

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:

2. Quali librerie vengono importate dal file eseguibile?
4. Quali sono le sezioni di cui si compone il file eseguibile del malware?

Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

2. Identificare i costrutti noti (creazione dello stack, eventuali cicli,altri costrutti)
3. Ipotizzare il comportamento della funzionalità implementata
5. **BONUS** fare tabella con significato delle singole righe di codice assembly

```
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_40102B
```

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

```
loc_40102B:          ; "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
```

Librerie importate

Analizzando la sezione [Import Directory](#) di CFF Explorer, è possibile osservare che il malware **U3 W2 L5** importa due librerie:

- **KERNEL32.dll**: La libreria Kernel32.dll in Windows svolge un ruolo fondamentale fornendo funzioni essenziali per il sistema operativo, come gestione della memoria, manipolazione dei file, gestione dei processi e thread, comunicazione tra processi e altro ancora. Le applicazioni Windows utilizzano queste funzioni per interagire con il sistema operativo.
- **WININET.dll**: è una libreria di sistema in Windows che gestisce operazioni di rete e connettività Internet. Fornisce funzioni per richieste HTTP/FTP, gestione dei cookie, cache Internet, autenticazione e configurazione del proxy. Essenziale per applicazioni Windows che richiedono accesso a risorse online.

File: Malware_U3_W2_L5.exe

Dos Header

Nt Headers

File Header

Optional Header

Data Directories [x]

Section Headers [x]

Import Directory

Address Converter

Dependency Walker

Hex Editor

Identifier

Import Adder

Quick Disassembler

Rebuilder

Resource Editor

UPX Utility

Malware_U3_W2_L5.exe

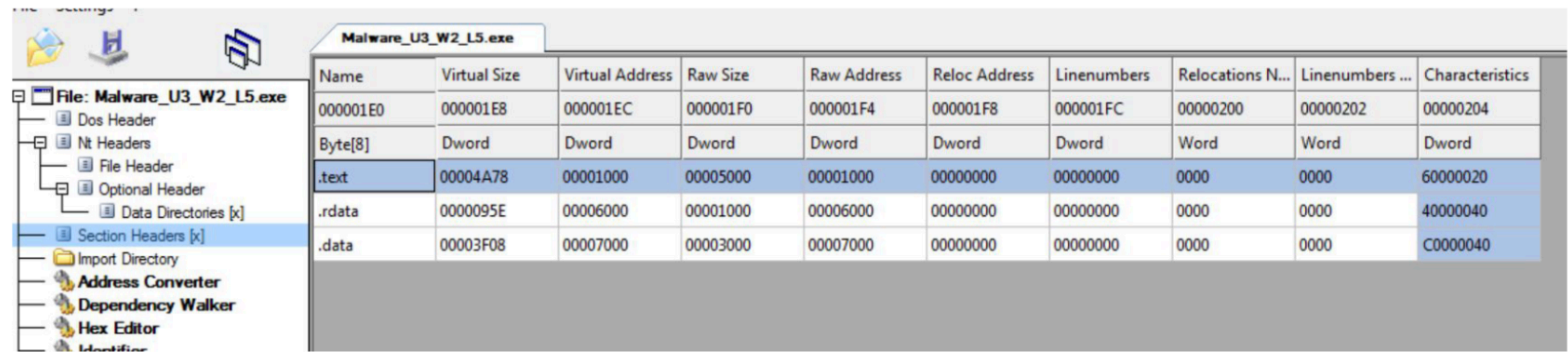
| Module Name | Imports | OFTs | TimeStamp | ForwarderChain | Name RVA | FTs (IAT) |
|--------------|--------------|----------|-----------|----------------|----------|-----------|
| 00006664 | N/A | 000064F0 | 000064F4 | 000064F8 | 000064FC | 00006500 |
| szAnsi | (nFunctions) | Dword | Dword | Dword | Dword | Dword |
| KERNEL32.dll | 44 | 00006518 | 00000000 | 00000000 | 000065EC | 00006000 |
| WININET.dll | 5 | 000065CC | 00000000 | 00000000 | 00006664 | 000060B4 |

| OFTs | FTs (IAT) | Hint | Name |
|----------|-----------|------|---------------------------|
| | | | |
| Dword | Dword | Word | szAnsi |
| 00006640 | 00006640 | 0071 | InternetOpenUrlA |
| 0000662A | 0000662A | 0056 | InternetCloseHandle |
| 00006616 | 00006616 | 0077 | InternetReadFile |
| 000065FA | 000065FA | 0066 | InternetGetConnectedState |
| 00006654 | 00006654 | 006F | InternetOpenA |

Section Headers

Analizzando la sezione Section Headers di CFF Explorer, è possibile osservare che il malware ***U3 W2 L5*** si compone di 3 sezioni:

- .text**: contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.
- .rdata**: include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, informazione che come abbiamo visto possiamo ricavare con CFF Explorer.
- .data**: contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Una variabile si dice globale quando non è definita all'interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione all'interno dell'eseguibile.



| Malware_U3_W2_L5.exe | | | | | | | | | |
|----------------------|--------------|-----------------|----------|-------------|---------------|-------------|------------------|-----------------|-----------------|
| Name | Virtual Size | Virtual Address | Raw Size | Raw Address | Reloc Address | Linenumbers | Relocations N... | Linenumbers ... | Characteristics |
| 000001E0 | 000001E8 | 000001EC | 000001F0 | 000001F4 | 000001F8 | 000001FC | 00000200 | 00000202 | 00000204 |
| 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 |

Identificare i costrutti noti

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

Creazione dello STACK

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

Chiamata di funzione. I parametri sono passati sullo stack tramite le istruzioni push

```
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

ciclo IF

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

```
loc_40102B:
push    offset aError1_1NoInte ; "Error 1.1: No Internet\n"
call    sub_40117F
add     esp, 4
xor     eax, eax
```

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

RIMOZIONE DELLO STACK

Comportamento

Il codice verifica lo stato della connessione a Internet utilizzando la funzione *InternetGetConnectedState*. Se la connessione è attiva, viene stampato un messaggio di successo, altrimenti viene stampato un messaggio di errore.

Pseudo codice C:

```
// Chiamata alla funzione InternetGetConnectedState
state = InternetGetConnectedState(par1, 0, 0);

// Verifica se lo stato è uguale a 0 (nessuna connessione)
if (state == 0) {
    printf("Error 1.1: No Internet\n");
} else {
    // Connessione attiva, stampa il messaggio di successo
    printf("Success: Internet Connection\n");
}

return 0;|
```

Analisi passo-a-passo:

Righe 1-3:

- push ebp: Salva il puntatore di base (EBP) corrente nello stack. Questo stabilisce un nuovo frame stack per la funzione, consentendo l'accesso alle variabili locali usando offset relativi a EBP.
- mov ebp, esp: Imposta il puntatore di base (EBP) sul puntatore stack corrente (ESP). Questo crea un punto di riferimento fisso per accedere alle variabili locali e agli argomenti all'interno del frame stack della funzione.
- push ecx: Salva il valore nel registro ECX nello stack. Questo è probabile perché ECX verrà utilizzato in seguito nella funzione e salvarlo previene potenziali conflitti con altre parti del codice.

Righe 4-5:

- push 0: Pushes due valori zero sullo stack. Questi zeri servono come argomenti segnaposto per la funzione InternetGetConnectedState, che potrebbe aspettarsi slot di argomenti specifici anche se alcuni valori non sono utilizzati.

Riga 6:

- call ds:InternetGetConnectedState: Chiama la funzione InternetGetConnectedState, presumibilmente dall'area di archiviazione dinamica (DS). Questa funzione controlla lo stato della connessione Internet e restituisce un valore che indica lo stato della connessione.

Riga 7:

- mov [ebp+var_4], eax: Memorizza il valore di ritorno da InternetGetConnectedState (che si trova nel registro EAX) in una variabile locale con un offset di 4 dal puntatore di base (EBP). Questa variabile, probabilmente chiamata var_4, viene utilizzata per controllare lo stato della connessione in seguito.

Righe 8-10:

- cmp [ebp+var_4], 0: Confronta il valore in var_4 (lo stato della connessione) con 0.
- jz short loc_40102B: Se il confronto è uguale a zero (connessione riuscita), salta all'etichetta loc_40102B, dove si verifica la gestione degli errori.

Righe 11-14:

- push offset aSuccessInterne: Pushes l'indirizzo di una stringa letterale "Successo: Connessione Internet\n" sullo stack, preparandola per la stampa.
- call sub_40117F: Chiama una funzione sub_40117F, probabilmente responsabile della stampa della stringa sullo stack (indirizzo memorizzato in EAX).
- add esp, 4: Pulisce lo stack rimuovendo l'indirizzo della stringa.
- mov eax, 1: Imposta il registro EAX su 1, presumibilmente indicando un'esecuzione corretta.
- jmp short loc_40103A: Salta all'etichetta loc_40103A, la fine della funzione.

Righe 15-19:

- loc_40102B: L'etichetta di gestione degli errori raggiunta se lo stato della connessione non è riuscito.
- push offset aError1_1NoInte: Pushes l'indirizzo di una stringa letterale "Errore 1.1: Nessuna Internet\n" sullo stack.
- call sub_40117F: Chiama sub_40117F di nuovo, probabilmente per stampare il messaggio di errore.
- add esp, 4: Pulisce lo stack.
- xor eax, eax: Imposta EAX su zero, indicando un errore.
- loc_40103A: La fine della funzione, raggiunta sia dal successo che dalla gestione degli errori.

Righe 20-22:

- mov esp, ebp: Ripristina il puntatore stack (ESP) al puntatore di base (EBP), rimuovendo in modo efficace il frame stack della funzione.
- pop ebp: Ripristina il puntatore di base precedente (EBP) dallo stack, tornando al frame del chiamante.
- retn: Ritorna dalla funzione.