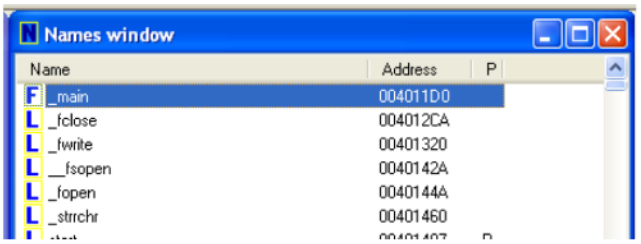
PROGETTO FINALE M6 D8

Analisi completa “Malware_Build_Week_U3”

• ANALISI STATICA DEL MALWARE

Per analisi statica del malware si intendono quell'insieme di tecniche che permettono l'analisi di un file sospetto senza metterlo in esecuzione, oa ogni modo possiamo utilizzare dei tool che ci permettono di esaminare il file per poter iniziare una analisi.

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




```
.text:004011D0
.text:004011D0 ; ╔═══════════════ S U B R O U T I N E ═══════════════════════════════════════
.text:004011D0
.text:004011D0 ; Attributes: bp-based frame
.text:004011D0
.text:004011D0 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:004011D0 _main           proc near               ; CODE XREF: start+AF↓p
.text:004011D0
.text:004011D0 hModule         = dword ptr -11Ch
.text:004011D0 Data            = byte ptr -118h
.text:004011D0 var_8           = dword ptr -8
.text:004011D0 var_4           = dword ptr -4
.text:004011D0 argc            = dword ptr  8
.text:004011D0 argv            = dword ptr  0Ch
.text:004011D0 envp            = dword ptr  10h
```

I suoi parametri sono i seguenti:

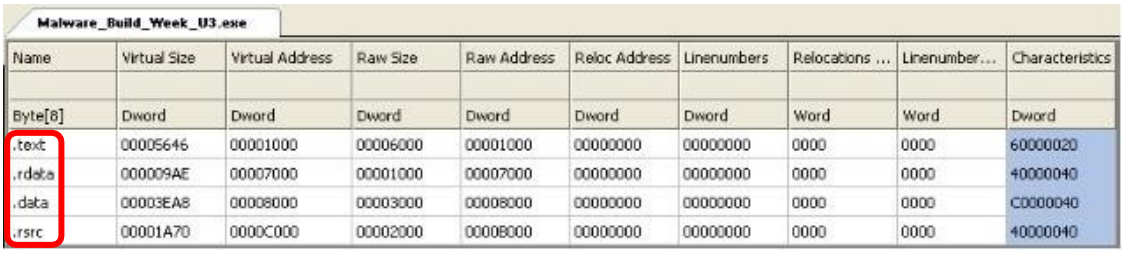
```
.text:004011D0 argc            = dword ptr  8
.text:004011D0 argv            = dword ptr  0Ch
.text:004011D0 envp            = dword ptr  10h
```

2 - Variabili dichiarate all’interno della funzione Main()

```
.text:004011D0 hModule         = dword ptr -11Ch
.text:004011D0 Data            = byte ptr -118h
.text:004011D0 var_8           = dword ptr -8
.text:004011D0 var_4           = dword ptr -4
```

3- Quali sezioni sono presenti all'interno del file eseguibile

Le sezioni presenti sono nel riquadro in rosso (queste informazioni si possono ottenere tramite il tool CFF Explorer



Malware_Build_Week_U3.exe

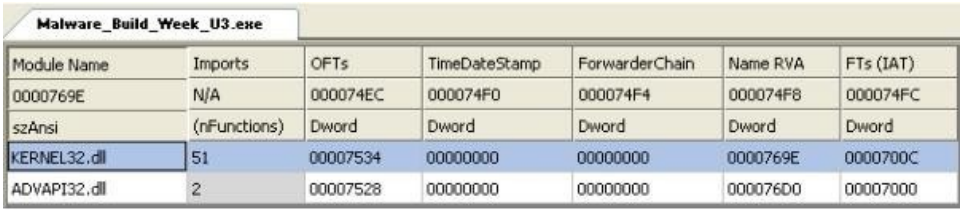
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00005646	00001000	00006000	00001000	00000000	00000000	0000	0000	60000020
.rdata	000009AE	00007000	00001000	00007000	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

- **.text**: questa sezione contiene il codice che la CPU esegue una volta che il software viene avviato;
- **.rdata**: include informazioni su librerie e funzioni importate ed esportate dal PE;
- **.data**: in questa sezione troviamo normalmente le variabili e i dati dell'eseguibile che devono essere disponibili;
- **.rsrc**: comprende le risorse utilizzate dal Malware come immagini, icone, stringhe e menù che non sono parte dell'eseguibile stesso.

4 - Quali libreria vengono importate

Il malware importa due librerie, KERNEL32.dll e ADVAPI32.dll.

KERNEL32.dll è una libreria che contiene le funzioni principali per comunicare con il sistema operativo, quali la gestione della memoria e la modifica dei file. In questo caso contiene al suo interno le seguenti funzioni:



Malware_Build_Week_U3.exe

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
0000769E	N/A	000074EC	000074F0	000074F4	000074F8	000074FC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

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
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
00007806	00007806	0152	GetStdHandle
00007816	00007816	0115	GetFileType
00007824	00007824	0150	GetStartupInfoA
00007836	00007836	0109	GetEnvironmentVariableA
00007850	00007850	0175	GetVersionExA
00007860	00007860	019D	HeapDestroy
0000786E	0000786E	019B	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

Il Malware sembra essere un **dropper** un eseguibile malevolo nascosto all'interno di un file formalmente innocuo; tutto ciò si potrebbe dedurre dal fatto che utilizza funzioni tipo **FindResourceA**, per trovare il l'eseguibile malevolo contenuto nella sezione risorse (.rsrc) utilizzando come parametri il nome e il tipo della risorsa; **LoadResource**, recupera un handle che può essere usato per ottenere un puntatore al primo byte della risorsa specificata in memoria; **LockResource**, per recuperare un puntatore alla risorsa specificata; **SizeofResource**, per identificare la dimensione della risorsa.
Altre funzioni come **CreateFileA/WriteFile** ci fanno capire dei tentavi di apertura/creazione di file e scrittura su altri file.

La libreria **ADVAPI32.dll** contiene, invece, tutte le funzioni che permettono di interagire con i registri e i servizi del sistema operativo. Contiene due funzioni al suo interno:

Malware_Build_Week_U3.exe

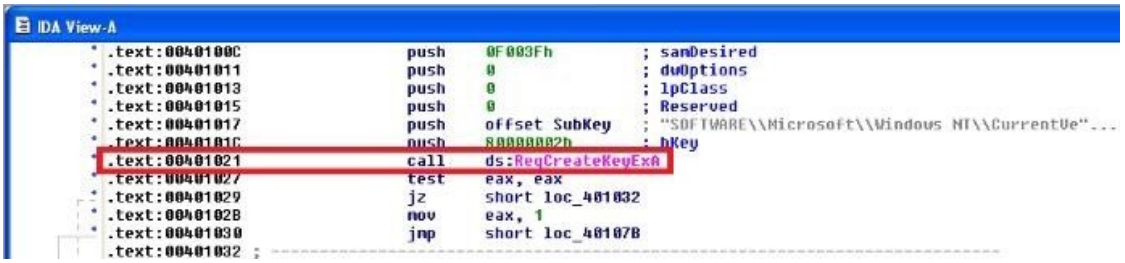
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000076D0	N/A	00007500	00007504	00007508	0000750C	00007510
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

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

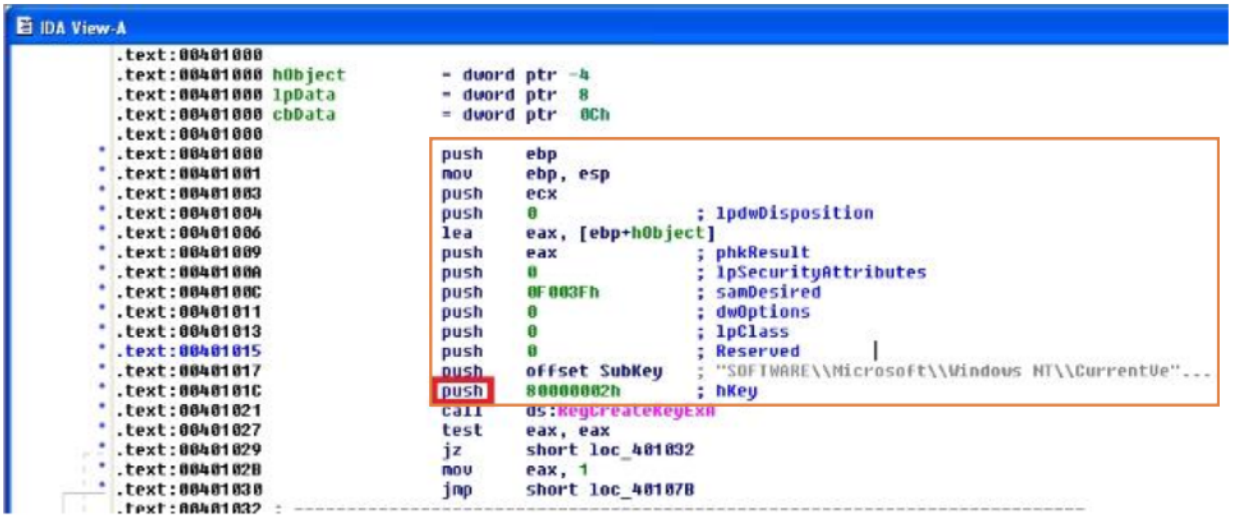
Il Malware richiama solo due funzioni tramite questa libreria: **RegCreateKeyExA**, per creare una chiave di registro o aprirla se è già esistente, e **RegSetValueExA**, per impostarne i valori della chiave.

• ANALISI DI BASSO LIVELLO DI ALCUNE ISTRUZIONI IN LINGUAGGIO ASSEMBLY CON TRADUZIONE IN “C” DELLE ISTRUZIONI TRA GLI INDIRIZZI 00401027 E 00401029

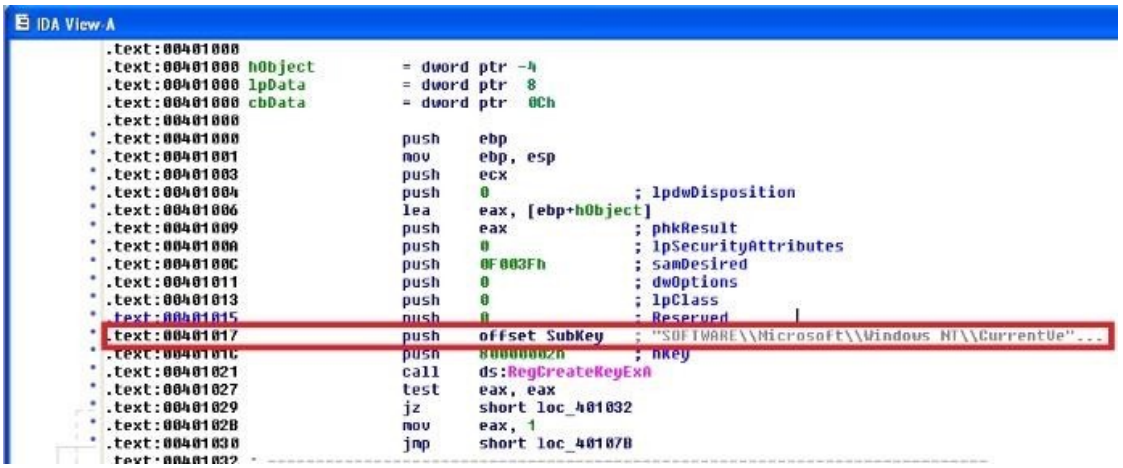
a) All’indirizzo di memoria **00401021**, si nota la chiamata di funzione **RegCreateKeyExA**, ovvero il Malware sta tentando di creare o aprire una chiave di registro:



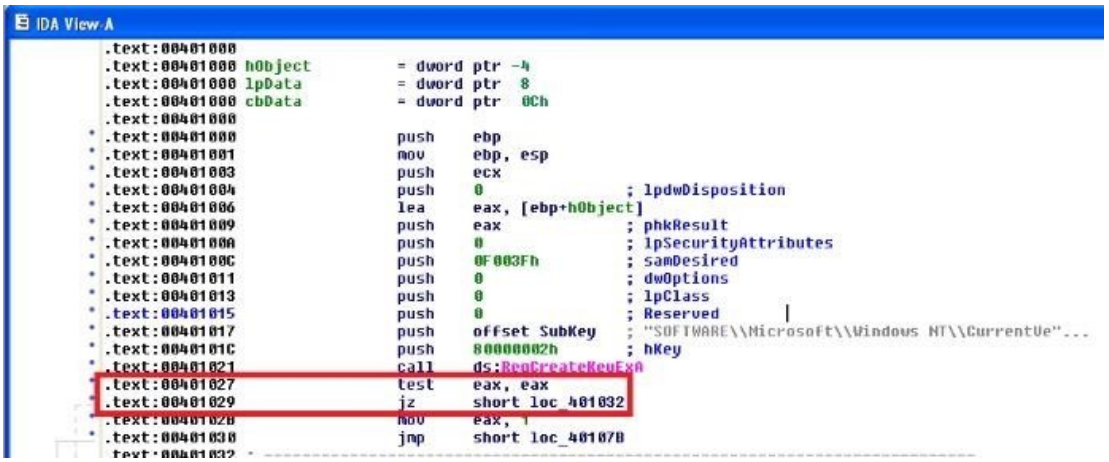
b) I parametri vengono passati tramite l'istruzione **push, che** appunto inserisce un valore sullo stack della funzione chiamata:



c) Il parametro alla locazione **00401017** rappresenta il nome della SubKey che questa funzione apre o crea. Qua sotto si nota ciò che viene “pushato” èm la directory “SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon”



d) Il significato delle istruzioni tra 00401027 e 00401029 - Le istruzioni test e jz controllano se il registro eax sia uguale a zero utilizzando l'istruzione "**test eax, eax**". Successivamente, viene eseguito un salto condizionale ("**jz**" che significa "jump if zero") alla locazione **00401032** se il flag ZF **=** 1. In sintesi, queste istruzioni controllano se il registro **eax** contiene il valore zero e, in caso positivo, eseguono un salto a un'altra locazione del codice (**loc_401032**), altrimenti continua con il normale flusso del programma:



e) Con riferimento al quesito precedente, riportiamo la rappresentazione del codice **Assembly** nel corrispondente costrutto **C**:

```
if (eax == 0)
   {funct_401032();}
else
   {
   eax = 1;
   funct_40107B();
}
```

e) Valutazione chiamata alla locazione **00401047 -** (utilizzo citazioni da microsoft.com link https://learn.microsoft.com/it-it/windows/win32/secauthn/gina) il valore del parametro «**ValueName**» che troviamo è "**GinaDLL**", una libreria dinamica **Win32** che opera nel contesto del processo **Winlogon** e che viene caricata molto presto nel processo di avvio. Winlogon è un componente Windows responsabile della gestione della SAS (Secure Attention Sequence), caricando il profilo utente al momento del login, e possibilmente bloccando il PC durante l'esecuzione dello screensaver. Invece , lo scopo di una DLL GINA è fornire procedure personalizzabili di identificazione e autenticazione dell'utente.