Target: ..... GIF movie Gear 3.0.1
Company: ..... gamani
Url: ..... http://www.moviegear.com
Protection type: ..... 30 days trial, serial, nagscreen at exit
What we want to achive: .. registration with a trivial serial and nop the nagscreens

Tools required:
- a disassembler (W32Dasm)
- a hexeditor
- a tool to spy API functions (APISpy)

----------01010100100101001010010010010010101001010

Welcome to my next tutorial ;-).
Today we're gonna discover the very easy to find security holes is GIF Movie Gear 3.0.1. A very nice software to build and edit gif-animations. 39$ is really not much for this program, so if you really like it, buy it!!!! This is some kind of morality a cracker, hacker should have.

OK, lets start. The program has a 30-day trial period with all features enabled. When quitting the program, a slow nagscreen will appear and tell us the days left. Once the trial period is expired the registration will eventually be done by entering a Serial number (or before the expiration > Menu: Help/register).

There are of course many ways to get rid of those protections. I'll show you just one. The way I cracked it it's surely not the best. Because I needed the program it in due course I've choose the fastest and most obvious way (O000Ps!!!!!!!!, I installed the trial version in order to make some nice gif animations, and while playing stupidly around with the systems clock with the intention to check another software, I blocked the software... so I had to crack it). You can figure out the other ways by yourself, it's a good exercise.

Ok, the first thing we want to get rid of, is the nagscreen that pops up when closing the program. Launch APISpy and load GIF Movie Gear's main executable into it. The first calls are some kind of registration routines from the registry -> **RegQueryValueExA()** and **RegQueryValueExA** = 0. If you want you can use a registry monitoring tools to check what operations will be performed inside the registry, but in the way I have cracked the software it's not necessary. Now close the program. As usual the dirty nagscreen pops up. Check the calls! Again there are of course calls inside the registry. Obviously the program will check the serial in the registry as the registered version won't have the nagscreen pops. At address **00406728** you'll notice a API function called **DialogBoxParamA** followed by other stuff. Remember this address!!!

The second step to do is to push up the system's clock in order to simulate the 30-day past trial. Try to launch the program and you'll see a new nagscreen stating "*The 30-day trial of Movie Gear has expired*". Click onto Register... and enter a casual Name and Serial. Press Ok and and a **MessageBox** will appear "*The information you have provided is invalid...*". Now open W32Dasm and disassemble the programs exe and search after the message just seen.

This is the snippet of code you'll see:

```
* Reference To: USER32.EndDialog, Ord:00B9h
|
:00431A2D FF15FC824400      call dword ptr [004482FC]
:00431A33 5F                pop edi
:00431A34 5E                pop esi
:00431A35 33C0              xor eax, eax
:00431A37 5B                pop ebx
:00431A38 81C41C010000      add esp, 0000011C
:00431A3E C21000            ret 0010



* Referenced by a (U)nconditional or (C)onditional Jump at Address:
```

```
|:0043198E(C)
|
:00431A41 6A30 push 00000030

* Possible Reference to String Resource ID=40213: "Invalid Registration Info"
|
:00431A43 68159D0000 push 00009D15

* Possible Reference to String Resource ID=40212: "The information you have provided
is invalid. Please be sure"
|
:00431A48 68149D0000 push 00009D14
:00431A4D 56 push esi
:00431A4E E83DD4FDFF call 0040EE90
:00431A53 83C410 add esp, 00000010
```

Note the conditional jump which handles this routine at address :0043198E

Here the program performs a check if the Serial entered is valid or not

```
:0043198C 85C0              test eax, eax
:0043198E 0F84AD000000      je 00431A41
:00431994 8D542410          lea edx, dword ptr [esp+10]
:00431998 8D44240C          lea eax, dword ptr [esp+0C]
```

Note that if the check results in equality, it jumps you again to the error message. Else it will continue and invoke some API like **RegCreateKeyExA**, **RegSetValueExA**... . Our intention is not to perform the jump. So write down this address and the corresponding offset. Now launch your favorite hexeditor and load the exe, than go to the offset noted at address 0043198E and change the 0F84 to 0F85 which is the opcode for the opposite instruction (jne). Before saving the file, be sure to make a copy of movgear.exe!!! Very important!!! DONE? Ok, save the file and try to start Gif Movie Gear. Try to enter the requested information and you'll see that apparently it has been registered successfully. But when clicking HELP/about Gif Movie Gear, the nagscreen is the same as before -> "*This trial software expires in... days*". Depending on how you have manipulated the system clock it will display 30 days or -1 day. Somewhat strange... Booh???

We can think of that once the software has been successfully registered, the same nagscreen should display something like "This software is registered to:" or "Name: Code:". etc. So search in the disassembled file for example for "registered". Pic up the most suitable result you got. Probably you also will aggree to this match:

```
:0040EA57 83C408            add esp, 00000008
:0040EA5A 83F801            cmp eax, 00000001
:0040EA5D 0F85E0000000      jne 0040EB43
:0040EA63 807C246073        cmp byte ptr [esp+60], 73
:0040EA68 7527              jne 0040EA91
:0040EA6A 8B15E4B64400      mov edx, dword ptr [0044B6E4]
:0040EA70 A1E8B64400        mov eax, dword ptr [0044B6E8]
:0040EA75 8B0DECB64400      mov ecx, dword ptr [0044B6EC]
:0040EA7B 8954245C          mov dword ptr [esp+5C], edx
:0040EA7F 8A15F0B64400      mov dl, byte ptr [0044B6F0]
:0040EA85 89442460          mov dword ptr [esp+60], eax
:0040EA89 894C2464          mov dword ptr [esp+64], ecx
:0040EA8D 88542468          mov byte ptr [esp+68], dl

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0040EA68(C)
|
:0040EA91 8D44245C          lea eax, dword ptr [esp+5C]
:0040EA95 8D8C24C0000000    lea ecx, dword ptr [esp+000000C0]
:0040EA9C 50                push eax
:0040EA9D 51                push ecx

* Possible Reference to String Resource ID=40326: "Name: %s
```

```
Code: %s"
|
:0040EA9E 68869D0000         push 00009D86
:0040EAA3 E8B8030000         call 0040EE60
:0040EAA8 83C404             add esp, 00000004
:0040EAAB 8D94242C010000     lea edx, dword ptr [esp+0000012C]
:0040EAB2 50                 push eax
:0040EAB3 52                 push edx
:0040EAB4 E8D3040300         call 0043EF8C
:0040EAB9 83C410             add esp, 00000010
:0040EABC 8D842424010000     lea eax, dword ptr [esp+00000124]
:0040EAC3 50                 ush eax

* Possible Reference to Dialog: DialogID_0091, CONTROL_ID:044F, ""
|
:0040EAC4 684F040000         push 0000044F
:0040EAC9 56                 push esi
:0040EACA FFD3               call ebx
:0040EACC 50                 push eax

* Reference To: USER32.SetWindowTextA, Ord:025Eh
|
:0040EACD FF1538844400       Call dword ptr [00448438]

* Possible Reference to String Resource ID=40312: "This Software is registered to:"
|
:0040EAD3 68789D0000         push 00009D78
:0040EAD8 E883030000         call 0040EE60
:0040EADD 83C404             add esp, 00000004
:0040EAE0 50                 push eax
:0040EAE1 689F040000         push 0000049F
:0040EAE6 56                 push esi
:0040EAE7 FFD3               call ebx
```

Notice that this operation is performed by a check at address :0040EA68 as a conditional jump. Again some comparison routines... The conditional jump at the address :0040EA5D also performs some kind of checking, but if not equal throws the program to another location. In the way we want to crack the program it's not interesting for us. So we may need to nop out this memory address in order to let the program perform the jump below, but again with the inverse instruction -> jne change to je.

```
:0040EA5D 0F85E0000000       jne 0040EB43
:0040EA63 807C246073         cmp byte ptr [esp+60], 73
:0040EA68 7527               jne 0040EA91
```

Open your hexeditor and go to the equivalent offset at address :0040EA5D. The following byte sequence =F85E0000000 has to be replaced with 90909090909 in order to nop out this address and to preserve the current memory allocation. At the offset of the other address change 7527 into 7427 in order to invert the instruction. OK. Remember the nagscreen that pops up each time the program is closed? You have noted the address at the beginning. Open W32Dasm and go to that location.

```
:00406707 6A00               push 00000000
:00406709 6A00               push 00000000
:0040670B E830AF0200         call 00431640
:00406710 83C408             add esp, 00000008
:00406713 83F801             cmp eax, 00000001
:00406716 7416               je 0040672E
:00406718 A168854600         mov eax, dword ptr [00468568]
:0040671D 6A01               push 00000001
:0040671F 68D0E84000         push 0040E8D0
:00406724 56                 push esi

* Possible Reference to Dialog: DialogID_0064
|
```

```
:00406725 6A64              push 00000064
:00406727 50                push eax

* Reference To: USER32.DialogBoxParamA, Ord:0093h
|
:00406728 FF150C834400 Call dword ptr [0044830C]

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00406716(C)
|
:0040672E 8B8C24B8000000    mov ecx, dword ptr [esp+000000B8]
:00406735 8B9424B4000000    mov edx, dword ptr [esp+000000B4]
:0040673C 51                push ecx
:0040673D 52                push edx
:0040673E 6A10              push 00000010
:00406740 56                push esi
```

Note the conditional jump at :00406716. Again a comparison routine. If the two statements are equal, the program jumps at address :0040672E. In this case the program results fully registered and no nagscreen will pop up. So we have to change it in a way to perform the jump anyway.

You can replace the je with jne or simply even with a jmp

7516 --> jne
EB16 --> jmp

Save the program and check the results. To be sure our changes has taken effect, change your system clock again. Now start the program. Click Register, enter some values and press OK. Shut down the program and start it again. A welcome message is displayed. Fortunately only the first time (or after each clock manipulation), after it won't. Check HELP/about Gif Movie Gear. Voilà, the nagscree says the software is registered. A Better solution will be "registered to:" and below the Name: and Serial: . But this task is left over for you. I've already spent too much time for this program and tutorial.

Now that you have patched the program you may want to make a patch executable. Here is some C++ code, just use DOS File comparison feature and replace the relative offsets and bytes. If you do not know how to do this, just refer to the tutorial about Ulead PhotoImpact 6.0.

```
#include <fcntl.h>
#include <fstream.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{

FILE *fp;

printf("\n");
printf("ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿\n");
printf("³ Gif movie Gear 3.0.1 Crack, Written by Rusty! ³\n");
printf("³ (Get the tutorial ) ³\n");
printf("ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ\n");


fp=fopen("movgear.exe","r+");

if(fp==NULL)
{
cout << " ß Could not open file for patching...\n\n";
return 1;
}

// Patch our locations //

fseek(fp,0x6CF8,SEEK_SET); // seek to location //
fputc(0x85,fp); // patch bytes //

fseek(fp,0x6FFC,SEEK_SET); // seek to location //
fputc(0x75,fp); // patch bytes //
```

```
fseek(fp,0x6FEB,SEEK_SET); // seek to location //
fputc(0x74,fp); // patch bytes //

fseek(fp,0x6F66,SEEK_SET); // seek to location //
fputc(0x85,fp); // patch bytes //

fseek(fp,0x6F56,SEEK_SET); // seek to location //
fputc(0x85,fp); // patch bytes //

fclose(fp);
printf(" ß Finished Patching!\n");
return 0;
}
```

See you at my next tutorial.

Bye ;-)

Rusty
rusty79@totalmail.com