Progetto S10L5

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?

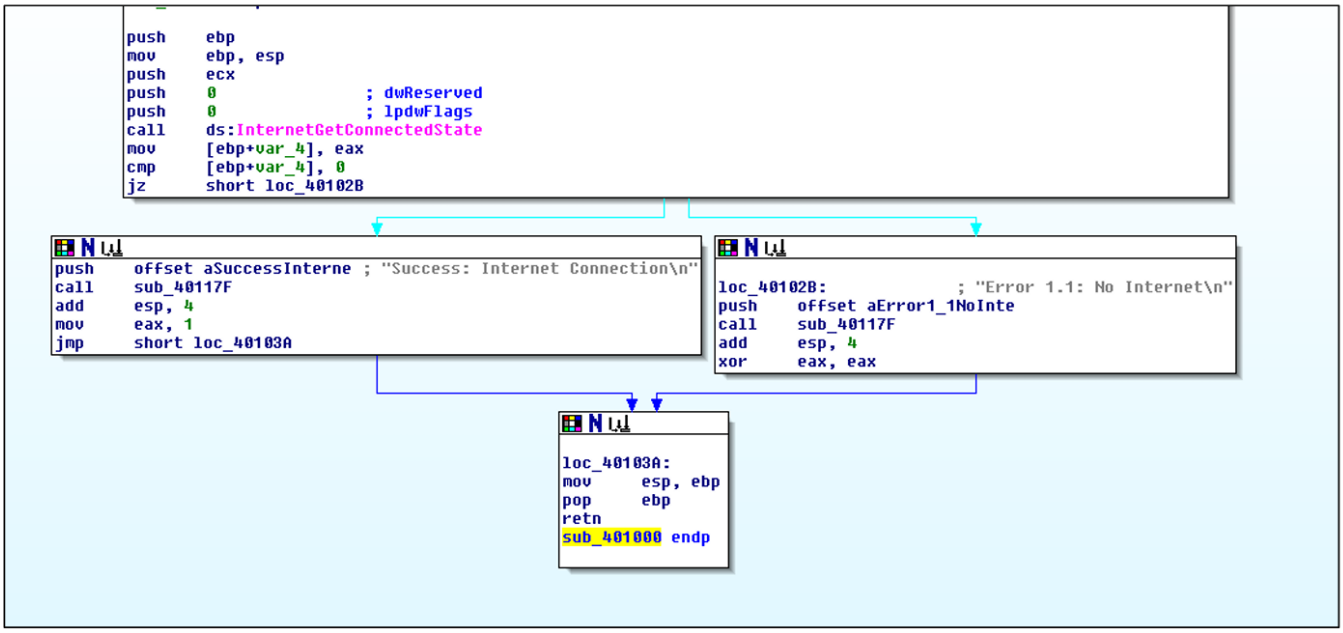
Con riferimento alla figura 1, risponde ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)

4. Ipotizzare il comportamento della funzionalità implementata

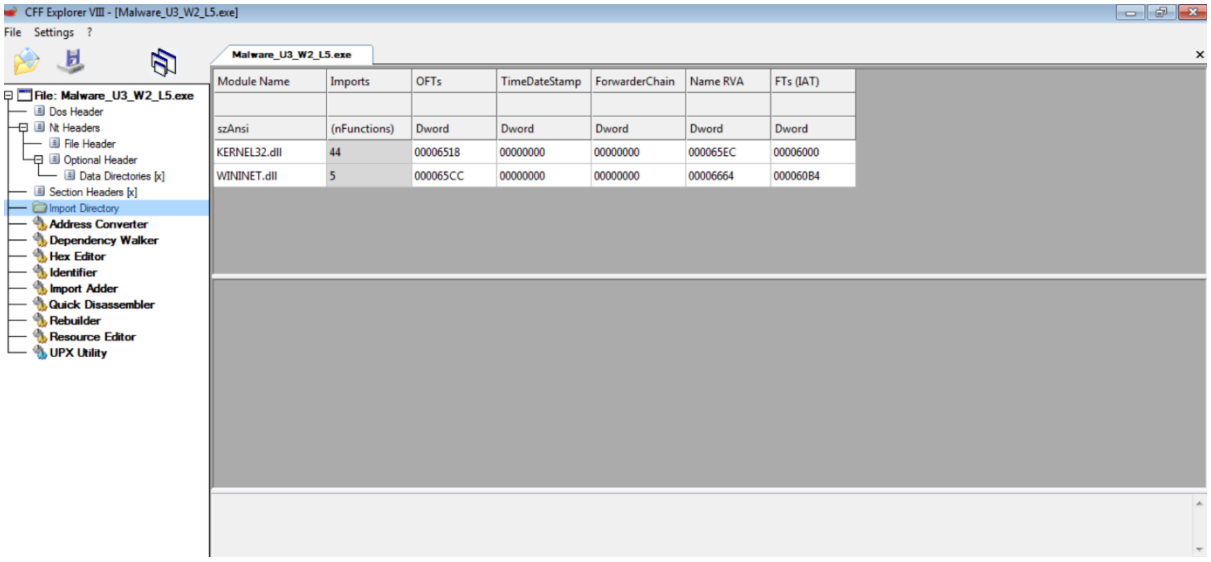
5. BONUS fare tabella con significato delle singole righe di codice assembly

Figura 1



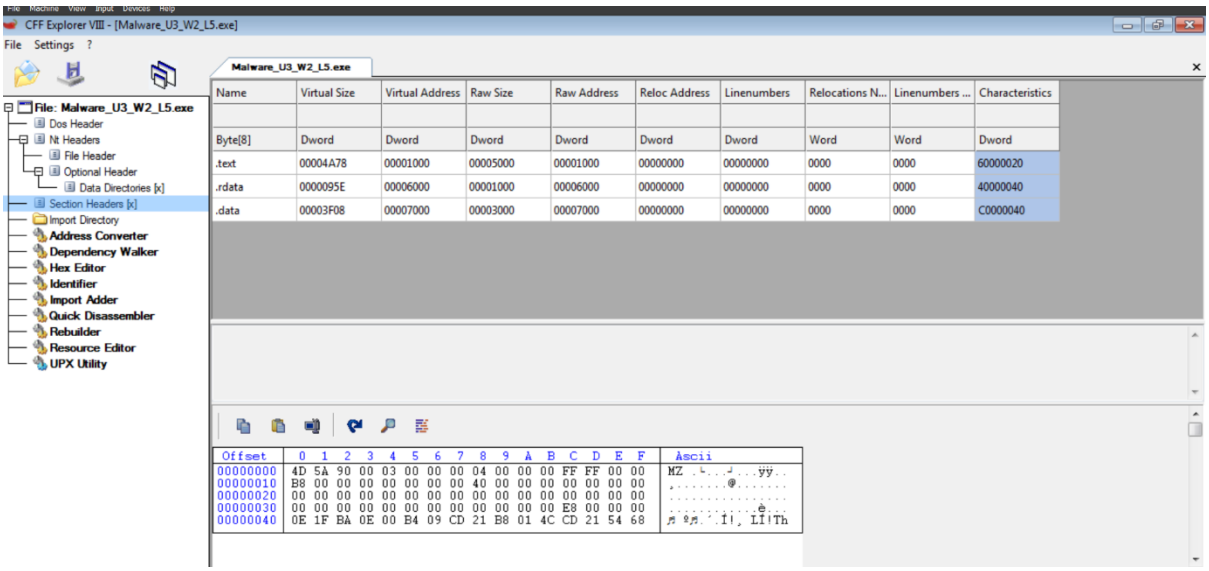
Risoluzione:

Quesito 1:



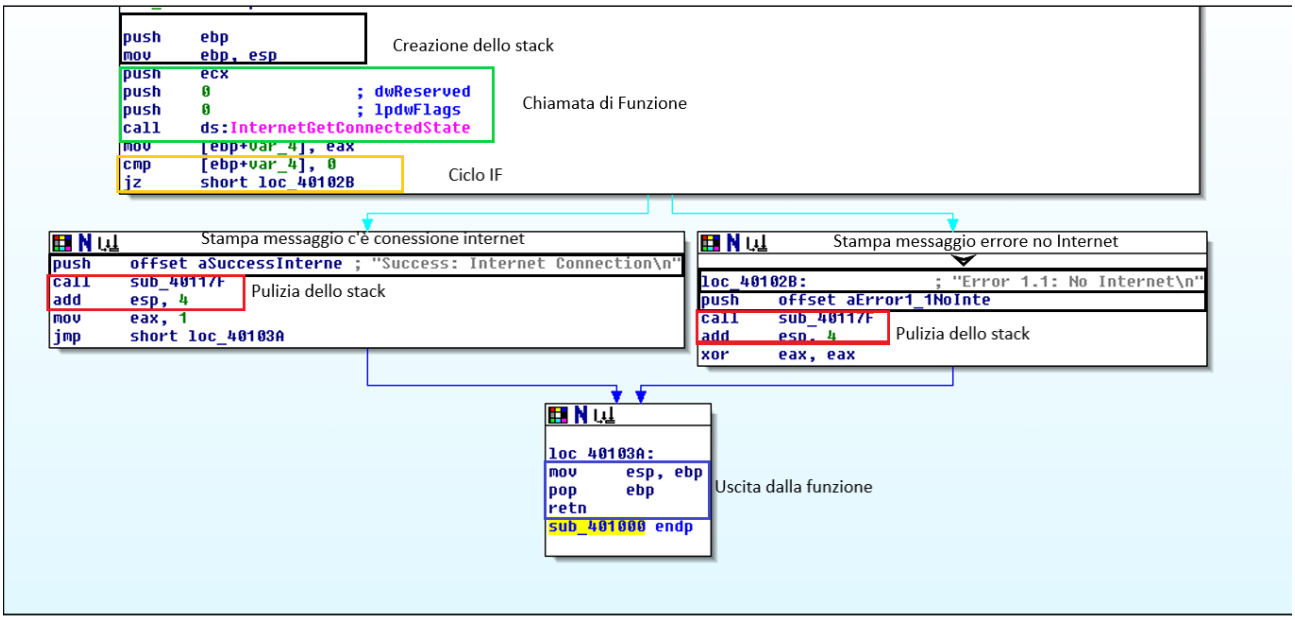
- **KERNEL32.DLL**: Questa è una libreria fondamentale di Windows che contiene funzioni di base per la gestione della memoria, delle operazioni di input/output e dei processi. La presenza di questa libreria è comune in molti eseguibili perché fornisce funzioni di base necessarie per qualsiasi programma Windows.
- **WININET.DLL**: Questa libreria è usata per le funzionalità di rete basate su Internet. La presenza di WININET.DLL potrebbe indicare che il malware è in grado di comunicare su internet o di eseguire operazioni di rete come il download o l'upload di dati.

Quesito 2:



- **.text**: Contiene il codice eseguibile del programma.
- **.rdata**: Include dati di sola lettura, come stringhe costanti e import tables.
- **.data**: Destinata a contenere dati inizializzati e variabili globali.

Quesito 3:



- **Creazione dello stack**: Le istruzioni **push ebp** e **mov ebp, esp** sono tipiche all'inizio di una funzione per salvare il base pointer corrente e impostare il nuovo base pointer per la funzione corrente.
- **Chiamata di Funzione**: L'istruzione **call ds:InternetGetConnectedState** è una chiamata a una funzione esterna che verifica lo stato della connessione internet.
- **Ciclo IF**: La coppia di istruzioni **cmp** e **jz** (jump if zero) forma un costrutto condizionale che verifica se il risultato di **InternetGetConnectedState** indica che non c'è connessione internet.
- **Stampa di messaggi**: I blocchi di codice che iniziano con push offset aSuccessInterne e push offset aError1_1NoIntern preparano i messaggi di successo e di errore da stampare, rispettivamente. Questi blocchi mostrano la gestione di due percorsi condizionali: uno per il successo e l'altro per l'errore.
- **Pulizia dello stack**: Dopo ogni chiamata a subroutine che stampa un messaggio (call sub_40117F), c'è un'istruzione add esp, 4 che serve a ripulire lo stack. Questo è importante per mantenere l'integrità dello stack e per evitare errori di esecuzione.
- **Uscita dalla funzione**: Il blocco che termina con **pop ebp** e **retn** gestisce il ritorno al chiamante, ripristinando il registro **ebp** e poi usando **retn** per saltare indietro al punto del programma da cui la funzione è stata chiamata.

Quesito 4:

l codice sembra verificare lo stato della connessione a Internet del computer su cui viene eseguito. Dopo la chiamata alla funzione **InternetGetConnectedState**, il codice verifica il risultato di quella funzione e se la connessione è attiva, esegue un blocco di codice, altrimenti esegue un altro blocco che probabilmente gestisce l'errore.

Se la connessione è attiva (**cmp** e **jz** non causano un salto), il codice procede a:

- **push offset aSuccessInterne**: Impila un messaggio di successo.
- **call sub_40117F**: Chiama una subroutine, che probabilmente stampa il messaggio.
- **add esp, 4**: Pulisce lo stack dopo la chiamata alla funzione.
- **mov eax, 1**: Imposta **eax** a 1, che potrebbe indicare uno stato di successo.
- **jmp short loc_40103A**: Salta a un'altra sezione di codice.

Se la connessione non è attiva, il flusso di esecuzione salta a **loc_40102B**:

- **push offset sError1_1NoIntern**: Impila un messaggio di errore.
- **call sub_40117F**: Chiama la stessa subroutine, probabilmente per stampare il messaggio di errore.
- **add esp, 4**: Pulisce lo stack.
- **xor eax, eax**: Imposta **eax** a 0, che potrebbe indicare uno stato di errore.

Infine, entrambi i rami del condizionale terminano al blocco **loc_40103A**:

- **mov esp, ebp**: Ripristina il puntatore dello stack a **ebp**.
- **pop ebp**: Ripristina **ebp** al suo valore originale.
- **retn**: Ritorna dalla funzione.

Quesito 5:

Istruzione	Significato
push ebp	Salva il valore corrente del base pointer nello stack.
mov ebp, esp	Imposta il base pointer corrente al top dello stack.
push ecx	Salva il valore corrente del registro ECX nello stack.
push 0	Push del valore 0 nello stack, forse per un parametro della funzione.
push 0	Push di un altro valore 0 nello stack, un altro parametro.
call ds:InternetGetConnectedState	Chiama la funzione che verifica la connessione internet.
mov [ebp+var_4], eax	Salva il risultato della funzione nel local stack frame.
cmp [ebp+var_4], 3	Compara il risultato con il valore 3.
jz short loc_40102B	Salta al codice di errore se il risultato è zero (nessuna connessione).
push offset aSuccessInterne	Prepara il messaggio di successo per la stampa.
call sub_40117F	Chiama la subroutine che probabilmente stampa il messaggio.
add esp, 4	Pulisce lo stack da un valore dopo la chiamata della funzione.
mov eax, 1	Imposta il registro EAX a 1, probabilmente come codice di successo.
jmp short loc_40103A	Salta al codice che segue il blocco condizionale.
push offset aError1_1NoIntern	Prepara il messaggio di errore per la stampa.
call sub_40117F	Chiama la subroutine che probabilmente stampa il messaggio di errore.
add esp, 4	Pulisce lo stack dopo la chiamata della funzione.
xor eax, eax	Imposta EAX a 0, probabilmente come codice di errore.
mov esp, ebp	Ripristina il valore di ESP dal base pointer.
pop ebp	Ripristina il valore originale di EBP dallo stack.
retn	Ritorna dalla funzione.
sub_401090 endp	Fine della subroutine.