# Build Week 3 - ESERCIZIO 6

## Estrarre un eseguibile da un PCAP

# Executive Summary

Nel presente esercizio **è stata analizzata una cattura di traffico di rete (PCAP) contenente il download di un file malevolo tramite protocollo HTTP**. L'obiettivo è stato ricostruire la sessione TCP, identificare la richiesta GET responsabile del trasferimento del file ed estrarre l'oggetto HTTP dalla cattura.

Attraverso l'utilizzo di Wireshark **sono stati osservati il three-way handshake TCP, la richiesta HTTP e il trasferimento del contenuto binario**. Mediante la funzione *Follow TCP Stream* **è stato possibile ricostruire l'intera conversazione tra client e server**, evidenziando la presenza di dati binari riconducibili a un eseguibile Windows.

Infine, tramite la funzione *Export Objects → HTTP*, il file è stato estratto correttamente dal PCAP e identificato come eseguibile Windows (formato PE), confermando la natura del download malevolo.

# Introduzione

L'analisi del traffico di rete rappresenta una competenza fondamentale per un analista di sicurezza, in quanto consente di ricostruire eventi di compromissione, identificare download sospetti e comprendere le modalità operative di un attaccante.

In questo laboratorio **è stata esaminata una cattura PCAP contenente il download di un malware tramite protocollo HTTP**. L'attività ha previsto:

- l'analisi della sequenza TCP (handshake e sessione),
- l'identificazione della richiesta HTTP GET,
- la ricostruzione della comunicazione applicativa,
- l'estrazione dell'oggetto HTTP contenente l'eseguibile,
- la verifica del tipo di file estratto.

L'esercizio dimostra come, anche senza avere accesso diretto al sistema compromesso, sia possibile recuperare e analizzare un file malevolo esclusivamente attraverso l'analisi del traffico di rete.

### Obiettivo

Analizzo una cattura **PCAP** già pronta e ricostruisco una transazione

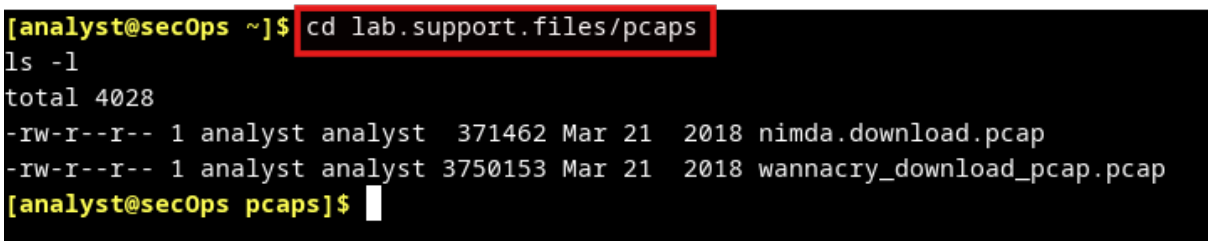
**HTTP su TCP**, poi **estraggo** il file scaricato (rinominato) **W32.Nimda.Amm.exe** dalla cattura.

---

# Parte 1 — Analizzare log e catture di traffico pre-catturati

## a) Entrare nella cartella dei PCAP e listare i file

1. Apro il **Terminale**.
2. Eseguo:
   **cd lab.support.files/pcaps**
   **ls -l**
3. Verifico che tra i file ci sia **nimda.download.pcap**.

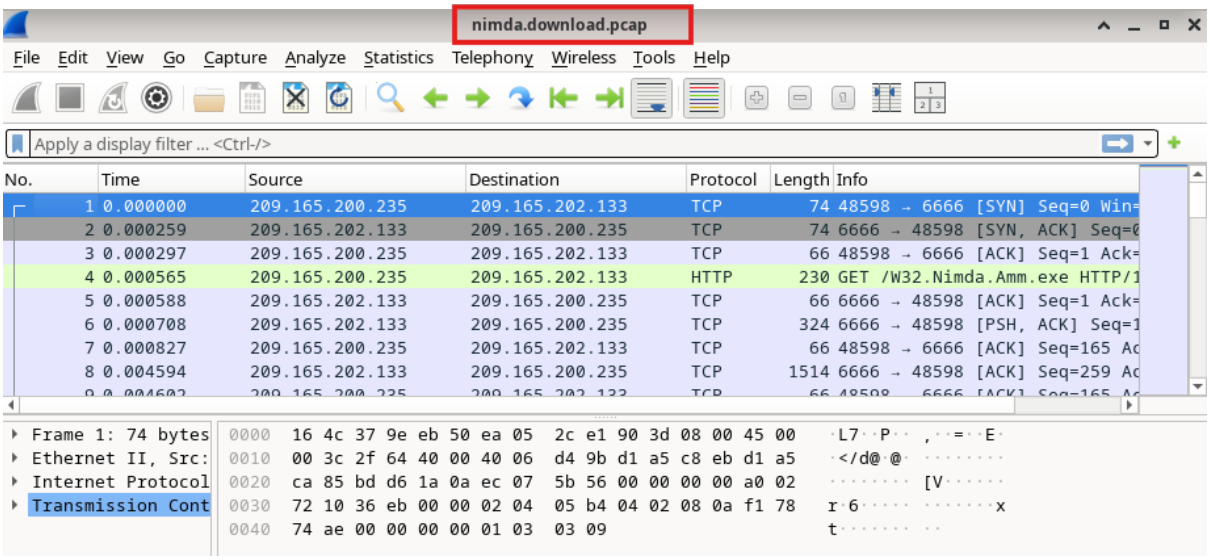



- Terminale con `cd .../pcaps` + output di `ls -l` dove si vede **nimda.download.pcap**.

---

## b) Aprire il PCAP con Wireshark

1. Eseguo:
   **wireshark nimda.download.pcap &**
2. Si apre Wireshark con la cattura caricata.




- Terminale con il comando `wireshark nimda.download.pcap &` (opzionale)
- Wireshark aperto con il file caricato (nome visibile in alto).

---

# c) Selezionare il 4° pacchetto ed espandere HTTP

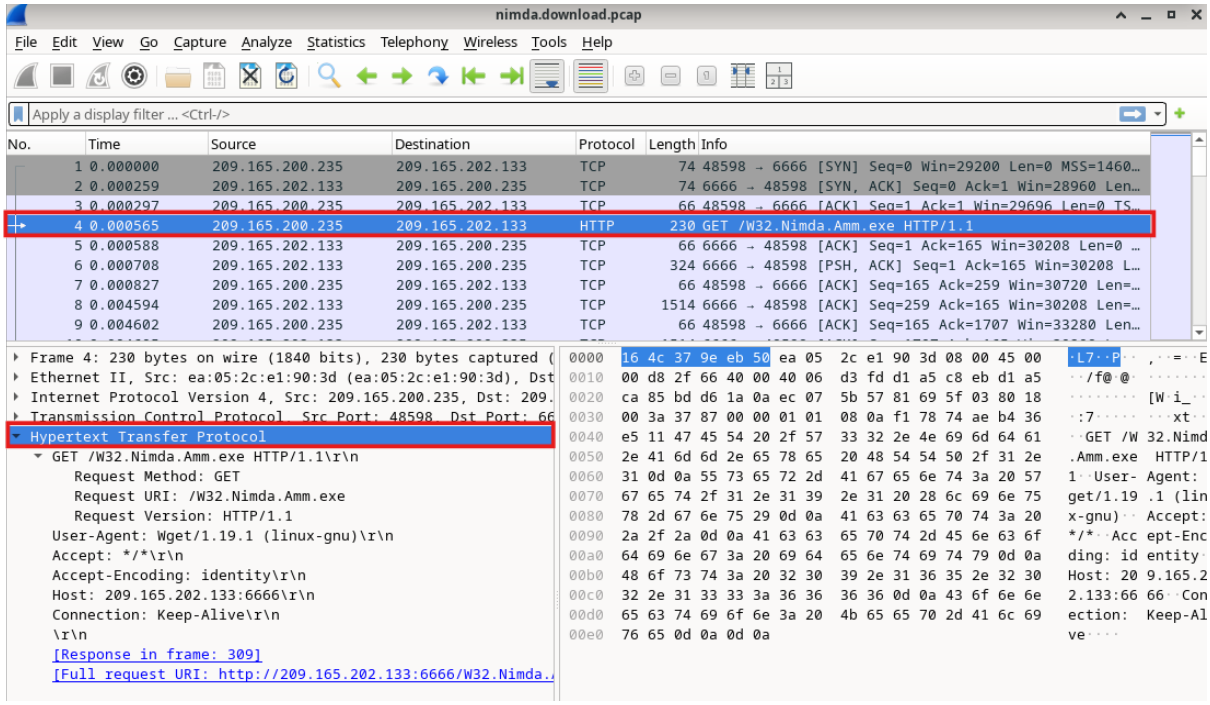
1. In Wireshark, **seleziono il 4° pacchetto** della lista.
2. Nel riquadro di dettaglio, **espando "Hypertext Transfer Protocol"** per vedere i campi HTTP.

- Wireshark con **4° pacchetto evidenziato** + sezione **HTTP espansa**.

---

# d) Interpretazione: handshake TCP + richiesta GET

- I pacchetti **1 → 3** sono l'**handshake TCP**.
- Il **4° pacchetto** è la richiesta del file malware via **HTTP GET**.




- Wireshark con i **primi 4 pacchetti visibili** (si devono intuire i primi 3 come handshake e il 4° come richiesta).

---

# e) Ricostruire la sessione: Follow TCP Stream

1. Seleziono il **primo pacchetto TCP** (un **SYN**).
2. Tasto destro → **Follow** → **TCP Stream**.

---

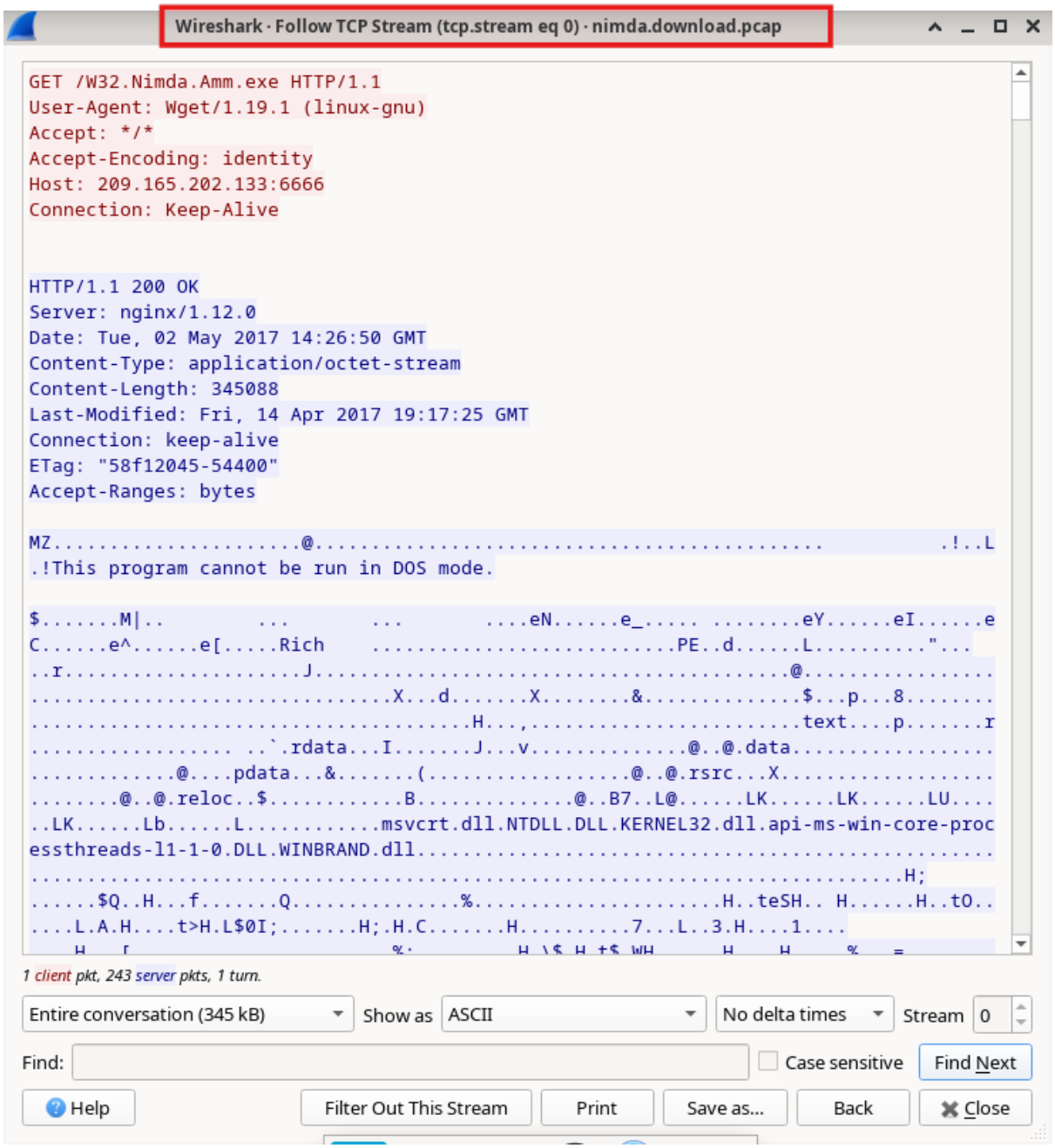
# f) Domande + Risposte

## DOMANDA 1

**“Cosa sono tutti quei simboli mostrati nella finestra Follow TCP Stream? Sono rumore di connessione? Dati? Spiega.”**

**RISPOSTA**
Sono **dati dell’applicazione trasportati nella sessione TCP**. In una ricostruzione “stream”, Wireshark mostra **l’intero contenuto scambiato**:

- la parte leggibile (es. header HTTP / testo)
- e la parte **binaria** (eseguibile scaricato), che appare come simboli “strani” perché **non è testo**, ma byte del file.




- Finestra **Follow TCP Stream** con la parte “illeggibile” ben visibile.

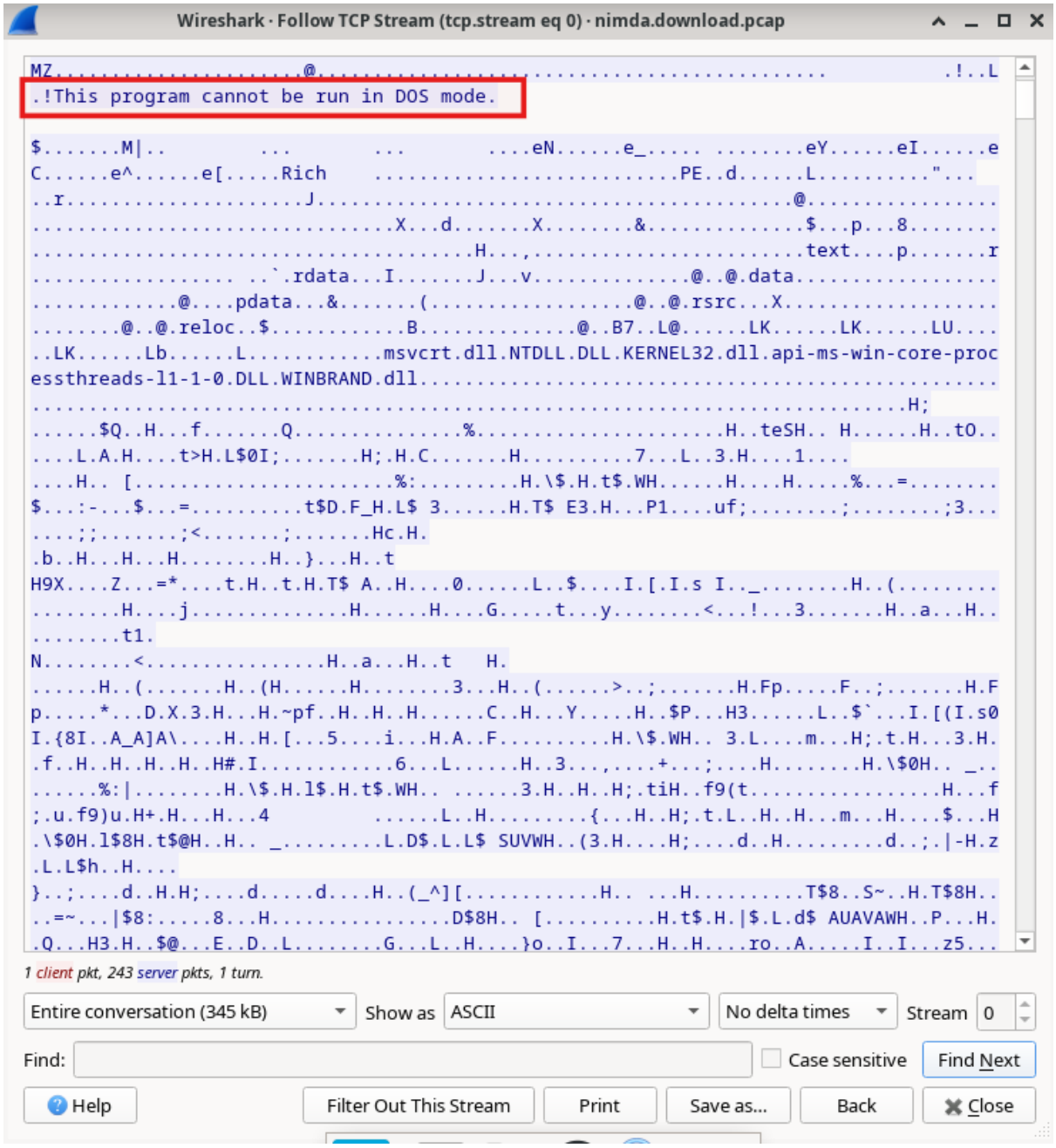
---

## DOMANDA 2

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

**RISPOSTA**
Perché nello stream ci sono porzioni che sono **testo in chiaro**, ad esempio:

- **header HTTP** (metadati della risposta/ richiesta),
- oppure **stringhe interne** presenti nel binario (alcuni eseguibili contengono stringhe ASCII/Unicode, nomi di sezioni, messaggi, percorsi, ecc.).
  Quindi “emergono” parole leggibili anche in mezzo a byte binari.




- Follow TCP Stream dove si notano **parole leggibili** in mezzo ai simboli.

---

## DOMANDA SFIDA

**“Usando i frammenti di parole visualizzati dalla finestra Follow TCP Stream, puoi dire quale eseguibile sia realmente?”**

**RISPOSTA**
Dalla sola finestra **Follow TCP Stream** si può **ipotezzare** la natura del file (eseguibile Windows/PE), ma **identificare con certezza quale programma specifico sia davvero** spesso **non è affidabile** solo “a occhio”, perché:

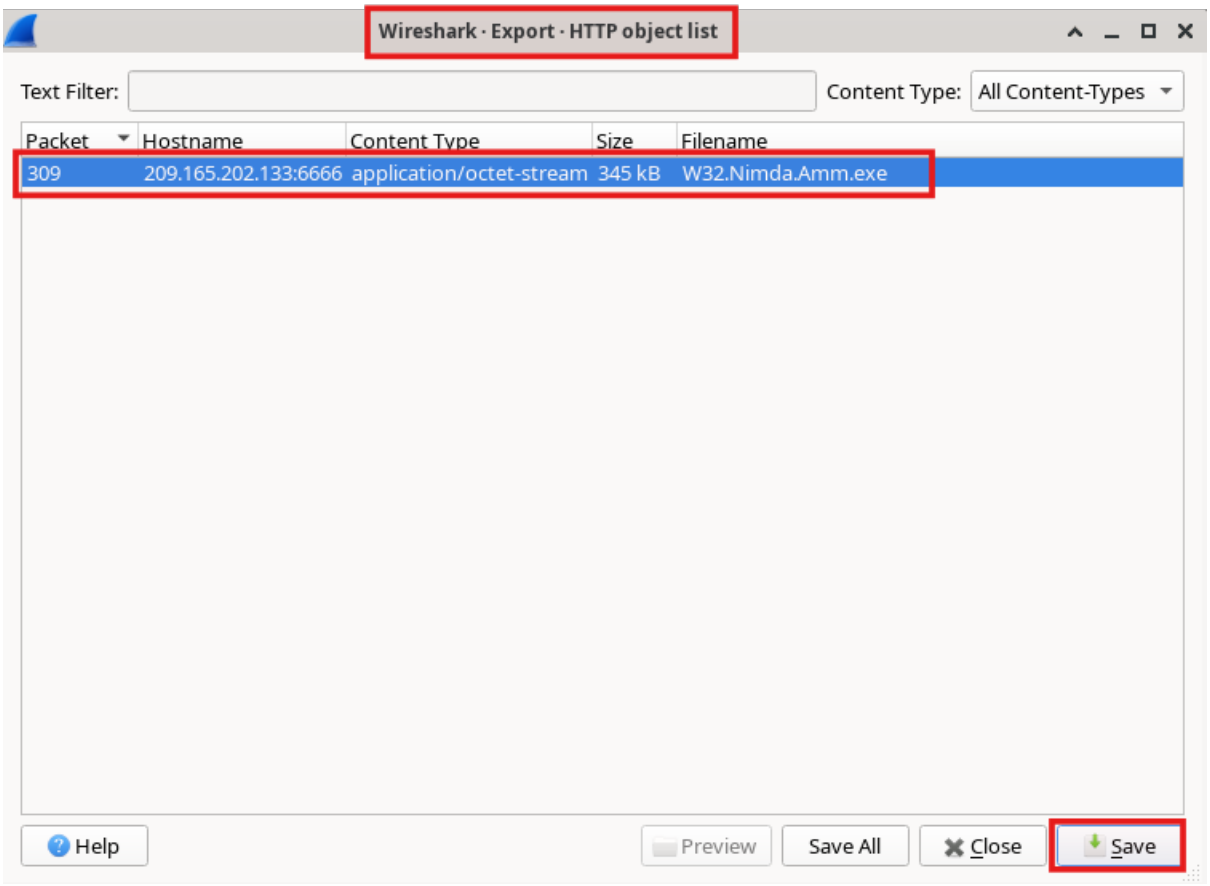
- i frammenti possono essere incompleti,

- molte stringhe sono comuni tra eseguibili diversi,
- l'eseguibile può essere compresso/obfuscato.

## Estrarre il file dal PCAP (Export Objects → HTTP)

1. In Wireshark andare su **File** → **Export Objects** → **HTTP**.
2. Aprirela lista degli oggetti HTTP.
3. Selezionare **W32.Nimda.Amm.exe**.
4. Cliccare **Save As**.
5. Salvare in:
    - **Home** → **analyst** oppure **Desktop** (`/home/analyst/`)

- Finestra **HTTP object list** con `W32.Nimda.Amm.exe` selezionato.
- Finestra di salvataggio **Save As** con percorso visibile (Home/analyst o Desktop) e nome file.



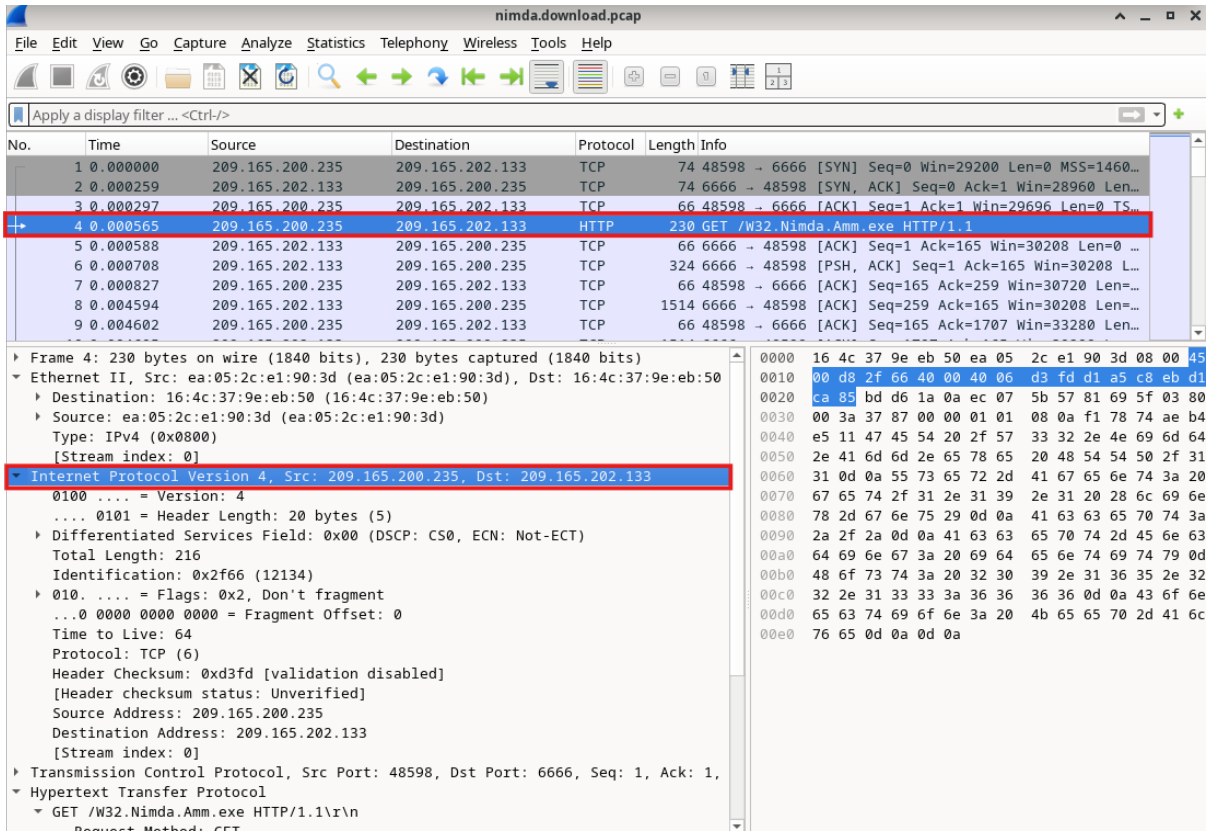
## g) Chiudere Follow TCP Stream e tornare al PCAP

1. Clicco **Close** nella finestra Follow TCP Stream.
2. Torno alla schermata principale di Wireshark con i pacchetti.

---

# Parte 2: Identificare la richiesta GET (IP sorgente/destinazione)

1. Torno al **4° pacchetto** (quello della GET).
2. Noto che la richiesta HTTP GET è stata generata da **209.165.200.235** verso **209.165.202.133** e la colonna “Info” indica la richiesta del file.




- Wireshark con il pacchetto GET selezionato e gli IP visibili (o in “Info” o nel dettaglio IP/HTTP).

---

## c) Domanda + risposta

### DOMANDA

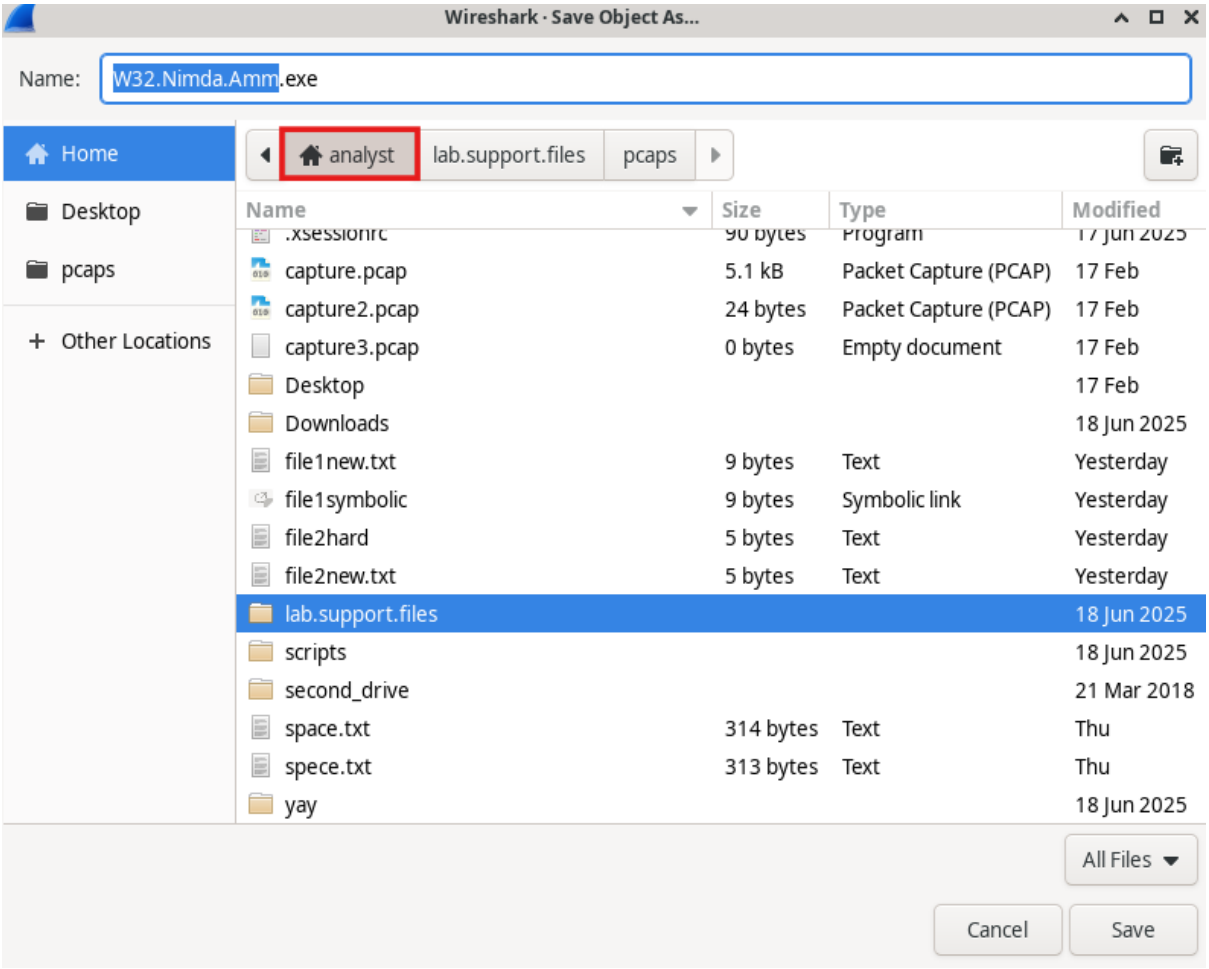
**“Perché W32.Nimda.Amm.exe è l’unico file nella cattura?”**

**RISPOSTA**
Perché questo PCAP è focalizzato su **una singola transazione HTTP di download**: nella sessione catturata risulta presente **solo l’oggetto HTTP corrispondente al file richiesto con GET**. Non ci sono (in quella cattura) altri download HTTP completi/oggetti ricostruibili nello stesso flusso.

---

# d–e) Salvare l’oggetto sul desktop/home dell’utente analyst

1. Nella “HTTP object list” seleziono **W32.Nimda.Amm.exe**.
2. Clicco **Save As**.
3. Navigo con la freccia indietro fino a vedere **Home**.
4. Clicco **Home** → cartella **analyst** (la cartella, non la “scheda”) → **Save**.




- HTTP object list con file selezionato + pulsante **Save As**

- Finestra di salvataggio con percorso **Home → analyst**.

---

# f) Verifica: il file è stato salvato?

1. Torno al terminale:
   **cd /home/analyst**
   **ls -l**
2. Controllo che compaia `W32.Nimda.Amm.exe`.

### DOMANDA

**“Il file è stato salvato?”**

**RISPOSTA**
Sì: se `ls -l` mostra `W32.Nimda.Amm.exe` nella directory `/home/analyst`, allora il salvataggio è riuscito.

```
drwxr-xr-x 2 analyst analyst 4096 Mar 21  2018 second_drive
-rw-r--r-- 1 analyst analyst  314 Feb 19 08:59 space.txt
-rw-r--r-- 1 analyst analyst  313 Feb 19 08:53 spece.txt
drwxr-xr-x 5 analyst analyst 4096 Jun 18  2025 yay
[analyst@secOps ~]$ cd /home/analyst
ls -l
total 396
-rw-r--r-- 1 root    root        24 Feb 17 10:11 capture2.pcap
-rw-r--r-- 1 root    root         0 Feb 17 10:33 capture3.pcap
-rw-r--r-- 1 root    root      5113 Feb 17 10:27 capture.pcap
drwxr-xr-x 3 analyst analyst   4096 Feb 17 09:25 Desktop
drwxr-xr-x 3 analyst analyst   4096 Jun 18  2025 Downloads
-rw-r--r-- 1 analyst analyst      9 Feb 23 10:19 file1new.txt
lrwxrwxrwx 1 analyst analyst      9 Feb 23 10:22 file1symbolic -> file1.txt
-rw-r--r-- 2 analyst analyst      5 Feb 23 10:20 file2hard
-rw-r--r-- 2 analyst analyst      5 Feb 23 10:20 file2new.txt
drwxr-xr-x 9 analyst analyst   4096 Jun 18  2025 lab.support.files
drwxr-xr-x 3 analyst analyst   4096 Jun 18  2025 scripts
drwxr-xr-x 2 analyst analyst   4096 Mar 21  2018 second_drive
-rw-r--r-- 1 analyst analyst    314 Feb 19 08:59 space.txt
-rw-r--r-- 1 analyst analyst    313 Feb 19 08:53 spece.txt
-rw-r--r-- 1 analyst analyst 345088 Feb 24 04:29 W32.Nimda.Amm.exe
drwxr-xr-x 5 analyst analyst   4096 Jun 18  2025 yay
```

- Terminale con `ls -l` dove si vede `W32.Nimda.Amm.exe`.

---

# g) Identificare il tipo file con `file`

1. Eseguo:
   **file W32.Nimda.Amm.exe**
2. L'output atteso indica che è un **eseguibile Windows** (formato PE).


- Terminale con comando `file W32.Nimda.Amm.exe` e output.

---

# Domanda finale + risposta

## DOMANDA

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

**RISPOSTA**
Un passo successivo tipico è passare a un'analisi più approfondita, ad esempio:

- calcolare **hash (SHA256/MD5)** e fare **reputation check** (VirusTotal/DB interni),
- eseguire **strings** per estrarre indicatori (URL, IP, percorsi, nomi DLL),
- fare **analisi statica** (PE headers/imports) e, se previsto, **analisi dinamica in sandbox/VM isolata** con monitoraggio processi/rete.

(Questo permette anche di rispondere in modo "definitivo" alla *Domanda Sfida* sull'identità reale del file.)

## Conclusioni

L'analisi del file **nimda.download.pcap** ha permesso di ricostruire correttamente una sessione **HTTP** su **TCP** e di identificare il download di un eseguibile malevolo. Attraverso **Wireshark** sono stati riconosciuti l'handshake TCP, la richiesta HTTP GET e il trasferimento del contenuto binario.

Mediante la funzione **Follow TCP Stream** è stata ricostruita l'intera comunicazione applicativa, distinguendo tra header HTTP in chiaro e dati binari dell'eseguibile. Successivamente, tramite **Export Objects - HTTP,** il file W32.Nimda. Amm.exe è stato estratto con successo e verificato come eseguibile Windows (formato PE).

L'esercizio dimostra come l'analisi del traffico di rete consenta di individuare e recuperare malware anche senza accesso diretto al sistema compromesso, evidenziando l'importanza della network forensics nelle Fonti