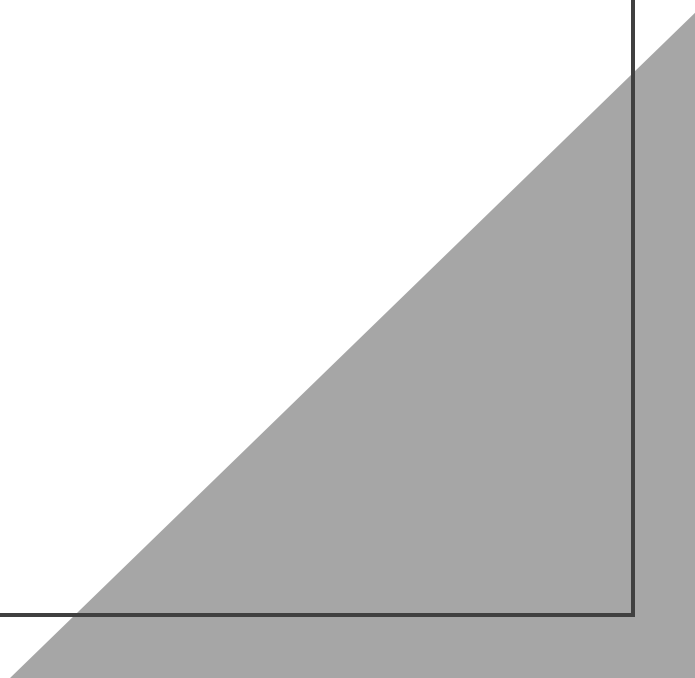


Progetto

Pablo Andres Balbuena Rios



Le Tracce

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:

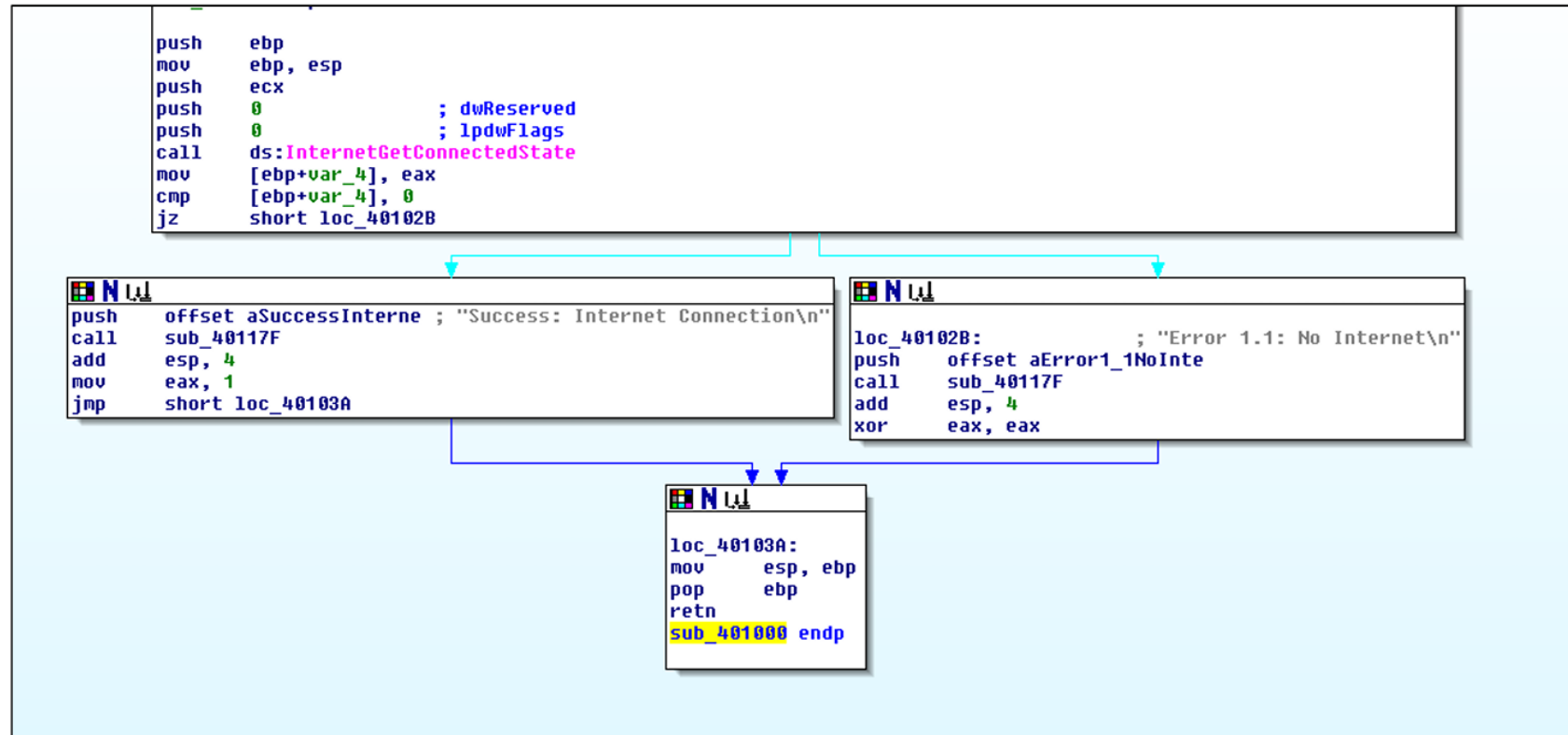
- 1. Quali librerie vengono importate dal file eseguibile?
- 2. Quali sono le sezioni di cui si compone il file eseguibile del malware?

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

- 3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
- 4. Ipotesizzare il comportamento della funzionalità implementata
- 5. BONUS fare tabella con significato delle singole righe di codice assembly

Le Tracce

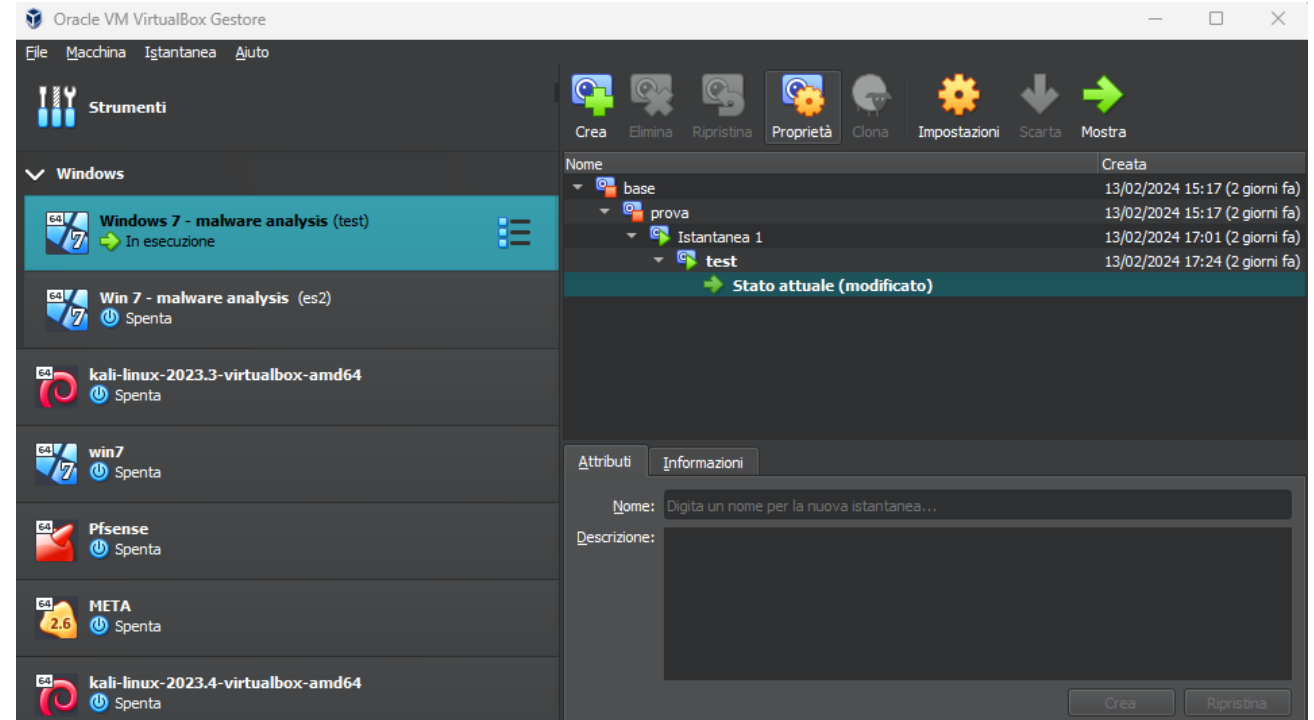
Figura 1



Preparazione

Prima di avviare l'analisi statica del malware, è necessario soddisfare le seguenti condizioni:

- Eliminare le interfacce di rete.
- Disabilitare il controller USB.
- Non condividere cartelle tra Host e guest.
- Creare delle istantanee

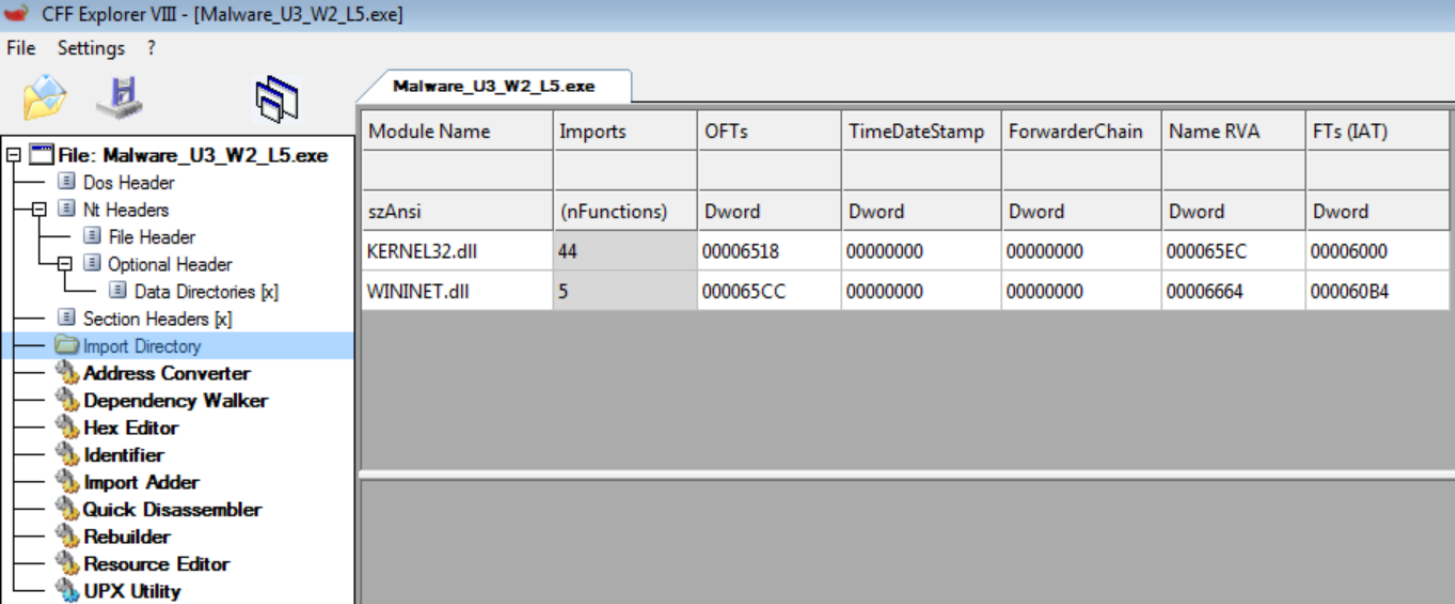


L'implementazione di queste misure di sicurezza riduce il rischio di contaminazione e assicura un ambiente di analisi controllato e sicuro. L'analisi statica può quindi essere condotta in modo più efficace, consentendo una migliore comprensione delle caratteristiche e delle funzionalità del malware senza compromettere la sicurezza dell'ambiente circostante.

Traccia 1

Per condurre un'analisi statica di base su un malware, è essenziale che l'analista utilizzi appositi strumenti al fine di identificare e comprendere la natura del malware.

Per comprendere il comportamento del malware, è fondamentale analizzare le librerie e le funzioni importate dal codice dannoso. A tale scopo, gli analisti spesso ricorrono a strumenti specializzati come il "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

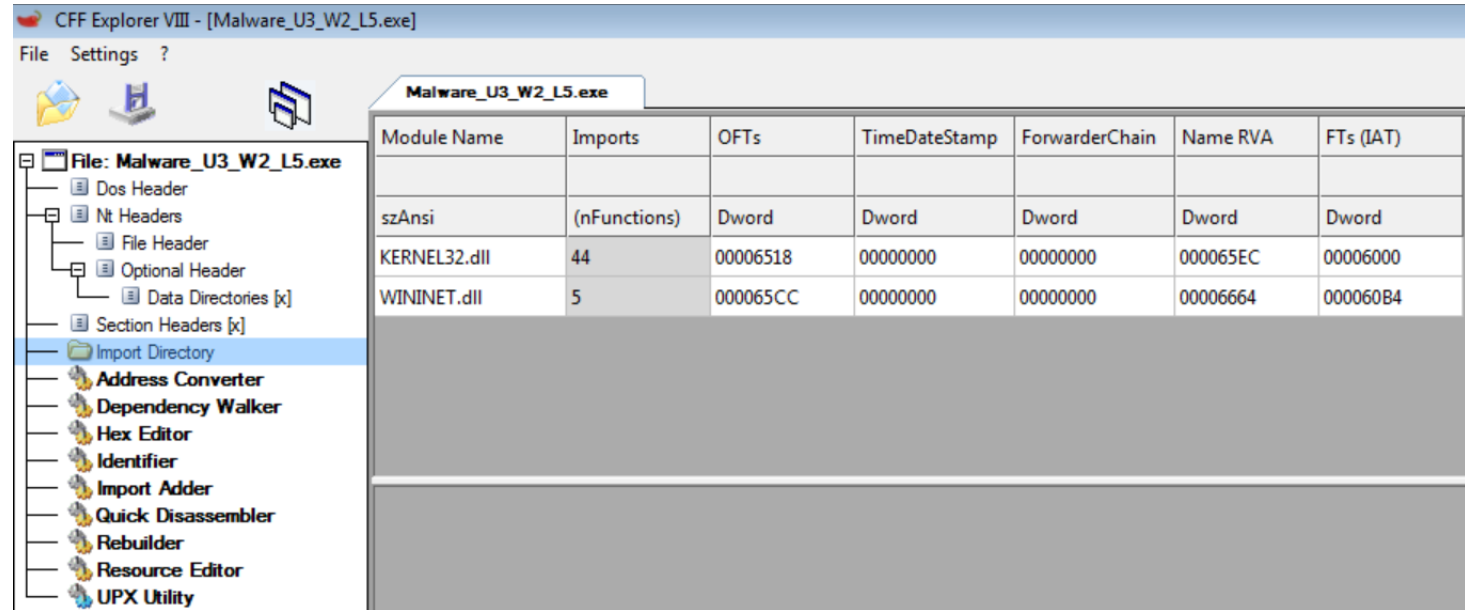
L'analisi statica basica è una metodologia utilizzata nell'ambito della sicurezza informatica per esaminare un file o un software malevolo senza eseguirlo. Questa analisi fornisce una visione dettagliata delle caratteristiche e del comportamento del malware, consentendo agli analisti di comprendere la sua struttura senza doverlo eseguire in un ambiente attivo.

Traccia 1

Durante il processo di analisi statica di un malware, dopo aver caricato il codice dannoso nel tool, è necessario navigare nella directory denominata "Import Directory". Questo passo è cruciale per esaminare attentamente le funzioni e le librerie importate dal malware.

Osserviamo che ci sono due librerie:

- KERNEK32.dll: è una delle principali librerie di sistema di Microsoft Windows che svolge diverse funzioni essenziali per il funzionamento del sistema operativo.
- WININET.dll: è anch'essa una libreria del sistema di Microsoft Windows che fornisce funzionalità per la comunicazione tramite protocolli Internet.



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

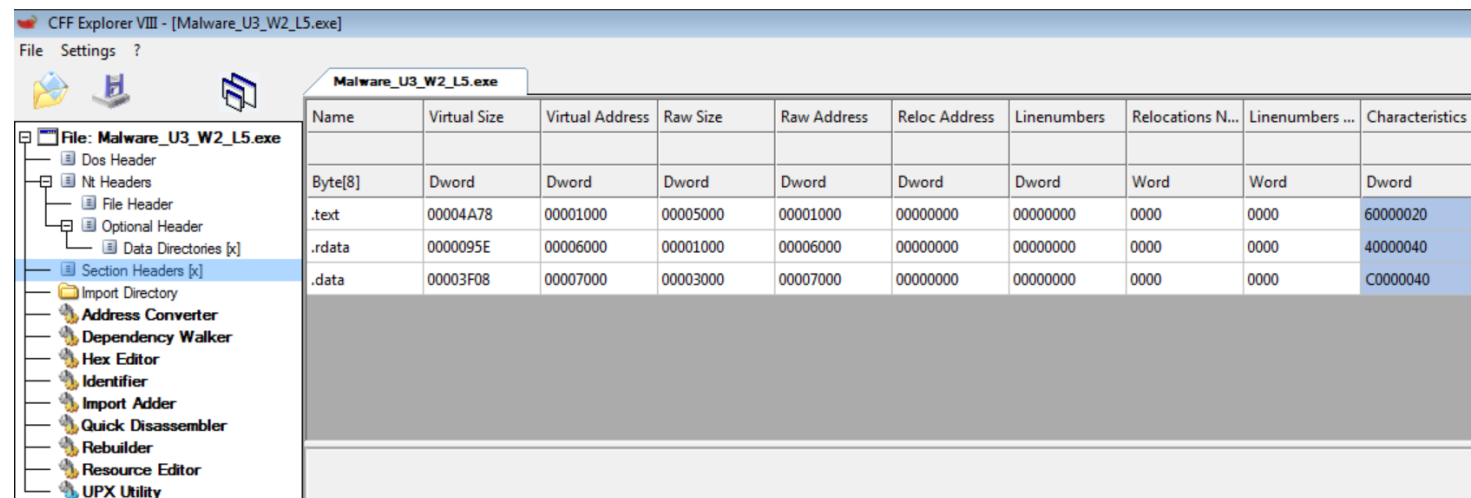
All'interno di questa sezione, gli analisti possono identificare le risorse esterne che il malware utilizza durante l'esecuzione. Queste informazioni sono fondamentali per comprendere come il malware interagisce con il sistema e quali operazioni potrebbe tentare di compiere. Esplorando le funzioni e le librerie importate, gli analisti possono acquisire una visione dettagliata delle attività che il malware potrebbe eseguire all'interno del sistema compromesso.

Traccia 2

Attraverso lo stesso strumento, è possibile individuare le sessioni del malware esplorando la sezione denominata "Selection Headers". Qui, è possibile ottenere informazioni dettagliate sulle sessioni o connessioni che il malware potrebbe instaurare con server remoti o risorse online.

Qui identifichiamo le seguenti sessioni:

- .text: Questa sessione contiene le istruzioni eseguibili dalla CPU e rappresenta l'essenza del programma.
- .rdata: Questa sezione contiene dati che sono destinati a essere letti durante l'esecuzione del programma, ma non modificati.
- .data: Questa sezione contiene variabili globali e dati che possono essere modificati durante l'esecuzione del programma.



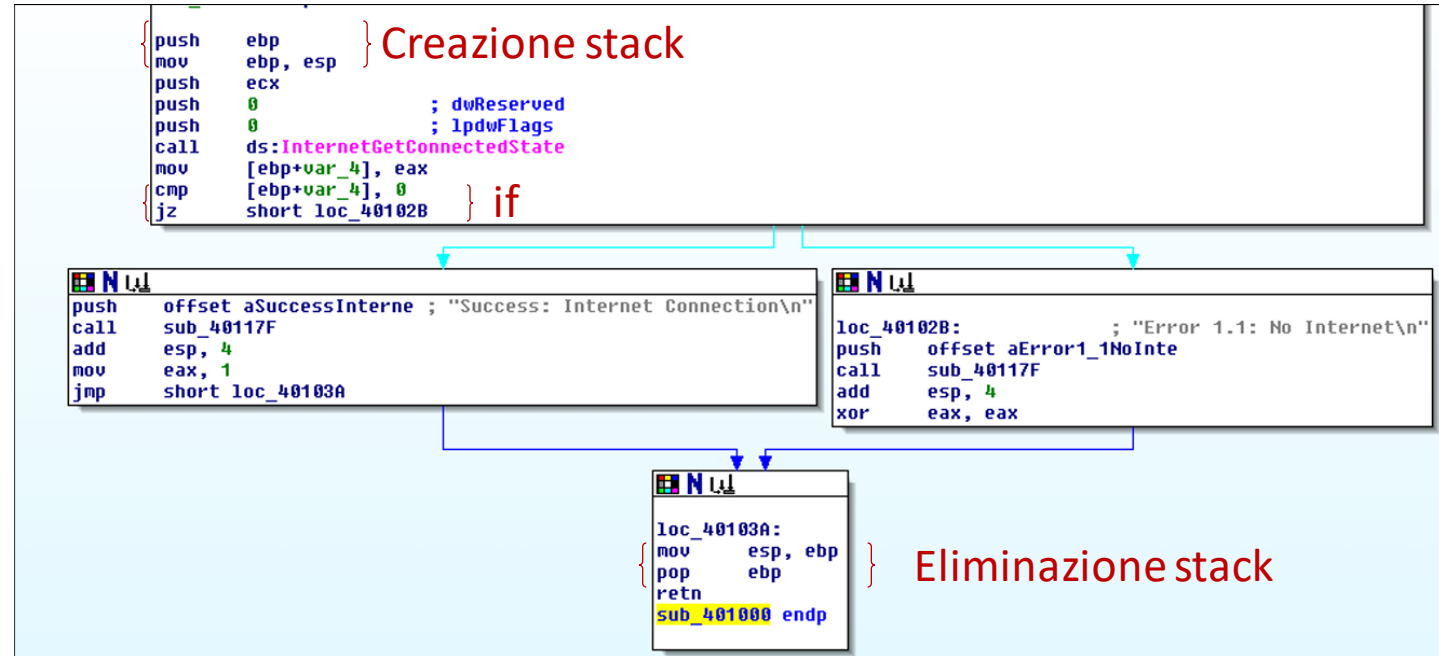
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000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

Questo passo è cruciale per comprendere le attività di rete del malware, inclusi eventuali tentativi di comunicazione con server esterni o il trasferimento di dati attraverso la rete. L'analisi delle "Selection Headers" consente agli investigatori di ottenere una visione più approfondita sulle interazioni del malware con ambienti esterni e facilita l'identificazione di comportamenti dannosi o sospetti.

Traccia 3

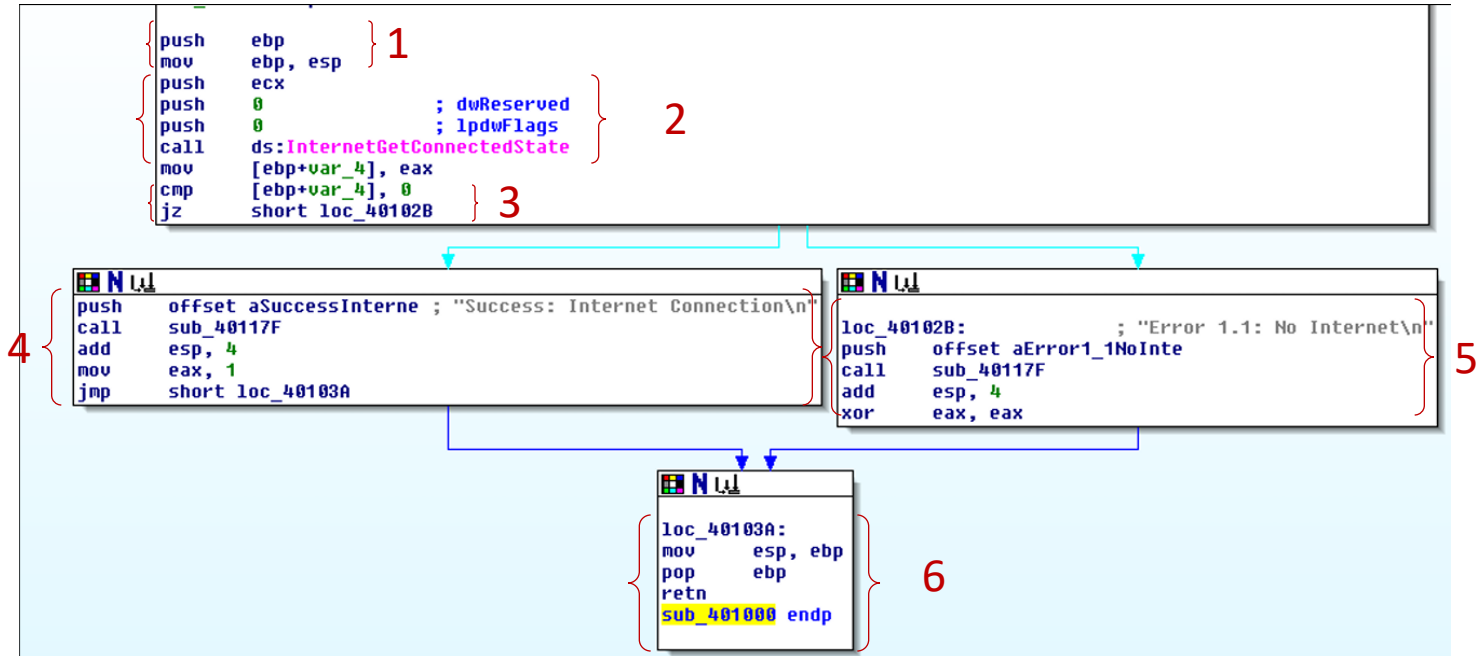
Possiamo identificare 3 costrutti:

- La creazione dello stack: La creazione dello stack è fondamentale per la gestione dell'esecuzione dei programmi, ed ha una struttura dati a pila che viene utilizzata per la gestione del flusso di esecuzione delle funzioni.
- If : il costrutto esegue un blocco di istruzioni se una condizione specificata è vera. Se non lo è, il blocco di istruzioni viene ignorato o potrebbe eseguire un altro blocco di istruzioni.
- Eliminazione dello stack: è spesso associato alla pulizia del frame dello stack di una funzione dopo la sua esecuzione.



Traccia 4

1. Si crea lo stack
2. La funzione prendendo in input 3 parametri, permette di controllare se una macchina ha accesso ad Internet.
3. Esegue una comparazione tra 0 ed il valore restituito dalla funzione
4. Se il confronto ha avuto esito positivo, esegue le seguenti istruzioni, pulisce lo stack dai parametri passati, cambia il valore di eax ad 1, e salta ad un'altra locazione.
5. Se il confronto ha avuto esito negativo, esegue le seguenti istruzioni, pulisce lo stack dai parametri passati cambia il valore di eax ad 0
6. Infine Ripristina il valore di ESP con il valore corrente di EBP e Ripristina il valore di EBP dallo stack ed Restituisce il controllo al chiamante.



Traccia 5 - Bonus

push ebp	Salva il valore corrente del registro EBP nello stack
mov ebp,esp	Crea un nuovo frame nello stack assegnando a EBP il valore corrente di ESP
push ecx	Salva il valore corrente del registro ECX nello stack.
push 0 ; dwReserved	Inserisce un valore zero nello stack, che potrebbe rappresentare un parametro per una funzione successiva.
push 0 ; lpdwFlags	Inserisce un valore zero nello stack, che potrebbe rappresentare un altro parametro per una funzione successiva.
call ds:InternetGetConnectionState	Esegue una chiamata di funzione a InternetGetConnectionState. Permette di controllare se una macchina ha accesso ad Internet.
mov [ebp+var_4],eax	Salva il valore restituito dalla chiamata a InternetGetConnectionState nella variabile locale [ebp+var_4].
cmp [ebp+var_4],0	Compara il valore salvato in [ebp+var_4] con zero.
jz short loc_40102B	Salta a loc_40102B, se il confronto precedente (cmp) ha risultato in zero (Zero Flag ZF)

Traccia 5 - Bonus

Se il confronto ha avuto esito positivo

push offset aSuccesInterne; "Success: Internet connection\n"	Stampa verso l'utente la stringa "Success: Internet connection\n"
call sub_40117F	Effettua una chiamata alla funzione denominata sub_40117F
add esp, 4	Aggiunge 4 byte a ESP (stack pointer), spostando lo stack verso l'alto.
mov eax, 1	Carica il valore 1 nel registro EAX. Questo potrebbe essere utilizzato come valore di ritorno da una funzione
jmp short loc_40103A	Salto incondizionato (jump) alla destinazione specificata loc_40103A

Se il confronto ha avuto esito negativo

loc_40102B:	Etichetta nel codice, rappresenta un punto di destinazione per un salto
push offset aError1_1NoInte	Stampa verso l'utente la stringa "Error: No Internet connection\n".
call sub_40117F	Effettua una chiamata alla funzione denominata sub_40117F
add esp, 4	Aggiunge 4 byte a ESP (stack pointer), spostando lo stack verso l'alto.
xor eax, eax	Esegue un'operazione XOR tra il registro EAX e se stesso, che ha l'effetto di impostare il registro EAX a zero.

Traccia 5 - Bonus

loc_40103A:	Etichetta nel codice, rappresenta un punto di destinazione.
mov esp, ebp	Ripristina il valore di ESP (stack pointer) con il valore corrente di EBP (base pointer). Questa operazione ripristina il puntatore allo stack allo stato precedente alla creazione del frame della funzione.
pop ebp	Estrae il valore dallo stack e lo assegna a EBP, ripristinando il valore originale di EBP prima della chiamata della funzione.
ret	Restituisce il controllo al chiamante della funzione.
sub_401000 endp	Segnala la fine della funzione denominata sub_401000. Il termine "endp" è spesso utilizzato per indicare la fine di una procedura.