# Exercise Session XI: DNS Security

Network Security

Matteo Scarlata

ETH Zurich

December 3rd, 2020

# Context: Domain Name System

# DNS: Origins

```
HOST : 192.5.89.50 : KENTARUS.ARPA,KENTARUS : SUN-2/120 : UNIX : TCP/TELNET,TCP/SMTP,TCP/FTP,ICMP,UDP :
HOST : 192.12.59.50 : MCCDB-ERATOSTHENES.ARPA,MCCDB-ERATOSTHENES,ERATOSTHENES : SUN-2/170 : UNIX :
TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.51 : TAUCETI.ARPA,TAUCETI : SUN-2/120 : UNIX : TCP/TELNET,TCP/SMTP,TCP/FTP,ICMP,UDP :
HOST : 192.12.59.51 : MCCDB-ACKERMANN.ARPA,MCCDB-ACKERMANN,ACKERMANN : SUN-2/50 : UNIX :
TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.12.59.52 : MCCDB-HILBERT.ARPA,MCCDB-HILBERT,HILBERT : SUN-2/50 : UNIX :
TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.12.59.53 : MCCDB-SUPPES.ARPA,MCCDB-SUPPES,SUPPES : SUN-2/50 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP
HOST : 192.12.59.54 : MCCDB-ZERMELO.ARPA,MCCDB-ZERMELO,ZERMELO : SUN-2/50 : UNIX :
TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.2.55, 192.12.12.1 : WISC-SPOOL.ARPA,WISC-SPOOL,LOOPS,NEEDLE : VAX-11/750 : UNIX :
TCP/TELNET,TCP/FTP,TCP/SMTP,TCP/ECHO,TCP/FINGER,ICMP :
HOST : 192.12.59.55 : MCCDB-TARTAGLIA.ARPA,MCCDB-TARTAGLIA,TARTAGLIA : SUN-2/50 : UNIX :
TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.12.59.56 : MCCDB-VIETE.ARPA,MCCDB-VIETE,VIETE : SUN-2/50 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.12.59.57 : MCCDB-LEGENDRE.ARPA,MCCDB-LEGENDRE,LEGENDRE : SUN-2/50 : UNIX :
TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.38.60 : SRI-GILLIGAN.ARPA,SRI-GILLIGAN,GILLIGAN,SRI-SAC0 : SUN-120 : UNIX :
TCP/TELNET,TCP/FTP,TCP/SMTP,UDP,ICMP  :
HOST : 192.5.89.61 : ZOTZ.ARPA,ZOTZ,SOATS : SUN-2/120 : UNIX : TCP/TELNET,TCP/SMTP,TCP/FTP,ICMP,UDP :
HOST : 192.5.89.62 : TZEC.ARPA,TZEC,ZAKE : SUN-2/120 : UNIX : TCP/TELNET,TCP/SMTP,TCP/FTP,ICMP,UDP :
HOST : 192.5.89.63 : XUL.ARPA,XUL,SHOOL : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.12.5.63 : SRI-MARVIN.ARPA,SRI-MARVIN,MARVIN : SUN-2/170 : UNIX : TCP/SMTP,TCP/TELNET,TCP/FTP :
HOST : 192.5.89.64 : YAXKIN.ARPA,YAXKIN,YOSHKEEN : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.65 : MOL.ARPA,MOL,MOLE : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.1.2.66, 8.4.0.12 : BBNF.ARPA,BBNF,BBN-TENEXF : DEC-2040T : TOPS20 :
TCP/FTP,TCP/TELNET,TCP/SMTP,TCP/TIME,UDP/TIME :
HOST : 192.5.89.66 : KANKIN.ARPA,KANKIN,CONKEEN : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.67 : ZIP.ARPA,ZIP,SEEP : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.68 : POP.ARPA,POP,POPE : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.69 : MUAN.ARPA,MUAN,MWAHN : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.70 : UO.ARPA,UO,WOE : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.71 : BUN.ARPA,BUN : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.72 : FUN.ARPA,FUN : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.73 : NUN.ARPA,NUN : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
HOST : 192.5.89.74 : RUN.ARPA,RUN : SUN-2/120 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,ICMP,UDP :
```

# DNS: Origins

## HOSTS.TXT (TCP/IP, DOMAIN NAME(The Initial and Temporary Domain))

See also RFC 897, RFC 881, and RFC 921.

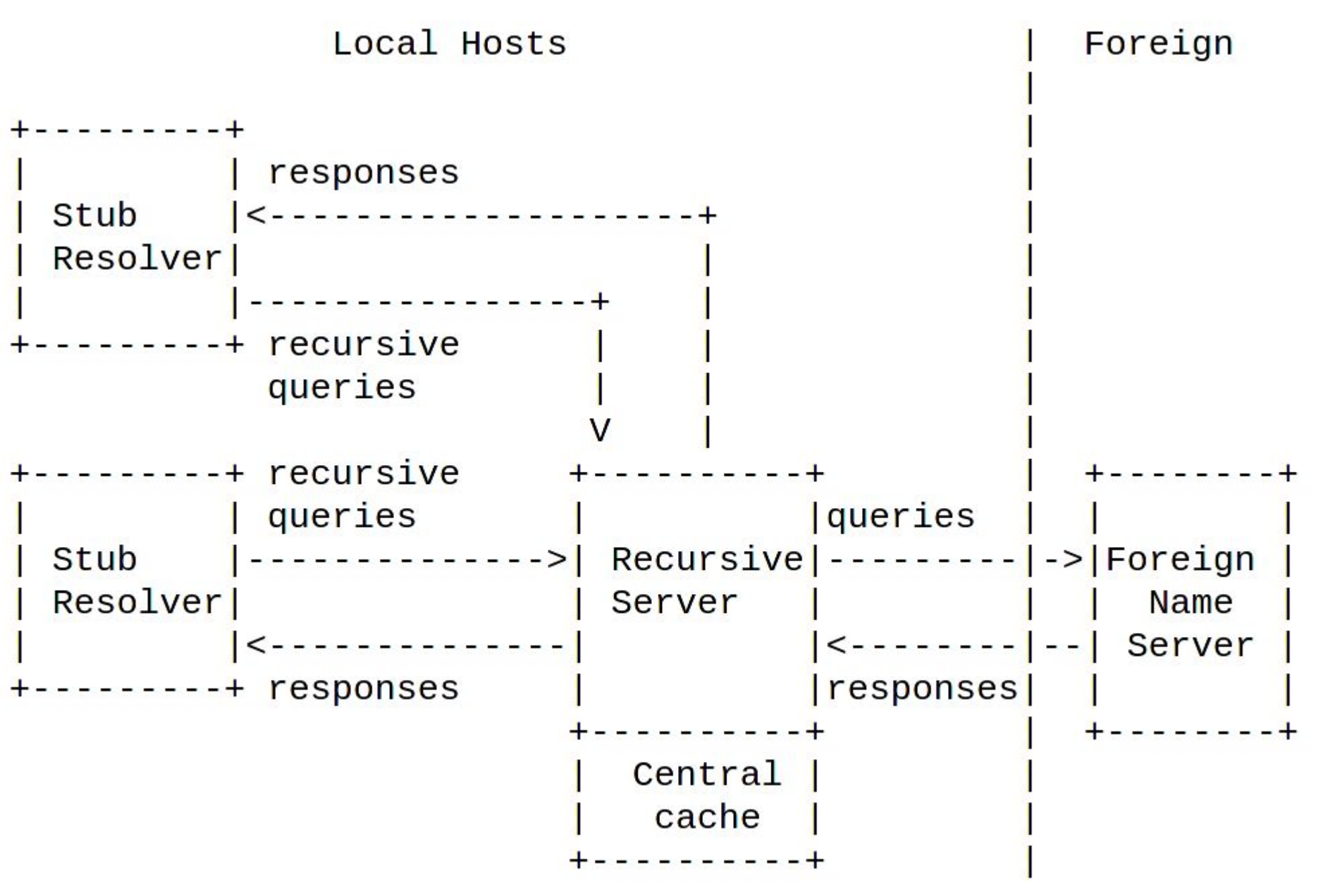
| Date | Version | File | NET | GATEWAY | HOST |
|---|---|---|---|---|---|
| 1984-03-27 | 340 | HOSTS.TXT (*1) | 135 | 69 | 732 |
| 1984-04-19 | 343 | HOSTS.TXT (*1) | 139 | 72 | 756 |
| 1984-04-27 | 344 | HOSTS.TXT (*1) | 140 | 73 | 766 |
| 1984-06-25 | 353 | HOSTS.TXT (*1) | 144 | 79 | 841 |
| 1984-07-13 | 359 | HOSTS.TXT (*1) | 147 | 82 | 877 |
| 1984-11-28 | 405 | HOSTS.TXT (*1) | 185 | 96 | 1122 |
| 1984-12-21 | 412 | HOSTS.TXT (*1) | 187 | 98 | 1138 |
| 1985-01-02 | 413 | HOSTS.TXT (*1) | 187 | 98 | 1142 |

https://github.com/ttkzw/hosts.txt

# DNS: Origins

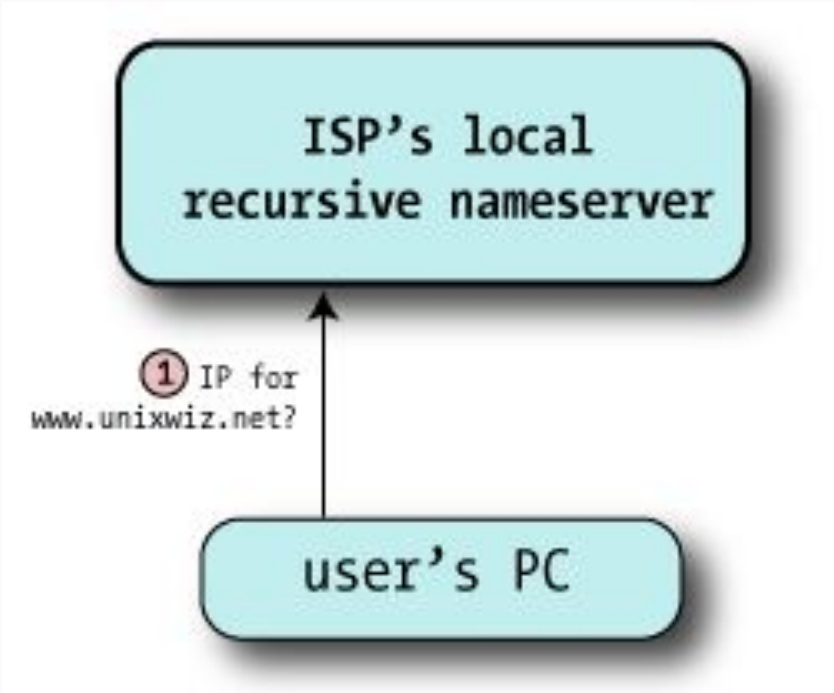
```
$ cat /etc/hosts
127.0.0.1  localhost loopback
::1        localhost
```

# DNS: Origins


Local Hosts
Foreign
responses
Stub Resolver
recursive queries
recursive queries
Stub Resolver
Recursive Server
queries
responses
Foreign Name Server
Central cache

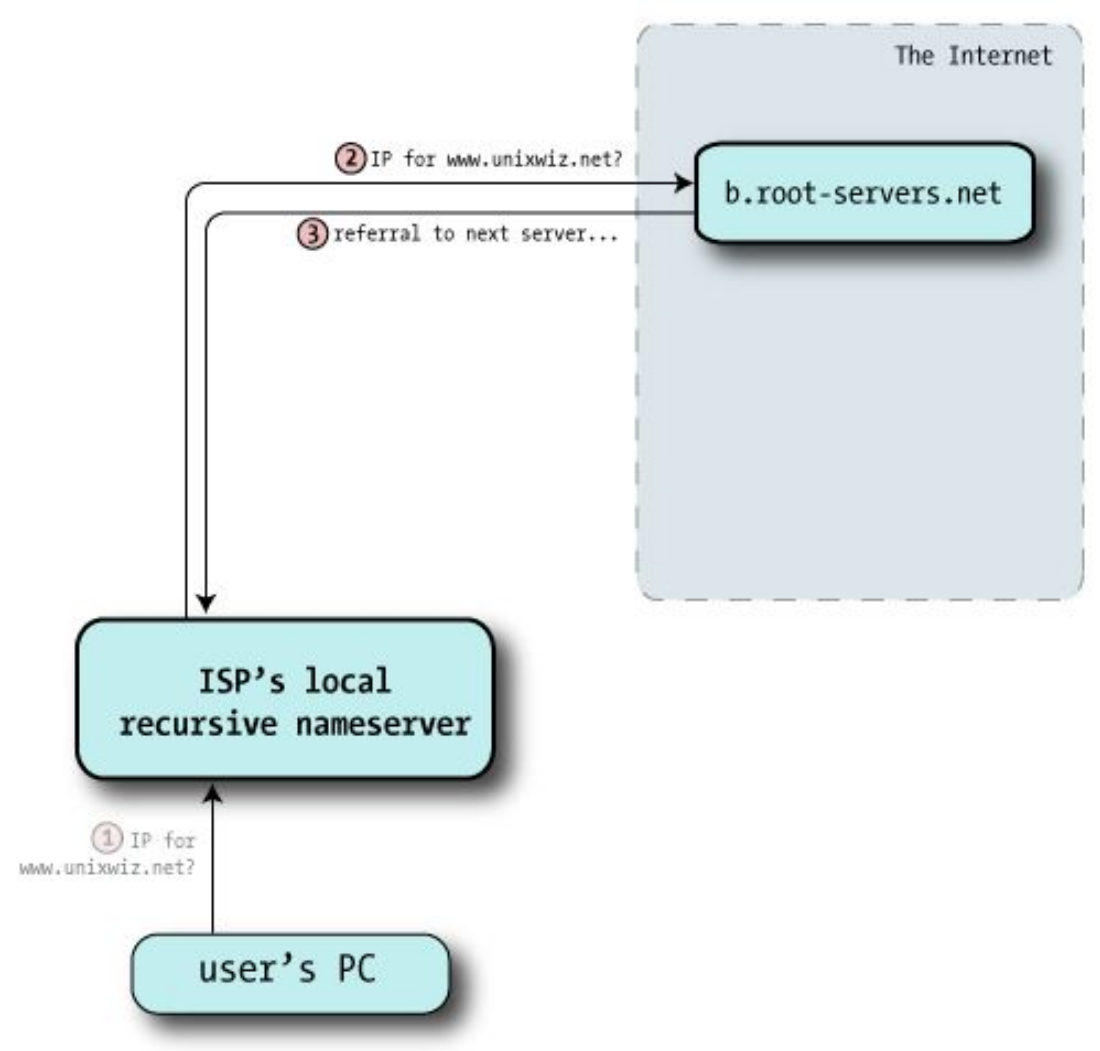
# Exercises

# Q1.2: DNS Recursive Resolution



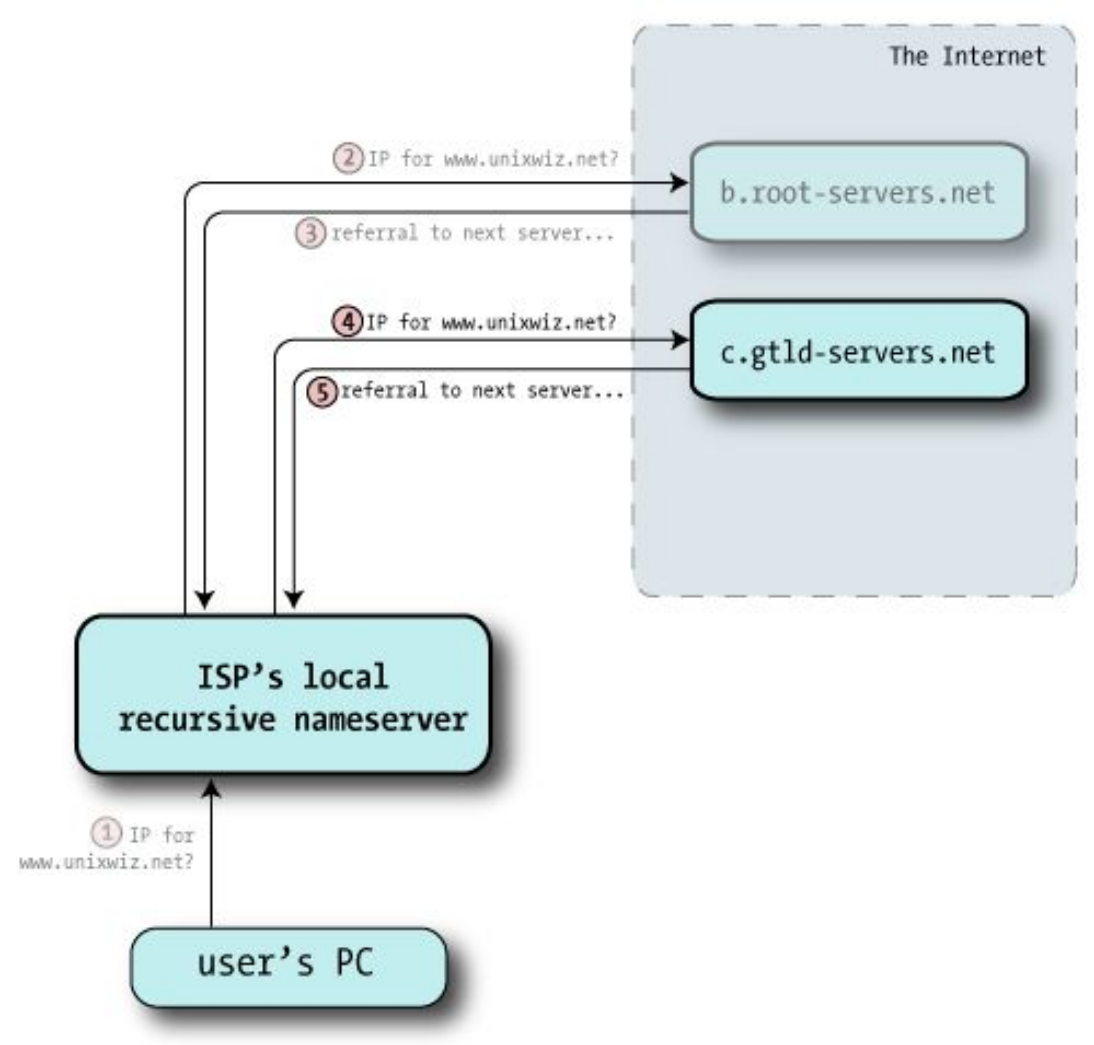

http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

# Q1.2: DNS Recursive Resolution



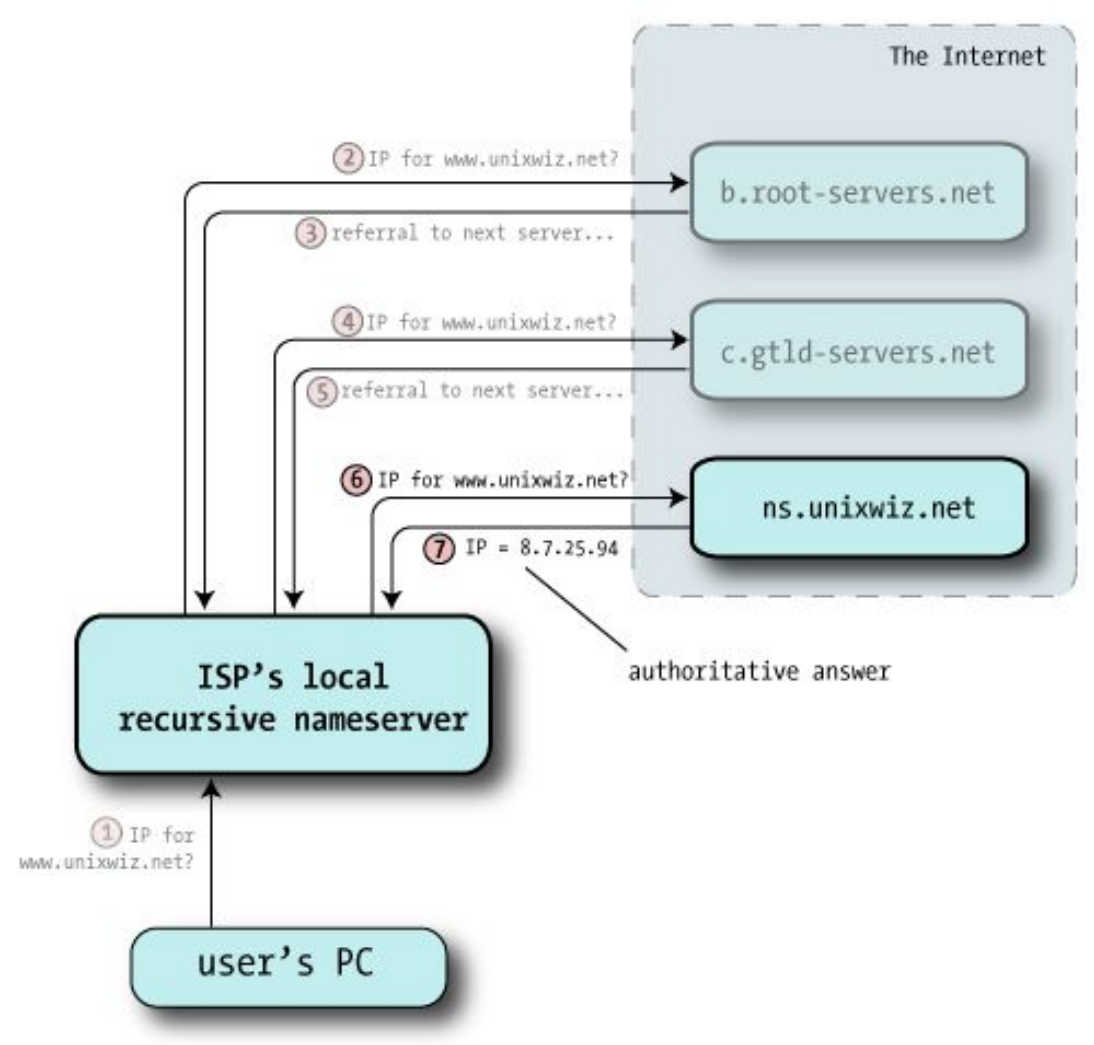

http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

# Q1.2: DNS Recursive Resolution



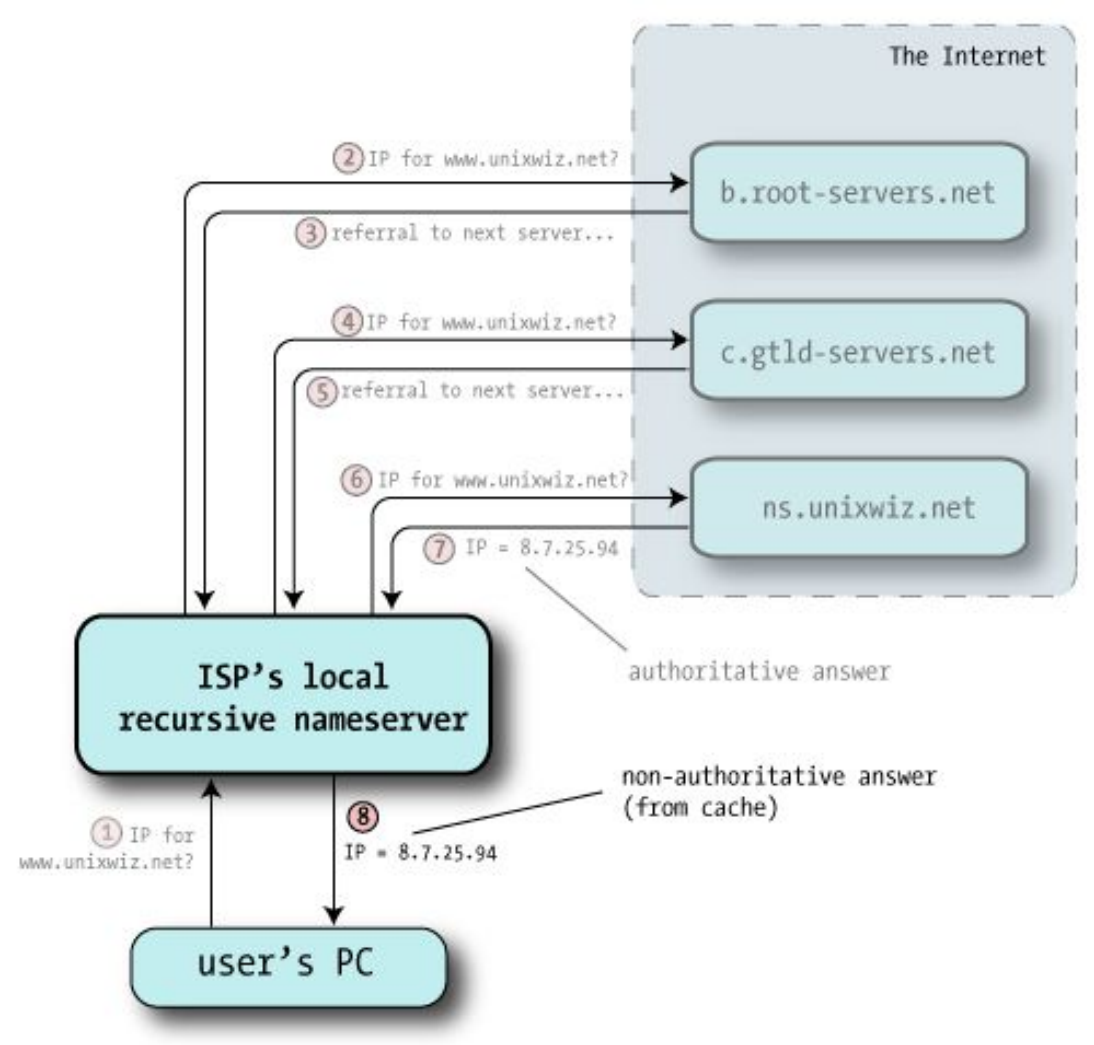

http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

# Q1.2: DNS Recursive Resolution




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

# Q1.2: DNS Recursive Resolution




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

# DNS: Glue

```
dig @a.nic.ch. ethz.ch
;; AUTHORITY SECTION:
ethz.ch.                3600    IN      NS      ns2.ethz.ch.
ethz.ch.                3600    IN      NS      ns2.switch.ch.
ethz.ch.                3600    IN      NS      ns1.ethz.ch.
```

# DNS: Glue

```
dig @a.nic.ch. ethz.ch
;; AUTHORITY SECTION:
ethz.ch.                3600    IN      NS      ns2.ethz.ch.
ethz.ch.                3600    IN      NS      ns2.switch.ch.
ethz.ch.                3600    IN      NS      ns1.ethz.ch.

;; ADDITIONAL SECTION:
ns2.ethz.ch.            3600    IN      AAAA    2001:67c:10ec::b
ns2.switch.ch.          3600    IN      AAAA    2001:620:0:ff::2f
ns1.ethz.ch.            3600    IN      AAAA    2001:67c:10ec::a
```

# DNS: Glue

```
dig @a.nic.ch. ethz.ch
;; AUTHORITY SECTION:
ethz.ch.                3600    IN      NS      ns2.ethz.ch.
ethz.ch.                3600    IN      NS      ns2.switch.ch.
ethz.ch.                3600    IN      NS      ns1.ethz.ch.

;; ADDITIONAL SECTION:
ns2.ethz.ch.            3600    IN      AAAA    2001:67c:10ec::b
ns2.switch.ch.          3600    IN      AAAA    2001:620:0:ff::2f
ns1.ethz.ch.            3600    IN      AAAA    2001:67c:10ec::a
```

# DNS: Glue

```
dig @ns1.attacker.com. attacker.com
;; AUTHORITY SECTION:
attacker.com.           3600    IN      NS      ns2.attacker.com
attacker.com.           3600    IN      NS      ns1.google.com

;; ADDITIONAL SECTION:
ns2.attacker.com        3600    IN      A       1.3.3.7
ns1.google.com          3600    IN      A       1.3.3.7
```

# Q2.1: DNS Lame Delegations

```
"authoritative" response.  If not, this is called a "lame
delegation".  A lame delegations exists when a nameserver is
delegated responsibility for providing nameservice for a zone (via NS
records) but is not performing nameservice for that zone (usually
because it is not set up as a primary or secondary for the zone).

The "classic" lame delegation can be illustrated in this example:

        podunk.xx.      IN      NS      ns1.podunk.xx.
                        IN      NS      ns0.widget.com.

"podunk.xx" is a new domain which has recently been created, and
"ns1.podunk.xx" has been set up to perform nameservice for the zone.
They haven't quite finished everything yet and haven't made sure that
the hostmaster at "ns0.widget.com" has set up to be a proper
```

https://www.ietf.org/rfc/rfc1912.txt

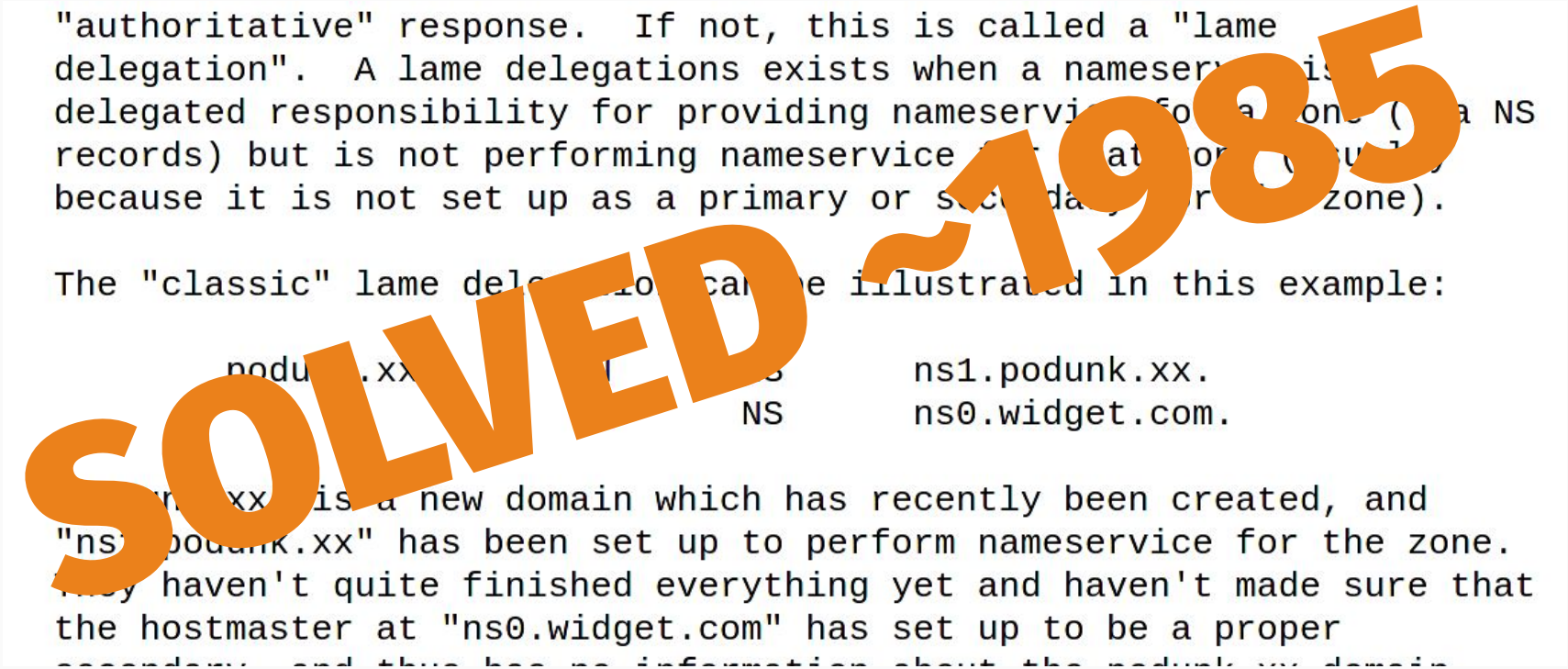
# DNS: Glue

```
dig @a.nic.ch. ethz.ch
;; AUTHORITY SECTION:
ethz.ch.                3600    IN      NS      ns2.ethz.ch.
ethz.ch.                3600    IN      NS      ns2.switch.ch.
ethz.ch.                3600    IN      NS      ns1.ethz.ch.

;; ADDITIONAL SECTION:
ns2.ethz.ch.            3600    IN      AAAA    2001:67c:10ec::b
ns2.switch.ch.          3600    IN      AAAA    2001:620:0:ff::2f
ns1.ethz.ch.            3600    IN      AAAA    2001:67c:10ec::a
```

# DNS: Glue

```
dig @ns1.attacker.com. attacker.com
;; AUTHORITY SECTION:
attacker.com.           3600    IN      NS      ns2.attacker.com
attacker.com.           3600    IN      NS      ns1.google.com

;; ADDITIONAL SECTION:
ns2.attacker.com        3600    IN      A       1.3.3.7
ns1.google.com          3600    IN      A       1.3.3.7
```

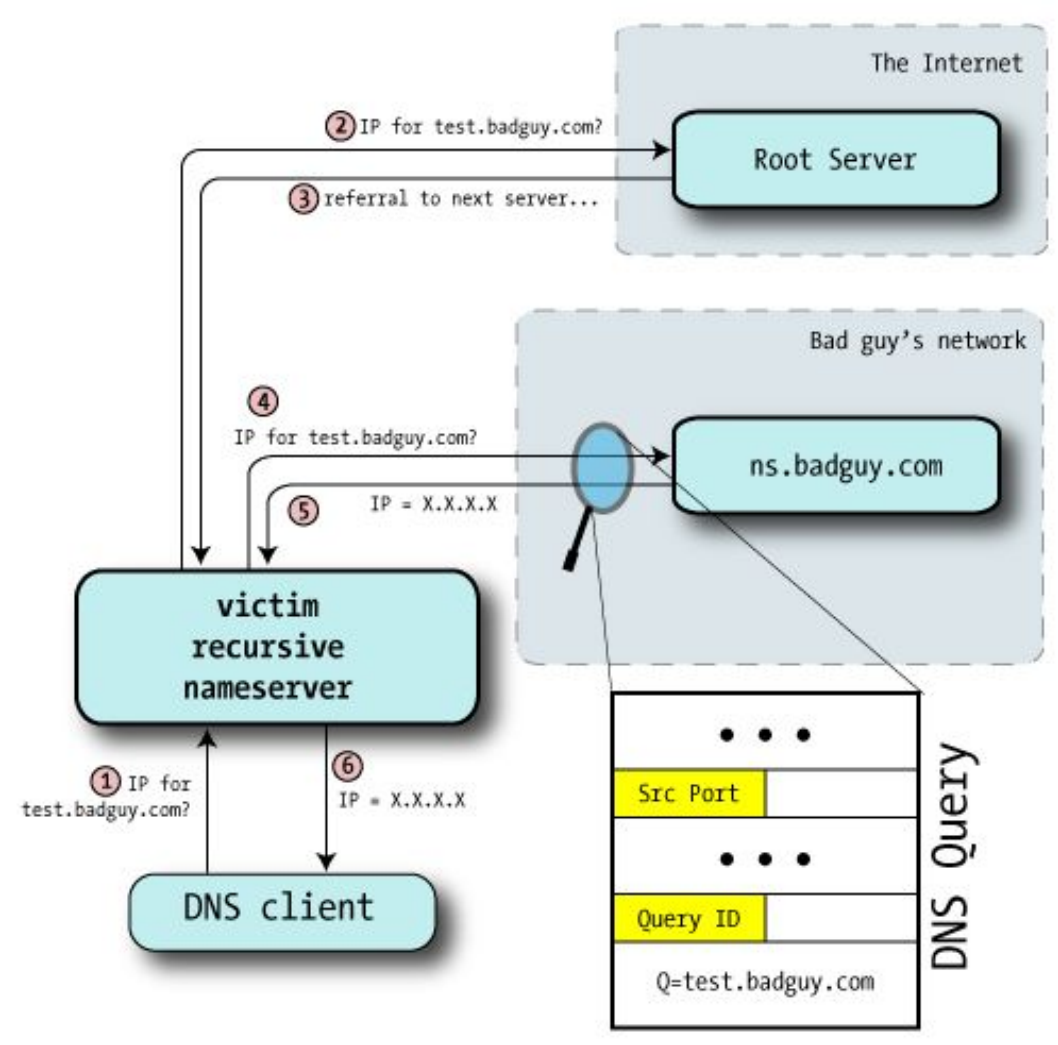
# Q2.1: DNS Lame Delegations




SOLVED ~1985

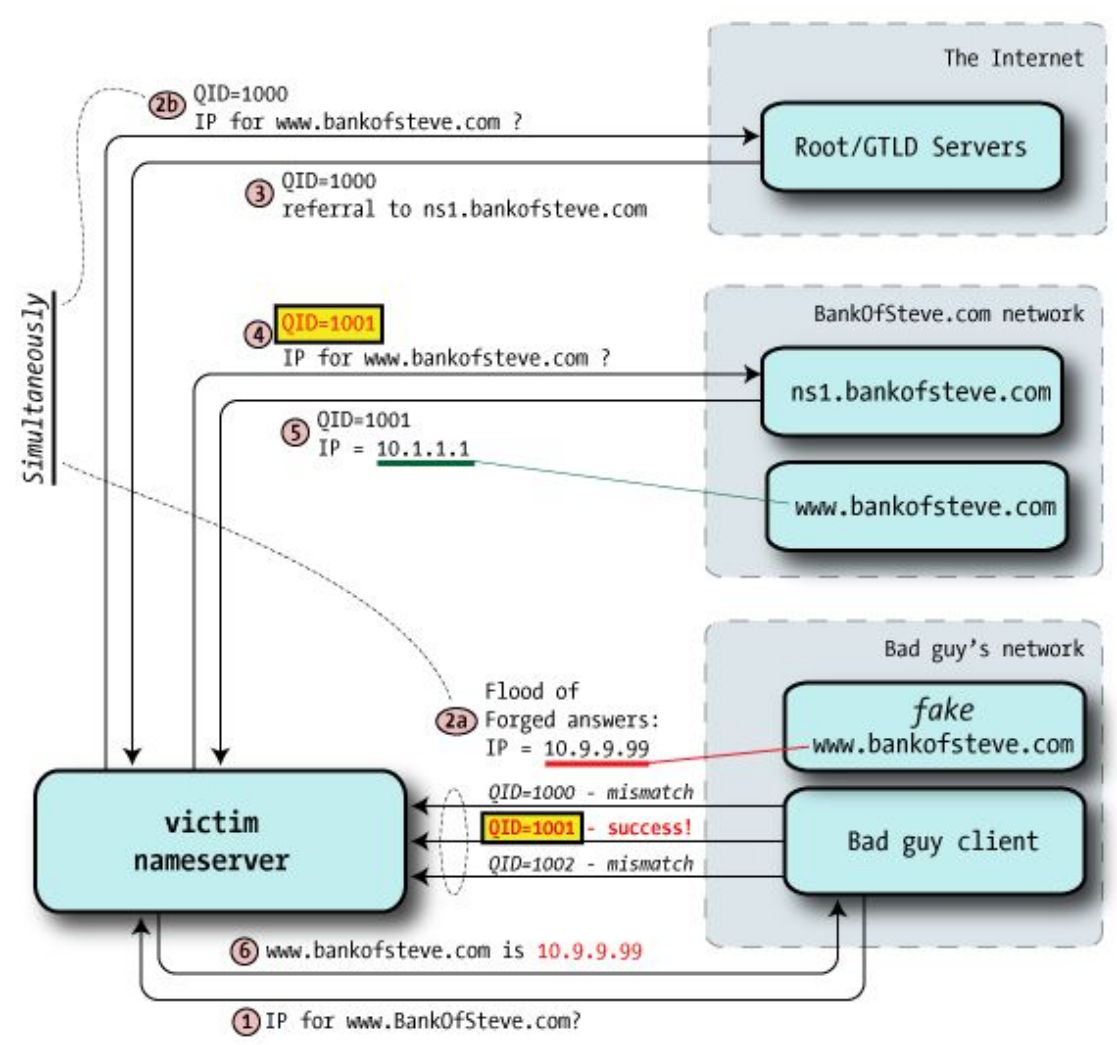
https://www.ietf.org/rfc/rfc1912.txt

# DNS: 2009 Kaminsky Attack



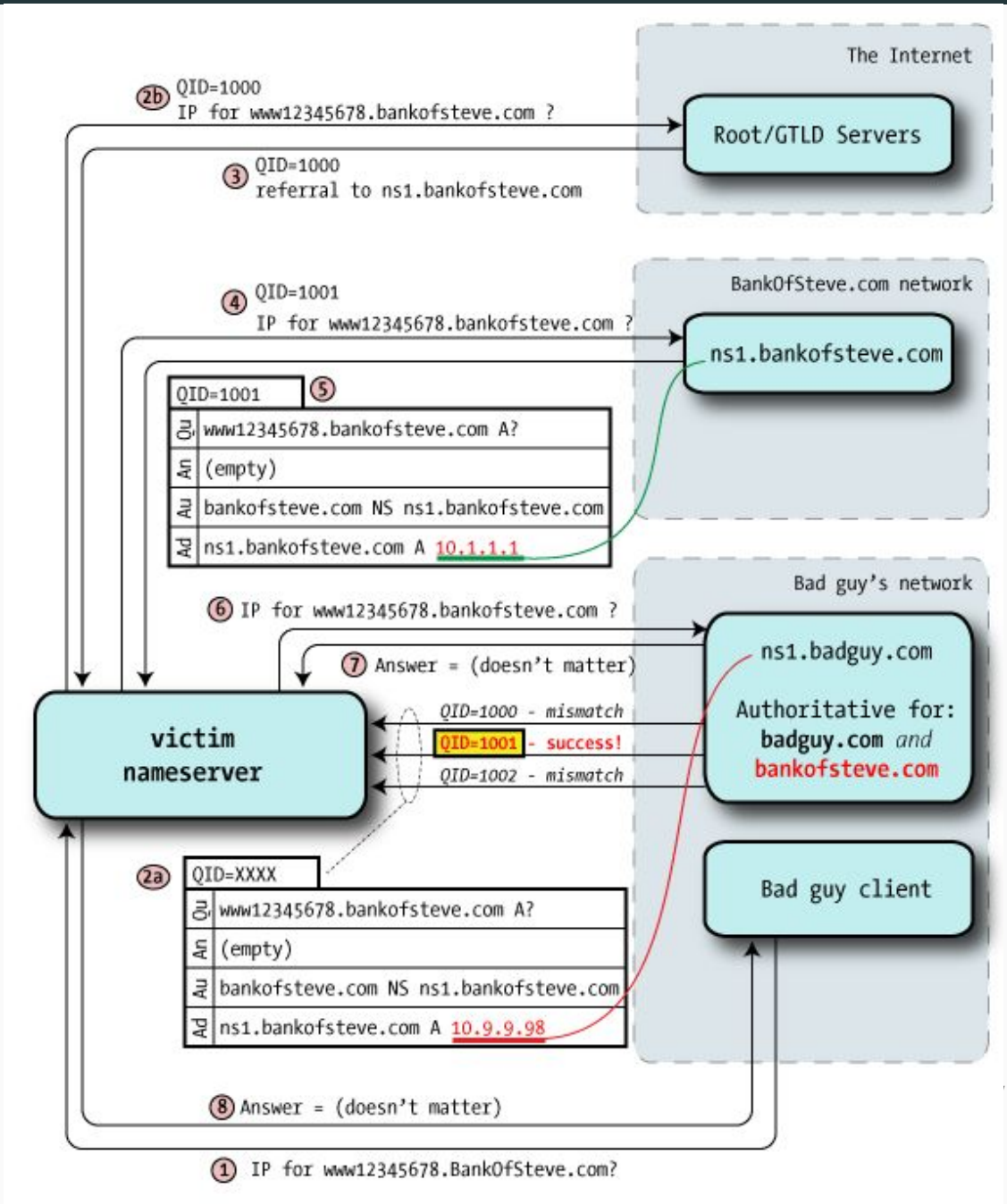

http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

# DNS: 2009 Kaminsky Attack




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

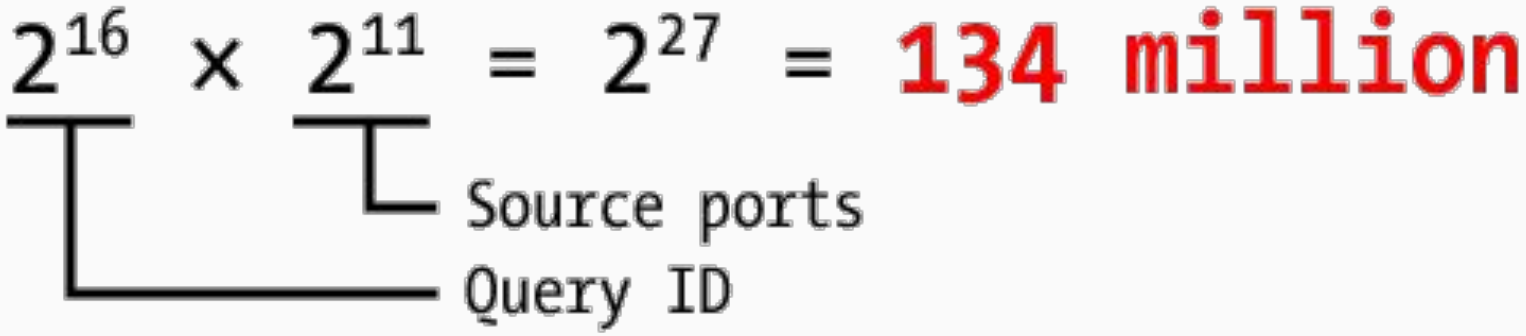
# DNS: 2009 Kaminsky Attack




ixwiz.net/techtips/iguide-
-dns-vuln.html

# DNS: 2009 Fix

Use random ports!




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

# DNS: 2009 Fix

Use random ports!

$2^{16} \times 2^{11} = 2^{27} =$ 134 million

Source ports

Query ID

SOLVED ~2009

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

# DNS: 2009 Fix

$2^{16} \times 2^{11} = 2^{27} = 134$ million

Source ports

Query ID

**DJB Was Right**

One nameserver is notable for having gotten **both** the query-id and source-port randomness right from the start: DJBDNS by the legendary Daniel J. Bernstein.

Though long a lightning rod for controversy, he's clearly walked the walk on security: there's been just one minor security vulnerability in DJBDNS.

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

# Q2.2.1: Forgery

Use random ports!

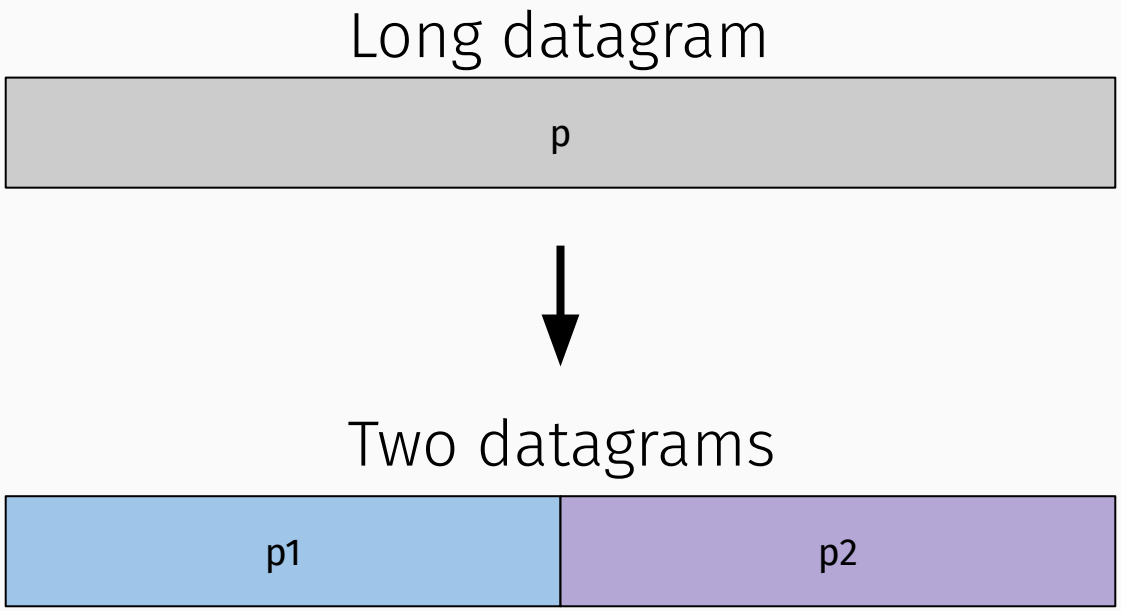
$2^{16} \times 2^{11} = 2^{27} =$ **134 million**

Source ports

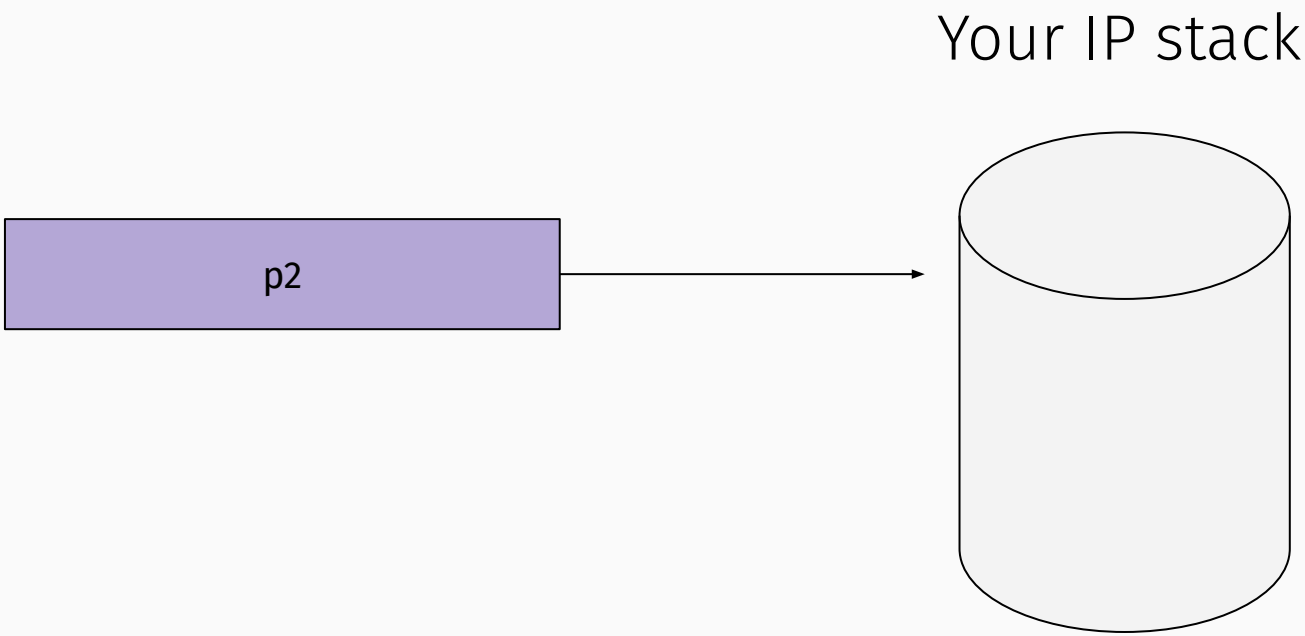
Query ID

# Q2.2.2: Fragmentation

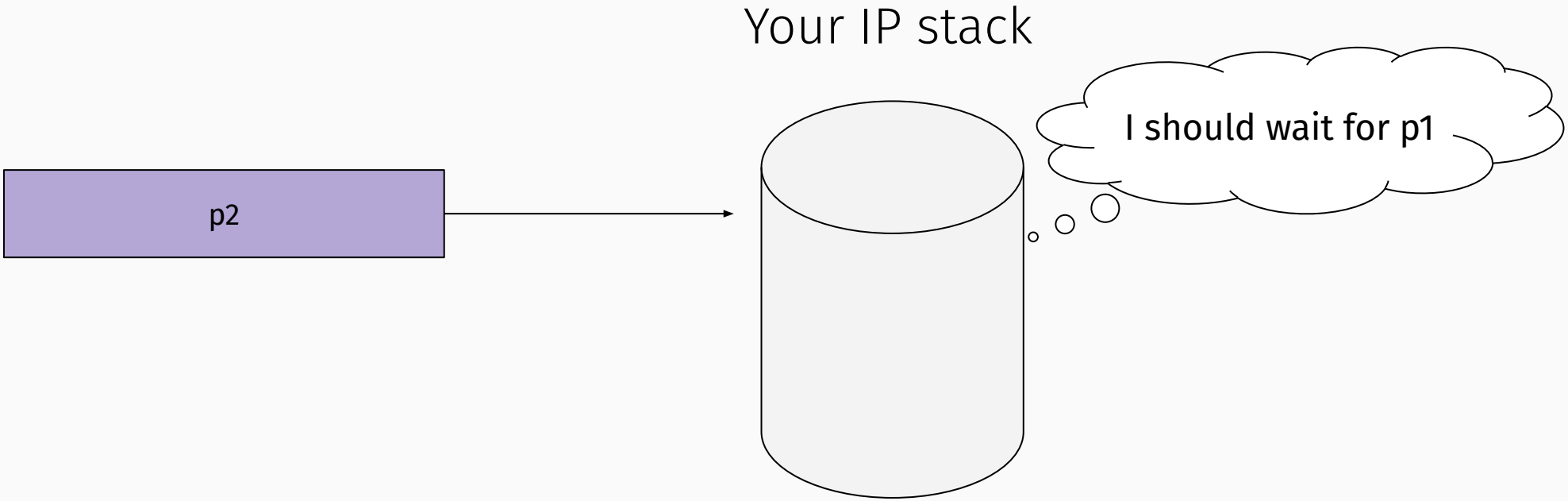
# IP Fragmentation



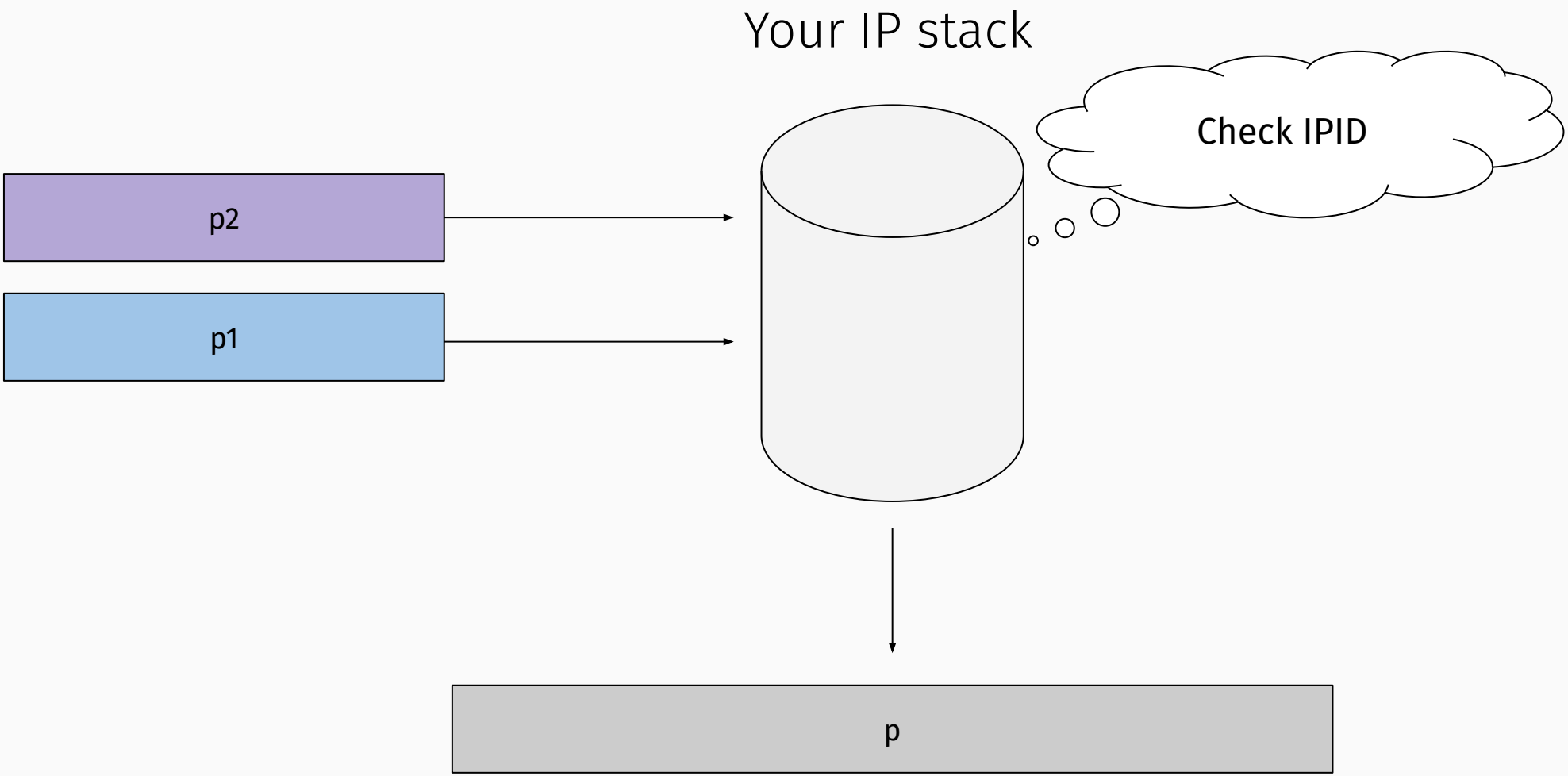
# IP Fragmentation



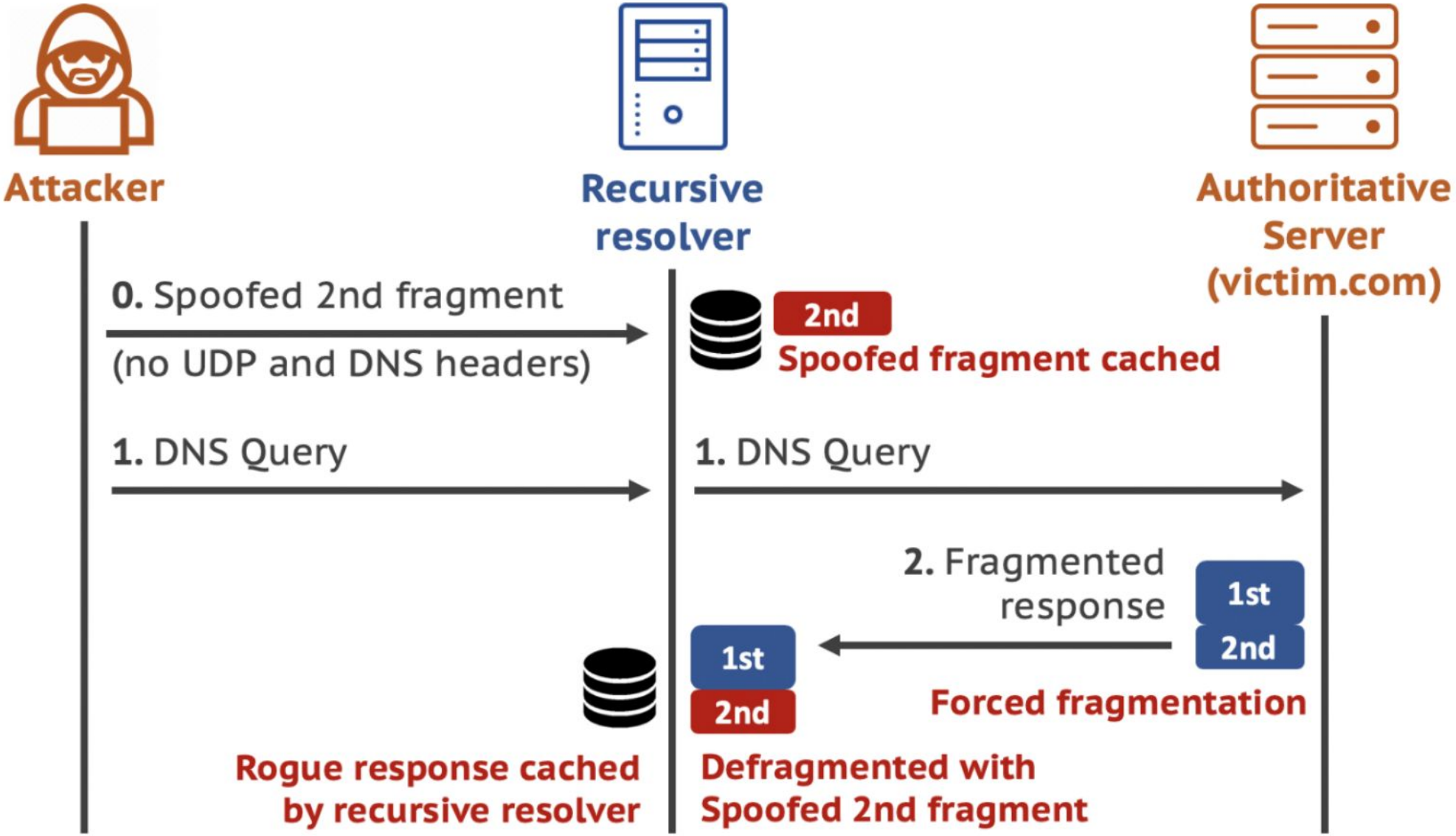
# IP Fragmentation



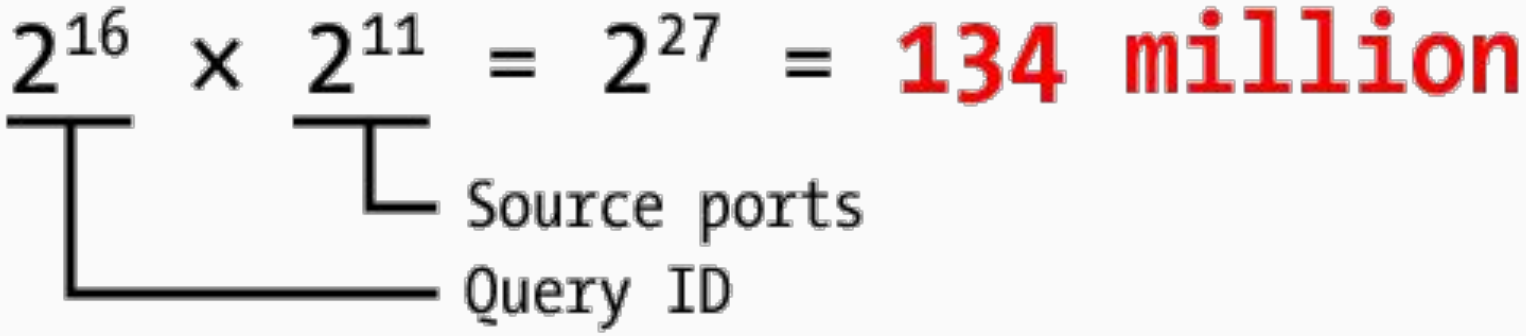
# IP Fragmentation



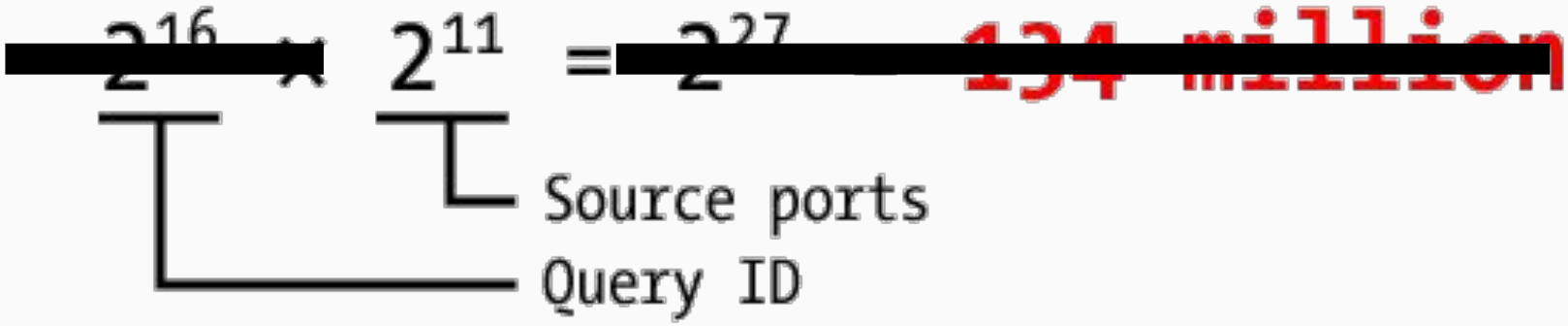
# Q2.2.2: Fragmentation



# Q2.2.2: Fragmentation



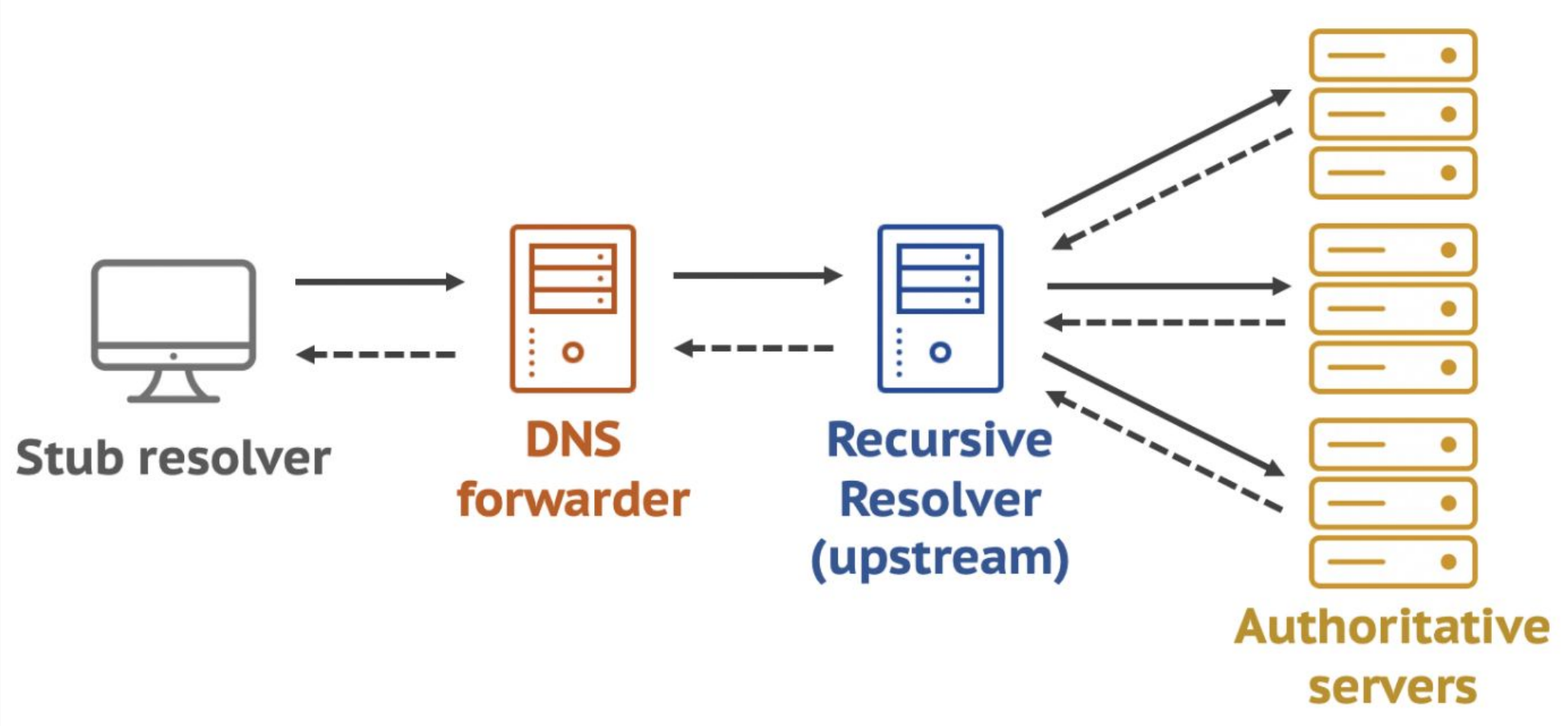
# Q2.2.2: Fragmentation



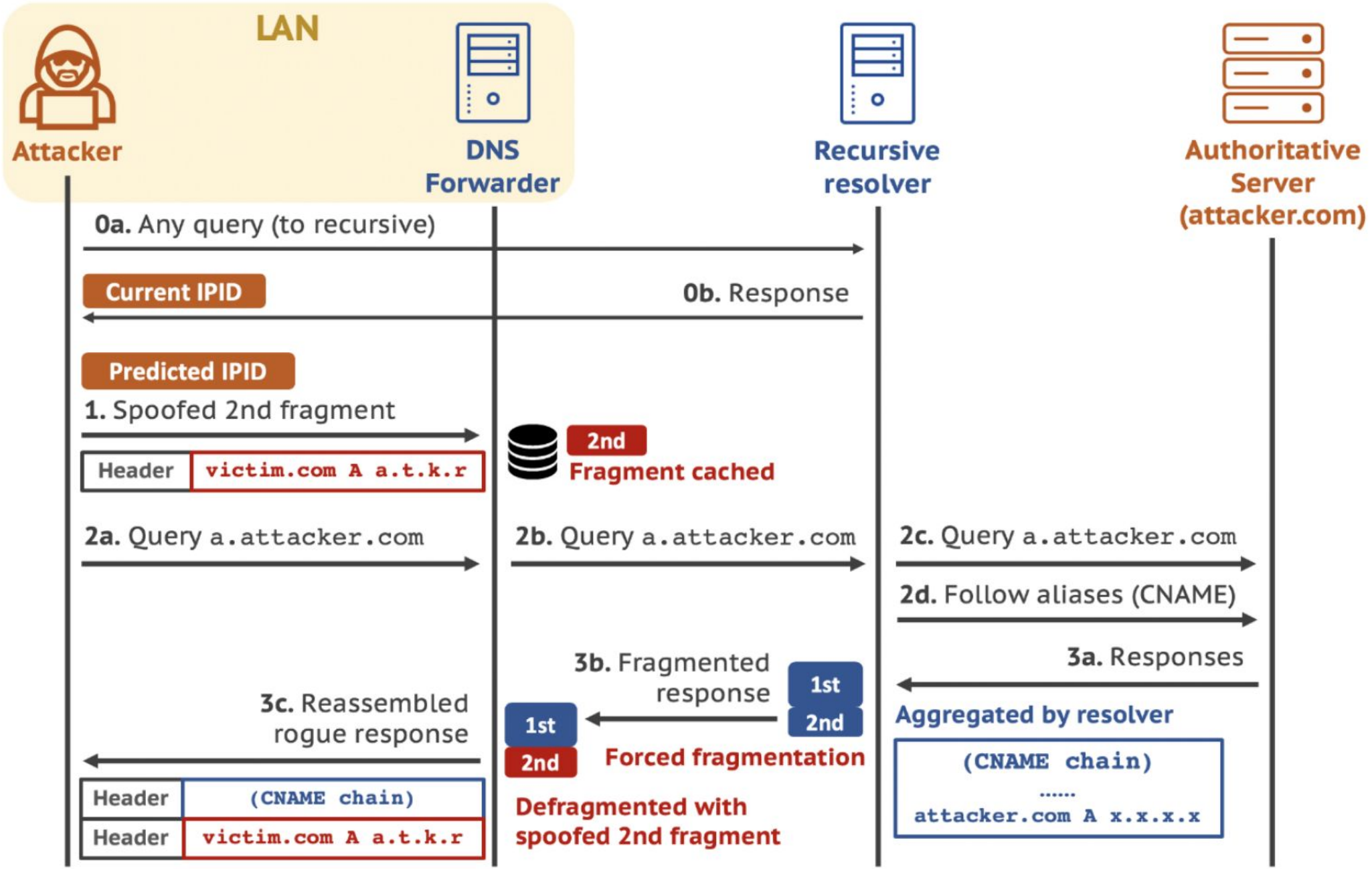

+ IPID (predictable)

# Q2.3: Forwarders

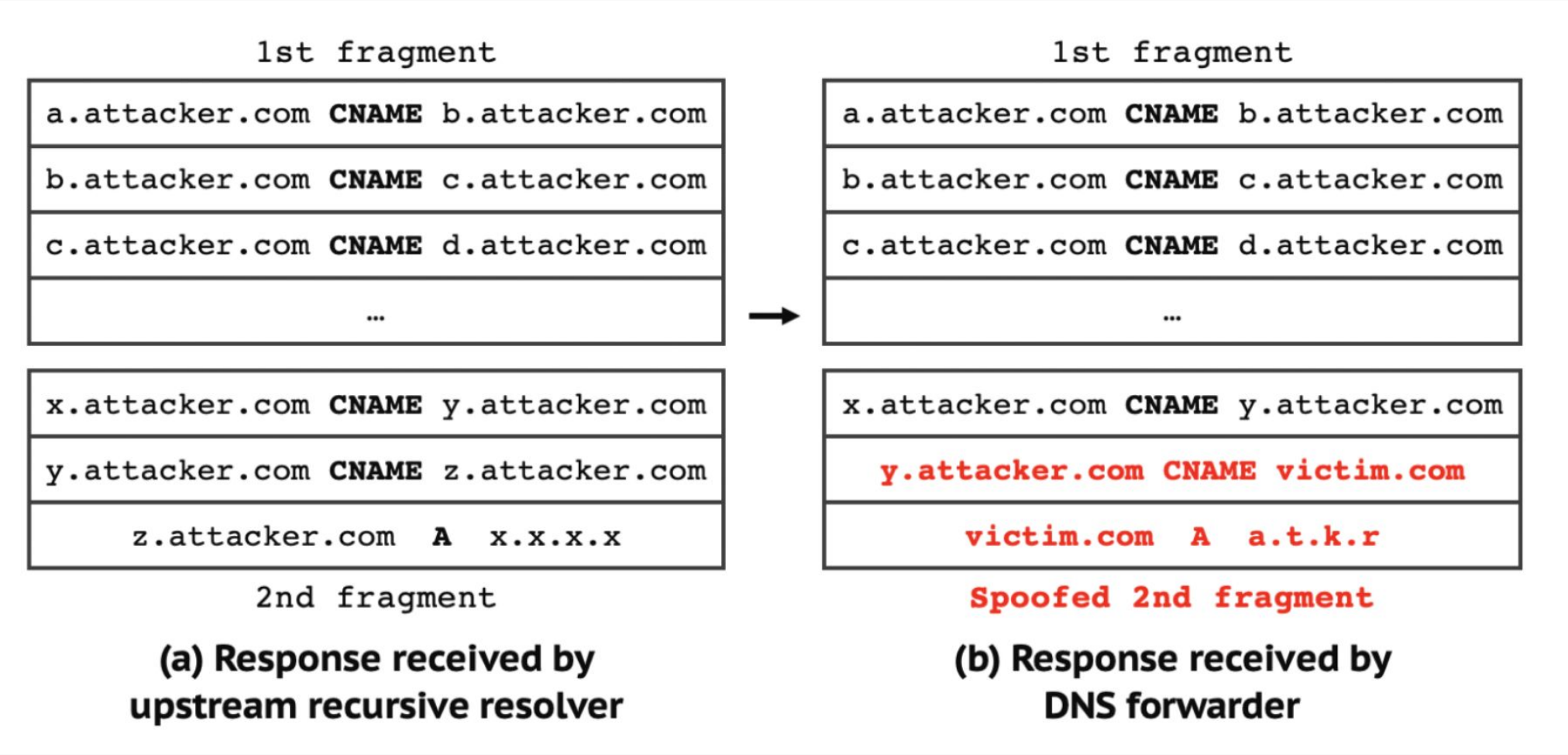
# Q2.3: Forwarders



# Q2.4: Fragmenting Forwarder

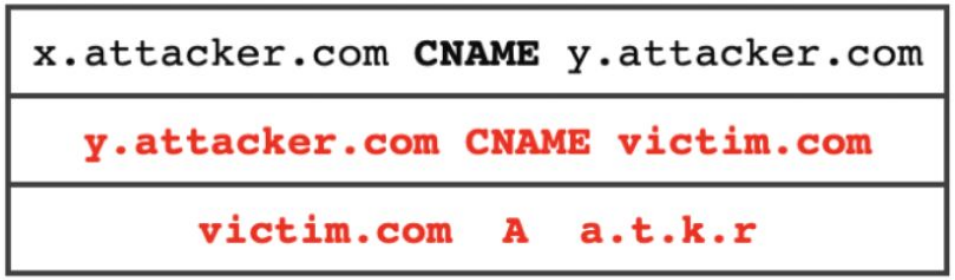


# Q2.4: Fragmenting Forwarder



# Q2.5: Attacking Recursive Resolvers?

# Q2.5: Attacking Recursive Resolvers?




**(b) Response received by DNS forwarder**

"authoritative" response. If not, this is called a "lame delegation". A lame delegations exists when a nameserver is delegated responsibility for providing nameservice for a zone (via NS records) but is not performing nameservice for that zone (usually because it is not set up as a primary or secondary for the zone).

The "classic" lame delegation can be illustrated in this example:

```
        podunk.xx.      IN      NS      ns1.podunk.xx.
                        IN      NS      ns0.widget.com.
```

"podunk.xx" is a new domain which has recently been created, and "ns1.podunk.xx" has been set up to perform nameservice for the zone. They haven't quite finished everything yet and haven't made sure that the hostmaster at "ns0.widget.com" has set up to be a proper

# Q2.5: Attacking Recursive Resolvers?

| x.attacker.com **CNAME** y.attacker.com |
| --- |
| y.attacker.com CNAME victim.com |
| victim.com A a.t.k.r |

Spoofed 2nd fragment

**(b) Response received by DNS forwarder**

SOLVED ~1985

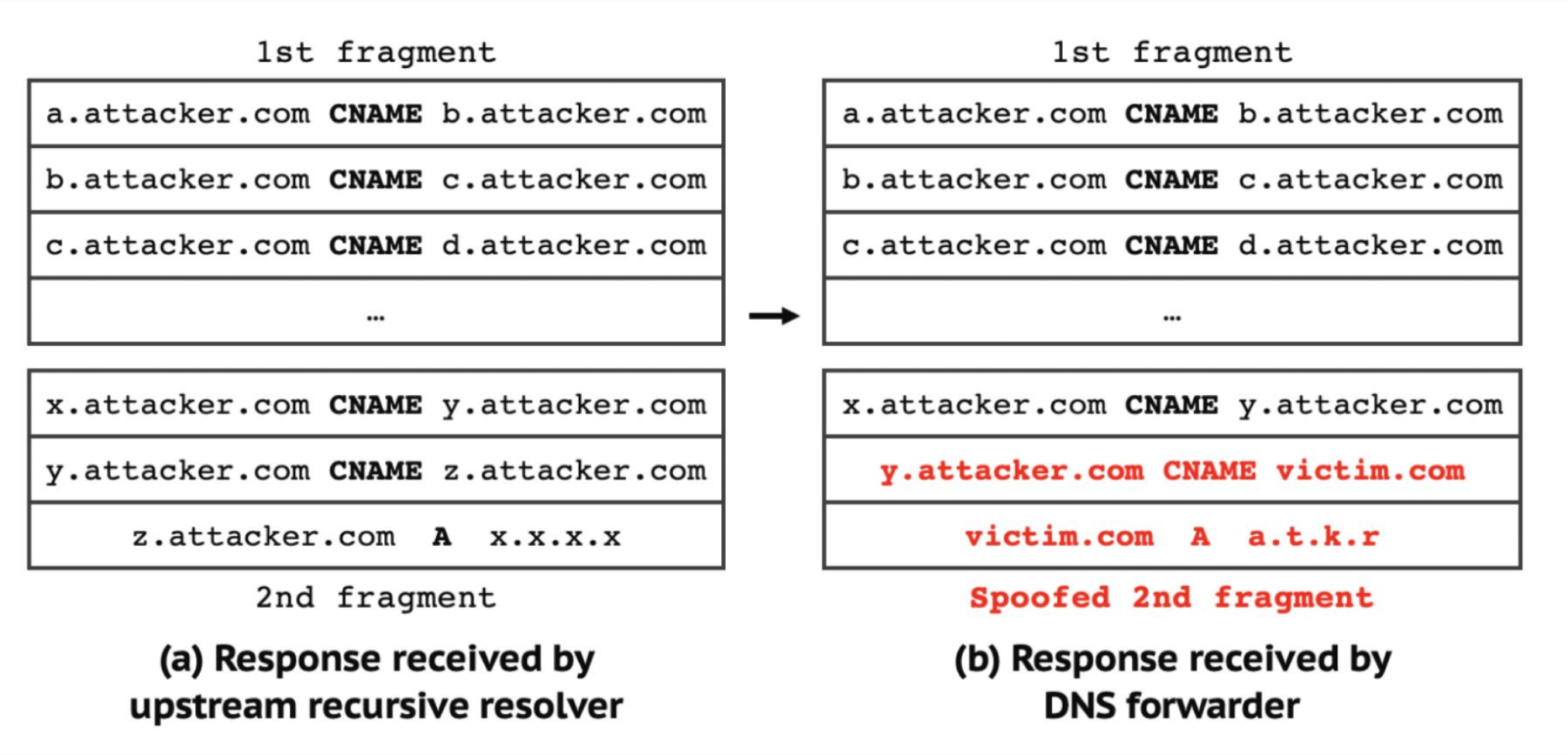
"author[illegible]" [illegible] not, this is called a "lame
[illegible] exists when a nameserver is
[illegible] for providing nameservice for a zone (via NS
[illegible] not performing nameservice for that zone (usually
[illegible] it is not set up as a primary or secondary for the zone).

The "classic" lame delegation can be illustrated in this example:

```
        podunk.xx.      IN      NS      ns1.podunk.xx.
                        IN      NS      ns0.widget.com.
```

"podunk.xx" is a new domain which has recently been created, and
"ns1.podunk.xx" has been set up to perform nameservice for the zone.
They haven't quite finished everything yet and haven't made sure that
the hostmaster at "ns0.widget.com" has set up to be a proper
[illegible]

# Q2.5: Attacking Recursive Resolvers?



# Q2.6: Mitigations

# Q2.6: Mitigations

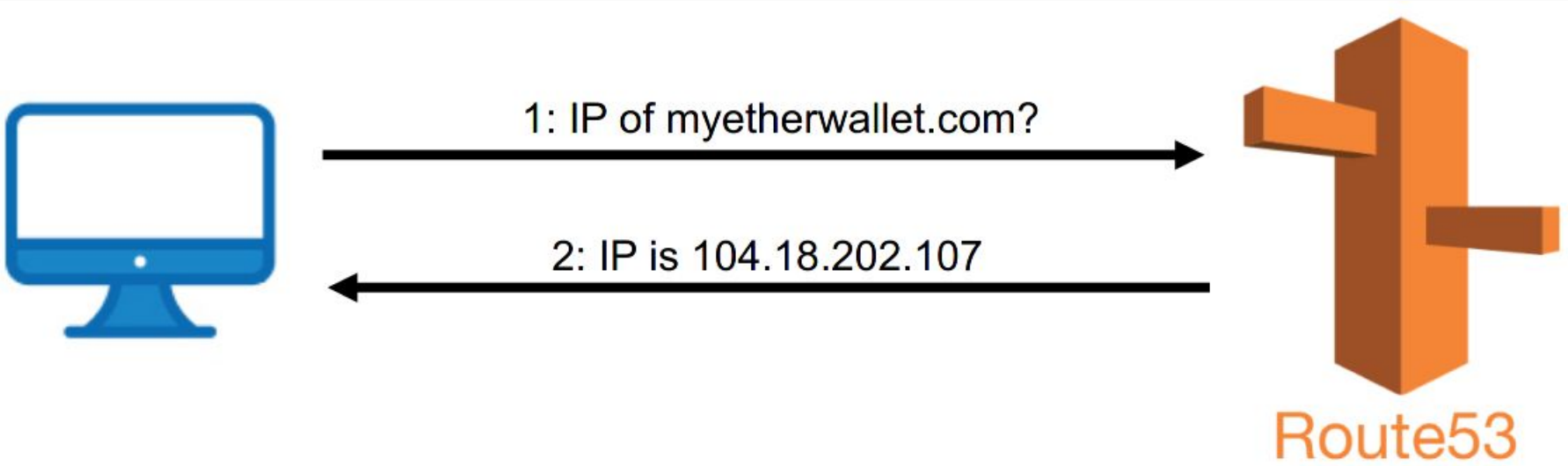
| |
|---|
| x.attacker.com **CNAME** y.attacker.com |
| **y.attacker.com CNAME victim.com** |
| **victim.com A a.t.k.r** |

Caching by record: victim.com -> a.t.k.r

Caching by query: victim.com -> new query

attacker.com -> ca

# Q3: MyEtherWallet

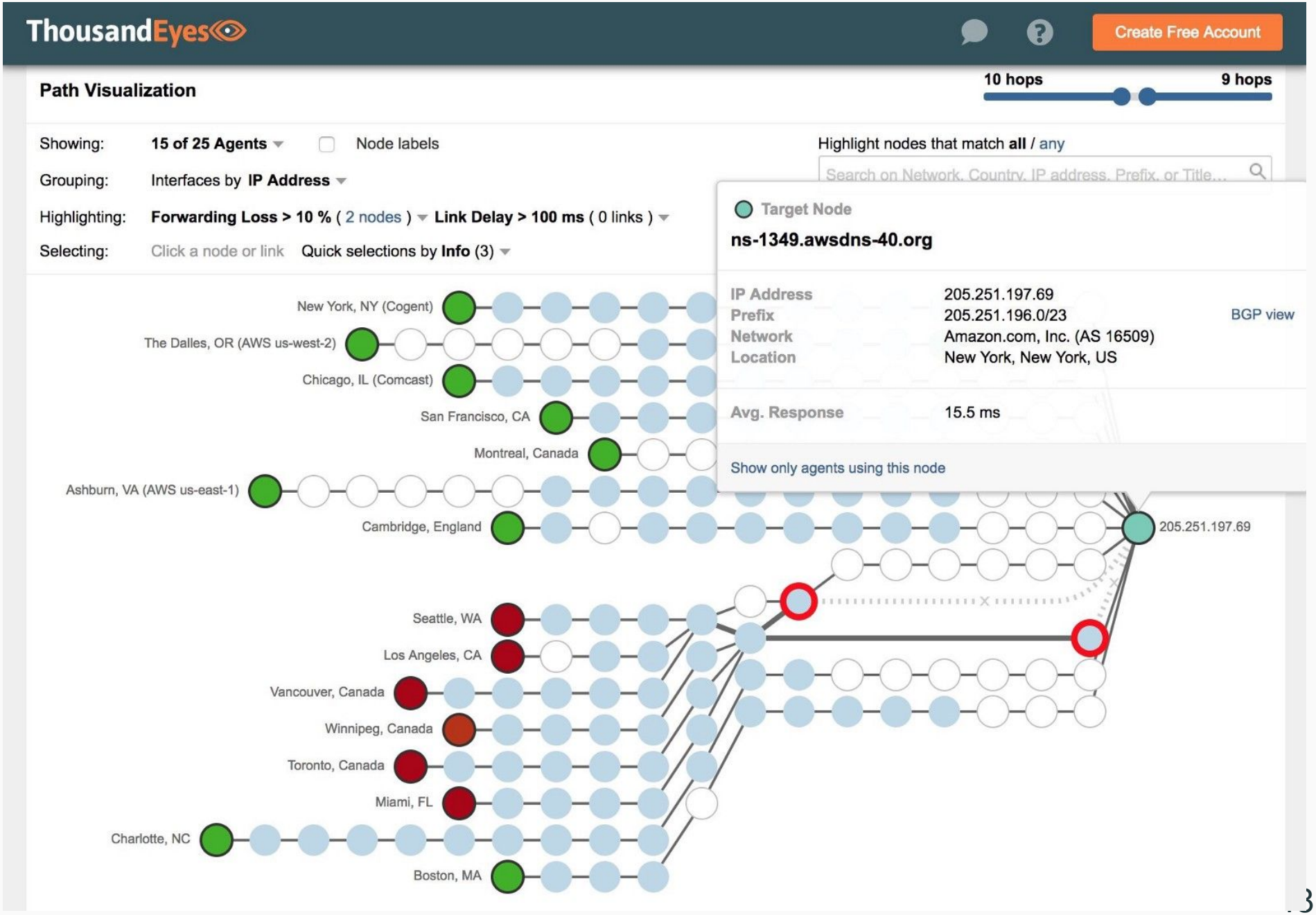



Authoritative servers of MyEtherWallet hosted at

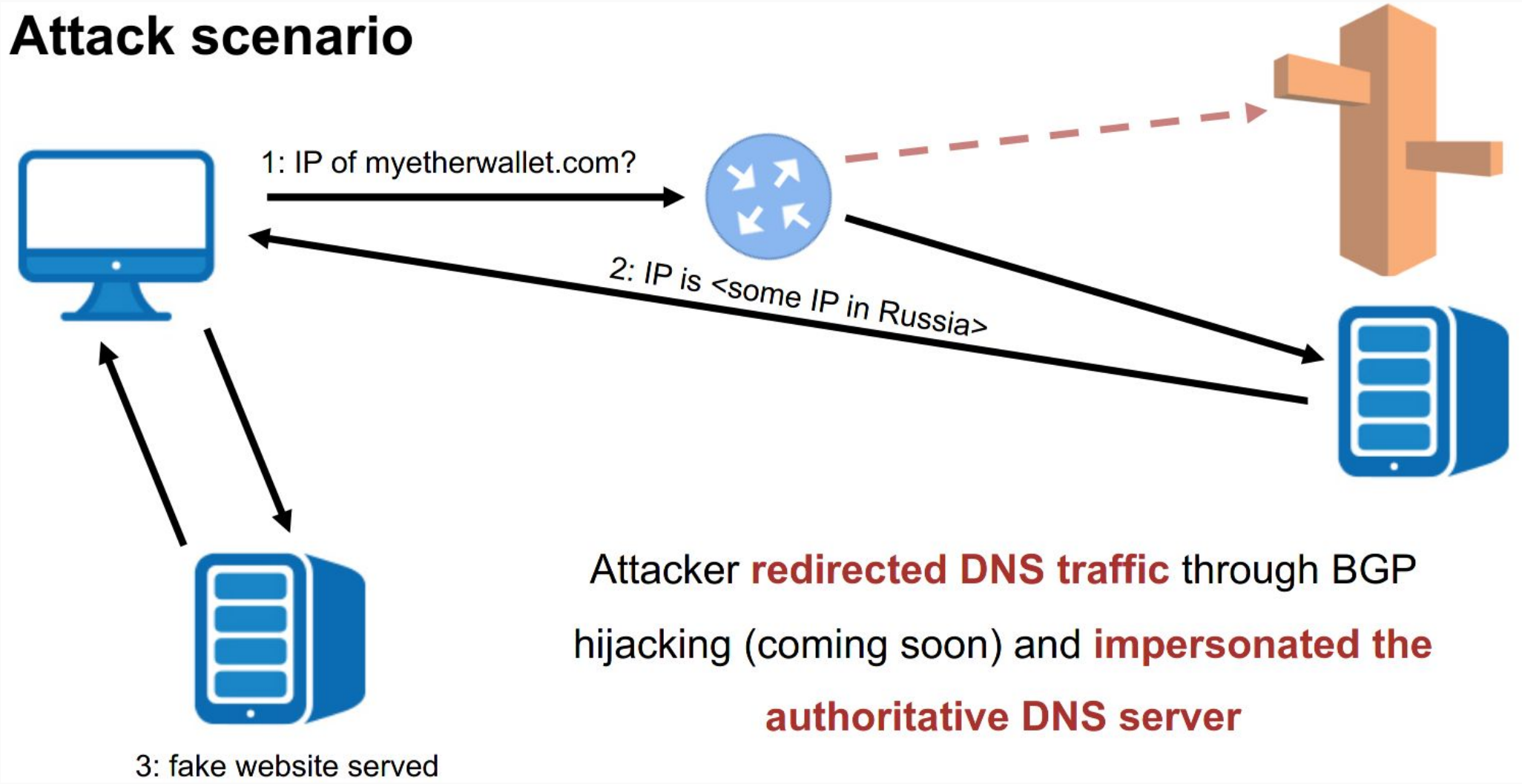
**Amazon AWS Route53**

Slide from Tommaso Ciussani

# Q3: MyEtherWallet



# Q3: MyEtherWallet

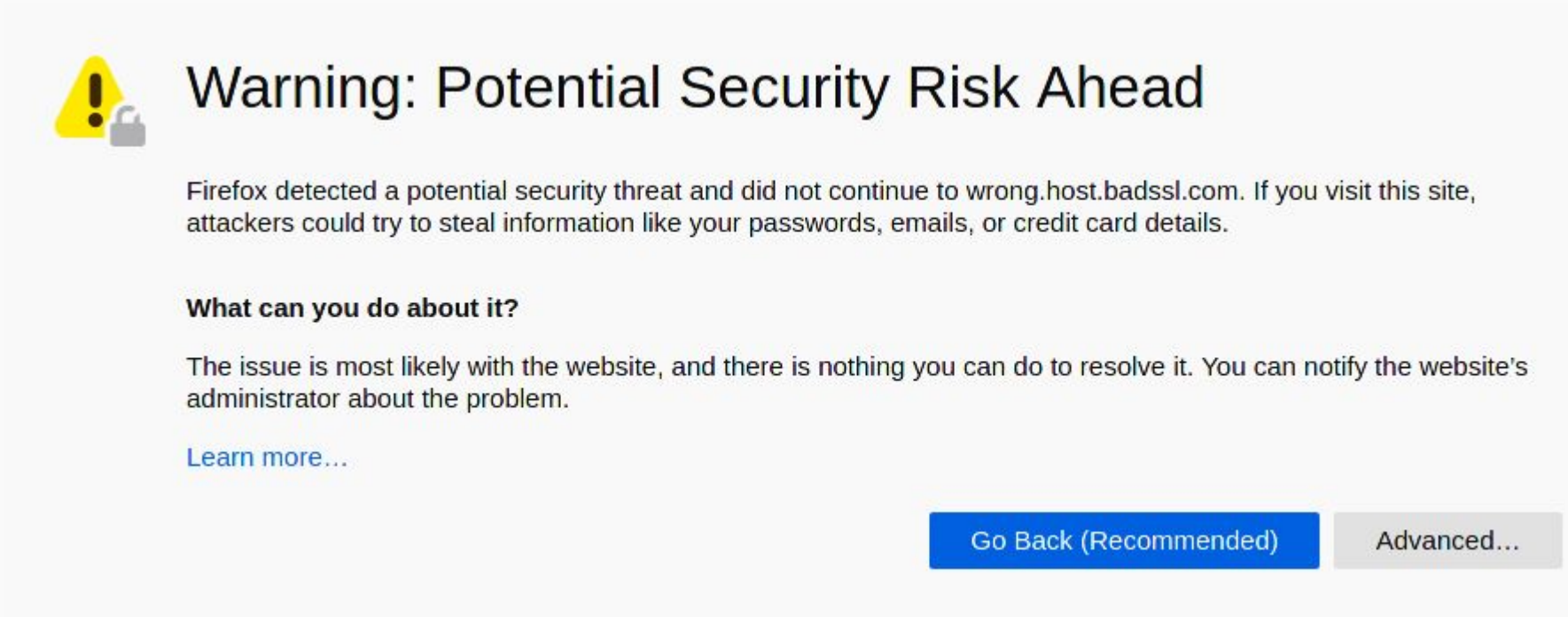



Slide from Tommaso Ciussani

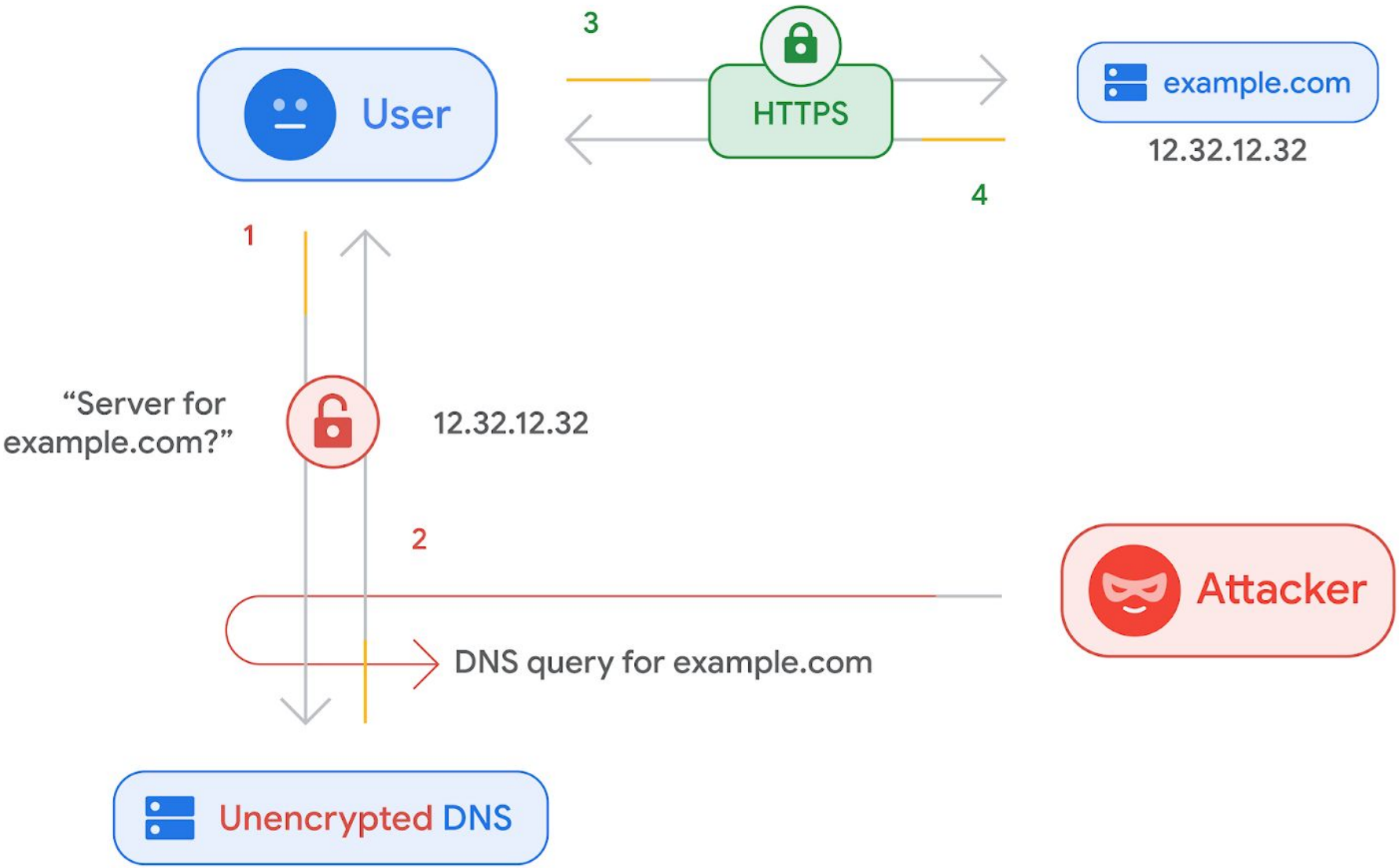
# Q3: Hard to fix

1. DNS not authenticated!
2. BGP!
3. Recursive resolvers cached the wrong authoritative answer!

# Q3: MyEtherWallet

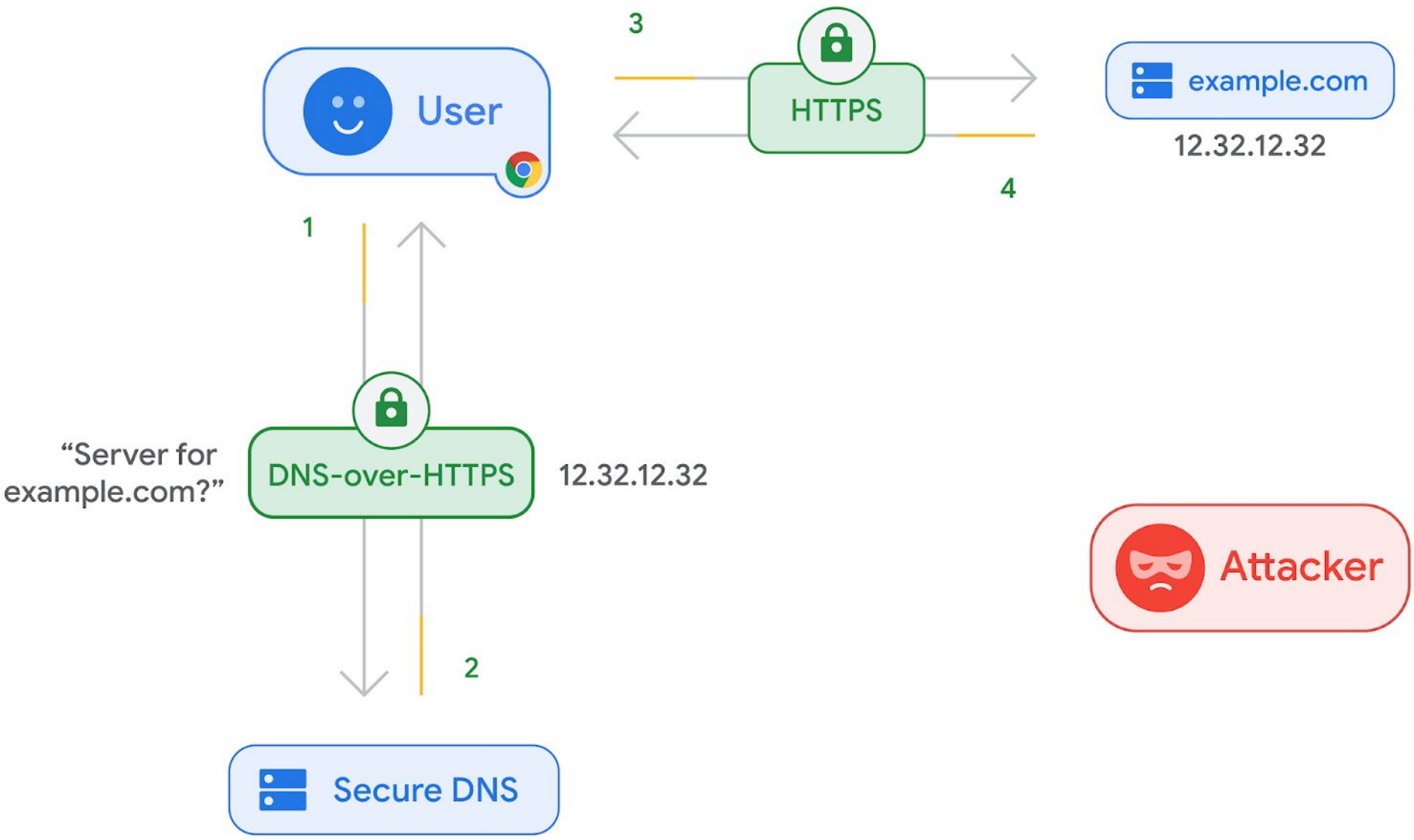


# DoH




https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html

# DoH




https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html

# DoH: Firefox

# DoH: Chrome

Google Public DNS +

```
205         new DohProviderEntry(
206             "Switch", base::nullopt /* provider_id_for_histogram */,
207             {"130.59.31.251", "130.59.31.248", "2001:620:0:ff::2",
208              "2001:620:0:ff::3"},
209             {"dns.switch.ch"} /* dns_over_tls_hostnames */,
210             "https://dns.switch.ch/dns-query", "" /* ui_name */,
211             "" /* privacy_policy */, false /* display_globally */,
212             {} /* display_countries */),
213   }};
214   return *providers;
215 }
```

# DoH: Fallback

**How does Firefox handle split-horizon DNS?**

If Firefox fails to resolve a domain via DoH, it will fall back to the DNS. This means that any domains that are only available on the ordinary DNS (because they aren't public) will be resolved that way. If you have a domain that is publicly resolvable but resolves differently internally, then you should use enterprise settings to disable DoH.

such as family-safe filtering, and therefore avoid breaking user expectations. Furthermore, if there's any hiccup with the DNS-over-HTTPS connection, Chrome will fall back to the regular DNS service of the user's current provider by default, in order to avoid any disruption, while periodically retrying to secure the DNS communication. Finally, to avoid an issue that otherwise could arise