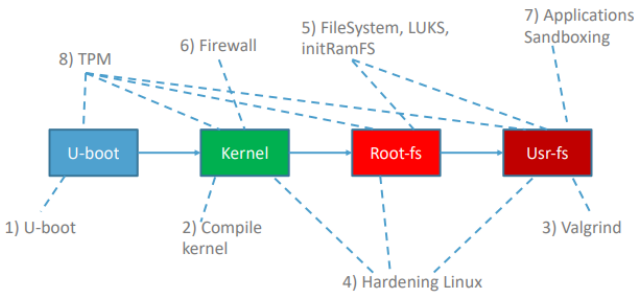
1 Introduction



1.1 Attaques

1. Attaques de surface
   (a) Utilisateurs des ports de debug
   (b) Connecteurs
   (c) Alimentations
2. Vecteurs d'attaque
   (a) Réseau (Ethernet, Wifi)
   (b) Application
   (c) Port série
   (d) USB, I2C, Flash, Bluetooth, GPS, etc...

1.2 Compilation pour nanopi

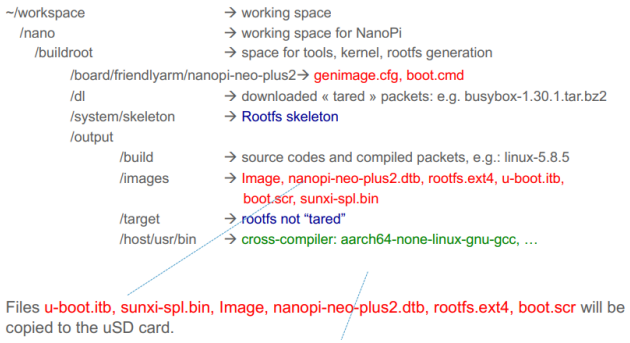
Cross-compilation (ARM) effectuée sur un système x86/x64. Buildroot est le toolchain utilisé. Les éléments suivants sont compilés :

1. Bootloader
2. Kernel
3. Rootfs

Puis les images sont copiées sur la carte SD

2 Buildroot

2.1 Répertoires



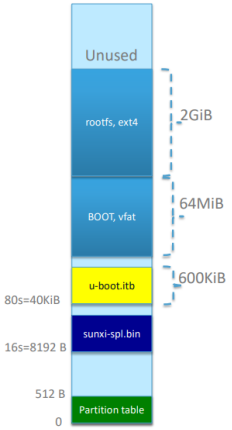
Ce qui est manquant de le dossier output sera recompilé lorsque la commande `make` est lancée (ou alors en faisant la commande `make <package>-rebuild`.
Le dossier `rootfs_overlay` permet d'ajouter des fichiers au `rootfs` (`/workspace/nano/buildrootboard/friendlyarm/nanopi-neo-plus2/rootfs_overlay`)

2.2 Compilation

Dans le répertoire `buildroot`, effectuer `make menuconfig` puis `make`. `make clean` pour effacer tous les fichiers compilés.
La configuration permet notamment de

1. Modifier le rootfs

2.3 Carte SD



Rootfs (ext4)		2GiB	
BOOT, vfat / ext4		64 MiB	
U-Biit-itb	80		600KiB
sunxi-spl.bin	16	79	32KiB
MBR (partition table)	0	15	512B

3 U-boot

3.1 Compilation

On configure avec `make ubiit-menuconfig` puis on effectue la compilation avec une des deux manières :

1. `make uboot-rebuild`
2. supprimer les fichiers puis `make`

4 Kernel

4.1 Compilation

On configure avec `make linux-menuconfig` (ou `make linux-xconfig`) puis on lance une compilation avec `make linux-rebuild`

4.2 Busybox

Busybox est un éxécutable qui combine beaucoup de fonctions de base (ls, mv, rm, cat, etc...). En mettant

toutes ces commandes dans un seul programme, on réduit énormément les redondances et par conséquent la taille de l'éxécutable.
On peut également configurer busybox avec `make busybox-menuconfig` puis le compiler avec `make busybox-rebuild`

5 Valgrind

6 Hardening

7 File system

7.1 Génération

Squelette de rootfs dans `workspace/nano/buildroot/system/skeleton`.
Il est ensuite copié dans `buildroot/output/target` et les fichiers nécessaires y sont ensuite ajoutés.
Une fois que tous les fichiers sont ajoutés, une image `rootfs.xxx` est créé (xxx est ext4, squashfs, etc...)

7.3.1 Choix d'un FS

7.2

7.3 1. De connaître les différents types de systèmes de fichiers ainsi que leurs applications

Pour les systèmes embarqués, il existe deux catégories de systèmes de fichiers : - Volatiles en RAM - Persitants sur des Flash (NOR et de plus en plus NAND)

Deux technologies principales sont disponible sur les Flash : - soit les MTD (Memory Technology Device) - les MMC/SD-Card (Multi-Media-Card / Secure Digital Card)

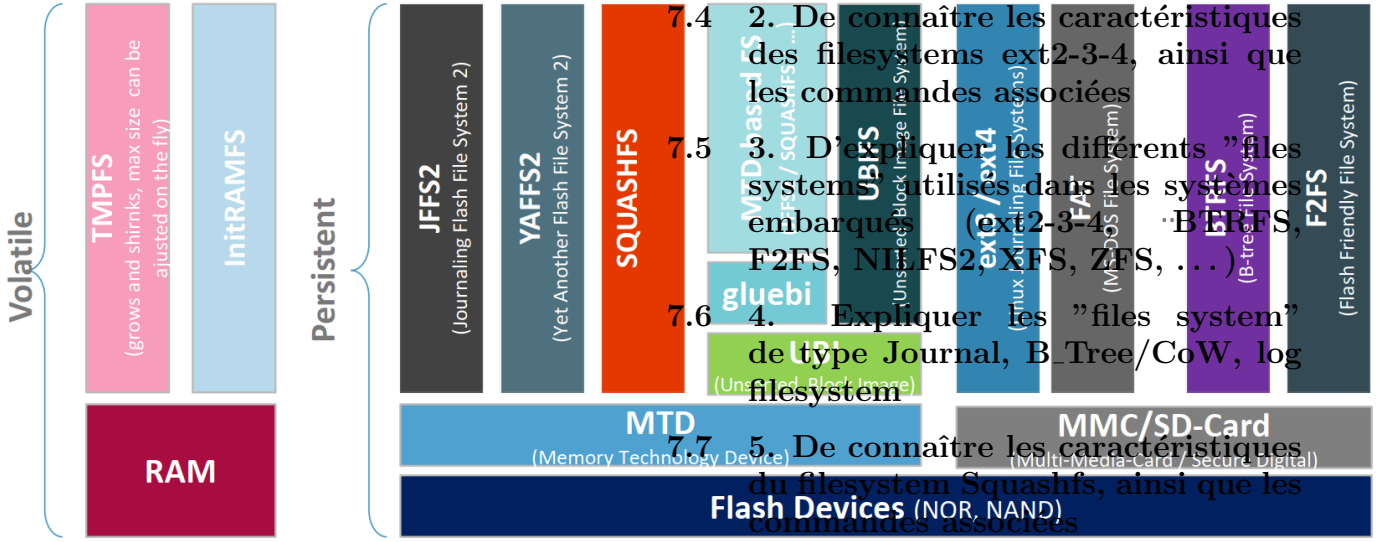


Figure 1: FS type

7.4 2. De connaître les caractéristiques des filesystems ext2-3-4, ainsi que les commandes associées

7.5 3. D'expliquer les différents "files systems" utilisés dans les systèmes embarqués (ext2-3-4, BTRFS, F2FS, NILFS2, XFS, ZFS, ...)

7.6 4. Expliquer les "files system" de type Journal, B_Tree/CoW, log filesystem

7.7 5. De connaître les caractéristiques du filesystem Squashfs, ainsi que les commandes associées

7.8 6. De connaître les caractéristiques du filesystem tmpfs, ainsi que les commandes associées

7.9 7. De connaître les caractéristiques du filesystem LUKS, ainsi que les commandes associées

7.10 8. Savoir expliquer la gestion des clés de LUKS 42

7.11 9. De connaître les caractéristiques du filesystem InitramFS, ainsi que les commandes associées

7.12 10. De savoir créer un initramFS

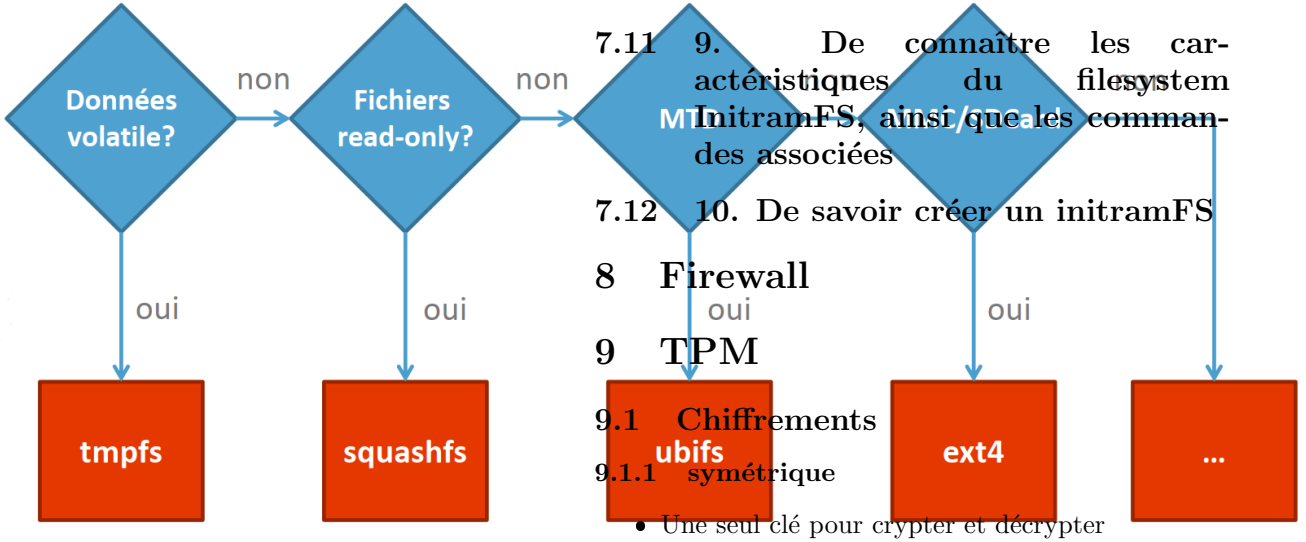


Figure 2: FS type

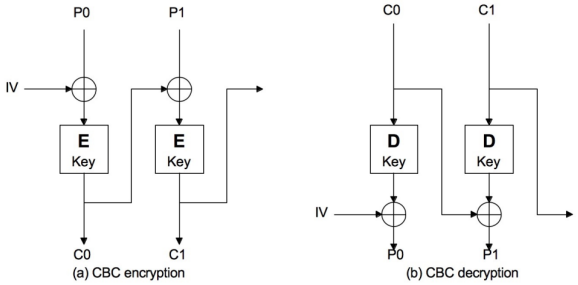
8 Firewall

9 TPM

9.1 Chiffrements

9.1.1 symétrique

- Une seul clé pour crypter et décrypter
- Codage par bloc ou par bloc chainé



- `openssl enc -aes-256-cbc -e -in t.txt -out t.enc` //encrypt
- `openssl enc -aes-256-cbc -d -in t.enc -out t`

9.1.2 asymétrique

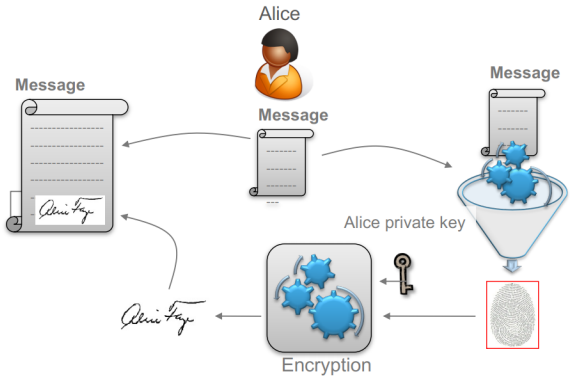
- Deux clés (publique et privée) clé publique disponible par des certificats (CMD pgp)
- Encrypt public -¿ Decrypt private =¿ confidentialité
- Encrypt private-¿ Decrypt public =¿ signature digitale

9.1.3 hash

Transforme un texte, document en un nombre de N bits unique (SHA-2, SHA-3, Blake2).
`md5sum file => a6a0e8d0522e2c5de921b1c455506320`
ou `openssl dgst -md5 file MD5(file)= a6a0e8d0522e2c5de921b1c455506320`

9.1.4 signature

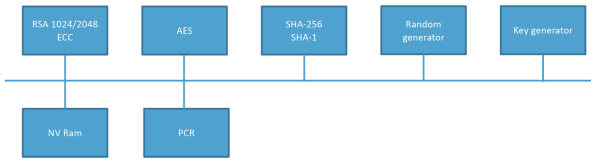
En deux parties: 1. Calcul du HASH puis encryptage avec clé privée.



9.2 Implémentations TPM

- discrete : Circuit dédié
- integrated : Partie du μC qui gère le TPM
- Hypervisor : virtuel fournis par personne fiable
- Software : virtuel pour faire des test pas sécurisé

9.3 Architecture interne



- RSA 1024/2048, ECC: Asymmetric algorithms, encrypt-decrypt, sign
- AES: Symmetric algorithm, encrypt-decrypt, sign
- SHA-256, SHA-1: hash function
- Random generator: create random value
- Key generator: Create key for asymmetric algo
- NV Ram: Store different objects (keys, data, ...) in NV Ram
- PCR (Platform Configuration Registers) stores hash values of different parts: code, files, partitions, ...

9.4 Hiérarchies

- endorsement : réservé au fabricant du TPM et fixé lors de la fabrication.
- platform : réservé au fabricant de l'hôte et peut être modifier par l'équipementier.
- owner : hiérarchie dédiée à l'utilisateur primaire du TPM peut être modifié en tout temps.
- null : réservé aux clés temporaires (RAM s'efface à chaque redémarrage)

9.5 Créer, utiliser clés

Create RSA endorsement key:	`tpm2_createprimary -C e -G rsa2048 -c e_primary.ctx`
Create RSA platform key:	`tpm2_createprimary -C p -G rsa2048 -c p_primary.ctx`
Create RSA owner key:	`tpm2_createprimary -C o -G rsa2048 -c o_primary.ctx`
Create RSA null key:	`tpm2_createprimary -C n -G rsa2048 -c n_primary.ctx`

9.6 Commandes principales

- `tpm2_createprimary -C o -G rsa2048 -c o_prim` // créer un clé primaire owner
- `tpm2_getcap handles-transient` // voir clé dans la RAM
- `tpm2_getcap handles-persistent` // voir clé dans la NV-RAM
- `tpm2_evictcontrol {c o_primary.ctx` // sauver une clé en NV-RAM
- `tpm2_flushcontext -t` // effacer toute la RAM
- `tpm2_create -C o_prim -G rsa2048 -u child_pub` // créer clé enfant
- `tpm2_load -C o_prim -u child_pub -r child_priv` // charger clé enfant

9.7 encrypter-décrypter, signer-vérifier

- `tpm2_rsaencrypt -c child -s rsaes clearfile -o`
- `tpm2_rsadecrypt -c child -s rsaes encryptedfil`
- `tpm2_sign -c child -g sha256 -o file.sign file`
- `tpm2_verifysignature -c child -g sha256 -s fil`

10 Autres

10.1 Commandes

Commande	Description
`netcat` (`nc`)	Couteau suisse du TCP/IP. Permet de scann
`nmap`	Analyse des ports ouverts
`ssh`	Connexion à un système par interpréteur de

10.2 Définitions

Nom	Description
Honeypot	"Pot de miel" ou leurre pour faire croire qu'un
Toolchain	Codes sources et outils nécessaires pour générer
Kernel	Coeur Linux (avec le format u-boot)
Rootfs	Root Filesystem (avec tous les dossiers et outils
Usrfs	User Filesystem (applications spécifiques à l'uti
Buildroot	Ensemble de makefiles et patchs qui simplifient
uClibc	Librairie c de base similaire à glibc mais plus c
Busybox	Binaire unique qui contient toutes les command