



**MALWARE ANALYSIS**

# **ANALISI STATICA AVANZATA CON IDA**

**RAFAEL MANGO**



# 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)

1

Individuare l'indirizzo della funzione  
DLLMain(così com'è, in esadecimale)

come possiamo vedere dall'immagine  
l'indirizzo della funzione DLLMain e  
1000D02E

per andare a trovare l'indirizzo siamo andati ad  
aprire il malware con IDA , come nell'immagine in  
alto a destra siamo andati a evidenziare le prime  
righe di codice e sotto il codice vicino alla riga con  
scritto 0000C42E troviamo l'indirizzo della  
funzione DLLMain per verificare che non ci  
sbagliamo andiamo nella finestra Hex View-a e ci  
ritroveremo nella finestra come nell'immagine in  
basso a destra

The screenshot shows the IDA Pro interface. The top pane displays assembly code for a function named `DllMain`. The code includes a `stdcall` declaration, a `near` proc, and several `hinstDLL`, `fdwReason`, and `lpvReserved` variables. The bottom pane shows a hex dump of memory starting at address `0000C42E`. The first four bytes of the dump are `8B 44 24 08`, which are highlighted in green. The address `0000C42E` is also highlighted in green in the address field of the hex dump.

```
; BOOL __stdcall DllMain(HINSTANCE, DWORD, LPVOID)
DllMain@12 proc near

hinstDLL= dword ptr 4
fdwReason= dword ptr 8
lpvReserved= dword ptr 0Ch

mov     eax, [esp+fdwReason]
dec     eax
```

Address	Hex
0000C42E	8B 44 24 08 48 0F 85 CE
0000C43E	A3 00 30 09 10 A1 44 90
0000C44E	E8 F9 7E 00 00 8B 1D 08
0000C45E	10 33 FF 59 85 C0 74 23
0000C46E	C0 0D 68 58 39 09 10 50
0000C47E	0A 57 57 57 68 74 10 00
0000C48E	83 C0 0D 50 E8 B5 7E 00
0000C49E	90 01 10 6A 06 83 C0 0D
0000C4AE	83 C4 0C 85 C0 75 11 57
0000C4BE	57 FF D3 A3 04 30 09 10
0000C4CE	62 01 10 83 C0 0D 6A 10
0000C4DE	A1 38 90 01 10 6A 05 83
0000C4EE	FF D6 83 C4 18 57 57 57
0000C4FE	D3 5F 5E A3 08 30 09 10
0000C50E	8B EC 81 EC 00 03 00 00
0000C51E	56 68 2A 04 00 00 FF 15

0000C42E 000000001000D02E: DllMain(x,x,x)

# 2

Dalla scheda «imports» individuare la funzione «gethostbyname». Qual è l'indirizzo dell'import?  
Cosa fa la funzione?

*Per andare a trovare la funzione gethostbyname siamo andati nella scheda imports come in figura cliccandoci ci troveremo nella scheda in figura. tornando nella scheda IDA ViewA troviamo anche l'indirizzo che sarà 10001656 .*

*La funzione gethostbyname è una funzione di libreria standard di Windows utilizzata per ottenere informazioni sulle entità host, come gli indirizzi IP, utilizzando il nome host. Questo tipo di funzione è comunemente utilizzato nelle applicazioni di rete per convertire nomi di dominio in indirizzi IP, consentendo alle applicazioni di comunicare con gli host identificati da nomi simbolici anziché da indirizzi numerici.*



Address	Ordinal	Name	Library
000000...		waveInPrepareHeader	WINMM
000000...		waveInAddBuffer	WINMM
000000...		waveInStart	WINMM
000000...	18	select	WS2_32
000000...	11	inet_addr	WS2_32
000000...	52	gethostbyname	WS2_32
000000...	12	inet_ntoa	WS2_32
000000...	16	recv	WS2_32
000000...	19	send	WS2_32
000000...	4	connect	WS2_32
000000...	15	ntohs	WS2_32
000000...	9	htons	WS2_32
000000...	21	setsockopt	WS2_32

```
Imports | X Exports |
; DWORD __stdcall sub_10001656(LPVOID)
sub_10001656 proc near

var_675= byte ptr -675h
var_674= dword ptr -674h
hLibModule= dword ptr -670h
timeout= timeval ptr -66Ch
name= sockaddr ptr -664h
var_654= word ptr -654h
Dst= dword ptr -650h
Parameter= byte ptr -644h
var_640= byte ptr -640h
CommandLine= byte ptr -63Fh
```



# 3

## Quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?

possiamo vedere che ci sono diverse variabili locali definite all'interno della funzione sub\_10001656. Queste variabili locali sono elencate con i loro relativi offset rispetto allo stack frame, indicati dai valori negativi (ad esempio, -675h, -674h, -670h, ecc.).

Contando il numero di variabili locali definite nel codice fornito, possiamo determinare quanti sono:

1. var\_675
2. var\_674
3. hLibModule
4. timeout
5. name
6. var\_654
7. Dst
8. Parameter
9. var\_640
10. CommandLine
11. Source
12. Data
13. var\_637
14. var\_544
15. var\_50C
16. var\_500
17. Buf2
18. readfds
19. phkResult
20. var\_3B0
21. var\_1A4
22. var\_194
23. WSADATA

Quindi, ci sono 23 variabili locali definite all'interno della funzione sub\_10001656 alla locazione di memoria 0x10001656.

# 4 Quanti sono, invece, i parametri della funzione sopra?

La funzione sub\_10001656 accetta un solo parametro, come indicato dalla dichiarazione della procedura sub\_10001656:

**sub\_10001656 proc near**

E dalla sua firma:

**; DWORD \_\_stdcall sub\_10001656(LPVOID)**

Il commento sopra la firma indica che la funzione accetta un parametro di tipo LPVOID, che è un puntatore a void. Inoltre, l'etichetta arg\_0 definita come dword ptr 4 indica il parametro passato alla funzione attraverso il registro eax.

Quindi, la funzione sub\_10001656 accetta un solo parametro.

# 5 Inserire altre considerazioni macro livello sul malware

Guardando il malware ho notato che ci sarebbero altre considerazioni macro livello da fare . Di seguito ho riportato alcune considerazioni sul malware, basate sul comportamento e sulle caratteristiche generali che potrebbero essere rilevanti per un'analisi più approfondita:

**Furtività:** Il malware potrebbe cercare di mascherare la propria presenza, ad esempio mimetizzandosi come un file legittimo o utilizzando tecniche di occultamento per evitare la rilevazione da parte di software antivirus o altri strumenti di sicurezza.

**Persistenza:** Potrebbe tentare di mantenere la sua presenza nel sistema dopo il riavvio, ad esempio aggiungendo voci al registro di sistema o inserendo codice nei punti di avvio del sistema operativo.

**Comunicazione remota:** Potrebbe tentare di stabilire una comunicazione con un server remoto per ricevere istruzioni, inviare dati sensibili o scaricare componenti aggiuntivi. Questa comunicazione potrebbe essere criptata o utilizzare protocolli non standard per sfuggire alla rilevazione.

**Rubare informazioni:** Il malware potrebbe cercare di rubare informazioni sensibili, come credenziali di accesso, dati finanziari o altre informazioni personali memorizzate sul sistema infetto.

**Infezione di file:** Potrebbe cercare di infettare altri file nel sistema per diffondere ulteriormente il malware e garantire la sua persistenza.

**Modifiche del sistema:** Potrebbe effettuare modifiche non autorizzate al sistema, ad esempio disabilitando il firewall, bloccando l'accesso a determinati siti web o modificando le impostazioni di sicurezza.

**Utilizzo delle risorse di sistema:** Il malware potrebbe consumare risorse di sistema, come CPU, memoria o larghezza di banda di rete, riducendo le prestazioni complessive del sistema e causando rallentamenti o blocchi.

**Evasione delle tecniche di analisi:** Potrebbe essere progettato per eludere l'analisi statica o dinamica utilizzando tecniche come l'opacizzazione del codice, l'auto-modifica o il rilevamento dell'ambiente di analisi.

**Polimorfismo:** Il malware potrebbe utilizzare tecniche di polimorfismo per modificare la propria struttura e il proprio comportamento nel tempo al fine di eludere la rilevazione da parte dei motori antivirus.

**Attacchi mirati:** In alcuni casi, il malware potrebbe essere progettato per condurre attacchi mirati contro specifici obiettivi, come aziende o istituzioni governative, al fine di rubare informazioni sensibili o compromettere la loro sicurezza.

**È importante esaminare attentamente il contesto e condurre un'analisi approfondita per comprendere appieno il comportamento e le intenzioni del codice sospetto.**

The background features a series of thin, black, wavy lines that create a sense of movement and depth. On the left side, the lines curve upwards and then downwards, forming a large, open shape. On the right side, the lines are more densely packed and curve inwards, creating a more complex, almost circular pattern. The overall effect is a minimalist yet dynamic abstract design.

**THANK YOU**