

W24D4 Progetto – Peppoli

Traccia:

Malware Analysis Il Malware da analizzare è nella cartella Build_Week_Unit_3 presente sul desktop della macchina virtuale dedicata.

Analisi statica Con riferimento al file eseguibile Malware_Build_Week_U3, rispondere ai seguenti quesiti utilizzando i tool e le tecniche apprese nelle lezioni teoriche:

- Quanti parametri sono passati alla funzione Main()?
- Quante variabili sono dichiarate all'interno della funzione Main()?
- Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate
- Quali librerie importa il Malware? Per ognuna delle librerie importate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.

Con riferimento al Malware in analisi, spiegare:

Lo scopo della funzione chiamata alla locazione di memoria 00401021

Come vengono passati i parametri alla funzione alla locazione 00401021;

Che oggetto rappresenta il parametro alla locazione 00401017

Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.

Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C.

Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?

Analisi dinamica Preparete l'ambiente ed i tool per l'esecuzione del Malware (suggerimento: avviate principalmente Process Monitor ed assicurate di eliminare ogni filtro cliccando sul tasto «reset» quando richiesto in fase di avvio). Eseguite il Malware, facendo doppio click sull'icona dell'eseguibile

-Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware?

Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda Analizzate ora i risultati di Process Monitor (consiglio: utilizzate il filtro come in figura sotto per estrarre solo le modifiche apportate al sistema da parte del Malware).

Fate click su «ADD» poi su «Apply» come abbiamo visto nella lezione teorica.

Filtrate includendo solamente l'attività sul registro di Windows.

- Quale chiave di registro viene creata?

- Quale valore viene associato alla chiave di registro creata?

Passate ora alla visualizzazione dell'attività sul file system.

- Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?

Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

Analisi Statica

Per eseguire analisi statica del malware “Malware_Build_Week_U3” utilizzerò uno strumento chiamato **IDA**, uno degli strumenti di disassemblaggio più avanzati e utilizzati. IDA consente di convertire il codice binario in un formato leggibile.

Le caratteristiche di **IDA** sono le seguenti:

Disassemblaggio: consente il disassemblaggio di binari, trasformando il codice macchina in un formato leggibile (assembly). Questa è una delle funzionalità fondamentali per comprendere come funziona un programma a basso livello.

Supporto per varie architetture: Generalmente, le architetture più comuni come x86 e ARM sono supportate.

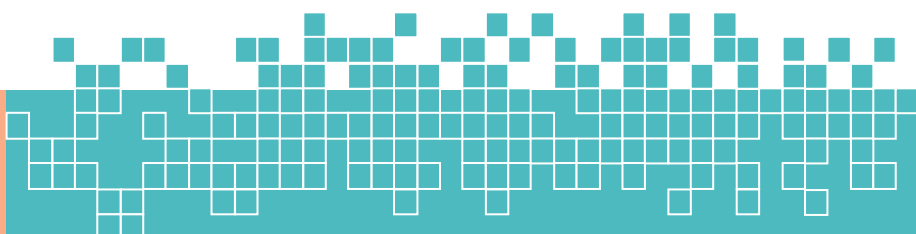
Navigazione interattiva: L'interfaccia interattiva permette agli utenti di navigare facilmente attraverso il codice disassemblato. Puoi seguire le chiamate di funzione, analizzare le referenze di variabili e spostarti rapidamente tra le varie sezioni del binario.

Grafici di flusso: IDA include una modalità di visualizzazione grafica che mostra il flusso del programma in forma di diagramma a blocchi. Questo è utile per comprendere la logica del programma e le relazioni tra le varie parti del codice.

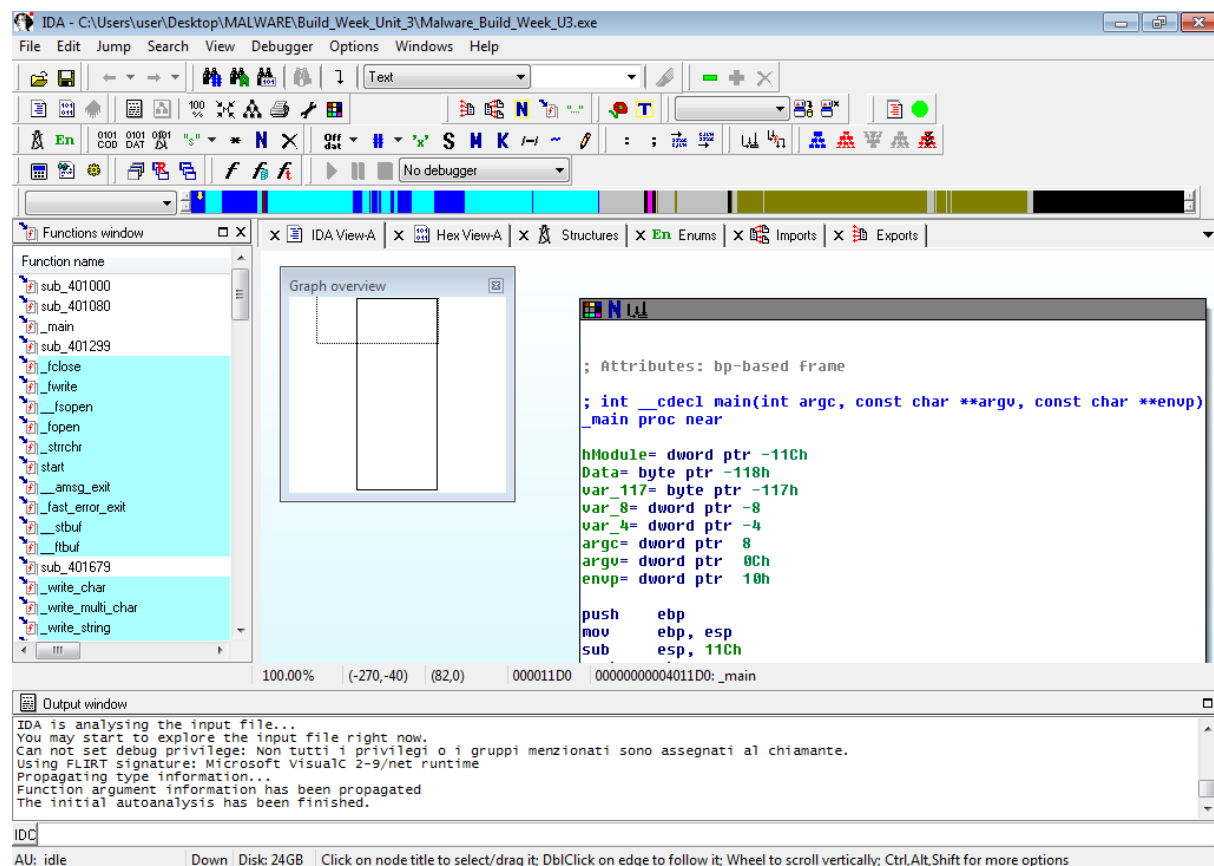
Riconoscimento automatico delle funzioni: L'analizzatore di IDA tenta automaticamente di identificare le funzioni nel codice binario, facilitando l'analisi.

Commenti e annotazioni: Gli utenti possono aggiungere commenti e annotazioni al codice disassemblato, utile per documentare le scoperte durante l'analisi.

Altri strumenti che avremmo potuto utilizzare per eseguire analisi statica sono: Ghidra, strumento di reverse engineering simile ad IDA ma open source, Strings, BinWalk o anche VirusTotal.



Tutto quanto verrà eseguito in un ambiente sicuro, ovvero attraverso una virtual machine, per evitare qualsiasi danno sulla macchina host.



A sinistra notiamo la funzione **Main()**, le variabili che riusciamo subito ad identificare sono hModule, Data, var_117, var_8, var_4 e i parametri argc, argv ed envp.

Per determinare quali sezioni sono presenti all'interno di un file eseguibile durante un'analisi statica, utilizzerò **CFF explorer**. Fornisce una visualizzazione dettagliata delle sezioni del file, delle intestazioni e di altri dati PE, come le tabelle di importazione/esportazione, i certificati di sicurezza, ecc.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word
.text	00005646	00001000	00006000	00001000	00000000	00000000	0000	0000
.rdata	000009AE	00007000	00001000	00007000	00000000	00000000	0000	0000
.data	00003EA8	00008000	00003000	00008000	00000000	00000000	0000	0000
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000030	00	00	00	00	00	00	00	00	00	00	00	E0	00	00	00	00
00000040	0E	1F	BA	0E	00	E4	09	CD	21	E8	01	4C	CD	21	54	68
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00
00000080	65	2D	2F	E7	21	4C	41	B4	21	4C	41	B4	21	4C	41	B4
00000090	17	6A	4A	B4	20	4C	41	B4	A2	50	4F	B4	2E	4C	41	B4
000000A0	17	6A	4B	B4	16	4C	41	B4	43	53	52	B4	24	4C	41	B4
000000B0	21	4C	40	B4	17	4C	41	B4	17	6A	54	B4	20	4C	41	B4
000000C0	E6	4A	47	B4	20	4C	41	B4	52	69	63	68	21	4C	41	B4

Nel file eseguibile mostrato tramite **CFF Explorer**, sono presenti le seguenti sezioni:

.text

.rdata

.data

.rsrc

.text

Descrizione: La sezione .text contiene il codice eseguibile del programma. È una delle sezioni più critiche, poiché include le istruzioni che la CPU esegue.

Caratteristiche:

Virtual Size: 0x00000600

Virtual Address: 0x00001000

Raw Size: 0x00000600

Raw Address: 0x00001000

Utilizzo: Questa sezione è normalmente di sola lettura e viene caricata nella memoria del processo per essere eseguita.

.rdata

Descrizione: La sezione .rdata (Read-Only Data) contiene dati di sola lettura utilizzati dal programma, come le tabelle di importazione, tabelle di esportazione, stringhe costanti e altri dati statici.

Caratteristiche:

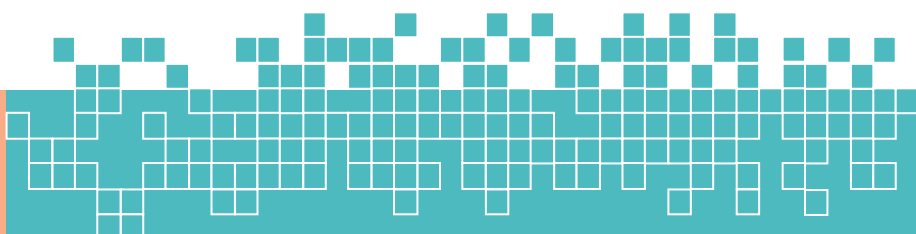
Virtual Size: 0x00000100

Virtual Address: 0x0000A000

Raw Size: 0x00000100

Raw Address: 0x00007000

Utilizzo: Questa sezione è di sola lettura e viene utilizzata dal programma per accedere a dati che non devono essere modificati durante l'esecuzione.



.data

Descrizione: La sezione .data contiene dati globali e statici che possono essere letti e scritti durante l'esecuzione del programma.

Caratteristiche:

Virtual Size: 0x00000300

Virtual Address: 0x0000B000

Raw Size: 0x00000300

Raw Address: 0x00008000

Utilizzo: Questa sezione è utilizzata per memorizzare variabili globali e statiche che il programma può modificare durante la sua esecuzione.

.rsrc

Descrizione: La sezione .rsrc (Resources) contiene risorse del programma, come icone, cursori, stringhe, dialoghi e altri dati di risorsa.

Caratteristiche:

Virtual Size: 0x00000200

Virtual Address: 0x0000C000

Raw Size: 0x00000200

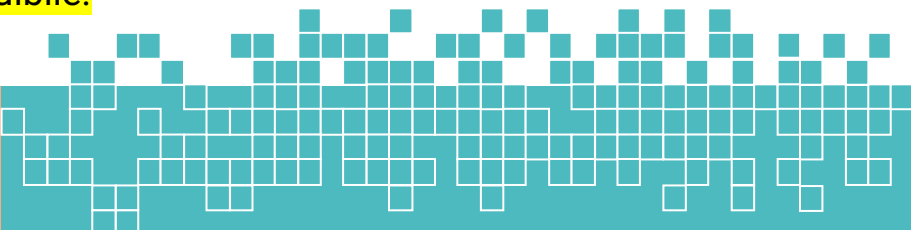
Raw Address: 0x0000B000

Utilizzo: Questa sezione è utilizzata per memorizzare risorse che vengono utilizzate dal programma, ma non sono parte del codice eseguibile o dei dati modificabili durante l'esecuzione.

Considerazioni:

Utilizzare strumenti come **CFF Explorer** ti permette di esaminare in dettaglio le sezioni presenti all'interno di un file eseguibile. Ogni sezione ha un ruolo specifico nel funzionamento del programma:

.text: Contiene il codice eseguibile.



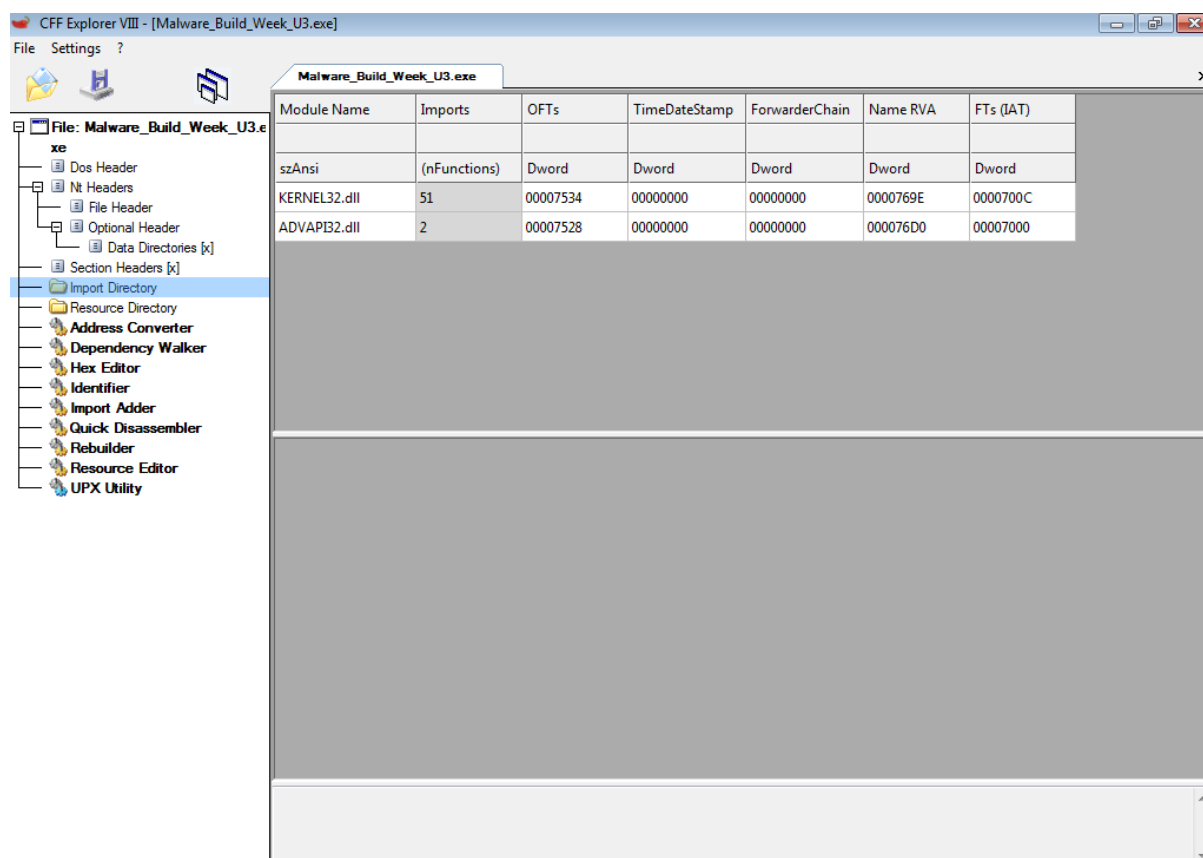
.rdata: Contiene dati di sola lettura utilizzati dal programma.

.data: Contiene dati globali e statici modificabili durante l'esecuzione.

.rsrc: Contiene risorse come icone e stringhe utilizzate dall'applicazione.

Conoscere la struttura delle sezioni di un file eseguibile è fondamentale per l'analisi statica, poiché fornisce una comprensione approfondita di come il programma è organizzato e come funziona internamente.

Per visualizzare le librerie importate dal malware, utilizziamo sempre **CFF Explorer**, ma ci spostiamo nella sezione **"Import Directory"**



Possiamo notare che il malware importa funzioni da due librerie:

KERNEL32.dll

ADVAPI32.dll

Libreria: **KERNEL32.dll**

Numero di funzioni importate: 51

La libreria **KERNEL32.dll** contiene molte funzioni di basso livello per la gestione di memoria, file, processi e thread. Ecco alcune funzioni comunemente importate da questa libreria e le possibili funzionalità che il malware potrebbe implementare:

CreateFile: Apertura e creazione di file o dispositivi. Il malware potrebbe utilizzarla per accedere o modificare file di sistema o creare nuovi file per la scrittura di dati malevoli.

ReadFile e WriteFile: Lettura e scrittura di dati nei file. Queste funzioni possono essere usate per rubare dati o scrivere payload malevoli su disco.

CreateProcess: Creazione di nuovi processi. Potrebbe essere utilizzato dal malware per eseguire altri programmi o script.

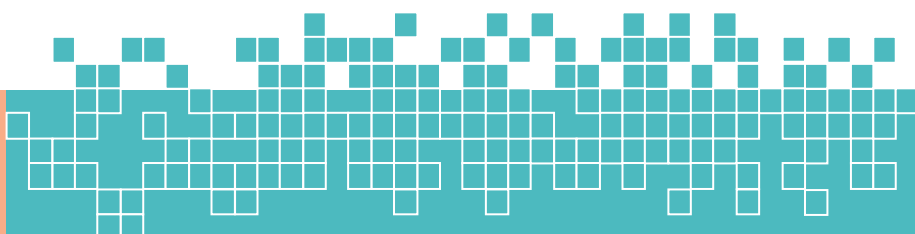
VirtualAlloc e VirtualFree: Gestione della memoria virtuale. Queste funzioni possono essere usate per allocare memoria per il codice eseguibile o dati del malware.

LoadLibrary e GetProcAddress: Caricamento dinamico di altre librerie e ottenimento degli indirizzi delle funzioni. Utilizzato per caricare moduli aggiuntivi o estendere le capacità del malware durante l'esecuzione.

Libreria: **ADVAPI32.dll**

Numero di funzioni importate: 2

La libreria **ADVAPI32.dll** fornisce funzioni avanzate per la gestione di sicurezza e registri di Windows. Ecco alcune funzioni comuni e le possibili funzionalità del malware:



RegOpenKeyEx e RegSetValueEx: Apertura e modifica di chiavi di registro. Il malware potrebbe usare queste funzioni per persistente nel sistema, modificando chiavi di registro critiche per eseguire il malware all'avvio del sistema.

OpenProcessToken e AdjustTokenPrivileges: Manipolazione dei token di sicurezza dei processi. Potrebbe essere utilizzato per elevare i privilegi del malware, consentendogli di eseguire operazioni che richiedono diritti di amministratore.

CryptAcquireContext e CryptReleaseContext: Funzioni di gestione dei contesti di crittografia. Potrebbero essere usate per implementare funzioni di crittografia o decrittografia, ad esempio, per cifrare dati esfiltrati o comunicazioni con un server di comando e controllo.

Considerazioni:

L'analisi delle librerie importate da un eseguibile è un passaggio cruciale nell'analisi statica del malware. Le librerie e le funzioni importate possono dare importanti indizi sulle capacità e i comportamenti potenziali del malware.

KERNEL32.dll suggerisce che il malware ha capacità di interagire con il file system, gestire la memoria e creare processi, indicando funzionalità di accesso ai file, esecuzione di processi secondari e possibili tecniche di evasione.

ADVAPI32.dll indica che il malware potrebbe interagire con il registro di Windows e manipolare token di sicurezza, suggerendo che potrebbe tentare di persistere nel sistema o elevare i propri privilegi per eseguire operazioni più pericolose.

Queste ipotesi basate sull'analisi statica dovrebbero essere confermate con un'analisi dinamica per comprendere appieno il comportamento del malware.

All'indirizzo di memoria **00401021**, troviamo la chiamata di funzione **RegSetValueExA**, ovvero si tratta di una chiamata API di Windows presente nella libreria ADVAPI32.dll. È utilizzata per impostare il valore di una voce di registro. La versione con il suffisso A indica che la funzione utilizza stringhe in formato ANSI.

Considerazioni

Persistenza: I malware possono utilizzare RegSetValueExA per creare o modificare voci di registro che permettono loro di persistere nel sistema, ad esempio, aggiungendosi alle chiavi di avvio automatico.

Configurazione: Può essere usata per configurare parametri che il malware utilizza per controllare il proprio comportamento.

Evasione: Modifiche al registro possono essere usate per disabilitare strumenti di sicurezza o modificare impostazioni di sistema per rendere il malware più difficile da rilevare o rimuovere.

I parametri vengono generalmente passati tramite lo stack. Questo metodo è comune nelle chiamate di funzione in molti linguaggi di basso livello e in molti standard di chiamata.

Push dei Parametri: I parametri vengono inseriti nello stack in ordine inverso, iniziando con cbData e terminando con hKey.

Chiamata alla Funzione: call RegSetValueExA effettua la chiamata alla funzione API di Windows.

Check del Risultato: Dopo la chiamata, il registro EAX contiene il valore di ritorno. Viene confrontato con 0 per verificare il successo della chiamata.

Gestione degli Errori: Se il valore di ritorno indica un errore, il flusso di controllo salta all'etichetta error per eseguire il codice di gestione degli errori.

Alla locazione **00401017** troviamo **offset Subkey**, in questo caso ha il compito di specificare o identificare la chiave di registro, RegCreateKeyExA.

Le istruzioni comprese tra gli indirizzi **00401027** e **00401029** sono:

test eax, eax

Questa istruzione effettua un'operazione AND bit a bit tra il registro EAX e sé stesso. Il risultato di questa operazione viene utilizzato per impostare i flag della CPU (Zero Flag, Sign Flag, Parity Flag, ecc.).

Poiché EAX viene AND con sé stesso, il valore di EAX non cambia. L'obiettivo principale di questa istruzione è impostare i flag della CPU in base al valore corrente di EAX.

Se EAX è zero, lo Zero Flag (ZF) sarà impostato a 1. Se EAX è diverso da zero, lo Zero Flag sarà impostato a 0.

jz short loc_401032

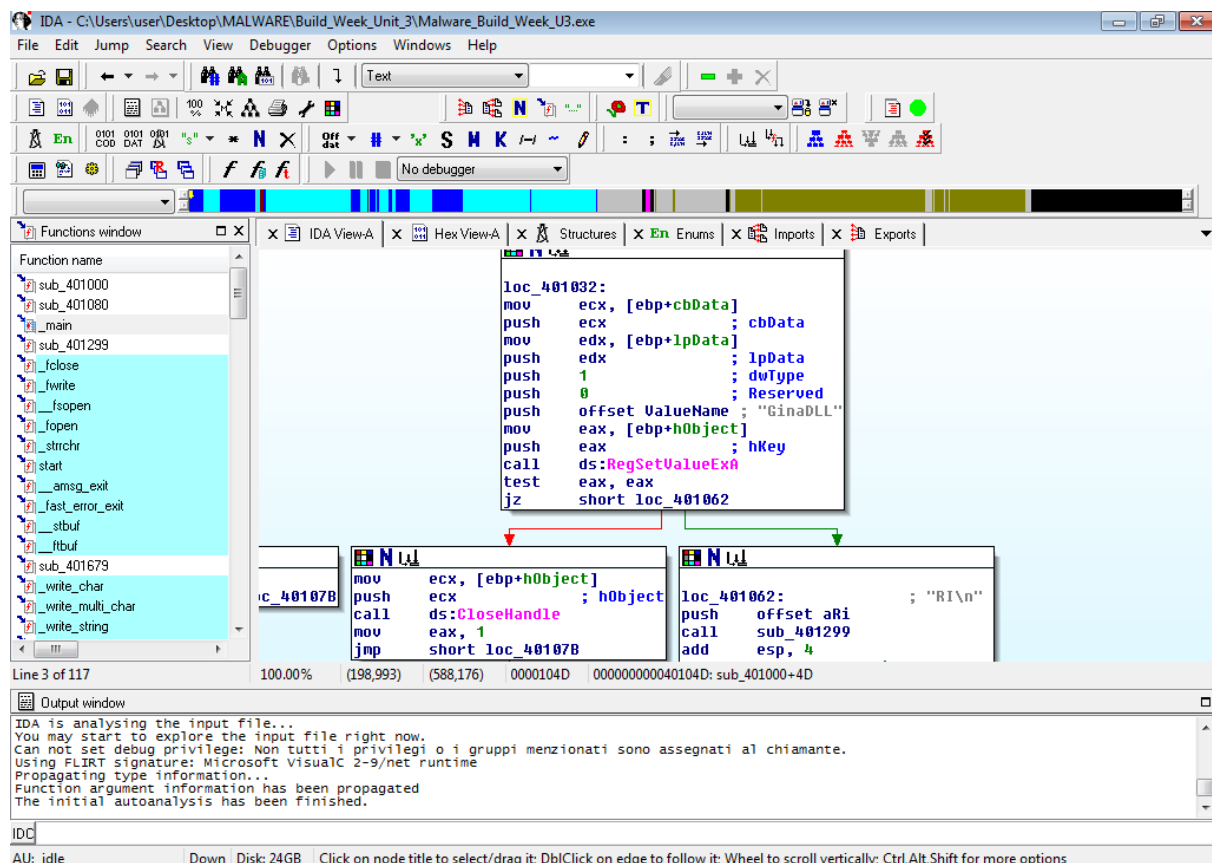
Questa istruzione significa "Jump if Zero" (salta se zero). Verifica il valore dello Zero Flag (ZF).

Se lo Zero Flag è impostato a 1 (che significa che il risultato dell'istruzione test era zero, cioè EAX era zero), il controllo del programma salta all'indirizzo di memoria etichettato come loc_401032.

Se lo Zero Flag è 0 (che significa che EAX non era zero), l'esecuzione continua con l'istruzione successiva.

Codice c:

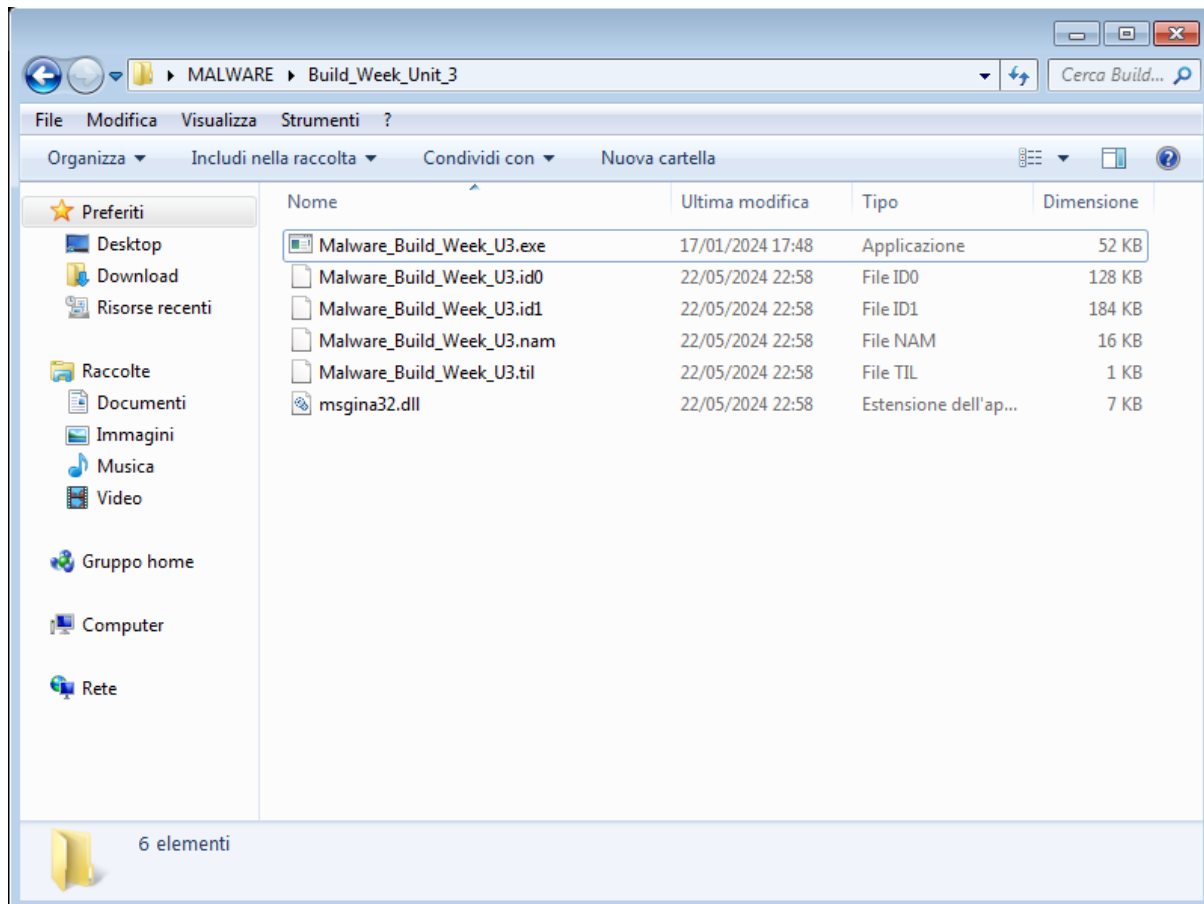
```
int eax;  
if (eax == 0) {  
    goto loc_401032;  
}
```



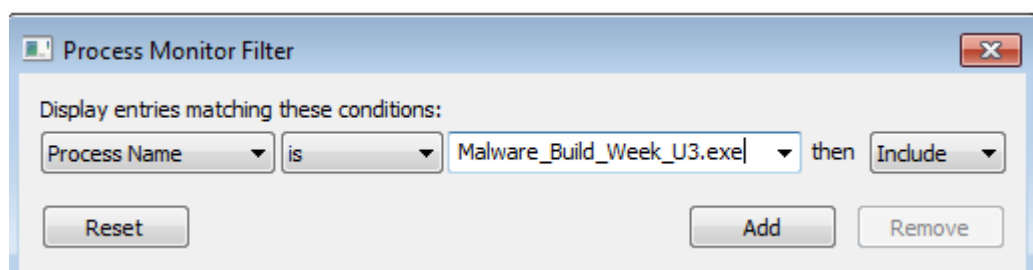
Il valore del parametro "ValueName" alla locazione di memoria 00401047 è "GinaDLL"

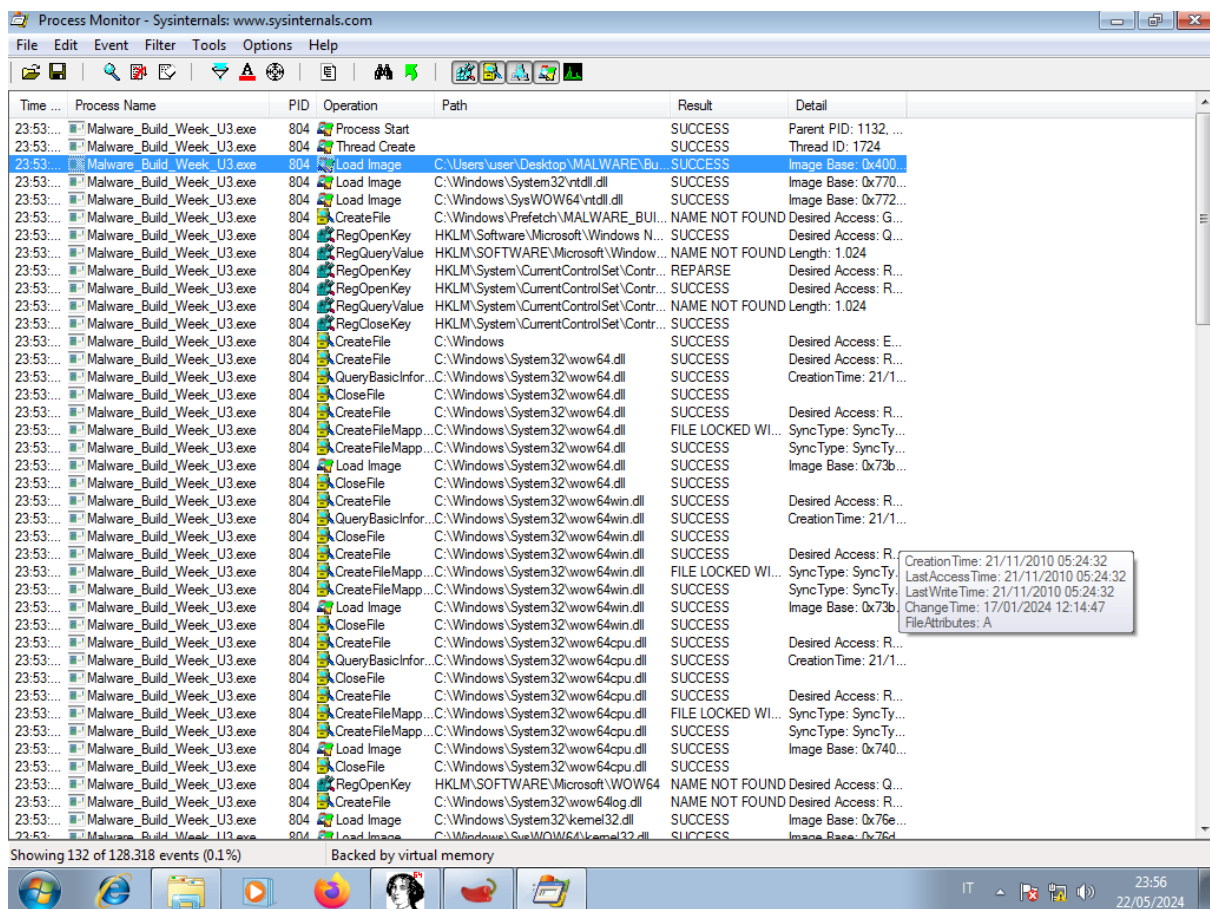
Analisi Dinamica

Una volta avviato il malware notiamo che all'interno della cartella sono stati creati dei file temporanei.

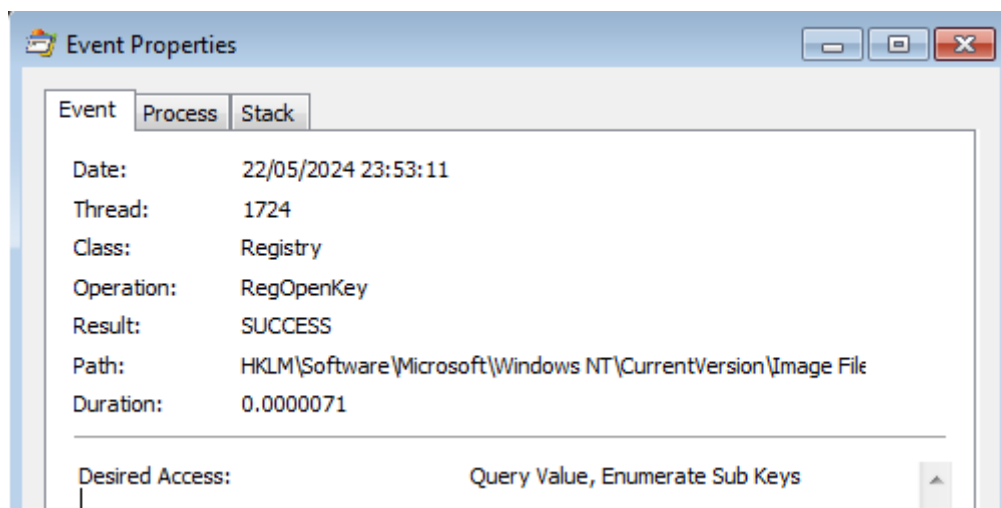


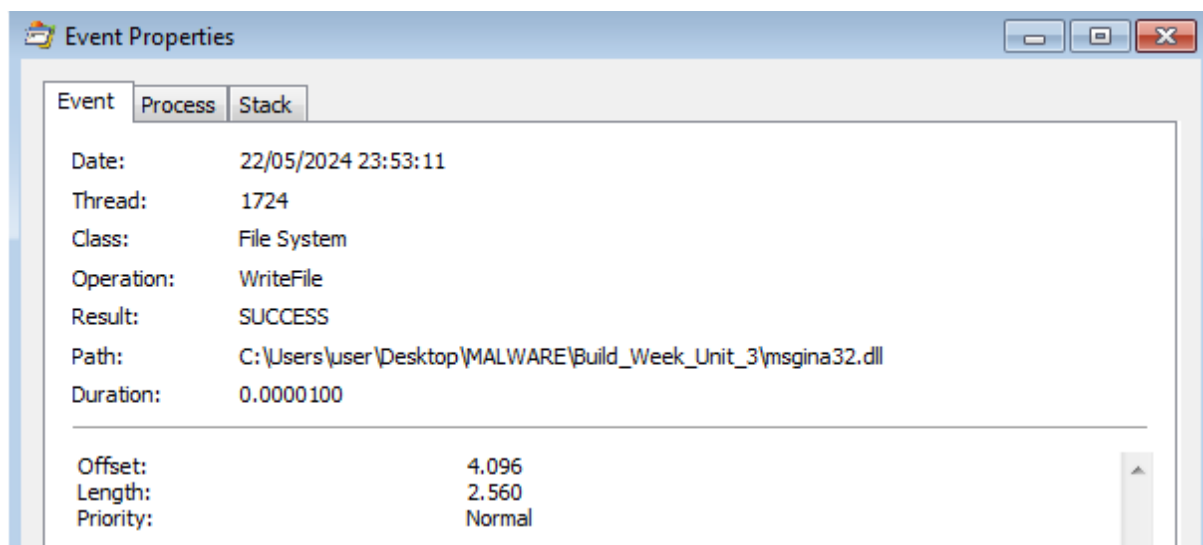
Per effettuare analisi dinamica andremo ad utilizzare procmon, un process monitor. Avviata l'applicazione creiamo un filtro, in modo tale da incentrare la ricerca solo ed esclusivamente al malware.





Notiamo immediatamente che il malware ha eseguito operazioni quali RegCreateKey e RegSetValue, che hanno lo scopo di creare e modificare chiavi di registro.





Il malware si è subito assicurato di impostare come valore della chiave di registro, il valore di msgina32.dll

Notiamo anche i momenti in cui il malware ha creato e modificato dei file, sempre msgina32.dll

Conclusioni:

La modifica della chiave di registro e il coinvolgimento del file msgina32.dll possono essere indicatori di un tentativo di compromissione del sistema. La modifica della chiave di sistema GINADLL potrebbe compromettere il processo di autenticazione degli utenti, rendendo possibile l'accesso non autorizzato al sistema. Inoltre, l'utilizzo del file msgina32.dll suggerisce un coinvolgimento più profondo nel sistema operativo.

Senza dubbio, il malware mira a influenzare le impostazioni del sistema infetto, intervenendo sulla chiave di registro tramite un'altra minaccia denominata msgina32.dll. Questo software dannoso agisce direttamente sulla chiave di sistema, nota come GINADLL. Si tratta di un componente fondamentale nei sistemi operativi Windows antecedenti a Windows Vista, responsabile della gestione dell'autenticazione degli utenti durante il processo di accesso. La manipolazione di questo file comporta il rischio di aprire varchi per accessi non autorizzati o indesiderati, potenzialmente compromettendo la sicurezza del sistema.