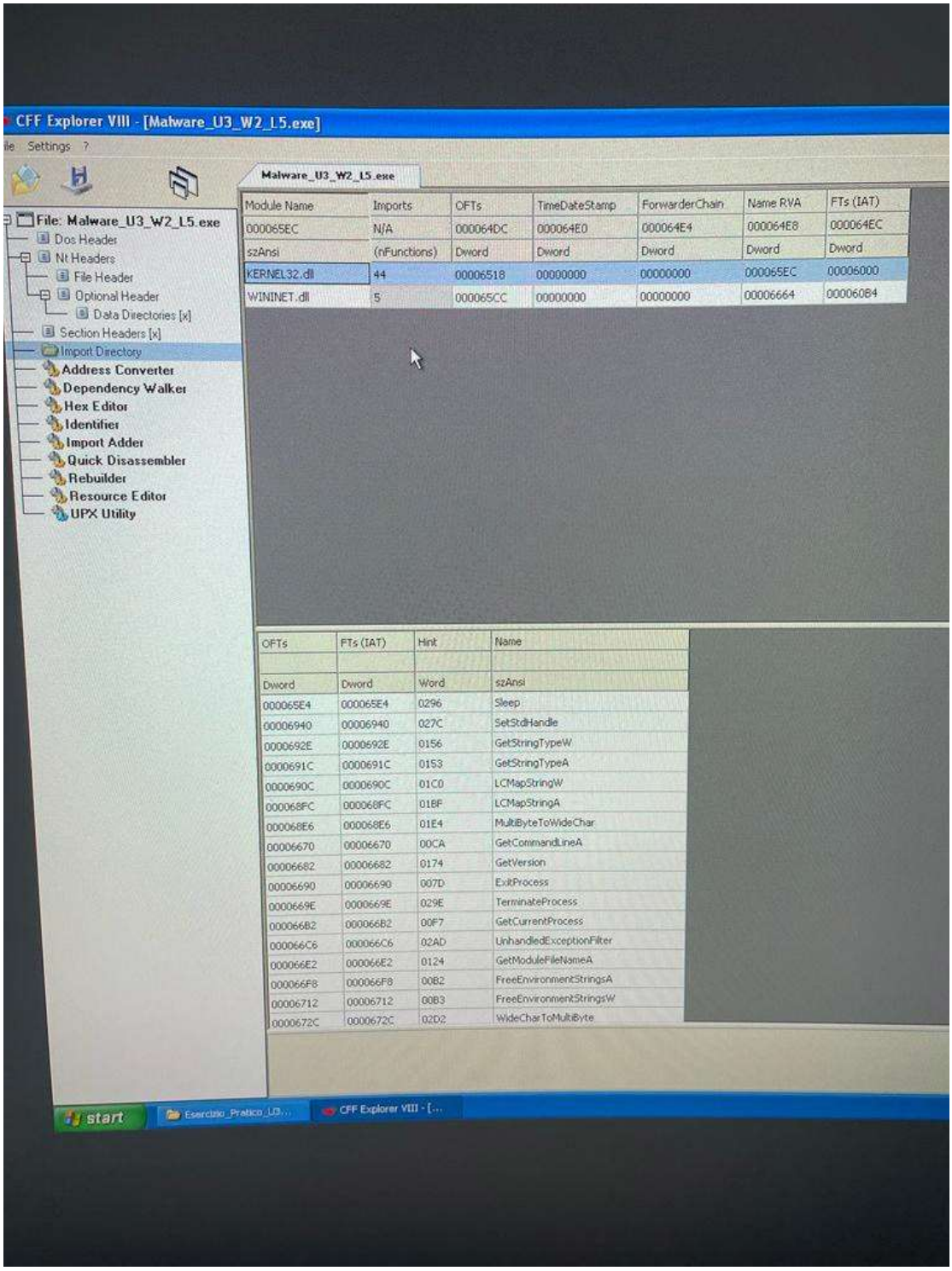
ANALISI MALWARE

Dato il file "Malware_u3_w2_l5.exe" tramite il programma "CFF EXPLORER" abbiamo fatto un'analisi per andare a recuperare informazioni importanti sul file quali:

1)LIBRERIE USATE



per andare a vedere che librerie vengono usate nel file andiamo nella sezione "Import Directory", dove vengono visualizzate li librerie usate, nel nostro caso abbiamo:

-**KERNEL32.dll**=La libreria "kernel32.dll" è una delle librerie principali del kernel di Windows e fornisce molte funzioni di basso livello necessarie per l'esecuzione dei programmi su Windows. Alcune delle funzionalità principali offerte da questa libreria

includono:

- Gestione dei processi e dei thread: La libreria kernel32 offre funzioni per creare, terminare e gestire i processi e i thread del sistema.

- Gestione dei file e dei dispositivi: Fornisce funzioni per la creazione, l'apertura, la lettura, la scrittura e la chiusura dei file, nonché per l'accesso a dispositivi di input/output.

- Gestione della memoria: Kernel32 fornisce funzioni per l'allocazione e la gestione della memoria del sistema.

- Gestione del tempo e dell'orario: Include funzioni per ottenere l'orario di sistema, misurare il tempo e impostare allarmi.

- Gestione delle librerie dinamiche (DLL): Offre funzioni per caricare e scaricare dinamicamente le librerie condivise (DLL) durante l'esecuzione di un programma.

-**WININET.dll**= La libreria "wininet.dll" è una libreria di sistema specifica per le funzionalità di connettività Internet di Windows. Questa libreria fornisce un'API (Application Programming Interface) per l'accesso e la gestione delle risorse Internet, consentendo alle applicazioni di comunicare tramite protocolli come HTTP, FTP e altri. Alcune delle funzionalità fornite da "wininet" includono:

- Accesso a risorse web: La libreria "wininet" offre funzioni per l'apertura, la lettura, la scrittura e la gestione di risorse web come pagine web, immagini e file scaricabili tramite protocollo HTTP.

- Supporto di protocolli: "wininet" supporta protocolli come HTTP, HTTPS, FTP, Gopher e File per l'accesso alle risorse Internet.

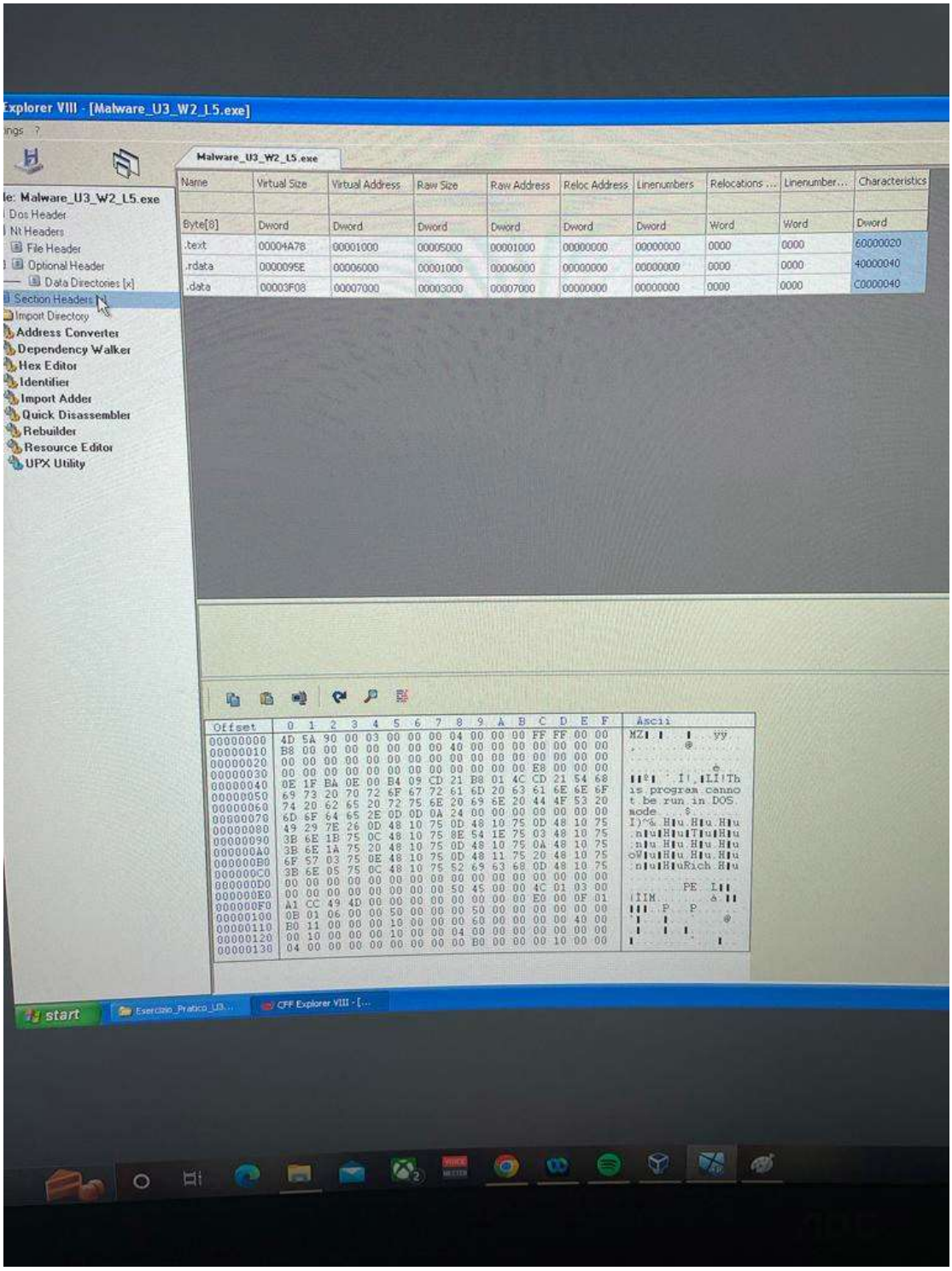
- Gestione delle sessioni di connessione: Fornisce funzioni per la gestione delle sessioni di connessione e delle richieste tra client e server.

- Gestione dei cookie: "wininet" offre funzionalità per la gestione dei cookie, consentendo alle applicazioni di inviare, ricevere e gestire i cookie durante la comunicazione con i server web.

- Proxy e configurazioni di connessione: Include funzioni per la gestione delle impostazioni di connessione proxy, consentendo alle applicazioni di configurare le impostazioni di rete necessarie per la comunicazione Internet.

2) SEZIONI DEL FILE

Per le sezioni del file invece rimanendo sempre sul programma usato in precedenza ci spostiamo nella sezione "Section Headers" dove ci verranno mostrete un elenco delle sezioni usate nel file.



le sezioni presenti sono:

.text=La sezione .tex è una sezione comune nei file eseguibili e viene utilizzata principalmente per contenere il codice eseguibile, ovvero le istruzioni in linguaggio macchina che vengono eseguite dalla CPU. Questa sezione contiene il codice effettivo del programma e rappresenta la parte eseguibile del file.

.rdata=La sezione .rdata (abbreviazione di "read-only data") è una sezione che contiene dati di sola lettura, cioè dati che non possono essere modificati durante l'esecuzione del programma. Questi dati sono spesso costanti, come stringhe, tabelle di lookup o altre strutture dati che vengono utilizzate dal programma. Essendo di sola lettura, i dati all'interno di questa sezione non possono essere modificati dal programma stesso.

.data=La sezione .data è una sezione che contiene dati modificabili durante l'esecuzione del programma. Questi dati includono variabili globali, variabili di stato, strutture di dati dinamiche e altri dati che possono essere modificati o aggiornati durante l'esecuzione del programma. A differenza della sezione .rdata, i dati all'interno di questa sezione possono essere letti e scritti dal programma.

3) IDENTIFICAZIONE COSTRUTTI

dato il seguente codice:

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

; Aggiunta della sezione Loc40102B
```

loc_40102B:

push offset aError1_noInte

call sub_40117F

add esp, 4

xor eax, eax

jmp short loc_40103A

loc_40103A:

mov esp, ebp

pop ebp

retn

sub_40117F:

; Implementazione della sotto-routine sub_ 40117F

Ora analizziamo i costrutti evidenti e ipotizziamo il comportamento:

1. "push ebp", "mov ebp, esp", "push ecx": Queste istruzioni fanno parte della configurazione iniziale dell'indicatore di frame e delle variabili locali per la funzione corrente.

2. "push 0; dwReserved", "push 0; ipdwFlags": Queste istruzioni mettono le costanti "0" nello stack. È probabile che questi valori vengano utilizzati come argomenti per la chiamata

di funzione successiva ("internetgetconnectedstate").

3. "call ds:internetgetconnectedstate": Questa istruzione chiama una funzione chiamata "internetgetconnectedstate" dal segmento di dati "ds". Presumibilmente, questa funzione è utilizzata per ottenere lo stato della connessione Internet.

4. "mov [ebp+var_4], eax": Questa istruzione copia il valore del registro eax nella memoria, in un'area specifica indicata da [ebp+var_4].

5. "cmp [ebp+var_4], 0", "jz short loc_40102B": Queste istruzioni confrontano il valore nella memoria (calcolato come [ebp+var_4]) con il valore "0". Se sono uguali, viene eseguito un salto ("jz") alla posizione "loc_40102B". Questo suggerisce che il codice sta verificando se il valore restituito da "internetgetconnectedstate" è uguale a zero e, in caso affermativo, passa alla sezione "loc_40102B" per gestire un errore.

6. "push offset aError1_noInte", "call sub_40117F", "add esp, 4", "xor eax, eax", "jmp short loc_40103A": Queste istruzioni fanno parte della sezione "Loc40102B". "push offset aError1 _noInte" mette l'indirizzo di una stringa denominata "aError1_noInte" nello stack, quindi "call sub_40117F" chiama una sotto-routine chiamata "sub_40117F". "add esp, 4" aggiusta lo stack pointer, "xor eax, eax" imposta il valore di eax a zero e infine "jmp short loc_ 40103A" salta a "loc_40103A".

7. "mov esp, ebp", "pop ebp", "retn": Queste istruzioni vengono eseguite alla fine della funzione. "mov esp, ebp" ripristina il valore di esp al valore originale di ebp, "pop ebp" ripristina il valore originale di ebp e infine "retn" restituisce il controllo al chiamante della funzione.

4)IPOTESI COMPORTAMENTO FILE

Il tipo di Malware analizzato sembrerebbe creare degli stack di memoria, ciò significa che ci troviamo d'avanti malware a stack overflow" o "malware a buffer overflow". Un stack overflow si verifica quando un malware o un altro software supera il limite di memoria allocata per lo stack di un programma. Lo stack è una regione di memoria utilizzata per gestire le chiamate di funzioni, le variabili locali e altre informazioni di stato durante l'esecuzione di un programma. Se un malware è in grado di sovrascrivere la memoria dello stack, può causare gravi problemi, come la corruzione dei dati, il crash del programma o persino l'esecuzione di codice dannoso. Un buffer overflow, in particolare, si verifica quando un malware scrive oltre i limiti di un buffer di memoria. Questo può causare danni al programma o consentire al malware di sovrascrivere altre porzioni di memoria, inclusi gli indirizzi di ritorno delle funzioni, consentendo al malware di eseguire codice malevolo. I malware che sfruttano queste vulnerabilità possono avere diversi obiettivi, come ottenere il controllo del sistema infetto, eseguire comandi dannosi, rubare dati sensibili o consentire l'accesso remoto al sistema.

5) REMEDIATION

1. Aggiornamenti del sistema e delle applicazioni: Assicurati di installare regolarmente gli

aggiornamenti del sistema operativo e delle applicazioni installate sul tuo dispositivo. Gli aggiornamenti spesso includono patch di sicurezza che risolvono le vulnerabilità note e riducono il rischio di attacchi di malware.

2. Utilizzo di software antivirus e antimalware: Installa un software antivirus o antimalware affidabile e mantienilo aggiornato. Questi strumenti possono rilevare e bloccare i malware noti, inclusi quelli che sfruttano vulnerabilità di stack o buffer overflow.

3. Protezione dei confini di rete: Implementa firewall, gateway e altre soluzioni di sicurezza di rete per filtrare il traffico indesiderato e prevenire l'accesso ai servizi o alle porte non autorizzate. Questo può aiutare a ridurre il rischio di attacchi che sfruttano le vulnerabilità di rete.

4. Validazione e sanificazione degli input: Assicurati che il tuo software esegua una valida validazione e sanificazione degli input. Questo può ridurre la probabilità di successo di attacchi di buffer overflow causati da input dannosi o non validi.

5. Utilizzo di linguaggi di programmazione sicuri: Utilizza linguaggi di programmazione che integrano misure di sicurezza per prevenire buffer overflow e altri problemi di sicurezza noti. Ad esempio, alcuni linguaggi come Rust e Ada implementano meccanismi di sicurezza integrati per gestire la memoria in modo più sicuro rispetto ad altri linguaggi come C o C++.

6. Analisi statica e dinamica del codice: Utilizza strumenti di analisi statica e dinamica del codice per identificare potenziali vulnerabilità e problemi di sicurezza nel software. Questi strumenti possono individuare aree di codice soggette a buffer overflow e fornire informazioni utili per applicare correzioni appropriate.

7. Best practices di sviluppo sicuro: Segui le best practices di sviluppo sicuro, come la gestione corretta della memoria, l'uso di funzioni di libreria sicure, la limitazione dei privilegi di accesso e la validazione dei dati di input.