AUGUST 9-10, 2023

BRIEFINGS

MaginotDNS: Attacking the Boundary of DNS Caching Protection

Speaker(s): Zhou Li

Contributor(s): Xiang Li and Qifan Zhang

August 2023

About Us

Zhou Li
Assistant Professor at UC Irvine
Research interests: DNS, Graph Security analytics (GSA), …

Xiang Li
PhD at Tsinghua University

Qifan Zhang
PhD at UC Irvine

Attack Impact

Our MaginotDNS attack could poison a whole TLD, e.g., .com, at one round.

All domains under that TLD can be hijacked.

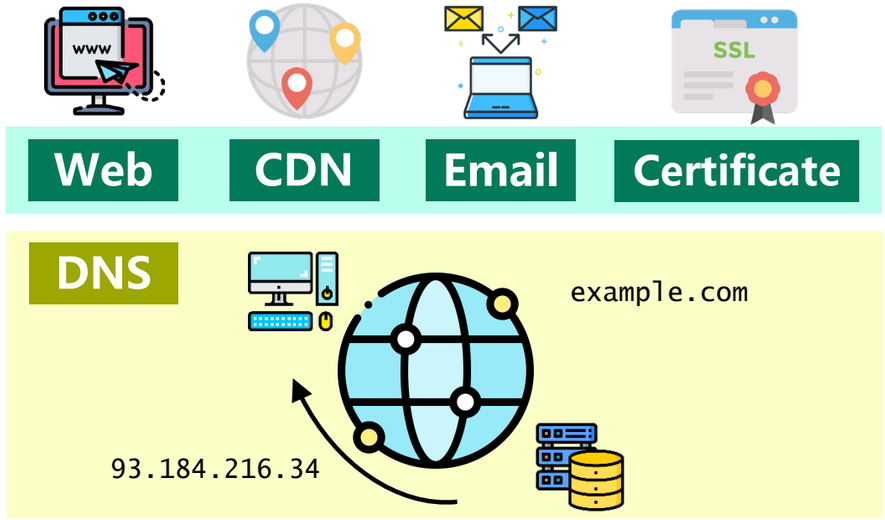
Outline

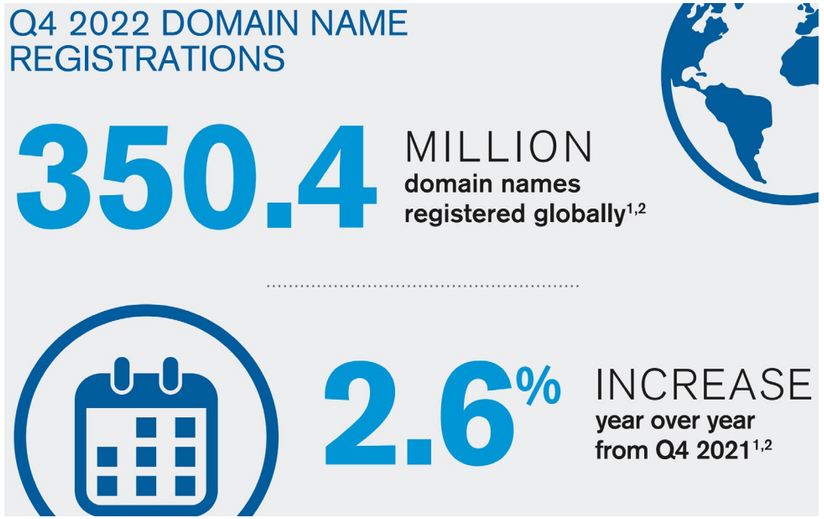
- DNS overview
- DNS cache poisoning
- MaginotDNS workflow
- Attack demo
- Large-scale scanning
- Discussion & conclusion

Domain Name System (DNS)

➢ DNS Overview

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

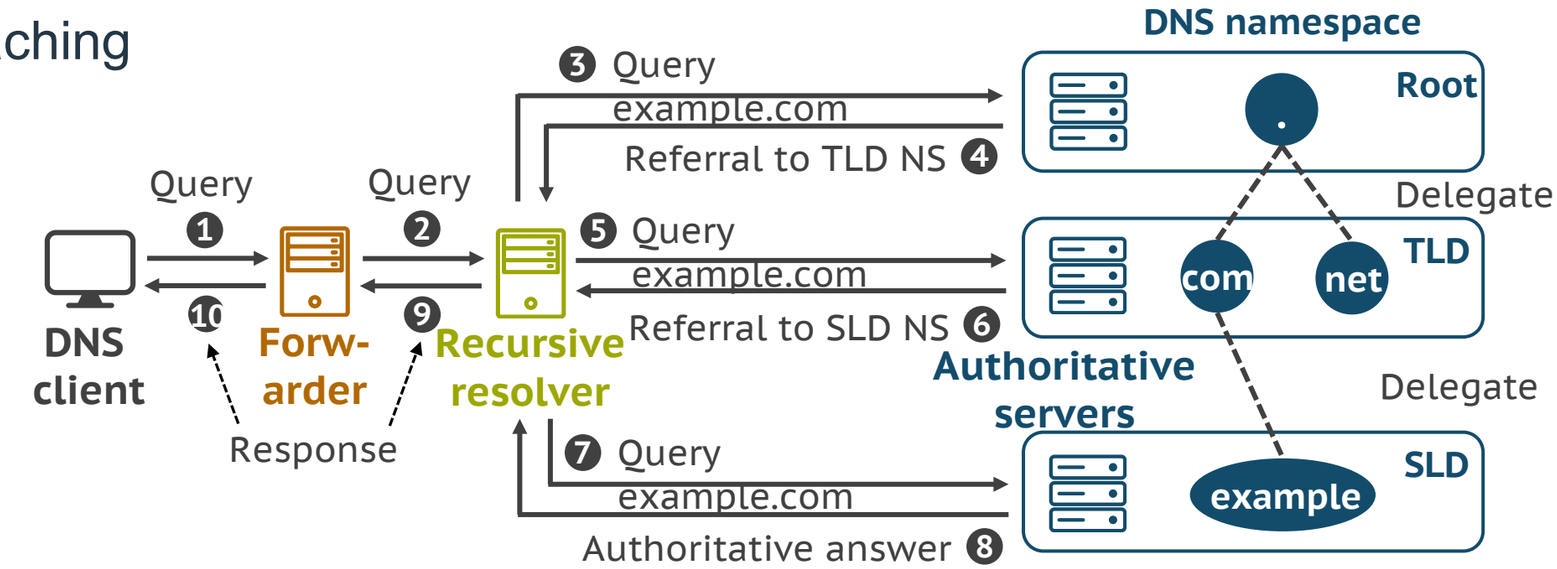




verisign.com/dnib

DNS Resolution

- **Resolution Process**
  - Primarily over UDP
  - Iterative and recursive
  - Record caching

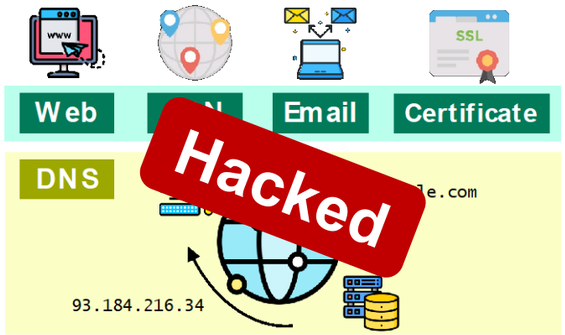


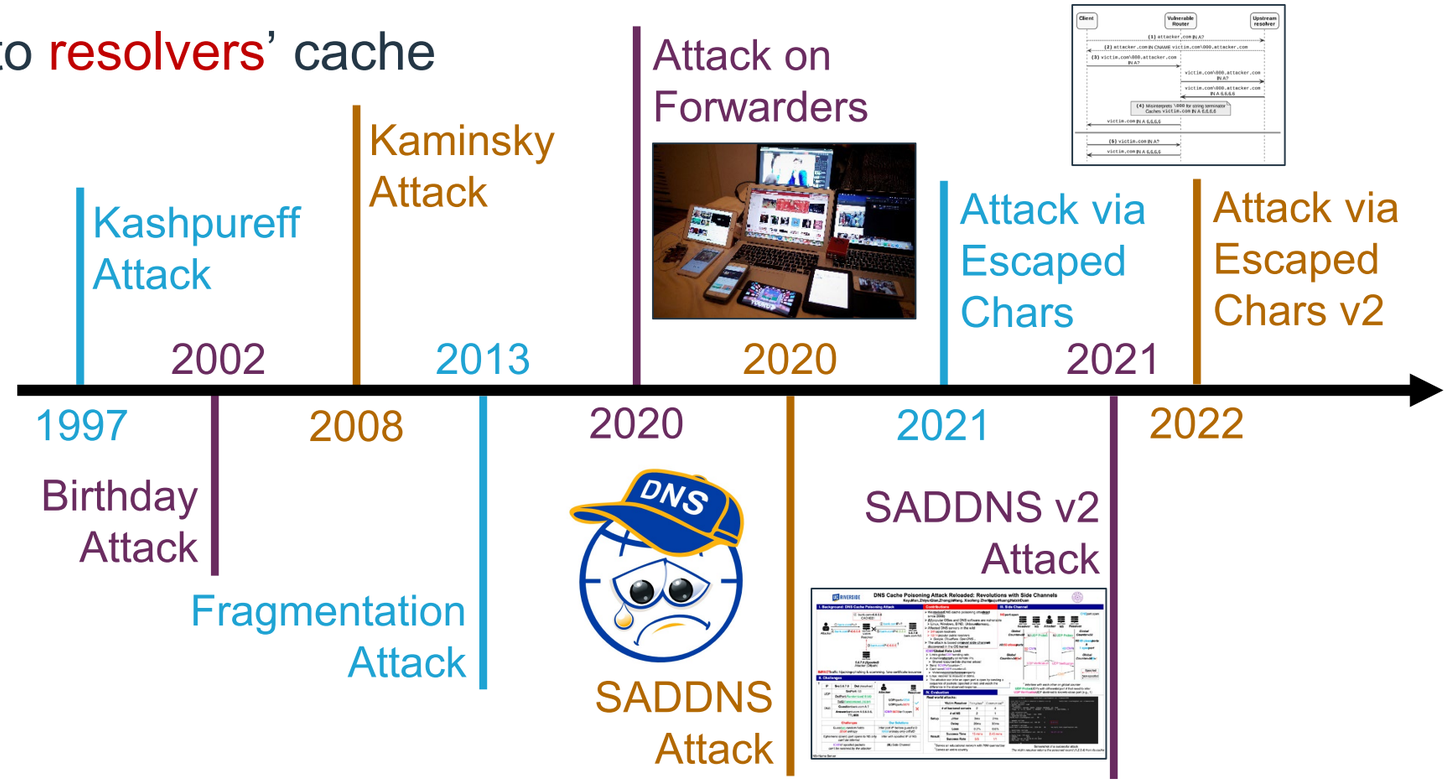
Outline

- DNS overview
- **DNS cache poisoning**
- MaginotDNS workflow
- Attack demo
- Large-scale scanning
- Discussion & conclusion

DNS Cache Poisoning

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





MaginotDNS

Massive DNS poisoning attacks in Brazil

INCIDENTS 07 NOV 2011 2 minute read

Unpatched DNS Bug Puts Millions of Routers, IoT Devices at Risk

Help Net Security
October 26, 2021

Share

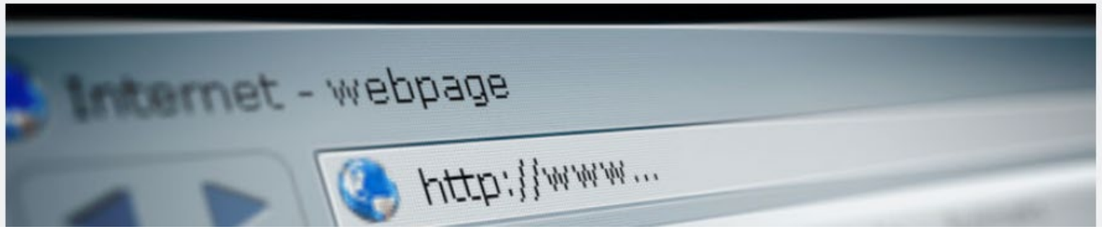
72% of organizations hit by DNS attacks in the past year

MASQUERADE PARTY —

DNS cache poisoning, the Internet attack from 2008, is back from the dead

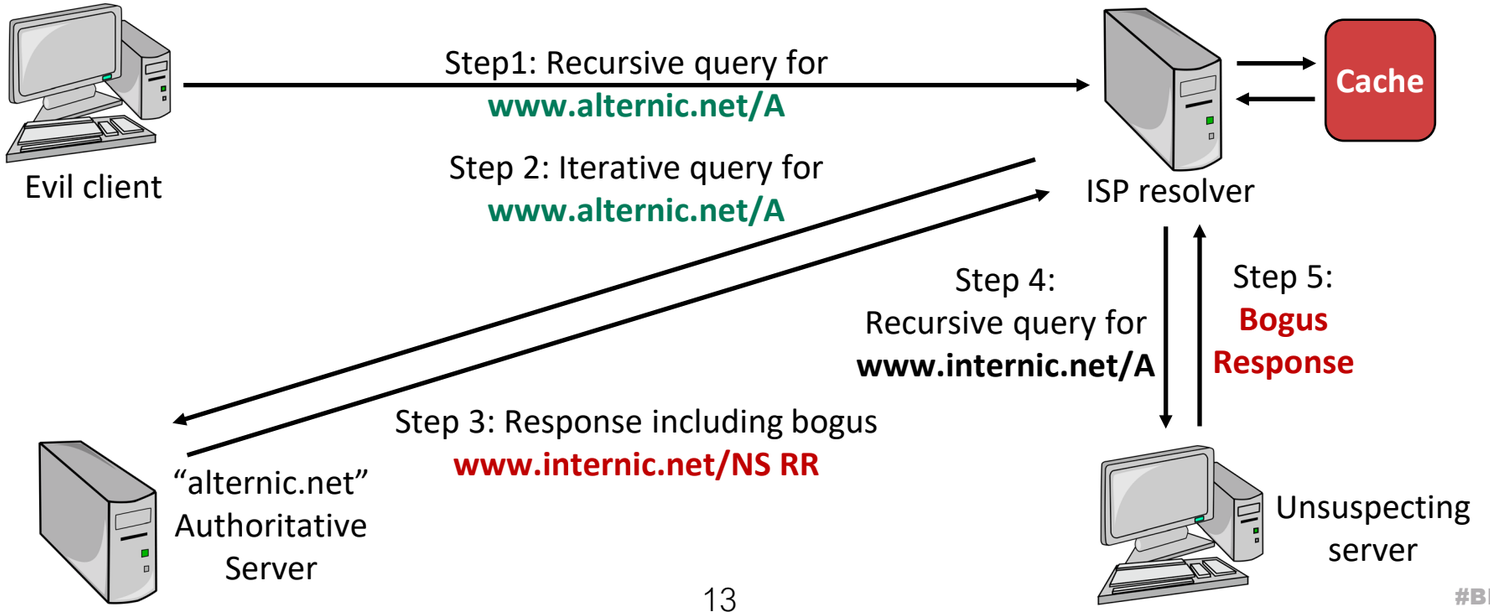
A newly found side channel in a widely used protocol lets attackers spoof domains.

DAN GOODIN - 11/12/2020, 6:30 AM



On-path DNS Cache Poisoning

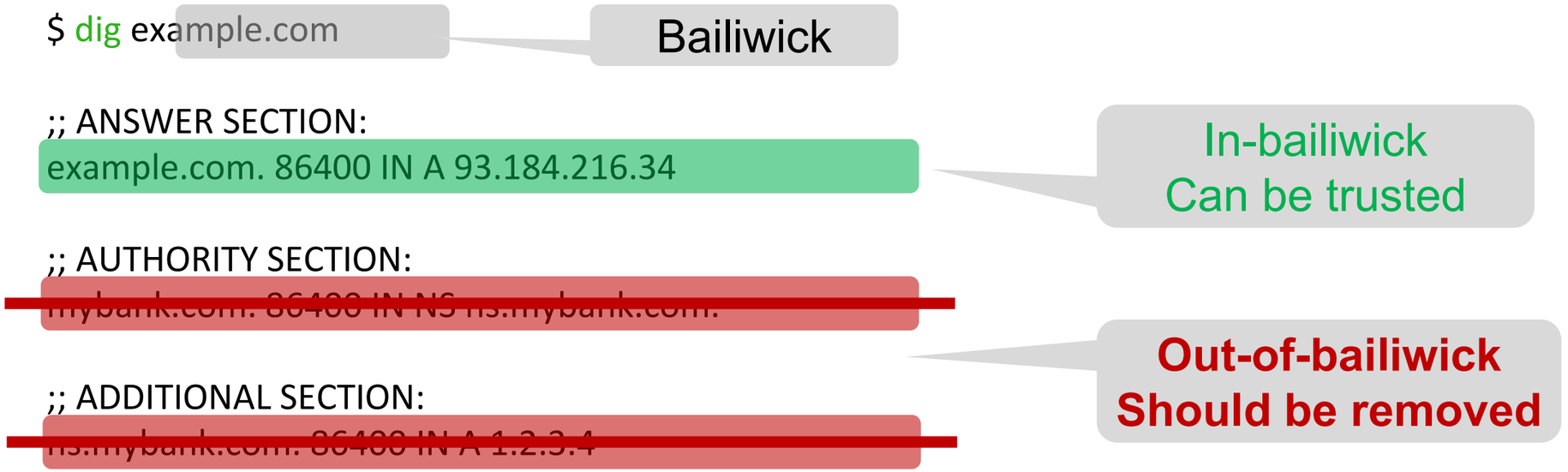
- **Kashpureff Attack (on-path, 1997)**
  - Method: returning forged responses from the authoritative
  - Result: resolver accepting all records in the response
  - Cause: lacking data verification (**bailiwick rules**)



DNS Bailiwick Rules

➢ Mitigating the Kashpureff Attack

- ❑ Record validation when storing cache entries
- ❑ Checking for "**in bailiwick**" in response data: **answer records must be from the same domain as the requested name**



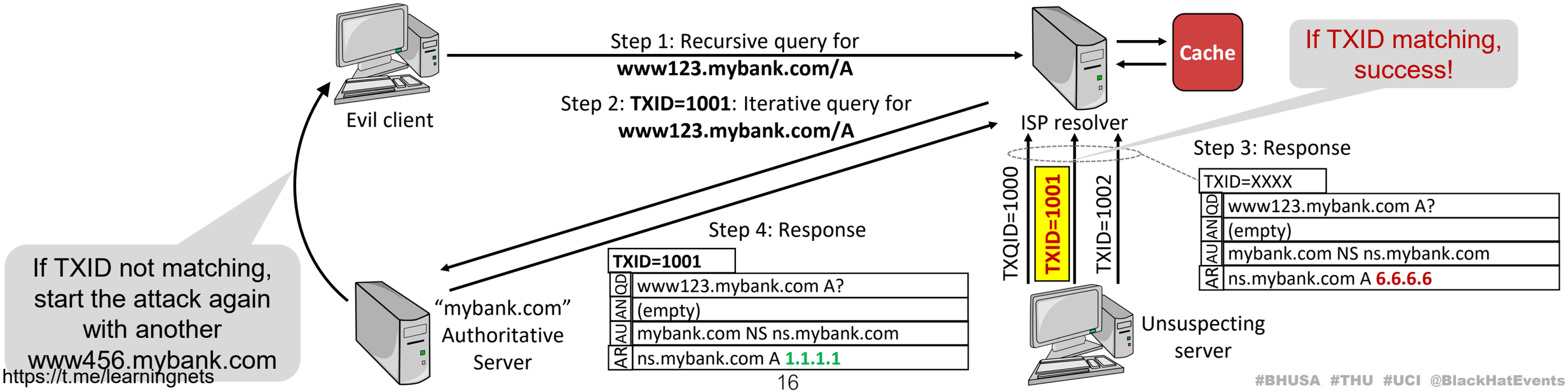
Takeaway

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.

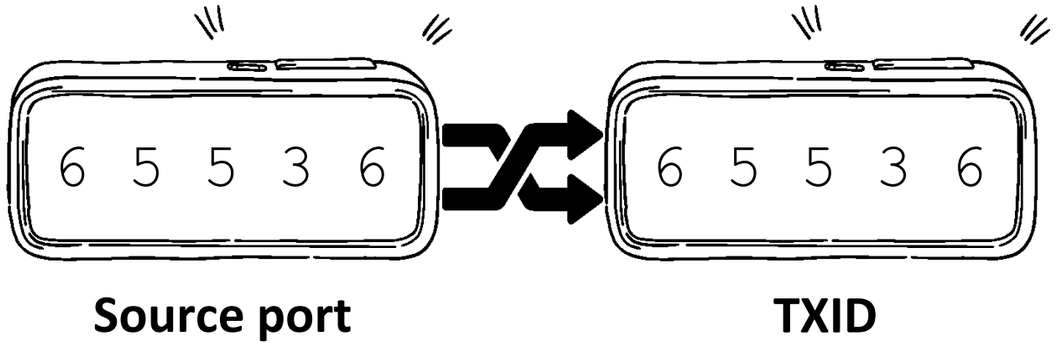
Off-path DNS Cache Poisoning

- **Kaminsky Attack (Off-path, 2008)**
  - Method: injecting forged responses with the birthday attack
  - 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.

Outline

- DNS overview
- DNS cache poisoning
- **MaginotDNS workflow**
- Attack demo
- Large-scale scanning
- Discussion & conclusion

Question

Are bailiwick checking and port randomization good enough?

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

MaginotDNS Attack

- **What is the MaginotDNS attack**
  - A new powerful DNS cache poisoning attack against **CDNS resolvers**
  - Can be launched from either **on-path** or **off-path**
  - Can poison **arbitrary domains** including **TLDs**, such as .com and .net
- **Name**
  - Exploiting **vulnerabilities** of bailiwick checking to bypass itself
  - Working like breaking the **Maginot Line** → **MaginotDNS**

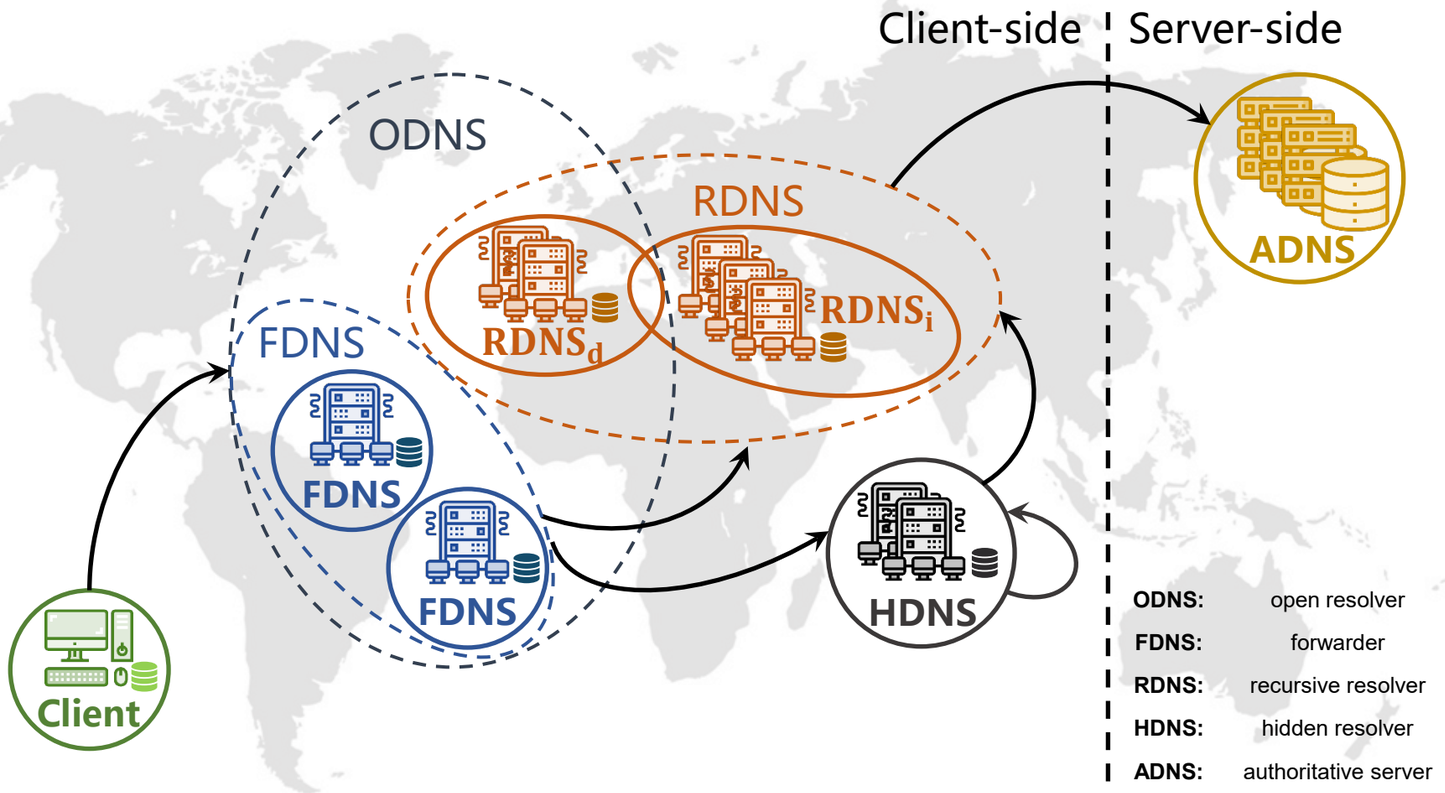
Question

What is the CDNS resolver?

A **conditional DNS resolver** with both **recursive** and **forwarding** query modes.

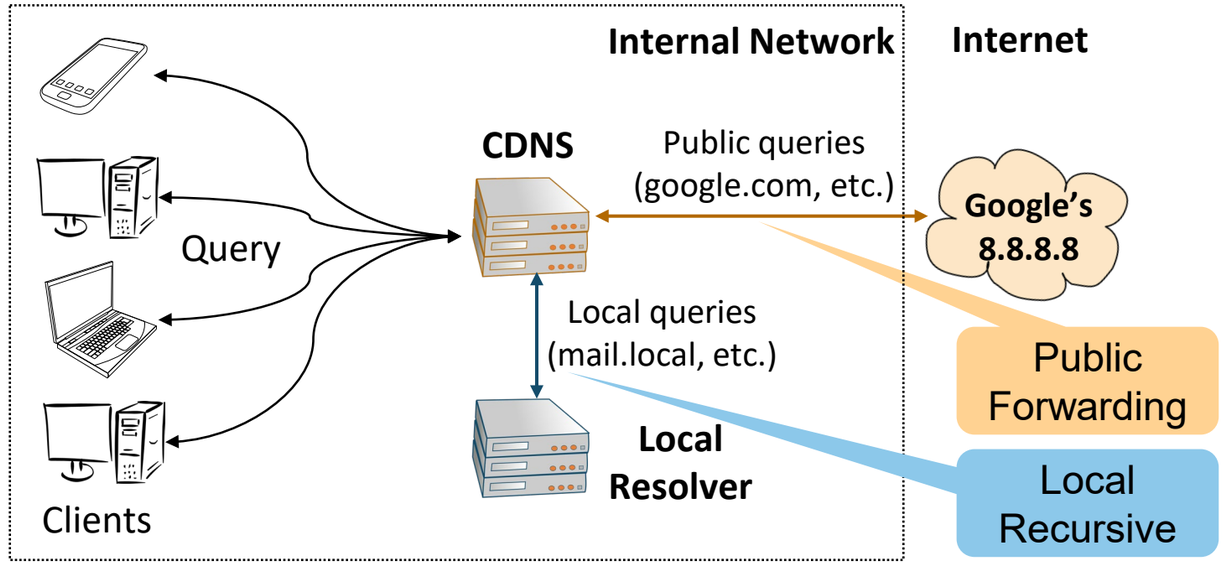
DNS Resolvers

- **Worldwide**
- **Multiple Roles**
  - Recursive, forwarder
  - Hidden DNS (HDNS)
- **Complex Interaction**
- **CDNS**
  - One of HDNSes
  - Never been studied



Attack Target: CDNS

- **Conditional DNS Resolver (CDNS)**
  - Forwarder + recursive resolver (shared cache)
  - 2 query zones used for different resolution
    - Z_F: domains for forwarding queries
    - Z_R: domains for recursive queries
- **Usage Scenarios**
  - Enterprise: splitting networks
  - ISP: reducing heavy traffic cost



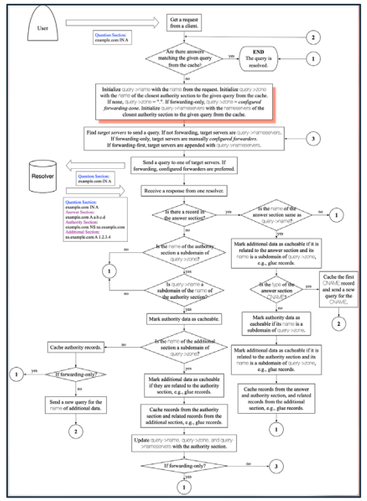
Attack Overview of MaginotDNS

- **Threat Model**
  - Assuming we discovered a CDNS and inferred its Z_F & Z_R
  - Attacking the forwarding mode
- **Why forwarding mode?**
  - Bailiwick checking of the recursive mode is well enforced
  - But the forwarder mode is not
  - Since they share the same global DNS cache
  - We can exploit the weak forwarder mode to attack the well-protected recursive mode
    - → Breaking the boundary of DNS cache protection

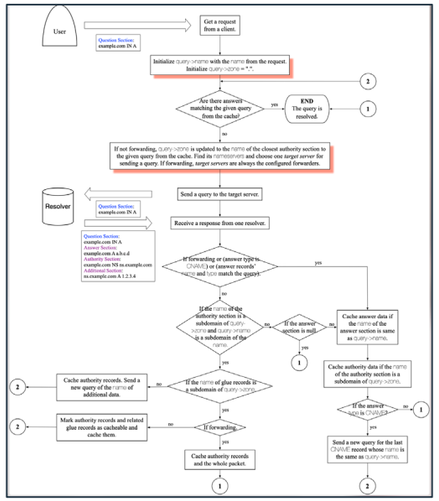
Software Analysis

➢ Finding Vulnerable Software

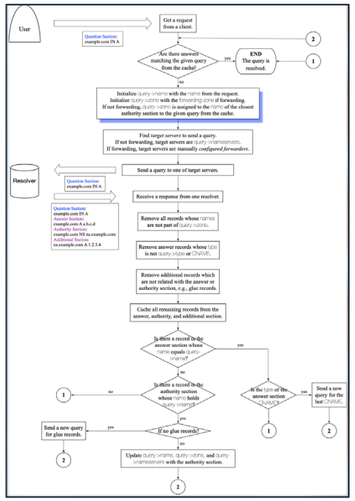
- ❑ In depth **bailiwick checking implementation** analysis
- ❑ Via source code review, debugging, and testing
- ❑ 8 mainstream DNS software, e.g., BIND and Microsoft DNS



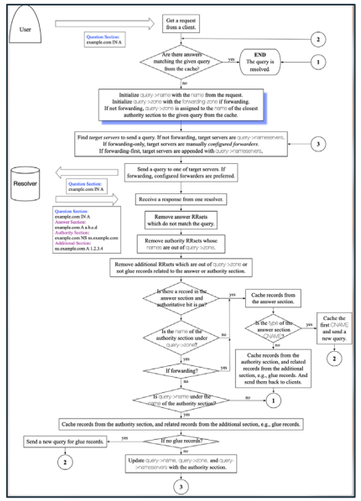
BIND



Knot



PowerDNS



Unbound

Inconsistent bailiwick checking implementations

Root Cause & Vulnerable Software

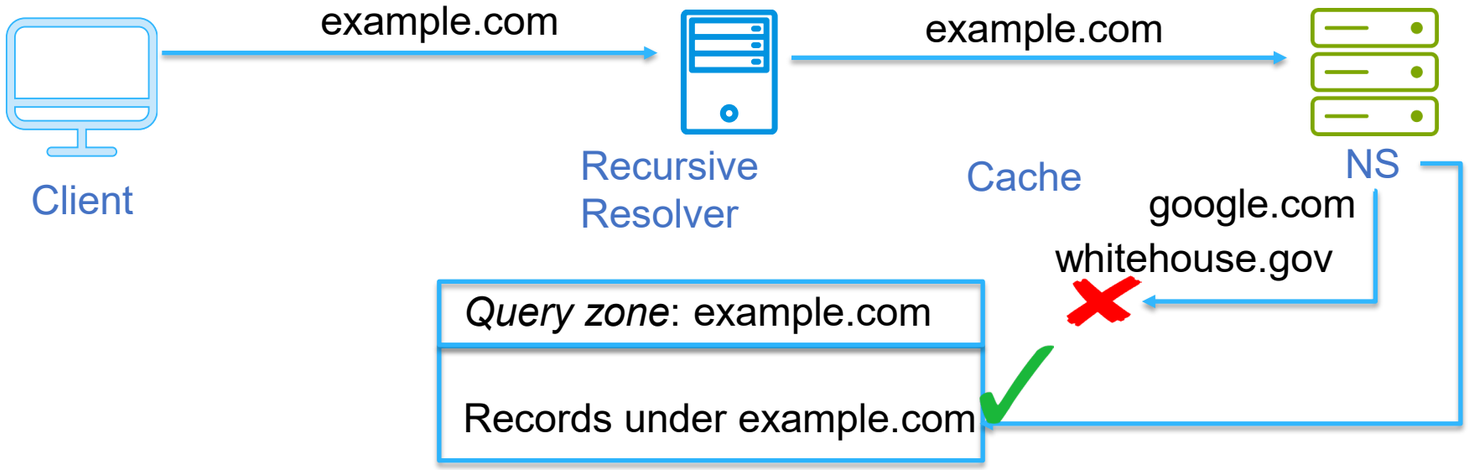
- **General Bailiwick Checking Logic**
  - Summarized by us
- **Root Cause**
  - In the `InitQuery` function:
    - `Qry.zone` is set to root → all records is **in-bailiwick** (root's subdomains)
- **Vulnerable Software**

DNS Software	Forwarding	Recursive	Vulnerable
BIND9	Enabled	Enabled	**Yes**
Knot Resolver	Enabled	Enabled	**Yes**
Microsoft DNS	Enabled	Enabled	**Yes**
Technitium	Enabled	Enabled	**Yes**

Algorithm 1: DNS resolution process

```
input  : A DNS Request from clients
output : A DNS Reply to clients
1  main()
2      step_0: InitQuery(Q, Request)
3      step_1: if SeachCache(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
```

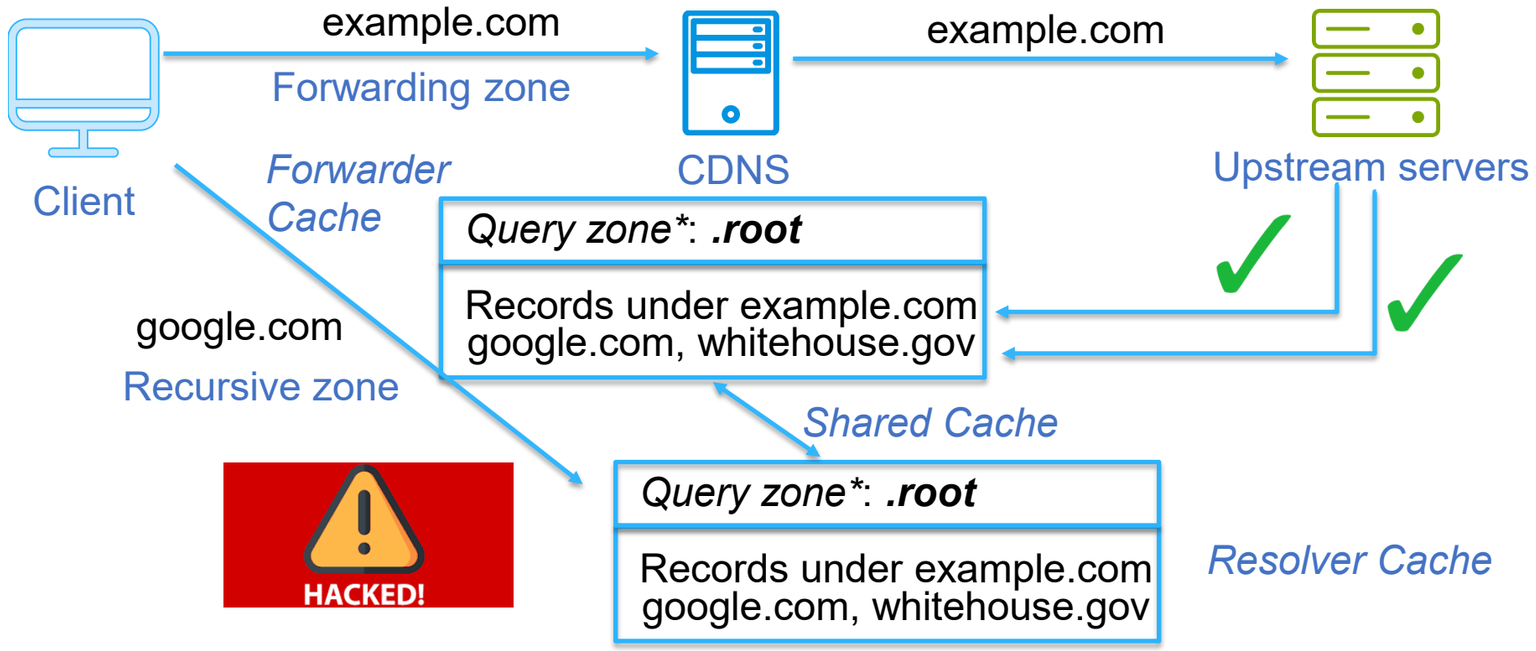
Bailiwick Checking (Done Right)



Bailiwick Checking (Done Wrong)

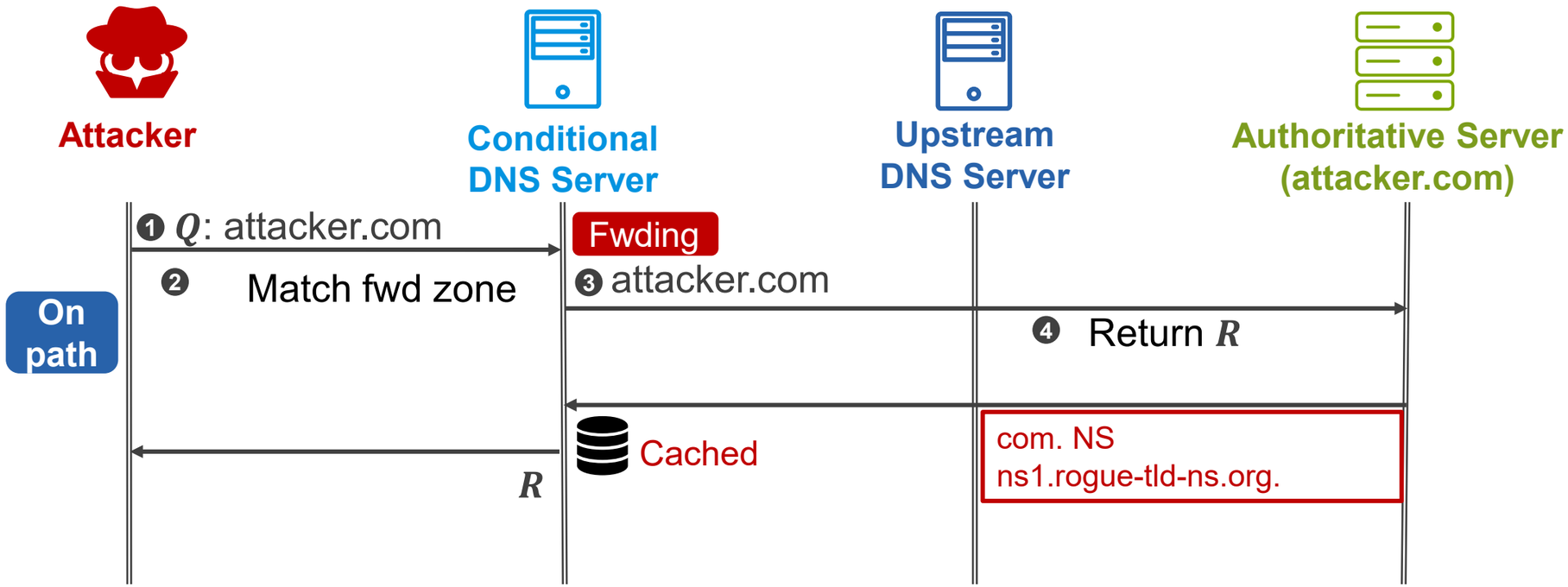
Forwarding zone: example.com

Recursive zone: {domains}-example.com



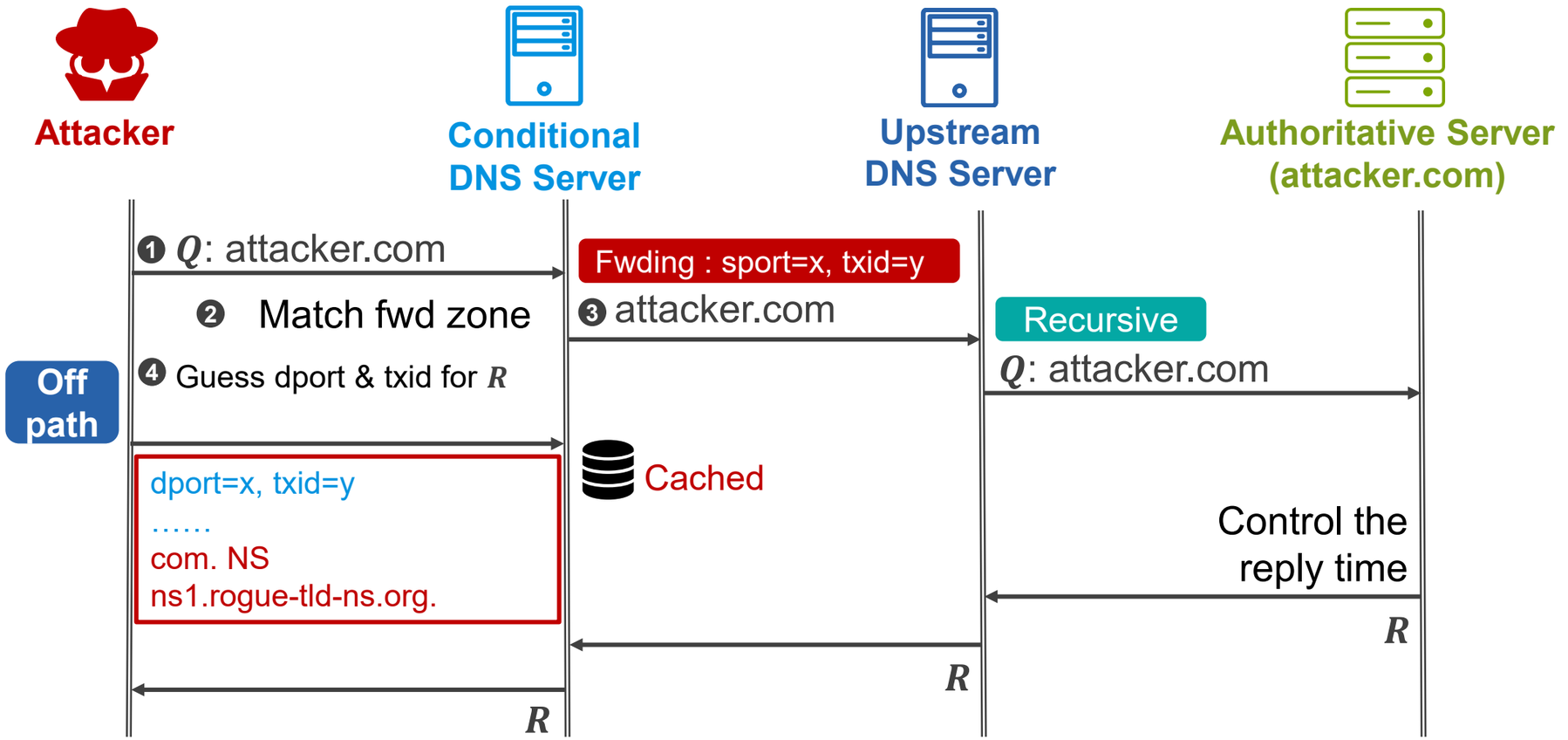
Attack Steps of MaginotDNS (On-path)

- Returning fake responses **directly**
- **BIND**, **Microsoft DNS**, **Knot**, and **Technitium**



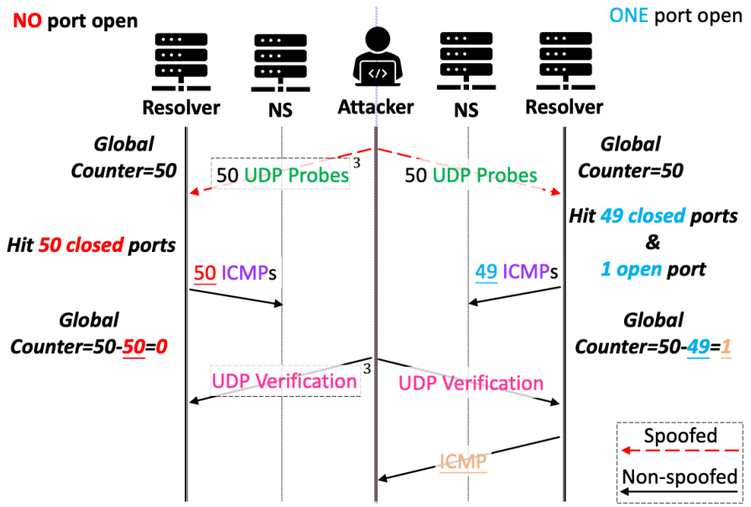
Attack Steps of MaginotDNS (Off-path)

- Guessing source port & TXID
- **Microsoft**: **new port vulnerability**
- **BIND9**: using the SADDNS attack



Off-path Attack on BIND9

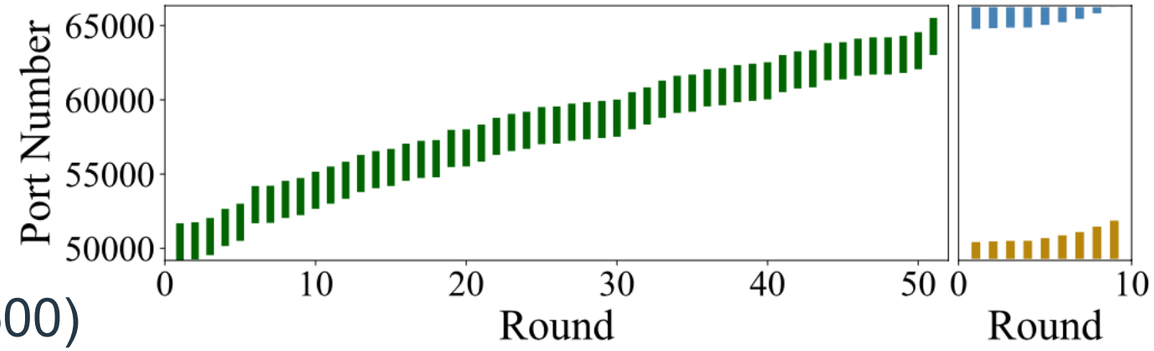
- **Guessing Source Port**
  - ❑ We use SADDNS to infer the source port
  - ❑ ICMP rate-limit side-channel (check the SADDNS paper for details)
- **Brute-forcing TXID**
- **Attack analysis**
  - ❑ Source port range: 32,768 - 60,999 (28,232)
  - ❑ Query timeout: 1.2s, guessing 50 ports each round
  - ❑ **Success rate** after 3,600 rounds:
    - $1 - [(28{,}232 - 50)/28{,}232]^{3{,}600} = 99.8\%$



SADDNS https://www.saddns.net/

Off-path Attack on Microsoft DNS

- **Guessing Source Port**
  - ❑ We found MS DNS only uses **~2,500 source ports** for resolution
  - ❑ 2,500 ports are **all in the open state** (SADDNS not working)
  - ❑ **Brute-forcing** all 2,500 ports
- **Brute-forcing TXID**
- **Attack analysis**
  - ❑ Source port range: probing in advance (2,500)
  - ❑ Query timeout: 5s, guessing 20 ports each round
  - ❑ **Success rate** after 720 rounds:
    - $1 - [(2{,}500 - 20)/2{,}500]^{720} = 99.7\%$



Source Port Range Examples of Microsoft DNS

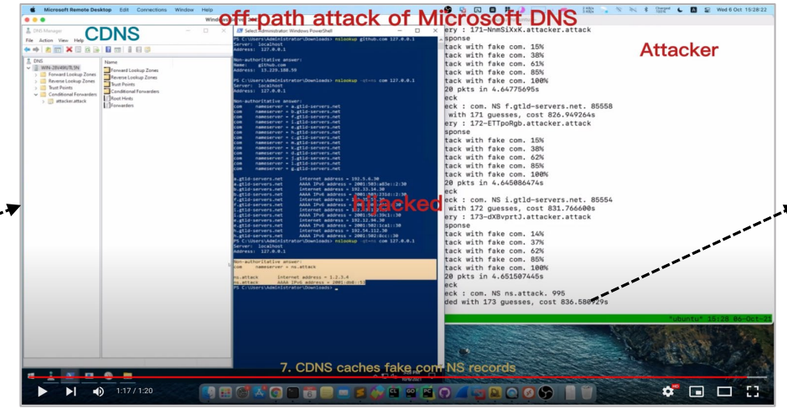
Outline

- DNS overview
- DNS cache poisoning
- MaginotDNS workflow
- **Attack demo**
- Large-scale scanning
- Discussion & conclusion

MaginotDNS Attack Demos

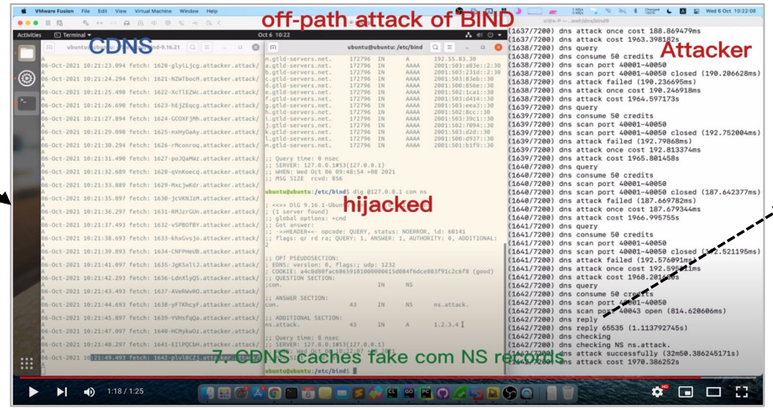
- **On-path Attack**
  - The result is determinative
- **Off-path Attack**
  - Microsoft: **avg. 802s**
  - BIND9: **avg. 790s**

Watch videos here.




```
Mon Aug  9 03:31:01 2021 : (2/360) dns query : 2-BatHkHSX.idealeer.com
Mon Aug  9 03:31:01 2021 : (2/360) dns response
Mon Aug  9 03:31:03 2021 : (2/360) dns attack with fake com. 15%
Mon Aug  9 03:31:04 2021 : (2/360) dns attack with fake com. 37%
Mon Aug  9 03:31:05 2021 : (2/360) dns attack with fake com. 60%
Mon Aug  9 03:31:06 2021 : (2/360) dns attack with fake com. 85%
Mon Aug  9 03:31:06 2021 : (2/360) dns attack with fake com. 100%
Mon Aug  9 03:31:06 2021 : to 202.112.238.57 : 1310720 pkts in 4.632276358s
Mon Aug  9 03:31:06 2021 : (2/360) dns check
Mon Aug  9 03:31:06 2021 : (2/360) dns check : com. NS gtld-servers.attack.
Mon Aug  9 03:31:06 2021 : dns attack succeeded with 2 guesses, cost 10.079395433s
```

Log of Attacking Microsoft

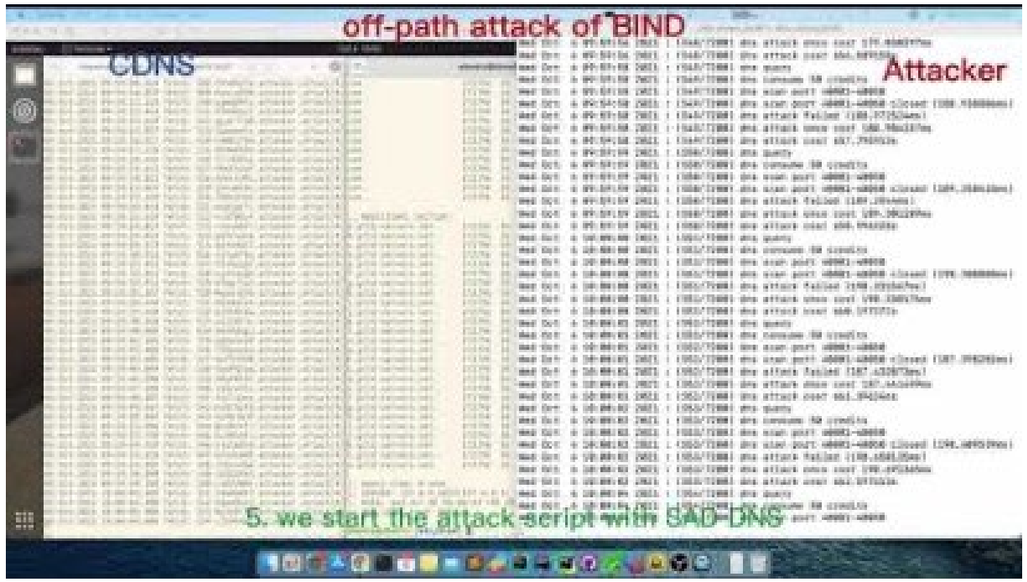



```
Thu Aug 26 23:10:53 2021 : (661/3600) dns querying
Thu Aug 26 23:10:53 2021 : (661/3600) dns consuming 50 credits
Thu Aug 26 23:10:53 2021 : (661/3600) dns scanning port 40001-40050
Thu Aug 26 23:10:54 2021 : (661/3600) dns scanning port 40020 open (651.902104ms)
Thu Aug 26 23:10:54 2021 : (661/3600) dns replying
Thu Aug 26 23:10:54 2021 : (661/3600) dns replying 65535 (928.938966ms)
Thu Aug 26 23:10:54 2021 : (661/3600) dns checking
Thu Aug 26 23:10:54 2021 : (661/3600) dns checking NS gtld-servers.attack.
Thu Aug 26 23:10:54 2021 : (661/3600) dns attack successfully (13m12.992182401s)
Thu Aug 26 23:10:54 2021 : (661/3600) dns attack cost (13m12.99219492s)
```

Log of Attacking BIND9

MaginotDNS Attack Demos

- Off-path Attacks on BIND9 & Microsoft DNS



BIND9

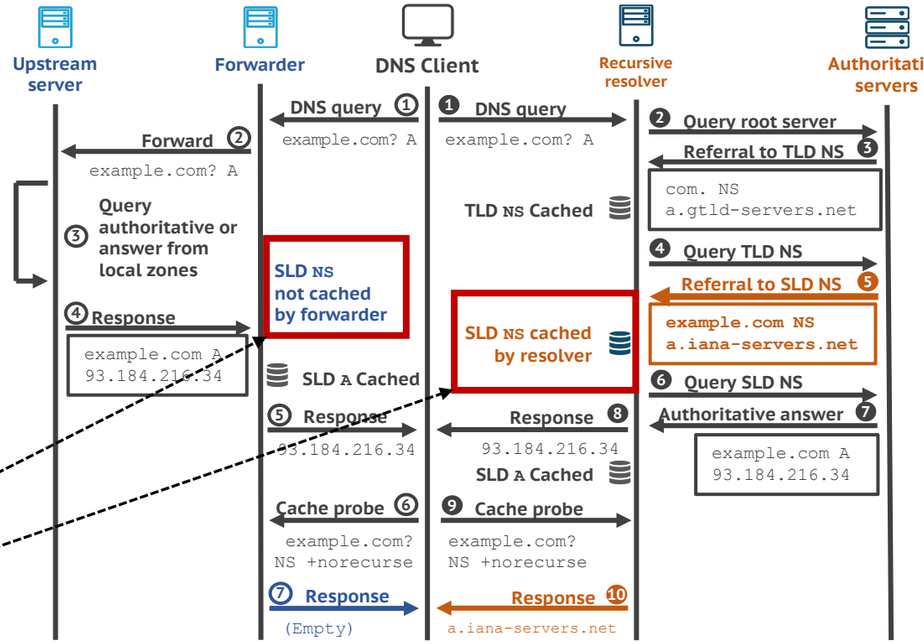
Microsoft DNS

Outline

- DNS overview
- DNS cache poisoning
- MaginotDNS workflow
- Attack demo
- **Large-scale scanning**
- Discussion & conclusion

Finding Vulnerable CDNSes

- **Differentiating Forwarder & Recursive**
  - Based on the DNS resolution mechanism
  - **Forwarders** do not cache **intermediate NS records**
- **Finding CDNSes**
  - New methodology
    1. Targeting one resolver
    2. Testing a group of domains, sending **NS&NR** queries
    3. For some domains, no NS responses (**forwarding**)
    4. For others, we get NS responses (**recursive**)
    5. The resolver does **both forwarding & recursive resolution**
    6. → **CDNS identified**



Vulnerable CDNS Population

➢ Measurement

- ❑ We collected **1.2M resolvers**
- ❑ Removing not-applicable ones, such as violating NR or multiple caches
- ❑ Applying our method to identify **154,955 CDNSes**
- ❑ Using **software fingerprints** to locate **54,949 vulnerable CDNSes**
  - Resolvers with DNSSEC or 0x20 are filtered out

CDNSes identified by probing	**154,955**	**41.8%**
– Version identifiable (in CDNS)	117,306	31.7%
– by `version.bind`	59,419	16.0%
– by `fpdns`	57,887	15.6%
– OS identified for BIND (in CDNS)	19,995	5.4%
– DNSSEC validation (in CDNS)	34,424	9.3%
– 0x20 encoding (in CDNS)	1,119	0.3%

Vulnerable CDNSes	**54,949**	**14.8%**
– On-path attack possible*	**54,949**	**14.8%**
– BIND	24,287	6.6%
– Microsoft DNS	30,662	8.3%
– Off-path attack possible*	**48,539**	**13.1%**
– BIND (OS exploitable)	17,877	4.8%
– Microsoft DNS	30,662	8.3%
– Recursive-default	10,445	5.0%
– Forwarding-default	36,581	9.9%

Outline

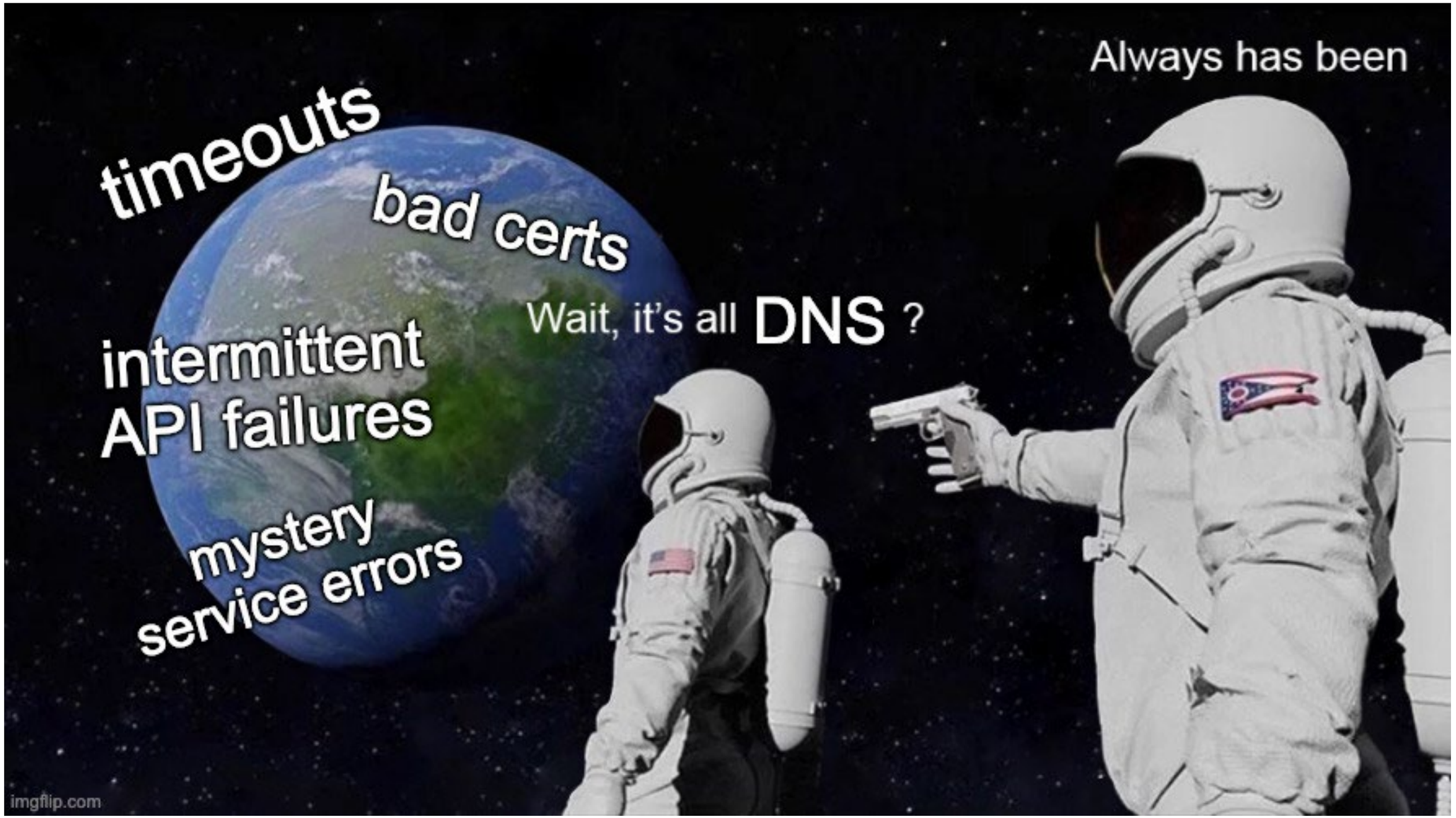
- DNS overview
- DNS cache poisoning
- MaginotDNS workflow
- Attack demo
- Large-scale scanning
- **Discussion & conclusion**

Discussion & Mitigation

- **Vulnerability Disclosure**
  - **Confirmed** and **fixed** by **all affected software**: BIND9, Knot, Microsoft, & Technitium
  - **4 CVE-ids** published & **Bounty** awarded by Microsoft
- **Root Cause**
  - Problematic forwarding bailiwick checking implementations (`Qry.zone` <- root)
    - **Why? Forwarder needs flexibility**
- **Mitigation Solution**
  - `Qry.zone` should be set to the forwarded domain in Z_F (query zone restriction)
  - Then only records under forwarded domain are acceptable (cache split)
  - Have been adopted by affected software

Black Hat Sound Bytes

- **Bailiwick checking is not bullet-proof!**
  - We thought it's perfect after **26 years** since it's born.
- **Inconsistent DNS implementations are common…**
  - Forwarder vs. resolver
  - BIND, Knot, Microsoft, ….
  - Partially caused by the vague RFCs
- **There might be more vulnerabilities we don't even know …**
  - We need **automated tools** (e.g., fuzzers) customized to analyze DNS software
  - My group is working on that ☺



Wrap-up

Thanks for listening!
Any questions?

Paper

Tool

Zhou Li, zhou.li@uci.edu
Xiang Li, x-l19@mails.tsinghua.edu.cn
Qifan Zhang, qifan.zhang@uci.edu