

# Analisi statica avanzata con Ida

S11L2

# Indice

- Traccia
- DLLMain
- GetHostByName
- Variabili vs Parametri
- Considerazioni macro livello

# 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)



# DLLMain

Come si può vedere dal seguente screenshot l'indirizzo della funzione DLLMain è: **1000D02E**.

```
.text: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
```

# GetHostByName

Dal seguente screenshot possiamo notare che la funzione è nella postazione **100163CC**.

```
.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:loc_100011AF↑p
.idata:100163CC     ; sub_10001074+1D3↑p ...
```

Questa funzione cerca di ottenere il nome dell'host.



# Variabili vs Parametri

```
.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 readfds      = fd_set ptr -48Ch
.text:10001656 phkResult    = byte ptr -388h
.text:10001656 var_380      = dword ptr -380h
.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
```

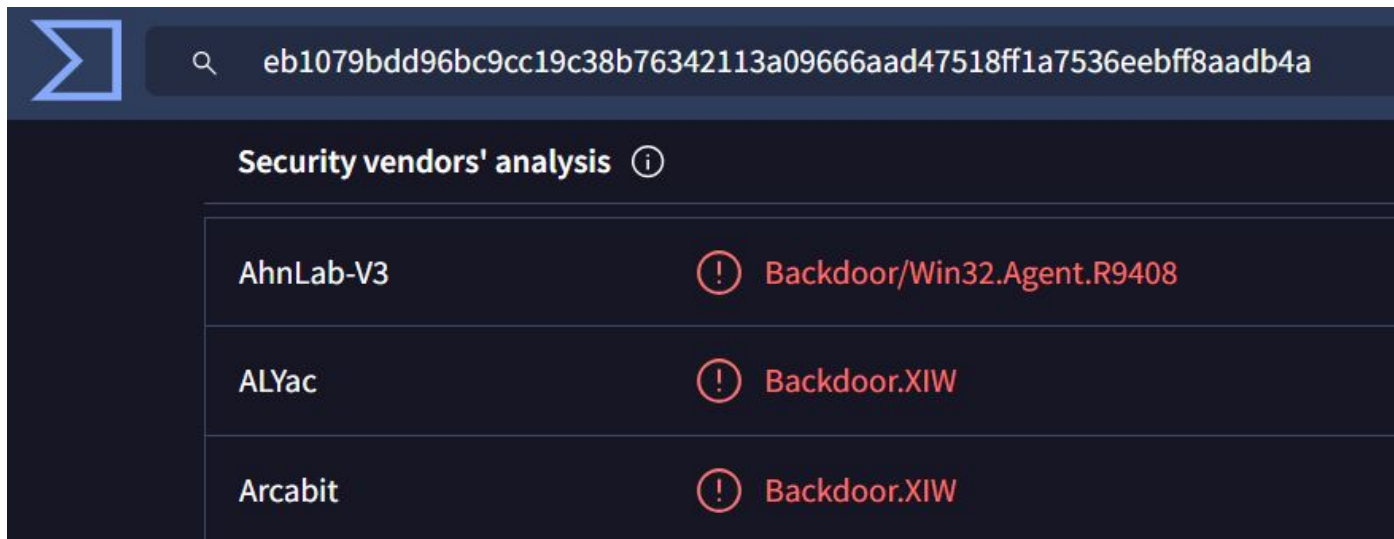
Come vediamo in figura le **variabili** sono quelle con un offset negativo e ne abbiamo identificate 22.

I **parametri** invece sono quelli con un offset positivo e ne abbiamo identificato solamente 1.



# Considerazioni macro livello

Attraverso il calcolo dell'ash siamo andati ad identificare su VirusTotal di che tipo di malware si trattasse e possiamo ipotizzare che si tratti di un malware che sfrutta un backdoor.



Security vendors' analysis ⓘ	
AhnLab-V3	⚠ Backdoor/Win32.Agent.R9408
ALYac	⚠ Backdoor.XIW
Arcabit	⚠ Backdoor.XIW



Grazie