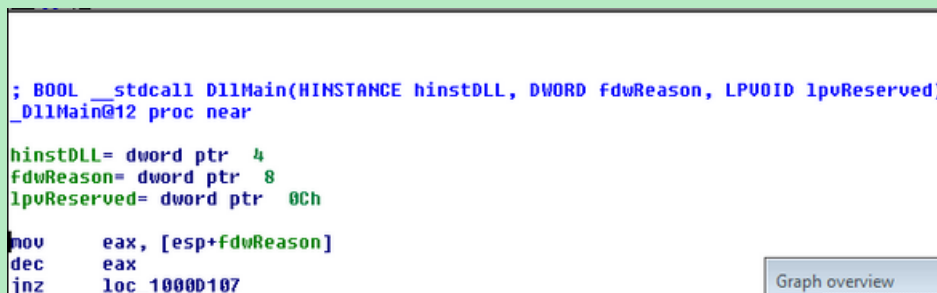


### Traccia:

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 FUNZIONE DLLMAIN



```
; 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
```

La funzione `DllMain` è una funzione standard in una DLL di Windows, che viene chiamata dal sistema operativo quando la DLL viene caricata o scaricata. I parametri utilizzati sono:

hinstDLL: L'handle dell'istanza della DLL .

fdwReason: La ragione per la quale `DllMain` è stata chiamata.

lpvReserved: Un puntatore riservato.

**mov eax, [esp+fdwReason] :**

Questa istruzione carica nel registro eax il valore di fdwreason che è stato passato alla funzione tramite lo stack.

**dec eax:**

Questo decrementa il valore di eax di 1.

**jnz loc\_1000D107:**

jnz` significa "jump if not zero", ovvero "salta se non è zero". Se eax non è zero, allora il codice salta all'indirizzo `loc\_1000D107`.

- Se eax invece è zero il codice continua con l'istruzione successiva.

**Indirizzo della funzione DllMain:**

L'indirizzo esadecimale all'inizio di questa funzione è 0x1000D020 Questo è l'indirizzo della funzione `DllMain`.



# 2 FUNZIONE GETHOSTBYNAME

```
.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 ...
```

L'indirizzo dell'import della funzione gethostbyname è 0x100163C:

- gethostbyname è una funzione della libreria Winsock utilizzata per risolvere un nome di host in un indirizzo IP.
- restituisce un puntatore a una struttura hostent che contiene informazioni sull'host, inclusi i suoi indirizzi IP.
- questa funzione è viene spesso utilizzata dai malware per contattare server remoti, spesso server di comando e controllo, utilizzando nomi di dominio invece di indirizzi IP statici.
- il codice **XREFs** indica che questa funzione è chiamata in diverse parti del codice del malware
- suggerisce che il malware potrebbe effettuare risoluzioni DNS dinamiche, probabilmente per connettersi a diversi server, forse come parte di una strategia di persistenza o per eludere il rilevamento.



# 3 0X10001656

```
.text:10001656 ; ===== S U B R O U T I N E =====
.text:10001656 |
.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 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 -48Ch
.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
.text:10001656 sub     esp, 678h
```



la funzione utilizza 20 variabili locali

- 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
- Source: byte ptr -63Dh
- Data: byte ptr -638h
- var\_637: byte ptr -637h
- var\_544: dword ptr -544h
- var\_50C: dword ptr -50Ch
- var\_500: dword ptr -500h
- Buf2: byte ptr -4FCh
- readfds: fd\_set ptr -4BCh
- phkResult: byte ptr -3B8h
- var\_3B0: dword ptr -3B0h

funzione appare centrale per il comportamento del malware, probabilmente gestendo operazioni di rete e altri compiti importanti.



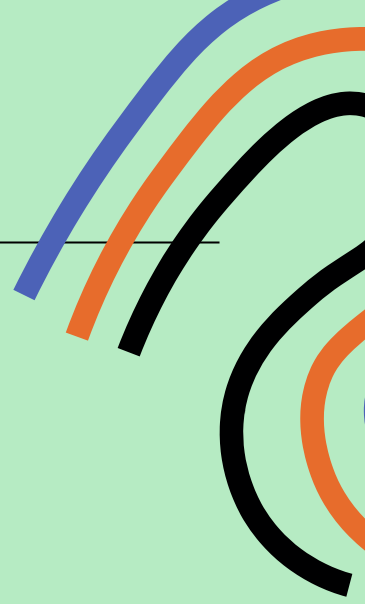
# 4

```
.text:10001656 ; SUBROUTINE
.text:10001656 |
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)
.text:10001656 sub_10001656 proc near ; DATA XREF: DllMain(x,x,x)+C8j0
.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_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
.text:10001656 sub esp, 678h
```

1 solo parametro che è arg\_0



# 5



Il malware in questione sembra essere un trojan o una backdoor progettata per comunicare con un server remoto, potenzialmente per esfiltrare dati o ricevere comandi.

La capacità di connettersi a un server remoto suggerisce che il malware potrebbe essere in grado di ricevere aggiornamenti, nuovi comandi, o addirittura scaricare e installare moduli aggiuntivi per estendere le sue funzionalità.

## Conclusioni

