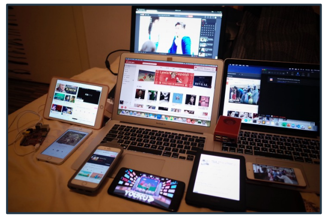
DNS Cache Poisoning

- **Target**
  - ❑ Injecting forged answers into resolvers' cache
- **Taxonomy**
  - ❑ On-path, off-path
- **Technique**
  - ❑ Cat-and-mouse game

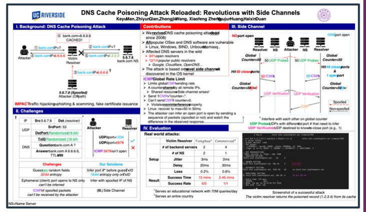


1997 Birthday Attack
2002 Kashpureff Attack
2008 Fragmentation Attack
2013 Kaminsky Attack
2020 SADDNS Attack

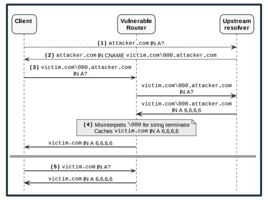
2020 Attack on Forwarders



2021 SADDNS v2 Attack



2021 Attack via Escaped Chars

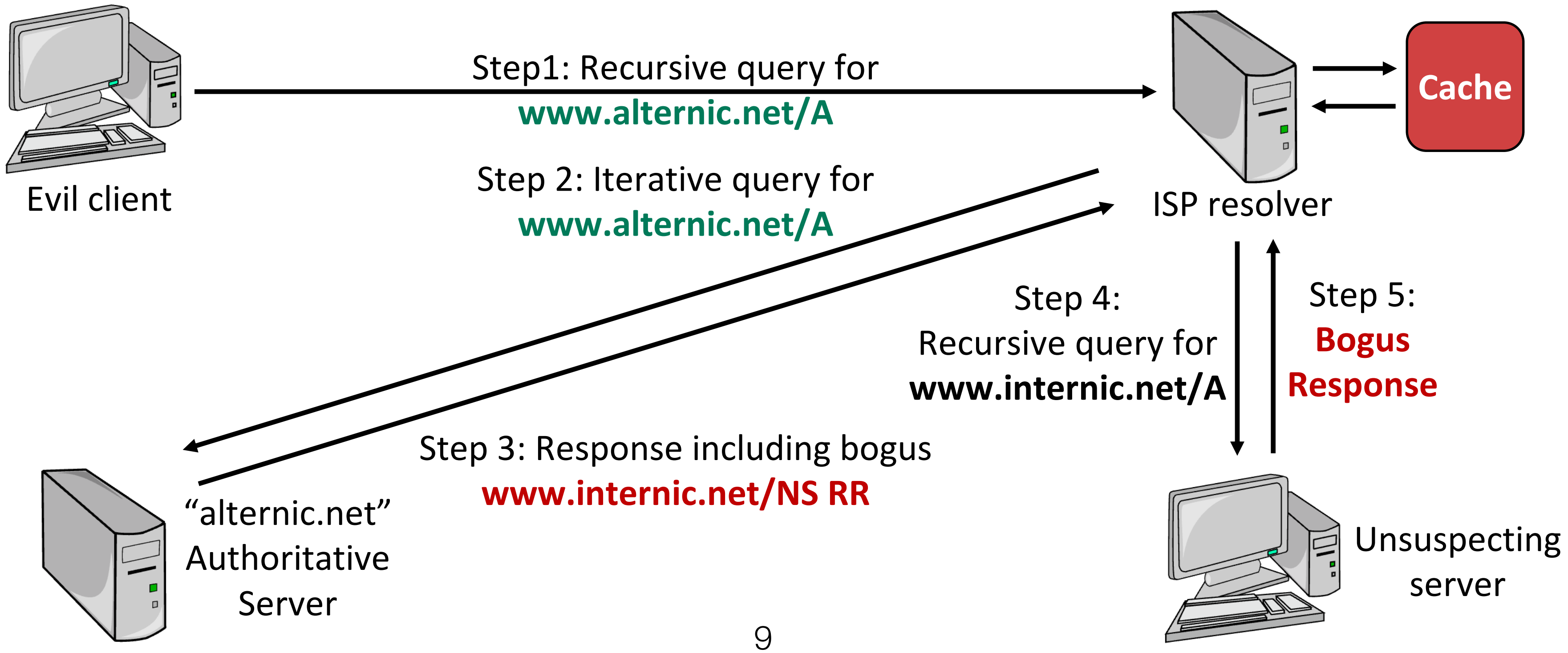


2022 MaginotDNS Attack
DNS
MAGINOT

2023 Attack via Escaped Chars v2

DNS Cache Poisoning (1/5)

- **Kashpureff Attack (on-path, 1997)**
  - **Method:** returning forged responses from the authoritative
  - **Result:** resolver accepting all records in the response
  - **Cause:** lacking data verification (**bailiwick rules**)



DNS Bailiwick Rules

- **Mitigating the Kashpureff Attack**
  - The credibility checking when storing cache entries
  - Checking for “**in bailiwick**” in response data: **answer records must be from the same domain as the requested name**

```
$ dig example.com

;; ANSWER SECTION:
example.com. 86400 IN A 93.184.216.34

;; AUTHORITY SECTION:
~~mybank.com. 86400 IN NS ns.mybank.com.~~

;; ADDITIONAL SECTION:
~~ns.mybank.com. 86400 IN A 1.2.3.4~~
```

Bailiwick

In-bailiwick
Can be trusted

Out-of-bailiwick
Should be removed

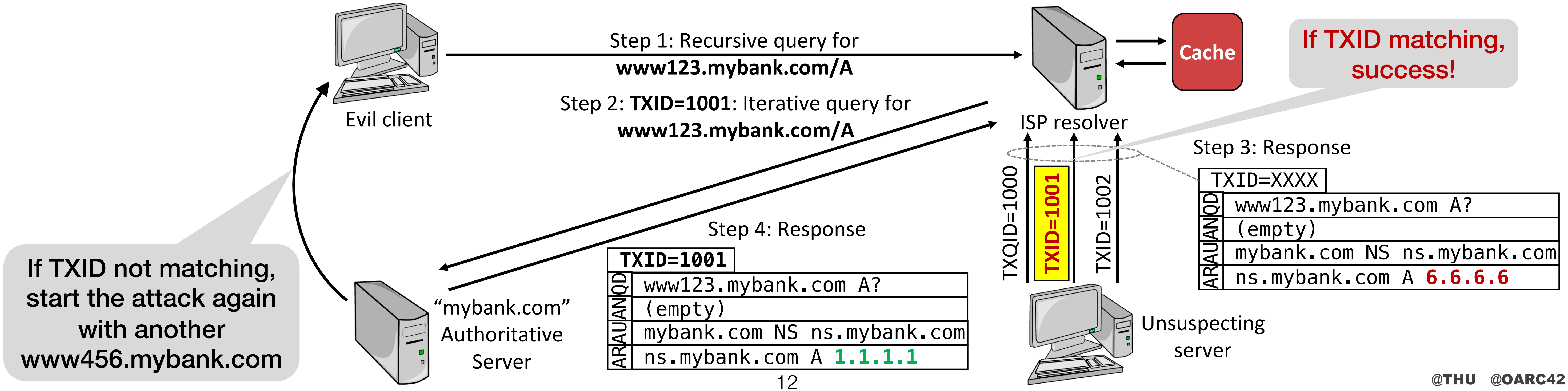
Takeaway

After the Kashpureff attack, bailiwick checking is integrated into the resolver's implementation,

DNS cache poisoning on recursives from the on-path seems **impossible** to conduct from 1997.

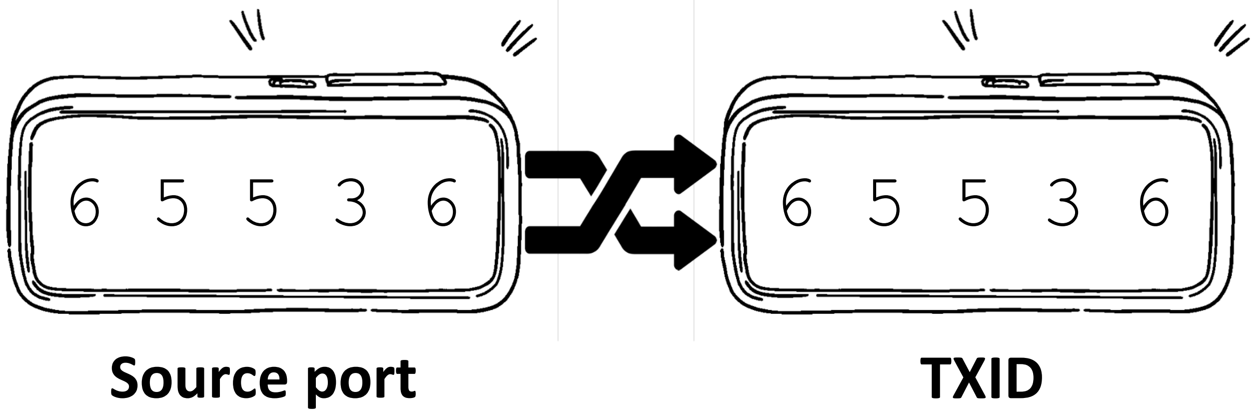
DNS Cache Poisoning (2/5)

- **Kaminsky Attack (Off-path, 2008)**
  - ❑ **Method:** injecting forged responses with the "birthday paradox"
  - ❑ **Result:** resolver accepting glue records in the response
  - ❑ **Cause:** lacking **source port randomization** (TXID only 16 bits)



DNS Source Port/TXID Randomization

- **Mitigating the Kaminsky Attack**
  - ❑ Increasing the query guessing entropy
  - ❑ 16-bit source port x 16-bit TXID = 32-bit space
  - ❑ **Hard to brute-force**



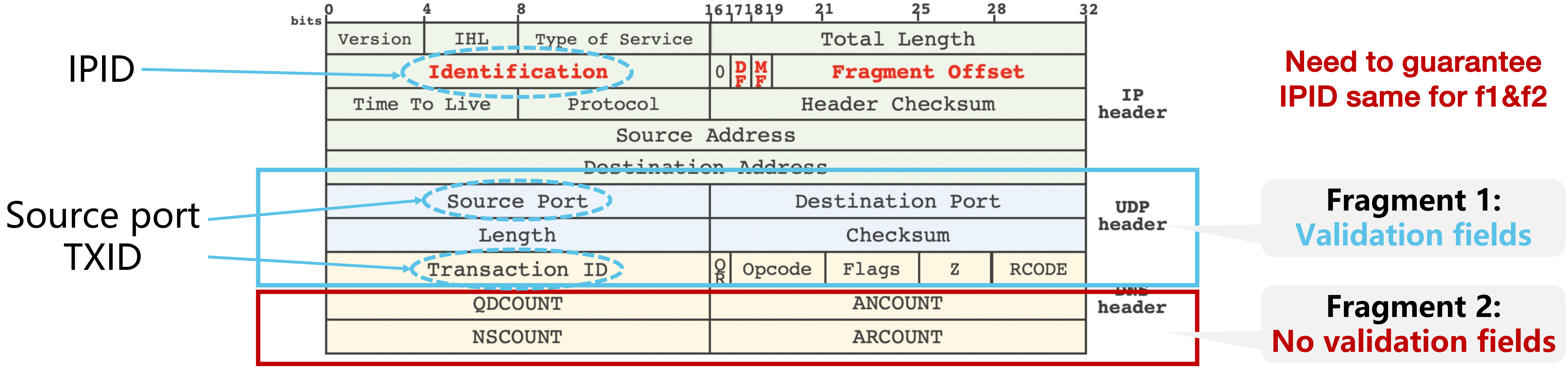
Takeaway

After the Kaminsky attack, source port randomization is integrated into the resolver's implementation,

DNS cache poisoning on resolvers from the off-path became **difficult** to conduct from 2008.

DNS Cache Poisoning (3/5)

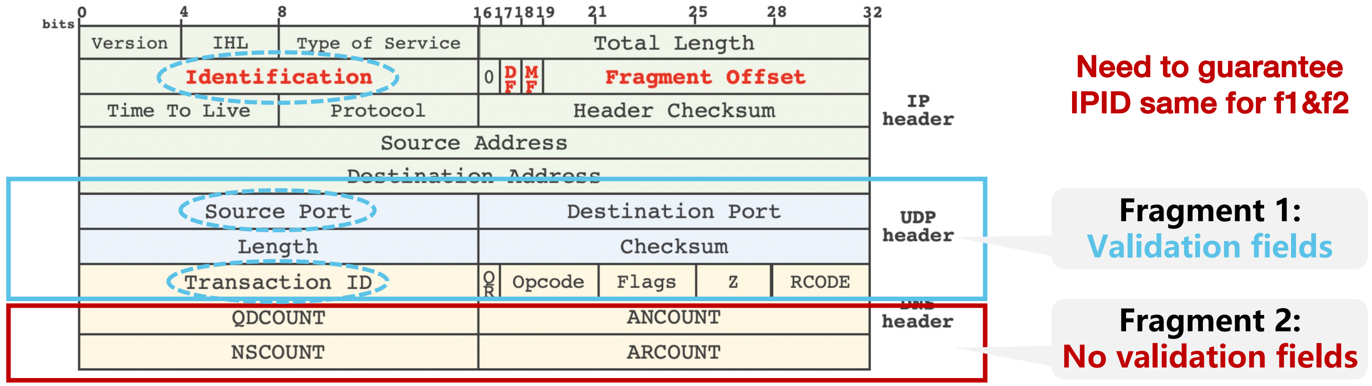
- **Fragmentation-based Attack (Off-path, 2013)**
  - **Method:** injecting forged responses by exploiting the 2nd fragment without checking
  - **Result:** resolver accepting records in the resembled response
  - **Cause:** accepting **small-sized packets** & **predictable IPID** (16-bits)

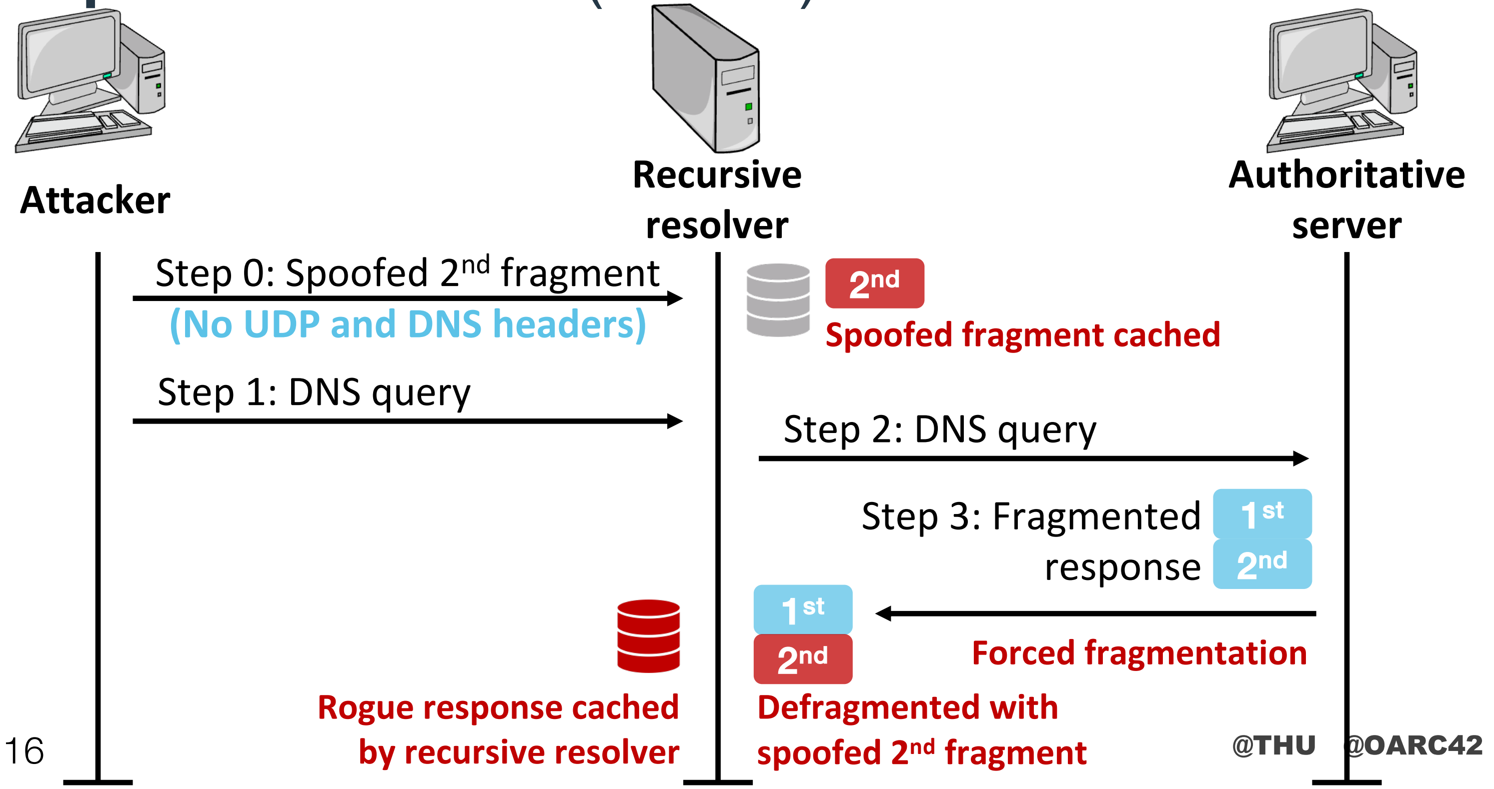


DNS Cache Poisoning (3/5)

➢ Fragmentation-based Attack (Off-path, 2013)

- ❑ **Method:** injecting forged responses by exploiting the 2nd fragment without checking
- ❑ **Result:** resolver accepting records in the resembled response
- ❑ **Cause:** accepting **small-sized packets** & **predictable IPID** (16-bits)





IPID Randomization! Restricting Frag.?

- **Mitigating the Fragmentation-based Attack**
  - **IPID randomization**
    - The fragmentation-based Attack needs to guess the IPID
    - Randomized IPID could prevent the 2nd fragment from being accepted
  - **Restricting fragmentation**
    - The root cause is fragmentation, no fragmentation or restricting it could be one mitigation
    - For example, reducing the packet size, falling back to TCP, restricting the frag_number/timeout
  - **Other methods**
    - Adding new validation fields, such as applying 0x20 encoding to each RRs

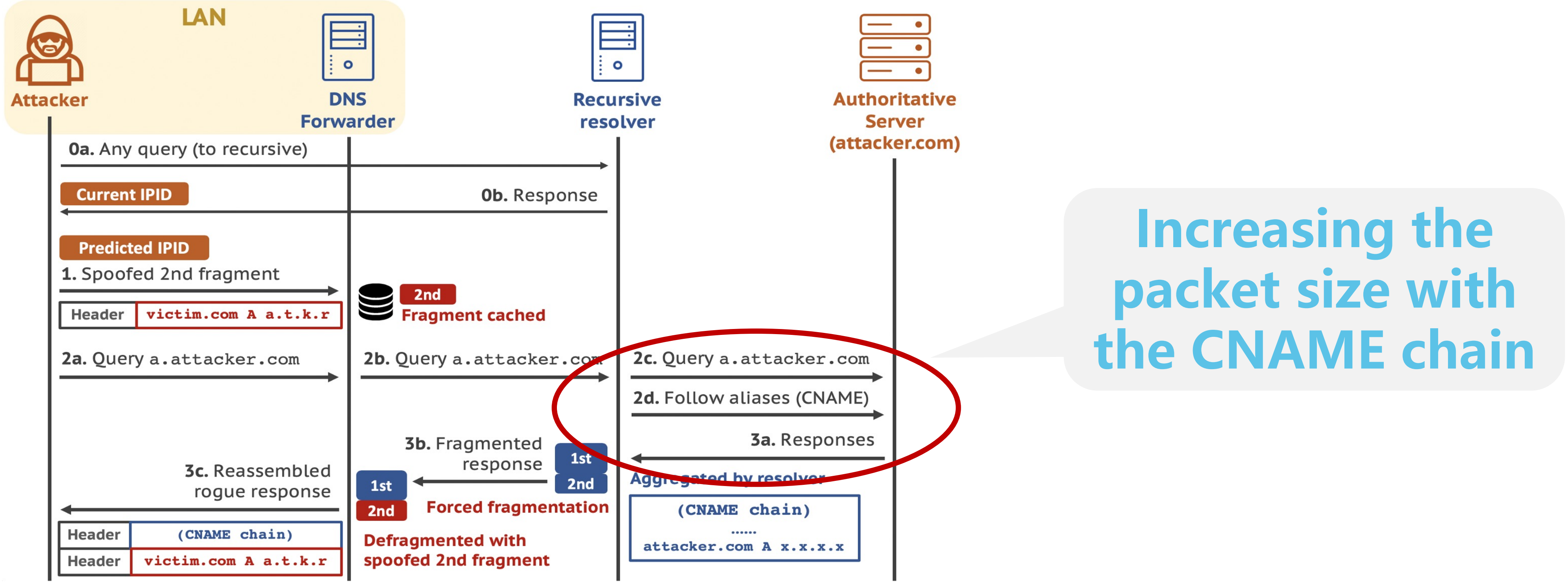
Takeaway

After the fragmentation-based attack, IPID randomization and fragmentation restriction are widely applied in the OS kernel,

DNS cache poisoning exploiting fragmentation became **difficult** to conduct from 2013.

DNS Cache Poisoning (3/5)

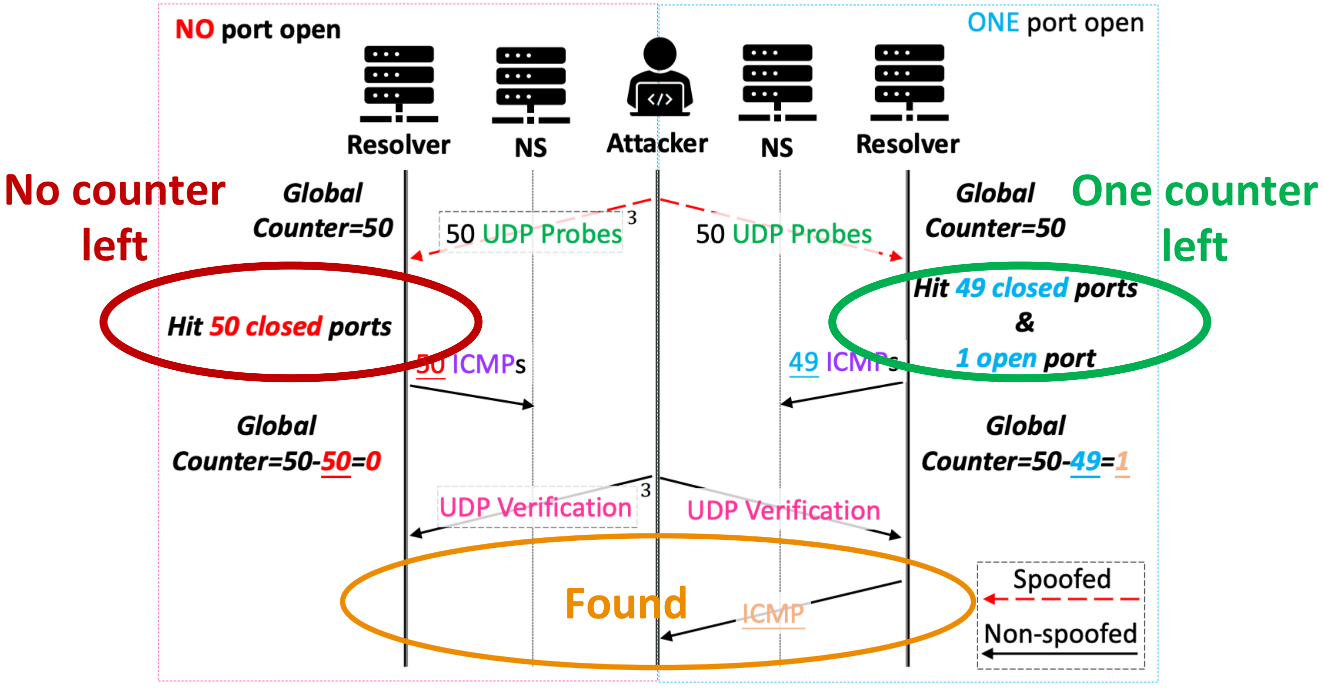
- **Fragmentation-based Attack on Forwarders (Off-path, 2020)**
  - **From our NISL lab, published at USENIX Security 2020**
  - **New method:** although it is not easy to trigger fragmentation for a normal response, we can **increase the packet size** with our own controlled domain



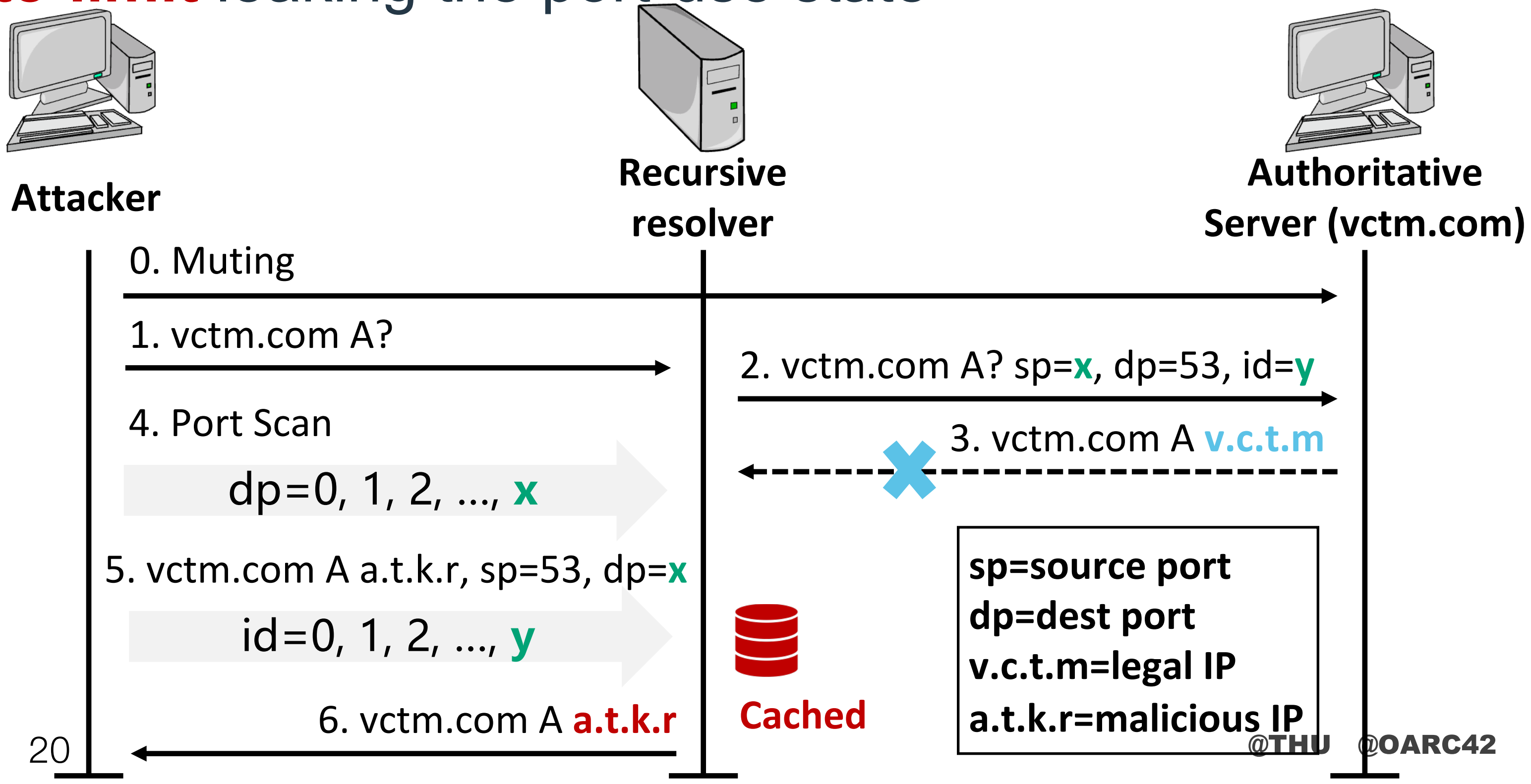
DNS Cache Poisoning (4/5)

➢ **SADDNS Attack (Off-path, 2020)**

- ❑ **Method:** inferring the source port using **Linux kernel's side-channel**
- ❑ **Result:** guessing the source port in a short time, resolver accepting fake records
- ❑ **Cause:** Linux kernel's **global ICMP rate-limit** leaking the port-use state



www.saddns.net



Patching the Linux Kernel

➢ Mitigating the SADDNS Attack

❑ ICMP global rate-limit counter randomization

- Implemented by Linux kernel

icmp: randomize the global rate limiter

```
Keyu Man reported that the ICMP rate limiter could be used
by attackers to get useful signal. Details will be provided
in an upcoming academic publication.

Our solution is to add some noise, so that the attackers
no longer can get help from the predictable token bucket limiter.

Fixes: 4cdf507d5452 ("icmp: add a global rate limitation")
Signed-off-by: Eric Dumazet <edumazet@google.com>
Reported-by: Keyu Man <kman001@ucr.edu>
Signed-off-by: Jakub Kicinski <kuba@kernel.org>
```

```
credit = min_t(u32, icmp_global.credit + incr, sysctl_icmp_msgs_burst);
if (credit) {
        credit--;
        /* We want to use a credit of one in average, but need to randomize
         * it for security reasons.
         */
        credit = max_t(int, credit - prandom_u32_max(3), 0);
        rc = true;
}
```

git.kernel.org

❑ Reducing domain resolution timeout

- SADDNS needs a long timeout to infer the source port
- Prevent the authoritative server from being muted easily

❑ General methods

- 0x20, DNSSEC

Question

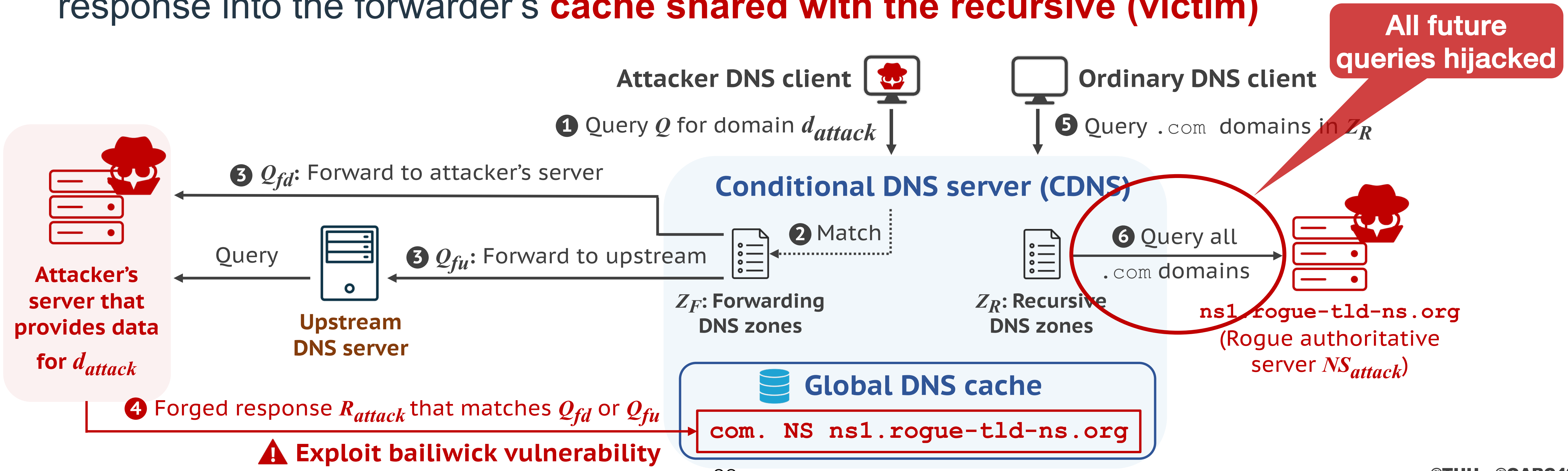
26 years later, does bailiwick checking work as desired after fixing the Kashpureff attack?

No. **MaginotDNS** breaks this guarantee with a new powerful **cache poisoning vulnerability**.

DNS Cache Poisoning (5/5)

➢ **MaginotDNS Attack (On-/Off-path, 2023)**

- ❑ **From our NISL lab, published at USENIX Security 2023**
- ❑ **New attack surface:** exploiting the **bailiwick checking vulnerability** to inject fake response into the forwarder's **cache shared with the recursive (victim)**

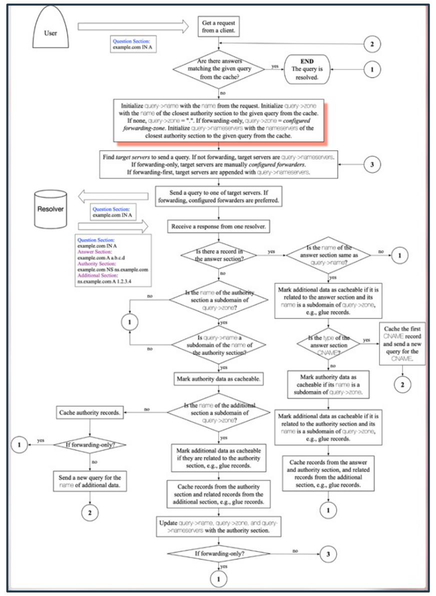


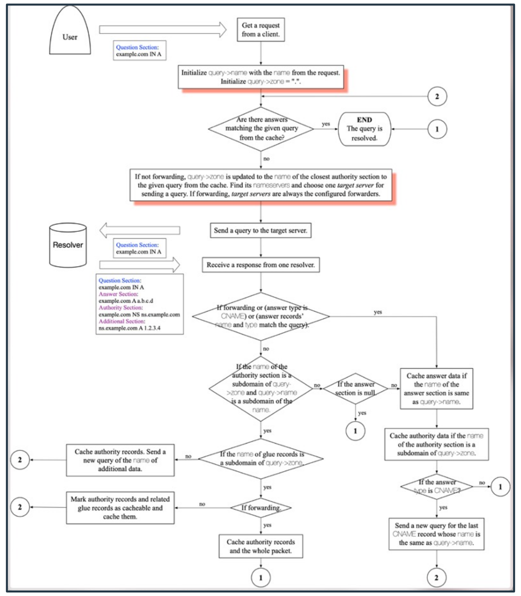
Patching the Resolver Implementation

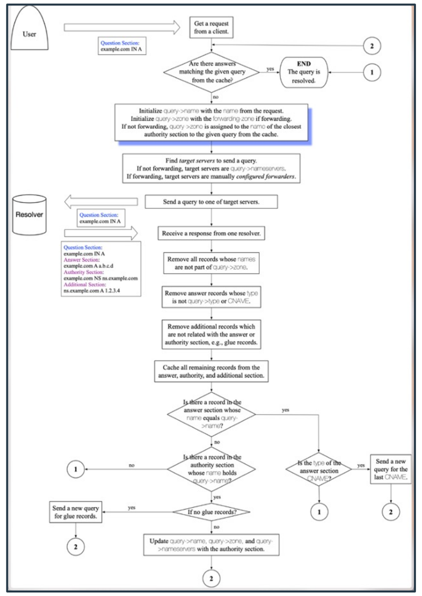
➢ Mitigating the MaginotDNS Attack

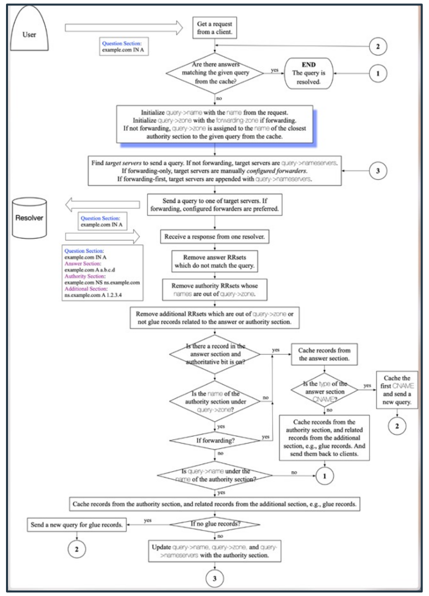
❑ Aligning the bailiwick checking logic between fwders & recurs

- The logic implementation of forwarders is flawed
- Adding bailiwick checking for the forwarder


BIND


Knot


PowerDNS

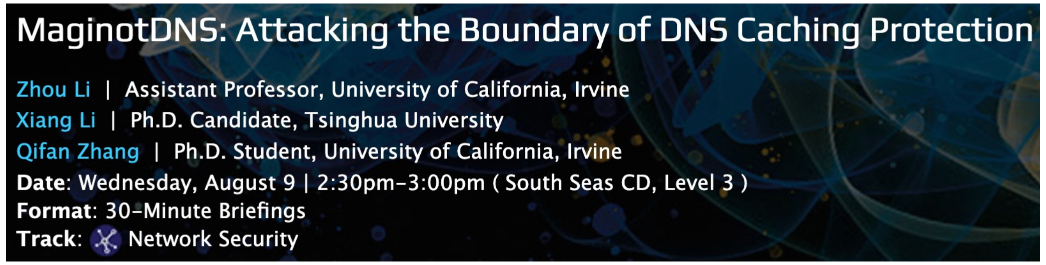

Unbound

Algorithm 1: DNS resolution process

```
input  : A DNS Request from clients
output : A DNS Reply to clients
main()
    step_0: InitQuery(Q, Request)
    step_1: if SeachCache(Q, Cache) then
        goto final
    step_2: FindServers(Q, TgtSvrs)
    step_3: SendQuery(Q, TgtSvrs)
    step_4: ProcessResponse(Q, R)
        if ServerIsError(Q, R) then
            goto step 3
        if not MatchQuery(Q, R) then
            goto final
        SanitizeRecords(Q, R)
        if IsReferral(Q, R) then
            if not IsFwding() then
                UpdateQuery(Q)
                goto step 2
        if IsCNAME(Q, R) then
            UpdateQuery(Q)
            goto step 1
        CacheRecords(R, Cache)
    final: ConstructReply(Reply)
    return Reply
InitQuery(Q, Request)
    initialize Q.name, Q.type, Q.zone
    if IsFwding() then
        ModifyFwdQuery(Q)
SanitizeRecords(Q, R)
    for RR ∈ R do
        if OutOfBailiwick(RR) then
            remove RR from R
UpdateQuery(Q, R)
    update Q.name, Q.type, Q.zone
```

Real-world Impact

- ➢ **Industry**
  - ❑ Presented at [Black Hat USA 2023]
- ➢ **Government/University**
  - ❑ An Austria government [CERT daily report]
  - ❑ A Sweden government [CERT weekly news]
  - ❑ A Bournemouth University (BU) [CERT news]
- ➢ **60+ News Coverage**
  - ❑ E.g., [BleepingComputer]
- ➢ **APNIC Blog**
- ➢ 数字寰宇大家讲堂[公开课]



End-of-Day report

Timeframe: Freitag 11-08-2023 18:00 - Montag 14-08-2023 18:00 Handler: Michael Schlagenhaufer Co-Handler: n/a

News

MaginotDNS attacks exploit weak checks for DNS cache poisoning

MaginotDNS attacks exploit weak checks for DNS cache poisoning (13 aug)
https://www.bleepingcomputer.com/news/security/maginotdns-attacks-exploit-weak-checks-for-dns-cache-poisoning/

MaginotDNS attacks exploit weak checks for DNS cache poisoning

Posted on 15 August 2023

From bleepingcomputer.com

MaginotDNS attacks exploit weak checks for DNS cache poisoning

By Bill Toulas August 13, 2023 10:12 AM 0

Question

Why is the new DNS cache poisoning attack still possible after researchers and vendors did lots of work?

We found that the **DNS response processing logic** has never been studied thoroughly.

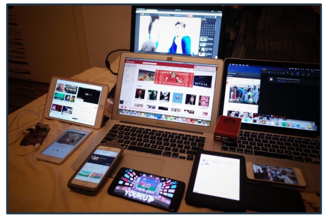
Takeaway

It is necessary to provide a systematic analysis of the DNS response processing logic and expose all potential threats.

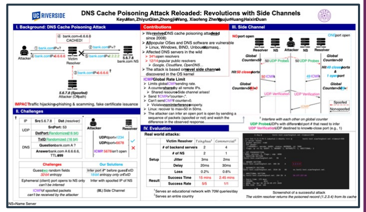
What we did in this paper. And we found,

History Not Over Yet

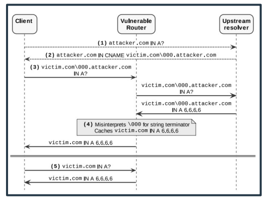
- 1997 — Birthday Attack
- 2002 — Kashpureff Attack
- 2008 — Kaminsky Attack
- 2013 — Fragmentation Attack
- 2020 — Attack on Forwarders



- 2020 — SADDNS Attack
- 2021 — SADDNS v2 Attack



- 2021 — Attack via Escaped Chars



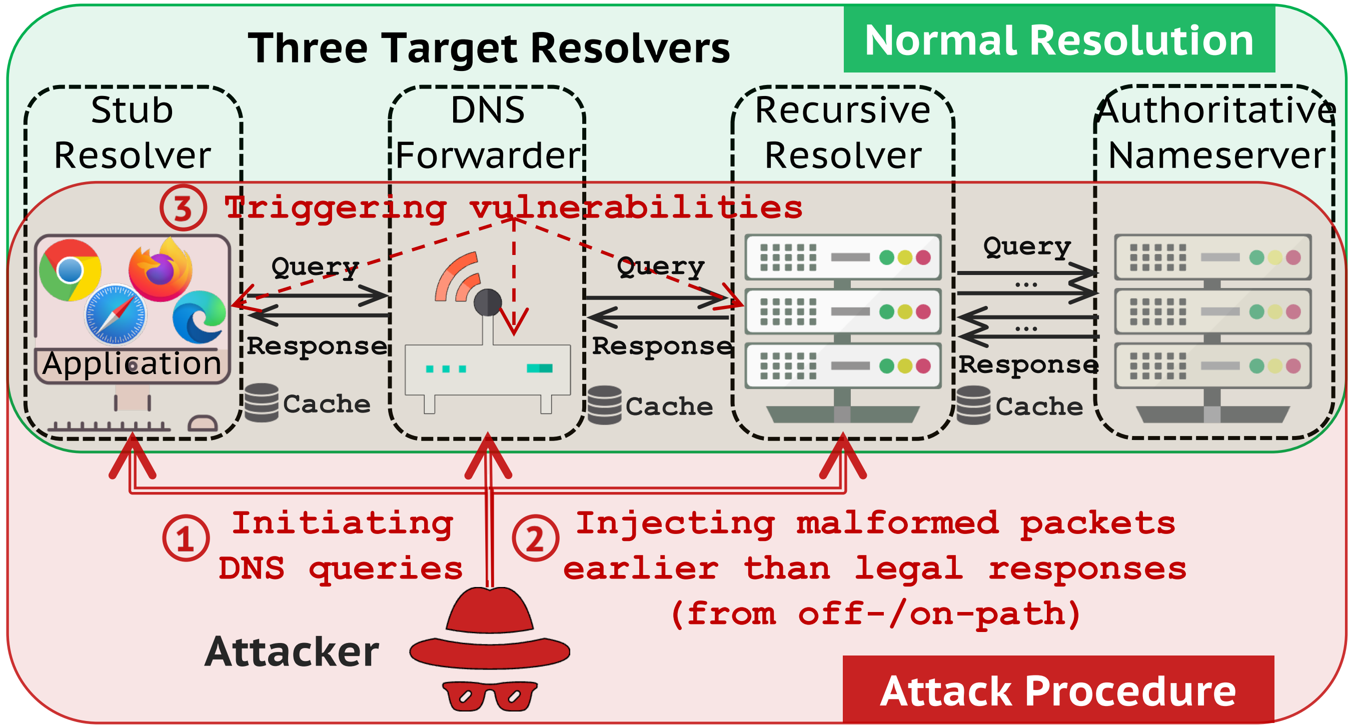
- 2022 — MaginotDNS Attack
- 2023 — Attack via Escaped Chars v2
- 2024 — **TuDoor Attack (Fastest)**

TuDoor Attack

- ➤ **What is the TuDoor attack**
  - ❑ Proposed by our **NISL** lab, published at **[IEEE S&P 2024]**
  - ❑ A new set of powerful DNS-related attacks
    - DNS cache poisoning, DoS, and resource consuming
  - ❑ Among them, **TuDoor can poison vulnerable resolvers within 1s**
- ➤ **Name**
  - ❑ Exploiting **vulnerabilities** of DNS response processing logic
  - ❑ A very covert response door → like 突门 in the Great Wall
  - ❑ Called the **TuDoor** attack

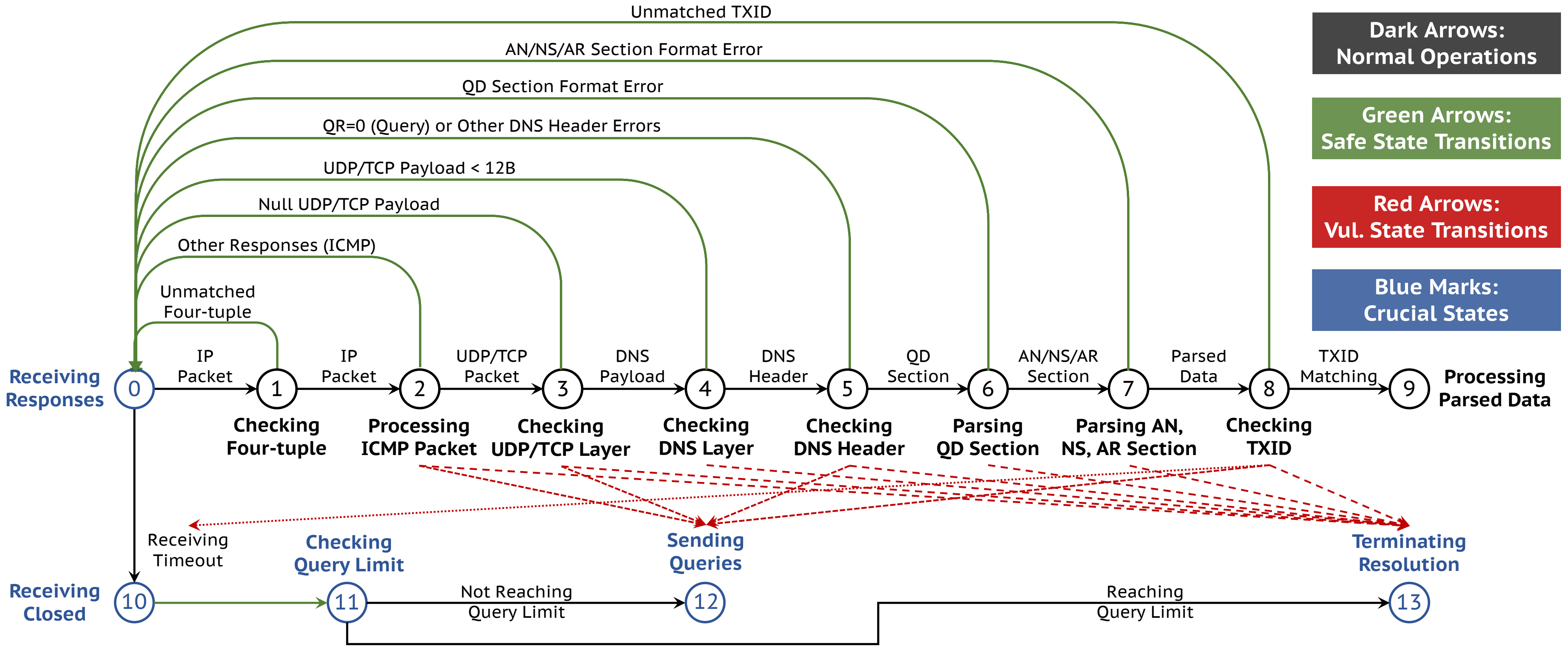
Attack Overview of TuDoor

- **Attack Target**
  - ❑ Resolvers, including stub resolver, DNS forwarders, and recursive resolvers
- **Threat Model**
  - ❑ Identifying the target resolver
  - ❑ Triggering different vulnerabilities
  - ❑ Conducting the attack



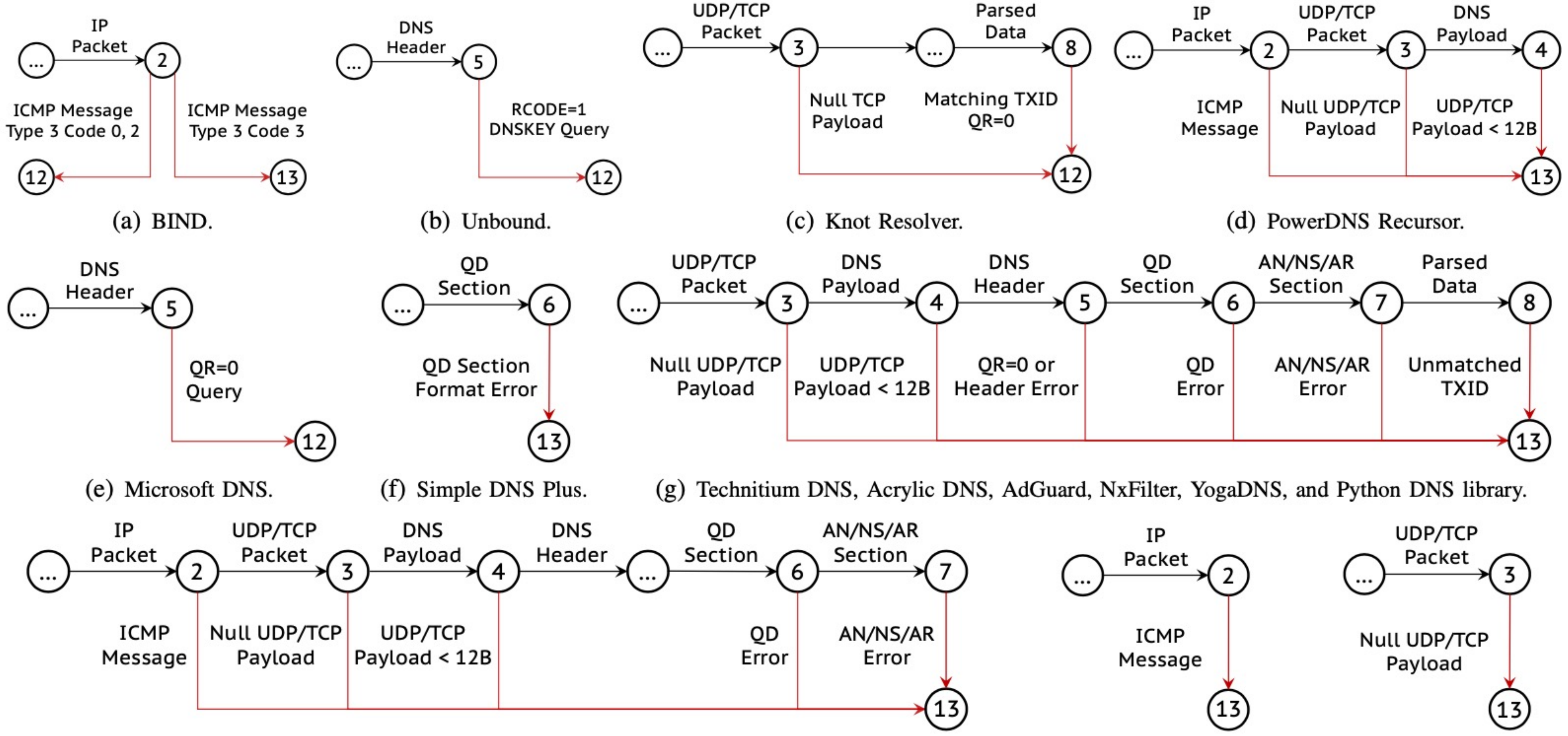
Analysis of DNS Response Processing

- **Systematic Analysis**
  - 28 DNS software → **Constructing processing states**
    - 8 recursive resolvers, 10 DNS forwarders, 6 stub resolvers, 4 DNS programming libraries



Vulnerable State Transitions

- ➢ **DNS Response Pre-processing Implementations**
  - ❑ Part software
  - ❑ **Red lines**
    - o **Vulnerable**



(a) BIND. (b) Unbound. (c) Knot Resolver. (d) PowerDNS Recursor.

(e) Microsoft DNS. (f) Simple DNS Plus. (g) Technitium DNS, Acrylic DNS, AdGuard, NxFilter, YogaDNS, and Python DNS library.

(h) CoreDNS, DNS Safety, and Golang DNS library. (i) pdnsd and Linux stub DNS. (j) Windows stub DNS.

Vulnerable Software & Public Resolvers

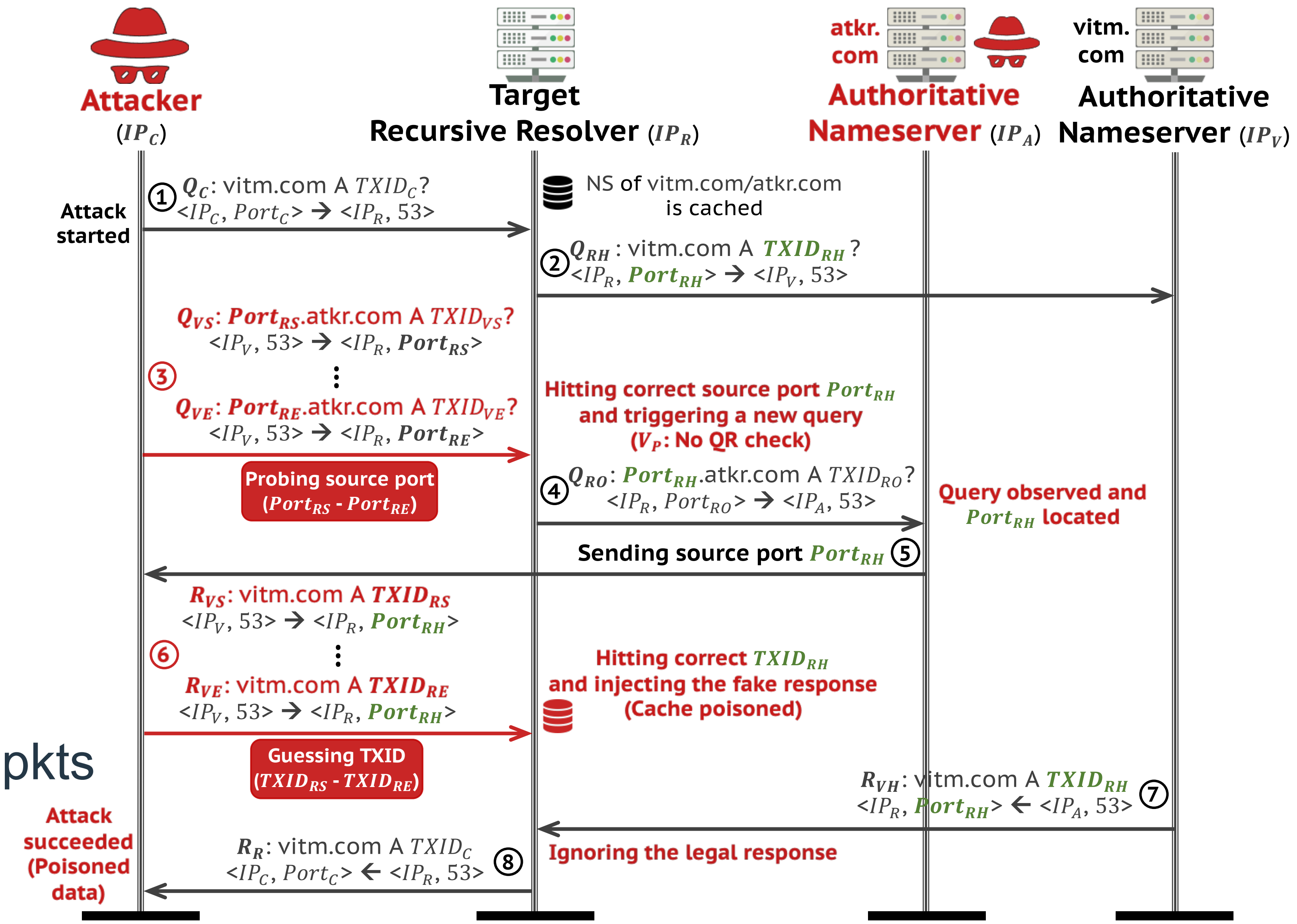
- **24/28 Software**
  - Vulnerable
- **18/42 Public Resolver**
  - Vulnerable
- **Cache poisoning**
- **DoS**
- **Resource consuming**

Resolver			Resolution		Vulnerable state transition												Vulnerability		
Role	Software	Version	Query count	Negative caching	⑧→⑩	②→⑫	③→⑫	⑤→⑫	⑧→⑫	②→⑬	③→⑬	④→⑬	⑤→⑬	⑥→⑬	⑦→⑬	⑧→⑬	V_{CP}	V_{DS}	V_{RC}
Recursive	BIND	9.18.14	13	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓
	Unbound	1.17.1	9	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
	Knot	5.5.3	3	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
	PowerDNS	4.8.3	1	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗
	Microsoft	2022	2	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
	Simple DNS+	9.1.111	3	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗
	Technitium	11.0.2	6	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗
	MaraDNS	3.5.0036	6	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Forwarder	Dnsmasq	2.89	1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	CoreDNS	1.10.1	3	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✓	✓	✗	✓	✓	✗
	Pi-hole	5.17.1	1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	pdnsd	1.2.9	1	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
	Acrylic DNS	2.1.1	1	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗
	AdGuard	7.14	2	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗
	DNS Safety	1.0	1	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✓	✓	✗	✓	✓	✗
	Dual DHCP DNS	8.00RC	1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
	NxFilter	4.6.7.6	3	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗
	YogaDNS	1.37	1	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗
Stub	Linux	253	6	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
	Windows	2023	5	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗
	MacOS	13.2.1	6	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
	IOS	16.3.1	6	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
	Android	13	4	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	ChromeOS	111.x	5	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗
Library	Python	2.3.0	1	-	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✗	✓	✗
	Golang	2023	1	-	✗	✗	✗	✗	✗	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗
	JavaScript	19.8.1	1	-	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗
	Java	3.5.2	1	-	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗

'-': Not applicable due to no caching. ✓: Yes. ✗: No. ✓ (red): Vulnerable. ✗ (green): Not vulnerable.

Attack Steps of TuDoor

- **Three Attacks**
  - ❑ Cache Poisoning
  - ❑ DoS
  - ❑ Resource Consuming
- **Attack steps**
  - ❑ Example: cache poisoning
  - ❑ One new side-channel vulnerability
  - ❑ Exposing the source port
  - ❑ Attackers just need to send <65,535 pkts
  - ❑ **Attack time: avg. 425ms**



Vulnerable Open Resolvers

- Internet Scanning
  - ❑ Designed probing policies
  - ❑ Using XMap (Open-sourced tool)
  - ❑ **423k (23.1%) out of 1.8M vulnerable**

xmap Public

XMap is a fast network scanner designed for performing Internet-wide IPv6 & IPv4 network research scanning.

C ☆ 291 44

Region	#	%	AS	#	%
China	658,312	35.8%	ASN 4134	247,572	13.5%
India	141,668	7.7%	ASN 4837	126,485	6.9%
United States	135,201	7.4%	ASN 4538	63,151	3.4%
South Korea	84,908	4.6%	ASN 24560	63,062	3.4%
Russia	79,978	4.4%	ASN 17488	54,148	2.9%
Indonesia	66,147	3.6%	ASN 4847	47,276	2.6%
Brazil	52,609	2.9%	ASN 4766	39,880	2.2%
Bangladesh	41,073	2.2%	ASN 4808	30,784	1.7%
Iran	38,739	2.1%	ASN 58224	27,598	1.5%
Japan	26,018	1.4%	ASN 3462	22,900	1.2%
Total 227 regions			Total 24,941 ASes		

Type	Resolver number and percentage	
Collected	Alive on 03/10/2023	**1,837,442 (100.0%)**
Software identified	Microsoft DNS	205,984 (11.2%)
	BIND	54,813 (3.0%)
	Unbound	12,765 (0.7%)
	PowerDNS Recursor	12,750 (0.7%)
	Knot Resolver	45 (0.0%)
	CoreDNS	8 (0.0%)
Vulnerable	DNSPoisoning	205,984 (11.2%)
	DNSDoS	216,317 (11.8%)
	DNSConsuming	67,623 (3.7%)
	TuDoor	**423,652 (23.1%)**

Discussion & Mitigation

- **Vulnerability Disclosure**
  - Confirmed and fixed by **all affected software**: BIND9, Knot, & Microsoft
  - **33 CVE-ids** published & **Bounty** awarded by Microsoft
  - Referenced by **RFC 9520**

Negative Caching of DNS Resolution Failures

A cache poisoning attack (see Section 2.2 of [RFC7873]) resulting in denial of service may be possible because failure messages cannot be signed. An attacker might generate queries and send forged failure messages, causing the resolver to cease sending queries to the authoritative name server (see Section 2.6 of [RFC4732] for a similar "data corruption attack" and Section 5.2 of [TuDoor] for a "DNSDoS attack"). However, this would require continued spoofing throughout the backoff period and repeated attacks due to the 5-minute cache limit. As in Section 4.1.12 of [RFC4686], this attack's effects would be "localized and of limited duration".

- **Root Cause**
  - Poor implementations failing to considering corner cases
- **Mitigation Solution**
  - Resolvers should await a time window for promising normal response
  - Ignoring queries sent to non-53 ports
- **Detection & Online Tool**

Wrap-up

Paper

Thanks for listening!
Any question?

Xiang Li, Tsinghua University

x-l19@mails.tsinghua.edu.cn

Tool