Malware analysis

Sommario

Traccia S10-L5

Step 1: 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?

Step 2: con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

1. Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)
2. Ipotizzare il comportamento della funzionalità implementata

Step 1

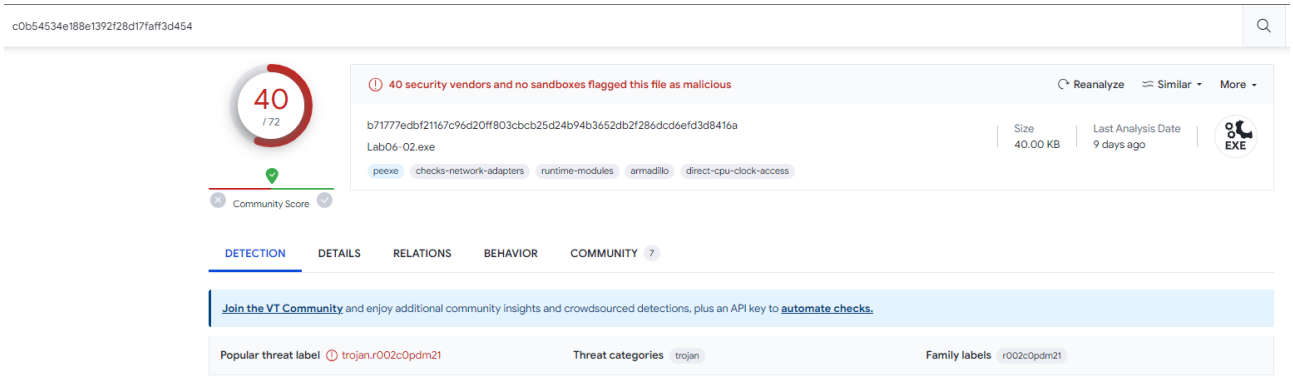
Verifica dell'eseguibile

Per verificare se l'eseguibile indicato nella traccia è effettivamente un malware calcoliamo l'hash tramite md5deep, una utility da riga di commando. L'hash è una stringa alfanumerica univoca che identifica un file, in pratica un'impronta digitale.

Avviamo il command prompt e cambiamo directory con quella contenente l'utility. Inseriamo il comando **md5deep "path del file da hashare".**

```
C:\Documents and Settings\Administrator\Desktop\md5deep-4.3>md5deep "C:\Document
s and Settings\Administrator\Desktop\Esercizio_Pratico_U3_W2_L5\Malware_U3_W2_L5
.exe"
c0b54534e188e1392f28d17faff3d454  C:\Documents and Settings\Administrator\Deskto
p\Esercizio_Pratico_U3_W2_L5\Malware_U3_W2_L5.exe
```

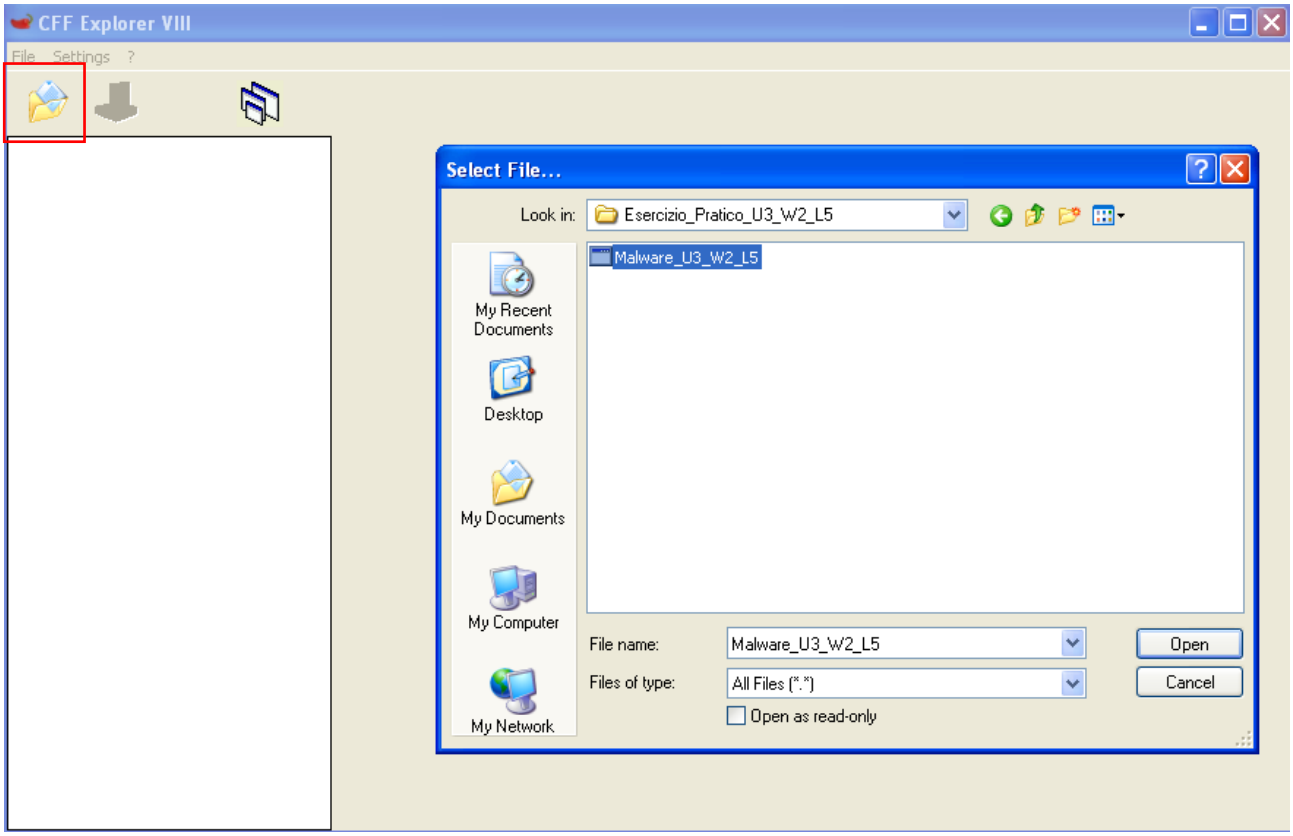
Ora che abbiamo ottenuto l'hash possiamo ricercalo su Virus Total, che mi indica che è un file malevolo.



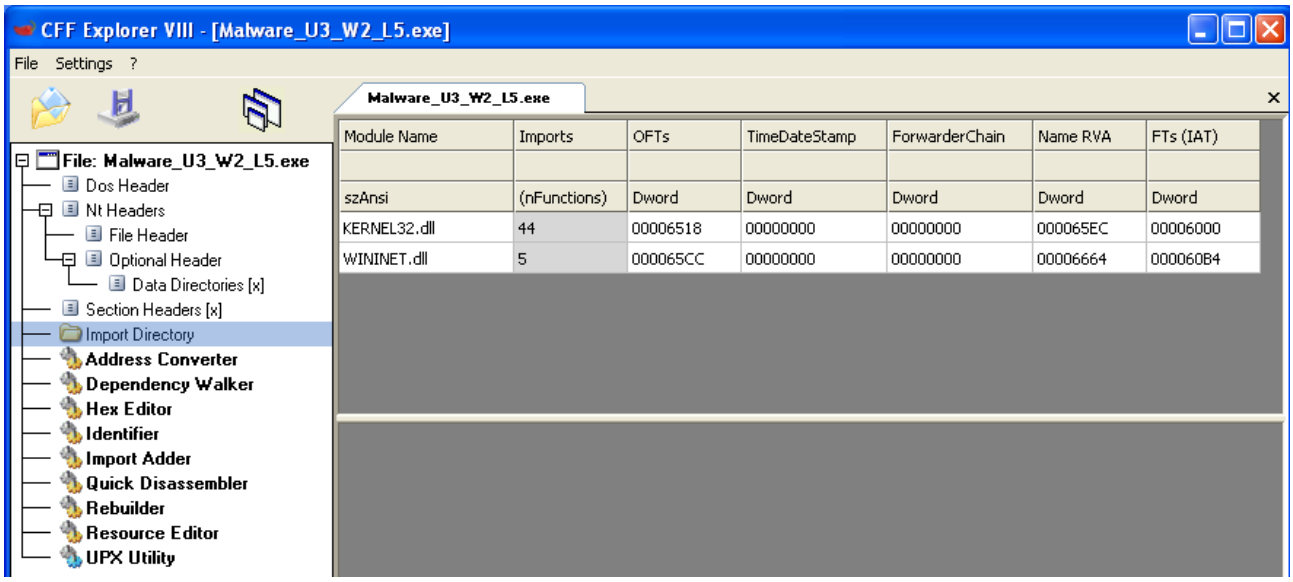
Analisi librerie

Per analizzare le librerie utilizziamo CFF Explorer. Le librerie sono un insieme di funzioni presenti nel sistema operativo che possono essere richiamate da un software.

Avviamo CFF Explorer e clicchiamo sul tasto nel quadrato rosso per aggiungere l'eseguibile del malware da analizzare.



Successivamente clicchiamo su Import directory per visualizzare le librerie.

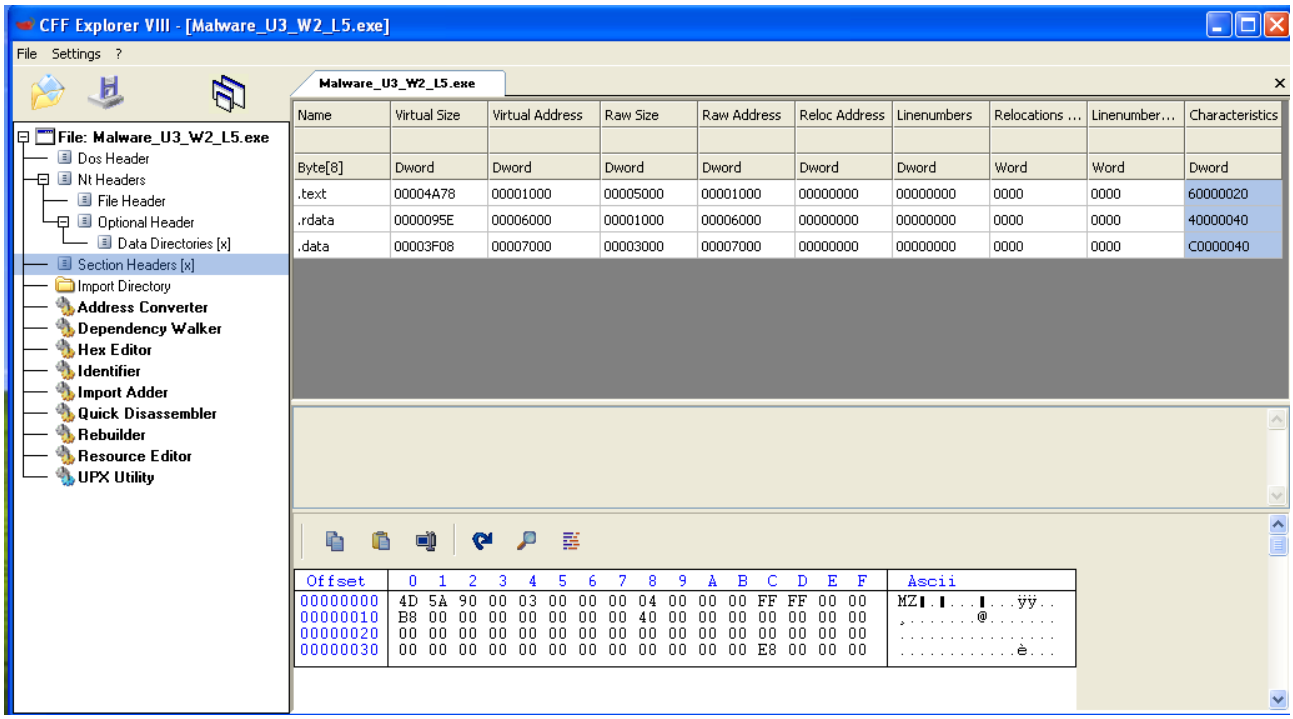


Questo malware utilizza due librerie:

- KERNEL32.DLL: contiene le funzioni principali per interagire con il sistema operativo
- WININET.DLL: contiene le funzioni per l'implementazione di protocolli di rete come HTTP, FTP, NTP

Analisi sezioni

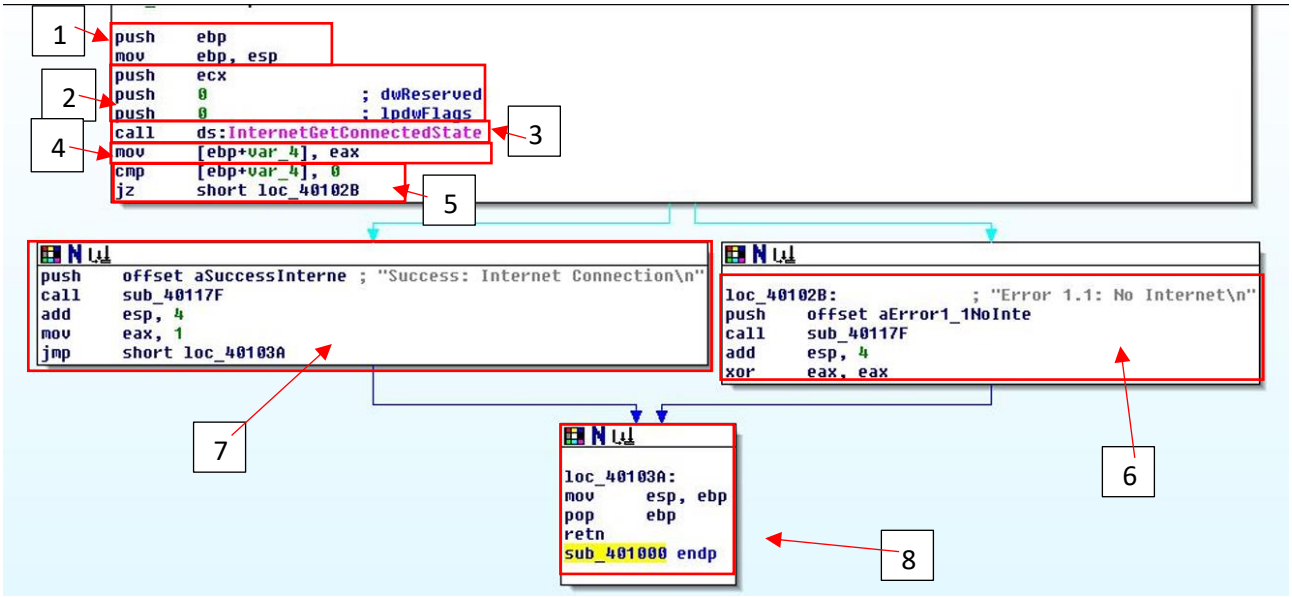
Per analizzare le sezioni dell'eseguibile spostiamoci nella sezione Section Headers.



L'eseguibile è composto di 3 sezioni:

- .text: contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato
- .rdata: include le informazioni sulle librerie e le funzioni importate ed esportate dell'eseguibile
- .data: contiene le variabili globali del programma eseguibile.

Step 2



Analisi del codice

1. Queste istruzioni creano lo stack che verrà utilizzato dalla funzione chiamata successivamente.

-push ebp: salva il valore corrente del puntatore base dello stack

-mov ebp, esp: imposta il puntatore base dello stack per puntare alla cima dello stack

2. Queste istruzioni inseriscono dei valori nello stack da passare alla funzione:

   -push ecx pusha il valore contenuto nel registro ecx nello stack

   -push 0 ; dwReserved pusha il valore 0 nel valore DWORD dwReserved. Questo è un parametro riservato

   -push 0 ; lpdwFlags pusha il valore 0 nel valore DWORD lpdwFlags. Questo parametro riceve informazioni aggiuntive sulla connessione internet. Se la funzione InternetGetConnectedState ritorna TRUE, allora lpdwFlags punta a un valore che specifica il tipo di connessione internet. In questo codice è 0, quindi il codice non sta cercando di ottenere queste informazioni aggiuntive.

3. Chiama la funzione **InternetGetConnectedState** che verifica se c'è una connessione internet disponibile.
4. Sposta il valore di ritorno della funzione **InternetGetConnectedState** (memorizzato in eax) in una variabile locale.
5. Ciclo if:

   -cmp [ebp+var_4], 0: confronta il valore della variabile locale con zero. Se i due valori sono uguali, imposta il flag zero (ZF) a 1.

   -jz short loc_40190B: salta a un'altra posizione nel codice (loc_40190B) se il flag zero (ZF) è impostato a uno.

6. **Nella posizione loc_40190B abbiamo le seguenti istruzioni:**

-loc_40102B ; "Error 1.1: No Internet\n": etichetta per il codice di errore

-push offset aError1_1NoInte: pusha l'indirizzo della stringa di errore nello stack

-call sub_40117f: chiama la funzione sub_40117F (presumibilmente per stampare la stringa)

-add esp, 4: pulisce lo stack dopo una chiamata di funzione. Aumenta il suo valore di 4, effettivamente rimuovendo i 4 byte superiori dello stack.

-xor eax, eax: imposta il valore del registro eax a zero. L'operazione xor di un valore con sé stesso produce sempre zero. È usato per inizializzare il registro eax.

7. **Se il flag zero (ZF) è impostato a zero le istruzioni eseguite sono:**

-**push offset aSuccessInterne ; "Success: Internet Connection\n"**: pusha l'indirizzo della stringa di successo nello stack

-**call sub_40117F**: chiama la funzione sub_40117F (presumibilmente per stampare la stringa)

-add esp, 4: pulisce lo stack dopo una chiamata di funzione. Aumenta il suo valore di 4, effettivamente rimuovendo i 4 byte superiori dello stack.

-mov eax, 1: imposta il registro eax a 1

-jmp short loc_40103: salta alla posizione loc_40103A

8. **Nella posizione loc_40103A abbiamo la pulizia dello stack con le seguenti istruzioni:**

-mov esp, ebp: ripristina il puntatore dello stack (esp) al valore originale (ebp)

-pop ebp: ripristina il valore originale di ebp rimuovendolo dallo stack

-Retn: ritorna dal sottoprogramma, saltando all'indirizzo salvato nello stack

-**sub_401000 endp**: indica la fine del sottoprogramma

Conclusioni

Il codice assembly presentato sembra essere progettato per verificare la presenza di una connessione internet sul computer in cui viene eseguito. Nel caso sia presente una connessione stampa il messaggio **"Success: Internet Connection"**, al contrario stampa **"Error 1.1: No Internet"**.