Epicode Unit 3 Week 2

Assembly language malware

Nell'esercizio odierno si analizza il codice di un probabile malware per cercare di risalire alla sua funzione. Si procede dapprima con l'analisi riga per riga, che semplificherà notevolmente il lavoro:

```
push    ebp
mov     ebp, esp
```

- Nella prima riga si "pusha" il valore contenuto nell'Extended Base Pointer sulla stack;
- Nella seconda, il valore dello stack pointer viene copiato nell'ebp.

Si può affermare che queste due righe siano responsabili della creazione della stack per l'esecuzione delle funzioni successive.

```
push    ecx
push    0                   ; dwReserved
push    0                   ; lpdwFlags
```

- Il registro generico EXC viene pushato in cima alla stack;
- Le due righe di seguito pushano due valori (in questo caso 0 in entrambi i casi) in cima alla stack. Tali valori verranno passati come argomenti alla funzione che verrà chiamata alla riga successiva

```
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

-

- Come già anticipato, nella riga numero sei viene chiamata la funzione "InternetGetConnectedState" per definire lo stato della connessione internet della macchina. Il prefisso "ds" indica che la chiamata fa riferimento ad una specifica sezione di memoria;
- Alla riga sette viene mosso il valore contenuto nel registro EAX (verosimilmente contenente il valore di risposta della funzione

precedentemente chiamata), nel registro [ebp+var_4], che conserverà tale valore come variabile locale.

- La riga otto eseguirà una comparazione tra il valore del registro e 0 e imposterà le flag di conseguenza;
- La riga nove esegue un "conditional jump", che porterà all'indirizzo di memoria indicato qualora il valore della flag risulti essere 0

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40105F
```

- Le righe dieci e undici servono per pushare una stringa sulla stack, per poi chiamare la funzione relativa all'indirizzo di memoria specificato che verosimilmente lavorerà con la stringa di cui sopra (quasi certamente si tratta di una funzione "print");

```
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

- La riga 12 va ad aggiungere 4 al valore dell'ESP, potenzialmente per "resettare" la stack;
- Questa riga, invece, copierà il valore 1 nell'EAX, probabilmente per restituire tale valore come quello della funzione;
- In ultimo, il codice esegue un "unconditional jump", cioè un salto che si verificherà sempre, alla porzione di memoria specificata.

Alla luce di quanto analizzato, possiamo supporre soltanto che un eventuale malware utilizzi questa sezione di codice per verificare la connessione con Internet. Possiamo dunque ipotizzare che tale programma comunichi con l'esterno per eventuali esecuzioni di comandi da remoto o per l'invio di dati dalla macchina, ma senza ulteriori informazioni ci è impossibile affermarlo con certezza.

Nella pagina successiva viene anche riportato un probabile Codice C di Origine dell'Assembly Analizzato:

```
#include <stdio.h>
#include <wininet.h>
#include <windows.h>

int main() {
   BOOL Connessione = 0;

   // Associo la mia varabile alla funzione necessaria
   Connessione = InternetGetConnectedState(0, 0);

   if (Connessione == TRUE) {
      printf("Success: Internet Connection Established\n");
   } else {
      printf("Failure: No Internet Connection\n");
   }

   return 0;
}
```