Analisi statica e dinamica: Un approccio pratico

Oggi esamineremo un malware specifico che ha infettato un sistema operativo Windows XP. Per svolgere questa analisi, useremo uno strumento chiamato CFF Explorer. CFF Explorer ci fornirà una panoramica dettagliata delle caratteristiche del malware, consentendoci di esaminare attentamente i seguenti punti essenziali:

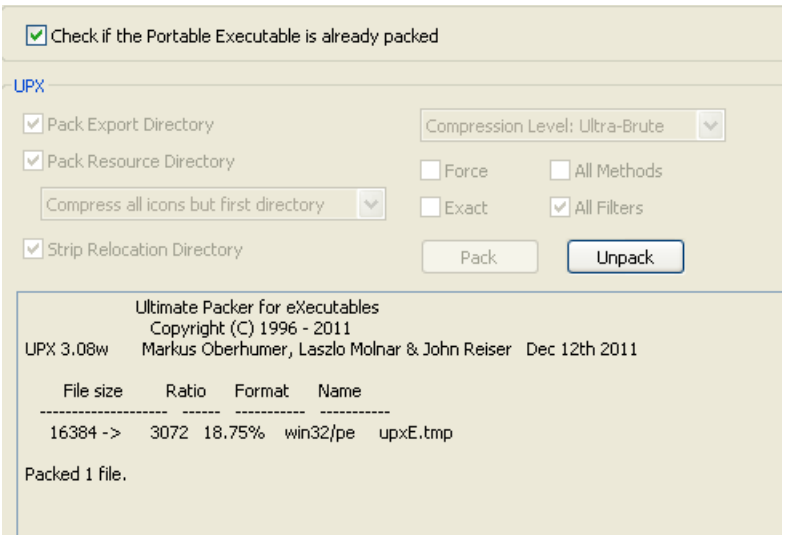
1. Quali librererie vengono importate dal file eseguibile? Per rispondere alla domanda sulle librerie importate dal file eseguibile utilizzando il programma CFF Explorer, esamineremo la cartella "Import Directory" e identificheremo le librerie coinvolte. Nel caso specifico, abbiamo individuato due librerie chiamate KERNEL32.DLL e WINNET.DLL.

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

KERNEL32.DLL: Questa è una libreria di sistema essenziale di Windows che fornisce una vasta gamma di funzioni di basso livello per l'interazione con il sistema operativo. Contiene molte funzioni comuni come la gestione della memoria, la gestione dei file, la creazione di processi, la gestione delle eccezioni, l'accesso alle risorse di sistema e molto altro. È ampiamente utilizzata da molti programmi Windows e fornisce un'interfaccia tra l'applicazione e il sistema operativo.

WINNET.DLL: Questa libreria potrebbe non essere una libreria di sistema standard di Windows. Potrebbe essere specifica dell'applicazione o del malware in questione. Senza ulteriori informazioni specifiche sulla libreria WINNET.DLL, è difficile fornire dettagli precisi sulle sue funzionalità. In generale, le librerie personalizzate possono essere create per fornire funzionalità aggiuntive o specifiche per un'applicazione o un malware particolare.

Per ottenere una maggiore comprensione del malware in questione, possiamo utilizzare un comando specifico per estrarre informazioni rilevanti. L'utilizzo di questo comando ci consentirà di acquisire dettagli cruciali sul funzionamento del malware e potrebbe rivelare informazioni sul suo scopo e sulle sue capacità.



Dopo aver eseguito l'unpacking del malware, abbiamo analizzato le librerie coinvolte e abbiamo scoperto la presenza di diverse sezioni, tra cui .text, .data e .rdata, all'interno del percorso "sections header".

Le sezioni sono parti del file eseguibile che organizzano e separano diverse porzioni di dati o codice. Ogni sezione ha uno scopo specifico e svolge una funzione particolare nel contesto dell'esecuzione del programma.

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

.text: Questa sezione contiene il codice eseguibile del programma. È dove sono presenti le istruzioni di macchina che vengono eseguite dal processore. La sezione .text contiene il codice del programma, inclusi i blocchi di istruzioni per le funzioni, le routine e le logiche di controllo.

.data: Questa sezione contiene dati inizializzati. Qui vengono allocate variabili globali e dati che devono essere memorizzati in modo persistente durante l'esecuzione del programma. Ad esempio, può contenere variabili globali, costanti o tabelle di lookup.

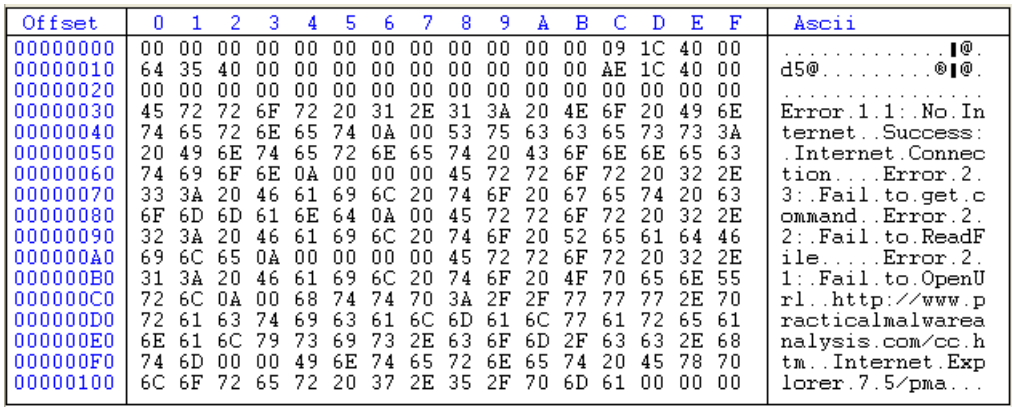
.rdata: Questa sezione contiene dati di sola lettura. Questi dati sono solitamente costanti o stringhe che vengono utilizzate dal programma ma non possono essere modificate durante l'esecuzione. Ad esempio, possono essere presenti messaggi di errore, stringhe di testo o altre costanti utilizzate nell'applicazione.

Le sezioni .text, .data e .rdata sono comuni nei file eseguibili e consentono di organizzare e gestire il codice e i dati nel programma. L'analisi di queste sezioni può fornire importanti informazioni sulle funzionalità del malware, sui dati utilizzati e sui punti di ingresso nel codice eseguibile.

Nell'ambito dell'analisi dei malware, l'esame delle sezioni può rivelare informazioni cruciali sul

comportamento del malware, sugli obiettivi dell'attaccante e sulle eventuali risorse o vulnerabilità sfruttate dal malware.

Nel .data vediamo:



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	00	00	00	00	00	00	00	00	00	00	00	00	09	1C	40	00	.....@.
00000010	64	35	40	00	00	00	00	00	00	00	00	00	AE	1C	40	00	d5@.....®@.
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	45	72	72	6F	72	20	31	2E	31	3A	20	4E	6F	20	49	6E	Error.1.1:.No.In
00000040	74	65	72	6E	65	74	0A	00	53	75	63	63	65	73	73	3A	ternet..Success:
00000050	20	49	6E	74	65	72	6E	65	74	20	43	6F	6E	6E	65	63	.Internet.Connec
00000060	74	69	6F	6E	0A	00	00	00	45	72	72	6F	72	20	32	2E	tion....Error.2.
00000070	33	3A	20	46	61	69	6C	20	74	6F	20	67	65	74	20	63	3:.Fail.to.get.c
00000080	6F	6D	6D	61	6E	64	0A	00	45	72	72	6F	72	20	32	2E	ommand..Error.2.
00000090	32	3A	20	46	61	69	6C	20	74	6F	20	52	65	61	64	46	2:.Fail.to.ReadF
000000A0	69	6C	65	0A	00	00	00	00	45	72	72	6F	72	20	32	2E	ile.....Error.2.
000000B0	31	3A	20	46	61	69	6C	20	74	6F	20	4F	70	65	6E	55	1:.Fail.to.OpenU
000000C0	72	6C	0A	00	68	74	74	70	3A	2F	2F	77	77	77	2E	70	rl..http://www.p
000000D0	72	61	63	74	69	63	61	6C	6D	61	6C	77	61	72	65	61	racticalmalwarea
000000E0	6E	61	6C	79	73	69	73	2E	63	6F	6D	2F	63	63	2E	68	nalysis.com/cc.h
000000F0	74	6D	00	00	49	6E	74	65	72	6E	65	74	20	45	78	70	tm..Internet.Exp
00000100	6C	6F	72	65	72	20	37	2E	35	2F	70	6D	61	00	00	00	lorer.7.5/pma...

per rispondere la secondo quesito dell'esercizio andremmo a cercare sempre nallo stesso percorso ma andiamo a vede le sezioni richimate dalle librerie che abbiamo trovato.

KERNELL32.DLL

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000066F8	000066F8	00B2	FreeEnvironmentStringsA
00006712	00006712	00B3	FreeEnvironmentStringsW
0000672C	0000672C	02D2	WideCharToMultiByte
00006742	00006742	0106	GetEnvironmentStrings
0000675A	0000675A	0108	GetEnvironmentStringsW
00006774	00006774	026D	SetHandleCount
00006786	00006786	0152	GetStdHandle
00006796	00006796	0115	GetFileType
000067A4	000067A4	0150	GetStartupInfoA
000067B6	000067B6	0126	GetModuleHandleA
000067CA	000067CA	0109	GetEnvironmentVariableA
000067E4	000067E4	0175	GetVersionExA
000067F4	000067F4	019D	HeapDestroy
00006802	00006802	019B	HeapCreate
00006810	00006810	02BF	VirtualFree
0000681E	0000681E	019F	HeapFree
0000682A	0000682A	022F	RtlUnwind
00006836	00006836	02DF	WriteFile
00006842	00006842	0199	HeapAlloc
0000684E	0000684E	00BF	GetCPInfo
0000685A	0000685A	00B9	GetACP
00006864	00006864	0131	GetOEMCP
00006870	00006870	02BB	VirtualAlloc
00006880	00006880	01A2	HeapReAlloc
0000688E	0000688E	013E	GetProcAddress
000068A0	000068A0	01C2	LoadLibraryA
000068B0	000068B0	011A	GetLastError
000068C0	000068C0	00AA	FlushFileBuffers
000068D4	000068D4	026A	SetFilePointer
00006950	00006950	001B	CloseHandle

WINNET.DLL

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006640	00006640	0071	InternetOpenUrlA
0000662A	0000662A	0056	InternetCloseHandle
00006616	00006616	0077	InternetReadFile
000065FA	000065FA	0066	InternetGetConnectedState
00006654	00006654	006F	InternetOpenA

Ecco una spiegazione di otto di alcune funzioni, presenti nella libreria **KERNEL32.DLL**:

1. GetEnvironmentStrings: Questa funzione restituisce un puntatore a una stringa che rappresenta l'intero ambiente delle variabili di ambiente del processo corrente. Questo include informazioni come le variabili di sistema, le variabili utente e altre impostazioni dell'ambiente.

2. SetHandleCount: Questa funzione consente di impostare il numero massimo di handle di file aperti che un processo può mantenere. Gli handle di file sono utilizzati per accedere a file, porte di comunicazione, dispositivi di input/output e altre risorse di sistema.

3. GetStdHandle: Questa funzione restituisce l'handle di un file standard predefinito, come l'input standard, l'output standard o l'errore standard, associato al processo corrente.

4. GetFileType: Questa funzione restituisce il tipo di file associato a un determinato handle. Può essere utilizzata per determinare se un handle fa riferimento a un file, a un dispositivo o a una pipe.

5. GetStartupInfoA: Questa funzione recupera le informazioni di avvio del processo corrente, inclusi il nome del file eseguibile, i flag di creazione del processo e le impostazioni relative alle finestre.

6. GetEnvironmentVariableA: Questa funzione restituisce il valore di una variabile di ambiente specificata. Può essere utilizzata per recuperare informazioni o impostazioni specifiche dell'ambiente.

7. HeapAlloc: Questa funzione alloca una quantità specifica di memoria dal heap del processo. Il heap è una regione di memoria gestita dal sistema operativo e utilizzata per l'allocazione dinamica di memoria.

8. VirtualAlloc: Questa funzione alloca una quantità specifica di memoria virtuale all'interno dello spazio di indirizzamento del processo. La memoria virtuale può essere utilizzata per scopi come l'allocazione di grandi blocchi di memoria o la mappatura di file nel processo.

Queste sono solo alcune delle funzioni presenti nella libreria KERNEL32.DLL. Ogni funzione ha un ruolo specifico e può essere utilizzata per svolgere compiti diversi all'interno di un'applicazione o di un sistema operativo Windows.

Ecco una spiegazione delle 5 funzioni presenti nella libreria **WINNET.DLL** :

1. InternetOpenA: Questa funzione inizializza una sessione di accesso a Internet. Restituisce un handle che identifica la sessione aperta, che sarà utilizzato come parametro nelle altre funzioni dell'API WinINet.

2. InternetOpenUrlA: Questa funzione apre una connessione a una risorsa specifica su Internet. Prende in input l'handle della sessione aperta tramite InternetOpenA e l'URL della risorsa desiderata. Restituisce un handle che rappresenta la connessione aperta. Questo handle sarà successivamente utilizzato per altre operazioni sulla connessione, come la lettura dei dati dalla risorsa o la chiusura della connessione.

3. InternetCloseHandle: Questa funzione chiude un handle aperto da InternetOpenA o InternetOpenUrlA. Viene utilizzata per rilasciare le risorse associate alla connessione o alla sessione aperta. È importante chiudere correttamente gli handle per evitare perdite di memoria o risorse.

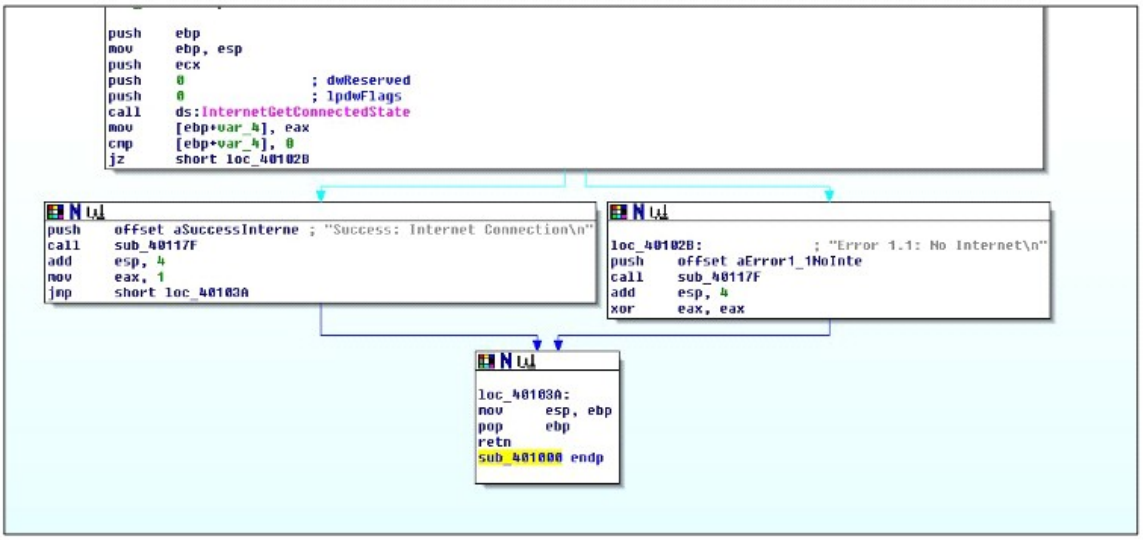
4. InternetReadFile: Questa funzione legge dati dalla risorsa Internet specificata. Prende in input l'handle della connessione aperta tramite InternetOpenUrlA e un buffer in cui verranno memorizzati i dati letti. Restituisce un valore booleano che indica se l'operazione di lettura è stata completata con successo o meno.

5. InternetGetConnectedState: Questa funzione restituisce informazioni sullo stato della connessione di rete. È utilizzata per determinare se il sistema è connesso a Internet e se è disponibile una connessione attiva. Restituisce un valore booleano che indica se il sistema è connesso a Internet o meno.

In sintesi, queste funzioni della libreria **WININET.DLL** forniscono un'interfaccia per l'accesso a risorse su Internet e la gestione delle connessioni di rete all'interno di applicazioni Windows. Consentono di aprire una sessione di accesso a Internet, aprire una connessione a una risorsa specifica, leggere dati da una risorsa Internet e ottenere informazioni sullo stato della connessione di rete.

Ora passiamo all' analisi dell'assemlìbly dato dal prof. qui dobbimao semplicemente trovare i cicli e

spiegare a grandi linee che cosa fa il seguente codice in assembly:



Dal codice fornito, possiamo identificare i seguenti costrutti noti:

1. Creazione dello stack:

- L'istruzione `push ebp` salva il valore corrente del registro di base del puntatore (EBP) nello stack.

- L'istruzione `mov ebp, esp` copia il valore dello stack pointer (ESP) nel registro di base del puntatore (EBP), creando un nuovo frame di stack per la funzione corrente.

- L'istruzione `push ecx` salva il valore corrente del registro ECX nello stack.

2. Chiamata di funzione:

- L'istruzione `push call [ebp+uar], eax` esegue una chiamata di funzione all'indirizzo memorizzato in `[ebp+uar]`, salvando il valore di ritorno in EAX.

- L'istruzione `call sub_40117F` esegue una chiamata di funzione all'indirizzo `sub_40117F`.

3. Operazioni di confronto e salto condizionato:

- L'istruzione `jz NW` esegue un salto condizionato a NW se il risultato precedente dell'operazione di confronto è zero.

4. Gestione dello stack:

- L'istruzione `add esp, 4` aggiusta il puntatore dello stack (ESP) per rimuovere il valore

precedentemente spinto nello stack.

- L'istruzione `esp, ebp` copia il valore di EBP nello stack pointer (ESP).

- L'istruzione `pop ebp` ripristina il valore originale del registro di base del puntatore (EBP) dallo stack.

Non sono presenti cicli o altri costrutti di iterazione (**come `jmp`, `jz`, `je`, `jne`, `jg`, `jl`, `loop`,** ecc.) nel codice fornito. Tuttavia, potrebbero esserci altre parti del codice mancanti che includono tali costrutti.

4.Ipotesi sul comportamento della funzionalità implementata:

- Il codice coinvolge l'accesso a Internet.
- Utilizza "InternetGetConnectedState" per interrogare lo stato di connessione di rete.
- In base al valore restituito, viene eseguito un percorso diverso nel programma.
- Se la connessione è disponibile, viene visualizzata la stringa "Success: Internet Connection".
- Se la connessione non è disponibile, viene visualizzata la stringa "Error 1.1: No Internet".
- Dopo l'esecuzione del blocco corrispondente, il programma termina con "retn"

Tuttavia, è importante sottolineare che senza informazioni aggiuntive sul contesto completo del codice assembly e senza avere accesso alle eventuali parti mancanti, le ipotesi formulate precedentemente sul comportamento e le funzionalità del codice potrebbero non essere del tutto accurate.

L'analisi di un codice assembly richiede una comprensione completa del contesto in cui viene utilizzato, inclusi i dati di input, le dipendenze di librerie esterne, le configurazioni di sistema e le specifiche delle funzionalità desiderate. Senza queste informazioni, è difficile trarre conclusioni definitive sulle funzionalità implementate o sul comportamento specifico del codice.

Pertanto, è fondamentale disporre di ulteriori dettagli sul contesto e sulle parti mancanti del codice assembly per fornire un'analisi più esaustiva e accurata. Solo con queste informazioni aggiuntive sarà possibile comprendere appieno le funzionalità e il comportamento del codice assembly.

BONUS:

Come membro senior del SOC, è necessario fornire una spiegazione tecnica convincente al dipendente per dimostrare che il file "IEXPLORE.EXE" presente nella cartella "C:\Program Files\Internet Explorer" non è maligno. Utilizzeremo strumenti di analisi statica basica e analisi dinamica basica per valutare l'integrità e il comportamento del file.

1. Analisi Statica Basica:

- Verificheremo l'integrità del file "**IEXPLORE.EXE**" controllando la firma digitale, la dimensione e i metadati del file.

- Confronteremo l'hash del file con fonti attendibili o database di hash noti per identificare eventuali modifiche o compromissioni del file.

- Effettueremo una scansione antivirus utilizzando motori di rilevamento multipli per individuare

eventuali segnalazioni di malware noto associato al file.

2. Analisi Dinamica Basica:

- Eseguiamo il file "**IEXPLORE.EXE**" in un ambiente controllato o in una sandbox, dove le azioni del programma possono essere monitorate senza influire sul sistema operativo principale.

- Utilizziamo strumenti di monitoraggio per registrare i processi avviati dal programma, le connessioni di rete stabilite e le modifiche apportate al sistema o ai file.

- Effettuiamo l'analisi del traffico di rete generato dal programma utilizzando strumenti come **Wireshark** per identificare eventuali comunicazioni con indirizzi IP o domini sospetti o noti per attività malevole.

- Verifichiamo se il programma tenta di eseguire azioni sospette o dannose, come l'accesso a risorse riservate, l'iniezione di codice in altri processi o la modifica dei file di sistema.

Sulla base dei risultati ottenuti da queste analisi, forniremo al dipendente un rapporto dettagliato che dimostra l'integrità e la sicurezza del file "IEXPLORE.EXE". Spiegheremo che abbiamo effettuato un'analisi approfondita utilizzando strumenti affidabili e monitoraggio accurato, non rilevando alcuna evidenza di attività maligna o comportamenti anomali nel file.

È importante sottolineare che la sicurezza informatica richiede un'analisi rigorosa e completa, quindi si consiglia sempre di coinvolgere esperti del SOC per approfondire l'indagine e garantire la sicurezza dei sistemi e dei dati aziendali.