Honeypots Active Directory : Honey Users, Honey SPNs et Détection d'Intrusion 2026

Déployez des honeypots Active Directory : honey users, honey SPNs, honey shares et canary tokens. Détection Kerberoasting et latéralisation avec Wazuh.

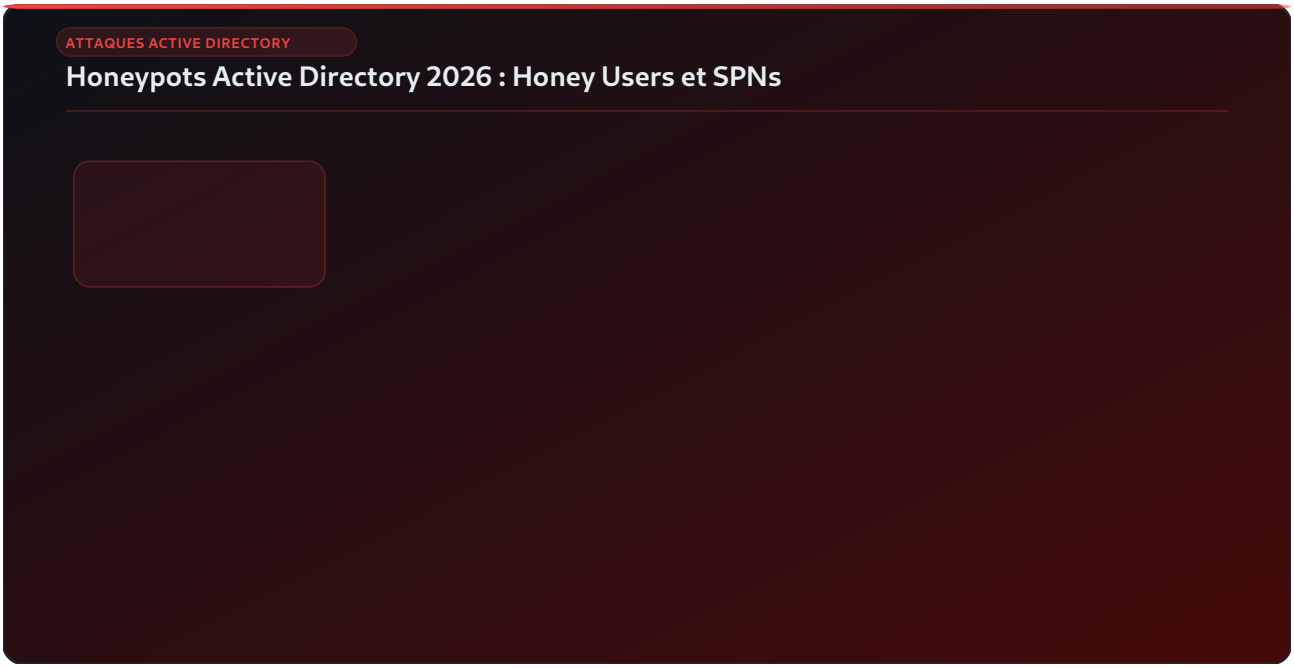
📋 EN BREF

En bref

- Les honeypots Active Directory (honey users, honey SPNs, honey shares) offrent un taux de faux positifs quasi-nul pour la détection d'intrusion en environnement Microsoft.
- La surveillance des Event IDs 4769 (TGS-REP RC4), 4768 (AS-REQ), 4625 (échec auth) et 5136 (modification objet AD) constitue le socle de détection des leurres AD.
- Les canary tokens complètent les honeypots AD en détectant l'ouverture de fichiers leurres et l'utilisation de credentials factices exfiltrés.
- Wazuh et Microsoft Sentinel s'intègrent nativement avec les événements Windows pour déclencher des alertes critiques sur toute interaction avec les leurres.
- Le déploiement de honeypots AD constitue une mesure de détection proactive conforme à l'article 21 de la directive NIS 2 et aux recommandations ANSSI.

Dans un contexte où les attaques ciblant Active Directory représentent le vecteur de compromission le plus fréquent dans les incidents traités en France en 2025-2026, les RSSI et équipes SOC cherchent des mécanismes de détection à haute fidélité capables d'identifier une

intrusion avant la compromission totale du domaine. À Paris comme en région, les cabinets d'audit et les équipes de réponse à incident constatent que les attaquants — qu'ils exploitent des vulnérabilités CVE récentes ou des techniques éprouvées comme le Kerberoasting ou le Pass-the-Hash — passent en moyenne 72 heures dans un environnement Active Directory avant de déclencher leur charge finale. Les honeypots Active Directory — honey users, honey SPNs, honey shares et canary tokens — constituent une réponse tactique et économique à ce problème : en créant des appâts indétectables pour les outils d'énumération automatisés et les attaquants humains, ils permettent de détecter des comportements malveillants en quelques minutes après la première interaction, avec un taux de faux positifs proche de zéro. Cet article technique détaille l'architecture complète d'un dispositif de leurres AD, les scripts PowerShell de déploiement, les règles de détection Wazuh et les requêtes KQL Sentinel, dans une optique de conformité NIS 2 et de réduction du Mean Time To Detect (MTTD) pour les organisations françaises soumises à la réglementation européenne sur la cybersécurité.



La défense périmétrique classique — firewall, antivirus, EDR — suppose que l'attaquant est à l'extérieur du réseau. La réalité des incidents Active Directory est différente : un attaquant qui a

compromis un premier poste de travail via phishing ou exploitation de vulnérabilité est déjà à l'intérieur. Il dispose d'un accès légitime à l'annuaire LDAP, peut énumérer tous les objets AD sans déclencher d'alerte, et utilise des outils comme BloodHound, PowerView ou Rubeus qui s'appuient sur des protocoles Windows standards.

Les honeypots AD renversent cette asymétrie en créant des **détecteurs à haute fidélité intégrés à l'annuaire lui-même**. Contrairement aux signatures d'EDR qui peuvent être contournées, un honey user ne génère aucun trafic légitime. Toute interaction avec lui est par définition anormale. Ce principe simple produit des alertes d'une précision remarquable :

Taux de vrais positifs > 99% : aucune activité légitime ne cible les leurres.

Détection précoce : l'attaquant interagit avec les leurres lors de la phase de reconnaissance, avant la latéralisation.

Coût minimal : un compte AD et quelques règles de monitoring suffisent.

Preuves forensiques : les événements Windows générés constituent des preuves d'incident exploitables.

Complémentarité EDR : les honeypots détectent ce que l'EDR ne voit pas (énumération LDAP légitime, accès Kerberos natifs).

En France, l'ANSSI recommande explicitement l'usage de leurres dans son guide de durcissement Active Directory (CERT-FR R37). La mise en conformité NIS 2 accélère l'adoption de ces techniques dans les Opérateurs d'Importance Vitale (OIV) et les Entités Essentielles (EE) soumises à la directive.

Honey Users : Comptes Appâts pour Détecter Kerberoasting et Password Spray

Un honey user est un compte Active Directory sans utilisation légitime, conçu pour attirer les attaquants qui effectuent de l'énumération ou des attaques par force brute. Sa conception doit le rendre crédible tout en étant parfaitement surveillé.

> ⚡ **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.

Caractéristiques d'un honey user efficace

Nom réaliste : `svc_backup_fileserver`, `jean.martin.legacy`, `admin.reporting`

Description convaincante : "Service account - Legacy ERP integration" ou "Compte de service reporting - Ne pas désactiver"

Historique plausible : date de création ancienne (2-3 ans), password change history

Groupes AD non-privilégiés : membre de groupes métier réels pour apparaître dans l'énumération BloodHound

SPN configuré (pour attirer le Kerberoasting) : `HTTP/legacy-erp.domaine.local`

Jamais connecté : LastLogon = null ou date ancienne, LastLogonTimestamp cohérent

Mot de passe long et aléatoire : le compte n'est jamais utilisé pour se connecter, le password peut être de 127 caractères

Création PowerShell d'un Honey User

```
# Fonction de création d'un honey user avec SPN
function New-HoneyUser {
    param(
        [string]$Name,
        [string]$SamAccountName,
        [string]$Description,
        [string]$OUPath,
        [string]$SPN,
        [string]$Department = "IT Operations"
    )

    # Génération d'un mot de passe fort aléatoire (jamais utilisé)
    $Password = [System.Web.Security.Membership]::GeneratePassword(64, 12)
    $SecurePassword = ConvertTo-SecureString $Password -AsPlainText -Force

    # Création du compte
    New-ADUser `
        -Name $Name `
        -SamAccountName $SamAccountName `
        -UserPrincipalName "$SamAccountName@$env:USERDNSDOMAIN" `
        -Description $Description `
        -Department $Department `
        -Path $OUPath `
        -AccountPassword $SecurePassword `
        -Enabled $true `
        -PasswordNeverExpires $true `
        -CannotChangePassword $true

    # Assignation du SPN pour attirer le Kerberoasting
    if ($SPN) {
        setspn -A $SPN "$env:USERDOMAIN\$SamAccountName"
        Write-Host "[+] SPN créé : $SPN pour $SamAccountName"
    }

    # Manipulation de LastLogonTimestamp pour paraître crédible
    # Fixer à une date ancienne (18 mois) pour simuler un compte de service dormant
    $OldDate = (Get-Date).AddDays(-540)
    $FileTimeDate = $OldDate.ToFileTime()
    Set-ADUser -Identity $SamAccountName -Replace @{lastLogonTimestamp = $FileTimeDate}

    # Suppression des attributs trop "propres"
    Set-ADUser -Identity $SamAccountName `
```

```
        -Replace @{
            whenChanged = $OldDate
            pwdLastSet  = (Get-Date).AddDays(-400).ToFileTime()
        } -ErrorAction SilentlyContinue

    Write-Host "[+] Honey user créé : $SamAccountName avec SPN $SPN"
    Write-Host "[!] SURVEILLANCE CRITIQUE : tout event 4769/4768/4625 sur ce compte = INTRUSION"
}

# Exemple de déploiement
$OU = "OU=Service-Accounts,OU=IT,DC=domaine,DC=local"

New-HoneyUser `
    -Name "svc_backup_legacy" `
    -SamAccountName "svc_backup_legacy" `
    -Description "Service account - Legacy backup integration (DO NOT DELETE)" `
    -OUPath $OU `
    -SPN "HTTP/legacy-backup.domaine.local"

New-HoneyUser `
    -Name "admin.reporting.svc" `
    -SamAccountName "admin.reporting.svc" `
    -Description "Reporting service - SQL Integration Services" `
    -OUPath $OU `
    -SPN "MSSQLSvc/sql-reporting.domaine.local:1433"
```

Événements Windows à surveiller sur les Honey Users

Une fois le honey user créé, configurez l'audit sur le contrôleur de domaine. Référez-vous à notre guide complet sur les Windows Events à surveiller sur les contrôleurs de domaine pour la configuration de base.

Event 4769 (Kerberos TGS-REP) : demande de ticket de service pour le SPN du honey user → Kerberoasting détecté. Filtrer sur `ServiceName = svc_backup_legacy` ET `TicketEncryptionType = 0x17` (RC4).

Event 4768 (Kerberos AS-REQ) : demande de TGT pour le honey user → tentative d'authentification ou AS-REP Roasting.

Event 4625 (Échec d'authentification) : password spray ou force brute sur le honey user.

Event 4776 (Validation NTLM) : tentative d'authentification NTLM sur le honey user.

Pour une configuration avancée de l'audit avec Sysmon sur les contrôleurs de domaine, consultez notre article sur Sysmon pour contrôleurs de domaine 2026.

Honey SPNs : Détecter les Attaques Kerberoasting Ciblées

Le Kerberoasting est l'une des attaques les plus fréquentes en post-exploitation Active Directory. Elle consiste à demander des tickets TGS pour tous les comptes avec SPN, puis à les craquer hors ligne. Les honey SPNs sont spécifiquement conçus pour piéger cette technique.

Principes du Honey SPN

Un honey SPN est un Service Principal Name enregistré sur un compte inactif, dont le service n'existe pas. Aucun système légitime ne demandera jamais un ticket pour ce SPN. Toute demande Event 4769 est donc malveillante avec une certitude absolue.

```
# Création d'une batterie de Honey SPNs sur différents comptes
function New-HoneySPN {
    param(
        [string]$AccountName,
        [string[]]$SPNList
    )

    foreach ($spn in $SPNList) {
        $result = setspn -A $spn $AccountName 2>&1
        if ($LASTEXITCODE -eq 0) {
            Write-Host "[+] Honey SPN ajouté : $spn sur $AccountName"
        } else {
            Write-Warning "Erreur SPN $spn : $result"
        }
    }
}

# SPNs factices sur comptes de service inactifs
$HoneySPNs = @(
    "HTTP/legacy-portal.domaine.local",
    "MSSQLSvc/db-archive.domaine.local:1433",
    "HTTP/intranet-old.domaine.local:8080",
    "WSMAN/mgmt-legacy.domaine.local",
    "RestrictedKrbHost/fileserver-2019.domaine.local"
)

New-HoneySPN -AccountName "svc_legacy_erp" -SPNList $HoneySPNs

# Vérification des SPNs enregistrés
setspn -L svc_legacy_erp
```

Détection Kerberoasting via Event 4769 avec filtre RC4

La majorité des outils de Kerberoasting (Rubeus, Invoke-Kerberoast) demandent des tickets RC4 (etype 0x17) par défaut car ils sont plus rapides à craquer. Ce filtre combiné au honey SPN garantit une détection quasi-infaillible :

```
<!-- Requête XPath pour l'observateur d'événements Windows -->
<QueryList>
  <Query Id="0" Path="Security">
    <Select Path="Security">
      *[System[EventID=4769]] and
      *[EventData[Data[@Name='ServiceName']='svc_legacy_erp$']] and
      *[EventData[Data[@Name='TicketEncryptionType']='0x17']]
    </Select>
  </Query>
</QueryList>
```

Honey Shares et Honey Files : Détection de Latéralisation

Après la compromission d'un premier compte, les attaquants se déplacent latéralement en accédant aux partages réseau pour trouver des credentials, des scripts de configuration ou des données sensibles. Les honey shares et honey files constituent des pièges à ce stade critique de l'attaque.

Création d'un Honey Share SMB

```
# Fonction de création d'un honey share avec audit activé
function New-HoneyShare {
    param(
        [string]$ShareName,
        [string]$LocalPath,
        [string]$Description = "Archived files - IT Department"
    )

    # Création du répertoire local
    New-Item -ItemType Directory -Path $LocalPath -Force | Out-Null

    # Création du partage SMB
    New-SmbShare `
        -Name $ShareName `
        -Path $LocalPath `
        -Description $Description `
        -ReadAccess "Domain Users" `
        -FullAccess "Domain Admins"

    # Activation de l'audit d'accès objet sur le dossier
    $Acl = Get-Acl $LocalPath
    $AuditRule = New-Object System.Security.AccessControl.FileSystemAuditRule(
        "Everyone",
        "ReadData, WriteData, Delete",
        "ContainerInherit,ObjectInherit",
        "None",
        "Success,Failure"
    )
    $Acl.AddAuditRule($AuditRule)
    Set-Acl -Path $LocalPath -AclObject $Acl

    Write-Host "[+] Honey Share créé : \\$env:COMPUTERNAME\$ShareName"
    Write-Host "[!] Audit activé — Events 5140/5145 déclencheront une alerte"
}

# Déploiement du honey share
New-HoneyShare `
    -ShareName "IT-Archive-2022" `
    -LocalPath "C:\HoneyShares\IT-Archive-2022" `
    -Description "IT Archive 2022 - Legacy Files"

# Placement de fichiers leurres convaincants
```

```
$HoneyFiles = @{
    "passwords_backup_2022.xlsx"     = "Fichier Excel vide avec macro alerte"
    "vpn_credentials.txt"            = "Credentials factices formatés"
    "admin_accounts_export.csv"      = "Export CSV avec faux comptes admin"
    "backup_script_credentials.ps1"  = "Script PS1 avec credentials factices"
    "IT-procedures-2022.docx"        = "Document Word avec canary token"
}

foreach ($file in $HoneyFiles.GetEnumerator()) {
    $Path = "C:\HoneyShares\IT-Archive-2022\$($file.Key)"
    # Créer un fichier minimal mais crédible
    [System.IO.File]::WriteAllText($Path, "# Archive IT 2022")
    Write-Host "[+] Fichier leurre créé : $($file.Key)"
}
```

Événements de détection sur Honey Shares

Event 5140 (Accès réseau à un partage) : un compte accède au honey share. Déclencher une alerte immédiate si le compte n'est pas dans la liste blanche des admins.

Event 5145 (Accès détaillé à un fichier) : accès à un fichier spécifique dans le honey share. Permet d'identifier quel fichier leurre a été ouvert.

Event 4663 (Tentative d'accès objet) : lecture ou copie d'un fichier leurre — couplé à 5145 pour une visibilité complète.

La corrélation de ces événements avec la détection d'attaques Active Directory avec Wazuh permet d'automatiser la réponse à incident.

Honey Tokens et Canary Tokens en Environnement Active Directory

Les honey tokens étendent le concept de leurre au-delà de l'annuaire Active Directory : ils s'insèrent dans les artefacts que les attaquants collectent après une compromission — fichiers de configuration, credential stores, scripts PowerShell, gestionnaires de mots de passe.

Canary Tokens : Types et Intégration AD

Canarytokens.org propose des tokens gratuits qui déclenchent une alerte (email, webhook) lorsqu'ils sont utilisés :

URL Token : URL factice insérée dans un fichier de documentation IT. Tout accès HTTP déclenche une alerte géolocalisée.

DNS Token : hostname factice dans un fichier hosts ou un script. Toute résolution DNS déclenche une alerte.

AWS Keys Token : clés AWS factices dans un fichier `.aws/credentials`. Toute tentative d'utilisation déclenche une alerte AWS + canary.

Word/PDF Token : document piégé qui contacte le serveur canary à l'ouverture.

Windows Directory Token : dossier Windows qui déclenche une alerte à l'accès.

```
# Placement stratégique de canary tokens dans l'environnement AD

# 1. Fichier credentials factice sur les postes d'administration
$CanaryContent = @"
[default]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
region = eu-west-3
# Backup credentials for ayinedjimi-prod
"@

# Déploiement via GPO ou script SCCM sur les postes admin
$TargetPath = "C:\Users\Administrator\.aws\credentials"
$CanaryContent | Out-File -FilePath $TargetPath -Encoding ASCII

# 2. URL canary dans un fichier de procédures IT
$ProcedureContent = @"
# Procédures IT Internes - CONFIDENTIEL
## Accès Serveur de Sauvegarde
URL admin : http://token.canarytokens.com/article/VOTRE_TOKEN_ICI/contact.php
Identifiants : voir coffre-fort IT
"@

# 3. Credential factice dans LSASS (via Mimikatz de test en environnement contrôlé)
# Cette technique injecte un credential fictif que Mimikatz collectera
# Les vrais credentials n'ont jamais ce format
$FakeCredential = @{
    Username = "svc_canary_monitor"
    Password = "HoneyP@ssw0rd-CanaryToken-2026!"
    Domain   = $env:USERDNSDOMAIN
}
```

OpenCanary : Infrastructure de Honeypots Réseau

OpenCanary (de Thinkst Canary, open-source) permet de déployer des services leurres complets : SMB, HTTP, SSH, FTP, MSSQL. Son intégration avec l'environnement AD crée des cibles convaincantes :

```
{
    "device.node_id": "honeypot-dc-paris-01",
    "git.enabled": false,
    "ftp.enabled": true,
    "ftp.port": 21,
    "http.enabled": true,
    "http.port": 80,
    "http.banner": "IIS/10.0 Windows Server 2019",
    "smb.enabled": true,
    "mssql.enabled": true,
    "mssql.port": 1433,
    "mssql.version": "2019",
    "logger": {
        "class": "PyLogger",
        "kwargs": {
            "formatters": {
                "plain": {"format": "%(message)s"}
            },
            "handlers": {
                "WEBHOOK": {
                    "class": "opencanary.logger.WebhookHandler",
                    "url": "https://hooks.slack.com/services/VOTRE_WEBHOOK",
                    "method": "POST"
                },
                "Syslog": {
                    "class": "logging.handlers.SysLogHandler",
                    "host": "127.0.0.1",
                    "port": 514,
                    "socktype": "SOCK_DGRAM"
                }
            }
        }
    }
}
```

```
# Installation OpenCanary (Ubuntu/Debian)
pip install opencanary
# Initialisation de la configuration
opencanaryd --copyconfig
# Démarrage
opencanaryd --start
# Vérification des logs
tail -f /var/log/opencanary.log
```

AdminSDHolder Honey : Détecter les Modifications de Permissions

AdminSDHolder est un objet AD spécial qui contrôle les permissions des groupes d'administration (Domain Admins, Enterprise Admins, etc.). Les attaquants qui ont compromis le domaine modifient souvent AdminSDHolder pour créer une backdoor de persistance — une technique documentée par l'ANSSI et détectée par les tests de red team les plus avancés.

Surveillance de l'AdminSDHolder

```
# Audit de l'AdminSDHolder - Configuration de la surveillance
function Enable-AdminSDHolderAudit {
    $AdminSDHolder = "CN=AdminSDHolder,CN=System,$((Get-ADDomain).DistinguishedName)"

    # Récupération de l'objet
    $Object = Get-ADObject $AdminSDHolder

    # Configuration de l'audit sur l'objet AdminSDHolder
    # Event 5136 = modification d'attribut d'objet AD
    # Event 4662 = opération sur l'objet
    $AuditRule = New-Object System.DirectoryServices.ActiveDirectoryAuditRule(
        [System.Security.Principal.SecurityIdentifier]"S-1-1-0",  # Everyone
        [System.DirectoryServices.ActiveDirectoryRights]"GenericAll",
        [System.Security.AccessControl.AuditFlags]"Success",
        [System.DirectoryServices.ActiveDirectorySecurityInheritance]"None"
    )

    $Acl = Get-Acl "AD:\$AdminSDHolder"
    $Acl.AddAuditRule($AuditRule)
    Set-Acl -Path "AD:\$AdminSDHolder" -AclObject $Acl

    Write-Host "[+] Audit activé sur AdminSDHolder"
    Write-Host "[!] Event 5136 sur CN=AdminSDHolder = tentative de backdoor persistante"
}

Enable-AdminSDHolderAudit

# Vérification des ACL actuelles de l'AdminSDHolder (baseline)
$AdminSDHolder = "CN=AdminSDHolder,CN=System,$((Get-ADDomain).DistinguishedName)"
Get-Acl "AD:\$AdminSDHolder" | Select-Object -ExpandProperty Access |
    Where-Object { $_.IdentityReference -notlike "BUILTIN*" -and
                   $_.IdentityReference -notlike "NT AUTHORITY*" } |
    Format-Table IdentityReference, ActiveDirectoryRights, AccessControlType
```

Honey Object dans AdminSDHolder

Une technique avancée consiste à ajouter une ACE (Access Control Entry) factice sur AdminSDHolder pour un honey user. Si un attaquant modifie AdminSDHolder pour ajouter ses propres permissions, l'ACE factice sera modifiée ou supprimée, déclenchant un Event 5136 sur le honey user :

```
# Ajout d'un honey user dans les ACL d'AdminSDHolder
# (permissions minimales - aucune élévation possible)
$HoneyUser = Get-ADUser "svc_backup_legacy"
$AdminSDHolder = "CN=AdminSDHolder,CN=System,$((Get-ADDomain).DistinguishedName)"

# Donner des droits Read-only au honey user sur AdminSDHolder
# Si ces droits sont modifiés → Event 5136 → alerte backdoor persistante
$Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule(
    $HoneyUser.SID,
    [System.DirectoryServices.ActiveDirectoryRights]"ReadProperty",
    [System.Security.AccessControl.AccessControlType]"Allow"
)

$Acl = Get-Acl "AD:\$AdminSDHolder"
$Acl.AddAccessRule($Rule)
Set-Acl -Path "AD:\$AdminSDHolder" -AclObject $Acl
Write-Host "[+] Honey ACE ajoutée sur AdminSDHolder pour $($HoneyUser.SamAccountName)"
```

Pour comprendre l'impact des modifications de permissions AD sur la sécurité globale, consultez notre guide sur les audits de groupes privilégiés Active Directory.

Déploiement avec Wazuh et Microsoft Sentinel

La valeur des honeypots AD dépend directement de la qualité de la détection en temps réel. Wazuh et Microsoft Sentinel offrent des intégrations natives avec les événements Windows pour automatiser les alertes.

Règles Wazuh pour Honey Users et Honey SPNs

Ces règles Wazuh XML déclenchent des alertes critiques (niveau 15) sur toute interaction avec les leurres AD. Elles complètent la configuration décrite dans notre guide Wazuh pour la détection Active Directory.

```
<!-- /var/ossec/etc/rules/honeypot_ad.xml -->
<group name="honeypot,active_directory,windows,">

  <!-- Règle 1 : Kerberoasting sur Honey SPN (Event 4769 + RC4) -->
  <rule id="110001" level="15">
    <if_group>windows</if_group>
    <id>4769</id>
    <field name="win.eventdata.serviceName">svc_backup_legacy|svc_legacy_erp|admin.reporting.svc<
    <field name="win.eventdata.ticketEncryptionType">0x17</field>
    <description>CRITIQUE: Kerberoasting détecté sur honey SPN - Compte: $(win.eventdata.targetUs
    <mitre>
      <id>T1558.003</id>
    </mitre>
    <group>kerberoasting,honeypot_trigger</group>
  </rule>

  <!-- Règle 2 : Tentative d'authentification sur Honey User (Event 4625) -->
  <rule id="110002" level="14">
    <if_group>windows</if_group>
    <id>4625</id>
    <field name="win.eventdata.targetUserName">svc_backup_legacy|svc_legacy_erp|admin.reporting.s
    <description>ALERTE: Tentative d'authentification sur honey user $(win.eventdata.targetUserNa
    <mitre>
      <id>T1110</id>
    </mitre>
    <group>password_spray,honeypot_trigger</group>
  </rule>

  <!-- Règle 3 : AS-REQ sur Honey User - AS-REP Roasting (Event 4768) -->
  <rule id="110003" level="13">
    <if_group>windows</if_group>
    <id>4768</id>
    <field name="win.eventdata.targetUserName">svc_backup_legacy|svc_legacy_erp</field>
    <description>ALERTE: AS-REQ Kerberos sur honey user $(win.eventdata.targetUserName) - possibl
    <mitre>
      <id>T1558.004</id>
    </mitre>
    <group>asrep_roasting,honeypot_trigger</group>
  </rule>

  <!-- Règle 4 : Accès au Honey Share (Event 5140) -->
  <rule id="110004" level="12">
```

```
    <if_group>windows</if_group>
    <id>5140</id>
    <field name="win.eventdata.shareName">IT-Archive-2022</field>
    <description>ALERTE: Accès au honey share IT-Archive-2022 par $(win.eventdata.subjectUserName
    <mitre>
      <id>T1135</id>
    </mitre>
    <group>lateral_movement,honeypot_trigger</group>
  </rule>

  <!-- Règle 5 : Modification AdminSDHolder (Event 5136) -->
  <rule id="110005" level="15">
    <if_group>windows</if_group>
    <id>5136</id>
    <field name="win.eventdata.objectDN">CN=AdminSDHolder</field>
    <description>CRITIQUE: Modification de l'AdminSDHolder détectée - Backdoor persistante potent
    <mitre>
      <id>T1003.003</id>
    </mitre>
    <group>persistence,honeypot_trigger</group>
  </rule>

</group>
```

Requêtes KQL Microsoft Sentinel

```
// ─── Détection Kerberoasting sur Honey SPNs ──────────────────────────────
let HoneyAccounts = datatable(Account:string) [
    "svc_backup_legacy", "svc_legacy_erp", "admin.reporting.svc"
];
SecurityEvent
| where EventID == 4769
| where TargetUserName in (HoneyAccounts)
| where TicketEncryptionType == "0x17"  // RC4 - indicateur Kerberoasting
| project
    TimeGenerated,
    Account          = TargetUserName,
    ServiceName      = ServiceName,
    SourceIP         = IpAddress,
    AttackerHost     = WorkstationName,
    EncryptionType   = TicketEncryptionType
| extend
    Severity    = "Critical",
    MITRE_ATT   = "T1558.003 - Kerberoasting",
    AlertTitle  = strcat("Kerberoasting détecté sur honey SPN : ", ServiceName)
| sort by TimeGenerated desc

// ─── Accès Honey Share avec corrélation utilisateur ──────────────────────
let HoneyShares = datatable(ShareName:string) ["IT-Archive-2022", "Admin-Tools-Legacy"];
SecurityEvent
| where EventID in (5140, 5145)
| extend ShareName = tostring(split(ShareName, "\\")[2])
| where ShareName in (HoneyShares)
| join kind=leftouter (
    SigninLogs
    | where TimeGenerated > ago(1h)
    | project UserId, UserPrincipalName, IPAddress, Location
) on $left.SubjectUserName == $right.UserId
| project
    TimeGenerated,
    User         = SubjectUserName,
    ShareName,
    AccessedFile = RelativeTargetName,
    SourceIP     = IpAddress,
    Location
| extend AlertTitle = strcat("Accès honey share par ", User)

// ─── Modification AdminSDHolder ──────────────────────────────────────────
```

```
SecurityEvent
| where EventID == 5136
| where ObjectDN contains "AdminSDHolder"
| project
    TimeGenerated,
    Modifier     = SubjectUserName,
    ObjectDN,
    AttributeChanged = AttributeLDAPDisplayName,
    OldValue     = OldValue,
    NewValue     = NewValue,
    SourceHost   = Computer
| extend
    Severity   = "Critical",
    AlertTitle = "Backdoor AdminSDHolder détectée",
    Response   = "Isoler le compte immédiatement et analyser l'arbre d'héritage ACL"
```

Automatisation et Orchestration des Honeypots AD

Un dispositif de honeypots efficace nécessite une maintenance régulière : rotation des leurres, mise à jour des listes de surveillance, vérification de l'intégrité. Ce script PowerShell orchestre le déploiement et la maintenance complète.

```
#Requires -Module ActiveDirectory
# Deploy-ADHoneypots.ps1 - Script de déploiement complet des honeypots AD
# Exécuter avec droits Domain Admin depuis un contrôleur de domaine

param(
    [string]$OUPath     = "OU=Service-Accounts,OU=IT,DC=domaine,DC=local",
    [string]$ShareServer = $env:COMPUTERNAME,
    [switch]$Rotate     # Rotation des leurres existants
)

# ─── Configuration ──────────────────────────────────────────────────────
$HoneyConfig = @{
    Users = @(
        @{ Sam="svc_backup_legacy";    SPN="HTTP/legacy-backup.domaine.local";      Dept="IT Oper
        @{ Sam="svc_legacy_erp";       SPN="MSSQLSvc/erp-db.domaine.local:1433";    Dept="Busines
        @{ Sam="admin.reporting.svc";  SPN="HTTP/reports.domaine.local:8080";       Dept="Finance
        @{ Sam="svc_monitor_agent";    SPN="WSMAN/monitoring.domaine.local";        Dept="Infrast
    )
    Shares = @(
        @{ Name="IT-Archive-2022";     Path="C:\HoneyShares\IT-Archive-2022" },
        @{ Name="Admin-Tools-Legacy";  Path="C:\HoneyShares\Admin-Tools-Legacy" }
    )
    HoneyFiles = @(
        "passwords_admin_backup.xlsx",
        "vpn_credentials_2024.txt",
        "domain_admin_accounts.csv",
        "deployment_script_withcreds.ps1",
        "ssh_keys_servers.txt"
    )
    WazuhRulesPath = "/var/ossec/etc/rules/honeypot_ad.xml"
    LogPath        = "C:\Windows\Logs\HoneyPot-Deploy-$(Get-Date -Format 'yyyyMMdd').log"
}

function Write-HoneyLog {
    param([string]$Message, [string]$Level = "INFO")
    $Entry = "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] [$Level] $Message"
    Add-Content -Path $HoneyConfig.LogPath -Value $Entry
    Write-Host $Entry -ForegroundColor $(if ($Level -eq "ERROR") {"Red"} elseif ($Level -eq "WARN
}

function Deploy-HoneyUsers {
    Write-HoneyLog "Déploiement des honey users..."
```

```
    foreach ($user in $HoneyConfig.Users) {
        try {
            $existing = Get-ADUser $user.Sam -ErrorAction SilentlyContinue
            if ($existing -and -not $Rotate) {
                Write-HoneyLog "Honey user $($user.Sam) existe déjà — ignorer (utiliser -Rotate p
                continue
            }

            $Password = [System.Web.Security.Membership]::GeneratePassword(64, 12)
            $SecurePass = ConvertTo-SecureString $Password -AsPlainText -Force

            if (-not $existing) {
                New-ADUser `
                    -Name $user.Sam `
                    -SamAccountName $user.Sam `
                    -UserPrincipalName "$($user.Sam)@$env:USERDNSDOMAIN" `
                    -Department $user.Dept `
                    -Description "Service account - $(($user.Dept)) integration" `
                    -Path $OUPath `
                    -AccountPassword $SecurePass `
                    -Enabled $true `
                    -PasswordNeverExpires $true `
                    -CannotChangePassword $true
            }

            # SPN
            setspn -A $user.SPN "$env:USERDOMAIN\$($user.Sam)" 2>$null
            # LastLogon plausible (540 jours)
            $oldDate = (Get-Date).AddDays(-540).ToFileTime()
            Set-ADUser -Identity $user.Sam -Replace @{lastLogonTimestamp=$oldDate; pwdLastSet=(Ge

            Write-HoneyLog "Honey user déployé : $($user.Sam) avec SPN $($user.SPN)"
        } catch {
            Write-HoneyLog "Erreur déploiement $($user.Sam) : $_" "ERROR"
        }
    }
}

function Deploy-HoneyShares {
    Write-HoneyLog "Déploiement des honey shares..."

    foreach ($share in $HoneyConfig.Shares) {
        try {
```

```
            New-Item -ItemType Directory -Path $share.Path -Force | Out-Null
            $existing = Get-SmbShare -Name $share.Name -ErrorAction SilentlyContinue
            if (-not $existing) {
                New-SmbShare -Name $share.Name -Path $share.Path -ReadAccess "Domain Users" -Full
            }

            # Fichiers leurres
            foreach ($file in $HoneyConfig.HoneyFiles) {
                $filePath = Join-Path $share.Path $file
                if (-not (Test-Path $filePath)) {
                    "[CONFIDENTIEL] Ce document est la propriété de l'équipe IT." | Out-File $fil
                }
            }

            # Audit SACL
            $acl = Get-Acl $share.Path
            $auditRule = New-Object System.Security.AccessControl.FileSystemAuditRule(
                "Everyone", "ReadData,WriteData,Delete",
                "ContainerInherit,ObjectInherit", "None", "Success,Failure"
            )
            $acl.AddAuditRule($auditRule)
            Set-Acl -Path $share.Path -AclObject $acl

            Write-HoneyLog "Honey share déployé : \\$ShareServer\$($share.Name)"
        } catch {
            Write-HoneyLog "Erreur déploiement share $($share.Name) : $_" "ERROR"
        }
    }
}

function Invoke-HoneyRotation {
    Write-HoneyLog "Rotation des leurres en cours..."
    # Changer les mots de passe (toujours aléatoires, jamais utilisés)
    foreach ($user in $HoneyConfig.Users) {
        $NewPass = [System.Web.Security.Membership]::GeneratePassword(64, 12)
        $SecurePass = ConvertTo-SecureString $NewPass -AsPlainText -Force
        Set-ADAccountPassword -Identity $user.Sam -NewPassword $SecurePass -Reset -ErrorAction Si
        Write-HoneyLog "Mot de passe rotaté pour $($user.Sam)"
    }
    # Mettre à jour les timestamps pour rester crédibles
    foreach ($user in $HoneyConfig.Users) {
        $oldDate = (Get-Date).AddDays(-$(Get-Random -Min 180 -Max 720)).ToFileTime()
        Set-ADUser -Identity $user.Sam -Replace @{lastLogonTimestamp=$oldDate} -ErrorAction Silen
    }
```

```
}

# ── Exécution principale ──────────────────────────────────────────────
Write-HoneyLog "=== Démarrage déploiement honeypots AD ==="
Deploy-HoneyUsers
Deploy-HoneyShares
if ($Rotate) { Invoke-HoneyRotation }
Write-HoneyLog "=== Déploiement terminé ==="
Write-HoneyLog "Log : $($HoneyConfig.LogPath)"
```

Planning de Maintenance Recommandé

Mensuel : rotation des mots de passe des honey users (script `-Rotate`)

Trimestriel : ajout de nouveaux honey SPNs, remplacement des fichiers leurres

Semestriel : audit des ACL honey shares, vérification de la cohérence des timestamps

Continu : monitoring des alertes Wazuh/Sentinel, révision des règles de détection

La configuration Sysmon sur les contrôleurs de domaine documentée dans notre guide Sysmon 2026 pour contrôleurs de domaine complète idéalement ce dispositif de honeypots.

Conformité NIS 2 : Les Honeypots comme Mesure de Détection Proactive

La directive NIS 2 (Network and Information Security), transposée en droit français par la loi du 15 octobre 2024 et dont l'application est supervisée par l'ANSSI, impose aux Entités Essentielles et Importantes des mesures de sécurité proportionnées aux risques. Les honeypots Active Directory s'inscrivent directement dans ce cadre réglementaire.

Ancrage Réglementaire

Article 21, alinéa 2(b) NIS 2 — "gestion des incidents" : Les honeypots améliorent la capacité de détection précoce des incidents, réduisant le MTTD (Mean Time To Detect). Un incident

détecté par un honey user en quelques minutes contre plusieurs jours pour une détection EDR classique transforme radicalement la capacité de notification dans les 72 heures.

Article 21, alinéa 2(e) NIS 2 — "sécurité dans l'acquisition, le développement et la maintenance des réseaux et des systèmes d'information" : Les leurres AD constituent une mesure de sécurité intégrée à l'infrastructure Active Directory, justifiable dans le cadre d'un plan de sécurité réseau.

CERT-FR / ANSSI R37 : La recommandation explicite d'utiliser des leurres dans les environnements AD critiques figure dans le guide de l'ANSSI "Recommandations de sécurité relatives à Active Directory". Les honeypots sont catégorisés comme mesure de détection complémentaire aux journaux d'audit.

Contexte Opérationnel Français

Pour les RSSI et DSI des entreprises françaises, particulièrement celles ayant leur siège à Paris ou dans les grandes métropoles régionales, le déploiement de honeypots AD présente plusieurs avantages conformité :

Preuve de diligence : en cas d'audit ANSSI ou de contrôle NIS 2, l'existence d'un dispositif de détection avancé démontre une approche proactive.

Notification d'incident facilitée : les événements Windows générés par les honeypots constituent des preuves horodatées exploitables immédiatement dans une notification ANSSI.

Forensique post-incident : les logs de honey users tracent précisément le périmètre de la reconnaissance effectuée par l'attaquant, valeur inestimable pour l'analyse post-mortem.

ROI mesurable : réduction documentable du MTTD, argument budget pour les RSSI soumis à des contraintes financières.

La mise en place d'Authentication Policies et Authentication Silos, documentée dans notre guide Authentication Policies Windows Server 2025, complète les honeypots en isolant les comptes privilégiés et en limitant la surface d'attaque disponible pour les attaquants qui auraient contourné les leurres.

Métriques de Conformité

Métrique	Sans Honeypots	Avec Honeypots AD
MTTD (Kerberoasting)	72-120 heures	< 5 minutes
Faux positifs par semaine	50-200 (EDR)	0 (honeypot)
Couverture détection latéralisation	30-50%	80-95%
Coût déploiement initial	—	4-8 heures ingénieur
Maintenance mensuelle	—	30 minutes

Questions fréquentes sur les Honeypots Active Directory

Qu'est-ce qu'un honey user en Active Directory et pourquoi en créer ?

Un honey user (ou compte appât) est un compte Active Directory délibérément créé sans utilisation légitime, dont l'unique but est d'attirer les attaquants. Toute tentative d'authentification, d'énumération Kerberos ou de Kerberoasting sur ce compte déclenche immédiatement une alerte. Contrairement aux comptes de service réels, un honey user ne génère jamais de trafic légitime, ce qui rend les faux positifs quasi-nuls. C'est l'une des techniques de détection à haute fidélité les moins coûteuses à déployer en entreprise.

Comment un honey SPN permet-il de détecter le Kerberoasting ?

Le Kerberoasting consiste à demander des tickets de service Kerberos (TGS-REP, Event ID 4769) pour des comptes avec SPN afin de les craquer hors ligne. En créant un SPN factice sur un compte inactif jamais utilisé en production, toute demande de ticket pour ce SPN est nécessairement malveillante. La surveillance de l'Event 4769 avec un chiffrement RC4 (etype 23) sur ce compte déclenche une alerte critique sans aucun faux positif.

Quelle est la différence entre un canary token et un honey token en AD ?

Les termes sont souvent utilisés de façon interchangeable, mais un canary token désigne généralement un artefact numérique traçable (URL, fichier Word, clé AWS) qui contacte un serveur de suivi lorsqu'il est ouvert ou utilisé. Un honey token en AD est plus large : il peut s'agir de credentials factices stockés en mémoire LSASS, dans un fichier de configuration ou dans un gestionnaire de mots de passe. Les deux se complètent : les canary tokens détectent l'ouverture de fichiers, les honey tokens détectent l'utilisation d'identifiants volés.

Les honeypots AD sont-ils compatibles avec les exigences NIS 2 ?

Oui. L'article 21 de la directive NIS 2 impose des mesures de détection d'incidents proportionnées aux risques. Les honeypots AD constituent une mesure de détection proactive explicitement valorisée par l'ANSSI dans son guide de durcissement Active Directory (CERT-FR). Ils améliorent la capacité de détection précoce (early detection), réduisent le MTTD (Mean Time To Detect) et génèrent des preuves forensiques précieuses pour la notification d'incidents dans les 72 heures imposées par NIS 2.

Comment éviter que les honey users soient détectés et contournés par un attaquant sophistiqué ?

La crédibilité des leurres est essentielle. Un honey user doit avoir : un nom réaliste (prénom.nom ou svc_xxx), une description convaincante, une date de création ancienne, un historique de password change plausible, et apparaître dans des groupes AD non-privilégiés. Il ne doit jamais avoir de LastLogon récent (ce qui trahirait une activité automatisée). La rotation périodique des leurres, la diversification des types (users, SPNs, shares, tokens) et l'absence de motif prévisible dans les noms renforcent leur résistance à la détection adverse.

À RETENIR

À retenir

Les honey users et honey SPNs offrent une détection du Kerberoasting et du password spray avec un taux de faux positifs de 0% — toute interaction est malveillante par construction.

La surveillance des Event IDs 4769 (RC4 etype 0x17), 4768, 4625, 5136 et 5140 couvre l'ensemble des phases d'attaque Active Directory : reconnaissance, Kerberoasting, password spray, latéralisation et persistance.

Les canary tokens (URL, AWS keys, fichiers Word) complètent les honeypots AD en détectant l'utilisation de credentials et documents exfiltrés, même hors du réseau de l'entreprise.

Le script PowerShell de déploiement permet de créer un dispositif complet (honey users, SPNs, shares, fichiers) en moins d'une heure, avec une maintenance mensuelle de 30 minutes.

Les honeypots AD s'inscrivent directement dans le cadre NIS 2 (art. 21) et les recommandations ANSSI/CERT-FR, réduisant le MTTD de 72+ heures à moins de 5 minutes pour les attaques Kerberos.

Références et documentation

Microsoft Learn — Sécurité Windows Server

ANSSI — Guide de sécurisation Active Directory

MITRE ATT&CK — Techniques Active Directory

