

ITSC304: Operating Systems Exploitation

Lab Quiz #1

**Answer the questions and attach requested screen captures to demonstrate results. Paste the captures under respective question. Submit Lab Quiz on D2L.**

**To complete these tasks you MUST use your virtual machines. You are allowed to use your labs as reference and virtual machines** **you used to complete the labs**

ITSC205: Operating Systems Internals

**NAME: \_Coleton Sanheim\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Mark:\_\_\_\_\_\_/77**

*“Academic dishonesty in any fashion is a serious offence.  Anyone caught cheating will be dealt with according to SAIT’s academic policy and procedure, Student Code of Conduct AC 3.4 and AC 3.4.1, and as has been detailed in the ETHI 110 Academic Honesty Awareness Tutorial.”*

1. **( 6 marks)**Use the respective databases to find vulnerability **CVE-2021-25274** and provide the following information:
   1. Brief Description

**The collector service in Solar Winds uses MSMQ and doesn’t set permissions on its private queues. This allows unauthenticated users to remotely send messages to TCP port 1901 that will be processed, and can allow for remote code execution.**

* 1. Severity

**9.8 CRITICAL**

* 1. CWE ID and description

**CWE-502, the application deserializes untrusted data and does not verify that the data will be valid.**

* 1. Likelihood of exploit

**Medium**

* 1. Example of vulnerable code

**try {**

**class ExampleProtocol(protocol.Protocol):**

**def dataReceived(self, data):**

**# Code that would be here would parse the incoming data**

**# After receiving headers, call confirmAuth() to authenticate**

**def confirmAuth(self, headers):**

**try:**

**token = cPickle.loads(base64.b64decode(headers['AuthToken']))**

**if not check\_hmac(token['signature'], token['data'], getSecretKey()):**

**raise AuthFail**

**self.secure\_data = token['data']**

**except:**

**raise AuthFail**

**}**

* 1. Potential Mitigation

**Explicitly define a final object() to prevent deserialization.**

1. **( 8 marks)** Use **attack.mitre.org** web site to find Microsoft **Powershell** attack

and provide:

* 1. The ID and a short description of the technique

**T1546.013, Attackers can gain persistence and elevate privileges by executing malicious content via PowerShell profiles.**

* 1. What tactic this PowerShell technique belongs to?

**Privilege Escalation/Persistence**

* 1. Data Sources

**Command**

**File  
Process**

* 1. A procedure example where PowerShell was used

**Turla has used PowerShell profiles to maintain persistence on an infected machine. ID – G0010**

* 1. Mitigation

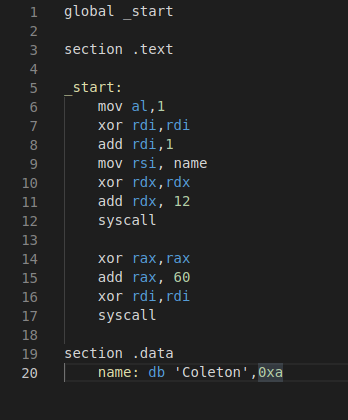
**ID – M1054, Code Signing. Only enforce execution of signed PowerShell scripts.**

* 1. List two detection techniques

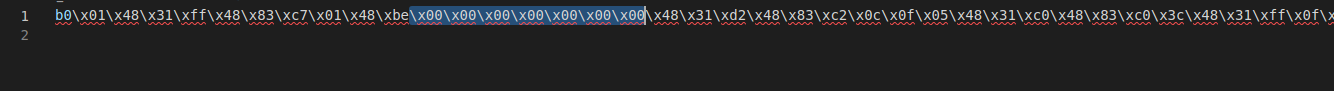
**ID – DS0017, Command Execution**

**ID – DS0022, File Creation/File Modification**

1. **( 8 marks)** Create a clean shellcode that prints your name instead of hello world. Extract the shellcode. Use generated shellcode to create code to inject and call it yourname.c Attach following screen captures:
   1. Assembly code

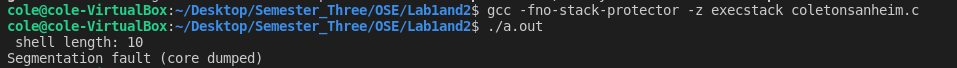


* 1. Extracted shellcode (opcode)



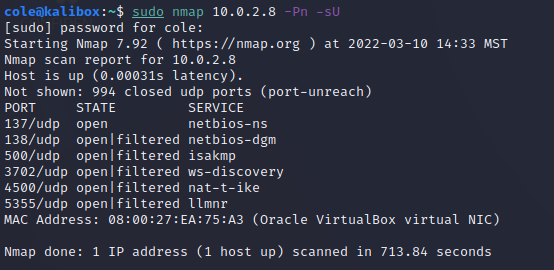


* 1. Results after compiling and executing yourname.c

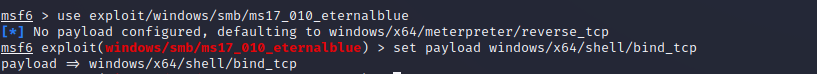


**You’ll notice there are nulls in the shellcode generated, even though I used the method described in the lab. So as a result the shellcode does not actually execute even though I followed the steps exactly.**

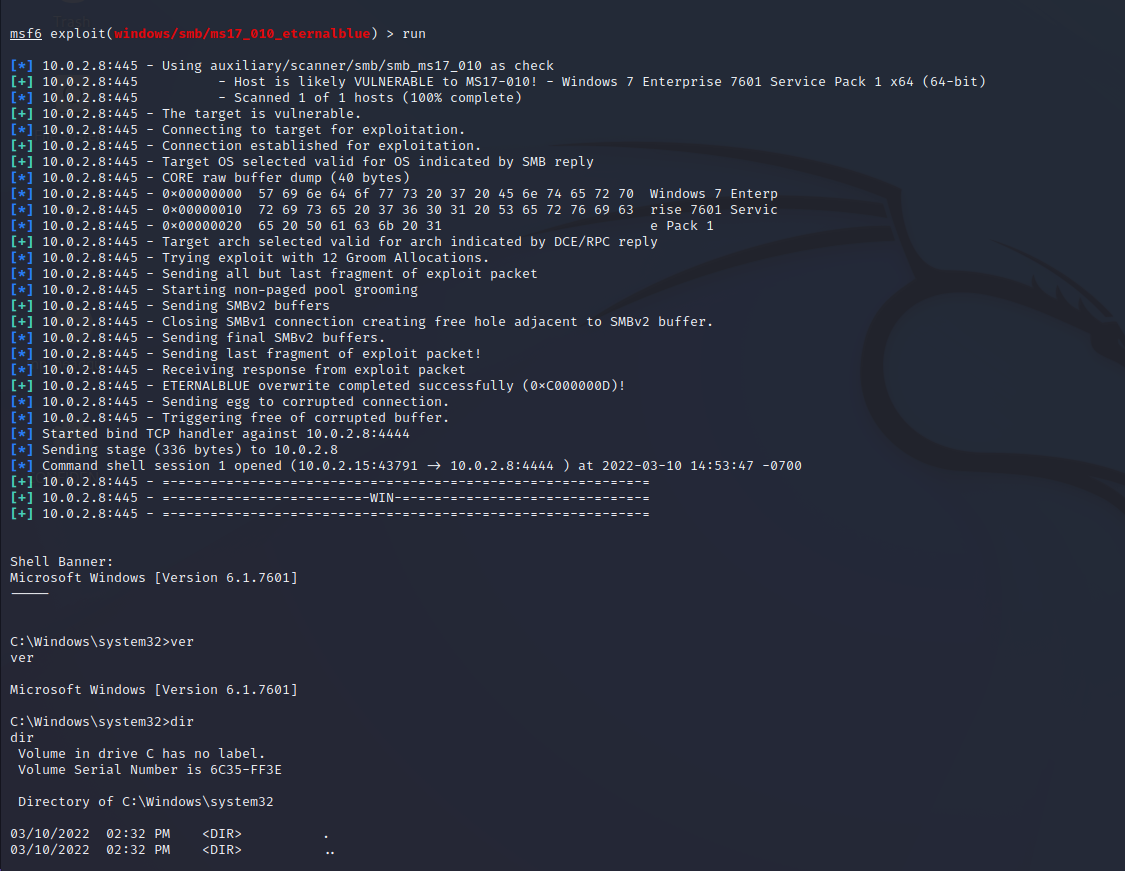
1. **(10 marks)** Use metasploit to exploit Windows-7-64 bit vulnerable machine using **ms17\_010-eternalblue** exploit with **shell/bind\_tcp payload**. Provide the following screen captures to demonstrate the results:
   1. Display **ALL** ports states on victim machine (Windows 7-64)



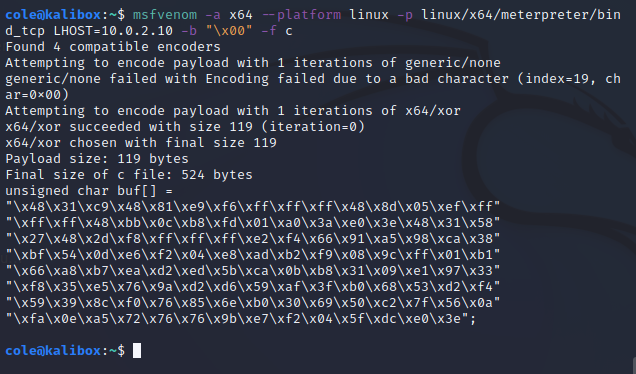
* 1. Used exploit **ms17\_010\_eternalblue** with payload **shell/bind\_tcp**



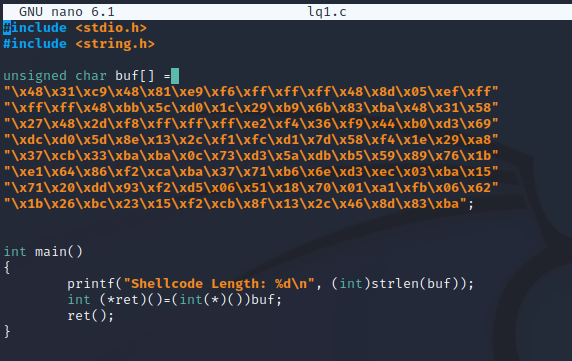
* 1. Run the exploit and display the compromised system using shell commands such as: ver and dir



1. **(10 marks)** Use Metasploit –msfvenom to generate a **linux meterpreter bind\_tcp** **payload** , compile and execute it in Ubuntu victim machine as **lq1.c** and exploit it. Provide the following screen captures to demonstrate the results:
   1. msfvenom that generates a linux meterpreter bind\_tcp payload

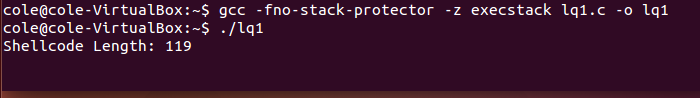


* 1. nano lq1.c

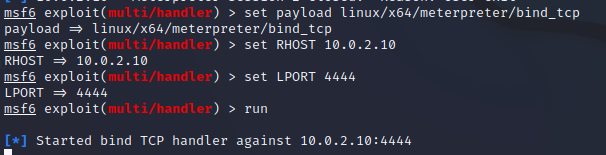


**Note that the shellcode is different as I ran the command again and directed it into the file.**

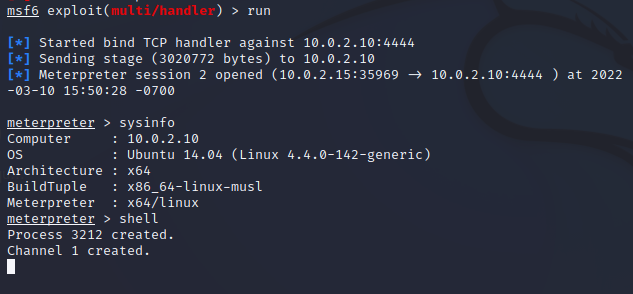
* 1. Compilation and execution of **lq1.c** on Ubuntu (victim machine )



* 1. Kali set up as a listener



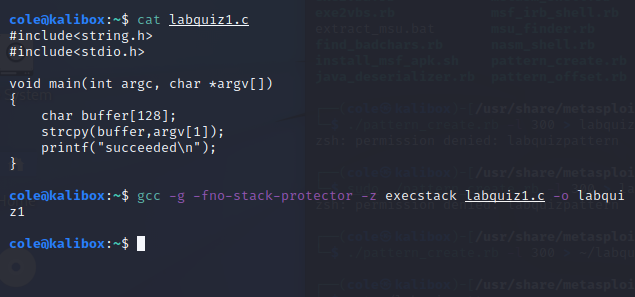
* 1. Exploit result. Compromised machine. Meterpreter commands that demo the results using meterpreter commands such as: **sysinfo and shell**



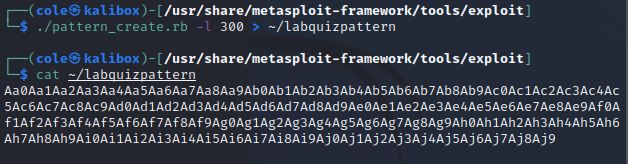
1. **(15 marks)** Use stack overflow to exploit a vulnerable system by implementing

**NOP sled**. Attach the following screen captures to demonstrate the results.

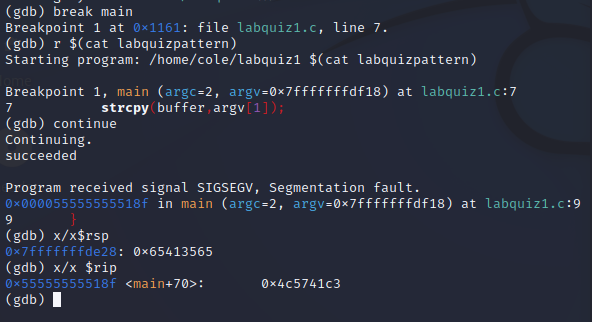
* 1. Modify the buffer size of the program **vuln.c** to 128. Save it as **labquiz1.c** and compile it. Attach screen capture of modified program called **labquiz1.c** and results after compilation



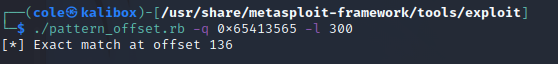
* 1. Use metasploit or peda debugger tools to create a pattern that generates overflow called the **labquizpattern.** Attach screen capture that demonstrates pattern created



* 1. Use gdb and run the program with the pattern generated. Attach screen capture that demonstrates results and the value of RIP.



* 1. Find the address of the last element of the stack and use metasploit or peda debugger tools to find the **offset** based on the address you found and attach screen capture that demonstrates the founded offset



* 1. If we want to inject a shellcode that is 32 bytes long, based on your offset result what will be the number of NOPs that can be written to the stack?

**Offset – 32**

**136 – 32**

**104**

* 1. Use one of the python scripts created in the Labs with **NOPs** and **32 bytes** long shellcode and modify it as follows:

1. Modify the number of NOPs with the offset you found.

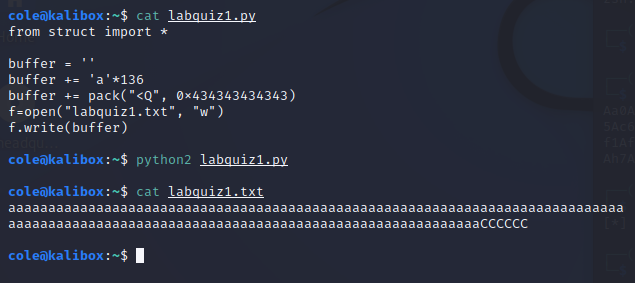
2. Overwrite RIP with **C (43)**

3. Send the results to a **labquiz.txt** save the script as **labquiz1.py**

and execute it.

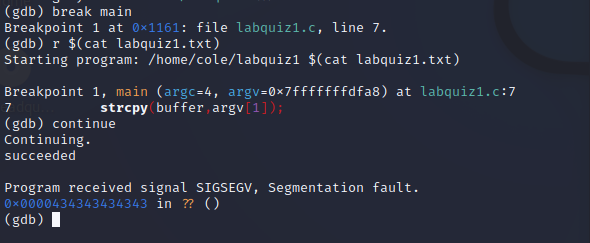
4. Attach screen capture of **labquiz1.py** and the results after

executing it



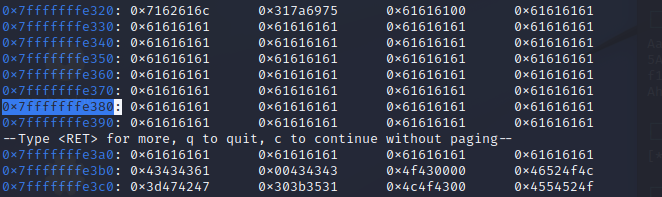
5. Use gdb debugger to inject the code (run the program with

**labquiz.txt** ) and attach screen capture that demo the results



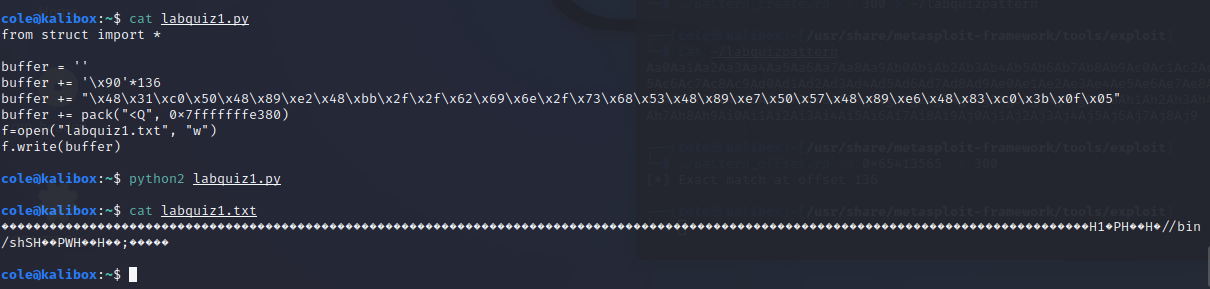
6. Attach a screen capture that demonstrates the **NOP area and**

**overwritten RIP with value C(43).**



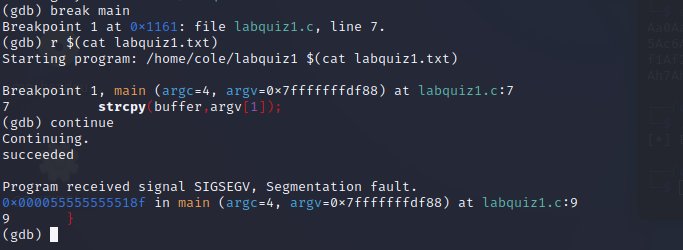
7. Replace RIP address on python script with selected NOP address.

Attach screen capture of modified python script **labquiz1.py**



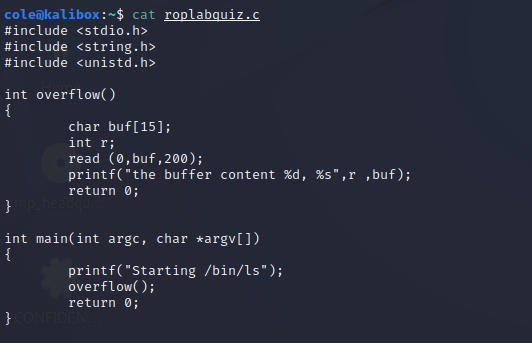
8. Use the debugger and run the exploit. Attach the screen capture

that demonstrates final result

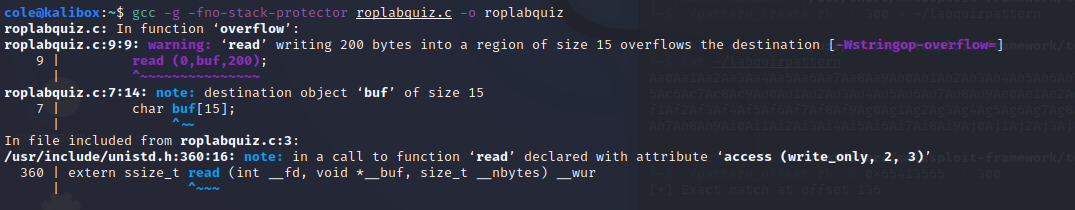


**No shell was created, however I followed the technique from the lab.**

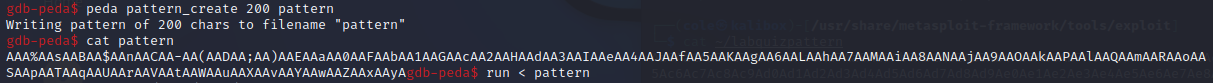
1. **( 20 marks)** Implement ROP technique:
   1. **(2 marks)** Modify the **rop.c code buf[15] and read () function** to read 200 buffer and attach screen capture of modified program



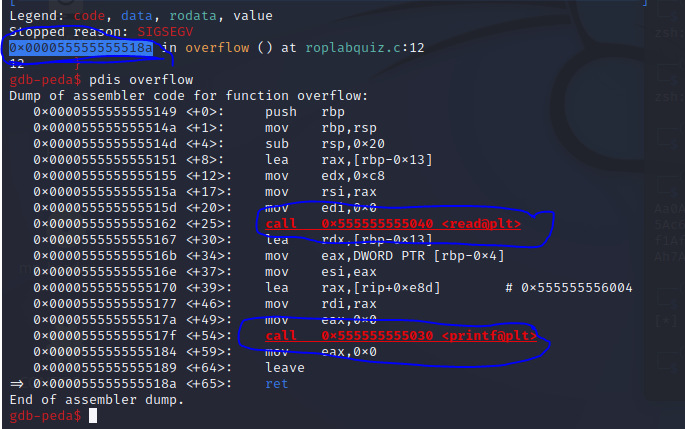
* 1. **(1 mark** **)**Compile the program as **roplabquiz.c with NX enabled** and attach screen capture that demonstrate results



* 1. **(1 mark)** Create a pattern that generates overflow and attach the screen capture with generated pattern

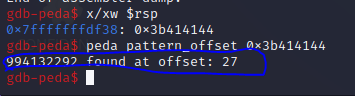


* 1. **( 3 mark)** Identify return address **(ret)** and address of called functions **read()** and **printf()** and attach screen capture underlining the 3 addresses



* 1. **(2 mark)** find RIP offset and attach the screen capture underlining found

RIP offset



* 1. **(3 marks)** Functions (libc) such as execve() or system() are very common

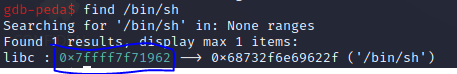
in ROP technique. Use execve() function and the first argument /bin/sh.

Find the address required to exploit using ROP

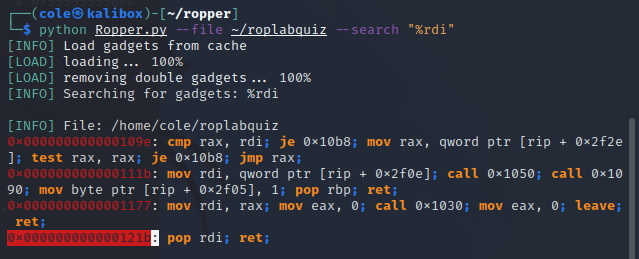
1. execve address

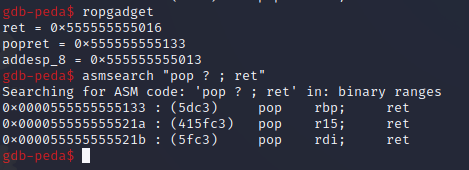


1. /bin/sh address

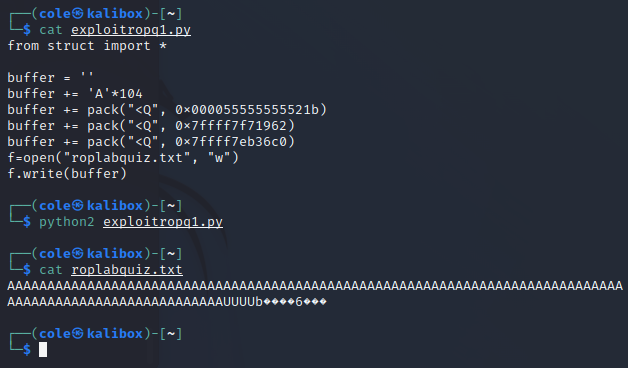


1. ROP gadgets address

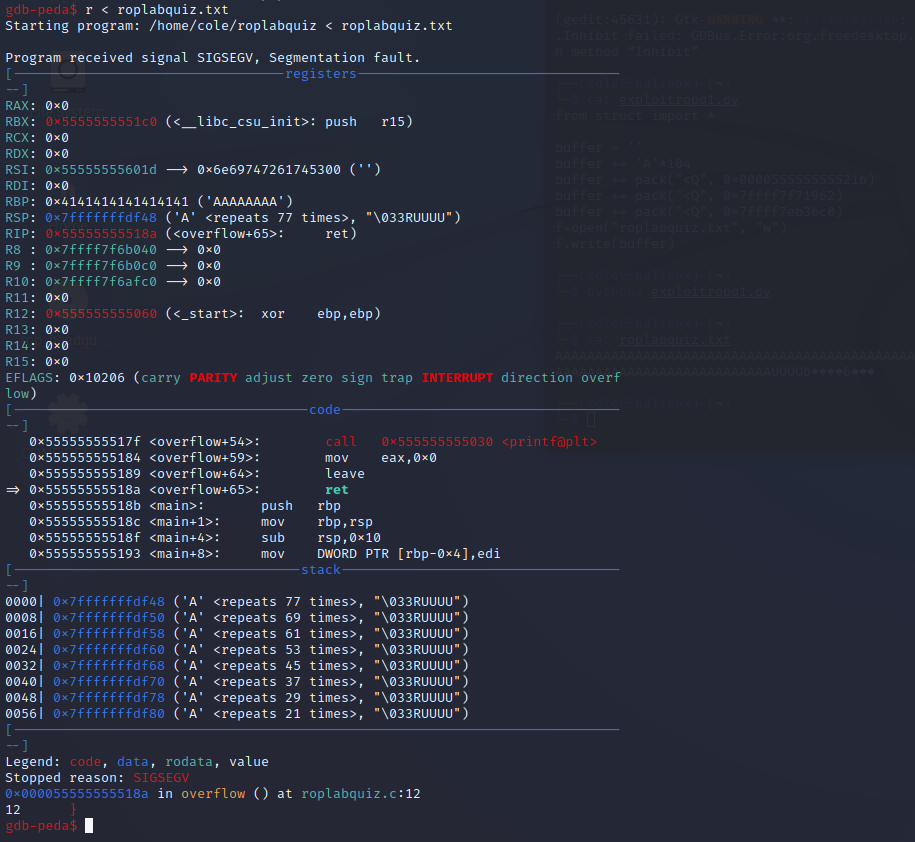




1. Attach the screen captures underlining found addresses
   1. **(5 marks)** Create a python script called **exploitropq1.py** with the respective **offset** and **addresses** and generate the **roplabquiz.txt file.** Attach screen captures with modified python script and generated file roplabquiz.txt



* 1. **(3 marks)** Run the exploit and attach the screen capture that demos final results



**NOTE (again) that the exploit didn’t work even though I followed the lab instructions.**