

REPORT

2024



PHANTOM s.r.l
**IMPOSSIBLE IS
OUR TARGET**

PREPARED BY
PHANTOM SRL

INDICE

Page 3: Traccia

Page 4: Funzione DLL Main

Page 5: GetHostByName

Page 6: Variabili Locali

Page 7: Parametri Presenti

TRACCIA

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)



DLL MAIN

```
.text:1000D02B
.text:1000D02E
.text:1000D02E ; ===== S U B R O U T I N E =====
.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
.text:1000D02E
.text:1000D02E      mov     eax, [esp+fdwReason]
.text:1000D032      dec     eax
.text:1000D033      jnz     loc_1000D107
.text:1000D039      mov     eax, [esp+hinstDLL]
0000C42B  000000001000D02B: ServiceMain+FB
```

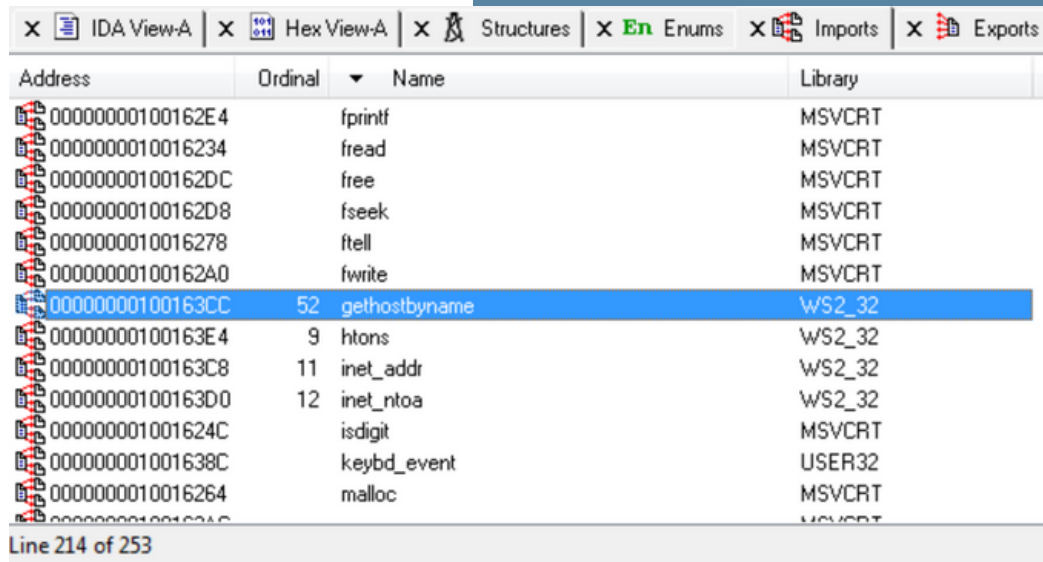
Per recuperare l'entry point
basta aprire il nostro
Malware tramite l'utilizzo
di Ida Pro (Spiegazione
accanto), un disassembler
già pre installato all'interno
della nostra macchina
Windows 7. Una volta
aperto, passiamo alla
modalità testuale per
recuperare il Dll Main
(1000D02E)

COS'È IDA PRO?

IDA Pro è uno strumento di
disassemblaggio e analisi del codice
sorgente ampiamente utilizzato
nell'ambito della sicurezza informatica
e della ricerca sulla sicurezza dei
software. Funziona in modo simile a un
debugger, ma va oltre nel fornire una
vasta gamma di funzionalità per
analizzare il codice binario.

IDA Pro analizza il file binario e lo
traduce in linguaggio assembly leggibile
dall'uomo.

GETHOSTBYNAME



Address	Ordinal	Name	Library
00000000100162E4		fprintf	MSVCRT
0000000010016234		fread	MSVCRT
00000000100162DC		free	MSVCRT
00000000100162D8		fseek	MSVCRT
0000000010016278		ftell	MSVCRT
00000000100162A0		fwrite	MSVCRT
00000000100163CC	52	gethostbyname	WS2_32
00000000100163E4	9	htons	WS2_32
00000000100163C8	11	inet_addr	WS2_32
00000000100163D0	12	inet_ntoa	WS2_32
000000001001624C		isdigit	MSVCRT
000000001001638C		keybd_event	USER32
0000000010016264		malloc	MSVCRT

Line 214 of 253

Per scoprire l'indirizzo dell'import gethostbyname basta aprire la finestra degli "imports" su ida pro. come vediamo gethostbyname è all'indirizzo 100163cc, come vediamo in figura.



VARIABILI LOCALI

```
.text:10001651
.text:10001656 ; ===== SUBROUTINE =====
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)
.text:10001656 sub_10001656 proc near ; DATA XREF: DllMain(x,x,x)+C840
.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 -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
.text:10001656
* .text:10001656 sub esp, 678h
```

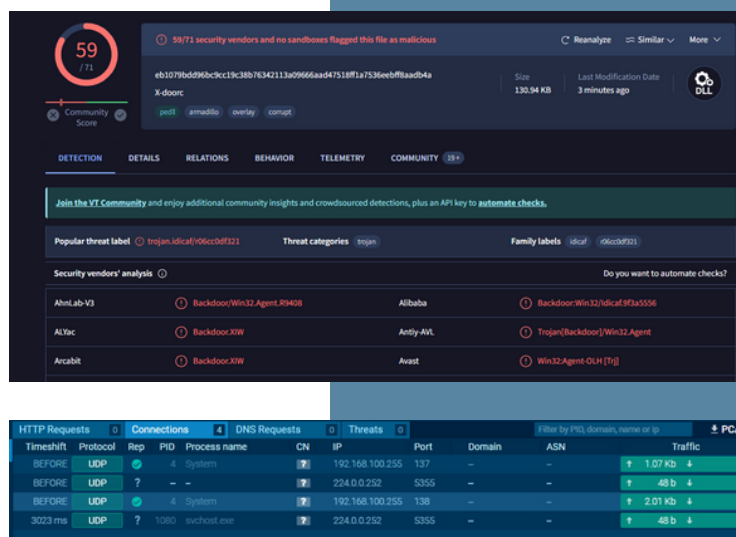
le variabili locali della funzione alla locazione di memoria 10001656 sono in totale 20 (qui nella foto se ne vedono 33, ma sappiamo che le variabili vengono considerate locali quando hanno offset negativo rispetto all'ebp).

Per quanto riguarda invece i parametri trovati, notiamo un solo argomento con offset positivo rispetto all' EBP.

IDA ha chiamato questo parametro arg_0.

ASSEMBLY

COMPORTAMENTO DEL MALWARE



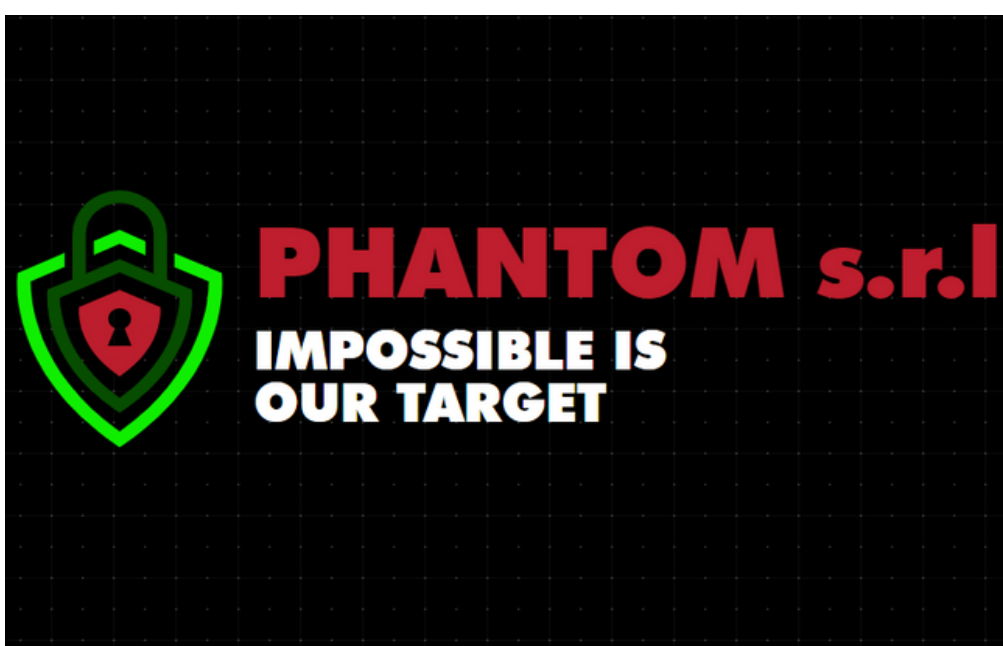
Il compito coinvolge l'esecuzione di un file DLL malevolo utilizzando il processo Windows legittimo "rundll32.exe" da una cartella temporanea nella directory del profilo utente. Il processo genitore per questa esecuzione è stato identificato come un processo con PID 1164.

Uno dei punti più interessanti in questo compito è l'uso del processo rundll32.exe per caricare ed eseguire il file DLL malevolo. Questa tecnica è comunemente utilizzata dai malware per evitare la rilevazione poiché rundll32.exe è un processo Windows legittimo. Inoltre, il fatto che il file DLL fosse situato nella cartella temporanea dell'utente indica un tentativo di mimetizzarsi con l'attività utente normale e evitare la rilevazione.

In conclusione, il compito coinvolgeva l'esecuzione di un file DLL potenzialmente malevolo utilizzando un processo Windows legittimo, indicando un possibile tentativo di evitare la rilevazione. Gli analisti di malware dovrebbero prestare particolare attenzione a tali tecniche che coinvolgono l'albero dei processi e posizioni di file sospette per identificare ed attenuare efficacemente le potenziali minacce.

GRAZIE

PHANTOM SRL



PREPARED BY

PIGNATELLO GIUSEPPE
D'OTTAVIO ALESSIO
IANNONE LUCA