Manuel Di Gangi

S10_L5

Progetto - Analisi malware

29 marzo 2024

INDICE

Traccia

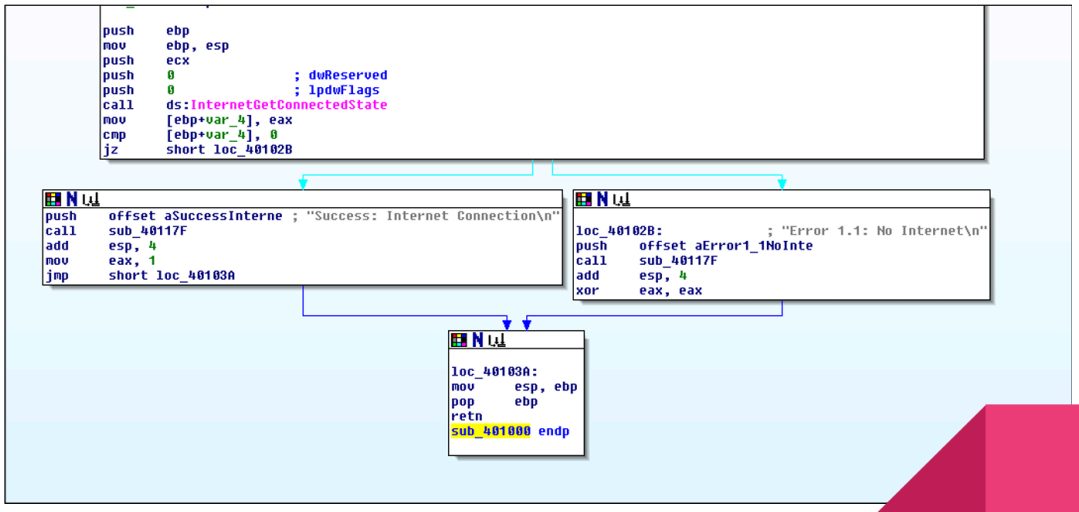
Con riferimento al file Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5 » sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

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

   Di entrambi dare una breve spiegazione

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



1. Librerie utilizzate dal malware

Con l'ausilio del tool CFF Explorer troviamo che il malware richiama le seguenti librerie:

- **KERNEL32.dll**: contiene numerose funzioni essenziali per il funzionamento del sistema operativo e delle applicazioni Windows. Queste funzioni includono la gestione della memoria, la gestione dei processi e dei thread, le operazioni di input/output, la gestione dei file, la gestione dei tempi e delle date, la gestione delle risorse, e molte altre ancora.
- **WININET.dll**: contiene le funzioni per l'implementazione di alcuni protocolli di rete come HTTP, FTP, NTP. Essenzialmente, offre un'API per consentire alle applicazioni Windows di effettuare richieste e gestire le operazioni di rete, come il download di file da Internet, l'invio di richieste HTTP ai server web e l'accesso ai servizi FTP.

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

Osservando l'immagine sopra possiamo osservare che la libreria KERNEL32 importa ben 44 funzioni tra le quali troviamo:

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW
000068FC	000068FC	01BF	LCMapStringA

- **Sleep:** è una funzione di sistema di Windows utilizzata per mettere in pausa l'esecuzione di un thread per un determinato periodo di tempo per

consentire ad altri thread di essere eseguiti nel sistema durante quel periodo di tempo.

- **CloseHandle:** è un'API utilizzata per chiudere un handle (un tipo di riferimento o puntatore) a un oggetto kernel. In sostanza, questa funzione viene utilizzata per rilasciare le risorse utilizzate da un handle dopo che è stato utilizzato.
- **GetProcAddress:** ha la funzione di libreria dinamica che consente ai programmatori di ottenere un puntatore a una funzione all'interno di una DLL caricata in memoria. Questo è spesso utilizzato quando si desidera chiamare una funzione esportata da una DLL in modo dinamico durante l'esecuzione del programma, anziché linkarla staticamente durante la compilazione.
- **VirtualAlloc:** è una funzione di programmazione utilizzata principalmente nella piattaforma Windows per riservare o allocare memoria virtuale per un processo. Essa consente ai programmatori di riservare una porzione di memoria virtuale senza necessariamente allocare memoria fisica corrispondente nello spazio di archiviazione fisica (RAM o disco rigido).
- **VirtualFree:** è una funzione di Windows API utilizzata per liberare la memoria allocata dinamicamente da un processo.

La libreria WININET importa 5 funzioni tra le quali troviamo:

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006640	00006640	0071	InternetOpenUrlA
0000662A	0000662A	0056	InternetCloseHandle
00006616	00006616	0077	InternetReadFile
000065FA	000065FA	0066	InternetGetConnectedState
00006654	00006654	006F	InternetOpenA

- **InternetOpenUrl**: è utilizzata per aprire una connessione Internet e recuperare il contenuto di una risorsa identificata da un URL specificato. Questa funzione è ampiamente utilizzata nelle applicazioni Windows per scaricare dati da risorse online, come pagine

web, file e altro ancora. Essenzialmente, permette alle applicazioni di effettuare richieste HTTP o FTP per ottenere contenuti da Internet.

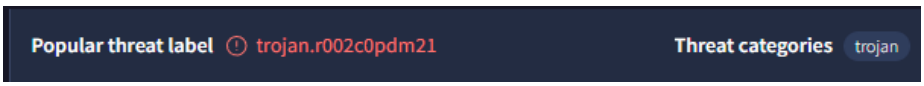
- **InternetCloseHandle:** è utilizzata per chiudere un gestore di risorse Internet precedentemente aperto o creato da altre funzioni della libreria WinINet. L'utilizzo di questa funzione è importante per rilasciare correttamente le risorse allocate e liberare la memoria associata, garantendo un corretto funzionamento del programma e prevenendo eventuali perdite di memoria.
- **InternetReadFile:** è utilizzata per leggere dati da una risorsa Internet identificata da un handle. Essa consente alle applicazioni di recuperare dati da risorse online, come pagine web o file su server FTP.
- **InternetGetConnectedState:**l è una funzione utilizzata per determinare lo stato della connessione di rete del sistema, consente alle applicazioni di verificare se il sistema è connesso a Internet o a una rete locale.
- **InternetOpen:** questa funzione crea un handle che rappresenta una sessione di comunicazione, fornendo un punto di ingresso per l'accesso alla rete. Tramite questa sessione, l'applicazione può effettuare operazioni di rete come il download di risorse da Internet, l'invio di richieste HTTP e altre attività di comunicazione.

2. Sezioni di cui è composto il malware

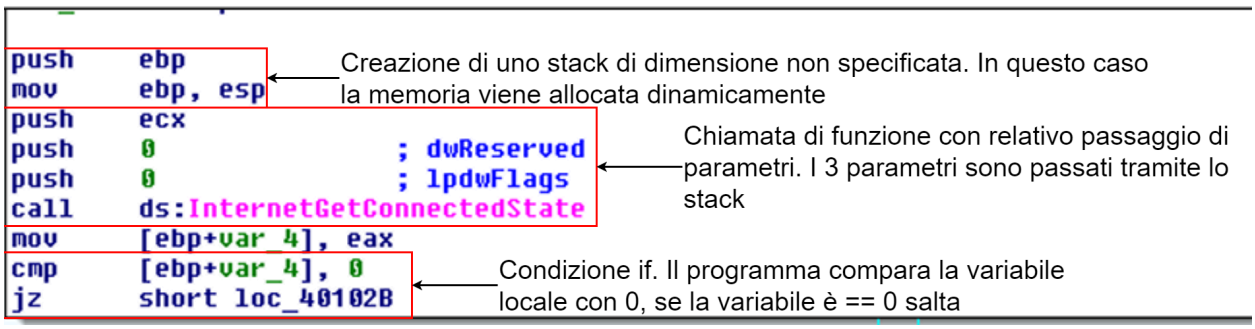
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000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

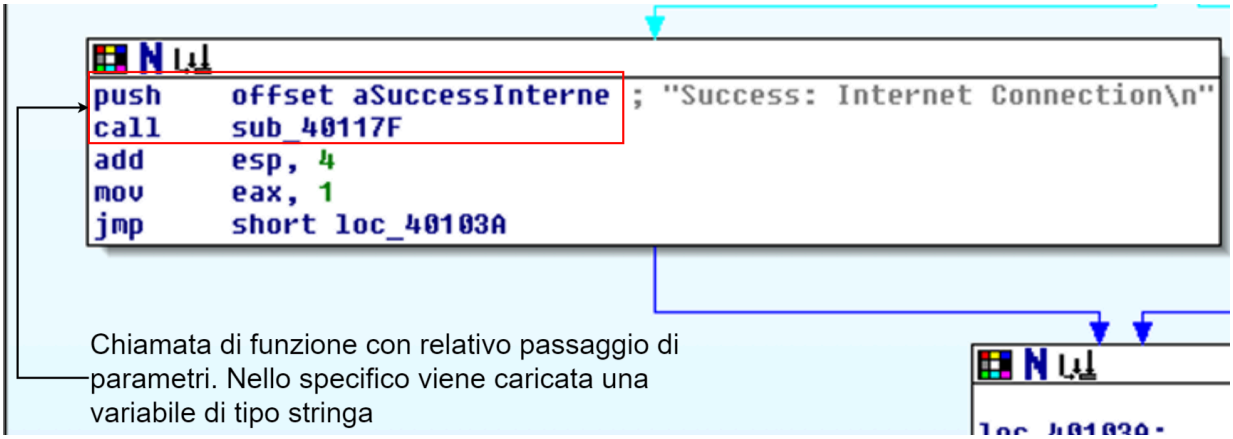
- **.text**: contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.
- **.rdata**: include 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**: contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. 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 all'interno dell'eseguibile.

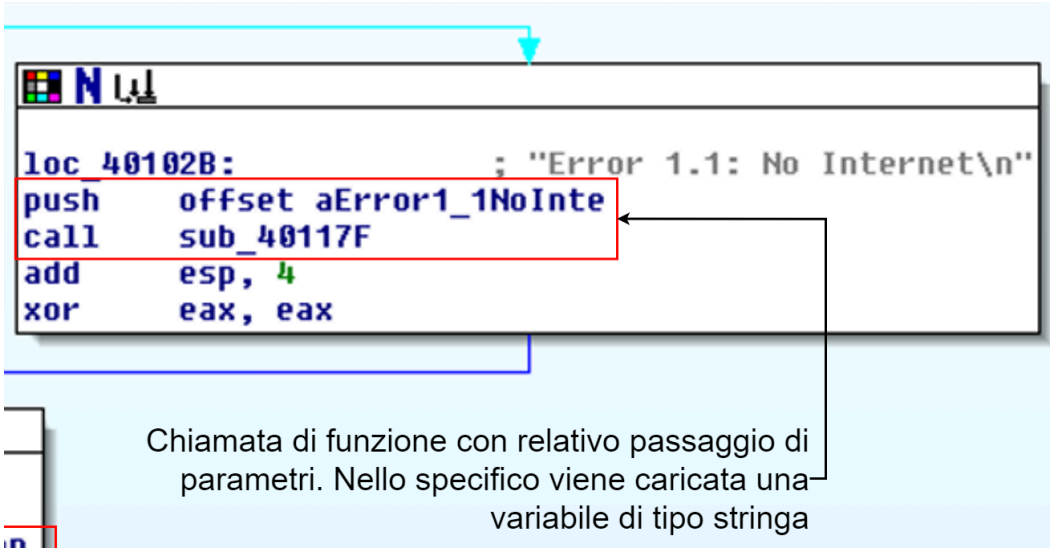
Effettuando una ricerca su virus totale tramite l'hash ricavato da CFF Explorer è stato riscontrato che il malware in questione potrebbe essere un **Trojan.**

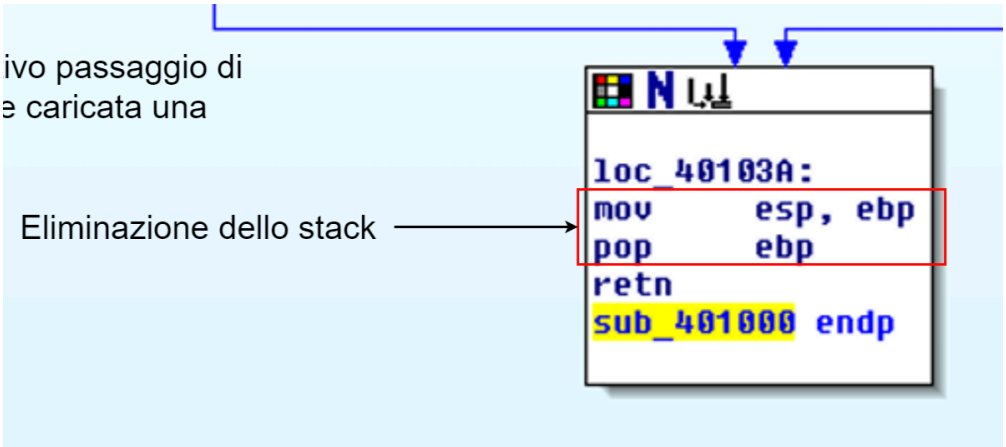


3. Costrutti noti









4. Ipotesi di comportamento

Il codice in esame attraverso la funzione **InternetGetConnectedState** verifica lo stato della connessione.

Nel caso in cui il valore ritornato sia diverso da zero la macchina è connessa ad internet, il programma salta il **jump** e dopo aver caricato una variabile di tipo stringa (La quale conterrà ipoteticamente il messaggio ”Succes, Internet Connection\n”) nello stack, richiama la subroutine (funzione) situata all'indirizzo specificato **40117F.**

Nel caso in cui il valore di ritorno invece sia uguale a zero la macchina non è connessa ad internet, questa volta il programma effettua il salto recandosi all’istruzione nell'indirizzo specificato. Qui verrà caricata un’altra variabile di tipo stringa (La quale conterrà ipoteticamente il messaggio “Error 1.1: No Internet\n") nello stack, e richiamata la subroutine situata all'indirizzo specificato **40117F.**

Possiamo ipotizzare che all’indirizzo 40117F ci sia una funzione che stampa la stringa caricata nello stack.

5. Analisi del codice

1	`push`	`ebp`	Il contenuto del registro ebp viene caricato sullo stack
2	`mov`	`ebp, esp`	Sposto il contenuto di esp in ebp
3	`push`	`ecx`	Il contenuto del registro ecx viene caricato sullo stack
4	`push`	`0 ;dwReserved`	Carico il valore 0 sullo stack *
5	`push`	`0 ;lpdwFlags`	Carico il valore 0 sullo stack **
6	`call`	`ds:InternetGetConnectState`	Viene chiamata la funzione InternetGetConnesctState ***
7	`mov`	`[ebp+var_4], eax`	Sposto il contenuto del registro eax nella variabile locale [ebp+var_4]
8	`cmp`	`[ebp+var_4],0`	Viene comparata la variabile locale con 0
9	`jz`	`short loc_40102B`	Se l'istruzione precedente ha impostato lo ZeroFlag a 1 allora viene effettuato il salto alla locazione indicata, altrimenti continuo l'esecuzione
10	`push`	`offset aSuccessInterne`	Viene caricata la variabile aSuccessInterne di tipo string nello stack
11	`call`	`sub_40117F`	Viene chiamata la funzione all'indirizzo 40117F
12	`add`	`esp,4`	Il contenuto del registro esp viene sommato con 4 ed il risultato memorizzato nel medesimo registro. Così facendo libero lo stack dalla push effettuata a riga 10
13	`mov`	`eax, 1`	Viene salvato il valore 1 all'interno del registro eax
14	`jmp`	`short loc_40103A`	Viene effettuato un salto incondizinato alla locazione di memoria 40103A, per saltare la porzione di codice che viene eseguita nel caso del salto a riga 9

15	loc_40102B		E’ un flag che sta a sottolineare che mi trovo alla locazione 40102B
16	push	offset aError1_1NoInte	Viene caricata la variabile aError1_1NoInte di tipo string nello stack
17	call	sub_40117F	Viene chiamata la funzione all’indirizzo 40117F
18	add	esp, 4	Il contenuto del registro esp viene sommato con 4 ed il risultato memorizzato nel medesimo registro. Così facendo libero lo stack dalla push effettuata a riga 16
19	xor	eax, eax	Viene effettuata l’operazione logica di xor sul contenuto di eax inizializzando a 0
20	loc_40103A		E’ un flag che sta a sottolineare che mi trovo alla locazione 40103A
21	mov	esp, ebp	Viene spostato il contenuto del registro ebp all’interno del registro esp
22	pop	ebp	Rimuove il contenuto della locazione indicizzata a ebp dallo stack
23	retn		Terminare una subroutine o una funzione e ritorna al punto di chiamata
24	sub_401000 endp		Viene utilizzato per delimitare il corpo di una subroutine e segnalarne la fine al compilatore o all'assemblatore ****

* **dwReserved:** Nella funzione “InternetGetConnectedState”, il parametro dwReserved viene riservato per l'uso futuro e non è attualmente utilizzato.

** **lpdwFlags:** Il prefisso "lp" sta per "long pointer", indica che si tratta di un puntatore a un valore DWORD. Nella funzione “InternetGetConnectedState” menzionata in precedenza, il parametro lpdwFlags è utilizzato per restituire lo stato della connessione Internet in particolare la funzione scrive i dati relativi allo stato della connessione all'indirizzo di memoria a cui punta lpdwFlags, consentendo al chiamante di ottenere tali informazioni dopo aver chiamato la funzione.

*** **InternetGetConnectedState:** Questa funzione viene utilizzata per verificare lo stato della connessione Internet su un sistema operativo Windows.Quando la funzione restituisce TRUE, la connessione Internet è attiva, se al contrario restituisce FALSE, non c'è alcuna connessione. Il codice C ti tale funzione è riportato nell'immagine seguente.

```c
#include <windows.h>
#include <wininet.h>
#include <iostream>

int main() {
    // Variabile per memorizzare lo stato della connessione
    DWORD flags;

    // Verifica lo stato della connessione Internet
    if(InternetGetConnectedState(&flags, 0)) {
        // La connessione è attiva
        std::cout << "Connessione Internet attiva." << std::endl;
    } else {
        // Nessuna connessione Internet
        std::cout << "Nessuna connessione Internet attiva." << std::endl;
    }

    return 0;
}
```

**** **endp:** Il comando ENDP è una direttiva utilizzata in assembly per indicare la fine di una procedura o una subroutine. Quando viene incontrata questa direttiva, il compilatore o l'assemblatore comprende che il codice che segue è parte della procedura o subroutine definita precedentemente con la direttiva PROC. Nel caso specifico del codice preso in esame con il comando "**sub_401000 endp**" viene indicata la fine della funzione iniziata all'indirizzo 401000

```
MyProcedure PROC
    ; Corpo della procedura
    ; ...
    RET
MyProcedure ENDP
```