The big SoftICE howto

A step by step Guide

The big SoftICE howto
A step by step Guide

Version:	1.1
Last Update:	04th April 2006
Author:	Frank Boldewin / www.reconstructer.org

Table of Contents

The big SoftICE howto

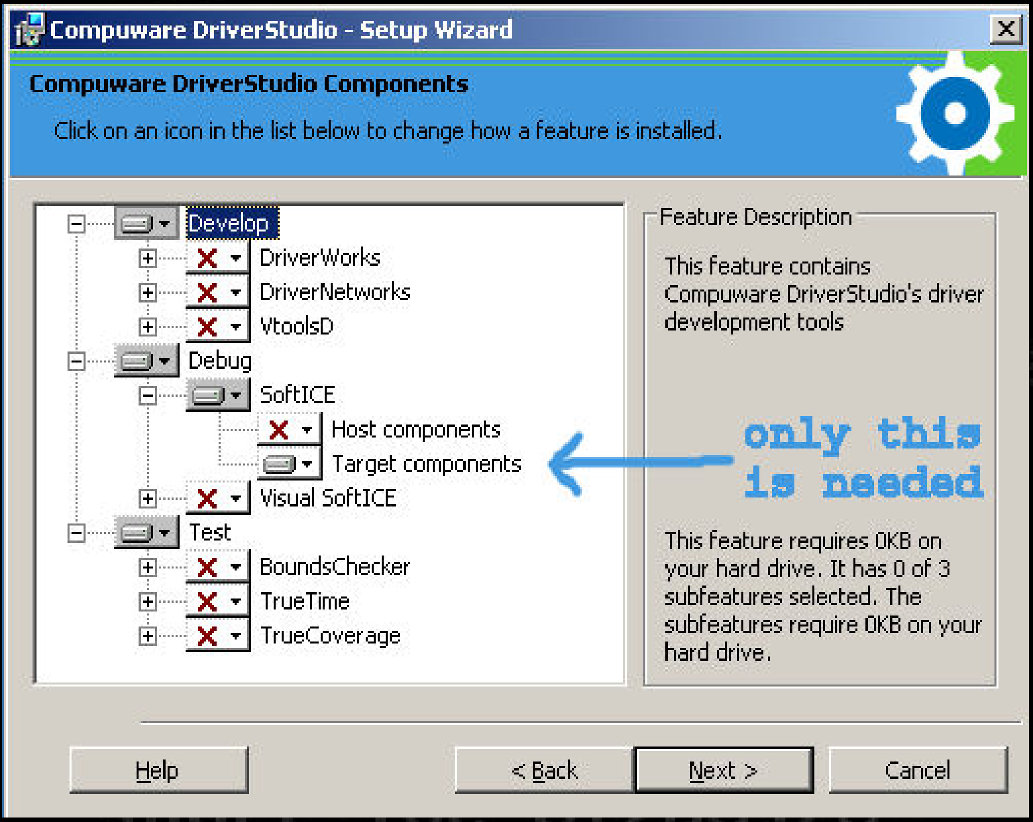
A step by step Guide

Abstract

Debugging applications or drivers are part of every programmer's day. Nearly every IDE I know has its own debugger. Most of them suck in several ways and often don't fulfil all the needs the coder has. Especially when developing a ring0 application, such as a driver for a video or audio device. Without a powerful kernel debugger it's very hard for the coding artist to fix problems, because as you know bigger programming faults lead to bluescreens, followed by a reboot. Referring to the security or antivirus scene a debugger is often used when reversing a binary for vulnerabilities or discovering the functionality of malware. The best disassembler IDA Pro from Datarescue also supports debugging for some time now and improves the reversers work when analyzing an application, particularly when the binary is compressed with an executable packer. Microsoft ships their Visual Studio with a nice debugger which has also the capability of kernel debugging. But almost all debuggers have still some disadvantages. In my opinion currently there's only one debugger that is nearly perfect, the world famous SoftICE. Formerly created by NuMega, sold to Compuware in 1997 and now implemented in Driverstudio, SoftICE is a fully featured debugger with dozens of commands I'll try to bring you closer in this essay. Have you ever wondered what the ICE stands for in SoftICE? Quite easy, it means "In Circuit Emulator". If you don't know what an ICE is, just google for it ;) This paper will give you a step by step introduction to SoftICE. First we'll discuss the most important things while installation and configuration as well as covering several problems that can happen. Subsequent to this I will discuss hotkeys, the most important basic and many advanced commands SoftICE has. Furthermore I will give examples how to use them as well as alluding stumbling blocks with some instructions. In the end of the document I prepared a list of useful API calls, which may help when searching for the right breakpoint in future debugging sessions. To reproduce all the things best, discussed here in the following, you should be armed with WinXP or Win2003, Driverstudio v3.2, IceExt v0.67, Spy & Capture v2.70 as well as VMWare Workstation v5.5. Watch the link list at the bottom where to get the tools needed. The reader of this document should have a basic understanding of x86 assembly and the fundamentals of debugging. Ok, let's getting started now.

Installation & basic configuration

As already mentioned before Compuware implemented SoftICE completely in their development framework for drivers called Driverstudio. It should be clear that this paper will only cover the important matters belonging to SoftICE itself. Further it should be noted that SoftICE has also remote debugging capabilities, which are not focused here as well. Sure, the idea of remote debugging is certainly very smart, especially when it comes to a troubleshooting session of an essential operating system driver, but for our purposes just the installation of the “Target components” is needed.
If you’re interested how remote debugging works, I suggest you to consult the “SoftICE Reference” and the “Using SoftICE” document shipped with Driverstudio. After all it’s not really a big deal, but rather intuitive to install and use, especially if you have experience with remote Microsoft WinDBG sessions. Below is a screenshot of the feature list Driverstudio offers and what to select for our concerns.


Compuware DriverStudio - Setup Wizard

Compuware DriverStudio Components

Click on an icon in the list below to change how a feature is installed.

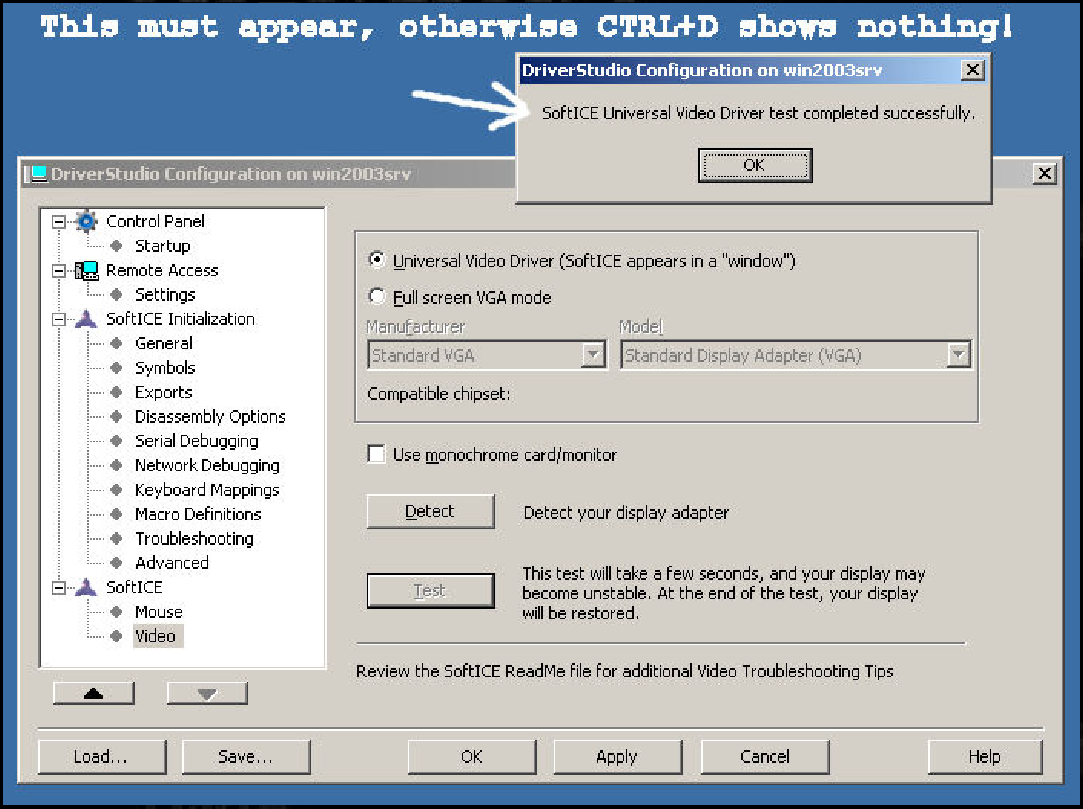
```
Develop
    DriverWorks
    DriverNetworks
    VtoolsD
Debug
    SoftICE
        Host components
        Target components
    Visual SoftICE
Test
    BoundsChecker
    TrueTime
    TrueCoverage
```

Feature Description

This feature contains Compuware DriverStudio's driver development tools

This feature requires 0KB on your hard drive. It has 0 of 3 subfeatures selected. The subfeatures require 0KB on your hard drive.

After all the files were copied to the system, the setup confronts us with several configuration settings. For the setup process only one section is of importance at the moment, the "Video" section. By default the radio button "Universal Video Driver" is set, which is good. Now click "Detect" to let the setup detect your display adapter. Afterwards click "Test" to test if your video driver was properly detected or not. If everything is fine a Messagebox should appear and tells us that the detection completed successfully as seen below in the screenshot.



This must appear, otherwise CTRL+D shows nothing!

DriverStudio Configuration on win2003srv

SoftICE Universal Video Driver test completed successfully.

[✓ Universal Video Driver (SoftICE appears in a "window")]
Full screen VGA mode
Manufacturer: Standard VGA
Model: Standard Display Adapter (VGA)
Compatible chipset:
Use monochrome card/monitor

Detect — Detect your display adapter

Test — This test will take a few seconds, and your display may become unstable. At the end of the test, your display will be restored.

Review the SoftICE ReadMe file for additional Video Troubleshooting Tips

When this operation fails it's fairly telltale that your video driver is not supported. You can try now "Full screen VGA mode" or contact the Compuware support for a hotfix to your problem if you're a registered member. If you plan to install SoftICE under VMWare, e.g. when debugging malware etc., it's mandatory that the "VMWare Tools" are installed as well, otherwise the detection fails. That's it for moment pals! Now finish the installation and reboot to activate SoftICE properly. If you wonder why we left the other configuration options alone, it's because we're discussing most of them separately, when we take a look at the configuration file SoftICE has, after the reboot was successful.

Configuring the WINICE.DAT

Back again after reboot we don't start SoftICE right now. First we take a look at the configuration file of the debugger. You can find it here:

```
%systemroot%\system32\drivers\winice.dat
```

Just edit this file with the editor of your choice. Below is my personal list of entries I always configure right after the installation.

As a start, we trim the design of the debugger (height, width and which windows should appear. The semicolons separate the several commands.

```
INIT="LINES 90; WIDTH 90; WC 50; WD 15; WW 2; WATCH *ESP; WATCH *EBP; CLS; X;"
```

Note that these values are only initial values when SoftICE gets started. You always have the ability to adjust every parameter, e.g. if the code window should be 40 lines instead of 50 lines just type `wc 40` in the debugger and so on.

```
CODEMODE=ON
```

I personally like it when the mnemonic codes to the disassembly are visible while debugging, so I've set `CODEMODE=ON`. If you want to turn them off sometimes, then type `code off` in the debugger.

The last important things for me are the exported API names SoftICE shows me instead of just a call to a memory address while debugging. It can improve the reversing speed drastically if you immediately know which function is behind a call to an address ;)

Note that by default some DLLs can be found already at the bottom of our `winice.dat` file. They're just commented out with a semicolon. Just delete the semicolon to activate these exports.

So here we go with the list:

```
EXP=\SystemRoot\System32\hal.dll
EXP=\SystemRoot\System32\ntoskrnl.exe
EXP=\SystemRoot\System32\ntdll.dll
EXP=\SystemRoot\System32\kernel32.dll
EXP=\SystemRoot\System32\user32.dll
EXP=\SystemRoot\System32\csrsrv.dll
EXP=\SystemRoot\System32\basesrv.dll
EXP=\SystemRoot\System32\winsrv.dll
EXP=\SystemRoot\System32\gdi32.dll
EXP=\SystemRoot\System32\advapi32.dll
EXP=\SystemRoot\System32\ws2_32.dll
EXP=\SystemRoot\System32\msvbvm60.dll
EXP=\SystemRoot\System32\msvcrt.dll
EXP=\SystemRoot\System32\netapi32.dll
EXP=\SystemRoot\System32\rpcrt4.dll
EXP=\SystemRoot\System32\dnsapi.dll
EXP=\SystemRoot\System32\comctl32.dll
EXP=\SystemRoot\System32\comdlg32.dll
EXP=\SystemRoot\System32\ole32.dll
EXP=\SystemRoot\System32\oleaut32.dll
EXP=\SystemRoot\System32\shell32.dll
```

Ok, now save the winice.dat and let's get ready to rumble. ;)

There are two easy options to start the debugger:

1. Click Start ---> All Programs ---> Compuware Driverstudio ---> Debug ---> Start SoftICE
2. Click Start ---> Run ---> cmd.exe <enter> ---> net start ntice

Important Note:

When you have installed SoftICE in VMWare I advise you first to add two parameters to your VMWare configuration file, e.g. W2K3SRV.vmx, otherwise you won't see the debugging screen after breaking into it. This is due to the fact that the "Universal Video Driver" will not draw properly in SoftICE.

Here are the needed values:

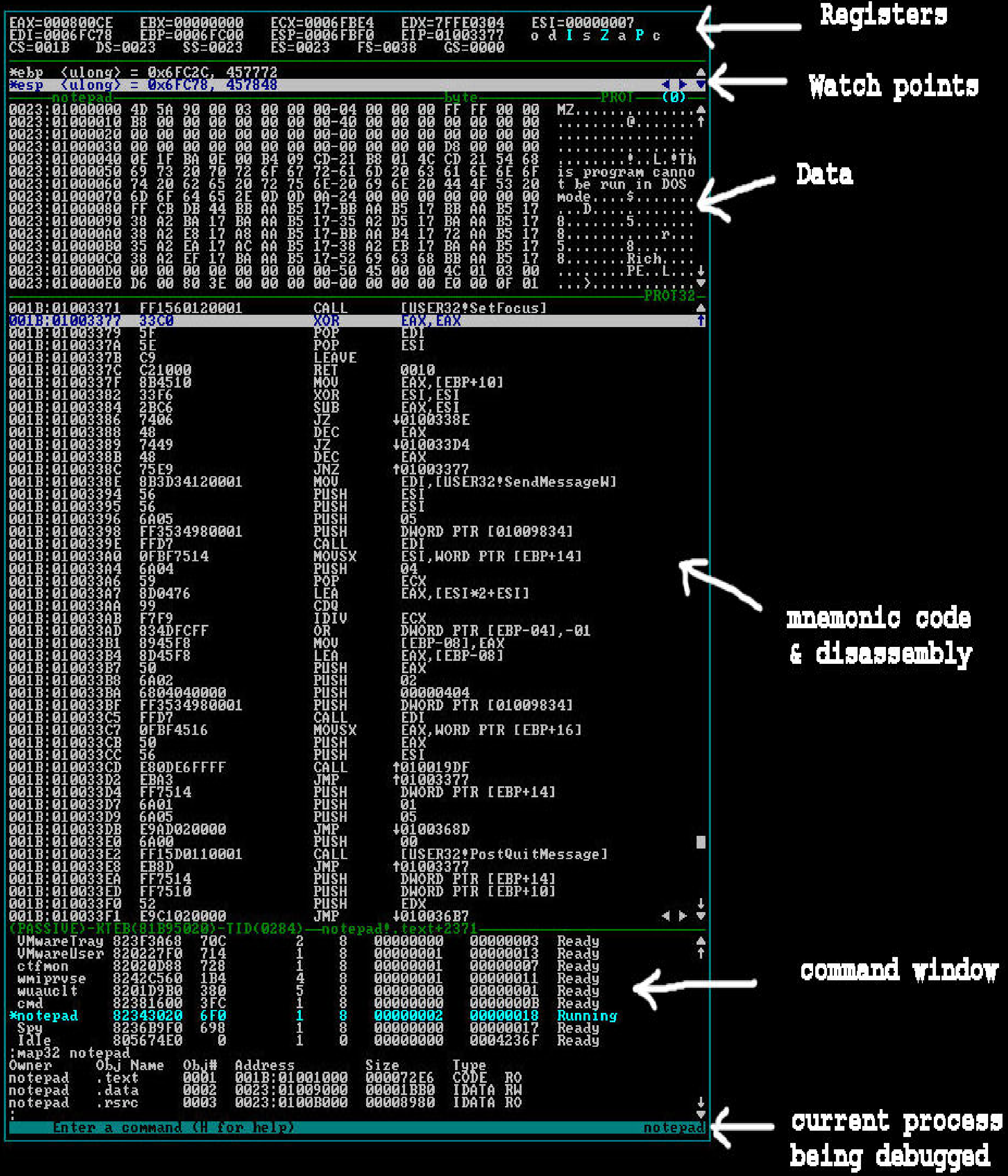
```
vmmouse.present = FALSE
svga.maxFullscreenRefreshTick = 5
```

To activate these settings shut down the potentially running VMWare session and reboot the OS of your choice.

So everything should be fine now and right after starting SoftICE the hotkey `CTRL+D` brings us to our expected debugging screen which should look like the screenshot on the next side now.

The big SoftICE howto

A step by step Guide



```
EAX=000800CE   EBX=00000000   ECX=0006FBE4   EDX=7FFE0304   ESI=00000007
EDI=0006FC78   EBP=0006FC00   ESP=0006FBF0   EIP=01003377   o d I s Z a P c
CS=001B   DS=0023   SS=0023   ES=0023   FS=0038   GS=0000
*ebp  <ulong> = 0x6FC2C, 457772
*esp  <ulong> = 0x6FC78, 457848
------notepad---------------------------------byte-------------PROT----(0)--
0023:01000000 4D 5A 90 00 03 00 00 00-04 00 00 00 FF FF 00 00  MZ..............
0023:01000010 B8 00 00 00 00 00 00 00-40 00 00 00 00 00 00 00  ........@.......
0023:01000020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0023:01000030 00 00 00 00 00 00 00 00-00 00 00 00 D8 00 00 00  ................
0023:01000040 0E 1F BA 0E 00 B4 09 CD-21 B8 01 4C CD 21 54 68  ........!..L.!Th
0023:01000050 69 73 20 70 72 6F 67 72-61 6D 20 63 61 6E 6E 6F  is program canno
0023:01000060 74 20 62 65 20 72 75 6E-20 69 6E 20 44 4F 53 20  t be run in DOS
0023:01000070 6D 6F 64 65 2E 0D 0D 0A-24 00 00 00 00 00 00 00  mode....$.......
0023:01000080 FF CB DB 44 BB AA B5 17-BB AA B5 17 BB AA B5 17  ...D............
0023:01000090 38 A2 BA 17 BA AA B5 17-35 A2 D5 17 BA AA B5 17  8.......5.......
0023:010000A0 38 A2 E8 17 A8 AA B5 17-BB AA B4 17 72 AA B5 17  8...........r...
0023:010000B0 35 A2 EA 17 AC AA B5 17-38 A2 EB 17 BA AA B5 17  5.......8.......
0023:010000C0 38 A2 EF 17 BA AA B5 17-52 69 63 68 BB AA B5 17  8.......Rich....
0023:010000D0 00 00 00 00 00 00 00 00-50 45 00 00 4C 01 03 00  ........PE..L...
0023:010000E0 D6 00 80 3E 00 00 00 00-00 00 00 00 E0 00 0F 01  ...>............
---------------------------------------------------------------------PROT32-
001B:01003371  FF1560120001        CALL      [USER32!SetFocus]
001B:01003377  33C0                XOR       EAX,EAX
001B:01003379  5F                  POP       EDI
001B:0100337A  5E                  POP       ESI
001B:0100337B  C9                  LEAVE
001B:0100337C  C21000              RET       0010
001B:0100337F  8B4510              MOV       EAX,[EBP+10]
001B:01003382  33F6                XOR       ESI,ESI
001B:01003384  2BC6                SUB       EAX,ESI
001B:01003386  7406                JZ        ↓0100338E
001B:01003388  48                  DEC       EAX
001B:01003389  7449                JZ        ↓010033D4
001B:0100338B  48                  DEC       EAX
001B:0100338C  75E9                JNZ       ↑01003377
001B:0100338E  8B3D34120001        MOV       EDI,[USER32!SendMessageW]
001B:01003394  56                  PUSH      ESI
001B:01003395  56                  PUSH      ESI
001B:01003396  6A05                PUSH      05
001B:01003398  FF3534980001        PUSH      DWORD PTR [01009834]
001B:0100339E  FFD7                CALL      EDI
001B:010033A0  0FBF7514            MOVSX     ESI,WORD PTR [EBP+14]
001B:010033A4  6A04                PUSH      04
001B:010033A6  59                  POP       ECX
001B:010033A7  8D0476              LEA       EAX,[ESI*2+ESI]
001B:010033AA  99                  CDQ
001B:010033AB  F7F9                IDIV      ECX
001B:010033AD  834DFCFF            OR        DWORD PTR [EBP-04],-01
001B:010033B1  8945F8              MOV       [EBP-08],EAX
001B:010033B4  8D45F8              LEA       EAX,[EBP-08]
001B:010033B7  50                  PUSH      EAX
001B:010033B8  6A02                PUSH      02
001B:010033BA  6804040000          PUSH      00000404
001B:010033BF  FF3534980001        PUSH      DWORD PTR [01009834]
001B:010033C5  FFD7                CALL      EDI
001B:010033C7  0FBF4516            MOVSX     EAX,WORD PTR [EBP+16]
001B:010033CB  50                  PUSH      EAX
001B:010033CC  56                  PUSH      ESI
001B:010033CD  E80DE6FFFF          CALL      ↑010019DF
001B:010033D2  EBA3                JMP       ↑01003377
001B:010033D4  FF7514              PUSH      DWORD PTR [EBP+14]
001B:010033D7  6A01                PUSH      01
001B:010033D9  6A05                PUSH      05
001B:010033DB  E9AD020000          JMP       ↓0100368D
001B:010033E0  6A00                PUSH      00
001B:010033E2  FF15D0110001        CALL      [USER32!PostQuitMessage]
001B:010033E8  EB8D                JMP       ↑01003377
001B:010033EA  FF7514              PUSH      DWORD PTR [EBP+14]
001B:010033ED  FF7510              PUSH      DWORD PTR [EBP+10]
001B:010033F0  52                  PUSH      EDX
001B:010033F1  E9C1020000          JMP       ↓010036B7
(PASSIVE)-KTEB(81B95020)-TID(0284)--notepad!.text+2371----------------------
 VMwareTray  823F3A68  70C        2    8   00000000   00000003  Ready
 VMwareUser  820227F0  714        1    8   00000001   00000013  Ready
 ctfmon      82020D88  728        1    8   00000001   00000007  Ready
 wmiprvse    8242C560  1B4        4    8   00000001   00000011  Ready
 wuauclt     8201D9B0  380        5    8   00000000   00000001  Ready
 cmd         82381600  3FC        1    8   00000000   0000000B  Ready
*notepad     82343020  6F0        1    8   00000002   00000018  Running
 Spy         8236B9F0  698        1    8   00000000   00000017  Ready
 Idle        805674E0    0        1    0   00000000   0004236F  Ready
:map32 notepad
Owner     Obj Name  Obj#  Address        Size      Type
notepad   .text     0001  001B:01001000  000072E6  CODE  RO
notepad   .data     0002  0023:01009000  00001BB0  IDATA RW
notepad   .rsrc     0003  0023:0100B000  00008980  IDATA RO
:
     Enter a command (H for help)                                   notepad
```

SoftICE commands & hotkeys

After we've learned how we enter the debugger our first command should be how we can leave it. Quite easy, type **x** or press `F5`, press enter and you left SoftICE.

Back again into the debugger after pressing `CTRL+D` we take a short look on the help SoftICE gives us.

```
help
```

Wow, a lot of commands, right? ;) Let's get more exactly:

```
help bpx
```

Outputs:

breakpoint on execution
bpx[.t.|.p|] address [IF expression] [DO bp-action]
***ex:** BPX 282FE0*

And so forth. Just use `help` and a command and you will get more info. Even for assembly instructions SoftICE has a small help called `opinfo`.

```
opinfo xor
```

Ok, now let's have a look which processes can be debugged.

```
proc
```

And which drivers

```
driver
```

But let's stay at the processes first. After typing **proc** we see several processes, their Process-IDs (PID), Threads etcetera.

If we want to enter the address space of a running process now, we have to tell this SoftICE by typing:

```
addr  <processname> or <pid>
```

For example:

```
addr explorer or addr 11c
```

This is due to the fact that a CPU doesn't know about processes at all, it only knows page tables and the operating system has to handle this.

So strictly keep in mind that you always have to use `addr` before SoftICE is able to display the memory space of a given process!

Sometimes addressing the PID makes more sense, e.g. when a program is started more than once and you need to debug a specific one.

Subsequently after addressing explorer.exe, let's have a look at the 32 bit section map of the process.

```
map32 explorer
```

This outputs us owner, object names like .text, .data, .rsrc, .reloc and so forth, the start addresses of the object names as well as their length and type. In case this is all Greek to you, I strongly suggest you to read something about the PE file format. Every newbie in the field of reverse engineering should also have knowledge about file formats like PE, ELF or COFF, especially when trying to reverse malware, because they are very often crunched and/or crypted with an executable packer/crypter. And without an understanding of the given file format I think it's nearly impossible to rebuild the binary after unpacking.

Ok, now some commands you'll need all the time.

d – Display virtual or physical memory

Example: `d ss:esp or d 401000`

This will show you the content of ss:esp in the data window and respectively 401000.

Btw, keep in mind that SoftICE always addresses in hexadecimal, which means for the case of 401000 = 4198400 in decimal. Also note that all commands in SoftICE are case-insensitive.

e – Edit memory

Example: `e ss:esp or e 401000`

Now you can edit the values in memory at ss:esp in the data window. If you have finished editing, just type **ESC** for leaving back to the command window.

And some code window actions.

u – Unassembles instructions

Example: `u cs:eip` or `u 401000`

The example displays the assembly at cs:eip in the code window.

a – Assemble code

Example: `a cs:eip` or `a 401000`

Sometimes it's useful to add/edit code in a debugging session, to directly see what happened after these changes, eg if a driver makes trouble and you want to see if the "hotfix" make things better. ;) So right after entering "a <address>", it's possible to write assembly code. If you're done just hit "enter" once more.

r – Display or change the contents of a register

Example: `r eip`

Switches to register eip and could be edited now. Pressing ESC switches back to the command window.

```
r eax=deadbeef
```

Immediately changes the register eax to the value deadbeef.

The next command searches for data in memory.

s – Search for data

Example: `s 0 L ffff “windows”`

Searches from offset 0 till offset ffff for the string “windows”

```
S –u 0 L ffff “windows”
```

Same search as above, but in Unicode style with one zero byte between each letter.

```
S 0 L ffff 77 69 6e 64 6f 77 73
```

Likewise, the same search as in our first example, but this time in hex.

So after we know how to handle the basic stuff in SoftICE, it’s time for another very important field every debugger supports. I’m speaking of breakpoints. SoftICE has dozens of possibilities to halt a process and even react dynamically, but let’s acquire this step by step. The most often used breakpoint command in SoftICE is certainly BPX.

BPX – Breakpoint on execution

Example: `bpx 401000`

This simple command halts execution when reaching address 0x401000. Quite easy, so let’s get more tricky ;)

```
bpx setfocus if(pid==0x6a4)
```

Which means, break on `setfocus`() but only if the process id is 0x6a4.

What about some macro magic?

Start Notepad.exe and break into SoftICE using `CTRL+D`.
Now search the pid of Notepad.exe using `proc` and enter the following:

```
addr notepad
macro shregkey = "d *(esp+8)"
bpx regopenkeyexa if (pid==0x42c) do "shregkey"
```

Got it?

Ok, some explanation. ;)

With `addr notepad` we set the address context of notepad you know. The command `macro shregkey` … means that if chosen, the contents of where esp+8 points to is shown in the data window. The last command `bpx regopenkeyexa if` … defines to break into SoftICE when regopenkeyexa() is called by notepad.exe (in my case 0x42c) and executes the shregkey macro and shows the second parameter of regopenkeyexa() which is lpSubKey.

Cool eh?

In addition it's also possible to define variables in a macro definition.

```
macro shregkey = "d %1"
bpx regopenkeyexa if (pid==0x42c) do "shregkey *(esp+8)"
```

These are really easy examples of what's feasible. In real-life situations, macros can, for example, help when debugging crypted malware, when manual tracing would take to long by hand.

bl – List current breakpoints

There's nothing more to say here. ;)

bc – Clear breakpoint

Example: `bc 0`

This one clears breakpoint 0 of the breakpoints list.

Sometimes a breakpoint just needs to be inactive rather than clearing it.

bd – Disable breakpoint

Example: `bd 0`

This one disables breakpoint 0 until getting reactivated by using:

be – Enable breakpoint

Example: `be 0`

Another nice breakpoint is bpm.

bpm – Breakpoint on memory access

Example: `bpm 401000+eax rw`

SoftICE breaks when a process is reading or writing from memory address 401000 + eax.

bpint – Breakpoint on interrupt

Example: `bpint 2e if (eax==95)`

Break on interrupt 2e if eax has the value 95 hex.

A nice list to software interrupts can be found here:

http://www.ctyme.com/rbrown.htm or **http://www.cs.cmu.edu/~ralf/files.html**

bpio – Breakpoint on I/O port access

Example: **bpio 1f7 r**

Break if port 1f7 gets read accessed.

A very complete list of I/O ports can be found here:

http://mudlist.eorbit.net/~adam/pickey/ports.html

Or here:

http://www.i-clique.com/dalhousie/cs4122/docs/ioports.pdf

The last breakpoint I am introducing here is called `bmsg`, which is a little bit more complicated to use and requires more explanation first.

bmsg – Breakpoint on Windows Message

Everyone who's familiar with Windows programming should know what window messages do. SoftICE has a short command that lists all the messages windows supports.

wmsg – Display window messages

Further Windows programming works with handles to address the right window when sending or receiving messages. And again SoftICE has a command called `hwnd` that shows us all the handles a process has.

hwnd – Display window handle information

Now start explorer.exe and break into SoftICE and enter:

```
wmsg wm_destroy
```

This outputs wm_destroy=0002 and wm_destroyclipboard. We keep the 0002 in mind.

```
addr explorer
hwnd explorewclass
```

Now we see several infos this class has, eg handle, tid, module etcetera.

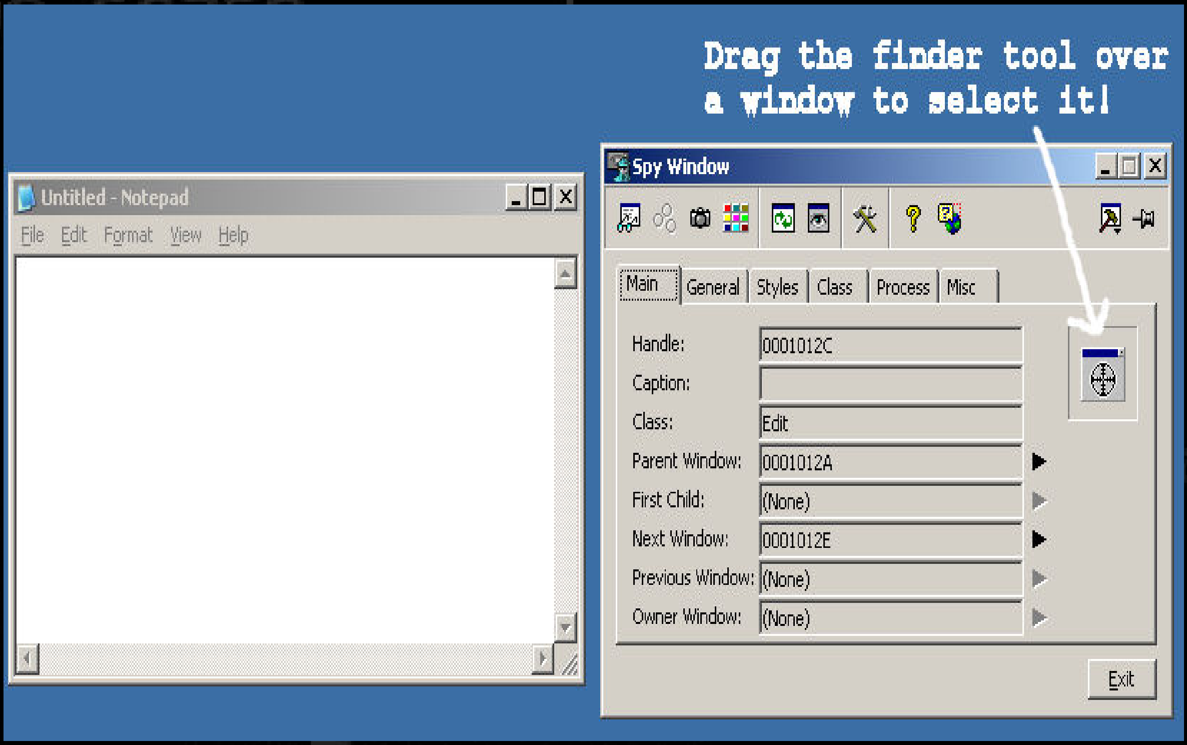
Armed with a windows message and a handle to the process explorer.exe, we are now able to enter the following command.

```
bmsg 2c032e 2
```

The first parameter is our handle we got from `hwnd explorewclass` and the second is the wm_destroy value 0002.

If the explorer.exe will be closed now, our breakpoint gets triggered.

As searching for handles in processes is always a pain for me, I am using a small tool to recognize a special window handle, eg an edit box like in notepad. The tool is called spy & capture and is freeware. For a link, scroll to the bottom and look at the references. Below is a screenshot how to use it. In my opinion it's fairly idiot proof. ;)


Drag the finder tool over a window to select it!
Untitled - Notepad
File Edit Format View Help
Spy Window
Main General Styles Class Process Misc
Exit

Handle:	0001012C
Caption:	
Class:	Edit
Parent Window:	0001012A
First Child:	(None)
Next Window:	0001012E
Previous Window:	(None)
Owner Window:	(None)

Note: When using SoftICE in VMWare don't wonder if you won't get a list of handles with hwnd. Even the handles grabbed with Spy & Capture are illegal under `bmsg`. Don't ask me why. :(

t – Single step one instruction

Example: `t 4`

This executes 4 single instructions. The command `t` can be used without a parameter as well and the default hotkey is `F8`.

p – Step skipping calls, interrupts and so forth.

The command `p` is usually used without any parameter. If `p` is used with the parameter `ret` SoftICE executes the program until one of the next "ret" instructions is encountered. The default hotkey for `p` is `F10`. The hotkey for `p ret` is `F12` and is useful after SoftICE halted the programs flow because of a breakpoint in an external function.

Example:

1. Start Notepad.exe and enter SoftICE using `CTRL+D`
2. addr notepad
3. bpx dialogboxparamw
4. Leave SoftICE with `x` or `F5`
5. In Notepad click Help --> About Notepad and SoftICE should break

As you see now, the function DialogBoxParamW() is placed in the user32.dll. To get a clue who's the caller of this function press `F12`. Subsequently, the "About Notepad" Dialogbox should appear. Press "OK" and you're back in SoftICE and now in shell32.dll, which was the caller of the function DialogBoxParamW(), another Windows library. Pressing `F12` again we see that DialogBoxParamW() was called by the function ShellAboutW(). And hitting `F12` a last time we're back into Notepad.exe, the original caller of the upper operations.

The last hotkey avowed here is `F6` and switches between code and command window.

For all other hotkeys and their meaning consult the winice.dat

rs – Restore program screen

I'm using this command when it's temporarily needed to hide the SoftICE screen, e.g. for viewing new appearances after tracing some code. Press any key to switch back to SoftICE.

faults – Enable/disable SoftICE fault trapping

Example: `faults off`

As SoftICE always reports every unhandled exception, it's sometimes needed to turn off fault trapping for some time, to avoid an infinite loop. To reactivate fault trapping use `faults on`.

i3here – Direct Interrupt to SoftICE

Example: `i3here on`

Every time an interrupt 3 occurs, mnemonic code 0xcc, SoftICE halts. Other parameters are `off` or `drv` for drivers only.

exp – Display export symbols

Example: `exp dialogbox*`

Will show all export symbols matching the regular expression `dialogbox*`, `exp` without a parameter, shows all known exports.

cls – Clear window

This one clears the code window.

In the end let's do some maths with SoftICE.

```
? eax+1a              Addition
? 10-7                Subtraction
? -100/2              Division
? esi*eax             Multiplication
? 10%3                Modulo
? 10>>2 or eax<<4     Logical shift right/left
```

IceExt – A useful SoftICE extension

As some might know, SoftICE has its own API, e.g. to extend the debuggers features. A very useful extension for our beloved debugger is called IceExt. It was written by a Russian guy called Sten and is freeware. The link to it can be found in the references.

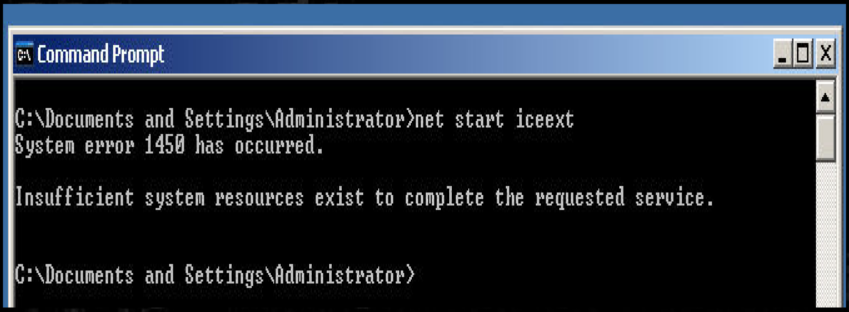
Right after downloading and installing the current version 0.67, IceExt can be started in two ways:

1. Click Start ---> All Programs ---> IceExt ---> Start IceExt
2. Start ---> Run… ---> cmd.exe <enter> ---> net start iceext

If no strange message occurs now, like in the screenshots below, you should be able to break into SoftICE and enter:

```
!help
```

To see what commands IceExt has. Otherwise their might be a small problem me and some friends had with version 0.67.



```
C:\Documents and Settings\Administrator>net start iceext
System error 1450 has occurred.

Insufficient system resources exist to complete the requested service.


C:\Documents and Settings\Administrator>
```

To fix this problem cut and paste the 3 liner below into a file called Fix_KDHeapSize.reg, save and execute it. To activate the changes, reboot the system and start IceExt again.

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTice]
"KDHeapSize"=dword:00008000
```

Here are some of the features IceExt adds to SoftICE.

!dump – Dump memory to disk

Example: `!dump \??\c:\unpackedmalware.bin 400000 fc00`

Very useful, for example, if a malware was compressed with an unknown packer before. The first parameter is the file to write, the second is the start address and the third is the length.

!dumpscreen – Dump SoftICE screen to disk in RAW format

Example: `!dumpscreen \??\c:\SoftICEScreen.raw`

In cases where you don't have the debugger running under VMWare it isn't possible to make a snapshot of the current SoftICE screen. The feature `!dumpscreen` eliminates this obstacle. To convert the saved .raw file into a .bmp file use the tool SiwRender. This commandline tool can be found in the directory `%ProgramFiles%\IceExt\SiwRender`. Before the tool is used assure that the settings in the `SiwRender.ini` are equal to your current SoftICE window settings, eg font, width and height. Otherwise adjust them like in the example below.

SiwRender.ini:

```
[Main]
FontName    = 8x8std.fnt
FontWidth   = 8
FontHeight  = 8
ScreenWidth  = 80
ScreenHeight = 90
```

!protect – Turn SoftICE protection on/off

Example: `!protect on`

Some applications and malware use several techniques to protect themselves from getting debugged. This nice feature works as a bypass for some of the common Anti-SoftICE tricks. It protects from MeltICE, UEF tricks, NtQuerySystemInformation and CR4 Debug Extensions bit protection.

Setting the right breakpoint – Useful APIs

If a reverser searches for a special action in an application it's important to know which API calls the system might use. Below is a categorized list of calls, often used in programs.

File:
```
CreateFileA / CreateFileW
ReadFile
ReadFileEx
WriteFile
WriteFileEx
SetFilePointer
SetFilePointerEx
GetSystemDirectoryA / GetSystemDirectoryW
GetFileAttributesA / GetFileAttributesW
GetFileAttributesExA / GetFileAttributesExW
GetFileSize
GetFileSizeEx
GetDriveTypeA / GetDriveTypeW
GetLastError
```

INI-File:
```
GetPrivateProfileStringA / GetPrivateProfileStringW
GetPrivateProfileIntA / GetPrivateProfileIntW
GetPrivateProfileSectionA / GetPrivateProfileSectionW
GetPrivateProfileStructA / GetPrivateProfileStructW
WritePrivateProfileStringA / WritePrivateProfileStringW
WritePrivateProfileSectionA / WritePrivateProfileSectionW
WritePrivateProfileStructA / WritePrivateProfileStructW
```

Registry:
```
RegCreateKeyA / RegCreateKeyW
RegCreateKeyExA / RegCreateKeyExW
RegDeleteKeyA / RegDeleteKeyW
RegQueryValue / RegQueryValueEx
RegEnumKeyA / RegEnumKeyW
RegEnumKeyExA / RegEnumKeyExW
RegEnumValueA / RegEnumValueW
RegSetValueA / RegSetValueW
RegSetValueExA / RegSetValueExW
RegOpenKeyA /RegOpenKeyW
RegOpenKeyExA /RegOpenKeyExW
RegCloseKey
```

Directory:

CreateDirectoryA / CreateDirectoryW
CreateDirectoryExA / CreateDirectoryExW
GetCurrentDirectoryA / GetCurrentDirectoryW
GetSystemDirectoryA / GetSystemDirectoryW
GetWindowsDirectoryA / GetWindowsDirectoryW
RemoveDirectoryA / RemoveDirectoryW

Message boxes:

MessageBoxA / MessageBoxW
MessageBoxExA / MessageBoxExW
MessageBoxIndirectA / MessageBoxIndirectW
MessageBoxTimeoutA / MessageBoxTimeoutW
SoftModalMessageBox
MessageBeep

Dialog boxes:

CreateWindowExA / CreateWindowExW
CreateDialogIndirectParamA / CreateDialogIndirectParamW
CreateDialogParamA / CreateDialogParamW
DialogBoxIndirectParamA / DialogBoxIndirectParamW
DialogBoxParamA / DialogBoxParamW
ShowWindow
EndDialog

Edit boxes:

GetWindowTextA / GetWindowTextW
GetDlgItemTextA / GetDlgItemTextW
GetDlgItemInt
SetWindowTextA / SetWindowTextW
SetDlgItemTextA / SetDlgItemTextW
SetDlgItemInt

Time:

GetLocalTime
GetFileTime
GetSystemTime
GetTickCount
FileTimeToSystemTime
SystemTimetoFileTime
CompareFileTime

CDROM:

GetDriveTypeA / GetDriveTypeW
GetLogicalDrives
GetLogicalDriveStringsA / GetLogicalDriveStringsW

For further info on these API calls consult the MSDN library from Microsoft. Note that other programming languages than C/C++ have other API calls as listed above, e.g. Visual Basic for instance uses RtcMsgBox() instead of MessageBox().

Conclusion

After studying this paper the reader now should have a better understanding how to use the most powerful debugger ever created. As a matter of course this essay hasn't covered all available commands SoftICE offers, but with the information given, it should be no problem now to expand the knowledge. I hope you found this document useful. If you have questions, comments or a constructive review, just drop me a mail. Happy reversing!

References

Driverstudio v3.2
http://www.compuware.com/products/driverstudio/782_ENG_HTML.htm
IceExt v.67
http://stenri.pisem.net
VMWare Workstation v5.5
http://www.vmware.com/products/ws/
Spy & Capture v2.70
http://programmerstools.org/node/348

Big thanks go to Sten, Trapflag and Marc Schönefeld for reviewing this essay.