



Riccardo Agostino
Monti

WEEK 10 PROJECT

Analisi Dinamica e Statica



INDICE

01

Traccia

02

Librerie Importate

03

Sezioni Malware

04

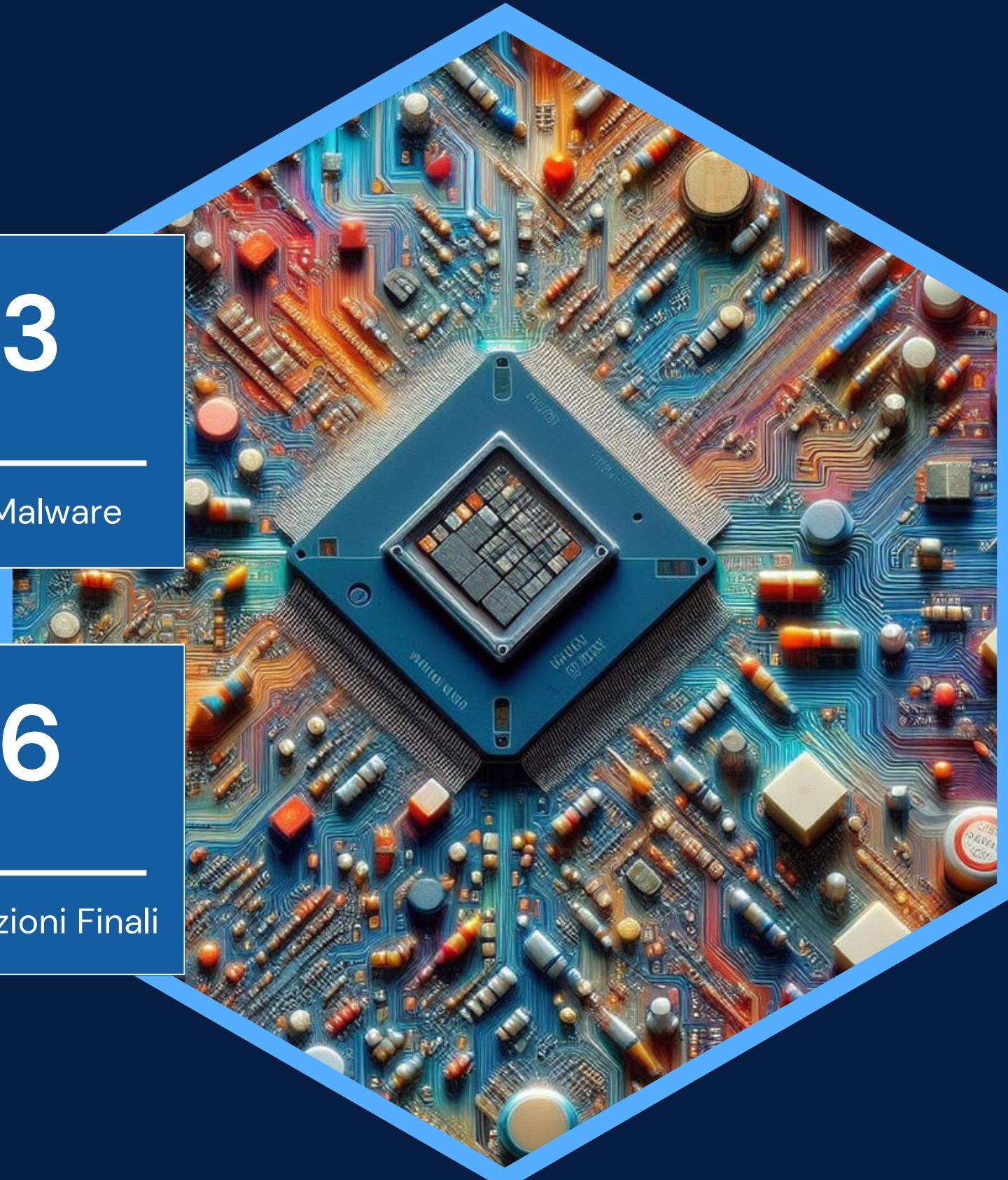
Costrutti Noti

05

Analisi Dettagliata

06

Considerazioni Finali



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:

- Quali librerie vengono importate dal file eseguibile? Quali sono le sezioni di cui si compone il file eseguibile del malware?
- Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:
Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti) Ipotizzare il comportamento della funzionalità implementata



A woman with long brown hair, wearing a white lab coat and a VR headset, is shown from the side. She is interacting with a large, translucent blue robotic hand that is holding a vaccine vial. The background is a laboratory setting with a computer monitor displaying code. A large blue diagonal shape covers the right side of the image.

LIBRERIE IMPORTATE

Per iniziare ad analizzare il nostro malware dobbiamo innanzitutto aprire il programma CFF Explorer presente sul desktop della nostra macchina. Una volta individuata l'icona fare doppio click per aprire CFF Explorer.

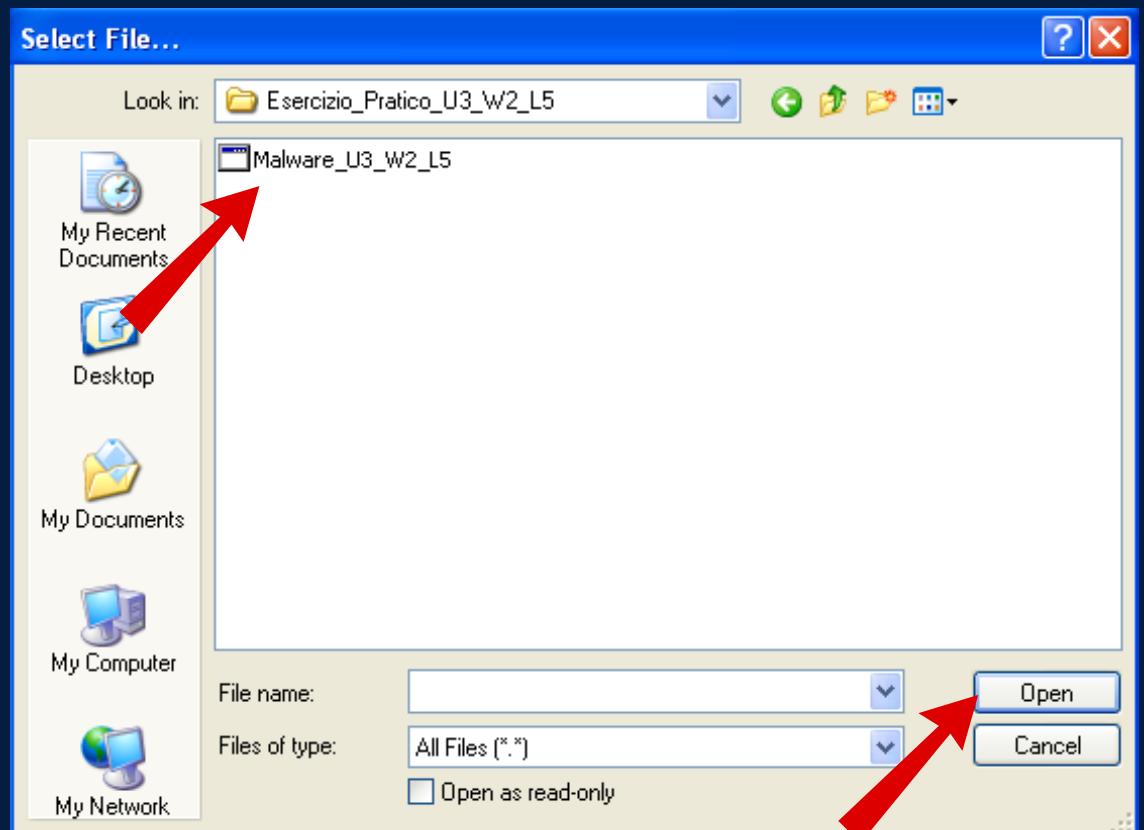


CFF Explorer è uno strumento che ci permette di esplorare, analizzare ed estrarre informazioni da file ed eseguibili. Supporta vari formati di file, come PE, ELF, ELF64, Mach-O, Mach-O64, DLL, DLL32, DLL64 e molti altri. Tra le funzionalità troviamo la possibilità di visionare librerie importate e le sezioni nella quale si divide l'eseguibile a analizzare.

Una volta aperto CFF Explorer doppiamo premere sull'icona dell'esplora risorse in alto a sinistra nella schermata ---->



Una volta cliccato sull''icona il programma ci chiederà di inserire il .exe da analizzare, a questo punto, sempre dall'esplora risorse del programma, rechiamoci sul desktop e cerchiamo la cartella chiamata:
Esercizio_Pratico_U3_W2_L5 click sull'eseguibile e poi click su open in basso a destra della finestra.



A questo punto, dal menù a sinistra ci spostiamo su import directory per verificare quale librerie sono state importate nel programma.

The screenshot shows two windows from the CFF Explorer VIII tool. The left window is titled 'CFF Explorer VIII - [Malware_U3]' and displays a tree view of file headers for 'File: Malware_U3_W2_L1.exe'. The 'Import Directory' node is selected and highlighted in blue. The right window is titled 'Malware_U3_W2_L5.exe' and contains a table of imported modules:

Module Name	Imports	OFTs	TimeStamp	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

Abbiamo così scoperto che le librerie importate sono:

KERNEL32.dll : La libreria fondamentale del sistema operativo Windows. Questa libreria è responsabile della gestione della memoria, delle operazioni si input/output, della gestione dei file per la comunicazione tra il programma e l'hardware e per il funzionamento base dei programmi windows installati.

WININET.dll : La libreria fornisce funzionalità di interfaccia per operazioni relative a Internet, come transazioni HTTP e FTP. Si occupa quindi di tutte le funzioni di internet, l'interfaccia con le applicazioni e la gestione dei possibili errori di comunicazione.

A woman with long dark hair, wearing a white lab coat and a VR headset, is shown from the side. She is interacting with a virtual interface that displays a robotic hand holding a syringe. The background is a laboratory setting with a computer monitor showing code. A large blue diagonal shape covers the right side of the image.

SEZIONI MALWARE

A questo punto, sempre dal menù a sinistra di CFF Explorer, ci spostiamo su Section Headers per verificare in quali sezioni è diviso il malware



Abbiamo così scoperto che il malware è diviso tra le sezioni.

.text : la sezione «text» contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.

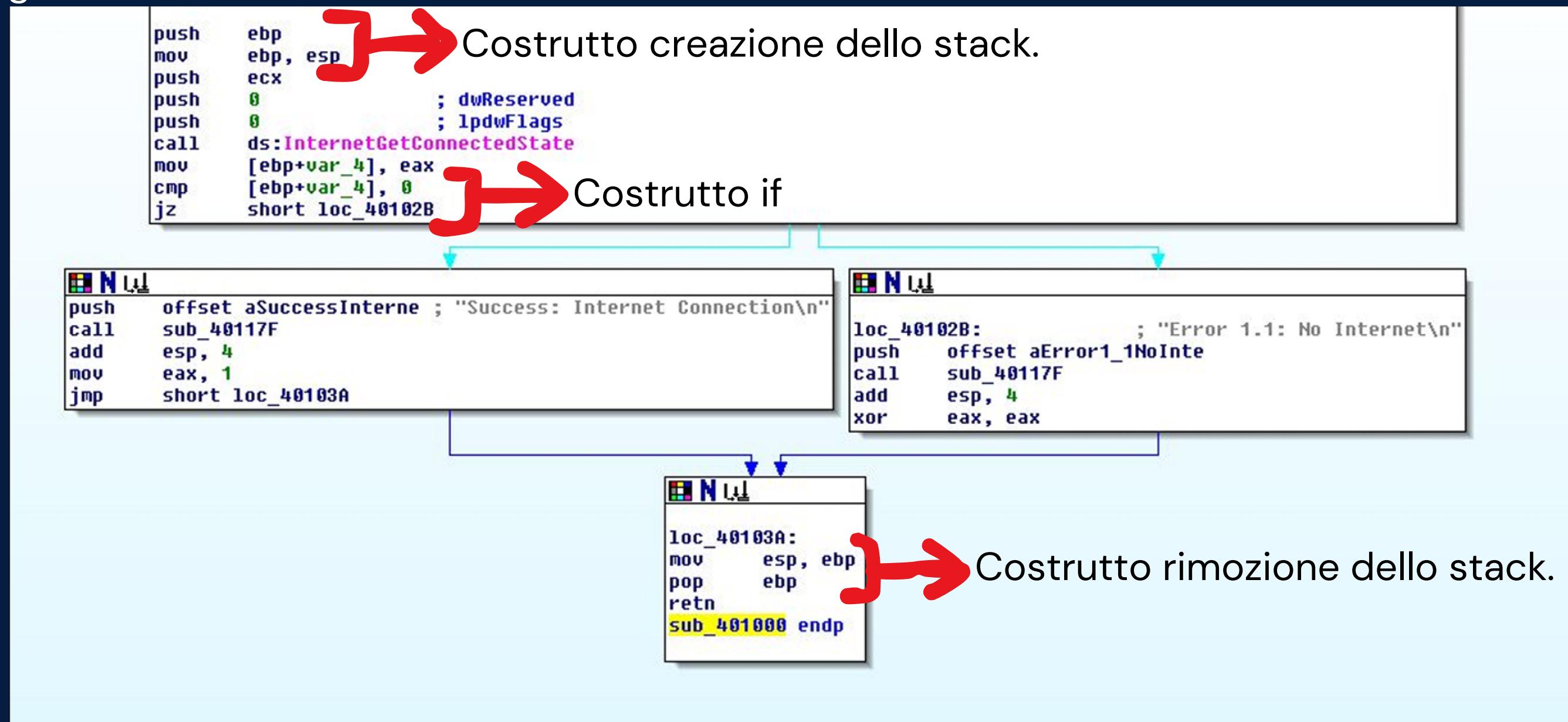
.rdata : la sezione «rdata» include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, informazione che come abbiamo visto possiamo ricavare con CFF Explorer

.data : la sezione «data» contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Ricordate che una variabile si dice globale quando non è definita all'interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione dell'eseguibile.



COSTRUTTI NOTI

Cominciamo quindi ad analizzare il codice Assembly fornito e segnalare i costrutti noti:



Dopo aver visionato il codice abbiamo rilevato 3 costrutti noti, molto diffusi:

1-Costrutto creazione dello stack.

Tale costrutto è formato dalle istruzioni <<push ebp>> e <<mov ebp, esp>>

2-Costrutto if

Tale costrutto è formata dalle istruzioni <<cmp [ebp+var_4], 0>> e <<jz short loc_40102>> ed è utilizzato per controllare se la connessione è presente o meno, vedremo dopo come.

3- Costrutto rimozione dello stack

Tale costrutto è formato dalle istruzioni <<mov esp, ebp>> e pop ebp



ANALISI DETTAGLIATA

Dopo aver evidenziato i costrutti noti mi è sembrato opportuno effettuare un analisi dettagliata del codice assembly, anche se non richiesta, andiamo ad analizzare quindi il codice riga per riga.

<<push ebp>> ; salva il valore del registro ebp nello stack

<<mov ebp, esp >>; copia il valore del registro esp nel registro ebp

Insieme formano il costrutto per la creazione dello stack

<<push ecx>> ; salva il valore del registro ecx nello stack per passarlo come primo parametro alla funzione

<<push 0>> ; mette il valore 0 nello stack come secondo parametro della funzione InternetGetConnectedState

<<push 0>> ; mette il valore 0 nello stack come terzo parametro della funzione InternetGetConnectedState

<<call ds:InternetGetConnectedState>> ; chiama la funzione InternetGetConnectedState, che restituisce 1 se il computer è connesso a Internet e 0 altrimenti, e mette il risultato nel registro eax

<<mov [ebp+var_4], eax>> ; copia il valore del registro eax, che contiene il ritorno della funzione precedente, nella variabile locale var_4, che si trova a 4 byte di distanza dal registro ebp

<<cmp [ebp+var_4], 0>> ; confronta il valore della variabile locale var_4 con 0

<<jz short loc_40102B>> ; salta alla locazione loc_40102B se il valore della variabile locale var_4 è uguale a 0, cioè se il computer non è connesso a Internet

<<push offset aSuccessInterne>> ; mette l'indirizzo della stringa "Success: Internet connection available" nello stack come parametro della funzione sub_40117F

<<call sub_40117F>> ; chiama la funzione sub_40117F, che stampa la stringa passata come parametro

<<add esp, 4>> ; aggiusta il valore del registro esp dopo aver rimosso il parametro dalla cima dello stack

<<mov eax, 1>> ; mette il valore 1 nel registro eax come valore di ritorno della funzione

<<jmp short loc 40103A>> ; salta alla locazione loc_40103A per terminare la funzione

<<loc_40102B:>>; etichetta che indica la locazione da raggiungere se il computer non è connesso a Internet

<<push offset aError1_1_NoInte>> ; mette l'indirizzo della stringa "Error 1.1: No Internet connection" nello stack come parametro della funzione sub_40117F

<<call sub_40117F>> ; chiama la funzione sub_40117F, che stampa la stringa passata come parametro

<<add esp, 4>> ; aggiusta il valore del registro esp dopo aver rimosso il parametro dalla cima dello stack

<<xor eax, eax >>; mette il valore 0 nel registro eax come valore di ritorno della funzione

<<loc_40103A:>> ; etichetta che indica la locazione da raggiungere per terminare la funzione

<<mov esp, ebp>> ; ripristina il valore originale del registro esp

<<pop ebp>> ; ripristina il valore originale del registro ebp

<<retn>> ; ritorna al chiamante

<<sub_401000 edp>> ; etichetta che indica la fine della funzione



CONSIDERAZIONI FINALI

Il codice è una funzione che verifica se il computer è connesso a Internet oppure no, e stampa un messaggio in caso di successo o di errore a seconda del risultato. Il codice usa la funzione InternetGetConnectedState, che è una funzione della libreria WININET.dll, analizzata nelle slide. Il codice utilizza anche la funzione sub_40117F, che non è definita nel codice ma che è usata per stampare le stringhe passate come parametri, da questo si deduce che si tratta di una funzione di stampa. La funzione potrebbe essere usata da un attaccante per verificare che la connessione a internet sia presente prima di procedere all'attacco vero e proprio del computer che ospita il malware così da non far allarmare immediata alcuni antivirus che restano perennemente funzionanti in background.

