ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΑΘΗΜΑ: «ΑΣΦΑΛΕΙΑ ΔΙΚΤΥΩΝ (8ο ΕΞΑΜΗΝΟ)»

ΚΑΘΗΓΗΤΕΣ: ΚΟΤΖΑΝΙΚΟΛΑΟΥ ΠΑΝΑΓΙΩΤΗΣ

ΟΜΑΔΑ ΕΡΓΑΣΙΑΣ:

ΔΗΜΗΤΡΕΛΛΟΣ ΠΑΝΑΓΙΩΤΗΣ, Π17026

ΚΑΡΑΜΠΟΪΚΗΣ ΝΙΚΟΛΑΟΣ, Π17040

ΡΟΥΝΤΟΥ ΑΝΝΑ-ΦΑΝΗ, Π17113

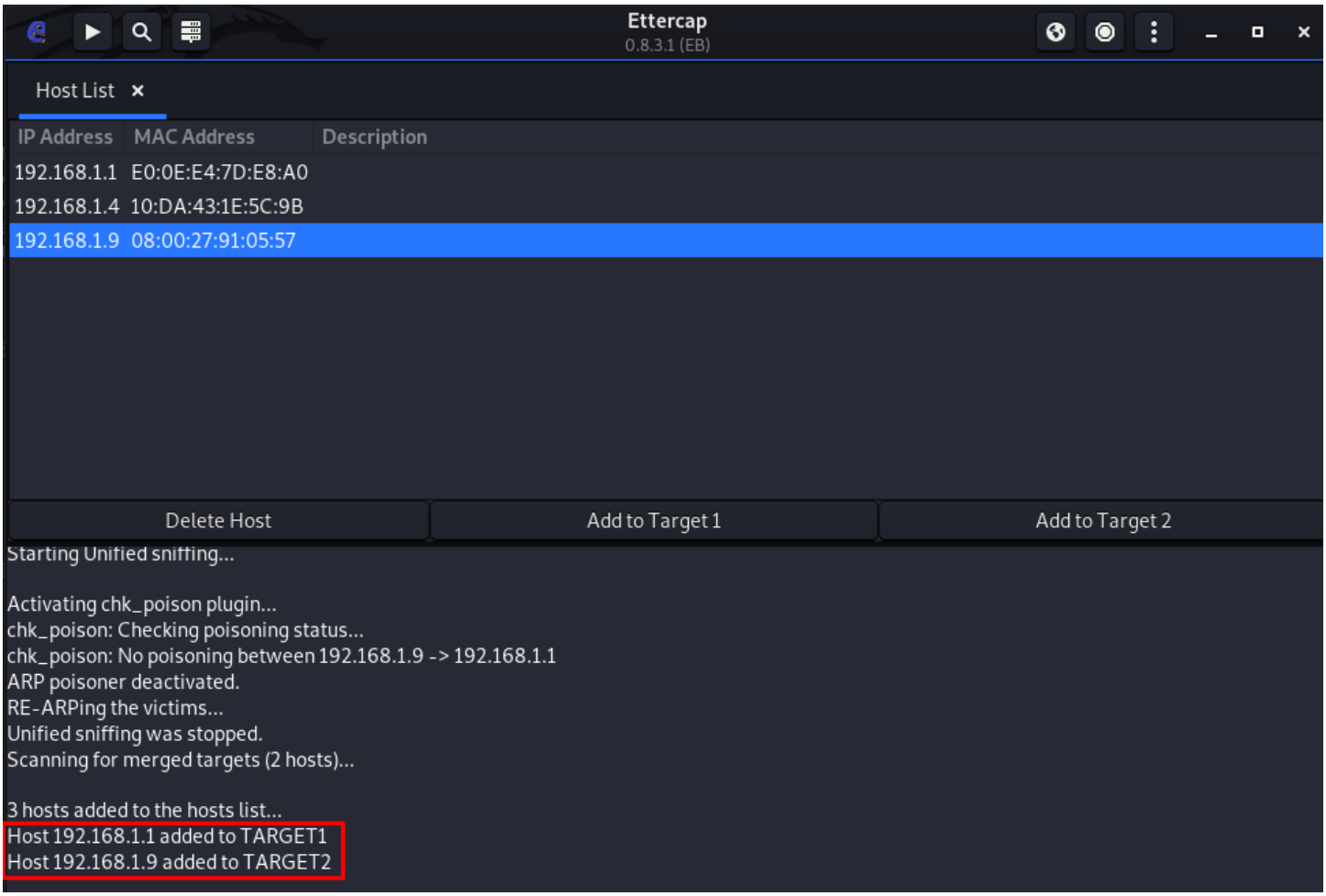
2η Άσκηση - Επίθεση ARP spoofing

ΠΕΡΙΕΧΟΜΕΝΑ

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

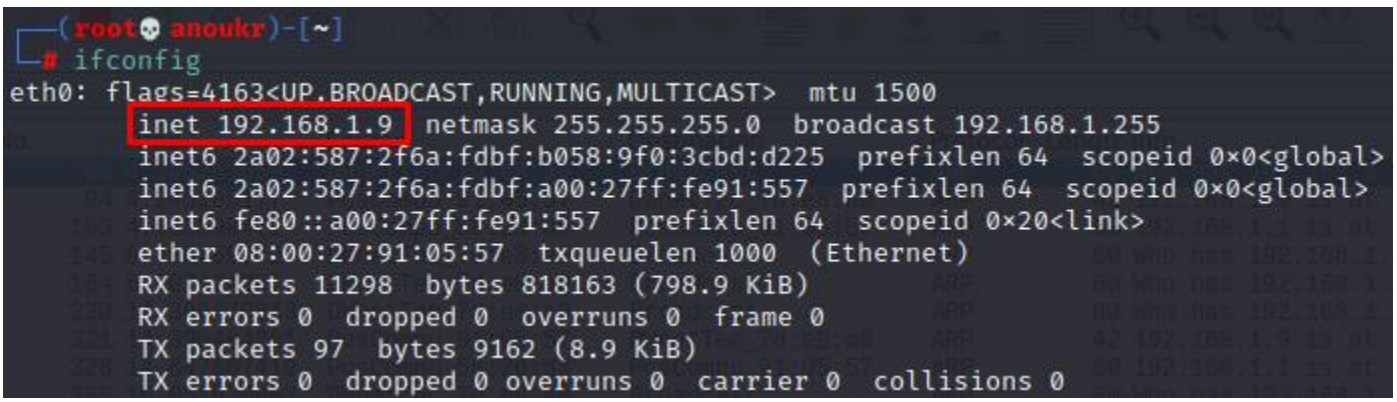
   a) Ettercap.

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".



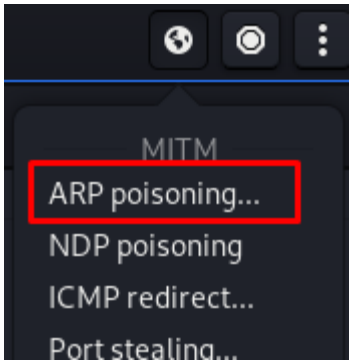
---

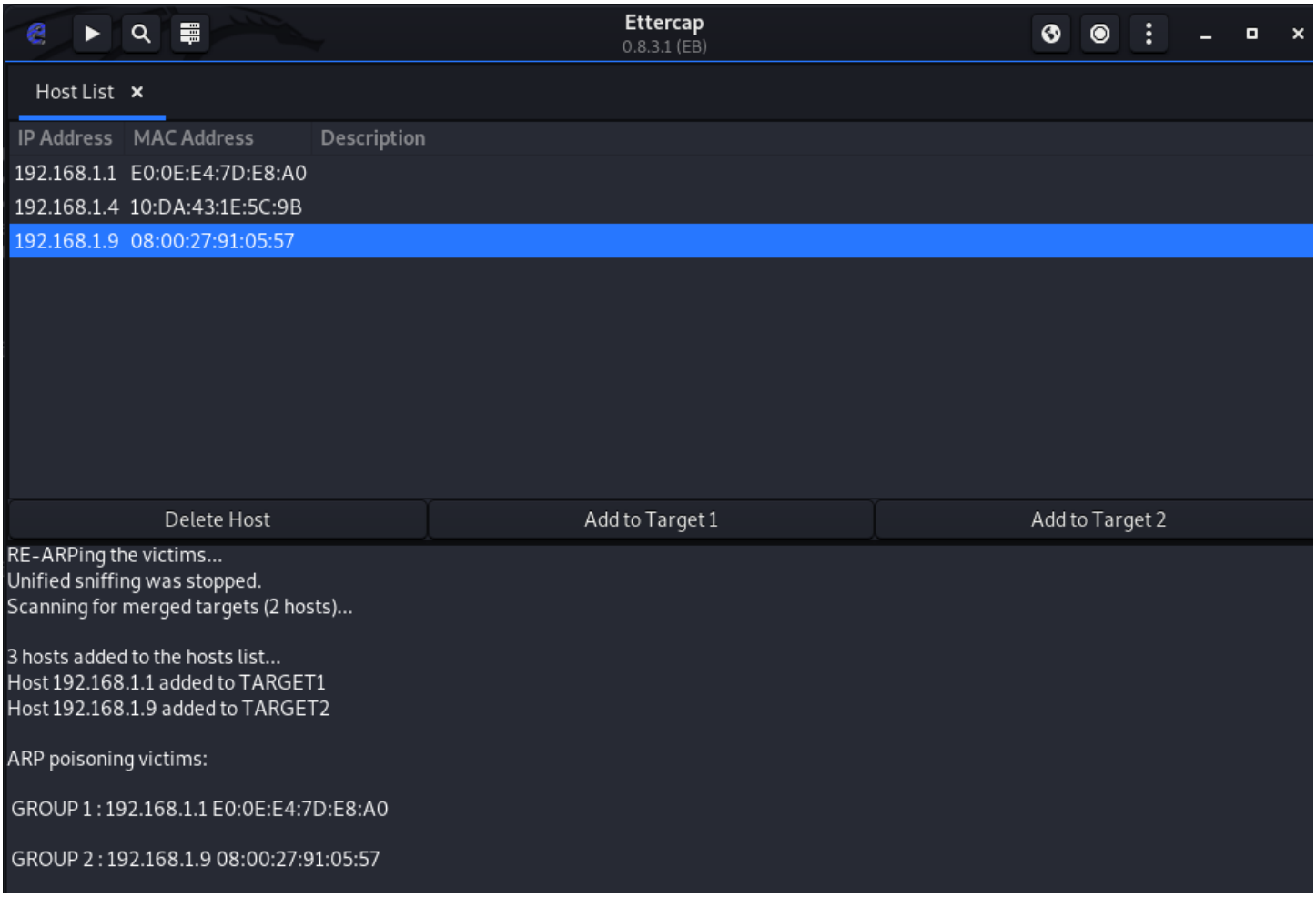
*αν τρέξουμε στο 2ο μηχάνημα την εντολή "ifconfig" βρίσκουμε την διεύθυνση του:



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

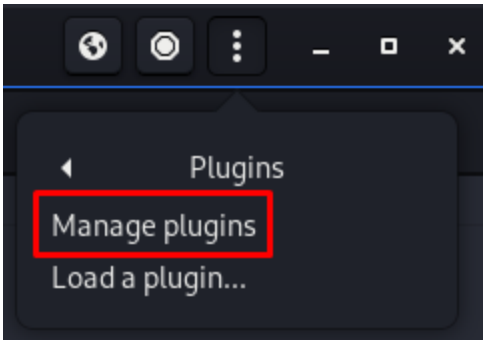
Και παρακάτω βλέπουμε το αποτέλεσμα.

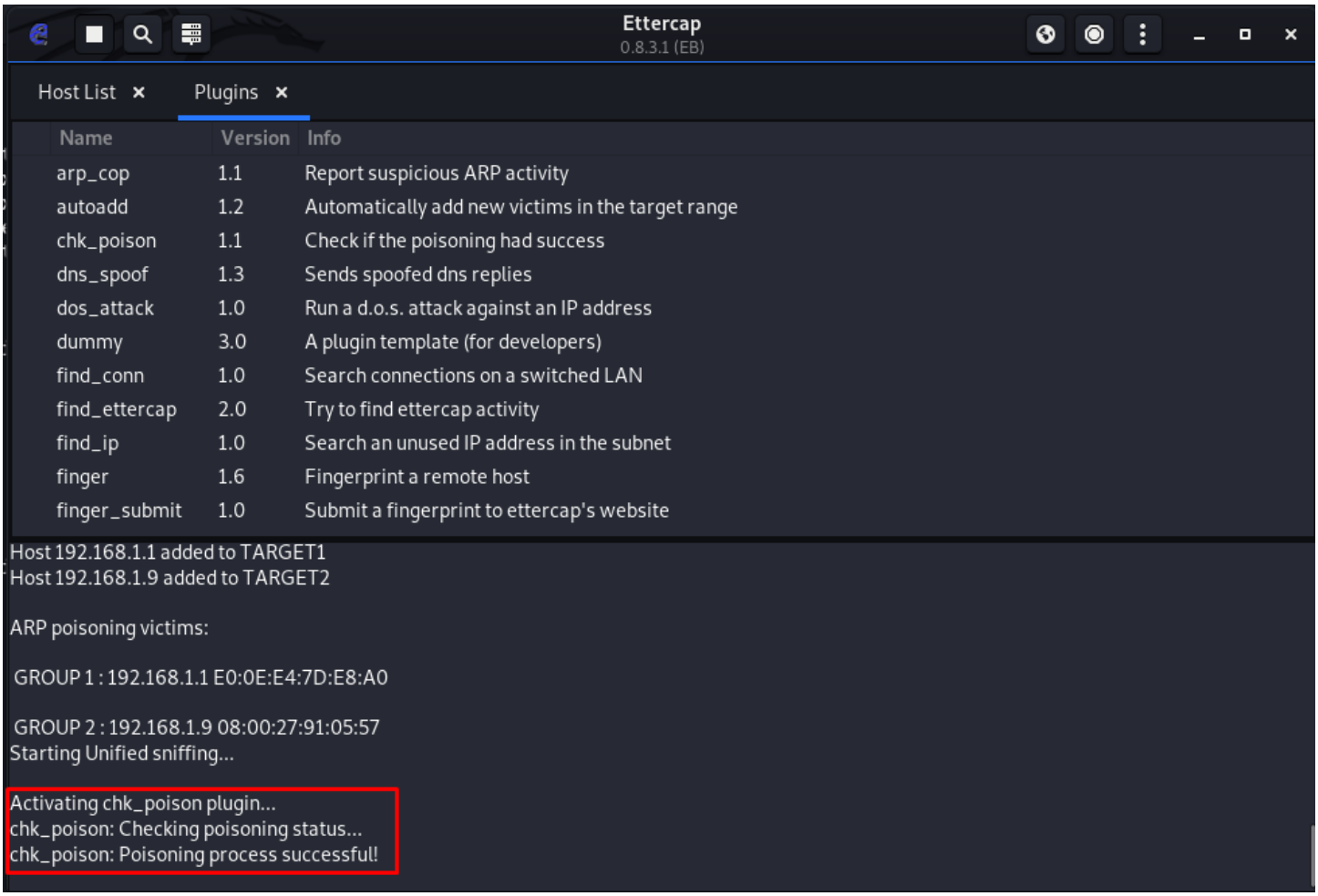




Ανοίγουμε την στήλη με τα plugins, όπως βλέπουμε στην εικόνα δίπλα, και ενεργοποιούμε το “chk_poison” για να δούμε αν έγινε η επίθεση μας.

Όπως φαίνεται στην παρακάτω εικόνα η επίθεση μας έγινε επιτυχώς.



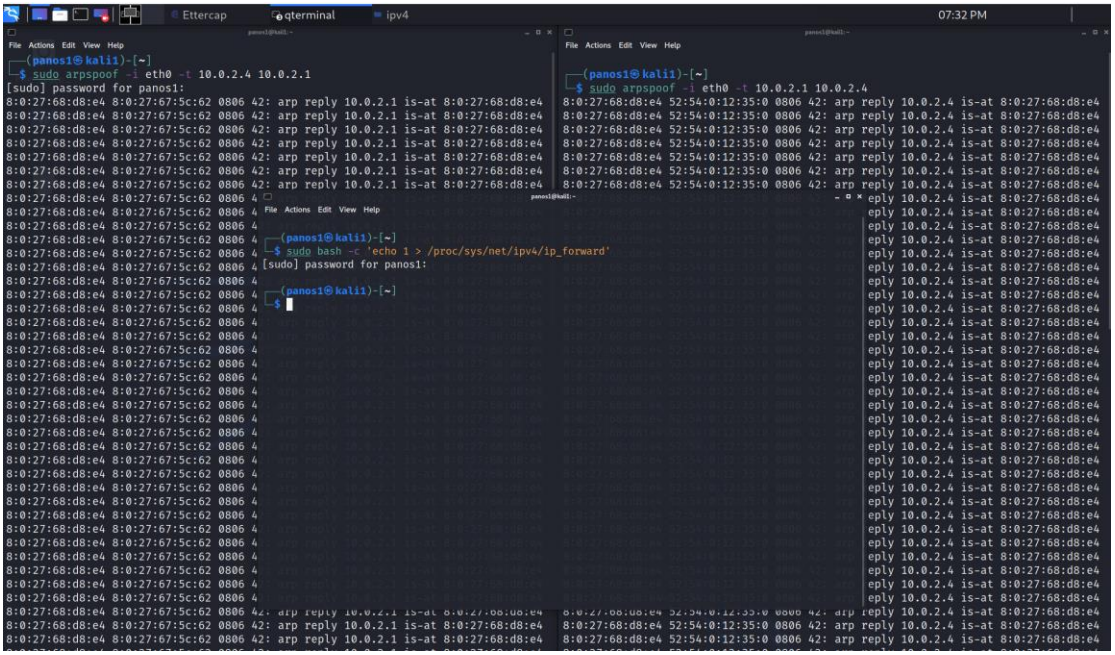


b) Arpspoof.

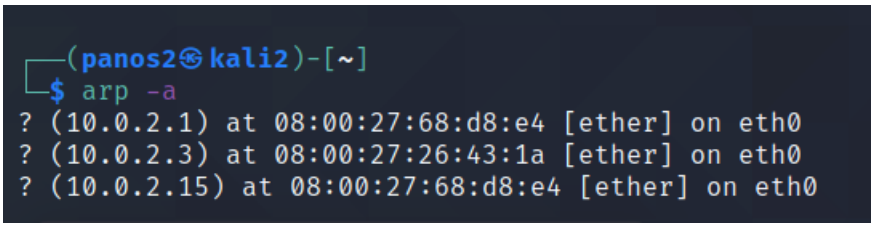
Κάνουμε επίθεση arpspoof εκτελώντας την εντολή arpspoof, ορίζουμε την κάρτα δικτύου στην οποία βρισκόμαστε, στην δικιά μας περίπτωση το eth0 και επίσης ορίζουμε την IP του θύματος η οποία είναι η 10.0.2.4 και την IP του access point η οποία είναι η 10.0.2.1 . Με αυτόν τον τρόπο λέμε στο access point ότι η ζητούμενη IP διεύθυνση έχει την δικιά μας MAC διεύθυνση.

Κάνουμε το ίδιο αλλά αντιστρέφουμε τις θέσεις των δυο αυτών IP λέγοντας τώρα στο θύμα ότι εμείς έχουμε την MAC διεύθυνση του access point.

Τέλος, ενεργοποιούμε το ip forwarding για να έρχεται σε εμάς όλη η κίνηση μεταξύ του θύματος και του access point.



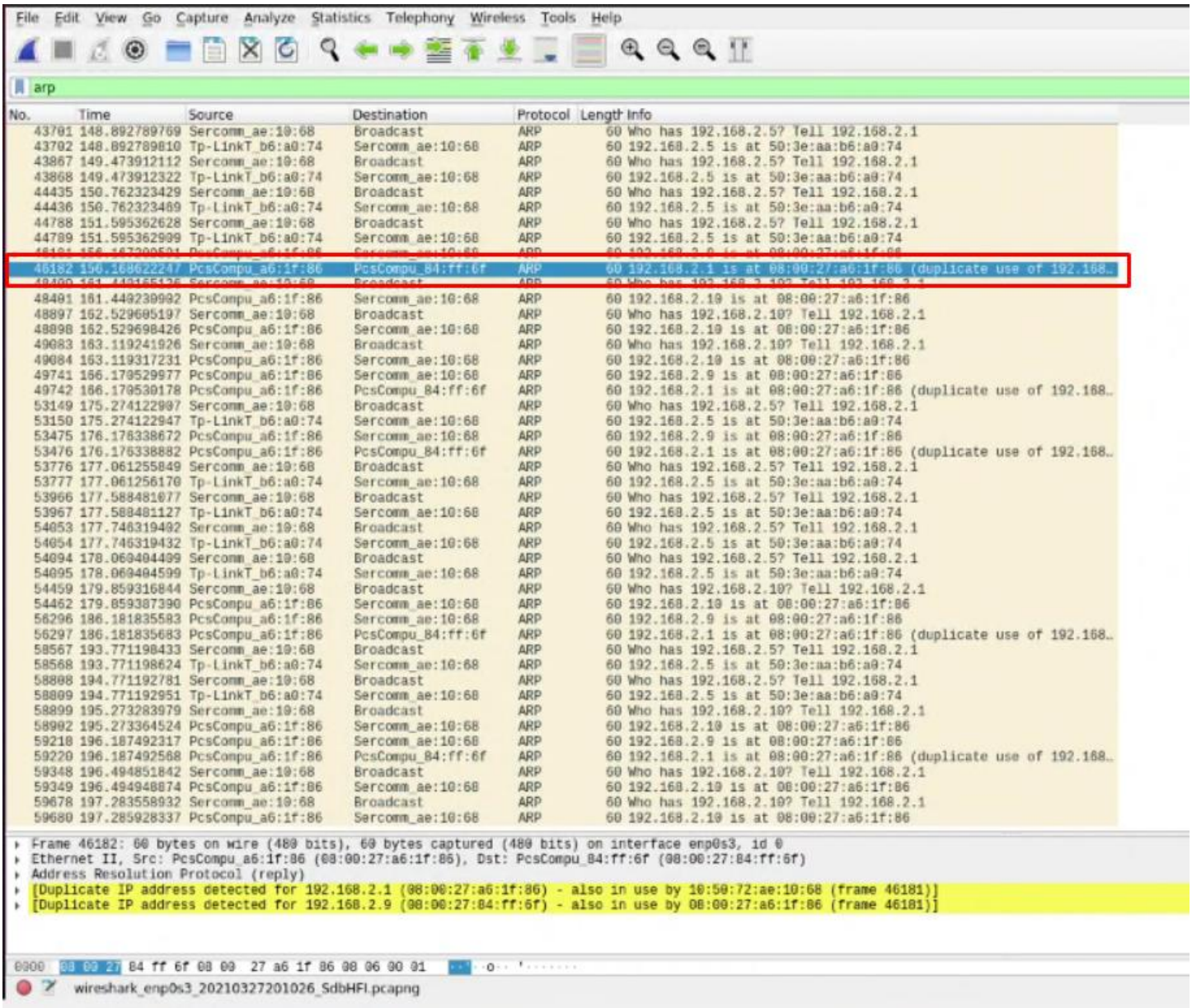
Στο μηχάνημα του θύματος:



2. Να εφαρμόσετε τα ακόλουθα μέτρα ανίχνευσης ή πρόληψης.

a) Ανίχνευση μέσω wireshark.

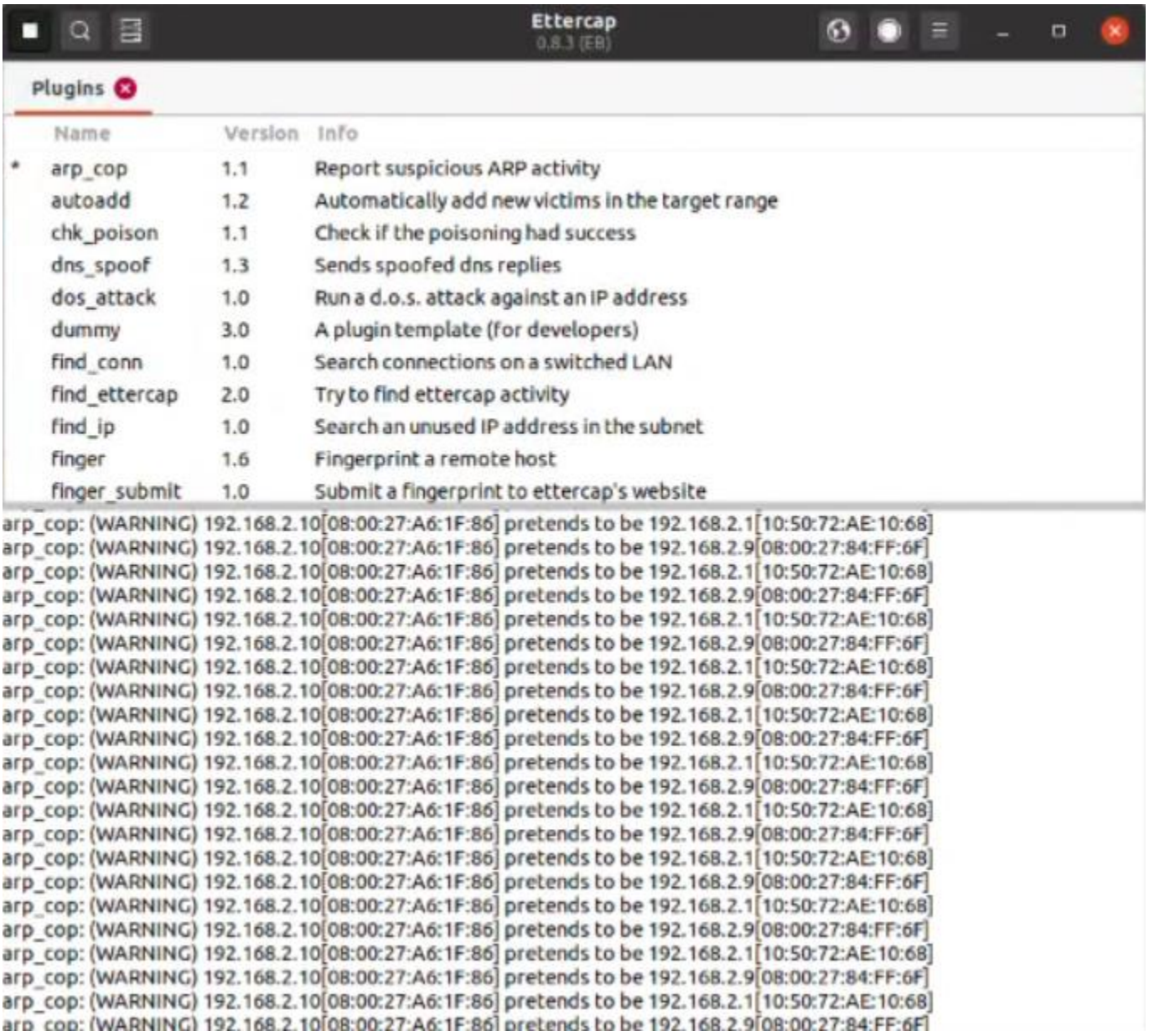
Τώρα στο δεύτερο μηχάνημα στο οποίο κάνουμε την επίθεση, ανοίγουμε το "Wireshark" και τρέχουμε την εντολή "arp". Παρατηρούμε ότι μας βγάζει πάνω από μια φορά την φράση "duplicate use of 192.168.1.9" που είναι η IP του μηχανήματος μας. Αυτό σημαίνει ότι κάποιος χρησιμοποιεί την διεύθυνση μας, δηλαδή ότι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



b) Ανίχνευση μέσω Ettercap.

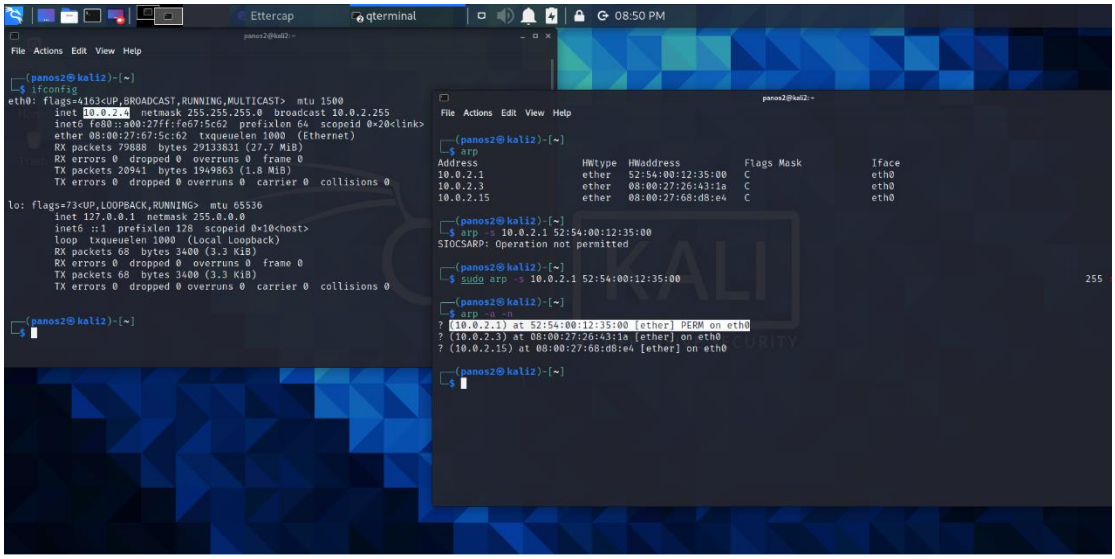
Ανοίγουμε το "Ettercap" στο δεύτερο μηχάνημα, πηγαίνουμε στα "plugins" και ενεργοποιούμε το "arp_cop".

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.

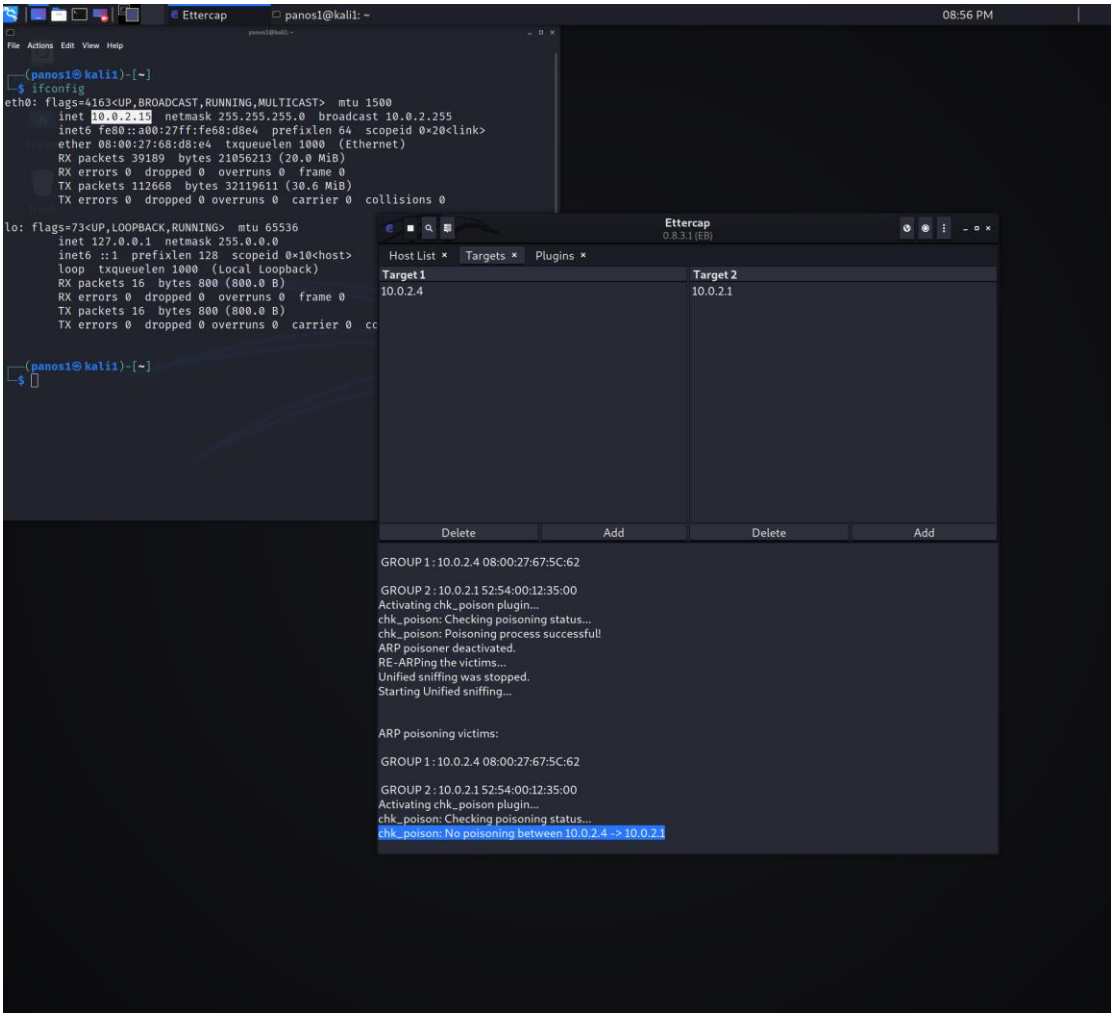


c) Πρόληψη μέσω στατικής ARP.

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.



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.



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.
```

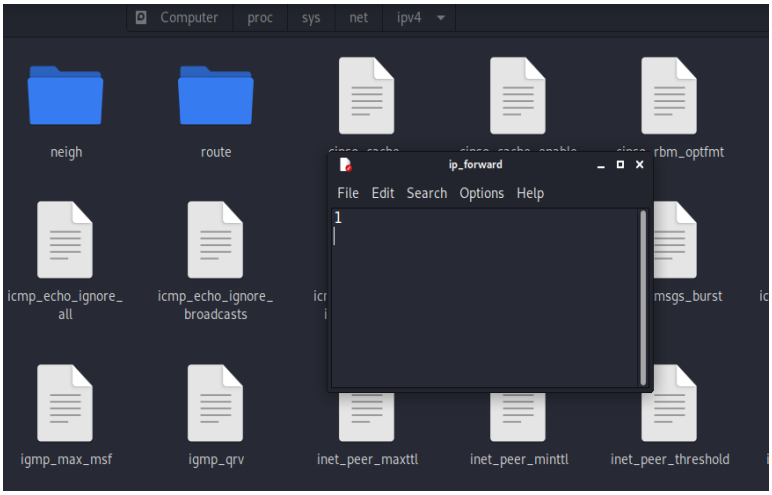
```
┌──(root💀kali)-[/home/kali]
└─# mana

┌──(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 Ctl-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

┌──(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/sslstri
p-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.

┌──(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   nospoofto.cfg  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

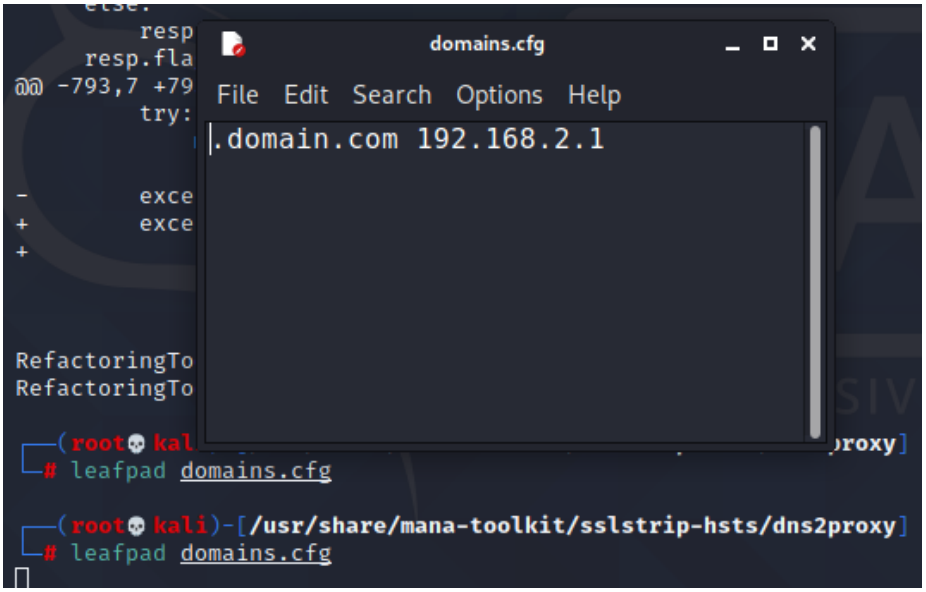
Ρυθμίζουμε τα Ip tables, ώστε να ανακατευθύνουν την κίνηση απ’ την πόρτα 80 στην 9000, καθώς και το firewall για να γίνει ανακατεύθυνση στο dns2proxy.

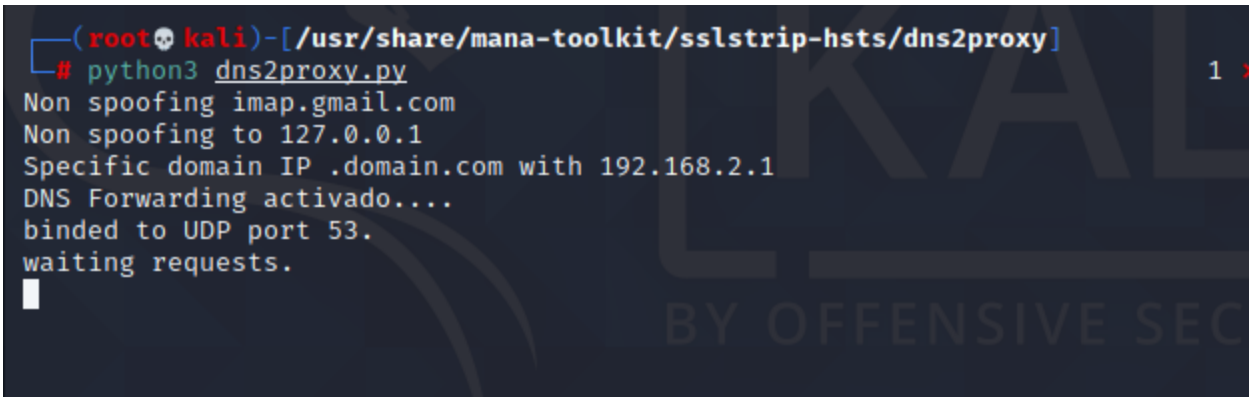
iptables -t nat -A PREROUTING -p tcp –destination-port 80 -j REDIRECT –to-port 9000

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

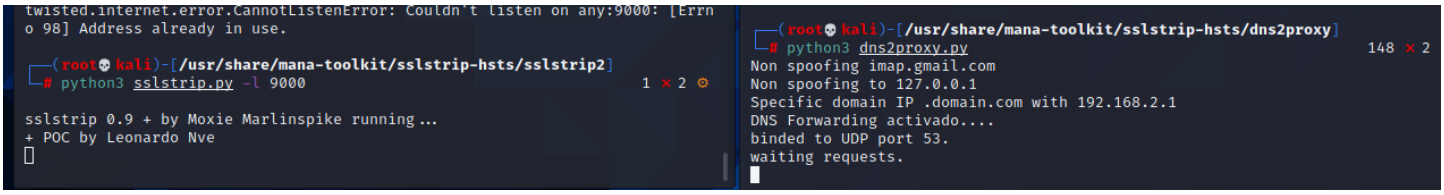
```
root@kali: /home/kali
File  Actions  Edit  View  Help

┌──(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
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.

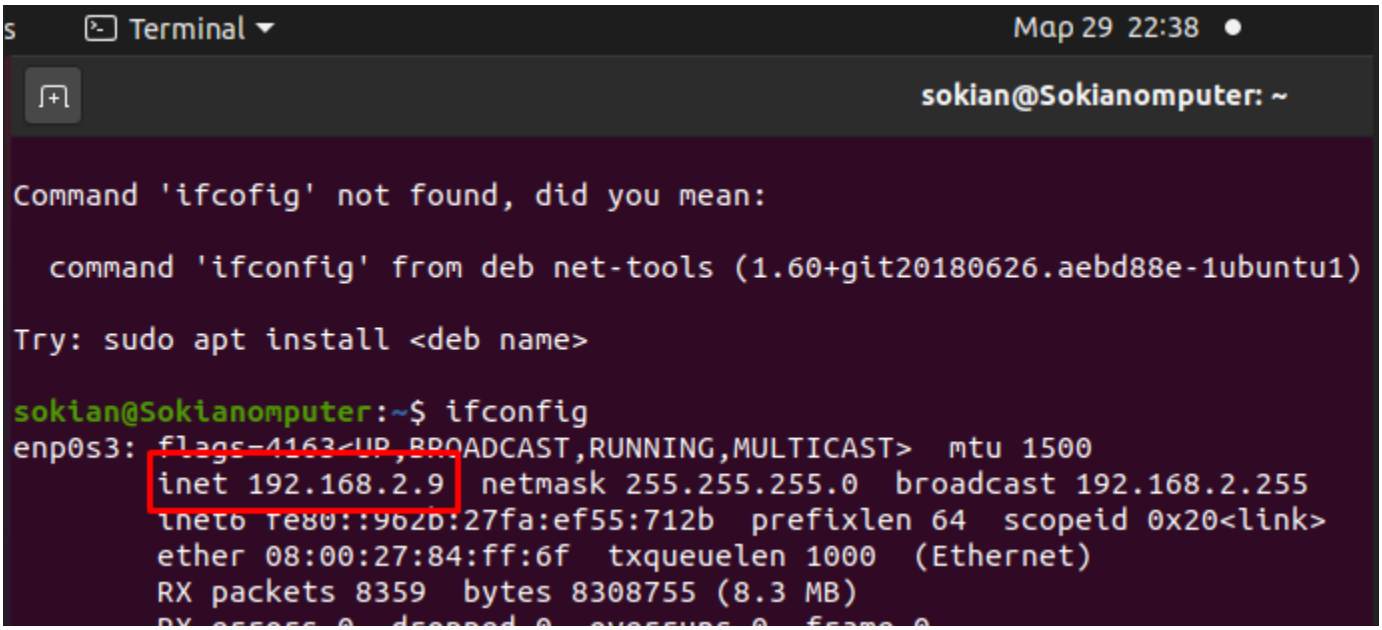


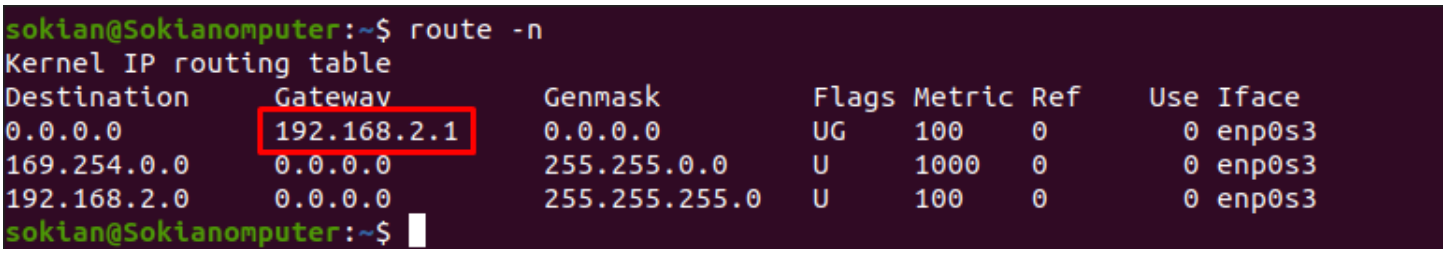


Σε ένα νέο terminal εκκινούμε το sslstrip, το οποίο θα μετατρέψει τα https website, σε http.

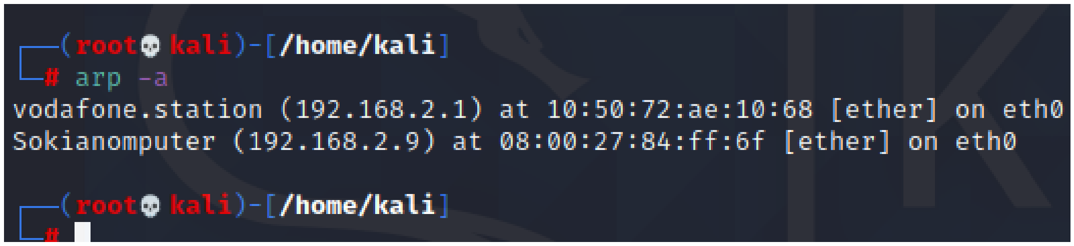


Μεταφερόμαστε στο μηχάνημα του θύματος και βρίσκουμε την IP και Gateway.



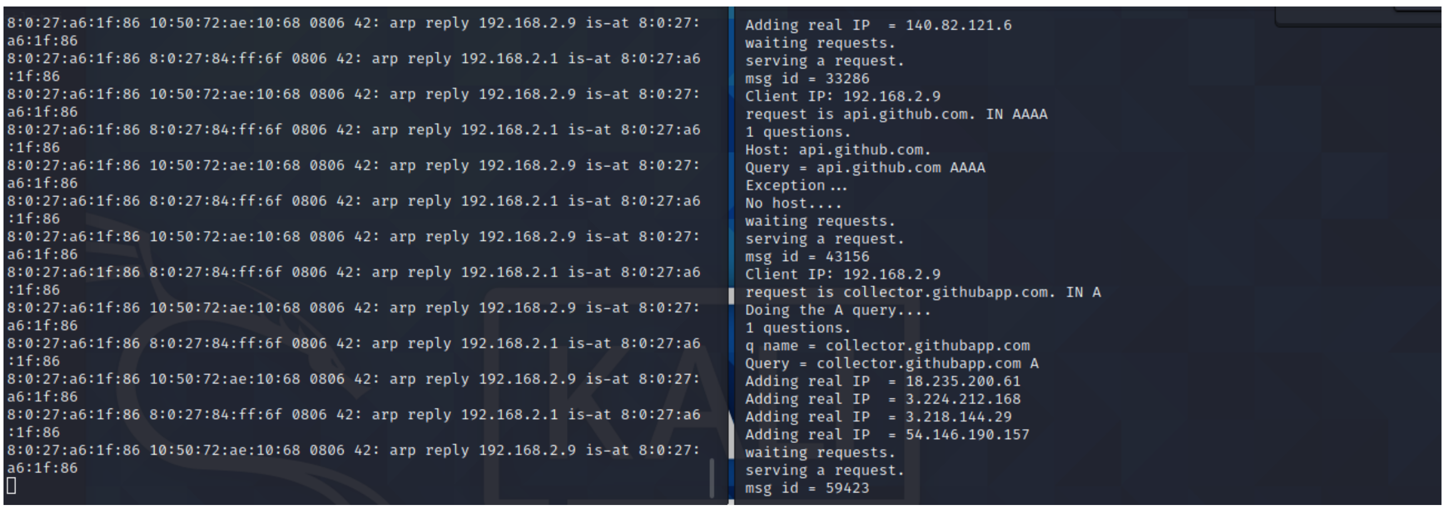


Επιστρέφουμε στον επιτιθέμενο και ξεκινάμε το arpspoof, το οποίο θα "ξεγελάσει" το router και το θύμα. Η κίνηση από τον server προς το θύμα θα περάσει από το μηχάνημά μας και εμείς θα την προωθήσουμε στο θύμα.



Πηγαίνουμε, λοιπόν, ξανά στο μηχάνημα του θύματος και προσπαθούμε να συνδεθούμε σε κάποια σελίδα βάζοντας τα στοιχεία μας, username και password, ώστε να επαληθεύσουμε ότι έγινε η επίθεση.

Στο μηχάνημα του επιτιθέμενου βλέπουμε τα αποτελέσματα:



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.

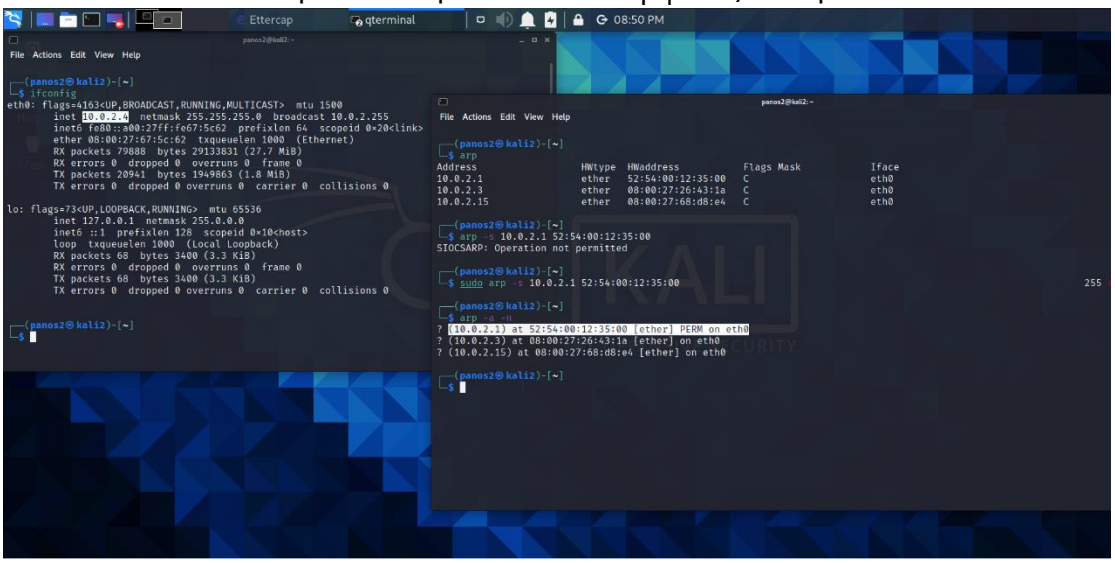
Ένα ARP spoofing, επίσης γνωστό ως ARP spoofing, είναι μια επίθεση “Man in the Middle” που επιτρέπει στους εισβολείς να παρακολουθούν την επικοινωνία μεταξύ συσκευών δικτύου. Η επίθεση λειτουργεί ως εξής:

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.

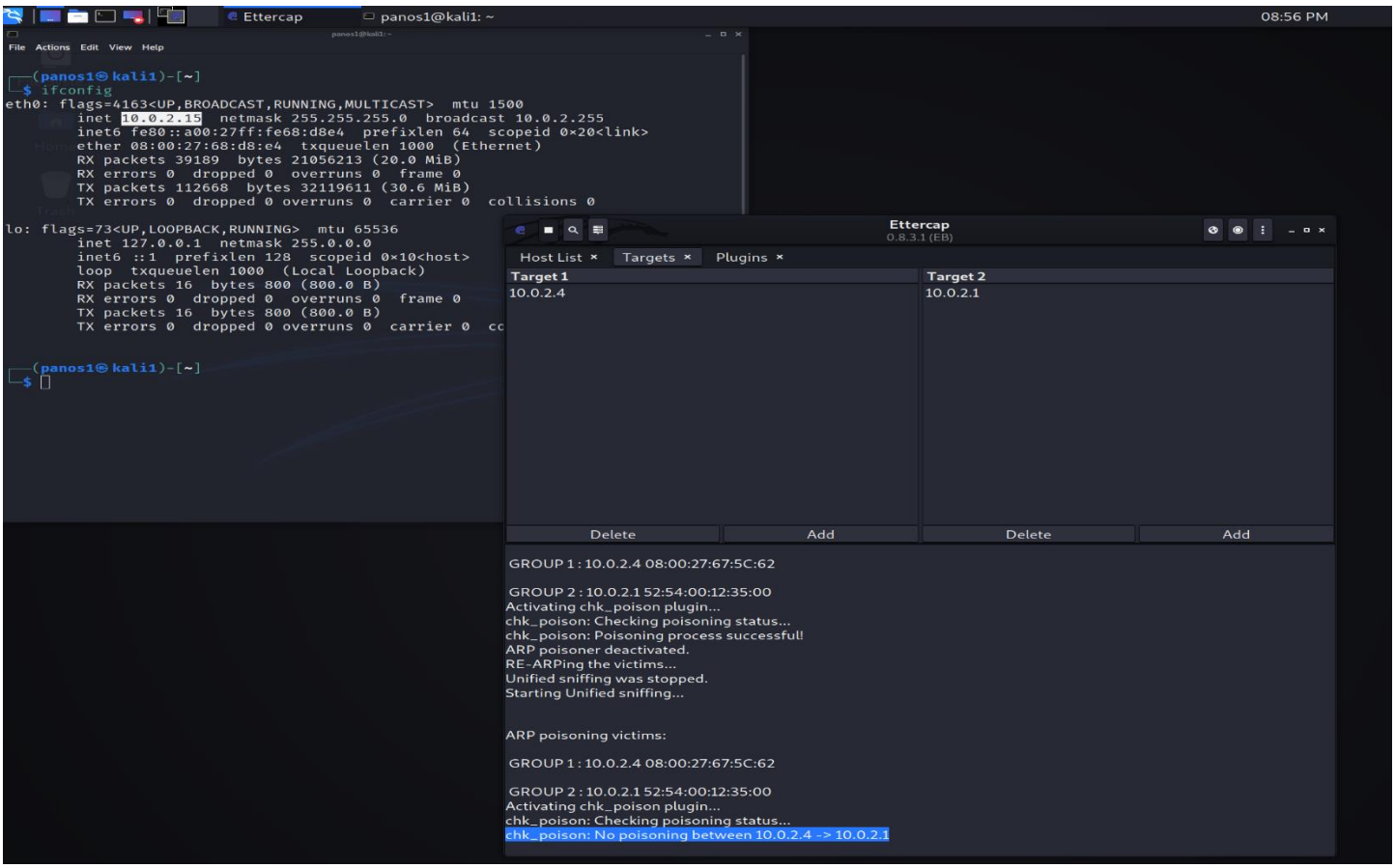
5. Να προτείνετε και να εφαρμόσετε μέτρα προστασίας από την παραπάνω επίθεση.

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.



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.



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.