Agosto 2024 - CSPT0124

BENCHMARK M6-W24

PROGETTO FINALE

di Bortolotti Chiara

TRACCIA

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:
1. Quanti parametri sono passati alla funzione Main()?
2. Quante variabili sono dichiarate all'interno della funzione Main()?
3. Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate
4. Quali librerie importa il Malware?

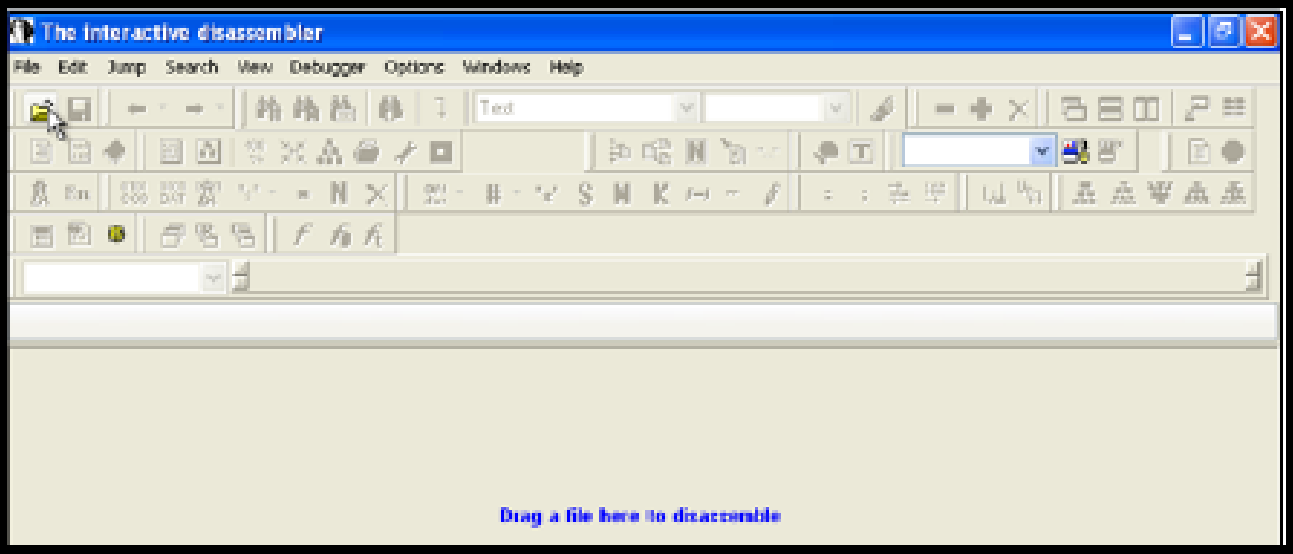
Per ognuna delle librerie importate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.

1. PARAMETRI E VARIABILI NELLA FUNZIONE MAIN())

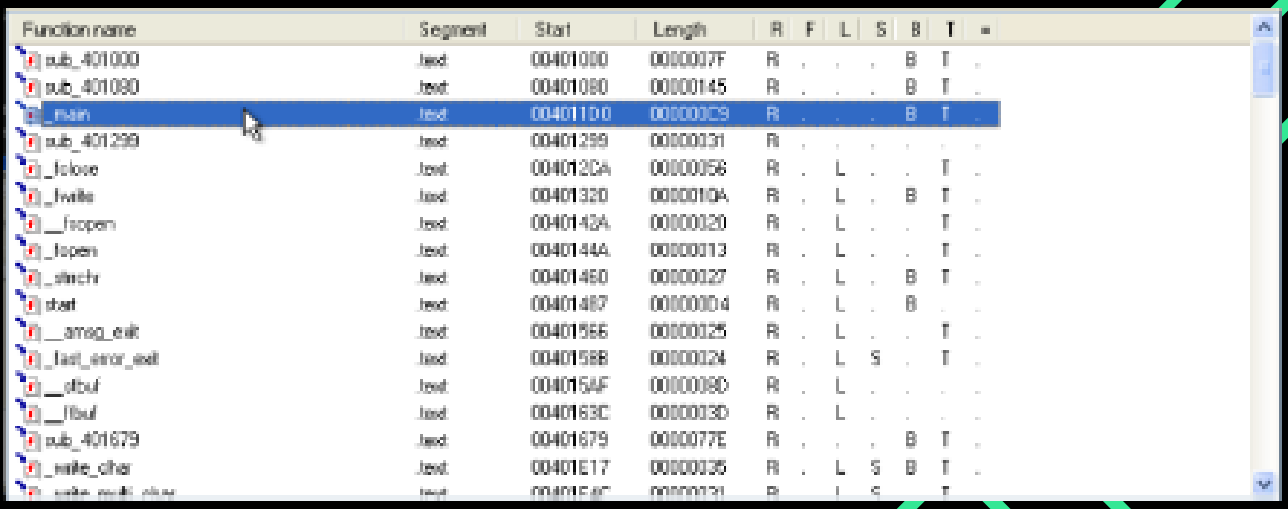
Il Malware da analizzare è nella cartella Build_Week_Unit_3 presente sul desktop della macchina virtuale dedicata.

Tool utilizzati per la raccolta delle informazioni:
- IDA Pro
- CFF Explorer

Per prima cosa andiamo ad aprire il tool IDA Pro presente nella nostra macchina virtuale. Una volta all'interno del programma, procediamo selezionando il Malware da analizzare nell'icona in alto a sinistra

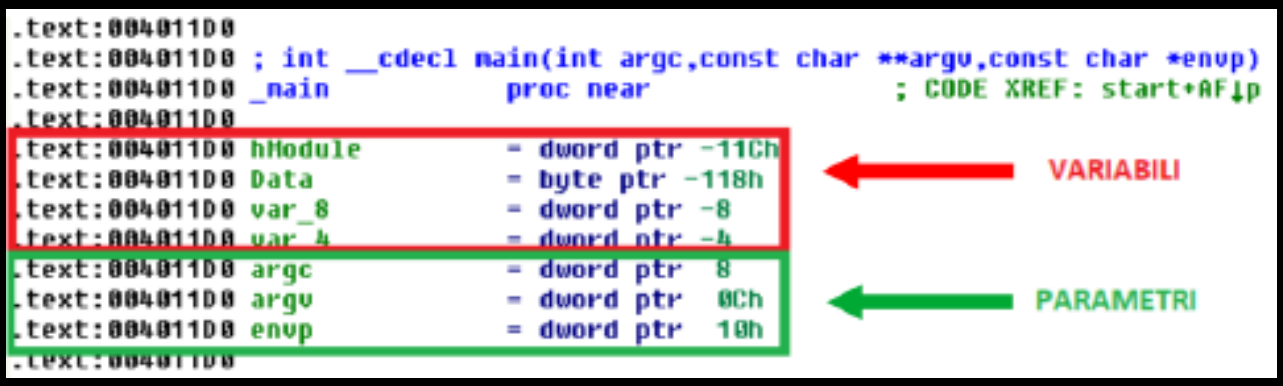


Per analizzare quali parametri e variabili siano passati alla funzione main occorre cliccare nella sezione "Functions" e cercare "main".



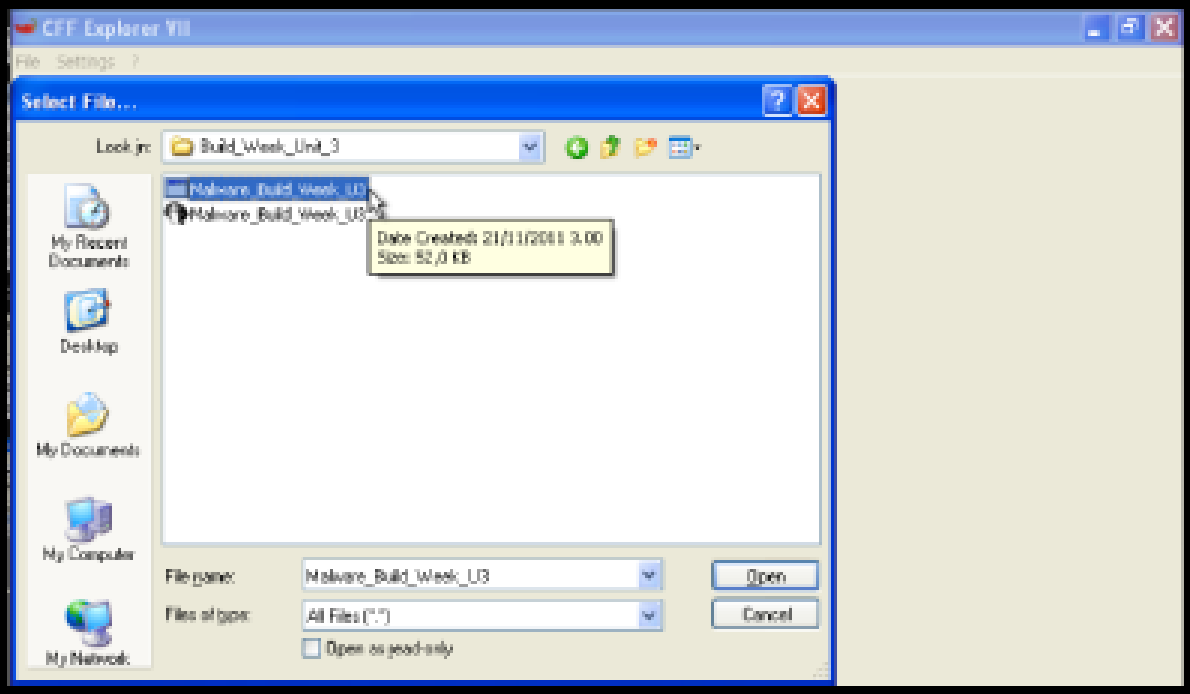
Fatto ciò otterremo un diagramma di flusso il quale ci darà le seguenti informazioni:

- Parametri passati alla funzione main: 3
- Variabili dichiarate all'interno della funzione main: 4

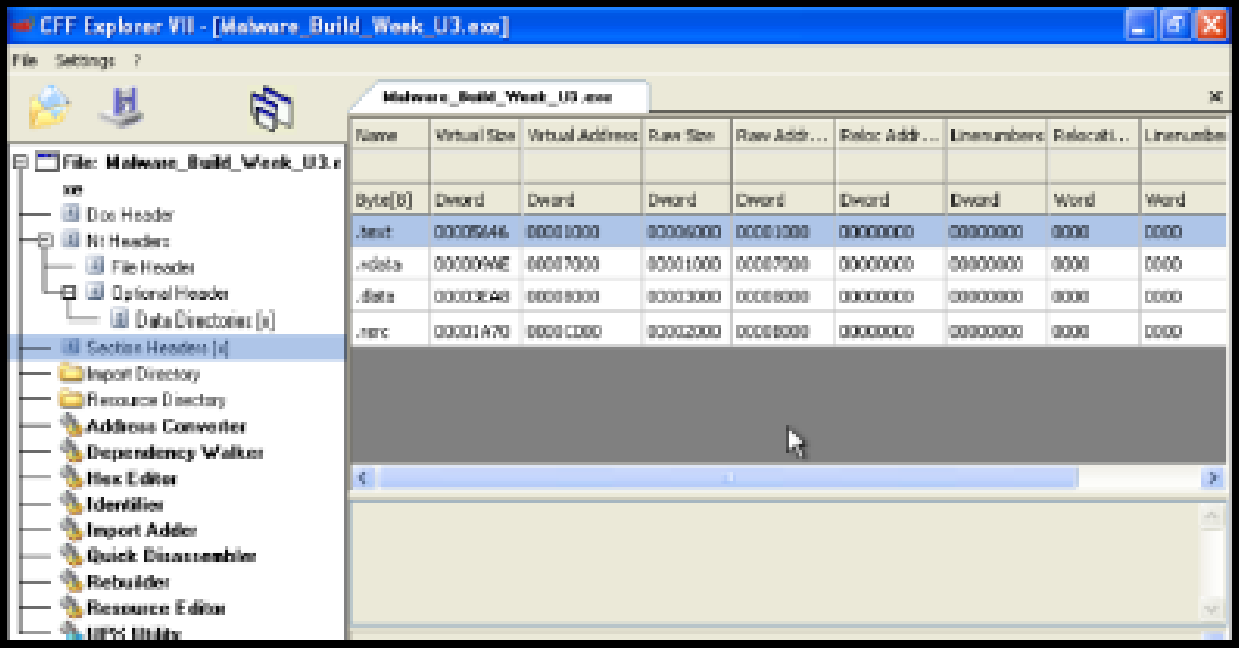


2. SEZIONI PRESENTI ALL'INTERNO DEL FILE ESEGUIBILE

Per sapere quante librerie questo Malware importa potremmo sempre avvalerci di IDA Pro, o alternativamente come in questo caso, per l'analisi statica possiamo utilizzare un altro tool utilissimo: CFF Explorer.
Andiamo ad aprire il file contenente il Malware da analizzare.



Per cercare le sezioni contenute nel file andiamo a cliccare su “Section Headers”.



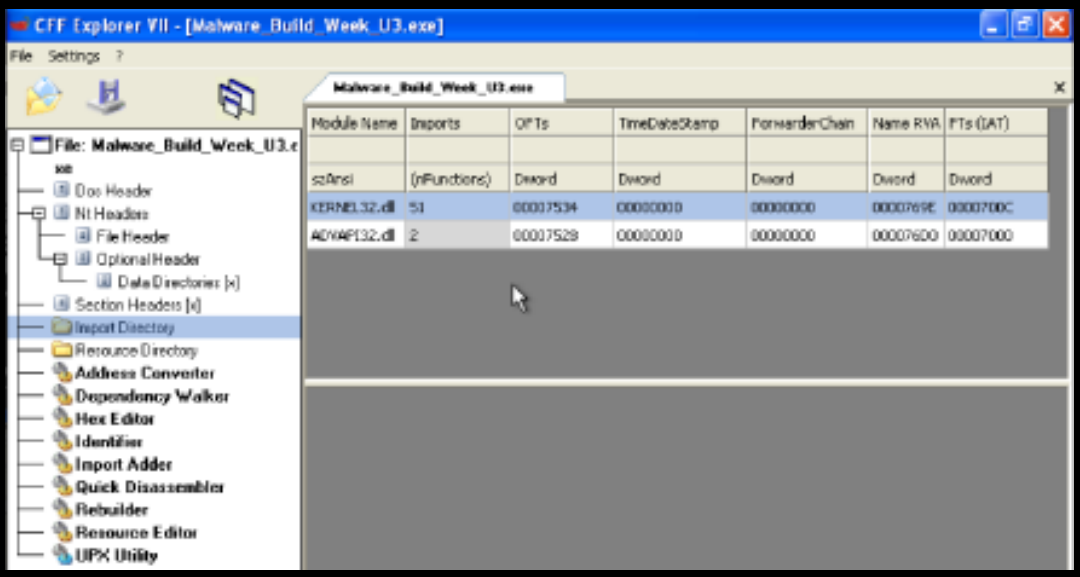
Come possiamo vedere dalle immagini questo file contiene sezioni di tipo:

.text	contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato.
.rdata	include informazioni riguardanti librerie e funzioni importate ed esportate dall’eseguibile.
.data	contiene dati e variabili globali del programma eseguibile, i quali devono essere disponibili in qualsiasi parte del programma.
.rsrc	include le risorse utilizzate dall’eseguibile, come ad esempio icone, immagini e stringhe che non sono parte dell’eseguibile stesso

3. LIBRERIE IMPORTATE DAL MALWARE

Per quanto riguarda la ricerca delle librerie, queste si possono trovare nella sezione “Import Directory”. Come possiamo vedere dall’immagine, questo malware importa due tipi di librerie:

- KERNEL32.dll: libreria che contiene le funzioni principali per interagire con il sistema operativo, come ad esempio manipolare i file o gestione della memoria.
- ADVAPI32.dll: libreria che contiene le funzioni per interagire con i servizi ed i registri del sistema operativo Microsoft



Qui sotto sono riportate le varie funzioni contenute all’interno di ciascuna libreria:

Libreria KERNEL32.dll (51 funzioni):

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0095	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	0018	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion
000076FE	000076FE	007D	ExitProcess
0000770C	0000770C	015F	HeapFree
00007718	00007718	011A	GetLastError

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0095	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	0018	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion
000076FE	000076FE	007D	ExitProcess
0000770C	0000770C	015F	HeapFree
00007718	00007718	011A	GetLastError

OFTs	FTs (IAT)	Hint	Name
000075A4	0000707C	00007850	00007852
Dword	Dword	Word	szAnsi
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

Libreria ADVAPI32.dll (2 funzioni):

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

kernel32.dll:

1-Gestione dei processi e dei thread: Il malware potrebbe utilizzare funzioni come CreateProcess, OpenProcess, CreateThread per avviare nuovi processi o thread e TerminateProcess, ExitProcess per terminarli.
2-Gestione dei file e delle directory: Le funzioni come CreateFile, ReadFile, WriteFile, DeleteFile, CreateDirectory fornite da questa libreria potrebbero
essere sfruttate per manipolare file e directory sul sistema.
3-Gestione del registro di sistema: Funzioni come RegOpenKey, RegSetValue, RegDeleteKey consentono al malware di manipolare il registro di sistema, ad esempio per persistere nel sistema o per nascondere le proprie tracce.
4-Gestione del tempo: Attraverso funzioni come GetSystemTime, GetLocalTime, il malware può acquisire informazioni temporali utili per pianificare azioni o per creare timestamp sui file.

ADVAPI32.dll:

1-Servizi di Windows: Il malware potrebbe utilizzare funzioni come OpenService, StartService, ControlService per interagire con i servizi di Windows, ad esempio per avviare o fermare servizi critici.
2-Gestione degli account utente: Funzioni come LogonUser, CreateProcessAsUser potrebbero essere utilizzate per eseguire azioni privilegiate o per impersonare un utente diverso.
3-Crittografia e sicurezza: ADVAPI32.dll fornisce funzioni per la crittografia, come CryptEncrypt, CryptDecrypt, che il malware potrebbe utilizzare per
nascondere le proprie attività o per cifrare dati sensibili.
4-Auditing e controllo degli eventi di sistema: Funzioni come RegNotifyChangeKeyValue, NotifyChangeEventLog consentono al malware di monitorare le modifiche al registro di sistema o agli eventi del registro di sistema.

Queste sono solo alcune delle funzionalità che un malware potrebbe implementare utilizzando le funzioni fornite dalle librerie kernel32.dll e ADVAPI32.dll.
La conoscenza di queste funzioni consente agli analisti di sicurezza di identificare e mitigare le minacce potenziali.

Conclusioni finali:

A valle di questa analisi statica, non possiamo avere prove concrete riguardo l'identità di questo Malware. Però possiamo ipotizzare che questo file richiamerà funzioni di modifica alle chiavi di registro tramite "RegCreateKeyExA" – "RegSetValueExA" contenute nella libreria "ADVAPI32.dll" per poi andare a creare un file ed eseguirlo con comando "CreateFileA" – "SetEndOfFile" attraverso la libreria KERNEL32.dll

TRACCIA

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 un'altra o altre due righe assembly)
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»?

Nel complesso delle due funzionalità appena viste, spiegate quale funzionalità sta implementando il Malware in questa sezione.

1. SCOPO DELLA FUNZIONE REGCREATEKEY

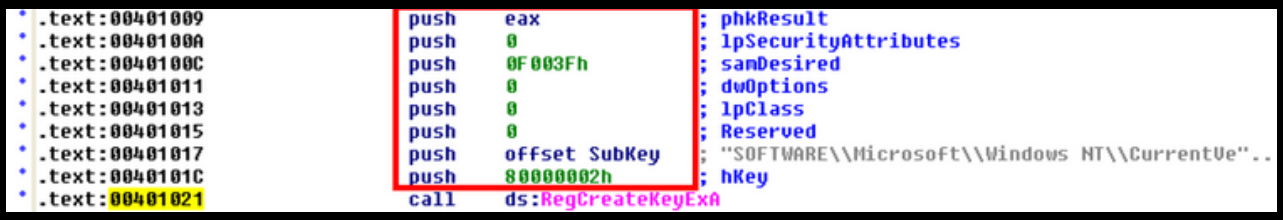
La funzione chiamata all'indirizzo di memoria 00401021 ha lo scopo di invocare la funzione RegCreateKeyExA. Quest'ultima funzione restituisce un valore di successo nel caso in cui la chiave specificata venga creata correttamente. In alternativa, se la creazione della chiave non riesce, la funzione restituisce un valore di errore, indicativo del fallimento dell'operazione.

```
.text:00401017                  push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows
.text:0040101C                  push    80000002h       ; hKey
.text:00401021                  call    ds:RegCreateKeyExA
.text:00401027                  test    eax, eax
.text:00401029                  jz      short loc_401032
.text:0040102B                  mov     eax, 1
.text:00401030                  jmp     short loc_40107B
.text:00401032 ; ---------------------------------------------------------------------------
.text:00401032
.text:00401032 loc_401032:                              ; CODE XREF: sub_401000+29↑j
.text:00401032                  mov     ecx, [ebp+cbData]
.text:00401035                  push    ecx             ; cbData
.text:00401036                  mov     edx, [ebp+lpData]
.text:00401039                  push    edx             ; lpData
.text:0040103A                  push    1               ; dwType
.text:0040103C                  push    0               ; Reserved
.text:0040103E                  push    offset ValueName ; "GinaDLL"
.text:00401043                  mov     eax, [ebp+hObject]
.text:00401046                  push    eax             ; hKey
.text:00401047                  call    ds:RegSetValueExA
.text:0040104D                  test    eax, eax
.text:0040104F                  jz      short loc_401062
.text:00401051                  mov     ecx, [ebp+hObject]
.text:00401054                  push    ecx             ; hObject
```

Nel processo di passaggio dei parametri dalla funzione al punto di memoria 00401021, viene impiegato il metodo noto come "passaggio per valore".
Tale metodo comporta la duplicazione dei valori dei parametri effettivi nello stack di esecuzione, dove vengono utilizzati come copie locali per l'elaborazione all'interno della funzione.

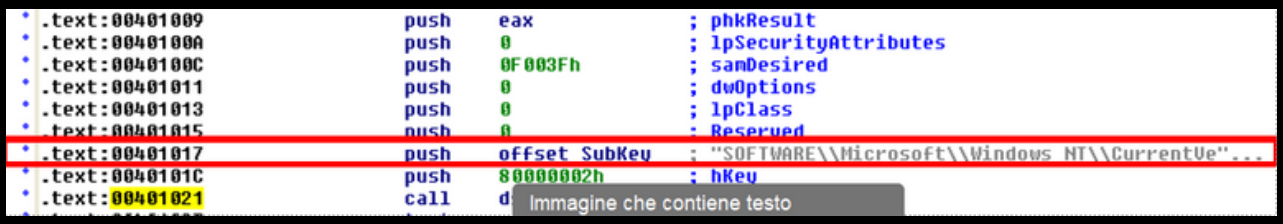
2. COME VENGONO PASSATI I PARAMETRI

Con le istruzioni push i parametri vengono passati nello stack di memoria, che la funzione poi andrà ad utilizzare



3. COSA RAPPRESENTA L'OGGETTO ALL'INDIRIZZO 00401017

L'oggetto alla locazione di memoria 00401017 rappresenta la chiave di registro utilizzata dal malware per la persistenza sul pc vittima. Questa chiave che ha come percorso "Software\\Microsoft\\Windows NT\\CurrentVersion\\WinLogon" viene creata dalla funzione "RegCreateKeyExA"



4. SPIEGARE LE DUE ISTRUZIONI ALL'INDIRIZZO 00401027 – 00401029

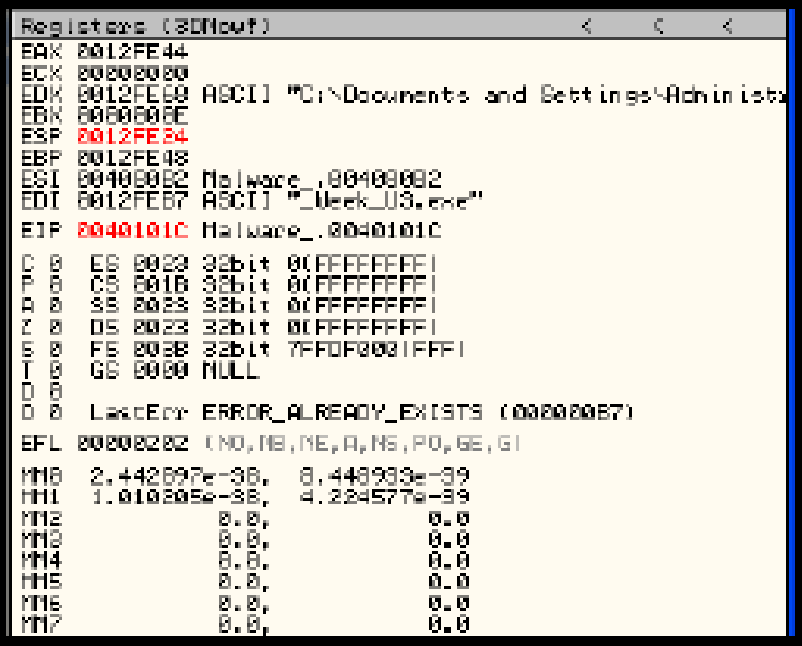
La prima istruzione è l'istruzione test nella locazione di memoria 00401027. È un'istruzione condizionale che è simile ad un "AND" logico, ma rispetto a quest'ultimo non modifica gli operandi. Quello che modifica è la ZF (zero-flag) che verrà settata a 1 se e solo se il risultato dell'operazione sarà 0.
La seconda istruzione è l'istruzione "JZ" la quale effettuerà un salto condizionale se e solo se lo ZF dell'operazione precedente sia settato a 1. in questo caso il salto verrà effettuato verso la locazione di memoria 401032.

5. TRADURRE IL CODICE ASSEMBLY IN C

Con questo codice abbiamo simulato il funzionamento del costrutto IF del nostro codice assembly. Abbiamo dichiarato le variabili per fare l'operazione di test tra i due operandi, dopodiché abbiamo impostato delle condizioni in base al risultato dell'operazione. Se il risultato del test fosse 0 allora la ZF verrà impostata a 1 e il salto avverrà, diversamente si continuerà con il codice.

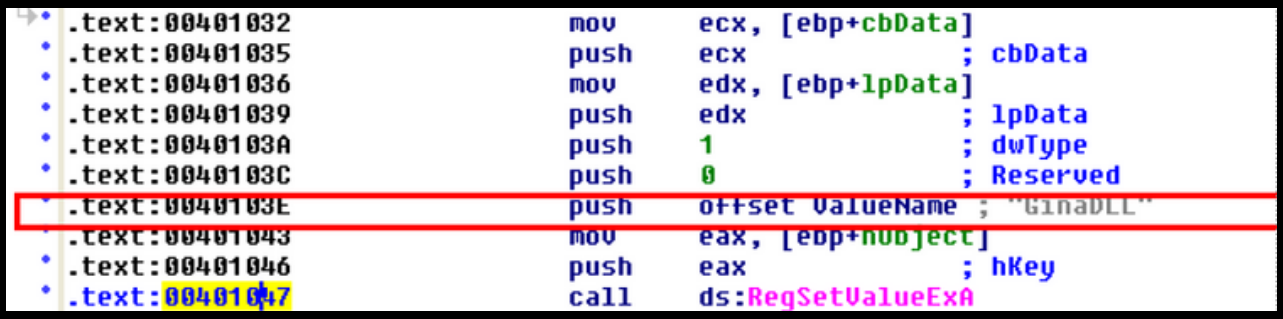
```c
#include <stdio.h>

int main()
{

int zf;  //questa variabile simula la ZF
int eax = 12; //questa variabile simula il registro eax
int test; //questa simula l'operatore test
test = eax - eax; // AND logico
if(test == 0){ // inizio del ciclo IF
    zf = 1;
}

if(zf == 1){ // se la ZF sarà uguale a 1 farà il salto alla locazione "401032"
    goto loc_401032;

}else{
    goto loc_40107B;} // altrimenti continuerà con il codice

loc_401032: printf("Salto avvenuto");
loc_40107B: printf("Salto non avvenuto");
    return 0;
}
```

Il valore che abbiamo impostato al registro "EAX" è 12, poiché è il valore decimale della versione esadecimale (0012FE44) che siamo riusciti a trovare tramite l'utilizzo di OllyDBG



6. VALORE DEL PARAMETRO “VALUENAME”

Con riferimento alla chiamata di funzione nella locazione di memoria 00401047, il valore che viene passato al parametro “ValueName” è GinaDLL



Per completezza abbiamo lanciato il file eseguibile del malware con il tool “OllyDBG” per verificare che il valore di “ValueName” fosse effettivamente quello riscontrato con IDA

FUNZIONALITA’ IMPLEMENTATE NELLA SEZIONE DI CODICE ANALIZZATA

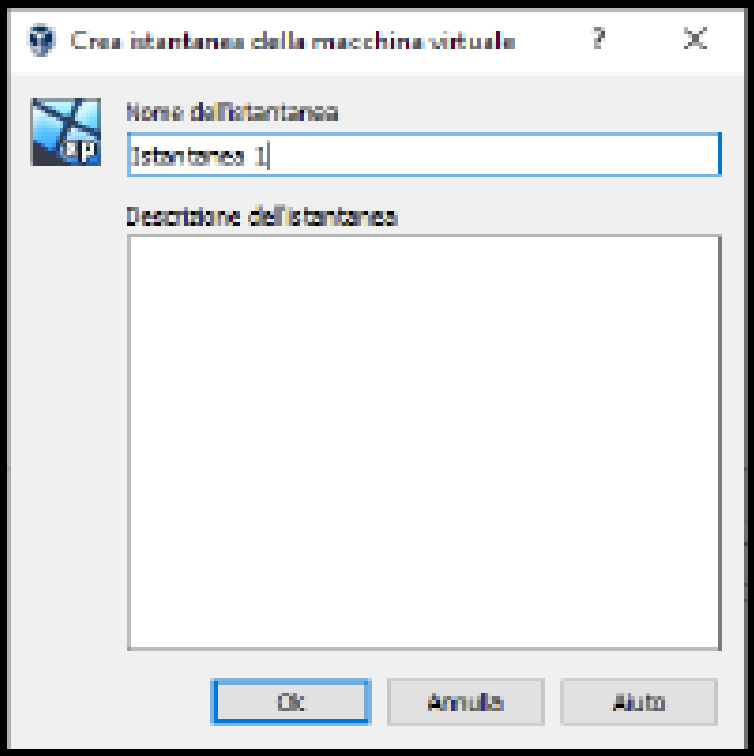
Con le righe di codice a nostra disposizione siamo riusciti a ipotizzare che il malware cerca di ottenere la persistenza all’interno del registro di Windows. Per fare ciò si serve della funzione “RegCreateKeyExA” (che crea la chiave di registro) e della funzione “RegSetValueExA” (che la configura).

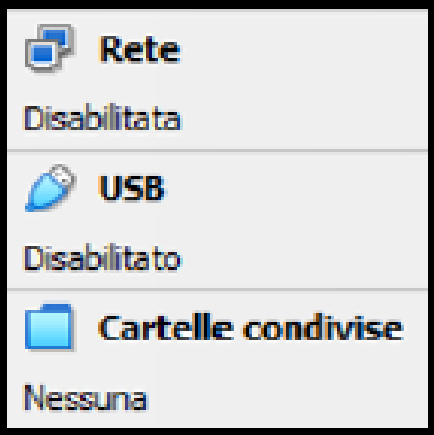
TRACCIA

1. Preparazione dell’ambiente di test sulla VM ed esecuzione del malware
2. Rilevazione di eventuali modifiche all’interno della cartella di origine dell’eseguibile in seguito all’esecuzione del malware
3. Analisi dell’interazione del malware con il registro: identificazione della chiave di registro creata e del valore ad essa associato
4. Analisi dell’interazione del malware con il file system: identificazione della chiamata di sistema responsabile della modifica del contenuto della cartella di origine dell’eseguibile
5. Descrizione del comportamento del malware in base alle informazioni raccolte tramite analisi statica e dinamica

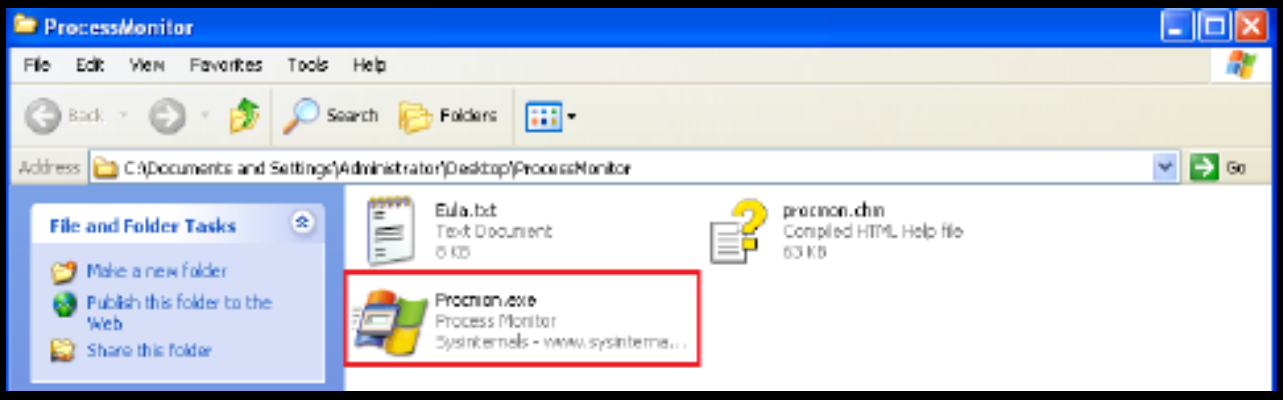
1. PREPARAZIONE DELL'AMBIENTE DI TEST SULLA VM ED ESECUZIONE DEL MALWARE

L'analisi dinamica basica comprende una gamma di attività di analisi che presuppongono l'esecuzione del malware in ambiente dedicato e protetto. È generalmente effettuata dopo l'analisi statica basica, per sopperire ai limiti di quest'ultima (infatti, al contrario dell'analisi statica, l'analisi dinamica permette di osservare in maniera diretta le funzionalità di un malware in esecuzione su un sistema) e confermare o smentire le ipotesi formulate durante l'analisi statica in merito al funzionamento del malware.
A monte delle operazioni di analisi, è utile creare un'istantanea della macchina virtuale di test (Malware Analysis_Final – OS Windows XP SP3) per poter eventualmente ripristinare le funzionalità della stessa in caso di problemi generati dall'esecuzione del malware; altrettanto importante è isolare l'ambiente di test disabilitando le schede di rete, la connettività USB ed eventuali cartelle condivise:

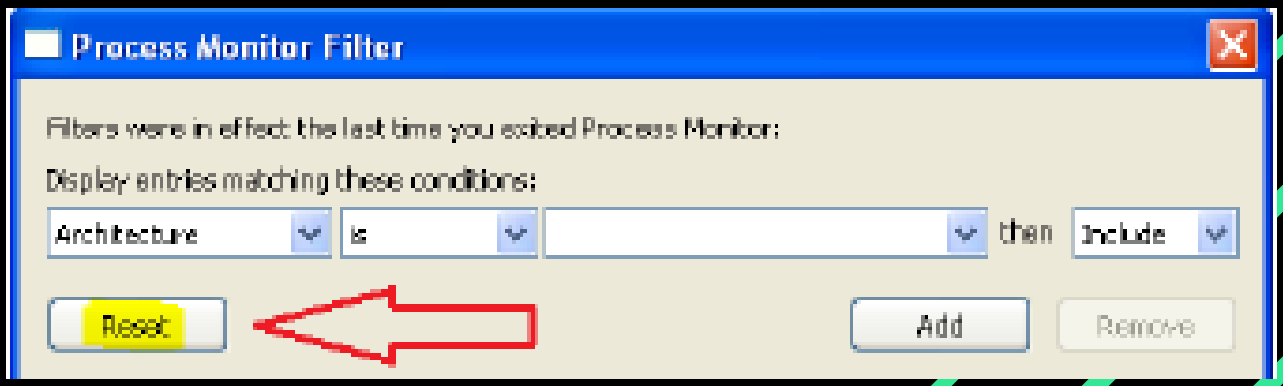




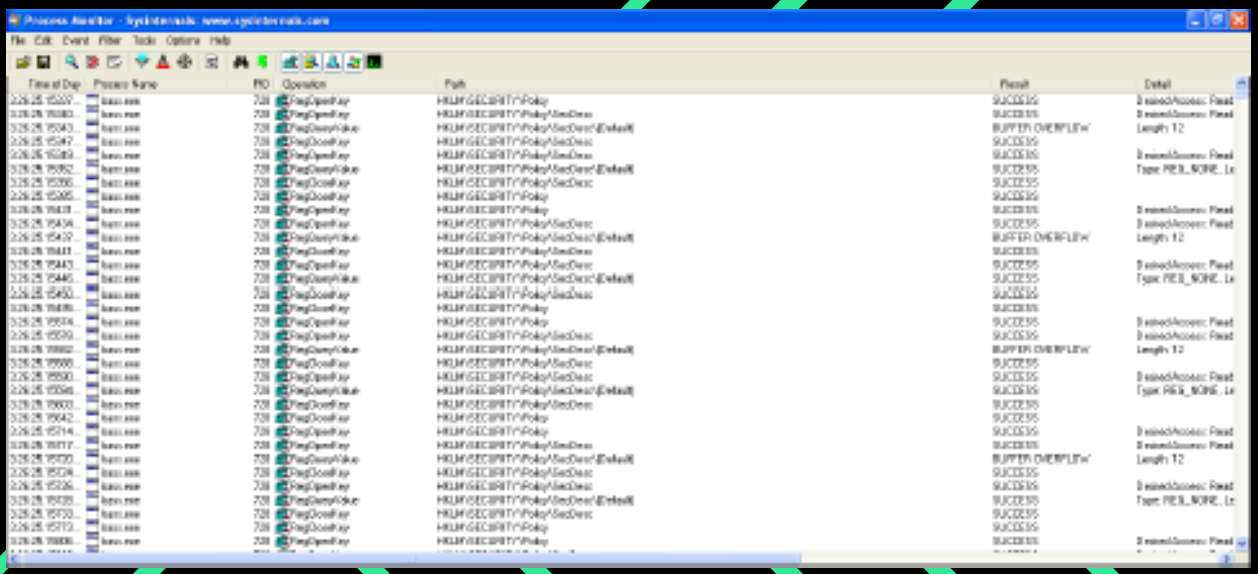
Adesso possiamo avviare la VM. Per le attività di analisi odierne, ci serviremo di Process Monitor: si tratta di un tool per sistemi operativi Windows che permette di monitorare i processi attivi, il file system, il registro di Windows e l'attività di rete.



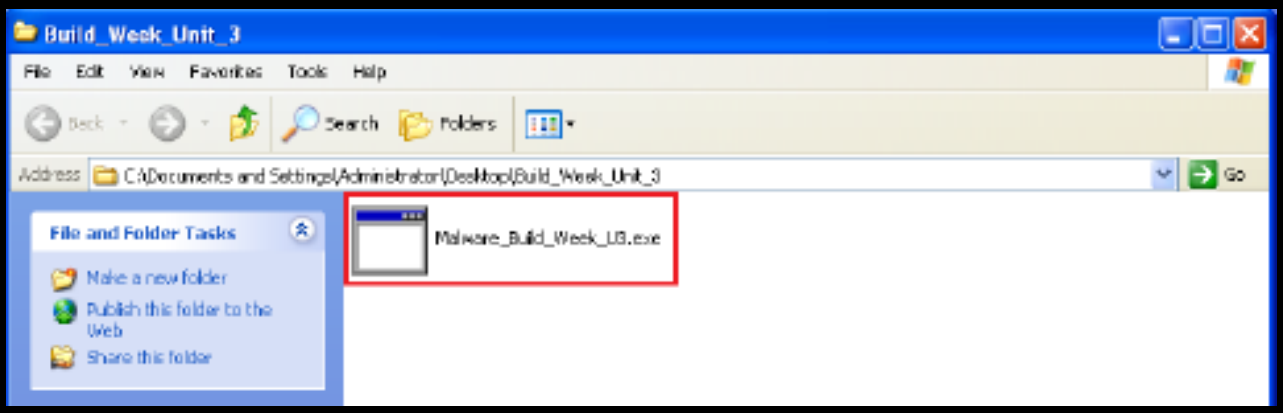
Una volta avviato il tool, ci accertiamo che non sia attivo alcun filtro cliccando sul tasto “Reset” per avere una panoramica completa della situazione iniziale:



Confermiamo la scelta con “Apply” ed “OK” ed il software inizia la cattura in tempo reale degli eventi relativi ai parametri di analisi già citati:

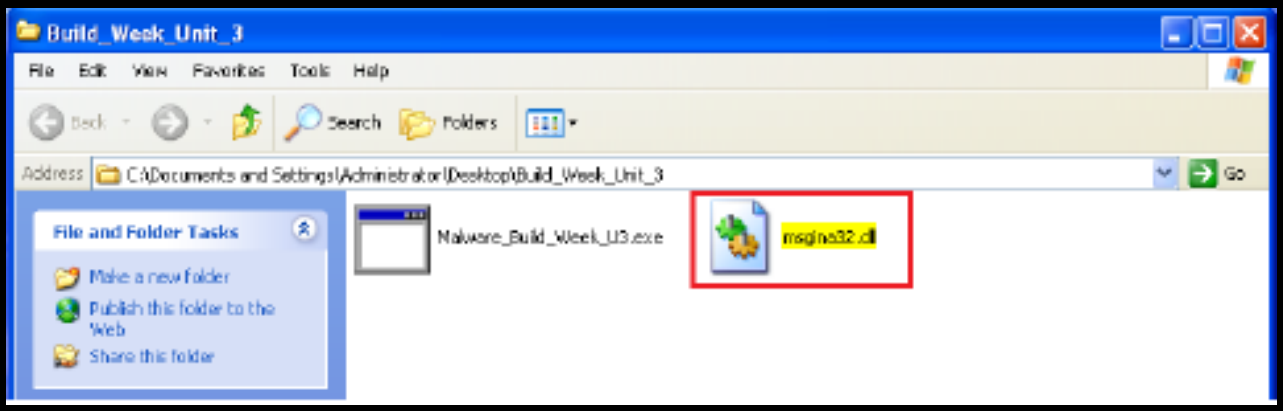


Adesso possiamo eseguire il malware da analizzare. Lo troviamo all'interno del path C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3. Lo avviamo facendo doppio click sull'icona dell'eseguibile.



2. RILEVAZIONE DI EVENTUALI MODIFICHE ALL'INTERNO DELLA CARTELLA DI ORIGINE DELL'ESEGUIBILE IN SEGUITO ALL'ESECUZIONE DEL MALWARE

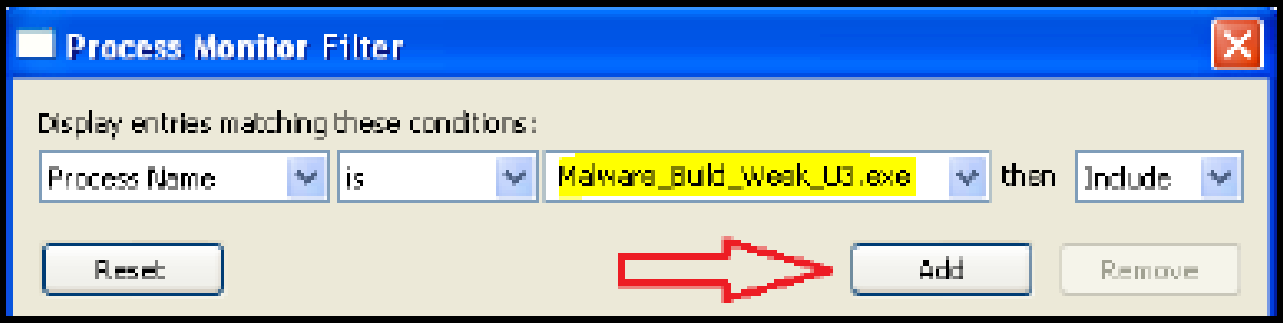
In seguito all'avvio del malware, notiamo che all'interno della cartella di origine dello stesso viene creato un nuovo file dal nome msgina32.dll:



Quanto appena visto conferma le ipotesi formulate durante l'analisi statica: il rilevamento delle chiamate di funzione FindResource, LoadResource, LockResource e SizeofResource importate dalla libreria KERNEL32.dll indica effettivamente che l'eseguibile analizzato è un dropper, ossia un programma malevolo che contiene al suo interno (specificamente, nella sezione "risorse" .rsrc dell'header del formato PE) un malware. Nel momento in cui viene eseguito, un dropper estrae il malware che contiene per avviarlo oppure salvarlo sul disco – in questo caso nello stesso path in cui si trova l'eseguibile.

3. ANALISI DELL'INTERAZIONE DEL MALWARE CON IL REGISTRO: IDENTIFICAZIONE DELLA CHIAVE DI REGISTRO CREATA E DEL VALORE AD ESSA ASSOCIATO

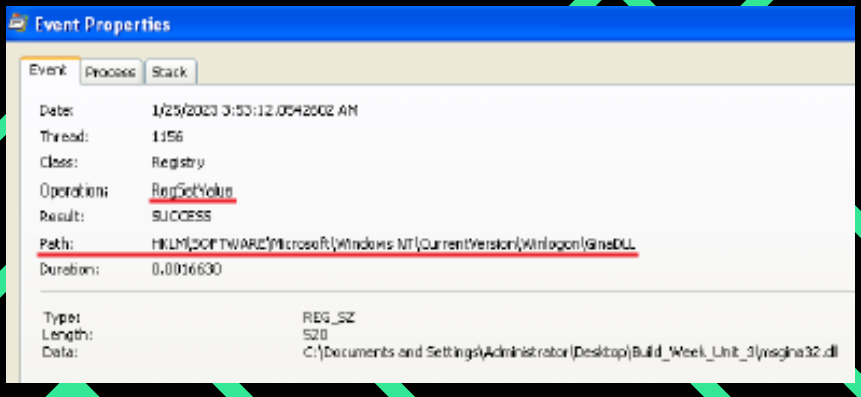
Per analizzare le modifiche apportate dal malware al sistema, filtriamo i risultati ottenuti dalla cattura di Process Monitor nel seguente modo:



In particolare, vogliamo filtrare ulteriormente i risultati in modo da mostrare solamente gli eventi relativi all'interazione del malware con il registro di Windows. Le chiavi di registro sono variabili di configurazione di sistema: di conseguenza è importante, in fase di analisi, conoscere le modifiche eventualmente apportate ad esse da parte di un malware, per capire quali configurazioni sono state alterate. I valori delle chiavi rappresentano tutto ciò che viene caricato all'avvio del sistema.

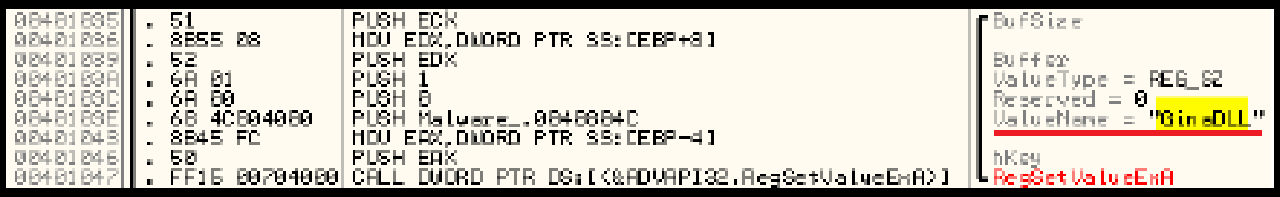
Impostiamo dunque il filtro sulla sezione Show Registry Activity →

Andiamo dunque ad analizzare gli eventi in base al filtro appena inserito e notiamo la seguente situazione:

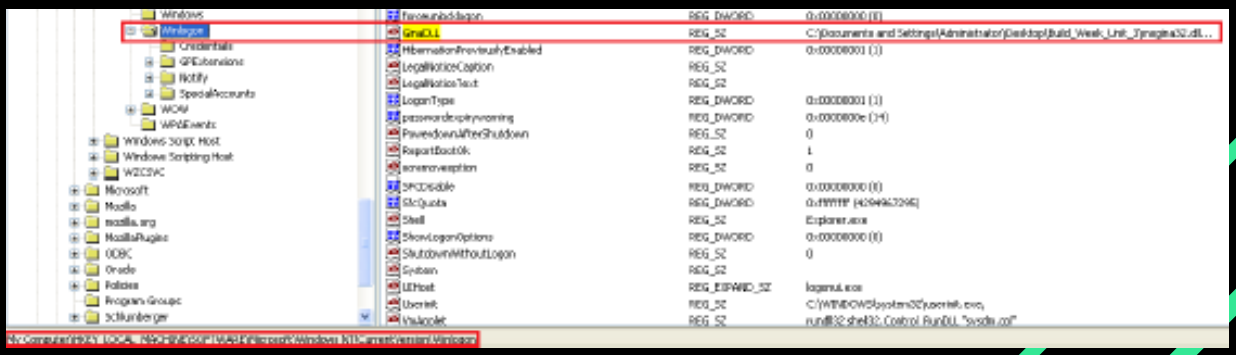


Come si può notare, tramite le chiamate di funzione RegCreateKey e RegSetValue il malware ha generato una chiave di registro all'interno del percorso Software\Microsoft\Windows NT\CurrentVersion\Winlogon e le ha successivamente assegnato il valore di "GinaDLL". Abbiamo verificato questa informazione anche servendoci di altri tool:

OllyDBG



Regedit (= Registry Editor, è uno strumento installato di default sui sistemi operativi Windows che viene utilizzato per visualizzare e/o modificare tutte le chiavi di registro)

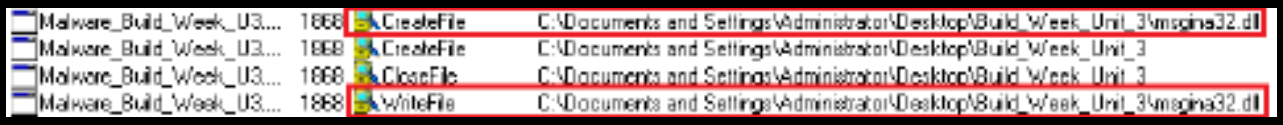


4. ANALISI DELL'INTERAZIONE DEL MALWARE CON IL FILE SYSTEM: IDENTIFICAZIONE DELLA CHIAMATA DI SISTEMA RESPONSABILE DELLA MODIFICA DEL CONTENUTO DELLA CARTELLA DI ORIGINE DELL'ESEGUIBILE

Adesso vogliamo invece analizzare l'interazione del malware con il file system allo scopo di individuare la chiamata di sistema responsabile della modifica del contenuto della cartella in cui si trova l'eseguibile in esame.

A tale scopo, impostiamo il filtro sulla sezione Show File System Activity →

Analizzando i risultati ottenuti, possiamo notare la presenza della chiamata di funzione CreateFile e, poco più avanti, della chiamata di funzione WriteFile:



Come si vede, entrambe le chiamate di funzione puntano al percorso in cui si trova il malware ed il loro scopo è quello di creare un file vuoto (CreateFile) e scrivere al suo interno il malware appena estratto (WriteFile).
Quanto appena emerso non ci sorprende in quanto il comportamento tipico di un dropper propone, generalmente, due possibili scenari che fanno seguito all'estrazione del malware:

- La creazione di un processo (tramite chiamata di funzione CreateProcess), al fine di eseguire immediatamente il malware appena estratto
-Il salvataggio del malware sul disco in vista di un futuro utilizzo. In questo scenario, analogamente a quanto visto oggi, il dropper utilizzerà la coppia di funzioni CreateFile e WriteFile rispettivamente per creare un file vuoto e scrivere al suo interno il malware appena estratto

5. DESCRIZIONE DEL COMPORTAMENTO DEL MALWARE IN BASE ALLE INFORMAZIONI RACCOLTE TRAMITE ANALISI STATICA E DINAMICA

A valle delle attività di analisi statica e dinamica fin qui svolte, è possibile trarre delle conclusioni sul comportamento del malware in esame. Inoltre, la verifica delle ipotesi formulate durante la fase di analisi statica ha avuto esito positivo: l'analisi dinamica ha confermato che il malware in oggetto è un dropper, in quanto contiene al suo interno un malware che viene effettivamente estratto (e salvato, in questo caso) al momento dell'esecuzione del file e ottiene la persistenza dentro il sistema della macchina vittima, in quanto aggiunge se stesso alle entry dei programmi che devono essere eseguiti all'avvio del PC, in modo tale da essere avviato in maniera automatica e senza alcun intervento da parte dell'utente.
A tale scopo, il malware crea una nuova chiave di registro tramite la chiamata di funzione RegCreateKeyExA e la configura con i valori desiderati tramite la chiamata di funzione RegSetValueExA. In fase di analisi dinamica, per verificare l'effettivo ottenimento della persistenza abbiamo riavviato la VM in seguito all'esecuzione del malware ed abbiamo potuto constatare che, al riavvio, la chiave "GinaDLL" è ancora presente nel registro di Windows.