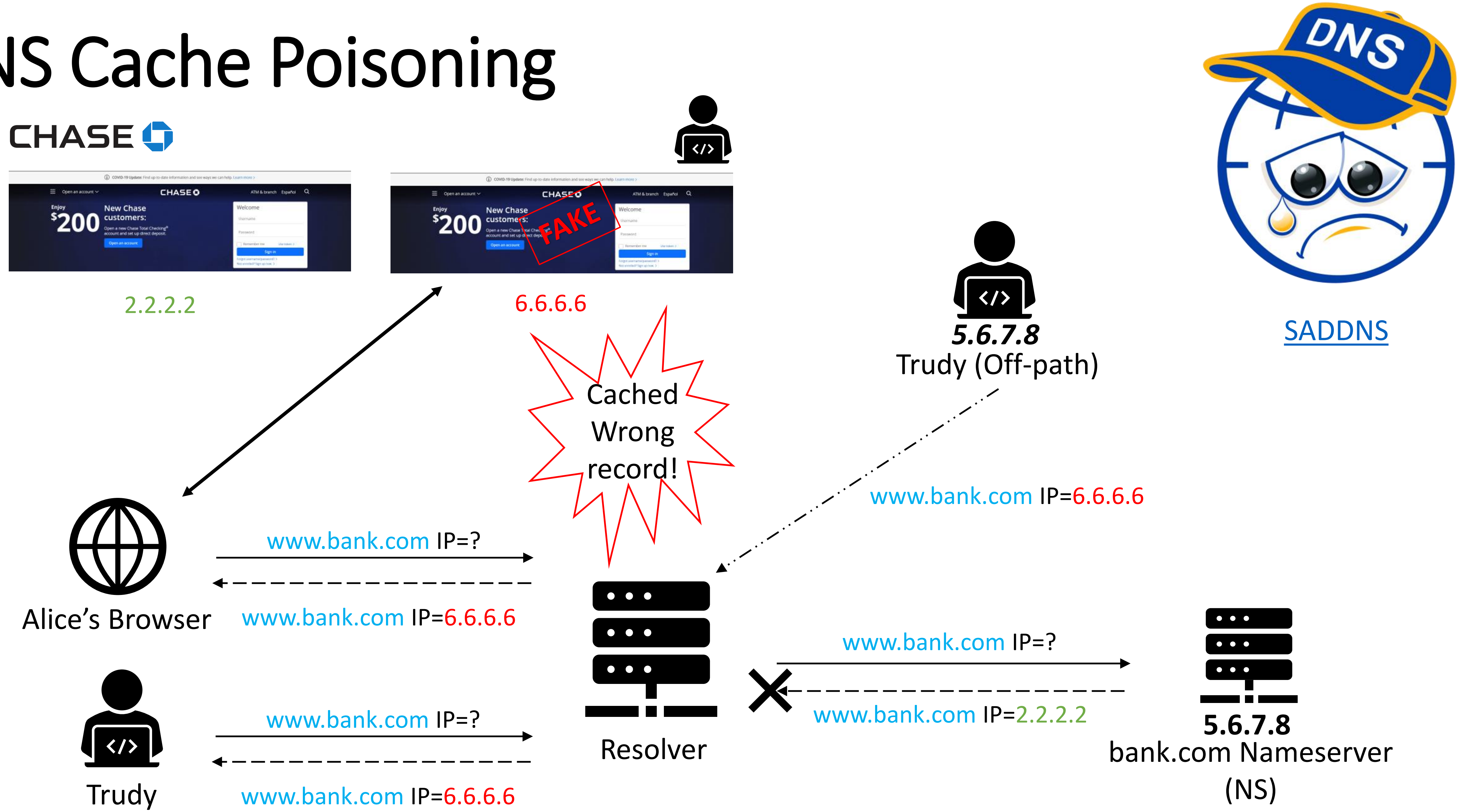
DNS Cache Poisoning Attack: Resurrections with Side Channels

Keyu Man Xin'an Zhou Zhiyun Qian

UC RIVERSIDE

Contents

DNS Cache Poisoning



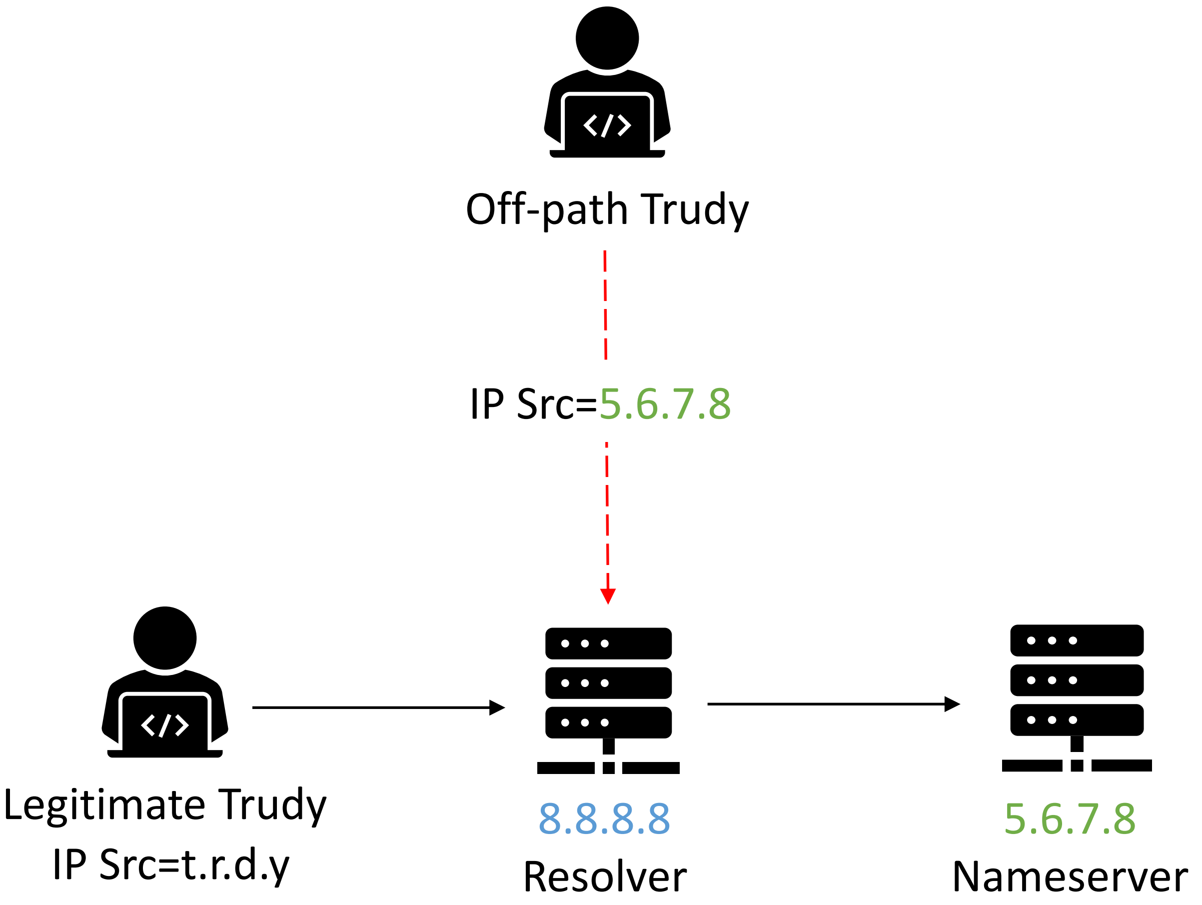
Threat Model

- Off-path attacker
- Attacker can trigger request
  - Wi-Fi router
  - Coffee shop
  - Airport

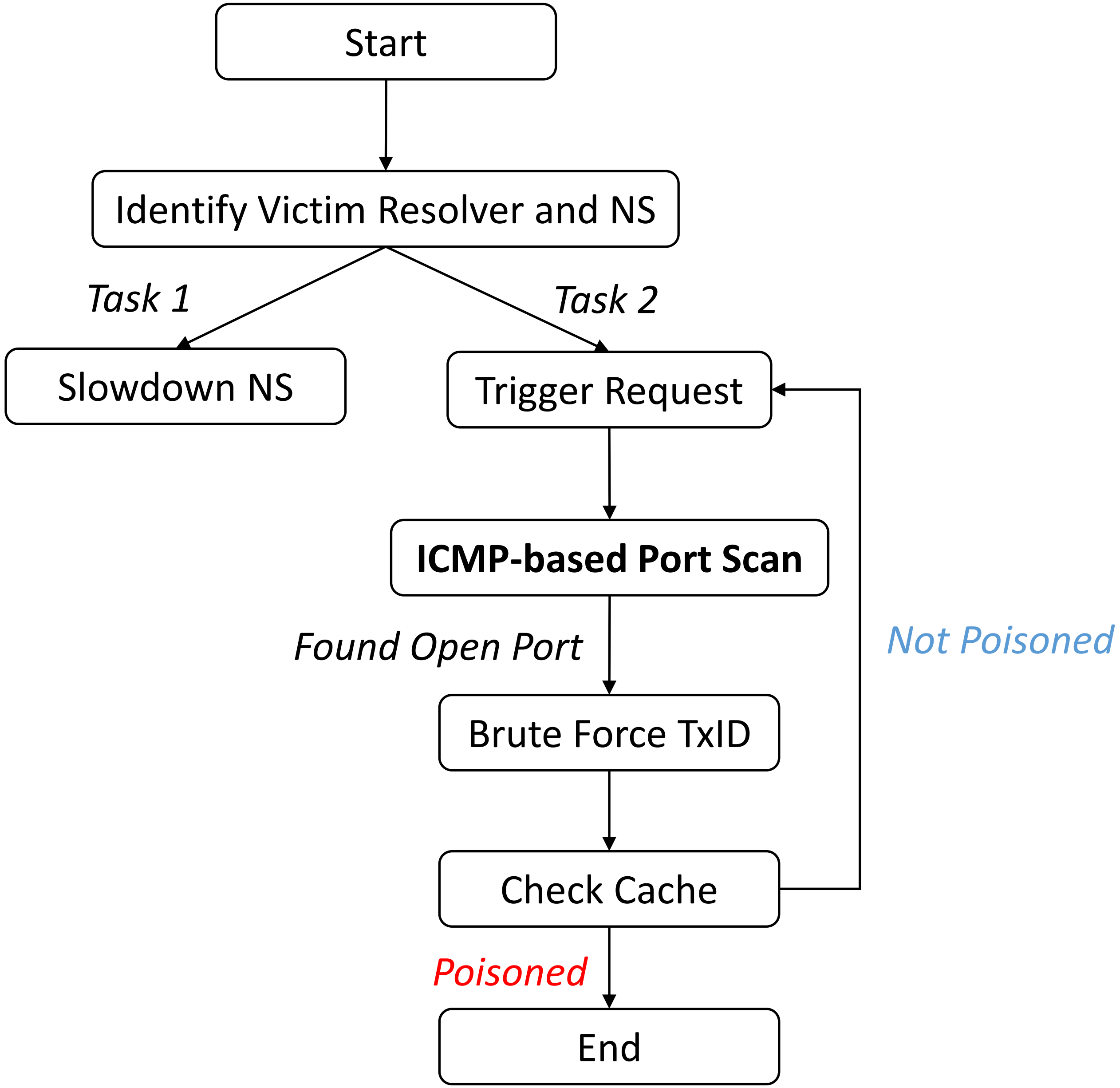
  Forwarders

  - 8.8.8.8
  - 9.9.9.9
  - Campus/ISP DNS server

  Resolvers



Attack Overview



Contents

- Background
- ICMP-Based Port Scan
  - ICMP Processing Logic
  - Public-facing Port Scan
  - Private-facing Port Scan
  - Colliding IP Inference
- Evaluation
- Defenses
- Conclusion
- Disclosure

ICMP Processing Logic

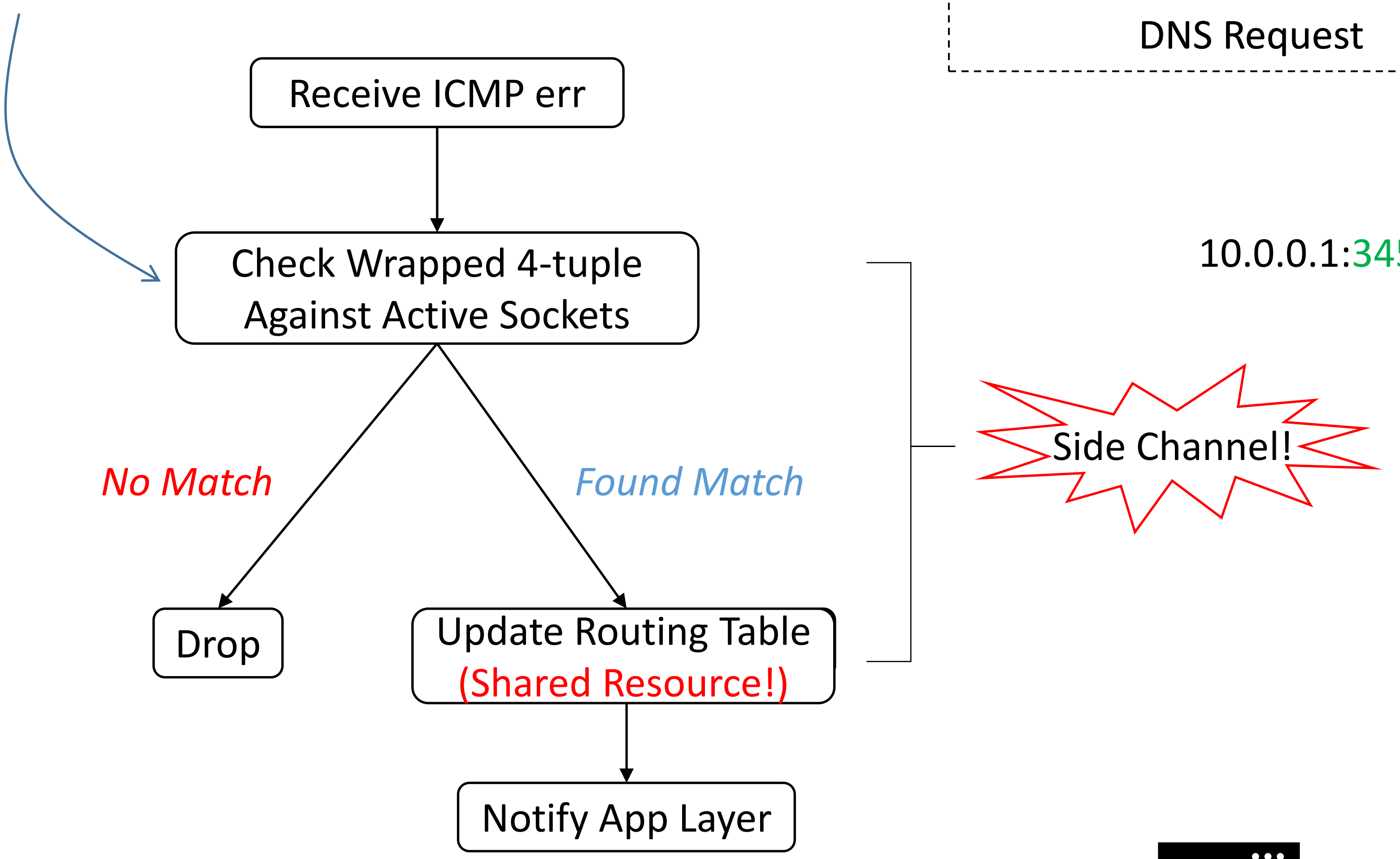
- Why ICMP has to do with DNS attack?
  - It can piggyback port info!

```
> Ethernet II, Src:                                        , Dst:
> Internet Protocol Version 4, Src:                    , Dst:
v Internet Control Message Protocol
    Type: 3 (Destination unreachable)
    Code: 3 (Port unreachable)
    Checksum: 0x89b3 [correct]
    [Checksum Status: Good]
    Unused: 00000000
  > Internet Protocol Version 4, Src:                 , Dst:
  > User Datagram Protocol, Src Port:      , Dst Port:
```

ICMP Processing Logic

How is the port info used?

5.5.5.5->10.0.0.1
ICMP Frag Needed: max MTU=800
Original packet:
10.0.0.1:34568->5.6.7.8:53
DNS Request



Socket table:
10.0.0.1:34567->5.6.7.8:53 CONNECTED
⋮

BIND

DNS Request

ICMP Processing Logic

ICMP Fragment Needed & ICMP Redirect

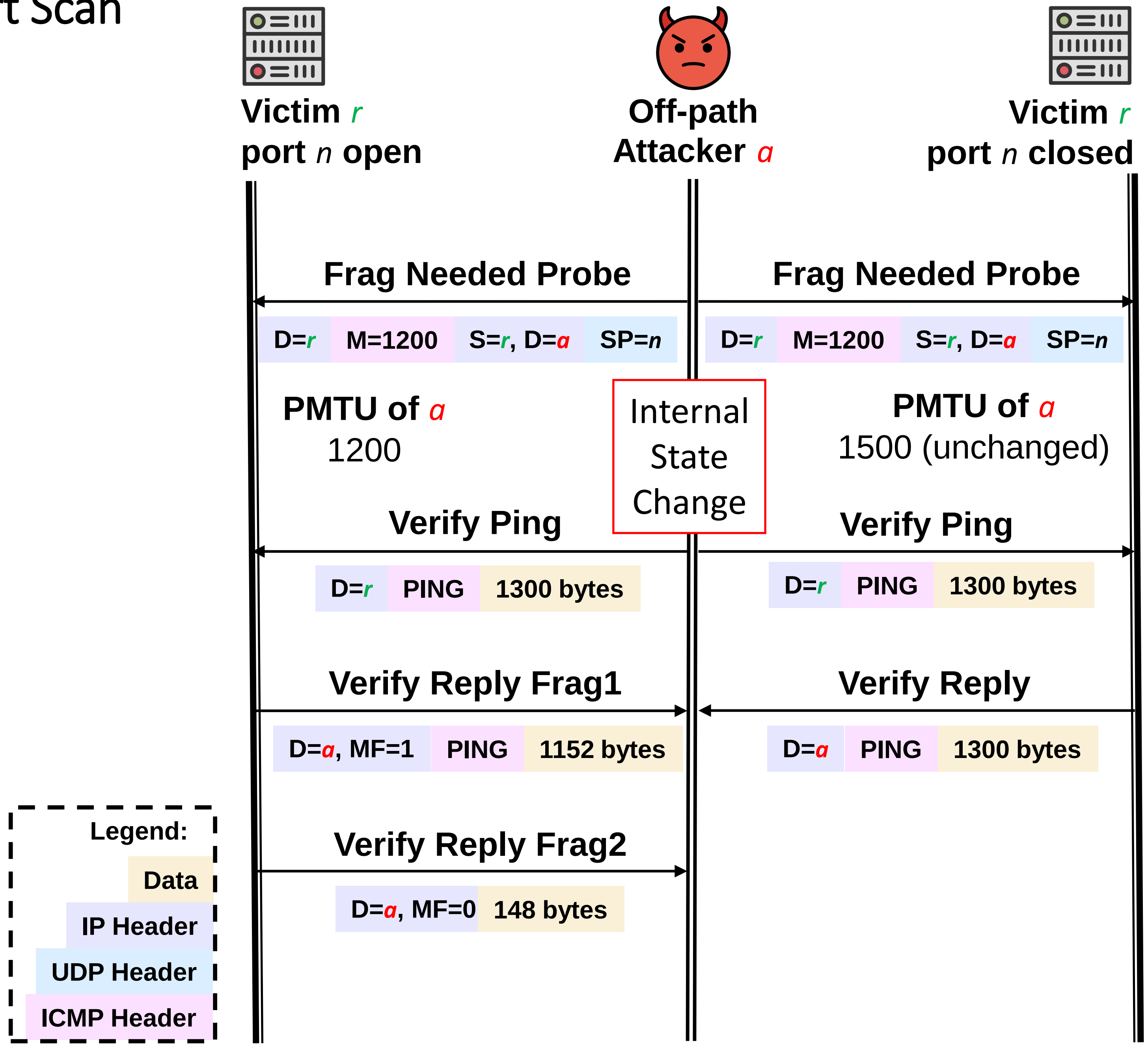
```
if (type == ICMPV6_PKT_TOOBIG) {
    ip6_sk_update_pmtu(skb, sk, info);
}
if (type == NDISC_REDIRECT) {
    ip6_sk_redirect(skb, sk);
}
```

- The only ICMP errs modifying the shared resources
  - i.e., update routing table
- Frag needed
  - Packet exceeds MTU
  - PMTU for a host is updated in routing table
- Redirect *(more details in the paper)*
  - Better routes available
  - Next hop to a host is updated in routing table

Public-facing Port Scan

- `listen()`ing ephemeral ports
  - No check on IP during socket matching
    - Only port # !
    - Alter PMTU of any IP!
  - “public-facing”
  - dnsmasq
- Idea
  - Try lowering attacker’s own PMTU
  - Check fragments

Public-facing Port Scan



Keys: D=Destination IP, S=Source IP, M=PMTU, SP=Source Port, MF=More Fragment

Private-facing Port Scan

- `connect()`ed ephemeral ports
  - Complete 4-tuple is checked
    - Only NS' PMTU can be modified
      - Unknown to the attacker directly
  - "private-facing"
  - BIND, Unbound, …
- How to observe the change of NS' PMTU?
  - Next Hop Exception (`fnhe`) Cache

Private-facing Port Scan

fnhe Cache

Buckets	0	1	...	756	...	2046	2047

Slots	
2001::1 / 3001::1	H(2001::1,key)=756 / H(3001::1,key)=756
2002::2	H(2002::2,key)=756
2003::3	H(2003::3,key)=756
2004::4	H(2004::4,key)=756
2005::5	H(2005::5,key)=756

- ICMP caused route changes
  - i.e., PMTU & next hop
  - Unexpected
  - “next hop exceptions” (fnhe)
- fnhe
  - Cached
    - 2048-bucket hash table
    - 5 slots per bucket to solve collision
    - Random seed as hash key
    - H(): IP addr->bucket
  - Garbage collected
    - Overwrite the oldest slot when bucket is full

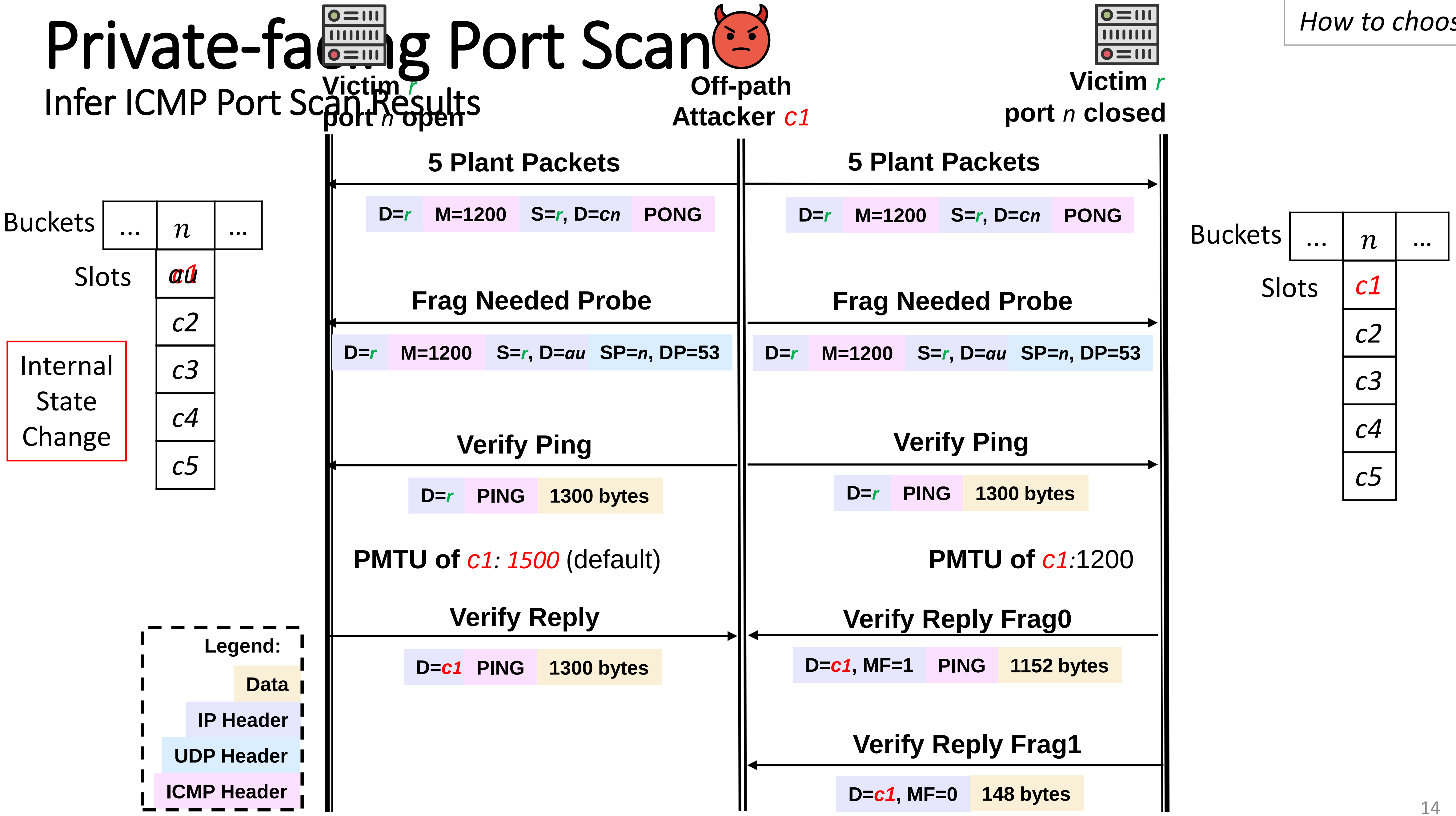
2001::1 and 3001::1 are colliding IPs

Infer ICMP Port Scan Results

Private-facing Port Scan

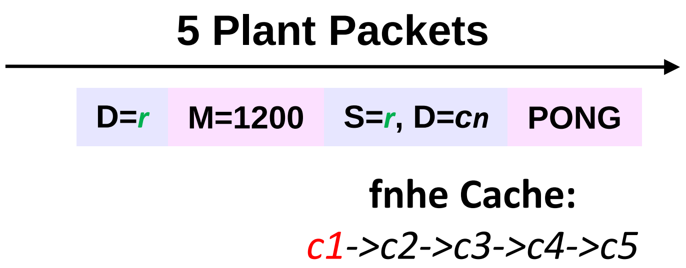
Infer ICMP Port Scan Results

Next slide:
How to choose c1-c5



Keys: D=Dst IP, S=Src IP, M=PMTU, SP=Src Port, DP=Dst Port, MF=More Fragment, cn(c1-c5)=Colliding IPs, au=Authoritative Name Server, PONG=Ping Reply

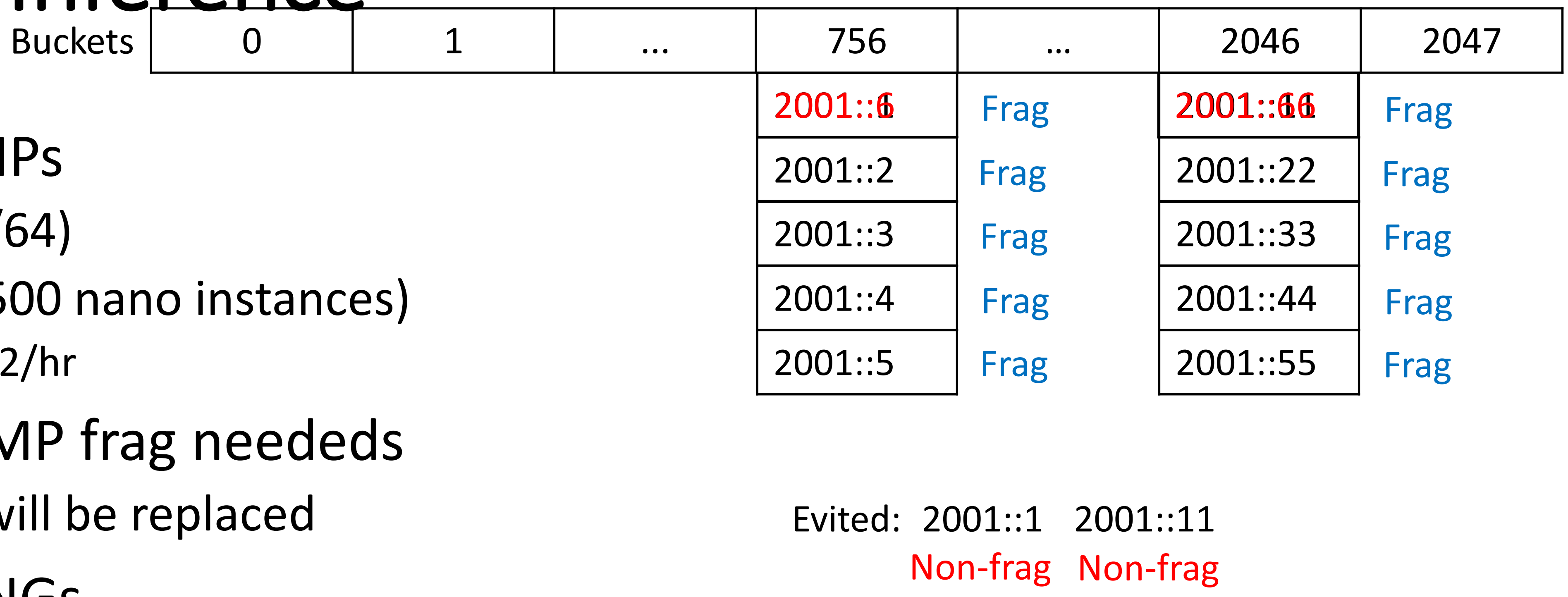
Colliding IP Inference



- How to choose c1-c5?
  - Collide with NS' IP (*au*)
  - Collide with each other
  - Attacker controls c1
- Idea: Hash key inference->calculate c1-c5

Colliding IP Inference

Key Inference



1. Control 1500 IPs
   - Easy for IPv6 (/64)
   - AWS if IPv4 (1500 nano instances)
     - 1500*$0.0042/hr
2. Send 1500 ICMP frag neededs
   - Some entries will be replaced
3. Send 1500 PINGs
   - Log IPs (evicted IPs) replying no frags
4. Brute-force key by simulating 2&3 locally
   - Key=0, key=1,…, key=0xffffffff
   - Check if the evicted IPs match
   - Only 2-min guess after distributing to 1500 nano instances

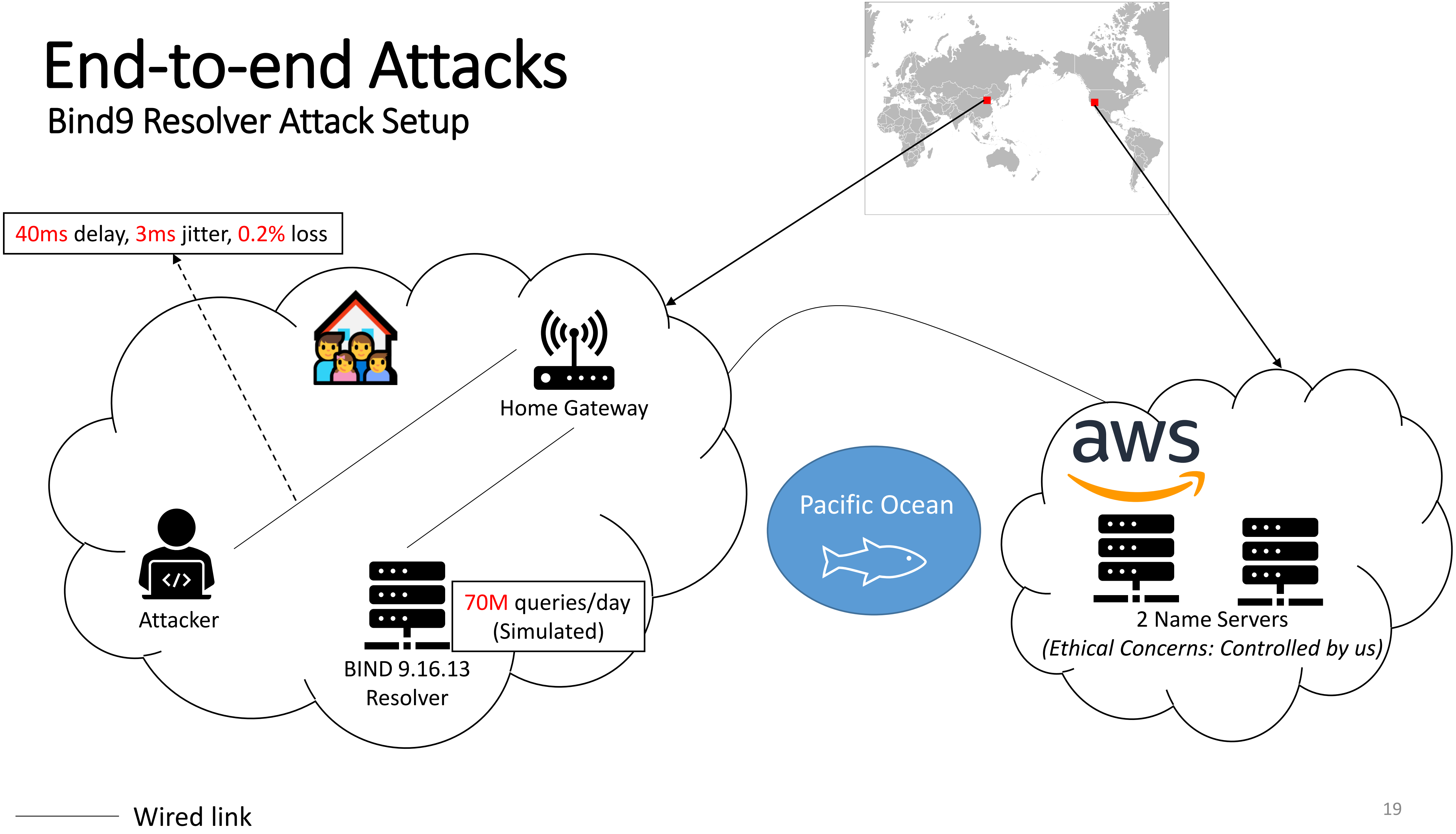
Contents

Comparison with SADDNS

+ Novel port inference method
    - ICMP vs. UDP
+ New dimension of the shared resource
    - Spatial vs. temporal
    - fnhe cache hash table vs. ICMP rate limit counter
+ Fast port scan speed
    - Unlimited vs. 1000 pps
+ Resistant to noise
    - No time sync vs. 50-ms time sync

- Preparation of the attack
    - Inferring colliding IPs (hash key)

End-to-end Attacks
Bind9 Resolver Attack Setup



End-to-end Attacks

Attack Results

- Average success time: 80-900s
  - Time varies due to slightly different setup *(more details in the paper)*
- Other attacks:
  - Forwarder attack: 13s
  - Real public resolver attack: 105s avg.

Attack Results

```
keyu@ubuntu:~$ dig @        a.xiaofengtest.net
; <<>> DiG 9.10.3-P4-Ubuntu <<>> @
a.xiaofengtest.net
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59301
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;a.xiaofengtest.net.            IN      A

;; ANSWER SECTION:
a.xiaofengtest.net.     28      IN      A       6.6.6.6

;; Query time: 190 msec
;; SERVER:        #53(       )
;; WHEN: Fri Sep 24 17:11:34 EDT 2021
;; MSG SIZE  rcvd: 63
```

```
keyu@ubuntu:~$ dig @        a.xiaofengtest.net
; <<>> DiG 9.10.3-P4-Ubuntu <<>> @
a.xiaofengtest.net
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57535
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;a.xiaofengtest.net.            IN      A

;; ANSWER SECTION:
a.xiaofengtest.net.     500     IN      A       1.2.3.4

;; Query time: 448 msec
;; SERVER:        #53(       )
;; WHEN: Fri Sep 24 17:11:36 EDT 2021
;; MSG SIZE  rcvd: 63
```

Vulnerable Population
Open-source Software

- Vulnerability in both OS and DNS software
  - Varies among diff. kernel and software combinations.
- OS
  - Linux: 3.6-5.14
- DNS
  - BIND: 9.3-9.16
  - Unbound: <1.13
  - dnsmasq: any *(at the time of testing)*

Vulnerable Population
Open Resolvers

- Open resolvers
  - 14% of backend IPs
  - 38% of frontend IPs
- Public resolvers
  - 6 out of 12

Contents

- Background
- ICMP-Based Port Scan
- Evaluation
- Defenses
- Conclusion
- Disclosure

Defenses

- Defeat off-path attacks
  - 0x20 eNcoDinG
  - DNS cookie
  - DNSSEC
- Mitigate the side channel
  - Set `IP_PMTUDISC_OMIT` socket option
  - Randomize `fnhe` caching
    - Eviction policy, bucket depth…

Contents

- Background
- ICMP-Based Port Scan
- Evaluation
- Defenses
- Conclusion
- Disclosure

Conclusion

- A novel side channel from next hop exception cache
- ICMP-based port scan
- Poison the cache of DNS in minutes
- Update Linux kernel to mitigate the attack

Contents

- Background
- ICMP-Based Port Scan
- Evaluation
- Defenses
- Conclusion
- Disclosure

Disclosure

Thank you!

Source code & more interesting projects
https://github.com/seclab-ucr/

Keyu Man
kman001@ucr.edu