Chapitre 2
Outils noyau et initialisation

1. Protection dès la phase de démarrage

Nous avons déjà rappelé dans l'introduction comment le système d'exploitation GNU/Linux se mettait en place et à quoi servaient les différents éléments que sont :

– le BIOS ;
– le *bootloader* (ou chargeur au démarrage) ;
– le noyau linux et ses modules ;
– l'image `initrd` (aussi appelée *initramfs*) ;
– les pseudo-systèmes de fichiers `/proc` et `/sys`.

1.1 Sécurisation du BIOS

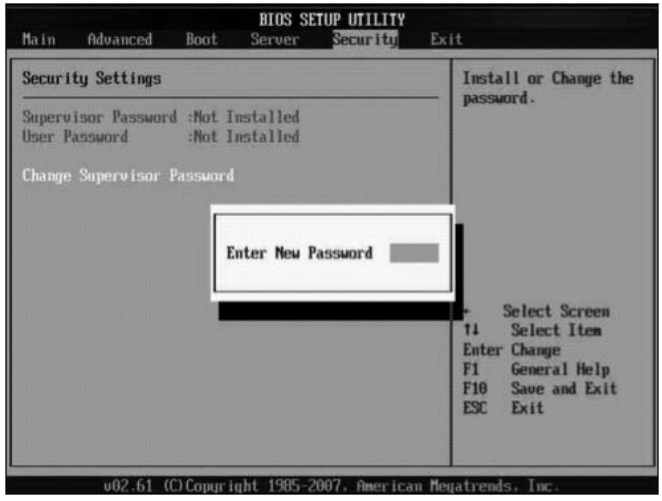
Le BIOS, comme il été rappelé lors de l'introduction, est un système élémentaire d'entrées/sorties composé d'un ensemble de fonctions, elles-mêmes contenues en zone mémoire morte (aussi appelée ROM ou ***R**ead-**O**nly **M**emory*), de la carte mère d'un ordinateur. Ces mécanismes permettent d'effectuer certaines opérations de vérifications de l'état du matériel (mémoire, disque dur, etc.), lors de la mise sous tension de l'équipement. Chaque serveur, poste de travail ou portable possède son propre BIOS. Il existe de grandes familles de fabricants de BIOS tels que :

– Asrock ;
– LSI, Tekram ;
– American Megatrends Incorporated.

Le site Internet http://www.filehelp.fr/drivers/bios liste les principaux acteurs de ce marché et il est possible d'avoir quelques précisions quant à telle ou telle carte mère. La seule véritable protection de ce sous-système consiste à positionner un mot de passe en ouvrant le menu principal du BIOS de sa machine et en allant dans le menu approprié. Il faut préciser que seuls les équipements dont l'accès n'est pas restreint et sont faciles à emporter, doivent être considérés. En effet, il n'y a aucun intérêt à mettre un mot de passe sur le BIOS d'un serveur, faisant partie d'un ensemble d'équipements au sein d'une salle machine, ou même des postes de travail dont l'accès est sécurisé dans les bureaux d'une entreprise. Cela n'a d'intérêt que pour protéger des portables pouvant être facilement perdus ou volés, d'autant qu'aujourd'hui, dans les entreprises, le personnel (internes et/ou prestataires) est tracé à l'aide de badge ou d'un moyen permettant d'identifer la personne. Aussi, tout acte malveillant comme un vol de matériel serait automatiquement détecté et le ou la coupable sanctionné.

Exemple

BIOS American Megatrends avec un mot de passe positionné :



On cherche alors principalement à empêcher toute modification des paramètres du BIOS, de la part d'un éventuel intrus pouvant alors configurer celui-ci, de sorte à démarrer à partir d'une disquette ou d'un CD, et pouvant alors entrer dans le mode secours ou mono-utilisateur (appelé aussi *single user*). Le pirate peut alors lancer divers processus depuis le système ou même, copier les données s'y trouvant stockées. Le fait de protéger le BIOS par un mot de passe permet principalement d'empêcher les intrus de pouvoir démarrer le système sans avoir, au préalable, saisi ledit mot de passe.

Comme nous l'avons dit ci-avant, il existe autant de méthodes d'activation du mot de passe au sein du BIOS que de fabricants. Aussi, il est conseillé de consulter le manuel de la machine afin d'obtenir des instructions précises concernant le paramétrage du BIOS de celle-ci. Une fois cette première barrière activée (ou non), on peut alors passer à la sécurisation du chargeur au démarrage (ou *bootloader*).

1.2 Présentation de GRUB

La phase de démarrage est très importante et l'on est alors amené à utiliser un certain nombre d'outils qu'il convient de fiabiliser avant leur utilisation en production. Nous allons notamment revenir sur la configuration du *bootloader* (ou chargeur au démarrage), afin de le rendre plus restrictif, et l'on enchaînera sur le chiffrement de partition et sur l'utilisation de volumes logiques, permettant de cloner les données qui s'y trouvent hébergées. Il existe également certaines fonctionnalités qu'il est préférable de ne pas laisser entre toutes les mains. Nous verrons alors comment désactiver ces « *Magic SysKeys* ». Comme nous l'avons déjà souligné, l'utilitaire GNU GRUB (signifiant ***GR****and* ***U****nified* ***B****ootloader*), est un programme d'amorçage de micro-ordinateur, généralisé aux serveurs d'entreprises. Il s'exécute lors de la mise sous tension de la machine, après les séquences de contrôle interne, et avant le système d'exploitation proprement dit. Son rôle est essentiellement d'organiser le chargement du système d'exploitation.

Remarque

Lorsque l'ordinateur héberge plusieurs systèmes, on parle alors de multiamorçage. Il permet à l'utilisateur de sélectionner celui sur lequel il souhaite travailler.

Cet outil est un logiciel libre permettant l'amorçage des systèmes GNU/Linux ou même Microsoft Windows. Mais, il est également adapté aux systèmes d'exploitation moins répandus tels que FreeBSD, OpenBSD, Hurd ou Solaris. Il permet aussi la lecture de la configuration au démarrage. Il n'est donc pas nécessaire de réinstaller GRUB dans le secteur d'amorçage après une modification de la configuration, contrairement à l'ancien mécanisme de bootloader qu'était LILO. On peut l'invoquer via une ligne de commande permettant de changer la configuration au démarrage. Ainsi, GRUB reconnaît nativement les divers systèmes de fichiers du marché. De plus, l'outil possède même un langage de base de commandes simples, permettant aux administrateurs de récupérer un amorçage qui se serait mal déroulé, à la suite d'un mauvais adressage d'une partition ou d'un mauvais paramétrage.

Remarque

IMPORTANT : du fait de sa compatibilité avec l'ensemble des systèmes de fichiers et du nombre important de fonctionnalités, GRUB est par contre beaucoup plus volumineux que son ancêtre LILO.

Lors du démarrage du serveur, le BIOS cherche le premier périphérique « bootable ». Habituellement, il s'agit du disque dur, mais les options du BIOS peuvent avoir configuré un lecteur CD/DVD ou une clé USB. Ce secteur sera chargé en tant que secteur d'amorçage (également appelé ***M**aster **B**oot **R**ecord* ou MBR), correspondant aux 512 premiers octets du disque sélectionné (ou de tout autre périphérique bootable), et transfère alors le contrôle au code chargé en mémoire. L'opération se décompose alors en deux phases :

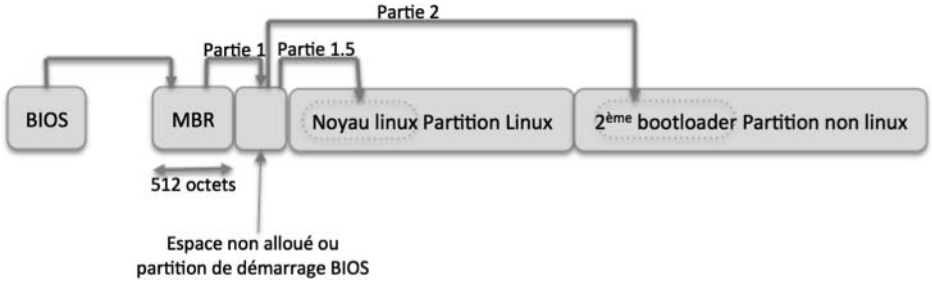
– GRUB partie 1
– GRUB partie 1.5

GRUB Partie 1

Le fichier `boot.img` (appelé partie 1) est stocké dans le MBR. Ceci dit, ce dernier peut aussi contenir un autre *bootloader* pouvant chaîner la partie 1 de GRUB depuis un autre secteur de boot (début d'un autre disque ou d'une autre partition logique type DOS, par exemple).

GRUB Partie 1.5 ou Partie 2

Du fait de la faible taille du MBR, la partie 1 se contente alors de charger la partie 2 ou la partie 1.5 (se trouvant dans les 30 kio juste après le MBR). C'est la partie 1.5, contenant des pilotes permettant l'accès à la partie 2, qui charge alors ce second segment de code.



Depuis quelque temps déjà, le logiciel GRUB a évolué, en passant de la version GRUBv1 à la version GRUBv2, afin de mieux maîtriser le processus de démarrage. Dans cette seconde édition, le programme fonctionne sensiblement de la même façon, excepté que le MBR peut charger un secteur depuis n'importe quelle adresse normalisée LBA48.

Cela favorise le chargement du premier secteur du fichier `core.img`, généré depuis `diskboot.img`. Puis, ce premier secteur est ensuite utilisé pour charger le reste du fichier généré par `core.img`.

Remarque

Le fichier `core.img` est alors normalement stocké au même emplacement que la partie 1.5, en gérant les mêmes problématiques de taille. Toutefois, il peut être déplacé plus facilement, dans une partition ou un système de fichiers moins contraints aux déplacements ou à l'omission de cette partie 1.5. Une fois chargé, le fichier `core.img` charge à son tour les fichiers de configuration ainsi que les autres modules nécessaires au bon fonctionnement du système d'exploitation cible.

Une fois GRUB chargé, il présente alors une interface permettant à l'utilisateur de sélectionner le système d'exploitation souhaité. L'interface prend généralement la forme d'un menu graphique. Toutefois, si celui-ci n'est pas accessible ou que l'utilisateur préfère un contrôle direct, GRUB embarque également sa propre invite de commande. L'utilisateur peut donc préciser manuellement les différents paramètres de démarrage et régler ainsi l'outil pour charger automatiquement un noyau linux en particulier, après un délai d'attente, défini également de ses mains. Une fois les options de démarrage sélectionnées, le *bootloader* charge le noyau souhaité en mémoire et lui transfère alors le contrôle.

Astuce

GRUB a également la possibilité de transférer le contrôle à un autre chargeur au démarrage, utilisant, lui aussi, le chargement en chaîne. Cette technique est utilisée pour charger les noyaux de type Microsoft Windows et ne supporte pas le multidémarrage standard. Dans ce cas, il faut effectuer des copies des autres bootloaders.

Ainsi, au lieu d'un noyau, l'autre système est alors chargé comme s'il l'avait été depuis le MBR et il peut s'agir d'un autre chargeur au démarrage, comme le menu de démarrage de Microsoft Windows, permettant la sélection du système d'exploitation. En fait, à l'inverse de LILO, l'utilitaire GRUB n'a nul besoin d'être installé dans le MBR, lors de chaque modification du fichier de configuration. Dans un système GNU/Linux, la commande *grub-install* est uniquement utilisée pour installer la partie 1 du programme dans le MBR ou dans une autre partition. Les fichiers de configuration doivent se trouver sur une partition utilisable. Dans le cas contraire, la partie 1 exécute l'interpréteur de commandes automatiquement. Le nom et l'emplacement de ce fichier de configuration peuvent varier d'un système à l'autre. Sur une distribution Debian, ce fichier se situe sur le répertoire `/boot/grub`. Il peut être nommé `grub.cfg`, `grub.conf` ou encore `menu.lst` selon les versions de l'outil présentes sur le système.

Remarque

Lorsqu'on mentionne GRUB, on pense à GRUB legacy (c'est-à-dire la première version du logiciel), alors que GRUB 2 fait référence à GRUBv2.

On peut tout à fait créer sa propre disquette ou clé USB de démarrage en copiant les fichiers correspondant à l'image utilisée pour lancer GRUB (Stage1) et l'image du noyau de GRUB (Stage2), sur les deux premiers blocs respectifs du périphérique cible :

```
# cd /usr/share/grub/<Architecture>
# dd if=stage1 of=/dev/sd0 bs=512 count=1
1+0 record in
1+0 record out
# dd if=stage2 of=/dev/fd0 bs=512 seek=1
153+1 records in
153+1 records out
```

Afin de pouvoir s'adresser au programme `grub`, il faut connaître les éléments à passer en paramètre, à savoir :

– le périphérique où se trouve le noyau (aussi appelé `kernel`)
– le nom de ce noyau
– le nom du programme d'initialisation (appelé `initrd`)
– la commande de "`boot`" d'initialisation de l'exécution

Grâce au prompt interactif, GRUB gère parfaitement la complétion des caractères. Ceci permet notamment de retrouver les noms de disque, de noyau ou encore du programme initiateur. Ceci est un gros avantage, surtout si l'on considère qu'il peut être compliqué de se souvenir de ces noms dans le cas où le chargement ne fonctionne plus. Le fichier de paramétrage `/boot/grub/menu.lst` (ou `grub.conf`, dans certains cas) contient les éléments optionnels de couleur et de délai nécessaire au menu d'affichage. On y trouve également, à la suite les uns des autres, différentes sections décrivant les différents noyaux disponibles que l'on peut démarrer :

Exemple

Fichier classique de menu.lst :

```
default 0
timeout 5
#
foreground = ffffff
background = 000000
#
splashimage=(hd0,1)/boot/grub/leaf_splash.xpm.gz
#
title Gentoo
root (hd0, 1)
```