

## Build Week 3 – Esercizio 6: Estrarre un Eseguibile da un PCAP

### Obiettivi:

- **Parte 1:** Analizzare Log e Catture di Traffico Pre-catturati
- **Parte 2:** Estrarre File Scaricati dal PCAP

### Contesto/Scenario

Guardare i log è molto importante, ma è anche importante capire come avvengono le transazioni di rete a livello di pacchetto. In questo laboratorio, analizzerai il traffico in un file pcap catturato in precedenza ed estrarrai un eseguibile dal file.

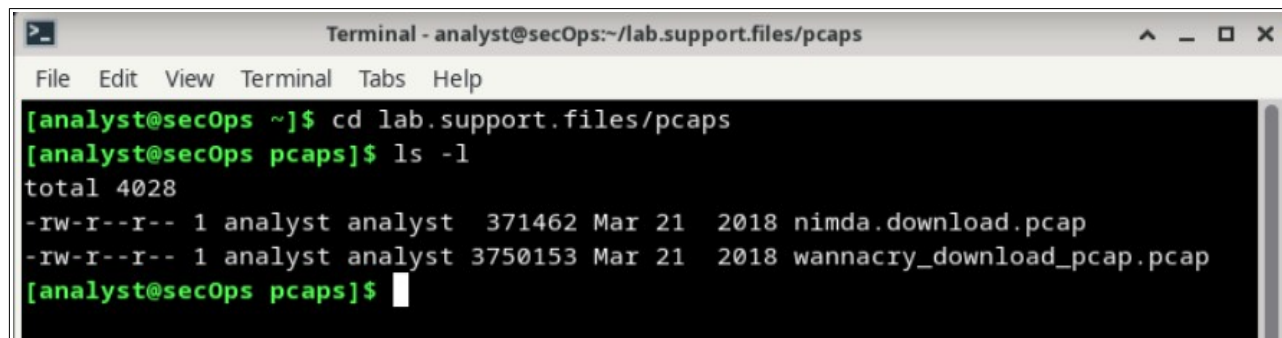
### Risorse Richieste

- **Macchina virtuale CyberOps Workstation**

## Parte 1: Analizzare Log e Catture di Traffico Pre-catturati

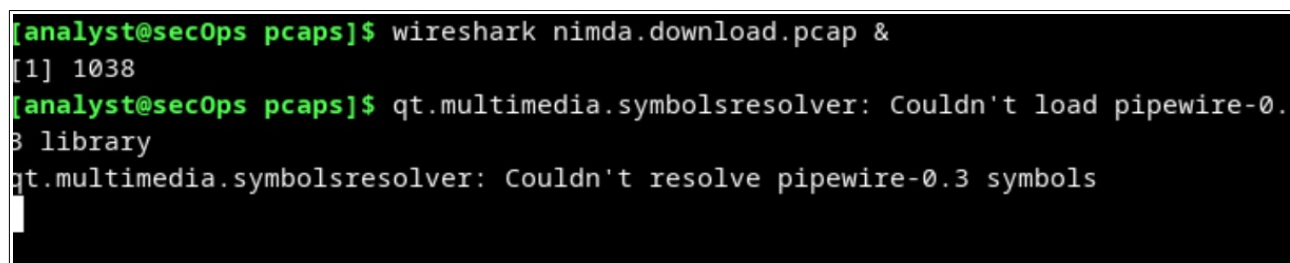
La prima parte dell'esercizio è finalizzata a comprendere come leggere e interpretare una cattura di rete già disponibile, senza dover generare del nuovo traffico. Viene utilizzato il file “*nimda.download.pcap*”, registrato in precedenza, che contiene i pacchetti relativi al download di un malware. Attraverso strumenti come Wireshark, è possibile osservare il comportamento delle connessioni TCP e HTTP, distinguere le varie fasi della comunicazione e ricostruire un intero flusso di dati. Questo consente di analizzare non solo i pacchetti singoli, ma anche il contenuto completo della transazione, evidenziando come i dati binari possano contenere stringhe leggibili che aiutano a identificare la natura del file scaricato.

In questo passo viene mostrato come navigare all'interno della directory che contiene i file PCAP di laboratorio. Con il comando “*cd*” ci si sposta nella cartella “*lab.support.files/pcaps*”, e con “*ls -l*” si ottiene l'elenco dei file disponibili.



```
Terminal - analyst@secOps:~/lab.support.files/pcaps
File Edit View Terminal Tabs Help
[analyst@secOps ~]$ cd lab.support.files/pcaps
[analyst@secOps pcaps]$ ls -l
total 4028
-rw-r--r-- 1 analyst analyst 371462 Mar 21 2018 nimda.download.pcap
-rw-r--r-- 1 analyst analyst 3750153 Mar 21 2018 wannacry_download_pcap.pcap
[analyst@secOps pcaps]$
```

Il comando “*wireshark nimda.download.pcap &*” apre direttamente il file con Wireshark. L'uso del simbolo & serve a mandare il programma in esecuzione in background, così da poter continuare a usare il terminale senza bloccarlo.



```
[analyst@secOps pcaps]$ wireshark nimda.download.pcap &
[1] 1038
[analyst@secOps pcaps]$ qt.multimedia.symbolsresolver: Couldn't load pipewire-0.3 library
qt.multimedia.symbolsresolver: Couldn't resolve pipewire-0.3 symbols
```

Applications: nimda.download.pcap analyst - Thunar Terminal - analyst@sec0...

nimda.download.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	209.165.200.235	209.165.202.133	TCP	74	48598 → 6666 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=4051203246 TSecr=0 WS=512
2	0.000259	209.165.202.133	209.165.200.235	TCP	74	6666 → 48598 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM TSval=3023496465 TSecr=4051203246 WS=512
3	0.000297	209.165.200.235	209.165.202.133	TCP	66	48598 → 6666 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=4051203246 TSecr=3023496465
4	0.000565	209.165.200.235	209.165.202.133	HTTP	230	GET /W32.Nimda.Amm.exe HTTP/1.1
5	0.000588	209.165.202.133	209.165.200.235	TCP	66	6666 → 48598 [ACK] Seq=1 Ack=165 Win=30208 Len=0 TSval=3023496465 TSecr=4051203246
6	0.000708	209.165.202.133	209.165.200.235	TCP	324	6666 → 48598 [PSH, ACK] Seq=1 Ack=165 Win=30208 Len=258 TSval=3023496465 TSecr=4051203246 [TCP PDU reassembly completed]
7	0.000827	209.165.200.235	209.165.202.133	TCP	66	48598 → 6666 [ACK] Seq=165 Ack=259 Win=30720 Len=0 TSval=4051203246 TSecr=3023496465
8	0.004594	209.165.202.133	209.165.200.235	TCP	1514	6666 → 48598 [ACK] Seq=259 Ack=165 Win=30208 Len=1448 TSval=3023496466 TSecr=4051203246 [TCP PDU reassembly completed]
9	0.004602	209.165.200.235	209.165.202.133	TCP	66	48598 → 6666 [ACK] Seq=165 Ack=1707 Win=33280 Len=0 TSval=4051203247 TSecr=3023496466
10	0.004605	209.165.202.133	209.165.200.235	TCP	1514	6666 → 48598 [ACK] Seq=1707 Ack=165 Win=30720 Len=1448 TSval=3023496466 TSecr=4051203246 [TCP PDU reassembly completed]

Frame 4: 230 bytes on wire (1840 bits), 230 bytes captured (1840 bits)

Ethernet II, Src: ea:05:2c:e1:90:3d (ea:05:2c:e1:90:3d), Dst: 16:4c:37:9e:eb:50 (16:4c:37:9e:eb:50)

Internet Protocol Version 4, Src: 209.165.200.235, Dst: 209.165.202.133

Transmission Control Protocol, Src Port: 48598, Dst Port: 6666, Seq: 1, Ack: 1, Len: 164

Hypertext Transfer Protocol

```

GET /W32.Nimda.Amm.exe HTTP/1.1\r\n
User-Agent: Mget/1.19.1 (linux-gnu)\r\n
Accept: */*\r\n
Accept-Encoding: identity\r\n
Host: 209.165.202.133:6666\r\n
Connection: Keep-Alive\r\n
\r\n
[Response in frame: 309]
[Full request URI: http://209.165.202.133:6666/W32.Nimda.Amm.exe]

```

Wireshark - Follow TCP Stream (tcp.stream eq 0) - nimda.download.pcap

GET /W32.Nimda.Amm.exe HTTP/1.1  
 User-Agent: Wget/1.19.1 (linux-gnu)  
 Accept: /\*/\*  
 Accept-Encoding: identity  
 Host: 209.165.202.133:6666  
 Connection: Keep-Alive

HTTP/1.1 200 OK  
 Server: nginx/1.12.0  
 Date: Tue, 02 May 2017 14:26:50 GMT  
 Content-Type: application/octet-stream  
 Content-Length: 345088  
 Last-Modified: Fri, 14 Apr 2017 19:17:25 GMT  
 Connection: keep-alive  
 ETag: "58f12045-54400"  
 Accept-Ranges: bytes

MZ.....@.....!..L!This program cannot be run in DOS mode.  
 \$....M]... ..eN....e\_....eY....eI....eC....e^....e[....Rich.....PE..d..  
 ...L...X...d...X...  
 ...&...\$...p...8...text...p...rdata...I..  
 ...J...V...@...@.data...@...pdata...&...@...@.rsrc...X...@...@.relo  
 c...\$...B...@...@.B7...@...LK...LK...LU...LK...Lb...L...msvcrt.dll.NTDLL.DLL.KERNEL32.dll.api-ms-win-core-proce  
 ss-threads-11-1-0.DLL.WINBRAND.dll  
 .....H;  
 .....SQ...H...f.....Q.....%.....H...teSH...H...H...tO...L.A.H...t>H.L\$OI;.....H;H.C.....H.....7...L...3.H...1..  
 ...H...[...H...%:.....H...\$H.t\$WH...H...H...%...=.....\$...:-...\$...=.....t\$D.F.H.L\$3...H.T\$E3.H...P1...uf;.....  
 ...3.....;.....%<.....].....<.....Hc.H..  
 .b...H...H...H...H...t  
 H9X...Z...=\*...t.H...t.H.T\$A.H...0...L...\$...I.[I.sI...H...H...j...H...H...G...t...y...<...!...3..  
 ...H...a...H...t1..  
 N...<.....H...a...H...tH..  
 ...H...((...H...H...3...H...H...>...H...Fp...F...H...Fp...\*...D.X.3.H...H...pf...H...H...C...H...Y...H...\$P...H3..  
 L...\$...I...([I.sOI.(BI...A.A)A\...H...H...[...S...i...H.A...F...H...\$WH...3...L...m...H;t.H...3.H...f...H...H...H...H#...6...L...H...3..  
 ...\*...\$...H...H...\$OH...\_...%:].....H...\$H.I\$H.t\$WH...3.H...H...H;t.H.f9(t...H...f;u.f9)u.H+H...H...4...L...H...>

1 client pkt, 243 server pkts, 1 turn.

Entire conversation (345 kB) Show as ASCII No delta times Stream 0

Find:   ☐ Case sensitive

### **Cosa sono tutti quei simboli mostrati nella finestra Follow TCP Stream?**

I simboli che compaiono non sono elementi casuali, ma il risultato della rappresentazione ASCII del flusso di byte trasmesso nella connessione TCP. In pratica, si tratta del contenuto effettivo trasferito durante la sessione. Poiché gran parte di questi dati è binaria e non corrisponde a caratteri leggibili, Wireshark li visualizza come punti o simboli particolari.

### **Sono rumore di connessione? Dati? Spiega.**

Non si tratta di rumore, ma di dati reali della comunicazione. Quello che vediamo è esattamente il contenuto del file o della transazione trasmessa sulla rete, così come è stato inviato e ricevuto dai dispositivi. I simboli poco comprensibili derivano dal fatto che un file binario contiene byte che non possono essere interpretati come testo. Wireshark, cercando di renderli leggibili, li mostra quindi con segni o punti.

### **Ci sono alcune parole leggibili sparse tra i simboli. Perché sono lì?**

All'interno di file binari possono comunque essere presenti stringhe di testo in formato ASCII, come nomi di funzioni, messaggi di errore o riferimenti a librerie utilizzate dal programma. Questi frammenti, a differenza dei byte non leggibili, vengono mostrati in chiaro da Wireshark perché corrispondono a caratteri testuali validi. Per questo, accanto ai simboli, si notano anche parole riconoscibili.

**Domanda Sfida: Nonostante il nome W32.Nimda.Amm.exe, questo eseguibile non è il famoso worm. Per motivi di sicurezza, questo è un altro file eseguibile che è stato rinominato come W32.Nimda.Amm.exe. Usando i frammenti di parole visualizzati dalla finestra Follow TCP Stream di Wireshark, puoi dire quale eseguibile sia realmente?**

Analizzando il flusso TCP con Wireshark, tra i dati binari sono emersi alcuni frammenti di testo leggibili che hanno permesso di identificare meglio il file. In particolare, compaiono riferimenti come:

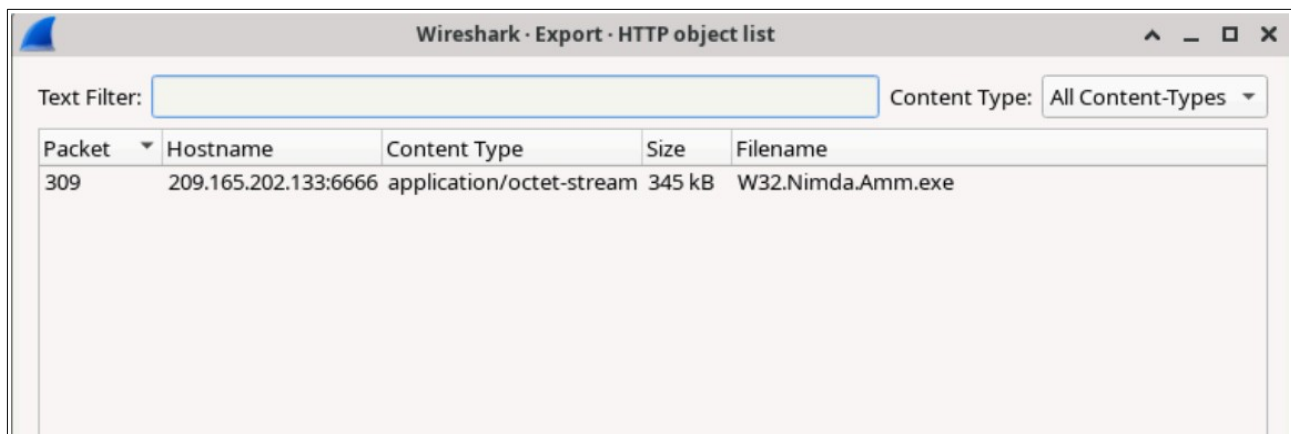
- name = "Microsoft.Windows.FileSystem.CMD"
- <description> Windows Command Processor

Questi elementi corrispondono al manifesto del Windows Command Processor (cmd.exe), l'eseguibile legittimo di sistema di Microsoft. Di conseguenza, anche se il file catturato era stato rinominato come "W32.Nimda.Amm.exe", non si tratta del worm Nimda, bensì di una copia rinominata di cmd.exe, utilizzata a scopo dimostrativo nel laboratorio.

## Parte 2: Estrarre File Scaricati dal PCAP

Nella seconda parte dell'esercizio l'attenzione si concentra sull'estrazione di file direttamente da una cattura di traffico di rete (PCAP). Poiché una cattura contiene tutti i pacchetti scambiati durante una sessione, è possibile ricostruire non solo le comunicazioni, ma anche i file eventualmente scaricati. In questo caso, il file analizzato è *"nimda.download.pcap"*, che documenta il download di un eseguibile rinominato come *"W32.Nimda.Amm.exe"*. Utilizzando le funzionalità di Wireshark, sarà quindi possibile identificare la richiesta HTTP associata, visualizzare gli oggetti trasferiti e salvare localmente l'eseguibile per ulteriori analisi.

Nel quarto pacchetto si individua la richiesta HTTP GET che richiede il file eseguibile: l'endpoint client e server (indirizzi IP e porta) e la riga *"GET /W32.Nimda.Amm.exe"* confermano che la sessione contiene il download dell'eseguibile. Con il pacchetto selezionato, si utilizza la funzione di Wireshark *"File → Export Objects → HTTP"* per elencare e recuperare gli oggetti HTTP trasferiti in quel flusso. Wireshark ricostruisce i dati del flusso e mostra nella finestra HTTP object list tutti gli oggetti disponibili. In questo caso è presente un solo file, *"W32.Nimda.Amm.exe"*. Dopo qualche secondo è possibile selezionarlo e salvarlo localmente per le analisi successive.



### Perché W32.Nimda.Amm.exe è l'unico file nella cattura?

Perché la cattura è stata avviata immediatamente prima del download e interrotta subito dopo il completamento: di conseguenza, l'unico oggetto HTTP presente nel flusso è proprio il file *"W32.Nimda.Amm.exe"*, che rappresenta l'intero contenuto scaricato durante quella sessione.

```
[analyst@secOps pcaps]$ cd /home/analyst
[analyst@secOps ~]$ ls -l
total 380
-rw-r--r-- 1 root    root      4951 Sep 23 13:45 capture.pcap
drwxr-xr-x 2 analyst analyst   4096 Jun 17 19:35 Desktop
drwxr-xr-x 3 analyst analyst   4096 Jun 18 20:17 Downloads
-rw-r--r-- 1 root    root        24 Sep 30 04:02 httpdump.pcap
-rw-r--r-- 1 root    root       178 Sep 30 03:56 httpsdump.pcap
drwxr-xr-x 9 analyst analyst   4096 Jun 18 20:17 lab.support.files
drwxr-xr-x 3 analyst analyst   4096 Jun 18 19:55 scripts
drwxr-xr-x 2 analyst analyst   4096 Mar 21 2018 second_drive
-rw-r--r-- 1 analyst analyst 345088 Sep 30 04:42 W32.Nimda.Amm.exe
drwxr-xr-x 5 analyst analyst   4096 Jun 18 19:27 yay
```

## Il file è stato salvato?

Lo screenshot mostra il terminale dopo essersi spostati nella cartella “/home/analyst” ed aver eseguito il comando `ls -l`. Nell’elenco compare il file “W32.Nimda.Amm.exe”, con i relativi dettagli (permessi, proprietario, dimensione, data e ora di creazione), confermando che il salvataggio dall’export di Wireshark è avvenuto correttamente.

Usando il comando “*file*” possiamo ottenere informazioni sul malware.

```
[analyst@secOps ~]$ file W32.Nimda.Amm.exe
W32.Nimda.Amm.exe: PE32+ executable for MS Windows 6.01 (console), x86-64, 6 sections
```

## Nel processo di analisi del malware, quale sarebbe un probabile passo successivo per un analista di sicurezza?

### 1. Raccolta iniziale e precauzioni

Salvare una copia del campione e calcolare hash (MD5 / SHA1 / SHA256) per tracciabilità.  
Esempio: `sha256sum “W32.Nimda.Amm.exe”`

Lavorare esclusivamente in una VM isolata o in una sandbox (nessuna rete diretta; usare rete controllata o fake DNS/HTTP) e mantenere la catena di custodia dei file.

### 2. Analisi statica

Estrarre stringhe leggibili (`strings W32.Nimda.Amm.exe | less`) utile per trovare URL, nomi, comandi e riferimenti a librerie.

Ispezionare la struttura PE (import table, sezioni) con strumenti come `peframe`, `pefile`, `pestudio`, `CFF Explorer`.

### 3. Analisi dinamica

Eseguire il campione in una sandbox/VM isolata (Cuckoo, AnyRun o una VM snapshot) con rete finta o filtrata per catturare il traffico.

Monitorare la creazione/lettura/scrittura di file, i processi figli, le modifiche al Registro, le comunicazioni di rete (inclusi handshake TLS) e i file droppati.

Strumenti utili: `Procmon`, `Process Explorer`, `Sysmon` (se presente), `Wireshark`, `Regshot` e l’EDR dell’ambiente.

### 4. Analisi memoria

Eseguire un dump della memoria della VM dopo l’esecuzione per recuperare stringhe in memoria, processi iniettati o credenziali in chiaro.

### 5. Analisi del traffico

Analizzare i pcap con `Wireshark` per estrarre URL, oggetti HTTP e identificare eventuali C2 o pattern di esfiltrazione.

Ricostruire i flussi TCP/HTTP e salvare file scaricati per ulteriori analisi statiche/dinamiche.

### 6. Analisi approfondita

Se necessario, deassemblare o decompilare il campione con `Ghidra`, `IDA Pro`, `radare2` o decompilatori .NET per comprendere routine critiche (persistenza, crittografia, esfiltrazione).

Generare regole YARA o firme basate su stringhe o funzioni peculiari individuate.

## **7. Creazione IoC e validazione**

Compilare una lista di indicatori di compromissione: hash, nomi file, percorsi di drop, domini/IP, stringhe HTTP, chiavi di registro modificate e comandi osservati.

Confrontare e ricercare gli IoC all'interno della rete aziendale e nei log (SIEM/EDR) per determinare la portata dell'infezione.

## **8. Raccomandazioni operative**

Se il campione è confermato malevolo: isolare l'host, raccogliere prove forensi, reimaging dell'endpoint, rotazione delle credenziali potenzialmente compromesse e blocco degli IoC su firewall/proxy/EDR.

Se l'analisi è inconcludente: escalare al team di threat intelligence o al vendor per approfondimenti.

## **9. Documentazione**

Redigere un report tecnico e uno executive contenente timeline, evidenze (hash, estratti da log e pcap), valutazione dell'impatto e raccomandazioni di remediation da distribuire al SOC.