

ByteRebels

S10 - L5

Malware analysis



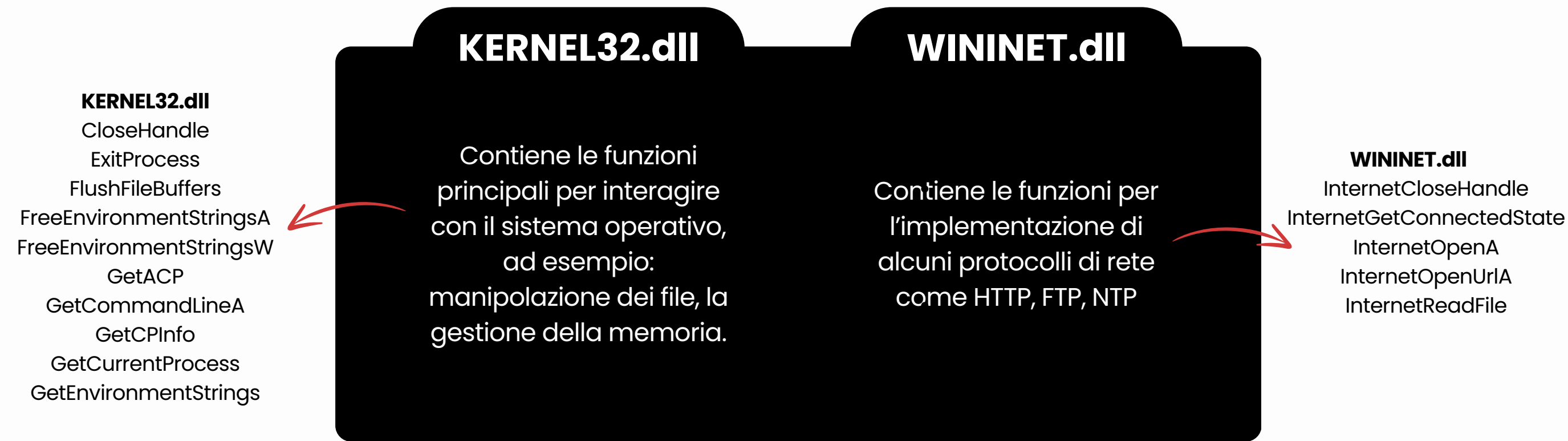
Traccia

Con riferimento al file **Malware_U3_W2_L5** presente all'interno della cartella «**Esercizio_Pratico_U3_W2_L5**» sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

1. Quali **librerie** vengono importate dal file eseguibile?
2. Quali sono le **sezioni** di cui si compone il file eseguibile del malware? Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:
3. Identificare i **costrutti noti** (creazione dello stack, eventuali cicli, altri costrutti)
4. Ipotizzare il **comportamento** della funzionalità implementata
5. **BONUS** fare tabella con significato delle singole righe di codice assembly



Quali librerie vengono importate dal file eseguibile?



La libreria KERNEL32.dll è **una libreria di collegamento dinamico** (DLL) fornita dal sistema operativo Windows. Essa **contiene numerose funzioni** e **procedure** che forniscono un'interfaccia per le funzionalità di base del sistema operativo.

La libreria WININET.dll è una libreria di collegamento dinamico (DLL) fornita dal sistema operativo Windows. Essa **fornisce** un'interfaccia per **l'accesso alle funzionalità di rete** e all'Internet Information Services (IIS) tramite il protocollo HTTP. La libreria WININET.dll offre una serie di funzioni per la gestione delle operazioni di rete, **come l'invio e la ricezione di richieste HTTP, il download e l'upload di file, la gestione dei cookie e la gestione delle connessioni di rete.**

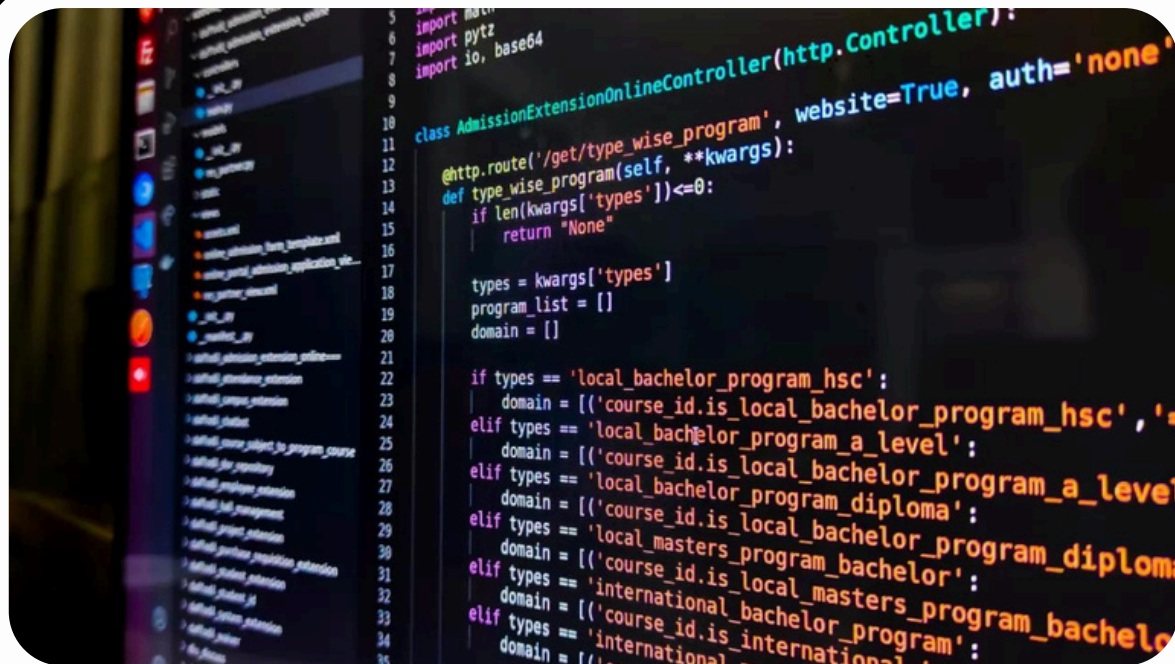
CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

Malware_U3_W2_L5.exe

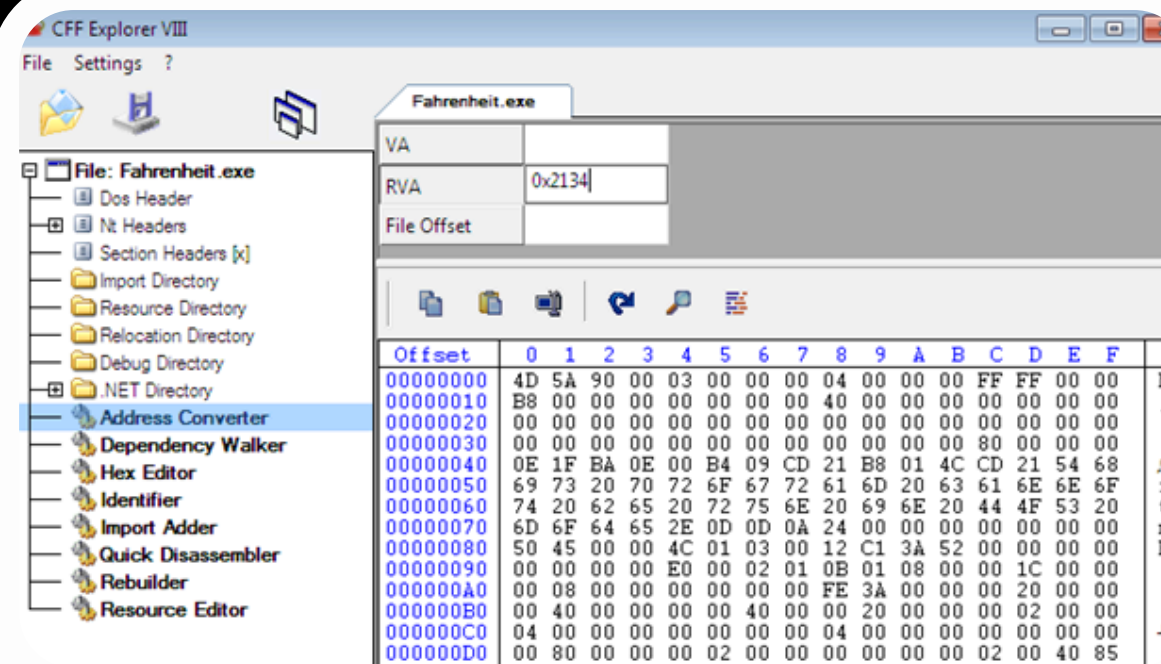
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
00006664	N/A	000064F0	000064F4	000064F8	000064FC	00006500
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

Quali sono le sezioni di cui si compone il file eseguibile del malware?



.text

Contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.



.rdata

Include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, informazione che possiamo ricavare con CFF Explorer.



.data

Contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Una variabile si dice globale quando non è definita all'interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione all'interno dell'eseguibile.

Malware_U3_W2_L5

Questo malware nell'esattezza è un **TROJAN**

Prevede l'esecuzione di un file denominato "Malware_U3_W2_L5.exe" situato nella cartella temporanea dell'utente, con il processo principale che è Windows Explorer (Explorer.EXE).

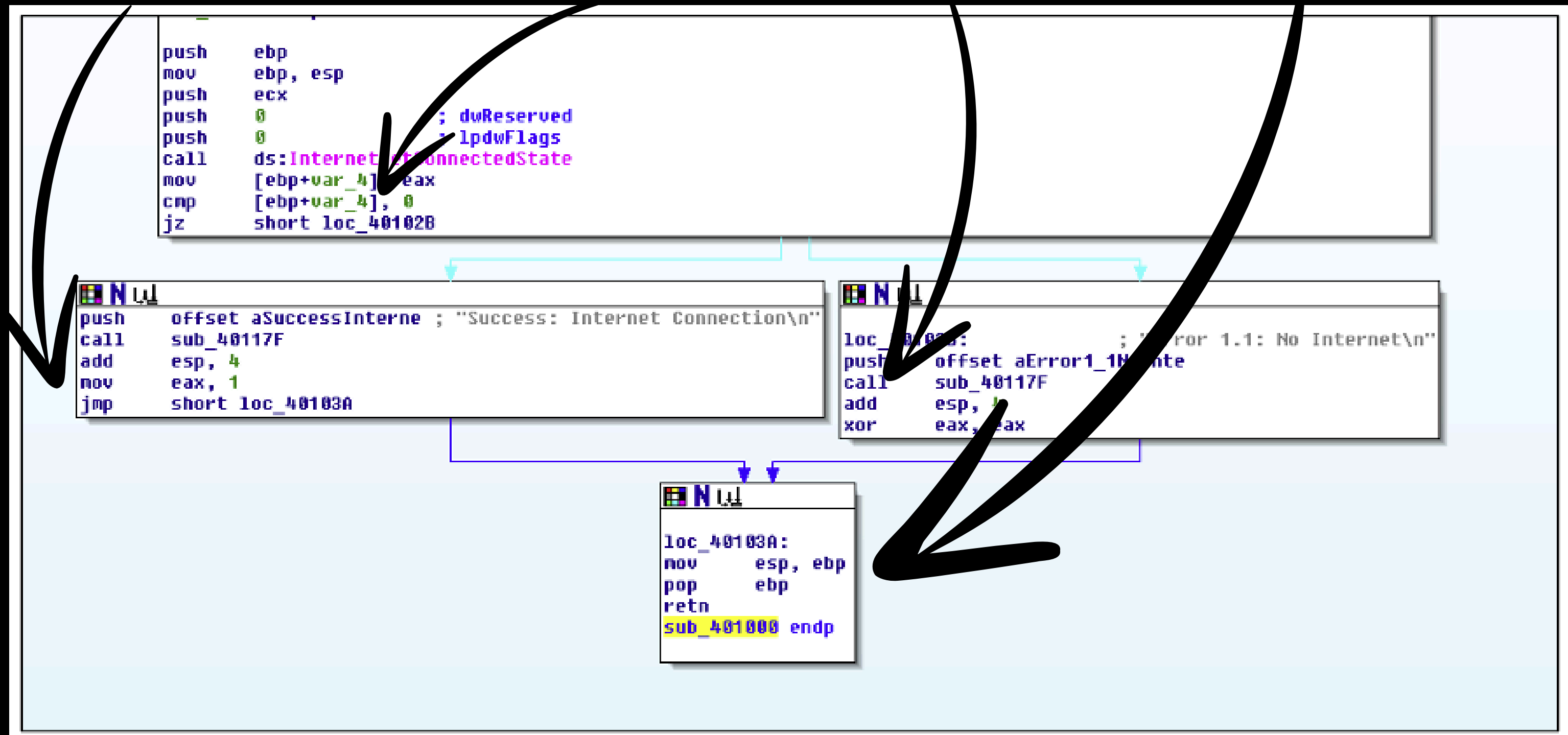
I programmi legittimi possono rilasciare file eseguibili in cartelle temporanee per vari motivi, come aggiornamenti software, installazione di plug-in o archiviazione di file temporanei durante l'esecuzione del programma. Il controllo delle lingue supportate, la lettura dei nomi dei computer, delle impostazioni Internet, delle informazioni sul server proxy e delle impostazioni di sicurezza sono azioni comuni eseguite da programmi legittimi per garantire la compatibilità e il corretto funzionamento.

In un contesto dannoso, come questo, **il rilascio di un file eseguibile in una cartella temporanea e l'avvio da parte di Esplora risorse può indicare un tentativo di eludere il rilevamento.** Il controllo delle informazioni di sistema come lingue supportate, nomi di computer, impostazioni Internet, informazioni sul server proxy e impostazioni di sicurezza può essere utilizzato dal malware per raccogliere informazioni sul sistema infetto, potenzialmente per ulteriori **attività dannose come l'esfiltrazione di dati sensibili o lo svolgimento di attacchi mirati.** La lettura del GUID della macchina dal registro può aiutare il malware a identificare in modo univoco il sistema infetto per monitorare o personalizzare il comportamento dannoso.

Identificare i costrutti noti

Si possono identificare costrutti, come ad esempio:

Istruzioni di salto condizionale, confronto tra due valori, chiamate di funzione, gestione dati stack (pop)



Costrutti noti

Abbiamo identificato una decina di costrutti noti, in dettaglio:

01

push: Viene utilizzato per spingere un valore nello stack. Ad esempio, `push ebp` e `push ecx` spingono i registri `ebp` e `ecx` nello stack.

02

mov: Viene utilizzato per copiare un valore da una locazione di memoria a un'altra. Ad esempio, `mov ebp, esp` copia il valore dello stack pointer `esp` nel registro `ebp`.

03

call: Viene utilizzato per chiamare una funzione o un'etichetta specifica. Ad esempio, `call ds:InternetConnecedState` chiama la funzione `InternetConnecedState`.

04

cmp: Viene utilizzato per confrontare due valori. Ad esempio, `cmp [ebp+var_4], 0` confronta il valore memorizzato all'indirizzo `[ebp+var_4]` con 0.

05

jz: È un'istruzione di salto condizionale che salta a un'etichetta specifica se l'ultima operazione di confronto ha dato come risultato zero. `jz short loc_40102B` salta a `loc_40102B` se il confronto precedente ha dato come risultato zero.

06

add: Viene utilizzato per aggiungere due valori. Ad esempio, `add esp, 4` aggiunge 4 al registro dello stack pointer `esp`.

07

xor: Viene utilizzato per eseguire l'operazione logica XOR tra due operandi. Ad esempio, `xor eax, eax` imposta il registro `eax` a zero.

08

jmp: È un'istruzione di salto che salta a un'etichetta specifica senza alcuna condizione. Ad esempio, `jmp short loc_40103A` salta a `loc_40103A`.

09

pop: Viene utilizzato per estrarre un valore dallo stack e memorizzarlo in un registro o in una locazione di memoria. Ad esempio, `pop ebp` estrae un valore dallo stack e lo memorizza nel registro `ebp`.



Ipotizzare il comportamento della funzionalità implementata

Il codice assembly della figura 1 sembra implementare una logica di gestione degli errori per una connessione a Internet. Leggendo queste righe si deduce che potrebbe anche **non** trattarsi di un malware, dipende dal contesto.

Comportamento

La chiamata alla funzione **InternetConnecedState** suggerisce che il malware potrebbe **cercare di stabilire una connessione a Internet**, questo potrebbe essere utilizzato per scopi come il **trasferimento di dati** da o verso un server remoto controllato dal malware. Il codice include una logica per gestire il caso di errore, **potrebbe utilizzare questa logica per gestire eventuali problemi di connessione** a Internet o errori nelle operazioni successive.

Se il malware è progettato per operare come parte di una rete di botnet o per ricevere comandi da un server di controllo remoto, potrebbe utilizzare la connessione a Internet stabilita per **comunicare con il server remoto e ricevere istruzioni o inviare dati**.

Nel codice sono presenti stringhe come "**Success**" e "**Error**", potrebbe dunque utilizzare una tecnica di cifratura per nascondere queste stringhe o altre informazioni sensibili all'interno del codice.



BONUS

Fare tabella con significato delle singole righe di codice ASSEMBLY

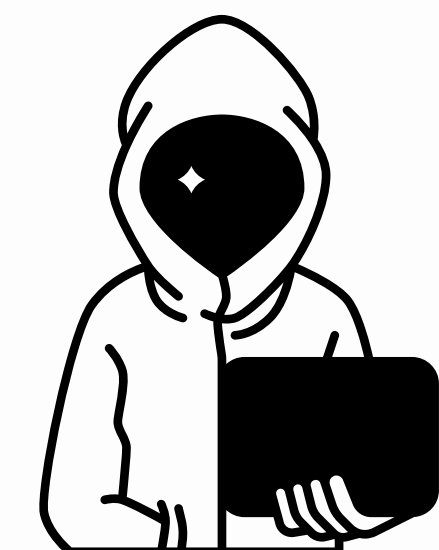
Codice Assembly	Significato
push ebp	Salva il valore corrente di EBP nello stack
mov ebp, esp	Copia il valore dello stack pointer ESP in EBP
push ecx	Spinge il valore corrente di ECX nello stack
push 0	Spinge il valore 0 nello stack
push 0	Spinge il valore 0 nello stack
call ds:InternetConnecedState	Chiama la funzione InternetConnecedState
mov [ebp+var_4], eax	Salva il valore di EAX nella memoria a [ebp+var_4]
cmp [ebp+var_4], 0	Confronta il valore a [ebp+var_4] con 0
jz short loc_40102B	Salta a loc_40102B se il confronto precedente è 0

Codice Assembly	Significato
push offset aSuccessInterne ; "Success"	Spinge l'offset della stringa "Success" nello stack
call sub_40117F	Chiama la funzione sub_40117F
add esp, 4	Aggiunge 4 a ESP
mov eax, 1	Copia il valore 1 in EAX
jmp short loc_40103A	Salta a loc_40103A

Codice Assembly	Significato
loc_40102B_ ; "Error"	Etichetta loc_40102B con commento "Error"
push offset aError1_1NoInte	Spinge l'offset della stringa "Error" nello stack
call sub_40117F	Chiama la funzione sub_40117F
add esp, 4	Aggiunge 4 a ESP
xor eax, eax	Esegue l'operazione XOR tra EAX e se stesso

Codice Assembly	Significato
loc_40103A:	Etichetta loc_40103A
mov esp, ebp	Copia il valore di EBP in ESP
pop ebp	Ripristina il valore di EBP dallo stack
retn	Termina la funzione e ritorna al punto di chiamata
sub_401000 endp	Fine della procedura sub_401000


GRAZIE



ByteRebels

ByteRebels

Team

 +0409200213

 www.ByteRebels.eu

 Ovunque

BY T E R E B E L S

[Home](#) [Team](#)



**Azienda leader nel
settore Cyber Security**

[GITHUB](#)

[I nostri progetti](#)

