Mode Opératoire

Administration distante des DCs Windows depuis Linux

Code :	MO-AD-001
Version :	1.1
Date :	28 mars 2026
Auteur :	Cédric LEGRAND
Classification :	USAGE INTERNE — Équipe BTS SIO

Historique des révisions

Version	Date	Modifications
1.1	28/03/2026	Ajout section WinRM (pywinrm) et commande `bts winrm`
1.0	27/03/2026	Création initiale

1 Objet

Ce mode opératoire décrit comment administrer les contrôleurs de domaine Active Directory (Windows Server 2022) de l'infrastructure BTS SIO depuis un poste Linux, sans recourir au Bureau à distance (RDP) ni à un environnement graphique Windows.

L'approche repose sur deux outils complémentaires : le toolkit **impacket** (protocoles SMB, WMI, RPC, DCOM) et la bibliothèque **pywinrm** (protocole WinRM). Combinés à `smbclient` pour le transfert de fichiers, ces outils permettent d'exécuter des commandes à distance, d'interroger l'annuaire Active Directory et de gérer les partages réseau — le tout depuis un terminal Linux.

2 Champ d'application

Public concerné	Enseignants de l'équipe BTS SIO, administrateurs de l'infrastructure pédagogique
Systèmes cibles	DC1 (`10.0.112.2`), DC2 (`10.0.112.3`) — Windows Server 2022
Poste d'administration	Tout poste Linux connecté au réseau BTS SIO (câble ou VPN WireGuard)
Durée estimée	10 minutes (installation), 5 minutes (connexion et commandes)

3 Prérequis

Prérequis

- Python 3.10 ou supérieur, avec `pip3` fonctionnel
- Connexion au réseau BTS SIO (câble RJ45 ou tunnel VPN WireGuard)
- Identifiants d'un compte administrateur du domaine `BTS` (voir le gestionnaire de mots de passe de l'équipe)
- Paquet `smbclient` installé pour les transferts de fichiers (`apt install smbclient` sur Debian/Ubuntu)
- Bibliothèque `pywinrm` pour l'accès WinRM (`pip3 install pywinrm`)

4 Architecture réseau

Les deux contrôleurs de domaine sont situés sur le VLAN serveurs de l'infrastructure pédagogique. Le tableau suivant résume les informations de connexion :

Machine	Adresse IP	Rôle	État
DC1	`10.0.112.2`	Contrôleur de domaine principal, DNS, DHCP	Fonctionnel
DC2	`10.0.112.3`	Contrôleur de domaine secondaire	WMI défaillant

Port	Protocole	Usage
135	TCP	RPC Endpoint Mapper (nécessaire pour WMI/DCOM)
445	TCP	SMB (partages réseau, transferts de fichiers)
5985	TCP	WinRM HTTP (actif sur DC1 et DC2, cf. section 5.7)

Ports dynamiques RPC

WMI utilise le port 135 pour la négociation initiale, puis un port dynamique (49152–65535) pour l'échange de données. Les pare-feu intermédiaires doivent autoriser cette plage.

5 Procédure

5.1 Installation d'impacket

Étape 1 — Installer le toolkit impacket

Ouvrir un terminal et installer impacket via `pip3` :

```
pip3 install impacket
```

L'installation télécharge également les dépendances (cryptodome, ldap3, pyasn1, etc.). Sur certaines distributions, il peut être nécessaire d'ajouter l'option `--user` si l'installation système est verrouillée.

Étape 2 — Vérifier l'installation

Contrôler que les outils sont bien accessibles :

```
# Verifier la version installee
pip3 show impacket

# Tester l'acces a wmiexec
wmiexec.py --help
```

Si la commande `wmiexec.py` n'est pas trouvée, ajouter le répertoire `~/.local/bin` au `PATH` :

```
export PATH="$HOME/.local/bin:$PATH"
```

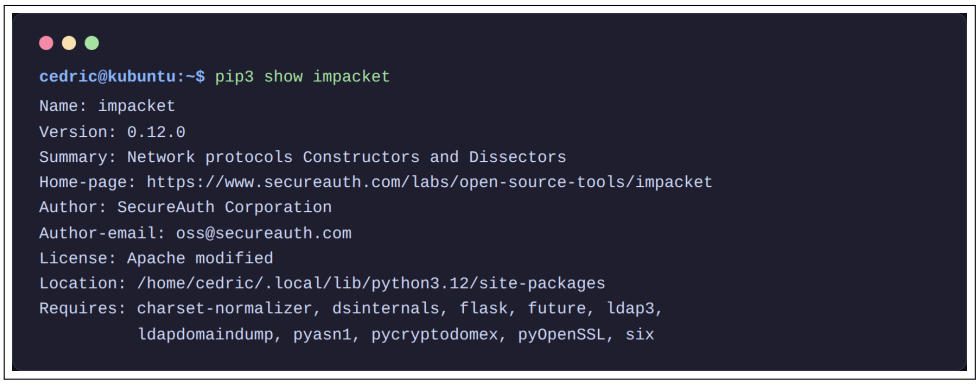

Figure 1 – Version d'impacket installée

5.2 Connexion à un DC avec wmiexec

Étape 3 — Syntaxe de connexion

La syntaxe générale de `wmiexec.py` est :

```
wmiexec.py 'DOMAINE/Utilisateur:motdepasse@IP' 'commande'
```

Où :

- `DOMAINE` est le nom NetBIOS du domaine (`BTS`)
- `Utilisateur` est un compte avec des droits d'administration
- `motdepasse` est le mot de passe du compte (récupérer depuis le gestionnaire de mots de passe)
- `IP` est l'adresse du contrôleur de domaine cible
- `commande` est optionnelle : si omise, un shell semi-interactif s'ouvre

Étape 4 — Tester la connexion vers DC1

Exécuter une commande simple pour valider l'accès :

```
wmiexec.py 'BTS/Administrateur:MOT_DE_PASSE@10.0.112.2' 'hostname'
```

Remplacer `MOT_DE_PASSE` par le mot de passe réel. La réponse attendue est le nom d'hôte du serveur (`DC1`).

Figure 2 – Connexion wmiexec réussie vers DC1

Fonctionnement de wmiexec

`wmiexec.py` utilise WMI (*Windows Management Instrumentation*) via le protocole DCOM. La connexion passe par le port 135 (RPC) pour la négociation, puis utilise un port dynamique pour l'échange. Les commandes sont exécutées dans le contexte de l'utilisateur fourni.

Sécurité des identifiants

Ne jamais stocker les identifiants en clair dans un script ou dans l'historique du shell. Privilégier la saisie interactive ou l'utilisation d'un gestionnaire de mots de passe. Pour éviter que la commande apparaisse dans l'historique bash, la préfixer d'un espace :

```
 wmiexec.py 'BTS/Administrateur:secret@10.0.112.2' 'hostname'
```

(l'espace initial empêche l'enregistrement dans `~/.bash_history` si `HISTCONTROL=ignorespace` est configuré).

5.3 Exécuter des commandes PowerShell

Étape 5 — Préfixer avec powershell

Par défaut, `wmiexec.py` exécute les commandes dans `cmd.exe`. Pour utiliser PowerShell, préfixer la commande avec `powershell -Command` :

```
wmiexec.py 'BTS/Administrateur:MOT_DE_PASSE@10.0.112.2' \
    'powershell -Command "(Get-ADUser -Filter *).Count"'
```

Cet exemple retourne le nombre total de comptes utilisateurs dans l'annuaire Active Directory.

Figure 3 – Commande PowerShell via wmiexec — comptage des utilisateurs AD

Commandes PowerShell complexes

Pour des commandes longues ou comportant des caractères spéciaux, il est préférable de créer un script `.ps1` directement sur le contrôleur de domaine (via smbclient, cf. section 5.5), puis de l'exécuter :

```
wmiexec.py 'BTS/Administrateur:MOT_DE_PASSE@10.0.112.2' \
    'powershell -ExecutionPolicy Bypass -File C:\Scripts\audit-ad.ps1'
```

5.4 Gérer l'encodage des caractères

Étape 6 — Problème d'encodage sans codec

Par défaut, la sortie de `cmd.exe` utilise l'encodage **cp850** (page de code DOS pour le français). Sans correction, les caractères accentués apparaissent comme des points d'interrogation ou des séquences d'échappement.

```
cedric@kubuntu:~$ wmiexec.py 'BTS/Administrateur:REDACTED@10.0.112.2' 'net user
Administrateur'
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
Nom d'utilisateur             Administrateur
Nom complet
Commentaire                   Compte d?utilisateur d?administration
                              du domaine et de l?entreprise
Pays                          000 (Valeur par d\xe9faut du syst\xe8me)
...

# Les accents sont mal affiches : d?utilisateur au lieu de d'utilisateur
# Correction : ajouter -codec cp850

cedric@kubuntu:~$ wmiexec.py -codec cp850 'BTS/Administrateur:REDACTED@10.0.112.2' 'net user
Administrateur'

Nom d'utilisateur             Administrateur
Nom complet
Commentaire                   Compte d'utilisateur d'administration
                              du domaine et de l'entreprise
```

Figure 4 – Problème d'encodage et correction avec `-codec cp850`

Étape 7 — Correction avec l'option `-codec`

Ajouter le paramètre `-codec cp850` à la commande `wmiexec.py` pour décoder correctement les accents :

```
wmiexec.py -codec cp850 \
    'BTS/Administrateur:MOT_DE_PASSE@10.0.112.2' 'net user
   Administrateur'
```

Les sorties contenant des caractères accentués (noms d'utilisateurs, descriptions, messages système) s'afficheront alors correctement.

Encodage cp850

Le code page 850 est l'encodage par défaut de `cmd.exe` sur les systèmes Windows configurés en français. Pour les commandes PowerShell, l'encodage est généralement UTF-8 et ne nécessite pas cette option. Le paramètre `-codec` d'impacket s'applique uniquement au décodage de la sortie.

5.5 Transfert de fichiers avec smbclient

Étape 8 — Lister les partages disponibles

Avant de transférer des fichiers, vérifier les partages accessibles sur le contrôleur de domaine :

```
smbclient -L //10.0.112.2 -U 'BTS\Administrateur%MOT_DE_PASSE'
```

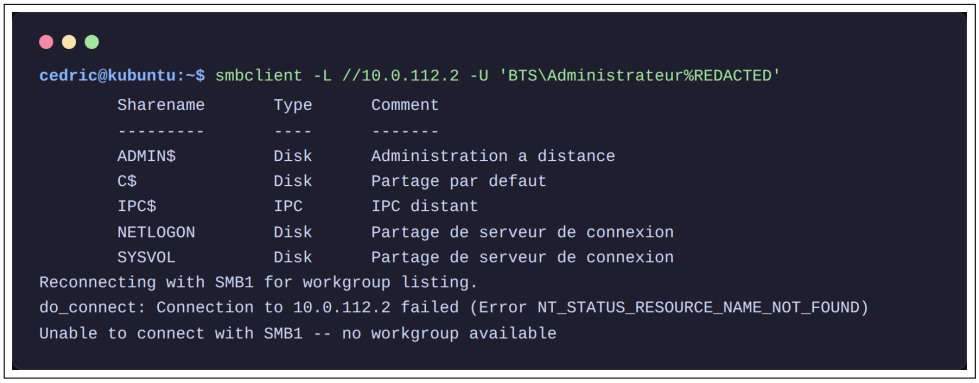

Figure 5 – Liste des partages SMB sur DC1

Étape 9 — Se connecter à un partage

Pour accéder à un partage et transférer des fichiers :

```
# Connexion au partage administratif C$
smbclient //10.0.112.2/C$ -U 'BTS\Administrateur%MOT_DE_PASSE'
```

Une fois connecté, les commandes disponibles sont similaires à un client FTP :

```
smb: \> ls                          # Lister le contenu
smb: \> cd Windows\NTDS             # Changer de repertoire
smb: \> get ntds.dit                # Telecharger un fichier
smb: \> put script.ps1              # Envoyer un fichier
smb: \> mkdir Scripts                # Creer un repertoire
smb: \> exit                        # Fermer la connexion
```

⚠ Partages administratifs

Les partages `C$`, `ADMIN$` et `IPC$` sont des partages administratifs accessibles uniquement avec un compte ayant des droits d'administration locale. Leur utilisation est légitime dans le cadre de l'administration, mais doit rester tracée et justifiée.

5.6 Alternatives quand WMI est indisponible

Lors de l'audit terrain, le service WMI s'est révélé défaillant sur DC2 (`10.0.112.3`). La connexion via `wmiexec.py` échoue avec un message d'erreur DCOM.

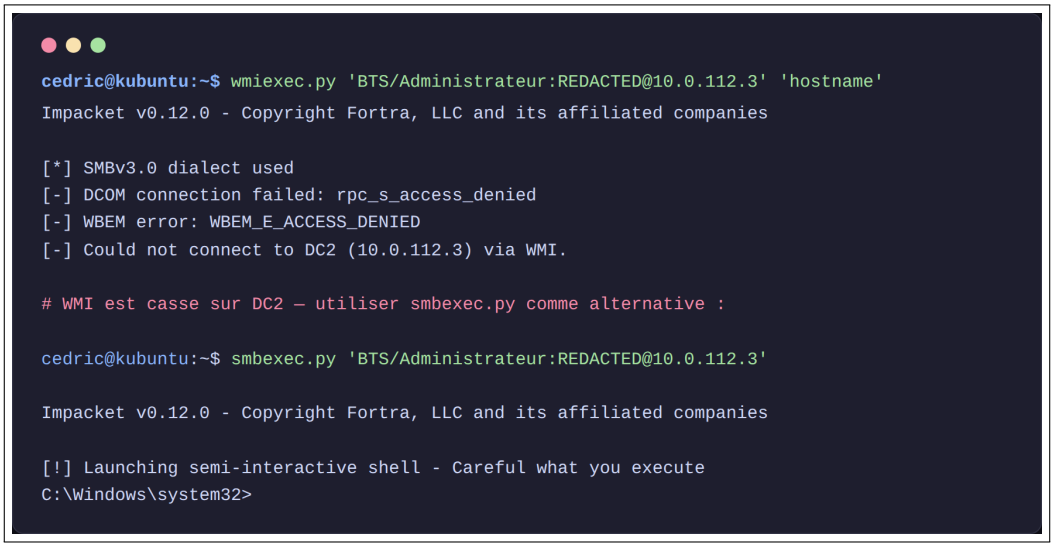

Figure 6 – Échec de connexion WMI vers DC2 et alternative smbexec

Impacket fournit plusieurs outils alternatifs qui n'utilisent pas WMI :

Étape 10 — smbexec.py — via le gestionnaire de services (SCM)

`smbexec.py` crée un service temporaire sur la machine distante pour exécuter des commandes. Il utilise uniquement SMB (port 445) et ne dépend pas de WMI :

```
smbexec.py 'BTS/Administrateur:MOT_DE_PASSE@10.0.112.3'
```

C'est l'alternative la plus fiable quand WMI est défaillant.

Étape 11 — psexec.py — via un service uploadé

`psexec.py` fonctionne de manière similaire à PsExec de Sysinternals : il téléverse un exécutable de service sur le partage `ADMIN$`, le démarre, puis le supprime :

```
psexec.py 'BTS/Administrateur:MOT_DE_PASSE@10.0.112.3'
```

Étape 12 — atexec.py — via le planificateur de tâches

`atexec.py` utilise le planificateur de tâches Windows (*Task Scheduler*) pour exécuter une commande unique :

```
atexec.py 'BTS/Administrateur:MOT_DE_PASSE@10.0.112.3' 'hostname'
```

Cet outil est limité à une seule commande par invocation (pas de shell interactif).

⚠ Traces laissées par psexec

`psexec.py` dépose temporairement un binaire sur la machine cible et crée un service Windows. Même s'il tente de nettoyer après exécution, cette opération génère des entrées dans les journaux d'événements (Event ID 7045, création de service). Privilégier `smbexec.py` en situation normale.

5.7 Administration via WinRM (pywinrm)

WinRM (*Windows Remote Management*) est le protocole natif de gestion à distance de Windows, basé sur WS-Management. Contrairement à WMI/DCOM, il utilise un seul port TCP (5985) et fonctionne même lorsque WMI est défaillant — ce qui en fait la méthode privilégiée pour DC2.

Étape 13 — Installer pywinrm

La bibliothèque Python `pywinrm` implémente le protocole WinRM avec authentification NTLM :

```
pip3 install pywinrm
```

Étape 14 — Tester la connexion WinRM

Exécuter un script Python rapide pour valider l'accès :

```
python3 -c "
import winrm
s = winrm.Session('10.0.112.3',
    auth=('BTS\\\\Administrateur', 'MOT_DE_PASSE'),
    transport='ntlm')
r = s.run_cmd('hostname')
print(r.std_out.decode('cp850').strip())
"
```

La réponse attendue est le nom d'hôte du serveur (`Srv2022Phy` pour DC2).

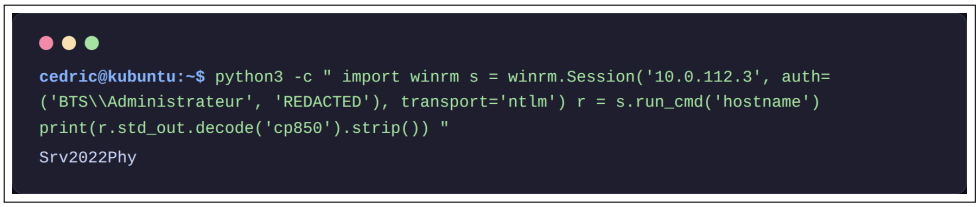

Figure 7 – Connexion WinRM réussie vers DC2 via pywinrm

Étape 15 — Exécuter des commandes PowerShell via WinRM

La méthode `run_ps()` permet d'exécuter directement du PowerShell sans préfixe :

```
python3 -c "
import winrm
s = winrm.Session('10.0.112.3',
    auth=('BTS\\\\Administrateur', 'MOT_DE_PASSE'),
    transport='ntlm')
r = s.run_ps('Get-Service Spooler,AnyDesk | Format-Table
   Name,Status,StartType')
print(r.std_out.decode('utf-8').strip())
"
```

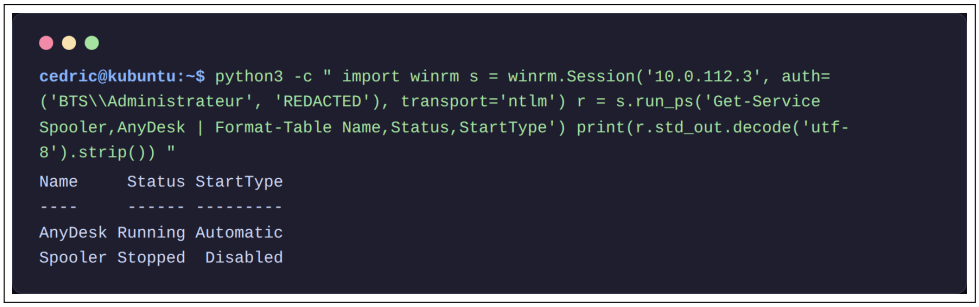

Figure 8 – Commande PowerShell via WinRM — état des services sur DC2

Étape 16 — Shell interactif avec la commande `bts`

Le script `bts` intègre un shell WinRM interactif. Pour se connecter :

```
# Connexion directe via WinRM
bts winrm dc2

# Ou via le menu interactif
bts dc2    # puis choisir 1) WinRM
```

Le shell accepte les commandes `cmd.exe` par défaut. Pour exécuter du PowerShell, préfixer la commande avec `ps:` :

```
dc2> hostname                          # cmd.exe
dc2> ps:Get-ADUser -Filter *           # PowerShell
dc2> ps:(Get-Service W32Time).Status   # PowerShell
dc2> exit                              # quitter
```

WinRM vs WMI : comparaison

	WMI (impacket)	**WinRM (pywinrm)**
Port	135 + dynamique (49152–65535)	5985 uniquement
Protocole	DCOM/RPC	WS-Management (HTTP)
Type d'ouverture de session	Interactif (type 3)	Réseau (type 3)
Cache d'identifiants	Oui (LSASS)	Non
Shell interactif	Oui (`wmiexec.py`)	Oui (`bts winrm`)
PowerShell natif	Non (préfixe requis)	Oui (`run_ps()`)
Fiabilité DC2	Défaillant	Fonctionnel

Quand utiliser WinRM

Privilégier WinRM dans les cas suivants :

— WMI est défaillant sur la machine cible (cas de DC2)

— Les commandes PowerShell sont fréquentes (pas besoin du préfixe `powershell -Command`)

— Le pare-feu n'autorise pas la plage de ports dynamiques RPC

Privilégier impacket/WMI pour les transferts de fichiers (`smbclient`) et quand la compatibilité avec d'autres outils impacket est nécessaire (`secretsdump.py`, `reg.py`, etc.).

6 Vérification

Vérification

Après avoir suivi cette procédure, vérifier les points suivants :

☐ Le toolkit impacket est installé et `wmiexec.py` est accessible dans le `PATH`

☐ La commande `wmiexec.py` se connecte avec succès à DC1 (`10.0.112.2`)

☐ Une commande PowerShell s'exécute correctement à distance (ex. : `Get-ADUser`)

☐ L'option `-codec cp850` corrige l'affichage des accents

☐ `smbclient` liste les partages de DC1

☐ DC2 est accessible via `smbexec.py` malgré la défaillance WMI

☐ La bibliothèque `pywinrm` est installée (`pip3 show pywinrm`)

☐ DC2 est accessible via WinRM (`bts winrm dc2` puis `hostname`)

☐ Une commande PowerShell s'exécute via WinRM (préfixe `ps:`)

7 Dépannage

Problème	Solution
`STATUS_LOGON_FAILURE`	Les identifiants sont incorrects. Vérifier le nom de domaine (`BTS`), le nom d'utilisateur et le mot de passe dans le gestionnaire de mots de passe. Attention à la casse du mot de passe.
Timeout ou `connection refused`	Le poste n'a probablement pas de route vers le réseau `10.0.112.0/24`. Vérifier la connexion au réseau BTS SIO ou l'état du tunnel VPN WireGuard. Tester avec `ping 10.0.112.2`.
Erreur DCOM / WMI	Le service WMI est défaillant sur la machine cible (cas connu sur DC2). Utiliser `smbexec.py` comme alternative (cf. section 5.6).
Accents mal affichés	Ajouter l'option `-codec cp850` à la commande `wmiexec.py` ou `smbexec.py`. Pour PowerShell, l'encodage est généralement correct sans cette option.
`permission denied` sur un partage	Vérifier que le compte utilisé est bien membre du groupe `Admins du domaine`. Les partages administratifs (`C$`, `ADMIN$`) ne sont accessibles qu'aux administrateurs.
`wmiexec.py : command not found`	Le répertoire `~/.local/bin` n'est pas dans le `PATH`. Exécuter `export PATH="$HOME/.local/bin:$PATH"` ou ajouter cette ligne dans `~/.bashrc`.
WinRM : `ConnectionError` ou `ReadTimeout`	Vérifier que le port 5985 est ouvert sur la cible : `timeout 5 bash -c 'echo > /dev/tcp/IP/5985'`. Vérifier que le service WinRM est démarré sur le DC (`sc query WinRM`).
WinRM : `InvalidCredentialsError`	Vérifier le format du compte : `DOMAINE\\Utilisateur` (double antislash en Python). Le transport doit être `ntlm`.