Task from the checklist	The solution and how to implement it	How to check the solution
Checks		
We will check in this section that the instructions have been respected. If only one of these points is wrong you must end the evaluation and give a score of 0. The VM runs well on a `Linux` OS. Technos such as `Traefik` as well as that `Docker`/`Vagrant`/etc. containers are not used in this project		# Check OS `lsb_release -a` # Run the following commands in order to make sure there are no Docker or other software that is prohibited to use in this project: `dpkg --get-selections | grep -iE "Docker|Vagrant|Traefik" || echo "I have not found any restricted packages"` # or `dpkg-query -l` # or `apt list --installed`
Install and Update		
The size of the VM disk is 8 GB There is at least one 4.2 GB partition		`apt install parted -y` `sudo parted -s /dev/sda unit GB print`
From the shell of the VM, run the command that lets you know if the OS and packages are up to date. If you discover that the OS or packages are not up to date, this test has failed.	`apt install vim -y` `apt install sudo -y` `sudo apt update -y && sudo apt upgrade -y`	# Run the commands: `sudo apt update -y && sudo apt upgrade -y` # The printed out result should say that no new packages were updated and upgraded.
From the shell of the VM, run the command that allows to know which packages are installed. If you discover that docker/vagrant/traefik type packages are installed, this test has failed.		# Run the following commands in order to check there are no Docker or other non-relevant packages: `dpkg --get-selections | grep -iE "Docker|Vagrant|Traefik" || echo "I have not found any restricted package"` # or: `dpkg-query -l` # or: `apt list --installed`
Network and Security		
Ask the evaluated person to create a user with SSH key to be able to connect to the VM. He must be part of the sudo group. If it's not in this case, this test is failed.	# If you don't have a user account on you VM then create a user account '`user`' and add it to the group '`sudo`' in the following manner: `adduser user -ingroup sudo` # If there is already a user account then run the following command to add it to the group '`sudo`': `su` *`<enter super user (root) account with root's password>`*	# Create a new user: `adduser newuser -ingroup sudo` Check that the user '`newuser`' is a member of the group '`sudo`': `groups newuser` # or: `id -Gn newuser`

	`usermod -aG sudo user` # Then reboot the VM: `sudo reboot`	# Check that the user '`user`' is allowed to connect to the VM via SSH: `ssh newuser@localhost`
If this user executes the sudo command he must be able to use commands that require special rights. If this is not the case, this test is failed.	# If you don't want to enter the `user`'s password every time you run a command with `sudo` then edit the file '`/etc/sudoers`' (open it with `sudo`). First switch to the `root` account: `su` *`<enter super user (root) account with root's password>`* # Open it with `visudo`: `sudo visudo /etc/sudoers` # Edit the next line after the line '`# Allow members of group sudo to execute any command`' so it would be as follows: `%sudo ALL=(ALL:ALL) NOPASSWD:ALL` # Save the file and quit.	# Being logged in as '`newuser`' run these commands: `whoami` `sudo whoami` # The output of the first commands should be '`newuser`'. The output of the second command should be '`root`'. # Check that there is no password prompt when you use `sudo` being a '`newuser`': `sudo ls /root` # You can then delete the user '`newuser`' using the script 02 from the project 'init'.
Check that the DHCP service of the VM is deactivated. If not, this test is failed.	# Open the file '`/etc/network/interfaces`' and comment the line '`iface enp0s3 inet dhcp`'. Add the following to the file: `auto enp0s3` `iface enp0s3 inet static` `address 10.0.2.1` `netmask 255.255.255.252` `gateway 10.0.2.2` # Save the file then run: `sudo ifup enp0s3` # Open '`/etc/sysctl.conf`' in order to disable IPv6 on all the interfaces and make this setting as a default one and add the following at the bottom of the file: `net.ipv6.conf.all.disable_ipv6=1` `net.ipv6.conf.default.disable_ipv6=1` `net.ipv6.conf.lo.disable_ipv6=1` `net.ipv6.conf.enp0s3.disable_ipv6=1`	# Check the settings of the primary network interface: `ip a` # Run the following commands in order to check the gateway address: `sudo apt install net-tools -y` `netstat -rn`
Choose a different netmask than /30, ask the evaluated person to configure a network connection with this netmask on the host and guest side. The evaluated person will choose the IPs. If it is not successful, this test is failed.		# You might need the help of an IP calc. Consider using this one - http://jodies.de/ipcalc # Open the file '`/etc/network/interfaces`', comment the lines '`address 10.0.2.1`' and '`netmask 255.255.255.252`' and add two new ones, '`address 10.0.3.100`' and '`netmask 255.255.252.0`', instead. Then run the following command and make sure the new address and netmask have been applied: `sudo ifup enp0s3`
From a shell on the VM, check that the port of the SSH has been successfully been modified. SSH access MUST be done with publickeys. The root user should not be able to connect in SSH. If this is not the case, this test is failed.	# The following command, which should be run on your mac, generates two files ('`id_rsa`' is a private key and should be securely kept, can be encrypted with a passphrase; '`id_rsa.pub`' is a public key, should be copied and stored on the VM): `ssh-keygen -t rsa` # Send the public key to the VM using the command (it will be added to	# Check out the SSH daemon configuration file '`/etc/ssh/sshd_config`' or simply run the following commands in order to check that the default SSH port of the VM has been successfully changed: `sudo grep ^Port /etc/ssh/sshd_config`

	'`~/.ssh/authorized_keys`' on the VM): `ssh-copy-id -i ~/.ssh/id_rsa.pub user@<you mac's IP address> -p <mac port that forwards to the VM's SSH port>` # In order to change SSH default port of the VM open '`/etc/ssh/sshd_config`' and comment the line 13 ('`Port 22`') if it is not yet, then add a new rule '`Port <desired port>`' on a new line. It is recommended to use ports from 50000 and higher. # Comment the line 33 ('`PermitRootLogin prohibit-password`') if it is not yet and add a new one '`PermitRootLogin no`' in order to disable root login via SSH. # Comment the line 58 ('`PasswordAuthentication yes`') if it is not yet and add a new one '`PasswordAuthentication no`' in order to disable SSH login with a password so as to connect to the VM via SSH with RSA keys only. # Restart SSH daemon on the VM: `sudo service sshd restart`	# Try to connect to the VM via SSH as '`root`'. Then try to connect to the VM via SSH as '`user`' without the RSA key stored on your mac. Both the attempts should not be successful.
From a shell on the VM, run the command that lists all firewall rules. If no rules are in place or that it is not sufficient in relation to the request from the subject, then this test is failed.	# More about Linux firewall '`ufw`' – http://blog.sedicomm.com/2018/07/06/kak-nastroit-brandmauer-ufw-na-ubuntu-i-debian/ Russian. # First install '`ufw`' - program for managing a netfilter firewall on Linux: `sudo apt install ufw -y` # Show processes that listen ports: `sudo lsof -i` `netstat -tulpn` # Open file '`/etc/default/ufw`', comment the line 7 and add a new one: `IPV6=no` # Enable firewall and create new rules for SSH, HTTP & HTTPS running the following commands: `sudo ufw enable` `sudo ufw default deny incoming` `sudo ufw default allow outgoing` `sudo ufw allow 22222/tcp` `sudo ufw allow 80/tcp` `sudo ufw allow 443/tcp` `sudo ufw logging low` `sudo ufw reload` `sudo ufw status verbose`	# Check the VM's firewall rules: `sudo ufw status verbose`
From a shell on your computer, run the command that allows you to to test a DOS (`Slowloris` or other). Check that everything is still working. In addition, make sure that a `fail2Ban` service (or similar service) is installed on the VM. If this is not the case, this test is failed.	# Install `nginx` web server: `sudo apt install nginx -y` `sudo service nginx status` # Edit `nginx` configuration file and increase the number of simultaneous	# The first way to implement a DOS attack is to run '`Slowloris`' (DOS attack tool) on your mac (not to the VM) so as to use it against the VM. Download it: `wget https://web.archive.org/web/20090620230243/http://ha.c`

connections:
```
sudo vim /etc/nginx/nginx.conf
```
Comment the line 7 '`worker_connections 768;`', add a new one below '`worker_connections 20000;`' and then restart the web server:
```
sudo service nginx restart

```
Install `fail2ban`:
```
sudo apt install fail2ban -y
sudo service fail2ban status
sudo service fail2ban stop

```
Create a new file '`/etc/fail2ban/jail.local`' and add to it:
```
[nginx-dos]
# Any IP requesting more than 240 pages in 60 seconds,
# or 4 p/s average, is suspicious. Block it for two
full days.
enabled = true
port = http
filter = nginx-dos
logpath = /var/log/nginx/access.log
findtime = 60
bantime = 172800
maxretry = 240

```
Create a file '`/etc/fail2ban/filter.d/nginx-dos.conf`' and add to it:
```
[Definition]
# Option: failregex
# Notes.: Regexp to catch a generic call from an IP
address.
# Values: TEXT
failregex = ^<HOST> -.*"(GET|POST).*HTTP.*"$

# Option: ignoreregex
# Notes.: regex to ignore. If this regex matches, the
line is ignored.
# Values: TEXT
#
ignoreregex =

```
Then restart `fail2ban`:
```
sudo service fail2ban restart
sudo fail2ban-client status
sudo fail2ban-client status nginx-dos
```

```
kers.org/slowloris/slowloris.pl
```
Open its source code and comment the line 4 if necessary (it is needed if you have not setup SSL for your VM's web server yet).

Check the status of the `fail2ban` profile '`nginx-dos`':
```
sudo fail2ban-client status nginx-dos

```
Send a request to the VM's web server – open a browser on your mac and enter your VM's IP address. Make sure that the VM's web server is accessible on your mac and you can access its default web page through the mac's browser. Then run '`slowloris.pl`' from your mac in the following way:
```
chmod +x slowloris.pl
perl slowloris.pl -dns <host computer IP>:<mac port
that forwards to the VM's HTTP port>
```
Or:
```
perl slowloris.pl -dns <host computer IP>:<mac port
that forwards to the VM's HTTPS port>
```
After that VM's web server would be attacked. `fail2ban` should read `nginx`'s logs, find there a lot of sudden requests and ban the IP address of the client which sends so many requests to the web server therefore you won't be able to see the VM web server's web page from the computer you have run `Slowloris` but the web page should be accessible when requesting it from other computers. See the list of the IP addresses that are banned:
```
sudo fail2ban-client status nginx-dos

```
You can then unban some IP addresses so as to repeat the attack:
```
sudo fail2ban-client set nginx-dos unbanip <banned IP
address>

```
The second way to implement a DOS attack is to use Apache HTTP server benchmarking tool that is already installed on your mac. For the first make sure that `fail2ban` list of the banned IP address is empty:
```
sudo fail2ban-client status nginx-dos
```
Run Apache HTTP server benchmarking tool in the following way:
```
ab -n 10000 -c 50 http://<mac's IP address>:<mac port
that forwards to the VM's HTTP port>
```
After that VM's web server would be attacked. `fail2ban` should read `nginx`'s logs, find there a lot of sudden requests and ban the IP address of the client which sends so many requests to the web server therefore you won't be able to see the VM web server's web page from the computer you have run `Slowloris` but the web page should be accessible when requesting it from other computers. See the list of the IP addresses that are banned:
```
sudo fail2ban-client status nginx-dos

```
You can then unban some IP addresses so as to repeat the attack:
```
sudo fail2ban-client set nginx-dos unbanip <banned IP
address>
```

		# Now let's make sure `fail2ban` can stop SSH login brute-forcing. Note that `fail2ban` by default won't work when: 1) an attacker have a correct RSA private key but enters a wrong passphrase. 2) an attacker does not have a correct RSA private key but tries to log in as an existing user (a user that exists on the VM). By default `fail2ban` should ban when an attacker tries to log in as a user that does not exist on the VM 5 times in a row. You can try such kind of attacks and then check `fail2ban` status: `sudo fail2ban-client status ssh` # The output of that command should consist of a list of IP addresses that have been banned. You can then unban some IP addresses so as to repeat the attack: `sudo fail2ban-client set sshd unbanip <banned IP address>`
From a shell on the VM, run the command that lists the open ports. Check that the open ports correspond to the subject's request. If not, this test is failed. You have to set a protection against scans on your VM's open ports (from Subject).	# Install `PortSentry`, an attack detection tool: `sudo apt install portsentry -y` # Open the file '`sudo vim /etc/default/portsentry`', comment the lines 9 and 10 ('`TCP_MODE="atcp"`', '`UDP_MODE="audp"`') and add two ones below them: `TCP_MODE="atcp"` `UDP_MODE="audp"` # Open the file '`/etc/portsentry/portsentry.conf`', comment the lines 35 and 36 (`BLOCK_UDP="0", BLOCK_TCP="0"`) and add two ones below them: `BLOCK_UDP="1"` `BLOCK_TCP="1"` # Restart `PortSentry` and see its status: `sudo systemctl restart portsentry` `sudo systemctl status portsentry`	# See the processes that listen ports: `sudo lsof -i` # or: `netstat -tulpn` # See the list the open ports `sudo apt install nmap -y` # or: `nmap --open localhost` # Make sure that the files (all these files are in the directory '`/var/lib/portsentry`') that contain lists of banned IP addresses are empty (clear them if they are not). Scan the ports in the following way (you have to scan the ports from the different machine than your VM): `nmap <mac's IP address> -p T:<forwarded ports to scan(ex:8080,22000,44300)>` # Or just some of them: `nmap <VM's IP address> -p T:<ports to scan (ex:22,80,443)>` # Then check whether `PortSentry` has managed to ban the IP address of the machine you used to scan the ports: `cd /var/lib/portsentry` `sudo cat *` # If `PortSentry` has done it properly you'll see the IP address of the machine that you used to scan the ports. Clear the bottom of the file '`/etc/hosts.deny`' and clear all the files in the directory '`/var/lib/portsentry`' if you want to unban the banned IP addresses.
Check if the active services of the machine are only those necessary for its proper functioning. If not, this test has failed.	# Print out at the active services: `sudo service --status-all` `systemctl list-unit-files --type service --state enabled` # You'll see the list of active and enabled services run on the VM. You	# Print out at the active services: `sudo service --status-all` `systemctl list-unit-files --type service --state enabled`

	may leave only the following ones: `cron`, `dbus`, `fail2ban`, `kmod`, `networking`, `nginx`, `portsentry`, `procps`, `rsyslog`, `ssh`, `udev`, `ufw`, `systemd-timesyncd`, `syslog`, `getty`, `autovt`, `console-setup`. Disable and remove from autoload the other services with the commands (both should be run for each particular service): `sudo systemctl stop <service>` `sudo systemctl disable <service>`	
Check that there is a script to update all sources of package, packages, which log into the right file and that it is in cron. If this is not the case, this test is failed.	# Create a new file '`/var/log/update_script.log`' and edit its permissions: `sudo chmod 666 /var/log/update_script.log` # Create a new script '`/etc/cron.d/update_packages.sh`' and edit its permissions: `sudo chmod +x /etc/cron.d/update_packages.sh` # Add the following to the file '`/etc/cron.d/update_packages.sh`': `#!/bin/sh` `LOG="/var/log/update_script.log"` `sudo apt update -y | tail -n1 > $LOG` `sudo apt upgrade -y | tail -n1 >> $LOG` `DATE=`date +%d.%m.%Y`` `TIME=`date +%T`` `team="R4Y7r4c1N9_C4P031R4"` `echo "\033[0;33mLast update: $DATE, $TIME \033[m"` `echo "created by\033[0;31m $team \33[m"` `exit` # Edit the file '`/etc/crontab`' and add these lines of text to it: `@reboot root /etc/cron.d/update_packages.sh` `0 4 * * 6 root /etc/cron.d/update_packages.sh` # Restart `cron` like that: `systemctl restart cron` `systemctl status cron`	# Check out the rules in the file '`/etc/crontab`', then check out that the script '`/etc/cron.d/update_packages.sh`'(which are in charge of the package manager updating and upgrading) is executable and on the needed path. Look at the code of the script '`/etc/cron.d/update_packages.sh`'. # Check out the logs in '`/var/log/update_script.log`'. # Make sure the `cron` service is enabled: `systemctl status cron`
Check that there is a script to monitor the changes in the file '`/etc/crontab`' and sends an email to root if it has been modified. You must therefore receive an email showing that the file has changed, either locally with the mail order, either in your own mailbox. If not, this test has failed.	# Install mailutils in order to be able to send emails: `sudo apt install mailutils -y` # Create a file '`/etc/cron.d/monitor_changes.sh`' and change its permissions: `sudo chmod +x /etc/cron.d/monitor_changes.sh` # Add the following code to the '`/etc/cron.d/monitor_changes.sh`': `#!/bin/sh` `MSG="File /etc/crontab has been modified!"` `SUBJECT="Monitor changes crontab"` `TO="root@localhost"` `HASH="/etc/cron.d/hash.txt"` `FILE="/etc/crontab"` `test -f $HASH || sudo touch $HASH` `CRON_HASH=$(sudo md5sum $FILE)` `if [ "$(cat $HASH)" != "$CRON_HASH" ]; then` `echo $CRON_HASH > $HASH`	# Check out the rules in the file '`/etc/crontab`', then check out that the script '`/etc/cron.d/monitor_changes.sh`'(which are in charge of the package manager updating and upgrading) is executable and on the needed path. Look at the code of the script '`/etc/cron.d/monitor_changes.sh`'. # Check out that a new email sent to root appears at '`/var/mail/user`' every time any changes added to the file '`/etc/crontab`'. # Make sure the `cron` service is enabled: `systemctl status cron`

	echo $MSG \| mail -s "$SUBJECT" $TO fi; exit # Add a line to the file '/etc/crontab': 0 0 * * * root /etc/cron.d/monitor_changes.sh # Restart cron and make sure root receives an email each time something has changed in '/etc/crontab': systemctl restart cron systemctl status cron	
Check that there is self-signed SSL on all services. If this is not the case, this test is failed.	# Install SSL on the VM using this manual for nginx web server: https://www.8host.com/blog/sozdanie-samopodpisannogo-ssl-sertifikata-dlya-nginx-v-ubuntu-16-04/ Russian. # Replace the '/etc/nginx/sites-available/default' active configuration file with a new one called for instance '/etc/nginx/sites-available/active_conf' and add to it the following lines: server { listen <*VM's IP address*>:80; server_name localhost:<*mac port that forwards to the VM's HTTPS port*>; return 302 https://$server_name$request_uri; } server { listen <*VM's IP address*>:443 ssl http2; include snippets/self-signed.conf; include snippets/ssl-params.conf; root /var/www/html; server_name _; location / { try_files $uri $uri/ =404; } } # Create a symbolic link: sudo ln -s /etc/nginx/sites-available/active_conf /etc/nginx/sites-enabled/active_conf # Then remove file '/etc/nginx/sites-enabled/default'. Restart nginx and check its status: sudo systemctl restart nginx sudo systemctl status nginx	# Try to access the VM's web server from your mac's browser: https://localhost:44300, http://localhost:8080. Upon the request to the address http://localhost:8080 you should be redirected to https://localhost:44300.
Deployment Part		

<table>
<tr>
<td>We will check in this section that the web server is implemented on the VM. To validate this part, the next three tests must be passed. If at least one of them of them fail, the web server evaluation is failed and finished. Proceed to the next section of the scale.
From a shell of the VM, check that the package of a Web server is installed. If this is not the case, this test is failed.
From a shell of the VM, check that there is only one active configuration on the web server and not the default one. In addition, it should not "Listen" on the localhost of the VM. If it is not respected, this test has failed.
Check that the web application corresponds to what is required in the subject and is available on any browser on the IP of the VM or a host (init.login.fr for example). If it's not in this case, this test is failed.</td>
<td># Create a web page/site/application and move it to ‘<code>/etc/www/html</code>’ directory. Then make you see it when accessing the VM's web server.</td>
<td># Check out the status of <code>nginx</code> web server:
<code>sudo service nginx status</code>

Print out the list of <code>nginx</code> modules and packages:
<code>dpkg --get-selections | grep -iE "nginx"</code>

Try to access the VM's web server from your mac's browser and look at the web page/site/application:
https://localhost:44300
http://localhost:8080</td>
</tr>
<tr>
<td>We will check in this section that the deployment is going well. To validate this part, the following two tests must be succeed. - Ask the student to explain how he chose to do the deployment and why he chose this solution. - Make a minor modification on the site to ensure that the deployment is working well.</td>
<td># Install <code>git</code> on the VM:
<code>sudo apt install git -y</code>

Create a new bash script ‘<code>~/Desktop/autodeploy.sh</code>’ on your mac with the following source code:
<code>#!/bin/bash
YELLOW='\033[0;33m'
if [[$# -ne 2]] ; then
echo "Usage: ./autodeploy.sh [user of the VM to connect to] [SSH port of the VM]";
exit
fi
if [[-d $1]]; then
echo "$1 already exists"
exit
fi
Create script for deployment in remote server
deploy="deploy.sh"
website="website"
echo '#!/bin/bash' > $deploy
echo "cd /home/$1" >> $deploy
echo "if [[-d $website]]; then" >> $deploy
echo "rm -Rf $website" >> $deploy
echo "fi" >> $deploy
echo "mkdir $website && cd $website" >> $deploy
echo "mkdir site.git && cd site.git" >> $deploy
echo "git init --bare" >> $deploy
echo "cd hooks" >> $deploy
echo "echo '#\!/bin/bash' > post-receive" >> $deploy
echo "echo 'git --work-tree=/var/www/html --git-dir=/home/$1/$website/site.git checkout -f' >> post-receive" >> $deploy
echo "chmod +x post-receive" >> $deploy
chmod +x $deploy
ssh $1@localhost -p $2 'bash -s' < deploy.sh
rm -Rf deploy.sh</code></td>
<td># Run the script ‘<code>~/Desktop/autodeploy.sh</code>’ on your mac and move to the new directory made by the script:
<code>./autodeploy.sh user <mac port that forwards to the VM's SSH port>
cd user</code>

Open ‘<code>~/Desktop/user/index.html</code>’ on your mac and make changes to the document. Then run the script:
<code>./Desktop/user/update.sh</code>
After that the a version of ‘<code>~/Desktop/user/index.html</code>’ with the changes you've just added would be committed and pushed to the repo hosted on the VM at ‘<code>/home/user/website</code>’ and automatically hooked by the VM's web server. Try to access the VM's web server from your mac's browser and look at the changed web page/site/application:
https://localhost:44300
http://localhost:8080</td>
</tr>
</table>

```
# Create a folder on the local machine
mkdir $1 && cd $1
scp -P $2 $1@localhost:/var/www/html/index.html ./
git init
git remote add live
ssh://$1@localhost:$2/home/$1/$website/site.git
# Create git push bash script (update.sh):
update="update.sh"
echo '#!/bin/bash' > $update
echo 'if [[ $# -ne 1 ]]; then' >> $update
echo -n 'echo "Usage: ./' >> $update
echo -n $update >> $update
echo ' [git commit comments]"' >> $update
echo 'exit' >> $update
echo 'fi' >> $update
echo 'git add .' >> $update
echo 'git commit -m "$1"' >> $update
echo 'git push live master' >> $update
chmod +x $update
echo 'exit' >> $update
echo '*.sh' > .gitignore
bash update.sh "init"
echo -e "${YELLOW}I work thanks to the efforts of the
members of"
echo -e "                  R4Y7r4c1N9_C4P031R4_CR3W"
echo -e "Consider visiting r4y7r4c1n9c4p031r4.xyz"
exit
```

Make the script '`~/Desktop/autodeploy.sh`' executable:

```
chmod +x ~/Desktop/autodeploy.sh
```