AYI NEDJIMI Consultants

ETW & WPR : Guide Complet et Bonnes Pratiques pour Experts

Catégorie : Forensics Lecture : 8 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

Guide expert ETW et WPR pour forensics Windows : architecture de traçage, collecte d Event Tracing (ETW) & Windows Performance Recorder. Expert en.

Event Tracing (ETW) & Windows Performance Recorder pour Investigations Forensiques Avancées

Cette analyse technique de ETW & WPR 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. Guide expert ETW et WPR pour forensics Windows : architecture de traçage, collecte d Event Tracing (ETW) & Windows Performance Recorder. Expert en. L'investigation numérique exige rigueur et méthodologie. ETW & WPR : Guide Complet et Bonnes Pratiques pour Experts couvre les aspects pratiques que les analystes forensics rencontrent sur le terrain. Nous abordons notamment : event tracing (etw) & windows performance recorder pour investigations forensiques avancées, introduction : l'architecture de traçage windows au service du forensics et architecture technique d'etw : les fondations du traçage système. Les professionnels y trouveront des recommandations actionnables, des commandes prêtes à l'emploi et des stratégies de mise en œuvre adaptées aux environnements d'entreprise.

Introduction : L'Architecture de Traçage Windows au Service du Forensics

Event Tracing for Windows (ETW) représente l'infrastructure de **traçage** la plus puissante et méconnue de l'écosystème Windows. Conçue initialement pour le débogage et l'optimisation des performances, cette technologie s'est révélée être un outil forensique d'une redoutable efficacité. Couplée à Windows Performance Recorder (WPR), elle offre aux analystes en sécurité une capacité d'observation granulaire des activités système, dépassant largement les capacités des journaux d'événements traditionnels. L'investigation numerique et l'analyse forensique constituent des disciplines essentielles de la cybersecurite moderne. Face a la multiplication des incidents de securite, les analystes DFIR doivent maitriser un ensemble d'outils et de methodologies pour identifier, collecter et analyser les preuves numeriques de maniere rigoureuse. Cet article detaille les techniques avancees, les processus de chaine de custody et les bonnes pratiques pour mener des investigations efficaces dans des environnements complexes.

L'**architecture** ETW fonctionne sur un modèle publish-subscribe hautement optimisé, capable d'enregistrer des millions d'événements par seconde avec un impact minimal sur les performances. Cette caractéristique la rend particulièrement précieuse dans les contextes d'investigation post-incident, où la reconstitution précise de la chronologie des événements devient cruciale. Les traces ETW capturent des informations que les attaquants ne peuvent facilement effacer, contrairement aux journaux classiques, offrant ainsi une source de preuves numériques particulièrement robuste.

La complexité apparente **d'ETW** cache une architecture élégante basée sur trois composants principaux : les providers (fournisseurs d'événements), les sessions de traçage, et les consumers (consommateurs). Cette séparation des responsabilités permet une flexibilité extraordinaire dans la collecte et l'analyse des données. Les providers génèrent des événements structurés, les sessions les capturent selon des critères définis, et les consumers les traitent pour extraire des informations exploitables.

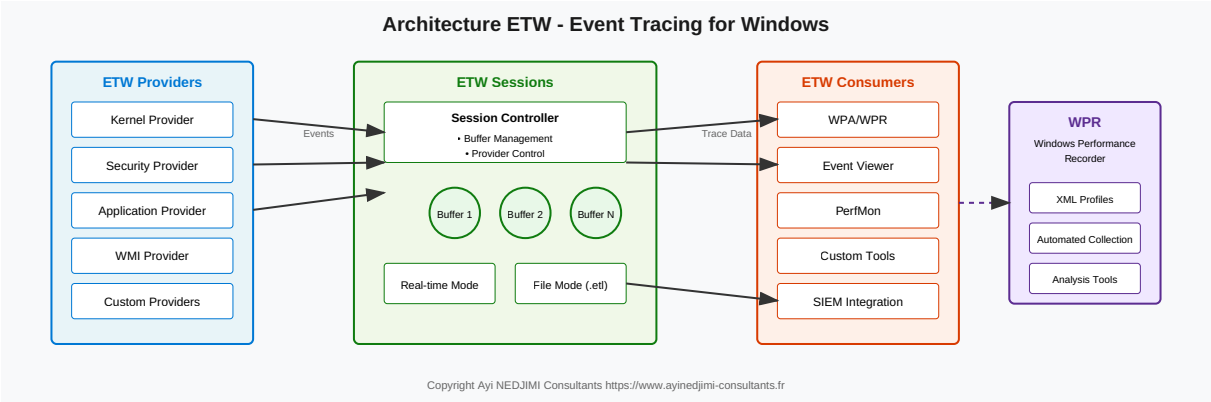

Illustration 1 : Architecture ETW - Event Tracing for Windows

Windows Performance Recorder, quant à lui, encapsule la complexité d'ETW dans une interface plus accessible, tout en préservant sa puissance. WPR utilise des profils XML personnalisables qui définissent précisément quels événements capturer, permettant aux analystes de créer des scénarios de collecte adaptés à leurs besoins spécifiques. Cette approche profile-based simplifie considérablement le déploiement de stratégies de collecte standardisées au sein d'une organisation.

L'adoption d'ETW et WPR dans le domaine forensique représente un changement de schéma. Au lieu de se limiter aux artefacts post-mortem traditionnels, les analystes peuvent désormais capturer l'activité **système** en temps réel avec une granularité microseconde. Cette capacité transforme radicalement l'approche des investigations, permettant non seulement de détecter ce qui s'est passé, mais aussi comment et pourquoi cela s'est produit.

Notre avis d'expert

L'analyse de la mémoire vive est devenue incontournable dans les investigations modernes. Les malwares fileless, les attaques living-off-the-land et les techniques d'injection en mémoire ne laissent souvent aucune trace sur le disque. Ignorer la RAM, c'est passer à côté de 60% des preuves.

Vos journaux d'événements sont-ils conservés suffisamment longtemps pour une investigation ?

Architecture Technique d'ETW : Les Fondations du Traçage Système

Le Kernel ETW Provider et l'Infrastructure de Base

Le noyau Windows implémente ETW au niveau le plus fondamental du système d'exploitation. **Le Kernel** Logger Session constitue la session de traçage privilégiée capable d'enregistrer les événements système critiques. Cette session spéciale, identifiée par le GUID {9E814AAD-3204-11D2-9A82-006008A86939}, capture des événements que les sessions utilisateur standard ne peuvent observer. Pour approfondir, consultez NTFS Forensics.

```
# Démarrage d'une session Kernel avec événements Process/Thread/Registry
logman create trace KernelSession -p "Windows Kernel Trace" (process,thread,registry) -ets

# Configuration avancée avec buffer sizing
logman update trace KernelSession -bs 1024 -min 300 -max 500 -ets
```

L'architecture interne d'ETW repose sur des buffers circulaires alloués en mémoire non paginable. Chaque session maintient plusieurs buffers, permettant l'écriture simultanée d'événements pendant que d'autres buffers sont vidés vers le disque. Cette architecture double-buffering garantit une perte minimale d'événements même sous charge intense. Les paramètres de configuration des buffers influencent directement la capacité de capture et la performance du système.

Providers Manifestes et Classiques : Deux Modèles de Traçage

Les providers ETW se divisent en deux catégories fondamentales. Les providers classiques, hérités de versions antérieures de Windows, utilisent des structures MOF (Managed Object Format) pour définir leurs schémas d'événements. Les providers manifestes, introduits avec Windows Vista, emploient des manifestes XML embarqués dans les binaires, offrant une meilleure performance et une intégration supérieure avec l'infrastructure Windows moderne.

```
<!-- Exemple de manifeste provider personnalisé -->
<instrumentationManifest xmlns="http://schemas.microsoft.com/win/2004/08/events">
  <instrumentation>
    <events>
      <provider name="CustomForensicProvider"
                guid="{12345678-1234-1234-1234-123456789ABC}"
                symbol="FORENSIC_PROVIDER"
                resourceFileName="forensic.dll"
                messageFileName="forensic.dll">
        <channels>
          <channel name="Forensic-Operational" type="Operational" enabled="true"/>
        </channels>
        <templates>
          <template tid="ProcessExecTemplate">
            <data name="ProcessId" inType="win:UInt32"/>
            <data name="CommandLine" inType="win:UnicodeString"/>
            <data name="ParentProcessId" inType="win:UInt32"/>
            <data name="SHA256Hash" inType="win:AnsiString"/>
          </template>
        </templates>
        <events>
          <event symbol="PROCESS_EXECUTION" value="1"
                 version="1" template="ProcessExecTemplate"
                 channel="Forensic-Operational" level="win:Informational"/>
        </events>
      </provider>
    </events>
  </instrumentation>
</instrumentationManifest>
```

La registration des providers dans le système s'effectue via l'API EventRegister pour les providers manifestes, ou via RegisterTraceGuids pour les providers classiques. Cette registration établit le callback qui sera invoqué lorsqu'une session demande l'activation du **provider**. Le mécanisme de contrôle permet une activation dynamique et sélective des providers, minimisant ainsi l'overhead en production.

Sessions de Traçage et Mécanismes de Contrôle

Les sessions ETW constituent le cœur du mécanisme de collecte. Chaque session possède des caractéristiques uniques définissant son comportement : mode de logging (temps réel, fichier, ou les deux), taille et nombre de buffers, flush timer, et niveau de privilège. La création programmatique de sessions offre un contrôle fin sur ces paramètres.

```
// Création d'une session ETW programmatique
EVENT_TRACE_PROPERTIES* pSessionProperties = nullptr;
ULONG bufferSize = sizeof(EVENT_TRACE_PROPERTIES) + sizeof(LOGFILE_PATH) +
sizeof(SESSION_NAME);
pSessionProperties = (EVENT_TRACE_PROPERTIES*)malloc(bufferSize);
ZeroMemory(pSessionProperties, bufferSize);

pSessionProperties->Wnode.BufferSize = bufferSize;
pSessionProperties->Wnode.Flags = WNODE_FLAG_TRACED_GUID;
pSessionProperties->Wnode.ClientContext = 1; // QPC clock resolution
pSessionProperties->Wnode.Guid = SessionGuid;
pSessionProperties->LogFileMode = EVENT_TRACE_FILE_MODE_SEQUENTIAL |
EVENT_TRACE_REAL_TIME_MODE;
pSessionProperties->MaximumFileSize = 100; // MB
pSessionProperties->LoggerNameOffset = sizeof(EVENT_TRACE_PROPERTIES);
pSessionProperties->LogFileNameOffset = sizeof(EVENT_TRACE_PROPERTIES) +
sizeof(SESSION_NAME);
pSessionProperties->FlushTimer = 1; // Flush every second
pSessionProperties->BufferSize = 64; // KB
pSessionProperties->MinimumBuffers = 20;
pSessionProperties->MaximumBuffers = 100;

TRACEHANDLE sessionHandle = 0;
ULONG status = StartTrace(&sessionHandle, SESSION_NAME, pSessionProperties);
```

Le contrôle des sessions s'effectue via les APIs ControlTrace et EnableTraceEx2. Ces fonctions permettent de modifier dynamiquement les paramètres de session, d'activer ou désactiver des providers, et d'ajuster les niveaux de verbosité. La granularité du contrôle s'étend jusqu'au niveau des keywords individuels, permettant un filtrage précis des événements capturés. Pour approfondir, consultez Malware Reverse : Analyse de Cobalt Strike 5.

Performance Counters et Métriques Système

ETW intègre nativement le support des compteurs de performance Windows, permettant la corrélation entre les événements et les métriques système. Cette intégration révèle des patterns d'activité invisibles autrement. Les compteurs capturés incluent l'utilisation CPU, la mémoire, les I/O disque, et l'activité réseau, créant un contexte riche pour l'analyse forensique.

```
# Ajout de compteurs de performance à une trace WPR
wpr -start CPU -start DiskIO -start Network
# Collecte pendant l'incident
Start-Sleep -Seconds 60
wpr -stop forensic_capture.etl

# Extraction des métriques via WPA
$xperfPath = "\${env:ProgramFiles(x86)}\\Windows Kits\\\10\\Windows Performance Toolkit\
\xperf.exe"
& $xperfPath -i forensic_capture.etl -o metrics.csv -target machine -a tracestats -detail
```

Artefact	Localisation	Information extraite
Registre	SYSTEM, SAM, SOFTWARE	Configuration, comptes, services
Event Logs	Security, System, Application	Connexions, erreurs, alertes
Prefetch	C:\Windows\Prefetch	Programmes executes et timestamps
MFT	$MFT sur volume NTFS	Fichiers crees, modifies, supprimes

Cas concret
Lors de l'investigation de l'attaque sur TV5Monde (2015), les analystes forensiques ont découvert que les attaquants — attribués au groupe APT28 — étaient présents dans le réseau depuis plus de 3 mois avant l'attaque destructrice. Cette phase de reconnaissance prolongée souligne l'importance du threat hunting proactif.

Windows Performance Recorder : L'Orchestrateur de Collecte

Profiles WPR et Personnalisation Avancée

Windows Performance Recorder utilise des profils XML pour définir les scénarios de collecte. Ces profils encapsulent la complexité de configuration ETW dans des templates réutilisables. Un profil WPR comprend plusieurs sections : collectors (définissant les sessions), providers (spécifiant les sources d'événements), et profiles (combinant collectors et providers pour des scénarios spécifiques).

```
<?xml version="1.0" encoding="utf-8"?>
<WindowsPerformanceRecorder Version="1.0">
  <Profiles>
    <SystemCollector Id="ForensicSystemCollector" Name="Forensic System Collector">
      <BufferSize Value="1024"/>
      <Buffers Value="100" PercentageOfTotalMemory="true"/>
    </SystemCollector>

    <EventCollector Id="ForensicEventCollector" Name="Forensic Event Collector">
      <BufferSize Value="1024"/>
      <Buffers Value="0.1" PercentageOfTotalMemory="true"/>
    </EventCollector>

    <SystemProvider Id="ForensicSystemProvider">
      <Keywords>
        <Keyword Value="0x10000000000"/> <!-- LOADER -->
        <Keyword Value="0x00000000010"/> <!-- PROCESS -->
        <Keyword Value="0x00000000020"/> <!-- THREAD -->
        <Keyword Value="0x00000000080"/> <!-- IMAGE/DLL -->
        <Keyword Value="0x00000000400"/> <!-- REGISTRY -->
        <Keyword Value="0x00000001000"/> <!-- FILE_IO -->
        <Keyword Value="0x00000002000"/> <!-- DISK_IO -->
      </Keywords>
      <Stacks>
        <Stack Value="ProcessCreate"/>
        <Stack Value="ProcessDelete"/>
        <Stack Value="ImageLoad"/>
        <Stack Value="ImageUnload"/>
      </Stacks>
    </SystemProvider>

    <EventProvider Id="Microsoft-Windows-Kernel-Network"
Name="7DD42A49-5329-4832-8DFD-43D979153A88">
      <Keywords>
        <Keyword Value="0xFFFFFFFFFFFFFFFF"/>
      </Keywords>
      <CaptureStateOnStart>
        <Keyword Value="0xFFFFFFFFFFFFFFFF"/>
      </CaptureStateOnStart>
    </EventProvider>

    <Profile Id="ForensicProfile.Verbose.File" LoggingMode="File" Name="ForensicProfile"
             DetailLevel="Verbose" Description="Comprehensive Forensic Collection
Profile">
      <Collectors>
        <SystemCollectorId Value="ForensicSystemCollector">
          <SystemProviderId Value="ForensicSystemProvider"/>
        </SystemCollectorId>
        <EventCollectorId Value="ForensicEventCollector">
          <EventProviders>
            <EventProviderId Value="Microsoft-Windows-Kernel-Network"/>
            <EventProviderId Value="Microsoft-Windows-DNS-Client"/>
            <EventProviderId Value="Microsoft-Windows-WebIO"/>
          </EventProviders>
        </EventCollectorId>
      </Collectors>
    </Profile>
  </Profiles>
</WindowsPerformanceRecorder>
```

La personnalisation des profils permet d'adapter la collecte aux besoins spécifiques de l'investigation. Les analystes peuvent créer des profils ciblés pour différents types d'incidents : compromission par malware, exfiltration de données, escalade de privilèges, ou mouvement latéral. Chaque profil optimise le ratio signal/bruit en capturant uniquement les événements pertinents.

Stratégies de Déploiement et Collecte à Grande Échelle

Le déploiement de WPR dans un environnement d'entreprise nécessite une stratégie réfléchie. L'utilisation de Group Policy Objects (GPO) permet la distribution centralisée de profils personnalisés et l'automatisation de la collecte. Les scripts PowerShell facilitent l'orchestration de collectes massives lors d'incidents affectant plusieurs systèmes.

```
# Script de déploiement WPR via PowerShell Remoting
$targetComputers = Get-ADComputer -Filter {OperatingSystem -like "*Windows*"} | Select-
Object -ExpandProperty Name
$customProfile = "\\\\FileServer\\Forensics\\Profiles\\AdvancedForensic.wprp"

$scriptBlock = {
    param($ProfilePath)

    # Copie du profil localement
    $localProfile = "$env:TEMP\\forensic.wprp"
    Copy-Item -Path $ProfilePath -Destination $localProfile -Force

    # Démarrage de la collecte
    wpr -start $localProfile

    # Collecte pendant 5 minutes
    Start-Sleep -Seconds 300

    # Arrêt et sauvegarde
    $timestamp = Get-Date -Format "yyyyMMdd_HHmmss"
    $outputFile = "C:\\ForensicData\\$env:COMPUTERNAME_$timestamp.etl"
    wpr -stop $outputFile

    # Retour du chemin pour récupération centralisée
    return $outputFile
}

# Exécution parallèle sur tous les systèmes
$jobs = foreach ($computer in $targetComputers) {
    Invoke-Command -ComputerName $computer -ScriptBlock $scriptBlock -ArgumentList
$customProfile -AsJob
}

# Attente et récupération des résultats
$results = $jobs | Wait-Job | Receive-Job
```

Gestion des Performances et Optimisation de la Collecte

L'impact sur les performances constitue une préoccupation majeure lors du déploiement d'ETW/WPR en production. La configuration des buffers, le choix des providers, et la fréquence de flush influencent directement l'overhead système. Une approche progressive permet d'identifier le sweet spot entre exhaustivité de la collecte et impact acceptable.

Les mécanismes de rate limiting et sampling intégrés à ETW permettent de réduire le volume de données sans compromettre la valeur forensique. L'utilisation du sampling pour les stack traces, par exemple, réduit significativement l'overhead tout en préservant la capacité d'analyse des call chains. La configuration du sampling s'effectue via les propriétés de session ou directement dans les profils WPR.

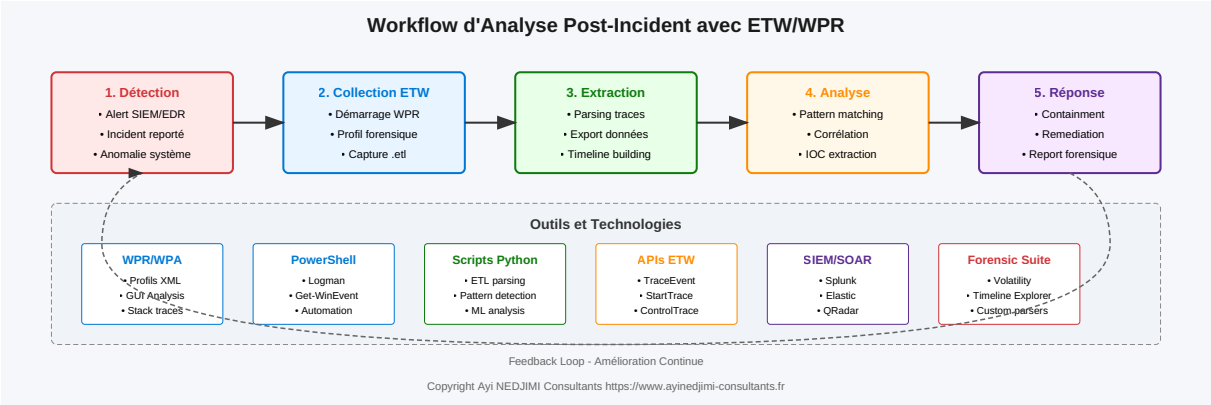

Illustration 2 : Workflow d'Analyse Post-Incident avec ETW/WPR

Analyse Post-Incident : Méthodologies et Techniques Avancées

Parsing et Interprétation des Traces ETL

Les fichiers ETL (Event Trace Log) générés par WPR contiennent une mine d'informations encodées dans un format binaire optimisé. L'analyse de ces traces nécessite des outils spécialisés capables de décoder les structures d'événements et de reconstruire la chronologie des activités. Windows Performance Analyzer (WPA) constitue l'outil de référence, mais des alternatives programmatiques offrent plus de flexibilité pour l'automatisation.

```
// Parsing programmatique des traces ETL avec TraceProcessing
using Microsoft.Windows.EventTracing;
using Microsoft.Windows.EventTracing.Processes;
using Microsoft.Windows.EventTracing.Security;
using System;
using System.Linq;

class ETLForensicAnalyzer
{
    public static void AnalyzeProcessBehavior(string etlFile)
    {
        using (ITraceProcessor trace = TraceProcessor.Create(etlFile))
        {
            IPendingResult<IProcessDataSource> processPending = trace.UseProcesses();
            IPendingResult<ISecurityDataSource> securityPending = trace.UseSecurity();

            trace.Process();

            IProcessDataSource processData = processPending.Result;
            ISecurityDataSource securityData = securityPending.Result;

            // Analyse des processus suspects
            var suspiciousProcesses = processData.Processes
                .Where(p => p.CommandLine != null &&
                       (p.CommandLine.Contains("powershell") ||
                        p.CommandLine.Contains("cmd") ||
                        p.CommandLine.Contains("wscript")))
                .OrderBy(p => p.CreateTime);

            foreach (var process in suspiciousProcesses)
            {
                Console.WriteLine($"[{process.CreateTime}] PID: {process.Id}");
                Console.WriteLine($"  Parent: {process.ParentId}");
                Console.WriteLine($"  Command: {process.CommandLine}");
                Console.WriteLine($"  Image: {process.ImageName}");

                // Analyse des images chargées
                var loadedImages = process.Images
                    .Where(i => i.LoadTime.HasValue)
                    .OrderBy(i => i.LoadTime);

                foreach (var image in loadedImages)
                {
                    Console.WriteLine($"    [{image.LoadTime}] Loaded: {image.FileName}");
                    Console.WriteLine($"      Size: {image.Size} bytes");
                }

                // Corrélation avec les événements de sécurité
                var relatedSecurityEvents = securityData.Events
                    .Where(e => e.ProcessId == process.Id)
                    .OrderBy(e => e.Timestamp);

                foreach (var secEvent in relatedSecurityEvents)
                {
                    Console.WriteLine($"    Security Event: {secEvent.EventId} at
{secEvent.Timestamp}");
                }
            }
        }
    }
}
```

L'interprétation des traces nécessite une compréhension profonde des patterns d'activité normaux et anormaux. Les analystes développent des heuristiques pour identifier les comportements suspects : injection de processus, escalade de privilèges, persistence mechanisms, et communication C2. La corrélation temporelle entre différents types d'événements révèle souvent la chaîne d'attaque complète.

Reconstruction de la Timeline d'Attaque

La reconstruction chronologique constitue l'essence de l'analyse forensique. ETW fournit des timestamps haute précision (QueryPerformanceCounter) permettant une reconstruction au niveau microseconde. Cette granularité révèle les séquences d'actions rapides caractéristiques des attaques automatisées ou des exploits.

Corrélation Multi-Sources et Enrichissement

L'efficacité de l'analyse ETW augmente exponentiellement lorsqu'elle est corrélée avec d'autres sources de données. La fusion des traces ETW avec les journaux d'événements Windows, les logs de pare-feu, et les données EDR crée une vue holistique de l'incident. Cette approche multi-sources permet de combler les lacunes individuelles de chaque source. Pour approfondir, consultez Mobile Forensics : Extraction et Analyse iOS/Android.

```
# Script PowerShell pour corrélation multi-sources
function Merge-ForensicDataSources {
    param(
        [string]$ETLFile,
        [string]$EvtxDirectory,
        [string]$NetworkCapture,
        [string]$OutputPath
    )

    # Conversion ETL vers format analysable
    $etlData = & wpa.exe -export $ETLFile -profile correlation.wpaProfile
    $etlEvents = Import-Csv "etl_export.csv"

    # Extraction des événements Windows
    $evtxEvents = @()
    Get-ChildItem -Path $EvtxDirectory -Filter "*.evtx" | ForEach-Object {
        $events = Get-WinEvent -Path $_.FullName -ErrorAction SilentlyContinue
        $evtxEvents += $events | Select-Object TimeCreated, Id, Message, ProviderName
    }

    # Parsing capture réseau (nécessite tshark)
    $networkEvents = & tshark -r $NetworkCapture -T fields \
```

Ressources open source associées :

- ETWThreatHunter — Threat hunter ETW (C++)
- SysmonEventCorrelator — Corrélateur événements Sysmon (C++)
- forensics-windows-fr — Dataset forensics Windows (HuggingFace)

Bonnes pratiques ETW et WPR

- Configuration des providers ETW critiques pour la securite
- Utilisation de WPR pour les captures de performance
- Analyse des traces avec Windows Performance Analyzer
- Integration des logs ETW dans les pipelines SIEM
- Surveillance des sessions ETW pour detecter le tampering

Questions frequentes

Comment mener une investigation forensique sur un systeme compromis ?

Une investigation forensique debute par la preservation des preuves via une image disque et un dump memoire, suivie de l'analyse des artefacts systeme (registres, journaux d'evenements, fichiers prefetch), la reconstruction de la timeline d'activite et la correlation des indicateurs de compromission pour identifier la source et l'etendue de l'attaque.

Quels sont les outils essentiels pour l'analyse forensique ?

Les outils essentiels pour l'analyse forensique incluent Volatility pour l'analyse memoire, Autopsy et FTK pour l'analyse disque, KAPE et Velociraptor pour la collecte automatisee, Plaso pour la creation de timelines, ainsi que des outils de triage comme Eric Zimmerman's tools pour l'analyse des artefacts Windows.

Pourquoi la chaine de custody est-elle importante en forensique ?

La chaine de custody garantit l'integrite et l'admissibilite des preuves numeriques en documentant chaque etape de manipulation, de la collecte a la presentation. Sans une chaine de custody rigoureuse, les preuves peuvent etre contestees juridiquement et perdre leur valeur probante.

Pour approfondir, consultez les ressources officielles : SANS White Papers, NVD - NIST et ANSSI.

Sources et références : SANS SIFT · MITRE ATT&CK

Articles connexes

- Windows Forensics : Guide Expert en Analyse Securite

Conclusion

Cet article a couvert les aspects essentiels de Architecture Technique d'ETW : Les Fondations du Traçage Système, Windows Performance Recorder : L'Orchestrateur de Collecte, Analyse Post-Incident : Méthodologies et Techniques Avancées. La mise en pratique de ces recommandations permet de renforcer significativement la posture de securite de votre organisation.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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