CONSEGNA S11L2

L'obbiettivo di oggi è quello di acquisire esperienza con IDA Pro, un tool per l'analisi statica dei malware.
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 funzionealla locazione di memoria 0x10001656?
4. Quanti sono, invece, i parametri della funzione sopra?
5. Inserire altre considerazioni macro livello sul malware (comportamento)

1) Come vediamo in Fig.1, l'indirizzo della funzione DLLMain è: 1000D02E

```
.text:1000D02E ; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
.text:1000D02E _DllMain@12     proc near               ; CODE XREF: DllEntryPoint+4B↓p
.text:1000D02E                                         ; DATA XREF: sub_100110FF+2D↓o
.text:1000D02E
.text:1000D02E hinstDLL        = dword ptr  4
.text:1000D02E fdwReason       = dword ptr  8
.text:1000D02E lpvReserved     = dword ptr  0Ch
.text:1000D02E
.text:1000D02E                 mov     eax, [esp+fdwReason]
```

Fig.1

2) Come vediamo in Fig.2, l'indirizzo dell'import è: 100163CC

```
.idata:100163C8                                        ; sub_10001074+1BF↑p ...
.idata:100163CC ; struct hostent *__stdcall gethostbyname(const char *name)
.idata:100163CC                 extrn gethostbyname:dword
```

Fig.2

La funzione **gethostbyname** fa parte delle Windows Socket API (Winsock), e viene usata per recuperare le funzioni di rete, come l'indirizzo IP, di un host dato il suo nome. È stato sostituito con la funzione più aggiornata **getaddrinfo**, che support sia IPV4 che IPV6.

3) Le variabili locali della funzione alla locazione di memoria **0x100001656** sono le seguenti (Fig.3):

```
.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
```

Fig.3

4) Il parametro è solo 1, come vediamo in Fig.4:

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

Fig.4

È importante ricordare la differenza tra parametro è variabile locale. Il parametro è un argomento che viene passato dalla funzione, la variabile invece è definita localmente nel contesto della funzione.

5) Dalla lettura delle stringhe, possiamo dedurre che questo malware sia una backdoor, (Fig.5)

```
'...' xdoors_d:1...  00000023  C  \r\n(1) Enter Current Directory '%s'
'...' xdoors_d:1...  00000067  C  \r\n\r\n**********\r\n[BackDoor Server Update Setup]\r\n*...
'...' xdoors_d:1...  00000006  C  -warn
'...' xdoors_d:1...  00000006  C  -erro
```

Fig.5