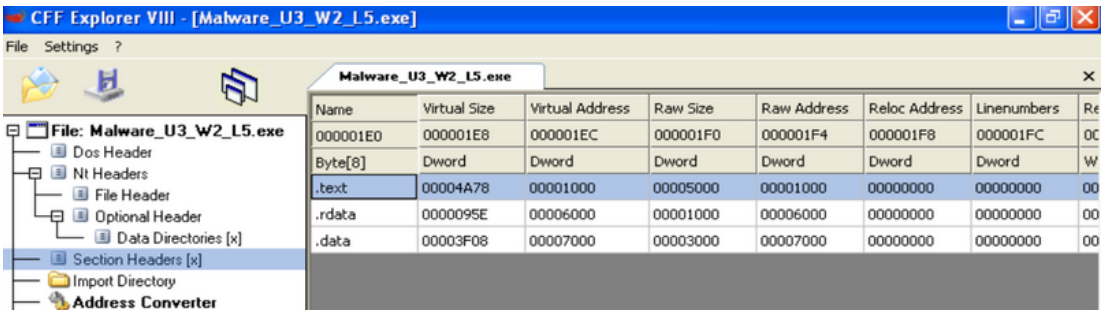
ANALISI STATICA

L'analisi statica basica di un malware è un processo fondamentale che consente agli analisti di sicurezza e ricercatori di comprendere le caratteristiche e il comportamento di un malware senza eseguirlo.
Questa analisi è veloce e fornisce informazioni preziose per sviluppare difese contro il malware e per comprendere la sua minaccia.
Come analisi statica ha come approccio lo studio di un malware senza un effettiva esecuzione. Questa forma di analisi è utile per individuare potenziali minacci in tempi ristretti ma può essere inefficiente contro i malware più sofisticati, per questo va sempre affiancata ad un analisi di tipo dinamico.

Come esercizio andremo ad analizzare un file Malware_U3_W2_L5 nella cartella «Esercizio_Pratico_U3_W2_L5 sulla nostra macchina virtuale Windows XP. Andremo a identificare quali librerie vengono importate dal file eseguibile e quali sono le sezioni di cui si compone il file eseguibile del malware.
Per far ciò utilizzeremo VirusTotal e CFF Explore.

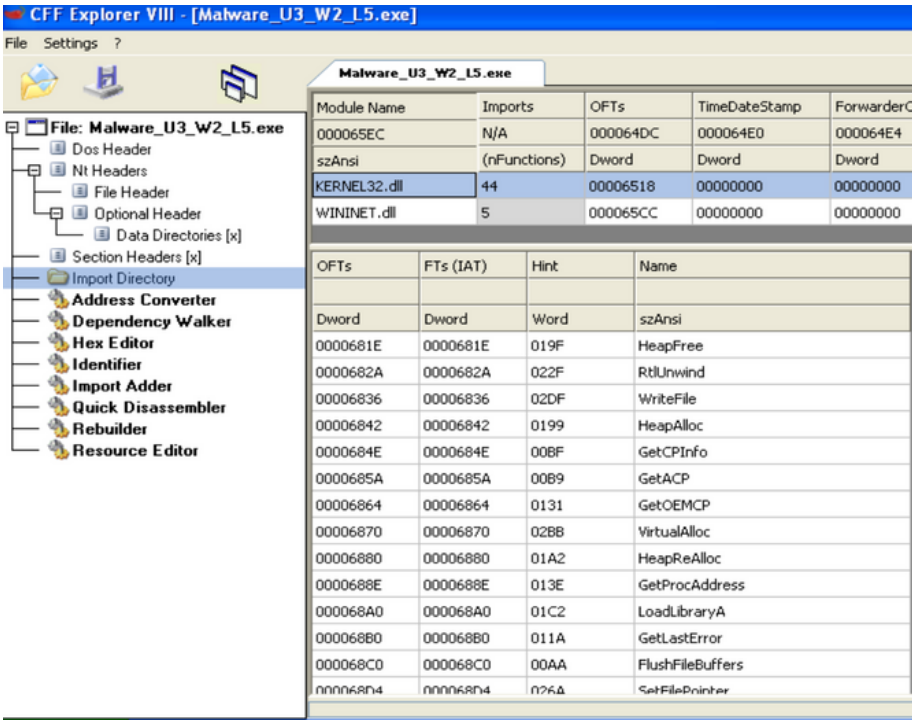
Dopo aver creato un ambiente sicuro (il nostro laboratorio virtuale) per prima cosa andremo ad analizzare le funzioni e le librerie (header e directory) importate ed esportate del malware con un tool chiamato CFF Explorer.
Con questo tool ora vedremo di preciso l'header PE che è una parte fondamentale della struttura di un file eseguibile nei sistemi operativi Windows.
L'analisi di questi header PE è un passo fondamentale poiché fornisce una panoramica completa della struttura interna del file e ci permette di capire quali sono le sezioni di cui si compone il software.

Come si può notare vediamo che il malware contiene le seguenti sezioni:
-.text: contiene le istruzioni del codice sorgente che verrà eseguito quando il malware verrà attivato.
-.rdata: contiene le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, può contenere stringhe di testo, indirizzi IP, chiavi di crittografia, o altri dati.
-.data: contiene tipicamente i dati utilizzati per inizializzare le variabili globali (accessibili da qualsiasi funzione all'interno dell'eseguibile) .

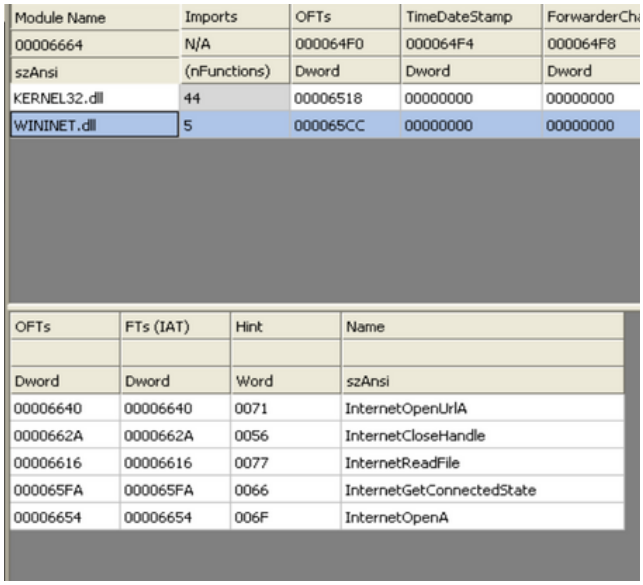


ANALISI STATICA

Ora andiamo a visualizzare quali sono e cosa fanno le librerie importate del malware.
Per prima cosa notiamo la libreria di nome KERNEL32.dll : Questa è una delle librerie di sistema principali di Microsoft Windows. Contiene numerose funzioni fondamentali che sono utilizzate dai programmi per interagire con il sistema operativo e svolgere varie operazioni di basso livello, come la gestione dei processi e dei thread oppure la Gestione dei File.
Inoltre notiamo che si utilizzano delle funzioni come «LoadLibrary» e «GetProcAddress».
In questa casistica, l'eseguibile richiama la libreria solamente quando necessità di una particolare funzione(Runtime).

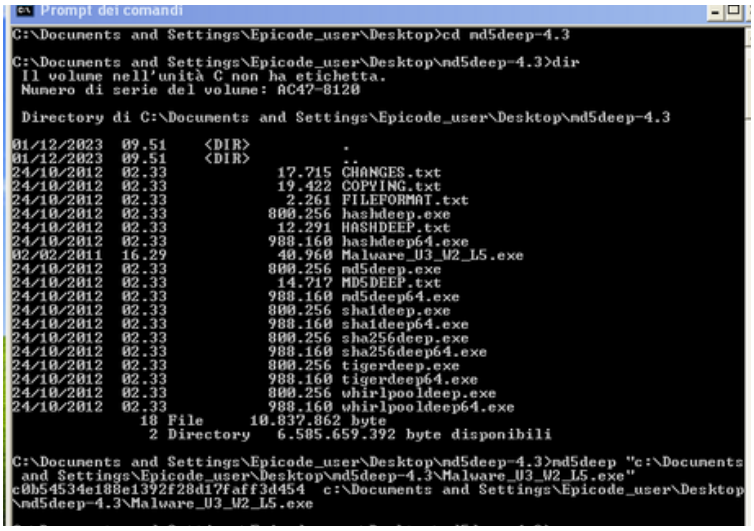


Come si può vedere il malware contiene un'altra libreria di nome Wininet.dll.
Questa libreria contiene le funzioni per l'implementazione di alcuni protocolli di rete come HTTP, FTP, NTP.
La presenza di wininet.dll in un malware può indicare che il malware sta cercando di sfruttare le funzionalità di rete del sistema operativo per scopi dannosi.
Ad esempio, potrebbe essere coinvolto in attività come il download di componenti aggiuntivi, l'invio di dati a un server remoto o altre attività di rete non autorizzate.



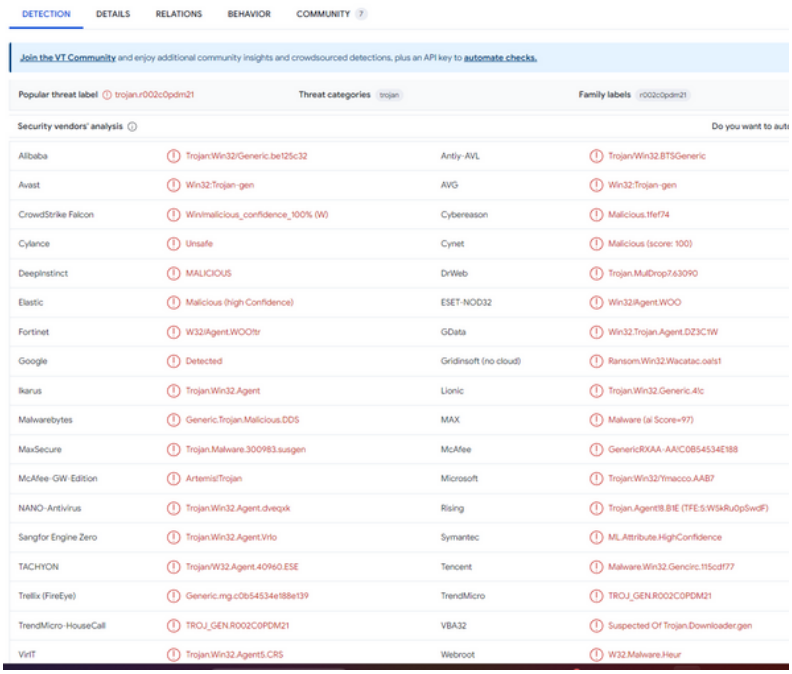
ANALISI STATICA

Andiamo ad effettuare un analisi più approfondita utilizzando VirusTotal.
VirusTotal è un servizio online gratuito che offre un'analisi antivirus e antimalware su file sospetti o maliziosi e consente agli utenti di caricare file, inserire URL, IP, Domain o file in hash per essere analizzati da una vasta gamma di motori antivirus e antimalware.
Noi abbiamo deciso di caricare il nostro malware su VirusTotal utilizzando l'hash.
Quindi per prima cosa andremo a creare il nostro codice hash in formato md5 del nostro malware utilizzando il tool md5deep come in figura.



Ora che abbiamo il nostro codice hash del malware possiamo inviarlo a VirusTotal e vedere che risultati ci lascia.

Come possiamo vedere il primo risultato che ci viene dato è una lista dei software che rilevano come il file sia realmente dannoso.
Inoltre alcuni di essi ci riportano una definizione più specifica riguardante il malware chiamandolo Trojan.
Un tipo di malware che si presenta come un file legittimo al fine di ingannare gli utenti e il sistema per ottenere l'accesso non autorizzato per poi ,durante l'esecuzione, importare codice malevolo.



Contacted URLs (6)

Scanned	Detections	Status	URL
2023-11-21	3 / 90	-	https://www.practicalmalwareanalysis.com/
2014-04-05	1 / 52	404	http://practicalmalwareanalysis.com/cc.htm
2023-11-15	2 / 90	404	http://www.practicalmalwareanalysis.com/cc.htm
2023-11-21	3 / 90	301	http://www.practicalmalwareanalysis.com/
2023-10-17	2 / 90	200	https://practicalmalwareanalysis.com/
2014-04-05	1 / 52	404	http://practicalmalwareanalysis.com/?post_type=feedback&p=374

Contacted Domains (14)

Domain	Detections	Created	Registrar
132.155.190.20.in-addr.arpa	1 / 88	-	-
150.32.88.40.in-addr.arpa	1 / 88	-	-
16.155.190.20.in-addr.arpa	1 / 88	-	-
24.78.0.192.in-addr.arpa	0 / 88	-	-
25.78.0.192.in-addr.arpa	1 / 88	-	-
29.91.21.72.in-addr.arpa	1 / 88	-	-
48.193.43.104.in-addr.arpa	1 / 88	-	-
login.live.com	1 / 88	1994-12-28	CSC CORPORATE DOMAINS, INC.
practicalmalwareanalysis.com	2 / 88	2011-01-22	GoDaddy.com, LLC
prda.aadg.msidentity.com	0 / 88	2016-03-21	MarkMonitor Inc.

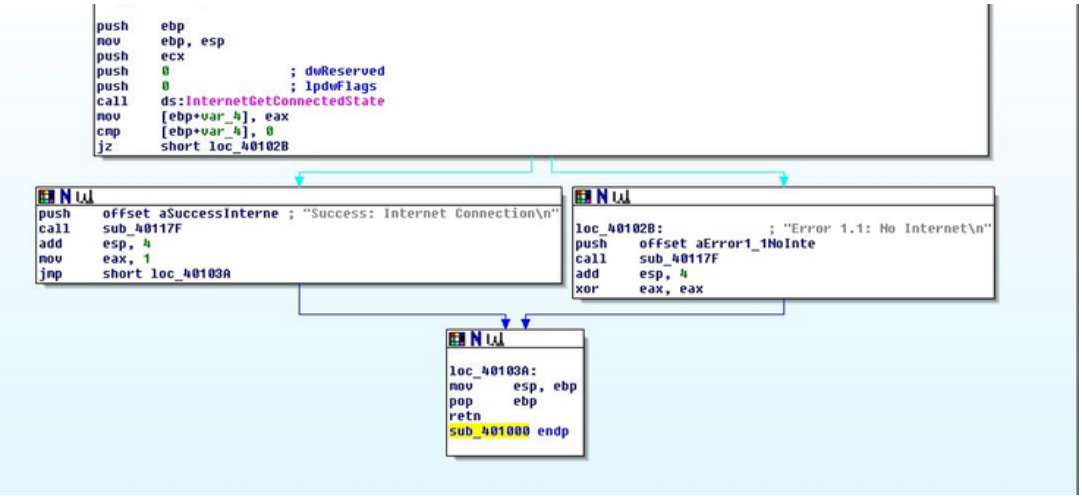
Contacted IP addresses (41)

IP	Detections	Autonomous System	Country
104.65.174.220	0 / 88	20940	US
104.80.88.81	0 / 88	20940	US
104.80.88.97	0 / 88	20940	US
104.98.118.138	0 / 88	20940	US
104.98.118.155	0 / 88	20940	US
104.99.72.226	0 / 88	20940	US
114.114.114.114	2 / 88	174	CN
13.107.4.50	4 / 88	8068	US
15.197.142.173	5 / 88	16509	US
192.0.78.24	1 / 88	2635	US

Si può vedere inoltre come questo malware, una volta avviato, cerchi connessioni all'esterno e addirittura abbiamo alcuni indirizzi IP ai quali si rivolge la connessione.

ANALISI CODICE ASSEMBLY X86

Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)
Ipotizzare il comportamento della funzionalità implementata



Il codice assembly x86 è un linguaggio di basso livello che rappresenta il set di istruzioni del processore x86. È una rappresentazione leggibile da parte dell'uomo delle istruzioni binarie eseguite direttamente dal processore. Il linguaggio assembly è specifico per l'architettura del processore e fornisce un'interfaccia più vicina all'hardware rispetto ai linguaggi di alto livello come C o Java.

Ipotesi sul comportamento:

- La funzione sembra verificare se c'è una connessione a Internet utilizzando InternetGetConnectedState.
- Se la connessione è presente, stampa un messaggio di successo e imposta il flag eax a 1.
- Se la connessione è assente, stampa un messaggio di errore e imposta il flag eax a 0.
- La funzione ritorna il valore del flag eax.

In sintesi, la funzione sembra essere progettata per controllare lo stato della connessione a Internet utilizzando la funzione **InternetGetConnectedState**. In base allo stato della connessione, **stampa** un messaggio appropriato e restituisce un valore tramite il registro **eax**.
Se c'è una connessione, **eax** viene impostato a 1; se la connessione è assente, **eax** viene impostato a 0.

ANALISI CODICE ASSEMBLY X86

Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)

```
// Creazione dello stack e inizializzazione dell'ambiente
push  ebp       ; Salva il valore corrente di ebp nello stack
mov   ebp, esp   ; Imposta ebp al valore corrente di esp
push ecx       ; Salva il valore corrente di ecx nello stack

// Chiamata alla funzione InternetGetConnectedState
push 0         ; dwReserved
push 0         ; lpdwFlags
call  ds:InternetGetConnectedState
mov  [ebp+var_4], eax ; Salva il risultato della chiamata in [ebp+var_4]

// Verifica dello stato della connessione      Costrutto IF
cmp [ebp+var_4], 0  ; Compara il risultato con 0 (connessione assente)
jz short loc_40102B  ; Salta a loc_40102B se la connessione è assente

// Caso di successo: connessione a Internet      ELSE
push offset aSuccessInterne ; "Success: Internet Connection\n"
call sub_40117F         ; Chiamata a una funzione (ipoteticamente per la stampa del messaggio)
add esp, 4              ; Aggiusta lo stack dopo la chiamata
mov eax, 1              ; Imposta eax a 1 (ipoteticamente un flag di successo)
jmp short loc_40103A     ; Salta a loc_40103A

// Caso di errore: nessuna connessione a Internet
loc_40102b:             ; "Error  1.1: No Internet\n"
push offset aError1_NoInte  ; Carica l'indirizzo della stringa di errore
call sub_40117F         ; Chiamata a una funzione (ipoteticamente per la stampa del messaggio)
add esp, 4              ; Aggiusta lo stack dopo la chiamata
xor eax, eax            ; Imposta eax a 0 (ipoteticamente un flag di errore)

// Fine della funzione
loc_40103A:
mov esp, ebp            ; Ripristina il valore di esp
pop ebp                 ; Ripristina il valore di ebp
retn                    ; Ritorna dalla funzione
sub_401000 endp         ; Fine della procedura
```