BW3- Esercizio 6

Analisi di Rete ed Estrazione Malware da PCAP

Autore: Cybereagles

Sintesi

Il presente report documenta l'attività di ***Network Forensics*** svolta per analizzare la cattura di traffico di rete **nimda.download.pcap**. È stata condotta un'analisi approfondita sui pacchetti per identificare un sospetto download malevolo non cifrato (HTTP).

L'attività ha permesso di estrarre con successo l'eseguibile, inizialmente identificato come **W32.Nimda.Amm.exe**, e di condurre un'analisi statica di base che ha svelato la vera natura dell'artefatto (un file di sistema mascherato per scopi di laboratorio).

Scopo del test e analisi dello scenario

L'attività si svolge in un ambiente isolato e controllato (macchina virtuale Linux ***CyberOps Workstation***). In uno scenario di ***Incident Response***, i log dei sistemi di sicurezza indicano un'anomalia, ma per confermare l'infezione o l'esfiltrazione è necessaria l'analisi del traffico di rete *raw* (**Full Packet Capture**). L'obiettivo è dimostrare le capacità di intercettare transazioni HTTP sospette, isolare i payload trasmessi, estrarli in sicurezza e decodificarne le informazioni statiche primarie per aggirare semplici tecniche di offuscamento (come la falsificazione del nome e dell'estensione del file).

Strumenti

- **Wireshark:** Analizzatore di protocolli di rete utilizzato per l'ispezione dei pacchetti, la ricostruzione dei flussi TCP e l'estrazione degli artefatti trasferiti in chiaro.
- **Linux CLI (strings, file):** Utility da riga di comando per l'estrazione dei metadati e delle stringhe di testo in chiaro dai file binari compilati.

Svolgimento

Fase 1: Preparazione dell'ambiente e caricamento del reperto

L'indagine è iniziata localizzando il file PCAP all'interno della workstation di analisi e aprendolo in un ambiente sicuro (Linux), in modo che l'eventuale malware (scritto per Windows) non potesse in alcun modo infettare il sistema dell'analista.

- **Comando eseguito:** `$ wireshark ~/lab.support.files/pcaps/nimda.download.pcap &`
- **Spiegazione:** Apre direttamente il file in **Wireshark** eseguendo il processo in background (&).
- **Output atteso:** Avvio dell'interfaccia grafica di Wireshark con i pacchetti pre-caricati.

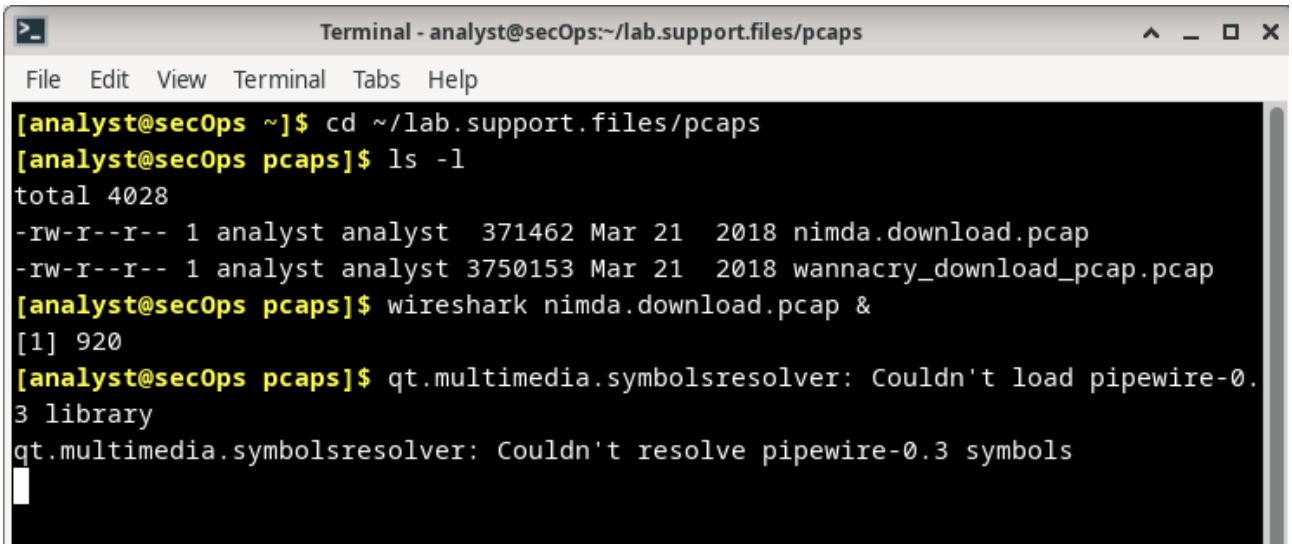

Figura 1 Comando su terminale per l'avvio di Wireshark in background

Fase 2: Analisi del traffico e ricostruzione della sessione

L'analisi è iniziata esaminando il file **nimda.download.pcap** tramite l'interfaccia di **Wireshark**. L'ispezione visiva dei primi frame ha evidenziato il normale completamento del ***Three-way Handshake* TCP** (SYN, SYN-ACK, ACK), seguito immediatamente da una richiesta applicativa in chiaro. Il quarto pacchetto, infatti, contiene una richiesta GET del protocollo HTTP mirata al prelievo della risorsa **/W32.Nimda.Amm.exe**.

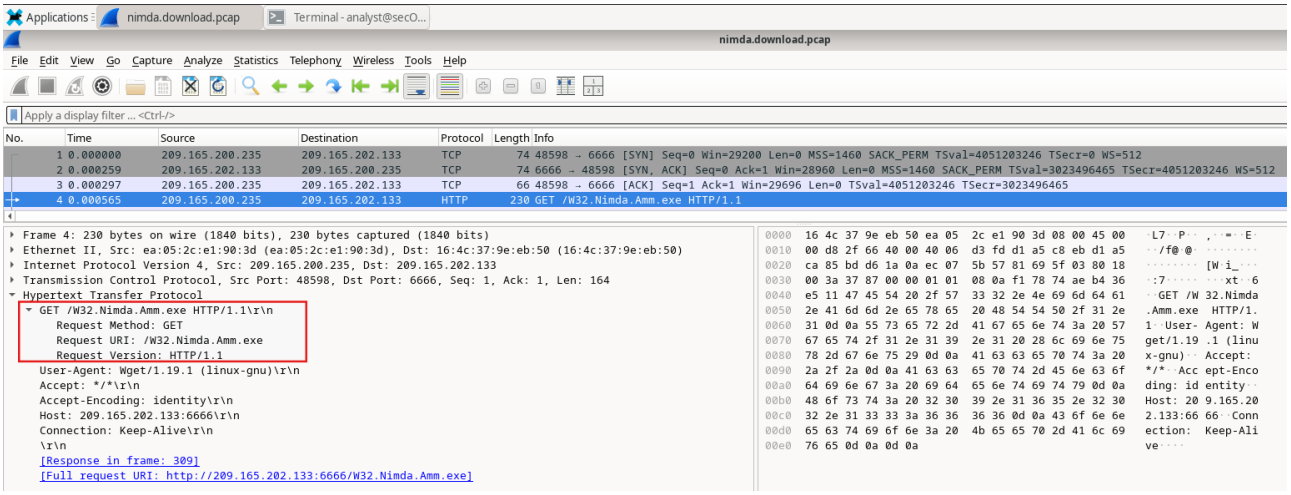

Figura 2 Intercettazione della richiesta HTTP GET per il download dell'eseguibile.

Fase 3: Ispezione del flusso dati (Follow TCP Stream)

Al fine di verificare l'effettivo trasferimento del file, è stata utilizzata la funzione ***Follow TCP Stream***, che permette di aggregare e leggere i payload dei pacchetti in un'unica vista coerente, simulando ciò che il livello applicativo riceve.

La finestra ha restituito la risposta del server (**HTTP/1.1 200 OK**) seguita dal contenuto del file. A una prima ispezione visiva, la finestra risulta colma di simboli incomprensibili intervallati da alcune parole leggibili in lingua inglese. In ambito di analisi di rete, **tali simboli non rappresentano rumore di connessione**, bensì gli effettivi **dati binari** dell'eseguibile trasferito. **Wireshark** tenta di decodificare l'intero flusso di byte in formato di testo ASCII; i byte del file compilato che non corrispondono a caratteri stampabili standard vengono tradotti graficamente come simboli errati o illeggibili.

Nonostante la natura binaria del file, sono presenti **alcune parole leggibili sparse tra i simboli**. Esse sono visibili perché all'interno del formato PE (***Portable Executable***, lo standard per gli eseguibili di Windows) vengono mantenute diverse porzioni di testo non cifrato (**hardcoded**). Nello specifico, si individuano:

- La firma dell'header DOS (MZ) e il messaggio standard *"This program cannot be run in DOS mode"*.
- I nomi delle librerie a collegamento dinamico (es. KERNEL32.dll, USER32.dll) caricate dalla *Import Address Table* (IAT) del programma.
- Eventuali messaggi di errore e metadati inseriti dal programmatore durante la fase di compilazione.

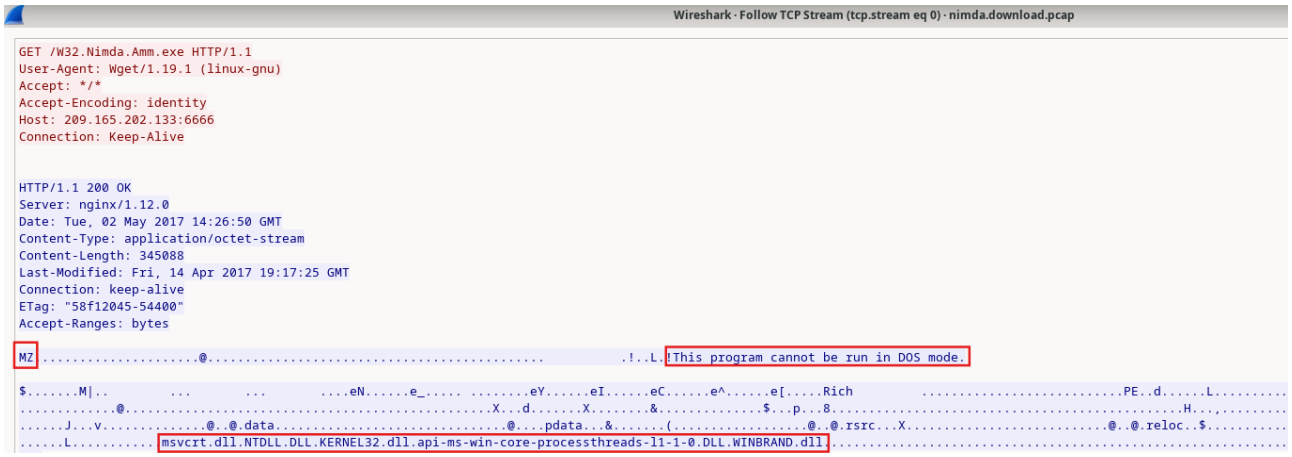

Figura 3 Analisi del flusso TCP con evidenza dell'header del Portable Executable e dei metadati in chiaro.

Fase 4: Estrazione sicura dell'artefatto (Carving)

Avendo accertato la presenza del payload, si è proceduto all'estrazione dello stesso per abilitare le analisi successive, senza dover manipolare manualmente **l'hex dump**. Utilizzando la funzionalità ***File > Export Objects > HTTP*** di **Wireshark**, è apparsa la lista degli oggetti trasferiti.

L'elenco ha presentato un'unica voce. ***W32.Nimda.Amm.exe*** risulta essere **l'unico file nella cattura** per il semplice fatto che il file .pcap in esame è una registrazione del traffico estremamente isolata, generata ad hoc in ambiente di laboratorio per catturare unicamente la sessione (singola richiesta/risposta) pertinente al download del malware, priva di qualsiasi altro traffico di background (es. ARP, DNS, traffico broadcast).

Il file **è stato regolarmente salvato** all'interno della directory `/home/analyst` sulla workstation di indagine, pronto per essere elaborato dagli strumenti a riga di comando locali.

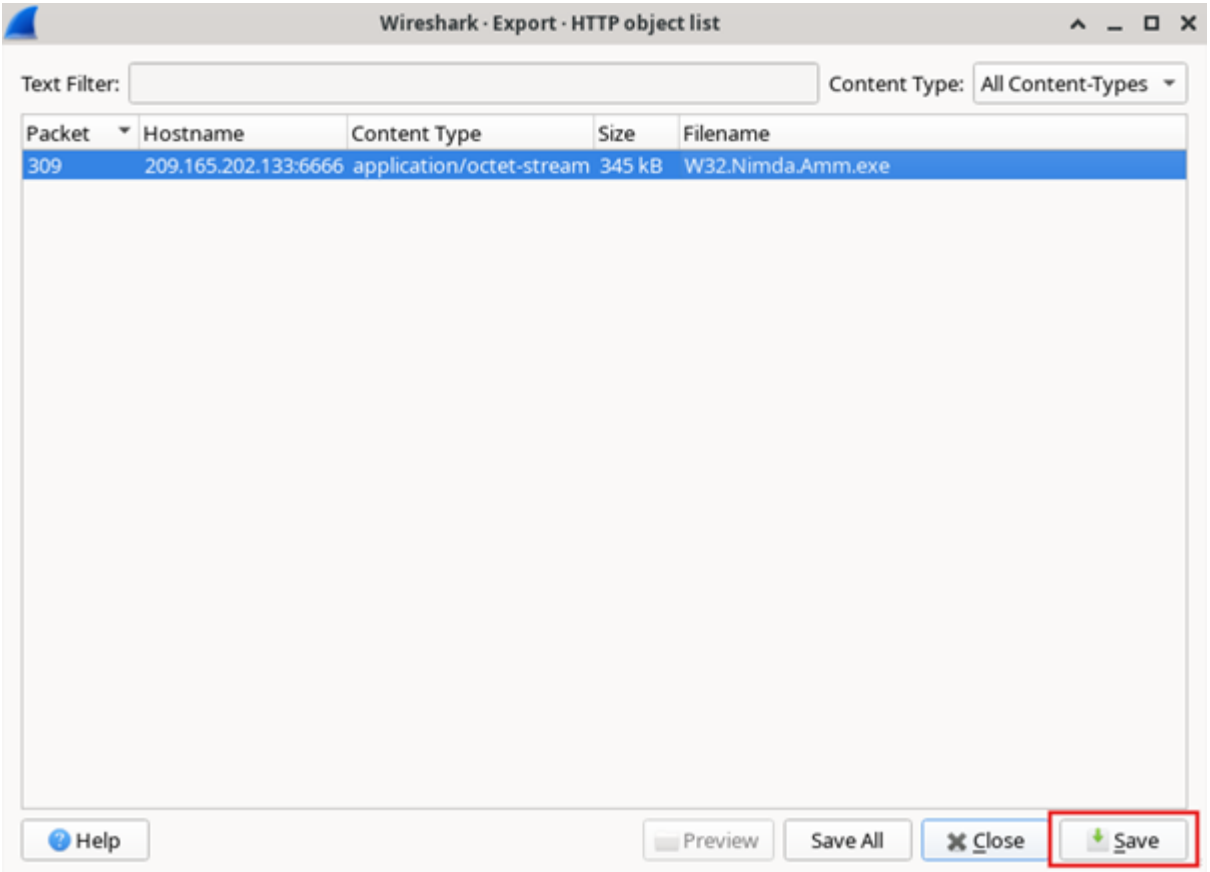

Figura 4 Finestra di esportazione degli oggetti HTTP trasferiti in chiaro.

Fase 5: Identificazione del reale eseguibile (Domanda Sfida)

Per prevenire infezioni accidentali sui sistemi degli analisti, l'eseguibile fornito nel laboratorio non è il reale worm **Nimda**. Sfruttando l'analisi statica dei frammenti di parole visualizzati in chiaro nel flusso TCP (e confermati estraendo le stringhe tramite l'utility Linux strings), è stato possibile determinare la vera identità del file.

Per validare l'artefatto in modo sicuro e riproducibile, in ambiente Linux isolato si ricorre alle seguenti utility da riga di comando:

- **Verifica del tipo di file (ignorando l'estensione):**
    - **Comando eseguito:** `$ file W32.Nimda.Amm.exe`
    - **Spiegazione:** Il comando analizza i ***magic bytes*** dell'header del reperto per determinare la sua vera natura, a prescindere dall'estensione .exe fornita. Si tratta di un'operazione a sola lettura che non esegue il potenziale malware.
    - **Output atteso:** Identificazione dell'artefatto come eseguibile PE32 per MS Windows.

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

Figura 5 Output del comando file

- **Estrazione delle stringhe e analisi della Import Address Table (IAT):**

- **Comando eseguito:** `$ strings W32.Nimda.Amm.exe | grep -iE "WINBRAND|CmdBatNotification"`
- **Spiegazione:** L'utility **strings** estrae il testo in chiaro dal binario. La *pipe* (|) passa l'output a grep per filtrare e isolare rapidamente gli indicatori anomali.

```
[analyst@secOps ~]$ strings W32.Nimda.Amm.exe | grep -iE "WINBRAND|CmdBatNotification"
WINBRAND.dll
WINBRAND.dll
CmdBatNotification
[analyst@secOps ~]$
```

Figura 6 Output del comando strings

Analizzando la tabella delle importazioni (IAT) e i metadati a fine file, sono emersi dettagli inequivocabili:

- **Librerie specifiche:** WINBRAND.dll (con la funzione BrandingFormatString, usata per stampare a video la versione del sistema operativo).
- **Funzioni non standard:** CmdBatNotification e GetVDMCurrentDirectories.
- **Moduli:** cmd.pdb (il file di Program Database associato al compilatore Microsoft).

Questi indicatori statici permettono di affermare con certezza che l'eseguibile è in realtà **cmd.exe** (il processore dei comandi di Windows). Il file originale di Microsoft è stato prelevato e rinominato in **W32.Nimda.Amm.exe** per simulare lo scaricamento di un payload malevolo garantendo però l'incolumità del sistema ospite.

Conclusioni

L'attività ha confermato la vulnerabilità intrinseca dei protocolli in chiaro. L'assenza di crittografia TLS ha permesso di intercettare, leggere e ricostruire perfettamente il binario in transito, che si è rivelato essere un tool di sistema (cmd.exe) spesso scaricato in scenari reali per facilitare l'apertura di *Reverse Shell* sui target compromessi.

Prossimi Passi

Nel reale processo di Incident Response, una volta estratto l'artefatto, il passo successivo per un analista di sicurezza consiste nell'eseguire il Calcolo dell'Hash crittografico per stabilire l'impronta digitale univoca del file.

- **Esecuzione del Calcolo dell'Hash (SHA256):**
  - **Comando (Linux):** `$ sha256sum W32.Nimda.Amm.exe`
  - **Spiegazione:** Genera un'impronta digitale crittografica irreversibile a 256 bit (algoritmo SHA256). Trattandosi di un'impronta univoca, la modifica di un singolo byte nel file produrrebbe un risultato completamente diverso.
  - **Output atteso:** Una stringa alfanumerica di 64 caratteri (es. e3b0c442...) associata al file.

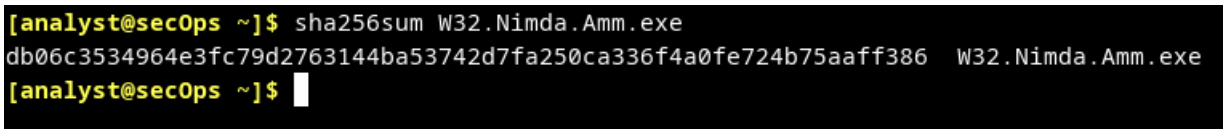

Figura 7 Output comando sha256sum

Questo hash verrebbe poi interrogato su piattaforme di ***Threat Intelligence Open Source (OSINT)***, come **VirusTotal**, per verificare se la community di sicurezza lo abbia già contrassegnato come malevolo.

In assenza di riscontri pubblici, il passo ulteriore prevederebbe il trasferimento del malware in una **Sandbox** (ambiente virtuale ristretto e monitorato) per l'analisi dinamica (***Dynamic Analysis***), allo scopo di osservare le modifiche al sistema operativo e le connessioni di rete generate al momento dell'esecuzione.