PHANTOM SRL


PIGNATELLO
GIUSEPPE

# REPORT

## ANALISI MALWARE

**Prepared For :**
EPIC EDUCATION srl

**Liceria & Co.**
Siracusa, SR

PHANTOM s.r.l
IMPOSSIBLE IS OUR TARGET

# INDICE

PHANTOM s.r.l
IMPOSSIBLE IS OUR TARGET

# TRACCIA

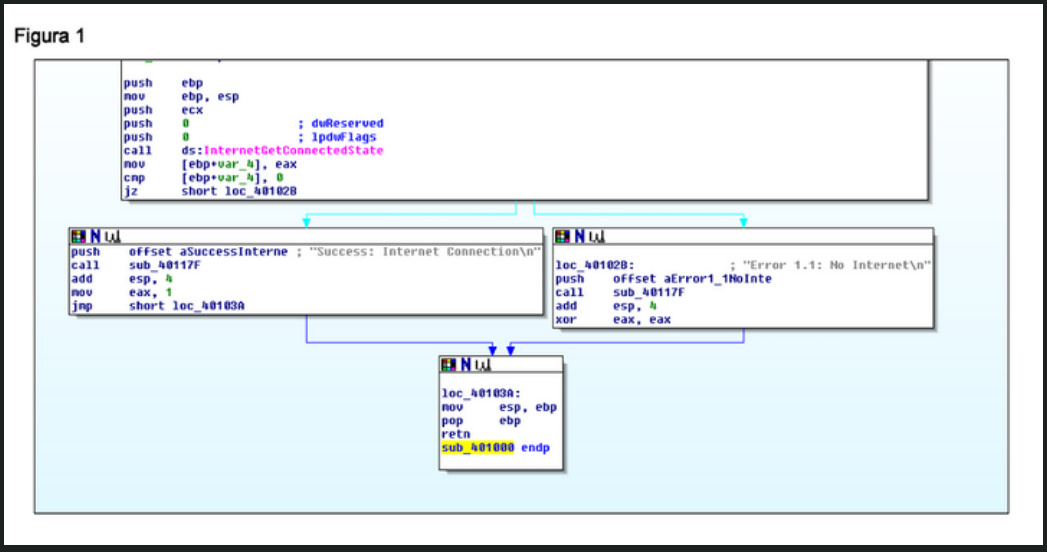
Con riferimento al file Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5 » sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

1. Quali librerie vengono importate dal file eseguibile?
2. Quali sono le sezioni di cui si compone il file eseguibile del malware?

Con riferimento alla Figura 1, risponde ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti )
4. Ipotizzare il comportamento della funzionalità implementata
5. BONUS fare tabella con significato delle singole righe di codice assembly

Figura 1



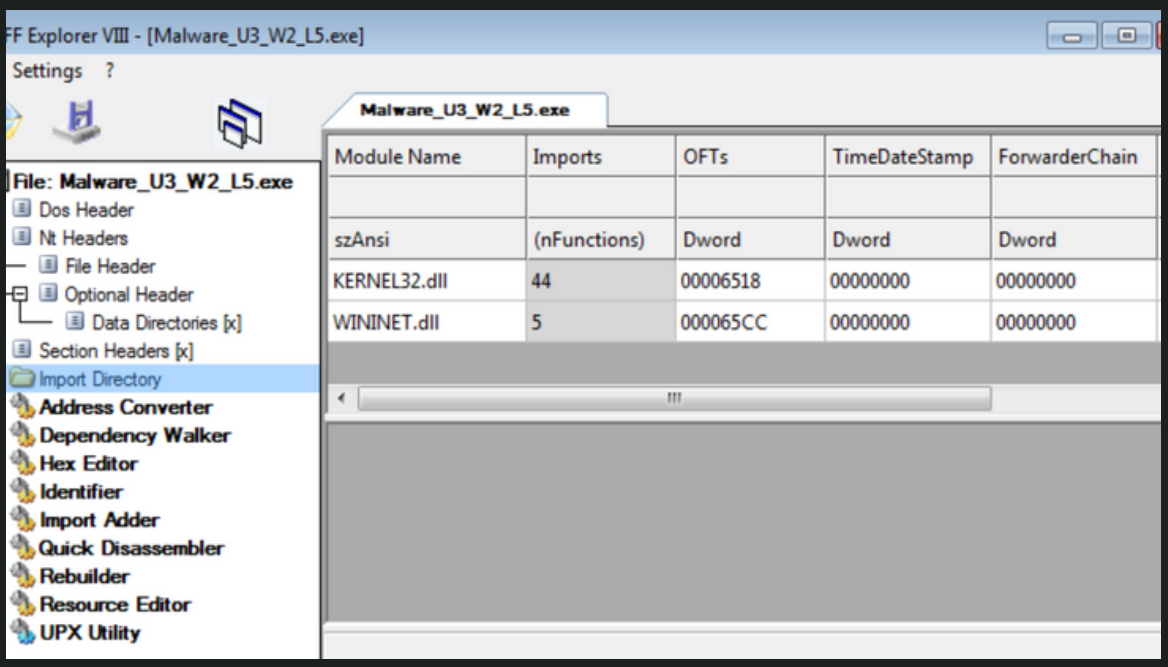
PHANTOM s.r.l
IMPOSSIBLE IS OUR TARGET

# LIBRERIE

Per scoprire quali librerie vengono utilizzate dal Malware, andiamo a utilizzare il tool CFF Explorer spostandoci su «Import directory» nel pannello a sinistra. Questo tool viene utilizzato durante l'analisi statica basica perchè non richiede l'esecuzione del programma. Tramite CFF Explorer, vediamo come il malware utilizzi due librerie note:

- Kernel32.dll --> Contiene le principali funzioni per interagire col Sistema Operativo, come ad esempio la gestione della memoria e le operazioni di input/output.

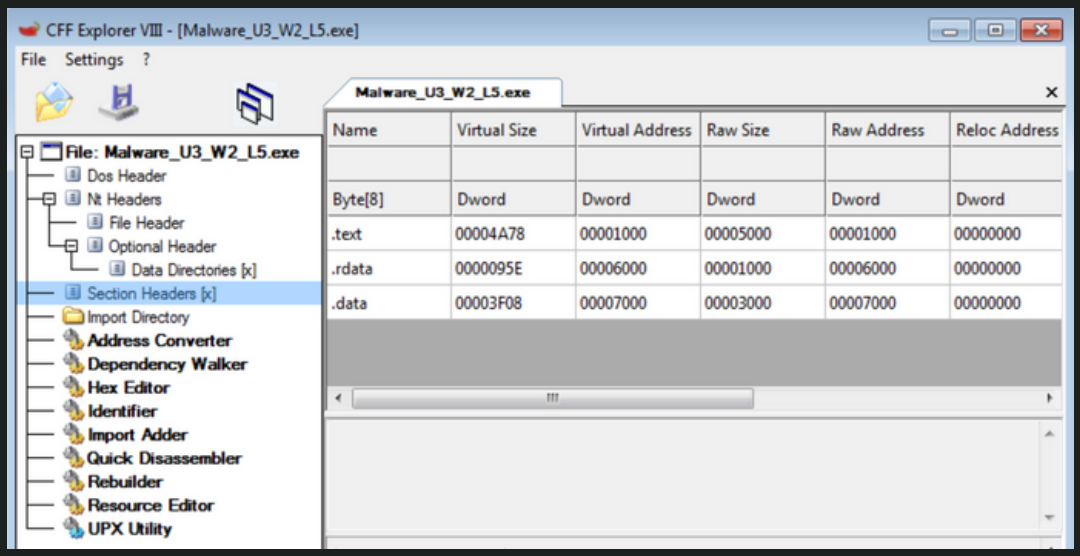
- WININET.dll --> Contine le funzioni per interagire con i protocolli FTP e HTTP per accedere alle risorse Internet.



# SEZIONI HEADER

Per quanto riguarda le sezioni, riusciamo a estrapolarle sempre utilizzando CFF Explorer spostandoci su «section headers», in questo caso vengono usate tre sezioni:

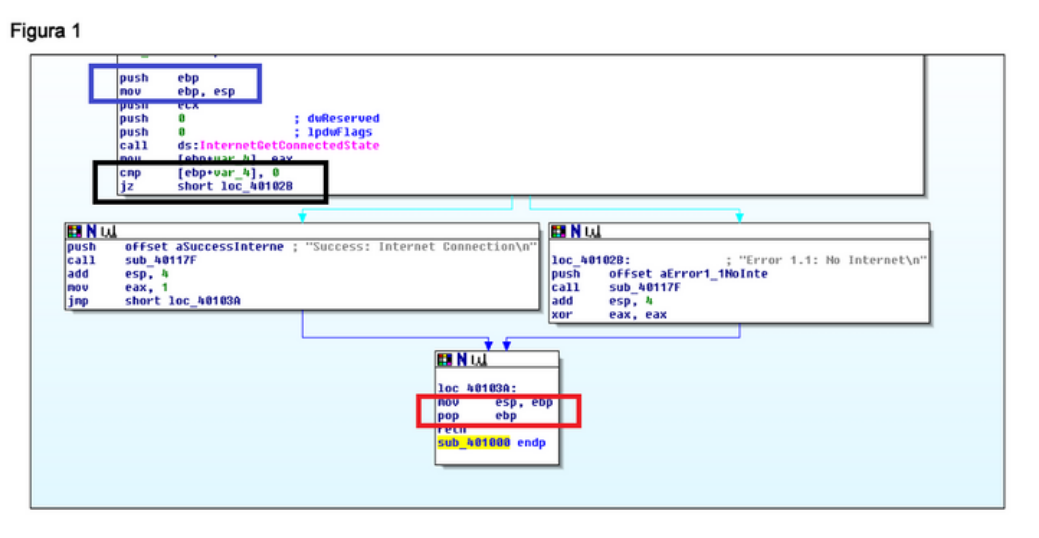
- .text --> contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato.
- .rdata --> include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile.
- .data --> contiene tipicamente i dati / le variabili globali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



# COSTRUTTI NOTI

I costrutti noti possiamo trovarli nel programma in assembly.
Fra i costrutti noti intendiamo:
Eventuali aperture dello stack, condizioni (IF, IF ELSE, WHILE)
Chiusura dello stack...
In questo programma i costrutti riconosciuti sono 3:

Figura 1




- Nel rettangolo blu, riconosciamo l'apertura dello stack (push e successivamente mov)
- Nel rettangolo nero, riconosciamo un costrutto condizionale IF (formato da cmp e successivamente jz)
- Nel rettangolo rosso invece riconosciamo la chiusura dello stack (mov e successivamente pop)

# COMPORTAMENTO DEL MALWARE

Il malware può essere utilizzato per controllare lo stato della connessione del sistema compromesso, nonché per acquisire comandi dal server C2 (in base all'URL fornito e al file in esso contenuto) e visualizzarli alla console. (KeyLogger).

Il termine C2 Server, abbreviazione di "command and control", potrebbe non essere ampiamente conosciuto, ma è il centro di comando principale per gli attacchi informatici. Gli attori malintenzionati utilizzano i server C2 per dirigere dispositivi compromessi, che vanno dai singoli computer a intere reti, nell'esecuzione dei loro attacchi digitali.

# TRACCIA BONUS

push ebp ; Salvare il valore del puntatore di base (EBP) nello stack
mov ebp, esp ; Imposta il puntatore di base (EBP) sul puntatore dello stack corrente (ESP)
push ECX ; Preserva il valore del registro ECX inserendolo nello stack
push 0 ; Inserire 0 nello stack come valore per dwReserved
push 0 ; Inserire 0 nello stack come valore per lpdwFlags
call ds: InternetGetConnectedState; chiama  la funzione InternetGetConnectedState dentro il segmento dei dati
mov [ebp+var_4], eax ; Sposta il valore restituito della chiamata di funzione nella variabile in [EBP-4]
CMP [EBP+var_4], 0 ; Confronta il valore restituito con 0
JZ short loc_40102B ; Passare a loc_40102B se il risultato del confronto è zero

push offset aSuccessInterne ; Inserire l'indirizzo di memoria della stringa "Operazione riuscita: connessione Internet" nello stack
call sub_40117F ; Chiamare la subroutine all'indirizzo di memoria sub_40117F
add esp, 4 ; Pulisci la pila aggiungendo 4 al puntatore dello stack (ESP)
mov eax, 1 ; Spostare il valore 1 nel registro EAX
JMP short loc_40103A; Vai alla loc_40103A
loc_40102B: ; Etichetta per il blocco di codice quando il risultato del confronto è zero
push l'offset aError1_1NoInte ; Inserire l'indirizzo di memoria della stringa "Errore 1.1: No Internet" nello stack
call sub_40117F ; Chiamare la subroutine all'indirizzo di memoria sub_40117F
add esp, 4 ; Pulire lo stack aggiungendo 4 al puntatore dello stack (ESP)
xor eax, eax ; Eseguire l'operazione XOR bit per bit su EAX con se stesso, impostandolo in modo efficace su zero

loc_40103A: ; Etichetta per il blocco di codice dopo i salti condizionali
MOV ESP, EBP ; Ripristinare il puntatore dello stack (ESP) al puntatore di base (EBP)
pop ebp ; Ripristinare il valore del puntatore di base (EBP) dallo stack
retn ; Ritorno dalla subroutine
sub_401000 endp ; Fine della subroutine in corrispondenza dell'indirizzo di memoria sub_401000

PIGNATELLO
GIUSEPPE

# GRAZIE

BY

PHANTOM SRL

**Prepared For :**
EPIC EDUCATION srl

**Liceria & Co.**
Siracusa, SR