~~CONFIDENTIEL DÉFENSE~~ ~~SPECIAL FRANCE~~

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PREMIER MINISTRE

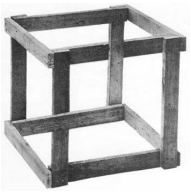
Secrétariat général de la défense et de la sécurité nationale

Agence nationale de la sécurité des systèmes d'information

DÉCLASSIFIÉ
par décision n°15699/ANSSI/SDE/ST/LAM
du 18 juillet 2018

Documentation CLIP 1004

Support de l'UEFI



ANSSI, 51 boulevard de la Tour Maubourg, 75700 Paris 07 SP.

~~CONFIDENTIEL DÉFENSE~~

Résumé

Ce document présente la façon dont a été intégré le support de microcode respectant la norme *Unified Extensible Firmware Interface* (UEFI) dans CLIP. Ce support permet d'installer et de démarrer un système CLIP sur les machines les plus récentes du marché. Certaines n'assurent en effet plus la compatibilité avec les microcodes de type BIOS. Le support véritable de l'UEFI est une fonctionnalité de la version 4.4.3.

HISTORIQUE

Révision	Date	Auteur	Commentaire
1.0	22/09/2015	Thomas Letan	Version initiale.

Table des matières

1 Présentation de la norme UEFI

Le standard *Unified Extensible Firmware Inteface* (UEFI) est une norme visant à décrire le fonctionnement du microcode d'une carte mère. Il s'agit d'un projet initié par Intel (sous le nom d'EFI, sans le U), puis repris par l'UEFI Forum en 2005. L'implémentation de référence de la norme UEFI est l'EDK II. Régulièrement, Intel publie une nouvelle version de l'UDK, basé sur EDK II.

1.1 Comparaison avec le BIOS

Il est un remplaçant de plus en plus généralisé du *Basic Input/Output System* (BIOS). L'implication de Microsoft dans l'UEFI Forum et son importance dans les *Hardware Requirement* de Windows 8 ont accéléré son adoption. Les différences entre le BIOS et l'UEFI sont nombreuses, mais trois sont particulièrement importantes :

- Le BIOS était écrit en assembleur et exécuté en mode réel, l'UEFI est majoritairement écrit dans un langage haut niveau (le C) et active très rapidement le *long mode* [1] ;
- Le BIOS repose sur la présence d'un *Master Boot Record* dans le premier secteur du disque dur, tandis que l'UEFI repose sur la présence d'une table de partitionnement de type GPT et la présence d'un exécutable EFI sur une partition formatée en FAT32 ;
- L'UEFI met à disposition du logiciel exécuté au *runtime* de la machine un certain nombre de services [2].

1.2 Prérequis pour supporter l'UEFI

La séquence de démarrage est précisément décrite par la norme UEFI [3]. La finalité d'un microcode implémentant la norme UEFI est de chargé et d'exécuté une « application UEFI » [4]. Cette application UEFI doit être stockée dans une partition FAT32, estampillée partition *EFI System* [5].

Un microcode respectant la norme UEFI intègre un chargeur d'amorçage configurable par le biais de « variables UEFI » particulière. La plupart des chargeurs d'amorçage grand publics sont cependant compatible avec UEFI : ils peuvent être compilés comme une application UEFI. À noter que l'application UEFI doit être compilée pour fonctionner dans le même mode d'exécution que le microcode. Dans les faits, cela signifie que l'application UEFI doit être compilée pour être exécutée en *long mode*.

Pour résumer, pour supporter l'UEFI, il faut :

- Un disque dur partitionné au format GPT ;
- Une parition formatée en FAT32 ;
- Un chargeur d'amorçage compilé comme une application UEFI ;
- Un noyau capable de libre une table de partition de type GPT ;

2 État du support

2.1 Versions impactées

Les premières modification en faveur d'un support de l'UEFI ont été intégrées dans CLIP 4.4.2 et supérieures, notamment le support du schéma de partitionnement GPT. Les modifications les plus impactantes

1. Les implémentations UEFI 64bit sont largement répendues, les implémentations UEFI 32bits sont *beaucoup* plus rares (a priori, uniquement quelques MacBook)
2. Le BIOS fournit un ensemble d'interruption que les systèmes d'exploitation, majoritairement exécuté en mode protégé ou supérieur, ne pouvaient pas utiliser.
3. Plus précisément, les premières étapes de la séquence de démarrage sont décrites par la norme *Platform Initialization* (PI), tandis que la norme UEFI décrit les étapes suivantes et les interactions entre le logiciel exécuté au *runtime* (typiquement, le système d'exploitation) et le microcode.
4. Un binaire au format PE/COFF.
5. Le schéma de partitionnement est nécessairement GPT.

ont été intégrées uniquement dans CLIP 4.4.3. Sur le Bugzilla, le ticket lié est le #3228. Le support est considéré comme fonctionnel à partir de la révision *subversion* numéro 14733 (`clip-core-conf-4.4.3-r3`).

2.2 Matériels supportés

Si le support UEFI est fonctionnel dans l'absolu, il doit cependant être adapté pour chaque profil matériel. Pour l'heure, CLIP vise le support de trois machines physiques et de l'hyperviseur QEMU/KVM. La Table 1 récapitule l'état du support UEFI pour la version 4.4.3-r3 de *clip-core-conf*.

KVM OVMF64 et KVM OVMF32 Ces deux profils matériels se basent sur le profil KVM Seabios (anciennement le profil KVM). Comme ils se basent sur un profil fonctionnel et éprouvé, la transition se limite à simplement modifier le contenu du fichier `fw`. Néanmoins, ces profils doivent être testés pour s'assurer qu'ils apportent bien le support de l'UEFI et que CLIP est toujours fonctionnel.

HP ProBook 430 G2 UEFI Le HP ProBook 430 G2 est capable d'effectuer une séquence de démarrage de type UEFI *et* BIOS. Il existe donc deux profils matériels (à compter de la version 3.14.19 de *clip-data/clip-hardware*). Le profil matériel estampillé UEFI de ce modèle est fonctionnel, mais une erreur dans l'implémentation du microcode rend impossible l'affichage du menu *syslinux*[6].

TOSHIBA Portege R30 UEFI Le TOSHIBA Portege R30 UEFI est capable d'effectuer une séquence de démarrage de type UEFI *et* BIOS. Il existe donc deux profils matériels (à compter de la version 3.14.16 de *clip-data/clip-hardware*). Le support UEFI n'est pas fonctionnel, le noyau refuse de démarrer car il n'arrive pas à charger ses modules[7].

MICROSOFT Surface PRO3 La MICROSOFT Surface PRO3 utilise un microcode de type UEFI. Il existe un profil matériel à son intention, mais ce dernier n'est pas encore fonctionnel. Une distribution clip-tiny démarre correctement, mais la *cover*[8] n'est pas correctement reconnu. Une distribution clip-rm ne démarre pas complètement, à cause d'une erreur de type *segmentation fault* impliquant le *splashscreen*.

3 Description technique du support

Le support de l'UEFI dans CLIP est rendu complexe par plusieurs caractéristiques du système d'exploitation. On citera notamment :

- CLIP est prévu pour être exécuté en mode protégé et ne peut donc pas être une application UEFI ;
- La configuration du système d'amorçage (syslinux) repose sur des liens symboliques qui ne sont pas disponible dans une partition FAT32 ;

3.1 Modifications liées à l'installeur

L'installeur CLIP est un outil générique permettant d'installer n'importe quel mirroir de paquetages CLIP (et donc, n'importe quelle distribution CLIP) sur un poste. L'installation peut se faire par ligne de commande (rare) ou *via* l'outil graphique dédié (cas d'usage classique).

Une installation est paramétrée par :

- Les miroirs à installer (ce qui définit la distribution à installer) ;

6. Ce n'est pas le chargeur d'amorçage qui est en cause : n'importe quelle application UEFI se servant des Runtime Services rencontrera le même problème.
7. La raison précise de cette erreur n'est pas encore connue.
8. Clavier spécifique à la Surface vendu par Microsoft

Profil matériel	État du support			Commentaires
	clip-rm	clip-gtw	clip-tiny	
KVM_OVMF64	À tester	À tester	À tester	KVM nécessite OVMF, le microcode de type UEFI à destination des machines virtuelles.
KVM_OVMF32	À tester	À tester	À tester	KVM nécessite OVMF, le microcode de type UEFI à destination des machines virtuelles.
HP ProBook 430 G2 UEFI	Complet	Non testé	Complet	*syslinux* ne s'affiche pas, à cause de l'implémentation UEFI de HP
TOSHIBA Portege R30 UEFI	Ne boot pas	Non testé	Ne boot pas	L'`initrd` ne parvient pas à charger les modules
MICROSOFT Surface PRO3	Ne boot pas	Non testé	Partiel	*Segfault* dans *fbsplash*, support *cover* manquant

TABLE 1 – Récapitulatif du support de l'UEFI dans CLIP par profil matériel

- Le profil matériel à appliquer (ce qui définit, entre autres, les modules noyaux nécessaires au bon fonctionnement de la machine) ;
- Le profil utilisateur (qui contient par exemple les clefs cryptographiques qui seront utilisées par CLIP) ;
- Le disque sur lequel installer CLIP.

Déterminer la nature de la séquence de démarrage Sans le support de l'UEFI, la question de la nature du microcode de la machine ne se posait pas. Le support de l'UEFI change la donne et il est important de savoir si, pour une machine donnée, la séquence de démarrage suit la logique du BIOS ou de l'UEFI. Dans le premier cas, le disque dur doit être formaté suivant un schéma de type MBR. Dans le second, c'est le schéma GPT qui doit être utilisé. Cela influence aussi la façon dont est installé le chargeur d'amorçage.

Un fichier `fw` (diminutif de *firmware*) a été ajouté aux profils matériels supportés par CLIP (paquetage `clip-data/clip-hardware`). Il sert à déterminer la nature de la séquence de démarrage pour une machine donnée. Son contenu doit être soit `bios`, soit `efi32`, soit `efi64`. Pour aux services de CLIP, une fois le système installé, quelle est la nature de la séquence de démarrage, le fichier est copié lors de l'installation de `clip-hardware` dans le dossier `/etc` (plus précisément, `/etc/fw.conf`).

Partitionnement du disque En fonction du contenu du fichier `fw`, l'installeur partitionne le disque soit en suivant le schéma de partition MBR (`bios`) ou GPT (`efi32` ou `efi64`).

La façon dont est géré le partitionnement du disque a été complètement modifiée avec le support de l'UEFI. Avant, l'installeur générait un script *sfdisk* (fichier `sbin-scripts/compute_sizes.sh` de `clip-livecd`). Désormais, *parted* est utilisé à la place de *sfdisk* (fichier `sbin-scripts/generate_parted_script.sh` de `clip-livecd`).

En fonction de la distribution CLIP à installer (`clip-rm`, `clip-gtw`, `clip-tiny`), l'installeur demande une taille minimum de disque plus ou moins grande. Ces minima sont définis dans le script `generate_parted_script` dans les variables de description de partition (*.*_PARTITION*). Le Tableau 2 résume ces minimas.

CLIP dépend d'un schéma de partitionnement fixe pour monter ses partitions. Cet état de fait implique l'insertion de partitions inutiles lorsque CLIP est installé :

- avec moins de deux cages RM (cas de `clip-gtw` et `clip-tiny`) ;
- pour une séquence de démarrage de type UEFI.

Partition	clip-rm	clip-gtw	clip-tiny
`/boot`	128	128	128
`/home`	10 000	1 000	100
`/var/log`	2 000	4 000	2 000
`/`	2 000	2 000	2 000
`/mounts`	4 000	4 000	400
Cages	12 000	16	16
Swap	2 000	2 000	1 000
Total	50 128	19 160	8 060

TABLE 2 – Tailles minimales (en Mio) pour les partitions en fonction de la distribution

Pour ce second cas, la raison s'explique par les différences entre un schéma de partitionnement de type MBR et GPT. Dans le cadre du MBR, la spécification limite le nombre de partitions physiques à quatre. Pour avoir plus de quatre partitions, il faut créer une partition étendue qui englobera plusieurs partitions logiques. Dans le cadre du GPT, cette limitation n'existe pas. Il faut donc créer une partition inutile pour forcer le numéro des partitions à s'incrémenter artificiellement.

Installation du chargeur d'amorçage *syslinux* La version *syslinux* utilisée par CLIP dans sa version 4.4.2 est la 4.05. *Syslinux* intègre le support d'UEFI à partir de sa version 6.00 et la possibilité d'utiliser un noyau 32bit lorsqu'il est compilé comme une application UEFI 64bit à partir de sa version 6.02. Cette dernière version est cependant fortement déconseillée par les mainteneurs, qui conseillent d'utiliser la version 6.03. Les changements apportés à *syslinux* sont décris plus précisément dans la Section 3.2. Nous nous concentrons ici sur la façon dont il est installé.

Dans CLIP 4.4.2, l'installation de *syslinux* est effectuée à deux reprises. D'abord, lors de l'installation du paquetage, ensuite par le script d'installation CLIP lui-même. Cette double installation n'était pas utile et a donc été retiré lors de l'ajout du support de l'UEFI.

Dans CLIP 4.4.3, trois versions de *syslinux* cohabitent. Pour une séquence de démarrage de type BIOS, *syslinux* 4.05 continue d'être utilisé. Pour un microcode de type UEFI (64 ou 32bit), une version modifiée de *syslinux* 6.03 est utilisée. L'installation a proprement parlé du chargeur d'amorçage est gérée dans le fichier `postinst`. Les fichiers sont initialement copiés dans le dossier `/boot/syslinux/{bios,efi32,efi64}`. Then, in `postint`, the script checks the content of `/etc/fw.conf` to know what version it should installs.

3.2 Modifications apportées à Syslinux

syslinux 6.03 Pour intégrer le support de l'UEFI, l'utilisation de la version 6.03 est nécessaire. Néanmoins, le SDK CLIP possède plusieurs particularités qu'il faut prendre en compte :

- Le SDK CLIP est (comme CLIP lui-même) compilé en 32bit : cela veut dire que l'application UEFI ne peut pas être compilée pour être exécutée en mode 64bit sans inclure de la compilation croisée ;
- La version GCC de SDK CLIP est la 4.7 : *syslinux* repose sur une fonctionnalité de GCC5 [9].

Dans ces conditions, il n'est **pas possible de compiler syslinux 6.03** dans le CLIP SDK. La solution adoptée est la création d'un nouveau *distfile* : *syslinux-efi*. Le contenu de ce *distfile* est versionné dans le dépôt `clip-dev`. Il contient des binaires déjà compilés de *syslinux* version 6.03. Ces binaires ont été obtenu à partir du dépôt officiel de *syslinux* [10], en se basant sur l'étiquette `syslinux-6.03`, auxquelles ont été appliquées un correctif

9. *syslinux* est compilé afin d'obtenir un binaire dit PIC (Position Independant Code). Le mécanisme utilisé pour obtenir un binaire PIC utilise intensivement le registre `ebx`. GCC version 5 et supérieures sait gérer le code d'assemblage *inline* qui utilisent explicitement le registre `ebx` (c'est le cas de *syslinux*), mais pas les version précédentes)

10. `http://repo.or.cz/syslinux.git`

développé par l'ANSSI [11].

Modifications apportées à l'*ebuild* L'*ebuild* de *syslinux* (paquetage *sys-boot/syslinux* de l'arbre portage `portage-overlay`) a été modifié pour gérer la présence de trois versions *syslinux* distinctes.

Par défaut, l'*ebuild* place les binaires, modules *syslinux* et fichiers de configuration dans les dossiers `/boot/syslinux/{bios,efi32,efi64}`. La version BIOS est obtenue en recompilant *syslinux* version 4.05 depuis les sources présentes dans le distfile `syslinux-4.05.tar.gz`. Les versions EFI (32 et 64bits) sont récupérées dans le distfile `syslinux-efi-6.03.tar.gz`. C'est ensuite le script `postinst` qui installe concrètement la bonne version de *syslinux*.

Dans le cas d'un microcode de type BIOS, le contenu du dossier `/boot/syslinux/bios` est copié dans le dossier `/boot` et l'exécutable `extlinux` est utilisé pour installer le MBR sur le disque. Dans le cadre d'un microcode de type UEFI, le contenu du dossier `/boot/syslinux/efi64` (respectivement `/boot/syslinux/efi32`) est copié dans le dossier `/boot/EFI/boot`, notamment l'application *syslinux* nommée selon l'usage `bootx64.efi` (respectivement `bootia32.efi`). Le respect de cette convention se justifie par l'impossibilité d'utiliser les *Runtime Services* UEFI et donc de modifier les variables UEFI [12].

3.3 Modifications apportées au système de bascule de versions de CLIP

L'architecture de CLIP permet de faire cohabiter deux versions du système d'exploitation. Le but recherché de cette fonctionnalité est de pouvoir gérer le case où une mise à jour s'est mal passée (le système ne démarre plus). Plusieurs partitions vitales au systèmes sont dédoublées, notamment la partition racine : la cinquième et la dixième partition servent chacune de racine à l'une des versions CLIP installées. Dans CLIP 4.4.2, le système de bascule repose entièrement sur une fonctionnalité du système de fichier `ext3` : les liens symboliques. Le fichier de configuration de *syslinux* (`extlinux.conf`) est un lien symbolique vers `extlinux_5.conf` ou `extlinux_10.conf` [13]. Lors d'une mise à jour, c'est la version la plus ancienne qui est modifiée et le lien symbolique est modifié en conséquence.

Le système de fichier `FAT32` ne supporte pas les liens symboliques. À partir de CLIP 4.4.3, le fichier `extlinux.conf` n'est donc plus un lien symbolique. À la place, un fichier « normal » est utilisé. Ce dernier ne contient qu'une seule ligne se basant sur la directive de configuration `include` de *syslinux*.

4 Évolutions futures

Le support par CLIP de l'UEFI est fonctionnel à partir de la révision

4.1 Réorganisation du paquetage *syslinux*

Pour l'heure, la cohabitation des trois versions de *syslinux* (BIOS, application UEFI 32bit, application UEFI 64bit) est gérée directement dans l'*ebuild* « historique » de *syslinux* : paquetage `sys-boot/syslinux` dans l'arbre portage `portage-overlay`. L'ajout d'un deuxième *distfile* content des binaires précompilés et l'installation finale réalisée dans le *postinst* « dénaturent » cet *ebuild*, dont le nom (actuellement, `syslinux-4.05-r3`) n'est absolument plus représentatif.

La solution retenue est de séparer l'*ebuild* `sys-boot/syslinux` en deux *ebuilds* distincts. Le premier installerait la version 4.05 [14] de *syslinux* dans le dossier `/boot/syslinux/bios`. Le second installerait les applications

11. Depuis, ce correctif a été intégré dans *syslinux* (*commit* ae853e9) et sera donc présent dans la version 6.04 du chargeur d'amorçage.
12. La raison étant, une nouvelle fois, l'incompatibilité entre un noyau 32bit et un microcode 64bit.
13. Le numéro correspond à la partition racine de la version principale ; ainsi, si `extlinux.conf` est un lien symbolique vers `extlinux_5.conf` (respectivement `extlinux_10.conf`), alors la version la plus récente de CLIP utilise la cinquième partition comme racine (respectivement la dixième).
14. En attendant d'avoir résolu le problème de compilation de la version 6.03 de *syslinux* décrit dans la Section 3.2.

UEFI 32bit et 64bit respectivement dans les dossiers `/boot/syslinux/efi32` et `/boot/syslinux/efi64`, puis grâce au ficher `/etc/hw.conf` installerait réellement la version adéquate.

Références

[BUSYBOX] *Busybox*. http://www.busybox.net/about.html.

[CCSDa] Règles et recommendations concernant le choix et le dimmensionnement des mécanismes cryptographiques de niveau standard ou renforcé, *2379/SDGN/DCSSI/SDS/LCR du 19 décembre 2006*, ANSSI.

[CCSDb] Couche Cryptographique pour la Sécurité de Défense, *Document d'interface client version 3.2*, DGA.

[CLIP 1001] Documentation CLIP, 1001, *Périmètre fonctionnel CLIP-RM*, ANSSI.

[CLIP 1002] Documentation CLIP, 1002, *Architecture de sécurité*, ANSSI.

[CLIP 1003] Documentation CLIP, 1003, *Paquetages CLIP*, ANSSI.

[CLIP 1101] Documentation CLIP, 1101, *Génération de paquetages*, ANSSI.

[CLIP 1102] Documentation CLIP, 1102, *Génération du support d'installation CLIP*, ANSSI.

[CLIP 1103] Documentation CLIP, 1103, *Environnement de développement*, ANSSI.

[CLIP 1201] Documentation CLIP, 1201, *Patch CLIP-LSM*, ANSSI.

[CLIP 1202] Documentation CLIP, 1202, *Patch Vserver*, ANSSI.

[CLIP 1203] Documentation CLIP, 1203, *Patch Grsecurity*, ANSSI.

[CLIP 1204] Documentation CLIP, 1204, *Privilèges Linux*, ANSSI.

[CLIP 1205] Documentation CLIP, 1205, *Intégration de CCSD en couche noyau*, ANSSI.

[CLIP 1206] Documentation CLIP, 1206, *Génération de nombres aléatoires*, ANSSI.

[CLIP 1301] Documentation CLIP, 1301, *Séquences de démarrage et d'arrêt*, ANSSI.

[CLIP 1302] Documentation CLIP, 1302, *Fonctions d'authentification CLIP*, ANSSI.

[CLIP 1303] Documentation CLIP, 1303, *X11 et cloisonnement graphique*, ANSSI.

[CLIP 1304] Documentation CLIP, 1304, *Cages CLIP*, ANSSI.

[CLIP 1305] Documentation CLIP, 1305, *Support des cartes à puce*, ANSSI.

[CLIP 1306] Documentation CLIP, 1306, *Hotplug sous CLIP*, ANSSI.

[CLIP 1401] Documentation CLIP, 1401, *Cages RM*, ANSSI.

[CLIP 1501a] Documentation CLIP, 1501, *Configuration réseau*, ANSSI.

[CLIP 1501b] Documentation CLIP, 1501, *Racoon2*, ANSSI.

[CLIP 2001] Documentation CLIP, 2001, *Procédure d'installation*, ANSSI.

[CLIP 2002] Documentation CLIP, 2002, *Guide de configuration du BIOS (version adaptée au matériel considéré)*, ANSSI.

[CLIP 3001] Documentation CLIP, 3001, *Maintien en condition de sécurité*, ANSSI.

[CLIP 3002] Documentation CLIP, 3002, *Algorithmes cryptographiques*, ANSSI.

[CLIP 3101] Documentation CLIP, 3101, *Liste de configuration*, ANSSI.

[CLIP 4001] Documentation CLIP, 4001, *Guide de création de paquetage*, ANSSI.

[CLIP 4002] Documentation CLIP, 4002, *Outils et debug*, ANSSI.

[CLIP-DCS-120] *Règles et procédures de développement CLIP, CLIP_MAP-12000-007-DCS.*

[CLIP-DCS-130] *Spécification fonctionnelle des outils de gestion des mises à jour, CLIP-ST-13000-006-DCS.*

[DEBIAN] *Debian GNU/Linux*. http://www.debian.org.

[DEBOOTSTRAP] *Debootstrap*. http://packages.debian.org/stable/admin/debootstrap.

[DEBPOLICY] *Debian Policy Manual*. http://www.debian.org/doc/debian-policy/.

[DEVREL] *Gentoo Developer Handbook*. http://www.gentoo.org/proj/fr/devrel/handbook/handbook.xml.

[LDD] *Linux Device Drivers, 3rd Edition*. http://lwn.net/images/pdf/LDD3/.

[LSM] *Linux Security Modules : General Security Support for the Linux Kernel*. http://www.usenix..org/event/sec02/full_papers/wright/wright.pdf.

[PAX] *PaX*. http://pax.grsecurity.net.

[SSP] *GCC extension for protecting applications from stack-smashing attacks*. http://www.trl.ibm.com/projects/security/ssp.

[TOOLCHAIN] *Gentoo Linux Documentation - The Gentoo Hardened Toolchain*. http://www.gentoo.org/proj/en/hardened/hardened-toolchain.xml.