Audit des Groupes Privilégiés Active Directory

Points essentiels

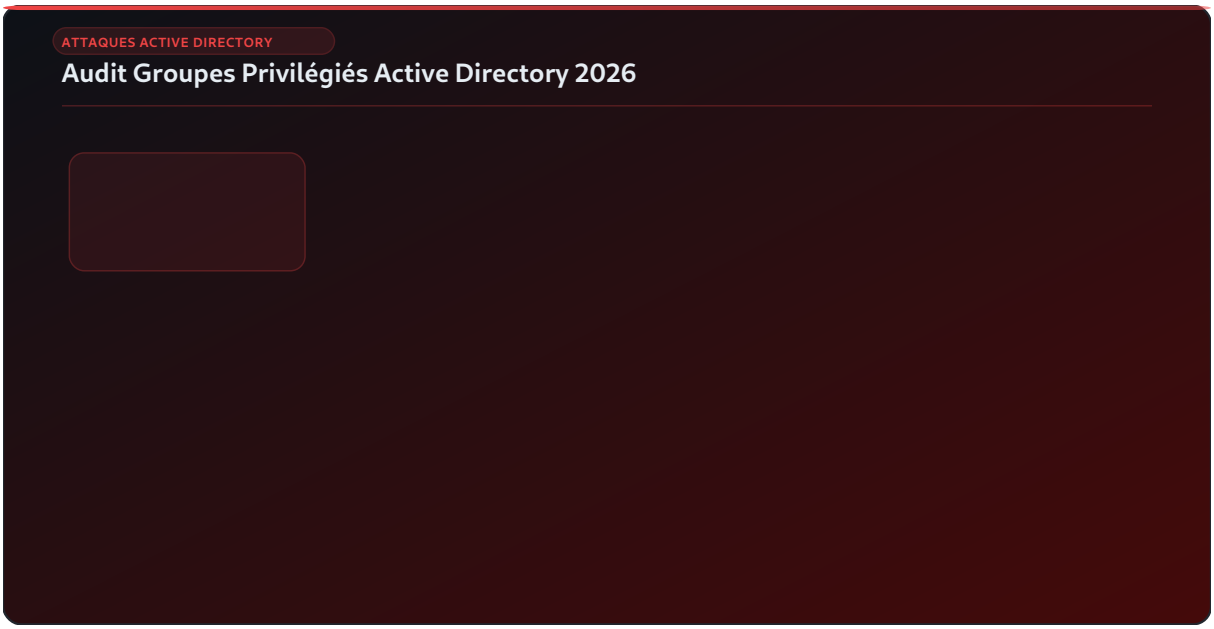
- Le mécanisme AdminSDHolder/SDProp réapplique les ACL des comptes sensibles toutes les 60 minutes
- BloodHound, PingCastle et PurpleKnight auditent les membres directs et récursifs des groupes privilégiés
- Le nesting de groupes reste le vecteur d'escalade de privilèges le plus négligé
- Les Event IDs 4728, 4732 et 4756 doivent déclencher une alerte SIEM temps réel

📋 EN BREF

- Les groupes privilégiés Active Directory (Domain Admins, Enterprise Admins, Schema Admins...) constituent la cible principale des attaquants lors d'une compromission de domaine.
- Le mécanisme AdminSDHolder/SDProp réapplique toutes les 60 minutes les ACL des comptes sensibles : tout objet AdminSDHolder mal configuré offre une porte de persistance discrète.
- PowerShell et les outils BloodHound, PingCastle, PurpleKnight permettent un audit complet des membres directs et récursifs de ces groupes.

> - Le nesting de groupes est le vecteur d'escalade de privilèges le plus souvent négligé lors des audits Active Directory en entreprise.
> - Les Event IDs 4728, 4732, 4756 doivent déclencher une alerte temps réel dans votre SIEM pour toute modification de groupe sensible.

En France, des ETI industrielles aux grands comptes parisiens, Active Directory reste l'épine dorsale de la gestion des identités : il concentre les authentifications, les droits et, trop souvent, des privilèges hérités que personne ne surveille plus. Pour un RSSI soumis aux obligations NIS 2 ou aux exigences ISO 27001, l'audit des groupes privilégiés Active Directory n'est donc pas une option, mais une priorité absolue. Une seule appartenance illégitime au groupe Domain Admins suffit à transformer la compromission d'un poste bureautique en prise de contrôle totale du domaine, avec chiffrement massif à la clé. Comptes de service jamais désactivés, prestataires partis depuis des mois, imbrications de groupes oubliées : ces angles morts constituent aujourd'hui le vecteur privilégié des opérateurs de ransomware. Cet article détaille une méthodologie complète pour cartographier, contrôler et réduire durablement ces privilèges.



Les Groupes Privilégiés AD : Cartographie et Risques

Active Directory embarque nativement une cinquantaine de groupes de sécurité prédéfinis. Parmi eux, une quinzaine sont qualifiés de **groupes sensibles** car leurs membres bénéficient de capacités d'administration étendues sur le domaine ou la forêt. Toute organisation qui ne cartographie pas précisément ces groupes s'expose à une surface d'attaque considérable.

Tableau de référence des groupes sensibles

Groupe	Niveau de risque	Capacités réelles	Membres par défaut
Domain Admins	**Critique**	Contrôle total du domaine, DCSync, accès tous systèmes	Administrator
Enterprise Admins	**Critique**	Contrôle de la forêt entière, ajout de domaines, trusts	Administrator (domaine racine)
Schema Admins	**Critique**	Modification du schéma AD, persistance via attributs custom	Administrator (domaine racine)
Administrators	**Très élevé**	Admin local des contrôleurs de domaine, DCSync possible	Administrator, Domain Admins, Enterprise Admins
Account Operators	**Élevé**	Création/modification de comptes, ajout membres groupes non-protégés	Vide
Backup Operators	**Élevé**	Lecture de tous les fichiers (ntds.dit), connexion locale aux DC	Vide
Print Operators	**Élevé**	Chargement de drivers noyau sur DC (élévation possible)	Vide
Server Operators	**Élevé**	Démarrage/arrêt services DC, connexion locale, gestion partages	Vide
Group Policy Creator Owners	**Moyen-élevé**	Création de GPO, peut déployer scripts malveillants via GPO	Administrator
DNSAdmins	**Moyen-élevé**	Chargement DLL arbitraire dans le service DNS (élévation SYSTEM sur DC)	Vide
DHCP Administrators	**Moyen**	Manipulation des baux DHCP, empoisonnement réseau potentiel	Vide
Remote Desktop Users	**Moyen**	Accès RDP aux DC — mouvement latéral facilité	Vide
Protected Users	Protecteur	Renforce la sécurité (désactive NTLM, RC4, délégation Kerberos)	Vide
Cert Publishers	**Élevé**		Vide

Groupe	Niveau de risque	Capacités réelles	Membres par défaut
		Publication de certificats dans AD — vecteur ESC8/ ESC9 ADCS	
`Exchange Windows Permissions`	**Critique**	WriteDACL sur le domaine → DCSync si Exchange installé (CVE-2019-0724)	Machines Exchange

Cette cartographie est le point de départ de tout audit. En environnement Exchange ou avec ADCS déployé, les vecteurs d'escalade via des groupes apparemment anodins comme *Cert Publishers* ou *Exchange Windows Permissions* sont particulièrement dangereux et souvent ignorés.

AdminSDHolder et SDProp : Mécanisme de Protection des Comptes Privilégiés

Active Directory intègre nativement un mécanisme de protection des comptes sensibles souvent méconnu des équipes IT : le couple **AdminSDHolder** et **SDProp**. Comprendre ce mécanisme est indispensable pour détecter certaines techniques de persistance très discrètes.

> ⚡ **Retour terrain**
>
> Dans mes missions de hardening Active Directory, le point le plus sous-estimé est la gestion du groupe 'Opérateurs de compte' et des délégations personnalisées créées au fil des années. Pour un groupe logistique de 1 500 utilisateurs, j'ai trouvé 312 ACL personnalisées dont 89 accordaient des permissions d'écriture sur des attributs sensibles

(adminCount, memberOf sur groupes privilégiés) à des comptes d'utilisateurs standards. La dette technique en sécurité AD est souvent invisible jusqu'au premier audit sérieux.

Fonctionnement du cycle SDProp (60 minutes)

L'objet `AdminSDHolder` (CN=AdminSDHolder,CN=System,DC=domaine,DC=fr) est un conteneur spécial dont les ACL servent de **modèle de référence**. Le processus **SDProp** (Security Descriptor Propagator) s'exécute toutes les 60 minutes sur le PDC Emulator et copie les ACL d'AdminSDHolder sur tous les objets dont l'attribut `adminCount` est égal à 1.

Cette protection garantit que même si un attaquant modifie les ACL d'un compte Domain Admin, elles seront réinitialisées à la prochaine exécution de SDProp. En contrepartie, tout attaquant qui parvient à modifier AdminSDHolder lui-même dispose d'une persistance qui survivra à tous les changements de mot de passe et révocations de groupe.

L'attribut adminCount=1 : implications pour l'audit

Lorsqu'un compte est ajouté à un groupe sensible, Active Directory positionne son attribut `adminCount` à 1 et ajuste ses ACL héritées. **Problème critique** : quand ce compte est retiré du groupe sensible, `adminCount` reste à 1 et les ACL héritées ne sont pas restaurées automatiquement. Il en résulte une désynchronisation silencieuse où des comptes ordinaires conservent indéfiniment des ACL réduites (blocage de l'héritage), ce qui peut masquer des droits ou créer des anomalies d'accès.

```
 # Vérifier l'ACL actuelle d'AdminSDHolder
$AdminSDHolder = "CN=AdminSDHolder,CN=System,DC=domaine,DC=fr"
(Get-Acl "AD:$AdminSDHolder").Access |
    Where-Object { $_.AccessControlType -eq "Allow" } |
    Select-Object IdentityReference, ActiveDirectoryRights |
    Sort-Object IdentityReference

# Forcer l'exécution immédiate de SDProp (utile lors d'un audit)
# Nécessite les droits DA
$RootDSE = [ADSI]"LDAP://RootDSE"
$RootDSE.Put("runProtectAdminGroupsTask", 1)
$RootDSE.SetInfo()
```

Persistance via AdminSDHolder : technique d'attaquant

Une technique documentée dans les frameworks offensifs consiste à ajouter un ACE (Access Control Entry) sur l'objet AdminSDHolder lui-même — par exemple, accorder *GenericAll* à un compte compromis de faible visibilité. Après 60 minutes, SDProp propagera cet ACE sur tous les comptes protégés, donnant à l'attaquant un contrôle total sur tous les comptes sensibles du domaine sans jamais apparaître dans les listes de membres des groupes privilégiés.

```
 # Détecter les ACE suspects sur AdminSDHolder
$AdminSDHolder = "CN=AdminSDHolder,CN=System,DC=domaine,DC=fr"
$ACL = Get-Acl "AD:$AdminSDHolder"
$ACL.Access | Where-Object {
    $_.IdentityReference -notmatch "^(BUILTIN|NT AUTHORITY|Domain Admins|Enterprise Admins|Adm
    $_.AccessControlType -eq "Allow"
} | Select-Object IdentityReference, ActiveDirectoryRights, ObjectType | Format-Table -AutoSize
```

Pour approfondir la protection des comptes sensibles, consultez notre article sur le groupe Protected Users et ses mécanismes de protection dans Active Directory.

Audit des Membres des Groupes Privilégiés avec PowerShell

L'audit manuel groupe par groupe est inefficace et incomplet. Le script suivant réalise un inventaire exhaustif de tous les groupes sensibles, en distinguant membres directs et membres récursifs, et génère un rapport HTML colorisé.

Enumération simple d'un groupe

```
 # Membres directs uniquement
Get-ADGroupMember -Identity "Domain Admins" | Select-Object Name, SamAccountName, ObjectClass

# Membres récursifs (inclut les membres des groupes imbriqués)
Get-ADGroupMember -Identity "Domain Admins" -Recursive | Select-Object Name, SamAccountName, Ob

# Comparer les deux pour identifier le nesting
$Direct = Get-ADGroupMember -Identity "Domain Admins" | Select-Object -ExpandProperty SamAccoun
$Recursive = Get-ADGroupMember -Identity "Domain Admins" -Recursive | Select-Object -ExpandProp
$ViaGroupes = $Recursive | Where-Object { $_ -notin $Direct }
Write-Host "Membres via nesting : $($ViaGroupes.Count)" -ForegroundColor Yellow
$ViaGroupes
```

Script complet d'inventaire multi-groupes avec rapport HTML

```powershell
 #!/usr/bin/env pwsh
# Audit-PrivilegedGroupsAD.ps1 — Inventaire complet des groupes sensibles AD
# Génère un rapport HTML colorisé

param(
    [string]$OutputPath = "C:\Audit\PrivilegedGroups_$(Get-Date -Format 'yyyyMMdd_HHmmss').html
    [string]$BaselinePath = "C:\Audit□aseline_privileged_groups.json"
)

Import-Module ActiveDirectory -ErrorAction Stop

$SensitiveGroups = @(
    "Domain Admins", "Enterprise Admins", "Schema Admins", "Administrators",
    "Account Operators", "Backup Operators", "Print Operators", "Server Operators",
    "Group Policy Creator Owners", "DNSAdmins", "DHCP Administrators",
    "Remote Desktop Users", "Cert Publishers", "Exchange Windows Permissions",
    "Organization Management", "Protected Users"
)

$Results = @{}
$Anomalies = @()

foreach ($GroupName in $SensitiveGroups) {
    try {
        $Group = Get-ADGroup -Identity $GroupName -Properties Description -ErrorAction Silently
        if (-not $Group) { continue }

        $DirectMembers = Get-ADGroupMember -Identity $GroupName -ErrorAction SilentlyContinue
        $RecursiveMembers = Get-ADGroupMember -Identity $GroupName -Recursive -ErrorAction Sile

        $Members = foreach ($M in $RecursiveMembers) {
            $Obj = switch ($M.objectClass) {
                "user"    { Get-ADUser $M -Properties Enabled, LastLogonDate, PasswordLastSet,
                "computer"{ Get-ADComputer $M -Properties Enabled, LastLogonDate }
                "group"   { Get-ADGroup $M -Properties MemberOf }
            }
            [PSCustomObject]@{
                Name            = $M.Name
                SamAccountName  = $M.SamAccountName
                ObjectClass     = $M.objectClass
                IsDirect        = ($DirectMembers.SamAccountName -contains $M.SamAccountName)
                Enabled         = if ($M.objectClass -eq "user") { $Obj.Enabled } else { "N/A"
                LastLogon       = if ($Obj.LastLogonDate) { $Obj.LastLogonDate.ToString("dd/MM/
```

```
                PasswordAge     = if ($M.objectClass -eq "user" -and $Obj.PasswordLastSet) {
                                      (New-TimeSpan -Start $Obj.PasswordLastSet -End (Get-Date
                                  } else { "N/A" }
                AdminCount      = if ($M.objectClass -eq "user") { $Obj.adminCount } else { "NA
            }
        }

        $Results[$GroupName] = $Members

        # Détection d'anomalies
        foreach ($M in $Members) {
            if ($M.Enabled -eq $false) {
                $Anomalies += "[$GroupName] Compte DÉSACTIVÉ membre : $($M.SamAccountName)"
            }
            if ($M.LastLogon -eq "Jamais") {
                $Anomalies += "[$GroupName] Compte sans connexion : $($M.SamAccountName)"
            }
            if ($M.PasswordAge -ne "N/A" -and [int]$M.PasswordAge -gt 365) {
                $Anomalies += "[$GroupName] Mot de passe > 1 an : $($M.SamAccountName) ($($M.Pa
            }
            if (-not $M.IsDirect) {
                $Anomalies += "[$GroupName] Membre VIA NESTING (indirect) : $($M.SamAccountName
            }
        }
    }
    catch { Write-Warning "Groupe $GroupName inaccessible : $_" }
}

# Comparaison avec baseline
$BaselineAlerts = @()
if (Test-Path $BaselinePath) {
    $Baseline = Get-Content $BaselinePath | ConvertFrom-Json
    foreach ($GroupName in $Results.Keys) {
        $CurrentSAMs = $Results[$GroupName].SamAccountName
        $BaselineSAMs = ($Baseline | Where-Object { $_.Group -eq $GroupName }).Members
        $NewMembers = $CurrentSAMs | Where-Object { $_ -notin $BaselineSAMs }
        $RemovedMembers = $BaselineSAMs | Where-Object { $_ -notin $CurrentSAMs }
        foreach ($N in $NewMembers) { $BaselineAlerts += "AJOUT non prévu [$GroupName] : $N" }
        foreach ($R in $RemovedMembers) { $BaselineAlerts += "SUPPRESSION [$GroupName] : $R" }
    }
}

# Sauvegarde nouvelle baseline
$NewBaseline = foreach ($G in $Results.Keys) {
```

```
    [PSCustomObject]@{ Group = $G; Members = $Results[$G].SamAccountName; Date = (Get-Date).ToS
}
$NewBaseline | ConvertTo-Json -Depth 5 | Set-Content $BaselinePath

# Génération rapport HTML
$HTMLHeader = @"
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Audit Groupes Privilégiés AD — $(Get-Date -Format 'dd/MM/yyyy HH:mm')</title>


<body>

Généré le $(Get-Date -Format 'dd/MM/yyyy à HH:mm:ss') | Domaine : $env:USERDNSDOMAIN


"@

$HTMLBody = ""

if ($BaselineAlerts.Count -gt 0) {
    $HTMLBody += "
```

🚨 Alertes Baseline (Modifications Détectées)

```
 "
    foreach ($A in $BaselineAlerts) { $HTMLBody += "
 $A
 " }
}

if ($Anomalies.Count -gt 0) {
    $HTMLBody += "
```

⚠️ Anomalies Détectées ($($Anomalies.Count))

```
"
    foreach ($A in $Anomalies) { $HTMLBody += "
$A
" }
}

$TotalPrivUsers = ($Results.Values | ForEach-Object { $_ } | Where-Object { $_.ObjectClass -eq
$HTMLBody += "
  Résumé : $($Results.Count) groupes analysés | $TotalPrivUsers comptes utilisateurs privilégi

"

foreach ($GroupName in ($Results.Keys | Sort-Object)) {
    $Members = $Results[$GroupName]
    $HTMLBody += "
```

$GroupName ($($Members.Count) membres)

```
"
    $HTMLBody += ""
    foreach ($M in $Members) {
        $Class = if (-not $M.IsDirect) { "indirect" } elseif ($M.Enabled -eq $false) { "disable
        $Direct = if ($M.IsDirect) { "Direct" } else { "Indirect (nesting)" }
        $HTMLBody += ""
    }
    $HTMLBody += "
```

Nom	SamAccountName	Type	Direct/Indirect	Activé	Dernièr
$($M.Name)	$($M.SamAccountName)	$($M.ObjectClass)	$Direct	$($M.Enabled)	$($M.La

```
"
}
```

```
$HTMLFooter = "</body></html>"
$null = New-Item -ItemType Directory -Force -Path (Split-Path $OutputPath)
($HTMLHeader + $HTMLBody + $HTMLFooter) | Out-File -FilePath $OutputPath -Encoding UTF8
Write-Host "Rapport généré : $OutputPath" -ForegroundColor Green
Write-Host "Anomalies : $($Anomalies.Count) | Changements baseline : $($BaselineAlerts.Count)"
```

Ce script combine inventaire, comparaison baseline et génération de rapport en une seule passe. Pour aller plus loin sur la surveillance des événements AD, consultez notre guide sur les événements Windows critiques à surveiller sur les contrôleurs de domaine.

Détection des Comptes avec adminCount=1 Non Membres de Groupes Sensibles

L'un des angles morts les plus fréquents lors des audits Active Directory est la population des comptes portant l'attribut `adminCount=1` sans être actuellement membres d'un groupe sensible. Ces comptes sont des **résidus d'élévation** : ils ont été membres d'un groupe sensible par le passé, l'attribut n'a pas été réinitialisé, et leurs ACL héritées restent bloquées.

Requête LDAP pour détecter ces comptes

```
 # Tous les comptes avec adminCount=1 (protection SDProp active)
Get-ADUser -LDAPFilter "(adminCount=1)" -Properties adminCount, MemberOf, LastLogonDate, Enable
    Select-Object Name, SamAccountName, Enabled, LastLogonDate,
        @{N="NbGroupes";E={$_.MemberOf.Count}} |
    Sort-Object NbGroupes | Format-Table -AutoSize

# Variante LDAP pure pour trouver les comptes adminCount=1 NON membres de Domain Admins
# (adapter le DN à votre domaine)
$DomainDN = (Get-ADDomain).DistinguishedName
$Filter = "(&(adminCount=1)(objectClass=user)(!(memberOf=CN=Domain Admins,CN=Users,$DomainDN))
Get-ADUser -LDAPFilter $Filter -Properties adminCount, MemberOf, Description |
    Select-Object Name, SamAccountName, Description, @{N="Groupes";E={($_.MemberOf | Get-ADGrou
```

Script de nettoyage PowerShell (remise à zéro adminCount)

```
 # Audit-AdminCount-Cleanup.ps1
# ATTENTION : tester en environnement de recette avant production

$SensitiveGroupsDN = @(
    "CN=Domain Admins,CN=Users", "CN=Enterprise Admins,CN=Users",
    "CN=Schema Admins,CN=Users", "CN=Administrators,CN=Builtin",
    "CN=Account Operators,CN=Builtin", "CN=Backup Operators,CN=Builtin",
    "CN=Print Operators,CN=Builtin", "CN=Server Operators,CN=Builtin",
    "CN=Group Policy Creator Owners,CN=Users"
)

$DomainDN = (Get-ADDomain).DistinguishedName
$SensitiveGroupsFull = $SensitiveGroupsDN | ForEach-Object { "$_,$DomainDN" }

$OrphansAdminCount = Get-ADUser -LDAPFilter "(adminCount=1)" -Properties adminCount, MemberOf
    Where-Object {
        $UserGroups = $_.MemberOf
        -not ($SensitiveGroupsFull | Where-Object { $UserGroups -contains $_ })
    }

Write-Host "Comptes adminCount=1 orphelins : $($OrphansAdminCount.Count)" -ForegroundColor Yel

foreach ($User in $OrphansAdminCount) {
    Write-Host "  Traitement : $($User.SamAccountName)" -ForegroundColor Cyan

    # Réinitialiser adminCount
    Set-ADUser $User -Replace @{ adminCount = 0 }

    # Réactiver l'héritage des ACL (dsacls ou via .NET)
    $UserDN = $User.DistinguishedName
    $ACL = Get-Acl "AD:$UserDN"
    $ACL.SetAccessRuleProtection($false, $true)  # Réactiver l'héritage
    Set-Acl "AD:$UserDN" $ACL

    Write-Host "  OK : adminCount réinitialisé, héritage ACL restauré pour $($User.SamAccountNa
}

Write-Host "Nettoyage terminé. $($OrphansAdminCount.Count) comptes corrigés." -ForegroundColor
```

Cette opération de nettoyage est souvent nécessaire avant de procéder à un audit de permissions DNS, car les résidus adminCount perturbent l'analyse. Voir notre article dédié sur l'audit des permissions DNS dans Active Directory.

Outils d'Audit Avancés : BloodHound, PingCastle, PurpleKnight

Les outils PowerShell natifs couvrent l'inventaire direct, mais la **détection des chemins d'escalade indirects** nécessite des outils spécialisés capables d'analyser l'intégralité du graphe des permissions AD.

BloodHound : Requêtes Cypher pour groupes privilégiés

BloodHound ingère les données AD via SharpHound et les stocke dans Neo4j. Les requêtes Cypher suivantes couvrent les cas les plus importants pour un audit des groupes sensibles :

```
 // 1. Tous les chemins vers Domain Admins depuis des utilisateurs non-DA
MATCH p=shortestPath(
  (u:User)-[*1..]->(g:Group {name:"DOMAIN ADMINS@DOMAINE.FR"})
)
WHERE NOT u.name STARTS WITH "DOMAIN ADMINS"
RETURN p

// 2. Tous les utilisateurs avec un chemin vers Domain Admins (direct ou indirect)
MATCH (u:User),(g:Group {name:"DOMAIN ADMINS@DOMAINE.FR"})
MATCH p=shortestPath((u)-[*1..]->(g))
RETURN u.name, length(p) as distance ORDER BY distance

// 3. Groupes non-Tier-0 avec membres Tier-0 (nesting dangereux)
MATCH (g1:Group)-[:MemberOf]->(g2:Group)
WHERE g2.name =~ ".*(DOMAIN ADMINS|ENTERPRISE ADMINS|SCHEMA ADMINS).*"
  AND NOT g1.name =~ ".*(ADMIN|OPERATOR|DOMAIN CONTROLLER).*"
RETURN g1.name as GroupeSource, g2.name as GroupeCible

// 4. Comptes de service (dont le nom contient "svc" ou "service") membres de groupes privilégi
MATCH (u:User)-[:MemberOf*1..]->(g:Group)
WHERE (u.name CONTAINS "SVC" OR u.name CONTAINS "SERVICE")
  AND g.name =~ ".*(ADMIN|OPERATOR).*"
RETURN u.name, collect(g.name) as GroupesPrivileges

// 5. Comptes avec DCSync rights (GetChanges + GetChangesAll)
MATCH (u)-[:DCSync|GetChanges|GetChangesAll]->(d:Domain)
RETURN u.name, labels(u) as type

// 6. Comptes avec WriteDACL ou GenericAll sur le domaine
MATCH (u:User)-[:WriteDACL|GenericAll]->(d:Domain)
RETURN u.name

// 7. Ordinateurs avec admin local sur un DC (mouvement latéral vers DC)
MATCH p=(c:Computer)-[:AdminTo]->(dc:Computer {unconstraineddelegation:true})
RETURN p

// 8. Kerberoastable users membres de groupes sensibles
MATCH (u:User)-[:MemberOf*1..]->(g:Group)
WHERE u.hasspn = true
  AND g.name =~ ".*(DOMAIN ADMINS|BACKUP OPERATORS|ACCOUNT OPERATORS).*"
RETURN u.name, u.serviceprincipalnames, collect(g.name)

// 9. Utilisateurs avec délégation sans contrainte (Unconstrained Delegation)
```

```
MATCH (u {unconstraineddelegation:true})
WHERE NOT u:DomainController
RETURN u.name, labels(u) as type

// 10. Chemins de compromission depuis les utilisateurs Kerberoastables
MATCH (u:User {hasspn:true})
MATCH p=shortestPath((u)-[*1..10]->(g:Group {name:"DOMAIN ADMINS@DOMAINE.FR"}))
RETURN u.name, length(p) as nbSauts, p

// 11. AdminSDHolder : objets ayant des ACE hors standards
MATCH (u)-[:GenericAll|WriteDACL|WriteOwner|GenericWrite]->
      (target {name:"ADMINSDHOLDER@DOMAINE.FR"})
RETURN u.name, labels(u)

// 12. Comptes avec PasswordNotRequired (attribut PASSWD_NOTREQD)
MATCH (u:User {passwordnotreqd:true})-[:MemberOf*1..]->(g:Group)
WHERE g.name =~ ".*(ADMIN|OPERATOR).*"
RETURN u.name, collect(g.name)
```

PingCastle : Analyse du score de risque groupes

PingCastle évalue automatiquement la configuration des groupes sensibles et génère un score de risque global (0 = parfait, 100 = critique). Les règles les plus importantes pour les groupes privilégiés sont :

P-AdminLogin : Comptes Domain Admin utilisés pour des connexions interactives non-DC

P-PrivilegedAccount : Trop de membres dans les groupes sensibles

P-ServiceDomainAdmin : Comptes de service configurés comme Domain Admin

P-ControlPathIndirectEveryone : Chemin de contrôle vers les groupes sensibles via Everyone

P-AdminSDHolderAnomalies : Anomalies sur l'objet AdminSDHolder

```
 # Lancement PingCastle en mode rapport complet (depuis un poste du domaine)
.\PingCastle.exe --healthcheck --server dc01.domaine.fr --output "C:\Audit\PingCastle"

# Rapport sécurité avancé avec toutes les règles
.\PingCastle.exe --healthcheck --level Full --server dc01.domaine.fr

# Export CSV des règles déclenchées pour intégration SIEM
.\PingCastle.exe --healthcheck --xmls --server dc01.domaine.fr
```

PurpleKnight : Indicateurs Tier 0

PurpleKnight (Semperis) fournit 100+ indicateurs de compromission spécifiques AD. Pour les groupes privilégiés, les indicateurs clés incluent :

Tier 0 Attack Path : Chemins d'escalade non documentés vers les actifs Tier 0

AdminSDHolder ACL Changes : Modifications récentes de l'ACL AdminSDHolder

Privileged Group Membership : Comptes de service, ordinateurs ou comptes désactivés dans les groupes Tier 0

SDProp Inconsistencies : Incohérences entre adminCount et l'appartenance réelle aux groupes

---

Nesting de Groupes : La Menace Cachée

Le **nesting de groupes** (imbrication de groupes dans d'autres groupes) est la technique d'escalade de privilèges la plus couramment exploitée dans les environnements Active Directory français. L'audit standard qui liste uniquement les membres directs de Domain Admins passe complètement à côté de cette menace.

Exemple d'attaque réelle via nesting

Scénario typique rencontré en audit : le groupe *GG_Finance_Responsables* est membre du groupe *GG_DSI_Acces_Serveurs*, qui est lui-même membre du groupe *SG_Backup_Operators*. Un contrôleur financier appartenant à GG_Finance_Responsables dispose ainsi de droits Backup Operators sur les contrôleurs de domaine — avec capacité de lire ntds.dit — sans qu'aucun administrateur n'en ait conscience.

Script PowerShell de récursion complète avec détection de dépendances circulaires

```powershell
# Get-NestedGroupMembers.ps1 — Récursion complète avec détection de cycles

function Get-NestedGroupMembers {
    param(
        [string]$GroupName,
        [int]$Depth = 0,
        [System.Collections.Generic.HashSet[string]]$VisitedGroups = $null
    )

    if ($null -eq $VisitedGroups) {
        $VisitedGroups = [System.Collections.Generic.HashSet[string]]::new([StringComparer]::O
    }

    # Détection de dépendance circulaire
    if ($VisitedGroups.Contains($GroupName)) {
        Write-Warning "CYCLE DÉTECTÉ : $GroupName (déjà visité) — arrêt de la récursion"
        return
    }
    $null = $VisitedGroups.Add($GroupName)

    $Indent = "  " * $Depth

    try {
        $Members = Get-ADGroupMember -Identity $GroupName -ErrorAction Stop
        foreach ($M in $Members) {
            $Prefix = if ($Depth -gt 0) { "$Indent[via $GroupName] " } else { "" }
            Write-Host "$Prefix$($M.Name) ($($M.SamAccountName)) — $($M.objectClass)" -Foregrou
                if ($M.objectClass -eq "group") { "Cyan" }
                elseif ($Depth -gt 0) { "Yellow" }
                else { "White" }
            )

            if ($M.objectClass -eq "group") {
                Get-NestedGroupMembers -GroupName $M.SamAccountName -Depth ($Depth + 1) -Visite
            }
        }
    }
    catch {
        Write-Warning "Impossible d'énumérer $GroupName : $_"
    }
}

# Usage : analyser tous les groupes sensibles
```

```
$SensitiveGroups = @("Domain Admins", "Enterprise Admins", "Backup Operators", "Account Operato
foreach ($G in $SensitiveGroups) {
    Write-Host "`n=== $G ===" -ForegroundColor Red -BackgroundColor DarkGray
    Get-NestedGroupMembers -GroupName $G
}

# Recherche des groupes "normaux" imbriqués dans des groupes sensibles
Write-Host "`n=== Groupes non-administratifs imbriqués dans groupes sensibles ===" -ForegroundC
foreach ($G in $SensitiveGroups) {
    $DirectGroups = Get-ADGroupMember -Identity $G | Where-Object { $_.objectClass -eq "group"
    foreach ($SubG in $DirectGroups) {
        $SubGroupObj = Get-ADGroup $SubG -Properties Description
        Write-Host "NESTING: '$($SubG.Name)' → '$G' | Description: $($SubGroupObj.Description)"
    }
}
```

La détection des chemins d'escalade via nesting s'intègre dans une démarche de tiering. Pour en savoir plus sur la séparation des niveaux d'accès, consultez notre article sur les Privileged Access Workstations (PAW) en 2026.

Monitoring Continu : Alertes sur Modifications des Groupes Sensibles

L'audit ponctuel ne suffit pas. Un monitoring en temps réel des modifications de membres des groupes sensibles est indispensable pour détecter une compromission en cours ou une élévation non autorisée.

Event IDs Windows à surveiller

Event ID	Description	Type de groupe	Criticité
4728	Membre ajouté à un groupe de sécurité global	Global (ex: Domain Admins)	Critique
4732	Membre ajouté à un groupe de sécurité local	Local (ex: Administrators, Backup Operators)	Critique
4756	Membre ajouté à un groupe de sécurité universel	Universel (ex: Enterprise Admins)	Critique
4735	Groupe de sécurité local modifié (description, nom...)	Local	Élevé
4737	Groupe de sécurité global modifié	Global	Élevé
4755	Groupe de sécurité universel modifié	Universel	Élevé
4729	Membre retiré d'un groupe de sécurité global	Global	Moyen

Règles Wazuh pour groupes sensibles

```
 <!-- wazuh-rules-privileged-groups.xml -->
<group name="windows,active_directory,privileged_groups,">

  <!-- Ajout membre Domain Admins (Event 4728) -->
  <rule id="100200" level="15">
    <if_group>windows</if_group>
    <id>4728</id>
    <field name="win.eventdata.targetUserName" type="pcre2">(?i)(Domain Admins|Enterprise Admi
    <description>CRITIQUE: Ajout d'un membre dans un groupe Tier-0 AD ($(win.eventdata.memberNa
    <options>alert_by_email</options>
    <group>ad_privilege_escalation,pci_dss_10.2.4,gdpr_IV_35.7.d,nist_800_53_AU.14</group>
  </rule>

  <!-- Ajout membre groupes Backup/Account/Print Operators (Event 4732) -->
  <rule id="100201" level="12">
    <if_group>windows</if_group>
    <id>4732</id>
    <field name="win.eventdata.targetUserName" type="pcre2">(?i)(Backup Operators|Account Opera
    <description>Ajout membre groupe opérateurs DC sensible: $(win.eventdata.memberName) → $(wi
    <options>alert_by_email</options>
    <group>ad_privilege_escalation</group>
  </rule>

  <!-- Ajout membre Enterprise Admins (Event 4756 - groupe universel) -->
  <rule id="100202" level="15">
    <if_group>windows</if_group>
    <id>4756</id>
    <field name="win.eventdata.targetUserName" type="pcre2">(?i)(Enterprise Admins|Schema Admi
    <description>CRITIQUE: Ajout dans groupe forêt privilégié: $(win.eventdata.memberName) → $
    <options>alert_by_email</options>
    <group>ad_privilege_escalation</group>
  </rule>

  <!-- Modification du groupe Domain Admins (Event 4737) -->
  <rule id="100203" level="12">
    <if_group>windows</if_group>
    <id>4737</id>
    <field name="win.eventdata.targetUserName" type="pcre2">(?i)(Domain Admins|GPO Creator Owne
    <description>Modification attributs groupe sensible: $(win.eventdata.targetUserName) par $
    <group>ad_group_modification</group>
  </rule>

  <!-- Modification AdminSDHolder -->
```

```
  <rule id="100204" level="15">
    <if_group>windows</if_group>
    <id>4662</id>
    <field name="win.eventdata.objectName" type="pcre2">(?i)AdminSDHolder</field>
    <field name="win.eventdata.accessMask" type="pcre2">%%1539</field>
    <description>CRITIQUE: Modification de l'objet AdminSDHolder — persistance possible (auteu
    <options>alert_by_email</options>
    <group>ad_adminsdholder,persistence</group>
  </rule>

</group>
```

KQL Microsoft Sentinel : Alerte ajout Domain Admins

```
 // KQL — Alerte : ajout dans un groupe Tier-0 AD
// Analytics Rule: Privileged Group Membership Change
let SensitiveGroups = dynamic([
    "Domain Admins", "Enterprise Admins", "Schema Admins",
    "Administrators", "Backup Operators", "Account Operators",
    "Print Operators", "Server Operators", "DNSAdmins"
]);
SecurityEvent
| where TimeGenerated > ago(1h)
| where EventID in (4728, 4732, 4756)
| where TargetUserName has_any (SensitiveGroups)
| extend
    AddedMember    = MemberName,
    GroupModified  = TargetUserName,
    PerformedBy    = SubjectUserName,
    DC             = Computer
| project TimeGenerated, EventID, GroupModified, AddedMember, PerformedBy, DC, Activity
| extend Severity = case(
    GroupModified in ("Domain Admins","Enterprise Admins","Schema Admins"), "High",
    GroupModified in ("Backup Operators","Account Operators","Administrators"), "Medium",
    "Low"
)
| order by TimeGenerated desc

// KQL — Corrélation : ajout DA suivi d'une connexion interactive (mouvement latéral rapide)
let DaAdditions = SecurityEvent
    | where EventID in (4728, 4756)
    | where TargetUserName has "Domain Admins"
    | project AddTime = TimeGenerated, AddedSAM = MemberSid;
SecurityEvent
| where EventID == 4624
| where LogonType in (2, 10)  // interactif ou RDP
| join kind=inner DaAdditions on $left.TargetUserSid == $right.AddedSAM
| where TimeGenerated between (AddTime .. (AddTime + 2h))
| project LogonTime = TimeGenerated, AddTime, AccountName, Computer, LogonType, IpAddress
| extend Alert = "Connexion interactive dans les 2h suivant l'ajout DA"
```

Pour une configuration complète de la journalisation sur les DC, voir notre guide sur la surveillance des événements Windows sur les contrôleurs de domaine.

Plan de Remédiation : Réduction de la Surface d'Attaque

Auditer sans remédier n'est qu'un exercice académique. Une fois les anomalies identifiées, un plan structuré s'impose, en s'appuyant sur le modèle de tiering Microsoft et le principe du moindre privilège.

Modèle de Tiering Microsoft (Tier 0 / 1 / 2)

Le modèle de tiering administrative définit trois niveaux d'isolation :

Tier 0 : Contrôleurs de domaine, ADCS, ADFS, Azure AD Connect. Seuls les comptes Tier 0 (Domain Admins, Enterprise Admins) y ont accès.

Tier 1 : Serveurs applicatifs, hyperviseurs. Comptes Tier 1 dédiés, jamais membres de groupes Tier 0.

Tier 2 : Postes de travail utilisateurs. Comptes Tier 2 (helpdesk), isolation totale des niveaux supérieurs.

L'isolation entre tiers est appliquée via des GPO de restriction de connexion (*Deny logon through Remote Desktop Services*, *Deny logon as a batch job*) couplées à des Authentication Policy Silos. Pour les détails d'implémentation, consultez notre article sur les Authentication Policy Silos dans Windows Server 2025.

Délégation fine : alternatives à l'élévation Domain Admin

```
 # Exemple : déléguer la gestion des mots de passe sur une OU sans Domain Admin
# Utiliser l'Assistant Délégation de contrôle ou dsacls

# Accorder Reset-Password sur l'OU Utilisateurs-Paris uniquement
$OU = "OU=Utilisateurs-Paris,DC=domaine,DC=fr"
$HelpDeskSID = (Get-ADGroup "GG_HelpDesk_Paris").SID.Value

# Droit Reset Password (GUID: 00299570-246d-11d0-a768-00aa006e0529)
$ResetPwdGUID = [GUID]"00299570-246d-11d0-a768-00aa006e0529"
$ACL = Get-Acl "AD:$OU"
$Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule(
    [System.Security.Principal.SecurityIdentifier]$HelpDeskSID,
    [System.DirectoryServices.ActiveDirectoryRights]::ExtendedRight,
    [System.Security.AccessControl.AccessControlType]::Allow,
    $ResetPwdGUID,
    [System.DirectoryServices.ActiveDirectorySecurityInheritance]::Descendents,
    [GUID]"bf967aba-0de6-11d0-a285-00aa003049e2"  # GUID User objectClass
)
$ACL.AddAccessRule($Rule)
Set-Acl "AD:$OU" $ACL
Write-Host "Délégation Reset-Password accordée à GG_HelpDesk_Paris sur $OU" -ForegroundColor G
```

JIT (Just-In-Time) via PAM et Entra PIM

La meilleure pratique 2026 consiste à n'octroyer les appartenances aux groupes sensibles que **temporairement et à la demande** :

Microsoft PAM (Privileged Access Management AD) : Pour les environnements on-premises, permet des appartenances à durée limitée via `Add-ADGroupMember -MemberTimeToLive` (Windows Server 2016+ en mode fonctionnel forêt).

Microsoft Entra PIM (Privileged Identity Management) : Pour les environnements hybrides/cloud, activation JIT avec workflow d'approbation, durée configurable (4h par défaut), auditabilité complète.

CyberArk, BeyondTrust, Delinea : Solutions PAM tierces pour les grandes entreprises françaises avec gestion des secrets et rotation automatique.

```
 # Ajout temporaire avec TTL (Windows Server 2016+, niveau forêt 2016)
# Expire automatiquement après 8 heures
$TTL = New-TimeSpan -Hours 8
Add-ADGroupMember -Identity "Domain Admins" -Members "jean.dupont" -MemberTimeToLive $TTL
Write-Host "Accès temporaire accordé pour 8h — expiration : $((Get-Date).Add($TTL))" -Foregrou

# Vérifier les membres avec TTL actifs
Get-ADGroup "Domain Admins" -Properties member |
    Select-Object -ExpandProperty member |
    ForEach-Object { Get-ADObject $_ -Properties msDS-MemberTimeToLive } |
    Where-Object { $_."msDS-MemberTimeToLive" -ne $null }
```

Conformité NIS 2 et ISO 27001 : Exigences sur les Accès Privilégiés

En France, les organisations soumises à NIS 2 (directive européenne transposée par la loi n°2023-703 du 3 août 2023) et à ISO 27001 ont des obligations explicites sur la gestion des accès privilégiés. L'audit des groupes AD n'est pas un simple exercice technique : c'est une obligation réglementaire.

NIS 2 — Article 21§2.i : Gestion des accès

L'article 21, paragraphe 2, alinéa i de la directive NIS 2 exige la mise en place de **politiques de contrôle d'accès et de gestion des actifs** incluant explicitement le principe du moindre privilège. Pour les entités essentielles

(OIV, secteurs bancaire, santé, énergie…) et importantes (ETI de plus de 250 salariés ou 50M€ de CA), cela se traduit concrètement par :

Inventaire documenté et à jour des comptes privilégiés (fréquence : au moins annuelle, recommandation ANSSI : trimestrielle)

Revue périodique des appartenances aux groupes sensibles avec validation formelle

Journalisation et alertes sur toute modification de groupe sensible (Event IDs 4728, 4732, 4756)

Procédure de révocation immédiate documentée

ISO 27001:2022 — Contrôle A.8.2 : Droits d'accès privilégiés

Le contrôle A.8.2 (anciennement A.9.2.3 dans la version 2013) exige que :

> *« L'attribution et l'utilisation de droits d'accès privilégiés doivent être limitées et maîtrisées. »*

Les mesures attendues dans un SMSI certifié ISO 27001 incluent l'identification des comptes nécessitant des droits privilégiés, une procédure d'autorisation formelle, une revue régulière (a minima semestrielle), et la suppression automatique des droits lors du changement de fonction ou de départ.

Guide IAM de l'ANSSI

Le guide ANSSI *« Recommandations relatives à l'administration sécurisée des systèmes d'information »* (PA-022) recommande pour les groupes AD :

Maximum 5 comptes Domain Admins actifs (hors comptes d'urgence break-glass)

Comptes DA dédiés, distincts des comptes utilisateurs quotidiens

Comptes DA sans boîte mail, sans accès Internet

Rotation des mots de passe DA tous les 90 jours maximum

Schema Admins et Enterprise Admins : membres zéro en dehors des opérations de maintenance planifiées

Checklist de conformité pour un audit AD en contexte français

```powershell
 # Génération d'un rapport de conformité NIS2/ISO27001
$Report = @()

# Vérification : nombre de Domain Admins
$DACount = (Get-ADGroupMember "Domain Admins" -Recursive | Where-Object { $_.objectClass -eq "u
$Report += [PSCustomObject]@{
    Contrôle = "NIS2/ISO A.8.2 — Nombre Domain Admins"
    Valeur   = $DACount
    Statut   = if ($DACount -le 5) { "CONFORME" } else { "NON CONFORME ($DACount > 5)" }
    Référence = "ANSSI PA-022 § 3.2"
}

# Vérification : Enterprise Admins vide
$EACount = (Get-ADGroupMember "Enterprise Admins").Count
$Report += [PSCustomObject]@{
    Contrôle = "ANSSI — Enterprise Admins vide en dehors maintenance"
    Valeur   = $EACount
    Statut   = if ($EACount -eq 0) { "CONFORME" } else { "À JUSTIFIER ($EACount membre(s))" }
    Référence = "ANSSI PA-022 § 3.3"
}

# Vérification : Schema Admins vide
$SACount = (Get-ADGroupMember "Schema Admins").Count
$Report += [PSCustomObject]@{
    Contrôle = "ANSSI — Schema Admins vide en dehors maintenance"
    Valeur   = $SACount
    Statut   = if ($SACount -eq 0) { "CONFORME" } else { "À JUSTIFIER ($SACount membre(s))" }
    Référence = "ANSSI PA-022 § 3.3"
}

# Vérification : comptes DA avec boîte mail
$DAWithMail = Get-ADGroupMember "Domain Admins" -Recursive |
    Where-Object { $_.objectClass -eq "user" } |
    ForEach-Object { Get-ADUser $_ -Properties mail } |
    Where-Object { $_.mail -ne $null }
$Report += [PSCustomObject]@{
    Contrôle = "ANSSI — Comptes DA sans boîte mail"
    Valeur   = "$($DAWithMail.Count) DA ont une adresse mail"
    Statut   = if ($DAWithMail.Count -eq 0) { "CONFORME" } else { "NON CONFORME" }
    Référence = "ANSSI PA-022 § 3.4"
}

$Report | Format-Table -AutoSize
```

Questions Fréquentes sur l'Audit des Groupes Privilégiés

Quelle est la différence entre un membre direct et un membre récursif dans un groupe AD ?

Un **membre direct** est explicitement listé dans l'attribut `member` du groupe. Un **membre récursif** (ou indirect) appartient au groupe via l'imbrication : son compte est membre d'un groupe intermédiaire, qui est lui-même membre du groupe cible. La commande `Get-ADGroupMember` sans le paramètre `-Recursive` ne retourne que les membres directs. Or, les attaquants exploitent précisément le nesting pour accéder à des privilèges sans apparaître dans la liste directe du groupe sensible. Un audit complet doit toujours utiliser `-Recursive`.

Pourquoi mon compte a-t-il adminCount=1 alors qu'il n'est membre d'aucun groupe sensible ?

C'est un résidu classique. Active Directory positionne `adminCount=1` lorsqu'un compte est ajouté à un groupe protégé par SDProp, mais **ne le remet jamais à zéro automatiquement** lors du retrait. Ce compte conserve indéfiniment ses ACL héritées bloquées, ce qui peut masquer des droits ou créer des anomalies d'héritage. La correction nécessite deux actions distinctes : remettre `adminCount` à 0 via `Set-ADUser` et réactiver l'héritage ACL via `Set-Acl` avec `SetAccessRuleProtection($false, $true)`.

Combien de membres Domain Admins est-il raisonnable d'avoir en production ?

L'ANSSI recommande un maximum de 5 comptes Domain Admins actifs dans un domaine de production, en dehors des comptes d'urgence break-glass (généralement 1 à 2 comptes, stockés sous enveloppe scellée). En pratique, lors de nos audits en France, nous constatons régulièrement des dizaines de membres — parfois plus de 50 dans les grandes organisations. Chaque compte DA supplémentaire est un vecteur de compromission potentiel. La

bonne pratique est : zéro DA actif en temps normal, utilisation JIT via PAM/PIM pour les opérations ponctuelles.

BloodHound nécessite-t-il un agent installé sur les DC ?

Non. BloodHound utilise **SharpHound** comme collecteur de données. SharpHound s'exécute depuis n'importe quel poste du domaine avec un compte utilisateur standard — il ne nécessite pas de droits administrateurs pour la collecte basique des relations de groupe, des sessions actives et des ACL. Pour les audits Red Team, SharpHound peut même être exécuté depuis un compte de faibles privilèges et révèle des chemins d'escalade que les équipes IT ignorent. La collecte complète (avec ACL) requiert un compte avec droits de lecture AD étendus, généralement un compte de domaine standard suffit.

Comment implémenter une revue des groupes privilégiés conforme NIS 2 ?

Une revue conforme NIS 2 implique : (1) une fréquence au minimum trimestrielle pour les groupes Tier 0 (mensuelle recommandée), (2) un workflow formel avec validation par le RSSI ou le responsable sécurité, (3) une trace d'audit documentée (date, validateur, membres approuvés/révoqués), (4) un délai de révocation sous 24h pour les comptes dont le besoin n'est plus justifié. Le script d'audit fourni dans cet article peut être intégré dans un workflow automatisé via Microsoft Power Automate ou un outil ITSM (ServiceNow, Jira Service Management) pour générer des tickets de revue périodiques.

Pour compléter cet audit, consultez également notre guide sur la vérification des mots de passe Active Directory avec Have I Been Pwned pour détecter les comptes avec des identifiants compromis dans les bases de fuites.

L'audit des groupes doit être complété par l'audit des délégations Active Directory avec BloodHound et Adalanche, qui révèle les chemins d'attaque

indirects vers Domain Admin inaccessibles à une simple analyse des membres.

À RETENIR

Les groupes Tier 0 (Domain Admins, Enterprise Admins, Schema Admins) doivent tendre vers zéro membres permanents : toute appartenance doit être justifiée, documentée et temporaire via JIT.

L'attribut adminCount=1 sur des comptes non membres de groupes sensibles est un indicateur d'hygiène défaillante — et potentiellement de persistance d'attaquant : auditez et corrigez systématiquement.

Le nesting de groupes est la menace la plus sous-estimée : n'auditez jamais les membres directs sans analyser les membres récursifs et les chemins d'escalade BloodHound.

Les Event IDs 4728, 4732 et 4756 doivent déclencher une alerte P1 dans votre SIEM avec notification immédiate au RSSI — toute modification de groupe sensible en dehors d'une maintenance planifiée est une anomalie à investiguer.

NIS 2 et ISO 27001 A.8.2 imposent une revue formelle et documentée des accès privilégiés : intégrez ces scripts dans votre programme d'audit trimestriel pour produire des preuves d'audit exploitables lors d'une certification ou d'une inspection ANSSI.

Références et documentation

Microsoft Learn — Sécurité Windows Server

ANSSI — Guide de sécurisation Active Directory

MITRE ATT&CK — Techniques Active Directory

Questions fréquentes

Quels sont les prérequis pour réaliser un audit Active Directory ?

La réponse dépend du contexte organisationnel, mais les principes fondamentaux restent constants : évaluation du périmètre, identification des actifs critiques et priorisation par risque réel plutôt que par vulnérabilité isolée.

Combien de temps dure un audit Active Directory complet ?

Une approche structurée et documentée est clé. Les outils et méthodologies évoluent rapidement — rester informé des ressources ANSSI, NIST et MITRE ATT&CK est indispensable pour adapter les recommandations génériques à chaque contexte.

Quelles remédiations traiter en priorité après un audit Active Directory ?

Les ressources officielles (ANSSI, CISA, CERT-FR) constituent le point de départ. Complétées par des retours d'expérience terrain, elles permettent

d'adapter les recommandations aux réalités opérationnelles de chaque organisation.

Conclusion

La sécurité d'Active Directory est un programme continu. Les chemins d'attaque vers le Tier 0 évoluent à chaque changement de configuration ou déploiement. Audit régulier, monitoring BloodHound et revue des délégations constituent le socle d'une posture durable.

Vous souhaitez évaluer la sécurité de votre Active Directory et identifier les chemins d'attaque ?
Demandez un audit Active Directory ou contactez-nous directement.