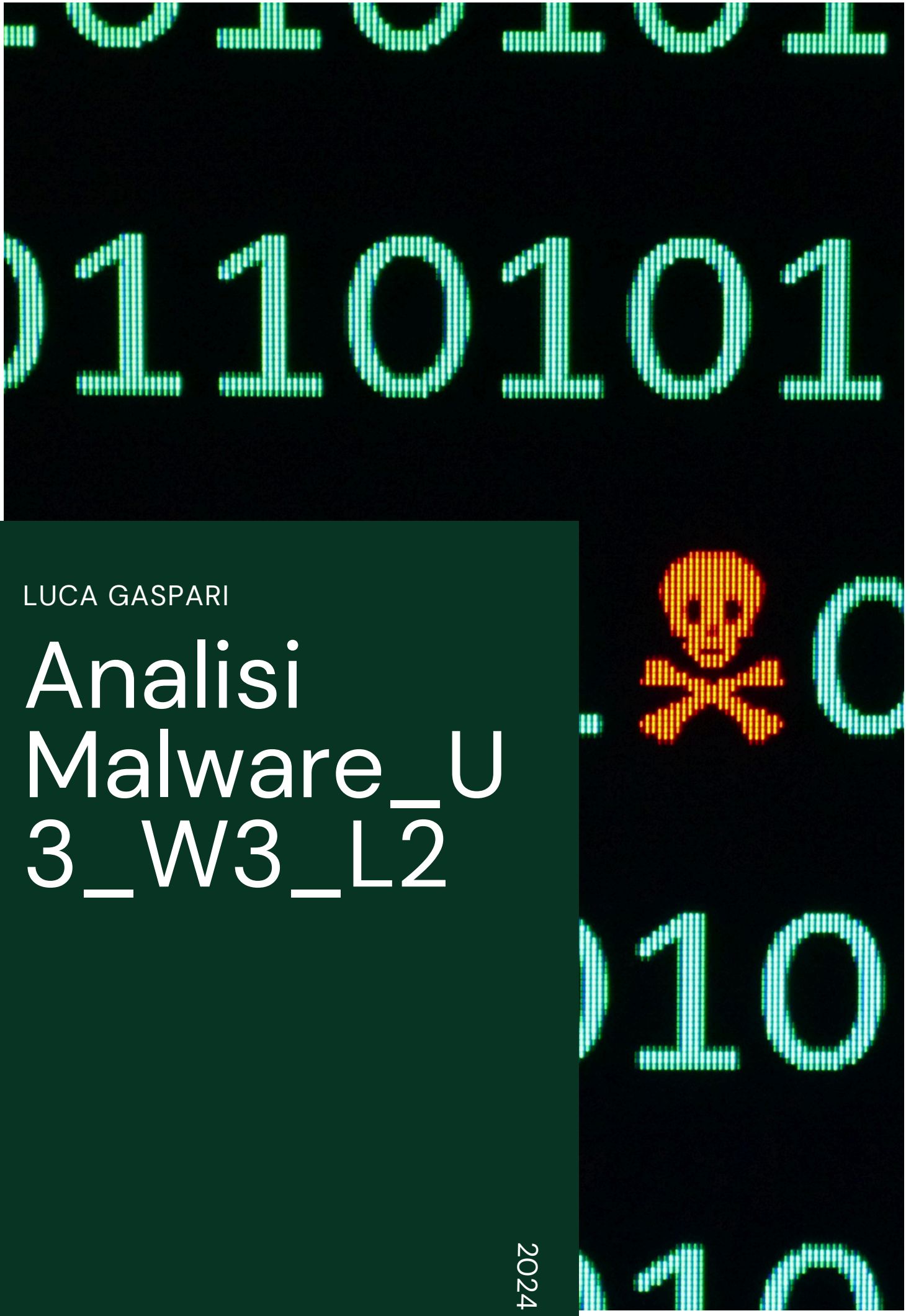
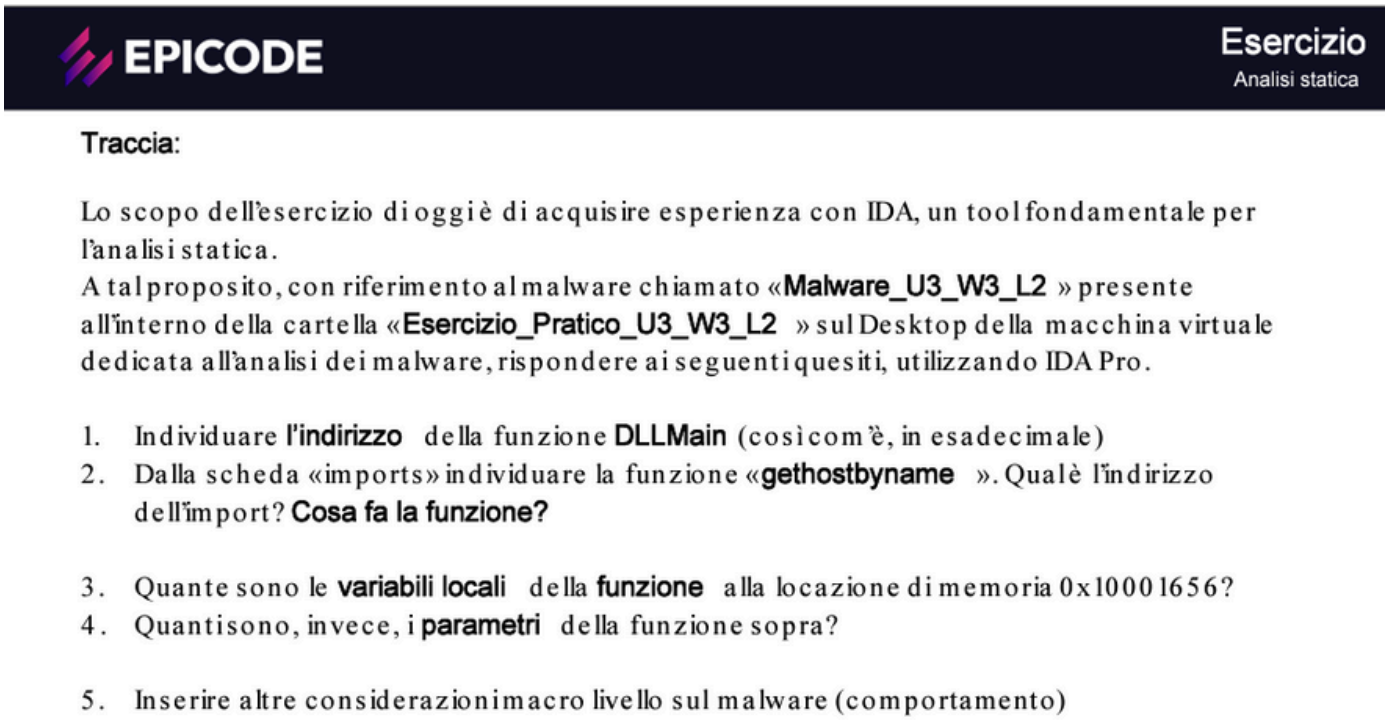


LUCA GASPARI

Analisi Malware_U3_W3_L2

2024

Traccia esercizio S11L2



EPICODE

Esercizio
Analisi statica

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)

Introduzione

Lo scopo dell'esercizio di oggi è acquisire esperienza con **IDA**, un tool fondamentale per l'analisi statica. Il compito riguarda l'analisi di un malware chiamato **Malware_U3_W3_L2** presente nella cartella **Esercizio_Pratico_U3_W3_L2** sul Desktop della macchina virtuale dedicata all'analisi dei malware.

1

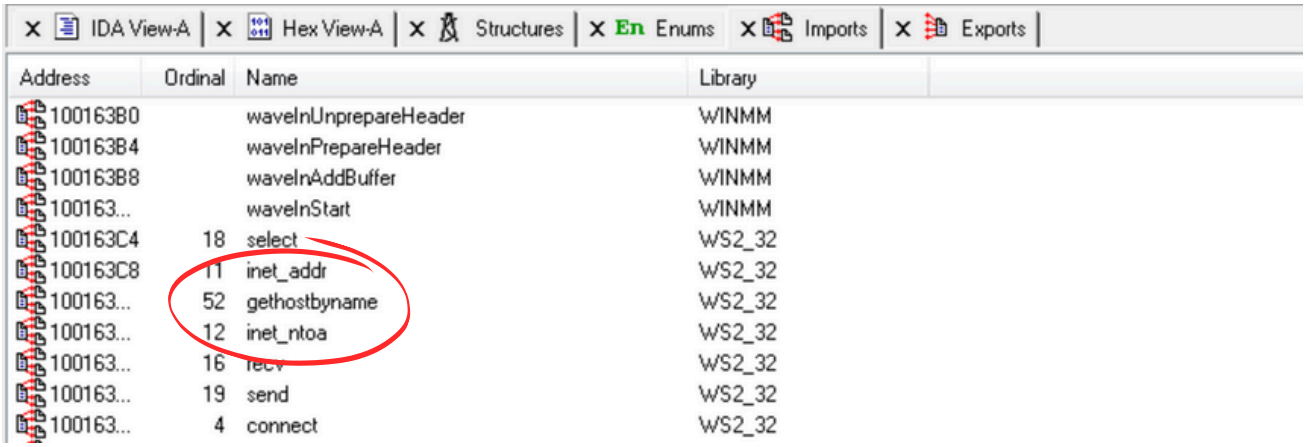
Per trovare l'indirizzo della funzione **DllMain**, inizio con caricare l'eseguibile in IDA pro.
Dopo averlo fatto, premo la barra spaziatrice per passare alla visualizzazione testuale e recupero l'indirizzo della funzione main cioè: **1000D02E**

```
.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
.text:1000D02E hinstDLL        = dword ptr  4
.text:1000D02E fdwReason       = dword ptr  8
.text:1000D02E lpvReserved     = dword ptr  0Ch
```

La funzione **DLLMain** è il punto di ingresso standard per le librerie di collegamento dinamico (DLL).

2

Per trovare l'indirizzo della funzione **gethostbyname**, ho selezionato la finestra degli **import** nell'interfaccia principale di IDA pro in questa finestra si possono trovare tutte le funzioni importate dal malware, successivamente ho individuato nell'elenco la funzione e il suo relativo indirizzo cioè: **0x100163C0**




```
.idata:100163CC ; struct hostent *__stdcall gethostbyname(const char *name)
.idata:100163CC                 extrn gethostbyname:dword
```

La funzione **gethostbyname** è una delle funzioni della libreria di Windows Sockets (Winsock) ed è utilizzata per la risoluzione dei nomi di host in indirizzi IP. Questa funzione è fondamentale nei malware che necessitano di convertire un nome di dominio (ad esempio, www.example.com) in un indirizzo IP (ad esempio, 192.168.1.1).

3 Le variabili locali che si trovano alla locazione di memoria **0x10001656 sono 23**

```
.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
.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
00000A56   10001656: sub_10001656

.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
00000A56   10001656: sub_10001656

.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
```

Le variabili locali possono essere identificate nella visualizzazione testuale del codice assembly, in questa modalità, le variabili locali sono solitamente indicate con nomi come **var_xxx**, dove **xxx** è un offset dal puntatore di base della funzione.

Quindi quando l'offset è negativo, indica che le variabili sono locali e sono allocate sullo stack.

4 C’è solo un argomento che è stato passato alla funzione cioè **arg_0,** questo è l’unico parametro con offset positivo rispetto ad ebp.

5 ## Considerazioni sul comportamento del malware

Comunicazioni di Rete: Il malware verifica se le stringhe iniziano con "**http://**" o "**ftp://**", indicando che potrebbe essere progettato per interagire con server web o FTP, questo comportamento suggerisce che il malware potrebbe essere utilizzato per trasferire dati o ricevere comandi da un server remoto

Creazione di Thread: Utilizza **CreateThread** per eseguire codice in parallelo, suggerendo che può compiere operazioni multiple simultaneamente, come connessioni di rete o raccolta di dati.

La copia di stringhe e la verifica di prefissi URL suggeriscono che il malware può essere configurato per comunicare con server specifici, questo indica un certo livello di modularità e adattabilità del malware a diverse situazioni o obiettivi.

Il malware analizzato sembra essere progettato per comunicare con server remoti utilizzando protocolli HTTP e FTP, la capacità di creare thread indica che può gestire più operazioni contemporaneamente, rendendolo più efficiente e difficile da rilevare.

Il suo comportamento suggerisce che è stato progettato per essere modulare e adattabile, facilitando la sua configurazione per vari scopi malevoli.