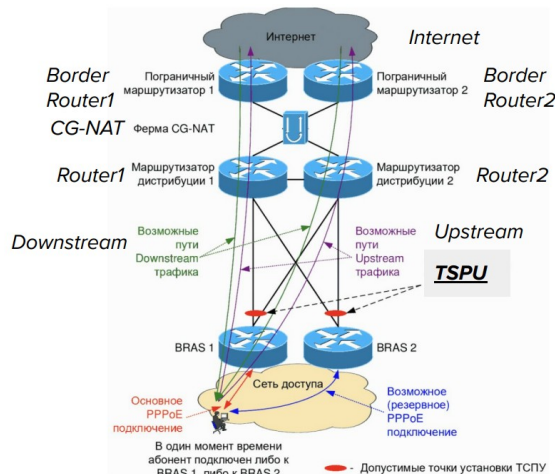
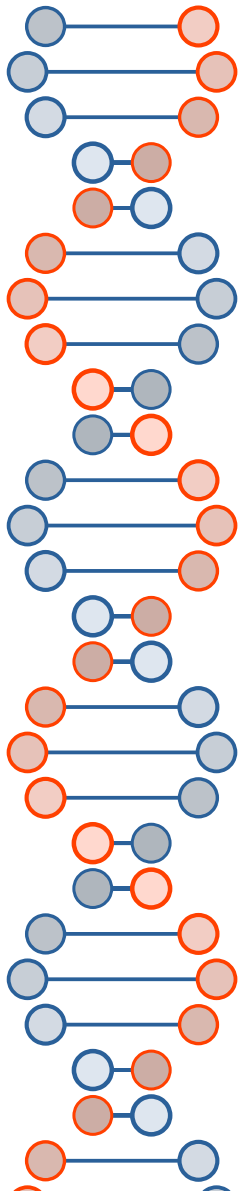


Shared Intermediary State Confusion

CSE 468 Fall 2026
jedimaestro@asu.edu





==Phrack Magazine==

Volume Four, Issue Forty-Three, File 6 of 27

-:[Phrack Prophile]:-

This issue our prophile features a hacker who has been around forever, who's been there and done that, literally. His handle is Dr. Who.

Hacking is the art of esoteric quests, of priceless and worthless secrets. Odd bits of raw data from smashed machinery of intelligence and slavery reassembled in a mosaic both hilarious in its absurdity and frightening in its power.

<https://phrack.org/issues/43/6>

“

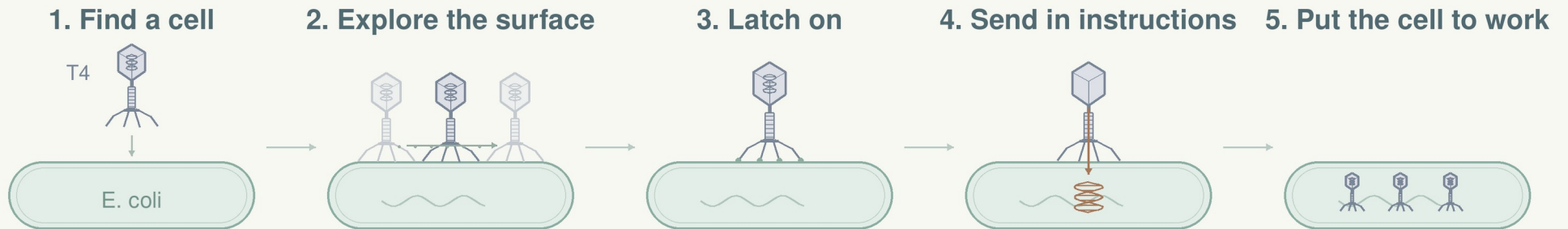
In effect, information only has meaning in that it is subject to interpretation.”

— Fred Cohen

Computer Viruses — Theory and Experiments (1984).

all.net/books/virus/part3.html

T4 virus infecting an E. coli bacterium



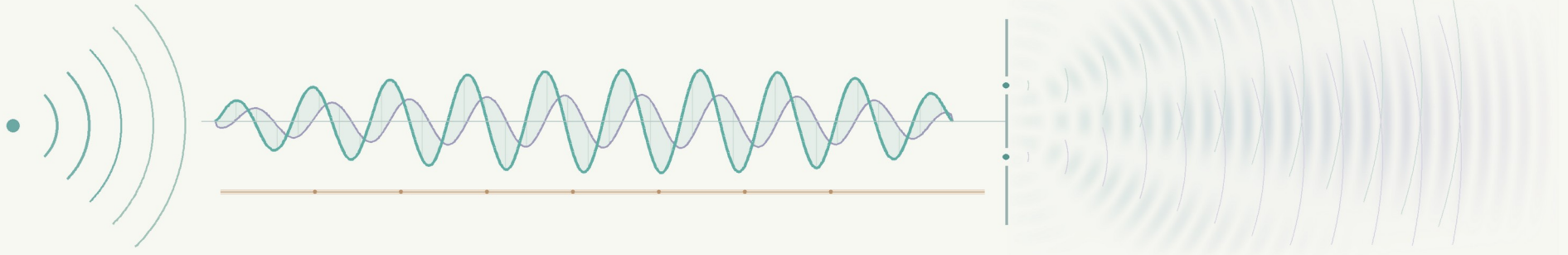
Surface exploration: Dreesens et al., Microbiology Spectrum (2025), doi:10.1128/spectrum.02509-25.

“Information is inevitably physical.”

— **Rolf Landauer**

“Information Is Inevitably Physical,” in *Feynman and Computation* (1999), pp. 77–92.

doi:10.1201/9780429500459-8



Outline

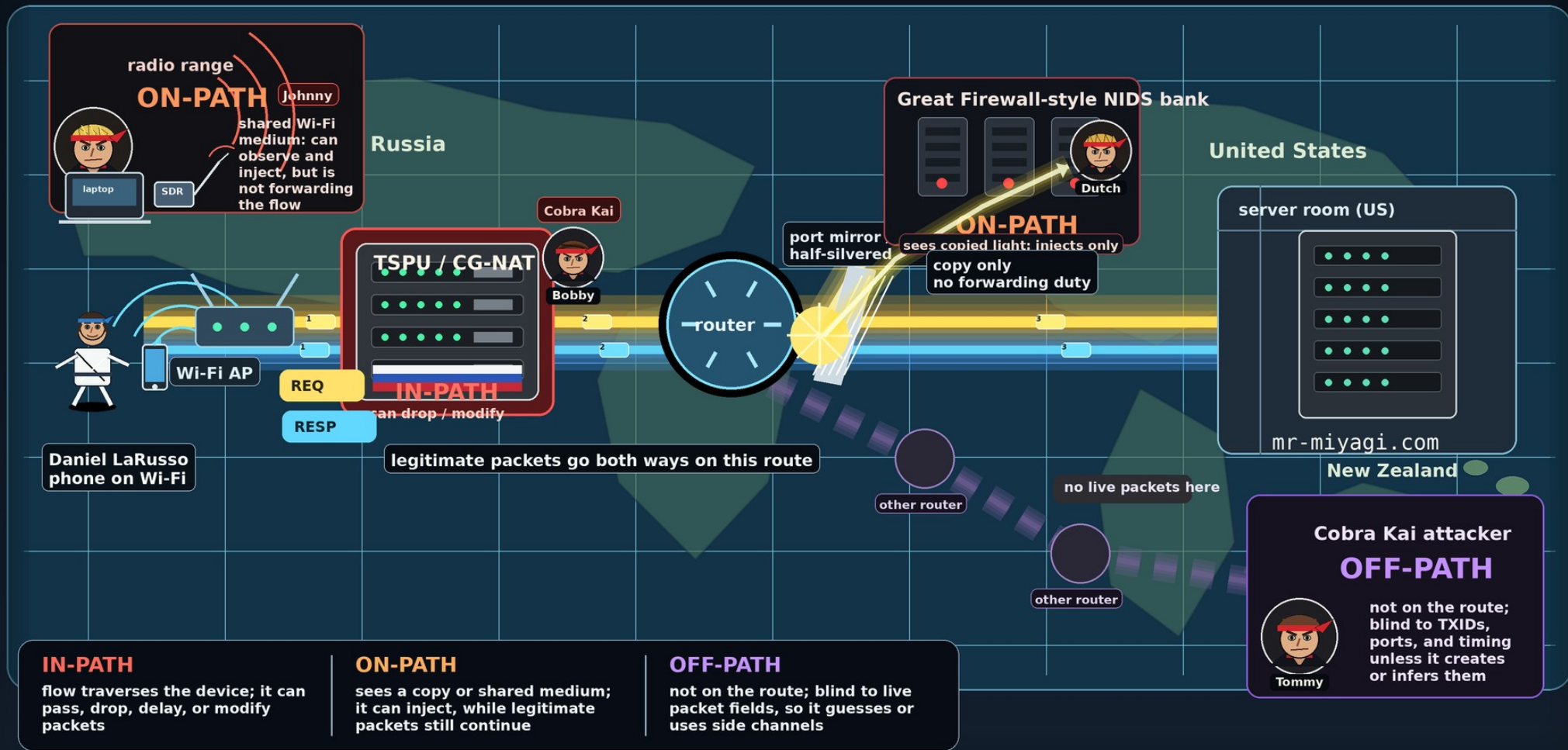
- Review: On-path vs. in-path vs. off-path
- PAD-to-PAD (1980s and early 1990s, X.25 networks)
- Other examples: ARP poisoning, BGP attacks, and ATIP
- Dan Kaminsky's DNS poisoning attack (2008) (concurrency)
- Future lectures
 - Port scanning
 - Side channel attacks (information theory)
 - Example: Fragmentation and IPIDs
 - Solution: RSA signatures
 - Important ingredient for signatures: *extended Euclidean algorithm*

Where do Internet standards come from?

- IETF = Internet Engineering Task Force
- RFC = Request for Comments
 - MUST, MUST NOT, SHOULD, SHOULD NOT, MAY (RFC 2119)
- “The only laws on the Internet are assembly and RFCs” -- Phrack 65
 - Assembly is an abstraction
 - RFCs are not always followed
 - Often ambiguous

In-Path, On-Path, and Off-Path Attackers

Bright route: Daniel LaRusso's legitimate packets to mr-miyagi.com Dark background: Cobra Kai vantage points



On-path DNS attack (easy)

- Need to respond faster than the DNS server
 - Not hard, tens of milliseconds is an eternity
 - Maybe DoS the DNS server
- Need to get the TXID and source port correct in the response
 - Trivial, just read them from the request packet

In-path DNS attack (even easier)

- ~~Need to respond faster than the DNS server~~
 - ~~Not hard, tens of milliseconds is an eternity~~
 - ~~Maybe DoS the DNS server (just don't forward the request, or do forward it but change the response on the way back)~~
- Need to get the TXID and source port correct in the response
 - Trivial, just read them from the request packet

Off-path attack

- Need to respond faster than the DNS server
 - DoS the DNS server
- Need to get the TXID and source port correct in the response
 - **Not easy**, being off path means you're *blind* to these values
 - Guessing might work ($2^{16} * 2^{16} = 2^{32}$)
 - Side channels and birthday attacks can do even better
- Need to know what was queried and when
 - Cache poisoning (you know these things because you caused it)



Shared Intermediary State Confusion

- Typically a way of going from off-path to in-path (e.g., DNS cache poisoning)
 - But you can also go from on-path to in-path (e.g., ARP cache poisoning), or off-path to on-path (can't think of an example, but ATIP attacks go from off-path to blind in-path)
- “Shared Intermediary” ... attacker and victim share a DNS resolver, layer 2 switch, BGP transit AS routing plane, VPN server, X.25 PAD, *etc.*
- “State Confusion” ... a port assignment, prefix routing state, *etc.* conflict happens, causing “wires to get crossed”



PAD-to-PAD as an example of Shared
Intermediary State Confusion...



PAD-to-PAD attack

Schematic map; numeric NUAs are fictional X.121-style examples, not assigned network addresses.

NUA Network User Address

A numeric X.121 address:
roughly a phone number
for a PAD, host, or port.

TRW employee

```
ATDT 555-1832
CONNECT 1200
PUBLIC X.25 PAD
PORT 12 PAD>
```

1. TRW employee dials into the public PAD.
The PAD allocates a temporary local slot: line 12.

Public PAD

NUA 031104170000

line 11
line 12
line 13
temporary port slots

X.25 packet-switched network

calls are routed by NUAs

Host NUA

```
031104170042
4.3BSD UNIX
(alphalamp)
tty07 login:
```

Parmaster

```
PARMASTER PAD>
ready
```

Parmaster's access site

note service site

PAD-to-PAD attack

Schematic map; numeric NUAs are fictional X.121-style examples, not assigned network addresses.

NUA Network User Address

A numeric X.121 address:
roughly a phone number
for a PAD, host, or port.

TRW employee

```
ATDT 555-1832
PUBLIC X.25 PAD
PORT 12
PAD> _
```

Public PAD

NUA 031104170000

line 11
line 12
line 13
temporary port slots

X.25 packet-switched network

calls are routed by NUAs

2. TRW employee is idle at the PAD prompt.
The line is assigned, but no host NUA
has been entered yet.

Remote service site

Host NUA

```
031104170042
4.3BSD UNIX
(alphalamp)
tty07 login:
```

Parmaster's access site

Parmaster

```
PARMASTER PAD>
ready
```

PAD-to-PAD attack

Schematic map; numeric NUAs are fictional X.121-style examples, not assigned network addresses.

NUA Network User Address

A numeric X.121 address:
roughly a phone number
for a PAD, host, or port.

TRW employee

```
ATDT 555-1832
PUBLIC X.25 PAD
PORT 12
PAD> _
```

Public PAD

NUA 031104170000

line 11
line 12
line 13

temporary port slots

X.25 packet-switched network

calls are routed by NUAs

3. Parmaster calls the public PAD.

His request travels through the X.25 network
and reaches the exposed line 12.

Remote service site

Host NUA

```
031104170042
4.3BSD UNIX
(alphalamp)
tty07 login:
```

Host's access site

Parmaster

```
PARMASTER PAD>
ready
C 031104170012
```

PAD-to-PAD attack

Schematic map; numeric NUAs are fictional X.121-style examples, not assigned network addresses.

NUA Network User Address

A numeric X.121 address:
roughly a phone number
for a PAD, host, or port.

X.25 packet-switched network

calls are routed by NUAs

Public PAD

NUA 031104170000

line 11

line 12

line 13

temporary port slots

TRW employee

ATDT 555-1832

PAD> C

031104170042

Host NUA

031104170042

4.3BSD UNIX

(alphalamp)

tty07 login:

Remote service site

Parmaster's access site

Parmaster

PARMASTER PAD>

ready

saw C 031104170042

4. TRW employee types the
destination NUA.

Parmaster sees it on line 12.

PAD-to-PAD attack

Schematic map; numeric NUAs are fictional X.121-style examples, not assigned network addresses.

NUA Network User Address

A numeric X.121 address:
roughly a phone number
for a PAD, host, or port.

X.25 packet-switched network

calls are routed by NUAs

Public PAD

NUA 031104170000

line 11

line 12

line 13

temporary port slots

TRW employee

ATDT 555-1832

PAD> C

031104170042

Remote service site

Host NUA

031104170042

4.3BSD UNIX

(alphalamp)

tty07 login:

5a. Request: Parmaster calls the host NUA.
He fetches the expected login banner.

Parmaster's access site

Parmaster

PARMASTER PAD>

ready

C 031104170042

PAD-to-PAD attack

Schematic map; numeric NUAs are fictional X.121-style examples, not assigned network addresses.

NUA Network User Address

A numeric X.121 address:
roughly a phone number
for a PAD, host, or port.

X.25 packet-switched network

calls are routed by NUAs

Public PAD

NUA 031104170000

line 11

line 12

line 13

temporary port slots

5b. Response: Parmaster gets the banner.

He relays it through the public PAD to the employee.

TRW employee

ATDT 555-1832

4.3BSD UNIX

(alphalamp)

tty07 login:_

Remote service site

Host NUA

031104170042

4.3BSD UNIX

(alphalamp)

tty07 login:

Parmaster

PARMASTER PAD>

ready

RX: 4.3BSD UNIX

Step 6: crack your knuckles



PAD-to-PAD attack

Schematic map; numeric NUAs are fictional X.121-style examples, not assigned network addresses.

NUA Network User Address

A numeric X.121 address:
roughly a phone number
for a PAD, host, or port.

7. TRW employee types credentials.

They travel geographically toward Parmaster's access site.

note service site

Host NUA

```
031104170042
4.3BSD UNIX
(alphalamp)
tty07 login:
```

X.25 packet-switched network

calls are routed by NUAs

Public PAD

NUA 031104170000

line 11

line 12

line 13

temporary port slots

TRW employee

```
ATDT 555-1832
4.3BSD UNIX
(alphalamp)
tty07 login: trw01
Password: *****
```

Parmaster's access site

Parmaster

```
PARMASTER PAD>
ready

captured: trw01 / *****
```

PAD-to-PAD attack

Schematic map; numeric NUAs are fictional X.121-style examples, not assigned network addresses.

NUA Network User Address

A numeric X.121 address:
roughly a phone number
for a PAD, host, or port.

TRW employee

ATDT 555-1832

Public PAD

NUA 031104170000

line 11

line 12

line 13

temporary port slots

X.25 packet-switched network

calls are routed by NUAs

Remote service site

Host NUA

031104170042
4.3BSD UNIX
(alphalamp)
tty07 login:

Parmaster's access site

Parmaster

PARMASTER PAD>
ready

What changed?

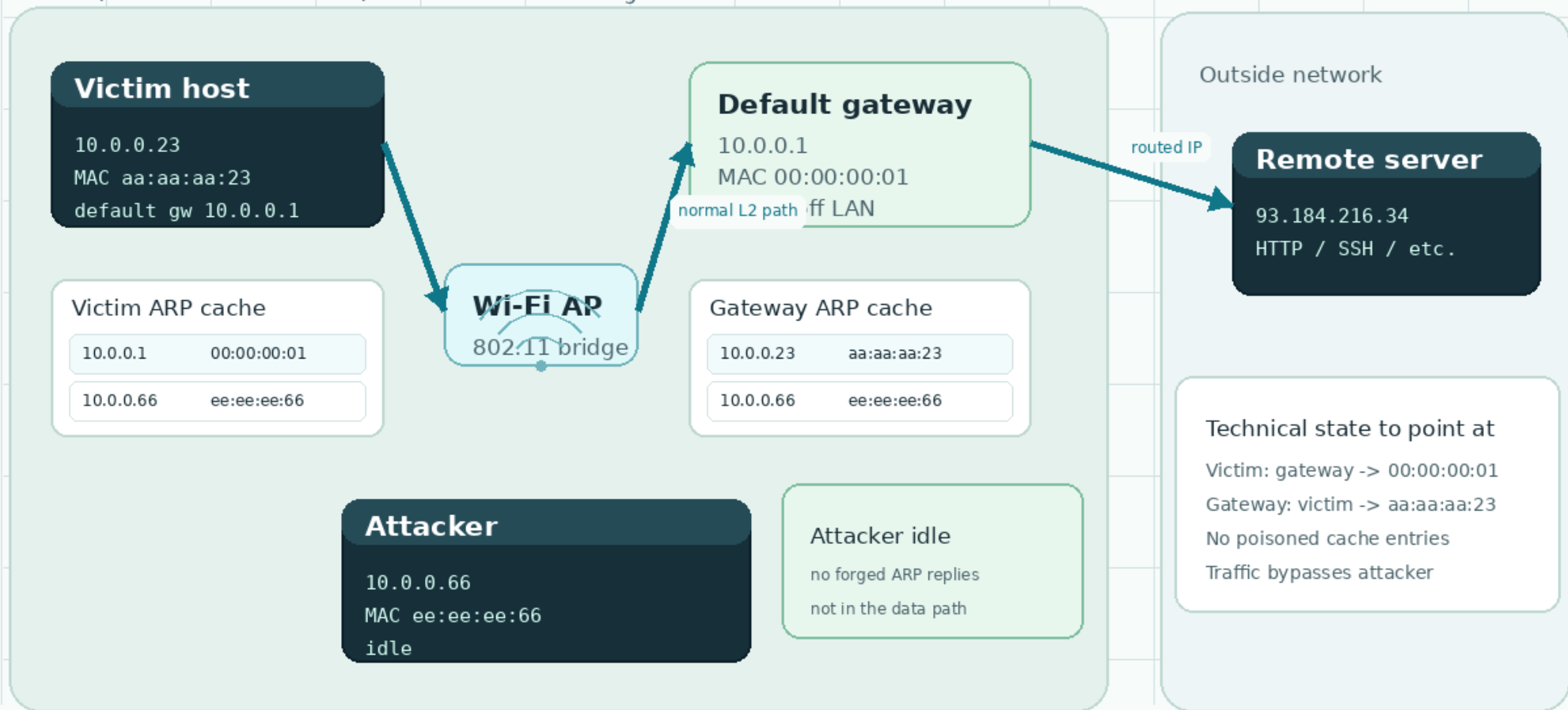
TRW employee thought line 12 connected to 031104170042.
The PAD's shared state instead made Parmaster the endpoint.
All traffic to/from the TRW employee passes through the public PAD.

ARP cache poisoning, BGP attacks, and ATIP as
examples of Shared Intermediary State
Confusion...



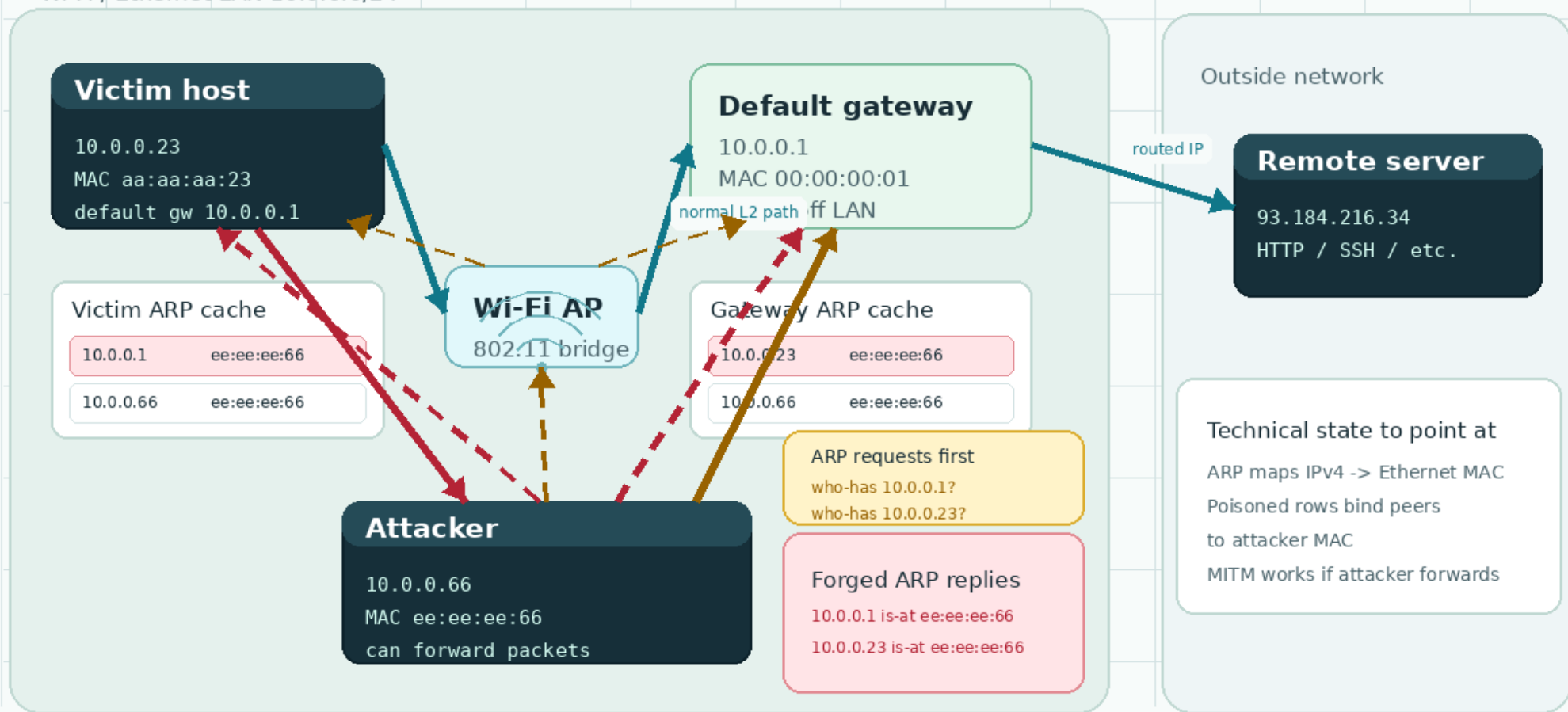
ARP cache state before poisoning

Wi-Fi / Ethernet LAN 10.0.0.0/24: correct MAC bindings



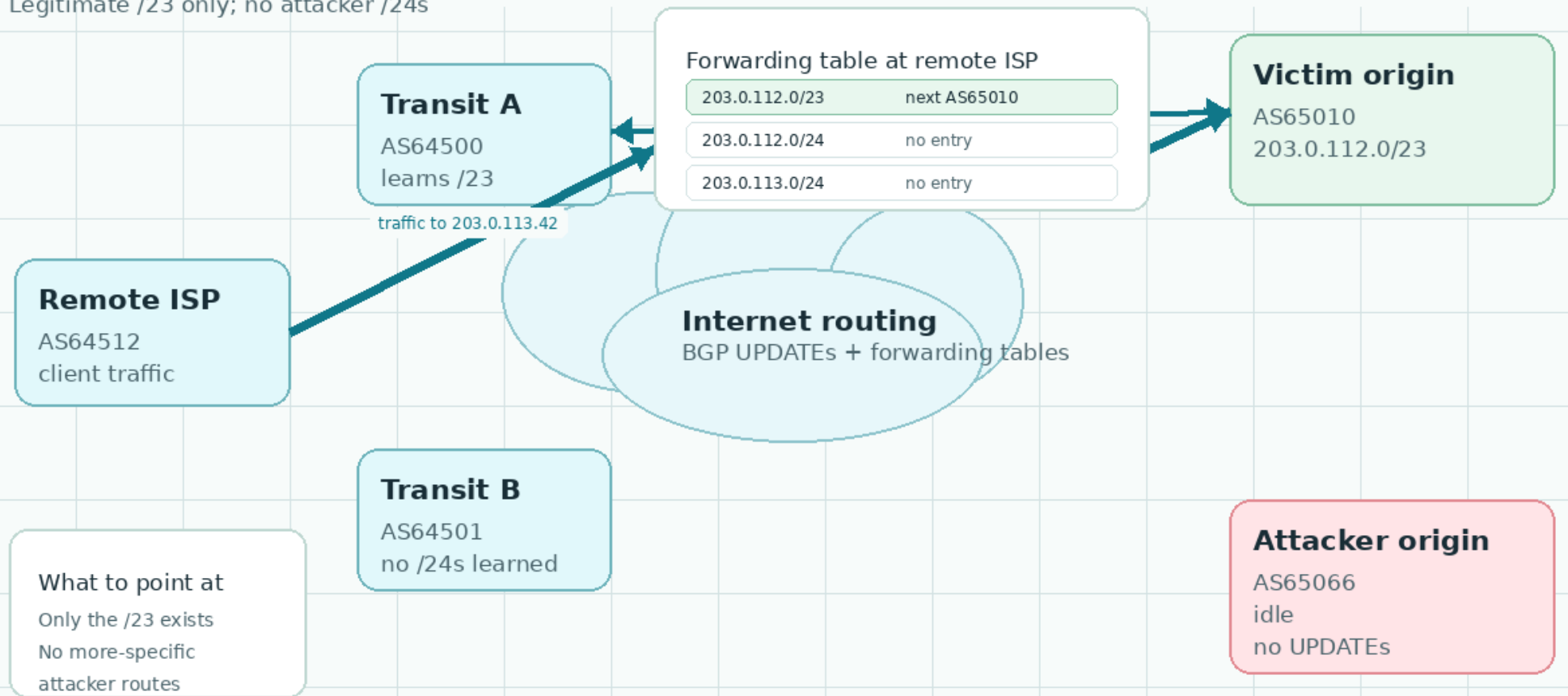
ARP cache poisoning

Wi-Fi / Ethernet LAN 10.0.0.0/24



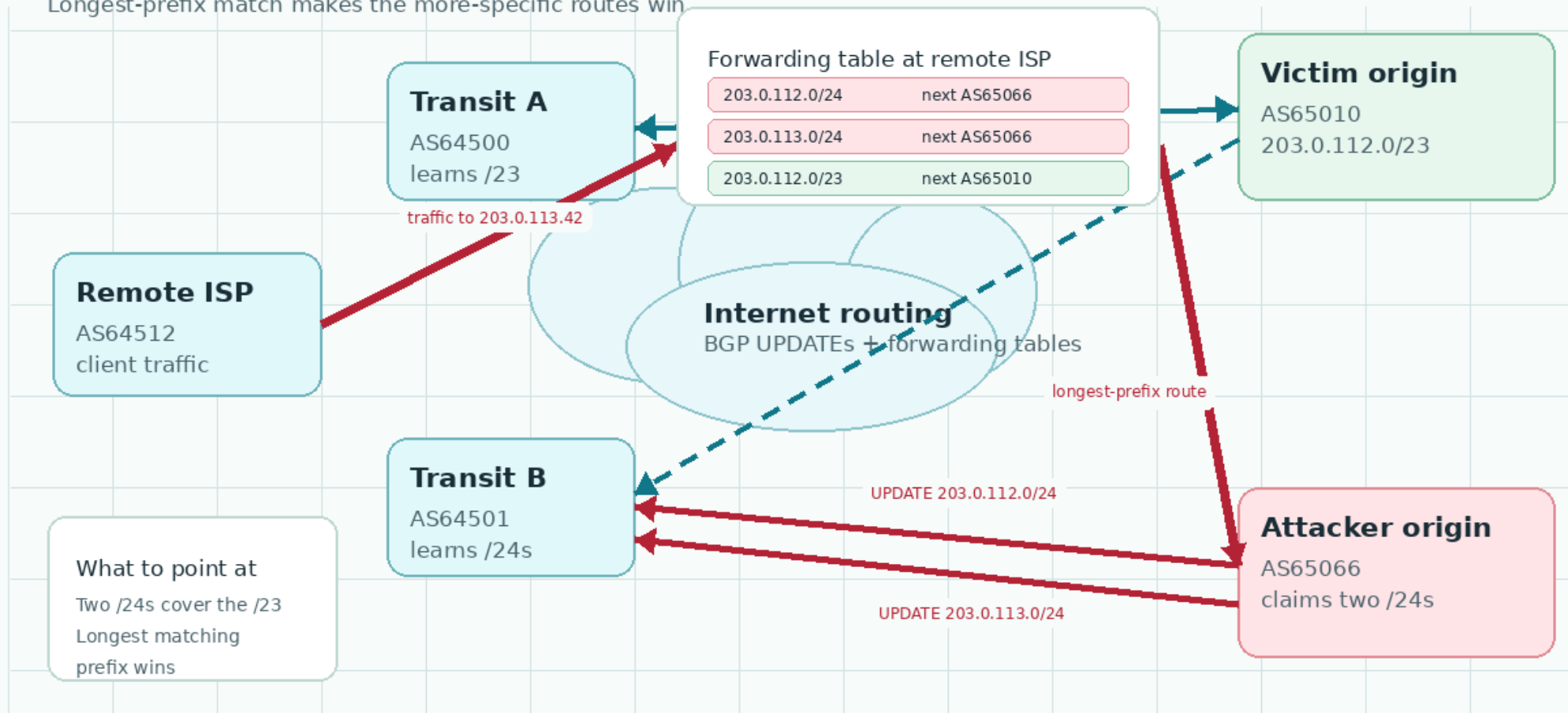
BGP route state before hijack

Legitimate /23 only; no attacker /24s



BGP more-specific prefix hijack

Longest-prefix match makes the more-specific routes win



ATIP before port shadow

A is only an adjacent VPN peer; no colliding row exists yet

Public side + VPN server

Target B

public 3.3.3.3:53144
VPN not started
no 10.0.0.3 yet

future handshake would reach N

VPN server N

public 2.2.2.2:1194
A tunnel only
no shadow collision

outer VPN tunnel



Attacker A

public 1.1.1.1:40001
private 10.0.0.2
idle; adjacent only

VPN server conntrack + NAT entries

ct state

A first VPN tunnel

orig 1.1.1.1:40001 -> 2.2.2.2:1194
reply 2.2.2.2:1194 -> 1.1.1.1:40001
action: ordinary VPN control flow

ESTABLISHED

No port shadow row

inside orig not present
reply match not present
action: handshake would go to VPN process

ABSENT

B VPN tunnel

orig not present yet
reply not present yet
action: target has not connected

ABSENT

Technical state to point at

Only A's tunnel exists
No row for B's public port
Handshake would hit VPN process
A remains adjacent only

ATIP middle: port shadow primed

A has created the inside-orig row; target has not started the VPN handshake yet

Public side + VPN server

Target B

public 3.3.3.3:53144
VPN not started
no 10.0.0.3 yet

target idle

TTL-limited probe



TTL expires before B

VPN server N

public 2.2.2.2:1194
conntrack row exists
waiting for handshake

outer VPN tunnel



tunneled packet

Attacker A

public 1.1.1.1:40001
private 10.0.0.2
sends TTL-limited probe

VPN server conntrack + NAT entries

ct state

A first VPN tunnel

orig 1.1.1.1:40001 -> 2.2.2.2:1194
reply 2.2.2.2:1194 -> 1.1.1.1:40001
action: ordinary tunnel for A

ESTABLISHED

PORT SHADOW NAT ROW (primed)

inside orig 10.0.0.2:1194 -> 3.3.3.3:53144
reply match 3.3.3.3:53144 -> 2.2.2.2:1194
NAT action: dst 2.2.2.2:1194 -> 10.0.0.2:1194

NEW / UNREPLIED

B VPN tunnel

orig not present yet
reply not present yet
action: target has not connected

ABSENT

Primed shared state

Inside-orig row already exists
Reply tuple will match B
Probe dies before target
Next handshake hits NAT first
Proto must match listener

ATIP after port shadow

A primes conntrack so the target's VPN handshake collides and routes through A

Public side + VPN server

Target B

```
public 3.3.3.3:53144
private 10.0.0.3
thinks N is upstream
```

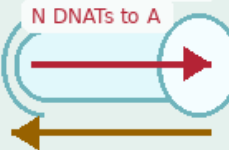
VPN handshake

VPN server N

```
public 2.2.2.2:1194
conntrack checked first
port collision wins
```

A first tunnel

N DNATs to A



A relays to N

Attacker A

```
public 1.1.1.1
two VPN tunnels
now blind in-path
```

VPN server conntrack + NAT entries

ct state

A first VPN tunnel

```
orig 1.1.1.1:40001 -> 2.2.2.2:1194
reply 2.2.2.2:1194 -> 1.1.1.1:40001
action: ordinary tunnel for A
```

ESTABLISHED

PORT SHADOW NAT ROW

```
inside orig 10.0.0.2:1194 -> 3.3.3.3:53144
reply match 3.3.3.3:53144 -> 2.2.2.2:1194
NAT action: dst 2.2.2.2:1194 -> 10.0.0.2:1194
```

ESTABLISHED_REPLY

A second VPN tunnel

```
orig 1.1.1.1:53144 -> 2.2.2.2:1194
reply 2.2.2.2:1194 -> 1.1.1.1:53144
action: relay target handshake as A
```

ESTABLISHED

Shared-state confusion

Reply tuple matches handshake
N translates dst to 10.0.0.2:1194
VPN process is bypassed first
A becomes in-path via relay
Intermediary: conntrack table

DNS cache poisoning as an example of Shared Intermediary State Confusion...



DNS poisoning: attacks and defenses

Four recurring failure modes in DNS cache poisoning

TXID: 16 bits since 1983 [1]. Match the question and endpoints before applying cache trust rules (2009 [4]).

Time advances within each row	Attack	Defense	→ Mitigation / response	--- Later bypass of an earlier defense	
More chances Concurrent queries	2002 [3] Sacramento's birthday attack Many identical questions. Many IDs can match.	2002 advisory [3] Query coalescing → One upstream query per question / server.	2025 [3] RebirthDay → ECS splits aggregation. Replies reopen matching.	2025/26 fixes + guidance [3] ECS checks + caps → Check reply subnet. Restore safe aggregation.	
Match secrets DNS IDs + UDP ports	1997–2007 [4] TXID prediction Sequential / weak IDs expose the next query. ↓ Random IDs / PRNG fixes	2008 [5] Kaminsky → Fresh names evade TTL. Forge zone delegation. ↓ Random ports (July 2008)	2020/21 + 2024 [6,7] SAD DNS + TuDoor → ICMP / malformed replies reveal the query's port. ↓ Kernel / resolver patches	2025 disclosure / 2026 paper [8] BIND PRNG attack Predict both the TXID and the source port. ↓ Crypto PRNG (Oct 2025)	
Fragment replacement Reassembly below DNS	2012 preprint / 2013 paper [9] Herzberg & Shulman Replace a later fragment. Keep real TXID + port.	2010s hardening [9] IPID hardening → Separate / randomize IDs. RFC 7739 (2016).	2019 [10] Palmer & Somers → Shared Linux counters leak IPv4 / IPv6 IDs.	2020 / 2025 [11] Avoid fragmentation → Small UDP, then TCP. Drop DNS fragments.	
Cache admission Authority + relevance	1993–97 [2] Extra-record injection Schuba / Kashpureff: stuff unrelated records.	1990s onward [2] Bailiwick + credibility → Restrict zone / glue scope. Rank record trust.	2020 / 2023 [12] Troubled Forwarders → MaginotDNS Wrong forwarding scope. One cache for both modes.	2020–23 fixes [12] Correct cache scope → Bind checks to query zone. Fix forwarding + CNAME.	
Other defenses	0x20 (Mar 2008) [13] Random letter case. Check exact echo.	Cookies (2016/21) [14] 64-bit cookie challenge. Needs both endpoints.	TCP fallback [4,15] 2009: spoof detection. 2025: POPS (research).	BCP38 + ACLs [16] 2000: filter spoofed IPs. Limit recursive clients.	Protected peers [17] TSIG, DoT, DoH. Authenticate each link.

DNSSEC (1997 / 2005) [18]

Authenticate signed data through a trust chain

Unsigned zones / glue and implementation flaws still need the defenses above.

[1] RFC 883 [2] Vixie 1995 / RFC 2181 [3] Sacramento 2002 / RebirthDay 2025 [4] RFC 9414 / 5452 [5] CERT 800113 [6] Man 2020/21 [7] TuDoor 2024 [8] Ben-Simhon / Klein 2026 [9] Herzberg / Shulman, RFC 7739 [10] Palmer / Somers 2019 [11] RFC 9715 [12] Zheng 2020 / MaginotDNS 2023 [13–18] IETF RFCs / POPS 2025. Full links in companion source notes.

DNS poisoning: attacks and defenses

Four recurring failure modes in DNS cache poisoning

TXID: 16 bits since 1983 [1]. Match the question and endpoints before applying cache trust rules (2009 [4]).

Time advances within each row	Attack	Defense	Mitigation / response	Later bypass of an earlier defense	
More chances Concurrent queries	2002 [21] Sac BirthDay Many IDs can match.	2002 advisory [3] Stream coalescing Poststream query per on / server.	2025 [3] RebirthDay ECS splits aggregation. Replies reopen matching.	2025/26 fixes + guidance [3] ECS checks + caps Check reply subnet. Restore safe aggregation.	
Match secrets DNS IDs + UDP ports	1997–2007 [4] TXID prediction Sequential / weak IDs expose the next query. ↓ Random IDs / PRNG fixes	2008 [5] Kaminsky Fresh names evade TTL. Forge zone delegation. ↓ Random ports (July 2008)	2020/21 + 2024 [6,7] SAD DNS + TuDoor ICMP / malformed replies reveal the query's port. ↓ Kernel / resolver patches	2025 disclosure / 2026 paper [8] BIND PRNG attack Predict both the TXID and the source port. ↓ Crypto PRNG (Oct 2025)	
Fragment replacement Reassembly below DNS	2012 preprint / 2013 paper [9] Herzberg & Shulman Replace a later fragment. Keep real TXID + port.	2010s hardening [9] IPID hardening Separate / randomize IDs. RFC 7739 (2016).	2019 [10] Palmer & Somers Shared Linux counters leak IPv4 / IPv6 IDs.	2020 / 2025 [11] Avoid fragmentation Small UDP, then TCP. Drop DNS fragments.	
Cache admission Authority + relevance	1993–97 [2] Extra-record injection Schuba / Kashpureff: stuff unrelated records.	1990s onward [2] Bailiwick + credibility Restrict zone / glue scope. Rank record trust.	2020 / 2023 [12] Troubled Forwarders MaginotDNS Wrong forwarding scope. One cache for both modes.	2020–23 fixes [12] Correct cache scope Bind checks to query zone. Fix forwarding + CNAME.	
Other defenses	0x20 (Mar 2008) [13] Random letter case. Check exact echo.	Cookies (2016/21) [14] 64-bit cookie challenge. Needs both endpoints.	TCP fallback [4,15] 2009: spoof detection. 2025: POPS (research).	BCP38 + ACLs [16] 2000: filter spoofed IPs. Limit recursive clients.	Protected peers [17] TSIG, DoT, DoH. Authenticate each link.

DNSSEC (1997 / 2005) [18] **Authenticate signed data through a trust chain**
Unsigned zones / glue and implementation flaws still need the defenses above.

[1] RFC 883 [2] Vixie 1995 / RFC 2181 [3] Sacramento 2002 / RebirthDay 2025 [4] RFC 9414 / 5452 [5] CERT 800113 [6] Man 2020/21 [7] TuDoor 2024 [8] Ben-Simhon / Klein 2026 [9] Herzberg / Shulman, RFC 7739 [10] Palmer / Somers 2019 [11] RFC 9715 [12] Zheng 2020 / MaginotDNS 2023 [13–18] IETF RFCs / POPS 2025. Full links in companion source notes.

DNS poisoning: attacks and defenses

Four recurring failure modes in DNS cache poisoning

TXID: 16 bits since 1983 [1]. Match the question and endpoints before applying cache trust rules (2009 [4]).

Time advances within each row	Attack	Defense	Mitigation / response	Later bypass of an earlier defense	
More chances Concurrent queries	2002 [3] Sacramento's birthday attack Many identical questions. Many IDs can match.	2002 advisory [3] Query coalescing → One upstream query per question / server.	2025 [3] RebirthDay → ECS splits aggregation. Replies reopen matching.	2025/26 fixes + guidance [3] ECS checks + caps → Check reply subnet. Restore safe aggregation.	
Match secrets DNS IDs + UDP ports	1997–2007 [4] TXID prediction Sequential / weak IDs expose the next query. ↓ Random IDs / PRNG fixes	2000 [5] Ka → Fre Fo ↓ Random ports (July 2008)	2020/21 + 2024 [6,7] DNS + TuDoor P / malformed replies all the query's port. ↓ Kernel / resolver patches	2025 disclosure / 2026 paper [8] BIND PRNG attack Predict both the TXID and the source port. ↓ Crypto PRNG (Oct 2025)	
Fragment replacement Reassembly below DNS	2012 preprint / 2013 paper [9] Herzberg & Shulman Replace a later fragment. Keep real TXID + port.	2010s hardening [9] IPID hardening → Separate / randomize IDs. RFC 7739 (2016).	2019 [10] Palmer & Somers → Shared Linux counters leak IPv4 / IPv6 IDs.	2020 / 2025 [11] Avoid fragmentation → Small UDP, then TCP. Drop DNS fragments.	
Cache admission Authority + relevance	1993–97 [2] Extra-record injection Schuba / Kashpureff: stuff unrelated records.	1990s onward [2] Bailiwick + credibility → Restrict zone / glue scope. Rank record trust.	2020 / 2023 [12] Troubled Forwarders → MaginotDNS Wrong forwarding scope. One cache for both modes.	2020–23 fixes [12] Correct cache scope → Bind checks to query zone. Fix forwarding + CNAME.	
Other defenses	0x20 (Mar 2008) [13] Random letter case. Check exact echo.	Cookies (2016/21) [14] 64-bit cookie challenge. Needs both endpoints.	TCP fallback [4,15] 2009: spoof detection. 2025: POPS (research).	BCP38 + ACLs [16] 2000: filter spoofed IPs. Limit recursive clients.	Protected peers [17] TSIG, DoT, DoH. Authenticate each link.

Future slides:

DNSSEC (1997 / 2005) [18] **Authenticate signed data through a trust chain**
Unsigned zones / glue and implementation flaws still need the defenses above.

[1] RFC 883 [2] Vixie 1995 / RFC 2181 [3] Sacramento 2002 / RebirthDay 2025 [4] RFC 9414 / 5452 [5] CERT 800113 [6] Mar 2020/21 [7] TuDoor 2024 [8] Ben-Simhon / Klein 2026 [9] Herzberg / Shulman, RFC 7739 [10] Palmer / Somers 2019 [11] RFC 9715 [12] Zheng 2020 / MaginotDNS 2023 [13–18] IETF RFCs / POPS 2025. Full links in companion source notes.

DNS poisoning: references

[1] Original DNS identifier

RFC 883 (1983), message-header ID field
RFC 1035 (1987), §4.1.1

[2] Early poisoning and record trust

Paul Vixie, DNS and BIND Security Issues (1995), §§5.1–5.6
RFC 2181 (1997), §5.4.1
RFC 3833 threat analysis and Schuba thesis reference
RebirthDay, §2.2

[3] Birthday attacks and ECS revival

Sacramento / CAIS advisory, 19 November 2002
Li et al., RebirthDay Attack, CCS 2025
Unbound CVE-2025-5994
BIND ECS CVE-2025-40776
ECS aggregation repair proposal (Internet-Draft, 2026)

[4] Predictable IDs and consolidated hardening

RFC 9414, §4.6, DNS ID history
ISC CVE-2007-2926 advisory (24 July 2007), Amit Klein
RFC 5452 (January 2009), §§5, 9.1–9.3

[5] Kaminsky and the 2008 rollout

CERT/CC VU#800113, public 8 July 2008
Man et al., CCS 2020, background

[6] SAD DNS and follow-up

Man et al., DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels, CCS 2020
Man, Zhou, Qian. DNS Cache Poisoning Attack: Resurrections with Side Channels (CCS 2021), research repository
NLnet Labs mitigation discussion, 2020, updated 2021

[7] TuDoor

Li et al., TuDoor Attack: Systematically Exploring and Exploiting Logic Vulnerabilities in DNS Response Pre-processing with Malformed Packets, IEEE S&P 2024
TuDoor project and vulnerability references

[8] BIND PRNG prediction

Ben-Simhon and Klein, DNS Cache Poisoning Like it's 2006, USENIX Security 2026
Ben-Simhon and Klein (USENIX Security 2026), full paper
ISC CVE-2025-40780, 22 October 2025

[9] Fragment poisoning and ID hardening

Herzberg and Shulman, Fragmentation Considered Poisonous, preprint 2012
IEEE CNS 2013 publication record
RFC 7739, February 2016

[10] Palmer and Somers

DEF CON 27 (2019), First-Try DNS Cache Poisoning with IPv4 and IPv6 Fragmentation, conference recording
Palmer and Somers, DEF CON 27 (2019), presentation slides
Daymude et al., A Taxonomy and Comparative Analysis of IPv4 Identifier Selection Correctness, Security, and Performance, §§3.1.5, 3.1.6

[11] Avoiding fragmentation

RFC 9715, January 2025, §§3, 7 and Appendix A
DNS Flag Day 2020

[12] Forwarder and bailiwick failures

Zheng et al., Poison Over Troubled Forwarders, USENIX Security 2020
Li et al., The Maginot Line: Attacking the Boundary of DNS Caching Protection, USENIX Security 2023
MaginotDNS project and vulnerability references

[13] 0x20 encoding

Vixie and Dagon, March 2008 Internet-Draft
Dagon et al., Increased DNS Forgery Resistance Through 0x20-Bit Encoding, CCS 2008

[14] DNS Cookies

RFC 7873, May 2016
RFC 9018, May 2021

[15] POPS

Afek, Berger, Bremner-Barr, POPS: From History to Mitigation of DNS Cache Poisoning Attacks, USENIX Security 2025
RFC 5452 (2009), §9.3: TCP fallback on suspected forgery

[16] Network and access controls

RFC 2827 / BCP38, May 2000
Sacramento / CAIS (2002), recursion-access recommendations

[17] Authenticated peers and transport

TSIG RFC 2845 (2000)
RFC 8945 (2020)
DoT RFC 7858 (2016)
DoH RFC 8484 (2018)

[18] DNSSEC

Original RFC 2065, January 1997
RFC 4033, March 2005
RFC 9715 (2025), §7.4: unsigned-delegation limitations

IPID → IP header (layer 3)
“IP identifier”

“Source port” → UDP header (layer 4)

TXID → DNS request and response
(layer 7)
“Transaction identifier”



udp.stream eq 2

No.	Time	Source	Destination	Info
8	1.285718287	10.42.0.14	10.42.0.1	Standard query 0x2b9f A hlx.meituan.com
73	4.510742303	10.42.0.1	10.42.0.14	Standard query response 0x2b9f A hlx.meituan

Frame 8: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface wlx6c5ab00ee69e, id 0
 Ethernet II, Src: d2:4c:cf:57:fe:a7 (d2:4c:cf:57:fe:a7), Dst: TP-Link_0e:e6:9e (6c:5a:b0:0e:e6:9e)
 Internet Protocol Version 4, Src: 10.42.0.14, Dst: 10.42.0.1
 User Datagram Protocol, Src Port: 63826, Dst Port: 53

Source Port: 63826

Destination Port: 53

0000	6c 5a b0 0e e6 9e d2 4c	cf 57 fe a7 08 00 45 00	lZ.....L .W....E.
0010	00 3d 63 56 40 00 40 11	c2 f7 0a 2a 00 0e 0a 2a	..=cV@.@.*...*
0020	00 01 f9 52 00 35 00 29	9d a6 2b 9f 01 00 00 01	...R.5.) ..+.....
0030	00 00 00 00 00 00 03 68	6c 78 07 6d 65 69 74 75	h lx.meitu
0040	61 6e 03 63 6f 6d 00 00	01 00 01	an.com.. ...

meituan.pcap

FileEditViewGoCaptureAnalyzeStatisticsTelephonyWirelessToolsHelp

udp.stream eq 2

Destination	Info	Protocol	Length
10.42.0.1	Standard query 0x2b9f A hlx.meituan.com	DNS	75
10.42.0.14	Standard query response 0x2b9f A hlx.meituan.com A 101.236.9.105 A...	DNS	107

↳ User Datagram Protocol, Src Port: 63826, Dst Port: 53

↳ Domain Name System (query)

Transaction ID: 0x2b9f

↳ Flags: 0x0100 Standard query

Questions: 1

Answer RRs: 0

Authority RRs: 0

00006c 5a b0 0e e6 9e d2 4c cf 57 fe a7 08 00 45 00lZ....L.W....E.

001000 3d 63 56 40 00 40 11 c2 f7 0a 2a 00 0e 0a 2a.=cV@.@...*...*

002000 01 f9 52 00 35 00 29 9d a6 2b 9f 01 00 00 01...R.5.)..+.....

003000 00 00 00 00 00 00 03 68 6c 78 07 6d 65 69 74 75.....h lx.meitu

004061 6e 03 63 6f 6d 00 00 01 00 01an.com... ..

Identification of transaction (dns.id), 2 bytes

Packets: 45595 · Displayed: 2 (0.0%)

Profile: Default



udp.stream eq 2

Destination	Info	Protocol	Length
10.42.0.1	Standard query 0x2b9f A hlx.meituan.com	DNS	75
10.42.0.14	Standard query response 0x2b9f A hlx.meituan.com A 101.236.9.105 A...	DNS	107

Ethernet II, Src: TP-Link_0e:e6:9e (6c:5a:b0:0e:e6:9e), Dst: d2:4c:cf:57:fe:a7 (d2:4c:cf:57:fe:a7)
 Internet Protocol Version 4, Src: 10.42.0.1, Dst: 10.42.0.14
 User Datagram Protocol, Src Port: 53, Dst Port: 63826

Source Port: 53

Destination Port: 63826

Length: 73

0020	00 0e 00 35 f9 52 00 49 a3 4c 2b 9f 81 80 00 01	...5.R.I.L+....
0030	00 02 00 00 00 00 03 68 6c 78 07 6d 65 69 74 75	h lx.meitu
0040	61 6e 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 00	an.com.. ..
0050	01 00 00 00 78 00 04 65 ec 09 69 c0 0c 00 01 00	...x.e .i....
0060	01 00 00 00 78 00 04 65 ec 41 22	...x.e .A"



udp.stream eq 2

Destination	Info	Protocol	Length
10.42.0.1	Standard query 0x2b9f A hlx.meituan.com	DNS	75
10.42.0.14	Standard query response 0x2b9f A hlx.meituan.com A 101.236.9.105 A...	DNS	107

Frame 73: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface wlx6c5ab00ee69e, interface
 Ethernet II, Src: TP-Link_0e:e6:9e (6c:5a:b0:0e:e6:9e), Dst: d2:4c:cf:57:fe:a7 (d2:4c:cf:57:fe:a7)
 Internet Protocol Version 4, Src: 10.42.0.1, Dst: 10.42.0.14
 User Datagram Protocol, Src Port: 53, Dst Port: 63826
 Domain Name System (response)

Transaction ID: 0x2b9f

0020	00 0e 00 35 f9 52 00 49	a3 4c 2b 9f 81 80 00 01	...5.R.I.L+....
0030	00 02 00 00 00 00 03 68	6c 78 07 6d 65 69 74 75	h lx.meitu
0040	61 6e 03 63 6f 6d 00 00	01 00 01 c0 0c 00 01 00	an.com.. ..
0050	01 00 00 00 78 00 04 65	ec 09 69 c0 0c 00 01 00	...x.e..i....
0060	01 00 00 00 78 00 04 65	ec 41 22	...x.e.A"

```
jedi@tortuga:~$ dig @8.8.8.8 ns meituan.com | head -n 21
```

```
; <<>> DiG 9.18.39-0ubuntu0.22.04.2-Ubuntu <<>> @8.8.8.8 ns meituan.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59326
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags::; udp: 512
;; QUESTION SECTION:
;meituan.com.                IN      NS

;; ANSWER SECTION:
meituan.com.                21600   IN      NS      ns3.dnsv5.com.
meituan.com.                21600   IN      NS      ns4.dnsv5.com.

;; Query time: 1029 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Mon Feb 02 09:58:19 MST 2026
;; MSG SIZE rcvd: 82
jedi@tortuga:~$
```

```
jedi@tortuga: ~  
jedi@tortuga:~$ dig @8.8.8.8 ns hlx.meituan.com | head -n 21  
  
; <<>> DiG 9.18.39-0ubuntu0.22.04.2-Ubuntu <<>> @8.8.8.8 ns hlx.meituan.com  
; (1 server found)  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30382  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 512  
;; QUESTION SECTION:  
;hlx.meituan.com.                IN      NS  
  
;; ANSWER SECTION:  
hlx.meituan.com.                180     IN      CNAME   bi-hreport.vip.meituan.com.  
  
;; AUTHORITY SECTION:  
meituan.com.                    180     IN      SOA     ns3.dnsv5.com. enterprise3dnsadm  
in.dnspod.com. 1770016267 3600 180 1209600 180  
  
;; Query time: 45 msec  
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)  
jedi@tortuga:~$
```

```
jedi@halfdome:~$ dig www.wikipedia.org A
```

```
; <<>> DiG 9.20.24-lubuntu0.3-Ubuntu <<>> www.wikipedia.org A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12073
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.wikipedia.org.                IN      A

;; ANSWER SECTION:
www.wikipedia.org.      169     IN      CNAME   dyna.wikimedia.org.
dyna.wikimedia.org.    169     IN      A       198.35.26.224

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Sat Oct 10 09:45:08 MST 2026
;; MSG SIZE rcvd: 91
```

```
jedi@halfdome:~$ █
```

What stops me from saying `www.wikipedia.org` is
a common name for `www.breakpointingbad.com`?

The primary goal of bailiwick checking is to prevent a malicious nameserver authoritative for example.com from providing (and a resolver from caching) records for other-example.net, or worse, for TLDs or the root zone.

<https://datatracker.ietf.org/doc/draft-qi-dnsop-enhanced-bailiwick/#:~:text=The%20primary%20goal%20of%20bailiwick,TLDs%20or%20the%20root%20zone.>

Introduced in the mid 90's

Birthday Attacks

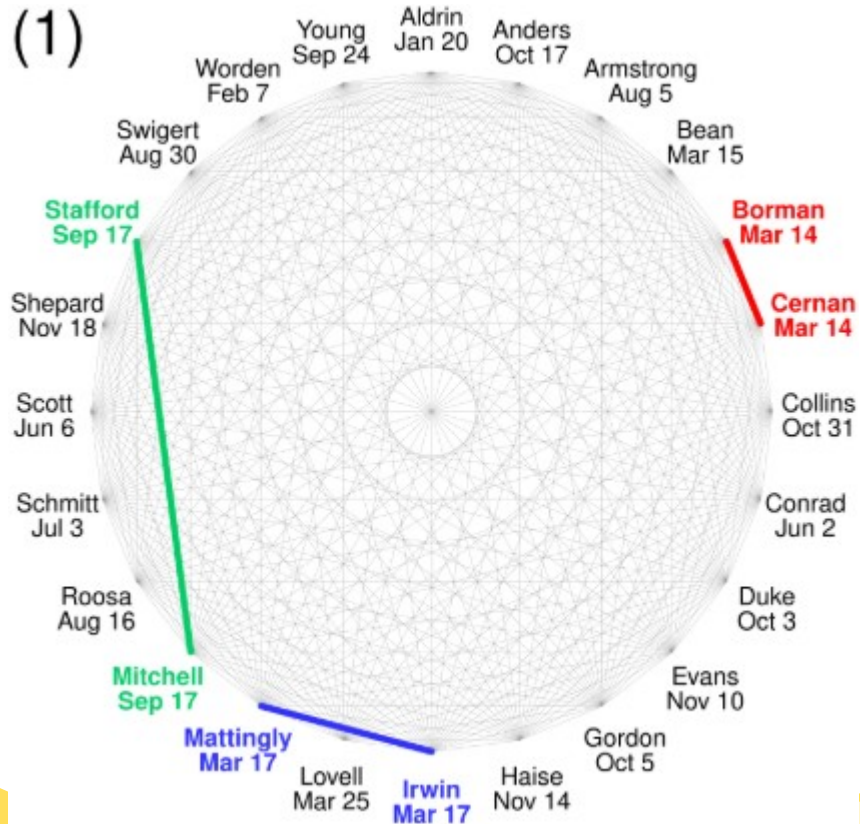
- <https://www.kb.cert.org/vuls/id/457>
- 2002

If the attacker has to guess...	...and is limited to the following number of open requests...	...it will take the following number of packets to achieve a 50% success rate (includes both requests and responses)
TID only (16bits)	1	32.7 k (2^{15})
TID only (16bits)	4	10.4 k
TID only (16bits)	200	427
TID only (16bits)	unlimited	426
TID and port (32 bits)	1	2.1 billion (2^{31})
TID and port (32 bits)	4	683 million
TID and port (32 bits)	200	15 million
TID and port (32 bits)	unlimited	109 k

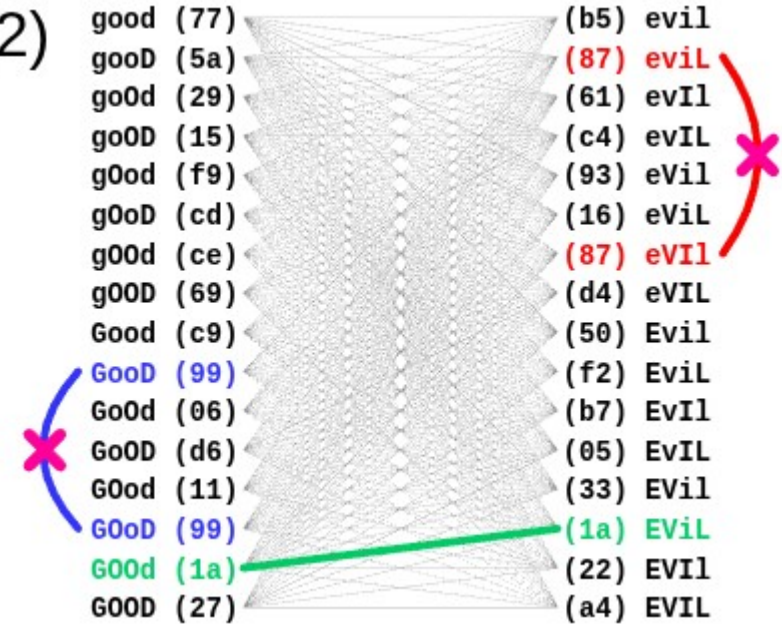
Table 1: Number of packets required to reach 50% success probability for various numbers of open queries

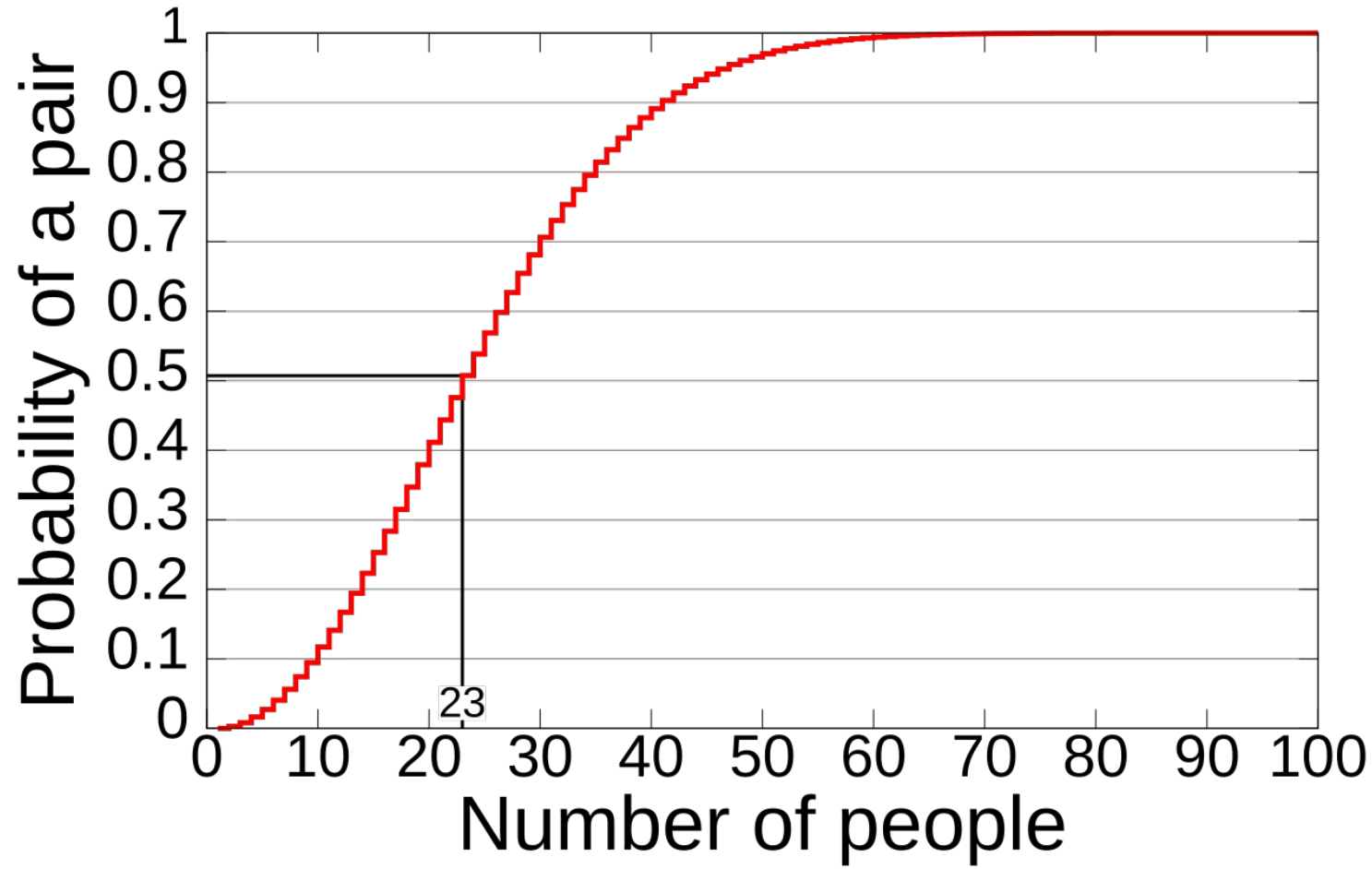
https://en.wikipedia.org/wiki/Birthday_attack

(1)



(2)





Solution to the specific birthday attack on DNS above... Don't allow multiple queries for the same domain at the same time.

Dan Kaminsky's attack (2008)

- <https://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf>

DNS is distributed

- Three possible answers to any question
 - “Here’s your answer”
 - “Go away”
 - “I don’t know, ask that guy over there”
 - This is delegation. You start with a request, and then get bounced around all over the place.
 - 13 root servers: “www.foo.com? I don’t know, go ask the com server, it’s at 1.2.3.4”
 - Com server: “www.foo.com? I don’t know, go ask the foo.com server, it’s at 2.3.4.5”
 - Foo.com server: “www.foo.com? Yeah, that’s at 3.4.5.6.”

```
jedi@tortuga: ~  
jedi@tortuga:~$ dig www.breakpointingbad.com  
  
; <<>> DiG 9.18.39-0ubuntu0.22.04.2-Ubuntu <<>> www.breakpointingbad.com  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33334  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 65494  
;; QUESTION SECTION:  
www.breakpointingbad.com.      IN      A  
  
;; ANSWER SECTION:  
www.breakpointingbad.com. 1799 IN      A      149.28.240.117  
  
;; Query time: 58 msec  
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)  
;; WHEN: Mon Feb 02 10:21:40 MST 2026  
;; MSG SIZE rcvd: 69  
  
jedi@tortuga:~$
```

```
jedi@halfdome:~$ dig www.wikipedia.org A
```

```
; <<>> DiG 9.20.24-lubuntu0.3-Ubuntu <<>> www.wikipedia.org A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12073
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.wikipedia.org.                IN      A

;; ANSWER SECTION:
www.wikipedia.org.      169     IN      CNAME   dyna.wikimedia.org.
dyna.wikimedia.org.    169     IN      A       198.35.26.224

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Sat Oct 10 09:45:08 MST 2026
;; MSG SIZE rcvd: 91
```

```
jedi@halfdome:~$ █
```

- If the bad guy can reply 100 times before the good guy returns, that 65536 to 1 advantage drops to 655 to 1.
 - Alas...still long odds. And when he loses, he has to wait the TTL. That could be 655 days – almost 2 years!
 - Or maybe not.

Finally, the bad guy doesn't actually need to wait to try again.

- If the bad guy asks the name server to look up www.foo.com ten times, there will only be one race with the good guy
 - The first race will be lost (most likely), and then the other nine will be suppressed by the TTL
 - No new races on this name for one more day! Here, use the answer from a while ago
 - So, can we race on other names?
- If the bad guy asks the name server to look up 1.foo.com, 2.foo.com, 3.foo.com, and so on, for ten names, there will be 10 races with the good guy
 - TTL only stops repeated races for the same name!
- Eventually, the bad guy will guess the right TXID before the good guy shows up with it
 - And now...the bad guy is the proud spoofer of ... 83.foo.com
 - So? He didn't *want* to poison 83.foo.com. He wanted www.foo.com

Bait and Switch

- Is it possible for a bad guy, who has won the race for 83.foo.com, to end up stealing [www.foo.com](#) as well?
 - He has three possible replies that can be associated with correctly guessed TXID
 - 1) “Here’s your answer for 83.foo.com – it’s 6.6.6.6”
 - 2) “I don’t know the answer for 83.foo.com.”
 - 3) “83.foo.com? I don’t know, go ask the [www.foo.com](#) server, it’s at 6.6.6.6”
 - This has to work – it’s just another delegation
 - 13 root servers: “83.foo.com? I don’t know, go ask the com server, it’s at 1.2.3.4”
 - Com server: “83.foo.com? I don’t know, go ask the foo.com server, it’s at 2.3.4.5”
 - Foo.com server: “83.foo.com? I don’t know, go ask the [www.foo.com](#) server, it’s at 6.6.6.6”

Does bailiwick checking save us?



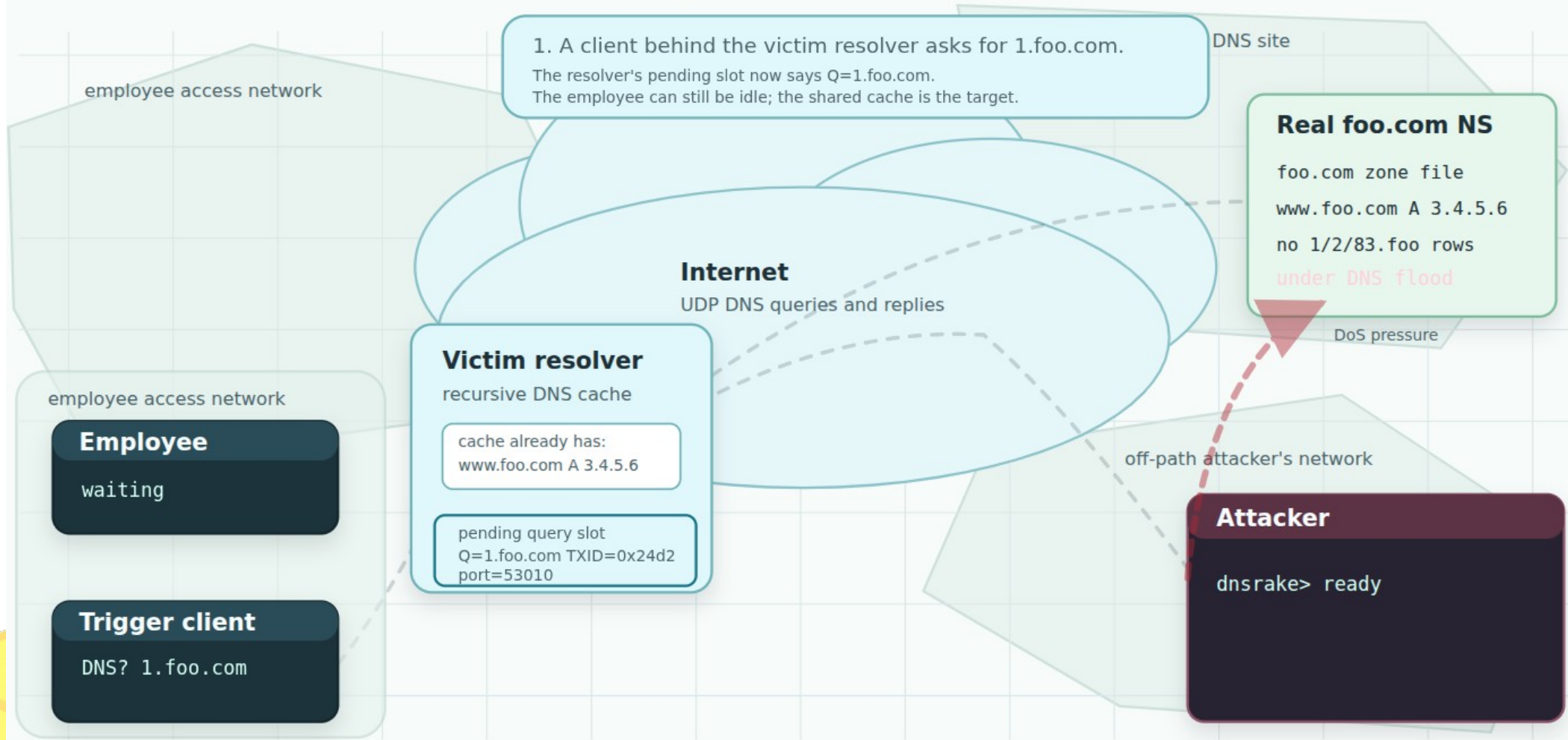
Does bailiwick checking save us?

*No! The only “authentication” is the source IP address
(attacker can spoof), source port, and TXID!*



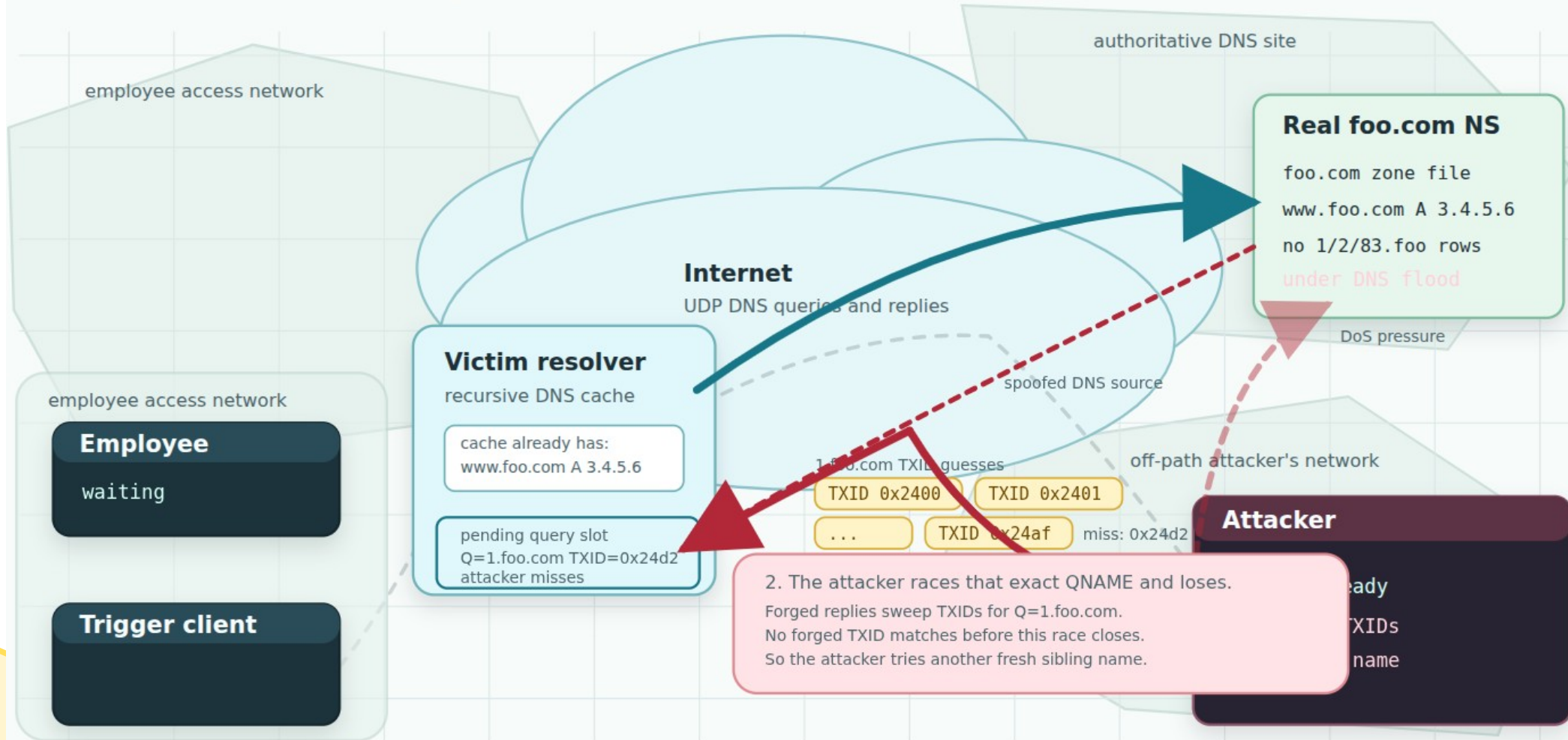
Kaminsky DNS cache poisoning

A client asks for 1.foo.com; the victim resolver opens matching state.



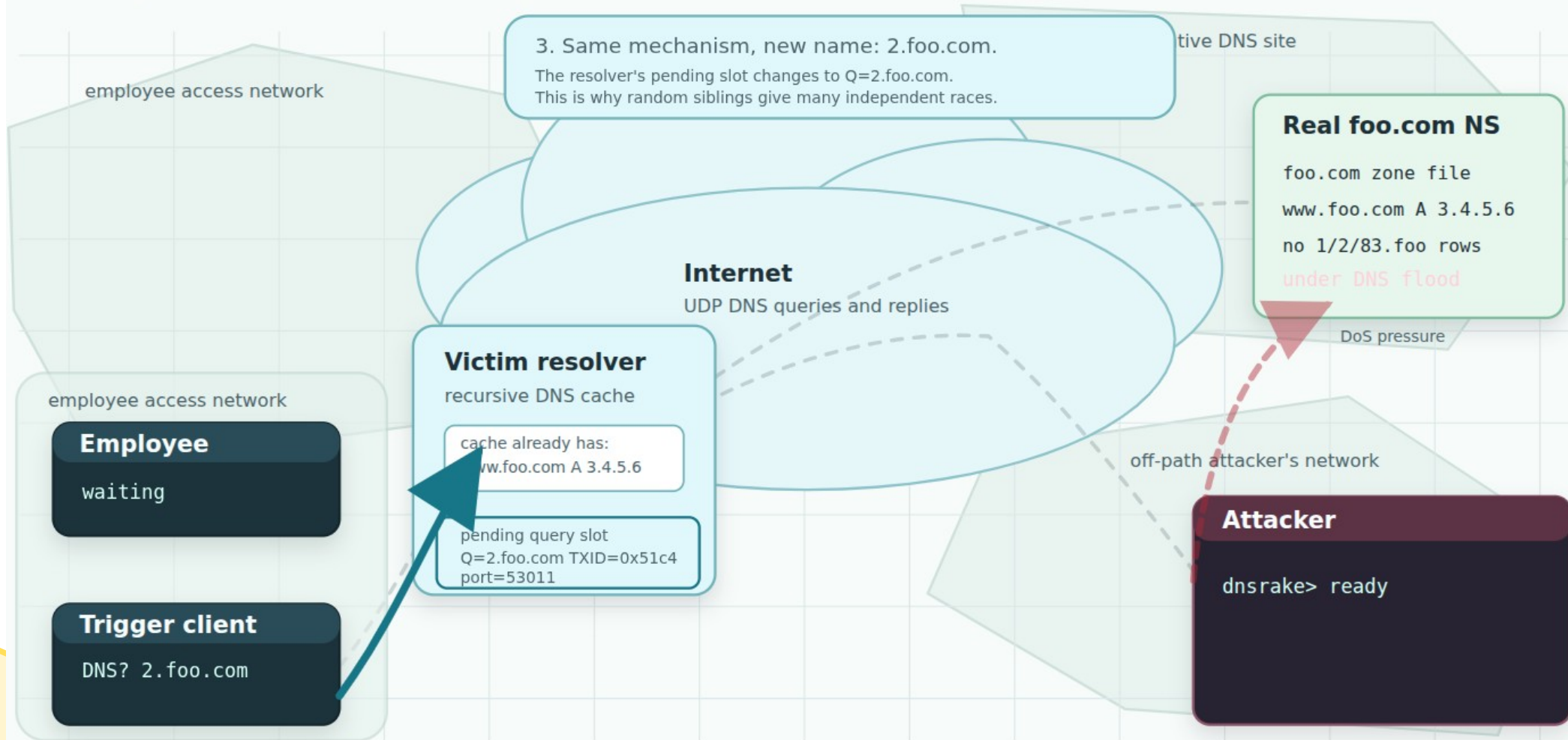
Kaminsky DNS cache poisoning

The attacker races 1.foo.com with TXID guesses and loses.



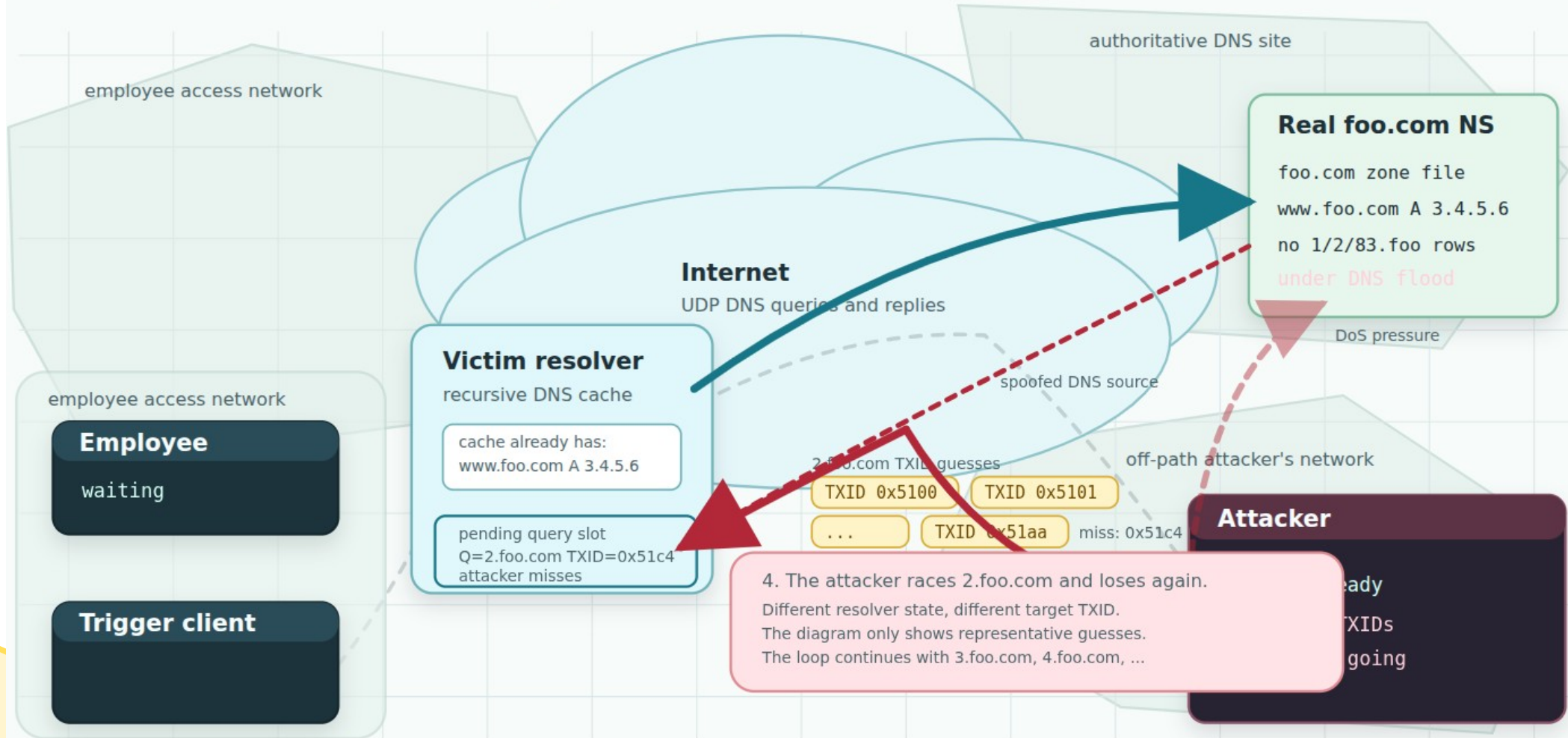
Kaminsky DNS cache poisoning

Next query: 2.foo.com; the resolver state changes with it.



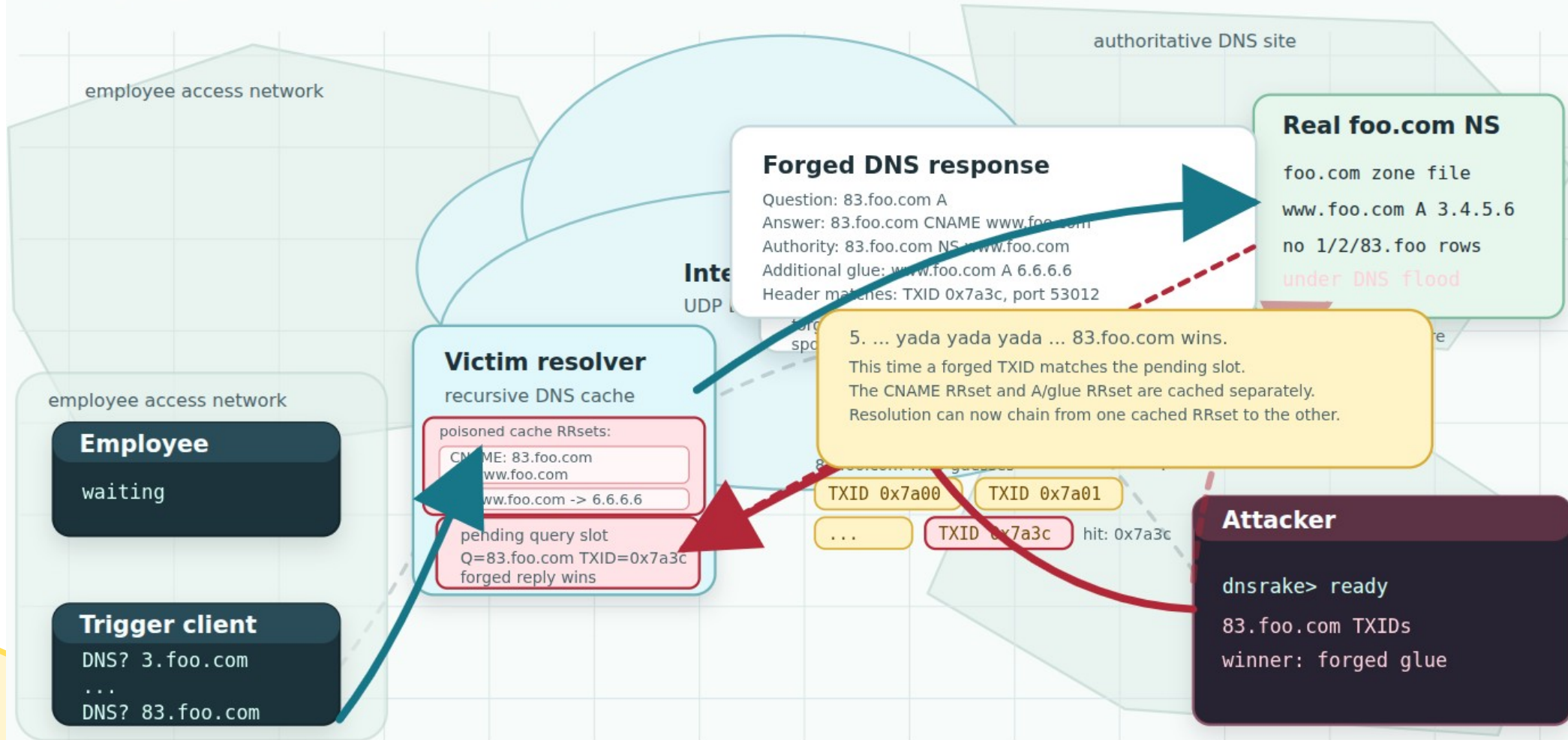
Kaminsky DNS cache poisoning

The attacker races 2.foo.com with different TXID guesses and loses.



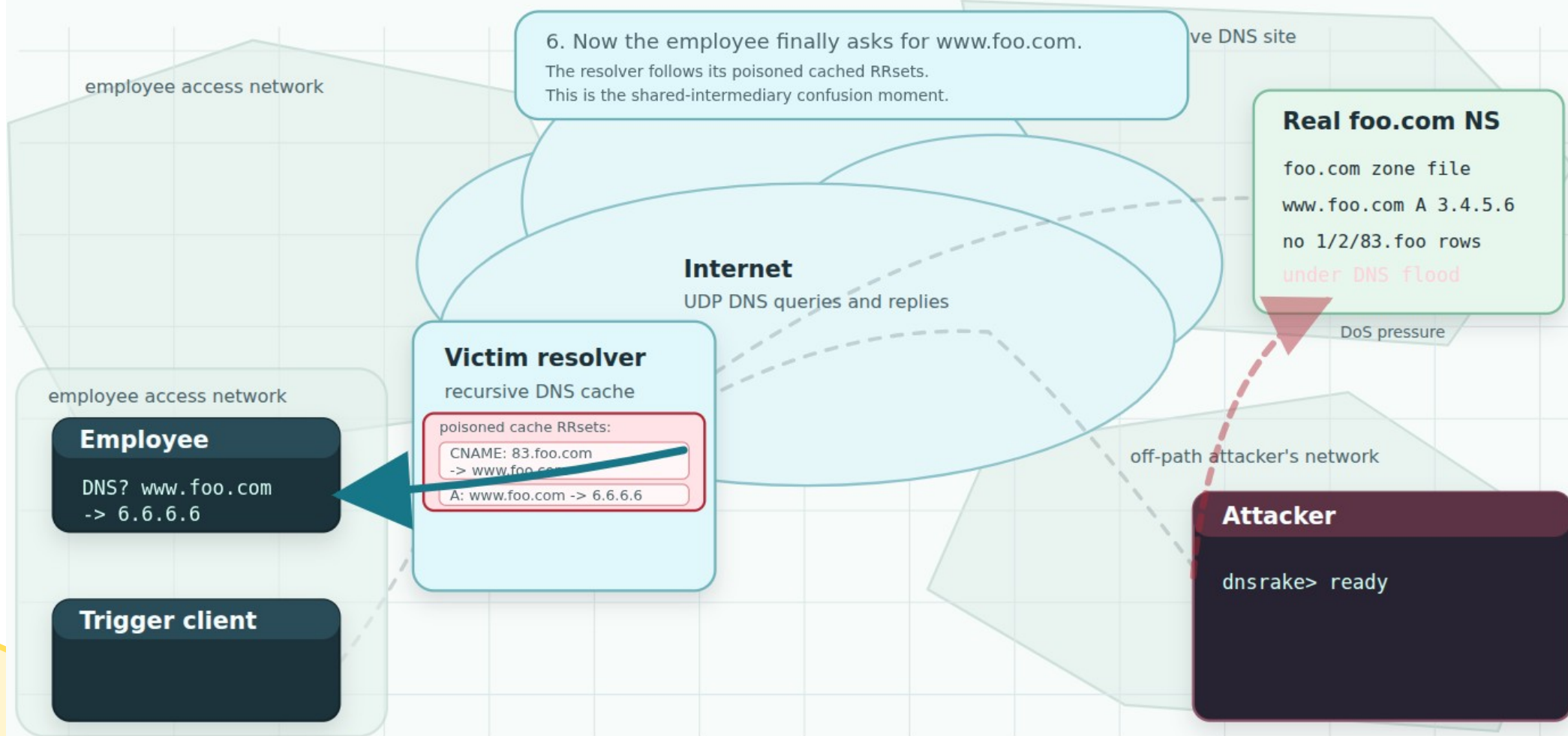
Kaminsky DNS cache poisoning

... yada yada yada ... at 83.foo.com the forged reply wins.



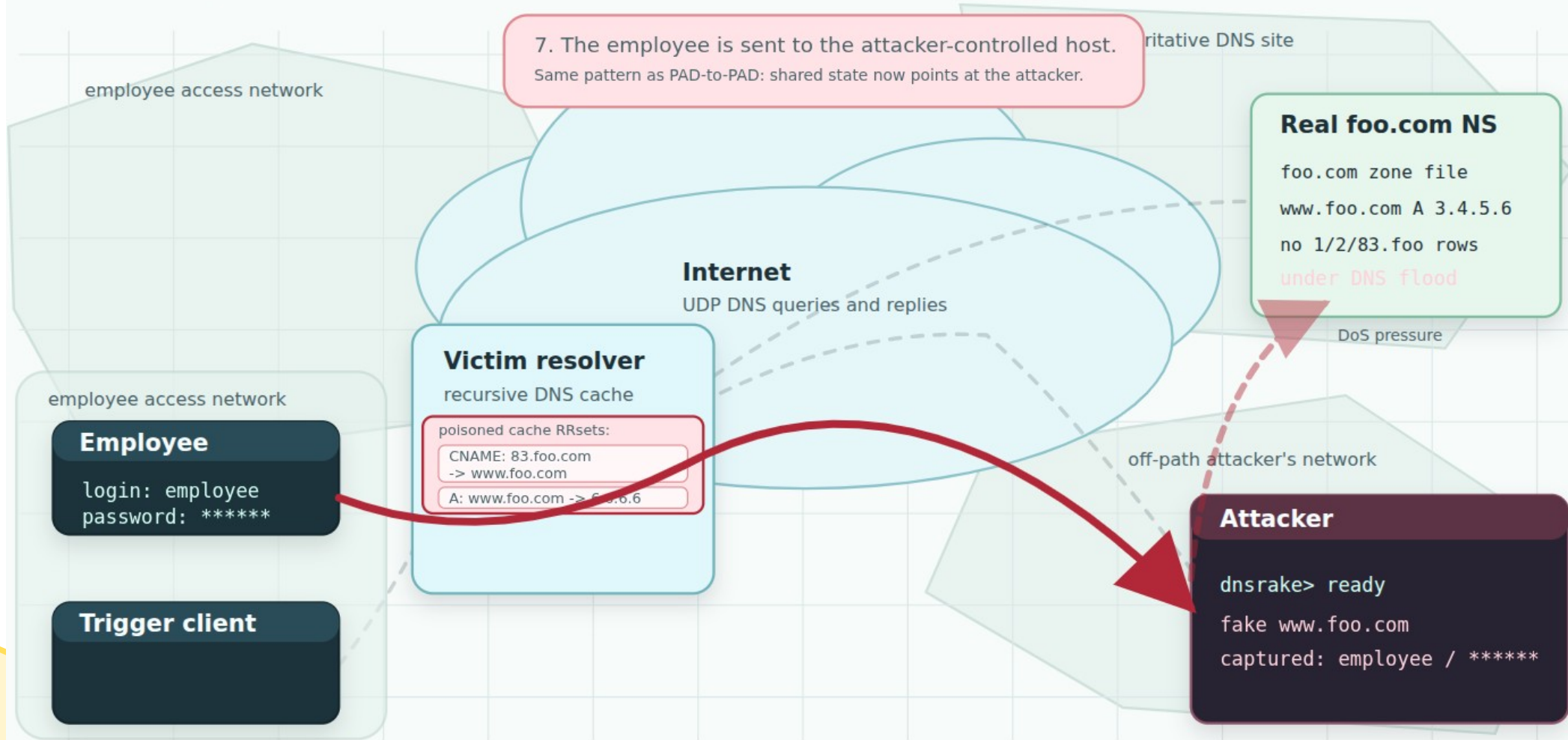
Kaminsky DNS cache poisoning

Now `www.foo.com` resolves from poisoned cache state.



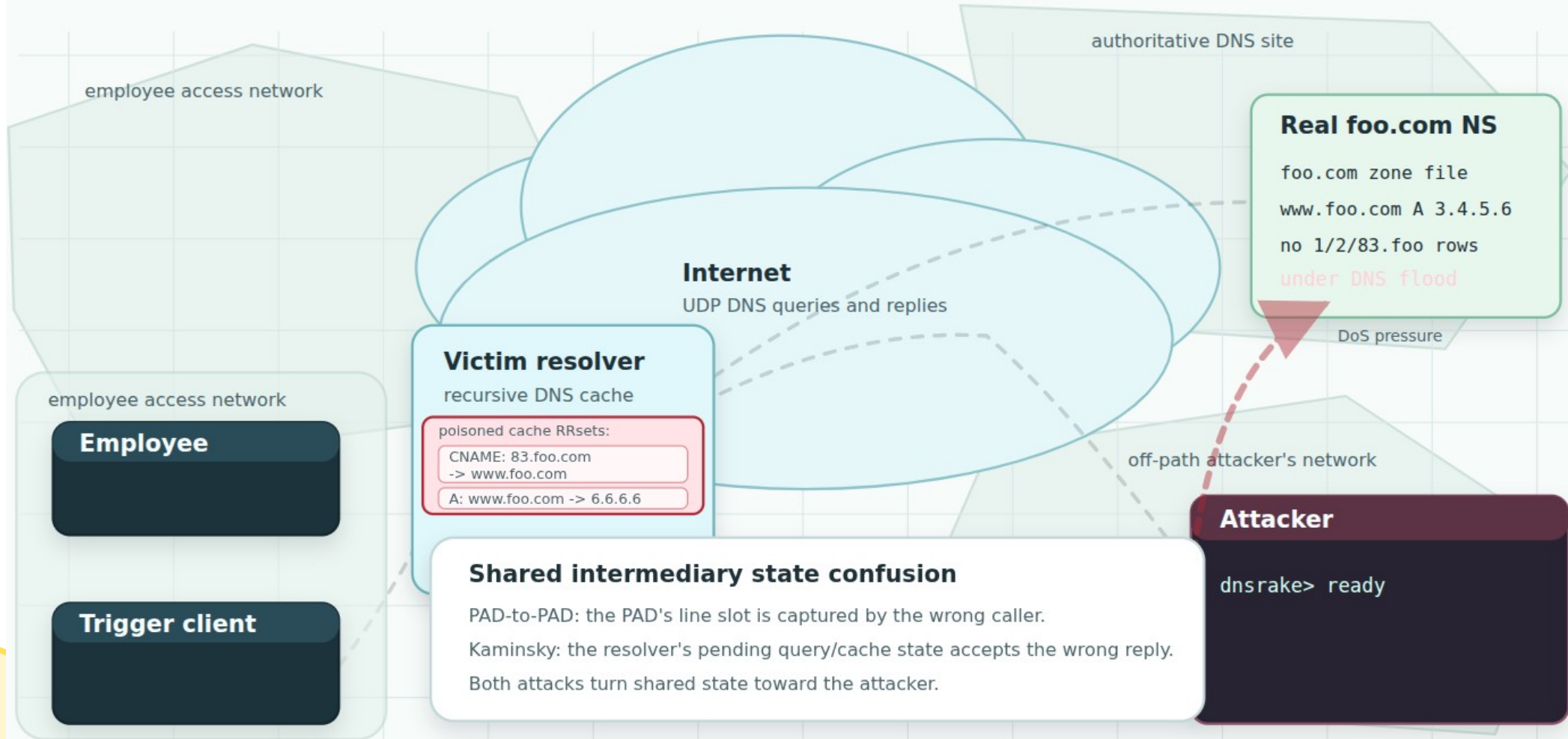
Kaminsky DNS cache poisoning

Traffic follows the poisoned answer toward the attacker-controlled host.



Kaminsky DNS cache poisoning

Same pattern: shared intermediary state binds to the attacker.



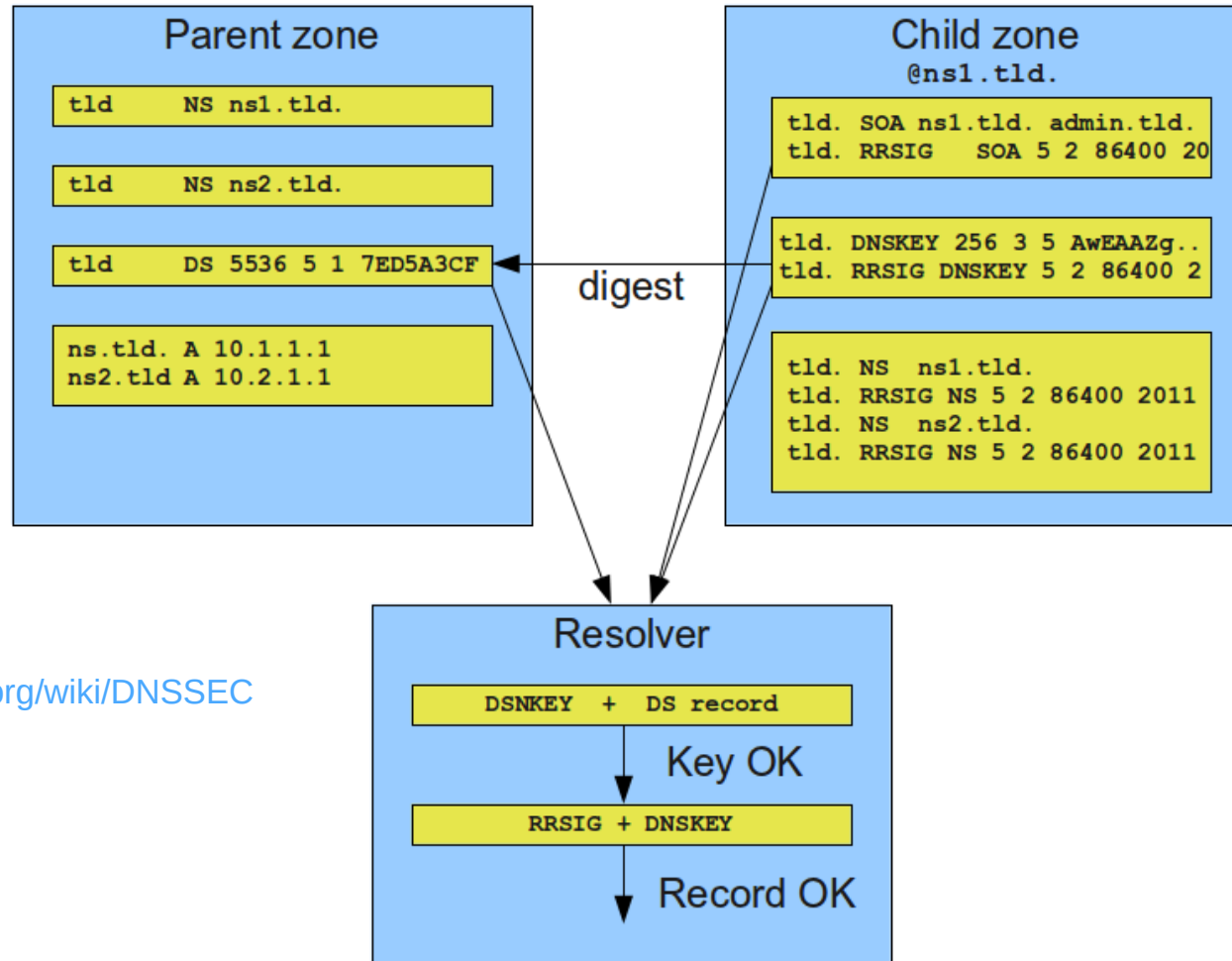
Solution to the Kaminsky attack... OSes now randomize source ports.

(Also, some other stuff, like 0x20 encoding:
BrEAkPoinTiNGBaD.COm)

Future slides: what if we didn't have to guess the TXID or source port?

A real solution would be a real form of authentication, like signatures...

DNSSEC



<https://ru.wikipedia.org/wiki/DNSSEC>

```
jedi@tortuga: ~  
jedi@tortuga:~$ dig fraunhofer.de @8.8.8.8 +dnssec | head -n 18  
  
; <<>> DiG 9.18.39-0ubuntu0.22.04.2-Ubuntu <<>> fraunhofer.de @8.8.8.8 +dnssec  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29859  
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags: do; udp: 512  
;; QUESTION SECTION:  
;fraunhofer.de.                IN      A  
  
;; ANSWER SECTION:  
fraunhofer.de.                3593    IN      A      192.102.162.236  
fraunhofer.de.                3593    IN      RRSIG   A 8 2 3600 20260227143958 202601  
28135244 57189 fraunhofer.de. SHIGBs5CC6hV+DigBXTs0wbBb2gQlnxLVhsVnJxU8Fd6xT/Pjl  
zXL1zz r2H2QdP/rgmL60ITTC/hGvfsC8QF601h5oVqSyQjwEMnKS46GXP2VHH4 4MvQAUMUABRN63t+  
pqSldpuYV00/2gjV6IDF09ne+qCj0zxVraA4e2aY 3Po=  
  
;; Query time: 20 msec  
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)  
jedi@tortuga:~$
```

More info

- <http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html>