

PROGETTO S10, L5

MATTIA PASTORELLI





CONSEGNA

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

1) LIBRERIE IMPORTATE

Kernel32.dll:

È una delle librerie principali di Windows che offre funzioni di basso livello per il controllo delle risorse di sistema come la gestione della memoria, dei file e dei processi. Include numerose funzioni di base utilizzate da altri programmi e librerie.

Wininet.dll:

È una libreria di Windows che supporta le operazioni di rete e Internet. Contiene funzioni per la connessione ai server Internet, il download di risorse da Internet (come pagine Web e file), l'invio di richieste HTTP e l'elaborazione delle risposte.

In breve, Kernel32.dll costituisce il nucleo del sistema operativo Windows, mentre Wininet.dll fornisce funzionalità per l'interazione con Internet e le operazioni di rete.

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

2) SEZIONI DEL FILE

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristi
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

Il file del malware è composto dalle seguenti sezioni:

Sezione .txt:

Questa sezione, nota anche come sezione del testo, all'interno di un file eseguibile, di solito contiene dati di testo non modificabili o istruzioni del programma. Qui sono presenti stringhe di testo utilizzate dal programma, come messaggi di errore, costanti o altre informazioni di testo.

Sezione .rdata:

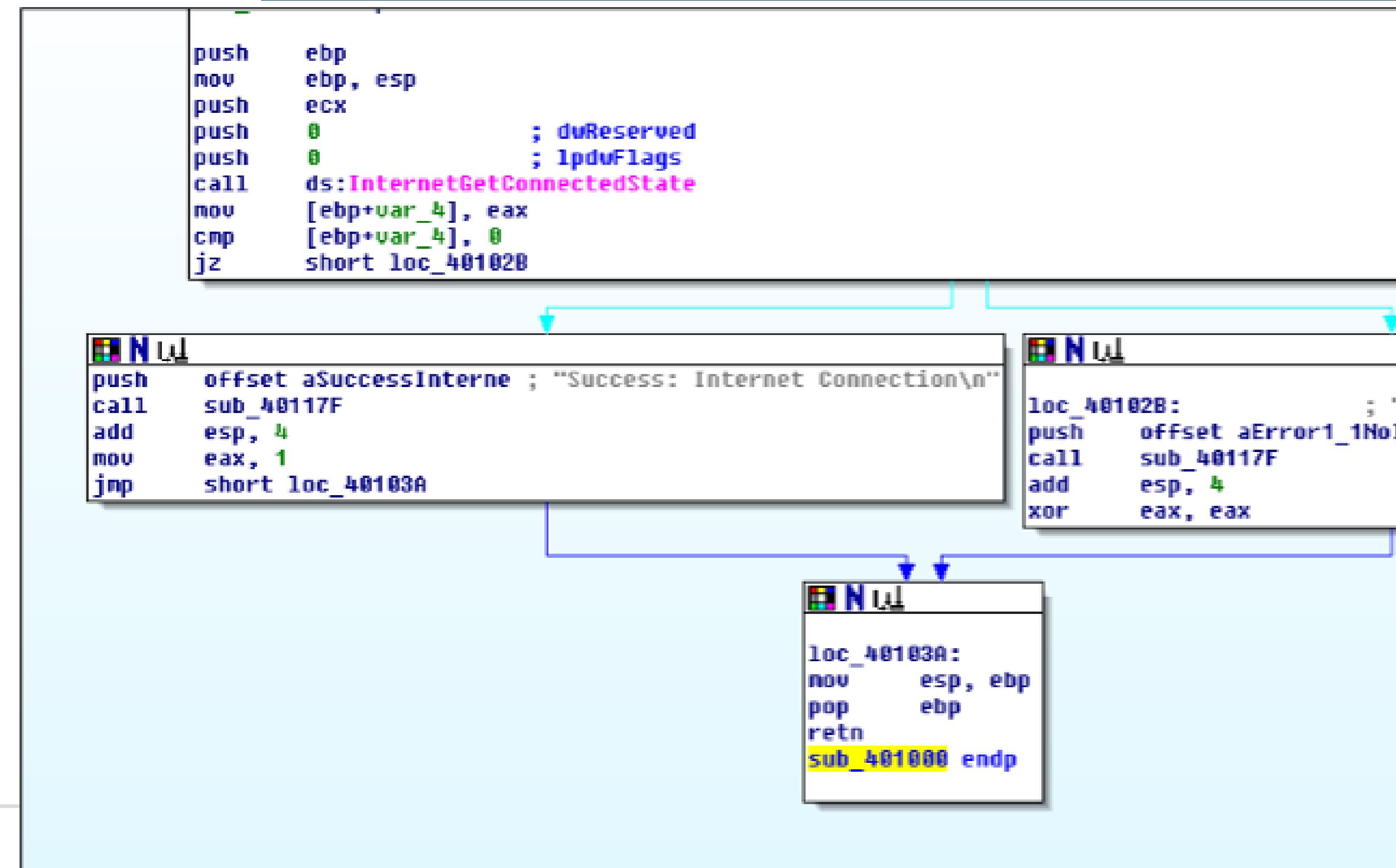
La sezione .rdata, chiamata anche sezione dati solo per la lettura, contiene dati di sola lettura, come costanti o variabili globali inizializzate. Questi dati sono accessibili in sola lettura durante l'esecuzione del programma. Di solito, questa sezione contiene informazioni statiche utilizzate dal programma, come tabelle di lookup o valori costanti.

Sezione .data:

La sezione .data contiene dati inizializzati che possono essere modificati durante l'esecuzione del programma. Qui sono presenti variabili globali o dati dinamici mentre il programma è in esecuzione. I dati in questa sezione possono essere letti e modificati durante l'esecuzione del programma.

3) COSTRUTTI NOTI

- **Push:** I primi push creano lo stack, spingendo i valori all'interno di esso.
- **Cmp e Jz :** Il primo mette a comparazione il valore 0 con il valore presente nel registro (EBP + VAR_4). Il JZ salta direttamente all'istruzione loc_40102B. Queste due istruzioni insieme possono essere ricollegate ad un ciclo If/Else in C.
- **JMP :** Salta direttamente all'istruzione segnata, paragonabile al GOTO del linguaggio C
- **Retn:** Paragonabile al Return 0 in C, esegue il ritorno alla funzione, con destinazione l'indirizzo nell'istruzione Call



4) FUNZIONAMENTO

Sembra che il codice sia progettato per controllare lo stato della connessione Internet utilizzando "InternetGetConnectedState", in caso di esito positivo, verrà stampato un messaggio di successo. In caso contrario verrà generato un messaggio d'errore a schermo.

5) BONUS SPIEGAZIONE CODICE

Istruzione	Spiegazione
push ebp	Pone il valore nel registro EBP nello stack
mov ebp, esp	Copia il valore ESP in EBP
push ecx	Mette il valore in ECX nello stack
push 0 : dwReserved	Mette il valore 0 nello stack e lo passa come dwReserved per la funzione
push 0 : lpdwFlags	Mette il valore 0 nello stack e lo passa come lpdwFlags per la funzione
call ds: InternetGetConnectedState	Chiama la funzione InternetGetConnectedState per valutare se la connessione è attiva
mov [ebp+var_4], eax	Copia il risultato dato dalla funzione precedente (EAX) in EBP + VAR_4
cmp [ebp+var_4], 0	Pone a confronto il valore 0 con il risultato della funzione
jz short loc_40102B	Se è uguale a 0 salta direttamente all'etichetta loc_40102B se non avviene connessione
push offset aSuccessInterne	Immette la stringa nello stack (Avvenuta connessione)
call sub_40117F	Chiama una subroutine sub_40105F per stampare il risultato di avvenuta connessione
add esp, 4	Ripristina lo stack dopo la chiamata alla subroutine
mov eax, 1	Copia il valore 1 nel registro EAX, indicando che la connessione è avvenuta con successo
jmp shor loc_40103A	Salta direttamente all'etichetta loc_40103A
loc_40102B:	Indirizzo di memoria
push offset aError1_1Nolnte	Mette sullo stack l'offset , ovvero l'indirizzo della stringa.
call sub_40117F	Questa istruzione chiama la funzione sub_40117F. La chiamata di funzione call mette l'indirizzo di ritorno nello stack e salta all'indirizzo specificato.
add esp,4	Ripristina lo stack dopo la chiamata alla subroutine
xor eax,eax	Effettua un'operazione XOR tra il registro eax e se stesso, ponendo il risultato a zero. Questo può essere usato per cancellare il registro eax,
loc_40103A:	Indirizzo di memoria
mov esp,ebp	Copia il valore di EBP nel registro stack ESP
pop ebp	Questa istruzione ripristina il registro di base della pila (ebp) dallo stack
retn	Questa istruzione esegue il ritorno dalla funzione, utilizzando l'indirizzo di ritorno messo sullo stack durante la chiamata di funzione call.
sub_401000 endp	Indica che la subroutine che inizia all'indirizzo di memoria 0x401000 termina qui.