

Rapport - Tâche 3 : Serveurs HTTP/HTTPS et serveur FTP/FTPS

HTTP est le protocole de communication client-serveur pour le Web. Sa variante, HTTPS, est sécurisée par chiffrement et authentification.

FTP est le protocole de transfert de fichiers par Internet. On peut le voir comme une sorte de partage de fichiers sur réseau TCP/IP (Internet donc). FTPS apporte une sécurité lors de ces échanges de fichiers qui se font par dessus TLS.

Un autre moyen de sécuriser ces échanges serait d'utiliser SFTP. Cette variante utilise le protocole SSH pour le transfert de fichier. Cela a le mérite de n'utiliser qu'un seul port pour tout.

L'objectif ici est de sécuriser nos services de façon native, c'est-à-dire mettre en place une solution qui offre déjà une sécurité et ne peut pas uniquement compter sur le firewall. Afin de l'atteindre, j'ai réalisé chacune de ces étapes :








1. Installation des serveurs HTTP/HTTPS

a) Choix du serveur à utiliser

J'avais particulièrement **lighttpd**, **apache** et **nginx** comme serveur web à implémenté. Mais pour des raisons **de simplicité et d'accessibilité**, j'ai éliminé **lighttpd**. De plus, nous disposons de ressources (mémoire notamment) nécessaires donc son aspect **léger** ne jouait pas trop.

Finalement, je suis tombé sur cette comparaison qui montre les fonctionnalités des deux serveurs restants. Avec une **communauté avérée** et aussi pour des **raisons d'intimité** (plus utilisé personnellement), j'ai décidé de mettre en place **Apache**. **Les autres ont également approuvé cette idée.**

NGINX vs Apache: Head to Head Comparison

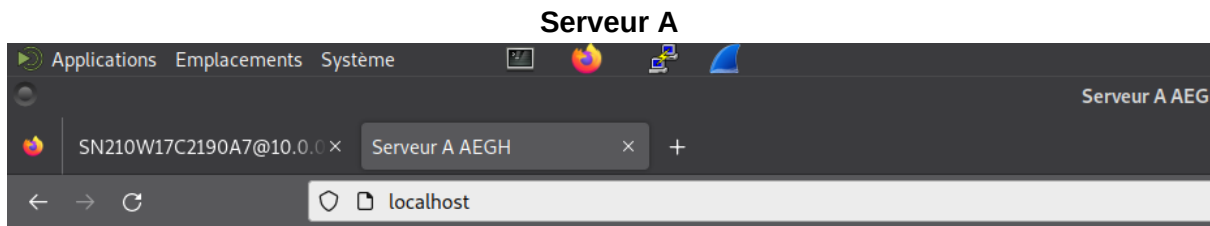
Feature	Apache	NGINX
 Simplicity	Easy to develop and innovate because of its one-connection-per-process model	Complex to develop as it has a sophisticated architecture to handle multiple connections concurrently.
 Performance – Static Content	Slow	2.5 times faster than Apache and consumes less memory
 Performance – Dynamic Content	Excellent	Excellent
 Operating system support	Supports all Unix OS and Windows	Supports all Unix OS and Windows; however, performance on Windows isn't as stable.
 Security	Comparatively the same level of security.	Comparatively the same level of security.
 Flexibility	Can be customized by adding modules. Apache had dynamic module loading for a long time.	NGINX version 1.11.5 and <u>NGINX Plus Release R11</u> introduced compatibility for dynamic modules.
 Support and Documentation	Excellent support and documentation are available, as it has been in the market for a very long time.	Though there was a weak start for support and documentation for NGINX, it has grown rapidly since.

> **hackr.io**

Source : <https://hackr.io/blog/nginx-vs-apache>

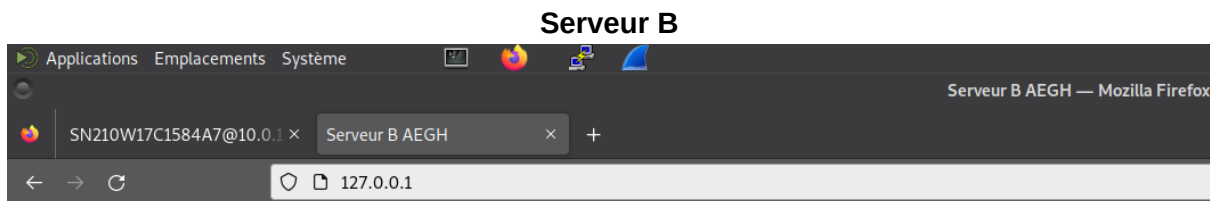
b) Mise en place du service Web

Après une mise à jour des dépôts Debian, `sudo apt-get update`, l'installation d'Apache se fait assez facilement avec une commande : `sudo apt-get install apache2`



Bienvenue !

SAE 401



Bienvenue !

SAE 401

c) Sécurisation avec HTTPS

La sécurisation du service Web commence premièrement par l'activation de HTTPS. Ce dernier ajoute une couche de **chiffrement TLS**. L'identité du serveur est alors vérifiée grâce à un **certificat d'authentification** émis par une **autorité tierce (ici, la machine elle-même)**. Ce qui garantit en théorie, la **confidentialité et l'intégrité** des données envoyées et reçues.

Pour y parvenir, j'ai premièrement activé le **module SSL** et vérifié que le **port 443** est en écoute. Ensuite j'ai **repris une demande de signature de certificat**. J'ai **créé la clé** puis **généralisé un certificat de 365 jours**. Une fois désigné comme **CA**, j'ai installé la **clé et le certificat**.

FR/Doubs/Montbeliard/UFC-AEGH/RT/serveurX/serveurx@ufc-aegh.fr (x= serveur).
commandes tapées dans cette ordre :

`sudo a2enmod ssl`

`mkdir /tmp/ssl_conf ; cd /tmp/ssl_conf`

`openssl req -config /etc/ssl/openssl.cnf -new -out csr_ssl.csr`

`openssl rsa -in privkey.pem -out cle_ssl.key`

`openssl req -new -x509 -days 365 -key cle_ssl.key > ca.crt`

`openssl x509 -req -in csr_ssl.csr -out crt_ssl.crt -CA ca.crt -CAkey cle_ssl.key -`

`CAcreateserial -CAserial ca.srl`

```
cp cle_ssl.key /etc/ssl/certs/ ; cp ca.crt /etc/ssl/certs/
```

Pour terminer, j'ai modifié le fichier **/etc/apache2/sites-available/default-ssl.conf** pour renseigner les chemins vers notre **certificat et sa clé**.

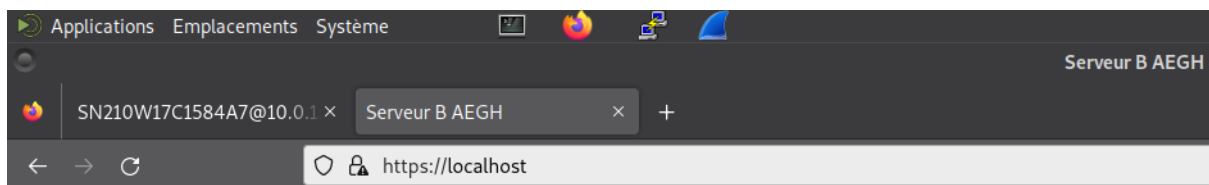
Module SSL

```
tpert:~$ sudo 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 to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
tpert:~$ sudo nano /etc/apache2/ports.conf
tpert:~$ sudo systemctl restart apache2
tpert:~$ netstat -nlt |grep 443
tcp6      0      0 :::443          :::*             LISTEN
tpert:~$
```

Certificat x509

```
tpert:/tmp/ssl_conf$ cat ca.crt
-----BEGIN CERTIFICATE-----
MIID+TCCAuGgAwIBAgIUyUw9xkkjXWrYFVdaJdhzljAH4nUwDQYJKoZIhvcNAQEL
BQAwYsxCzAJBgNVBAYTAKZSMQ4wDAYDVQQIDAVEb3ViczeUUMBGA1UEBwwLTW9u
dGJlbGhcmQxETAPBgNVBAoMCFVGQy1BRUdIMQswCQYDVQQLDAJSVDERMA8GA1UE
AwwIc2VydMv1ckIxIzAhBgkqhkiG9w0BCQEFHnlnZldXJiQHVmYy1hZWdoLmZy
MB4XDTE0MDMyNTE0MjkwOFoXDTI1MDMyNTE0MjkwOFowYsxCzAJBgNVBAYTAKZS
MQ4wDAYDVQQIDAVEb3ViczeUUMBGA1UEBwwLTW9udGJlbGhcmQxETAPBgNVBAoM
CFVGQy1BRUdIMQswCQYDVQQLDAJSVDERMA8GA1UEAwwIc2VydMv1ckIxIzAhBgkq
hkiG9w0BCQEFHnlnZldXJiQHVmYy1hZWdoLmZyMIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAph08Lb0tE5/KHlhrd+xKjFxFuUaYuCaAMSqLPkD5/Jh6Mq
FmrHd+ZDdU/CCIg/C4Tix9NywqfU1ZVZubXXD908jMJZTi+wIrkvDLWJYb0uueE3
HVoavpzcXFS14606yi4FPFNHGHMRF0KtVd3BHBk+mX84eISfz3kBbysR20p8GzU
eBaBaz4Ywxdu6ohAW86fOM+3mHzQRvjA9kXymZ24SjS2VfKDC0adotk9TbIyW1W
XPUp5tuXFNLV+ELTgZ0LWuHAX5KQMig70xCg8XL7GqGWej2ipGdVn+bcD2G1ho4u
glfWDF16xyFHYDd1UNPM8HdCwg/6Nx5yG93kR5jvGwIDAQABo1MwUTAdBgNVHQ4E
FgQUQT3s5fBL84U7cHhfGiA3+3LGXHcwHwYDVR0jBBgwFoAUQT3s5fBL84U7cHhf
GiA3+3LGXHcwHwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCQAQEAiVAe
F1mP/iSp9Gaps9Sd99066tjqBn91e8uLCRDE+GCw5ZaUTqAkII+S/iekWk9bXkCT
WguN//mqAVR+TpZ6ccyZqGgUY30XfxgXpnpdJBKxHDeKkDwAcjTLMu0ld1S3SKPG
QNBMAjSrIA2MBMMFVpCL8bT7Qh3NQV/n/7Y+rKzUbGsahvvc5F9vQhfItQkwHgJ0
Q0vbskL+5S1zdQiJUQEuD27vTpiFBU0TPyukSs+egM1fpp2IdC8LVxj1l2nBw/zR
9TL/V7ksGJHZGZnwo0mz/AwKnUnvtn1KyM+JHpg9jbb8tZquNM56NbDBz3Y6QGYk
HS9qdZRESJsdprrXvQ==
-----END CERTIFICATE-----
```

Site HTTPS



Bienvenue !

SAE 401

d) Installation d'un CMS

Un CMS est un système de gestion de contenu. Il permet de gérer (conception et mise à jour) les sites web dynamiques.

Pour mettre en place un tel système, il faut avoir au préalable un serveur web, une base de données et choisir un langage de programmation en fonction du CMS.

J'ai choisi d'utiliser **WordPress** comme CMS. Ce dernier requiert une **base de données type MySQL** et **PHP** comme langage de programmation. J'ai installé **MariaDB** puis **phpMyAdmin** pour une simplicité dans la manipulation. Le certificat **SSL/TLS** étant déjà généré, je procède à la configuration du CMS.

commandes tapées dans cet ordre :

```
sudo apt-get update ; sudo mkdir /var/www/html/sae401/
```

MariaDB et phpMyAdmin

```
sudo apt-get install -y php php-json php-mbstring php-zip php-gd php-xml php-curl  
php-mysql
```

```
sudo apt-get install mariadb-server -y
```

```
sudo apt policy mariadb-server ; cd /tmp
```

```
sudo wget https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-all-  
languages.zip
```

```
sudo mv /tmp/phpMyAdmin-5.2.1-all-languages/ /usr/share/phpmyadmin
```

```
openssl rand -base64 32
```

```
sudo nano /usr/share/phpmyadmin/config.inc.php
```

```
sudo mkdir -p /var/lib/phpmyadmin/tmp
```

```
sudo chown -Rfv www-data:www-data /usr/share/phpmyadmin/
```

```
sudo chown -Rfv www-data:www-data /var/lib/phpmyadmin/
```

```
sudo cp /usr/share/phpmyadmin/config.sample.inc.php
```

```
/usr/share/phpmyadmin/config.inc.php
```

```
mysql -u root -p < /usr/share/phpmyadmin/sql/create_tables.sql
```

```
mysql -u root -p
```

```
    MariaDB > CREATE DATABASE word_press_aegh;
```

```
    MariaDB > CREATE USER 'notre_user_admin'@'localhost' IDENTIFIED BY  
'notre_super_mdp';
```

```
    MariaDB > GRANT ALL PRIVILEGES ON notre_bdd.* TO  
notre_user_admin@localhost;
```

```
    MariaDB > FLUSH PRIVILEGES; EXIT;
```

```
sudo nano /etc/apache2/conf-available/phpmyadmin.conf
```

```
sudo a2enconf phpmyadmin ; sudo apachectl configtest
```

WordPress (toujours dans /tmp)

```
sudo wget https://wordpress.org/latest.zip
```

```
sudo unzip latest.zip -d /var/www/html/sae401/
```

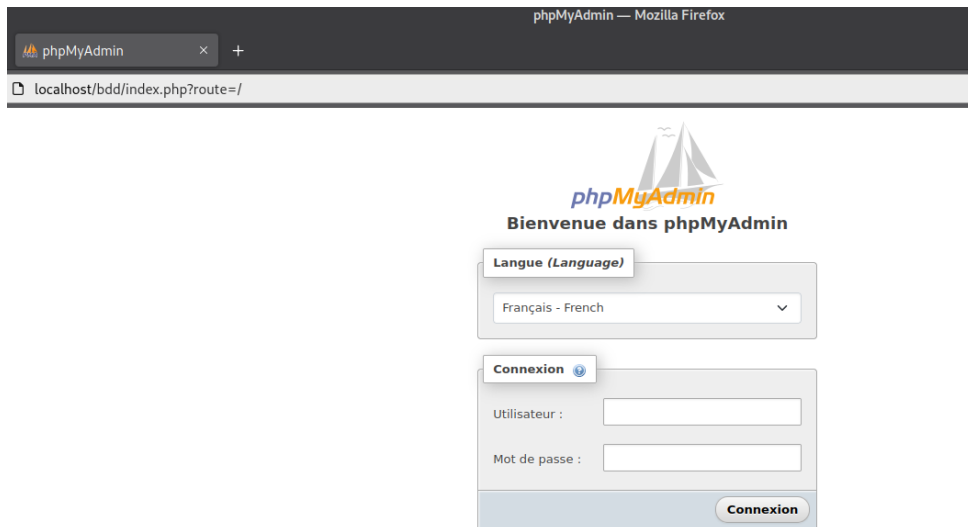
```
cd /var/www/html/sae401/
```

```
sudo mv wordpress/* . ; sudo rm wordpress/ -Rf
```

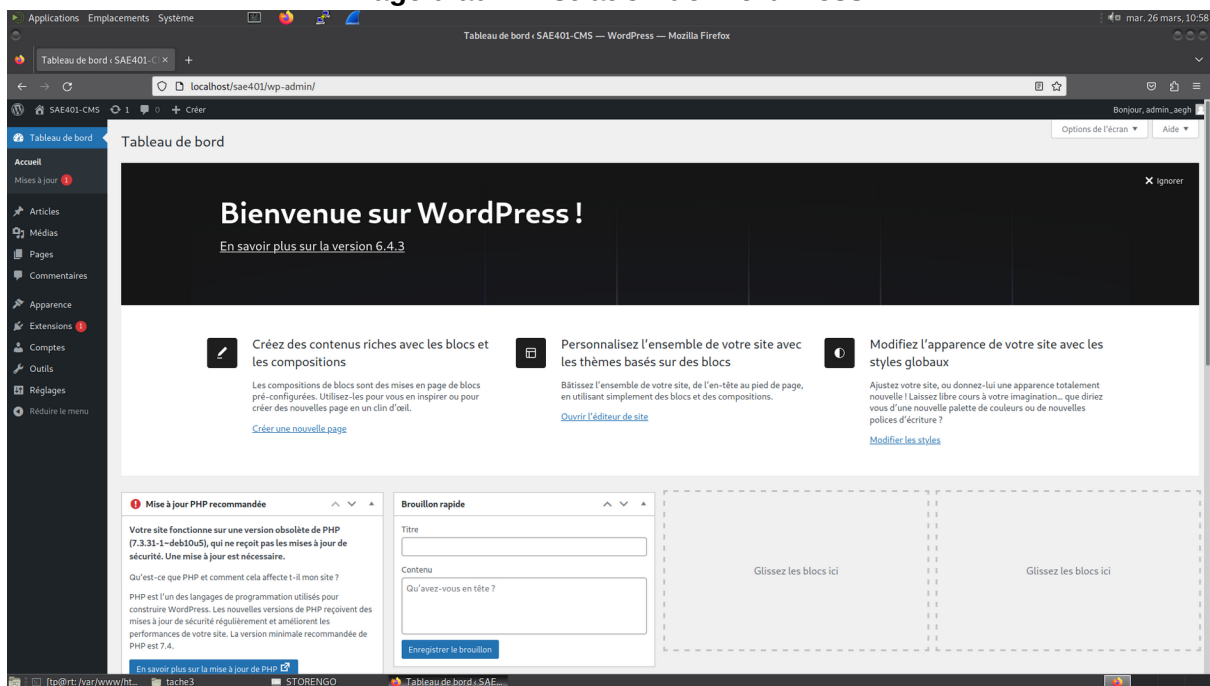
```
sudo chown -R www-data:www-data /var/www/html/
```

```
sudo rm /var/www/html/sae401/wp-config-sample.php
```

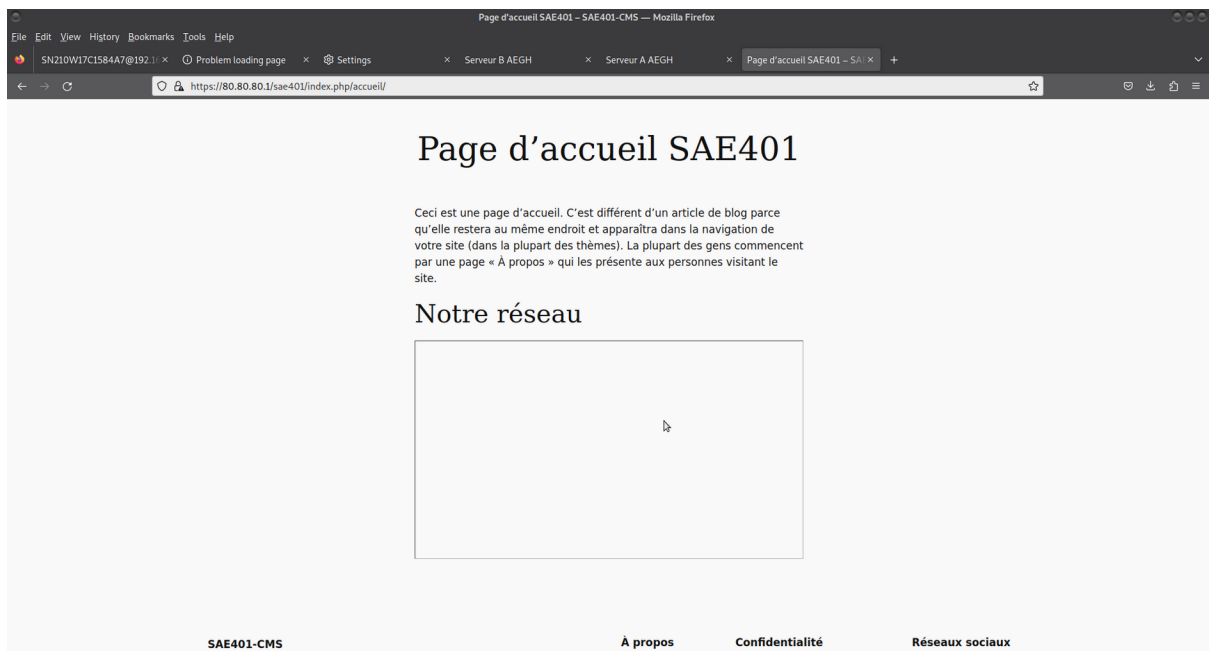
Page d'accueil phpMyAdmin



Page d'administration de WordPress



e) Accessibilité

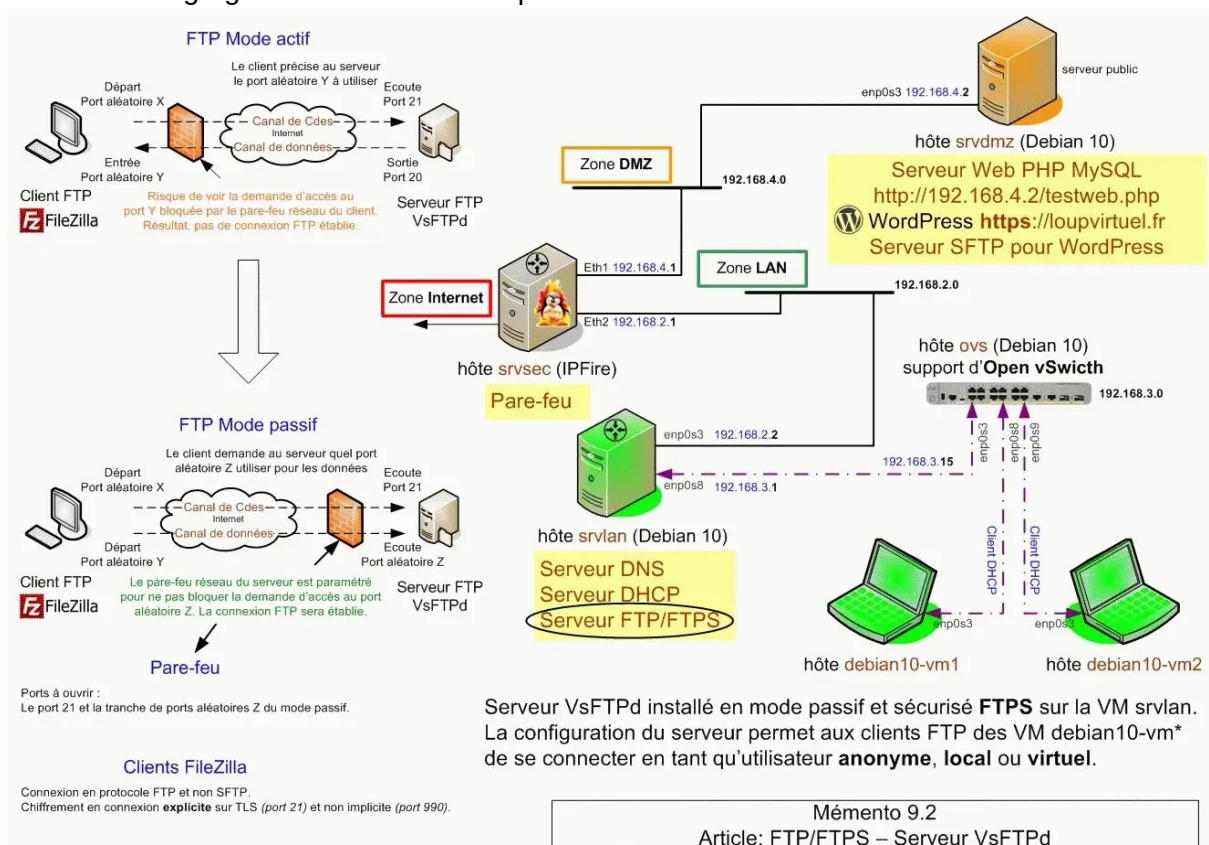


Petit plus :

- blocage de la mention du nom et de la version du serveur Web (cas des requêtes erronées) ! Histoire de ne pas renseigner sur le système utilisé.
- Redirection automatique vers HTTPS
- Activation du module status (statistiques du serveur Web)

2. Installation des serveur FTP/FTPS A et B

Voici une image générale de l'utilité du protocole FTP.



Source : <https://infouloup.no-ip.org/wp-content/uploads/2022/05/ftp-ftpserver-vsftpd-memento-9.2.webp>

a) Mise en place du service FTP

Un serveur FTP se met en place rapidement grâce au paquet Debian. Le paquet permettant la configuration d'un serveur FTP s'appelle **vsftpd**.

sudo apt-get install vsftpd ; sudo nano /etc/vsftpd.conf

The screenshot shows a terminal window on the left and a Wireshark packet capture window on the right. The terminal window displays the installation of vsftpd and the configuration of the service. The Wireshark window shows a packet capture of the FTP protocol, including the welcome message and the login attempt.

```
tp@rt:~$ sudo nano /etc/vsftpd.conf
tp@rt:~$ sudo systemctl restart vsftpd.service
tp@rt:~$ sudo systemctl status vsftpd.service
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor pre
   Active: active (running) since Mon 2024-03-25 16:21:41 CET; 1s ago
   Process: 4602 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=ex
   Main PID: 4603 (vsftpd)
     Tasks: 1 (limit: 4915)
    Memory: 776.0K
   CGroup: /system.slice/vsftpd.service
           └─4603 /usr/sbin/vsftpd /etc/vsftpd.conf

mars 25 16:21:41 rt systemd[1]: Starting vsftpd FTP server...
mars 25 16:21:41 rt systemd[1]: Started vsftpd FTP server.
tp@rt:~$ ftp tp@localhost
ftp> tp@localhost: Name or service not known
ftp> open localhost
ftp> connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
220 Welcome to our FTP server (AEGH) !
Name (localhost:tp): tp
530 Non-anonymous sessions must use encryption.
Login failed.
421 Service not available, remote server has closed connection
ftp> open 10.0.0.1
Connected to 10.0.0.1.
220 Welcome to our FTP server (AEGH) !
Name (10.0.0.1:tp): clientA
530 Non-anonymous sessions must use encryption.
Login failed.
421 Service not available, remote server has closed connection
ftp>
```

The Wireshark window shows a packet capture of the FTP protocol. The packet list shows a sequence of packets including the welcome message and the login attempt. The packet details pane shows the structure of the FTP protocol, including the welcome message and the login attempt.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.003839486	10.0.0.1	10.0.0.1	FTP	106	Response: 220 Welcome to our FTP server (
6	3.983649160	10.0.0.1	10.0.0.1	FTP	80	Request: USER clientA
8	3.983649160	10.0.0.1	10.0.0.1	FTP	115	Response: 530 Non-anonymous sessions must
11	3.983649160	10.0.0.1	10.0.0.1	FTP	72	Request: SYST

b) Les différents modes

Il existe deux modes de connexion à un serveur FTP : soit en actif soit en passif.

La différence réside surtout dans l'usage du protocole. En effet, en **mode actif** c'est le client qui ouvre un port de données et l'échange de données s'effectue vers ce port. S'il y a un routeur ou un firewall entre le client et le serveur, ce mode ne fonctionnera pas. D'où la nécessité d'utiliser le **mode passif** car le serveur choisit une plage de port qui sera ouverte afin que l'échange de données s'effectue par un de ces ports. Cela facilite l'administration des règles du firewall.

```

Fichier  Editor  Affichage  Rechercher  Terminal  Aide
GNU nano 3.2 /etc/vsftpd.conf

# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
ssl_enable=YES
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.key

#établissement du mode passif (nécessaire pour serveur derrière firewall)
#pasv_enable=YES
pasv_min_port=21000
pasv_max_port=21010
allow_anon_ssl=YES
force_anon_logins_ssl=YES
force_anon_data_ssl=YES
force_local_data_ssl=YES
force_local_logins_ssl=YES

#ssl_ciphers=DES-CBC3-SHA
ssl_tlsv1=YES
ssl_sslv2=YES
ssl_sslv3=YES

#implicit_ssl=YES

#contrôle de sécurité (assure que la connexion provient d'une même ip)
pasv_promiscuous=NO
port_promiscuous=NO
pasv_address=10.0.0.1
port_enable=YES

```

c) Sécurisation avec FTPS

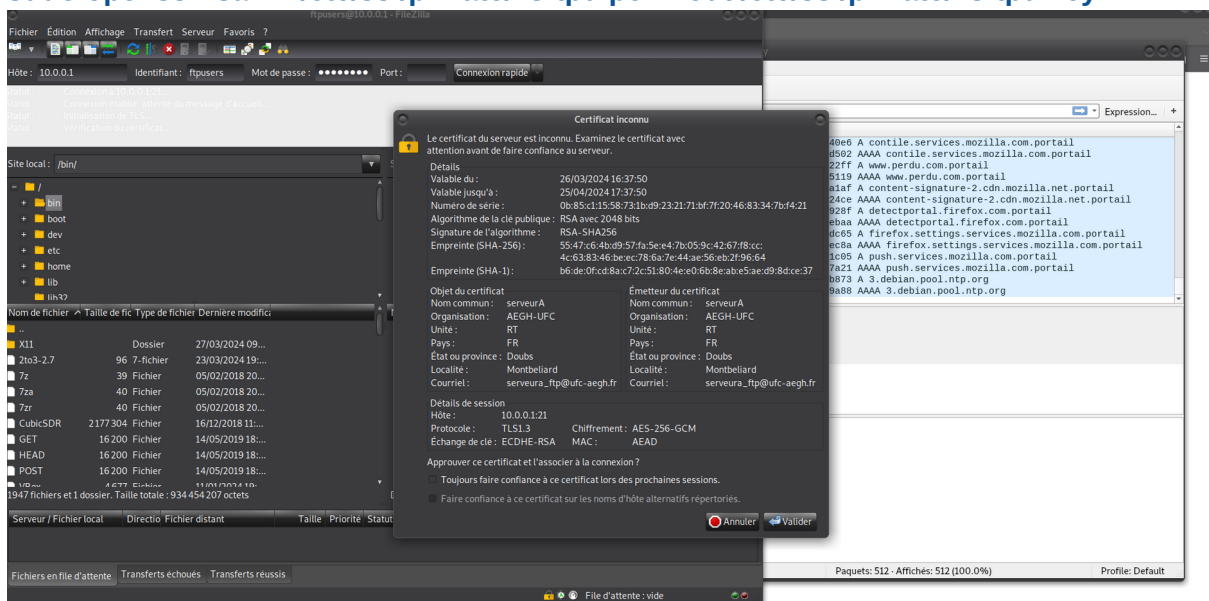
Malheureusement, FTP est un vieux protocole. Il a été conçu lorsque les soucis de sécurité étaient moindres. Par conséquent, les données transitent en clair. Pour y remédier, on ajoute une couche supplémentaire sécurisée via **TLS** lors des connexions FTP : **FTPS**. Il faut donc générer un **certificat** puis **activer la communication SSL**.

commandes tapées dans cet ordre :

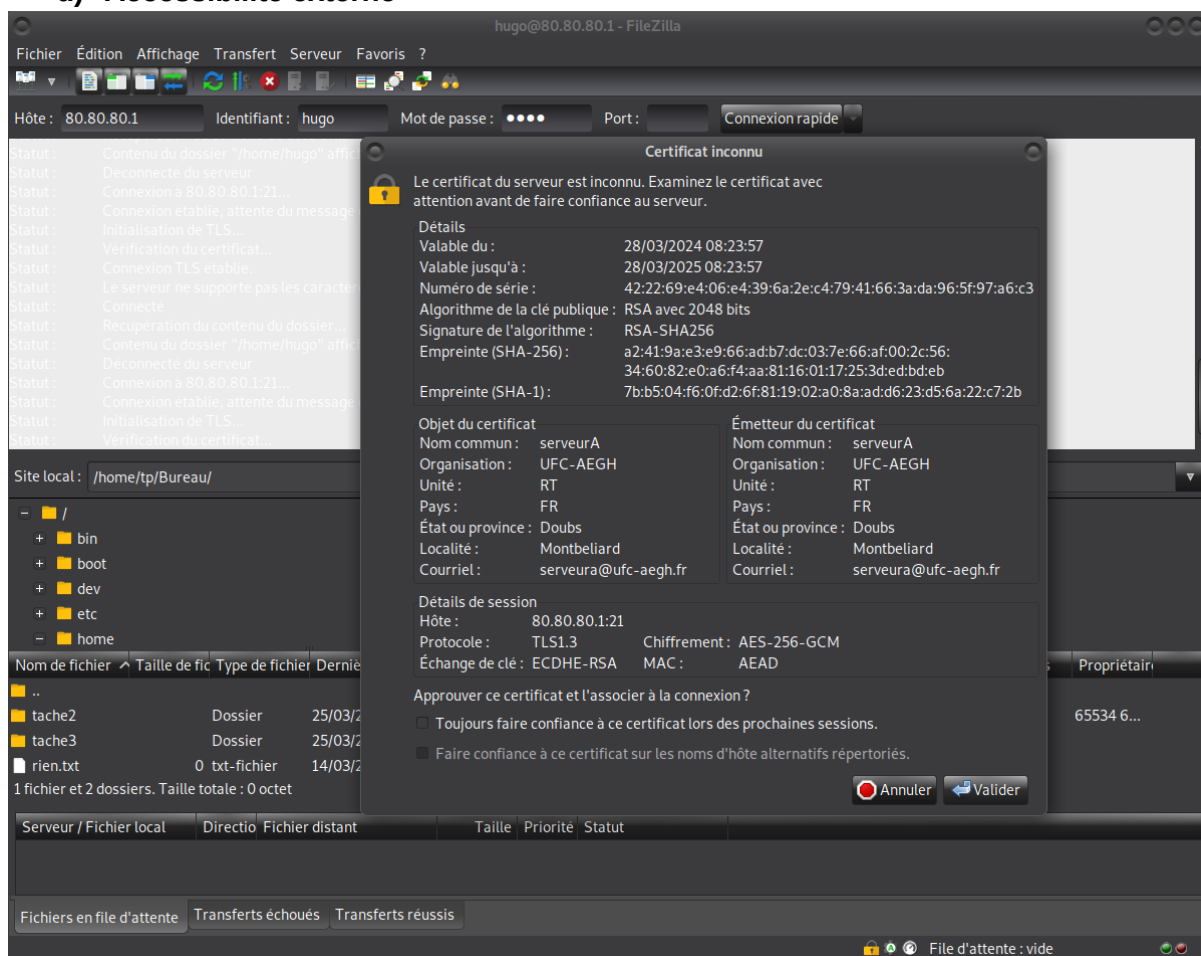
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout

/etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem

sudo openssl rsa -in /etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.key



d) Accessibilité externe



Problème rencontré et résolu :

J'ai rencontré un problème pour déployer **la connexion FTP sécurisée en dehors de mon réseau local**. Je pensais uniquement au fait d'activer le mode passif vu que mon serveur était derrière un firewall. Mais rien ne fonctionnait pour autant. Après maintes tentatives, j'ai décidé de refaire la configuration pas à pas. Mais il fallait plutôt se pencher sur le message d'erreur du client FTP "**ECONNRESET**". En gros, il envoyait un **Client Hello** qui n'avait pas de réponse **car le serveur ne le recevait pas**. Nous avons regardé avec **Hugo** (vu qu'il a fait les règles) mais **aucune règle ne bloquait la connexion et même dans les logs tout se passait normalement**, aucun signe d'anomalie ou message d'erreurs.

J'ai décidé de tester depuis un autre hôte dans le même réseau : la connexion s'établit.

Alors en regardant les règles, on a remarqué le niveau d'inspection **IPS** et une fois passé en **IDS**, la connexion était possible.

En effet, **IPS = Détecter et bloquer**. La connexion était sûrement bloquée à cause du certificat auto-signé. IPS a estimé qu'il était une menace pour le système.

IDS = Détecter et générer une alerte.

Petit plus :

- **force la connexion anonyme via SSL/TLS aussi**
- **interdit à la connexion anonyme de téléverser des fichiers**

Conclusion

A ce niveau, nos deux réseaux offrent des services accessibles en interne et en externe. J'ai mis un minimum de sécurité sur les serveurs indépendamment des règles des firewall.