

Progetto S10L5

*By Conti
Samuele*

Il progetto settimanale prevede l'analisi di un malware attraverso l'analisi statica basica e avanzata.

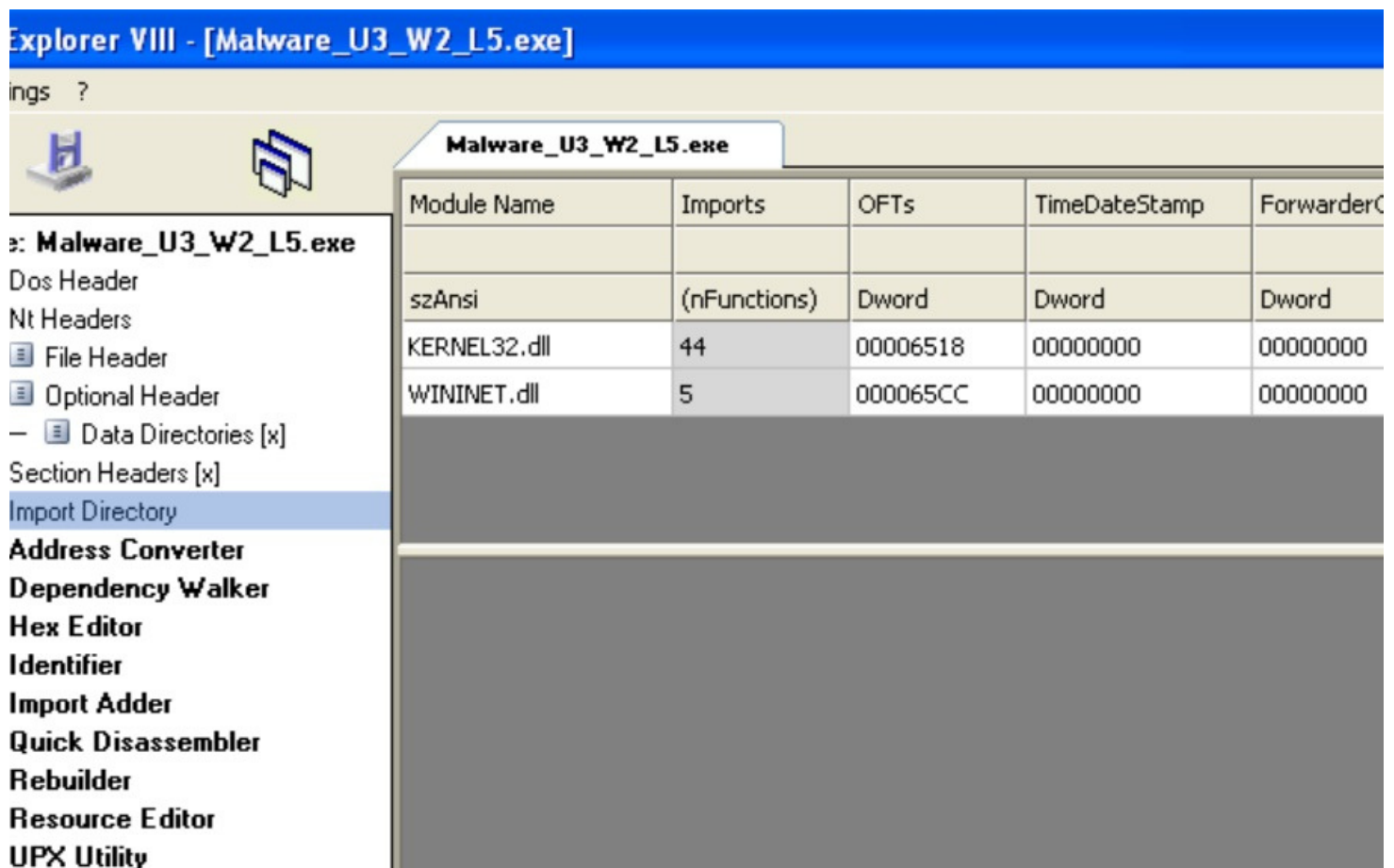
Nell'analisi statica basica dobbiamo individuare:

- le librerie che vengono importate dal file eseguibile
- le sezioni di cui il malware è composto

Attraverso il tool CFF Explorer, possiamo identificare le librerie importate che sono:

Kernel32.dll = contiene funzione principali per interagire con il sistema operativo

Wininet.dll = contiene funzioni inerenti ai protocolli di rete come HTTP,FTP,NTP



Explorer VIII - [Malware_U3_W2_L5.exe]

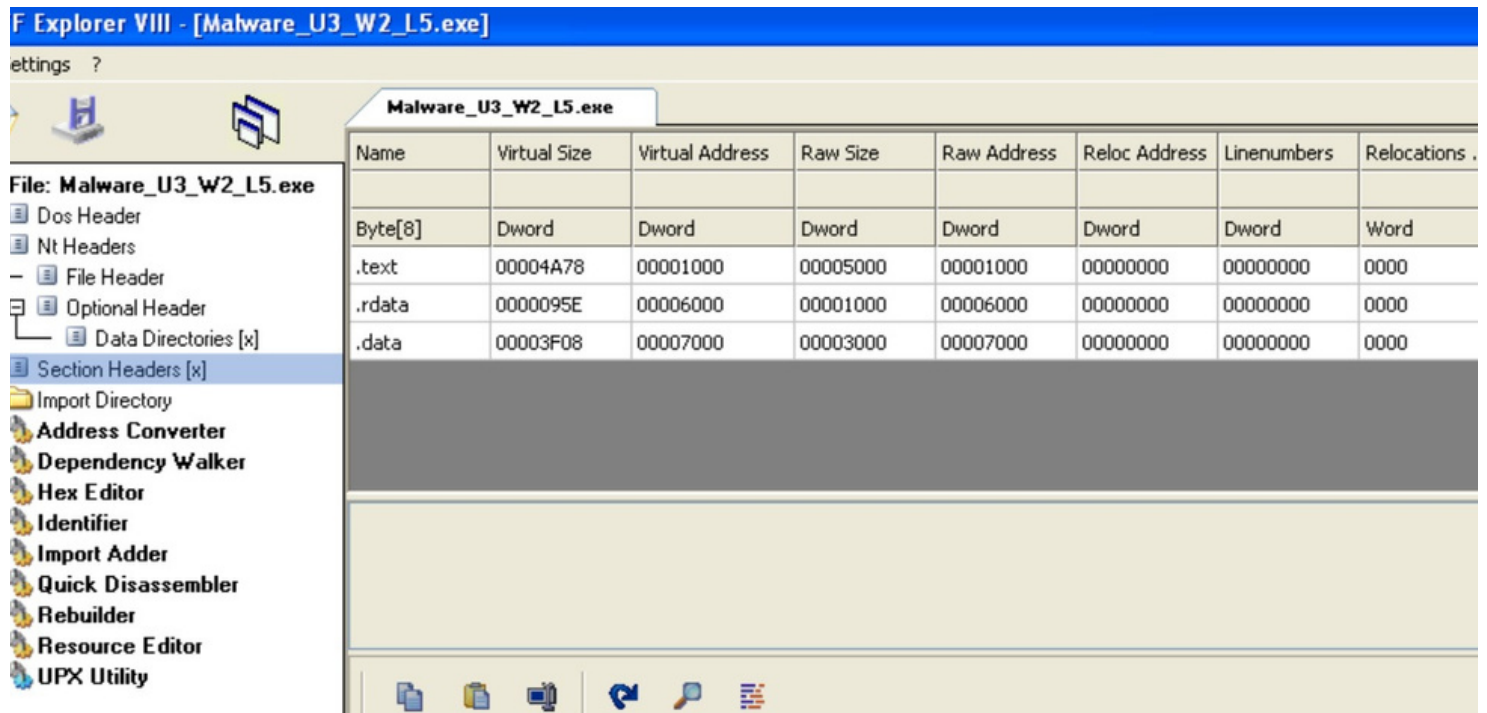
Malware_U3_W2_L5.exe

Module Name	Imports	OFTs	TimeStamp	Forwarder
szAnsi	(nFunctions)	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000
WININET.dll	5	000065CC	00000000	00000000

Left sidebar menu:

- 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

Sempre con CFF Explorer possiamo individuare anche le sezioni di cui è composto il Malware



Possiamo identificare le seguenti sezioni:

.text = contiene informazioni che la CPU eseguirà una volta che il software sarà avviato

.rdata = contiene informazione sulle librerie e funzione importate ed esportate dall'eseguibile

.data = contiene dati/variabili globali del programma eseguibile che devono essere disponibili da qualsiasi parte del programma

Con l'analisi statica avanzata andremo

a identificare i costrutti noti e a ipotizzare il comportamento della funzionalità implementata

```
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
```

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

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

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

I costrutti noti che possiamo identificare sono:

La creazione dello Stack:

```
//.text:00401000  push ebp
```

```
//.text:00401001  mov  ebp, esp
```

La chiamata di funzione (i parametri sono passati sullo stack attraverso l'istruzione push):

```
//.text:00401004  push 0      ;  
                    dwReserved
```

```
//.text:00401006  push 0      ;  
                    lpdwFlags
```

```
//.text:00401008  call  
ds:InternetGetConnectedState
```

Ciclo IF:

```
//.text:00401011  cmp [ebp+var_4],0
```

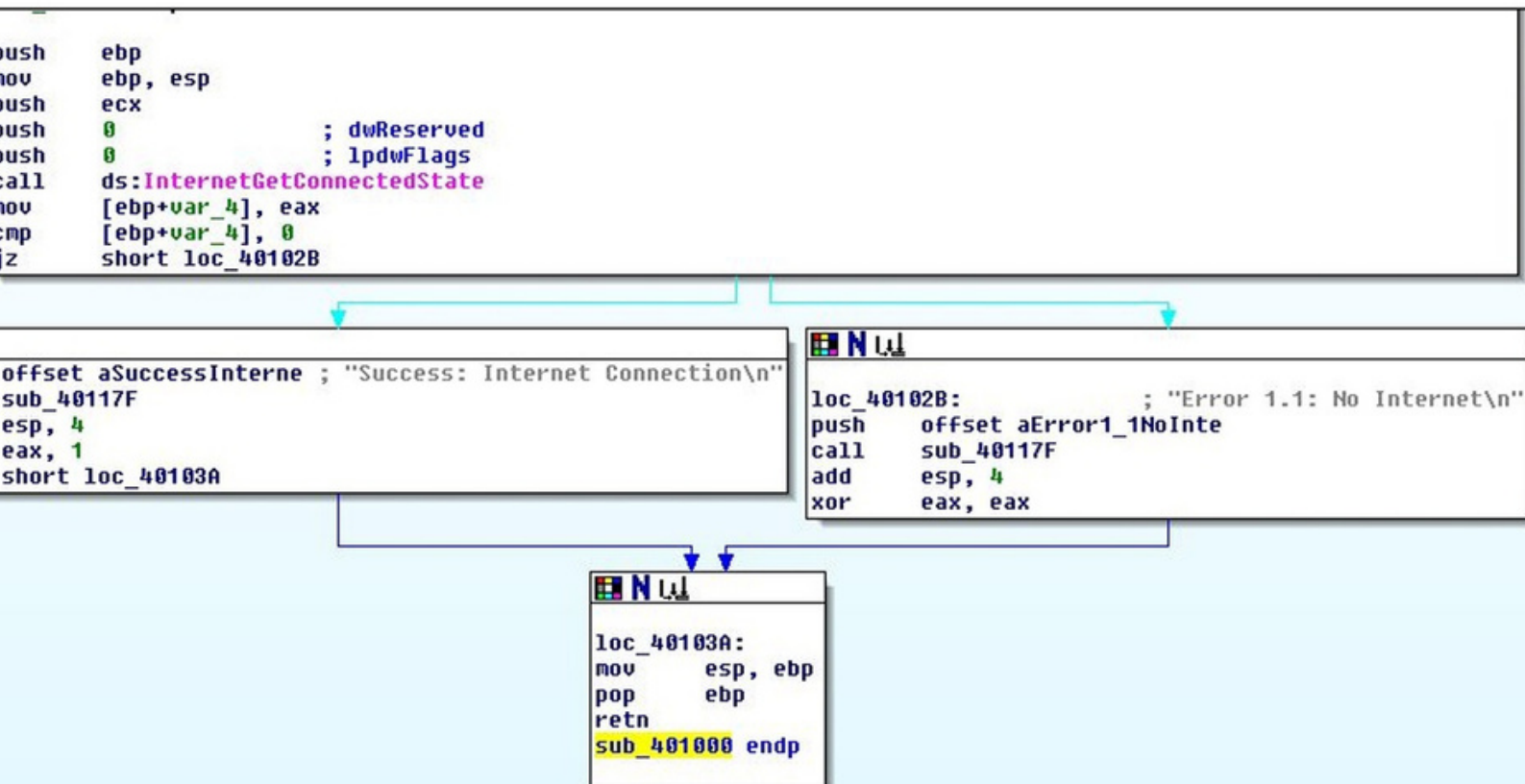
```
//.text:00401015  jz short  
                    loc_40102B
```

La chiusura dello stack:

mov esp, ebp

pop ebp

Ipotesi:



Possiamo intuire che il malware
chiama la funzione
internetgetconnectedstate, e ne
verifica con IF se il valore di ritorno
della funzione è diverso da 0.

Se così fosse, allora vuol dire che è presente una connessione attiva.

Task Bonus: Analisi delle singole righe del codice

Assembly:

Allego file txt sull'analisi del codice al progetto