Progetto S11-L2

Studente: Simone Mininni

Traccia:

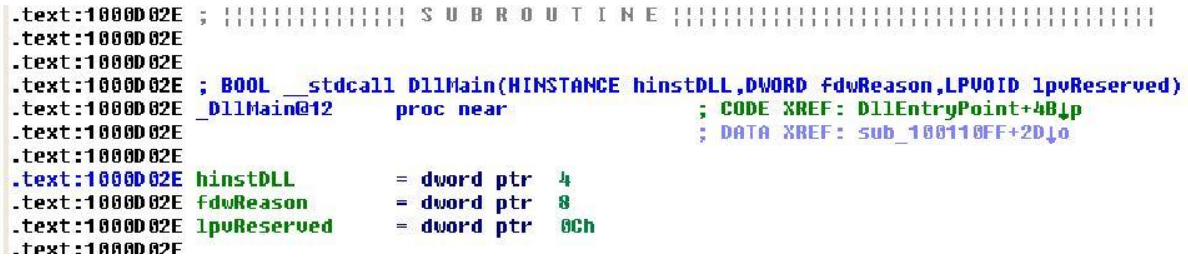
Lo scopo dell'esercizio di oggi è di acquisire esperienza con IDA, un tool fondamentale per l'analisi statica.
A tal proposito, con riferimento al malware chiamato «**Malware_U3_W3_L2** » presente all'interno della cartella «**Esercizio_Pratico_U3_W3_L2** » sul Desktop della macchina virtuale dedicata 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? **Cosa fa la funzione?**

3. Quante sono le **variabili locali** della **funzione** alla locazione di memoria 0x10001656?
4. Quanti sono, invece, i **parametri** della funzione sopra?

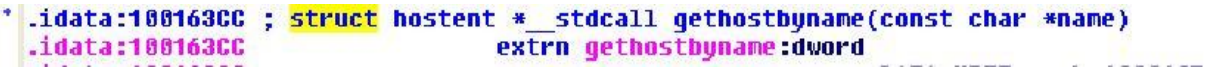
5. Inserire altre considerazioni macro livello sul malware (comportamento)

1.



L' indirizzo della funzione DllMain è 0x1000D02E

2.



L' indirizzo dell' import è 0x100163CC
La funzione "gethostbyname" recupera le informazioni host corrispondenti a un nome host da un database host.
https://learn.microsoft.com/it-it/windows/win32/api/winsock/nf-winsock-gethostbyname

3.

```
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)
.text:10001656 sub_10001656    proc near               ; DATA XREF: DllMain(x,x,x)+C8↓o
.text:10001656
.text:10001656 var_675         = byte ptr -675h
.text:10001656 var_674         = dword ptr -674h
.text:10001656 hModule         = dword ptr -670h
.text:10001656 timeout         = timeval ptr -66Ch
.text:10001656 name            = sockaddr ptr -664h
.text:10001656 var_654         = word ptr -654h
.text:10001656 in              = in_addr ptr -650h
.text:10001656 Parameter       = byte ptr -644h
.text:10001656 CommandLine     = byte ptr -63Fh
.text:10001656 Data            = byte ptr -638h
.text:10001656 var_544         = dword ptr -544h
.text:10001656 var_50C         = dword ptr -50Ch
.text:10001656 var_500         = dword ptr -500h
.text:10001656 var_4FC         = dword ptr -4FCh
.text:10001656 readfds         = fd_set ptr -4BCh
.text:10001656 phkResult       = HKEY__ 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
.text:10001656 arg_0           = dword ptr  4
```

Le variabili locali presenti nella funzione in questione sono 20.
Mentre troviamo solo un parametro passato alla funzione(arg_0).

5.

Studiando il codice si può osservare le funzioni della libreria ws2 di windows.

```
.text:10007ED4                 push    ebx
.text:10007ED5                 push    esi
.text:10007ED6                 lea     eax, [ebp+WSAData]
.text:10007EDC                 push    edi
.text:10007EDD                 push    eax             ; lpWSAData
.text:10007EDE                 push    202h            ; wVersionRequested
.text:10007EE3                 call    ds:WSAStartup
```

Con WSAStartup si apre un socket.

```
.text:10008002                 push    6               ; protocol
.text:10008004                 push    1               ; type
.text:10008006                 push    2               ; af
.text:10008008                 call    ds:socket
.text:1000800E                 cmp     eax, 0FFFFFFFFh
.text:10008011                 mov     [ebp+s], eax
.text:10008014                 jnz     short loc_1000802A
.text:10008016                 call    ds:WSAGetLastError
.text:1000801C                 push    eax
.text:1000801D                 push    offset aSocketGetlaste ; "socket() GetLastError reports %d\n"
.text:10008022                 call    ds:__imp_printf
```

Si costruisce l ‘oggetto socket.

Si chiama la funzione connect, quindi il programma si comporta da client, e tenta una connessione verso l' esterno.
Si potrebbe trattare di una reverse shell.