ITC8250 - Cyber Security Technologies I
Group Work 2

214322IV - Maëlie LEBARON
214366IV - Jordan Béziaud
214307IV - Vincent Rossignol
214319IV - Marine Hervet
214310IV - Victor Thévin

November 1, 2021

1 Telnet and Remote Shell

1.1

We started the packet sniffer on the network interface of mallet with the following commands :

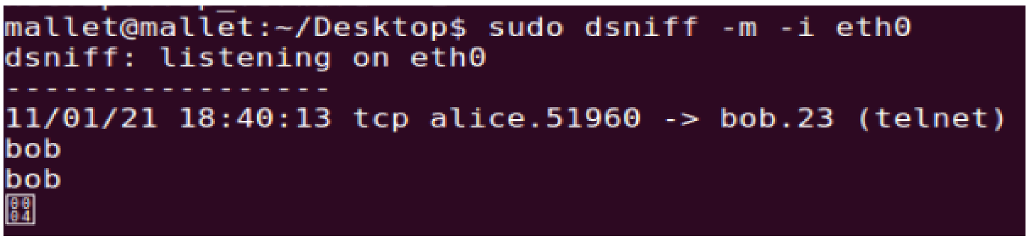
```
sudo sysctl -w  net.ipv4.ip_forward=1
sudo dsniff -m -i eth0
```

1.2

We can open the telnet session to bob with the following command :

```
telnet 192.168.1.1
```

1.3 What does the program dsniff display?



```
mallet@mallet:~/Desktop$ sudo dsniff -m -i eth0
dsniff: listening on eth0
-----------------
11/01/21 18:40:13 tcp alice.51960 -> bob.23 (telnet)
bob
bob
```

Figure 1: dnsiff display

Using the `dsniff` command-line tool, we can analyze the packets and retrieve all the password that are in clear in the LAN. As Alice opened a telnet session and entered bob's credentials, `dsniff` retrieved, therefore, we got Bobs credentials !

2 SSH

2.1

We enter the following command to create a ssh-key :

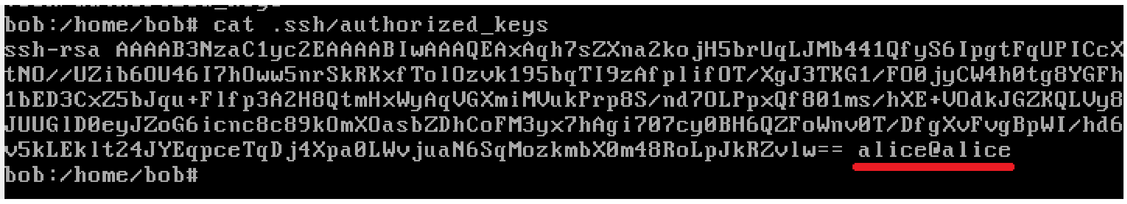
```
ssh-keygen -t rsa
```

2.2

Using the newly generated key, we can use `ssh-copy-id bob@192.168.1.1` to give to Bob the **public key**. Then, we can directly connect to Bob machine with `ssh` using `ssh bob@192.168.1.1`

2.3 In order to access machine bob from alice, what needs to be present on bobs machine?

There must be the public key generated by Alice using the `ssh-keygen` command-line tool in Bob's ssh authorized keys file (`~/.ssh/authorized_keys`). By the way, the `ssh-copy-id` tool automatically shares the public key on a remote user and even create the `authorized_keys` files if it's not created yet on the target user's machine !

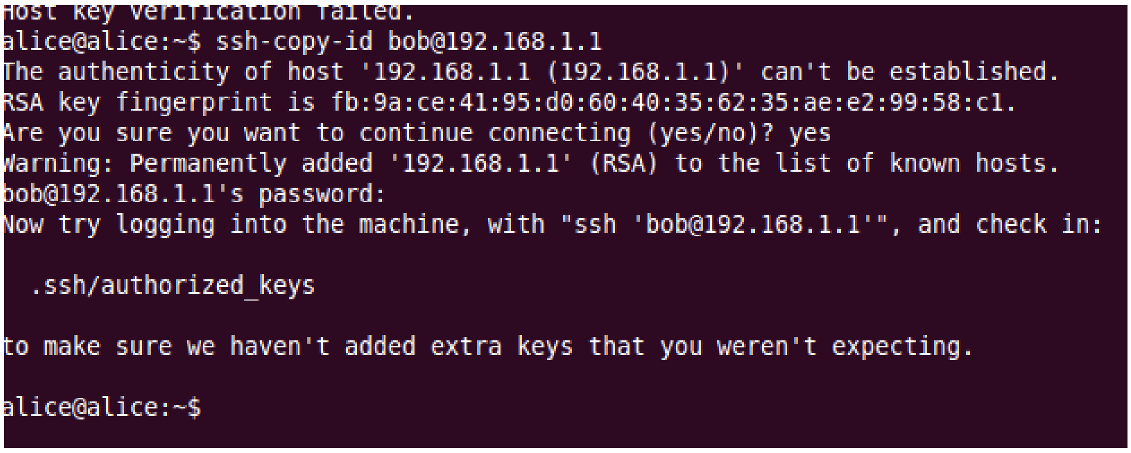


```
bob:/home/bob# cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAxAqh7sZXna2kojH5brUqLJMb441QfyS6IpgtFqUPICcX
tNO//UZib6OU46I7hOww5nrSkRKxfTolOzvk195bqTI9zAfplifOT/XgJ3TKG1/FO0jyCW4h0tg8YGFh
1bED3CxZ5bJqu+Flfp3AZH8QtmHxWyAqVGXmiMVukPrp8S/nd7OLPpxQf801ms/hXE+VOdkJGZKQLVy8
JUUGlD0eyJZoG6icnc8c89kOmXOasbZDhCoFM3yx7hAgi707cy0BH6QZFoWnv0T/DfgXvFvgBpWI/hd6
v5kLEklt24JYEqpceTqDj4Xpa0LWvjuaN6SqMozkmbX0m48RoLpJkRZvlw== alice@alice
bob:/home/bob#
```

Figure 2: Bob's ssh authorized keys

2.4 Where did you store the public key file?

After creating the key using `ssh-keygen`, it's stored in the public key file : `/.ssh/id_rsa.pub` (and `~/.ssh/id_rsa` for the private key counterpart)



```
Host key verification failed.
alice@alice:~$ ssh-copy-id bob@192.168.1.1
The authenticity of host '192.168.1.1 (192.168.1.1)' can't be established.
RSA key fingerprint is fb:9a:ce:41:95:d0:60:40:35:62:35:ae:e2:99:58:c1.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.1' (RSA) to the list of known hosts.
bob@192.168.1.1's password:
Now try logging into the machine, with "ssh 'bob@192.168.1.1'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

alice@alice:~$
```

Figure 3: Installing Bob's key on Alice's

2.5 What are the differences between authentication based on public keys and password-based authentication?

A password is a user created a secret phrase while a key is a system created phrase used to lock and unlock cryptographic functions.

Passwords and keys are used in encryption and authentication, but keys are also used in authorization, digital certificates, secure communication or non-repudiation though.

Keys are created by algorithms that are designed to make them difficult to guess. They are typically based on random or pseudorandom data. Because passwords are designed to be remembered, they are not as secure as keys as they are created by users. Users commonly use dictionary words and other patterns in passwords that make them easier to guess than keys. They are easier to store compared to huge 4096 bits keys which is convenient for businesses that needs spaces in their databases.

Passwords that are used for encryption are typically converted to a key first using a type of algorithm known as a key derivation function. For example, a cryptographic hash function may be used to create keys from passwords.

2.6 What are the security-relevant advantages of authentication using public key cryptography compared to the IP address-based authentication of rsh? What is the main threat that still remains?

- Using the key-pair approach, there is no need to input a password anymore when authenticating through ssh, therefore avoiding man-in-the-middle attacks with packets sniffers such as `dsniff` to steal credentials (as in the example with Mallet just before)
- The main threat that remains is that the keys can be stolen (as passwords) by the attacker if he knows when they are gonna be sent and how to detect them.

3 User IDs and Permissions

3.1 Use "find" command to find all world readable files in /etc and /var/log. Which files or directories might not be configured properly? Which permissions could be more restrictive?

By using the find command with the following syntax `find /etc /var/log -type f - perm -a=r`
The following directories (and the files inside) and files should have more restrictive permissions :

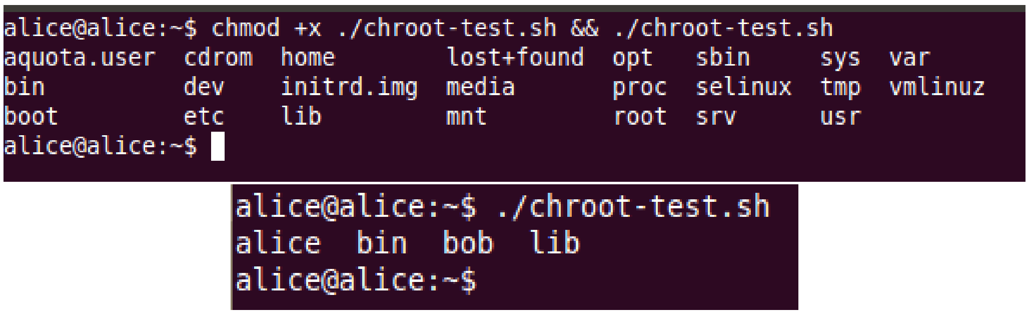
- `/etc/init.d/` because the attacker can see which scripts to modify and then let the server reboot (or make him crash using DDOS attacks for example) to execute the malicious payload in it !
- `/etc/grub.d/` and especially the `10_linux` file, for the same reason as above
- `/etc/crond.daily/` because the attacker know which scripts to infect, and if he succeeds then the malicious payload will be executed daily like keylogging, daily database leaking, ...
- `/etc/bluetooth/` because the attacker knows which devices connected through bluetooth to the target to infect and execute an indirect attack, or even read sensitives data that can be exchanged through this protocol.
- `/var/log/ConsoleKit/history.*.gz` because the attacker can see the commands or even the typing errors of the user, for example a password leaked while the user was thinking that he was logging in but typed his password in clear

3.2 Determine the default umask value, which is set in /etc/profile on alice. Why is this a good default?

The umask 022 which is translated in 755 permissions for all directories and 644 permissions for files created by Alice, means that only Alice can have read/write/execute accesses on it, and all the users from her group and the other users have read-only access to Alice's files and read-execute access to Alice's directories.

4 Change Root

4.1 `chroot-test.sh` execution



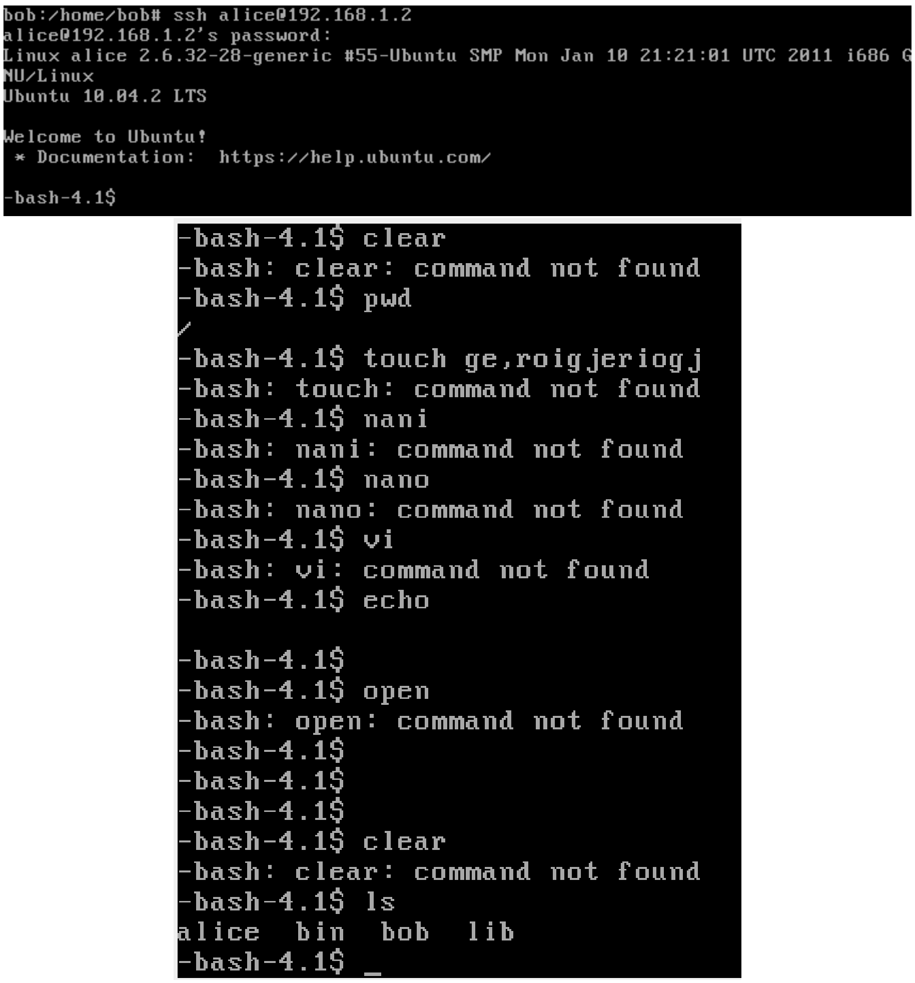
```
alice@alice:~$ chmod +x ./chroot-test.sh && ./chroot-test.sh
aquota.user  cdrom  home        lost+found  opt   sbin     sys  var
bin          dev    initrd.img  media       proc  selinux  tmp  vmlinuz
boot         etc    lib         mnt         root  srv      usr
alice@alice:~$ █
```

```
alice@alice:~$ ./chroot-test.sh
alice  bin  bob  lib
alice@alice:~$
```

Figure 4: chroot-test.sh execution

4.2 ssh config for chrooted jail

- We can edit the sshd config file to create the chrooted environment with `nano /etc/ssh/sshd_config`
- We can specify the root jail dir by adding the following lines at the end :
- And then restart the ssh service with `sudo service ssh restart`
- And finally when login from another user to Alice's machine through ssh :



```
bob:/home/bob# ssh alice@192.168.1.2
alice@192.168.1.2's password:
Linux alice 2.6.32-28-generic #55-Ubuntu SMP Mon Jan 10 21:21:01 UTC 2011 i686 G
NU/Linux
Ubuntu 10.04.2 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

-bash-4.1$
```

```
-bash-4.1$ clear
-bash: clear: command not found
-bash-4.1$ pwd
/
-bash-4.1$ touch ge,roigjeriogj
-bash: touch: command not found
-bash-4.1$ nani
-bash: nani: command not found
-bash-4.1$ nano
-bash: nano: command not found
-bash-4.1$ vi
-bash: vi: command not found
-bash-4.1$ echo

-bash-4.1$
-bash-4.1$ open
-bash: open: command not found
-bash-4.1$
-bash-4.1$
-bash-4.1$
-bash-4.1$ clear
-bash: clear: command not found
-bash-4.1$ ls
alice  bin  bob  lib
-bash-4.1$ _
```

Figure 5: Login to Alice's machine with ssh

We cannot non-authorized commands and go outside of the ”root directory”.

4.3 What problem arises when building sophisticated chroot environments? What difficulties do you expect in terms of the administration and the operation of such environments?

The arising problem to build a/some sophisticated chroot(s) is that you have to recreate the whole tree for each chroot, and you have to update the libraries for each of them. In addition, if there are several users, you have to create an individual chroot for each one. Therefore, a large scale infrastructure can quickly become a nightmare to administrate.

4.4 Explain what chroot does and what kinds of security problems can be tackled using this command.

According to Wikipedia, ”Chroot changes the apparent of root directory for the current running process and its children”

Using this command, we can prevent user to harm our system when attending our server through ssh by restricting the commands they can enter, and the processes they can execute. For example, if the `apt` command is removed, the attacker cannot install foreign packages to help him bypass the securities in place and malicious scripts

4.5 Unfortunately, chroot has some security weaknesses. Explain two of these.

One major security flaw is that a custom chroot environment is not as secured as the original one because it needs to stay convenient for the user. Thus, if an attacker get root access, it can ”escapes from the jail”