

Roberto Rossi
Giacomo di Giacinto
Federico Conti Dragà

Fatjon Selimaj
Lorenzo Sala
Irene Graziano

Andrea Marraro
Daniel Carrubba
Vasyl Popovych



Esercizio
Progetto

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.

Malware Analysis

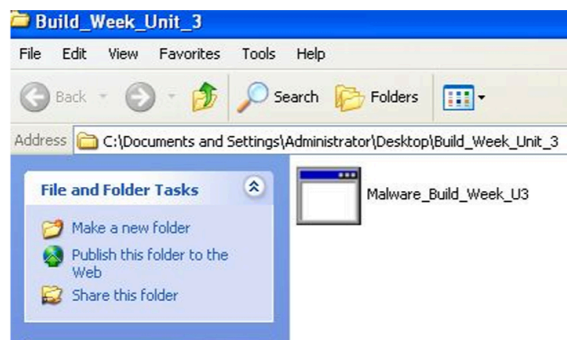
Con riferimento al Malware in analisi, spiegare:

- 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»?

Malware Analysis

Analisi dinamica

Preparate 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



Malware Analysis

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.

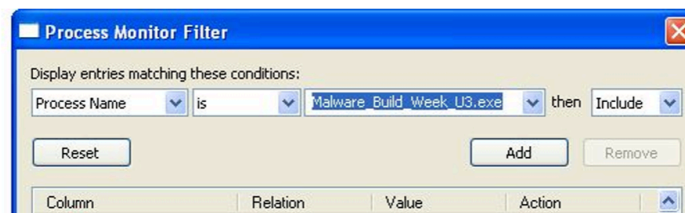
- Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?

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

Malware Analysis

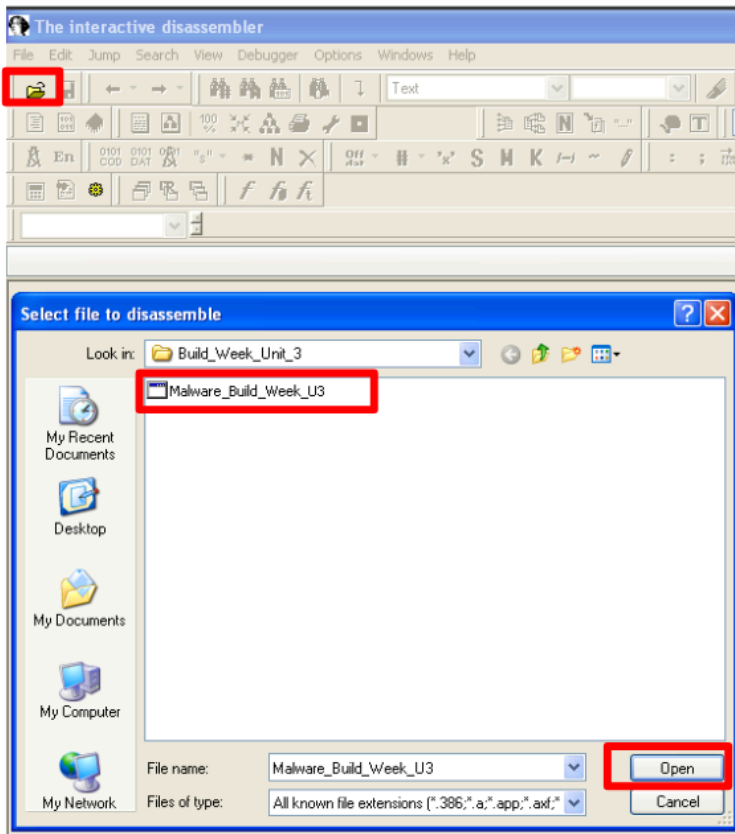
- Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware? Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda

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.



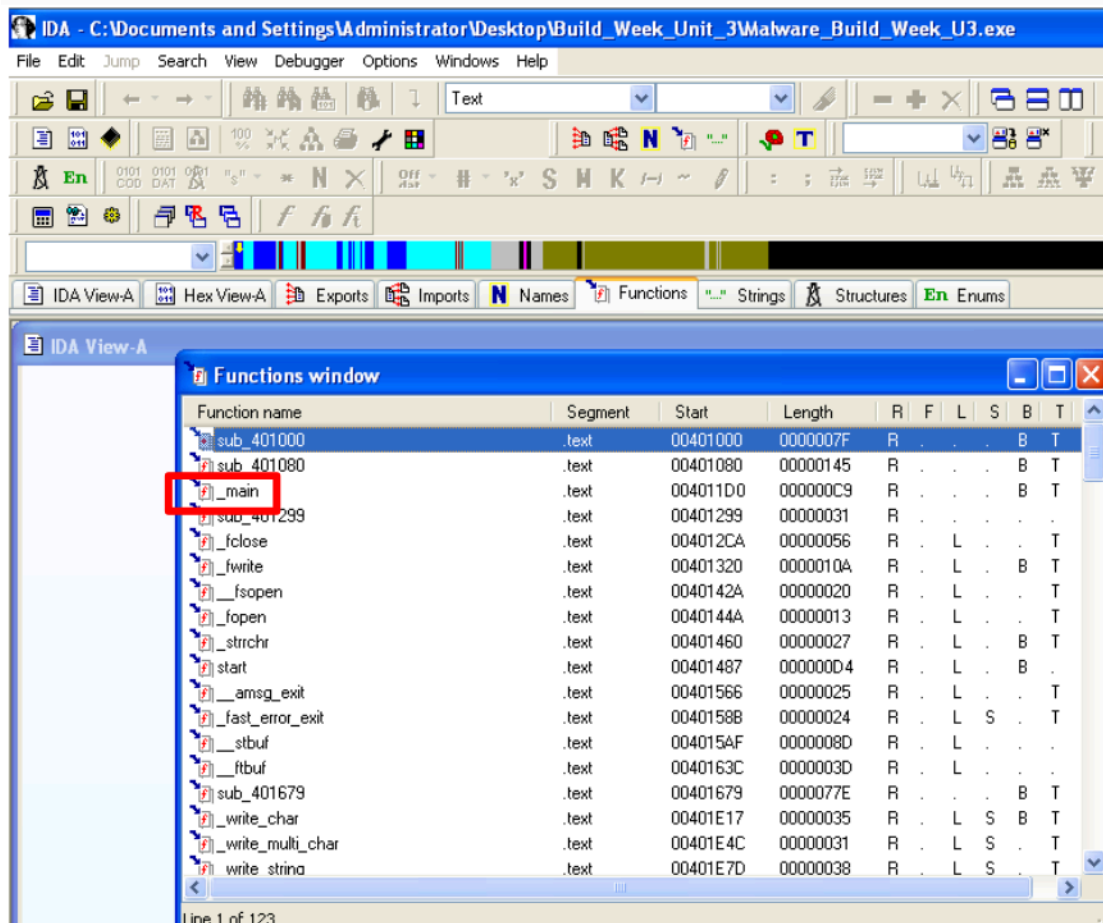
1 Analisi Statica

Per l'analisi statica del Malware_Build_Week_U3, ho utilizzato IDA (Interactive Disassembler). IDA è uno degli strumenti più usati per analizzare malware grazie alla sua capacità di esaminare il codice binario di programmi o eseguibili e di analizzare il codice assembly. L'analisi è stata eseguita in un ambiente sicuro per evitare danni alla macchina principale, utilizzando una VM con un'istantanea e una copia di sicurezza.



2 Parametri e Variabili

La consegna richiedeva di identificare i parametri passati alla funzione Main() e le variabili dichiarate al suo interno. Utilizzando IDA, nella sezione functions, sono stati trovati i parametri e le variabili. Sono stati identificati quattro variabili (hModule, var_8, var_4, data) e tre parametri (argv, argv, envp).



Da qui poi possiamo identificare i parametri e le variabili utilizzando come logica il loro valore , ovvero che : i Parametri ahno valori positivo , mentre le variabili hanno un valore negativo . Di conseguenza ci ritroviamo ad avere :

4 variabili : hModule / var_8 / var_4 / data

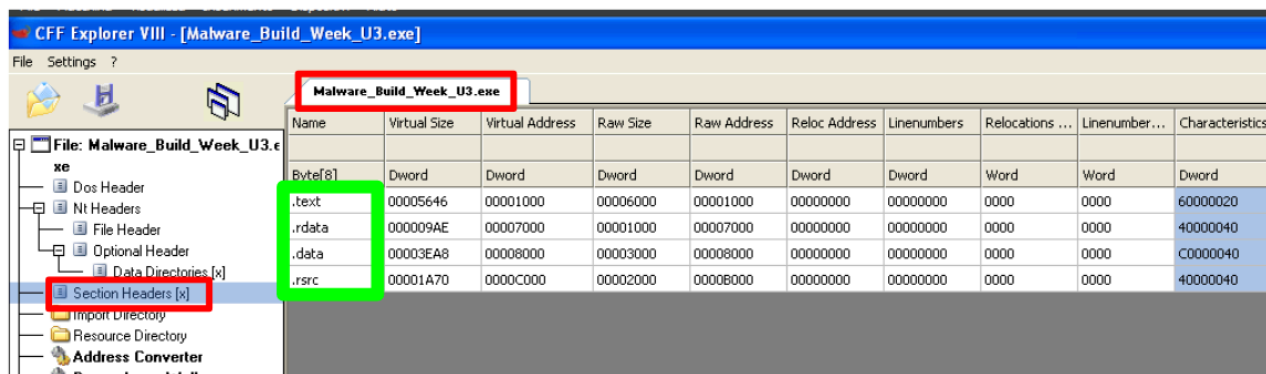
3 parametri : argv / argv / envp .

```
int __cdecl main(int argc, const char **argv, const char *envp)
_main proc near
    hModule= dword ptr -11Ch
    Data= byte ptr -118h
    var_8= dword ptr -8
    var_4= dword ptr -4
    argc= dword ptr 8
    argv= dword ptr 0Ch
    envp= dword ptr 10h

    push    ebp
    mov     ebp, esp
    sub     esp, 11Ch
    push    ebx
    push    esi
    push    edi
```

3 Identificazione Sezioni

Per identificare le sezioni del malware ho utilizzato CFF Explorer, un software specifico per l'analisi dettagliata dei file eseguibili Windows. Le sezioni identificate sono: .text, .rdata, .data, .rsrc.



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Bytef81	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00005646	00001000	00006000	00001000	00000000	00000000	0000	0000	60000020
.rdata	000009AE	00007000	00001000	00007000	00000000	00000000	0000	0000	40000040
.data	00003EA8	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000	40000040

.text: Contiene il codice eseguibile del programma.

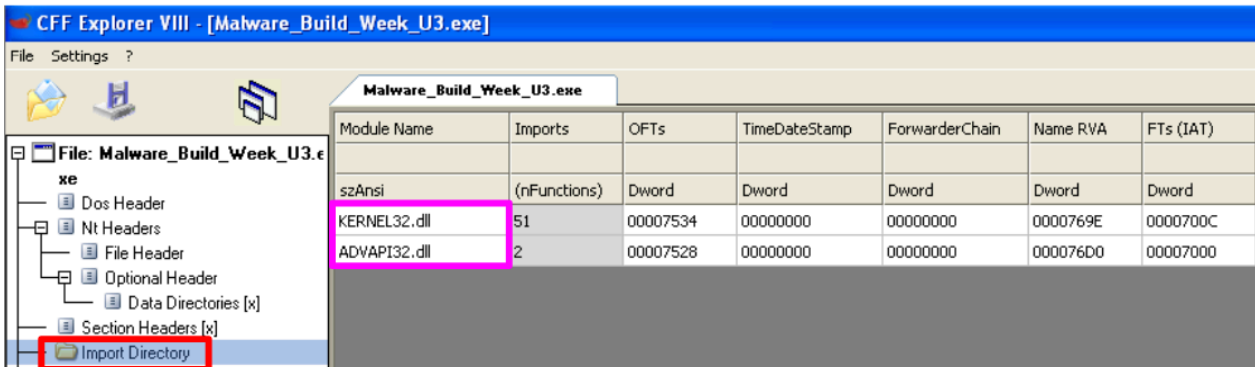
.data: Contiene dati globali e variabili modificabili durante l'esecuzione.

.rdata: Archivia dati di sola lettura come stringhe di testo.

.rsrc: Contiene dati di supporto come immagini e icone.

4 Librerie Malware

Le librerie importate dal malware, individuate con CFF Explorer, sono KERNEL32.dll e ADVAPI32.dll.



KERNEL32.dll: Fornisce funzioni di basso livello necessarie per Windows, come gestione della memoria e dei file.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	001B	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion
000076FE	000076FE	007D	ExitProcess
0000770C	0000770C	019F	HeapFree
00007718	00007718	011A	GetLastError
00007728	00007728	02DF	WriteFile
00007734	00007734	029E	TerminateProcess
00007748	00007748	00F7	GetCurrentProcess
0000775C	0000775C	02AD	UnhandledExceptionFilter
00007778	00007778	00B2	FreeEnvironmentStringsA
00007792	00007792	00B3	FreeEnvironmentStringsW
000077AC	000077AC	02D2	WideCharToMultiByte
000077C2	000077C2	0106	GetEnvironmentStrings
000077DA	000077DA	0108	GetEnvironmentStringsW
000077F4	000077F4	026D	SetHandleCount

OFTs	FTs (IAT)	Hint	Name
00007590	00007068	000077F4	000077F6
Dword	Dword	Word	szAnsi
00007836	00007836	0109	GetEnvironmentVariableA
00007850	00007850	0175	GetVersionExA
00007860	00007860	019D	HeapDestroy
0000786E	0000786E	019B	HeapCreate
0000787C	0000787C	02BF	VirtualFree
0000788A	0000788A	022F	RtlUnwind
00007896	00007896	0199	HeapAlloc
000078A2	000078A2	01A2	HeapReAlloc
000078B0	000078B0	027C	SetStdHandle
000078C0	000078C0	00AA	FlushFileBuffers
000078D4	000078D4	026A	SetFilePointer
000078E6	000078E6	0034	CreateFileA
000078F4	000078F4	00BF	GetCPInfo
00007900	00007900	00B9	GetACP
0000790A	0000790A	0131	GetOEMCP
00007916	00007916	013E	GetProcAddress
00007928	00007928	01C2	LoadLibraryA
00007938	00007938	0261	SetEndOfFile
00007948	00007948	0218	ReadFile
00007954	00007954	01E4	MultiByteToWideChar
0000796A	0000796A	01BF	LCMapStringA
0000797A	0000797A	01C0	LCMapStringW
0000798A	0000798A	0153	GetStringTypeA
0000799C	0000799C	0156	GetStringTypeW

ADVAPI32.dll: Gestisce funzioni avanzate di sicurezza e registro di sistema, come gestione degli account utente e crittografia.

ADVAPI32.dll

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000076AC	000076AC	0186	RegSetValueExA
000076BE	000076BE	015F	RegCreateKeyExA

5 Ipotesi Malware

Secondo quanto riportato dalle librerie, l'ipotesi più valida è che il malware possa essere un dropper, ovvero un malware che contiene al suo interno un altro malware.

Questo perché, esaminando le librerie ADVAPI32.dll e KERNEL32.dll, possiamo notare alcune funzioni sospette (all'interno dei riquadri). Le principali funzioni che ci portano a formulare questa ipotesi sono CreateFileA e WriteFile. Come suggeriscono i loro nomi, queste funzioni rispettivamente creano file e scrivono/modificano file già aperti, quindi potrebbero essere utilizzate per salvare il malware.

A queste si abbinano FindResourceA, che identifica risorse negli eseguibili, come i file .dll. LoadResource viene utilizzata per caricare una risorsa in memoria. Prende un handle a una risorsa restituito da FindResourceA e restituisce un handle a un blocco di memoria contenente la risorsa. Per accedere ai dati, è necessario utilizzare LockResource.

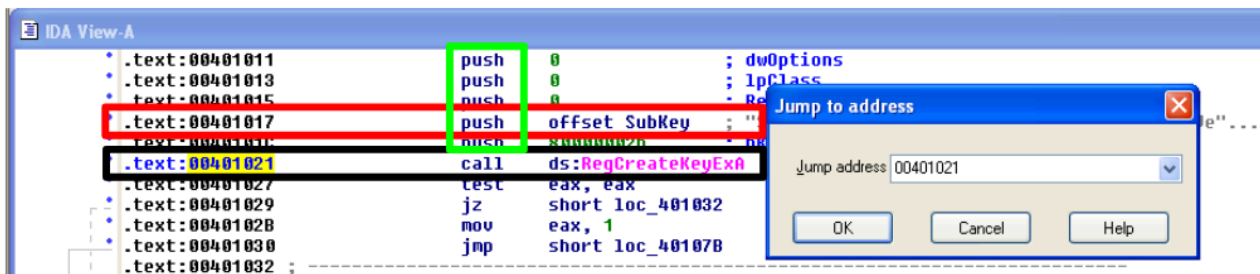
Queste funzioni sono spesso utilizzate quando si manipolano risorse di programmi Windows, come immagini o altri dati dell'eseguibile.

6 Funzioni Malware

Con riferimento al Malware in analisi, spiegare:

1. Lo scopo della funzione chiamata alla locazione di memoria 00401021
2. Come vengono passati i parametri alla funzione alla locazione 00401021;
3. Che oggetto rappresenta il parametro alla locazione 00401017
4. Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.
5. Con riferimento all'ultimo quesito, tradurre Assembly nel corrispondente costrutto C.
6. Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?

1. 2. Come si può notare la locazione di memoria 00401201 ha come funzione RegSetValueExA , che ha come compito creare/aprire una chiave di registro , i parametri gli vengono passati attraverso la funzione PUSH (riquadro verde)



3. alla Funzione 0401017 troviamo offset Subkey (riquadro rosso immagine sopra) il cui compito è quello di specificare / identificare la chiave di registro in questo caso per noi è RegCreateKeyExA

4. Le funzioni sottostanti stanno ad indicare :

test eax, eax : un AND logico del registro EAX e se stesso , con la differenza che in questo caso viene modificato il Zero flag (ZF) del registro EFLAGS impostando il suo valore a 1

jz short loc_401032 : un salto condizionale verso loc_401032 solo se il ZF = 1

In pratica, se la funzione precedente test eax, eax ha come risultato eax = 0 il controllo salterà all'indirizzo specificato (401032) altrimenti, verrà eseguita l'istruzione successiva.

```
• .text:00401027
• .text:00401029
```

```
test    eax, eax
jz      short loc_401032
```

5. il testo in C corrisponde a :

```
int main(){
int a; //eax
int b; // ecx
int c; // [ebp + cbData]
```

```

    if (a == 0){
        b = c;
    }
    else {
        a = 1;
    }
return 0;
}

```

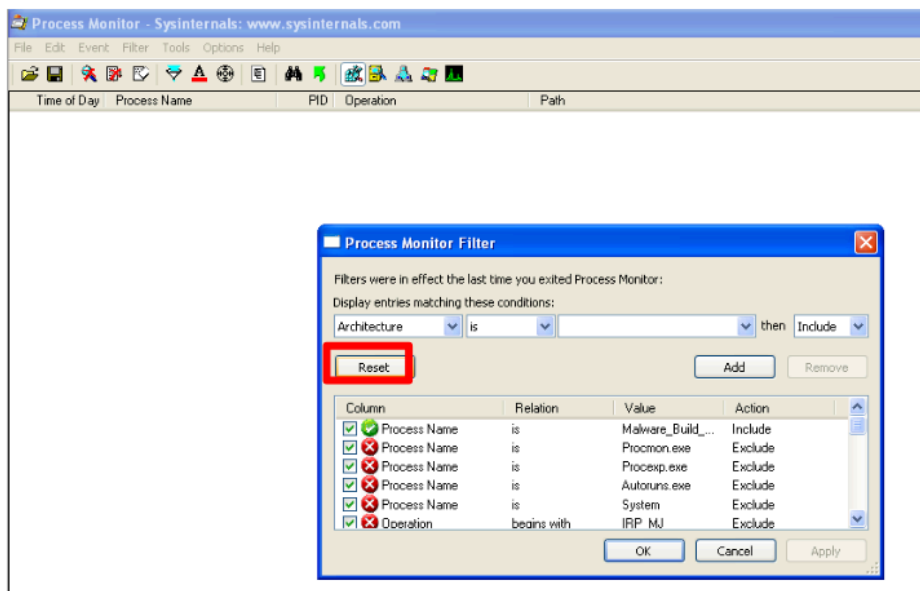
6. Possiamo notare che 00401047 fa riferimento a RegSetValueExa , salendo invece troviamo che il valore di ValueName è appunto " GinaDLL" , ValueName è appunto il parametro che specifica il valore/nome della chiave di registro RegSetValueExa

• .text:0040103E	push	offset ValueName ; "GinaDLL"
• .text:00401043	mov	eax, [ebp+hObject]
• .text:00401046	push	eax ; hKey
• .text:00401047	call	ds:RegSetValueExa

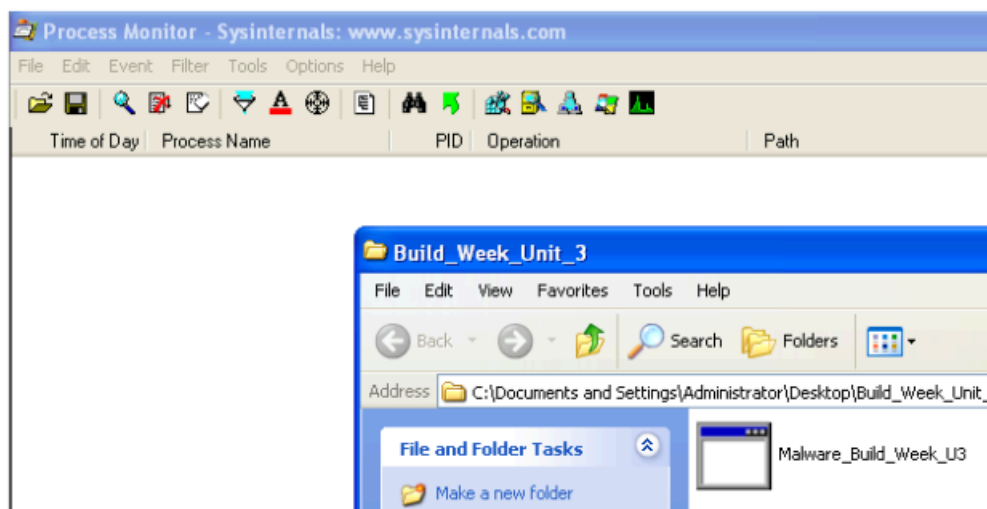
7 Analisi dinamica

Per eseguire l'analisi dinamica utilizzeremo ProcessMonitor, uno strumento per Windows che fornisce un'analisi dettagliata delle attività di sistema in tempo reale, come ad esempio i processi in esecuzione.

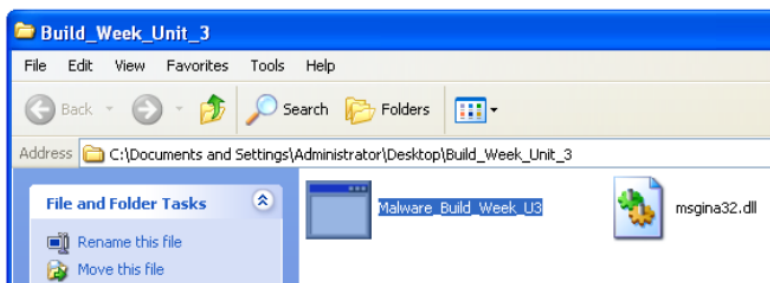
Una volta avviato ProcessMonitor, si aprirà una schermata simile alla seguente, che ci permetterà di impostare i primi filtri (modificabili successivamente).



Attualmente il pc Risultava normale , ora si può procedere all'avvio del malware per vedere come si comporta



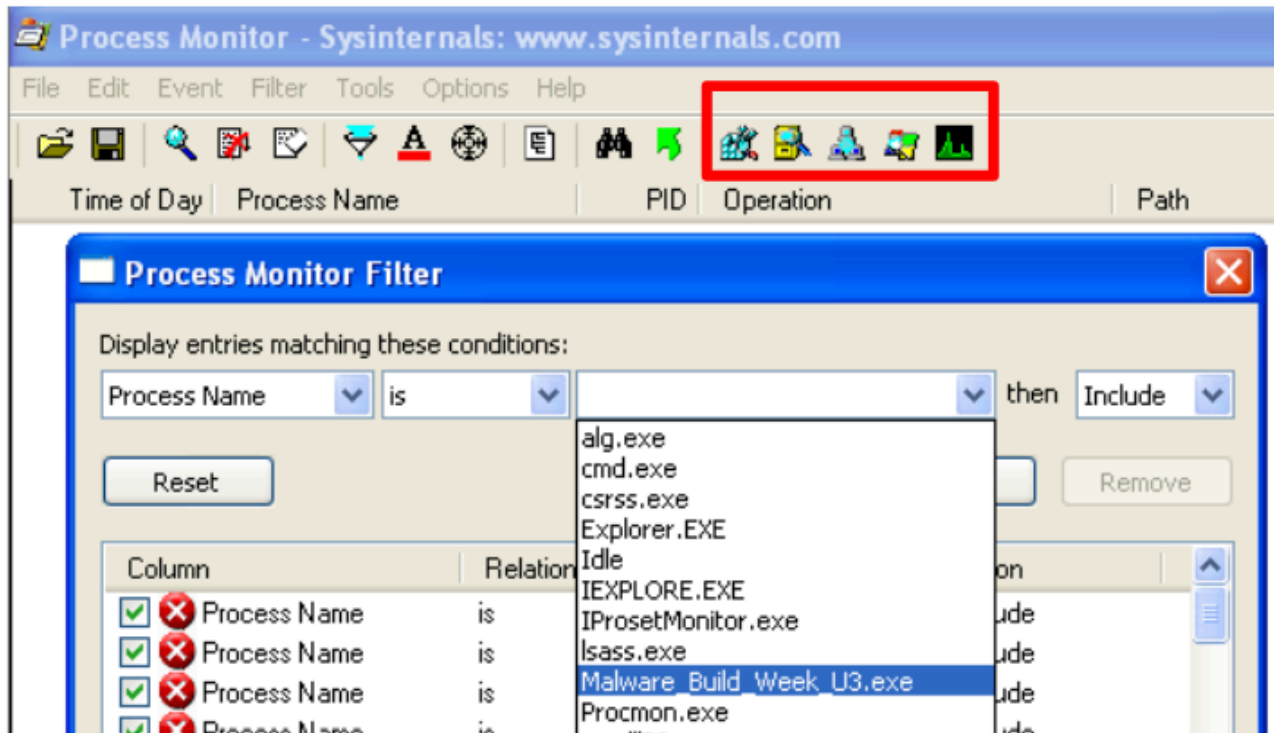
Una volta avviato, la prima cosa che è salata fuori è stata la comparsa di un altro file



8 Malware Analysis

ATTIVITA' DI REGISTRO

A questo punto non mi restava che controllare che processi ha eseguito/tentato di eseguire il malware, per farlo ho usato il pannello in alto per "filtri e selezionando il nome del processo " Malware "



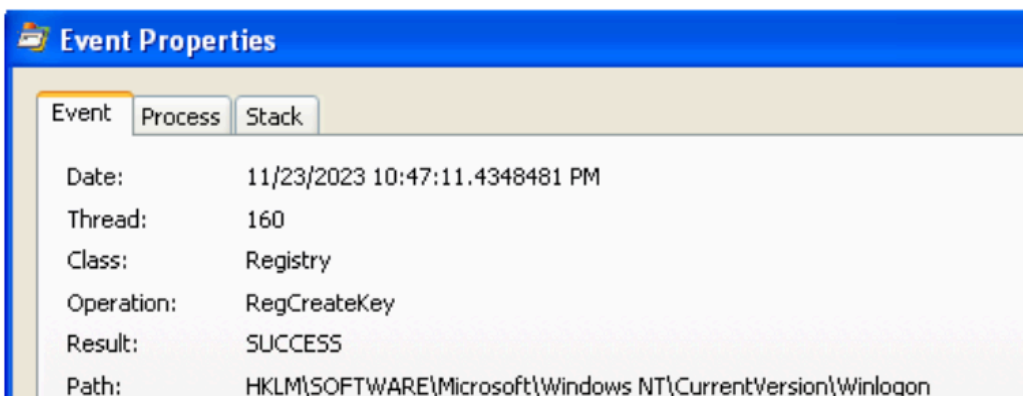
Nel riquadro rosso invece ci sono i filtri per il tipo di ricerca che voglio fare , ovvero : attività di registro, attività file system etc etc..

attraverso questo controllo ho notato delle operazioni insolite, ovvero RegCreateKey e RegSetValue

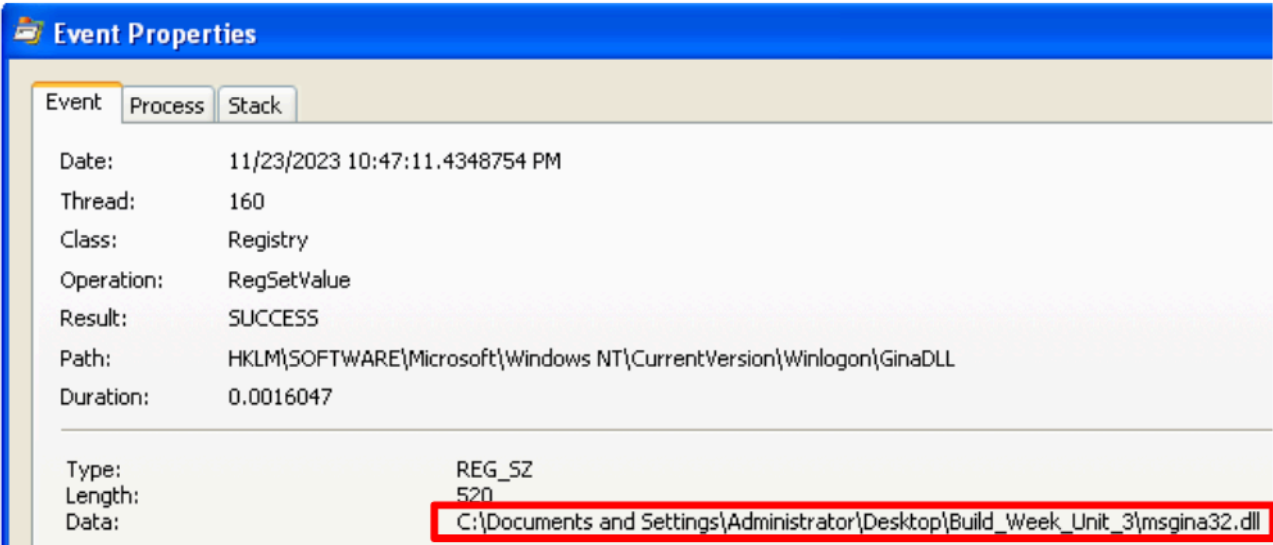
10:47:11.4348...	Malware_Build_Week_U3...	352	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:47:11.4348...	Malware_Build_Week_U3...	352	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL

che hanno come scopo modificare le chiavi di registro, In questo caso modificandone una , nello specifico lo possiamo vedere cliccando sulla voce RegCreateKey , che ci offre un dettaglio più preciso come ad esempio il Path HKLM (HKEY_LOCAL_MACHINE)

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon



Stesso discorso l'ho applicato su RegSetValue notando che ha impostato come nuovo valore alla chiave di registro quello del file che si è creato in precedenza , ovvero msgina32.dll



ATTIVITA' FILE SYSTEM

10:47:11.4278...	Malware_Build_Week_U3...	352	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:47:11.4280...	Malware_Build_Week_U3...	352	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
10:47:11.4281...	Malware_Build_Week_U3...	352	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
10:47:11.4293...	Malware_Build_Week_U3...	352	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:47:11.4302...	Malware_Build_Week_U3...	352	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:47:11.4305...	Malware_Build_Week_U3...	352	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll

Dall'immagine soprastante invece si può notare che il malware ha creato e modificato dei file (sempre msgina32.dll) nonché la nuova chiave di registro creata da esso .

9 Conclusioni

Possiamo quindi affermare con certezza che il malware ha lo scopo di impattare le configurazioni del sistema infettato, modificando la chiave di registro attraverso un altro malware chiamato msgina32.dll.

Questo malware va a modificare la nostra chiave di sistema (GINADLL). Gina.dll è un file utilizzato nei sistemi operativi Windows precedenti a Windows Vista per gestire il processo di autenticazione degli utenti durante il login.

Modificando questo file, si rischia di permettere accessi non autorizzati o indesiderati.