Table of Contents

Files used as an example

- BackupDVD - http://www.backupdvd.info/
- BrowserMaster - http://www.vasile.com/racecar/stampware/
- iOpus Password Recovery XP - http://www.iopus.com
- TrayDay - http://www.mjmsoft.com
- BackupMagic v1.3.1 - http://www.moonsoftware.com/cgi-bin/download.exe?bmagic

About

I have decided to write a tutorial about cracking because in doing this it helps me remember things that I might forget, can use it as a quick reference and also gives a chance for someone else to learn something. You have paid no money for this tutorial and I am not a professional write so don't take it too serious. This tutorial is for educational purposes only! I hold no responsibility of the misuse of this material.

Tools of the Trade

- SoftIce

In case you don't know what SoftIce is which I will reefer to as SICE from now on, it is a all purpose debugger that can debug virtually any type of code. This Software will be the most valuable tool you have, and most used.

- A good Disassembler

A disassembler in my opinion is the second most valuable tool you can have, a dissembler allows you to open a program and view its code in ASM. This can be most helpful as you can get a overview of the program your about to crack. The best two dissemblers I know off are IDA and W32Dasm.

- Hex editor

A good hex editor will be used plenty when cracking programs. Hiew and Hex workshop is the most well know hex editors.

- PE identifier

A PE (Portable Execution) identifier is a program which can tell you what a program has been build in, e.g. if a program has been coded in C++, ASM, Delphi or Visual Basic to name a few. It can all so tell you if the .exe has been packed.

- Regmon

Regmon is a program, which allows you to monitor the system's registry. It can show you what registry keys programs are using or trying to access.

- Filemon

Same as above, instead it shows what files programs are using or trying to access.

- ProcDump

ProcDump is a tool that allows you to Dump, unpack/decrypt some protected PE files without any need of a debugger. It can alter a given file PE header and restore the Import table and PE header

- ICEDUMP

ICEDUMP is a program that allows you to dump data in memory into files .It can also hide SICE and allow you to take screen shots of SICE among other things.

- FROGSICE

Same as above. Never used this program.

- SmartCheck

SmartCheck is a run-time debugging tool for VB (Visual Basic) programs. It gives detail program errors and detailed tracking and logging of program events.

- For Reference you will definitely need these when cracking, a ASM reference Manual, Win32 Programmer's Reference and a ASCII table

Getting Started

- First steps down a long road

When you got the tools listed above and more, make a new Folder on your HDD (Hard Disk Drive) called Cracking. In this Folder make three Folders, one for tutorials you read, one for tools and one for the programs you crack or keyGens you make. Now go through the tools you got and RTFM (Read the F**KING Manual) that came with each tool. This can save a lot of time in the long run.

- Assembly language

Lets face it the more ASM you know the quicker you can crack programs, in addition to understanding what the program is doing or trying to do. There is a great set of tutorials on ASM by Iczelion, I think some of these come with MASM, a compiler for ASM. Below is a set of general ASM functions that you will come across a lot. These are only a small percentage of what you should know, but you can crack small programs with BAD Protection (just reversing a jump or two) schemes just by knowing these.

Hex:	**ASM:**	**Meaning:**
75 or 0F85	Jne	Jump if not equal
74 or 0F84	Je	Jump if equal
EB	Jmp	Jump directly to
90	Nop	no operation
77 or 0F87	Ja	Jump if above
0F86	Jna	Jump if not above
0F83	Jae	Jump if above or equal
0F82	Jnae	Jump if not above or equal
0F82	Jb	Jump if below
0F83	Jnb	Jump if not below
0F86	Jbe	Jump if below or equal
0F87	Jnbe	Jump if not below or equal
0F8F	Jg	Jump if greater
0F8E	Jng	Jump if not greater
0F8D	Jge	Jump if greater or equal
0F8C	Jnge	Jump if not greater or equal
0F8C	Jl	Jump if less
0F8D	Jnl	Jump if not less
0F8E	Jle	Jump if less or equal
0F8F	Jnle	Jump if not less or equal

Having a good ASM reference Manual will help you out a lot, when you come across a function you don't understand, quickly look it up. You will need all off these when using SICE. Below is more ASM functions you will come across when cracking.

MOVS	Moves a (byte or word)
CMPS	Compare string (Byte or word)
CMP	Compares two numbers by subtracting the source from the destination and updates the flags.
MOVSX	This copies a source to a destination, used to preserve sign when copying from 8-bit to 16-bit or from 16-bit to 32-bit
MOVZX	Nearly the same as above but copies a source to destination operand and zero-extends it
SUB	Subtracts the source operand from the destination operand and stores the results in destination.
IMUL	Multiplies two unsigned integers (always positive)
MUL	Multiplies two signed integers (either positive or negative)
DIV	Divides two unsigned integers(always positive)
IDIV	Divides two signed integers (either positive or negative)
INC	Increments a register or a variable by 1
DEC	Decrements a register or a variable by 1

- Protection schemes

When it comes to cracking, programmers knows all the tricks used by crackers, they don't want their software cracked. They will pull out all the stops to try and make your life that bit harder.

1) Anti-SICE

Most programmers will but in some code that will stop the program running if a debugger is detected. The most common way of detecting a Debugger is (Ill use SICE as an example) to find certain files (VXD, DLL) that SICE or any another debugger uses when it's running. Then the program, if it finds any of these files will try to read/write to these, if it cant then a debugger is detected. You could patch the program, so it will run with a debugger. Look for two strings \\.\SICE or \\.\NTICE with you favorite hex editor. Change them to \\.\XXXX. You can also run one of the programs above ICEDUMP or FROGICE; these two programs will hide SICE among other things and let you run the program.

2) Anti-SmartCheck

If the program is coded in VB (Visual Basic) then you can use SmartCheck, Ill call it SC from now on. So you run SC and load the program, What! Messagebox saying "BYE MR CRACKER!" so the program knows you are running SC. This just makes me more determent to crack this program. So how does this program know that SC is running? There is two ways I know off that SC can be found.

- Find a DLL file SC is using
- Look at the Window ID / Task List

So it could be one of these two or something else. You could patch the program or patch SC. Lets try patching SC, open you favorite hex editor and look for strings "NuMega SmartCheck" and change them to what ever you want. Try run the program now! No good, damn! Let's try patching the program. Look for the string the Messagebox gives you .My example is "BYE MR CRACKER!" with you favorite hex editor. Get the Offset for that string. Disassemble the program and go to that offset, look above for any JUMPS that can be reversed. You could also use SICE, set a breakpoint on MessageboxA or MesageboxIndirectA and scroll up and set a breakpoint on any jumps, then run the program SICE breaks, reverse the jump before the Messagebox is called. It's just trial and error finding the right jump.

So you pass the programs anti-debugger protection, now its time to find the Serial Number. A Programmer uses many tricks to keep you from finding the real serial number.

1) Name Length
Yes you read it correctly. The programmer might decide that the name has to be a secretion length. I came across a program where the name had to be more than seventeen characters in length, Including spaces. This also applies to the serial number as well.

2) Serial style
The programmer might check the style of the serial number e.g.

SN-12345
SN-1234-67890
SN_1234
SN_1234-67890
1234-567890

It could be any style. Some programs start the serial number with the two starting letters of the name of the program. E.g. a program named CODE IT could start the serial number with CI-1234. If the serial number you typed in is not in the same style as it should be, then the real serial number will not be generated.

3) Fake serial number
The programmer might be that bit cleverer and try and confuse you with a fake serial number. You come across a calculating routine where the name you typed in is used to produce a serial number. You think you hit the jackpot. So you exited SICE type in the serial number you have, get a Thank you Messagebox, restart the program and you get a Messagebox telling you that the program is PIRATED or something. This is all well and good you can go about trying to find the real serial number. What if the program waits a day then stops working. This is really clever, you think its just something wrong with the program. Re-install it, a day later it stops working again. You will only find this out if you use the program. Chances are once cracked, you'll never used the program again. Just think about it.

4) Serial Number is different for each Computer
I hate programs that do this, give you a Special ID number or something, this is usually generated from the user name logged in. Then the serial number is generated from the ID number you are given.

5) A compare or two
You come across a compare (CMP), great its comparing the serial number you typed in with some other string/number. What's this another compare, and another using your serial number? These are more than likely fake compares, or maybe not, the program could have generated more than one serial number for your name. This just means you have to be carefully of your choice of which way to go. You think you have found the real serial number type it in, no good? If the serial number is digits, covert it to hex. Then try it.

- Packers

Packers are used to make an EXE file smaller as well as some putting in ant-SICE code and encrypting the EXE file, which protects the code and stops you from disassembly/decompling the program. Example of how a packer might work:

DECYPT CODE A
ENCRYPT CODE A
DECYPT CODE B
ENCRYPT CODE B

This means that most of the code is encrypted in memory. Every file packed/encrypted can also be unpacked/decrypted. There are loads of good tutorials out there on how to unpack EXE files. Here is a small list of packers UPX, ASPACK, PEPACK, SHRINKER, NEOLITE. How do you know if the file is packed? Well see if you can disassemble the EXE or use a PE-identifier. It will read the PE headers/ Look for secreting signatures and be able to tell you what the EXE is packed with. Some will even unpack the EXE for you. You could also use ProcDump to unpack the EXE. You should know how to use ProcDump, as I told you RTFM. ICEDUMP can also be used; you need to know where the Call is to decrypt the code and where the code is dumped in memory. Manual unpacking is sometimes really hard, depends on the packer. There are loads of great programs for unpacking secretion packers.

- HEX

Why learn about HEX? When cracking you are dealing with hex. When you open a file with a hex editor most of the hex you see is ASM instructions. Been able to understand the base 16 system is a bonus.

Hex is a number system base 16, maybe one of the reasons we use DECIMAL system base 10 is because we have ten fingers. Hex makes really large numbers much smaller and which is easier to graph to memory chips than decimal. Hex is based on 16 digits which are 0-9 then A=10,B=11,C=12,D=13,E=14,F=15.

1) Converting decimal into binary

It is much easier that you might think Let's take the number 100 as example. First thing to do is covert the number into BINARY, you know all them zeros and ones. You do this by dividing the number by two, write down the remainder, then divide your answer by two until your remainder is zero.

Divide Number	Answer	Remainder
100 / 2	50	0
50 / 2	25	0
25 / 2	12	1
12 / 2	6	0
6 / 2	3	0
3 / 2	1	1
1 /2	0	1

The answer is the remainders in reverse order, 1100100. You can leave the zeros out at the end, so 11001 is the same number. What if you have a number like this to covert FE, just start with F= 15 then E=14.

2) Converting binary to decimal
Next thing to do is convert from Binary to Decimal. This is easy just place your binary number over numbers doubled starting from one.

1	1	0	0	1	0	0
64	32	16	8	4	2	1

Wherever there is a one above the number add this to the rest of the numbers with one above them. I know it easy but hard to explain. So you will end up with, 64+32+4 = 100 Let's take another example just to make sure you understand. Binary number: 10110111

1	0	1	1	0	1	1	1	0
256	128	64	32	16	8	4	2	1

You will end up with 256+64+32+8+4+2 = 366. Now you have the trick.

3) Converting from Binary to Hex
Now to convert from binary to hex, First thing to do is multiple each digit by 2 raised to the number of numbers from the right it is Lets take 1100100 as a example .So group the digits into pairs of four. You can but a extra zero at the start of the binary number to make it even. IF you don't know what ^ means it's the same as XOR. You will come across this in ASM.
0 1 1 0
2^3 2^2 2^1 2^0
2*2 = 4 and 2*1 = 2. 4+ 2 = 6. Answer = 6

0 1 0 0
2^3 2^2 2^1 2^0
2*2 = 4. Answer = 4

But the two answers together and you get 64. That's you're binary number in hex. Now, test your self by converting numbers in hex to binary and then to decimal. Just to make sure you got the hang of it. It easy to covert one base system to another, but when working with hex it is best <u>not</u> to do calculations like add, multiply etc., yourself mistakes are two easily made. Use a calculator.

- SoftIce (SICE)

SoftIce is an all-purpose debugger that can debug virtually any type of code. This Software will be the most valuable tool you have, and most used, or in our case it is used to get a look at what a program is doing.

Getting SICE ready

When installing SICE you can get the installer to change your AUTOEXE.BAT file for you. This insures that SICE will load on startup. You could change the AUTOEXE.BAT file your self by adding the following line <drive>:\<dir>\WINICE.EXE. By adding this line to your startup will start SICE and automatically launch Windows. After doing this you most reboot.

Once SICE/Windows is loaded you can by pressing CTRL+D at any time break to SoftIce. To return from breaking press CTRL+D (or F5). Another important thing to note is that in WINICE.DAT are some lines to include symbols. There is most likely a ; in front of some of these lines: If you see a ; you should remove it as these are the most common DLL's to use. Only do this if you have the file. If you change anything in the WINICE.DAT file you most reboot .Also you should not remove the ; if you don't have the file!

Once all that is done press CTRL+D to break to SICE, now press CTRL+F1, that's better. Lets see what symbols are loaded do this by typing EXP <RETURN>. You should see a list of symbols.

Most used SICE commands

1) Step Into (Press F8) - step into a call
3) Step over (Press F10) - step over a call
4) Return to a Call (F11) – return to a call
5) Step Out (Press F12) - step out of a call
6) Modifying a Register - R EAX FFFFFF
7) Reversing a FLAG - R FL Z (When on a JUMP)

Setting Breakpoints

Just incase you don't know what a breakpoint is, ill tell you. A breakpoint lets you break at a secretion point in a program. Example if a program gives you a Message box telling you typed a wrong serial number. You could set a breakpoint before you get the message box. Knowing which breakpoint to set takes a bit of time, and practice.

There are a few types of breakpoints you can set,

1) BPX <breakpoint on APIs/breakpoint you want> e.g. BPX HMEMCPY
2) BPM <Set a breakpoint for memory address> e.g. BPX 4A0950 RW (The RW is Read/Write)
3) BMR <Set a breakpoint for memory range> e.g. BPR 4A0950 4A0950+4 RW

- BPX is breakpoint on execute
- BPM is breakpoint on memory address
- BMR is breakpoint memory range, this is used when in memory more than one place is going to be RW to in a range.

Searching for a string/hex with SICE

Searching for a string/hex in SICE is quite easy.
At the SICE prompt type S 0 L FFFFFFF “STRING TO SEARCH FOR” you can do the same with hex

The S means the start
The 0 means start at address zero (Of course you can change the range)
The L means the length
After L is the end of the range

For VB (Visual Basic) programs you have to include spaces, VB uses wide characters.
E.g. S 0 L FFFFFFF “S”00”T”00”R”00”I”00”N”00”G” (The 00s are the spaces)

Other commands you will use often
? : With this you can see what’s in side a register e.g. ? EAX
D, DB, DW, DD, DS, DL, and DT: Display memory e.g. DD EAX
E, EB, EW, ED, ES, EL, and ET: Edit memory

Below is a list of the most used API’s. API (Application Programming Interface) is something like a procedure or function. You call the API (or function) and give it some parameters for input. It’s just like calling a function in C.

API's that put a nag-screen on the screen	
MessageBox(A)	Function creates, displays, and operates a message box.
MessageBoxExA	Function creates, displays, and operates a message box
MessageBeep	Function makes a BEEP
EnableWindow	Function enables or disables mouse and keyboard input to the specified window or control.
UpdateWindow	Function updates the client area of the specified window
CreateWindow	Function creates an overlapped, pop-up, or child window
CreateWindowExA	Function creates an overlapped, pop-up, or child window with an style; otherwise, this function is identical to the CreateWindow function
DialogBoxParamA	Display a DialogBox on the screen
ShowWindow	Display a DialogBox on the screen
SendMessage	Function sends the specified message to a window or windows

API's that are used to get text from the screen	
GetDlgItemText(A)	Retrieves the title or text associated with a control in a dialog box.
GetWindowText(A)	Copies the text of the specified window's title bar into a buffer.
GetDlgItemInt	Translates the text of a specified control in a dialog box into an integer value
Hmemcpy	Makes a copy of data that it reads from a textbox,
BozosLiveHere	32-bit version of Hmemcpy

API's that CD protections use	
GetDriveTypeA	Function determines whether a disk drive is a removable, fixed, CD-ROM, RAM disk, or network drive
GetLogicalDrives	Function returns a bitmask representing the currently available disk drives
FindFirstFileA	Function searches a directory for a file whose name matches the specified filename.
GetVolumeInformationA	Function returns information about a file system and volume whose root directory is specified
GetLogicalDriveStrings	Function fills a buffer with strings that specify valid drives in the system
GetDiskFreeSpace	Function retrieves information about the specified disk, including the amount of free space on the disk
GetFileAttributesA	Function returns attributes for a specified file or directory

API's that programs use to access the registry	
RegCreateKey(A)	Function creates the specified key
RegDeleteKey(A)	Function deletes a key and all its descendants
RegQueryValue(A)	Function retrieves the value associated with the unnamed value for a specified key in the registry
RegOpenKey(A)	Function opens the specified key
RegCloseKey(A)	Function releases the handle of the specified key

API's that are used to read/write from plain text files	
GetPrivateProfileString(A)	Retrieves a string from the specified section in an initialization file
WritePrivateProfileString(A)	Copies a string into the specified section of the specified initialization file
WriteProfileString	Copies a string into the specified section of the WIN.INI file

API's that have something to do with time	
GetSystemTime	Function retrieves the current system date and time
GetLocalTime	Function retrieves the current local date and time
SystemTimeToFileTime	Function converts a system time to a file time

API's that keyfiles protections use	
CreateFileA	The CreateFile function creates or opens files and returns a handle that can be used to access the file
_lopen	16-bit version of CreatefileA
ReadFile	The ReadFile function reads data from a file
_lread	16-bit version of ReadFile
SetFilePointer	Function moves the file pointer of an open file

Visual Basic APIs.
There are two many to list, so at SICE type EXP __ yes that's two underscores, to get a list of APIs. Below are the most important ones.

Compare APIs	
__vbaStrCmp	Very important, compares two strings.
__vbaStrTextCmp	Very important, compares two strings
__vbaVarCmpEq	Compare Equal
__vbaVarCmpGe	Compare Greater/Equal
__vbaVarCmpGt	Compare Greater Then
__vbaVarCmpLe	Compare Lower Equal
__vbaVarCmpLt	Compare Lower Then
__vbaVarCmpNe	Compare Not Equal

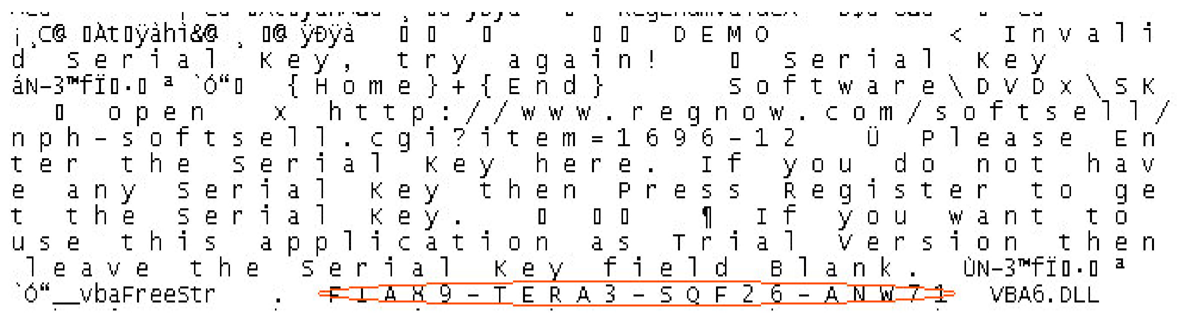
- Get Cracking

Cracking you first program in less than one minute

Well we have done a lot of theory work, time to but it all into practice. The first program we are going to crack is `Backup DVD v1.8a` for `Windows 95/98/Me/NT/2000/XP.`

`About` program
BackupDVD 1.8a allows you to convert DVD to VCD2.0/SVCD1.0/AVI in one step (Including multiplexing, splitting). It produces good quality movies in AVI/MPEG1/2 format and you don't need to have 5Gb free on your hard disk

Open the DIR for BackupDVD and look around. We see that there is two EXE files BackupDVD.exe and BackupDVDSK.exe. Let's run BackupDVDSK.exe, The SK at the end of it wouldn't mean SerialKey? Run It. It tells us Please Enter Serial Key Here..CUT...If you want to use this application as <u>TRIAL VERSION</u> then leave the Serial Key field Blank. Lets find out what the program is coded in, get you PE identifier out. It's coded in VB and its not packed, lets open it up with a hex editor. With VB programs you can see what strings the program might use. We see the string " Please Enter Serial Key Here..CUT...If you want to use this application as <u>TRIAL VERSION</u> then leave the Serial Key field Blank" and right beside this string is, the serial number in wide character format. I all so know that this serial number works in newer version of BackupDVD.



```
¡¸C@ ▯At▯ÿàhì&@ ¸ ▯@ ÿÐÿà   ▯ ▯   ▯      ▯ ▯  D E M O          <  I n v a l i
d  S e r i a l  K e y ,  t r y  a g a i n !   ▯  S e r i a l  K e y
áN-3™fÏ▯·▯ ª `Ó“▯   { H o m e } + { E n d }      S o f t w a r e \ D V D x \ S K
  ▯  o p e n    x  h t t p : / / w w w . r e g n o w . c o m / s o f t s e l l /
n p h - s o f t s e l l . c g i ? i t e m = 1 6 9 6 - 1 2    Ü  P l e a s e  E n
t e r  t h e  S e r i a l  K e y  h e r e .  I f  y o u  d o  n o t  h a v
e  a n y  S e r i a l  K e y  t h e n  P r e s s  R e g i s t e r  t o  g e
t  t h e  S e r i a l  K e y .   ▯  ▯ ▯   ¶  I f  y o u  w a n t  t o
u s e  t h i s  a p p l i c a t i o n  a s  T r i a l  V e r s i o n  t h e n
 l e a v e  t h e  S e r i a l  K e y  f i e l d  B l a n k .  ÙN-3™fÏ▯·▯ ª
`Ó“__vbaFreeStr    .  F I A X 9 - T E R A 3 - S Q F 2 6 - A N W 7 1   VBA6.DLL
```

This should have giving you a taste for cracking, but all programs are not as easy as this one. In fact this is the easiest program I have ever come across for getting a serial number. BackupDVD is a very good program (Just bad protection scheme) that deserves your money if you use it. Any legal issues arising from the misuse of this information are not the writer's responsibilities.

- Patching

Before we start patching programs, you have to know how to reverse jumps and change code with a hex editor. The best hex editor for patching is HVIEW, it allows you to decode the file into Text/HEX/ASM. Load your program into W32asm and then highlight the jump you want to change. Then look at the bottom task bar and look for the OFFSET. Load you copy of the program with HVIEW then press F4 3 times to decode it into ASM. Then press F5 and type the OFFSET W32asm gave you. Then press F3 and then the <TAB> key. Change the code to what you want and press F9 to update and then F10 to exit.

The second program we are going to crack is BrowserMaster - Version 2.0. We are going to crack this program by patching it.

About program
BrowserMaster, the successor to BrowserSizer, is a web development tool, made by web developers, for web developers. (Kind of cliche, huh?) It is a low-fat, NOT in-your-face, tool to help you easily perform very common and tedious tasks required in the development of professional web sites. Time-consuming tasks such as testing for low screen resolutions or low color count. Designed to be used throughout the development process, not just upon completion, BrowserMaster is the Professional Web Developer's helper

I am just quickly going to list what you should do
1) Open DIR for BrowserMaster (Look for any EXE files)
2) Scan BrowserMaster.exe with a PE identifier (Find out if packed/what it is coded in)
3) Run the Program

Information about program
1) Its written in Borland C++
2) Protection scheme Name/E-mail/Serial No
3) Protection scheme two Nag Screen

When we run the program we see a nag screen with the OK button disabled (A bug). The Enter License button is enabled, click on it and fill in the textboxes. Then click the OK button. We then get a messagebox telling us "You entered an invalid serial - retry?" Remember this or write it down.

Let's disassemble (W32Dasm or IDA) the program and get a look under the hood. Make a copy of BrowserMaster and disassemble it. When you have the program disassembled click on REFs then click on SDR (String Data References). Look for the string the messagebox gave you when you entered the wrong serial No. Found it good, click on it, and click on it again to make sure its not used anywhere else in the program.

```
* Possible StringData Ref from Data Obj ->"You entered an invalid serial "
                                        ->"- retry?"
|
:00428267 BACBDC4C00mov     edx, 004CDCCB     ← we land here
:0042826C 8B00              mov eax, dword ptr [eax]
:0042826E E8758D0800        call 004B0FE8
:00428273 48                dec eax
:00428274 7504              jne 0042827A
:00428276 B001              mov al, 01
:00428278 EB02              jmp 0042827C
```

Scroll up a bit till you see what calls the string

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00428244(C)  ← This is the jump that calls this routine to give us the message box
|
:0042825B A130744D00              mov eax, dword ptr [004D7430]
00428260 6A01                     push 00000001


* Possible StringData Ref from Data Obj ->"Invalid Entry"
|
:00428262 B9F2DC4C00              mov ecx, 004CDCF2
```

Right click on this call 00428244(C) and you will land

```
* Reference To: BrowserMaster.TRacecarRegister::DoRequirementCheck(())
|
:0042823D E87EF8FFFF              call 00427AC0
:00428242 84C0                    test al, al
:00428244 7415                    je 0042825B ←this is our jump
:00428246 8B4DF0                  mov ecx, dword ptr [ebp-10]
:00428249 51                      push ecx
:0042824A 8B4DF4                  mov ecx, dword ptr [ebp-0C]
:0042824D 8B55F8                  mov edx, dword ptr [ebp-08]
:00428250 8BC6                    mov eax, esi

* Reference To: BrowserMaster.TRacecarRegister::ValidateSerial(())
|
:00428252 E8A1060000              call 004288F8 ← call to validate serial No
:00428257 84C0                    test al, al
:00428259 7525                    jne 00428280
```

We could patch the jump at code address 00428244, but what if the program checks the serial number at start up (moist do). This would mean we are back at the start, program still unregistered. Lets look around a bit more. If you see below there is a Reference to ValidateSerial(()). This is real interesting. But we are not going to get this far because above is a reference to DoRequirementCheck(()) this means (guessing by the name) that the serial number is in a certain style. Let's go to the PEP (Program Entry Point) and look for anything interesting. Press PGDOWN a few times, we see a few APIs and a few URLs.

This is interesting we can see (After 3 PGDOWN) a String Data Ref. for dataobj "BMSTR-" and below this we see "ValidateRegistration(VOid())" this is the routine for validating the serial number at startup.

```
* Reference To: BrowserMaster.TRacecarRegister::ValidateRegistration(void())
|
:004015C8 E8FF650200              call 00427BCC ← Call to check serial No
:004015CD 84C0                    test al, al ← Flags
:004015CF 0F840F010000            je 004016E4 ← This is the jump that decides if we are registered
:004015D5 66C78548FFFFFF4400      mov word ptr [ebp+FFFFFF48], 0044
:004015DE C60594784D0001          mov byte ptr [004D7894], 01
```

What about reversing the jump at code location 004015CF from JE to JNE, this is all well and good but what if we have the right serial number? Then we will be unregistered. Make sense? Let's change it to a jump instead. Run the program and No nag Screen! Great! Go to about, no name or anything. We could find out where BrowserMaster keeps the users information by using RegMon and FileMon.

Do you remember the String Data Ref. for dataobj "BMSTR-"? Want to know what this is used for? Well ill tell you BMSTR- is the start of serial numbers for this program. This is the easiest program I came across for patching. Don't worry it will get harder. The Programmer named the ROUTINES, which made it easier for us to find out where the serial number is checked. BrowserMaster is a very good program (Just bad protection scheme) that deserves your money if you use it. Any legal issues arising from the misuse of this information are not the writer's responsibilities

The next program we are going to patch is iOpus Password Recovery XP v4.00

About program
This new version the popular iOpus Password Recovery software allows users to decrypt and display passwords stored behind the asterisks on Windows XP systems (as well as all other Windows versions). iOpus Password Recovery XP is the first tool of its kind that works automatically on Web pages, Windows 9x/NT and Windows 2000/XP systems. Windows systems allow a convenient storage of frequently used passwords, such as the password of your dial-in ISP or FTP connection. However, since you no longer enter the saved password manually, you tend to forget them. What are you to do when you need to know one of the saved passwords? The password is staring right at you, but is hiding behind a row of "*****" asterisks.

You should know what to do by now.
1) Open DIR for iOpus Password Recovery (Look for any EXE files)
2) Scan iOpus.exe with a PE identifier (Find out if packed/what it is coded in)
3) Run the Program

Information about program
1) Microsoft Visual C++ 6.0
2) Protection scheme String telling you not Trial version
3) Protection scheme two Just shows first three characters of password

Let's disassemble the program and get a look under the hood. Make a copy of iOpus and disassemble it. When you have the program disassembled click on REFs then click on SDR (String Data References). Look for the string "<= First 3 characters!" when you highlight a password box this is showing. Found it good, click on it, and click on it again to make sure its not used anywhere else in the program. You will land at code location 00404237(C) left click on it.

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00404237(C) ← left click here
|
:00404313 51                     push ecx
:00404314 8BCC                   mov ecx, esp
:00404316 8965E0                 mov dword ptr [ebp-20], esp

* Possible StringData Ref from Data Obj ->"  <= First 3 characters!"
|
:00404319 6860784300             push 00437860 ← we land here
:0040431E E862DA0100             call 00421D85
:00404323 51                     push ecx
:00404324 C645FC05               mov [ebp-04], 05
:00404328 8BCC                   mov ecx, esp
:0040432A 8965E4                 mov dword ptr [ebp-1C], esp
```

After left clicking on 00404237(C)

```
:00404223 E8083B0000            call 00407D30
:00404228 84C0                  test al, al
:0040422A 0F856D010000          jne 0040439D
:00404230 A012C24300            mov al, byte ptr [0043C212]
:00404235 84C0                  test al, al
:00404237 0F85D6000000          jne 00404313 ← the jump that stops showing all characters
:0040423D 6A44                  push 00000044
:0040423F 51                    push ecx
:00404240 8BCC                  mov ecx, esp
```

We now reverse the jump from JNE to JE. Try the program out, it still only shows us the first three letters of our password. We could NOP it. NOP means no operation. This should be only done if there is no other way. Looking at the above code we could trick about with the flags and making sure that the JNE is never taking. We could do this by changing the line at 00404230 from mov al, byte ptr [0043C212] to mov al,0001, by moving 1 (1 =True 0 =False) into AL the jump on line 404237 will never be taking. Try the program now. It works. But we still have that String [Trial Version]. Ill leave you to change this. iOpus Password Recovery XP is a very good program (Just bad protection scheme) that deserves your money if you use it. Any legal issues arising from the misuse of this information are not the writer's responsibilities.

- Serial Fishing

First thing to do when serial fishing, is make a copy of the program. Scan it with you PE identifier, if not packed, disassemble it. If the program is packed, unpack it. Sometimes you can still find the serial number with out unpacking it, it depends on the packer. Run the program and enter in any user name serial No. Write down the message you are given (if any) and look for it in the SDR. Find the jump that brings you here, and write the code location.

Next thing to do is look at the Functions --> Imports to see what APIs are used. Look for things like getdlgitemtext(A), getwindowtext(A), getwindowtextLenght(A), something to but a BP (Break point) on.

Start the program again and enter a Name/Serial No. Then Break SICE (CRTL-D) and set a breakpoint. Hit the button to Register and SICE should break if you have set the right BP (This just takes partice knowing which BPS to set) and now you can begin tracing.

Serial Fishing for TrayDay v6.21, for Windows 95, NT4

Description
TrayDay - date, calendar and counter utility for the Windows 95, 98 or NT 4 tray. TrayDay places an icon showing the date (day of the month) in the system tray (the part of the task bar which includes the clock). The icon resembles a page of a "tear-off" calendar.

You should know what to do by now. I'm not going to tell you which BP to set its one of the most common ones used to get text from Dialog Boxes. I'm going to enter "heywire" as name and "9876543211" as serial Number.

Hit: When picking your serial number don't use something like 123456.When you pick the serial number your going to use make sure you know what it looks like in HEX, so if the registers change you can spot you serial number. The serial number I use all the time is 9876543211, which is 4CB016EB in hex.

Hit the OK button and SICE will break. Press F5, you press F5 to allow the program to call the API to get you serial Number. The first time SICE breaks the program is getting you name. So we land in the API

USER.GETDLGITEMTEXTA (if you hadn't that BP set that one) so press F11 to go to the call, you should land in the TrayDay code. If not press F12 to go to the next RET till you land in the Trayday code. Keep tracing till you land here (Note you can reverse the JUMP with R FL Z) so you can find out what the next number is for the serial No.

```
:0040898F A067A14100          mov al, byte ptr [0041A167] <--Moves byte of your S/N
:00408994 83F838              cmp eax, 00000038 <--Cmp eax with 38 (dec = 58, ascii = 8)
:00408997 0F8510000000        jne 004089AD <-- bad jump
:0040899D 33C0                xor eax, eax
:0040899F A063A14100          mov al, byte ptr [0041A163]
:004089A4 83F82D              cmp eax, 0000002D <-- Cmp eax with 2D = "-"
:004089A7 0F8407000000        je 004089B4
```

Keep tracing, and you will see the strings below being moved in the data window, the program checks them against your name, most be a list of crackers or groups black listed. "MDC#", "Mfpd", "Lpuw", "Csqz", "VfIh", "BviK", "Fvno".

Keep tracing till you land here:

```
:00408A84 8A0D60A14100        mov cl, byte ptr [0041A160]
:00408A8A BB0A000000          mov ebx, 0000000A
:00408A8F 8B45FC              mov eax, dword ptr [ebp-04]
:00408A92 99                  cdq
:00408A93 F7FB                idiv ebx
:00408A95 83C030              add eax, 00000030
:00408A98 3BC8                cmp ecx, eax <-- another check for S/N
:00408A9A 0F851E000000        jne 00408ABE
:00408AA0 33C9                xor ecx, ecx
:00408AA2 8A0D61A14100        mov cl, byte ptr [0041A161]
:00408AA8 BB0A000000          mov ebx, 0000000A
:00408AAD 8B45FC              mov eax, dword ptr [ebp-04]
:00408AB0 99                  cdq
:00408AB1 F7FB                idiv ebx
:00408AB3 8D4230              lea eax, dword ptr [edx+30]
:00408AB6 3BC8                cmp ecx, eax <-- another check for S/N
:00408AB8 0F8407000000        je 00408AC5

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00408A9A(C)
|
:00408ABE 33C0                xor eax, eax
:00408AC0 E9B2000000          jmp 00408B77

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00408AB8(C)
|
: 00408AC5 B926000000         mov ecx, 00000026
:00408ACA 8B45D4              mov eax, dword ptr [ebp-2C]
:00408ACD 99                  cdq
:00408ACE F7F9                idiv ecx
:00408AD0 8D420A              lea eax, dword ptr [edx+0A]
:00408AD3 8945FC              mov dword ptr [ebp-04], eax
:00408AD6 33C9                xor ecx, ecx
:00408AD8 8A0D62A14100        mov cl, byte ptr [0041A162]
```

```
:00408ADE BB0A000000            mov ebx, 0000000A
:00408AE3 8B45FC                mov eax, dword ptr [ebp-04]

:00408AE6 99                    cdq
:00408AE7 F7FB                  idiv ebx
:00408AE9 83C030                add eax, 00000030
:00408AEC 3BC8                  cmp ecx, eax <-- another check for S/N
:00408AEE 0F851E000000          jne 00408B12
:00408AF4 33C9                  xor ecx, ecx
:00408AF6 8A0D64A14100          mov cl, byte ptr [0041A164]
:00408AFC BB0A000000            mov ebx, 0000000A
:00408B01 8B45FC                mov eax, dword ptr [ebp-04]
:00408B04 99                    cdq
:00408B05 F7FB                  idiv ebx
:00408B07 8D4230                lea eax, dword ptr [edx+30]
:00408B0A 3BC8                  cmp ecx, eax
:00408B0C 0F8407000000          je 00408B19


* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00408AEE(C)
|
:00408B12 33C0                  xor eax, eax
:00408B14 E95E000000            jmp 00408B77


* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00408B0C(C)
|
:00408B19 B911000000            mov ecx, 00000011
:00408B1E 8B45D4                mov eax, dword ptr [ebp-2C]
:00408B21 99                    cdq
:00408B22 F7F9                  idiv ecx
:00408B24 8D420A                lea eax, dword ptr [edx+0A]
:00408B27 8945FC                mov dword ptr [ebp-04], eax
:00408B2A 33C9                  xor ecx, ecx
:00408B2C 8A0D65A14100          mov cl, byte ptr [0041A165]
:00408B32 BB0A000000            mov ebx, 0000000A
:00408B37 8B45FC                mov eax, dword ptr [ebp-04]
:00408B3A 99                    cdq
:00408B3B F7FB                  idiv ebx
:00408B3D 83C030                add eax, 00000030
:00408B40 3BC8                  cmp ecx, eax <-- another check for S/N
:00408B42 0F851E000000          jne 00408B66
:00408B48 33C9                  xor ecx, ecx
:00408B4A 8A0D66A14100          mov cl, byte ptr [0041A166]
:00408B50 BB0A000000            mov ebx, 0000000A
:00408B55 8B45FC                mov eax, dword ptr [ebp-04]
:00408B58 99                    Cdq
:00408B59 F7FB                  idiv ebx
:00408B5B 8D4230                lea eax, dword ptr [edx+30]
:00408B5E 3BC8                  cmp ecx, eax <-- another check for S/N
:00408B60 0F8407000000          je 00408B6D
```

As you should know by now the serial Number has to be a length of seven. Fourth charter has to be "-". Enter in your name Serial Number and your registered. If you get a dialog box saying update register number or something like that change the date in the registry where the Serial Number is. TrayDay is a very good program (Just bad protection scheme) that deserves your money if you use it. Any legal issues arising from the misuse of this information are not the writer's responsibilities.

- License Keys

A license key is a file that contains an array of bytes inside a small file. These files contain a user's information and checksums. All key file holds the user's information encrypted, this is done mostly with XOR. Below is an example code in VB how a program might encrypt information using XOR. This is just to give you a idea.

```
-----------------------------------------------------------------------------
Sub Encrypt(secret$, PassWord$)
    L = Len(PassWord$)
    For X = 1 To Len(secret$)
      Char = Asc(Mid$(PassWord$, (X Mod L) - L * ((X Mod L) = 0), 1))
      Mid$(secret$, X, 1) = Chr$(Asc(Mid$(secret$, X, 1)) Xor Char)
    Next
  End Sub
-----------------------------------------------------------------------------
Sub Form_Load()
    Form1.Show  ' Must Show form in Load event before Print is visible.

    secret$ = "This is the string that will be encrypted."
    PassWord$ = "password"

    'secret$ could have the string all ready encrypted and then
    'the subroutine would decrypt it. This is a great way to stop
    'newbie crackers that just dead list programs. Having strings
    'like "Invalid Serial No" is like a map for crackers.

    Call Encrypt(secret$, PassWord$)     'Encrypt the string.
    Print " After encrypting it once: "  'Print the result.
    Print secret$
    Print

    Call Encrypt(secret$, PassWord$)     'A second call to the Encrypt
    Print "After a second encryption:"   'subroutine now decrypts the
    Print secret$                        'encrypted string.

  End Sub
-----------------------------------------------------------------------------
```

To beat a key file based protection scheme you should write a small program to code a bogus key file, this program should allow you to create files under different names and sizes. A Key file based protection scheme does not have to look for a file with the extension "*.KEY" it could look for "*.LIC" or even "*.INI".

How do you know if the program is key file based protection?

If you see no way to register the program the next step is to see if it looks in the registry for a users information, if not the next approach is to see if the program is looking for a key file. You could disassemble the program and look for strings "*.KEY" "*.LIC" or even "*.INI". Note many programs use .INI files to store other information like FORM size, date etc.. But some programs use them to store a user's information as well. You could also use FileMon to see what files the program is accessing or trying to access.

Your program is a key file based protection, now what?

Well if you have the name of the file the program is looking for create a bogus file and copy it to the same path as the main target executable. Sometimes the key File does not have to be in the same path as the main target, it might have to be in your WINDOWS DIR or even WINDOWS/SYSTEM DIR. Start the program, if you get a message box saying "FILE INVAILD" or something, your work has been cut down. If you don't get a message box then you can use SICE, Break Points to use with SICE for R/W files are CreatFile, ReadFile, GetFileAttributes(A) to name a few. I raider get SICE and trace the code and try and find out what the program is looking for. The trick when having to use SICE to beat a program with a key file based protection is to get your encrypted information printed out to the file. Some file based protection programs are really tricky.

- Manual Unpacking (UPX)

We talked about packers, now we are going to talk about unpacking. As you know UPX is a packer (One of the easiest to unpack) that packs an EXE. Find a program packed with UPX and disassemble it in W32asm or IDA and search for POPAD with a jump under it, it could be near the end of the program code as well. It might look something like this

```
:004d3D52       61              POPAD
:004d3D53       E97C31EEFF      JMP 004A7dD4 ← program jumps to OEP
```

Find it good, write down the code location, we have to get SICE to break on this jump. Set a BP on showwindow or createdialog. Then get into the program's code. Then BP on the jump e.g. BPX 4A7dD4 .SICE breaks on the jump and its highlighted .Now take note of where the jmp will jump to, in the example above it will jump to 004A7dD4, write it down while the jump is highlighted type in SICE:

```
a <RETURN>
jmp eip <RETURN>
```

Then clear all BPs and return to windows. What we have done is set the program into a loop, so we can unpack it. Run Procdump. Check the tasks of Procdump and scroll down to your program. Right click on it and choose Dump (Full). Don't try and run the program it will only crash. The OEP is not correct. Remember where the jmp will jump, you should have written it down. What we are going to do with it is get the Imagebase of you program its usually 400000 and take this away from the jmp you written down (OEP).

From the example above

004A7dD4 - 400000 = A7DD4 ← This is our OEP

So load the PE-editor and change the Entry Point to A7DD4. Try run the program now, it works great.

- KeyGens

Load the program you want to make a keygen for, enter in your registration details and break SICE. Set BPX HMEMCPY, leave SICE and hit the OK button in the registration dialog. SICE should break. You have to find the instructions that are used for generating a serial from your registration details.

You can find the serial routine being called before the real serial check. So the CALLS before the serial check need to be observed. When you find the algorithms you have to work out what is done to the registration details you have entered. You should look out for instructions like ADD, IMUL, IDIV, etc.

When you have found the routine that generates you serial number, get a pen and paper and write down what the routine is doing. The next step is to find out what the minimal length of your name needs to be. Also be sure to check if spaces are allowed and special characters.

Keygen for Backup Magic v1.3.1

Start the program enter in you registration details, break SICE, set BPX hmemcpy. Hit the OK button and SICE should break. Press F11 to return to the call, hit F12 a few times to get into program's code and start tracing or quicker still set a BPM on your user name and trace till you land here.

```
MOV       EAX,00000001              ;Move 1 into eax
XOR       ECX,ECX                   ;ecx = 0
MOV       CL,[EAX+ESI-01]           ;move hex value of first character of name into CL
ADD       ECX,EAX                   ;add hex value of first character and eax = 1
MOVZX     EDI,WORD PTR [EBP-02]     ;move word which is 661
IMUL      ECX,EDI ;edi              ; multiply ECX = hex value of first character by edi = 661
IMUL      ECX,ECX,000000B2          ;multiply ECX by 178 (B2 in hex)
ADD       EBX,ECX                   ;add total to EBX
INC       EAX                       ;increase EAX   use to get next character
DEC       EDX                       ;decrease EDX  use for loop
JNZ       0044C525                  ;all characters (YES- No jmp NO- Jmp)
```

You should be able to follow what is going on above. EAX will increase each time the loop starts. Our keygen should do the same as the above routine. To make a keygen for a program you have to know exactly what the program is doing. The above example is easy to make a keygen for. Some programs have long algos that takes lots of time to make a keygen for. Now that you understand the algo lets make a keygen for it. Below is the keygen written in VB, but any program language will do its just simple add and multiply the sum off all characters.

```
--------------Code for Keygen---------------------
namelen = Len(Text1.Text)
x = 1
Do While x <= namelen
namechar = Mid(Text1.Text, x, 1)
nameasc = Asc(namechar)
namecharvalue = nameasc + x
namecharvalue = namecharvalue * 661
namecharvalue = namecharvalue * 178
total = total + namecharvalue
x = x + 1
Loop
Text2.Text = total
--------------------------------------------------
```

- Tips

1) If you come across a piece of code like this

```
 CMP EAX,[EBP-48]
 JZ     xxxxxxxx
```

and you type DD eax and find its you serial number, so you type DD ebp-48 and there is nothing in the data window. You know this piece of code is checking you serial number with the real one. What you do is type DD *(ebp-48), there still might be nothing in the data window. The serial number might be the CODE location of where you land.

2) If you type in a "AbCdEfGhIjKlM" as your name, sometimes you can see what pattern a serial number has straight away.

3) If you cracking a VB program and cant find the real serial number, enter in your name/serial number and set a bpx messageboxindirect(A), hit the OK button, SICE breaks. Then hit F11 and search for you serial number (Wide character) and sometimes you will find the real serial number some where about your fake serial number.

4) If you come across a program you just cant seem to crack, leave it, go to some other program and come back to it thinking fresh.

5) This real takes the name serial fishing to another level, if you cant seem to find the real serial number in SICE type WD 90, this means set the window data to 90 lines. Then use the PageUP and PageDOWN buttons to try and find that serial Number. It's a long shot but sometimes you can find the real serial number this way.

- Conclusion

Well that's it for this tutorial, I hope you learned something. I would like to say, if you like any program you crack and continue to use it you should buy it. Cracking is an art, and you should always be moving forward, on to bigger better protection schemes. Cracking takes time, patience and a bit of hard work. Peace hEYWIRE.

"It is only when they go wrong that computers remind you how powerful they are." - hEYWIRE

- Shout Outs

Shout outs goes to Damo AKA y0ke, thanks for all your help, Ruu , [phrozen] , Sea4 , Bits , Killercris , Psyco , Blitz and all MDC crew.