

Malware Analysis W24D4

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:

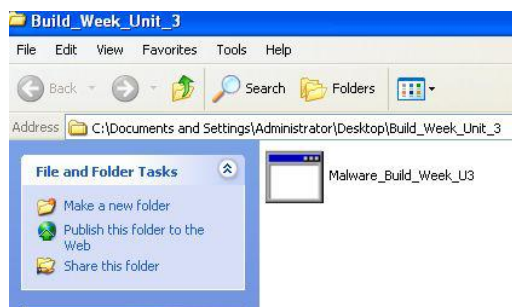
1. Quanti parametri sono passati alla funzione Main()?
2. Quante variabili sono dichiarate all'interno della funzione Main()?
3. Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate
4. Quali librerie importa il Malware? Per ognuna delle librerie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

Con riferimento al Malware in analisi, spiegare:

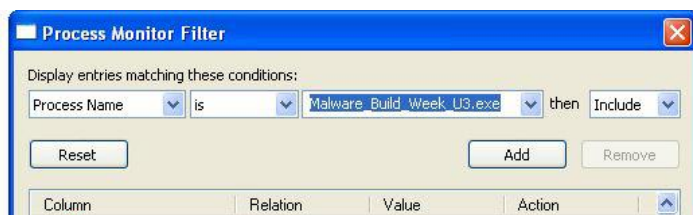
- Lo scopo della funzione chiamata alla locazione di memoria 00401021
- Come vengono passati i parametri alla funzione alla locazione 00401021;
- Che oggetto rappresenta il parametro alla locazione 00401017
- Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.
- Con riferimento 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

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



- 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?

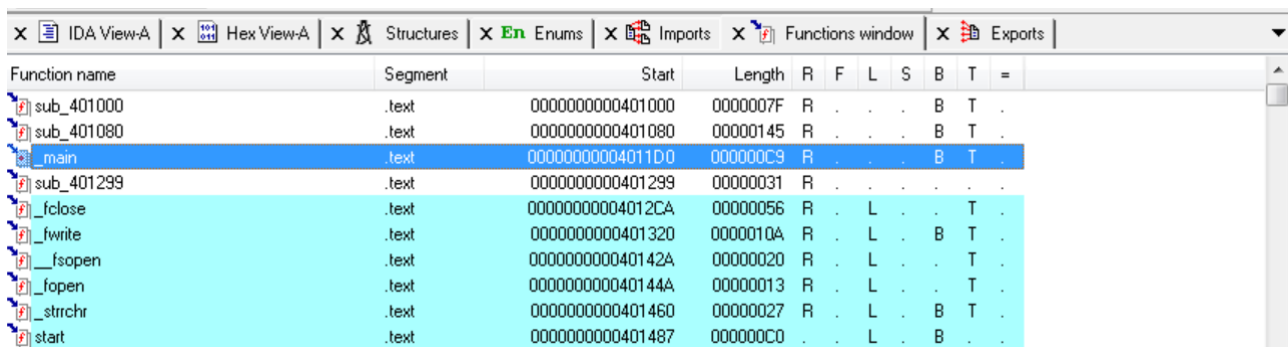
Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

Analisi Statica

Per l'analisi statica del Malware_Build_Week_U3, ho impiegato IDA (Interactive Disassembler). Questo strumento è ampiamente adottato per l'analisi del malware grazie alla sua capacità di decodificare il codice binario di programmi o eseguibili e di interpretare il codice assembly.

1. Quanti parametri sono passati alla funzione Main()?
2. Quante variabili sono dichiarate all'interno della funzione Main()?

Utilizzando IDA, sezione Functions, troviamo i parametri e le variabili. Sono stati identificati quattro variabili (hModule, var_8, var_4, data) e tre parametri (argc, argv, envp).

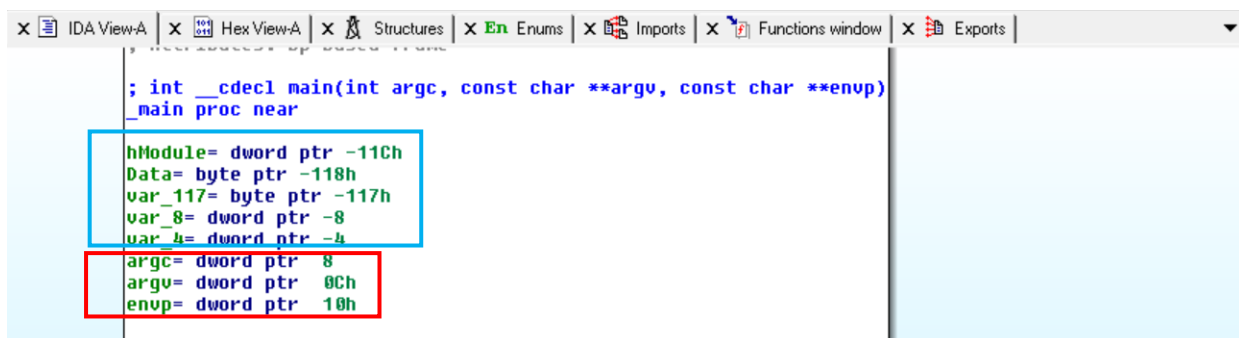


Function name	Segment	Start	Length	R	F	L	S	B	T	=
sub_401000	.text	0000000000401000	0000007F	R	.	.	.	B	T	.
sub_401080	.text	0000000000401080	00000145	R	.	.	.	B	T	.
main	.text	00000000004011D0	000000C9	R	.	.	.	B	T	.
sub_401299	.text	0000000000401299	00000031	R
_fclose	.text	00000000004012CA	00000056	R	.	L	.	.	T	.
_fwrite	.text	0000000000401320	0000010A	R	.	L	.	B	T	.
_fsopen	.text	000000000040142A	00000020	R	.	L	.	.	T	.
_fopen	.text	000000000040144A	00000013	R	.	L	.	.	T	.
_strchr	.text	0000000000401460	00000027	R	.	L	.	B	T	.
start	.text	0000000000401487	000000C0	.	.	L	.	B	.	.

Qui possiamo vedere i parametri e le variabili. Parametri valore positivo, variabili valore negativo.

3 Parametri: argc / argv / envp (quadrato rosso)

5 Variabili: hModule / Data / var_8 / var_4 / data (quadrato blu)



```
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

hModule= dword ptr -11Ch
Data= byte ptr -118h
var_117= byte ptr -117h
var_8= dword ptr -8
var_4= dword ptr -4

argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
```

3. Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate

Per le sezioni del malware ho usato CFF Explorer, un software per l'analisi dei file eseguibili Windows. Le sezioni trovate sono text, .rdata, .data, .rsrc.

.text: Questa sezione contiene il codice eseguibile del programma. È la sezione dove risiede il codice macchina che il processore esegue. Generalmente, questa sezione è marcata come sola lettura e di esecuzione, per impedire modifiche accidentali o maliziose al codice eseguibile.

.rdata: La sezione .rdata (read-only data) contiene dati di sola lettura che sono utilizzati dal programma. Questo può includere stringhe costanti, tabelle di indirizzi, e altri tipi di dati che non devono essere modificati durante l'esecuzione del programma. Questa sezione è generalmente marcata come sola lettura.

.data: La sezione .data contiene dati variabili, cioè variabili globali e statiche che possono essere modificate durante l'esecuzione del programma. Questa sezione è leggibile e scrivibile.

.rsrc: La sezione .rsrc (resource) contiene risorse utilizzate dall'applicazione, come icone, immagini, dialoghi, stringhe, e altri tipi di dati che sono accessibili attraverso le API di Windows. Questi dati possono essere utilizzati dal programma ma non sono parte del codice eseguibile.

Malware_Build_Week_U3.exe						
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword
.text	00005646	00001000	00006000	00001000	00000000	00000000
.rdata	000009AE	00007000	00001000	00007000	00000000	00000000
.data	00003EA8	00008000	00003000	00008000	00000000	00000000
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000

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.

Le librerie importate dal Malware sono **KERNEL32.dll** e **ADVAPI32.dll**.

Malware_Build_Week_U3.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IA)
0000769E	N/A	000074EC	000074F0	000074F4	000074F8	000074F0
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	00007000
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

ADVAPI32.dll: ADVAPI32.dll (Advanced API 32-bit) fornisce funzioni avanzate per la gestione della sicurezza, del registro di sistema e delle operazioni di servizio. Le funzioni RegSetValueA e RegCreateKeyA sono entrambe funzioni utilizzate per manipolare il Registro di sistema di Windows attraverso la libreria ADVAPI32.dll.

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

KERNEL32.dll: La libreria KERNEL32.dll è una parte fondamentale del sistema operativo Windows, e viene utilizzata per gestire molte funzionalità di basso livello come la gestione della memoria, i processi, i file e le comunicazioni con il sistema operativo. **Le funzioni importate sono 51**

KERNEL32.dll	51	00007534	00000000	00000000	0000769E	000070C
--------------	----	----------	----------	----------	----------	---------

Malware_Build_Week_U3.exe			
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

Malware_Build_Week_U3.exe			
OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
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

5. Ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare

1. Il malware può dinamicamente accedere e utilizzare risorse esterne, come funzioni o librerie, tramite l'utilizzo di metodi come **GetProcAddress** e **LoadLibraryA**.
2. È probabile che il malware faccia uso di risorse del sistema per svolgere determinate azioni, come trovare risorse, liberare risorse, caricare risorse in memoria, bloccare risorse e ottenere le dimensioni delle risorse. (**FindResourceA** , **FreeResource** , **LoadResource** , **LockResource** e **SizeofResource**)
3. Esistono numerose importazioni che consentono al malware di interagire con i file presenti sul sistema, tra cui scrivere su file, creare nuovi file, leggere file esistenti, e altre operazioni simili. (**CreateFileA**, **ReadFile**, **WriteFile**, etc.)
4. Inoltre, il malware potrebbe sfruttare le funzioni di gestione della memoria per scopi dannosi, come l'allocazione di spazio per eseguire codice maligno, nascondere dati sensibili o influenzare il funzionamento di altre applicazioni. Queste funzioni includono **VirtualAlloc**, **HeapAlloc**, **VirtualFree**, **HeapFree** e altre simili.

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

1 e 2: 00401021 ha come funzione RegSetValueExA , che consente di creare una nuova chiave nel registro di sistema di Windows, specificando il nome della chiave, i diritti di accesso, un oggetto di sicurezza opzionale che determina l'accesso alla chiave e altri parametri opzionali, i parametri gli vengono passati attraverso la funzione PUSH (riquadro rosso);

```
* .text:00401011      push     0             ; dwOptions
* .text:00401013      push     0             ; lpClass
* .text:00401015      push     0             ; Reserved
* .text:00401017      push     offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\Cu
* .text:0040101C      push     80000002h     ; hKey
* .text:00401021      call     ds:RegCreateKeyExA
```

Per la funzione RegCreateKeyExA, l'ordine dei parametri è il seguente:

- hKey (handle della chiave di registro padre)
- lpSubKey (puntatore alla stringa che specifica il nome della sottochiave)
- Reserved (generalmente usato come zero)
- lpClass (puntatore alla stringa che contiene il nome della classe del valore da creare o NULL)
- dwOptions (opzioni per l'apertura/creazione della chiave)
- samDesired (accesso desiderato alla chiave)
- lpSecurityAttributes (puntatore a strutture di sicurezza, solitamente NULL)
- phkResult (puntatore a una variabile dove viene restituito l'handle della chiave aperta o creata)
- lpdwDisposition (indica se una chiave di registro è stata creata nuova o aperta se già esistente, in questo è stata creata.)

```
push     0             ; lpdwDisposition
lea      eax, [ebp+hObject]
push     eax           ; phkResult
push     0             ; lpSecurityAttributes
push     0F003Fh       ; samDesired
push     0             ; dwOptions
push     0             ; lpClass
push     0             ; Reserved
push     offset SubKey ; "SOFTWARE\\Microsoft\\Windows
push     80000002h     ; hKey
```

3. Alla funzione 00401017 troviamo **offset Subkey**. Nel contesto della funzione 00401017, l'offset Subkey svolge il ruolo di indicare o identificare la chiave di registro. In questo caso specifico, si riferisce alla funzione RegCreateKeyExA, utilizzata per creare una nuova chiave nel registro di sistema di Windows.

```

.text:00401004      push     0                ; lpdwDispos
.text:00401006      lea     eax, [ebp+hObject]
.text:00401009      push     eax             ; phkResult
.text:0040100A      push     0                ; lpSecurity
.text:0040100C      push     0F003Fh         ; samDesired
.text:00401011      push     0                ; dwOptions
.text:00401013      push     0                ; lpClass
.text:00401015      push     0                ; Reserved
.text:00401017      push     offset SubKey    ; "SOFTWARE\

```

La sottochiave indica il percorso all'interno del Registro di sistema in cui verrà creata o aperta la chiave. Nell'istanza attuale, il nome della sottochiave sembra essere "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon".

```

; char SubKey[]
SubKey      db 'SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon',0
; DATA XREF: sub_401000+17fo

```

4 Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029. (Riquadro rosso)

- **test eax, eax:** Questa istruzione esegue un'operazione logica di AND tra il registro eax e se stesso. Questa operazione imposta i flag del processore in base al risultato del test.
- **jz short loc_401032:** Questa istruzione è un salto condizionale che viene eseguito solo se il flag Zero (ZF) è impostato dopo l'istruzione di test precedente. In particolare, "jz" significa "jump if zero" e salta a "loc_401032" se il flag Zero (ZF) è impostato, altrimenti continua l'esecuzione sequenziale del codice.

```

.text:0040100C      push     0F003Fh         ; samDesired
.text:00401011      push     0                ; dwOptions
.text:00401013      push     0                ; lpClass
.text:00401015      push     0                ; Reserved
.text:00401017      push     offset SubKey    ; "SOFTWARE\
.text:0040101C      push     80000002h        ; hKey
.text:00401021      call     ds:RegCreateKeyExA
.text:00401027      test     eax, eax
.text:00401029      jz       short loc_401032

```

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

```

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

```

In questa traduzione, si verifica se il valore nel registro eax è uguale a zero. Se lo è, il controllo del programma viene trasferito all'indirizzo loc_401032. Altrimenti, il programma continua a eseguire le istruzioni successive.

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

Il valore del parametro ValueName è "GinaDLL". ValueName è il parametro che identifica il nome del valore associato alla chiave di registro utilizzato nella funzione RegSetValueExA.

```
.text:00401039      push    eax                ; 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]
```

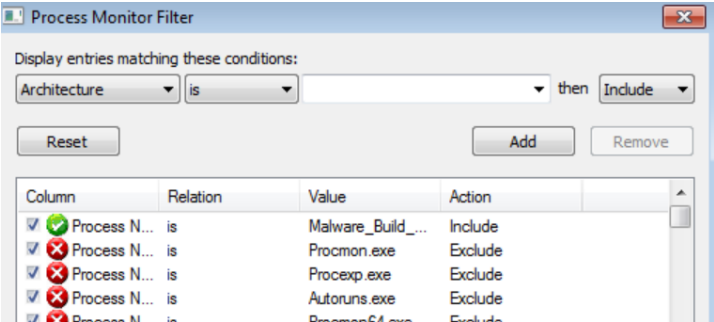
```
; char ValueName[]
ValueName db 'GinaDLL',0 ; DATA XREF: sub_401000+3E10
```

Analisi dinamica

Process Monitor è un'applicazione per il monitoraggio avanzato del sistema operativo Windows, sviluppata da Microsoft. Consente agli utenti di visualizzare in tempo reale l'attività del sistema operativo, inclusi i processi in esecuzione, le operazioni sui file, l'accesso al registro di sistema e altre attività di sistema.

Questo strumento fornisce una dettagliata registrazione di tutte le attività del sistema, consentendo agli utenti di analizzare e risolvere problemi di sistema, diagnosticare malware, monitorare l'attività di applicazioni e driver, e identificare eventuali inefficienze o problemi di prestazioni.

Una volta aperto il programma possiamo resettare i filtri e impostare quelli che ci servono per l'analisi. Possiamo procedere all'avvio del Malware con doppio clic, dopo averlo caricato nella sezione di monitoraggio.

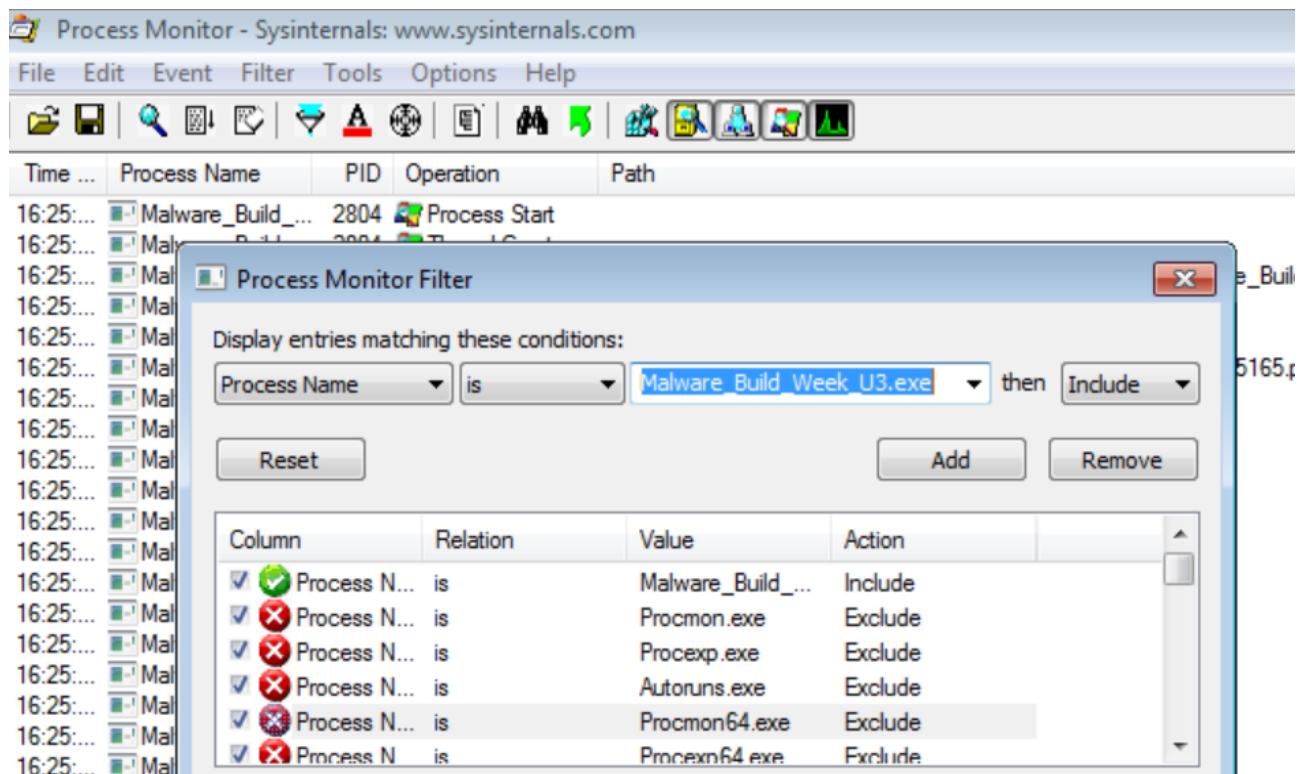


Appena caricato sono comparsi altri file.

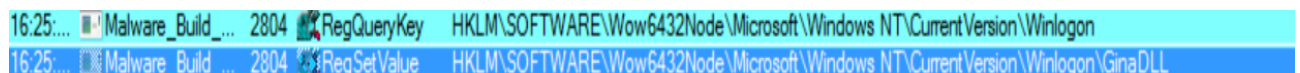
Nome	Ultima modifica	Tipo	Dimensione
Malware_Build_Week_U3	17/01/2024 17:48	Applicazione	52 KB
Malware_Build_Week_U3.id0	29/05/2024 15:37	File ID0	544 KB
Malware_Build_Week_U3.id1	29/05/2024 15:38	File ID1	184 KB
Malware_Build_Week_U3.nam	29/05/2024 15:38	File NAM	16 KB
Malware_Build_Week_U3.til	29/05/2024 15:38	File TIL	1 KB
msgina32.dll	29/05/2024 16:25	Estensione dell'ap...	7 KB

Malware Analysis

Ora bisogna analizzare i processi che il malware ha eseguito o tentato di eseguire, in questo caso si utilizzano i filtri per filtrare e il nome del processo.

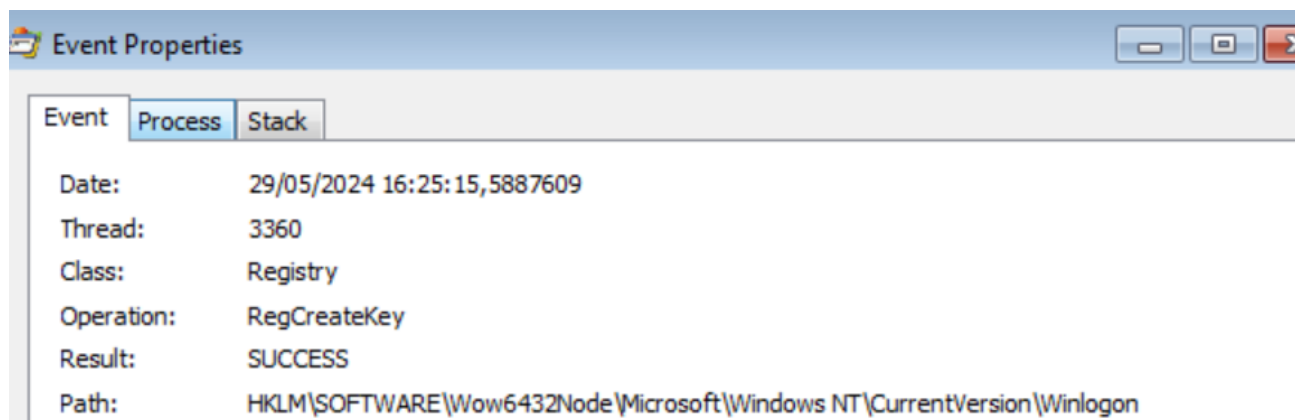


Grazie a questo controllo sono spuntate delle funzioni, in questo caso **RegCreateKey** e **RegSetValue**

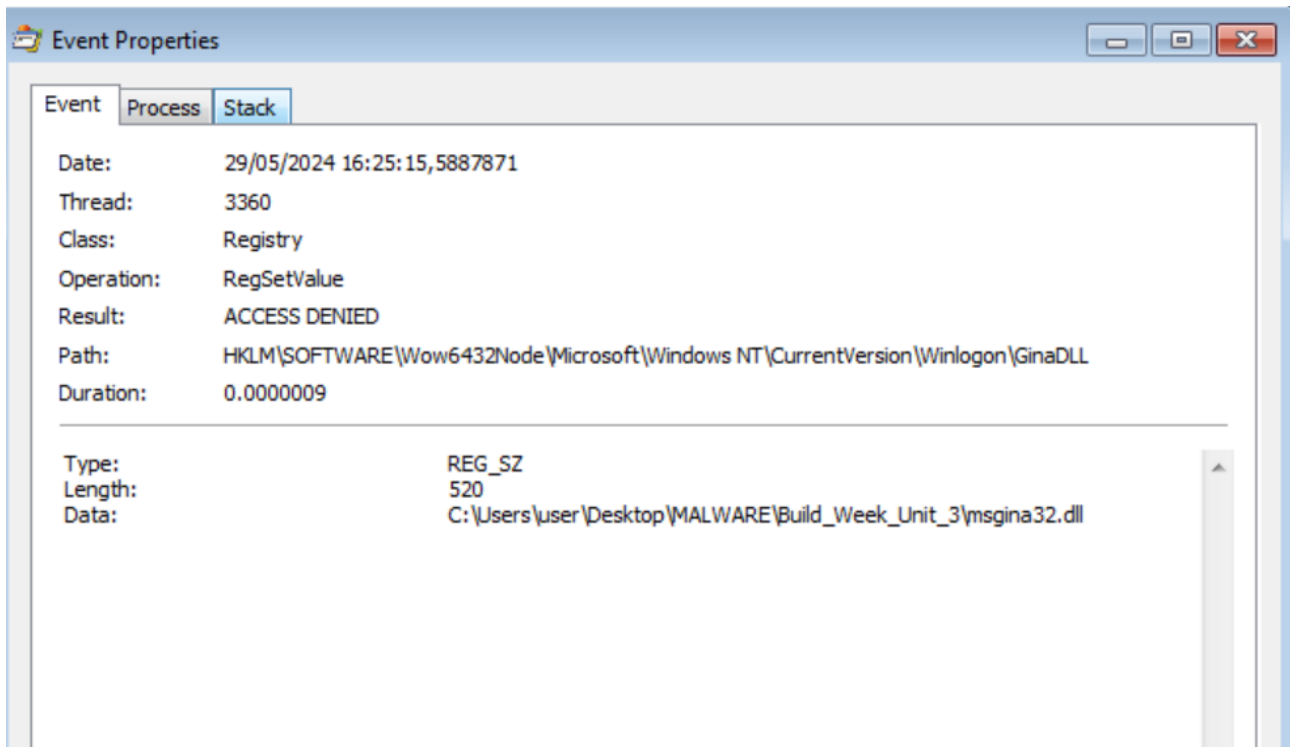


RegCreateKey: Questa funzione viene utilizzata per creare una nuova chiave nel registro di sistema o aprire una chiave esistente se già presente. Cliccando sulla voce RegCreateKey, che ci offre un dettaglio più preciso come ad esempio il Path HKLM (HKEY_LOCAL_MACHINE)

HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon



RegSetValue: Questa funzione viene utilizzata per impostare il valore dei dati associati a una chiave di registro esistente. **RegSetValue** ha assegnato il valore della chiave di registro utilizzando il file precedentemente creato, ossia **msgina32.dll**.



File System: Il malware ha creato e modificato dei file msgina32.dll.

16:25:...	Malware_Build_...	2804	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
16:25:...	Malware_Build_...	2804	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
16:25:...	Malware_Build_...	2804	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
16:25:...	Malware_Build_...	2804	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
16:25:...	Malware_Build_...	2804	Thread Exit	

In conclusione, possiamo affermare con sicurezza che il malware mira a influenzare le configurazioni del sistema compromesso, agendo sulla chiave di registro tramite un altro malware identificato come msgina32.dll. Questo agente dannoso interviene modificando una chiave critica del sistema, GINADLL, la quale è strettamente associata al file gina.dll utilizzato nei sistemi operativi Windows precedenti a Windows Vista per gestire il processo di autenticazione degli utenti durante il login.

L'intervento su questa chiave potrebbe aprire la porta a accessi non autorizzati o indesiderati, mettendo così a rischio la sicurezza complessiva del sistema. La modifica di tali impostazioni costituisce un chiaro segnale di intenti dannosi, e richiede un'azione immediata per prevenire potenziali danni e ripristinare l'integrità del sistema compromesso.