

Sprint 3 ASIST

Table of contents

Context	2
SSH Access by certificate only and for the administrator only	2
Setting up the connection by certificate	2
Generation of public and private keys for the administrator	2
Configuration of the SSH connection through these keys	3
Modification of the rights on the private key	4
Restart of the SSH Service	4
Disable password login	4
Verification of correct operation.....	4
Conclusion	5
Setting up a SMB public file share.....	6
Installation and configuration of Samba.	6
Configuration of the public share	7
Creation of a user "student1220379" and the group "isepTeams.....	8
Accessing the Share	10
Conclusion	12

Context

We chose the following user cases :

- Number 10: « As a system administrator I want the administrator to have SSH access to the virtual machine, by certificate only, without using a password.»
- Number 11: « As a system administrator I want a public file share, SMB/CIFS or NFS format, to be created to speed up the process among the various teams.»

The operations will be operated on a machine running Debian 11.

SSH Access by certificate only and for the administrator only

Setting up the connection by certificate

In order not to be blocked without any means of access to the SSH service of our server, we need to configure an alternative authentication method before disabling the password authentication. We will therefore configure authentication by certificate, or key pair.

Generation of public and private keys for the administrator

In order to generate a key pair to authenticate the administrator to the SSH service, we must first open a shell with his account.

Once we have a root shell, we can navigate to the `/root/.ssh` directory with `$ cd /root/.ssh` (or create it with `$ mkdir /root/.ssh`).

Then we create the key pair with the command `$ ssh-keygen -b 4096`. The `-b` option is used to specify the length of the key.

We press the Enter key to validate the default location `"/root/.ssh/id_rsa"`, then as we don't want passwords associated with the keys, we press Enter twice.

```

root@debian:~/.ssh# ssh-keygen -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:GLDTaK0Mn7mcY1IUhBpVq5mh0eL6oauCxaJFSfacFE8 root@debian
The key's randomart image is:
+---[RSA 4096]-----+
|  ..+=                |
|  . . 0               |
| oo 0 E              |
| .o @ 0 o            |
| = * 0 o S           |
| o=oB +              |
|  o*oX               |
| +oo+ .              |
| B=.                 |
+---[SHA256]-----+
root@debian:~/.ssh# █

```

Figure 1 - Creation of the private and public keys

Two files are then created:

- id_rsa: This is the private key, to be kept secret and that only the administrator must hold
- id_rsa.pub: This is the public key associated with the private key, It is present on the server in order to authorize SSH access by the private key.

```

root@debian:~/.ssh# ls
id_rsa id_rsa.pub
root@debian:~/.ssh# █

```

Figure 2 - Contents of the /root/.ssh folder

Configuration of the SSH connection through these keys

Once the key pair is created, we need to authorize it in order to open an SSH session with the root user.

To do this, we need to write the contents of the public key to the authorized_keys file with the command `cat id_rsa.pub > authorized_keys`.

```

root@debian:~/.ssh# cat id_rsa.pub > authorized_keys
root@debian:~/.ssh# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDGKns/4000DYwScSk+pit6/balrDtFQ0kQUjcu7LgAGThoF4E
Clcu3XZRZ7QHA/7eLh0chSjqNhPA90sh2mgIIew3Y3Ua0TF27d0rv7ThTdISyoLWp6QLyJk3tp/z0bFtKluwid
/XwgbwzngjCKnvG7GmV9NGdTpehVW2y0o5aQWXIsbByJvVsgkavtQ3XD7rjExbQvCKSNGmfJZfe4kUCKMwbjiay
8QY4uXqv4+iW+SMih7eI10QMZT9uCGRTMik5UeiZ1j/csD0rQD+u1b6VIGLXvc8UsRgCET0SKxbHcmbFG487n/a
E+Tsgiu9XC9+VSt/0vQijxh8Naf15A6RNxstMn+o1CVpfUD2czIjcYoHmym3Iik30obTpWwrv35VcyYkkKEkxLY
rFCDdvcxGpCYBBQ3ZnAJRuxcH0GgQX07hTVnvZ3pwq0wXn+w6tMCmuHnLwMq9/CxedyQyExDQfDR4oqby5qP3sj
YN8Goh9yBS09pamrmvikF3Z94913Q0gEywQMEGVmoxqKJNMWkqNj/0l6dadaCPzRGs3hi8v+r2zTzgIIEDePiCs
q5iuwP0aJnnAfgdt/GB4lIDQpjE/+JIXPxJVjxHoHjw2RB8DrV4SNn0NJdjgVqrBQE4t67jzApLV2DUhwEw5GT+
P8NXAGy3oFqrFcNXI7Gc6Vhvxw== root@debian
root@debian:~/.ssh# █

```

Figure 3 - Contents of the authorized_keys file after copying the contents of the public key into it

Modification of the rights on the private key

In order to use the private key when authenticating to the SSH session, we need to reduce its rights. To do this, we use the command `chmod 600 id_rsa`. This means that only the owner of the file can read and write to it, but he cannot execute it, and other users cannot do anything with it.

Restart of the SSH Service

In order to make all our modifications operational, we just have to restart the SSH service with the command `systemctl restart sshd`.

Disable password login

The connection by certificate being now functional, we only have to disable the connection by password.

To do this, we go to the file and replace **#PasswordAuthentication yes** with **PasswordAuthentication no**.

Verification of correct operation

Test of the connection by certificate to the root account

```
root@localhost:~# ssh 217.160.104.201 -i id_rsa
The authenticity of host '217.160.104.201 (217.160.104.201)' can't be established.
ECDSA key fingerprint is SHA256:Jp/bbyqCaeJkK63zFPtujsXf/shnzNyOtrvfdUf1IHY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '217.160.104.201' (ECDSA) to the list of known hosts.
Last login: Tue Jan  3 16:54:46 2023 from 213.58.234.205
root@localhost:~#
```

Figure 4 - Creation a root SSH Session with certificate

Our previously created certificate is accepted when creating an SSH session to our server's root account from our local machine.

Testing the connection to the root account with password

```
root@localhost:/home/debian# ssh 217.160.104.201
root@217.160.104.201: Permission denied (publickey,keyboard-interactive).
root@localhost:/home/debian#
```

Figure 5 - Our server refuses our attempt to connect to the root account by password

Our server denies SSH login to the root account when we log in with a password and not with a certificate.

Conclusion

Having disabled password login and enabled certificate login for the root account, the router now denies SSH logins from users other than root, and denies SSH logins on the root account by password.

Setting up a SMB public file share

So that each member of the team can exchange files easily and quickly, we will set up an SMB file sharing system with Samba.

Installation and configuration of Samba.

First, we begin to install Samba. We can do this with the command:

```
apt-get install samba
```

```

debian1@debian1:~$ sudo apt-get install samba
[sudo] password for debian1:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  attr libverbs-providers libcephfs2 libgfapi0 libgfrpc0 libgfxdr0 libglusterfs0 libibverbs1 librados2 librdmacm1 liburing1 python3-cffi-backend python3-cryptography python3-dnspython python3-requests-toolbelt python3-yaml samba-samba-vfs-modules tdb-tools
Suggested packages:
  python3-cryptography-doc python3-cryptography-vectors python3-sniffio python3-trio python-markdown-doc python-pygments-doc ttf-bitstream-vera bind9 bind9utils ctdb ldb-tools ntp | chrony samba-doc
The following NEW packages will be installed:
  attr libverbs-providers libcephfs2 libgfapi0 libgfrpc0 libgfxdr0 libglusterfs0 libibverbs1 librados2 librdmacm1 liburing1 python3-cffi-backend python3-cryptography python3-dnspython python3-requests-toolbelt python3-yaml samba-samba-vfs-modules tdb-tools
0 upgraded, 21 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/19.5 MB of archives.
After this operation, 56.5 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figure 6 - Beginning of the installation of Samba

```

Setting up samba (2:4.13.13+dfsg-1~deb11u5) ...
Adding group `sambashare' (GID 127) ...
Done.
Samba is not being run as an AD Domain Controller: Masking samba-ad-dc.service
Please ignore the following error about deb-systemd-helper not finding those services.
(samba-ad-dc.service masked)
Created symlink /etc/systemd/system/multi-user.target.wants/nmbd.service → /lib/systemd/system/nmbd.service.
Failed to preset unit: Unit file /etc/systemd/system/samba-ad-dc.service is masked.
/usr/bin/deb-systemd-helper: error: systemctl preset failed on samba-ad-dc.service: No such file or directory
Created symlink /etc/systemd/system/multi-user.target.wants/smbd.service → /lib/systemd/system/smbd.service.
samba-ad-dc.service is a disabled or a static unit, not starting it.
Setting up libgfxdr0:amd64 (9.2-1) ...
Setting up librdmacm1:amd64 (33.2-1) ...
Setting up librados2 (14.2.21-1) ...
Setting up samba-vfs-modules:amd64 (2:4.13.13+dfsg-1~deb11u5) ...
Setting up libcephfs2 (14.2.21-1) ...
Setting up python3-cryptography (3.3.2-1) ...
Setting up libgfrpc0:amd64 (9.2-1) ...
Setting up libgfapi0:amd64 (9.2-1) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for libc-bin (2.31-13+deb11u5) ...
debian1@debian1:~$
```

Figure 7 - End of the installation of Samba

Then, we ask the system to run the service automatically with the command

```
Systemctl enable smbd
```

```

debian1@debian1:~$ sudo systemctl enable smbd
Synchronizing state of smbd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable smbd
debian1@debian1:~$
```

Figure 8 - Execution of the command that enables Samba

```

debian1@debian1:~$ sudo systemctl status smbd
● smbd.service - Samba SMB Daemon
   Loaded: loaded (/lib/systemd/system/smbd.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2023-01-04 17:51:32 CET; 4min 47s ago
     Docs: man:smbd(8)
           man:samba(7)
           man:smb.conf(5)
    Main PID: 1965 (smbd)
      Status: "smbd: ready to serve connections..."
        Tasks: 4 (limit: 4659)
       Memory: 14.1M
          CPU: 168ms
    CGroup: /system.slice/smbd.service
            └─1965 /usr/sbin/smbd --foreground --no-process-group
              └─1967 /usr/sbin/smbd --foreground --no-process-group
                └─1968 /usr/sbin/smbd --foreground --no-process-group
                  └─1969 /usr/sbin/smbd --foreground --no-process-group

Jan 04 17:51:31 debian1 systemd[1]: Starting Samba SMB Daemon...
Jan 04 17:51:31 debian1 update-apparmor-samba-profile[1959]: grep: /etc/apparmor.d/samba/smbd-shares: No such file or directory
Jan 04 17:51:31 debian1 update-apparmor-samba-profile[1962]: diff: /etc/apparmor.d/samba/smbd-shares: No such file or directory
Jan 04 17:51:32 debian1 systemd[1]: Started Samba SMB Daemon.
debian1@debian1:~$

```

Figure 9 - Status of the Samba service

Configuration of the public share

Now that Samba is installed and running automatically, we need to configure a share in the configuration file. This file is called "smb.conf" and is located in the "/etc/samba/" directory.

We then edit this file and add the following lines:

[isepShare]¹

comment = Shared Folder

path = /srv/share ²

guest ok = no ³

read only = no ⁴

browseable = yes

valid users = @isepTeams⁵

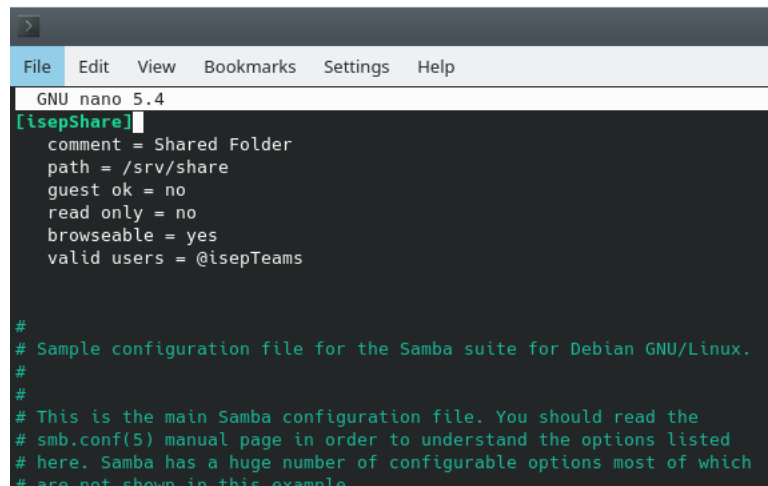
¹ This is the name of the share

² This is the path of the shared folder containing all the shared files

³ We don't allow that anybody can access the share

⁴ We want to be able to modify the files in the share

⁵ All the members belonging the the group « isepTeams » can access and edit the files in the share



```

GNU nano 5.4
[isepShare]
comment = Shared Folder
path = /srv/share
guest ok = no
read only = no
browseable = yes
valid users = @isepTeams

#
# Sample configuration file for the Samba suite for Debian GNU/Linux.
#
#
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
# here. Samba has a huge number of configurable options most of which
# are not shown in this example

```

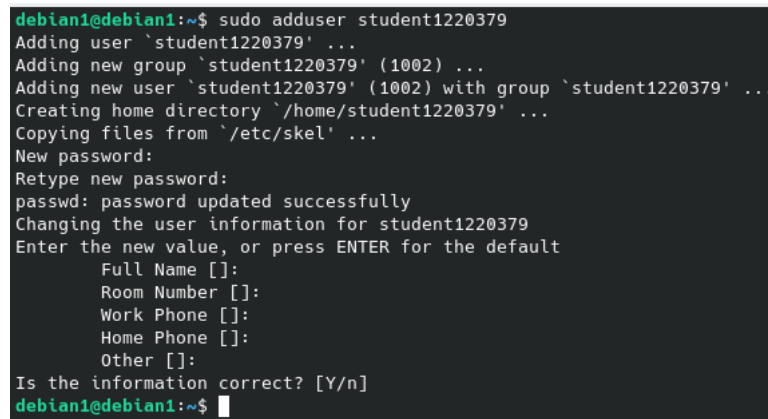
Figure 10 - Content of the smb.conf file

Creation of a user "student1220379" and the group "isepTeams"

To access the samba share, we need to create the group isepTeams and add users to it.

Let's create the user student1220379 with the command

```
sudo adduser student1220379
```



```

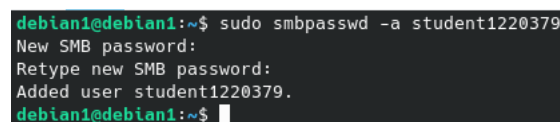
debian1@debian1:~$ sudo adduser student1220379
Adding user `student1220379' ...
Adding new group `student1220379' (1002) ...
Adding new user `student1220379' (1002) with group `student1220379' ...
Creating home directory `/home/student1220379' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for student1220379
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n]
debian1@debian1:~$

```

Figure 11 - Creation of the user student1220379

Then we have to create add the user to the samba users with the command

```
sudo smbpasswd -a student1220379
```



```

debian1@debian1:~$ sudo smbpasswd -a student1220379
New SMB password:
Retype new SMB password:
Added user student1220379.
debian1@debian1:~$

```

Figure 12 - Adding the user student1220379 to the samba users

And we create a group that will contains allowed users to use the share. This with the command `sudo groupadd isepTeams`

```
debian1@debian1:~$ sudo groupadd isepTeams
[sudo] password for debian1:
debian1@debian1:~$
```

Figure 13 - Creation of the *isepTeams* group on the system

Once we have our user and our group, let's add the user to the group *isepTeams* with the command

```
sudo gpasswd -a student1220379 isepTeams
```

```
debian1@debian1:~$ sudo gpasswd -a student1220379 isepTeams
Adding user student1220379 to group isepTeams
debian1@debian1:~$
```

Figure 14 - Adding the user *student1220379* to the group *isepTeams*

As we will need a folder to represent our share, we will create a new folder dedicated to it called « share » and located in the « /srv » directory :

```
sudo mkdir /srv/share
```

```
debian1@debian1:~$ sudo mkdir /srv/share
debian1@debian1:~$
```

Figure 15 - Creation of the share folder in the */srv* directory

Now quite everything is ready, It remains to make the group *isepTeams* owner of the share folder. The *-R* option make this operation recursive (To all subfolders) :

```
sudo chgrp -R isepTeams /srv/share
```

```
debian1@debian1:~$ sudo chgrp -R isepTeams /srv/share
debian1@debian1:~$
```

Figure 16 - Definition of the group *isepTeams* as the owner of the */srv/share* folder recursively

Finally, we allow the owner group to read and write inside the share with the command :

```
sudo chmod -R g+rw /srv/share
```

```
debian1@debian1:~$ sudo chmod -R g+rw /srv/share
debian1@debian1:~$
```

Figure 17 - Allowing the group to read and write into the folder

Then, we just have to restart the samba daemon with the command

```
sudo systemctl restart smbd
```

Accessing the Share

Now that our share is set up, we can access it through any machine in the same network. To reach the share with Windows, we just have to access this address in the navigation bar «\\debian1\isepShare » with debian1 the name of our server in the network and isepShare the name of the share.

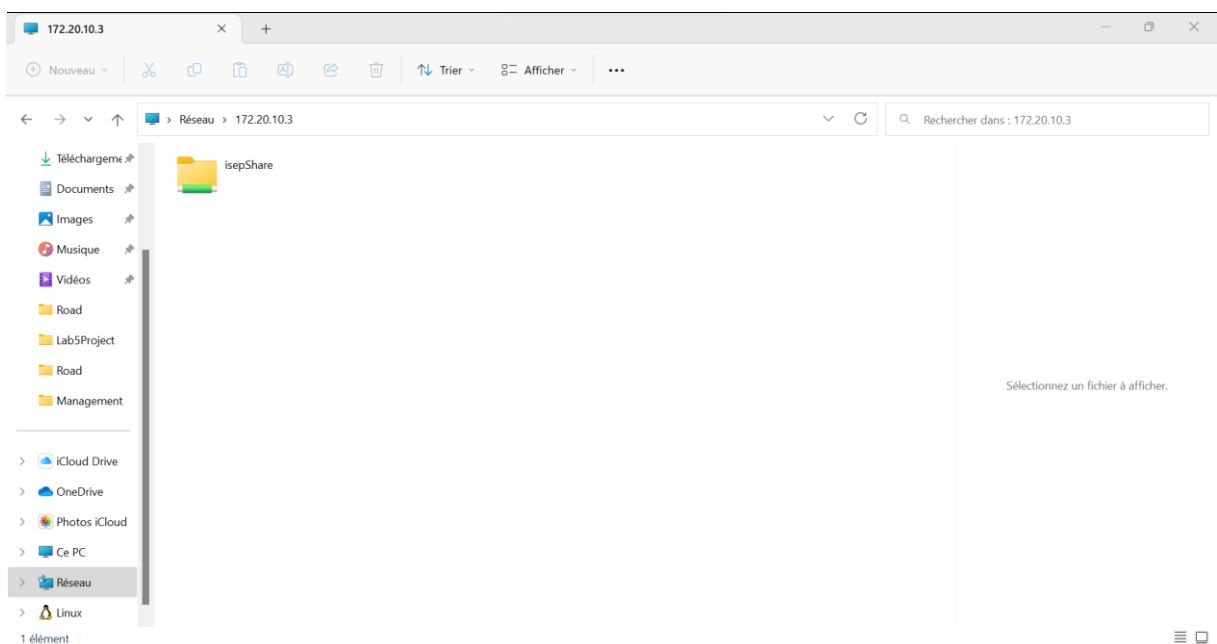


Figure 18 - We can find our share from our local Windows machine

When we try to access the share, a windows prompt us for credentials, we must fill our previously created user and password with (student1220379)

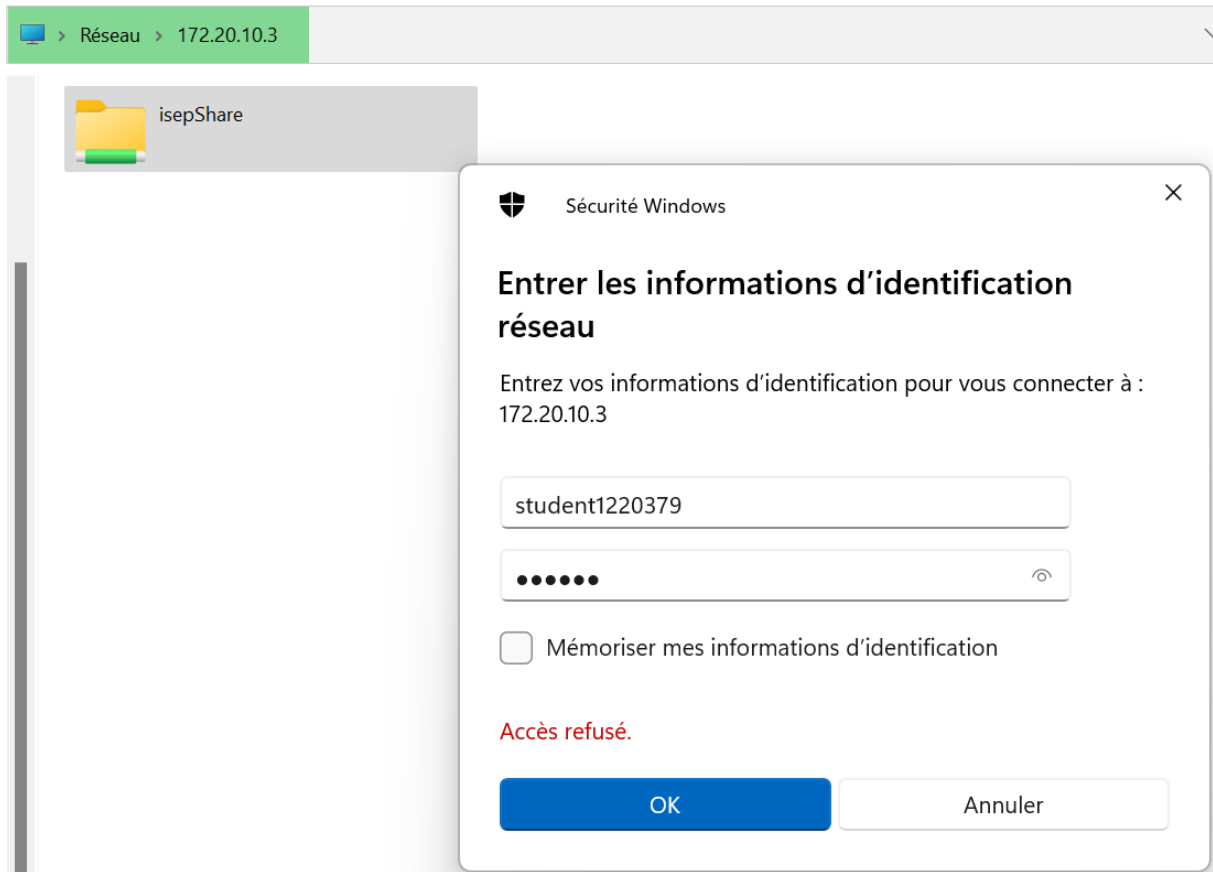


Figure 19 - We log in with the account previously created

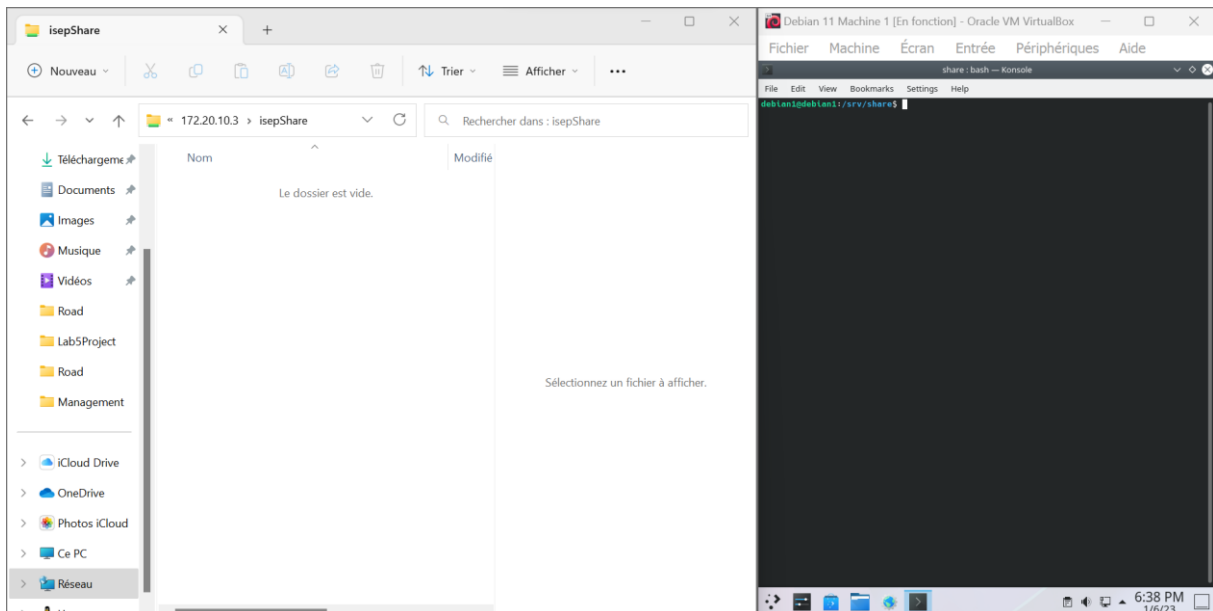


Figure 20 - We manage to enter the Share

We are now in the Share. We can read, add, remove or delete files from here and the modifications will be applied on the share.

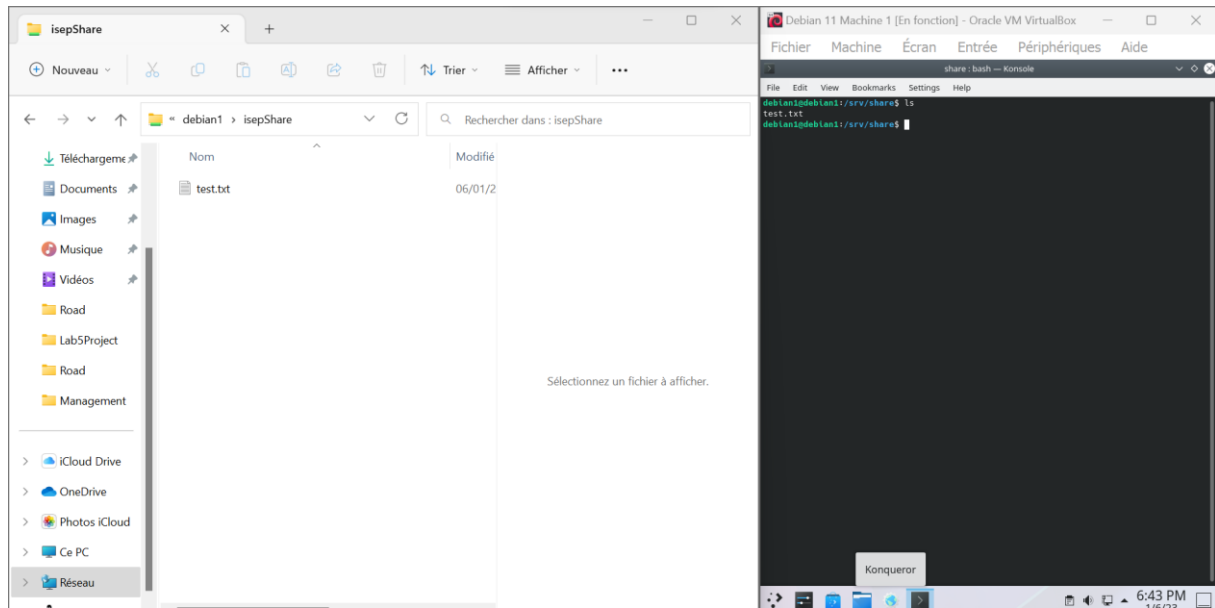


Figure 21 - We can write on the share

Here we created a new file called “test.txt” and when we list the files inside the share directory in the server, we can find this “test.txt” file.

Conclusion

Now that our share is created and well configured, every member of the group “isepTeams” on the server will be able to access it through the same network. We could access it from anywhere if we opened the port 445 with a whitelist (To avoid attacks).