Le pense-bête du hackeur

Club*Nix

ESIEE Paris

January 23, 2024

Contents

1 ls

La commande **ls** sur Linux est comme un agent secret qui vous permet de voir tout ce qui se trouve dans un dossier. Elle révèle tous les fichiers et dossiers présents dans le répertoire courant, en montrant leur nom, leur type, leur propriétaire et leurs permissions.

C'est un peu comme si vous regardiez dans un tiroir pour voir tous les objets qu'il contient. **ls** est l'outil idéal pour savoir où vous vous trouvez et pour naviguer dans les répertoires de votre ordinateur. Par exemple, si vous êtes dans le dossier Documents et que vous exécutez **ls**, vous verrez tous les fichiers et dossiers présents dans ce dossier.

Voir les fichiers et dossiers du dossier où vous vous trouvez (ici Documents)

```
kali@kali:~/Documents ls
Bureau
password.txt
lettre_demission.pdf
```

2 cd

La commande **cd** sur Linux vous permet de changer de dossier en utilisant la ligne de commande. C'est comme une carte qui vous permet de naviguer dans les dossiers de votre ordinateur. Vous pouvez utiliser **cd** pour passer d'un dossier à un autre en spécifiant le chemin d'accès relatif ou absolu du dossier que vous souhaitez visiter.

Un chemin **absolu** commence à la racine du système de fichiers, indiquant le chemin complet vers le dossier. Par exemple, pour accéder au dossier "Bureau", vous pouvez utiliser:

```
kali@kali:$ cd /home/kali/Bureau
kali@kali:/Bureau
```

Un chemin **relatif** est défini par rapport à votre emplacement actuel. Par exemple, si vous êtes déjà dans le dossier "/home/kali/", vous pouvez accéder au dossier "Bureau" de manière relative:

```
kali@kali:$ cd Bureau
kali@kali:/Bureau
```

Les symboles **.** et **..** sont utilisés pour représenter le dossier actuel et le dossier parent, respectivement. Par exemple:

```
kali@kali:/Bureau$ cd .
kali@kali:/Bureau

kali@kali:/Bureau$ cd ..
kali@kali:$
```

En utilisant ces concepts, la commande **cd** vous permet de naviguer efficacement dans la structure de dossiers de votre système, que ce soit en spécifiant des chemins absolus ou relatifs, ou en utilisant les symboles **.** et **..** pour représenter les dossiers actuel et parent.

3 sudo

La commande **sudo** est un peu comme donner des superpouvoirs à une commande. Normalement, lorsque vous exécutez des commandes dans un terminal, votre ordinateur vérifie si vous avez la permission nécessaire pour effectuer cette action. Cependant, certaines actions nécessitent des pouvoirs spéciaux, souvent réservés à l'administrateur du système.

sudo permet à un utilisateur de bénéficier temporairement de ces pouvoirs administratifs pour effectuer des tâches qui nécessitent des autorisations spéciales. Cela peut inclure l'installation de logiciels, la modification de configurations système, ou d'autres actions qui pourraient affecter l'ensemble du système.

Voici comment utiliser **sudo**:

```
kali@kali:~$ whoami
kali

kali@kali:~$ touch fichier.txt
touch: cannot touch 'fichier.txt': Permission denied

kali@kali:$ sudo touch fichier.txt
[sudo] password for kali:
kali@kali:$ ls
fichier.txt
```

Dans cet exemple, l'utilisateur **kali** tente de créer un fichier, mais reçoit une erreur de permission. En utilisant **sudo**, il peut exécuter la même commande avec des privilèges administratifs, lui permettant ainsi de créer le fichier avec succès.

En résumé, **sudo** est comme activer le mode "super-admin" pour une commande spécifique, vous permettant de faire des choses que vous ne pourriez pas normalement faire avec vos pouvoirs d'utilisateur standard. Cependant, cela doit être utilisé avec précaution, car des actions incorrectes peuvent avoir des conséquences sur le système.

4 ping

La commande **ping** est comme frapper à la porte d'un ordinateur pour voir s'il répond. Imaginez que chaque ordinateur a une porte, et en utilisant la commande **ping**, vous envoyez un petit message à cette porte. Si la porte est ouverte, l'ordinateur répond en renvoyant un message, indiquant qu'il est là et prêt à communiquer. Cela permet de vérifier si un ordinateur distant est accessible à travers le réseau. Voici comment cela se présente en pratique:

```
kali@kali:~$ ping 192.168.0.50
PING 192.168.0.50 (192.168.0.50) 56(84) bytes of data.
64 bytes from 192.168.0.50: icmp_seq=1 ttl=64 time=1.02 ms
64 bytes from 192.168.0.50: icmp_seq=2 ttl=64 time=1.20 ms
64 bytes from 192.168.0.50: icmp_seq=3 ttl=64 time=0.95 ms

--- 192.168.0.50 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.951/1.061/1.202/0.104 ms
```

Dans cet exemple, le message a été envoyé à l'adresse IP 192.168.0.50, et nous avons reçu des réponses. Cela signifie que l'ordinateur à cette adresse est accessible et réactif. Si la porte était fermée, l'ordinateur ne répondrait pas, indiquant qu'il n'est pas disponible ou hors ligne. C'est donc une manière simple de vérifier la connectivité entre deux ordinateurs sur un réseau.

5 nmap

nmap est la commande qui permet de sonder les ports ouverts sur un pc. La commande **nmap** est un outil qui permet de **scanner** les réseaux pour découvrir les appareils connectés et les services qui y sont proposés. C'est comme si vous étiez en train de faire une liste des appareils et des services disponibles sur un réseau.

```
kali@kali:~$ nmap 192.168.0.50
Starting Nmap 7.80 ( https://nmap.org ) at 2023-04-03 16:06 CEST
Nmap scan report for 192.168.0.50
Host is up (0.014s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

```

Dans notre cas, les ports 22, 139 et 445 sont ouverts.

6 hostname

hostname est une commande qui permet entre autre de voir le nom de l'ordinateur couramment utilisé, elle peut aussi permettre de le modifier.

```
kali@kali:~$ hostname
kali
kali@kali:~$ hostname gogole
hostname: you must be root to change the host name
kali@kali:~$ sudo hostname gogole
[sudo] password for kali:
kali@gogole:~$ hostname
gogole
```

7 ip a

La commande **ip a** sur Linux vous permet de consulter les informations sur les interfaces réseau de votre système. C'est comme obtenir une carte réseau détaillée qui montre les adresses IP et d'autres informations associées à chaque interface réseau.
Voici comment utiliser **ip a**:

```
kali@kali:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
link/ether 00:0c:29:28:fd:60 brd ff:ff:ff:ff:ff:ff
inet 192.168.0.100/24 brd 192.168.0.255 scope global dynamic noprefixroute eth0
valid_lft 1707sec preferred_lft 1707sec
inet6 fe80::20c:29ff:fe28:fd60/64 scope link noprefixroute
valid_lft forever preferred_lft forever
```

La première ligne "1: lo" représente l'interface **lo** (loopback), qui est une interface virtuelle utilisée pour les communications internes de l'ordinateur. L'adresse IP associée est 127.0.0.1, utilisée pour la communication locale.

La deuxième ligne "2: eth0" représente une interface réseau physique (**eth0** dans cet exemple). On y trouve l'adresse MAC, l'adresse IP (192.168.0.100 dans cet exemple), et d'autres informations relatives à cette interface.

Les lignes suivantes présentent des détails tels que la MTU (Maximum Transmission Unit), le type de file d'attente, l'état de l'interface, et d'autres informations spécifiques à chaque interface réseau.

La commande **ip a** fournit ainsi une vue détaillée des interfaces réseau de votre système, aidant à comprendre comment votre ordinateur est connecté au réseau et quelles adresses IP sont associées à chaque interface.

8 ssh

En utilisant la commande **ssh**, vous pouvez établir une connexion **sécurisée** avec un autre ordinateur et accéder à ses ressources à distance. Vous pouvez utiliser **ssh** pour accéder à un serveur distant, exécuter

des commandes sur cet ordinateur, transférer des fichiers, ouvrir des sessions interactives et bien plus encore. Lorsque le mot de passe est demandé vous devrez ecrire le mot de passe. Pour des raisons de securité le mot de passe n'est **jamais** affiché. Donc c'est est normal si vous ne voyez rien apparaitre lorsque vous tapez.

```
kali@kali:~$ ssh username@10.0.3.20
```

9 scp

La commande **scp** est utilisée pour copier des fichiers entre des ordinateurs distants de manière sécurisée. Cela permet de transférer des fichiers entre des ordinateurs situés sur différents réseaux, de manière à la fois rapide et sécurisée. Il est à noter que cette commande utilise **ssh** comme protocole pour fonctionner !

Exemple :

```
kali@kali:~$ scp fichier.txt utilisateur@10.0.3.10:/chemin/vers/dossier/
```

Dans cet exemple, nous copions le fichier "fichier.txt" depuis votre ordinateur local vers le dossier "/chemin/vers/dossier/" situé sur le serveur distant, en utilisant le nom d'utilisateur "utilisateur"

10 Execution de code executable

Sur Linux, le préfixe "./" est utilisé pour spécifier le chemin relatif vers un fichier exécutable dans le répertoire courant. Quand tu tapes "./nom_du_fichier", cela signifie que tu veux exécuter le fichier qui se trouve dans le même répertoire que celui où tu te trouves actuellement.

```
kali@kali:~$ ./nom_du_fichier
```

En d'autres termes, c'est une manière de dire au système d'exploitation Linux : "Hey, regarde dans le répertoire actuel (représenté par le './'), et exécute le fichier avec le nom que je t'ai donné."

11 Execution de code python

Vous voulez executer du code python? Rien de plus simple:

```
kali@kali:~$ python3 [nom du fichier python]
```

A noter que vous devez etre sur le meme repertoire que votre fichier sinon **ça ne marche pas** !! (ou alors il faut spécifier le chemin qui amène de votre position à celle du fichier)

12 Rappels programmation et Python

12.1 variables

Une variable permet de stocker une valeur. C'est un peu comme une boîte dans laquelle on va mettre un objet et où on va attacher un post-it pour lui donner un nom. Il est possible de modifier le contenu de cette boîte ou bien la détruire, c'est pareil pour les variables

```
>>> a = 5       # On met la valeur 5 dans la variable 'a'
>>> b = "J'aime le chiffre 5" # On met une chaîne de caractères (un message) dans la variable 'b'
>>> print(a)    # On affiche la valeur 'a' (donc c'est son contenu qui sera affiché)
5
>>> print(b)
J'aime le chiffre 5
>>>
```

12.2 Les types

Il existe différents types primitifs (c'est à dire qui existent par défaut et dont tous les autres types dépendent). Ils permettent de tout représenter et sont derrière toutes les variables qui sont créés, le système détecte automatiquement le type mais si ce n'est pas précisé. En python ces types sont :

- Entier (nombres) : int
- Booléen (Vrai ou Faux) : bool
- String (Chaîne des caractères) : str
- Flottant (nombre à virgules, dits flottants) : float
- Liste (Une liste de valeurs) : list
- Et bien d'autres comme les dict, les tuple etc... mais pas utiles pour aujourd'hui

12.3 Les conditions

Une condition est un bloc de code qui va permettre de vérifier une condition (par exemple une égalité entre deux variables, comme "Est ce que ces deux boîtes possèdent le même contenu ?").
Cela permet de passer de séparer son code en différentes sections si l'on souhaite par exemple ne pas exécuter certaines lignes si l'utilisateur n'a pas 18 ans ou n'est pas membre du Nix. Il y a différents opérateurs pour vérifier une condition :

- `var1 == var2` : Vérifier **l'égalité** entre deux variables
- `var1 != var2` : Vérifier si les deux variables sont **différentes**
- `var1 < var2` : Vérifier si var1 est **inférieure** à var2
- `var1 > var2` : Vérifier si var1 est **supérieure** à var2
- `var1 <= var2` : Vérifier si var1 est **inférieure ou égale** à var2
- `var1 >= var2` : Vérifier si var1 est **supérieure ou égale** à var2
- `___ or ___` : Vérifier si le bloc de gauche **ou** le bloc de droit sont "Vrai"

- `___` and `___` : Vérifier si le bloc de gauche **et** le bloc de droit sont "Vrai"

Les conditions reviennent à obtenir une valeur booléenne (Vrai ou Faux), ainsi `var1 >= var2` revient à faire `var1 > var2 or var1 == var2` !
Ces conditions sont utilisées dans des blocs de code if/else qui vont vérifier la condition et donner accès au bloc de code voulu

```
a = 5
b = 25

if a < b:
    # Bloc de code que l'on utilise si a est inférieur à b
else:
    # Bloc de code que l'on utilise si b est supérieur ou égal à b
# Retour au bloc de code commun
```

12.4 Les boucles

Les boucles sont un moyen de réaliser de répéter un bloc de code. Il en existe deux versions, la boucle **for** et la boucle **while**. Leur principale avantage est d'éviter la répétition de ligne de code et donc de rendre le code plus lisible et facile.

La boucle **for** en python va boucler dans tous les éléments d'un objet (par exemple une liste, ou les caractères d'un string). Celle-ci se présente de la manière suivante :

```
# Ce bloc va afficher les chiffres de 0 à 25 !
for i in range(25):
    print(i)

# Ce bloc va afficher chaque caractère un à un de la variable txt
txt = "We hope you have a good time"
for j in txt:
    print(j)
```

La boucle **while** quant à elle va exécuter en continu son bloc de code tant que sa condition de vérification est vraie :

```
# Code pour afficher tous les entiers jusqu'à 0
a = 25
while a > 0:
    print(i)
    a = a - 1

# Boucle infinie
while True:
    print("Oups...")
```

12.5 Les fonctions

Une fonction est un bloc de code qui peut être ”exécuté à distance”. Une fonction va contenir un bloc de code et peut être appelée à d'autres endroits dans le programme. Lorsqu'elle est appelée alors son contenu va être exécuté, de plus vous pouvez lui donner des informations utiles à l'exécution. C'est exactement la même chose lorsque vous remplissez un formulaire sur un site internet, vous donnez les informations requises et vous appuyez sur un bouton qui va exécuter du code avec ces informations !

```
# Fonction pour afficher le nom envoyé par l'utilisateur
def afficherNom(nom):
    print(nom)

var = "Hitsuji"
afficherNom(var) # On appelle la fonction 'afficherNom' avec le nom de l'utilisateur
```