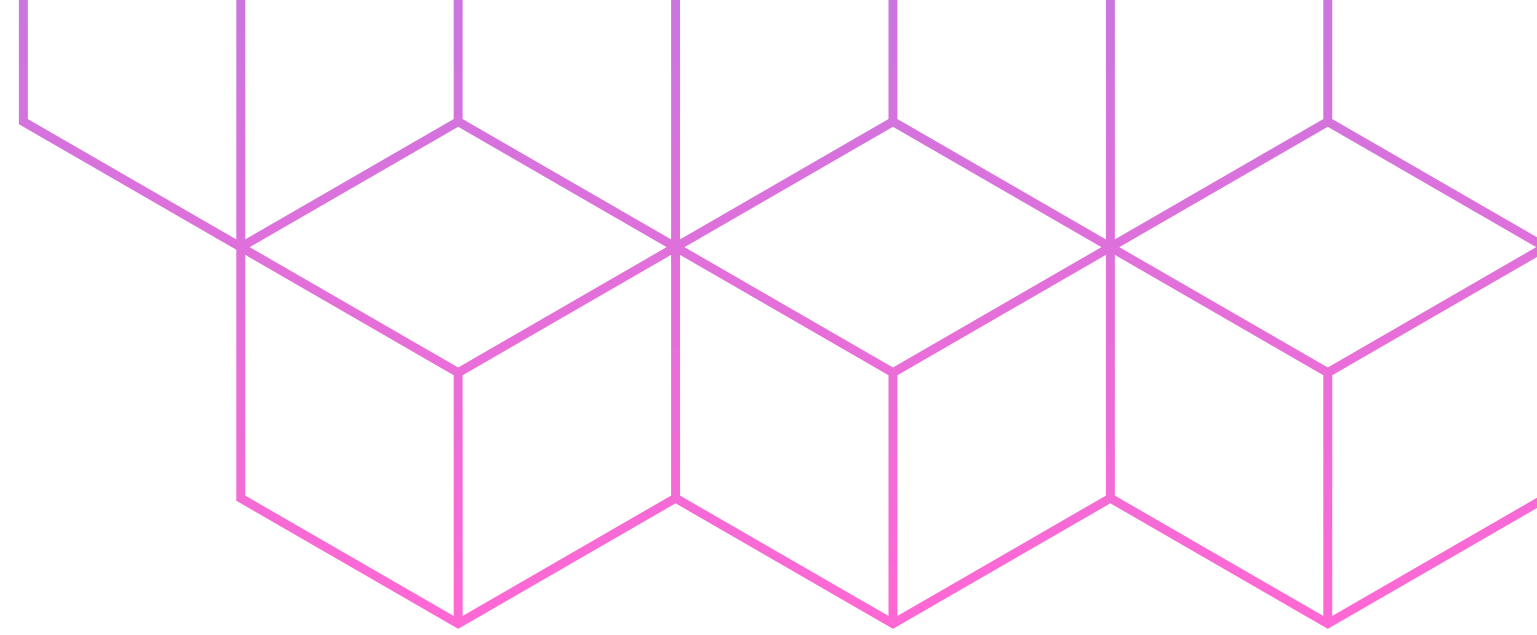


EPICODE



ANALISI STATICA E ASSEMBLY

CS_0124 Progetto S10-L5

Traccia

Analisi Statica

Librerie

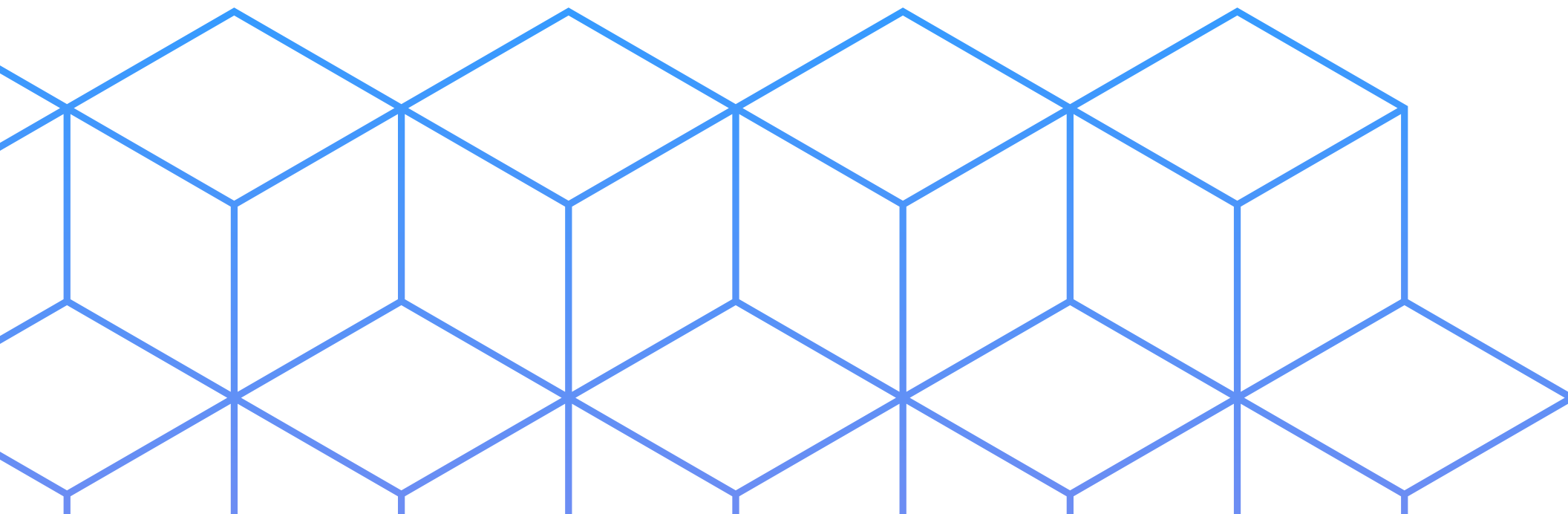
Sezioni

Assembly e costrutti noti

Ipotesi sul codice

Analisi Codice

Indice



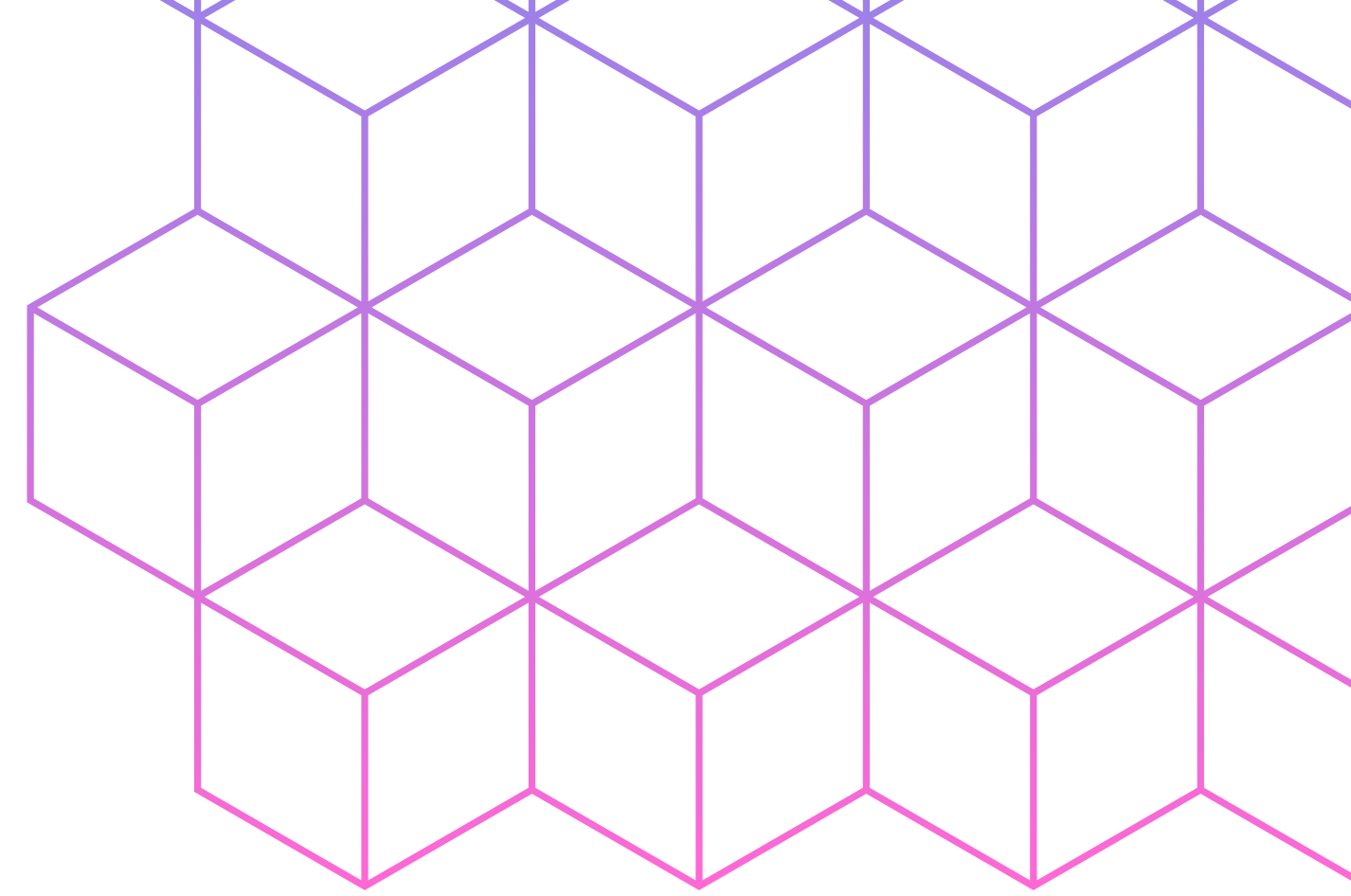
Traccia

Prima parte progetto

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 ai seguenti quesiti:

1. Quali librerie vengono importate dal file eseguibile?
2. Quali sono le sezioni di cui si compone il file eseguibile del malware?

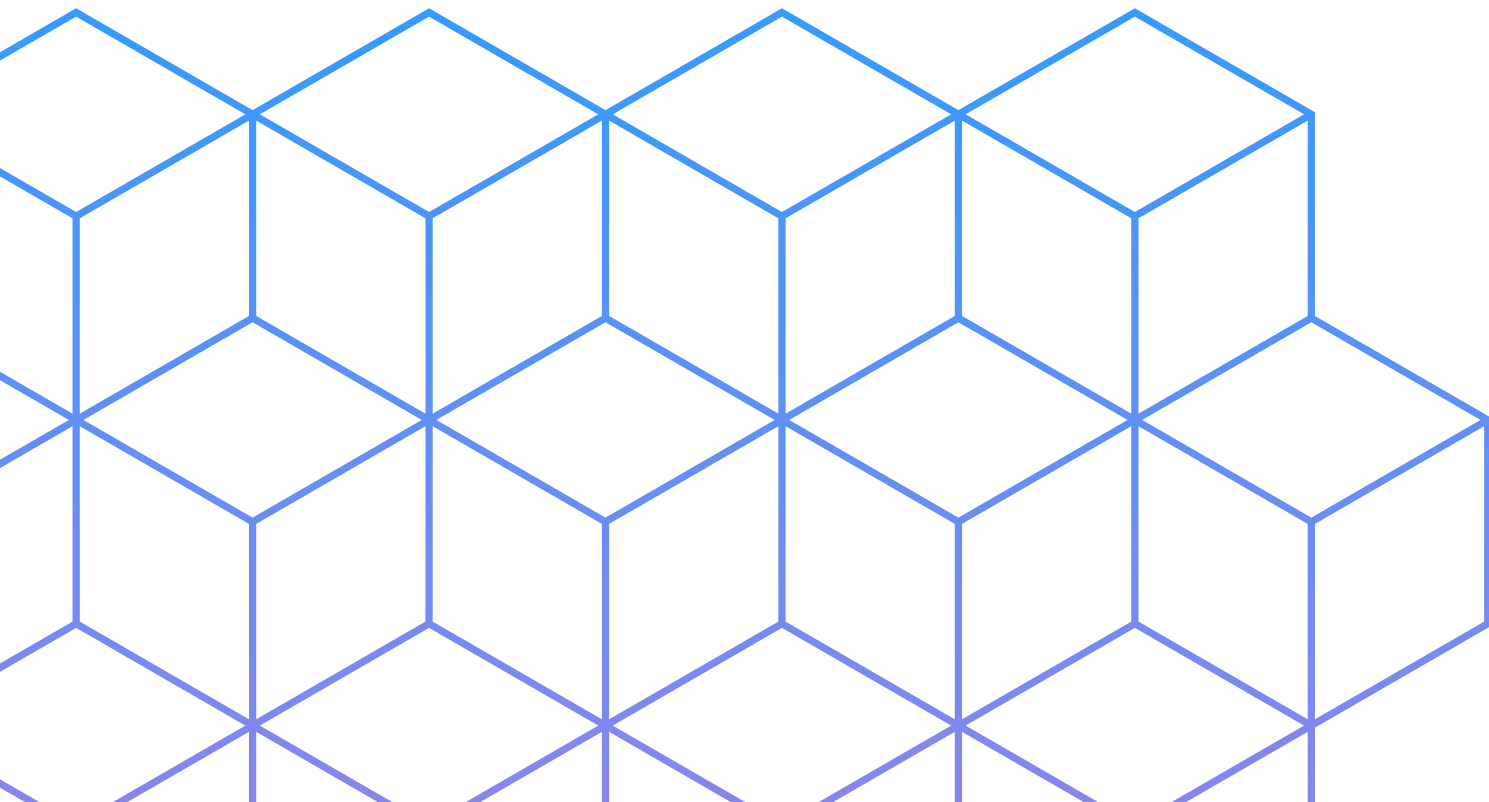
Di entrambi dare una breve spiegazione



Seconda parte progetto

Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

1. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
2. Ipotizzare il comportamento della funzionalità implementata
3. BONUS fare tabella con significato delle singole righe di codice assembly



Analisi statica

01

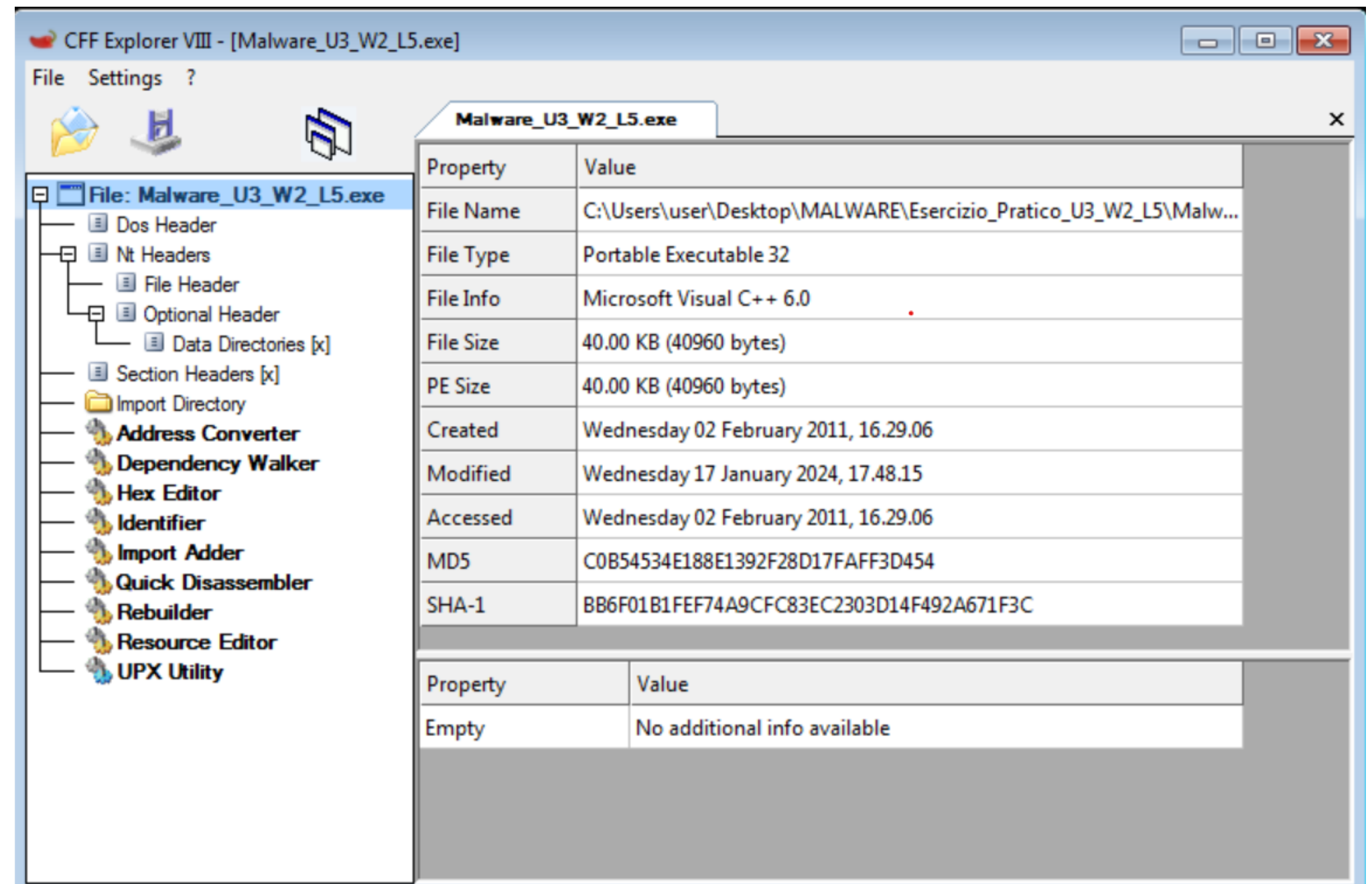
Nella prima fase del progetto effettueremo l'analisi statica di base del malware operando su una Virtual machine completamente isolata. Tra le buone pratiche da adottare per configurare un ambiente sicuro troviamo:

1. La configurazione della scheda di rete; l'ambiente di test infatti non deve avere accesso diretto ad internet, né ad altri client presenti nella rete.
2. Dispositivi USB; è buona pratica non abilitare o disabilitare il controller USB, in quanto esso può essere riconosciuto anche dall'ambiente di test.
3. Cartelle condivise; esse potrebbero essere usate dal malware per propagarsi al di fuori del laboratorio. Dunque, è consigliato non averne.
4. Creare istantanee; questo perché durante l'analisi di un malware spesso si potrebbe danneggiare l'ambiente di test. Una buona pratica è creare istantanee della VM per ripristinarla qualora ce ne fosse bisogno

Queste pratiche servono soprattutto per l'analisi dinamica del malware, ma è comunque essenziale avere un ambiente sicuro per condurre delle analisi accurate.

Analisi statica

Essendo questa un'analisi statica non eseguiamo il malware, ma semplicemente ispezioneremo il contenuto del file eseguibile. Per questo il tool utilizzato è CFF Explorer che, attraverso una comoda interfaccia, permette all'utente di analizzare l'eseguibile ed estrarre informazioni essenziali sul malware quali: librerie, sezioni, etc. Nella sezione "import directory" possiamo verificare quali sono le librerie importate dal malware.



1. Librerie

Nella sezione Import Directory vediamo le librerie importate dall'eseguibile, queste servono al .exe per accedere a delle funzionalità esterne. Per ogni libreria possiamo visualizzare i dettagli come il nome, l'indirizzo virtuale, e le funzioni specifiche.

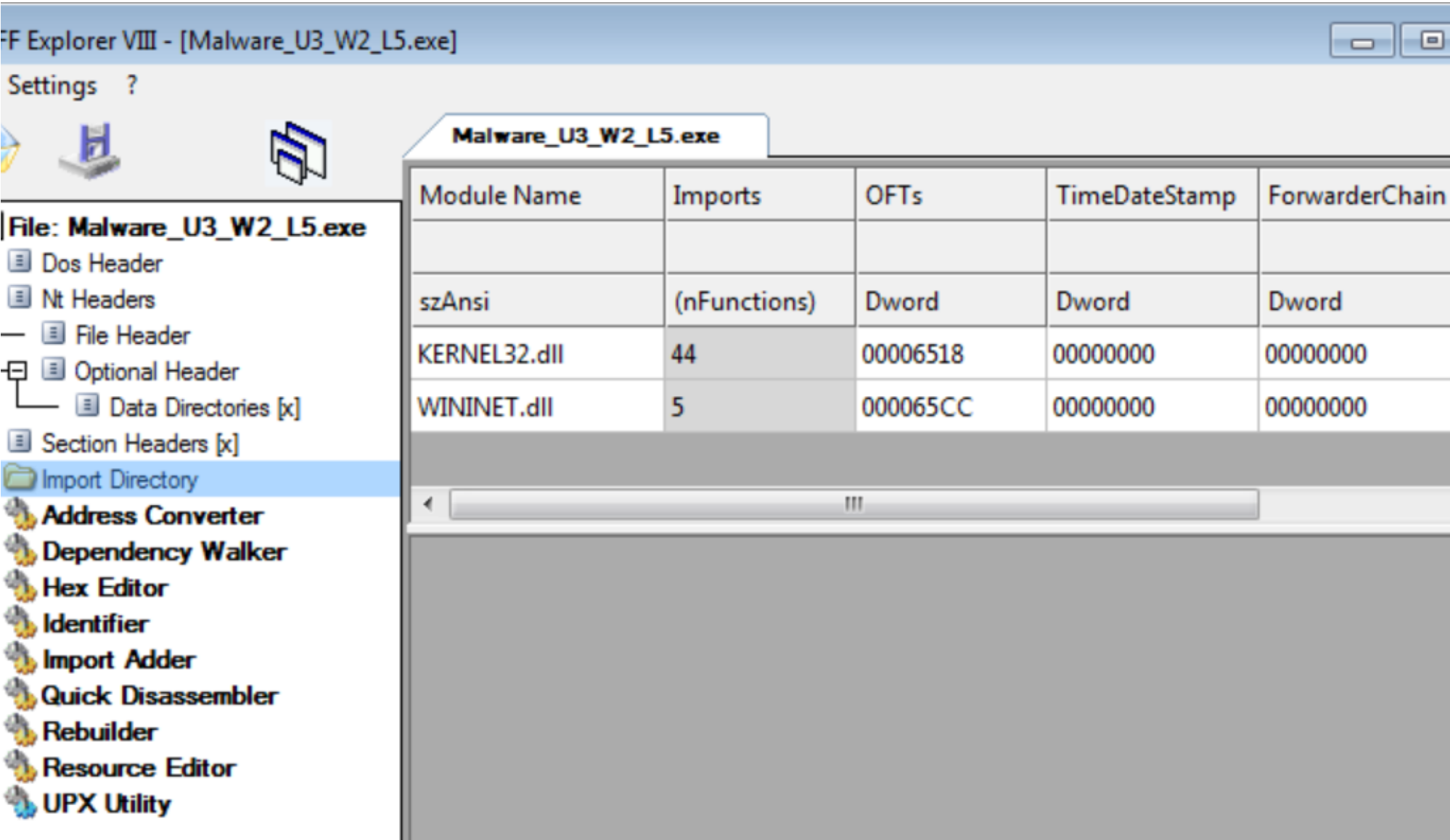
Tutte queste sono informazioni fondamentali per l'analisi del malware, infatti esse ci permettono di individuare le risorse esterne usate dall'eseguibile.

KERNEL32.dll > è un file DLL di Windows. DLL è l'acronimo di Dynamic Link Library. I file DLL sono programmi o estensioni del browser web necessari, perché contengono le risorse, i dati e il codice del programma. Quindi include le funzioni core del sistema operativo.

Da questo modulo il malware importa 44 funzioni.

WININET.dll > è un modulo che contiene le funzioni Internet-relative usate dalle applicazioni di Windows, le funzioni per implementare i servizi di rete come FTP, NTP, HTTP.

Da questo modulo il malware importa 5 funzioni



Module Name	Imports	OFTs	TimeStamp	ForwarderChain
szAnsi	(nFunctions)	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000
WININET.dll	5	000065CC	00000000	00000000

2. Sezioni

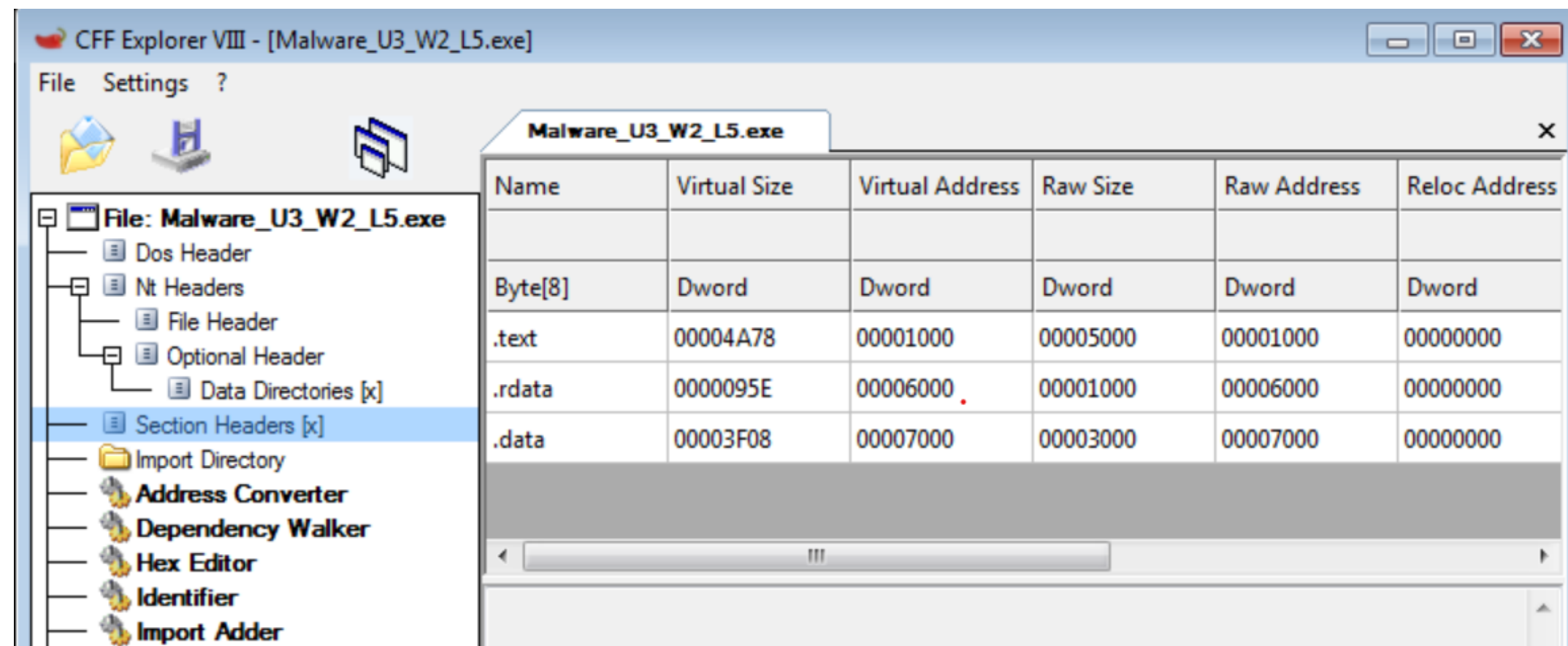
Le sezioni riscontrate sono:

.text > la sezione contiene le istruzioni o il codice che saranno inviate alla CPU che eseguirà una volta avviato il software;

.rdata > la sezione include informazioni sulle librerie e le funzioni importate ed esportate dal file eseguibile

.data > la sezione contiene le variabili globali del file eseguibile, accessibili da qualsiasi parte del programma

Nella sezione "Section Headers" reperiamo informazioni dettagliate sulle varie sezioni del malware. Queste informazioni possono includere: nomi delle sezioni, indirizzi in memoria in cui sono allocate, le dimensioni, i permessi di accesso etc, dunque dati utili per la comprensione della struttura del file.

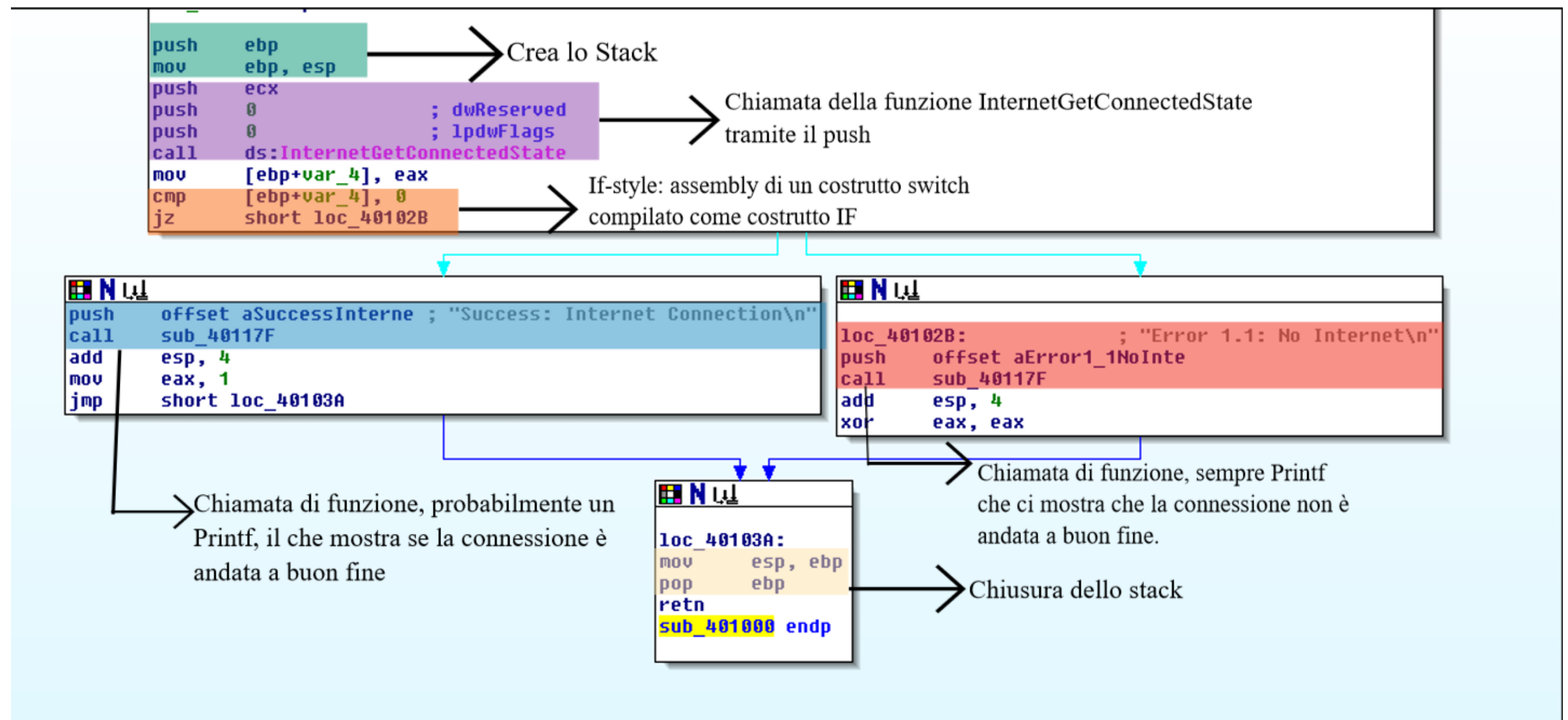


Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	00004A78	00001000	00005000	00001000	00000000
.rdata	0000095E	00006000	00001000	00006000	00000000
.data	00003F08	00007000	00003000	00007000	00000000

1. Assembly e costrutti noti

02

La seconda parte del progetto tratta l'analisi del codice assembly di un malware. Viene richiesto l'identificazione dei costrutti noti:



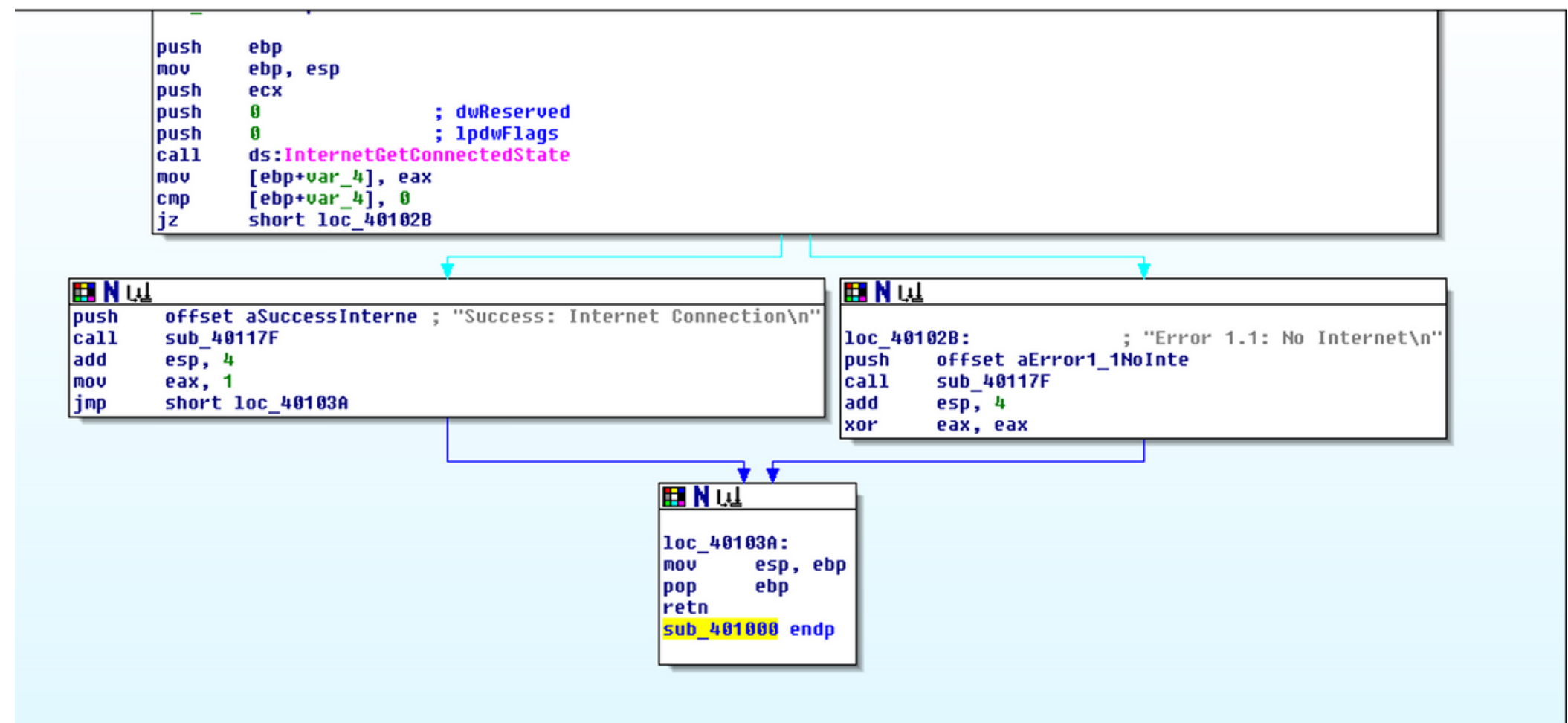
2. Ipotesi sul codice

Il codice scritto in assembly x86 inizia impostando il frame del registro base - ebp - e dello stack pointer - esp. In seguito, dall'analisi emerge che il malware esegue una funzione di controllo della connessione sulla macchina bersaglio, utilizzando la chiamata di funzione "InternetGetConnectedState".

Se il valore di ritorno della funzione è diverso da zero, indica una connessione attiva, e il malware potrebbe procedere con determinate azioni o attivare funzionalità specifiche dipendenti dalla presenza di una connessione attiva.

Al contrario, se il valore di ritorno è uguale a zero, indica l'assenza di connessione, e dunque il malware conclude la sua esecuzione.

Infine, ripristina lo stack pointer e il registro base, terminando la funzione.



3. Analisi codice

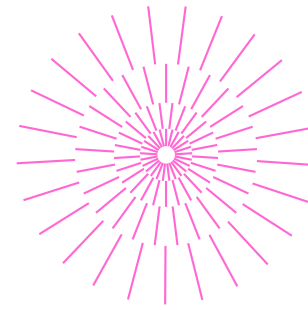
ISTRUZIONE	DESCRIZIONE
push ebp	Mette il valore attuale del registro EBP nello stack
mov ebp, esp	Copia il valore dello stack di ESP in EBP.
push ecx	Mette il valore di ECX nello stack
push 0; dwReserved:	Mette il valore 0 nello stack. E lo passa come argomento dwReserved alla funzione.
push 0 ; lpdwFlags	Mette il valore 0 nello stack. E lo passa come argomento lpdwFlags alla funzione.
call ds:InternetGetConnectedState:	Chiama la funzione InternetGetConnectedState che controlla lo stato della connessione a Internet.

3. Analisi codice

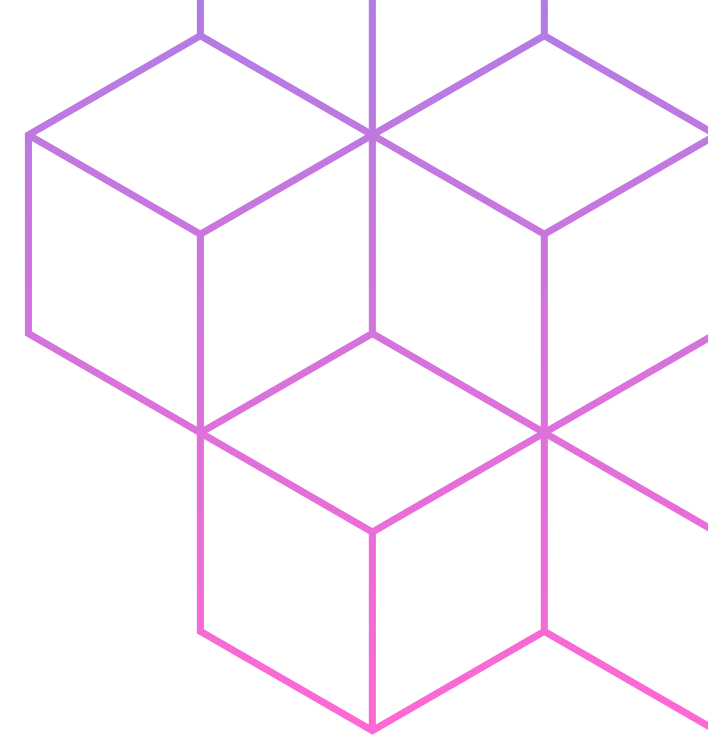
ISTRUZIONE	DESCRIZIONE
mov [ebp+var_4], eax:	Copia il valore restituito dalla funzione InternetGetConnectedState in [ebp+var_4].
cmp [ebp+var_4], 0:	Confronta il valore memorizzato con 0.
jz short loc_40102B:	Passa ad loc_40102B se il valore nella variabile è zero. Quindi se non c'è connessione a Internet, il programma salta a loc_40102B.
push offset aSuccessInterne	Mette l'indirizzo della stringa aSuccessInterne nello stack. Sembra essere un messaggio che indica la corretta connessione a internet.
call sub_40117F	Chiama una subroutine a sub_40117F, per stampare il messaggio di successo sulla connessione.
add esp, 4	Ripristina lo stack dopo la chiamata alla funzione.

3. Analisi codice

ISTRUZIONE	DESCRIZIONE
move eax, 1	copia 1 nel registro EAX.
jmp short loc_40103A	Passa a loc_40103A. Questa azione viene eseguita a prescindere se è presente una connessione a internet.
push offset aERROR1_NoInte	Mette l'indirizzo del messaggio di errore sulla pila
xor eax, eax	Esegue un'operazione XOR per azzerare eax
mov esp, ebp	Ripristina il valore dello stack pointer
pop ebp	Ripristina il valore del registro base
retn	Restituisce il controllo al chiamante



Epicode



Grazie!

