Attaques applicatives via périphériques USB modifiés : infection virale et fuites d'informations

Benoit Badrignans
benoit.badrignans@seclab.fr

Seclab FR

Résumé Les infrastructures critiques font face à des menaces de malwares et de fuites d'informations, y compris lorsqu'elles sont isolées d'Internet. Les attaques récentes ciblant les systèmes industriels, comme Stuxnet ou Duqu, utilisent l'USB pour contourner cette isolation et atteindre leur cible. Si de bonnes pratiques lors de la configuration des systèmes d'exploitation peuvent éviter certaines attaques véhiculées par des périphériques USB standards, cet article montre que l'usage de périphériques USB dont le logiciel embarqué a été modifié par un attaquant permet de contourner la plupart des protections connues. Les attaques USB peuvent intervenir à plusieurs niveaux (driver de l'OS, pile USB de l'OS, protocole USB comme HID ou Mass Storage, au niveau applicatif). Cet article se concentre sur les attaques au niveau applicatif, ce qui rend l'implémentation relativement simple et leur détection complexe car difficile à distinguer des actions d'un utilisateur légitime. L'article détaille les protections recommandées aujourd'hui, met en évidence leurs limites vis à vis des dernières attaques publiées, propose deux nouvelles attaques permettant d'écrire sur des clefs ou des disques dur USB même si ceux-ci sont montés en lecture seule par le système d'exploitation, et montre comment contourner certaines protections basées sur des antivirus.

1 Introduction

Le contexte de cet article est celui des infrastructures critiques qui possèdent généralement un réseau isolé totalement déconnecté d'Internet, ou plus généralement des sites où sont stockées des données à forte valeur dont la fuite aurait de graves conséquences. Ces sites disposent d'un contrôle d'accès strict, et les ordinateurs y résidant disposent d'antivirus et de logiciel permettant de protéger l'information qu'ils contiennent. Le réseau étant protégé ou isolé d'Internet, les ports USB de ces machines sont généralement les seules interfaces directement ou indirectement accessibles à un attaquant. Par exemple les attaques récentes comme Stuxnet [9] ou Duqu [23] utilisent l'USB pour contourner ce type d'isolation et atteindre leur cible. De plus, si les clefs USB et les périphériques USB en général sont des sources d'infections potentielles, ils peuvent également aboutir à la fuite d'informations sensibles.

Cet article débute par un état de l'art des protections et des attaques existantes, qu'elles concernent l'infection ou la fuite d'information via USB. Il présente ensuite des attaques pouvant être menées en utilisant une clef USB dont le logiciel embarqué a été modifié par un attaquant. La première permet la fuite d'information via ce type de clef USB, même si celle-ci *montée* en lecture seule par le système d'exploitation hôte, elle fonctionne quel que soit le système d'exploitation sans nécessiter les droits administrateurs. L'attaque est d'abord décrite du point de vue théorique, puis un exemple d'implémentation est décrit. Une autre attaque est présenter pour leurrer les logiciels antivirus en changeant dynamiquement les fichiers contenus sur la clef USB. Enfin des contre-mesures sont proposées pour limiter la portée de ces attaques.

2 État de l'art

2.1 Protections

Les problèmes de sécurité liés à l'USB sont généralement pris en compte dans les infrastructures critiques depuis Stuxnet ou Duqu, néanmoins les ports USB sont quasiment indispensables par exemple pour permettre d'utiliser des claviers et des souris USB, les ports PS/2 tendant à disparaître. La protection la plus radicale consistant à supprimer physiquement tous les ports accessibles sur chacune des machines du système d'information (SI) n'est donc généralement pas possible. Une politique de sécurité pourra interdire l'usage de périphériques USB autres que des claviers ou des souris. Cette interdiction pourra être appliquée en empêchant l'introduction de matériel électronique USB non validé en effectuant des fouilles régulières, cependant ceci devient difficile avec la miniaturisation des clefs USB et la prolifération des téléphones portables pouvant être vu comme des périphériques USB de toutes sortes. L'interdiction est donc généralement implémentée au niveau de chaque machine du SI en autorisant uniquement l'usage de périphériques USB HID. Ceci peut être réalisé sous Windows en modifiant la base de registre, ou sous Linux en définissant des règles *udev* appropriées.

Il peut également être commode d'autoriser l'usage de clef USB ou de disque durs, par exemple pour introduire des données sur des machines non connectées à Internet comme des mises à jour antivirus ou des données nécessaires au fonctionnement du SI.

La suite de ce chapitre décrit les bonnes pratiques recommandées actuellement par différentes agences publiques ([11], [13], [14]) pour se protéger au mieux lorsque l'on autorise les périphériques USB HID et/ou

de stockage de masse. Les attaques considèrent un utilisateur mal intentionné mais également un utilisateur véhiculant l'attaque à son insu.

Protection face à la fuite d'information Les périphériques HID (*Human Interface Device*) que sont les claviers et les souris standards ne sont pas prévus pour permettre la fuite d'information. Ces périphériques permettent plutôt à l'utilisateur d'insérer de l'information dans le SI. Ils ne semble donc pas nécessaire de mettre en place des mesures de protection pour éviter la fuite d'information via ces périphériques.

Dans le cas des clefs USB ou des disques durs portables, la menace est plus évidente. Pour éviter la fuite d'information via ces médias, il est conseillé de les interdire ou de mettre en place une politique interdisant l'écriture sur ces périphériques. Ceci peut être réalisé sous Windows en modifiant la valeur de la clef de registre suivante (passage à la valeur "1") :

```
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\ \
StorageDevicePolicies\WriteProtect
```

Sous Linux une règle *udev* de la forme suivante peut être définie :

```
KERNEL=="sd[b-z][0-9]"
ACTION=="add", RUN+="/bin/mkdir -p /mnt/usb-%k", \
RUN+="/bin/mount -t auto -o ro /dev/%k /mnt/usbhd-%k"
```

Des périphériques appartenant à d'autres classes USB peuvent également aboutir à de la fuite d'information non maîtrisée, le cas le plus évident est celui des imprimantes USB. Il est donc recommandé de limiter les classes USB aux seules classes indispensables au bon fonctionnement du SI, par exemple en définissant un liste blanche des classes autorisées dans *udev* sous Linux [8] ou dans la base de registre sous Windows [1].

Il est également préférable de restreindre, non seulement les classes USB autorisées, mais également le descripteur USB de chaque périphérique autorisé au sein du SI. Le descripteur USB est une structure définissant le périphérique USB, les informations comprennent : la classe USB, l'identifiant unique du vendeur et du produit (*vendor ID/product ID*), la chaîne de caractère décrivant le dispositif, ainsi que de nombreuses autres informations comme le numéro de série. Ce descripteur est fourni par chaque périphérique USB sur requête du système hôte. Cette restriction peut une fois de plus être réalisé en modifiant la base de registre de Windows ou en définissant des règles *udev* sous Linux.

Protections face à l'infection virale Les périphériques USB peuvent également être une source d'infection virale, comme dans le cas de *Stuxnet*. La charge virale peut être activée lorsque l'utilisateur ouvre un fichier présent sur une clef USB infectée, ou si aucune précaution n'a été prise simplement lors du branchement via la fonctionnalité de démarrage automatique présent sous Windows [25] et sur certaines distributions Linux [21].

Généralement seuls les périphériques pouvant transporter des fichiers, comme les clefs USB ou les disques durs, sont considérés comme des sources d'infection virale potentielles. Néanmoins, comme le chapitre concernant les attaques le démontre, d'autres classes de périphériques peuvent représenter des menaces, par exemple la classe HID.

La première précaution consiste à interdire complètement l'usage de ce type de dispositif, toujours via *udev* ou la base de registre Windows. Lorsque ceci n'est pas possible il est recommandé de désactiver les fonctionnalités de démarrage automatique [25] qui sont toujours présentes dans Windows 8. L'activation du virus nécessite ainsi une action de l'utilisateur.

Cette précaution basique n'est pas suffisante, un document ou un programme infecté peut être ouvert par l'utilisateur par malveillance ou inadvertance. Il est donc indispensable de disposer d'un antivirus efficace et à jour. Dans certains cas le contrôle antivirus est réalisé sur une machine dédiée qui a pour but de nettoyer les dispositifs USB avant leur usage sur une machine sensible.

Il est possible d'augmenter le niveau de sécurité en restreignant les périphériques USB acceptés aux périphériques présentant un descripteur USB connu. Ainsi un utilisateur ne pourrait pas utiliser une clef USB personnelle, potentiellement vérolée. Dans ce cas, la politique de sécurité doit mettre en place une fouille systématique pour prévenir les utilisateurs de quitter le SI avec les clefs USB autorisées pour éviter qu'elles soient infectées en dehors du SI puis réintroduites par un utilisateur, mesure qui en pratique est difficilement réalisable.

Généralisation Il existe de nombreuses classes de périphériques USB comme par exemple les classes : *Audio, Image, SmartCard, Video, Printer, Wireless controller ou Ethernet controller*. La plupart d'entre elles peuvent aboutir à des fuites d'information, ceci va de soit pour les périphériques du type imprimante ou contrôleur Ethernet que le responsable de la sécurité consciencieux prendra soin d'interdire sur les machines sensibles.

Dans le cas de la menace liée à l'infection virale, l'usage d'antivirus et la limitation à des périphériques dont le descripteur USB est connu sont généralement les deux seules mesures de protection possibles.

Produits logiciels existants de protection des ports USB De nombreux logiciels commerciaux [27] [12] [24] [17] offrent des fonctionnalités de protection face à la menace que représentent les ports USB.

Ces logiciels permettent généralement de limiter l'accès à des périphériques USB dont le descripteur USB est connu. Ils peuvent également forcer le système à *monter* les dispositifs de stockage de masse en lecture seule. Ils mettent donc en œuvre les précautions précédemment décrites, l'usage de ce type de produit simplifie la tâche de la personne en charge de la sécurité du système considéré, et évite par exemple les erreurs de manipulation de la base de registres de Windows pour implémenter ces règles.

De plus, la plupart des antivirus [31] effectuent des contrôles de sécurité sur les périphériques USB de stockage, ils vérifient notamment que le système de fichier ne contient pas d'erreurs ou de code malveillant et appliquent un contrôle antivirus des fichiers présents sur le dispositif.

Conclusion Dans la mesure du possible il est préférable de condamner physiquement les ports non utilisés d'une machine que l'on souhaite protéger.

Quand certains périphériques USB sont absolument nécessaires les logiciels existants sur le marché peuvent contrecarrer les attaques les plus basiques et éviter la fuite d'information face à un utilisateur peu scrupuleux. Néanmoins ils sont dans l'incapacité de protéger le système face à des attaques avancées utilisant des périphériques USB dont le logiciel embarqué a été modifié par l'attaquant comme le montre le chapitre suivant.

2.2 Attaques

La surface d'attaque exposée par les simples ports USB d'une machine est relativement vaste comme le montre la figure 1. De nombreuses attaques ont été publiées sur quasiment tous les niveaux d'abstraction représentés.

Les règles décrites dans l'état de l'art des protections et les contrôles antivirus sont efficaces face à la plupart des menaces véhiculées par des périphériques USB standards comme les clefs USB que l'on peut

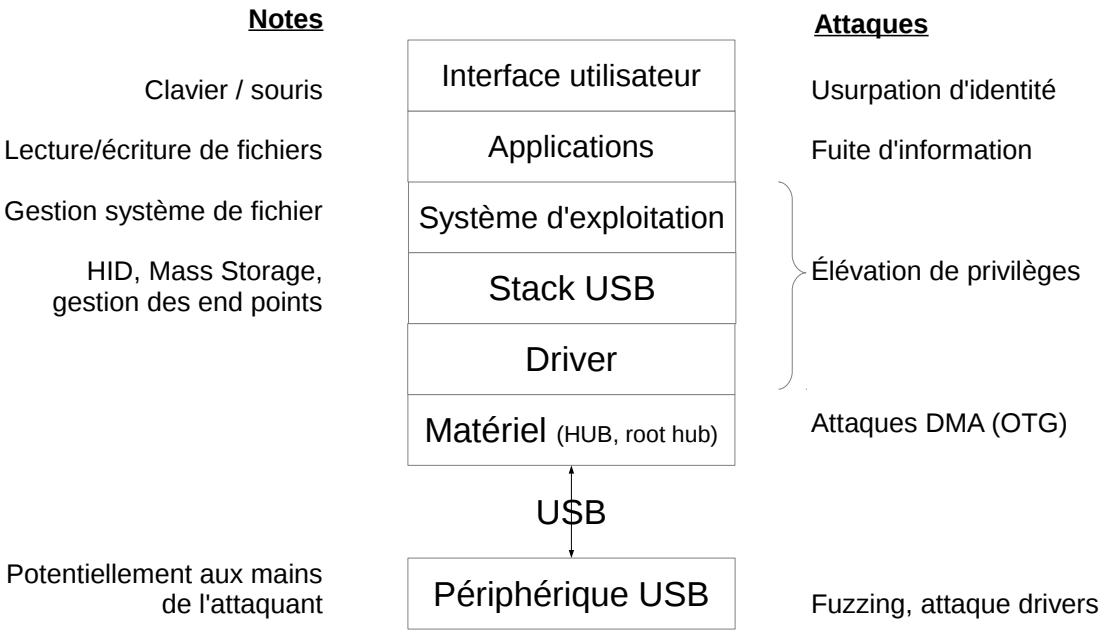

Figure 1. Couches d'abstraction entre l'utilisateur et un périphérique USB

trouver dans le commerce. Néanmoins ce chapitre montre également qu'elles sont généralement inefficace face à des périphériques USB dont le logiciel embarqué a été modifié afin d'implémenter différents types d'attaques. Ces périphériques sont nommés par la suite périphériques USB malicieux.

Attaques avec dispositifs USB standards

Autorun/autoplay/prévisualisation (thumbnails) La fonctionnalité d'autorun (démarrage automatique d'un programme en interprétant un fichier autorun.inf à la racine d'une clef USB) existe depuis 2004 dans Windows XP SP2. Cette fonctionnalité permet donc de lancer et de propager des virus sans intervention de l'utilisateur, cette fonctionnalité a donc généralement été désactivée par les responsables de la sécurité consciencieux. Elle a été retirée de Windows 7 à partir de 2009 mais fait maintenant parti de la fonctionnalité dite *autoplay* qui permet par exemple de lancer un fichier mp3 vérolé automatiquement. Il est donc toujours nécessaire de désactiver ces fonctionnalités des systèmes Windows. On peut également noter que ceci n'est pas spécifique à Windows, des attaques via autorun ou autoplay ont été publié pour Mac OSX [6] et Linux [22]. Pour cette dernière référence l'attaque exploite des vulnérabilités présentent dans

le système de prévisualisation du navigateur de fichier Nautilus et que l'attaque fonctionne même si l'ordinateur est verrouillé.

Le célèbre Stuxnet s'est propagé via des clef USB au sein de systèmes critiques iraniens. Il exploite principalement la faille LNK [9] qui permet de charger une DLL arbitrairement simplement en navigant dans un dossier à l'aide de l'explorateur Windows. Cette exploit est lancé automatiquement depuis une clef USB même si la fonctionnalité de démarrage automatique est désactivée.

Système de fichier Les attaques peuvent également se baser sur des systèmes de fichiers spécialement conçus pour exploiter des failles dans le système d'exploitation hôte.

[32] reporte par exemple une vulnérabilité dans la gestion des tables de partition MAC et LDM (*Logical Disk Manager*) sous Linux. Elle peut mener à une élévation de privilège, à des dénis de service ou à des fuites d'information. Il est important de noter ici que qu'elle que soit la configuration du système Linux celui-ci va automatiquement lire la table de partition afin de pouvoir présenter les fichiers correspondants à l'utilisateur (ex : /dev/sdb, /dev/sdb1). Il est donc particulièrement difficile de s'en protéger sans supprimer du noyau le support de ce type de partition.

Drivers Dernièrement (12 mars 2013), Microsoft [26] annonçait la correction d'une faille dans le driver chargé de la gestion des périphériques USB et plus précisément de leur descripteur USB. Cette vulnérabilité permettait une élévation de privilèges jusqu'au mode noyau. Cette attaque ne nécessite aucune action de l'utilisateur puisque déclenchée lors de l'énumération USB du dispositif et fonctionne même si la session est verrouillée.

D'autres attaques sont possibles en exploitant la manière dont Microsoft gère l'installation automatique de driver dans ses systèmes (Windows Update), en effet Windows peut rechercher automatiquement un driver adéquat pour un nouveau périphérique, driver qui peut se trouver sur le périphérique. Windows n'accepte par défaut que des drivers signés par Microsoft, cependant la soumission du code source n'est pas nécessaire et seul le binaire est signé par Microsoft, un attaquant peut donc faire valider un driver comportant une vulnérabilité.

Conclusion Malgré les contre-mesures présentées dans le premier chapitre, des vulnérabilités peuvent encore exister même lorsque l'on considère uniquement des clefs USB standards. La surface d'attaque et donc

les potentielles vulnérabilités augmentent sensiblement quand on considère des attaques basées sur des dispositifs malicieux, comme le montre le paragraphe suivant.

Attaques avec dispositifs USB malicieux Le développement de dispositifs USB s'étant radicalement simplifié ces dernières années [33], de nombreux exploits utilisant des dispositifs malicieux ont vu le jour. Ce type de dispositif peut être implémenté sur un microcontrolleur bas-coût disposant d'un port USB, ou grâce à un téléphone portable disposant d'un port USB OTG (On-The-Go) et en utilisant par exemple le module Linux nommé USB-Gadget sur le téléphone.

Ces périphériques USB malicieux peuvent disposer d'une *intelligence* leur permettant de détecter les caractéristiques du système hôte, d'usurper un descripteur USB autorisé, de ne pas réaliser des opérations demandées par l'hôte (comme par exemple d'effacer un fichier vérolé), d'interpréter différemment certaines commandes USB ou même de se faire passer pour l'utilisateur. Ces périphériques peuvent prendre la forme d'une clef USB mais se présenter au système comme un clavier et/ou une souris. Par exemple le dispositif PHUKD [2] permet d'émuler des frappes clavier à l'insu de l'utilisateur, pour par exemple lancer des programmes malveillants. Les paragraphes suivants décrivent des attaques de ce type.

Attaques DMA et écoute sur le bus Il est généralement reconnu que les attaques DMA sur le bus USB ne sont pas possible étant donné que le hub USB central (root hub) d'une machine est initiateur de toutes les transactions. Un périphérique ne peut donc pas de lui même lire ou écrire une zone mémoire arbitraire. Ce type d'attaque est plus simple à implémenter sur un bus comme le PCIExpress où le DMA fait partie intégrante de la norme.

Cependant, depuis l'apparition des périphériques OTG (On-The-Go) permettant d'être vu comme un périphérique ou comme un hôte USB, des preuves de concept [16] laissent penser que ce type d'attaque est réalisable également via USB.

De plus tous les périphériques connectés sur un contrôleur USB hôte reçoivent les mêmes informations même si elles ne leur sont pas destinées. Un périphérique USB standard ne va pas prendre en compte ces informations, mais un périphérique malicieux peut facilement les intercepter. Un cas d'attaque typique est une clef malicieuse branchée sur le

même contrôleur USB qu'un lecteur de carte à puce qui va enregistrer le code PIN à l'insu de l'utilisateur.

Attaque sur drivers USB Les drivers USB des systèmes d'exploitation attendent généralement que le dispositif se comporte d'une manière *normale*, notamment en renvoyant toujours les mêmes informations pour une même requête. De plus les couches d'abstraction au sein des systèmes sont nombreuses et donc difficiles à valider entièrement. Par exemple la gestion d'une clef USB standard nécessite un driver USB bas niveau (descripteurs USB), un driver pour le standard *USB Mass Storage* ainsi qu'un driver SCSI.

De nombreuses attaques utilisant des périphériques USB malicieux ont été reportées. Par exemple celle ayant permis le *jailbreak* de la PlayStation 3 en exploitant une corruption mémoire dans le code lisant les descripteurs USB, ceci permettant d'exécuter du code arbitraire.

Acquisition d'information sur le système hôte Chaque système d'exploitation se comporte différemment lorsqu'il accède à un périphérique USB, chacun lira le descripteur USB de manière différente. Dans le cas des périphériques de stockage de masse, le parcours du système de fichier de chaque partition se fera différemment (séquences de lecture différentes), ou même dans le cas de Mac OSX le système tentera de créer des répertoires cachés pour son usage propre.

Une clef USB malicieuse peut, de cette manière, découvrir quel est le système d'exploitation hôte sans intervention de l'utilisateur. Elle peut ainsi adapter son attaque aux vulnérabilités connues dans chacun des systèmes qu'elle souhaite attaquer.

Usurpation d'identité USB La plupart des logiciels de protection des ports USB se basent sur le descripteur USB fourni par le périphérique USB pour appliquer un filtrage. Généralement seuls quelques couples *Product Id/Vendor Id (VID/PID)* sont autorisés par le responsable de la sécurité, ils appartiennent typiquement à des périphériques USB que ce dernier a sélectionné comme étant autorisés au sein du SI. D'autres produits vont jusqu'à comparer scrupuleusement la totalité du descripteur USB (numéro de série, chaînes de caractères...) avec les descripteurs autorisés plutôt que de se contenter du couple *VID/PID*.

Néanmoins, ces informations peuvent être retrouvées par un attaquant relativement simplement, par *ingénierie sociale* ou en empruntant un périphérique USB autorisé pour recopier l'intégralité de son descripteur USB qui est facilement accessible (commande *lsusb -v* sous Linux).

Une fois ces informations recueillies, il est trivial de réaliser un dispositif USB malicieux présentant le même descripteur USB qu'un périphérique légitime qui sera ainsi accepté sur le système. L'attaquant peut alors implémenter une attaques adaptée à sa cible. [28] est un exemple d'implémentation de cette usurpation, elle peut même aller jusqu'à l'attaque par force brute du couple *VID/PID* si celui-ci est inconnu de l'attaquant.

Attaques via USB Mass Storage La maîtrise du logiciel embarqué par l'attaquant lui permet également de modifier à la volée le contenu d'une partition ou de fichiers. Cette méthode a été utilisée avec succès par [15]. Le principe est de présenter au système cible (en l'occurrence une télévision connectée basée sur Linux) une mise à jour de binaire signée numériquement, puis de forcer le système à relire le fichier après qu'il ai vérifié sa signature, cette deuxième lecture ne renverra pas les mêmes informations que la première permettant ainsi d'installer du code non autorisé. Une variante de cette attaque, nommée *Read It Twice*, est présentée dans le dernier chapitre de cet article.

Attaques via HID La possibilité d'utiliser des périphériques USB HID comme les claviers et les souris est souvent indispensable. On pourrait penser naïvement que ceux-ci ne présentent pas de problématique de sécurité, cependant différents travaux ont montré que ça n'est pas le cas.

Deux types d'attaques ont été mise en œuvre sur ces périphériques :

- L'usurpation de l'identité de l'utilisateur
- La fuite d'information

Si les claviers et les souris sont les interfaces classiques entre un utilisateur et le système, dans le cas d'un périphérique HID malicieux ils peuvent être une interface entre l'attaquant et le système. En effet il est possible de développer un dispositif [2] se présentant comme un clavier et/ou une souris classique mais qui soit capable de générer une suite de frappe clavier, de mouvement de souris et de clics sans que l'utilisateur n'effectue aucune action. Cette attaque est particulièrement dangereuse car l'attaquant ayant *piégé* le dispositif HID dispose automatiquement des mêmes droits que l'utilisateur cible. Cet utilisateur pouvant être l'administrateur.

Les démonstrations réalisées dans les travaux [2] montrent qu'il est possible de lancer automatiquement un programme présent sur un périphérique USB, valider la demande de Windows lorsque l'on exécute un programme d'un éditeur non connu ou demandant les droits administrateur. Le dispositif malicieux peut également ouvrir un fichier texte (par exemple avec la séquence de frappe suivante sous Windows : *Windows* +

R , notepad.exe), écrire une charge virale codée en base 64 dans ce fichier texte, le sauvegarder sur la machine ciblée (*Ctrl+s*), ouvrir une invite de commande (*Windows + R , cmd*), convertir cette charge virale en binaire via une commande (si disponible sur le système) puis exécuter le virus ainsi généré. L'utilisateur, qui peut utiliser à son insu un tel dispositif, peut ne pas voir la succession de fenêtre s'ouvrir tant la vitesse de frappe émulée par le clavier malicieux est rapide.

En ce qui concerne la fuite d'information, il faut noter qu'un périphérique HID est généralement utilisé pour transmettre de l'information depuis l'utilisateur vers le système hôte. Cependant dans le cas des claviers, le système hôte peut également envoyer de l'information au périphérique USB. Il peut en effet agir sur 5 bits [5] représentant respectivement : *NUM LOCK, CAPS LOCK, SCROLL LOCK, COMPOSE* et *KANA*. En effet c'est le système hôte qui est responsable par exemple de gérer la diode du clavier relative à la touche *caps lock* car plusieurs claviers peuvent être connectés à un même ordinateur.

Les travaux [10] montrent qu'il est possible de développer un dispositif HID capable d'interpréter certains de ces bits pour permettre d'écrire des informations sur une mémoire Flash présente sur le dispositif USB malicieux. L'attaquant doit ensuite utiliser un programme capable de convertir un fichier pour le transmettre au périphérique malicieux en exploitant ces bits. Ce programme ne nécessite pas les droits administrateur. Cette méthode est peu performante (1,25 octets par seconde) car le mode de transfert de l'information utilise à contre emploi ces bits de données et la norme HID n'est pas prévue pour offrir un débit important. On peut également noter que cet exploit est également applicable sur le port PS/2.

Conclusion Il est donc possible avec un minimum de connaissance du protocole USB et du développement sur micro-controller (ou en utilisant le module noyau *USB Gadget* [3] sous Linux et un dispositif offrant une interface *USB device*) de créer des périphériques USB malicieux capables d'introduire du contenu malveillant ou de permettre la fuite d'information tout en contournant la majeure partie des protections existantes. Des plateformes simples d'emploi et dédiées aux attaques USB voient le jour et vont certainement permettre de mettre à jour de nouvelles vulnérabilités dans les différentes couches d'abstractions nécessaires au fonctionnement de l'USB, notamment la plateforme Facedancer USB [34] programmable facilement en python pour faciliter la recherche de vulnérabilités.

La norme USB permet également de combiner au sein d'un seul dispositif les classes *USB Mass Storage* et *USB HID*, ceci donnant encore plus de latitude à l'attaquant pour outrepasser les protections mises en place.

Les chapitres suivants décrivent deux attaques proposées par cet article. Elles se focalisent sur les attaques HID et sur l'usage à contre-emploi du système de fichier des clefs. En effet ceci est difficilement discernable de l'usage normal que pourrait avoir un utilisateur. Les attaques ne portent donc pas sur la pile USB, les drivers, ou le système de fichier. Les autres classes USB ne sont pas utilisées, considérant qu'elles sont interdites par configuration de l'OS ou par l'usage d'un logiciel de protection USB.

3 Attaque 1 : Fuite d'information sur périphérique en lecture seule

Afin de sensibiliser les responsables de la sécurité à la dangerosité des interfaces d'un poste, Seclab FR a développé une preuve de concept de dispositif *USB Mass Storage* capable de récupérer de l'information même si celui-ci est *monté* en lecture seule par le système d'exploitation. L'attaque part du principe que le système accepte de *monter* le périphérique, soit car aucun filtrage du descripteur USB n'est en place, soit car l'attaquant a pris connaissance d'un descripteur USB autorisé.

L'idée principale de l'attaque est de remarquer que même si une clef USB est *montée* en lecture seule, des commandes de lecture sont envoyées par le poste hôte jusqu'au périphérique USB, ce dernier peut alors les interpréter à sa guise, par exemple comme des écritures sur sa mémoire de masse.

Les attaques de ce type ne sont pas nouvelles et sont souvent référencées dans la littérature sous le nom de *Covert Channel Data Leakage* [7]. Les implémentations sur des protocoles réseaux sont les plus répandues, aucune implémentation sur le bus USB ne semble avoir été rendue publique.

3.1 Principe

Principe général Cette première attaque ne cherche pas à infecter le système hôte mais uniquement à contourner les systèmes de protections face à la fuite d'information. L'attaque part du principe qu'une clef USB malicieuse est *montée* en lecture seule sur le système hôte (quel que soit le

système d'exploitation). Il est alors possible de lire les fichiers présents sur la clef USB via des commandes shell (*cat*, *dd*, *vim*) ou via un programme en C sans que les droits administrateurs ne soient nécessaires.

Le périphérique USB en question dispose à minima d'un micro-contrôleur disposant d'un port *USB device* et d'une mémoire de masse, cette dernière servant d'une part à stocker le système de fichier qui sera présenté au système hôte, et d'autres part à stocker les informations récupérées sur le système hôte dans un espace non visible depuis l'hôte.

Afin d'effectuer une fuite d'information depuis le système hôte vers la clef USB, il est nécessaire que le périphérique USB interprète les commandes de lectures d'une manière différente d'une clef USB classique. Une commande de lecture USB, provenant du système hôte, transmet deux informations : L'adresse de lecture et la taille de la lecture. La taille et l'adresse doivent être un multiple de la taille des secteurs de la clef USB, soit 512 octets pour la grande majorité des clefs USB. En retour le périphérique renverra les données demandées par le système hôte.

L'attaque proposée utilise uniquement l'adresse comme vecteur d'attaque. Afin de ne pas nécessiter les droits administrateurs, les lectures ne sont pas effectuées directement sur le disque présenté par le périphérique USB, mais dans des fichiers présents au sein du système de fichier. En effet Windows ne permet pas à un utilisateur de lire ou d'écrire directement sur un disque sans les droits administrateurs. Il serait possible de réaliser cette attaque à un niveau d'abstraction plus bas : en employant des transferts *USB Builk* sur un *end point* supplémentaire ou avec des commandes SCSI inutilisées par les clefs USB standards. Cette méthode ne nécessite pas les droits administrateurs, cependant ces commandes sont facilement détectables par des logiciels antivirus ou de protection USB et peuvent donc être filtrées ou déclencher des alarmes. Cette attaque propose donc d'opérer au plus haut niveau d'abstraction offert par une clef USB, à savoir la lecture de fichiers, et ainsi rendre sa détection complexe.

Pour mettre en œuvre l'attaque le système de fichier doit contenir au moins deux fichiers, nommés par la suite *commands.txt* et *output.txt*, d'autres fichiers peuvent être présents mais ne sont pas utilisés pour l'attaque. Le premier fichier, *command.txt* permet de lancer l'attaque, ceci se fait en lisant consécutivement le deuxième secteur du fichier puis le premier, l'attaque est arrêté en réalisant la même opération. On pourrait imaginer lancer l'attaque en effectuant uniquement la lecture du premier secteur du fichier, mais la plupart des systèmes d'exploitation lanceraient l'attaque dès le branchement de la clef car ceux-ci réalisent, dès

le montage du périphérique, la lecture des fichiers de petite taille afin de les placer en cache et améliorer l'expérience utilisateur. Il est donc nécessaire d'utiliser une séquence de lecture qui soit reconnaissable par rapport aux opérations automatiques réalisées par le système.

Une fois l'attaque lancée, le périphérique USB entre dans un mode particulier dans lequel toute lecture dans le fichier *output.txt* abouti à une écrire de 16 bits sur la mémoire de masse. La donnée écrite est en fait l'adresse dans le fichier demandée par le système hôte, par exemple si le système hôte lit le secteur 0x1234 du fichier, la donnée 0x1234 sera écrite sur la mémoire de masse.

Pour permettre d'écrire une valeur comprise entre 0 et 0xFFFF le fichier doit avoir une taille suffisante. Comme les adresses USB représentent des adresses de secteurs et non des adresses logiques, le fichier doit avoir une taille minimale de $512 * 2^{16}$ octets, soit environ 34 Mo. Après chaque lecture réalisée par l'hôte, la donnée correspondante est ajoutée sur la mémoire de masse et l'index d'écriture est incrémenté par le logiciel embarqué sur le périphérique.

Le programme de transfert Afin de transférer des données vers le périphérique il est nécessaire d'utiliser un programme ou un simple script pour réaliser les opérations précédemment décrites. Un tel programme pourrait s'écrire avec le pseudo-code suivant :

```
writeToUsb ( inputFile )
    i=0
    read512Bytes ( command.txt, 512 )
    read512Bytes ( command.txt, 0 )
    while ( inputFile )
        data2Bytes = read2Bytes ( inputFile, i )
        read512Bytes ( output.txt, data2Bytes * 512)
        i++
    read512Bytes ( command.txt, 512 )
    read512Bytes ( command.txt, 0 )
```

3.2 Mise en pratique

La mise en pratique de cette attaque théorique a été réalisée sur une carte de développement disponible dans le commerce, elle dispose du composant SEC2410 de SMSC. Il inclut un processeur ARM Cortex M3 cadencé à 60 MHz dont les performances sont suffisantes pour gérer les

transferts USB considérés. Elle dispose de nombreuses entrées/sorties mais seules les interfaces vers la carte SD et l'interface *USB 2.0 High-Speed* sont utilisées. L'usage d'un bus *High-Speed* permet d'obtenir de bonnes performances. La mémoire RAM nécessaire pour exécuter le code embarqué malveillant est intégrée directement dans le processeur ARM de la carte, il est donc relativement aisé de miniaturiser cette carte pour en faire une clef USB de proportion standard.

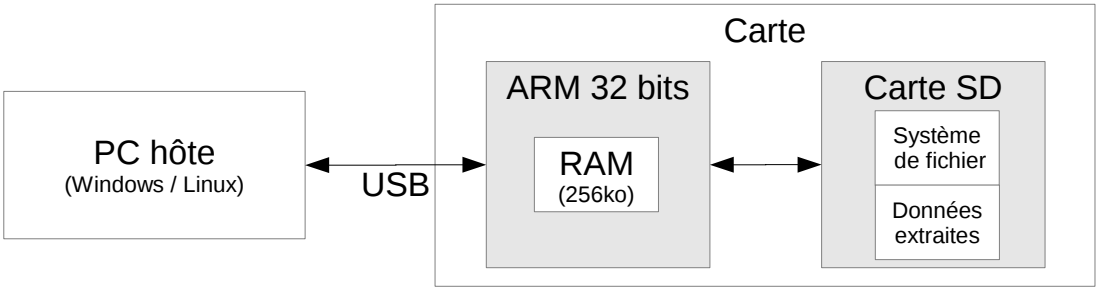

Figure 2. Architecture globale du dispositif

L'implémentation des classes *USB Mass Storage* et *HID* est fournie par le fabricant sous accord de confidentialité. L'implémentation d'un dispositif composite est donc relativement triviale (dispositif se présentant à la fois comme une clef USB, un clavier et une souris).

La carte SD est séparée en deux zones, la première stocke le système de fichier présenté au système hôte lorsqu'une lecture standard est effectuée se sont ces données qui sont renvoyées. Les données récupérées et envoyées au dispositifs via les lectures détournées sont stockées dans la deuxième partie de la carte SD, ces données ne sont pas accessibles directement depuis le système hôte, elles peuvent être lues après l'attaque en lisant directement la carte SD sans passer par le dispositif USB.

Le descripteur USB est défini via plusieurs constantes aisément modifiables dans le code source si l'usurpation d'un descripteur USB autorisé est nécessaire.

Description de l'attaque Si l'utilisateur est confondu avec l'attaquant l'attaque est simple à mettre en œuvre. Par exemple l'attaquant pourra utiliser un programme présent dans le système de fichier de la clef USB, il lui suffira alors de réaliser un *glisser/déposer* des fichiers cibles sur le programme. Il peut également écrire un simple script sous Linux utilisant principalement la commande *dd* ainsi que des utilitaires basiques pour

déterminer la taille du fichier d'entrée ou convertir du format binaire vers un format hexadécimal.

Si l'attaque se déroule à l'insu de l'utilisateur (attaquant et utilisateur étant deux personnes différentes) se déroulerait de la manière suivante :

- L'utilisateur insère la clef USB
- Le système *monte* la clef en lecture seule, conformément à la politique de sécurité
- La clef USB détecte le système hôte
- Elle tente de lancer le programme présent sur la clef USB (combinaison de frapes clavier adéquates)
  - Si elle y parvient l'attaque se poursuit
  - Si elle n'y parvient pas il se peut que le système hôte soit configuré pour ne pas exécuter des programmes présents sur des disques amovibles. La clef tente alors de copier le fichier sur le disque de la machine cible. Elle peut également écrire le programme encodé en base 64, puis le convertir en binaire comme décrit précédemment.
- Le programme scanne le disque de la machine cible et copie les fichiers sur la carte SD du dispositif malicieux en ne réalisant que des lectures.

Résultats Le débit maximal théorique du bus *USB High-Speed* est de 60 Mo/s, mais en réalité limitée à environ 35Mo/s. La vitesse maximale de transfert mesurée entre le système hôte et la carte de développement est de proche de cette limite de 35 Mo/s (sans lecture ou écriture sur la carte SD). Le débit entre le processeur de la carte de développement et la carte SD est limité par les performances de la carte elle-même (fréquence d'horloge maximale, nombre de broche dédiées aux données, performances de la technologie Flash employée). Dans un usage normal (lectures/écritures classiques) et avec une carte SD de bonne qualité, la carte de développement permet d'atteindre un débit de 10Mo/s en lecture et d'environ 8Mo/s en écriture.

Les performances obtenues durant l'attaque : lorsque l'on écrit via des lectures, sont d'environ 100 ko/s. Étant donné que les commandes de lectures sont utilisées à contre-emploi, elles ne véhiculent que 16 bits de données utiles ce qui réduit fortement la bande passante réelle. Cependant, les commandes de lecture initiées par l'hôte n'aboutissent pas à une réelle lecture sur la carte SD mais au stockage des 16 bits utiles en mémoire RAM. Les données sont écrites sur la carte SD uniquement lorsque le tampon en RAM atteint 65536 octets afin d'optimiser les per-

formances. L'écriture réelle est donc réalisée une fois toutes les 32768 commandes de lectures. Les commandes de lectures peuvent donc avoir lieu à une vitesse quasiment idéales (35 Mo/s), de plus que le processeur de la carte de développement est capable de réaliser l'écriture des 65536 octets en *DMA* (Direct Memory Acccess) pendant que l'hôte continue à effectuer des lectures. Il en résulte que l'écriture sur la carte SD est réalisé en temps masqué, le débit est alors limité à :

Débit idéal * Données utiles a chaque lecture / taille des lectures

Dans le cas présent le débit est donc de l'ordre de : (35*2)/512 = 0,136 Mo/s ce qui est proche des performances relevées en pratique. Il est possible d'améliorer ces performances en jouant sur la taille des données utiles à chaque lecture, par exemple en utilisant un fichier ayant une taille virtuelle plus élevée, ou en utilisant différents fichiers. Le débit idéal est limité par le processeur utilisé sur la carte de développement et par les performances intrinsèques du bus *USB High-Speed*, la taille des lectures ne pouvant être inférieure à 512 octets. L'attaquant peut donc améliorer les débits en jouant sur le premier paramètre, les performances maximales sont atteintes lorsqu'elles sont proches des performances maximales de la carte SD en écriture.

Cette attaque est réalisable sur la plupart des systèmes d'exploitation sans les droits administrateurs, il suffit que l'utilisateur ai la possibilité de lire des fichiers sur un périphérique USB *monté* en lecture seule. Les lectures ne doivent cependant pas être réalisées dans le cache du système d'exploitation mais doivent donner lieu à de véritables lectures sur le dispositif USB. Ceci peut être fait sous Linux en ouvrant le fichier avec le drapeau *O_DIRECT* ou en utilisant l'option *direct* de la commande *dd*. Sous Windows en utilisant la fonction *CreateFile* avec l'option *FILE_FLAG_NO_BUFFERING*.

3.3 Contre-mesures

Contrer cette attaque tout en permettant le *montage* en lecture seule de clef USB n'est pas chose aisée.

Une contre-mesure serait d'empêcher les lectures directes depuis l'espace utilisateur, ainsi le cache mis en place par le système d'exploitation rendrait l'attaque plus complexe à implémenter. Néanmoins si un très grand nombre de fichier est utilisé par l'attaquant, il est possible que le cache ne soit pas capable d'empêcher finalement que des lectures directes soient nécessaires. De plus une fonctionnalité de la norme *USB Mass Storage* [4] permet de signaler à l'hôte que les données présentes sur le périphérique ont changé en renvoyant une erreur notée dans la

norme comme *NOT READY TO READY TRANSITION - MEDIA CHANGED*. Ainsi le cache du système d'exploitation supportant cette erreur sera effacé, rendant l'attaque à nouveau possible, quoique plus lente étant donné que chaque renvoi d'erreur génère des lectures ne véhiculant pas de données utiles.

Il reste possible de détecter un comportement suspect sur le bus USB (ex : lecture à des adresses semblant aléatoires). Ceci pourrait être réalisé par des éditeurs de logiciels de protection des ports USB comme [24] ou par des éditeurs d'antivirus, une telle détection nécessitant une forte intégration avec le système d'exploitation hôte.

Cette dernière contre-mesure paraît la plus robuste, mais est sans doute imparfaite tant le nombre de solutions disponibles à l'attaquant semble important (lecture sur de nombreux fichiers de petites tailles, usage en sa faveur de la norme *USB Mass Storage*).

4 Attaque 2 : Contournement des contrôles antivirus

Contrairement à l'attaque précédente, le but de celles-ci sont de permettre l'infection de la machine hôte du périphérique USB. Se sont des variantes de l'attaque [15], nommée *Read It Twice*, qui modifie dynamiquement le contenu du système de fichiers de la clef USB afin de leurrer l'OS hôte.

4.1 Contournement des sas de décontamination USB

Certains SI sensibles mettent en place des sas de décontamination des périphériques de stockage de masse USB afin d'éviter l'introduction de virus [14]. Ces sas peuvent prendre la forme de PC standard sur lequel un antivirus à jour nettoie chaque clef USB connectée. Ceci est une précaution légitime pour une clef USB classique, mais cette mesure devient totalement inopérante si l'on considère un dispositif USB malicieux.

L'attaque la plus triviale consiste à utiliser un dispositif qui se présente en lecture/écriture mais dont toutes les écritures ne sont pas réellement effectuées. Par exemple si le sas USB détecte un fichier vérolé, celui-ci va le supprimer du système de fichier présenté par la clef. Du point de vue du système hôte, l'opération d'effacement réussi mais le dispositif USB ne réalise aucune modification sur sa mémoire de masse. Comme l'hôte maintient un cache du système de fichier présent sur la clef, celui-ci ne verra plus le fichier même si il est toujours présent physiquement sur la clef. Après cette tentative de décontamination l'utilisateur

branchera à nouveau la clef sur une autre machine sur laquelle le fichier vérolé sera toujours présent.

Une contre-mesure simple consisterait à forcer le système à rafraichir son cache de système de fichier après l'effacement, si le fichier est toujours présent une alerte pourra être remontée à l'utilisateur ou au responsable de la sécurité. Néanmoins il est relativement simple de contourner cette contre-mesure. L'attaquant peut réaliser un dispositif qui ne fait apparaître les fichiers vérolés qu'après un certain nombre de branchement/débranchement. Par exemple, en considérant le schéma suivant, la clef USB présentera plusieurs fois le système de fichier sain avant de présenter sa version vérolée. Ainsi même si le sas USB lit plusieurs fois le système de fichier, celui-ci sera toujours exempt de fichier vérolé. Lorsque l'utilisateur branchera le dispositif sur la machine cible le fichier malveillant apparaîtra et pourra donner lieu à une infection virale. Cette attaque peut évidement être combinée avec l'attaque permettant de prendre le contrôle du clavier et de la souris de l'utilisateur pour automatiser l'activation de la charge virale.

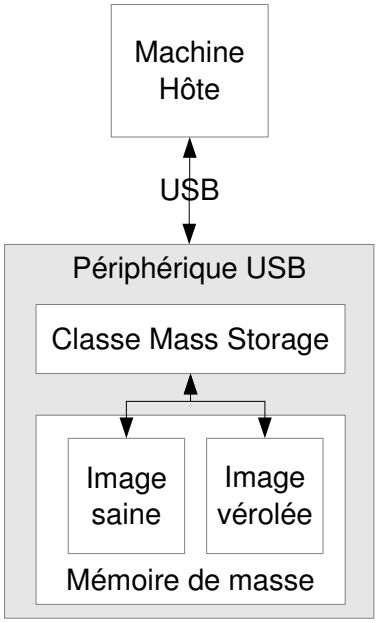

Figure 3. Modification dynamique du système de fichier

Une contre-mesure réellement efficace consiste dans l'utilisation d'un sas qui recopie les fichiers non vérolés sur une autre clef USB de confiance. Cette clef ne doit pas sortir de la zone de confiance pour éviter qu'elle soit remplacée par une clef USB malicieuse.

4.2 Contournement des contrôles antivirus

De la même manière on peut chercher à contourner le logiciel antivirus de la machine sur laquelle la clef USB malicieuse est branchée. Pour cela il semble pertinent d'employer la même approche que pour le contournement des sas de décontamination. Néanmoins le cache réalisé par l'OS sur les systèmes de fichiers fait que le fichier n'est lu qu'une seule fois. Ces attaques consistants à modifier le contenu des fichiers après le contrôle antivirus, pour le remplacer par du contenu malveillant ne fonctionnent donc pas.

On peut néanmoins utiliser deux méthodes pour forcer l'OS à relire le fichier malgré le cache :

- Utiliser un fichier de très grande taille afin qu'il ne puisse entièrement être stocké dans le cache de l'OS.
- Renvoyer une erreur spécifique depuis le périphérique USB malicieux vers l'OS : *NOT READY TO READY TRANSITION - MEDIA CHANGED*. Cet erreur, spécifié dans la norme USB Mass Storage permet de forcer la machine hôte à relire le contenu réellement présent sur la clef USB.

Cependant les essais effectués sur différents antivirus, montrent jusqu'ici qu'un contrôle viral est effectué avant le chargement en mémoire lorsque l'utilisateur ouvre un programme, et non pas uniquement lors de sa première lecture sur le disque. Aucun des essais réalisés n'a été concluant, néanmoins d'autres essais vont être effectués pour s'assurer que les principaux antivirus du marché ne sont pas vulnérables à ce type d'attaques, notamment en utilisant plutôt des documents vérolés de grandes taille (pdf ou documents Word).

5 Contre-mesures

Comme le montre cet article, la surface d'attaque du support USB d'un ordinateur est vaste et composée de nombreux sous éléments (matériel, driver, système de fichier, application). Une protection basée sur une solution logicielle classique (comme un anti-virus ou un logiciel spécialisé dans la protection USB) peut être relativement facilement contournée. Deux pistes peuvent néanmoins être envisagées : la première consiste dans un cloisonnement strict des fonctionnalités USB dans une machine virtuelle dédiée ; la seconde consiste à déléguer, à un système embarqué dédié, la gestion d'un port USB hôte, dans les deux cas la communication avec le système principal se fait d'une manière maîtrisée.

5.1 Approche Qubes OS

Qubes OS [18] est une distribution Linux, basée sur Fedora, qui revendique un haut niveau de sécurité en garantissant l'isolation des différentes tâches effectuées par l'utilisateur via l'usage de machines virtuelles Xen. Par exemple la machine virtuelle utilisée pour naviguer sur Internet est isolée de la machine utilisée pour les tâches sensibles.

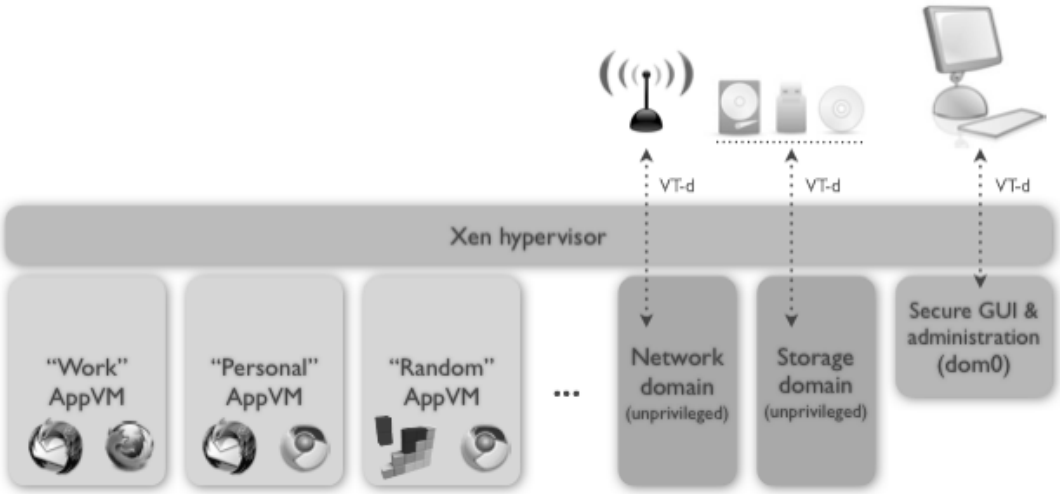

Figure 4. Architecture de Qubes OS

Cette approche de la sécurité par l'isolation permet notamment, grâce à la configuration de l'IOMMU, de restreindre la portée des attaques DMA pouvant provenir des différents périphériques de la machine (ex : cartes PCI Express). Elle permet également de protéger la machine virtuelle (VM) de plus haute sécurité d'attaques logiques pouvant subvenir sur les VMs de moindre niveau de sécurité (buffer overflow, attaques sur la pile réseau). Cette approche est également employée pour éviter les problématiques liées à l'USB [19]. Il est possible de connecter un contrôleur USB à une machine virtuelle particulière, par exemple à une machine virtuelle de faible niveau de sécurité dont la compromission est peu problématique.

Le fonctionnement de l'IOMMU, sur laquelle repose en grande partie la sécurité de Qubes OS, empêche cependant d'assigner un port USB particulier à une machine virtuelle mais uniquement de lui assigner un contrôleur USB dans son ensemble. De ce fait différents périphériques USB branchés sur un même contrôleur ne peuvent pas être partagés entre plusieurs machines.

Ceci pose donc différents problèmes, par exemple une clef 3G qui devrait être assignée à la VM chargée de la gestion du réseau ne pourra pas être utilisée en même temps qu'un lecteur biométrique pour l'authentification qui devrait être réservé à une VM de haut niveau de sécurité. De même l'usage de périphériques USB HID n'est pas réellement possible car clavier et souris sont gérés par la machine de plus haut niveau de sécurité qui pourrait ainsi être compromise par cette classe USB, Qubes OS recommande donc l'usage de périphériques PS/2. Cette approche permet néanmoins d'utiliser des clefs USB, de confiance ou non, afin de transférer des fichiers depuis une machine de faible niveau de confiance vers une zone plus sûre en utilisant le mécanisme de copie de fichier inter domaines de Qubes OS, via une action de l'utilisateur.

Il apparaît donc que la plus part des attaques présentées dans cet article peuvent être limitées par Qubes OS. Les attaques sur les différents niveau d'abstraction de l'USB resteront cantonnées à la VM gérant l'USB. Les attaques utilisant des périphériques *Mass Storage* malicieux (présentées au chapitre 4) serait également cantonnées à la VM gérant l'USB, le mécanisme de copie inter domaine permettant de ne pas réaliser les lectures directement sur le périphérique USB. L'attaque présentée au chapitre 3 ne pourrait aboutir qu'à la fuite de données présentes au sein de la VM gérant l'USB.

Cette méthode est certainement l'une des meilleurs solutions pour protéger des utilisateurs Linux avertis. Mais le remplacement des systèmes existants, la qualification et le déploiement de ce type de solution ainsi que la formation nécessaire pour les utilisateurs rend cette approche peu applicable. De plus, une grande partie des choix de sécurité revient à l'utilisateur, comme le transfert de fichiers d'une VM de bas niveau de sécurité (VM USB) vers une VM de plus haut niveau, ce qui peut aboutir à des erreurs de manipulations. À noter qu'une version Windows est en cours de développement ce qui pourrait rendre cette solution plus attractive [20].

5.2 Approche matérielle

Nous proposons ici une approche matérielle comme contre-mesure. Le dispositif, prenant la forme d'une carte PCI ou PCI Express, que l'on dispose dans un poste de bureau standard, la carte dispose en face arrière d'un port USB, les autres ports USB de la machine doivent être désactivés (résine, suppression physique).

La carte est vue par le système comme un contrôleur USB classique. Sans qu'aucun dispositif ne soit branché sur la carte, elle présente au

poste hôte trois périphériques USB : un clavier, une souris et une clef USB. La carte peut donc fonctionner avec les driver natifs USB (HID et *Mass Storage*) ce qui évite toute installation ou paramétrage du poste hôte. Lorsqu'un clavier, une souris ou une clef USB sont branchées sur ce port (via un hub ou directement) le système embarqué les reconnait et les fait correspondre aux périphériques virtuels présentés au poste hôte. Si un périphérique d'une autre classe USB est branché, il ne sera pas présenté au poste hôte.

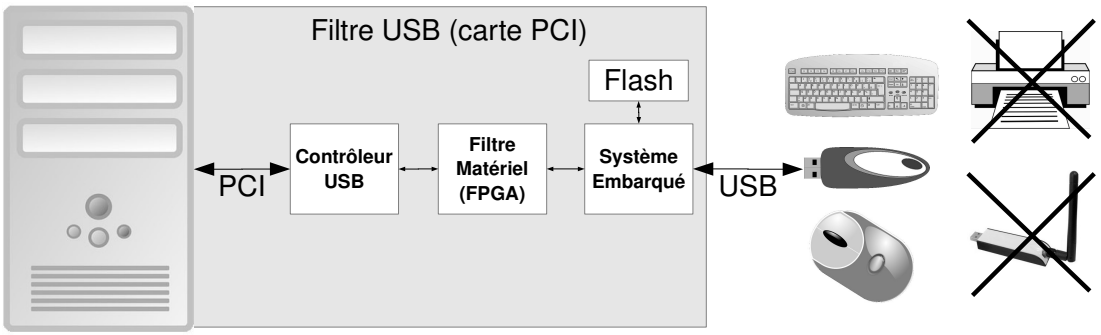

Figure 5. Contre-mesure matérielle

La carte est pourvue d'un système embarquée qui est chargé de :
- De gérer un clavier et/ou une souris via son port USB dédié
- De lire et de recopier le contenu des clefs USB, que l'on branche sur son port USB, vers une mémoire Flash interne
- De relayer au filtre matériel les informations provenant du clavier et de la souris, et de lui présenter la mémoire Flash interne comme une clef USB standard

Le filtre matériel réalise un filtrage configuré en usine. Concernant la classe HID, le filtrage fonctionne comme suit :
- Certaines combinaisons de touches du clavier peuvent être interdites (ex : *Win+R*)
- La vitesse de frappe et de déplacement de la souris peuvent être limités
- Seuls les codes HID connus et autorisés sont transférés au système hôte (les touches standards d'un clavier, le mouvement et les clics de la souris), les codes inconnus ou interdits ne sont pas transmis.
- Les descripteurs USB du clavier et de la souris virtuelles présentés au système hôte sont codés en dur et ne proviennent pas des périphériques USB réels, ils ne sont pas modifiables.

Pour la classe *Mass Storage* le filtrage suivant est effectué :

- Comme pour le HID, le descripteur USB de la clef USB virtuelle ne provient pas du périphérique réel et n'est pas modifiable
- La table des partitions ne provient pas de la clef USB réelle et n'est pas modifiable
- L'entête du système de fichier est également codé en dur
- Des vérifications sont réalisées sur la table d'allocation des fichiers et la structure des répertoires
- Les noms de fichiers et leurs extensions peuvent être filtrés (liste blanche, caractères autorisés)
- Le contenu des fichiers peut être filtré au caractère près (ex : uniquement des caractères imprimables, interdire tous les caractères spéciaux)
- Le sens des transfert peuvent être restreint (lecture seule, écriture seule, lecture/écriture)

Ce filtrage matériel est effectué indépendamment du système embarqué. Si malgré son durcissement celui-ci venait à être attaqué, le filtrage matériel reste opérationnel et fonctionnera comme spécifié en usine.

Les différentes couches d'abstraction USB présentées au système hôte (matériel, descripteurs et classe USB, table des partitions, systèmes de fichiers) proviennent uniquement du matériel de sécurité et non pas des périphériques réels ce qui élimine les vulnérabilités introduites par les périphériques USB classiques et malicieux. Les lectures effectuées sur la clef USB virtuelle sont faites directement sur la mémoire Flash interne et non pas sur la clef USB réelle évitant ainsi les attaques présentées au chapitres 3 et 4. Ce dispositif implémente donc une isolation stricte des périphériques externes tout en permettant aux utilisateurs d'avoir un usage normal des classes HID et *Mass Storage*.

On peut également noter que ce dispositif réduit les risques liés aux périphériques HID malicieux en limitant la vitesse de frappe et l'usage de certaines combinaisons de touches, mais il n'annule pas totalement la possibilité d'un attaquant d'usurper l'identité de l'utilisateur en contrôlant son clavier et sa souris.

Bien que cette solution ne soit pas adhérente à la configuration du poste hôte, elle impose des contraintes notamment au niveau des fichiers pouvant être lus. En effet pour garantir un haut niveau de sécurité la méthode la plus drastique consiste à filtrer tous les caractères spéciaux (*'/'*, *'{'*, *'&'*, *'%'* , ...) pour éviter par exemple les documents pdf vérolés. Ceci impose donc un format de fichier texte restreint mais qui convient pour certains cas d'usage (lecture de fichiers de logs, de fichiers de configuration spécifiques).

Ce dispositif ne supporte actuellement que les classe HID et *Mass Storage*, il ne permet donc pas l'usage d'imprimantes USB par exemple, ce qui peut être contraignant pour certains usages.

6 Conclusion

Comme tout bus bi-directionnel le port USB d'une machine l'expose à la fois à des infections virales mais également à la fuite d'informations confidentielles. Il n'est généralement pas possible de réaliser des opérations de lecture ou d'écriture directement dans la mémoire du système hôte (accès *DMA*) comme c'est le cas avec les bus *firewire, PCI ou PCI Express*. Néanmoins il apparaît dans cet article que de nombreuses vulnérabilités sont tout de même présentes. De plus la norme USB OTG (*On The Go*) pourrait suivant certains travaux [29] introduire de nouvelles vulnérabilités.

Comme le montre la figure 6, il apparaît que les mesures de précautions généralement préconisées sont globalement efficaces face à des périphériques USB classique, mais sont quasiment impuissantes face à des dispositifs USB dont le logiciel embarqué a été modifié par un attaquant pour lui conférer des fonctionnalités pour contourner ces mesures de protections.

Vulnérabilité	Protection	Protection Valide pour un dispositif USB classique	Protection alide pour un dispositif USB malicieux
Fuite d'information	Supprimer toutes les interfaces USB	Oui	Oui
	Interdire tous les périphériques sauf HID	Oui	Non (fuite via Caps Lock, Num lock …)
	Monter les disques USB en lecture seule	Oui	Non (cet article)
Virus	Supprimer toutes les interfaces USB	Oui	Oui
	Interdire tous les périphériques sauf HID	Oui	Non (PHUKD)
	Utilisation de sas de décontamination	Oui	Non (cet article)
	Utilisation d'antivirus	Oui (si virus connu)	…

Figure 6. Tableau comparatif des menaces induites par les dispositifs USB malicieux

Les attaques présentées dans cet article permettent d'une part de contourner certaines protections basées sur des sas antivirus, et d'autre part de réussir à écrire sur un disque USB même si celui-ci est *monté* en lecture seule par le système hôte. Ces attaques peuvent être combinées à des attaques existantes pour améliorer leur efficacité, notamment les attaques permettant de prendre le contrôle du clavier et de la souris de l'utilisateur automatisant ainsi l'exécution de l'attaque.

Cet article se concentre sur les failles présentes dans le protocole USB mais il est important de noter que tout port, même une prise jack audio,

peut être un vecteur d'attaque. La conversion de l'audio vers des données numériques est par exemple utilisée par la société Square [30] pour permettre le dialogue entre un iphone et leur lecteur de carte à puce. Il est donc tout à fait envisageable de concevoir un périphérique permettant de récupérer de l'information via un canal audio rarement protégé par les responsables de la sécurité. Le même type d'attaque est également possible en sens inverse, elle consisterait à introduire un contenu malveillant sur la machine hôte en utilisant un logiciel permettant de convertir l'entrée audio en fichier. Il est donc préférable de supprimer physiquement tous les ports non utilisés d'une machine contenant des informations sensibles ou de les protéger en utilisant un matériel de sécurité adéquat.

Références

1. Locking down Windows Vista and Windows 7 against Malicious USB devices. http://www.irongeek.com/i.php?page=security/locking-down-windows-vista-and-windows-7-against-malicious-usb-devices.
2. Programmable HID USB Keystroke Dongle : Using the Teensy as a pen testing device. http://www.irongeek.com/i.php?page=security/programmable-hid-usb-keystroke-dongle.
3. USB Gadget API for Linux. http://www.kernel.org/doc/htmldocs/gadget.html.
4. Universal Serial Bus Mass Storage Class UFI Command Specification. http://www.usb.org/developers/devclass_docs/usbmass-ufi10.pdf, 1998.
5. Device Class Definition for Human Interface Devices (HID). http://www.usb.org/developers/devclass_docs/HID1_11.pdf, 2001.
6. BACKDOORING MP3 FILES. http://www.gnucitizen.org/blog/backdooring-mp3-files/, 2006.
7. A DAM C ORNELISSEN. Covert Channel Data Leakage Protection. www.ru.nl/publish/pages/578936/acornelissen.pdf, 2012.
8. Adrian Crenshaw. Plug and Prey : Malicious USB Devices. http://www.irongeek.com/i.php?page=security/plug-and-prey-malicious-usb-devices#3.2_Locking_down_Linux_using_UDEV, 2011.
9. Aleksandr Matrosov, Eugene Rodionov, David Harley, Juraj Malcho. Stuxnet Under the Microscope . http://go.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf, 2012.
10. Andras Veres-Szentkiralyi . USB = Universal Security Bug ? . http://silentsignal.hu/docs/S2_VSzA_Hacktivity2012.pdf.
11. ANSSI. Maîtriser la SSI pour les systèmes industriels . http://www.ssi.gouv.fr/IMG/pdf/Guide_securite_industrielle_Version_finale.pdf, 2012.
12. BECRYPT. Becrypt Advanced Port Control . http://www.becrypt.com/assets/files/Advanced%20Port%20Control.pdf.
13. CERTA. NOTE D'INFORMATION DU CERTA : Risques associés aux clés USB. http://www.certa.ssi.gouv.fr/site/CERTA-2006-INF-006/, 2009.

14. CNRS. Sécurité de l'information, Clefs USB : pratiques mais risquées . http://www.dgdr.cnrs.fr/fsd/securite-systemes/revues-pdf/Si11.pdf.
15. Collin Mulliner and Benjamin Michèle. Read It Twice ! A Mass-Storage-Based TOCTTOU Attack. In *WOOT '12*, 2012.
16. David Maynor. 0wn3d by everything else - USB/PCMCIA Issues. In *CanSecWest/core05*, 2005.
17. EVERSTRIKE. Secure computers against unwanted USB devices. http://www.everstrike.com/usbsecurity/.
18. Joanna Rutkowska. Site Web de Qubes OS. www.qubes-os.org/.
19. Joanna Rutkowska. USB Security Challenges . http://theinvisiblethings.blogspot.fr/2011/06/usb-security-challenges.html.
20. Joanna Rutkowska. Windows support coming to Qubes ! http://theinvisiblethings.blogspot.fr/2012/03/windows-support-coming-to-qubes.html.
21. Jon Larimer. USB Autorun attacks against Linux. http://blogs.iss.net/archive/papers/ShmooCon2011-USB_Autorun_attacks_against_Linux.pdf.
22. Jon Larimer IBM X-Force Advanced R&D . USB Autorun attacks against Linux . http://blogs.iss.net/archive/papers/ShmooCon2011-USB_Autorun_attacks_against_Linux.pdf, 2011.
23. Laboratory of Cryptography of Systems Security (CrySyS). Duqu : A Stuxnet-like malware found in the wild, technical report. http://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf, 2011.
24. LUMENSION. Lumension Endpoint Management and Security Suite : Device Control. http://www.lumension.com/device-control-software/usb-security-protection.aspx.
25. MICROSOFT. How to disable the Autorun functionality in Windows. http://support.microsoft.com/kb/967715.
26. Microsoft. Vulnerabilities in Kernel-Mode Drivers Could Allow Elevation Of Privilege. http://technet.microsoft.com/en-us/security/bulletin/MS13-027, 2013.
27. NETWRIX. NetWrix USB Blocker . http://www.netwrix.com/usb_blocker_freeware.html.
28. Phil Polsta. Bypassing endpoint security with USB keys. https://media.defcon.org/dc-20/presentations/Polstra/DEFCON-20-Polstra-Bypassing-Endpoint-Security.pdf.
29. Rory Breuk, Albert Spruyt1 . Integrating DMA attacks in exploitation frameworks . http://staff.science.uva.nl/~delaat/rp/2011-2012/p14/report.pdf, 2012.
30. SQUARE. Site Internet de la société Square. https://squareup.com/.
31. SYMANTEC. How Symantec Endpoint Protection Device Control processes Windows device GUIDs and device IDs. http://www.symantec.com/business/support/index?page=content&id=HOWTO60964, 2011.
32. Timo Warns. Multiple Linux kernel vulnerabilities in partition handling code of LDM and MAC partition tables. http://www.securityfocus.com/archive/1/516615.

33. Travis Goodspeed. Writing a Thumbdrive from Scratch, Prototyping Active Disk Antiforensics. `http://events.ccc.de/congress/2012/Fahrplan/events/5327.en.html`.

34. Travis Goodspeed. Facedancer USB : Exploiting the Magic School Bus. In *Recon 2012*, 2012.