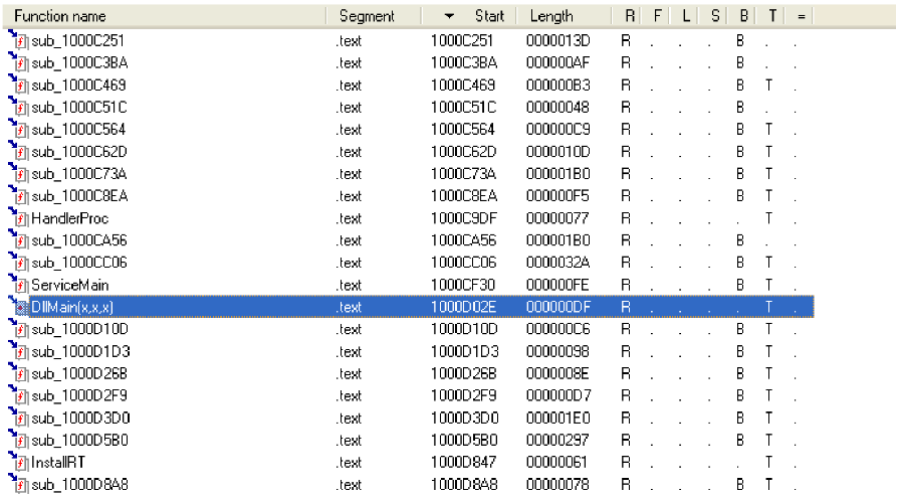
Amarfii Adrian 17/01/2023
Report
Task

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.

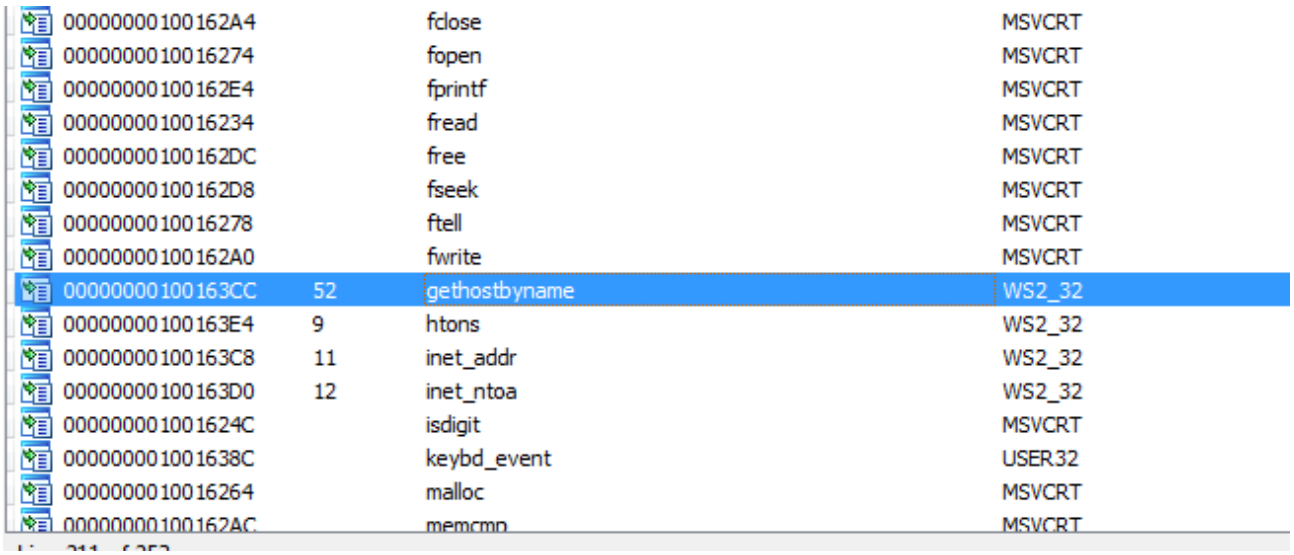
1. Individuare l'indirizzo della funzione DLLMain
2. Dalla scheda «imports» individuare la funzione «gethostbyname». Qual è l'indirizzo dell'import?
3. Quante sono le variabili locali della funzione alla locazione di memoria 0×10001656?
4. Quanti sono, invece, i parametri della funzione sopra?

Partiamo con il primo punto. Dopo aver aperto il nostro malware con l'aiuto di IDA PRO andiamo sulla sezione -Function -.e digitiamo la funzione che ci serve che nel nostro caso è DLLMAIN e vediamo anche l'indirizzo che nel nostro caso → 1000D02E



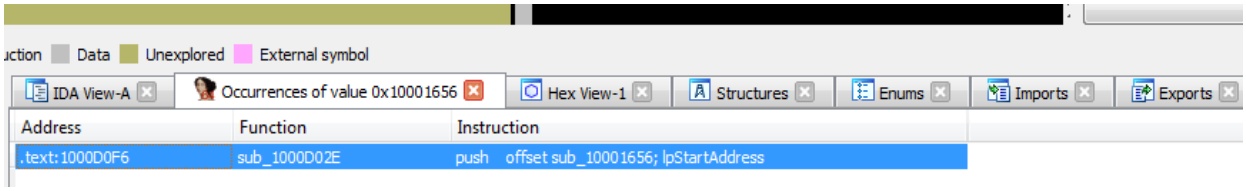
| Function name | Segment | Start | Length | R | F | L | S | B | T | = |
|---|---|---|---|---|---|---|---|---|---|---|
| sub_1000C251 | .text | 1000C251 | 0000013D | R | . | . | . | B | . | . |
| sub_1000C3BA | .text | 1000C3BA | 000000AF | R | . | . | . | B | . | . |
| sub_1000C469 | .text | 1000C469 | 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 | . |
| sub_1000D10D | .text | 1000D10D | 000000C6 | R | . | . | . | B | T | . |
| sub_1000D1D3 | .text | 1000D1D3 | 00000098 | R | . | . | . | B | T | . |
| sub_1000D26B | .text | 1000D26B | 0000008E | R | . | . | . | B | T | . |
| sub_1000D2F9 | .text | 1000D2F9 | 000000D7 | R | . | . | . | B | T | . |
| sub_1000D3D0 | .text | 1000D3D0 | 000001E0 | R | . | . | . | B | T | . |
| sub_1000D5B0 | .text | 1000D5B0 | 00000297 | R | . | . | . | B | T | . |
| InstallRT | .text | 1000D847 | 00000061 | R | . | . | . | . | T | . |
| sub_1000D8A8 | .text | 1000D8A8 | 00000078 | R | . | . | . | B | T | . |

2)Per fare il punto numero 2 dobbiamo andare nella sezione imports e trovare il nostro ricercato – gethostbyname- e troviamo il suo indirizzo che nel nostro caso è → 100163CC



| | | | |
|---|---|---|---|
| 00000000100162A4 | | fclose | MSVCRT |
| 0000000010016274 | | fopen | MSVCRT |
| 00000000100162E4 | | fprintf | MSVCRT |
| 0000000010016234 | | fread | MSVCRT |
| 00000000100162DC | | free | MSVCRT |
| 00000000100162D8 | | fseek | MSVCRT |
| 0000000010016278 | | ftell | MSVCRT |
| 00000000100162A0 | | fwrite | MSVCRT |
| 00000000100163CC | 52 | gethostbyname | WS2_32 |
| 00000000100163E4 | 9 | htons | WS2_32 |
| 00000000100163C8 | 11 | inet_addr | WS2_32 |
| 00000000100163D0 | 12 | inet_ntoa | WS2_32 |
| 000000001001624C | | isdigit | MSVCRT |
| 000000001001638C | | keybd_event | USER32 |
| 0000000010016264 | | malloc | MSVCRT |
| 00000000100162AC | | memcmp | MSVCRT |

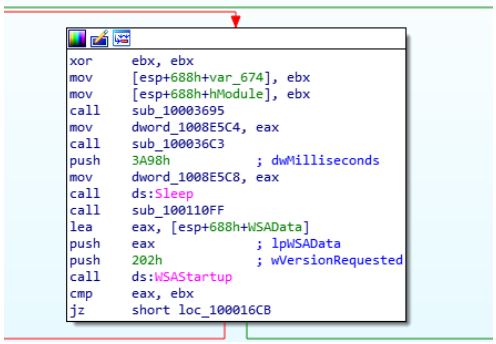
3)Punto numero 3 dobbiamo vedere quali sono le variabili locali della funzione alla locazione di memoria 0x10001656 :per fare questo andiamo nella sezione IDA View-A e con la funzione cerca cerchiamo ox10001656 find all occurences.




E se clichiamo 2 volte ci indirizza alla parte di codice :

```
.text:1000D0F4                 push    edi             ; dwCreationFlags
.text:1000D0F5                 push    edi             ; lpParameter
.text:1000D0F6                 push    offset sub_10001656 ; lpStartAddress
.text:1000D0FB                 push    edi             ; dwStackSize
.text:1000D0FC                 push    edi             ; lpThreadAttributes
.text:1000D0FD                 call    ebx ; CreateThread
.text:1000D0FF                 pop     edi
.text:1000D100                 pop     esi
.text:1000D101                 mov     ds:dword_10093008, eax
.text:1000D106                 pop     ebx
.text:1000D107
.text:1000D107 loc_1000D107:                           ; CODE XREF: sub_1000D02E+5↑j
.text:1000D107                 push    1
.text:1000D109                 pop     eax
.text:1000D10A                 retn    0Ch
.text:1000D10A sub_1000D02E    endp
.text:1000D10A
.text:1000D10D
.text:1000D10D ; =============== S U B R O U T I N E =======================================
.text:1000D10D
```

Facendo doppio clik ci vengono fuori tutte le variabili:




```
; DWORD __stdcall sub_10001656(LPVOID lpThrea
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
Dst= dword ptr -650h
Str1= byte ptr -644h
var_640= byte ptr -640h
CommandLine= byte ptr -63Fh
Str= byte ptr -63Dh
var_638= byte ptr -638h
var_637= byte ptr -637h
var_544= byte ptr -544h
var_50C= dword ptr -50Ch
var_500= byte ptr -500h
Buf2= byte ptr -4FCh
readfds= fd_set ptr -4BCh
buf= byte ptr -3B8h
var_3B0= dword ptr -3B0h
var_1A4= dword ptr -1A4h
var_194= dword ptr -194h
WSAData= WSAData ptr -190h
lpThreadParameter= dword ptr  4

sub     esp, 678h
```