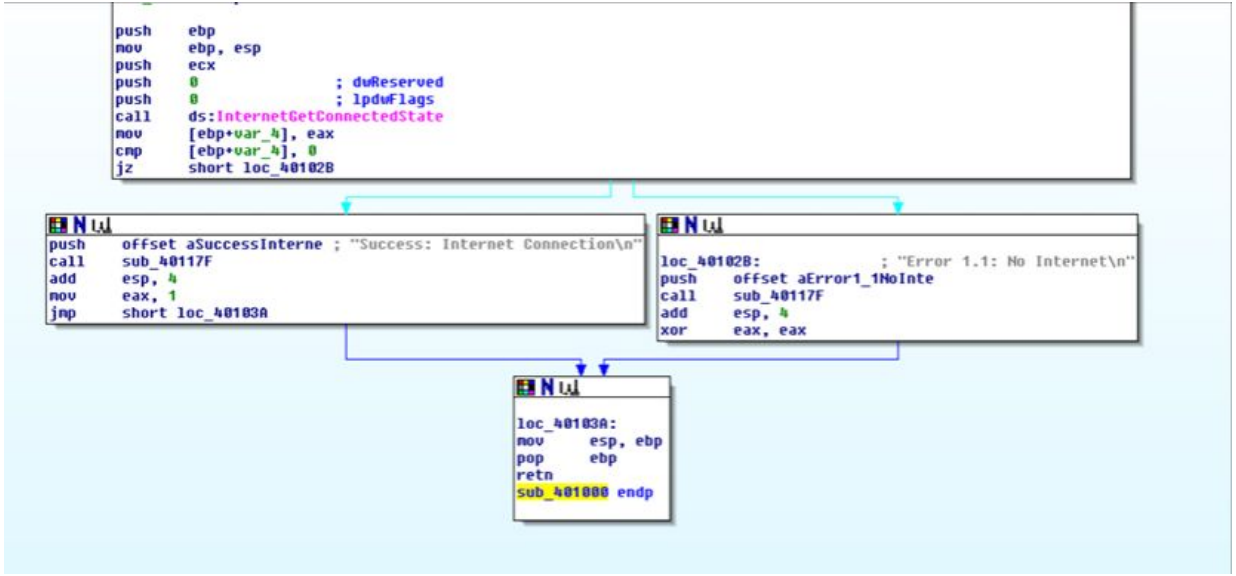
Progetto S10/L5

Analisi statica di un malware

Traccia

Nel progetto di oggi ci viene chiesto di identificare le librerie e le sezioni di un malware. Inoltre ci viene chiesto di identificare i costrutti noti e la funzionalità del codice assembly in figura.

Per l’analisi delle librerie e sezioni del malware useremo CFF Explorer.



Librerie di un malware

Come possiamo vedere dalla figura sotto sono presenti due librerie: Kernel32 e Wininet. Esse sono librerie importate dinamicamente (dynamic link library), ovvero caricate dal sistema operativo quando l'eseguibile viene avviato.

Kernel32 contiene tutte quelle funzioni che interagiscono con il sistema operativo, come ad esempio la gestione della memoria, gestione del disco e i flussi di input e output.

Wininet invece contiene le funzioni per l'implementazione dei protocolli di rete, come ad esempio HTTP e FTP. Può essere usata per cercare una connessione a internet, connettersi ad altri domini e scaricare altri file.

KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

Sezioni di un malware

Con riferimento alla figura sotto possiamo identificare tre sezioni:

- .text contiene le istruzioni che la CPU eseguirà una volta che il malware sarà avviato. Questa è l'unica sezione che contiene codice eseguibile;
- .rdata contiene dati di sola lettura riguardanti librerie e funzioni importate ed esportate;
- .data contiene i dati/variabili globali inizializzati del programma.

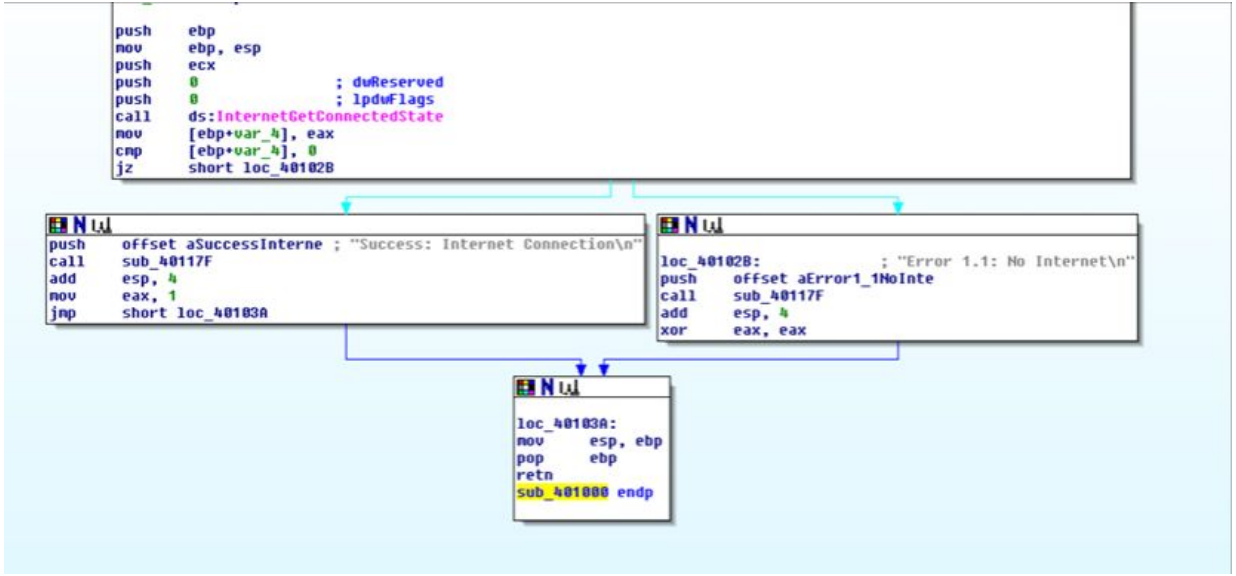
Come abbiamo visto le librerie e le sezioni possono fornirci informazioni preziose sulla struttura di un malware e sul suo comportamento.

.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000

Costrutti noti

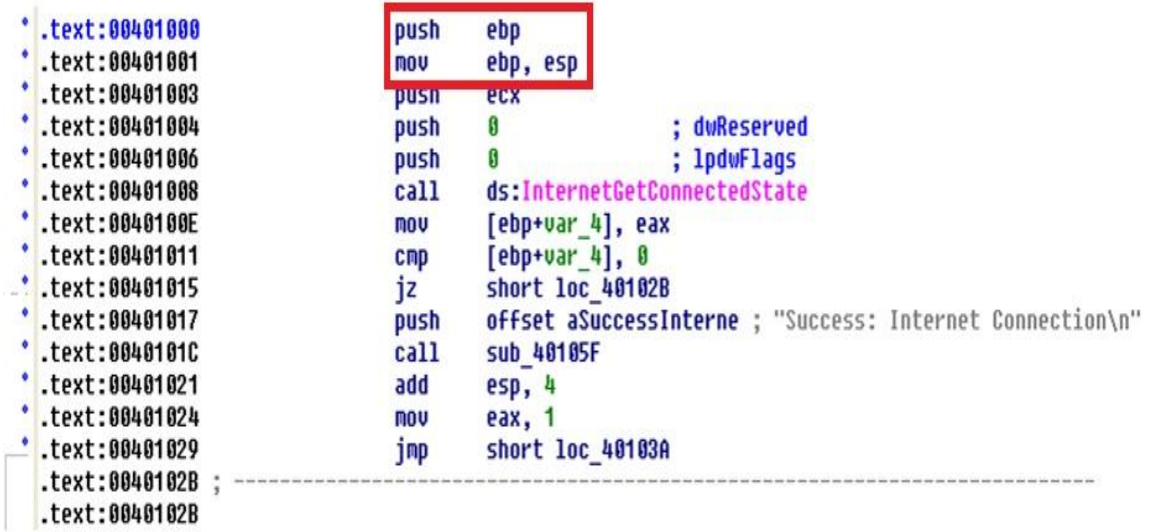
Nell’immagine a fianco ci viene mostrata una porzione di codice assembly di un malware. Da questa ci viene chiesto di identificare i costrutti noti e provare a ipotizzarne la funzione.

Possiamo identificare quattro costrutti: un ciclo if, il push dei parametri della funzione e la creazione e rimozione dello stack.



Creazione dello stack

Nell’immagine viene evidenziata la parte di codice in cui viene creato lo stack della funzione.



Push dei parametri

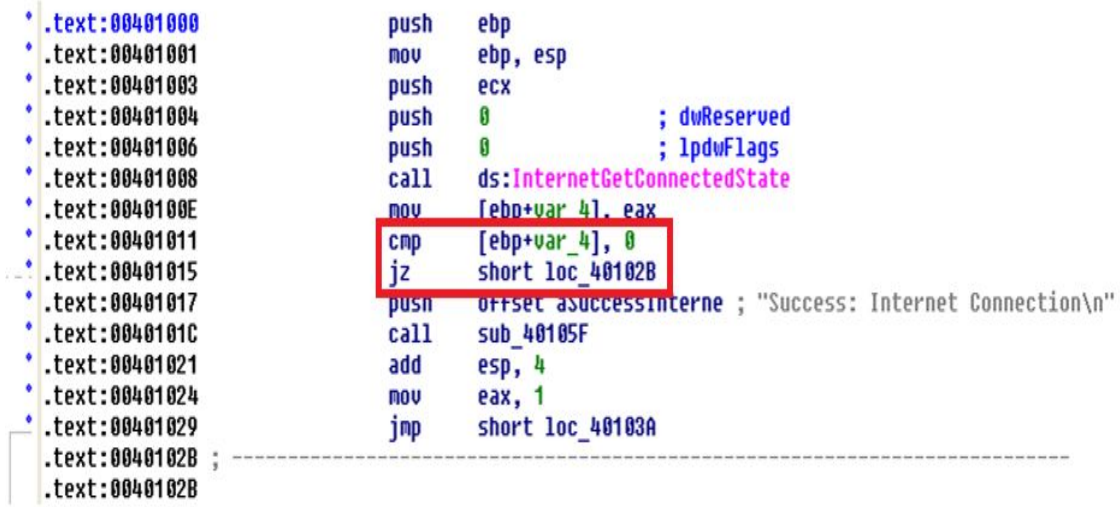
Qui possiamo vedere la porzione di codice in cui vengono passati i parametri alla funzione tramite il comando “push”.

```
.text:00401000                push    ebp
.text:00401001                mov     ebp, esp
.text:00401003                push    ecx
.text:00401004                push    0               ; dwReserved
.text:00401006                push    0               ; lpdwFlags
.text:00401008                call    ds:InternetGetConnectedState
.text:0040100E                mov     [ebp+var_4], eax
.text:00401011                cmp     [ebp+var_4], 0
.text:00401015                jz      short loc_40102B
.text:00401017                push    offset aSuccessInterne ; "Success: Internet Connection\n"
.text:0040101C                call    sub_40105F
.text:00401021                add     esp, 4
.text:00401024                mov     eax, 1
.text:00401029                jmp     short loc_40103A
.text:0040102B ; ---------------------------------------------------------------------------
.text:0040102B
```

Ciclo if

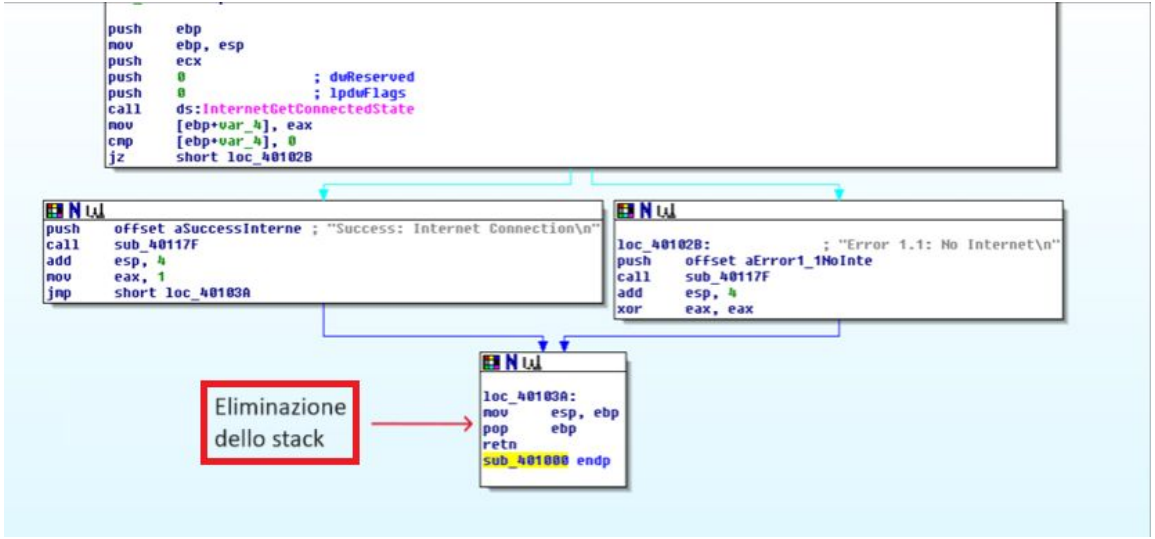
Nell'immagine viene evidenziato il ciclo if, composto da due istruzioni:

- "cmp" (compare) che compara due operandi effettuando una sottrazione tra di essi. In base al risultato cambia gli status flag;
- "jz" effettua un salto a una locazione specifica se lo zero flag è 1.



Rimozione dello stack

Una volta che una funzione esaurisce il suo compito lo stack di memoria creato deve essere eliminato non essendo più necessario. Nell'ultimo riquadro in basso viene mostrata questa pratica detta pulizia dello stack.



Funzionalità del codice

Osservando bene le parti di codice indicate dalle frecce in figura possiamo provare a ipotizzare il funzionamento del codice.

Il malware cerca una connessione attiva tramite la funzione “InternetGetConnectedState”. Se lo zero flag sarà 1 stamperà a schermo “Error”, altrimenti stamperà “Success”.