Progetto S10

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?

2. Quali sono le sezioni di cui si compone il file eseguibile del malware?

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. BONUS fare tabella con significato delle singole righe di codice assembly

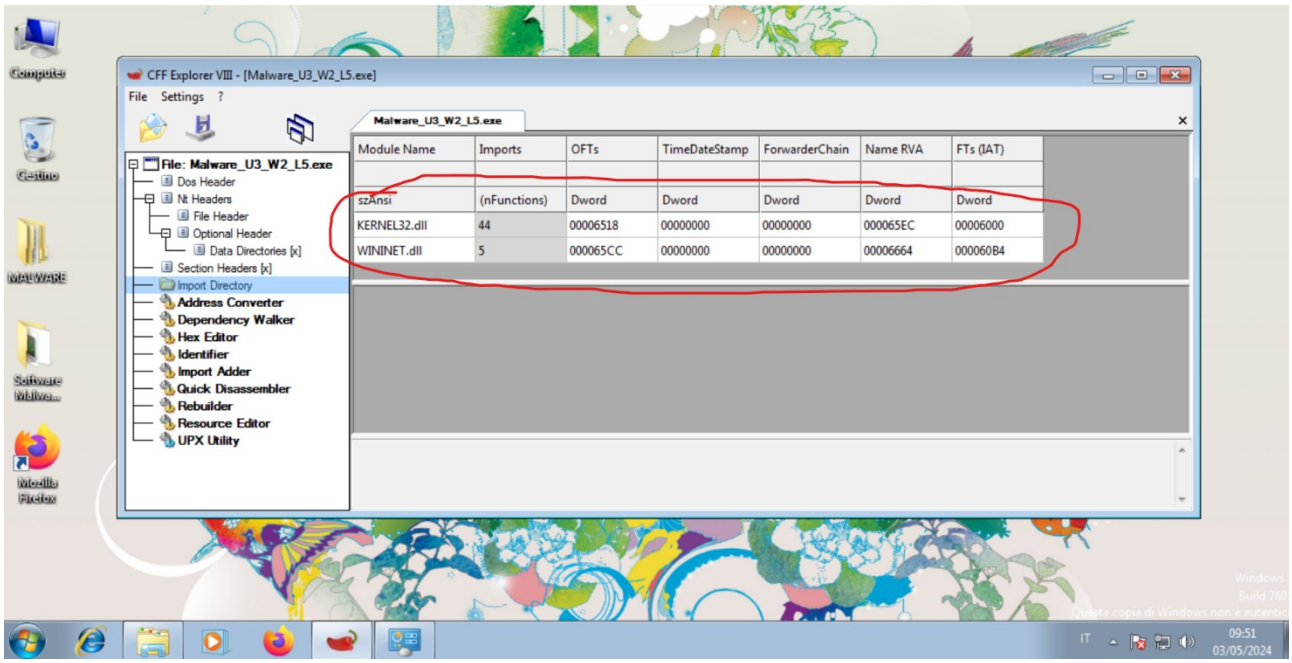
5. BONUS fare tabella con significato delle singole righe di codice assembly

1)

Le librerie che vengono importate dal file eseguibile sono:

Kernel32.dll: questa libreria contiene tutte le funzioni che interagiscono con il sistema operativo ad esempio la gestione della memoria, la gestione dei processi, ecc.

Wininet.dll: questa libreria contiene le funzioni che permettono ai programmi di stabilire connessioni, trasferire dati e gestire i cookie e la cache.



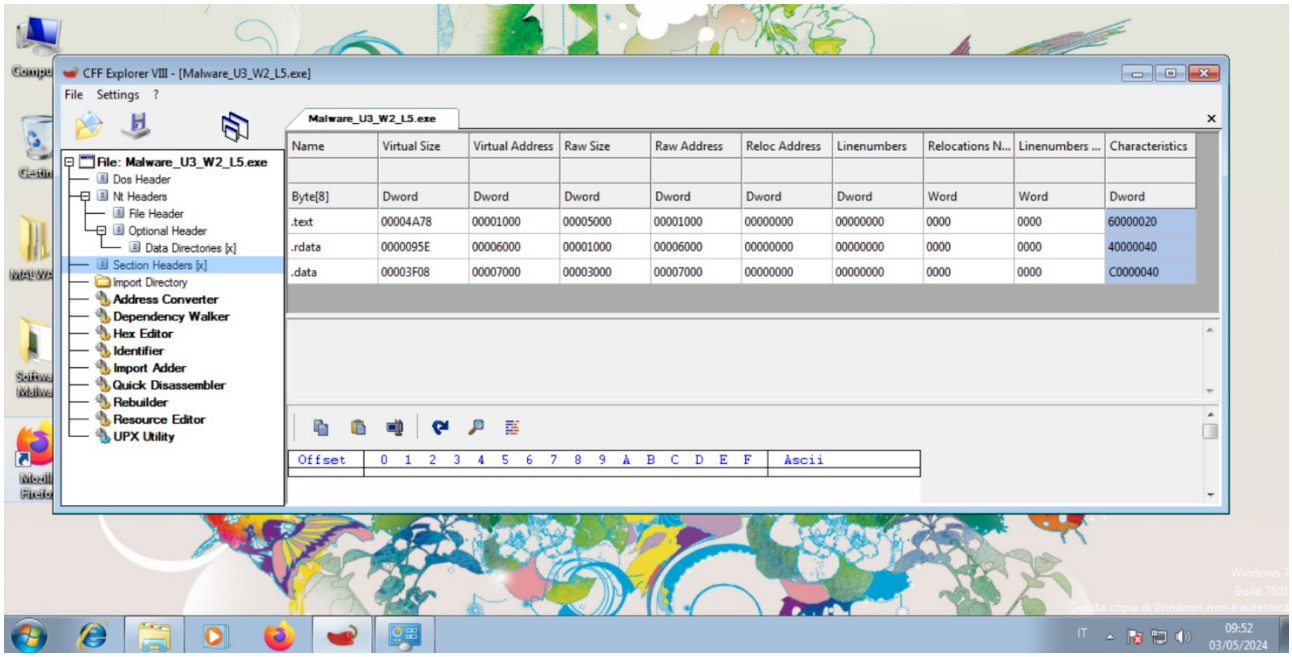
2)

le sezioni del malware sono:

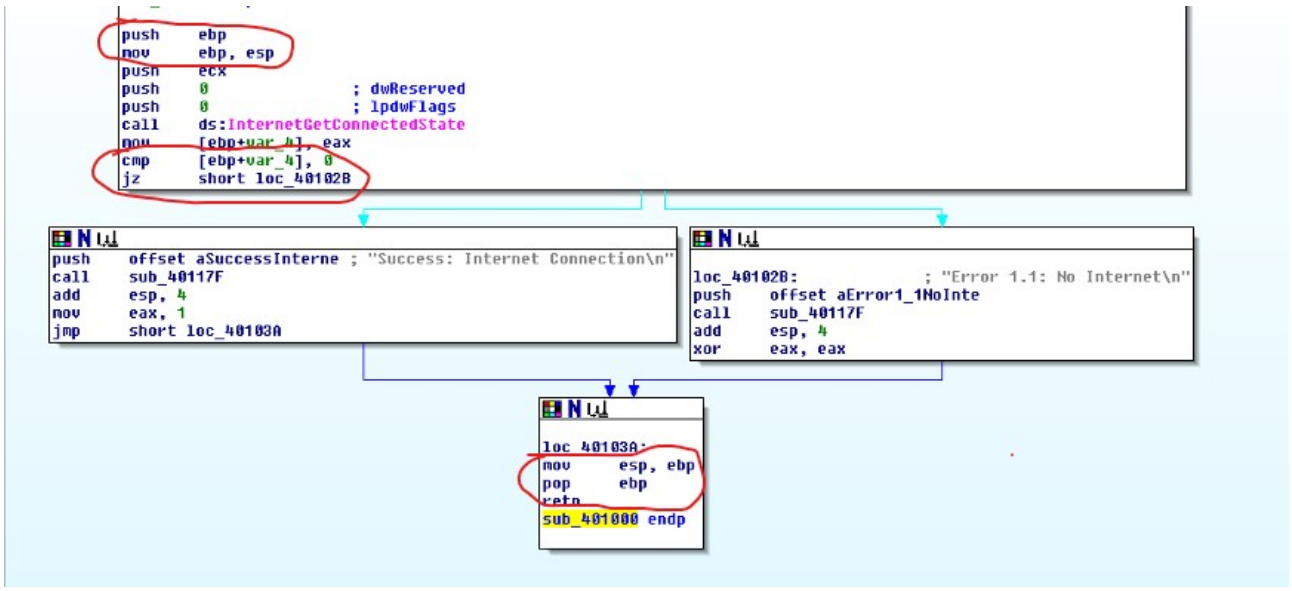
.text: contiene il codice che viene eseguito quando il malware viene avviato

.rdata: contiene i dati di sola lettura che riguardano librerie e funzioni

.data: contiene dati inizializzati del programma



3)



Ci sono 3 costrutti noti:

il primo evidenziato è la creazione di uno stack

il secondo evidenziato è un ciclo if lo si riconosce dalle 2 istruzione cmp: compara due operandi effettuando la sottrazione e in base al risultato cambia lo status flag

jz: fa fare il salto ad una locazione specifica

il terzo è la rimozione dello stack

4)

Possiamo ipotizzare che questo codice cerca una connessione attiva tramite la funzione “internetgetconnectedstate” e poi con il ciclo if controlla il risultato che arriva.

Se è uguale a 0 darà un errore

5)

```
push    ebp
mov     ebp, esp
```
Si punta alla base dello stack e poi si copia il contenuto di esp in ebp

```
push    ecx
push    0
push    0
```
si crea una variabile nello stack

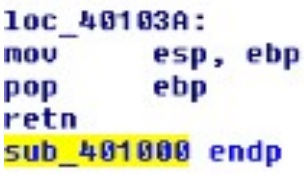
```
call    ds:InternetGetConnectedState
```
Fa una chiamata alla funzione InternetGetConnectedState

```
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```
Controlla se il risultato di [ebp+var_4] è uguale a 0 e se lo è effetua un jump alla sezione di memoria chiamata loc_40102B

```
loc_40102B:             ; "Erro
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```
Qui il push significa che stamperà l errore poi si fa una chiamata alla memoria sub_40117F poi si

esegue una addizione tra esp e il valore 4 e infine con xor si inizializza eax riportandola a 0

Se invece non è uguale a 0 il codice continua e con il push stamperà che la connessione a internet ha avuto successo poi si fa la chiamata sempre alla memoria sub_40117F poi si esegue l addizione tra esp e il valore 4 e insi porta eax a 1



Si copia il contenuto di ebp in esp poi si effettua la rimozione di ebp tramite il pop e infine si chiude la funzione della sub_401000