



# ALLA RICERCA DI .EXE

COME RECUPERARE UN FILE .EXE DA  
TRAFFICO HTTP

# INTRODUZIONE

Il progetto ha riguardato l'analisi di un file eseguibile sospetto scaricato da un sistema aziendale, in particolare dalla compagnia Theta. L'obiettivo principale è stato quello di utilizzare strumenti di analisi del traffico di rete, come Wireshark, per catturare i pacchetti di rete e recuperare il file .exe durante il suo trasferimento. Attraverso questa attività, abbiamo potuto identificare e analizzare un file sospetto che, inizialmente, sembrava innocuo, ma che in realtà si è rivelato essere un malware mascherato.

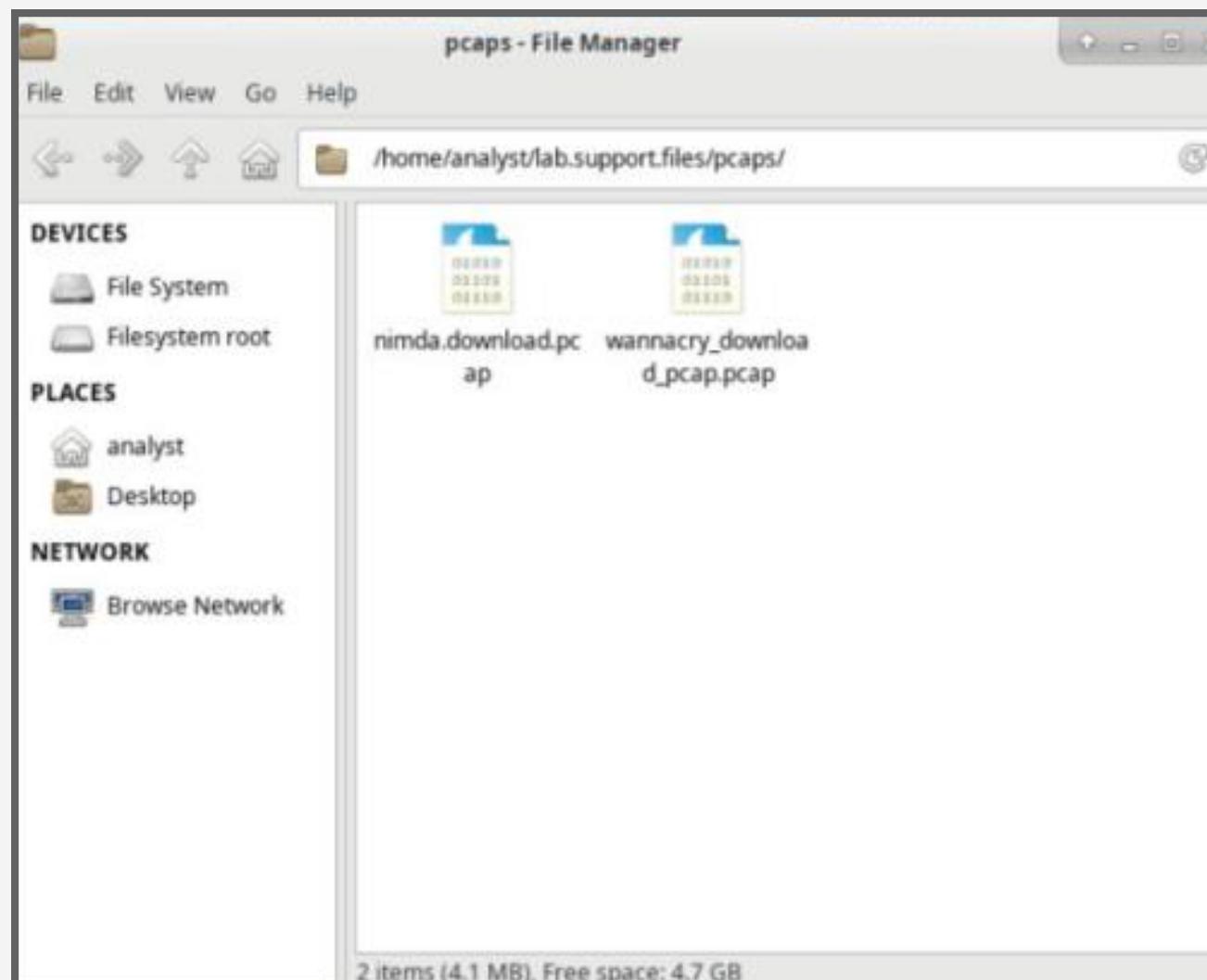
La gestione della sicurezza informatica e la protezione dei sistemi aziendali è fondamentale per prevenire accessi non autorizzati e danni ai dati sensibili. Durante il progetto, abbiamo approfondito come strumenti come Wireshark possano essere utilizzati per monitorare e analizzare il traffico di rete, identificare eventuali minacce e recuperare file dannosi prima che possano compromettere l'integrità dei sistemi aziendali. Inoltre, abbiamo esaminato come il rispetto dei principi di confidenzialità, integrità e accessibilità (CIA) siano essenziali per garantire la sicurezza delle risorse aziendali, in modo da prevenire possibili perdite o alterazioni di dati sensibili.





# I FILE PCAP

Un file PCAP (Packet Capture) è una registrazione del traffico di rete, utilizzato per catturare i pacchetti in transito tra due o più dispositivi. In un contesto di indagine forense, permette di analizzare eventi sospetti, come il trasferimento di file dannosi. Con Wireshark, uno strumento di analisi dei pacchetti, possiamo visualizzare il traffico e ricostruire il contenuto trasmesso, identificando attività malintenzionate, come nel caso del file sospetto scaricato da Theta.



S  
I  
R  
E  
T  
H  
A  
—  
M  
A  
N  
—



**Nei primi tre pacchetti catturati, si osserva la stretta di mano a tre vie del protocollo TCP. Questo processo, fondamentale per stabilire connessioni affidabili, si compone di tre fasi:**

- 1 Il client invia un pacchetto **SYN (Synchronize)** al server per iniziare la connessione.
- 2 Il server risponde con un pacchetto **SYN/ACK (Synchronize-Acknowledge)**, confermando la richiesta.
- 3 Il client completa il processo con un pacchetto **ACK (Acknowledge)**, sancendo l'inizio dello scambio di dati.

TCP	74 48598 → 6666 [SYN] Seq=0 Win=29200 Len=0
TCP	74 6666 → 48598 [SYN, ACK] Seq=0 Ack=1 Win=2
TCP	66 48598 → 6666 [ACK] Seq=1 Ack=1 Win=29696

La stretta di mano è un meccanismo progettato per garantire che entrambe le parti siano pronte a comunicare. È anche un indicatore importante per gli analisti, poiché segnala l'inizio di una connessione sospetta. Nel caso della compagnia Theta, questa connessione è il punto di partenza per l'attacco.

# LA RICHIESTA GET PER IL FILE SOSPETTO

Nel quarto pacchetto, viene individuata una richiesta HTTP di tipo GET, utilizzata dal client per scaricare il file W32.Nimda.Amm.exe. Questa richiesta è un esempio di come gli attaccanti utilizzino protocolli standard come HTTP per trasferire file dannosi senza destare sospetti.

La richiesta GET è visibile nel payload del pacchetto e include dettagli come il nome del file richiesto e il percorso sul server remoto. Questa fase è cruciale per l'indagine, poiché ci conferma il momento esatto in cui il file sospetto è stato scaricato. Questo tipo di analisi permette alle organizzazioni come Theta di individuare vulnerabilità nei loro sistemi e implementare regole di blocco mirate nei firewall o nei sistemi di prevenzione delle intrusioni.

```
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 TStamp=3023496465 TSecr=40
6 0.000709 209.165.202.133 209.165.200.235 TCP 234 FFFF → 48598 [ACK] Seq=1 Ack=165 Win=30208 Len=359 TStamp=3023496465 TSecr=40
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: Wget/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
[Full request URI: http://209.165.202.133:6666/W32.Nimda.Amm.exe]
[HTTP request 1/1]
[Response in frame: 309]
```





# IL FLUSSO TCP E IL CONTENUTO BINARIO

Analizzando il Follow TCP Stream, è stato possibile osservare i dati binari del file eseguibile in transito. Questi caratteri binari non sono leggibili direttamente, ma rappresentano il contenuto effettivo del file .exe che stiamo cercando di recuperare. L'utilizzo di questa funzione ci ha permesso di esaminare il flusso dei pacchetti, ricostruendo il file e svelando informazioni cruciali per l'indagine. Si tratta di una parte fondamentale nell'analisi forense del traffico di rete.

```
Stream Content
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

M2.....@.....! L!This program cannot be run in DOS mode.
$.....M|.....eN.....e.....eY.....eI.....eC.....e^.....eJ.....Rich.....PE.d.....L.....".....r
.....J.....@.....H.....text_p.....r....."rdata_I.....j..v.....@..@.dat
.....a.....@..pdata_A.....@.....@..rsrc_X.....@..@.reloc
.....$.....B.....@..87.L@.....LK.....LU.....LK.....lb.....L.....movcr.dll.NTDLL.DLL.KERNEL32.dll.ap
ms-win-core-processThreads-
II1-1-0.DLL.WINBRAND.dll.....H
.....$Q.H.....F.....Q.....%.....H.....teSH.....H.....H.tO.....LA.H.....DH.L$01.....H.....H.C.....H
.....7.....L.....3.....H.....1.....H.....H.....%.....H.....H.S.....H.....H.....%.....=.....S.....$.....=.....ISO.F.....H.....L$3.....H.TS
.....E3.H.....P1.....uf.....3.....<.....;.....Hc.H
.....b.....H.....H.....H.....H.....H.....H
.....H9X.....Z.....>.....tH.....I.....H.TS.....A.....H.....0.....L.....$.....I.....{.....I.....s1.....H
.....(.....H.....j.....H.....H.....G.....t.y.....<.....I.....3.....H.....a.....H.....t1
.....N.....<.....H.....a.....H.....H
.....H.....f.....H.....H.....H.....3.....H.....f.....>.....H.....F.....F.....H.....F.....*.....D.Y3.....H.....H.....-of.....H.....H.....H.....C.....H.....V.....H
```

## PROTOCOLLO TCP

Transmission Control Protocol (TCP) è un protocollo di trasporto che garantisce una connessione affidabile tra due host, utilizzato per trasmettere dati.

# ANALISI STRINGHE LEGGIBILI



Oltre ai caratteri binari, sono emerse stringhe leggibili, che sono frammenti di testo contenuti nel file. Queste stringhe forniscono informazioni utili per identificare il comportamento del file eseguibile, come comandi di sistema o URL di server di controllo. Un'analisi approfondita delle stringhe può rivelare il tipo di azione che il malware intende eseguire, come l'esecuzione di comandi sul sistema compromesso. Queste informazioni sono cruciali per determinare se il file è dannoso, sostanzialmente come degli indizi all'interno di un'indagine.

```
.RtlCaptureContext...RtlLookupFunctionEntry....RtlVirtualUnwind..K.RtlFreeHeap.*.NtFsControlFile.I.NtOpenThreadToken...NtClose.d.NtOpenProcessToken....NtQueryInformationToken...RtlDosPathNameToNtPathName_U...9.RtlFindLeastSignificantBit....NtSetInformationProcess...NtQueryInformationProcess...RtlNtStatusToDosError...GetTimeFormatW...GetTickCount....QueryPerformanceCounter...SetUnhandledExceptionFilter...Sleep...DelayLoadFailureHook..?.LoadLibraryExA..g.FreeLibrary...CreateHardLinkW...CreateSymbolicLinkW...GetVolumePathNameW...GetThreadLocale...ResumeThread....SetProcessAffinityMask...0.GetNumaNodeProcessorMaskEx....GetThreadGroupAffinity...9.FindFirstFileExW....GetDiskFreeSpaceExW.K.FindNextStreamW.@.FindFirstStreamW....DeviceIoControl.`.CompareFileTime...RemoveDirectoryW....GetCurrentDirectoryW...GetExitCodeProcess....WaitForSingleObject...TerminateProcess..X.SetCurrentDirectoryW.t.SetFileTime...DeleteFileW.^..SetEndOfFile..k.SetFileAttributesW.u.CopyFileW..CreateDirectoryW.Q.SetConsoleTextAttribute...+..FillConsoleOutputAttribute..&.ScrollConsoleScreenBufferW.m.GetACP..c.FormatMessageW...!.FlushFileBuffers....DuplicateHandle...HeapSize....HeapReAlloc...VirtualAlloc....VirtualFree...HeapSetInformation..GetCurrentThreadId....OpenThread....GetFileAttributesExW....GetDriveTypeW..GetVersion.;.LeaveCriticalSection...EnterCriticalSection....GetModuleFileNameW....GetWindowsDirectoryW...8..SetConsoleCtrlHandler...InitializeCriticalSection.".ExpandEnvironmentStringsW.D.CancelSynchronousIo...GetVolumeInformationW..GlobalFree....GlobalAlloc.q.SetFilePointerEx..1..WriteFile...!.SearchPathW.J.LocalFree.S..SetConsoleTitleW..a.MoveFileExW.d.MoveFileW...QueryFullProcessImageNameW....ReadProcessMemory.A.LoadLibraryW....RegSetValueExW....RegCreateKeyExW...UnhandledExceptionFilter....GetCurrentProcess..~..GetSystemTimeAsFileTime...VirtualQuery...!.CmdBatNotification..w.GetCPInfo...GetConsoleOutputCP....SetThreadLocale.I..GetProcAddress....GetModuleHandleW..R.CloseHandle...GetLastError..p..SetFilePointer....GetFullPathNameW..>..FindFirstFileW.J..FindNextFileW...8..FindClose...CreateFileW..ReadFile..h..MultiByteToWideChar...GetFileSize...WideCharToMultiByte.U..lstrcmpiW.R..lstrcmpW..i..GetStdHandle...!.FlushConsoleInputBuffer...HeapAlloc.N..GetProcessHeap....HeapFree....GetConsoleScreenBufferInfo....ReadConsoleW..<..SetConsoleCursorPosition..~..FillConsoleOutputCharacterW...0..WriteConsoleW...GetFileType...GetUserDefaultLCID....GetLocaleInfoW....SetLocalTime..|..GetSystemTime...SystemTimeToFileTime..).FileTimeToLocalFileTime.*..FileTimeToSystemTime....GetDateFormatW....RegDeleteValueW..GetLocalTime....GetConsoleMode..H..SetConsoleMode....GetEnvironmentVariableW..GetCommandLineW..GetNumaHighestNodeNumber....GetEnvironmentStringsW..f..FreeEnvironmentStringsW.b..SetEnvironmentVariableW..`..SetE
```

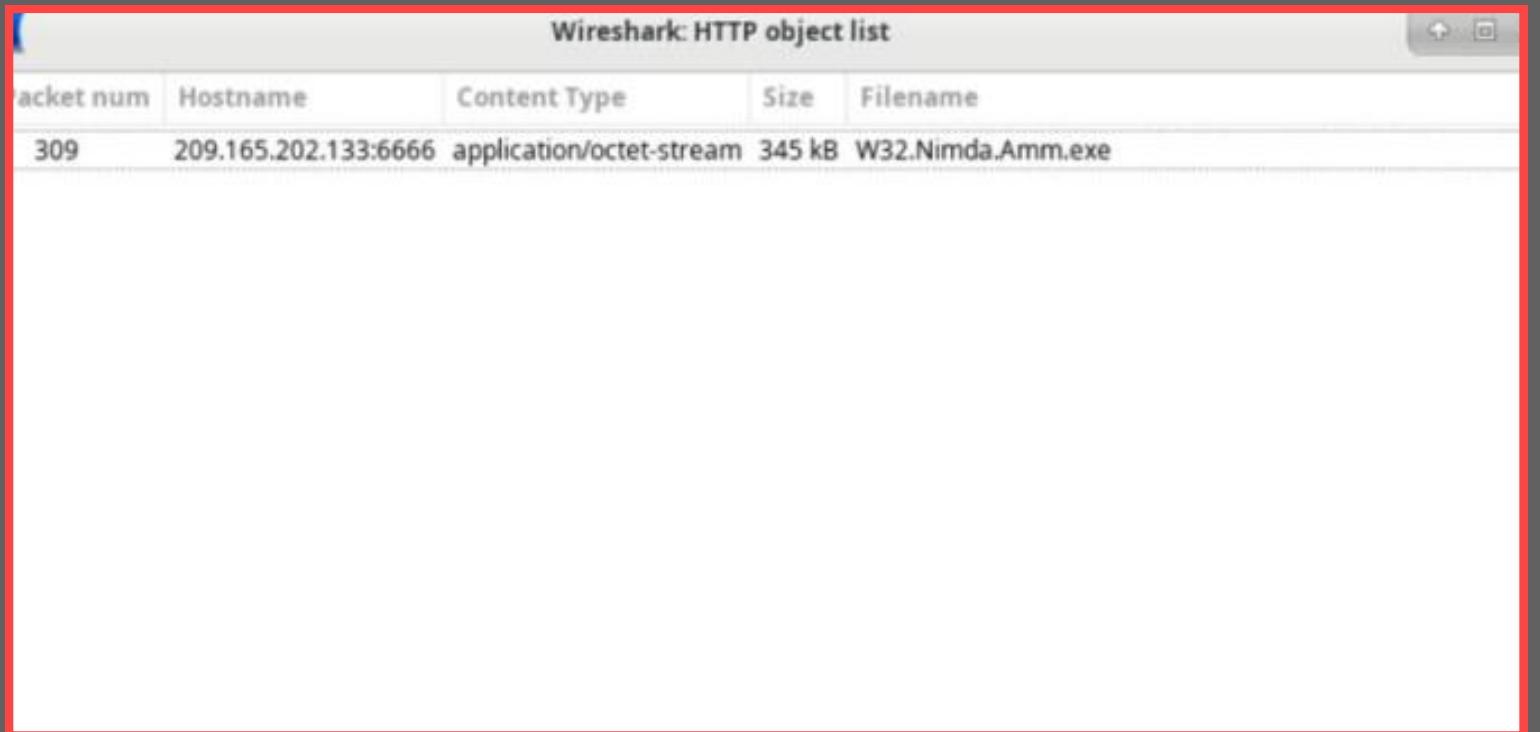
# IDENTITA' DEL FILE

Il file che inizialmente sembrava essere W32.Nimda.Amm.exe è in realtà una versione mascherata di cmd.exe, il prompt dei comandi di Windows. Questo tipo di rinominamento è una tecnica usata per ingannare gli utenti e i sistemi di sicurezza. Un file come cmd.exe può essere utilizzato per eseguire comandi dannosi e ottenere l'accesso al sistema, aumentando così il rischio di compromettere il sistema operativo e di eseguire attività malevole come l'installazione di backdoor.

.00.....h.....(....00.....h.....00.....%.....h...  
.....4..V.S.\_V.E.R.S.I.O.N.\_I.N.F.O.....jD....jD.?.....S.t.r.i.n.g.F.i.l.e.I.n.f.o.....  
0.4.0.9.0.4.B.0...L....C.o.m.p.a.n.y.N.a.m.e....M.i.c.r.o.s.o.f.t .C.o.r.p.o.r.a.t.i.o.n...  
\....F.i.l.e.D.e.s.c.r.i.p.t.i.o.n....W.i.n.d.o.w.s .C.o.m.m.a.n.d .P.r.o.c.e.s.s.o.r...r)...F.i.l.e.V.e.r.s.i.o.n....  
6...1...7.6.0.1...1.7.5.1.4. .(w.i.n.7.s.p.1.\_r.t.m...1.0.1.1.1.9.-1.8.5.0.)....  
(....I.n.t.e.r.n.a.l.N.a.m.e...c.m.d.....L.e.g.a.l.C.o.p.y.r.i.g.h.t.... M.i.c.r.o.s.o.f.t .C.o.r.p.o.r.a.t.i.o.n... A.I.I .r.i.g.h  
.t.s .r.e.s.e.r.v.e.d....8....O.r.i.g.i.n.a.l.F.i.l.e.n.a.m.e...C.m.d...E.x.e...j.  
%...P.r.o.d.u.c.t.N.a.m.e....M.i.c.r.o.s.o.f.t... W.i.n.d.o.w.s... O.p.e.r.a.t.i.n.g .S.y.s.t.e.m....B....P.r.o.d.u.c.t.V.e.r.s.i.  
o.n...6...1...7.6.0.1...1.7.5.1.4....D....V.a.r.F.i.l.e.I.n.f.o....\$....T.r.a.n.s.l.a.t.i.o.n.....J..  
7....0...@..../...!  
8..d..... M.U.I..... M.U.I..... e.n.-U.S.....



# ESPORTAZIONE DEL FILE HTTP



Grazie alla funzione Export Objects > HTTP di Wireshark, il file eseguibile è stato isolato e salvato. Questa funzione ricostruisce il file originale dai dati binari catturati, permettendo agli analisti di esaminarlo direttamente.

Nell'elenco degli oggetti HTTP esportabili, il file W32.Nimda.Amm.exe è l'unico presente. Questo riflette il fatto che il file PCAP è stato catturato in un momento specifico, durante il download del file. Questo approccio dimostra come catture mirate possano fornire informazioni preziose con il minimo impatto sulle risorse di rete. Il sospetto è stato identificato.



# CATTURATO



Una volta esportato il file tramite la funzione Export Objects > HTTP di Wireshark, si è confermato che il file sospetto era W32.Nimda.Amm.exe, un eseguibile per Windows. La cattura PCAP era limitata al momento del download, quindi non conteneva altri file. Dopo l'esportazione, il file è stato verificato tramite il comando file W32.Nimda.Amm.exe, che ha confermato la natura del file come eseguibile Windows. Questo passaggio è cruciale per validare i risultati e garantire l'integrità dell'analisi.

```
File Edit View Terminal Tabs Help
[analyst@secOps ~]$ cd /home/analyst
[analyst@secOps ~]$ ls -l
total 8740
drwxr-xr-x 2 analyst analyst 4096 Mar 22 2018 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22 2018 Downloads
-rw-r--r-- 1 root   root   8420337 Dec 13 12:29 httpdump.pcap
-rw-r--r-- 1 root   root   162562 Dec 13 13:36 httpsdump.pcap
drwxr-xr-x 9 analyst analyst 4096 Jul 19 2018 lab.support.files
drwxr-xr-x 2 analyst analyst 4096 Mar 21 2018 second_drive
-rw-r--r-- 1 analyst analyst 345088 Dec 16 11:16 W32.Nimda.Amm.exe
[analyst@secOps ~]$ 

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

L'intero processo riflette l'importanza dei principi di confidenzialità, integrità e accessibilità (CIA). La confidenzialità protegge i dati sensibili, l'integrità garantisce che non siano stati alterati, e l'accessibilità assicura che le informazioni siano disponibili per analisi e interventi tempestivi. Questi aspetti sono essenziali per la sicurezza di qualsiasi organizzazione, soprattutto in caso di file dannosi che potrebbero minare i sistemi informatici.

# CONCLUSIONI

Questa indagine ha dimostrato l'importanza di strumenti come Wireshark per l'analisi del traffico di rete e il recupero di file sospetti. Attraverso la cattura e l'analisi del flusso TCP, è stato possibile identificare e recuperare un file dannoso, W32.Nimda.Amm.exe, camuffato per eludere i controlli di sicurezza. Un approccio investigativo come questo permette di individuare schemi di attacco, comprendere il comportamento di malware e prevenire compromissioni future.

L'analisi del file non si limita a identificare una singola minaccia ma può aprire la strada a indagini più ampie, come scoprire infrastrutture di comando e controllo o catene di attacco più complesse. I concetti di confidenzialità, integrità e accessibilità (CIA) sono fondamentali in questa prospettiva: proteggere le informazioni sensibili, garantire l'autenticità dei dati e assicurare la loro disponibilità.

Questo caso evidenzia come la sicurezza informatica dipenda dalla capacità di rispondere prontamente e con precisione a eventi sospetti. Un'indagine accurata non solo protegge le risorse aziendali ma contribuisce a rafforzare la resilienza di un'organizzazione contro future minacce, confermando la necessità di una vigilanza continua e di competenze avanzate nell'analisi dei rischi informatici.