



AUGUST 7-8, 2024

BRIEFINGS

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

Speaker(s): Qi Wang, Tsinghua University

Contributor(s): Xiang Li, Nankai University & Chuhan Wang, Tsinghua University

Attack Impact

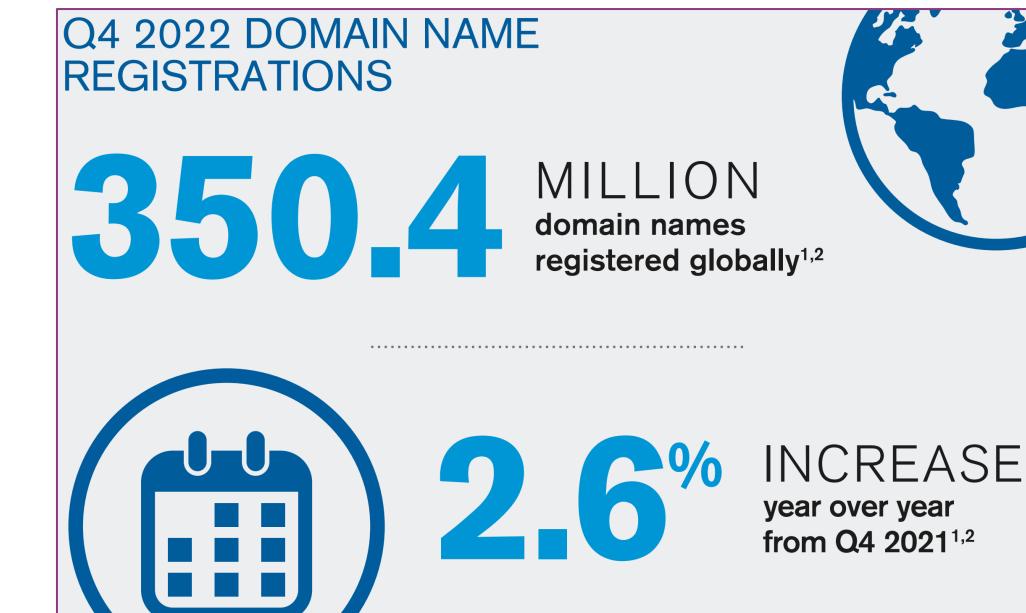
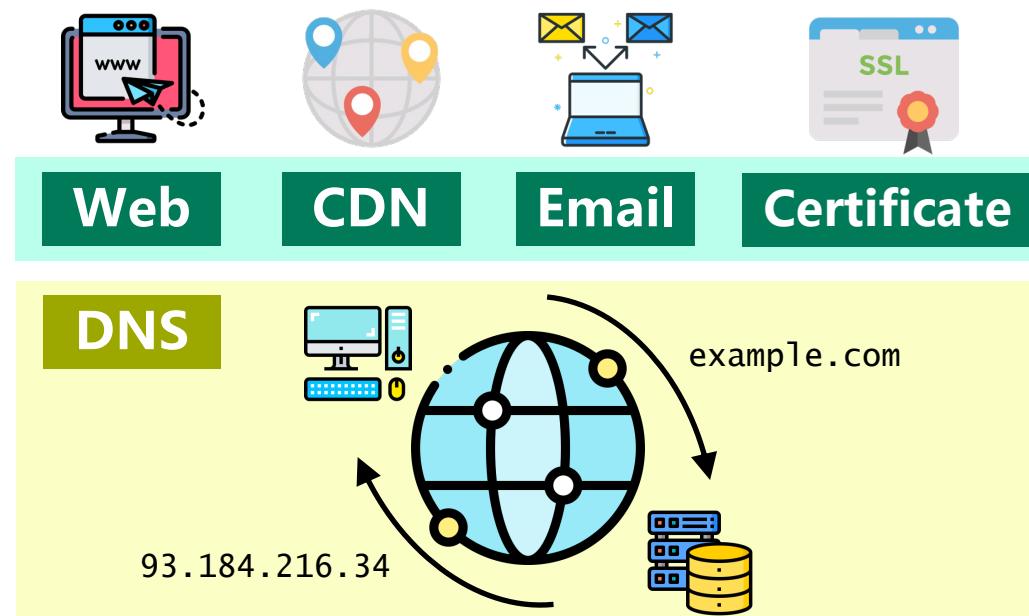
Our TuDoor attack could poison arbitrary domains, e.g., .com and .net.

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



Domain Name System (DNS)

➤ Hierarchical Name Space

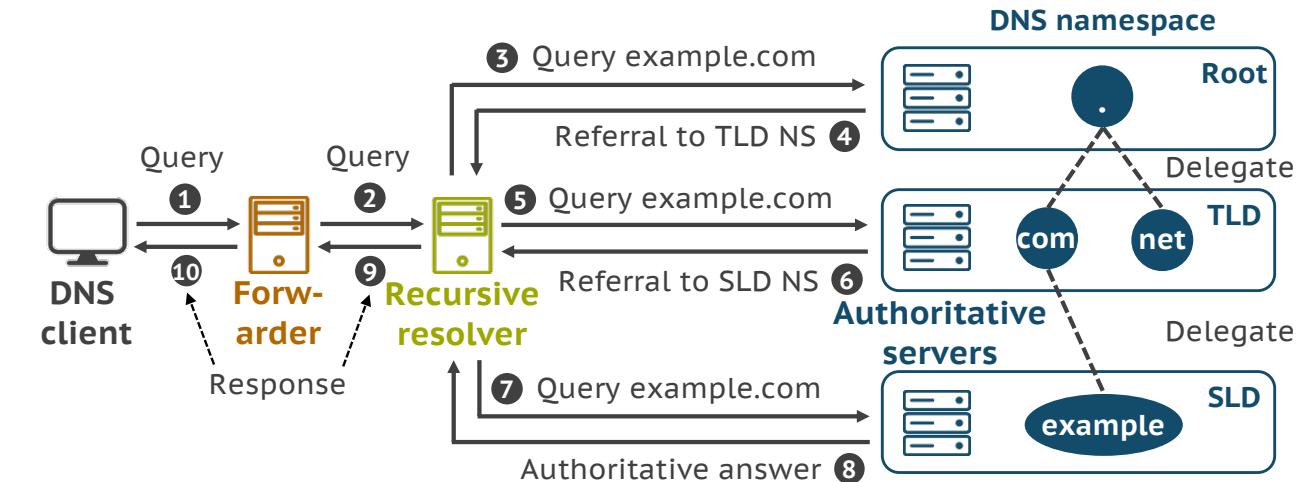
- Authoritative zones: root, TLD, SLD → DNS records
- Domain delegation → Domain registration

➤ Multiple Resolver Roles

- Client, forwarder, recursive, authoritative
- Caching

➤ Iterative Resolution Process

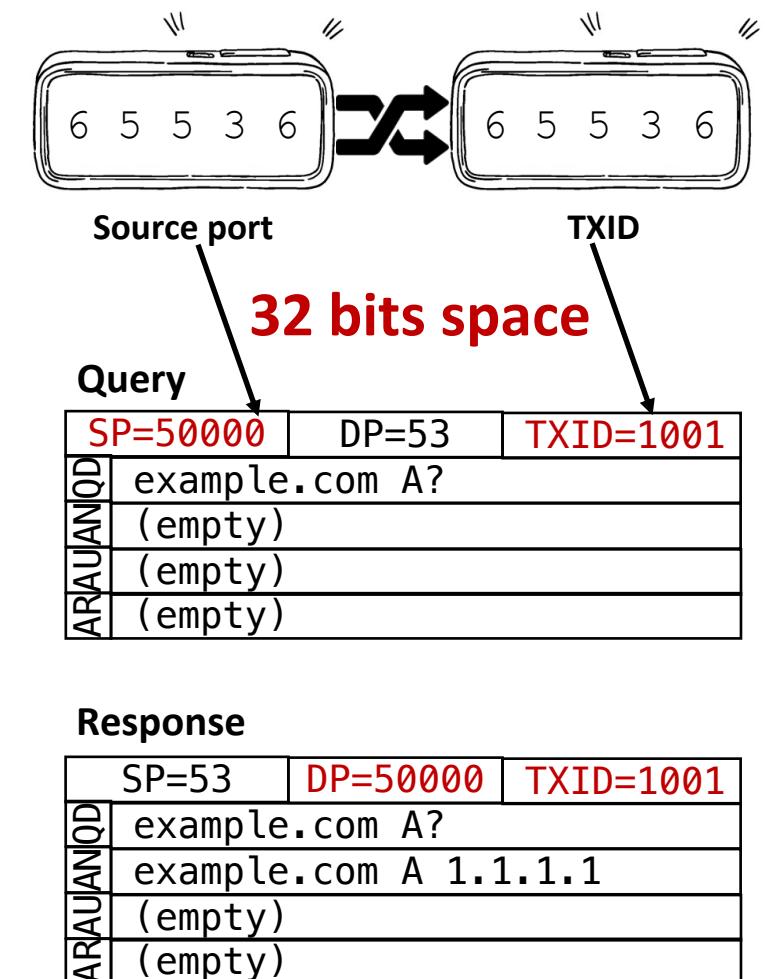
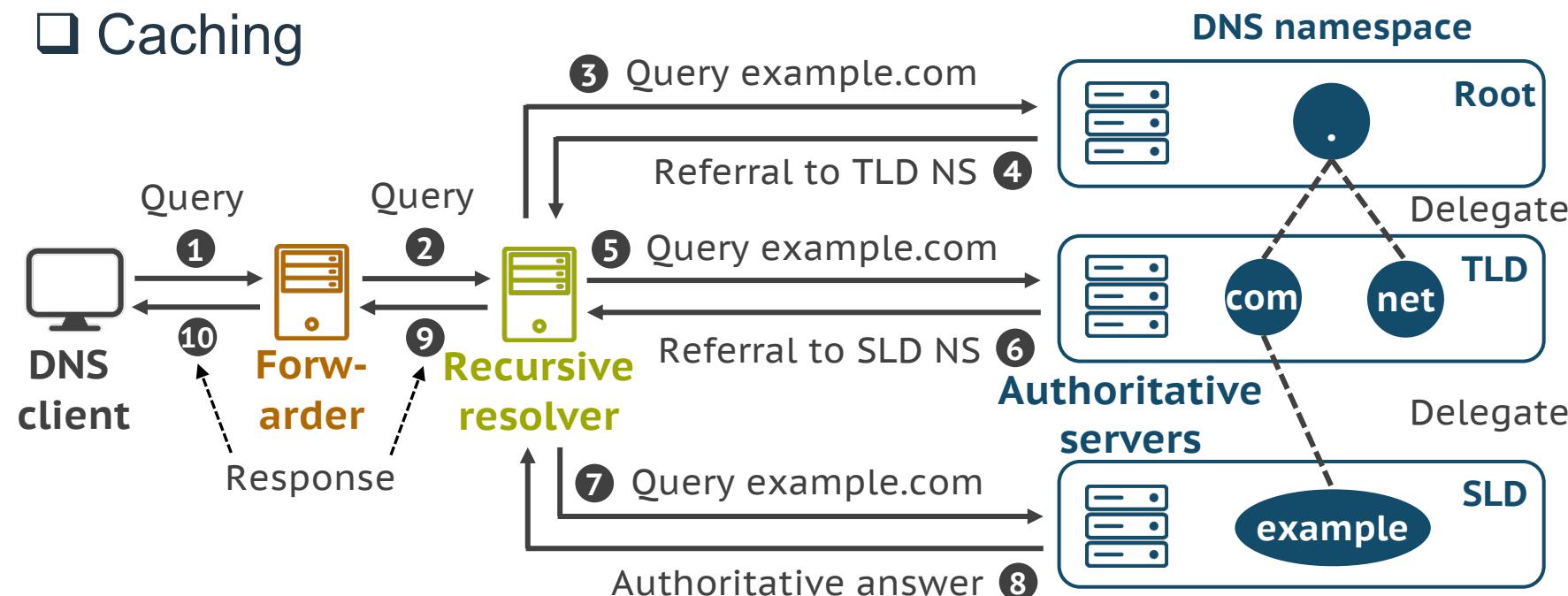
- Client-server style



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

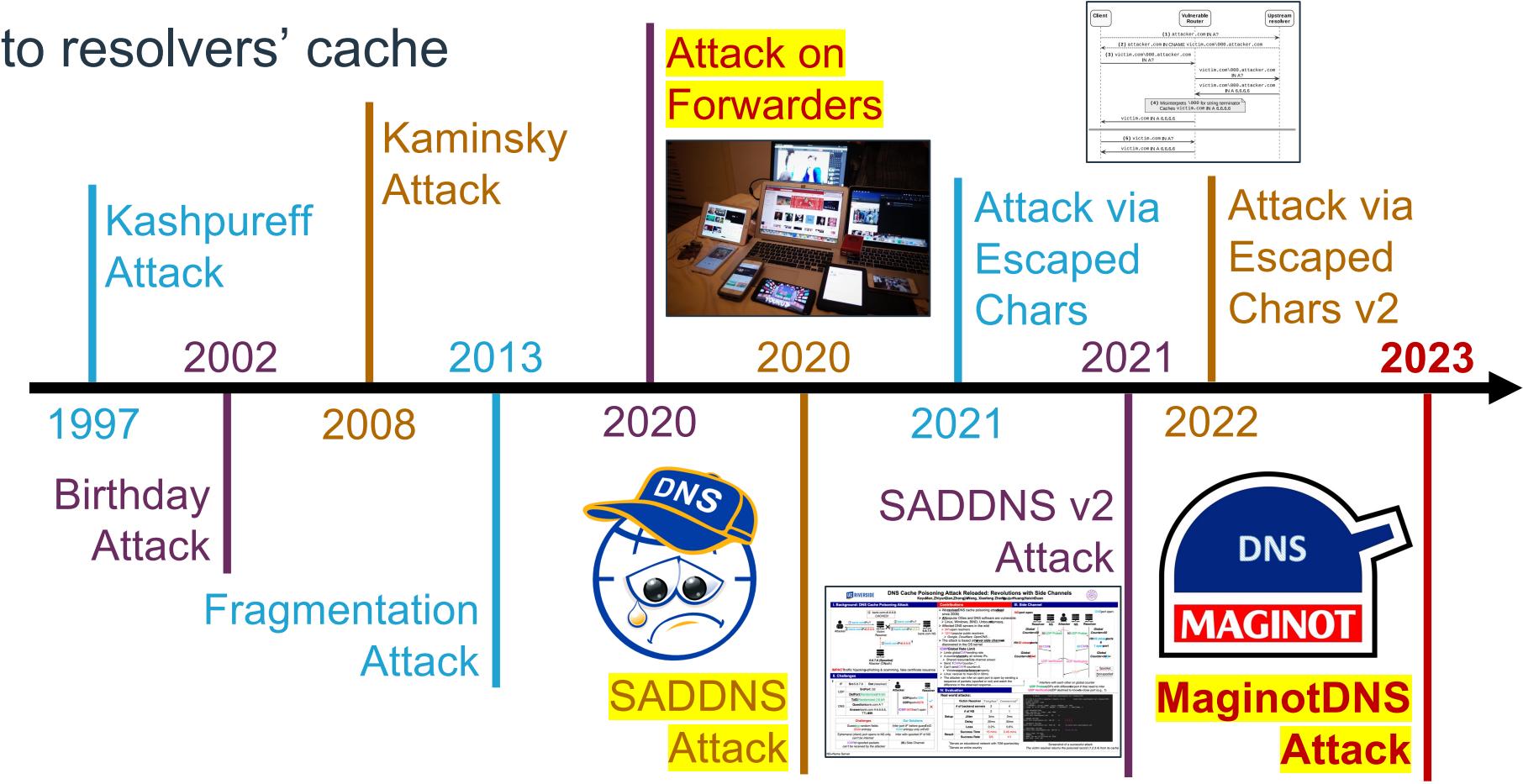
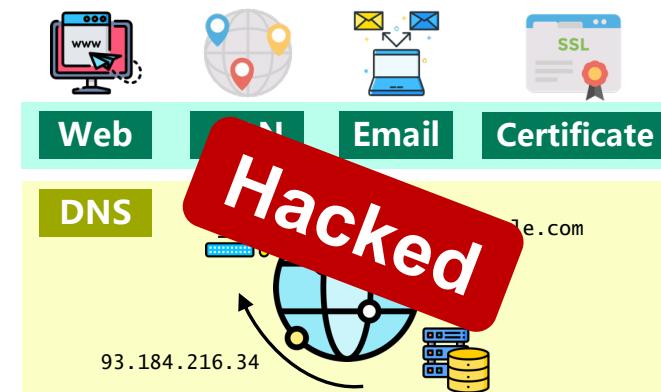
- Injecting forged answers into resolvers' cache

➤ Taxonomy

- On-path, off-path

➤ Technique

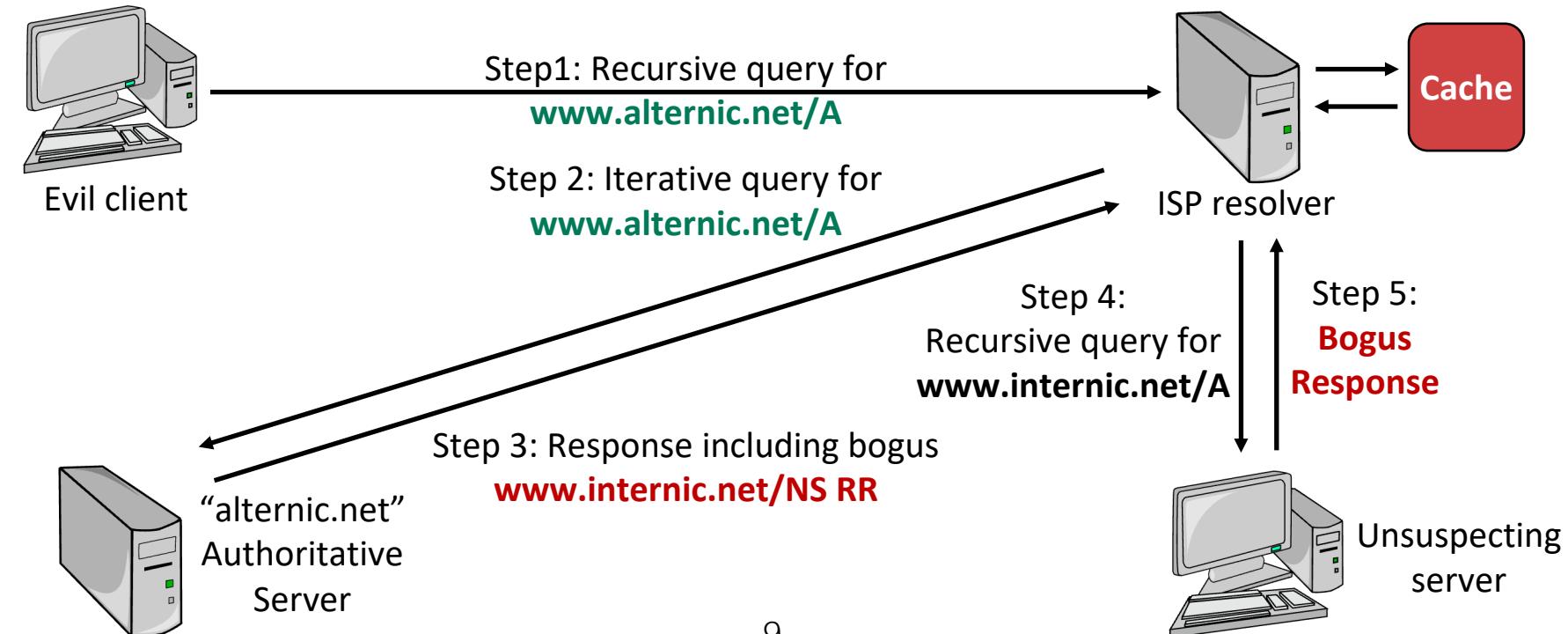
- Cat-and-mouse game



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
```

Bailiwick

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

In-bailiwick
Can be trusted

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

Out-of-bailiwick
Should be removed

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

Takeaway

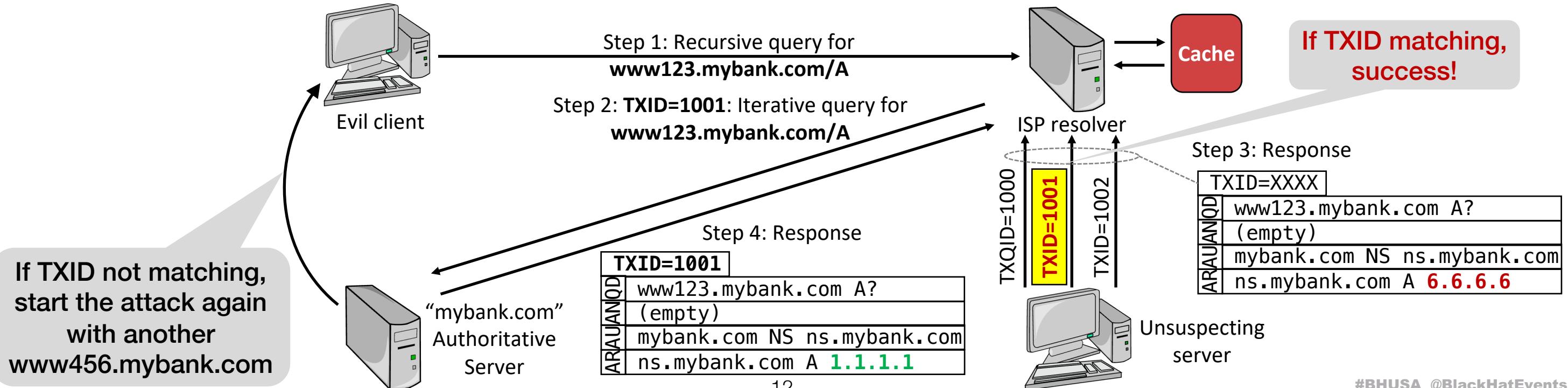
After the Kashpureff attack, bailiwick checking is integrated into the resolver's implementation,

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)

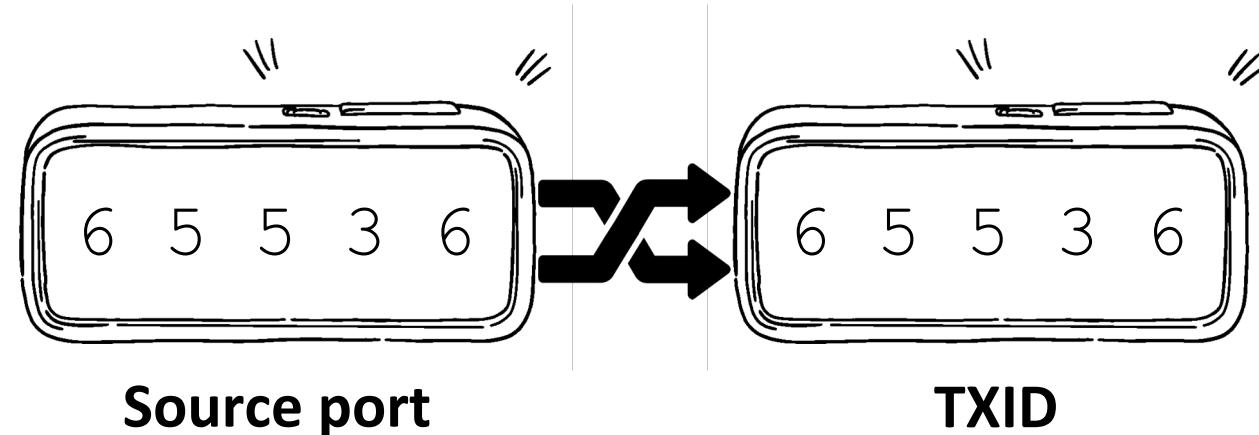
- **Method:** injecting forged responses with the “birthday paradox”
- **Result:** resolver accepting glue records in the response
- **Cause:** lacking source port randomization (TXID only 16 bits)



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

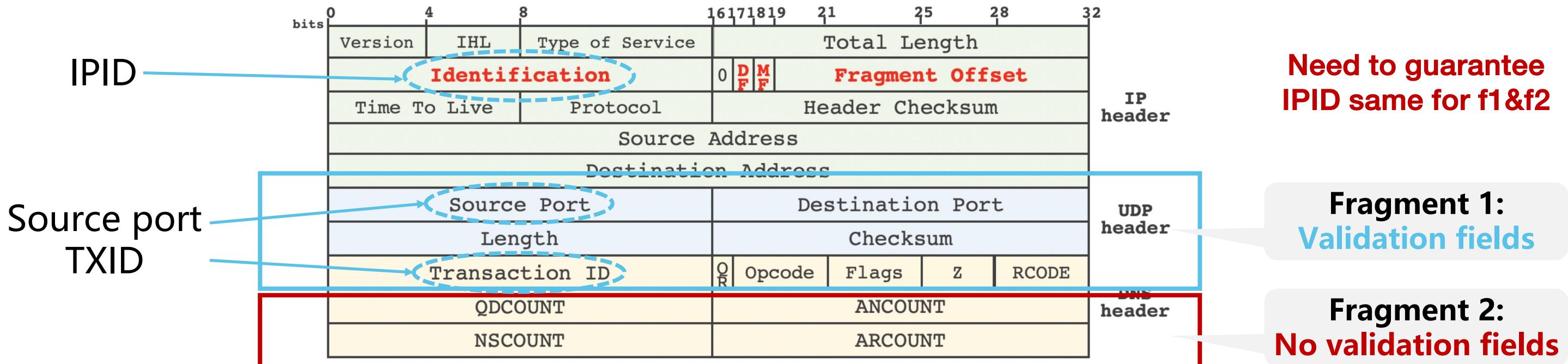
After the Kaminsky attack, source port randomization is integrated into the resolver's implementation,

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)

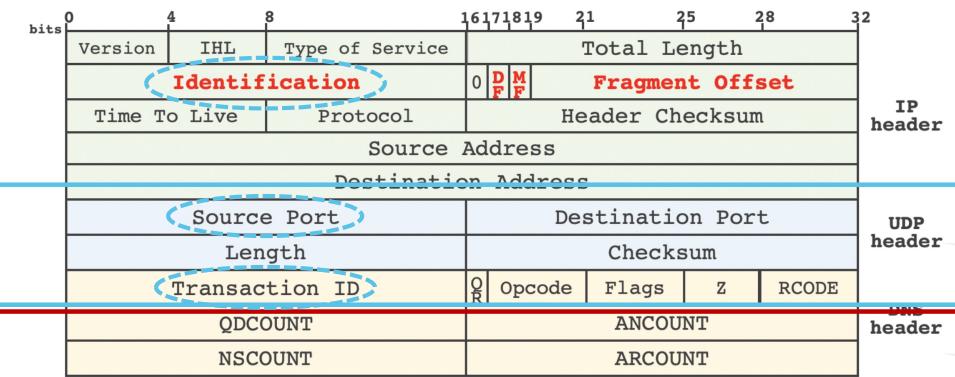
- Method: injecting forged responses by exploiting the 2nd fragment without checking
- Result: resolver accepting records in the resembled response
- Cause: accepting **small-sized packets & predictable IPID** (16-bits)



DNS Cache Poisoning (3/5)

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

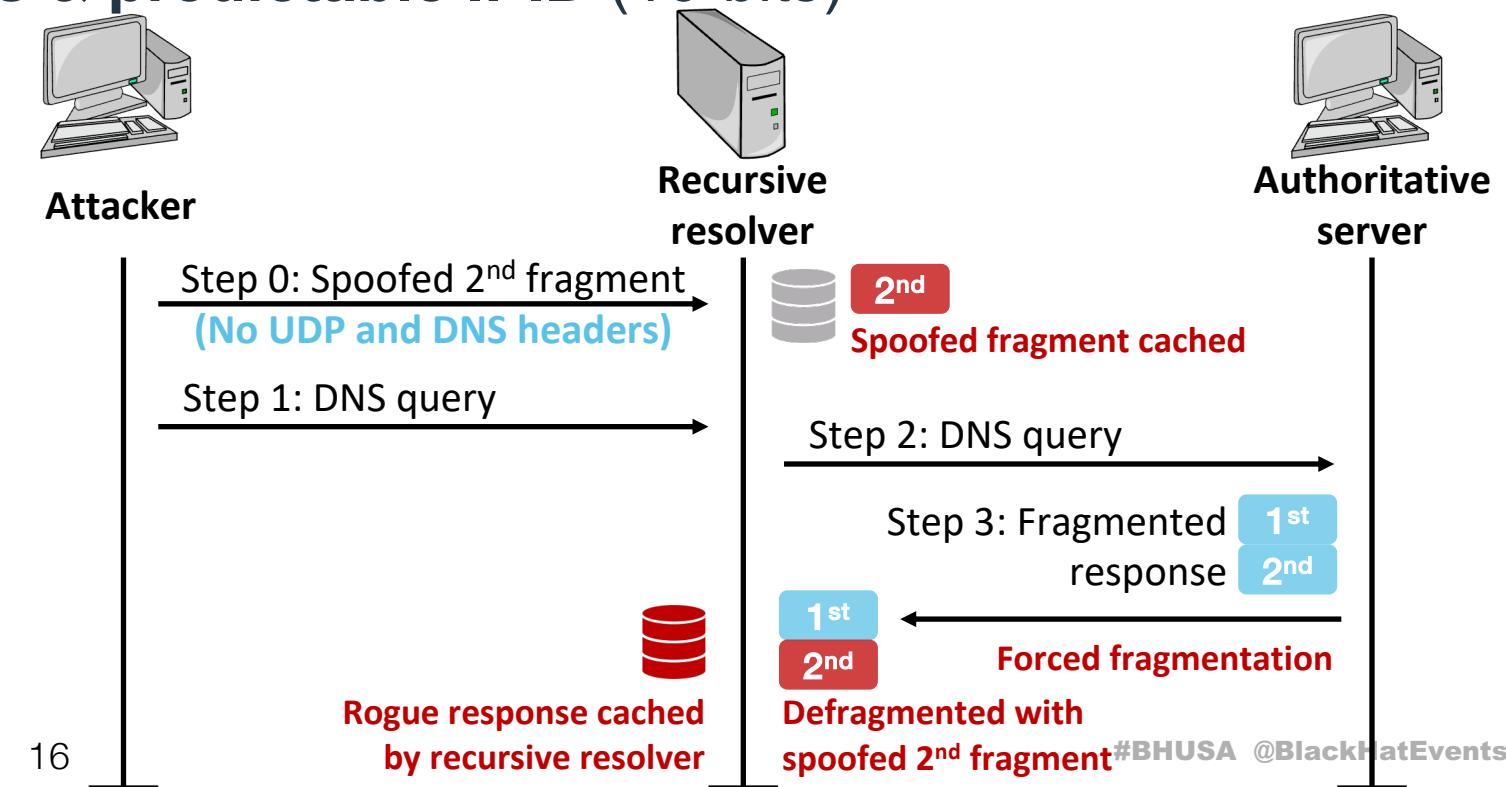
- **Method:** injecting forged responses by exploiting the 2nd fragment without checking
- **Result:** resolver accepting records in the resembled response
- **Cause:** accepting small-sized packets & predictable IPID (16-bits)



Need to guarantee
IPID same for f1&f2

Fragment 1:
Validation fields

Fragment 2:
No validation fields



IPIPID Randomization! Restricting Frag.?

➤ Mitigating the Fragmentation-based Attack

□ IPIPID randomization

- The fragmentation-based Attack needs to guess the IPIPID
- Randomized IPIPID 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

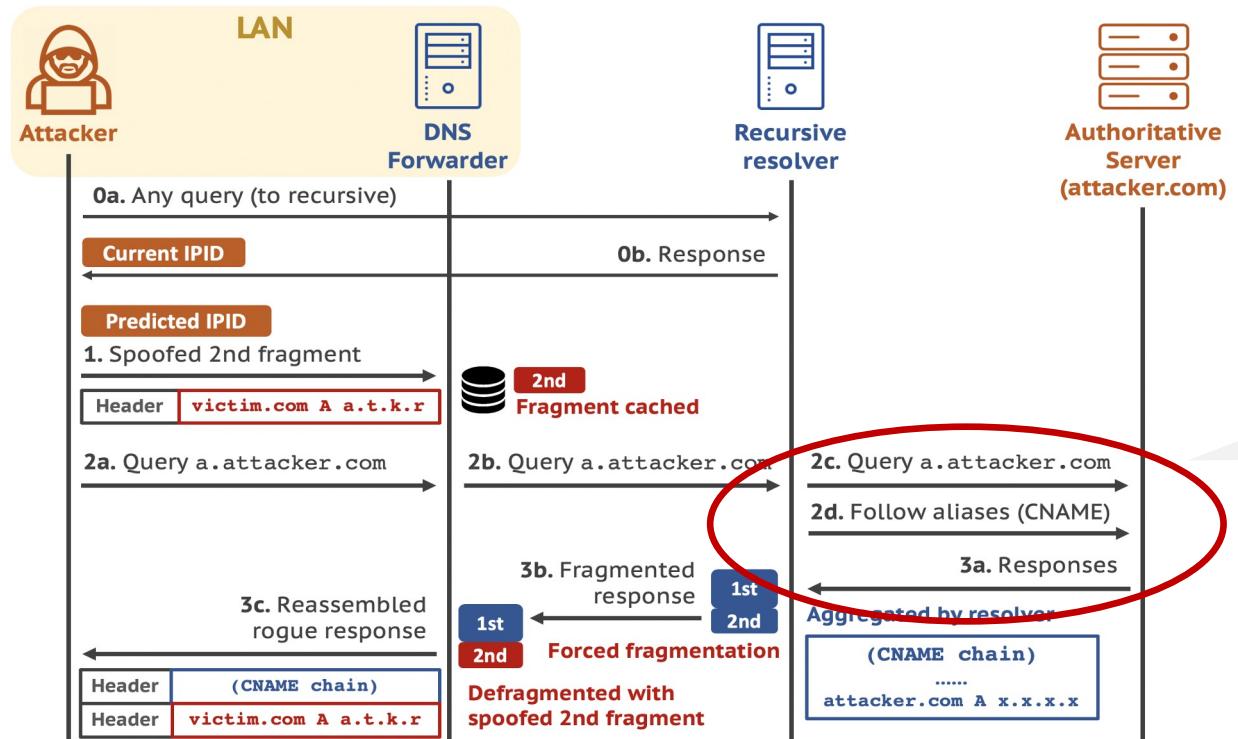
Takeaway

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

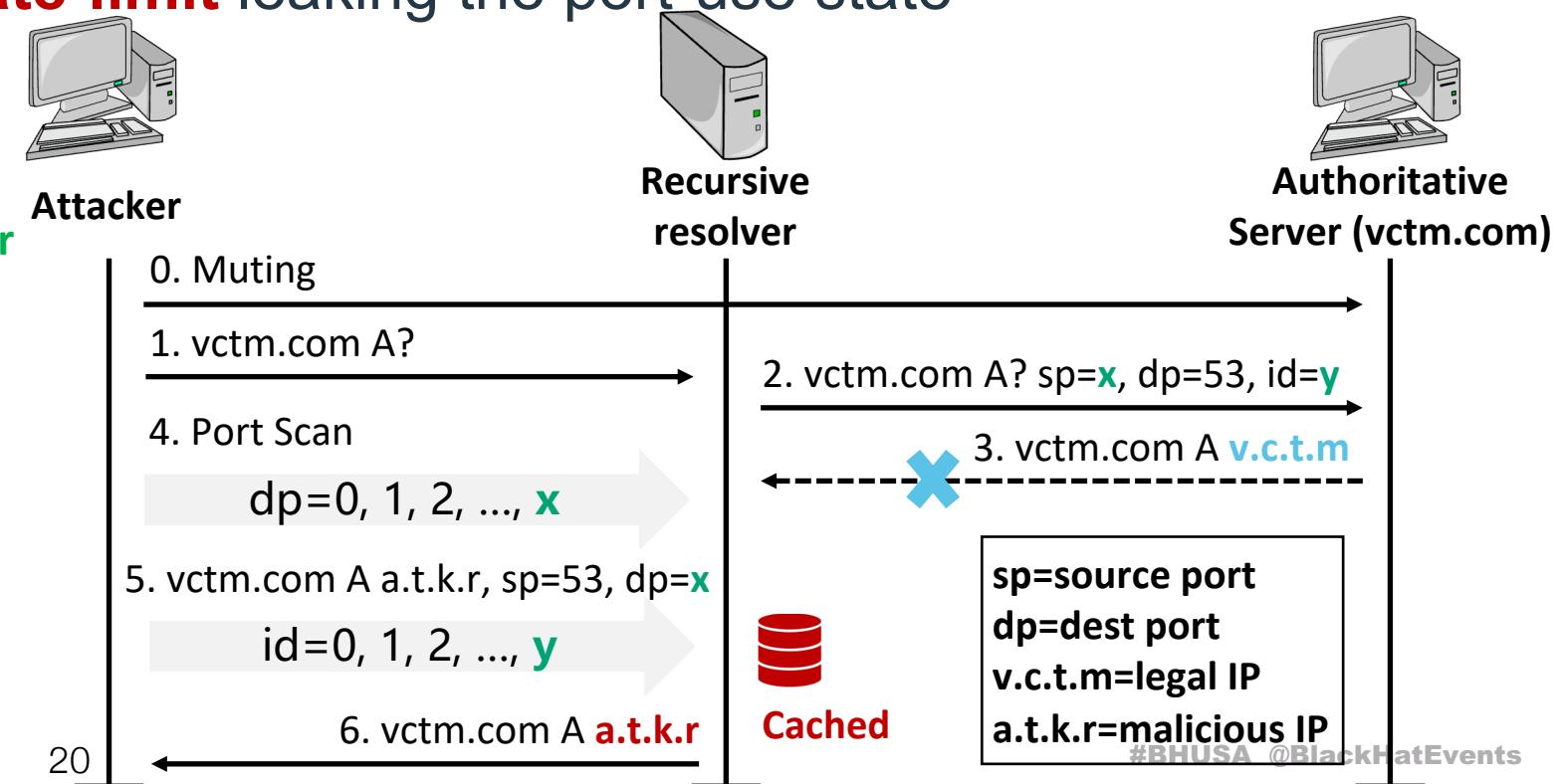
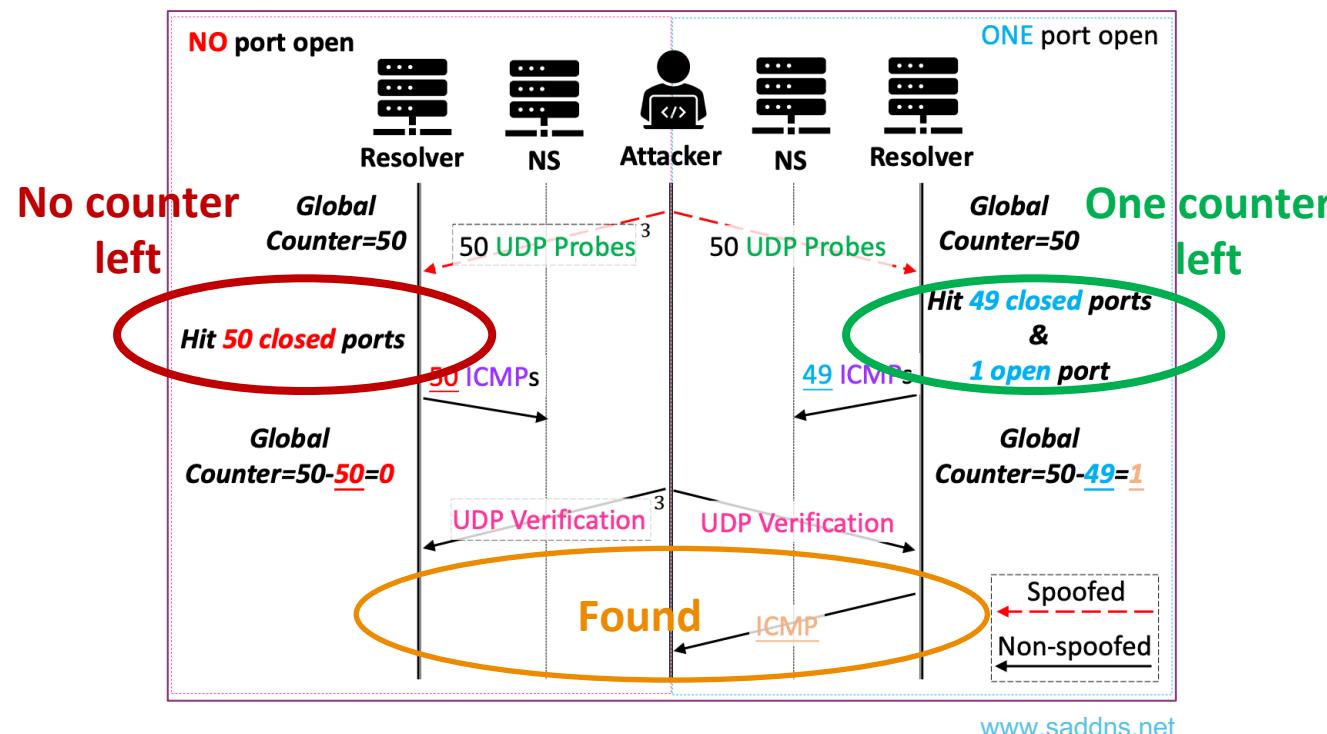


Increasing the
packet size with
the CNAME chain

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



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

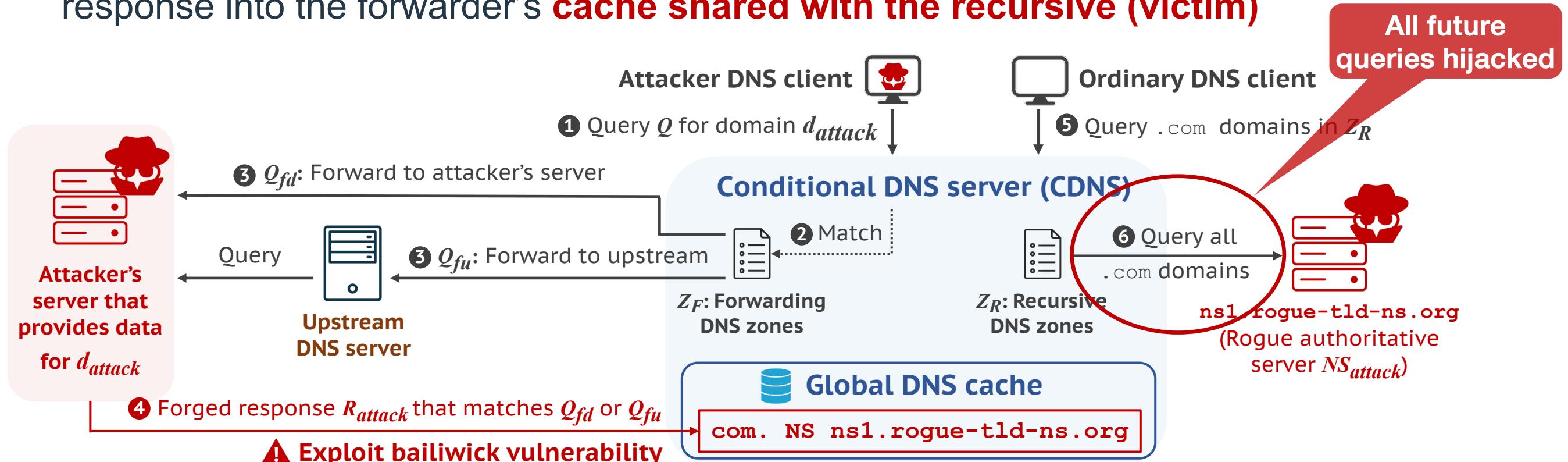
26 years later, does bailiwick checking work as desired after fixing the Kashpureff attack?

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

DNS Cache Poisoning (5/5)

➤ MaginotDNS Attack (On-/Off-path, 2023)

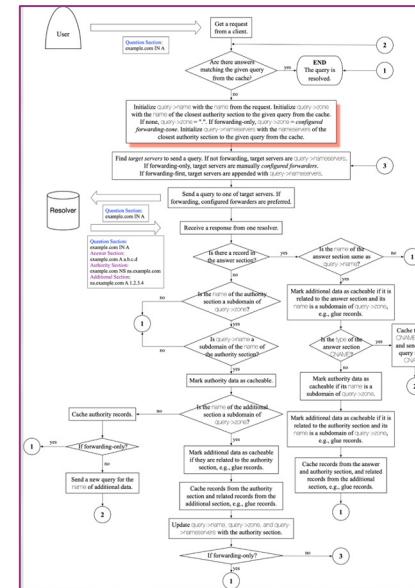
- From our NISL lab, published at USENIX Security 2023
- New attack surface: exploiting the **bailiwick checking vulnerability** to inject fake response into the forwarder's **cache shared with the recursive (victim)**



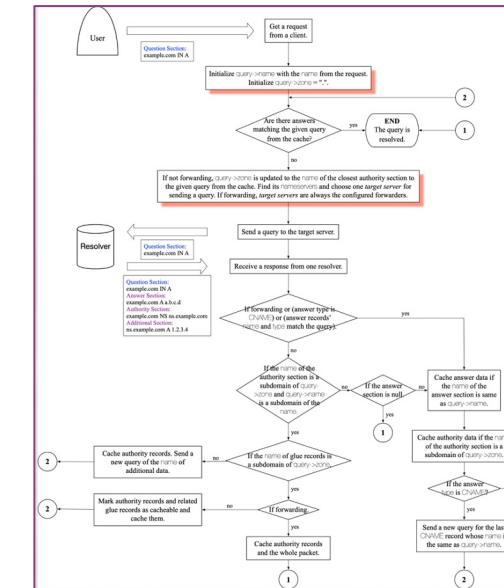
Patching the Resolver Implementation

- Mitigating the MaginotDNS Attack
 - Aligning the bailiwick checking logic between fwders & recurs

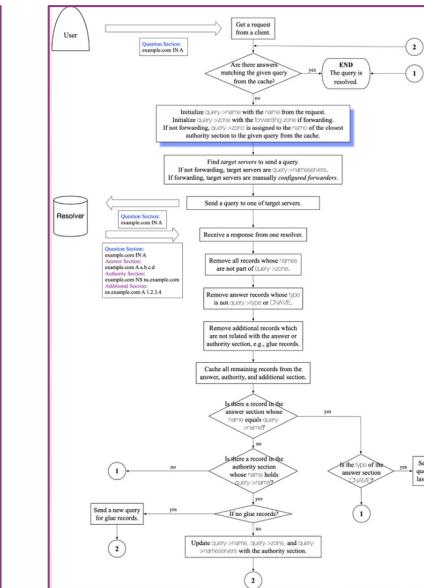
- The logic implementation of forwarders is flawed
- Adding bailiwick checking for the forwarder



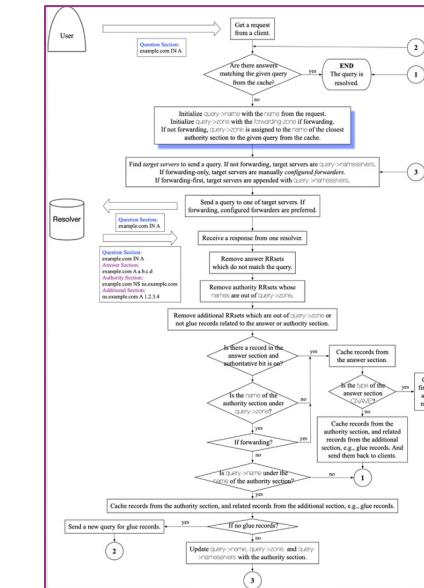
BIND



Knot



PowerDNS



Unbound

```

Algorithm 1: DNS resolution process
: A DNS Reply to clients

1 main()
2   step_0: InitQuery(Q, Request)
3   step_1: if SearchCache(Q, Cache) then
4     goto final
5   step_2: FindServers(Q, TgtSvrs)
6   step_3: SendQuery(Q, TgtSvrs)
7   step_4: ProcessResponse(Q, R)
8   if ServerIsError(Q, R) then
9     goto step 3
10  if not MatchQuery(Q, R) then
11    goto final
12  SanitizeRecords(Q, R)
13  if IsReferral(Q, R) then
14    if not IsFwding() then
15      UpdateQuery(Q)
16    goto step 2
17  if IsCNAME(Q, R) then
18    UpdateQuery(Q)
19    goto step 1
20  CacheRecords(R, Cache)
21  final: ConstructReply(Reply)
22  return Reply

23 InitQuery(Q, Request)
24   initialize Q.name, Q.type, Q.zone
25   if IsFwding() then
26     ModifyFwdQuery(Q)
27   SanitizeRecords(Q, R)
28   for RR ∈ R do
29     if OutofBailiwick(RR) then
30       remove RR from R
31   UpdateQuery(Q, R)
32   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](#)



MaginotDNS: Attacking the Boundary of DNS Caching Protection

Zhou Li | Assistant Professor, University of California, Irvine
Xiang Li | Ph.D. Candidate, Tsinghua University
Qifan Zhang | Ph.D. Student, University of California, Irvine
Date: Wednesday, August 9 | 2:30pm-3:00pm (South Seas CD, Level 3)
Format: 30-Minute Briefings
Track: Network Security

End-of-Day report

Timeframe: Freitag 11-08-2023 18:00 - Montag 14-08-2023 18:00 Handler: Michael Schlagenhauf 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 Tolias

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?

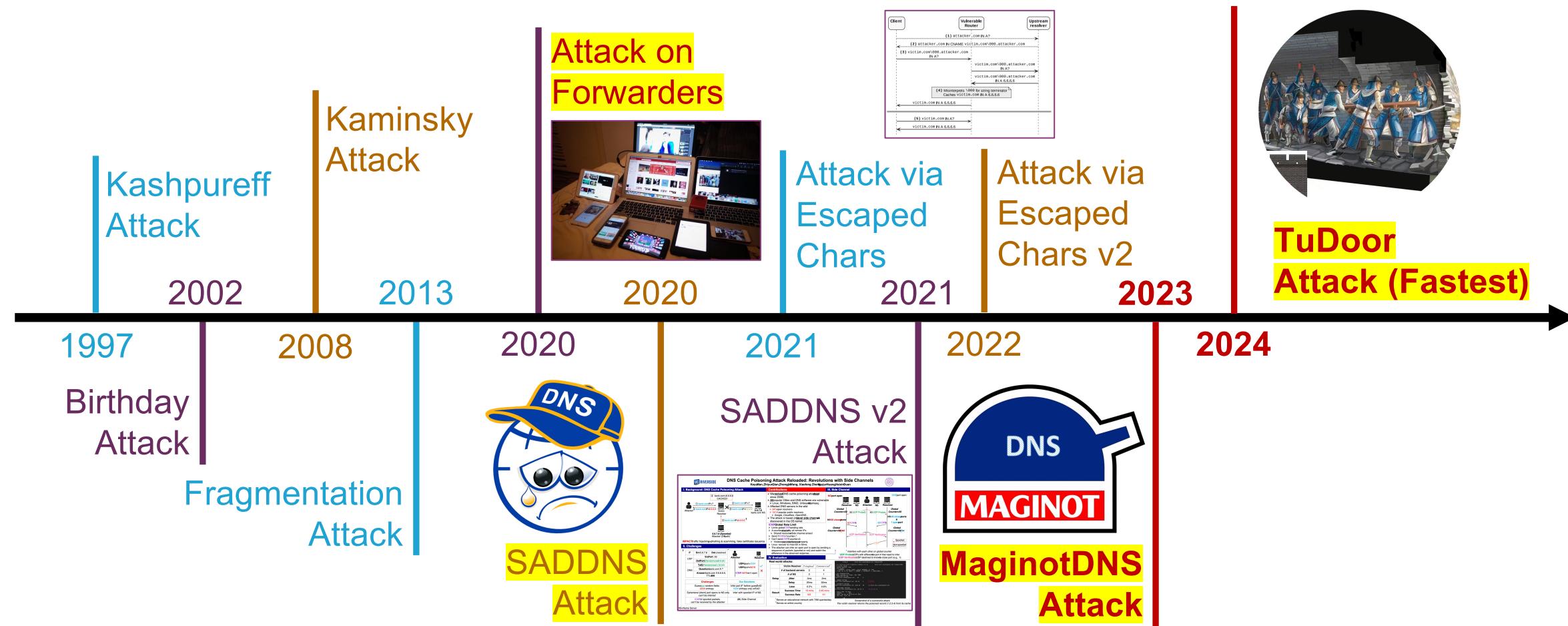
We found that the **DNS response processing logic** has never been studied thoroughly.

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



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**

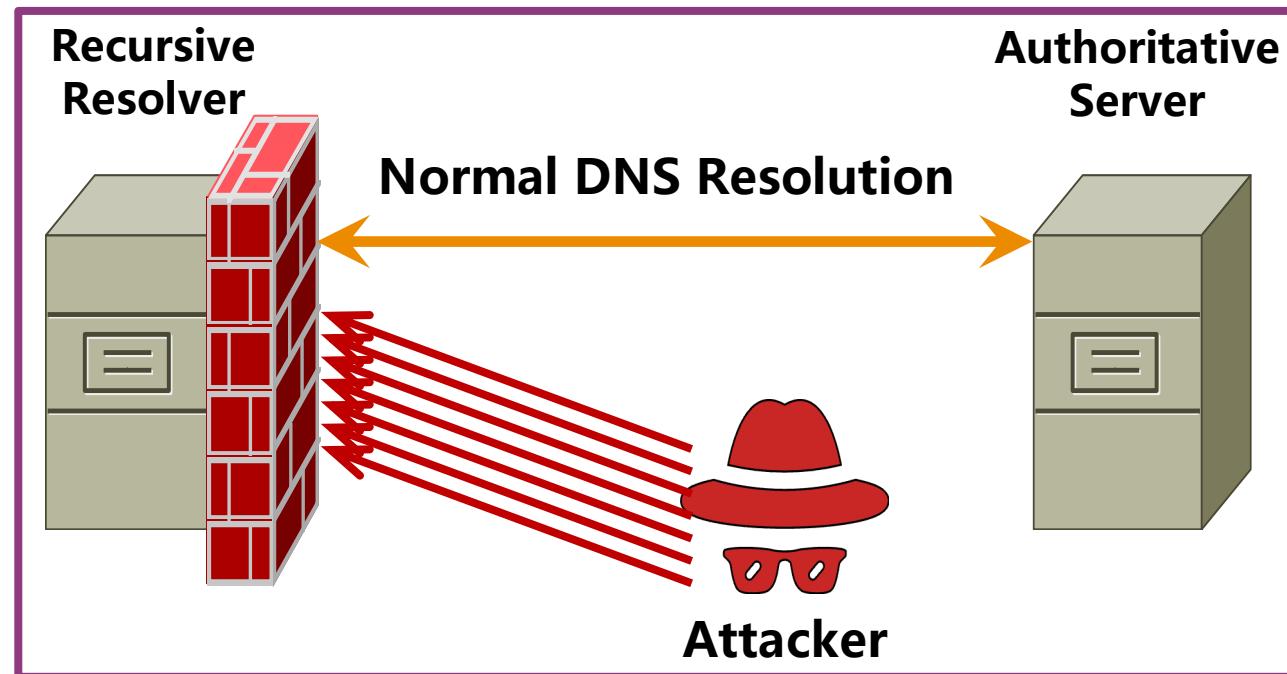
➤ Name

- Exploiting **vulnerabilities** of DNS response processing logic
- A very covert response door → like **突门** in the Great Wall
- Called the **TuDoor** attack

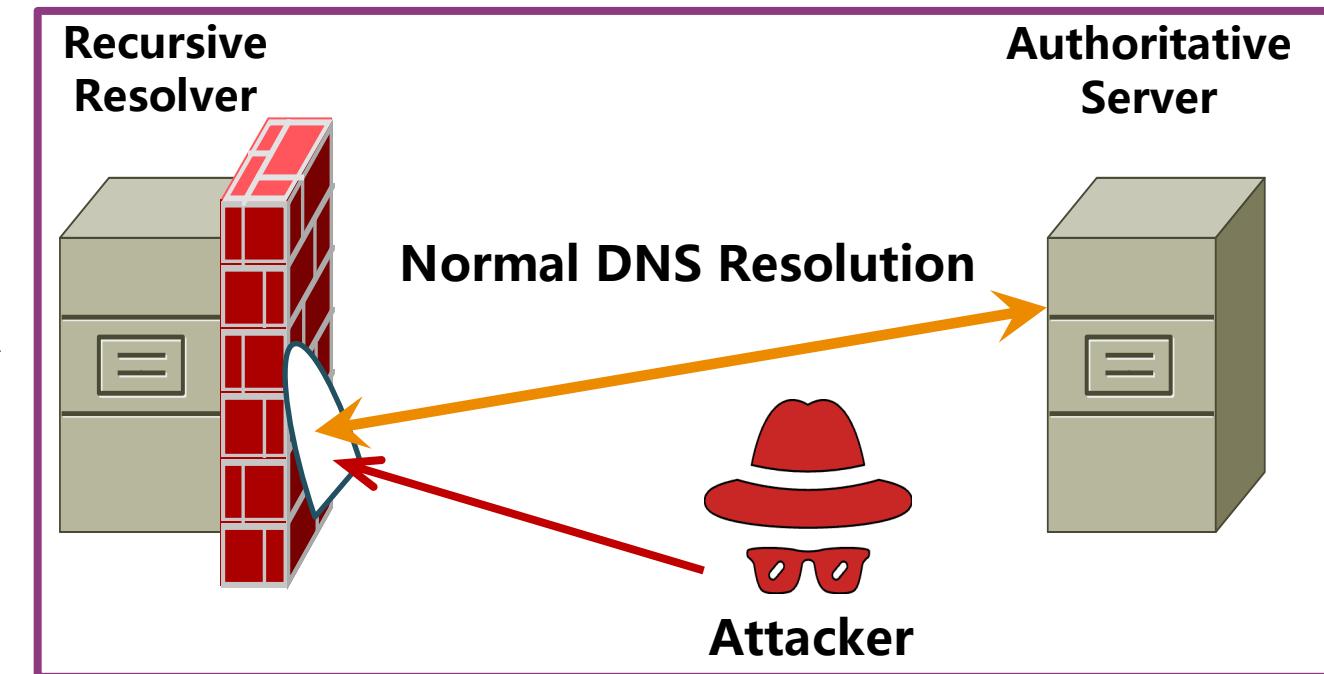


TuDoor Attack

- TuDoor in the DNS Wall
- Vulnerabilities in DNS Response Processing Logic
 - Covert side-channel exploited by attackers



Attackers need to attempt many times with a low success rate.



TuDoor Attackers just need to attack once with a success rate of 100% using side-channels.

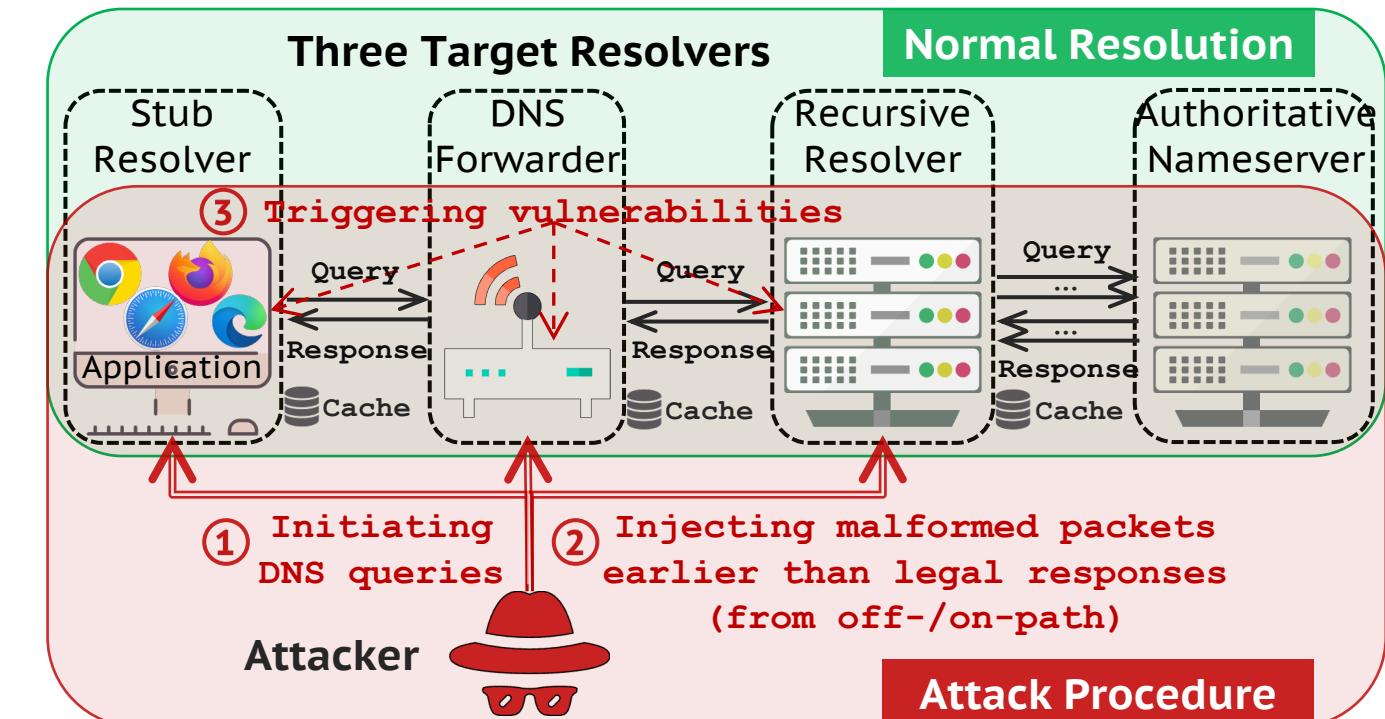
Attack Overview of TuDoor

➤ Attack Target

- ❑ Resolvers, including stub resolver, DNS forwarders, and recursive resolvers

➤ Threat Model

- ❑ Identifying the target resolver
- ❑ Triggering different vulnerabilities
- ❑ Conducting the attack

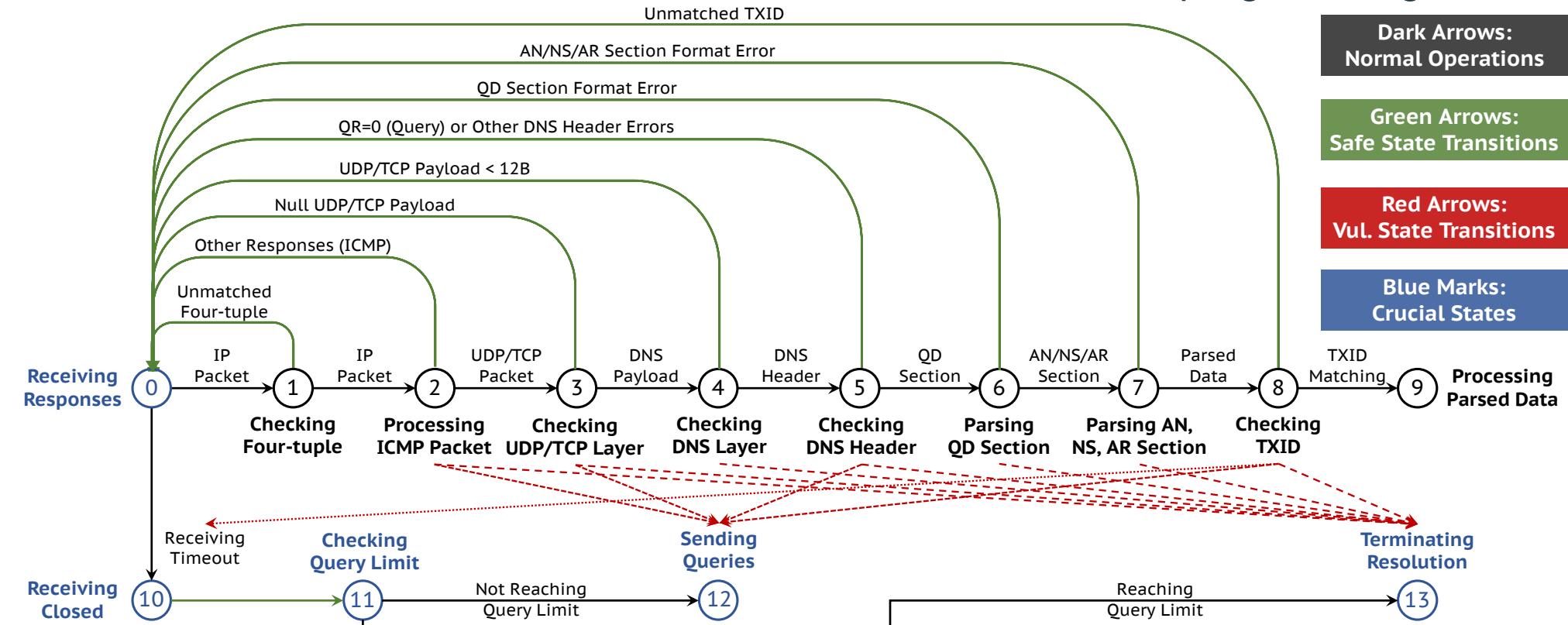


Analysis of DNS Response Processing

➤ Systematic Analysis

□ 28 DNS software → Constructing processing states

- 8 recursive resolvers, 10 DNS forwarders, 6 stub resolvers, 4 DNS programming libraries



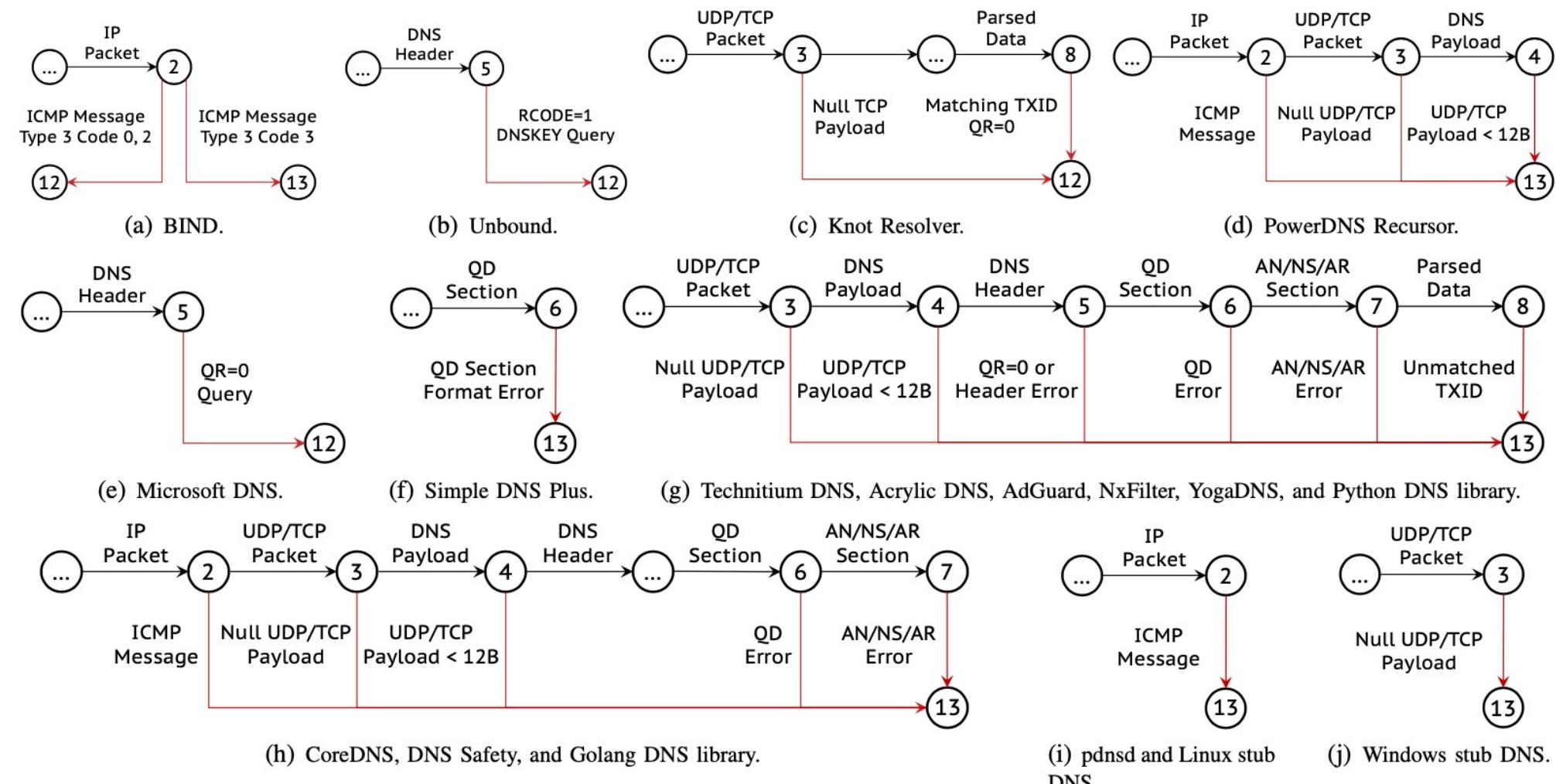
Vulnerable State Transitions

➤ DNS Response Pre-processing Implementations

□ Part software

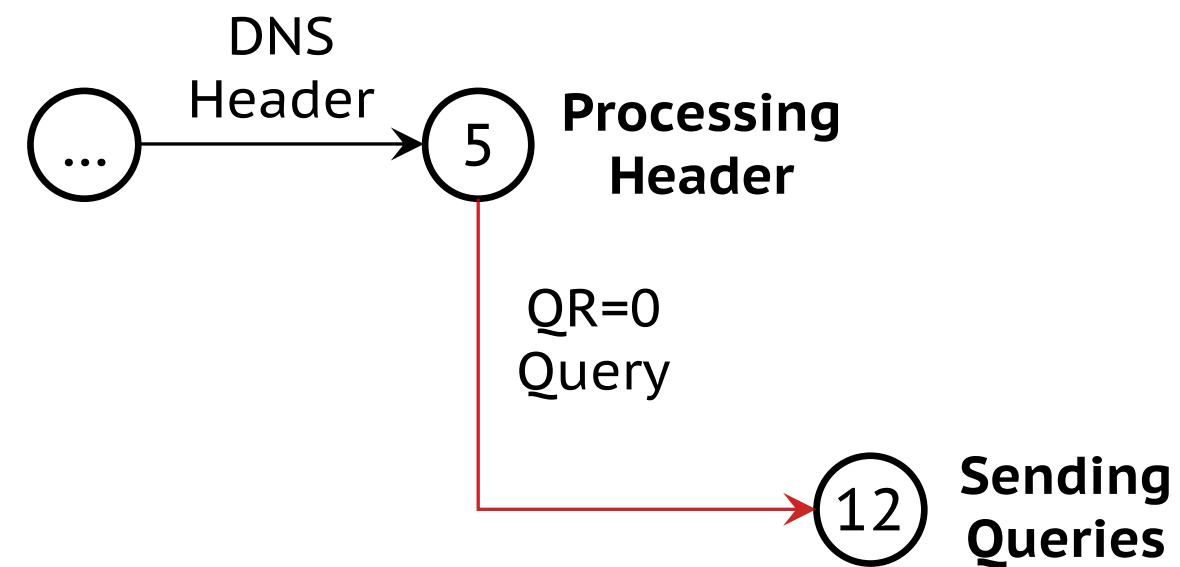
□ Red lines

○ Vulnerable



Vulnerable State Transitions

- DNS Response Pre-processing Implementations
- Microsoft DNS (MS DNS)
 - If receiving **new DNS query packets (QR=0)** when waiting for responses
 - MS DNS will **accept this new query and start new resolution for any domains**



Vulnerable DNS Software

➤ 24/28 Software

□ Vulnerable to **cache poisoning, DoS, resource consuming**

Role	Software	Cache	DoS	Resource
		Poisoning		Consuming
Recursive	BIND	No	Vul	Vul
	Unbound	No	No	Vul
	Knot	No	No	Vul
	PowerDNS	No	Vul	No
	Microsoft	Vul	No	No
	Simple DNS+	No	Vul	No
	Technitium	Vul	Vul	No
Forwarder	CoreDNS	Vul	Vul	No
	pdnsd	No	Vul	No
	Acrylic DNS	Vul	Vul	No
	AdGuard	Vul	Vul	No
	DNS Safety	Vul	Vul	No

Role	Software	Cache	DoS	Resource
		Poisoning		Consuming
Forwarder	Dual DHCP	Vul	No	No
	NxFilter	Vul	Vul	No
	YogaDNS	Vul	Vul	No
Stub	Linux	No	Vul	No
	Windows	No	Vul	No
	MacOS	No	Vul	No
	IOS	No	Vul	No
Library	ChromeOS	No	Vul	No
	Python	No	Vul	No
	Golang	No	Vul	No
	JavaScript	No	Vul	No
	Java	No	Vul	No

Vulnerable Public DNS Services

➤ 1/42 Public Resolver

Vulnerable to **cache poisoning**

114DNS

➤ 17/42 Public Resolver

Vulnerable to **DoS**



Baidu DNS

CIRA Canadian
Shield
Free public DNS

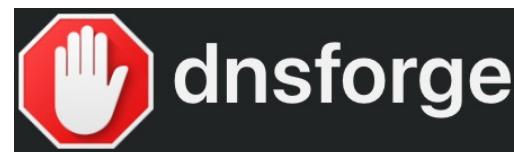
OpenDNS



1.1.1.1



cz.nic



DNSify DNS



Quad101

<--> LibreDNS

OpenNIC DNS



Strongarm DNS

Attack Steps of TuDoor

➤ Three Attacks

- Cache Poisoning

- DoS

- Resource Consuming

➤ Attack steps

- Example: **cache poisoning**

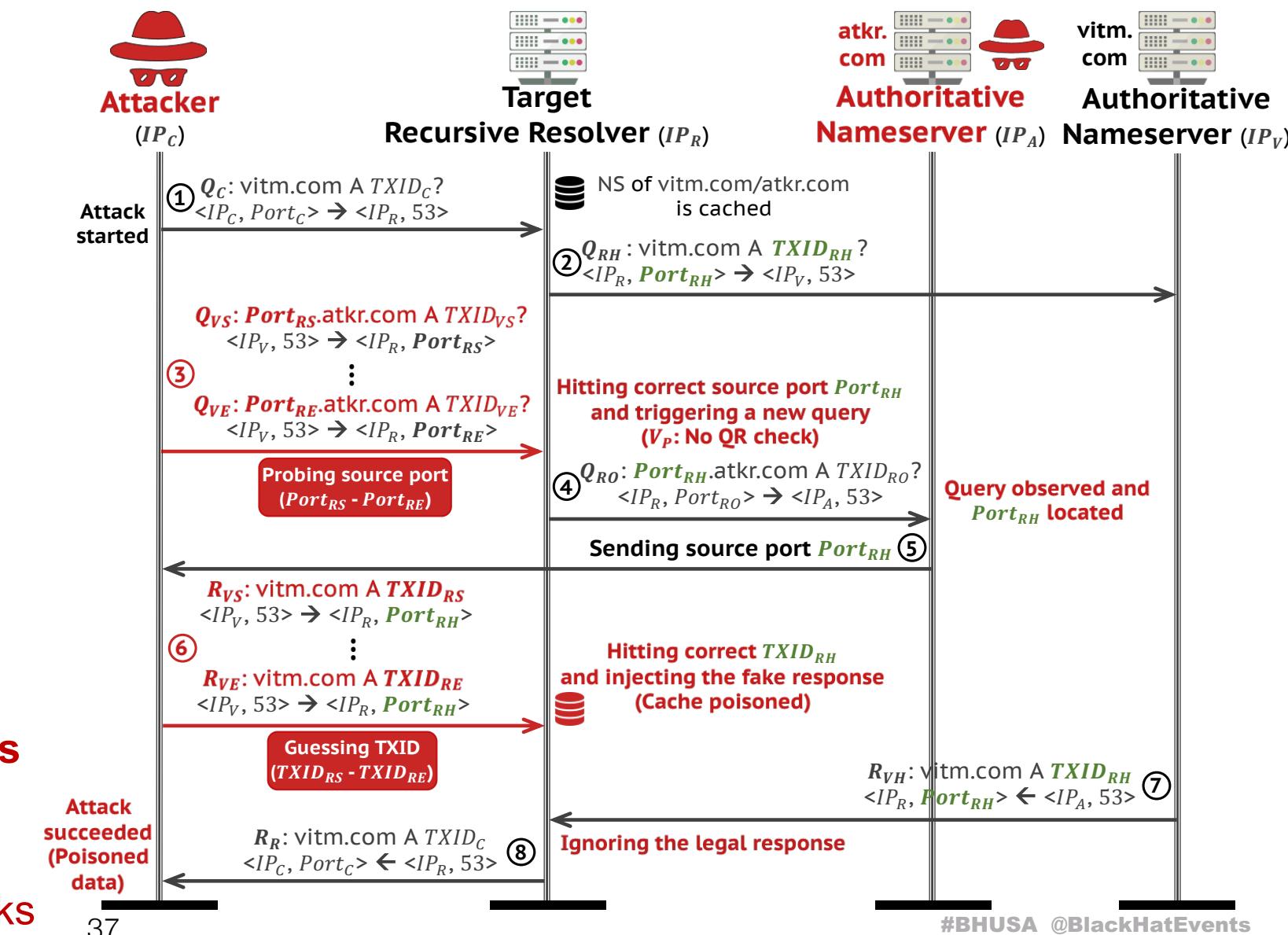
- One new side-channel vulnerability

- Exposing the source port

- Attackers just need to send **<65,535 pkts**

- **Attack time: avg. 425ms**

- 200 – 1,000 times faster than prior attacks



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 ⭐ 316 🗃 47

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%)

Type	Resolver number and percentage	
Vulnerable	Cache poisoning	205,984 (11.2%)
	DoS	216,317 (11.8%)
	Resource consuming	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

➤ Root Cause

- Poor DNS response pre-processing 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: tudoor.net

TuDoor Attack: Test

Click the button below to test if your DNS resolver is vulnerable to TuDoor attack.

Test!!

Vulnerable!

Black Hat Sound Bytes

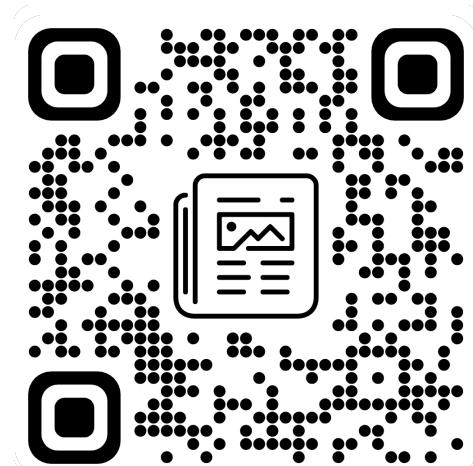
- **The DNS RFCs and specifications are not clear to provide a deterministic definition for each operation, hence leaving a large attack window for ambiguous implementations.**
 - We should check the RFC's essential specifications.
- **The DNS implementations are not consistent across software, even for identical client queries.**
 - This inconsistency is likely to conceal possible risks, which should be thoroughly researched and evaluated.
- **The original DNS mechanism is insufficient to defend against several types of attacks.**
 - To improve it, we should propose new patches or redesign some structures.

Wrap-up

Thanks for listening!
Any question?

Xiang Li, Nankai University

Paper



Tool

