Ghiribizzo's Cracking Tutorial

A Christmas Digression

Presents for All!

I was introduced to a nice piece of software by a friend of mine recently. It's called Virtual CD-ROM and is basically 'fakecd for Windows 95'. But this isn't really a tutorial for cracking it (although I will give a crack), it's more of a moan by me. :-)

PGP and Signed Tutorials

My tutorials and programs should be signed electronically using PGP. PGP 5 supports DSS/Diffie-Hellman keys. These keys are not supported by previous versions of PGP.
You should check the signature to make sure that the tutorial and especially its program files have not been tampered with. All cracks, tutorials and zip files I release will be signed. This will prevent tampering and will hopefully reduce the chances of viral infection.
My signature will also be the only way you can identify me as my email address will often change.

My Web Site: http://Ghiribizzo.home.ml.org
My Email: Ghiribizzo@geocities.com
My Backup Email: Ghiribizzo@hotmail.com

Virtual CD-ROM 1.0

A friend gave me a copy of this along with a ready made key generator. When I installed the program and ran it I almost fell off my chair laughing: our good friend the TimeLOCK trial remaining dialogue popped up. VCD uses TimeLOCK version 1. I assumed that it was pretty much the same as version 2 which I have already cracked so I decided to use HIEW and take a look through TLOCK32.DLL. What I was trying to do was look for the tell-tale blocks of code which generate the key (see my previous TimeLOCK tutorial). However, it took me a little longer than expected because this version of TimeLOCK doesn't generate the key in the same way (i.e. a call to a digit generating function) however I still found the code. What I did was to look at the right hand side of the HIEW screen for the jump references. I was looking for regularly spaced references and found them close to the beginning (17 pages down a 50 line screen):

```
.0000126F: 8B15C0F90010                mov    edx,[01000F9C0]
.00001275: 8A0D56FB0010                mov    cl,[01000FB56]
.0000127B: 8A4201                      mov    al,[edx][00001]
.0000127E: 3AC1                        cmp    al,cl
.00001280: 7C02                        jl     .000001284   ---------- (1)
.00001282: 8AC1                        mov    al,cl
.00001284: 8B15C0F90010                mov    edx,[01000F9C0]
.0000128A: 8A0D59FB0010                mov    cl,[01000FB59]
.00001290: 88442400                    mov    [esp][00000],al
.00001294: 8A4205                      mov    al,[edx][00005]
.00001297: 3AC1                        cmp    al,cl
.00001299: 7F02                        jg     .00000129D   ---------- (2)
.0000129B: 8AC1                        mov    al,cl
.0000129D: 8B15C0F90010                mov    edx,[01000F9C0]
.000012A3: 8A0D5AFB0010                mov    cl,[01000FB5A]
.000012A9: 88442401                    mov    [esp][00001],al
.000012AD: A052FB0010                  mov    al,[01000FB52]
.000012B2: 88442402                    mov    [esp][00002],al
.000012B6: 8A4202                      mov    al,[edx][00002]
.000012B9: 3AC1                        cmp    al,cl
.000012BB: 7F02                        jg     .0000012BF   ---------- (3)
.000012BD: 8AC1                        mov    al,cl
.000012BF: 8B15C0F90010                mov    edx,[01000F9C0]
.000012C5: 8A0D58FB0010                mov    cl,[01000FB58]
.000012CB: 88442403                    mov    [esp][00003],al
.000012CF: A055FB0010                  mov    al,[01000FB55]
.000012D4: 88442404                    mov    [esp][00004],al
.000012D8: 8A4203                      mov    al,[edx][00003]
.000012DB: 3AC1                        cmp    al,cl
.000012DD: 7C02                        jl     .0000012E1   ---------- (4)
.000012DF: 8AC1                        mov    al,cl
.000012E1: 8A0D5BFB0010                mov    cl,[01000FB5B]
.000012E7: 88442405                    mov    [esp][00005],al
.000012EB: A05AFB0010                  mov    al,[01000FB5A]
.000012F0: 88442406                    mov    [esp][00006],al
.000012F4: A057FB0010                  mov    al,[01000FB57]
.000012F9: 3AC1                        cmp    al,cl
.000012FB: 7F02                        jg     .0000012FF   ---------- (5)
.000012FD: 8AC1                        mov    al,cl
.000012FF: 8B4C2410                    mov    ecx,[esp][00010]
.00001303: 88442407                    mov    [esp][00007],al
.00001307: 8D442400                    lea    eax,[esp][00000]
.0000130B: 50                          push   eax
.0000130C: 51                          push   ecx
.0000130D: FF158C120110                call   lstrcmpA ;KERNEL32.dll
.00001313: 83F801                      cmp    eax,001
.00001316: 1BC0                        sbb    eax,eax
.00001318: 83C40C                      add    esp,00C
.0000131B: F7D8                        neg    eax
.0000131D: C3                          retn
```

Not quite the CALLs I were expecting but very suspicious nonetheless. I knew that TimeLOCK 2.0 had a trial reset key which was based on a similar scheme to the main registration key so I searched a bit further down for a similar set of instructions and found this:

```
.00001A9B: 8A4203                      mov    al,[edx][00003]
.00001A9E: 3AC1                        cmp    al,cl
```

```
.00001AA0: 7C02                     jl    .000001AA4   ---------- (1)
.00001AA2: 8AC1                     mov    al,cl
.00001AA4: 8B15C0F90010             mov    edx,[01000F9C0]
.00001AAA: 8A0D5BFB0010             mov    cl,[01000FB5B]
.00001AB0: 88442400                 mov    [esp][00000],al
.00001AB4: 8A4204                   mov    al,[edx][00004]
.00001AB7: 3AC1                     cmp    al,cl
.00001AB9: 7F02                     jg    .000001ABD   ---------- (2)
.00001ABB: 8AC1                     mov    al,cl
.00001ABD: 8B15C0F90010             mov    edx,[01000F9C0]
etc.
```

This looks very promising. Take another look at the code in blue in the first listing. Does it look familiar? At least it was improved slightly in version 2: I really don't believe that they used lstrcmpA to test the key. Well we can easily sniff the password using an embedded breakpoint and SoftICE. But we haven't used the live technique so far so let's not use it at all.

Taking a look at the code we can easily see that there are 3 locations: the serial we are given, the product code and the key (see my BoundsChecker tutorial). Looking at the serial generation scheme we can see that it is a simple mathematical manipulation. I summarise it as:

$$R_1 = \mathrm{Min}\ (V_4, S_{11})$$
$$R_2 = \mathrm{Max}\ (V_5, S_{12})$$
$$R_3 = S_{10}$$
$$R_4 = \mathrm{Max}\ (V_2, S_7)$$
$$R_5 = S_9$$
$$R_6 = \mathrm{Min}\ (V_6, S_8)$$
$$R_7 = S_6$$
$$R_8 = \mathrm{Max}\ (V_3, S_{12})$$

I hope the notation is clear. But how do we find the product code (V)? We could fish it out using SoftICE but we have decided to try without the live technique (sometimes, with such banal protection, you must make it more difficult yourself!). Let's cheat a little and use the key generator that we were given:

```
Virtual CD-ROM (TM) Version 1.0 Unlock Code Generator
Cracked & Coded by MegaByte of -=[ PhRoZeN CReW '96 ]=-
Enter Registration Number: 000000000000
The Unlock Code is: 03050007

Virtual CD-ROM (TM) Version 1.0 Unlock Code Generator
Cracked & Coded by MegaByte of -=[ PhRoZeN CReW '96 ]=-
Enter Registration Number: 999999999999
The Unlock Code is: 39999299
```

So we can deduce that:

$$V_2 = 5$$
$$V_3 = 7$$
$$V_4 = 3$$
$$V_5 = 3$$
$$V_6 = 2$$

We now have all we need to make a key generator, but we also have a rather good search string. Searching through VCDROM.EXE we find the full product code: 6573322. In fact if we search any TimeLOCK 1.0 protected EXE file for 'ini' we will find the code immediately after it. So instead of hardwiring this product code into our key generator we can make a generic TimeLOCK 1.0 key generator and allow the user to enter the relevant product code. Here is my first version (I lost my improved version but they are identical in functionality):

begin 644 tl1key.com

end

Now for the Moan...

Take a look at my key generator it's 396 bytes long and that's for the inefficient first version (you'll see later). Why do crackers still insist on writing bloated key generators? The PhRoZeN CReW keygen (VCDGEN.EXE) is 17530 bytes packed and ~35600 bytes unpacked. That's 45 times larger than my keygen (90 times larger if unpacked)!

Come on guys let's take some pride in our work. Cracks are exactly the type of program suited to assembly programming. They are very easy to code in assembly. Let's disassemble my crack:

W32Dasm 8.5 (cracked)

This is such a poor disassembler it didn't even distinguish between the data and code in this most basic of configurations.

Bubble Chamber (BETA release)

An old favourite of mine. I learned assembly from the output this program produced. In fact, the resemblance to the actual source is scary! The data is shown in a very nice way and was detected perfectly (and could be set manually if needed). Unfortunately, the source doesn't compile due to the way some of the instructions are shown.

IDA Pro 3.7 (registered)

Again disassembled nicely, the data output is arranged in a rather clumsy manner. I use to use an earlier version of IDA along with bubble chamber, and it drove me mad back then. IDA is powerful, but it's user interface is still as clumsy as it ever was. The cross-references are good and even show up my inefficient programming: the keygen is a modified version of my TimeLOCK 2 keygen and there is still 16 bytes allocated for the serial number. Also, the DOS calls are well documented, though any cracker worth his salt will know these from memory. However it is useful when skimming through disassembled files.

HIEW 6.65

Not a disassembler, but the perfect tool for analysing files such as my keygen. You can follow the flow of the program from the beginning to the end executing the instructions in your head. Hiew is the Swiss army knife of crackers... well, perhaps after DEBUG.EXE :-)

If you're not an assembly programmer, disassemble my code (it's not protected) and have a look. You can learn a lot from dissected code. Next time, make sure your crack is in assembly :-)

Competition

Re-write my keygen to make it as small as possible. There are 2 categories: the first is to leave all the banners and prompts intact. The second is to make it as small as possible without restrictions. Fame and glory to the winner!