Tp N°1

Mise en œuvre d'un serveur web Sécurisé

But de l'atelier

Le but de ce TP est de mettre en œuvre un serveur web Apache2 et le sécuriser grâce à l'authentification et à SSL :

- Authentification simple
- Authentification chiffrée par utilisation de certificats X509
- Analyse du trafic circulant entre le client et le serveur.

Préparation de l'environnement de travail :

1. Installer VirtualBox,
2. Télécharger l'image iso d'Ubuntu 22 (la dernière version stable)
3. Créer les deux machines virtuelles, une pour le serveur et l'autre le client.
4. Installer les modules aditions Guest sur les deux machines
    a. Apt update
    b. Apt install -y build-essentials linux-headers$(uname -r)
    c. Monter le CD virtuel et installer les modules nécessaires
5. Mettre les cartes réseaux des deux machines en mode bridged pour qu'elles soient sur le même réseau et aient au même temps accès à internet (tester la connectivité via le ping).

Partie 1 : Installation et configuration de http, protection des répertoire web à l'aide d'un mot de passe.

1. Introduction

Apache est le logiciel qui est chargé de la fonction de serveur Web. Il est inclus dans toutes les distributions de Linux et existe également pour Windows. Il est totalement gratuit et représente une grosse part des serveurs Internet à travers le monde.

SSL (Secure Socket Layer) est un protocole de sécurisation des échanges de données. Il a été conçu pour assurer la sécurité des transactions sur Internet et est intégré depuis 1994 dans les navigateurs. Plusieurs versions de ce protocole existent :

- la version 2.0 développée par Netscape (inventeur de ce protocole);
- la version 3.0, la plus répandue,
- et la version 3.1 rebaptisée TLS (Transport Layer Security, RFC 2246) et standardisé par l'IETF.

SSL est un protocole de niveau 4 du modèle OSI et est conçu pour fonctionner de manière transparente pour les applications qui l'utilisent. De par sa simplicité de mise en oeuvre et le niveau de sécurité qu'il offre, SSL est aujourd'hui un protocole de sécurité incontournable pour les applications sécurisées du monde Internet.

2. Configuration matérielle :

Pour les besoins du TP, vous travaillerez sur des machines virtuelles. Deux machines Linux une pour le client et l'autre pour le serveur. La machine <<Client>> vous servira à tester votre serveur web ainsi que sa fiabilité depuis le réseau. Dans cette machine vous devez installer les outils OpenSsl, whireshark et *TinyCA* . Le <<Serveur>> sera votre serveur web Apache. Aussi sur cette machine vous devez installer les mêmes outils.

2.1. Configuration des machines

Même si Apache est aussi déjà installé sur votre machine, vous devez le réinstaller en cours de TP, le package se trouve dans le répertoire /RPM du CD REDHAT.

Pour l'installer *:* ***rpm -ivh httpd-2.0.52-3.i386.rpm***

Ou via le réseau en utilisant la commande apt: ***apt install apache2***

2.2. Configuration d'Apache 2.5 Emplacement des pages web

Configurer le répertoire des utilisateurs nommé comme **/etc/apache2/mods-available/userdir.conf**

- Quel est le format de ce fichier ?
- Quelle directive permet de définir l'emplacement des fichiers web ?

Pour le besoin du TP nous allons définir l'accès au serveur en utilisant le répertoire associé à l'utilisateur *admin*. Donc on pourra y accéder avec l'url http://serveur/~admin.

- Créer un utilisateur qui se nomme ***admin*** –
- Créer le répertoire ***/home/admin/web***
- Créer un fichier **index.html** dans ce répertoire qui contient votre CV en HTML de base (saisie à l'aide du gedit ou vim)

Repérez le tag <***IfModule mod_userdir.c***> dans le fichier **/etc/apache2/mods-available/userdir.conf** .

- Comment peut-on activer l'utilisation des répertoires utilisateurs ?

2.2.1. Démarrage du démon

Redémarrer le service http : #**systemctl restart apache2**

Maintenant essayer de vous connectez au répertoire de l'utilisateur admin

- Quesque que vous constatez ?

Pour résoudre ce problème il faudrait taper :

chmod 771 /home/admin, ***chmod 771 /home/admin/web***, ***chmod 644 /home/admin/web/index.html***

- Que signifient ces commandes ?

Retester l'accès aux pages de l'utilisateur *admin*

2.3. Sécurisation par authentification

Il est parfois impératif de protéger certains répertoires du serveur pour éviter que n'importe qui puisse y accéder. Que ce soit par Internet ou en Intranet, il y a différentes méthodes de procéder : utiliser des scripts serveurs (PHP, ASP, CFM...) pour interdire la lecture de manière sélective et dynamique ou utiliser plus simplement les fonctions de protection de répertoires offertes par Apache. Le module nécessaire à l'authentification est *mod_auth*. Il est chargé par défaut dans la configuration d'*Apache*.

.htaccess* et *.htpasswd

La protection d'un répertoire particulier se fait de deux manières différentes : soit on la configure dans le fichier de configuration d'Apache, soit on crée un fichier nommé *.htaccess* (le point fait partie du nom de fichier) qui sera placé dans le répertoire à protéger. La protection des répertoires par le fichier *httpd.conf* est du seul ressort du webmaster.

Cette commande devra être écrite pour chaque répertoire. Ce système est assez lourd à gérer au niveau des droits et nécessite un redémarrage d'Apache pour prendre en compte la nouvelle configuration. Les fichiers *.htaccess* (point htaccess) sont plus faciles à gérer. Ils s'appliquent à un répertoire et à tous ses sous-répertoires, mais peuvent être modifiés par un autre fichier *.htaccess* dans un sous-répertoire. Si vous souhaitez que le dossier */home/admin/web* soit protégé via ce fichier *.htaccess*, il vous faudra ajouter la directive *AllowOverride AuthConfig entre les tags <Directory /home/admin/web> et < /Directory>*. Cela implique que tous ses sous-dossiers vont également accepter cette directive.

- Que signifie *AllowOverride AuthConfig*?

Un fichier *.htaccess* est un fichier texte simple contenant des commandes Apache comme celles-ci :

AuthUserFile password/.htpasswd AuthGroupFile /dev/null AuthName "Identification"
AuthType Basic <Limit GET POST> require valid-user </Limit>

Quelques explications :

- *AuthUserFile*: c'est le nom et le chemin d'accès du fichier qui contiendra les mots de passe. S'il ne commence pas par un slash (/), ce sera un sous-répertoire du serveur web. Avec l'exemple ci-dessus, mes mots de passe seront dans */etc/httpd/password/.htpasswd*. Dans la pratique il vaudra mieux placer ce fichier en dehors du site web pour que personne ne puisse y accéder.

- *AuthGroupFile*: pointe toujours vers */dev/null*. Il faut que cette ligne soit présente.
- *AuthName* : c'est le texte qui apparaîtra dans la fenêtre demandant les mots de passe.
- *AuthType* : L'authentification est en générale <<basic>>. Les mots de passe sont alors envoyés en clair sur le réseau. Pour sécuriser davantage l'accès, on peut utiliser la méthode d'authentification « *digest* » qui crypte les mots de passe en *MD5* (clef de cryptage). Mais ce système n'est supporté que par certains navigateurs (Opera 4+, Internet Explorer 5+, Amaya).
- *Limit*: C'est ici qu'on va indiquer ce qui est autorisé et interdit dans le répertoire.

Les commandes GET et POST indiquent la récupération de pages web et la réponse à certains Formulaires.

- *Require*: On peut entrer ici <<valid-user>>, ce qui accepte tous les utilisateurs qui ont un (login : mot de passe) dans *.htpasswd*. Mais on peut aussi limiter à un ou plusieurs utilisateurs précis: *<<require user pierre paul techno>>*. Dans ce cas les utilisateurs sont séparés par des espaces.

Testez votre configuration

2.4. Générer des mots de passe

Pour créer le fichier *.htpasswd* sous Linux, vous générez les mots de passe avec la commande ***htpasswd*** sous la forme suivante :

htpasswd -c .htpasswd username

On vous demandera alors un mot de passe qu'il faudra répéter deux fois et cela ajoutera au fichier *.htpasswd* de ce répertoire une ligne de ce type: username:v3l0KWx6v8mQM Si vous ne disposez pas d'une machine Unix/Linux, vous pouvez utiliser des générateurs de mots de passe sur le web:

Raptor.golden.net: http://home.golden.net/generator/htpasswd file generator: http://www.wells.org.uk/htpasswdgen.html

2.5. Faiblesses de l'authentification

Relancez Apache par */etc/init.d/httpd* restart vérifiez que le serveur web demande une authentification pour la consultation de la page web. Refaites le test en capturant le trafic avec *Ethereal* et notamment les trames concernant l'authentification: *401 authorization required*

- Qu'est ce que vous constatez ?

Essayer de capturer la signature MD5 de votre mot de passe. Utiliser le logiciel *John the Ripper password cracker* (à télécharger de http://www.openwall.com/john/) pour le décrypter .

- Conclure sur la protection par authentification avec mot de passe en MD5.

Partie 2 : utilisation de TinyCA et OpenSSL afin de créer une autorité de certification (CA) puis sécurisation de httpd par SSL

1. TinyCA

TinyCA est un utilitaire graphique permettant de créer des certificats X.509. Il se base sur l'utilitaire de Cryptographie *OpenSSL* qui implémente SSLv2/v3 (Secure Socket Layer) et TLSv1 (Transport Layer Security).

Avant toute manipulation, vérifiez que *TinyCA* est installé sur votre machine, sinon ***apt-get install TinyCA***. Pour lancer TinyCA, il suffit de taper dans une console ***tinyca2***.

1.1. Création de l'autorité racine

Au lancement de TinyCA, vous avez une fenêtre qui s'affiche vous demandant un ensemble de paramètres à saisir pour créer la création de l'autorité racine.

Il faut saisir "racine" dans le Common Name et le nom. Vous fixerez la durée de validé du Certificat égale à une année. Pour l'empreinte de la clé vous laisserez l'algorithme par défaut. Par la suite cliquez sur Valider. Vous aurez alors une nouvelle fenêtre qui s'affichera, il suffira de valider.

1.2. Signature de Certificats

A présent vous allez créer un certificat serveur. La procédure de création d'un certificat, signé par une autorité de confiance se fait en deux étapes :

- Préparation du certificat (création de requête de signature)
- Signature de la requête

1.2.1. Préparation d'un certificat Serveur

Dans l'onglet principal, sélectionnez Requêtes de certification, avec le bouton droit de votre souris choisissez création de nouvelle requête.

- Qu'elles sont les algorithmes de chiffrement disponibles ?
- Qu'elles sont les algorithmes de Hashage (empreinte) disponibles ?

Ici il faut spécifier "serveur" comme nom pour cette requête et valider.

2.1.1. Signature des certificats par l'autorité racine

Dans cette deuxième étape il faut signer la requête précédente. En fait, c'est grâce au CA racine que vous allez signer cette requête. Dans l'onglet principal, sélectionnez la requête serveur et choisissez signer pour pouvoir créer le certificat. Validez et sélectionnez un certificat serveur. Le nouveau certificat serveur apparait dans l'onglet Certificat.

- Affichez son contenu, et commentez les informations affichées ?

Il est à noter que les certificats crées par l'intermédiaire de *TinyCA* sont disponibles dans le répertoire. *./tinyca*.

2.Open SSL

Pour cette partie vous allez refaire l'ensemble des étapes vues précédemment, mais en utilisant directement Open SSL en lignes de commandes. Pour comprendre les commandes qui suivent, utilisez la le manuel disponible avec cette commande via ***man openssl***.

2.1. Création de l'autorité racine

Ouvrez une console et tapez la commande suivante : ***openssl*** *genrsa –des3 –out ca-racine.key*

- Quel est le but de cette commande ? donnez la signification des paramètres utilisés ?

Tapez la commande suivante : ***openssl*** *rsa –in ca-racine.key –nout –text*.

- Selon vous, quelles sont les informations affichées à l'écran ?

A présent il faudrait créer le certificat de la racine et l'autosigner, pour cela tapez la commande : ***openssl*** *req –new –x509 –days 365 –key ca-bancx.key –out ca-racine.crt*. Vous reprendrez les mêmes informations que la section 1.1.

- Donnez la signification des paramètres utilisés avec la précédente commande.
- Tapez la commande : ***openssl*** *x509 –in ca-bancx.crt –noout –subjet/text.* Selon vous, quelles sont les informations affichées à l'écran ?

Rangez le certificat racine (*ca-racine.crt*) ainsi que le jeu de clés associé (*ca-racine.key*) dans le répertoire */usr/ssl/certificats*, il faudrait créer le répertoire s'il n'existe pas.

- Créez la requête de signature, pour cela tapez la commande :

openssl *req –new –key ca-serveur.key –out ca-serveur.csr*.

- Ici vous utiliserez le CN=serveur.

2.2. Préparation d'un certificat Serveur

Comme pour la section 1.2, il faudrait commencer par préparer une requête de signature.

- *Créez un jeu de clé privé/publique pour le certificat serveur, vous nommerez le fichier résultat par ca-serveur.key.*
- *Créez la requête de signature, pour cela tapez la commande :*

openssl req –new –key ca-serveur.key –out ca-serveur.csr.

- *Ici vous utiliserez le CN=serveur.*

2.3. Signature du certificat serveur par l'autorité racine

Pour effectuer une signature, *openssl* a besoin d'une configuration spécifique. Cette configuration est typiquement fournie dans un fichier de paramètre et non dans la ligne de commande, d'où la nécessité d'utiliser le script *sign.sh*.

- Editez le ficher *sign.sh*, ce fichier est adapté du fichier fourni avec *openssl*. Il est à noter que :

- CA_DIR représente le dossier du CA (*/usr/ssl/certificats*)

- CA_CRT représente le certificat signataire (*ca-racine.crt*)
- CA_KEY la clé privée du CA signataire (*ca-racine.key*)

- Signez la requête CSR du CA fille en tapant ***sh sign.sh*** *ca-serveur.csr*

- Affichez le contenu du certificat serveur.crt ?

Rangez le certificat serveur et son jeu de clé dans le répertoire */usr/ssl/certificats*, vous en aurez besoin pour la suite de atelier.

3. Sécurisation par SSL

Pour cette partie en va mettre à jour le serveur Apache afin de prendre en charge SSL.

- Arrêtez le serveur Apache par la commande : ***/etc/init.d/httpd stop***

- Tapez ***yum install mod_ssl***

- Quels sont les données qui sont apparues dans le répertoire */etc/httpd*

- Tapez les commandes suivantes :

cd /home/admin/web

openssl req –new > new.cert.csr

openssl rsa –in privkey.pem –out new.cert.key

openssl x509 –in new.cert.csr –out new.cert.cert –req –signkey new.cert.key –days 999

cp new.cert.key /etc/httpd/conf/ssl.key/server.key cp new.cert.cert /etc/httpd/conf/ssl.crt/server.crt

Que signifient toutes ces commandes ? (***man openssl***, ***openssl –help***, etc) Ne pas oublier de relancer le serveur Web par: ***/etc/init.d/httpd start*** **Analyse du trafic**

Lancer *Ethere*al sur le poste <<serveur>>, se mettre en capture sur l'interface utilisé par votre réseau local (@IP) et demander depuis le navigateur du client, les pages suivantes : *https://@ip/~admin*.

Arrêter ensuite la capture et isoler les différentes trames dans ce trafic : pour le serveur web sans SSL, Pour le serveur web avec SSL.

Ecrivez les trames que vous pouvez isoler, notamment la phase de connexion (handshake) et l'établissement du mode SSL.

Pouvez-vous toujours isoler le login utilisateur ?