

LAB 2 Cyber

U7 CYBER

SPINELLI Dylan
BTS SIO SISR | PARIS YNOV CAMPUS

Table des matières

1-	Introduction	2
2-	Empoisonnement du cache ARP avec ARPspooft	3
3-	Capture de trafic http	8
4-	Serveur web local http	10
5-	Serveur web local en HTTPS	12

1-Introduction

Attaque Man in the middle

Un serveur et un client communiquent grâce à un gateway

L'attaquant va se faire passer pour le gateway et recevoir toutes les informations du client et du serveur

Usurpation ARP : une requête ARP sera envoyée pour chercher l'adresse IP du gateway, et l'attaquant va répondre pour se faire identifier comme étant le gateway

Le switch va enregistrer dans sa table l'adresse MAC de l'attaquant et lui envoyer toutes les communications

2-Empoisonnement du cache ARP avec ARPspooof

Voici les machines utilisées pour ce TP :

Machine	Adresse IP	Gateway
Kali	192.168.49.141/24	192.168.49.2
Client Linux	192.168.49.135/24	192.168.49.2
Serveur Linux	192.168.49.142/24	192.168.49.2

Dans le sujet :

.10 client

.20 kali

.254 Gateway

```
kali@kali: ~/Desktop
Session Actions Edit View Help

(kali@kali)-[~/Desktop]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:1e:9d:40 brd ff:ff:ff:ff:ff:ff
    inet 192.168.49.141/24 brd 192.168.49.255 scope global dynamic noprefixroute eth0
        valid_lft 1689sec preferred_lft 1689sec
    inet6 fe80::c283:57b7:330a:9131/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

(kali@kali)-[~/Desktop]
└─$ ip route show
default via 192.168.49.2 dev eth0 proto dhcp src 192.168.49.141 metric 100
192.168.49.0/24 dev eth0 proto kernel scope link src 192.168.49.141 metric 100

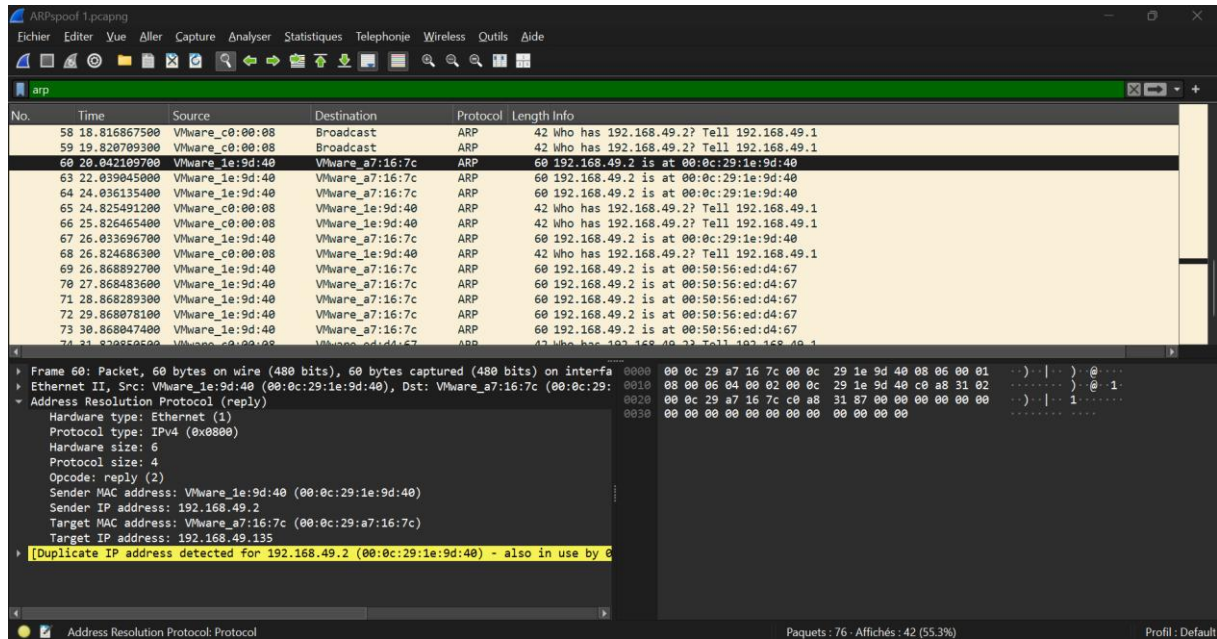
(kali@kali)-[~/Desktop]
└─$
```

```
debian@dylan:~
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
    valid_lft forever preferred_lft forever
:: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
ip default qlen 1000
link/ether 00:0c:29:a7:16:7c brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 192.168.49.135/24 brd 192.168.49.255 scope global dynamic noprefixroute ens33
    valid_lft 1693sec preferred_lft 1693sec
inet6 fe80::20c:29ff:fea7:167c/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
lebian@dylan:~$ arp -a
bash: arp : commande introuvable
lebian@dylan:~$ arp
bash: arp : commande introuvable
lebian@dylan:~$ arp ?
bash: arp : commande introuvable
lebian@dylan:~$ ip route show
default via 192.168.49.2 dev ens33 proto dhcp src 192.168.49.135 metric 100
192.168.49.0/24 dev ens33 proto kernel scope link src 192.168.49.135 metric 100
lebian@dylan:~$

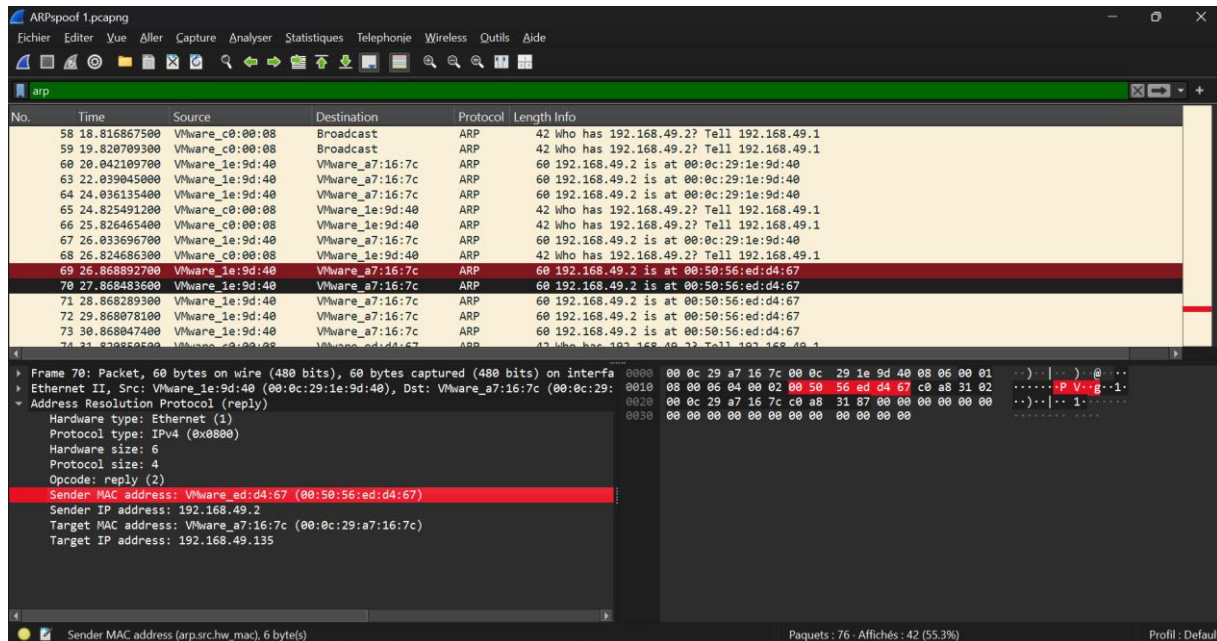
debian@dylan:~
debian@dylan:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:51:b3:8d brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.49.142/24 brd 192.168.49.255 scope global dynamic noprefixroute ens33
        valid_lft 1706sec preferred_lft 1706sec
    inet6 fe80::20c:29ff:fe51:b38d/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
debian@dylan:~$ arp
bash: arp : commande introuvable
debian@dylan:~$ ip route show
default via 192.168.49.2 dev ens33 proto dhcp src 192.168.49.142 metric 100
192.168.49.0/24 dev ens33 proto kernel scope link src 192.168.49.142 metric 100
debian@dylan:~$
```


Analysons le trafic sur wireshark

On voit que lors des premiers échanges, l'adresse MAC du gateway 192.168.49.2 est 00:0c:29:1e:9d:40



Puis ensuite, une fois l'usurpation ARP effectuée, l'adresse MAC correspondant au gateway est désormais 00:50:56:ed:d4:67



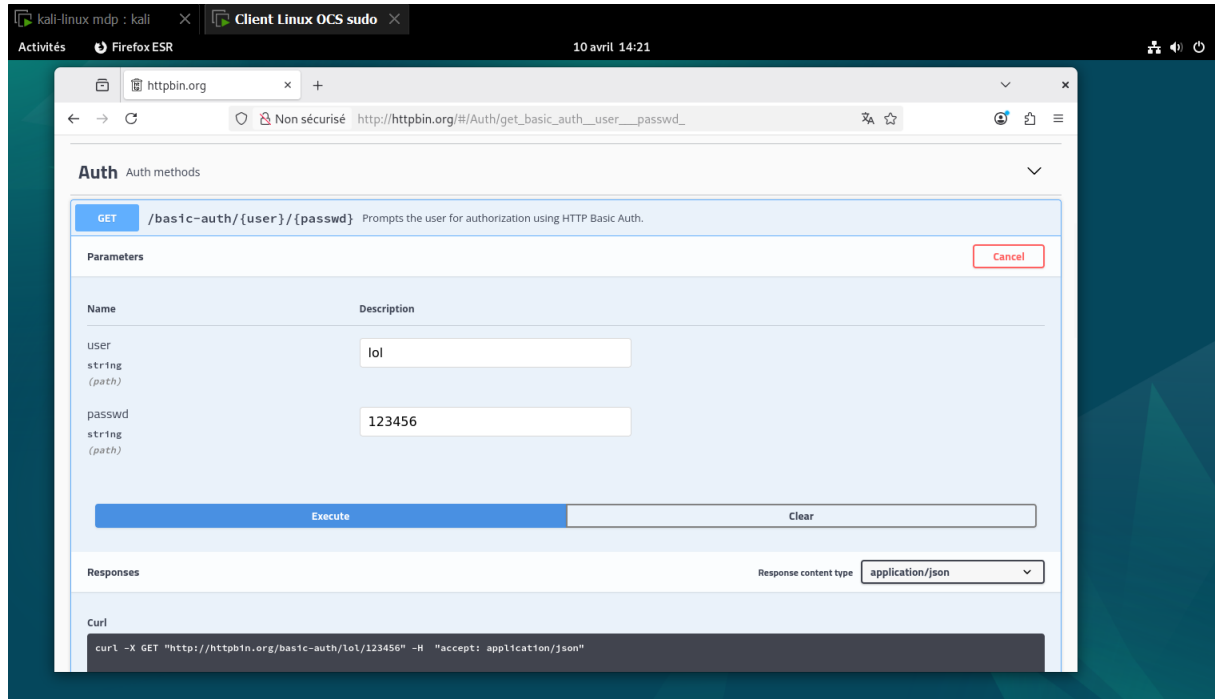
Dans ce TP, les adresses MAC sont renseignées de manière aléatoire. Dans un cas concret, c'est l'adresse MAC de l'attaquant qui serait renseignée à la place de l'adresse MAC de départ. De plus, chaque adresse MAC est unique et il est impossible d'avoir 2 adresses MAC identique dans un réseau. L'attaquant devra donc d'abord modifier l'adresse MAC du client puis remonter ensuite au serveur pour finir par le Gateway.

Le Gateway est la cible finale car tout le trafic sur le réseau passe par le Gateway.

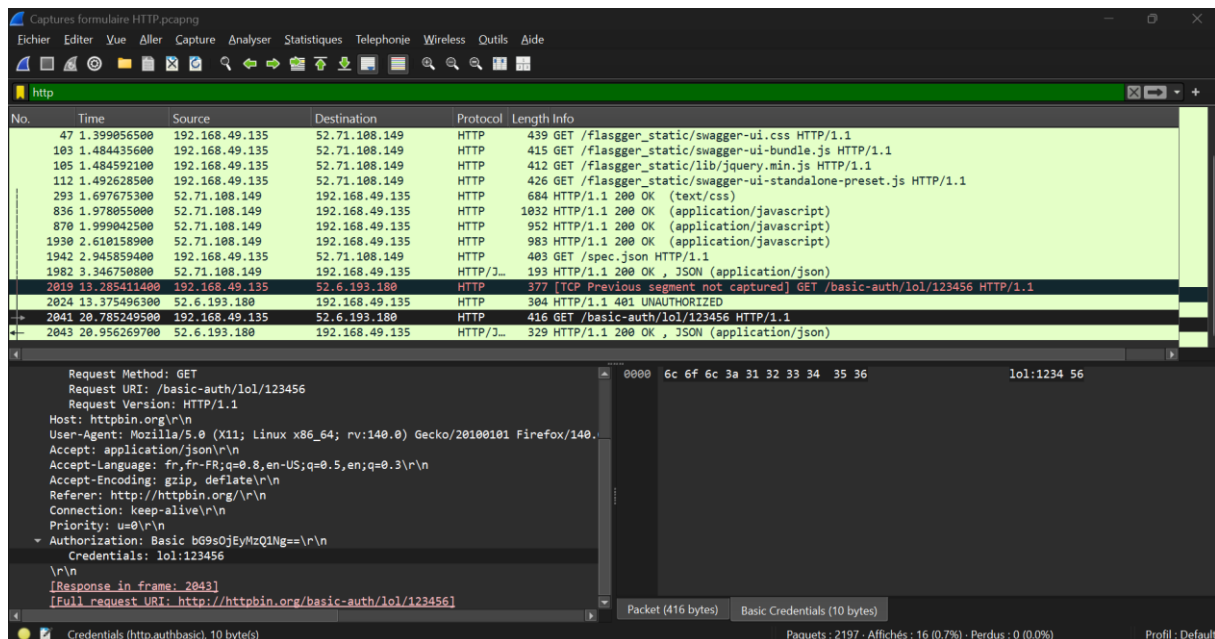
3-Capture de trafic http

On cherche à trouver les informations de connexion d'un formulaire http

Pour cela nous allons nous connecter à un site web vulnérable <http://httpbin.org/>



Analysons ensuite le trafic http pour voir si on retrouve bien nos logins :

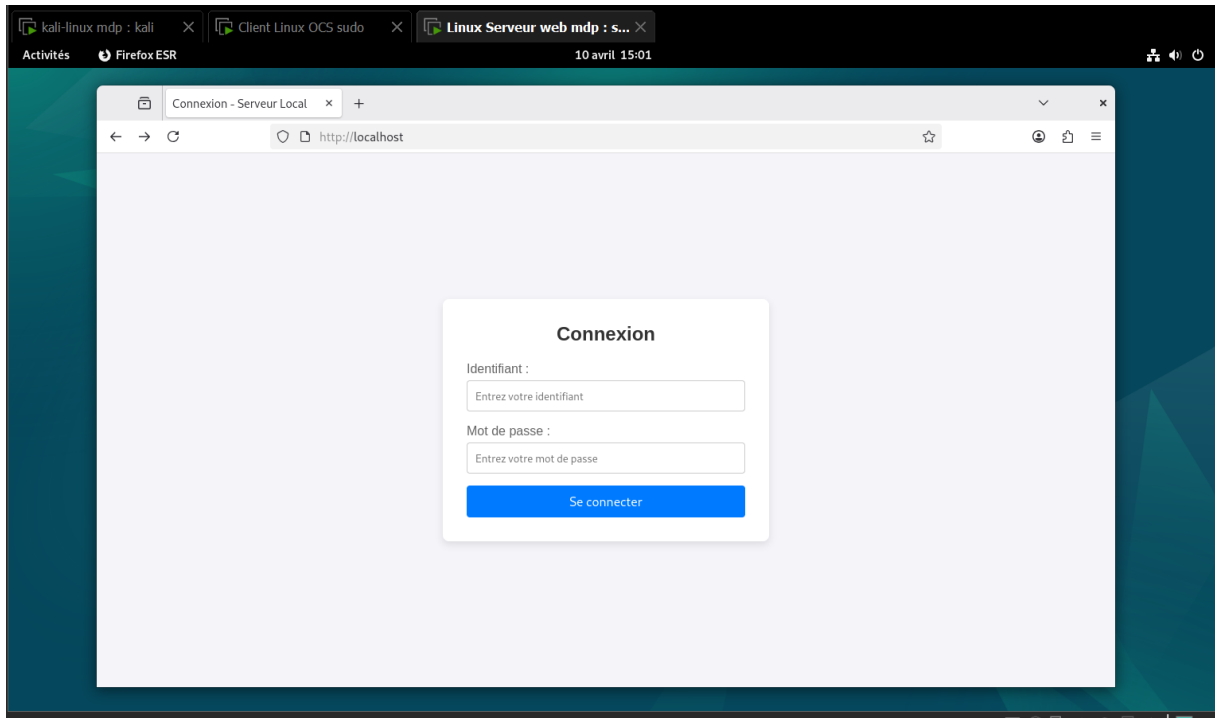


Les logins apparaissent bien en clair dans le trafic. Le protocole http n'est pas sécurisé. Il faut utiliser HTTPS afin que ces informations soient chiffrées.

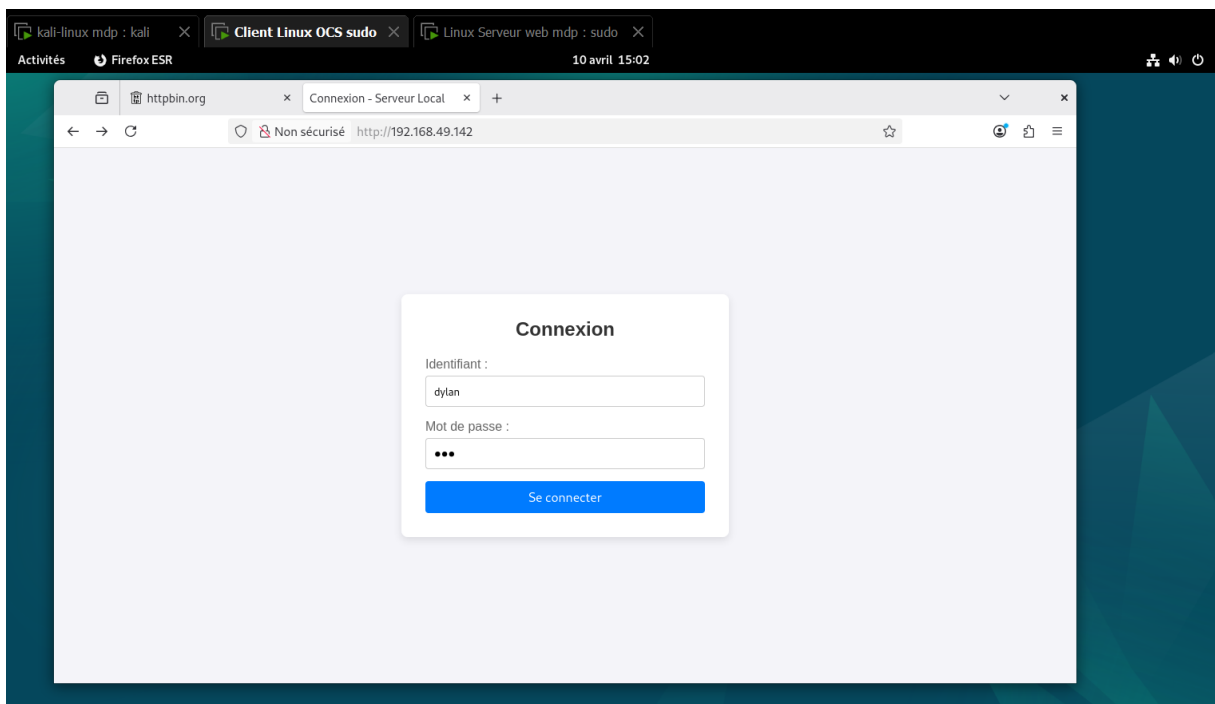
Un attaquant pourra toujours récupérer ces données, mais elles seront chiffrées et illisibles.

4-Serveur web local http

On crée un formulaire http avec un serveur web local apache sur une machine virtuelle :



Connectons-nous maintenant depuis notre client qui a été attaqué :



L'attaquant peut intercepter les informations envoyées par le client au serveur :

The screenshot shows a Wireshark interface with a network traffic capture. The main pane displays a list of captured packets. The selected packet (No. 10) is an HTTP POST request. The packet details pane shows the following information:

- Frame 10: Packet, 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits) on interface 0
- Ethernet III, Src: Vmware_a7:16:7c (08:00:2b:16:7c), Dst: Vmware_51:b3:8d (08:00:27:51:b3:8d)
- Internet Protocol Version 4, Src: 192.168.49.135, Dst: 192.168.49.142
- Transmission Control Protocol, Src Port: 49390, Dst Port: 80, Seq: 1, Ack: 1, Len: 590
- Hypertext Transfer Protocol
- HTML Form URL Encoded: application/x-www-form-urlencoded
 - Form item: "login" = "dylan"
 - Form item: "mdp" = "123"

The packet bytes pane shows the raw data of the request, including the form data: `guage: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3;Accept-Encoding:gzip,deflate;Content-Type: application/x-www-form-urlencoded;Content-Length: 19;Origin: http://192.168.49.142;Connection: keep-alive;Referer: http://192.168.49.142/;Upgrade-Insecure-Requests: 1;Priority: u=0, i=;login=dylan&mdp=123`

On retrouve bien le login et le mdp dans le trafic

5-Serveur web local en HTTPS

Nous allons maintenant sécuriser le serveur web en HTTPS

Pour cela, on installe SSL sur notre serveur et on crée les clés de sécurité ainsi qu'un certificat auto-signé :

```
debian@dylan: /var/www/html
debian@dylan: /var/www/html$ sudo a2enmod ssl
[sudo] Mot de passe de debian :
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create
elf-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
debian@dylan: /var/www/html$ sudo systemctl restart apache2
debian@dylan: /var/www/html$ sudo openssl req -x509 -nodes -days 365 -newkey rsa
2048 \
-keyout /etc/ssl/private/serveur-local.key \
-out /etc/ssl/certs/serveur-local.crt
```

On renseigne les nouvelles clés de sécurité dans la configuration SSL :

```
GNU nano 7.2 /etc/apache2/sites-available/default-ssl.conf *
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf

#
# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
SSLEngine on

#
# A self-signed (snakeoil) certificate can be created by installing
# the ssl-cert package. See
# /usr/share/doc/apache2/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only the
# SSLCertificateFile directive is needed.
SSLCertificateFile /etc/ssl/certs/serveur-local.crt
SSLCertificateKeyFile /etc/ssl/private/serveur-local.key

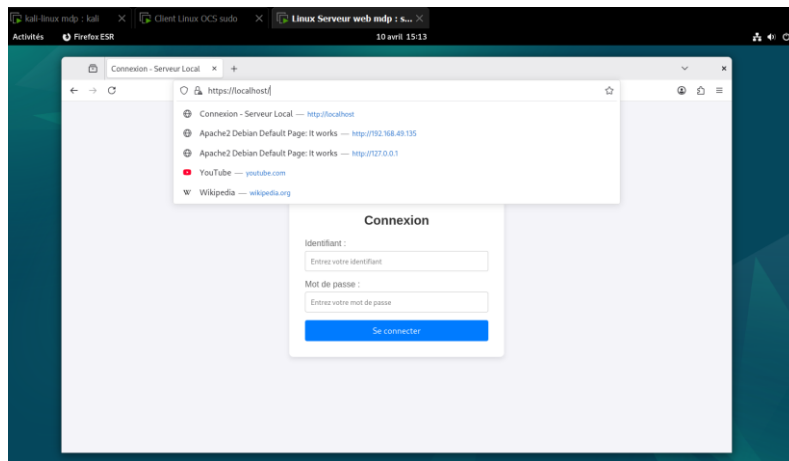
#
# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
```

```
G Aide      ^O Écrire   ^W Chercher  ^K Couper    ^T Exécuter  ^C Emplacement
X Quitter   ^R Lire fich ^M Remplacer ^U Coller    ^J Justifier ^_ Aller liene
```

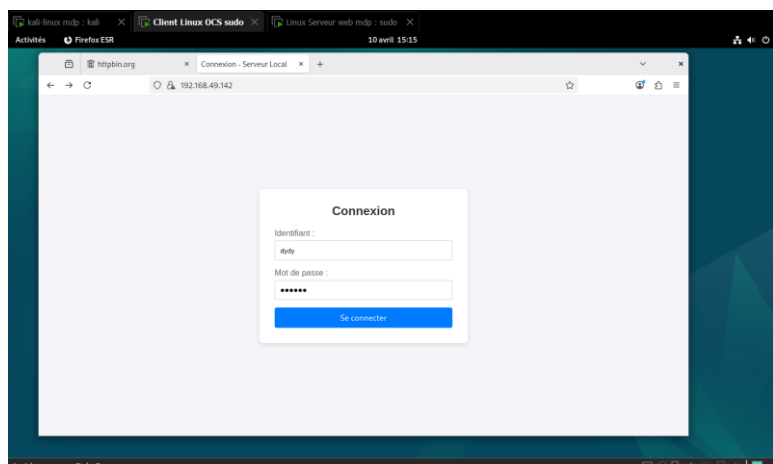
On termine en activant le site HTTPS et on relance apache :

```
-----
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.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
debian@dylan: /var/www/html$
debian@dylan: /var/www/html$ sudo nano /etc/apache2/sites-available/default-ssl.c
onf
debian@dylan: /var/www/html$ sudo a2ensite default-ssl.conf
Enabling site default-ssl.
To activate the new configuration, you need to run:
    systemctl reload apache2
debian@dylan: /var/www/html$ sudo systemctl reload apache2
debian@dylan: /var/www/html$ █
```

Notre serveur est maintenant en HTTPS :



Connectons-nous depuis le client et analysons le trafic depuis notre attaquant :



Le trafic sur wireshark est désormais en TLS :

The screenshot shows the Wireshark interface with a capture from the eth0 interface. The packet list pane shows several packets, with packet 4 selected. The packet details pane shows the following structure:

- Frame 4: Packet, 2081 bytes on wire (16648 bits), 2081 bytes captured (16648 bits) on interface eth0
- Ethernet II, Src: VMware_a7:16:7c (08:0c:29:a7:16:7c), Dst: VMware_51:b3:8d (08:00:0e:51:b3:8d)
- Internet Protocol Version 4, Src: 192.168.49.135, Dst: 192.168.49.142
- Transmission Control Protocol, Src Port: 34732, Dst Port: 443, Seq: 1, Ack: 1, Len: 2080
- Transport Layer Security
 - Stream index: 0
 - TLSv1.3 Record Layer: Handshake Protocol: Client Hello
 - Content type: Handshake (22)
 - Version: TLS 1.0 (0x0301)
 - Length: 2010
 - Handshake Protocol: Client Hello

The packet bytes pane shows the raw hex and ASCII data for the selected packet, including the TLS Client Hello structure.

Les communications sont désormais cryptées. L'attaquant pourra toujours les intercepter mais il ne pourra pas les utiliser car elles seront illisibles.