







MARIA HUAPAYA




ANALISI Malware

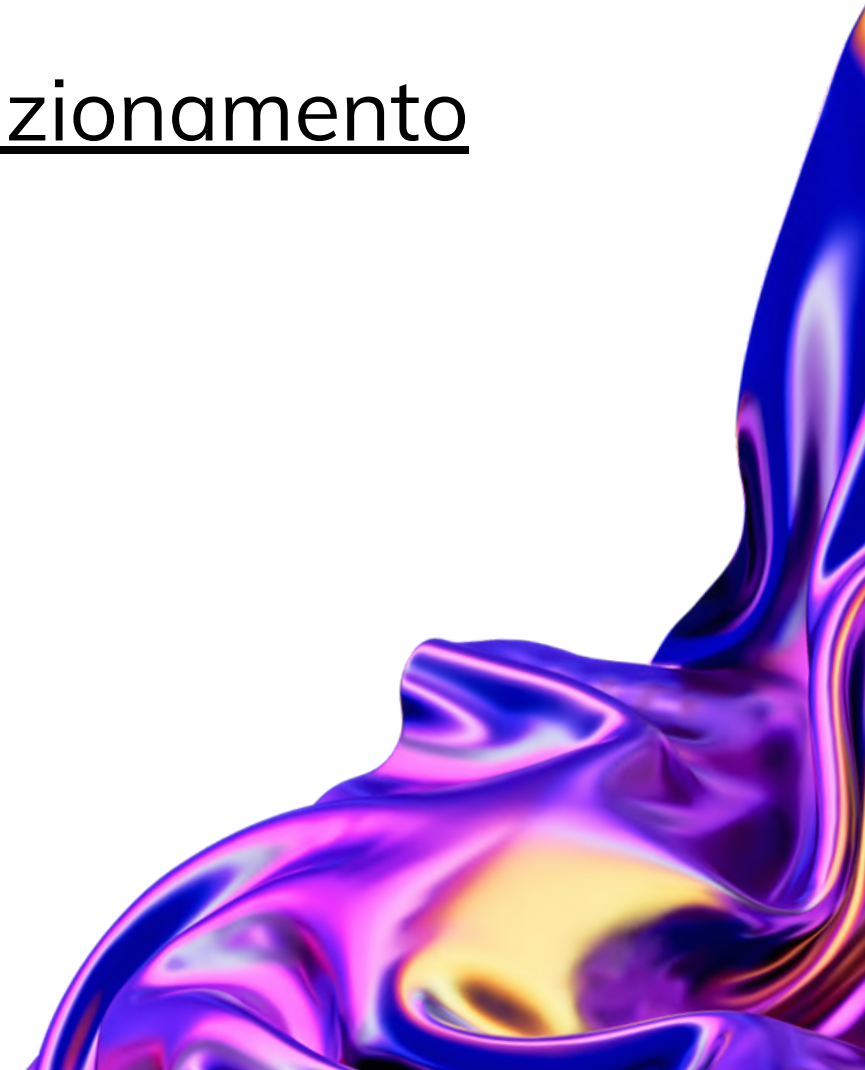
Indice

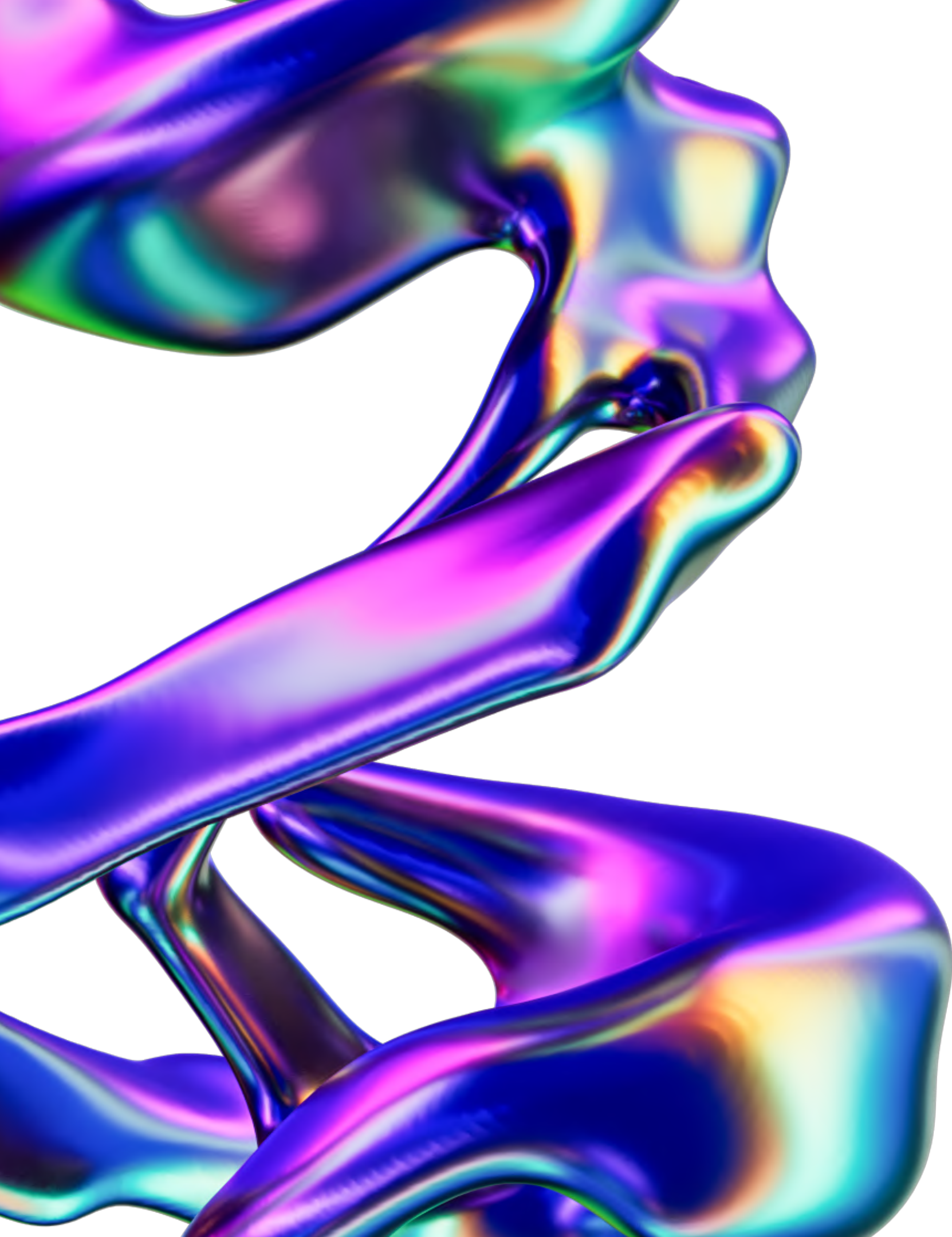
Suggerimento: utilizza i link per passare a un'altra pagina della presentazione.

Come: evidenzia il testo, clicca sul simbolo del link e seleziona la pagina della presentazione che vuoi collegare.

-  Traccia (3-4)
-  Task 1: Cenni alla teoria (5-10)
-  Task 1: Librerie importate (11-19)
-  Task 1: Sezioni che compongono il file eseguibile (20-21)

-  Task 2: Cenni alla teoria (22-23)
-  Task 2: Costrutti noti (24-25)
-  Task 2: Ipotesi sul funzionamento (26-29)





Traccia: task 1

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:

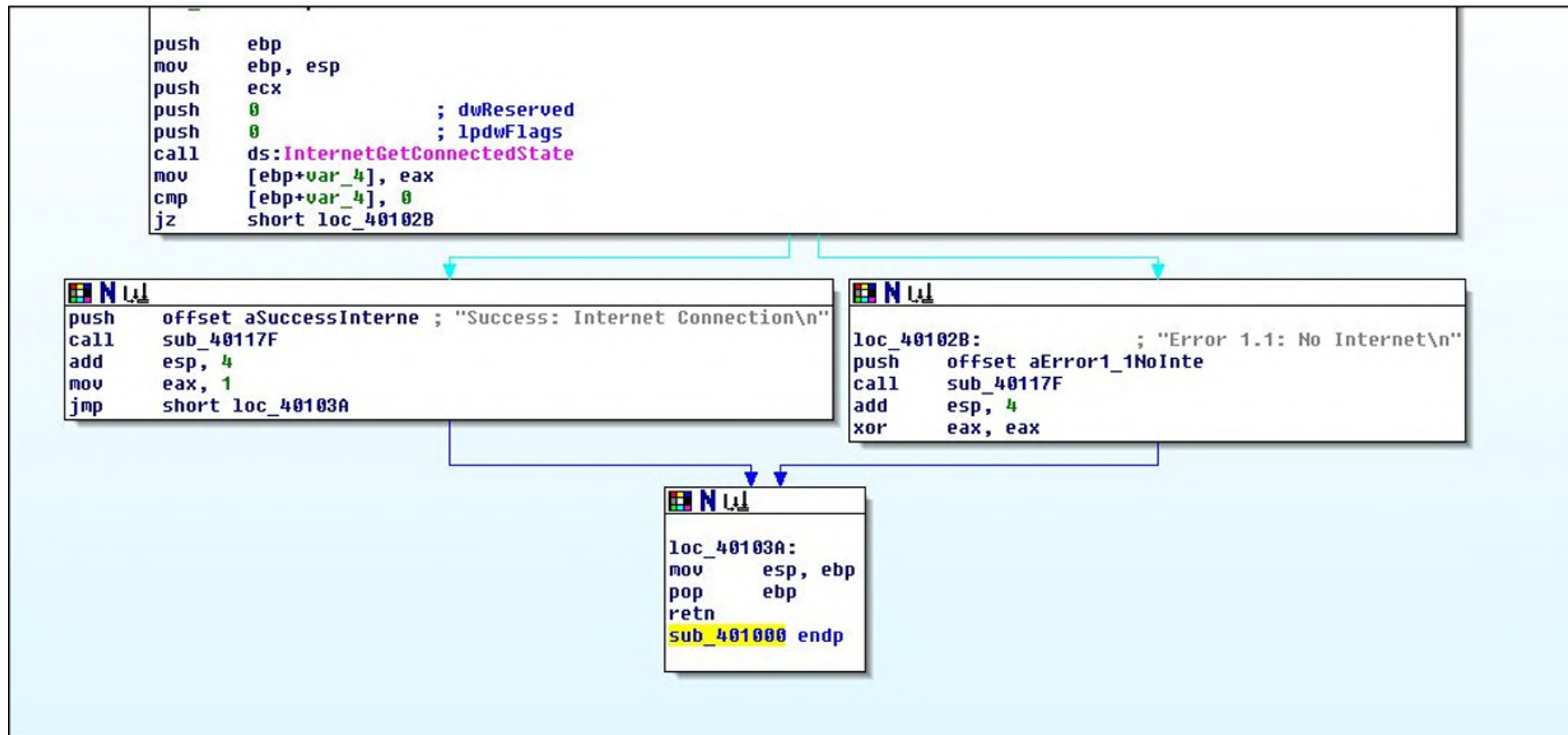
- Quali librerie vengono importate dal file eseguibile?
- Quali sono le sezioni di cui si compone il file eseguibile del malware?

[TORNA ALL'INDICE](#)

Traccia: task 2

Con riferimento alla seguente figura, rispondere ai seguenti quesiti:

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



[TORNA ALL'INDICE](#)

Task 1: Cenni alla teoria

Definizione e categorie di malware

I malware, o software malevoli, sono programmi creati per danneggiare, infiltrarsi o altrimenti compromettere l'integrità dei dati, la privacy e la funzionalità dei sistemi informatici. Si distinguono in varie categorie in base alla loro modalità di azione e agli obiettivi: virus (che si replicano attaccandosi a file), worm (autoreplicanti attraverso le reti), trojan (mascherati da software legittimi), ransomware (che criptano i dati richiedendo un riscatto), spyware (per il furto di informazioni), adware (che mostrano pubblicità indesiderate), e backdoor (che forniscono accesso remoto non autorizzato).

[TORNA ALL'INDICE](#)

Metodi di diffusione

I malware possono diffondersi attraverso vari canali: email phishing, che ingannano gli utenti affinché aprano allegati o clicchino su link malevoli; drive-by download, che sfruttano vulnerabilità del browser per installare malware senza consenso; dispositivi USB infetti; e sfruttamento di vulnerabilità di rete, dove gli aggressori utilizzano debolezze software per infiltrarsi nei sistemi.

Obiettivi dei malware

Gli obiettivi variano dal furto di informazioni sensibili (dati personali, credenziali finanziarie), al danneggiamento dei sistemi (cancellazione o corruzione di dati), allo spionaggio industriale (furto di proprietà intellettuale), al criptovalute mining non autorizzato, sfruttando le risorse del sistema infetto.

Tecniche di evasione

Per evitare il rilevamento, i malware possono utilizzare varie tecniche: offuscamento del codice per nascondere la loro vera natura, rootkit per ottenere il controllo a basso livello del sistema operativo eludendo i software di sicurezza, polimorfismo e metamorfismo per cambiare il proprio codice ad ogni infezione, rendendo difficile la loro identificazione da parte dei software antivirus.

L'analisi del malware

È un processo cruciale nella cybersecurity, mirato a comprendere il comportamento, la funzionalità e l'intento di un software malevolo. Si divide principalmente in due categorie: analisi statica e analisi dinamica.

- **Analisi statica:** questo approccio non richiede l'esecuzione del malware. Si basa sull'ispezione del codice, l'analisi dei binari, e l'esame delle stringhe, delle firme digitali e delle strutture di controllo per identificare il potenziale comportamento malevolo senza attivarlo. Gli strumenti utilizzati possono includere disassemblatori e decompilatori per esaminare il codice sorgente.
- **Analisi dinamica:** invece di analizzare il codice sorgente, questo metodo comporta l'esecuzione del malware in un ambiente controllato e isolato, noto come sandbox, per osservarne il comportamento in tempo reale. L'analisi dinamica permette di rilevare le attività di rete, le modifiche al file system, i processi generati e le modifiche al registro, fornendo insight preziosi sull'operato del malware e sulle sue strategie di evasione.

Entrambi gli approcci sono complementari: l'analisi statica fornisce una comprensione preliminare del codice e delle potenziali capacità del malware, mentre l'analisi dinamica offre una visione pratica del suo comportamento in un ambiente operativo. La combinazione di entrambi gli approcci garantisce una comprensione approfondita e completa delle minacce, essenziale per sviluppare strategie di mitigazione efficaci.



Per la prima parte dell'esercizio è utile **approfondire l'analisi statica del malware**.

Si tratta di una tecnica fondamentale nel campo della cybersecurity, che permette di esaminare il codice di un programma sospetto senza eseguirlo. Questo tipo di analisi fornisce una visione approfondita delle potenziali funzionalità malevole e dei metodi di attacco che il malware potrebbe utilizzare.

Principi Fondamentali dell'Analisi Statica

L'analisi statica si basa sull'osservazione diretta del codice sorgente, dei binari, o di altri componenti del software senza eseguirlo. Questo approccio mira a identificare stringhe sospette, funzioni di libreria, chiamate di sistema, e altre caratteristiche che possono suggerire un comportamento malevolo.

[TORNA ALL'INDICE](#)



CFF Explorer

CFF Explorer è uno strumento avanzato di analisi e editing per file eseguibili Windows, parte della suite di strumenti NTCore's Explorer Suite. È ampiamente utilizzato nell'ambito dell'analisi di malware, della reverse engineering e dello sviluppo di software per esaminare le intricate strutture dei file Portable Executable (PE), che includono .EXE, .DLL, e altri formati eseguibili o di libreria su sistemi Windows.

Caratteristiche Principali di CFF Explorer

- **Visualizzazione e Modifica dell'Header PE:** CFF Explorer permette di esaminare dettagliatamente l'header PE di un file eseguibile, fornendo informazioni vitali come la tabella delle importazioni/esportazioni, le sezioni del file, i metadati e molto altro.
- **Analisi delle Dipendenze:** attraverso la visualizzazione delle librerie importate e delle funzioni specifiche richieste dal file, CFF Explorer aiuta a identificare le potenziali interazioni del malware con il sistema operativo o con altri software.
- **Modifica delle Risorse:** gli utenti possono visualizzare e modificare le risorse incorporate in un file eseguibile, come icone, stringhe e dialoghi, consentendo una personalizzazione avanzata o l'analisi delle componenti nascoste.

[TORNA ALL'INDICE](#)

- Supporto per i Formati di File Multipli: oltre ai file PE, CFF Explorer gestisce una varietà di altri formati, rendendolo uno strumento versatile per l'analisi di file di diversa natura.
- Analisi dell'Entropia: questa funzionalità consente di valutare la distribuzione dei dati all'interno del file, fornendo indizi sull'uso di tecniche di compressione o crittografia, spesso indicatori di offuscamento nel malware.

Utilità nell'Analisi di Malware

Durante l'analisi di malware, CFF Explorer si rivela uno strumento prezioso per svelare le tecniche utilizzate dagli sviluppatori di malware per nascondere le loro tracce o per implementare funzionalità complesse. Ad esempio, l'esame della tabella delle importazioni può rivelare l'intenzione di un malware di manipolare i registri di sistema, di stabilire connessioni di rete, o di eseguire operazioni sui file.

Interfaccia Utente e Funzionalità Avanzate

CFF Explorer è stato progettato con un'interfaccia utente intuitiva che semplifica la navigazione tra le complesse strutture dei file eseguibili. Le sue funzionalità avanzate, come l'editor hex integrato e il visualizzatore di assembly, offrono agli analisti di malware e ai reverse engineer gli strumenti necessari per un'analisi approfondita.

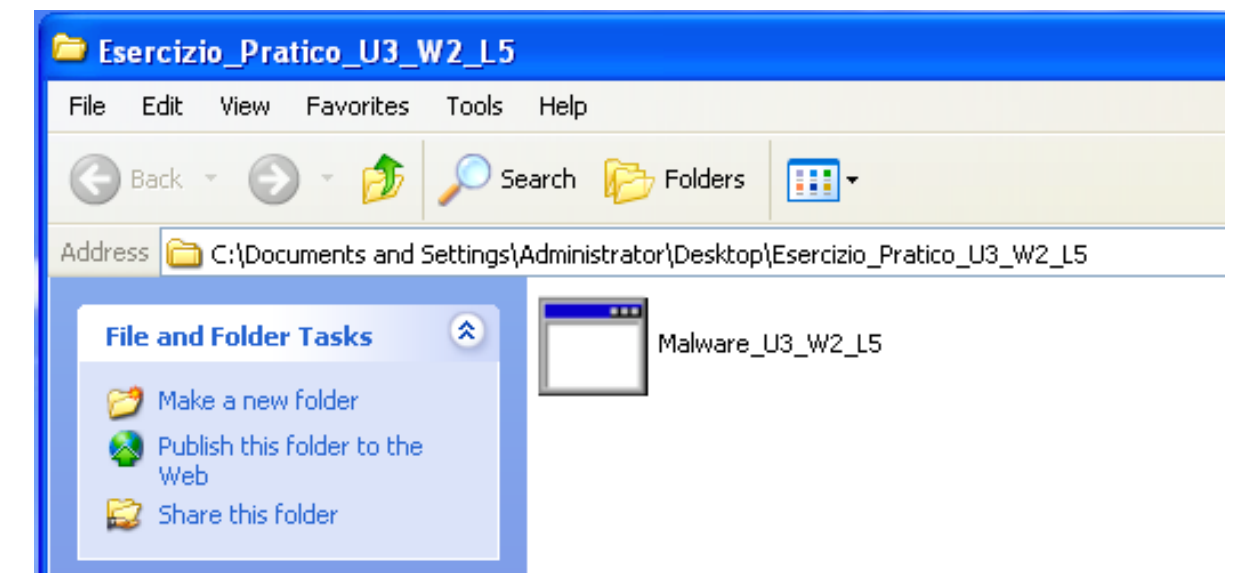
[TORNA ALL'INDICE](#)

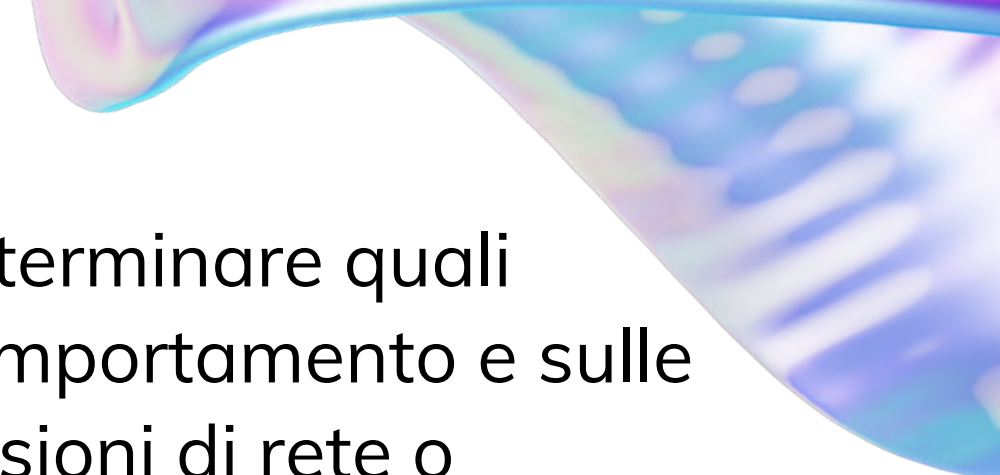
Librerie importate dal file eseguibile

Obiettivo iniziale è l'**identificazione delle librerie importate dal file eseguibile** posizionato all'interno della macchina virtuale "Malware Analysis_Final-001".

La VM opera sotto il sistema operativo Windows XP Service Pack 3 (SP3), una piattaforma comunemente studiata in contesti di analisi del malware per la sua storica popolarità e le sue vulnerabilità note.

Questo processo è fondamentale per comprendere come il malware interagisca con il sistema operativo e quali funzionalità di sistema cerchi di sfruttare per la sua esecuzione e diffusione.





Identificare le librerie importate dal file eseguibile permette agli analisti di determinare quali chiamate di sistema siano effettuate dal malware, fornendo indizi sul suo comportamento e sulle sue potenziali capacità, come la manipolazione di file, la creazione di connessioni di rete o l'esecuzione di codice arbitrario. Utilizzando strumenti di analisi statica come CFF Explorer, gli analisti possono esaminare il file eseguibile senza eseguirlo, riducendo così il rischio di infezione. Questi strumenti consentono di visualizzare le dipendenze del file, inclusi i dettagli sulle librerie importate e le funzioni specifiche utilizzate.

È doveroso sottolineare che questo approccio può risultare limitato di fronte a minacce sofisticate, le quali possono impiegare tecniche avanzate per occultare la loro vera natura, rendendo così l'analisi statica basica una prima tappa piuttosto che una soluzione definitiva all'identificazione del malware.

Per procedere con l'analisi e lo svolgimento della prima task, faremo uso di CFF Explorer, uno strumento specificamente progettato per l'ispezione degli header dei formati Portable Executable (PE), il fondamento dei file eseguibili su sistemi operativi Windows. Questo formato è cruciale poiché incapsula le informazioni necessarie al sistema per gestire adeguatamente il codice del file, fungendo da vero e proprio DNA del software in esecuzione.

L'header del formato PE si rivela un terreno fertile per l'analisi, contenendo dati critici come:

- **Elenco delle librerie importate e delle funzioni richieste:** essenziale per comprendere con quali componenti del sistema operativo il file intende interagire. L'importazione di specifiche librerie può suggerire determinate funzionalità o obiettivi del malware, offrendo indizi sulla sua strategia di attacco.
- **Funzioni esportate:** indicano quali operazioni il file mette a disposizione per altri programmi, una caratteristica che può essere sfruttata per moduli malware che operano in sinergia o per consentire ad applicazioni legittime di avvalersi di determinate funzionalità.
- **Struttura delle sezioni del software:** offre una mappa del file eseguibile, delineando come è organizzato e quali segmenti potrebbero nascondere codice malevolo o dati significativi.

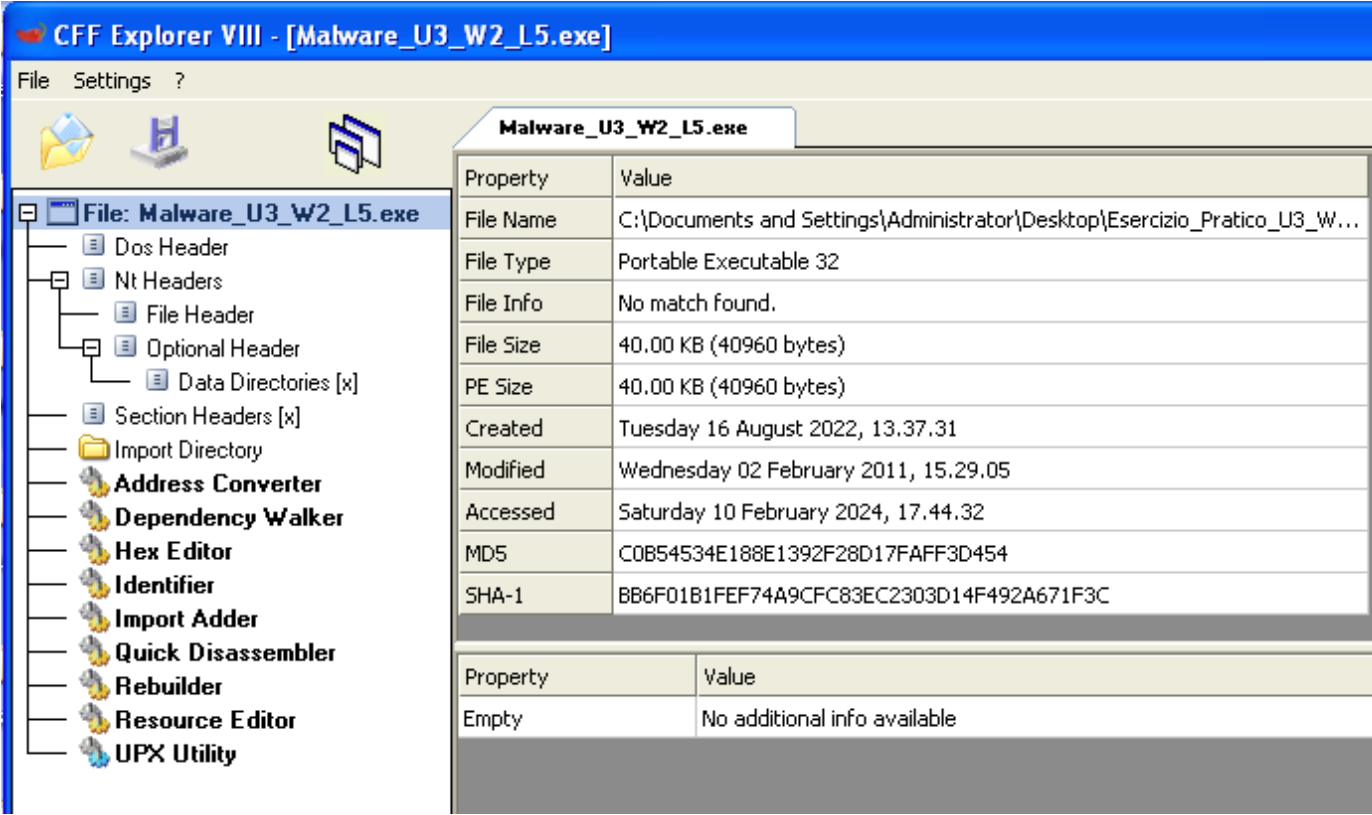
Le librerie, essendo collezioni di funzioni predefinite, giocano un ruolo vitale nell'esecuzione di un malware, fornendo le risorse necessarie per le sue operazioni. Analizzare quali librerie e funzioni vengono importate da un file eseguibile può svelare molto del suo comportamento previsto e delle potenziali minacce che rappresenta.

Iniziando l'analisi con CFF Explorer, apriremo il file Malware_U3_W2_L5.exe per esaminare in dettaglio questi elementi.

L'interfaccia di CFF Explorer offre una panoramica immediata del file Malware_U3_W2_L5.exe, presentando informazioni essenziali come il nome del file, le dimensioni, la data di creazione e gli hash MD5 e SHA-1.

Questi hash sono fondamentali: agiscono come impronte digitali del file, permettendo di confrontarlo con database come VirusTotal per verificare se è già noto come parte di software dannoso e per ottenere dettagli sul suo comportamento precedentemente osservato.

Le dimensioni del file sono contenute e ammontano a 40.00 KB (40,960 bytes), un attributo che, sebbene non sia un indicatore diretto di malevolenza, può essere correlato a un payload di malware di piccole dimensioni. Il file è stato creato il 16 agosto 2022 e modificato l'ultima volta il 2 febbraio 2011, suggerendo una possibile discrepanza nelle date che potrebbe necessitare di ulteriori indagini, poiché tattiche comuni di malware includono la manipolazione delle date per evitare il rilevamento.



L'hash MD5 è
C0B54534E188E1392F28D17FAFF3D454, mentre
l'SHA-1 è
BB6F01B1FEF74A9CFC83EC2303D14F492A671F3C.
Questi valori univoci sono utilizzati per confermare
l'integrità del file e per cercare corrispondenze
all'interno dei database di indicatori di compromissione
(IoC). Una corrispondenza potrebbe confermare la
presenza di codice malevolo o potrebbe indicare che il
file è stato analizzato precedentemente nella comunità
di sicurezza informatica.
Eseguendo una rapida ricerca su VirusTotal abbiamo
conferma che il file è stato segnalato come malevolo
da 40 security vendors ed etichettato come trojan.



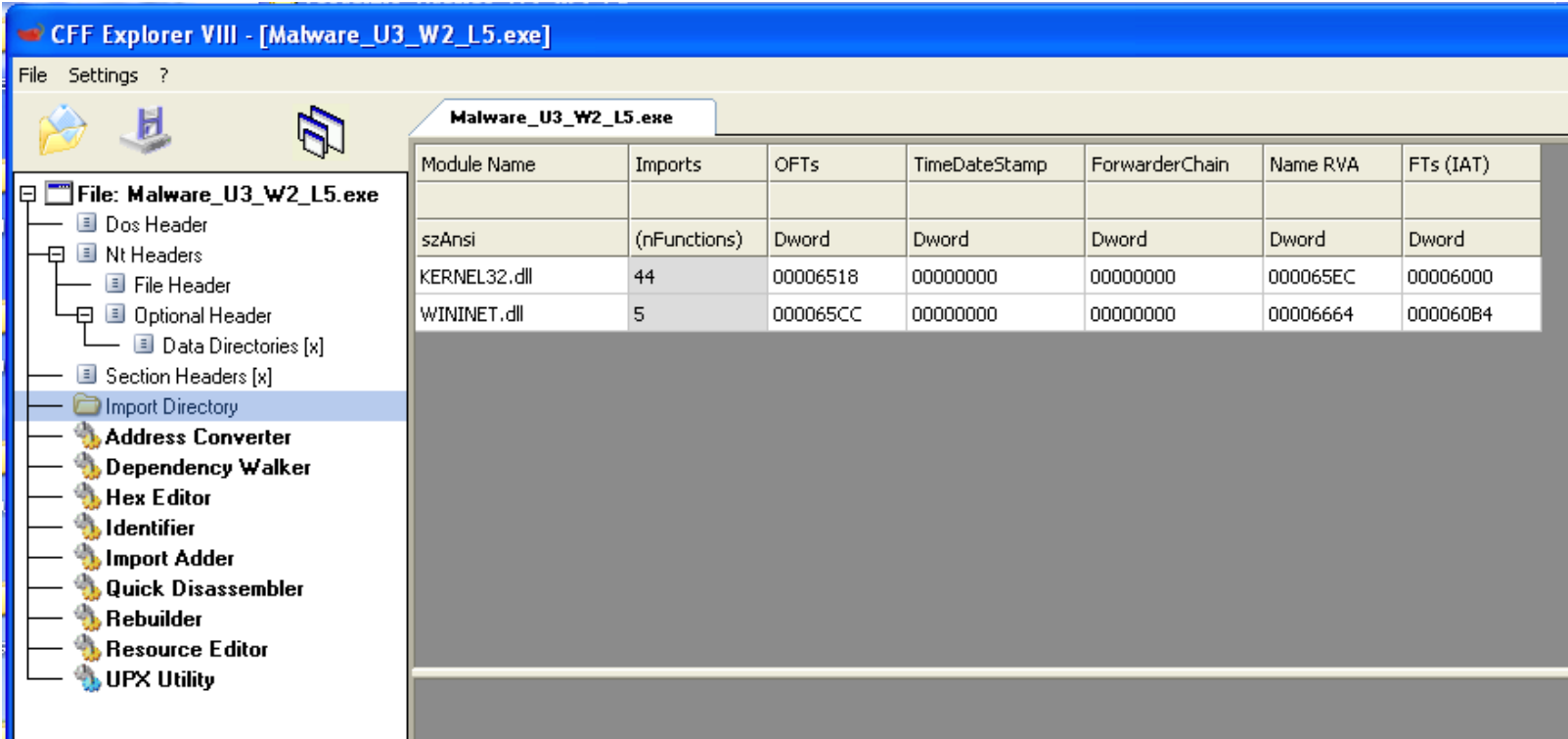
The screenshot displays the VirusTotal interface. At the top, the VirusTotal logo is visible. Below it, a search bar contains the MD5 hash: C0B54534E188E1392F28D17FAFF3D454. The search results show a file named "Lab06-02.exe" with a SHA-256 hash of b71777edbf21167c96d20ff803cbcb25d24b94b3652db2f286dcd6efd3d8416a. The file is labeled as "trojan" and has a "Community Score" of 40/72, indicated by a red circular progress bar. A warning message states: "40 security vendors and no sandboxes flagged this file as malicious". The file is also labeled with "peexe", "checks-network-adapters", "runtime-modules", "armadillo", and "direct-cpu-clock-access". The interface includes tabs for "DETECTION", "DETAILS", "RELATIONS", "BEHAVIOR", and "COMMUNITY". A banner at the bottom encourages joining the VT Community. The "Popular threat label" is "trojan.r002c0pdm21" and the "Threat categories" are "trojan".

[TORNA ALL'INDICE](#)

Spostandoci ora alla sezione "Import Directory" tramite l'interfaccia di CFF Explorer, possiamo esaminare quali librerie specifiche vengono importate dal file eseguibile. Questo è un passaggio cruciale dell'analisi statica, in quanto fornisce un'indicazione delle API di Windows che il malware intende utilizzare. Per esempio, un'esecuzione frequente di librerie di rete potrebbe indicare che il malware è progettato per comunicare con un server di comando e controllo, mentre l'importazione di librerie grafiche potrebbe suggerire un tentativo di manipolare l'interfaccia utente o di mostrare annunci pubblicitari ingannevoli.

All'interno della "Import Directory" di CFF Explorer, si nota che il file eseguibile Malware_U3_W2_L5.exe importa due librerie particolarmente significative: KERNEL32.dll e WININET.dll.

La libreria KERNEL32.dll è fondamentale per qualsiasi eseguibile Windows, poiché fornisce le funzioni di base per le operazioni essenziali come la manipolazione dei file e la gestione della memoria. È quasi universale nel suo utilizzo, pertanto la sua presenza non è di per sé un indicatore di attività malevola.



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

Tuttavia, dato che i malware spesso richiamano funzioni standard per mascherare le loro attività dannose, è importante analizzare il contesto in cui queste funzioni vengono richiamate.

D'altra parte, la libreria WININET.dll suggerisce capacità di connettività di rete, poiché contiene funzioni per interagire con protocolli Internet come HTTP e FTP. L'uso di questa libreria può indicare la possibilità che il malware esegua attività di rete, come la comunicazione con un server remoto, il download o l'upload di file, o l'interazione con servizi web.

Da un'attenta analisi risulta evidente che il file Malware_U3_W2_L5.exe utilizza funzioni specifiche dalle librerie KERNEL32.dll e WININET.dll.

File: Malware_U3_W2_L5.exe

Dos Header

Nt Headers

File Header

Optional Header

Data Directories [x]

Section Headers [x]

Import Directory

Address Converter

Dependency Walker

Hex Editor

Identifier

Import Adder

Quick Disassembler

Rebuilder

Resource Editor

UPX Utility

Malware_U3_W2_L5.exe

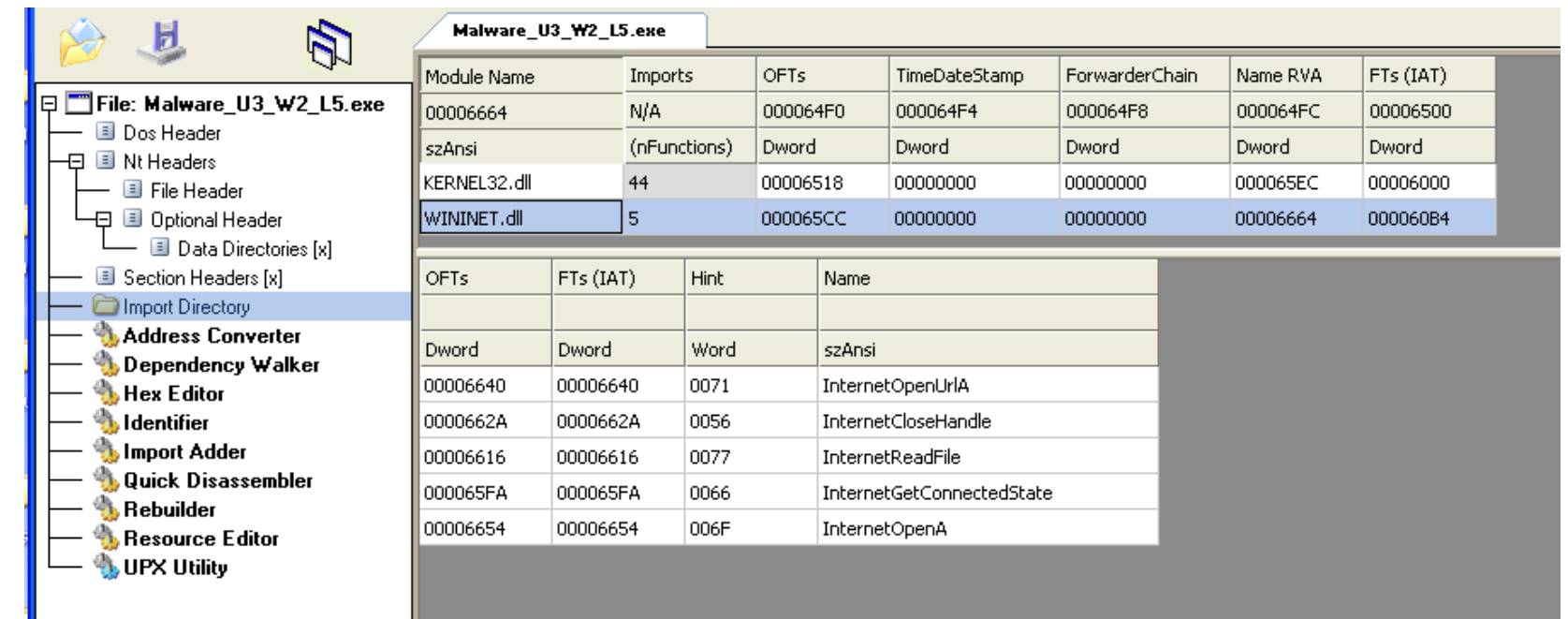
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000065EC	N/A	000064DC	000064E0	000064E4	000064E8	000064EC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
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
0000681E	0000681E	019F	HeapFree
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

La presenza di LoadLibraryA e GetProcAddress nel contesto della libreria KERNEL32.dll è significativa: queste funzioni permettono di caricare altre librerie dinamicamente e di localizzare gli indirizzi delle funzioni al loro interno durante il runtime. La presenza di LoadLibraryA e GetProcAddress nel contesto della libreria KERNEL32.dll è significativa: queste funzioni permettono di caricare altre librerie dinamicamente e di localizzare gli indirizzi delle funzioni al loro interno durante il runtime.

Tale modalità di importazione dinamica è una caratteristica comune nei software legittimi, ma viene spesso sfruttata anche dai malware per caricare moduli dannosi in modo discreto, evitando di esporre le proprie intenzioni prima dell'esecuzione e di conseguenza riducendo la propria rilevabilità.

Le funzioni di gestione della memoria come VirtualAlloc e HeapAlloc, anch'esse presenti nell'elenco delle funzioni importate da KERNEL32.dll, sono essenziali per la corretta esecuzione di ogni processo, ma possono anche essere utilizzate per tecniche come l'heap spraying, comune in attacchi di exploit.



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

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

Esaminando la libreria WININET.dll, si osserva la funzione InternetGetConnectedState, che verifica la connettività a Internet della macchina. Questa funzione è spesso utilizzata dai malware per determinare se una connessione di rete è disponibile prima di tentare attività come il download di payload aggiuntivi o la comunicazione con un server di comando e controllo (C&C).

La presenza di InternetOpenUrlA e InternetReadFile indica inoltre che il malware potrebbe essere progettato per scaricare o caricare dati dalla rete. Queste funzioni facilitano operazioni di rete come l'accesso a risorse Internet specifiche o la lettura di contenuti online, consentendo potenzialmente al malware di ricevere configurazioni aggiornate o inviare informazioni raccolte da una macchina infetta.

[TORNA ALL'INDICE](#)

Sezioni di cui si compone il file eseguibile

Proseguiamo con l'identificazione delle sezioni di cui si compone il file eseguibile del malware.

Spostandoci nella sezione "Section Headers" abbiamo una visuale delle parti costitutive del file eseguibile.

Le sezioni rilevate sono .text, .rdata e .data, ognuna con ruoli distinti nell'architettura del file eseguibile.



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

La sezione .text è dove risiedono le istruzioni eseguibili, ovvero il codice che la CPU elaborerà una volta avviato il programma. È la parte critica di un eseguibile perché contiene il codice sorgente compilato che determina il comportamento del software.

La sezione .rdata include le informazioni readonly, come le librerie importate e le funzioni esportate, fornendo metadati essenziali per la corretta esecuzione del file. Questi metadati aiutano il sistema operativo a mappare le funzioni e le librerie necessarie per l'eseguibile.

La sezione .data contiene le variabili globali del programma, ossia quelle variabili che sono accessibili da qualsiasi parte del codice durante l'esecuzione del programma. Queste variabili sono fondamentali per il mantenimento degli stati durante l'esecuzione del programma e possono includere impostazioni di configurazione, buffer di dati, o anche potenzialmente spazi per dati malevoli.

È importante notare che, mentre queste sono le sezioni standard trovate in molti file eseguibili, i malware avanzati possono manipolare o nascondere queste sezioni, o addirittura crearne di nuove con nomi ingannevoli o inusuali per eludere l'analisi statica. Ciò può confondere gli analisti o i software di sicurezza automatici che cercano di identificare il comportamento malevolo basandosi sulla struttura standard di un file eseguibile.

Task 2: Cenni alla teoria

Assembly

Assembly è un linguaggio di programmazione di basso livello che serve come collegamento tra il codice sorgente ad alto livello e il codice macchina. Ogni processore ha il suo set di istruzioni assembly, specifico per l'architettura della CPU.

Caratteristiche di assembly:

- vicinanza al hardware: consente una comunicazione quasi diretta con le componenti fisiche del computer.
- performance: offre la possibilità di scrivere codice estremamente ottimizzato.
- controllo: garantisce un controllo dettagliato sulle operazioni di sistema.

[TORNA ALL'INDICE](#)

Elementi di un programma assembly:

- istruzioni: comandi per la CPU che eseguono operazioni specifiche.
- registri: piccole locazioni di memoria utilizzate per operazioni rapide e per conservare i risultati intermedi.
- direttive: istruzioni per l'assemblatore che influiscono sull'assemblaggio del programma.
- etichette: nomi assegnati a blocchi di codice o dati che agevolano il riferimento a essi durante l'esecuzione del codice.

Assemblaggio e linking:

- l'assemblaggio converte il codice assembly in codice macchina.
- il linking unisce il codice macchina con altre parti del codice o librerie, creando un file eseguibile.

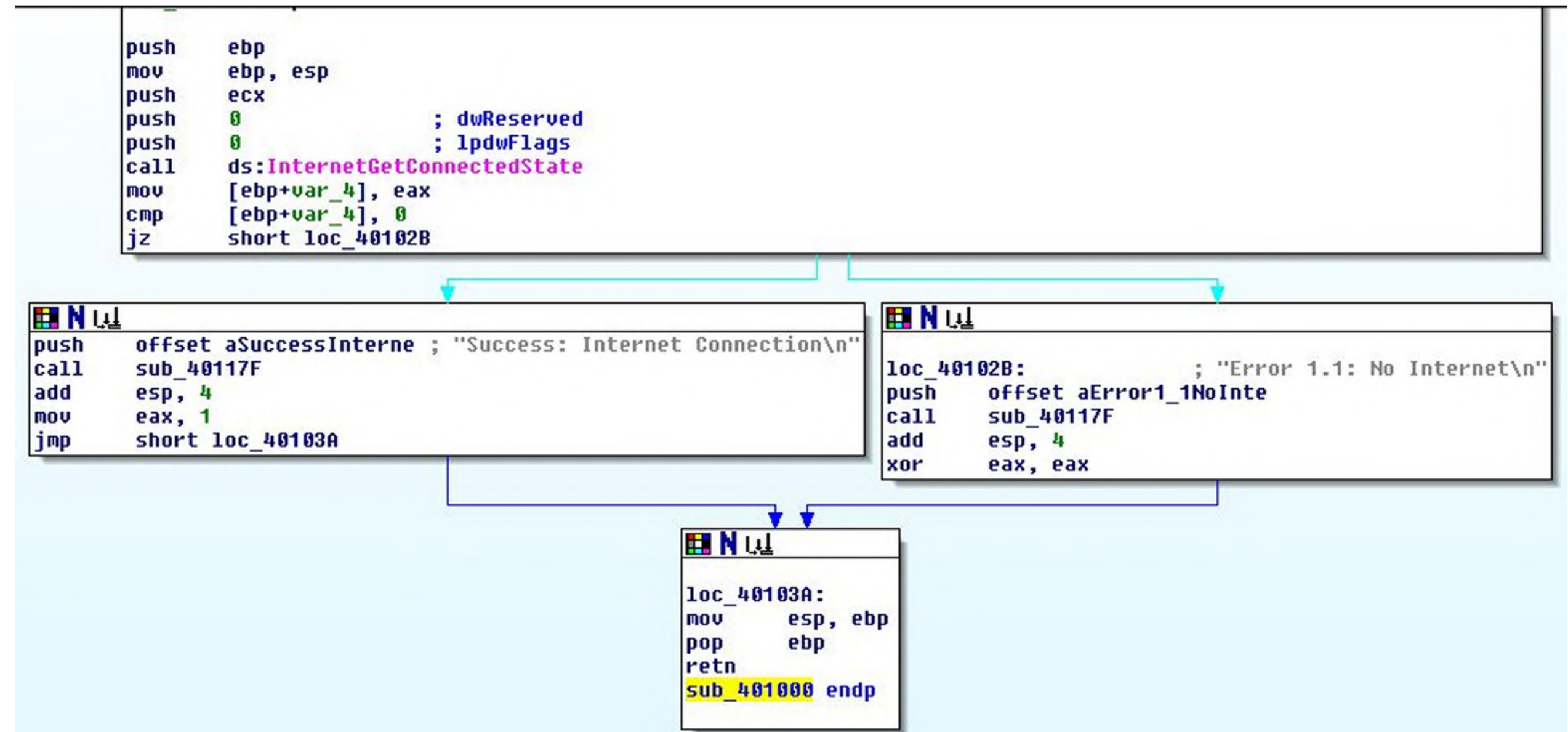
Uso di assembly oggi:

- sviluppo di sistemi embedded: per il massimo controllo dell'hardware.
- codice critico per le prestazioni: come gli algoritmi di crittografia o la grafica computerizzata.
- reverse engineering: per analizzare software e identificare vulnerabilità o malware.
- driver di dispositivo: per gestire la comunicazione diretta con l'hardware.



Identificazione dei costrutti noti

Nella figura fornita abbiamo un frammento di codice assembly, presumibilmente estratto durante un'analisi dinamica o un reverse engineering di un file eseguibile.



Costrutti noti

- Creazione dello Stack Frame: all'inizio del frammento, le istruzioni *push ebp; mov ebp, esp* stabiliscono un nuovo frame nello stack. Questo è un tipico pattern di prologo di una funzione in assembly x86, dove *ebp* è il base pointer che tiene traccia dell'inizio dello stack frame corrente e *esp* è lo stack pointer che tiene traccia della cima dello stack.
- Preparazione dei parametri per la chiamata di funzione: viene fatto *push* di vari valori nello stack prima di chiamare la funzione `InternetGetConnectedState`. Questi valori rappresentano i parametri per la funzione.
- Controllo Condizionale: dopo la chiamata a `InternetGetConnectedState`, l'istruzione *cmp* confronta il risultato con zero e *jz* (jump if zero) determina il flusso del programma basato sul risultato di tale confronto. Se il risultato è zero (nessuna connessione a Internet), il programma salta a un blocco di codice che presumibilmente gestisce questa situazione (mostrando un messaggio di errore, come suggerito dal testo "Error 1-1: No Internet").
- Fine della Funzione: le istruzioni *mov esp, ebp; pop ebp* sono il tipico epilogo di una funzione, che ripristina lo stack pointer (*esp*) e il base pointer (*ebp*) ai loro valori precedenti. *retn* è l'istruzione che ritorna il controllo al chiamante della funzione.

Ipotesi sul funzionamento

Le istruzioni in linguaggio Assembly che abbiamo di fronte descrivono una sequenza di operazioni che il programma esegue per gestire e manipolare lo stato della connettività Internet del sistema ospite. Il frammento di codice inizia con la costruzione di uno stack frame, un'operazione standard per la preparazione di una funzione in esecuzione, con i puntatori EBP e ESP che servono rispettivamente come ancoraggi per la base e la cima dello stack.

Dopo aver preparato lo stack, il programma invoca la funzione `InternetGetConnectedState`, un chiaro indicatore che sta verificando la presenza di una connessione Internet attiva. Se la connessione è attiva, il programma sembra eseguire un sotto-programma, che potrebbe essere responsabile della notifica dell'utente o della preparazione per successive operazioni che richiedono accesso a Internet. Questo percorso è segnalato dalla stringa "Success: Internet Connection" e dall'incremento del puntatore dello stack (`add esp, 4`), suggerendo un'operazione di pulizia o preparazione post-controllo.

[TORNA ALL'INDICE](#)

Nel caso in cui la connessione a Internet non sia presente, l'output è un messaggio di errore "Error 1-1: No Internet", seguito da un'operazione che azzerava il registro EAX (`xor eax, eax`), comunemente usata per indicare un valore di ritorno falso o un esito negativo di una funzione.

Questa biforcazione logica nel codice suggerisce che il malware possa avere una dipendenza critica dalla connettività a Internet per eseguire le proprie operazioni nefaste, che possono includere, ma non limitarsi a, il caricamento di dati sensibili a un server controllato dall'attaccante, il download di ulteriori payload malevoli, la connessione a domini compromessi, o l'istituzione di una backdoor per una connessione persistente con l'aggressore.

Possiamo ipotizzare che il malware in questione possa agire come un downloader, che cerca di recuperare componenti aggiuntivi, un trojan che mira a fornire un accesso remoto all'attaccante, o una backdoor che si propone di mantenere un canale nascosto aperto tra l'attaccante e il sistema infetto.

Ipotizzando di aver interpretato correttamente le istruzioni in assembly, il pseudocodice del comportamento del programma potrebbe essere rappresentato nel modo seguente:

```
1 ▾ inizio_funzione:
2     prepara lo stack per le variabili locali
3     stato_connessione = chiama InternetGetConnectedState
4
5 ▾     se stato_connessione è attivo:
6         mostra messaggio "Success: Internet Connection"
7         esegue operazioni che necessitano di connessione
8 ▾     altrimenti:
9         mostra messaggio "Error 1-1: No Internet"
10        gestisce l'assenza di connessione
11
12     pulisce lo stack
13     ritorna l'esito dell'operazione
14 fine_funzione|
```

Basandoci sullo pseudocodice possiamo scrivere una bozza del codice in C: utilizziamo la funzione `InternetGetConnectedState()` dell'API di Windows per determinare se il computer è connesso a Internet. Il risultato viene memorizzato nella variabile `isConnected`. Se il risultato è `TRUE`, il programma stamperà un messaggio che indica il successo della connessione a Internet. Se il risultato è `FALSE`, stamperà un messaggio di errore.

```
1 #include <windows.h>
2 #include <stdio.h>
3
4 int main() {
5     DWORD dwFlags;
6     BOOL isConnected = InternetGetConnectedState(&dwFlags, 0);
7
8     if (isConnected) {
9         printf("Success: Internet Connection\n");
10        //
11    } else {
12        printf("Error 1-1: No Internet\n");
13        //
14    }
15
16    return 0;
17 }
18
```


Grazie!

[TORNA ALL'INDICE](#)