

Cybersecurity Analyst

Week 24 - W24D4

M6: Malware Analysis

Progetto

ANALISI STATICA

Per effettuare l'analisi statica del malware in questione (Malware_Build_Week_Unit_3.exe), è stato utilizzato il tool **IDA**, il quale permette di disassemblare il software in codice Assembly.

1. Quanti parametri sono passati alla funzione Main()?

Nel codice Assembly è possibile vedere che i parametri passati alla funzione Main() sono 3: **argc**, **argv**, **envp**.

Quest'ultimi sono parametri in quanto hanno **offset positivo rispetto ad EPB** ed inoltre sono dichiarati nella definizione della funzione stessa.

```
; 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
```

2. Quante variabili sono dichiarate all'interno della funzione Main()?

Come per i parametri, anche le variabili sono visualizzabili nella sezione iniziale del codice; è possibile infatti notare che le variabili nella funzione Main() sono 5:

hModule, **Data**, **var_117**, **var_8**, **var_4**.

Quest'ultimi, a differenza dei parametri, **hanno offset negativo rispetto ad EPB**.

```
; 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.





Le sezioni presenti nell'eseguibile sono:

.text (contiene il codice eseguibile del programma)

.data (contiene dichiarazioni di variabili e costanti)


















.rdata (contiene info riguardo le librerie e le funzioni utilizzate)

.idata

Name	Start	End	R	W	X	D	L	Align	Base	Type	Class
 .text	00401000	00407000	R	.	X	.	L	para	0001	public	CODE
 .idata	00407000	004070DC	R	.	.	.	L	para	0002	public	DATA
 .rdata	004070DC	00408000	R	.	.	.	L	para	0002	public	DATA
 .data	00408000	0040C000	R	W	.	.	L	para	0003	public	DATA

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 **ADVAPI32.dll** e **KERNEL32.dll**.

Address	Ordinal	Name	Library
 00407000		RegSetValueExA	ADVAPI32
 00407004		RegCreateKeyExA	ADVAPI32
 0040700C		SizeofResource	KERNEL32
 00407010		LockResource	KERNEL32
 00407014		LoadResource	KERNEL32
 00407018		VirtualAlloc	KERNEL32
 0040701C		GetModuleFileNameA	KERNEL32
 00407020		GetModuleHandleA	KERNEL32
 00407024		FreeResource	KERNEL32
 00407028		FindResourceA	KERNEL32
 0040702C		CloseHandle	KERNEL32
 00407030		GetCommandLineA	KERNEL32
 00407034		GetVersion	KERNEL32
 00407038		ExitProcess	KERNEL32
 0040703C		HeapFree	KERNEL32
 00407040		GetLastError	KERNEL32
 00407044		WriteFile	KERNEL32

ADVAPI32.dll

È una libreria che include funzioni principalmente orientate alla gestione del registro di sistema, per poterne appunto modificare i valori e gestire utenti e permessi.

Nel caso specifico preso in esame, possiamo notare che il malware infatti importa la seguente libreria per poter utilizzare le funzioni **RegCreateKeyExA** (utilizzata per creare una nuova chiave di registro) e **RegSetValueExA** (usata per impostare il valore di una chiave di registro).

KERNEL32.dll

È una libreria che contiene le funzioni principali per interagire con il sistema operativo, ad esempio manipolazione dei file e la gestione della memoria.

Nel caso del malware in questione, notiamo che tramite questa libreria vengono poi richiamate le funzioni di manipolazione dei file come **CreateFileA**, **ReadFile**, **WriteFile**.

Con riferimento al Malware in analisi, spiegare:

1. Lo scopo della funzione chiamata alla locazione di memoria 00401021:

```
.text:00401021      call     ds:RegCreateKeyExA
```

L'indirizzo 00401021 corrisponde ad una call alla funzione **RegCreateKeyExA**, utilizzata per operare sul registro di sistema e creare (o aprire qualora esistesse) una chiave specifica all'interno dello stesso.

2. Come vengono passati i parametri alla funzione alla locazione 00401021:

I parametri per la funzione alla locazione 00401021 sono passati utilizzando istruzioni **push** per inserirli nello stack prima di effettuare la chiamata.

3. Che oggetto rappresenta il parametro alla locazione 00401017:

```
.text:00401017 push offset SubKey;"SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
```

Il parametro alla locazione 00401017 rappresenta il **percorso nel registro di sistema** che la funzione RegCreateKeyExA utilizzerà in seguito per creare una nuova chiave di registro.

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

```
.text:00401027 test    eax, eax
.text:00401029 jz      short loc_401032
```

L'istruzione **test** effettua un'operazione logica AND tra il registro eax e sé stesso. Questa istruzione non modifica il valore di eax, ma questo confronto serve a settare lo **ZF (Zero Flag)** a 1 nel caso eax valga zero, in caso contrario ZF verrà settato a 0.

Una volta settato lo ZF, la successiva istruzione **jz (jump if zero)** eseguirà eventualmente un salto condizionale alla locazione 401032 nel caso ZF sia stato precedentemente settato a 1 dopo l'istruzione di test.

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

Il codice Assembly visto sopra può essere tradotto in un corrispondente codice C partendo dal presupposto che la comparazione test e l'eventuale successivo salto condizionale sono paragonabili all'utilizzo dell'istruzione **if** in C.

Dato questo, si potrebbe avere il seguente costruito C:

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

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

```

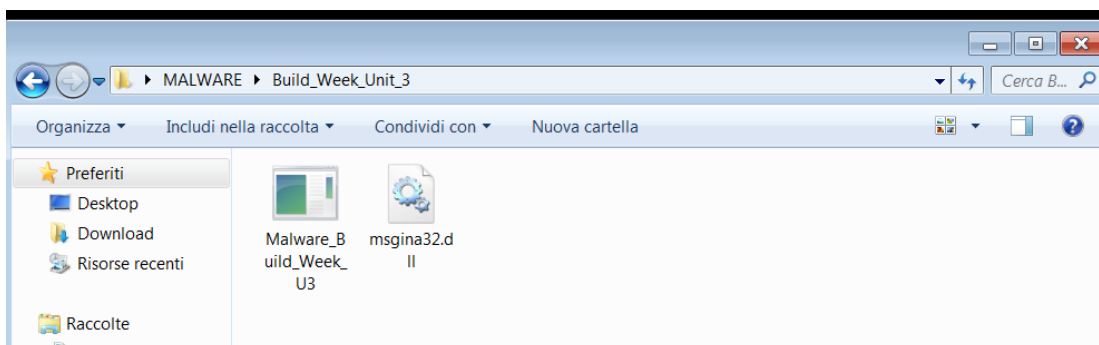
.text:00401032 loc_401032:
.text:00401032 mov     ecx, [ebp+cbData] ; CODE XREF: sub_401000+29↑j
.text:00401035 push    ecx               ; cbData
.text:00401036 mov     edx, [ebp+lpData]
.text:00401039 push    edx               ; lpData
.text:0040103A push    1                 ; dwType
.text:0040103C 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
  
```

Il valore del parametro «ValueName» è la stringa "GinaDLL", che è specificata dall'istruzione push offset ValueName. Considerata la chiamata alla locazione 00401047 alla funzione RegSetValueExA possiamo ipotizzare che il malware possa essere progettato per settare un nuovo valore di nome "GinaDLL".

ANALISI DINAMICA

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.

All'interno della cartella dove risiede l'eseguibile, è stato creato un nuovo file nominato **msgina32.dll**, il quale presumibilmente è legato al parametro «ValueName» che il malware associa alla funzione RegSetValueExA vista durante l'analisi statica.



Per continuare l'analisi dinamica, è stato utilizzato il tool **Procmon** (Process Monitor), uno strumento di monitoraggio progettato per analizzare le attività di sistema in tempo reale su sistemi Windows, fornendo una visione dettagliata delle operazioni dei processi, inclusi accessi al registro di sistema, operazioni su file, attività di rete etc.

Dal tool in questione, filtrando i risultati tramite nome del processo, è possibile rilevare le operazioni coinvolte dopo l'apertura dell'eseguibile.

REGISTRO DI SISTEMA

Quale chiave di registro viene creata e quale valore viene associato?

1725...	1748	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	SUCCESS	KeySetInformationClass: KeySetHandleTa...
1725...	1748	RegQueryValue	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions\{Default}	SUCCESS	Type: REG_SZ, Length: 36, Data: 00060101...
1725...	1748	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	REPARSE	Desired Access: Read
1725...	1748	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Access: Read
1725...	1748	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	KeySetInformationClass: KeySetHandleTa...
1725...	1748	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	NAME NOT FOUND	Length: 548
1725...	1748	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSUserEnabled	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
1725...	1748	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
1725...	1748	RegOpenKey	HKLM	SUCCESS	Desired Access: Maximum Allowed, Grante...
1725...	1748	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
1725...	1748	RegOpenKey	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Diagnostics	NAME NOT FOUND	Desired Access: Read
1725...	1748	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
1725...	1748	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All Access, Disposition: R...
1725...	1748	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	KeySetInformationClass: KeySetHandleTa...
1725...	1748	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Query: HandleTags, HandleTags: 0x400
1725...	1748	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	ACCESS DENIED	Type: REG_SZ, Length: 520, Data: C:\User...
1725...	1748	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	
1725...	1748	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	SUCCESS	
1725...	1748	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	SUCCESS	
1725...	1748	RegCloseKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	SUCCESS	
1725...	1748	RegCloseKey	HKLM	SUCCESS	

Dal log di Procmon è possibile vedere che la chiave di registro creata dal malware (o aperta qualora già esistesse) è:

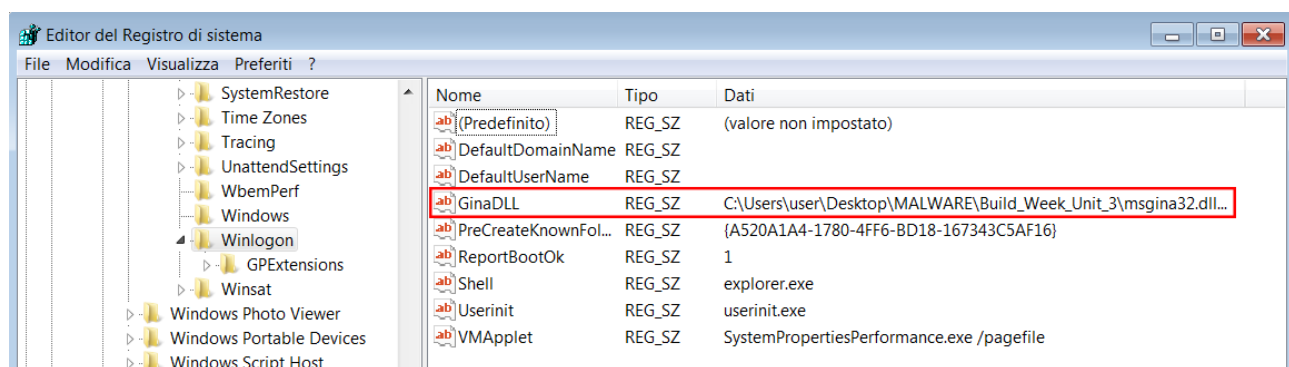
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon.

Il nuovo valore invece che il malware ha cercato di aggiungere ad essa è **GinaDLL**, che (seguendo il percorso nei dettagli dell'operazione) fa riferimento al file **msgina32.dll** creato dal malware al suo avvio.

Tuttavia, possiamo notare come quest'ultima operazione abbia come risultato **ACCESS DENIED**, il che significa sia stata negata probabilmente a causa di autorizzazioni insufficienti.

Per avvalorare questa ipotesi, è stata effettuata una seconda analisi eseguendo volontariamente il software con **privilegi da amministratore**: in questo scenario si può notare come il malware riesca a completare questa operazione e aggiungere il valore GinaDLL alla chiave di registro specificata.

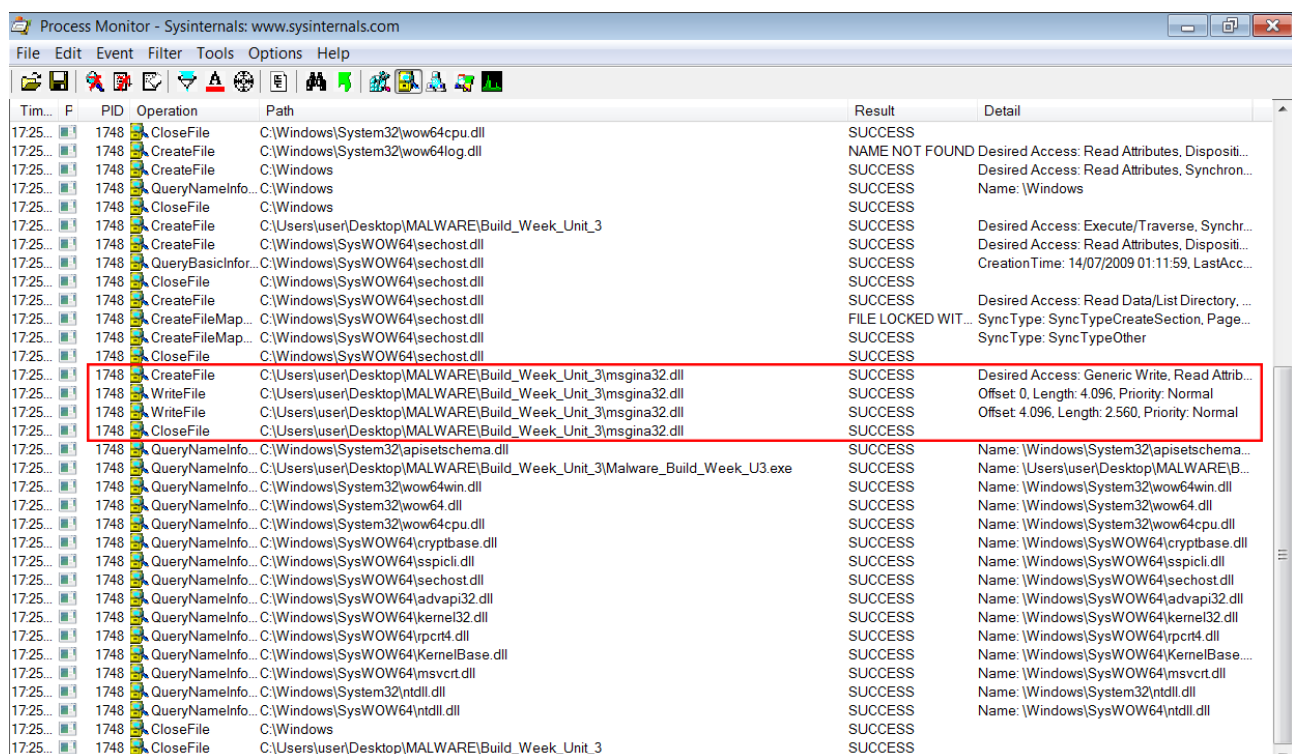
11:53:25,7...	Malw...	3056	RegOpenKey	HKLM	SUCCESS	Desired Access: Maximum Allowed, Gra...
11:53:25,7...	Malw...	3056	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
11:53:25,7...	Malw...	3056	RegOpenKey	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Diagnostics	NAME NOT FOUND	Desired Access: Read
11:53:25,7...	Malw...	3056	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
11:53:25,7...	Malw...	3056	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All Access, Disposition...
11:53:25,7...	Malw...	3056	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	KeySetInformationClass: KeySetHandle...
11:53:25,7...	Malw...	3056	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Query: HandleTags, HandleTags: 0x400
11:53:25,7...	Malw...	3056	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	SUCCESS	Type: REG_SZ, Length: 520, Data: C:\Us...
11:53:25,7...	Malw...	3056	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	
11:53:25,7...	Malw...	3056	RegCloseKey	HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\Image File Execution O...	SUCCESS	
11:53:25,7...	Malw...	3056	RegCloseKey	HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\Image File Execution O...	SUCCESS	



Nome	Tipo	Dati
(Predefinito)	REG_SZ	(valore non impostato)
DefaultDomainName	REG_SZ	
DefaultUserName	REG_SZ	
GinaDLL	REG_SZ	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll...
PreCreateKnownFol...	REG_SZ	{A520A1A4-1780-4FF6-BD18-167343C5AF16}
ReportBootOk	REG_SZ	1
Shell	REG_SZ	explorer.exe
Userinit	REG_SZ	userinit.exe
VMApplet	REG_SZ	SystemPropertiesPerformance.exe /pagefile

ATTIVITÀ SUL FILE SYSTEM

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



Time	PID	Operation	Path	Result	Detail
17:25...	1748	CloseFile	C:\Windows\System32\wow64cpu.dll	SUCCESS	
17:25...	1748	CreateFile	C:\Windows\System32\wow64log.dll	NAME NOT FOUND	Desired Access: Read Attributes, Dispositi...
17:25...	1748	CreateFile	C:\Windows	SUCCESS	Desired Access: Read Attributes, Synchron...
17:25...	1748	QueryNameInfo...	C:\Windows	SUCCESS	Name: \Windows
17:25...	1748	CloseFile	C:\Windows	SUCCESS	
17:25...	1748	CreateFile	C:\Users[user\Desktop\MALWARE]\Build_Week_Unit_3	SUCCESS	Desired Access: Execute/Traverse, Synchr...
17:25...	1748	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Desired Access: Read Attributes, Dispositi...
17:25...	1748	QueryBasicInfo...	C:\Windows\SysWOW64\sechost.dll	SUCCESS	CreationTime: 14/07/2009 01:11:59, LastAcc...
17:25...	1748	CloseFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	
17:25...	1748	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Desired Access: Read Data/List Directory, ...
17:25...	1748	CreateFileMap...	C:\Windows\SysWOW64\sechost.dll	FILE LOCKED WITH...	SyncType: SyncTypeCreateSection, Page...
17:25...	1748	CreateFileMap...	C:\Windows\SysWOW64\sechost.dll	SUCCESS	SyncType: SyncTypeOther
17:25...	1748	CloseFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	
17:25...	1748	CreateFile	C:\Users[user\Desktop\MALWARE]\Build_Week_Unit_3\msgina32.dll	SUCCESS	Desired Access: Generic Write, Read Attrib...
17:25...	1748	WriteFile	C:\Users[user\Desktop\MALWARE]\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset 0, Length: 4,096, Priority: Normal
17:25...	1748	WriteFile	C:\Users[user\Desktop\MALWARE]\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset 4,096, Length: 2,560, Priority: Normal
17:25...	1748	CloseFile	C:\Users[user\Desktop\MALWARE]\Build_Week_Unit_3\msgina32.dll	SUCCESS	
17:25...	1748	QueryNameInfo...	C:\Windows\System32\apisetschema.dll	SUCCESS	Name: \Windows\System32\apisetschema...
17:25...	1748	QueryNameInfo...	C:\Users[user\Desktop\MALWARE]\Build_Week_Unit_3\Malware_Build_Week_U3.exe	SUCCESS	Name: \Users[user\Desktop\MALWARE]\B...
17:25...	1748	QueryNameInfo...	C:\Windows\System32\wow64win.dll	SUCCESS	Name: \Windows\System32\wow64win.dll
17:25...	1748	QueryNameInfo...	C:\Windows\System32\wow64cpu.dll	SUCCESS	Name: \Windows\System32\wow64cpu.dll
17:25...	1748	QueryNameInfo...	C:\Windows\SysWOW64\cryptbase.dll	SUCCESS	Name: \Windows\SysWOW64\cryptbase.dll
17:25...	1748	QueryNameInfo...	C:\Windows\SysWOW64\sspicli.dll	SUCCESS	Name: \Windows\SysWOW64\sspicli.dll
17:25...	1748	QueryNameInfo...	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Name: \Windows\SysWOW64\sechost.dll
17:25...	1748	QueryNameInfo...	C:\Windows\SysWOW64\advapi32.dll	SUCCESS	Name: \Windows\SysWOW64\advapi32.dll
17:25...	1748	QueryNameInfo...	C:\Windows\SysWOW64\kernel32.dll	SUCCESS	Name: \Windows\SysWOW64\kernel32.dll
17:25...	1748	QueryNameInfo...	C:\Windows\SysWOW64\iprct4.dll	SUCCESS	Name: \Windows\SysWOW64\iprct4.dll
17:25...	1748	QueryNameInfo...	C:\Windows\SysWOW64\KernelBase.dll	SUCCESS	Name: \Windows\SysWOW64\KernelBase...
17:25...	1748	QueryNameInfo...	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Name: \Windows\SysWOW64\msvcrt.dll
17:25...	1748	QueryNameInfo...	C:\Windows\System32\ntdll.dll	SUCCESS	Name: \Windows\System32\ntdll.dll
17:25...	1748	CloseFile	C:\Windows	SUCCESS	
17:25...	1748	CloseFile	C:\Users[user\Desktop\MALWARE]\Build_Week_Unit_3	SUCCESS	

Dopo aver filtrato per attività di file system, dal log di Procmon è possibile vedere che la chiamata interessata è la **CreateFile**, responsabile della creazione del nuovo file **msgina32.dll** dentro la cartella dove risiede il malware.

Analisi finale

Dalle info raccolte durante le due tipologie di analisi, è possibile ipotizzare che il malware cerchi di interferire con la DLL di sistema (**msgina.dll**) che gestisce l'interfaccia utente durante il processo di login di Windows, sostituendola con il suo payload (**msgina32.dll**) per catturare credenziali o eseguire azioni malevole durante l'autenticazione.