REPORT 12/01/2023 **LORENZO SCORBAIOLI**

TASK:

Dato il seguente estratto di codice malware:

```
push    ebp
mov    ebp, esp
push    ecx
push    0          ; dwReserved
push    0          ;lpdwFlags
call    ds:InternetGetConnectedState
mov    [ebp+var_4], eax
cmp    [ebp+var_4], 0
jz    short loc_401102B
push    offset aSuccessInterne ;"Success: Internet Connection\n"
call    sub_40105F
add    esp,    4
mov    eax, 1
jmp    short loc_40103A
```

1. Identificare i costrutti noti
2. Ipotizzare la funzionalità
3. Bonus: studiare e spiegare ogni singola riga di codice

1)

Tra i costrutti noti che ho individuato in queste righe di codice c'è un ciclo if, che ho identificato nelle seguenti righe di codice:

cmp   [ebp+var_4], 0
jz   short loc_401102B

Con queste 2 righe di codice il programma va a fare la differenza tra la variabile scritta nel registro "ebp+var-4" e 0. Se il risultato di questa operazione dà come risultato 0 la zero flag assume valore 1. In questo caso la condizione del jz (jump zero) viene confermata e quindi viene fatto un salto breve alla locazione di memoria con indirizzo 401102B. Se invece dà un risultato diverso da 0, la zero flag assume valore 0 e quindi non viene attuato il jump e continua a eseguire le righe di codice immediatamente successive fino ad arrivare allo short jump verso l'indirizzo di memoria 40103A.

Tra le altre cose rilevanti che ho notato c'è la creazione dello stack, che ho identificato nelle prime 2 righe di codice:

push   ebp
mov   ebp, esp

tuttavia non viene definito immediatamente lo spazio dedicato alle variabili locali.

Altra parte interessante è rappresentata dalle seguenti righe di codice:

```
push    ecx
push    0          ; dwReserved
push    0          ;lpdwFlags
call    ds:InternetGetConnectedState
```

con le prime 3 operazioni push vengono inserite in cima allostack 3 parametri, che sono “ecx”, “0” e “0”. Questi 3 parametri vengono dati in pasto alla funzione, chiamata tramite l’istruzione call, “InternetGetConnectedState”.

2)

Andando ad ipotizzare la funzionalità di questo estratto di codice, mi verrebbe da dire che questa parte serve a controllare se ci sia una connessione internet attiva o meno sulla macchina che viene colpita dal malware.

3)

Andando ad analizzare il codice riga per riga abbiamo:

-push   ebp

Con questa istruzione viene “pushato” l’extended base pointer sulla cima dello stack

-mov   ebp, esp

Qua invece viene assegnato il valore del registro dell’extended stack pointer al registro dell’extended base pointer

-push   ecx

Qua, tramite l’istruzione push, viene posto il valore inserito nel registro “ecx” in cima allo stack

-push   0   ; dwReserved

Crea un buffer vuoto di 4 byte sullo stack. Dal commento e tramite una ricerca su internet ho visto che questo parametro, che verrà utilizzato dalla successiva funzione chiamata, è riservato e deve essere 0

-push   0   ;lpdwFlags

Come sopra, questa istruzione crea un buffer vuoto di 4 byte. Dal commento accanto ho visto che è relativo a un puuntatore che riceve la descrizione della connessione tramite la funzione chiamata dalla riga di comando immediatamente successiva.

-call   ds:InternetGetConnectedState

Viene chiamta la funzione “InternetConnectedState”.

-mov [ebp+var_4], eax

Viene copiato il valore contenuto nel registro “eax” nel registro “ebp+var_4”, cioè nel registro a distanza di 4 byte verso la cima dello stack

-cmp [ebp+var_4], 0

Con questa istruzione viene attuata una sottrazione tra il parametro contenuto nel registro “ebo+4_var” e 0 andando a modificare la zeta flag del registro. Questo valore sarà fondamentale per la successiva istruzione di jump

-jz short loc_401102B

Con questa istruzione di jump viene controllata la zeta flag ottenuta dalla precedente istruzione “cmp”. Se questa risulterà uguale a 1 verrà effettuato uno “short jump” all’indirizzo di memoria “401102B2”, altrimenti verranno eseguite normalmente le successive righe di codice

-push offset aSuccessInterne ;”Success: Internet Connection\n”

Con questa istruzione viene inserita la stringa “aSuccessInterne” in un registro in cima allo stack

-call sub_40105F

Viene chiamata una funzione inserita nell’indirizzo di memoria “40105F”

-add esp, 4

Con questa istruzione viene effettuata una somma tra il valore inserito all’interno del registro “esp” e 4

-mov eax, 1

Viene sostituito il valore contenuto all’interno del registro “eax” con il valore 1

-jmp short loc_40103A

Viene effettuato un salto all’indirizzo di memoria “40103A”

http://masm32.com/board/index.php?topic=5060.0
https://learn.microsoft.com/it-it/windows/win32/api/wininet/nf-wininet-internetgetconnectedstate
https://www.syncfusion.com/succinctly-free-ebooks/assemblylanguage/data-segment