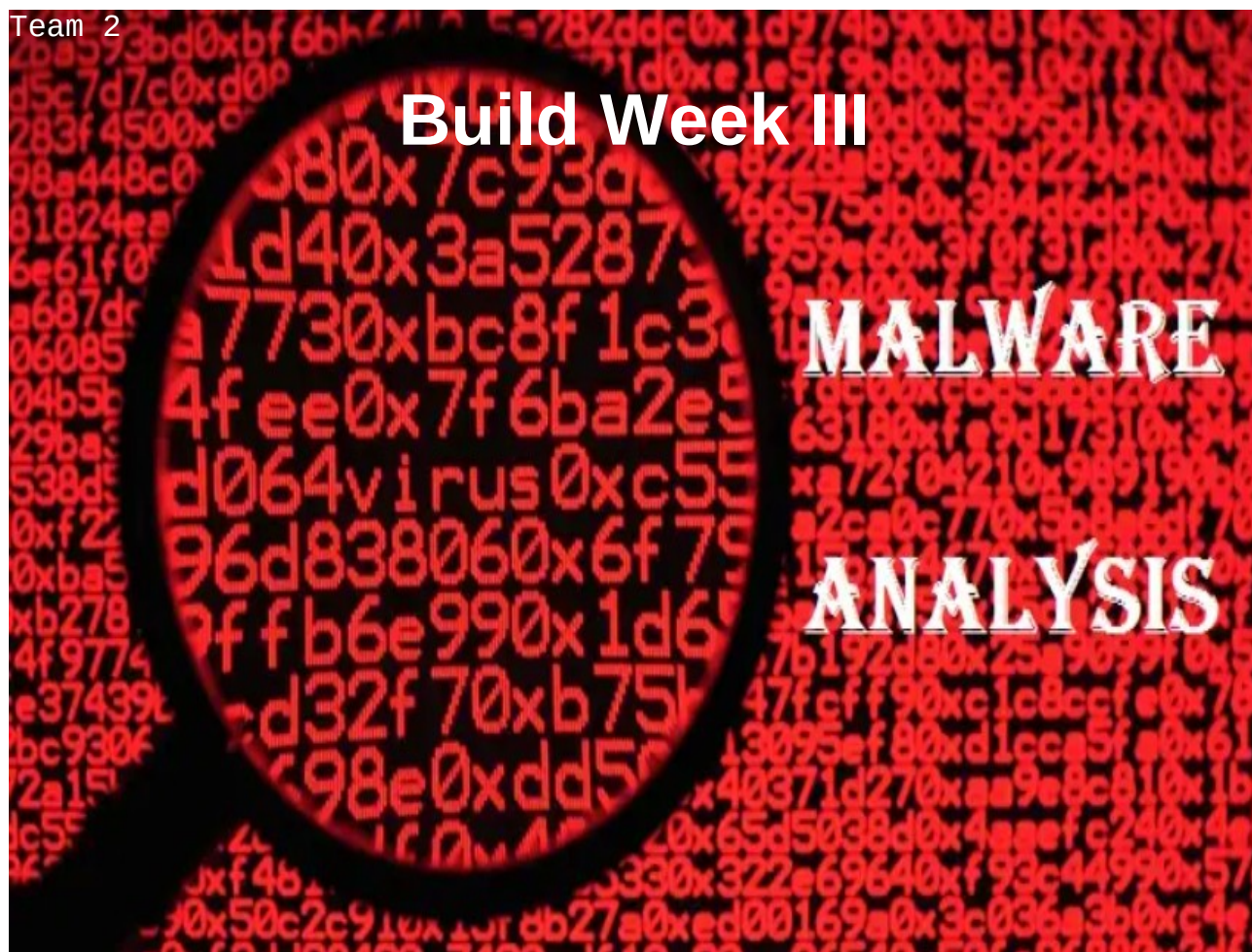


Team 2



Eseguibile: Build_Week_U3

Tools: CFF Explorer, IDA Pro Free, OllyDBG, Process Monitor

OS: Windows XP 32-bit

Team 2:

Fabio De Rosa

Letizia Lo Iacono

Claudio De Cicco

Filip Stojimirovic

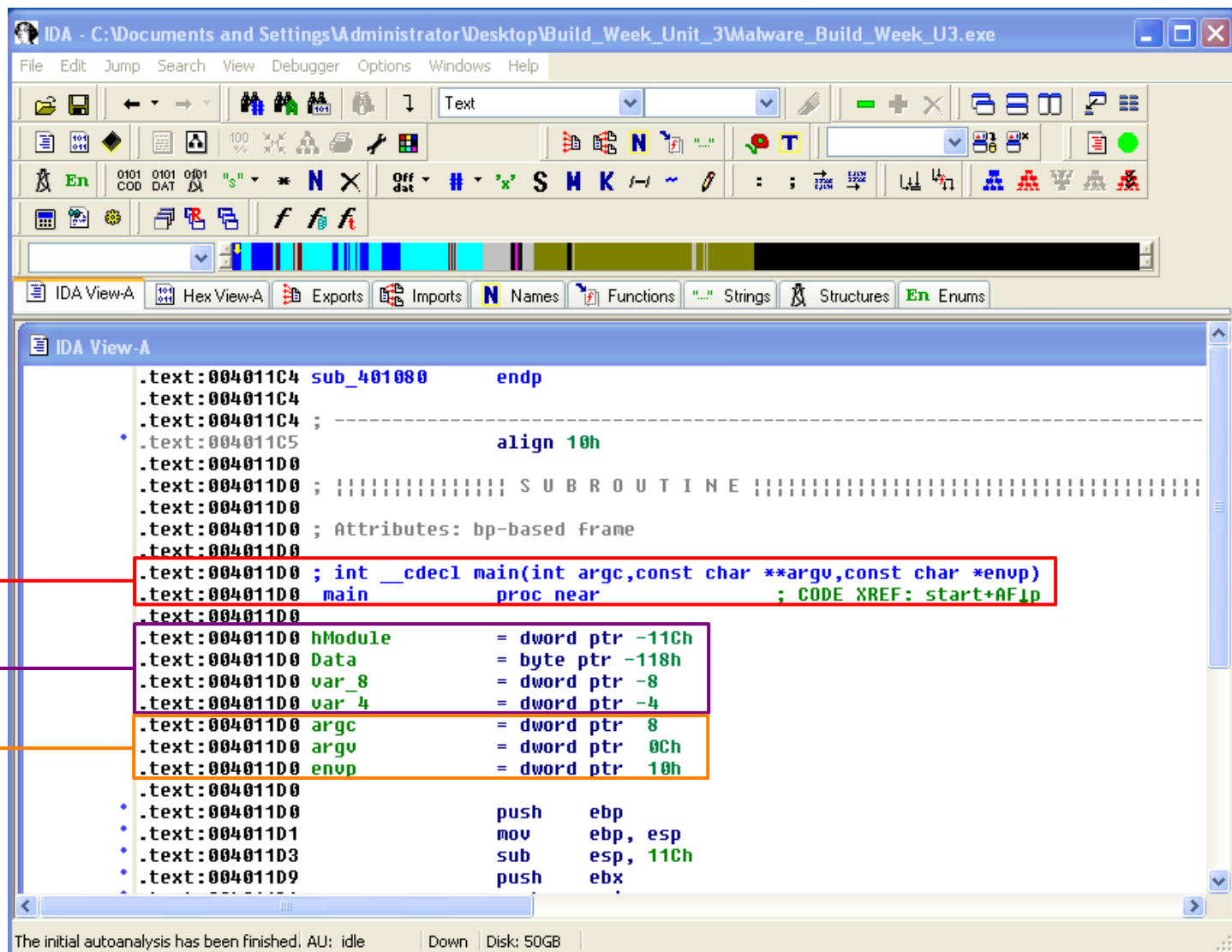
Alessandro Alaimo

Armando Battaglini

Riccardo Mascheroni

[Giorno 1]

Per iniziare apriremo l'eseguibile Build_Week_U3, trovandolo sul Desktop dentro la cartella Build_Week_Unit_3, con il disassembler IDA Pro Free.



Funzione main()

```
.text:004011D0 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:004011D0 _main proc near ; CODE XREF: start+AF1p
```

1.1 Quanti parametri sono passati alla funzione Main()?

Parametri: 3 (offset **positivo**)

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

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

Variabili: 4 (offset **negativo**)

```
.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
```

1.3 Quali sezioni sono presenti all'interno del file eseguibile? Descrivere brevemente almeno due di quelle identificate

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

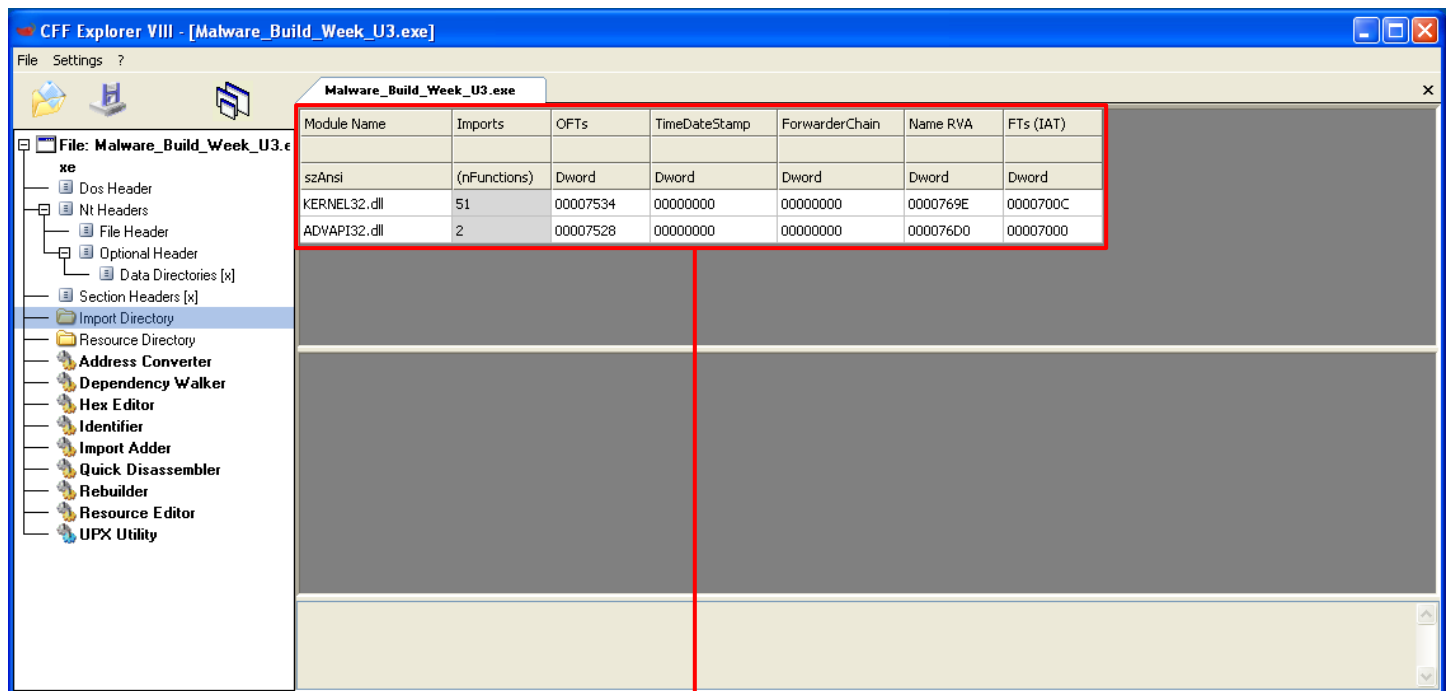
4 sezioni presenti nel eseguibile:

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	Istruzioni da eseguire per la CPU al avvio del programma, spesso read-only per prevenire al programma di modificare accidentalmente le istruzioni.
.rdata	Read-only initialized data, include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile
.data	Initialized data (free format) include tipicamente i dati / le variabili globali del programma eseguibile
.rsrc	Include le risorse utilizzate dall'eseguibile come ad esempio icone, immagini, menù e stringhe che non sono parte dell'eseguibile stesso.

Tabella n.1

1.4 Quali librerie importa il Malware? Per ognuna delle librerie importate, fare delle ipotesi sulla base della sola analisi statica delle funzionalità che il malware potrebbe implementare. Utilizzare le funzioni che sono richiamate all'interno delle librerie per supportare le ipotesi.



librerie importate:

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

Le funzioni richiamate

KERNEL32.dll

ADVAPI32.dll

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
Malware_Build_Week_U3.exe	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	LoadResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02B8	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	GetCPIInfo
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

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
Malware_Build_Week_U3.exe	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

KERNEL32.dll	
SizeofResource	Controlla e recupera la dimensione di una determinata risorsa. Di solito si trova nei droppers
LockResource	Viene utilizzato con FindResource() , LoadResource() e SizeOfResource() di solito per lavorare con eseguibili incorporati nella sezione .rsrc (droppers)
LoadResource	Viene utilizzato per caricare una risorsa da un file PE in memoria. Il malware a volte utilizza le risorse per archiviare stringhe, informazioni di configurazione o altri file dannosi.
FindResourceA	Viene utilizzato per trovare una risorsa in una DLL eseguibile o caricata. Il malware a volte utilizza le risorse per archiviare stringhe, informazioni di configurazione o altri file dannosi.
CreateFileA	Viene utilizzato per creare un nuovo file o per aprire un file esistente.
GetProcAddress	Viene utilizzato per ottenere l'indirizzo di memoria di una funzione in una DLL. Questo viene spesso utilizzato dal malware per scopi di offuscamento ed evasione per evitare di dover chiamare direttamente la funzione.
LoadLibraryA	Viene utilizzato per caricare un modulo specificato nello spazio degli indirizzi del processo chiamante. Il malware lo usa comunemente per caricare dinamicamente le DLL a scopo di evasione.

Tabella n.2

ADVAPI32.dll	
RegSetValueExA	Viene utilizzato per impostare un valore e un tipo per una determinata chiave di registro.
RegCreateKeyExA	Viene utilizzato per creare una chiave di registro specificata. Se la chiave esiste già, la funzione la apre.

Tabella n. 3

Si può ipotizzare che questo file eseguibile sia un Malware di tipo dropper, poiché con la libreria **KERNEL32.dll** include funzioni che localizzano il Malware in questione per estrarlo e caricarlo in memoria per l'esecuzione salvandolo su disco per eseguirlo in un secondo momento.

Inoltre otterrà persistenza con l'uso della libreria **ADVAPI32.dll** poiché le funzioni usate andranno a creare o aprire una chiave di registro cambiandone il valore.

[Giorno 2]

2.1 Lo scopo della funzione chiamata alla locazione di memoria 00401021

```
• .text:00401015      push    0                ; Reserved
• .text:00401017      push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
• .text:0040101C      push    80000002h        ; hKey
• .text:00401021      call     ds:RegCreateKeyExA
```

Alla locazione di memoria 00401021 troviamo la funzione **RegCreateKeyExA**

È una funzione delle API di Windows che consente di creare una nuova chiave del Registro di sistema o per aprire una chiave esistente per la modifica. La funzione accetta come argomento un handle alla chiave di base (**hKey**), il nome della nuova chiave da creare, eventuali opzioni e informazioni di sicurezza. Restituisce un handle alla nuova chiave se la creazione ha il successivo, altrimenti restituisce un codice di errore.

Handle = Puntatore a qualsiasi oggetto

2.2 Come vengono passati I parametri alla funzione alla locazione 00401021

```
• .text:00401009      push    eax                ; phkResult
• .text:0040100A      push    0                ; lpSecurityAttributes
• .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\\Microsoft\\Windows NT\\CurrentVe"...
• .text:0040101C      push    80000002h        ; hKey
• .text:00401021      call     ds:RegCreateKeyExA
```

I parametri sono passati sullo stack tramite la chiamata **push**.

Possiamo ipotizzare che il puntatore ad una variabile **phkResult** riceva l'handle della chiave aperta, e che sia una chiave di registro predefinita, poiché non viene richiamata la funzione **RegCloseKey** dopo l'utilizzo dell'handle.

2.3 Che oggetto rappresenta il parametro alla locazione 00401017

```
• .data:00408054 SubKey      db 'SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon',0
• .data:00408054                                     ; DATA XREF: sub_401000+17↑o
• .data:0040808A      align 4
```

L'oggetto rappresenta la chiave registro che il malware utilizza per ottenere **persistenza** su sistema target.

In particolare si avvale del processo **Winlogon** che implementa la funzione di Windows di gestione dei login, è inoltre responsabile del caricamento dei profili utente dopo il login, del supporto del login automatico.

2.4 Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029

Questo codice assembly x86 esegue un controllo di uguaglianza tra il valore contenuto nelle registrazioni **EAX** e **EAX**.

.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]	

La istruzione "test" esegue una operazione bit a bit "and" tra i due valori e imposta i flag del processore in base al risultato. L'istruzione "jz" (**jump if zero**) esegue un salto condizionale all'indirizzo specificato "loc_401032" solo se lo **Zero Flag** è impostato a 1, il che significa che i due valori sono uguali.

Quindi , questo codice sta confrontando il valore contenuto nella registrazione EAX con quello contenuto in EAX. Se i due valori sono uguali, il codice esegue un salto all'indirizzo "loc_401032", altrimenti continua eseguendo il codice successivo.

```
hObject= dword ptr -4
lpData= dword ptr 8
cbData= dword ptr 0Ch

push    ebp
mov     ebp, esp
push    ecx
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 NT\\CurrentVe"...
push    80000002h     ; hKey
call    ds:RegCreateKeyExA
test    eax, eax
jz      short loc_401032
```

loc_401032:

```
mov     ecx, [ebp+cbData]
push    ecx           ; cbData
mov     edx, [ebp+lpData]
push    edx           ; lpData
push    1             ; dwType
push    0             ; Reserved
push    offset ValueName ; "GinaDLL"
mov     eax, [ebp+hObject]
push    eax           ; hKey
call    ds:RegSetValueExA
test    eax, eax
jz      short loc_401062
```

Grazie all'interfaccia grafica di Ida possiamo vedere come il salto venga effettuato alla locazione di memoria 401032.

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

```
if (Eax == Eax) {  
    //jump to loc_401032  
}
```

2.6 Valutare la chiamata alla locazione **00401047**, qual è il valore del parametro “ValueName”?

```
* .text:0040103E      push    offset ValueName ; "GinaDLL"  
* .text:00401043      mov     eax, [ebp+hObject]  
* .text:00401046      push    eax                ; hKey  
* .text:00401047      call   ds:RegSetValueExA
```

Alla locazione **00401047** troviamo la funzione **RegSetValueExA**.

Il malware utilizza questa funzione per ottenere la **persistenza** e far sì che il sistema operativo lo avvii nelle fasi iniziali di start-up.

Questa funzione permette di aggiungere un nuovo valore all'interno del registro e di settare i rispettivi dati.

Il valore del parametro “ValueName” è **GinaDll**.

[Giorno 3]

Riprendendo l'analisi del codice, analizzando le routine tra le locazioni di memoria **00401080** e **00401128**, rispondere alle prossime domande:

3.1 Qual è il valore del parametro “ResourceName” passato alla funzione FindResourceA()

```
.text:00401088 ; -----  
.text:00401088  
.text:00401088 loc_401088:      ; CODE XREF: sub_401080+2F↑j  
* .text:00401088      mov     eax, lpType  
* .text:004010BD      push    eax                ; lpType  
* .text:004010BE      mov     ecx, lpName  
* .text:004010C4      push    ecx                ; lpName  
* .text:004010C5      mov     edx, [ebp+hModule]  
* .text:004010C8      push    edx                ; hModule  
* .text:004010C9      call   ds:FindResourceA  
* .text:004010CF      mov     [ebp+hResInfo], eax  
* .text:004010D2      cmp     [ebp+hResInfo], 0  
* .text:004010D6      jnz     short loc_4010DF  
* .text:004010D8      xor     eax, eax  
* .text:004010DA      jmp     loc_4011BF  
.text:004010DF ; -----
```


Usando IDA possiamo vedere che la funzione FindResourceA() va a cercare la risorsa nel parametro di nome **lpName**

```
* .data:00408034 ; LPCSTR lpName
.data:00408034 lpName          dd offset aTgad          ; DATA XREF: sub_401080+3E1r
.data:00408034 aTgad          db 'TGAD',0          ; "TGAD"
.data:00408038 aTgad          align 10h          ; DATA XREF: .data:lpName10
.data:0040803D aBinary       db 'BINARY',0          ; DATA XREF: .data:lpType10
.data:00408040 aBinary       align 4
.data:00408048 aRi          db 'RI',0Ah,0          ; DATA XREF: sub_401080:loc_40106210
.data:0040804C ; char ValueName[]
.data:0040804C ValueName     db 'GinaDLL',0          ; DATA XREF: sub_401080+3E10
.data:00408054 ; char SubKey[]
```

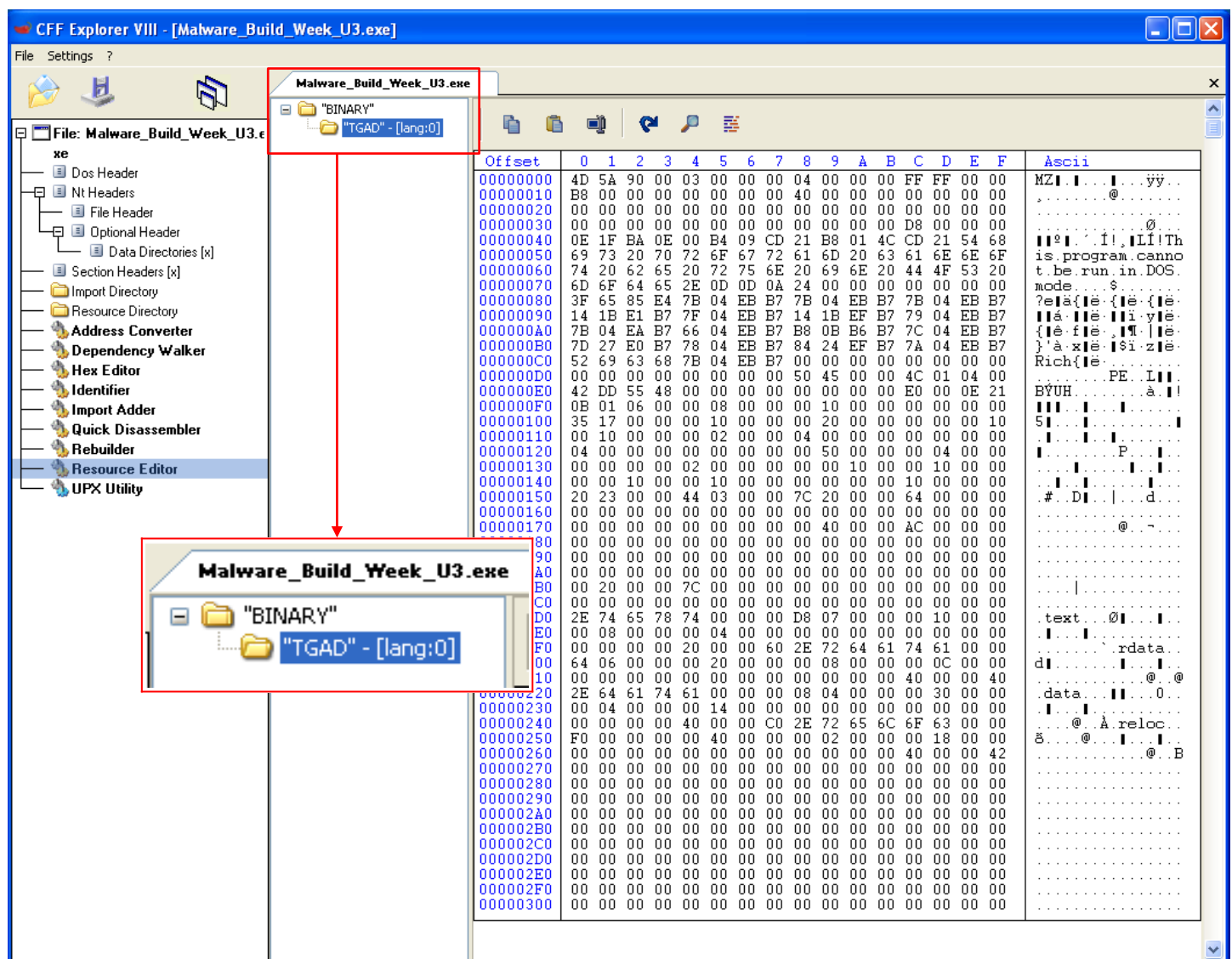
Seguendo il detto parametro scopriamo che il parametro è stato riempito con "TGAD"

Per la conferma possiamo analizzare l'eseguibile con OllyDBG

0040107F	CC	INT3	
00401080	55	PUSH EBP	
00401081	8BEC	MOV EBP,ESP	
00401083	83EC 18	SUB ESP,18	
00401086	56	PUSH ESI	
00401087	57	PUSH EDI	
00401088	C745 EC 00000000	MOV DWORD PTR SS:[EBP-14],0	
0040108F	C745 E8 00000000	MOV DWORD PTR SS:[EBP-18],0	
00401096	C745 F8 00000000	MOV DWORD PTR SS:[EBP-8],0	
0040109D	C745 F0 00000000	MOV DWORD PTR SS:[EBP-10],0	
004010A4	C745 F4 00000000	MOV DWORD PTR SS:[EBP-C],0	
004010AB	837D 08 00	CMP DWORD PTR SS:[EBP+8],0	
004010AF	75 07	JNZ SHORT Malware_.004010B8	
004010B1	33C0	XOR EAX,EAX	
004010B3	E9 07010000	JMP Malware_.004011BF	
004010B8	A1 30804000	MOV EAX,DWORD PTR DS:[408030]	
004010BD	50	PUSH EAX	
004010BE	8B0D 34804000	MOV ECX,DWORD PTR DS:[408034]	
004010C4	51	PUSH ECX	
004010C5	8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
004010C8	52	PUSH EDX	
004010C9	FF15 28704000	CALL DWORD PTR DS:[<&KERNEL32.FindResourceA>]	ResourceType => "BINARY" Malware_.00408038 ResourceName => "TGAD"
004010CF	8945 EC	MOV DWORD PTR SS:[EBP-14],EAX	hModule FindResourceA
004010D2	837D EC 00	CMP DWORD PTR SS:[EBP-14],0	
004010D6	75 07	JNZ SHORT Malware_.004010DF	
004010D8	33C0	XOR EAX,EAX	
004010DA	E9 E0000000	JMP Malware_.004011BF	
004010DF	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]	
004010E2	50	PUSH EAX	
004010E3	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
004010E6	51	PUSH ECX	
004010E7	FF15 14704000	CALL DWORD PTR DS:[<&KERNEL32.LoadResource>]	hResource hModule LoadResource
004010ED	8945 E8	MOV DWORD PTR SS:[EBP-18],EAX	
004010F0	837D E8 00	CMP DWORD PTR SS:[EBP-18],0	
004010F4	75 05	JNZ SHORT Malware_.004010FB	
004010F6	E9 A0000000	JMP Malware_.004011A5	
004010FB	8B55 E8	MOV EDX,DWORD PTR SS:[EBP-18]	
004010FE	52	PUSH EDX	
004010FF	FF15 10704000	CALL DWORD PTR DS:[<&KERNEL32.LockResource>]	nHandles SetHandleCount
00401105	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX	
00401108	837D F8 00	CMP DWORD PTR SS:[EBP-8],0	
0040110C	75 05	JNZ SHORT Malware_.00401113	
0040110E	E9 92000000	JMP Malware_.004011A5	
00401113	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]	
00401116	50	PUSH EAX	
00401117	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
0040111A	51	PUSH ECX	
0040111B	FF15 0C704000	CALL DWORD PTR DS:[<&KERNEL32.SizeofResource>]	hResource hModule SizeofResource
00401121	8945 F0	MOV DWORD PTR SS:[EBP-10],EAX	
00401124	837D F0 00	CMP DWORD PTR SS:[EBP-10],0	
00401128	77 02	JR SHORT Malware_.0040112C	
0040112A	EB 79	JMP SHORT Malware_.004011A5	

dove abbiamo lo stesso risultato.

Riguardo il file appena trovato, non abbiamo molte informazioni su di esso, ma possiamo vedere se è presente all'interno del malware stesso usando CFF Explorer.



3.2 Quali funzionalità implementa il malware?

Andando ad analizzare qualche riga seguente all'indirizzo **00104080**, possiamo vedere come nella stessa routine siano invocate le funzioni: **LoadResource**, **LockResource** e **SizeofResource**

```
.text:004010DF
.text:004010DF loc_4010DF:      ; CODE XREF: sub_401080+56tj
.text:004010DF      mov     eax, [ebp+hResInfo]
.text:004010E2      push    eax                ; hResInfo
.text:004010E3      mov     ecx, [ebp+hModule]
.text:004010E6      push    ecx                ; hModule
.text:004010E7      call    ds:LoadResource
.text:004010ED      mov     [ebp+hResData], eax
.text:004010F0      cmp     [ebp+hResData], 0
.text:004010F4      jnz     short loc_4010FB
.text:004010F6      jmp     loc_4011A5
;
.text:004010FB loc_4010FB:      ; CODE XREF: sub_401080+74tj
.text:004010FB      mov     edx, [ebp+hResData]
.text:004010FE      push    edx                ; hResData
.text:004010FF      call    ds:LockResource
.text:00401105      mov     [ebp+var_8], eax
.text:00401108      cmp     [ebp+var_8], 0
.text:0040110C      jnz     short loc_401113
.text:0040110E      jmp     loc_4011A5
;
.text:00401113 loc_401113:      ; CODE XREF: sub_401080+8Ctj
.text:00401113      mov     eax, [ebp+hResInfo]
.text:00401116      push    eax                ; hResInfo
.text:00401117      mov     ecx, [ebp+hModule]
.text:0040111A      push    ecx                ; hModule
.text:0040111B      call    ds:SizeofResource
.text:00401121      mov     [ebp+dwSize], eax
.text:00401124      cmp     [ebp+dwSize], 0
.text:00401128      ja      short loc_40112C
.text:0040112A      jmp     short loc_4011A5
;
.text:0040112C
```

3.3 È possibile identificare questa funzionalità tramite l'analisi statica basica?

Come abbiamo visto in domanda 1.4 e la tabella n.2 la risposta è sì.

3.4 Elencare le evidenze a supporto.

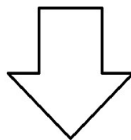
Module Name	Imports	OFs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
0000769E	N/A	000074EC	000074F0	000074F4	000074F8	000074FC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

OFs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	028B	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

OFs	FTs (IAT)	Hint	Name
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource

3.5 Diagramma di flusso

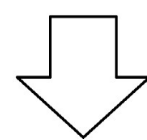
FUNZIONE: "_MAIN"



SUB_401080:
CONTIENE LE FUNZIONI:
FINDRESOURCE, LOADRESOURCE & SIZEOFRESOURCE

TALE ROUTINE IMPLEMENTA LA FUNZIONE DI DROPPER DEL MALWARE, INFATTI UTILIZZA LE FUNZIONI SOPRA CITATE (E ALTRE) PER COPIARE E SCRIVERE ALTRI FILE ALL'INTERNO DELLA MACCHINA ATTACATA.

CERCA E UTILIZZA IL FILE "TGAD.BIN"(RISORSA INTERNA), MODIFICA IL FILE DI REGISTRO "NSGINA32.DLL"



SUB_401000:
CONTIENE LE FUNZIONI:
REGCREATEKEYEX, REGSETVALUEEX

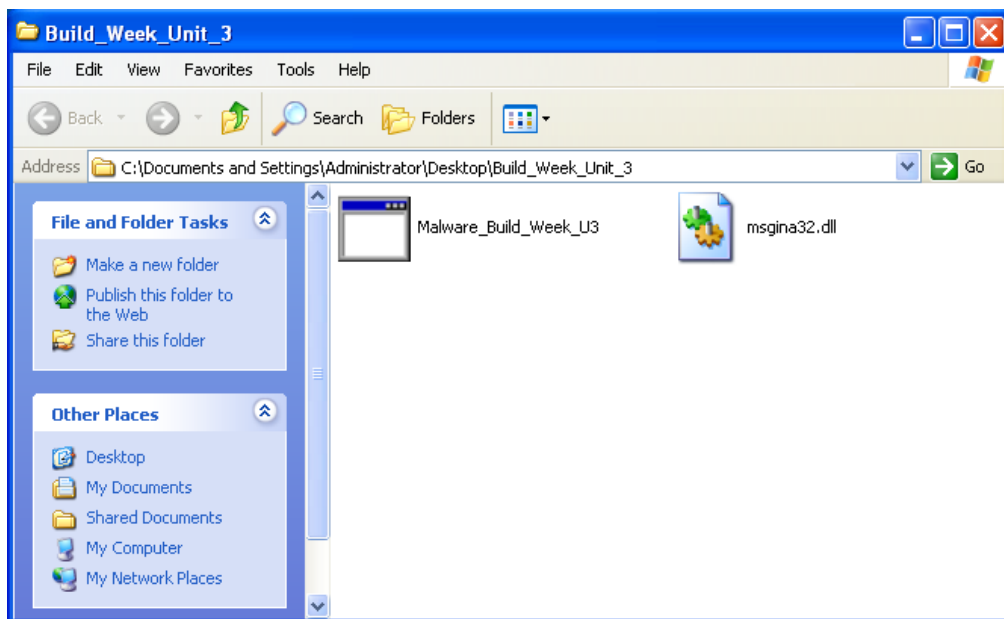
TALE ROUTINE IMPLEMENTA LA FUNZIONE DI PERSISTENZA DEL MALWARE, INFATTI UTILIZZA LE FUNZIONI SOPRA CITATE (E ALTRE) PER MODIFICARE LE CHIAVI DI REGISTRO.

INSTALLA "GINADLL" NELLA CARTELLA DI "WINLOGON".

[Giorno 4]

4.1 Cosa avviene dentro la cartella dove è situato l'eseguibile del malware?

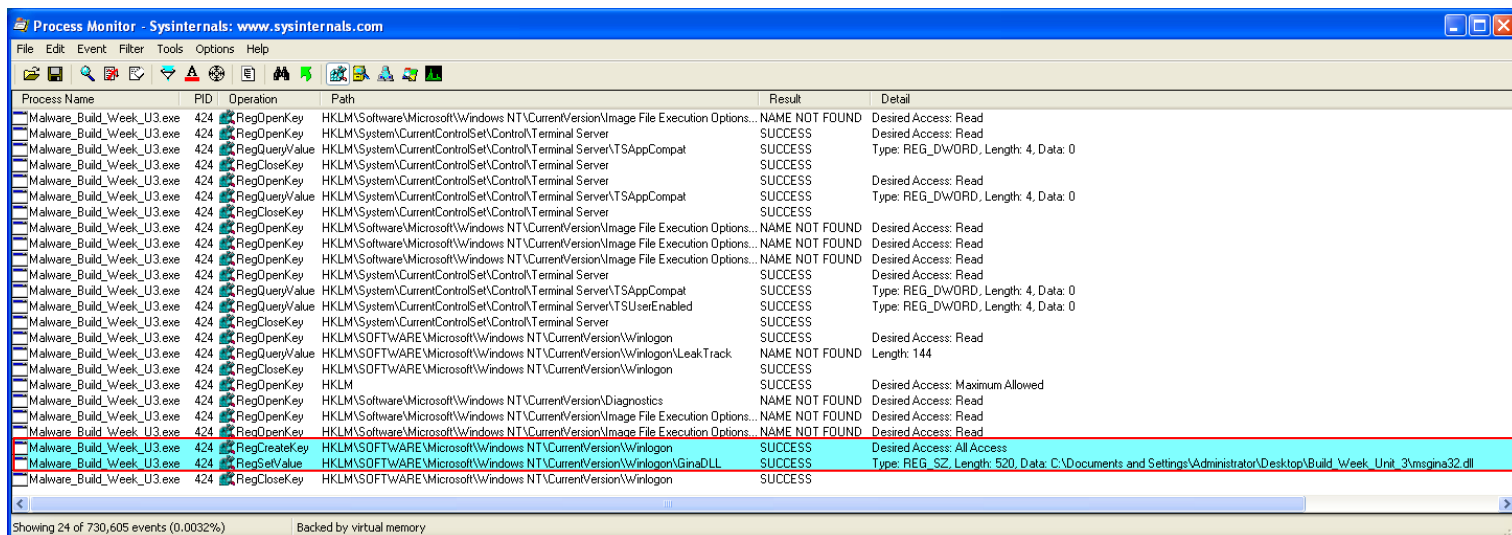
Dentro la cartella Build_Week_Unit_3 dove si trova l'eseguibile del malware, viene creato un nuovo file di nome msgina32.dll



Questo accade al momento della sua esecuzione, poichè il codice esegue le istruzioni del PE al suo interno, cercando ed estraendo una dll contenente le vere istruzioni (e funzioni) necessarie per portare a compimento il suo intento malevolo, compiendo un “injecting” di una libreria malevola tra “winlogon.exe msgina32.dll per intercettare le credenziali di accesso.

4.2 Quale chiave di registro viene creata?

Una volta filtrati avremmo una visione più chiara dei cambiamenti avvenuti, possiamo notare come sia stata creata **una chiave di registro** e il caricamento di una **libreria: GinaDll.dll**



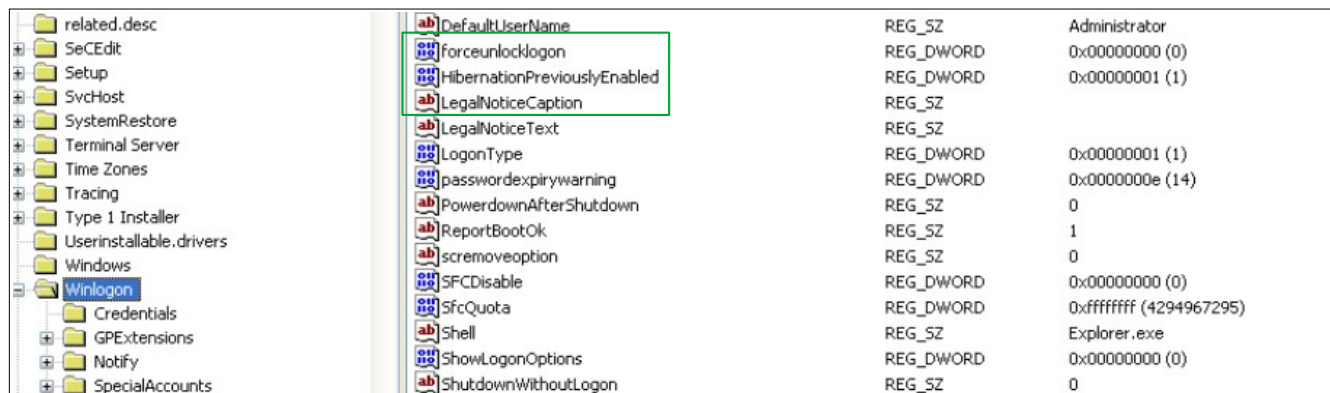
Il malware installa **msgina32.dll** aggiungendolo nel Registro di sistema, che causa il caricamento della DLL dopo il riavvio in: **HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL**. Così da ottenere persistenza e poter accedere alle informazioni sulle credenziali tramite log off e log on utente.

RegCreateKey è una funzione che consente di creare una nuova chiave del Registro di sistema o di aprire una chiave esistente per la modifica. La chiave del Registro contiene informazioni sul sistema, come le impostazioni di configurazione, i programmi installati e le impostazioni per gli utenti.

4.3 Quale valore viene associato alla chiave di registro creata?

Ulteriore prova di ciò possiamo ottenerla accedendo al “Registry” (Regedit) Windows e aprendo la cartella **Winlogon**, noteremo che prima dell’esecuzione del malware infatti non sono presenti né file “Ginadll” né una chiave ad esso associata che invece troveremo in seguito, proveniente per l’appunto dal path del nostro malware.

PRIMA



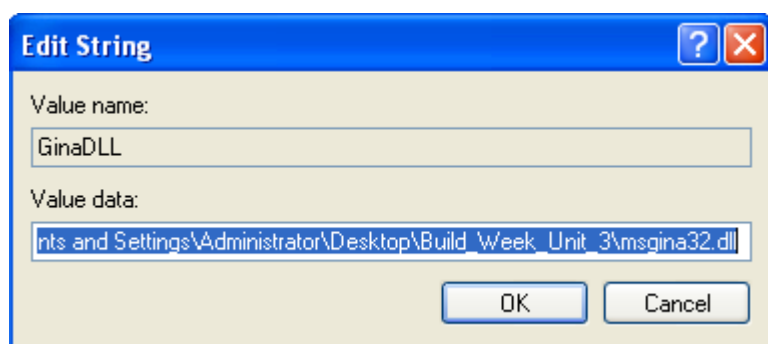
DefaultUserName	REG_SZ	Administrator
Forceunlocklogon	REG_DWORD	0x00000000 (0)
HibernationPreviouslyEnabled	REG_DWORD	0x00000001 (1)
LegalNoticeCaption	REG_SZ	
LegalNoticeText	REG_SZ	
LogonType	REG_DWORD	0x00000001 (1)
passwordexpirywarning	REG_DWORD	0x0000000e (14)
PowerdownAfterShutdown	REG_SZ	0
ReportBootOk	REG_SZ	1
scremoveoption	REG_SZ	0
SFCDisable	REG_DWORD	0x00000000 (0)
SfcQuota	REG_DWORD	0xffffffff (4294967295)
Shell	REG_SZ	Explorer.exe
ShowLogonOptions	REG_DWORD	0x00000000 (0)
ShutdownWithoutLogon	REG_SZ	0

DOPO



DefaultUserName	REG_SZ	Administrator
Forceunlocklogon	REG_DWORD	0x00000000 (0)
GinaDLL	REG_SZ	C:\Documents and Settings\Administrator\Desktop\Build...
HibernationPreviouslyEnabled	REG_DWORD	0x00000001 (1)
Key	REG_BINARY	98 33 07 01

Entrando nel particolare in riguardo al valore della nuova chiave di registro creata possiamo vedere che:



Il valore associato alla nuova chiave di registro creata è assegnato tramite l’operazione **RegSetValue** è "msgina32.dll" insieme ad un valore binario dell’ulteriore chiave creata, **Key**, tipo di valore può essere utilizzato per archiviare informazioni come immagini, file o altro tipo di dati binari.

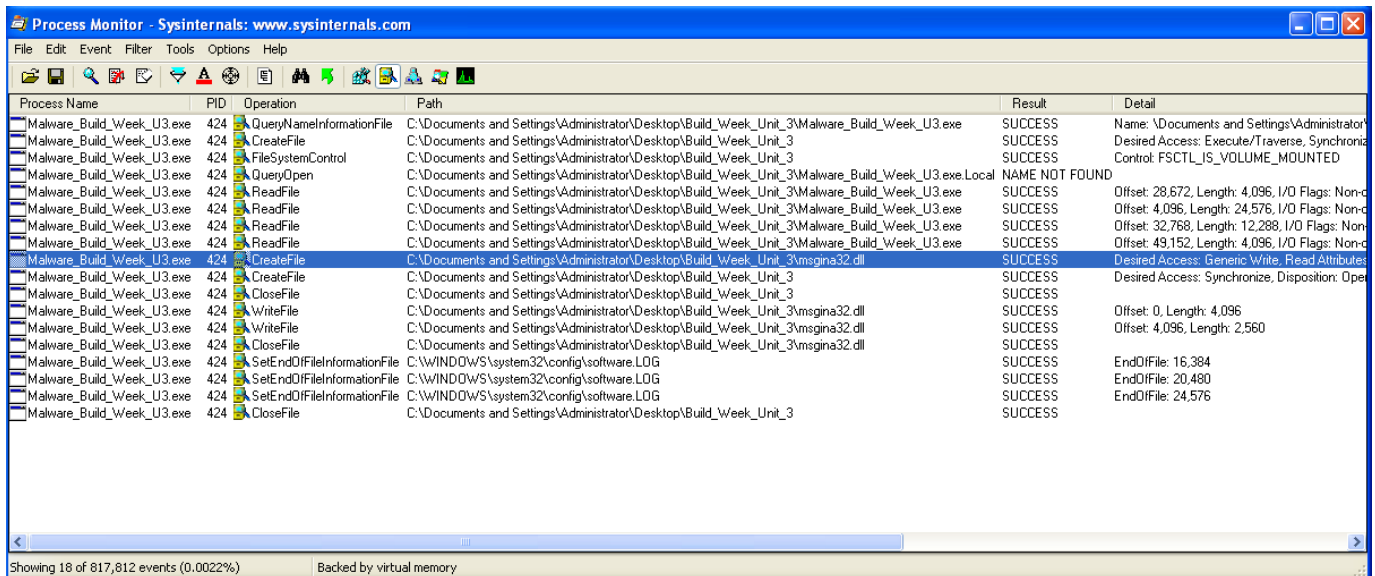
Inoltre il tipo valore associato al file “**GinaDll**” è di tipo “**REG_SZ**” (stringa di testo) che non è una chiave o un valore predefinito del Registro di sistema (regedit) di Windows.

RegSetValue: è una funzione che consente di impostare un valore specifico all'interno di una chiave del Registro di sistema. La funzione accetta come parametri la chiave del Registro in cui si desidera impostare il valore, il nome del valore e i dati da impostare.

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

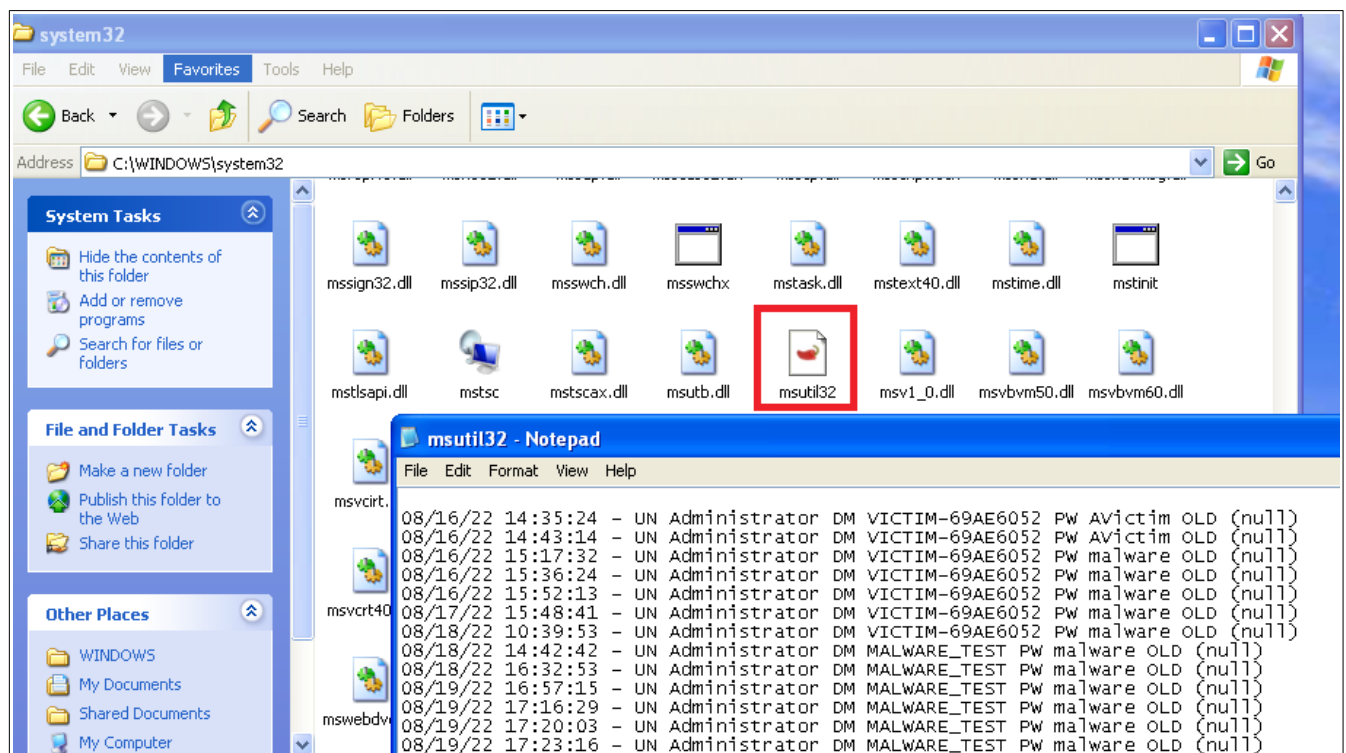
In questo caso abbiamo una chiamata di sistema che effettua un **“CreateFile()”**, quindi tipologia File Management responsabili della manipolazione file con azioni quali creazione, lettura, scrittura ed altro.

Osservando le attività dei File System possiamo vedere la **chiamata di sistema** per la creazione del file **msgina32.dll** nella stessa cartella dov'è presente il malware senza alcun avviso allo user legittimo



Process Name	PID	Operation	Path	Result	Detail
Malware_Build_Week_U3.exe	424	QueryNameInformationFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe	SUCCESS	Name: \Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe
Malware_Build_Week_U3.exe	424	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	Desired Access: Execute/Traverse, Synchronize, Read Attributes
Malware_Build_Week_U3.exe	424	FileSystemControl	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	Control: FSCTL_IS_VOLUME_MOUNTED
Malware_Build_Week_U3.exe	424	QueryOpen	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe	NAME NOT FOUND	
Malware_Build_Week_U3.exe	424	ReadFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe	SUCCESS	Offset: 28,672, Length: 4,096, I/O Flags: Non-cached, Read Attributes
Malware_Build_Week_U3.exe	424	ReadFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe	SUCCESS	Offset: 4,096, Length: 24,576, I/O Flags: Non-cached, Read Attributes
Malware_Build_Week_U3.exe	424	ReadFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe	SUCCESS	Offset: 32,768, Length: 12,288, I/O Flags: Non-cached, Read Attributes
Malware_Build_Week_U3.exe	424	ReadFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe	SUCCESS	Offset: 49,152, Length: 4,096, I/O Flags: Non-cached, Read Attributes
Malware_Build_Week_U3.exe	424	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	Desired Access: Generic Write, Read Attributes, Synchronize, Disposition: Open
Malware_Build_Week_U3.exe	424	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	
Malware_Build_Week_U3.exe	424	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	
Malware_Build_Week_U3.exe	424	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset: 0, Length: 4,096
Malware_Build_Week_U3.exe	424	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset: 4,096, Length: 2,560
Malware_Build_Week_U3.exe	424	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	
Malware_Build_Week_U3.exe	424	SetEndOfFileInformationFile	C:\WINDOWS\system32\config\software.LOG	SUCCESS	EndOfFile: 16,384
Malware_Build_Week_U3.exe	424	SetEndOfFileInformationFile	C:\WINDOWS\system32\config\software.LOG	SUCCESS	EndOfFile: 20,480
Malware_Build_Week_U3.exe	424	SetEndOfFileInformationFile	C:\WINDOWS\system32\config\software.LOG	SUCCESS	EndOfFile: 24,576
Malware_Build_Week_U3.exe	424	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	

Il malware registra le credenziali rubate in **%SystemRoot%\System32\msutil32.sys**



msutil32 - Notepad

```
File Edit Format View Help
08/16/22 14:35:24 - UN Administrator DM VICTIM-69AE6052 PW Avictim OLD (null)
08/16/22 14:43:14 - UN Administrator DM VICTIM-69AE6052 PW Avictim OLD (null)
08/16/22 15:17:32 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/16/22 15:36:24 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/16/22 15:52:13 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/17/22 15:48:41 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/18/22 10:39:53 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/18/22 14:42:42 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/18/22 16:32:53 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 16:57:15 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 17:16:29 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 17:20:03 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 17:23:16 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
```

La **msutil32.sys** è un file presente nel sistema di default, quindi di fatto non viene creato nessun nuovo file ma soltanto manipolato affinché **nome utente, il dominio e la password** registrate nel file con un timestamp possano essere letti da terze parti illegittime.


```

; int __cdecl sub_10001570(DWORD dwMessageId, wchar_t *, char)
sub_10001570 proc near

hMem= dword ptr -854h
var_850= word ptr -850h
var_828= word ptr -828h
var_800= word ptr -800h
dwMessageId= dword ptr 4
arg_4= dword ptr 8
arg_8= byte ptr 0Ch

mov     ecx, [esp+arg_4]
sub     esp, 854h
lea     eax, [esp+854h+arg_8]
lea     edx, [esp+854h+var_800]
push    esi
push    eax                ; va_list
push    ecx                ; wchar_t *
push    800h              ; size_t
push    edx                ; wchar_t *
call    _vsnwprintf
push    offset word_10003320 ; wchar_t *
push    offset aMsutil32_sys ; "msutil32.sys"
call    _wfopen
mov     esi, eax
add     esp, 18h
test    esi, esi
jz      loc_1000164F

```

Per la conferma possiamo analizzare il file creato dal malware, msgina32.dll con dissassembler IDA dove possiamo trovare la prova che vada a interagire con il msutil32.sys

[Giorno 5]

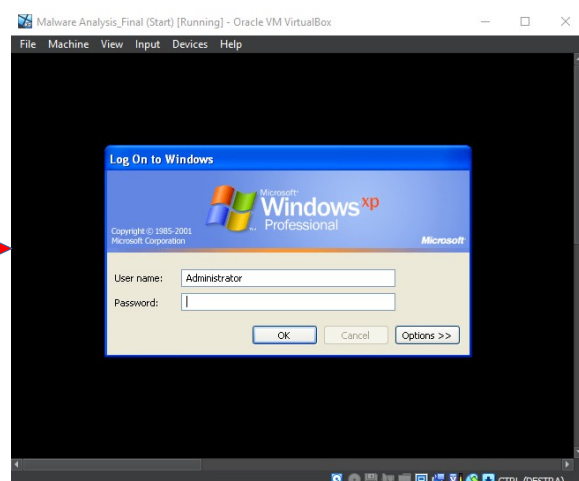
5.1 Cosa può succedere se il file .dll lecito viene sostituito con un file .dll malevolo, che intercetta i dati inseriti?

Sostituire un file .dll lecito con un file .dll malevolo può causare diversi problemi di sicurezza. Il file malevolo potrebbe contenere codice dannoso che può essere utilizzato per eseguire azioni dannose all'interno del sistema, come:

- Raccogliere le informazioni sensibili come password e dati personali
- Installare software dannosi come keylogger o rootkit
- Aprire le backdoor per permettere l'accesso remoto al sistema
- Rendere il sistema vulnerabile ad ulteriori attacchi
- Inoltre, il file .dll malevolo può anche interferire con il funzionamento normale delle applicazioni che lo utilizzano causando problemi di stabilità e anche crash.

Per proteggersi da questo tipo di attacchi, è importante mantenere aggiornati sia, il S.O che le applicazioni installate, utilizzare un software di sicurezza per rilevare e rimuovere i file dannosi e non scaricare o aprire i allegati o link sospetti da fonti sconosciute.

Possiamo vedere nel nostro caso, come una volta avviato il Malware venga modificata l'interfaccia Log-In di Windows con quella di GINA



5.2 Sulla Base della risposta sopra, delineate il profilo del Malware e delle sue funzionalità. Unite tutti i punti per creare un grafico che ne rappresenti lo scopo ad alto livello

