



ANALISI STATICA E DINAMICA DI UN MALWARE

Un approccio pratico



Obbiettivi

Parte I

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?

Parte II

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

Librerie importate

CFF explorer – parte I

Per poter eseguire una analisi statica di base sul malware ci avvarremo di un software chiamato CFFExplorer.

Si tratta di un software avanzato che ci permette di analizzare file eseguibili e, utile al nostro scopo, individuare eventuali librerie importate.

Cosa sono le librerie?

Una libreria e' un insieme di dati e funzioni predefinite presenti nel sistema, che possono essere richiamate da altri programmi attraverso specifiche chiamate. Le librerie hanno funzioni diverse e a seconda del loro uso possono essere sfruttate anche da malware per eseguire un attacco al sistema.



Kernel32.dll

La libreria Kernel32 è una parte fondamentale del sistema operativo Windows. Essa fornisce funzionalità di basso livello per la gestione delle risorse di sistema, come la gestione della memoria, la gestione dei file, la gestione dei processi e la comunicazione tra processi.



Wininet.dll

La libreria WinINet è una libreria di Microsoft Windows che fornisce funzionalità per la gestione delle operazioni di rete, come la connessione e il trasferimento di dati attraverso protocolli come HTTP, FTP.

Sezioni del malware

CFF explorer – parte II

Sempre tramite CFFExplorer ci e' possibile controllare in quante sezioni viene suddiviso il malware. Per poter fare cio', in certi casi occorre prima decomprimere i file binari tramite la utility UPX unpacker.

Una volta decompresso ci sara' possibile verificare le sezioni in cui e' suddiviso il file eseguibile.

Cos'è UPX?

UPX è uno strumento di compressione per eseguibili che consente di ridurre le dimensioni dei file binari senza comprometterne la funzionalità. Quando un file eseguibile è stato compresso con UPX, può essere decompresso dinamicamente durante l'esecuzione per ripristinare la sua forma originale.

.data

Questa sezione contiene dati globali e variabili inizializzate utilizzate dal programma durante l'esecuzione. Questi dati possono includere variabili globali, costanti e altri dati che vengono inizializzati prima che il programma venga eseguito.

.rdata

Questa sezione contiene dati di sola lettura che vengono utilizzati dal programma durante l'esecuzione, ma che non vengono modificati. Ad esempio, qui potrebbero essere presenti costanti o altre informazioni di sola lettura utilizzate dal programma.

.text

Contiene il codice eseguibile del programma, ovvero le istruzioni macchina che vengono eseguite dal processore. Questa sezione è di solito di sola lettura e contiene le istruzioni del programma in linguaggio macchina.

Analisi di un codice assembly

```
push    ebp
mov     ebp, esp
push    ecx
push    0          ; duReserved
push    0          ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_401028
```

CREAZIONE DELLO STACK

CHIAMATA ALLA FUNZIONE "InternetGetConnectedState"

COSTRUTTO IF

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

COSTRUTTO GO-TO

```
loc_401028:          ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

RIMOZIONE DELLO STACK

Comportamento del codice

Il codice mostrato accanto implementa un controllo sullo stato della connessione a internet.

Dopo aver creato la memoria nello stack, richiama la funzione

"InternetGetConnectedState" e salva il risultato della funzione chiamata nel registro `ebp+var_4`.

Procede quindi ad effettuare un paragone tra il suddetto valore e 0 (per confermare eventualmente la connessione a internet).

Se il risultato dovesse essere diverso da 0 (quindi connessione avvenuta), procede con la stampa di "Success: Internet Connection".

In caso contrario procede alla stampa di "Error 1.1: No Internet" per poi terminare.

BONUS – analisi avanzata di un codice assembly

```
push ebp                                //Salva il valore del puntatore di base dello stack (EBP) nella memoria stack
mov ebp,esp                             //Crea un nuovo frame dello stack impostando EBP uguale al puntatore stack corrente (ESP)
push ecx                                //Salva il contenuto del registro ECX nello stack
push 0 ;dwReserved                       //Mette 0 nello stack riferito al parametro dwReserved
push 0 ;lpwdFlags                        //Mette 0 nello stack riferito al parametro lpwdFlags
call ds:InternetGetConnectedState        //Richiama la funzione InternetGetConnectedState
mov [ebp+var_4], eax                     //Salva il valore di ritorno della funzione precedente nella variabile locale ebp+var_4
cmp [ebp+var_4], 0                       //Compara il valore salvato nella variabile locale con 0
jz short loc_40102B                     //Salta a loc_40102B se il risultato della comparazione è zero

push offset aSuccessInterne ;"Success: Internet Connection" //Mette l'indirizzo della stringa nello stack
call sub_40117F                          //Chiama una sottofunzione a sub_40117F
add esp, 4                               //Rimuove il parametro dallo stack
mov eax, 1                               //Imposta EAX a 1
jmp short loc_40103A                     //Salta a loc_40103A

loc_40102B: - Posizione se non c'è connessione Internet.
push offset aError1_NoInte              //Mette l'indirizzo della stringa nello stack
call sub_40117F                          //Chiama una sottofunzione a sub_40117F
add esp, 4                               //Rimuove il parametro dallo stack
xor eax, eax                             //Esegue uno xor logico tra EAX e EAXquindi imposta EAX a zero

loc_40103A: - Posizione dopo il controllo di connessione Internet
mov esp, ebp                             //Ripristina ESP al valore del puntatore di EBP ripristinando lo stack allo stato precedente
pop ebp                                  //Rimuove EBP dalla memoria stack
retn                                     //Restituisce il controllo alla chiamata della funzione
sub_401000 endp                          //Indica la terminazione di una sottofunzione sub_401000
```