Progetto S10L5: Simone Esposito

EPICODE

Esercizio
Traccia e requisiti

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

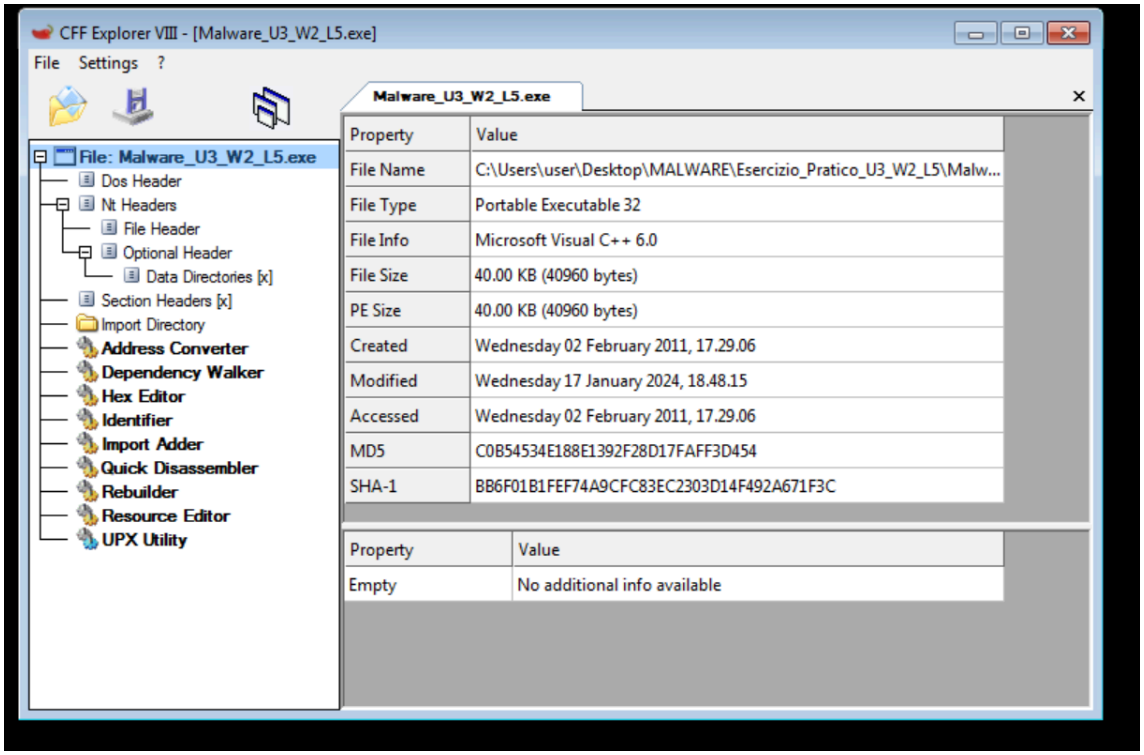
Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

3. Identificare i **costrutti** noti (creazione dello stack, eventuali cicli, altri costrutti)
4. **Ipotizzare il comportamento della funzionalità implementata**

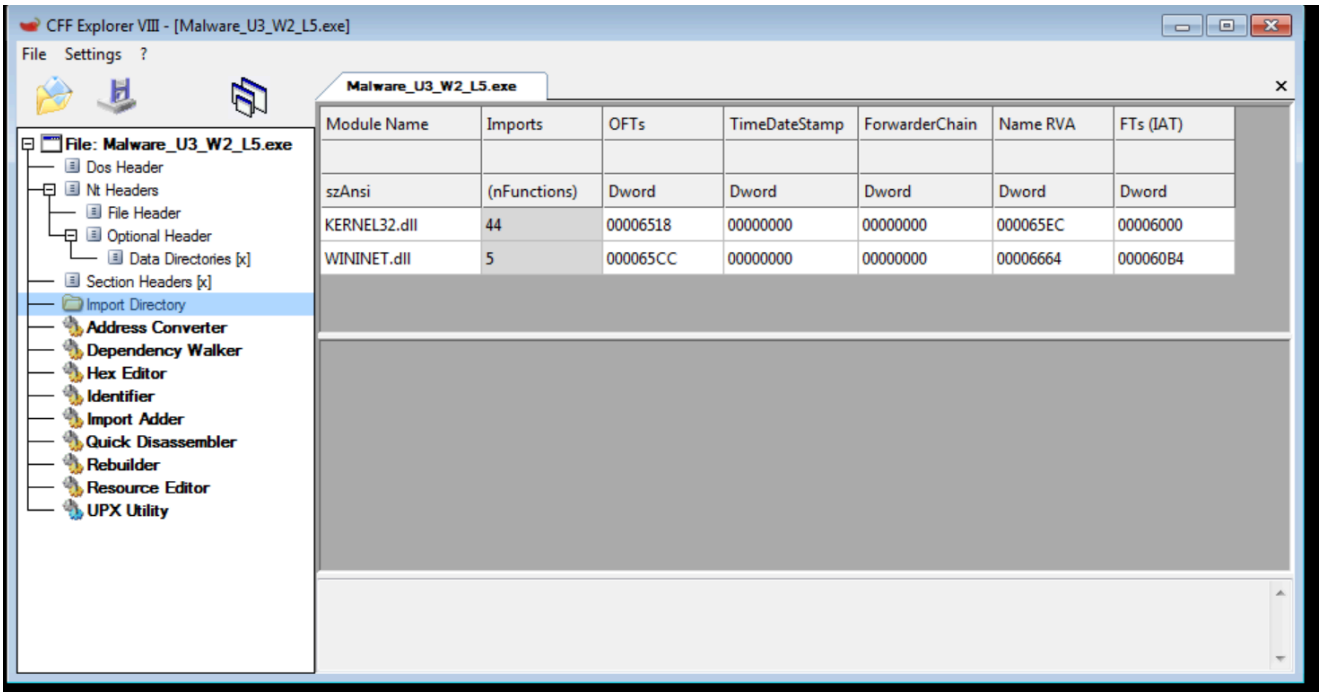
5. Fare una tabella per spiegare il significato delle singole righe di codice

Svolgimento punti 1,2:

Per analizzare il malware, sulla nostra macchina virtuale (windows 7) andiamo ad eseguire "CFF explorer" ed apriamo il malware indicatoci nella traccia.



In seguito andiamo nella sezione “Import Directory” per visualizzare tutte le librerie importate:



Analizziamo queste due librerie per capire il loro funzionamento:
Esempio di librerie comuni importate dai malware:

- **kernel32.dll**: Gestione delle operazioni di base del sistema come la gestione della memoria e dei file. E’ una delle librerie più fondamentali e critiche del sistema operativo Microsoft Windows. Fornisce una vasta gamma di funzioni di base che permettono alle applicazioni di interagire con il sistema operativo a un livello molto basso.

All’interno possiamo rilevare 44 funzioni tra cui:

La funzione Sleep sospende l'esecuzione del thread chiamante per un intervallo di tempo specificato. Il parametro della funzione è il numero di millisecondi per cui il thread deve essere sospeso.

Uso nei malware:

- Ritardare l'esecuzione di codice dannoso per evitare il rilevamento da parte dei software di sicurezza.
- Sincronizzare le attività malevole con altre azioni o condizioni specifiche.
- Implementare tecniche di anti-debugging: un malware potrebbe usare Sleep per verificare se viene eseguito in un ambiente di debugging, rallentando l'esecuzione per confondere i debugger.

La funzione CloseHandle chiude un handle di un oggetto aperto. Gli handle possono riferirsi a file, processi, thread, eventi e altri oggetti di sistema.

Uso nei malware:

- Pulire le tracce di attività malevole chiudendo gli handle dopo l'uso.
- Chiudere handle di risorse di sistema per interferire con il funzionamento di altri software o sistemi di sicurezza.
- Parte della logica di gestione delle risorse del malware, garantendo che le operazioni malevole non causino problemi che potrebbero attirare l'attenzione dell'utente o di software di sicurezza.

- **WININET.dll** è una libreria di Microsoft Windows che fornisce funzioni per l'accesso a Internet. È una libreria essenziale per applicazioni che necessitano di comunicare attraverso protocolli Internet come HTTP e FTP.

Funzionalità principali di WININET.dll

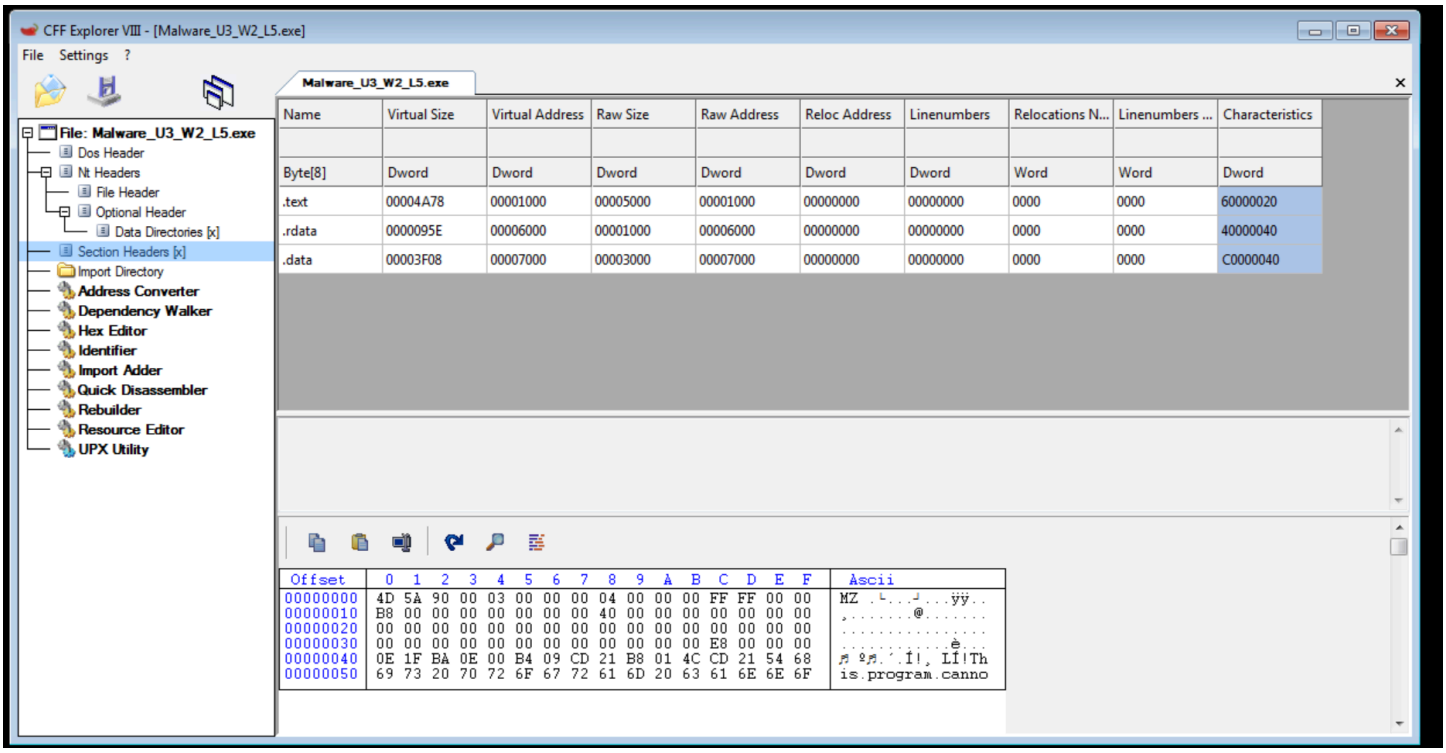
1. **HTTP e HTTPS**: Fornisce supporto per la comunicazione attraverso il protocollo HTTP e HTTPS.
2. **FTP**: Implementa funzionalità per il trasferimento di file tramite il protocollo FTP.
3. **Caching**: Gestisce la cache dei contenuti Internet per migliorare le prestazioni delle applicazioni.
4. **Gestione dei cookie**: Fornisce supporto per la gestione dei cookie HTTP.
5. **Autenticazione**: Supporta diversi meccanismi di autenticazione per le comunicazioni HTTP.
6. **Connessioni sicure**: Implementa il supporto per le connessioni SSL/TLS.

Tra le funzioni piu importanti ritroviamo:

La funzione InternetOpenUrlA apre una URL specificata utilizzando una sessione internet esistente. Questa funzione è utilizzata per recuperare dati da una URL tramite HTTP, HTTPS o FTP.

La funzione InternetCloseHandle chiude un handle aperto da una funzione WinINet. Questa funzione è utilizzata per liberare le risorse allocate per una sessione internet, una richiesta HTTP o altre operazioni di rete.

Invece per vedere le sezioni del file eseguibile basta andare nella sezione “Headers”:



Ogni sezione può presentare codici, dati, risorse o informazioni necessarie per l’esecuzione del programma. Possiamoi notare dallo screenshot che il malware presenta 3 sezioni:

.text: Questa sezione è dove risiede il cuore del malware, cioè il codice che esegue le operazioni malevole.

.data: Questa sezione può contenere variabili globali, configurazioni, chiavi di decrittazione o dati necessari per l'esecuzione del malware.

.rdata: Questa sezione può includere stringhe di testo utilizzate per comunicazioni, messaggi di errore, URL, o altre risorse necessarie. Le tabelle di importazione sono cruciali per capire quali funzioni e librerie esterne vengono utilizzate dal malware.

CONCLUSIONE:

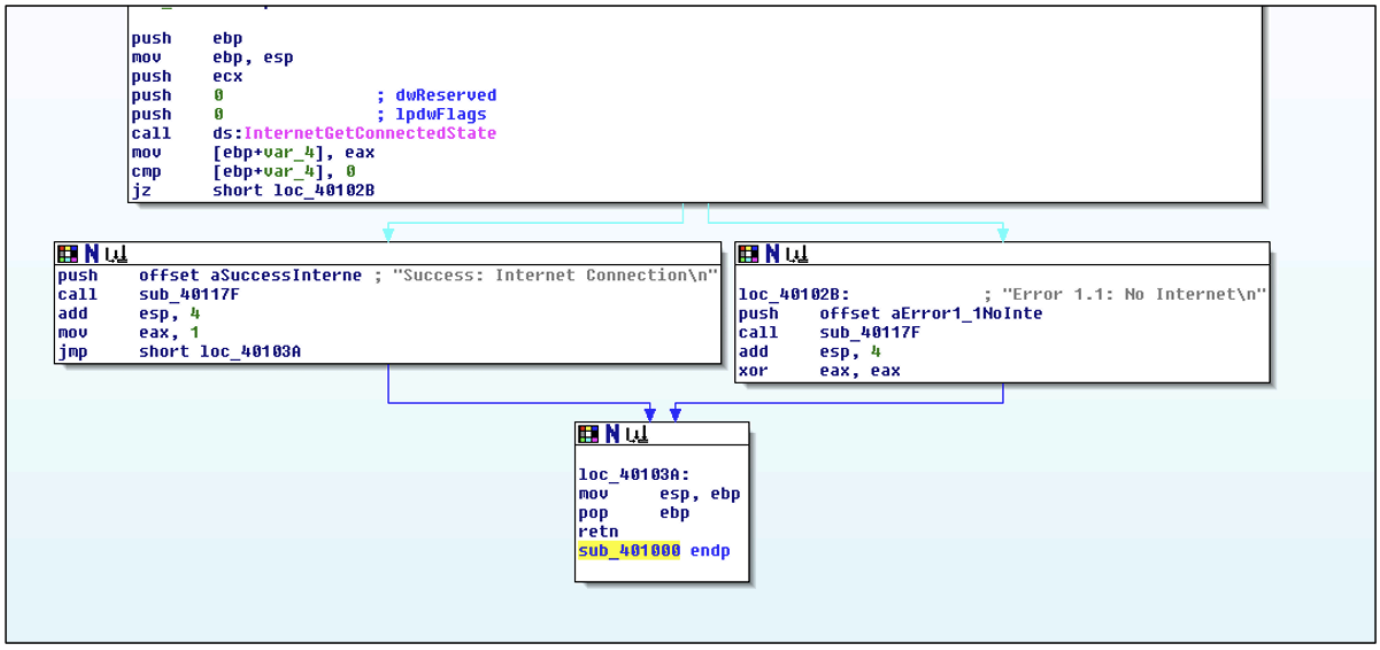
I malware spesso usano queste librerie per compiere operazioni critiche e manipolative come:

- **Creazione e gestione di processi e thread**: Per eseguire codice malevolo.
- **Manipolazione della memoria**: Per nascondere codice malevolo o iniettare codice in altri processi.

- **Gestione dei file**: Per creare, leggere, scrivere o eliminare file necessari per le operazioni del malware.
- **Comunicazione con server remoti**: I malware possono inviare e ricevere dati da server di comando e controllo (C&C).
- **Scaricamento di ulteriori payload**: Può essere usata per scaricare componenti aggiuntivi o aggiornamenti del malware.

Il malware è un tipo di Trojan, lo abbiamo confermato attraverso VirusTotal, ed è un malware che richiede una stretta integrazione con il sistema operativo e la capacità di comunicare via Internet.

Svolgimento punti 3,4,5 in riferimento alla seguente slide:



3. Identificare i costrutti noti

Creazione dello stack:

- push ebp
- mov ebp, esp

- push ecx

  Questa sequenza è comune nelle funzioni per preparare lo stack e garantire che i dati importanti non vengano persi durante l'esecuzione della funzione.

Chiamata alla funzione InternetGetConnectedState:

- push 0
- push 0
- call ds:InternetGetConnectedState
- mov [ebp+var_4], eax
- cmp [ebp+var_4], 0
- jz short loc_40102B

La funzione InternetGetConnectedState serve a determinare se il sistema è attualmente connesso a Internet. Restituisce TRUE se è presente una connessione a Internet e FALSE se non lo è. Può anche fornire informazioni aggiuntive sul tipo di connessione tramite una variabile di bandiere di stato.

Se la connessione è presente:

- push offset aSuccessInterne
- call sub_40117F
- add esp, 4
- mov eax, 1
- jmp short loc_40103A

Se la connessione non è presente:

- loc_40102B:
- push offset aError1_1NoInte
- call sub_40117F
- add esp, 4
- xor eax, eax

  Questi blocchi di codice gestiscono rispettivamente la presenza o l'assenza della connessione a Internet, stampando messaggi appropriati e impostando un valore di ritorno in eax.

Chiusura dello stack frame e ritorno:

- loc_40103A:
- mov esp, ebp
- pop ebp
- retn

Salti Condizionali:

- cmp [ebp+var_4], 0 e jz short loc_40102B controllano se c'è connessione a Internet.

- jmp short loc_40103A è un salto condizionale per evitare l'esecuzione del blocco di codice per la mancanza di connessione.

I salti condizionali sono istruzioni utilizzate nei linguaggi di programmazione e nei linguaggi assembly per controllare il flusso di esecuzione del programma in base a una condizione specifica. Queste istruzioni consentono di saltare a una determinata parte del codice se una condizione è vera o falsa.

4. Ipotizzare il comportamento della funzionalità implementata

Il frammento di codice sembra essere una funzione che verifica lo stato della connessione a Internet e restituisce un messaggio di successo o di errore.

- Utilizza la funzione InternetGetConnectedState per determinare se c'è una connessione a Internet.
- Se la connessione è presente, stampa "Success: Internet Connection".
- Se la connessione non è presente, stampa "Error 1.1: No Internet".

5. Tabella per spiegare il significato delle singole righe di codice

CODICE	SIGNIFICATO
push ebp	Salva il base pointer corrente sullo stack.
mov ebp, esp	Imposta il base pointer al valore dello stack pointer.
push ecx	Salva il registro ecx sullo stack.
push 0	Pusha 0 sullo stack (argomento per dwReserved).
push 0	Pusha 0 sullo stack (argomento per lpdwFlags).
call ds:InternetGetConnectedState	Chiama la funzione InternetGetConnectedState.
mov [ebp+var_4], eax	Memorizza il valore di ritorno della funzione in [ebp+var_4].
cmp [ebp+var_4], 0	Confronta il valore memorizzato con 0.
jz short loc_40102B	Se il valore è 0, salta a loc_40102B.
push offset aSuccessInterne	Pusha l'offset della stringa "Success: Internet Connection" sullo stack.
call sub_40117F	Chiama la funzione sub_40117F (probabilmente una funzione di stampa).
add esp, 4	Libera lo spazio dello stack usato per l'argomento.
mov eax, 1	Imposta il registro eax a 1 (successo).
jmp short loc_40103A	Salta a loc_40103A.

loc_40102B:	Etichetta per il punto di salto se la connessione non è presente.
push offset aError1_1NoInte	Pusha l'offset della stringa "Error 1.1: No Internet" sullo stack.
call sub_40117F	Chiama la funzione sub_40117F (probabilmente una funzione di stampa).
add esp, 4	Libera lo spazio dello stack usato per l'argomento.
xor eax, eax	Imposta eax a 0 (fallimento).
loc_40103A:	Etichetta per il punto di salto dopo la stampa del messaggio di successo.
mov esp, ebp	Ripristina il valore di esp dal base pointer.
pop ebp	Ripristina il base pointer salvato.
retn	Ritorna dalla funzione.

ESERCIZIO BONUS:

BONUS:

Un giovane dipendente neo assunto segnala al reparto tecnico la presenza di un programma sospetto.
Il suo superiore gli dice di stare tranquillo ma lui non è soddisfatto e chiede supporto al SOC.
Il file "sospetto" è **iexplore.exe** contenuto nella cartella C :\Programmi \Internet Explorer
(no, non ridete ragazzi)

Come membro senior del SOC ti è richiesto di convincere il dipendente che il file non è maligno.

Possono essere usati gli strumenti di analisi statica basica e/o analisi dinamica basica visti a lezione.
No disassembly no debug o similari

VirusTotal non basta, ovviamente
Non basta dire iexplorer è Microsoft quindi è buono, punto.

Come membro senior del SOC dimostro che il file non è maligno attraverso alcuni strumenti che andremo ad elencare qui in basso:

Analisi Statica Basica

Esaminare le proprietà del file e il suo contenuto senza eseguirlo, per cercare eventuali anomalie.

Svolgimento:

- Strumenti:
  - PEiD per identificare il packer/protector utilizzato (se presente).
  - Exeinfo PE per ulteriori dettagli.

- Procedura:
  1. Apri PEiD o Exeinfo PE.
  2. Carica iexplore.exe e verifica se il file è compresso o protetto.
  3. Esamina le sezioni PE (Portable Executable) per identificare eventuali anomalie.

Analisi Dinamica Basica

Eseguire il file in un ambiente controllato per osservare il suo comportamento.

Svolgimento:

- Strumento: Windows Sandbox, Procmon (Process Monitor).

- Procedura:
  1. Avvia Windows Sandbox.
  2. Copia iexplore.exe nella sandbox.
  3. Esegui Procmon per monitorare le attività del file.
  4. Esegui iexplore.exe e osserva le operazioni di file, registro e rete.
  5. Verifica se il comportamento è congruente con un normale browser Internet Explorer.

Verifica della Firma Digitale

Possiamo verificare che il file iexplore.exe sia firmato digitalmente da Microsoft. Questo è un buon indicatore dell'autenticità del file.

Svolgimento:

- Strumento: Windows Explorer

- Procedura:
  1. Naviga fino al file iexplore.exe in C:\Programmi\Internet Explorer.
  2. Fai clic destro sul file e seleziona "Proprietà".
  3. Vai alla scheda "Firma digitale".
  4. Verifica che ci sia una firma digitale da parte di "Microsoft Corporation".

Il file iexplore.exe situato in C:\Programmi\Internet Explorer è stato verificato come firmato digitalmente da Microsoft Corporation, inoltre utilizzando l'analisi statica e dinamica non hanno rilevato comportamenti anomali o sospetti.

Possiamo concludere che il file non è maligno e fa parte della normale installazione di Internet Explorer.