

Progetto Modulo 6

Indice generale

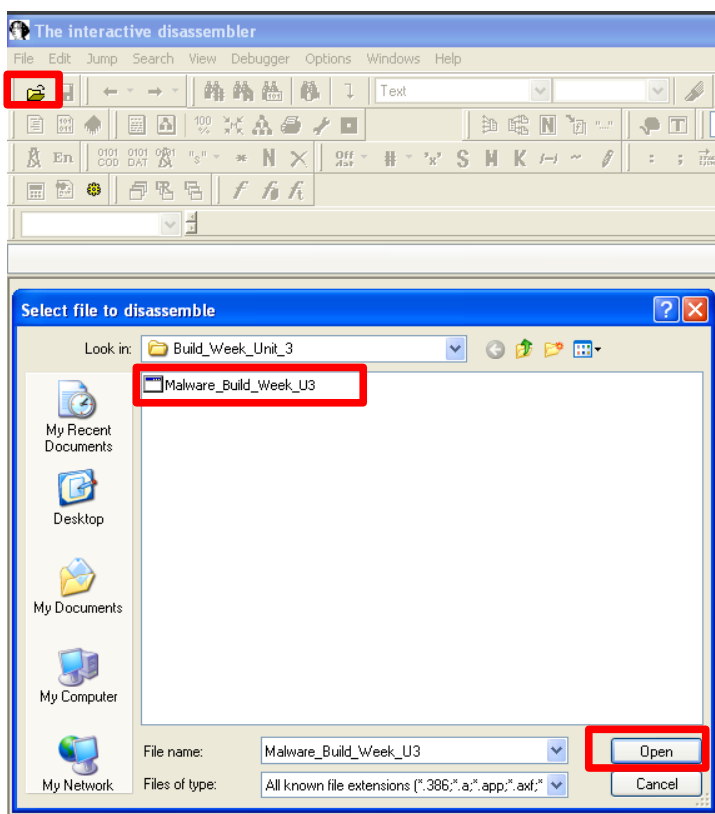
ANALISI STATICA.....	2
PARAMETRI E VARIABILI.....	3
Identificazione sezioni.....	4
LIBRERIE MALWARE.....	6
IPOTESI MALWARE.....	7
FUNZIONI MALWARE.....	8
Con riferimento al Malware in analisi, spiegare:.....	8
1)Lo scopo della funzione chiamata alla locazione di memoria 00401021.....	8
2)Come vengono passati i parametri alla funzione alla locazione 00401021;.....	8
3)Che oggetto rappresenta il parametro alla locazione 00401017.....	8
4)Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.....	8
5)Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C.	8
6)Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?.....	8
ANALISI DINAMICA.....	10
MALWARE ANALYSIS.....	11
CONCLUSIONI SUL MALWARE.....	12

ANALISI STATICA

Per procedere all'analisi statica del Malware_Build_Week_U3 ho sfruttato l'utilizzo di IDA (interactive disassembler)

IDA è uno degli strumenti di disassemblaggio (come suggerito da nome) tra i più utilizzati per l'analisi dei malware , grazie alla sua capacità di esaminare il codice binario di programmi o eseguibili. Un'altra sua funzione molto utile è quella che permette di analizzare il codice assembly.

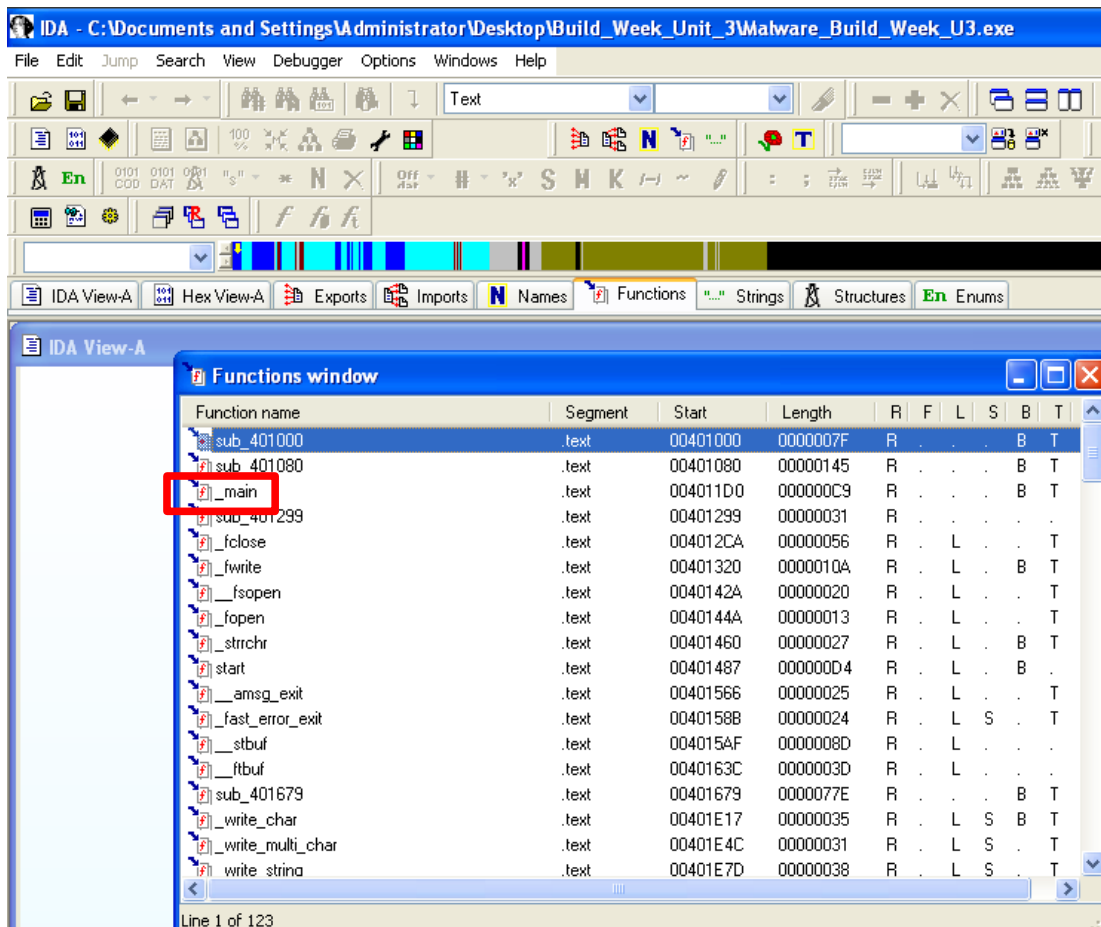
Per procedere all'analisi statica del Malware in questione ho eseguito i seguenti passaggi , ricordando però che il tutto è stato eseguito in un ambiente sicuro per l'analisi , assicurandoci così che il malware non possa arrecare danni alla nostra macchina principale , per quanto riguarda la VM , ho eseguito un'istantanea e una copia per sicurezza.



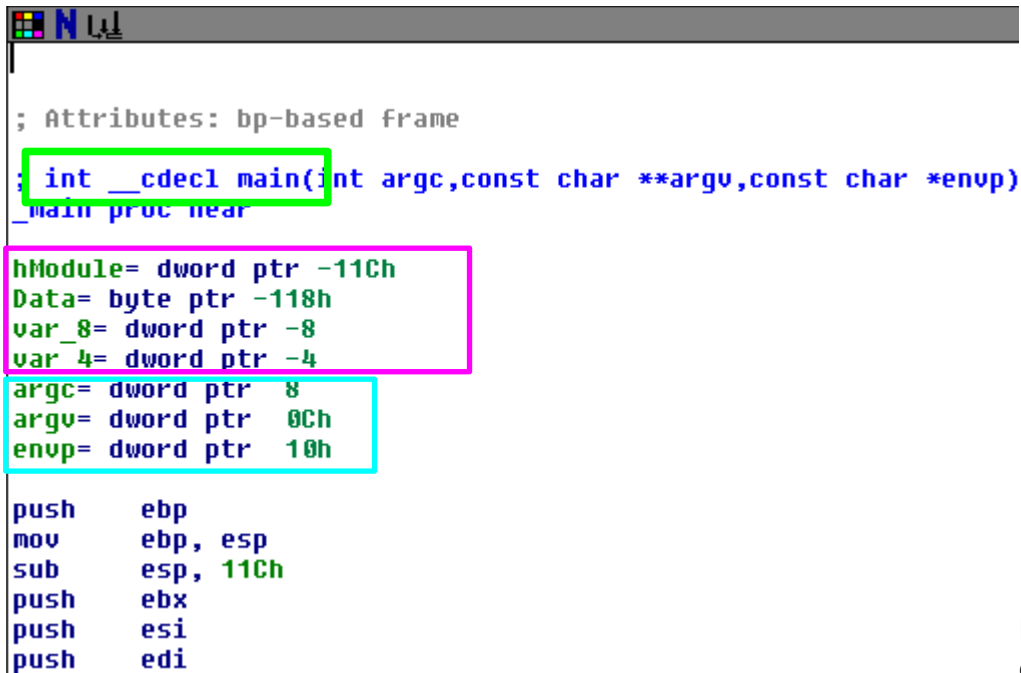
PARAMETRI E VARIABILI

La consegna chiede quali parametri sono passati alla funzione Main() e quali variabili sono dichiarate al suo interno.

Per capirlo ho eseguito i passaggi seguenti , una volta aperto il malware all interno di IDA , ho utilizzato la sezione functions per trovare i parametri e le variabili all interno della funzione Main() come da richiesta della consegna.



Ottenendo questa schermata come risultato :



```
; Attributes: bp-based frame
; 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
```

Nel riquadro verde
otteniamo la

conferma di trovarci nella funzione Main()

Da qui poi possiamo identificare i parametri e le variabili utilizzando come logica il loro valore ,
ovvero che : i **Parametri** ahnno valori positivo , mentre le **variabili** hanno un valore negativo .

Di conseguenza ci ritroviamo ad avere :

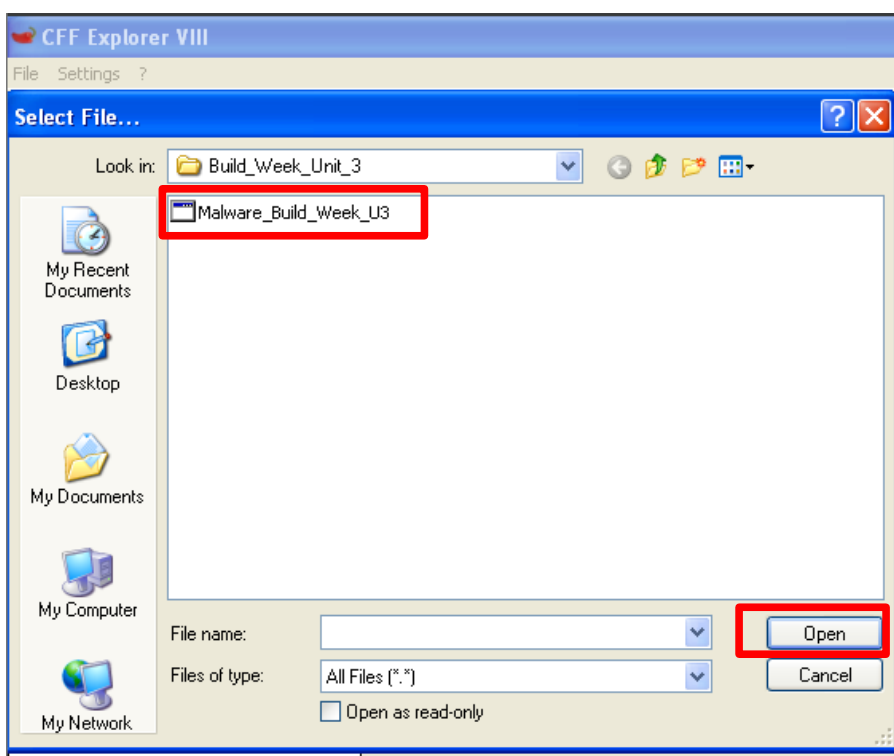
4 variabili : hModule / var_8 / var_4 / data

3 parametri : argc / argv / envp .

Identificazione sezioni

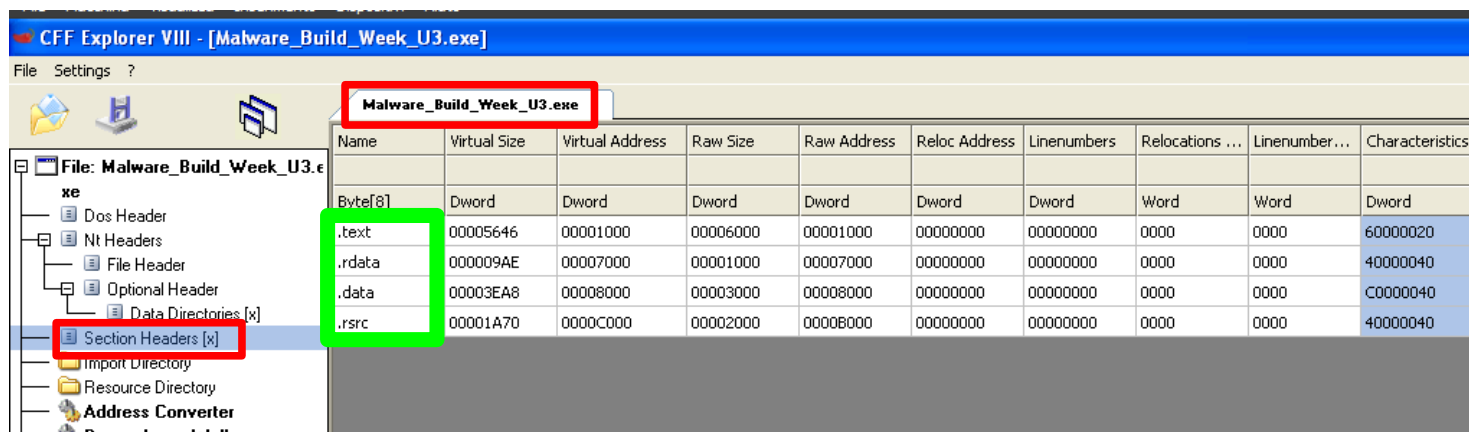
Per l'identificazione delle sezioni presenti nel malware ho fatto utilizzo di un altro tool che si
chiama CFF EXPLORER

CFF EXPLORER è software il cui utilizzo specifico è per l'analisi dettagliata di file eseguibili Windows.
La schermata che ci apparirà all'avvio sarà la seguente :



Dopo di che andando su Section Header possiamo appunto vedere le sezioni presenti all'interno dell'eseguibile del malware in questione.

Le sezioni risultano essere : **.text / .rdata / .data / .rsrc**



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	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

La funzione **.text** è una delle sezioni più importanti , perché è quella che contiene il codice che esegue il programma, quindi senza questa sezione, il programma non potrebbe svolgere alcuna operazione. In sostanza corrisponde alla sequenza di istruzioni che il processore dovrà eseguire

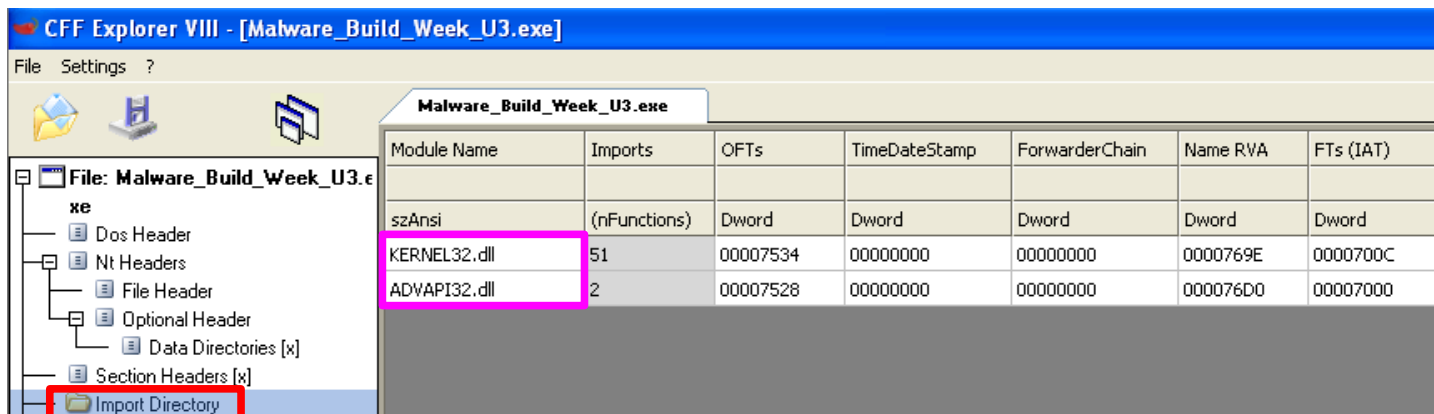
la sezione **.data** invece contiene dati globali e variabili che possono essere modificati durante l'esecuzione del programma. Al suo interno ci sono variabili globali, dati dinamici e informazioni utili al funzionamento del programma.

La sezione **.rdata** è importante perchè archivia i dati di sola lettura che il programma utilizza durante l'esecuzione come ad esempio stringhe di testo e altre informazioni che non devono essere modificate durante l'esecuzione del programma.

La sezione **.rsrc** è importante per il supporto delle risorse del programma, al suo interno contiene dati come immagini, icone, e altri elementi grafici il cui scopo è quello di fornire un'interfaccia utente completa e funzionalità visive.

LIBRERIE MALWARE

Per capire quali fossero le librerie importate dal malware ho ri utilizzato CFF EXPLORER , andando nella sezione del programma chiamata “ Import Directory “



Da cui possiamo notare che emergono due librerie :

KERNEL32.dll

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	028B	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	0198	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

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

KERNEL32.dll: contiene funzioni di basso livello che necessarie

per il funzionamento di Windows , come ad esempio la gestione della memoria, la gestione dei file, la creazione di processi, l'accesso al registro altre operazioni di sistema di base. Nell immagine soprastante ci sono alcune delle funzioni di cui ho appena parlato.

ADVAPI32.dll: contiene funzioni legate alla gestione della sicurezza dei servizi e del registro di sistema. La **ADV** significa "Advanced", specificando che fornisce funzionalità avanzate rispetto alle operazioni di base rispetto al KERNEL32.dll.

ADVAPI32.dll serve anche alla gestione degli account utente, l'accesso al registro, la crittografia, la gestione dei servizi di sistema, etc. etc.

IPOTESI MALWARE

Stando a ciò che riportano le librerie l'ipotesi più valida è quella che il Malware possa essere un Dropper ovvero un malware con all'interno un altro malware

Questo perché dando un'occhiata alle librerie **ADVAPI32.dll** e **KERNEL32.dll** possiamo notare alcune funzioni sospette (all'interno dei riquadri)

Le principali a suscitarmi quest'ipotesi sono CreateFileA e WriteFile , che come suggeriscono loro stesse dal nome , rispettivamente creano file e scrivono/editano file già aperti , quindi salvare il malware

A cui si vanno ad abbinare FindResourceA , che identifica risorse negli eseguibili esempio 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 poi , è necessario utilizzare LockResource.

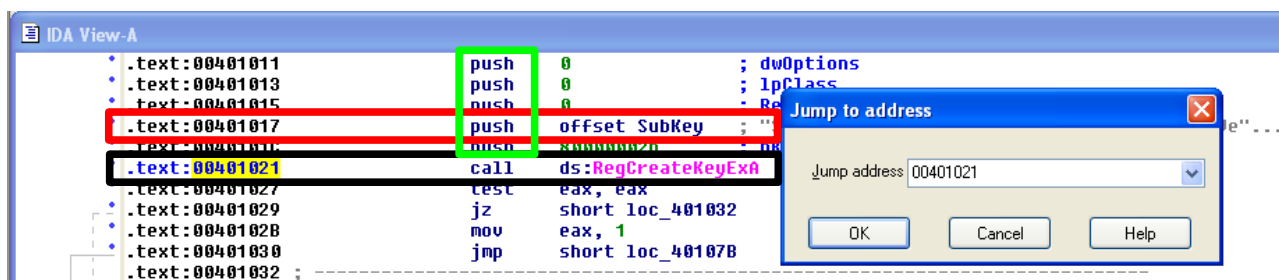
Queste funzioni sono spesso utilizzate quando si manipolano risorse di programmi Windows, come immagini, o altri dati dell'eseguibile.

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 il codice 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 00401021 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 praica, 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      test     eax, eax
.text:00401029      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;
}
```

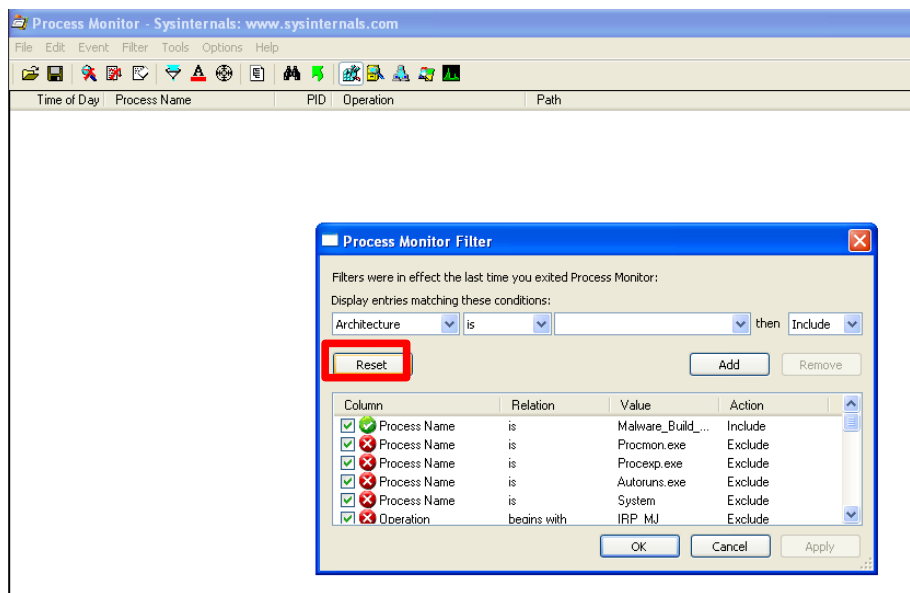
6) Possiamo notare che 00401047 fa riferimento a RegSetValueExa , salendo invece troviamo che il valore di ValueName è appunto “ GinaDLL” , ValueName è appunto il parametro che specifica il valore/nome della chiave di registro RegSetValueExa

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

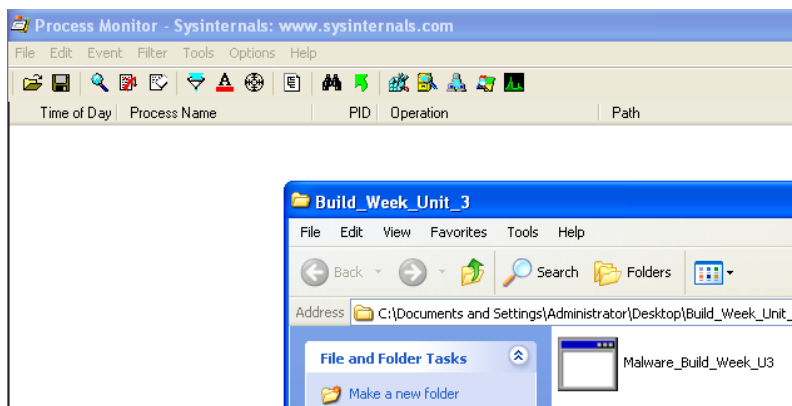
ANALISI DINAMICA

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

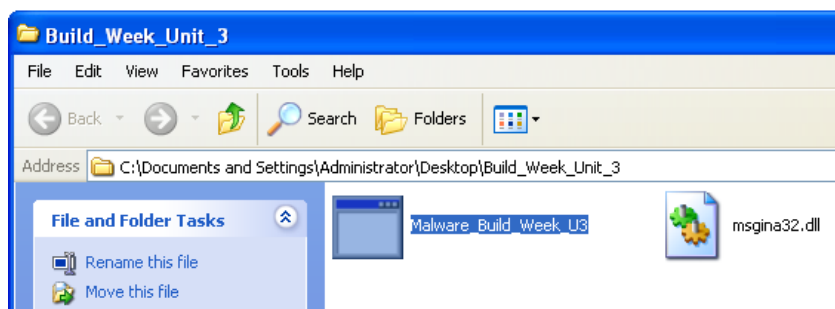
Una volta avviato ProcessMonitor si attiverà una schermata simile alla seguente , ovvero la schermata che ci permette di impostare i primi filtri (successivamente modificabili)



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



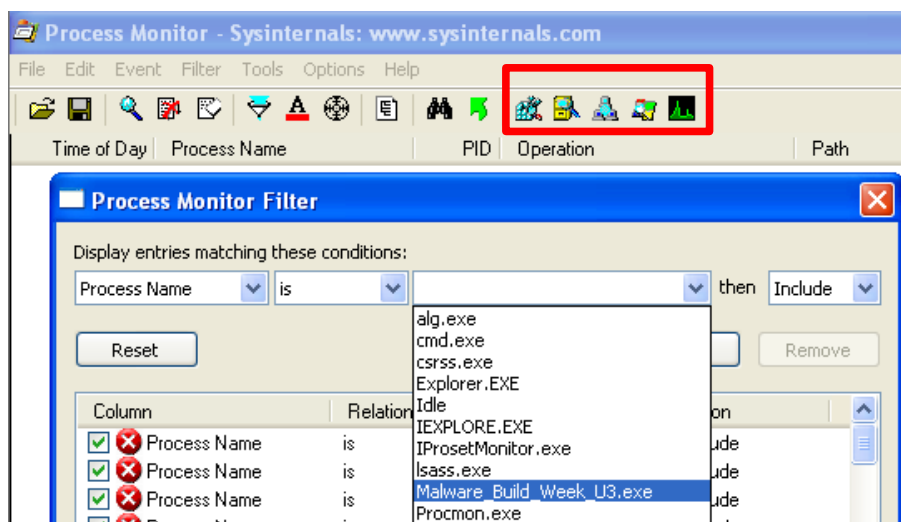
Una volta avviato, la prima cosa che è salata fuori è stata la comparsa di un altro file



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 “



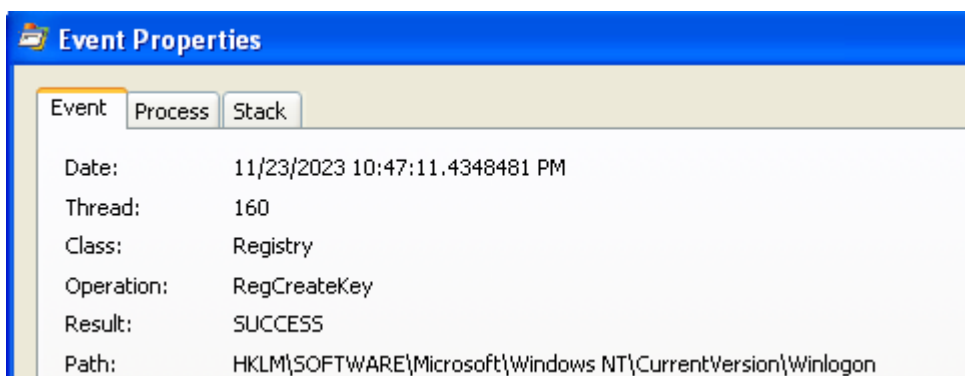
Nel riquadro rosso invece ci sono i filtri per il tipo di ricerca che voglio fare , ovvero : attività di registro, attività file system etc etc..

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

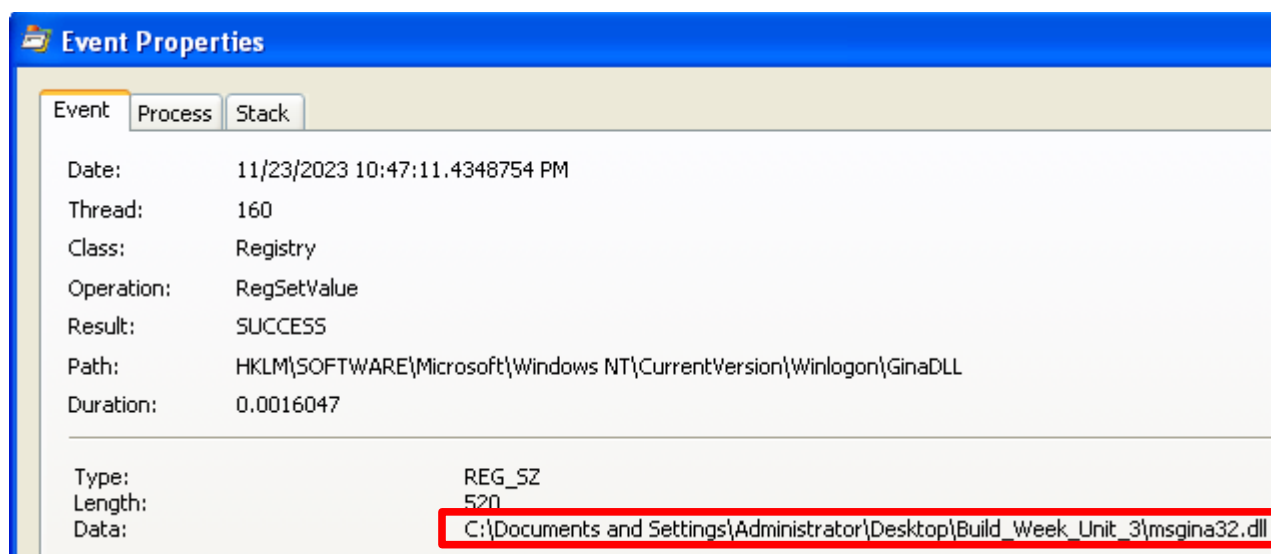
10:47:11.4348...	Malware_Build_Week_U3...	352	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:47:11.4348...	Malware_Build_Week_U3...	352	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL

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** (vedi p.10)



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.

CONCLUSIONI SUL MALWARE

Posso quindi dire con certezza che il malware ha come compito andare ad impattare le configurazioni del sistema che viene infettato, modificando la chiave di registro attraverso un altro malware chiamato msgina32.dll che va a modificare la nostra chiave di sistema (GINADLL)

gina.dll è un file che è stato utilizzato negli OS Windows precedenti a Windows Vista per gestire il processo di autenticazione degli utenti durante il login.

Modificando questo file rischiamo che venga effettuato un accesso non autorizzato o indesiderato.