

[2024.10.15] Main Seminar

A Flushing Attack on the DNS Cache

USENIX Security 24' Fall

Yehuda Afek*, Anat Bremler-Barr*, Shani Stajnsrod

Tel-Aviv University, Reichman University*

Jungbum Lee

jblee@mmlab.ac.kr

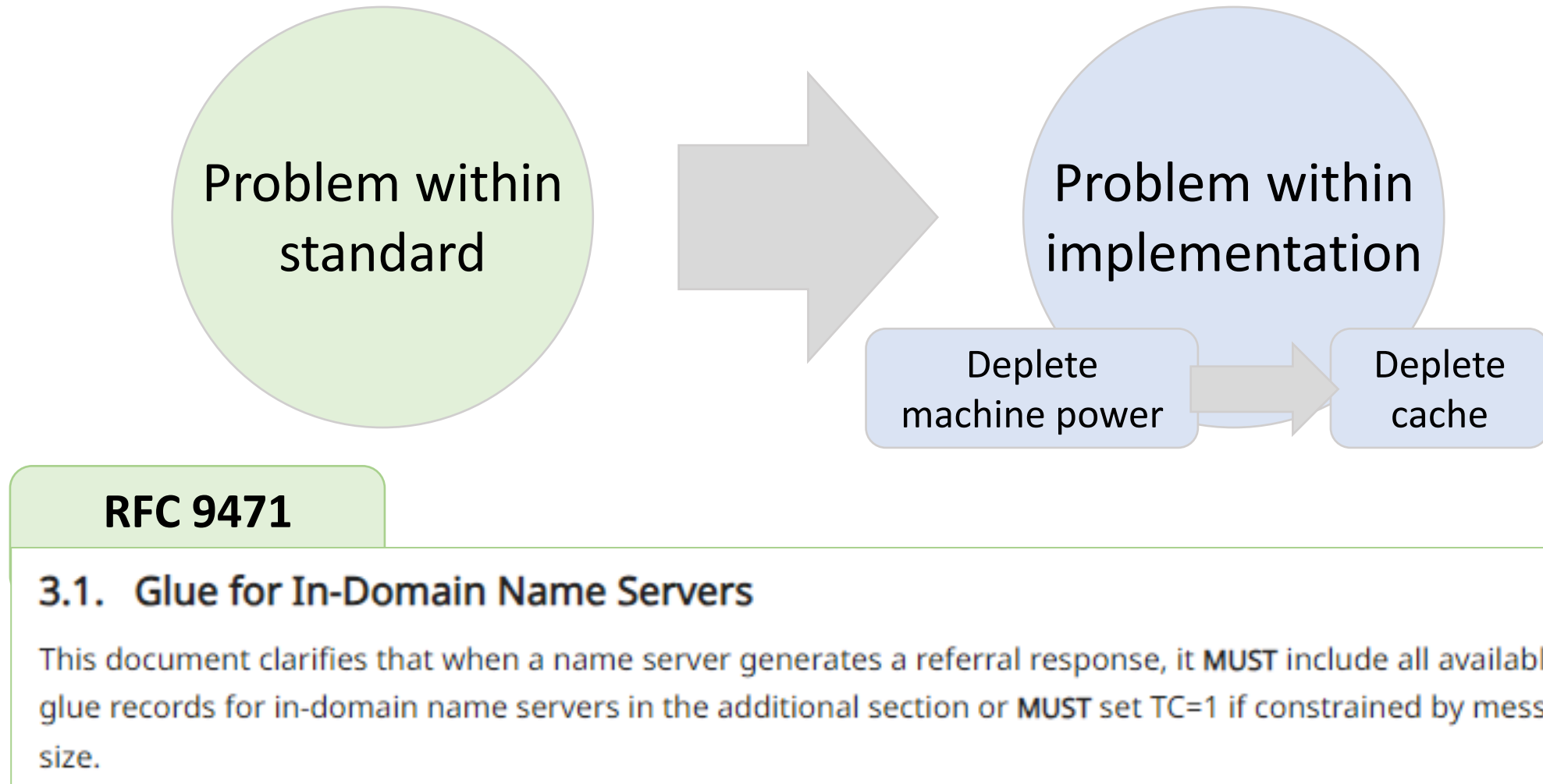


mmlab
Network Convergence & Security Lab

Contents

- Current trends in DNS amplification attack
- Background *DNS resolving in depth*
- Cache flushing attack and limitations
- Conclusion


Current Trends in DNS Amplification Attacks



Current Trends in DNS Amplification Attacks (cont.)

x10+
iDNS
OARC 15'

x500+
tsuNAME
IMC 20'

 **DNS and TTL**
ArXiv

x1600

NXNS

Security 20'

x5600*

NRDelegation

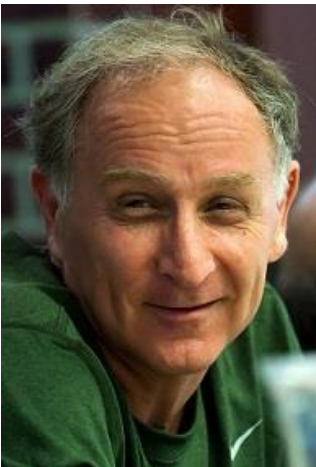
Security 23'

12MB/s*

FlushingAttack

Security 24'

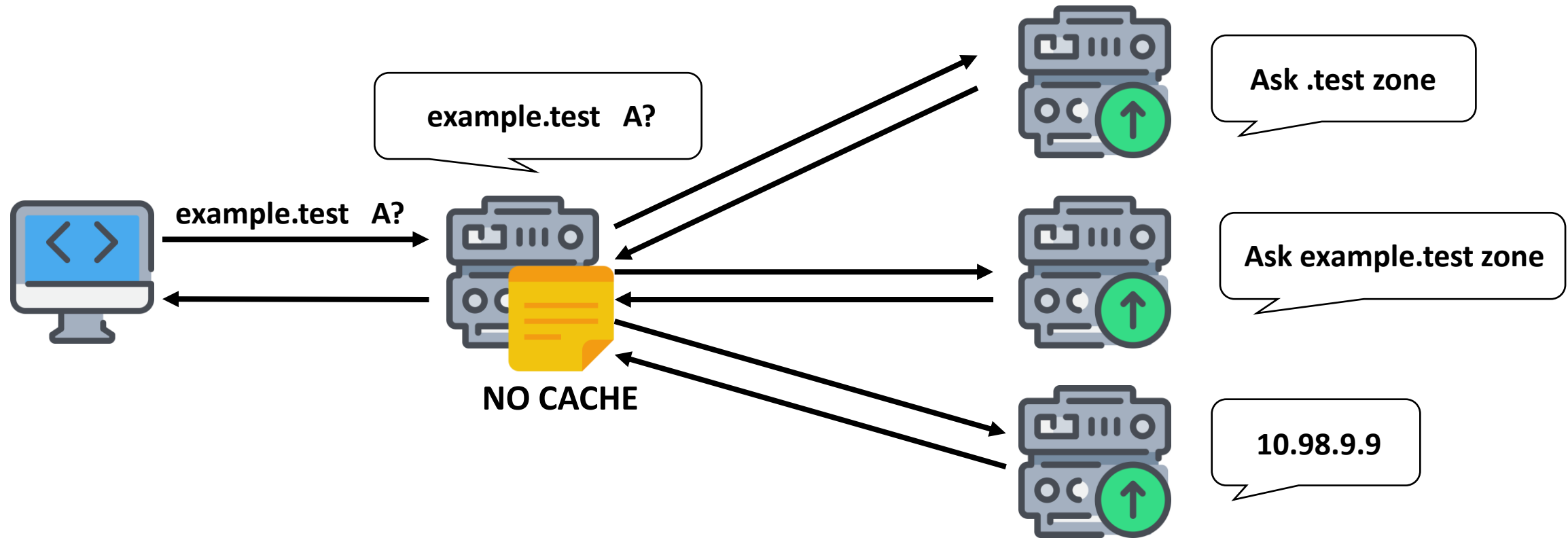
Problem with BIND, UNBOUND, etc.



Afek, Y

DNS Basic

- DNS is a distributed database that stores some values (such as IP address) that map domain names to the values



Delegation and Referral in DNS

- To answer the resolver's query, an **authoritative nameserver** can choose whether to answer the question directly **or delegate the answer to another nameserver**
- The delegation is mostly driven by performance gains and enables integration with third-party services
- **Referral response** is a multiple delegation response
- Motivated by fault-tolerance and managing latency
- **Delegation has been a vulnerable attack vector in DNS**

Glue records (after RFC 9471)

- DNS uses **glue records** to allow iterative clients to find the addresses of name servers that are contained within a delegated zone
- Glue records are added to the parent zone as part of the delegation process and returned in referral response
- Name server **MUST** include all available glue records for **in-domain** name servers

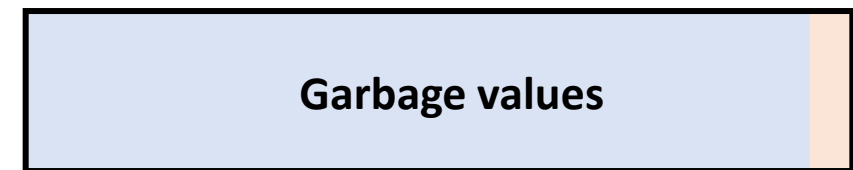
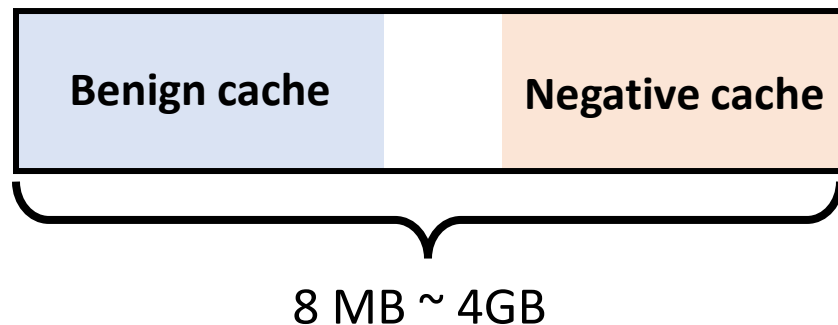
```
;; QUESTION SECTION:
;www.foo.test.      IN      A

;; AUTHORITY SECTION:
foo.test.          86400    IN      NS      ns1.foo.test.
foo.test.          86400    IN      NS      ns2.foo.test.

;; ADDITIONAL SECTION:
ns1.foo.test.      86400    IN      A        192.0.2.1
ns2.foo.test.      86400    IN      AAAA       2001:db8::2:2
```

DNS Resolver Cache

- DNS resolver cache many answers including DNS RR types, domain names, IP addresses, etc.
- There are two type of caches, **benign cache** and **negative cache**
 - **Benign cache** saves successful resolution
 - **Negative cache** saves failed resolution, such as NXDOMAIN, NODATA, time-out, etc.
- Cache is **dynamically distributed**, means the storage where benign cache uses, and negative cache uses are integrated
 - Benign cache has higher priority

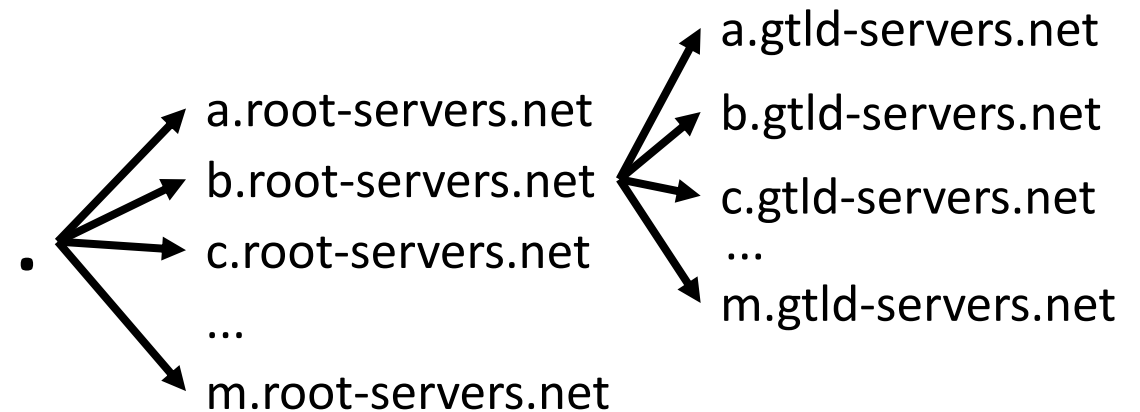


SLIST

- SLIST is a scratch pad memory for name servers and zone which the resolver is **currently trying to query**
- SLIST keeps track of the resolver's best guess about **what to query next**

Query? `google.com.` **A**

<code>google.com.</code>	A	<code>142.251.42.142</code>
<code>.</code>	NS	<code>e.root-servers.net.</code>
<code>com.</code>	NS	<code>c.gtld-servers.net.</code>
<code>google.com.</code>	NS	<code>ns3.google.com.</code>
<code>e.root-servers.net.</code>	A	<code>x.x.x.x</code>
<code>c.gtld-server.net.</code>	A	<code>x.x.x.x</code>
<code>ns3.google.com.</code>	A	<code>x.x.x.x</code>

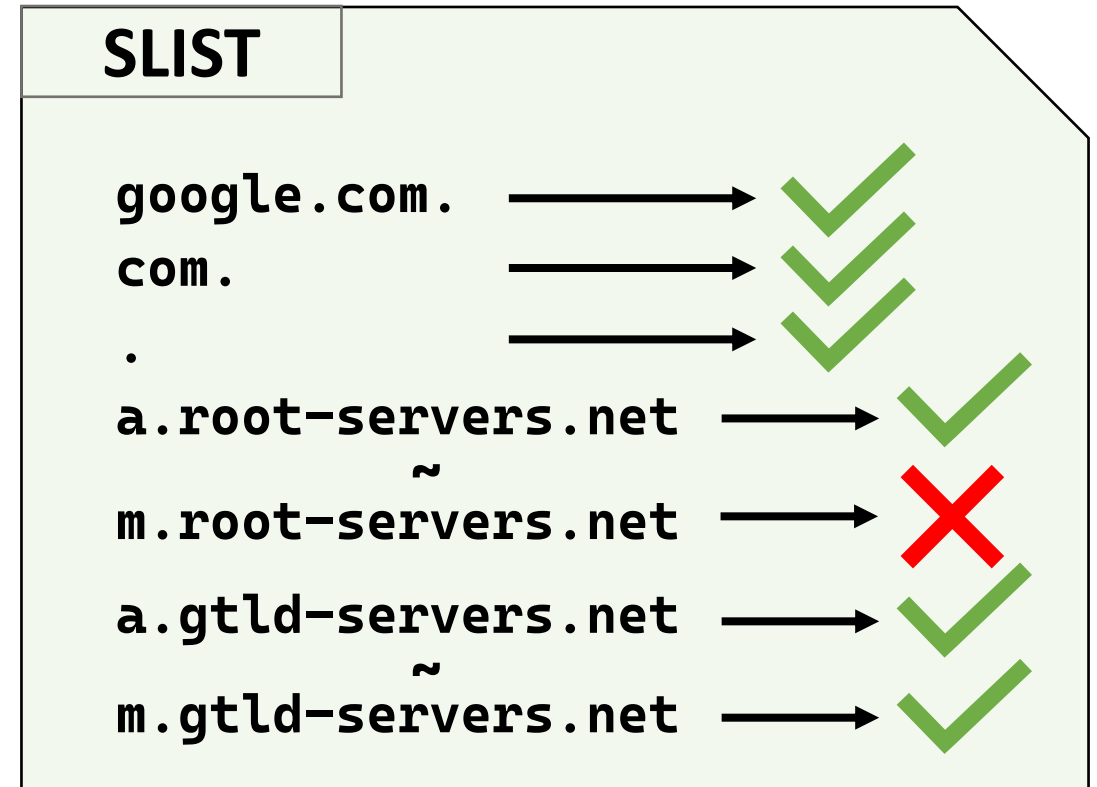


SLIST (cont.)

- SLIST is a scratch pad memory for name servers and zone which the resolver is **currently trying to query**
- SLIST keeps track of the resolver's best guess about **what to query next**

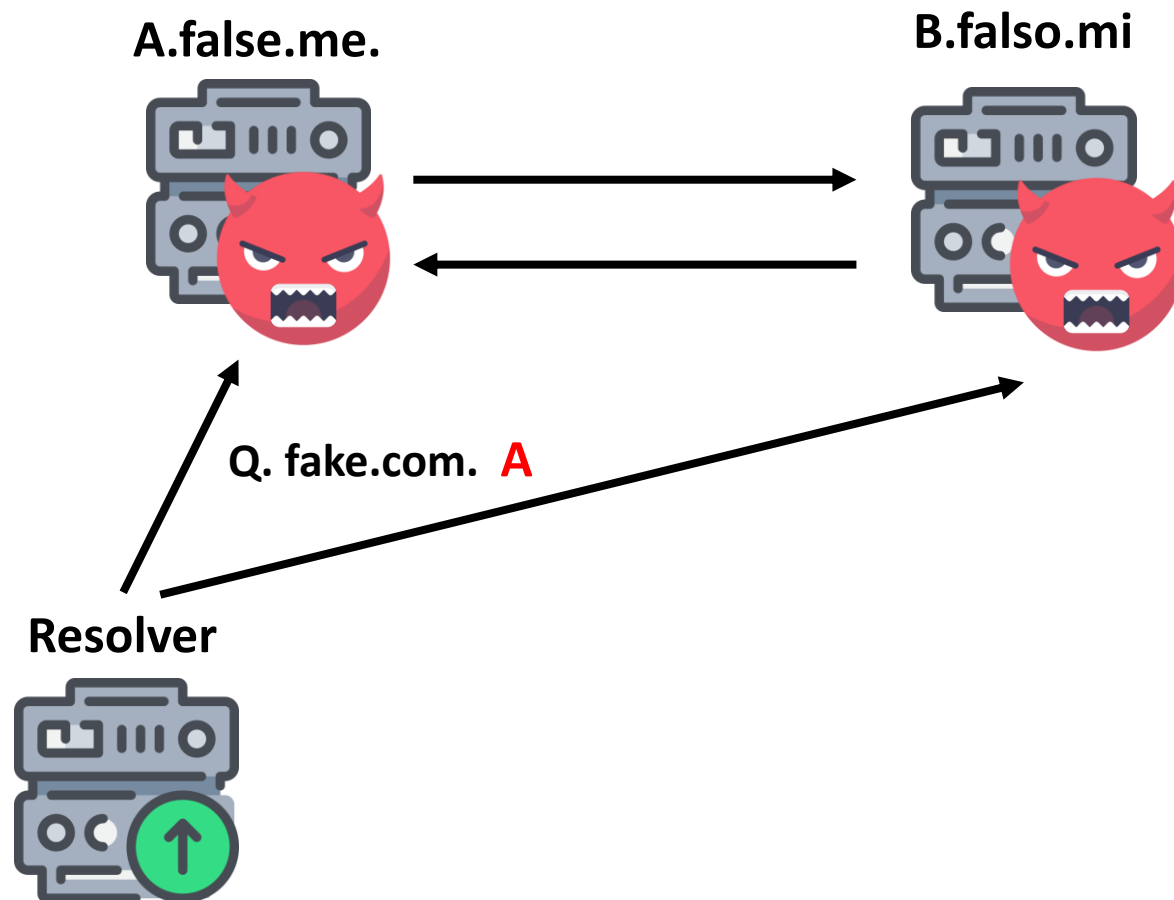
Query? google.com. A

google.com.	A	142.251.42.142
.	NS	e.root-servers.net.
com.	NS	c.gtld-servers.net.
google.com.	NS	ns3.google.com.
e.root-servers.net.	A	x.x.x.x
c.gtld-server.net.	A	x.x.x.x
ns3.google.com.	A	x.x.x.x



Why we need SLIST?

- To prevent redundancy and infinite looping

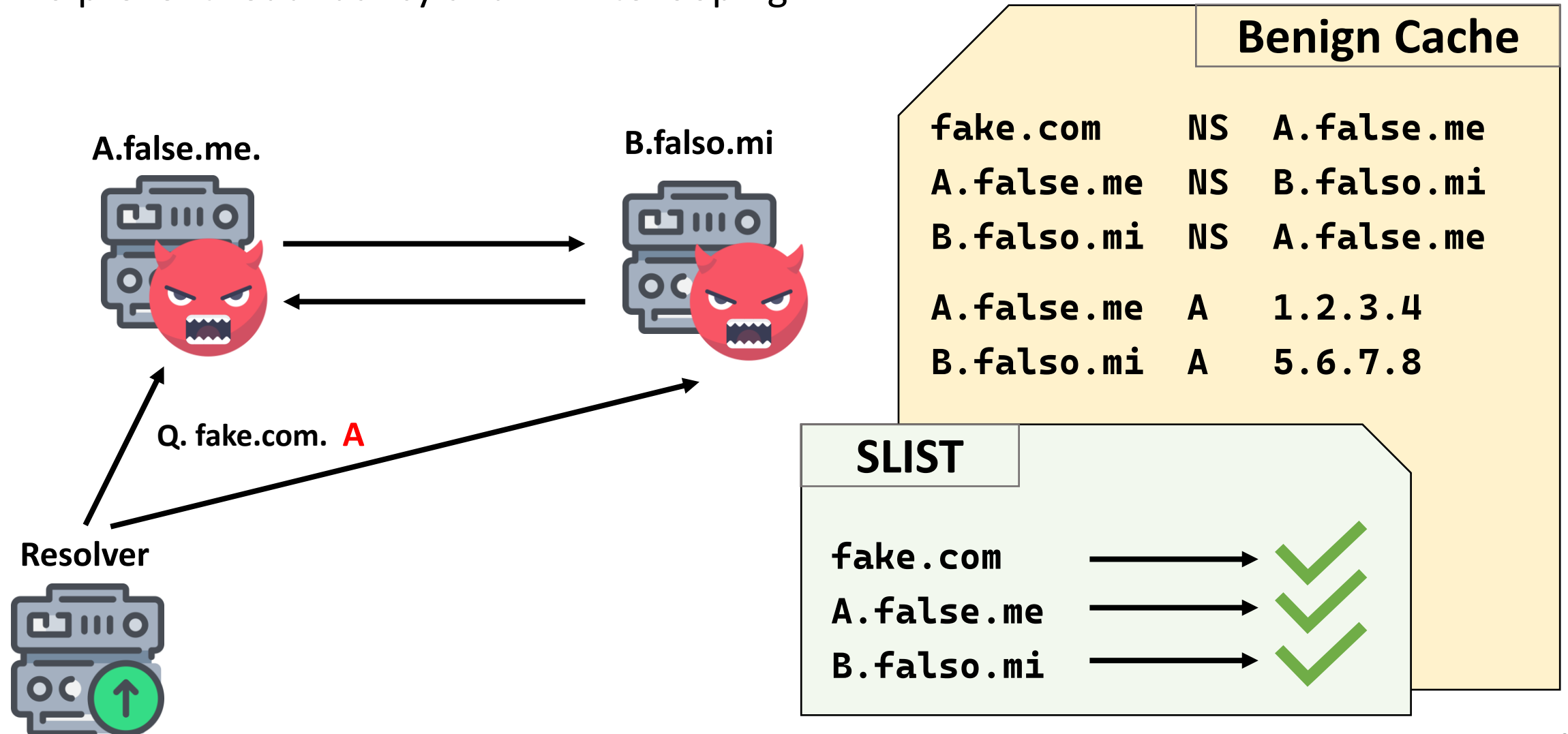


Benign Cache

fake.com	NS	A.false.me
A.false.me	NS	B.falso.mi
B.falso.mi	NS	A.false.me
A.false.me	A	1.2.3.4
B.falso.mi	A	5.6.7.8

Why we need SLIST? (cont.)

- To prevent redundancy and infinite looping



SLIST and Cache

- The resolver fetches all name server information from the target server and checks the connectivity to each server in parallel, storing the results in the cache based on whether the servers are reachable
- Therefore, the resolver with SLIST **relies heavily on cache**
- Without cache, resolver should query all nameservers they met from root for each query

```
;; AUTHORITY SECTION:  
com.      NS      a.gtld-servers.net.  
com.      NS      b.gtld-servers.net.  
com.      NS      c.gtld-servers.net.  
  
;; ADDITIONAL SECTION:  
a.gtld-servers.net.  A      192.5.6.30  
b.gtld-servers.net.  A      192.33.14.30  
c.gtld-servers.net.  A      192.26.92.30
```

This is NOT resolved by glue record



Resolver traverse all this IPs in parallel
and check its reachability


Two Issues with Cache Management in Resolver

- There are **no limits** of the NS record number in a single domain to benign cache
- When the hard limit is reached, the resolver usually drops all requests

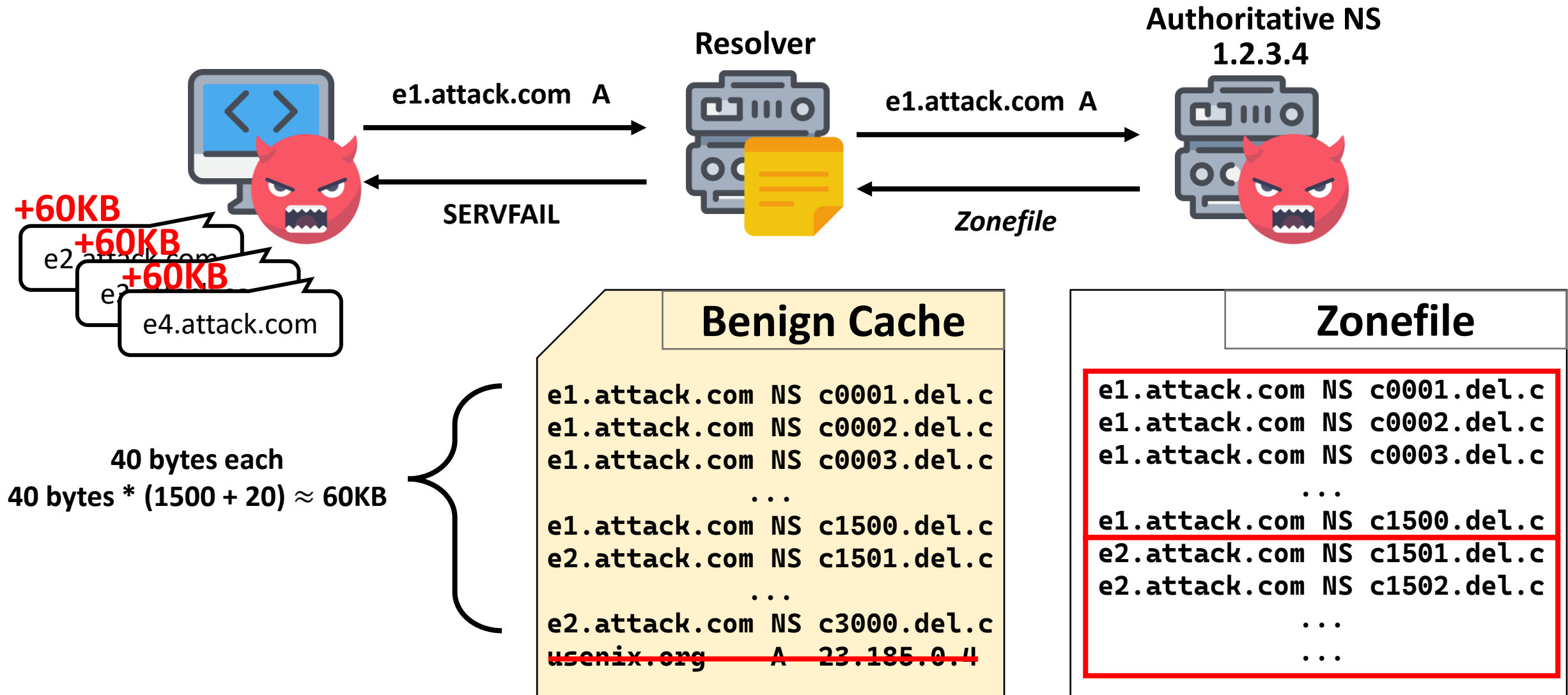
google.com	NS	ns2.google.com
google.com	NS	ns1.google.com
google.com	NS	ns4.google.com
google.com	NS	ns3.google.com

e1.attack.com	NS	c0001.del.c
e1.attack.com	NS	c0002.del.c
e1.attack.com	NS	c0003.del.c
	...	
e1.attack.com	NS	c1499.del.c
e1.attack.com	NS	c1500.del.c

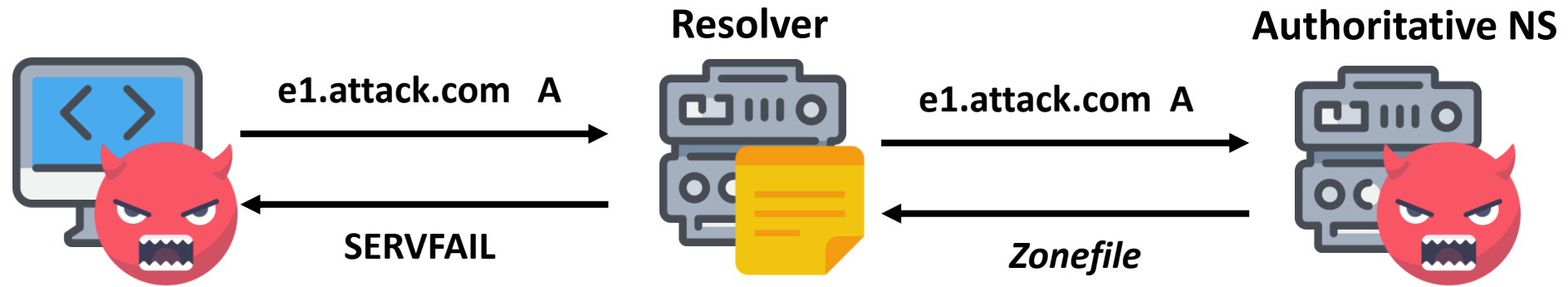
A google.com

	NS c1483.del.c
	NS c1482.del.c
	NS c1481.del.c
	NS c1480.del.c

NSCacheFlush Attack



CNAMECacheFlush Attack

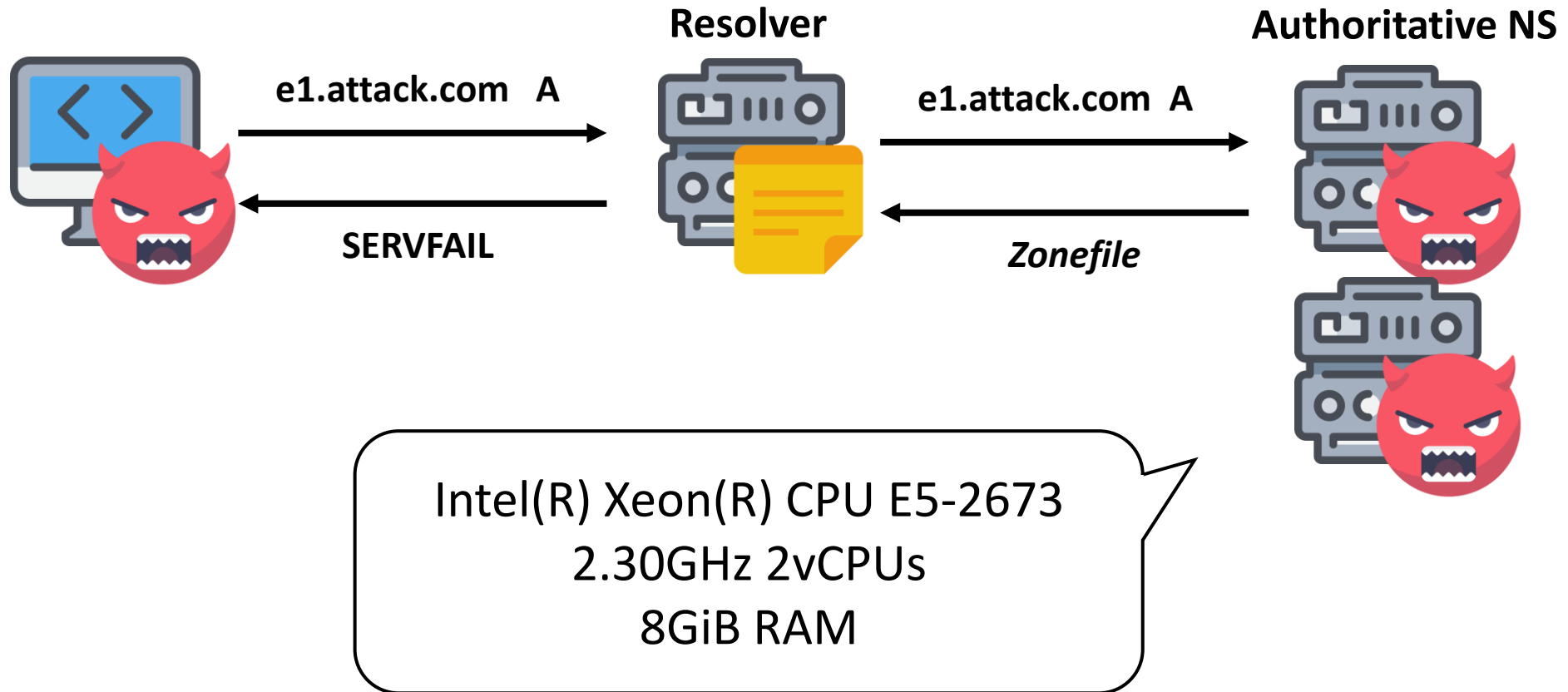


Benign Cache		
<code>e1.attack.com</code>	<code>CNAME</code>	<code>e2.attack.com</code>
<code>e2.attack.com</code>	<code>CNAME</code>	<code>e3.attack.com</code>
<code>e3.attack.com</code>	<code>CNAME</code>	<code>e4.attack.com</code>
<code>e4.attack.com</code>	<code>CNAME</code>	<code>e5.attack.com</code>
...		
<code>e98.attack.com</code>	<code>CNAME</code>	<code>e99.attack.com</code>
...		
<code>usenix.org</code>	<code>A</code>	<code>23.185.0.4</code>

Zonefile		
<code>e1.attack.com</code>	<code>CNAME</code>	<code>e2.attack.com</code>
<code>e2.attack.com</code>	<code>CNAME</code>	<code>e3.attack.com</code>
<code>e3.attack.com</code>	<code>CNAME</code>	<code>e4.attack.com</code>
<code>e5.attack.com</code>	<code>CNAME</code>	<code>e6.attack.com</code>
...		
<code>e98.attack.com</code>	<code>CNAME</code>	<code>e99.attack.com</code>
...		

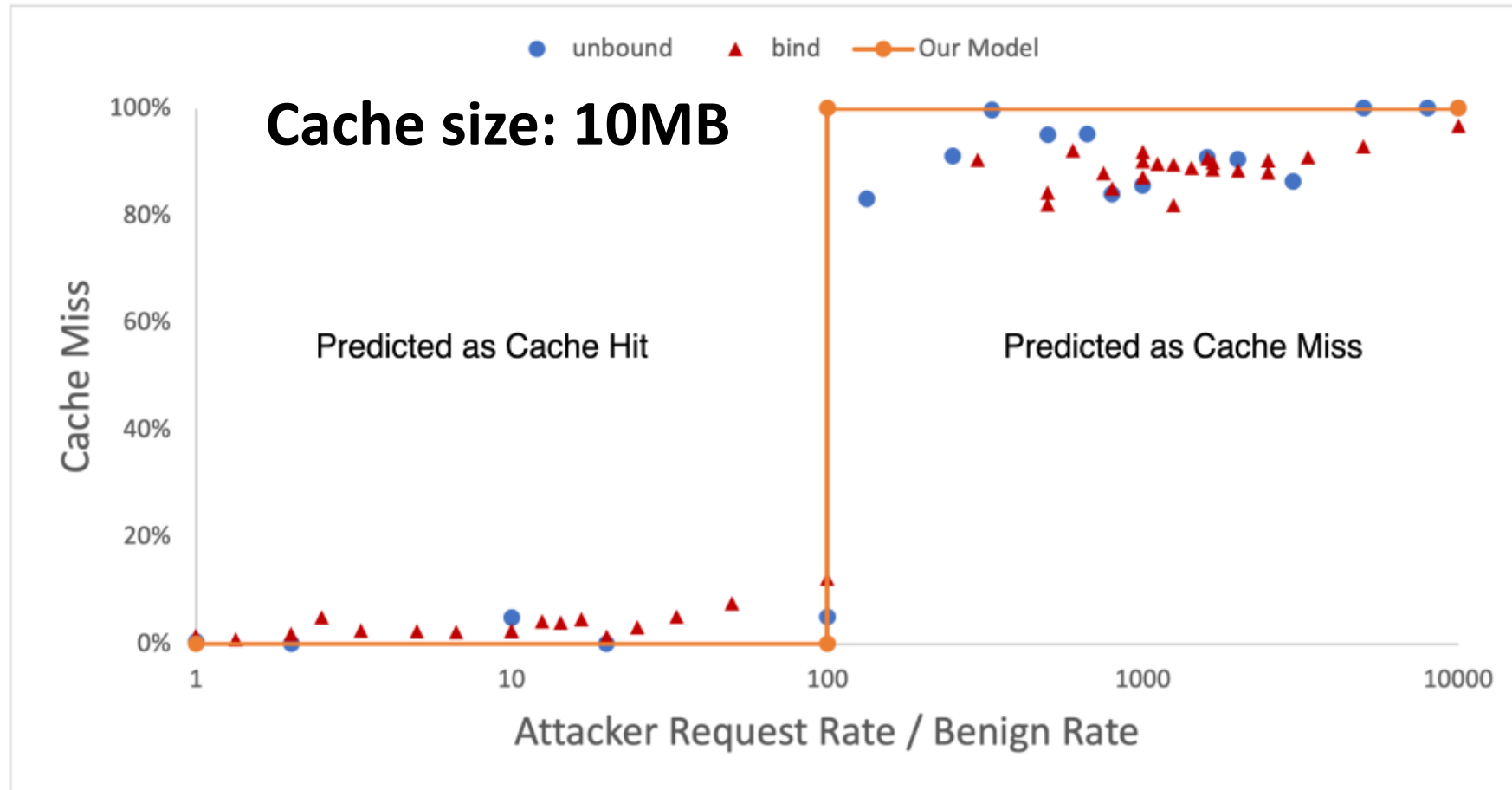
Computational Load of Authoritative Name Server

- Computational load is relatively **low**



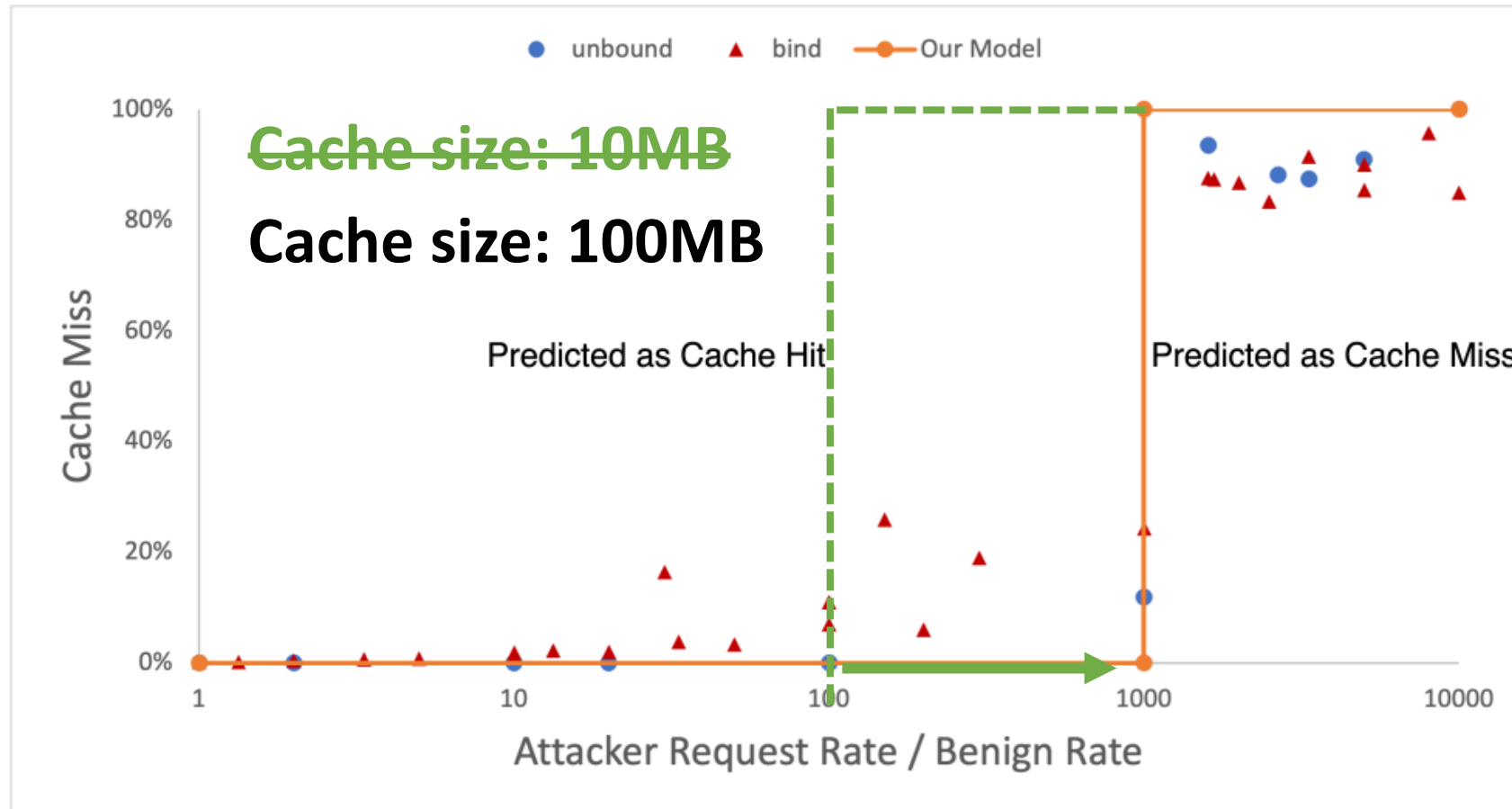
Result

- After a certain request rate builds up, the **cache miss ratio suddenly increases**




Result (cont.)

- The request rate required for the attack to succeed **increases proportionally to the cache size**




CacheFlush Mitigation – Create a hard limit!

- Bounding NS referral list
- Bounding the length of CNAME chains

Benign Cache			
e1.attack.com	NS	c0001.del.c	
e1.attack.com	NS	c0002.del.c	
e1.attack.com	NS	c0003.del.c	
			
e1.attack.com	NS	c1500.del.c	
c0001.del.c	A	1.2.3.4	
...			
c0020.del.c	A	1.2.3.4	
usenix.org	A	23.185.0.4	

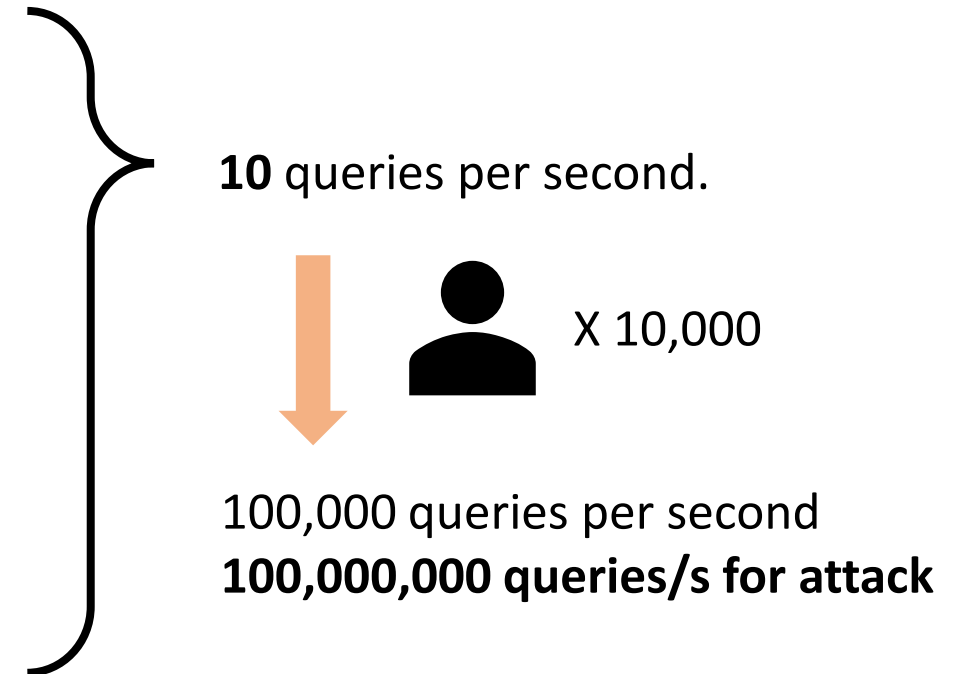
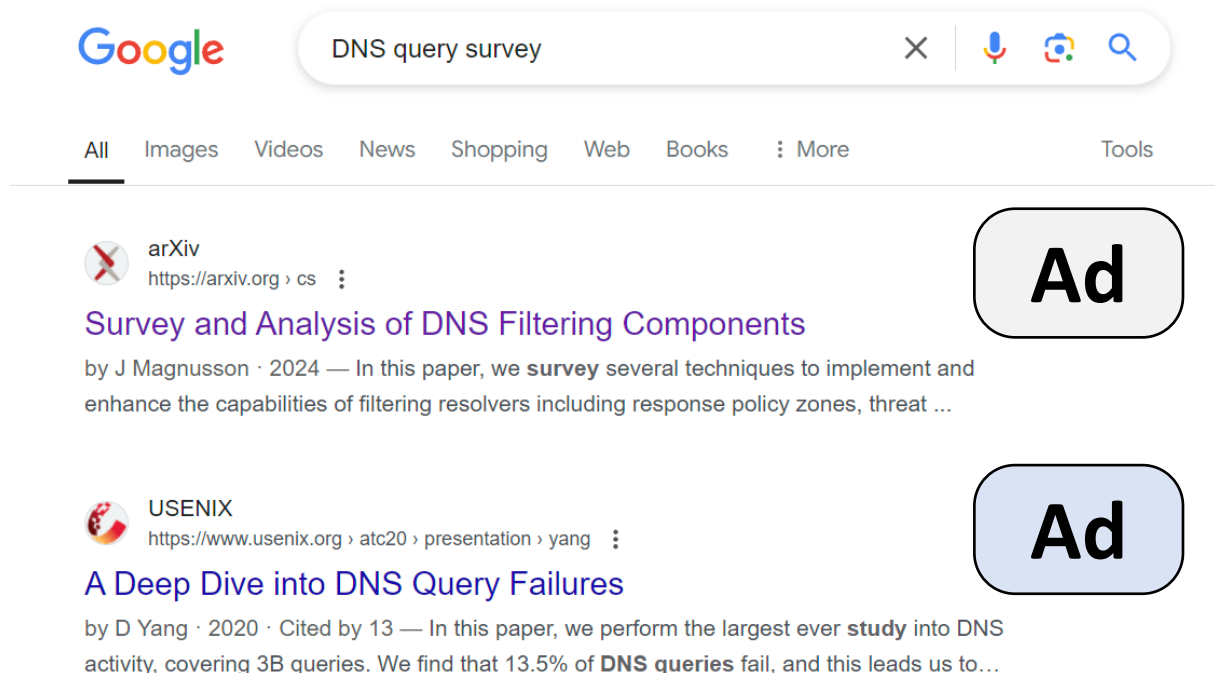
NSCacheFlush Attack

Benign Cache			
e1.attack.com	CNAME	e2.attack.com	
e2.attack.com	CNAME	e3.attack.com	
e3.attack.com	CNAME	e4.attack.com	
e4.attack.com	CNAME	e5.attack.com	
			
e98.attack.com	CNAME	e99.attack.com	
...			
usenix.org	A	23.185.0.4	

CNAMECacheFlush Attack

Limitation

- Attack rate should highly exceed original benign request
 - If cache size is 1GB, attacker's request rate should be 10,000 times higher than benign request
- Public DNS is highly distributed, it's hard to attack specific query



Conclusion

- BIND and UNBOUND were vulnerable to cache flushing attack
- Since DNS has had no restrictions on CNAMEs and NSs, there is no obvious choice but to impose hard limits
 - The pattern of using CNAME and NS is highly scattered
- For small name servers, a small amount of computing power can cause most queries to cache miss
 - University, company's centralized resolver
- For large name servers, such as public resolver, it's hard to succeed the attack

Thank you

CNAME

- DNS uses CNAME record to **offer canonical name** associated with an alias name
- CNAME **CAN** point to another CNAME record
- CNAME **CAN** make unresolvable loops, while it is not recommended
- MX and NS records must **NEVER** point to CNAME RR

NAME	TYPE	VALUE

bar.example.com.	CNAME	foo.example.com.
foo.example.com.	A	192.0.2.23



example.com.	MX	0	foo.example.com.
foo.example.com.	CNAME		host.example.com.
host.example.com.	A		192.0.2.1

NSCacheFlush Attack

