

Sécurité des serveurs et des systèmes**TP n°10 : Cryptographie et création d'une architecture à clé publique (PKI)
à partir l'outil OpenSSL sous Linux Debian**Durée : 2h

Moyens mis à disposition des étudiants	
Matériels : • 2 ordinateurs fixes ou portables • Switch cisco SG 300-10 • Câbles RJ45	Logiciels et systèmes d'exploitation : • VirtualBox 7.2.6 • Windows 10 ou 11 professionnel • Linux Debian 12.11 Bookworm • Wireshark 4.4.9 • OpenSSL 3.0.18 • Navigateur Internet (au choix)
Divers : Accès à Internet	

Savoir-faire mis en œuvre
❖ Installer et configurer des éléments d'infrastructure ❖ Déployer une solution d'infrastructure ❖ Administrer un système ❖ Tester l'intégration et l'acceptation d'une solution d'infrastructure ❖ Rédiger ou mettre à jour la documentation technique et utilisateur d'une solution d'infrastructure

Conditions de travail :

- Binôme (certains tests peuvent être réalisés avec un autre étudiant si votre poste de travail n'est pas suffisamment performant pour lancer 2 machines virtuelles simultanément).
- **Le compte-rendu de ce TP sera rédigé sous la forme d'une documentation technique (page de garde + sommaire + bibliographie et/ou sitographie)** présentant les étapes d'installation, de configuration, de paramétrage ainsi qu'une phase de recette (les tests) des services déployés via Linux Debian Bookworm.
- Les configurations matérielles requises pour les serveurs et les postes de travail sont présentées en annexes **document 1**.
- L'infrastructure que vous devez mettre en place est présentée en annexes **document 2**.

Contexte professionnel : Madame Bourgeois, Responsable de l'entreprise « Jardins de Saint-Eloi », souhaite sécuriser les transactions (échanges) réalisées entre son site web et les clients, internautes.

Madame Bourgeois a sollicité l'ESN (Entreprise de services numériques) C2I Caraïbes, implantée en Guadeloupe et en Martinique, afin de mener à bien ce projet.

Ce projet sera mené par un Développeur de C2I Caraïbes, spécialiste dans la création d'applications métiers basées sur les technologies web traditionnelles et 3.0.

Annexe I : Infrastructure réseau à réaliser.

MISSION : Vous êtes stagiaire chez C2I Caraïbes, le Responsable de la MOE fait appel à vous pour aider le Développeur dans la réalisation d'une solution technique afin de sécuriser à l'aide des protocoles SSL/TLS le site web hébergé.

Etape 1 : Mettre à jour les paquets et installer OpenSSL

Avant toute installation : `Ent#sudo apt upgrade`

Installez le paquet :

`#sudo apt install openssl`

Pour vérifier l'installation :

`#openssl version`

Le fichier de configuration OpenSSL : le fichier principal est généralement : `/etc/ssl/openssl.cnf`

Pour l'éditer :

`#sudo nano /etc/ssl/openssl.cnf`

Ce fichier permet de configurer :

- les autorités de certification
- les paramètres TLS
- les politiques de certificats
- les extensions X509

Etape n°2 : OpenSSL et le chiffrement symétrique

Le **chiffrement symétrique** utilise la **même clé pour chiffrer et déchiffrer**. C'est rapide et très utilisé pour protéger des fichiers ou des flux de données. Les algorithmes les plus courants :

- **AES** (recommandé)
- **Triple DES**

1. La commande « `$openssl enc -list` », affiche les algorithmes de chiffrement disponibles.
2. Depuis le répertoire personnel de « **tepson** », créez un fichier texte « **chiffre_sym.txt** » avec pour contenu « **HelloWorld ! Ceci est un test de chiffrement symétrique** ».
3. Chiffrez le fichier « **chiffre_sym.txt** » à l'aide de la commande suivante, puis indiquez le rôle de chaque option et paramètre :
 - `$openssl enc -aes-256-cbc -salt -in chiffre_sym.txt -out chiffre_sym.enc`
4. Supprimez le fichier « **chiffre_sym.txt** », indiquez la commande utilisée.
5. A l'aide la commande suivante, déchiffrez le fichier « **chiffre_sym.enc** », puis indiquez le rôle de chaque option et paramètre :
 - `$openssl enc -d -aes-256-cbc -in chiffre_sym.enc -out chiffre_sym.txt`
6. Editez le fichier « **chiffre_sym.txt** », que constatez-vous ? Que représente l'option « **-salt** » pour ce mode de chiffrement.

Etape n°3 : OpenSSL et le chiffrement asymétrique

Le **chiffrement asymétrique** avec **OpenSSL** utilise **deux clés différentes** :

- **clé publique** → pour chiffrer
- **clé privée** → pour déchiffrer

Contrairement au chiffrement symétrique (ex : **AES**), les clés ne sont pas identiques.

Les algorithmes asymétriques courants : **RSA**, **Elliptic Curve Cryptography (ECC)**

7. Depuis le répertoire personnel de « **tepson** », créez un fichier texte « **chiffre_asym.txt** » avec pour contenu « **HelloWorld ! Ceci est un test de chiffrement asymétrique** ».

8. Générez une clé privée à l'aide de la commande suivante, puis indiquez le rôle de chaque option et paramètre :
 - **\$openssl genpkey -algorithm RSA -out clepriv.pem -pkeyopt rsa_keygen_bits:2048**
9. Extraire la clé publique, à partir de clé privée, à l'aide de la commande suivante. Puis, indiquez le rôle de chaque option et paramètre :
 - **\$openssl rsa -pubout -in clepriv.pem -out clepub.pem**
10. Listez le contenu du répertoire personnel de « **tepson** ». Que constatez-vous ? Maintenant, éditez le contenu des deux fichiers créés. Que constatez-vous ?
11. Chiffrez le fichier « **chiffre_asym.txt** » avec la « **clé publique** », à l'aide de la commande suivante et indiquez le rôle de chaque option et paramètre :
 - **\$openssl pkeyutl -encrypt -pubin -inkey clepub.pem -in chiffre_asym.txt -out chiffre_asym.enc**
12. Supprimez le fichier « **chiffre_asym.txt** », indiquez la commande que vous avez saisie.
13. Maintenant, avec la clé privée car seul son propriétaire pourra, déchiffrez le fichier « **chiffre_asym.enc** » et indiquez le rôle de chaque option et paramètre :
 - **\$openssl pkeyutl -decrypt -inkey clepriv.pem -in chiffre_asym.enc -out chiffre_asym.txt**
14. Editez le fichier « **chiffre_asym.txt** », que constatez-vous ?

Etape n°4 : OpenSSL et le hachage (fonction de hachage)

Les fonctions **SHA** servent à calculer une **empreinte cryptographique/numérique (hash/condensat)** d'un fichier ou d'un texte. SHA appartient à la famille **Secure Hash Algorithm**.

Un hash/condensat sert à :

- vérifier l'intégrité d'un fichier
- stocker des mots de passe
- signer des données
- vérifier des téléchargements

Les versions souvent utilisées sont : **SHA-256** (très utilisé) et **SHA-512**.

15. Depuis le répertoire personnel de « **tepson** », créez un fichier texte « **hachage.txt** » avec pour contenu « **HelloWorld ! Ceci est un hash.** ».
16. La commande « **\$openssl list -digest-algorithms** », affiche les algorithmes de hachages disponibles.
17. A l'aide de la commande suivante, créez un condensat du fichier « **hachage.txt** ». Puis, redirigez le résultat dans un fichier texte nommé « **hash.txt** » et affichez son contenu :
 - **\$openssl dgst -sha256 hachage.txt > hash.txt**
 - **\$cat hash.txt**
18. Ensuite, modifiez le fichier « **hachage.txt** » afin de supprimer le point « **.** » à la fin de la phrase « **Ceci est un hash** » et enregistrez. Ensuite, créez à nouveau un condensat du fichier « **hachage.txt** » et redirigez, **en complément**, le résultat dans le fichier « **hash.txt** » et affichez son contenu :
 - **\$openssl dgst -sha256 hachage.txt >> hash.txt**
 - **\$cat hash.txt**
19. Que constatez-vous ? Indiquez le rôle de chaque option et paramètre.
20. Maintenant, corrigez le contenu du fichier « **hachage.txt** » en rajoutant le point « **.** », enregistrez et créez à nouveau un condensat :
 - **\$openssl dgst -sha256 hachage.txt >> hash.txt**
 - **\$cat hash.txt**
21. Que concluez-vous ?

Etape n°5 : Réalisation d'une infrastructure à clé publique (PKI) personnalisée à partir de OpenSSL

OBSERVATIONS : Pour créer et gérer *une PKI (Infrastructure à clé publique)* avec la dernière version de OpenSSL, il faut mettre en place une arborescence de répertoires et créer un fichier « *openssl.cnf* » qui contiendront *les éléments constitutifs et de configuration* propres à votre *PKI personnalisée*.

Par ailleurs, créer une PKI (Public Key Infrastructure – Infrastructure à clé publique) avec OpenSSL consiste à mettre en place :

1. une **Root CA (une autorité racine)**
2. éventuellement une **Intermediate CA (autorité de certification intermédiaire)**
3. et des **certificats signés** pour serveurs ou utilisateurs.

Dans notre cas pratique, nous *omettrons l'étape de la création d'une CA intermédiaire* et *signerons les certificats « serveurs » avec la clé privée de notre CA racine*.

Notez aussi *qu'à l'installation de OpenSSL*, un fichier « *openssl.cnf* » est automatiquement créé et stocké à l'emplacement « */etc/ssl/* ». *Ce fichier ne doit en aucun cas être supprimé, déplacé et surtout modifié dans le cadre d'une PKI personnalisée* par *risque d'impacter le bon fonctionnement* et la *sécurité de votre OS Linux Debian 12*.

Enfin, *OpenSSL est très sensible à la « casse »* c'est-à-dire il distingue les minuscules des majuscules notamment dans la syntaxe des paramètres du fichier « *openssl.cnf* ».

De plus, il faut être vigilant et faire correspondre les FQDN des serveurs, pour lesquels les certificats seront générés, avec les CommonName (CN) saisis lors de la création des certificats :

- Exemples de FQDN : *www.jardinsstelo.com, ftp.jardinsstelo.com, mail.jardinsstelo.com, ...*

Cela évite des erreurs émises par les protocoles SSL/TLS comme « *SSL_ERROR_BAD_CERT_DOMAIN* »...

A. Création de l'arborescence de répertoires et du fichier « *openssl.cnf* » pour la configuration d'une PKI personnalisée

22. Créez, depuis le répertoire personnel de l'administrateur « **root** » et à partir des commandes suivantes, une structure de répertoires et sous-répertoires standards pour une PKI réalisée via OpenSSL :

```
#su
#cd
#mkdir -p ~/pki/{certs,crl,newcerts,private}
#chmod 700 ~/pki/private
#touch ~/pki/index.txt
#echo 1000 > ~/pki/serial
```

Voici le résultat de la structure attendue :

```
pki/
├── certs/
├── crl/
├── newcerts/
├── private/
├── index.txt
└── serial
```

23. Précisez le rôle et/ou le contenu de chaque répertoire et de chaque fichier.

24. Créez le fichier « *openssl.cnf* » à l'emplacement suivant : « */root/pki* » et ajoutez-y les paramètres de configuration suivants :

Remarque : *RootCAJSE* → *CA Racine de JSE (Jardins de Saint-Eloi)*

```

[ ca ]
default_ca = CA_default

[ CA_default ]
dir = /root/pki
certs = $dir/certs
crl_dir = $dir/crl
database = $dir/index.txt
new_certs_dir = $dir/newcerts
certificate = $dir/certs/cajse.crt
serial = $dir/serial
private_key = $dir/private/cajse.key
default_days = 3650
default_md = sha256
policy = policy_loose
x509_extensions = v3_ca

[ policy_loose ]
commonName = supplied

[ req ]
default_bits = 4096
default_md = sha256
prompt = no
distinguished_name = dn
x509_extensions = v3_ca

[ dn ]
CN = RootCAJSE

[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_server ]
basicConstraints = CA:false
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = jardinsstelo.com
DNS.2 = www.jardinsstelo.com

```

25. Indiquez l'objectif de chaque section « [] » ainsi que le rôle des paramètres qui y sont rattachés.

B. Création de la clé privée et du certificat de l'autorité racine (RootCA)

Créez la clé privée de l'autorité racine (Root CA) depuis la commande suivante, précisez comme « *pem pass phrase* » le mot de passe de l'administrateur « **root** » :

```

#openssl genrsa -out ~/pki/private/cajse.key 4096
#chmod 600 ~/pki/private/cajse.key

```

26. Indiquez le rôle de chaque paramètre et option.

Créez le certificat de l'autorité racine depuis la commande suivante :

```
# openssl req -x509 -new -nodes -key ~/pki/private/cajse.key -sha256 -days 3650 -out ~/pki/certs/cajse.crt  
-config ~/pki/openssl.cnf
```

27. Indiquez le rôle de chaque paramètre et option.

28. Vérifiez le contenu du certificat du CA avec la commande :

```
#openssl x509 -in ~/pki/certs/cajse.crt -text -noout | more (tapez sur « entrée » pour faire défiler)
```

C. Création d'une clé privée du CA racine et d'un certificat serveur pour « srvlinux » signé cette même CA racine (Root CA)

D'abord, générez une clé privée pour le serveur « srvlinux », à l'aide de la commande :

```
#openssl genrsa -out ~/pki/private/srvlinux.key 2048
```

Puis, créez une demande de signature de certificat au format CSR (Certificate Signing Request). Elle sert à demander à une autorité de certification de créer un certificat pour une clé publique. Dans les outils comme OpenSSL, le certificat au format CSR est une étape essentielle pour obtenir un certificat signé.

Ainsi, saisissez la commande suivante :

```
#openssl req -new -key ~/pki/private/srvlinux.key -out ~/pki/certs/srvlinux.csr
```

Précisez les informations suivantes lors de la création du certificat numérique :

- PAYS : FR
- Etat/province : Guadeloupe
- Localité : Les Abymes
- Organisation/entreprise : Jardins de StEloi
- Section : Entreprise
- Nom commun : **www.jardinssteloi.com**
- Email : **m.bourgeois@jardinssteloi.com**

Conservez les valeurs par défaut (vide) pour les autres informations, il suffit de taper sur la touche « Entrée » pour laisser vide.

Ensuite, il vous faut signer le certificat « serveur » avec la clé privée du CA racine à l'aide de la commande suivante :

```
#openssl ca -config ~/pki/openssl.cnf -extensions v3_server -in ~/pki/certs/srvlinux.csr -out  
~/pki/certs/srvlinux.crt
```

29. Indiquez l'objectif de cette commande puis précisez le rôle de chaque option et paramètre.

30. Observez et commentez le contenu des fichiers : « **/root/pki/serial** » et « **/root/pki/index.txt** ».

31. Vérifiez la **validité du certificat « signé »** avec la commande suivante, indiquez le résultat obtenu :

```
#openssl verify -CAfile ~/pki/certs/cajse.crt ~/pki/certs/srvlinux.crt
```

D. Installation du certificat de l'autorité de certification (CA) sur le serveur « srvlinux » basé sous Linux Debian

Pour que le système d'exploitation (ici Linux Debian) fasse confiance à votre autorité de certification (CA), il faut installer le certificat racine « **RootCAJSE** » sur le serveur « **srvlinux** ».

Ce certificat racine « **RootCAJSE** » sera enregistré à l'emplacement suivant : « **/usr/local/share/ca-certificates/** » à partir de la commande suivante :

```
#sudo cp ~/pki/certs/cajse.crt /usr/local/share/ca-certificates/
```

32. Recherchez à l'emplacement suivant « **/etc/ssl/certs** », le fichier nommé « **cajse.pem** ». Que constatez-vous ?

Maintenant à l'aide de la commande suivante, mettez à jour la liste des autorités de certification présents sur votre système Linux Debian :

```
#sudo update-ca-certificates
```

33. Observez le contenu du répertoire « **/etc/ssl/certs** », recherchez le fichier « **cajse.pem** » est-il présent ? Justifiez la génération du même certificat mais au format « **PEM** ».

Etape n°6 : Sécurisation du site « Les Jardins Saint-Eloi » basé sous le service web Apache2 avec les éléments de votre PKI et des protocoles SSL/TLS

Pour sécuriser le site web des Jardins de Saint-Eloi, il faut d'abord créer une chaîne de confiance entre les certificats finaux (certificats « serveur ») et les autorités de certification (racine et/ou intermédiaire).

A l'aide de la commande suivante, établissez cette chaîne/relation de confiance entre votre serveur « **srvlinux** » et votre autorité de certification racine « **RootCAJSE** » et redirigez la dans le fichier « **fullchain.crt** » :

```
#cat ~/pki/certs/srvlinux.crt ~/pki/certs/cajse.crt > ~/pki/certs/fullchain.crt
```

34. Proposez une commande permettant de consulter et surtout d'appréhender/de comprendre le contenu du fichier « **fullchain.crt** ». Que remarquez-vous ? Justifiez !

Maintenant, il faut configurer le « **VirtualHost** » du site géré par le service web Apache 2 pour la prise en compte de la clé privée et du certificat x509 du serveur « **srvlinux** » et de la chaîne de confiance. Editez le fichier « **sainteloi.conf** » et configurez-le avec les paramètres suivants :

- **Port** : 443
- **SSLEngine** : activé
- **SSLprotocol** : toutes les versions en plus de TLSv1.3
- **SSLCertificateFile** : chemin d'accès au fichier contenant la chaîne de confiance entre le certificat numérique du serveur et le certificat du CA racine
- **SSLCertificateKeyFile** : chemin d'accès au fichier contenant la clé privée du serveur « **srvlinux** »
- **SSLCipherSuite** : le plus élevé, pas d'algorithme MD5 et de valeur vide → « **NULL** »
- **SSLHonorCipherOrder** : activé

35. Indiquez le rôle des paramètres et des valeurs associées pour « **SSLCipherSuite** » et « **SSLHonorCipherOrder** » dans le fichier de configuration « **sainteloi.conf** ».

Enfin, réalisez les opérations suivantes :

- Activez le module **SSL** pour le service web Apache 2 à partir de la commande « **a2enmod** ».
- Relancez le service web Apache 2 à partir de la commande « **systemctl** ».
- Maintenant, désactivez et activez le site des Jardins de St-Eloi depuis la commande « **a2dissite** » et « **a2ensite** ».
- Enfin, relancez le service web Apache 2 depuis la commande « **systemctl** ».

Etape n°7 : Tests du dispositif de sécurisation du site web « Jardins de Saint-Eloi » géré par OpenSSL et basé sur le service Apache 2

Tout d'abord, il faudra utiliser l'outil logiciel SSH pour récupérer le certificat de l'autorité de certification racine « **RootCAJSE** » → fichier représentant le certificat numérique du CA « **cajse.crt** » pour l'intégrer dans le navigateur web de vos postes clients « **poste1** » et/ou « **poste2** ».

Le service SSH s'appuie sur deux clés, une clé publique et une clé privée, pour sécuriser les échanges entre les serveurs. Pour configurer une connexion SSH, vous devez générer une clé de chiffrement privée qui sera stockée sur le système local, serveur « **srvlinux** ».

Ensuite, vous devez copier une clé de chiffrement publique du système local, serveur « **srvlinux** », vers le système distant, poste « **poste0/1** ».

Cette configuration doit aussi permettre une connexion sécurisée sans demande de mot de passe. La clé publique est souvent distribuée sur les machines distantes permettant ainsi au serveur à l'origine de la demande de connexion de s'authentifier, c'est-à-dire attester/confirmer son identité.

Les clés « **id_rsa**, fichier contenant la clé privée et **id_rsa.pub**, fichier contenant la clé publique » seront générées et enregistrées dans le répertoire « **~/ssh** », présent dans le répertoire personnel du compte « **root** ». Je ne reviendrai pas sur l'explication du symbole « **~** ». Par contre, le « **.** » point qui précède le nom du répertoire « **ssh** » est spécifique aux systèmes Unix/Linux et consiste à cacher ce dernier. Pour le visualiser dans la liste des répertoires, utilisez la commande « **ls** » avec l'option « **-al** ».

36. Proposez une procédure pas à pas pour récupérer le certificat du CA racine « **cajse.crt** » sur le serveur « **srvlinux** » et le stocker sur le navigateur web de vos postes clients « **poste0/1** ».

Remarques :

- Vous pouvez créer les clés privée et publique à l'aide des commandes suivantes :
 - **# ssh-keygen -t rsa -b 2048** //pour générer des clés de 2048 bits (minimum recommandé et pour le chiffrement l'algorithme utilisé sera RSA).
- Les clés « **id_rsa**, fichier contenant la clé privée et **id_rsa.pub**, fichier contenant la clé publique » seront générées et enregistrées dans le répertoire « **~/ssh** », présent dans le répertoire personnel du compte « **root** ». Je ne reviendrai pas sur l'explication du symbole « **~** ». Par contre, le « **.** » point qui précède le nom du répertoire « **ssh** » est spécifique aux systèmes Unix/Linux et consiste à cacher ce dernier. Pour le visualiser dans la liste des répertoires, utilisez la commande « **ls** » avec l'option « **-al** ».

Afin de vérifier que les informations qui sont transmises sont bien sécurisées, vous devez capturer et analyser les trames Ethernet échangées entre le poste de travail (**poste0** ou **poste1**) et le serveur web « **srvlinux** », ci-dessous, la procédure à suivre pas à pas :

- 37. Indiquez ce que vous avez constaté quand vous accédez au site web et expliquez ce nouveau résultat.
- 38. Indiquez la démarche que vous avez mise en œuvre pour effectuer vos tests, à l'aide de Wireshark.
 - Remarque : Vous devez préciser également le « **filtre d'affichage** » retenu.
- 39. Affichez le certificat numérique et listez/relevez les informations qui s'y trouvent.
- 40. Concluez en justifiant la sécurisation ou non des données échangées.

Bon travail !!!