

S11-L2



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)

```
IDA View-A | Hex View-A | Structures | En Enums | Imports | Exports |
.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
```

Una volta aver individuato la funzione DllMain all'interno del codice, ci basterà selezionare in alto Hex-View per visualizzare l'indirizzo della funzione in esadecimale

```
1000D02E  F8 05 74 05 05 F8 01 75 E9 5F 5E 5B C9 C2 08 00  .C.a .00_ l+..
1000D02E  8B 44 24 08 48 0F 85 CE 00 00 00 8B 44 24 04 53  iD$.H.à+...iD$.S
1000D02E  02 00 20 00 10 01 44 00 01 10 56 82 C0 00 57 50  ú 0 iDÉ u$+ WP
```


Address	Ordinal	Name	Library
000000...		fprintf	MSVCRT
000000...		fread	MSVCRT
000000...		free	MSVCRT
000000...		fseek	MSVCRT
000000...		ftell	MSVCRT
000000...		fwrite	MSVCRT
000000...	52	gethostbyname	WS2_32

Una volta nella scheda “imports” ci si presenteranno tutte le funzioni importate all’interno del codice, potremmo sia scorrere per controllarle tutte sia cercarla se ne cerchiamo una specifica. Questa funzione è spesso utilizzata per ottenere l’indirizzo IP associato a un determinato nome host.

```

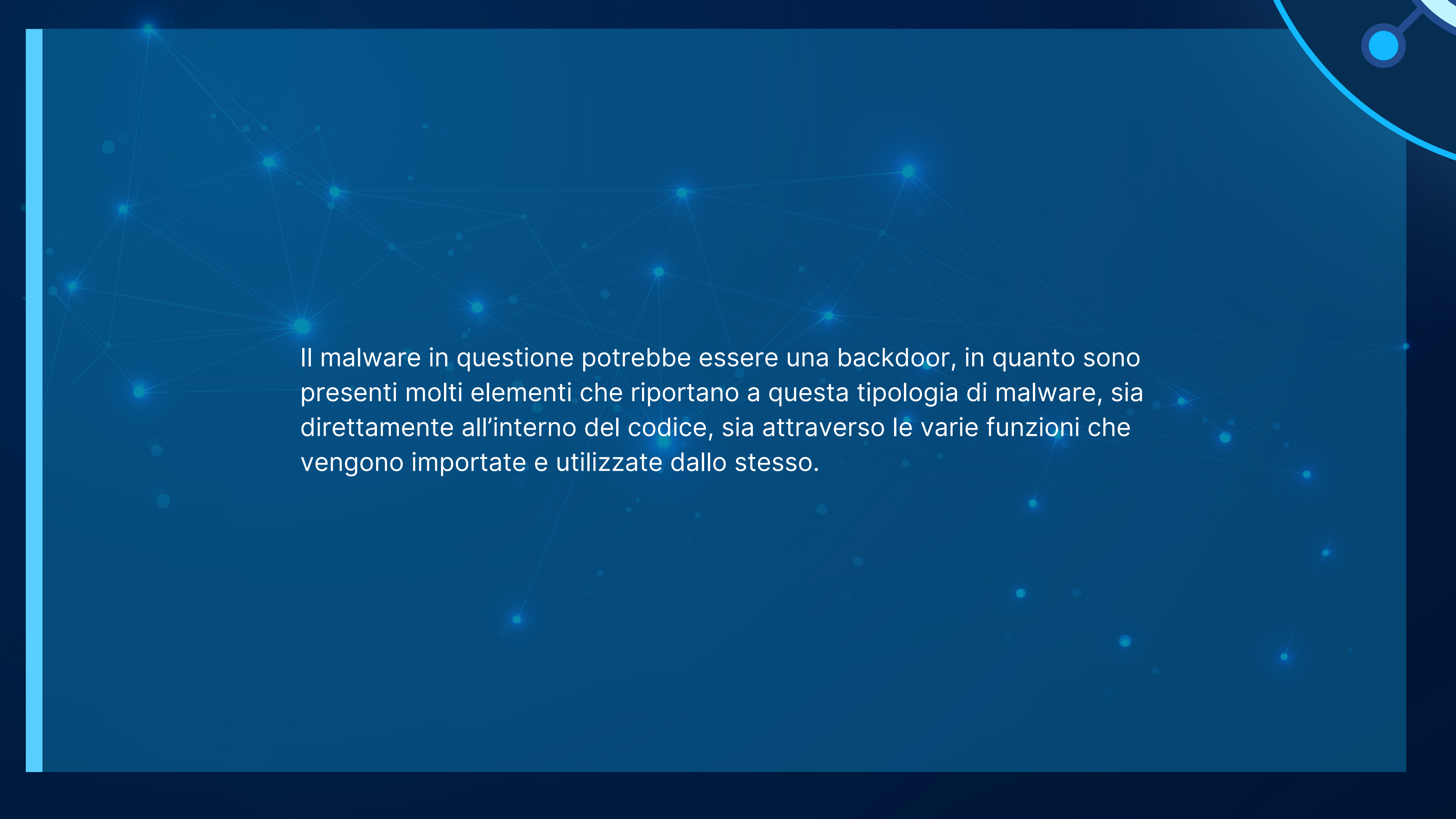
.idata:100163C8 ; sub_10001074+1BF↑p ...
* .idata:100163CC ; struct hostent *__stdcall gethostbyname(const char *name)
.idata:100163CC extrn gethostbyname:dword
.idata:100163CC : CODE XREF: sub_10001074

```

per andare a un indirizzo di memoria specifico possiamo selezionare "jump to address" seguito dall'indirizzo che ci interessa.

```
.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
```

Possiamo notare che all'indirizzo specificato sono presenti molte variabili locali, in quanto sono tutte variabili locali eccetto arg_0 che è un parametro. Le variabili sono indicate dalle etichette seguite da un offset negativo, i parametri invece sono rappresentati dall'etichetta "arg_0" con un offset positivo.



Il malware in questione potrebbe essere una backdoor, in quanto sono presenti molti elementi che riportano a questa tipologia di malware, sia direttamente all'interno del codice, sia attraverso le varie funzioni che vengono importate e utilizzate dallo stesso.