AYI NEDJIMI Consultants

Architecture Windows Server 2025 : Noyau NT Expert

Catégorie : Articles Techniques Lecture : 21 min Publié le : 26/03/2026 Auteur : Ayi NEDJIMI

Architecture interne Windows Server 2025 et Windows 11 : ntoskrnl, HAL, Win32k, LSASS, SAM, VBS/HVCI, Credential Guard. Guide expert securite.

En bref : L'**architecture windows server 2025** et Windows 11 repose sur un noyau NT mature mais profondément remanié — ntoskrnl, HAL, Win32k, LSASS, SAM, Credential Providers, VBS/HVCI, Secure Kernel. Cet article démonte chaque couche, des rings de protection x86/x64 jusqu'aux mécanismes de Virtualization Based Security, avec une attention particulière aux vecteurs d'attaque et aux contre-mesures modernes que tout ingénieur système et sécurité doit maîtriser. Nous couvrons l'anatomie du noyau NT (ntoskrnl.exe, HAL, Executive), le flux d'authentification Winlogon/GINA/Credential Providers, LSASS et ses authentication packages (MSV1_0, Kerberos, NTLM, WDigest), la base SAM, l'architecture des services (SCM, svchost, Protected Services), et la rupture architecturale majeure de Server 2025 : VBS, HVCI, Credential Guard, et le Secure Kernel dans VTL1. Un article de référence pour les Security Engineers et System Architects qui veulent aller au-delà des schémas marketing.

Quand on parle d'architecture Windows en 2026, la plupart des ressources s'arrêtent à un schéma "User Mode / Kernel Mode" pompeux et vide de sens. L'**architecture windows server 2025** mérite mieux. En dix ans de pentests, de réponses à incident et de certifications — dont l'OSCP — j'ai vu des systèmes Windows pilotés par des équipes qui ne savaient pas ce que faisait ntoskrnl.exe, ni pourquoi LSASS tournait en mode protégé. Ce n'est pas de la théorie académique : comprendre l'architecture interne du noyau NT, c'est comprendre pourquoi Credential Guard empêche Pass-the-Hash, pourquoi VBS isole le Secure Kernel, et pourquoi mimikatz ne fonctionne plus de la même façon sur un Server 2025 durci. Windows Server 2025 marque une rupture architecturale franche par rapport à 2019 : le modèle de sécurité est désormais fondé sur l'hyperviseur, les composants critiques s'exécutent dans un monde isolé, et le noyau traditionnel a perdu le monopole de la confiance. Nous allons disséquer chaque composant — de la HAL au SAM, de Winlogon aux Credential Providers, de svchost à la Protected Service Architecture — avec le niveau de détail qu'exige une vraie pratique terrain. Pas de survol, pas de contenu recyclé depuis une certification. Du solide.

Architecture NT : le modèle en couches et les rings de protection

Le noyau Windows NT s'articule autour d'une *séparation stricte User Mode / Kernel Mode* implémentée au niveau matériel via les rings de protection x86/x64. **Ring 0** est le domaine exclusif du kernel, des drivers, et de la HAL. **Ring 3** est celui des processus utilisateurs. Les rings 1 et 2 ne sont pas utilisés par Windows.

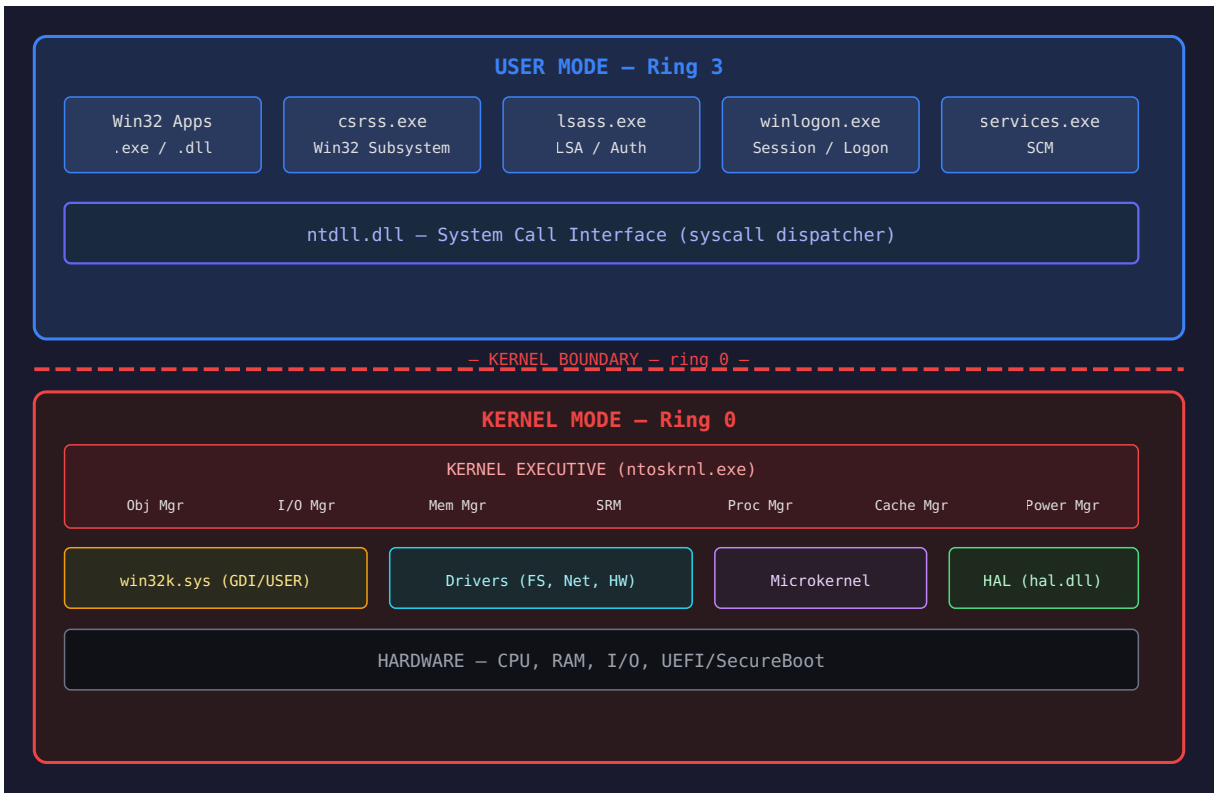


Schéma de l'architecture NT — séparation User Mode / Kernel Mode avec les composants principaux

La **transition ring 3 vers ring 0** s'effectue exclusivement via un mécanisme de *system call* (syscall/sysenter). Depuis Windows 8.1, le dispatch table des syscalls est directement dans ntdll.dll — les numéros de syscall varient entre chaque build Windows, ce qui est une technique d'obfuscation implicite contre les exploits portables. Sur Windows 11 24H2, le numéro de NtOpenProcess est différent de celui de Server 2025, et différent de celui de Windows 10 22H2. Cette variabilité force les rootkits et les EDR à utiliser des techniques de résolution dynamique plutôt que des tables hardcodées.

La différence architecturale majeure entre Windows 11 et Server 2025 n'est pas dans la couche NT classique, mais dans la **couche hyperviseur** : Server 2025 active par défaut VBS (Virtualization Based Security), ce qui ajoute un Ring -1 conceptuel — l'hyperviseur Hyper-V minimal qui supervise le kernel lui-même. Windows 11 supporte VBS mais ne l'active pas nécessairement selon le hardware. Server 2025 l'impose si le hardware est compatible.

ntoskrnl.exe : anatomie du noyau NT

ntoskrnl.exe (NT OS Kernel) est le composant central de Windows. Le fichier réel chargé dépend de la configuration matérielle : ntoskrnl.exe (monoprocesseur, rare aujourd'hui), ntkrnlmp.exe (multiprocesseur), et leurs variantes PAE. Sur les systèmes modernes 64 bits, c'est systématiquement ntkrnlmp.exe mappé sous le nom ntoskrnl.exe par le chargeur. Sa taille dépasse 20 Mo sur Server 2025.

Le noyau NT est souvent décrit comme un *noyau hybride* : il combine un microkernel (scheduling, synchronisation primitives, interruptions) avec un Executive qui s'exécute en Ring 0 pour des raisons de performance. Les appels entre ces couches n'impliquent pas de changement de ring.

Les sous-systèmes de l'Executive

Composant	Rôle principal	Exports critiques
Object Manager	Gestion du namespace d'objets NT (\Device, \BaseNamedObjects, \Sessions)	ObOpenObjectByName, ObReferenceObjectByHandle
I/O Manager	Stack de drivers, IRP (I/O Request Packets), dispatch routines	IoCreateDevice, IoCallDriver, IoBuildDeviceIoControlRequest
Memory Manager	Virtual Address Space, paging, VAD tree, pool allocator	MmAllocateNonCachedMemory, MmMapIoSpace, ExAllocatePoolWithTag
Process & Thread Manager	EPROCESS/ETHREAD structures, création/terminaison	PsLookupProcessByProcessId, PsGetCurrentProcess
Security Reference Monitor (SRM)	Vérification des Access Control Entries, impersonation, tokens	SeAccessCheck, SeCreateAccessState, SePrivilegeCheck
Cache Manager	Cache fichiers unifié, Section objects, VAD intégration	CcInitializeCacheMap, CcScheduleReadAhead
Power Manager	États S0-S5, Connected Standby, Modern Standby (S0ix)	PoRequestPowerIrp, PoSetPowerState

L'export **ZwQuerySystemInformation** mérite une attention particulière. Cette fonction, exposée par ntoskrnl et ntdll, permet de récupérer des informations système via des classes numérotées (SystemProcessInformation, SystemHandleInformation, SystemModuleInformation...). C'est historiquement l'une des API les plus utilisées par les malwares et les outils de sécurité pour énumérer les processus, handles, et modules chargés. Sur Windows Server 2025, certaines classes sont désormais filtrées ou retournent des données incomplètes lorsque le caller n'est pas en mode kernel ou n'est pas un processus de confiance.

Le *microkernel* au sein de ntoskrnl gère le scheduling (scheduler multi-niveaux à 32 priorités), les **DPCs** (Deferred Procedure Calls), les APCs (Asynchronous Procedure Calls), la gestion des interruptions et les primitives de synchronisation bas niveau (spinlocks, KeWaitForSingleObject). C'est cette couche qui interagit directement avec la HAL pour les opérations hardware-dépendantes.

HAL : Hardware Abstraction Layer

La **HAL** (*Hardware Abstraction Layer*, hal.dll) est le composant qui isole le noyau NT des spécificités matérielles. Concrètement, c'est une DLL chargée très tôt dans le processus de boot, avant même ntoskrnl dans certaines configurations. Elle expose une interface uniforme pour : la gestion des interruptions (HAL Interrupt Controller), l'accès aux ports I/O, la gestion du timer système, les opérations DMA, et l'interface ACPI.

Avant l'ère UEFI, plusieurs variantes de hal.dll coexistaient : *halacpi.dll* (ACPI standard), *halmacpi.dll* (multiprocesseur ACPI), *halaacpi.dll* (ACPI avancé). Windows Vista a unifié cette approche en un seul hal.dll configurable. Sous Windows Server 2025 et Windows 11, il n'existe qu'un seul hal.dll, mais son comportement varie selon la présence d'un firmware UEFI et du Secure Boot.

L'impact de Secure Boot sur la HAL est structurel : en mode UEFI + Secure Boot actif, le firmware vérifie la signature du bootloader (bootmgr.efi), qui à son tour vérifie la signature du noyau NT et de hal.dll avant de les charger. Ce mécanisme de *Measured Boot* trace chaque composant dans les PCR (Platform Configuration Registers) du TPM. Toute modification non signée de hal.dll déclenche un refus de démarrage. C'est la raison pour laquelle les bootkits modernes ciblent désormais le firmware UEFI lui-même plutôt que la HAL — un niveau plus bas dans la chaîne de confiance.

Sur un système Server 2025 avec VBS activé, la HAL n'a même plus accès direct au contrôleur d'interruptions APIC pour certaines opérations : c'est l'hyperviseur Hyper-V qui intercepte ces accès via VMCS (Virtual Machine Control Structure) et les virtualise. La HAL du guest (le noyau NT) pense interagir avec le hardware, mais interagit en réalité avec les interfaces virtualisées exposées par le VMCS.

Win32 Subsystem : csrss, win32k.sys, dwm.exe

Le **Win32 Subsystem** est l'environnement d'exécution des applications Win32 traditionnelles. Son architecture s'étale sur deux couches : un composant User Mode (csrss.exe) et un composant Kernel Mode (win32k.sys).

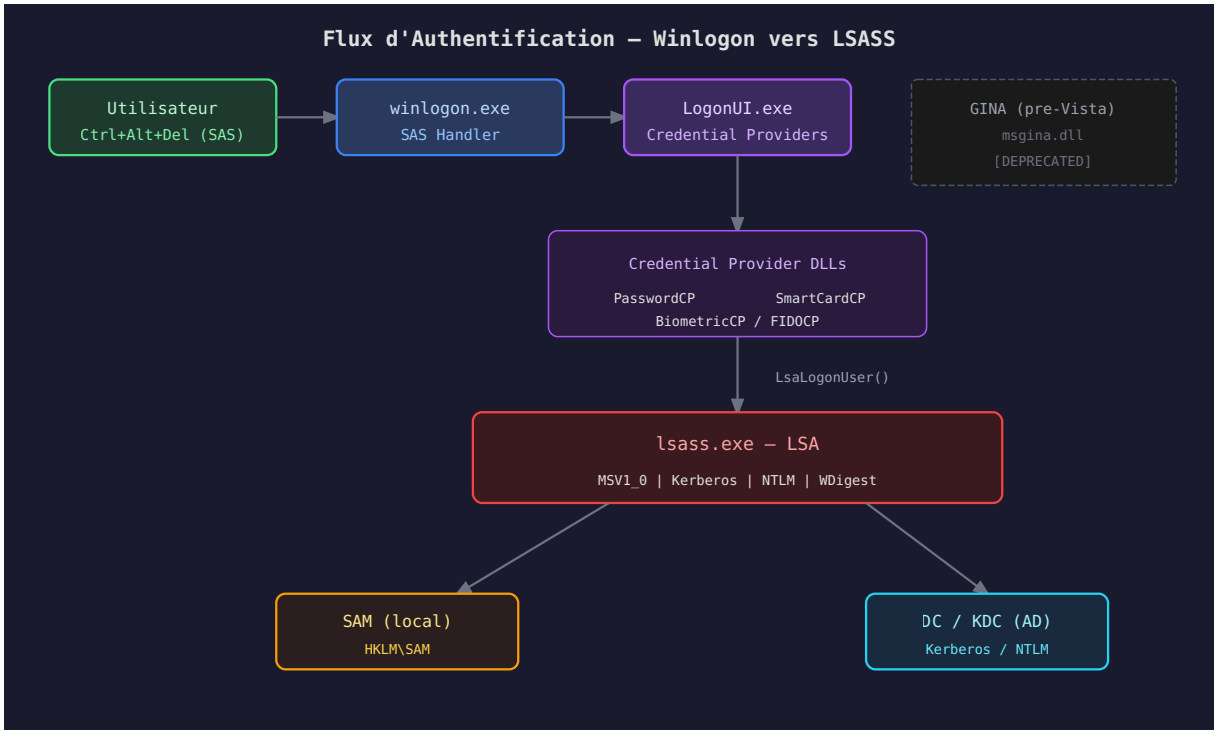
csrss.exe (Client/Server Runtime SubSystem) est l'un des processus les plus critiques du système. Il gère la création et la destruction des processus Win32, la gestion des consoles (conhost.exe lui est fonctionnellement lié), et les shutdown/logoff messages. Tuer csrss.exe provoque un BSOD immédiat — c'est une caractéristique que j'ai vue exploitée dans des attaques DoS locales lors de CTF. Sous Windows 11/Server 2025, csrss.exe tourne comme processus protégé (Protected Process Light, PPL) ce qui limite fortement les possibilités d'injection.

win32k.sys est le driver kernel qui gère l'interface graphique : le sous-système *GDI* (Graphics Device Interface) et le sous-système USER (gestion des fenêtres, messages, hooks). C'est historiquement l'un des composants les plus vulnérables du noyau Windows — une large proportion des élévations de privilèges locaux de la dernière décennie exploitaient des bugs dans win32k.sys. Microsoft a répondu en implémentant le **win32k syscall filtering** depuis Windows 8, permettant aux processus sandboxés (comme les renderers Chrome) de bloquer les syscalls win32k via SetProcessMitigationPolicy.

dwm.exe (Desktop Window Manager) est le compositeur graphique introduit avec Vista. Il s'exécute en User Mode dans la session des utilisateurs connectés, exploite les APIs DirectX pour la composition, et gère toutes les animations, transparences et effets visuels. Sur Windows 11, dwm.exe a été profondément modifié pour le nouveau design Fluent — il intègre des shaders

GPU et des effets de flou (Acrylic) qui consomment plus de ressources GPU que les versions précédentes. Sur Server 2025 en mode Core (sans Desktop Experience), dwm.exe ne s'exécute pas.

Winlogon, GINA et Credential Providers : l'évolution du logon Windows



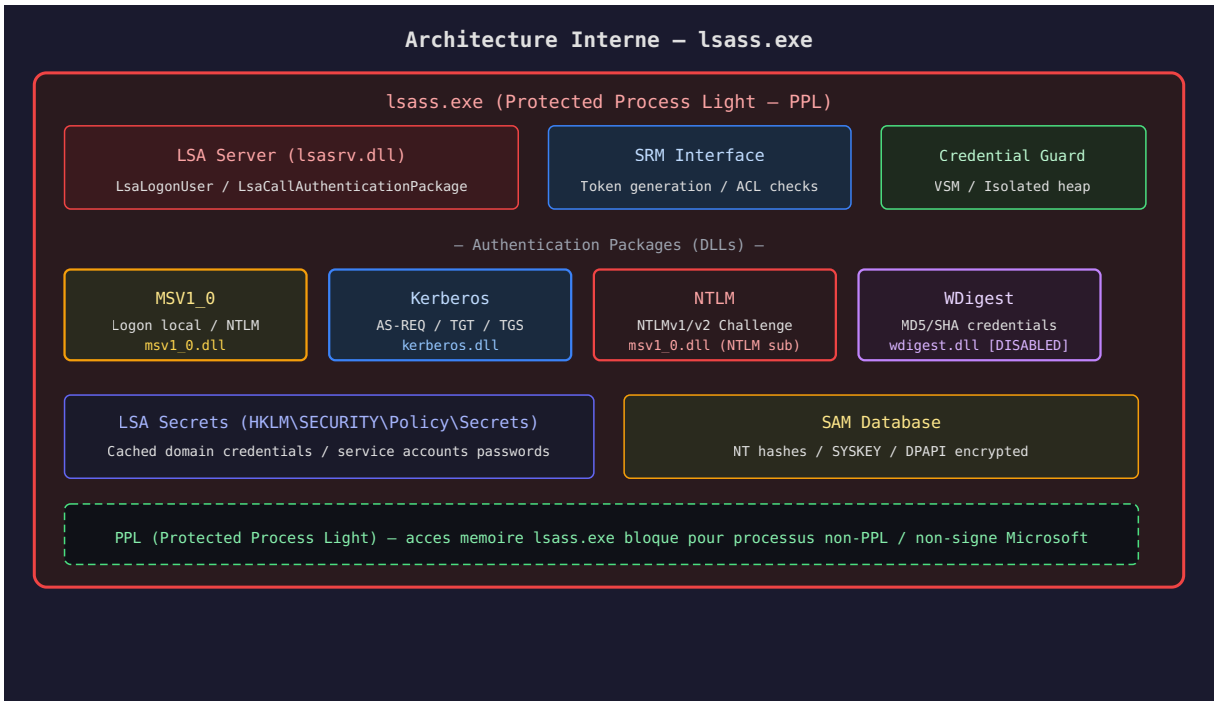
Flux complet d'authentification de Winlogon jusqu'à LSASS et les backends d'authentification

L'évolution du mécanisme de logon Windows est un cas d'école en matière de refactorisation sécurisée. Avant Windows Vista, le composant responsable de l'interface de connexion était **GINA** (*Graphical Identification and Authentication*), une DLL (msgina.dll) chargée par winlogon.exe. Le problème de GINA : son modèle d'extensibilité permettait de remplacer msgina.dll par une implémentation tierce — une fonctionnalité légitime pour les éditeurs de solutions SSO, mais aussi un vecteur d'attaque redoutable. Des malwares comme Trojan.Gina remplaçaient la GINA par une DLL malveillante qui interceptait les credentials avant de les transmettre à la GINA légitime. Le mécanisme de chaînage GINA aggravait le problème : si plusieurs GINA voulaient coexister, elles devaient se "chaîner", et une seule GINA malveillante dans la chaîne compromettait l'intégralité du processus d'authentification.

Depuis Vista, Microsoft a remplacé GINA par les **Credential Providers**. L'architecture est fondamentalement différente : **winlogon.exe** détecte la *SAS* (Secure Attention Sequence, Ctrl+Alt+Del) et lance **LogonUI.exe**, qui charge les Credential Provider DLLs enregistrées sous `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\Credential Providers`. Chaque provider est isolé — un provider défaillant ou compromis n'empêche pas les autres de fonctionner. Les providers standards incluent le Password CP, le Smart Card CP, le Biometric CP (Windows Hello), et depuis Windows 11 le FIDO2/Passkey CP.

La **SAS** mérite qu'on s'y attarde. Ctrl+Alt+Del est intercepté au niveau du driver de clavier (kbdclass.sys) qui envoie une notification directement à winlogon via un canal sécurisé — les applications User Mode normales ne peuvent pas simuler cette séquence via SendKeys ou des API clavier standard. C'est la garantie que l'écran de logon affiché est bien celui du système d'exploitation légitime et non un fake logon screen d'un malware. Sur Windows 11, la SAS peut être configurée pour être optionnelle (déconseillé en enterprise) ou remplacée par Windows Hello (biométrie, PIN) qui présente ses propres garanties cryptographiques.

LSASS.exe : le coeur de l'authentification Windows



Architecture interne de lsass.exe : authentication packages, LSA secrets, et protections PPL/Credential Guard

LSASS (*Local Security Authority Subsystem Service*) est le processus User Mode qui implémente la politique de sécurité locale du système. C'est lui qui valide les credentials lors des logons, maintient en mémoire les informations de session des utilisateurs connectés, gère les tokens d'accès, et exporte les APIs LSA utilisées par les providers d'authentification.

Le processus LSASS charge plusieurs **Authentication Packages** sous forme de DLLs dans son espace mémoire :

- **MSV1_0** (msv1_0.dll) : gère les logons locaux. Lors d'une authentification locale, MSV1_0 interroge la base SAM, calcule le hash NT des credentials fournis, et compare. C'est aussi ce package qui gère le cache de credentials de domaine (les 10 derniers logons par défaut, stockés chiffrés dans HKLM\SECURITY\Cache).
- **Kerberos** (kerberos.dll) : protocole d'authentification réseau par défaut dans un environnement Active Directory. Gère l'acquisition des TGT (Ticket Granting Tickets) auprès du KDC, le renouvellement, et le service tickets. Les tickets Kerberos sont mis en cache dans la mémoire LSASS — c'est ce cache qu'exploite Pass-the-Ticket (PtT).

- **NTLM** : protocole challenge-response historique, implémenté via MSV1_0 en sous-composant. Utilisé en fallback lorsque Kerberos n'est pas disponible (absence de connectivité DC, accès par IP, etc.). NTLMv1 est cryptographiquement cassé. NTLMv2 reste utilisé massivement malgré ses faiblesses face aux attaques NTLM relay.
- **WDigest** (wdigest.dll) : historiquement, stockait les credentials en clair dans la mémoire LSASS pour supporter l'authentification HTTP Digest. *Désactivé par défaut depuis Windows 8.1/2012 R2*. Sa réactivation (valeur 1 dans la clé UseLogonCredential) expose immédiatement les credentials en clair en mémoire LSASS.

Protections LSASS : PPL, Credential Guard et sécurisation Server 2025

La mémoire LSASS est la cible de prédilection des attaquants post-exploitation. **Mimikatz**, **sekurlsa**, et les nombreux clones permettent d'extraire depuis cette mémoire : les hashes NT, les tickets Kerberos en cache, et parfois les credentials en clair (si WDigest est activé). Les protections mises en place par Microsoft pour contrer cela :

- **PPL** (Protected Process Light) : depuis Windows 8.1, LSASS peut être configuré en mode PPL via la clé de registre RunAsPPL dans HKLM\SYSTEM\CurrentControlSet\Control\Lsa. Un processus PPL ne peut être accédé en mémoire que par un processus de niveau de protection égal ou supérieur, signé Microsoft. OpenProcess sur LSASS en PPL retourne ACCESS_DENIED même pour un compte SYSTEM non-PPL.
- **Credential Guard** : va plus loin que PPL — les credentials (hashes NT, TGT) sont stockés dans un processus isolé dans le monde VTL1 (Virtual Trust Level 1) de VBS. Même avec des droits kernel dans VTL0, accéder à ces secrets est impossible sans compromettre le Secure Kernel.

SAM et Security Hive : stockage des identités locales

La base **SAM** (*Security Account Manager*) est l'annuaire local des comptes Windows. Elle est stockée dans le fichier `%SystemRoot%\System32\config\SAM`, correspond à la ruche de registre HKLM\SAM, et est chargée par LSASS au démarrage. Son accès en ligne est restreint : même en tant qu'Administrateur local, un export direct du registre est bloqué par une ACL spécifique — seul LSASS (SYSTEM) peut ouvrir ce fichier tant que le système est en marche.

Deux vecteurs d'extraction classiques contournent cette protection :

1. **Volume Shadow Copy** : la commande vssadmin crée un snapshot du volume incluant le fichier SAM, SYSTEM, et SECURITY. Ces fichiers peuvent ensuite être copiés depuis le shadow. C'est la technique utilisée par impacket's secretsdump.py et par les attaques VSS-based credential theft.
2. **Registry hive export offline** : si l'attaquant a accès physique ou peut monter le disque (WinPE, bootCD), les fichiers SAM, SYSTEM et SECURITY peuvent être copiés directement et analysés offline avec samdump2, hashcat, ou impacket.

Le chiffrement du SAM a évolué plusieurs fois. Jusqu'à Windows 2000, les hashes NT dans le SAM étaient chiffrés avec un mécanisme faible (LM hash + RC4 avec une clé dérivée du RID). **SYSKEY** (Windows NT 4.0 SP3+) a introduit un chiffrement supplémentaire de la base SAM avec une clé de 128 bits pouvant être stockée sur disque (mode 1, par défaut), en mémoire avec mot de passe (mode 2), ou sur disquette/USB (mode 3). Depuis Windows 10, SYSKEY est retiré en tant que fonctionnalité admin mais le chiffrement existe toujours — il est désormais géré par **DPAPI** (Data Protection API) qui utilise le TPM comme ancre cryptographique sur les systèmes modernes.

La structure de HKLM\SECURITY\SAM est organisée par domaine (SAM\Domains\Account\Users) avec pour chaque utilisateur une sous-clé numérique (le RID, ex: 000001F4 pour Administrator). Les valeurs critiques sont `V` (contient le hash NT chiffré, le nom, et d'autres attributs) et `F` (flags de compte, dates de connexion). L'extraction et le déchiffrement de ces structures est documenté dans le code d'impacket (impacket/examples/secretsdump.py sur GitHub) et constitue une référence technique solide.

Services Architecture : SCM, svchost et Protected Services

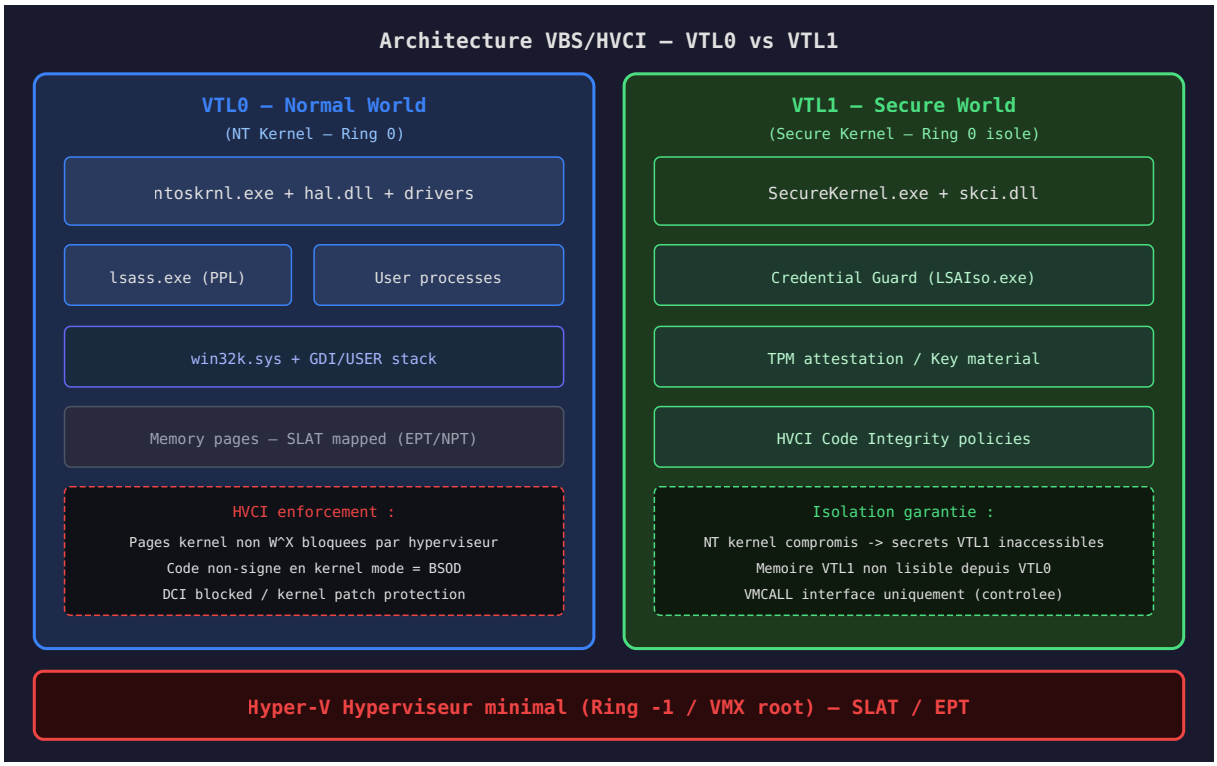
Le **SCM** (*Service Control Manager*, services.exe) est le gestionnaire de services Windows. Il démarre au boot les services marqués Auto, gère les dépendances entre services, et expose une interface RPC (ServicesActive named pipe) pour la gestion des services. Compromettre services.exe équivaut à compromettre le contrôle de tous les services du système.

svchost.exe est le host générique pour les services implémentés sous forme de DLLs. Avant Windows 10, beaucoup de services partageaient le même processus svchost — une économie de mémoire qui avait un coût sécuritaire : un service compromis pouvait potentiellement accéder à la mémoire des autres services du même svchost. Depuis Windows 10 version 1703, sur les systèmes avec plus de 3.5 Go de RAM, chaque service a son propre processus svchost — ce que Microsoft appelle le **per-service SID** isolation. Sur Server 2025, c'est le comportement par défaut et il est renforcé par la Protected Service Architecture.

Les **Protected Services** (ou Windows Protected Processes pour les services) fonctionnent sur le même principe que PPL pour LSASS : un service marqué comme Protected ne peut être arrêté, suspendu, ou injecté par un processus non-protégé, même SYSTEM. Les services de sécurité Microsoft Defender, Windows Defender Credential Guard, et plusieurs services de boot critique utilisent ce mécanisme. La liste des services Protected sur Server 2025 inclut notamment : WdFilter (Defender), lsass (si RunAsPPL activé), SgrmBroker (System Guard Runtime Monitor Broker).

Pour approfondir la surveillance de ces services et la détection des manipulations, voir la documentation Microsoft sur la System Guard root of trust. La configuration optimale des services pour un durcissement GPO Active Directory est également critique.

VBS, HVCI et Secure Kernel : l'architecture de sécurité Windows Server 2025



Architecture VBS/HVCI : separation VTL0 (monde normal) et VTL1 (monde securise) avec hyperviseur Hyper-V

VBS (*Virtualization Based Security*) est la transformation architecturale la plus significative de Windows depuis l'introduction du mode 64 bits. Le principe : utiliser le hyperviseur Hyper-V pour créer deux environnements d'exécution parallèles, appelés **VTL0** (Virtual Trust Level 0 — le "monde normal" où tourne le NT kernel classique) et **VTL1** (Virtual Trust Level 1 — le "monde sécurisé" où tourne le Secure Kernel).

L'isolation entre VTL0 et VTL1 est matériellement garantie par la **SLAT** (*Second Level Address Translation*, Intel EPT ou AMD NPT) : l'hyperviseur contrôle la table de traduction des adresses physiques, et les pages mémoire appartenant à VTL1 ne sont tout simplement pas mappées dans l'espace adressable de VTL0. Même un rootkit Ring 0 dans VTL0 ne peut pas lire la mémoire de VTL1. La seule interface de communication est **VMCALL**, un appel hyperviseur contrôlé dont l'interface est fortement restreinte.

HVCI (*Hypervisor-Protected Code Integrity*) exploite VBS pour renforcer l'intégrité du code kernel. Sans HVCI, un driver ou un rootkit Ring 0 peut allouer une page mémoire avec des permissions Write+Execute (W+X) et y placer du code arbitraire. Avec HVCI, l'hyperviseur surveille toutes les modifications des tables de pages kernel : il est impossible de créer une page kernel W+X. Tout code exécuté en Ring 0 doit provenir d'une page marquée executable mais non-writable, et cette page doit avoir été validée par le processus de signature HVCI avant d'être marquée comme telle.

Le **Credential Guard** utilise VTL1 pour isoler le stockage des credentials LSASS. En pratique : un processus isolé **LSAIso.exe** tourne dans VTL1 et détient les material keys (clés de déchiffrement des tickets Kerberos et des hashes NT). LSASS dans VTL0 n'a accès qu'à des "handles" opaques vers ces secrets — il ne peut pas les lire directement. Même en dumpant la mémoire LSASS, on obtient des handles inutilisables. C'est la fin de la plupart des attaques basées sur l'extraction de credentials depuis la mémoire LSASS.

skci.dll (Secure Kernel Code Integrity) est la DLL qui tourne dans VTL1 et valide les signatures des drivers avant qu'HVCI les marque comme exécutables. **SecureKernel.exe** est le binaire du Secure Kernel lui-même — environ 2 Mo signé Microsoft, responsable de l'ensemble du monde VTL1. La documentation officielle de référence se trouve sur Microsoft Learn — Virtualization Based Security.

Device Guard, System Guard et attestation

Device Guard est le terme historique pour l'ensemble des politiques HVCI + WDAC (Windows Defender Application Control). WDAC permet de définir des politiques CodeIntegrity qui spécifient quels binaires (signés par quelles autorités, ou identifiés par hash) peuvent s'exécuter. Sur Server 2025, WDAC est recommandé en remplacement de AppLocker qui reste supporté mais ne bénéficie plus des améliorations.

System Guard est une technologie complémentaire qui utilise le TPM et les mécanismes de démarrage sécurisé pour attester de l'état du système depuis le boot. *System Guard Secure Launch* (aussi appelé Dynamic Root of Trust for Measurement, DRTM) utilise les instructions CPU Intel TXT ou AMD SKINIT pour établir une chaîne de confiance mesurée indépendante du UEFI — particulièrement pertinent face aux attaques UEFI sophistiquées. Cette technologie est requise pour le niveau de conformité Secured-core Server.

Pour les ingénieurs sécurité qui travaillent sur la détection de rootkits et l'analyse mémoire kernel, voir rootkits kernel mode et rétro-ingénierie — un complément direct à cet article sur l'architecture. Les techniques d'évasion d'EDR en 2026 sont directement conditionnées par ces mécanismes HVCI.

Nouveautés Windows Server 2025 et Windows 11 24H2

Windows Server 2025 (build 26100) introduit plusieurs nouveautés architecturales substantielles :

- **Hotpatch** : mécanisme de patching en mémoire sans reboot, disponible sur Server 2025 Azure Arc. Le principe est similaire au live-patching Linux (kpatch, livepatch) mais implémenté via des mécanismes VBS : le patch modifie le code dans un contexte VTL1 protégé, ce qui garantit l'intégrité du patch et évite les manipulations post-application. Le cycle standard Server 2025 prévoit 3 Hotpatches par an + 1 baseline update (avec reboot).
- **SMB over QUIC** : support natif du protocole QUIC (UDP-based, TLS 1.3) pour SMB, permettant des connexions sécurisées sans VPN. L'implication sécuritaire est significative :

pas d'exposition du port 445 sur Internet, authentification mutuelle via certificats, résistant aux attaques NTLM relay classiques sur SMB.

- **Delegation Tracing improvements** : Server 2025 améliore la traçabilité des délégations Kerberos — un point critique pour détecter les abus de délégation contrainte/non-contrainte documentés par BadSuccessor et les attaques RBCD (voir RBCD : attaque et défense Active Directory).
- **AD LDS améliorations** : nouvelles API Graph, support passkeys (FIDO2) en natif pour l'authentification AD.

Windows 11 24H2 apporte côté architecture kernel :

- **Kernel Data Protection (KDP)** : extension de HVCI pour protéger les structures de données kernel critiques (EPROCESS, ETHREAD flags) contre les modifications depuis VTL0. Des rootkits comme DKOM (Direct Kernel Object Manipulation) qui modifiaient l'ActiveProcessLinks d'EPROCESS pour se cacher sont neutralisés.
- **Smart App Control** : politique WDAC automatisée basée sur des modèles d'apprentissage machine Microsoft pour autoriser ou bloquer les applications inconnues — intégré au niveau kernel.
- **Recall** (fonctionnalité Copilot+ PC) : capture périodique d'écran analysée par IA locale sur NPU. La fonctionnalité a suscité de vives controverses sécuritaires — je recommande sa désactivation systématique en contexte enterprise, les snapshots étant stockées dans une base SQLite locale avec des protections DPAPI insuffisantes face à un attaquant ayant compromis la session.

Points clés à retenir

- L'architecture NT repose sur un modèle hybride microkernel/Executive — le Ring 0 contient ntoskrnl, HAL, win32k.sys, et les drivers
- VBS/HVCI sur Server 2025 crée un second niveau d'isolation (VTL1) matériellement garanti par l'hyperviseur Hyper-V et la SLAT
- LSASS en mode PPL + Credential Guard rend l'extraction de credentials depuis la mémoire LSASS pratiquement impossible sur un système durci
- WDigest doit rester désactivé (UseLogonCredential=0) — sa réactivation expose les credentials en clair en mémoire
- Le SAM ne peut être lu en live que par LSASS/SYSTEM — les extractions passent par VSS ou accès offline
- HVCI empêche les pages kernel W+X — les rootkits traditionnels ne fonctionnent plus sur les systèmes HVCI activé
- Hotpatch sur Server 2025 permet les patches de sécurité sans reboot — à activer en priorité sur Azure Arc
- win32k.sys syscall filtering est un contrôle de réduction de surface d'attaque majeur pour les processus sandboxés

Monitoring et audit de l'architecture NT : événements critiques

Comprendre l'architecture NT sans savoir quoi monitorer est une demi-victoire. Le *Security Reference Monitor* (SRM) dans ntoskrnl génère les événements d'audit Windows fondamentaux pour la détection des compromissions. Les Event IDs critiques à surveiller dans tout SIEM enterprise sont : 4624/4625 (logon réussi/échoué), 4672 (assignation de privilèges sensibles à un nouveau logon — typiquement les comptes avec SeDebugPrivilege), 4768/4769 (requêtes Kerberos AS-REQ et TGS-REQ — détection des Kerberoasting et AS-REP Roasting attacks), et 4648 (logon avec credentials explicites — indicateur de Pass-the-Hash ou d'utilisation de runas).

Pour la surveillance spécifique de LSASS, deux approches coexistent. L'audit natif Windows (Event ID 4656 avec Object Access auditing activé sur le processus lsass.exe) est bruyant mais exhaustif. **Sysmon** avec l'Event ID 10 (ProcessAccess) filtre les accès à LSASS avec les masques d'accès critiques (0x1010, 0x1410 — READ_PROCESS_MEMORY). Sur Server 2025, le *Kernel Patch Protection* (PatchGuard) génère des événements kernel-level supplémentaires lorsque des tentatives de modification des structures kernel sont détectées.

L'audit des Credential Providers (surveillance de la clé de registre d'enregistrement) doit être intégré à toute politique de monitoring enterprise. Une modification de cette clé non attendue est un indicateur fort de persistence ou d'implantation d'un credential stealer.

Implications offensives et défensives : ce que change Server 2025

Pour un pentest sur un environnement Server 2025 correctement durci, plusieurs vecteurs classiques sont neutralisés. Pass-the-Hash fonctionne encore si Credential Guard n'est pas activé — mais sur un Server 2025 by-design, il l'est. Pass-the-Ticket devient plus complexe : les tickets Kerberos résidant dans LSAIso.exe (VTL1), un dump LSASS ne les expose pas. Les techniques d'injection de driver non signé sont bloquées par HVCI. Les attaques DKOM classiques (manipulation d'EPROCESS) sont contrecarrées par KDP.

Ce qui fonctionne encore : les attaques au niveau applicatif (vol de tokens via handle duplication si l'attaquant est déjà SYSTEM), les attaques Kerberos Delegation abuse (RBCD, Constrained Delegation, voir exploitation Kerberos en Active Directory), les techniques d'escalade via misconfiguration (services paths non quotés, DLL hijacking dans les répertoires writables, escalade de privilèges Windows user vers SYSTEM), et les attaques via le plan de management (WMI, PowerShell Remoting, RDP) qui opèrent au-dessus de la couche kernel.

Sur le plan défensif, la priorité en 2026 est l'activation systématique de la stack VBS/HVCI/Credential Guard, dont les prérequis hardware (TPM 2.0, Secure Boot, IOMMU/VT-d, 64 bits) sont désormais satisfaits sur tout hardware de moins de 5 ans. Le guide de sécurisation Active Directory 2025 documente les GPOs correspondantes. Pour le forensics, les outils doivent être adaptés : Volatility 3 et le forensics mémoire en 2026 couvre les adaptations nécessaires pour analyser la mémoire LSASS sur des systèmes Credential Guard. La NIST SP 800-123 Guide to General Server Security reste une référence de base pour la configuration des serveurs.

Côté monitoring, le *Security Reference Monitor* (SRM) dans ntoskrnl génère les événements d'audit Windows (Event IDs 4624, 4625, 4672, 4768, 4769...). Une architecture correcte implique de centraliser ces événements dans un SIEM avec une rétention adaptée. L'activation de l'audit Process Creation (Event ID 4688 avec command line logging) et des Kernel Object accesses permet de tracer les tentatives d'accès à LSASS (Event ID 10 Sysmon, ou 4656 Windows native audit). Pour approfondir la détection des mouvements latéraux qui exploitent ces mécanismes, voir mouvement latéral : détection et prévention.

Conclusion et Perspectives

L'architecture Windows Server 2025 représente une rupture franche avec l'ère pré-VBS. Le modèle de menace a évolué : les attaques kernel Ring 0 classiques — injection de driver non signé, DKOM, patching LSASS — se heurtent à des barrières matériellement garanties par l'hyperviseur. Cela ne signifie pas que Windows est invulnérable. Les vecteurs se déplacent vers les couches supérieures : abus de délégation Kerberos, misconfiguration des politiques WDAC, attaques sur le firmware UEFI (au-dessous de la chaîne de confiance VBS), et exploitation applicative sophistiquée.

Ce que je retiens de l'évolution architecturale : Microsoft a fait le choix stratégique de sacrifier la compatibilité (certains vieux drivers non signés ne tournent plus) pour des gains sécuritaires réels. C'est une décision correcte. La prochaine évolution probable touche la confidential computing (AMD SEV-SNP, Intel TDX) qui étend VBS à l'isolation entre VMs sur un même host — pertinent pour les clouds multi-tenants. On va dans la bonne direction.

Sources et références : MITRE ATT&CK · CERT-FR

Questions Fréquentes

Quelle est la différence entre ntoskrnl.exe et ntkrnlmp.exe ?

Ces deux fichiers sont des variantes du même noyau NT. **ntoskrnl.exe** est la version monoprocesseur (uniprocesseur), historiquement utilisée sur les systèmes à un seul CPU physique. **ntkrnlmp.exe** est la version multiprocesseur, supportant les systèmes SMP (Symmetric MultiProcessing). Sur tous les systèmes Windows modernes 64 bits — qu'il s'agisse de Windows 11 ou Server 2025 — c'est ntkrnlmp.exe qui est effectivement chargé, mais il est mappé sous le nom ntoskrnl.exe par le chargeur de démarrage (bootmgr/winload). En pratique, en 2026, vous ne verrez jamais la version monoprocesseur sur un système en production. La distinction est devenue anecdotique mais reste utile lors d'analyses forensiques de systèmes anciens.

Credential Guard protège-t-il contre toutes les formes de vol de credentials ?

Non — Credential Guard isole les hashes NT et les tickets Kerberos dans VTL1, empêchant leur extraction depuis la mémoire LSASS dans VTL0. Cependant, plusieurs vecteurs restent exploitables : le vol de tokens via handle duplication (si l'attaquant est déjà SYSTEM et que la cible a un token valide en mémoire), les attaques Kerberos delegation abuse (RBCD, S4U2Proxy)

qui exploitent des misconfiguration Active Directory plutôt que la mémoire LSASS, l'interception des credentials au niveau réseau via NTLM relay (si NTLM n'est pas désactivé), et les attaques via les Credential Provider DLLs si une DLL malveillante est installée. Credential Guard est un contrôle fort mais pas une solution complète — il doit s'inscrire dans une stratégie de défense en profondeur incluant la désactivation de NTLM, le tiering Active Directory, et une politique WDAC stricte.

Comment vérifier que VBS et HVCI sont bien activés sur un Windows Server 2025 ?

Plusieurs méthodes permettent de vérifier l'état de VBS et HVCI. En PowerShell : la commande `Get-ComputerInfo -Property "DeviceGuard*"` retourne l'état de toutes les fonctionnalités Device Guard incluant VBS et HVCI. L'outil msinfo32.exe (Informations système) affiche sous "Résumé système" les lignes "Virtualisation Based Security" et "Hypervisor Code Integrity" avec leur état (Activé/Désactivé/Non supporté). Via le registre, la valeur Enabled sous HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegrity doit être à 1. L'Event Log System contient des événements spécifiques au démarrage de Credential Guard (Source: Microsoft-Windows-DeviceGuard). Sur un Server 2025 fraîchement installé sur hardware compatible, VBS est activé par défaut — vérifiez que HVCI et Credential Guard le sont aussi avant toute mise en production.

Pourquoi GINA a-t-elle été abandonnée et quelles sont les implications pour la sécurité ?

GINA (Graphical Identification and Authentication, msgina.dll) a été abandonnée avec Windows Vista pour plusieurs raisons structurelles. Le modèle de remplacement GINA permettait à des DLLs tierces de s'insérer dans le flux d'authentification — un vecteur d'attaque que des malwares ont largement exploité pour intercepter les credentials en clair lors du logon. Le modèle de chaînage GINA manquait de robustesse : une GINA défaillante bloquait l'accès à la machine. Les Credential Providers qui remplacent GINA fonctionnent en parallèle et en isolation, avec un mécanisme d'enregistrement contrôlé par le registre et une validation de signature. Côté sécurité, les Credential Providers sont mieux isolés mais restent un vecteur de persistance documenté dans plusieurs frameworks offensifs si une DLL malveillante est installée sous la clé d'enregistrement des providers — il convient de surveiller les modifications de cette clé de registre dans tout SIEM enterprise.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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