

ANALISI MALWARE STATICÀ

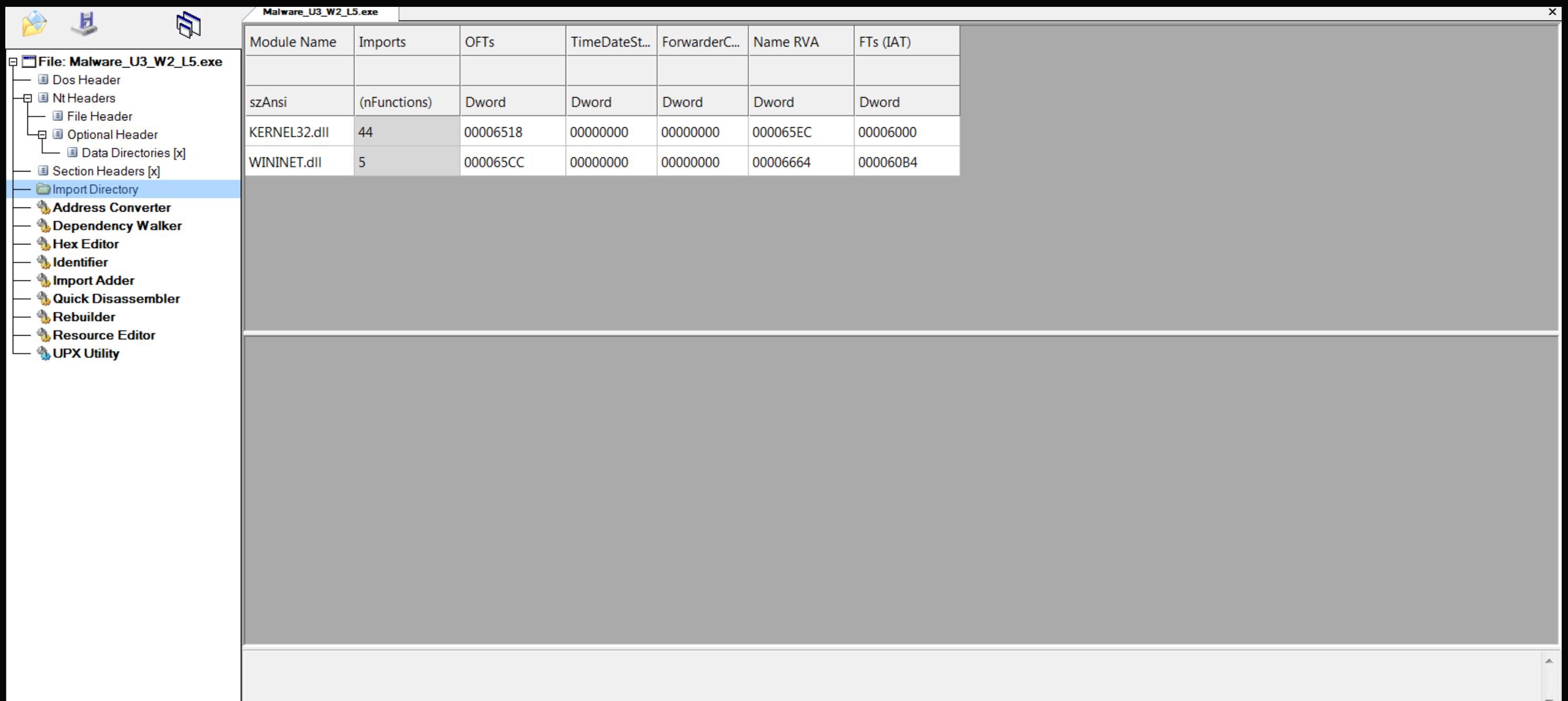
# Progetto S10-L5

# Traccia

## Analizzare il malware\_U3\_W2\_L5

- Quali librerie vengono importate dal file eseguibile
- Quali sono le sezioni di cui si compone il file eseguibile
- Identificare i costrutti noti
- Ipotizzare il comportamento della funzionalità implementata
- fare la tabella con significato delle singole righe di codice

# Librerie

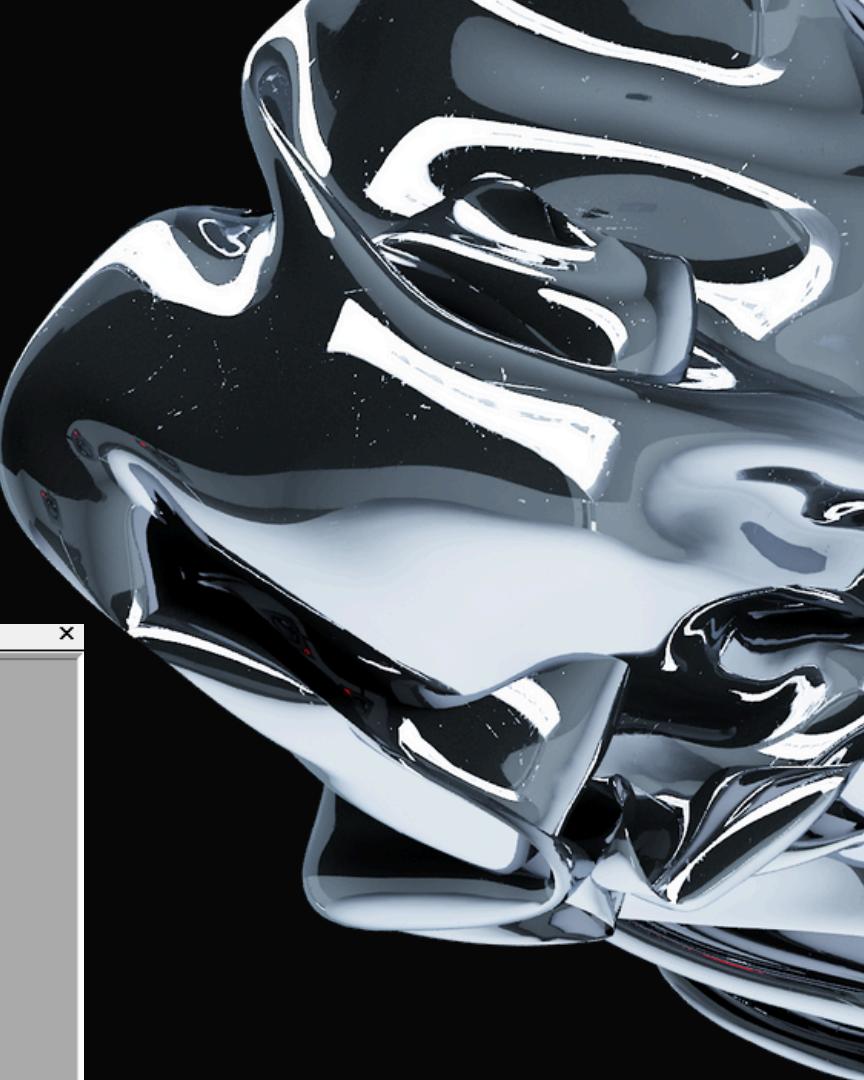


The screenshot shows a debugger interface with the following details:

**File Path:** File: Malware\_U3\_W2\_L5.exe

**Imports Table:**

Module Name	Imports	OFTs	TimeDateSt...	ForwarderC...	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4



# Librerie nel dettaglio

File Settings ?

File: Malware\_U3\_W2\_L5.exe

Dos Header  
Nt Headers  
File Header  
Optional Header  
Data Directories [x]  
Section Headers [x]  
Import Directory  
Address Converter  
Dependency Walker  
Hex Editor  
Identifier  
Import Adder  
Quick Disassembler  
Rebuilder  
Resource Editor  
UPX Utility

Malware\_U3\_W2\_L5.exe

Module Name	Imports	OFTs	TimeDateSt...	ForwarderC...	Name RVA	FTs (IAT)
000065EC	N/A	000064DC	000064E0	000064E4	000064E8	000064EC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW
000068FC	000068FC	01BF	LCMapStringA
000068E6	000068E6	01E4	MultiByteToWideChar



# • Librerie nel dettaglio

The screenshot shows a debugger interface with a sidebar on the left containing navigation links such as File, Dos Header, Nt Headers, File Header, Optional Header, Data Directories [x], Section Headers [x], Import Directory, Address Converter, Dependency Walker, Hex Editor, Identifier, Import Adder, Quick Disassembler, Rebuilder, Resource Editor, and UPX Utility. The main window displays two tables. The top table, titled 'Module Name: Imports', lists imports from the file 'Malware\_U3\_W2\_L5.exe'. The bottom table, titled 'Import Directory', lists the specific functions imported from the 'WININET.dll' library.

Module Name	Imports	OFTs	TimeDateStamp	ForwarderCount	Name RVA	FITS (d/w)
00006664	N/A	000064F0	000064F4	000064F8	000064FC	00006500
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006640	00006640	0071	InternetOpenUrlA
0000662A	0000662A	0056	InternetCloseHandle
00006616	00006616	0077	InternetReadFile
000065FA	000065FA	0066	InternetGetConnectedState
00006654	00006654	006F	InternetOpenA

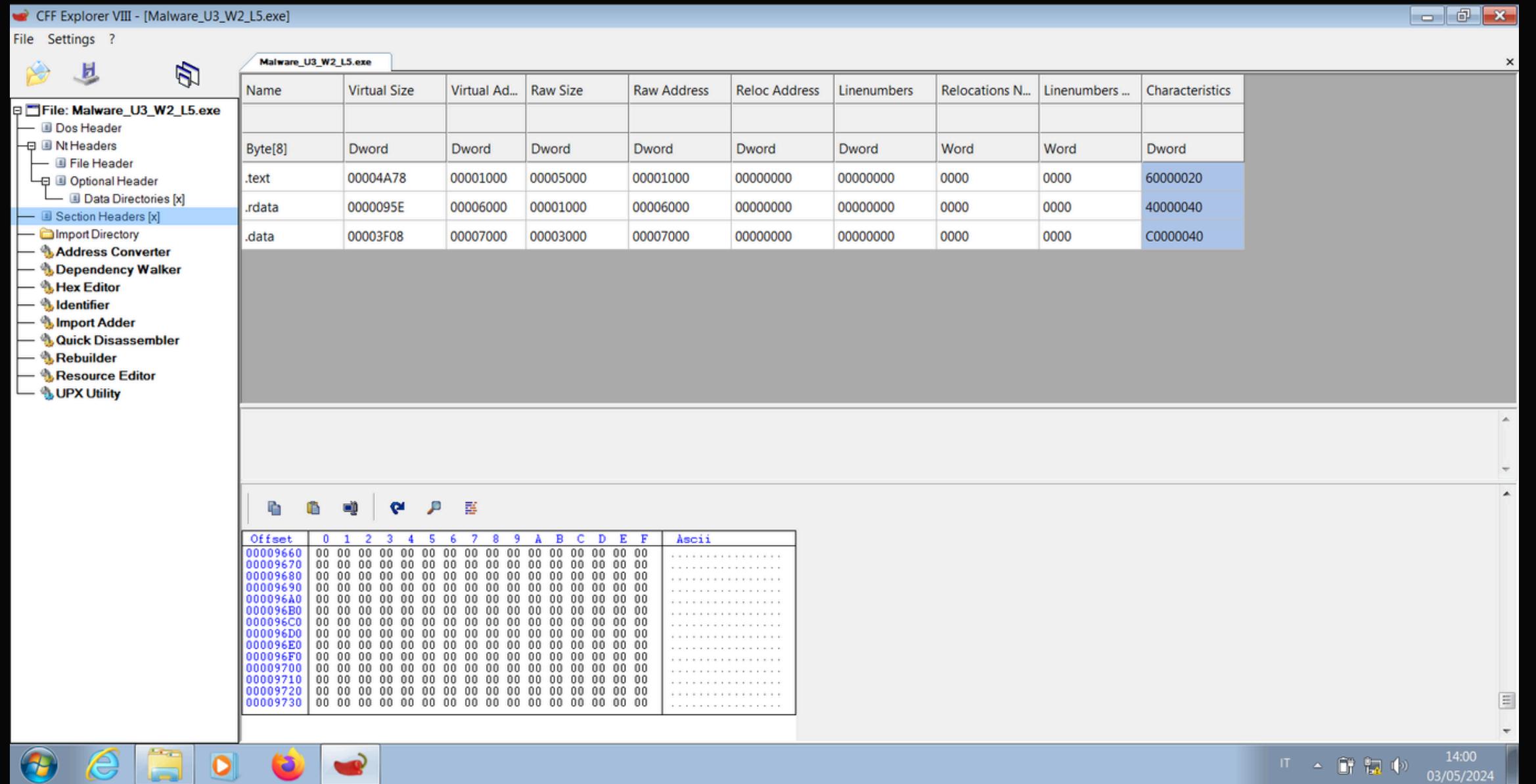


- **Kernel32.dll:** contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria.
- **Wininet.dll:** contiene le funzioni per l'implementazione di alcuni protocolli di rete come HTTP, FTP, NTP.

## Analisi delle librerie

Come analizzato vediamo che il malware fa delle interazioni con il sistema operativo bersaglio, ma grazie all'analisi più approfondita delle operazioni con la libreria wininet si suppone che faccia una richiesta http e apra un url malevolo scarichi qualcosa, chiuda la connessione e avvia una connessione

# Analisi exe



Nulla di rilevante nel file eseguibile del malware

# Analisi codice assembly

```
push    ebp  
mov     ebp, esp  
push    ecx  
push    0          ; dwReserved  
push    0          ; lpdwFlags  
call    ds:InternetGetConnectedState  
mov     [ebp+var_4], eax  
cmp     [ebp+var_4], 0  
jz      short loc_40102B
```

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

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

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



# Costrutti noti

Nel seguente codice analiziamo la creazione dello stack per la gestione di una connessione attraverso una chiamata alla funzione InternetConnectedState e un ciclo if che nel caso dia errore o meno fa altre chiamate ad altre funzioni in base al valore restituito dalla funzione.

# Analisi del codice per riga

```
push ebp // salva sul registro ebp
    mov ebp,esp //punta ad ebp
    push ecx salva il valore su ecx
    push 0 ; dwReserved // aggiunge uno 0 allo stack
    push 0 ; lpdwFlags //aggiunge uno 0 allo stack
call ds:InternetGetConnectedState //verifica la connessione tramite funzione
    mov [ebp+var_4], eax //memorizza il valore di ritorno della funzione
    cmp [ebp+var_4], 0 //confronta la var temp con 0
    jz short loc_40102B //salta se il valore del confronto è 0
push offset aSuccessInterne //se c'è una connessione aggiunge sullo stack l'indirizzo della stringa
    call sub_40117F //chiama una funzione
add esp, 4 // ripristina lo stack rimuovendo l'ultimo valore
    mov eax, 1// imposta il valore 1 del registro eax
    jmp short loc_40103A salta all'etichetta
loc_40102B: etichetta se non c'è connessione
push offset_aError1_1Nolnte //aggiunge sullo stack l'indirizzo di una stringa
    call sub_40117F //chiama la funzione di prima
        add esp,4 // ripristina lo stack
        xor eax, eax // imposta eax a 0
loc_40103A: //etichetta per uscire dal ciclo if
mov esp, ebp //riporta lo stack al suo stato iniziale
    pop ebp ripristina ebp
retn restituisce il controllo sulla chiamata precedente
sub_401000 enp fine della sezione di codice
```