

# MALWARE ANALYSIS

# INDICE

- 1. TRACCIA**
- 2. LIBRERIE IMPORTATE**
- 3. SEZIONI DEL FILE ESEGUIBILE DEL MALWARE**
- 4. FIGURA 3, RISPOSTE QUESITI**
  - 4.1 IDENTIFICARE I COSTRUTTI NOTI (CREAZIONE DELLO STACK, EVENTUALI CICLI,**
  - 4.2 IPOTIZZARE IL COMPORTAMENTO DELLA FUNZIONALITÀ**
  - 4.3 BONUS FARE TABELLA CON SIGNIFICATO DELLE SINGOLE RIGHE DI CODICE**

# TRACCIA

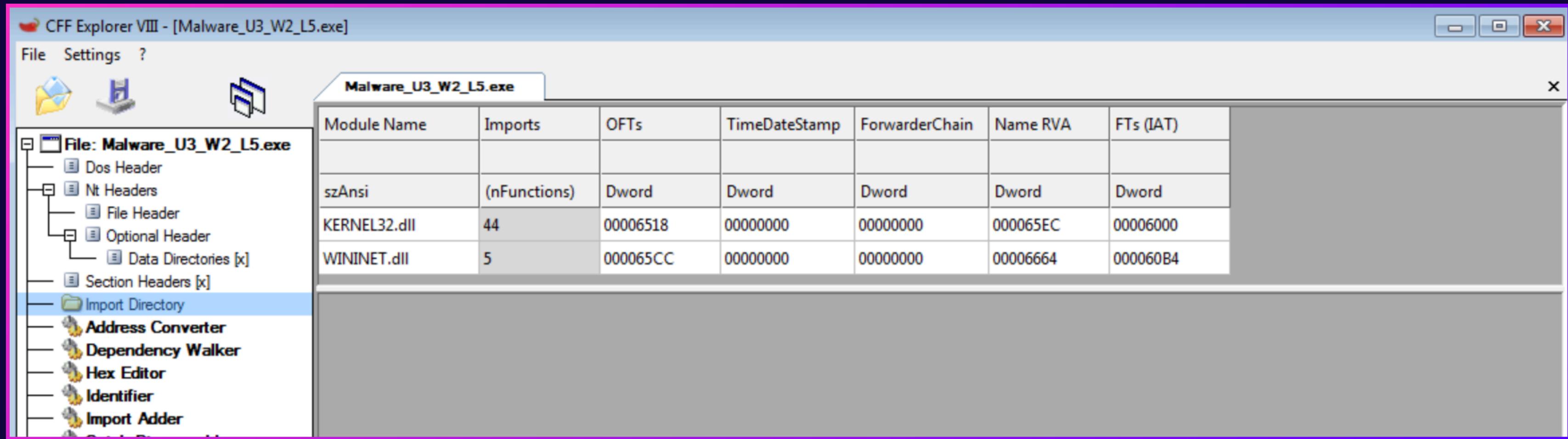
Con riferimento al file Malware\_U3\_W2\_L5 presente all'interno della cartella «Esercizio\_Pratico\_U3\_W2\_L5 » sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

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

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

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli,
4. Ipotizzare il comportamento della funzionalità
5. BONUS fare tabella con significato delle singole righe di codice Esercizio Traccia e requisiti altri costrutti ) implementata assembly

# LIBRERIE UTILIZZATE



Dopo aver aperto il malware dal tool di CFF EXPLORER, dalla sezione “import directory” , vediamo che utilizza 2 librerie:

- **KERNEL32.DLL**: Creazione di processi per eseguire payload dannosi.
- **WININET.dll**: Comunicazione con server di comando e controllo (C&C) per scaricare ulteriori componenti del malware o esfiltrare dati.

# DESCRIZIONE LIBRERIE

**1. KERNEL32.DLL:** Questa libreria contiene funzioni di base del sistema operativo Windows, come la gestione della memoria, operazioni sui file e sui processi, e altre operazioni fondamentali del sistema.

Funzioni comuni includono:

Creazione e gestione di file e directory.

Allocazione e gestione della memoria. Creazione e terminazione di processi e thread.

**2. WININET.dll:** Questa libreria fornisce funzioni per l'accesso a Internet e per la gestione delle comunicazioni di rete tramite protocolli come HTTP e FTP.

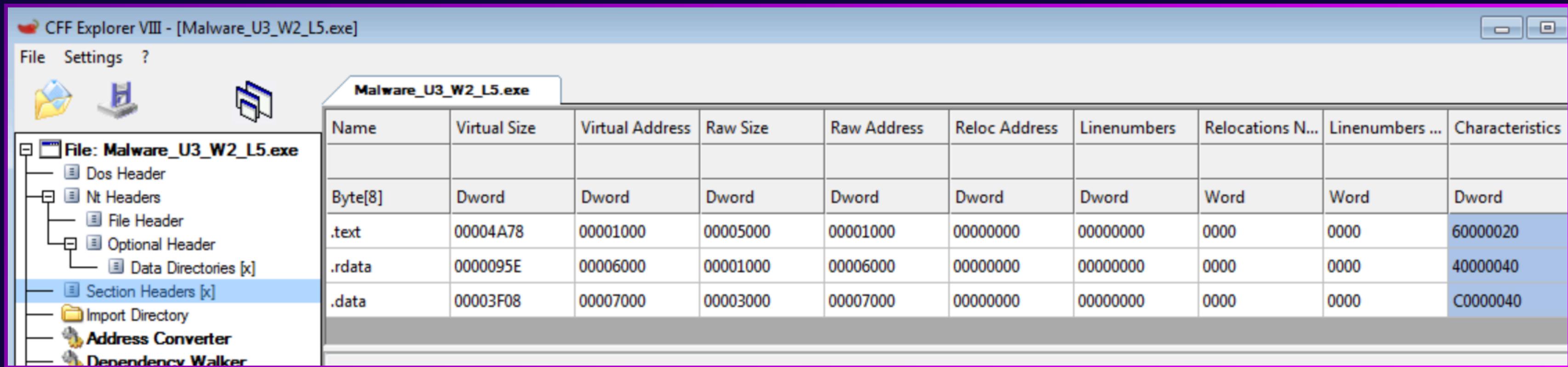
Funzioni comuni includono:

Connessione a server web.

Download e upload di file tramite HTTP/FTP.

Gestione delle sessioni di rete.

# SEZIONI MALWARE



L'immagine mostra la sezione "Section Headers" di un file eseguibile aperto con CFF Explorer. Questa sezione fornisce informazioni dettagliate sulle diverse sezioni del file PE. Le sezioni comuni visualizzate in questo file sono **.text**, **.rdata**, e **.data**.

## .text:

- **Descrizione:** Questa sezione contiene il codice eseguibile del programma. È la sezione principale dove risiede il codice macchina che viene eseguito dalla CPU.
- **Caratteristiche:** Solitamente, questa sezione è leggibile ed eseguibile, ma non scrivibile.

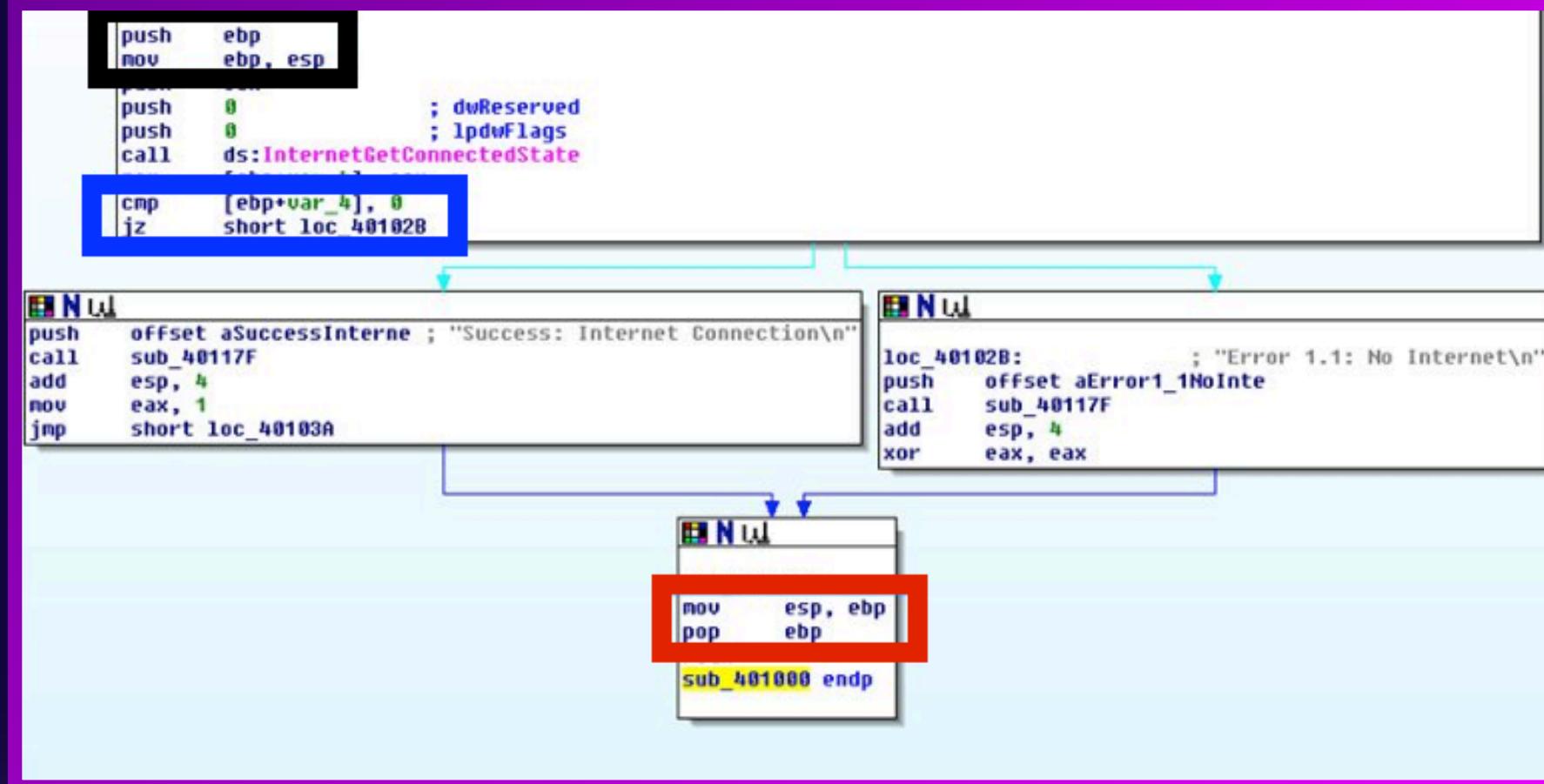
## .rdata:

- **Descrizione:** Questa sezione contiene dati di sola lettura utilizzati dal programma, come stringhe, tabelle di importazione, e altre risorse costanti.
- **Caratteristiche:** Solitamente, questa sezione è leggibile ma non eseguibile né scrivibile.

## .data:

- **Descrizione:** Questa sezione contiene dati di lettura e scrittura utilizzati dal programma, come variabili globali e strutture dati modificate durante l'esecuzione del programma.
- **Caratteristiche:** Questa sezione è leggibile e scrivibile, ma non eseguibile.

# COSTRUTTI NOTI



## Creazione dello stack:

- Descrizione: La creazione dello stack è un processo che inizializza una nuova struttura dati di tipo stack. Uno stack è una collezione ordinata di elementi che segue il principio LIFO (Last In, First Out), ovvero l'ultimo elemento aggiunto è il primo a essere rimosso.
- Dettagli: Questo processo include l'allocazione della memoria necessaria per contenere gli elementi dello stack e l'inizializzazione del puntatore in cima allo stack (spesso chiamato "top") a un valore che indica che lo stack è vuoto.
- Identificazione: Questo processo è rappresentato dal blocco di istruzioni all'interno del rettangolo nero nella figura.

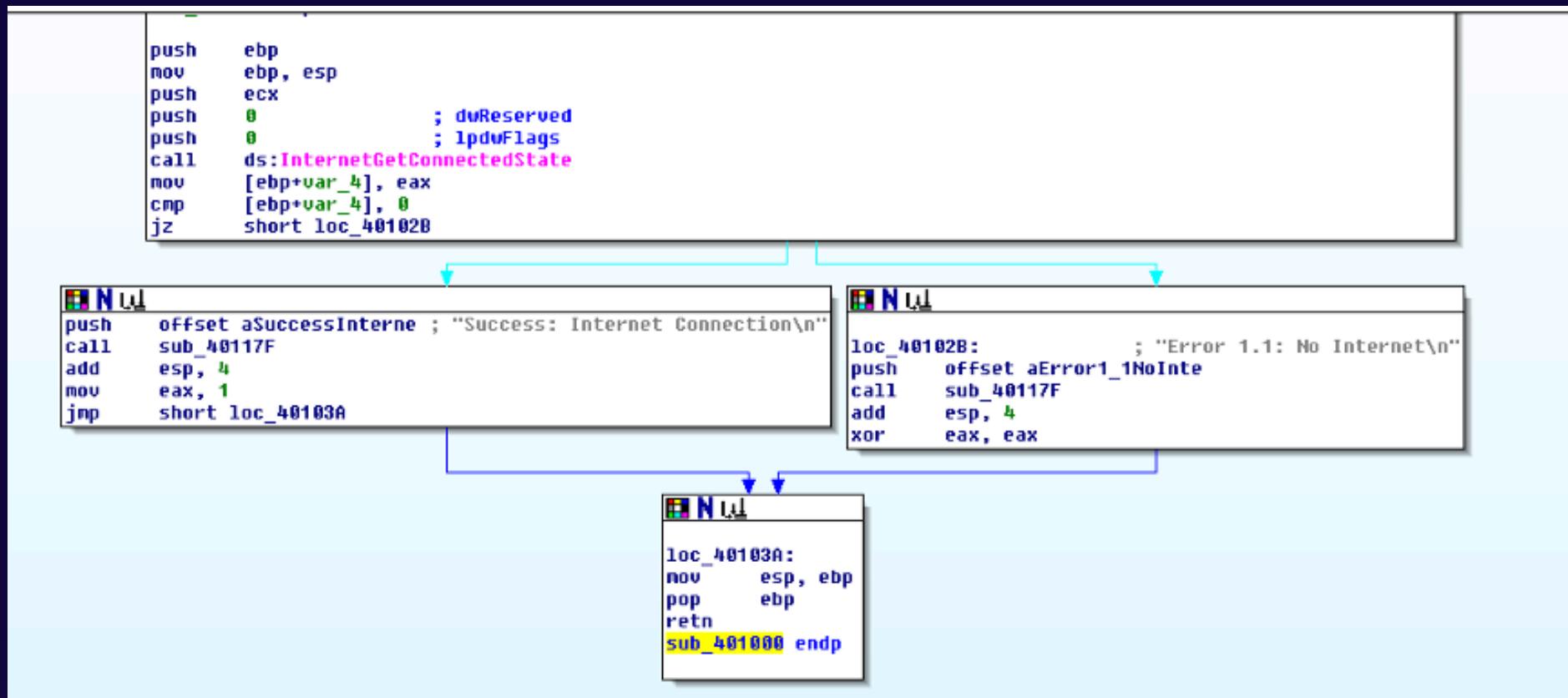
## Costrutto condizionale «IF»:

- Descrizione: Il costrutto condizionale «IF» è una struttura di controllo che permette di eseguire istruzioni diverse a seconda che una condizione specificata sia vera o falsa.
- Dettagli: Se la condizione è vera, viene eseguito un blocco di istruzioni; se la condizione è falsa, può essere eseguito un blocco alternativo (opzionale, solitamente rappresentato dal costrutto «ELSE»). Questo permette di prendere decisioni dinamiche durante l'esecuzione del programma.
- Identificazione: Questo costrutto è identificato dalla coppia di istruzioni contenute nel rettangolo blu nella figura.

## Rimozione dello stack:

- Descrizione: La rimozione dello stack è il processo di liberazione delle risorse associate allo stack, spesso effettuata quando lo stack non è più necessario.
- Dettagli: Questo include la deallocazione della memoria utilizzata per memorizzare gli elementi dello stack e il reset del puntatore in cima allo stack. La rimozione dello stack assicura che le risorse siano restituite al sistema e che non si verifichino perdite di memoria.
- Identificazione: Questo processo è rappresentato dal blocco di istruzioni all'interno del rettangolo rosso nella figura.

# SPIEGAZIONE DEL CODICE



Il codice in questione esegue una verifica dello stato della connessione a Internet utilizzando la funzione `InternetGetConnectedState`. Questa funzione determina se il sistema è connesso a Internet.

Dettagli del funzionamento del codice:

## 1. Verifica dello stato della connessione:

- La funzione `InternetGetConnectedState` viene chiamata per determinare lo stato della connessione a Internet del sistema. Questa funzione ritorna un valore che indica se la connessione è presente o assente.

## 2. Gestione del risultato della verifica:

- Connessione presente:**
  - Se la funzione determina che il sistema è connesso a Internet, il codice stampa un messaggio di successo per notificare che la connessione è attiva.
  - Il registro `eax` viene impostato a 1 per indicare il successo della verifica.
- Connessione assente:**
  - Se la funzione determina che il sistema non è connesso a Internet, il codice stampa un messaggio di errore per notificare l'assenza della connessione.
  - Il registro `eax` viene impostato a 0 per indicare il fallimento della verifica.

# SPIEGAZIONE DEL CODICE

**Push ebp:** Salva il valore corrente del registro ebp sullo stack. Questo è parte del prologo della funzione per preservare il contesto della funzione chiamante.

**Mov ebp, esp:** Copia il valore del registro esp nel registro ebp. Questo imposta il frame pointer al valore corrente dello stack pointer.

**Push ecx:** Salva il valore corrente del registro ecx sullo stack. Questo è fatto per preservare il valore di ecx poiché potrebbe essere usato all'interno della funzione.

**Push 0:** Push del valore 0 sullo stack come parametro dwReserved per la funzione InternetGetConnectedState.

**Push 0:** Push del valore 0 sullo stack come parametro lpdwFlags per la funzione InternetGetConnectedState.

**Call ds:InternetGetConnectedState:** Chiama la funzione InternetGetConnectedState che verifica lo stato della connessione Internet. Il risultato della chiamata sarà nel registro eax.

**Mov [ebp+var\_4], eax:** Memorizza il valore di eax (il risultato della funzione) nella variabile locale [ebp+var\_4].

**Cmp [ebp+var\_4], 0:** Confronta il valore memorizzato in [ebp+var\_4] con 0.

**Jz short loc\_40102B:** Salta alla locazione loc\_40102B se il confronto precedente è uguale a 0 (cioè, non c'è connessione Internet).

# CONNESIONE RIUSCITA

- **Push offset aSuccessInterne:** Push dell'indirizzo della stringa "Success: Internet Connection\n" sullo stack.
- **Call sub\_40117F:** Chiama la funzione sub\_40117F per gestire il messaggio di successo (presumibilmente per stamparlo o loggarlo).
- **Add esp, 4:** Aggiunge 4 a esp per ripulire lo stack dal parametro appena pushato.
- **Mov eax, 1:** Imposta il registro eax a 1 per indicare successo.
- **Jmp short loc\_40103A:** Salta alla locazione loc\_40103A per terminare la funzione.

# CONNESIONE FALLITA

- **Loc\_40102B:** Etichetta di salto per il caso in cui non c'è connessione Internet.
- **Push offset aError1\_1\_NoInte:** Push dell'indirizzo della stringa "Error 1.1: No Internet\n" sullo stack.
- **Call sub\_40117F:** Chiama la funzione sub\_40117F per gestire il messaggio di errore.
- **Add esp, 4:** Aggiunge 4 a esp per ripulire lo stack dal parametro appena pushato.
- **Xor eax, eax:** Imposta eax a 0 per indicare fallimento (usando xor per azzerare il registro).

# EPILOGO DELLA FUNZIONE

- **loc\_40103A:** Etichetta di salto per il termine della funzione.
- **mov esp, ebp:** Ripristina il valore originale dello stack pointer (esp) al valore salvato in ebp.
- **pop ebp:** Ripristina il valore originale di ebp dal valore salvato sullo stack.
- **retn:** Ritorna dalla funzione, usando l'indirizzo di ritorno che era stato salvato sullo stack.

```
#include <stdio.h>
#include <windows.h> // Include i parametri necessari per InternetGetConnectedState

void handle_message(const char *message) {
    printf("%s", message);
}

int check_internet_connection() {
    DWORD dwFlags = 0;
    BOOL isConnected = InternetGetConnectedState(&dwFlags, 0);

    if (isConnected) {
        handle_message("Success: Internet Connection\n");
        return 1; // Success
    } else {
        handle_message("Error 1.1: No Internet\n");
        return 0; // Error
    }
}

int main() {
    int result = check_internet_connection();
    return result;
}
```

Open

ANTONIO PERNA  
FABIO NOBILI



THANK YOU