

MISE EN ŒUVRE DES FONCTIONNALITÉS DE SÉCURITÉ DE WINDOWS 10 REPOSANT SUR LA VIRTUALISATION

GUIDE ANSSI

PUBLIC VISÉ :

Développeur

Administrateur

RSSI

DSI

Utilisateur



ANSSI-BP-039

08/11/2017

Informations



Attention

Ce document rédigé par l'ANSSI présente les « **Mise en œuvre des fonctionnalités de sécurité de Windows 10 reposant sur la virtualisation** ». Il est téléchargeable sur le site www.ssi.gouv.fr. Il constitue une production originale de l'ANSSI. Il est à ce titre placé sous le régime de la « Licence ouverte » publiée par la mission Etalab (www.etalab.gouv.fr). Il est par conséquent diffusable sans restriction.

Ces recommandations n'ont pas de caractère normatif, elles sont livrées en l'état et adaptées aux menaces au jour de leur publication. Au regard de la diversité des systèmes d'information, l'ANSSI ne peut garantir que ces informations puissent être reprises sans adaptation sur les systèmes d'information cibles. Dans tous les cas, la pertinence de l'implémentation des éléments proposés par l'ANSSI doit être soumise, au préalable, à la validation de l'administrateur du système et/ou des personnes en charge de la sécurité des systèmes d'information.

Évolutions du document :

VERSION	DATE	NATURE DES MODIFICATIONS
1.0	08/11/2017	Version initiale du document

Table des matières

1	Introduction	4
2	Sécurité reposant sur la virtualisation	6
2.1	Caractéristiques et fonctionnement du <i>Virtual Secure Mode</i>	6
2.2	Prérequis des équipements	7
2.3	Limites de sécurité	8
2.4	Activation du démarrage sécurisé UEFI	9
2.5	Activation de la sécurité reposant sur la virtualisation	11
2.6	Incompatibilité avec les solutions de virtualisation tierces	12
3	Device Guard et l'intégrité du code	13
3.1	Caractéristiques générales de l'intégrité du code	13
3.2	Architecture des mécanismes d'intégrité du code et apports de <i>Device Guard</i>	13
3.3	Différences entre <i>Device Guard</i> et Applocker	14
3.4	Mise en œuvre de <i>Device Guard</i>	15
3.4.1	Stratégie d'intégrité du code configurable (CCI)	15
3.4.2	Intégrité du code protégé par l'hyperviseur (HVCI)	16
3.5	Points d'attention à prendre en compte avant toute mise en œuvre de <i>Device Guard</i> .	17
4	Credential Guard	20
4.1	Principes techniques et apports en matière de sécurité	20
4.2	Périmètre de mise en œuvre	21
4.3	Limites de <i>Credential Guard</i>	22
4.4	Mise en œuvre de <i>Credential Guard</i>	24
Annexe A	Prérequis matériels et logiciels	25
A.1	Prérequis	25
A.2	Certification du matériel	27
A.3	Script de diagnostic	28
A.4	Adaptation du script de diagnostic à un système d'exploitation en langue française .	28
A.4.1	Test de la version de Windows	28
A.4.2	Test de l'architecture	29
Annexe B	Installation des fonctionnalités Windows requises	30
B.1	Installation des fonctionnalités Windows nécessaires au VSM	30
B.2	Script PowerShell exécuté par GPO	30
Annexe C	Méthodes d'identification des postes de travail cibles	33
C.1	Cibles de la GPO d'installation du VSM	33
C.2	Cibles des GPO de mise en œuvre de <i>Device Guard</i> et de <i>Credential Guard</i>	35
C.2.1	Construction d'une classe WMI personnalisée	35
C.2.2	Copie de la class WMI Win32_DeviceGuard en base de registres	38
Annexe D	Désactivation après verrouillage UEFI	40
D.1	Suppression du verrouillage UEFI	40
D.2	Validation de la désactivation par intervention physique	40

Annexe E La classe WMI Win32_DeviceGuard	42
Annexe F GPO de mise en œuvre des fonctionnalités de VBS	43
F.1 Téléchargement des modèles d'administration de Windows 10	43
F.2 GPO d'activation de la VBS seule	43
F.3 GPO d'activation de <i>Credential Guard</i>	44
F.4 GPO d'activation de <i>Device Guard</i>	44
Annexe G Contrôle de conformité de l'IGC UEFI	45
G.1 Particularités de la Platform Key (PK)	45
G.2 Comparaison à une base de référence	46
G.2.1 Clés de l'IGC UEFI	46
G.2.2 Extraction des clés de l'IGC UEFI	46
G.3 Limites d'un contrôle de conformité de l'IGC UEFI	47
Liste des recommandations	48
Bibliographie	49

1

Introduction

Les technologies dites de « virtualisation » permettent d'opérer un cloisonnement entre des domaines fonctionnels différents, indépendamment des multiples couches physiques et logiques sous-jacentes. Pour cela, elles s'appuient sur des mécanismes logiciels ou des technologies embarquées au niveau du matériel. Plusieurs systèmes d'exploitation peuvent ainsi être virtualisés, c'est-à-dire exécutés sur une seule machine physique tout en étant isolés les uns des autres. La virtualisation a ainsi connu un essor important ces dernières années, lié notamment au développement de nouveaux usages comme l'informatique en nuage (*cloud computing*) mais également à la flexibilité qu'elle apporte à l'administration des systèmes d'information.

Les technologies de virtualisation ne sont toutefois pas l'apanage de l'informatique en nuage ou des infrastructures des systèmes d'information. Elles peuvent également être utilisées sur des postes de travail afin d'y exécuter différents environnements cloisonnés entre eux. Il s'agit d'un usage notamment illustré dans le guide d'administration sécurisée des systèmes d'information [6]. Des solutions de sécurité peuvent également tirer profit des technologies de virtualisation pour isoler certains processus, qu'ils soient sensibles et critiques ou, à l'inverse, dangereux voire assurément malveillants.

Sous Windows 10, Microsoft met à profit les technologies de virtualisation à travers des fonctionnalités de sécurité reposant sur la virtualisation (VBS, *Virtualization Based Security*¹) et destinée aux éditions « Entreprise » et « Éducation ». La VBS s'appuie elle-même sur le *Virtual Secure Mode* (VSM) de Microsoft, qui est un environnement virtualisé permettant de cloisonner, dans une machine virtuelle spécifique, des processus critiques pour la sécurité.

À la date de publication de ce guide, Microsoft a regroupé les fonctionnalités de sécurité reposant sur la VBS en deux ensembles : *Device Guard* (DG) et *Credential Guard* (CG) qui ont tous deux un objectif général d'isolation de certains processus du système d'exploitation. Ils sont décrits dans ce document de manière à expliquer leurs apports et leurs limites en matière de sécurité. Les modalités de leur mise en œuvre sur des postes de travail exécutant le système d'exploitation Windows 10 de Microsoft dans un environnement *Active Directory* (AD) sont également détaillées. Au préalable, une présentation de la VBS et du VSM est toutefois indispensable à la compréhension du fonctionnement des mécanismes de sécurité mis en œuvre et leurs prérequis.

Windows 10 est par ailleurs amené à fortement évoluer au fur et à mesure des mises à jour du système, apportant de nouvelles fonctionnalités (dont certaines reposent sur la virtualisation) et de nouveaux paramétrages à effectuer. Contrairement à ses prédécesseurs qui faisaient l'objet de mises à jour majeures via des *Service Packs* facilement identifiables, Windows 10 présente des évolutions via des mises à jour mineures et majeures. Le niveau de version de Windows 10 n'est donc

1. La fonctionnalité appelée « *Virtualization Based Security* » (VBS) a été traduite en « Sécurité basée sur la virtualisation » par Microsoft tant au niveau du système d'exploitation que de leur documentation. Dans ce guide, la VBS est désignée sous le terme de « Sécurité reposant sur la virtualisation ».

plus représenté par un *Service Pack* mais par un numéro de version ou par son nom de code associé. L'historique des versions publiques à ce jour est :

- la version 1507 datée du 29 juillet 2015 (nom de code *Threshold 1*);
- la version 1511 datée du 12 novembre 2015 (nom de code *Threshold 2*);
- la version anniversaire 1607 datée du 2 août 2016 (nom de code *Redstone 1*);
- la version *Creators Update* 1703 datée du 11 avril 2017 (nom de code *Redstone 2*).

Le présent document s'appuie sur la version 1703 de Windows 10 (nom de code *Redstone 2*) pour postes de travail, en éditions « Entreprise » ou « Éducation ».

Il est à noter que les fonctionnalités de sécurité de *Device Guard* et *Credential Guard* sont également disponibles sur Windows Server 2016. Bien qu'elles présentent un intérêt certain dans une démarche de durcissement, leur mise en œuvre sur des systèmes serveurs est toutefois hors périmètre du présent document.

Enfin, pour un même sujet, plusieurs recommandations peuvent être proposées dans le document. Elles se distinguent par le niveau de sécurité atteint. Elles doivent permettre au lecteur de retenir les recommandations offrant la meilleure protection au regard du contexte et des étapes nécessaires à leur mise en œuvre. Ainsi, les recommandations seront présentées de la manière suivante :

Rx	Recommandation générale et synthétique, faisant abstraction du contexte d'application.
Rx *	Cette implémentation technique de la recommandation Rx offre un niveau de protection adapté à un besoin de sécurité standard.
Rx **	Cette implémentation technique de la recommandation Rx offre un niveau de protection adapté à un besoin de sécurité élevé.

Tableau 1.1 – Priorisation des recommandations

2

Sécurité reposant sur la virtualisation

2.1 Caractéristiques et fonctionnement du Virtual Secure Mode

Le *Virtual Secure Mode* (VSM) tire partie des jeux d'instructions processeurs spécifiques à la virtualisation, connus principalement sous les termes *Intel-VT* chez Intel et *AMD-V* chez AMD. Les premiers processeurs dotés de ces extensions de virtualisation sont apparus sur le marché vers fin 2005.

Ces jeux d'instructions permettent d'exécuter un hyperviseur comme système hôte, c'est-à-dire une plateforme de virtualisation dont le rôle est, dans le cas présent, d'isoler l'exécution du système invité principal (c'est-à-dire le système d'exploitation traditionnel avec lequel l'utilisateur interagit) du système invité secondaire (le VSM). Ces deux environnements virtualisés ont ainsi leur propre noyau et sont isolés l'un de l'autre grâce aux technologies de virtualisation apportées par le matériel (et notamment le SLAT, *Second Level Address Translation*, technique qui repose sur le support matériel de la virtualisation pour traduire les adresses physiques telles que vues par les systèmes invités en adresses physiques, telles que vues par le système hôte).

Le code de l'hyperviseur est bien plus petit que celui du système d'exploitation traditionnel. L'hyperviseur étant chargé avant les systèmes invités, il s'exécute à un niveau de privilèges supérieur et contrôle les ressources système clés. Il ne comprend par ailleurs aucun pilote et aucun code ne peut y être ajouté, sa surface d'attaque se trouve ainsi réduite et son intégrité peut être vérifiée lors du démarrage sécurisé (voir section 2.4).

Le VSM, à la différence du système d'exploitation traditionnel, est doté d'un micro-noyau léger (le SKM, *Secure Kernel Mode*) qui intègre ses propres pilotes signés ainsi qu'un environnement d'exécution de processus isolé (l'IUM, *Isolated User Mode*), rien d'autre ne peut y être ajouté. Les processus qui s'y exécutent (appelés *Trustlets*) sont uniquement édités par Microsoft et font partie intégrante du système d'exploitation. Ce sont par ailleurs des composants logiciels minimaux, qui ont un rôle simple et précis et qui appliquent des techniques de durcissement logiciel. Ils sont donc considérés comme des processus de confiance, et chaque *Trustlet* est protégé en intégrité par une signature numérique². Le VSM se trouve ainsi allégé au maximum et présente une surface d'attaque réduite qui lui permet de répondre au besoin d'isolation des processus critiques du système d'exploitation.

2. Les *Trustlets* sont signés par Microsoft avec un certificat qui contient *Isolated User Mode* (1.3.6.1.4.311.10.3.37) comme *Extended Key Usage* (EKU).

Les *Trustlets* sont ces processus de confiance qui vont porter les fonctionnalités telles que *Credential Guard* en VSM. En cas de compromission du système d'exploitation Windows, la sécurité des *Trustlets* du VSM reste ainsi assurée et les services rendus par ces derniers restent protégés en intégrité et en confidentialité. Les processus de l'environnement Windows standard peuvent communiquer avec ces *Trustlets* par RPC. En revanche les noyaux du VSM et de l'environnement Windows standard ne peuvent pas communiquer directement entre eux. Une représentation simplifiée de cette architecture est donnée dans la figure 2.1.

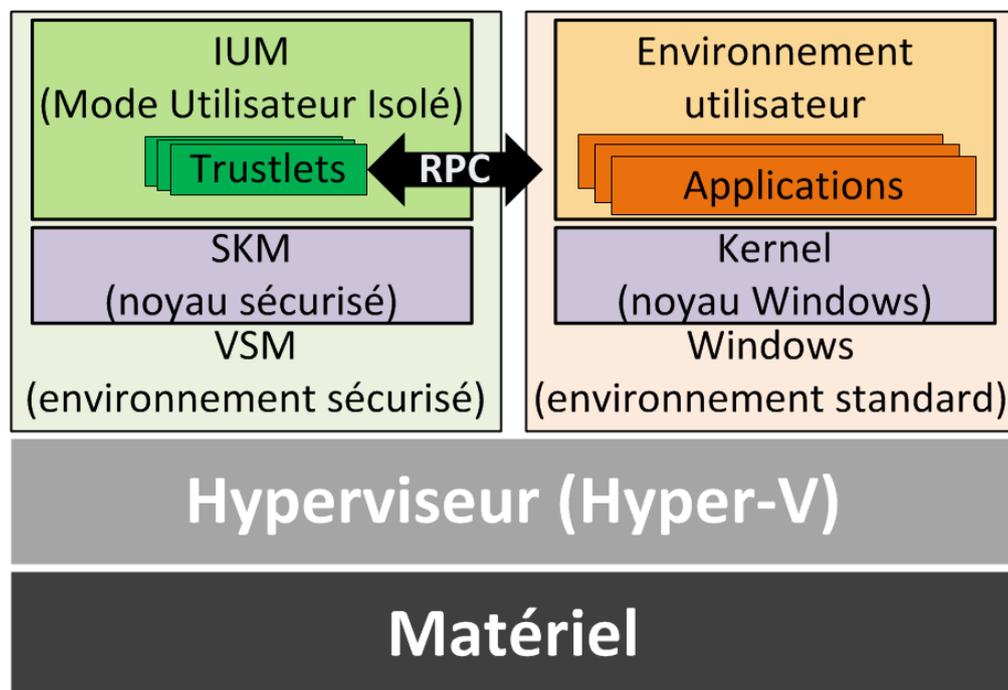


FIGURE 2.1 – Représentation simplifiée de l'architecture du *Virtual Secure Mode*

2.2 Prérequis des équipements

Mettre en œuvre le VSM sur les postes de travail requiert du matériel supportant la virtualisation. L'intérêt du VSM réside principalement dans la mise en œuvre des fonctionnalités de sécurité reposant sur la virtualisation (VBS), et ces dernières ont quelques prérequis matériels et logiciels. À la date de publication de ce guide, rares sont toutefois les systèmes d'information dont l'ensemble des équipements satisfont ces prérequis. Il n'est donc généralement pas possible de déployer ces fonctionnalités de VBS à l'échelle d'un système d'informations (SI) complet. L'[annexe A](#) énumère les prérequis nécessaires à la mise en œuvre de ces fonctionnalités et présente également un script de test de compatibilité des matériels.

Dans une démarche de défense en profondeur, la mise en œuvre des fonctionnalités de VBS est intéressante sur tout poste de travail Windows 10 compatible. En fonction de la maturité du SI en matière de sécurité des systèmes d'information (SSI), le déploiement de ces fonctionnalités présentera néanmoins plus ou moins d'intérêt en fonction du niveau de criticité des différentes catégories de postes de travail. Cet aspect sera abordé plus spécifiquement dans les sections 3 et 4 respectivement relatives aux fonctionnalités *Device Guard* et *Credential Guard*.

R1

Intégrer la compatibilité VSM/VBS dans le processus d'achat de matériel

Les processus d'achat de matériel doivent dès maintenant intégrer les spécifications minimales (définies en [annexe A](#)) nécessaires à la mise en œuvre, à court ou moyen terme, des fonctionnalités de sécurité reposant sur la virtualisation. Certains prérequis matériels optionnels, tels que l'IOMMU pour les protections DMA³, sont par ailleurs recommandés pour améliorer le niveau de sécurité atteint.

i

Information

Si le support de certains éléments de configuration peut souvent être vérifié à partir des spécifications détaillées et certifications matérielles de l'équipement, la vérification de certaines caractéristiques pourra en revanche nécessiter soit des échanges techniques avec les équipes commerciales et les experts produits des constructeurs, soit des tests opérés directement sur le matériel (tel qu'indiqué dans l'[annexe A](#)).

Lorsque les acquisitions de matériel se font sur des volumes importants donnant lieu à des appels d'offre, il est dans ce cas recommandé d'y intégrer des exigences relatives à ces prérequis.

2.3 Limites de sécurité

Si la chaîne de démarrage⁴ de l'équipement est compromise (depuis le système d'exploitation en exploitant des vulnérabilités UEFI, ou bien par des attaques physiques directement sur le poste de travail⁵), l'hyperviseur peut également être compromis ainsi que tout ce qu'il exécute (environnement sécurisé et environnement standard). Disposer d'équipements qui satisfont les prérequis techniques à la mise en œuvre du VSM et des fonctionnalités de VBS n'est toutefois pas suffisant et leur mise en œuvre doit également s'accompagner de mesures de sécurité concernant l'UEFI.

R2

Protéger les accès à l'UEFI

L'accès à l'UEFI doit être protégé par mot de passe ou toute autre méthode d'authentification plus robuste. Les secrets d'authentification ne doivent être connus que des administrateurs autorisés à modifier la configuration des chaînes de démarrage des postes de travail. Un utilisateur ne doit donc pouvoir ni modifier la configuration de l'UEFI ni l'ordre de démarrage.

R3

Maintenir à jour les composants logiciels UEFI

Il est recommandé de maintenir à jour les composants logiciels UEFI (micrologiciels, pilotes, etc.) des postes de travail afin de disposer des derniers correctifs et mises à jour de sécurité publiés par les éditeurs et constructeurs.

3. Le DMA (*Direct Memory Access*) consiste en un accès direct à la mémoire vive par des périphériques sans passer par le processeur. Une attaque DMA consiste donc à accéder de cette manière à des espaces mémoire non autorisés afin de lire ou altérer leur contenu. L'IOMMU, si le matériel en est pourvu (se référer aux prérequis matériels en [annexe A](#)), permet au système de se protéger contre ces attaques.

4. C'est-à-dire l'ensemble des composants logiciels successivement exécutés jusqu'au démarrage du système d'exploitation.

5. Pour plus d'informations sur les faiblesses de l'UEFI, se reporter à l'article [7].



Information

Comme évoqué dans l'annexe A, certains UEFI permettent leur mise à jour sécurisée via *Windows Update*. La mise à jour de ces derniers devrait donc pouvoir être gérée relativement simplement. En revanche, certains équipements nécessiteront des procédures spécifiques en fonction des constructeurs et des modèles de matériel. La problématique de maintien en conditions de sécurité (MCS) des chaînes de démarrage et composants logiciels UEFI n'est pas l'objet du présent document.

La section 2.4 suivante traite des mécanismes de vérification de l'intégrité du boot UEFI dans une optique de protection de la chaîne de démarrage.

2.4 Activation du démarrage sécurisé UEFI

Le démarrage sécurisé UEFI, plus communément appelé *Secure Boot*, est une fonctionnalité sur laquelle repose la VBS. Le démarrage sécurisé UEFI est abordé dans l'annexe A comme un prérequis à l'utilisation de la sécurité reposant sur la virtualisation (VBS), de *Device Guard* et de *Credential Guard*.

Le démarrage sécurisé UEFI repose sur une infrastructure de gestion de clés (IGC) pour vérifier l'intégrité et l'authenticité des composants logiciels avant d'en autoriser l'exécution. Ces composants logiciels peuvent être :

- des pilotes de micrologiciels (*firmware drivers*) ;
- des pilotes UEFI stockés sur disque dur ;
- des applications UEFI ;
- des chargeurs de démarrage UEFI (*bootloaders*) ;
- des OROMS (*Option ROMS*).

L'IGC est stockée en mémoire non volatile (mémoire flash⁶) au cours des étapes d'installation dans la chaîne de fabrication du matériel. De manière synthétique, cette IGC se compose essentiellement :

- d'une base de données (DB ou *signatures database*) de signatures cryptographiques et d'empreintes de composants logiciels UEFI approuvés. Cette base de données fait office de liste blanche de composants logiciels (chargeurs de démarrage compris) ;
- d'une base de données de composants logiciels UEFI révoqués (DBX ou *revoked signatures database*) dont le fonctionnement similaire à la DB mais faisant office de liste noire ;
- d'un magasin de clés publiques d'enrôlement (KEK, *Key Exchange Key* ou *Key Enrollement Key*) qui valident l'intégrité et l'authenticité des éléments intégrés aux DB et DBX. Cela permet de mettre à jour ces deux bases pendant la durée de vie du système (via *Windows Update* par exemple) sans rompre la confiance dans les listes blanches et noires de composants logiciels de l'UEFI ;

6. La mémoire flash est une mémoire ré-inscriptible dont les données sont conservées lors d'une mise hors tension de l'équipement.

- d'une clé de plateforme (PK, *Platform Key*) qui peut être qualifiée de clé principale de l'UEFI. La partie publique de la clé est stockée en NVRAM du matériel tandis que la partie privée⁷ est séquestrée de manière sécurisée par le fabricant.

En authentifiant les composants logiciels UEFI par des mécanismes de signature cryptographique, et en attestant de leur appartenance à la configuration souhaitée matérialisée par des listes blanches, le démarrage sécurisé UEFI réduit les risques d'exécution de code malveillant pendant la phase de démarrage⁸.

R4

Activer le démarrage sécurisé UEFI

Que l'on utilise ou non les fonctionnalités de sécurité reposant sur la virtualisation, il est fortement recommandé d'activer le démarrage sécurisé UEFI afin de protéger la chaîne de démarrage des postes de travail. Sans démarrage sécurisé, la VBS présente un gain de sécurité bien moindre.

L'IGC UEFI joue le rôle d'une base de confiance. Dès lors que cette IGC contient des clés ou empreintes indésirables, la confiance dans la chaîne de démarrage est perdue. En fonction des besoins de sécurité de l'entité, l'activation du démarrage sécurisé UEFI devrait s'accompagner de procédures de contrôle de conformité de l'IGC UEFI des équipements :

R4 *

Ignorer le contrôle de conformité de l'IGC UEFI

Pour la plupart des systèmes d'information, qui ont un besoin de sécurité standard, les attaques qui reposent sur une compromission de la chaîne de démarrage sécurisé UEFI sont généralement d'un degré de sophistication au-delà des scénarios de menace redoutés. Dans ce cas, le contrôle de conformité de l'IGC UEFI des équipements n'est pas une priorité.

R4 **

Mettre en œuvre du contrôle de conformité de l'IGC UEFI

Lorsque le besoin de sécurité des postes de travail est élevé, il est recommandé de compléter l'activation du démarrage sécurisé UEFI par un contrôle de conformité régulier de l'IGC UEFI des équipements.

L'annexe G détaille les enjeux de sécurité relatifs à l'IGC UEFI et propose des pistes pour en contrôler la conformité.



Information

La mise en œuvre d'une solution de chiffrement du disque dur⁹ est une mesure de sécurité conseillée en complément du démarrage sécurisé.

7. La partie privée de la PK peut être utilisée pour modifier le magasin des KEK ou pour signer des mises à jour de firmware UEFI.

8. Les termes *bootkit* et *rootkit* sont souvent utilisés pour désigner des codes malveillants généralement bien dissimulés et chargés à bas niveau avant ou pendant le démarrage du système d'exploitation.

9. La liste des produits de chiffrement de disques qualifiés par l'ANSSI peut être consultée sur la page [4].

2.5 Activation de la sécurité reposant sur la virtualisation

L'hyperviseur est en capacité de faire respecter les permissions W^X ¹⁰ (*Write or Execute*) sur la mémoire vive. Pour mettre en place ces permissions de manière efficace, un niveau de privilèges supérieur à celui nécessaire à la modification de la mémoire protégée est nécessaire. Dans le cas présent, l'hyperviseur possède un niveau de privilèges plus élevé que le noyau Windows standard (le système traditionnel avec lequel l'utilisateur interagit), il peut donc mettre en place une telle protection pour la mémoire système de ce dernier. Ainsi, si des programmes malveillants accèdent au noyau Windows standard, leurs effets peuvent être sérieusement limités car l'hyperviseur peut les empêcher d'exécuter du code.

Mettre en œuvre la VBS sans utiliser un seul *trustlet* est donc déjà un apport de sécurité dans la mesure où l'hyperviseur protège, entre autres, le noyau du système d'exploitation. Par ailleurs, la VBS intègre également la fonctionnalité *Hyper Guard* qui utilise l'hyperviseur pour détecter d'éventuels *rootkits*. *Hyper Guard* empêche la modification de certains registres processeurs (tels que les MSRs de débogage via l'instruction [23] par exemple) et surveille certaines structures du noyau en isolant le composant *PatchGuard* du noyau Windows lui-même. Bien que la VBS soit un prérequis de *Device Guard* et *Credential Guard*, c'est donc également une fonctionnalité de sécurité en soit.

R5

Activer la VBS sur tout poste de travail compatible

Il est recommandé d'activer la sécurité reposant sur la virtualisation (VBS) sur tout poste de travail compatible afin de réduire le risque que des programmes malveillants, ayant compromis le noyau Windows, puissent endommager des structures de ce dernier.



Attention

Le niveau de sécurité apporté par la VBS est fortement réduit dès lors que l'intégrité de la chaîne de démarrage n'est pas assurée par l'activation du démarrage sécurisé (R4). Il est également réduit lorsque les protections DMA ne sont pas activées.

L'annexe F illustre une méthode de déploiement de la VBS sur les postes de travail, par GPO, via une administration centralisée reposant sur un annuaire *Active Directory*. Pour rappel, l'annexe C aborde la problématique d'identification des postes cibles afin de limiter le périmètre de déploiement aux seuls postes compatibles avec les fonctionnalités de VBS.

10. Les permissions W^X permettent de séparer la mémoire inscriptible de la mémoire exécutable. Ainsi, si une vulnérabilité résulte en une corruption de la mémoire inscriptible, elle ne peut être exploitée directement.



Information

Pour les versions de Windows 10 antérieures à la version 1607, l'utilisation du VSM sur les postes de travail nécessite l'installation préalable de deux fonctionnalités Windows :

- le Mode Utilisateur Isolé (IUM, *Isolated User Mode*) ;
- l'hyperviseur Hyper-V.

Ces fonctionnalités ne sont pas nécessairement installées par défaut. Une entité pourra choisir soit de déployer des images (master) Windows dans lesquelles elles sont déjà installées, soit devra les installer *a posteriori*, par exemple par GPO.

Depuis la version 1607 de Windows 10 (*Redstone 1*), l'activation des fonctionnalités Windows permettant d'utiliser la sécurité reposant sur la virtualisation n'est plus nécessaire. La fonctionnalité Windows de Mode Utilisateur Isolé n'existe plus et l'hyperviseur nécessaire au VSM a été décorrélé de la fonctionnalité Windows d'hyperviseur Hyper-V.

L'[annexe B](#) illustre une méthode de déploiement de ces fonctionnalités sur les postes de travail par GPO, via une administration centralisée reposant sur un annuaire *Active Directory*. L'[annexe C](#) aborde la problématique de l'identification des postes cibles afin de limiter le périmètre de déploiement aux seuls postes compatibles avec les fonctionnalités de VBS.

2.6 Incompatibilité avec les solutions de virtualisation tierces

Tout comme Hyper-V, de nombreuses solutions de virtualisation du marché, telles que VMWare Workstation ou VirtualBox, utilisent les extensions de virtualisation du matériel *Intel-VT* ou *AMD-V* pour fonctionner. Ces extensions ne peuvent pas être partagées entre plusieurs solutions de virtualisation sur un même système mais un hyperviseur est bien souvent capable de les exposer aux machines virtuelles qu'il exécute, pour réaliser de la virtualisation imbriquée, ou *nested virtualization*.

Une fois la VBS activée, c'est donc Hyper-V qui utilise les extensions de virtualisation du matériel et qui les expose au système Windows qu'il virtualise. À la date de publication de ce guide, certaines solutions de virtualisation tierces telles que VMWare Workstation ou VirtualBox refusent toutefois de les utiliser. Des mises à jour sont attendues des éditeurs respectifs afin de corriger ces problèmes de compatibilité.

3

Device Guard et l'intégrité du code

Device Guard n'est pas, à proprement parler, une fonction de sécurité mais un terme qui désigne les deux fonctionnalités de sécurité suivantes :

- l'intégrité du code protégé par hyperviseur (HVCI, *HyperVisor Code Integrity*);
- l'intégrité du code configurable (CCI, *Configurable Code Integrity*).

D'autre part et bien que *Credential Guard* soit souvent mentionné comme étant une fonctionnalité de *Device Guard*, il est considéré par Microsoft comme étant indépendant de *Device Guard*. *Credential Guard* se configure toutefois dans la section *Device Guard* des modèles d'administration de Windows 10. La mise en œuvre de *Credential Guard* est abordée en section 4.

Enfin, pour rappel et tel qu'indiqué en section 2.4, la mise en œuvre de *Device Guard* nécessite l'activation du démarrage sécurisé UEFI ainsi que de la sécurité reposant sur la virtualisation (VBS). La liste des prérequis de *Device Guard* peut être consultée en [annexe A](#).

3.1 Caractéristiques générales de l'intégrité du code

L'intégrité du code (*Code Integrity*, CI) est un mécanisme qui améliore la sécurité du système en validant l'intégrité de certains fichiers avant leur chargement en mémoire vive. Les fichiers concernés peuvent être :

- les pilotes ;
- les binaires (exécutables, bibliothèques, etc.) en mode noyau ou en mode utilisateur ;
- les scripts¹¹ ;
- les paquets d'installation au format .MSI.

Cette validation d'intégrité repose essentiellement sur une vérification de leur signature cryptographique, mais peut également utiliser leur empreinte cryptographique voire leur nom de fichier.

3.2 Architecture des mécanismes d'intégrité du code et apports de Device Guard

L'architecture de *Code Integrity* se compose de deux mécanismes de vérification du code exécuté en mode noyau (KMCI, *Kernel Mode Integrity*) et en mode utilisateur (UMCI, *User Mode Integrity*) :

11. Les scripts PowerShell, .VBS, .JS, .WSF et .WSC sont concernés, mais les scripts .BAT et .CMD sont exclus.

- KMCI a été introduit sous Windows Vista pour interdire le chargement, dans le noyau, des programmes et pilotes non signés par une liste restreinte d'autorités de certification (la politique *Kernel Mode Code Signing* (KMCS)). Avec Windows 10 et *Device Guard*, KMCI devient configurable et permet aux entités de contrôler le code autorisé à s'exécuter en mode noyau, c'est-à-dire les pilotes principalement ;
- UMCI a été introduit sous x86¹² avec Windows 10 et *Device Guard* pour permettre la vérification des signatures du code chargé en espace utilisateur.

Une stratégie d'intégrité du code sera ainsi réellement appliquée par les composants KMCI et UMCI de manière complémentaire.



Information

Sur les versions 64 bits de Windows 10, les pilotes en mode noyau doivent être signés numériquement. Le certificat du signataire doit contenir un *Extended Key Usage* (EKU) de type *CodeSigning* et être émis par une autorité racine déclarée dans le magasin ordinateur des « Autorités de certification racines de confiance ».

De plus, avec Windows 10, les pilotes doivent être co-signés par Microsoft, et plus précisément par le portail WHQL (*Windows Hardware Quality Labs*). En outre, depuis octobre 2015, le portail WHQL accepte uniquement les envois de pilotes qui possèdent un certificat de signature de code à validation étendue valide¹³).

Avec Windows 10 et *Device Guard*, *Code Integrity* devient donc configurable, on parle alors d'intégrité du code configurable (*Configurable Code Integrity* ou CCI voire également *CI Policy* en fonction des articles et documentations). L'intégrité du code configurable permet ainsi aux entités de contrôler de manière granulaire le code autorisé ou non à s'exécuter sur les ordinateurs.

3.3 Différences entre Device Guard et Applocker

L'intégrité du code configurable offre une granularité plus importante que les stratégies de restriction logicielle *Applocker*, dont les recommandations de mise en œuvre sont définies dans [3].

Il contrôle l'exécution des pilotes et programmes chargés en mode noyau, ce que ne permettent pas les stratégies de restriction logicielle *Applocker*. Il permet également de définir les *plug-ins*, extensions et modules qui peuvent s'exécuter à partir d'applications spécifiques. Il est ainsi possible de n'autoriser l'usage de certaines bibliothèques que pour une application donnée, ou au contraire de n'interdire l'utilisation d'une bibliothèque que pour une application bien précise. Par ailleurs, KMCI et UMCI sont implémentés plus profondément dans le système d'exploitation que ne l'est *Applocker*, ce qui apporte une plus grande sécurité mais également un périmètre fonctionnel plus étendu.

En revanche, l'intégrité du code configurable est bien moins aisée à mettre en œuvre. De par son périmètre fonctionnel étendu au niveau noyau, une mauvaise configuration ou un pilote incompatible peut entraîner un blocage complet du système (se référer à la section 3.5 pour plus de

12. Le composant UMCI existe en fait pour la plateforme ARM depuis Windows Phone 7.0 et Windows RT, mais n'a été introduit sous x86 qu'avec Windows 10.

13. *Extended Validation certificate*.

détails concernant les points de vigilance à retenir). Les règles de la stratégie d'intégrité du code configurable doivent donc prendre en compte les spécificités des programmes et des pilotes chargés en mode noyau de chaque ordinateur.

3.4 Mise en œuvre de Device Guard

3.4.1 Stratégie d'intégrité du code configurable (CCI)

Le déploiement d'une stratégie d'intégrité du code configurable (CCI) est tout à fait possible sans VBS et sans démarrage sécurisé UEFI. Il ne nécessite donc pas de prérequis matériels particuliers. Par contre, quelques points de vigilance abordés en section 3.5 sont à retenir avant tout déploiement d'une stratégie d'intégrité du code configurable.

Le présent document n'a pas vocation à être un guide de mise en œuvre d'une stratégie d'intégrité du code configurable. Microsoft propose en revanche un guide de déploiement [25] de cette fonctionnalité sur *Windows IT Center*.

R6

Appliquer CCI en mode enforced sur les postes de travail sensibles

La maîtrise du code autorisé à s'exécuter en mode noyau et dans l'espace utilisateur des postes de travail est un moyen efficace de lutte contre les codes malveillants. Le déploiement d'une stratégie d'intégrité du code en mode de mise en conformité (*enforced*) sur les postes sensibles est donc recommandé dans une démarche de défense en profondeur.

Sa mise en œuvre doit néanmoins être réalisée avec précaution étant donné les blocages qu'une mauvaise configuration pourrait occasionner (se référer à la section 3.5).

R7

Appliquer CCI sur les autres postes de travail

Il est recommandé de mettre en œuvre une stratégie d'intégrité du code configurable sur l'ensemble des postes de travail.

Pour appliquer cette recommandation R7, plusieurs scénarios de mise en œuvre de se dessinent en fonction de la maturité en SSI de l'entité et des besoins de sécurité :

R7 *

Appliquer CCI en mode audit sur les autres postes de travail

Le déploiement de stratégies d'intégrité du code configurable en mode audit sur les postes de travail ne présente aucun risque tout en étant transparent pour les utilisateurs. Le journal des événements Windows contiendra alors la liste de tous les codes qui auraient été potentiellement bloqués par *Device Guard* en mode de mise en conformité (*enforced*), ce qui peut s'avérer utile en matière de traçabilité lorsqu'une investigation forensique doit être menée.

Cette première étape permettra par la suite de disposer d'une base de connaissance du code exécuté sur les postes de travail (pilotes, binaires, scripts etc.) dans une éventuelle optique de durcissement ultérieur.

R7 **

Appliquer CCI en mode enforced sur les autres postes de travail

La maîtrise du code autorisé à s'exécuter en mode noyau et dans l'espace utilisateur de tous les postes de travail est un moyen efficace de lutte contre les codes malveillants. Le déploiement d'une stratégie d'intégrité du code en mode de mise en conformité (*enforced*) sur l'ensemble des postes de travail est donc pertinent dans une démarche de défense en profondeur lorsqu'un fort besoin de sécurité le justifie.

Le déploiement d'une stratégie d'intégrité en mode de mise en conformité doit être réalisé avec précaution, il peut s'avérer complexe à mettre en œuvre et à maintenir pour la plupart des systèmes d'information (se référer à la section 3.5)

i

Information

Pour les applications qui ne sont pas signées numériquement ou qui sont signées avec un certificat qui n'est pas inclus dans la stratégie d'intégrité du code configurable, la documentation de *Device Guard* [25] sur *Windows IT Center* détaille les étapes à suivre pour générer un fichier catalogue recensant les empreintes numériques de type SHA-256 des fichiers approuvés. Ce catalogue peut ensuite être signé et distribué avec l'application afin de s'exécuter sur un système protégé par les mécanismes d'intégrité du code.

L'annexe C présente des méthodes d'identification des postes de travail cibles pour l'application de GPO de mise en œuvre de *Device Guard*.

3.4.2 Intégrité du code protégé par l'hyperviseur (HVCI)

Pour renforcer la sécurité de la fonctionnalité *Code Integrity*, la VBS rend possible le déport de certaines opérations de contrôle (c'est-à-dire, entre autres, de vérification cryptographique pour l'autorisation ou le refus d'exécution) dans un *Trustlet* protégé par le VSM. Cette fonctionnalité est appelée « intégrité du code protégé par l'hyperviseur » (HVCI).

HVCI renforce la confiance dans les mécanismes de sécurité suivants :

- la vérification de l'intégrité du code exécuté en mode noyau (KMCI), qui protège contre les pilotes ou fichiers système incorrects. Lorsque le composant KMCI est protégé par la sécurité reposant sur la virtualisation (VBS), le système d'exploitation et le code tiers qui s'exécutent en mode noyau bénéficient d'une protection complémentaire contre les vulnérabilités de type corruption de mémoire ;
- la vérification de l'intégrité du code configurable (CCI) puisque ces mécanismes sont inclus dans la chaîne de confiance apportée par la sécurité reposant sur la virtualisation (VBS). Les règles d'intégrité du code configurable sont ainsi censées être toujours appliquées même si une vulnérabilité permettait un accès non autorisé à la mémoire en mode noyau.

R8

Mettre en œuvre HVCI sur les postes de travail compatibles

Sur les postes de travail compatibles, il est recommandé de mettre en œuvre la fonctionnalité de protection par hyperviseur de l'intégrité du code (HVCI). Cela permet de renforcer la sécurité des mécanismes de KMCI et d'UMCI, qu'une stratégie d'intégrité du code configurable ait été déployée ou non.

Il est en revanche primordial de vérifier au préalable que les pilotes de périphériques présents sur ces ordinateurs soient bien compatibles avec le mode HVCI sans quoi leurs systèmes d'exploitation pourraient ne plus démarrer (se référer à la section 3.5).

L'annexe C présente des méthodes d'identification des postes de travail cibles pour l'application de GPO de mise en œuvre de *Device Guard*. Sur ces postes de travail cibles, HVCI peut être configuré de deux manières :

- avec verrouillage UEFI ;
- sans verrouillage UEFI.

Le verrouillage UEFI assure que la désactivation de la fonctionnalité de sécurité depuis le système d'exploitation n'est pas possible, quel que soit le niveau de privilèges de l'utilisateur. Sa désactivation nécessite alors une intervention physique sur le poste de travail au moment de son démarrage et avant même le chargement du système d'exploitation, ce qu'un attaquant ne peut pas réaliser à distance. Cette configuration s'effectue par GPO, comme illustré en annexe F.

R9

Mettre en œuvre HVCI avec verrouillage UEFI

Sur les postes de travail où HVCI est activé, il est recommandé qu'il le soit avec verrouillage UEFI.

3.5 Points d'attention à prendre en compte avant toute mise en œuvre de Device Guard

Avant tout déploiement d'une stratégie d'intégrité du code configurable ou de la fonctionnalité de protection par hyperviseur de l'intégrité du code (HVCI), il est vivement conseillé de systématiquement tester les politiques en environnement de test. Plusieurs points de vigilance sont par ailleurs à prendre en considération, ici abordés de manière non exhaustive.

Pour commencer, notons que toute modification malveillante d'une stratégie d'intégrité du code configurable pourrait se traduire par un déni de service, car son application pourrait empêcher des systèmes de démarrer. Il est donc important de protéger l'intégrité de cette stratégie.



Attention

Les permissions d'accès au fichier de stratégie d'intégrité du code configurable doivent être configurées de manière à ce que seuls les administrateurs en charge de sa maintenance y aient accès en écriture. Lorsque le chemin d'accès désigne un fichier stocké localement sur les postes de travail, ce dernier ne devrait être accessible en écriture que par les administrateurs locaux et le disque dur devrait être chiffré.



Attention

Une stratégie d'intégrité du code configurable devrait idéalement être protégée en intégrité par signature en ayant recours à un certificat de signature de code, tel qu'indiqué par Microsoft sur la page [26]. Les clés privées associées à ces certificats devraient par ailleurs être protégées et utilisées sur des postes hors ligne dédiés aux IGC ou, à défaut, des postes d'administration du niveau de sécurité adéquat. En effet, tout vol d'une de ces clés privées pourrait se traduire par une modification malveillante de la stratégie d'intégrité du code configurable.

Il est également important d'avoir conscience qu'une stratégie d'intégrité du code configurable pourrait être contournée. Certaines mesures devraient donc être appliquées pour le limiter.



Attention

Les points de vigilance mentionnés par le guide [3] de l'ANSSI concernant les règles de type « nom de fichier » ou « empreinte cryptographique » sont également applicables à *Code Integrity*. Le lecteur est donc invité à les consulter.



Attention

Il existe plusieurs méthodes de contournement connues de Code Integrity, qu'il est recommandé de bloquer par des règles de type *Deny*. La page [27] du *Windows IT Center* les recense sous la forme d'une politique au format XML et explique comment la mettre en œuvre en l'intégrant à la politique de l'entité via la *Cmdlet Merge-CIPolicy*.

Enfin, notons la nécessité de faire attention aux problèmes de compatibilité des postes de travail.



Attention

La mise en œuvre d'une stratégie d'intégrité du code configurable ou de la fonctionnalité de protection par hyperviseur de l'intégrité du code (HVCI) peut empêcher le chargement de pilotes indispensables au démarrage du système d'exploitation. Il est conseillé de vérifier la compatibilité des pilotes des postes de travail concernés au préalable et de consulter la page [28] afin de gérer convenablement cette problématique.

L'outil *Device Guard and Credential Guard hardware readiness tool* de Microsoft, téléchargeable depuis la page [18], peut être utilisé pour vérifier la compatibilité des pilotes (se reporter à la section A.4 pour l'adaptation de cet outil à des systèmes de langue française).



Attention

L'utilisation d'une stratégie d'intégrité du code configurable avec l'option `HvciOptions` configurée en mode `Strict`¹⁴ est fortement déconseillée. Il est en effet très probable que les protections additionnelles apportées par ce mode, non supporté par Microsoft, empêchent le démarrage du système d'exploitation.

14. Le mode strict correspond à l'option `HvciOptions = 3` dans les fichiers XML de stratégie de *Code Integrity*, ajouté manuellement ou par l'argument `-Strict` du *Cmdlet Set-HVCIOptions*.

4

Credential Guard

Les scénarios d'attaque les plus courants mettent en œuvre la récupération de secrets d'authentification sur les systèmes, pour ensuite procéder à des élévations de privilèges successives jusqu'à accéder aux biens essentiels d'un système d'information. *Credential Guard* est une fonctionnalité dont l'objectif est de protéger les secrets d'authentification en mémoire vive.

Credential Guard ne peut être activé que sur des postes qui utilisent le VSM (préalablement abordé en section 2) et qui répondent aux prérequis additionnels abordés en [annexe A](#).

4.1 Principes techniques et apports en matière de sécurité

L'autorité de sécurité locale `LSASS.exe` est un processus sensible du système d'exploitation qui est responsable de l'authentification utilisateur. Ce processus stocke en mémoire vive les secrets d'authentification NTLM sous forme de condensats (*Hash*) ainsi que les tickets TGT (*Ticket Granting Ticket*) Kerberos ¹⁵.

Les condensats NTLM, s'ils sont extraits de la mémoire par un attaquant qui dispose des privilèges d'administration locaux, peuvent par exemple être utilisés à la place du mot de passe lui-même (tant qu'il ne change pas) pour s'authentifier auprès d'autres ressources de l'environnement AD. C'est également le cas lors d'une authentification par carte à puce ¹⁶. Cette technique d'authentification par réutilisation des condensats s'appelle communément *Pass-the-Hash*. Elle est détaillée dans le document [16] de Microsoft.

La problématique est identique avec les tickets TGT Kerberos. Ils peuvent être extraits du processus `LSASS.exe` et utilisés pour obtenir des tickets TGS (*Ticket Granting Service*) Kerberos auprès du KDC (*Key Distribution Center*). Ces TGT peuvent être utilisés à la place du mot de passe lui-même (qu'il change ou non, tant que le TGT délivré est valide) pour s'authentifier auprès d'autres ressources de l'environnement. Ce scénario est toujours fonctionnel lorsque l'utilisateur s'authentifie à l'aide d'une carte à puce. Cette technique d'authentification par réutilisation des TGT et TGS Kerberos s'appelle communément le *Pass-the-Ticket* et elle est également détaillée dans le document [16].

La sécurisation d'un annuaire *Active Directory* (AD) consiste en grande partie à se protéger contre ce type d'attaques sur les postes de travail, en réduisant les possibilités de récupération de secrets d'authentification de comptes privilégiés de l'AD.

Credential Guard décompose le processus `LSASS.exe` de sorte à déporter ses composantes sensibles (la mémorisation des secrets d'authentification) dans un *Trustlet* `LSAIso.exe` (voir section 2). Le

15. Pour se familiariser avec Kerberos, se reporter à [12].

16. Pour plus d'informations sur la particularité d'une authentification par carte à puce, se référer à l'annexe VII du document [2].

processus `LSASS.exe` du système d'exploitation ne mémorise donc plus que des blobs¹⁷ chiffrés, éphémères et valides seulement jusqu'au prochain arrêt ou redémarrage du système. Ces blobs chiffrés sont ensuite transmis au `Trustlet LSALSO.exe`, seul capable de les déchiffrer, lorsque des authentifications doivent être effectuées.

Le processus `LSASS.exe` exposé sur le système d'exploitation ne mémorise ainsi plus de secrets d'authentification réutilisables, c'est le `Trustlet LSALSO.exe` qui le fait et sa surface d'attaque minimale ainsi que son isolation par les mécanismes de virtualisation protègent ce dernier contre des attaques opérées depuis le système d'exploitation. Les secrets d'authentification de `LSALSO.exe` ne pouvant pas être extraits par un attaquant sans compromettre l'hyperviseur puis le VSM, les risques de déplacement latéral d'un attaquant puis d'élévation de privilèges se trouvent alors atténués.

4.2 Périmètre de mise en œuvre

L'objectif des attaquants est généralement d'exfiltrer ou de saboter les biens essentiels d'une entité. Pour mener à bien cet objectif, les attaquants cherchent généralement en premier lieu à compromettre de simples postes de travail de bureautique via les méthodes classiques d'exploitation de vulnérabilités des logiciels les plus exposés (suites bureautiques, visionneuses diverses, navigateurs Web, etc.).

Une fois un poste de travail compromis, celui-ci sert aux attaquants de porte d'entrée sur le SI pour obtenir des comptes à privilèges qui permettent d'accéder aux ressources hébergeant des biens essentiels. Pour obtenir ces comptes à privilèges à partir d'un simple compte utilisateur de l'AD, les attaquants se déplacent latéralement d'un poste de travail à l'autre (par exemple par les techniques de *Pass the Hash* ou de *Pass the Ticket* abordées section 4.1) dans l'espoir de finir par récupérer des secrets d'authentification de comptes qui disposent de privilèges plus élevés. Les attaquants procèdent ainsi par itération jusqu'à obtenir les secrets d'authentification d'un compte disposant des privilèges d'administration du domaine AD.

En fonction de la maturité en SSI de l'entité, les comptes à privilèges d'administration du domaine AD peuvent généralement être récupérés par les attaquants sur de simples postes de travail de bureautique ou sur des postes d'administration de l'AD accessibles depuis le réseau de bureautique après déplacement latéral. Pour s'en prémunir, il est entre autres recommandé d'adopter une bonne gestion des comptes à privilèges de l'AD, une architecture d'administration sécurisée [6], ainsi que de bonnes pratiques de télé-assistance des utilisateurs [5] et de téléadministration des postes de travail.

Dans une démarche de défense en profondeur, la mise en œuvre de *Credential Guard* est intéressante sur tout poste de travail. Néanmoins, c'est véritablement sur le périmètre des postes de travail sur lesquels sont manipulés des comptes à privilèges de l'AD que sa mise en œuvre est prioritaire. Comme abordé ci-dessus, ce périmètre peut, en fonction du niveau de maturité en SSI de l'entité, inclure tout poste de travail ou seulement les postes d'administration de l'AD. En effet, dès lors que l'architecture AD est compartimentée en tiers selon les recommandations de Microsoft [15] et de l'ANSSI [2], et que les comptes à privilèges de l'AD ne peuvent pas être utilisés sur des postes de travail de bureautique exposés aux menaces, le déploiement de *Credential Guard* sur ces derniers revêt dans ce cas une importance moindre.

17. Un blob est un ensemble de données binaires sérialisées de grande taille.

Par ailleurs, bien que les prérequis matériels de *Credential Guard* (voir [annexe A](#)) ne vont pas nécessairement permettre son déploiement à l'échelle d'un SI complet, il reste cependant largement envisageable de le déployer sur le périmètre bien plus restreint des postes d'administration de l'AD (*tiers 0*¹⁸) et des ressources de l'AD (*tiers 1*).



Attention

La mise en œuvre de *Credential Guard* doit être considérée comme une mesure de sécurité parmi d'autres dans une démarche de défense en profondeur. Une entité ne doit en aucun cas considérer que *Credential Guard* permet de s'affranchir d'une gestion sécurisée des comptes à privilèges.

R10

Mettre en œuvre Credential Guard

La mise en œuvre de *Credential Guard* est recommandée pour protéger les secrets d'authentification stockés sous forme de condensats NTLM ou de tickets Kerberos.

Deux scénarios de mise en œuvre de cette recommandation R10 sont envisageables en fonction des besoins de sécurité de l'entité :

R10 *

Mettre en œuvre Credential Guard sur les postes de travail sensibles

Le déploiement de *Credential Guard* est au minimum recommandé sur les postes de travail sensibles, sur lesquels sont par exemple manipulés des comptes à privilèges de l'annuaire Active Directory (du Tiers 0 et éventuellement du Tiers 1), afin d'atténuer les risques de déplacement latéral et d'élévation de privilèges des attaquants.

R10 **

Mettre en œuvre Credential Guard sur tous les postes de travail

Le déploiement de *Credential Guard* sur l'ensemble des postes de travail (de bureau et d'administration) est un atout notable dans une démarche de défense en profondeur du système d'information.

4.3 Limites de Credential Guard

En plus des limites de sécurité intrinsèques à la VBS évoquées en section 2.3, *Credential Guard* ne permet pas de protéger les secrets d'authentification :

- des comptes locaux et des comptes Microsoft¹⁹ ;
- des programmes n'utilisant pas le processus `lsass.exe` de l'autorité de sécurité locale (mots de passe enregistrés dans les logiciels, dans les scripts, etc.) ;
- des enregistreurs de frappes matériels ou logiciels qui intercepteraient leur saisie ;
- des attaques physiques de manière générale.

18. Les notions de *Tiers* dans la gestion des comptes à privilèges d'un annuaire Active Directory sont abordées par Microsoft sur la page [15].

19. Les comptes Microsoft désignent un service d'authentification de Microsoft dans le nuage.

R11

Assurer la sécurité physique des postes de travail

La mise en œuvre de *Credential Guard* n'est pas une mesure de sécurité protégeant les secrets d'authentification contre des attaques physiques. En particulier, la sécurité physique des postes de travail sur lesquels sont manipulés des comptes à privilèges de l'Active Directory (du *tiers 0* et éventuellement du *tiers 1*) doit impérativement être assurée par ailleurs.

R12

Mettre en œuvre une solution de gestion des mots de passe administrateurs locaux

Credential Guard ne protège pas les secrets d'authentification des comptes locaux. Il reste donc recommandé de mettre en œuvre une solution de gestion des comptes administrateurs locaux intégrés (*Built-in*) dont les principaux objectifs sont d'assurer :

- un mot de passe unique par système ;
- la complexité des mots de passe et leur changement régulier (se référer aux recommandations de l'ANSSI concernant les mots de passe [1]) ;
- le stockage centralisé et sécurisé des mots de passe, accessibles par les équipes habilitées à la télé-administration des postes de travail.



Information

Microsoft LAPS (*Local Administrator Password Solution* [13]) est un exemple de solution de gestion des comptes administrateurs locaux.

R13

Ne pas mémoriser les comptes à privilèges de l'AD

La mémorisation des comptes à privilèges de l'AD par quelque logiciel que ce soit (en dehors de celle réalisée par l'autorité de sécurité locale isolée *LSAIso.exe*) sur des postes de travail doit être proscrite. Dans le cas contraire, les bénéfices de *Credential Guard* et les efforts de gestion des comptes à privilèges en matière de sécurité pourraient être réduits à néant par de mauvaises pratiques.

Credential Guard n'empêche pas non plus un attaquant d'utiliser de manière légitime le contexte utilisateur d'une session ouverte sur un poste de travail compromis. En disposant des droits *System*, un code malveillant pourrait également utiliser les blobs chiffrés du processus *LSASS.exe* pour lui-même obtenir des authentifications auprès du *Trustlet LSAIso.exe* (mais il ne pourrait toutefois le faire que depuis la machine compromise et les blobs chiffrés obtenus ne seraient valides que jusqu'à l'arrêt ou le redémarrage du système).

La solution *Credential Guard* n'est donc pas une fonctionnalité de sécurité qui résout à elle seule le problème de la gestion des comptes à privilèges. Ses limites doivent être prises en compte afin de mettre en œuvre les mesures de sécurité complémentaires nécessaires dans une démarche de défense en profondeur.



Attention

Certains fournisseurs SSP (*Security Support Provider*)²⁰ tiers peuvent ne pas être compatibles avec *Credential Guard* s'ils demandent les condensats de mots de passe à l'autorité de sécurité locale. Cela peut d'ailleurs mener à une dégradation du niveau de sécurité si ces condensats sont par la suite utilisés ou mémorisés de manière peu sécurisée par le fournisseur SSP.

Si des fournisseurs SSP tiers (et points d'accès associés) sont utilisés, il est impératif de tester leur compatibilité avant tout déploiement de *Credential Guard* et de se rapprocher de l'éditeur du fournisseur SSP tiers le cas échéant.

4.4 Mise en œuvre de Credential Guard

L'annexe C présente des méthodes d'identification des postes de travail cibles pour l'application de GPO de mise en œuvre de *Credential Guard*. Sur ces postes de travail cibles, *Credential Guard* peut être configuré de deux manières :

- avec verrouillage UEFI ;
- sans verrouillage UEFI.

Le verrouillage assure que la désactivation de la fonctionnalité de sécurité depuis le système d'exploitation n'est pas possible, quel que soit le niveau de privilèges de l'utilisateur. La désactivation nécessite alors une intervention physique sur le poste de travail au moment de son démarrage et avant même le chargement du système d'exploitation, ce qu'un attaquant ne peut pas réaliser à distance. Cette configuration se fait simplement par GPO, comme illustré en annexe F.

R14

Mettre en œuvre Credential Guard avec verrouillage UEFI

Sur les postes de travail, *Credential Guard* doit être activé avec verrouillage UEFI.

L'annexe D explique le processus de désactivation de *Credential Guard* suite à son activation avec verrouillage UEFI.

R15

Sensibiliser les utilisateurs au verrouillage UEFI de Credential Guard

L'activation de *Credential Guard* avec verrouillage UEFI doit s'accompagner d'une sensibilisation des utilisateurs. En cas d'affichage de la demande de désactivation de *Credential Guard* lors du démarrage de leur poste de travail, les utilisateurs doivent comprendre la question qui leur est posée, refuser la désactivation et contacter aussitôt leur référent SSI, afin que soient menées les investigations nécessaires.

20. L'Interface SSP (SSPI) est une interface de programmation générique, sur laquelle repose l'authentification sous Windows. Cette SSPI permet d'ajouter de nouvelles méthodes d'authentification au système d'exploitation Windows via des fournisseurs SSP. Les méthodes d'authentification par Kerberos et NTLM sont, par exemple, des fournisseurs SSP nativement intégrés au système. Certains éditeurs tiers proposent des méthodes d'authentification alternatives utilisant des fournisseurs SSP propriétaires.

Annexe A

Prérequis matériels et logiciels

A.1 Prérequis

Pour utiliser les fonctionnalités de sécurité reposant sur la virtualisation, certains prérequis matériels et logiciels doivent impérativement être satisfaits. Le tableau A.1 suivant les récapitule et indique, pour chacun d'eux, s'il s'agit de prérequis applicables à *Credential Guard* (CG), *Device Guard* (DG) ou à la VBS seule.

Élément	Configuration minimale requise	CG	DG	VBS
Processeur	Processeurs avec jeux d'instructions Intel VT-x ou AMD-V et du SLAT (<i>Second Level Address Translation</i>).	X	X	X
Windows	Windows 10 éditions « Entreprise » ou « Éducation ».	X	X	X
UEFI	UEFI répondant aux spécifications UEFI 2.3.1 errata C minimum.	X	X	X
UEFI	UEFI avec démarrage sécurisé activé afin d'éviter la compromission de la chaîne de démarrage (voir section 2.4).	X	X	
Architecture	Architecture 64 bits	X	X	X
Module TPM	Les modules TPM 1.2 et 2.0 fournissent une protection des clés de chiffrement de CG. Elles sont ainsi automatiquement stockées de manière sécurisée dans un microcontrôleur matériel.	X		

Tableau A.1 – Prérequis à l'utilisation des fonctionnalités de sécurité reposant sur la virtualisation

Pour rappel, le niveau de sécurité apporté par la VBS est fortement réduit dès lors que l'intégrité de la chaîne de démarrage n'est pas assurée par l'activation du démarrage sécurisé (R4). Pour renforcer la sécurité des fonctionnalités reposant sur la virtualisation, ou de la VBS seule, il est également conseillé de satisfaire aux prérequis complémentaires du tableau A.2 suivant.

Élément	Configuration complémentaire recommandée	CG	DG	VBS
Processeur	Processeur avec unité de gestion de mémoire d'entrée/sortie IOMMU <i>Intel VT-d</i> ou <i>AMD Vi</i> afin de protéger le système contre les attaques mémoire (comme les attaques DMA).	X	X	X
UEFI	L'UEFI doit pouvoir être sécurisé par mot de passe, ou toute autre méthode d'authentification plus robuste, afin d'empêcher l'accès à ce dernier par un utilisateur non autorisé.	X	X	X
UEFI	Pour prévenir certaines attaques mémoire avancées, il est recommandé d'avoir un UEFI qui implémente la mesure de sécurité MORLock ²¹ .	X		

21. *Memory Overwrite Request control*, voir la page [14] pour plus d'informations.

Élément	Configuration complémentaire recommandée	CG	DG	VBS
UEFI	Le microprogramme UEFI devrait supporter la mise à jour sécurisée afin de pouvoir être mis à jour si des vulnérabilités sont découvertes (voir notamment la section 2.4 évoquant la clé de signature des mises à jour de micrologiciel). Une vulnérabilité dans le microprogramme UEFI pourrait déboucher sur une compromission de la chaîne de démarrage. Ce microprogramme UEFI doit par ailleurs pouvoir être mis à jour via <i>Windows Update</i> .	X	X	X
UEFI	Le fabricant de l'équipement devrait permettre l'ajout de clés de signature dans le magasin KEK et les magasins DB et DBX (voir section 2.4) au moment de la fabrication, afin de permettre à des clients ou vendeurs indépendants d'y intégrer les leurs.	X	X	X
UEFI	Si le mode <i>Connected StandBy</i> (mode intermédiaire entre les états « allumé » et « éteint », utilisé sur les postes de travail conçus pour la mobilité) est pris en charge par le matériel, des spécifications complémentaires devraient être implémentées sur ce dernier afin de protéger l'intégrité du démarrage et de la sortie du mode <i>Connected StandBy</i> contre des attaques physiques.	X	X	X
UEFI	Le matériel devrait être doté d'une interface HSTI ²² de test de sécurité du matériel, et l'équipementier doit fournir la documentation et les outils permettant d'utiliser cette interface pour vérifier la sécurité matérielle et micrologicielle du poste de travail (et notamment sa sécurité par défaut).	X	X	X
UEFI	Les spécifications WSMT (<i>Windows SMM Security Mitigations Table</i>) détaillent de nouvelles tables ACPI définies par Microsoft pour apporter une couche de sécurité complémentaire de protection du SMM (<i>System Management Mode</i>). Cette sécurité s'appelle « protection SMM » ou <i>SMM Mitigation</i> . Elle est recommandée pour que le niveau de sécurité apporté par le VSM ne soit pas dégradé par le micrologiciel UEFI (SMM est sujet à nombreuses vulnérabilités qui peuvent permettre une compromission de l'hyperviseur).	X	X	X
UEFI	Aucune zone mémoire UEFI ne devrait être marquée à la fois exécutable et inscriptible. Cette sécurité s'appelle la « protection NX ». Elle est recommandée pour que le niveau de sécurité apporté par le VSM ne soit pas dégradé par le micrologiciel UEFI.	X	X	X
Pilotes	L'utilisation de pilotes compatibles HVCI permet de renforcer la sécurité en limitant l'exécution de code malveillant au niveau noyau (voir la page MSDN [20]).		X	

Tableau A.2 – Configuration complémentaire recommandée pour l'utilisation des fonctionnalités de sécurité reposant sur la virtualisation

22. Pour plus d'informations sur les interfaces HSTI, se référer à la page MSDN [11].



Information

Les tableaux A.1 et A.2 recensent les prérequis applicables à la version 1703 de Windows 10 (nom de code *Redstone 2*). Ces prérequis peut être amenés à changer avec les versions ultérieures de Windows 10, il est donc conseillé de se référer aux listes de prérequis [24] et [29] à jour sur *Windows IT Center*.

Il est également à noter que des équipements peuvent satisfaire ces prérequis sans pour autant avoir certains mécanismes activés par défaut. Afin de faciliter le déploiement des fonctionnalités de sécurité reposant sur la virtualisation, il est en complément conseillé de vérifier que :

- le firmware UEFI dispose de la virtualisation de la mémoire (jeux d'instructions Intel VT-x ou AMD-V) activée par défaut ;
- le firmware est en mode natif UEFI par défaut et non pas en mode de support de compatibilité (CSM ou *Compatibility Support Module*) ;
- le firmware UEFI est configuré par défaut avec le démarrage sécurisé activé ;
- le TPM en version 1.2 ou 2.0 est activé par défaut ou peut être automatiquement configuré par Windows 10 ;

A.2 Certification du matériel

La prise en charge de certains des prérequis listés dans les tableaux A.1 et A.2 est garantie par la « certification Windows 10 » du matériel²³ :

- la chaîne de démarrage répond aux spécifications UEFI 2.3.1 errata C minimum ;
- le démarrage sécurisé est activé par défaut ;
- un module TPM 2.0 est présent (ajout daté du 28 juillet 2016).

La « certification Windows 10 » du matériel n'est donc pas un critère suffisant de prise en charge des prérequis matériels et logiciels des fonctionnalités de sécurité reposant sur la virtualisation.

En revanche, Microsoft a également publié des spécifications relatives à *Device Guard* et *Credential Guard* (qui peuvent être consultées sur la page MSDN [19]). Ces spécifications s'appliquent indifféremment à *Device Guard* et *Credential Guard* et s'intitulent :

- *Capable* si les spécifications minimales énumérées dans le tableau A.1 sont prises en charge²⁴ mais nécessitent d'être configurées et activées ;
- *Ready* si les spécifications minimales énumérées dans cette section sont prises en charge, configurées et activées par défaut.

Les fabricants doivent donc s'y conformer s'il veulent obtenir, par exemple, les mentions *Device Guard Ready* ou *Credential Guard Ready* de leur matériel, ce qui est fort peu répandu à la date de publication de ce document.

23. Pour plus d'informations sur les prérequis à la « certification » Windows 10 du matériel, se référer à la page [21].

24. Les prérequis concernant les protections NX et SMM ne sont applicables que depuis la version 1703 de Windows 10.

A.3 Script de diagnostic

Microsoft a publié un script PowerShell de test de certains prérequis (HSTI, SMM, NX, MORLock, etc.) matériels et logiciels d'un équipement pour l'utilisation des fonctionnalités de VBS intitulé *Device Guard and Credential Guard hardware readiness tool*²⁵.



Information

Ce script enregistre ses résultats en base de registre, dans la clé HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities, sous forme de valeurs DWORD :

- CG_Capable (vaut 1 si compatible et 0 sinon);
- DG_Capable (vaut 1 si compatible et 0 sinon);
- HVCI_Capable (vaut 1 si compatible et 0 sinon).

Si ce script était systématiquement exécuté au moins une fois sur chaque poste de travail (par GPO ou dans le cadre du processus de déploiement des images Windows), ces valeurs en base de registre pourraient par exemple servir pour le « ciblage au niveau de l'élément »²⁶ de stratégies de préférence afin de mettre en œuvre les fonctionnalités de sécurité reposant sur la virtualisation sur les postes qui les prennent en charge (problématique abordée dans l'annexe C).

A.4 Adaptation du script de diagnostic à un système d'exploitation en langue française

Le script de diagnostic abordé en section A.3, et dont la dernière version est la 3.2 du 8 mai 2017 à date de publication de ce document, ne prend pas en charge les systèmes d'exploitation installés en langue française. Pour rendre ce script compatible, il est nécessaire d'y apporter les modifications indiquées dans les sections A.4.1 et A.4.2 suivantes :

A.4.1 Test de la version de Windows

Dans la section « *Supported OS SKU* », il est testé si l'édition de Windows 10 est compatible via le tableau suivant :

```
$SKUarray = @("Enterprise", "Education", "IoT", "Windows_Server")
```

Exemple A.1 – Diagnostic - Tableau « *SKUarray* » d'origine

25. Le script *Device Guard and Credential Guard hardware readiness tool* est téléchargeable depuis la page [18].

26. Le principe de ciblage au niveau de l'élément est expliqué sur la page TECHNET [9] de Microsoft.

En langue française, « *Enterprise* » est remplacé « *Entreprise* ». Ce tableau doit donc être complété comme illustré ci-dessous :

```
$SKUarray = @("Enterprise", "Entreprise", "Education", "IoT", "Windows_Server")
```

Exemple A.2 – Diagnostic - Tableau « *SKUarray* » modifié

Si des systèmes en d'autres langues doivent être gérés, la même méthode doit être appliquée en ajoutant la traduction de ces éditions dans le tableau *SKUarray*.

A.4.2 Test de l'architecture

Dans la section « *OS Architecture* », il est testé si le système d'exploitation est en 32 ou 64 bits via les conditions suivantes :

```
if($OSArch.Contains("64-bit"))
{
    LogAndConsoleSuccess "64_bit_arch...."
}
elseif($OSArch.Contains("32-bit"))
{
    LogAndConsoleError "32_bit_arch...."
    $DGVerifyCrit.AppendLine("32_Bit_OS, OS_Architecture_failure..") | Out-Null
}
else
{
    LogAndConsoleError "Unknown_architecture"
    $DGVerifyCrit.AppendLine("Unknown_OS, OS_Architecture_failure..") | Out-Null
}
```

Exemple A.3 – Diagnostic - Condition « *OS Architecture* » d'origine

En langue française, l'architecture est indiquée non pas avec un tiret mais avec un espace entre le nombre (32 ou 64) et le mot *bit*. Ces conditions peuvent ainsi être remplacées par :

```
if($OSArch.Contains("64"))
{
    LogAndConsoleSuccess "64_bit_arch...."
}
elseif($OSArch.Contains("32"))
{
    LogAndConsoleError "32_bit_arch...."
    $DGVerifyCrit.AppendLine("32_Bit_OS, OS_Architecture_failure..") | Out-Null
}
else
{
    LogAndConsoleError "Unknown_architecture"
    $DGVerifyCrit.AppendLine("Unknown_OS, OS_Architecture_failure..") | Out-Null
}
```

Exemple A.4 – Diagnostic - Condition « *OS Architecture* » modifiée

Annexe B

Installation des fonctionnalités Windows requises

Comme abordé en section 2, l'utilisation du VSM sur les postes de travail nécessite l'installation de fonctionnalités Windows sur les versions 1511 et antérieures de Windows 10. Une entité pourra choisir de déployer des images (*master*) Windows dans lesquelles ces fonctionnalités sont déjà installées, ou bien de les installer *a posteriori*, par exemple par GPO (cas illustré dans cette annexe). Bien entendu, il est également nécessaire que les jeux d'instructions Intel-VT ou AMD-V soient activés au niveau de l'UEFI des postes de travail.

B.1 Installation des fonctionnalités Windows nécessaires au VSM

L'installation des fonctionnalités peut se faire de différentes manières, et par exemple avec DISM en PowerShell (avec les privilèges d'administration) :

```
# Rappel : Il est inutile d'exécuter ce script sur les postes de travail Windows 10 en versions
# 1607 et ultérieures puisque l'activation des fonctionnalités Windows permettant d'utiliser
# la VBS n'est plus nécessaire.
Enable-WindowsOptionalFeature -Online -FeatureName "IsolatedUserMode" -all -norestart
Enable-WindowsOptionalFeature -Online -FeatureName "Microsoft-Hyper-V-Hypervisor" -all -norestart
```

Exemple B.1 – Commandes PowerShell d'installation des fonctionnalités pour le VSM

Afin de déployer ces fonctionnalités sur un ensemble de postes de travail, il est par exemple possible d'utiliser une GPO qui exécute un script PowerShell au démarrage ou via une tâche (planifiée ou à exécution immédiate). Bien entendu, l'installation de ces fonctionnalités Windows ne devrait être effectuée que sur des postes de travail qui prennent en charge les prérequis obligatoires des fonctionnalités de VBS (prérequis indiqués dans l'annexe A).

B.2 Script PowerShell exécuté par GPO

Cette section illustre le déploiement de ces fonctionnalités Windows sur les postes de travail par une GPO. Cette dernière, détaillée figure B.1, exécute un script PowerShell via une tâche à exécution immédiate.



Attention

Il est imprudent de créer une tâche planifiée ou à exécution immédiate configurée pour s'exécuter avec un compte utilisateur à privilèges du domaine, le mot de passe de ce dernier se trouvant alors enregistré sur les postes de travail et récupérable par un attaquant. L'utilisation du compte NT AUTHORITY\SYSTEM est donc une bonne pratique à laquelle il convient de ne pas déroger dès lors que l'exécution de cette tâche nécessite des privilèges d'administration.

Cette tâche s'exécute dans le contexte du compte NT AUTHORITY\SYSTEM et exécute le programme powershell avec pour argument le script ici stocké dans le partage SYSVOL de l'Active Directory (et idéalement dans le sous-dossier spécifique aux fichiers de la GPO). Comme l'exécution du script d'installation des fonctionnalités Windows peut durer un certain temps, l'avantage d'une tâche à exécution immédiate par rapport à un script de démarrage est que cette exécution n'est pas bloquante pour l'utilisateur et ne ralentira donc pas le démarrage du poste ni l'ouverture de session. Elle peut être exécutée une fois que l'ordinateur est inactif ou simplement plusieurs minutes après le démarrage.

Configuration ordinateur (activée)

Préférences

Paramètres du Panneau de configuration

Tâches planifiées

Tâche immédiate (au minimum Windows 7) (nom : Installation du VSM)

Installation du VSM (ordre : 1)

Général

Tâche		
Nom	Installation du VSM	
N'exécuter que si un utilisateur a ouvert une session	InteractiveToken	
IdUtilisateur	NT AUTHORITY\System	
Exécuter avec les autorisations maximales	LeastPrivilege	
Activé	Oui	

Actions

1. Démarrer un programme	Programme/script	powershell
	Arguments	-ExecutionPolicy bypass -File \\ad.lan\SysVol\ad.lan\scripts\VSMFeaturesInstall.ps1

Paramètres

Arrêter si l'ordinateur n'est plus inactif	Non
Redémarrer si l'état inactif recommence	Non
Ne démarrer la tâche que si l'ordinateur est branché sur le secteur	Oui
Arrêter si l'ordinateur passe en alimentation par batterie	Non
Autoriser l'exécution de la tâche à la demande	Non
Exécuter la tâche dès que possible si un démarrage planifié est manqué	Oui
Arrêter la tâche si elle s'exécute plus de	1 heure
Si la tâche ne se termine pas au moment demandé, forcer son arrêt	Non
Supprimer la tâche si son exécution n'est plus planifiée après	Immédiatement
Si la tâche s'exécute déjà, la règle suivante s'applique	IgnoreNew

Commun

Options

Interrompt le traitement des éléments sur cette extension si une erreur se produit sur cet élément	Non
Supprimer cet élément lorsqu'il n'est plus appliqué	Non
Appliquer une fois et ne pas réappliquer	Non

FIGURE B.1 – Exemple de GPO d'exécution du script PowerShell de déploiement des fonctionnalités Windows requises pour l'utilisation du VSM



Attention

Les emplacements des scripts de démarrage ainsi que des scripts exécutés par tâche planifiée ou à exécution immédiate dans un contexte de compte NT AUTHORITY\SYSTEM doivent être choisis avec vigilance. Seuls les comptes à plus hauts privilèges de l'annuaire *Active Directory* doivent y avoir des droits d'accès en écriture. L'accès en écriture à ces emplacements par un individu malveillant aurait pour conséquence une possibilité d'exécution de code arbitraire privilégié sur les postes de travail.

L'exemple de script PowerShell B.2 illustre une manière possible de procéder à l'installation des fonctionnalités sur les seuls postes qui disposent du démarrage sécurisé UEFI activé. Il s'agit d'un exemple qu'il convient d'adapter au contexte ainsi qu'aux pratiques internes (politique de journalisation, gestion des exceptions, etc.).

```
# Rappel : Il est inutile d'exécuter ce script sur les postes de travail Windows 10 en versions 1607 et ultérieures.
# Pour la journalisation, configurer selon la politique de l'entité :
# - la source des événements (ici "ANSSI.W10.VSM")
# - les identifiants d'événements souhaités (ici 6420x)
New-EventLog -LogName System -Source "ANSSI.W10.VSM" -ErrorAction SilentlyContinue
Write-EventLog -LogName System -Source "ANSSI.W10.VSM" -EntryType Information -Message "Running Microsoft Hyper-V Hypervisor and
IsolatedUserMode features installation script ..." -EventId 64200
Try{
    $IsW10 = [environment]::OSVersion.Version.Major -eq 10
    # Si ce n'est pas un Windows 10, on termine l'exécution du script
    if (-not $IsW10) {
        exit
    }
    $IsRedStone = [environment]::OSVersion.Version.Build -ge 14393
    # Si le système est en version redstone 1 minimum (build >= 14393) alors il n'y a rien à installer
    if ($IsRedStone) {
        Write-Host "This version of Windows 10 does not require any Windows feature installation for VBS"
        Write-EventLog -LogName System -Source "ANSSI.W10.VSM" -EntryType Information -Message "This version of Windows 10 does
not require any Windows feature installation for VBS" -EventId 64201
        exit
    }
    $HasSecureBoot = Confirm-SecureBootUEFI
    # Si le démarrage sécurisé n'est pas activé, inutile d'installer les fonctionnalités Windows
    if (-not ($HasSecureBoot)) {
        Write-Host "No secure boot detected: Hardware is not VSM ready"
        Write-EventLog -LogName System -Source "ANSSI.W10.VSM" -EntryType Information -Message "No windows features will be
installed because secure boot is not enabled" -EventId 64202
        exit
    }
    $HyperVInstallState = (Get-WMIObject -Query 'Select InstallState from Win32_OptionalFeature where name like
"Microsoft-Hyper-V-Hypervisor").InstallState
    $IUMInstallState = (Get-WMIObject -Query 'Select InstallState from Win32_OptionalFeature where name like
"IsolatedUserMode').InstallState
    # Installation de la fonctionnalité Windows IsolatedUserMode si nécessaire
    if (-not ($IUMInstallState -eq 1)){
        Write-Host "Installing IsolatedUserMode feature ..."
        $result1 = Enable-WindowsOptionalFeature -Online -FeatureName "IsolatedUserMode" -all -norestart -verbose
        if ($result1){
            Write-Host "Installation of IsolatedUserMode feature succeeded"
            Write-EventLog -LogName System -Source "ANSSI.W10.VSM" -EntryType Information -Message "IsolatedUserMode feature
installation succeeded" -EventId 64203
        } else {
            Write-Host "Installation of IsolatedUserMode feature failed"
            Write-EventLog -LogName System -Source "ANSSI.W10.VSM" -EntryType Information -Message "IsolatedUserMode feature
installation failed: $result" -EventId 64204
        }
    }
    # Installation de la fonctionnalité Windows Microsoft-Hyper-V-Hypervisor si nécessaire
    if (-not ($HyperVInstallState -eq 1)){
        Write-Host "Installing Microsoft-Hyper-V-Hypervisor feature ..."
        $result2 = Enable-WindowsOptionalFeature -Online -FeatureName "Microsoft-Hyper-V-Hypervisor" -all -norestart -verbose
        if ($result2){
            Write-Host "Installation of Microsoft-Hyper-V-Hypervisor feature succeeded"
            Write-EventLog -LogName System -Source "ANSSI.W10.VSM" -EntryType Information -Message "VMware feature installation
succeeded" -EventId 64205
        } else {
            Write-Host "Installation of Microsoft-Hyper-V-Hypervisor feature failed"
            Write-EventLog -LogName System -Source "ANSSI.W10.VSM" -EntryType Information -Message "Microsoft-Hyper-V-Hypervisor
feature installation failed: $result" -EventId 64206
        }
    }
} catch {
    Write-Host "Microsoft Hyper-V Hypervisor and IsolatedUserMode features installation raised an error: $_.Exception.Message"
    Write-EventLog -LogName System -Source "ANSSI.W10.VSM" -EntryType Information -Message "Microsoft Hyper-V Hypervisor and
IsolatedUserMode features installation raised an error: $_.Exception.Message" -EventId 64207
}
```

Exemple B.2 – Script PowerShell d'installation des fonctionnalités de VBS

Annexe C

Méthodes d'identification des postes de travail cibles

L'activation par GPO des fonctionnalités de VBS sur un équipement qui ne satisfait pas aux prérequis matériels et logiciels cause des erreurs lors de l'application de la GPO. Ces erreurs sont non bloquantes pour la mise en œuvre des stratégies applicables, mais auront pour effet négatif de polluer les journaux systèmes des équipements en question, tout en ralentissant l'application de la GPO. Il est alors préférable de n'appliquer les GPO relatives aux fonctionnalités de VBS que sur les postes qui satisfont aux prérequis matériels et logiciels. Pour cela, il est tout d'abord nécessaire de pouvoir les identifier et les recenser.

La première solution consiste à créer des groupes d'ordinateurs dans l'annuaire *Active Directory* qui ne contiennent que les équipements compatibles avec ces fonctionnalités, et de n'appliquer les GPO qu'à ces groupes d'ordinateurs. Ces groupes peuvent, par exemple, être peuplés automatiquement depuis les données remontées par des logiciels d'inventaire de parc, ou bien manuellement. Lorsque la méthode manuelle est retenue, chaque entité procédera en fonction de son inventaire de parc, de ses processus organisationnels et de sa stratégie de déploiement. Il s'agit de la solution à privilégier dans la mesure du possible.

La deuxième solution consiste à procéder par filtres WMI, scripts ou « ciblage au niveau de l'élément » de stratégies de préférence. L'objectif est ainsi d'appliquer les GPO de configuration de ces mécanismes aux postes techniquement compatibles sans avoir à les identifier au préalable. Il s'agit du cas illustré dans cette annexe.

C.1 Cibles de la GPO d'installation du VSM

Pour cibler les postes de travail compatibles avec le VSM, et installer les fonctionnalités Windows nécessaires (*IsolatedUserMode* et *Microsoft-Hyper-V-Hypervisor*) uniquement sur les équipements adéquats, un filtre WMI contenant les requêtes C.1 et C.2 sur l'espace de noms par défaut `root\CIMv2` peut être utilisé, dans la mesure où l'exécution des requêtes est instantanée et ne ralentit pas l'application des GPO.

```
Select Caption from Win32_OperatingSystem where  
Caption like "Microsoft_Windows_10_Enterprise" or  
Caption like "Microsoft_Windows_10_Enterprise"
```

Exemple C.1 – Requête WMI : Postes en Windows 10 édition Entreprise



Information

Les conditions sur la propriété `Caption` ici utilisées sont valables pour des systèmes en langues anglaise ou française. Pour supporter d'autres langues, il est nécessaire de compléter cette requête WMI.

```
Select Caption from Win32_Processor where  
(Address = 64 and VirtualizationFirmwareEnabled = True and  
SecondLevelAddressTranslationExtensions = True)
```

Exemple C.2 – Requête WMI : CPU 64 bits avec prise en charge de Intel-VT ou AMD-V et de SLAT

Le filtre WMI illustré en figure C.1 met en œuvre les deux requêtes ci-dessus et permet d'appliquer des GPO seulement sur les postes qui satisfont à l'ensemble des critères suivants :

- le système d'exploitation est un système 64 bits en édition « Entreprise » ;
- le matériel prend en charge Intel-VT ou AMD-V, ainsi que le SLAT.

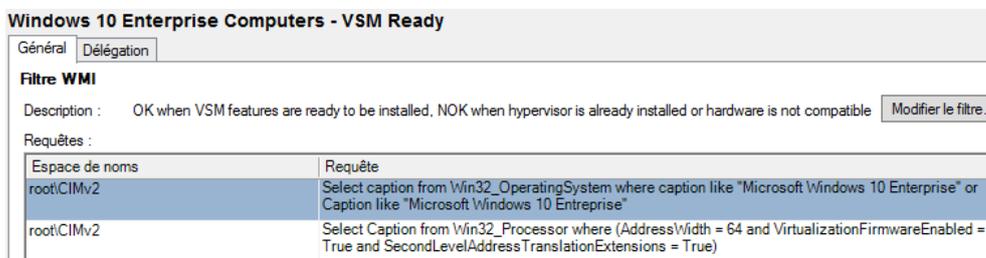


FIGURE C.1 – Filtre WMI de ciblage des postes compatibles avec le VSM



Information

Une fois qu'un hyperviseur est présent sur le système, les propriétés `SecondLevelAddressTranslationExtensions` et `VirtualizationFirmwareEnabled` ont pour valeur `False`, bien que le processeur supporte ces jeux d'instructions. En utilisant les deux filtres WMI ci-dessus, une GPO d'installation de ces fonctionnalités Windows ne sera donc plus appliquée lorsque ces fonctionnalités auront été installées.



Information

Il n'est pas possible de vérifier les autres prérequis (UEFI, démarrage sécurisé, etc.) par filtre WMI en utilisant les classes WMI natives. Des tests complémentaires doivent donc être réalisés par script ou par ciblage de stratégies de préférence, par exemple. Ces méthodes sont abordées dans la section C.2 suivante.

C.2 Cibles des GPO de mise en œuvre de Device Guard et de Credential Guard

Il est intéressant de noter que la classe WMI `Win32_DeviceGuard` a fait son apparition sous Windows 10 (voir [annexe E](#)), néanmoins cette classe ne peut pas être utilisée pour élaborer des filtres WMI. Le ciblage des postes compatibles *Device Guard* ou *Credential Guard* s'avère donc délicat à réaliser, aucune solution simple n'étant proposée par le système au jour de publication de ce guide. Plusieurs méthodes peuvent alors être imaginées pour réaliser ce ciblage en deux temps :

1. L'enregistrement des propriétés de la classe WMI `Win32_DeviceGuard` (ou d'un équivalent) sous une forme utilisable par filtre WMI ou par « ciblage au niveau de l'élément » de stratégies de préférence. Ceci pourrait par exemple consister en :
 - a) la construction d'une classe WMI personnalisée identique à la classe `Win32_DeviceGuard` mais avec les propriétés sous forme de chaînes de caractères au lieu de tableaux d'entiers, ainsi utilisable par des filtres WMI. Cette classe WMI peut elle aussi être créée par script PowerShell et être intégrée à des scripts de démarrage existants, ou bien faire l'objet d'un script de démarrage dans une GPO spécifique (voir section [C.2.1](#));
 - b) des commandes PowerShell qui vont stocker ces propriétés WMI dans des clés en base de registre. Ces commandes peuvent alors être intégrées à des scripts de démarrage existants (tel que le script PowerShell de l'exemple [B.1](#)) ou bien faire l'objet d'un script de démarrage dans une GPO spécifique (voir section [C.2.2](#));
 - c) l'utilisation de la clé de registre créée par le script de diagnostic PowerShell présenté dans l'[annexe A](#) et préalablement exécuté sur les postes de travail (par GPO ou lors de son provisionnement).
2. L'utilisation de ces informations non natives pour l'application des GPO relatives à *Device Guard* ou *Credential Guard* sur le seul périmètre des postes compatibles (par filtre WMI pour la classe WMI personnalisée ou par « ciblage au niveau de l'élément » de stratégies de préférence pour la clé de registre).

Les exemples a) et b) sont illustrés dans les sections [C.2.1](#) et [C.2.2](#) suivantes.

C.2.1 Construction d'une classe WMI personnalisée

L'exemple de script PowerShell [C.3](#) suivant illustre une méthode possible de création d'une classe WMI personnalisée `Win32_DeviceGuard`, contenant les valeurs sous forme de chaînes de caractères plutôt que de tableaux d'entiers (ainsi utilisables dans des filtres WMI).

```

# Script de création d'une classe WMI Win32_DeviceGuard à l'identique de l'originale mais avec des propriétés sous
# forme de chaînes de caractères plutôt que de tableaux d'entiers afin de pouvoir utiliser ces valeurs dans des filtres WMI.
#
# Nom de la classe et de l'espace de noms (à personnaliser avant application)
$MyClass = "Win32_CustomDeviceGuard"
$MyNameSpace = "root\ANSSI"
$TheDate = Get-Date -format "yyyy/MM/dd hh:mm:ss"
# Création de l'espace de noms s'il n'existe pas déjà
Try{
    write-host "Checking Namespace $($MyNameSpace) ..."
    if (!(Get-WmiObject -NameSpace $MyNameSpace -Class "__namespace" | select-string ($MyNameSpace -replace
        "^(.*?)\\(.*?)", "$2"))){
        $NS = ([WMICLASS]"\\.\$($MyNameSpace.split("\")[0]:__Namespace").CreateInstance()
        $NS.name = ($MyNameSpace -replace "^(.*?)\\(.*?)", "$2")
        $NS.put()
        write-host "Namespace $($MyNameSpace) created successfully"
    } else {write-host "Namespace $($MyNameSpace) already exists"}
} catch {
    write-host "An error occured when creating the namespace $($MyNameSpace): $_.Exception.Message"
    exit
}
# Création de la classe si elle n'existe pas déjà
[wmiclass]$WMIClass = ""
Try{
    write-host "Checking Class $($MyClass) ..."
    if (!(Get-WmiObject -NameSpace $MyNameSpace -Class $MyClass -list)){
        $WMIClass = New-Object System.Management.ManagementClass($MyNameSpace, [string]::Empty, $null)
        $WMIClass.Name = $MyClass
        $WMIClass.Qualifiers.add("Static", $true)
        $WMIClass.Qualifiers.add("Description", "Win32_DeviceGuard clone with string properties instead of arrays")
        write-host "Class $($MyClass) created successfully"
        # Création des propriétés de la classe
        Try{
            Write-Host "Updating Class properties ..."
            (Get-WmiObject -NameSpace "root/Microsoft/Windows/DeviceGuard" -Query 'Select * from Win32_DeviceGuard').Properties
            | foreach {
                $WMIClass.Properties.add($_.Name, ([string]$_ .Value).replace("_", ","))
            }
            $WMIClass.Properties.add("LastUpdateDate", $TheDate)
            write-host "Updating the key property qualifier ..." $_.Name
            $WMIClass.Properties["InstanceIdentifier"].Qualifiers.Add("Key", $true)
            $WMIClass.Put() | out-null
            write-host "Class properties and key property qualifier saved successfully"
        } catch {
            write-host "An error occured when updating Class $($MyClass) properties and key property qualifier: $_.Exception.Message"
            exit
        }
    } else {
        $WMIClass = Get-WmiObject -NameSpace $MyNameSpace -Class $MyClass -list
        write-host "Class $($MyClass) already exists. Skipping ..."
    }
} catch {
    write-host "An error occured when creating WMI Class $($MyClass): $_.Exception.Message"
    exit
}
# Création ou mise à jour des propriétés de l'instance de la classe
Try{
    if (($WMIClass.GetInstances()).Count -eq 1){
        write-host "Deleting existing Instance of the Class"
        $WMIInstance = $WMIClass.GetInstances()[0].delete()
    }
    write-host "Creating new Instance of the Class"
    $WMIInstance = $WMIClass.CreateInstance()
    (Get-WmiObject -NameSpace "root/Microsoft/Windows/DeviceGuard" -Query 'Select * from Win32_DeviceGuard').Properties |
    foreach {
        $WMIInstance.SetPropertyValue($_.Name, ([string]$_ .Value).replace("_", ","))
    }
    $WMIInstance.SetPropertyValue("LastUpdateDate", $TheDate)
    $WMIInstance.Put() | out-null
    write-host "Instance of the Class saved"
} catch {
    write-host "An error occured when updating the Class Instance: $_.Exception.Message"
    exit
}
}

```

Exemple C.3 – Création d'une classe WMI personnalisée



Information

Le temps d'exécution du script PowerShell d'exemple C.3 est insignifiant. Il est donc tout à fait envisageable de l'exécuter sur l'ensemble du parc des postes de travail installés en Windows 10, éditions « Entreprise » ou « Éducation », sans réelle conséquence sur les performances. Il reste néanmoins recommandé de procéder à des tests représentatifs avant toute exécution en production.

Les propriétés de la classe WMI personnalisée (créée par le script d'exemple C.3) étant enregistrées sous formes de chaînes de caractères, il devient possible de les utiliser dans des filtres WMI. La figure C.2 suivante illustre les différences entre la classe WMI d'origine et la classe personnalisée nouvellement créée.

```
PS C:\> get-wmiobject -namespace "root\ANSSI" -Query 'Select * from win32_CustomDeviceGuard'

__GENUS                : 2
__CLASS                : Win32_CustomDeviceGuard
__SUPERCLASS           :
__DYNASTY              : Win32_CustomDeviceGuard
__RELPATH              : Win32_CustomDeviceGuard.InstanceIdentifier="4ff40742-2649-41b8-bdd1-e80fad1cce80"
__PROPERTY_COUNT       : 10
__DERIVATION           : {}
__SERVER               : PAW02
__NAMESPACE            : root\ANSSI
__PATH                : \\PAW02\root\ANSSI:Win32_CustomDeviceGuard.InstanceIdentifier="4ff40742-2649-41b8-bdd1-e80fad1cce80"
AvailableSecurityProperties : {1, 2}
CodeIntegrityPolicyEnforcementStatus : 0
InstanceIdentifier        : 4ff40742-2649-41b8-bdd1-e80fad1cce80
LastUpdateDate            : 2017/08/07 14:45:10
RequiredSecurityProperties : {1, 2}
SecurityServicesConfigured : {1, 2}
SecurityServicesRunning   : 0
UsermodeCodeIntegrityPolicyEnforcementStatus : 0
Version                   : 1.0
VirtualizationBasedSecurityStatus : 2
PSComputerName            : PAW02

PS C:\> get-wmiobject -namespace "root\Microsoft\Windows\DeviceGuard" -Query 'Select * from win32_DeviceGuard'

__GENUS                : 2
__CLASS                : Win32_DeviceGuard
__SUPERCLASS           :
__DYNASTY              : Win32_DeviceGuard
__RELPATH              : Win32_DeviceGuard.InstanceIdentifier="4ff40742-2649-41b8-bdd1-e80fad1cce80"
__PROPERTY_COUNT       : 9
__DERIVATION           : {}
__SERVER               : PAW02
__NAMESPACE            : root\Microsoft\Windows\DeviceGuard
__PATH                : \\PAW02\root\Microsoft\Windows\DeviceGuard:Win32_DeviceGuard.InstanceIdentifier="4ff40742-2649-41b8-bdd1-e80fad1cce80"
AvailableSecurityProperties : {1, 2}
CodeIntegrityPolicyEnforcementStatus : 0
InstanceIdentifier        : 4ff40742-2649-41b8-bdd1-e80fad1cce80
RequiredSecurityProperties : {1, 2}
SecurityServicesConfigured : {1, 2}
SecurityServicesRunning   : {0}
UsermodeCodeIntegrityPolicyEnforcementStatus : 0
Version                   : 1.0
VirtualizationBasedSecurityStatus : 2
PSComputerName            : PAW02
```

FIGURE C.2 – Classe WMI Win32_DeviceGuard d'origine et classe personnalisée

L'avantage de cette méthode est qu'elle permet d'utiliser les modèles d'administration pour configurer les fonctionnalités de VBS. Il ne reste alors plus qu'à créer les filtres WMI adéquats (sur l'espace de noms de la classe WMI personnalisée, soit « root\ANSSI » dans notre exemple) pour cibler les postes de travail compatibles avec *Device Guard* ou *Credential Guard* et prenant en charge ou non les protections DMA (se référer aux valeurs des différentes propriétés de la classe WMI en annexe E) :

```
Select AvailableSecurityProperties from Win32_CustomDeviceGuard
where AvailableSecurityProperties like "%1%" AND
AvailableSecurityProperties like "%2%" AND
AvailableSecurityProperties like "%3%"
```

Exemple C.4 – Requête WMI : Prise en charge de DG/CG avec protection DMA

```
Select AvailableSecurityProperties from Win32_CustomDeviceGuard
where AvailableSecurityProperties like "%1%" AND
AvailableSecurityProperties like "%2%" AND NOT
AvailableSecurityProperties like "%3%"
```

Exemple C.5 – Requête WMI : Prise en charge de DG/CG sans protection DMA

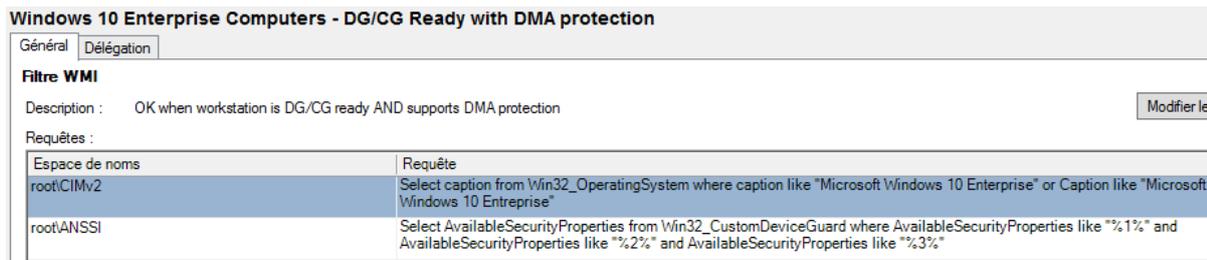


FIGURE C.3 – Filtre WMI de ciblage des postes qui prennent en charge de DG/CG avec protection DMA

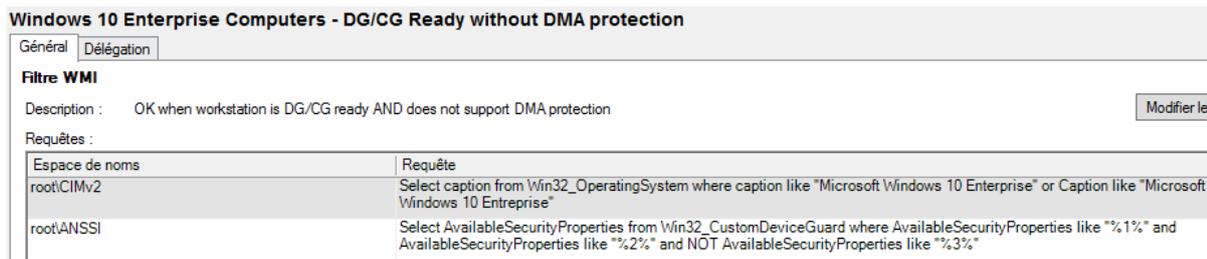


FIGURE C.4 – Filtre WMI de ciblage des postes qui prennent en charge de DG/CG sans protection DMA

C.2.2 Copie de la class WMI Win32_DeviceGuard en base de registres

L'exemple de script PowerShell C.6 suivant illustre une méthode possible de copie en base de registre de la classe WMI Win32_DeviceGuard.

```
(Get-WMIObject -Namespace "root/Microsoft/Windows/DeviceGuard" -Query 'Select * from Win32_DeviceGuard').Properties | foreach {
  Invoke-Expression "REG_ADD \"HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Win32DeviceGuard\" /v $_.Name /t REG_SZ /d
    ([string]$_ . Value) . Replace(\" \", \" \") /f'
}
```

Exemple C.6 – Copie de la class WMI Win32_DeviceGuard en base de registres

Les propriétés de cette classe WMI étant ainsi stockées en base de registres, il devient possible de les utiliser en « ciblage au niveau de l'élément » de stratégies de préférence, par le biais d'une correspondance de sous-chaîne de valeur de registre (voir [10]). L'inconvénient de cette méthode est qu'elle nécessite de configurer les fonctionnalités de VBS par valeurs en clé de registre, et ne permet pas d'utiliser les modèles d'administration.



Information

Le temps d'exécution du script PowerShell d'exemple C.6 est insignifiant y compris lorsque la classe WMI Win32_DeviceGuard n'existe pas. Il est donc tout à fait envisageable de l'exécuter sur l'ensemble du parc sans réelle conséquence sur les performances. Il reste néanmoins recommandé de procéder à des tests préalables représentatifs avant toute exécution en production.

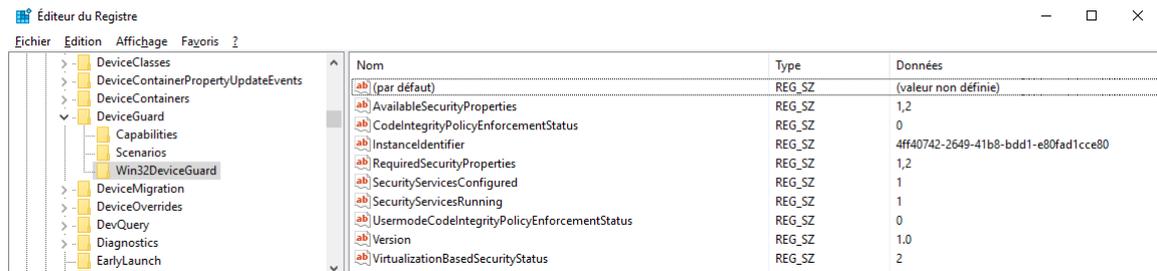


FIGURE C.5 – Résultat de la copie de la class WMI Win32_DeviceGuard en base de registres

L'exécution de ce script PowerShell peut être réalisée par GPO tel qu'illustré en annexe B pour l'installation des fonctionnalités du VSM.

Annexe D

Désactivation après verrouillage UEFI

Une fois que *Device Guard* ou *Credential Guard* ont été activés sur des postes de travail avec verrouillage UEFI, la désactivation de ces fonctionnalités de sécurité n'est plus possible à distance, elle nécessite une intervention physique. Cette désactivation doit être réalisée en deux étapes.

D.1 Suppression du verrouillage UEFI

La première étape de désactivation consiste à exécuter, sur les postes de travail, les exemples de scripts Batch D.1 et D.2 suivants avec les privilèges d'administrateur local :

```
mountvol X :/s
copy %WINDIR%\System32\SecConfig.efi X :EFI\Microsoft\Boot\SecConfig.efi /Y
bcdedit /create {0cb3b571-2f2e-4343-a879-d86a476d7215} /d "DebugTool" /application osloader
bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} path "\EFI\Microsoft\Boot\SecConfig.efi"
bcdedit /set {bootmgr} bootsequence {0cb3b571-2f2e-4343-a879-d86a476d7215}
bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} loadoptions DISABLE-LSA-ISO
bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} device partition=X :
mountvol X :/d
```

Exemple D.1 – Script batch de désactivation du verrouillage UEFI de CG

```
mountvol X :/s
copy %WINDIR%\System32\SecConfig.efi X :EFI\Microsoft\Boot\SecConfig.efi /Y
bcdedit /create {0cb3b571-2f2e-4343-a879-d86a476d7215} /d "DebugTool" /application osloader
bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} path "\EFI\Microsoft\Boot\SecConfig.efi"
bcdedit /set {bootmgr} bootsequence {0cb3b571-2f2e-4343-a879-d86a476d7215}
bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} loadoptions DISABLE-VBS
bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} device partition=X :
mountvol X :/d
```

Exemple D.2 – Script batch de désactivation du verrouillage UEFI de DG

Ces scripts modifient la séquence de démarrage de l'équipement et y enlèvent le verrouillage UEFI de *Credential Guard* ou *Device Guard*. Les deux désactivations peuvent être opérées en un seul et unique script exécutant les deux lignes contenant l'instruction `loadoptions` l'une après l'autre. Ces scripts peuvent tout à fait être exécutés sur les postes de travail sous la forme de scripts de démarrage par GPO.

D.2 Validation de la désactivation par intervention physique

La deuxième étape consiste à redémarrer le poste de travail pour valider cette désactivation. Au cours de ce redémarrage, et avant même le chargement de Windows, une invite sera présentée à l'utilisateur demandant confirmation de la désactivation (respectivement *Do you want to disable Credential Guard ?* et *Do you want to disable VBS ?*). L'utilisateur peut alors confirmer (touche F3) ou

annuler (touche ESC) la désactivation, ce dernier choix étant automatique après un certain délai sans réponse. La validation ne peut s'effectuer que physiquement sur le poste de travail puisqu'elle intervient en cours de séquence de démarrage de l'équipement. Toute désactivation à distance est donc impossible sans cette intervention physique dès lors qu'un verrouillage UEFI a été demandé.

Annexe E

La classe WMI Win32_DeviceGuard

Sous Windows 10, éditions « Entreprise » et « Éducation », une classe WMI Win32_DeviceGuard a fait son apparition. Celle-ci se trouve dans l'espace de noms root\Microsoft\Windows\DeviceGuard. Les principales propriétés de cette classe sont énumérées dans le tableau E.1.

Propriété et description	Valeurs
AvailableSecurityProperties Tableau d'entiers indiquant les propriétés de sécurité disponibles	0 si la virtualisation de base n'est pas prise en charge
	1 si la virtualisation de base est prise en charge
	2 si le démarrage sécurisé est activé
	3 si la protection DMA est disponible
	4 si Secure MOR est pris en charge
	5 si la protection NX est prise en charge
RequiredSecurityProperties Tableau d'entiers indiquant les propriétés de sécurité requises par configuration	0 si rien n'est requis
	1 si la virtualisation de base est requise
	2 si le démarrage sécurisé est requis
	3 si la protection DMA est requise
	4 si Secure MOR est requis
	5 si la protection NX est requise
SecurityServicesConfigured Tableau d'entiers indiquant quelles fonctionnalités de sécurité sont configurées	0 lorsqu'aucune fonctionnalité n'est configurée
	1 lorsque CG est configuré
	2 lorsque HVCI est configuré
SecurityServicesRunning Tableau d'entiers indiquant quelles fonctionnalités de sécurité sont actives	0 lorsqu'aucune fonctionnalité n'est active
	1 lorsque CG est actif
	2 lorsque HVCI est active
VirtualizationBasedSecurityStatus Entier indiquant le status de la VBS	0 lorsque la VBS est active
	1 lorsque la VBS est configurée mais pas active
	2 lorsque la VBS est active

Tableau E.1 – Classe WMI Win32_DeviceGuard



Information

Il n'est pas possible de réaliser une condition **where** sur un tableau au sein d'une requête WMI.

Annexe F

GPO de mise en œuvre des fonctionnalités de VBS

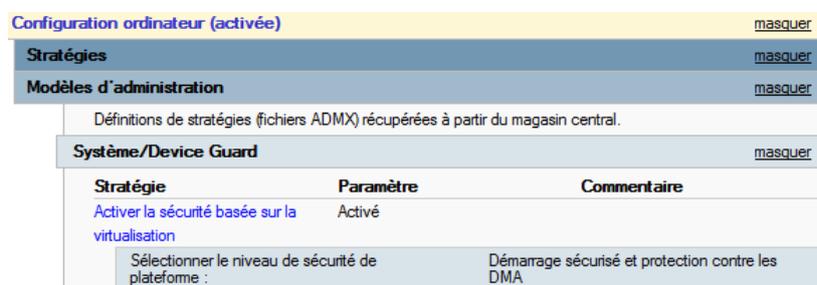
F.1 Téléchargement des modèles d'administration de Windows 10

Pour pouvoir activer et configurer les fonctionnalités de VBS par GPO, il est nécessaire de préalablement télécharger les modèles d'administration de Windows 10 fournis par Microsoft (téléchargeables depuis [17] et déjà présents sous Windows Server 2016). Dans un scénario de déploiement en domaine Active Directory, ces modèles ADMX doivent être placés dans un dépôt central au sein du dossier SYSVOL présent sur les contrôleurs de domaine et contenant les GPO. Pour plus d'informations, un guide pas à pas de gestion des modèles ADM et ADMX est disponible sur le site technet de Microsoft (voir [8]).

Grâce aux modèles d'administration de Windows 10, les différentes fonctionnalités de VBS se configurent simplement par GPO comme illustré dans les sections qui suivent.

F.2 GPO d'activation de la VBS seule

La VBS se configure comme illustré par la figure F.1 suivante, que *Credential Guard* ou HVCI (voir section 3.4.2) soient activés ou non. Notons que conformément aux prérequis, le démarrage sécurisé (idéalement avec protection DMA) doit également être configuré.

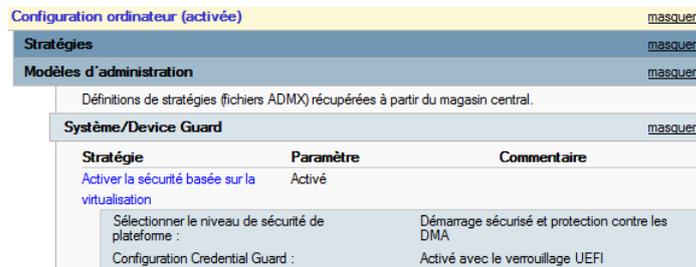


Configuration ordinateur (activée)		
Stratégies		
Modèles d'administration		
Définitions de stratégies (fichiers ADMX) récupérées à partir du magasin central.		
Système/Device Guard		
Stratégie	Paramètre	Commentaire
Activer la sécurité basée sur la virtualisation	Activé	
Sélectionner le niveau de sécurité de plateforme :		Démarrage sécurisé et protection contre les DMA

FIGURE F.1 – GPO de configuration de la VBS seule

F.3 GPO d'activation de Credential Guard

Credential Guard se configure comme illustré par la figure F.2 suivante pour une activation avec verrouillage UEFI. Notons que conformément aux prérequis, le démarrage sécurisé (idéalement avec protection DMA) doit également être configuré.



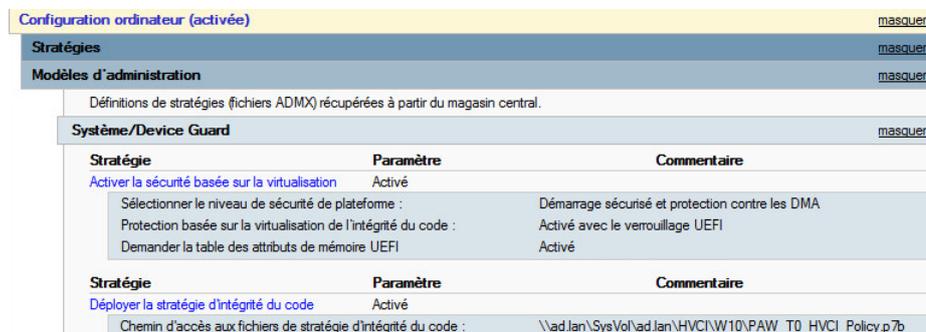
The screenshot shows the Group Policy Configuration console for 'Configuration ordinateur (activée)'. It is expanded to 'Système/Device Guard'. The 'Activer la sécurité basée sur la virtualisation' strategy is active. Below it, the 'Sélectionner le niveau de sécurité de plateforme' parameter is active, with a comment: 'Démarrage sécurisé et protection contre les DMA'. The 'Configuration Credential Guard' parameter is also active, with a comment: 'Activé avec le verrouillage UEFI'.

Stratégie	Paramètre	Commentaire
Activer la sécurité basée sur la virtualisation	Activé	
Sélectionner le niveau de sécurité de plateforme :	Activé	Démarrage sécurisé et protection contre les DMA
Configuration Credential Guard :	Activé	Activé avec le verrouillage UEFI

FIGURE F.2 – GPO de configuration de *Credential Guard*

F.4 GPO d'activation de Device Guard

HVCI se configure comme illustré par la figure F.3 suivante pour une activation avec verrouillage UEFI. Notons que conformément aux prérequis, le démarrage sécurisé (idéalement avec protection DMA) doit également être configuré. Cette figure illustre également le déploiement d'une stratégie d'intégrité du code configurable au format .p7b²⁷.



The screenshot shows the Group Policy Configuration console for 'Configuration ordinateur (activée)'. It is expanded to 'Système/Device Guard'. The 'Activer la sécurité basée sur la virtualisation' strategy is active. Below it, three parameters are active: 'Sélectionner le niveau de sécurité de plateforme' (comment: 'Démarrage sécurisé et protection contre les DMA'), 'Protection basée sur la virtualisation de l'intégrité du code' (comment: 'Activé avec le verrouillage UEFI'), and 'Demander la table des attributs de mémoire UEFI' (comment: 'Activé'). Below this, the 'Déployer la stratégie d'intégrité du code' strategy is active, with the 'Chemin d'accès aux fichiers de stratégie d'intégrité du code' parameter set to '\\ad.lan\SysVol\ad.lan\HVC\W10\PAW_T0_HVCI_Policy.p7b'.

Stratégie	Paramètre	Commentaire
Activer la sécurité basée sur la virtualisation	Activé	
Sélectionner le niveau de sécurité de plateforme :	Activé	Démarrage sécurisé et protection contre les DMA
Protection basée sur la virtualisation de l'intégrité du code :	Activé	Activé avec le verrouillage UEFI
Demander la table des attributs de mémoire UEFI	Activé	Activé
Déployer la stratégie d'intégrité du code	Activé	
Chemin d'accès aux fichiers de stratégie d'intégrité du code :		\\ad.lan\SysVol\ad.lan\HVC\W10\PAW_T0_HVCI_Policy.p7b

FIGURE F.3 – GPO de configuration de DG

27. Le format .p7b désigne un fichier au format ASCII encodé en base64 et utilisé comme conteneur de certificats ou de chaînes de certificats. Seules des clés publiques y figurent.

Annexe G

Contrôle de conformité de l'IGC UEFI

Cette annexe aborde la problématique du contrôle de conformité d'une IGC UEFI pour des entités qui ont des besoins de sécurité élevés et pour lesquelles la confiance dans la chaîne de démarrage UEFI est une préoccupation forte. Le présent document n'a pas vocation à traiter cette problématique de manière exhaustive, simplement à sensibiliser le lecteur et à proposer quelques éléments de réflexion sur le sujet.

G.1 Particularités de la Platform Key (PK)

Les clés de l'IGC UEFI sont stockées dans des variables UEFI en mémoire non volatile. Ces variables ne peuvent être mises à jour qu'en étant en possession de la partie privée de la clé utilisée pour les signer (ou lorsque le matériel est en mode *setup*).

La PK signe les variables UEFI de stockage des *Key Exchange Key* (KEK), qui elles-mêmes signent les variables UEFI de stockage des clés et empreintes cryptographiques en DB et DBX. Si la partie privée de la PK d'un poste de travail venait à être divulguée à un individu malveillant, ce dernier pourrait l'utiliser pour signer du code malveillant. Ce code serait alors considéré de confiance et pourrait être injecté dans la chaîne de démarrage UEFI du poste de travail en question. L'attaquant obtiendrait ainsi du code malveillant persistant chargé à bas niveau et difficilement détectable au niveau du système, ce qui est intéressant dans le cadre d'une attaque évoluée. En résumé, la divulgation de la partie privée de la PK d'un ordinateur implique la capacité d'exécuter du code arbitraire sur ce dernier.

En fonction du fabricant, la PK peut être unique par ordinateur, par modèle, par gamme ou être plus simplement identique sur tout ordinateur produit par le fabricant. En cas de divulgation de la partie privée d'une PK, c'est donc potentiellement tous les ordinateurs d'un même modèle, d'une même gamme ou tout ordinateur produit par le fabricant qui peuvent être directement concernés et potentiellement compromis. Or, par simplicité, les fabricants ont plutôt tendance à utiliser une seule et même PK pour l'ensemble de leurs chaînes de production.

Cela pose donc un réel problème de confiance envers les fabricants ainsi que dans la sécurité du sequestre de la partie privée de ces PK.

Bien que l'utilisation d'une PK unique par modèle d'ordinateur est le compromis que recommande²⁸ Microsoft aux fabricants, cela pourrait s'avérer insuffisant pour des clients ayant des contraintes de sécurité fortes.

28. Consulter le guide de création et de gestion des clés de démarrage sécurisé de Microsoft [22].

G.2 Comparaison à une base de référence

G.2.1 Clés de l'IGC UEFI

Le tableau G.1 ci-dessous liste²⁹, à titre indicatif, les clés de vérification de signatures numériques qui devraient normalement être présentes dans une IGC UEFI. Les postes de travail et serveurs qui ne sont pas équipés par un OEM avec un système d'exploitation Microsoft Windows peuvent ne pas contenir les clés des autorités de certification de Microsoft. Ils devraient dans ce cas contenir des clés d'éditeurs tiers pour, par exemple, autoriser des *bootloaders* Linux. En tout état de cause, toute déviance par rapport à cette liste de référence devrait faire l'objet d'une investigation.

Type	Nom commun	Empreinte SHA1.
PK	Clé PK du fabricant du matériel	Dépend du fabricant
KEK	Microsoft Corporation KEK CA 2011 (nécessaire pour autoriser les chargeurs de démarrage de Microsoft Windows)	31590bfd89c9d74ed087dfac66334b3931254b30
KEK	AC KEK du fabricant du matériel	Dépend du fabricant
DB	Microsoft Windows Production PCA 2011	580a6f4cc4e4b669b9ebdc1b2b3e087b80d0678d
DB	Microsoft Corporation UEFI CA 2011	46def63b5ce61cf8ba0de2e6639c1019d0ed14f3
DB	Clé(s) du fabricant	Dépend du fabricant

Tableau G.1 – Clés de vérification de signatures numériques normalement présentes dans l'UEFI d'un équipement livré avec Microsoft Windows

G.2.2 Extraction des clés de l'IGC UEFI

L'extraction des clés peut être réalisée en PowerShell. Les scripts d'exemple G.1, G.2 et G.3 permettent l'extraction des différentes clés mentionnées dans le tableau G.1 au format X509.

```
Write-Host "Extracting KEK"
$i_StartByte = 0
While ($i_StartByte -ne (Get-SecureBootUEFI -name KEK).Bytes.Count)
{
    Write-Host "Extracting KEK cert from Byte" $i_StartByte
    $nextcerthex = (Get-SecureBootUEFI -name KEK).Bytes[($i_StartByte + 22)..($i_StartByte + 16)]
    $nextcerthex = $nextcerthex | ForEach-Object {"0" + [String]::Format("{0:x}", $_) | }
    ForEach-Object {$_.substring($_.length - 2)}
    $i_NextCertByte = [convert]::ToInt32($nextcerthex -join "",16)
    [io.file]::WriteAllBytes(("c:\scripts\uefi\KEK_" + $i_StartByte + ".der"),(Get-SecureBootUEFI
-name KEK).Bytes[($i_StartByte+44)..($i_NextCertByte + $i_StartByte - 1)])
    $i_StartByte = $i_NextCertByte + $i_StartByte
}
```

Exemple G.1 – Extraction des clés KEK de l'IGC UEFI

29. L'utilisation de l'algorithme de hachage cryptographique SHA1 est déconseillée étant donné qu'il est sensible à la collision. Néanmoins, Microsoft continue d'utiliser SHA1 pour calculer l'empreinte numérique des certificats affichés visuellement sous Windows. Par simplicité pour le lecteur, c'est donc l'empreinte SHA1 qui figure dans le tableau G.1.

```

Write-Host "Extracting PK"
$_StartByte = 0
While ($_StartByte -ne (Get-SecureBootUEFI -name PK).Bytes.Count)
{
    Write-Host "Extracting PK cert from byte" $_StartByte
    $nextcerthex = (Get-SecureBootUEFI -name PK).Bytes[$_StartByte + 22..($_StartByte + 16)]
    $nextcerthex = $nextcerthex | ForEach-Object {"0" + [String]::Format("{0:x}",$_)} |
ForEach-Object {$_ . substr($_.length - 2)}
    $_NextCertByte = [convert]::ToInt32($nextcerthex -join " ",16)
    [io.file]::WriteAllBytes(("c:\scripts\uefi\PK_" + $_StartByte + ".der"),(Get-SecureBootUEFI
-name PK).Bytes[$_StartByte+44..($_NextCertByte + $_StartByte - 1)])
    $_StartByte = $_NextCertByte + $_StartByte
}

```

Exemple G.2 – Extraction des clés PK de l'IGC UEFI

```

Write-Host "Extracting DB"
$_StartByte = 0
While ($_StartByte -ne (Get-SecureBootUEFI -name DB).Bytes.Count)
{
    Write-Host "Extracting DB cert from byte" $_StartByte
    $nextcerthex = (Get-SecureBootUEFI -name DB).Bytes[$_StartByte + 22..($_StartByte + 16)]
    $nextcerthex = $nextcerthex | ForEach-Object {"0" + [String]::Format("{0:x}",$_)} |
ForEach-Object {$_ . substr($_.length - 2)}
    $_NextCertByte = [convert]::ToInt32($nextcerthex -join " ",16)
    [io.file]::WriteAllBytes(("c:\scripts\uefi\DB_" + $_StartByte + ".der"),(Get-SecureBootUEFI
-name DB).Bytes[$_StartByte+44..($_NextCertByte + $_StartByte - 1)])
    $_StartByte = $_NextCertByte + $_StartByte
}

```

Exemple G.3 – Extraction des clés DB de l'IGC UEFI

Lorsque le besoin de sécurité est élevé et que la confiance dans la chaîne de démarrage sécurisé UEFI est une préoccupation, ces clés devraient alors être contrôlées à la livraison des postes de travail de manière à s'assurer de la conformité de l'IGC UEFI d'origine. Cette IGC devrait ensuite être régulièrement comparée à une base de référence, et des alertes de sécurité devraient être remontées en cas de dérive. Cette vérification peut notamment être réalisée par un script Powershell exécuté par GPO.

G.3 Limites d'un contrôle de conformité de l'IGC UEFI

Du fait d'une standardisation encore perfectible de l'UEFI, Microsoft propose un service de signature de *bootloaders* tiers, moyennant un enregistrement sur le portail pour développeurs³⁰, la signature d'un accord et une vérification d'identité. Ces *bootloaders* sont ensuite analysés par Microsoft, pour vérifier leur innocuité. *In fine*, Microsoft signe, avec son autorité de certification intégrée aux IGC UEFI, ce code d'éditeur tiers qui sera exécutable pendant la chaîne de démarrage.

Bien qu'un contrôle assidu du code soit attendu de Microsoft, on ne peut toutefois pas exclure qu'un code présentant une porte dérobée ou une vulnérabilité se retrouve signé (c'est d'ailleurs ce qui arrive régulièrement sur les magasins d'applications pour ordiphones). Un attaquant avec des moyens importants pourrait donc utiliser ce biais pour obtenir du code malveillant intégré légitimement à la chaîne de démarrage UEFI. L'attaquant disposerait ainsi d'un *rootkit* difficilement détectable.

Dans ce cas de figure, et si un tel code venait à être découvert, Microsoft aurait alors recours à la DBX de composants logiciels UEFI révoqués (voir section 2.4) pour interdire le composant logiciel malveillant. Les firmwares UEFI des postes de travail et des serveurs devraient ensuite être mis à jour pour actualiser leur DBX, ce qui dans la plupart des cas serait un processus relativement long.

30. Voir <https://sysdev.microsoft.com>.

Liste des recommandations

R1	Intégrer la compatibilité VSM/VBS dans le processus d'achat de matériel	8
R2	Protéger les accès à l'UEFI	8
R3	Maintenir à jour les composants logiciels UEFI	8
R4	Activer le démarrage sécurisé UEFI	10
R4*	Ignorer le contrôle de conformité de l'IGC UEFI	10
R4**	Mettre en œuvre du contrôle de conformité de l'IGC UEFI	10
R5	Activer la VBS sur tout poste de travail compatible	11
R6	Appliquer CCI en mode <i>enforced</i> sur les postes de travail sensibles	15
R7	Appliquer CCI sur les autres postes de travail	15
R7*	Appliquer CCI en mode audit sur les autres postes de travail	16
R7**	Appliquer CCI en mode <i>enforced</i> sur les autres postes de travail	16
R8	Mettre en œuvre HVCI sur les postes de travail compatibles	17
R9	Mettre en œuvre HVCI avec verrouillage UEFI	17
R10	Mettre en œuvre <i>Credential Guard</i>	22
R10*	Mettre en œuvre <i>Credential Guard</i> sur les postes de travail sensibles	22
R10**	Mettre en œuvre <i>Credential Guard</i> sur tous les postes de travail	22
R11	Assurer la sécurité physique des postes de travail	23
R12	Mettre en œuvre une solution de gestion des mots de passe administrateurs locaux	23
R13	Ne pas mémoriser les comptes à privilèges de l'AD	23
R14	Mettre en œuvre <i>Credential Guard</i> avec verrouillage UEFI	24
R15	Sensibiliser les utilisateurs au verrouillage UEFI de <i>Credential Guard</i>	24

Bibliographie

- [1] *Recommandations de sécurité relatives aux mots de passe.*
Note technique DAT-NT-001/ANSSI/SDE/NP v1.1, ANSSI, juin 2012.
<https://www.ssi.gouv.fr/mots-de-passe>.
- [2] *Recommandations de sécurité relatives à Active Directory.*
Note technique DAT-NT-017/ANSSI/SDE/NP v1.1, ANSSI, septembre 2014.
<https://www.ssi.gouv.fr/Active-Directory>.
- [3] *Recommandations pour la mise en œuvre d'une politique de restrictions logicielles sous windows.*
Note technique DAT-NT-013/ANSSI/SDE/NP v2.0, ANSSI, janvier 2017.
<https://www.ssi.gouv.fr/windows-restrictions-logicielles>.
- [4] *Achat de produits de sécurité et de services de confiance qualifiés.*
Guide Version 1.0, ANSSI, septembre 2014.
<https://www.ssi.gouv.fr/achat-rgs/>.
- [5] *Recommandations de sécurité relatives à la télé assistance.*
Note technique DAT-NT-004/ANSSI/SDE/NP v1.1, ANSSI, janvier 2017.
<https://www.ssi.gouv.fr/teleassistance>.
- [6] *Recommandations relatives à l'administration sécurisée des systèmes d'information.*
Note technique DAT-NT-022/ANSSI/SDE/NP v1.0, ANSSI, février 2015.
<https://www.ssi.gouv.fr/securisation-admin-si>.
- [7] *UEFI et bootkits PCI : le danger vient d'en bas.*
Pierre Chifflier.
Publication scientifique, ANSSI, juin 2013.
<https://www.ssi.gouv.fr/agence/publication/uefi-et-bootkits-pci-le-danger-vient-den-bas/>.
- [8] *Managing Group Policy ADMX Files Step-by-Step Guide.*
Technet, MICROSOFT, 2008.
<https://technet.microsoft.com/fr-fr/library/cc709647>.
- [9] *Ciblage au niveau de l'élément de préférence.*
Technet, MICROSOFT, 2009.
<https://technet.microsoft.com/fr-fr/library/cc733022>.
- [10] *Ciblage de correspondance dans le Registre.*
Technet, MICROSOFT.
[https://technet.microsoft.com/fr-fr/library/cc733051\(v=ws.11\).aspx](https://technet.microsoft.com/fr-fr/library/cc733051(v=ws.11).aspx).
- [11] *Hardware Security Testability Specification (anglais).*
Msdn, MICROSOFT, décembre 2016.
[https://msdn.microsoft.com/en-us/library/windows/hardware/mt712332\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/mt712332(v=vs.85).aspx).

- [12] *Kerberos Explained (anglais)*.
Msdn, MICROSOFT.
<https://msdn.microsoft.com/en-us/library/bb742516.aspx>.
- [13] *Local Administrator Password Solution (LAPS)*.
Microsoft download center, MICROSOFT, septembre 2016.
<https://www.microsoft.com/en-us/download/details.aspx?id=46899>.
- [14] *Secure MOR Implementation (anglais)*.
Msdn, MICROSOFT, juillet 2016.
<https://msdn.microsoft.com/windows/hardware/drivers/bringup/device-guard-requirements>.
- [15] *Documentation de référence sur la sécurisation de l'accès privilégié*.
Technet, MICROSOFT, décembre 2016.
<https://technet.microsoft.com/fr-fr/windows-server-docs/security/securing-privileged-access/securing-privileged-access-reference-material>.
- [16] *Mitigating Pass-the-Hash Attacks and Other Credential Theft, versions 1 and 2 (anglais)*.
Microsoft download center, MICROSOFT, juillet 2014.
<https://microsoft.com/en-us/download/details.aspx?id=36036>.
- [17] *Administrative Templates (.adm) for Windows 10 and Windows Server 2016, version 2.0*.
Microsoft download center, MICROSOFT, janvier 2017.
<https://www.microsoft.com/fr-FR/download/details.aspx?id=53430>.
- [18] *Device Guard and Credential Guard hardware readiness tool (anglais)*.
Microsoft download center, MICROSOFT, mai 2017.
<https://www.microsoft.com/en-us/download/details.aspx?id=53337>.
- [19] *PC OEM requirements for Device Guard and Credential Guard (anglais)*.
Msdn, MICROSOFT, mai 2017.
<https://msdn.microsoft.com/windows/hardware/commercialize/design/minimum/device-guard-and-credential-guard>.
- [20] *Driver compatibility with Device Guard in Windows 10 (anglais)*.
Msdn, MICROSOFT, mai 2015.
https://blogs.msdn.microsoft.com/windows_hardware_certification/2015/05/22/driver-compatibility-with-device-guard-in-windows-10/.
- [21] *Minimum hardware requirements for Windows 10 and Windows Server 2016 (anglais)*.
Msdn, MICROSOFT, juillet 2016.
[https://msdn.microsoft.com/en-us/library/windows/hardware/dn915086\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/dn915086(v=vs.85).aspx).
- [22] *Windows Secure Boot Key Creation and Management Guidance*.
Msdn, MICROSOFT, mai 2017.
<https://msdn.microsoft.com/fr-fr/windows/hardware/commercialize/manufacture/desktop/windows-secure-boot-key-creation-and-management-guidance>.
- [23] *Méthode WMSR*.
Msdn, MICROSOFT.
<https://msdn.microsoft.com/library/windows/hardware/ff561424>.

- [24] *Windows Defender Credential Guard : Requirements.*
Windows it center, MICROSOFT, mai 2017.
<https://docs.microsoft.com/en-us/windows/access-protection/credential-guard/credential-guard-requirements>.
- [25] *Deploy Device Guard : deploy code integrity policies.*
Windows it center, MICROSOFT, mai 2017.
<https://docs.microsoft.com/en-us/windows/device-security/device-guard/deploy-device-guard-deploy-code-integrity-policies>.
- [26] *Optional : Create a code signing certificate for code integrity policies.*
Windows it center, MICROSOFT, mai 2017.
<https://docs.microsoft.com/en-us/windows/device-security/device-guard/optional-create-a-code-signing-certificate-for-code-integrity-policies>.
- [27] *Deploy code integrity policies : steps.*
Windows it center, MICROSOFT, mai 2017.
<https://docs.microsoft.com/en-us/windows/device-security/device-guard/deploy-code-integrity-policies-steps>.
- [28] *Driver compatibility with Device Guard in Windows 10.*
Windows it center, MICROSOFT, avril 2015.
https://blogs.msdn.microsoft.com/windows_hardware_certification/2015/05/22/driver-compatibility-with-device-guard-in-windows-10/.
- [29] *Requirements and deployment planning guidelines for Windows Defender Device Guard.*
Windows it center, MICROSOFT, mai 2017.
<https://docs.microsoft.com/en-us/windows/device-security/device-guard/requirements-and-deployment-planning-guidelines-for-device-guard>.

ANSSI-BP-039

Version 1.0 - 08/11/2017

Licence ouverte/Open Licence (Établ - v1)

AGENCE NATIONALE DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

ANSSI - 51, boulevard de La Tour-Maubourg, 75700 PARIS 07-SP

www.ssi.gouv.fr / conseil.technique@ssi.gouv.fr

