# S10L5

Con riferimento al file "MALWARE_U3_W2_L5" situato all'interno della cartella "ESERCIZIO_PRATICO_U3_W2_L5" sul desktop della macchina virtuale dedicata all'analisi dei malware, rispondiamo alle seguenti domande:

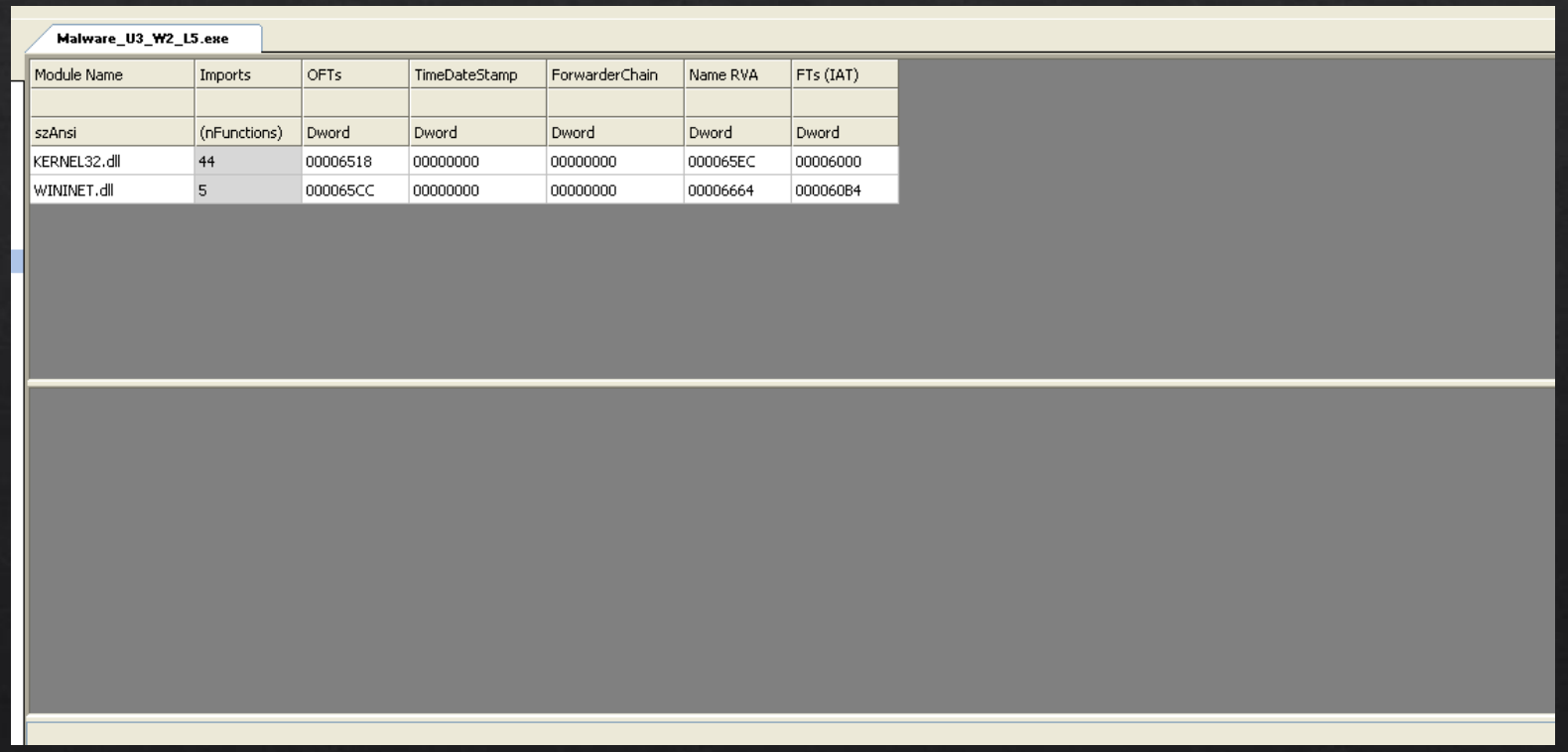
1. Quali librerie vengono importate dal file eseguibile?

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

Nel CFF Explorer, è possibile trovare le librerie importate del file eseguibile accedendo alla sezione "Import Directory" dal pannello principale sulla sinistra. Le librerie importate dal file eseguibile includono:

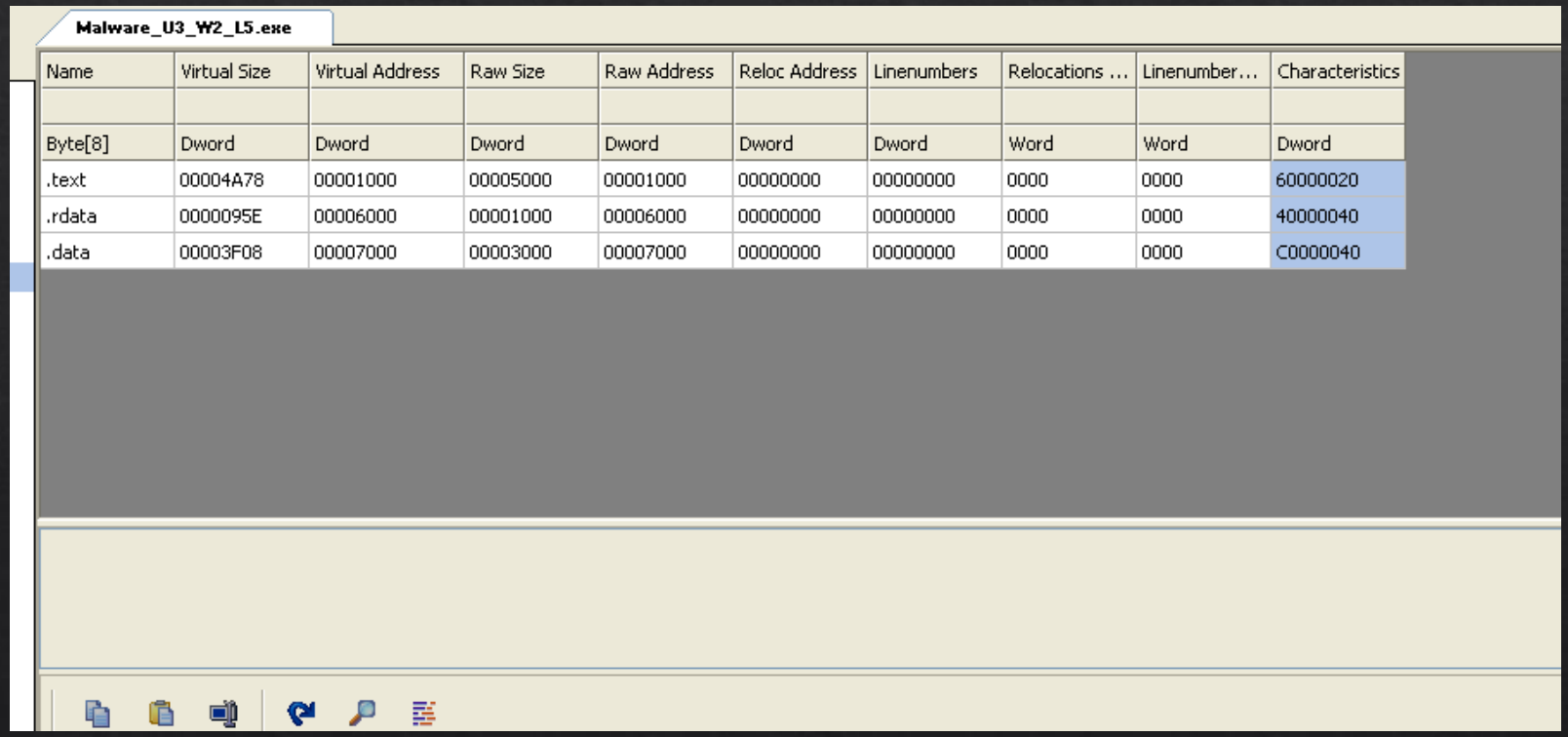
KERNEL32.DLL: Questa libreria, estremamente comune, contiene le funzioni principali per interagire con il sistema operativo. Queste funzioni includono operazioni come la manipolazione dei file e la gestione della memoria.

WININET.DLL: Questa libreria contiene le funzioni necessarie per l'implementazione di alcuni protocolli di rete, tra cui HTTP, FTP e NTP.

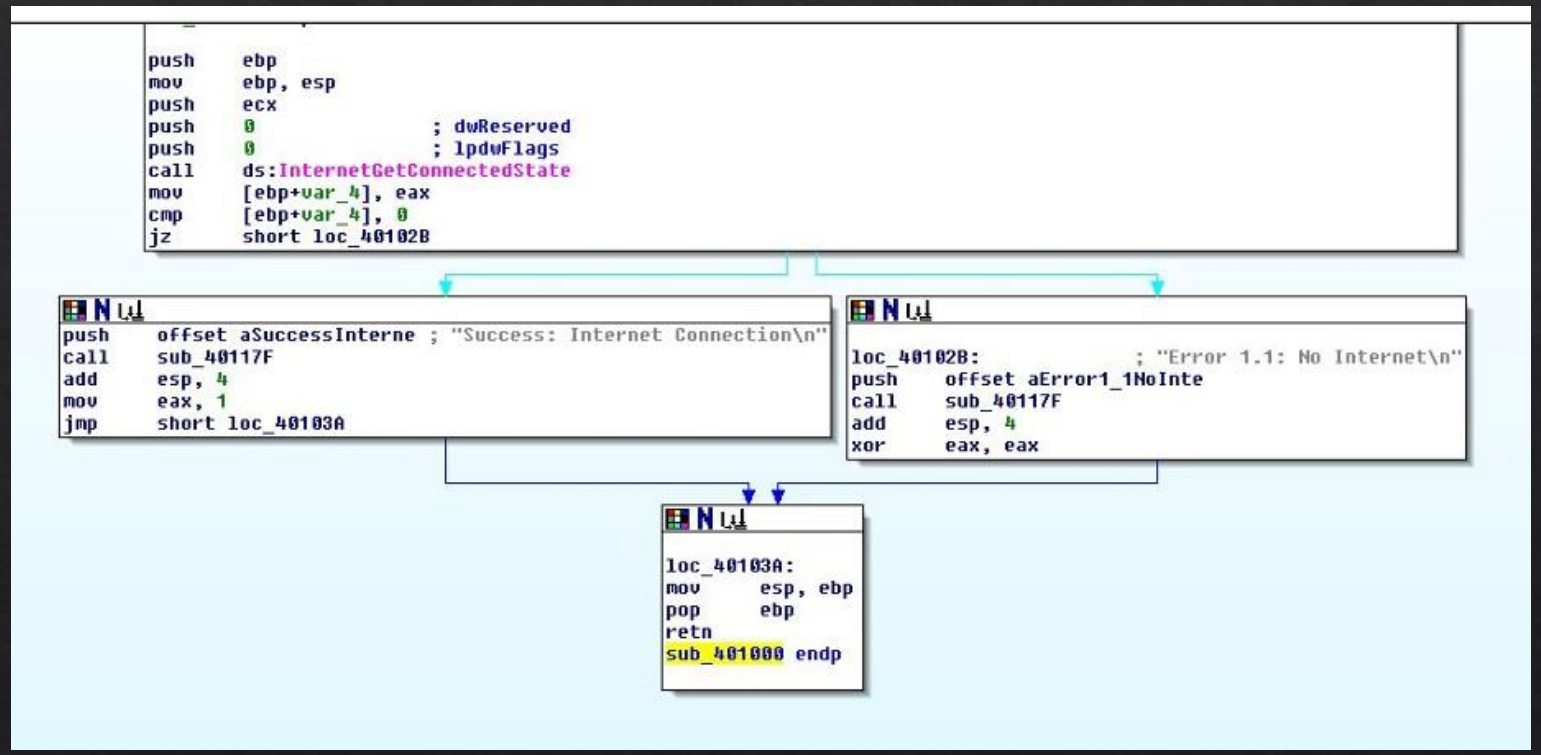


**.TEST:** Un termine generico che viene usato per indicare una sezione di verifica o potrebbe essere specifico per un .RDATA:. Questa sezione principalmente contiene dati di sola lettura, come stringhe di costanti e tabelle di ricerca, che l'eseguibile utilizza durante l'esecuzione in un contesto o applicazione specifica.

**.DATA:** Questa sezione contiene dati globali e statici che possono essere modificati durante l'esecuzione del programma.



# PRENDENDO IN CONSIDERAZIONE LA FIGURA RISPONDIAMO AI SEGUENTI QUESITI: IDENTIFICARE I COSTRUTTI NOTI (CREAZIONE DELLO STACK, EVENTUALI CICLI, COSTRUTTI) IPOTIZZARE IL COMPORTAMENTO DELLA FUNZIONALITÀ IMPLEMENTATA



- Il frammento di codice assembly sembra essere parte di una routine che esegue una verifica dello stato della connessione internet. Se la connessione è attiva, viene visualizzato un messaggio di conferma; altrimenti, viene visualizzato un messaggio di errore.

- Il codice utilizza alcune istruzioni comuni, come la gestione dello stack con le istruzioni **push ebp** e **mov ebp, esp**, che servono per salvare e impostare il puntatore dello stack. Inoltre, usa push e call per passare parametri e chiamare subroutine. Ad esempio**, push ecx** e push offset **lpdwFlags** vengono utilizzati per passare parametri a **InternetGetConnectedState**, una funzione chiamata con call ds:InternetGetConnectedState. Questa funzione restituisce un valore booleano in eax, che indica la presenza o l'assenza di una connessione internet.

- Altri costrutti noti includono **cmp** e **jz** per confrontare valori e saltare a seconda del risultato. Ad esempio, cmp [**ebp+var_4**], 0 viene utilizzato per confrontare il valore restituito da InternetGetConnectedState con zero, e jz short **loc_40102B** per saltare a loc_40102B se il valore è zero, indicando l'assenza di connessione internet.

- È presente anche l'uso di push offset e call per stampare messaggi di stringa. Ad esempio, push offset **aSuccessInterne** e **push offset** aError1NoInte vengono utilizzati per passare i messaggi di successo e di errore alla subroutine **sub_40117F**, chiamata con call **sub_40117F**, che probabilmente stampa i messaggi sullo schermo o su un file.

- Infine, viene utilizzata l'istruzione xor eax, eax per azzerare il registro eax, che equivale a mov eax, 0, ma è più efficiente. Questa istruzione viene utilizzata per impostare il valore di ritorno della funzione a zero.

L'impiego di jmp per eseguire un salto incondizionato a un altro indirizzo è evidente nel codice, che utilizza jmp short loc_40103A per bypassare il blocco di codice relativo alla stampa del messaggio di errore e saltare direttamente alla fine della funzione.

La pulizia dello stack è eseguita tramite le istruzioni mov esp, ebp e pop ebp. Queste istruzioni sono necessarie per ripristinare il valore del registro esp e recuperare il valore del registro ebp dallo stack, assicurando così un corretto ripristino dello stato dello stack.

Il termine della funzione è segnato dall'utilizzo di retn, il quale indica il ritorno al chiamante. Questa istruzione salta all'indirizzo salvato sullo stack dalla precedente chiamata (call), consentendo al flusso del programma di proseguire da dove è stato invocato il codice.

L'ipotesi sul comportamento della funzionalità implementata può essere formulata nel seguente modo: il codice chiama la funzione InternetGetConnectedState, la quale restituisce un valore booleano in eax che indica la presenza o l'assenza di una connessione internet. Se il valore restituito è diverso da zero, ciò indica la presenza di una connessione internet, e il codice salta all'etichetta loc_40101F, dove viene visualizzato il messaggio "Success: Internet Connection", seguito dal termine della funzione con un valore di ritorno pari a zero. Al contrario, se il valore restituito è zero, segnalando l'assenza di connessione internet, il codice salta all'etichetta loc_40102B, dove viene stampato il messaggio "Error1.1: No Internet", seguito anch'esso dal termine della funzione con un valore di ritorno pari a zero.