

# Progetto S11 L2

Davide Andreozzi

## 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 a macro livello sul malware (comportamento)

# 1) Individuare l'indirizzo della funzione DLLMain

Su IDA andiamo a cercare nella sezione **Search** «DLLMain» e possiamo vedere che la funzione è localizzata in **1000D02E**.

```
.text:1000D02E ; ===== S U B R O U T I N E =====
.text:1000D02E
.text: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]
* .text:1000D032      dec     eax
* .text:1000D033      jnz     loc_1000D107
* .text:1000D039      mov     eax, [esp+hinstDLL]
* .text:1000D03D      push    ebx
* .text:1000D03E      mov     ds:hModule, eax
* .text:1000D043      mov     eax, off_10019044
```

## 2) Individuare gethostbyname – Cosa fa?

- Andando nella sezione import possiamo cercare o trovare la funzione all'indirizzo **100163CC**.
- Questa funzione recupera le informazioni host corrispondenti a un nome host, in sostanza va ad ottenere l'indirizzo IPv4 di un nome di dominio.

00000000100163C8	11	inet_addr	WS2_32
00000000100163CC	52	gethostbyname	WS2_32
00000000100163D0	12	inet_ntoa	WS2_32
00000000100163D4	13	...	...

```
.idata:100163CC ; struct hostent *__stdcall gethostbyname(const char *name)
.idata:100163CC         extrn gethostbyname:dword
.idata:100163CC         ; CODE XREF: sub_10001074:loc_100011AF↑p
.idata:100163CC         ; sub_10001074+1D3↑p ...
```

### 3) Quante variabili in 0x10001656

Come sappiamo le variabili sono valori con offset negativo.

Spostandoci all'indirizzo sopracitato possiamo contare 20 Variabili come evidenziato nella figura accanto.

```
.text:10001656  
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)  
.text:10001656 sub_10001656 proc near ; DATA XREF: DllMain(x,x,  
.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  
.text:10001656  
.text:10001656 sub esp, 678h
```

## 4) Quanti parametri

A differenza delle variabili, i parametri sono valori con offset positivo.

In funzione di quanto detto possiamo notare la presenza di un solo parametro evidenziato in figura accanto.

**arg\_0**

```
.text:10001656
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)
.text:10001656 sub_10001656 proc near ; DATA XREF: DllMain(x,x,
.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
.text:10001656
.text:10001656 sub esp, 678h
```

## 5) Considerazioni sul malware

Questo malware va a importare le seguenti librerie:

GDB2.dll	17	00016BE0	00000000	00000000	000170A2	00016084
PSAPI.DLL	2	00016E88	00000000	00000000	000170DA	0001632C
WS2_32.dll	15	00016F20	00000000	00000000	000170E4	000163C4
iphlpapi.dll	1	00016F60	00000000	00000000	00017102	00016404
KERNEL32.dll	89	00016C28	00000000	00000000	0001773E	000160CC
USER32.dll	26	00016E94	00000000	00000000	00017920	00016338
ADVAPI32.dll	32	00016B5C	00000000	00000000	00017BA6	00016000
ole32.dll	5	00016F68	00000000	00000000	00017C0C	0001640C
OLEAUT32.dll	2	00016E7C	00000000	00000000	00017C16	00016320
MSVFW32.dll	5	00016E64	00000000	00000000	00017C68	00016308
WINMM.dll	7	00016F00	00000000	00000000	00017CEC	000163A4
MSVCRT.dll	52	00016D90	00000000	00000000	00017EC8	00016234

Il malware potrebbe andare a modificare valori di registro leggere e gestire i processi di sistema e installare una backdoor.

Analizzando il codice in Assembly possiamo notare una parte in cui tramite l'utilizzo di cmd.exe si va ad avviare una shell.

Inoltre, in una parte di codice il programma tramite la funzione «sleep» permette al processo di mettersi in pausa temporaneamente.