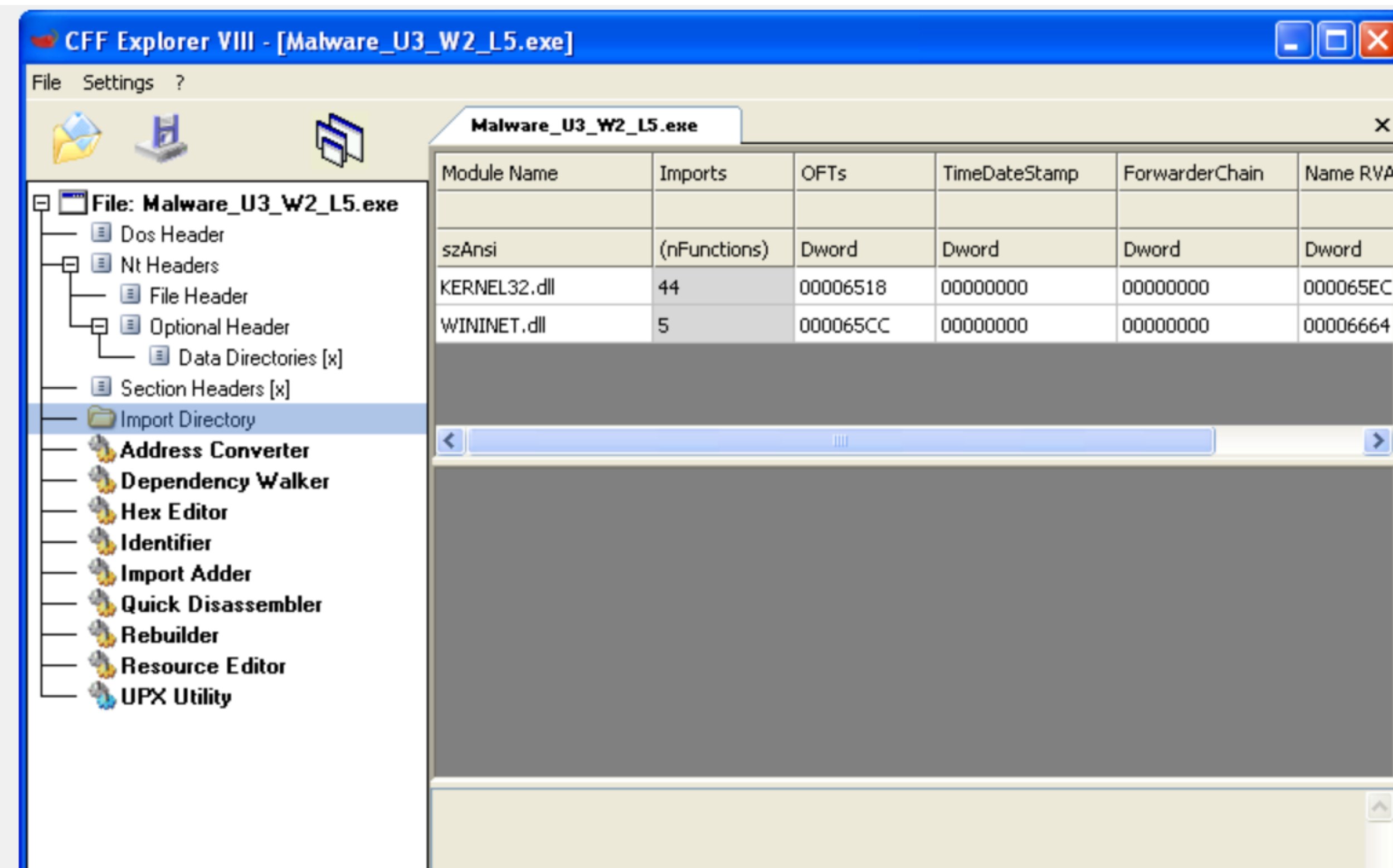


S10 L5 Progetto

Analisi statica basica

Nel esercizio di oggi ci viene chiesto di eseguire un'analisi statica basica del malware "Malware_U3_W_L5.exe" presente sulla nostra macchina virtuale e di ricavarne le librerie importate e le sezioni che lo compongono. Per fare ciò ho a disposizione diversi approcci e diversi tool, ho scelto di utilizzare CFF Explorer che può completare entrambi i punti richiesti.



Dopo aver avviato il tool e caricato il PE che vogliamo analizzare, possiamo spostarci su "Import Directory" per analizzare le librerie e le funzioni che il malware va ad importare. Notiamo che ce ne sono due e sono :

- Kernel32.dll
- WININET.dll

Analizziamole velocemente :

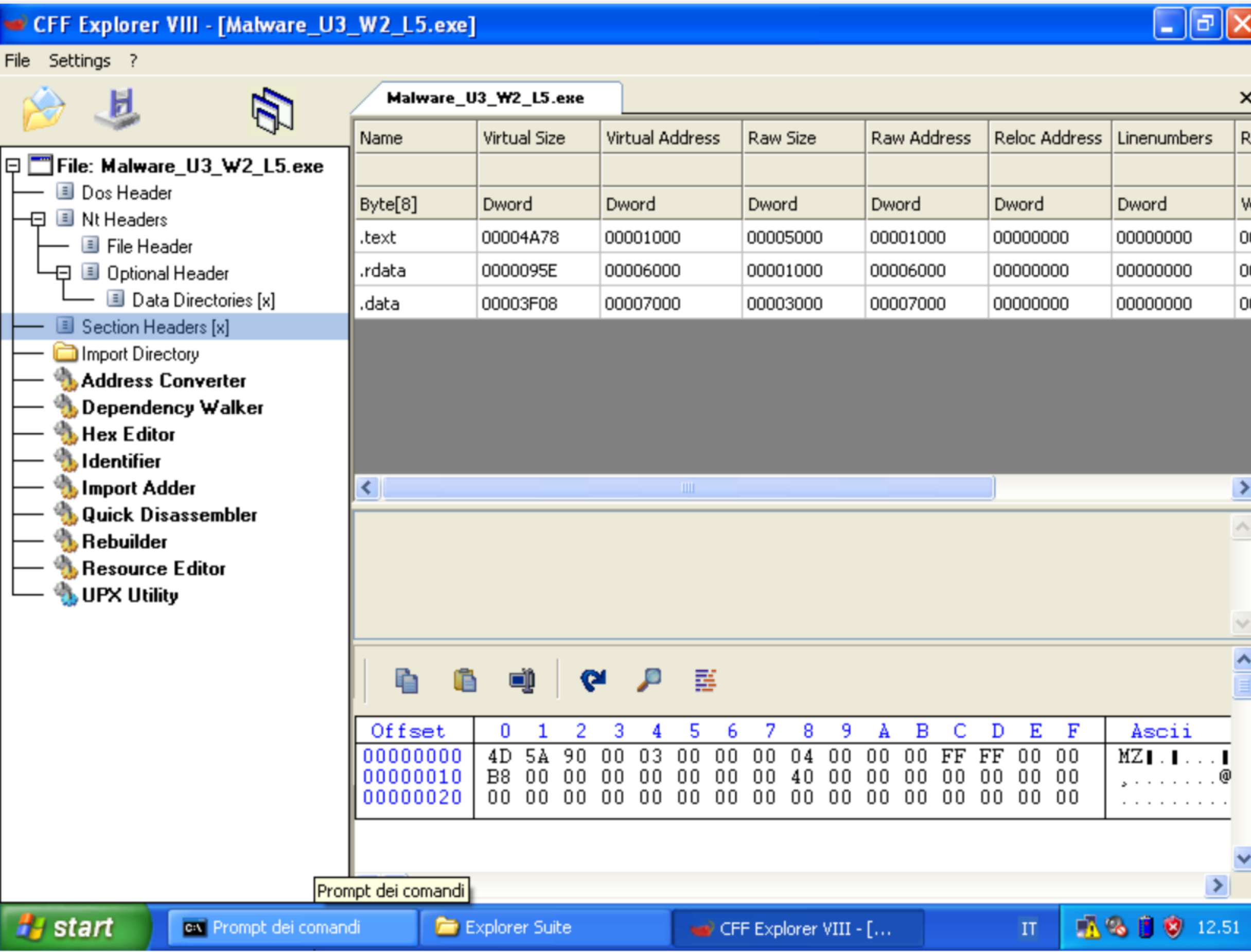
1. Kernel32.dll:

- Ruolo: Kernel32.dll è una delle librerie di sistema più importanti in Windows. Contiene funzioni di base che sono utilizzate da programmi e applicazioni per eseguire operazioni di basso livello, come la gestione della memoria, la gestione dei file, la gestione delle eccezioni e la comunicazione con il kernel del sistema operativo.
- Funzioni comuni: Tra le sue funzioni comuni ci sono la creazione e la gestione dei thread, la gestione dei file e delle directory, la gestione della memoria virtuale, e molte altre operazioni di sistema.

2. WININET.dll:

- Ruolo: WININET.dll (Windows Internet API) è una libreria che fornisce un'interfaccia di programmazione per la comunicazione con il Web attraverso il protocollo HTTP (Hypertext Transfer Protocol). In sostanza, è coinvolta nella gestione delle operazioni di rete e della connettività Internet.
- Funzioni comuni: Tra le sue funzioni comuni ci sono l'apertura di connessioni HTTP, il download e l'upload di dati, la gestione dei cookie, la gestione delle cache, e altre operazioni relative alla comunicazione su Internet.

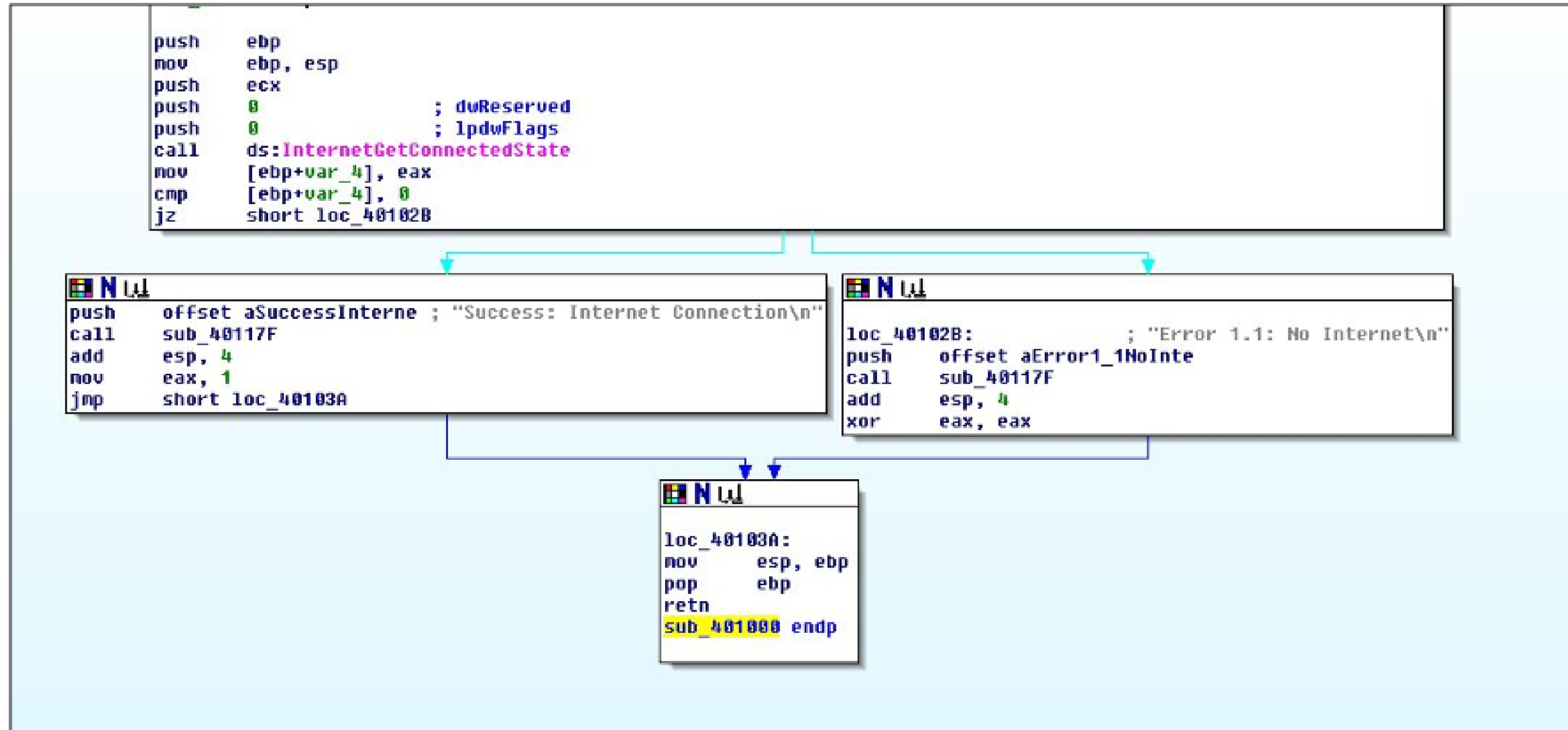
Spostiamoci ora nella sezione "Section headers" per analizzare le sezioni che compongono il malware.



Notiamo come l'eseguibile è composto da tre sezioni :

- .text che contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato.
- .rdata che : include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile
- .data che contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.

Per la seconda parte del esercizio di oggi prendiamo in considerazione il seguente codice:



Nella prossima slide andremo ad analizzare i costrutti noti ed ipotizzare il funzionamento di questo codice assembly

Con queste due righe del codice precedente siamo andati a creare lo stack:

```
push    ebp
mov     ebp, esp
```

Qui invece vediamo come richiamare una funzione passandole dei parametri tramite il push:

```
push    ecx
push    0          ; dwReserved
push    0          ; lpdwFlags
call    ds:InternetGetConnectedState
```

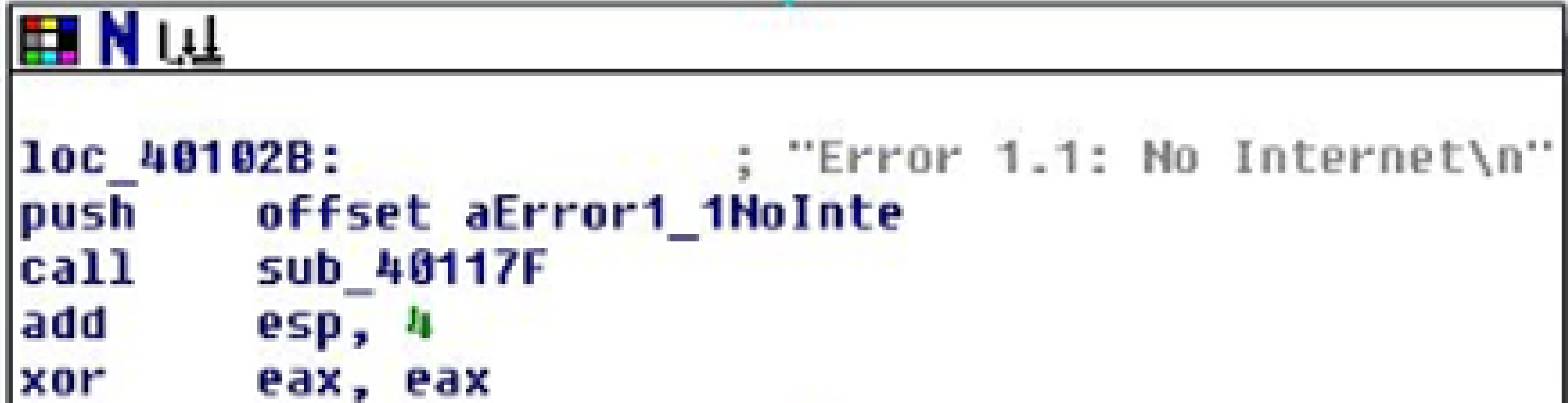
Quanto segue invece è la struttura di un ciclo if che controlla che il valore della variabile sia 0:

```
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

Abbiamo poi la gestione dei risultati del ciclo if, quanto segue nel caso di valore 1 dello ZF:

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jnp     short loc_40103A
```

E la gestione nel caso in cui lo ZF sia 0:



A screenshot of a debugger window with a title bar containing a color palette icon, the letter 'N', and a cursor icon. The window displays assembly code for a function. The code starts at address `loc_40102B` with a comment `; "Error 1.1: No Internet\n"`. The instructions are: `push offset aError1_1NoInte`, `call sub_40117F`, `add esp, 4` (where the '4' is highlighted in green), and `xor eax, eax`.

```
loc_40102B:          ; "Error 1.1: No Internet\n"
push     offset aError1_1NoInte
call     sub_40117F
add      esp, 4
xor      eax, eax
```

E infine abbiamo la chiusura della funzione con relativa pulizia dello stack:



A screenshot of a debugger window with a title bar containing a color palette icon, the letter 'N', and a cursor icon. The window displays assembly code for the end of a function. The code starts at address `loc_40103A` with instructions: `mov esp, ebp`, `pop ebp`, `retn`, and `sub_401000 endp` (where `sub_401000` is highlighted in yellow).

```
loc_40103A:
mov      esp, ebp
pop      ebp
retn
sub_401000 endp
```

È stato notato che la funzione controlla lo stato della connessione a Internet, gestisce il successo e l'errore, stampando messaggi appropriati e restituendo un valore (1 in caso di successo, 0 in caso di errore). Tuttavia, la specifica implementazione richiede la definizione di alcune funzioni e stringhe che non sono state incluse nel codice fornito.

FINE

Grazie
Massimo Cinquegrana