UEFI et *bootkits PCI* : le danger vient d'en bas

Pierre Chifflier

6 juin 2013

Article

Plus de détails dans l'article

- Séquence de démarrage
- Fonctions UEFI et utilisations :
  - Interception du *bootloader*
  - Tables ACPI
  - Fonctions réseau
  - ...
- Cartes PCI
- Contre-mesures

PCI *bootkits*

Bootkits PCI

- *Bootkit* : *bootloader rootkit*
- Modifications visibles
- Difficile si mot de passe
- Utilisation du matériel ?

Exemple : carte graphique

Objectif : élévation de privilèges
Profil d'attaquant : motivé

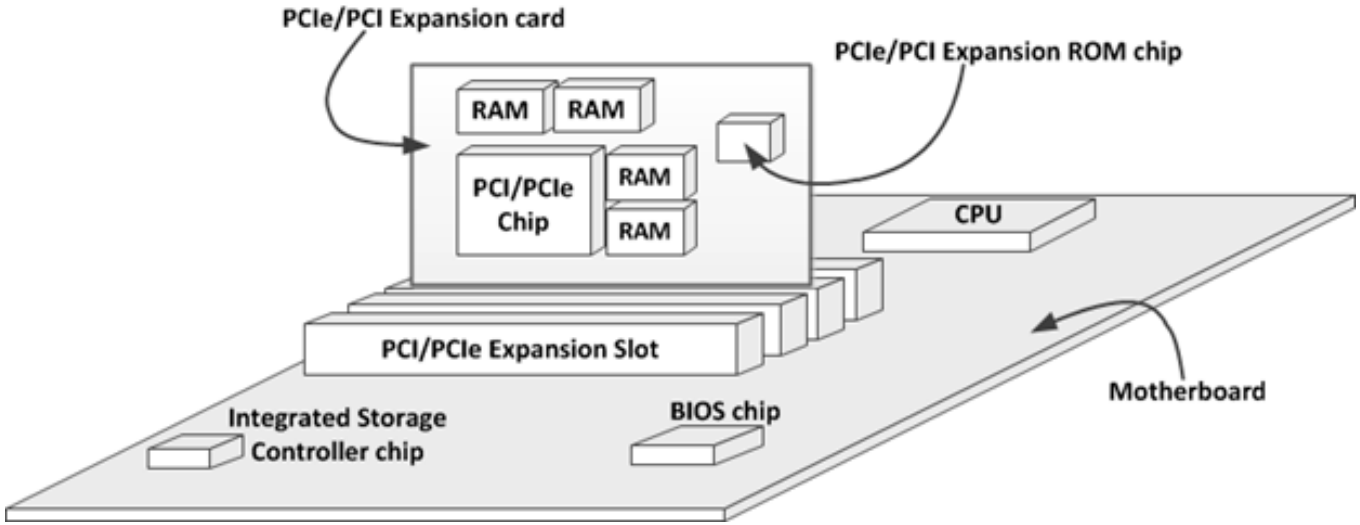
Problèmes

- OS ? Pas en mémoire
- Pas d'accès au disque (+ chiffrement possible)
- Exécution de code ?
- En quelques ko !

Réactions initiales

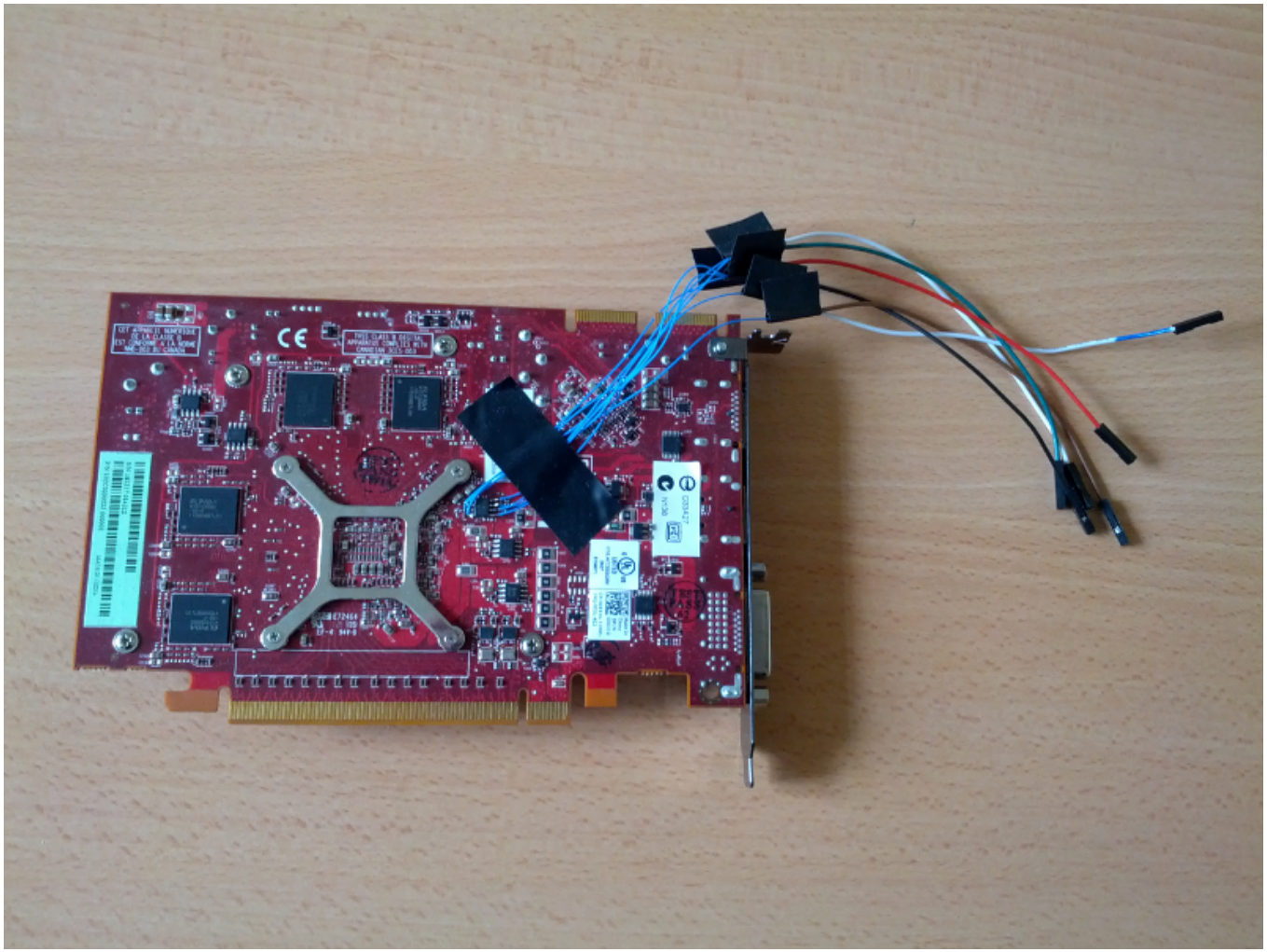
...
All combined : nice story for Matrix fans ...
Need I go on ?
Mrk

Rappels : PCI Expansion ROMs

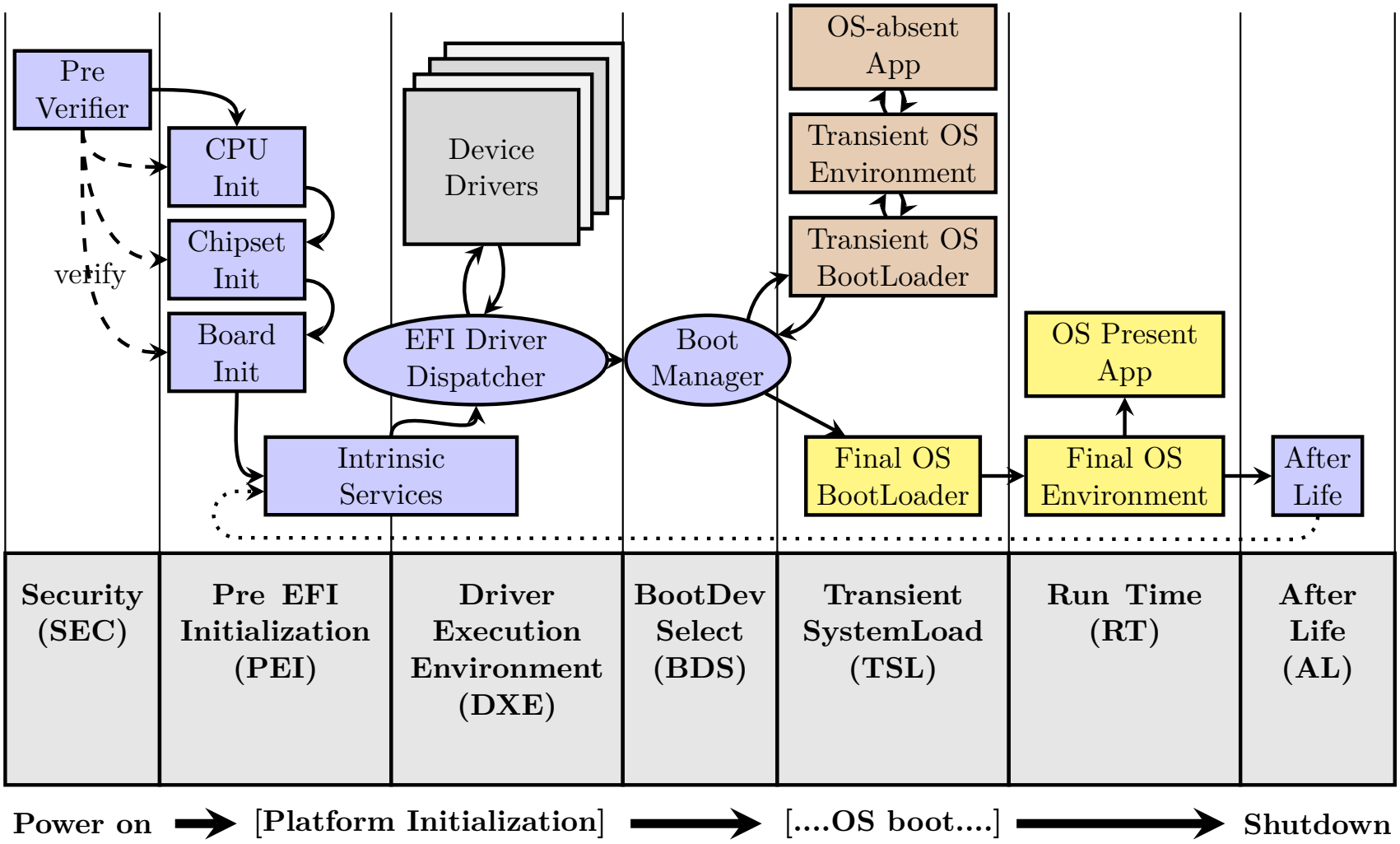


- (petit) espace mémoire optionnel des cartes PCI/PCIe/Thunderbolt
- Fournit du code exécuté par le Firmware
- Déjà exploité par le passé en version BIOS
- UEFI ?

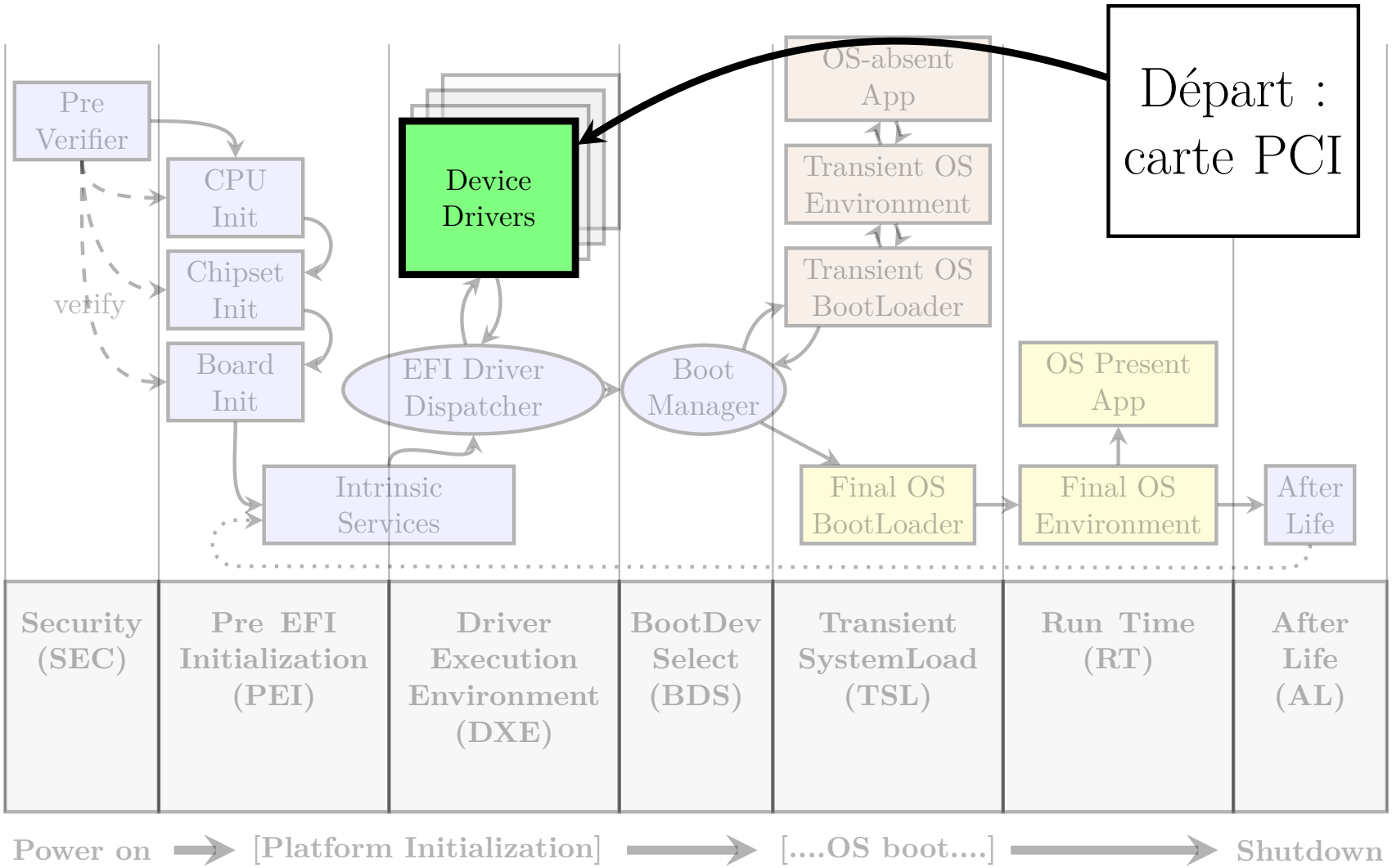
Carte VGA



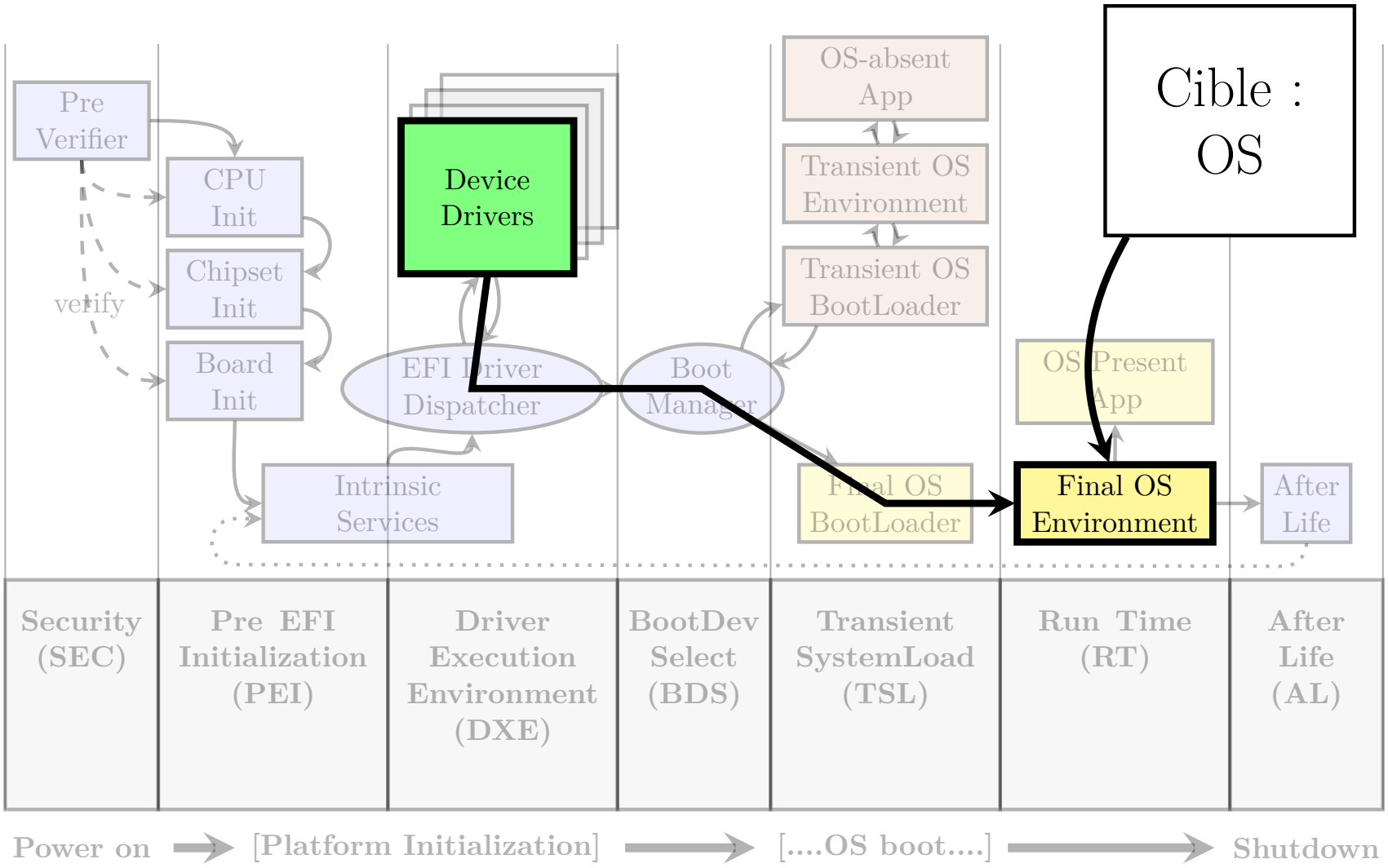
Séquence de démarrage avec UEFI



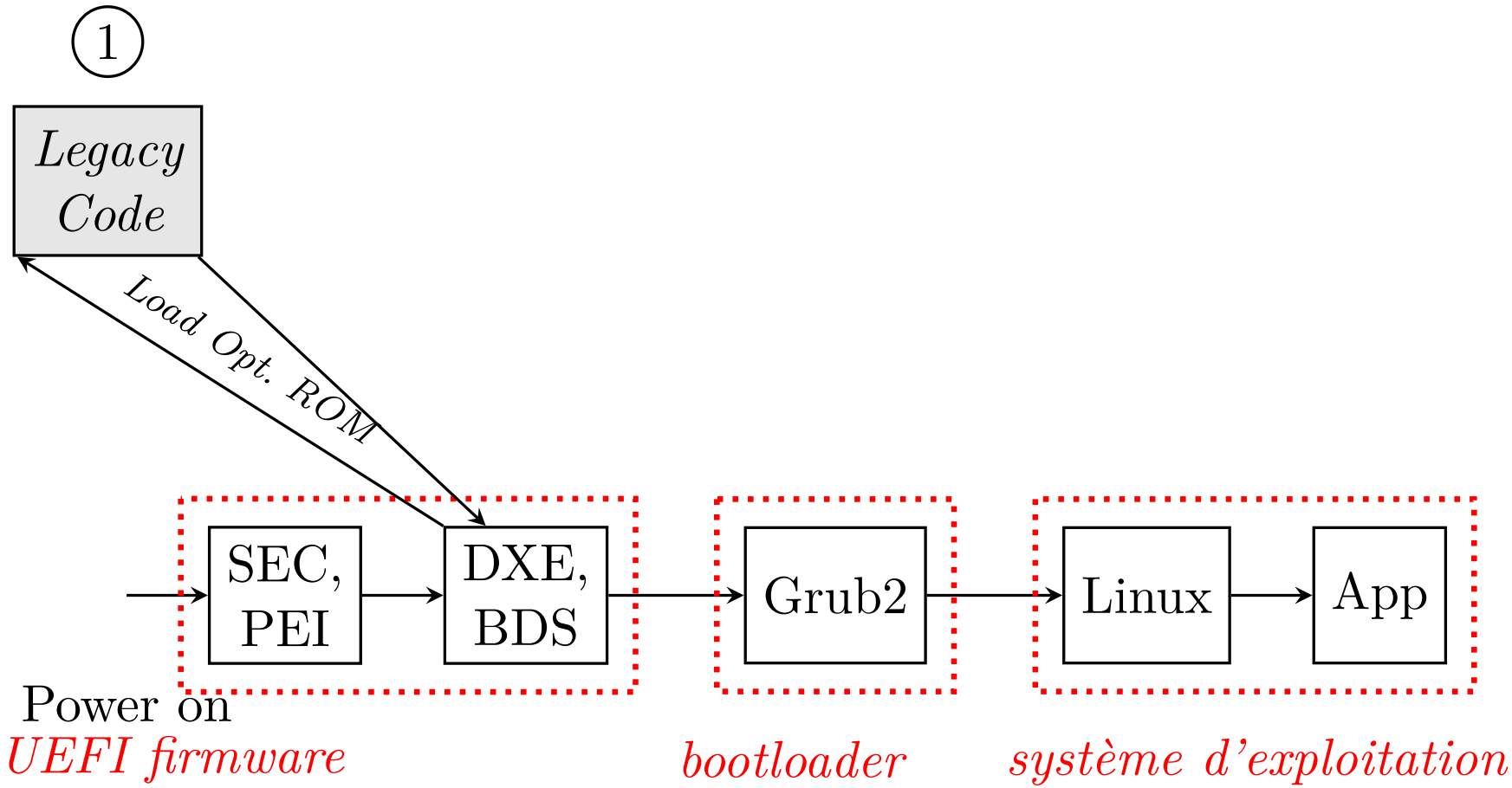
Scénario (La route du Rhum^W ROM)



Scénario (La route du Rhum^W ROM)



Séquence de démarrage avec UEFI



ROM arrangée

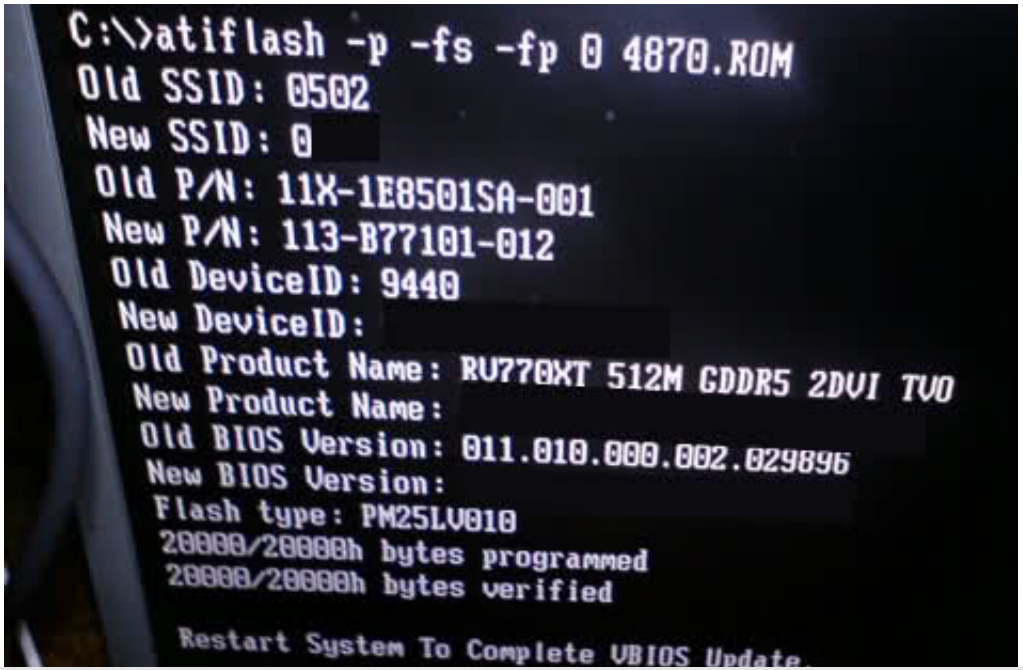
- identification de l'expansion ROM (vanilla)
- dump de la PCI expansion ROM
- ajout de la version UEFI pour créer une ROM hybride
- flash

Dump(*Importation*) de ROM

- Cat `/sys/bus/pci/devices/0000\:00\:02.0/rom`
- Outils constructeur

Exemple ATI

```
E:\> atiflash.exe -unlockrom 0
E:\> atiflash.exe -p -f 0 myrom.bin
```



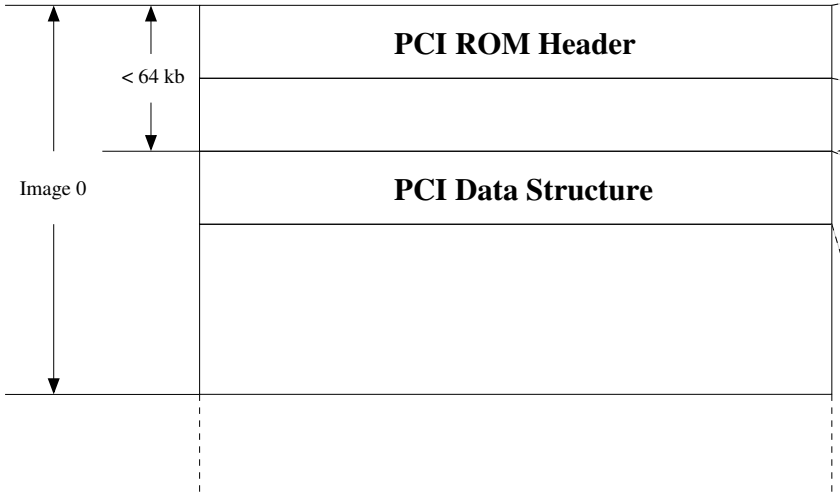
Création ROM arrangée [1]

Création d'une "ROM hybride"

- Utilisation du Kit de Développement (`vim + gcc`)
- Création driver DXE : code C, mode 64 bits (`make`)
- Choix des identifiants PCI
- Conversion au format ROM (`EfiRom`)
- Patch image (`cat`)

1. ROM ne s'est pas faite en un jour

PCI Expansion ROM format

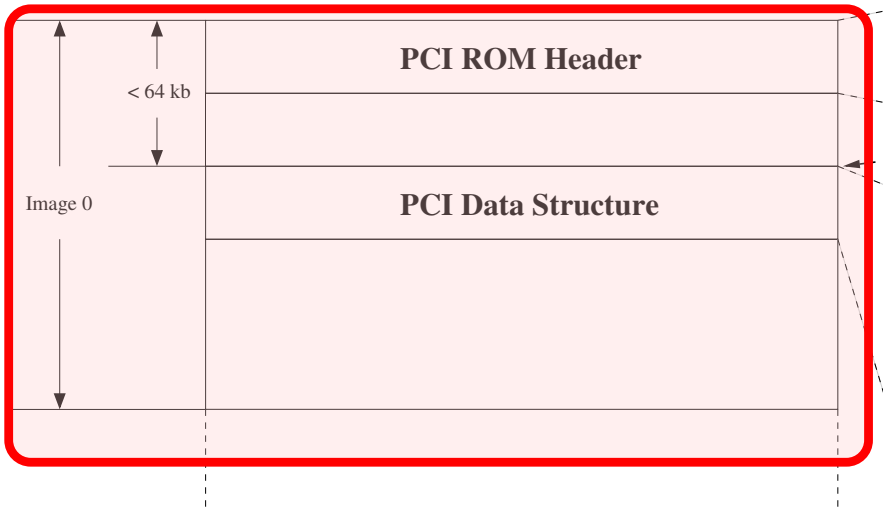


Offset	Length	Value	Header Field
00h–01h	2	55AAh	PCI Expansion ROM signature
02h–17h	16h		Architecture Specific Data
18h–19h	2		Pointer to PCI Data Structure

Offset	Length	Header Field
00h–03h	4	Signature, the "PCIR" string
04h–05h	2	Vendor Identification
06h–07h	2	Device Identification
08h–09h	2	Device List Pointer
0Ah–0Bh	2	PCI Data Structure Length
0Ch	1	PCI Data Structure Revision
0Dh–0Fh	3	Class Code
10h–11h	2	Image Length
12h–13h	2	Vendor ROM Revision Level
14h	1	Code Type
15h	1	Last Image Indicator
16h–17h	2	Maximum Runtime Image Length
18h–19h	2	Pointer to Configuration Utility Code Header
1Ah–1Bh	2	Pointer to DTMF CLP Entry Point

Modification de la PCI Expansion ROM

PCI Expansion ROM format



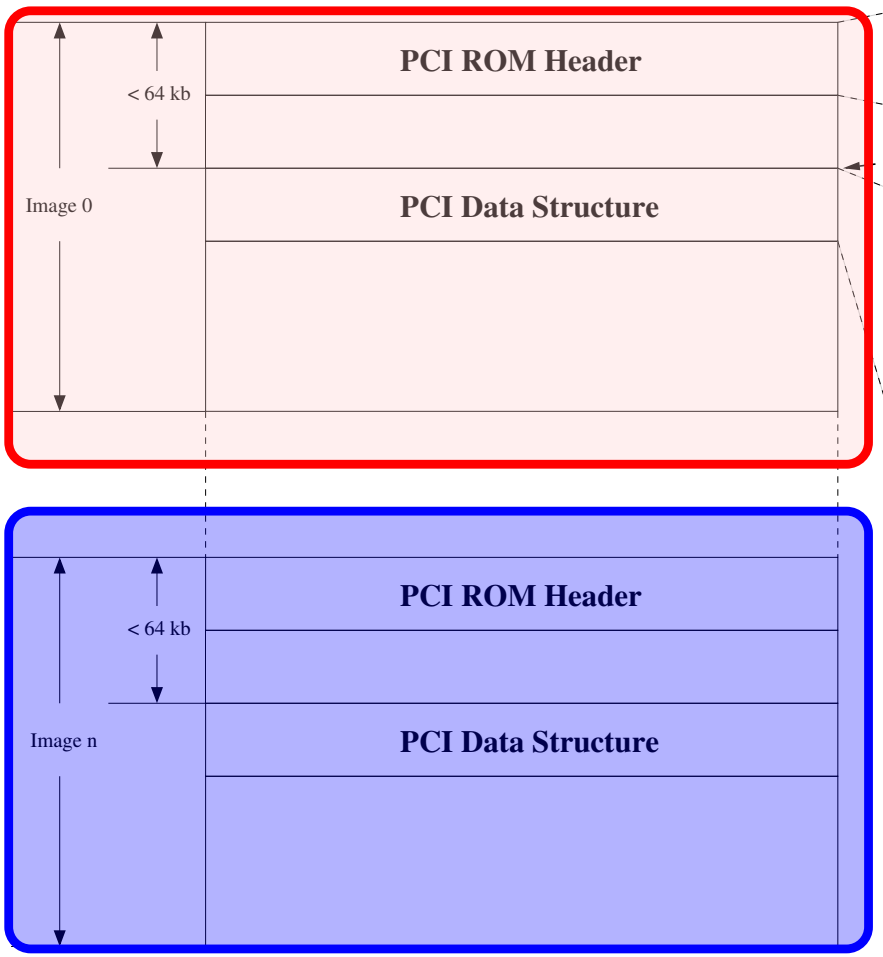
Offset	Length	Value	Header Field
00h–01h	2	55AAh	PCI Expansion ROM signature
02h–17h	16h		Architecture Specific Data
18h–19h	2		Pointer to PCI Data Structure

ROM *vanilla*

Offset	Length	Header Field
00h–03h	4	Signature, the "PCIR" string
04h–05h	2	Vendor Identification
06h–07h	2	Device Identification
08h–09h	2	Device List Pointer
0Ah–0Bh	2	PCI Data Structure Length
0Ch	1	PCI Data Structure Revision
0Dh–0Fh	3	Class Code
10h–11h	2	Image Length
12h–13h	2	Vendor ROM Revision Level
14h	1	Code Type
15h	1	Last Image Indicator
16h–17h	2	Maximum Runtime Image Length
18h–19h	2	Pointer to Configuration Utility Code Header
1Ah–1Bh	2	Pointer to DTMF CLP Entry Point

Modification de la PCI Expansion ROM

PCI Expansion ROM format



Offset	Length	Value	Header Field
00h–01h	2	55AAh	PCI Expansion ROM signature
02h–17h	16h		Architecture Specific Data
18h–19h	2		Pointer to PCI Data Structure

ROM *vanilla*

Offset	Length	Header Field
00h–03h	4	Signature, the "PCIR" string
04h–05h	2	Vendor Identification
06h–07h	2	Device Identification
08h–09h	2	Device List Pointer
0Ah–0Bh	2	PCI Data Structure Length
0Ch	1	PCI Data Structure Revision
0Dh–0Fh	3	Class Code
10h–11h	2	Image Length
12h–13h	2	Vendor ROM Revision Level
14h	1	Code Type
15h	1	Last Image Indicator
16h–17h	2	Maximum Runtime Image Length
18h–19h	2	Pointer to Configuration Utility Code Header
1Ah–1Bh	2	Pointer to DTMF CLP Entry Point

Code UEFI

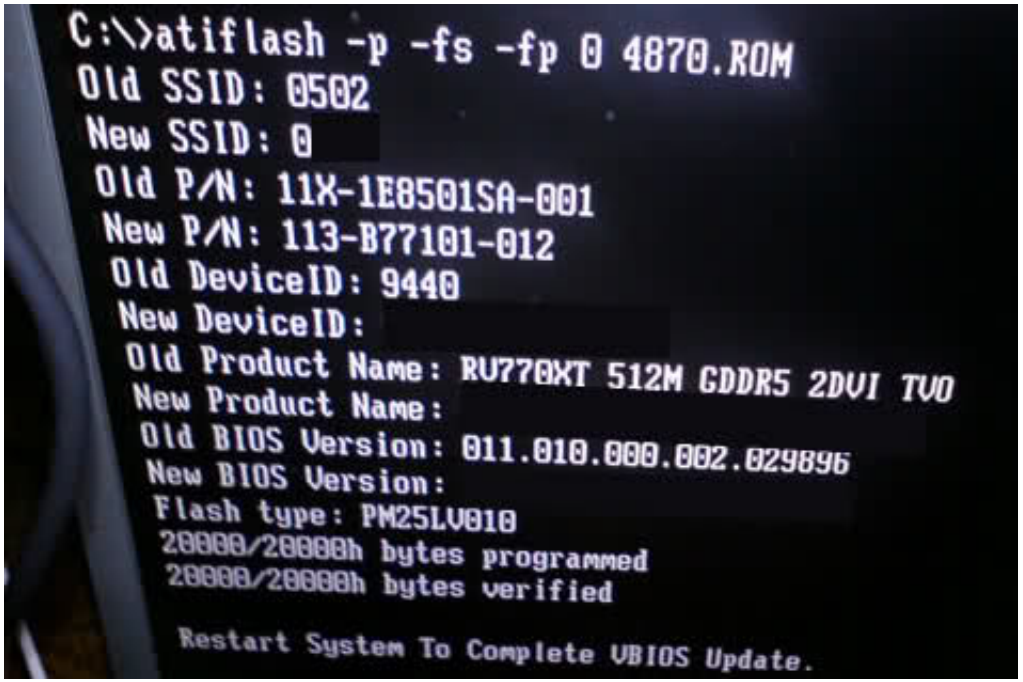
Modification de la PCI Expansion ROM

Écriture ROM (1/2)

- Outils constructeur

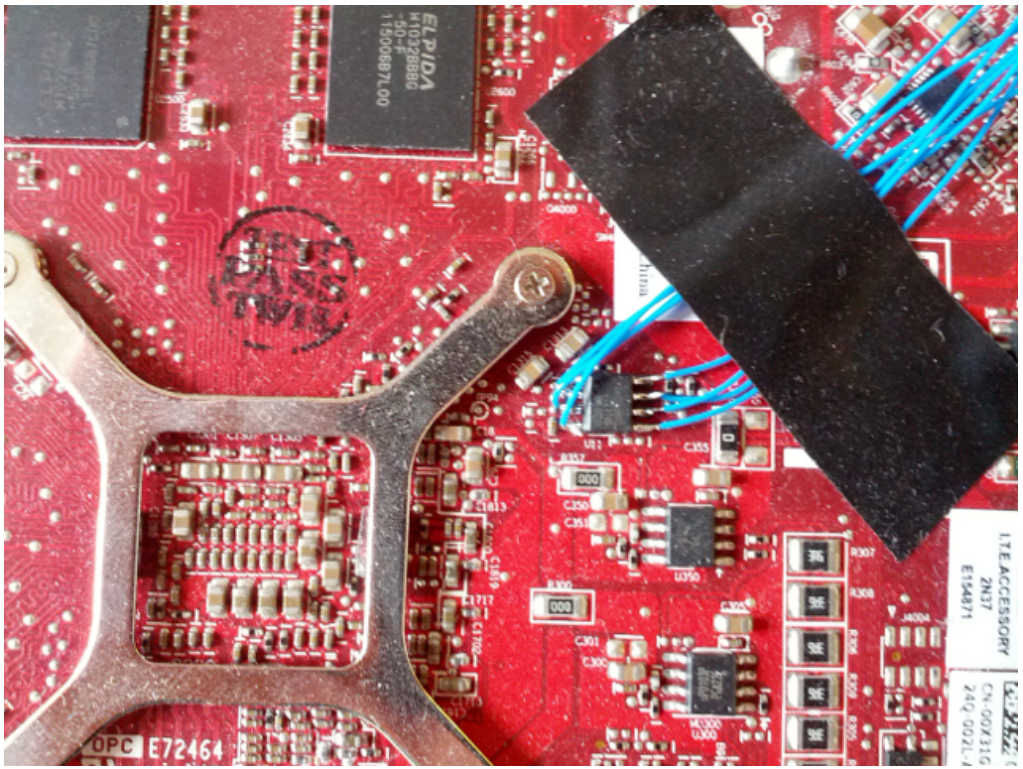
Exemple ATI

```
E:\> atiflash.exe -unlockrom 0
E:\> atiflash.exe -p -f 0 myrom.bin
```



Écriture ROM (2/2)

- Flash SPI : outils low-level

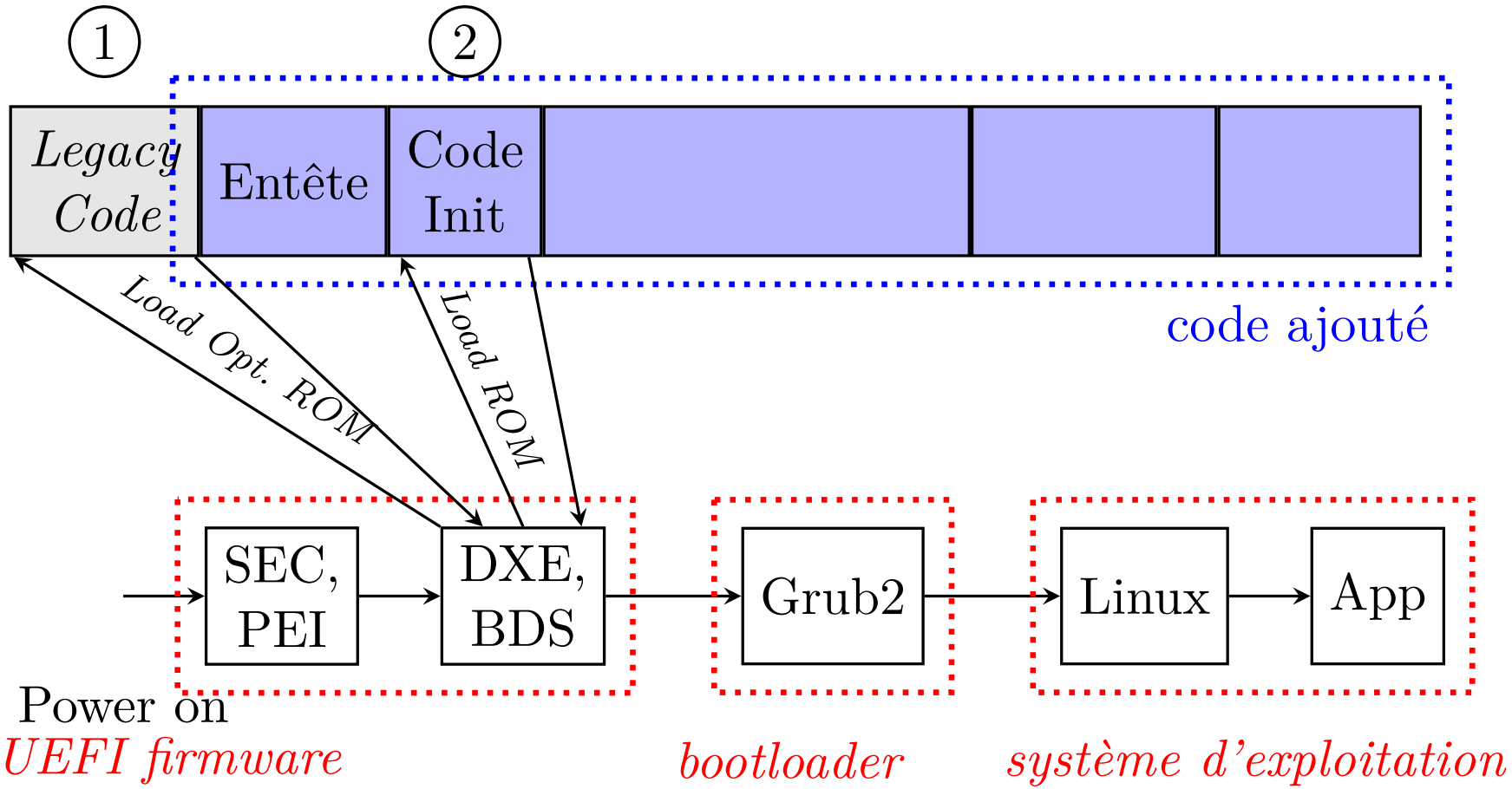


Exécution dans UEFI

- Le *firmware* UEFI énumère les périphériques PCI
- Les Expansion ROM sont chargées en mémoire [2] :
  - Legacy : (`0xc0000` → `0xfffff`)
  - UEFI : dynamique
- ROM d'origine chargée par le CSM
- ROM UEFI chargée ensuite
- Le point d'entrée C est appelé
- La fonction `ExitBootServices` est *hookée*

2. Tous les chemins mènent à ROM

Chargement des *PCI Expansion ROM*



Interception du bootloader

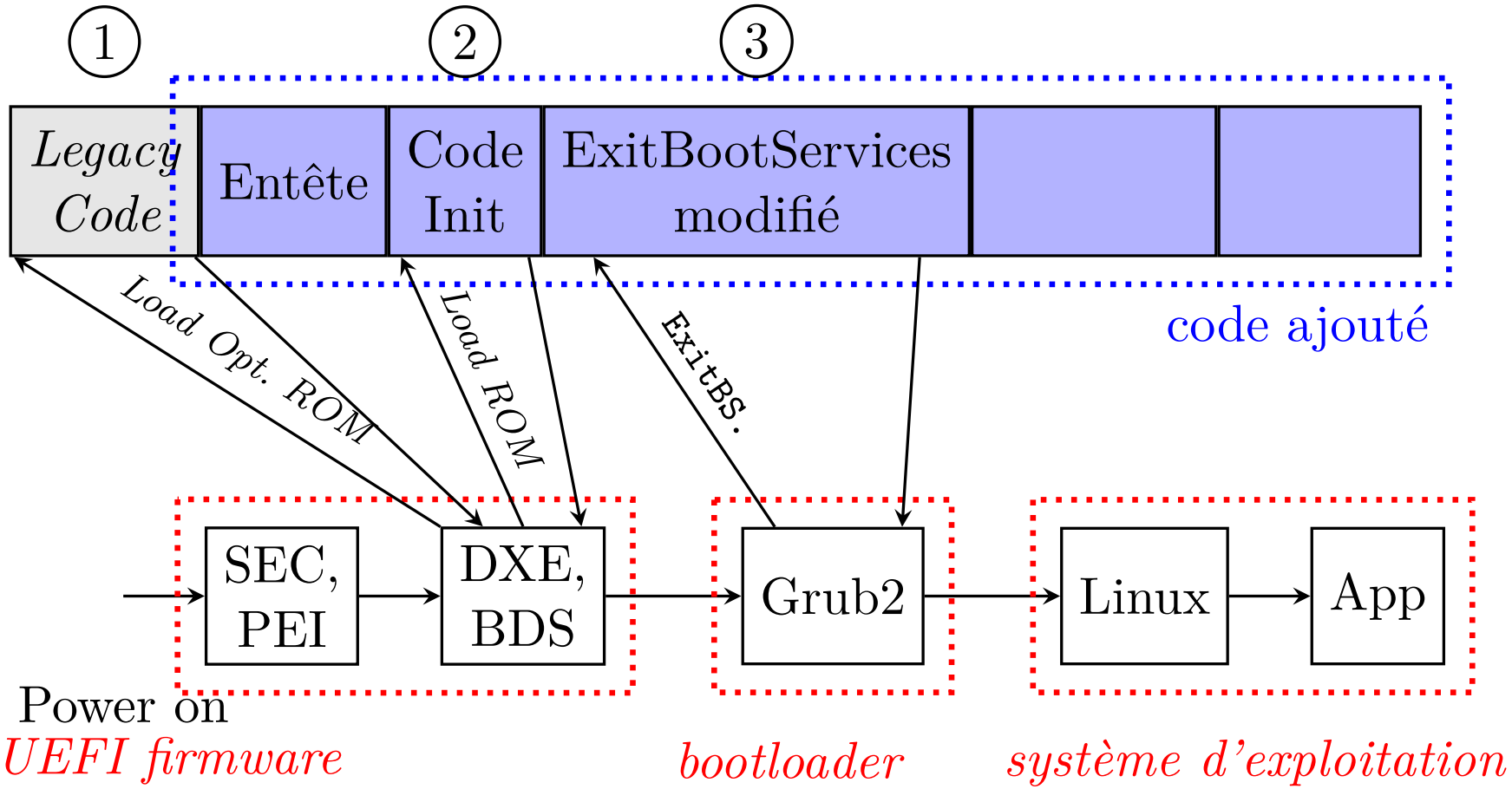
- Cas de `Grub2`
- Copie de l'image du noyau en mémoire (adresse ?)
- Appel à `ExitBootServices`
- Problème : réutilisation de la mémoire

Interception du bootloader

- Cas de `Grub2`
- Copie de l'image du noyau en mémoire (adresse ?)
- Appel à `ExitBootServices`
- Problème : réutilisation de la mémoire

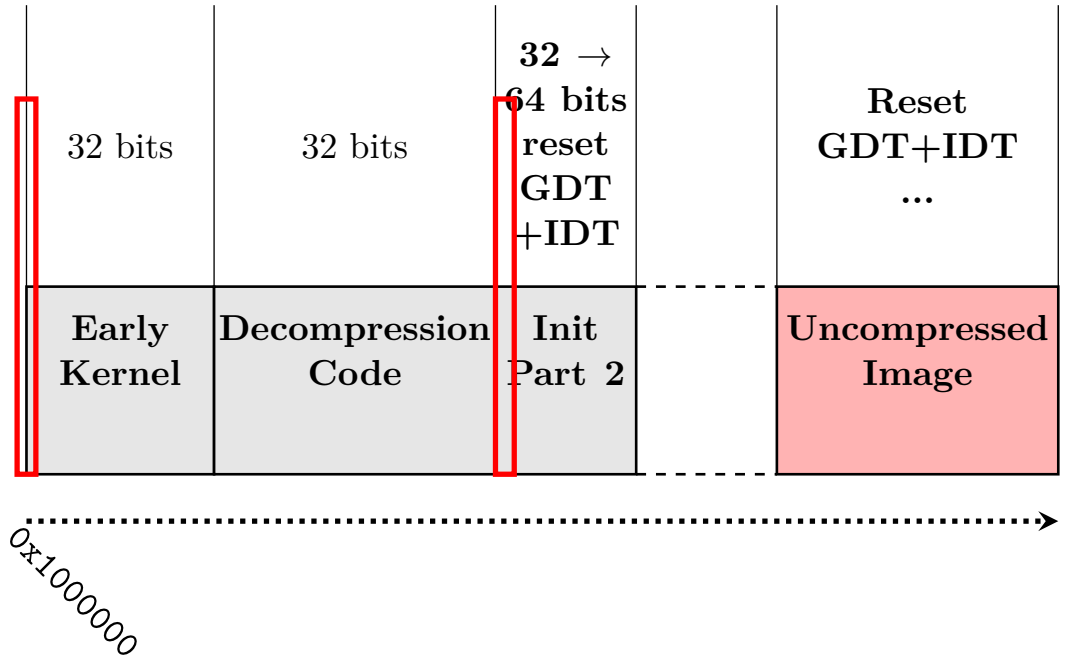
- Allocation mémoire persistante
- Reconstruction de la *call stack*
- Identification de l'adresse
- Préparation de l'étape suivante

Étape suivante : *bootloader*



Interception du noyau (1/2)

- Image mémoire avant décompression
- Adresse physique ≠ adresse virtuelles
- Noyau : initialise l'IDT, GDT, pagination, etc.
- Changement de mode (32 -> 64 bits), CS et DS, ...
- Point d'arrêt ? Non (IDT)

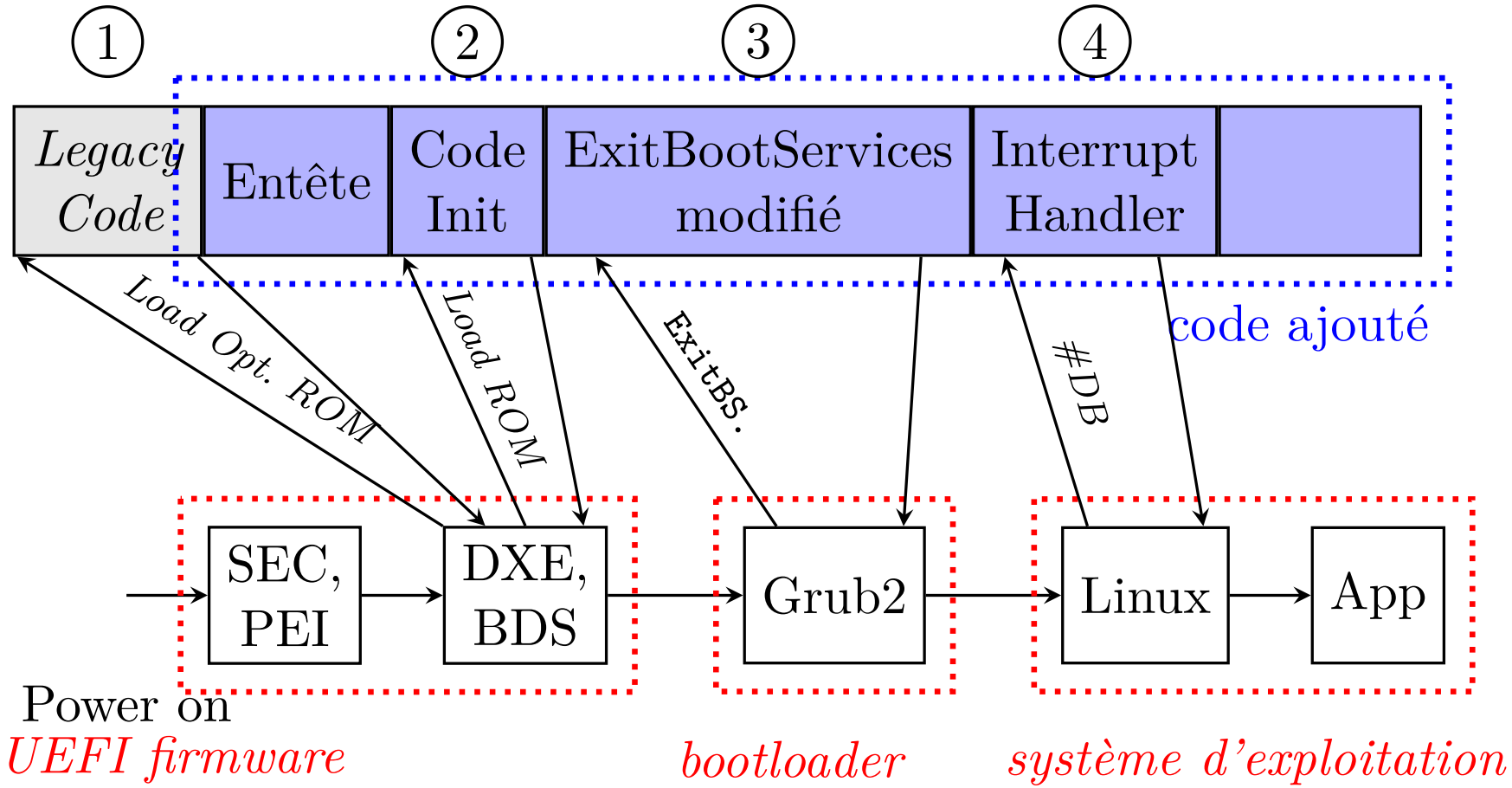


Interception du noyau (2/2)

Utilisation des *Hardware Breakpoints*

- Hardware Debug Registers
- Utilisation de l'interruption `#DB`
- Vecteur d'interruption 1
- Hardware BP 1 : `0x1000000`
- Hardware BP 2 : avant chargement IDT, dans `Init Part 2`

Étape suivante : *early kernel*



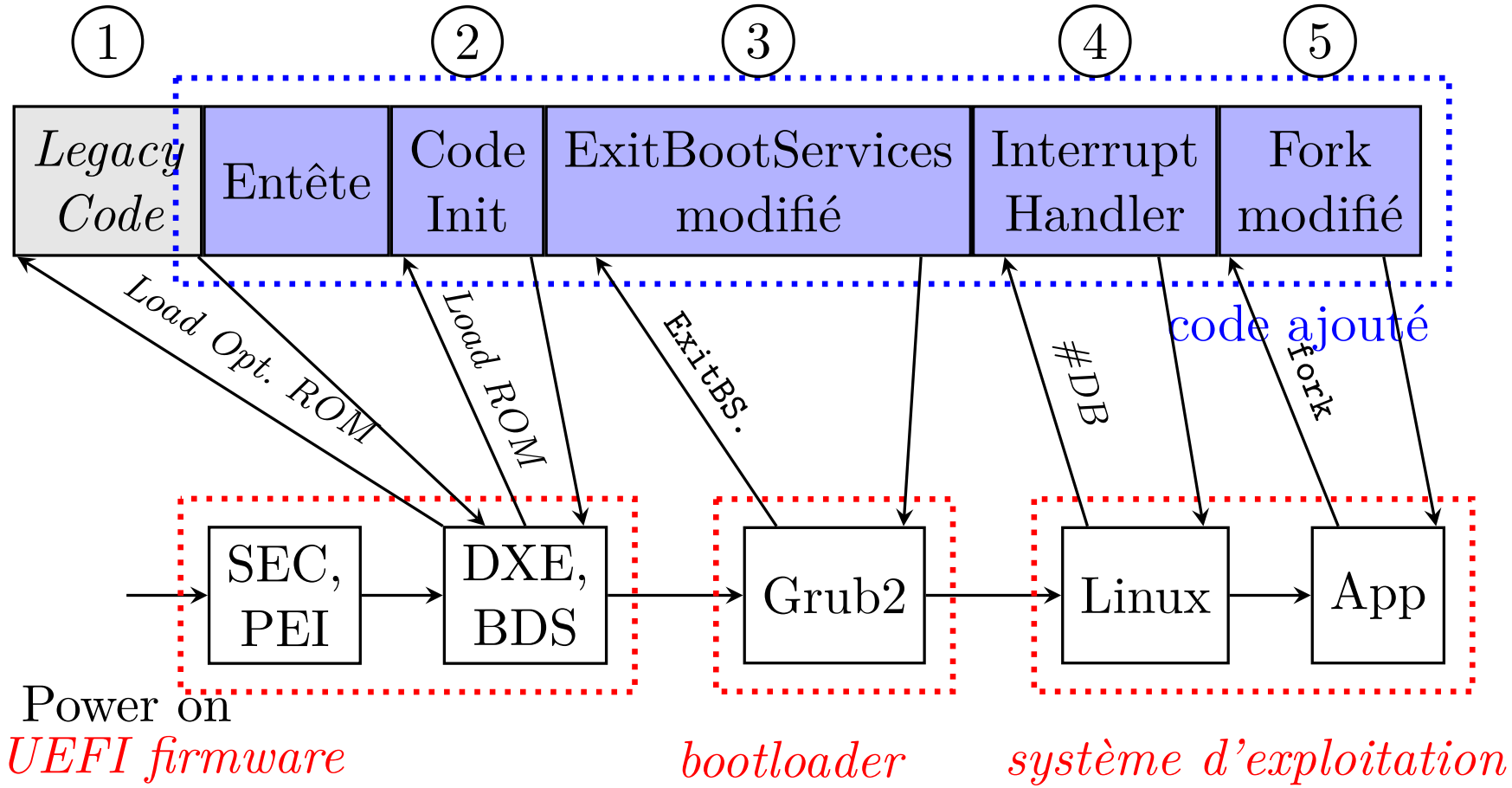
Modification du syscall

- Modification d'un appel système
- Remplacement du code en mémoire
- Élévation de privilèges
- Appel choisi : `fork`
- Adresse du *syscall* ?
- Adresses des fonctions internes ?

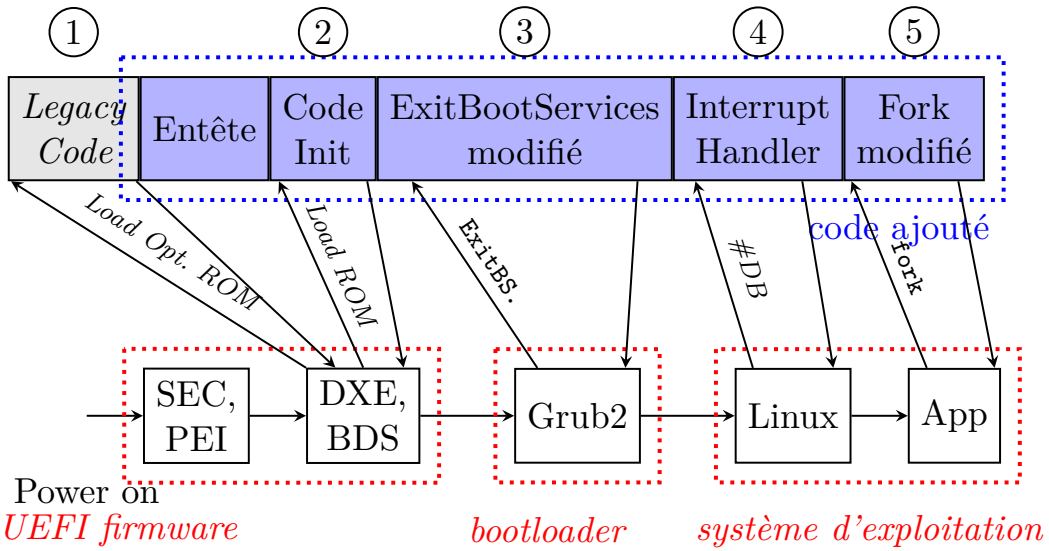
Appel système modifié

```
xor    %rdi,%rdi
call   *0xffffffff8106406f  ; prepare_kernel_cred
call   *0xffffffff81063db6  ; commit_creds
ret
```

Étape suivante : *syscall*



Démo


1
2
3
4
5
Legacy Code
Entête
Code Init
ExitBootServices modifié
Interrupt Handler
Fork modifié
code ajouté
Load Opt. ROM
Load ROM
ExitBS.
#DB
fork
SEC, PEI
DXE, BDS
Grub2
Linux
App
Power on
UEFI firmware
bootloader
système d'exploitation

Conséquences

- Furtif : pas de modification disque, empreinte mémoire discrète
- Indépendant d'un chiffrement de disque
- Mot de passe BIOS : ne bloque pas
- Antivirus (même UEFI) : inutile
- grsec / randomisation / … : complique, mais n'empêche pas
- Solutions ?

Solution 1 : TPM

- Composant **passif**
- Présent sur (presque) tous les PC
- Principe : mesures des éléments
- Utilisé *via* le scellement de données

Difficultés

- Pas/peu utilisé
- Pas de support dans tous les *bootloaders*
- Complique les mises à jour
- Impose un chiffrement de disque pour être effectif
- Pas parfait [a]

a. cf. NIST / *BIOS Chronomancy*, NoSuchCon 2013

Solution 2 : Secure Boot

- Composant d'UEFI
- Vérification de signatures cryptographiques (RSA2048)
- Tous les éléments (exécutables, *drivers*, *expansion ROM*, etc.)

Difficultés

- Optionnel (même si requis pour *Windows 8 Hardware Certification*)
- Gestion des autorités de certification
- Gestion de la compatibilité
- Restrictions d'usage (ex. tablettes ARM)

Conclusion

Matériel

- *Les protections existent mais ne sont pas toujours (bien) implémentées*
- Confiance indispensable dans le matériel
- Conséquences importantes
  - Contournement de toutes les protections
  - Possibilité de grande furtivité

Suggestions possibles pour les constructeurs / éditeurs

- Protéger l'UEFI des écritures SPI (sauf en mode *reboot*)
- N'autoriser que les MAJ signées
- Protéger les étapes initiales (SEC/PEI)
- Protéger la racine de confiance S-CRTM
- *et le faire sans bugs*

Travaux futurs

Suite

- Utilisation des (nombreuses) fonctions UEFI
- *EFI Byte Code*[a]
- Virtualisation *à la* Blue Pill
- Étude des *firmwares* UEFI
- Étude des implémentations (*Secure Boot*, *IPsec*)

a. Ils sont fous ces ROM, hein ?

Questions ?