

Gestione Malware U3_W2_L5

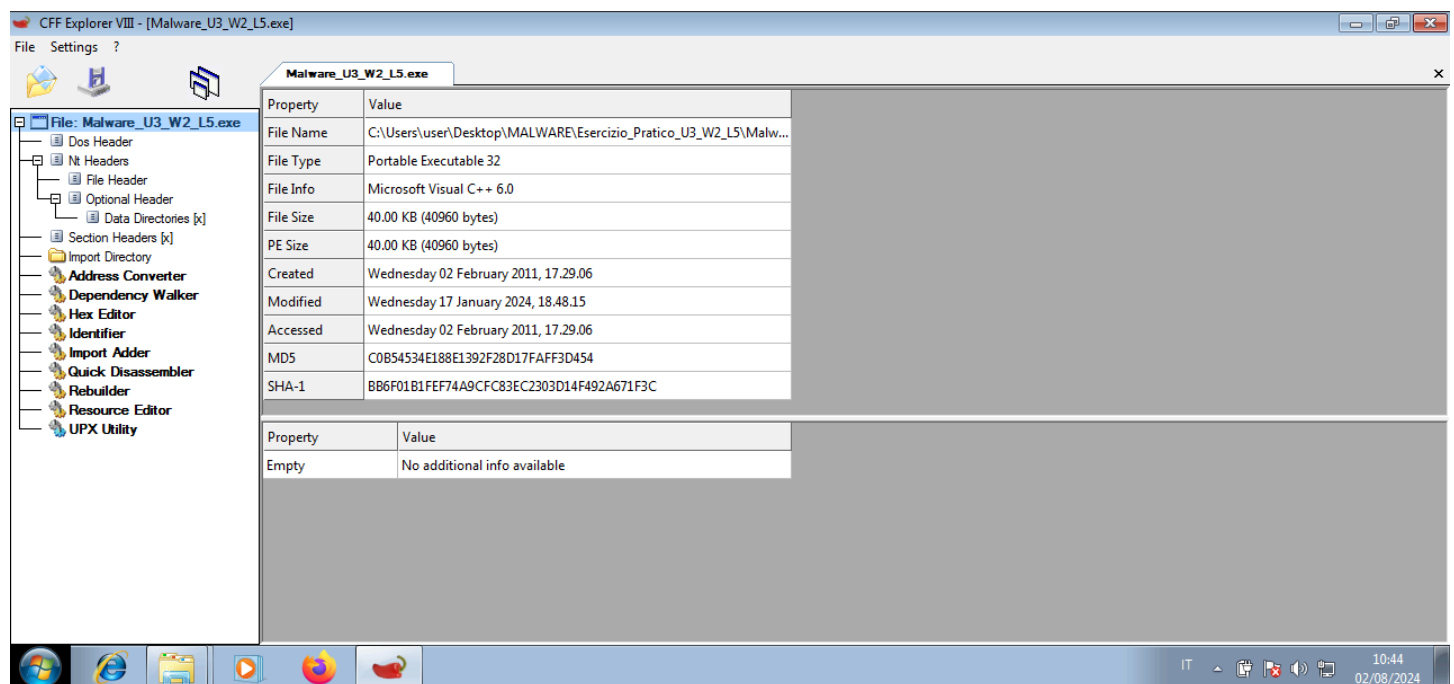
Traccia:

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:

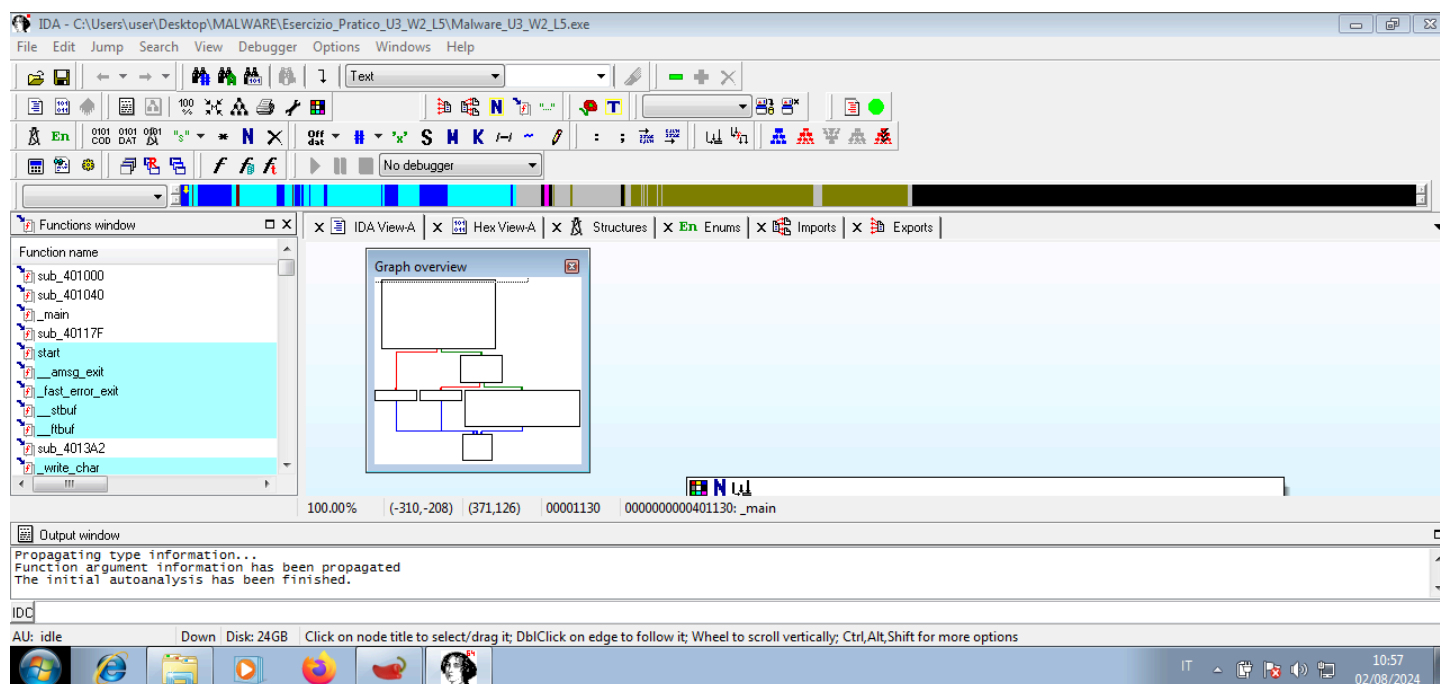
1. Quali librerie vengono importate dal file eseguibile? Fare anche una descrizione
2. Quali sono le sezioni di cui si compone il file eseguibile del malware? Fare anche una descrizione Con riferimento alla figura in slide 3, risponde ai seguenti quesiti.
3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
4. Ipotizzare il comportamento della funzionalità implementata
5. Fare una tabella per spiegare il significato delle singole righe di codice

Premessa:

Per la ricerca delle **librerie** utilizzate dal malware e anche per la ricerca delle **sezioni**, ho utilizzato il software **CFF Explorer (PE Compact File Format Explorer)** :



Mentre per la gestione dei costrutti e l'ipotesi della del funzionamento, mi sono fatto aiutare dal software **IDA pro (Interactive DisAssembler)**:



Librerie Importate e le loro Funzioni:

Le librerie possono essere trovate tramite il software CFF Explorer

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

1. KERNEL32.dll

- **Funzione Principale:** La **KERNEL32.dll** è una libreria centrale del sistema operativo Windows che gestisce operazioni critiche e di base. Alcune delle funzioni più rilevanti che può fornire includono:
 - Gestione della **memoria** (allocazione e deallocazione)
 - Gestione dei **file** e delle **directory** (creazione, apertura, lettura, scrittura, eliminazione)
 - Gestione dei processi e dei **thread** (creazione, terminazione, sincronizzazione)
 - Gestione delle operazioni di **input/output**
- **Ruolo nel Malware:** Un malware potrebbe utilizzare **KERNEL32.dll** per eseguire operazioni di base come la scrittura di dati su disco, la creazione di nuovi processi per mantenere la persistenza, e la manipolazione della memoria per iniettare codice dannoso in altri processi.

2. WININET.dll

- **Funzione Principale:** La **WININET.dll** fornisce una serie di **API** per la comunicazione Internet. Queste funzioni sono cruciali per le applicazioni che richiedono accesso a risorse di rete. Alcune delle funzioni più comuni includono:
 - **InternetOpen:** Inizializza l'utilizzo delle funzioni di rete da parte di un'applicazione.

- **InternetConnect:** Stabilisce una connessione a un server specifico.
- **HttpOpenRequest:** Crea una richiesta HTTP.
- **HttpSendRequest:** Invia una richiesta HTTP a un server.
- **InternetReadFile:** Legge i dati da un handle di file Internet.
- **Ruolo nel Malware:** Un malware potrebbe utilizzare **WININET.dll** per comunicare con un server di comando e controllo (C2), **scaricare ulteriori payload** dannosi, e filtrare dati, o ricevere istruzioni su operazioni da eseguire.

Descrizione delle Sezioni:

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

1. .text

- **Descrizione:** Questa sezione contiene il codice eseguibile del programma. È marcata come eseguibile e leggibile, il che significa che le istruzioni CPU che il programma esegue risiedono in questa sezione.
- **Ruolo nel Malware:** La logica principale del malware, incluse tutte le sue funzionalità dannose, viene eseguita da questa sezione.

2. .rdata

- **Descrizione:** La sezione .rdata contiene dati di sola lettura. Può includere stringhe di testo, informazioni di debug e altre costanti che il programma utilizza.
- **Ruolo nel Malware:** Questa sezione potrebbe contenere stringhe di messaggi di errore, URL, chiavi di crittografia o altre informazioni statiche necessarie al funzionamento del malware.

3. .data

- **Descrizione:** La sezione .data contiene dati inizializzati usati dal programma durante la sua esecuzione. È leggibile e scrivibile.
- **Ruolo nel Malware:** Questa sezione può contenere variabili globali e statiche che il malware modifica durante la sua esecuzione. Ad esempio, potrebbe contenere contatori, flag di stato, o altre informazioni che il malware aggiorna dinamicamente.

Identificare i costrutti noti:

- **Creazione dello stack:** Identificare istruzioni come **push** e **mov** che manipolano lo stack.
- **Cicli:** Cercare istruzioni di salto condizionato (**jz**, **jmp**) che possono indicare cicli o condizioni.
- **Funzioni:** Identificare chiamate a funzioni (**call**)
- **Registri:** manipolazione di registri (**mov**, **xor**)
- **Matematica:** operazioni aritmetiche (**add**).












```

push    ebp
mov     ebp, esp
sub     esp, 8
call    sub_401000
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jnz     short loc_401148

```

Ipotizzare il comportamento della funzionalità implementata:

Dall'analisi approfondita grazie ad **IDA** possiamo intuire che il malware importa diverse funzioni:

Address	Ordinal	Name	Library
 000000...		GetProcAddress	KERNEL32
 000000...		LoadLibraryA	KERNEL32
 000000...		GetLastError	KERNEL32
 000000...		FlushFileBuffers	KERNEL32
 000000...		SetFilePointer	KERNEL32
 000000...		CloseHandle	KERNEL32
 000000...		InternetOpenUrlA	WININET
 000000...		InternetCloseHandle	WININET
 000000...		InternetReadFile	WININET
 000000...		<u>InternetGetConnectedState</u>	WININET
 000000...		InternetOpenA	WININET

e dallo studio dell'**assembly** rilasciato nella traccia, possiamo notare che:

- Il codice chiama la funzione **InternetGetConnectedState** (**call ds:InternetGetConnectedState**) per verificare lo stato della connessione Internet.
- In base al risultato della chiamata (**cmp [ebp+var_4], 0**), il codice decide quale messaggio stampare:
 - **Successo:** Stampa "**Success:** Internet Connection".
 - **Errore:** Stampa "**Error** 1.1: No Internet".

5. Fare una tabella per spiegare il significato delle singole righe di codice

Indirizzo	Istruzione	Descrizione
401000	push ebp	Salva il valore del registro EBP sullo stack
401001	mov ebp, esp	Imposta EBP a ESP, creando un nuovo frame dello stack
401003	push ecx	Salva il valore del registro ECX sullo stack
401004	push 0	Spinge il valore 0 sullo stack (dwReserved)
401006	push 0	Spinge il valore 0 sullo stack (lpdwFlags)
401008	call ds:InternetGetConnectedSta te	Chiama la funzione per verificare la connessione Internet
40100D	mov [ebp+var_4], eax	Salva il valore di ritorno in una variabile locale
401010	cmp [ebp+var_4], 0	Confronta la variabile locale con 0
401014	jz short loc_40102B	Salta a loc_40102B se la variabile è zero (nessuna connessione)
401016	push offset aSuccessInterne	Spinge il messaggio di successo sullo stack
40101B	call sub_40117F	Chiama la subroutine sub_40117F
401020	add esp, 4	Aggiunge 4 a ESP (pulizia dello stack)
401023	mov eax, 1	Imposta EAX a 1 (indicando

		successo)
401028	jmp short loc_40103A	Salta a loc_40103A
40102B	push offset aError1_1NoInte	Spinge il messaggio di errore sullo stack
401030	call sub_40117F	Chiama la subroutine sub_40117F
401035	add esp, 4	Aggiunge 4 a ESP (pulizia dello stack)
401038	xor eax, eax	Imposta EAX a 0 (indicando errore)
40103A	mov esp, ebp	Ripristina ESP dal valore di EBP
40103C	pop ebp	Ripristina EBP
40103D	retn	Ritorna dalla funzione