Maïssane El Mjidi
Joséphine Vetu
Inès Benalia

TP cybersécurité ***-Brèche dans la configuration SSL***

Objectif de l'activité : L'objectif de cette activité est de détourner une redirection HTTPS pour forcer une connexion en HTTP non sécurisé et ainsi intercepter les identifiants de connexion.

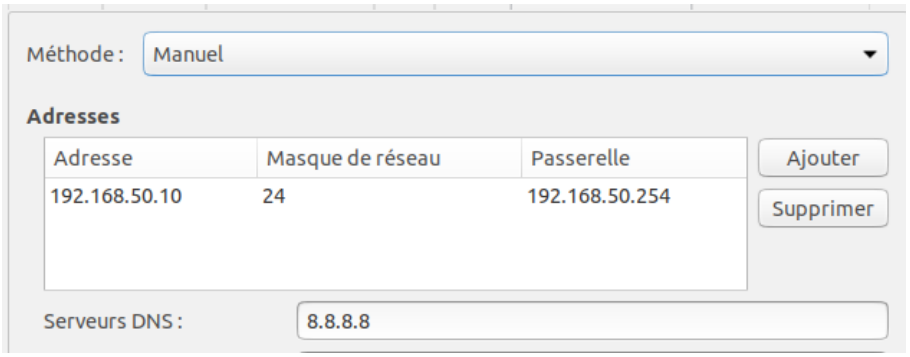
1 - Mise en place de l'attaque et de son environnement :

→ *rediriger une connexion HTTPS en HTTP*

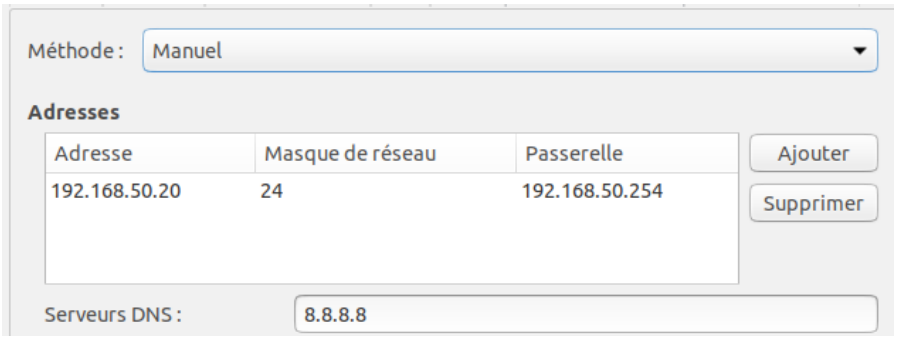
1. Préparation de l'environnement de travail :

   Les machines sont configurées de la manière suivante:

- La machine client



- La machine hacker



- Le serveur Ubuntu est configuré grâce à la commande:
  sudo nano /etc/network/interfaces

*TP cybersécurité -**Brèche dans la configuration SSL***

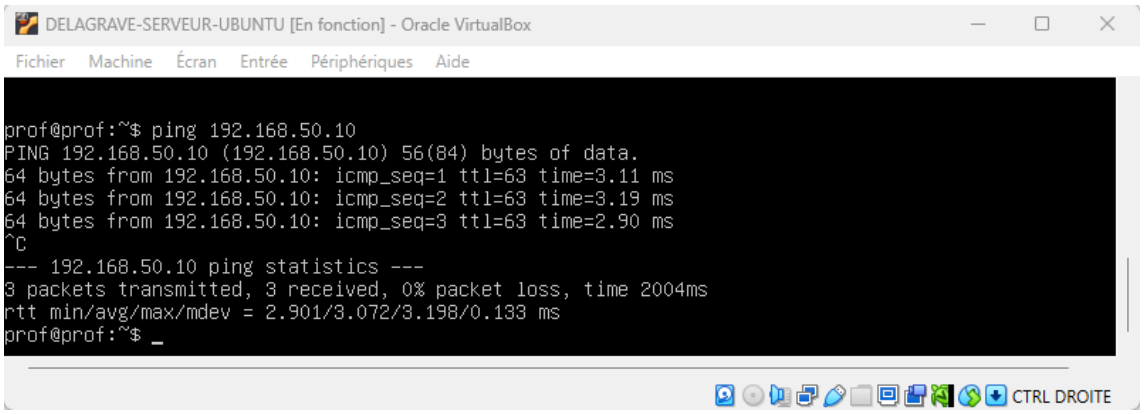
```
auto enp0s3

iface enp0s3 inet static
address 172.16.10.5
netmask 255.255.255.0
gateway 172.16.10.254
```

Les machines clientes communiquent entre elles:

```
prof@prof:~$ ping 192.168.50.10
PING 192.168.50.10 (192.168.50.10) 56(84) bytes of data.
64 bytes from 192.168.50.10: icmp_seq=1 ttl=64 time=1.94 ms
64 bytes from 192.168.50.10: icmp_seq=2 ttl=64 time=1.14 ms
64 bytes from 192.168.50.10: icmp_seq=3 ttl=64 time=1.21 ms
^C
--- 192.168.50.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.146/1.435/1.943/0.360 ms
```

```
prof@prof:~$ ping 192.168.50.20
PING 192.168.50.20 (192.168.50.20) 56(84) bytes of data.
64 bytes from 192.168.50.20: icmp_seq=1 ttl=64 time=1.55 ms
64 bytes from 192.168.50.20: icmp_seq=2 ttl=64 time=3.23 ms
64 bytes from 192.168.50.20: icmp_seq=3 ttl=64 time=0.910 ms
64 bytes from 192.168.50.20: icmp_seq=4 ttl=64 time=1.66 ms
^C
--- 192.168.50.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 0.910/1.841/3.235/0.855 ms
```

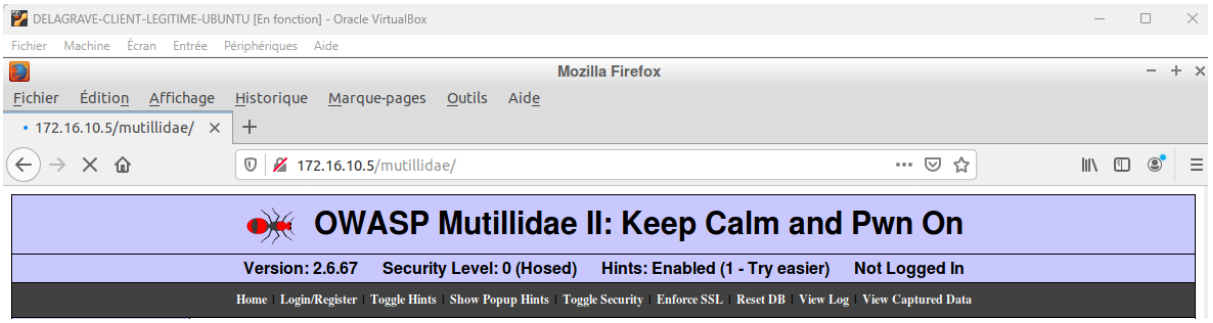
Le serveur communique vers les machines clientes:



*TP cybersécurité -**Brèche dans la configuration SSL***

```
prof@prof:~$ ping 192.168.50.20
PING 192.168.50.20 (192.168.50.20) 56(84) bytes of data.
64 bytes from 192.168.50.20: icmp_seq=1 ttl=63 time=2.83 ms
64 bytes from 192.168.50.20: icmp_seq=2 ttl=63 time=3.66 ms
^C
--- 192.168.50.20 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 2.830/3.248/3.666/0.418 ms
prof@prof:~$
```

Les machines clientes ont accès au site du serveur:



*TP cybersécurité -**Brèche dans la configuration SSL***

Sur la machine virtuelle contenant le serveur web mutillidae, il faut activer le module SSL (Secure Sockets Layer). C'est un composant logiciel utilisé par les serveurs web, comme Apache, qui permet d'activer les communications sécurisées via le protocole HTTPS.

1. *faire la commande :*

   a2enmod ssl

explication de la commande faite :
a2enmod : script qui permet d'activer les modules Apaches
ssl : activation du module ssl

2. *faire la commande :*

   a2ensite default-ssl.conf

explication de la commande faite :
a2ensite : script qui configure les sites sur le serveur web Apache
default-ssl.conf : nom du fichier que l'on appelle , il permet d'activer le mode ssl pour sécuriser la communication entre les sites web .

3. *faire la commande :*

   service apache2 restart

explication de la commande faite :
redémarre le serveur Apache

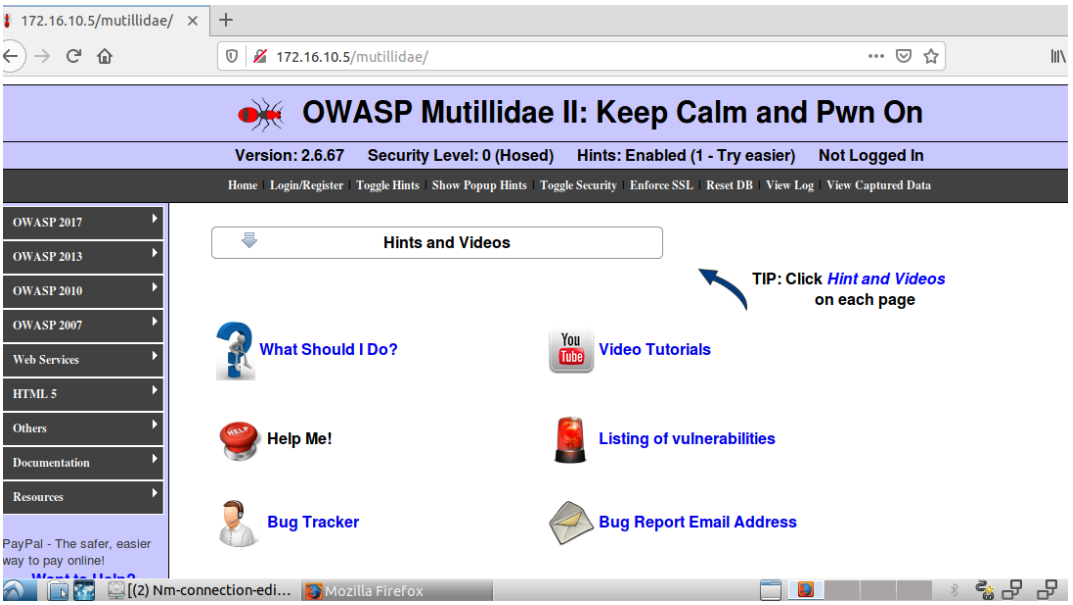
Voici la fenêtre après l'exécution des commandes. Avant de redémarrer, il faut remettre les login et mot de passe de la machine :

*TP cybersécurité -**Brèche dans la configuration SSL***

```
prof@prof:~$ a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
prof@prof:~$ a2ensite default-ssl.conf
Site default-ssl already enabled
prof@prof:~$ service apache2 restart
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to restart 'apache2.service'.
Authenticating as: prof
Password:
```

Sur la machine ‘victime’ :

accéder à mutillidae depuis la machine client :



Sur la machine ‘attaquante’ (hacker) :

il faut installer arpspoof (réaliser l’attaque MITM) et sslstrip (tronque la redirection HTTPS)

1. *faire la commande :*

sudo apt install dsniff sslstrip
sudo apt-get update

*TP cybersécurité -**Brèche dans la configuration SSL***

explication de la commande:

sudo: donne les privilèges administrateur
apt install : permet une installation
dsniff : package d'outils de sécurité réseau utilisés pour intercepter et analyser le trafic réseau.
sslstrip : outil qui permet de réaliser des attaques comme MITM .

On obtient la fenêtre suivante, on entre "o" pour continuer. L'installation est effectuée.

On accède au fichier /etc.sysctl.conf avec la commande **sudo nano /etc.sysctl.conf** et on décommente la ligne suivante. Ce fichier permet de configurer les paramètres du noyau Linux. Cette ligne active le routage en ipv4.

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

On effectue ensuite la commande suivante.

sudo iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080

C'est une règle iptables qui configure une redirection de port en utilisant la table NAT.

```
prof@prof:~$ sudo iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080
prof@prof:~$
```

2 - Déroulement de l'attaque :

Il s'agit désormais de passer à l'attaque depuis le poste du hacker. Nous allons réaliser une attaque de **spoofing**. C'est une technique qui consiste à usurper l'identité d'une entité (adresse IP, adresse MAC, email, etc.) pour tromper un système ou une personne et accéder à des informations ou perturber des communications.

On effectue ici une attaque ARP spoofing avec la commande suivante:

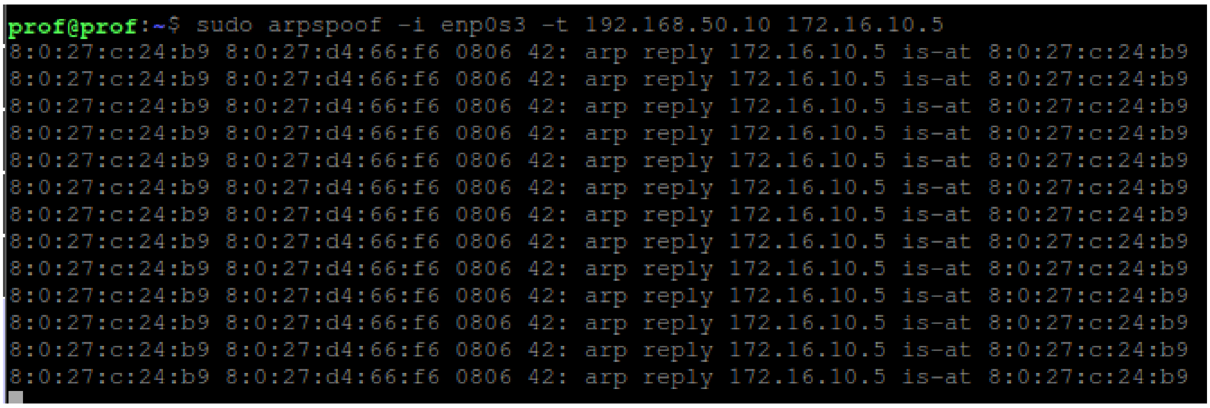
sudo arpspoof -i enp0s3 -t 192.168.50.10 172.16.10.5

arpspoof: outil utilisé pour effectuer une attaque ARP spoofing. Il falsifie les réponses ARP (Address Resolution Protocol) pour tromper les machines sur un réseau local (LAN).

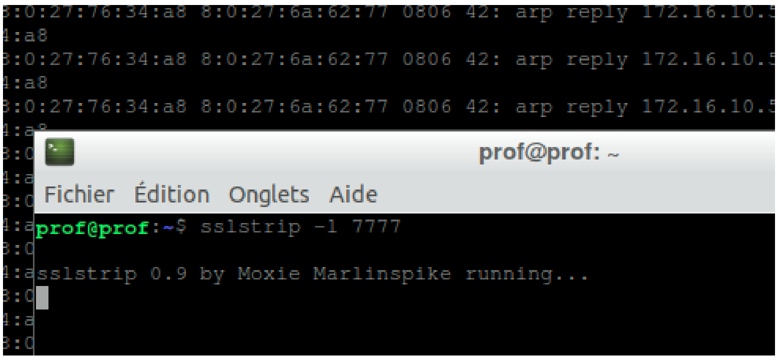
Maïssane El Mjidi
Joséphine Vetu
Inès Benalia

*TP cybersécurité -**Brèche dans la configuration SSL***

L'outil arpspoof falsifie les réponses ARP envoyées à la victime (192.168.50.10), lui faisant croire que l'adresse MAC du serveur (172.16.10.5) est en réalité celle de l'attaquant. Cela entraîne un empoisonnement du cache ARP, redirigeant ainsi tout le trafic destiné au serveur vers l'attaquant. Ce dernier peut alors intercepter, modifier ou relayer les communications entre la victime et le serveur, mettant ainsi en place une attaque Man-in-the-Middle (MITM).

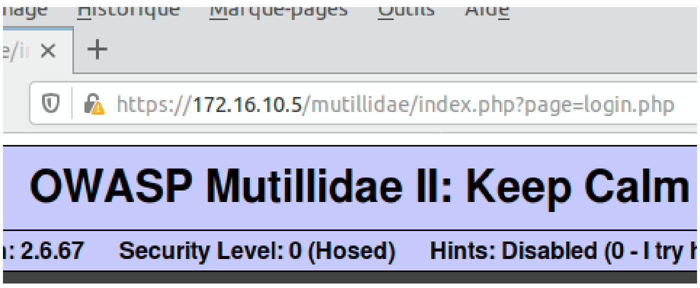


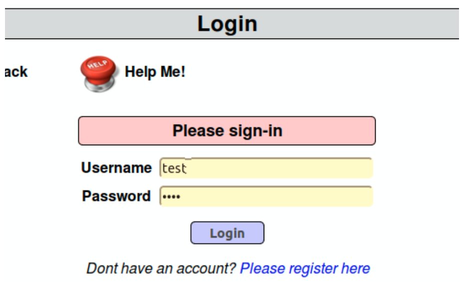
On ouvre ensuite un deuxième terminal afin d'exécuter sslstrip et permettre la redirection vers le port défini plus tôt



Sur Machine victime:

On réalise une authentification depuis mutillidae avec le niveau de sécurité le plus bas. Le site web n'est plus sécurisé.





*TP cybersécurité -**Brèche dans la configuration SSL***

```
rof@prof:~$ cat sslstrip.log
rof@prof:~$
```

On exécute ensuite une commande afin d'afficher le contenu du fichier sslstrip.log,répertoriant les données interceptées par le trafic non sécurisé. Mais le fichier, censé contenir les identifiants interceptés par l'attaque, est anormalement vide.

3 - Observations sur le dysfonctionnement:

```
prof@prof:~/mutillidae$ sudo lsof -i -P -n | grep LISTEN
sshd       943            root    3u  IPv4  19949      0t0  TCP *:22 (LISTEN)
sshd       943            root    4u  IPv6  19960      0t0  TCP *:22 (LISTEN)
mysqld     983           mysql   30u  IPv4  21146      0t0  TCP 127.0.0.1:3306 (LISTEN)
apache2   1189            root    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1189            root    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1309        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1309        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1310        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1310        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1311        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1311        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1312        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1312        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1313        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1313        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1436        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1436        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1437        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1437        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1438        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1438        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1439        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1439        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
apache2   1466        www-data    4u  IPv6  19999      0t0  TCP *:80 (LISTEN)
apache2   1466        www-data    6u  IPv6  20003      0t0  TCP *:443 (LISTEN)
systemd-r 1803 systemd-resolve   13u  IPv4  26331      0t0  TCP 127.0.0.53:53 (LISTEN)
prof@prof:~/mutillidae$ _
```

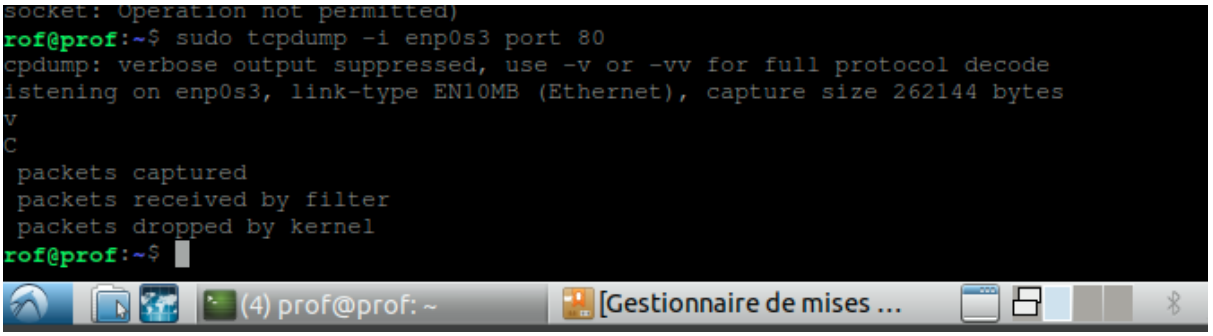
vérification des ports utilisés par le serveur Apache2

```
Fichier  Édition  Onglets  Aide
rof@prof:~$ sslstrip -l 7777

slstrip 0.9 by Moxie Marlinspike running...
Cprof@prof:~$ sslstrip -l 7777

slstrip 0.9 by Moxie Marlinspike running...
Cprof@prof:~$ sslstrip -l 8080

slstrip 0.9 by Moxie Marlinspike running...
```

```
rof@prof:~$ sudo iptables -t nat -L -v -n
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REDIRECT   tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80 redir ports 7777
    0     0 REDIRECT   tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80 redir ports 8080

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 8 packets, 224 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 8 packets, 224 bytes)
```

Utilisation des deux ports configurés lors de la redirection donc normalement tout trafic est redirigé vers eux.

*TP cybersécurité -**Brèche dans la configuration SSL***



Commande permettant de vérifier le trafic sur ce port, on peut voir que celui-ci est anormalement vide.

4 - Etude du code permettant de renforcer la sécurité du site web en forçant l'utilisation de protocole sécurisé:

Le script a pour objectif de forcer l'utilisation de HTTPS (SSL) sur certaines pages de ton site Web, mais uniquement si la session utilisateur l'exige:

```php
<?php



    case "1": // sécurité niv 1

        if ($_SESSION["EnforceSSL"] == "True") {  // Si la session exige l'utilisation de HTTPS

            // Vérifie si le serveur ne sert pas la page en HTTPS (si 'HTTPS' n'est pas défini ou si ce n'est pas "on")

            if (isset($_SERVER['HTTPS']) || $_SERVER['HTTPS'] != "on") {

                // Crée l'URL en HTTPS, en utilisant le nom du serveur et l'URI actuelle

                $iSecureRedirect = "https://".$_SERVER['HTTP_HOST'].$_SERVER['REQUEST_URI'];



                // Effectue une redirection vers l'URL HTTPS

                header("Location: $iSecureRedirect");



                // Terminer immédiatement l'exécution du script après la redirection

                exit();
```

*TP cybersécurité -**Brèche dans la configuration SSL***

```
            }

        }

        break;

case "5":  //  sécurité niv 5

    // Vérifie si la session exige l'utilisation de HTTPS (EnforceSSL est défini sur "True")

    if ($_SESSION["EnforceSSL"] == "True") {

        // Vérifie si le serveur ne sert pas la page en HTTPS (si 'HTTPS' n'est pas défini ou si ce
n'est pas "on")

        if (isset($_SERVER['HTTPS']) || $_SERVER['HTTPS'] != "on") {

            // Charge un fichier externe 'ssl-enforced.php' qui peut gérer la logique de forçage de
HTTPS

            require_once('ssl-enforced.php');



            // Termine immédiatement l'exécution du script après avoir inclus le fichier
'ssl-enforced.php'

            exit();

        }

    }

    break;
```

Niveau de sécurité 1 :

Q1. Est-ce que le niveau de sécurité 1 permet d'empêcher cette attaque ?

Non, le niveau 1 n'empêche pas l'attaque SSLSTRIP. L'attaquant peut intercepter la redirection HTTPS et forcer la connexion HTTP.

Q2. Est-ce que le niveau de sécurité 1 permet d'empêcher les conséquences de cette attaque ? Où se situe la faille ?

Non, le niveau 1 ne protège pas contre les conséquences de l'attaque. La faille réside dans l'absence de mécanisme empêchant une interception MITM, permettant ainsi à l'attaquant de capturer les identifiants envoyés en HTTP.

Q3. Expliquer le code source mis en œuvre avec ce niveau de sécurité dans index.php.

*TP cybersécurité -**Brèche dans la configuration SSL***

Le code tente de rediriger vers HTTPS si $_SESSION["EnforceSSL"] est activé, mais il ne protège pas contre les attaques MITM car la redirection peut être interceptée.

Niveau de sécurité 5 :

Q4. Est-ce que le niveau 5 permet d'empêcher l'attaque ?

Oui, le niveau 5 empêche l'attaque en utilisant **HSTS** (HTTP Strict Transport Security), forçant le navigateur à se connecter uniquement en HTTPS.

Q5. Est-ce que le niveau 5 permet d'empêcher les conséquences de l'attaque ?

Oui, le niveau 5 empêche les conséquences, car HSTS garantit que les utilisateurs utilisent uniquement HTTPS, évitant les interceptions de données.

Q6. Expliquer le code mis en œuvre avec ce niveau de sécurité dans index.php.

Le code du fichier index.php vérifie si HTTPS est activé, et si ce n'est pas le cas, il force la redirection avec require_once('ssl-enforced.php'). En cas d'attaque, l'utilisateur est redirigé vers une page sécurisée (HTTPS). La bonne pratique est de toujours forcer HTTPS avec HSTS pour garantir la confidentialité.