# Sécurité matérielle des systèmes et des données
# Vers une phase de boot sécurisée

Martin Peres

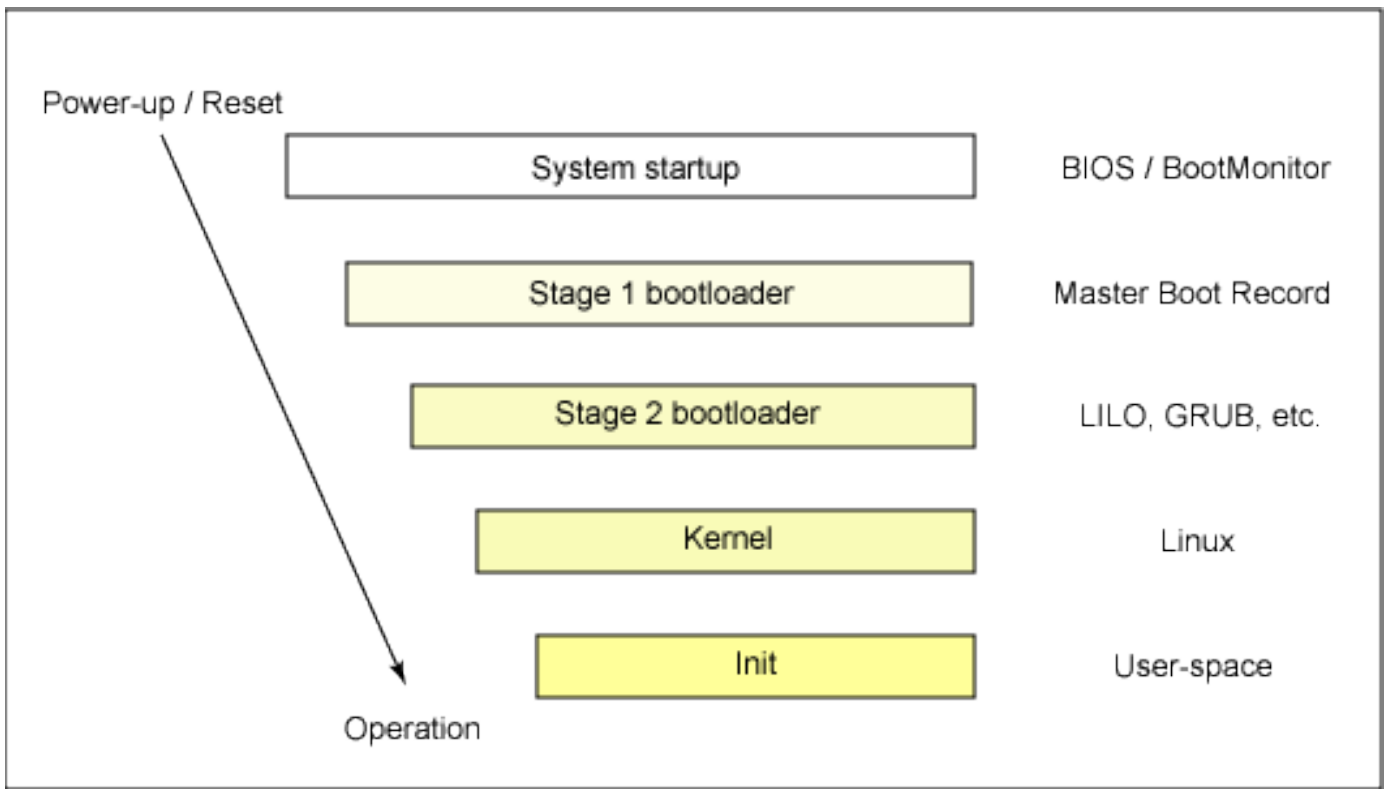
Doctorant de Francine Krief au LaBRI

October 11, 2013

# Sommaire

# La procédure de boot

- Matériel/BIOS: power-on self test (POST)
- Bootloader: Sélection de l'OS
- Kernel: Démarrage du noyau
- User-space: Démarrage du système d'exploitation



## Le Basic Input Output System (BIOS)

- permet de faire abstraction du hw pour les fonctions de base
- est physiquement stocké sur la carte mère, dans une EEPROM ou une mémoire flash
- peut être "flashé"
- devrait plutôt être appelé firmware
- est en cours de remplacement par l'UEFI

## Démarage

Le BIOS est responsable:

- du power-on self test: Initialisation du matériel avec exécution du "BIOS" qui leur est associé
- d'abstraire le hw en proposant une interface appelable par des interruptions logicielles

## Rôle du bootloader

Le bootloader fait la transition entre le POST et le système d'exploitation. Il est composé de 2 étages (stages):

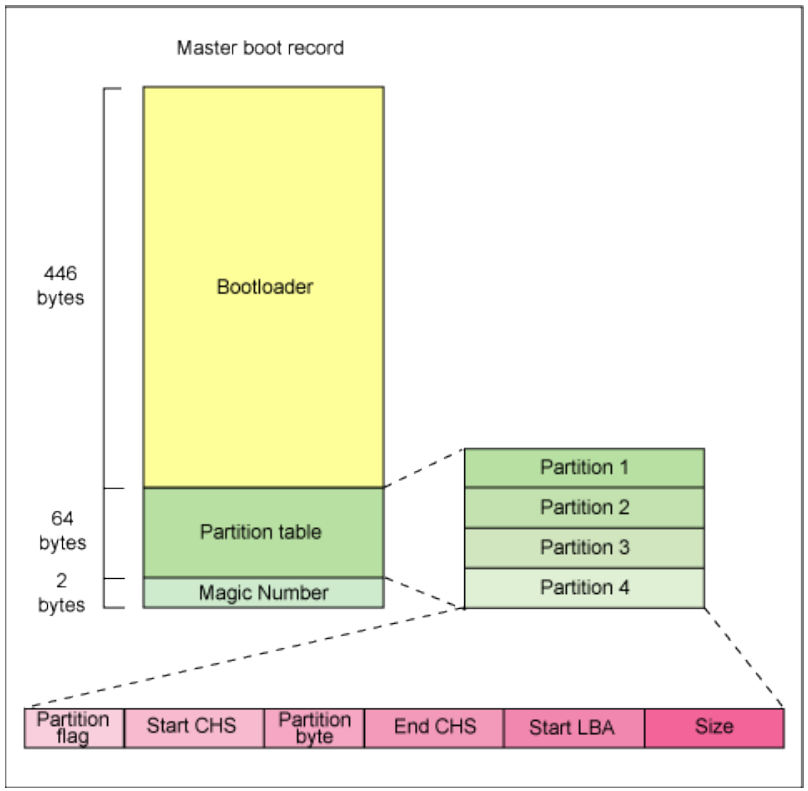
- Stage 1: Code situé sur le Master Boot Record, charge le stage 2
- Stage 2: Code situé sur une partition, permet de charger le noyau

## Exemple de bootloaders

- LiLo: Linux Loader
- GRUB: GRand Unified Bootloader

## Master Boot Record

- Les 512 premiers octets d'un disque dur
- contient la table de partition ainsi que le code pour charger le stage 2



## Grub: Exemple de configuration (menu.lst)

```
# general configuration:
timeout   5
default   0
color light-blue/black light-cyan/blue

# (0) Arch Linux
title  Arch Linux
root   (hd0,4)
kernel /boot/vmlinuz-linux root=/dev/sda7 ro quiet
initrd /boot/initramfs-linux.img

# (1) Windows
title Windows 7
rootnoverify (hd0,1)
makeactive
chainloader +1
```

## Rôle du noyau dans le boot

- Initialise le CPU (passage du mode 16 bit à 32 ou 64)
- puis le matériel avec les pilotes
- pour finalement lancer le processus d'init spécifié en paramètre (par défaut: init=/sbin/init)

## Rôle du processus d'init

- Sélectionne la bonne locale (langue), layout clavier et timezone
- Monte les partitions, charge les modules matériels non-essentiels via udevd
- Lance les daemons (services en tâche de fond)
- Devient le père des processus dont leur père est mort

## Il existe différents gestionnaires de démarrage

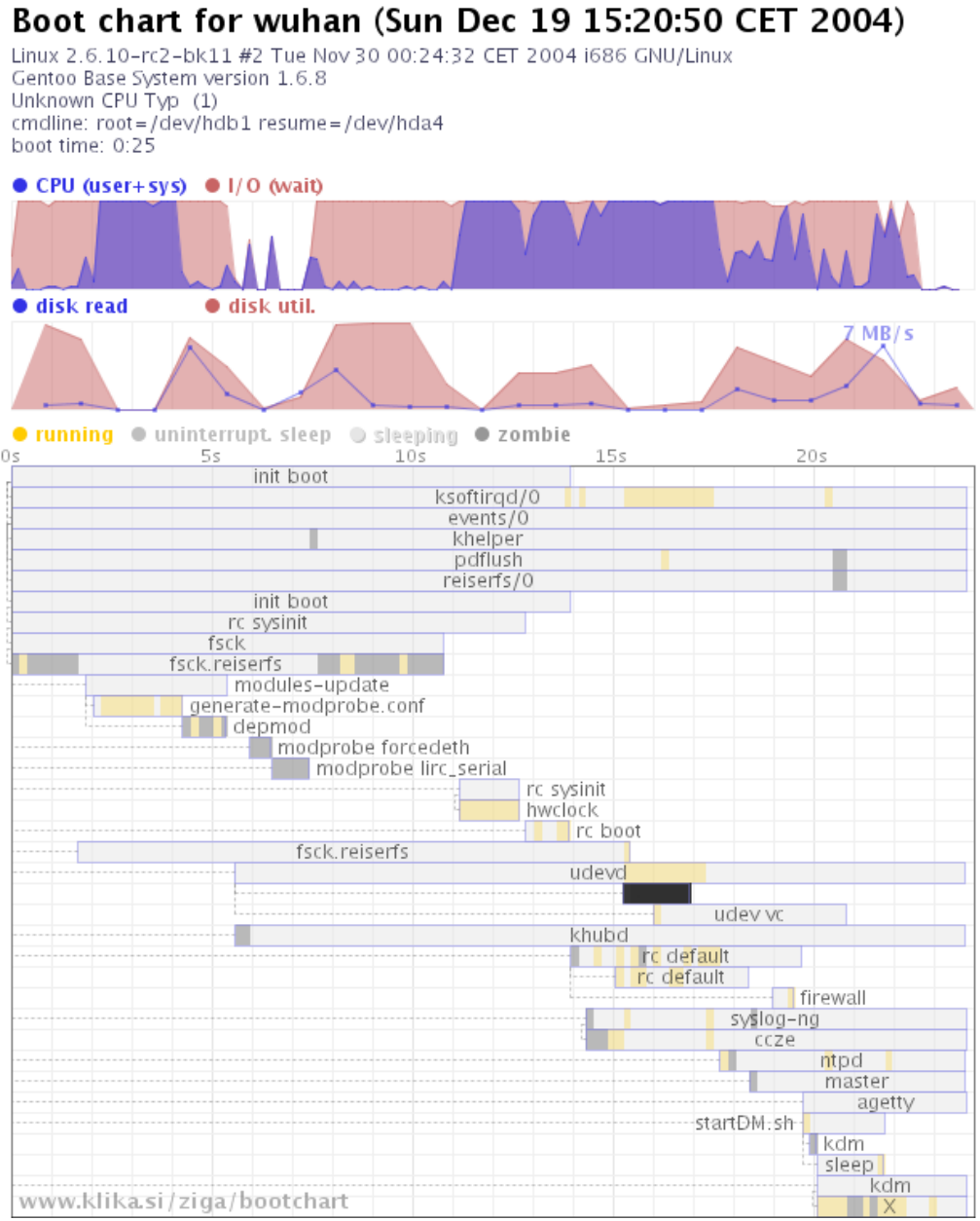
- Systemd: Utilisé par presque toutes les distributions
- Upstart: Utilisé par Ubuntu
- Linux System V Init: Le classique, utilisé par Debian et BSD
- Et d'autres...

## Fonctionnement: Les runlevels (Exemple pour Archlinux)

- 0: Mode arrêt
- 1: Mode single user/maintenance
- 3: Multi utilisateur
- 5: Mode graphique (potentiellement multi-user)
- 6: Mode redémarrage

## Utilisation

- Configuration: /etc/inittab
- Runlevel courant et précédent: $ runlevel
- Changement: $ init X (X = [0-6])
- Spécification du runlevel voulu au démarrage: mettre le numéro dans la ligne de commande du kernel. 3 par défaut.


Boot chart for wuhan (Sun Dec 19 15:20:50 CET 2004)
Linux 2.6.10-rc2-bk11 #2 Tue Nov 30 00:24:32 CET 2004 i686 GNU/Linux
Gentoo Base System version 1.6.8
Unknown CPU Typ (1)
cmdline: root=/dev/hdb1 resume=/dev/hda4
boot time: 0:25
CPU (user+sys)
I/O (wait)
disk read
disk util.
7 MB/s
running
uninterrupt. sleep
sleeping
zombie
0s
5s
10s
15s
20s
init boot
ksoftirqd/0
events/0
khelper
pdflush
reiserfs/0
init boot
rc sysinit
fsck
fsck.reiserfs
modules-update
generate-modprobe.conf
depmod
modprobe forcedeth
modprobe lirc_serial
rc sysinit
hwclock
rc boot
fsck.reiserfs
udevd
udev vc
khubd
rc default
rc default
firewall
syslog-ng
ccze
ntpd
master
agetty
startDM.sh
kdm
sleep
kdm
X
www.klika.si/ziga/bootchart

# Sommaire

## Contexte

- Un attaquant a réussi à faire exécuter du code à lui sur une machine (exploitation de buffer overflow?)
- Il veut avoir plus de privilèges, devenir root (privilege escalation)
- Il veut se garder un accès pour retourner plus facilement sur la machine (backdoor)

## privilege escalation

- Profiter d'une faille kernel
- Profiter d'une faille d'un logiciel tournant avec les droits root
- Modifier la variable d'environnement PATH pour abuser l'utilisateur

## TD: Installation d'une backdoor

- echo "PATH=/home/user/.secret:$PATH" >> .bashrc
- ls .secret/
  su sudo ls
- Quelle est la vulnérabilité et les scénarios possibles?

## Attaques off-line

L'ajout d'une backdoor peut se faire:

- en sortant le disque dur de la machine à attaquer puis en changeant les binaires
- en modifiant les données depuis un livecd ou liveusb

# Sommaire

## Solution classique aux problèmes trouvés

Politique MAC:

- Exécution seulement depuis /usr/bin et /bin
- Seul l'utilitaire de mise à jour à le droit de modifier /usr et /bin
- Limiter les droits des applications privilégiées
- Utiliser un ordinateur distant pour vérifier les signatures md5 des programmes installés

## Limites de la solution

- Ne protège pas des failles dans le kernel ou du matériel
- Ne protège pas entre 2 vérifications des md5
- On peut attaquer ce système en modifiant le kernel/bios puis attendre le prochain reboot
- Ne protège pas des attaques off-line
- $--> $ Il faut protéger la phase de boot

## Protéger la phase de boot

- Empêcher le boot depuis des medias amovibles (configuration du BIOS) + mettre un mot de passe sur le BIOS
- Empêcher les modifications de la configuration du bootloader
- Chiffrer le disque dur

## Limites de la solution

- Aucune validation à l'exécution des binaires exécutés
- Il y a toujours une possibilité qu'un attaquant ait réussi à corrompre le système
- $--> $ Il faut établir une chaîne de confiance au boot pour avertir l'utilisateur

# Sommaire

## Trusted Platform Module (TPM)

- Composant présent sur certaines cartes mères récentes
- Connecté au chipset par le bus LPC (Low Pin Count)
- Peut être comparé à une carte à puce
- N'est pas protégé contre les attaques physiques lourdes (analyse des courants, reset forcé)

## Un TPM sait

- Générer des nombres aléatoires et des clés RSA (2048 bit max)
- Faire des hashs cryptographiques SHA-1
- Authentifier un message
- Chiffrer, déchiffrer et signer en utilisant RSA

## Services proposés par un TPM

- Mesure d'intégrité: cf suite du cours
- Scellement de données: permet l'accès à des données seulement dans un certain contexte
- Attestation: Permet de prouver à un tiers l'état actuel de la machine

## Mesure d'intégrité

- Le TPM contient plusieurs registres de 160 bits (PCR)
- Un registre peut être lu ou étendu pour une mesure
- Mesure = SHA-1(données)
- PCR[i] = SHA-1(PCR[i] — Mesure)
- $--> $ Static Root of Trust for Measurement (SRTM)

## Scellement de données

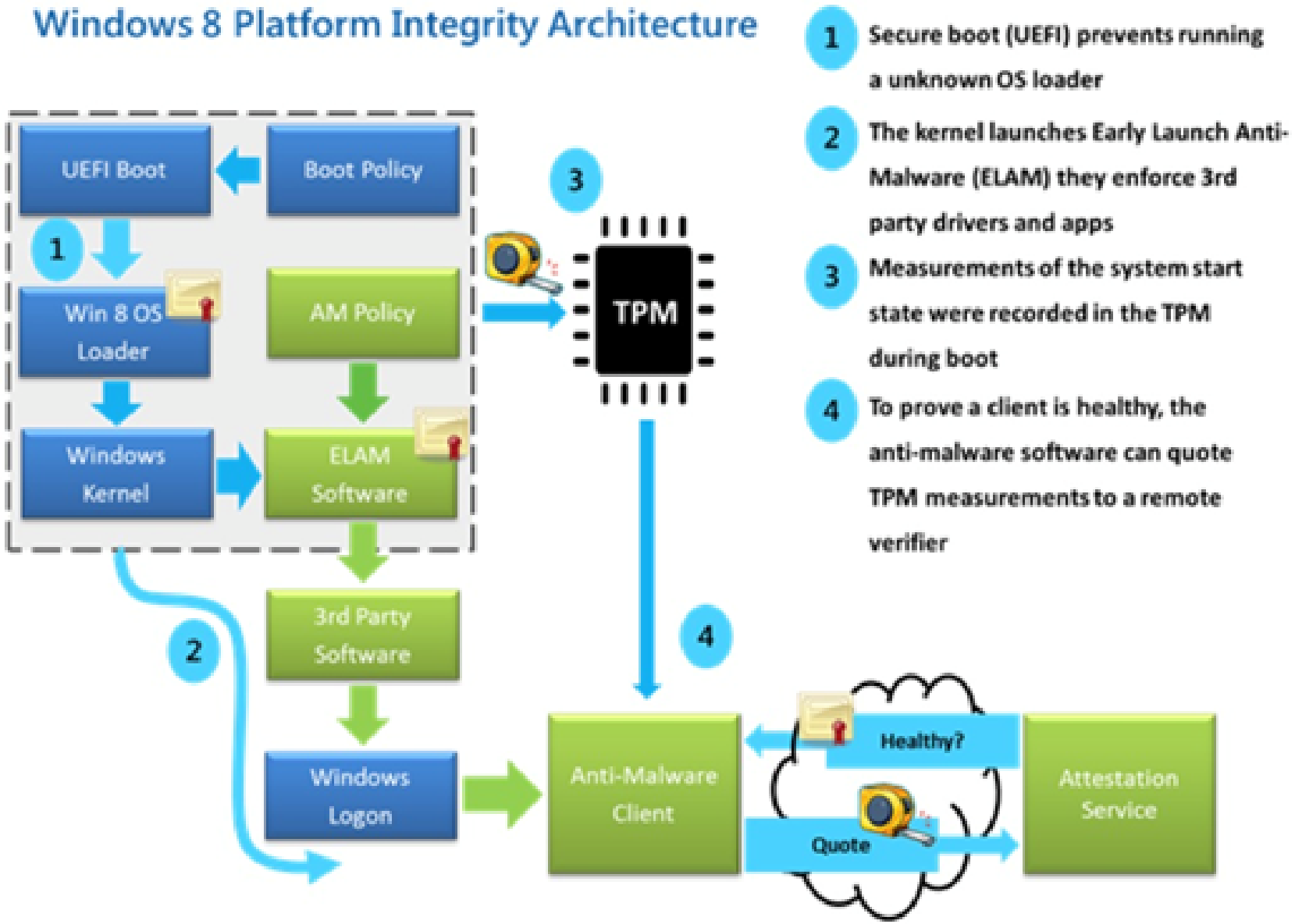
- On peut stocker des données qui seront liées à l'état de certains PCR
- Permet de rendre des données accessibles seulement à certains moments

## L'attestation d'état

- Le TPM signe le contenu des registres PCR
- Cette signature peut ensuite être transmise et vérifiée à distance
- Controverse: Peut être utilisée pour restreindre la lecture de vidéos sur certaines plateformes

## Example d'applications

- Trusted Grub
- Windows 8




Source: http://resources.infosecinstitute.com/uefi-and-tpm/

## Limites de la SRTM

Pour fonctionner, la SRTM doit vérifier tout code exécuté depuis le démarrage, cependant:

- Après le démarrage, l'ordre des applications lancées n'est pas connu
- Toutes les applications ne sont pas forcément connues du TPM
- $--> $ La SRTM permet de garantir la phase de uniquement démarrage

## Intel Trusted eXecution Technology (TXT)

- Permet la mesure d'intégrité (MLE) après l'exécution (Late Launch)
- Peut se lancer depuis le ring 0 avec les instructions GETSEC[SENTER—SEXIT]
- La mesure affecte les PCR 17(SINIT), 18(MLE) et 19(noyau)
- $--> $ On peut donc sceller des données qui nécessiteront un noyau ainsi qu'une procédure de validation connue
- $--> $ On appelle ça la Dynamic Root of Trust for Measurement (DRTM)

## Limites de la DRTM

- Nécessite un processeur supportant TXT, un TPM, un chipset compatible (avec code SINIT associé), un BIOS compatible (VT-x, VT-d, TPM et TXT)
- La procédure de vérification ne doit pas pouvoir être modifiée (adresse non-mappable)
- $--> $ La DRTM est encore très jeune et peu/pas utilisée

## Limites à l'utilisation d'un TPM

- Il peut être ré-initialisé (même si c'est une attaque visible)
- La procédure de validation de l'OS n'est pas encore standardisée et pose des problèmes pour ceux qui compilent leur propre système
- Les Platform-Keys (PF) peuvent ne pas être modifiables par les utilisateurs
- Il est possible d'analyser le fonctionnement du TPM en analysant le courant consommé pour retrouver les clés de chiffrement ou simplement récupérer sur le bus les clés
- $--> $ Accès physique prolongé = pas de sécurité possible

## Solution pour l'open source si PK modifiables

Si changement des clés possibles, alors il est possible de démarrer Linux directement! Plus besoin de bootloader!
`http://www.kroah.com/log/blog/2013/09/02/booting-a-self-signed-linux-kernel/`

## Solution pour l'open source si PK non-modifiables

La Linux Foundation a écrit un bootloader authentifié par Microsoft qui charge n'importe quel kernel sans authentification. Cela permet la compatibilité avec les vieux systèmes mais enlève la sécurité du trusted boot.
`http://www.linuxfoundation.org/news-media/blogs/browse/2012/10/linux-foundation-uefi-secure-boot-system-open-source`

# Sommaire

## Conclusion et perspectives

- La sécurité d'un système dépend de sa sécurité physique, de sa phase de boot puis des politiques de sécurité appliquées aux processus s'exécutant sur le système
- Plus on descend vers le hardware, plus la sécurité devient difficile à auditer et plus l'implémentation est laissée aux OEM (qui sont la plupart du temps douteux)
- Cette sécurité peut sembler abusive pour les ordinateurs fixes mais devient importante sur un terminal mobile (téléphone, laptop)
- $--> $ Accès physique prolongé $=$ pas de sécurité possible