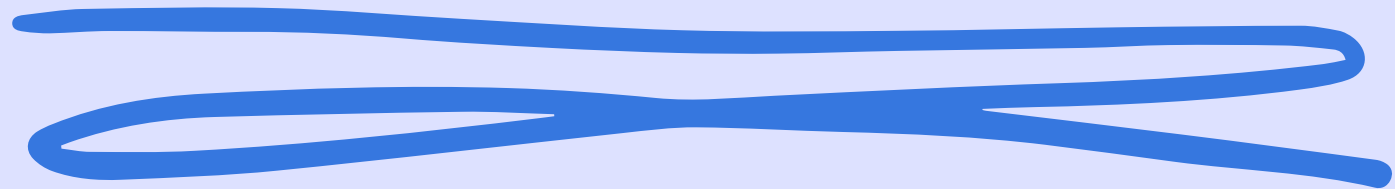


S11 L2

Analisi Statica Avanzata



Daniele Zizzi

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)

L'indirizzo della funzione main è
1000D02E.

In basso abbiamo i parametri e la
copia dell'indirizzo in eax.

La funzione in questione, non fa
nulla di particolare. Main.dll viene
richiamata dal sistema operativo
quando viene caricata o scaricata
una dll.

Functions window											
Function name	Segment	Start	Length	R	F	L	S	B	T	=	
sub_1000BD79	.text	1000BD79	000000AC	R	.	.	.	B	T	.	
sub_1000BE25	.text	1000BE25	0000012D	R	.	.	.	B	T	.	
sub_1000BF52	.text	1000BF52	00000231	R	.	.	.	B	T	.	
sub_1000C183	.text	1000C183	000000CE	R	.	.	.	B	T	.	
sub_1000C251	.text	1000C251	0000013D	R	.	.	.	B	.	.	
sub_1000C3BA	.text	1000C3BA	000000AF	R	.	.	.	B	.	.	
sub_1000C469	.text	1000C469	000000B3	R	.	.	.	B	T	.	
sub_1000C51C	.text	1000C51C	00000048	R	.	.	.	B	.	.	
sub_1000C564	.text	1000C564	000000C9	R	.	.	.	B	T	.	
sub_1000C62D	.text	1000C62D	0000010D	R	.	.	.	B	T	.	
sub_1000C73A	.text	1000C73A	000001B0	R	.	.	.	B	T	.	
sub_1000C8EA	.text	1000C8EA	000000F5	R	.	.	.	B	T	.	
HandlerProc	.text	1000C9DF	00000077	R	T	.	
sub_1000CA56	.text	1000CA56	000001B0	R	.	.	.	B	.	.	
sub_1000CC06	.text	1000CC06	0000032A	R	.	.	.	B	T	.	
ServiceMain	.text	1000CF30	000000FE	R	.	.	.	B	T	.	
DllMain(x,x,x)	.text	1000D02E	000000DF	R	T	.	
...	R	.	.	.	B	T	.	

```
.text:1000D02E
.text:1000D02E ; SUBROUTINE
.text:1000D02E
.text:1000D02E
.text:1000D02E ; BOOL __stdcall DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPUVOID 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]
```

L'indirizzo dell'import di
"gethostbyname" è 100163CC.
Questa funzione permettere di
ottenere l'indirizzo ip dal nome
della macchina.

Imports			
Address	Ordinal	Name	Library
10016274		fopen	MSVCRT
100162E4		fprintf	MSVCRT
10016234		fread	MSVCRT
100162DC		free	MSVCRT
100162D8		fseek	MSVCRT
10016278		ftell	MSVCRT
100162A0		fwrite	MSVCRT
100163CC	52	gethostbyname	WS2_32

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

Le variabili locali sono tutte quelle che precedono EBP(evidenziate in blu). I parametri si trovano dopo la locazione di memoria di EBP, quindi solo arg_0.

```

; DWORD __stdcall sub_10001656(LPVOID)
sub_10001656 proc near

var_675= byte ptr -675h
var_674= dword ptr -674h
hModule= dword ptr -670h
timeout= timeval ptr -66Ch
name= sockaddr ptr -664h
var_654= word ptr -654h
in= in_addr ptr -650h
Parameter= byte ptr -644h
CommandLine= byte ptr -63Fh
Data= byte ptr -638h
var_544= dword ptr -544h
var_50C= dword ptr -50Ch
var_500= dword ptr -500h
var_4FC= dword ptr -4FCh
readfds= fd_set ptr -4BCh
phkResult= HKEY__ ptr -3B8h
var_3B0= dword ptr -3B0h
var_1A4= dword ptr -1A4h
var_194= dword ptr -194h
WSAData= WSAData ptr -190h
arg_0= dword ptr 4

sub     esp, 678h
push    ebx
push    ebp
push    esi
push    edi
call    sub_10001000
test    eax, eax
jnz     short loc_100016BC

```

```

.text:10001656
.text:10001656 ; :::::::::::::::::::: S U B R O U T I N E ::::::::::::::::::::
.text:10001656
.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
.text:10001656
.text:10001656 sub             esp, 678h

```

Il malware in questione è una backdoor. Richiama delle funzioni di windows mentre è in esecuzione, modifica, crea e si inietta nei servizi in esecuzione. Fa una scalata dei privilegi, ottiene informazioni sulla macchina. Scarica e scrive file sull'host vittima.