



02/04/2024

# Analisi statica avanzata con IDA

Prepared by:  
Manuel Buonanno

Organized by:



# Indice

1) Traccia.....	3
2) DLLMain.....	4
3) gethostbyname .....	5
4) variabili e parametri.....	6
5) Considerazioni.....	7

# Traccia

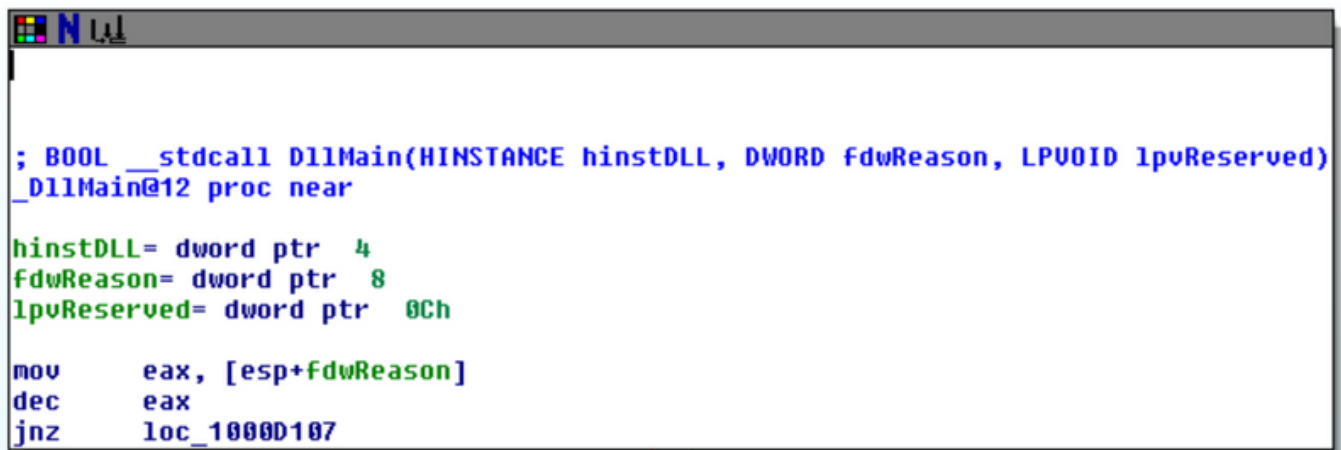
Lo scopo dell'esercizio di oggi è di acquisire esperienza con IDA, un tool fondamentale per l'analisi statica.

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).

# DLLMain

Completato il caricamento del file eseguibile IDA ci presenterà l'interfaccia di analisi, come da figura. Questo è chiamato «disassemblypanel» ed è dove IDA Pro ci mostra la traduzione del codice macchina dell'eseguibile in codice Assembly. come IDA Pro abbia riconosciuto la funzione «intmain» durante la fase di traduzione. Il codice assembly che segue è dunque parte della funzione «main»



```
; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
_DllMain@12 proc near

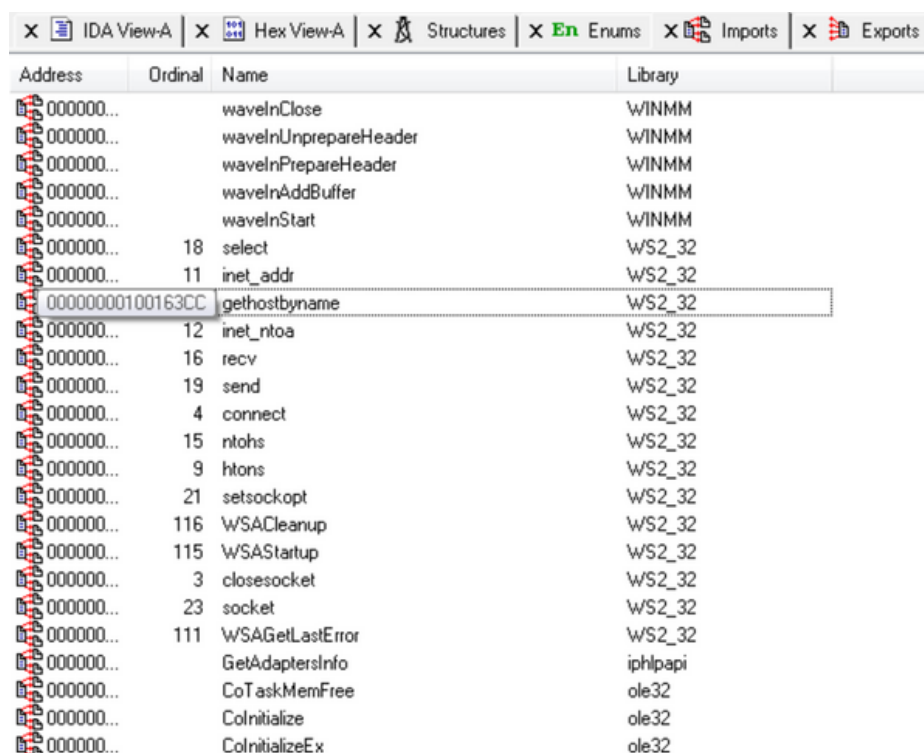
hinstDLL= dword ptr  4
fdwReason= dword ptr  8
lpvReserved= dword ptr  0Ch

mov     eax, [esp+fdwReason]
dec     eax
jnz     loc_1000D107
```

Possiamo notare come IDA Pro abbia riconosciuto la funzione «intmain» durante la fase di traduzione. Il codice assembly che segue è dunque parte della funzione «main»

# gethostbyname

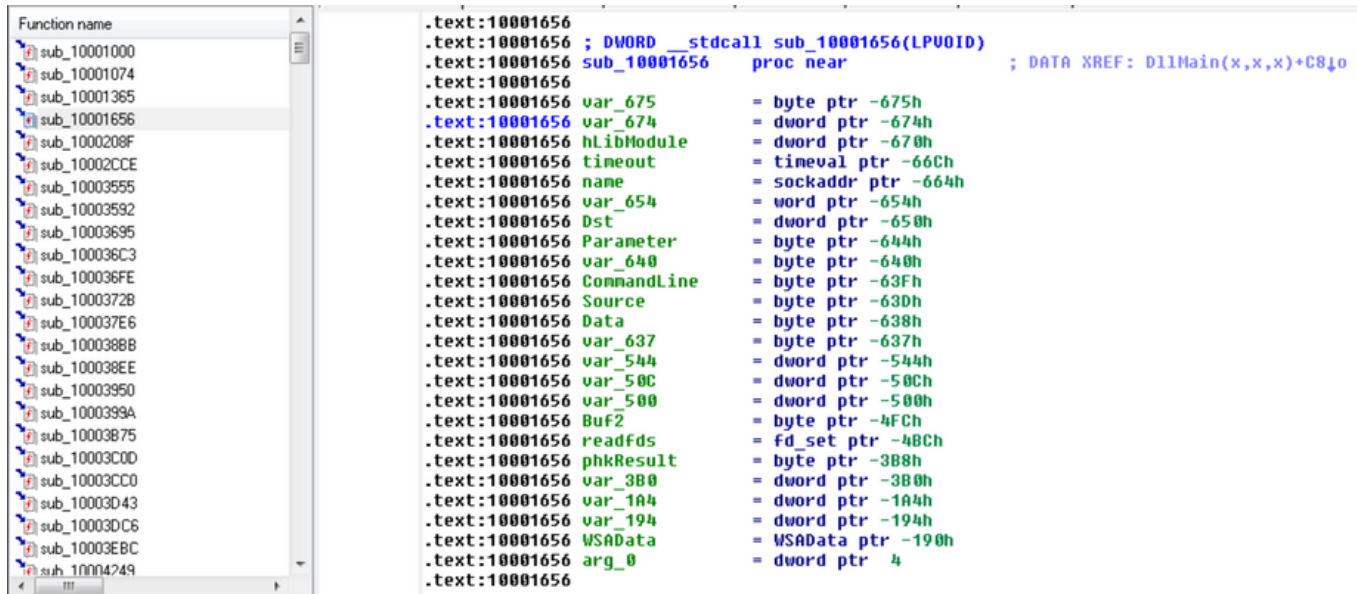
La funzione <<gethostbyname>> è una chiamata di sistema che viene utilizzata per risolvere un nome di host in un indirizzo IP. Questa funzione è comunemente utilizzata nei programmi di rete per ottenere l'indirizzo IP di un determinato nome di dominio. Quando un programma chiama <<gethostbyname>> con un nome di dominio come parametro, il sistema operativo si occupa di cercare l'indirizzo IP associato a quel nome di dominio utilizzando il sistema di risoluzione dei nomi, come DNS (Domain Name System). Una volta che l'indirizzo IP è stato trovato, viene restituito al programma chiamante.



Address	Ordinal	Name	Library
000000...		waveInClose	WINMM
000000...		waveInUnprepareHeader	WINMM
000000...		waveInPrepareHeader	WINMM
000000...		waveInAddBuffer	WINMM
000000...		waveInStart	WINMM
000000...	18	select	WS2_32
000000...	11	inet_addr	WS2_32
00000000100163CC		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
000000...	116	WSACleanup	WS2_32
000000...	115	WSAStartup	WS2_32
000000...	3	closesocket	WS2_32
000000...	23	socket	WS2_32
000000...	111	WSAGetLastError	WS2_32
000000...		GetAdaptersInfo	iphlpapi
000000...		CoTaskMemFree	ole32
000000...		CoInitialize	ole32
000000...		CoInitializeEx	ole32

Quindi, la funzione <<gethostbyname>> in assembly svolge il ruolo di interfacciamento con il sistema di risoluzione dei nomi del sistema operativo per ottenere gli indirizzi IP associati a nomi di dominio.

# locali e parametri in 0x10001656



The screenshot shows a debugger window. On the left, a list of functions is displayed, with `sub_10001656` selected. On the right, the assembly code for `sub_10001656` is shown. The code starts with a `stdcall` instruction and a `proc near` declaration. It then lists several local variables and their addresses, followed by a list of parameters and their addresses. The parameters include `hLibModule`, `timeout`, `name`, `var_654`, `Dst`, `Parameter`, `var_640`, `CommandLine`, `Source`, `Data`, `var_637`, `var_544`, `var_50C`, `var_500`, `Buf2`, `readfds`, `phkResult`, `var_3B0`, `var_1A4`, `var_194`, `WSAData`, and `arg_0`.

```
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)
.text:10001656 sub_10001656 proc near ; DATA XREF: DllMain(x,x,x)+C8Jo
.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 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
.text:10001656
```

- Parametri: I parametri sono valori o indirizzi passati a una subroutine o a una funzione per definire l'input necessario per eseguire un'operazione specifica. Possono essere passati attraverso registri specifici, come registri general-purpose, o attraverso lo stack.
- Variabili: Le variabili sono dati memorizzati in memoria che possono essere modificati durante l'esecuzione del programma. Questi dati possono includere sia i valori assegnati direttamente nel codice (variabili statiche) che quelli allocati durante l'esecuzione (variabili dinamiche). Le variabili possono essere utilizzate per mantenere lo stato del programma, memorizzare risultati intermedi o qualsiasi altra informazione necessaria.

Le variabili locali si distinguono dal segno "-". Mentre è presente solo un parametro.

```
.text:10001656 arg_0 = dword ptr 4
```

# Considerazioni

In generale, il malware sembra essere progettato per eseguire azioni dannose o indesiderate su un sistema compromesso. Ecco una descrizione generale delle sue funzionalità:

1. **Controllo del processo corrente:** Il malware verifica se il processo corrente è "rundll32.exe" o "rundll64.exe". Questo potrebbe indicare che il malware si nasconde dietro questi processi legittimi per evitare di destare sospetti.
2. **Ottenimento di un indirizzo IP:** Se il processo corrente non corrisponde a "rundll32.exe" o "rundll64.exe", il malware cerca di ottenere un indirizzo IP tramite la funzione <<gethostbyname>>.
3. **Esecuzione del comando "ipconfig /flushdns":** Dopo aver ottenuto l'indirizzo IP, il malware esegue il comando di sistema "ipconfig /flushdns". Questo comando serve a cancellare la cache del sistema DNS, il che potrebbe essere utile per nascondere le tracce dell'attività del malware o interferire con la risoluzione dei nomi di dominio sulla macchina compromessa.