Cracking With +Vip-Vop

Written by: +Vip-Vop Published by: Splatter Industries

Cracking CircuitMaker 6.0 Pro

(Download at : http://www.microcode.com)

After the program was installed, looking through the directory I noticed tl32v20.dll. Recognizing that as the timelock dll, I thought it would be a simple matter of getting the correct reg number from the dll, as you can do with most programs protected by t.l. But as I ran the program I noticed it wasn't the usual timelock nag screen, it was a msg box with the days remaining and there was no purchase option. Looking into the program further, I found it still uses timelock calls for the time limit and starting up and crap like that. Also, if you look at toolwnd.dll also in the same directory, you see calls like IsDemo and AboutToolWnd. So lets divide cracking this into 2 different parts. First, we will get rid of timelock, and second, look into the calls in toolwnd.dl

First - Getting rid of timelock. Well looking at timelock calls, there are interesting calls like "_getUserNumDaysLeft@4" (guess what that returns) and "getUserState" (state maybe meaning are we registered, not registered, etc). Well add tl32v20.dll to your winice.dat and restart your comp, or you can use Numega's soft-ice loader.start the loader, choose file, then load exports, the choose the tl32v20.dll in your circuit maker diretory. Ok the next part I happened upon by accident. I set a bpx on getuserstate, because earlier I had seen it was called right beofre the getusernumdaysleft, and when it broke I mistakingly though I was in the code for getusernumdays. Well looking at both of the calls you can tell they return their value in eax, and I thought I would try making it move the value 30 into eax, then have it return (30 hex = 48 decimal, I just wanted to see if it would accept 48 days instead of the usual 45 or less). So I assembled the very first instruction of the getuserstate call (sorry dont have code to put it if I opened up the dead listing of Circuit Maker in MS Word it would take forever to open). So anyway I changed the first instruction to "mov eax, 30" and then "ret 0004" (i knew to add the 0004 becuase scrolling down through the code for the call I saw it used that return instruction. So anyway I traced through the code after the call with F10, and noticed I had mistakingly changed it so it didnt even call getUserNumDaysLeft and I wasn't nagged by any screen, and even the save functions were enabled when I opened a circuit (accept they still dont work, more on that later). Well now I decided its time to look into all those toolwnd.dll calls. Oh yea so anyway with tl32v20.dll, search and replace the following bytes:

SEARCH: C3833D38100110007507B8FC
REPLACE: C3B830000000C20400909090

Part 2: looking into toolwnd.dll
Well naturally the first thing you want to check is the "IsDemo" call. Well I looked into it pretty deeply, and seeing things checking strings to FULL or LOCK etc, and seeing that a certain memory location should be 1, etc, Im pretty sure I got it cracked, but it made no noticable difference in the running of CircuitMaker, even though I saw it thought I was "FULL". But now lets look at "AboutToolWnd". Add toolwnd.dll to your exports section or use the symbol loader to load it, add a "bpx AboutToolWnd". now start circuit maker and choose about. You will pop into softice at the start of abouttool-wnd. Scroll down until you see this code, with a call to IsDemo:
* Reference To: Toolwnd.IsDemo

```
:10002D66 E8F5F4FFFF            call 10002260
:10002D6B 83C404                add esp, 00000004
:10002D6E 85C0                  test eax, eax

* Possible StringData Ref from Data Obj ->"DemoBox"

:10002D70 B884BE0010            mov eax, 1000BE84
:10002D75 7505                  jne 10002D7C

* Possible StringData Ref from Data Obj ->"AboutBox"

:10002D77 B878BE0010            mov eax, 1000BE78
```

But you dont see the strings DemoBox and AboutBox in softice, write down that address in s-ice and look at toolwnd.dll in W32dsm, so you see the code above. Well looking into the code, see that DemoBox is moved into eax, if eax wasn't 0 (if isdemo didnt return 0) aboutbox is moved into eax. But lets take an even close look at AboutToolWnd (there will be a lot of code next, but its needed):

```
Exported fn(): AboutToolWnd - Ord:0001h
:10002CE0 8B442408              mov eax, dword ptr [esp+08]
:10002CE4 53                    push ebx
:10002CE5 56                    push esi
:10002CE6 B9FFFFFFFF            mov ecx, FFFFFFFF
:10002CEB 57                    push edi
:10002CEC A350D60010            mov dword ptr [1000D650], eax
:10002CF1 8B7C2418              mov edi, dword ptr [esp+18]
:10002CF5 2BC0                  sub eax, eax
:10002CF7 F2                    repnz
:10002CF8 AE                    scasb
:10002CF9 F7D1                  not ecx
:10002CFB 2BF9                  sub edi, ecx
:10002CFD 8BD1                  mov edx, ecx
:10002CFF C1E902                shr ecx, 02
:10002D02 8BF7                  mov esi, edi
:10002D04 BFA0BC0010            mov edi, 1000BCA0
:10002D09 F3                    repz
:10002D0A A5                    movsd
:10002D0B 8BCA                  mov ecx, edx
:10002D0D 83E103                and ecx, 00000003
:10002D10 F3                    repz
:10002D11 A4                    movsb
:10002D12 803DCCB0001000        cmp byte ptr [1000B0CC], 00
:10002D19 7431                  je 10002D4C ;<--HERE! it jumps over the
                                  UserNameBox string!
:10002D1B 8B7C2410              mov edi, dword ptr [esp+10]
:10002D1F 6A00                  push 00000000
:10002D21 6860240010            push 10002460
:10002D26 8B1D30B00010          mov ebx, dword ptr [1000B030]
:10002D2C 57                    push edi

* Reference To: USER32.DialogBoxParamA, Ord:008Ah

:10002D2D 8B357CF30010          mov esi, dword ptr [1000F37C]

* Possible StringData Ref from Data Obj ->"UserNameBox"

:10002D33 688CBE0010            push 1000BE8C
:10002D38 53                    push ebx
:10002D39 FFD6                  call esi
:10002D3B 8B1D30B00010          mov ebx, dword ptr [1000B030]
:10002D41 83F802                cmp eax, 00000002
:10002D44 7516                  jne 10002D5C
:10002D46 33C0                  xor eax, eax
:10002D48 5F                    pop edi
:10002D49 5E                    pop esi
:10002D4A 5B                    pop ebx
:10002D4B C3                    ret
```

*** Referenced by a (U)nconditional or (C)onditional Jump at Address:**
|:10002D19(C)
|
:10002D4C 8B1D30B00010 mov ebx, dword ptr [1000B030]

*** Reference To: USER32.DialogBoxParamA, Ord:008Ah**

:10002D52 8B357CF30010 mov esi, dword ptr [1000F37C]
:10002D58 8B7C2410 mov edi, dword ptr [esp+10]

*** Referenced by a (U)nconditional or (C)onditional Jump at Address:**
|:10002D44(C)
|
:10002D5C 6A00 push 00000000
:10002D5E 68E02D0010 push 10002DE0
:10002D63 57 push edi
:10002D64 6A00 push 00000000
*** Reference To: Toolwnd.IsDemo**

:10002D66 E8F5F4FFFF call 10002260
:10002D6B 83C404 add esp, 00000004
:10002D6E 85C0 test eax, eax

*** Possible StringData Ref from Data Obj ->"DemoBox"**

:10002D70 B884BE0010 mov eax, 1000BE84
:10002D75 7505 jne 10002D7C

*** Possible StringData Ref from Data Obj ->"AboutBox"**

:10002D77 B878BE0010 mov eax, 1000BE78

Well the string UserNameBox sounded interesting,so now choose about again, and when you pop into softice for the AboutToolWnd call, press F10 till you reach this:
:10002D12 803DCCB0001000 cmp byte ptr [1000B0CC], 00
:10002D19 7431 je 10002D4C

Use the command "r fl z" to make it so you won't jump if it wants to jump. Now press ctrl-d. You will see a box asking you to enter your reg name and number. I entered "+VipVop" and "444333222". Press ok and a msg box says invalid number. Standard stuff here. so "bpx messageboxa". press ok again, you will pop into s-ice. press f12 to return to whoever called this msg box, and you will see the following:

```
:100026B1 68D8D50010          push 1000D5D8
:100026B6 E8C5FCFFFF          call 10002380
:100026BB 83C404              add esp, 00000004
:100026BE 85C0                test eax, eax
:100026C0 752A                jne 100026EC
:100026C2 6AFF                push FFFFFFFF

* Reference To: USER32.MessageBeep, Ord:0187h

:100026C4 FF1568F30010        Call dword ptr [1000F368]
:100026CA 6A00                push 00000000

* Possible StringData Ref from Data Obj ->"NOTE"

:100026CC 6854BD0010          push 1000BD54

* Possible StringData Ref from Data Obj ->"Invalid registration number."

:100026D1 6808BD0010          push 1000BD08
:100026D6 56                  push esi

* Reference To: USER32.MessageBoxA, Ord:0188h

:100026D7 FF158CF30010        Call dword ptr [1000F38C]
```

Well we see a call, and a jump over the msg box. The "push 1000D5D8" is prob the reg number we entered, the "call 10002380" is prob the reg check. Note that since there is only one push the program most likely doesnt use our name to compute our reg number. Now we have 2 options from here, patch the call or try to get a working reg number. Well we are feeling adventuresome today, aren't we :) ? So lets get that working reg number. So double click on that "push 1000D5D8" line in s-ice to set a bpx on it (if you need to get to be able to see that part of that code use the same bpx messageboxa and the F12, then ctrl-uparrow to scroll up. Anyway click ok again, and now when you are on that line. (note: due to the way dlls load, the line numbers and addresses pushed might not be the same as this dead listing, but you should be able to figure it out). Anyway when you are on the line with the push before the call, display whats being pushed ("d [address]"), and you will see its the reg number you entered. Now press F8 to trace into that call. heres the first code you will see: **(continued on Page 6)**

```
:10002380 53                push ebx
:10002381 B9FFFFFFFF          mov ecx, FFFFFFFF
:10002386 56                push esi
:10002387 2BC0               sub eax, eax
:10002389 8B74240C            mov esi, dword ptr [esp+0C]
:1000238D 57                push edi
:1000238E 8BFE               mov edi, esi
:10002390 F2                repnz
:10002391 AE                 scasb
:10002392 F7D1               not ecx
:10002394 49                dec ecx
:10002395 83F909             cmp ecx, 00000009 ;(ecx holds length of
                                 your reg number)
:10002398 7406               je 100023A0
:1000239A 33C0               xor eax, eax
:1000239C 5F                 pop edi
:1000239D 5E                 pop esi
:1000239E 5B                 pop ebx
:1000239F C3                 ret
```

well this code is pretty self-explanatory. Using softice, you can see esi holds your reg number, and your reg number should be 9 numbers long. So if yours isn't, eax is set to 0 (not regged), and you return. So go back and enter a 9 digit reg number (or if you used 444333222 you won't need to do that). Now lets look at the code you jump to if your reg number was 9 long:

```
:100023A0 803E31             cmp byte ptr [esi], 31
:100023A3 7406               je 100023AB
:100023A5 33C0               xor eax, eax
:100023A7 5F                 pop edi
:100023A8 5E                 pop esi
:100023A9 5B                 pop ebx
:100023AA C3                 ret

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
:100023A3(C)

:100023AB 8A4601             mov al, byte ptr [esi+01]
:100023AE 3C31               cmp al, 31
:100023B0 0F8C97000000         jl 1000244D
:100023B6 3C39               cmp al, 39
:100023B8 0F8F8F000000         jg 1000244D
:100023BE BF02000000           mov edi, 00000002
:100023C3 BB01000000           mov ebx, 00000001
```

well [esi] is the first letter of your reg number, and is compared to 31. Entering "? 31" in s-ice shows that 31 is the ASCII code for the number "1". So reviewing what we know about our reg number we know it should look like this: 1xxxxxxxx , where the x's our numbers we dont know.
Now at line 100023AB esi+01 is moved into al. esi+01 is the 2nd number of our reg number. ? 31 shows that 31 is "1", and ? 39 is "9". If you look at the jl 1000244D you see 1000244D isnt a good place to go, it makes eax 0 and returns. But see what the program does? it checks to see if the 2nd digit is less than 1, or more than 9, and if it is than you fail the reg check.
so now we know the format is: 1[1-9]xxxxxxx. So now scroll down for a while through code which doesn't really effect us.(note: since the code seems to loop a lot, you can do a "d esi". now see the location of esi (mine was 0058d5d8). do a "bpr 0058d5d8 0058d5df RW". bpr is breakpoint on range, and we entered those 2 address as our range, which is all of our reg number. So now any time toolwnd.dll accessing our reg number we will pop into softice.
so now you can press ctrl-d a couple of times to skip all that looping.
Eventually we will come to code like this:

```
:100023FE 8A4606          mov al, byte ptr [esi+06]
:10002401 3C30            cmp al, 30
:10002403 741C            je 10002421
:10002405 3C32            cmp al, 32
:10002407 7418            je 10002421
:10002409 3C34            cmp al, 34
:1000240B 7414            je 10002421
:1000240D 3C36            cmp al, 36
:1000240F 7410            je 10002421
:10002411 3C38            cmp al, 38
:10002413 740C            je 10002421
:10002415 33C0            xor eax, eax
:10002417 5F              pop edi
:10002418 5E              pop esi
:10002419 5B              pop ebx
:1000241A C3              ret
```

Esi+06 is the 7th number of our reg number. Looking at the code we see we want to follow one of those je 10002421 or eax will be made 0. well looking at 30,32,34,36,and 38 we see they are the ASCII values for 2,4,6, and 8.
So now our reg number format is: 1[1-9]xxxx[2,4,6,8]xx
Now lets look at 10002421: **(Continued on next page)**

:10002421 8A4607 mov al, byte ptr [esi+07]
:10002424 3C31 cmp al, 31
:10002426 7416 je 1000243E
:10002428 3C33 cmp al, 33
:1000242A 7412 je 1000243E
:1000242C 3C35 cmp al, 35
:1000242E 740E je 1000243E
:10002430 3C37 cmp al, 37
:10002432 740A je 1000243E
:10002434 3C39 cmp al, 39
:10002436 7406 je 1000243E
:10002438 33C0 xor eax, eax

esi+07 is the 8th reg number, and is compared to see if its 1,3,5,7 or 9.
By now you should be getting the hang of this. Now our reg number looks like this: 1[1-9]xxxx[2,4,6,8],[1,3,5,7,9],x
well lets look at this location 1000243E (it seems to be checking the numbers in order, so can probable assume it ignore whatever at the 3rd to 6th spot of our reg number, and as we find out later we are right).

:1000243E 8A4608 mov al, byte ptr [esi+08]
:10002441 5F pop edi
:10002442 2C37 sub al, 37
:10002444 5E pop esi
:10002445 5B pop ebx
:10002446 3C01 cmp al, 01
:10002448 1BC0 sbb eax, eax
:1000244A F7D8 neg eax
:1000244C C3 ret

Ok now this is the interesting part of the code. al holds the last number of our reg number. 37 is subtracted from al (remember AL holds the ascii code, so if your last number was 9 al would hold 39. Al is compared to 1 (why? i cant tell). Eax (which AL is part of) is sbb'ed (I have no idea what that does), and then the negative of eax is taken. well we want eax to be 1, so lets see what we can do about that. Put a breakpoing on line 1002446. lets enter our reg number as 123456219. When we get there we see after the sub al,37 al is 2, which means eax is also 2. sbb eax,eax makes eax 0, neg eax means eax is still 0, and eax is 0 when we return. Trying with 8 as the last digit the same thing happens. Now even though we dont understand how sbb works, we can see if eax is any number 1 or higher (and incidentally lower) it will make eax 0. but what if eax is 0 when it is sbb'ed. if the last number is the ASCII code 37, which is the number "7", al will be 0 when it is sbb'ed. Lets try that. Now we see eax is -1 after that sbb, and then the neg eax turns it into 1. So to review our reg code the format is: 1,[1-9],xxxx,[0,2,4,6,8],[1,3,5,7,9],7
so valid numbers could be 110000217 or anything like that. Now when we click about we see registered to and our name. But we still can't save. Looking through the program I didnt see code which lets me think you actually can save, although I plan on looking into it more.

This concludes another crack lession
with +Vip-Vop

Any questions or comments, please send to:

vipvop@hotmail.com or sii_@hotmail.com

You can find this and other tutorial by +Vip-Vop
at: http://www.splatter.net

Cracking With +Vip-Vop (Cracking CircuitMaker 6.0 Pro)
Was written by : +Vip-Vop
And published by : Splatter Industries,Inc.

The End