ANALISI STATICA AVANZATA CON IDA

S11/L2

ELEONORA VIOLA

TABLE OF CONTENT

INTRODUCTION

Traccia:

Lo scopo dell'esercizio di oggi è di acquisire esperienza con IDA, un tool fondamentale per l'a n a lis i s t a t ic a .
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?

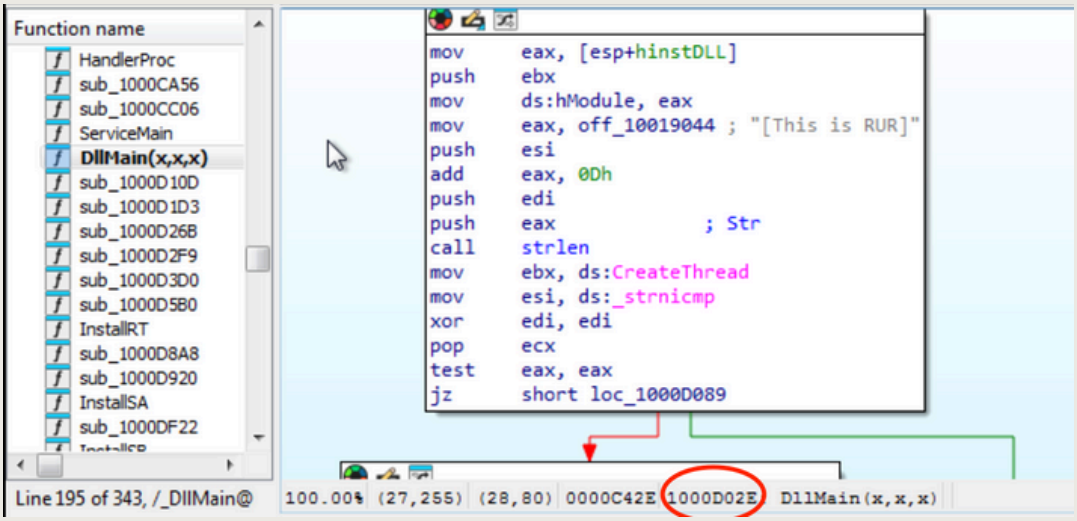
4. Quanti sono, invece, i parametri della funzione sopra?

5. Inserire altre considerazioni macro livello sul malware (comportamento)

PROCEDURE

1. **DLLMAIN:**

La funzione DllMain è una funzione speciale nelle Dynamic Link Library (DLL) in ambienti Windows che viene chiamata automaticamente quando una DLL viene caricata o scaricata da un processo, o quando si verificano determinati eventi legati alla DLL. Questa funzione può essere utilizzata per l'inizializzazione e la pulizia delle risorse della DDL, per l'esecuzione di operazioni specifiche del thread e altro ancora, a seconda del motivo per cui viene chiamata. Questo fornisce un punto di ingresso per l'inizializzazione e la pulizia della DLL.



Nel contesto dell'analisi statica avanzata di un file PE con IDA Pro, l'identificazione della funzione **DllMain** all'indirizzo **0x1000D02E** potrebbe essere un punto chiave per comprendere il comportamento del malware in quanto rappresenta il punto di ingresso esatto durante il caricamento della DLL e costituisce il luogo in cui il malware avvia le sue attività iniziali.

Da qui, il malware può propagarsi, eseguire codice dannoso e intraprendere altre azioni dannose L'analisi di questa istruzione fornisce un quadro iniziale fondamentale delle attività del malware, consentendo agli analisti di identificare le minacce e sviluppare strategie di mitigazione adeguate.

PROCEDURE

2. **gethostbyname**

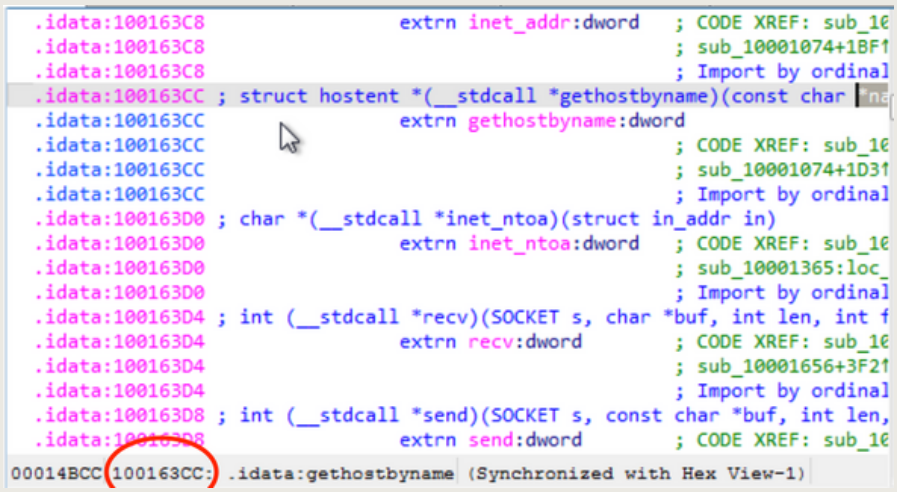
Con Ida Pro è stato possibile individuare la funzione "gethostbyname", componente fondamentale della Libreria importata WS2_32, all'indirizzo di Import 0x100163CC.

La funzione gethostbyname è una funzione di libreria standard di Windows inclusa nella libreria WS2_32 (Winsock 2).

Questa funzione è comunemente utilizzata per risolvere il nome host in un indirizzo IP
WS2_32 è la libreria di Winsock 2, che fornisce API per la programmazione di socket su piattaforme Windows.
È ampiamente utilizzata per implementare la comunicazione di rete e, nel contesto del malware, l'utilizzo di WS2_32 suggerisce che il codice del malware sta coinvolto in operazioni di rete avanzate, quali connessioni di rete, invio o ricezione di dati attraverso socket.

L'uso della funzione **gethostbyname** in un malware suggerisce la possibilità che il programma stia cercando di stabilire una connessione con un server remoto. Questo comportamento è tipico dei malware che comunicano con un server di comando e controllo (C2), un'infrastruttura centrale che guida le azioni del malware. Attraverso questa connessione, il malware potrebbe ricevere istruzioni da operatori malintenzionati, inviare dati compromettenti o persino scaricare componenti aggiuntivi.

Un aspetto notevole è che l'uso di **gethostbyname** potrebbe essere una tattica per nascondere l'indirizzo IP del server remoto. Invece di codificare staticamente l'indirizzo IP nel codice, il malware risolve dinamicamente il nome host durante l'esecuzione. Questa strategia rende più difficile per i ricercatori di sicurezza e gli strumenti di analisi tracciare il server remoto, poiché l'indirizzo IP potrebbe variare dinamicamente nel tempo o essere distribuito su diversi nomi host. In sintesi, l'utilizzo di **gethostbyname** in un contesto malware suggerisce che il programma sta cercando di stabilire una connessione con un server remoto e potrebbe adottare misure per complicare la tracciabilità dell'indirizzo IP del server.



3 e 4. Variabili e parametri

Procediamo dunque a cercare la funzione della locazione di memoria richiesta.

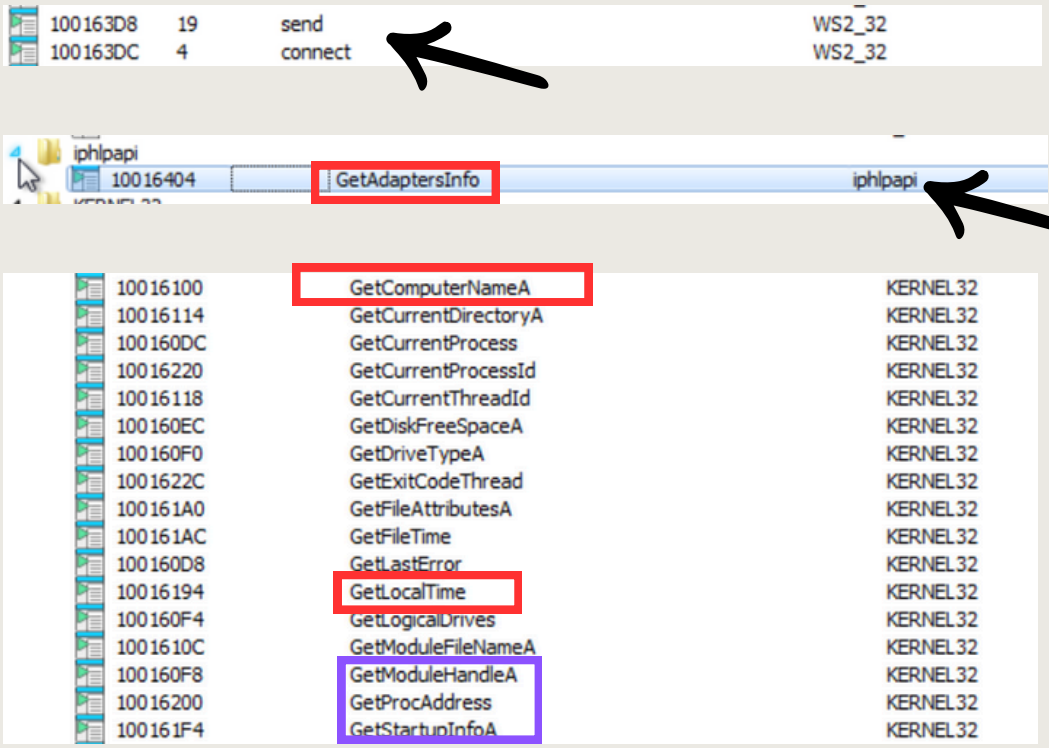
Possiamo trovare 23 variabili e 1 parametro. Se il valore dell'offset è negativo, rispetto al registro EBP sono delle variabili

```
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
var_1A4= dword ptr -1A4h
var_194= dword ptr -194h
WSAData= WSAData ptr -190h
```

Parametro

```
lpThreadParameter= dword ptr  4
```

5. Considerazioni macro livello sul malware:



1. **UTILIZZO DI API DI SISTEMA:**
   - il malware fa ampio uso di API di sistema, come quelle fornite da **KERNEL32** (per gestione di file e processi), **IPHLPAPI** (per informazioni di rete), e **WS2_32** (per comunicazioni di rete). Questo suggerisce che il malware ha bisogno di interagire profondamente con il sistema operativo per eseguire le sue operazioni, indicando un comportamento potenzialmente invasivo o dannoso.
2. **FUNZIONI DI RETE:**
   - le funzioni **send** e **connect** utilizzate da **WS2_32** indicano che il malware potrebbe comunicare con server remoti. Questo comportamento è tipico di malware che esfiltrano dati o ricevono comandi da un server di comando e controllo (C2).
3. **RACCOLTA DI INFORMAZIONI DI SISTEMA:**
   - funzioni come **GetComputerNameA**, **GetAdaptersInfo**, e **GetLocalTime** indicano che il malware raccoglie informazioni sul sistema su cui è in esecuzione. Queste informazioni potrebbero essere utilizzate per profilare il sistema vittima, aiutando il malware a determinare il miglior metodo di attacco o a evitare macchine non interessanti.
4. **PERSISTENZA E EVASIONE:**
   - l'uso di funzioni come **GetModuleHandleA**, **GetProcAddress**, e **GetStartupInfoA** suggerisce che il malware potrebbe avere meccanismi per mantenere la persistenza nel sistema o per evadere il rilevamento. Queste funzioni sono spesso utilizzate per caricare dinamicamente moduli o per modificare il comportamento del processo corrente senza essere rilevati.
5. **POTENZIALI AZIONI MALIGNE:**
   - il malware potrebbe eseguire operazioni dannose come la modifica di file di sistema, la creazione di processi nascosti, o l'utilizzo delle risorse di rete per attività illecite. La combinazione delle funzioni mostrate suggerisce un comportamento complesso, potenzialmente mirato a ottenere accesso non autorizzato, raccogliere informazioni sensibili e mantenere un accesso persistente al sistema compromesso.

Queste considerazioni indicano un malware che è progettato per avere un impatto significativo sul sistema vittima, con capacità di raccolta di informazioni, comunicazione remota, e potenzialmente meccanismi di evasione del rilevamento.