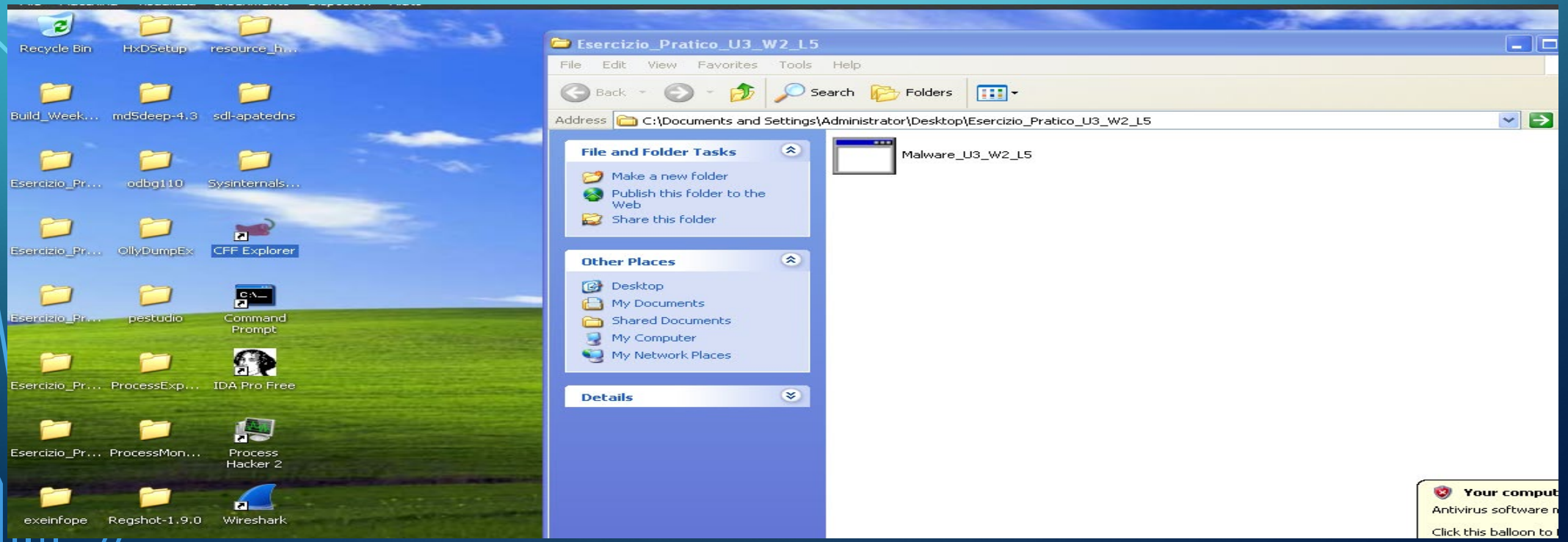
S10L5

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: QUALI LIBRERIE VENGONO IMPORTATE DAL FILE ESEGUIBILE?

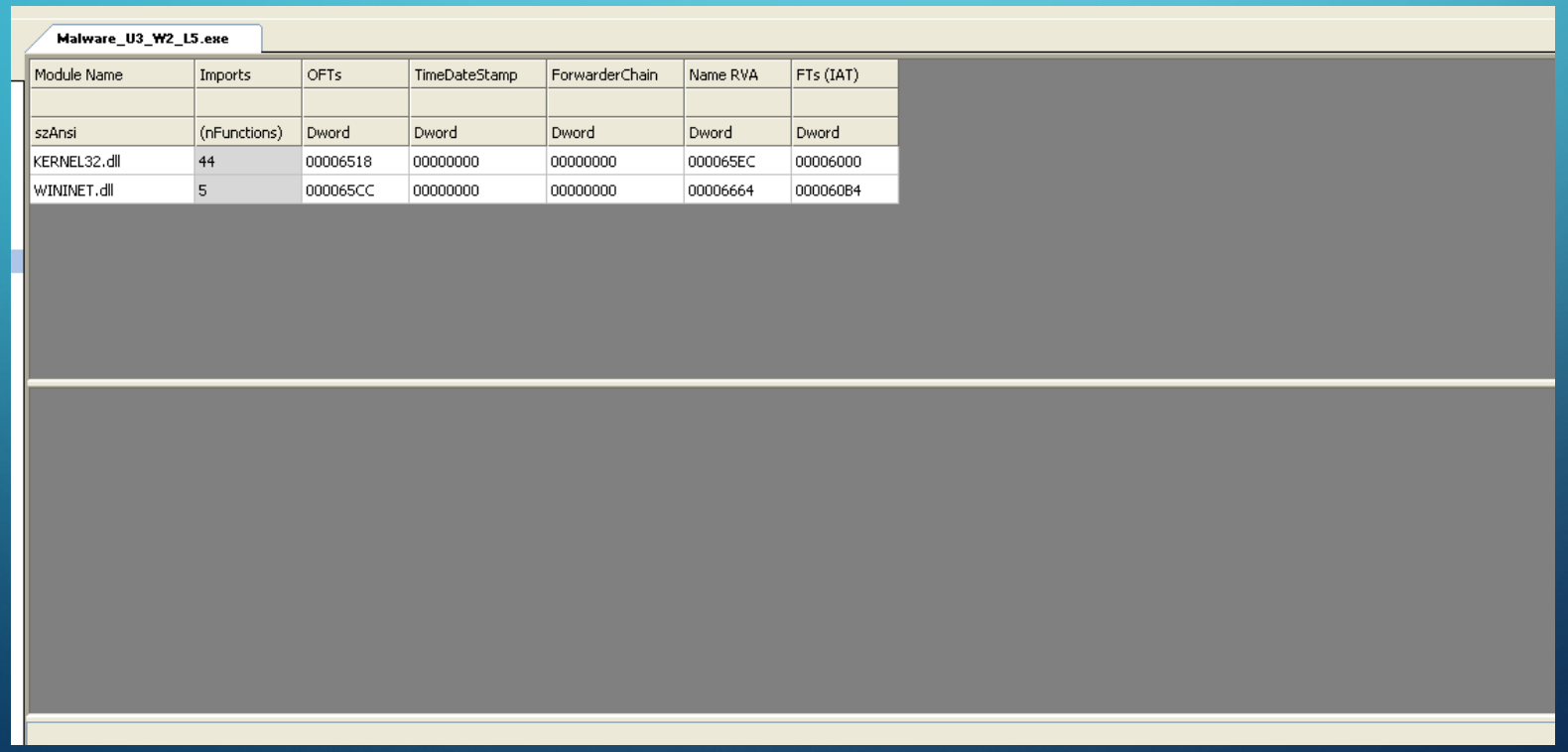
QUALI SONO LE SEZIONI DI CUI SI COMPONE IL FILE ESEGUIBILE DEL MALWARE?



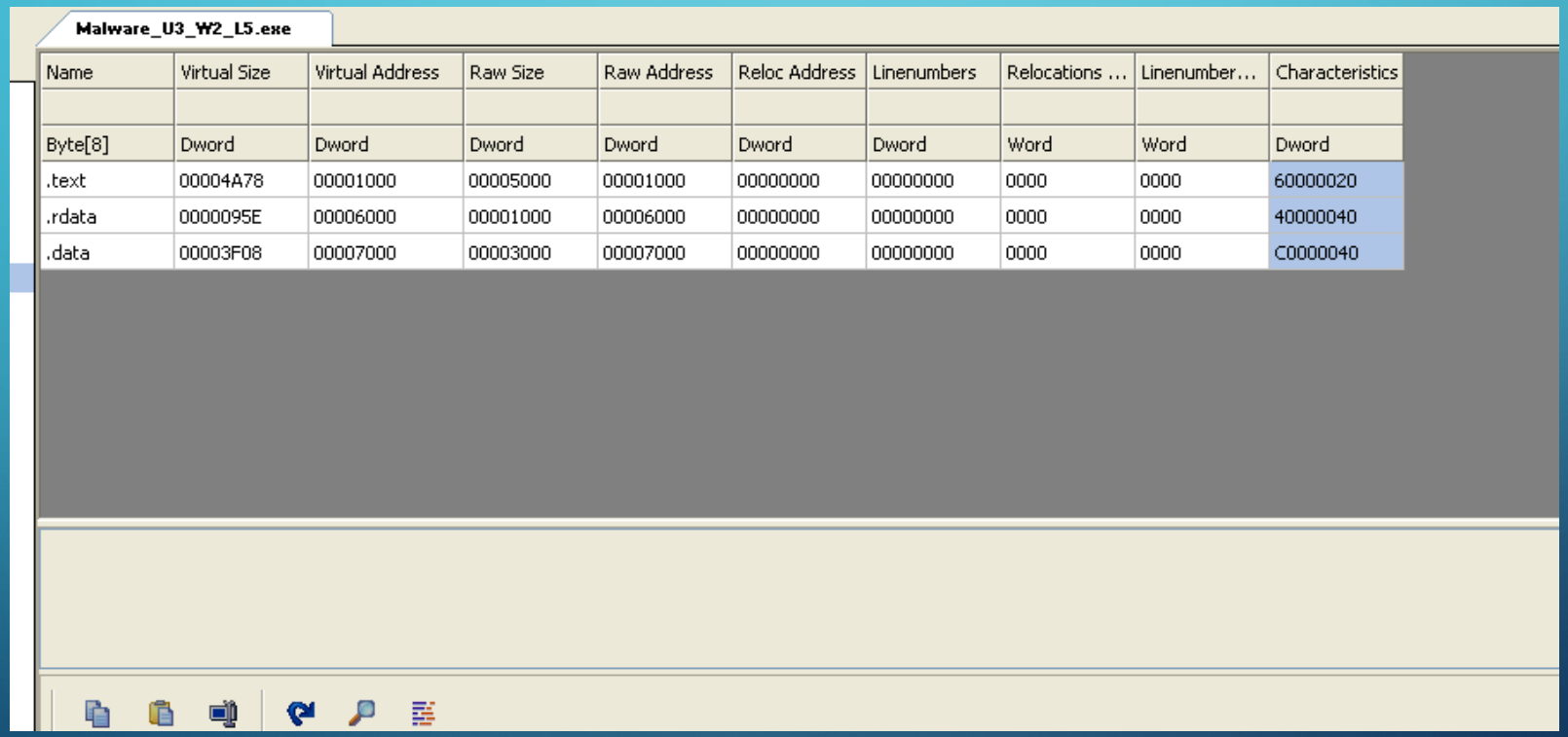
LE LIBRERIE IMPORTATE DEL FILE ESEGUIBILE UTILIZZANDO CFF EXPLORER, ANDIAMO SU«IMPORT DIRECTORY» DAL PANNELLO PRINCIPALE A SINISTRA. LE LIBRERIE IMPORTATE DAL FILE ESEGUIBILE

SONO**:KERNEL32.DLL**: LIBRERIA PIUTTOSTO COMUNE CHE CONTIENE LE FUNZIONI PRINCIPALI PER INTERAGIRE CON IL SISTEMA OPERATIVO, AD ESEMPIO: MANIPOLAZIONE DEI FILE, LA GESTIONE DELLA MEMORIA

WININET.DLL: LIBRERIA CHE CONTIENE LE FUNZIONI PER L'IMPLEMENTAZIONE DI ALCUNI PROTOCOLLI DI RETE COME HTTP, FTP, NTP.

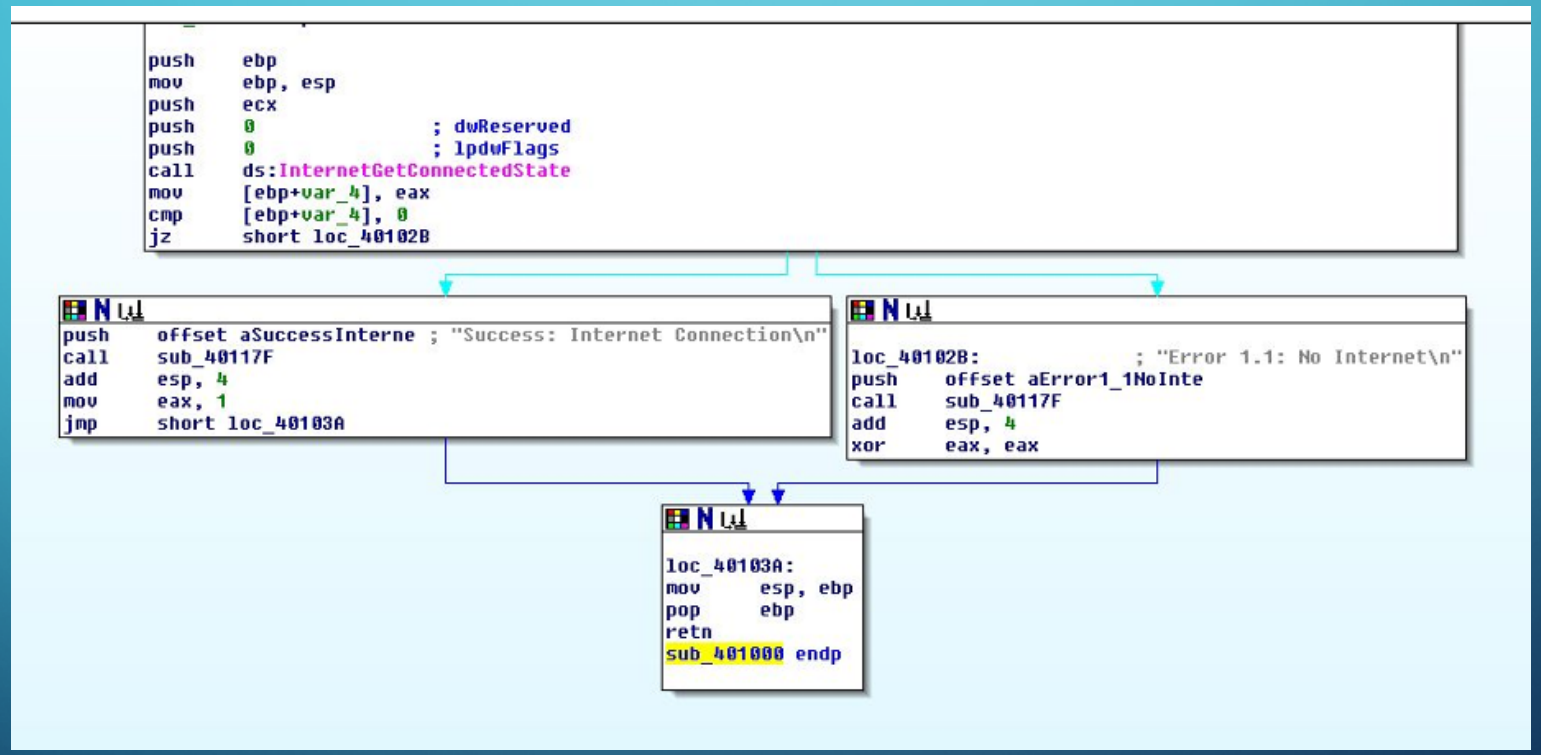


.TEST: UN TERMINE GENERICO UTILIZZATO PER FARE RIFERIMENTO A UNA SEZIONE DI TEST O POTREBBE ESSERE SPECIFICO PER UN
.RDATA:. CONTIENE PRINCIPALMENTE DATI DI SOLA LETTURA, COME STRINGHE DI COSTANTI E TABELLE DI LOOKUP, CHE L'ESEGUIBILE UTILIZZA DURANTE L'ESECUZIONE.DETERMINATO CONTESTO O APPLICAZIONE.
.DATA:CONTIENE DATI GLOBALI E STATICI CHE POSSONO ESSERE MODIFICATI DURANTE L'ESECUZIONE DEL PROGRAMMA



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

CON RIFERIMENTO ALLA FIGURA IN SLIDE , RISPONDE AI SEGUENTI QUESITI: IDENTIFICARE I COSTRUTTI NOTI (CREAZIONE DELLO STACK, EVENTUALI CICLI, COSTRUTTI) IPOTIZZARE IL COMPORTAMENTO DELLA FUNZIONALITÀ IMPLEMENTATA



- Il codice assembly sembra essere parte di una funzione che verifica lo stato della connessione internet. Se la connessione è attiva, stampa un messaggio di successo; in caso contrario, stampa un messaggio di errore.

I costrutti noti che si possono riconoscere nel codice sono:

La creazione dello stack con le istruzioni **push ebp** e **mov ebp, esp**. Queste istruzioni servono a salvare il valore del registro ebp e a impostarlo uguale al valore del registro esp, che punta alla cima dello stack. L'uso di **push** e **call** per passare parametri e chiamare una subroutine. Il codice usa **push ecx** e push offset **lpdwFlags** per passare due parametri alla funzione **InternetGetConnectedState,** che viene chiamata con call ds:InternetGetConnectedState. Questa funzione restituisce un valore booleano in eax che indica se c'è o meno una connessione internet.

L'uso di **cmp** e **jz** per confrontare valori e saltare a seconda del risultato. Il codice usa cmp [ebp+var_4], 0 per confrontare il valore restituito dalla funzione InternetGetConnectedState con zero, e usa jz short loc_40102B per saltare all'indirizzo loc_40102B se il valore è uguale a zero, cioè se non c'è connessione internet. L'uso di push offset e call per stampare messaggi stringa. Il codice usa push offset **aSuccessInterne e push offset** aError1NoInte per passare i messaggi di successo e di errore alla subroutine **sub_40117F**, che viene chiamata con call **sub_40117F**. Questa subroutine probabilmente stampa il messaggio passato come parametro sullo schermo o su un file. L'uso di xor eax, eax per azzerare il registro eax. Questa istruzione è equivalente a mov eax, 0, ma è più veloce e compatta. Il codice usa questa istruzione per impostare il valore di ritorno della funzione a zero.

- L’uso di jmp per saltare incondizionatamente a un altro indirizzo. Il codice usa jmp short **loc_40103A** per saltare alla fine della funzione, saltando il blocco di codice che stampa il messaggio di errore.

La distruzione dello stack con le istruzioni mov esp, ebp e pop ebp. Queste istruzioni servono a ripristinare il valore del registro esp e a recuperare il valore del registro ebp dallo stack. L’uso di retn per terminare la funzione e tornare al chiamante.

Questa istruzione salta all’indirizzo salvato sullo stack dal call precedente. Il comportamento della funzionalità implementata può essere ipotizzato come segue: Il codice chiama la funzione InternetGetConnectedState, che restituisce un valore booleano in eax che indica se c’è o meno una connessione internet. Se il valore è diverso da zero, significa che c’è una connessione internet, e il codice salta a **loc_40101F**, dove stampa il messaggio “Success: Internet Connection” e poi termina la funzione con il valore di ritorno zero. Se il valore è uguale a zero, significa che non c’è una connessione internet, e il codice salta a **loc_40102B**, dove stampa il messaggio “Error 1.1: No Internet” e poi termina la funzione con il valore di ritorno zero.