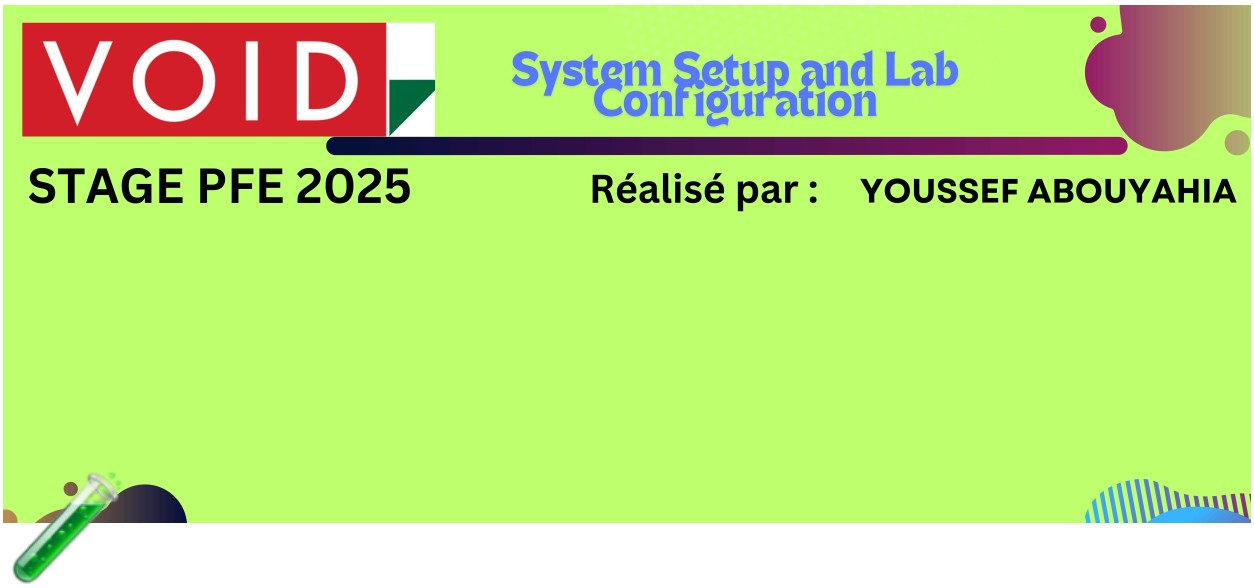


System Setup and Lab Configuration

GETTING STARTED

Enoncé:

- https://gist.github.com/Bahlaouane-Hamza/94eef1209856dc7ee7a2c168b4537327#file-system-setup-and-lab-configuration-md

Quelques ressources à voir:

- https://serversforhackers.com/s/start-here
- https://linuxjourney.com/
- https://www.youtube.com/@ServersforHackers/videos
- https://sourabhbajaj.com/mac-setup/
- https://learngitbranching.js.org/
- https://sadservers.com/

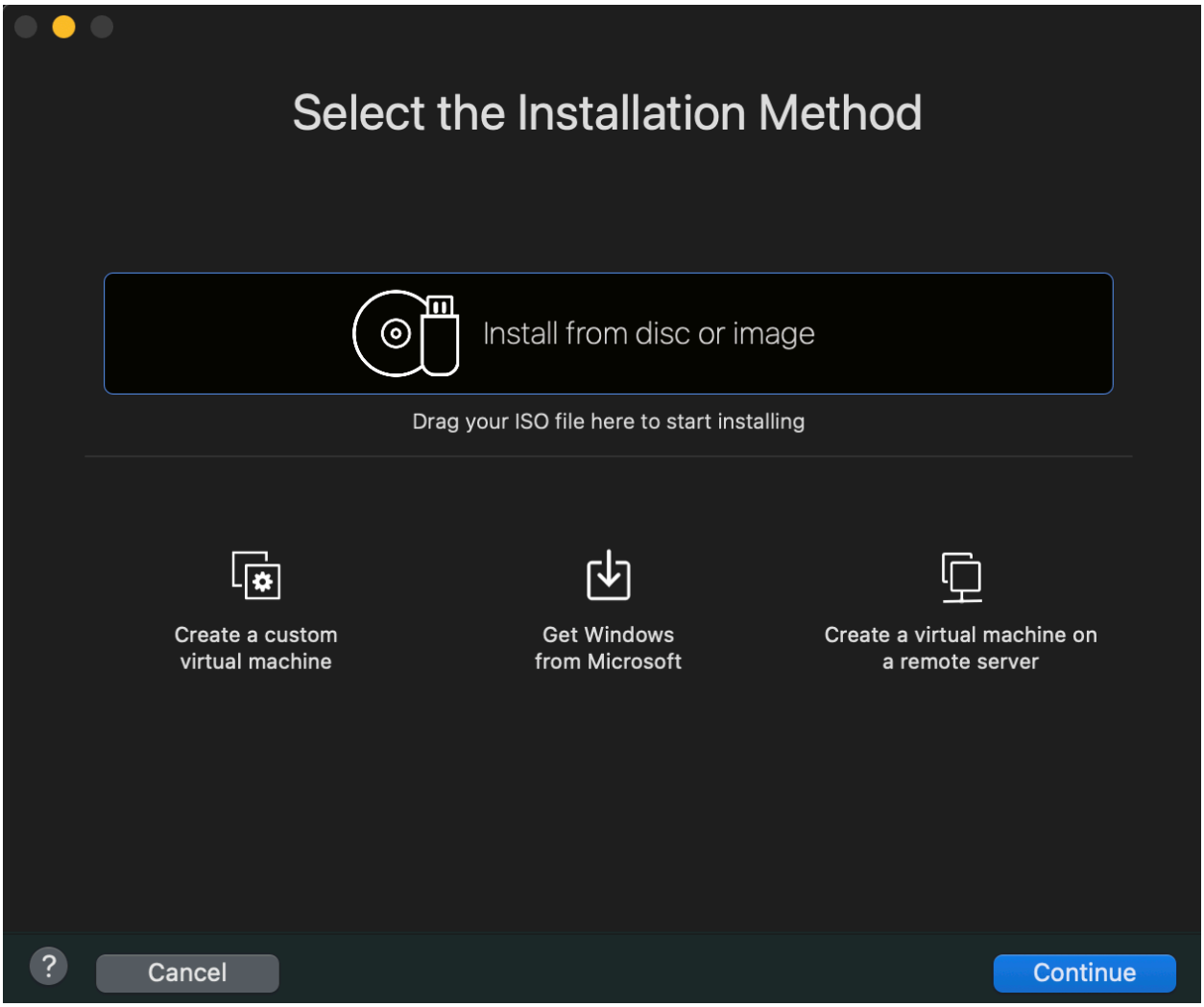
Objectif :

- Environnement de développement entièrement fonctionnel.
- Compréhension des bases de Linux et de Git.
- Familiarité avec les concepts fondamentaux du web.

REPORT

Chapter 1: Setup

- First I had to download the VMware Fusion (created on Broadcom an account to do that)



- Downloaded `vagrant-vmware-utility` **and** `vagrant` **using brew**

```
==> Installing Cask vagrant
==> Running installer for vagrant with sudo; the password may be necessary.
Password:
installer: Package name is Vagrant
installer: Upgrading at base path /
installer: The upgrade was successful.
🍺  vagrant was successfully installed!
~                                                    ✔ 59s 11:15:15
```

```
==> Downloading https://releases.hashicorp.com/vagrant-vmware-utility/1.0.23/vag
######################################################################## 100.0%
==> Installing Cask vagrant-vmware-utility
==> Running installer for vagrant-vmware-utility with sudo; the password may be
Password:
installer: Package name is Vagrant VMware Utility
installer: Installing at base path /
installer: The install was successful.
🍺  vagrant-vmware-utility was successfully installed!
```

- Download the plugin `vagrant-vmware-dekstop`

```
~ vagrant plugin install vagrant-vmware-desktop                ✔ 11:31:58
Installing the 'vagrant-vmware-desktop' plugin. This can take a few minutes...
Fetching vagrant-vmware-desktop-3.0.4.gem
Installed the plugin 'vagrant-vmware-desktop (3.0.4)'!
~                                                    ✔ 25s 11:32:43
```

- Then create a directory where you'll put you'll pull the vagrant file that has the config of the vm to provision then use the command: vagrant-up to create a virtual machine, using the VMware Fusion/Workstation provider to manage the VM.

```
~/Doc/PI/Stag/V/L/S/VM vagrant up --provider vmware_fusion
Bringing machine 'default' up with 'vmware_fusion' provider...
==> default: Box 'bento/debian-11' could not be found. Attempting to find and in
stall...
    default: Box Provider: vmware_desktop, vmware_fusion, vmware_workstation
    default: Box Version: 202407.22.0
==> default: Loading metadata for box 'bento/debian-11'
    default: URL: https://vagrantcloud.com/api/v2/vagrant/bento/debian-11
==> default: Adding box 'bento/debian-11' (v202407.22.0) for provider: vmware_de
sktop (arm64)
    default: Downloading: https://vagrantcloud.com/bento/boxes/debian-11/version
s/202407.22.0/providers/vmware_desktop/arm64/vagrant.box
Progress: 9% (Rate: 18.5M/s, Estimated time remaining: 0:00:38)
```

For me I created directory as follows '`/VOID/Labs/System Setup and Lab Configuration/VM`'

- After doing that I had some problems that I fixed by toggling from share my mac to bridged and I did rerun the command

```
~/Doc/PI/Stag/V/L/S/VM vagrant up --provider vmware_fusion
Bringing machine 'default' up with 'vmware_fusion' provider...
==> default: Verifying vmnet devices are healthy...
==> default: Preparing network adapters...
==> default: Starting the VMware VM...
==> default: Waiting for the VM to receive an address...
==> default: Forwarding ports...
    default: -- 22 => 2222
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
```

Chapter 2: SSH

- Then first thing to do is to ssh into the vm we created

```
~/Doc/PI/Stag/V/L/S/VM vagrant ssh                    INT ✘ 1m 28s 12:02:51
Linux debian-11 5.10.0-31-arm64 #1 SMP Debian 5.10.221-1 (2024-07-14) aarch64

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento

Use of this system is acceptance of the OS vendor EULA and License Agreements.

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
vagrant@debian-11:~$ whoami
vagrant
vagrant@debian-11:~$
```

- In this example I used vagrant ssh but didn't specify the vm name as it's the only one I have but I believe if you have multiple vm you have to specify the machine name like this: `vagrant ssh <machine_name>`

Note: I believe the username Alpha/123456 are incorrect

```
~/Doc/PI/Stag/V/L/S/VM ssh alpha@127.0.0.1 -p 2222          0|1 ✘ 12:00:39
The authenticity of host '[127.0.0.1]:2222 ([127.0.0.1]:2222)' can't be established.
ED25519 key fingerprint is SHA256:h0iyYPHwTPVy4UDx+Nlc5oaoeW04CHxy4S4Y6xv0Nig.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[127.0.0.1]:2222' (ED25519) to the list of known hosts.
alpha@127.0.0.1's password:
Permission denied, please try again.
alpha@127.0.0.1's password:
Permission denied, please try again.
alpha@127.0.0.1's password:
```

- Side Note you can use vagrant ssh-config to generate a valid OpenSSH configuration file for accessing Vagrant virtual machines.

```
~/Doc/PI/Stag/V/L/S/VM  vagrant ssh-config                    255 ✘  11:47:46
Host default
  HostName 127.0.0.1
  User vagrant
  Port 2222
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile "[redacted]
[redacted]
  IdentitiesOnly yes
  LogLevel FATAL
  PubkeyAcceptedKeyTypes +ssh-rsa
  HostKeyAlgorithms +ssh-rsa
```

- Id to return user identity then exit

```
vagrant@debian-11:~$ whoami
vagrant
vagrant@debian-11:~$ id
uid=1000(vagrant) gid=1000(vagrant) groups=1000(vagrant),24(cdrom),25(floppy),27(sudo),
29(audio),30(dip),44(video),46(plugdev),108(netdev)
vagrant@debian-11:~$ exit
logout
~/Doc/PI/Stag/V/L/S/VM                                  ✔  14m 22s  12:17:16
```

- Since I wasn't provided with the correct username/password, I'll got ahead and create another user to log in as let's name it `Joe`

```
vagrant@debian-11:~$ sudo adduser joe
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to a fallback locale ("en_US.UTF-8").
Adding user `joe' ...
Adding new group `joe' (1001) ...
Adding new user `joe' (1001) with group `joe' ...
Creating home directory `/home/joe' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for joe
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]
vagrant@debian-11:~$ ls
vagrant@debian-11:~$ pwd
/home/vagrant
vagrant@debian-11:~$ cd ..
vagrant@debian-11:/home$ ls
joe  vagrant
vagrant@debian-11:/home$ cd joe/
vagrant@debian-11:/home/joe$
```

- Then test it and try to log in as joe

```
~/Doc/PI/Stag/V/L/S/VM ssh joe@127.0.0.1 -p 2222          ✔ 2m 6s 12:28:20
joe@127.0.0.1's password:
Linux debian-11 5.10.0-31-arm64 #1 SMP Debian 5.10.221-1 (2024-07-14) aarch64

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento

Use of this system is acceptance of the OS vendor EULA and License Agreements.

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
joe@debian-11:~$
```

- And add the user joe to the group `vagrant` & `root` and give access to the directory and the file authorized_keys so he can access the `.ssh` directory and add the public key, remember this is just for learning and having fun but it's against best practices to do that!

- Now let's generate our keys (in the below command I added a comment that will help me recognize the keys)

```
~/.ssh ssh-keygen -t rsa -b 4096 -C "Void Lab" -f keys_server_void   1 ✘ 13:07:16
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in keys_server_void
Your public key has been saved in keys_server_void.pub
The key fingerprint is:
SHA256:QAqR73IpJUxiYdJiz2oQ6XMoKEZfghSrzdHDM68mWKg Void Lab
The key's randomart image is:
+---[RSA 4096]----+
|oB*o  .          |
|B=== +           |
|*B*.X .          |
|*Oo*o= .         |
|B B+ .. S        |
|.=o +.           |
|E .+o            |
|   o             |
|                 |
+----[SHA256]-----+
~/.ssh                                          ✔ 4s 13:08:39
```

```
~/.ssh ls                                                    4s 13:08:39
keys_server_void     keys_server_void.pub known_hosts        known_hosts.old
~/.ssh                                                       13:09:45
```

```
~/.ssh cat keys_server_void.pub                              13:09:45
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCk6e6gawtO9wQuF/cN6ySKU7fN44MwiTE9/J6OxwwugpPM6K0U0Orr+xRarmX
8aB4FykOD9nUVpDkJjBOFAeDiFrZAZGbgsLUTUkk752WDX+uIHuFeJJaA2fXR+eLUW1tpEcZACYUtEVuG7cJUfGW87oUQzxBY7f
knn456NfmXFLJxUCLIDk6g1f5FcJXFKzxgDQKZH4PlvwTblrjmGqDVY/Ic70pX0HmbhUGraBLomaI2HrJLN7YkJW9CTY3BMAVmo
EkCqnwASr/30PsaQ0t+8UfcJluB7FQBVA5zJI7Zq6iucmpmtfPPYg7VLaTUW6560rAom1um/r6oV8+mwTHfTql59wA3nNiRRWIn
ZVm3tEs9rLl+MlqEGJtpGqFQwaKaHYnQ0lIAypVUNByXWcWbmxzZM7yTA4A2Uvml8FMuTgQwjaTqkbghsdyrVEwKtbEqLFjHDLz
JDp+/iaqqwg6jJJEQV0JTyQaOka1tshE6oKYZOaHtHzH7oKOxkyylcn0FqK+8nKWi4rNRwEqXKNUuRbW+FgLLOb4QQ+YjQ7t8la
7RsVSZYfZuxorXia3ZIYDI+69VYipHH/jaO1radbmMq/Ko4CUkTAIRv4r7yvGC5060Ote/3v/ZmoqlOnyi+8VFQqtHiRyyLoKvf
I02V0q2TL0D4NpzyvPt3p4LPtx03w== Void Lab  <-- Comment
```

- Now it's time to use scp to send the public key in authorized_key in our server which is our vm in this case!

```
~/.ssh scp -P 2222 keys_server_void.pub joe@127.0.0.1:/home/vagrant/.ssh/   1 ✘ 13:19:55
joe@127.0.0.1's password:
keys_server_void.pub                                  100%  734   955.7KB/s   00:00
~/.ssh                                                       13:20:06
```

- Now it's time to go and check if we have our public key uploaded to the server, then I copied and pasted the key into the file `authorized_keys` files

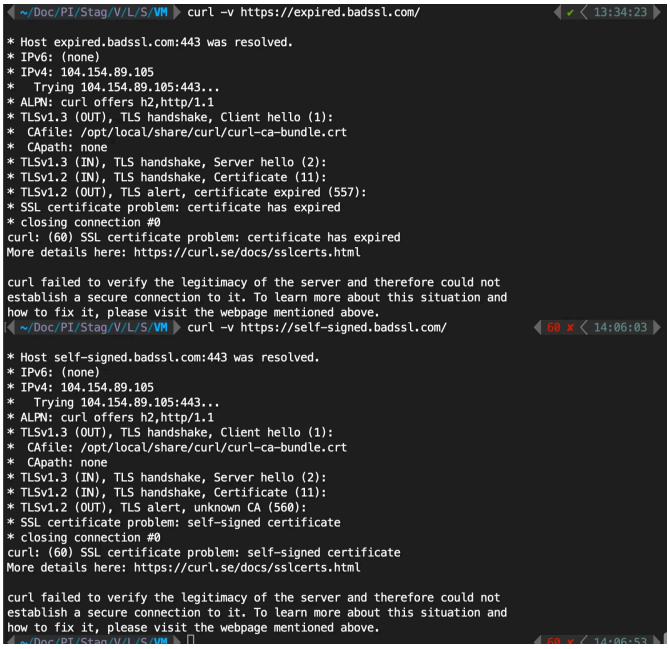
```
joe@debian-11:/home/vagrant/.ssh$ ls
authorized_keys  keys_server_void.pub
joe@debian-11:/home/vagrant/.ssh$ cat authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBtC6H8evaiRjDHvhI6jlaStJGqMlruo6EeEpfjI3btK vagrant
joe@debian-11:/home/vagrant/.ssh$ cat keys_server_void.pub >> authorized_keys
joe@debian-11:/home/vagrant/.ssh$ cat authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBtC6H8evaiRjDHvhI6jlaStJGqMlruo6EeEpfjI3btK vagrant
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCk6e6gawtO9wQuF/cN6ySKU7fN44MwiTE9/J6OxwwugpPM6K0U0Orr+xRarmX
8aB4FykOD9nUVpDkJjBOFAeDiFrZAZGbgsLUTUkk752WDX+uIHuFeJJaA2fXR+eLUW1tpEcZACYUtEVuG7cJUfGW87oUQzxBY7f
knn456NfmXFLJxUCLIDk6g1f5FcJXFKzxgDQKZH4PlvwTblrjmGqDVY/Ic70pX0HmbhUGraBLomaI2HrJLN7YkJW9CTY3BMAVmo
EkCqnwASr/30PsaQ0t+8UfcJluB7FQBVA5zJI7Zq6iucmpmtfPPYg7VLaTUW6560rAom1um/r6oV8+mwTHfTql59wA3nNiRRWIn
ZVm3tEs9rLl+MlqEGJtpGqFQwaKaHYnQ0lIAypVUNByXWcWbmxzZM7yTA4A2Uvml8FMuTgQwjaTqkbghsdyrVEwKtbEqLFjHDLz
JDp+/iaqqwg6jJJEQV0JTyQaOka1tshE6oKYZOaHtHzH7oKOxkyylcn0FqK+8nKWi4rNRwEqXKNUuRbW+FgLLOb4QQ+YjQ7t8la
7RsVSZYfZuxorXia3ZIYDI+69VYipHH/jaO1radbmMq/Ko4CUkTAIRv4r7yvGC5060Ote/3v/ZmoqlOnyi+8VFQqtHiRyyLoKvf
I02V0q2TL0D4NpzyvPt3p4LPtx03w== Void Lab
joe@debian-11:/home/vagrant/.ssh$
```

- Last thing to do is to connect to our server using the private key in our local machine, and yeah!

Chapter 3: SSL

- **badssl.com** is a website designed for testing SSL/TLS configurations in browsers, applications, and systems. It provides a variety of intentionally

misconfigured SSL/TLS scenarios to help developers, testers, and security professionals verify how their software handles different SSL/TLS setups and edge cases.



- For the first example: The website's SSL certificate has expired, so curl refuses to connect because it can't trust that the server is who it claims to be.
- For the second example The website is using a self-signed certificate, which means your computer can't automatically verify that the server is legitimate, hence curl refuses to connect.
- Then we will use openssl to inspect certificated using the following command: `openssl s_client -connect expired.badssl.com:443` . Basically this command attempts to establish an SSL/TLS connection to the server at `expired.badssl.com` on port 443. By doing the following:
  - **Initiate a TLS Handshake:**
  - **Receive Server Certificate:**
  - **Perform Certificate Validation:**

```
~/Doc/PI/Stag/V/L/S/VM openssl s_client -connect expired.badssl.com:443    60 ✘ 14:06:53

Connecting to 104.154.89.105
CONNECTED(00000005)
depth=2 C=GB, ST=Greater Manchester, L=Salford, O=COMODO CA Limited, CN=COMODO RSA Certification Au
thority
verify return:1
depth=1 C=GB, ST=Greater Manchester, L=Salford, O=COMODO CA Limited, CN=COMODO RSA Domain Validatio
n Secure Server CA
verify return:1
depth=0 OU=Domain Control Validated, OU=PositiveSSL Wildcard, CN=*.badssl.com
verify error:num=10:certificate has expired
notAfter=Apr 12 23:59:59 2015 GMT
verify return:1
depth=0 OU=Domain Control Validated, OU=PositiveSSL Wildcard, CN=*.badssl.com
notAfter=Apr 12 23:59:59 2015 GMT
verify return:1
---
Certificate chain
 0 s:OU=Domain Control Validated, OU=PositiveSSL Wildcard, CN=*.badssl.com
   i:C=GB, ST=Greater Manchester, L=Salford, O=COMODO CA Limited, CN=COMODO RSA Domain Validation S
ecure Server CA
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
```

→ So the problem in here is:

The website's security certificate is too old – it's like using an expired ID card. Your computer doesn't trust the website because it can't prove it's really who it says it is.

How to Resolve (If You Owned the Website):

1. **Renew the Certificate:** Pay a Certificate Authority (like Let's Encrypt) to issue a new, valid SSL certificate for your website.
2. **Install the New Certificate:** Follow the instructions from your web hosting provider or server administrator to install the new certificate on your web server.

Chapter 4: Cronjob

▼ **What is a cronjob?**

A cron job is like a little computer alarm clock that you can set to run these tasks automatically, based on a schedule you define. It's a way to tell your computer, "Hey, run this command at this specific time, every day/week/month.

Now let’s go ahead and create some dummy files on `/tmp` for example (I used bash to automate the process)

```
joe@debian-11:~$ nano script.sh
joe@debian-11:~$ ./script.sh
Created files: /tmp/file1 through /tmp/file10
joe@debian-11:~$ cd /tmp/
joe@debian-11:/tmp$ ls
file1   file5  systemd-private-b1d10d57d8ec4b6eb3ccd268125c2a04-systemd-logind.service-hREFtg
file10  file6  systemd-private-b1d10d57d8ec4b6eb3ccd268125c2a04-systemd-timesyncd.service-gcSEWg
file2   file7  vagrant-shell
file3   file8  vmware-root_525-4281712298
file4   file9
joe@debian-11:/tmp$
```

We used the following script:

```
#!/bin/bash

# Loop from 1 to 10
for i in $(seq 1 10); do
  # Create the file
  touch /tmp/file$i
done

echo "Created files: /tmp/file1 through /tmp/file10"
exit 0
```

- Now let's write the script `clean_temp.sh`

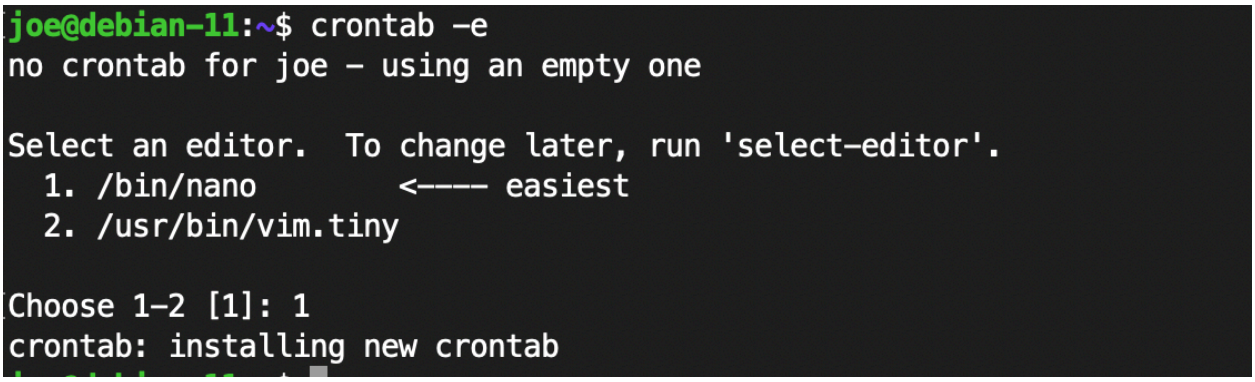
This script finds all regular files inside your `~/temp` directory that haven't been modified in more than 7 days and then *permanently deletes* them.

```
joe@debian-11:~$ nano clean_temp.sh
joe@debian-11:~$ chmod +x clean_temp.sh
joe@debian-11:~$ ls
clean_temp.sh  script.sh
joe@debian-11:~$ cat clean_temp.sh
#!/bin/bash
find ~/temp -type f -mtime +7 -delete
joe@debian-11:~$
```

→ To understand how this cronjob work and it's gonna get triggered let me create a file with a modification date more than 7 days, as you can see below:

This is basically the format: **[[CC]YY]MMDDhhmm[.ss]**

- Now when we type `cronjob -e` then you add the path to your cronjob script and the specify the cron schedule (explained below)
  - `0 0 * * *`: This is the cron schedule expression.
    - 0: Minute (0 means the top of the hour, or minute 0)
    - 0: Hour (0 means midnight)
    - : Day of the month (all days)
    - : Month (all months)
    - : Day of the week (all days of the week)



Here the cronjob:

```
  GNU nano 5.4                 /tmp/crontab.RsgbVT/crontab *
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
6 10 * * * /home/vagrant/clean_temp.sh
```

Note: You could see the path has changed since I switched back to the work machine so I had to redo al the previous steps in the work machine, hence why I path changed as for the expression of the cronjob I changed the triggered time to be triggered as soon as I save the file.

After the script is run we can see the file `testfile` has been deleted

```
vagrant@debian-11:/tmp$ ls
file1
file10
file2
file3
file4
file5
file6
file7
file8
file9
systemd-private-1d461b56bb8b4045a3f77a5256066f02-apache2.service-fGDZMf
systemd-private-1d461b56bb8b4045a3f77a5256066f02-systemd-logind.service-kjGlMg
systemd-private-1d461b56bb8b4045a3f77a5256066f02-systemd-timesyncd.service-pzeHm
f
vmware-root_507-2083928996
vagrant@debian-11:/tmp$
```

tmp directory after running the script!

As if you want to Schedule the script to run daily at midnight.

- Basically you use: `0 0 * * *`

→ What happens if the script is not executable?

Basically if the script is not executable t**he Cron Job Fails Silently (Usually):** By default, cron is designed to run in the background without direct user interaction. This means you typically *won't* see an error message pop up on your screen when the cron job fails due to a permission problem. Cron *might* send an email to the user who owns the cron job, but that depends on how cron is configured on your system.

- To check if the cronjob is running (I did run it twice changed the time the script will be run 2 times and it's reflected in the logs)

```
vagrant@debian-11:/var/log$ sudo grep CRON /var/log/syslog
Mar  4 16:15:30 debian-11 cron[502]: (CRON) INFO (pidfile fd = 3)
Mar  4 16:15:30 debian-11 cron[502]: (CRON) INFO (Running @reboot jobs)
Mar  4 16:17:01 debian-11 CRON[12101]: (root) CMD (   cd / && run-parts --report
 /etc/cron.hourly)
Mar  5 09:49:27 debian-11 cron[514]: (CRON) INFO (pidfile fd = 3)
Mar  5 09:49:27 debian-11 cron[514]: (CRON) INFO (Running @reboot jobs)
Mar  5 10:08:01 debian-11 CRON[1142]: (vagrant) CMD (/home/vagrant/clean_temp.sh
 )
Mar  5 10:08:01 debian-11 CRON[1141]: (CRON) info (No MTA installed, discarding
output)
Mar  5 10:13:01 debian-11 CRON[1175]: (vagrant) CMD (/home/vagrant/clean_temp.sh
 )
Mar  5 10:13:01 debian-11 CRON[1174]: (CRON) info (No MTA installed, discarding
output)
vagrant@debian-11:/var/log$
```

Or we use the command `joournalctl` and what is that command?

→ `journalctl` is a command-line tool for querying and displaying logs collected by systemd, the system and service manager used by many Linux distributions. Unlike traditional log files (like /var/log/syslog), journalctl accesses a binary log format, offering more structured and powerful filtering capabilities.

How can you add logging to your script to capture errors ?

I added on the basic script to become something like this:

```
#!/bin/bash

# Script to delete files in ~/temp older than 7 days
```

```
# Configuration
LOG_FILE="$HOME/temp_cleanup.log" #Logfile will be in user's home
TEMP_DIR="/tmp"
DAYS=7
SCRIPT_NAME=$(basename "$0")
TIMESTAMP=$(date +%Y-%m-%d_%H-%M-%S)


# Function to log messages
log() {
  echo "$TIMESTAMP: $SCRIPT_NAME: $1" >> "$LOG_FILE"
}

# Function to log errors and exit
error_exit() {
  log "ERROR: $1"
  exit 1
}

# Check if the temp directory exists
if [ ! -d "$TEMP_DIR" ]; then
  error_message="Error: Temp directory '$TEMP_DIR' does not exist."
  error_exit "$error_message"
fi

log "Script started - Deleting files in '$TEMP_DIR' older than $DAYS days."

# Find and delete files
find "$TEMP_DIR" -type f -mtime +"$DAYS" -delete

# Check the exit code of the find command
if [ $? -ne 0 ]; then
  error_message="Error: find command failed with exit code $?.  Check permissio
  error_exit "$error_message"
fi
```

```
log "Script completed successfully."
exit 0
```

→ Basically checking if the /tmp exists and if the files can be found and deleted otherwise we write in the home directory of our user in a file named `temp_cleanup.log`

- Now let's put our script to the test

```
vagrant@debian-11:~$ ./clean_temp.sh
find: '/tmp/systemd-private-1d461b56bb8b4045a3f77a5256066f02-systemd-logind.serv
ice-kjGlMg': Permission denied
find: '/tmp/vmware-root_507-2083928996': Permission denied
find: '/tmp/systemd-private-1d461b56bb8b4045a3f77a5256066f02-apache2.service-fGD
ZMf': Permission denied
find: '/tmp/systemd-private-1d461b56bb8b4045a3f77a5256066f02-systemd-timesyncd.s
ervice-pzeHmf': Permission denied
vagrant@debian-11:~$ ls
clean_temp.sh  temp_cleanup.log
vagrant@debian-11:~$ cat temp_cleanup.log
2025-03-05_10-40-44: clean_temp.sh: Script started - Deleting files in '/tmp' ol
der than 7 days.
2025-03-05_10-40-44: clean_temp.sh: ERROR: Error: find command failed with exit
code 0.  Check permissions in temp directory.
vagrant@debian-11:~$
```

Chapter 5: Permissions

Fist thing first let me check the existence of the user `alpha` by searching for the user in the `/etc/group`

```
vagrant@debian-11:~$ cat /etc/group | cut -d : -f 1 | grep al
dialout
systemd-journal
alpha
vagrant@debian-11:~$
```

Now we exit and connect with the user alpha then we try to run the command and see what happens.

```
alpha@debian-11:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.65.128  netmask 255.255.255.0  broadcast 192.168.65.255
        inet6 fe80::20c:29ff:fe70:f1fd  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:70:f1:fd  txqueuelen 1000  (Ethernet)
        RX packets 10570  bytes 2107752 (2.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5522  bytes 585241 (571.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 18  memory 0xfd4a0000-fd4c0000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

alpha@debian-11:~$
```

We can play more and access the sudoers file using the command: `sudo visudo` on the vagrant user, but in my case it wasn't needed, if you needed access you can add this line to the sudoers file: `alpha ALL = NOPASSWD: /sbin/ifconfig` (basically granting to alpha the right to use sudo to run the command ifconfig)

→ Now it's time to take a look at file permissions for our user `alpha` here

```
alpha@debian-11:~$ curl -o /opt/bat.zip https://github.com/sharkdp/bat/archive/r
efs/tags/v0.24.0.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
Warning: Failed to create the file /opt/bat.zip: Permission denied
alpha@debian-11:~$
```

As you noticed we don't have the permission to do that

‼️ What's considered a solution but it's dangerous for your protection and security please don't opt for this solution:

- **Do NOT Change Permissions on /opt/:** Resist the temptation to change the permissions on the /opt/ directory. It's a system directory and should be protected.

What I opted to do here is to curl the file and put it in the /tmp directory then connect to user vagrant and place the downloaded file in the `/opt` directory

```
alpha@debian-11:~$ curl -o /tmp/bat.zip https://github.com/sharkdp/bat/archive/r
efs/tags/v0.24.0.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
alpha@debian-11:~$ cd /tmp/
alpha@debian-11:/tmp$ ls
bat.zip
file1
file10
file2
file3
file4
file5
file6
file7
file8
file9
systemd-private-1d461b56bb8b4045a3f77a5256066f02-apache2.service-fGDZMf
systemd-private-1d461b56bb8b4045a3f77a5256066f02-systemd-logind.service-kjGlMg
systemd-private-1d461b56bb8b4045a3f77a5256066f02-systemd-timesyncd.service-pzeHm
f
vmware-root_507-2083928996
alpha@debian-11:/tmp$
```

```
vagrant@debian-11:/tmp$ mv bat.zip /opt/
mv: cannot move 'bat.zip' to '/opt/bat.zip': Operation not permitted
vagrant@debian-11:/tmp$ sudo mv bat.zip /opt/
vagrant@debian-11:/tmp$ cd /opt/
vagrant@debian-11:/opt$ ls | grep bat.zip
bat.zip
vagrant@debian-11:/opt$
```

The exercice probably wanted me to use the `chmod` to understand how to use it, but it's just dangerous and I'm familiar with the intricacies of the command so no need!

→ Now let's unzip the file (Before hand I had to download unzip)

```
vagrant@debian-11:/opt$ unzip -t bat.zip
Archive:  bat.zip
  End-of-central-directory signature not found.  Either this file is not
  a zipfile, or it constitutes one disk of a multi-part archive.  In the
  latter case the central directory and zipfile comment will be found on
  the last disk(s) of this archive.
unzip:  cannot find zipfile directory in one of bat.zip or
        bat.zip.zip, and cannot find bat.zip.ZIP, period.
vagrant@debian-11:/opt$ file bat.zip
bat.zip: empty
```

Chapter 5: Webserver

Here we'll be using a web server by installing an apache server on our VM

```
vagrant@debian-11:/opt$ sudo su -
root@debian-11:~# sudo apt-get install -y apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.62-1~deb11u2).
0 upgraded, 0 newly installed, 0 to remove and 60 not upgraded.
root@debian-11:~# sudo systemctl start apache2
root@debian-11:~#
```

→ Then we head back to our browser and type in the ip of the machine in our case it's a local one just for testing so it's the local ip `127.0.0.1`

It works!

"Yep, that's all for this lab, hope to see you soon in another lab!" Youssef ABOUYAHIA