

ANALISI STATICÀ E DINAMICA DI UN MALWARE



CONTENT

3

Introduzione

4

Librerie Importate dal File Eseguibile

5

Sezioni

6

Analisi del Codice Assembly - costrutti noti

7

Comportamento

8

Commento

10

Conclusioni





INTRODUZIONE

L'analisi dei malware è una pratica essenziale per comprendere e mitigare le minacce informatiche. Essa si suddivide principalmente in due approcci: l'analisi statica e l'analisi dinamica. L'analisi statica consiste nell'esaminare il codice del malware senza eseguirlo, permettendo di capire la struttura e il funzionamento del codice malevolo. L'analisi dinamica, invece, prevede l'esecuzione del malware in un ambiente controllato per osservare il suo comportamento in tempo reale. In questa relazione eseguiremo l'analisi statica, verranno illustrate le librerie importate dal file eseguibile del malware, le sezioni che compongono il file eseguibile, e un'analisi dettagliata di un frammento di codice assembly. Verrà inoltre fornita un'ipotesi sul comportamento del malware basata sul codice analizzato.

LIBRERIE IMPORTATE DAL FILE ESEGUIBILE

Il malware analizzato importa le seguenti librerie:

Kernel32.dll è una libreria di sistema fondamentale per Windows, fornendo funzioni essenziali per la gestione di operazioni di basso livello come gestione della memoria, creazione e controllo di processi e thread, input/output di file e sincronizzazione.

Funzioni utilizzate:

- Sleep: Sospende l'esecuzione del thread corrente per un intervallo di tempo specificato.
- SetStdHandle: Imposta l'handle di un dispositivo standard (input, output, errore).
- GetStringTypeW: Determina le proprietà dei caratteri in una stringa wide.
- GetStringTypeA: Determina le proprietà dei caratteri in una stringa ANSI.

Wininet.dll fornisce funzioni per l'accesso a Internet e gestione di protocolli di rete come HTTP e FTP, permettendo alle applicazioni di interagire con risorse di rete.

Funzioni utilizzate:

- InternetOpenUrlA: Inizia una sessione di URL HTTP.
- InternetCloseHandle: Chiude un handle Internet.
- InternetReadFile: Legge dati da un handle Internet aperto.
- InternetGetConnectedState: Determina lo stato della connessione a Internet.

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC
WININET.dll	5	000065CC	00000000	00000000	00006664

SEZIONI

Il file eseguibile del malware è composto dalle seguenti sezioni:

- **.text**: Contiene il codice eseguibile. Questa è la sezione principale dove risiede il codice macchina che viene eseguito dal processore.
- **.rdata**: Sezione dei dati di sola lettura. Contiene dati che non vengono modificati durante l'esecuzione del programma, come stringhe costanti e tabelle di importazione/esportazione.
- **.data**: Sezione dei dati di lettura/scrittura. Include variabili globali e statiche che possono essere modificate durante l'esecuzione del programma.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000
.data	00003F08	00007000	00003000	00007000	00000000	00000000

ANALISI DEL CODICE ASSEMBLY - COSTRUTTI NOTI

```
push    ebp
mov     ebp, esp
push    ecx
push    0          ; dwReserved
push    0          ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add    esp, 4
mov    eax, 1
jmp    short loc_40103A
```

```
loc_40102B:           ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add    esp, 4
xor    eax, eax
```

```
loc_40103A:
mov    esp, ebp
pop    ebp
ret
sub_401000 endp
```

Creazione dello Stack:

push ebp

mov ebp, esp

if:

cmp [ebp+var_4], 0

jz short loc_40102B

Chiamate di Funzione:

call ds:InternetGetConnectedState

call sub_40117F

COMPORTAMENTO

Il codice assembly presente nell'immagine implementa una funzionalità che verifica la connessione a Internet e esegue diverse azioni in base al risultato:

1. Verifica della Connessione: La funzione `InternetGetConnectedState` è chiamata per controllare se il sistema è connesso a Internet. Il risultato della chiamata è salvato in una variabile locale [ebp+var_4].
2. Esecuzione Condizionale:
 - Se la variabile [ebp+var_4] è zero, significa che non c'è connessione a Internet, e il flusso del programma salta all'etichetta `loc_40102B` dove viene gestito l'errore.
 - Se invece la variabile [ebp+var_4] è diversa da zero, il flusso prosegue normalmente, indicando una connessione Internet attiva.
3. Messaggi di Log:
 - Se connesso a Internet, viene visualizzato il messaggio "Success: Internet Connection" e eax viene impostato a 1.
 - Se non connesso a Internet, viene visualizzato il messaggio "Error 1.1: No Internet" e eax viene impostato a 0.

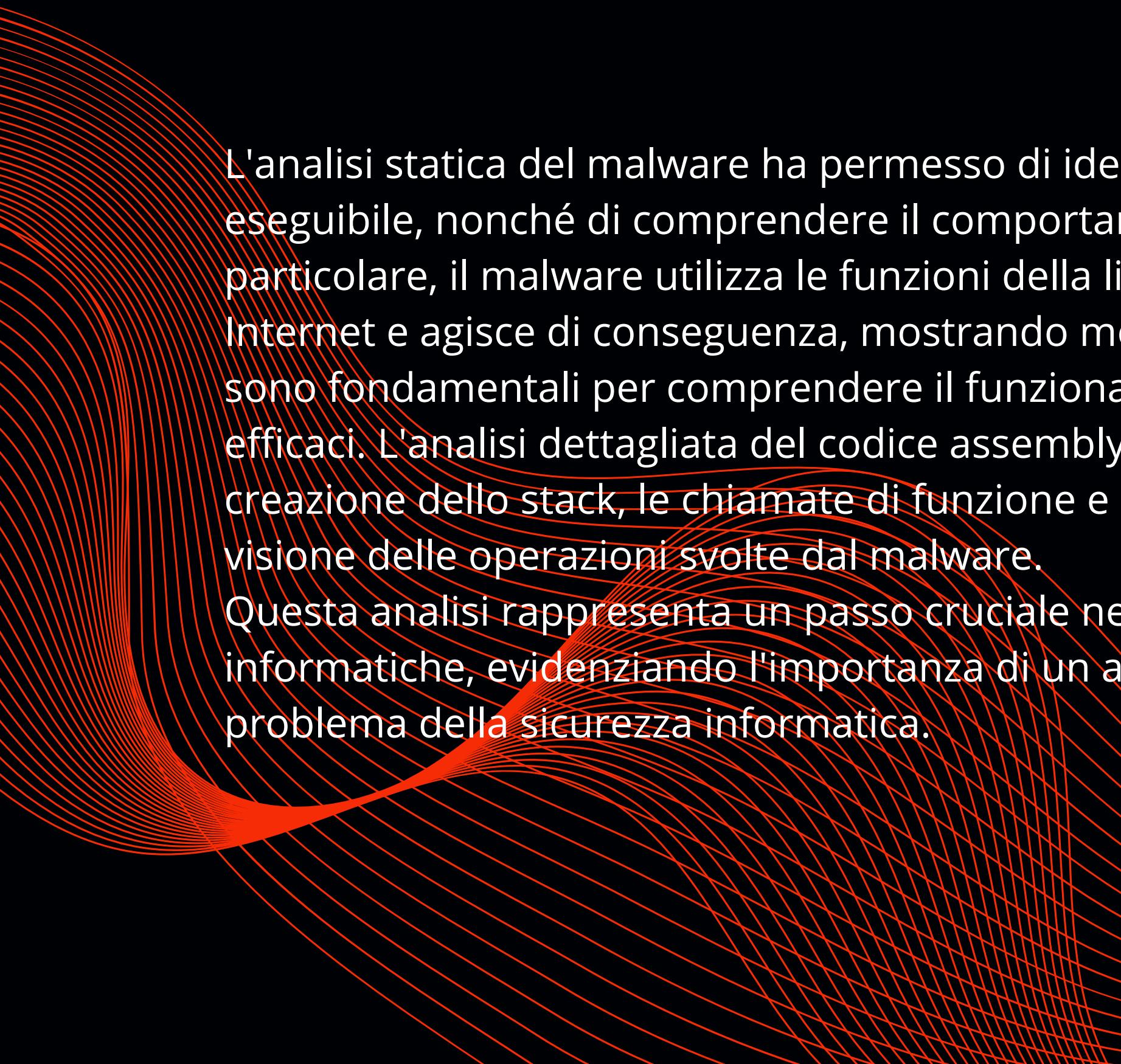
COMMENTO

push ebp	Salva il valore attuale del registro base pointer (EBP)	cmp [ebp+var_4], 0	Confronta il valore della variabile locale con 0
mov ebp, esp	Imposta il registro base pointer (EBP) allo stack pointer (ESP)	jz short loc_40102B	Se il valore è zero, salta all'etichetta loc_40102B
push ecx	Salva il valore del registro ECX sullo stack	push offset aSuccessInterne	Spinge l'offset del messaggio di successo sullo stack
push 0	Spinge il valore 0 (dwReserved) sullo stack	call sub_40117F	Chiama la funzione per visualizzare il messaggio
push 0	Spinge il valore 0 (lpdwFlags) sullo stack	add esp, 4	Pulisce lo stack rimuovendo il messaggio
call ds:InternetGetConnectedState	Chiama la funzione per verificare la connessione a Internet	mov eax, 1	Imposta il registro EAX a 1 (indicando successo)
mov [ebp+var_4], eax	Salva il risultato della chiamata di funzione in una variabile locale	jmp short loc_40103A	Salta all'etichetta loc_40103A

loc_40102B:	Etichetta per la gestione degli errori
push offset aError1NoInte	Spinge l'offset del messaggio di errore sullo stack
call sub_40117F	Chiama la funzione per visualizzare il messaggio di errore
add esp, 4	Pulisce lo stack rimuovendo il messaggio
xor eax, eax	Imposta il registro EAX a 0 (indicando fallimento)

loc_40103A:	Etichetta per l'epilogo della funzione
mov esp, ebp	Ripristina il valore dello stack pointer (ESP)
pop ebp	Ripristina il valore del base pointer (EBP)
retn	Ritorna dalla funzione

CONCLUSIONI



L'analisi statica del malware ha permesso di identificare le librerie importate e le sezioni del file eseguibile, nonché di comprendere il comportamento di un frammento di codice assembly. In particolare, il malware utilizza le funzioni della libreria wininet.dll per verificare la connessione a Internet e agisce di conseguenza, mostrando messaggi di successo o errore. Queste informazioni sono fondamentali per comprendere il funzionamento del malware e sviluppare strategie di difesa efficaci. L'analisi dettagliata del codice assembly ha rivelato l'uso di costrutti comuni come la creazione dello stack, le chiamate di funzione e le istruzioni condizionali, fornendo una chiara visione delle operazioni svolte dal malware.

Questa analisi rappresenta un passo cruciale nella comprensione e mitigazione delle minacce informatiche, evidenziando l'importanza di un approccio dettagliato e sistematico nell'affrontare il problema della sicurezza informatica.

