

Attaque MITM d'un service SSH et mise en place de contre-mesures

Description du thème

Propriétés	Description
Intitulé long	Ce TP a pour but de simuler une attaque de l'homme du milieu sur un service SSH afin de pointer différentes vulnérabilités et de proposer des contre-mesures.
Formation(s) concernée(s)	BTS Services Informatiques aux Organisations
Matière(s)	Bloc 3 SISR – Cybersécurité des services informatiques
Présentation	Après avoir remobilisé les savoirs fondamentaux en matière de cryptographie, ce TP permet de mettre en évidence certaines vulnérabilités du service SSH. À travers l'exploitation de ces vulnérabilités, l'étudiant sera amené à approfondir le fonctionnement de certains protocoles réseaux et de certaines attaques informatiques puis à mettre en place des contre-mesures visant à améliorer son hygiène numérique et ses pratiques professionnelles.
Savoirs	Chiffrement symétrique, asymétrique et fonction de hachage ; appliquer le principe de Kerckhoffs ; respecter l'état de l'art en matière de choix d'algorithmes cryptographiques ; authentification faible, authentification forte ; exploitation de vulnérabilité du protocole ARP ; analyse de trames.
Compétences	<ul style="list-style-type: none">• Participer à la vérification des éléments contribuant à la sûreté d'une infrastructure informatique.• Mettre en œuvre et vérifier la conformité d'une infrastructure à un référentiel, une norme ou un standard de sécurité.• Prévenir les attaques• Analyser les incidents de sécurité, proposer et mettre en œuvre des contre-mesures e sécurité.
Prérequis	Connaissances de base concernant l'administration d'un système GNU/Linux, fondamentaux en matière de cryptographie, fondamentaux réseaux (Ethernet, IP, TCP).
Mots-clés	cryptographie, chiffrement, exploitation de vulnérabilités, remédiations, hygiène numérique, respect des bonnes pratiques.
Durée	6 heures
Auteur.e(s)	Quentin Demoulière avec les précieuses relectures de Valérie Emin-Martinez, David Duron et Gilles Loiseau.
Version	v 1.1
Date de publication	02 septembre 2021

I. Préambule

Le TP proposé est uniquement à visée pédagogique. Son objectif est l'analyse de failles liées à l'usage de certains protocoles réseaux afin de proposer une amélioration de la sécurité informatique d'un système d'information et de l'hygiène numérique des étudiants. Il permet également l'acquisition de compétences associées au bloc 3 Cybersécurité SISR du BTS SIO.

 **Les outils abordés dans ce support sont uniquement utilisés à des fins éthiques (Ethical Hacking) et pédagogiques. Leur usage est formellement interdit en dehors de ce cadre sur un réseau tiers sans autorisation explicite.**

Pour rappel, l'article 323-1 du code pénal stipule que le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni de deux ans d'emprisonnement et de 60 000 € d'amende.

Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de trois ans d'emprisonnement et de 100 000 € d'amende.

II. Présentation

Le TP qui vous est proposé dispose d'une maquette sous VirtualBox contenant 4 machines virtuelles préconfigurées :

Intitulé de la machine	Nom de domaine pleinement qualifié	Configuration réseau	Applications et services
Serveur SSH sous Debian 10	srvssh.local.sio.fr	Adresse IPv4 : 192.168.56.10/24 Passerelle : 192.168.56.254 Serveur DNS : 127.0.0.1	Service OpenSSH port 22/TCP Service DNS Bind port 53/UDP
Client SSH sous Debian 10	clissh.local.sio.fr	Adresse IPv4 : 192.168.56.11/24 Passerelle : 192.168.56.254 Serveur DNS : 192.168.56.10	Environnement de bureau XFCE Client OpenSSH
Attaquant sous Kali Linux	NA	Adresse IPv4 : 192.168.56.12/24 Passerelle : 192.168.56.254 Serveur DNS : 192.168.56.10	Ettercap Git ssh-mitm Netfilter/Iptables
Routeur OpenBSD	rwbsd.local.sio.fr	Adresses IPv4 : em0 - DHCP em1 -192.168.56.254/24	PacketFilter

Voici les comptes et les mots de passe vous permettant d'accéder aux différentes machines virtuelles :

Intitulé de la machine	Nom d'utilisateur	Mot de passe
Serveur SSH sous Debian 10	etusio	Fghijkl1234*
Client SSH sous Debian 10	etusio	Fghijkl1234*
Attaquant sous Kali Linux 2020.1b	etusio	Fghijkl1234*
Routeur OpenBSD	etusio	Fghijkl1234*

Lorsque des commandes nécessitant des privilèges administrateurs seront utilisées, il sera nécessaire d'utiliser la commande sudo sous Debian GNU/Linux et doas sous OpenBSD.

```
etusio@srvssh:~$ sudo service ssh restart
rwbsd$ doas sh /etc/netstart
```

Voici une représentation logique de la maquette proposée dans le cadre de ce TP :

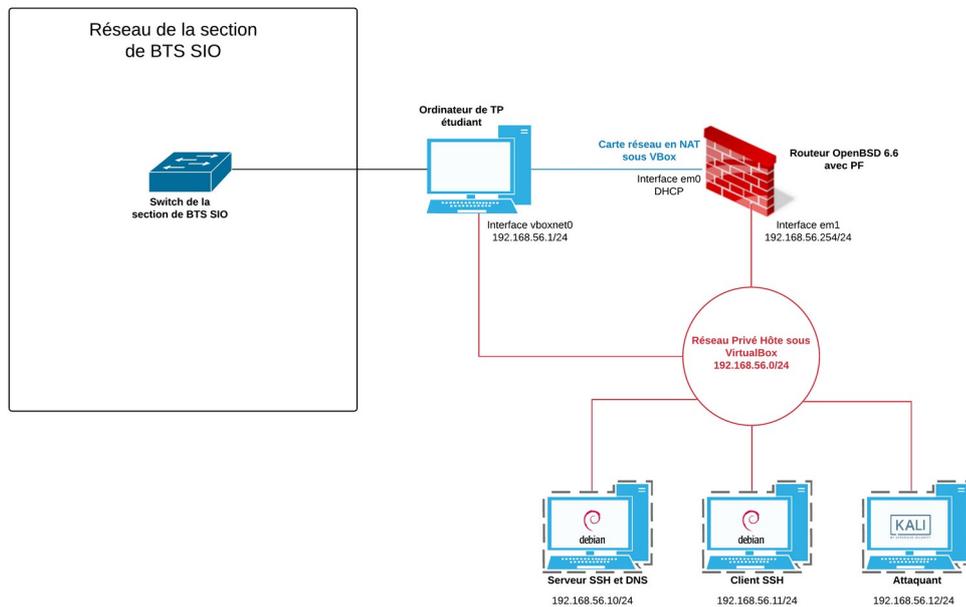
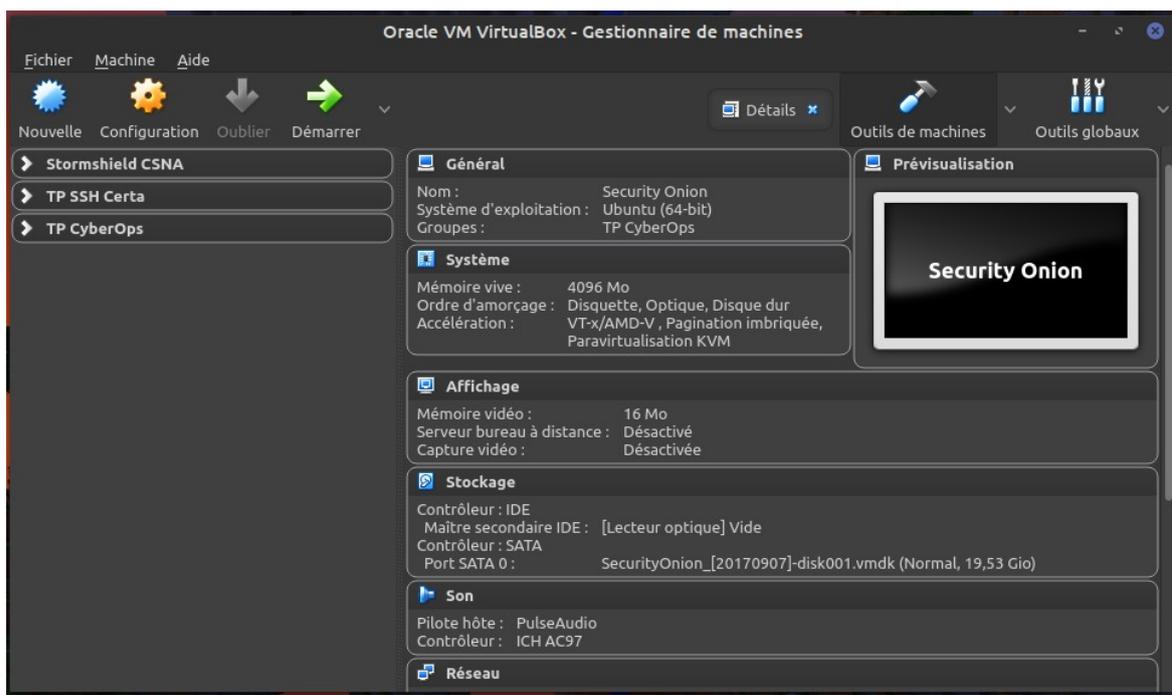
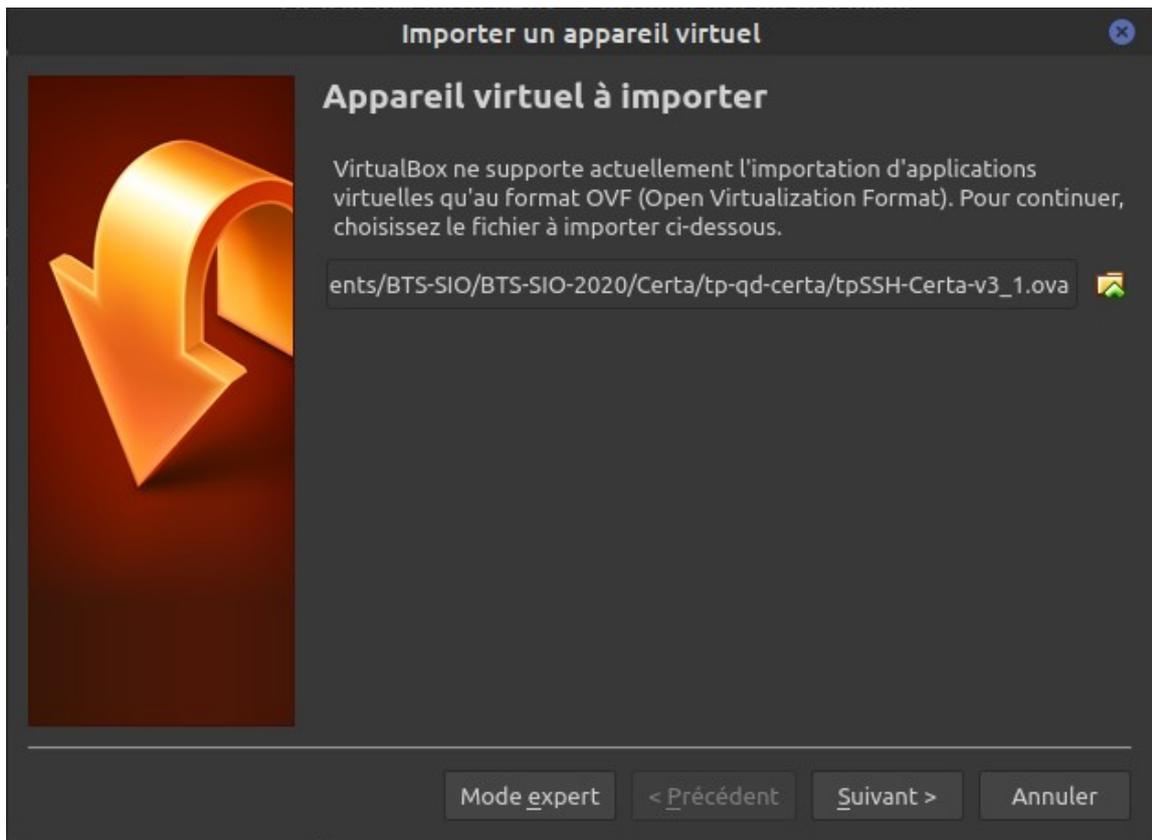


Illustration 1: Schéma logique de l'infrastructure de TP

III. Mise en œuvre de la maquette

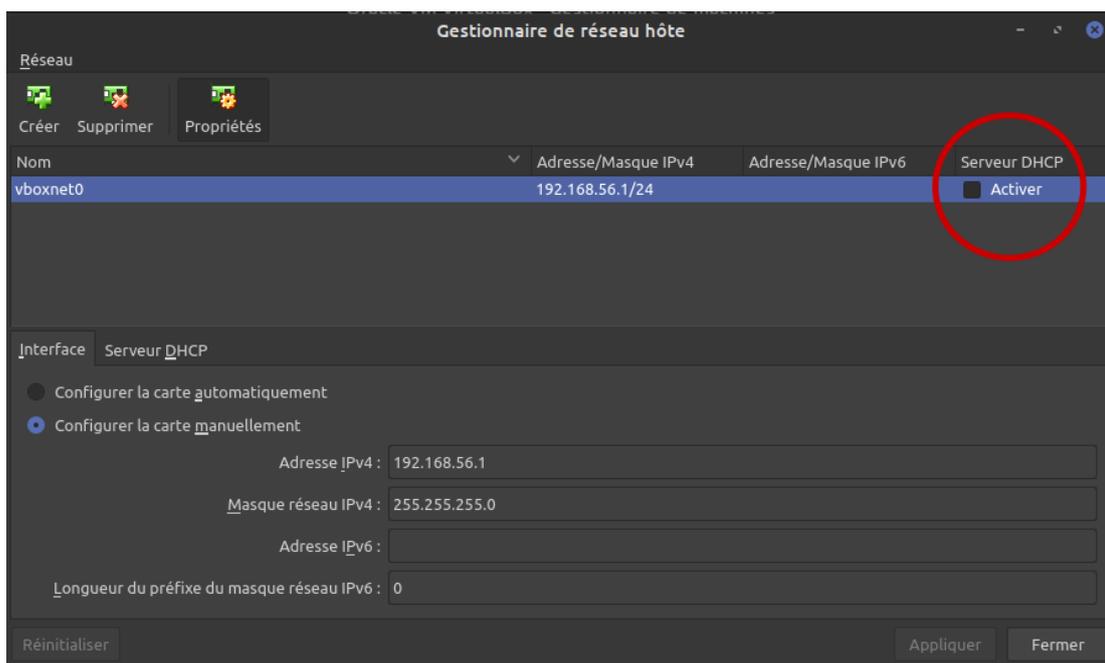
Récupérer le fichier tpSSH-Certa.ova et importez-le sur le logiciel VirtualBox (**Fichier>Importer un appareil virtuel**).

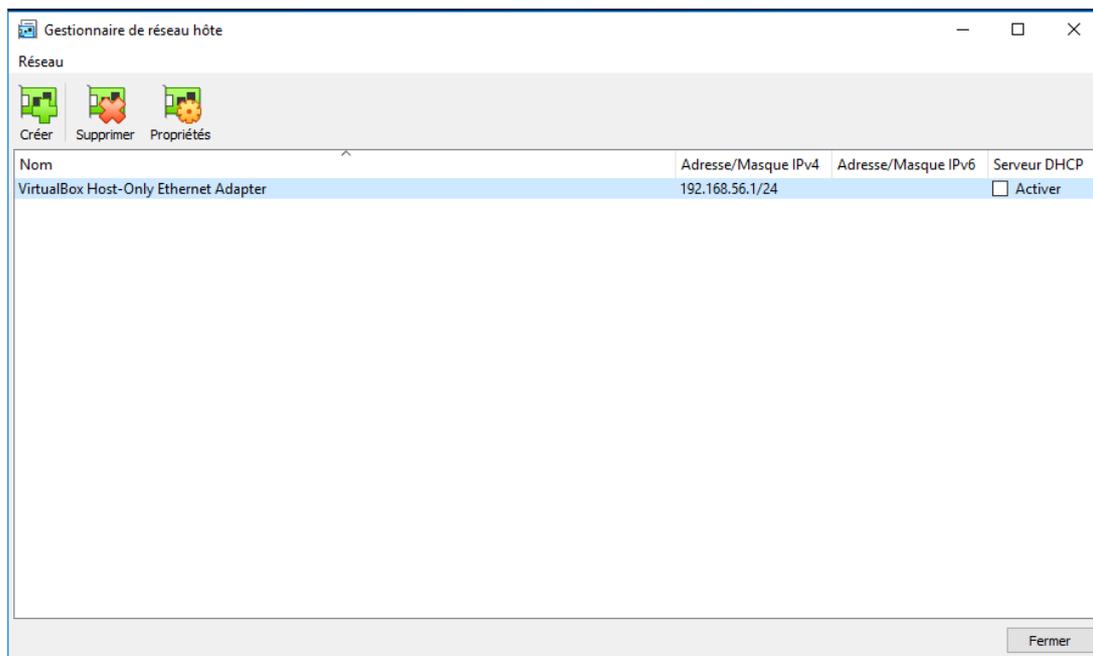




Une fois les différentes machines virtuelles importées, il faut s'assurer qu'une carte réseau (vboxnet0 ou VirtualBox Host-Only Ethernet Adapter) a bien été créée dans le gestionnaire de réseau hôte. Pour cela, cliquer sur **Fichier>Gestionnaire de réseau hôte**.

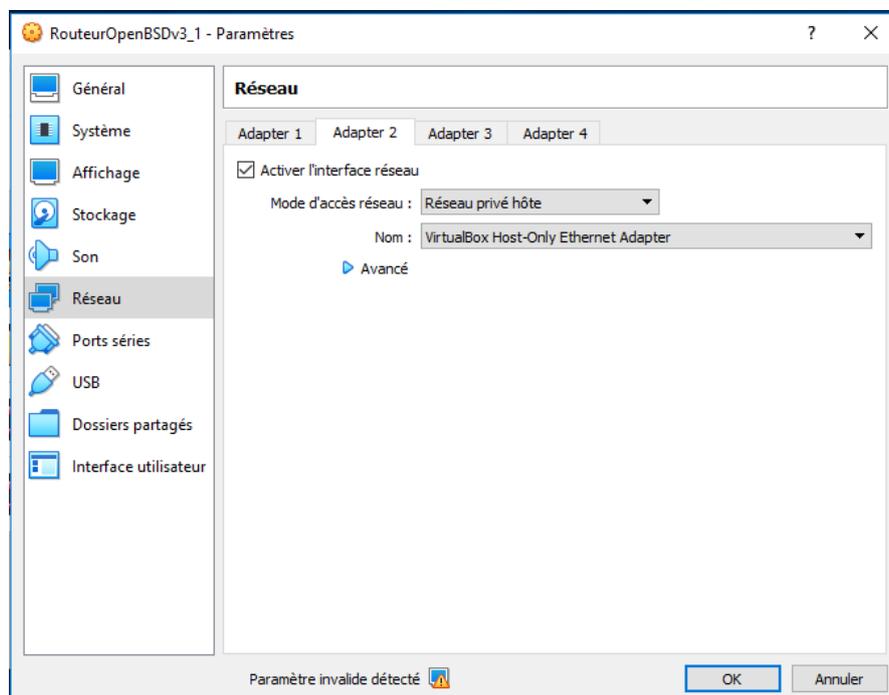
Vous devriez voir apparaître une nouvelle carte réseau virtuelle nommée vboxnet0 ou VirtualBox Host-Only Ethernet Adapter sous Windows avec comme adresse IP **192.168.56.1/24**. **Décocher** l'activation du serveur DHCP. Si aucune carte réseau n'apparaît, cliquer sur **Créer** puis indiquer les paramètres présents dans la capture d'écran ci-dessous.





Enfin, s'assurer dans les paramètres réseaux des machines virtuelles que les cartes réseaux définies en mode réseau privé hôte sont correctement liées à vboxnet0 ou VirtualBox Host-Only Ethernet Adapter.

⚠ Attention ! Il a été décidé de fournir une image ova la plus petite possible. Par conséquent, la machine virtuelle Kali Linux a été réduite à sa plus petite taille. **Il est donc déconseillé d'effectuer des mises à jour ou d'installer de nouvelles applications sur cette dernière sous peine de saturer le disque dur.**



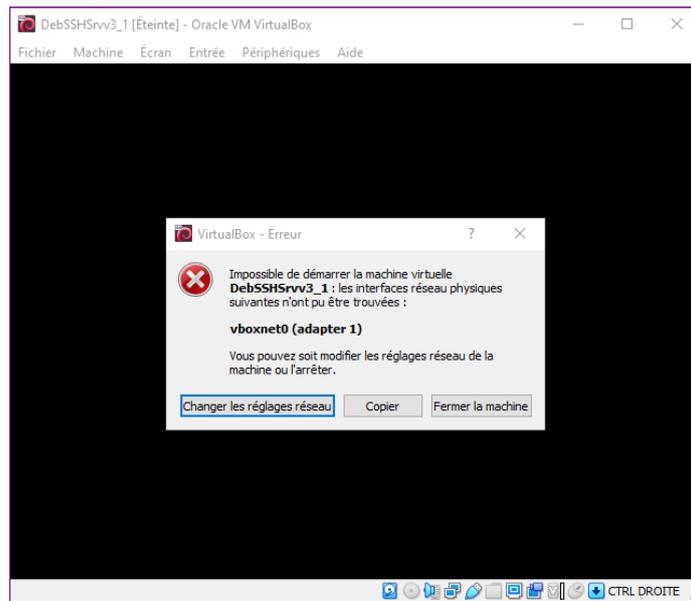
Ensuite, démarrer l'ensemble des machines virtuelles de la maquette y compris le routeur OpenBSD. C'est grâce à ce routeur que les autres VM auront accès à Internet (voir ci-après si un message d'erreur concernant la carte réseau apparaît au démarrage des machines).

Vous pouvez administrer l'ensemble de la maquette à l'aide de la console VirtualBox mais aussi à l'aide du protocole SSH depuis votre machine hôte à l'aide d'un client natif, de putty ou kitty.

⚠ Attention ! Il n'est pas nécessaire de modifier les configurations réseaux des machines virtuelles.

Le choix d'un réseau privé hôte plutôt que d'un réseau interne permet à l'étudiant de pouvoir se connecter en SSH sur chaque machine depuis l'ordinateur hôte. Ce dernier dispose en effet d'une carte réseau virtuelle nommée vboxnet0 sous GNU/Linux ou VirtualBox Host-Only sous Windows qui lui permet d'avoir une configuration réseau dans le même réseau que les machines virtuelles. Ainsi, cela offre une vraie souplesse en permettant notamment le copier/coller à partir de client SSH dédié depuis la machine hôte.

Sur une plateforme Windows, si vous n'avez pas accédé à la configuration réseau pour valider le nom de l'interface réseau avant de démarrer la machine, vous aurez probablement le message d'erreur suivant :



Il suffira de cliquer sur « Changer les réglages réseau » et de valider la prise en compte du « VirtualBox Host-Only Ethernet Adapter » à la place de « vboxnet0 » pour régler le problème.

Sur une plateforme Linux il suffit de :

- « fermer la machine » ;
- reconfigurer les paramètres réseaux des machines virtuelles en décochant et réactivant la case « Activer l'interface réseau » ;
- enregistrer en cliquant sur « OK ».

IV. Généralités

Utilisation des annexes 1 à 5

Q1. Pourquoi l'accès aux machines virtuelles par la console ou l'interface graphique n'est pas possible avec le super-administrateur root ?

.....

.....

.....

.....

Q2. Expliquer à quoi sert la commande sudo et quels avantages a-t-elle sur l'utilisation de la commande su - ?

.....

.....

.....
.....
Q3. Quelles commandes permettent de savoir si le service OpenSSH (serveur) est déjà installé et démarré ?

.....
.....
.....
Q4. Indiquer le répertoire où sont stockées les clés publique et privée créées ainsi que le positionnement des permissions appliquées sur les fichiers correspondants. Puis indiquer quel est le fichier de configuration du service SSH.

V. Première utilisation

Utilisation des annexes 1 à 5

📖 Depuis le client, se connecter au serveur SSH à l'aide de la commande ssh à taper dans un émulateur de terminal. Un message proche de celui présenté ci-dessous apparaît :

```
etusio@clissh:~$ ssh etusio@srvssh.local.sio.fr
The authenticity of host 'srvssh.local.sio.fr (192.168.56.10)' can't be established.
ECDSA key fingerprint is SHA256:IP1xEKxsYHPP3i7iMiZZXLYUoW9viLwSfF39MNoWIM4.
Are you sure you want to continue connecting (yes/no)?
```

Q5. Que signifie cette alerte qui est affichée à l'écran ? Devez-vous continuer l'opération ? Pourquoi ?

.....
.....
.....
Q6. Lors d'une prochaine connexion depuis le même client sur ce serveur, ce message apparaîtra-t-il à nouveau ? Pourquoi ?

L'analyse des résultats permet à l'attaquant de cibler plus particulièrement le serveur DNS.

VII. Simulation d'une attaque de l'homme du milieu entre votre client et votre serveur SSH

Utilisation de l'annexe 8

Une personne malveillante s'est introduite sur votre réseau dans le but de récupérer entre autres des informations confidentielles dont des noms d'utilisateurs et mots de passe disposant de privilèges sur le réseau.

Après avoir analysé l'architecture réseau et découvert l'existence d'un serveur SSH, elle décide de réaliser une attaque Man in the Middle afin d'obtenir un accès sur ce dernier.

Q9. Expliquer les principes généraux d'une attaque de l'homme du milieu (Man in the Middle).

.....

.....

.....

.....

.....

 Dans un premier temps, sur le client et le serveur SSH, analyser le cache ARP respectif des deux machines à l'aide de la commande (pour avoir des informations dans la cache arp, vous devrez peut-être au préalable lancer un ping sur chaque machine présente dans le réseau ou relancer les commandes nmap) :

```
etusio@clissh:~$ ip neigh show
etusio@srvssh:~$ ip neigh show
```

Q10. Noter les associations adresse IP / adresse MAC présentes sur les deux machines. Sont-elles cohérentes ?

.....

.....

.....

.....

.....

 Se connecter sur Kali Linux. Puis à l'aide de l'outil git, récupérer le logiciel ssh-mitm.

```
etusio@kali:~$ git clone https://github.com/ktux/ssh-mitm
```

 Se rendre ensuite dans le répertoire nouvellement récupéré, puis lancer le script d'installation.

```
etusio@kali:~$ cd ssh-mitm/
etusio@kali:~/ssh-mitm$ export LANG=en_US.utf-8
etusio@kali:~/ssh-mitm$ sudo ./install.sh
```

La commande « `export LANG=en_US.utf-8` » sert à exporter, via une variable d'environnement, la langue par défaut utilisée au moment de l'installation de l'outil (anglais avec un encodage UTF-8). Sans cet export, un message d'erreur empêche l'installation.

Lors de cette installation, il vous est demandé d'installer AppArmor pour restreindre les droits de ssh-mitm. Accepter la proposition.

```
Kali Linux detected with no AppArmor installed. For added safety, it is highly recommended (though not required) that sshd_mitm is run in a restricted environment. Would you like to automatically enable AppArmor? (y/n) y
```

☞ S'assurer d'être dans le répertoire ssh-mitm puis lancer le service ssh-mitm.

```
etusio@kali:~/ssh-mitm$ sudo ./start.sh
```

L'attaquant sera ainsi positionné entre le client et le serveur SSH. Il se fera passer pour le serveur légitime auprès du client, il recevra et journalisera toutes les informations transmises par le client avant de les transmettre au serveur légitime

La machine attaquante doit donc être en mesure de router les paquets le temps de l'attaque. **Ainsi le script start.sh exécute la commande suivante automatiquement** afin d'activer le routage.

```
sysctl -w net.ipv4.ip_forward=1
```

Q11. Pourquoi l'activation du routage sur la machine de l'attaquant est indispensable au bon fonctionnement de l'attaque MITM ?

.....

.....

.....

.....

.....

Puis **le script réalise** à l'aide de NETFILTER/IPTABLES une redirection de ports afin de rediriger tous les flux à destination de la machine attaquante sur le port 22/TCP vers le port 2222 du système Kali Linux.

```
iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-ports 2222
```

☞ Nous pouvons observer quel service écoute sur le port 2222 en localhost à l'aide de la commande

```
etusio@kali:~$ ss -ltnp
```

☞ Il est également possible d'observer quelles sont les règles de filtrage et de NAT en cours d'utilisation sur la distribution Kali Linux.

```
etusio@kali:~$ sudo iptables -L
```

Q12. Pourquoi cette redirection de ports est indispensable au succès de l'attaque de l'homme du milieu ?

.....

.....

.....

.....

.....

1. Mise en place d'une attaque ARP Spoofing afin d'obliger le client et le serveur SSH à faire transiter les trames Ethernet échangées par l'attaquant.

Q13. Indiquer en quoi une attaque de type ARP Spoofing peut être utile ici au pirate ?

.....
.....
.....
.....
.....
.....

☞ Réaliser cette attaque sur la machine Kali Linux à l'aide de la commande ettercap.

```
etusio@kali:~/ssh-mitm$ sudo ettercap -i eth0 -T -M arp /192.168.56.10// /192.168.56.11//
```

- -T : lance ettercap en mode texte ;
- -M : indique que l'on veut une attaque de type "Man in the middle" ;
- 192.168.56.10 (serveur ssh) et 192.168.56.11 (client SSH) sont les adresses IP des victimes.

☞ À nouveau sur le client puis le serveur SSH, analyser le cache ARP respectif des deux machines à l'aide de la commande :

```
etusio@clissh:~$ ip neigh show  
etusio@srvssh:~$ ip neigh show
```

Q14. Comparer les caches ARP du client et du serveur avec les associations notées précédemment lors de la question 10. Qu'en concluez-vous ?

.....
.....
.....
.....
.....

☞ Sur la machine cliente et sur Kali Linux, exécuter le logiciel de capture de trame Wireshark et réaliser une capture de trames d'une vingtaine de secondes et enregistrez-les. Analyser plus particulièrement les trames ARP émises et reçues.

Pour lancer Wireshark sur le client, cliquer sur **Applications>Internet>Wireshark** puis sélectionner l'interface Ethernet qui se nomme enp0s3 (contrairement à la machine Kali Linux qui utilise une carte Ethernet nommée eth0).

Q15. À partir de ces différentes observations, expliquer en détails comment fonctionne une attaque ARP Spoofing.

.....
.....
.....
.....

Q18. Que contient le fichier `/home/ssh-mitm/shell_session_0.txt` présent sur Kali Linux ?
NB : L'accès à ce fichier nécessite l'utilisation de la commande `sudo`.

.....

.....

.....

📖 Sur le poste de la victime, vider le cache ARP puis tenter de se connecter à nouveau sur le serveur SSH.

```
etusio@clish:~$ sudo ip neigh flush all
```

```
etusio@clish:~$ ssh etusio@srvssh.local.sio.fr
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)! It is also possible that a
host key has just been changed. The fingerprint for the ED25519 key sent by the remote host is
SHA256:iO+me7aX2v3eHBi+5cdiGOjHiAO/prPgk8Jgz1a9n24.
Please contact your system administrator.
Add correct host key in /home/etusio/.ssh/known_hosts to get rid of this message.
Offending ED25519 key in /home/etusio/.ssh/known_hosts:1 remove with:
ssh-keygen -f "/home/etusio/.ssh/known_hosts" -R "srvssh.local.sio.fr"
ED25519 host key for srvssh.local.sio.fr has changed and you have requested strict checking. Host key
verification failed.
```

Q19. Expliquer pourquoi ce message d'erreur apparaît ?

.....

.....

.....

.....

.....

.....

.....

.....

Q20. Proposer une solution afin de pouvoir à nouveau se connecter au service SSH depuis le client.

.....

.....

.....

.....

.....

.....

.....

.....

VIII. Renforcement de la sécurité du service OpenSSH et remédiation

1. Mise en place d'une authentification par clés de chiffrement

Utilisation des annexes 6 et 7

Lorsque vous générez une paire de clés de chiffrement, vous devez vous assurer que celle-ci n'est pas prédictible par un attaquant. L'entropie permet de mesurer l'imprédictibilité lors de la génération d'une paire de clés, et donc la difficulté qu'un attaquant rencontrera à découvrir cette dernière.

⚠ Attention ! Dans un contexte professionnel, la génération de clés de chiffrement doit être réalisée sur une machine physique. De plus, cette machine doit disposer de plusieurs sources d'entropie indépendantes. La génération de clés sur des machines virtuelles conduit à une baisse trop importante de l'entropie et donc à une mauvaise qualité des clés créées.

L'authentification par mot de passe est considérée comme un mécanisme plutôt faible. Elle peut être soumise à différents types d'attaque comme les attaques par dictionnaire, par force brute ou par Man In The Middle comme nous l'avons vu précédemment.

Ainsi lorsque l'on configure un service SSH, il est recommandé de mettre en œuvre une authentification plus robuste par double facteur d'authentification (2FA) ou par clés de chiffrement.

📄 Editer le fichier de configuration serveur et autoriser ce mécanisme de connexion :

```
etusio@srvssh:~$ sudoedit /etc/ssh/sshd_config
```

```
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```

📄 Redémarrer le service.

```
etusio@srvssh:~$ sudo service ssh restart
```

Les manipulations sur le serveur sont maintenant terminées. Quitter éventuellement la connexion SSH sur le serveur puis basculer sur le client.

📄 Sur la machine cliente, l'utilisateur **etusio** va devoir générer sa clé publique, et sa clé privée afin de pouvoir s'authentifier sur le serveur OpenSSH :

```
etusio@clissh:~$ ssh-keygen -b 256 -t ecdsa
```

Le générateur de clés va en générer deux, une clé publique et une clé privée. Il va placer la clé privée dans un endroit qui, par défaut, est `$HOME/.ssh/id_ecdsa` :

```
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/etusio/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/etusio/.ssh/id_ecdsa.
Your public key has been saved in /home/etusio/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:EqNMWPDfxKHfPCkejRqkytTlqGHTRe+TJWv90DOV8tM etusio@clissh
```

➤ Appuyer sur **Entrée** pour accepter la localisation de la clé par défaut. `ssh-keygen` demande ensuite une phrase de chiffrement (ou « passphrase », équivalent d'un mot de passe, mais sous forme de phrase).

➤ Tapez **TPMITMcerta** comme phrase de chiffrement et confirmez-là. Vous obtenez l'empreinte de votre clé (« key fingerprint ») et une « randomart image ».

Q21. Pourquoi l'algorithme ECDSA a été préféré à l'algorithme RSA lors de la génération de la paire de clés ?

.....

.....

.....

.....

Q22. A votre avis, pourquoi la mise en place de cette phrase de chiffrement pour accéder à la clé privée est extrêmement importante ?

.....

.....

.....

.....

Que faire de la paire de clés ? Si l'on récapitule, ssh-keygen a généré deux clés. Une **clé privée** qui est `$HOME/.ssh/id_ecdsa` à laquelle vous seul devez avoir accès et une **clé publique** qui est `$HOME/.ssh/id_ecdsa.pub`, qui peut être connue par tout le monde.

Q23. Lister le contenu du répertoire `$HOME/.ssh/` puis afficher le contenu du fichier `id_ecdsa.pub`.

.....

.....

.....

 **Définition** : qu'est-ce que ce « \$HOME » ? \$HOME est ce qu'on appelle une variable d'environnement, qui sert à indiquer aux programmes quel est votre répertoire personnel (la racine de votre compte). Faites echo \$HOME pour savoir quel est le vôtre.

Sur le serveur, vous devez maintenant autoriser explicitement votre compte présent sur la machine cliente à accéder via ssh à celui-ci. Pour ce faire, vous devez ajouter dans le répertoire `.ssh` de l'utilisateur qui sera choisi pour se connecter (exemple : `/home/etusio/.ssh`) la clé publique générée précédemment (`id_ecdsa.pub`) dans un fichier `authorized_keys`.

La méthode la plus simple est d'utiliser la commande `ssh-copy-id` qui copiera la clé dans un fichier `authorized_keys` à l'emplacement défini dans votre chemin (`~/.ssh/`), même si celui-ci n'existe pas au préalable.

```
ssh-copy-id -i <chemin/nomfichier><utilisateur>@<adresseIP ou nom> -p <numport>
```

 Sur le poste client :

```
etusio@clissh:~$ ssh-copy-id -i ~/.ssh/id_ecdsa.pub etusio@srvssh.local.sio.fr
```

 Depuis le poste client, relancez une connexion ssh au serveur, la phrase de chiffrement vous est bien demandée, cela vous permet de vérifier que la connexion avec clé publique est fonctionnelle.

 Sur le serveur Debian, désactiver l'authentification par mot de passe pour juste conserver celle par clés dans le fichier de configuration `/etc/ssh/sshd_config` :

 **Attention !** Si vous désactivez l'authentification par mot de passe, vous devez vous assurer que l'authentification par clé est opérationnelle sous peine de ne plus du tout pouvoir accéder à la machine distante !

Dans le fichier de configuration, décommenter la directive *PasswordAuthentication* puis affecter le paramètre no.

```
# Autorisation de connexion par mot de passe
PasswordAuthentication no
```

 Redémarrer le service SSH pour que la modification du fichier soit prise en compte :

```
etusio@srvssh:~$ sudo service ssh restart
```

Nous allons maintenant vérifier que ssh se préoccupe de la sécurité de vos clés :

Q24. Sur la machine cliente `clissh.local.sio.fr`, modifier les droits du fichier `~/.ssh/id_ecdsa` (droits initiaux : 600 `etusio:etusio`) qui contient votre clé privée en 644. Se connecter sur le serveur distant `srvssh.local.sio.fr` qui contient la clé publique. Que se passe-t-il ? Pourquoi ?

.....

.....

.....

.....

.....

Q25. Rétablir les droits de `~/.ssh/id_ecdsa` sur le poste client. Maintenant sur le serveur distant, afficher les droits d'accès appliqués au fichier `~/.ssh/authorized_keys`.

.....

.....

.....

Q26. Puis, modifier les droits de `~/.ssh/authorized_keys` en 666. Se déconnecter puis se reconnecter. Vérifier si un changement est apparu ou non et tenter d'expliquer pourquoi (ne pas oublier de rétablir les droits d'origine ensuite).

.....

.....

.....

.....

Q27. En analysant la connexion par clés, proposer une hypothèse de fonctionnement de cette nouvelle forme d'authentification. Expliquer ce qui différencie une connexion par mot de passe d'une connexion par clé de chiffrement.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2. Utilisation de ssh-agent

L'agent ssh est particulièrement utile si vous vous connectez très régulièrement à une machine et que vous ne souhaitez pas retaper la phrase de passe liée à votre clé privée à chaque connexion. Mais attention, cela peut ouvrir certaines brèches en matière de sécurité.

- ☞ Sur le client, taper les commandes suivantes, pour mettre en place le ssh-agent. Par défaut sous Debian, le bureau GNOME intègre un trousseau de clés qui fonctionne en adéquation avec ssh-agent et qui permet d'éviter de saisir à chaque fois des mots de passe.

```
etusio@clish:~$ exec ssh-agent $SHELL
```

NB : L'utilisation de la variable \$SHELL permet d'exécuter l'agent SSh dans l'interpréteur en cours d'utilisation.

```
etusio@clish:~$ ssh-add
Enter passphrase for ~/.ssh/id_ecdsa:
Identity added: ~/.ssh/id_ecdsa (~/.ssh/id_ecdsa)
```

Entrer la phrase de passe configurée au préalable. Pendant toute la durée de la connexion, il est possible d'avoir accès à la machine distante sans avoir à taper un mot de passe. Si le processus ssh-agent disparaît, il sera nécessaire de retaper le mot de passe.

Q28. Suite à la mise en place de cette authentification par clés de chiffrement, tenter à nouveau de simuler une attaque MITM entre le client et le serveur SSH. Quel résultat obtenez-vous ? Pourquoi ?

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

- ☞ Effacer à nouveau le contenu du fichier /home/etusio/.ssh/known_hosts avant de poursuivre la suite du TP et penser également à arrêter l'attaque si cela n'a pas encore été fait.


```
etusio@srvssh:~$ sudo ssh-keygen -r srvssh
srvssh IN SSHFP 1 1 6fa8aeb8227f09bc4f8c8afe08245a8c7718d324
srvssh IN SSHFP 1 2 ef3c9989952d8591cc761f1b26a658354a93230dae730edbe42732bf0a1219a8
srvssh IN SSHFP 3 1 4c16f0e9298469630445a24c57c342b15540253f
srvssh IN SSHFP 3 2 94f23110ac6c6073cfde2ee23226595cb614a16f6f88bc127c5dfd30da1694ce
srvssh IN SSHFP 4 1 7ef6631af09e0585ff8c3169255189195f7b7ddf
srvssh IN SSHFP 4 2 88efa67bb697dafdde1c18bee5c76218e8c78803bfa6b3e093c260cf56bd9f6e
```

📖 Ajouter ces nouveaux enregistrements à la fin du fichier de zone présent sur le serveur **SSH/DNS**.

```
etusio@srvssh:~$ sudoedit /var/named/db.local.sio.fr
```

📖 Recharger le service.

```
etusio@srvssh:~$ sudo service bind9 reload
```

📖 Sur la machine cliente, si vous souhaitez que cette vérification se fasse au moment de la connexion, voici la commande à utiliser :

```
etusio@clissh :~$ ssh -o VerifyHostKeyDNS=true etusio@srvssh.local.sio.fr
```

Si vous souhaitez que cette vérification ait lieu à chaque nouvelle connexion depuis votre poste client, vous pouvez rajouter cette option dans le fichier de configuration du client (/etc/ssh/ssh_config) :

```
VerifyHostKeyDNS yes
```

4. Amélioration de la sécurité du service OpenSSH sur le serveur

Q30. Mettre en œuvre les préconisations suivantes tirées des recommandations pour un usage sécurisé de SSH publié par l'ANSSI :

1. Vérifier que les clés privées de chiffrement présentes dans le répertoire /etc/ssh/ appartiennent à l'utilisateur root en lecture-écriture seulement ;
2. S'assurer que c'est bien la version 2 du protocole SSH qui est utilisée ;
3. Le serveur SSH doit dorénavant écouter sur le port 222/TCP ;
4. Vérifier que les droits sur les fichiers sont appliqués de manière stricte par SSH ;
5. L'accès SSH par l'utilisateur root doit être interdite ;
6. Mettre en œuvre une séparation des privilèges à l'aide d'un bac à sable (sandbox) ;
7. L'accès à distance par des comptes ne disposant pas de mot de passe doit être interdit ;
8. Autoriser 3 tentatives de connexion successives en cas d'erreur dans le mot de passe ;
9. Le service doit afficher les informations de dernière connexion à l'utilisateur quand il se connecte ;
10. N'autoriser que l'utilisateur etusio à se connecter sur le serveur.

Le lien suivant liste les différentes directives qui existent dans le fichier sshd_config :

https://man.openbsd.org/OpenBSD-6.0/sshd_config.5

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....
.....
.....
.....

5. Durcissement des algorithmes de chiffrement utilisés entre le client et le serveur SSH

Utilisation de l'annexe 10

Lorsque l'on implémente des protocoles chiffrés comme https ou ssh, il est primordial de respecter l'état de l'art en matière de choix des algorithmes utilisés.

Q31. Expliquer dans une définition succincte ce qu'est l'état de l'art dans le domaine de la cybersécurité.

.....
.....
.....
.....

Q32. Définir ce qu'est le principe de Kerckhoffs.

.....
.....
.....
.....

Q33. Selon ce principe, pourquoi est-il pertinent de choisir des algorithmes cryptographiques connus et respectant l'état de l'art ?

.....
.....
.....
.....

Des agences de sécurité nationale comme l'ANSSI en France ou les experts en sécurité informatique de grandes entreprises comme Mozilla publient régulièrement les algorithmes à privilégier lorsque l'on utilise OpenSSH ou TLS.

 Pour empêcher l'usage d'algorithmes de chiffrement dépréciés, il est nécessaire d'éditer le fichier de configuration du serveur SSH et ajouter les lignes suivantes :

```
etusio@srvssh:~$ sudoedit /etc/ssh/sshd_config
```

```
KexAlgorithms curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group-exchange-sha256
```

```
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
```

```
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com
```

 Puis redémarrer le service ssh.

```
etusio@srvssh:~$ sudo service ssh restart
```

Annexes

1. Quelques rappels historiques

L'apparition des premiers Unix et systèmes d'information communicants s'est accompagnée de l'émergence de piles protocolaires visant l'échange de données entre machines comme FTP, TELNET ou encore RSH.

Bien qu'encore largement utilisés aujourd'hui, ces protocoles n'ont pas été conçus pour être sécurisés ; leurs fonctionnalités sont particulièrement pauvres lorsqu'il s'agit d'authentifier la source ou l'émetteur, ou encore de garantir l'intégrité et la confidentialité des flux.

Leur usage est même devenu problématique d'un point de vue filtrage. FTP nécessite par exemple une ouverture dynamique de port sur une passerelle utilisant du NAT. Pour ces raisons, le besoin d'un protocole applicatif sécurisé capable de remplacer ces briques logicielles s'est fait rapidement sentir : SSH est né.

2. Qu'est-ce que OpenSSH ?

OpenSSH (OpenBSD Secure Shell) est un ensemble d'outils informatiques libres permettant des communications sécurisées sur un réseau informatique en utilisant le protocole SSH. Créé comme alternative Open Source à la suite logicielle proposée par la société SSH Communications Security, OpenSSH est développé depuis 1999 par l'équipe d'OpenBSD, dirigée par son fondateur, Theo de Raadt, et diffusé sous licence BSD.

OpenSSH est à la fois une brique logicielle du système OpenBSD et l'implémentation SSH la plus utilisée sur les systèmes BSD et GNU/Linux. OpenSSH utilise la cryptographie asymétrique comme mécanisme d'authentification. Contrairement à TLS, le modèle de sécurité de ce service n'utilise pas par défaut une infrastructure à clés publiques mais la méthode Trust on the first use.

OpenSSH couvre les mécanismes suivants :

- ❖ le chiffrement ;
- ❖ l'authentification ;
- ❖ l'intégrité des données transmises.

3. L'approche Trust on the first use

TOFU signifie "accorder sa confiance lors du premier usage". Cette approche est utilisée lorsqu'on ajoute de manière permanente l'empreinte de la clé publique du serveur sur lequel on se connecte pour la première fois dans un fichier présent sur le client.

Le principe général est de considérer, par un acte de foi (TOFU est également appelé, en anglais, "leap of faith"), que la première fois que l'on reçoit une empreinte de clé publique, celle-ci n'a pas été émise par un attaquant. Une fois cette empreinte de clé acceptée une première fois, il est admissible que toute communication future impliquant cette clé publique soit avec le même correspondant. Le client n'émettra dès lors un avertissement qu'à la réception d'une nouvelle clé pour un serveur sur lequel il ne s'est jamais encore connecté.

Si cette approche est plébiscitée par certains, elle est difficilement applicable par l'utilisateur lambda qui ignore généralement les warnings et validera n'importe quelle empreinte de clé publique sans se soucier qu'il s'agisse d'une clé légitime ou non. Cette approche a cependant un avantage certain : sa simplicité de mise en œuvre.

4. Configuration du client et du serveur SSH

Les informations de configuration SSH qui s'appliquent à l'ensemble du système sont stockées dans le répertoire `/etc/ssh` où figurent :

- ❖ `moduli` — Fichier contenant les groupes Diffie-Hellman utilisés pour l'échange de clés Diffie-Hellman qui est crucial pour la création d'une couche de transport sécurisée. Lorsque les clés sont échangées au début d'une session SSH, une valeur secrète partagée ne pouvant être déterminée que conjointement par les deux parties est créée. Cette valeur est ensuite utilisée pour effectuer l'authentification de l'hôte.
- ❖ `ssh_config` — Fichier de configuration client SSH pour l'ensemble du système. Il est écrasé si un même fichier est présent dans le répertoire personnel de l'utilisateur (`~/.ssh/config`).
- ❖ `sshd_config` — Fichier de configuration pour le démon `sshd`.
- ❖ `ssh_host_dsa_key` — Clé DSA privée utilisée par le démon `sshd`.
- ❖ `ssh_host_dsa_key.pub` — Clé DSA publique utilisée par le démon `sshd`.
- ❖ `ssh_host_rsa_key` — Clé RSA privée utilisée par le démon `sshd` pour la version 2 du protocole SSH.
- ❖ `ssh_host_rsa_key.pub` — Clé RSA publique utilisée par le démon `sshd` pour la version 2 du protocole SSH.
- ❖ `ssh_host_ecdsa_key` — Clé ECDSA privée utilisée par le démon `sshd` pour la version 2 du protocole SSH.
- ❖ `ssh_host_ecdsa_key.pub` — Clé ECDSA publique utilisée par le démon `sshd` pour la version 2 du protocole SSH.

Les informations de configuration SSH spécifiques à l'utilisateur sont stockées dans son répertoire personnel à l'intérieur du répertoire `~/.ssh/` où figurent :

- ❖ `authorized_keys` — Fichier contenant une liste de clés publiques autorisées pour les serveurs. Lorsque le client se connecte à un serveur, ce dernier authentifie le client en vérifiant sa clé publique signée qui est stockée dans ce fichier.
- ❖ `id_dsa` — Fichier contenant la clé DSA privée de l'utilisateur.
- ❖ `id_dsa.pub` — Clé DSA publique de l'utilisateur.
- ❖ `id_rsa` — Clé RSA privée utilisée par `ssh` pour la version 2 du protocole SSH.
- ❖ `id_rsa.pub` — Clé RSA publique utilisée par `ssh` pour la version 2 du protocole SSH.
- ❖ `id_ecdsa` — Clé ECDSA privée utilisée par `ssh` pour la version 2 du protocole SSH.
- ❖ `id_ecdsa.pub` — Clé ECDSA publique utilisée par `ssh` pour la version 2 du protocole SSH.
- ❖ `known_hosts` — Fichier contenant les clés d'hôtes des serveurs SSH auxquelles l'utilisateur a accédé. Ce fichier est très important car il permet de garantir que le client SSH se connecte au bon serveur SSH.

5. Comment fonctionne la mise en œuvre du chiffrement d'une connexion SSH ?

La méthode de chiffrement d'une connexion SSH diffère de celle utilisée avec le protocole TLS. Ainsi le chiffrement entre le client et le serveur sera réalisée à l'aide d'une clé de chiffrement symétrique de session commune au serveur et au client. Cette clé sera créée à l'aide d'un algorithme d'échange de clés type Diffie-Hellman.

Client SSH		Serveur SSH
1. Demande de connexion (Port 22/TCP).		2. Le serveur envoie les versions des algorithmes acceptés.
3. Choix parmi les algorithmes de celui qui convient le mieux.		
4. Échange de la clé de session (Diffie-Hellman) et authentification du serveur par l'envoi de l'empreinte de sa clé publique.		4. Échange de la clé de session (Diffie-Hellman) et authentification du serveur par l'envoi de l'empreinte de sa clé publique.
		

Voici une illustration conceptuelle permettant de mieux appréhender le fonctionnement d'un échange de clés type Diffie-Hellman.

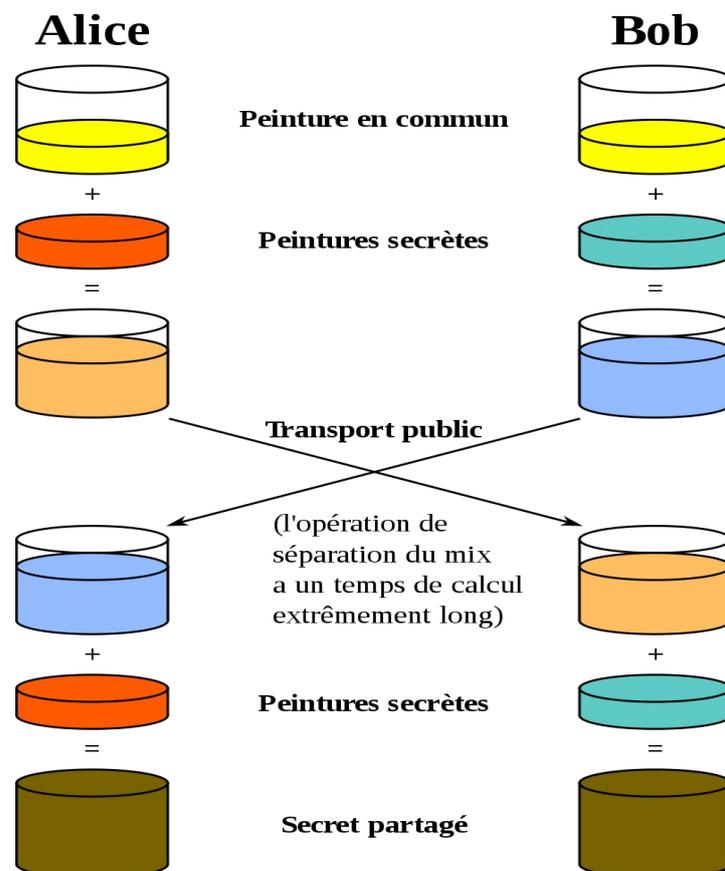


Illustration 2: Schéma conceptuel représentant l'échange de clés Diffie-Hellman Wikipédia

6. Fonctionnement de l'authentification par clés cryptographiques avec OpenSSH

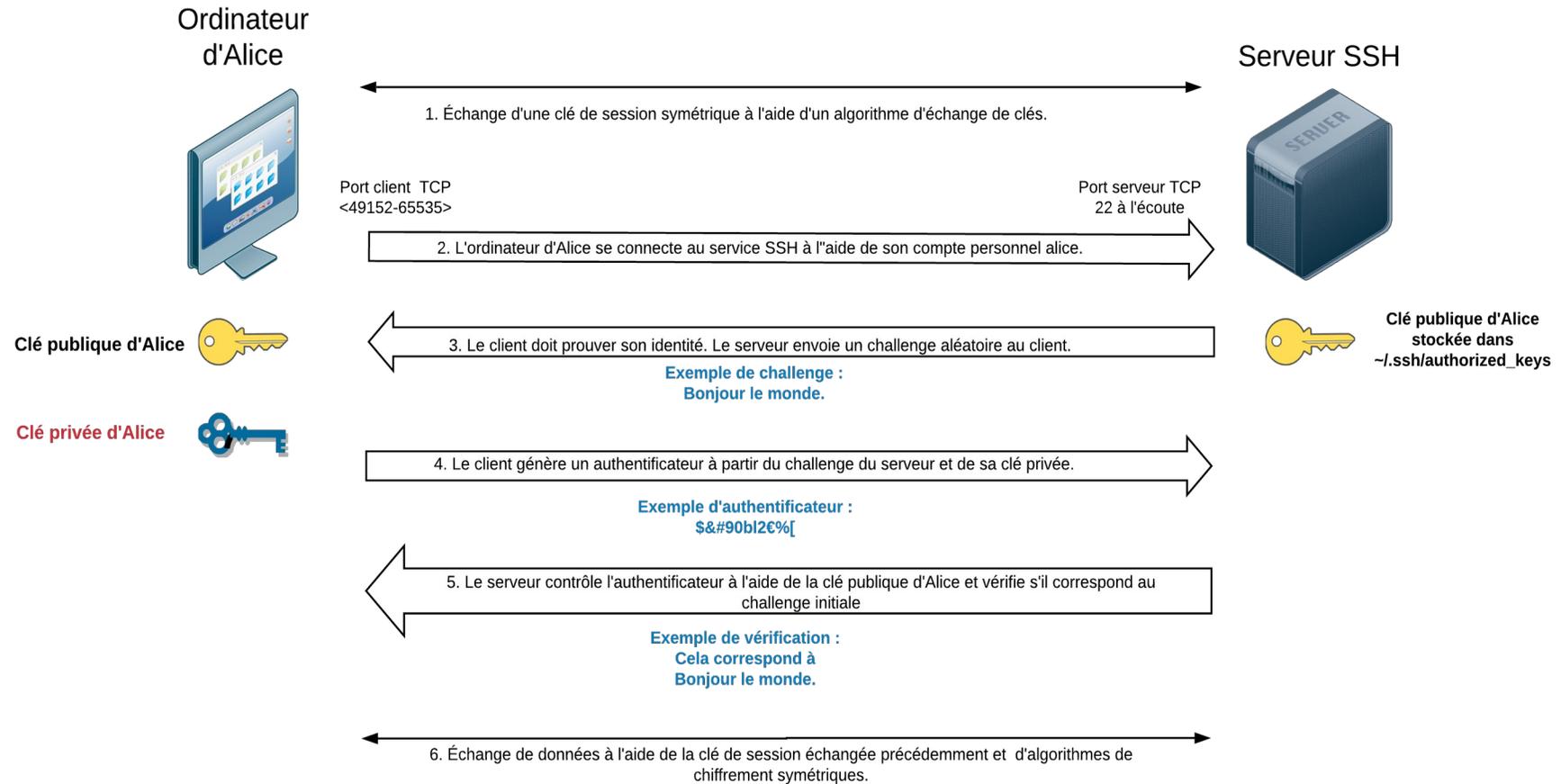


Illustration 3: Schéma représentant le fonctionnement de l'authentification par clés avec SSH

7. La notion d'entropie

L'entropie de Shannon, due à Claude Shannon, est une fonction mathématique qui, intuitivement, correspond à la quantité d'information contenue ou délivrée par une source d'information. Cette source peut être un texte écrit dans une langue donnée, un signal électrique ou encore un fichier informatique quelconque (collection d'octets).

L'entropie est une manière de mesurer la qualité d'une méthode de chiffrement. Elle mesure la densité d'information, le bruit ajouté, la non redondance ou encore l'anti signal-bruit. Une entropie basse signifie une faible densité d'information. Une entropie haute en revanche est le signe d'une haute densité d'information. Une forte entropie provenant de différentes sources rend difficile la prédictibilité des clés de chiffrement générées.

Sous GNU/Linux, il est possible de s'assurer du niveau d'entropie disponible sur le système :

```
$ cat /proc/sys/kernel/random/entropy_avail
```

Plus la valeur s'approche de 0 et plus l'entropie disponible diminue. On peut considérer qu'en dessous de la valeur 1000, la génération de nombres aléatoires s'avère compliquée.

Si l'on souhaite augmenter le niveau d'entropie en amont de la génération d'une paire de clés de chiffrement :

```
$ sudo apt-get install -y rng-tools  
$ sudo /usr/sbin/rngd -r /dev/urandom
```

Il y a deux dispositifs de génération de nombres aléatoires sous Linux : `/dev/random` et `/dev/urandom`.

Les nombres les plus aléatoires proviennent de `/dev/random` car ce périphérique se bloque chaque fois que sa réserve d'entropie devient insuffisante. Il attendra qu'une entropie suffisante soit de nouveau disponible pour continuer à fournir une sortie.

En supposant que votre entropie soit suffisante, vous devriez avoir la même qualité de caractère aléatoire dans `/dev/urandom` ; cependant, comme ce périphérique est non bloquant, il continuera à produire des données « aléatoires », même lorsque le réservoir d'entropie sera faible ou épuisé.

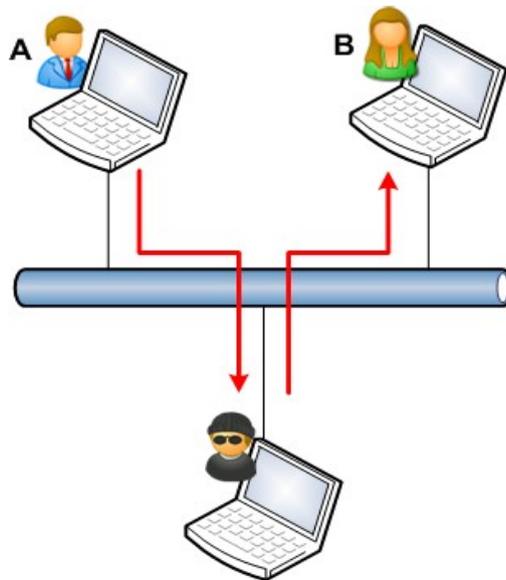
8. Les différents types d'attaque abordés dans ce TP

8.1 Attaque de l'homme du milieu (MITM)

L'attaque de l'homme du milieu (HDM) ou man-in-the-middle attack (MITM), parfois appelée attaque de l'intercepteur, est une attaque qui a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles a été compromis.

L'attaque « homme du milieu » est particulièrement applicable dans la méthode d'échange de clés Diffie-Hellman.

Dans l'attaque de l'homme du milieu, l'attaquant a non seulement la possibilité de lire, mais aussi de modifier les messages.



Dessin 1: Exemple d'une attaque MITM active tiré des supports pédagogiques CyberÉdu

Le but de l'attaquant est de se faire passer pour l'un des correspondants (voire les 2), en utilisant, par exemple :

- l'imposture ARP (ARP Spoofing) : c'est probablement le cas le plus fréquent. Si l'un des interlocuteurs et l'attaquant se trouvent sur le même réseau local, il est possible, voire relativement aisé, pour l'attaquant de forcer les communications à transiter par son ordinateur en se faisant passer pour un « relais » (routeur, passerelle) indispensable. Il est alors assez simple de modifier ces communications ;
- l'empoisonnement DNS (DNS Poisoning) : L'attaquant altère le ou les serveur(s) DNS des parties de façon à rediriger vers lui leurs communications sans qu'elles s'en aperçoivent ;
- l'analyse de trafic afin de visualiser d'éventuelles transmissions non chiffrées ;
- le déni de service : l'attaquant peut par exemple bloquer toutes les communications avant d'attaquer une cible. L'ordinateur ne peut donc plus répondre et l'attaquant a la possibilité de prendre sa place.

8.2 Attaque ARP Spoofing ou ARP Poisoning

L'ARP spoofing (« usurpation » ou « parodie ») ou ARP poisoning (« empoisonnement ») est une technique utilisée en informatique pour attaquer tout réseau local utilisant le protocole de résolution d'adresse ARP, les cas les plus répandus étant les réseaux Ethernet et Wi-Fi. Cette technique permet à l'attaquant de détourner des flux de communications transitant entre une machine cible et un hôte sur le réseau : ordinateur, routeur, box, etc. L'attaquant peut ensuite écouter, modifier ou encore bloquer les paquets réseaux.

9. Le protocole SSHFP

Le protocole SSHFP (rfc 4255) permet de publier les empreintes de clés publiques des hôtes disposant d'un service SSH dans votre zone DNS. Ainsi, lorsque le client établira une nouvelle connexion auprès d'un serveur SSH, il comparera l'empreinte de la clé publique proposée par le serveur avec celles enregistrées dans la zone DNS. Si elles sont identiques, alors l'identité du serveur est prouvée.

L'enregistrement SSHFP se compose de plusieurs parties :

```
srvssh IN SSHFP 1 2 ef3c9989952d8591cc761f1b26a658354a93230dae730edbe42732bf0a1219a8
```

La valeur après SSHFP (1 dans l'exemple) définit le type de clés. La valeur 1 correspond à une clé publique RSA, 2 correspond à une clé publique DSA, 3 correspond à une clé ECDSA et 4 à une clé Ed25519.

La valeur suivante (2 dans l'exemple) correspond au type d'empreinte publiée. La valeur 1 correspond à une empreinte SHA1 (appelée aussi condensat), la valeur 2 correspond à une empreinte SHA256.

10. Le principe de Kerckhoffs

Le principe de Kerckhoffs a été énoncé par Auguste Kerckhoffs à la fin du XIX^{ème} siècle dans un article en deux parties « La cryptographie militaire » du Journal des sciences militaires. Ce principe exprime que la sécurité d'un cryptosystème ne doit reposer que sur le secret de la clé.

Autrement dit, tous les autres paramètres doivent être supposés publiquement connus. Il a été reformulé, peut-être indépendamment, par Claude Shannon : « l'adversaire connaît le système ». Cette formulation est connue sous le nom de la maxime de Shannon. Il est considéré aujourd'hui comme un principe fondamental par les cryptologues, et s'oppose à la sécurité par l'obscurité.

Le principe de Kerckhoffs n'implique pas que le système de chiffrement soit public, mais seulement que sa sécurité ne repose pas sur le secret de celui-ci. Une tendance plus récente est de considérer que quand les systèmes de chiffrement sont publics, largement étudiés et qu'aucune attaque significative n'est connue, ils sont d'autant plus sûrs.

11. Sources et références ayant permis l'élaboration de ce TP

Recommandations pour un usage sécurisé d'(Open)SSH, publié par l'ANSSI le 17 août 2015

SSH, The Secure Shell : The definitive guide, de Daniel Barrett et Richard Silverman aux éditions O'Reilly, publié en février 2001

La fin annoncée des autorités de certification, alternatives : TOFU, Convergence, CATA, Clés souveraines, DANE publié en 2011 par Florian Maury, spécialiste sécurité des services et des réseaux

Les supports pédagogiques cybersécurité proposés par le label CyberÉdu.

Logiciel permettant de réaliser un SSH MITM
<https://github.com/jtesta/ssh-mitm>

Informations sur les différents fichiers clients et serveurs nécessaires au fonctionnement de SSH
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/4/html/reference_guide/s1-ssh-configfiles

Directives qu'il est possible d'ajouter dans le fichier de configuration du serveur SSH
https://man.openbsd.org/OpenBSD-6.0/sshd_config.5

Les bonnes pratiques à adopter pour améliorer la sécurité d'un serveur SSH
<https://www.cyberciti.biz/tips/linux-unix-bsd-openssh-server-best-practices.html>

La liste des algorithmes cryptographiques recommandée par l'entreprise Mozilla
<https://infosec.mozilla.org/guidelines/openssh.html>

Pourquoi est-il recommandé de ne plus utiliser l'algorithme RSA quand cela est possible pour générer une paire de clés SSH ?
<https://blog.g3rt.nl/upgrade-your-ssh-keys.html>

Définition Wikipédia de l'échange de clés Diffie-Hellman
https://fr.wikipedia.org/wiki/%C3%89change_de_cl%C3%A9s_Diffie-Hellman

Guide d'utilisation de l'outil nmap
<https://nmap.org/man/fr/>

Définition Wikipédia de l'attaque MITM
https://fr.wikipedia.org/wiki/Attaque_de_l'homme_du_milieu

Définition Wikipédia de l'attaque ARP Poisoning
https://fr.wikipedia.org/wiki/ARP_poisoning

Définition Wikipédia du principe de Kerckoffs
https://fr.wikipedia.org/wiki/Principe_de_Kerckhoffs

Définition Wikipédia du principe de l'entropie de Shannon
https://fr.wikipedia.org/wiki/Entropie_de_Shannon

Quelques mots sur la cryptographie, exposé au séminaire étudiant du LSP, de Djalil Chafaï, 1999
<https://repo.zenk-security.com/Cryptographie%20.%20Algorithmes%20.%20Steganographie/Quelques%20mots%20sur%20la%20cryptographie.pdf>

Comment augmenter l'entropie sur un serveur ?
<https://www.yakati.com/art/comment-augmenter-l-entropie-sur-un-serveur-avec-havaged.html>

Article expliquant le principe de la génération d'une entropie externe lors de la création de clés de chiffrement sur un ordinateur
<https://www.deltasight.fr/entropie-linux-generation-nombres-aleatoires/>

Définition Wikipédia du protocole SSHFP
https://fr.wikipedia.org/wiki/Enregistrement_DNS_SSHFP

Explications concernant la RFC 4255 sur le protocole SSHFP
<https://www.bortzmeyer.org/4255.html>

Article qui décrit la mise en œuvre de la technologie SSHFP
<https://quentin.demouliere.eu/2016/03/20/sshfp-faciliter-l-authentification-d-un-serveur-ssh-depuis-un-client.html>