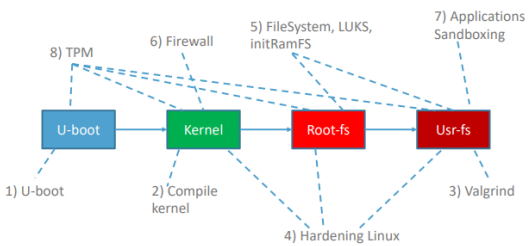
1 Introduction



1.1 Cross-compilation

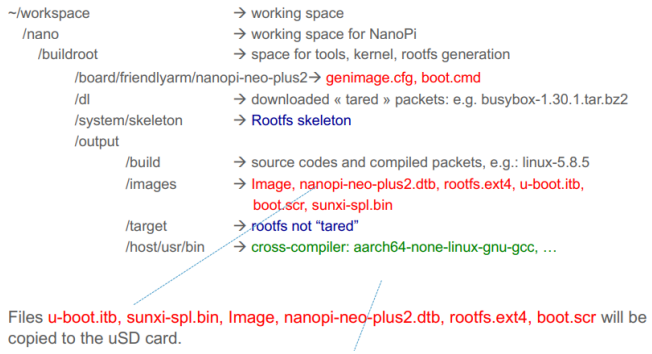
Cross-compilation (ARM) effectuée sur un système x86/x64. Buildroot est le toolchain utilisé. Les éléments suivants sont compilés : Bootloader, Kernel, Rootfs. Puis les images sont copiées sur la carte SD

1.2 Configuration

`<*>` added inside the kernel
`<M>` added as module

2 Buildroot

2.1 Répertoires



— `board/` : contient les fichiers de configuration pour les différentes cartes matérielles prises en charge par Buildroot.
— `package/` : contient les fichiers de configuration pour les différents paquets logiciels qui peuvent être inclus dans l'image de système.
— `toolchain/` : contient les fichiers de configuration pour les différents outils de compilation (comme les compilateurs et les bibliothèques) qui peuvent être utilisés pour construire l'image de système.
— `output/` : contient les fichiers générés lors de la construction de l'image de système, tels que les images d'amorçage, les fichiers système de fichiers, etc.
— `configs/` : contient les fichiers de configuration de base pour les différents systèmes d'exploitation pris en charge par Buildroot.
— `target/` : contient les fichiers générés pour la cible (comme les bibliothèques, les exécutables, les fichiers de configuration, etc.)
— `support/` : contient des scripts et des fichiers de configuration supplémentaires utilisés par Buildroot.

`make` compile les fichiers manquant au dossier `output` (compiler que un paquet avec `make <package>-rebuild`).

2.2 Configuration → Compilation

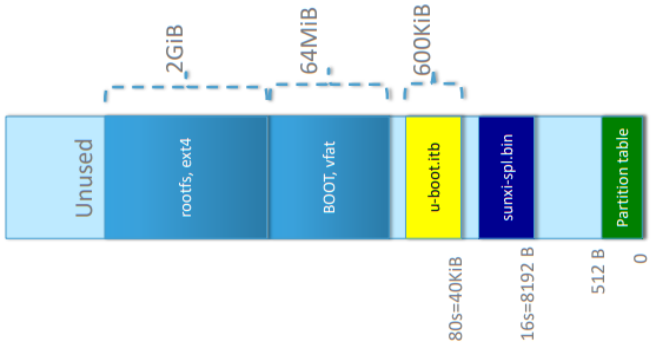
— paramétré avec `make menuconfig`
— Sauvée dans
  — `/buildroot/.config` : full default config file
  — `/buildroot/xxx_defconfig` : stores only the values for options for which the non-default value is chosen

2.2.1 Patch

— `git checkout -b [new feature branch]`
— `git commit -am "Description of modif."`
— `git format-patch [main branch name]`

Une option dans `make menuconfig` permet de définir un dossier dans lequel se trouvent les patchs à appliquer (Build options → global patch directories → p.ex. `/board/friendlyarm/nanopi-.../patches`)

2.3 Carte SD



— `rootfs` : `/bin`, `/sbin`, `/root`, etc.
— `BOOT` : `Image`, `nanopi-neo-plus2.dtb`, `boot.scr`.

`genimage.cfg`→`genimage`→`sdcard.img`→`dd`→ carte SD

Les fichiers pour l'initialisation sont

`rootfs.ext`	Root file system
`Image`	Noyau Linux
`nanopo-neo-plus2.dtb`	Flattened device tree
`boot.scr`	Commandes boot compilées utilisées par u-boot
`boot.vfat`	Partition boot
`u-boot.itb`	Boot loader
`sunxi-spl.bin`	Secondary Program Loader

`boot.vfat` contient `Image`, `nanopi-neo-plus2.dtb` et `boot.scr`. `boot.vfat` (ou `boot.ext4`) permet de créer `BOOT` sur la carte SD

2.3.1 rootfs

Fichier dans un certain format (p.ex. `.ext4`), organisé comme une partition. Contient `/bin`, `/sbin`, `/root`, `/etc`, etc...

2.3.2 rootfs_overlay

Permet de personnaliser un système de fichiers en utilisant des répertoires supplémentaires pour écraser ou ajouter des fichiers au système de fichiers de base généré par Buildroot.

2.3.3 boot.scr

Le fichier `boot.scr` est utilisé par u-boot pour charger le kernel Linux. Il est créé avec la commande `mkimage`

2.3.4 boot.cmd

`boot.cmd` contient des informations de démarrage, notamment les emplacements des différents l'emplacement de `nanopi-neo-plus2.dtb`, du kernel et (si présent) de l'initramfs

2.4 Installer un package

Package se trouvent dans `/buildroot/packages`
Contient :

- `Config.in` file, written in kconfig language, describing the configuration options for the package.
- `foo.mk` makefile, describing where to fetch the source, how to build and install it, etc.
- `Sxx_foo` it is the start script for the foo package.

3 U-boot

3.1 Compilation

Configuration d'u-boot : `make uboot-menuconfig`.

1. `make uboot-rebuild` (verbose : ajout `V=1`)
2. supprimer les fichiers puis `make`

La configuration de u-boot est stockée dans `/buildroot/output/build/[version].config`

3.1.1 Amélioration sécurité

L'option `-fstack-protector-all` (Makefile) ajoute des vérifications contre les buffer overflows (e.g. stack smashing attack).
Concrétement ajout d'une variable de garde (canary). Si modifié lors de l'éxécution d'un morceau de code ⇒ dépassement dans le stack ⇒ appel de la fonction `__stack_chk_fail()`
Peut être statique (valeur fixe) ou dynamique (généré à la volée à chaque exécution de code par une fonction de hachage).

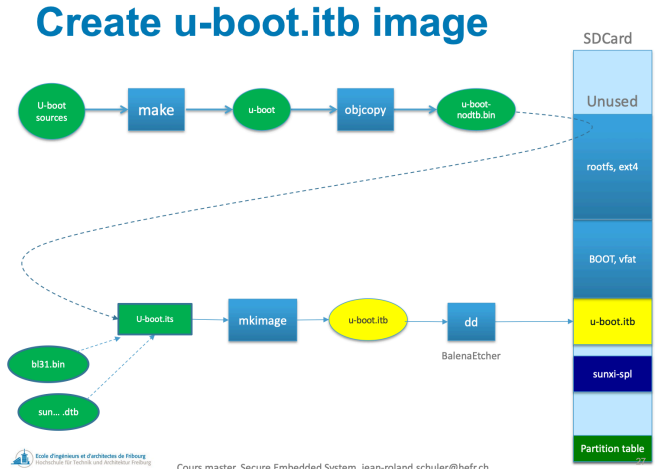
3.1.2 Strip exécutable

`strip` sur un fichier ELF (Executable and Linkable Format) supprime les symboles de débogage et les sections inutiles d'un fichier exécutable.
Cmde complète : `aarch64-linux-strip u-boot`

3.2 Commandes u-boot

`boot`	boot en exécutant le fichier `boot.scr`
`booti`	boot arm64 Linux Image
`ext2load`	load binary file from ext2 Filesystem
`ex2ls`	list files in a directory (default : \)
`fatinfo`	print info about FAT system
`fatload`	load binary file from FAT Filesystem
`printenv`	print environment variables
`mmc`	access memory card MMC/SD

Avec les commandes présentes dans `boot.cmd`, on indique l'emplacement dans la ram de `Image` et `nanopi-neo-plus.dtb`
Lors du démarrage, le Secondary Program Loader (sunxi-spl) va charger le fichier `u-boot.itb`



3.3 FDT (Flattened Device-Tree)

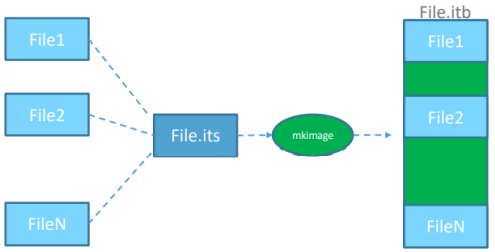
Le FDT contient une description hardware du système utilisée par Linux pour sa configuration (infos sur le port série, le processeur, etc.). le FDT utilise deux fichiers :

- `.dts` : Device Tree Source (fichier ascii)
- `.dtb` : Device Tree Blob (fichier binaire)

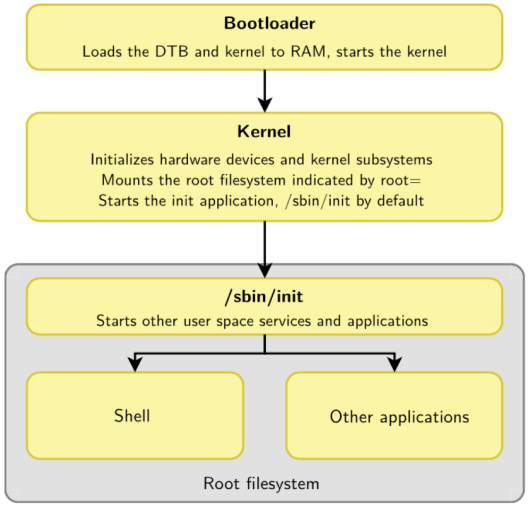
La commande `dtc` permet de passer de `.dts` à `.dtb`.
Le FDT est stocké dans le fichier `u-boot.itb`

3.4 FIT (Flattened Image Tree)

Nouveau format qui permet d'insérer plusieurs fichiers dans un seul :



La commande `mkimage` permet de convertir un fichier `.its` (ascii) en un fichier `.itb` (binaire).

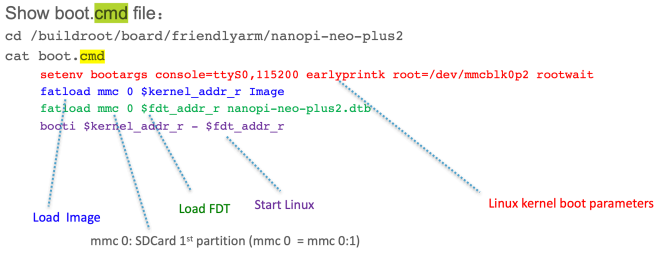


3.5 Séquence de démarrage (6 phases)

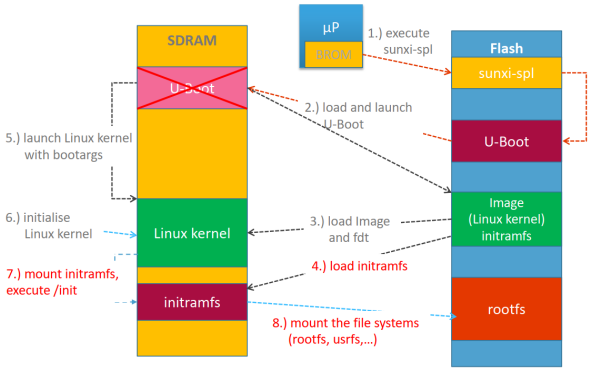
1. Lorsque le μP est mis sous tension, le code stocké dans son BROM va charger dans ses 32KiB de SRAM interne le firmware `sunxi-spl` stocké dans le secteur no 16 de la carte SD / eMMC et l'exécuter.
2. Le firmware `sunxi-spl` (Secondary Program Loader) initialise les couches basses du μP, puis charge l'U-Boot dans la RAM du μP avant de le lancer.
3. L'U-Boot va effectuer les initialisations hardware nécessaires (horloges, contrôleurs, ...) avant de charger l'image non compressées du noyau Linux dans la RAM, le fichier `Image`, ainsi que le fichier de configuration FDT (flattened device tree).
4. L'U-Boot lancera le noyau Linux en lui passant les arguments de boot (bootargs)
5. Le noyau Linux procédera à son initialisation sur la base des bootargs et des éléments de configuration contenus dans le fichier FDT (sun50i-h5-nanopi-neo plus2.dtb).
6. Le noyau Linux attachera les systèmes de fichiers (rootfs, tmpfs, usrfs, ...) et poursuivra son exécution.

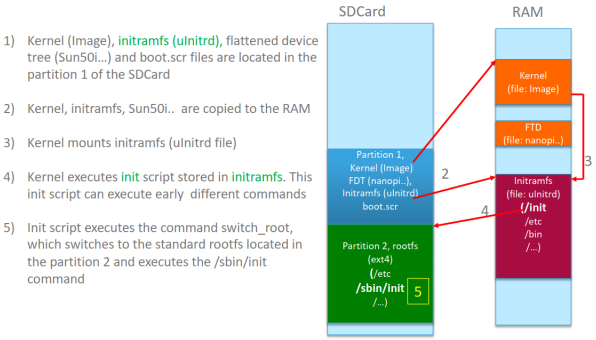
3.5.1 Chargement du Kernel

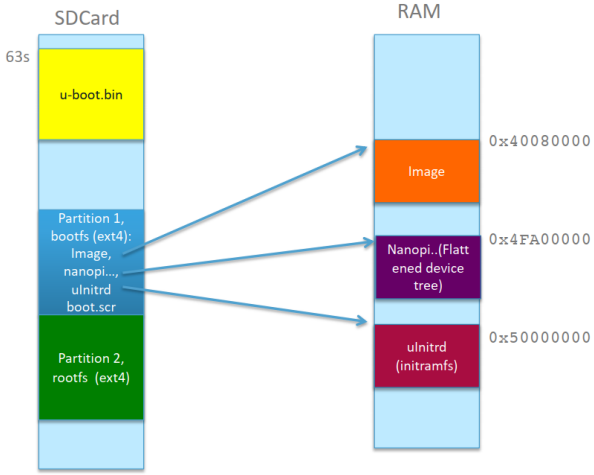
`mkimage` : `boot.cmd` → `boot.scr`



`fatload` charge les images en RAM
`booti` start le kernel en lui donnant l'adresse du kernel et l'adresse du FDT.







4 Kernel

4.1 Compilation

— Configuration : `make linux-menuconfig`
— Compilation : `make linux-rebuild`

4.1.1 Amélioration

Option `-fstack-protector-all` : (idem u-boot) ajout d'un canary.
Option `Randomize_va_space` : permet de placer les éléments à des emplacements mémoire aléatoires (pour éviter d'en cibler un facilement).
Optimisation du kernel pour la place OU pour les performances.
Strip l'assembleur : suppression des symboles non nécessaires (commentaire de debug), pour éviter reverse-engineering du code.
Restriction de l'accès au syslog (system logs).
Mise à 0 lors de l'allocation dynamique sur le tas ou la pile : permet d'éviter à l'attaquant de récupérer des données ou du code.

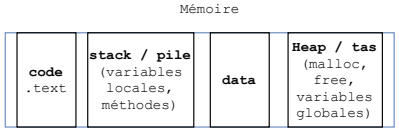
4.2 Busybox

Busybox : logiciel qui regroupe plusieurs outils/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.
— Configuration : `make busybox-menuconfig`
— Compilation : `make busybox-rebuild`

4.3 Réseau

Si le système n'est pas un routeur, on peut choisir de désactiver le routage et le `rp_filter` doit être activé sur toutes les interfaces.

4.4 Attaques



Éxecution sur le stack (buffer overflow) : Insertion de code exécutable dans le stack. Attaque plus possible à présent car le stack est non-exécutable.

ret2libc : Permet de bypasser la non-éxécution du stack. Consiste à éxécuter du code dans une librairie comme libc.

ROP (Return-Oriented Programming) : Éxécution de code malveillant à l'intérieur du programme lui-même. Exploitation des vulnérabilités de sécurité dans le code pour exécuter du code arbitraire en combinant des fragments de code valides déjà présents dans le processus. Au lieu d'injecter du code malveillant dans le processus, on utilise des instructions de retour pour construire une chaîne d'instructions spécifiques.

4.5 Protections

ASLR (Address Space Layout Randomization) : déplacement aléatoire des adresses mémoires (du stack et du heap) à chaque redémarrage du système, il n'est ainsi plus possible de prédire la localisation des instructions placées en mémoire p.ex. avec un buffer overflow. ⇒ évite les attaques `ret2libc`.

PIE (Position Independent Executable) : Rend tout le code exécutable aléatoirement positionné, similaire à ASLR mais agit sur tout l'exécutable.

Canary : Variable qui permet de détecter un dépassement dans le stack. Elle peut être de valeur fixe, ou générée aléatoirement.

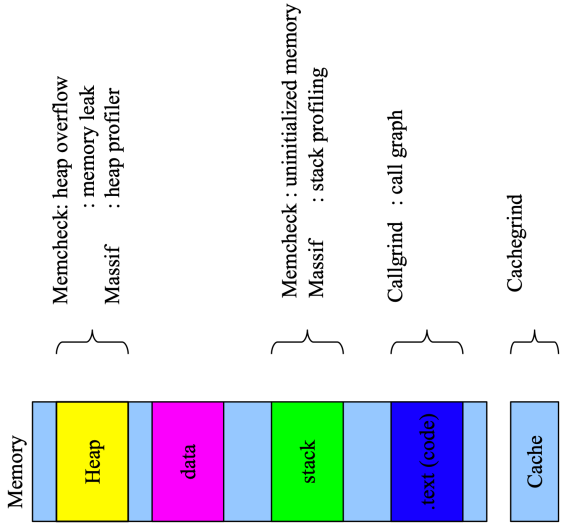
5 Valgrind

Valgrind regroupe des outils d'analyse **dynamique**. L'analyse se fait en exécutant le programme.

5.1 Outils offerts par Valgrind

- Memcheck : Détection d'erreur mémoire (accès à de la mémoire non-allouée, valeurs non initialisées, double free, memcpy, fuites).
- Cachegrind : Profiler de mémoire cache (hit et miss), va aider à faire des programmes ayant un temps d'exécution plus rapide.
- Callgrind : Profiler de cache, à utiliser en complément de Cachegrind.
- Helgrind : Pour la détection d'erreur de thread dans des programmes multithreading. Aide à la programmation pour du multithread. (p.ex. problèmes liés l'absence de mutex.)
- DRD : idem helgrind
- Massif : Profiler de heap et stack (mémoire restante, fuites). Aide à faire des programmes moins gourmands en mémoires. Heap (Tas) grow from the bottom RAM, Stack (Pile) from the top.
- DHAT : Profiler de bloc dans le heap

5.2 Utilisation des outils



6 Hardening

6.1 Intégrité package, programme

Après download d'un package sur un site → contrôler intégrité (état) et authenticité (origine) avec clé public. `gpg` = "Pretty Good Privacy".

```
gpg --verify "package"
gpg --keyserver keyserver.ubuntu.com --
    search-keys "KEY"
```

6.2 Configurer un package, programme

```
tar xvzf package1.tar.gz # unzip
cd package1

# Analyze the different options
less INSTALL or less README
# or
./configure --help
```

6.3 Cross-compiler un programme

```
# Ajout de host et prefix:
./configure --host=aarch64-none-linux-gnu
    --prefix=/home/dir

make
make install
```

Parfois la cross-compilation n'est facilement configurable et il faut éditer directement le Makefile.

6.4 Contrôler les services, les ports ouverts

- `ps -ale` : montre tous les process
- `ps -aux` : montre les droits des process
- `netstat` : affiche les ports TCP/UDP ouverts
- `lsof` : montre les ports ouverts
- `nmap` : scan les ports ouvert liés à une IP

6.5 Permissions des fichiers, dossiers

```
ls -al => -rwxrwxrwx usr grp ..... t.txt
chmod 755 t.txt => -rwxr-xr-x usr grp .....
    t.txt
```

6.6 Sécuriser le réseau

- Désactiver l'IPv6
- Désactiver le routage source IP
- Désactiver le port forwarding
- Bloquer la redirection des msg ICMP
- Activer la vérification de routage source
- Log paquet erroné et ignore bogus ICMP
- Désactiver ICMP echo et temps
- Activer syn cookies (pour TCP)

6.7 Contrôler-sécuriser user

Modifier le umask à 0027 réduit les droits

$$\text{Droit} = \overline{\text{umask}} \ \& \ 0777$$

6.8 Limiter le login root

```
chmod 700 /root #limite l'accès au dossier
    root
sudo #pour avoir les droits root
```

Ne pas mettre le . dans la path

6.9 Sécuriser le noyau

Aller voir dans la section 4.5

6.10 Sécuriser une application

Activer l'option de compilation *−fstack − protector − all* et *noexecstack*

```
gcc -Wall -Wextra -z noexecstack -pie -fPIE
     -fstack-protector-all -Wl,-z,relro,-z,
    now -O -D_FORTIFY_SOURCE=2 -ftrapv -o
    test test.c
```

7 File system

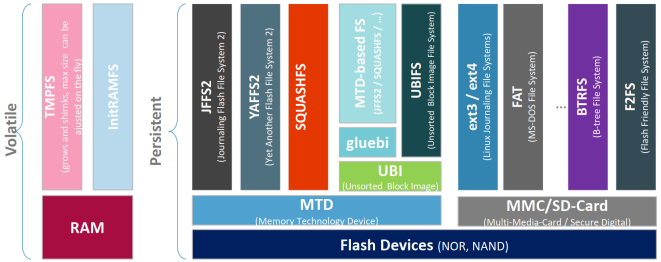
7.1 Systèmes de fichiers

Pour les systèmes embarqués, il existe deux catégories de systèmes de fichiers :

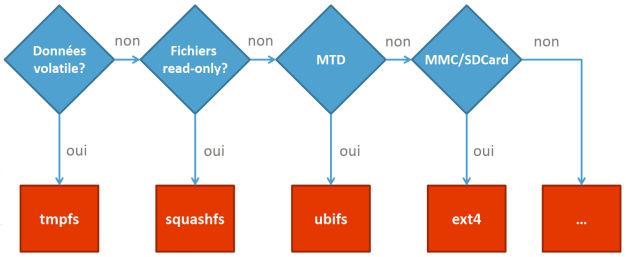
1. Volatiles (RAM)
2. Persitants (Flash NOR et de plus en plus NAND)

Deux technologies principales sont disponible sur les Flash :

— MTD (Memory Technology Device)
— MMC/SD-Card (Multi-Media-Card/Secure Digital Card)



7.1.1 Choix d'un FS



7.1.2 MMC technologies

MMC/eMMC/SD Card composés de 3 éléments :

— MMC interface : Gère la communication avec l'hôte
— FTL (Flash translation layer)
— Zone de stockage (table de NAND)

FTL Petit contrôleur qui fait tourner un firmware qui transforme l'adresse secteur logique en adresse NAND.

7.2 Architecture des FS

7.2.1 Journalisation

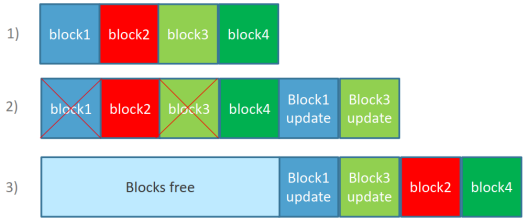
Qui garde une trace de chaque modification dans un journal.
Permet la restauration de fichiers corrompus comme les fichiers sont d'abord écrite dans le journal avant d'être écrite sur le disque.

7.2.2 B-Tree et CoW (Copy-on-Write)

Architecture de stockage en arbre. Gère efficacement des grandes quantités de données en utilisant des nœuds qui contiennent plusieurs clés et valeurs, plutôt que des nœuds individuels.
La stratégie de CoW consiste à ne pas copier immédiatement les données lorsqu'une modification est apportée, mais copie que si cela s'avère nécessaire.

7.2.3 Log FS

Utilisation du support de stockage comme tampon circulaire ⇒ les nouveaux blocs sont toujours écrits jusqu'à la fin.



1. Etat initial
2. Modification des blocs 1 et 3
3. Copie des blocs 2 et 4 dans une nouvelle zone avec les blocs 1 et 3 modifiés.

7.3 Systèmes de fichiers alternatifs

BTRFS : système récent (2007, stable 2014), système B-Tree, potentiellement meilleur que ext4 (selon développeur principal de ext4).

F2FS (Flash-Friendly File System) : log FS, opérations atomiques (exécuté sans interruption), défragmentation, support du TRIM (informe qu'un bloc de données n'est plus utilisé et peut être supprimé).

XFS : journalisé, support de très gros FS, designé pour être extensible, il ne semble cependant pas bien gérer les pertes de puissance (état de veille).

NILFS2 : Log FS et B-Tree, Userspace garbage collector.

ZFS : B-Tree, support de FS volumineux, pas très adapté à l'embarqué (utilise RAM).

7.4 Ext2-3-4

Ext2 : non journalisé, utilise le block mapping pour réduire la fragmentation

Ext3 : (2001), journalisé, prévient la perte de donnée même en cas de d'extinction brutale du système contrairement à Ext2, rétro-compatible vec ext2.

Ext4 : (2008), actuellement meilleur FS pour l'embarqué utilisant MMC, rétro-compatible avec ext3 et ext4, supporte système de fichiers volumineux, utilisation des extents.
Les extents permettent de regrouper plusieurs blocs consécutifs de données en un seul objet appelé "extent", ce qui permet d'optimiser les performances d'écriture et de lecture. les extents permettent également de réduire le nombre de métadonnées nécessaires pour stocker les informations sur l'emplacement des données d'un fichier, ce qui permet d'économiser de l'espace disque et d'améliorer les performances.

7.5 SquashFS

Linux FS compressé, lecture seule.
Destiné à une utilisation générale en lecture seule, à une utilisation archivistique et dans les systèmes embarqués avec de petits processeurs où de faibles charges sont nécessaires.

7.6 Tmpfs

Garde tous les fichiers en mémoire virtuelle. Tout dans tmpfs est temporaire. Si une instance tmpfs est démontée, tout ce qui y est stocké est perdu. Tmpfs est un système de fichier en mémoire qui permet de stocker des fichiers temporaires qui ont besoin d'un accès rapide aux données, d'utiliser l'espace libre de la mémoire vive, d'être sécurisé pour les données sensibles et d'être flexible pour stocker différents types de fichiers temporaires.

7.6.1 Devtmpfs

Devtmpfs est un système de fichiers qui remplit automatiquement les fichiers de nœuds (/dev) connus du noyau.

7.7 LUKS (Linux Unified Key Setup)

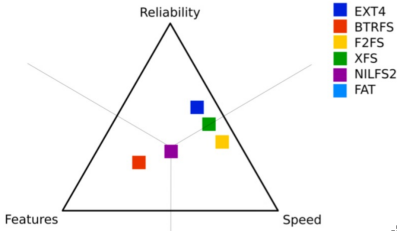
Système de chiffrement de disque utilisé pour chiffrer les partitions de disque sur les systèmes Linux. Protège les données sur le disque en utilisant une clé de chiffrement qui est utilisée pour chiffrer les données sur le disque. Lorsqu'un utilisateur veut accéder à des données chiffrées sur le disque, il doit d'abord entrer la clé de chiffrement pour déchiffrer les données.

7.8 Conclusion

Performances :
- EXT4 : meilleure solution pour les systèmes embarqués utilisant MMC
- F2FS et NILFS2 hautes performances en écriture

Features :
- BTRFS FS de prochaine génération
- NILFS2 Fournit des fonctionnalités plus simples mais similaires

Scalability :
- EXT4 n'évolue clairement pas aussi bien que BTRFS et F2FS



8 File System Security

8.1 Files permissions

A chaque utilisateur est assigné un user ID (UID). Chaque utilisateur peut être membre d'un ou plusieurs groupes (désigné par des groupes ID - GID).
La list des utilisateurs peut être consultées dans le fichier `/etc/passwd`.
La liste des groupes est disponible sur le fichier `/etc/group`.
Les mots de passes se trouvent dans le fichier `/etc/shadow`.

Access type	File	Directory
Read	Read the file content	Allow to read the folder content
Write	Modify (create-delete-rename) the file	Allow to create-delete-rename files in the folder
Execute	Execute the file	Allow to go in the folder

8.2 Real-effective userID and groupID

Chaque processus possède un effective UID et un real UID, idem pour les GID.
Linux utilise seulement le effective userID. Si le bit userID est actif alors le fichier exécuté prend les droits du propriétaire du fichier.

8.3 ACL - Access Control List

Les filesystems ext3,ext4,tmpfs,btrfs autorise les ACL u : : User, g : : Group, o : : Other

```
setfacl -m u::rwx,g::r--,o:--- test
setfacl -Rm u:user1:rw TestDirectory # -R:
    Recursive
# remove
setfacl -b test #The file test has no ACL
setfacl -x u:user1,g:group1 test #The file
    test has no rights for user1
and group1
```

8.4 Attributs particuliers des FS ext2-3-4

On peut utiliser la commande `lsattr` ou `chattr`

```
chattr +i file #add i attribute
chattr -i file #del i attribute
chattr =i file #equal i attribute
lsattr file
```

- -i :file/directory can not be modified, deleted, renamed or linked symbolically, not even by root. Only root or a binary with the necessary rights can set this attribute.
- -A :Date of last access is not updated (only useful for reducing disk access)
- -S :File is synchronous, the records in the file are made immediately on the disc. (equivalent to the sync option of mount)
- -a :File can only be open in append mode for writing (log files, etc.) Only redirection >> can be used, the file can not be deleted.
- -c :File is automatically compressed before writing to disk, and unpacked before playback
- -d :File will not be saved by the dump
- -j :Ext3-ext4 :A file with the 'j' attribute has all of its data written to the ext3 or ext4 journal before being written to the file itself
- -s :When the file is destroyed, all data blocks are being released to zero.

8.5 Recherche de permissions faibles

```
find . -perm 200 #file permissions = 200
find . -perm -220 #write bit for user and
    group = 1
find . -perm /220 #write bit for user or
    group = 1
find . -perm +220 #write bit for user or
    group = 1 (like /220)
```

8.6 Sécuriser les répertoires temporaires

- mettre le `\tmp` dans une autre partition
- no suid programme sont permis
- rien ne peut être exécuté
- aucun node file existe dans le `\tmp`

8.7 Mémorisation des mots de passes

Les mot de passes sont sotckés en HASH (preuve sans connaissance) dans le fichier `/etc/shadow`.

9 Hashcat

5 méthodes de craquage de mot de passe

Attaque brute force Test de toutes les possibilités.

Attaque par dictionnaire Test tous les mots à partir d'un dictionnaire.

Attaque combinatoire Test toutes les combinaisons possible à partir de deux dictionnaires.

Attaque hybride Combinaison de mots à partir d'un dictionnaire et ajoute un suffixe composé de chiffres et/ou lettres.

Attaque par masque Brute force assistée avec des indices (longeur du mot de passe, connaissance d'un ou plusieurs caractères, jeux de caractères utilisés, etc.).

10 Firewall `iptables`

`iptables -t table -COMMAND chain ... -j TARGET`
Le kernel doit être configuré pour activer netfilter.
Un hook est une étape lors du passage d'une trame dans le stack de protocoles. Le framework netfilter sera appelé à chaque hook (combinaison chain-table).

10.1 Chains

Quand un paquet arrive il traverse différentes chains contenues dans différentes tables :

- INPUT : le paquet est pour l'hôte local
- OUTPUT : le paquet est émis par l'hôte local
- FORWARD : le paquet arrive sur une interface (eth0) et est transmis à une autre (eth1).
- PREROUTING : pour modifier paquets dès réception. (NAT)
- POSTROUTING : pour modifier paquets juste avant émission. (NAT)

10.1.1 Tables

- filter : principalement utilisée (default).
- mangle : alternation-modification particulière sur des paquets.
- nat : consultée lorsqu'un paquet créé une nouvelle connexion (Network Address Translation).

10.2 Features

1. **Stateless packet** firewall (table filter et ACCEPT, DROP, REJECT). Permet de protéger au niveau réseau (bloquage d'une ip, d'un port, etc.). → paquets analysés de manière individuelle.
2. **Stateful** firewall. Permet de protéger au niveau du paquet en fonction du contexte (précédents paquets). Il est possible d'accepter des paquets venant de l'extérieur seulement s'ils sont des réponses à des requêtes venant de l'intérieur.
   - Utilisation de connection tables pour traiter les différentes parties des protocoles.
   - NEW : Nouveau paquet qui n'est pas lié à une connexion active
   - ESTABLISHED : Une connexion passe de NEW à ESTABLISHED losrque la connexion est validée par la direction op-

posée
— RELATED : Paquets qui ne font pas partie d'une connexion existante mais qui sont liés à une autre. (Par exemple réponses ICMP pour une communication FTP).

3. Translation d'adresses / ports (NAT)
4. API pour autres applications

10.3 NFQUEUE

NFQUEUE permet de transmettre un paquet au userspace (hors du noyau).

10.4 knockd

Réside dans le user space. Configuration dynamique du firewall Netfilter avec des commandes `iptables`. Par exemple ouvertures de port spécifiques lorsque des séquences de *Port Knocking* (PK) sont reconnues.

10.5 fwknop (FireWall KNock OPerator)

Identique que `knockd` mais il utilise le contenu des paquets TCP/UDP (frame SPA - Single Packet Authorization). Si un SPA est valide, le pare-feu est ouvert pendant un certain temps (par exemple 30s).

11 TPM

11.1 Chiffrements

Une fonction de hachage H prend une entrée m de longueur variable et retourne un string de taille fixe h (valeur hachée).

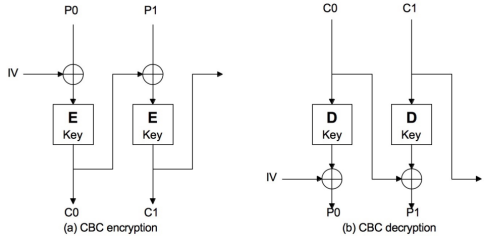
$$h = H(m)$$

Propriétés

— $H(m)$ est rapide à calculer
— H est à sens unique
— H est "collision-free"

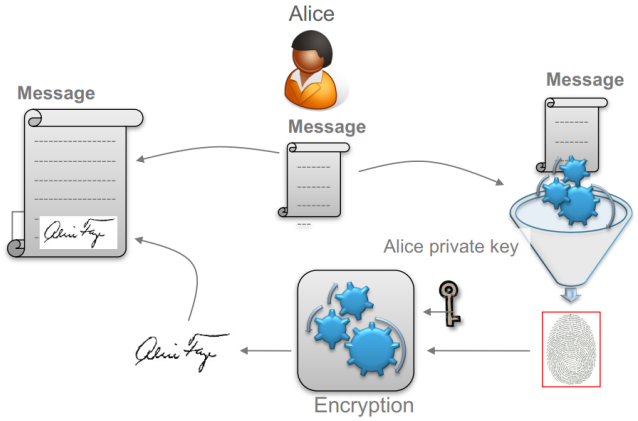
11.1.1 Symétrique

— Même clé permet le chiffrement et déchiffrement
— Chiffrement par bloc, éventuellemnt chaîné (CBC - Cipher Block Chaining). IV (Initial Vector) non secret.



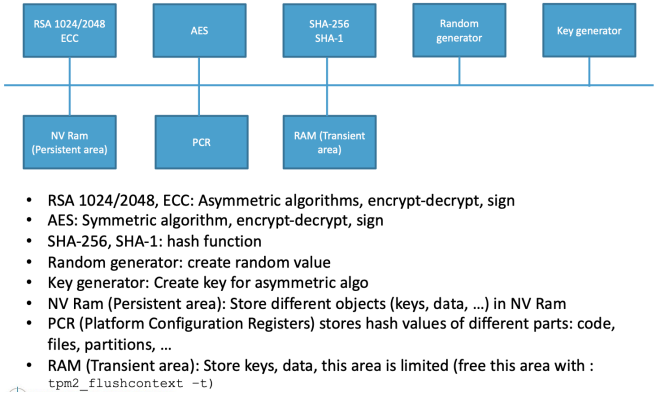
11.1.2 Asymétrique

— Deux clés (publiques et privées) pour chaque partie, clé publique disponible par des certificats.
— Encrypt public → Decrypt private ⇒ **confidentialité**, intégrité.
— Encrypt private → Decrypt public ⇒ **authenticité** (signature numérique), intégrité.



11.2 TPM (Trusted Platform Module)

⇒ coprocesseur cryptographique.



— Discret : Circuit dédié (tamperproof)
— Intégré : Partie du μC qui gère le TPM
— Hyperviseur : fournis par personnes fiables

Peut aussi être software

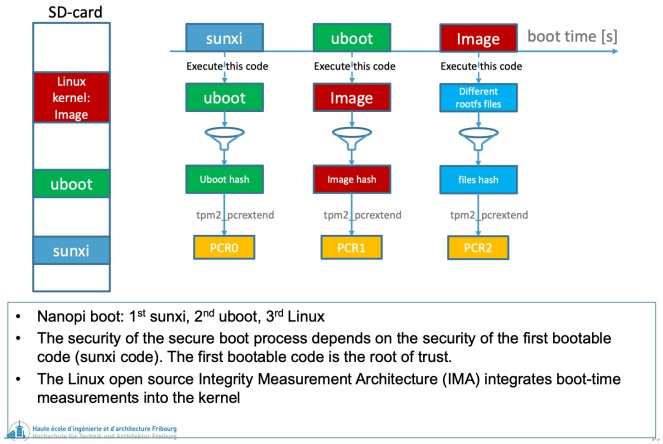
11.2.1 Hiérarchies

Stockage des clés

— 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 éphémères (RAM s'efface à chaque redémarrage)

11.3 Platform Configuration Registers (PCR)

The prime use case is to provide a method to cryptographically record (measure) software state or configuration data used by a device.



12 Autres

12.1 Commandes

netcat (`nc`) : Couteau suisse du TCP/IP. Permet de scanner des ports

nmap : Analyse des ports ouverts

ssh : Connexion à un système par interpréteur de commande

dd : copie byte à byte entre des streams. (`sudo dd if=/dev/zero/ of=/dev/null bs=512 count=100 seek=16`)

parted : création / modification de partiations (`sudo parted /dev/sdb mklabel msdos`

mkfs.ext4 : commandes ext4 pour créer / modifier une partition

12.2 Définitions

Honeypot : "Pot de miel" ou leurre pour faire croire qu'un système non-sécurisé est présent (à tord)

Toolchain : Codes sources et outils nécessaires pour générer une image éxécutable (sur un système embarqué)

Kernel : Coeur Linux (avec le format u-boot)

Rootfs : Root Filesystem (avec tous les dossiers et outils utilisés par Linux)

Usrfs : User Filesystem (applications spécifiques à l'utilisation du système embarqué)

Buildroot : Ensemble de makefiles et patchs qui simplifient et automatisent la création d'un Linux pour système embarqué

uClibc : Librairie c de base similaire à glibc mais plus compacte (pour systèmes MMU-less)

Busybox : Binaire unique qui contient toutes les commandes de base (ls, cat, mv)

12.2.1 Sauver config pour prochaine installation

```
rsync -a /buildroot/board/...
      /workspace/config/board
cp /buildroot/configs/... workspace/configs/...
```