12 FEBBRAIO 2024

PROGETTO S10-L5

Corrado li quadri

ANALISI STATICA: -BASICA -AVANZATA

TRACCIA

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:

- Quali librerie vengono importate dal file eseguibile?
- Quali sono le sezioni di cui si compone il file eseguibile del malware?

Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

- Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)
- Ipotizzare il comportamento della funzionalità implementata

Le richieste del progetto includono l'utilizzo di tool come "cff explorer" per condurre un analisi statica basica, ovvero esaminare un eseguibile senza vedere le istruzioni che lo compongono, per poi concludere con un analisi statica avanzata che presuppone la conoscenza dei fondamenti di "reverse-engineering" al fine di identificare il comportamento del malware analizzando il codice Assembly associato.

INDICE

ANALISI STATICA BASICA

ANALISI STATICA AVANZATA

ANALISI STATICA BASICA

Windows utilizza per la maggior parte dei file eseguibili il formato PE, Portable Executable. Formato che contiene informazioni necessarie al sistema operativo per capire come gestire il codice del file. Sono di particolare interesse le librerie importate ed esportate dal malware, questo perchè quando un programma ha bisogno di una funzione, “chiama” una libreria al cui interno è definita la funzione necessaria. Esistono vari modi per importare librerie, quello più comunemente utilizzato è il runtime, ovvero l’importazione a tempo di esecuzione.

RUNTIME

L’eseguibile richiama la libreria solamente quando necessita di una particolare funzione, metodo che permette ai malware di risultare quanto meno invasivi e rilevabili possibile. Solitamente per chiamare la libreria all’occorrenza si utilizzano delle funzioni messe a disposizione dal sistema operativo come <<LOADLIBRERY>> e <<GETPROCADDRESS>>.

CFF EXPLORER

"CFF Explorer" è un software utilizzato per l'analisi di file eseguibili Windows, come i file PE (Portable Executable). L’editor PE è lo strumento più popolare della suite, permette di esaminare e modificare vari aspetti dei file, inclusi header, sezioni, importazioni, esportazioni, risorse e altro ancora. Questo può essere utile per attività come il “reverse-engineering” e l’analisi di crash.

Una volta avviato il tool, bisogna scegliere un file eseguibile da caricare per poter analizzare l’header del PE. Per controllare le librerie e le funzioni importate ci si sposta sulla sezione “import directory” presente all’interno del menù sulla sinistra.

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

KERNEL32.DLL

Kernel32 è essenziale per il funzionamento base di windows, senza di esso il sistema operativo non sarebbe in grado di caricare applicazioni, gestire risorse o interagire con l'hardware. La sua stabilità e integrità sono fondamentali per la stabilità generale del sistema. Danni o corruzioni a questo file possono causare gravi crash e malfunzionamenti del sistema.

È una libreria a collegamento dinamico (DLL) che contiene funzioni essenziali per diverse operazioni di base del sistema.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW
000068FC	000068FC	01BF	LCMapStringA
000068E6	000068E6	01E4	MultiByteToWideChar
00006670	00006670	00CA	GetCommandLineA
00006682	00006682	0174	GetVersion
00006690	00006690	007D	ExitProcess
0000669E	0000669E	029E	TerminateProcess
000066B2	000066B2	00F7	GetCurrentProcess
000066C6	000066C6	02AD	UnhandledExceptionFilter
000066E2	000066E2	0124	GetModuleFileNameA
000066F8	000066F8	00B2	FreeEnvironmentStringsA
00006712	00006712	00B3	FreeEnvironmentStringsW
0000672C	0000672C	02D2	WideCharToMultiByte
00006742	00006742	0106	GetEnvironmentStrings
0000675A	0000675A	0108	GetEnvironmentStringsW
00006774	00006774	026D	SetHandleCount
00006786	00006786	0152	GetStdHandle
00006796	00006796	0115	GetFileType
000067A4	000067A4	0150	GetStartupInfoA
000067B6	000067B6	0126	GetModuleHandleA
000067CA	000067CA	0109	GetEnvironmentVariableA
000067E4	000067E4	0175	GetVersionExA
000067F4	000067F4	019D	HeapDestroy
00006802	00006802	019B	HeapCreate
00006810	00006810	02BF	VirtualFree
0000682A	0000682A	022F	RtlUnwind
00006836	00006836	02DF	WriteFile
00006842	00006842	0199	HeapAlloc
0000684E	0000684E	00BF	GetCPInfo
0000685A	0000685A	00B9	GetACP
00006864	00006864	0131	GetOEMCP
00006870	00006870	02BB	VirtualAlloc
00006880	00006880	01A2	HeapReAlloc
0000688E	0000688E	013E	GetProcAddress
000068A0	000068A0	01C2	LoadLibraryA
000068B0	000068B0	011A	GetLastError
000068C0	000068C0	00AA	FlushFileBuffers
000068D4	000068D4	026A	SetFilePointer
00006950	00006950	001B	CloseHandle

Quando un malware chiama la libreria kernel, possono verificarsi diverse conseguenze a seconda del tipo di malware e delle funzioni specifiche della libreria kernel che vengono chiamate. Ecco alcuni scenari possibili:

Iniezione di codice:

Il malware può iniettare codice dannoso all'interno della libreria kernel, ottenendo il controllo del sistema e la possibilità di eseguire azioni non autorizzate. Questo tipo di attacco è molto pericoloso perché può compromettere la sicurezza del sistema in modo significativo.

Modifica delle funzioni:

Il malware può modificare il comportamento di alcune funzioni della libreria kernel per ottenere specifici vantaggi. Ad esempio, potrebbe modificare la funzione di gestione della memoria per ottenere più memoria di quanta ne dovrebbe avere, oppure potrebbe modificare la funzione di accesso ai file per rubare dati sensibili.

Sovraccarico del sistema:

Il malware può chiamare ripetutamente le funzioni della libreria kernel in modo da sovraccaricare il sistema e causarne il malfunzionamento. Questo tipo di attacco, noto come Denial-of-Service (DoS), può rendere il sistema inutilizzabile per gli utenti legittimi.

Escalation dei privilegi:

Il malware può sfruttare vulnerabilità nella libreria kernel per ottenere privilegi di accesso più elevati all'interno del sistema. Questo può consentire al malware di eseguire azioni che normalmente non sarebbero possibili, come installare software dannoso o modificare le impostazioni di sistema.

Evasione delle difese:

Il malware può utilizzare le funzioni della libreria kernel per nascondere la propria presenza e sfuggire ai software antivirus e antimalware.Questo rende più difficile la rimozione del malware dal sistema.

WININET.DLL

Wininet.dll è un file Dynamic Link Library (DLL) di sistema presente in tutti i sistemi operativi Windows. Sviluppata da microsoft, fornisce un set completo di funzioni per la gestione di connessioni internet e l'accesso a dati online da parte di applicazioni software.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006640	00006640	0071	InternetOpenUrlA
0000662A	0000662A	0056	InternetCloseHandle
00006616	00006616	0077	InternetReadFile
000065FA	000065FA	0066	InternetGetConnectedState
00006654	00006654	006F	InternetOpenA

Quando un malware chiama la libreria wininet, possono verificarsi diverse conseguenze a seconda del tipo di malware e delle funzioni specifiche della libreria kernel che vengono chiamate. Ecco alcuni scenari possibili:

Funzioni di download e upload:

Download di malware: Il malware può utilizzare la funzione di download di file di Wininet.dll per scaricare e installare software dannoso sul sistema.
Exfiltrazione di dati: Il malware può utilizzare la funzione di upload di file di Wininet.dll per rubare dati sensibili come password, numeri di carta di credito o informazioni finanziarie e inviarli a un server remoto controllato dagli hacker.

Funzioni di gestione di connessioni:

Botnet: Il malware può utilizzare la funzione di connessione di rete di Wininet.dll per connettersi a un server di comando e controllo e unirsi a una botnet, una rete di computer infetti controllati da un hacker.
Attacco Denial-of-Service (DoS): Il malware può utilizzare la funzione di connessione di rete di Wininet.dll per bombardare un server con richieste di connessione, rendendolo inutilizzabile per gli utenti legittimi.

Funzioni di autenticazione:

Keylogging: Il malware può utilizzare la funzione di autenticazione di Wininet.dll per registrare le sequenze di tasti premuti dall'utente, acquisendo password e altre informazioni sensibili.
Attacco "credential stuffing": Il malware può utilizzare le credenziali rubate per accedere ad altri account online dell'utente.

Funzioni di gestione cookie e cache:

cookie: Il malware può utilizzare la funzione di gestione cookie di Wininet.dll per rubare i cookie di autenticazione dell'utente, che possono essere utilizzati per accedere ai suoi account online.
Analisi della cache: Il malware può analizzare la cache del browser web dell'utente per identificare informazioni sensibili come dati di accesso a siti web o cronologia di navigazione.

SECTION HEADERS

L'header del formato PE fornisce molte altre informazioni importanti oltre alle funzioni/librerie importate ed esportate, come ad esempio le sezioni di cui si compone il software. Ogni sezione rappresenta una porzione di memoria del programma, come il codice eseguibile, i dati iniziali, i dati costanti, le risorse, e così via. Le informazioni contenute in ciascuna sezione includono il nome, l'offset, la dimensione, i flags (ad esempio, sezione di dati, sezione di codice, sezione di risorse), e altre proprietà specifiche della sezione.

Per controllare le sezioni di un file eseguibile, sempre attraverso l'uso del tool cff explorer, ci spostiamo sulla sezione "section headers" nel pannello a sinistra.

Malware_U3_W2_L5.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers
00000230	00000238	0000023C	00000240	00000244	00000248	0000024C	00000250	00000252
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000

.text

La sezione `.text` è una sezione fondamentale del formato file PE che contiene il codice eseguibile vero e proprio di un programma. È composto da istruzioni in linguaggio Assembly o linguaggio macchina, durante l'esecuzione del programma, il contenuto viene caricato in memoria e viene eseguito dal processore, ogni istruzione è rappresentata da un numero specifico di byte che il processore interpreta e esegue.

Trattandosi di un malware la sezione .text potrebbe contenere il vero e proprio payload del malware, che include istruzioni per svolgere attività dannose sul sistema della vittima. Questo potrebbe includere azioni come il furto di informazioni sensibili, l'installazione di backdoor per consentire l'accesso remoto al sistema, il danneggiamento dei file di sistema o la distribuzione di spam.

Analizziamo parte del codice in dettaglio.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00007070	33	3A	20	46	61	69	6C	20	74	6F	20	67	65	74	20	63	3:.Fail.to.get.c
00007080	6F	6D	6D	61	6E	64	0A	00	45	72	72	6F	72	20	32	2E	ommand..Error.2.
00007090	32	3A	20	46	61	69	6C	20	74	6F	20	52	65	61	64	46	2:.Fail.to.ReadF
000070A0	69	6C	65	0A	00	00	00	00	45	72	72	6F	72	20	32	2E	ile.....Error.2.
000070B0	31	3A	20	46	61	69	6C	20	74	6F	20	4F	70	65	6E	55	1:.Fail.to.OpenU
000070C0	72	6C	0A	00	68	74	74	70	3A	2F	2F	77	77	77	2E	70	rl..http://www.p
000070D0	72	61	63	74	69	63	61	6C	6D	61	6C	77	61	72	65	61	racticalmalwarea
000070E0	6E	61	6C	79	73	69	73	2E	63	6F	6D	2F	63	63	2E	68	nalysis.com/cc.h
000070F0	74	6D	00	00	49	6E	74	65	72	6E	65	74	20	45	78	70	tm..Internet.Exp
00007100	6C	6F	72	65	72	20	37	2E	35	2F	70	6D	61	00	00	00	lorer.7.5/pma...

Notiamo che il malware tenta di collegare la macchina vittima dell'attacco al sito http://www.practicalmalwareanalysis.com, un sito dannoso che si visitato potrebbe arrecare ingenti danni sul sistema compromesso.

.rdata

La sezione rdata all'interno di un file eseguibile, è spesso utilizzata per memorizzare dati di sola lettura che il programma utilizza durante l'esecuzione. Questi dati possono includere costanti criptografiche, stringhe di testo sensibili, costanti di configurazione, variabili utilizzate in tecniche di evasione e altro ancora.

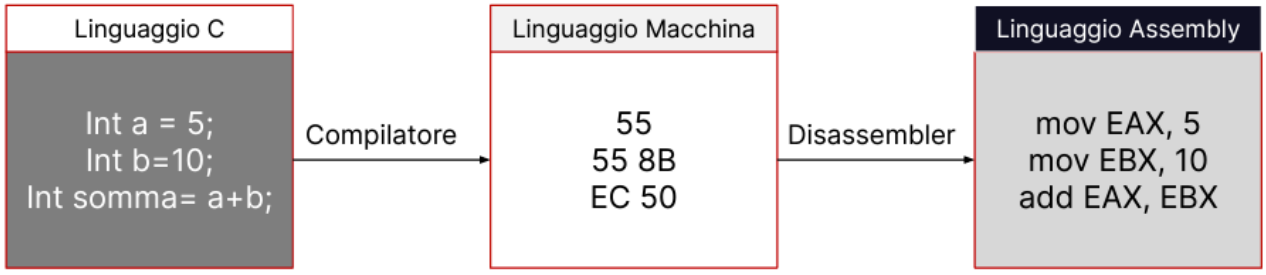
.data

La sezione `.data` all'interno di un file eseguibile, è una parte critica del programma che memorizza dati inizializzati utilizzati durante l'esecuzione del malware. Questi dati possono includere variabili, costanti, strutture dati e altre informazioni che il malware utilizza per svolgere le sue funzionalità

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 "disassembler" che ricevono in input un file eseguibile e restituiscono in output il linguaggio Assembly

ASTRAZIONE

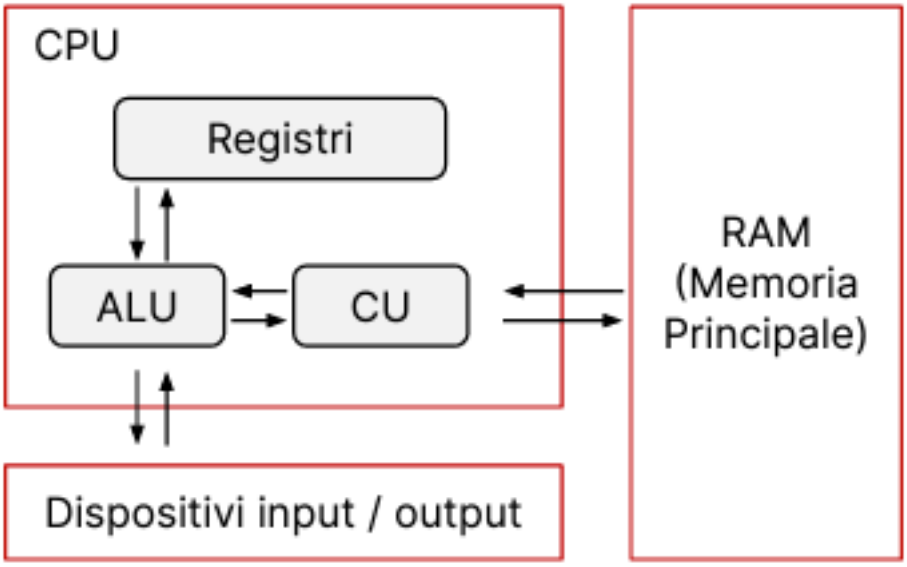


La figura sopra evidenzia quelli che vengono chiamati livelli di astrazione di un sistema informatico. Ogni livello inferiore viene infatti astratto da un livello superiore, l'hardware viene astratto dal sistema operativo per essere utilizzato in maniera semplificata dall'utente.

X86

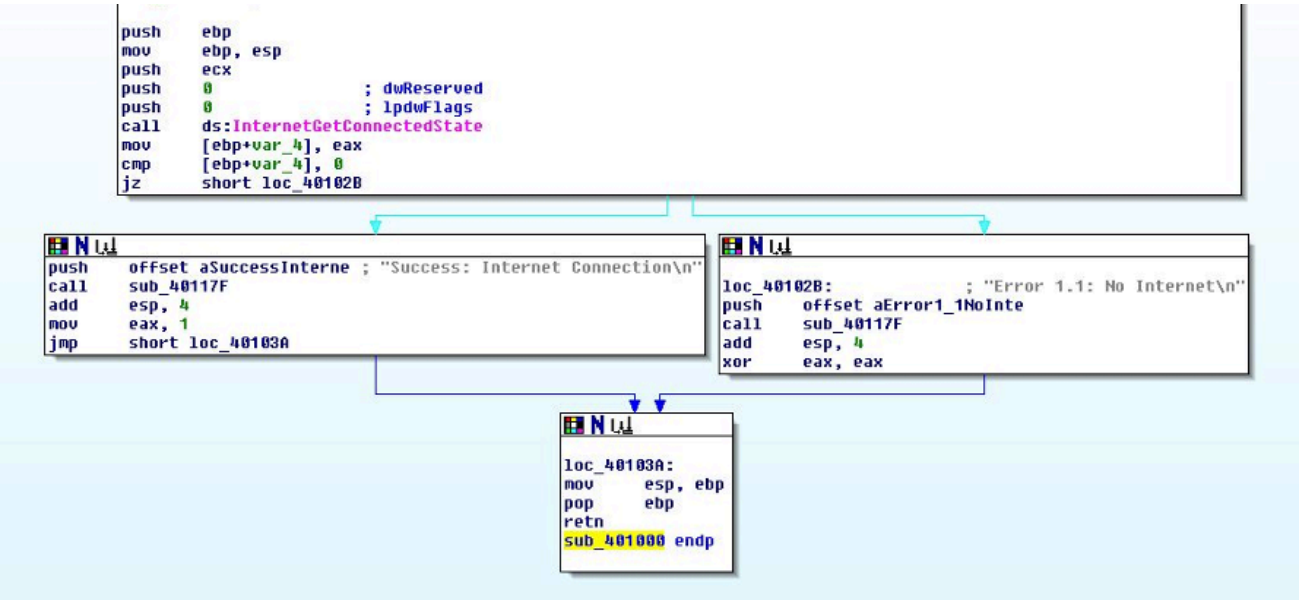
L'assembly è un linguaggio a basso livello che dipende dall'architettura del calcolatore, in questo caso specifico trattiamo l'assembly per il set di istruzioni x86, ovvero per i processori a 32 bit.

X86 segue il modello di architettura di von Neumann.



La RAM, ovvero la memoria principale, memorizza il codice e le istruzioni da eseguire. La CU(control unit) recupera l'istruzione da eseguire dalla RAM utilizzando memorie rapida come i registri della CPU, quest'ultima esegue quindi l'istruzione con il supporto dell'ALU per svolgere calcoli/operazioni laddove necessario.

CODICE ASSEMBLY

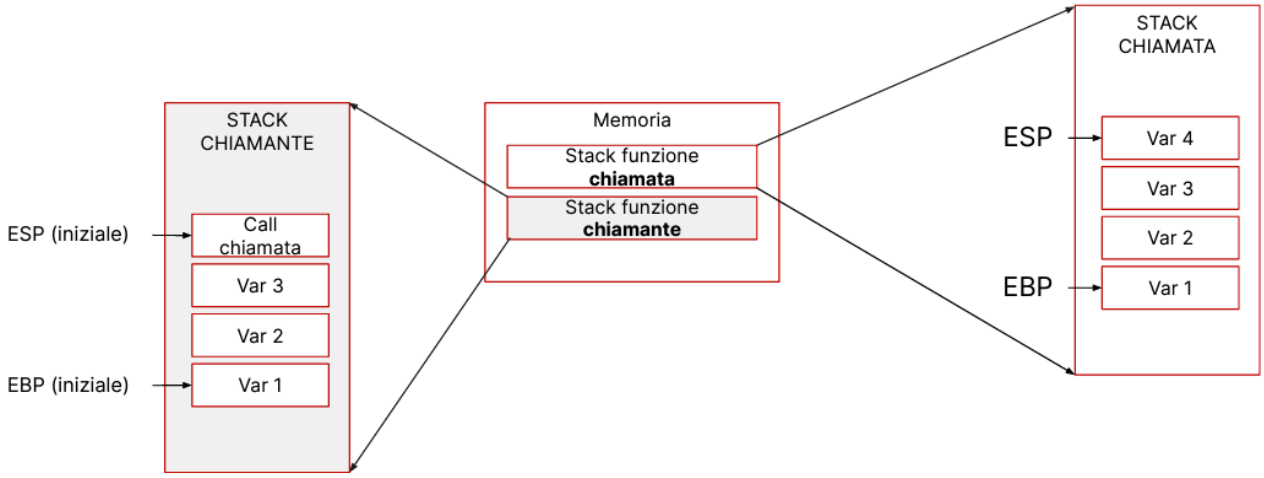


Le istruzioni nel codice assembly sono molto vicine all'hardware del computer e possono includere operazioni come il caricamento di dati dalla memoria, l'esecuzione di operazioni aritmetiche, il trasferimento di dati tra registri e l'interazione con i dispositivi di input/output.

Creazione dello Stack

```
push    ebp
mov     ebp, esp
```

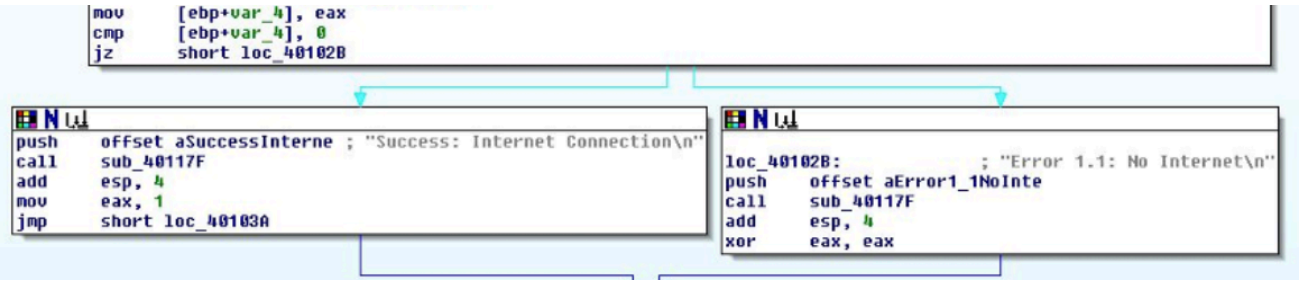
La CPU x86 prevede un sistema di registri per tenere sotto controllo lo Stack durante le chiamate di funzione, questi registri sono EBP e ESP, che sono rispettivamente i puntatori alla base ed alla cima dello Stack. Lo Stack è una porzione di memoria dedicata per il salvataggio delle variabili locali di una data funzione



Viene creato uno Stack per ogni chiamata di funzione (call), la funzione call passa l'esecuzione del programma alla funzione chiamata per la quale verrà creato un nuovo Stack. Il valore di ESP viene copiato in EBP per poter creare un riferimento al frame dello Stack della funzione chiamante.

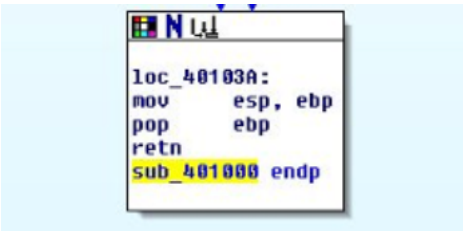
La funzione può utilizzare ESP per accedere alle variabili locali e i parametri passati.

Costrutto if



Il conditional jump utilizza il contenuto dei flag per determinare se “saltare” o meno alla locazione specificata come operando dell’istruzione jump. Nel caso specifico jz è strettamente collegato al risultato dell’istruzione “cmp”, se gli operandi sono uguali l’istruzione modifica il flag ZF=1, altrimenti ZF=0. Riconosciamo il costrutto if dal momento in cui l’istruzione jz effettua il salto condizionale alla locazione 40102B se ZF=1, altrimenti il flusso del programma riprende normalmente passando alla successiva istruzione.

Rimozione stack



Quando la funzione chiamata finisce il suo compito, è necessario eliminare il suo stack e le sue variabili locali non più necessarie per riportare l’esecuzione alla funzione chiamante. Il valore di EBP viene copiato in ESP ripristinando lo stack allo stato precedente alla chiamata della funzione. L’istruzione pop è utilizzata per rimuovere il piatto in cima allo stack seguendo il protocollo LIFO, last in first out.

`stdcall`, `fastcall` e `cdecl` sono convenzioni di chiamata, che specificano come i parametri vengono passati a una funzione e come viene gestita la pulizia dello stack dopo la chiamata. Queste convenzioni sono spesso utilizzate nei linguaggi di programmazione come C e C++ per stabilire un'interfaccia coerente tra funzioni scritte in diversi moduli o linguaggi.

Ecco una spiegazione di ciascuna:

Stdcall (standard call):

In questa convenzione di chiamata, i parametri della funzione sono passati nello stack da destra a sinistra. È responsabilità della funzione chiamante ripulire lo stack dopo la chiamata. È comunemente utilizzato nei sistemi operativi Windows per le API di sistema.

Fastcall (fast call):

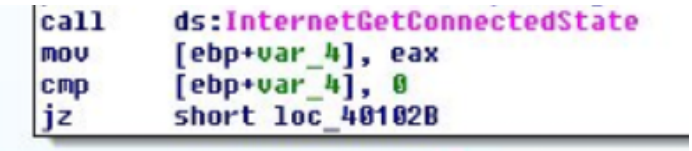
In questa convenzione di chiamata, alcuni parametri della funzione (generalmente i primi) sono passati tramite i registri anziché nello stack. Questo può migliorare le prestazioni, poiché l'accesso ai registri è solitamente più veloce rispetto all'accesso allo stack. Il numero e i registri utilizzati possono variare a seconda dell'implementazione e della convenzione specifica. È responsabilità della funzione chiamata ripulire lo stack dopo la chiamata.

Cdecl (C declaration):

In questa convenzione di chiamata, i parametri della funzione sono passati nello stack da destra a sinistra, proprio come `stdcall`. La differenza principale risiede nel fatto che è responsabilità della funzione chiamante ripulire lo stack dopo la chiamata, anziché della funzione chiamata. Questa è la convenzione di chiamata predefinita in molti compilatori C e C++, e viene spesso utilizzata in sistemi Unix e Linux.

IPOTESI COMPORTAMENTO

Correlata al costrutto if è la chiamata della funzione **internetgetconnectedstate,** utilizzata nella programmazione windows per controllare lo stato della connessione internet di un sistema locale



Questa informazioni può essere utilizzate per determinare se il malware è in grado di eseguire determinate azioni, come scaricare dati o inviare informazioni rubate.

Un’altra ipotesi che potrebbe verificarsi è il reindirizzamento del traffico internet verso siti web dannosi o controllati dal malware. Il costrutto if verifica il parametro restituito dalla funzione getinternetconnectstate, se è uguale a 0, la funzione stampa a schermo “no internet”, in caso contrario stamperà a schermo “success: internet connection”.