SiceDbg: Making OllyDbg act more like our beloved Softice

by Crudd [RET]

Intro...:

What we are going to do is make OllyDbg use the same buttons as SI does. This will make using Olly easier for SI users. OllyDBG uses different keyboard setup than SI does. This makes things more difficult for people who are use to using SoftIce. We either have to remember another set of keys, or use the mouse and click, click, click everywhere. So we are going change Olly to make it use SI keys. There are probably several ways to go about doing this. This is the method I choose. It's probably not the easiest, but its what I came up with.

Location, location, location...:

The first thing we need to do is to find out where our keyboard presses get handled. Go ahead and start loading Olly into IDA. Now we need to load up OllyDbg and load a file. OK. So we need to we need to get inside of Olly and find where the keys are processes. Seeing the possible keys we have and thier respective actions, we decide to use CTRL-F2 (Restart program). We are hoping this will call the 'CreateFileA' API and from there we can find the loop that processes our keypresses. So let's jump to our lovely SI and set a breakpoint on 'CreateFileA'. Now in Olly we hit Ctrl-F2 (RestartProgram). And *BOOM* we break in SI. Hit F12 to execute til return. We land at 4A49F8. Let's set our BP here and check this out in IDA:

```
.text:004A4882                    align 4
.text:004A4884 ; [000001F6 BYTES: COLLAPSED FUNCTION ___open. PRESS KEYPAD "+" TO EXPAND]
.text:004A4A7A                    align 4
```

So we are inside this function. This doesn't look very interesting, so let's check out where this is called from. CTRL-F2 in Olly, F12 (out of CreateFileA), F12 (out of ___open) and we land at 4A52E1. This happens to be inside another uninteresting proc named: ___openfp. So lets F12 on more time and see where we are. 4A5393. And this is another uninteresting proc: _fopen. So we F12 one more time and we land at 47795D. Now, let's check this out in IDA. Well it seems good old IDA went and named this proc for us too. _OpenEXEfile is the name of this proc.

Lets take a look where this proc is called from. F12 one more time and we land at 431D57. Let's take a look in IDA again and see what it can tell us.

```
.text:00431D3B ; ---------------------------------------------------------------------------
.text:00431D3B
.text:00431D3B loc_431D3B:                             ; CODE XREF: sub_431B90+189↑j
.text:00431D3B                                         ; sub_431B90+18E↑j ...
.text:00431D3B                 cmp     eax, 100h
.text:00431D40                 jnz     short loc_431D5F
.text:00431D42                 cmp     edx, 71h
.text:00431D45                 jnz     short loc_431D5F
.text:00431D47                 test    ecx, ecx
.text:00431D49                 jz      short loc_431D5F
.text:00431D4B                 push    0FFFFFFFFh
.text:00431D4D                 push    offset byte_4D5B80
.text:00431D52                 call    _OpenEXEfile
.text:00431D57                 add     esp, 8
.text:00431D5A                 jmp     loc_4322AC
.text:00431D5F ; ---------------------------------------------------------------------------
```

Looking at this code and the code around it, we can guess that this is keyboard shortcut handling routine. So let's set a bp at the start of the proc, at: 431B90. Run Olly again. This time let's hit F8 and see what happens. It looks like we've found our spot. So let's give it a cool name that we can remember. KeyboardShortcutHandlingRoutine works great.

Reconnaissance...:

After hitting F8 we can see all the registers are initilized. Then they are checked for certain conditions. After stepping past the initilizations, we can see that: EAX==100 ECX==0 & EDX=77. Now its time to gather all the information that we are gonna need to start the patch. Let's do the same thing that we did for F8, for the other buttons. We need F5 (Execute), F8(Step Into), F10(Step Over), and F12(Execute til Return). We are also going to handle F7 (Run to Cursor) and F9 (BPX on Cursor). And we'll need F2, F4, and Ctrl-F9 becuase Olly uses them for 'BPX on Cursor', 'Run to Cursor', and 'Execute til Return'. After checking all the buttons this is what we have:

KEY	EAX	ECX	EDX
F2	100	0	71
F4	100	0	73
F5	100	0	74
F7	100	0	76
F8	100	0	77
F9	100	0	78
CTRL-F9	100	0	78
F10	104	8000	79
F12	100	0	7B

That should be all the info we need. Lets study the code a bit and see where and what we need to patch.

'Step Into' Ollys secrets...:

So lets go ahead and start with F7 (Olly)/F8(SI) 'Step Into'. From our chart, we can see that after F7 is pressed EAX==100, ECX==0, and EDX==76. So let's find where this is checked in IDA.

```
.text:00431F48 ; ---------------------------------------------------------------------------
.text:00431F48
.text:00431F48 StepInto:                               ; CODE XREF: KeyboardShortcutHandlingR
.text:00431F48                                         ; KeyboardShortcutHandlingRoutine+37A↑
.text:00431F48                 cmp     eax, 100h
.text:00431F4D                 jnz     short StepOver
.text:00431F4F                 cmp     edx, 76h
.text:00431F52                 jnz     short StepOver
.text:00431F54                 test    ecx, ecx
.text:00431F56                 jz      short loc_431F62
.text:00431F58                 push    1
.text:00431F5A                 call    _Animate
.text:00431F5F                 pop     ecx
.text:00431F60                 jmp     short loc_431F6A
.text:00431F62 ; ---------------------------------------------------------------------------
```

So it looks like all we need to do is change the cmp at 431F4F to check for F8 instead of F7. So patch 431F51 from 76h (F7) to 77h (F8). RVA can be found at the bottom of IDA: 3154F, but we want to patch the 3rd byte, so patch 31551 to 77. Save it and try using F8 in Olly. Beautiful, works like a charm. F7 doesn't do anything right now, but that will be fixed that later. Same goes for the buttons at the top of Olly.

Since we are trying to use SI keys instead of the mouse, we don't care about this, but maybe we'll fix it later just for cleanliness. All finished with F8, now onto F10.

'Step Over' the same obstacles as last time...:

Ollys 'Step Over' snippet starts at 431F80.

```
.text:00431F80 ; ---------------------------------------------------------------------------
.text:00431F80
.text:00431F80 StepOver:                               ; CODE XREF: KeyboardShortcutHandlingR
.text:00431F80                                         ; KeyboardShortcutHandlingRoutine+3C2↑
.text:00431F80                 cmp     eax, 100h
.text:00431F85                 jnz     short ExecuteTilReturn
.text:00431F87                 cmp     edx, 77h
.text:00431F8A                 jnz     short ExecuteTilReturn
.text:00431F8C                 test    ecx, ecx
.text:00431F8E                 jz      short loc_431F9A
.text:00431F90                 push    2
.text:00431F92                 call    _Animate
.text:00431F97                 pop     ecx
.text:00431F98                 jmp     short loc_431FA2
.text:00431F9A ; ---------------------------------------------------------------------------
```

This is pretty much the same deal. The only difference is that EAX is 104 when F10 is hit, instead of 100. So, we need to change the two cmp to compare with F10 values. So change 31581 to 04 (cmp eax, 104h) and 31589 to 79 cmp edx, 79). Again, anything we messed up, we are going to fix later. Try our patch and see if F10 gives us the desired results. Should work perfectly. Two down, four to go.

'til I return'...:

The next two shortcuts we need to switch are 'Execute' and 'Execute til Return'. As luck would have it both of these use the same Fkey. So we have both of our needed procs right together. We just need to make space for two compares instead of one. We need to check EDX twice and EAX once. Now we just need to find the space to do this.

Looking at the code a bit, I choose 431F52 as my start. Here's why: at this point, we know that EAX==100 (because it passed the check at 431F48), so if we just change the 'jnz 431F80' to 'jnz 431FB8' then we can finish up the rest of our patch (the reason we are jumping here, is because the next check, checks EAX for 104, but we know its 100, so we don't need to mess with this). I changed it first in SI to see what we need to change 431F53 to. It's 64h. So Plug that into you hex editor. Next we're going to need to edit the snippet at 431FB8 a bit.

```
.text:00431FB8 ; ---------------------------------------------------------------------------
.text:00431FB8
.text:00431FB8 ExecuteTilReturn:                       ; CODE XREF: KeyboardShortcutHandlingR
.text:00431FB8                                         ; KeyboardShortcutHandlingRoutine+3FA↑
.text:00431FB8                 cmp     eax, 100h
.text:00431FBD                 jnz     short PopupMenu
.text:00431FBF                 cmp     edx, 78h
.text:00431FC2                 jnz     short PopupMenu
.text:00431FC4                 test    ecx, ecx
.text:00431FC6                 jz      short Execute
.text:00431FC8                 push    3
.text:00431FCA                 call    _Animate
.text:00431FCF                 pop     ecx
.text:00431FD0                 push    1
.text:00431FD2                 push    esi
.text:00431FD3                 push    2
.text:00431FD5                 push    0
.text:00431FD7                 push    0
.text:00431FD9                 call    _Go
.text:00431FDE                 add     esp, 14h
.text:00431FE1                 jmp     loc_4322AC
.text:00431FE6 ; ---------------------------------------------------------------------------
```

So here's what we need to change. We know EAX is 100h (we just checked it at 431F48). That means we can change the 'cmp eax, 100h' to 'cmp edx, 7Bh' (F12 'Execute til Return). Looking at the disassembly, we can see that 'cmp edx, XXh' is 83h,0FAh,XXh. So lets make the change. Patch 315B8 to 83h, 0FAh, 7Bh, 90h, 90h. The 90h's are NOPS to fill in the byte not used by our new instruction. Now we need to change the 'jnz 432004' at 431FBD to 'jz 431FC8'. This is the beginning of 'Execute til Return'. So change 315BD to: 74h, 09h (jz 431FC8). Save it and test it. F12 should now be working.

'Execute'ing F5...:

Now it's time for 'Execute'. All we need to do is check if EDX is 74h (F5) If so, then jump to Execute. So just change 431FC1 to 74h. Run it. Hit F5. WTF? It doesn't work. Why not? Because F5 is handled earlier in the proc, remember? So lets jump back to where its handled it and make one of our unused keys do F5s old job. I used F2, but it doesn't matter. So just replace the byte at 3144Ch with 71h, and try running it again. Ok, F2 works and we break at our bp when F12 is hit. Now we can continue. Leave the 'jnz 432004' and add a 'jz Execute' after it. This will continue execution at the right place (if its not one our buttons). So just patch 431FC4 to 74h, 22h. And that does it. Olly now uses SI main 'stepping' keys. We just need to add our two BP keys and we are done.

'Break' on through...:

Ok, now we are on our last two keys. But, as luck would have it. They don't appear to be handled by the routine that we have been dealing with (at 431B90). So, we need to find where else our keys are being processed. For this, i had no idea(s). After taking a break, I thought that where ever processes our 'Run to Cursor' key would have to call Olly's '_Go' proc. This is called by all the keys in the previous section that run the code. Here's just one example to show you what I mean:

```
.text:00431FA2                 push    1
.text:00431FA4                 push    esi
.text:00431FA5                 push    2
.text:00431FA7                 push    0
.text:00431FA9                 push    0
.text:00431FAB                 call    _Go
.text:00431FB0                 add     esp, 14h
.text:00431FB3                 jmp     loc_4322AC
```

So lets check out _Go and see where all its called from. Cool, 6 refs from our proc and two refs from another proc. Lets check them out. The call from 41F937 looks like a winner. A few lines below it, we have:

```
.text:0041F944                 cmp     [ebp+c], 73h
.text:0041F948                 jnz     short loc_41F99E
```

73h, isn't that the number we are looking for? That was surprisingly easy. So just change the 73h to 76h (F7) and see if it works. Awesome. Now just one more key. And knowing how our last proc was set up, lets scroll up a bit:

```
.text:0041F8CB                  cmp     [ebp+c], 71h
.text:0041F8CF                  jnz     short loc_41F91D
.text:0041F8D1                  mov     edx, [ebp+var_44]
.text:0041F8D4                  cmp     edx, [ebp+var_48]
.text:0041F8D7                  jnb     short loc_41F91D
.text:0041F8D9                  mov     ecx, [ebp+var_44]
.text:0041F8DC                  cmp     ecx, dword_4CDA1D
.text:0041F8E2                  jb      short loc_41F91D
.text:0041F8E4                  mov     eax, dword_4CDA1D
.text:0041F8E9                  add     eax, dword_4CDA21
.text:0041F8EF                  cmp     eax, [ebp+var_44]
.text:0041F8F2                  jbe     short loc_41F91D
.text:0041F8F4                  mov     edx, [ebp+var_14]
.text:0041F8F7                  push    edx
.text:0041F8F8                  mov     ecx, [ebp+var_10]
.text:0041F8FB                  push    ecx
.text:0041F8FC                  mov     eax, dword_4CD8E1
.text:0041F901                  push    eax
.text:0041F902                  push    0
.text:0041F904                  mov     edx, [ebp+var_18]
.text:0041F907                  push    edx
.text:0041F908                  mov     ecx, [ebp+c]
.text:0041F90B                  push    ecx
.text:0041F90C                  mov     eax, [ebp+var_44]
.text:0041F90F                  push    eax
.text:0041F910                  call    sub_419974
.text:0041F915                  add     esp, 1Ch
.text:0041F918                  jmp     loc_41FFBF
```

Well, hot shit. There's our F2. So change that to 71h, to 76h and we should be all set. Save and fire up Olly. Load us a program and try setting a BP with F9. *POOF* Nothing happens. It looks like we forgot something again. Let's have another look at our last snippet. At 41F908 we see 'mov ecx, [ebp+c]', 'push ecx'. This is pushing our button number. Evidently this is used by the call at 41F910, but we don't care. We can just change 41F908 to 'push 71', 'nop', 'nop'. Which is 6Ah, 71h, 90h, 90h. Fix that, and now Olly uses all SI keys.

This concludes this installment. I'll have to finish the rest up later. I'm tired and its time to drink. :) I've included the patched Olly and an .ini file using SI color scheme (Thanks Maaaarius).

Crudd [RET]