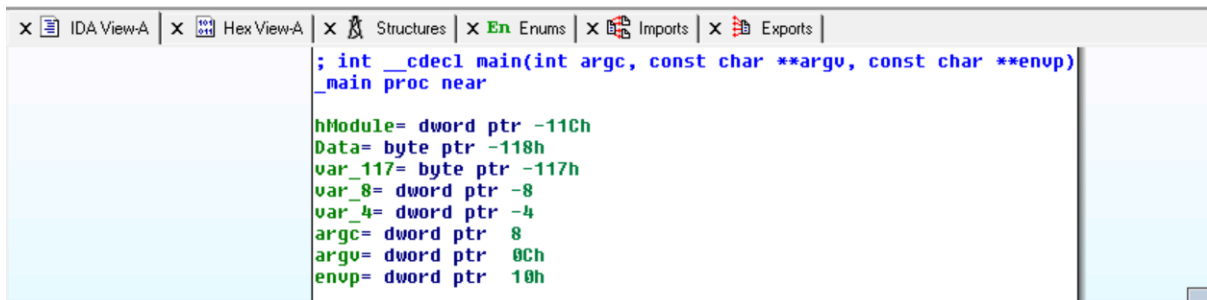


Analisi di un malware

L'esercitazione ci chiedeva di analizzare il funzionamento di un malware e di analizzare il suo comportamento.

Analisi Statica - Analisi del codice con Ida Pro

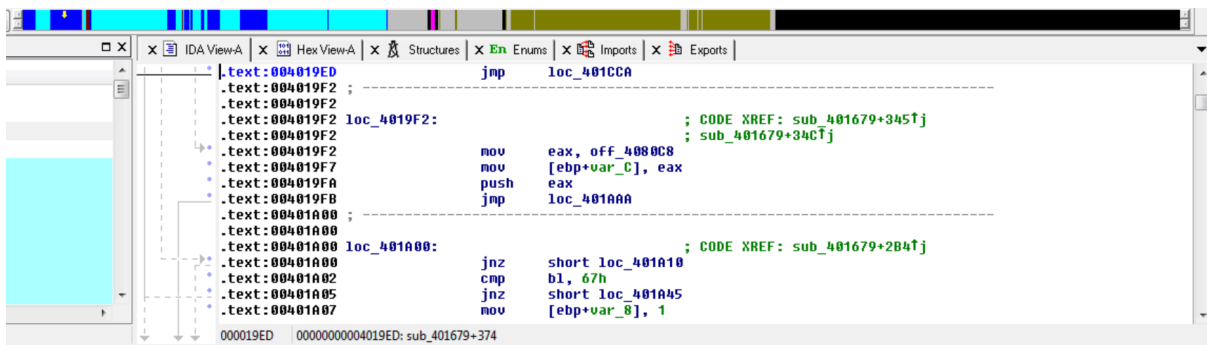
-I primi due punti ci richiedevano di analizzare quanti parametri e quante variabili venivano passate all'interno della funzione Main(), ce ne sono in totale 8 di entrambi come si può vedere nella diapositiva che segue.



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

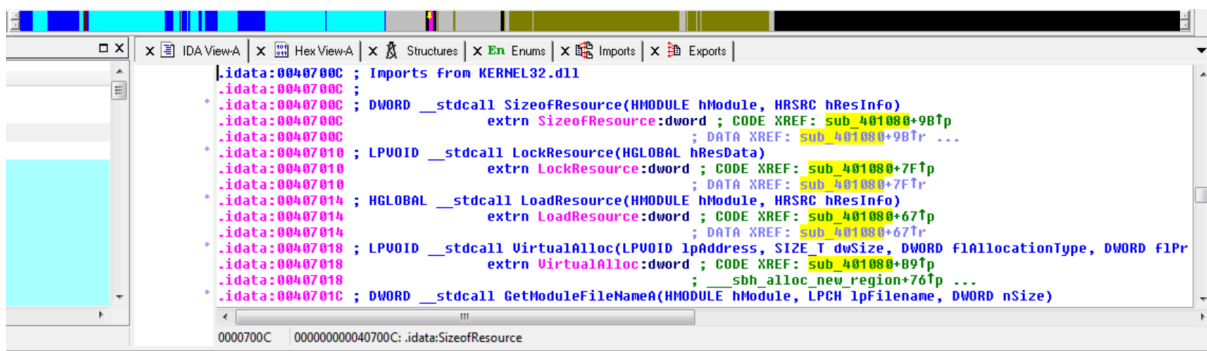
hModule= dword ptr -11Ch
Data= byte ptr -118h
var_117= byte ptr -117h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
```

-Viene poi richiesto di identificare le sezioni all'interno del codice e di descriverle brevemente;



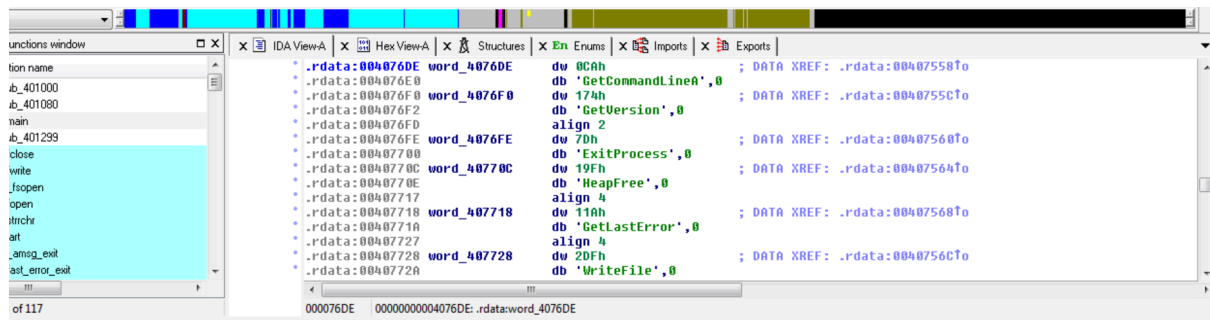
```
.text:004019E0 jmp loc_401CCA
.text:004019F2
.text:004019F2
.text:004019F2
.text:004019F2
loc_4019F2:
mov eax, off_4080C8
mov [ebp+var_C], eax
push eax
jmp loc_401A0A
.text:00401A00
.text:00401A00
loc_401A00:
jnz short loc_401A10
cmp bl, 67h
jnz short loc_401A45
mov [ebp+var_8], 1
; CODE XREF: sub_401679+345J
; sub_401679+34CJ
; CODE XREF: sub_401679+2B4J
```

Prima di tutto troviamo la sezione .Text che contiene le istruzioni eseguibili del programma

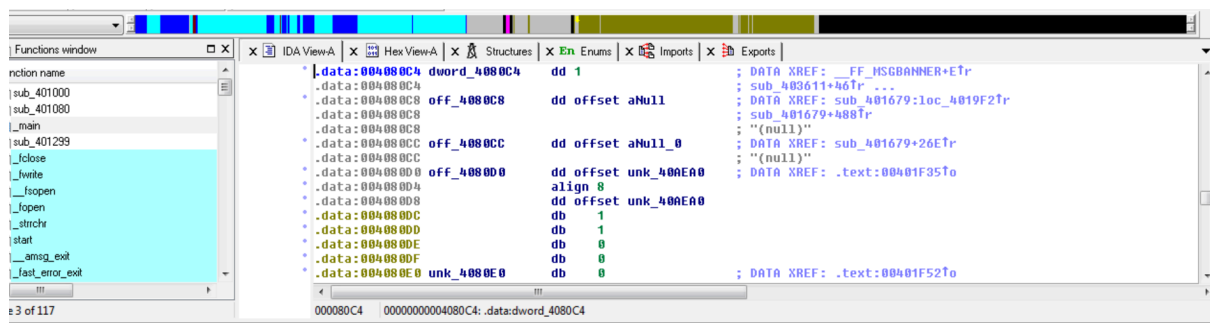


```
.idata:0040700C ; Imports from KERNEL32.dll
.idata:0040700C
.idata:0040700C ; DWORD __stdcall SizeofResource(HMODULE hModule, HRSRC hResInfo)
extrn SizeofResource:dword ; CODE XREF: sub_401080+9B1p
.idata:0040700C ; LPVOID __stdcall LockResource(HGLOBAL hResData)
extrn LockResource:dword ; CODE XREF: sub_401080+7F1p
.idata:00407010 ; HGLOBAL __stdcall LoadResource(HMODULE hModule, HRSRC hResInfo)
extrn LoadResource:dword ; CODE XREF: sub_401080+671p
.idata:00407014 ; LPVOID __stdcall VirtualAlloc(LPVOID lpAddress, SIZE_T dwSize, DWORD flAllocationType, DWORD flProtect)
extrn VirtualAlloc:dword ; CODE XREF: sub_401080+B91p
.idata:00407018 ; DWORD __stdcall GetModuleFileNameA(HMODULE hModule, LPCH lpFilename, DWORD nSize)
```

Troviamo poi la sezione *.idata* che contiene informazioni specifiche per il programma



Segue la sezione *.rdata* che contiene dati di sola lettura o “dati costanti”



Infine la sezione *.data* contiene dati globali e variabili inizializzate

-La prossima richiesta è di riconoscere le librerie che utilizza il malware

Address	Ordinal	Name	Library
00000000...		RegSetValueExA	ADVAPI32
00000000...		RegCreateKeyExA	ADVAPI32

La prima che troviamo è *ADVAPI32* che potrebbe utilizzare per creare e impostare dati di chiavi di registro come visto con le due funzioni che utilizza.

Address	Ordinal	Name	Library
00000000...		GetModuleFileNameA	KERNEL32
00000000...		GetModuleHandleA	KERNEL32
00000000...		FreeResource	KERNEL32
00000000...		FindResourceA	KERNEL32
00000000...		CloseHandle	KERNEL32
00000000...		GetCommandLineA	KERNEL32
00000000...		GetVersion	KERNEL32
00000000...		ExitProcess	KERNEL32
00000000...		HeapFree	KERNEL32
00000000...		GetLastError	KERNEL32
00000000...		WriteFile	KERNEL32
00000000...		TerminateProcess	KERNEL32
00000000...		GetCurrentProcess	KERNEL32

Segue poi la libreria ***KERNEL32*** che potrebbe interagire con le risorse di sistema creando, modificando o eliminando le stesse, oppure cercando informazioni sullo stesso come si può notare da alcune funzioni che utilizza.

-Analisi codice assembly

il prossimo punto richiesto è quello di analizzare alcune righe di codice in assembly del malware

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

La funzione in questa linea di codice ha lo scopo di creare una chiave di registro i cui parametri sono passati con il blocco di codice alla locazione ***loc_401032***.

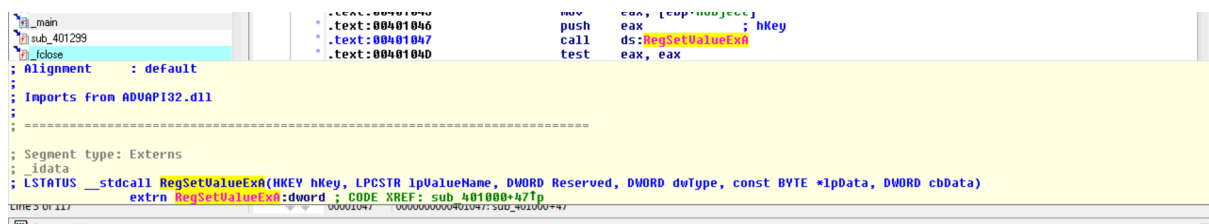
```
.text:00401017          push     offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentUe"...
```

Il parametro in questa linea di codice rappresenta l'oggetto ***SubKey***

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

Queste due righe verificano se la condizione di *Test* è vera (ovvero se è pari a 0) e se si verifica la condizione viene eseguito il blocco di codice alla locazione di memoria ***loc_401032*** che può essere tradotta in linguaggio C come:

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



Qui ci viene richiesto il valore del parametro *ValueName* che è ***GinaDLL***

Analisi dinamica – Analisi del malware con *Procmon*

Come dalla prima richiesta notiamo che una volta eseguito il file eseguibile del malware all'interno della sua cartella si crea una chiave di registro chiamata

GinaDLL

malware_build_week_03	17/01/2024 17:40	Applicazione	32 KB
msgina32.dll	21/04/2024 21:17	Estensione dell'ap...	7 KB

Utilizzando Procmon possiamo notare che è stata creata una chiave di registro

The screenshot shows the Process Monitor (Procmon) window with a list of events. The 'Event Properties' dialog is open, showing details for a 'RegCreateKey' event. The event details include: Date: 21/04/2024 17:29:19, Thread: 2852, Class: Registry, Operation: RegCreateKey, Result: SUCCESS, Path: HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion, Duration: 0.0000250. The 'Desired Access' is 'All Access' and the 'Disposition' is 'REG_OPENED_EXISTING_KEY'.

E che gli è stato assegnato il valore 520

The screenshot shows the Procmon window with a list of events. The 'RegSetValue' event is highlighted, showing details: Type: REG_SZ, Length: 520, Data: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll. A tooltip is visible over the event, showing the details.

Infine come ultimo punto della nostra analisi, sempre utilizzando procmon vediamo che è stato creato un file con la chiamata di sistema **CreateFile**

The screenshot shows the Procmon window with a list of events. The 'CreateFile' event is highlighted, showing details: Name: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll, Result: SUCCESS, Desired Access: Generic Write, Read Attributes. A tooltip is visible over the event, showing the details.

Conclusioni

Possiamo quindi dedurre che il malware in analisi ha la capacità di creare e modificare una chiave di registro del sistema di windows assegnando nome e valore personalizzati