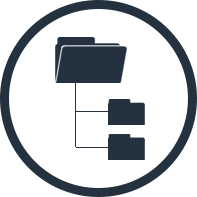
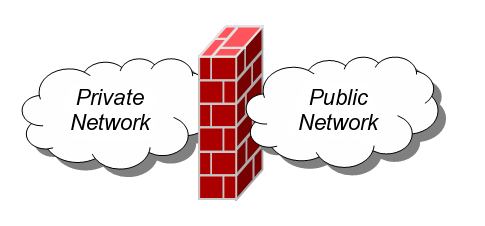
Secure Embedded System (SeS)

SSHD FileSystem Firewall

Auteur : Spinelli Isaia

Prof : [Schuler Jean-Roland](https://mse.hes-so.ch/consultation/master/horairesMSE.php?annee=2020&id=81&type=1)

Date : 26.10.2020

Salle : A4 – Lausanne

Classe : SeS

Table des matières

[Introduction - 2 -](#_Toc55232466)

[SSHD - 2 -](#_Toc55232467)

[Introduction - 2 -](#_Toc55232468)

[Installer et vérifier la signature d’OpenSSH - 2 -](#_Toc55232469)

[Configurer et installer sur intel - 4 -](#_Toc55232470)

[Configurer et installer sur Nano Pi - 6 -](#_Toc55232471)

[Création de clefs - 7 -](#_Toc55232472)

[Configuration sshd - 7 -](#_Toc55232473)

[Test final - 8 -](#_Toc55232474)

[Modification des sources - 10 -](#_Toc55232475)

[Conclusion SSHD - 10 -](#_Toc55232476)

[Annexes - 11 -](#_Toc55232477)

# Introduction

Ce rapport est composé de trois laboratoires distincts :

1. SSHD
2. FileSystem
3. Firewall

# SSHD

## Introduction

Durant ce laboratoire nous allons principalement nous familiariser avec le protocole SSH et les utilitaires OpenSSH. SSH est protocole de connexion qui impose un échange de clés de chiffrement en début de connexion. Le port utilisé par défaut est le 22. OpenSSH est un ensemble d’outils open source utilisant le protocole SSH.

Durant ce laboratoire, plusieurs étapes nous a été demandé :

1. Installer et vérifier la signature d’OpenSSH
2. Configurer les packages sur notre machine Intel.
3. Configurer les packages sur la cible Nani Pi
4. Installer les différents utilitaires sur la cible.
5. Configurer des options de SSHD.
6. Modifier SSHD afin de ne plus indiquer la version utilisée.

## Installer et vérifier la signature d’OpenSSH

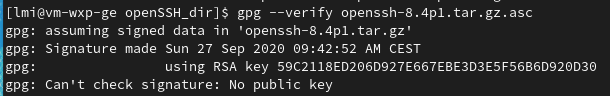
Afin d’installer OpenSSH il a fallu se rendre sur son site officiel (<https://www.openssh.com/>) et récupérer le lien http Suisse afin d’installer la dernière version :

*Wget https://mirror.ungleich.ch/pub/OpenBSD/OpenSSH/portable/*[*openssh-8.4p1.tar.gz*](https://mirror.ungleich.ch/pub/OpenBSD/OpenSSH/portable/openssh-8.4p1.tar.gz)

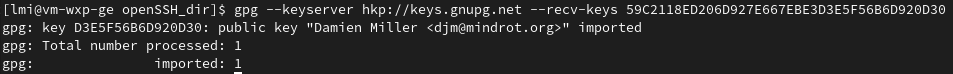
Dans le même temps, on peut télécharger la signature de celui-ci :

*Wget https://mirror.ungleich.ch/pub/OpenBSD/OpenSSH/portable/*[*openssh-8.4p1.tar.gz.asc*](https://mirror.ungleich.ch/pub/OpenBSD/OpenSSH/portable/openssh-8.4p1.tar.gz.asc)

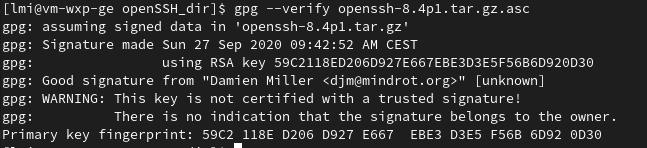
Pour commencer, il est important de vérifier l’authenticité et l’intégrité du logiciel :



On peut voir que le check n’est pas possible car la clef public n’est pas présente. Il faut importer cette clef sur notre PC comme suit :



On peut voir que la clef public de Damien Millet a correctement été importée. Il est maintenant possible de vérifier la signature du package :



## Configurer et installer sur intel

Maintenant que nous avons téléchargé OpenSSH, nous pouvons le configurer et l’installer sur notre machine hôte. Voici la commande à exécuter :

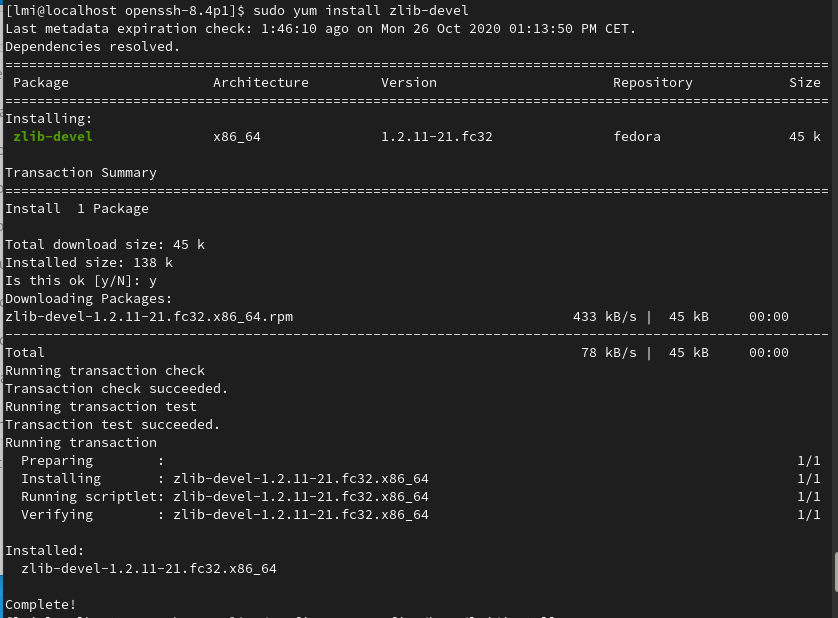
*./configure --prefix=/home/lmi/install*

L’option « –prefix » permet de modifier le répertoire d’installations.

Personnellement, j’ai eu l’erreur suivante :



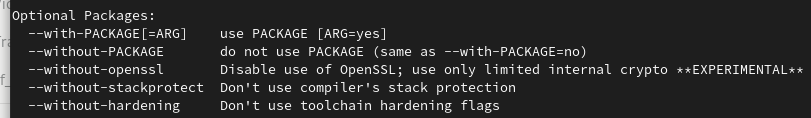
La librairie « zlib » n’est pas présente. Donc, je l’ai installée comme suit :



OpenSSH permet énormément de configuration et de différentes options, pour en savoir plus, il est intéressant de lire les fichiers « README » et « INSTALL ». Il est aussi possible d’utiliser la commande suivante :



Il nous est demandé d’activer l’option « hardening », cependant, grâce à la commande précédente, on peut apprendre que cette option est par défaut activée. Il est possible de la désactiver avec l’option « --without-hardening ».



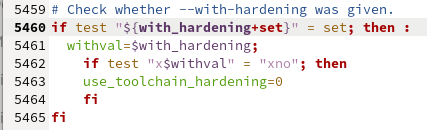
En m’intéressant aux différentes options proposées, j’ai pu voir celle-ci :



Il aurait été possible d’ajouter cette option dès le début afin de ne pas dépendre de la libraire « zlib ».

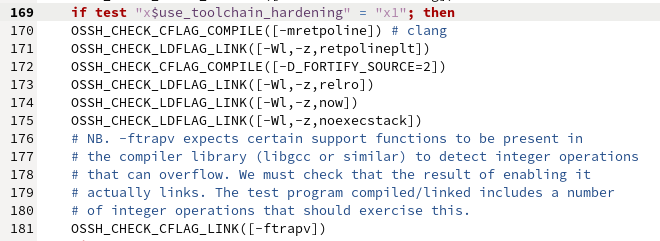
Afin de mieux comprendre l’impact de l’option « hardening », j’ai effectué une recherche de cette option et de son influence. Voici ce que j’ai pu trouver :

1. Dans le fichier « configure » :

7

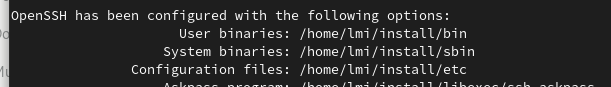
On peut voir que si l’option « hardening » activera, ou pas, la variable « use\_toolchain\_hardening ».

1. Dans le fichier « configure.ac » :



Dans le cas où cette même variable est active, cela va ajouter plusieurs option de compilation de protection listées ci-dessus.

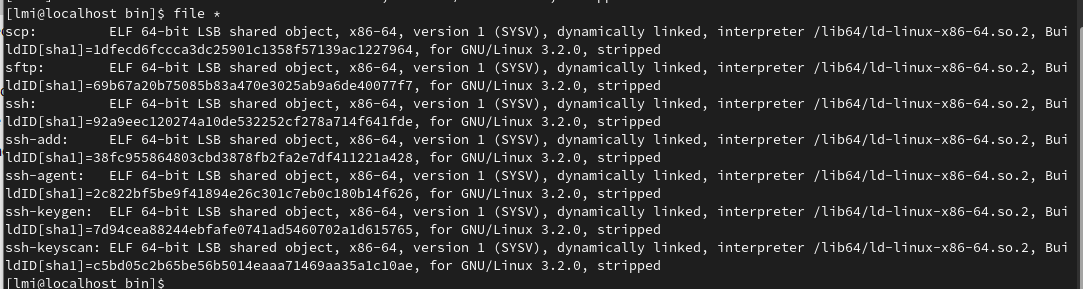
Une fois la configuration effectuée, un message résumant les informations de configuration son affichée :

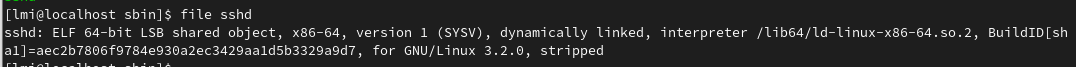


Ensuite, il est possible d’effectuer la compilation et l’installation avec les commandes suivantes :



Une fois tous les exécutables générés, il est possible de vérifier s’ils sont « stripped » :





On peut constater que tous les exécutables sont bien « stripped ».

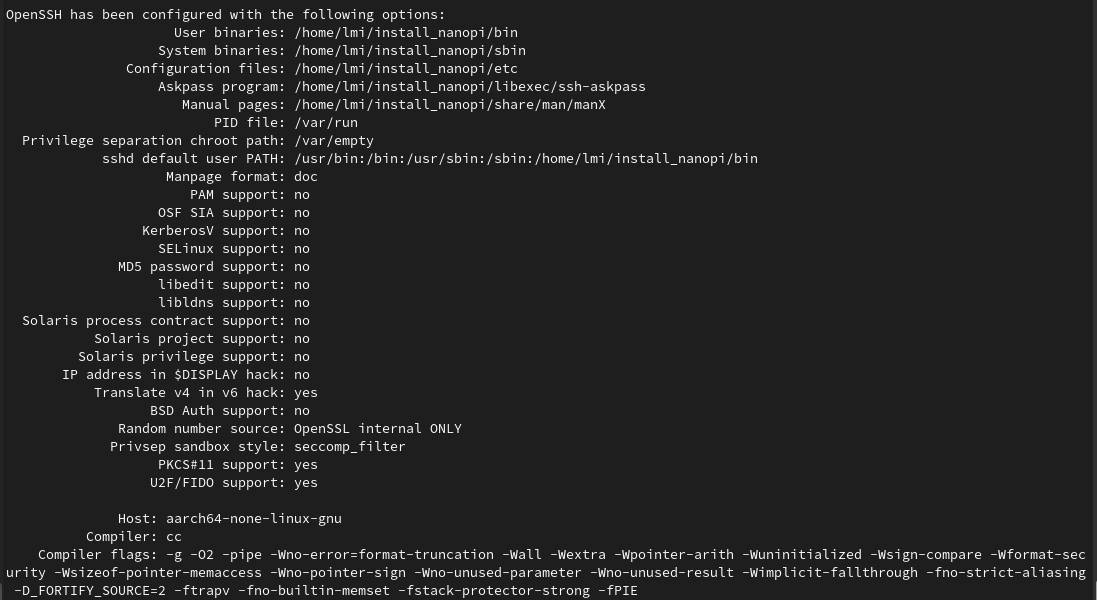
## Configurer et installer sur Nano Pi

L’objectif de cette étape est de configurer et installer OpenSSH sur la cible Nani Pi en utilisant la cross-compilation.

Étant donné cet utilitaire est relativement bien fait, il est facile de le cross-compiler. Il suffit d’ajouter une option lors de la configuration des packages. On peut voir la commande exécutée ci-dessous :

*./configure --host=aarch64-none-linux-gnu --prefix=/home/lmi/install\_nanopi*

On peut voir ci-dessous la sortie de la configuration. On peut voir en bas tous les flags de compilation activés.

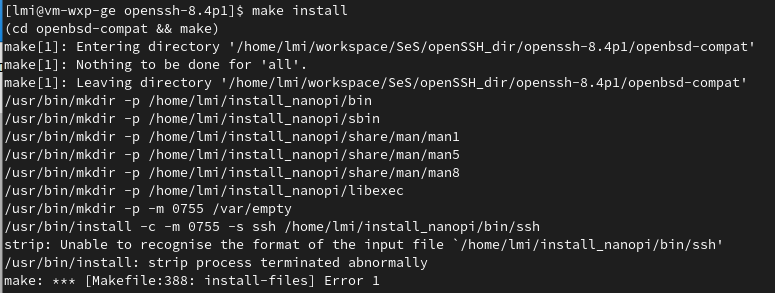


Ensuite, les deux commandes de compilation et d’installation peuvent être exécutées :

*make*

*make install*

Dans mon cas, la commande « make install » provoque une erreur :



La commande « strip » effectuée dans le script « /usr/bin/install » provoque une erreur. En faisant une recherche, voici ce que j’ai pu trouver :



On peut voir que la commande « strip » executée est celle par défaut avec un flag « -s ». Cependant, comme nous cross-compilons, il est nécessaire d’indiquer la commande « strip » pour notre architecture grâce au flag « --strip-program » :



Après cela, l’installation a pu se faire correctement.

Une fois que tous les fichiers exécutables sont correctement générés pour notre cible, il suffit de les placer sur le système de fichier de la Nano PI. Il y a plusieurs manières de procéder, pour ma part, j’ai utilisé le système de fichier réseau (NFS) préalablement installé.

## Création de clefs

L’objectif est de créer différents types de clef sur la cible à l’aide de l’utilitaire « ssh-keygen » précédemment installé. Voici ci-dessous la liste des différents type de clef crées et les commandes permettant leur création :

1. rsa 4096 bits



1. dsa 1024 bits



1. ecdsa 521 bits



1. ed25519 256bits



## Configuration sshd

La configuration SSHD se fera via le fichier « sshd\_config ». Afin de répondre aux configurations demandées, voici les lignes à ajouter au fichier :

**Utilisation uniquement de IPv4 :**

AddressFamily inet

**Ne pas permettre la redirection de port :**

AllowTcpForwarding no

AllowStreamLocalForwarding no

GatewayPorts no

PermitTunnel no

**Autoriser des algorithmes de chiffrement de hachage :**

Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes256-cbc

MACs hmac-sha1,hmac-sha2-256

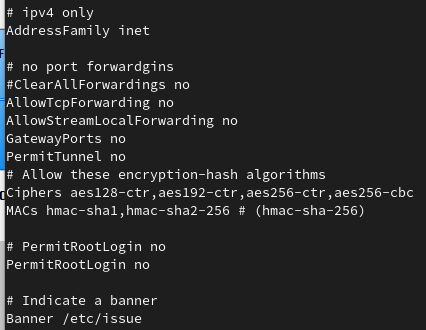
**Connexion root pas permis :**

PermitRootLogin no

**Indiquer une bannière :**

Banner /etc/issue

Voici plus précisément le fichier de configuration « sshd\_config ».



## Test final

L’objectif maintenant est de tester l’outil sshd sur la cile.

Afin de contrôler si un service ssh est en cours, on peut utiliser la commande « netstat -atu » afin de vérifier les programmes qui écoute les ports. Ensuite, il faut désactiver le service ssh en cours par défaut avec la commande « /etc/init.d/S50dropbear stop ».

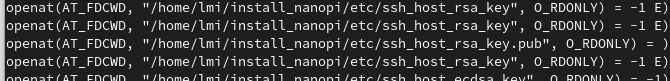
Afin de lancer le programme sshd précédemment configuré et installé, voici la commande à effectuer tout en précisant le fichier de configuration précédemment crée :



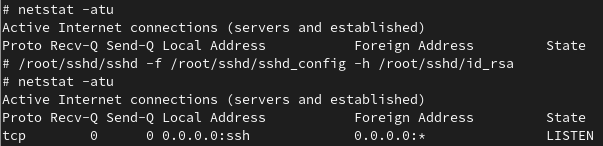
On peut remarquer qu’il nous indique qu’il ne dispose pas de clef hôte. Dans ce cas, il peut être intéressant d’utiliser la commande « strace » afin de mieux comprendre d’où vient le problème :



On peut voir ci-dessous que les clefs sont recherchées dans le dossier « /home/install\_nanopi/etc/ » :

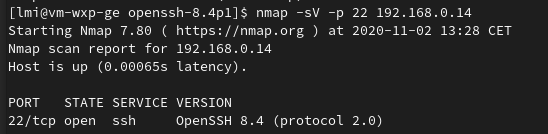


Il existe plusieurs façons d’indiquer l’emplacement des clefs. Personnellement, en analysant les paramètres possible au programme sshd, j’ai pu remarquer que l’option -h permet ceci. On peut voir ci-dessous la lancement de sshd en indiquant la clef rsa :



Grâce à la commande « netstat -atu », j’ai pu confirmer que le programme sshd c’était correctement lancé.

## Modification des sources



## Conclusion SSHD

Ce laboratoire m’a permis de comprendre les principes et le fonctionnement des clefs et des signatures dans le domaine de la sécurité. De plus, il m’a fait prendre conscience que la cross-compilation pouvait rapidement être complexe. J’ai pu accroitre ma connaissance en ce qui concerne la configuration du protocole SSH et des packages en général. Finalement, j’ai pu découvrir quelques commandes pratiques tels que « strace » et « netstat ».

# File system

## Introduction

Durant ce laboratoire nous allons principalement nous familiariser avec

Ce laboratoire est composé de plusieurs étapes distinctes :

1. Kernel and rootfs configuration
2. EXT4
3. Btrfs, f2fs, nilfs2 et xfs
4. LUKS, cryptsetup et dmcrypt
5. initramfs
6. initramfs-LUKS partition

## Kernel and rootfs configuration

Ce chapitre permet de configurer le noyau et le rootfs grâce à l’interface menu-config. Plusieurs configurations ont été ajoutées :

1. mkfs.ext2 (busybox-menuconfig)
2. tune2fs (busybox-menuconfig)
3. Btrfs, NILFS2, F2FS, XFS (linux-menuconfig)
4. USB Mass Storage (linux-menuconfig)
5. Cryptsetup (buildroot et linux-menuconfig)
6. Initramfs (linux-menuconfig)

Toutes ces étapes de configurations sont détaillées dans la donnée du laboratoire en annexe [1].

Une fois que la configuration est terminée, il faut générer et installer le nouveau kernel et rootfs sur la carte SD.

## EXT4

### Comment le noyau sait que le rootfs is sur la deuxième partition de la carte SD ?

### Comment monter la première partition de la carte SD sur /mtn ?

Mount mmcnlkp1 /mnt (nano)

### Quel est le numéro majeur et mineur du fichier nœud gérant la carte SD ?

## btrfs, f2fs, nilfs2, xfs

### Créer deux nouvelles partitions (3 et 4)

### Formater ces deux nouvelles partitions avec deux différents systèmes de fichier

### Compare les temps d’écriture entre les différents système de fichier

## LUKS, cryptsetup et dmcrypt

### Créer une troisième partition LUKS

### LUKS, cryptsetup options

1. « Plain Mode » vs « LUKS extension mode » : La différence est que LUKS utilise un en-tête de métadonnées et peut donc offrir plus de fonctionnalités que dm-crypt (Plain Mode) . D'autre part, la tête est visible et vulnérable aux dommages. À moins de très bien comprendre les contextes de cryptographie, il est en général conseillé d’utiliser LUKS.
2. –hash permet de remplacer la fonction de hachage par défaut pour le hachage de phrase de passe (passphrase).
3. La valeur par défaut actuelle dans les sources distribuées pour le chiffrement "aes-xts-plain64" pour LUKS.
4. --key-file permet principalement de fournir le passphrase par un fichier.

### LUKS test 1

Initialise une partition LUKS en format ext4 et la monter dans le répertoire /mtn/usr

Copier un fichier dans la partition LUKS (aaaaa)

Ajouter un nouveau passphrase à la partition LUKS (2 clef)

Dump l’en-tête de la partition et la clé principale cryptée ()

Dump 1 Mbytes de la partition /dev/sbd3 sur un fichier. Peut-on trouver l’en-tête de la partition et le clé principale cryptée ? (aaaaaa ?)

Activer la partition cryptée sur la NanoPi

### rootfs dans une partition LUKS

Le but de ce chapitre est d’avoir un rootfs crypté sur la partition 3.

Générer un passphrase aléatoire dans un fichier

Initialise une partition LUKS sur la partition 3

Créer un mapping /dev/mapper/usrfs1

Formater la partition LUKS en format ext4

Copier le rootfs sur la partition LUKS

Démarrer la NanoPi et monter manuellement la partition LUKS

Écrire un script (/etc/S40luks) afin de monter automatiquement la partition

### initramfs

Générer un initramfs

Initialise la NanoPi afin de démarrer l’initramfs

Boot.scr

Démarre la NanoPi et démarre manuellement

Depuis le shell sur l’initramfs, monter la partition LUKS

Démarrer manuellement la partition roofs encryptée

Écrire un scrpit afin de démarrer automatiquement la partition rootfs encryptée.

# Annexes

1. Donnée laboratoire « FileSystem »

Date : x.yxx.20

Nom de l’étudiant : Spinelli Isaia