MALWARE ANALYSIS

TASK 1: IDENTIFICARE LE LIBRERIE IMPORTATE DAL FILE ESEGUIBILE

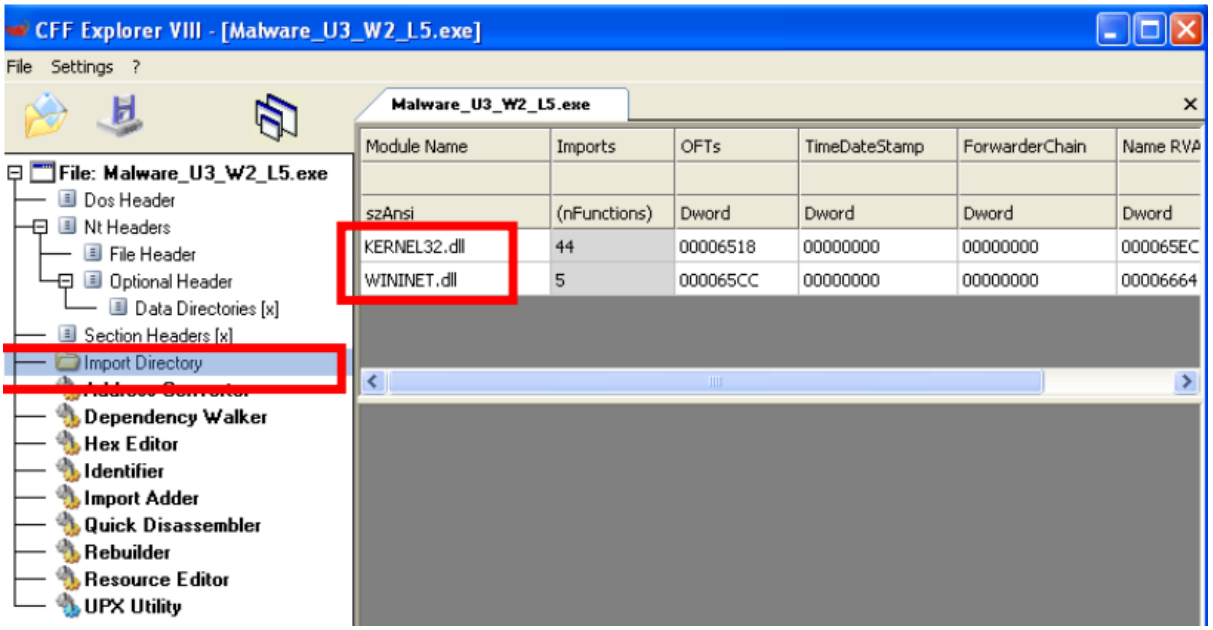
Dato il file eseguibile oggetto d'interesse è Malware_U3_W2_L5.exe, iniziamo un processo di Malware Analysis sullo stesso per indagare e studiare il comportamento.

MALWARE ANALYSIS è una procedura complessa che coinvolge l'insieme di competenze e tecniche utilizzate dagli esperti di sicurezza informatica per indagare accuratamente un malware al fine di studiare e capire esattamente il suo comportamento per poi rimuoverlo dal sistema. Queste competenze sono di vitale importanza per i membri tecnici del CSIRT (Computer Security Incident Response Team) durante la gestione degli incidenti di sicurezza.

Per queste prime task ci serviremo dell'ANALISI STATICA BASICA, che consiste nell'esaminare un file eseguibile senza tener conto delle istruzioni che lo compongono, al fine di:
• confermare la natura malevola o meno del file oggetto di interesse
• fornire informazioni generiche circa le sue funzionalità

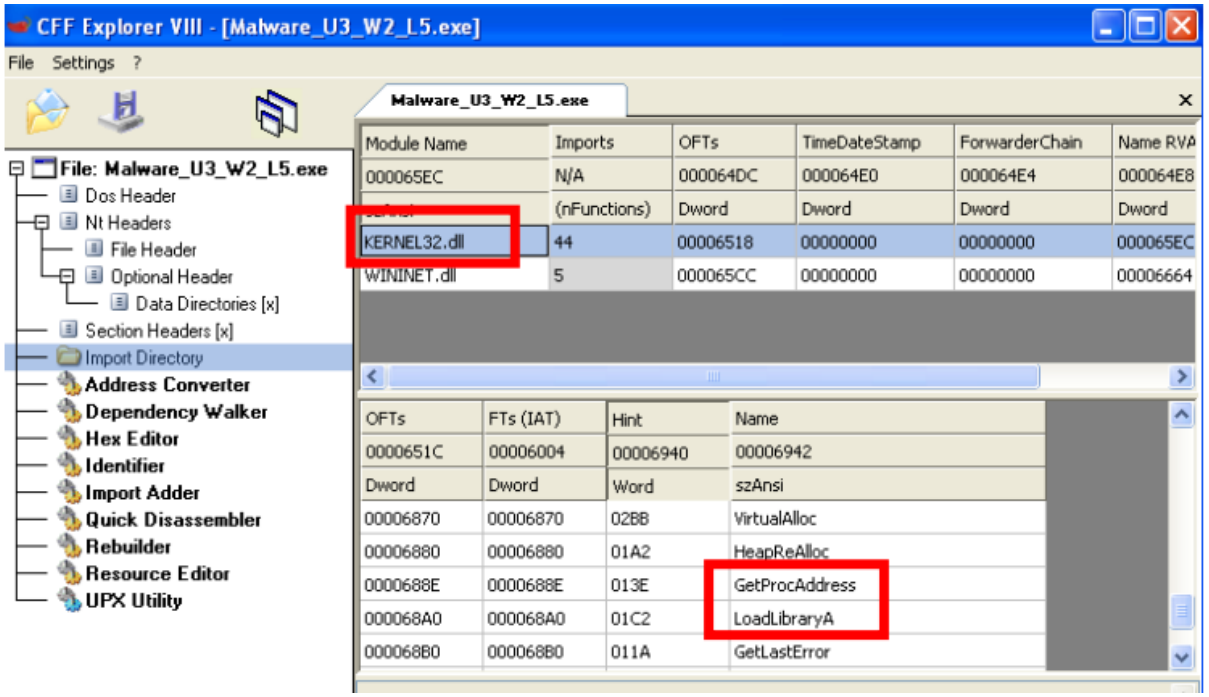
A tal proposito, ci serviremo del tool CFF EXPLORER VIII, il quale ci consente di caricare un file eseguibile (nel nostro caso Malware_U3_W2_L5.exe) per analizzarne l'HEADER.

Selezionando la tab IMPORT DIRECTORY, il tool ci restituisce librerie importate nel malware:
• kernel32.dll: libreria usata per le funzioni principali per interagire con il sistema operativo. Un malware potrebbe sfruttare tale libreria per manipolare i file e per accedere la gestione della memoria
• wininet.dll: libreria che contiene le funzioni per l'implementazione di alcuni protocolli di rete come http, FTP, NTP. Un malware potrebbe utilizzare le funzioni presenti nella libreria per comunicare con server remoti, trasferire file o inviare dati sensibili senza il consenso dell'utente

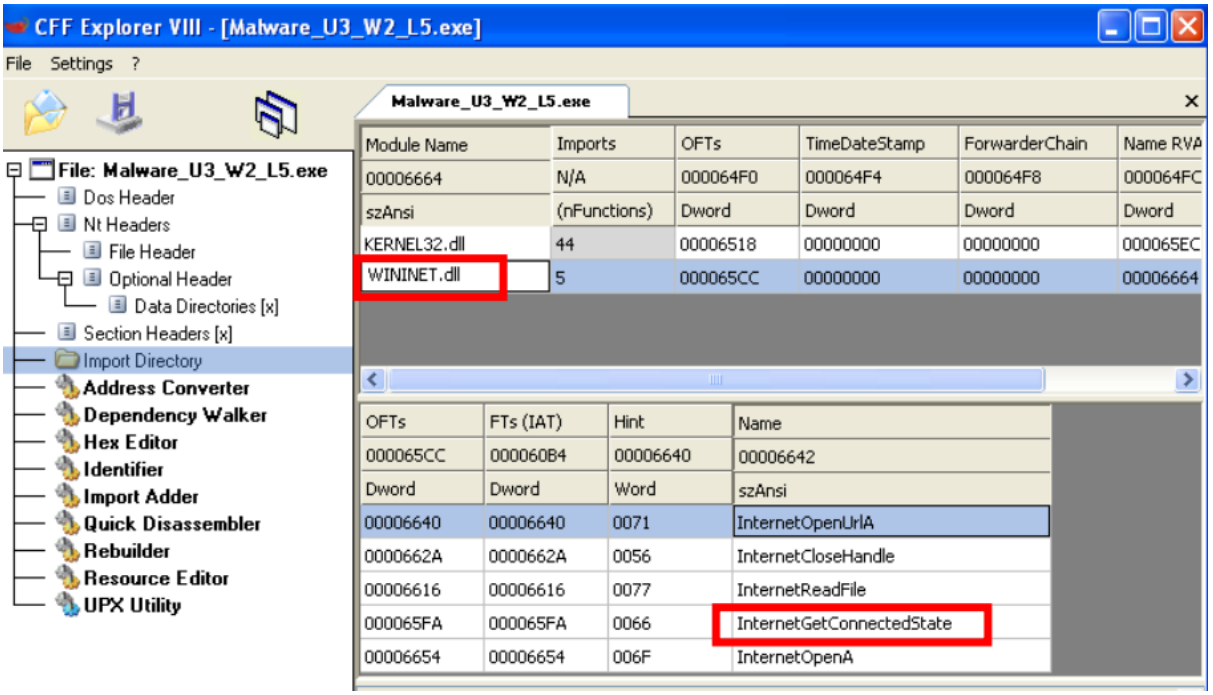


Indagando nello specifico, selezionando la libreria KERNEL 32.DLL, è possibile notare che al suo interno vi sono le funzioni LoadLibraryA e GetProcAddress, che permettono di importare le funzioni della libreria a tempo di esecuzione (runtime). Ciò significa che l'eseguibile richiama la libreria solo quando ha bisogno di utilizzare una specifica funzione.

Questo è un comportamento tipico dei malware, i quali, attraverso questo meccanismo, cercano di risultare meno invasivi e rilevabili.

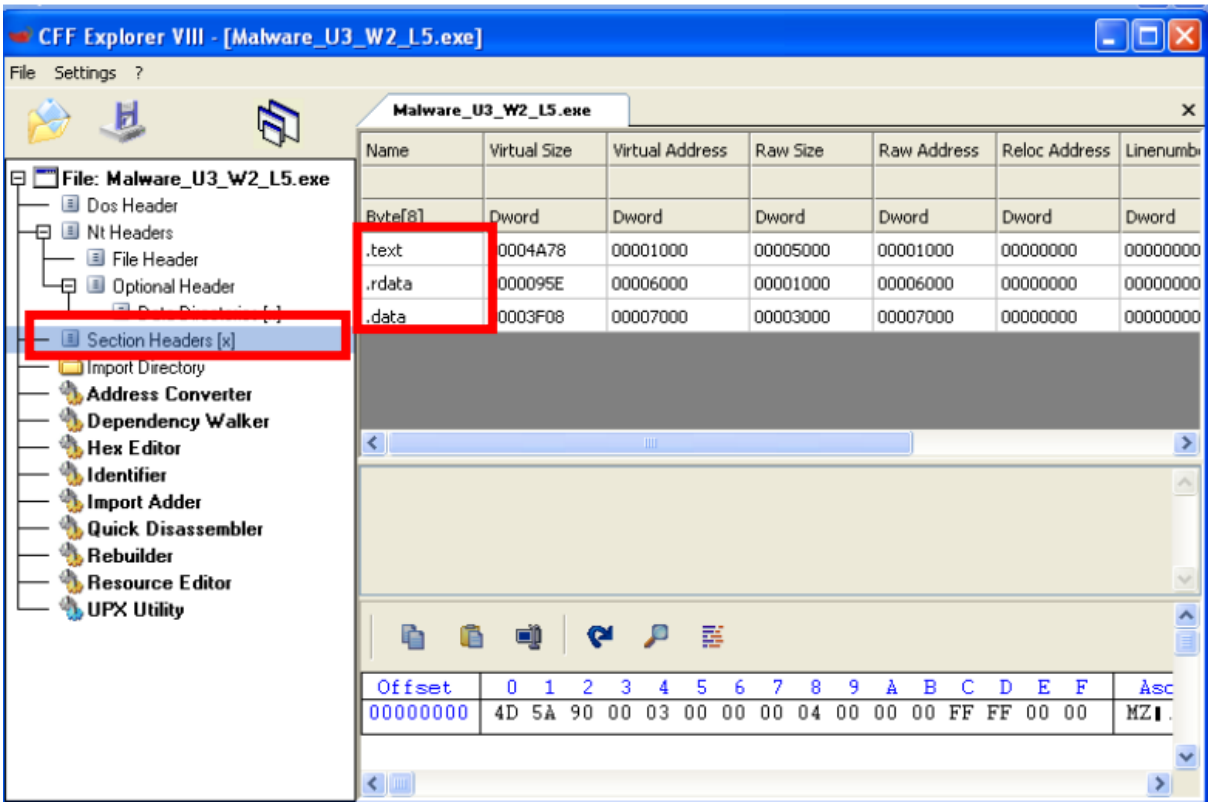


Eguale procedimento per la libreria wininet.dll, dove troviamo ad esempio la funzione InternetGetConnectedState, la quale verifica se la macchina infetta ha accesso o meno ad Internet.



TASK 2: IDENTIFICARE LE SEZIONI DI CUI SI COMPONE IL FILE ESEGUIBILE

Restando all'interno del tool CFF EXPLORER, spostandoci nella tab SECTION HEADER, abbiamo la possibilità di identificare le sezioni di cui si compone il file oggetto d'interesse.



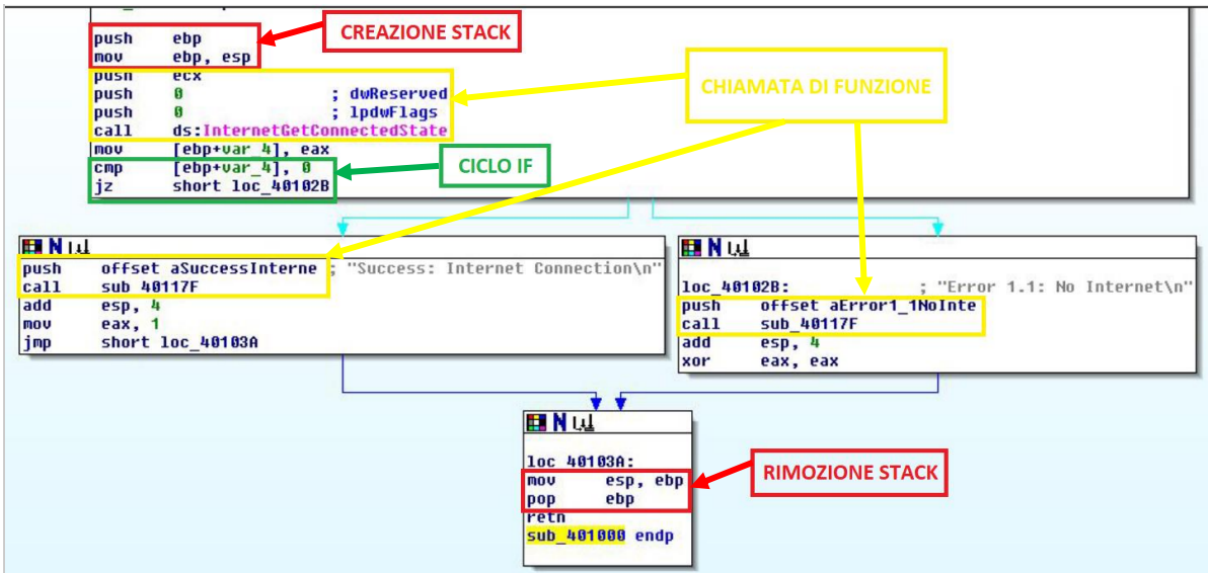
Il file eseguibile oggetto d'interesse è composto da tre sezioni, non compresse da UPX:

• .text: sezione che contiene le istruzioni, ovvero le righe di codice che la CPU eseguirà quando il software viene avviato. È la sezione principale di un file eseguibile, poiché contiene il codice effettivo che viene eseguito per far funzionare il programma. Tutte le altre sezioni contengono dati o informazioni di supporto per questa sezione.
• .rdata: sezione che contiene informazioni sulle librerie e le funzioni importate ed esportate dall'eseguibile. Qui vengono memorizzate le informazioni sui moduli esterni che l'eseguibile utilizza, come librerie di sistema o librerie condivise, e le funzioni che vengono importate o esportate per l'utilizzo all'interno del programma.
• .data: sezione che contiene dati e variabili globali del programma eseguibile. Le variabili definite in questa sezione sono accessibili da qualsiasi parte del programma, poiché sono globalmente dichiarate.

Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword
.text	0004A78	00001000	00005000	00001000	00000000	00000000
.rdata	000095E	00006000	00001000	00006000	00000000	00000000
.data	0003F08	00007000	00003000	00007000	00000000	00000000

TASK 3: IDENTIFICARE I COSTRUTTI NOTI :

1) Push ebp
2) Mov ebp, esp
3) Push ecx
4) Push 0 ; dwReversed
5) Push 0 ; lpdwFlags
6) Call ds: InternetGetConnectedState
7) Mov [ebp+var_4], eax
8) Cmp [ebp+var_4], 0
9) Jz short loc_40102B
10) Push offset aSuccessInterne ; “Success: Internet Connection\n”
11) Call sub_40117F
12) Add esp, 4
13) Mov eax, 1
14) Jmp short loc_40103A
15) Loc_40102B:
16) Push offset aError1_1NoInte
17) Call sub_40117F
18) Add esp,4
19) Xor eax, eax
20) Loc 40103°:
21) Mov esp, ebp
22) Pop ebp
23) Retn
24) Sub_401000 endp



TASK 4: IPOTIZZARE IL COMPORTAMENTO DELLE FUNZIONALITÀ IMPLEMENTATE

A seguito dell'analisi del codice assembly oggetto d'interesse, abbiamo notato il programma, tramite la funzione InternetGetConnectedState, verifica se la macchina target ha o meno una connessione ad Internet.
Dopo tale verifica, il programma, tramite la stampa di messaggi a schermo, ci fornirà in output un feedback positivo in caso di connessione avvenuta, o un feedback negativo in caso di mancata connessione.

Si specifica che, in caso di mancata connessione, il programma re-inizializza il valore del registro eax a zero, operazione questa operazione che è chiaro sintomo che il probabile malware potrebbe non essere in grado di sfruttare completamente le sue funzionalità in caso di mancata connessione internet.

Premesso ciò, il malware potrebbe utilizzare la connessione ad Internet per eseguire operazioni specifiche di cui tuttavia non abbiamo certezza, in quanto non specificate nella porzione di codice in nostro possesso.

In base a queste informazioni, possiamo soltanto ipotizzare che il malware sia stato progettato per sfruttare una connessione a Internet al fine di eseguire varie operazioni, come l'invio di file e dati sensibili a server controllati dall'attaccante, connessione a domini infetti con conseguenti download di ulteriori malware, o la creazione di una backdoor per consentire una comunicazione persistente tra la macchina vittima e l'attaccante in caso la vulnerabilità sfruttata da quest'ultimo venisse "patchata".

Dopo tali considerazioni, si potrebbe ipotizzare che il malware in cui è contenuta la porzione di codice in nostro possesso, potrebbe essere un DOWNLOADER, una BACKDOOR o un TROJAN.