Trojan-Downloader.Win32.Small or Win32/PolyCrypt Analysis

Author: Giuseppe 'Evilcry' Bonfa'
E-Mail: evilcry@gmail.com
Website: http://evilcry.altervista.org

Introduction

MD5 Hash Signature: 5f9e38abd1c20ba44ff07903489bac10
Identification: **AVG Antivirus ->** Win32/PolyCrypt
Kaspersky -> Trojan-Downloader.Win32.Small.ihj

Format: EXE and Embedded DLLs

The Essay

PolyCrypt is spreaded through infected Websites by using Exploits or every other form of abusive Download mechanism.
PolyCrypt is weakly Packer Protected, so with VMUnpack we can suddenly obtain the full working unpacked copy.

Let's trace from the EP:

```
00401000     mov     eax, 104h
00401005     mov     edx, offset dword_403033
0040100A     push    eax
0040100B     inc     ecx
0040100C     push    edx
0040100D     push    offset loc_4013BE ;points to jmp GetSystemDirectoryA
00401012     call    sub_4012BD ;Call GetSystemDirectoryA
```

PolyCrypt uses an basilar method for API call, just to deceit basical fast analysis, the call sub_4012BD access directly the jump table at the entry passed as parameter.

```
0040101B   push    offset aMsstub_dll ; "\\msstub.dll"
00401020   push    offset dword_403033 ;System Directory
00401025   push    offset loc_4013E2
0040102A   call    sub_4012BD         ;lstrcat
0040102F   pop     dword_402027
00401035   pop     ebx
00401036   push    ebx
00401037   push    80h
0040103C   push    2
0040103E   push    ebx
0040103F   push    1
00401041   push    40000000h
00401046   push    offset dword_403033 ;Full Path
0040104B   push    offset CreateFileA
00401050   call    sub_4012BD
00401060   mov     edx, esp
00401062   push    ebx
00401063   push    edx
00401064   push    1000h
00401069   push    offset dword_402027
0040106E   push    dword_403027
00401074   push    offset WriteFile
00401079   call    sub_4012BD
0040107E   pop     ecx
0040107F   push    dword_403027
```

```
00401085   push    offset CloseHandle
0040108A   call    sub_4012BD
```

This piece of code builds the a string path **c:\windows\system32\msstub.dll** and next creates this DLL (msstub.dll) and fills if it with embedded data.

```
0040108F    push    offset aDb5825eaB434C6 ; "{DB5825EA-B434-
C69E-8E2D-81387140521A}"
00401094    push    offset aClsid   ; "CLSID\\"
00401099    push    offset byte_403137
0040109E    push    offset wsprintfA
004010A3    call    sub_4012BD
004010A8    add     esp, 0Ch
004010AB    push    eax
004010AC    push    esp
004010AD    push    offset dword_40302F
004010B2    push    ebx
004010B3    push    3
004010B5    push    0
004010B7    push    ebx
004010B8    push    ebx
004010B9    push    offset byte_403137 ; "CLSID\\{DB5825EA.."
004010BE    push    80000000h
004010C3    push    offset RegCreateKeyExA
```

To overcome basical detecting attemps it's used the CLSID Splitting, the complete string is CLSID\\{DB5825EA-B434-C69E-8E2D-81387140521A}, obviously next operation is to create this Registry Key Entry.

```
004010D6    push    eax
004010D7    push    esp
004010D8    push    offset dword_40302B
004010DD    push    ebx
004010DE    push    2
004010E0    push    0
004010E2    push    ebx
004010E3    push    ebx
004010E4    push    offset aInprocserver32 ; "InprocServer32"
004010E9    push    dword_40302F
004010EF    push    offset RegCreateKeyExA
00401111    inc     eax
00401112    push    eax
00401113    push    offset aApartment ; "Apartment"
00401118    push    1
0040111A    push    ebx
0040111B    push    offset aThreadingmodel ; "ThreadingModel"
00401120    push    dword_40302B
00401126    push    offset RegSetValueExA
0040112B    call    sub_4012BD
0040113A    inc     eax
0040113B    push    eax
0040113C    push    offset dword_403033
00401141    push    1
00401143    push    ebx
00401144    push    ebx
00401145    push    dword_40302B
0040114B    call    RegSetValueExA
00401150    push    dword_40302B
00401156    call    RegCloseKey
```

This piece of code creates into the previously builded CLSID the following entry:

HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID
{CLSID}\InprocServer32 = iexplorer.exe
\ThreadingModel = Apartment (which is single threaded)

In other words Registers a 32-bit in-process server and specifies the threading model of the apartment the server can run in, in our case the InprocServer32 is Internet Explorer.

So the malicious dll (msstub.dll) could be called by IE, indeed the next operation accomplished by PolyCrypt is to Open IE with ShellExecuteA(), finally builds a .bat script file, called dmfg.bat to delete the Executable..

PolyCrypt is completly Reversed, let's see now what happens into msstub.dll