Compito S11-L2

Traccia

1. Individuare l’indirizzo della funzione DLLMain.
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 10001656?
4. Quanti sono, invece, i parametri della funzione sopra?
5. Inserire altre considerazioni macro livello sul malware.

Punto 1: DllMain

Al fine di trovare l'indirizzo della funzione DllMain, carichiamo l'eseguibile in IDA Pro. Una volta fatto, premiamo la barra spaziatrice per passare alla modalità testuale e recuperare l'indirizzo della funzione main, che sarà: 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
```

Punto 2: gethostbyname

Apriamo la finestra degli "imports" da IDA Pro e localizziamo la funzione cercata. "gethostbyname" si trova all'indirizzo 100163CC, come mostrato in figura.

00000000100163E4	9	htons	WS2_32
00000000100163E0	15	ntohs	WS2_32
00000000100163DC	4	connect	WS2_32
00000000100163D8	19	send	WS2_32
00000000100163D4	16	recv	WS2_32
00000000100163D0	12	inet_ntoa	WS2_32
00000000100163CC	52	gethostbyname	WS2_32

La funzione gethostbyname è una funzione della libreria Winsock (WS2_32) in ambienti Windows, utilizzata per ottenere informazioni su un host specificato dal suo nome. In altre parole, la funzione converte un nome di dominio in un indirizzo IP.

Punto 3: variabili locali

Per prima cosa, bisogna spostarsi all'indirizzo ricercato tramite la ricerca o la barra laterale. A questo indirizzo, troviamo 20 variabili con offset negativo rispetto a EBP

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

Punto 4: offset positivo

Possiamo notare un solo argomento passato alla funzione, avente offset positivo rispetto a EBP ovvero il parametro “arg_0”.

```
.text:10001656 WSAData         = WSAData ptr -190h
.text:10001656 arg_0           = dword ptr  4
.text:10001656
```

Punto 5: Comportamento malware

Il malware sembra coinvolgere operazioni di rete, in quanto utilizza la funzione gethostbyname per ottenere informazioni su un host specificato dal suo nome di dominio. Questo suggerisce la possibilità che il malware possa essere progettato per comunicare con un server remoto. Per questa motivazione andiamo a supporre che sia un malware di tipo Backdoor.