université PARIS-SACLAY

IUT D'ORSAY

Faustin MILLET, Jean COSTREL DE CORAINVILLE and Romain REN,
students in first year

Project S103

Novembre 2022

Contents

Web server is one of the most needed component of our actual usage of technology. During, the S103 lab, we install a web server on one of the most simple computer, a Raspberry Pi.

We choose to install Arch Linux, it's not a very usual production environment. But it's a fun choice ! The root password is `root` and the login for our private website is `toto` with the password `toto`.

0.1 Distribution of work

Everybody contributes equally to the project, but of course there is only one keyboard attach to the RPi. **Romain** (30%) was in charge of the configuration of Apache, **Jean** (30%) was in charge of the pre-installation and **Faustin** (40%) was in charge of the troubleshooting and finalize the report.

Part I
Pre-installation

1 Preparation

1.1 Partitioning

We had chosen to install Arch Linux ARMv7, but without the permissions on the machine, we were not allowed to format the disk. We chose to install Arch from our personal computer, and we export it on the SD card. With a root access from our personal terminal, we had set SD-Card's partitions with the following command.

```
fdisk /dev/sdb
```

After creating our new partitions, we create and mound the FAT file system and the ext4 file system. With those couple of commands, we format a disk partition, we prepare the file system, so that the operating system can use these partitions and store files.

```
mkfs.vfat /dev/sdb1
mkdir boot
mount /dev/sdb1 boot
```

```
mkfs.ext4 /dev/sdb2
mkdir root
mount /dev/sdb2 root
```

1.2 Preparation of the file system

As root, we download the root file system, then we extracted it and redirected it to the root partition. Finally, we have synchronized cached writes to persistent storage.

```
wget http://os.archlinuxarm.org/os/ArchLinuxARM-rpi-armv7-latest.tar.gz
bsdtar -xpf ArchLinuxARM-rpi-armv7-latest.tar.gz -C root
sync
```

Thereafter, we move the boot files to the first partition, and we unmount the two partitions.

```
mv root/boot/* boot
umount boot root
```

First boot Next, we finally insert the SD Card on the Raspberry Pi. We connect it to an Ethernet Cable, now the Raspberry Pi is connected and is running. Then we switch on the Raspberry Pi and login in with default user and default password, both are `root`, and we never change those. For easier manipulation, we have changed the keyboard from the default `qwerty` to an `azerty` keyboard. At the same time, we took the opportunity to change the timezone of the system, With a simple command.

```
    localectl set-keymap -no-convert fr-pc
    timedateectl set-timezone Europe/Paris
```

To complete the installation, we initialize the pacman keyring, and populate the Arch Linux ARM package signing keys. The installation of the keys ensures that the packets received are those of the author of the packet.

```
    pacman-key --init
    pacman-key --populate archlinuxarm
```

1.3 Set up an ssh server

Then we set up an ssh server to be connected from the computer, and benefit from the advantages of our computer including its interface. This allows us to import our websites and to check more easily if our websites are working. To do this, we used the following command:

```
    nano /etc/ssh/sshd_config
```

We modified two lines in `/etc/ssh/sshd_config`

```
-- PermitRootLogin prohibit-password
++ PermitRootLogin yes

-- #PasswordAuthentification yes
++ PasswordAuthentification yes
```

The first one specifies that the root user can log in using ssh, and the second one specifies that password authentication is allowed.

To finish the ssh configuration we confirmed the changes by reloading the service to save the changes. With this one command.

```
    systemctl reload sshd
```

Part II
Installation and configuration

1 Installation of Apache2

To prepare the configuration of a website, searched what package we will use. Than update package database on the system with just one command. Finally, again with pacman, we downloaded and install the Apache package.

```
    pacman -Syy
    pacman -S apache
```

With this we have a working operating system on the Raspberry, and we are ready to configure a website on it.

2 Configuration of Apache2

We installed an Apache Server on the Raspberry and we tested with simple HTML file, now we have to configure it so we can put a website on it. But how can we host multiples sites on the same Apache server ?

2.1 Virtual hosting

We can use Virtual host to host multiple website on the same physical host. Here we first choose to use domain-based virtual hosting then, according to teacher Zema, we choose to switch to port-based virtual hosting. We choose to put ours EPUBs in the Raspberry as the website, so we transferred our files using SFTP with our SSH server. So now the EPUBs files are in the Raspberry at `/srv/vhost/` and the server is ON, we just have to make the Raspberry read it.

We added a new configuration file, `S103-vhosts.conf`, that replaces, `httpd-vhosts.conf`, with the website virtual host configuration, for example :

```
## Domain based
# Main index
<VirtualHost *:80>
    ServerAdmin admin@faustin-yuno.ga
    DocumentRoot "/srv/http"
    ErrorLog "/var/log/httpd/error_log"
    TransferLog "/var/log/httpd/access_log"
</VirtualHost>

# Romain websites index
<VirtualHost *:80>
    ServerAdmin admin@faustin-yuno.ga
    DocumentRoot "/srv/vhost/romain"
    ServerName romain.rpi.faustin-yuno.ga
    ErrorLog "/var/log/httpd/romain-error_log"
    CustomLog "/var/log/httpd/romain-access_log" common
</VirtualHost>

## Port based
<VirtualHost *:81>
    ServerAdmin admin@faustin-yuno.ga
    DocumentRoot "/srv/vhost/romain"
    ErrorLog "/var/log/httpd/romain-error_log"
    CustomLog "/var/log/httpd/romain-access_log" common
</VirtualHost>
```

And used `chmod -R 755 /srv/vhost/romain/*` to ensure that dedicated user `httpd` could read the files.

There is the same configuration for Faustin and Jean website.

So now you can access the main index and choose which website you will go to, or directly access using the port or the domain. BUT there is no security, so we will have to put a auth using simple password.

2.2 Password protection

To ensure password protection, we have added to the `S103-vhosts.conf` file those lines who specify the policy for reading the directory.

```
<Directory "/srv/vhost/*">
    Options Indexes FollowSymLinks
```

```
        AllowOverride None
        # The lines we added :
        AuthType Basic
        AuthName "Connectez-vous pour acceder au serveur"
        AuthBasicProvider file
        AuthUserFile "/etc/httpd/passwd"
        Require valid-user
    </Directory>
```

That Apache will now ask a password when you will want to access the personnal website.
We used `htpasswd -c /etc/httpd/passwd toto` to create the file with the login `toto` hashed password.

2.3 SSL/TLS

HTTPS ensure that the communication between our server and the client can't be intercepted and read. In a production environment, we need a certificate from a certificate authority like Let's encrypt but in an environment like our we can self-signed our certificate.

```
# Generate the private RSA pair
openssl genrsa 2048 > server.key
# Create the certificate with the private RSA key
openssl req -new -x509 -nodes -sha256 -days 365 -key server.key -out server.crt
```

OpenSSL generates the private key and the certificate ready to be use by Apache.
We added some module that was required for configuration of the websites on the Raspberry in the `httpd.conf` file, at `/etc/httpd/conf/` directory, which are the following :

```
LoadModule log_config_module modules/mod_log_config.so
LoadModule log_config_module modules/mod_setenvif.so
LoadModule log_config_module modules/mod_ssl.so
LoadModule log_config_module modules/socache_shmcb_module.so
```

Finally we add `Include conf/extra/httpd-ssl.conf` to main configuration file, and changed some lines in `httpd-ssl.conf`.

```
#   General setup for the virtual host
DocumentRoot "/srv/http"
ServerAdmin admin@faustin-yuno.ga
ErrorLog "/var/log/httpd/error_log"
TransferLog "/var/log/httpd/access_log"
```

And of course we reload the httpd service with

```
systemctl reload httpd
```

Part III
Recovery and system troubleshooting

1 Create a backup image of the SD-Card

Everybody knows that ArchLinux is a very easy to break Linux distribution. When we have a stable Arch installation, we take the initiative to create a image of our SD-Card with the `dd` utility. Here is the command to create an image of the SD-Card.

```
    sudo dd if=/dev/disk4 of=Downloads/RPIBackup.iso
```

It was very useful to have a safe, ready to use, image of our SD-Card. In fact, we break our Arch installation 2 hours after the backup by updating the `openssl` package.

2 Arch and the OpenSSL package

Arch doesn't like partial update. So the first time we install apache2, we haven't upgrade already installed packages. When we begin to use our Apache server we don't have any troubles. But when we activate the SSL module, Apache won't start with this error message `libssl.so.3 cannot open shared object file no such file or directory`. When we upgrade the package `openssl`, we broke our system installation, a lot of essentials packages need a correct OpenSSL package, and we need to use our backup. So I use a Arch virtual machine with QEMU, mount the SD-Card and recover our config files. To upgrade OpenSSL, we need to do a full system upgrade with `pacman -Syu`.

3 Domain and internal IP

You can point a A DNS record to a private IP, and we quickly notice that our RPi have always the `10.42.0.2` address. So we point subdomains to the private IP address to get a certificate with Let's Encrypt and do virtual hosting by domain. But in some hall, the IP address of the RPi is different, and it's not a reliable solution.