Le chiffrement de disque sous linux : Vrai ou faux sentiment de sécurité ?

Kevin DENIS *kevin2nis@gmail.com*,

Ingénieur CNAM, option Réseaux et Systèmes Informatiques,

Auteur d'articles pour MISC et Linux magazine.

Conférence donnée le mercredi 7 juillet 2010, aux 11émes RMLL, Rencontres Mondiales du Logiciel Libre :
`http://2010.rmll.info`

URL de la conférence :
`http://2010.rmll.info/Le-chiffrement-de-disque-sous-linux-vrai-ou-faux-sentiment-de-securite.html`

Table des matières

Abstract de l'article :

Cet article vise à démontrer que le chiffrement de disque sous linux, réalisé via l'utilisation du module noyau dm-crypt et du programme cryptsetup, fait l'objet d'une faille conceptuelle. En effet, le programme déchiffrant le disque réside sur une partition en clair. Il s'ensuit qu'un attaquant peut le modifier à sa guise pour obtenir le mot de passe des partitions chiffrées, ou pour placer un programme malveillant suite au déchiffrement des partitions.

Pour répondre à cette faille, le trusted computing est nécessaire, plus particulièrement la puce TPM et ses mesures des métriques du démarrage. Ainsi, les outils utilisant cette puce (TrustedGrub, trousers et tpm-tools) permettent de présenter une méthode novatrice pour obtenir un boot sécurisé d'une machine. Cette méthode assure que système démarré est bien un système sain et non un système modifié par un attaquant.

Mots-clé : *Linux, sécurité informatique, chiffrement, dm-crypt, TPM, trusted computing, TrustedGrub, trousers, tpm-tools, Evil Maid.*

1 Introduction

La sécurité d'une machine est faible lorsqu'elle est physiquement accessible. En effet, n'importe quel utilisateur peut démarrer la machine en ajoutant au bootloader la commande `init=/bin/bash`, lui donnant ainsi immédiatement des droits root. Une autre méthode tout aussi simple consiste à démarrer depuis un liveCD et de monter les partitions afin de lire les données, les modifier, ou ajouter des programmes malveillants.

Le chiffrement de disque est une réponse fiable pour éviter ces risques. Il protège les données lorsque le disque est éteint et empêche donc toute perte de confidentialité et d'intégrité des données enregistrées sur celui-ci. Les ordinateurs portables sont une cible évidente, du fait de leur risque de vol ou perte. Le voleur aura le matériel, mais il n'accédera pas aux données. Les machines de bureaux peuvent également être chiffrées car il est courant que de multiples personnes gravitent autour de celles-ci. Un collègue indélicat peut très simplement devenir administrateur de votre machine. Un troisième type de machine pour lesquelles le chiffrement joue un rôle important sont les machines virtuelles. Leur disque dur n'est qu'un fichier situé généralement sur un datastore ou sur un partage réseau. Toute personne accédant à ce partage accède aussi à vos données. Les seules machines n'ayant rien à gagner à être chiffrées sont les serveurs allumés 24h/24 situés dans des pièces fermées où les accès sont surveillés et maîtrisés.

Cet article est construit en deux parties. La première explique comment le chiffrement de disque sous linux est rendu possible en justifiant le choix de dm-crypt. Des exemples d'utilisation sont donnés, ainsi qu'un procédé original de *plausible deniability*[1]. La critique majeure de ce système est ensuite étudiée : le programme de déchiffrement de la partition peut très facilement être remplacé par un autre, maîtrisé par l'attaquant, laissant toute latitude à ce dernier pour dupliquer le mot de passe de déchiffrement ou pour toute autre action malveillante. La deuxième partie explique l'apport majeur du *trusted computing*. En effet, la puce TPM est indépendante du système d'exploitation et permet de valider les programmes lancés lors du boot. Comme aucune méthode n'existe, j'ai inventé une manière de valider le démarrage de manière sécurisée à l'aide de TrustedGrub et des outils tpm-tools. Cette méthode combine l'utilisation d'une clé utilisateur et des métriques du boot faites par la puce TPM. Enfin, cette méthode est vérifiée pour valider son niveau de sécurité.

1. Procédé empêchant un attaquant de prouver qu'une méthode cryptographique est utilisée pour cacher des données,
`http://en.wikipedia.org/wiki/Plausible_deniability`

2 Mise en œuvre du chiffrement sous linux

Le chiffrement des données peut se réaliser de deux façons distinctes. La première consiste à chiffrer manuellement chaque fichier confidentiel : cette solution a pour inconvénient de devoir trouver des programmes compatibles et d'obliger à chiffrer/ déchiffrer/ rechiffrer à chaque opération et de n'oublier aucun fichier. La seconde consiste à chiffrer l'intégralité des partitions au niveau du pilote de périphérique. C'est cette seconde méthode qui est exposée dans cet article.

Cette première partie fournit les raisons qui penchent en faveur de l'utilisation de cryptsetup [2], pour enchaîner sur la manière dont le chiffrement fonctionne à l'aide du module noyau `dm-crypt`, *device mapper crypt*, et du programme `cryptsetup` qui permet de paramétrer le chiffrement. Divers cas d'utilisation sont donnés, principalement celui permettant le Full Disk Encryption (FDE, ou chiffrement complet de disque) et une présentation innovante de plausible deniablity est expliquée en utilisant simplement ces outils. Plusieurs attaques seront ensuite menées contre ce procédé de chiffrement, avec les contre-mesures associées. Ces attaques montreront que l'utilisation seule d'un procédé de FDE n'est pas suffisant pour garantir une sécurité suffisante aux données stockées sur le disque.

2.1 Pourquoi utiliser dm-crypt et cryptsetup

Un certain nombre de procédés de chiffrement de volumes, partitions, disques, ou images disques, sont disponibles sous linux. Une liste relativement complète est fournie par wikipedia : `http://en.wikipedia.org/wiki/Comparison_of_disk_encryption_software` [3].

Pour les partitions, les documentations mentionnent généralement :

- Cryptoloop et Loop-AES : deux programmes qui sont aujourd'hui considérés comme obsolètes. Ils permettent d'utiliser une image disque (loop) chiffrée.
- TrueCrypt : de `http://www.truecrypt.org`, un logiciel open-source permettant de chiffrer des partitions ou des images disques. Il fonctionne sous linux, Mac OS et Windows.
- Cryptsetup : `http://www.saout.de/misc/dm-crypt/`, qui se compose d'un module intégré au noyau linux, et d'un programme permettant de l'utiliser. Ce programme bénéficie d'une extension LUKS signifiant *Linux Unified Key Setup*. Cryptsetup permet de chiffrer des images disques via un périphérique loop, ou des partitions.

Le choix de Cryptsetup pour chiffrer un disque est naturel. En effet, il a été créé en remplacement de cryptoloop [4] : *"Device-mapper is a new*

2. `http://www.saout.de/misc/dm-crypt/`
3. Tous les OS sont compris dans cette liste, pas uniquement linux.
4. `http://www.saout.de/misc/dm-crypt/`

infrastructure in the Linux 2.6 kernel that provides a generic way to create virtual layers of block devices that can do different things on top of real block devices. dm-crypt is such a device-mapper target that provides transparent encryption" Alors que cryptoloop ou loopAES étaient plutôt orientés dans le chiffrement des images disques, cryptsetup a été écrit pour chiffrer des partitions. Cryptsetup utilise un module noyau, dm-crypt, intégré depuis longtemps dans le noyau standard. Enfin, la troisième raison et sans doute la meilleure, reste la raison du choix des distributions. En effet, toutes les distributions les plus courantes proposent dès l'installation de chiffrer le disque à l'aide de cryptsetup.

Le chapitre suivant explique comment fonctionnent dm-crypt et cryptsetup.

Biblio
`http://www.saout.de/misc/dm-crypt/`
`http://alien.slackbook.org/dokuwiki/doku.php?id=slackware:setup`

2.2 Fonctionnement de dm-crypt

Le chiffrement de disque est effectué par le noyau via son module dm-crypt, son paramétrage se fait via le programme cryptsetup. De manière analogue aux autres extensions device-mapper [5], dm-crypt est une couche supplémentaire située entre le filesytem et l'écriture sur le disque physique. Si la clé de déchiffrement est présente, alors les écritures et lectures sur le disque sont chiffrées et déchiffrées au vol.

2.2.1 Le programme cryptsetup et le module dm-crypt

Cryptsetup est le composant côté administrateur permettant d'utiliser dm-crypt. Son fonctionnement est simple :

```
cryptsetup [--cipher <cipher>] [--key-file <file>] create $NAME
    /dev/$PARTITION
```

- Le *cipher* est à choisir parmi la liste des méthodes de chiffrement connues du noyau : `ls /lib/modules/‘uname -r‘/kernel/crypto` ou bien : `zgrep CRYPTO /proc/config.gz`. Par défaut, l'AES 256 est utilisé.
- Si le fichier de clé est omis, alors un mot de passe est demandé de manière interactive sur la ligne de commande.
- Le mot clé *create* permet de créer ou d'ouvrir le fichier block spécial `/dev/mapper/$NAME` qui est la vision déchiffrée de `/dev/$PARTITION` par la clé fournie.

Ce mode de fonctionnement est très simple à appréhender. Les octets sont toujours chiffrés sur le disque, et déchiffrés à la volée lors des accès systèmes. Cela remplit parfaitement les objectifs en matière de sécurité. Si un attaquant

5. `http://sourceware.org/dm/`

récupère le disque sans le mot de passe, il n'aura pas accès aux données. Cette manière de faire est également parfaitement indépendante du système de fichiers, des données ou de l'utilisation de la partition. dm-crypt ne s'occupe que du flot d'octets sans s'intéresser à son contenu. Le système linux *communique* avec la version en clair des données.

Toutefois, cette méthode présente plusieurs limites. Le mot de passe est fourni par un utilisateur. Si l'utilisateur n'y prend pas garde, il peut utiliser un mot de passe faible, permettant ainsi à un attaquant de réaliser une attaque par force brute sur la partition. Ensuite, il n'existe qu'une clé unique, le mot de passe. Si l'utilisateur le perd ou souhaite le changer, il n'existe aucun moyen de le faire si ce n'est tout sauvegarder, de recréer un mapping avec nouveau mot de passe et de réinjecter la sauvegarde. Enfin, il n'existe aucun moyen de vérifier ou non la validité du mot de passe. Le chiffrement étant effectué par flot, tout mot de passe est accepté, seul le résultat du déchiffrement est incohérent en cas de mot de passe erroné.

2.2.2 L'extension LUKS de cryptsetup

Cryptsetup s'est donc vu adjoindre LUKS qui signifie Linux Unified Key Setup pour répondre à ces limitations.

```
cryptsetup <luks action> [Argument]
```

Les actions et arguments sont fournis par la page de manuel.

Les ajouts de LUKS sont les suivants : tout d'abord, LUKS se réserve un espace en en-tête de partition. Cet espace est constitué de huit slots, et du mot de passe de chiffrement de la partition généré par cryptsetup de manière à ce qu'il rende les attaques par dictionnaire ou force brute inutiles. Chacun des slots est protégé par mot de passe et son accès permet l'utilisation de la clé de déchiffrement qui ouvre la partition.

Les avantages sont intéressants : pas de risque d'attaque de force brute sur les données de la partition [6], possibilité simple d'ajouter ou supprimer un mot de passe. LUKS est également capable de vérifier la validité d'un mot de passe ou pas. Enfin, l'en-tête connu des partitions LUKS permet à différents systèmes de réagir correctement lors de la découverte de celui-ci [7].

2.2.3 Le chiffrement de disque complet

L'utilisation du chiffrement pour une distribution est simple. Le disque est généralement coupé en deux partitions. La première, de petite taille, `/boot` contient le bootloader, le noyau ainsi qu'un initrd. La seconde est chiffrée via LUKS. Pour éviter d'utiliser autant de mots de passes que de partitions, les distributions utilisent généralement un chiffrement au dessus

6. Il reste à l'attaquant les mots de passe des slots à forcer.

7. Lors de l'insertion d'une clé USB chiffrée via LUKS, un environnement de bureau pourra le détecter et présenter un pop-up demandant le mot de passe, par exemple.

d'un LVM. Ainsi, un mot de passe unique permet le déchiffrement du LVM, qui lui-même est découpé en plusieurs partitions : la `racine (/)` le swap et `/home` par exemple. Lors du boot, le ramdisk initial détecte la partition chiffrée et demande un mot de passe à l'utilisateur. Si le mot de passe est correct, alors le déchiffrement a lieu, le *mapping* est réalisé et le démarrage peut continuer.

Pour un utilisateur, la situation semble idéale. Le disque est chiffré de manière transparente. Le chiffrement ne consomme que très peu de puissance CPU [8]. Lorsque la machine est éteinte, elle peut être perdue sans qu'il n'existe le moindre risque que les données présentes sur le disque soient dévoilées [9]. Bien entendu sous réserves que l'administrateur a choisi un *cipher* suffisement fort et un mot de passe non trivial.

2.2.4 Plausible deniability

Le plausible deniability est une notion qui a été popularisée par Truecrypt [10]. Le principe repose sur l'impossibilité pour un attaquant de prouver que des données chiffrées sont présentes sur un disque [11]. TrueCrypt permet de placer des données chiffrées à l'intérieur d'un premier conteneur chiffré. Ainsi, un utilisateur forcé à révéler son mot de passe peut le fournir, puisque ce mot de passe n'ouvrira que le premier conteneur, n'ayant que des données pseudo-confidentielles.

Sous certaines conditions précises, cryptsetup permet aussi de faire de la plausible deniability.

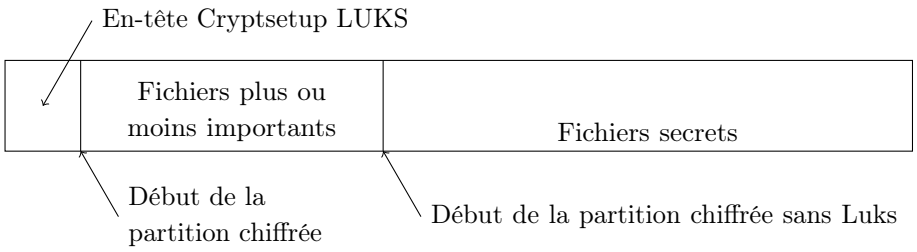
Tout d'abord, il faut créer un conteneur, partition réelle, ou fichier représentant un disque. Ce disque est rempli préalablement de données aléatoires. Sur ce disque, un chiffrement de type cryptsetup Luks est créé. Ensuite, il faut créer un deuxième conteneur bien plus loin sur le disque, ce qui est aisément réalisable grâce à l'outil `losetup` qui permet de démarrer à un `offset` précis. A cet `offset`, un second conteneur. Personne ne peut distinguer à ce second offset des données chiffrées de l'aléatoire :

8. `http://www.saout.de/tikiwiki/tiki-index.php?page=UserPageChonhulio` par exemple qui donne des réductions de temps de lecture/écriture brute assez faible.

9. Il ne faut pas négliger non plus le problème de fin de vie des machines. Si le disque est chiffré, il est alors possible de distribuer les machines à des personnes tierces (associations, amis) sans que les données soient *'at risk'* après un simple formatage.

10. `http://www.truecrypt.org/docs/?s=plausible-deniability`

11. Un des principes d'un chiffrement fiable repose sur l'impossibilité de différencier un flux chiffré d'un flux aléatoire.



Les précautions à prendre sont les suivantes :
- Ne pas utiliser le premier conteneur. En effet, le premier conteneur écrira sur l'intégralité du disque si on y prête pas garde.
- Le choix du système de fichiers n'est pas libre pour le premier conteneur. Par exemple, *ext* duplique son superblock en plusieurs endroits du disque. Si ce superblock disparaît à partir d'un seuil, cela peut soulever des doutes. Le filesystem FAT n'a pas ce problème.
- Ne pas utiliser LUKS pour le second conteneur. En effet, l'en-tête Luks est un en-tête connu. S'il est détectable en milieu de partition, alors cela signifie que des données sont présentes. L'utilisation de cryptsetup seul est obligatoire.

Pour ces raisons, il est conseillé d'utiliser un fichier plutôt qu'une partition. Sous les réserves exposées, la plausible deniability est parfaite, et personne ne peut prouver qu'il existe un second conteneur dans le premier ayant des données secrètes.

Biblio
Article linux magazine numéro 101 : *"Le chiffrement de disque sous linux"*, par Kevin DENIS.

2.3 Situation malgré tout insatisfaisante

Le chiffrement de disque est mis en œuvre comme indiqué dans la documentation. Les attaquants 'naïfs' peuvent chercher à attaquer AES ou le procédé de chiffrement utilisé, mais je considère dans cet article que les choix fait par l'administrateur sont corrects. Je laisse également de côté les keyloggers hardware. Le risque est réel sur les machines de bureau, bien plus faible sur les portables du fait de leur miniaturisation rendant ce genre d'ajout difficile. Dès lors, deux types d'attaques restent réalisables, cold boot attack et les attaques en deux temps.

2.3.1 Cold Boot attack

Une attaque indirecte a été développée par l'université de Princeton : `http://citp.princeton.edu/memory/` et repose sur la rémanence des données en RAM. En effet, la clé de déchiffrement est en mémoire afin de pouvoir accéder aux données. Un attaquant face à une machine aux disques chiffrés

et allumée peut rebooter brutalement la machine sur un autre OS, et dumper ensuite le contenu de la RAM pour retrouver à l'intérieur la clé. Contrairement à une croyance courante, la RAM n'est pas remise à zéro lors de l'extinction d'une machine. L'attaque a été perfectionnée en noyant les barrettes de RAM dans de l'azote liquide, ce qui pousse la rémanence des données à l'intérieur de celle-ci à plus de dix minutes.

Différents proof of concepts ont été développés, ciblant la majorité des procédés de chiffrement sous différents OS. Ceci dit, le principe du chiffrement n'est pas vraiment remis en cause. Il est acquis que le chiffrement de disque protège les données lorsque le disque est éteint. Cette attaque modifie légèrement ce principe : Le chiffrement protège les données lorsque le disque est éteint depuis plusieurs instants.

Il reste toutefois une autre surface d'attaque, bien plus problématique. En effet, le programme réalisant le déchiffrement de la partition repose sur une partition en clair, non protégée !

2.3.2 Attaques en deux temps

Un attaquant peut contourner le chiffrement en deux temps. Tout d'abord, remplacer le programme de déchiffrement par un autre qui enregistre le mot de passe fourni quelque part, puis déchiffre ensuite les partitions [12].

Ce risque n'est pas que purement théorique. Joanna Rutkowska, chercheuse en sécurité du poste de travail l'a non seulement démontré, mais à de plus écrit un programme réalisant ces actions en visant TrueCrypt sous Windows [13]. Le risque est identique sous linux, et un *proof of concept* a été publié dans MISC N ° 48 consistant à modifier l'initrd d'une distribution debian afin de logger sur la partition `/boot` le mot de passe.

Si nous reprenons les cas d'usage donnés en introduction, il est clair qu'ils sont tous impactés par cette faille. Il suffit de pouvoir accéder deux fois à une machine. Un ordinateur portable est souvent laissé sans surveillance : Joanna Rutkowska prend comme exemple un conférencier qui dépose son ordinateur portable dans une chambre d'hôtel et une femme de ménage malicieuse : EvilMaid qui a donc accès plusieurs fois à l'ordinateur. Une machine desktop reste dans un bureau alors que des personnes gravitent encore autour. La machine virtuelle est identiquement impactée par l'administrateur du datastore. La source du risque est clairement identifiable : L'utilisateur fait confiance au programme à qui il fournit la clé mais ce programme n'a jamais été vérifié.

Une autre famille de failles s'appelle les MBR attacks [14], basées sur la

12. Cette faille conceptuelle touche linux, mais également tous les OS qui chiffrent leurs disques !

13. `http://theinvisiblethings.blogspot.com/2009/10/evil-maid-goes-after-truecrypt.html`

14. Voir biblio

même logique, sauf qu'au lieu de modifier l'initrd de linux, il est modifié le MBR de la machine dans le même but : logger le mot de passe quelque part.

Afin d'éviter ces failles, il devient nécessaire de pouvoir s'assurer que le programme de déchiffrement n'a pas été modifié. Les vérifications habituelles, hachage ou signature sont battues en brèche pour les mêmes raisons qu'évoquées précedemment. Le programme de vérification peut également être modifié par l'attaquant pour produire une signature valide.

Il s'ensuit que lors d'un vol de portable, il n'est nullement garanti que les données confidentielles le restent... En effet, si le vol est consécutif à une attaque EvilMaid, la confidentialité est rompue. Et comment l'administrateur peut-il savoir après le vol si la machine avait été infectée ? Comment l'administrateur peut-il faire confiance à sa machine de bureau ? Il est tout à fait plausible qu'un attaquant ait pu ajouter un rootkit alors que l'administrateur avait éteint sa machine [15]. La plausible deniability ne répond pas non plus à ces failles. Puisques les programmes de déchiffrement de disque sont considérés comme étant sous l'emprise d'un attaquant, la plausible deniability sera pareillement déjouée.

2.3.3 Les *poor man solutions*

Comme l'indique Joanna Rutkoswka sur son blog [16], il n'existe que des *poor man solutions*. Un utilisateur avisé et isolé pourra développer un bout de code ou une méthode non connue de l'attaquant, lui permettant ainsi de constater que son démarrage est modifié. Mais l'inconvénient de cette méthode est qu'elle ne passe pas à l'échelle, que ce soit dans les scripts d'installation d'une distribution ou une politique d'entreprise. La sécurité par l'obscurité n'est jamais la bonne solution. Il est aussi possible d'utiliser une clé USB contenant le bootloader, le noyau et l'initrd, mais cette clé doit être surveillée en permanence. Les protections types mots de passe BIOS ne sont pas non plus suffisantes. Des mots de passe administrateurs permettent d'outrepasser les mots de passe admins, informations librement consultables par internet, ce qui n'empêche pas non plus le démontage / remontage des machines.

2.3.4 Quelle solution fiable ?

Une autre solution plus radicale consiste à ne jamais laisser sa machine sans surveillance mais dans ce cas, on peut s'interroger sur l'utilité du chiffrement...

La solution rigoureuse serait qu'une entité tierce, non maîtrisable par l'attaquant puisse valider que le programme de déchiffrement est sain. Cette méthode existe et s'appelle le trusted computing.

15. Le pirate n'a que l'embarras du choix. Par exemple un initrd modifié pourrait ajouter une clé dans le fichier `/root/.ssh/authorized_key`...

16. Voir liens en biblio

Biblio
MISC N ° 48, Boot sécurisé, page 55 et suivantes.
http://theinvisiblethings.blogspot.com/2009/01/why-do-i-miss-microsoft-bitlocker.html
http://theinvisiblethings.blogspot.com/2009/10/evil-maid-goes-after-truecrypt.html
http://en.wikipedia.org/wiki/Cold_boot_attack
http://www.trustedcomputinggroup.org/files/temp/6452209B-1D09-3519-AD815636FC36C5CF/PlatformResetAttackMitigationSpecification.pdf

3 L'ajout nécessaire du trusted computing

Le chapitre précédent a montré que le chiffrement de disque présente une faille conceptuelle. De plus cette faille ne peut être corrigée par une évolution des méthodes de chiffrement ou une évolution du programme réalisant ce chiffrement. Pour combler cette faille il est nécessaire que le programme déchiffrant le disque soit validé comme sain.

3.1 Le trusted computing : la puce TPM

3.1.1 Le TCG et la puce TPM

Le Trusted Computing Group (TCG) est un regroupement d'industriels qui souhaite promouvoir l'installation et l'utilisation d'une puce dédiée sur les cartes mères des machines. Cette puce s'appelle la puce TPM pour Trusted Platform Module. Ce processeur cryptographique permet de signer ou chiffrer des données, et conserve matériellement les clés réalisant ces opérations. Il est impossible pour un attaquant d'extraire ces clés car elles ne sortent jamais de la puce TPM. Les opérateurs, aussi bien attaquants qu'utilisateurs légitimes, ne peuvent qu'utiliser les services de cette puce. Cette puce équipe des machines de type serveurs ou bien des ordinateurs portables [17].

Le paragraphe qui suit est une traduction personnelle du site web du Trusted Computing [18] :

> Le Trusted Computing est un type de technologie développé et dont la promotion est faite par le Trusted Computing Group. Le mot Trust (confiance) est utilisé dans le sens de "trusted systems" ou systèmes de confiances.
>
> Trusted Computing est la réponse de l'industrie aux nombreux problèmes de sécurité présents en entreprise et est construite sur une racine de confiance hardware (Hardware root of trust).

17. L'embarqué et certaines consoles de jeux emploient aussi ce type de puce.
18. http://www.trustedcomputinggroup.org/trusted_computing

> Depuis cette base de confiance, les sytèmes, applications et réseaux pourront être construits plus sûrs. Avec le Trusted Computing, les machines et systèmes vont se comporter de manière consistante, et ces comportements seront vérifiés par le hardware et le software lorsque cette technologie sera activée.
>
> La technologie Trusted Computing fera des ordinateurs plus sûrs, moins sensibles aux virus, et donc plus fiables. De fait, le Trusted Computing permettra aux systèmes informatiques d'offrir une sécurité et une efficacité accrues.

Ce principe est en effet intéressant. Cette puce permet de disposer d'un outil de mesure neutre et indépendant, qui valide cryptographiquement certaines conditions. La puce étant en dehors du champ d'action de l'ordinateur, il devient possible de construire de la confiance à partir de celle-ci comme une chaîne. Si un maillon est de confiance et qu'il valide le maillon suivant, alors ce maillon est de confiance. La problématique repose donc sur le premier maillon. En effet, on peut vouloir faire confiance en un hash. Mais qui vérifie que le programme calculant le hash est de confiance ? La première confiance réside dans la puce TPM, et les outils dérivant de cette confiance sont à leur tour considérés sûrs.

Cette "confiance" qui est contenue dans une puce crypto à laquelle l'utilisateur n'a pas accès a bien entendu attiré l'attention de nombreux acteurs plus intéréssés par le verrouillage logiciel que par la vérification cryptographique, provoquant des commentaires acerbes des utilisateurs de logiciels libres.

3.1.2 Le point de vue de GNU

GNU, par l'intérmédiaire de Richard Stallman [19] a violemment réagi contre l'introduction des puces TPM sur les cartes mères des ordinateurs. En effet, cette arrivée des puces a fait la promotion des DRM [20] et annonçait l'impossibilité de modifier le système d'exploitation ou les programmes de sa machine [21]. Ces critiques sont parfaitement légitimes.

Depuis, le consortium a évolué. Aujourd'hui, la puce TPM ne sert pas qu'a valider la version d'un navigateur internet ou d'autoriser l'accès à une bibliothèque musicale. Il s'agit d'une puce passive dont le fonctionnement et la mise en route restent sous l'emprise de l'utilisateur. Cette puce TPM va permettre de rendre plus sûr le démarrage d'une machine dont le disque est chiffré, et à ce titre son utilisation est intéressante. De plus en plus de libristes

19. Programmeur et militant du logiciel libre. Il est à l'origine du projet GNU et de la licence publique générale GNU connue aussi sous l'acronyme GPL. Source Wikipedia.

20. Digital Right Management

21. `http://www.gnu.org/philosophy/can-you-trust.html` est le document qui explicite leur point de vue.

proposent l'utilisation raisonnée de cette puce, par exemple une conférence donnée pendant l'édition 2009 des RMLL [22].

Biblio
`http://www.gnu.org/philosophy/can-you-trust.html`
`http://www.trustedcomputinggroup.org/`
`http://2009.rmll.info/IMG/pdf/trustedcomputing.pdf`

3.2 Outils pour utiliser la puce TPM

Avant d'utiliser une puce TPM, il est nécessaire de vérifier sa disponibilité. Les puces TPM ne sont pas présentes sur l'ensemble des machines, de plus elles sont souvent inhibées dans le BIOS.

A titre d'exemple, la machine que j'ai utilisée pour valider la solution dispose d'un menu dans le BIOS appelé *security* et qui permet d'activer cette puce ou de la réinitialiser à zéro. Aujourd'hui les puces doivent utiliser les spécificités TPM1.2.

3.2.1 Module noyau

Une puce TPM est un système cryptographique hardware, et des outils logiciels sont nécessaires pour l'utiliser.

Le noyau linux dispose de plusieurs modules chargés de comuniquer avec cette puce :

```
root@darkstar:~# modprobe -l | grep tpm
kernel/drivers/char/tpm/tpm.ko
kernel/drivers/char/tpm/tpm_bios.ko
kernel/drivers/char/tpm/tpm_tis.ko
kernel/drivers/char/tpm/tpm_nsc.ko
kernel/drivers/char/tpm/tpm_atmel.ko
kernel/drivers/char/tpm/tpm_infineon.ko
```

Un modprobe du module TPM et du module correspondant à la puce physique permet de la rendre visible du noyau et utilisable par des outils de plus haut niveau :

```
root@darkstar:~# modprobe tpm_tis
root@darkstar:~# dmesg | tail -1
tpm_tis 00:0d: 1.2 TPM (device-id 0x1001, rev-id 2)
root@darkstar:~# lsmod | grep tpm
tpm_tis                 9256  1
tpm                    12544  1 tpm_tis
tpm_bios                5532  1 tpm
root@darkstar:~#
```

La machine sur laquelle la commande a été lancée dispose d'une puce TPM en révision 1.2.

22. `http://2009.rmll.info/IMG/pdf/trustedcomputing.pdf`

3.2.2 tcsd, tpm-tools et la librairie tss

Le site web `http://trousers.sourceforge.net/` fournit les liens vers les codes sources du démon chargé de communiquer avec la puce : *tcsd* ainsi que des outils *tpm_* permettant de lui soumettre des commandes à exécuter. Le démon tcsd se lance en root et prend l'identité d'un utilisateur tss, qui n'a aucun privilèges. La compilation, l'installation des programmes et la création d'un utilisateur appelé tss ne posent aucun problèmes.

Généralement, il est nécessaire de créer deux paires de clés RSA. La première paire est l'Endorsment Key (EK), la seconde est le Storage Root Key (SRK). L'utilisation de ces clés peut être protégée par un mot de passe, bien que les documentations préconisent l'utilisation du mot de passe *well-known*, c'est à dire 20 octets de zéros.

```
root@darkstar:~# tcsd
root@darkstar:~# tpm_version
  TPM 1.2 Version Info:
  Chip Version:        1.2.2.10
  Spec Level:          0
  Errata Revision:     0
  TPM Vendor ID:       BRCM
  TPM Version:         01010000
  Manufacturer Info:   4252434d
root@darkstar:~# tpm_takeownership -z -y
root@darkstar:~#
```

La puce est désormais prête à l'emploi. La commande *tpm_ takeownership* n'est à réaliser qu'une seule fois. La clé Owner et la clé SRK sont toutes les deux mises *well-known* (options -z et -y).

La liste des commandes *tpm_* est donnée ici à titre de référence :

```
root@darkstar:~# ls /usr/local/sbin/tpm_*
/usr/local/sbin/tpm_changeownerauth /usr/local/sbin/tpm_setactive
/usr/local/sbin/tpm_clear           /usr/local/sbin/tpm_setclearable
/usr/local/sbin/tpm_createek        /usr/local/sbin/tpm_setenable
/usr/local/sbin/tpm_getpubek
    /usr/local/sbin/tpm_setoperatorauth
/usr/local/sbin/tpm_resetdalock     /usr/local/sbin/tpm_setownable
/usr/local/sbin/tpm_restrictpubek   /usr/local/sbin/tpm_setpresence
/usr/local/sbin/tpm_restrictsrk     /usr/local/sbin/tpm_takeownership
/usr/local/sbin/tpm_revokeek        /usr/local/sbin/tpm_version
/usr/local/sbin/tpm_selftest
root@darkstar:~# ls /usr/local/bin/tpm_*
/usr/local/bin/tpm_sealdata   /usr/local/bin/tpm_unsealdata
root@darkstar:~#
```

Une puce TPM ne réalise que peu d'opérations une fois activée. Elle peut sceller ou désceller un bloc de données via les commande *seal* ou *unseal* selon l'état cryptographique de la puce. L'autre moyen d'utiliser la puce se fait via des clés RSA. Une moitié est conservée sur disque, l'autre moitié repose dans la puce TPM. Il n'existe pas d'outil de haut niveau dans la suite d'outils *trousers* pour manipuler ces clés RSA [23].

23. Il semble exister un *engine* openssl pour se faire, mais aucun test ni étude n'ont été réalisés.

3.2.3 Les mesures du TPM : les registres PCR

La puce TPM offre un large éventail de possibilités qui ne sont pas toutes traités ici. Voir les sites suivants, par exemple :
`http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/TrustedComputingTCG.html` ou
`http://www.trustedcomputinggroup.org` ou
`http://2009.rmll.info/Trusted-computing-et-Logiciel.html`
ou le MISC N ° 45.

Le point qui nous intéresse ici concerne le SRTM, ou Static Root Trust Measurement. Le principe en est le suivant : la puce TPM va mesurer chacune des étapes du démarrage. Une mesure est le hash SHA-1 d'un fichier. Ces mesures serviront à vérifier le boot afin d'être sûr que ce boot est identique à une valeur de référence. Un démarrage sain placera toujours la puce dans le même état.

Une puce TPM dispose de plusieurs PCR [24], qui sont des emplacements sécurisés et dont le contenu ne peut être choisi. Ils ne peuvent qu'être mis à jour. Une mise à jour est un hachage de la valeur précédente combiné avec la nouvelle mesure. Ces PCR sont au minimum au nombre de 20. Lors du démarrage de la machine, chaque PCR vaut 0. Certains PCR sont assignés à des rôles particuliers.

Ces PCRs sont garants de l'état de la machine. Comme il n'est pas possible de forcer leur valeur, mais seulement de le mettre à jour, et qu'un hash n'est pas réversible (il n'est pas possible de prédire le résultat d'un hash), alors les PCRs sont identiques d'un démarrage sur l'autre seulement si les mesures sont identiques ce qui signifie que les fichiers sont les mêmes. Donc en se basant sur ces PCRs il est possible d'attester de l'état d'une machine.

3.2.4 TrustedGrub

TrustedGrub [25] est un bootloader basé sur GRUB [26] qui est capable d'utiliser une puce TPM pour updater certains PCR durant le démarrage de la machine :

PCR 4 Contient la mesure du MBR (le stage 1 de Grub). Il va de soi que cette mesure est faite par le BIOS lui-même, avant d'exécuter ce code.

PCR 8 Contient le stage2 partie 1

PCR 9 Contient le stage2 partie 2

PCR 12 Contient la ligne de commande du noyau (la mesure du */proc/cmdline* utilisé) et la ligne d'invocation utilisée par le *menu.lst*.

PCR 13 Contient la liste des fichiers mesurés par Grub (en effet Grub peut aussi hasher un nombre arbitraire de fichiers).

24. Platform Configuration Register
25. `http://www.sirrix.com/content/pages/trustedgrub.htm`
26. `http://www.gnu.org/software/grub/` : GRand Unified Bootloader

PCR 14 Contient la liste des fichiers chargés par Grub : Noyau, initrd, modules etc...

Par exemple, lors d'un démarrage normal :

```
kevin@darkstar:~$ cat /proc/cmdline
root=/dev/sda1 ro vt.default_utf8=0 rootwait
kevin@darkstar:~$ cat /sys/class/misc/tpm0/device/pcrs | grep PCR-12
PCR-12: 64 21 E3 79 31 F8 44 15 CF 73 F2 E1 2C D4 4D B7 EA B5 09 FE
kevin@darkstar:~$
```

Et lors d'un démarrage ou les variables données au noyau ont changées :

```
kevin@darkstar:~$ cat /proc/cmdline
root=/dev/sda1 ro vt.default_utf8=0 rootwait foo=bar
kevin@darkstar:~$ cat /sys/class/misc/tpm0/device/pcrs | grep PCR-12
PCR-12: 35 3A C7 F2 C9 8B 29 B9 C9 03 6E FA FA 25 DA E3 CC 89 26 73
kevin@darkstar:~$
```

Les PCRs sont tous identiques, sauf le 12. Si je compare les deux listes de PCRs :

```
kevin@darkstar:~$ cat /sys/class/misc/tpm0/device/pcrs > ListePCRs-foo
kevin@darkstar:~$ sudo reboot
```

Le reboot est lancé de manière classique, sans différenciation.

```
kevin@darkstar:~$ cat /sys/class/misc/tpm0/device/pcrs > ListePCRs
kevin@darkstar:~$ diff ListePCRs*
13c13
< PCR-12: 35 3A C7 F2 C9 8B 29 B9 C9 03 6E FA FA 25 DA E3 CC 89 26 73
---
> PCR-12: 64 21 E3 79 31 F8 44 15 CF 73 F2 E1 2C D4 4D B7 EA B5 09 FE
kevin@darkstar:~$
```

Ceci montre qu'un boot identique donne bien les mêmes valeurs, et que la moindre modification du boot a des conséquences sur les PCR.

Comme ces métriques sont stables et dépendent du démarrage effectué, il est possible de se baser dessus pour demander à la puce TPM de sceller ou desceller des données. Par exemple, voici l'exemple d'un scellement de données fait via les PCR 4, 8, 9, 12 et 14 et une clé unique et interne à la puce TPM :

```
kevin@darkstar:~$ echo -n "RMLL 2010" | tpm_sealdata -z -p4 -p8 -p9
    -p12 -p14 -o sealed.secret
kevin@darkstar:~$ cat sealed.secret
-----BEGIN TSS-----
-----TSS KEY-----
AQEAAAARAAAABAEAAAABAAMAAQAAAAwAAAgAAAAAgAAAAAAAAAAAAAABAKuw5t+S
IpXKaZBGLN4FBroyNfdxwXUsO9A0tVvOLf3Ga3ho/n5WDtVSNnFTHCHwifMhFoy7
i7h48aUvaBy5CLTIkFEytlQK9cFQkEX6fwGa5AdpJzHyXggi3kJVZsZ93Z6/ysX0
3eja+w9uzrQg4gyMjdlJxxN9X29xzgys/padwdHxIoVzmYJRGdzS0vqHIkGHQe6M
QuKrXfOVgU4ilWtJSm0zd16K/YYjiSWIYN0ePQIG9K8q9/lW4aI4C5XbR11mPTX6
cflessTPcWjMwlQZBb/wH70blIqGzfbyocTpDFQfKxslw5dPQE/otIQR2Ub2IT1E
k3EwZUe54Ulz7J0AAAEAUTJAAS1EnkQJdx9EAULFt0ZcDVJluj36jNN4fgxaNQ4V
PfiJoGR/RwihnqAhK5qDy7WakJa7h2yu65moXtDUXopThoY8PRIXA85w4ktgCZ2t
yvFJvWrY22ffyRHyC3cQeF4cin8I+aJ26Xb9O/B2JTQvJ5Vl0bi4x5/Nn8ZqgMGa
B5tLh9J61q+X18kNl22gFwFhLlnC/Y+7qOZ9H/uj09qwkidypPSMVPSTr6wINgvO
BGV0nKDCd9i3CJWmRTl+hJn4bhdO8nSZVXyb4QU4KXgjNHGeWY+TKwK6/bk5hJ72
IH6TzsZDYi7EuB3MOua2s5EPowAykmMV9KUn+NNB2A==
-----ENC KEY-----
```

```
Symmetric Key: AES-256-CBC
AQEAAAAAACwAAhATIvL43SPvH6j4RZKdnHAaocw2aY0i8vjdI+8fqPhFkp2ccBqh
zDZpjQAAAQBu3l32GZ7wmXWsBGh9dt9wdQzYkcs/Y3iY4IMw6Ind9qvKUD+vTVo6
fE9X98oCDl2EjmecCNmMD3vod+q6XhTqlo5WK994xCMwVvRqjupVXivEoqFMEkv/
fWA3EwEXmf88li8jwQKpRnO+yGYsg4BSqXSe1loRbP2oh0zLmv29BOgocz7HEbgk
u/HQkKKWjFygdJ159TA62S9JkveTgZi5CVHbpN5ywLmmpWB22N8UBPym7NuUiD6i
Z2ghEJliV2dWYV1i2YdYHdple482TAynW4tx8RHsr0QXfjd5i+OFraqsMGdliftv
In056TU88ss3ZB4+I7raGVbUcP2+Lho9
-----ENC DAT-----
welC83t0ad+yYhN5tt6jsg==
-----END TSS-----
kevin@darkstar:~$
```

La clé n'est déchiffrable que lorsque l'état du système est exactement celui utilisé lors du scellement. En effet, dans cet état, la puce TPM déchiffre bien le fichier :

```
kevin@darkstar:~$ tpm_unsealdata -i sealed.secret -z
RMLL 2010
```

Et lorsque le boot est modifié, la clé n'est pas déchiffrée :

```
kevin@darkstar:~$ tpm_unsealdata -i sealed.secret -z
kevin@darkstar:~$ echo $?
24
kevin@darkstar:~$ cat /proc/cmdline
root=/dev/sda1 ro vt.default_utf8=0 rootwait foo=bar
kevin@darkstar:~$
```

Un document d'IBM explique comment utiliser ce principe pour monter un filesystem chiffré si le boot s'est déroulé sans modifications. Mais ce principe peut être facilement étendu pour contrôler l'intégrité du système pendant le démarrage et le déchiffrement de la racine.

Biblio

http://www.sirrix.com/content/pages/trustedgrub.htm
http://trousers.sourceforge.net/
http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaai/ecrypts/liaaiecryptfs.htm
http://www.rsa.com/rsalabs/technotes/tpm/sealedstorage.pdf

3.3 Utilisation des métriques TPM pour vérifier le boot

3.3.1 Aucune méthode n'existe à ce jour

La puce TPM est utilisable sous linux, tout comme le chiffrement de disque. Actuellement, aucune distribution ne propose leur combinaison. Les documentations ne mentionnent pas ce point.

Le point principal de la méthode repose sur les métriques TPM. Le principe en est simple. La clé de déchiffrement du disque sera une clé descellée par le TPM. Si le boot est sain, alors le disque sera déchiffré. Si le boot est altéré, alors la puce TPM n'aura pas des registres PCR correct, la clé de déchiffrement du disque ne sera pas déscellé et le boot stoppera à cet endroit. Voici le résumé le plus succint possible de ce concept :

```
tpm_unsealdata -z -i seal.key | cryptsetup luksOpen /dev/sda2 secret
```

Cette méthode à l'avantage non négligeable de ne pas demander de mot de passe [27].

3.3.2 Création d'un initramfs spécialisé

L'ensemble de ces actions est effectué dans l'initramfs. Il est donc nécessaire de disposer dans celui-ci des outils nécessaires. Pour cet article, j'ai utilisé une slackware. L'initramfs est crée à partir de l'arbre situé sous `/boot/initrd-tree`. Les modules noyaux sont copiés, ainsi que les programmes et librairies permettant d'utiliser la puce TPM : le démon *tcsd* et *tpm_ unsealdata*. Parmi les précautions à prendre pour rendre l'ensemble fonctionnel, il faut faire attention à créer un utilisateur et un groupe *tss*, ainsi que monter l'interface réseau localhost [28].

Le script */boot/initrd-tree/init* est alors adapté, une version réduite en est donnée ici :

```
 (...)
if [ -x /sbin/cryptsetup ]; then
 echo "We are in the cryptsetup magic part"
 mount -t auto $BOOTPART /key
 if [ -f /key/seal.key ]; then
  echo "TPM boot mode activated.."
  ifconfig lo 127.0.0.1
  tcsd
  tpm_unsealdata -z -i /key/seal.key | cryptsetup luksOpen $ROOTPART
      $ROOT
  killall tcsd
 else
  #asking user to unlock
  cryptsetup luksOpen $ROOTPART $ROOT
 fi
 umount /key
 echo "Finishing cryptsetup.."
fi
 (...)
```

3.3.3 Installation en deux temps d'une distribution

Il n'est pas possible de connaître à priori les valeurs que vont prendre les PCR de la puce TPM. L'installation d'une distribution chiffrée doit donc s'effectuer en plusieurs étapes.

La première étape consiste à installer une distribution en choisissant le chiffrement de partition. Le mot de passe utilisé ici n'a aucun enjeu.

---

27. Ce qui ressemble à l'utilisation de Bitlocker, le mécanisme de chiffrement de disque de Microsoft.

28. L'utilisation de localhost n'est pas mentionnée dans les documentations, et le démon tcsd est très peu verbeux concernant l'échec de ses lancements. Si localhost n'est pas monté, alors le lancement de tcsd échoue.

La seconde étape consiste à modifier l'initramfs afin qu'il détecte la présence d'un fichier *seal.key* sur la partition de boot, comme vu précedemment. Si le fichier de clé n'existe pas, alors un mot de passe cryptsetup classique est demandé, s'il existe alors la puce TPM est utilisée.

La troisième étape consiste à rebooter avec cet initramfs. Les registres PCR de la puce sont donc corrects, pour tous les démarrages suivants, ils seront identiques. Il faut alors ajouter une clé LUKS puis la sceller via le TPM.

```
dd if=/dev/urandom of=random_key bs=1024 count=1
sha1sum random_key > rendom
cryptsetup luksAddKey /dev/sda1 random
tpm_sealdata -z -p4 -p8 -p9 -p12 -p14 -i random -o seal.key
shred random_key
shred random
cp seal.key /boot
cryptsetup luksDelKey /dev/sda1 0
```

Le disque ne s'ouvrira désormais que lorsque le boot aura été sain. Tout changement empêchera la machine de démarrer. Cette méthode est très élégante puisque le disque a beau être chiffré, aucun mot de passe n'est demandé à l'utilisateur.

Ceci dit, une attaque de type Cold Boot Attack reste réalisable. Pour s'en prémunir, il est possible d'utiliser un mot de passe TPM pour la clé SRK. Lors de sa création, il faut donc utiliser :

```
tpm_takeownership -y
Enter SRK password:
```

Un mot de passe sera demandé lors de la prise en main de la puce, et lors de toutes ses opérations futures. Une meilleure solution consisterait à utiliser un mécanisme basé sur les clés RSA afin de faciliter le cycle de vie de ce mot de passe (entre autre la révocation) sans devoir réinitialiser à chaque fois la puce TPM. Le développement de cet outil est bien entendu possible, mais son écriture dépasse le cadre de cet article.

Biblio
Cet article est la démonstration de cette méthode, il n'y a pas encore eu de publication sur ce sujet.

3.4 Validité de la solution

Afin de vérifier la solidité de la méthode, il est nécessaire de l'attaquer et de vérifier si elle pallie bien les failles précédemment indiquées qui ont provoqués sa mise en œuvre. Il faut ensuite vérifier si la méthode est industrialisable à grande échelle et comment se comporte la machine au cours de sa vie (sauvegardes).

3.4.1 Rejeu des attaques précédemment nommées

Il existe deux attaques, la *cold boot attack*, et l'*EvilMaid*.

Cold Boot Attack Pour que cette attaque fonctionne, il faut que la machine soit démarrée, disque déchiffré. Cette attaque a été spécifiquement nommée et évitée dans le paragraphe précédent dès lors qu'un mot de passe SRK est demandé à l'utilisateur.

Evil Maid L'attaque en deux temps n'est plus possible. En effet, avec la méthode mesures TPM + déscellement via mot de passe, il devient impossible pour un attaquant de duper un utilisateur. Au mieux, il est possible de récupérer le mot de passe SRK en le loguant et en rebootant la machine dans un démarrage sain. L'utilisateur doit réagir face à ce comportement (reboot douteux), et supprimer ce mot de passe, réinitialiser la puce TPM avec un nouveau mot de passe SRK, et recréer une clé de chiffrement.

En conclusion, la méthode présentée ici répond bien aux attaques soulevées en première partie du document.

3.4.2 Récupération de données en cas de problème

Si vous perdez votre clé de déchiffrement, il n'y a rien à faire, vous êtes perdu. Ce qui veut dire aussi que si la puce TPM brûle, ou que si le portable doit changer de carte mère, les données ne seront plus déchiffrables. Toutefois, grâce à cryptsetup, il est possible d'obtenir une solution de repli grâce aux multiples slots. L'utilisation d'un mot de passe de secours semble être risqué, car il finit toujours à un moment ou à un autre par être divulgué. Par contre, il est possible d'utiliser un fichier comme clé de déchiffrement d'un slot. Ce fichier sera conservé sur une clé USB, fortement sécurisée, comme par exemple dans un coffre fort.

Ainsi, en cas de problème, il sera toujours possible de récupérer le disque et de monter ses partitions pour en extraire les données. Le niveau de sécurité du chiffrement du disque n'est pas diminué.

3.4.3 Industrialisation de la solution

Pour être efficace, et au contraire des *poor man solution*, cette méthode doit être industrialisable. Elle l'est par construction : une fois l'initramfs écrit, il peut resservir sur tous les postes. Deux points particuliers doivent être prévus.

Le premier point concerne l'initramfs. Le jour ou une distribution grand public prévoit d'emblée dans ses initramfs la détection et l'utilisation de la puce TPM, le plus difficile sera fait. Actuellement, il faut encore aller le modifier manuellement. Une fois modifié, il suffit de le mettre en place sur les machines installées.

Le second point concerne un frontend *user-friendly* qui permettrait de détecter la puce TPM, l'initialiser avec mot de passe, et mettre en œuvre le chiffrement via TPM. La liste des commandes à lancer n'est pas très longue, mais peut devenir rébarbative. Il est clairement possible d'automatiser l'ensemble.

Rien n'empêche l'adoption massive du chiffrement de disque sous linux épaulé par la puce TPM lorsque ces deux points seront résolus.

3.4.4 Upgrade de la distribution

L'upgrade de la distribution est un point également sensible. Puisque le noyau et l'initramfs vont changer lors d'un upgrade, il est nécessaire de prévoir également ce changement et d'upgrader en plusieurs boot successifs. Le premier doit ajouter un mot de passe de déchiffrement et supprimer la *seal.key*. Le second reboot permet l'upgrade. Le troisième reboot remet en place le chiffrement via la clé scellée.

4 Conclusion

Comme indiqué dans ce document, la puce TPM est un apport non négligeable en terme de sécurisation du démarrage d'un ordinateur dont le disque est chiffré. Pour les machines sans TPM, il ne reste que des *poor man solutions*. Il faut néanmoins être conscient des limites de la solution. Cette méthode ne protège que l'intégrité du démarrage et rien d'autre.

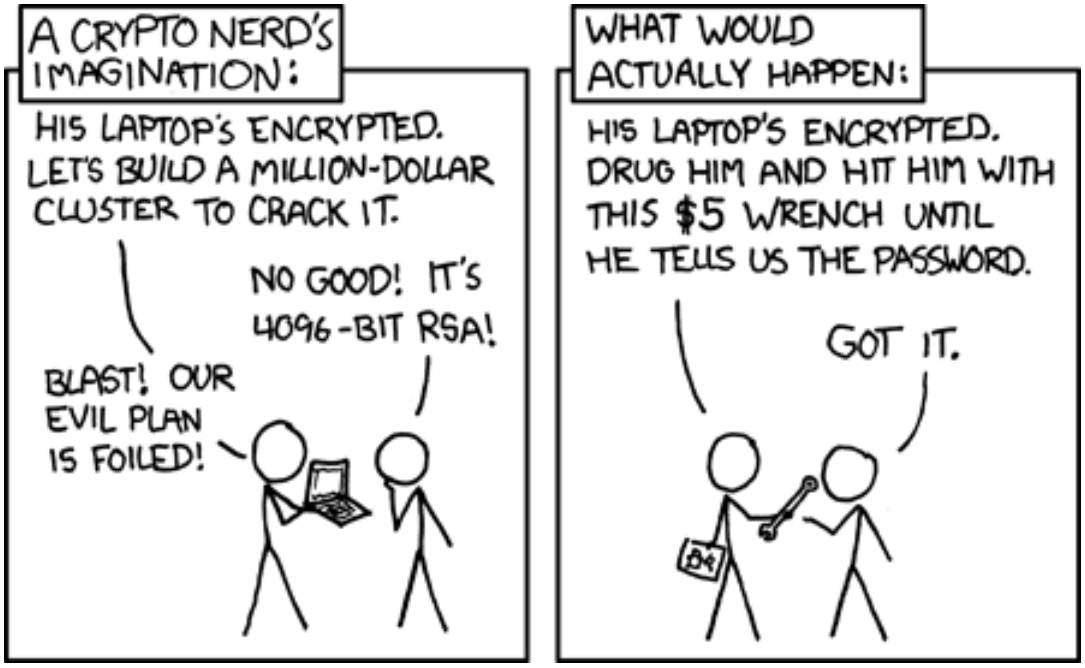
La puce TPM est malgré sa diffusion et son utilisation restreinte attaquée par les chercheurs en sécurité. Plusieurs axes d'attaques sont employés. Tout d'abord des attaques matérielles ou l'attaquant cherche à manipuler la puce pour soit obtenir les clés enregistrées, soit pour sélectionner les valeurs à enregistrer dans les PCR. Ensuite, d'autres attaques ou l'attaquant n'intervient qu'une fois les mesures des PCR effectuées, par exemple en modifiant l'espace mémoire depuis un composant qui a l'autorisation de faire des transferts DMA.

Ceci dit, la puce TPM dans le cadre d'utilisation montré ici est un atout non négligeable pour renforcer la protection offerte par le chiffrement de disque.

Biblio

`http://rdist.root.org/2007/07/16/tpm-hardware-attacks/`
`http://www.sstic.org/2010/presentation/Trusted_Computing_Limitations_actuelles_et_perspectives/a`
`http://actes.sstic.org/SSTIC09/ACPI_et_routine_de_traitement_de_la_SMI/`

Épilogue :



Source : `http://xkcd.com/538/`
License : `http://xkcd.com/license.html`