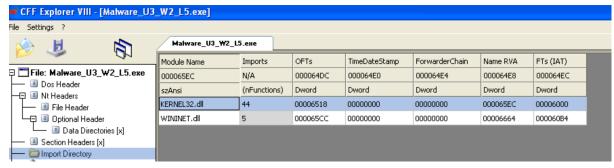
1) Librerie del Malware



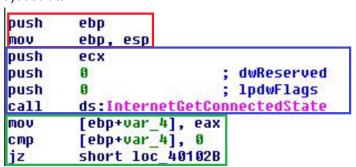
Notiamo che il programma carica 2 librerie:

KERNEL32.dll con 44 funzioni, necessaria a comunicare con il sistema operativo WININET.dll con 5 funzioni utilizzate principalmente per accedere a servizi di rete quindi ci aspettiamo che il programma possa comunicare.

2)

e Settings ?										
	Malware_U3_W2_L5.exe									
	Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations	Linenumber	Characteristics
File: Malware_U3_W2_L5.exe										
— I Dos Header	Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
→ ■ Nt Headers ■ File Header	.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
□ □ Optional Header	.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
☐ Data Directories [x]	.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

3)costrutti



- in ROSSO possiamo vedere la creazione di un stack di memoria di grandezza indefinita.
- in BLU il codice ci suggerisce che il programma ha utlizzato 2 variabili per verificare con la funzione invocata in CALL la connesione ad internet della macchina.
- -in VERDE è evidente che il programma abbia eseguito una condizione di tipo IF/ELSE data dalla coppia CMP-JZ (compare e jump)

```
push offset aSuccessInterne; "Success: Internet Connection\n"
call sub 40117F
add esp, 4
mov eax, 1
jmp short loc 40103A
```

- -in ROSSO vediamo che il malware esegue la funzione PRINTF per stampare a video la frase di avvenuta connesione.
- con le altre istruzioni sposta e aggiunge valori ai registri esp e eax, infine salta a allo stack conclusivo.

```
loc_401028: ; "Error 1.1: No Internet\n"
push offset aError1_1NoInte
call sub_40117F
add esp, 4
xor eax. eax
```

-in ROSSO possiamo intuire che la connesione ai servizi di rete non sia andata a buon fine, il programma crea un messaggio di errore e lo stampa a video con la funzione printf.

-in VERDE ripulisce il registro eax con xor per salvare memoria

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

-con questa porzione di codice il programma ripulisce lo stack.

4)