S11-L5

Traccia

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

- Spiegate, motivando, quale salto condizionale effettua il Malware.
- 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.
- Quali sono le diverse funzionalità implementate all'interno del Malware? Esercizio Traccia e requisiti
- 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.

Salti condizionali

I salti condizionali sono un costrutto fondamentale della programmazione in assembly.
Essi sono dei meccanismi implementati attraverso istruzioni finalizzate ad effettuare dei “salti” in base allo stato del processore o a specifiche condizioni.

Le istruzioni dei salti condizionali si basano, generalmente, sui valori contenuti nei “flag” della CPU ottenuti come output da operazioni effettuate precedentemente.
Spesso l’istruzione che precede un salto condizionale è quella di confronto CMP, seguita, ad esempio, da un Jz (jump if zero) o Jnz (Jump if not zero).

Grazie ai salti condizionali possiamo creare cicli e strutture di controllo rappresentanti lo scheletro logico computazionale permettenti al programma di reagire dinamicamente in base alle varie situazioni che affronta, adattando il codice in “run-time”.

Prendendo in esame il codice fornito, analizziamo cosa accade giunti alla prima istruzione condizionale.

- Copia il valore 5 nel registro “EAX”
- Confronta due valori uguali, (EAX, contenente 5, e 5) viene utilizzato l’operatore “sottrazione”, il risultato è 0
- Lo 0 ottenuto dal cmp fa assumere allo ZF (Zero Flag) il valore di 1.
- L’istruzione jnz salta solo quando lo Zero Flag ha valore 0, in questo caso, quindi, non salta.

Salti condizionali

Prendendo in esame il codice fornito, analizziamo cosa accade giunti alla seconda istruzione condizionale.

- Nel registro EBX avevamo storato 10
- inc va ad incrementare il valore ad 11
- cmp confronta due valori uguali, (EBX, contenente 11, e 11) il risultato è 0.
- Lo 0 ottenuto dal cmp fa assumere allo ZF (Zero Flag) il valore di 1.
- Essendo jz ed essendo il flag 1, il jump avviene.

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

Il primo salto non avviene, il secondo, invece, si.

Diagramma di flusso (IDA)

Locazione	Istruzione	Operandi
00401040	mov	EAX, 5
00401044	mov	EBX, 10
00401048	cmp	EAX, 5
0040105B	jnz	loc 0040BBA0

0040105F	inc	EBX
00401064	cmp	EBX, 11
00401068	jz	loc 0040FFA0

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

Funzioni e passaggio di argomenti

Passiamo, ora, ad analizzare le funzioni.
Le due funzioni che analizzeremo sono quelle invocate con l'istruzione call.

DownloadToFile

Dall'analisi del pezzo di codice ove ritroviamo la suddetta funzione possiamo dedurre come stiamo trattando di un downloader che andrà a scaricare un file tramite il link presente nella stringa URL "ww.malwaredownload.com" inserita nel registro eax.

Nello specifico vediamo come:

- mov eax, edi -- copia il contenuto del registro edi in eax (la stringa)
- push eax -- inserisce il valore del registro eax nello stack, ci stiamo preparando per passare il contenuto come argomento della funzione DownloadToFile
- call DownloadToFile -- passa l'URL contenuto nello stack come argomento della funzione (link ove scaricherà il file)

WinExec

Questa funzione, sfruttando le API di Windows, consente al malware di eseguire un programma/file precedentemente scaricato sul sistema compromesso.

Da questo, possiamo dedurre come il malware sia progettato per compiere azioni post download del file malevolo sul sistema.

Nello specifico, vediamo come:

- mov edx edi -- copia il contenuto del registro edi nel registro edx (il path che servirà nella chiamata WinExec)
- push edx -- carica il valore del registro edx nello stack, ci stiamo preparando per passare il contenuto come argomento della funzione WinExec
- call WinExec -- utilizza il path appena fornitogli per avviare il processo target.

Ipotesi di funzionamento

Quello che stiamo analizzando, probabilmente, è un downloader avente come fine ultimo quello di scaricare ed installare ulteriori componenti e risorse esterne da un server remoto senza consenso dell'utente.

Per quanto possa valere, analizzando il path di esecuzione del file notiamo come il nome del file scaricato, e pronto per essere avviato, sia "Ransomware.exe".
Ammesso che il nome rispecchi la funzione dell'esecutivo, il programma lancerà un ransomware finalizzato a limitare l'accesso al sistema (criptando i file o bloccando completamente l'uso del dispositivo), per riportare (probabilmente) il sistema alla normalità una volta ricevuto il pagamento del riscatto.

Analisi di Tcpvcon.exe

Proseguiamo all'analisi del codice andando a spiegare le porzioni più ostiche:

- ***push offset aTcpview***, prima di invocare la funzione mediante l'istruzione call, bisogna caricarla nello stack, questo avviene tramite l'istruzione push. Lo strumento TCPView serve per monitorare in real-time le connessioni di rete, sia TCP che UDP. Ci fornisce una visione dettagliata delle porte locali, remote, e degli indirizzi IP. Una volta caricato nello stack verra invocato tramite l'istruzione ***call sub_420CE0***.

- ***WSAStartup***, questa funzione fa parte dell'API WinSock, la sua utilità è quella di inizializzare l'uso di Winsock in una data applicazione.

- ***push offset stru_42BC20***, questa funzione serve a caricare l'indirizzo stru_42BC20 nello stack, preparandolo come argomento per la successiva chiamata di funzione (call ds: InitializeCriticalSection).

- ***call ds: InitializeCriticalSection***, la funzione invocata serve per inizializzare una "sezione critica" (tale funzione rientra sempre nelle API di Windows). Il suo compito è quello di proteggere porzioni di codice che devono essere eseguite esclusivamente un thread alla volta. Deve essere anticipato dal comando sopra riportato in quanto l'istruzione push offset stru_42BC20 fornirà il puntatore alla struttura critica necessario per inizializzare correttamente la sessione critica stessa.

Analisi di Tcpvcon.exe

- ***push offset aSedebugprivile***, la spiegazione è analoga alla precedente dal punto di vista logico. Circa, invece, “aSedebugprivile”, possiamo vedere come il prefisso “a” viene usato da IDA Pro per indicare un’array di caratteri. Potrebbe essere una stringa descrivente un privilegio di sicurezza.

- ***push offset aCouldNotiInitia***, grazie anche al commento possiamo capire come sembrerebbe essere un messaggio di errore utilizzato per informare l’utente. L’errore riguarda l’inizializzazione della libreria Winsock (precedentemente spiegata). Il prefisso “a” ha lo stesso ruolo della precedente spiegazione.

Conclusione

Abbiamo analizzato le principali funzioni del programma, TCPView è uno strumento utile a visualizzare in tempo reale le connessioni attive sul sistema.

Le sue caratteristiche principali sono:

- Monitorare le connessioni di rete ed il loro stato;
- Rilevare connessioni sospette (utile per individuare attività di rete sospette).

Per ulteriori informazioni basta effettuare una ricerca;
dal momento che non sembra essere un malware è possibile trovare tutte le informazioni desiderate semplicemente (è stato creato da Sysinternals).