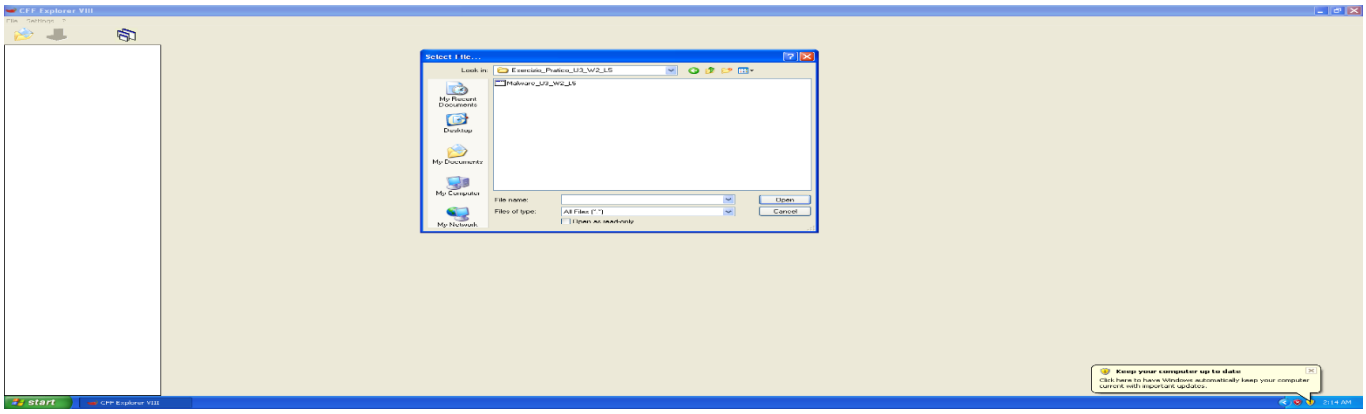
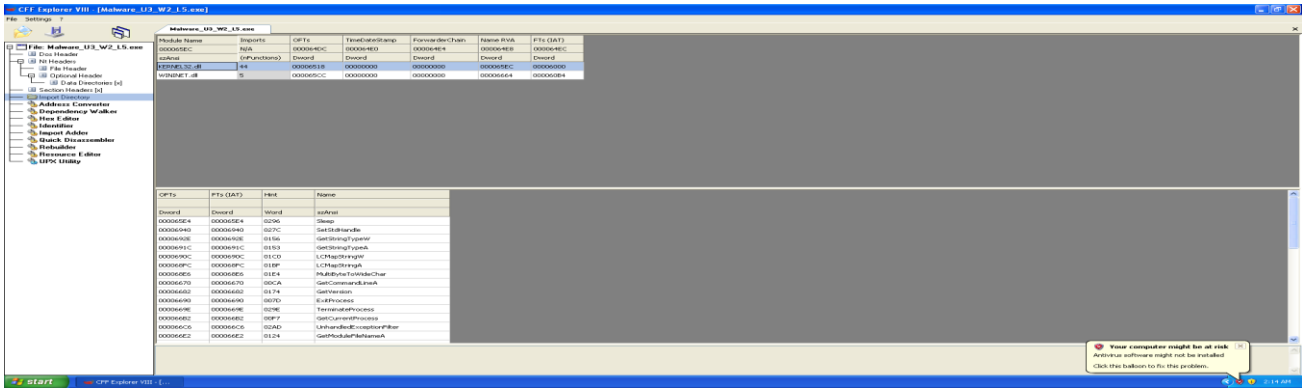
Per prima cosa apriamo il tool CFF EXPLORER e importiamo l'eseguibile che vogliamo esaminare:



Le librerie importate sono 2:

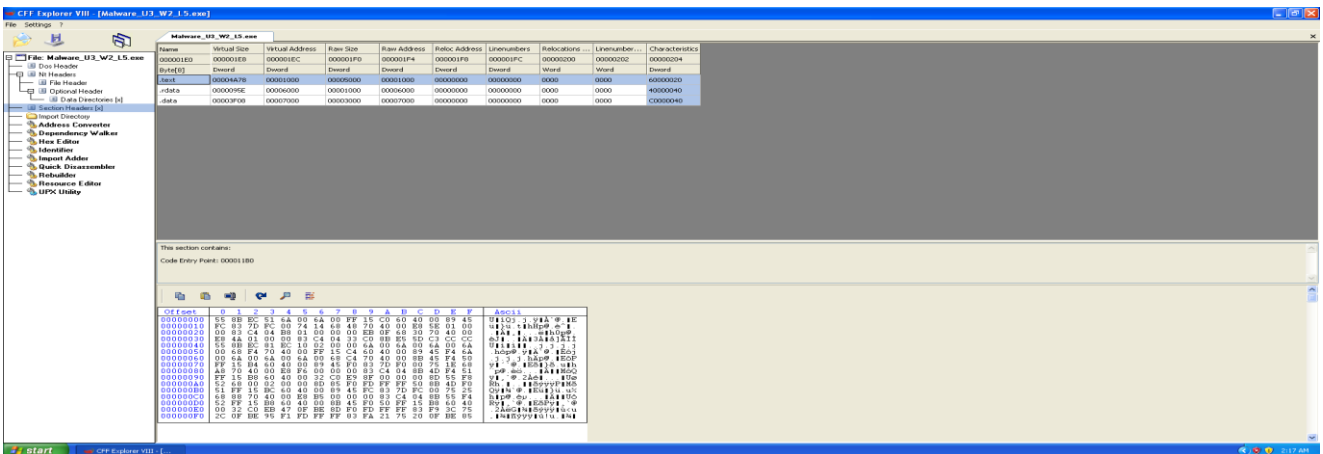
- **Kernell32.dll:** contenente le funzioni principali per interagire con il sistema operativo, come la manipolazione dei file e la gestione della memoria

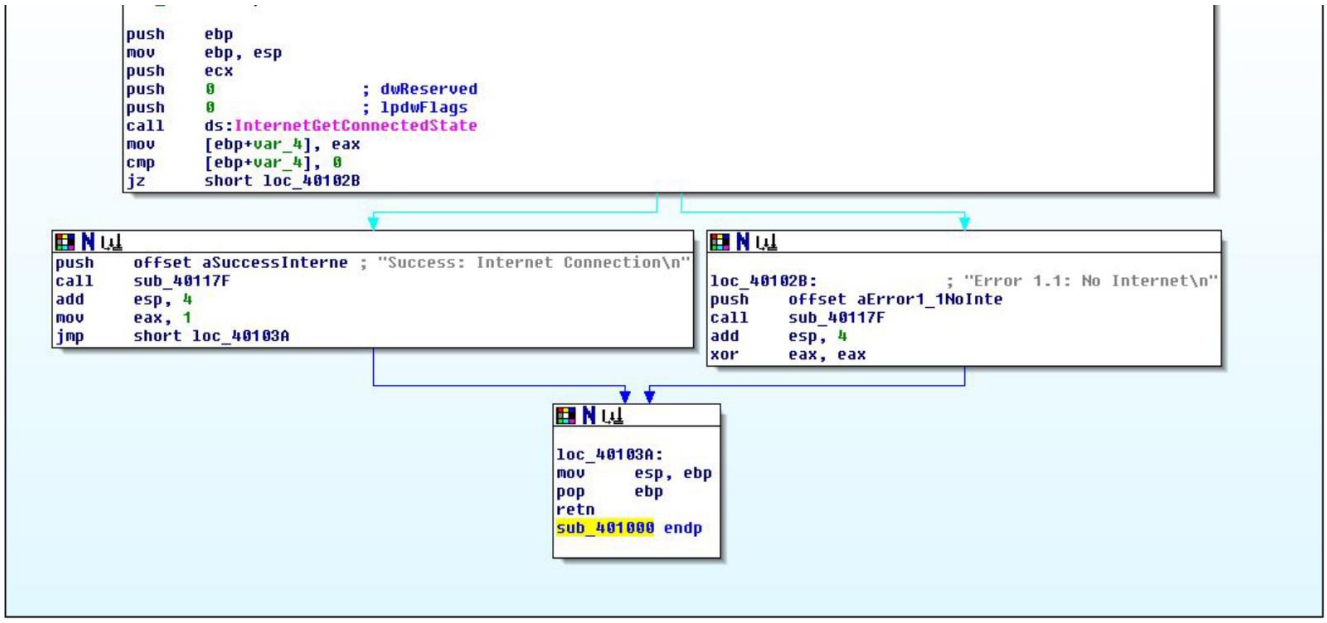
- **Wininet.dll:** contenente le funzioni per l'implementazione dei protocolli di rete come http, ftp, ntp



Le sezioni di cui si compone il file eseguibile sono 3:

- **.text**: contenente le istruzioni del codice che la cpu dovrà eseguire una volta che il software sarà avviato.
- .**rdata**: contenente tutte le informazioni riguardanti le librerie e le funzioni importate ed esportate dall'eseguibile.
- **.data**: contenente i dati e le variabili globali dell'eseguibile i quali dovranno essere sempre disponibili.





1.) *CREAZIONE DELLO STACK*: una funzione chiamante chiama un'altra funzione (funzione chiamata ovvero ***InternetGetConnectedState***) e viene creato un nuovo stack
Push ebp
Mov ebp,esp

2.) *CHIAMATA DI FUNZIONE E PASSAGGIO DI PARAMETRI :* Vengono passati i parametri(nel nostro caso 3 parametri) alla funzione chiamata sullo stack tramite istruzioni *push* e secondo la regola del LIFO, poi la funzione viene chiamata direttamente con l'istruzione *call:*
Push ecx
Push 0
Push 0
Call ds: InternetGetConnectedState

3.) *ASSEGNAZIONE VALORE:* viene assegnato il valore contenuto nel registro eax all'indirizzo di memoria specificato tramite l'istruzione *mov: (praticamente è il valore di ritorno della funzione che viene messo in eax e poi assegnato all'indirizzo di memoria specificato quindi viene creata una variabile locale contenente il valore di ritorno)*
Mov [ebp+var_4], eax

4.) *IF/ELSE:* viene controllato il valore di ritorno(return) della funzione chiamata tramite un ciclo if, confrontando il valore assegnato all'indirizzo di memoria **ebp+var_4 con 0.** Se questo valore di ritorno è uguale a 0 allora parte l'else e significa che non c'è connessione, se diverso da 0 allora c'è connessione:
Cmp [ebp+var_4],0

Jz short_loc…..
Infatti ricordiamo che dopo il confronto con CMP c'è un jump da effettuare e nel nostro caso viene fatto se lo ZF è impostato a 1(JZ) ovvero quando il valore di sorgente e destinazione sono uguali, per cui se il valore di ritorno della funzione è 0 allora è uguale al valore con cui si fa il confronto (0) e di conseguenza lo ZF si setterà su 1 e viene effettuato il jump (in C partirebbe l'ELSE)

Possiamo infatti ipotizzare il codice in C come il seguente:

```
state=internetgetconnectedstate[par1,0,0];
if (state!=0)
    printf("internet connection");
else
    printf ("no internet");
    return 0;
```

5.) *CASO IF:* Se il valore di ritorno della funzione è diverso da 0, vediamo che prima viene aggiunto nello stack tramite istruzione *push* la stringa <<success internet connection>> e poi viene chiamata la funzione con *call.* Dopodiche si aggiunge/elimina spazio nell'ordine di 4 byte allo stack, per cui per 1 variabile locale(infatti dopo per eliminare lo stack una volta che la funzione ha terminato il suo compito, occorre passare a esp il contenuto di ebp), si inizializza il registro eax a 1 e si passa alla rimozione dello stack.
Push offset …"success internet connection"
Call sub_...
Add esp,4
Mov eax, 1
Jmp short….

6.) *CASO ELSE: Se il valore di ritorno della funzione è uguale a 0, allora scatta l'else, il procedimento è analogo al precedente, eccetto che si inizializza il registro eax a 0 e anche in questo caso si elimina successivamente lo stack.*
7.) *PULIZIA DELLO STACK*
Sia che si verifichi la condizione if sia che scatti l'else, viene rimosso lo stack della funzione creata:
MOV esp, ebp
POP ebp

Lo scenario ipotetico è il return (exit) della funzione if/else