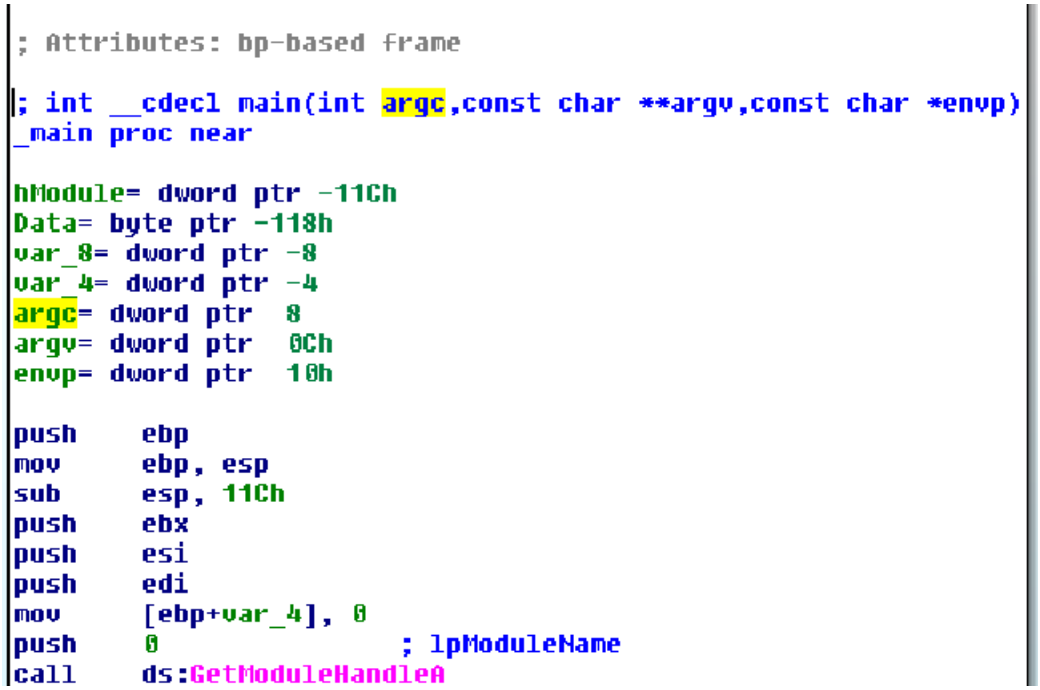
Progetto Modulo 6

Analisi statica:

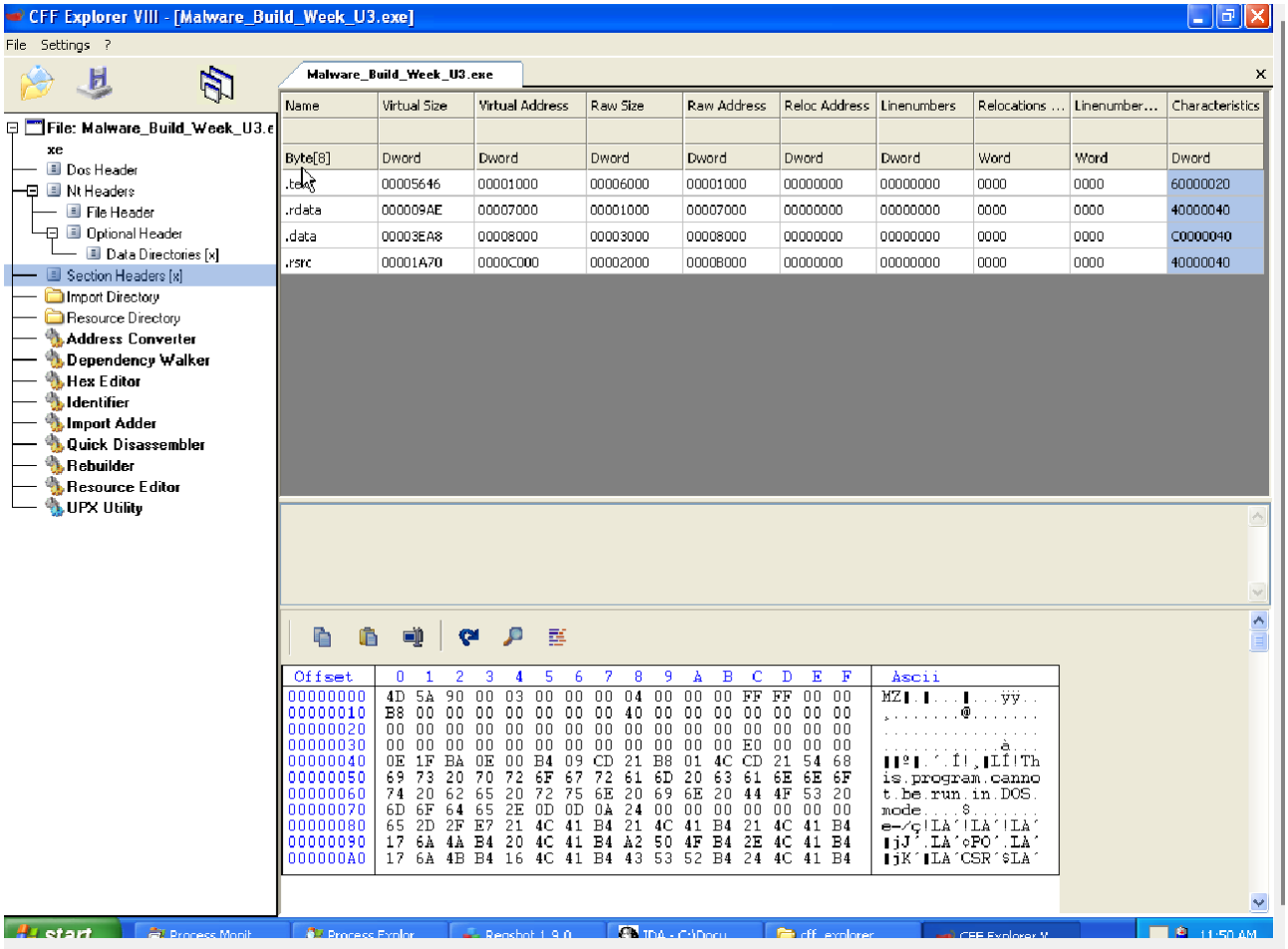
Per rispondere al primo e al secondo quesito lo screen sotto è abbastanza esaustivo:



```
; 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

push    ebp
mov     ebp, esp
sub     esp, 11Ch
push    ebx
push    esi
push    edi
mov     [ebp+var_4], 0
push    0                ; lpModuleName
call    ds:GetModuleHandleA
```

I parametri sono quelli indicati dopo la funzione main(), all'interno della parentesi, ovvero argc, argv e envp, mentre le variabili sono quelle in verde sotto dove i valori hanno il meno davanti e sono in una posizione negativa rispetto alla base dello stack epb, e sono hModule, Data, var_8 e var_4. Ovviamente ben divisi dalla parte dove ci sono le istruzioni dell'eseguibile.

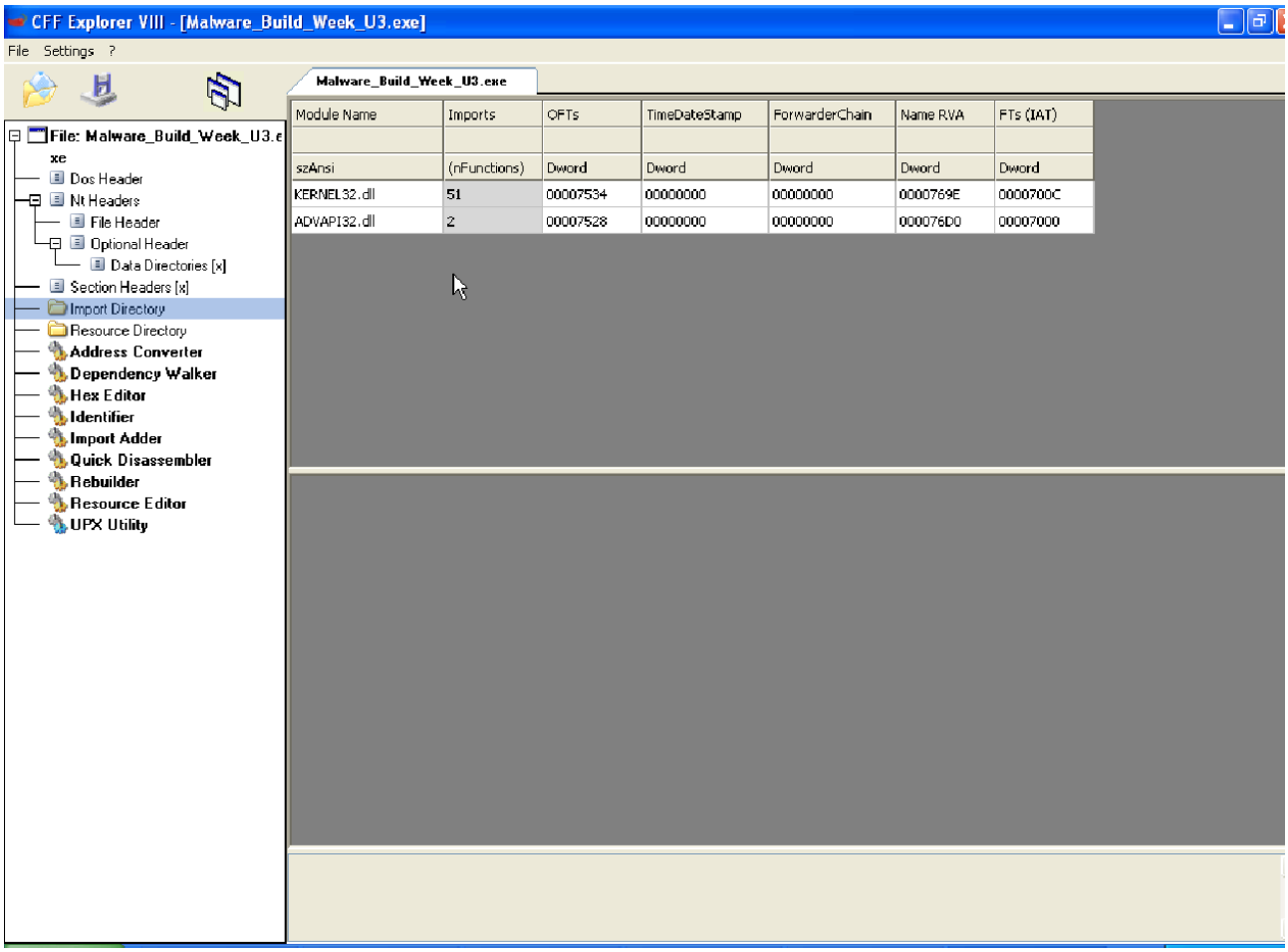
Le sezioni principali invece sono 4, e sono le seguenti in foto, visibili con CFF Explorer:



La sezione .rsrc va ad includere le risorse utilizzate dall'eseguibile che non vengono considerate parte dell'eseguibile, tipo immagine, menu, stringhe ed icone.

La sezione .rdata invece contiene le informazioni sull'import e sull'export. Può inoltre salvare dei dati read-only usati dal programma.

Le librerie importate invece sono due, le seguenti in foto:



La libreria Kernel32.dll è una libreria che può permettere al malware di utilizzare funzioni per la gestione della memoria oppure funzioni per interagire con il sistema operativo.

La libreria ADVAPI32.dll invece permette al malware di avere accesso alle chiavi di registro.

Lo scopo della funzione all'indirizzo di memoria 00401021, indicato sotto in foto, è creare la chiave di registro "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon".

Di seguito invece i parametri passati alla funzione:

```
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
```

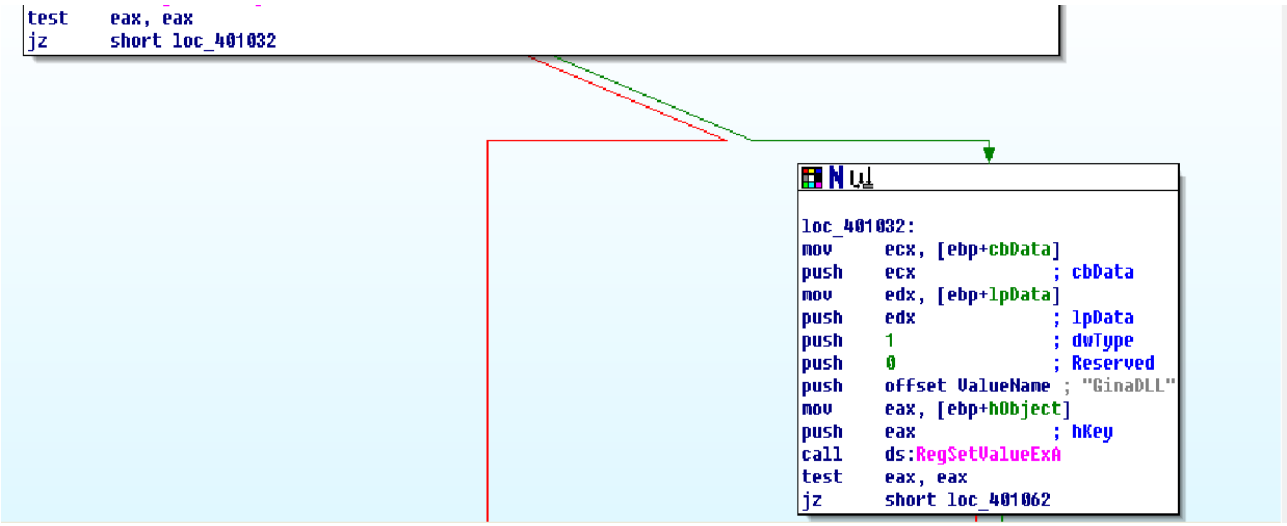
All’indirizzo di memoria 00401017 invece troviamo la chiave di registro che porta all’avvio automatico della DLL compromessa:

```
.text:00401000
.text:00401000                  push    ebp
.text:00401001                  mov     ebp, esp
.text:00401003                  push    ecx
.text:00401004                  push    0               ; lpdwDisposition
.text:00401006                  lea     eax, [ebp+hObject]
.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
```

Per quanto riguarda le istruzioni tra l’indirizzo 00401027 e l’indirizzo 00401029, stanno a verificare se il malware è stato avviato come da prassi, in caso contrario, cioè negativo, fa direttamente il salto alla loc_401032. Se si è aperto correttamente parte l’istruzione che chiude il proseguimento:

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

Qui il salto nel diagramma a flusso:



Qui sotto invece ci sono le istruzioni trasformate in codice C che vanno a mostrare come il tutto sia un classico ciclo if:

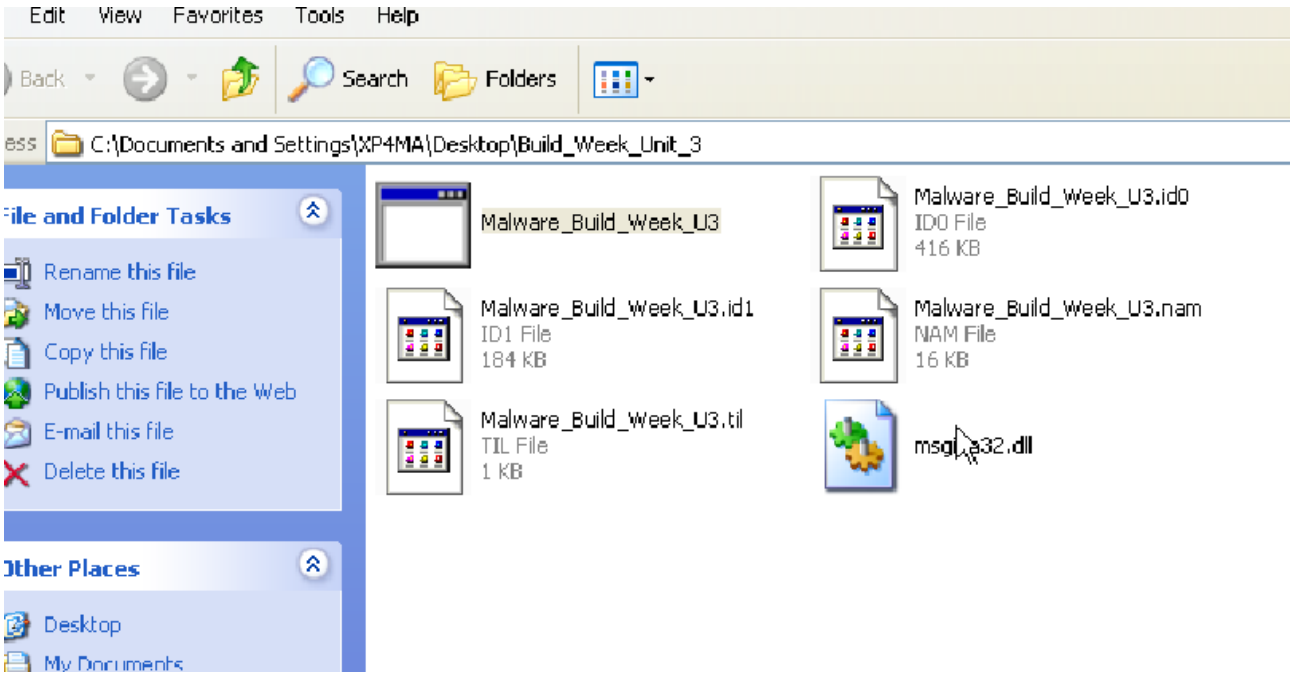
```
if(eax == 0)
 {
  funct_401032();
}
else
 {
 eax = 1;
 funct_40107B();
}
```

Analizzando la chiamata alla funzione “RegSetValueExa” alla posizione di memoria 00401047 il valore del parametro ValueName è ‘GinaDLL’, come mostrato in foto sotto:

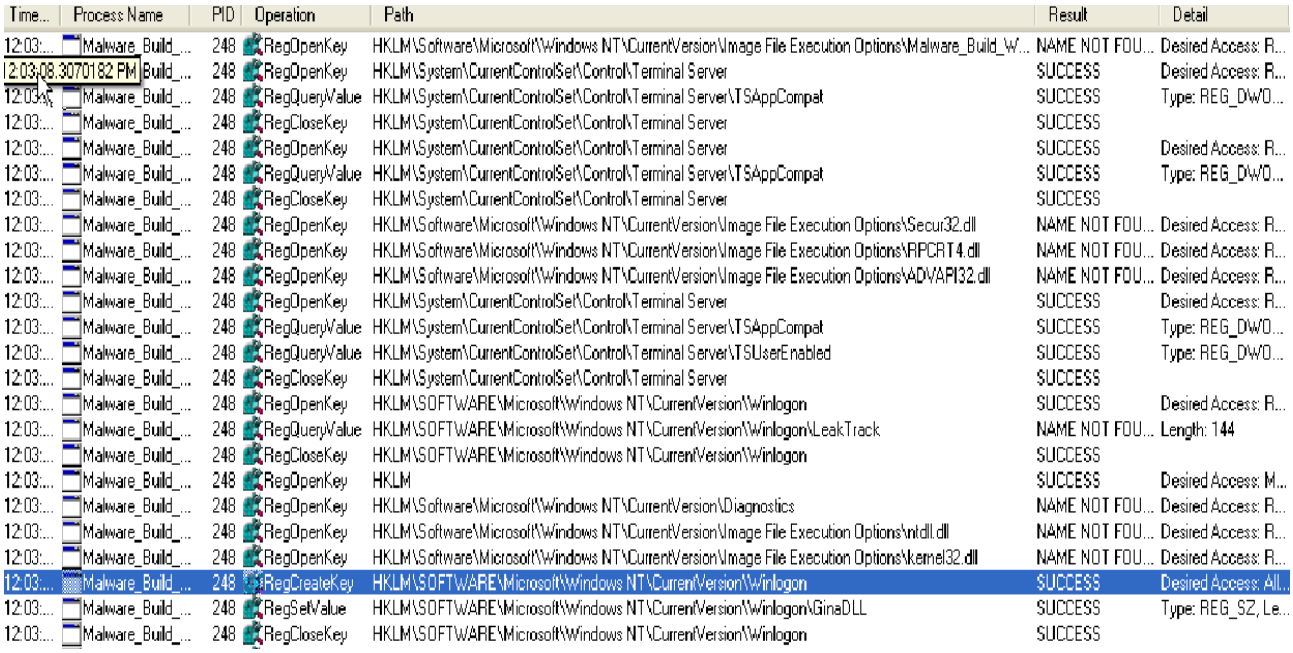
```
.text:0040103C                  push    0               ; Reserved
.text:0040103E                  push    offset ValueName ; "GinaDLL"
.text:00401043                  mov     eax, [ebp+hObject]
.text:00401046                  push    eax             ; hKey
.text:00401047                  call    ds:RegSetValueExA
```

Analisi dinamica:

Una volta avviato il Malware, all'interno della cartella dove era situato, si è creato il file msgina32.dll, ovvero la versione corrotta della GINA DLL. Infatti, come spiega Microsoft, lo scopo vero e proprio della GINA DLL è di fornire procedure di identificazione e autenticazione dell'utente personalizzabili.

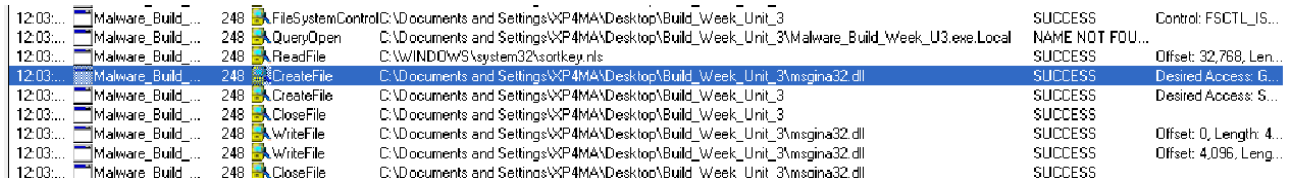


Analizzando poi le chiavi di registro con ProcMon, si evince che il malware crea la chiave di registro Winlogon e gli viene assegnato il valore msgina32.dll che aveva precedentemente trovato nella cartella del malware:



Time...	Process Name	PID	Operation	Path	Result	Detail
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Malware_Build_W...	NAME NOT FOU...	Desired Access: R...
12:03:08.3070182 PM	Build_...	248	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Access: R...
12:03:...	Malware_Build_...	248	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DWO...
12:03:...	Malware_Build_...	248	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Access: R...
12:03:...	Malware_Build_...	248	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DWO...
12:03:...	Malware_Build_...	248	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Secur32.dll	NAME NOT FOU...	Desired Access: R...
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\RPCRT4.dll	NAME NOT FOU...	Desired Access: R...
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ADVAPI32.dll	NAME NOT FOU...	Desired Access: R...
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Access: R...
12:03:...	Malware_Build_...	248	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DWO...
12:03:...	Malware_Build_...	248	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSUserEnabled	SUCCESS	Type: REG_DWO...
12:03:...	Malware_Build_...	248	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: R...
12:03:...	Malware_Build_...	248	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\LeakTrack	NAME NOT FOU...	Length: 144
12:03:...	Malware_Build_...	248	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM	SUCCESS	Desired Access: M...
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics	NAME NOT FOU...	Desired Access: R...
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ntdll.dll	NAME NOT FOU...	Desired Access: R...
12:03:...	Malware_Build_...	248	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\kernel32.dll	NAME NOT FOU...	Desired Access: R...
12:03:...	Malware_Build_...	248	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All...
12:03:...	Malware_Build_...	248	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	SUCCESS	Type: REG_SZ, Le...
12:03:...	Malware_Build_...	248	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	

Qui sotto vediamo l'assegnazione del valore msgina32.dll:



12:03:...	Malware_Build_...	248	FileSystemControl	C:\Documents and Settings\XP4MA\Desktop\Build_Week_Unit_3	SUCCESS	Control: FSCTL_IS...
12:03:...	Malware_Build_...	248	QueryOpen	C:\Documents and Settings\XP4MA\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe.Local	NAME NOT FOU...	
12:03:...	Malware_Build_...	248	ReadFile	C:\WINDOWS\system32\sortkey.nls	SUCCESS	Offset: 32,768, Len...
12:03:...	Malware_Build_...	248	CreateFile	C:\Documents and Settings\XP4MA\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	Desired Access: G...
12:03:...	Malware_Build_...	248	CreateFile	C:\Documents and Settings\XP4MA\Desktop\Build_Week_Unit_3	SUCCESS	Desired Access: S...
12:03:...	Malware_Build_...	248	CloseFile	C:\Documents and Settings\XP4MA\Desktop\Build_Week_Unit_3	SUCCESS	
12:03:...	Malware_Build_...	248	WriteFile	C:\Documents and Settings\XP4MA\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset: 0, Length: 4...
12:03:...	Malware_Build_...	248	WriteFile	C:\Documents and Settings\XP4MA\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset: 4,096, Leng...
12:03:...	Malware_Build_...	248	CloseFile	C:\Documents and Settings\XP4MA\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	

In conclusione, si può affermare che il malware in questo caso sia un DROPPER, ovvero un tipo di malware che al suon interno contiene e rilascia un altro malware. Infatti si avvale della sezione .rsrc e contiene al suo interno un logger che copia le credenziali di accesso.