MISE EN ŒUVRE DE SSL,WIRESHARK, IPTABLES,FAIL2BAN

TABLE DES MATIERES

PREREQUIS :

Savoir installer un serveur web et PHP

CONDITION DE REALISATION :

2 machines virtuelles Debian 11 sur le même réseau

OBJECTIFS :

Création d'un site web de test

Voir la différence de sécurité avant et après l'installation de SSL et de l'utilisation de certificats autosignés.

Étude des trames avec Wireshark Installation d'Iptables.

Création de scripts exécutables

Installation et étude de Fail2Ban

PREPARATION DE LA PREMIERE MACHINE VIRTUELLE.

La première machine virtuelle fera office de machine bureautique basique. Son nom sera **vmclient**. Installez un bureau léger de type xfce (https://www.xfce.org/)

CHANGEMENT DU NOM DE L'ORDINATEUR

Vous changerez le nom de la machine si nécessaire :

Passez en super-Utilisateur avec la commande **su**

```
jlb@vmclient:~$ su
Mot de passe :
root@vmclient:/home/jlb# nano /etc/hosts
```

Puis lancez l'éditeur Nano

nano /etc/hosts

La machine s'appellera **vmclient**

Ctrl+O et **Ctrl+X** pour sauvegarder et quitter

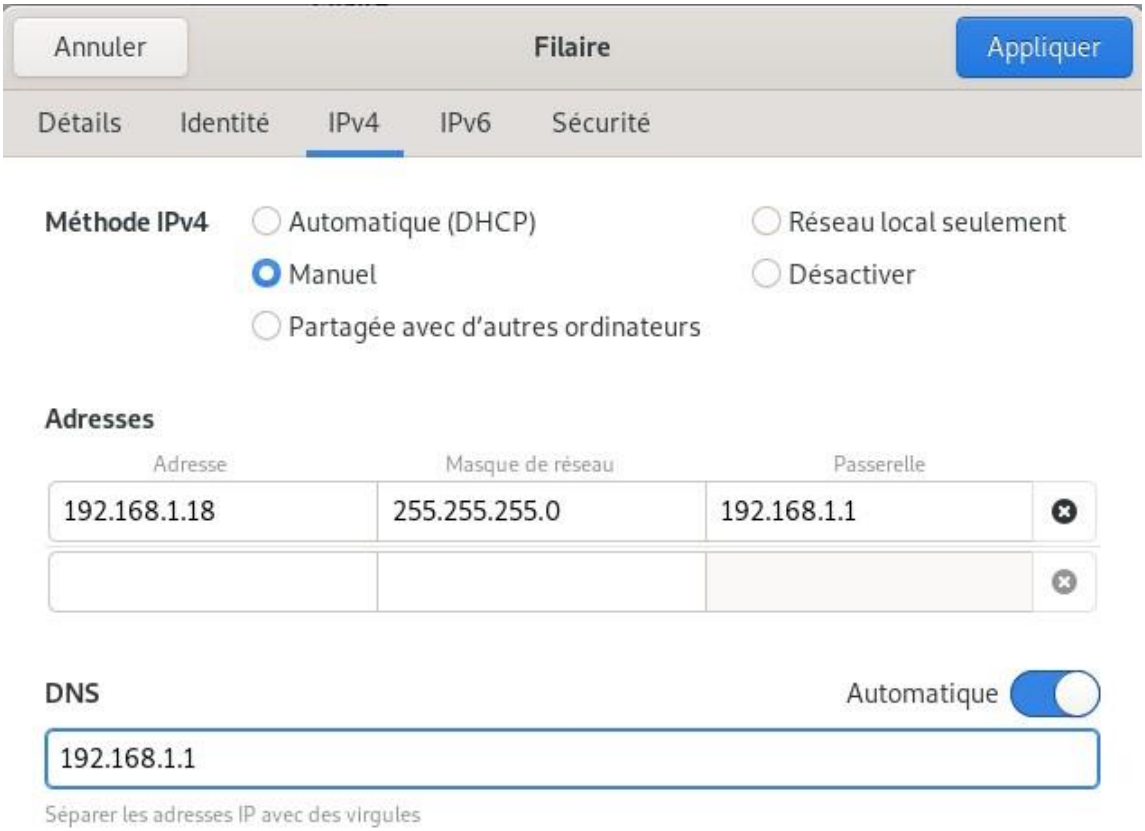
Il est ensuite peut-être nécessaire de modifier le fichier **/etc/hostname** et de remplacer son contenu par le nouvel hostname :

nano /etc/hostname

Ctrl+O et **Ctrl+X** pour sauvegarder et quitter.

CHANGEMENT DE L'ADRESSE IP

Pour fixer l'adresse Ip, c'est ici. Dans mon cas l'Ip sera 192.168.1.18



Redémarrer votre vm pour que les modifications soient prises en compte.

Vous terminerez avec un snapshot.

PREPARATION DE LA DEUXIEME MACHINE VIRTUELLE

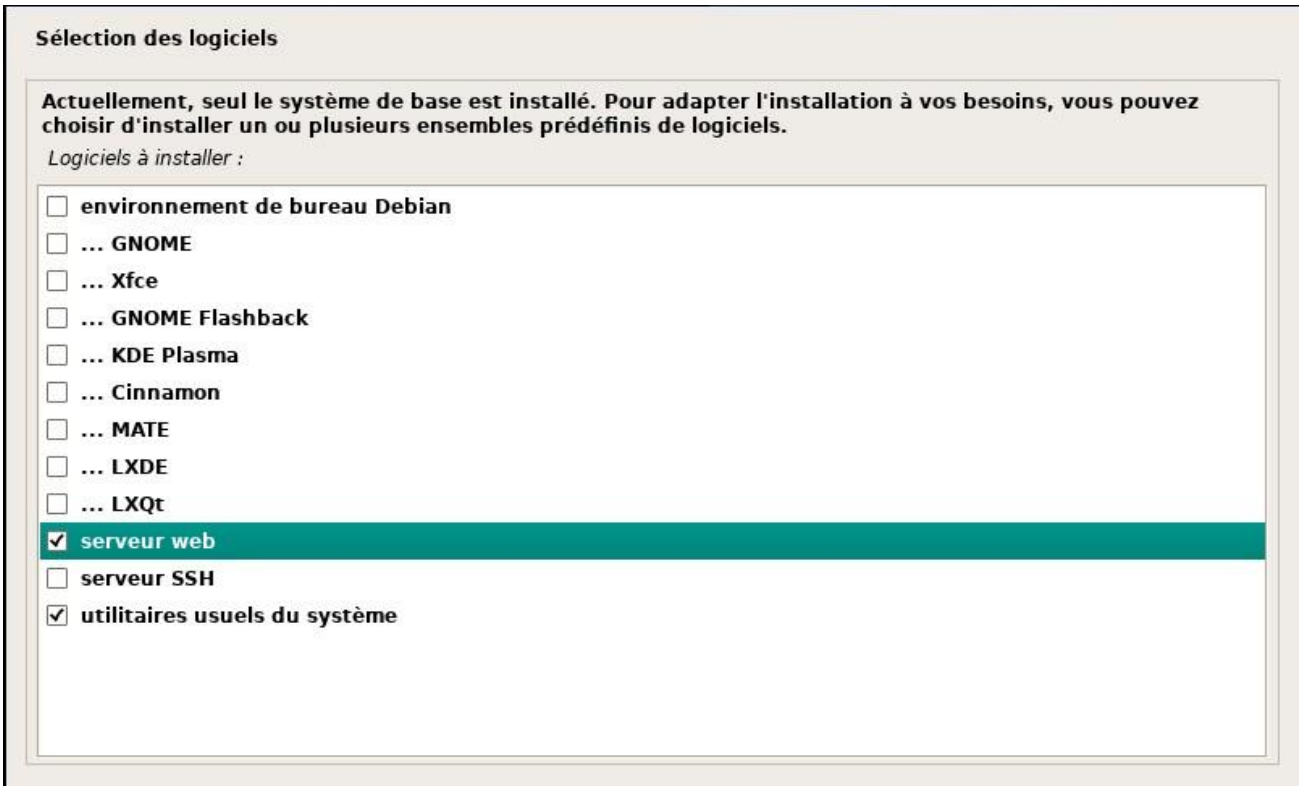
La deuxième machine virtuelle sera une machine sous Debian avec le bureau **Xfce** (pour vous aider a créer les scripts et modifier les fichiers de config) vous supprimerez le bureau à la fin du TP pour réduire la surface d'attaque.

Installez aussi l'éditeur **BlueFish**.

En fait, c'est une Vm identique à la précédente. Vous pouvez faire un clone de la VmCliente, mais vous devrez changer le nom de la machine, ses adresses IP….

Ici, le nom de la machine sera **vmserveur**

À la sélection des logiciels veillez à cocher comme ci-dessous :



Il n'y aura donc pas de bureau et le serveur web sera installé par défaut

Vérifiez donc le fichier **hosts** et le fichier **hostname** en conséquence (si nécessaire)

L'adresse IP dans mon cas : 192.168.1.25

Par défaut php7 est installé. Par contre le code PHP n'est pas interprété par défaut. Utilisez la commande suivante en mode **SU -**:

```
apt install libapache2-mod-php
```

CREATION DES PAGES WEB DANS LE DOSSIER /VAR/WWW/HTML

INDEX.HTML

Vous allez créer en premier le fichier suivant que vous nommerez index.html

```
<p>
    Cette page ne contient que du HTML.<br />
    Veuillez taper votre nom :
</p>

<form action="cible.php" method="post">
<p>
    <input type="text" name="nom" />
    <input type="submit" value="Valider" />
</p>
</form>
```

CIBLE.PHP

Le second fichier (cible.php) contient le code suivant :

```
<p>Bonjour !</p>

<p>Vous vous appelez <?php echo htmlspecialchars($_POST['nom']); ?> !</p>

<p><a href="index.html">cliquez ici</a> pour revenir à la page d'accueil.</p>
```

Les 2 fichiers devront être positionnés dans le dossier /var/www/html N'oubliez pas de prendre un Snapshot

TEST DE FONCTIONNEMENT DU SITE

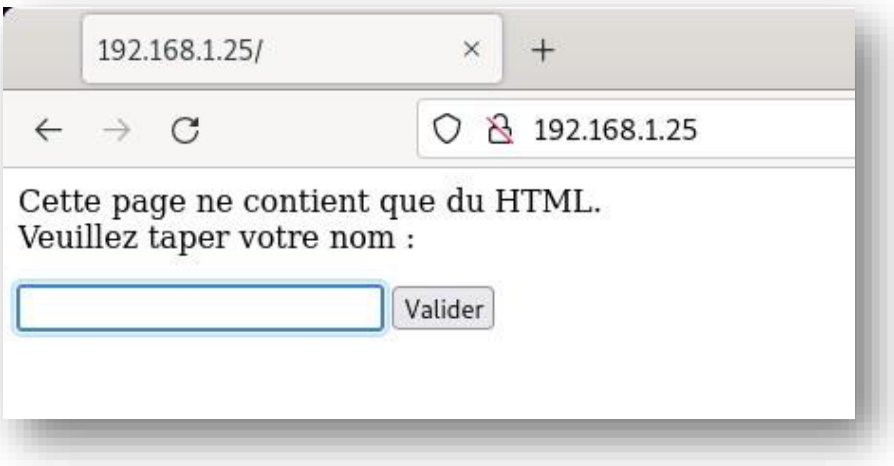
Sur Debian 11, la commande ifconfig n'existe plus, pour connaître l'adresse IP de votre poste veuillez saisir :

Ip addr

POSITIONNEZ-VOUS SUR LA MACHINE VIRTUELLE CLIENTE

Sur votre machine virtuelle cliente lancez Firefox et saisissez l'adresse IP de votre Vm serveur.

Si les 2 vm sont sur le même réseau, vous devriez voir le site apparaître.



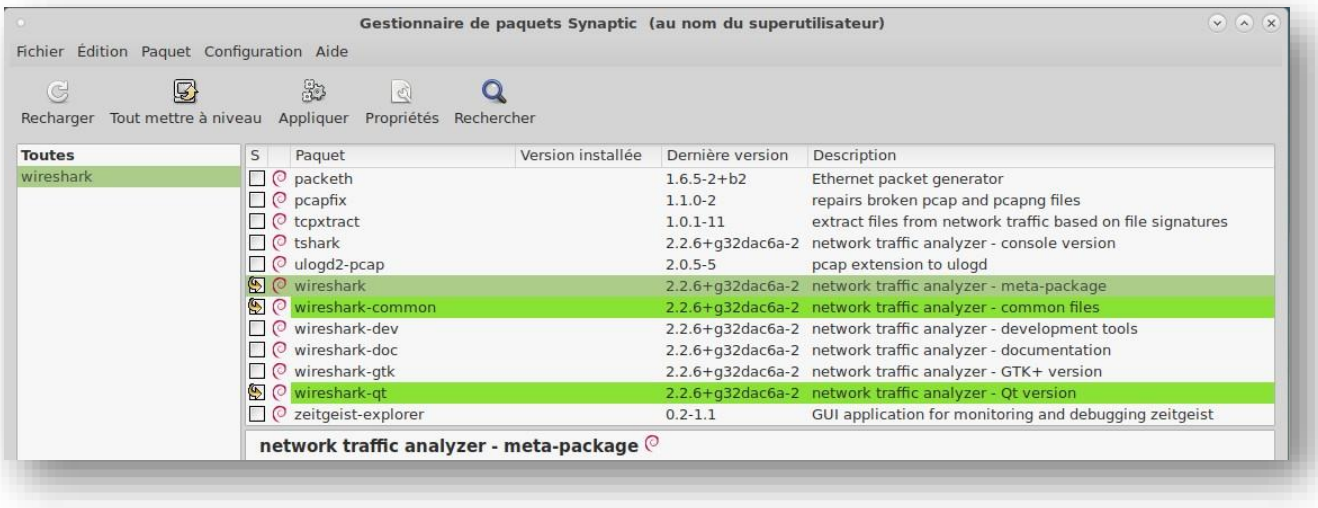
Sinon, contrôlez que les 2 vm se répondent au ping, que les pages du site soient au bon endroit… Bref, cherchez votre erreur.

Si vous tapez votre nom, vous devriez voir comme ci-dessous :

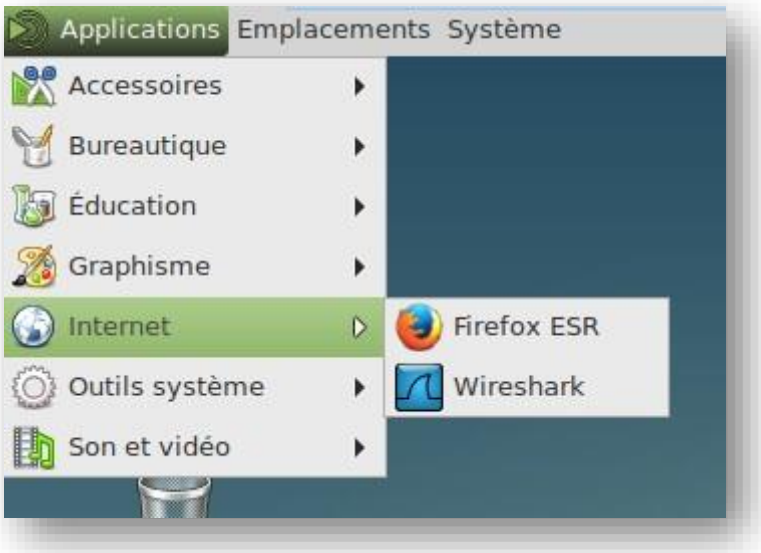
Comme vous le voyez, le script en Php a bien fonctionné, mon nom apparait bien. Comme vous pouvez le constater, les transactions ne sont pas sécurisées, car le cadenas est barré.

ÉTUDE DES TRAMES AVEC WIRESHARK

Pour installer Wireshark, allez dans le gestionnaire de paquet Synaptic



Une fois installé Wireshark sera placé dans la rubrique Internet du menu Application.



Lancez Wireshark

Si vous voyez le message d'erreur ci-dessus, il va falloir que vous donniez les droits à l'utilisateur sous lequel vous êtes logué, d'utiliser Wireshark.

Commencez par vous ajouter au groupe des super-utilisateurs :

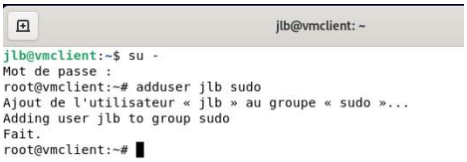
Taper :

Su -

Saisissez le mot de passe du root Ensuite :

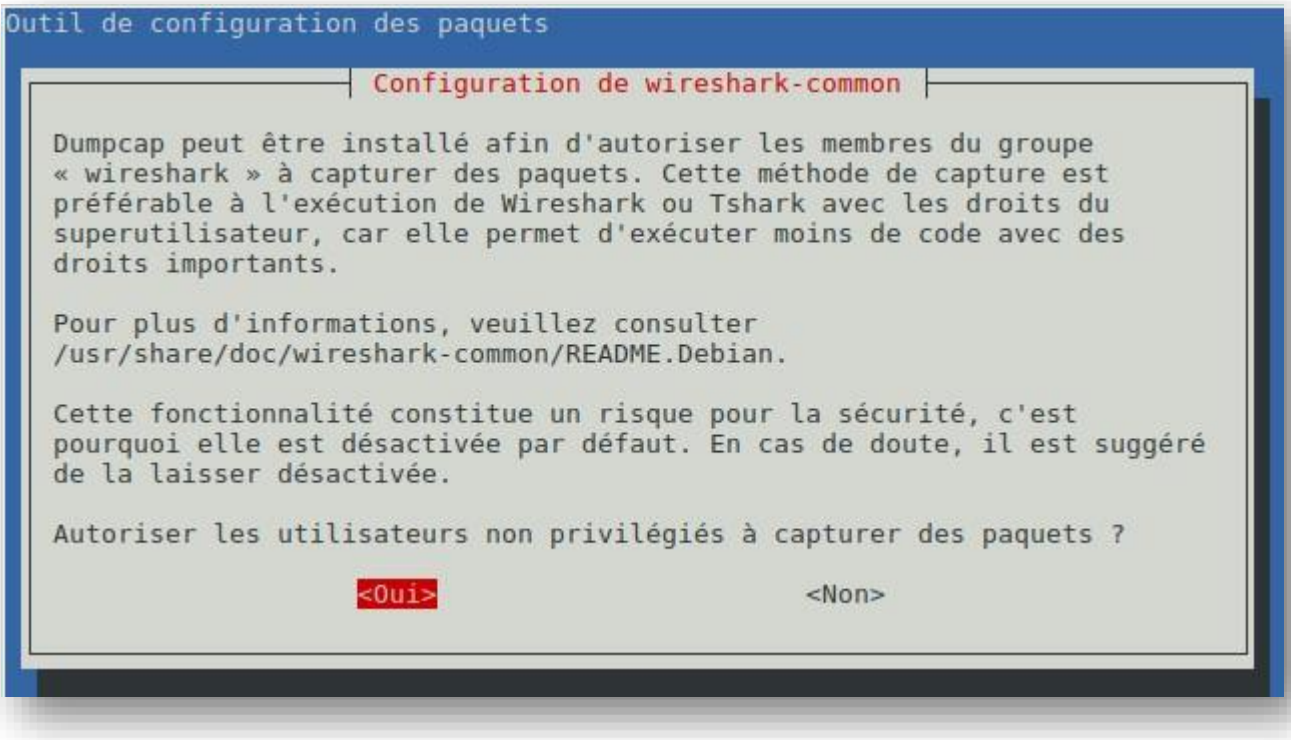
adduser « votre login » sudo

Par exemple :



Ensuite, saisissez :

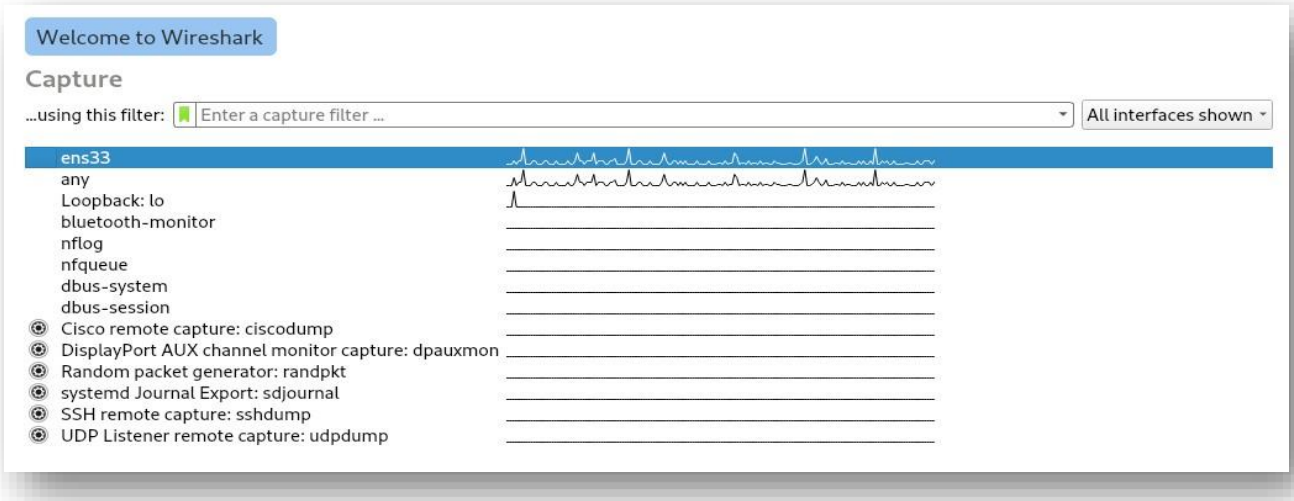
apt-get install libpcap-dev dpkg-reconfigure wireshark-common



Répondez **Oui** à la question, les utilisateurs non root pourront ainsi capturer des paquets. Ensuite nous ajoutons l'utilisateur au groupe Wireshark

gpasswd -a « votre login » wireshark

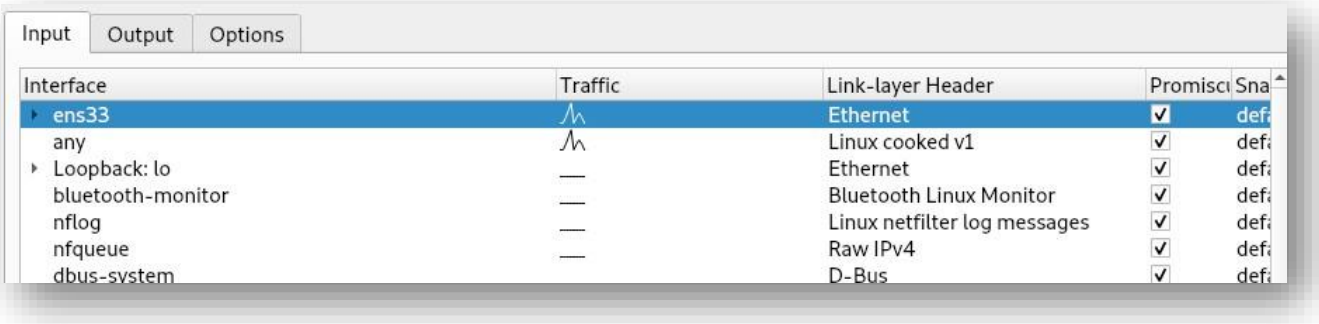
Voilà, le problème est résolu !



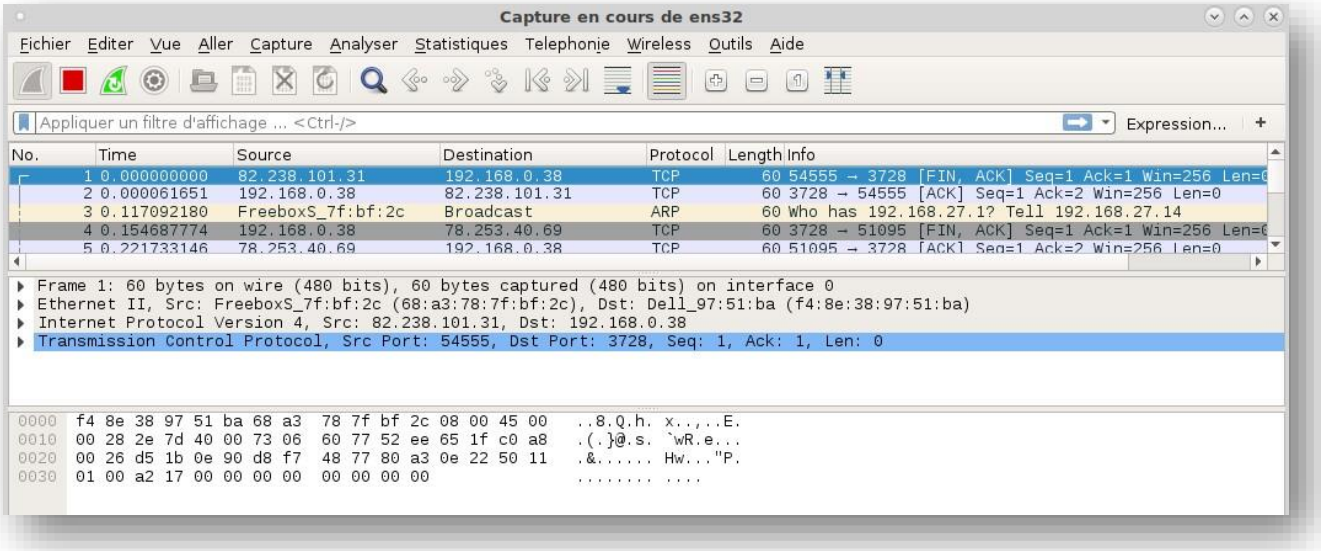
CAPTURE DE TRAMES

Pour commencer la capture, il faut indiquer à Wireshark quelle interface il doit écouter.

Dans le menu Capture, cliquez sur Options

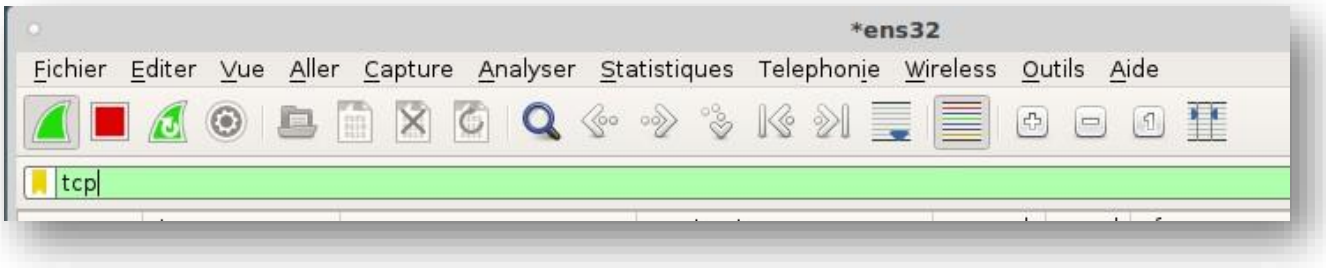


Choisissez l'interface ens33 (possible que votre interface ait un autre numéro), puis cliquez sur le bouton Démarrer.



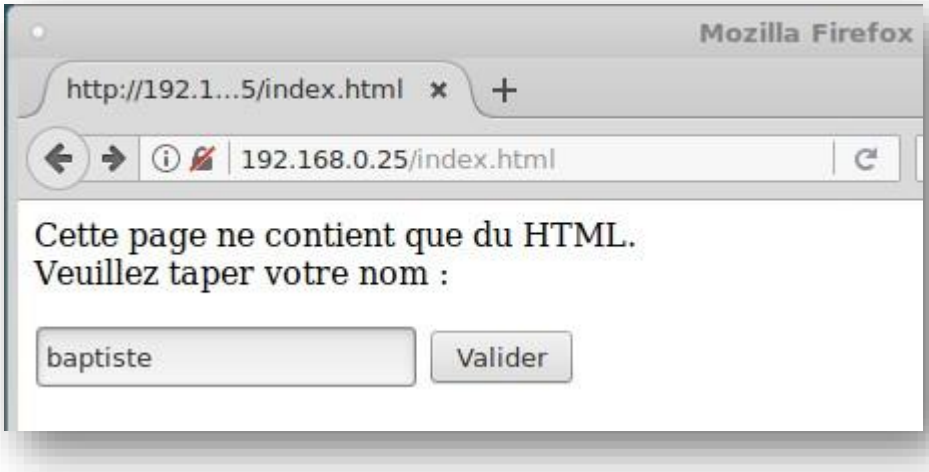
Wireshark est maintenant prêt à écouter toutes les trames. Nous allons filtrer juste les trames **TCP**.

Dans la zone Appliquer un filtre d'affichage saisissez TCP. Le bandeau devient vert.



Arrêtez les captures en cours si nécessaire (**Menu Capture-> Arrêter**).

Lancez Firefox et connectez-vous au serveur pour pouvoir afficher votre site



Ne cliquez pas encore sur le bouton Valider.

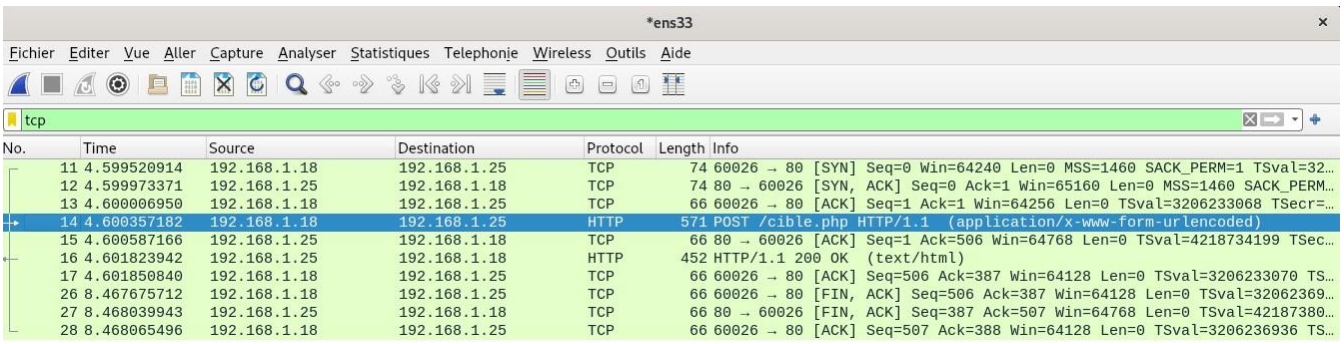
Sur Wireshark, lancez la capture de trames (Menu Capture->Démarrer).

Maintenant que Wireshark et démarré, cliquez sur le bouton valider du site.

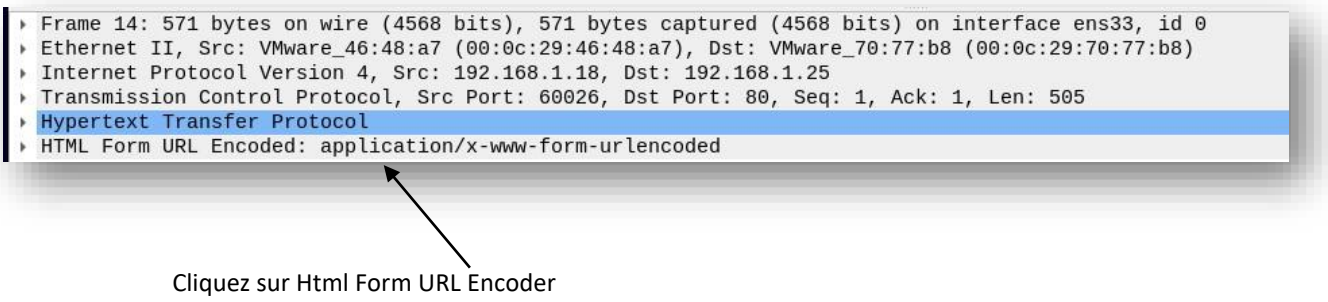
Arrêter la capture Wireshark.

Voilà nous venons d'enregistrer une courte séquence qui va être édifiante !

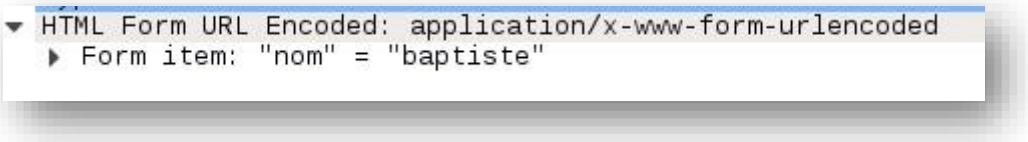
Dans l'ensemble des trames capturées recherchez une trame comme celle-ci :



Celle en bleu, comme vous le voyez, l'ordinateur source est le 192.168.1.18, le serveur est l'ordinateur de destination (192.168.1.25), le protocole http (qui est transporté par http)

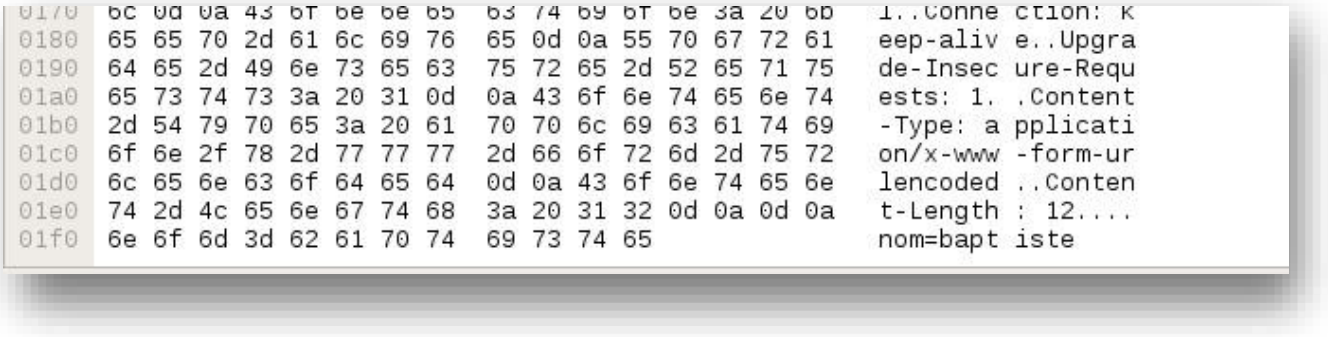


Cliquez sur Html Form URL Encoder



Comme vous le voyez, les informations transitent en clair !

C'est aussi visible dans le contenu de la trame :



L'information aurait été un mot de passe il passait en clair aussi !

Nous allons sécuriser maintenant les transits pour que les informations soient chiffrées entre le client et le serveur.

CREATION D'UN CERTIFICAT AUTOSIGNE ET SSL

SSL = **S**ecure **S**ocket **L**ayer

C'est un système qui permet d'échanger des informations entre 2 ordinateurs de façon sûre. SSL assure 3 choses :

- **Confidentialité** : Il est impossible d'espionner les informations échangées.
- **Intégrité** : Il est impossible de truquer les informations échangées.
- **Authentification** : Il permet de s'assurer de l'identité du programme, de la personne ou de l'entreprise avec laquelle on communique.

SSL est un complément à TCP/IP et permet (potentiellement) de sécuriser n'importe quel protocole ou programme utilisant TCP/IP.

SSL a été créé et développé par la société *Netscape* et *RSA Security*. On trouve désormais des versions open source ainsi qu'un protocole libre similaire : TLS (voir plus loin).

POURQUOI UTILISER SSL PLUTOT QU'UN AUTRE SYSTEME ?

POURQUOI UTILISER OPENSSL ?

SSL est standardisé.

Il existe une version libre de SSL: OpenSSL (http://www.openssl.org) que vous pouvez utiliser dans vos programmes sans payer de royalties.

OpenSSL est opensource: tout le monde peut contrôler et vérifier le code source (Le secret réside dans les clés de chiffrement, pas dans l'algorithme lui-même).

SSL a été cryptanalysé: ce système a été plus analysé que tous ses concurrents. SSL a été passé en revue par de nombreux spécialistes en cryptographique. On peut donc le considérer comme sûr.

Il est répandu : on peut facilement créer des programmes qui dialogueront avec d'autres programmes utilisant SSL.

Il faut se méfier des systèmes propriétaires : contrairement à ce qu'on pourrait penser, la sécurité d'un système de chiffrement ne réside pas dans le secret de l'algorithme de chiffrement, mais dans le secret de la clé. Il ne faut faire confiance qu'aux systèmes qui ont été publiés et analysés.

COMMENT ÇA MARCHE SSL ?

SSL consiste en 2 protocoles :

- **SSL Handshake protocol**: avant de communiquer, les 2 programmes SSL négocient des clés et des protocoles de chiffrement communs.
- **SSL Record protocol**: Une fois négociés, ils chiffrent toutes les informations échangées et effectuent divers contrôles.

LA NEGOCIATION SSL ("HANDSHAKE")

Au début de la communication, le client et le serveur s'échangent :

- La version SSL avec laquelle ils veulent travailler,
- La liste des méthodes de chiffrement (symétrique et asymétrique) et de signature que chacun connaît (avec longueurs de clés),
- Les méthodes de compression que chacun connaît, • Des nombres aléatoires,
- Les certificats.

Client et serveur essaient d'utiliser le protocole de chiffrement le plus puissant et diminuent jusqu'à trouver un protocole commun aux deux. Une fois que cela est fait, ils peuvent commencer à échanger des données.

LA COMMUNICATION SSL ("RECORD")

Avec SSL, l'expéditeur des données :

- Découpe les données en paquets,
- Compresse les données,
- Signe cryptographiquement les données,
- Chiffre les données,
- Les envois.

Celui qui réceptionne les données :

- Déchiffre les données,
- Vérifie la signature des données,
- Décompresse les données,
- Réassemble les paquets de données.

COMMENT SSL FAIT-IL POUR PROTEGER LES COMMUNICATIONS ?

SSL utilise :

- Un système de chiffrement asymétrique (comme RSA ou Diffie-Hellman). Vous pouvez en savoir plus ici : http://sebsauvage.net/comprendre/encryptage/crypto_asy.html. Il est utilisé pour générer la master key (clé principale) qui permettra de générer des clés de session.
- Un système de chiffrement symétrique (DES, 3DES, IDEA, RC4...) en utilisant les clés de session pour chiffrer les données.
- Un système de signature cryptographique des messages (HMAC, utilisant MD5, SHA...) pour s'assurer que les messages ne sont pas corrompus.

C'est lors de la négociation SSL que le client et le serveur choisissent des systèmes communs (chiffrement asymétrique, symétrique, signature et longueur de clé).

Dans votre navigateur, vous pouvez voir la liste des systèmes utilisés en plaçant votre curseur sur le petit cadenas quand vous êtes dans une page en HTTPS.

A QUOI SERVENT LES CERTIFICATS ?

Lors d'une négociation SSL, il faut s'assurer de l'identité de la personne avec qui on communique. Comment être sûr que le serveur auquel vous parlez est bien celui qu'il prétend être ?

C'est là qu'interviennent les certificats. Au moment de vous connecter sur un serveur web sécurisé, ce dernier vous enverra un certificat contenant le nom de l'entreprise, son adresse, etc. C'est une sorte de pièce d'identité.

COMMENT VERIFIER L'AUTHENTICITE DE CETTE PIECE D'IDENTITE ?

Ce sont les PKI (Public Key Infrastructure), des sociétés externes (auxquelles vous faites implicitement confiance), qui vont vérifier l'authenticité du certificat.

(La liste de ces PKI est incluse dans votre navigateur. Il y a généralement VeriSign, Thawte, etc.)

Ces PKI signent cryptographiquement les certificats des entreprises (et ils se font payer pour ça).

MISE EN ŒUVRE RAPIDE

Par défaut Apache 2 contient deux sites préconfigurés : « default » et « default-ssl » qui pointent tous les deux vers le répertoire « /var/www », mais le premier écoute sur le port 80 (HTTP) et le second sur le port 443 (HTTPS).

Dans la configuration d'origine, seul le site « default » est actif ce qui permet d'accéder à la page « It Works ! » d'Apache tout de suite après avoir effectué l'installation.

Vu que le site par défaut SSL, il est préconfiguré pour fonctionner. De ce fait, il suffit d'effectuer deux choses pour le rendre actif et opérationnel :

1. Activer le module SSL d'Apache
2. Activer le site « default-ssl » d'Apache

Une fois que ces deux activations sont effectuées, il suffit de recharger Apache et le site sera accessible en HTTPS. Voici les commandes à saisir sur l'ordinateur hébergeant les sites et en mode SU:

```
a2enmod ssl
```

```
a2ensite default-ssl
```

```
service apache2 reload
```

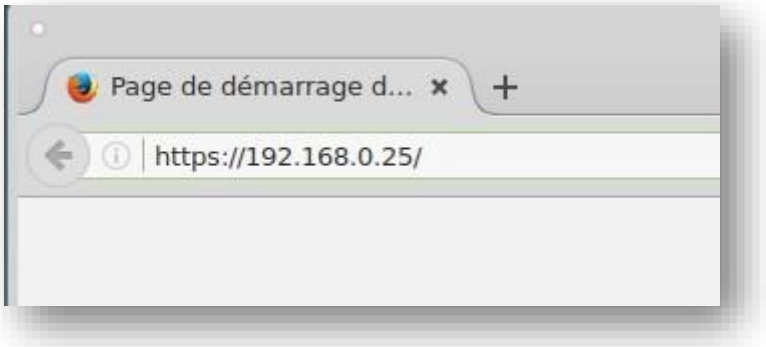
Vous remarquerez qu'il n'y a pas eu besoin de générer de certificat SSL.

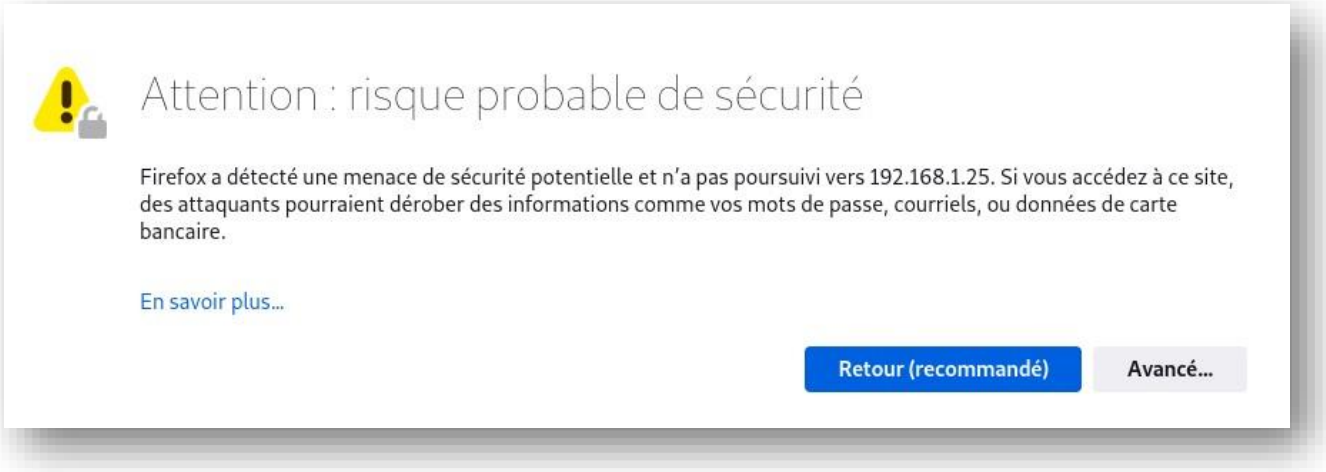
En effet, il y en a déjà un par défaut (valable 10 ans) comme je vous le disais et on peut voir où il se trouve en regardant de plus près le fichier « default-ssl » situé dans « /etc/apache2/sites-available » :

TEST DE CONNEXION

Sur l'ordinateur faisant office de client, nous allons nous connecter de nouveau sur le site web, mais cette fois-ci en utilisant https.

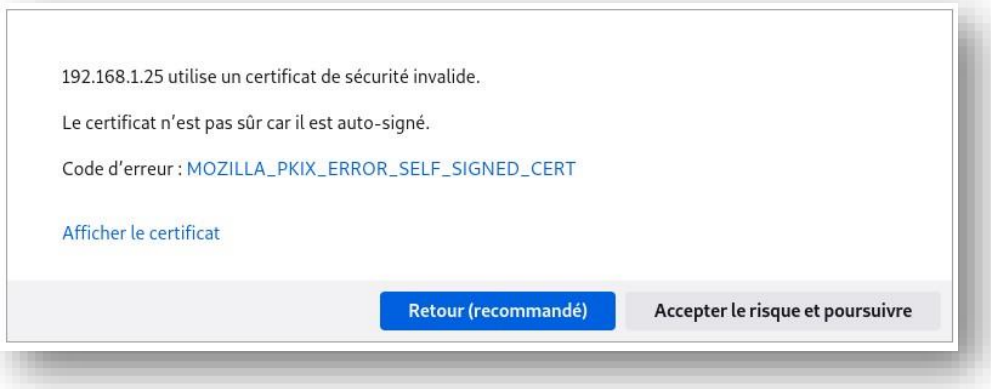
Lancez Firefox et dans la barre d'adresse saisissez https://@ip du site sur le serveur





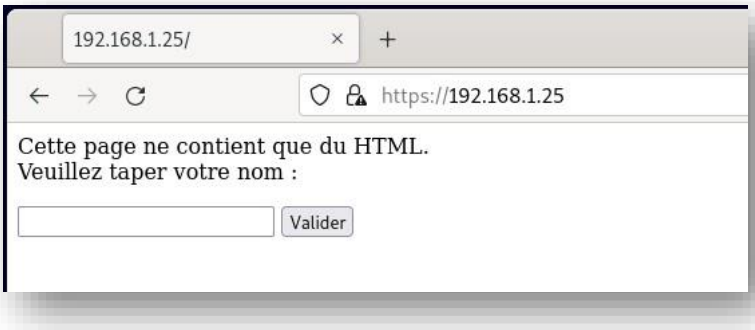
Vous avez un message d'avertissement, car comme le certificat n'est pas garanti par un tiers de confiance, Firefox préfère vous le signaler. Ce n'est pas grave nous allons lui faire comprendre d'avoir confiance dans ce site.

Cliquez sur le bouton **Avancé**.



Le message est explicite : Le certificat n'est pas de confiance, car il est auto signé.

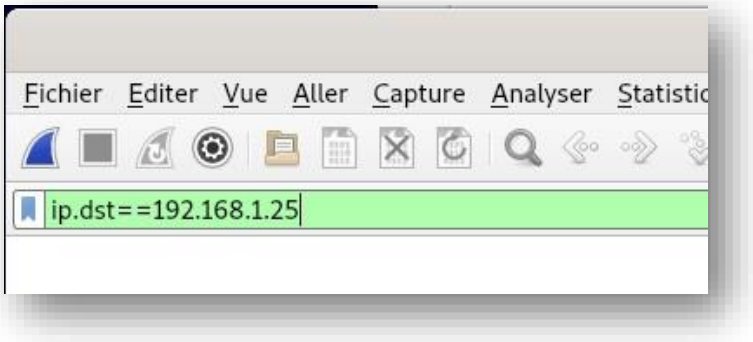
Cliquez sur **Accepter le risque et poursuivre**.



Comme vous le voyez, nous sommes bien en Https, nous avons bien le petit cadenas. Mission accomplie !

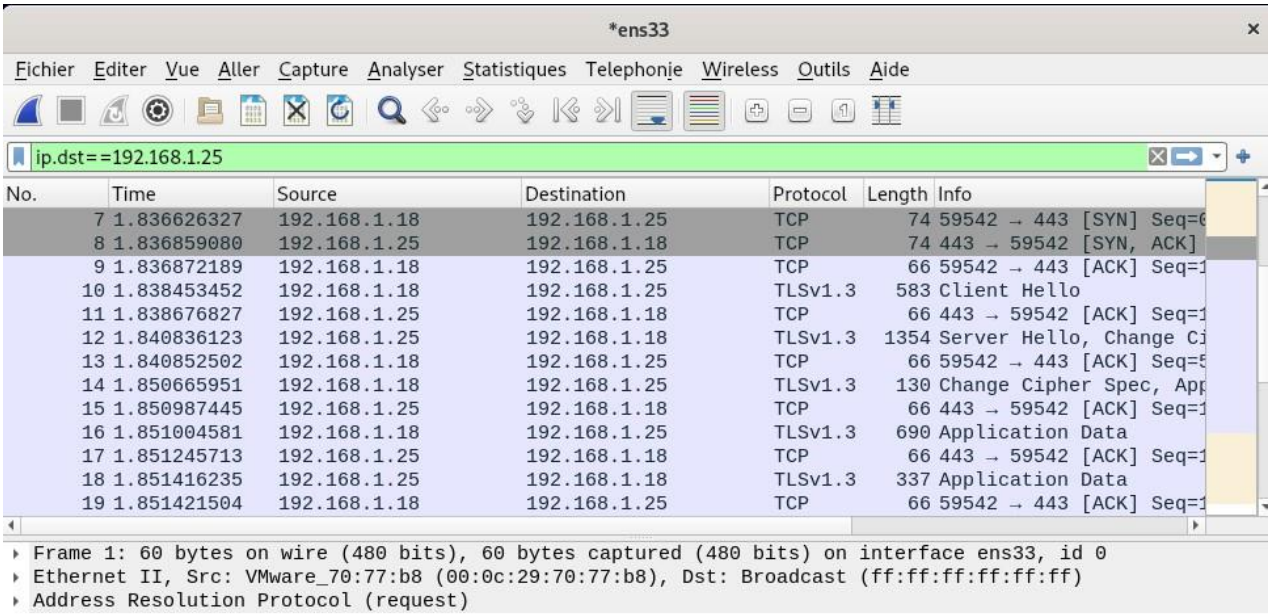
Lançons maintenant Wireshark pour capturer les échanges et voir si notre nom paraît en clair ou chiffré.

Nous allons nous intéresser qu'aux échanges avec l'ordinateur serveur. Dans mon cas le 192.168.1.25



Renseignez le champ du filtre en l'adaptant à votre situation. Ensuite, cliquez sur **Capture** puis **Démarrer**.

Une fois la capture faite appliquez le filtre pour ne garder que les trames à destination du serveur.



Observez les échanges :

Retrouvez-vous la chaîne en clair ?

Que représente le protocole : TLSv1.2 .

Que signifie Client Hello ?

Que signifie Client Key Exchange ?

Que signifie Application Data ?

WEBOGRAPHIE

https://www.it-connect.fr/configurer-le-ssl-avec-apache-2/

http://igm.univ-mlv.fr/~duris/NTREZO/20042005/Lamotte-Robert-Seigneurin_SSH-TLS.pdf

http://www.sebsauvage.net/comprendre/ssl/

ATTAQUE PAR FORCE BRUTE

Nous allons modifier le site web pour que la page d'accueil nous demande de nous logger avec un champ login et un autre contenant le mot de passe. Cette page enverra le login et le mot de passe à une autre page PHP qui testera la validité et si c'est ok affichera une page de bienvenue et sinon affichera une information d'erreur dans l'identification.

PREPARATION DU CONTEXTE.

Nous allons créer la page **index.php**

Ouvrez le fichier **index.php** avec nano en mode **Su** et copiez le code suivant :

```
<?php
session_start();
?>
<!doctype html>
<html lang="fr">
	<head>
		<meta charset="utf-8">
		<title>Accueil</title>
	</head>

	<body>
	<h1>Bonjour, Veuillez vous connecter : </h1>


	<form action="login.php" method="POST">
		<label for="Login">Votre login : </label>
		<input type="text" name="Login" id="login" />
		<label for="Password">Votre mot de passe : </label>
		<input type="password" name="Password" id="password" />
		<input type="submit" name="Connexion" value="OK" />
	</form>
	</body>
</html>
<?PHP
	// on affiche un bouton de déconnexion
	if($_SESSION)
	{
		echo "<a href='deconnexion.php' name='deconnexion'>Se deconnecter</a>";
	}
?>
```

Vous venez de créer une page de connexion qui ressemble à ceci :

Lorsque vous cliquerez sur le bouton Ok, les variables **Login** et **Password** seront envoyées à la page **login.php** qui fera les contrôles de validités nécessaires.

Maintenant, nous allons créer la page login.php, copiez ce code :

```
<?PHP
If (($_POST['Login']) && isset($_POST['Password'])) {
      $Login=$_POST['Login'];
      $Password=$_POST['Password'];
}
$LogAdmin="lgb";
$MdpAdmin="sio-42800";

if ($Password==$MdpAdmin & $Login==$LogAdmin )
{
      session_start();
      $_SESSION['Login']=$Login;
      header('Location: bienvenue.php');
} else {
      echo "<center><h1>Veuillez vérifier vos identifiants</h1></center>";
}
?>
```

Cette page vérifie que le login est **lgb** et le mot de passe : **sio-42800**

Si le couple login/Mdp est correct alors elle affiche la page **bienvenue.php**, sinon un message indiquant de vérifier les identifiants.

Création de la page **bienvenue.php** :

```
<?php
session_start();
?>

<!doctype html>
<html lang="fr">
	<head>
		<meta charset="utf-8">
		<title>Bienvenue !</title>
	</head>

<body>
	<?php		if($_SESSION)
		{		echo "<center><h1>	Bienvenue : 
".$_SESSION['Login']."</h1></center>.";
   echo "<center><hi><a href='index.php' name='Retour'>Retour à 
l'accueil</a></h1></center>";
		}
		?>
</body>
</html>
```

Si la personne saisit correctement les identifiants le résultat sera le suivant :

MISE EN PLACE DES OUTILS D'ATTAQUE FORCE BRUTE.

Nous allons considérer la machine cliente comme machine attaquante.

Il faut installer Python et le module pip.

Pour cela, veuillez installer les paquages correspondants avec la commande suivante :

```
apt-get install python3-pip
```

```
pip3 install mechanize
```

CREATION DU DICTIONNAIRE.

Nous allons créer un fichier txt nommé dico.txt que nous déposerons dans le répertoire home :

```
1234
test
toto
rotor
totor
sio-42800
root
toor
pifou
glop
pasglop
421314
gazzz
bumeuh
```

Comme vous le voyez en 6ieme position j'ai volontairement placé le mot : sio-42800

CREATION DU SCRIPT PYTHON

Nous allons maintenant créer le script python qui va parcourir le dictionnaire et tester l'ensemble des mots de passe du dictionnaire. Ce script doit être écrit dans le même répertoire que le fichier dico.txt, vous le nommerez **forceb.py**.

```
import mechanize

b = mechanize.Browser()

URL = "http://192.168.1.25/index.php"
dictionnary = "dico.txt"

try:
    dictionnary = open(dictionnary, "r")
except:
    print("fichier non trouvé")
    quit()

for password in dictionnary:
    response = b.open(URL)
    b.select_form(nr=0)
    b.form['Login'] = 'jlb'
    b.form['Password'] = password.strip()
    b.method = "POST"
    response = b.submit()

    if response.geturl() == "http://192.168.1.25/bienvenue.php":
        print(("le mot de passe recherche est : ") + password.strip())
    else:
        print("Le mot de passe n'a pas été trouvé")
```

Ce script relativement simpliste va tester tous les mots de passe du fichier dico associé au login 'lgb'. Comme vous le voyez, nous l'aidons en fournissant le login. Le script aurait pu être plus « pointu » et rechercher aussi le login. Mais le nombre de connexions aurait été trop élevé et le temps de résolution aussi.

MISE EN ŒUVRE

Pour lancer le script, il vous suffit d'exécuter la commande suivante python3 forceb.py

Vous devrez obtenir ce résultat-là :

```
root@vmclient:/home/lgb# python3 forceb.python3
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de pass rechercher est : sio-42800
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
Le mot de passe n'a pas été trouvé
python3 forceb.python3
```

Comme vous le voyez, le mot de passe a été trouvé, il nous faut passer maintenant à l'offensive en installant des contre-mesures !

IPTABLES ET FAIL2BAN

Les manipulations suivantes seront réalisées sur le serveur web.

Le but est de le sécuriser grâce à **iptables** et **fail2ban**.

Pour ceux qui ne connaissent pas fail2ban c'est un service qui va aller lire les logs d'autres services comme celui d'apache que nous voulons sécuriser ou de ssh, FTP et bien d'autres à la recherche de tentatives d'authentifications infructueuses répétée. Une fois que fail2ban va détecter une tentative d'intrusion, il va ajouter une règle dans **iptables** pour bannir l'adresse IP de la source pendant une période définie.

INSTALLATION DE 'IPTABLES

Commençons par faire les mises à jour du serveur

```
apt-get update
apt-get upgrade
```

Le service nftables est installé par défaut sur toutes les dernières distributions Debian nous allons le supprimer pour ne garder qu'iptables

```
apt-get remove --auto-remove nftables
apt-get purge nftables
```

Maintenant installez iptables

```
apt-get install iptables
```

Logiquement, sous Debian 11, Iptables va paraitre comme déjà installé.

Pour voir les règles de filtrages de votre iptable saisissez la commande suivante :

Iptables -L

```
root@vmserveur:/etc/init.d# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@vmserveur:/etc/init.d#
```

Comme vous le voyez, aucune règle restrictive n'est inscrite. Je vais vous présenter un petit script qui va vous permettre d'ouvrir le trafic de votre serveur uniquement pour les services de base.

On va créer notre fichier **firewall.sh** dans le répertoire **/etc/init.d/** où l'on retrouve tous les scripts lancés au démarrage de la machine

```
#!/bin/sh
#Vider les tables actuelles iptables -t filter -F


# Interdire toute connexion entrante et sortante iptables -t filter -P INPUT DROP
iptables -t filter -P FORWARD DROP iptables -t filter -P OUTPUT DROP


# Ne pas casser les connexions etablies
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT iptables -A OUTPUT
-m state --state RELATED,ESTABLISHED -j ACCEPT


# Autoriser le loopback
iptables -t filter -A INPUT -i lo -j ACCEPT iptables -t filter -A OUTPUT -o lo -j
ACCEPT


# ICMP (ping)
iptables -t filter -A INPUT -p icmp -j ACCEPT iptables -t filter -A OUTPUT -p icmp -
j ACCEPT
# FTP
iptables -t filter -A INPUT -p tcp --dport 21 -j ACCEPT iptables -t filter -A OUTPUT
-p tcp --dport 21 -j ACCEPT


# SSH
iptables -t filter -A INPUT -p tcp --dport 5285 -j ACCEPT iptables -t filter -A
OUTPUT -p tcp --dport 5285 -j ACCEPT


# DNS (bind)
```

```
iptables -t filter -A OUTPUT -p tcp --dport 53 -j ACCEPT iptables -t filter -A
OUTPUT -p udp --dport 53 -j ACCEPT iptables -t filter -A INPUT -p tcp --dport 53 -j
ACCEPT iptables -t filter -A INPUT -p udp --dport 53 -j ACCEPT
```

```
# APACHE : HTTP + HTTPS
iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT iptables -t filter -A
OUTPUT -p tcp --dport 443 -j ACCEPT iptables -t filter -A INPUT -p tcp --dport 80 -j
ACCEPT iptables -t filter -A INPUT -p tcp --dport 443 -j ACCEPT
```

```
# WEBMIN
iptables -t filter -A INPUT -p tcp --dport 42351 -m state --state NEW,ESTABLISHED -j
ACCEPT iptables -t filter -A OUTPUT -p tcp --dport 42351 -m state --state
ESTABLISHED -j ACCEPT
```

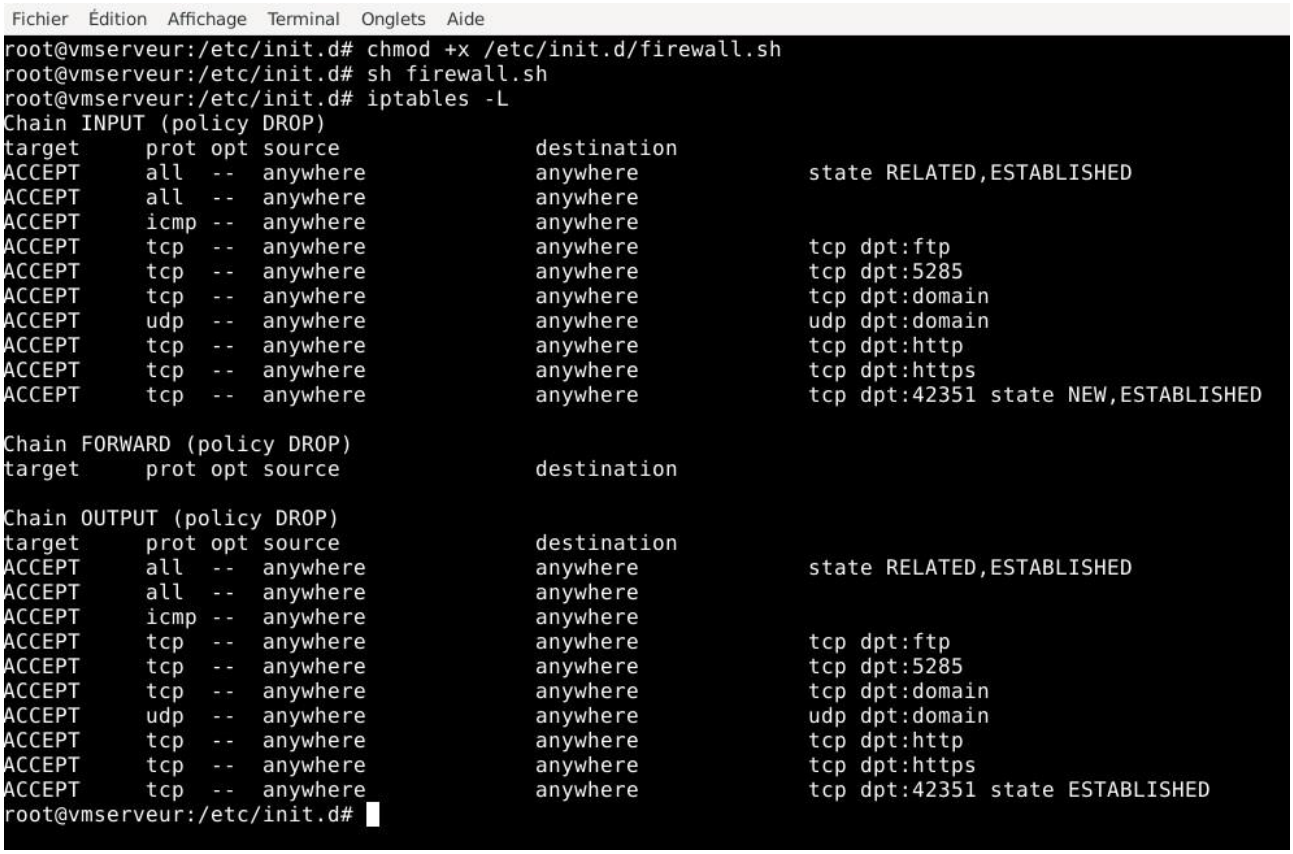
Après avoir enregistré, il faut rendre votre fichier exécutable **chmod +x /etc/init.d/firewall.sh**

Ensuite exécutez le

sh firewall.sh

Nous pouvons voir l'ensemble des règles qui sont maintenant inscrites dans iptables :

Iptables -L



Voilà, à partir de maintenant, tout ce qui n'est pas explicitement autorisé et interdit.

Nous allons maintenant installer un package qui va rendre persistantes les modifications des règles iptable.

En effet au reboot les règles de notre script vont d'êtres perdues. Il y a 2 solutions :

1. La première consiste à lancer le script au démarrage de l'ordinateur.
2. La seconde consiste à rendre les règles persistantes en installant un package.

Nous allons tester la 1iere solution car elle met en œuvre la notion de services.

LANCEMENT DU SCRIPT AU DEMARRAGE DE L'ORDINATEUR

SYSTEMD

La meilleure solution est d'utiliser le "nouveau" système d'initialisation de Linux **systemd**.

Depuis 2015, ce système est utilisé par la grande majorité des distributions Linux, dont Debian, Ubuntu et Raspbian. **Systemd** remplace avantageusement le vieux system V, car il est à la fois plus simple à utiliser et offre plus de possibilités, comme de gérer les logs d'exécutions.

Chaque service généré par **systemd** est configuré par un fichier **service** qui se trouve dans le répertoire /etc/systemd/system.

Pour notre exemple, il faut donc créer un fichier /etc/systemd/system/firewall.service avec au minimum le contenu suivant :

```
 [Unit]
Description=recharge les règles Iptables

[Service]
Type=simple
ExecStart=/etc/init.d/firewall.sh

[Install]
WantedBy=multi-user.target
```

Le paramètre **Description** permet de définir une description qui est affichée lors de certaines commandes de systemctl.

Le paramètre **ExecStart** de la section **Service** définit la commande à exécuter pour démarrer l'application, dans notre cas il s'agit de notre script /etc/init.d/firewall.sh.

Le paramètre **Type** permet de choisir le type de service, à moins que votre application ne fasse des forks, la valeur "simple" est suffisante.

Enfin le paramètre **WantedBy** définit à quel moment du boot le script doit être démarré. La valeur "**multi-user.target**" est la valeur qui devrait correspondre à 99% des cas, c'est-à-dire juste avant l'écran de login quand le réseau et les autres services critiques ont déjà été démarrés.

L'utilitaire **systemctl** permet de contrôler **systemd**. Les principales commandes sont :

- **Enable** : active le service, c'est-à-dire que le service sera démarré lors des prochains boots.
- **Disable** : désactive le service, c'est-à-dire que le service sera ignoré lors des prochains boots.
- **Status** : affiche l'état courant du service.
- **Start** : démarre immédiatement le service.
- **Stop** : stoppe immédiatement le service.

Donc, pour que notre script démarre automatiquement lors des prochains boots, il faut exécuter la commande suivante :

```
systemctl enable firewall.service
```

Tout de suite, on comprend un premier avantage de ce système : il est possible de stopper et redémarrer le service sans redémarrer l'OS. Ce qui permet par exemple de modifier le script **firewall.sh** sur un système en production. Mais ce n'est pas tout.

La commande **journalctl** permet d'afficher les logs d'exécution de notre programme. En plus, ces logs ne peuvent pas remplir complètement le disque car **systemd** stocke de la même manière tous les logs de l'OS. Donc, en fonction de la configuration de votre distribution, **systemd** gardera les X logs les plus récents du service, de manière à ne pas saturer le disque.

La commande pour afficher les logs de notre application **firewall.sh** :

```
journalctl -u firewall.service
```

Rebootez et relancez la commande **iptables -L**. Vous devriez voir toutes les règles.

INSTALLATION DE FAIL2BAN

```
apt-get install fail2ban
```

Ensuite :

```
systemctl start fail2ban
```

Puis

```
systemctl enable fail2ban
```

Nous allons configurer fail2ban pour créer une « prison virtuelle » d'adresses IP ayant trop fait d'essai de connexions.

Pour cela nous devons créer un fichier de nous nommerons **jail.local** dans **/etc/fail2ban/**

```
##Block the remote host that is trying to request suspicious URLs.
 [apache-overflows]
  enabled = true
  port = http,https
  filter = apache-overflows
  logpath = /var/log/apache2/*error.log
  maxretry = 2  bantime = 10m
  ignoreip = 128.0.0.1/8

##Block the remote host that is trying to search for scripts on the website to
execute.
 [apache-noscript]
  enabled = true
  port = http,https
  filter = apache-noscript
  logpath = /var/log/apache2/*error.log
  maxretry = 4
  bantime = 10m
  ignoreip = 128.0.0.1/8

 ##Block the remote host that is trying to request malicious bot.
 [apache-badbots]
  enabled = true
  port = http,https
  filter = apache-badbots
  logpath = /var/log/apache2/*error.log
  maxretry = 2
  bantime = 10m
  ignoreip = 128.0.0.1/8

 ##Stop DOS attack from remote host.
[http-get-dos]
  enabled = true
  port = http,https
  filter = http-get-dos
  logpath = /var/log/apache*/access.log
  maxretry = 2
  findtime = 400
  bantime = 10m
  ignoreip = 128.0.0.1/8
  action = iptables[name=HTTP, port=http, protocol=tcp]
```

```
##Block the failed login attempts on the SSH server.
[ssh]
 enabled = true
 port = ssh
 filter = sshd
 logpath = /var/log/auth.log
 maxretry = 2
 bantime = 10m
 ignoreip = 128.0.0.1/8
```

En voyant ce fichier, vous comprenez aisément le fonctionnement. Comme vous le voyez, il va lutter contre :

- Les URL suspectes,
- Les scripts malicieux,
- Les attaques Ddos,
- Les attaques brute-force sur le ssh,
- ….

Dans ce fichier, on va s'intéresser aux variables suivantes :

- **ignoreip** : correspond à la suite d'adresses IP qui ne se feront jamais bannir;
- **maxretry** : correspond au nombre d'essais;
- **findtime** : correspond à la période pendant laquelle les essais vont incrémenter maxretry;
- **bantime** : correspond au temps où l'adresse IP ne peut pas se connecter.

FILTRER LES ATTAQUES FORCE-BRUTE OU DOS

Nous allons maintenant créer le fichier de filtrage que nous nommerons **http-get-dos.conf** dans le répertoire /etc/fail2ban/filter.d/.

*Note : Le nom du filtre correspond à un des noms des sections présentes dans le fichier **jail.local***

```
Fail2Ban configuration file
 [Definition]


 # Option: failregex
 # Note: This regex will match any GET entry in your logs, so basically all valid and not valid entries are a match.
 # You should set up in the jail.conf file, the maxretry and findtime carefully in order to avoid false positives.
 failregex = ^<HOST> -.*"(GET|POST).*
 # Option: ignoreregex
   ignoreregex =#
```

Une fois le fichier créé nous allons redémarrer fail2ban :

```
systemctl restart fail2ban
```

Nous allons vérifier le statut de nos « prisons »

```
fail2ban-client status
```

Comme vous le voyez nous avons nos 6 prisons virtuelles :

```
Fichier  Édition  Affichage  Terminal  Onglets  Aide
root@vmserveur:~# fail2ban-client status
Status
|- Number of jail:      6
`- Jail list:   apache-badbots, apache-noscript, apache-overflows, http-get-dos, ssh, sshd
root@vmserveur:~#
```

TESTS

En lançant le script sur la machine Cliente, nous faisons 5 échecs avant de trouver le bon mot de passe, étant donné que nous avons bloqué le nombre de tentatives à 3, nous allons observer ce qui se passe après avoir lancé le script python **forceb.py**:

```
python3 forceb.py
```

Nous avons un message d'erreur

```
root@vmclient:/home/jlb# python3 forceb.py
Traceback (most recent call last):
  File "/home/jlb/forceb.py", line 14, in <module>
    response = b.open(url)
  File "/usr/local/lib/python3.9/dist-packages/mechanize/_mechanize.py", line 257, in open
    return self._mech_open(url_or_request, data, timeout=timeout)
  File "/usr/local/lib/python3.9/dist-packages/mechanize/_mechanize.py", line 313, in _mech_open
    raise response
mechanize._response.httperror_seek_wrapper: HTTP Error 403: b'request disallowed by robots.txt'
root@vmclient:/home/jlb#
```

Logiquement l'adresse IP de la machine cliente doit être en prison. Vérifions ça en repassant sur la machine serveur.

Saisissez :

```
fail2ban-client status http-get-dos
```

```
root@vmserveur:~# fail2ban-client status http-get-dos
Status for the jail: http-get-dos
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     22
|  `- File list:        /var/log/apache2/access.log
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   192.168.1.18
root@vmserveur:~#
```

Voilà qui a le mérite d'être clair ! Voyons le

résultat dans iptable :

iptables -S

```
root@vmserveur:~# iptables -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT DROP
-N f2b-HTTP
-A INPUT -p tcp -m tcp --dport 80 -j f2b-HTTP
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p tcp -m tcp --dport 21 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 5285 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 42351 -m state --state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 21 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 5285 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 53 -j ACCEPT
-A OUTPUT -p udp -m udp --dport 53 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 443 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 42351 -m state --state ESTABLISHED -j ACCEPT
-A f2b-HTTP -s 192.168.1.18/32 -j REJECT --reject-with icmp-port-unreachable
-A f2b-HTTP -j RETURN
root@vmserveur:~#
```

Voilà le résultat dans les règles de filtrage. L'adresse IP 192.168.1.18 est rejetée !

Nous venons de mettre en place une série de protections que vous savez maintenant mettre en œuvre.

Les plus rusés d'entre vous auront remarqué que nous ne bloquons pas les échecs de connexions, mais la reconnexion répétitive (Le Dos).

EXERCICE :

Affinez les filtres pour emprisonner l'IP si les connexions https sont refusées au bout de 3 essais de connexions.

Sites web consultés :

https://www.alibabacloud.com/blog/how-to-install-fail2ban-to-protect-against-brute-forcelogin-attacks_595078?spm=a2c41.13195052.0.0

https://www.gabinhocity.eu/securiser-son-serveur-avec-iptables-et-fail2ban/

https://www.geek17.com/en/node/125

https://fr-wiki.ikoula.com/fr/Mettre_en_place_fail2ban_sur_Debian