ANALISI STATICA E DINAMICA

PROGETTO Rafael mango

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, altri costrutti)

4. Ipotizzare il comportamento della funzionalità implementata

5. BONUS fare tabella con significato delle singole righe di codice assembly

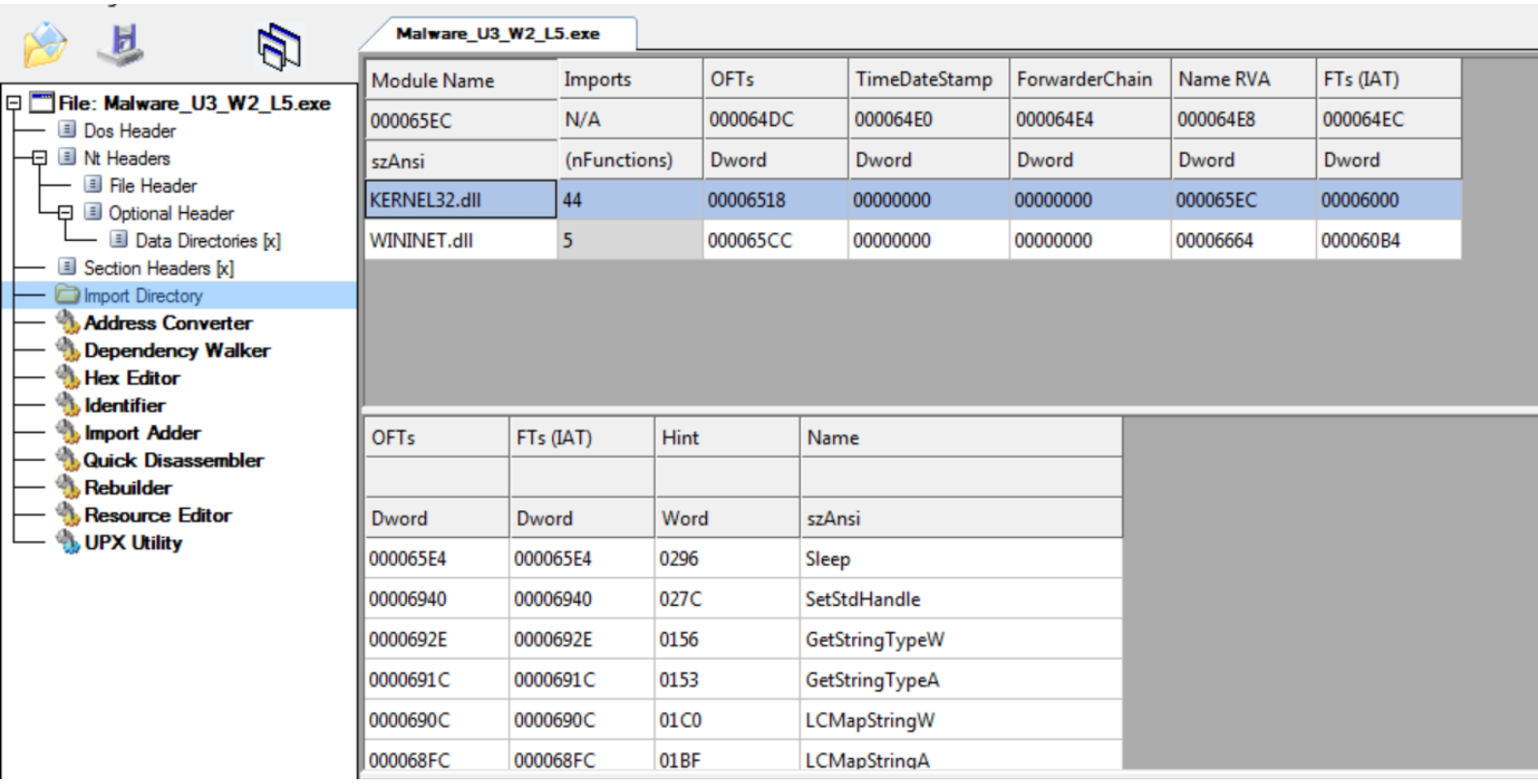
---

---

1 Quali librerie vengono importate dal file eseguibile?

Librerie
Per determinare le librerie utilizzate dal file eseguibile, è possibile esaminare il suo contenuto

tramite CFF Explorer. All'interno del file, troviamo che le librerie coinvolte includono Kernel32.dll e WININET.dll. Come possiamo vedere in figura .



Vediamo che ci sono tante funzioni nella libreria Kernel32.dll elencheremo le prime 5 funzioni :

sleep:La funzione Sleep è una funzione di sistema che permette di sospendere l'esecuzione di un programma per un certo periodo di tempo, specificato in millisecondi. Quando la funzione Sleep viene chiamata, il thread corrente viene messo in uno stato di "sospensione" per il tempo specificato, permettendo ad altri thread di essere eseguiti durante questo intervallo. Dopo il tempo specificato, il thread riprende la sua esecuzione normale.

Questa funzione è spesso utilizzata per aggiungere ritardi o per controllare la frequenza di esecuzione di un programma, ad esempio nel caso di animazioni, timeout, o per ridurre l'uso della CPU quando il thread non ha nulla da fare per un certo periodo di tempo.

setStdHandle : La funzione SetStdHandle nella libreria Kernel32.dll permette di cambiare il modo in cui un programma gestisce l'input, l'output e gli errori standard. Ad esempio, si può utilizzare per dire al programma di leggere o scrivere da un file anziché dalla tastiera o verso uno schermo.

GetStringType:La funzione GetStringType nella libreria Kernel32.dll è utilizzata per ottenere informazioni sul tipo di carattere per una stringa di testo. Questa funzione prende in input una stringa e restituisce informazioni sui tipi di carattere presenti in essa. Ad esempio, può determinare se i caratteri sono lettere, numeri, spazi bianchi o caratteri di punteggiatura. Questo tipo di

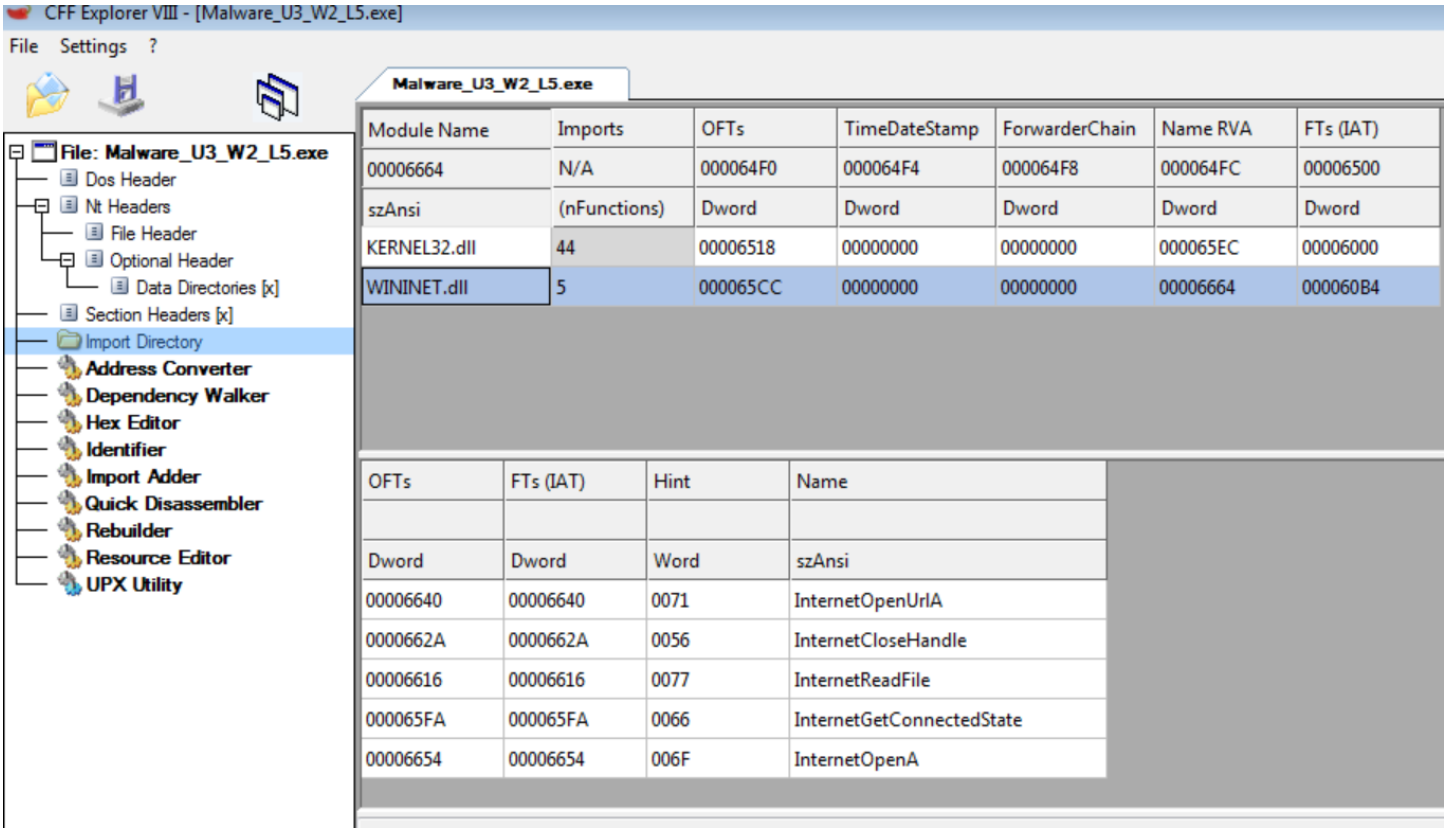
informazioni può essere utile per eseguire operazioni di analisi o manipolazione del testo in un programma.

GetStringTypeA:
La funzione GetStringTypeA nella libreria Kernel32.dll è utilizzata per ottenere informazioni sul tipo di carattere per una stringa di testo codificata in formato ANSI. Questa funzione accetta una stringa di testo in formato ANSI come input e restituisce informazioni sui tipi di carattere presenti in essa. Queste informazioni possono includere se i caratteri sono lettere, numeri, spazi bianchi o caratteri di punteggiatura. In sostanza, GetStringTypeA analizza la struttura dei caratteri nella stringa e fornisce informazioni dettagliate sul tipo di ciascun carattere.

LCMapStringW:La funzione LCMapStringW nella libreria Kernel32.dll è utilizzata per eseguire la mappatura dei caratteri di una stringa in base alle regole di confronto specificate dalla localizzazione corrente. Questa funzione accetta una stringa di testo wide (Unicode) come input e applica la mappatura dei caratteri in base alle regole di confronto specificate.

Le regole di confronto includono la conversione tra maiuscole e minuscole, la rimozione o la normalizzazione dei diacritici, la conversione dei caratteri di metacarattere e altre trasformazioni di caratteri basate sulla localizzazione. Questa funzione è utile per operazioni di confronto di stringhe che richiedono la sensibilità alla lingua e alla localizzazione, come ad esempio nell'ordinamento e nella ricerca di stringhe in diverse lingue.

2



come vediamo in figura in alto nella libreria WININET.dll ci sono solo 5 funzioni a differenza la libreria di Kernel32.dll

Come in precedenza andiamo a spiegare le varie funzioni

InternetOpenUrlA:
La funzione InternetOpenUrlA nella libreria WININET.dll è utilizzata per aprire una connessione a un URL specifico attraverso Internet. Questa funzione accetta come input un gestore di sessione Internet, ottenuto tramite la funzione InternetOpen, e un URL da aprire. Una volta chiamata, questa funzione avvia una richiesta per l'URL specificato e restituisce un gestore di connessione Internet per l'URL richiesto.

In sostanza, InternetOpenUrlA consente di aprire una connessione a una risorsa su Internet (come una pagina web, un file remoto, o un servizio web) e di ottenere un gestore per gestire la comunicazione con tale risorsa. Questo gestore può poi essere utilizzato per inviare richieste HTTP, ricevere dati e interagire con la risorsa online.

InternetCloseHandle:La funzione InternetCloseHandle nella libreria WININET.dll viene utilizzata per chiudere un handle associato a una risorsa Internet precedentemente aperta tramite le funzioni della libreria WININET.dll.

Quando si apre una connessione Internet o si effettua una richiesta HTTP utilizzando le funzioni come InternetOpenUrlA, viene restituito un gestore (handle) che rappresenta questa connessione o richiesta. Dopo aver completato tutte le operazioni desiderate con questa risorsa Internet, è importante chiudere l'handle associato per rilasciare le risorse di sistema allocate e prevenire eventuali perdite di memoria.

Quindi, InternetCloseHandle viene chiamata per chiudere l'handle associato a una risorsa Internet, rilasciando le risorse allocate per gestire questa risorsa e consentendo al sistema di liberare eventuali risorse utilizzate per la connessione o la richiesta Internet.

InternetReadFile:La funzione InternetReadFile nella libreria WININET.dll viene utilizzata per leggere dati da una risorsa Internet, come ad esempio un file o una pagina web, che è stata precedentemente aperta tramite la funzione InternetOpenUrlA. Questa funzione legge un certo numero di byte dalla risorsa Internet e li memorizza in un buffer fornito dall'utente.

In particolare, InternetReadFile accetta come input un handle della risorsa Internet da cui leggere i dati, il buffer in cui memorizzare i dati letti e la dimensione del buffer. Quando viene chiamata, la funzione legge un numero di byte dalla risorsa Internet e li copia nel buffer fornito. Restituisce TRUE se la lettura ha avuto successo e FALSE in caso di errore.

Questa funzione è comunemente utilizzata quando si desidera scaricare dati da una risorsa Internet, come il testo di una pagina web o un file remoto, per elaborarli localmente all'interno di un programma.

InternetGetConnectedState:La funzione InternetGetConnectedState nella libreria WININET.dll viene utilizzata per determinare lo stato della connessione Internet sul sistema locale. Questa funzione restituisce informazioni riguardanti la connessione Internet, come ad esempio se il computer è connesso o meno a Internet e il tipo di connessione attiva (ad esempio, una connessione via modem, LAN, Wi-Fi, ecc.).

La funzione InternetGetConnectedState restituisce un valore booleano che indica se il sistema è connesso a Internet. Inoltre, se viene fornito un parametro aggiuntivo, può restituire informazioni più dettagliate sul tipo di connessione attiva.

Questa funzione è utile per i programmi che devono adattare il loro comportamento in base alla disponibilità di una connessione Internet, come ad esempio applicazioni che scaricano dati da risorse online o che richiedono una connessione per funzionare correttamente.

InternetOpenA: La funzione InternetOpenA nella libreria WININET.dll viene utilizzata per inizializzare una sessione di comunicazione Internet nel contesto di un'applicazione. Questa funzione apre una "sessione Internet" che può essere utilizzata per eseguire operazioni su Internet, come l'apertura di connessioni a risorse web, l'invio di richieste HTTP e altro ancora.

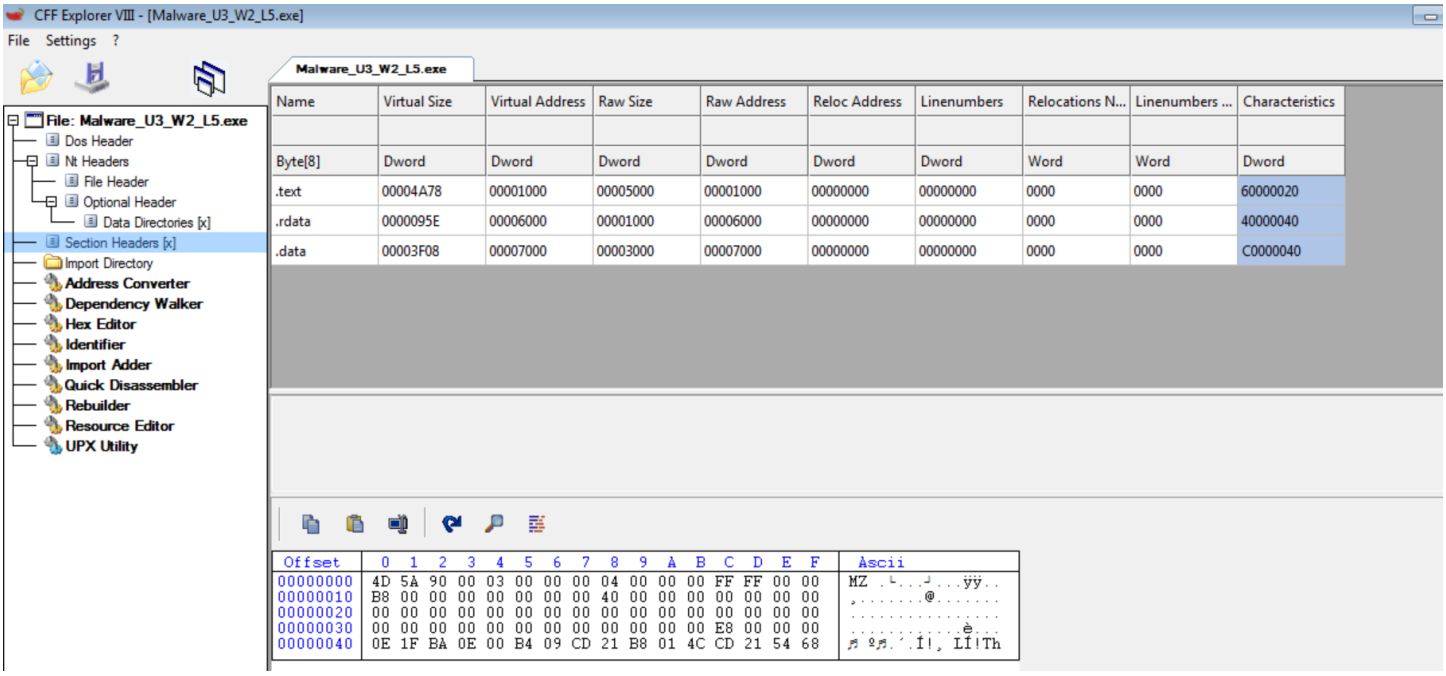
In particolare, InternetOpenA accetta come input alcuni parametri, come ad esempio il nome dell'applicazione, il tipo di accesso desiderato (ad esempio, accesso diretto o tramite proxy), e altre opzioni di configurazione. Restituisce quindi un "handle" (o gestore) che rappresenta la sessione Internet appena aperta.

Questa funzione è solitamente la prima chiamata effettuata da un'applicazione che desidera interagire con risorse su Internet utilizzando la libreria WININET.dll. Una volta aperta la sessione Internet, è possibile utilizzare altri metodi e funzioni della libreria per eseguire operazioni di rete, come l'apertura di URL tramite InternetOpenUrlA, la lettura di dati tramite InternetReadFile, e altro ancora.

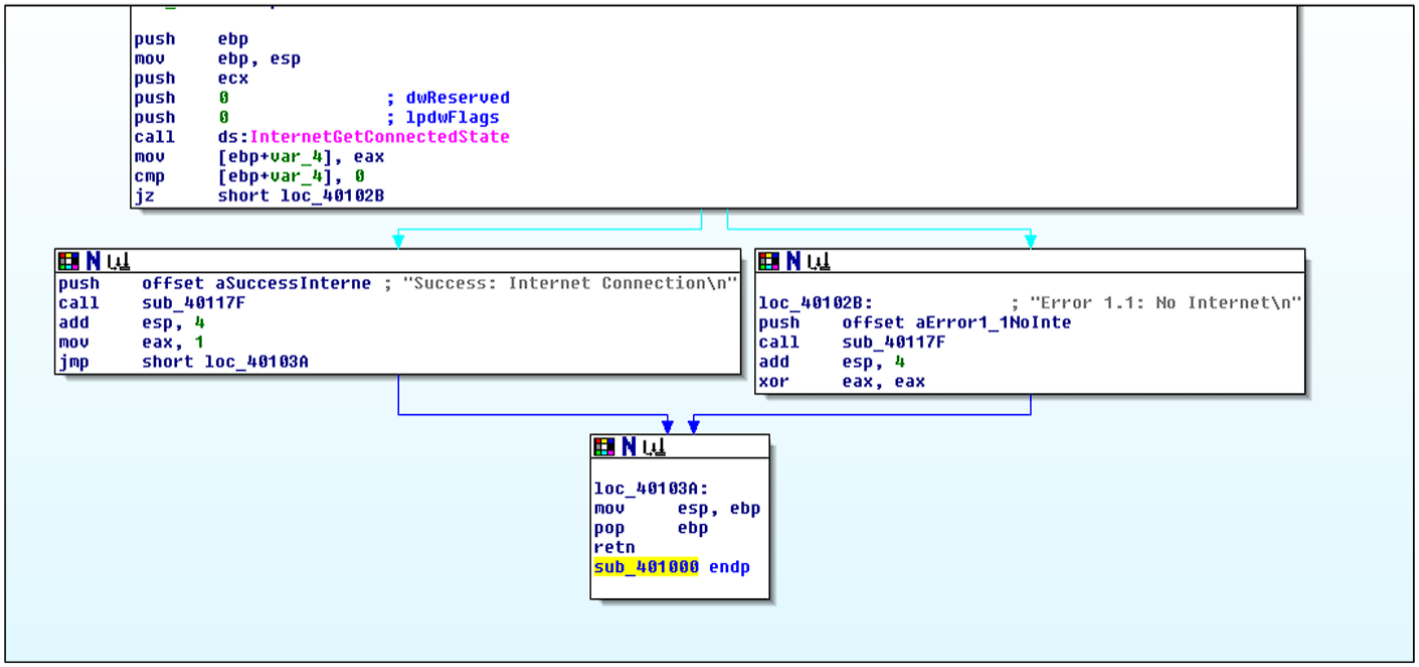
---

2 Quali sono le sezioni di cui si compone il file eseguibile del malware?

Possiamo verificare le sezioni del file esaminando la sezione degli header delle sezioni utilizzando CFF Explorer. Le sezioni individuate includono: .test, .rdata e .data. come in figura .



Con riferimento alla figura in slide :



3 Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti) :

-Le istruzioni fornite abbreviano il processo di creazione del frame dello stack all'inizio di una funzione in assembly x86. La sequenza abbreviata è:

push ebp: Salva il valore attuale di EBP nello stack.

mov ebp, esp: Imposta EBP uguale a ESP.

Questo serve a creare un frame dello stack per la funzione corrente, semplificando l'accesso ai parametri e alle variabili locali all'interno della funzione stessa.

- cmp [ebp+var_4], 0

  jz short loc_40102B

Questo frammento di codice assembly ci mostra un costrutto condizionale (IF), Il costrutto condizionale formato dalle istruzioni cmp e jz è equivalente a un'istruzione if nei linguaggi di programmazione ad alto livello.

In pseudocodice, potremmo rappresentare il costrutto condizionale come:

if (valore in [ebp+var_4] è uguale a 0) {

salta a loc_40102B

}

Quindi, se il valore memorizzato in [ebp+var_4] è zero, il controllo del programma viene trasferito a loc_40102B. Altrimenti, il programma continuerà con l'istruzione successiva dopo il blocco condizionale.

.Il secondo costrutto condizionale è implicito nella chiamata alla funzione

call ds:internetGetConnectedState

Questa chiamata di funzione è seguita dall'istruzione mov [ebp+var_4], eax, che memorizza il risultato della chiamata (presumibilmente lo stato della connessione Internet) nella variabile locale [ebp+var_4]. Successivamente, l'istruzione cmp [ebp+var_4], 0 e il salto condizionale jz determinano se il risultato della funzione è zero o meno, eseguendo un'azione in base a questo risultato.

In entrambi i casi, il costrutto condizionale è simile all'istruzione if nei linguaggi di programmazione ad alto livello e viene utilizzato per eseguire determinate azioni in base al valore di una condizione specifica.

-**mov esp, ebp**

pop ebp

Come possiamo vedere da questo breve frammento di codice assembly possiamo dire che si tratta di una rimozione dello stack ,**mov esp, ebp**: Questa istruzione imposta il puntatore stack (ESP) uguale al registro base del puntatore (EBP). Questa operazione sposta il puntatore stack all'indirizzo memorizzato in EBP, che è la posizione iniziale del frame dello stack della funzione chiamante.

pop ebp: Questa istruzione rimuove il valore superiore dello stack e lo assegna al registro base del puntatore (EBP). Poiché EBP è spesso utilizzato per accedere alle variabili locali e ai parametri della funzione corrente, ripristinare il valore precedente di EBP consente al programma di accedere correttamente ai dati della funzione chiamante dopo il ritorno.

In sintesi, queste istruzioni sono comunemente utilizzate per pulire il frame dello stack e ripristinare il contesto dello stack al livello precedente prima dell'invocazione della funzione corrente.

retn: l'istruzione retn termina l'esecuzione della funzione corrente e restituisce il controllo al punto del programma da cui la funzione è stata chiamata.

Quando l'istruzione retn viene eseguita, il processore estrae l'indirizzo di ritorno dalla cima dello stack. Questo indirizzo di ritorno è l'indirizzo della prossima istruzione da eseguire dopo la chiamata della funzione. Una volta ottenuto l'indirizzo di ritorno, il controllo del programma viene trasferito a tale

indirizzo, facendo continuare l'esecuzione del programma dalla posizione successiva all'istruzione di chiamata della funzione.

In breve, retn è l'istruzione che gestisce il ritorno dal sottoprogramma al programma principale, utilizzando l'indirizzo di ritorno memorizzato nello stack durante la chiamata della funzione.

In sintesi, queste istruzioni sono comunemente utilizzate per pulire il frame dello stack e ripristinare il contesto dello stack al livello precedente prima dell'invocazione della funzione corrente.

---

4 Ipotizzare il comportamento della funzionalità implementata

Dalla lettura del codice e delle istruzioni fornite, possiamo fare delle supposizioni sul modo in cui la funzione potrebbe comportarsi. In sostanza, sembra che il codice stia cercando di verificare se il sistema è attualmente connesso a Internet

Chiamata a InternetGetConnectedState:

Il codice inizia con la preparazione dei parametri necessari per la chiamata alla funzione di sistema InternetGetConnectedState. Questa funzione viene chiamata per verificare lo stato della connessione Internet.

La funzione inizia preparando i parametri necessari per la chiamata alla funzione di sistema InternetGetConnectedState, che serve per verificare lo stato della connessione Internet. I parametri dwReserved e lpdwFlags vengono entrambi impostati a zero.

Il risultato della chiamata a InternetGetConnectedState viene memorizzato nella variabile locale [ebp+var_4] tramite l'istruzione mov [ebp+var_4], eax.

Successivamente, il codice esegue un controllo condizionale per verificare se il risultato della chiamata è zero o meno. Se il risultato è zero, il programma salta a loc_40102B, probabilmente per gestire il caso in cui la connessione Internet non sia disponibile. Altrimenti, se il risultato è diverso da zero, il programma prosegue con l'esecuzione normale.

Alla fine della funzione, vengono eseguite le istruzioni per ripristinare il frame dello stack e tornare al chiamante. Se il valore restituito da InternetGetConnectedState è zero, la funzione stampa a schermo "No internet". Se il valore restituito è diverso da zero, stampa "Success: internet connection". Infine, la funzione completa la sua esecuzione.

---

5 BONUS fare tabella con significato delle singole righe di codice assembly

Righe di Codice Assembly	Significato
push ecx	Salva il valore corrente del registro ECX nello stack.
push 0	Mette lo zero nello stack.
push 0	Mette lo zero nello stack.
call ds:internetGetConnectedState	Chiama la funzione di sistema InternetGetConnectedState.
mov [ebp+var_4], eax	Memorizza il risultato della funzione nella variabile locale [ebp+var_4].
cmp [ebp+var_4], 0	Confronta il valore memorizzato in [ebp+var_4] con zero.
jz short loc_40102B	Salta a loc_40102B se il risultato della comparazione è zero (cioè, se la connessione è assente).
mov esp, ebp	Imposta il puntatore stack (ESP) uguale al registro base del puntatore (EBP).
pop ebp	Ripristina il valore precedente del registro base del puntatore (EBP) dalla pila.
retn	Termina la funzione corrente e restituisce il controllo al chiamante.

push offset aSuccessInterne	Mette l'offset della stringa "Success: internet connection" nello stack
call sub_40117F	Chiama la subroutine sub_40117F.
add esp, 4	Aggiunge 4 byte a ESP per liberare lo spazio nello stack dopo la chiamata di subroutine.
mov eax, 1	Mette il valore 1 nel registro EAX.
jmp short loc_40103A	Salta incondizionatamente a loc_40103A.
loc_401012B	Etichetta per una posizione nel codice.
push offset aError1_1Nointe	Mette l'offset della stringa "Error: no internet" nello stack.
call sub_40117F	Chiama la subroutine sub_40117F.
add esp, 4	Aggiunge 4 byte a ESP per liberare lo spazio nello stack dopo la chiamata di subroutine.
xor eax, eax	Esegue un'operazione di XOR tra il registro EAX e se stesso, azzerando così il registro EAX.

Questa tabella offre una panoramica delle istruzioni e delle azioni svolte da ciascuna riga di codice assembly.

Parole Chiave

-funzione in assembly x86:Una funzione assembly x86 è un piccolo pezzo di codice che esegue un'azione specifica all'interno di un programma. Utilizza istruzioni di basso livello per eseguire operazioni come calcoli, accesso alla memoria e gestione del flusso di esecuzione. Le funzioni possono essere chiamate da altre parti del programma e possono anche chiamarsi a vicenda. Sono spesso utilizzate per rendere più modulare e organizzato il codice di un programma.

-**che cosa è il linguaggio assembly** : Il linguaggio assembly è un linguaggio di programmazione di basso livello che fornisce un'interfaccia diretta con l'hardware del computer. È una rappresentazione simbolica delle istruzioni di un processore, utilizzando mnemonici e simboli per operazioni e registri. Ogni istruzione assembly corrisponde a un'operazione specifica eseguita dal processore, come il caricamento di dati nei registri, operazioni aritmetiche e logiche, trasferimento di dati tra memoria e registri, e gestione del flusso di esecuzione del programma. Poiché è strettamente legato all'architettura del processore, il linguaggio assembly è altamente efficiente ma richiede una conoscenza dettagliata dell'hardware e delle istruzioni specifiche del processore target. È comunemente usato per ottimizzare codice critico per le prestazioni o per scrivere driver di sistema e codice a livello di microcontrollore.


THANK YOU