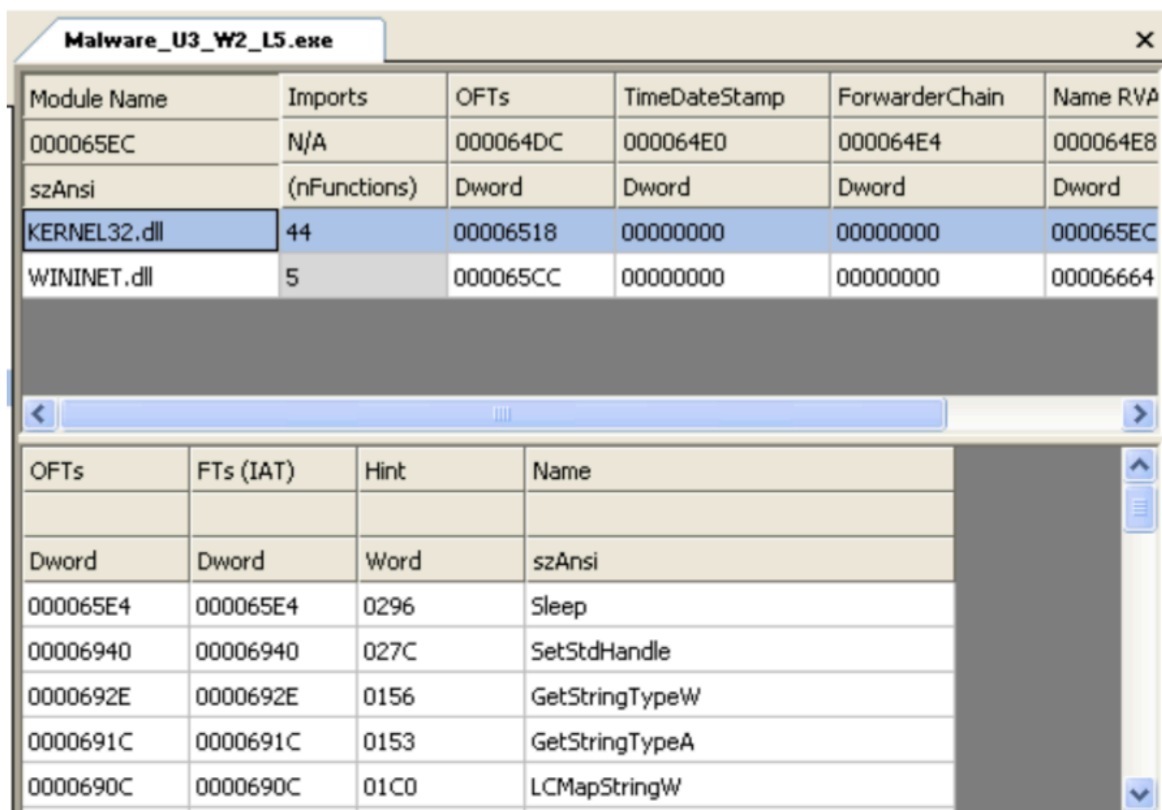


S10L5

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

SVOLGIMENTO:

Le librerie importate: Dal file eseguibile possono essere esaminate tramite CFF Explorer, selezionando l'opzione "Directory di importazione" dal menu principale situato sulla sinistra dello schermo.



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA
000065EC	N/A	000064DC	000064E0	000064E4	000064E8
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC
WININET.dll	5	000065CC	00000000	00000000	00006664

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW

KERNEL32.dll è una libreria a collegamento dinamico (DLL) essenziale nei sistemi operativi Microsoft Windows. Il suo nome, "kernel32", deriva da "kernel", che rappresenta il nucleo centrale di un sistema operativo. Questa DLL offre una vasta gamma di funzionalità cruciali per garantire il corretto funzionamento del sistema operativo Windows.

KERNEL32.dll, come già menzionato, rappresenta uno degli elementi fondamentali del sistema operativo Windows. La sua essenzialità deriva dalla vasta gamma di funzionalità che offre, le quali sono essenziali per garantire il corretto funzionamento di qualsiasi sistema Windows.

Uno dei principali ruoli di KERNEL32.dll riguarda la gestione della memoria. Questa libreria fornisce funzioni per l'allocazione e la liberazione della memoria, consentendo alle applicazioni di utilizzare e gestire in modo efficiente le risorse di memoria disponibili. Inoltre, KERNEL32.dll offre funzionalità per la gestione dei processi e dei thread, permettendo alle applicazioni di eseguire operazioni in parallelo e di gestire correttamente i processi in esecuzione.

Le operazioni di I/O sui file sono un'altra area cruciale supportata da KERNEL32.dll. Queste funzioni consentono alle applicazioni di leggere e scrivere file su disco, nonché di manipolare gli attributi dei file e dei dispositivi. Ciò è fondamentale per molte operazioni quotidiane, come la creazione, la modifica e l'accesso ai file sul sistema.

Inoltre, KERNEL32.dll svolge un ruolo chiave nella gestione degli errori e delle eccezioni. Fornisce funzioni per catturare e gestire gli errori durante l'esecuzione delle applicazioni, consentendo loro di gestire situazioni impreviste in modo appropriato e di garantire una maggiore affidabilità e stabilità del software.

La gestione delle DLL è un'altra importante responsabilità di KERNEL32.dll. Questa libreria gestisce il caricamento e lo scaricamento dinamico delle DLL, consentendo alle applicazioni di utilizzare librerie esterne per estendere le proprie funzionalità. Ciò favorisce la modularità del software e facilita lo sviluppo di applicazioni complesse e scalabili.

Il KERNEL32.dll offre funzionalità per recuperare informazioni di sistema, gestire date e ore, garantire la sicurezza e l'autenticazione, nonché per la gestione dei thread e la sincronizzazione.

quindi il KERNEL32.dll rappresenta il fulcro su cui si basa il sistema operativo Windows, fornendo una vasta gamma di funzionalità essenziali per il funzionamento delle applicazioni e per garantire un'esperienza utente fluida e affidabile. La sua importanza non può essere sottovalutata, poiché molte delle operazioni quotidiane eseguite su un sistema Windows dipendono direttamente dalle sue funzionalità.

WININET.dll è una libreria a collegamento dinamico (DLL) essenziale nei sistemi operativi Microsoft Windows. L'acronimo sta per Windows Internet Services e fornisce una serie di funzioni relative ai protocolli e alle comunicazioni Internet. Questa DLL è ampiamente utilizzata dalle applicazioni per gestire varie operazioni di

rete, come la connessione a Internet, il download di file e la gestione delle connessioni di rete.

Le principali funzionalità offerte da WININET.dll includono:

Operazioni HTTP e HTTPS: Supporta l'implementazione dei protocolli HTTP e HTTPS, permettendo alle applicazioni di inviare e ricevere dati sul web in modo sicuro.

Operazioni FTP: Facilita le operazioni FTP (File Transfer Protocol), consentendo alle applicazioni di caricare e scaricare file da e verso i server FTP.

Gestione degli URL: Aiuta ad analizzare e gestire gli URL (Uniform Resource Locator), permettendo alle applicazioni di lavorare con diversi tipi di indirizzi web in modo efficiente.

Gestione della cache: Gestisce la cache dei contenuti web per migliorare le prestazioni e ridurre la necessità di scaricare ripetutamente gli stessi dati, ottimizzando l'esperienza di navigazione.

Gestione dei cookie: Si occupa della gestione dei cookie, utilizzati per memorizzare informazioni sul lato client al fine di mantenere sessioni e preferenze degli utenti durante la navigazione web.

Protocolli di sicurezza: Supporta vari protocolli di sicurezza, come SSL/TLS, garantendo una comunicazione sicura su Internet e proteggendo i dati sensibili dagli attacchi esterni.

quindi il WININET.dll svolge un ruolo fondamentale nel consentire alle applicazioni Windows di accedere e utilizzare le risorse di rete in modo sicuro ed efficiente. Grazie alle sue funzionalità, le applicazioni possono offrire un'esperienza utente ottimale durante l'interazione con Internet e altri servizi di rete.

Le sezioni di cui si compone il malware:

Malware_U3_W2_L5.exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
000001E0	000001E8	000001EC	000001F0	000001F4	000001F8	000001FC	00000200	00000202	00000204
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

Come illustrato nell'immagine allegata, il malware in questione si suddivide in tre distinte sezioni, ciascuna svolgendo un ruolo cruciale nel funzionamento del programma dannoso.

La prima sezione, denominata .text, contiene il codice eseguibile. Questo segmento comprende le istruzioni che saranno interpretate e eseguite dalla CPU al momento dell'avvio del software infetto. In pratica, qui risiedono le direttive che definiscono le azioni e le operazioni da compiere per il corretto funzionamento del malware.

La seconda sezione, identificata come .rdata, ospita le informazioni relative alle librerie e alle funzioni utilizzate dal malware. Questo comprende sia le librerie importate che quelle esportate dal programma dannoso. In sostanza, la sezione .rdata fornisce al malware l'accesso alle funzionalità esterne necessarie per svolgere le sue attività dannose.

Infine, la terza sezione, chiamata .data, contiene le variabili globali del programma. Queste variabili sono accessibili da tutte le parti del codice e vengono utilizzate per memorizzare dati cruciali per il funzionamento del malware. Essi possono includere informazioni quali configurazioni, parametri di attivazione o qualsiasi altra informazione dinamica necessaria per il comportamento del software dannoso.

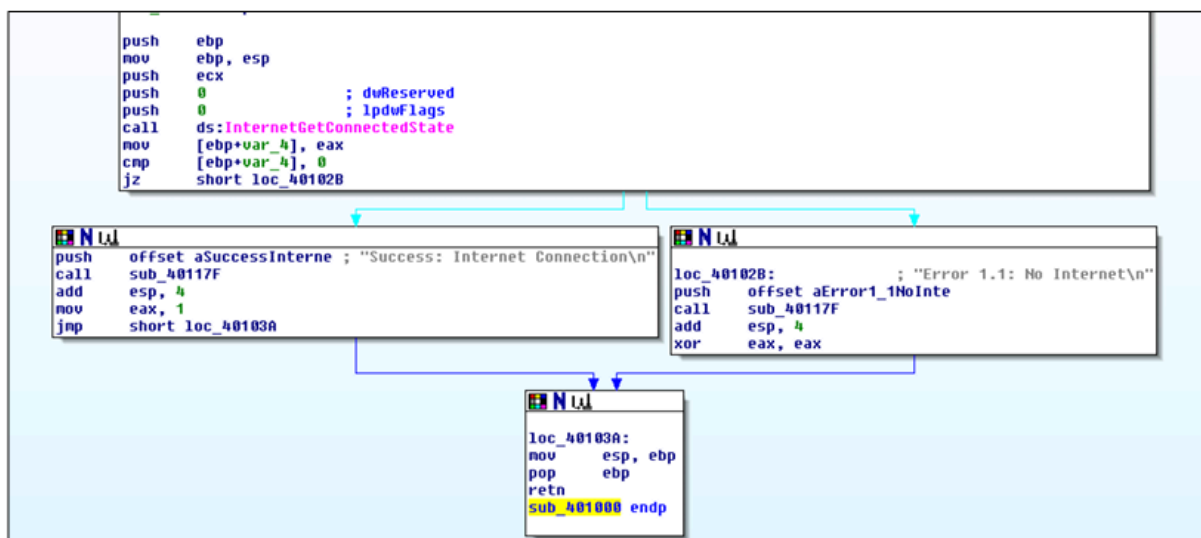
Per ottenere alcune informazioni aggiuntive ho effettuato uno step ulteriore che fa sempre parte dell'analisi statica basica, ovvero ho estrapolato l'hash del file in questione e l'ho inserito su VirusTotal per un'analisi preliminare.

Popular threat label ⓘ trojan.r002c0pdm21		Threat categories trojan		Family labels r002c0pdm21	
Security vendors' analysis ⓘ				Do you want to automate checks?	
Alibaba	ⓘ Trojan:Win32/Generic.be125c32	Antiy-AVL	ⓘ Trojan/Win32.BTSGeneric		
Avast	ⓘ Win32:Trojan-gen	AVG	ⓘ Win32:Trojan-gen		
CrowdStrike Falcon	ⓘ Win/malicious_confidence_100% (W)	Cybereason	ⓘ Malicious.1fef74		
Cylance	ⓘ Unsafe	Cynet	ⓘ Malicious (score: 100)		
DeepInstinct	ⓘ MALICIOUS	DrWeb	ⓘ Trojan.MulDrop7.63090		
Elastic	ⓘ Malicious (high Confidence)	ESET-NOD32	ⓘ Win32/Agent.WOO		
Fortinet	ⓘ W32/Agent.WOO!tr	GData	ⓘ Win32.Trojan.Agent.DZ3C1W		
Google	ⓘ Detected	Gridinsoft (no cloud)	ⓘ Ransom.Win32.Wacatac.oals1		
Ikarus	ⓘ Trojan.Win32.Agent	Lionic	ⓘ Trojan.Win32.Generic.4lc		
Malwarebytes	ⓘ Generic.Trojan.Malicious.DDS	MAX	ⓘ Malware (ai Score=97)		
MaxSecure	ⓘ Trojan.Malware.300983.susgen	McAfee	ⓘ GenericRXAA-AAIC0B54534E1B8		
McAfee-GW-Edition	ⓘ Artemis!Trojan	Microsoft	ⓘ Trojan:Win32/Ymacco.AAB7		

L'identificazione del file come trojan da diverse fonti, come mostrato nell'immagine, fornisce una conferma aggiuntiva sulla sua natura dannosa.

Ho effettuato anche un'analisi delle stringhe presenti all'interno del file, tuttavia i risultati sono piuttosto limitati e non forniscono molte informazioni, tranne il fatto che il file sembra tentare di stabilire una connessione a Internet.

i costrutti noti :



Nel codice analizzato sono identificabili diversi costrutti noti che svolgono ruoli specifici nel programma:

Costrutto dello stack: Questo costrutto è evidenziato dalle istruzioni di push utilizzate per inserire elementi nello stack e dall'utilizzo dei registri EBP e ESP per indicare rispettivamente la base e la cima dello stack. Si tratta della struttura di base per l'allocazione di dati e il mantenimento dello stato durante l'esecuzione del programma.

Costrutto dell'if: Il secondo costrutto rappresenta un'istruzione condizionale, evidenziata dal confronto tra due valori seguito dall'utilizzo dell'istruzione di salto condizionale jz (jump zero). Questo costrutto determina la diramazione del flusso del programma in base al risultato del confronto, eseguendo un ramo di codice se la condizione è vera e un altro se è falsa.

Chiusura dello stack: Un altro costrutto identificato è quello relativo alla chiusura dello stack, simile all'apertura ma con l'utilizzo inverso dei puntatori e dell'istruzione pop per rimuovere gli elementi. Questo costrutto è importante per mantenere l'integrità dello stack e liberare la memoria allocata.

Identificazione dei rami del vero e del falso: È possibile distinguere quale parte del codice corrisponde al ramo del vero e quale al ramo del falso. Il ramo del vero è caratterizzato da un salto verso la chiusura dello stack, mentre il ramo del falso presenta un salto iniziale nel caso in cui il confronto sia falso. Questa distinzione è utile per comprendere il flusso di esecuzione del programma in base alle condizioni.

Chiamata alla funzione InternetGetConnectedState: Infine, è identificabile il costrutto relativo alla chiamata della funzione InternetGetConnectedState e all'inizializzazione dei suoi parametri nelle righe precedenti. Questo costrutto è cruciale per il funzionamento del programma, in quanto coinvolge l'interazione con il sistema operativo per verificare lo stato della connessione a Internet.

Ipotizzo il comportamento della funzionalità:

Nel codice fornito, l'uso dell'istruzione jmp per eseguire un salto incondizionato a un altro indirizzo è evidente. Ad esempio, viene utilizzato jmp short loc_40103A per bypassare il blocco di codice relativo alla stampa del messaggio di errore e saltare direttamente alla fine della funzione.

La pulizia dello stack è eseguita tramite le istruzioni mov esp, ebp e pop ebp. Queste istruzioni sono cruciali per ripristinare il valore del registro esp e recuperare il valore del registro ebp dallo stack, garantendo così un corretto ripristino dello stato dello stack dopo l'esecuzione della funzione.

Il termine della funzione è segnato dall'utilizzo di retn, il quale indica il ritorno al chiamante. Questa istruzione consente al flusso del programma di proseguire da dove è stato invocato il codice, saltando all'indirizzo salvato sullo stack dalla precedente chiamata (call).

Basandoci sulle operazioni eseguite nel codice, possiamo formulare un'ipotesi sul comportamento della funzionalità implementata. Il codice chiama la funzione InternetGetConnectedState, la quale restituisce un valore booleano in eax che indica la presenza o l'assenza di una connessione internet. Se il valore restituito è diverso da zero, ciò indica la presenza di una connessione internet, e il codice salta all'etichetta loc_40101F, dove viene visualizzato il messaggio "Success: Internet Connection". Successivamente, la funzione termina con un valore di ritorno pari a

zero. Al contrario, se il valore restituito è zero, segnalando l'assenza di connessione internet, il codice salta all'etichetta loc_40102B, dove viene stampato il messaggio "Error1.1: No Internet". Anche in questo caso, la funzione termina con un valore di ritorno pari a zero.

Bonus

Istruzione	Significato
Push ebp	Inizializza lo stack inserendo il primo valore
Mov ebp, esp	Copia il valore di esp in ebp
Push ecx	Salva il valore di ecx sulla pila dello stack
Push 0; dwReserved	Aggiunge il valore 0 alla pila, rappresentando l'argomento dwReserved
Push 0; lpdwFlags	Come sopra ma per l'argomento lpdwFlags
Call ds:InternetGetConnectedState	Chiama la funzione InternetGetConnectedState utilizzando il segmento dati ds, inizializzando i parametri a 0
Mov [ebp+var_4], eax	Copia il contenuto di eax nella posizione di memoria indicata da var_4
Cmp [ebp+var_4], 0	Confronta il valore copiato con 0 e imposta lo Zero Flag di conseguenza
Jz short loc_40102B	Effettua un salto condizionale all'indirizzo indicato se lo Zero Flag è impostato a 1

Push offset aSuccessInterne	Inserisce la stringa "Success: Internet Connection\n" nella variabile aSuccessInterne
Call sub_40117F	Chiama la subroutine corrispondente all'etichetta sub_40117F
Add esp, 4	Aggiunge 4 al puntatore dello stack esp, solitamente utilizzato per pulire la pila dopo una chiamata di funzione
Mov eax, 1	Assegna il valore 1 al registro eax
Jmp short loc_40103A	Effettua un salto incondizionato all'indirizzo indicato
Push offset aError1_1NoInte	Inserisce la stringa "Error 1.1: No Internet\n" nella variabile aError1_1NoInte
Call sub_40117F	Chiama la subroutine corrispondente all'etichetta sub_40117F
Add esp, 4	Aggiunge 4 al puntatore dello stack esp, solitamente utilizzato per pulire la pila dopo una chiamata di funzione
Xor eax, eax	Esegue un'operazione XOR tra eax e sé stesso, azzerandolo
Mov esp, ebp	Copia il valore di ebp in esp
Pop ebp	Estrae il valore dalla pila, ripristinando così il frame di stack precedente
Retn	Corrisponde al return della funzione
Sub_401000 endp	Indica la fine della funzione o subroutine denominata Sub_401000

Marco Fasani