AYI NEDJIMI Consultants

Secured-Core PC : Architecture de Sécurité Matérielle Windows 11 & Server 2025

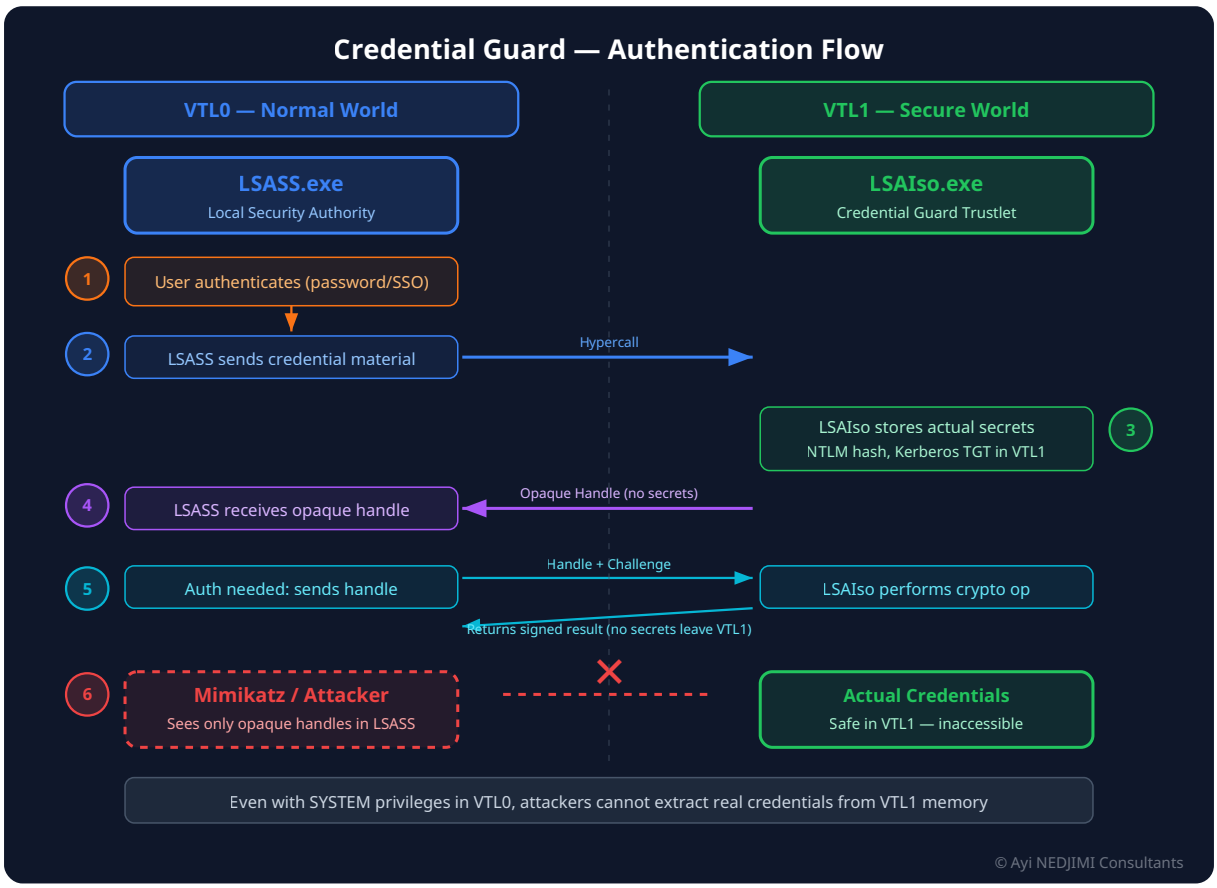
Catégorie : Articles Techniques Lecture : 56 min Publié le : 30/03/2026 Auteur : Ayi NEDJIMI

Guide expert Secured-Core Microsoft : DMA protection, IOMMU, VBS, Credential Guard, UEFI lock, Secure Boot, TPM 2.0, HVCI. Architecture complète Windows 11 e...

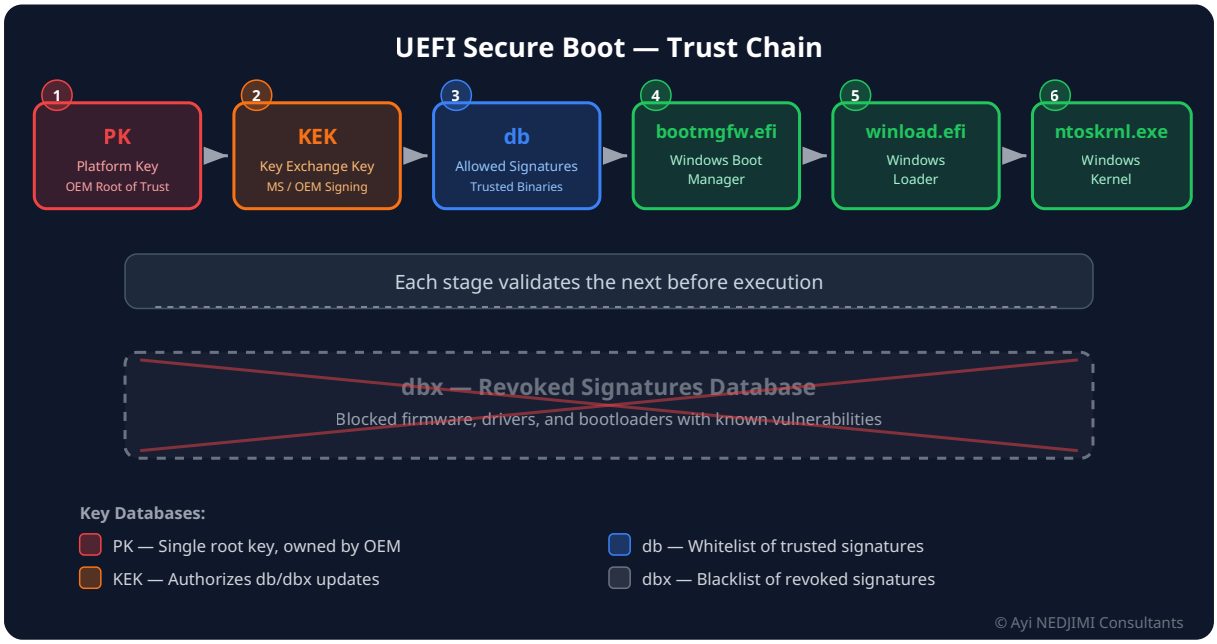
Dans un paysage de menaces où les attaques firmware représentent désormais plus de 80 % des incidents de sécurité les plus critiques recensés par le CERT-FR, la protection logicielle seule ne suffit plus. Les bootkits UEFI comme BlackLotus, les rootkits firmware comme MosaicRegressor, et les attaques DMA via Thunderbolt ont démontré que le modèle de sécurité traditionnel — fondé sur un antivirus opérant dans le même espace que le code malveillant — est structurellement compromis. Microsoft a répondu en 2019 avec l'initiative Secured-Core PC, une architecture de sécurité matérielle qui empile sept couches de protection depuis le silicium jusqu'à l'espace utilisateur : TPM 2.0, UEFI Secure Boot verrouillé, protection DMA via IOMMU, Virtualization-Based Security, Hypervisor-Protected Code Integrity, Credential Guard et System Guard. En 2025, cette architecture est désormais mature, intégrée nativement dans Windows 11 24H2 et Windows Server 2025, et constitue le socle de sécurité exigé par les référentiels les plus stricts. Cet article déconstruit chaque composant, explique leur interaction, et analyse leur impact concret sur la sécurité offensive et défensive en environnement d'entreprise.

Introduction — Pourquoi la sécurité matérielle est devenue critique

Pendant deux décennies, la sécurité des systèmes d'exploitation a reposé sur un postulat implicite : le firmware est fiable, le matériel est neutre, et la menace réside essentiellement dans l'espace utilisateur ou, au pire, dans le noyau. Ce postulat s'est effondré progressivement à mesure que les attaquants sophistiqués — groupes étatiques et APT — ont compris que compromettre le firmware ou le processus de démarrage offrait une persistance quasi indétectable et une résistance totale aux réinstallations système.



LSASS (VTL0) ne manipule que des handles opaques, les vrais secrets sont dans LSAIso (VTL1) — Mimikatz ne voit rien



Chaîne de validation Secure Boot : chaque composant de démarrage est signé et vérifié avant exécution

L'escalade des attaques firmware

Le tournant conceptuel remonte à 2016 avec **ThinkPwn**, une vulnérabilité dans le firmware UEFI des ThinkPad Lenovo permettant l'exécution de code arbitraire en mode SMM (System Management Mode), le niveau de privilège le plus élevé d'un processeur x86 — supérieur même au Ring 0 du noyau. Un attaquant exploitant SMM peut modifier le firmware de manière persistante, désactiver Secure Boot, et installer un implant invisible à tout logiciel de sécurité.

En 2020, ESET a documenté **MosaicRegressor**, le premier bootkit UEFI découvert en conditions réelles (in the wild), attribué à un groupe lié à la Chine. MosaicRegressor modifiait les modules DXE (Driver Execution Environment) du firmware pour injecter un dropper à chaque démarrage, survivant aux réinstallations de Windows et même au remplacement du disque dur. La découverte a confirmé ce que les chercheurs redoutaient depuis des années : les bootkits UEFI ne sont plus théoriques.

Puis, en 2023, **BlackLotus** a bouleversé l'écosystème en devenant le premier bootkit UEFI capable de contourner Secure Boot sur des systèmes entièrement à jour. En exploitant la vulnérabilité CVE-2022-21894 (Baton Drop), BlackLotus pouvait charger un bootloader non signé même avec Secure Boot actif, désactiver BitLocker, HVCI et Windows Defender, et installer un driver noyau persistant. Le prix de ce kit sur les forums cybercriminels — environ 5 000 dollars — a démontré la démocratisation de ces techniques autrefois réservées aux groupes étatiques.

Les limites de la protection logicielle

Le problème fondamental est architectural. Un antivirus, un EDR, ou même le noyau Windows opèrent tous dans le même niveau de confiance — Ring 0 pour le noyau, Ring 3 pour les applications. Un attaquant qui compromet le firmware ou le processus de démarrage s'exécute *avant* ces protections et peut les désactiver ou les manipuler avant même qu'elles ne se chargent. C'est l'équivalent numérique d'un cambrioleur qui modifie les serrures de votre maison avant votre réveil.

De plus, les protections logicielles reposent sur l'intégrité du noyau, qui lui-même repose sur l'intégrité du bootloader, qui repose sur l'intégrité du firmware. Sans racine de confiance matérielle, toute cette chaîne peut être compromise de bas en haut.

La réponse Microsoft : Secured-Core

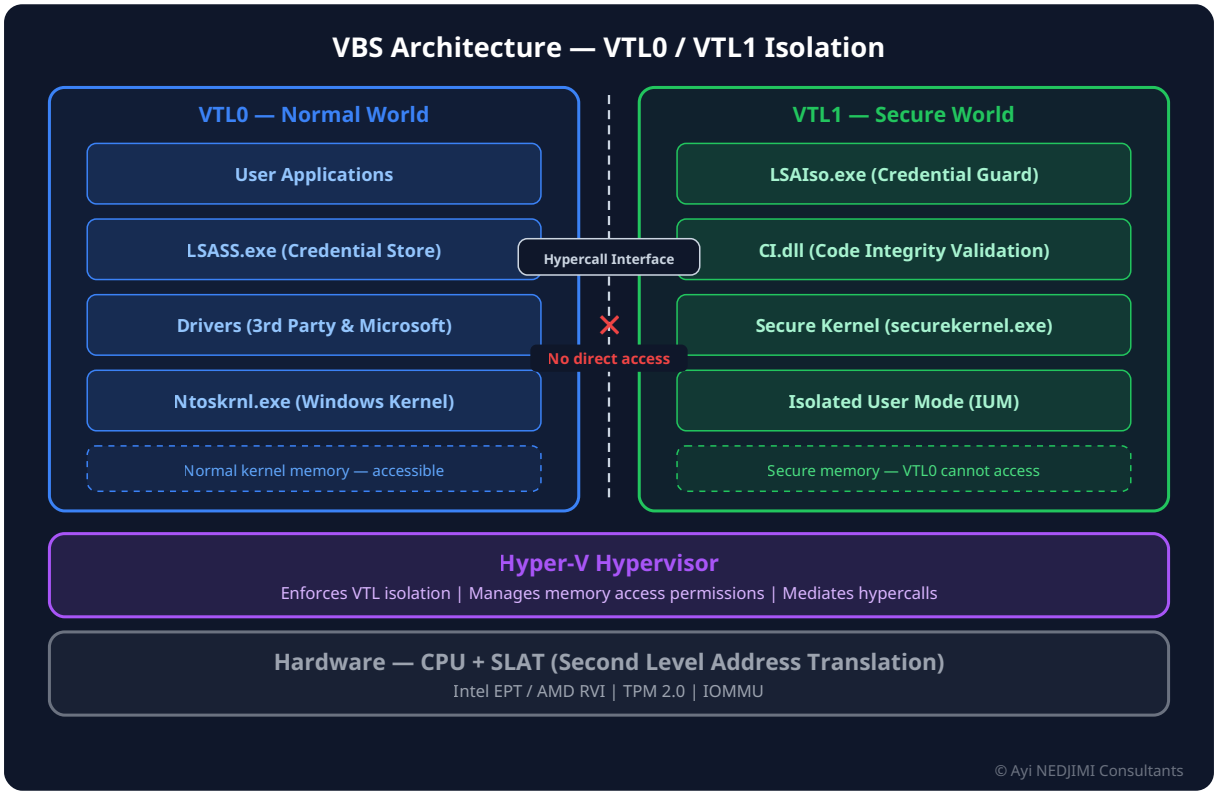
Microsoft a lancé l'initiative Secured-Core PC en octobre 2019, en partenariat avec les fabricants OEM (Dell, HP, Lenovo, Panasonic, Dynabook) et les fabricants de processeurs (Intel, AMD, Qualcomm). L'idée fondatrice est de déplacer la sécurité *sous* le système d'exploitation, en s'appuyant sur le matériel et l'hyperviseur pour créer des zones de confiance que même un noyau compromis ne peut pas atteindre.

Entre 2019 et 2026, Secured-Core a considérablement maturé. Windows 11 a rendu le TPM 2.0 obligatoire. Windows 11 22H2 a activé HVCI par défaut sur les nouvelles installations. Windows 11 24H2 a optimisé VBS pour réduire l'impact sur les performances. Et Windows Server 2025 a intégré Secured-Core comme configuration par défaut pour les déploiements Azure Stack HCI, avec des fonctionnalités spécifiques comme le hotpatch et SMB over QUIC.

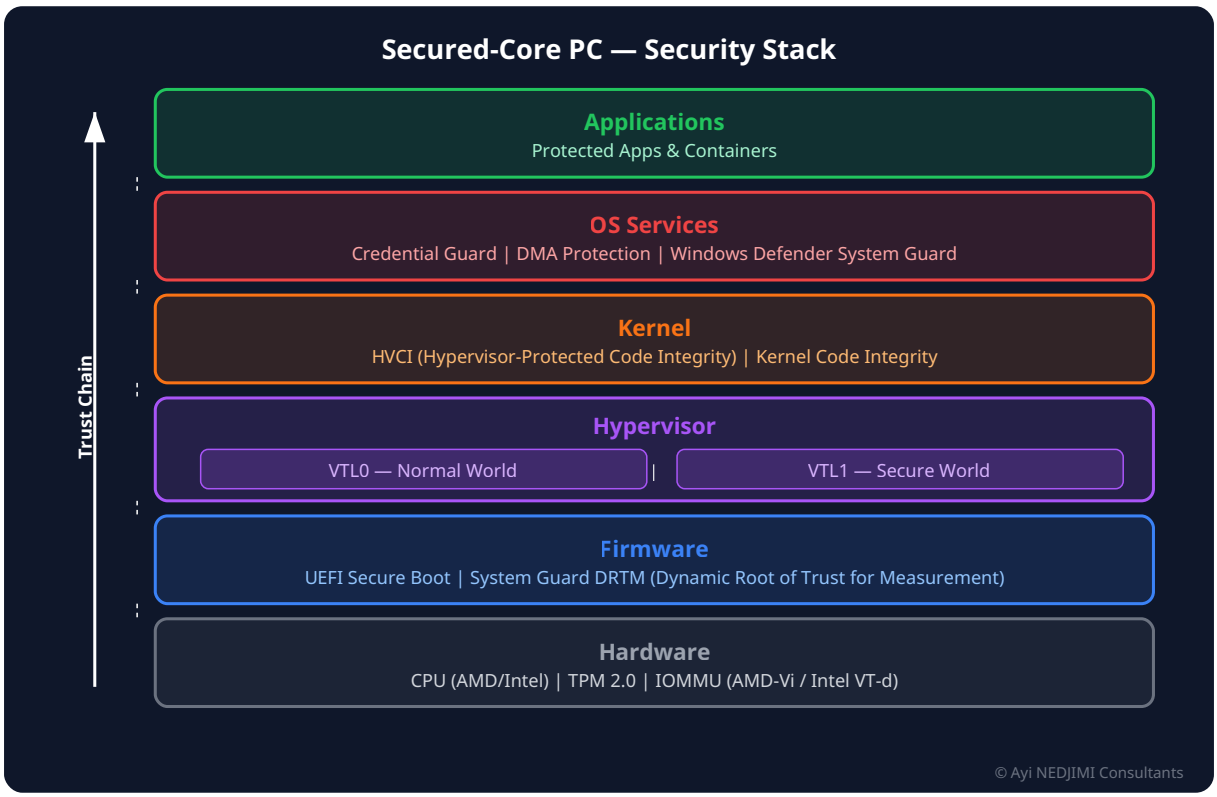
Le périmètre de cet article couvre l'ensemble de l'architecture Secured-Core telle qu'implémentée dans Windows 11 et Windows Server 2025, avec une attention particulière aux implications pour la sécurité offensive et le déploiement en entreprise. Pour une analyse approfondie de l'architecture générale de Windows Server 2025, consultez notre article dédié sur l'architecture système de Windows Server 2025.

Architecture Secured-Core — Vue d'ensemble

L'architecture Secured-Core repose sur sept piliers complémentaires qui s'empilent pour créer une chaîne de confiance depuis le silicium jusqu'à l'espace utilisateur. Chaque couche protège la couche supérieure et dépend de la couche inférieure pour sa propre intégrité. Cette approche, inspirée des modèles de sécurité militaires (compartimentalisation et hiérarchie de confiance), transpose dans le monde civil ce que les processeurs sécurisés des cartes à puce et des enclaves matérielles fournissent depuis des décennies : une séparation matérielle entre le code de confiance et le code potentiellement compromis.



Architecture VBS : VTL0 (monde normal) et VTL1 (monde sécurisé) séparés par l hyperviseur Hyper-V



Architecture en couches Secured-Core : du matériel (TPM, IOMMU) aux applications, chaque couche valide la suivante

La philosophie fondatrice de Secured-Core repose sur un principe simple mais profond : **ne jamais faire confiance au logiciel pour protéger le logiciel**. Si le noyau peut être compromis, les protections qui s'exécutent dans le noyau sont compromises aussi. Si le bootloader peut être remplacé, les protections qui dépendent du bootloader sont contournables. La seule manière de briser ce cercle vicieux est d'ancrer la confiance dans le matériel — une puce TPM qui ne peut pas être reprogrammée, un hyperviseur isolé par les tables SLAT du processeur, un IOMMU qui contrôle les accès DMA indépendamment du système d'exploitation.

Les sept piliers de Secured-Core

Pilier	Couche	Fonction principale	Dépendance
TPM 2.0	Matériel	Racine de confiance, stockage clés, mesures d'intégrité	Silicium (processeur/ puce dédiée)
UEFI Secure Boot	Firmware	Validation cryptographique de la chaîne de démarrage	TPM 2.0 pour les mesures PCR
UEFI Secure Config	Firmware	Verrouillage de la configuration firmware	Secure Boot + OEM firmware
Protection DMA (IOMMU)	Matériel/ Firmware	Isolation mémoire des périphériques	Intel VT-d / AMD-Vi
VBS	Hyperviseur	Création d'environnements d'exécution isolés (VTL0/VTL1)	Virtualisation matérielle + IOMMU
HVCI	Hyperviseur/ Noyau	Intégrité du code noyau appliquée par l'hyperviseur	VBS (s'exécute dans VTL1)
Credential Guard	Hyperviseur/ Utilisateur	Isolation des secrets d'authentification	VBS (LSAIso.exe dans VTL1)

Le modèle en couches

La chaîne de confiance Secured-Core suit un modèle strictement hiérarchique :

- **Couche 0 — Silicium** : Le processeur (Intel, AMD, Qualcomm) fournit les primitives matérielles : extensions de virtualisation (VT-x/AMD-V), IOMMU (VT-d/AMD-Vi), TPM intégré (fTPM/PTT), et DRTM (Intel TXT/AMD PSP). C'est la racine de confiance ultime.
- **Couche 1 — Firmware UEFI** : Le firmware valide sa propre intégrité via Secure Boot, mesure chaque composant dans les PCR du TPM, et verrouille sa configuration contre les modifications non autorisées.
- **Couche 2 — Hyperviseur Hyper-V** : L'hyperviseur se charge avant le noyau Windows et crée deux Virtual Trust Levels : VTL0 (monde normal) et VTL1 (monde sécurisé). L'hyperviseur contrôle les tables SLAT (Second Level Address Translation) pour isoler la mémoire de VTL1.
- **Couche 3 — Noyau Windows (VTL0)** : Le noyau s'exécute dans VTL0. Ses pages de code sont validées par HVCI (qui s'exécute dans VTL1). Le noyau ne peut pas charger de code non signé ni modifier les pages marquées comme exécutables.
- **Couche 4 — Espace utilisateur** : Les applications bénéficient de la protection en cascade. LSASS ne manipule que des handles opaques (Credential Guard), les drivers sont validés (HVCI), et les périphériques DMA sont isolés (IOMMU).

Exigences OEM pour la certification Secured-Core

Pour obtenir la certification Secured-Core, un OEM doit satisfaire un ensemble strict d'exigences matérielles et firmware définies par Microsoft dans le programme Windows Hardware Compatibility :

- TPM 2.0 (discret ou firmware) conforme aux spécifications TCG
- Processeur avec extensions de virtualisation et IOMMU
- Firmware UEFI supportant Secure Boot avec verrouillage de configuration
- Support DRTM (Intel TXT ou AMD PSP Secure Launch)
- Protection DMA pré-démarrage (Kernel DMA Protection)
- Activation par défaut de VBS, HVCI, et Credential Guard en sortie d'usine
- Firmware UEFI conforme au NIST SP 800-147 pour les mises à jour sécurisées

En 2025, les gammes certifiées Secured-Core incluent les Dell Latitude/Precision série 7000, HP EliteBook/ZBook, Lenovo ThinkPad X1/T14s, et Microsoft Surface Pro/Laptop. Côté serveur, Dell PowerEdge R760/R660, HPE ProLiant DL380 Gen11, et Lenovo ThinkSystem SR650 V3 sont certifiés.

Processeurs et architectures de sécurité : Intel, AMD, Qualcomm

Chaque fabricant de processeurs implémente les primitives de sécurité Secured-Core de manière légèrement différente, ce qui a des implications sur le niveau de protection effectif :

Intel : Les processeurs Intel Core de 11ème génération et supérieurs fournissent l'ensemble complet des primitives requises. Intel Platform Trust Technology (PTT) fournit le fTPM. Intel Trusted Execution Technology (TXT) fournit le DRTM. Intel VT-x avec Extended Page Tables (EPT) fournit la virtualisation matérielle pour VBS. Intel VT-d fournit l'IOMMU pour la protection DMA. Les processeurs 12ème génération et supérieurs ajoutent le Mode-Based Execution Control (MBEC), qui améliore significativement les performances de HVCI en éliminant les VM exits pour les vérifications d'exécution utilisateur vs noyau. Intel Control-Flow Enforcement Technology (CET) sur les processeurs 12ème gen+ ajoute également le shadow stack matériel, complétant HVCI avec une protection contre les attaques ROP (Return-Oriented Programming).

AMD : Les processeurs AMD Ryzen/EPYC basés sur Zen 2 et supérieurs supportent Secured-Core. AMD fTPM est implémenté dans le Platform Security Processor (PSP), un coprocesseur ARM Cortex-A5 intégré au die. L'instruction SKINIT du PSP fournit le DRTM. AMD Secure Memory Encryption (SME) et Secure Encrypted Virtualization (SEV) ajoutent une couche supplémentaire de chiffrement mémoire matériel non disponible chez Intel dans le segment grand public. AMD-Vi (IOMMU) fournit la protection DMA. Les processeurs Zen 3+ supportent le Guest Mode Execute Trap (GMET), équivalent AMD du MBEC d'Intel pour l'optimisation des performances HVCI. AMD Pluton, intégré depuis Ryzen 6000, fournit un iTPM directement dans le die du processeur, éliminant le vecteur d'attaque par interception du bus.

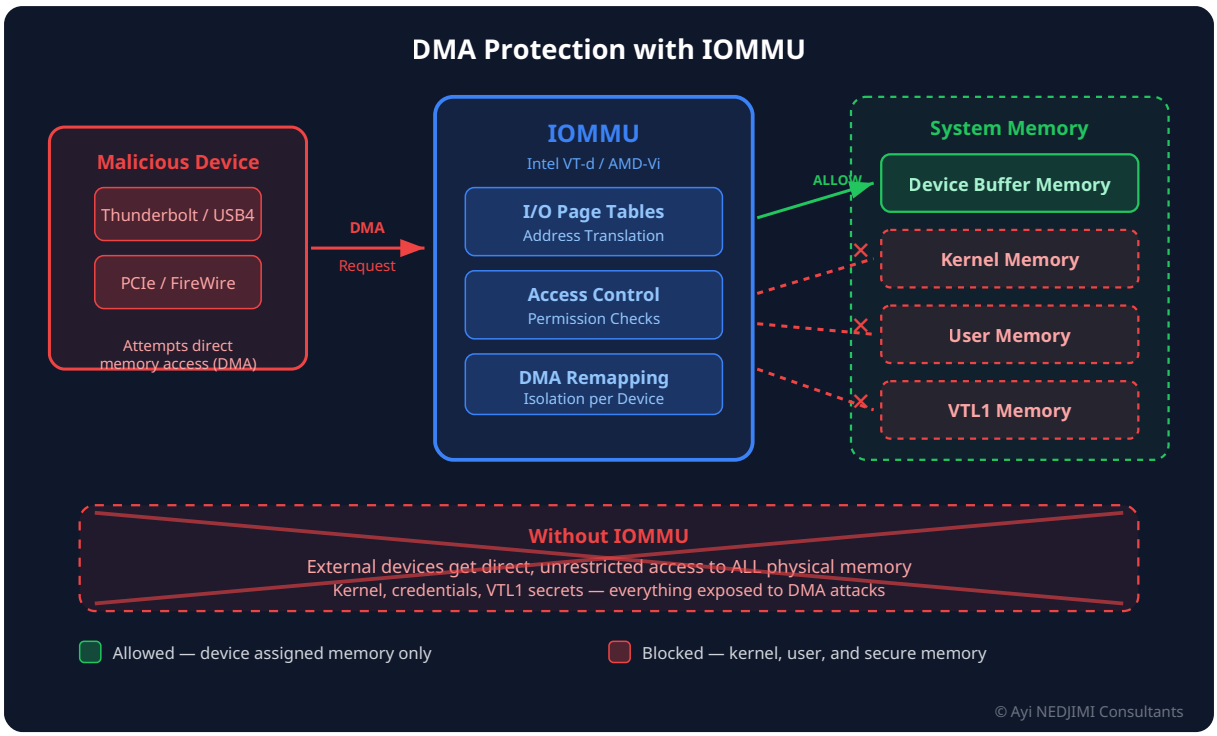
Qualcomm : Les processeurs Qualcomm Snapdragon 8cx Gen 3+ (ARM64) supportent Secured-Core pour les appareils Windows on ARM. Le modèle de sécurité ARM diffère significativement de x86 : ARM TrustZone fournit un environnement d'exécution sécurisé (TEE) matériel, et le TPM est implémenté dans la Secure Processing Unit (SPU) de Qualcomm. Qualcomm Pluton est

également intégré dans les Snapdragon X Elite/Plus pour les PC Copilot+. L'architecture ARM a l'avantage d'une surface d'attaque firmware réduite (pas de SMM, pas de ME/PSP legacy), mais l'écosystème de drivers est plus restreint.

> **Architecture en couches Secured-Core** : La sécurité repose sur un empilement matériel → firmware → hyperviseur → noyau → utilisateur. Chaque couche est validée par la couche inférieure. L'hyperviseur Hyper-V est l'élément central : il crée une zone de confiance (VTL1) inaccessible même au noyau compromis, et héberge HVCI et Credential Guard.

TPM 2.0 — La racine de confiance matérielle

Le Trusted Platform Module (TPM) 2.0 est la fondation sur laquelle repose l'ensemble de l'architecture Secured-Core. Sans TPM, il est impossible de garantir l'intégrité du processus de démarrage, de stocker des clés cryptographiques de manière sécurisée, ou d'attester de l'état de santé d'un système.



L'IOMMU filtre les requêtes DMA : les périphériques externes ne peuvent accéder qu'aux zones mémoire autorisées

Architecture du TPM

Le TPM est un coprocesseur cryptographique spécialisé, défini par les spécifications du Trusted Computing Group (TCG). Il existe en trois variantes architecturales, chacune avec des compromis différents en termes de sécurité et de coût :

- **TPM discret (dTPM)** : Puce physique séparée soudée sur la carte mère, connectée via SPI ou I2C. Offre la meilleure isolation matérielle (le silicium du TPM est physiquement distinct du processeur), mais le bus de communication peut être intercepté (attaque par interposer). Fabricants : Infineon SLB 9672, STMicroelectronics ST33, Nuvoton NPCT750.

- **TPM firmware (fTPM)** : Implémenté en logiciel dans un environnement d'exécution sécurisé du processeur — Intel PTT (Platform Trust Technology) dans l'enclave ME, ou AMD fTPM dans le PSP (Platform Security Processor). Pas de bus externe à intercepter, mais la surface d'attaque inclut le firmware du ME/PSP.
- **TPM intégré (iTPM)** : Implémenté directement dans le die du processeur (Pluton chez Microsoft/AMD/Intel/Qualcomm). Élimine à la fois le bus externe et les vulnérabilités firmware du ME/PSP, car le TPM partage le silicium du processeur avec une isolation matérielle interne.

Hiérarchie des clés TPM

Le TPM 2.0 organise ses clés cryptographiques dans une hiérarchie stricte qui sert à la fois à l'identification unique du matériel et à la protection des secrets :

Clé	Nom complet	Type	Rôle	Extractible ?
EK	Endorsement Key	RSA 2048 / ECC P-256	Identité unique du TPM, générée en usine	Non (clé privée jamais exposée)
SRK	Storage Root Key	RSA 2048 / ECC P-256	Racine de la hiérarchie de stockage, protège les clés enfants	Non
AIK	Attestation Identity Key	RSA 2048 / ECC P-256	Signature des quotes d'attestation sans révéler l'EK	Non
Clés enfants	User keys, sealing keys	Variable	Chiffrement, signature, scellement de secrets	Uniquement sous contrôle de la SRK

La séparation entre EK (identité) et AIK (attestation) répond à un problème de confidentialité : l'EK identifie uniquement le matériel, et révéler l'EK à chaque service d'attestation permettrait le traçage. L'AIK, générée via un processus d'activation par une autorité de certification (Privacy CA), permet d'attester sans exposer l'identité matérielle.

Mesures PCR et secrets scellés

Les Platform Configuration Registers (PCR) sont des registres SHA-256 dans le TPM qui enregistrent les mesures d'intégrité du processus de démarrage. Chaque PCR est initialisé à zéro et étendu (extend) avec le hash de chaque composant mesuré. L'opération d'extension est irréversible : PCR_new = SHA-256(PCR_old || measurement). Cette propriété garantit que toute modification d'un composant produit une valeur PCR finale différente.

Les PCR standard dans un environnement Windows Secured-Core sont les suivants :

- **PCR[0]** : Code firmware UEFI (BIOS principal)
- **PCR[1]** : Configuration firmware UEFI
- **PCR[2]** : Modules Option ROM
- **PCR[3]** : Configuration Option ROM
- **PCR[4]** : Code MBR / bootloader (bootmgfw.efi)

- **PCR[7]** : État Secure Boot (variables PK, KEK, db, dbx)
- **PCR[11]** : BitLocker (utilisé pour le scellement de la clé VMK)
- **PCR[12-14]** : Événements système d'exploitation (mesures Windows Boot)

Le **scellement** (sealing) est le mécanisme par lequel un secret est chiffré par le TPM de telle sorte qu'il ne peut être déchiffré que si les PCR correspondent à des valeurs prédéfinies. C'est le mécanisme utilisé par BitLocker : la clé VMK (Volume Master Key) est scellée contre les PCR[0,2,4,7,11]. Si le firmware, le bootloader ou la configuration Secure Boot changent, le TPM refuse de libérer la clé VMK, et BitLocker demande la clé de récupération.

Intégration BitLocker

BitLocker est le consommateur principal du TPM dans l'écosystème Windows. Le schéma de protection par défaut utilise le TPM seul (sans PIN ni clé USB) pour un démarrage transparent. Le processus est le suivant :

- Au chiffrement, BitLocker génère une FVEK (Full Volume Encryption Key) et une VMK (Volume Master Key)
- La VMK est scellée dans le TPM contre un ensemble de PCR (par défaut PCR[7,11] sur Windows 11)
- Au démarrage, le TPM mesure chaque composant. Si les PCR correspondent, il libère la VMK
- La VMK déchiffre la FVEK, qui déchiffre le volume

Pour renforcer la sécurité, il est recommandé d'ajouter un PIN pré-démarrage (TPM + PIN) ou une clé USB (TPM + clé), ce qui constitue une authentification multi-facteurs au niveau du démarrage.

Le choix du mode de protection BitLocker a des implications directes sur la résistance aux attaques physiques. Le mode TPM seul (sans PIN) est vulnérable à l'attaque par démarrage à froid (cold boot attack) : un attaquant avec un accès physique peut redémarrer le système, le TPM libère automatiquement la VMK, et la mémoire peut être congelée (azote liquide) pour préserver les clés en RAM. Le mode TPM + PIN empêche ce scénario car le TPM refuse de libérer la VMK sans le PIN correct, même si les PCR correspondent. Le mode TPM + Network Key Protector, disponible dans les environnements d'entreprise, ajoute un facteur réseau : le système doit être connecté au réseau d'entreprise pour déverrouiller automatiquement, combinant sécurité physique et localisation réseau.

Windows 11 24H2 et Server 2025 supportent également le chiffrement de l'appareil par défaut (Device Encryption), une version simplifiée de BitLocker activée automatiquement sur les appareils conformes. La clé de récupération est automatiquement sauvegardée dans le compte Microsoft ou Entra ID de l'utilisateur. Pour les déploiements d'entreprise, cette configuration automatique doit être complétée par des politiques BitLocker explicites imposant le PIN pré-démarrage et le chiffrement intégral du disque (pas seulement l'espace utilisé).

Attestation TPM pour la vérification d'intégrité à distance

L'attestation TPM permet à un service distant de vérifier l'état de santé d'un poste. Le processus repose sur la **quote** TPM : le TPM signe les valeurs PCR avec l'AIK, et le service distant compare ces valeurs à des références connues. Microsoft intègre ce mécanisme via le **Windows Health Attestation Service** et **Azure Attestation**, utilisés par Intune pour l'accès conditionnel : un poste dont les PCR ne correspondent pas aux valeurs attendues (firmware modifié, Secure Boot désactivé) se voit refuser l'accès aux ressources d'entreprise.

Attaques TPM et atténuations

Le TPM n'est pas invulnérable. Les principales attaques connues sont :

- **TPM sniffing** : Interception du bus SPI entre un dTPM et le processeur pour capturer la VMK BitLocker en transit. Démontré par Denis Andzakovic (Pulse Security) en 2019 avec un analyseur logique à 30 euros. Atténuation : utiliser fTPM (pas de bus externe) ou ajouter un PIN pré-démarrage BitLocker.
- **TPM interposer** : Insertion d'un dispositif physique entre le dTPM et la carte mère pour intercepter et rejouer les communications. Atténuation : fTPM ou iTPM (Pluton).
- **Vulnérabilités fTPM** : En 2023, des chercheurs de TU Berlin ont démontré faulTPM, une attaque par injection de faute sur le voltage du processeur AMD pour extraire les secrets du fTPM via le PSP. L'attaque nécessite un accès physique et un équipement spécialisé. Atténuation : utiliser un dTPM certifié FIPS 140-2/3 ou un iTPM Pluton.

Commandes de vérification TPM

```
# Vérifier la présence et la version du TPM
Get-Tpm

# Détails complets du TPM via WMI
Get-CimInstance -Namespace "root\cimv2\Security\MicrosoftTpm" -ClassName Win32_Tpm

# Vérifier les PCR actuels (nécessite le module TrustedPlatformModule)
Get-TpmEndorsementKeyInfo -HashAlgorithm Sha256

# Vérifier la version TPM dans le registre
Get-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\TPM\WMI" -Name
"SpecVersion"

# État d'intégrité du TPM via tpmtool
tpmtool getdeviceinformation
```

> **TPM 2.0 comme racine de confiance** : Le TPM fournit les fondations cryptographiques de Secured-Core — stockage sécurisé des clés, mesures d'intégrité via PCR, et scellement des secrets BitLocker. Privilégiez le fTPM (ou iTPM Pluton) au dTPM pour éliminer le vecteur d'attaque par interception du bus SPI, et ajoutez systématiquement un PIN pré-démarrage BitLocker en environnement à haut risque.

UEFI Secure Boot et verrouillage firmware

UEFI Secure Boot est le deuxième pilier de Secured-Core, responsable de la validation cryptographique de chaque composant chargé pendant le processus de démarrage. Mais dans le contexte Secured-Core, Secure Boot seul ne suffit pas — il est complété par le verrouillage de la configuration firmware et le System Guard Secure Launch (DRTM).

La chaîne de confiance Secure Boot

Secure Boot repose sur une hiérarchie de clés stockées dans les variables UEFI non volatiles :

Variable	Nom	Propriétaire	Fonction
PK	Platform Key	OEM (Dell, HP, etc.)	Clé racine, autorise les modifications de KEK. Une seule PK par système.
KEK	Key Exchange Key	OEM + Microsoft	Autorise les modifications de db/dbx. Plusieurs KEK possibles.
db	Signature Database	Microsoft + OEM	Liste des certificats/hashes autorisés pour les bootloaders et drivers UEFI.
dbx	Forbidden Signature Database	Microsoft (via UEFI Forum)	Liste de révocation : certificats/hashes explicitement interdits.

Le processus de validation au démarrage suit une séquence stricte :

- **Étape 1 — Initialisation firmware** : Le firmware UEFI s'initialise et vérifie sa propre intégrité (UEFI Secure Boot Phase 0, dépendant du constructeur).
- **Étape 2 — Chargement PEI/DXE** : Les modules Pre-EFI Initialization et Driver Execution Environment sont chargés. Leurs signatures sont vérifiées contre db/dbx.
- **Étape 3 — Bootloader Windows** : Le firmware charge `bootmgfw.efi` (Windows Boot Manager). Sa signature est vérifiée contre db. Si la signature est dans dbx, le chargement est bloqué.
- **Étape 4 — Chargeur noyau** : `bootmgfw.efi` charge `winload.efi` (ou `winload.exe` pour le mode legacy), qui charge le noyau `ntoskrnl.exe` et les drivers de démarrage. Chaque composant est vérifié.
- **Étape 5 — Noyau** : Le noyau prend le contrôle et applique ses propres politiques d'intégrité de code (CI.dll, HVCI).

Secure Boot vs Secured-Core UEFI Config Lock

Secure Boot standard protège le *processus* de démarrage, mais pas la *configuration* du firmware. Un attaquant avec un accès physique ou un exploit SMM peut désactiver Secure Boot dans les paramètres UEFI. Le UEFI Secure Configuration de Secured-Core va plus loin :

- Les paramètres firmware critiques (Secure Boot on/off, boot order, virtualisation) sont protégés par un mot de passe administrateur stocké dans le TPM
- Les modifications de configuration firmware sont journalisées et mesurées dans les PCR

- Le firmware empêche le démarrage sur des médias non autorisés même si l'attaquant accède au BIOS setup
- Les mises à jour firmware doivent être signées par l'OEM (conformément au NIST SP 800-147)

System Guard Secure Launch (DRTM)

Le Static Root of Trust for Measurement (SRTM) — le modèle standard — commence les mesures dès le premier cycle d'horloge du processeur. Tout le firmware doit être mesuré et fiable. Le problème : le firmware UEFI est une surface d'attaque énorme (des millions de lignes de code), et une vulnérabilité dans n'importe quel module DXE compromet toute la chaîne.

Le **Dynamic Root of Trust for Measurement (DRTM)**, implémenté par System Guard Secure Launch, adopte une approche radicalement différente : au lieu de mesurer tout le firmware depuis le début, il *réinitialise* la chaîne de confiance à un point précis du démarrage, en utilisant des instructions matérielles spécifiques (Intel GETSEC[SENTER] via TXT, ou AMD SKINIT via PSP).

Le processus DRTM est le suivant :

- Le bootloader Windows invoque l'instruction DRTM du processeur
- Le processeur met tous les coeurs en pause sauf le BSP (Bootstrap Processor)
- Le processeur réinitialise les PCR DRTM (PCR[17-22])
- Un Authenticated Code Module (ACM) signé par Intel/AMD est chargé et mesuré
- L'ACM mesure l'hyperviseur Hyper-V et le Secure Kernel dans les PCR DRTM
- La chaîne de confiance est établie à partir de ce point, indépendamment de l'intégrité du firmware en amont

L'avantage majeur du DRTM : même si le firmware UEFI est compromis (comme dans le cas de MosaicRegressor), le DRTM réétablit une chaîne de confiance propre à partir de l'hyperviseur. Le firmware compromis est « en dessous » du point de réinitialisation et ne peut pas influencer les mesures DRTM.

L'attaque BlackLotus expliquée

BlackLotus (découvert par ESET en mars 2023) illustre parfaitement pourquoi Secure Boot seul ne suffit pas. L'attaque exploite CVE-2022-21894, une vulnérabilité dans le chargeur de démarrage Windows qui permet de désactiver les politiques de sécurité au démarrage. Voici la chaîne d'exploitation complète :

- **Phase 1** : L'attaquant inscrit un bootloader légitime mais vulnérable (`bootmgfw.efi` d'une ancienne version de Windows) sur la partition EFI. Ce bootloader est signé par Microsoft et donc accepté par Secure Boot.
- **Phase 2** : Le bootloader vulnérable est exploité via Baton Drop pour désactiver les vérifications d'intégrité ultérieures.
- **Phase 3** : BlackLotus charge son propre bootloader non signé, désactive HVCI, BitLocker et Windows Defender.
- **Phase 4** : Un driver noyau malveillant est installé, offrant persistance et contrôle total.

La leçon cruciale : l'ancien bootloader était dans **db** (autorisé) mais n'avait pas encore été ajouté à **dbx** (interdit). Microsoft a dû révoquer l'ancien certificat, ce qui a pris plus d'un an pour être déployé complètement. Secured-Core atténue ce scénario grâce au DRTM : même si le bootloader est compromis, le DRTM réinitialise la confiance au niveau de l'hyperviseur.

Protection du firmware contre la rétroversion (rollback)

Un vecteur d'attaque sophistiqué consiste à rétrograder le firmware UEFI vers une version antérieure contenant une vulnérabilité connue. Secured-Core adresse ce problème via plusieurs mécanismes complémentaires :

- **Anti-rollback firmware (fuses/eFuses)** : Les processeurs Intel et AMD intègrent des fusibles électroniques (eFuses) qui sont brulés lors des mises à jour firmware critiques. Une fois le fusible brulé, le processeur refuse d'exécuter un firmware dont le numéro de version de sécurité (SVN — Security Version Number) est inférieur à la valeur stockée dans les eFuses. Ce mécanisme est irréversible : même un attaquant avec un accès physique complet ne peut pas annuler le brulage d'un eFuse.
- **dbx updates** : Microsoft publie régulièrement des mises à jour de la Forbidden Signature Database (dbx) pour révoquer les anciens bootloaders vulnérables. C'est le mécanisme utilisé pour bloquer BlackLotus, bien que le déploiement ait pris plus d'un an en raison des risques de briquage de systèmes mal configurés.
- **Windows UEFI CA 2023** : Microsoft a lancé en 2024 une nouvelle autorité de certification UEFI (Windows UEFI CA 2023) qui remplace l'ancienne Windows UEFI CA 2011. Tous les nouveaux bootloaders sont signés avec le nouveau certificat, et l'ancien sera progressivement révoqué via dbx, invalidant l'ensemble des anciens bootloaders vulnérables.

Cette transition CA est l'une des opérations de sécurité les plus délicates de l'histoire de l'écosystème PC : révoquer l'ancien certificat bloque le démarrage de tout système qui n'a pas été mis à jour vers un bootloader signé par la nouvelle CA. Le déploiement est donc extrêmement progressif, avec des phases de validation sur plusieurs années.

Vérification de l'état Secure Boot

```
# PowerShell : vérifier Secure Boot
Confirm-SecureBootUEFI
# Retourne True ou False

# Détails via msinfo32
# Démarrer > msinfo32 > Résumé système > "État du démarrage sécurisé"

# Vérifier les variables Secure Boot
Get-SecureBootUEFI -Name PK
Get-SecureBootUEFI -Name KEK
Get-SecureBootUEFI -Name db
Get-SecureBootUEFI -Name dbx

# Lister les certificats Secure Boot db
[System.Text.Encoding]::ASCII.GetString((Get-SecureBootUEFI db).bytes) | Select-String
-Pattern "Microsoft"

# Vérifier le mode de démarrage (UEFI vs Legacy)
bcdedit /enum firmware
```

> **DRTM est la réponse à BlackLotus** : Secure Boot seul ne protège pas contre les bootkits qui exploitent d'anciens bootloaders signés. Le System Guard Secure Launch (DRTM) de Secured-Core réinitialise la chaîne de confiance au niveau de l'hyperviseur, rendant les compromissions firmware en amont inopérantes. Exigez le DRTM (Intel TXT / AMD PSP) sur tout matériel déployé en environnement sensible.

VBS — Virtualization-Based Security

Virtualization-Based Security (VBS) est l'innovation architecturale centrale de Secured-Core. En interposant un hyperviseur entre le matériel et le système d'exploitation, VBS crée des environnements d'exécution isolés que même un noyau compromis ne peut pas atteindre. C'est un changement de paradigme fondamental dans la sécurité Windows.

Architecture VTL0 / VTL1

VBS utilise l'hyperviseur Hyper-V (le même que pour la virtualisation de serveurs, mais configuré en mode root partition) pour créer deux **Virtual Trust Levels** (VTL) :

- **VTL0 (monde normal)** : C'est l'environnement Windows standard. Le noyau (ntoskrnl.exe), les drivers, les services et les applications utilisateur s'exécutent dans VTL0. VTL0 est l'environnement le « moins privilégié » dans le modèle VBS — un concept contre-intuitif car il inclut le Ring 0 (noyau).
- **VTL1 (monde sécurisé)** : Un environnement d'exécution entièrement isolé, invisible depuis VTL0. VTL1 exécute le Secure Kernel (securekernel.exe), le processus Isolated User Mode (IUM), et les composants critiques comme LSAIso.exe (Credential Guard) et CI.dll (HVCI). VTL1 possède son propre noyau, sa propre pile mémoire et ses propres services.

La séparation entre VTL0 et VTL1 est appliquée par l'hyperviseur via les **Second Level Address Translation tables** (SLAT, implémentées comme EPT chez Intel ou NPT chez AMD). L'hyperviseur configure les tables SLAT de telle sorte que VTL0 ne peut physiquement pas accéder aux pages mémoire de VTL1. Cette isolation est matérielle — elle ne repose sur aucun contrôle logiciel dans VTL0.

Ce qui s'exécute dans VTL1

Le contenu de VTL1 est intentionnellement minimal pour réduire la surface d'attaque :

- **Secure Kernel (SK)** : securekernel.exe — Un micro-noyau minimaliste qui gère la mémoire et l'ordonnancement dans VTL1. Il ne charge pas de drivers tiers.
- **CI.dll (Code Integrity)** : Le module de vérification d'intégrité du code noyau, déplacé de VTL0 vers VTL1 quand HVCI est actif. Toutes les validations de signatures de code noyau sont effectuées dans VTL1.
- **LSAIso.exe** : Le processus isolé de Credential Guard qui stocke et manipule les secrets d'authentification (NTLM hashes, Kerberos TGTs).
- **Trustlets IUM** : Des processus légers exécutés dans l'Isolated User Mode de VTL1 pour des tâches cryptographiques spécifiques.

Isolation mémoire par SLAT

Le mécanisme d'isolation SLAT fonctionne de la manière suivante. L'hyperviseur maintient deux jeux de tables SLAT : un pour VTL0 et un pour VTL1. Les tables SLAT de VTL0 ne contiennent aucun mapping vers les pages physiques allouées à VTL1. Ainsi, même si un attaquant obtient l'exécution de code en Ring 0 dans VTL0 (compromettre le noyau), toute tentative d'accéder à la mémoire de VTL1 provoque un **EPT violation** interceptée par l'hyperviseur, qui termine le processus fautif.

De plus, l'hyperviseur contrôle les permissions d'exécution de chaque page mémoire dans VTL0. C'est la base de HVCI : l'hyperviseur marque une page comme exécutable uniquement après validation par CI.dll dans VTL1. Un driver noyau dans VTL0 ne peut pas allouer de mémoire exécutable directement — il doit passer par le processus de validation HVCI.

Prérequis matériels pour VBS

- Processeur avec extensions de virtualisation : Intel VT-x avec EPT, ou AMD-V avec NPT (RVI)
- IOMMU : Intel VT-d ou AMD-Vi (obligatoire pour protéger contre les attaques DMA ciblant l'hyperviseur)
- UEFI firmware (pas de BIOS legacy)
- Minimum 4 Go de RAM (recommandé 8 Go+)
- TPM 2.0 (requis pour l'attestation d'intégrité VBS)
- 64 bits obligatoire (VBS n'existe pas en 32 bits)

Activation de VBS

```
# Méthode 1 : Via la stratégie de groupe (GPO)
# Configuration ordinateur > Modèles d'administration > Système > Device Guard
# "Activer la sécurité basée sur la virtualisation" = Activé
# Sélectionner le niveau de sécurité de la plateforme : Démarrage sécurisé et protection
DMA

# Méthode 2 : Via le registre
reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
EnableVirtualizationBasedSecurity /t REG_DWORD /d 1 /f
reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
RequirePlatformSecurityFeatures /t REG_DWORD /d 3 /f
# Valeur 1 = Secure Boot, 3 = Secure Boot + DMA Protection

# Méthode 3 : Via DISM (déploiement d'image)
DISM /Online /Enable-Feature /FeatureName:Microsoft-Hyper-V-Hypervisor /All /NoRestart

# Méthode 4 : Via Intune (Microsoft Endpoint Manager)
# Endpoint Security > Account Protection > Device Guard
# Virtualization Based Security = Enable VBS with Secure Boot and DMA
```

Vérification de l'état VBS

```
# Vérifier VBS via msinfo32
# Démarrer > msinfo32 > Résumé système > "Sécurité basée sur la virtualisation"
# Doit afficher : "En cours d'exécution"

# Vérifier VBS via PowerShell
$vbs = Get-CimInstance -ClassName Win32_DeviceGuard -Namespace
root\Microsoft\Windows\DeviceGuard
$vbs.VirtualizationBasedSecurityStatus
# 0 = Désactivé, 1 = Activé mais pas en cours, 2 = En cours d'exécution

# Services VBS actifs
$vbs.SecurityServicesRunning
# 1 = Credential Guard, 2 = HVCI, 3 = System Guard Secure Launch

# Propriétés de sécurité requises
$vbs.RequiredSecurityProperties
$vbs.AvailableSecurityProperties
```

Impact sur les performances

L'activation de VBS introduit un hyperviseur entre le matériel et le système d'exploitation, ce qui a un coût mesurable. Les benchmarks indépendants montrent :

- **Overhead général** : 5-15 % sur les opérations intensives en CPU et mémoire, principalement dû aux VM exits et aux vérifications SLAT.
- **I/O disque** : 2-8 % d'overhead sur les opérations de lecture/écriture aléatoires, dû à la couche supplémentaire d'adressage mémoire.
- **Gaming / GPU** : 0-5 % d'impact, les opérations GPU n'étant pas directement affectées par VBS.

- **Compilation / développement** : 10-15 % d'overhead, ces charges de travail étant intensives en création/destruction de processus et mappings mémoire.

Il convient de noter que ces mesures de performances varient considérablement selon les charges de travail spécifiques. Les applications de base de données en mémoire (Redis, SAP HANA) peuvent observer un overhead plus élevé en raison de leur pattern d'accès mémoire intensif. Les serveurs web et applicatifs (IIS, Kestrel, Tomcat) subissent un overhead minimal car leur profil de charge est dominé par les I/O réseau. Les environnements de virtualisation imbriquée (Hyper-V sur Hyper-V, ou Docker sur VBS) ont un overhead plus significatif car chaque couche de virtualisation ajoute un niveau de traduction d'adresses supplémentaire.

Windows 11 24H2 a introduit des optimisations significatives pour réduire cet overhead, notamment le **MBEC** (Mode-Based Execution Control) matériel sur les processeurs Intel 11ème génération+ et AMD Zen 3+, qui élimine le besoin de certains VM exits pour les vérifications d'intégrité de code, réduisant l'overhead à 2-5 % dans la plupart des scénarios.

> **VBS : l'hyperviseur comme gardien** : VBS interpose l'hyperviseur Hyper-V entre le matériel et le noyau, créant VTL1 — un monde sécurisé invisible depuis VTL0. Même avec un noyau entièrement compromis (Ring 0), un attaquant ne peut pas accéder aux secrets ni au code de VTL1. L'isolation est matérielle (SLAT/EPT), pas logicielle. L'overhead de 5-15 % est réduit à 2-5 % avec MBEC sur les processeurs récents.

HVCI — Hypervisor-Protected Code Integrity

Hypervisor-Protected Code Integrity (HVCI), également appelé *Memory Integrity* dans l'interface Windows, est le mécanisme qui empêche le chargement et l'exécution de code noyau non signé ou malveillant. C'est la protection qui transforme VBS d'une isolation passive en une défense active contre les rootkits et les drivers malveillants. Pour une analyse technique approfondie, consultez notre article HVCI Deep Dive : Intégrité du code par l'hyperviseur.

Principe de fonctionnement

Sans HVCI, le module Code Integrity (CI.dll) s'exécute dans le même espace que le noyau (VTL0). Un attaquant avec un accès noyau peut patcher CI.dll, modifier ses décisions, ou directement mapper du code exécutable en mémoire sans passer par la validation. Avec HVCI, CI.dll est déplacé dans VTL1. Le processus de validation du code noyau devient le suivant :

- Un driver ou module noyau doit être chargé dans VTL0
- Le noyau (VTL0) demande à l'hyperviseur d'allouer des pages de code exécutables
- L'hyperviseur transmet la requête à CI.dll dans VTL1
- CI.dll valide la signature Authenticode du code, vérifie le hash contre les catalogues de drivers, et vérifie la conformité WHQL
- Si la validation réussit, l'hyperviseur marque les pages mémoire comme exécutables (Execute) mais non écrivables (pas de Write) dans les tables SLAT

- Si la validation échoue, les pages restent non exécutables et le driver ne peut pas fonctionner

Le point crucial est que l'hyperviseur applique l'**invariant W^X** (Write XOR Execute) : une page mémoire noyau peut être écrivable OU exécutable, mais jamais les deux simultanément. Cela élimine entièrement les techniques d'attaque fondées sur l'écriture de shellcode dans des pages exécutables ou la modification de code noyau en mémoire.

Impact sur les drivers noyau

HVCI impose des contraintes strictes sur les drivers noyau. Tous les drivers doivent être compatibles HVCI, ce qui signifie :

- **Pas de code auto-modifiant** : Un driver ne peut pas générer ou modifier du code en mémoire à l'exécution
- **Pas de pages RWX** : Aucune allocation mémoire avec permissions Read+Write+Execute
- **Pas de manipulation directe des tables de pages** : Les modifications SLAT sont exclusivement contrôlées par l'hyperviseur
- **Signature obligatoire** : Tout code noyau doit être signé par un certificat reconnu par Microsoft (WHQL ou signature de test en mode développeur)
- **Pas de mapping de sections non alignées** : Les sections PE doivent être alignées sur les limites de page (4 Ko)

Ces contraintes ont causé des problèmes de compatibilité significatifs lors du déploiement initial de HVCI. Certains drivers anciens (pré-2018) utilisaient des techniques de code auto-modifiant pour des raisons de performance ou d'obfuscation. Les drivers de certains produits antivirus utilisant des hooks noyau étaient également incompatibles. Microsoft maintient une liste de compatibilité HVCI, et le Windows Hardware Compatibility Program exige la compatibilité HVCI depuis 2020.

Intégration WDAC

HVCI travaille en tandem avec **Windows Defender Application Control (WDAC)** pour créer une politique complète de contrôle de code. HVCI gère l'intégrité du code noyau (drivers, modules), tandis que WDAC contrôle le code utilisateur (applications, scripts). Ensemble, ils constituent une défense en profondeur :

- WDAC définit quelles applications sont autorisées à s'exécuter (allow list)
- HVCI garantit que seul du code noyau signé et validé peut s'exécuter
- Les deux politiques sont appliquées indépendamment : même si une application WDAC-autorisée tente de charger un driver non signé, HVCI le bloque

WDAC en mode Managed Installer (intégration SCCM/Intune) facilite considérablement le déploiement en environnement d'entreprise : tout logiciel déployé via le système de gestion est automatiquement autorisé, sans nécessiter de règles manuelles. La politique peut être complétée par des règles basées sur le hash du fichier, le certificat de signature, ou le chemin d'accès. En 2025, WDAC supporte également les politiques supplémentaires (supplemental

policies), permettant d'ajouter des exceptions sans modifier la politique de base — essentiel pour les déploiements à grande échelle où chaque service peut avoir des besoins applicatifs spécifiques.

Microsoft Vulnerable Driver Blocklist

Un complément critique de HVCI est la **Microsoft Vulnerable Driver Blocklist**, une liste maintenue par Microsoft de drivers légitimes (signés WHQL) mais contenant des vulnérabilités exploitées en conditions réelles. HVCI vérifie la signature des drivers mais ne détecte pas les vulnérabilités dans du code signé — c'est le problème du BYOVD (Bring Your Own Vulnerable Driver). La Vulnerable Driver Blocklist comble cette lacune en bloquant les drivers connus pour être exploités.

La liste est mise à jour via Windows Update et intégrée dans les politiques HVCI. Elle contient, entre autres, les drivers suivants fréquemment exploités par les ransomwares et les APT :

- **RTCore64.sys / RTCore32.sys** (MSI Afterburner) : Accès arbitraire en lecture/écriture à la mémoire physique et aux MSR. Exploité par BlackByte, AvosLocker, et de nombreux ransomwares.
- **DBUtil_2_3.sys** (Dell BIOS Utility) : CVE-2021-21551. Accès noyau arbitraire. Exploité par Lazarus Group.
- **gdrv.sys** (GIGABYTE) : Accès mémoire physique arbitraire. Utilisé dans des chaînes d'exploitation pour désactiver les EDR.
- **WinRing0.sys** : Utilisé par de nombreux outils de monitoring matériel, permet l'accès aux MSR et ports I/O. Largement exploité pour les élévations de privilèges.

La limitation de cette approche est que la liste ne couvre que les drivers connus et rapportés. Des chercheurs estiment que des centaines de drivers signés contiennent des vulnérabilités non encore répertoriées. Le projet LOLDrivers (Living Off The Land Drivers) maintient une base de données communautaire plus exhaustive que la liste officielle Microsoft.

Vérification de l'état HVCI

```
# Vérifier HVCI via msinfo32
# "Sécurité basée sur la virtualisation - Services en cours d'exécution"
# Doit inclure : "Application de l'intégrité du code appliquée par l'hyperviseur"

# Vérifier HVCI via PowerShell
$dg = Get-CimInstance -ClassName Win32_DeviceGuard -Namespace
root\Microsoft\Windows\DeviceGuard
$dg.SecurityServicesRunning -contains 2
# True = HVCI est actif

# Vérifier les drivers incompatibles
$dg.CodeIntegrityPolicyEnforcementStatus

# Registre : état HVCI
Get-ItemProperty -Path "HKLM:
\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegrity"
-Name Enabled
# 1 = Activé

# Outil de vérification de compatibilité des drivers
# https://learn.microsoft.com/en-us/windows-hardware/test/hlk/
# Ou via le Device Guard Readiness Tool
DG_Readiness_Tool_v3.6.ps1 -Ready
```

> **HVCI : W^X matériel pour le noyau** : HVCI déplace la validation du code noyau dans VTL1 et impose l'invariant W^X via l'hyperviseur. Résultat : impossible de charger un driver non signé, d'exécuter du shellcode en noyau, ou de modifier du code noyau en mémoire. Vérifiez la compatibilité de tous vos drivers avant l'activation en production.

Credential Guard — Isolation des secrets d'authentification

Credential Guard est peut-être la protection Secured-Core la plus immédiatement pertinente pour les équipes de sécurité offensive et défensive, car elle neutralise directement l'une des techniques les plus utilisées en pentest et par les attaquants : l'extraction de credentials depuis LSASS avec Mimikatz.

Le problème : LSASS dans VTL0

Historiquement, le processus Local Security Authority Subsystem Service (lsass.exe) stocke en mémoire les secrets d'authentification des utilisateurs connectés : hashes NTLM, Kerberos TGTs (Ticket Granting Tickets), mots de passe en clair (via WDigest), et credentials dérivés. LSASS s'exécute dans l'espace utilisateur (Ring 3) de VTL0, ce qui signifie que tout processus avec les privilèges `SeDebugPrivilege` (typiquement un administrateur local) peut lire sa mémoire.

L'outil Mimikatz, créé par Benjamin Delpy, exploite exactement cette architecture : la commande `sekurlsa::logonpasswords` lit la mémoire de LSASS et extrait tous les secrets en clair ou sous forme de hashes. Ces credentials sont ensuite utilisés pour du mouvement latéral (Pass-the-

Hash, Pass-the-Ticket, Overpass-the-Hash), constituant l'un des vecteurs d'attaque les plus dévastateurs en environnement Active Directory. Pour plus de détails sur les attaques AD, consultez notre guide complet du pentest Active Directory.

La solution : LSAIso.exe dans VTL1

Credential Guard déplace les opérations cryptographiques sensibles de LSASS vers un nouveau processus isolé, `LSAIso.exe` (LSA Isolated), qui s'exécute dans VTL1. L'architecture révisée fonctionne ainsi :

- LSASS continue de s'exécuter dans VTL0, mais il ne manipule plus directement les secrets cryptographiques
- Quand LSASS a besoin d'effectuer une opération cryptographique (par exemple, dériver un TGS à partir d'un TGT), il envoie une requête à LSAIso via un canal RPC sécurisé contrôlé par l'hyperviseur
- LSAIso effectue l'opération dans VTL1 et retourne un **handle opaque** à LSASS
- LSASS stocke le handle opaque mais n'a jamais accès au secret sous-jacent

Conséquence directe : quand Mimikatz lit la mémoire de LSASS sur un système avec Credential Guard, il ne trouve que des handles opaques — des références sans valeur cryptographique. Les hashes NTLM, TGTs Kerberos et credentials dérivés résident dans la mémoire de VTL1, physiquement inaccessible depuis VTL0.

Ce qui est protégé et ce qui ne l'est pas

Secret	Protégé par Credential Guard ?	Détail
Hashes NTLM	Oui	Stockés dans LSAIso (VTL1)
Kerberos TGTs	Oui	Stockés dans LSAIso (VTL1)
Kerberos session keys	Oui	Opérations dans LSAIso (VTL1)
Credentials dérivés (NTLM relay)	Oui	Calcul dans LSAIso (VTL1)
Credentials en cache (domain cached)	Non	Stockage local, pas dans LSASS runtime
DPAPI Master Keys	Non	Stockage différent, hors LSASS
Tokens SSO / cookies navigateur	Non	Espace utilisateur, hors périmètre LSASS
Mots de passe WDigest	Oui (désactivé par défaut)	WDigest désactivé depuis Windows 8.1 Update
Comptes locaux (SAM)	Non	SAM hashes stockés dans le registre, pas dans LSASS

Remote Credential Guard pour RDP

Remote Credential Guard étend la protection aux sessions Remote Desktop. Sans Remote Credential Guard, une connexion RDP transmet les credentials de l'utilisateur au serveur distant, où ils sont stockés dans le LSASS du serveur — une cible idéale pour Mimikatz sur un serveur compromis. Avec Remote Credential Guard :

- Les credentials ne quittent jamais le client
- Le serveur RDP reçoit un ticket Kerberos à usage unique (compound identity)
- Mimikatz sur le serveur ne trouve aucun credential exploitable
- Le Single Sign-On fonctionne normalement pour l'utilisateur

Activation : `mstsc.exe /remoteGuard` ou via GPO (Configuration ordinateur > Modèles d'administration > Système > Délégation de credentials > Restreindre la délégation de credentials aux serveurs distants).

Impact sur la sécurité offensive

L'impact de Credential Guard sur les techniques de pentest classiques est considérable :

- **Mimikatz** `sekurlsa::logonpasswords` : Ne retourne que des handles opaques pour les credentials protégés. Les comptes locaux (SAM) restent vulnérables.
- **Pass-the-Hash (NTLM)** : Les hashes NTLM des comptes domaine sont dans VTL1. Seuls les hashes de comptes locaux restent exploitables.
- **Pass-the-Ticket (Kerberos)** : Les TGTs sont dans VTL1. Impossible d'extraire un TGT pour la réutilisation.
- **Golden Ticket** : Le krbtgt hash est dans Active Directory, pas dans LSASS — Credential Guard ne le protège pas directement. Mais un Golden Ticket injecté dans une session sur un système avec Credential Guard aura des limitations.
- **DCSync** : Utilise les droits de réplication AD, pas LSASS — non affecté par Credential Guard.

Les adaptations red team incluent le pivot vers les tokens SSO, DPAPI, les comptes locaux, ou l'exploitation de systèmes sans Credential Guard dans le réseau. Pour approfondir ces techniques, consultez notre article sur l'escalade de privilèges Windows 2025.

Activation et configuration

```
# Méthode 1 : GPO
# Configuration ordinateur > Modèles d'administration > Système > Device Guard
# "Activer la sécurité basée sur la virtualisation"
# Credential Guard : "Activé avec verrouillage UEFI"

# Méthode 2 : Registre
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v LsaCfgFlags /t REG_DWORD /d 1 /f
# 0 = Désactivé, 1 = Activé sans verrouillage UEFI, 2 = Activé avec verrouillage UEFI

# Méthode 3 : Intune
# Endpoint Security > Account Protection
# Credential Guard : Enable with UEFI lock

# Vérification
Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGuard
| Select-Object -Property SecurityServicesRunning
# La valeur 1 dans SecurityServicesRunning indique Credential Guard actif
```

Credential Guard et Kerberos : implémentation détaillée

La manière dont Credential Guard protège les secrets Kerberos mérite une explication détaillée, car c'est l'un des mécanismes les plus critiques pour la sécurité des environnements Active Directory.

Dans un flux Kerberos standard sans Credential Guard, voici ce qui se passe lors d'une authentification : l'utilisateur fournit son mot de passe, LSASS dérive la clé de session à partir du mot de passe (via la fonction de dérivation AES-256 ou RC4-HMAC), envoie un AS-REQ au KDC, reçoit un TGT chiffré et une clé de session, et stocke le tout en mémoire. Ce TGT et cette clé de session sont les cibles principales de Mimikatz.

Avec Credential Guard actif, le flux change fondamentalement : le mot de passe (ou son hash) est transmis à LSAIso dans VTL1 via le canal sécurisé. LSAIso effectue la dérivation de clé et construit le AS-REQ dans VTL1. Le TGT reçu du KDC est stocké dans la mémoire de VTL1. Quand LSASS dans VTL0 a besoin de demander un TGS (Ticket Granting Service) pour accéder à un service, il envoie une requête à LSAIso. LSAIso utilise le TGT stocké dans VTL1 pour construire le TGS-REQ, le signe avec la clé de session, et retourne le TGS à LSASS. LSASS reçoit le TGS mais n'a jamais accès au TGT ni à la clé de session originale.

Cette architecture a une conséquence importante pour les attaques de type **Overpass-the-Hash** (Pass-the-Key) : même si un attaquant obtient le hash NTLM d'un compte (depuis la base SAM d'un autre système, par exemple), il ne peut pas l'utiliser pour obtenir un TGT sur un système avec Credential Guard, car la demande de TGT passe par LSAIso qui effectue des vérifications supplémentaires sur l'origine de la requête.

Recherche sur les contournements (2024-2025)

La recherche en sécurité sur les contournements de Credential Guard est un domaine actif. Les approches étudiées incluent :

- **Évasion VTL0 → VTL1** : Exploiter une vulnérabilité dans l'hyperviseur pour accéder à la mémoire VTL1. En théorie possible mais extrêmement difficile en pratique — l'hyperviseur Hyper-V a une surface d'attaque réduite et est soumis à des audits intensifs (Microsoft offre jusqu'à 250 000 $ de bug bounty pour les vulnérabilités hyperviseur).
- **Attaque par canal auxiliaire** : Utiliser des attaques de type Spectre/Meltdown pour lire la mémoire VTL1 depuis VTL0. Les processeurs récents intègrent des mitigation matérielles, et l'hyperviseur applique des protections supplémentaires (IBRS, STIBP, SSBD).
- **Attaque du canal RPC LSASS ↔ LSAIso** : Intercepter ou manipuler les communications entre LSASS et LSAIso. Le canal est protégé par l'hyperviseur et chiffré, rendant cette approche non viable en pratique.
- **Contournement fonctionnel** : Plutôt que de contourner Credential Guard techniquement, utiliser des techniques qui ne dépendent pas de LSASS (SSO tokens, DPAPI, Kerberoasting, comptes de service, comptes locaux). C'est l'approche la plus réaliste pour les red teams.

> **Credential Guard neutralise Mimikatz** : En déplaçant les secrets d'authentification (NTLM hashes, Kerberos TGTs) dans VTL1 via LSAIso.exe, Credential Guard rend l'extraction de credentials depuis LSASS inopérante. Les red teams doivent pivoter vers les tokens SSO, DPAPI, Kerberoasting et les comptes locaux. Activez avec verrouillage UEFI (`LsaCfgFlags = 2`) pour empêcher la désactivation à distance.

Protection DMA et IOMMU

La protection DMA (Direct Memory Access) est le pilier de Secured-Core qui neutralise les attaques exploitant les périphériques externes pour lire ou écrire directement dans la mémoire physique du système, contournant entièrement le système d'exploitation et ses protections.

Les attaques DMA expliquées

Le DMA est un mécanisme matériel légitime : il permet aux périphériques (cartes réseau, contrôleurs de stockage, GPU) d'accéder directement à la mémoire physique sans passer par le processeur, ce qui est essentiel pour les performances. Le problème : les ports d'extension externes (Thunderbolt, ExpressCard, FireWire, M.2) exposent le bus PCIe, et donc le DMA, à des périphériques potentiellement malveillants.

Un attaquant connecté via Thunderbolt peut :

- **Lire la mémoire physique intégralement** : Extraire les clés BitLocker, les credentials LSASS, les clés de chiffrement en mémoire
- **Écrire en mémoire physique** : Patcher le noyau en mémoire, désactiver des protections, injecter du code

- **Contourner l'écran de verrouillage** : Modifier en mémoire les structures d'authentification pour déverrouiller le système

Les outils principaux pour les attaques DMA sont :

- **PCILeech** (Ulf Frisk) : Framework open source utilisant des FPGA (Screamer, LambdaConcept) pour effectuer des opérations DMA à haut débit. Peut lire la mémoire physique complète et charger des modules noyau directement via DMA.
- **Inception** : Outil spécifique Thunderbolt/FireWire pour le déverrouillage d'écran et l'extraction de credentials.
- **Volatility (avec acquisition DMA)** : Analyse forensique de la mémoire acquise via DMA.

IOMMU : la solution matérielle

L'IOMMU (Input/Output Memory Management Unit) est l'équivalent de la MMU (Memory Management Unit) du processeur, mais pour les périphériques d'entrée/sortie. Implémenté comme **Intel VT-d** (Virtualization Technology for Directed I/O) ou **AMD-Vi** (AMD I/O Virtualization), l'IOMMU crée des **tables de pages I/O** qui limitent les accès DMA de chaque périphérique à des régions mémoire spécifiques.

Sans IOMMU, un périphérique PCIe a potentiellement accès à toute la mémoire physique. Avec IOMMU, chaque périphérique ne peut accéder qu'aux pages mémoire explicitement autorisées par le système d'exploitation dans les tables de remapping DMA. Toute tentative d'accès hors des régions autorisées est interceptée par l'IOMMU et génère une faute I/O.

Kernel DMA Protection dans Windows

Windows implémente la **Kernel DMA Protection** qui utilise l'IOMMU pour protéger spécifiquement contre les périphériques externes malveillants connectés via Thunderbolt, USB4, ou tout port externe avec accès PCIe :

- **Avant la connexion de l'utilisateur** : Tous les ports DMA externes sont bloqués par l'IOMMU. Aucun périphérique Thunderbolt ne peut accéder à la mémoire.
- **Après la connexion** : Les périphériques connectés obtiennent un accès DMA limité à leurs buffers alloués par le driver, via les tables IOMMU configurées par Windows.
- **Nouveaux périphériques** : Un périphérique connecté après le démarrage est placé en quarantaine IOMMU jusqu'à ce que son driver soit chargé et ait explicitement demandé des plages DMA.

Protection DMA pré-démarrage (Server 2025)

Windows Server 2025 introduit la **protection DMA pré-démarrage**, une évolution significative. Sur les versions précédentes, l'IOMMU n'était configuré qu'après le chargement du noyau, laissant une fenêtre de vulnérabilité pendant le processus de démarrage. Sur Server 2025 avec firmware Secured-Core, le firmware UEFI configure l'IOMMU dès les premières phases du démarrage, éliminant cette fenêtre.

Attaques DMA en conditions réelles : études de cas

Pour comprendre la menace concrète que représentent les attaques DMA, examinons plusieurs scénarios réels documentés par des chercheurs et utilisés en pentest physique :

Scénario 1 — Extraction BitLocker via PCILeech : L'attaquant connecte un FPGA (type LambdaConcept PCIe Screamer, environ 400 euros) au port Thunderbolt d'un portable verrouillé. Le FPGA énumère la mémoire physique via DMA et localise la structure BitLocker FVEK en mémoire. En l'absence de protection IOMMU, l'intégralité de la mémoire est lisible, et la clé FVEK peut être extraite en moins de 30 secondes. Avec cette clé, le volume chiffré peut être déchiffré hors ligne. La Kernel DMA Protection via IOMMU neutralise complètement cette attaque : le FPGA ne peut accéder qu'aux pages mémoire explicitement allouées par le driver Thunderbolt, qui ne contiennent pas les clés BitLocker.

Scénario 2 — Injection de code noyau via DMA : Au-delà de la lecture mémoire, le DMA permet également l'écriture. Un attaquant peut modifier en mémoire les structures du noyau pour désactiver les protections (patcher le PatchGuard, désactiver la vérification de signature des drivers) ou injecter un shellcode directement en espace noyau. PCILeech inclut des modules pré-construits pour l'injection de code noyau sur différentes versions de Windows. L'IOMMU empêche l'écriture dans les pages noyau, et HVCI garantit que même si l'écriture réussissait (via un contournement IOMMU), le code injecté ne pourrait pas être exécuté (pages non marquées comme exécutables).

Scénario 3 — Déverrouillage d'écran : L'outil Inception peut déverrouiller un écran de connexion Windows en patchant en mémoire la fonction de vérification du mot de passe dans LSASS. L'attaque prend environ 5 secondes via FireWire ou Thunderbolt. Avec la protection DMA et VBS, cette attaque est doublement bloquée : l'IOMMU empêche l'accès mémoire, et Credential Guard isole le processus d'authentification dans VTL1.

Niveaux de sécurité Thunderbolt

Les contrôleurs Thunderbolt supportent plusieurs niveaux de sécurité, configurés dans le firmware :

- **Niveau 0 (None)** : Aucune restriction — tout périphérique a un accès DMA complet. Vulnérable.
- **Niveau 1 (User Authorization)** : L'utilisateur doit approuver chaque périphérique. Ne protège pas le pré-démarrage.
- **Niveau 2 (Secure Connect)** : Authentification cryptographique du périphérique + approbation utilisateur.
- **Niveau 3 (USB Only)** : Désactive Thunderbolt, seul USB fonctionne via le port Type-C. Pas de DMA possible.

Secured-Core exige au minimum le niveau 2, combiné avec Kernel DMA Protection via IOMMU.

Vérification de la protection DMA

```
# Vérifier Kernel DMA Protection
# msinfo32 > Résumé système > "Protection DMA du noyau"

# Via PowerShell (registre)
Get-ItemProperty -Path "HKLM:\Software\Policies\Microsoft\Windows\Kernel DMA Protection"
-Name DeviceEnumerationPolicy -ErrorAction SilentlyContinue

# Vérifier le support IOMMU
Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGuard
| Select-Object -Property AvailableSecurityProperties
# La valeur 7 dans AvailableSecurityProperties inclut DMA Protection

# Vérifier les périphériques DMA autorisés
Get-PnpDevice | Where-Object { $_.Class -eq "Thunderbolt" } | Format-List

# Stratégie de groupe pour la protection DMA
# Configuration ordinateur > Modèles d'administration > Système > Kernel DMA Protection
# "Politique d'énumération pour les périphériques externes incompatibles DMA Remapping"
```

> **IOMMU bloque les attaques DMA physiques** : Sans IOMMU, un périphérique Thunderbolt malveillant peut lire intégralement la mémoire physique en quelques secondes (clés BitLocker, credentials, code noyau). La Kernel DMA Protection de Secured-Core utilise l'IOMMU pour isoler chaque périphérique dans son espace mémoire. Server 2025 étend cette protection au pré-démarrage. Configurez Thunderbolt en niveau 2 minimum.

System Guard et attestation d'intégrité

System Guard complète l'architecture Secured-Core en fournissant deux capacités essentielles : le Secure Launch (DRTM, déjà détaillé dans la section UEFI) et l'**attestation d'intégrité runtime** — la capacité de vérifier en continu l'état de sécurité d'un système et de conditionner l'accès aux ressources à cet état.

DRTM — Dynamic Root of Trust for Measurement

Le DRTM, implémenté par System Guard Secure Launch, est la première fonction critique de System Guard. Comme détaillé précédemment, il réinitialise la chaîne de confiance à un point précis du démarrage, indépendamment de l'intégrité du firmware en amont. L'implémentation concrète diffère selon le processeur :

- **Intel TXT (Trusted Execution Technology)** : Utilise l'instruction GETSEC[SENTER] pour lancer un Measured Launch Environment (MLE). L'ACM Intel, signé par Intel et vérifié par le processeur, initialise un environnement d'exécution minimal qui mesure l'hyperviseur dans les PCR DRTM. Requis : processeur Intel avec TXT, TPM, et chipset compatible.
- **AMD PSP (Platform Security Processor)** : Utilise l'instruction SKINIT pour démarrer un Secure Loader Block (SLB). Le PSP, un coprocesseur ARM intégré au die AMD, vérifie et

charge le SLB qui mesure l'hyperviseur. Avantage : le PSP est isolé matériellement du processeur x86 principal.

Attestation runtime

Au-delà du démarrage, System Guard effectue des mesures d'intégrité en continu pendant l'exécution du système. Ces mesures incluent :

- **État du noyau** : Vérification périodique que le code noyau n'a pas été modifié en mémoire (HVCI assure la protection en temps réel, System Guard vérifie rétrospectivement).
- **Configuration sécurité** : Vérification que VBS, HVCI, Credential Guard et Secure Boot sont toujours actifs.
- **Drivers chargés** : Liste et hashes des drivers noyau actifs, comparés aux valeurs de référence.
- **État du bootloader** : Mesures du processus de démarrage stockées dans le TCG Event Log.

Ces mesures sont signées par le TPM (via une AIK) et transmises à un service d'attestation distant pour vérification.

Intégration Azure Attestation

Microsoft Azure Attestation (MAA) est le service cloud qui vérifie les quotes TPM et les mesures System Guard. Le flux d'attestation est le suivant :

- Le client Windows collecte les mesures (PCR values, TCG Event Log, AIK certificate)
- Ces données sont envoyées à l'endpoint Azure Attestation (régional, conforme RGPD pour l'Europe)
- MAA vérifie la signature TPM, valide la chaîne de certificats AIK → EK, et compare les PCR aux politiques définies
- MAA retourne un token d'attestation signé, contenant l'état de santé du système
- Ce token est consommé par Microsoft Entra ID (ex-Azure AD) pour les décisions d'accès conditionnel

TCG Event Log et forensique du démarrage

Le TCG Event Log est un journal horodatagé de toutes les mesures effectuées pendant le processus de démarrage et étendues dans les PCR du TPM. Contrairement aux valeurs PCR elles-mêmes (qui sont des hashes cumulés et ne révèlent pas les composants individuels), le TCG Event Log détaille chaque événement de mesure : quel composant a été mesuré, dans quel PCR, et quelle était la valeur du hash. Ce journal est essentiel pour l'investigation forensique.

En cas de suspicion de compromission firmware, l'analyse du TCG Event Log permet de déterminer précisément quel composant du démarrage a changé par rapport à la référence. Par exemple, si un module DXE a été modifié (comme dans le cas de MosaicRegressor), le hash mesuré dans PCR[0] diffère de la valeur attendue, et le TCG Event Log identifie le module DXE spécifique responsable de la déviation. L'outil `tpmtool gatherlogs` de Windows permet de collecter le TCG Event Log pour analyse.

```
# Collecter les logs TCG pour analyse forensique
tpmtool gatherlogs %TEMP%\tpmlogs

# Analyser les mesures de démarrage
Get-WinEvent -LogName "Microsoft-Windows-Measured-Boot/Operational" | Select-Object -First
50

# Vérifier les événements Device Guard
Get-WinEvent -LogName "Microsoft-Windows-DeviceGuard/Operational" -MaxEvents 20 | Format-
List
```

Accès conditionnel basé sur l'attestation

L'intégration System Guard + Azure Attestation + Intune + Entra ID permet un scénario puissant : l'accès conditionnel basé sur l'état de santé matériel du poste. Concrètement :

- Un poste avec Secure Boot désactivé se voit refuser l'accès à SharePoint
- Un poste avec HVCI inactif est redirigé vers un portail de remédiation
- Un poste dont les PCR ne correspondent pas aux valeurs attendues (firmware modifié) est mis en quarantaine
- Un poste pleinement conforme Secured-Core obtient un accès complet aux ressources d'entreprise

Cette approche constitue une implémentation concrète du modèle Zero Trust au niveau matériel — la confiance n'est pas accordée en fonction de la localisation réseau, mais en fonction de l'état vérifiable du poste, attesté par le matériel.

> **Attestation = Zero Trust matériel** : System Guard combine DRTM (confiance au démarrage) et attestation runtime (confiance continue) pour permettre un accès conditionnel basé sur l'état de santé matériel vérifiable. Via Azure Attestation + Intune + Entra ID, un poste non conforme est automatiquement bloqué. C'est l'implémentation la plus avancée du Zero Trust au niveau endpoint.

Secured-Core pour Windows Server 2025

Windows Server 2025 étend l'architecture Secured-Core aux charges de travail serveur, avec des fonctionnalités spécifiques qui répondent aux besoins des datacenters et des infrastructures cloud hybrides. Pour une analyse complète de l'architecture serveur, consultez notre article sur l'architecture système de Windows Server 2025.

VBS activé par défaut

Contrairement à Windows Server 2022 où VBS était optionnel, Windows Server 2025 active VBS par défaut sur les nouvelles installations avec matériel compatible. Cela signifie que HVCI, Credential Guard et System Guard sont actifs en sortie d'usine pour tous les serveurs Secured-Core certifiés.

SMB over QUIC

SMB over QUIC permet l'accès aux partages fichiers Windows via QUIC (UDP, port 443), éliminant le besoin de VPN pour les travailleurs distants. Dans le contexte Secured-Core, SMB over QUIC bénéficie de :

- Chiffrement TLS 1.3 obligatoire (intégré à QUIC)
- Authentification certificat client (Kerberos ou certificat)
- Protection contre les attaques SMB relay (le canal est chiffré de bout en bout)
- Intégration avec les politiques d'accès conditionnel Entra ID

Secured-Core DNS

Windows Server 2025 introduit le DNS over HTTPS (DoH) côté serveur, combiné avec DNSSEC validation. Sur un serveur Secured-Core :

- Le service DNS s'exécute avec VBS actif, protégeant la mémoire du processus DNS
- HVCI empêche l'injection de code dans le service DNS
- Les clés DNSSEC sont protégées par le TPM
- Le DNS-over-HTTPS protège les requêtes clients contre l'interception

Credential Guard avancé dans Server 2025

Windows Server 2025 étend Credential Guard avec des fonctionnalités spécifiques au serveur. La protection des secrets des comptes de service gérés (gMSA — Group Managed Service Accounts) est améliorée : les mots de passe gMSA, qui sont automatiquement renouvelés par Active Directory et stockés en mémoire pour l'authentification des services, bénéficient désormais de la même isolation VTL1 que les credentials utilisateur. Cela élimine un vecteur d'attaque significatif : sur Server 2022, un attaquant avec des privilèges administrateur local pouvait extraire les mots de passe gMSA depuis LSASS et les utiliser pour s'authentifier auprès d'autres services. Avec Server 2025, ces mots de passe sont stockés dans LSAIso et ne sont plus accessibles depuis VTL0.

De plus, Server 2025 introduit la **protection des tickets Kerberos de service-à-service** : les TGS (Ticket Granting Service) utilisés pour l'authentification entre services (par exemple, un serveur IIS s'authentifiant auprès d'un serveur SQL) sont également protégés par VTL1, réduisant la surface d'attaque pour le mouvement latéral entre serveurs.

Hotpatch — Mises à jour sans redémarrage

Le hotpatching est une fonctionnalité majeure de Server 2025 qui permet d'appliquer des patchs de sécurité sans redémarrage du serveur. Dans le contexte Secured-Core, le hotpatch présente des défis spécifiques :

- Les patchs hotfix doivent être compatibles HVCI (signés, pas de code auto-modifiant)
- L'hyperviseur doit autoriser la modification des pages de code noyau pendant le patching
- Les mesures PCR ne changent pas (le hotpatch ne modifie pas le processus de démarrage)

• Disponible via Azure Update Manager ou Windows Server Update Services (WSUS)

Le hotpatching réduit considérablement la fenêtre d'exposition : un serveur peut être patché en quelques secondes au lieu des 15-30 minutes nécessaires pour un redémarrage complet.

Différences d'exigences matérielles

Exigence	Secured-Core Client (Windows 11)	Secured-Core Server (Server 2025)
TPM	2.0 obligatoire	2.0 obligatoire
IOMMU	Obligatoire	Obligatoire
DRTM	Recommandé	Obligatoire pour Azure Stack HCI
RAM minimum	4 Go	16 Go (VBS réserve ~1 Go pour VTL1)
VBS par défaut	Oui (depuis 22H2 nouvelles installations)	Oui (nouvelles installations Server 2025)
Hotpatch	Non disponible	Disponible (Azure Arc ou licence SA)
SMB over QUIC	Client uniquement	Client + Serveur
Protection DMA pré-boot	Dépend du firmware OEM	Exigé pour certification Secured-Core serveur

Azure Stack HCI Secured-Core

Azure Stack HCI (Hyperconverged Infrastructure) est la plateforme de Microsoft pour le cloud hybride on-premises. La certification Secured-Core est obligatoire pour Azure Stack HCI, ce qui signifie que tous les nœuds HCI doivent implémenter la pile complète : TPM 2.0, DRTM, VBS, HVCI, Credential Guard, protection DMA. De plus, Azure Stack HCI ajoute :

• Attestation obligatoire via Azure Attestation pour chaque nœud
• Chiffrement SMB 3.1.1 entre nœuds (trafic de stockage)
• BitLocker sur les volumes CSV (Cluster Shared Volumes)
• Intégration native avec Microsoft Defender for Cloud pour la surveillance continue

Implications pour le pentest et la sécurité offensive

L'architecture Secured-Core transforme fondamentalement le paysage de la sécurité offensive. Chaque protection neutralise ou complique significativement des techniques d'attaque spécifiques qui font partie de la boîte à outils standard des pentesters et red teams. Comprendre ces impacts est essentiel pour adapter les méthodologies d'évaluation. Pour les techniques de persistance en environnement Windows Server 2025, consultez notre article dédié sur les techniques de persistance Windows Server 2025.

Matrice d'impact : protection vs technique d'attaque

Technique d'attaque	Protection Secured-Core	Impact	Contournement possible ?
Mimikatz sekurlsa::logonpasswords	Credential Guard	Bloqué (handles opaques)	Pivoter vers tokens SSO, DPAPI, comptes locaux
Pass-the-Hash (NTLM domaine)	Credential Guard	Bloqué (hashes dans VTL1)	PtH sur comptes locaux (SAM) reste possible
Pass-the-Ticket (Kerberos)	Credential Guard	Bloqué (TGTs dans VTL1)	Kerberoasting, AS-REP Roasting non affectés
Driver rootkit (chargement noyau)	HVCI	Bloqué (signature requise)	Exploit de driver légitime signé (BYOVD)
Shellcode noyau (pool spraying)	HVCI (W^X)	Bloqué (pas de pages RWX noyau)	ROP/JOP sur code signé existant (très complexe)
Bootkit UEFI (BlackLotus)	DRTM + Secure Boot	Atténué (DRTM réinitialise la confiance)	Vulnérabilité dans l'ACM/ hyperviseur (zéro-day)
DMA via Thunderbolt (PCILeech)	Kernel DMA Protection	Bloqué (IOMMU isole les périphériques)	Vulnérabilité dans le driver IOMMU (rare)
Modification firmware UEFI	Secure Boot + Config Lock	Bloqué (firmware signé, config verrouillée)	Vulnérabilité SMM ou flash SPI physique
BitLocker bypass (cold boot)	TPM + PIN pré-démarrage	Atténué (PIN empêche le déverrouillage automatique)	Si pas de PIN : TPM sniffing sur dTPM
DCSync (réplication AD)	Aucune (niveau AD)	Non affecté	Technique toujours viable avec droits suffisants
Kerberoasting	Aucune (niveau AD)	Non affecté	Technique toujours viable
Token impersonation	Aucune (niveau OS)	Non affecté	Technique toujours viable avec privilèges adéquats

Ce qui fonctionne encore dans un environnement pleinement Secured-Core

Malgré la robustesse de Secured-Core, plusieurs vecteurs d'attaque restent viables :

- **Attaques Active Directory pur** : Kerberoasting, AS-REP Roasting, DCSync, exploitation des délégations Kerberos, abus ADCS (Active Directory Certificate Services). Ces attaques opèrent au niveau du protocole et du service d'annuaire, hors du périmètre Secured-Core.
- **Exploitation applicative** : Les vulnérabilités dans les applications utilisateur (navigateurs, Office, services web) ne sont pas affectées par Secured-Core. Un RCE dans une application donne toujours un accès utilisateur.

- **Token manipulation** : L'impersonation de tokens, la manipulation de privileges, et l'exploitation de services mal configurés (SeImpersonatePrivilege → Potato attacks) restent fonctionnelles.
- **DPAPI et secrets locaux** : Les Master Keys DPAPI, les mots de passe enregistrés dans les navigateurs, les credentials Wi-Fi et VPN ne sont pas protégés par Credential Guard.
- **BYOVD (Bring Your Own Vulnerable Driver)** : Un attaquant avec des privilèges administrateur peut charger un driver légitime mais vulnérable (signé WHQL) et exploiter la vulnérabilité pour obtenir une exécution noyau. HVCI vérifie la signature, pas les vulnérabilités. Microsoft atténue ce vecteur via la Microsoft Vulnerable Driver Blocklist, mais la liste n'est pas exhaustive.
- **Social engineering et phishing** : Secured-Core ne protège pas contre l'utilisateur qui clique sur un lien malveillant ou fournit ses credentials à un site de phishing.

Adaptations red team

Les red teams opérant contre des environnements Secured-Core doivent adapter leur méthodologie de manière fondamentale. L'approche traditionnelle — compromettre un poste, extraire les credentials, pivoter latéralement — est considérablement affaiblie. Voici les adaptations recommandées :

- **Privilégier les attaques Active Directory** : Investir dans la reconnaissance AD approfondie, l'exploitation ADCS (Active Directory Certificate Services — ESC1 à ESC13), les chemins d'attaque via BloodHound/ADExplorer, et l'abus des délégations Kerberos (constrained delegation, resource-based constrained delegation). Ces techniques sont indépendantes de Secured-Core car elles opèrent au niveau du protocole et du service d'annuaire. Pour approfondir ces techniques, consultez notre guide complet du pentest Active Directory.
- **Cibler les systèmes non Secured-Core** : Dans un réseau mixte, identifier les postes sans Credential Guard ou HVCI pour le mouvement latéral. Un seul système non protégé peut fournir des credentials exploitables sur l'ensemble du domaine. Utilisez des requêtes LDAP pour identifier les systèmes d'exploitation anciens, et ciblez en priorité les serveurs applicatifs legacy qui sont souvent les derniers à être migrés.
- **Exploiter DPAPI** : Extraire les Master Keys DPAPI pour déchiffrer les credentials Chrome/Edge, les mots de passe enregistrés dans les gestionnaires intégrés, les clés Wi-Fi, les mots de passe de connexion VPN, et les credentials RDP enregistrés. SharpDPAPI et ses variantes (DonPAPI, Mimikatz dpapi::*) restent pleinement fonctionnels sous Secured-Core car DPAPI n'est pas protégé par Credential Guard. Cette lacune est significative : un utilisateur qui enregistre son mot de passe Active Directory dans Chrome expose ce mot de passe via DPAPI, même avec Credential Guard actif.
- **BYOVD pour l'accès noyau** : Utiliser des drivers vulnérables signés pour contourner HVCI de manière limitée. HVCI vérifie la signature mais pas les vulnérabilités. Un driver signé WHQL qui expose une primitive de lecture/écriture mémoire physique (comme RTCore64.sys) est accepté par HVCI mais permet des opérations privilégiées. La Vulnerable Driver Blocklist de Microsoft réduit ce vecteur mais ne l'élimine pas — de nouveaux drivers vulnérables sont découverts régulièrement.

- **Living off the Land (LOLBins)** : Maximiser l'utilisation d'outils légitimes signés Microsoft. PowerShell, WMI, CertUtil, BITS, MSBuild, Regsvr32, WMIC ne sont pas bloqués par HVCI car ils sont signés. L'utilisation de WDAC en parallèle réduit ce vecteur, mais les LOLBins signés Microsoft sont rarement bloqués par les politiques WDAC standard car ils sont essentiels au fonctionnement du système.
- **Attaques sur les services cloud et SaaS** : Pivoter vers l'exploitation des tokens OAuth, des cookies de session Azure/Entra ID, et des configurations cloud mal sécurisées. Un token OAuth volé via un navigateur compromis donne accès aux ressources cloud sans déclencher aucune protection Secured-Core. Roadtx, AADInternals et TokenTactics sont des outils spécifiquement conçus pour exploiter ce vecteur.
- **Exploitation des applications web internes** : Les vulnérabilités applicatives (injection SQL, SSRF, déserialisation) dans les applications internes ne sont pas affectées par Secured-Core. Un SSRF contre un serveur interne peut fournir un accès initial que Secured-Core ne peut pas prévenir, car il s'agit d'une exploitation applicative légitime du point de vue du système d'exploitation.

Scénario red team complet en environnement Secured-Core

Pour illustrer concrètement les adaptations nécessaires, voici un scénario red team réaliste dans un environnement entièrement Secured-Core :

- **Phase 1 — Accès initial** : Phishing ciblé avec un document contenant une macro ou un lien vers une page de phishing Evilginx2 capturant les tokens MFA. Secured-Core ne protège pas contre le phishing.
- **Phase 2 — Exécution** : Payload .NET exécuté en mémoire via PowerShell ou Assembly.Load. HVCI ne bloque pas le code utilisateur (Ring 3) — seul le code noyau est validé.
- **Phase 3 — Persistance** : Scheduled task, COM hijacking, ou DLL side-loading. Pas de driver rootkit (bloqué par HVCI), mais les techniques utilisateur restent viables. Pour les techniques de persistance détaillées, consultez notre article sur les techniques de persistance Windows Server 2025.
- **Phase 4 — Collecte de credentials** : DPAPI dump (SharpDPAPI) pour les mots de passe Chrome et les credentials enregistrés. Mimikatz ne fonctionne pas pour les credentials domaine (Credential Guard), mais les credentials locaux et DPAPI sont accessibles.
- **Phase 5 — Mouvement latéral** : Exploitation des délégations Kerberos, ou ciblage de systèmes non Secured-Core dans le réseau. Utilisation des credentials DPAPI pour s'authentifier auprès de services internes.
- **Phase 6 — Escalade de privilèges** : Kerberoasting des comptes de service avec des mots de passe faibles, exploitation ADCS (ESC1 — template mal configuré), ou abus des droits AD (GenericAll, WriteDACL). Ces chemins ne sont pas affectés par Secured-Core.

> **Secured-Core ne rend pas le pentest impossible** : Il neutralise les techniques post-exploitation classiques (Mimikatz, rootkits, DMA) mais n'affecte pas les attaques AD, l'exploitation applicative, DPAPI, ou le social engineering. Les red teams doivent pivoter vers les vecteurs hors périmètre Secured-Core. Le BYOVD reste un contournement partiel de HVCI tant que la Vulnerable Driver Blocklist n'est pas exhaustive.

Déploiement et configuration

Le déploiement de Secured-Core en entreprise nécessite une approche méthodique, de la vérification matérielle à la validation post-déploiement. Voici le guide complet pour une implémentation réussie.

Stratégie de déploiement par phases

Le déploiement de Secured-Core ne doit jamais être effectué en « big bang » sur l'ensemble du parc. Les incompatibilités de drivers, les applications legacy qui dépendent de WDigest ou NTLMv1, et les workflows métier spécifiques peuvent causer des dysfonctionnements critiques si Secured-Core est activé sans préparation. La stratégie recommandée comporte quatre phases distinctes.

Phase 1 — Audit et inventaire (2-4 semaines) : Inventorier le parc matériel pour identifier les machines compatibles Secured-Core. Vérifier la version du firmware UEFI de chaque modèle. Exécuter le Device Guard Readiness Tool sur un échantillon représentatif pour identifier les drivers incompatibles HVCI. Recenser les applications qui utilisent WDigest, NTLMv1, ou des mécanismes d'authentification legacy. Documenter les exceptions et les plans de remédiation pour chaque incompatibilité.

Phase 2 — Pilote (4-6 semaines) : Déployer Secured-Core sur un groupe pilote de 50 à 100 postes, représentatif des différents profils métier. Activer VBS et HVCI en mode audit (journalise les blocages sans les appliquer). Activer Credential Guard sans verrouillage UEFI (permettant un retour arrière rapide). Surveiller les journaux d'événements pendant 4 semaines pour identifier les problèmes. Résoudre les incompatibilités identifiées.

Phase 3 — Déploiement progressif (8-12 semaines) : Basculer HVCI en mode enforcement sur le groupe pilote. Déployer sur des groupes successifs de 200 à 500 postes, en commençant par les populations les moins risquées (postes standards, pas de logiciels métier critiques). Activer le verrouillage UEFI pour Credential Guard et HVCI après validation sur chaque groupe. Configurer les politiques d'accès conditionnel en mode « rapport seul » (report-only) pour valider sans bloquer.

Phase 4 — Consolidation et enforcement (4 semaines) : Basculer les politiques d'accès conditionnel en mode enforcement. Déployer WDAC en mode audit, puis en enforcement. Gérer les exceptions documentées (postes incompatibles, applications legacy isolées dans des VMs). Mettre en place la surveillance continue via Microsoft Defender for Endpoint et les alertes Secured-Core.

Checklist de déploiement

- **1. Inventaire matériel** : Vérifier que les postes sont certifiés Secured-Core ou disposent du matériel nécessaire (TPM 2.0, virtualisation matérielle, IOMMU)
- **2. Mise à jour firmware** : Appliquer la dernière version du firmware UEFI OEM pour chaque modèle
- **3. Configuration BIOS/UEFI** : Activer Secure Boot, activer la virtualisation (VT-x/AMD-V), activer VT-d/AMD-Vi, configurer le TPM en mode 2.0
- **4. Installation Windows** : Installer Windows 11 24H2+ ou Server 2025 en mode UEFI (pas de Legacy/CSM)
- **5. Activation VBS** : Vérifier que VBS est actif (activé par défaut sur les nouvelles installations)
- **6. Activation HVCI** : Vérifier que HVCI est actif. Tester la compatibilité des drivers avant l'activation en production
- **7. Activation Credential Guard** : Activer avec verrouillage UEFI pour les postes à haut risque
- **8. Configuration BitLocker** : Activer BitLocker avec TPM + PIN pré-démarrage
- **9. Configuration DMA Protection** : Vérifier que Kernel DMA Protection est active, configurer la politique Thunderbolt
- **10. Configuration WDAC** : Définir et déployer une politique WDAC en mode audit, puis en mode enforcement
- **11. Inscription attestation** : Configurer Azure Attestation et les politiques d'accès conditionnel Intune
- **12. Surveillance** : Configurer Microsoft Defender for Endpoint pour les alertes Secured-Core
- **13. Validation** : Exécuter le script de vérification PowerShell sur un échantillon représentatif
- **14. Documentation** : Documenter les exceptions (postes incompatibles, drivers problématiques)
- **15. Formation** : Former le SOC aux alertes spécifiques Secured-Core et aux procédures de réponse

Configuration via Intune (Microsoft Endpoint Manager)

Intune est l'outil recommandé pour le déploiement à grande échelle des politiques Secured-Core :

- **Endpoint Security > Account Protection** : Credential Guard (Enable with UEFI lock)
- **Endpoint Security > Account Protection** : VBS (Enable VBS with Secure Boot and DMA Protection)
- **Endpoint Security > Endpoint Detection and Response** : Microsoft Defender for Endpoint integration
- **Device Configuration > Endpoint Protection** : Windows Defender Application Guard, Exploit Guard
- **Compliance Policies** : Require BitLocker, Require Secure Boot, Require Code Integrity, Require TPM
- **Conditional Access** : Require device compliance (inclut l'attestation de santé)

Configuration via GPO

Paramètre GPO	Chemin	Valeur recommandée
VBS	Computer Configuration\Admin Templates\System\Device Guard\Turn on VBS	Enabled - Secure Boot and DMA Protection
HVCI	Computer Configuration\Admin Templates\System\Device Guard\Turn on VBS > Virtualization Based Protection of Code Integrity	Enabled with UEFI lock
Credential Guard	Computer Configuration\Admin Templates\System\Device Guard\Turn on VBS > Credential Guard Configuration	Enabled with UEFI lock
Secure Launch	Computer Configuration\Admin Templates\System\Device Guard\Turn on VBS > Secure Launch Configuration	Enabled
BitLocker (OS Drive)	Computer Configuration\Admin Templates\Windows Components\BitLocker Drive Encryption\Operating System Drives	Require additional authentication at startup: Require PIN with TPM
DMA Protection	Computer Configuration\Admin Templates\System\Kernel DMA Protection	Enumeration policy: Block all
Remote Credential Guard	Computer Configuration\Admin Templates\System\Credentials Delegation\Restrict delegation of credentials	Enabled: Require Remote Credential Guard

Clés de registre

```
# VBS - Activation
reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
EnableVirtualizationBasedSecurity /t REG_DWORD /d 1 /f
reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
RequirePlatformSecurityFeatures /t REG_DWORD /d 3 /f

# HVCI - Activation avec verrouillage UEFI
reg add
"HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegri
ty" /v Enabled /t REG_DWORD /d 1 /f
reg add
"HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegri
ty" /v Locked /t REG_DWORD /d 1 /f

# Credential Guard - Activation avec verrouillage UEFI
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v LsaCfgFlags /t REG_DWORD /d 2 /f
# 0 = Désactivé, 1 = Activé sans verrouillage, 2 = Activé avec verrouillage UEFI

# System Guard Secure Launch
reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\SystemGuard" /v
Enabled /t REG_DWORD /d 1 /f

# Désactiver WDigest (doublement sûr)
reg add "HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest" /v
UseLogonCredential /t REG_DWORD /d 0 /f

# Activer LSA Protection (RunAsPPL) en complément de Credential Guard
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RunAsPPL /t REG_DWORD /d 1 /f
```

Script de vérification PowerShell complet

```
# ============================================
# Script de vérification Secured-Core
# Ayi NEDJIMI Consultants - 2025
# ============================================

Write-Host "=== Vérification Secured-Core ===" -ForegroundColor Cyan

# 1. TPM
Write-Host "`n[1] TPM 2.0" -ForegroundColor Yellow
$tpm = Get-Tpm
Write-Host "  Présent : $($tpm.TpmPresent)"
Write-Host "  Actif   : $($tpm.TpmReady)"
$tpmInfo = Get-CimInstance -Namespace "root\cimv2\Security\MicrosoftTpm" -ClassName
Win32_Tpm -ErrorAction SilentlyContinue
if ($tpmInfo) {
    Write-Host "  Version : $($tpmInfo.SpecVersion)"
}

# 2. Secure Boot
Write-Host "`n[2] Secure Boot" -ForegroundColor Yellow
try {
    $sb = Confirm-SecureBootUEFI
    Write-Host "  Secure Boot : $sb"
} catch {
    Write-Host "  Secure Boot : Non supporté (mode Legacy ?)" -ForegroundColor Red
}

# 3. VBS
Write-Host "`n[3] Virtualization-Based Security" -ForegroundColor Yellow
$dg = Get-CimInstance -ClassName Win32_DeviceGuard -Namespace
root\Microsoft\Windows\DeviceGuard
$vbsStatus = switch ($dg.VirtualizationBasedSecurityStatus) {
    0 { "Désactivé" }
    1 { "Activé (non démarré)" }
    2 { "En cours d'exécution" }
}
Write-Host "  VBS : $vbsStatus"

# 4. HVCI
Write-Host "`n[4] HVCI (Memory Integrity)" -ForegroundColor Yellow
$hvci = $dg.SecurityServicesRunning -contains 2
Write-Host "  HVCI actif : $hvci"

# 5. Credential Guard
Write-Host "`n[5] Credential Guard" -ForegroundColor Yellow
$cg = $dg.SecurityServicesRunning -contains 1
Write-Host "  Credential Guard actif : $cg"
$lsaCfg = (Get-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Lsa" -Name
LsaCfgFlags -ErrorAction SilentlyContinue).LsaCfgFlags
$cgLock = switch ($lsaCfg) {
    0 { "Désactivé" }
    1 { "Activé sans verrouillage UEFI" }
    2 { "Activé avec verrouillage UEFI" }
}
Write-Host "  Mode : $cgLock"

# 6. System Guard
Write-Host "`n[6] System Guard Secure Launch" -ForegroundColor Yellow
$sg = $dg.SecurityServicesRunning -contains 3
Write-Host "  System Guard actif : $sg"

# 7. DMA Protection
```

```
Write-Host "`n[7] Protection DMA" -ForegroundColor Yellow
$dmaProtection = $dg.AvailableSecurityProperties -contains 7
Write-Host "  DMA Protection disponible : $dmaProtection"

# 8. BitLocker
Write-Host "`n[8] BitLocker" -ForegroundColor Yellow
$bl = Get-BitLockerVolume -MountPoint "C:" -ErrorAction SilentlyContinue
if ($bl) {
    Write-Host "  Protection : $($bl.ProtectionStatus)"
    Write-Host "  Méthode chiffrement : $($bl.EncryptionMethod)"
    Write-Host "  Protecteurs : $($bl.KeyProtector.KeyProtectorType -join ', ')"
} else {
    Write-Host "  BitLocker non configuré sur C:" -ForegroundColor Red
}

# 9. LSA Protection (RunAsPPL)
Write-Host "`n[9] LSA Protection (RunAsPPL)" -ForegroundColor Yellow
$ppl = (Get-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Lsa" -Name RunAsPPL
-ErrorAction SilentlyContinue).RunAsPPL
Write-Host "  RunAsPPL : $(if ($ppl -eq 1) { 'Actif' } else { 'Inactif' })"

# 10. Résumé
Write-Host "`n=== Résumé ===" -ForegroundColor Cyan
$score = 0
if ($tpm.TpmReady) { $score++ }
if ($sb -eq $true) { $score++ }
if ($dg.VirtualizationBasedSecurityStatus -eq 2) { $score++ }
if ($hvci) { $score++ }
if ($cg) { $score++ }
if ($sg) { $score++ }
if ($bl.ProtectionStatus -eq "On") { $score++ }
if ($ppl -eq 1) { $score++ }

Write-Host "Score Secured-Core : $score / 8" -ForegroundColor $(if ($score -ge 7)
{ "Green" } elseif ($score -ge 5) { "Yellow" } else { "Red" })
```

Surveillance et alertes Secured-Core

Un déploiement Secured-Core n'est complet que s'il est accompagné d'une surveillance active. Microsoft Defender for Endpoint intègre des alertes spécifiques à l'écosystème Secured-Core qui méritent une attention particulière du SOC :

- **Alerte « VBS integrity violation »** : Indique une tentative de compromission de l'intégrité VBS. Cet événement est extrêmement rare et doit être traité comme un incident critique (probable tentative d'attaque avancée ciblant l'hyperviseur).
- **Alerte « HVCI policy violation »** : Un driver non signé ou incompatible a tenté de se charger. En mode audit, cela génère un log sans blocage. En mode enforcement, le driver est bloqué. Cet événement peut indiquer un BYOVD ou simplement un driver légitime mal configuré.
- **Alerte « Credential Guard isolation failure »** : LSAIso a échoué à initialiser correctement. Cela peut indiquer un problème de configuration VBS ou une tentative de manipulation du processus d'isolation.
- **Alerte « Secure Boot state change »** : L'état Secure Boot a changé depuis la dernière attestation. Potentiellement très grave — indique que quelqu'un a accédé au BIOS setup et modifié la configuration.

- **Alerte « TPM attestation failure »** : Le poste a échoué l'attestation de santé. Les PCR ne correspondent plus aux valeurs attendues. Cela peut indiquer une mise à jour firmware (bénigne) ou une compromission firmware (critique).

Il est recommandé de configurer des règles de détection personnalisées dans Azure Sentinel (Microsoft Sentinel) pour corréler les alertes Secured-Core avec d'autres indicateurs de compromission. Par exemple, un échec d'attestation TPM combiné à une connexion réseau inhabituelle depuis le même poste devrait déclencher une investigation immédiate.

Dépannage des problèmes courants

- **VBS ne démarre pas** : Vérifier que la virtualisation matérielle est activée dans le BIOS. Vérifier qu'Hyper-V n'est pas désactivé via `bcdedit /set hypervisorlaunchtype auto`. Vérifier les logs : `Get-WinEvent -LogName "Microsoft-Windows-DeviceGuard/Operational"`.
- **HVCI casse un driver** : Identifier le driver incompatible via le Device Guard Readiness Tool. Vérifier si une version HVCI-compatible existe. En dernier recours, exclure temporairement le driver et le signaler au fournisseur.
- **Credential Guard désactive NTLMv1** : Normal — Credential Guard refuse les protocoles d'authentification faibles. Migrer les applications qui dépendent de NTLMv1 vers NTLMv2 ou Kerberos.
- **BitLocker demande la clé de récupération après mise à jour firmware** : Les PCR ont changé suite à la mise à jour. Suspendre BitLocker (`Suspend-BitLocker`) avant les mises à jour firmware, puis le réactiver.
- **Performances dégradées avec VBS** : Vérifier que le processeur supporte MBEC (Intel 11e gen+, AMD Zen 3+). Mettre à jour vers Windows 11 24H2 pour bénéficier des optimisations VBS.

Comparaison avec les approches concurrentes

Secured-Core n'est pas la seule approche de sécurité matérielle dans l'industrie. Il est utile de le comparer avec les initiatives concurrentes pour comprendre ses avantages et ses limites spécifiques.

Apple Silicon et le modèle Apple

Apple a adopté une approche différente avec les puces M1/M2/M3/M4 : contrôle vertical intégral du matériel et du logiciel. Le Secure Enclave Processor (SEP) d'Apple joue un rôle similaire au TPM + Pluton, mais avec une intégration plus profonde car Apple contrôle à la fois la puce et le système d'exploitation. La différence majeure est que l'écosystème PC est hétérogène (multiples OEM, multiples processeurs), ce qui rend la standardisation plus complexe mais offre plus de flexibilité aux entreprises. Secured-Core est la réponse de Microsoft à cette intégration verticale, mais dans un contexte horizontal multi-fournisseurs.

ChromeOS et le modèle Verified Boot

Google ChromeOS implémente un modèle de démarrage vérifié radical : le firmware, le noyau et le système de fichiers racine sont tous vérifiés cryptographiquement à chaque démarrage, et le système de fichiers racine est en lecture seule. Toute modification déclenche une réinstallation automatique. Ce modèle est extrêmement robuste contre les rootkits et les bootkits, mais incompatible avec l'écosystème applicatif Windows. Secured-Core adopte une approche plus pragmatique : il ne rend pas le système immuable, mais il isole les composants critiques dans des environnements matériellement protégés.

Linux et Confidential Computing

L'écosystème Linux adopte progressivement des mécanismes similaires. Le Confidential Computing via AMD SEV-SNP (Secure Nested Paging) et Intel TDX (Trust Domain Extensions) permet de créer des machines virtuelles dont la mémoire est chiffrée et inaccessible même à l'hyperviseur. IMA (Integrity Measurement Architecture) fournit des mesures d'intégrité similaires aux PCR du TPM. Secure Boot est supporté via shim + GRUB2 + noyau signé. Cependant, l'intégration est moins cohérente que Secured-Core car l'écosystème Linux est fragmenté entre les distributions, les bootloaders et les configurations noyau.

Questions fréquentes sur Secured-Core

Secured-Core est-il disponible sur tous les PC Windows 11 ?

Non. Secured-Core nécessite une certification matérielle spécifique de l'OEM. Tous les PC Windows 11 disposent de TPM 2.0 (exigence minimale), mais la pile complète Secured-Core (DRTM, verrouillage firmware, activation par défaut de VBS/HVCI/Credential Guard) n'est disponible que sur les modèles certifiés. Cependant, les composants individuels (VBS, HVCI, Credential Guard) peuvent être activés manuellement sur tout PC Windows 11 disposant du matériel nécessaire, même sans certification Secured-Core. La différence est que les PC certifiés Secured-Core ont ces protections activées en sortie d'usine et bénéficient du DRTM et du verrouillage firmware.

Quel est l'impact réel sur les performances en environnement de production ?

L'impact varie selon la charge de travail et le matériel. Sur des processeurs récents (Intel 12e gen+, AMD Zen 4+) avec MBEC, l'overhead est généralement de 2 à 5 % pour les charges bureautiques. Les charges intensives en I/O ou en création de processus (compilation, CI/CD) peuvent voir 5 à 10 % d'overhead. Pour les serveurs de base de données ou les hyperviseurs, il est recommandé de benchmarker spécifiquement avant le déploiement. Microsoft a optimisé VBS considérablement dans Windows 11 24H2 et Server 2025. Le consensus général en 2025 est que le coût en performances est acceptable au regard du gain de sécurité, surtout avec du matériel récent.

Peut-on désactiver Secured-Core après l'activation ?

Cela dépend du mode de déploiement. Si VBS, HVCI et Credential Guard sont activés **sans verrouillage UEFI** (LsaCfgFlags = 1, HVCI Locked = 0), ils peuvent être désactivés via GPO, registre ou Intune. Si activés **avec verrouillage UEFI** (recommandé en production), la désactivation nécessite un accès physique au BIOS et une procédure spécifique de suppression des variables UEFI. Cette procédure est documentée par Microsoft mais intentionnellement complexe pour empêcher la désactivation à distance par un attaquant. Dans un contexte Secured-Core certifié, le verrouillage UEFI est la configuration recommandée.

Comment gérer les applications anciennes incompatibles avec HVCI ?

L'approche recommandée est progressive : d'abord activer HVCI en **mode audit** (enregistre les incompatibilités sans bloquer) pour identifier les drivers et applications problématiques. Ensuite, rechercher des mises à jour compatibles HVCI auprès des éditeurs. Pour les applications legacy sans alternative, envisager l'isolation dans une VM Hyper-V dédiée (sans HVCI dans la VM invitée) tout en maintenant HVCI sur l'hôte. En dernier recours, créer une exception WDAC pour le driver spécifique, en acceptant le risque résiduel documenté. Ne jamais désactiver HVCI globalement pour un seul driver incompatible.

Secured-Core protège-t-il contre les ransomwares ?

Secured-Core n'est pas spécifiquement conçu contre les ransomwares, mais il réduit considérablement la surface d'attaque. HVCI empêche le chargement de drivers noyau malveillants (utilisés par certains ransomwares avancés pour désactiver les EDR). Credential Guard limite le mouvement latéral en protégeant les credentials domaine. BitLocker avec TPM + PIN empêche le démarrage sur un média de récupération non autorisé. Cependant, un ransomware qui chiffre les fichiers utilisateur en espace utilisateur n'est pas directement bloqué par Secured-Core — c'est le rôle des solutions EDR, des sauvegardes, et des contrôles d'accès. Secured-Core est un composant d'une stratégie de défense en profondeur, pas une solution unique.

Quel est le coût total de déploiement de Secured-Core en entreprise ?

Le coût se décompose en plusieurs catégories. Le surcoût matériel est modéré : les postes certifiés Secured-Core coûtent en moyenne 5 à 15 % de plus que leurs équivalents non certifiés, principalement en raison du TPM discret haut de gamme et du firmware Secured-Core. Cependant, ce surcoût tend à disparaître car les composants Secured-Core deviennent standard sur les gammes professionnelles. Le coût principal est opérationnel : l'audit de compatibilité des drivers et applications (2 à 4 semaines pour un parc de 1 000 postes), la configuration des politiques Intune/GPO (1 à 2 semaines), la formation des équipes IT et SOC (3 à 5 jours), et le support initial pendant la phase de transition (incompatibilités drivers, questions BitLocker). Pour un déploiement de 500 postes, comptez un budget projet de 15 000 à 30 000 euros en intégration et accompagnement, amortissable sur la durée de vie du parc (4-5 ans). Le retour sur investissement se calcule en réduction du risque de compromission majeure, dont le coût moyen en France dépasse 800 000 euros selon l'ANSSI.

Secured-Core est-il compatible avec la virtualisation imbriquée et les conteneurs ?

Oui, avec des nuances. VBS utilise l'hyperviseur Hyper-V en mode racine, ce qui est compatible avec Hyper-V pour la virtualisation serveur et avec Windows Sandbox pour l'isolation d'applications. La virtualisation imbriquée (Hyper-V dans Hyper-V) est supportée depuis Windows Server 2016 et fonctionne avec VBS actif, bien que les performances soient réduites. Docker Desktop pour Windows utilise WSL2 (qui s'appuie sur Hyper-V), entièrement compatible avec VBS. Les conteneurs Windows Server (process isolation) fonctionnent normalement avec HVCI. Le cas problématique est VMware Workstation / VirtualBox : ces hyperviseurs tiers nécessitent un accès direct aux extensions de virtualisation du processeur, ce qui entre en conflit avec Hyper-V/VBS. VMware Workstation 15.5+ résout ce problème en fonctionnant en tant que client Hyper-V, mais avec un overhead supplémentaire. VirtualBox 7.0+ propose également un mode compatible Hyper-V.

Conclusion

L'architecture Secured-Core représente un changement de paradigme fondamental dans la sécurité des systèmes Windows. En déplaçant les protections critiques sous le système d'exploitation — dans le matériel, le firmware et l'hyperviseur — Microsoft a créé un modèle où la compromission du noyau n'est plus suffisante pour atteindre les secrets les plus sensibles.

Les sept piliers de Secured-Core (TPM 2.0, Secure Boot, UEFI Config Lock, DMA Protection, VBS, HVCI, Credential Guard) forment une défense en profondeur où chaque couche renforce les autres. Le TPM fournit la racine de confiance cryptographique. Secure Boot et DRTM garantissent l'intégrité du démarrage. VBS crée un monde sécurisé matériellement isolé. HVCI protège le code noyau. Credential Guard protège les secrets d'authentification. Et l'IOMMU neutralise les attaques physiques par DMA.

Pour les équipes de sécurité défensive, Secured-Core constitue le socle de sécurité le plus robuste jamais proposé par Microsoft. Pour les équipes de sécurité offensive, il impose une évolution des méthodologies vers les vecteurs non couverts : attaques AD, exploitation applicative, DPAPI, et social engineering.

En 2026, Secured-Core n'est plus une option de luxe mais une exigence minimale pour toute organisation traitant des données sensibles. Le coût en performances est désormais négligeable avec le matériel récent, et les outils de déploiement (Intune, GPO, SCCM) sont matures. L'investissement initial en compatibilité des drivers et en formation est largement compensé par la réduction drastique de la surface d'attaque post-exploitation.

La recommandation d'Ayi NEDJIMI Consultants est claire : déployez Secured-Core sur l'ensemble de votre parc, privilégiez le verrouillage UEFI, exigez la certification Secured-Core dans vos appels d'offres matériels, et intégrez l'attestation de santé dans vos politiques d'accès conditionnel Zero Trust.

Cependant, Secured-Core ne doit pas créer un faux sentiment de sécurité. Comme notre analyse des implications offensives l'a démontré, de nombreux vecteurs d'attaque restent viables — les attaques Active Directory, l'exploitation applicative, le social engineering, DPAPI, et le BYOVD.

Secured-Core élève considérablement la barre pour les attaquants qui dépendent du post-exploitation classique (Mimikatz, rootkits, DMA), mais les attaquants sophistiqués pivoter ont vers les vecteurs non couverts. Une stratégie de défense en profondeur complète doit combiner Secured-Core (sécurité matérielle), WDAC (contrôle applicatif), Microsoft Defender for Endpoint (EDR), Azure Sentinel (SIEM), une hygiène Active Directory rigoureuse (tiering model, LAPS, délégations minimales), et un programme de sensibilisation des utilisateurs au phishing.

L'avenir de la sécurité matérielle Windows s'annonce encore plus intégré. Microsoft Pluton, déjà présent dans les processeurs AMD Ryzen 6000+ et Qualcomm Snapdragon 8cx Gen 3+, promet d'éliminer les derniers vecteurs d'attaque physique en intégrant le TPM directement dans le die du processeur. Windows 12, attendu pour 2026-2027, devrait renforcer encore l'intégration VBS avec des performances quasi natives grâce aux optimisations matérielles. Et le Confidential Computing (AMD SEV-SNP, Intel TDX), qui chiffre la mémoire même contre l'hyperviseur, étend le modèle Secured-Core au cloud, où la confiance dans l'infrastructure physique ne peut jamais être garantie.

Ressources

- Microsoft Learn — Secured-Core PC Requirements
- Microsoft Learn — Virtualization-Based Security
- Microsoft Learn — Credential Guard Overview
- Microsoft Learn — Kernel DMA Protection
- Microsoft Learn — Secured-Core Server
- NIST NVD — CVE-2022-21894 (Baton Drop / BlackLotus)
- NIST SP 800-147 — BIOS Protection Guidelines
- CERT-FR — Centre gouvernemental de veille, d'alerte et de réponse aux attaques informatiques
- TCG — TPM 2.0 Library Specification
- ESET Research — BlackLotus UEFI Bootkit Analysis
- Ayi NEDJIMI Consultants — HVCI Deep Dive : Intégrité du code par l'hyperviseur
- Ayi NEDJIMI Consultants — Architecture système de Windows Server 2025
- Ayi NEDJIMI Consultants — Guide complet du pentest Active Directory
- Ayi NEDJIMI Consultants — Escalade de privilèges Windows 2025
- Ayi NEDJIMI Consultants — Techniques de persistance Windows Server 2025

Synthèse Secured-Core PC : L'architecture Secured-Core empile 7 couches de protection matérielle et hyperviseur pour créer un modèle de sécurité où la compromission du noyau ne suffit plus. **TPM 2.0** fournit la racine de confiance cryptographique et le scellement BitLocker. **Secure Boot + DRTM** garantissent l'intégrité du démarrage même contre les bootkits comme BlackLotus. **VBS** crée un monde isolé (VTL1) via l'hyperviseur Hyper-V et les tables SLAT. **HVCI** impose W^X sur le code noyau, bloquant rootkits et shellcode. **Credential Guard** neutralise Mimikatz en isolant les credentials dans VTL1. **IOMMU** bloque les attaques DMA physiques. **System Guard** fournit l'attestation pour le Zero Trust. Déployez avec verrouillage UEFI, exigez la certification Secured-Core dans vos achats, et intégrez l'attestation dans vos politiques d'accès conditionnel. Les techniques post-exploitation classiques sont neutralisées — les red teams pivotent vers les attaques AD, DPAPI, exploitation applicative et BYOVD.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr