

[2024.07.09] Main Seminar

NRDelegationAttack: Complexity DDoS attack on DNS Recursive Resolvers

USENIX Security 23' Fall

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

Tel-Aviv University, Reichman University*

Jungbum Lee

jblee@mmlab.ac.kr

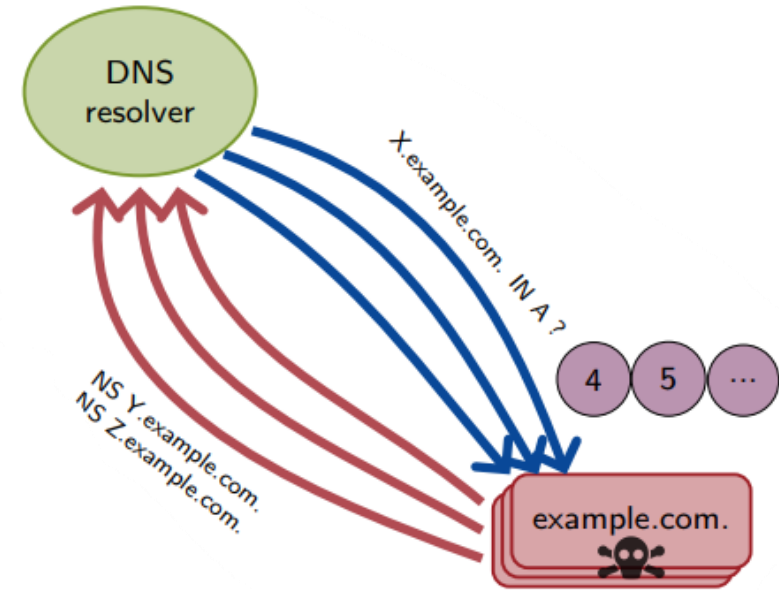
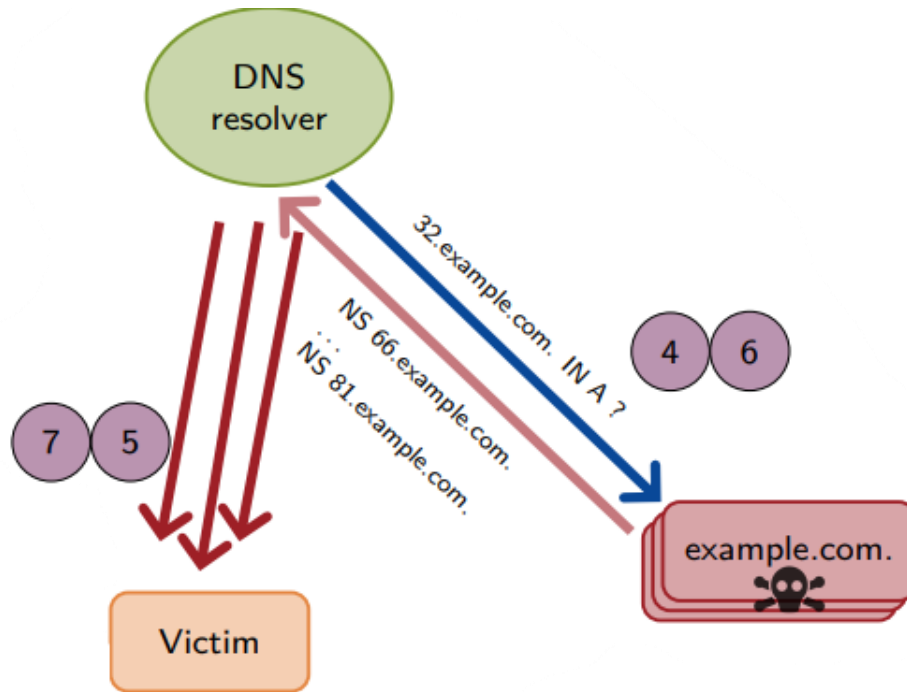
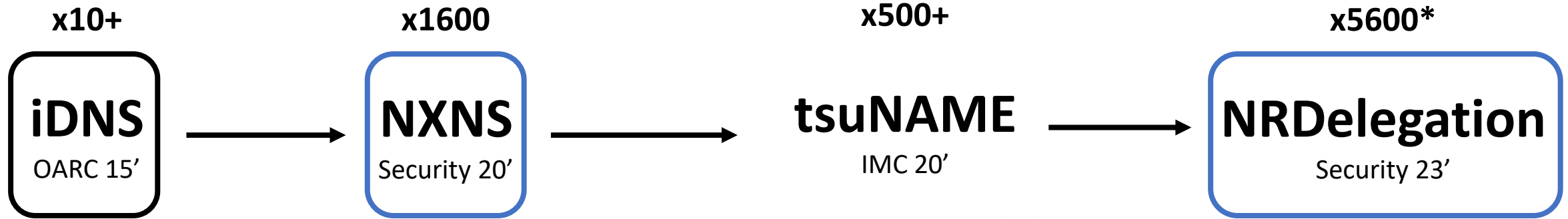


mmlab
Network Convergence & Security Lab

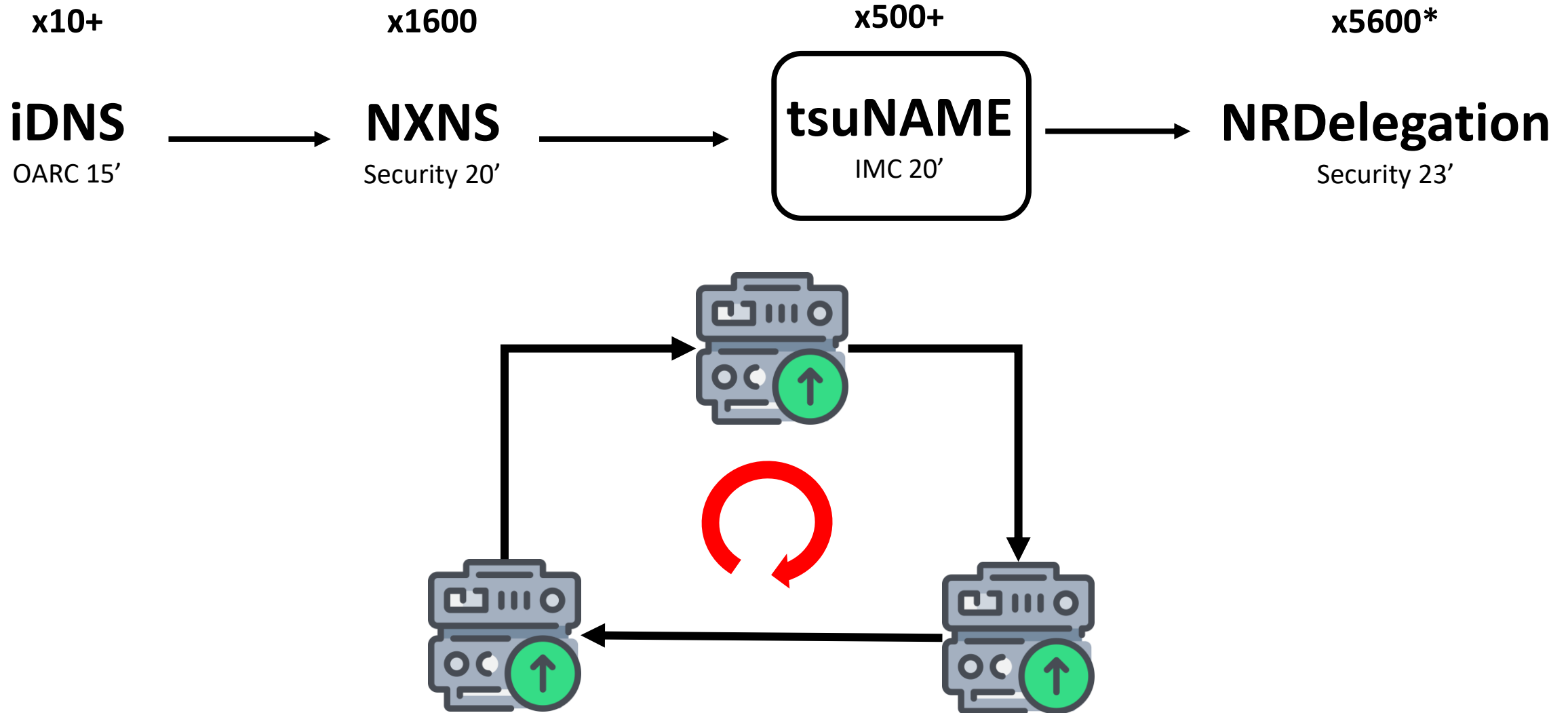
Delegation and referral in DNS

- 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**

Amplification attack on DNS



Amplification attack on DNS (cont.)

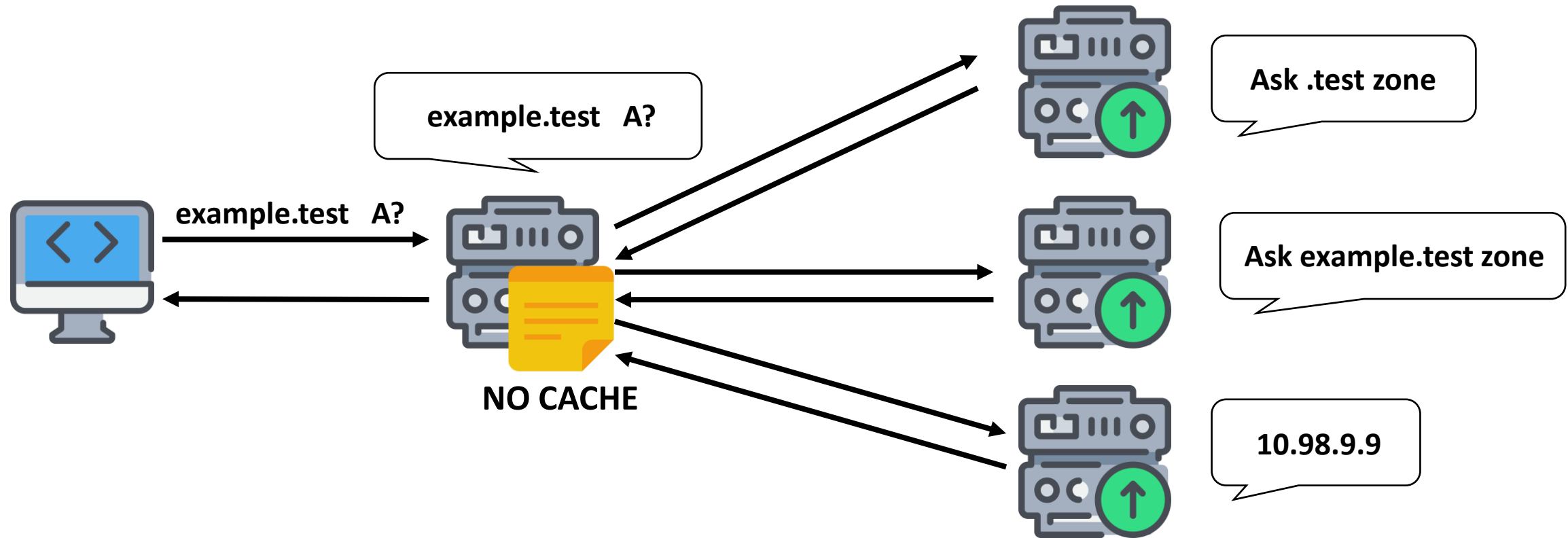


Contents

- Background *Dig into DNS query*
- Prior delegation response attack - NXNSAttack (Usenix Security 20')
- Latest delegation response attack - NRDelegation Attack
- Solutions
- Evaluation

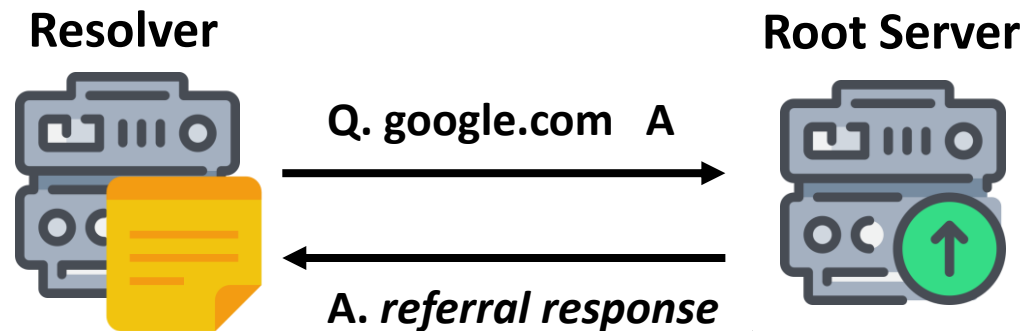
Background

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



Dig into query

- DNS server sends **authority sections list** to the resolver
- Resolver selects a server based on its own internal policy
 - Usually choose the server with the **fastest response time**
- Servers sometimes **glue** IP addresses together for efficiency (optional)



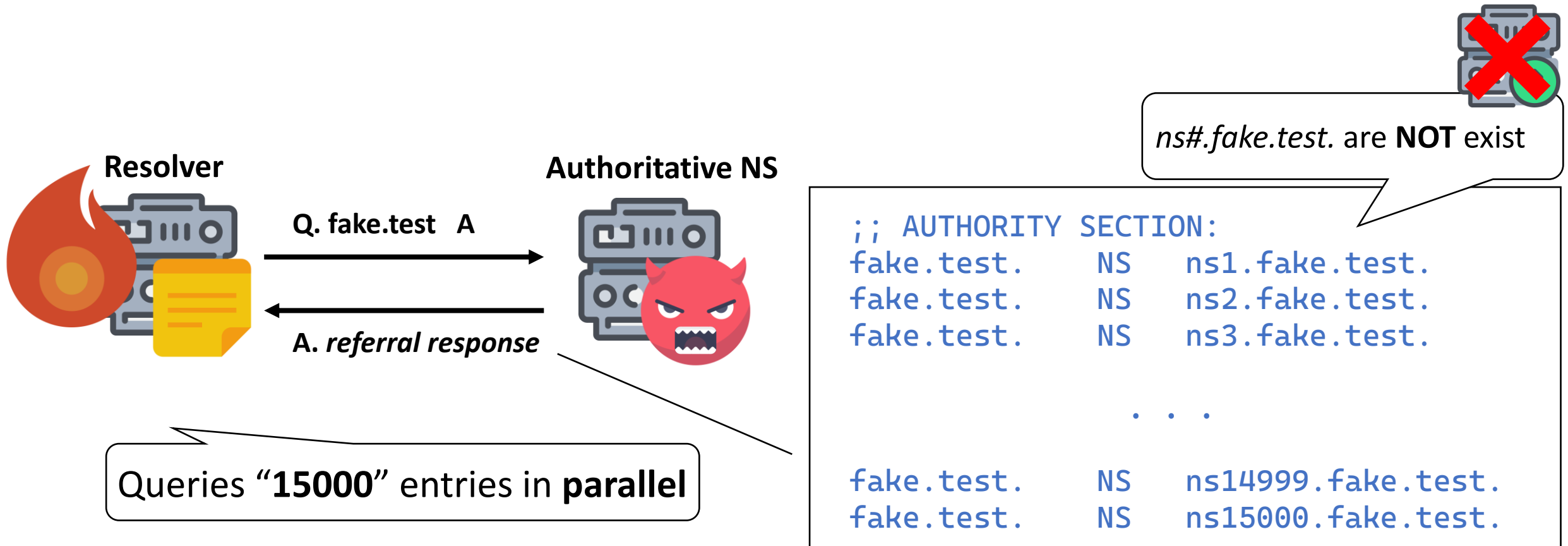
Query each entry in the list **parallel**

```
;; 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
```

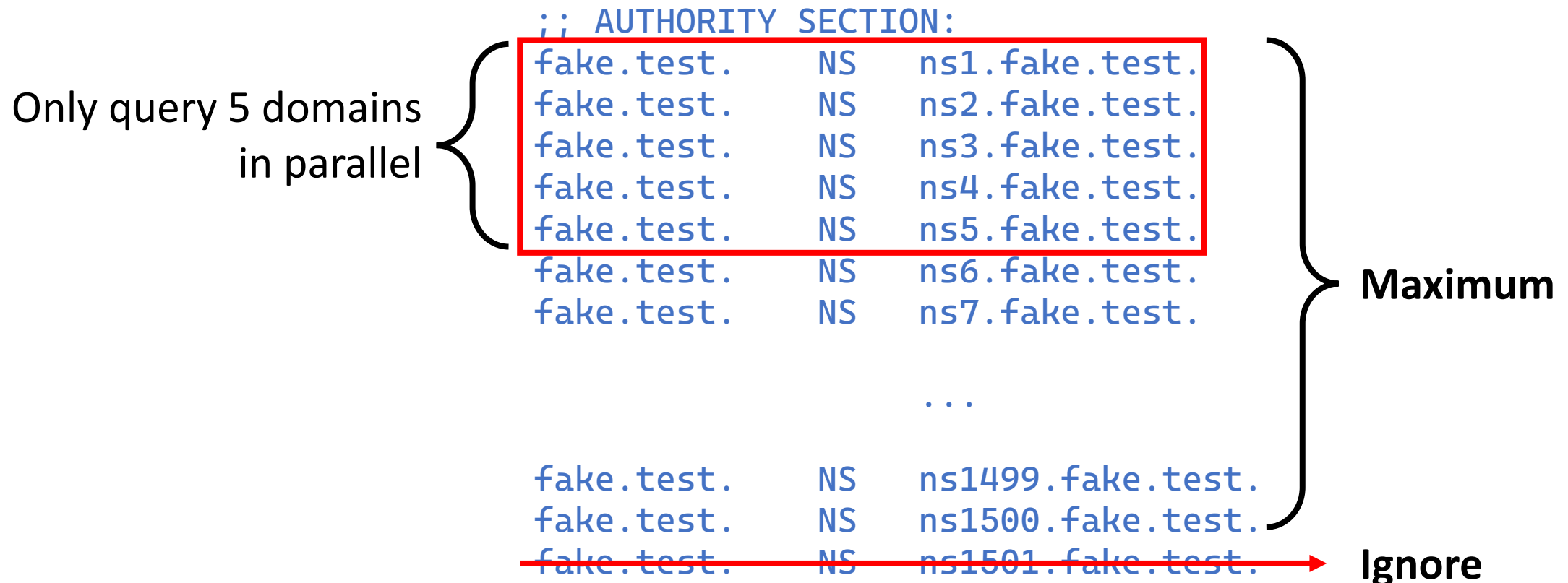
NXNS Attack (Security 20')

- Recent DNS attack, NXNS attack employs a malicious NS referral response using a **long list of non-existing name server names**
- Until 20', resolvers query domain names in referral response **ALL in parallel**



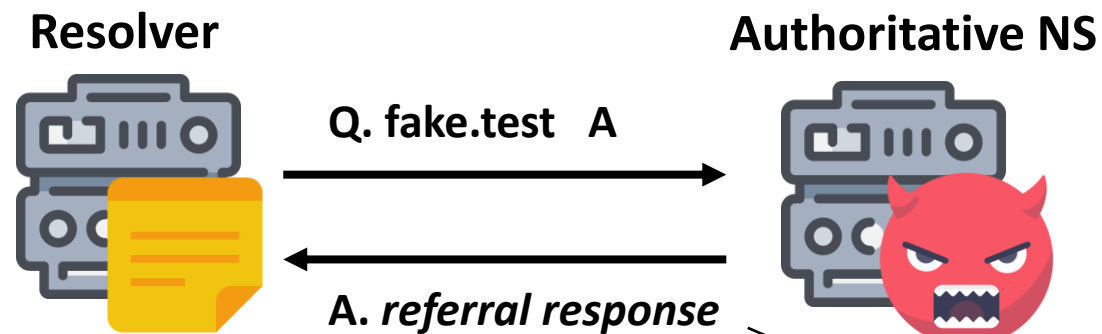
NXNS Attack (Security 20') - Solution

- To mitigate the attack, some **limits** are introduced




NXNS Attack (Security 20') - Solution

- Process only five queries in parallel at a time
- If all five queries fail, the next five are fetched and queried
- This method can still occupy the resolver's resources for **a long time**, but it **doesn't consume more than a limited amount of resources**.



Works "5" entries in **parallel**

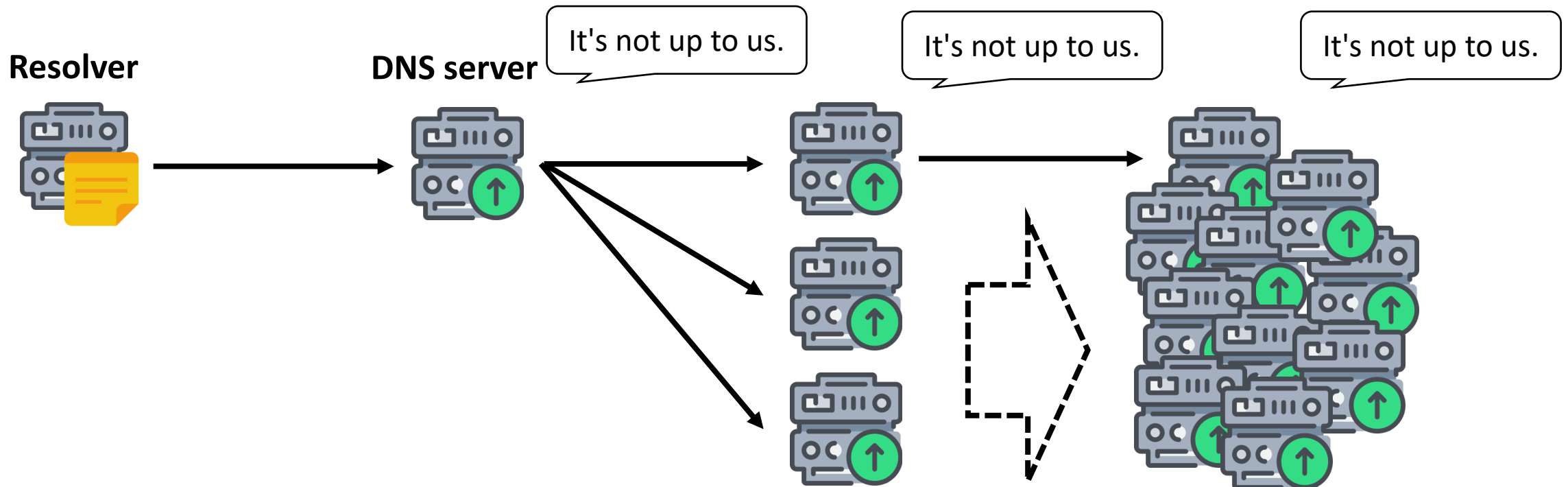

ns#.fake.test. are **NOT** exist

;; AUTHORITY SECTION:

fake.test.	NS	ns1.fake.test.
fake.test.	NS	ns2.fake.test.
fake.test.	NS	ns3.fake.test.
fake.test.	NS	ns4.fake.test.
fake.test.	NS	ns5.fake.test.
fake.test.	NS	ns6.fake.test.
fake.test.	NS	ns7.fake.test.
...		

What if the malicious nameserver responds?

- The previous attack assumed that the malicious nameserver **doesn't respond at all**
- What happens if the server sends a **referral response** to another malicious server?
- And what happens if the server sends **multiple referral responses** recursively?



If not fail but success with delegation response?

- The resolver handles each referral response processing **independently**
- This means that the previous limits worked **independently**
 - Query count increased in **logarithmic scale**

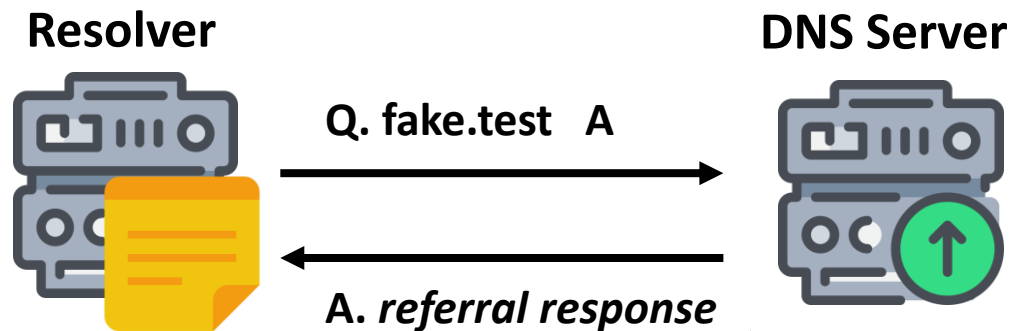
;; AUTHORITY SECTION:

fake.test.	NS	ns1.fake.test.
fake.test.	NS	ns2.fake.test.
fake.test.	NS	ns3.fake.test.
fake.test.	NS	ns4.fake.test.
fake.test.	NS	ns5.fake.test.
fake.test.	NS	ns6.fake.test.
fake.test.	NS	ns7.fake.test.
...		
fake.test.	NS	ns999.fake.test.
fake.test.	NS	ns1000.fake.test.

{	fake.test.	NS	ns6.fake.test.
	fake.test.	NS	ns7.fake.test.
	fake.test.	NS	ns8.fake.test.
	fake.test.	NS	ns9.fake.test.
	fake.test.	NS	ns10.fake.test.
{	fake.test.	NS	ns11.fake.test.
	fake.test.	NS	ns12.fake.test.
	fake.test.	NS	ns13.fake.test.
	fake.test.	NS	ns14.fake.test.
	fake.test.	NS	ns15.fake.test.
{	fake.test.	NS	ns16.fake.test.
	fake.test.	NS	ns17.fake.test.
	fake.test.	NS	ns18.fake.test.
	fake.test.	NS	ns19.fake.test.
	fake.test.	NS	ns20.fake.test.

Is the attack feasible?

- We don't actually need N servers for all N servers to get a response
- Servers sometimes **glue** IP addresses together for efficiency (optional)
- If an IP address in *ADDITIONAL SECTION* is the same, **the resolver uses the data in its own cache rather than querying other nameservers**



```
;; AUTHORITY SECTION:  
fake.test.      NS      ns1.fake.test.  
fake.test.      NS      ns2.fake.test.  
fake.test.      NS      ns3.fake.test.
```

```
;; ADDITIONAL SECTION:
```

```
ns1.fake.test.  A      192.9.9.9  
ns2.fake.test.  A      192.9.9.9  
ns3.fake.test.  A      192.9.9.9
```

Is the attack feasible?

- Omit the corresponding IP addresses (IP glue) of the NS record
- The resolver then **should traverse all the NS by itself, without help from the cache**

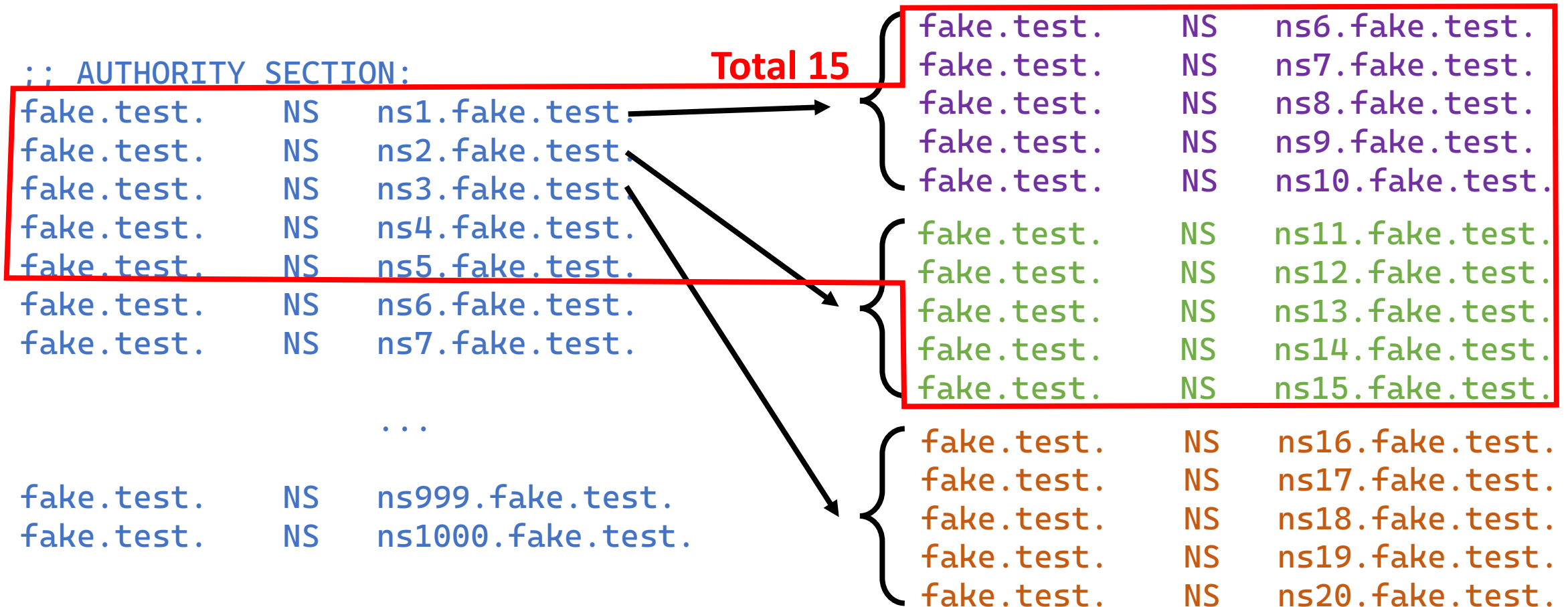
;; AUTHORITY SECTION:

```
fake.test.      NS      ns1.fake.test.  
fake.test.      NS      ns2.fake.test.  
fake.test.      NS      ns3.fake.test.  
fake.test.      NS      ns4.fake.test.  
fake.test.      NS      ns5.fake.test.  
fake.test.      NS      ns6.fake.test.  
fake.test.      NS      ns7.fake.test.  
  
...  
  
fake.test.      NS      ns999.fake.test.  
fake.test.      NS      ns1000.fake.test.
```

```
ns1.fake.test.  
      NS      ns1.fake.test., ns2. ...  
ns2.fake.test.  
      NS      ns1.fake.test., ns2. ...  
ns3.fake.test.  
      NS      ns1.fake.test., ns2. ...  
ns4.fake.test.  
      NS      ns1.fake.test., ns2. ...  
ns5.fake.test.  
      NS      ns1.fake.test., ns2. ...  
ns6.fake.test.  
      NS      ns1.fake.test., ns2. ...  
ns7.fake.test.  
      NS      ns1.fake.test., ns2. ...
```

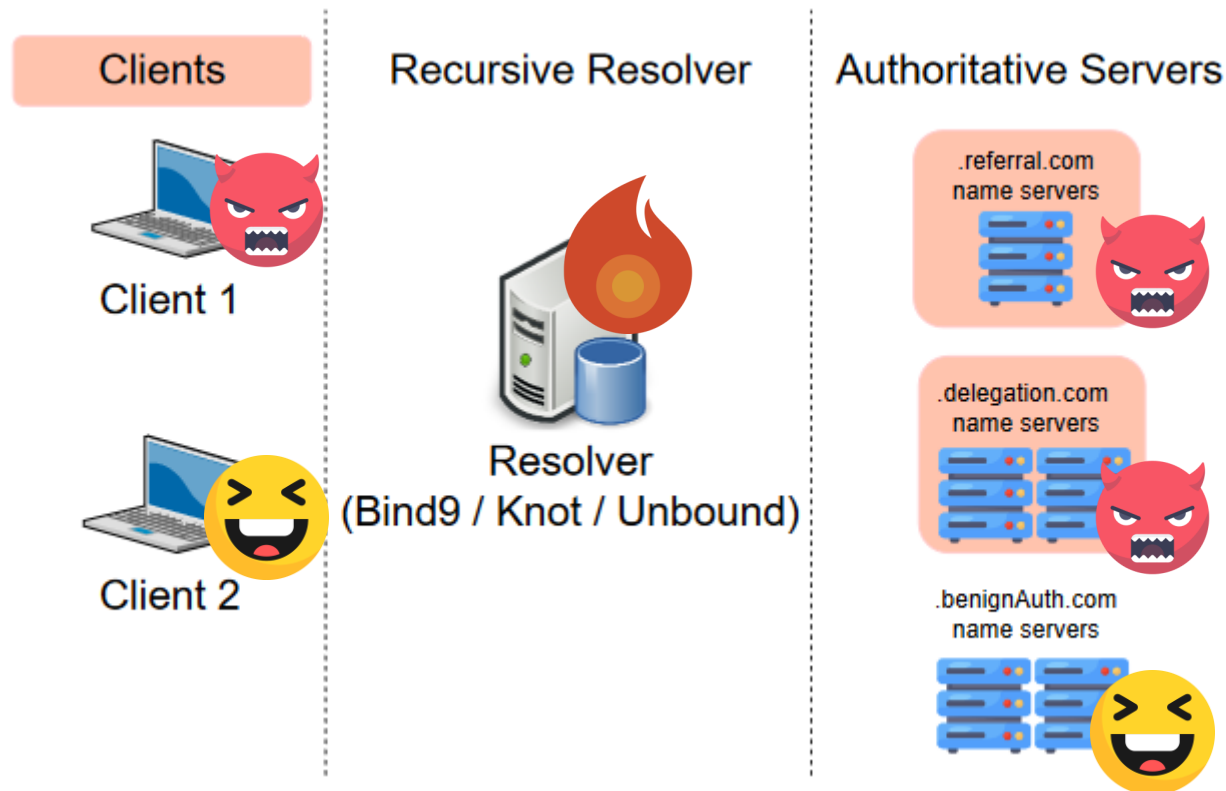
Solutions

- Consider only k of the NS names in the referral response

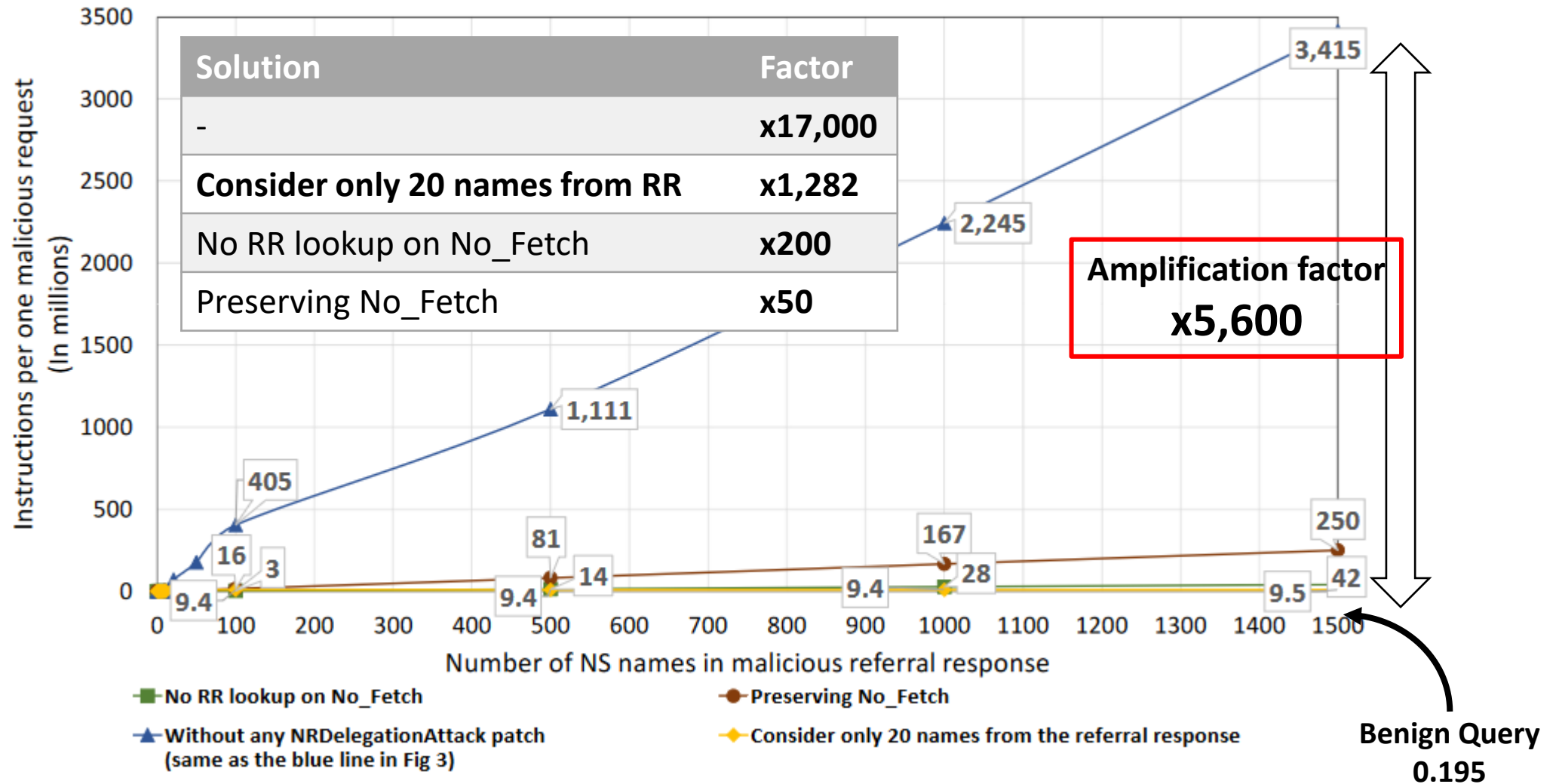


Measurement and Setups

- Placed the client, resolver, and authoritative servers in the **same cloud region (Azure)**
 - Simulated environment
- Intel Xeon Platinum 8272CL (**only 4 vCPU**), 16GB RAM



Complexity factor



Appendix. DNS glue requirements in referral responses

- Published in **September 2023** (after paper) - RFC 9471
- Name server **MUST** include all available glue records for **in-domain** name servers
- Name server **SHOULD** include all available glue records for **sibling 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
```

In-domain (In-bailiwick) name server

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

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

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

Sibling name server

Conclusion

- Well-known open resolvers were vulnerable to NRDelegation attack
- Using delegation response can bypass the parallel queries limit
- A single query from client can make **5,600 queries** with **10^9 machine instructions**
- The solution author suggested still has a high complexity factor

Thank you
