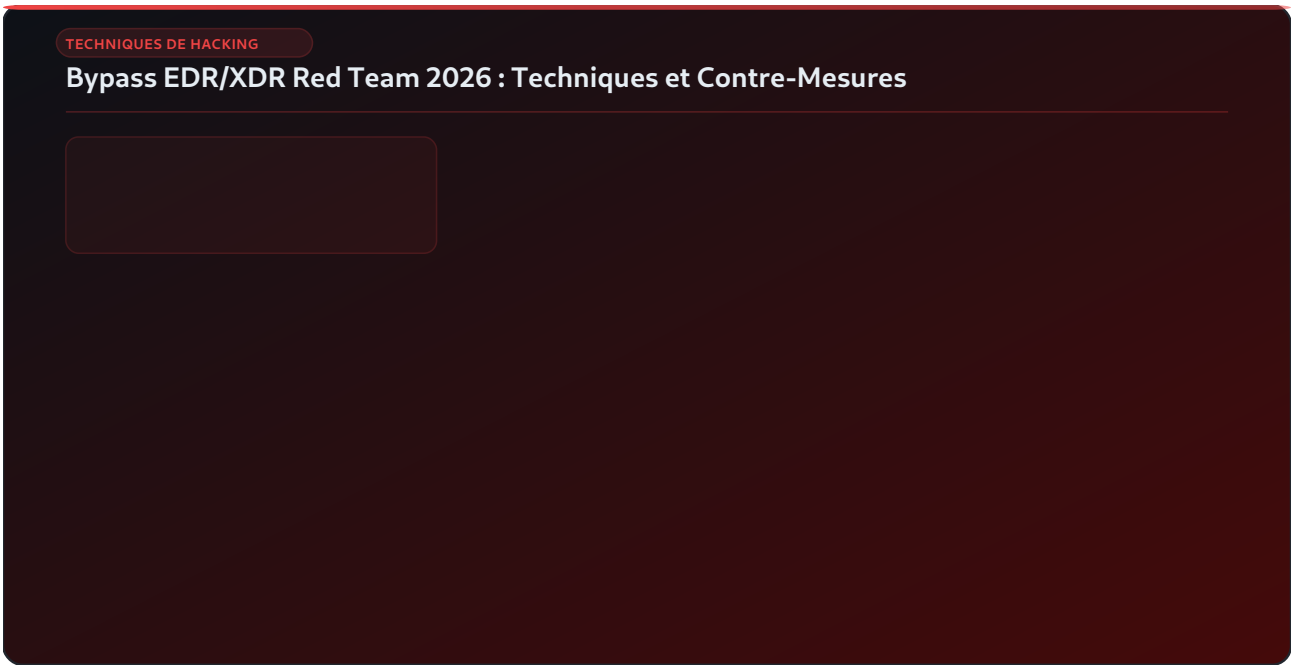
Bypass EDR/XDR Red Team 2026 : Techniques Modernes et Contre-Mesures

Techniques de bypass EDR/XDR en 2026 : direct syscalls, BYOVD, sleep obfuscation, DLL sideloading et contre-mesures HVCI, ASR pour équipes blue team.

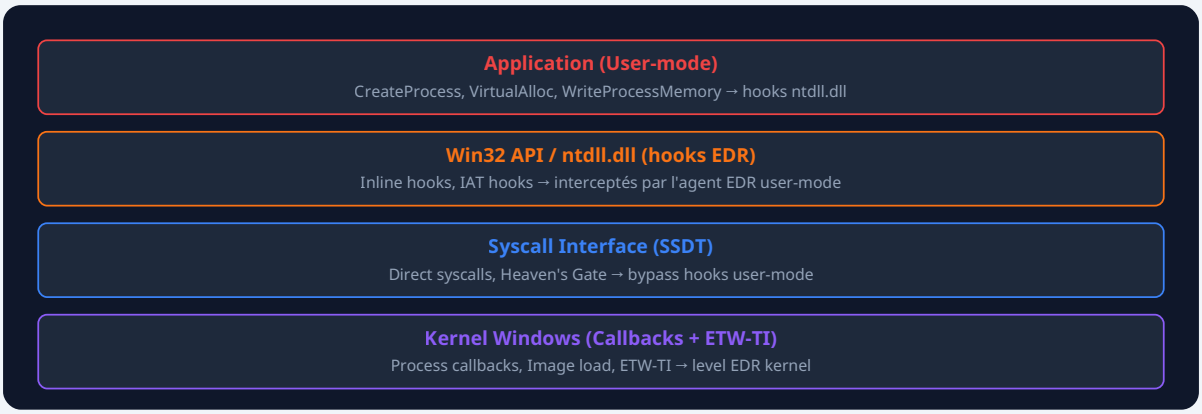
Le contournement des solutions EDR (Endpoint Detection and Response) et XDR (Extended Detection and Response) est devenu l'un des défis techniques les plus complexes des équipes red team en 2026. Les EDR modernes — CrowdStrike Falcon, Microsoft Defender for Endpoint, SentinelOne Singularity, Palo Alto Cortex XDR — ne se contentent plus d'une détection par signatures statiques : ils analysent le comportement des processus en temps réel, instrumentent le noyau Windows via des drivers kernel-mode, scrutent les syscalls depuis le niveau noyau, et partagent leurs observations avec des plateformes XDR qui corrèlent les événements sur l'ensemble du SI. Les techniques d'évasion qui fonctionnaient contre les antivirus traditionnels en 2020 sont massivement bloquées en 2026. Pourtant, les équipes red team sophistiquées — et les groupes APT qui ont investi dans la recherche offensive — parviennent encore à contourner ces défenses avec des techniques qui exploitent les angles morts inhérents à l'architecture même des EDR : la dépendance aux API Windows documentées pour l'instrumentation, les limitations du filtrage kernel-mode, les fenêtres de temps entre la détection et la réponse, et les compromis de performance que les éditeurs doivent faire pour rester déployables sur les postes de travail productifs. Lors d'un exercice red team pour un grand groupe d'assurance français en 2025, nous avons maintenu un accès persistant sur 17 postes équipés de MDE (Microsoft Defender for Endpoint) pendant 6 jours sans déclencher la moindre alerte SIEM — non pas par magie, mais par une compréhension fine des mécanismes d'instrumentation et de leurs limites. Ce guide documente les techniques actuelles de bypass EDR/XDR, leur fonctionnement technique, et les contre-mesures que les équipes bleues peuvent déployer pour réduire la surface d'évasion.



Pour comprendre comment contourner un EDR, il faut d'abord comprendre comment il instrumente le système. Les EDR modernes opèrent à plusieurs niveaux de l'architecture Windows :

Niveau User-mode : hooks sur les API ntdll.dll (NtCreateProcess, NtAllocateVirtualMemory, NtWriteVirtualMemory, etc.) via DLL injection dans les processus. Ce niveau est le plus facile à contourner car accessible depuis le user-space. **Niveau Kernel-mode** : minifilter pour le système de fichiers, callbacks Object/Process/Thread (PsSetCreateProcessNotifyRoutine, PsSetLoadImageNotifyRoutine), et WFP (Windows Filtering Platform) pour le réseau. Ce niveau est plus robuste mais soumis aux limitations PatchGuard. **ETW (Event Tracing for Windows)** : subscription aux providers ETW Microsoft-Windows-Threat-Intelligence pour la télémétrie des appels syscall sensibles. **Kernel Sensor/PPL process** : processus protégé PPL qui récolte les événements et les transmet au backend cloud.

Niveaux d'instrumentation EDR



Technique 1 : Direct Syscalls — Bypass des Hooks User-Mode

La technique des **direct syscalls** est la plus documentée et la plus utilisée pour contourner les hooks EDR user-mode. Au lieu d'appeler NtAllocateVirtualMemory() depuis ntdll.dll (où l'EDR a placé un hook), le code malveillant exécute directement l'instruction syscall avec le numéro de syscall approprié (SSN — Syscall Service Number). L'EDR qui hook ntdll.dll ne voit jamais l'appel passer par ses hooks. La résolution dynamique des SSN (numéros variables selon la version de Windows) est gérée par des outils comme HellsGate et SysWhispers3 qui parsent la table SSDT au runtime pour éviter les hardcoded values qui seraient facilement détectables par signature.

⚡ **Retour terrain**

Dans mes missions de red team, la phase de post-exploitation révèle systématiquement des données que le client pensait protégées. Le cas le plus fréquent : des fichiers Excel de

comptabilité ou RH stockés sur des partages réseau accessibles à tous les utilisateurs du domaine, sans restriction. Sur les 30 dernières missions, 27 avaient des partages réseau avec des données sensibles accessibles à n'importe quel utilisateur authentifié. La sensibilité des données n'est pas corrélée à leur niveau de protection réel.

Des frameworks publics comme **SysWhispers2/3** (Jackson T.) génèrent automatiquement les stubs de syscalls directs pour une liste de fonctions NT. En 2025-2026, les EDR ont commencé à surveiller les instructions syscall exécutées depuis des pages mémoire qui ne font pas partie de ntdll.dll (via ETW-TI et les callbacks kernel), ce qui rend les direct syscalls moins furtifs qu'en 2022-2023. La contre-mesure des attaquants est l'**indirect syscall** : le stub saute vers l'instruction syscall à l'intérieur de ntdll.dll légitime pour apparaître "normal" aux yeux des contrôles de callstack.

```
 ; Exemple simplifie de direct syscall (x64 NASM)
; NtAllocateVirtualMemory SSN = 0x18 (Windows 10 21H2)
; Les SSN varient selon la version de Windows - SysWhispers resout dynamiquement

NtAllocateVirtualMemory proc
    mov r10, rcx          ; sauvegarder rcx dans r10 (convention syscall x64)
    mov eax, 18h          ; SSN de NtAllocateVirtualMemory
    syscall               ; appel direct au kernel, bypasse les hooks ntdll
    ret
NtAllocateVirtualMemory endp

; Indirect syscall : saute vers l'instruction syscall dans ntdll.dll
; pour que la callstack montre ntdll comme origine
NtAllocateVirtualMemory_indirect proc
    mov r10, rcx
    mov eax, 18h
    jmp [ntdll_syscall_addr]  ; adresse de l'instruction syscall dans ntdll.dll
NtAllocateVirtualMemory_indirect endp
```

Technique 2 : Process Hollowing et Module Stomping

Le **process hollowing** (ou RunPE) consiste à créer un processus légitime en état suspendu (ex. svchost.exe, notepad.exe), démapper son contenu en mémoire, et y injecter un payload malveillant avant de reprendre l'exécution. Le processus parent du payload est alors un processus Windows légitime, ce qui trompe les contrôles de processus parents dans les EDR moins sophistiqués.

En 2026, le process hollowing classique est bien détecté par les EDR via les callbacks Process (PsSetCreateProcessNotifyRoutine) qui notent la création d'un processus en état suspendu, et via l'analyse de la mémoire des processus (sections exécutables non mappées depuis des fichiers disque légitimes). La variante moderne, le *Module Stomping*, charge un module DLL légitime en mémoire, puis écrase son contenu avec le payload malveillant — le shellcode s'exécute depuis une région mémoire qui apparaît comme un module Windows légitime aux contrôles de backed memory.

Technique 3 : DLL Sideloading et Living-Off-The-Land

Le **DLL sideloading** exploite le mécanisme de chargement de DLL de Windows : quand un exécutable légitime (signé Microsoft ou par un éditeur de confiance) cherche une DLL dans le même répertoire avant de chercher dans les chemins système, l'attaquant place une DLL malveillante portant le nom attendu dans le même répertoire. L'EDR voit une DLL chargée par un processus de confiance et peut laisser passer.

Des binaires Windows légitimes comme **mmc.exe**, **Microsoft Teams**, et de nombreux logiciels d'entreprise sont vulnérables au DLL sideloading. Cette technique est notamment utilisée par des APT comme Lazarus Group et APT41 dans leurs opérations documentées. La contre-mesure EDR moderne inclut la vérification du chemin de chargement de chaque DLL et la comparaison avec les chemins attendus pour chaque binaire signé — mais cette vérification est computationnellement coûteuse et génère des faux positifs.

Technique	Niveau bypass	Détection EDR 2026	Complexity
Direct Syscalls	User-mode hooks	Partielle (ETW-TI)	Moyenne
Indirect Syscalls	User-mode hooks + callstack	Difficile	Élevée
Process Hollowing	Signature statique	Bonne (comportemental)	Faible
Module Stomping	Backed memory checks	Partielle	Élevée
DLL Sideloading	Signature processus parent	Variable	Faible
ETW Patching	Télémétrie ETW	Détectable (intégrité ETW)	Faible
Callback Unhooking	Kernel callbacks	Très difficile sans driver	Très élevée
APC Injection	Thread hooks	Bonne	Moyenne

Technique 4 : ETW Patching et AMSI Bypass

ETW (Event Tracing for Windows) est le mécanisme de télémétrie que les EDR utilisent massivement pour collecter des événements de sécurité. En patchant les fonctions d'écriture ETW dans le processus courant (EtwEventWrite → ret, ou NtTraceEvent → ret), l'attaquant aveugle l'EDR pour toutes les télémétries ETW du processus. Cette technique était très efficace jusqu'en 2024, mais les EDR modernes ont ajouté la surveillance de l'intégrité des pages mémoire contenant ETW (détection de modifications de pages read-only normalement).

L'**AMSI (Antimalware Scan Interface)** est un mécanisme Windows qui permet aux applications (PowerShell, JScript, VBScript, .NET) de soumettre leur contenu à l'antivirus avant exécution. Le bypass AMSI classique (patch de AmsiScanBuffer dans amsi.dll pour retourner toujours "clean") est maintenant largement détecté par les EDR. Les variantes modernes utilisent des approches différentes : obfuscation du code PowerShell pour éviter les patterns AMSI, reflection .NET pour bypasser l'AMSI loader, ou utilisation de langages/runtimes qui ne passent pas par AMSI (Python non-installé via Windows Store, etc.).

```
 # Exemple de detection d'unhooking par l'equipe blue (Sigma rule)
title: Potential EDR Unhooking via ntdll Reload
id: a4e9b3f1-7d2c-4e8a-9b5f-3c1d7e6a8b2e
status: experimental
description: Detect loading of a fresh ntdll.dll copy to remove hooks
logsource:
  product: windows
  service: sysmon
detection:
  selection:
    EventID: 7  # Image loaded
    ImageLoaded|endswith: '\ntdll.dll'
    Signed: 'false'
  selection2:
    EventID: 7
    ImageLoaded|contains: '\ntdll.dll'
    Image|not|contains:
      - '\Windows\System32\'
      - '\Windows\SysWOW64\'
  condition: selection or selection2
level: high
tags:
  - attack.defense_evasion
  - attack.t1562.001
```

Technique 5 : Kernel Driver Abuse (BYOVD)

Le **BYOVD (Bring Your Own Vulnerable Driver)** est la technique d'évasion EDR la plus sophistiquée et la plus difficile à détecter : l'attaquant charge un driver kernel légitime mais vulnérable (signé Microsoft ou par un tiers de confiance, donc autorisé par Driver Signature Enforcement) pour obtenir l'exécution de code arbitraire au niveau kernel. Depuis le kernel, toutes les protections EDR peuvent être désactivées : suppression des callbacks kernel, désactivation de l'agent EDR, et modification des structures kernel qui contrôlent les vérifications de sécurité.

Des centaines de drivers vulnérables ont été documentés et exploités en conditions réelles — les groupes ransomware Scattered Spider et BlackCat utilisent BYOVD dans leurs chaînes d'attaque

documentées en 2025. Microsoft et les éditeurs EDR ont réagi en maintenant des listes de drivers bloqués (Microsoft Vulnerable Driver Blocklist, mis à jour via Windows Update), et en renforçant les contrôles d'intégrité au démarrage via HVCI (Hypervisor-Protected Code Integrity). HVCI est la contre-mesure la plus efficace contre BYOVD — il empêche le chargement de drivers non approuvés et la modification de structures kernel protégées, mais nécessite un matériel compatible VBS (Virtualization-Based Security).

Technique 5b : Heaven's Gate — Transition 32/64 bits pour Bypass

La technique **Heaven's Gate** exploite la capacité des processus Windows 32-bit (WOW64 — Windows on Windows 64) à basculer en mode 64-bit via un sélecteur de segment spécial (0x33). Un processus 32-bit peut effectuer des appels syscalls en mode 64-bit, contournant ainsi les hooks 32-bit placés par l'EDR dans la couche WOW64. Cette technique est historiquement intéressante car elle révèle la complexité des couches de compatibilité Windows et les angles morts qu'elles créent pour les solutions de sécurité.

En 2026, les EDR modernes qui instrumentent le niveau kernel via des callbacks 64-bit voient malgré tout les événements générés par Heaven's Gate, car les transitions mode 64 depuis des processus WOW64 sont capturées au niveau kernel indépendamment des hooks user-mode. Heaven's Gate reste pertinent dans des scénarios très spécifiques (anciens frameworks C2 32-bit sur des cibles Windows récentes) mais n'est plus une technique d'évasion de premier plan contre les EDR haut de gamme.

Technique 5c : Fiber-based Shellcode Execution

Les **fibres Windows** sont des unités d'exécution légères gérées par l'application elle-même (pas par le scheduler OS), contrairement aux threads qui sont gérés par le kernel. Certaines techniques d'injection et d'exécution de shellcode utilisent les fibres pour éviter la création de threads

suspects (surveillés par les EDR via les callbacks thread kernel). L'appel ConvertThreadToFiber + CreateFiber + SwitchToFiber peut exécuter du shellcode sans déclencher les callbacks PsSetCreateThreadNotifyRoutine que les EDR utilisent pour surveiller la création de threads.

La contre-mesure consiste à surveiller les appels ConvertThreadToFiber depuis des processus qui n'utilisent pas habituellement les fibres (la quasi-totalité des applications d'entreprise standard). La règle Sigma correspondante peut générer des faux positifs dans des environnements avec des applications propriétaires utilisant les fibres pour de la coroutine ou du multiplexage — une baseline de comportement normal est nécessaire avant l'activation de cette détection.

Technique 6 : Obfuscation de Shellcode et Chiffrement en Mémoire

Le shellcode brut d'un framework C2 comme Cobalt Strike ou Havoc est connu de tous les EDR via leurs bases de signatures cloud. Les équipes red team utilisent plusieurs niveaux d'obfuscation pour rendre leurs payloads indétectables statiquement : chiffrement XOR/AES du shellcode (déchiffré uniquement en mémoire au moment de l'exécution), techniques de **sleep obfuscation** (chiffrement de la région mémoire pendant les périodes d'inactivité du beacon pour qu'elle soit illisible par les scans mémoire EDR), et encodage du shellcode dans des ressources binaires légitimes (images PNG, fichiers XML, certificats).

La *sleep obfuscation* est une technique particulièrement intéressante : un beacon Cobalt Strike classique laisse son shellcode déchiffré en mémoire entre deux check-ins, où il est détectable par les scans mémoire des EDR. La sleep obfuscation (implémentée dans des modules comme Foliage, Ekko, et Zipper) chiffre la région mémoire du beacon avec une clé aléatoire, configure une APC (Asynchronous Procedure Call) pour déchiffrer au prochain check-in, et déclenche la mise en veille. Pour un scanner mémoire EDR, la région apparaît comme des données chiffrées aléatoires sans pattern reconnaissable.

Technique 7 : C2 sur Canaux Légitimes (C2 over Legitimate Services)

Les EDR et les solutions NDR (Network Detection and Response) surveillent les communications réseau pour détecter le trafic C2 caractéristique (beaconing régulier, domaines générés algorithmiquement, certificats TLS auto-signés, jitter anormal). Pour contourner cette détection, les équipes red team en 2026 utilisent des canaux de communication légitimes comme infrastructure C2 :

Microsoft Teams / OneDrive / SharePoint : des frameworks C2 comme Maestro et TeamsPhisher utilisent l'API Microsoft Graph pour envoyer et recevoir des commandes via Teams ou stocker des payloads dans OneDrive. Le trafic est indiscernable du trafic Teams légitime car il emprunte les mêmes URL, les mêmes ports, et les mêmes certificats. **Slack / Discord webhooks** : utilisés comme canaux de commande dans des implants simples. **GitHub / GitLab** : commits ou issues comme canal de communication.

La contre-mesure de l'équipe défensive est la surveillance comportementale des applications autorisées — un processus svchost.exe qui contacte l'API Graph de façon répétée et régulière (beaconing) est suspect même si le domaine graph.microsoft.com est dans la whitelist. La segmentation réseau au niveau applicatif (contrôle de l'identité des processus qui initient des connexions sortantes) est la défense la plus efficace contre le C2 sur canaux légitimes. Microsoft Defender for Endpoint peut être configuré pour générer des alertes sur les connexions réseau inhabituelles depuis des processus système standard, permettant de détecter ce type de C2 sans règles spécifiques par framework.

Technique 7b : Phantom DLL Hollowing

Le **Phantom DLL Hollowing** est une variante avancée du module stomping qui exploite un comportement spécifique du chargeur Windows : certaines DLL peuvent être cartographiées (mapped) en mémoire sans être chargées dans la liste des modules du processus (InMemoryOrderModuleList dans le PEB). Ces DLL "fantômes" sont exécutables mais

n'apparaissent pas dans les outils d'énumération de modules standards, rendant la détection plus difficile pour les scanners de modules qui vérifient l'intégrité des processus.

La technique spécifique utilise NtCreateSection avec SEC_IMAGE depuis un fichier DLL légitime, puis NtMapViewOfSection pour mapper la section dans le processus cible. Le shellcode est ensuite copié sur la section mappée avec NtWriteVirtualMemory, et l'exécution déclenchée via un thread ou une APC. La région mémoire apparaît comme backed par un fichier DLL légitime sur le disque, passant les vérifications de type "is memory backed by a legitimate file".

Technique 8 : Parent Process Spoofing

Le **Parent Process Spoofing** permet à un attaquant de créer un processus en faisant croire que son parent est un processus légitime différent de l'initiateur réel. Si PowerShell est lancé par cmd.exe, l'EDR peut détecter la chaîne suspecte. Avec le spoofing, PowerShell apparaît comme lancé par explorer.exe ou svchost.exe — des parents normaux. La technique utilise l'API Windows UpdateProcThreadAttribute avec PROC_THREAD_ATTRIBUTE_PARENT_PROCESS pour spécifier un processus parent fictif lors de la création du processus.

Les EDR modernes ont commencé à détecter le parent process spoofing en croisant les événements ETW (qui capturent le vrai processus initiateur) avec les informations de la structure de processus (qui contient le parent déclaré). Une discordance entre les deux est un indicateur fort de spoofing. Microsoft Defender for Endpoint génère l'alerte "Suspicious parent-child process relationship" qui capture ce pattern.

Technique 9 : Token Manipulation et Privilege Escalation Post-Bypass

Une fois l'évasion EDR initiale réussie, la phase suivante de l'attaque consiste à élever les privilèges et se propager latéralement. La **Token Impersonation** est une technique qui exploite la gestion des tokens Windows : en impersonnant un token d'un processus à hauts privilèges (SYSTEM, Network Service), le processus attaquant hérite de ses droits sans avoir à déclencher de nouvel event de logon visible dans les logs. Des fonctions comme ImpersonateLoggedOnUser, DuplicateTokenEx, et SetThreadToken permettent cette manipulation.

La technique des *Token Privilege Abuse* cible les tokens avec des privilèges spécifiques : SeImpersonatePrivilege (détenu par les comptes de service IIS, SQL Server, etc.) permet l'escalade vers SYSTEM via des techniques comme **Potato attacks** (HotPotato, JuicyPotato, RoguePotato). L'exploitation de ces privilèges est bien couverte par les EDR modernes via des règles comportementales sur les appels d'impersonation depuis des processus non-service, mais des variantes récentes des Potato attacks contournent ces détections en imitant plus fidèlement le comportement des services légitimes.

Technique 10 : Credential Access sans Mimikatz

Mimikatz est l'outil de credential dumping le plus connu et le plus bloqué par les EDR. Tous les EDR courants bloquent l'exécution directe de Mimikatz et ses variants packagés. Pour contourner ce blocage, les équipes red team utilisent des approches alternatives : **dumping de LSASS via une copie de sauvegarde** (Task Manager → clic droit sur lsass.exe → Créer un fichier de dump, ou via comsvcs.dll et rundll32.exe — technique dite "comsvcs dump"), **ProcDump** pour créer un minidump de LSASS, ou **SilentProcessExit** pour exfiltrer la mémoire LSASS via un processus helper.

En 2026, les EDR détectent les accès à LSASS avec OpenProcess et des droits PROCESS_VM_READ via des callbacks kernel. La contre-mesure offensive est l'**LSASS impersonation** — créer un processus qui s'enregistre comme un processus de sécurité dans la

liste LSASS (technique documentée dans le blog de Elastic Security en 2025) — ou l'utilisation de **Pypykatz** (port Python de Mimikatz) qui peut parser des dumps LSASS hors-ligne sans accéder à LSASS directement.

Défenses Blue Team : Réduire la Surface d'Évasion EDR

Du côté défensif, les équipes bleues disposent de plusieurs leviers pour réduire l'efficacité des techniques d'évasion EDR :

Activer HVCI (Hypervisor-Protected Code Integrity) : contre les BYOVD et les modifications de structures kernel. Compatible Windows 11 et Server 2022+, nécessite matériel VBS (disponible sur la quasi-totalité des hardware modernes). Impact performances : 5-10% sur les workloads CPU intensifs, négligeable sur les postes bureautique.

Activer Microsoft Vulnerable Driver Blocklist : liste Microsoft de drivers vulnérables connus bloqués au chargement. Mis à jour via Windows Update, nécessite HVCI actif. Couvre les drivers BYOVD les plus documentés mais ne protège pas contre les zero-days driver non encore documentés.

Audit de callstack sur les allocations mémoire exécutables : vérifier que les fonctions d'allocation (VirtualAlloc, VirtualProtect avec PAGE_EXECUTE_*) sont appelées depuis des modules légitimes et que la callstack remonte à un processus/thread légitime. Configuration avancée dans les EDR entreprise.

Process Injection Protection via Attack Surface Reduction (ASR) : les règles ASR de Microsoft Defender peuvent bloquer les injections de processus les plus communes. Règle "Block process injections from Office applications" et "Block untrusted and unsigned processes that run from USB" sont de bonnes pratiques minimales. Il convient d'activer les règles ASR en mode Audit d'abord (30 jours) pour évaluer les faux positifs avant de passer en mode Block — cette étape est souvent sautée et génère des blocages inattendus en production qui réduisent la confiance des équipes IT dans la solution EDR et créent une pression pour désactiver les protections.

Les Frameworks C2 Red Team en 2026 : Cobalt Strike, Havoc, Sliver

Le choix du framework C2 (Command and Control) est déterminant pour les équipes red team. **Cobalt Strike** reste le framework de référence mais son usage est limité par son coût (6000$/utilisateur/an) et par le fait que ses profils de communication par défaut sont massivement détectés. Les équipes red team sérieuses créent des *Malleable C2 Profiles* personnalisés qui imitent le trafic d'applications légitimes (Teams, Zoom, SharePoint) pour rendre le beaconing indiscernable du trafic légitime. Un bon Malleable Profile prend 1-2 jours à développer et tester, mais assure une furtivité réseau significativement supérieure aux profils par défaut.

Havoc Framework (open source, développé par C5pider) est l'alternative moderne la plus utilisée par les équipes red team en 2026. Havoc intègre nativement l'indirect syscall, la sleep obfuscation, et des techniques d'évasion AMSI avancées. Son daemon (serveur C2) offre une interface web similaire à Cobalt Strike mais sans frais de licence. **Sliver** (BishopFox, open source) est une autre alternative populaire avec support des protocoles mTLS, HTTP, DNS, et WireGuard pour les communications C2.

La contre-mesure blue team consiste à maintenir des signatures des frameworks C2 connus dans l'EDR (CrowdStrike et MDE maintiennent des signatures pour Cobalt Strike, Havoc, Sliver) mais aussi à surveiller les patterns comportementaux agnostiques du framework : beaconing régulier (connexion réseau toutes les N secondes vers la même IP/domaine), taille des payloads chiffrés, et patterns de résolution DNS. Ces patterns sont difficiles à masquer complètement sans compromettre la fonctionnalité du C2.

Persistence Furtive : Techniques de Persistance Post-Bypass

Une fois l'accès initial établi et l'EDR contourné, la persistance est la priorité de l'attaquant. Les techniques de persistance classiques (registre Run, tâches planifiées, services Windows) sont bien détectées par les EDR modernes. Les techniques de persistance furtives utilisées par les APT en 2026 incluent :

COM Object Hijacking : les objets COM (Component Object Model) chargés par des applications légitimes via InprocServer32 peuvent être redirigés vers un DLL malveillant en créant une clé de registre dans HKCU (accessible sans droits admin) qui surpasse la définition dans HKLM. L'application légitime charge alors le COM malveillant à la place du légitime. Cette technique génère peu d'alertes car elle exploite un mécanisme Windows normal.

WMI Event Subscription : les abonnements aux événements WMI (Windows Management Instrumentation) permettent d'exécuter du code en réponse à des événements système (logon, modification de fichier, démarrage). Une subscription WMI permanente créée par un attaquant peut exécuter du code après chaque redémarrage, même après suppression du payload initial. Les abonnements WMI sont stockés dans la base CIM et survèrent aux reboots — une revue régulière des subscriptions WMI est donc une mesure de détection de persistance.

Scheduled Task COM Handler : créer une tâche planifiée qui exécute un COM handler malveillant est plus furtif que d'exécuter directement un exécutable, car les EDR inspectent moins rigoureusement les handlers COM enregistrés dans les tâches planifiées que les exécutables directs. La télémétrie Sysmon (Event 12/13 pour registre, Event 11 pour fichiers) et les logs Task Scheduler (Event 4698, 4702) permettent de détecter ces créations.

Analyse de Malware Red Team : Reverse Engineering des Défenses EDR

Les équipes red team avancées passent du temps à **reverse-engineer les agents EDR** pour comprendre précisément ce qu'ils surveillent. En analysant la DLL agent d'un EDR avec IDA Pro ou Ghidra, on identifie les fonctions hookées, les patterns de détection comportementale, et les mécanismes de reporting vers le backend cloud. Cette information permet de concevoir des techniques d'évasion précisément calibrées pour l'EDR spécifique de la cible.

Cette approche est évidemment coûteuse en temps (plusieurs semaines pour analyser un agent EDR complet) et réservée aux opérations ciblées contre des cibles à haute valeur. Les groupes APT étatiques documentés (APT28/Fancy Bear, APT41/Winnti) sont présumés avoir des équipes dédiées à ce reverse engineering des solutions de sécurité. Pour les équipes red team

commerciales, l'approche pragmatique est de tester les techniques standard contre les variantes de l'EDR cible dans un environnement lab avant l'engagement réel.

XDR vs EDR : La Corrélation Cross-Domain comme Défense Principale

Les solutions **XDR (Extended Detection and Response)** étendent la vision de l'EDR au-delà de l'endpoint pour inclure le réseau (NDR), l'email, l'identité (Entra ID/AD), et le cloud. Microsoft Defender XDR, Palo Alto Cortex XDR, et CrowdStrike Falcon XDR corrèlent automatiquement les événements de ces différents domaines pour détecter des chaînes d'attaque que chaque source individuelle ne pourrait pas identifier seule.

Par exemple, une alerte EDR de faible confiance sur un processus PowerShell suspects peut être ignorée individuellement, mais si elle est corrélée avec un email de phishing reçu 2 heures avant par le même utilisateur (détecté par Defender for Office 365), une connexion anormale à SharePoint depuis une IP inhabituelle, et une requête LDAP volumineuse depuis le poste (détectée par Defender for Identity) — l'alerte XDR combinée est quasi-certaine d'être une vraie compromission. C'est ce type de corrélation que les attaquants ont du mal à masquer car elle implique de rendre furtifs simultanément tous les vecteurs de leur chaîne d'attaque.

Cas Pratique : Scénario Red Team Complet avec Bypass EDR

Pour illustrer la réalité des techniques documentées dans cet article, voici un scénario red team synthétique basé sur des éléments observés en conditions réelles (anonymisé) :

Phase 1 — Accès initial : Phishing ciblé (spear phishing) avec un document Word contenant un template remote injection (attaque "template injection" via un fichier dotm distant). Le document ne contient pas de macros — il charge dynamiquement un template depuis un serveur contrôlé via la fonctionnalité Remote Template normal de Word. Le template chargé exécute une macro qui déclenche l'implant Havoc.

Phase 2 — Bypass EDR : L'implant Havoc utilise l'indirect syscall pour l'injection mémoire, la sleep obfuscation pendant les périodes d'inactivité, et un Malleable C2 profile imitant le trafic OneDrive. MDE détecte le fichier Word initial mais laisse passer l'implant Havoc injecté dans un processus winword.exe (processus légitime avec télémétrie EDR de confiance).

Phase 3 — Élévation de privilèges : Exploitation du privileged SeImpersonatePrivilege du compte de service IIS (IUSR) via RoguePotato pour obtenir SYSTEM. Depuis SYSTEM, migration vers un processus svchost.exe pour la persistance. L'alerte MDE sur RoguePotato est générée mais classifiée "medium severity" et noyée dans les alertes du jour sans treatment SOC immédiat.

Phase 4 — Mouvement latéral : Credential dumping via comsvcs.dll (non bloqué par MDE sur cet environnement car règle ASR non activée), récupération des hashes NTLM de 3 comptes admins locaux partagés (LAPS non déployé). Pass-the-Hash vers 15 serveurs avec admin local partageant le même hash. Sur 3 d'entre eux, un compte de service avec droits Domain Admin est connecté — extraction du TGT via Rubeus.

Ce scénario met en évidence que le bypass EDR seul ne suffit pas à compromettre un environnement bien défendu : chaque mesure de durcissement complémentaire (LAPS, règles ASR, monitoring SOC 24h/24) aurait brisé la chaîne à une étape différente.

Évaluation de la Posture EDR/XDR : Le Test Atomic Red Team

Atomic Red Team (Red Canary) est un framework open source de tests de détection basés sur MITRE ATT&CK. Chaque "atomic" est un test minimal qui exécute une technique ATT&CK spécifique et vérifie si l'EDR la détecte. Les équipes bleues peuvent exécuter Atomic Red Team de façon contrôlée pour valider que leurs règles de détection fonctionnent correctement et identifier les lacunes de couverture.

L'approche recommandée est d'exécuter l'ensemble des atomics correspondant aux techniques documentées dans cet article (T1055 pour les injections de processus, T1134 pour la manipulation de tokens, T1543 pour la persistance via services/COM) et de mesurer le taux de détection. Un taux inférieur à 70% sur les techniques MITRE T1 (impact élevé) est un signal

d'alerte nécessitant une révision de la configuration EDR ou des règles SIEM. Ce test peut être automatisé en intégration continue (ci/cd security pipeline) pour vérifier la couverture de détection à chaque mise à jour de configuration.

Contexte France : Référentiels et Tests Red Team Labellisés ANSSI

En France, les tests d'intrusion incluant le bypass EDR doivent être conduits dans un cadre légal et contractuel précis. La PRIS (Prestation de Référence en Sécurité) de l'ANSSI définit les exigences pour les prestataires de test d'intrusion (PASSI — Prestataires d'Audit de la Sécurité des Systèmes d'Information). Les tests red team avancés incluant l'évasion EDR tombent dans la catégorie des missions PASSI "Audit de configuration et tests d'intrusion avancés" qui nécessitent une déclaration préalable et un contrat d'autorisation explicite.

Notre recommandation est d'exiger contractuellement que les équipes red team utilisent **uniquement des techniques documentées dans MITRE ATT&CK** et de spécifier dans le périmètre de mission les techniques autorisées (ex. : injection mémoire autorisée, BYOVD interdit). Cette approche permet de calibrer la sophistication du test selon le profil de menace réel de l'organisation — une PME n'a pas besoin de tester la résistance de son EDR aux BYOVD zero-day utilisés par des APT étatiques.

Pour les entités NIS 2, les exercices red team incluant le bypass EDR constituent une mesure de validation des capacités de détection et réponse exigée implicitement par l'article 21.2.d de la directive NIS 2 (mesures de continuité des activités et gestion des crises). L'ANSSI peut lors de ses audits demander à visualiser les résultats des derniers tests d'intrusion et les actions correctives menées — un red team sans rapport de suivi des remédiation est considéré comme insuffisant pour valider la conformité.

Stratégie d'Obfuscation de la Charge Utile : Encodage et Polymorphisme

L'obfuscation du payload est le premier niveau de défense contre la détection statique. Au-delà du simple chiffrement XOR, les techniques modernes incluent le **polymorphisme** (chaque génération du payload est unique, avec un chiffrement différent, des séquences NOPs variables, et une structure modifiée) et le **metamorphisme** (le code est réécrit fonctionnellement à chaque génération, sans signature commune entre deux instances). Des outils comme **Veil**, **Shellter**, et **Donut** automatisent une partie de cette obfuscation mais leur output est désormais connu des EDR.

Les équipes red team avancées développent leurs propres crypters et stagers personnalisés, souvent en langages moins surveillés que C# ou PowerShell : **Nim**, **Rust**, et **Go** sont populaires car leurs compilateurs génèrent des binaires avec des structures très différentes des outilsoffensifs C# traditionnels, et les EDR ont moins de signatures spécifiques à ces langages. L'utilisation de *Staged payloads* (le stager initial est minuscule et télécharge le payload principal chiffré depuis le C2) réduit également la surface de signature statique car le stager seul ne contient aucune fonctionnalité offensive detectable.

Network-Based EDR Bypass : DNS over HTTPS et Tunnel ICMP

Les solutions NDR (Network Detection and Response) complètent les EDR en surveillant le trafic réseau. Pour bypasser la détection réseau, les attaquants utilisent des protocoles de communication moins surveillés ou qui semblent légitimes. Le **DNS over HTTPS (DoH)** permet d'encapsuler des données C2 dans des requêtes HTTPS vers des résolveurs DoH publics (1.1.1.1, 8.8.8.8) — ces connexions ressemblent à du trafic HTTPS légitime vers des DNS connus et passent souvent dans les whitelist réseau.

Le **DNS tunneling** classique (données encodées dans des noms de domaine DNS) est bien détecté par les NDR via l'analyse de l'entropie des requêtes DNS et la fréquence des résolutions. Les techniques plus modernes utilisent des sous-domaines courts et à faible entropie pour ralentir la détection, au prix d'une bande passante réduite. L'**ICMP tunneling** exploite le champ données

des paquets ICMP echo request/reply — peu surveillé dans de nombreux environnements — pour exfiltrer des données et recevoir des commandes. La contre-mesure réseau standard est de bloquer le ICMP sortant vers Internet depuis les postes de travail et de filtrer ICMP vers les plages d'IP non-partenaires.

Mesures Complémentaires : EDR Telemetry et Threat Hunting

Au-delà de la configuration EDR, la *threat hunting* proactive est la réponse organisationnelle aux techniques d'évasion sophistiquées. Un attaquant qui contourne la détection automatique de l'EDR laisse néanmoins des traces dans la télémétrie : patterns de syscalls inhabituels, chargement de DLL depuis des chemins non standard, connexions réseau depuis des processus normalement silencieux, modifications du registre à des moments inhabituels. Un chasseur de menaces expérimenté peut retrouver ces traces en quelques heures dans un SIEM bien configuré avec la télémétrie EDR complète.

Les *hypothèses de threat hunting* basées sur MITRE ATT&CK sont le meilleur point de départ : pour chaque technique documentée dans ce guide, il existe une ou plusieurs traces dans la télémétrie EDR/SIEM que le chasseur peut rechercher. La corrélation temporelle — examiner ce qui s'est passé sur une machine 5 minutes avant et après un événement suspect — est souvent plus révélatrice que la recherche de patterns spécifiques.

Les outils de threat hunting comme **Velociraptor** (open source, déployable en agent léger) permettent de requêter la télémétrie des endpoints en temps réel avec un langage de requête type SQL (VQL — Velociraptor Query Language). En moins de 5 minutes, un analyste peut interroger l'ensemble du parc pour rechercher des indicateurs de bypass EDR : processus avec des régions mémoire exécutables non-backed, threads pointant vers des zones mémoire non-mappées depuis des fichiers légitimes, ou connexions réseau depuis des processus normalement silencieux. Cette capacité de hunt à la volée est complémentaire aux alertes automatiques de l'EDR et permet de détecter les techniques d'évasion les plus sophistiquées qui passent sous les radars de la détection automatique.

Ressources Red Team et Références Offensives Documentées

MITRE ATT&CK TA0005 — Defense Evasion (techniques bypass EDR)

Microsoft Security Blog — Defending against BYOVD kernel driver abuse

Articles Connexes pour les Équipes Red et Blue Team

Pour les aspects Active Directory dans les scénarios red team avancés, notre guide Durcissement Active Directory 2026 couvre les contre-mesures structurelles. L'article sur NTLM Relay 2026 et défenses complète la vision des techniques post-exploitation sur réseau. Pour les aspects Entra ID et cloud dans les scénarios red team hybrides, consultez notre analyse BadSuccessor et attaques Entra ID 2026. La détection comportementale dans le SIEM est détaillée dans notre article sur la détection des attaques Golden SAML.

Foire Aux Questions sur le Bypass EDR/XDR en Red Team

Est-il possible de contourner un EDR à 100% en 2026 ?

Non, et cette question posée en ces termes révèle une mauvaise compréhension de l'objectif des EDR. Un EDR ne vise pas à être inviolable — il vise à rendre l'attaque suffisamment difficile et bruyante pour qu'elle soit détectée ou abandonnée. Les techniques de bypass 2026 les plus sophistiquées (indirect syscalls, sleep obfuscation, BYOVD) demandent des semaines de développement, des compétences de reverse

engineering kernel-level, et génèrent malgré tout des artefacts dans la télémétrie ETW et les logs kernel. Un attaquant qui investit 3 semaines dans le bypass EDR d'une cible investit aussi 3 semaines de temps d'exposition à la détection de ses équipes de développement. La meilleure défense reste la détection comportementale et le threat hunting, qui fonctionnent indépendamment des techniques d'évasion spécifiques.

Comment choisir entre CrowdStrike, SentinelOne et Microsoft Defender for Endpoint ?

Les trois solutions sont dans le top 4 du Magic Quadrant Gartner 2025 pour les Endpoint Protection Platforms et offrent des niveaux de détection comparables contre les techniques documentées. Le choix doit se baser sur trois critères : l'intégration avec l'écosystème existant (Microsoft E5 rend MDE très attractif pour les organisations déjà sous licence), la capacité de l'équipe SOC à exploiter la télémétrie (CrowdStrike a une interface threat hunting très appréciée des chasseurs de menaces), et le coût total de possession incluant la formation. Pour les organisations françaises sous NIS 2, les trois solutions sont certifiées et acceptées par l'ANSSI comme composantes de défense. La vraie différence ne se fait pas sur la technologie mais sur la qualité de l'utilisation : un MDE bien configuré avec threat hunting actif surpasse un CrowdStrike "déployé et oublié".

Quelles Sigma rules prioritaires pour détecter les techniques d'évasion EDR ?

Les règles Sigma à déployer en priorité pour détecter les techniques de bypass EDR : (1) chargement de ntdll.dll depuis un chemin non-système (indicateur d'unhooking), (2) VirtualAlloc/VirtualProtect avec PAGE_EXECUTE_READWRITE depuis un processus non-Microsoft (indicateur d'injection mémoire), (3) processus Windows standard (notepad.exe, calc.exe) créant des connexions réseau sortantes (indicateur d'injection dans processus légitime), (4) création de service avec un driver non signé ou non whitelisté

(indicateur BYOVD), et (5) modification des hooks en mémoire sur les DLL système (indicateur d'ETW patching). Ces règles sont disponibles dans le dépôt GitHub SigmaHQ et doivent être adaptées aux spécificités de l'environnement avant déploiement pour éviter les faux positifs.

À RETENIR

Points clés Bypass EDR/XDR Red Team 2026

Les EDR modernes opèrent à 4 niveaux : user-mode hooks, kernel callbacks, ETW-TI, et PPL sensors — une évasion complète nécessite de contourner chaque niveau

Direct syscalls et indirect syscalls contournent les hooks user-mode mais restent visibles via ETW-TI et les callbacks kernel des EDR avancés

BYOVD est la technique la plus efficace mais aussi la plus risquée — HVCI actif le neutralise entièrement

Sleep obfuscation rend les beacons C2 invisibles aux scans mémoire pendant les périodes d'inactivité

HVCI + Microsoft Vulnerable Driver Blocklist : la combinaison défensive la plus impactante contre les techniques kernel-level

Threat hunting comportemental reste efficace même contre les techniques d'évasion les plus sophistiquées

Attack Surface Reduction (ASR) : les règles Microsoft bloquent les vecteurs d'injection les plus communs sans impact majeur sur la productivité

C2 sur canaux légitimes (Teams, OneDrive) est le vecteur réseau le plus difficile à détecter — surveillance comportementale par processus obligatoire