Esercitazione W23D1 - Pratica 1

Windows Malware

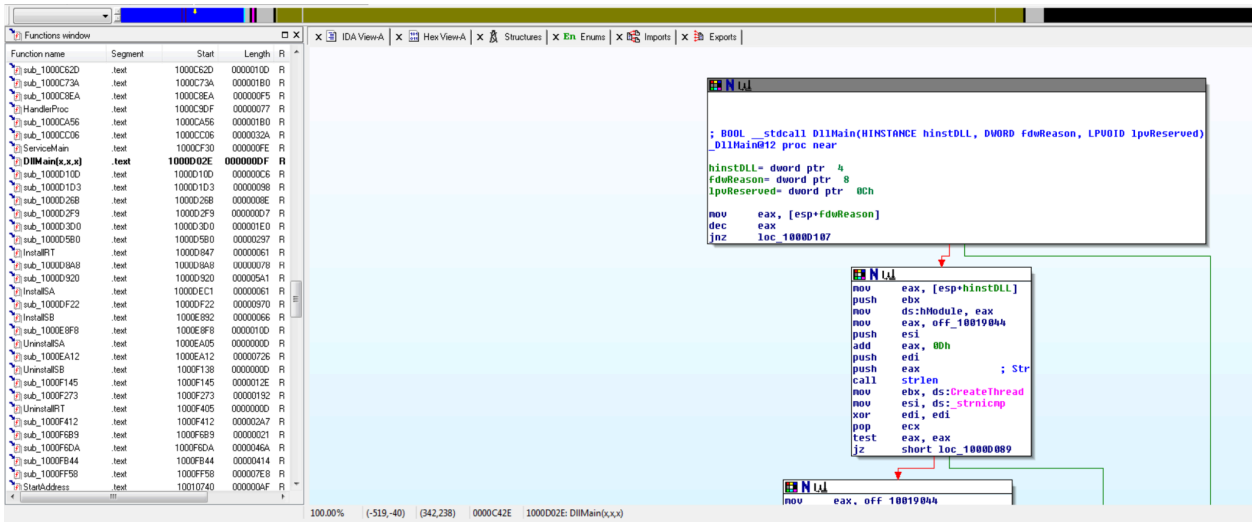
Fabio Benevento - 09/04/2024

Traccia

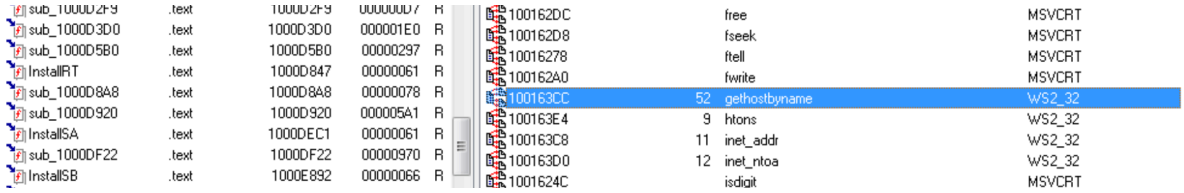
Lo scopo dell'esercizio di oggi è di acquisire esperienza con IDA, un tool fondamentale per l'analisi statica.
A tal proposito, con riferimento al malware chiamato «`Malware_U3_W3_L2`» presente all'interno della cartella «`Esercizio_Pratico_U3_W3_L2`» sul Desktop della macchina virtuale dedicata all'analisi dei malware, rispondere ai seguenti quesiti, utilizzando IDA Pro.
1. Individuare l'indirizzo della funzione DLLMain (così com'è, in esadecimale)
2. Dalla scheda «imports» individuare la funzione «`gethostbyname`». Qual è l'indirizzo dell'import?
3. Quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?
4. Quanti sono, invece, i parametri della funzione sopra?
5. Inserire altre considerazioni macro livello sul malware

Svolgimento

1. Analizzando il codice, l'indirizzo della funzione DllMain corrisponde a 0x1000D02E come è possibile vedere in figura sia in basso che nella barra laterale una volta cliccato sul blocco di funzione corrispondente.



2. L'indirizzo di import della funzione gethostbyname è 0x100163CC ed è situata nella libreria WS2_32



3. La funzione alla locazione di memoria 0x10001656 non ha un nome, molto probabilmente si tratta di una funzione interna.
Per quanto riguarda le variabili locali, sono tutte quelle variabili che hanno un indirizzo negativo rispetto all'indirizzo della funzione, ovvero le variabili elencate di seguito

```
.text:10001656 var_675          = byte ptr -675h
.text:10001656 var_674          = dword ptr -674h
.text:10001656 hLibModule       = dword ptr -670h
.text:10001656 timeout          = timeval ptr -66Ch
.text:10001656 name             = sockaddr ptr -664h
.text:10001656 var_654          = word ptr -654h
.text:10001656 Dst              = dword ptr -650h
```

```
.text:10001656 Parameter       = byte ptr -644h
.text:10001656 var_640         = byte ptr -640h
.text:10001656 CommandLine     = byte ptr -63Fh
.text:10001656 Source          = byte ptr -63Dh
.text:10001656 Data            = byte ptr -638h
.text:10001656 var_637         = byte ptr -637h
.text:10001656 var_544         = dword ptr -544h
.text:10001656 var_50C         = dword ptr -50Ch
.text:10001656 var_500         = dword ptr -500h
.text:10001656 Buf2            = byte ptr -4FCh
.text:10001656 readfds         = fd_set ptr -4BCh
.text:10001656 phkResult       = byte ptr -3B8h
.text:10001656 var_3B0         = dword ptr -3B0h
.text:10001656 var_1A4         = dword ptr -1A4h
.text:10001656 var_194         = dword ptr -194h
.text:10001656 WSAData         = WSAData ptr -190h
```

4. I parametri della funzione sono invece quelli con indirizzo positivo rispetto all'indirizzo della funzione. Nel caso in esame è presente un solo parametro

```
.text:10001656 arg_0           = dword ptr  4
```

5. Esaminando il codice è possibile individuare la creazione di alcuni socket per cui il malware tenterò di connettersi ad un endpoint, così come la presenza di chiamate per la manipolazione del registro di sistema (RegOpenKey, RegCreateKey, RegSetKey...) che servono al malware per ottenere alcuni comportamenti come la persistenza del malware stesso.