

Tramite l'utilizzo di **IDA pro** sono stati trovati i seguenti parametri:

```
; Attributes: bp-based frame

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

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

Funzione Principale

Parametri dichiarati

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

Variabili Dichiarate:

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

Qui possiamo vedere le sezioni del file PE:

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,contiene il codice che la CPU andrà ad eseguire una volta avviato il malware
- .rdata, contiene le info riguardo le librerie e le funzioni da importare ed esportare dall'eseguibile
- .data contiene le variabili globali del programma

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

Come possiamo vedere dall'immagine le librerie utilizzate sono le seguenti:

- KERNEL32.dll, contiene le funzioni principali utili ad interagire con il sistema operativo
- ADVAPI32.dll, libreria che contiene le funzioni per interagire con i servizi ed i registri del sistema operativo

L'ipotesi che ci possiamo fare in base alle funzioni importate tramite le librerie è che il malware potrebbe far parte della famiglia dei dropper

## Build week giorno 2

## Malware Analysis

1. Lo scopo della funzione chiamata alla locazione della memoria 00401021
2. Come vengono passati i parametri alla locazione della memoria 00401021
3. Che oggetto rappresenta il parametro alla locazione della memoria 00401017
4. Il significato delle istruzioni comprese tra gli indirizzi comprese tra 00401027 è 00401029
5. Con riferimento all'ultimo quesito tradurre il codice assembly nel corrispondente costruito c
6. valutate la chiamata alla locazione della memoria 00401047, qual'è il valore del pr. ValueName

```
.text:00401000      push    ebp
.text:00401001      mov     ebp, esp
.text:00401003      push    ecx
.text:00401006      push    0          ; lpdwDisposition
.text:00401009      lea     eax, [ebp+hObject]
.text:0040100A      push    eax          ; phkResult
.text:0040100C      push    0          ; lpSecurityAttributes
.text:00401011      push    0F003Fh     ; samDesired
.text:00401013      push    0          ; dwOptions
.text:00401015      push    0          ; lpClass
.text:00401017      push    0          ; Reserved
.text:0040101C      push    offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:00401021      push    80000002h    ; hKey
.text:00401027      call    ds:RegCreateKeyExA
.text:00401029      test    eax, eax
.text:0040102B      jz      short loc_401032
.text:0040102E      mov     eax, 1
.text:00401030      jmp     short loc_40107B
```

1. Lo scopo della funzione alla locazione della memoria 00401021 [RegCreateKeyExA] crea la chiave di registro specifica, nel caso la chiave esista già la funzione la apre.
2. Come si può notare dall'immagine, i parametri della funzione vengono passati tramite l'istruzione **push**.
3. Il parametro alla locazione 00104017 rappresenta il registro che il malware andrà ad utilizzare, per ottenere persistenza sul sistema operativo.
4. Le istruzioni comprese tra 00104027 è 00104029, dove avviene un controllo condizionato, tramite l'istruzione condizionale test che è simile all'istruzione AND, modifica il flag ZF che viene settata ad 1 solo se il risultato dell' AND è 0, questa istruzione viene utilizzata per controllare se un valore è 0 o meno  
mentre con l'istruzione jz salta alla locazione di memoria specificata se ZF è uguale ad 1.  
nel nostro caso visto che eax è uguale a 0 e quindi il flag ZF viene settato a 1 si verificherà il salto alla locazione di memoria loc\_401032
5. Traduzione dal linguaggio Assembly al linguaggio C:  

```
if (eax==0)
    vai alla locazione di memoria loc_401032
```
6. Il valore passato al parametro ValueName è "GinaDLL", la GINA opera nel contesto del processo Winlogon e, di conseguenza, la DLL GINA viene caricata molto presto nel processo di avvio. La DLL GINA deve seguire le regole in modo che venga mantenuta l'integrità del sistema, in particolare per quanto riguarda l'interazione con l'utente.

```
.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
```

Lo scopo di una DLL GINA è fornire procedure personalizzabili di identificazione e autenticazione dell'utente.

In base alle analisi svolte possiamo supporre che la funzionalità che il malware andrà ad implementare in questa sezione di codice è la **persistenza**

## Build week giorno 3

## Malware Analysis

1. Qual è il valore del parametro «ResourceName» passato alla funzione FindResourceA();
  2. Il susseguirsi delle chiamate di funzione che effettua il Malware in questa sezione di codice l'abbiamo visto durante le lezioni teoriche. Che funzionalità sta implementando il Malware?
  3. È possibile identificare questa funzionalità utilizzando l'analisi statica basica?
- In caso di risposta affermativa, elencare le evidenze a supporto.

```
.text:00401088 loc_401088: ; CODE XREF: sub_401080+2F1j
.text:00401088 mov     eax, lpType
.text:0040108D push    eax ; lpType
.text:0040108E mov     ecx, lpName
.text:004010C4 push    ecx ; lpName
.text:004010C5 mov     edx, [ebp+hModule]
.text:004010C8 push    edx ; hModule; LPCSTR lpName
.text:004010C9 call    ds:FindResourceA ; DATA XREF: sub_401080+3E1r
.text:004010CF mov     [ebp+hResInfo], eax ; "TGAD"
.text:004010D2 cmp     [ebp+hResInfo], 0
.text:004010D6 jnz     short loc_4010DF
.text:004010D8 xor     eax, eax
```

1. Come si può vedere dall'immagine tramite l'uso di OLLY Dbp è IDA Pro il valore del parametro "ResourceName" passato alla funzione FindResourceA() **"TGAD"**

004010B1	33C0	XOR EHX,EHX	
004010B3	JMP 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.FindResou	

ResourceType => "BINARY"  
Malware\_.00408038  
ResourceName => "TGAD"

hModule  
FindResourceA

2. In base alle analisi effettuate possiamo affermare che il malware faccia parte della famiglia dei dropper per tanto le funzioni che andrà ad implementare sono le seguenti

**[FindResourceA] [LoadResource] [LockResource] [SizeofResource]**

```
.text:00401088 loc_401088: ; CODE XREF: sub_401080+2F1j
.text:00401088 mov     eax, lpType
.text:0040108D push    eax ; lpType
.text:0040108E 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 loc_4010DF: ; CODE XREF: sub_401080+561j
.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+741j
.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+8C1j
.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
```

Queste APIs permettono di localizzare all'interno della sezione «risorse» il malware da estrarre, e successivamente da caricare in memoria per l'esecuzione o da salvare sul disco per esecuzione futura.

3. Si è possibile, è sufficiente andare a verificare le librerie Kernel

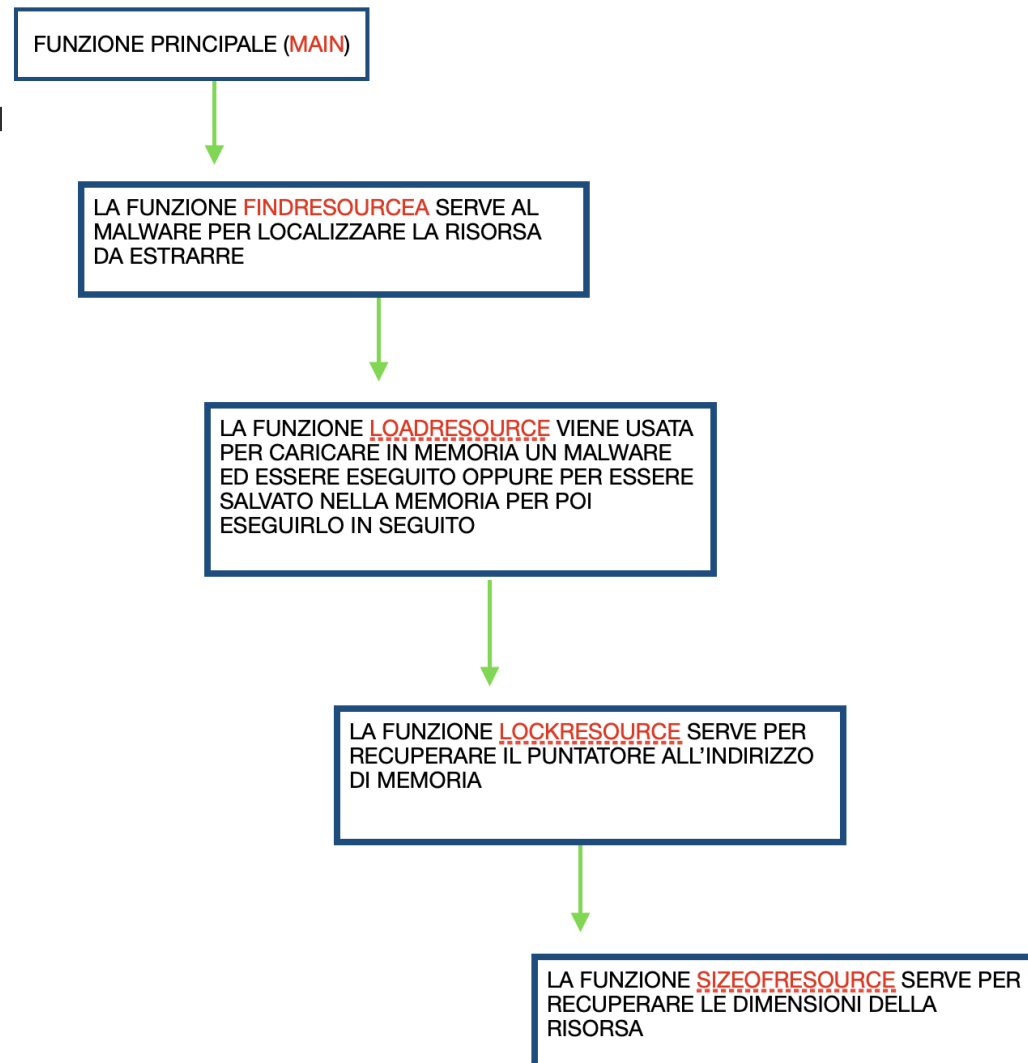
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
0000769E	N/A	000074EC	000074F0	000074F4	000074F8	000074FC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

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

4. Diagramma di flusso



## DIAGRAMMA DI FLUSSO

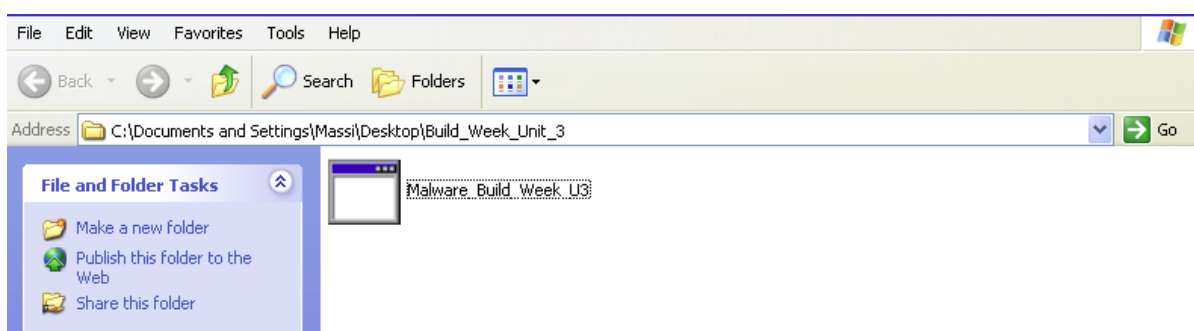


### Build week giorno 4

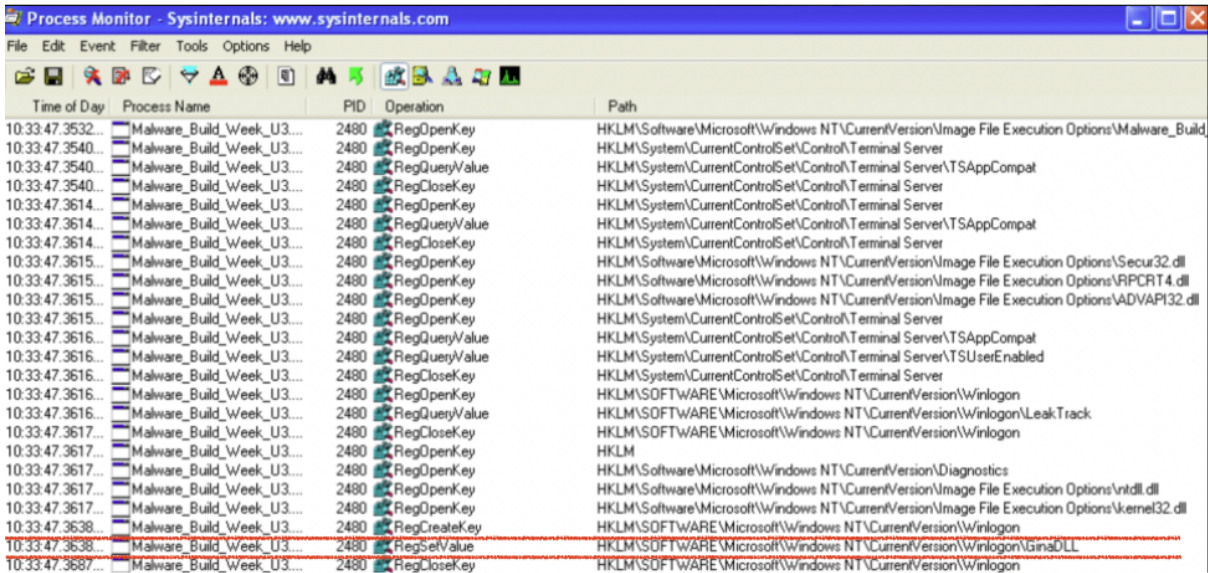
### Malware Analysis

#### 1. Monitorare l'attività del malware attraverso ProcessMonitor

Prima di avviare il malware, nella sua cartella, risulta esserci solo il .exe



Dopo aver avviato process monitor, abbiamo avviato il malware, ottenendo i seguenti risultati: aprendo il registro attività, dopo averlo filtrato al fine di mostrarci la sola l'attività del malware abbiamo recuperato le informazioni relative alla creazione della chiave di registro. Il malware preso in esame carica la libreria GINA DLL,



Time of Day	Process Name	PID	Operation	Path
10:33:47.3532...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Malware_Build...
10:33:47.3540...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:33:47.3540...	Malware_Build_Week_U3...	2480	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat
10:33:47.3540...	Malware_Build_Week_U3...	2480	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:33:47.3614...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:33:47.3614...	Malware_Build_Week_U3...	2480	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat
10:33:47.3614...	Malware_Build_Week_U3...	2480	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:33:47.3615...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Secur32.dll
10:33:47.3615...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\RPCRT4.dll
10:33:47.3615...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ADVAPI32.dll
10:33:47.3615...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:33:47.3616...	Malware_Build_Week_U3...	2480	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat
10:33:47.3616...	Malware_Build_Week_U3...	2480	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSUserEnabled
10:33:47.3616...	Malware_Build_Week_U3...	2480	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:33:47.3616...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:33:47.3616...	Malware_Build_Week_U3...	2480	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\LeakTrack
10:33:47.3617...	Malware_Build_Week_U3...	2480	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:33:47.3617...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM
10:33:47.3617...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics
10:33:47.3617...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ntdll.dll
10:33:47.3617...	Malware_Build_Week_U3...	2480	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\kernel32.dll
10:33:47.3638...	Malware_Build_Week_U3...	2480	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:33:47.3638...	Malware_Build_Week_U3...	2480	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
10:33:47.3687...	Malware_Build_Week_U3...	2480	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

GINA DLL gestisce login e processi di autenticazione sulle macchine windows, come si evince dall'immagine sotto riportata, alla terzultima riga il malware crea la chiave di registro per ottenere tutti gli accessi desiderati.

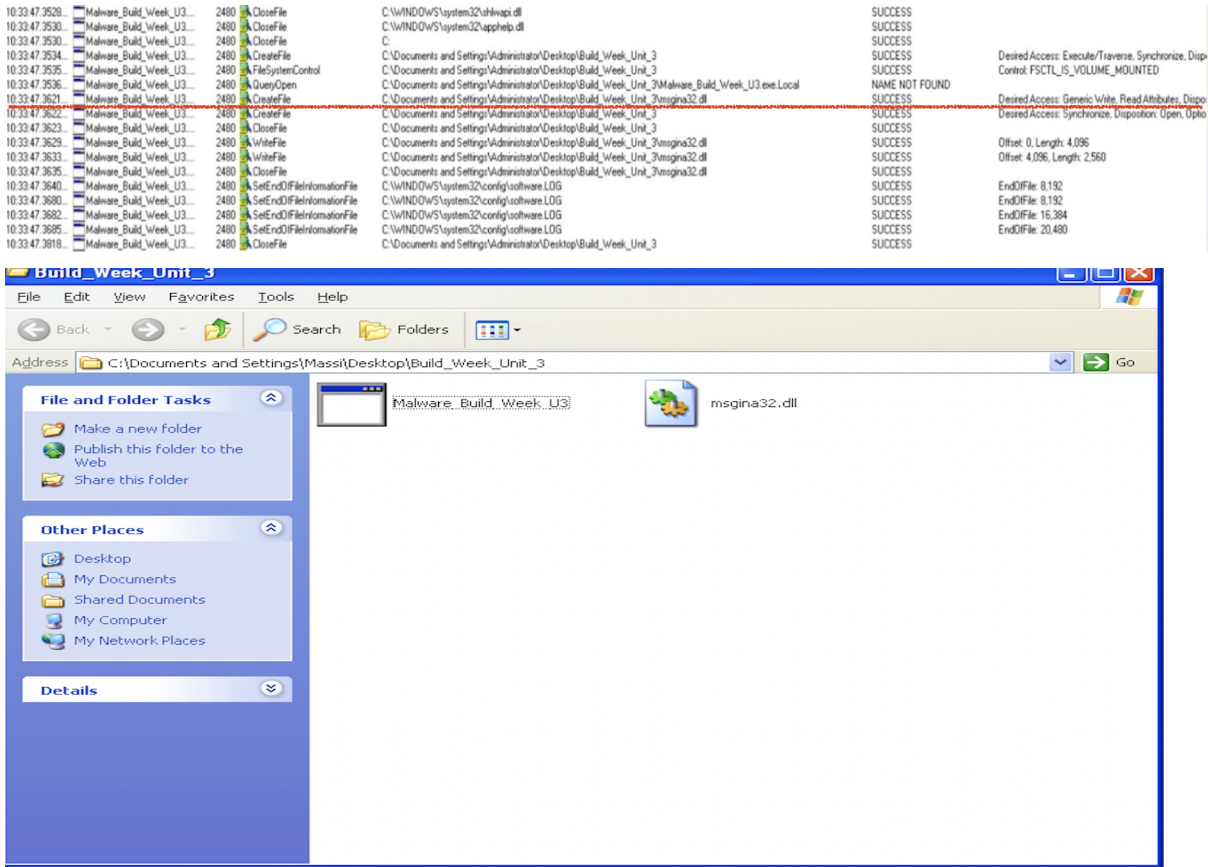
La Chiave di registro creata è **msGINA32.dll** mentre il valore associato alla sua chiave di registro è **RegSetValue**.

Come si può notare dall'immagine sotto il malware ha creato la seguente libreria

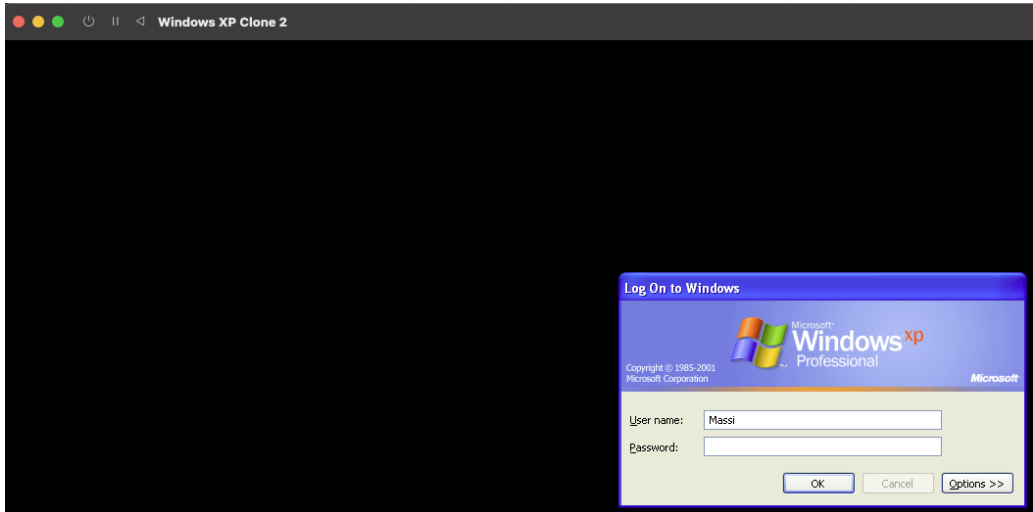
3. Una volta visualizzata l'attività sul file system, abbiamo avuto riscontro di quale chiamata di sistema ha modificato il contenuto della cartella dov'è presente l'eseguibile del malware qui di



sotto la riga interessata



Per contro prova abbiamo riavviato la macchina, è come si può notare dall' immagine sottostante il malware richiede le credenziali d'accesso.



Build week giorno 5

Malware Analysis

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

Il file .dll malevolo sostituito con un .dll lecito potrebbe camuffarsi come .dll lecito anche sotto stesso nome, e potrebbe intercettare i dati inseriti generando una connessione possibilmente in reverse\_tcp creando una connessione dalla macchina vittima alla macchina attaccante a sua insaputa in modo da essere difficilmente rilevabile e registrando gli input della macchina vittima mandando quindi i dati inseriti alla macchina attaccante. Lo si potrebbe definire un keylogger oppure si potrebbe trattare di uno spyware a tutti gli effetti, perché per reperire i dati di accesso si può procedere generando un file di testo in cui gli input dell'utente vengono registrati direttamente nel file di testo generato nella macchina attaccante (keylogger) oppure creando una connessione che permette la visualizzazione oppure generando l'upload delle credenziali inserite nella macchina attaccante (spyware)

