Machines virtuelles protégées

Jean-Baptiste Galet
jean-baptiste.galet@ssi.gouv.fr

ANSSI

Résumé. Cet article traite de la sécurité des machines virtuelles vis-à-vis des hyperviseurs. L'externalisation croissante et la généralisation de la virtualisation apportent de nouvelles problématiques liées à la confiance des hyperviseurs et des administrateurs. Les différents éditeurs de produits de virtualisation ont développé des solutions permettant de répondre à ces problématiques. Dans un premier temps, au travers d'une étude comparative de trois produits, cet article vise à mettre en lumière les risques et les solutions apportées ; puis dans un second temps, l'étude détaillée du fonctionnement du produit Hyper-V permettra de comprendre les mécanismes de protection utilisés.

1 Introduction

Il est de plus en plus fréquent de rencontrer des environnements où la majeure partie des systèmes sont virtualisés, y compris les plus critiques. Cette virtualisation peut être mise en place dans l'entreprise, chez des hébergeurs tiers ou chez des fournisseurs de cloud.

Du fait de la généralisation du recours à la virtualisation, les menaces historiques (VM vers VM ou VM vers l'hôte) se voient complétées par les menaces de l'hôte vers les VM et plus généralement de l'infrastructure d'hébergement vers les VM.

Dans ce cadre on cherche à protéger l'intégrité et la confidentialité de la VM contre plusieurs types d'acteurs :

- les administrateurs des systèmes de stockage : ceux-ci peuvent accéder aux disques des VM et peuvent en extraire des informations (bases d'authentification, données métier) ou les altérer (ajout de portes dérobées) ;
- les administrateurs des systèmes de sauvegarde : comme pour le stockage, ceux-ci ont un accès en lecture aux disques ;
- les administrateurs réseau : ils peuvent observer le trafic réseau entre les différents composants de l'infrastructure et obtenir des informations sur les VM :

    - entre les systèmes de stockage et les hyperviseurs : accès à des données du disque ;
    - entre les hyperviseurs lors des opérations de migration : accès aux données en mémoire vive (secrets cryptographiques, éléments d'authentification en clair) ;
- les administrateurs des hyperviseurs : ils peuvent disposer d'accès bas niveau aux systèmes d'exploitation et ainsi obtenir un contrôle total sur la VM (accès et altération des données des disques et de la mémoire vive) ;
- les personnes ayant un accès physique aux hyperviseurs : ils peuvent modifier le matériel (configuration de l'UEFI, ajout de périphériques), modifier le système d'exploitation et accéder aux données locales (accès aux disques physiques).

Évidemment protéger les machines virtuelles de l'infrastructure qui les héberge est un problème complexe qui nécessite l'utilisation de mécanismes avancés et une grande confiance dans les solutions qui les mettent en œuvre.

2 Glossaire

2.1 Matériel / Plateforme

- *TPM – Trusted Platform Module* : standard pour des processeurs cryptographiques – le terme est aussi utilisé pour les implémentations de ce standard
- *PCR – Platform Configuration Register* : registres dans un TPM, utilisés principalement pour réaliser des mesures d'intégrité
- *EKPub* : Partie publique de l'*Endorsement Key* du TPM, unique pour chaque TPM (non modifiable)
- *TcgLog* : Journal des mesures du TPM, stockées dans une zone mémoire de l'ACPI pointée par le *Measurement Log Pointer* (LASA) dans la table TCPA.
- *PK – Platform Key* : clé racine dans la hiérarchie des clés UEFI
- *KEK – Key Exchange Key* : clés autorisées à modifier les bases de signature UEFI

2.2 Technologies VMware

- *KMS – Key Management Service*
- *vMotion* : mécanisme de transfert d'une machine d'un hyperviseur à l'autre

– *VIB – vSphere Installation Bundle* : paquets de modules noyau et d'applications
– *DEK – Data Encryption Key* : clé utilisée pour chiffrer les données
– *KEK – Key Encryption Key* : clé utilisée pour chiffrer la DEK

2.3 Technologies Microsoft

– *HGS – Host Guardian Service* : serveur d'attestation et de clés
– *VBS – Virtualisation Based Security* : modèle de sécurité utilisant la virtualisation
– *VTL – Virtual Trust Level* : environnement dans l'architecture VBS
– *SKM – Secure Kernel Mode* : noyau du VTL 1
– *PPL – Protected Process Light* : protection appliquée sur certains processus limitant les interactions possibles avec celui-ci
– *KVP – Key-Value Pair* : mécanisme de communication entre Hyper-V et une machine virtuelle (Hyper-V Data Exchange Service)

3 Rappels

3.1 TPM

Le fonctionnement et l'architecture des TPM font l'objet de nombreux articles (dont [1]). Ces dernières années, la présence d'un TPM s'est généralisée dans les ordinateurs fixes et mobiles (notamment grâce aux prérequis pour les OEM Windows 10 depuis juillet 2016). Ces TPM peuvent être de plusieurs types :

– dédiés : sur des puces séparées ;
– intégrés : utilisant du matériel spécifique embarqué dans un autre composant, mais isolé logiquement des autres composants ;
– firmware : logiciel s'exécutant dans le firmware d'un processeur ;
– logiciels : émulateurs logiciels ;
– virtuels : fournis par l'hyperviseur à une machine virtuelle (implémentation logicielle).

Deux générations existent, non compatibles entre elles :

– TPM 1.2 (2011) ;
– TPM 2.0 (2014 – 2015).

TPM 2.0 apporte notamment de nouveaux algorithmes cryptographiques (SHA-256, ECC, AES), des fonctions de dérivation de clé et une architecture à plusieurs niveaux.

Les TPM peuvent stocker des clés cryptographiques symétriques ou asymétriques de manière sécurisée. Une de ces clés est un biclé généré par le fabricant du TPM, unique par matériel : l'*Endorsement Key* (*EK*) dont les parties privées et publiques sont notées respectivement *EKPriv* et *EKPub*. Cette clé peut éventuellement être signée par le fabricant avec un certificat (*EKCert*). Ce biclé est utilisé pour l'identification du TPM.

Les TPM contiennent des registres appelés PCR (*Platform Configuration Register*) qui ont la particularité de ne pas pouvoir être écrits directement, mais « étendus ».

$$TPM\_PCRExtend(n, digest) := PCR[n] \leftarrow hash(PCR[n], digest)$$

Ces PCR sont intéressants notamment pour leur capacité à mesurer le démarrage de la machine : les différents composants (CRTM, UEFI, bootloader) étendent les PCR du TPM avec des données liées au démarrage (données de configuration, mesure des composants suivants, etc.). Les données utilisées pour étendre les PCR sont par ailleurs journalisées dans une zone mémoire de l'ACPI afin de permettre à un tiers de vérifier les valeurs des registres en reproduisant leur génération (avec un TPM logiciel par exemple).

Les PCR sont répartis en *banks* : chaque *bank* associe un algorithme de hash avec un ensemble de registres. Avec TPM 2.0, Les opérations de mesures sont généralement effectuées sur plusieurs *banks* (e.g. SHA1 et SHA256).

Le TPM permet ensuite de lire les valeurs des PCR ainsi que d'effectuer des opérations à partir de leur valeurs :

- générer une *quote* (signature des PCR et d'un nonce avec une clé du TPM), par exemple ;
- chiffrer / déchiffrer des données (*seal* / *unseal*).

3.2 Démarrage sécurisé

Secure Boot Le premier composant logiciel à s'exécuter sur un PC est le bootloader qui a pour rôle de charger le système d'exploitation (e.g. ntldr, bootmgr, GRUB). Sans Secure Boot, le bootloader est chargé sans vérifications, qu'il soit légitime ou bien un « Bootkit ».

L'UEFI d'un PC disposant de Secure Boot vérifie l'intégrité et la signature du bootloader avant de le charger. L'UEFI dispose de trois bases pour vérifier les signatures :

- **db** : autorités de certification pouvant émettre des certificats de signature de code, utilisés pour signer les bootloaders ;

- **dbx** : certificats de signature de code révoqués ou compromis ;
- **dbt** : certificats de signature de temps utilisés pour contre signer les bootloaders.

Une autre base, la **KEK** (*Key Exchange Key*) contient les certificats autorisés à modifier les bases db, dbx et dbt. Enfin, la KEK est protégée par la **PK** qui est la racine de confiance de la plateforme.

Trusted Boot Le bootloader peut mettre en place des mécanismes de vérification des composants qu'il charge, et ainsi de suite en cascade. Ces mécanismes sont désignés par le terme Trusted Boot. Il est possible de réaliser du Trusted Boot sans Secure Boot ; cependant, il peut être contourné en modifiant le bootloader.

Measured Boot Les différents composants impliqués dans Secure Boot (UEFI) et Trusted Boot (bootloader, noyau, etc.) peuvent effectuer des mesures dans le TPM, permettant par la suite au système d'exploitation d'auditer sa chaîne de démarrage ou de les fournir à un service d'attestation.

4 Présentation des solutions

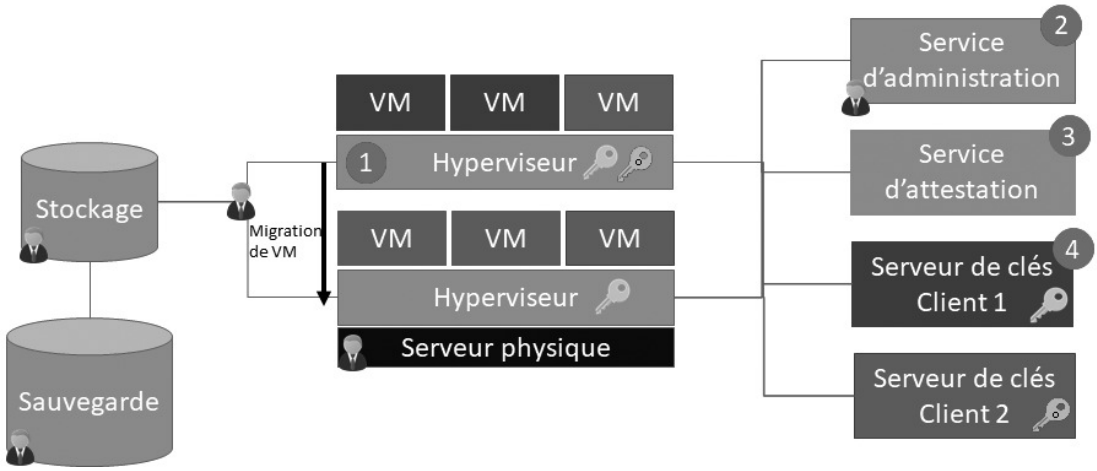
Les éditeurs de solutions de virtualisation ont pris en compte cette problématique et proposent des solutions techniques dans leurs versions les plus récentes. Le développement de ces solutions vise à lever certaines craintes liées à l'externalisation vers le cloud et constitue un argument commercial.

On peut citer :

- vSphere 6.5 (VM encryption, Secure Boot, vMotion encryption), sorti en octobre 2016 ;
- Hyper-V sous Windows Server 2016 (Shielded-VM, Guarded Fabric), sorti en septembre 2016 ;
- Open CIT, publié en avril 2016.

Ces solutions sont relativement récentes et ne font pas l'objet de présentations, fonctionnelles ou techniques. Leur étude permet d'introduire les différentes technologies sous-jacentes qu'elles mettent en place et quels en sont leurs usages, avantages et limitations. Une étude plus poussée sur la solution de Microsoft, Shielded-VM sous Hyper-V, sera présentée, avec notamment des détails précis sur les protocoles ou les mécanismes bas niveau mis en place dans le système d'exploitation.

Les solutions présentées s'articulent autour des mêmes composants (voir figure 1) : un ensemble d'hyperviseurs, un service d'administration, un service d'attestation et des serveurs de clés.



Services	vSphere 6.5	Hyper-V	Open CIT
①	ESXi 6.5	Windows Server 2016	Nova Compute
②	vCenter	System Center VMM	Horizon
③	vCenter	HGS	Attestation Server
④	KMS KMIP	HGS	Barbican

Fig. 1. Architecture générique

4.1 vSphere 6.5

La solution proposée par VMware a pour but de simplifier le chiffrement des données qui reposent traditionnellement sur des solutions physiques (disque chiffrant, chiffrement sur le fabric SAN, etc.) ou spécifiques à chaque système d'exploitation. Elle permet aussi de limiter les possibilités d'accès aux données notamment aux administrateurs réseau, stockage ou fonctionnels.

Protection des hyperviseurs Les hyperviseurs ESXi 6.5 supportent SecureBoot. Le démarrage sécurisé est réalisé par :

- l'UEFI qui vérifie le bootloader ;
- le bootloader qui vérifie le noyau de l'hyperviseur (VMKernel) et le *VIB verifier* ;
- le *VIB verifier* qui vérifie la signature de l'ensemble des VIB (*vSphere Installation Bundle*).

En cas de problème, l'hyperviseur ne démarre pas ou affiche un écran d'erreur (PSOD – *Purple Screen Of Death*). L'état de la machine est ensuite remonté au vCenter.

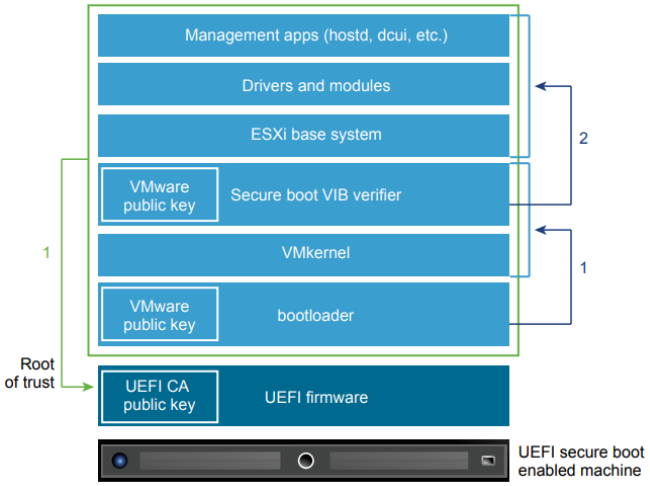


Fig. 2. Démarrage sécurisé d'un ESXi [10]

Protection des machines virtuelles Dans la version 6.5 de vSphere, VMware a introduit la notion de *VM Encryption* qui permet de chiffrer les machines virtuelles au niveau de l'hyperviseur et non du système invité (le système invité ne possède donc pas les clés de chiffrement dans sa mémoire). Le vCenter sert d'intermédiaire entre l'ESXi et un service de gestion de clés (KMS – *Key Management Service*) compatible avec la norme *KMIP 1.1*[1].

Le chiffrement d'une machine est réalisé avec 2 clés (voir figure 3) :

- la KEK (Tenant key – AES-256), stockée dans le KMS ;
- la DEK (Internal key – AES-256) stockée dans le fichier de configuration de la machine virtuelle (VMX) dans le paramètre **encryption.data**, chiffrée avec la KEK (référencée dans le paramètre **encryption.keySafe**).
  Elle est générée par l'hyperviseur qui réalise le chiffrement initial (XTS-AES-256) de la machine virtuelle.

Lors du démarrage d'une machine l'ESXi demande au vCenter de fournir la KEK correspondant à la machine virtuelle à partir de l'identifiant

[1] `https://wiki.oasis-open.org/kmip/KnownKMIPImplementations`

de la clé. Cette clé est conservée dans la RAM de l'ESXi pour chiffrer ou déchiffrer les DEK associées.

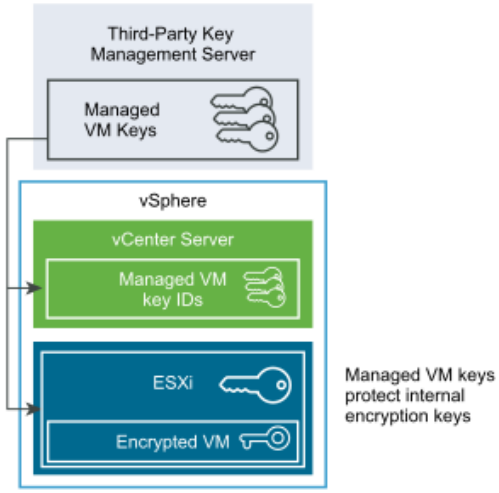


Fig. 3. Gestion des clés pour VMEncryption [9]

Avec cette solution, le disque virtuel est chiffré, ainsi que les clichés instantanés, la mémoire vive, etc. Le déplacement à la volée de la machine virtuelle est aussi chiffré (*Encrypted vMotion*).

Les machines supportent par ailleurs le démarrage avec *Secure Boot* : l'EFI virtuel de l'hyperviseur réalise les vérifications, similaires à celles de l'UEFI d'une machine physique.

Limites Il n'existe pas à ce jour de mécanisme qui limite la diffusion des clés de chiffrement aux hyperviseurs ayant démarré avec SecureBoot. Les interfaces graphiques vSphere Client ne permettent pas de distinguer les hyperviseurs avec ou sans SecureBoot.

Les clés de chiffrement sont demandées par les hyperviseurs dès leur démarrage (le démarrage de VM n'est pas nécessaire).

Il est ainsi possible de piéger un hyperviseur, d'obtenir les clés de chiffrement et de déchiffrer les machines virtuelles.

Conclusion La solution retenue par vSphere rend le chiffrement des machines virtuelles transparent pour les systèmes d'exploitation des machines virtuelles. La protection des données repose sur la bonne gestion des droits (notamment avec l'utilisation du rôle *No Cryptography Administrator*), la protection des serveurs vCenter (qui possèdent les informations de

connexion aux KMS) et à la protection des hyperviseurs (les clés sont en RAM et des outils console permettent d'obtenir les clés auprès du vCenter).

De plus, l'absence de mécanisme d'attestation à distance ne permet pas au vCenter de vérifier l'intégrité d'un hyperviseur avant de délivrer les clés.

Cette solution permet donc bien de se protéger contre les administrateurs réseau, les administrateurs de stockage et les administrateurs des systèmes de sauvegarde ; mais pas des administrateurs des hyperviseurs ou d'un accès physique à l'hyperviseur.

4.2 Hyper-V

Windows Server 2016 a introduit plusieurs concepts :

- *Host Guardian Service* (HGS) : ce rôle Windows Server permet de mesurer la santé (*Attestation*) et de distribuer des clés (*Key Protection*) à des machines Hyper-V ;
- *Guarded Host* : une machine ayant été jugée saine par le HGS et pouvant accueillir des *Shielded-VM* ;
- *Shielded-VM* : une machine virtuelle « blindée » pour laquelle des fonctionnalités de sécurité sont forcées et des restrictions d'accès sont appliquées ;
- *Guarded Fabric* : une infrastructure mettant en œuvre des *Shielded-VM*.

Historiquement réservées aux éditions serveur, ces fonctionnalités sont disponibles sur Windows 10 Enterprise depuis la version 1709 (RS3).

Fonctionnalités de sécurité reposant sur la virtualisation Les systèmes Windows 10 et Windows Server 2016 permettent de mettre en place des fonctionnalités appelées VBS (*Virtualisation-based security*) qui reposent sur l'hyperviseur Hyper-V. Ainsi, lorsque ces fonctionnalités sont activées, le système d'exploitation de la machine n'est plus démarré directement, mais est exécuté dans un hyperviseur Hyper-V.

À côté du système d'exploitation standard, est aussi virtualisé un environnement sécurisé avec un « noyau sécurisé ». Les deux environnements sont appelés VTL (*Virtual Trust Level*) :

- *VSM Normal mode* - VTL0 ;
- *VSM Secure mode* - VTL1.

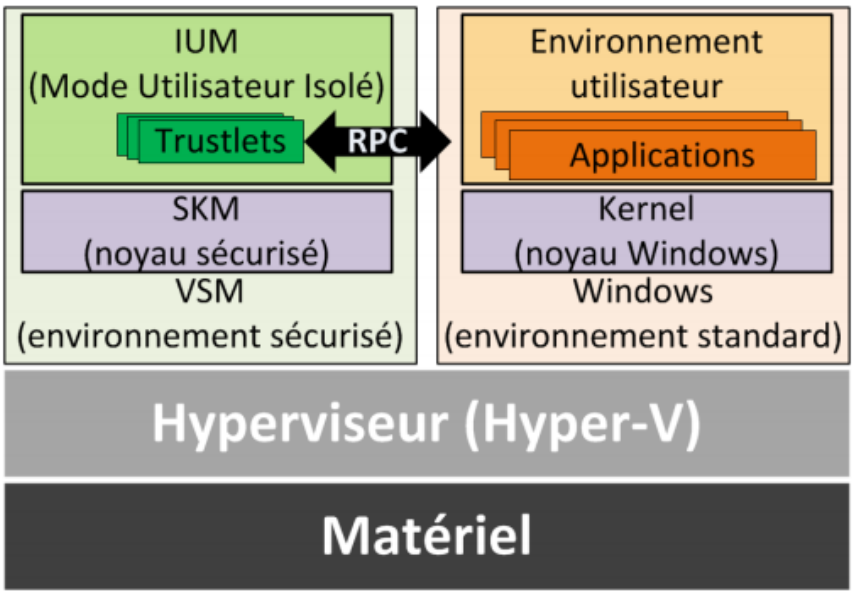


Fig. 4. Représentation simplifiée de l'architecture du Virtual Secure Mode [2]

Dans l'environnement sécurisé, VTL1, les processus sont des exécutables spécifiques appelés trustlets et doivent être signés de manière spécifique par Microsoft pour être chargés. Les trustlets n'ont accès qu'à un nombre limité de bibliothèques (aussi signées de manière spécifique) et ne peuvent communiquer vers le VTL0 qu'avec un nombre limité de moyens :

- chaque trustlet dispose de 8 *Mailbox Slots* de 4ko pour échanger avec le noyau du VTL0, ces *Mailbox* peuvent être protégées par une clé positionnée lors de son lancement ;
- les trustlets peuvent utiliser des objets de synchronisation (événements, sémaphores, etc.) communs avec le VTL0 ;
- les trustlets peuvent exposer des interfaces RPC au VTL0.

La mise en œuvre de ces fonctionnalités est décrite dans un guide de l'ANSSI [2].

Protection des hyperviseurs Ici nous ne nous intéressons qu'aux *Guarded Hosts*.

Afin de pouvoir être jugées saines par un HGS, les machines hébergeant les services Hyper-V doivent disposer :

- d'un TPM 2.0 ;

– d'un UEFI 2.3.1 ou supérieur (support de SecureBoot) ;
– d'une IOMMU et de SLAT (Second Level Address Translation) ;
– de SecureBoot activé.

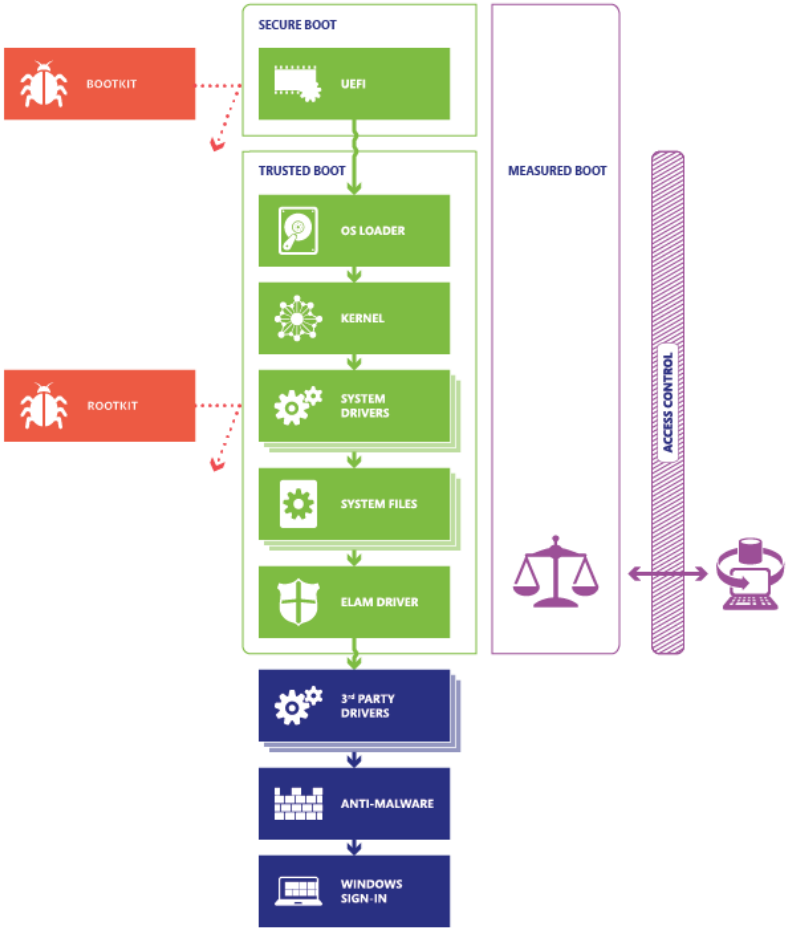


Fig. 5. Démarrage sécurisé d'un OS Windows [6]

Il existe deux niveaux de protection :

– mode Active Directory : ce mode n'offre aucune garantie sur la santé réelle de la machine et est simplement basé sur l'appartenance à un groupe Active Directory ;
– mode TPM : la machine doit mettre en œuvre une politique Device Guard (*CIPolicy*) et fournir ses traces de démarrage (TcgLog) au HGS pour qu'il évalue sa santé.

Protection des machines virtuelles Lors de leur passage en *Shielded-VM*, les machines virtuelles bénéficient des protections suivantes :

- SecureBoot ;
- TPM 2.0 (TPM virtuel Hyper-V) ;
- Restriction d'accès à la console virtuelle ;
- Restriction des interactions avec la machine hôte (WMI, KVP, copie directe, injection d'éléments) ;
- *Live Migration* (déplacement à chaud entre hyperviseurs) chiffré ;
- États et crash dumps chiffrés ;
- Le processus `vmwp.exe` (worker process) de la machine virtuelle s'exécute en PPL.

Les machines blindées, notamment le fichier contenant les données du TPM virtuel, sont protégées par 2 types de clés :

- la clé du propriétaire ;
- les clés des HGS pouvant déverrouiller la machine.

Ainsi, il est possible de chiffrer la machine virtuelle directement avec BitLocker en s'appuyant sur ce TPM virtuel et ainsi garantir la confidentialité des données ainsi que l'ancre de confiance.

4.3 OpenCIT

Open CIT est une solution Open Source[2] issue des travaux d'Intel (Intel CIT). Cette solution repose sur les composants suivants :

- *Attestation Server* : détermine l'état de santé d'un hyperviseur ;
- *Key Broker Service* : délivre les clés aux hyperviseurs ;
- *Trust Agent* : composant logiciel sur les hyperviseurs utilisé pour communiquer avec l'*Attestation Server*.

L'attestation est réalisée par l'Attestation Server auprès des Trust Agents pour les systèmes Linux (OpenStack, Docker ou KVM) et Windows, directement auprès des systèmes Xen Server et via vCenter pour les hyperviseurs ESXi.

Cette solution garantit l'intégrité des hyperviseurs en se basant sur le principe d'attestation à distance en utilisant le TPM.

[2] `https://github.com/opencit/opencit`

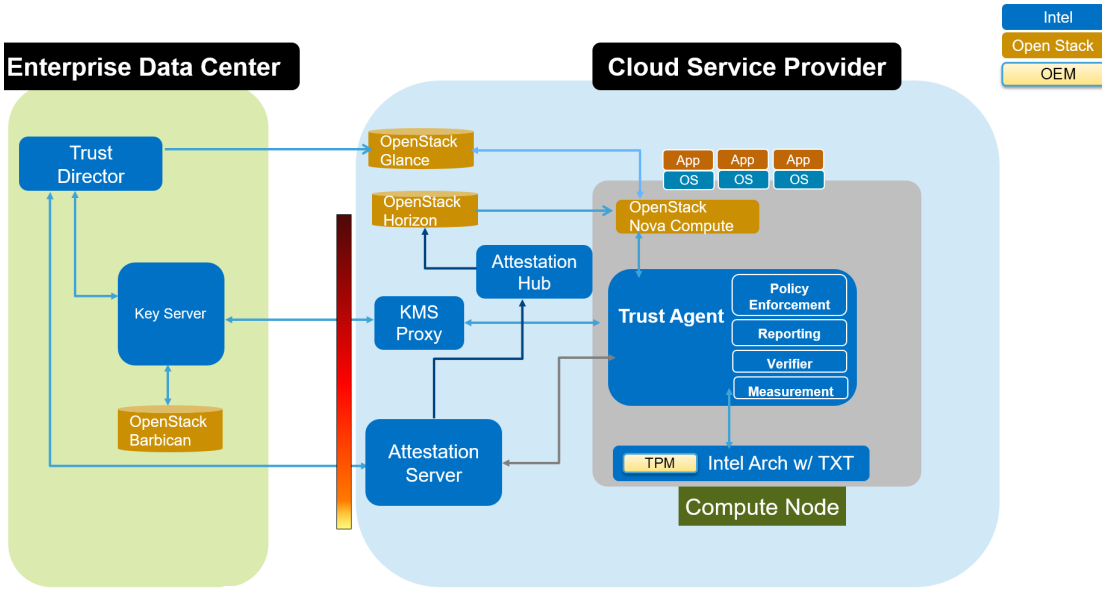


Fig. 6. Architecture d'OpenCIT [8]

Attestation L'attestation est réalisée par l'Attestation Server. Celui-ci doit être chargé avec des mesures de références (MLE - *Measured Launch Environment*) pour le BIOS et pour l'hyperviseur, qui peuvent être spécifiques à l'OEM ou à la machine physique (ex. pour les serveurs HP, le PCR[0] est différent pour chaque machine).

Ces mesures comprennent la valeur des PCR ainsi que les événements associés à ces valeurs (TcgLog).

L'Attestation Server est aussi en charge de positionner des tags sur les serveurs (localisation, clients, etc.) qui pourront être utilisés par la suite pour autoriser certaines images sur ces machines. Ces tags sont signés et insérés dans le TPM.

L'attestation est réalisée à intervalles réguliers (15 minutes par défaut) à l'initiative de l'Attestation Server qui fournit en retour une attestation SAML. La communication est réalisée avec un protocole XML sur HTTPS.

Gestion des clés Le Key Broker Service a pour rôle de vérifier les demandes de clés en vérifiant auprès du service d'attestation que la politique d'intégrité est respectée. Le serveur de gestion de clés peut être de deux types :

- un serveur KMIP ;
- un serveur OpenStack Barbican (qui peut utiliser des HSM ou des serveurs KMIP comme gestionnaires de clés).

Conclusion OpenCIT est une solution technique complète qui permet de garantir l'intégrité des hyperviseurs, des machines virtuelles, ainsi que la confidentialité des données. La solution a l'avantage d'être Open Source et de supporter plusieurs solutions techniques, matérielles et logicielles. Cependant, cette solution semble complexe à mettre en œuvre et correspond plus à de grandes infrastructures d'hébergement cloud sous OpenStack que pour un système d'information classique.

5 Shielded-VM Internals

Cette partie s'intéresse plus précisément à la solution *Shielded-VM* et *Guarded Fabric* de Microsoft, les différents composants qui la composent ainsi que ses limites en matière de sécurité.

5.1 HGS et Guarded Host

5.2 HGS

L'installation d'un serveur HGS se fait à travers du rôle serveur *Host Guardian Service*. Il est fortement recommandé d'isoler les serveurs HGS au sein d'une forêt Active Directory dédiée. En fonction du mode choisi, il peut être nécessaire de créer une relation d'approbation entre cette forêt et la forêt contenant les hyperviseurs.

Trois biclés sont nécessaires pour le fonctionnement du service :

- un biclé de signature ;
- un biclé de chiffrement ;
- un biclé pour la signature des certificats de santé.

Les serveurs HGS peuvent fonctionner en cluster et utilisent la *Cluster API* de Windows pour cela. Chaque membre du cluster doit pouvoir accéder aux biclés, et il est recommandé de les stocker dans un HSM. Les HGS utilisent un groupe de service Active Directory (gMSA) pour accéder aux certificats et l'accès au cluster est réalisé en utilisant un compte machine virtuel (VCO).

Pour le mode de fonctionnement Active Directory il est nécessaire de créer une relation d'approbation unidirectionnelle vers le domaine qui contient les serveurs Hyper-V.

Les serveurs Hyper-V doivent pouvoir joindre les serveurs HGS en HTTP (par défaut) ou HTTPS.

5.3 Guarded Host

Un Guarded Host peut être :

- un système Windows Server 2016 Datacenter avec le rôle *Hyper-V* et la fonctionnalité *Host Guardian Hyper-V Support* ;
- un système Windows 10 Enterprise 1709 (RS3) avec les fonctionnalités *Hyper-V* et *Guarded-Host*.

Avec le fonctionnement en mode TPM, la machine doit fournir au HGS :

- l'identifiant de son TPM (EKPub) ;
- une *baseline* TPM (TcgLog et PCR) ;
- la politique d'intégrité de code utilisée.

5.4 Images / Shielded Templates

Les systèmes invités[3] pouvant être protégés sont :

- Windows 8 / Server 2012 ;
- Windows 8.1 / Server 2012R2 ;
- Windows 10 / Server 2016 ;
- Ubuntu 16.04 LTS avec un noyau 4.4 ;
- RHEL 7.3 ;
- SLES 12 SP2.

Une *Shielded-VM* peut être déployée à partir d'un *Shielded Template*, image de référence pour les VM blindées. Pour les systèmes Microsoft, il s'agit d'une machine virtuelle installée, sans chiffrement puis généralisée avec `sysprep`.

Le disque de la machine est alors chiffré avec BitLocker et signé avec le certificat fourni ; la signature est stockée dans une métadonnée du disque identifiée par le GUID (`{6185B556-0C60-493A-80E37DC845FAE0E6}`)[4]

Lors de son déploiement, la machine sera personnalisée avec un fichier de réponse (*unattend.xml*), qui peut contenir des informations sensibles (mots de passe, clés de chiffrement, etc.). Ce fichier est lié au gabarit de la machine et chiffré avec les clés du propriétaire et des gardiens autorisés à instancier une machine virtuelle.

[3] Les machines Linux ne peuvent être manipulées qu'avec des OS 1709.
[4] `PtpTemplateNative.dll` / `SvmRetrieveSignatureCatalogBlobFromDisk`

La protection des machines Linux est réalisée par *lsvmtools*[5] qui permet de mettre en place une chaîne de démarrage sécurisée ainsi que le chiffrement des données à partir du TPM 2.0.

Par ailleurs, les machines virtuelles Windows de seconde génération peuvent être directement converties en *Shielded-VM*.

Note : il n'est pas possible de convertir une *Shielded-VM* en machine virtuelle non protégée.

5.5 Remote Attestation

Le protocole de communication entre un Guarded Host et le HGS est partiellement décrit dans les spécifications ouvertes de Microsoft [4].

La communication est réalisée en HTTP ou HTTPS depuis le Guarded Host vers le HGS avec une interface REST.

L'objectif de l'attestation à distance est de :

- vérifier que la machine est « saine » ;
- obtenir la clé publique du VTL1 (VsmIdk) ;
- signer cette clé pour former le certificat de santé de la machine.

Mode TPM Dans ce mode, le Guarded Host présente la clé *EKPub* de son TPM au serveur ce qui permet de vérifier que celle-ci a bien été enregistrée auprès du HGS.

La clé EKPub doit être fournie au HGS lors d'une phase d'enrôlement afin d'identifier la machine.

Le HGS déclenche ensuite le processus d'attestation à distance :

- le TPM de la machine et le HGS établissent une *Salted Session* :
  - le HGS envoie un nonce (nonceCaller) et un sel chiffré avec la clé *EKPub* du TPM (encryptedSalt) ;
  - le TPM retourne un nonce (nonceTPM) et un *handle* pour cette session.
- le HGS demande la valeur des PCR choisis :
  - PCR[7] : État du Secure Boot (PK, KEK, db/dbx, SecureBoot variable) ;
  - PCR[12] : Événements volatiles (compteurs, hash et bases des modules, VsmIdk) ;
  - PCR[13] : Détails des modules (Signature des modules, SiPolicy, Etat du VBS) ;

[5] `https://github.com/Microsoft/lsvmtools`

– le host renvoie :
  - les valeurs des PCR demandés ;
  - le TcgLog ;
– le HGS vérifie l'intégrité des PCR vis-à-vis du TcgLog ;
– le HGS vérifie les paramètres dans le TcgLog et en extrait la clé publique du VSM.

Le HGS délivre l'attestation de santé si l'ensemble des vérifications réussit. Le protocole d'attestation à distance est implanté dans le composant *rtpm.dll* (identique côté serveur et client). Le protocole utilise les structures standard définies dans la spécification du TrustedComputingGroup.

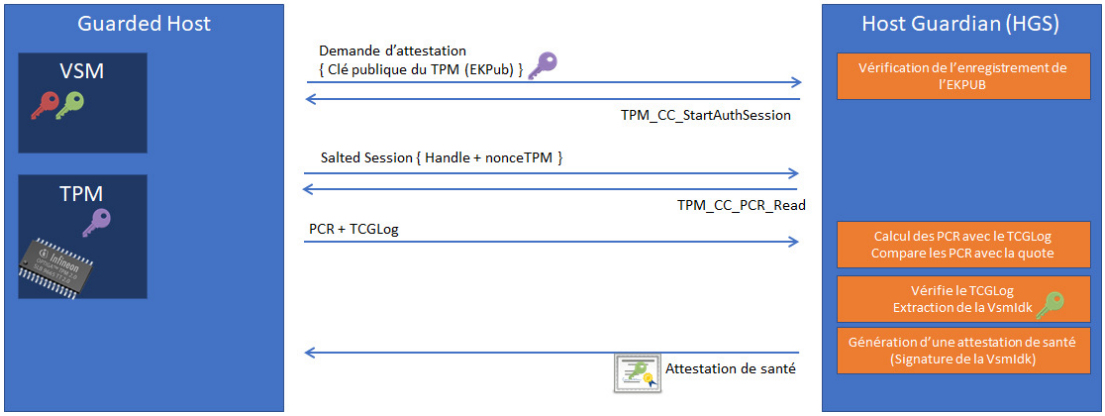


Fig. 7. Attestation à distance TPM

```
# Host -> HGS
# Demande d'attestation - envoi de l'EKPub
{
    "__type":"TpmRequestInitial:#Microsoft.Windows.RemoteAttestation
        .Core",
    "SessionId":"{6A866F8E-4BD8-0002-AD72-866AD84BD301}",
    "RequestedContent":[1],
    "RtpmPublicEndorsementKey":[221,21,148,...,64,23,50,37]
}
# Host <- HGS
# Demarrage de la salted session (nonceCaller + encryptedSalt)
{
    "__type":"TpmReplyContinue:#Microsoft.Windows.RemoteAttestation.
        Core",
    "RtpmActiveContext":[239,6,0,0,1,...,6,0,128,0,67,0,11]
}
# Host -> HGS
# Établissement de la salted session (nonceTPM + handle)
```

```
{
    "__type":"TpmRequestContinue:#Microsoft.Windows.
        RemoteAttestation.Core",
    "SessionId":"{6A866F8E-4BD8-0002-AD72-866AD84BD301}",
    "RequestedContent":[1],
    "RtpmPublicEndorsementKey":[221,21,148,...,64,23,50,37],
    "RtpmNewContext":[224,5,0,0,1,0,...,185,232,55]
}
# Host <- HGS
# Demande des PCR
{
    "__type":"TpmReplyContinue:#Microsoft.Windows.RemoteAttestation.
        Core",
    "RtpmActiveContext":[31,6,0,0,1...,0,0,0]
}
# Host -> HGS
# Reponse avec les PCR + TcgLog
{
    "__type":"TpmRequestContinue:#Microsoft.Windows.
        RemoteAttestation.Core",
    "SessionId":"{6A866F8E-4BD8-0002-AD72-866AD84BD301}",
    "RequestedContent":[1],
    "RtpmPublicEndorsementKey":[221,21,148,...,64,23,50,37],
    "RtpmNewContext":[144,195,0,0,1,0,...,99,247,144]
}
# Host <- HGS
# Obtention d'attestation de sante
{
    "__type":"HealthCertificateReply:#Microsoft.Windows.
        RemoteAttestation.Core",
    "Content":[{
        "m_Item1":1,
        "m_Item2":[48,130,3,233,...,253,203,62,128]
    }]
}
```

Listing 1. Échanges entre l'hyperviseur et le HGS pour l'attestation en mode TPM

Les structures `RtpmActiveContext` et `RtpmNewContext` se composent de la manière suivante :

L'objet `EncryptedVTPMState` contient l'état complet du TPM virtuel utilisé par le HGS pour réaliser ses vérifications. Il est présent dans chaque requête pour permettre le fonctionnement des HGS en cluster et sans état.

Le fonctionnement de ce protocole d'attestation est surprenant pour plusieurs raisons :

- La clé publique *EKPub* est utilisée pour chiffrer lors de l'établissement de la session ; ce qui n'est pas l'usage normal de cette clé qui est de prouver l'identité du TPM (via des signatures) ;
- Il existe des primitives d'attestation comme `TPM2_Quote` qui permettent de signer l'état des PCR avec une clé du TPM.

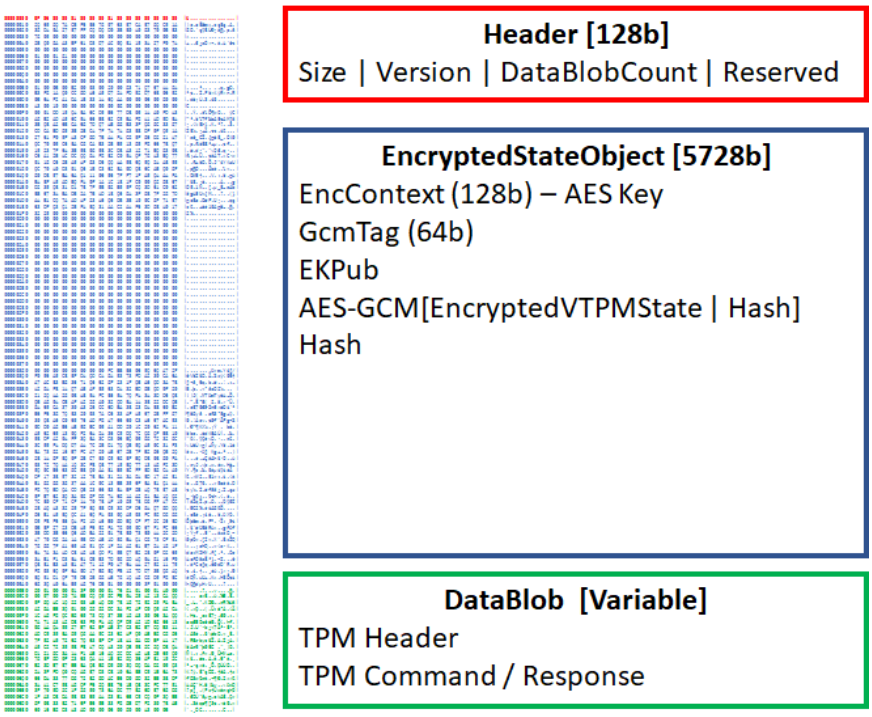


Fig. 8. TPM Context

Ces éléments ne remettent toutefois pas en question la sécurité de l'attestation à distance.

Mode AD Dans ce mode, le client s'identifie auprès du HGS en réalisant une authentification web (NTLM ou Kerberos) ; le serveur vérifie ainsi l'appartenance aux groupes autorisés puis délivre le certificat de santé.

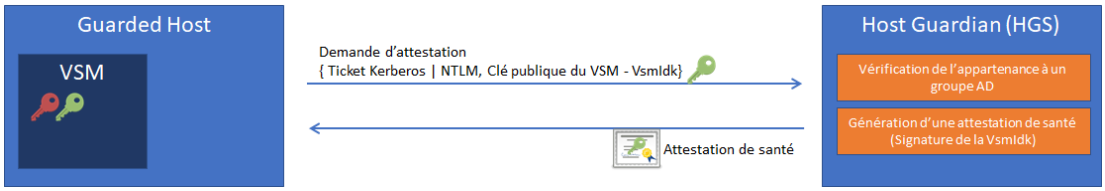


Fig. 9. Attestation Active Directory

Le client tente toujours d'utiliser le mode TPM en premier, et passe sur le mode AD lorsque le serveur rejette les requêtes en mode TPM. Le client envoie la clé publique du VSM et s'authentifie lorsque le serveur le demande (HTTP 401). L'authentification est réalisée en `Negotiate` ou `NTLM`.

Les échanges entre l'hyperviseur prennent la forme suivante :

```
# Host -> HGS
# Demande d'attestation - envoi de l'EKPub
{
    "__type":"TpmRequestInitial:#Microsoft.Windows.RemoteAttestation
        .Core",
    "SessionId":"{6A866F8E-4BD8-0002-AD72-866AD84BD301}",
    "RequestedContent":[1],
    "RtpmPublicEndorsementKey":[221,21,148,...,64,23,50,37]
}
# Host <- HGS
# Refus du mode TPM, passage en mode ActiveDirectory
{
    "__type":"OperationModeErrorReply:#Microsoft.Windows.
        RemoteAttestation.Core",
  "Retryable":false,
  "ExpectedOperationMode":2
}
# Host -> HGS
# Envoi de la VsmIdk
{
    "__type":"ADRequest:#Microsoft.Windows.RemoteAttestation.Core",
    "SessionId":"{6A866F8E-4BD8-0002-AD72-866AD84BD301}",
    "RequestedContent":[{
        "m_Item1":1,
        "m_Item2":[82,83,65,49,0,...,121,23,232,7]
    }]
}
# Host <- HGS
# Refus du serveur pour declencher l'authentification
HTTP/101 401 Unauthorized
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM

# Host -> HGS
Authorization: Negotiate TlR...AAAADw==
{
    "__type":"ADRequest:#Microsoft.Windows.RemoteAttestation.Core",
    "SessionId":"{6A866F8E-4BD8-0002-AD72-866AD84BD301}",
    "RequestedContent":[{
        "m_Item1":1,
        "m_Item2":[82,83,65,49,0,...,121,23,232,7]
    }]
}
# Host <- HGS
# Negotiation NTLMSSP
HTTP/101 401 Unauthorized
WWW-Authenticate: Negotiate TlR...AAAA==

# Host -> HGS
# Negotiation NTLMSSP
Authorization: Negotiate TlR...8HBo=
{
    "__type":"ADRequest:#Microsoft.Windows.RemoteAttestation.Core",
    "SessionId":"{6A866F8E-4BD8-0002-AD72-866AD84BD301}",
    "RequestedContent":[{
```

```
        "m_Item1":1,
        "m_Item2":[82,83,65,49,0,...,121,23,232,7]
    }]
}
# Host <- HGS
# Obtention d'attestation de sante
{
    "__type":"HealthCertificateReply:#Microsoft.Windows.
        RemoteAttestation.Core",
    "Content":[{
        "m_Item1":1,
        "m_Item2":[48,130,3,233,...,253,203,62,128]
    }]
}
```

Listing 2. Échanges entre l'hyperviseur et le HGS pour l'attestation en mode AD

5.6 Démarrage d'une machine blindée

Le processus de démarrage d'une machine virtuelle blindée est le suivant (voir figure 10) :

- Le service `vmcompute.exe` démarre un processus `vmwp.exe` (worker process) en PPL (*Protected Process Light*) avec un SID de machine virtuelle unique (`S-1-5-83-1-X-Y-Z`) ①
- `vmwp.exe` lance une instance du trustlet `vmsp.exe` (en VTL1), en charge de fournir un TPM virtuel à la machine virtuelle, avec deux arguments : le GUID de la machine virtuelle et une chaîne aléatoire, utilisée comme nom de l'événement de synchronisation et comme clé pour les Mailbox ②
- `vpsp.exe` initialise une terminaison RPC avec un nom aléatoire qu'il place dans la Mailbox SKM 0 et signale l'événement de synchronisation ③
- `vmwp.exe` récupère le nom de la terminaison RPC dans la Mailbox SKM 0 à travers `vid.dll` et `vid.sys` ④
- `vmwp.exe` se connecte à la terminaison RPC ⑤
- `vmwp.exe` initialise le TPM virtuel ⑥

Les composants `vid.dll` et `vid.sys` (*Hyper-V Virtualization Infrastructure Driver Library*) sont utilisés pour gérer les machines virtuelles et les interactions bas niveau. Parmi les méthodes disponibles, deux sont spécifiques à l'interaction avec `vmsp.exe` :

- `VidSetMailboxKey` : permet de fournir la clé des Mailbox du trustlet qui sera passée à la fonction `VslRetrieveMailbox` du noyau ;

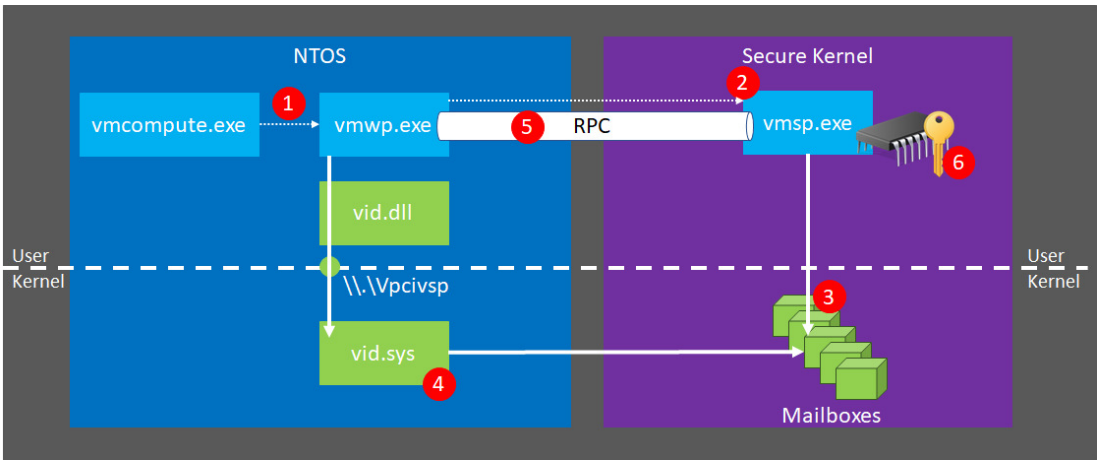


Fig. 10. Composants utilisés pour le démarrage d'une machine virtuelle blindée

– `VidGetRpcSession` : récupère la Mailbox 0 du trustlet.

La terminaison RPC expose deux interfaces :

– `VmspKeyManagement`
  - `RpcVmspOpenHandle`
  - `RpcVmspCloseHandle`
  - `RpcKmSetEncryptionKeys`
  - `RpcKmEgressKeyForDecryption`
– `VmspTpm`
  - `RpcVmspOpenSecureHandle`
  - `RpcVmspCloseSecureHandle`
  - `RpcVTpmInitialize`
  - `RpcVTpmShutdown`
  - `RpcVTpmExecuteCommand`
  - `RpcVTpmGetRuntimeSize`
  - `RpcVTpmGetRuntimeState`
  - `RpcVTpmSetCancelFlag`
  - `tpm12class::TpmDataObject::Deserialize(void)`
  - `RpcVTpmCreateReport`
  - `RpcVmspFreeContext`

La protection des machines virtuelles blindées éteintes repose sur la protection des données du TPM virtuel de la machine, qui sont stockées dans le fichier VMRS (*Virtual Machine Runtime State*) de la machine, chiffrées avec une *TransportKey*. Cette *TransportKey* est elle-même chiffrée par la clé publique du propriétaire ainsi que par la clé publique du HGS ; ces informations sont stockées dans un *KeyProtector* dans le fichier VMRS [7].

Lors du démarrage de la machine virtuelle, les opérations suivantes sont réalisées après l'établissement du canal RPC :

- l'hyperviseur transmet au HGS le *KeyProtector* et son certificat de santé ;
- le HGS vérifie la validité du certificat de santé (expiration, signature, conformité à la politique de santé) ;
- le HGS extrait les éléments du *KeyProtector* et calcule la réponse :
  - extraction de la *TransportKey* (TK1) avec la clé privée du HGS ;
  - génération d'une nouvelle *TransportKey* (TK2) ;
  - chiffrement de TK2 avec la clé publique du propriétaire et la clé publique du HGS ;
  - chiffrement de TK1 et TK2 avec la clé publique du VSM (extraite depuis le certificat de santé).

Les clés TK1 et TK2 sont ensuite transmises et déchiffrées par `vmsp.exe` qui peut instancier le vTPM avec TK1. Lors de l'arrêt de la machine, l'état du vTPM est chiffré avec TK2.

La machine est ensuite démarrée normalement, une partition Hyper-V est créée, la mémoire virtuelle est allouée dans le processus `vmmem`, les périphériques initialisés puis le processeur virtuel est démarré.

5.7 Limites du blindage

L'objectif présenté des machines virtuelles blindées est de protéger les machines virtuelles contre les administrateurs du socle de virtualisation. Ceux-ci ne doivent pas pouvoir acquérir des droits ou des informations sur les machines virtuelles exécutées.

Quels sont les risques ?

1. L'accès à la mémoire physique de la machine permet d'accéder au secret du VTPM, à la mémoire de la machine virtuelle (périphériques en DMA, iLO, iDRAC, etc.) ;
2. L'accès à la mémoire du VTL0 - où la mémoire de la machine virtuelle réside (processus `vmmem`) ;
3. La modification des composants logiciels du VTL0 pour lever les limitations d'accès.

Les risques 2. et 3. sont traités par la politique d'intégrité de code (*CiPolicy*). Celle-ci doit être la plus stricte possible pour n'autoriser que ce qui est essentiel au système. Ces politiques permettent de décrire finement quels drivers, exécutables, bibliothèques ou scripts peuvent s'exécuter sur

la machine. Cependant des *bypass* sont régulièrement découverts par des chercheurs en sécurité [3, 5].

Dès qu'un driver non maîtrisé peut être chargé en VTL0, les machines blindées sont mises en danger, car leur mémoire est accessible depuis le VTL0 en mode noyau ; avec notamment la clé de chiffrement Bitlocker ou les données d'authentification.

5.8 Utilisations possibles

Parmi les utilisations possibles de cette solution, Microsoft propose :

- la réalisation de stations d'administration (*PAW – Privileged Access Workstation*) où l'utilisateur dispose de plusieurs machines (bureautique, administration tier 1, administration tier 0) dont les plus sensibles sont blindées, permettant ainsi de disposer de postes nomades d'administration en préservant l'intégrité et la confidentialité des données ;
- le déploiement de systèmes sensibles comme des contrôleurs de domaine ou sur des sites exposés (agences, filiales, sites de repli) en protégeant l'hyperviseur avec des mécanismes d'intégrité supplémentaires comme le chiffrement BitLocker.

6 Conclusion

Les machines blindées ne répondent pas entièrement à la problématique de protection dans des environnements mal maîtrisés comme le cloud ou les hébergements mutualisés.

Elles représentent toutefois un intérêt certain pour le déploiement de systèmes sensibles ou pour des stations d'administration.

On note toutefois une volonté des éditeurs de solutions de travailler sur ces problématiques qui devraient évoluer dans le bon sens dans les prochaines années.

Références

1. Frédéric Guihéry, Frédéric Remi, Goulven Guiheux. Trusted Computing : Limitations actuelles et perspectives. In *SSTIC*, 2010.
2. ANSSI. Mise en œuvre des fonctionnalités de sécurité de Windows 10 reposant sur la virtualisation. `https://www.ssi.gouv.fr/guide/mise-en-oeuvre-des-fonctionnalites-de-securite-de-windows-10-reposant-sur-la-virtualisation/`.

3. Matt Nelson. UMCI Bypass Using PSWorkFlowUtility : CVE-2017–0215. https://posts.specterops.io/umci-bypass-using-psworkflowutility-cve-2017-0215-71c76c1588f9.
4. Microsoft. Host Guardian Service : Attestation Protocol. https://msdn.microsoft.com/en-us/library/mt781332.aspx.
5. Microsoft. Steps to Deploy Windows Defender Application Control. https://docs.microsoft.com/en-us/windows/security/threat-protection/device-guard/steps-to-deploy-windows-defender-application-control.
6. howpublished = https://docs.microsoft.com/en-us/windows/security/hardware-protection/secure-the-windows-10-boot-process Microsoft, title = Secure the Windows 10 boot process.
7. M.F. Novak, N. Ben-Zvi, and N.T. Ferguson. Secure transport of encrypted virtual machines with continuous owner access, November 12 2015. WO Patent App. PCT/US2015/028,991.
8. Open CIT. Open CIT 3.2.1 Product Guide. https://github.com/opencit/opencit/wiki/Open-CIT-3.2.1-Product-Guide.
9. VMware. Encryption Process Flow. https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.security.doc/GUID-4A8FA061-0F20-4338-914A-2B7A57051495.html.
10. VMware. UEFI Secure Boot for ESXi Hosts. https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.security.doc/GUID-5D5EE0D1-2596-43D7-95C8-0B29733191D9.html.