Introduction to Linux

Iker M. Canut

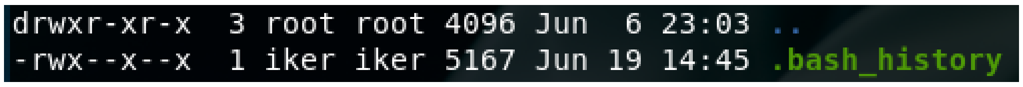
June 28, 2020

1 Locate

If you want to locate a file you can use the **locate** command. If you do not find something, you can try running **updatedb**, because is has to build a database of the information that it's finding in order to locate what you're searching for.

2 Permissions

For the first letter of **ls -la**, if we have a **-** that is a file, if we have a **d** that is a directory.



```
drwxr-xr-x  3 root root 4096 Jun  6 23:03 ..
-rwx--x--x  1 iker iker 5167 Jun 19 14:45 .bash_history
```

Number	Permissions
0	No permissions
1	Execute
2	Write
3	Write, Execute
4	Read
5	Read, Execute
6	Read, Write
7	Read, Write, Execute

User	Group	Others
- - -	- - -	- - -
- - x	- - x	- - x
- w -	- w -	- w -
- w x	- w x	- w x
r - -	r - -	r - -
r - x	r - x	r - x
r w -	r w -	r w -
r w x	r w x	r w x

3 The System

3.1 Color Coded

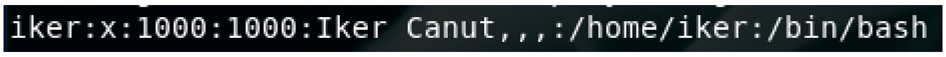
- Blue: Directory.
- Green: Executable or recognized data file.
- Sky Blue: Symbolic link file.
- Yellow with black background: Device
- Pink: Graphic image file.
- Red: Archive file.
- Red with black background: Broken link.

3.2 The Filesystem Hierarchy Standard (FHS)

- **/bin**: Essential user command binaries.
- **/etc**: Configuration files for the system.
- **/sbin**: Essential system binaries.
- **/usr**: Read-only user application support data & binaries.
  - /usr/bin: Most user commands.
  - /usr/include: Standard include files for C code.
  - /usr/lib: Obj, bin, lib files for coding and packages.
  - /usr/local: Local software (./bin ./lib ./man ./sbin ./share).
  - /usr/share: Static data shareable across all architectures.
    - /usr/share/man: Manual pages.
- **/var**: Variable data files.
  - /var/cache: Application cache data.
  - /var/lib: Data modifies as programmes run.
  - /var/lock: Lock files to track resources in use.
  - /var/log: Log files.
  - /var/opt: Variable data for installed packages.
  - /var/spool: Tasks waiting to be processed:
    - /var/spool/cron
    - /var/spool/cups
    - /var/spool/mail
  - /var/tmp: Temporary files saved between reboots.
- **/dev**: Device files included /dev/null.
- **/home**: User home directories.
- **/lib**: Libraries & kernel modules.
- **/mnt**: Mount files for temporary filesystems.
- **/opt**: Optional software applications.
- **/proc**: Process & kernel information files.
- **/root**: Home directory for the root user.
- **/tmp**: Is a directory used to hold temporary files. Is is a d777, so we might upload exploits to that directory, because we can execute them there.

4 Add users

To add users we can use the **adduser [name]** command from the terminal. Then we can write its password. Now, if we **cat /etc/passwd**, we can see a new user. But it no longer has the password there. Password are now in the shadow file.



```
iker:x:1000:1000:Iker Canut,,,:/home/iker:/bin/bash
```

So today we have a little bit of access and information disclosure here, at hands of poor configuration. We know the name and we could use SSH and that username to try to break into the machine, but we have no hash available.

If we **cat /etc/shadow** we can see the passwords, hashed. Therefore, we can use a tool to break it down. It might not be easy, but there is a good chance of cracking a password.

If we create a new user, it will have lower priviliges. But we can change this: anybody in the sudoers file can use sudo and have root access.

5 Basic Networking Commands

Some useful basic commands:

- **ping**: Pinging an IP address or domain name.
- **ifconfig**: Find interface information. Enumerates the interfaces and shows their IP Addresses, the network addresses, the broadcast addresses, packing information,... And then the Loop Back interface that's running on the system as well.
- **ifconfig [interface] promisc**: You turn an interface into promiscuous mode. Making an interface promiscuous is essentially going to mean that is going to capture traffic it wouldn't otherwise pick up on or listen to. So it's going to pick up on everything and listen to everything. This could be good for packet captures. To turn off the promiscuous mode of an interface, just write the next command: **ifconfig [interface] -promisc**.
- **traceroute**: Is a troubleshooting tool that can help you map out a path, see the route to a particular domain or maybe an internal system, ...
- **netstat**: It's going to give you a slew of information about the different things that are listening and running on a system.
- **netstat -ano**: Show the active connections that are running on your machine. You can see what's open and what's talking.
- **nslookup**: DNS query. It shows public DNS information.
- **dig**: Domain Information Groper. It's like nslookup but it shows more information, such us feedback from our internal DNS server.
- **iwconfig**: Lists wireless extensions.
- **arp -a**: Show the IP Addresses it talks to and the MAC Addresses associated with them.
- **route**: This will print your routing table: it tells you where your traffic exits. There could be a machine that has multiple routes, so you might see 2 different IPs because it might have a dual home NIC, so it's talking to a completely different network that you didn't know it existed.

6 Services

- **systemctl [service-name] start**: Start now a service.
- **systemctl [service-name] stop**: Stop a service that is running.
- **systemctl [service-name] enable**: Enable a service (start it on boot).
- **systemctl [service-name] disable**: Disable a service (it won't start on boot).

Or instead of **systemctl [service-name] start** you can use **service [service-name] start**.

6.1 Apache2

For example, to start an Apache Server: **service apache2 start**

6.2 Python Server

To start a simple server with python: **python -m SimpleHTTPServer 80**, or 8080 so it doesn't have problems with Apache. Also you can create your FTP Server with another module.

6.3 Metasploit

To make Metasploit start faster, you can enable the service: **systemctl enable postgresql**. Metasploit can take some time to start, and this is a tool that we're going to use quite a bit.

7 Update and Install

apt update && apt upgrade. The first thing that appears are the repositories. You check what's updated and what's not.

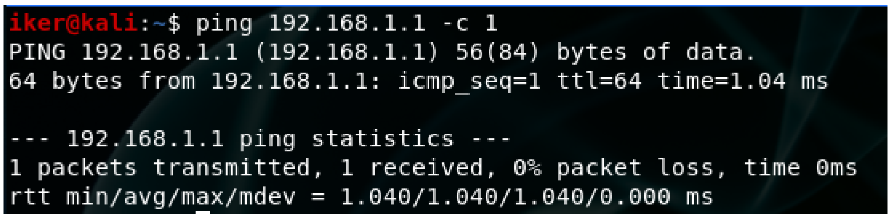
7.1 PIP

apt install python-pip. Then we can clone a repository, for example the *impacket* package:
git clone https://github.com/SecureAuthCorp/impacket.git

A good practice is to install all tools in /**opt**. In this case, to install this programme (check the README), just unpack it and run **pip install .** from the directory.

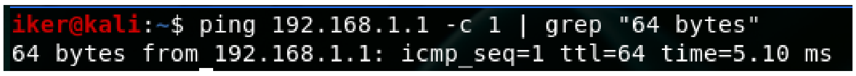
8 Scripting with Bash

In this example, we're gonna code a ping sweeper. To start off, lets analyze a **ping** response:



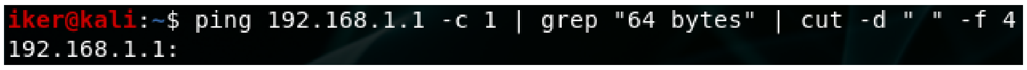
```
iker@kali:~$ ping 192.168.1.1 -c 1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.04 ms

--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.040/1.040/1.040/0.000 ms
```

But we wanna narrow this down, so we can extract the IP, so we could pipe it to a **grep**:



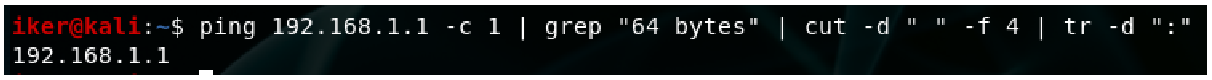
```
iker@kali:~$ ping 192.168.1.1 -c 1 | grep "64 bytes"
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=5.10 ms
```

Then, we can **cut** it. The **-d** is the delimiter, and the **-f** is which part we're going to return.



```
iker@kali:~$ ping 192.168.1.1 -c 1 | grep "64 bytes" | cut -d " " -f 4
192.168.1.1:
```

But we have a colon at the end, so to get rid of it, we can **tr** it. Tr stands for translate, and the **-d** is another delimiter.



```
iker@kali:~$ ping 192.168.1.1 -c 1 | grep "64 bytes" | cut -d " " -f 4 | tr -d ":"
192.168.1.1
```

8.1 Scripting

We can create file named *ipsweep.sh* and write this:

```
#!/bin/bash

if [ "$1" == "" ]
then
echo "You forgot an IP address!"
echo "Syntax: ipsweep.sh 192.168.1"

else
for ip in 1..254
do
 ping -c 1 $1.$ip | grep "64 bytes" | cut -d " " -f 4 | tr -d ":" &
done
fi
```

At the very top, we need to declare the **hashbang**: #!/bin/bash. We need it in all our scripts. The **for** is a simple for loop in bash. The $... are variables: **$ip** is the iterator, **$1** is user input. The **&** is for having multiple threads.

Then we can save it to a file, like for example **ipsweep.sh 192.168.1 > iplist.txt**

8.2 Looping in one line

If we want to do an nmap scan on all these IP addresses, we could simply:

```
for ip in $(cat iplist.txt); do nmap -sS -p 80 -T4 $ip; done
```