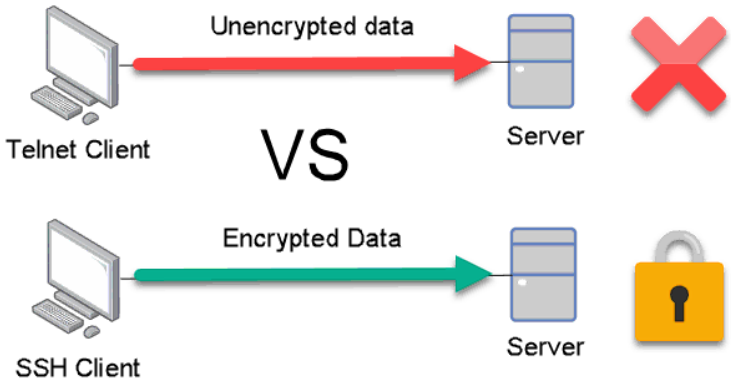
UNIVERSITÉ CAEN NORMANDIE

SSH 06.03.2023

Attention aux pirates ☹

Auteur : Pascal Fougeray



source : https://networkverge.com/telnet-vs-ssh/

1 Introduction

Dans le domaine des réseaux, il est nécessaire de pouvoir se connecter à distance sur une machine distante...
Il fut un temps que les moins de 20 ans ne peuvent pas connaître... il y avait **telnet**
C'était très bien, on saisissait son login et son mot de passe et si un méchant pirate sniffait le réseau à ce moment là et bien il voyait tout en clair, c'est ballot ...
Puis vint le monde moderne du chiffrement et ssh
Il est devenu un outil indispensable aux administrateurs systèmes mais pas qu'à eux à tout le monde !

2 Qu'est-ce que SSH ?

SSH signifie ***Secure Shell***, il s'agit d'un **shell sécurisé**.
Je rappelle qu'un **shell** est le premier logiciel qui tourne dans un **terminal.**
Il permet de dialoguer avec un serveur via l'exécution de différentes commandes qui retourneront des informations.
SSH est une application donc sur la couche 7 de OSI ou 5 de TCP/IP.
Il s'appuie sur SSL (Ah non 'est TLS...), couche 4 de OSI.

3 Qu'est-ce que SSL ?

Le protocole SSL, ***Secure Sockets Layer*** était le protocole cryptographique le plus largement utilisé pour assurer la sécurité des communications sur Internet avant d'être précédé par TLS ***Transport Layer Security*** en 1999.
Malgré la **dépréciation** du protocole SSL et l'adoption du TLS pour le remplacer, le terme SSL est toujours le seul utilisé pour faire référence à ce type de technologie. (Même le prof l'utilise...)
SSL crée un canal sécurisé entre 2 machines communiquant sur Internet ou un réseau interne.
Son usage le plus courant est la sécurisation de la communication entre un navigateur web et un serveur web.
L'adresse URL passe alors de HTTP (port 80) a HTTPs (port 443), le **s** signifiant **sécurisé**.
Il permet 3 choses :

1. Le **chiffrement** : **protège** les transmissions de données.
2. L'**authentification** : **garantit** que le serveur auquel vous êtes connecté est le **bon** serveur.

3. L'**intégrité des données** : **garantit** que les données qui sont demandées ou soumises **sont bien celles** qui sont fournies.

Je ne vais pas m'attarder sur SSL qui pourra être vu en Master à Caen ou dans un autre module sur la sécurité des réseaux dans une autre formation.

Pour plus d'informations voir le site : **https ://www.openssl.org/**

4 Telnet c'est fini !

4.1 Rappel Telnet

Telnet est un protocole permettant d'émuler un terminal à distance, cela signifie qu'il permet d'exécuter des commandes saisies au clavier sur une machine distante. Le programme Telnet est une implémentation de ce protocole Telnet.

Telnet fonctionne en mode client/serveur, c'est-à-dire que la machine distante est configurée en serveur et par conséquent attend qu'une machine distante lui demande un service. Ainsi, étant donné que la machine distante envoie les données à afficher, l'utilisateur a l'impression de travailler directement sur la machine distante. Sous Linux, le service est fourni par ce que l'on appelle un démon, une petite tâche qui fonctionne en arrière-plan. Le démon Telnet s'appelle **Telnetd**.

```
gns3@box:~$ telnet
Usage: telnet [-a] [-l USER] HOST [PORT]

Connect to telnet server

        -a       Automatic login with $USER variable
        -l USER Automatic login as USER

gns3@box:~$
```

Telnet

On n'utilise plus telnet, sauf sous GNS3 mais là c'est un cas particulier. Il est impossible de venir écouter ce qui se dit entre 2 machines en réseaux !

Le souci de Telnet c'est que le MDP est en clair, c'est ballot ☺

On l'utilisera lors du TP sur les protocoles, mais ce n'est qu'un TP !

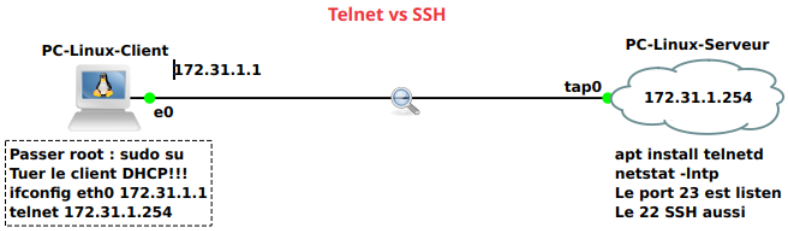
Il est facile d'installer (du moins sous une debian !) : ***apt install telnetd***

On vérifie qu'il fonctionne à l'aide de la commande ***netstat -lntp*** et si on voit le port 23 c'est que tout est ok.

Si vous avez ubuntu c'est plus "chiant", je parle par expérience récente

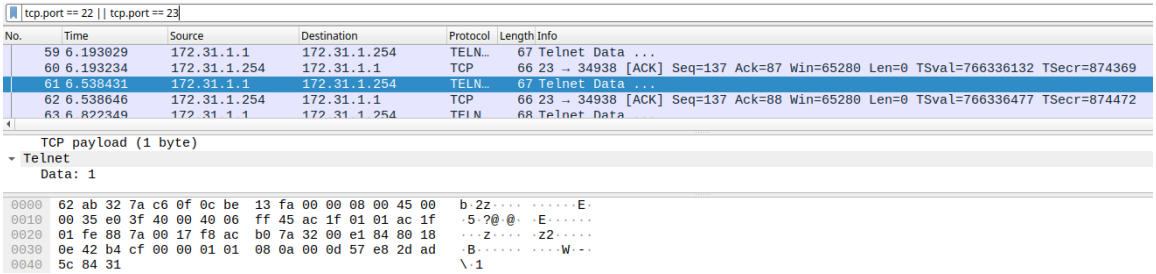
Voir le site : https://linuxways.net/ubuntu/how-to-install-telnet-server-and-client-on-ubuntu/

Il y a juste une erreur : Dans le fichier /etc/xinetd.d/telnet remplacer **server = /usr/sbin/in.telnetd** par **server = /usr/sbin/telnetd**



Du PC-client au PC-serveur : telnet 172.31.1.254

Si on fait une capture wireshark on obtient :

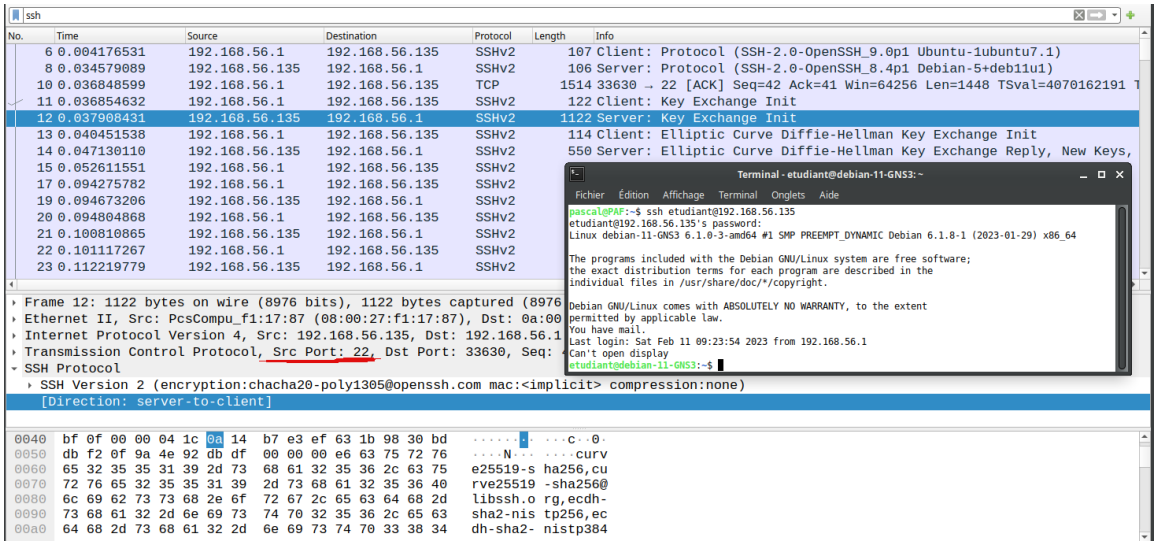


Ici on voit le 1 de Etudiant1, car chaque caractère occupe un paquet !

Le mot de passe passe en clair... !

Alors qu'en SSH **Le mot de passe est chiffré**, comme le montre la figure suivante, difficile de trouver le MDP Etudiant1☺

C'est juste un ***ssh etudiant@192.168.56.135*** donc du Host vers la VM en passant par l'interface VboxNet0 du Host vers l'interface Enp0s9 de la VM.



On peut voir que l'algorithme de chiffrement est chacha20 (C'est quoi cette bête un chat à 2 têtes ☺?)

C'est un algorithme de chiffrement sur 256 bits qui remplace AES, très à la mode

Plus d'informations ici : https://en.wikipedia.org/wiki/Salsa20#ChaCha_variant

Voici son algo en Langage C

ChaCha20 uses 10 iterations of the double round.[20] An implementation in C/C++ appears below.

```c
#define ROTL(a,b) (((a) << (b)) | ((a) >> (32 - (b))))
#define QR(a, b, c, d) (            \
    a += b,  d ^= a,  d = ROTL(d,16),   \
    c += d,  b ^= c,  b = ROTL(b,12),   \
    a += b,  d ^= a,  d = ROTL(d, 8),   \
    c += d,  b ^= c,  b = ROTL(b, 7))
#define ROUNDS 20

void chacha_block(uint32_t out[16], uint32_t const in[16])
{
    int i;
    uint32_t x[16];

    for (i = 0; i < 16; ++i)
        x[i] = in[i];
    // 10 loops × 2 rounds/loop = 20 rounds
    for (i = 0; i < ROUNDS; i += 2) {
        // Odd round
        QR(x[0], x[4], x[ 8], x[12]); // column 0
        QR(x[1], x[5], x[ 9], x[13]); // column 1
        QR(x[2], x[6], x[10], x[14]); // column 2
        QR(x[3], x[7], x[11], x[15]); // column 3
        // Even round
        QR(x[0], x[5], x[10], x[15]); // diagonal 1 (main diagonal)
        QR(x[1], x[6], x[11], x[12]); // diagonal 2
        QR(x[2], x[7], x[ 8], x[13]); // diagonal 3
        QR(x[3], x[4], x[ 9], x[14]); // diagonal 4
    }
    for (i = 0; i < 16; ++i)
        out[i] = x[i] + in[i];
}
```

À voir en TP de Parallélisme ?

5 SSH

Lors de la connexion vers une machine distante, ssh demande à l'utilisateur son MDP. Imaginez un administrateur système et/ou réseau qui doit exécuter cette opération des dizaines de fois par jour...

Heureusement que tout est bien fait dans le meilleur des mondes du réseau

5.1 SSH avec authentification par clés

Avec SSH, l'authentification peut se faire sans l'utilisation de MDP ou de phrase secrète en utilisant la **cryptographie asymétrique**.

Le principe est le suivant :

1. La **clé publique** est distribuée sur les systèmes distants auxquels on souhaite se connecter.
2. La **clé privée**, protégée par un MDP ou pas, reste sur le poste à partir duquel on se connecte.
3. L'utilisation d'un ńăagent sshăż permet de stocker le mot de passe de la clé privée pendant la durée de la session utilisateur.

Cette configuration est aussi utilisée par les utilitaires **scp** (***Secure CoPy***) et **sftp** (***Simple File Transfer Protocol***) qui se connectent au même serveur.

6 En pratique sous Linux

Il est facile d'installer et de configurer un serveur SSH sous Linux quelque soit la distribution que l'on utilise.

6.1 Installation

Un simple ***apt install openssh-server*** et le tour est joué.
Vous pouvez vérifier avec la commande ***netstat -lntp*** qu'il écoute bien sur le **port 22**
Attention si vous voulez pouvoir vous loguer à distance en tant que root, il faudra modifier le fichier de configuration **/etc/ssh/sshd_config**

1. **Éditez** le fichier
2. **Naviguez** dans le fichier, et **recherchez** la ligne **PermitRootLogin without-password**
3. **Modifiez** ensuite la ligne comme suit **PermitRootLogin yes**
4. **Sauvegardez** le fichier de configuration
5. **Redémarrez** le service SSH à l'aide de la commande : ***/etc/init.d/ssh restart***
6. **Vérifiez** que cela fonctionne

6.2 La commande et le protocol SCP

Avec **ssh** on peut se loguer, mais on eut aussi utiliser la commande **scp** (pour ***secure copy***)
Voici l'exemple de commande SCP Linux de base :
scp [autres options] [nom d'utilisateur source@IP] :/[dossier et nom de fichier] [nom d'utilisateur de destination@IP] :/[dossier de destination]
Exemple :
scp etudiant@162.168.1.1 :/repertoire/fichier.txt autre_login@162.168.1.2 :/bureau/fichier.txt
Ne pas oublier les deux points :
Si vous voulez utiliser un autre port que le 22 par exemple 1664.
scp -P 1664
Quelques options :
-P permet de spécifier un port différent au serveur
-c vous donne la possibilité de spécifier l'algorithme de chiffrement que le client utilisera.
'aes256-ctr', 'aes256-cbc', 'blowfish-cbc', 'arcfour', 'arcfour128', 'arcfour256', 'cast128-cbc', aes128-ctr', 'aes128-cbc', 'aes192-ctr', 'aes192-cbc', et 3des-cbc'. L'option par défaut dans la configuration du shell est ńăAnyStdCipherăż.
-q en mode silencieux, seules les erreurs critiques sont affichées.
-r est pour la copie récursive, qui comprendra tous les sous-répertoires.
-4 ou -6 peuvent être utilisés si vous souhaitez choisir la version de protocole utilisée, soit IPv4 ou IPv6.
-p préserve les temps de modification et les attributs initiaux du fichier. TB de Linux à Linux !
-u supprime le fichier source une fois le transfert terminé.
-c permet la compression des données pendant l'opération de transfert. TB si vous avez une bande passante en ADSL ☹

6.3 Le protocole sftp

A surtout ne pas confondre avec **Simple File Transfer Protocol**, un protocole de transfert de fichiers, non sécurisé et rarement utilisé !
Ici SFTP veut dire SSH File Transfer Protocol, c'est du FTP sécurisé.
Comparaison avec SCP
Comparé au protocole **scp** vu précédemment, le protocole **sftp** supporte beaucoup plus d'opérations sur des fichiers à distance.
Il se comporte plus comme un protocole de système de fichiers.
Il est plus indépendant de la plate-forme d'utilisation.
Certaines implémentations de la commande **scp** utilisent le protocole **sftp** à la place du protocole **scp**.

7 Sous Windows

Je donne dans ce cours rarement des exemples avec windows.

Mais là c'est pour vous aider si vous utilisez la VM chez vous et que vous voulez travailler, c'est donc pour une bonne cause ☺.

7.1 En tant que client

Je vous recommande d'utiliser le logiciel **MobaXterm** https://mobaxterm.mobatek.net/ qui servira de client ssh mais pas que.

En effet il supporte tous les protocoles suivants :

SSH, Telnet, Rsh, Xdmcp, RDP, VNC, FTP, SFTP, Serial File, Local Shell (avec le Bash WSL si installé, Powershell, Bash sur Windows, et le fameux DOS Prompt), Navigateur Web, Mosh (mobile shell) et S3...

7.2 En tant que serveur

Je vous recommande d'utiliser le logiciel **MobaSSH https://mobassh.mobatek.net/**

8 Conclusion

La couche 5 du modèle OSI est très très très utilisée ☺

On n'utilise plus qu'elle dans le Web avec http devenu https, imap devenu imaps, pop devenu pops, smtp devenu smpts, etc devenu etcs euh non ...

Et vous aller rire : telnet est devenu telnets en passant du port 23 au port 992 ☺

Je ne l'ai jamais utilisé...