PROGETTO S10/L5

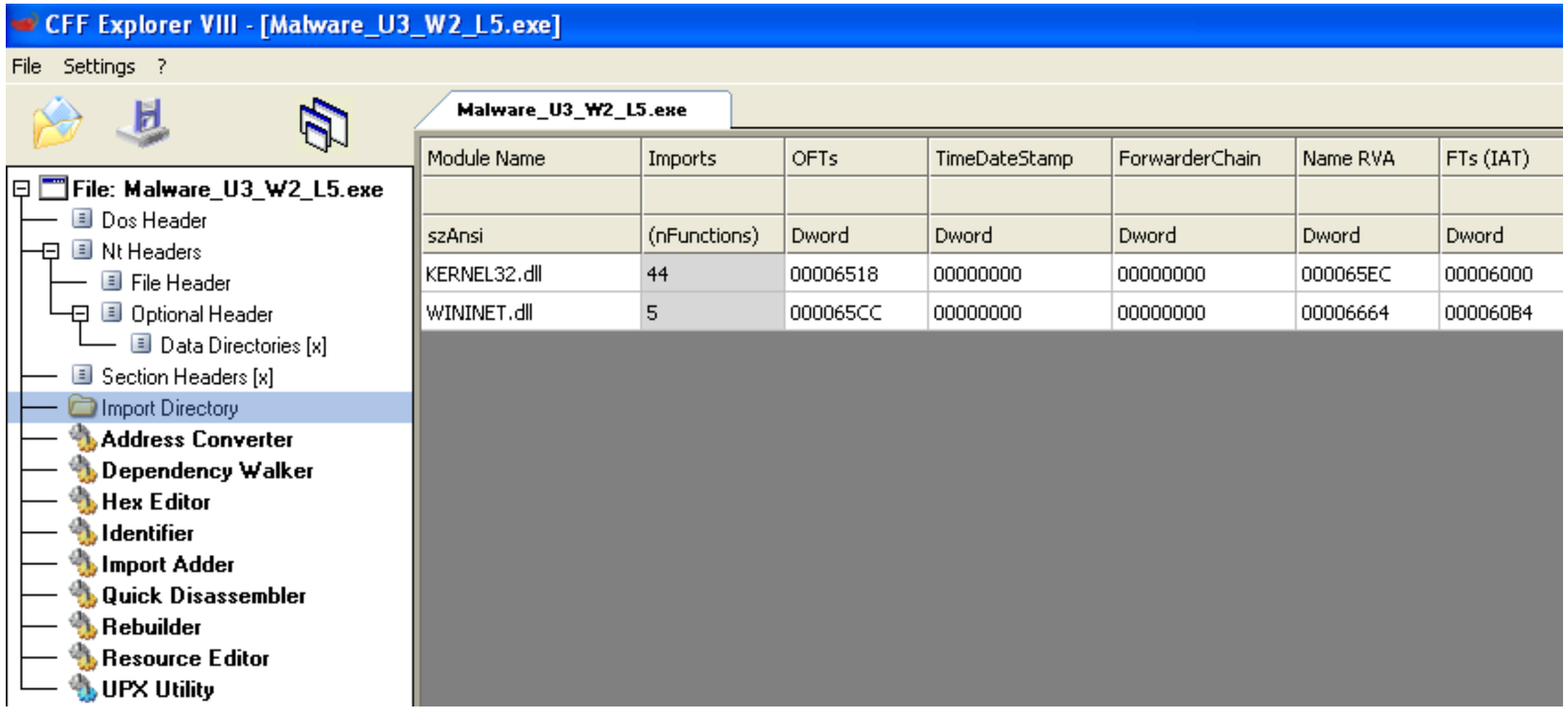
BENEDETTA FORESTIERI

LIBRERIE

Con riferimento al file Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5» sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere al seguente quesito:

- **Quali librerie vengono importate dal file eseguibile?**

Per esaminare le librerie importate, avvio il CFF Explorer e carico il file interessato. Navigando nella sezione "Import directory" identifico le librerie coinvolte nel processo.



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

Le librerie coinvolte sono KERNEL32.dll e WININET.dll.

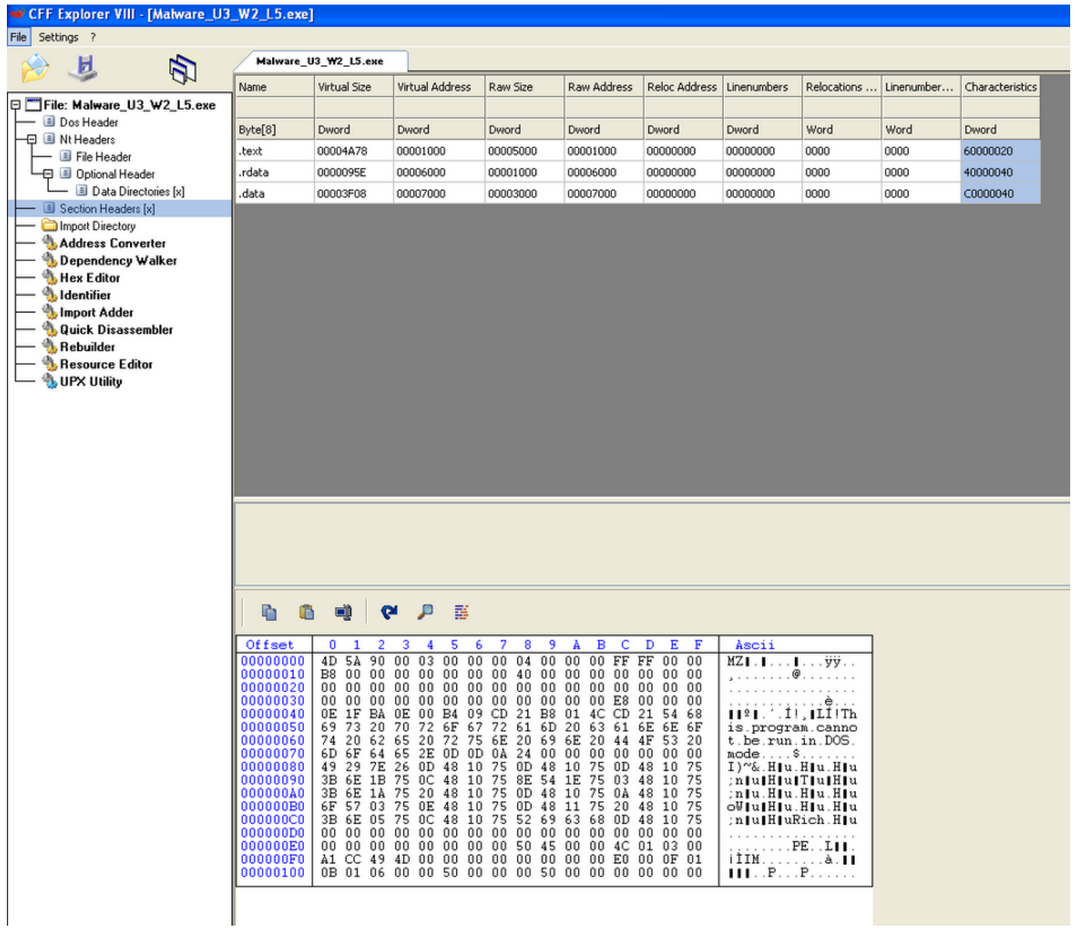
KERNEL32.dll: svolge un ruolo fondamentale nell'interazione con il sistema operativo, gestendo funzioni essenziali come la gestione dei file e dei processi.

WININET.dll: facilita l'accesso a Internet, fornendo funzionalità per comunicazioni di rete come l'utilizzo dei protocolli HTTP e FTP.

SEZIONI

Con riferimento al file Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5» sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere al seguente quesito:

- **Quali sono le sezioni di cui si compone il file eseguibile del malware?**



Nella sezione "section headers" del pannello a sinistra del CFF Explorer, trovo le informazioni sulle sezioni del file. Esaminando il pannello principale a destra, identifico le tre sezioni principali di questo eseguibile: .text, .rdata e .data.

La sezione **.text** comprende le istruzioni di codice che la CPU elabora all'avvio del software. Generalmente, questa sezione è la sola direttamente eseguita dalla CPU, mentre le altre sezioni contengono dati o supporto informativo.

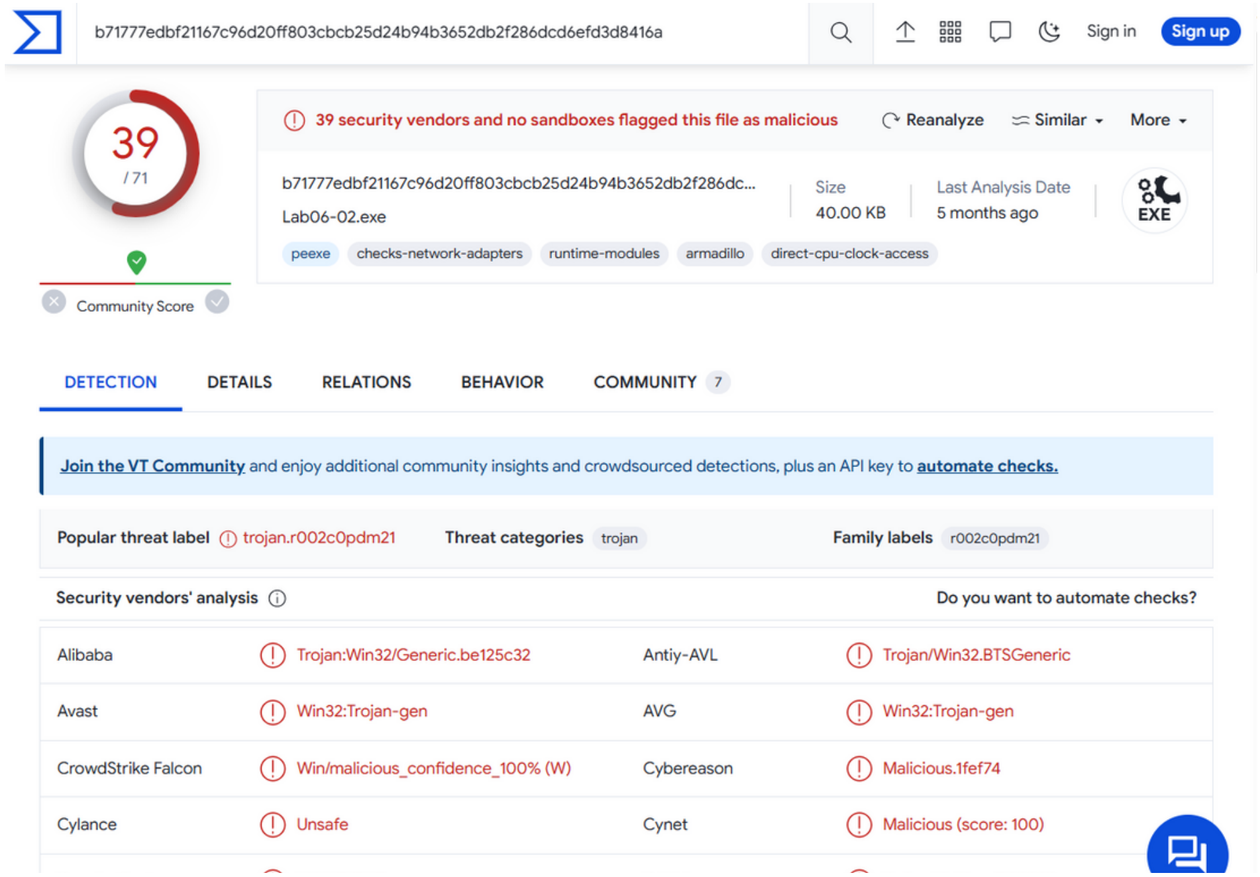
La sezione **.rdata** include informazioni riguardanti le librerie e le funzioni importate o esportate dall'eseguibile.

La sezione **.data** contiene dati e variabili globali utili nel programma eseguibile, essenziali per il funzionamento globale dell'applicazione. Le variabili definite a livello globale risultano accessibili da qualsiasi funzione all'interno dell'eseguibile, poiché non sono limitate al contesto di una singola funzione.

VIRUSTOTAL

Nella valutazione di un possibile malware, la prima tappa consiste nell'accertarsi della sua natura malevola. Ogni file possiede una firma unica, incluso il malware. Per determinare se ci troviamo di fronte a un malware, è possibile consultare i database degli antivirus per verificare se la firma di quel particolare malware è già registrata. Piattaforme come VirusTotal consentono di valutarne la reputazione basandosi su diversi software antivirus..

Dalla sezione "Dependency Walker" di CFF Explorer possiamo risalire all'ash MD5 del malware (in alternativa, si può calcolare con Md5deep, tool da riga di comando).



Inserendo l'hash su VirusTotal, si evidenzia che il malware è un **Trojan**, si mimetizza da software affidabile per infiltrarsi nel sistema senza attirare l'attenzione. Una volta introdotto, può sottrarre dati, compromettere il funzionamento del sistema o consentire accessi non autorizzati. Attraverso quest'inganno da software legittimo e sfruttando varie vie come email, link sospetti o vulnerabilità, rappresenta una minaccia rilevante e ampia per la sicurezza informatica.

Con l'aiuto di VirusTotal, è possibile accedere a una vasta gamma di dettagli riguardanti il malware, tra cui segnalazioni da parte di altri utenti, le librerie e le sezioni importate.

Sections

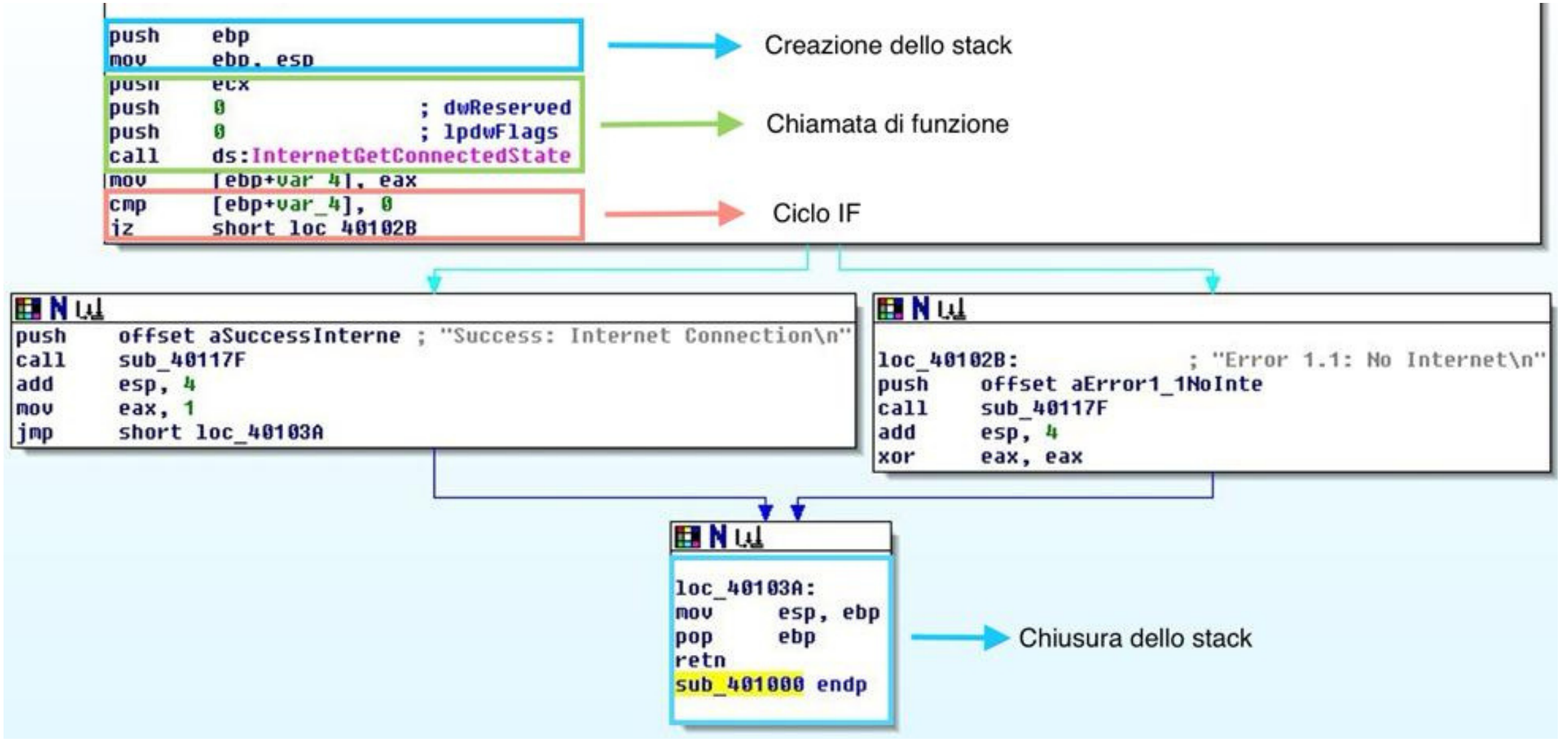
Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	19064	20480	6.37	4b8aaeb128744c00b1f9b29dd120616e	196535.5
.rdata	24576	2398	4096	3.66	e5e39acc53e64c50fa5a35693a911478	304856
.data	28672	16136	12288	0.7	305514f6ece00473b7ff8bc023f57e15	2765274

Imports

- KERNEL32.dll
- WININET.dll

ASSEMBLY

- **Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)**



Con riferimento alla figura, **ipotizzare il comportamento della funzionalità implementata.**

Il codice **esamina lo stato della connessione Internet** e agisce di conseguenza in due possibili situazioni:

1. **Connessione attiva**: Se la connessione è attiva, viene eseguita la sezione di codice associata al successo della connessione. Questa parte del codice stampa il messaggio "Success: Internet Connection" e imposta una variabile specifica (in questo caso eax) a 1. Dopo di ciò, viene eseguita un'operazione di pulizia per ripristinare lo stato precedente prima di ritornare al flusso principale del programma.
2. **Connessione non attiva**: Nel caso in cui non ci sia connessione, il codice gestisce questa situazione eseguendo la parte di codice associata all'errore di connessione. Questa sezione stampa il messaggio "Error 1.1: No Internet" e annulla il valore di una variabile (in questo caso eax). Anche in questo caso, viene eseguita un'operazione di pulizia per ripristinare lo stato precedente prima di tornare al flusso principale del programma.

ANALISI

```
push    ebp
mov     ebp, esp
push    ecx
push    0                       ; dwReserved
push    0                       ; 1pdwFlags
call    ds: InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

Salva il valore corrente del registro base dello stack (ebp) nello stack.
Preparazione del registro base dello stack (ebp) per stabilire un nuovo frame nello stack.
Mette nel registro ecx un valore nello stack.
Inserisce due valori O nello stack
Chiama la funzione InternetGetConnectedState per verificare lo stato della connessione Internet.
Salva il risultato della chiamata nella variabile [ebp+var_4]
Confronta il valore memorizzato in [ebp+var_4] con 0 per verificare lo stato della connessione.
Se il valore in [ebp+var_4] e uguale a 0, salta a loc_40102B, probabilmente gestendo il caso di mancanza di connessione.

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jnp     short loc_40103A
```

Carica l'indirizzo di memoria della stringa "Success: Internet Connection\n" nello stack
Chiama la funzione sub_40117F
Libera 4 bytes dallo stack
Imposta il valore di eax a 1
Salta a loc_40103A

```
loc_40102B:             ;"Error1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

Inizio del blocco di codice per il caso di mancanza di connessione
Carica l'indirizzo di memoria della stringa "Error1.1: No Internet\n" nello stack.
Chiama la funzione sub_40117F (potrebbe gestire l'output o il log dell'errore)
Libera 4 bytes dallo stack (ripulitura dopo la chiamata della funzione)
Esegue un'operazione XOR sul registro eax, impostandolo a zero (0)

```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

Ripristina il registro esp al valore del registro ebp (pulisce il frame dello stack corrente)
Ripristina il valore precedente del registro ebp
Ritorna dalla funzione
Fine della procedura