Target: ..... Ulead PhotoImpact 6.0 trial
URL: ..... http://www.ulead.com
Protection type: ... expires after 30 days

Tools required:
a disassembler (Win32Dasm v8.9 -> recommendet)
a hexeditor of your choice (I prefer UltraEdit)
a C++ compiler (to compile the patch)

Sometimes I wonder about the high vulnerability of software protections. In many cases it's really just a matter of changing obvious sets of instructions. Ok, have you arranged all those things? Excellent, let's start:

Install the program and run it. A Dirty nag-screen will pop up and inform you courteously that you have 30 days of the trial period left until the program will commit suicide and occupy precious memory on your machine without any productiveness. Ok, so what you probably want to achieve is to be able to run the program again, this time perhaps without the dirty nag-screen and most important without time limit. The next step is to push up your system clock so that the software won't run again. So push it up 2 months to be sure the time limit has been bypassed. Now try to run the program! As expected it won't. The boring nag-screen appears and states something like: "Your 30-day trial period for PhotoImpact 6 has expired...".

Now launch Win32Dasm, load the main executable and look at the modules imported. Especially look at the Name: section. Pay attention in which modules there is a possible reference to Registration in any form. As the most attractive you'll probably find u32Cfg.dll. In some cases you'll need to browse several dynamic link libraries in order to find the right one. You will find information like this in the address references:

```
Addr.80000012      hint(0012)          Name:       ulcDispReg
Addr.80000011      gubt(0011)          Name:       ulcProductKeyName
...etc.
```

Ok, close the current disassembled and start disassembling u32Cfg.dll. The next step to do is to search the code after the message got before "Your 30-day trial..." in the String data references. When found double click the string reference. You will be pushed at that location. There are two such locations, but in this case only one is interesting for us, namely the last one, placed towards the end of file. Here is the code:

```
Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:4EB06F55(C), :4EB06F65(C), :4EB06FEB(C), :4EB06FFC(C)

:4EB07053 8D442438          lea eax, dword ptr [esp+38]
:4EB07057 50                push eax
:4EB07058 6808040000        push 00000408
:4EB0705D E89ECAFFFF        call 4EB03B00
:4EB07062 83C408            add esp, 00000008
:4EB07065 85C0              test eax, eax
:4EB07067 7515              jne 4EB0707E
:4EB07069 896C2438          mov dword ptr [esp+38], ebp

Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:4EB07082(C)

:4EB0706D 8B4C2428          mov ecx, dword ptr [esp+28]
:4EB07071 51                push ecx
:4EB07072 6A10              push 00000010

Possible Reference to String Resource ID=22006: "The %d-day trial period has expired.
Please update to the fu"

:4EB07074 68F6550000        push 000055F6
:4EB07079 E964010000        jmp 4EB071E2
```

```
Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:4EB07067(C)

:4EB0707E 396C2438          cmp dword ptr [esp+38], ebp
:4EB07082 74E9              je 4EB0706D
:4EB07084 E8A7E0FFFF        call 4EB05130
:4EB07089 3D33010000        cmp eax, 00000133

:4EB0708E 0F8556010000      jne 4EB071EA
:4EB07094 8B1570B0B04E      mov edx, dword ptr [4EB0B070]
:4EB0709A 8D4C2464          lea ecx, dword ptr [esp+64]
:4EB0709E 6804010000        push 00000104
:4EB070A3 51                push ecx
:4EB070A4 52                push edx
```

Note that this snippet of code is referenced by several conditional/unconditional jumps. The conditional jump at 4EB07082 is just a jump back to the preview message. So look at the other ones. There you will find some compare functions followed by a je:

```
test eax, eax
je 4EB07053
```

All four conditional jumps will throw the program to the beginning of the dirty nag-screen code. So our intention is to change it in a way to reverse this process. We'll change all 4 conditional jumps to jne -> jump if not equal. I'll show you later how to fix this. Ah, sorry, I forgot to advise you to sign up the address offsets of those location... done? Ok.

[VERY IMPORTANT] The first thing to do before you go ahead and do some byte manipulation is to make a copy of the file (u32Cfg.dll), since a minimal error would cause the program not to be able to run anymore.

Ok, now we will go ahead and patch the program to see if the work done was effective. Launch your favorite hex-editor and load our target. Now go to every single offset of the addresses noted above and change the bytes in a way to change the value to the opposite instruction.
This are the relative opcodes of the two instructions:
je = 84 or 74
jne = 85 or 75

After having changed the bytes, save the program. Some programs do not allow byte modifications. In this case save the hex file with the modified one and launch the program. Ok. Now replace the original u32Cfg.dll with the modified one and launch the program. The classic nag-screen will appear, but this time it allows you to run the program anyway. Wahyyyyaahhh. CRAcKKKKKKED!!!!!!!!!!!!!!!!

There is of course also a way to eliminate that fucking nag-screen, but I'd say, just to force your brain thinking ;-), try it on your own. I'll publish the modified tutorial in some ....s. However the patch code below does already contain the patch to kill the nag-screen - just check the locations and try to get affiliate with the program flow.

Ok, now that we discovered the protection hole in the program, you're probably willing to make a patch (a small executable to distribute around) in order to crack the software.
Ok, we'll use DOS file-comparison utility to compare the just modified u32Cfg.dll with the original one. Rename the modified u32Cfg.dll to u32Cfg.crk and copy it to the location where you stored the original one. Type in a DOS console:

```
FC /B u32Cfg.crk u32Cfg.dll > u32Cfg.dif
```

Open the just created file (u32Cfg.dif) in a text editor and you'll see this:

```
Confronto in corso dei file u32Cfg.crk e U32CFG.DLL (italian OS ;-) )
00006F56: 85 84
```

```
00006F66: 85 84
00006FEB: 74 75
00006FFC: 75 74
```

The first column of bytes are the modified ones, the second the original ones. Now I'll show you how to write a very simple patch for that program. Probably it's not the way a pro would, but it will do it anyway. There are of course a lot of software programs available to simplify the process of creating a patch, but in my opinion the satisfaction is much greater in coding the own patches.

(choosen language C++):

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

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

FILE *fp;

printf("\n");
printf("ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿\n");
printf("³ PhotoImpact6.0 Crack, Written by Rusty! ³\n");
printf("³ (Get the tutorial ) ³\n");
printf("ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ\n");


fp=fopen("u32Cfg.dll","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;
}
```

Compile this code and voila! We have created a patch to force the software program to run beyond the time limit.

THE CODE below is a much more stable method of creating a patch, you just have to change the offsets and the bytes. Have fun, and remember to check my site for coming tutorials. Bye.

Rusty
rusty79@totalmail.com

----------.....----------.....-.-.-.---.--.--.--.----------

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BANNER "www.rusty79.da.ru"
#define HEADER "PhotoImpact6.0 CRAck"
#define AUTHOR " ---- Buy the program if you really like it ----"
#define FILENAME "u32Cfg.dll"

#define OPENTYPE "r+"
#define HI(x) (x >> 8)
#define LO(x) (x & 0x00FF)
#define MAXBYTESPERCRACK 8
#define BYTE unsigned char
#define ULONG unsigned long

/* crkLen indicates the number of differant offsets to make changes to */
#define crkLen 2

/* I usually paste the diffs from a fc /b <Orig> <Patched> Here.
0000231B: 74 90 See.. this is the first location
0000231C: 03 90 2 bytes long...
00005977: 74 90 and this is the second location
00005978: 03 90 also 2 bytes long...
*/

/* the list of offsets to start patching the file at (count should match crkLen )
*/
ULONG crkOfs[crkLen] =
{
0x0000231B,
0x00005977
};

/* crkTbl first states the bytes to change at the given location
** followed by a list of 0xBefore|After byte sets...
** for instance.. there are 2 differant offsets (as shown in crkOfs, above)
** of which both have 2 bytes to change.. for the first location the
** patch will look for 7403 and replace it with 9090....
*/

ULONG crkTbl[crkLen][MAXBYTESPERCRACK] =
{
{2, 0x7490, 0x0390},
{2, 0x7490, 0x0390}
};

main(int argc, char *argv[])
{
BYTE inbuf[MAXBYTESPERCRACK], inchar, bSame, patchbyte;
FILE *fp;
int bts, Ix, Jx;
ULONG fOffset;
long io_num;
int result, numread, dupchars, alreadpatched;


/*-----------------------------------------*/
/* print out the Header and the Author lines */
/*-----------------------------------------*/
printf("%s\n%s\n%s\n", BANNER, HEADER, AUTHOR);

/*-----------------------------------------*/
/* Open the File */
```

```
/*-----------------------------------------*/
if( (fp = fopen( FILENAME, OPENTYPE )) == NULL )
{
printf("Cannot open %s. Please move %s to the directory containing %s. Exiting.\n", FILENAME,
argv[0], FILENAME);
return(0);
}

/*-----------------------------------------*/
/* Verify the patch data */
/*-----------------------------------------*/
bSame = 1;
alreadpatched = 0;
for (Ix = 0; Ix < crkLen; Ix++)
{
/* Go to the spot */
fOffset = crkOfs[Ix];
result = fseek( fp, fOffset, SEEK_SET);
if( result )
{
printf( "Error Seeking in File.(s)\n" );
fclose(fp);
return(0);
}

/* Read the signature buffer */
bts = (int) crkTbl[Ix][0];
dupchars = 0;
numread = fread( inbuf, sizeof( char ), bts, fp);
for (Jx = 0; Jx < bts; Jx++)
{
inchar = inbuf[Jx];
patchbyte = (BYTE) HI(crkTbl[Ix][Jx+1]);

if (patchbyte != inchar)
{
if (inchar == ((BYTE) LO(crkTbl[Ix][Jx+1])))
dupchars++;

bSame = 0;
}
}

if (bSame == 0)
{
if (dupchars == bts)
{
alreadpatched++;
}
}
}

/* Are they the same? */
if (!bSame)
{
if (((alreadpatched+1) == crkLen) || (alreadpatched == crkLen))
printf("Patch has already been applied.\n");
else
printf("Incorrect version\n");
return(0);
}

/*-----------------------------------------*/
/* Patch the Data */
/*-----------------------------------------*/
for (Ix = 0; Ix < crkLen; Ix++)
{
/* Go to the spot */
fOffset = crkOfs[Ix];
result = fseek( fp, fOffset, SEEK_SET );
if( result )
{
printf( "Error Seeking in File.(s)\n" );
fclose(fp);
```

```
return(0);
}

/* Read the signature buffer */
bts = (int) crkTbl[Ix][0];
for (Jx = 0; (Jx < bts); Jx++)
{
patchbyte = (BYTE) LO(crkTbl[Ix][Jx+1]);
io_num = fputc(patchbyte, fp );
}
}


/*-----------------------------------------*/
/* Verify the patch was successful data */
/*-----------------------------------------*/
bSame = 1;
for (Ix = 0; Ix < crkLen; Ix++)
{
/* Go to the spot */
fOffset = crkOfs[Ix];
result = fseek( fp, fOffset, SEEK_SET);
if( result )
{
printf( "Error Seeking in File.(s)\n" );
fclose(fp);
return(0);
}

/* Read the patched buffer */
bts = (int) crkTbl[Ix][0];
numread = fread( inbuf, sizeof( char ), bts, fp);
for (Jx = 0; Jx < bts; Jx++)
{
inchar = inbuf[Jx];
patchbyte = (BYTE) LO(crkTbl[Ix][Jx+1]);

if (patchbyte != inchar)
{
bSame = 0;
}
}

}


/* Are they the same? */
if (!bSame)
{
printf("File was not patched successfully (write protected?)\n");
return(0);
}

/* Notify Success */
printf( "File was patched Successfully.\n" );


/* Close the file */
fclose(fp);

/* Now return positively */
return (1);
}
```