Portal World Domination Introduces:
The 'Finding The Best Route' Tutorial

The Possibilities are endless

Although the ability to find the best route to a crack is a skill needed for every crack you will do, this tutorial focuses on one very interesting program. In our first tutorial we presented the program InstallSHIELD Professional Evaluation which would stop running after Dec. 20 1996, now I give you InstallSHIELD Professional. The difference is in they type of check it performs and the ability of the second one to be "registered." This time the program gives you 30 days from the time you install it to evaluate and/or register it.

Run! Its a Dialog Box!

Yeah, I'm sure you've seen these before! Those little dialog boxes when you first start the program proclaiming your time left and begging you to register. If you press the "Register" button you are given a screen with 3 text boxes. If you press "Cancel" booted out of the program. And if you press "OK" you are given the right to try the the program, unless time has expired then you are booted once again. Where does this leave us?

First crack at it.

Now, if we take what we learned in the first of our tutorials and apply it here we would put a BPX on GetSystemTime. So lets do that. OK, step a while, perhaps try it with a date that works and compare what happens with that of a wrong date. Taking forever, eh? Plus even when you find out where the execution branches and "fix" it, you will still end up with a dialog box! and an broken program. It must check several times...

That was a disaster!

What should we do next? I know! Try to "register" it, yeah, crack the register dialog. Well go ahead, I'll be waiting. Give up yet? So that’s not very good.

This program can be attacked many more ways. For instance placing a BPX on the Registry functions or profile string functions. How about attacking it from the button press? Maybe a BPX on CreateDialog? But we are looking for the simplest, and easiest method.

The best method

Lets try something new! Go ahead and disassemble the program to see if we can, well, see something. First look through the imports hmmm tl32v20.dll?? What is that? Disassemble tl32v20.dll and look at the exports. "verifyTimeLock32?!" "trialEnvironmentOpen?!" We found something good!!!

WinICE to the rescue

Ahh yes just when you thought winice couldn't get any better it does! Remember when you had to remove the ';' before all those lines in the winice.dat file? What you were doing was telling winice to load the addresses of the exported functions from those DLLs into memory. Wouldn't it be nice if we could put a BPX on verifyTimeLock32? Well we can! But not now...first you have to copy tl32v20.dll into c:\ I put it in the root so I am sure winice can load it. Then edit WinICE.dat and add at the bottom:

•0	exp=c:\tl32v20.dll

save and then restart your computer, I'll wait...

What now brown m0m0?

Well now you can set a BPX on any of those functions so start by setting one on verifyTimeLock32 then running the program. When it breaks look at the bottom of the code window it tells you that you are in the DLL (tl32v20.dll) press F11 to step out and now your in the application (isx.exe). Look around what do you see in the code window? Probably this(LONG):

```
014F:004244DA    FF15ACBD4900       CALL  [tl32v20!verifyTimeLock32]
014F:004244E0    83F8FD             CMP   EAX, -03
014F:004244E3    747F               JZ    00424564
014F:004244E5    83F8FE             CMP   EAX, -02
014F:004244E8    747A               JZ    00424564
014F:004244EA    83F8FF             CMP   EAX, -01
```

```
014F:004244ED     7475              JZ     00424564
014F:004244EF     83F801            CMP    EAX, 01
014F:004244F2     7570              JNZ    00424564
014F:004244F4     6878FB4800        PUSH   0048FB78
014F:004244F9     8D45DC            LEA    EAX, [EBP-24]
```

then...

```
014F:004244FC     50                PUSH   EAX
014F:004244FD     E89E5F0200        CALL   0044A4A0
014F:00424502     83C408            ADD    ESP, 08
014F:00424505     85C0              TEST   EAX, EAX
014F:00424507     755B              JNZ    00424564
014F:00424509     6894FB4800        PUSH   0048FB94
014F:0042450E     FF15B0BD4900      CALL   [tl32v20!showMainDialog]
014F:00424514     83F8FB            CMP    EAX, -05
014F:00424517     744B              JZ     00424564
```

Ok, that's far enough for now(more later!). See all those CMP's and JMP's after verifyTimeLock? They test various error conditions and have nothing to do with the crack(though we will use them later) but why is this **00424564** address? Well use CTRL and the arrow keys to scroll down to it silly! It XOR's EAX by itself then does cleanup. When you XOR something by itself you always get 0. So it sets EAX to 0 then RET's.

Continue following the code and you see that it calls showMainDialog, that's the source of the dialog you see. After that are the tests again for error codes then 1 line that just pops out at you:

```
014F:00424523     3DBDDC0000        CMP    EAX, 0000DCBD
```

then...

```
014F:00424528     750F                     JNZ    00424539
014F:0042452A     FF15B4BD4900      CALL   [tl32v20!trialEnvironmentOpen]
014F:00424530     B801000000        MOV    EAX, 00000001
014F:00424535     8903                     MOV    [EBX], EAX
014F:00424537     EB2D                     JMP    00424566
014F:00424539     3DE6AA0100        CMP    EAX, 0001AAE6
014F:0042453E     750F                     JNZ    0042454F
```

You probably noticed all these MOV EAX, 00000001 instructions well when a CALL RETurns's it can place a value in EAX. That value will be checked by the calling function if and what errors occurred. In this program a value of 1 means success and a value of 0 means an error.   Every time a JMP goes to 00424564 it sets EAX to 0 and RET's we want EAX to be 1 and we need to do it before showMainDialog ('cause that's fucking annoying) so where? But first answer this question: Is trialEnvironmentOpen required for the program to run? Well if you look at the all the CMP's and JMP's you can find a path that does RET with 1 without ever calling trialEnvirnomentOpen. So the answer is an almost certain YES! But only testing can tell. So lets fix the program...

The smallest crack

Well now you know the goal of the crack but how do we do it? Well look through the code and I'll show you something interesting. Just after you break in verifyTimeLock32 find the lines that read:

```
CMP EAX, 01
JNZ 00424564
PUSH 0048FB78
```

now together those lines constitute 10 bytes of code. In winice, after the break on verifyTimeLock32, F10 down to the CMP line but don't execute it then type A and enter to 'Assemble' a new command. The command window now displays the address that you will overwrite. So at the first prompt type:

```
MOV EAX, 01 <enter>
JMP 00424566 <enter>    (remember 00424564 would XOR EAX, and an XOR)
                        (is 1 byte long, so we go to 00424566       )
NOP <enter>
```

NOP <enter>
NOP <enter>

Wrap it UP!

That’s the crack, you should know how to make a publishable crack using StarFury's tutorial and a hex editor. See how easy that was? Much easier than any of the others. We could have also removed the splash screen(a tad later) but I think we should always give the developers credit.

Remember that when an application uses a DLL for protection it is EXTREMELY easy to crack using this method.

Good luck, and keep cracking.

How to contact us!

We're always looking for new members who really want to learn! See any of the senior members or Head members for membership! We can help you learn. If your ever on Undernet stop by #PWD. Or look for our members scattered around the world. Here is a list:

First the two founding (Senior) Members and the best of our group:

- DiM
- StarFury

Then the Head Members, those of us that cracked something:

- _Lasher_ / Imajix
- B_Spline
- Kratz
- VoXeL

Finally those valuable Members that are there to help and learn:

- BigD_
- Cyah
- DrmWEaver
- EvilAngel
- Fouton
- Goa
- IcedFire
- Locote
- Maug
- |No_One|
- Quequog
- Slvrmoon
- TimbrWlf
- Tungsten
- Goa
- ^Arj

VoXeL