# Notes from Exploring Binaries

## Instructions

Make a copy of this document, rename it to “exploring-binaries-notes” and move it to your CSE 523 Google Docs collection. If at any point in this exercise you feel stuck, raise your hand and get some guidance. When you reach each GATE below, switch over to the Tracking Progress document and update your position. Try to be efficient with your time.

## Overview

Today we will consider how to approach new binaries for the purpose of discovering vulnerabilities. This will give us an opportunity to revisit earlier work. Keep detailed notes below (place your comments in between the provided horizontal lines); you will be referring to these in the future to do your work.

We will be working in your CSE 523 Ubuntu VM, so start that now and open a terminal window.

**Gate 1**

First, you will review the in-class exercises that we have worked through in the past. Look over your notes from previous classes (ie, your previous exploring-\*-notes documents in Google Docs). For each document that taught you something about how A) to discover vulnerabilities or B) to develop exploits, list the document name and a provide a brief description of the lesson taught. Think of this as your private topic index into our past in-class exercises.

Exercise 12 exploring vulnerabilities

That lesson teaches about the setup of virualbox and using msfconsole to nmap to manipulate the virtual machine’s record of hosts and services. We also used smb\_version in another machine to connect ot the host.

**Gate 2**

In one of the exercises above, you learned that one way to discover a vulnerability is to determine whether you could provide an input that would cause the program to crash. (That’s not the last step in the process, but it is certainly one of the important early ones.) The process of providing inputs was handled manually.

Create a new directory titled “binaries”. Enter it, and inside copy the binary “~/stack\_addresses/ans\_check5” into the new directory. Consider the following script.

import subprocess, os

# The following variables control the command line

program = "./ans\_check5 "

arg\_pattern = 'a'

pattern\_max = 10

for i in range(pattern\_max):

print "Trying input with length", i

cs = program + " " + arg\_pattern\*i

print "Command: %s" % cs

print "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

proc = subprocess.Popen([cs], shell=True,

stdin=subprocess.PIPE,

stdout=subprocess.PIPE)

print proc.communicate()[0]

print "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

print "Return value: %i, %s" % (proc.returncode,

os.strerror(proc.returncode))

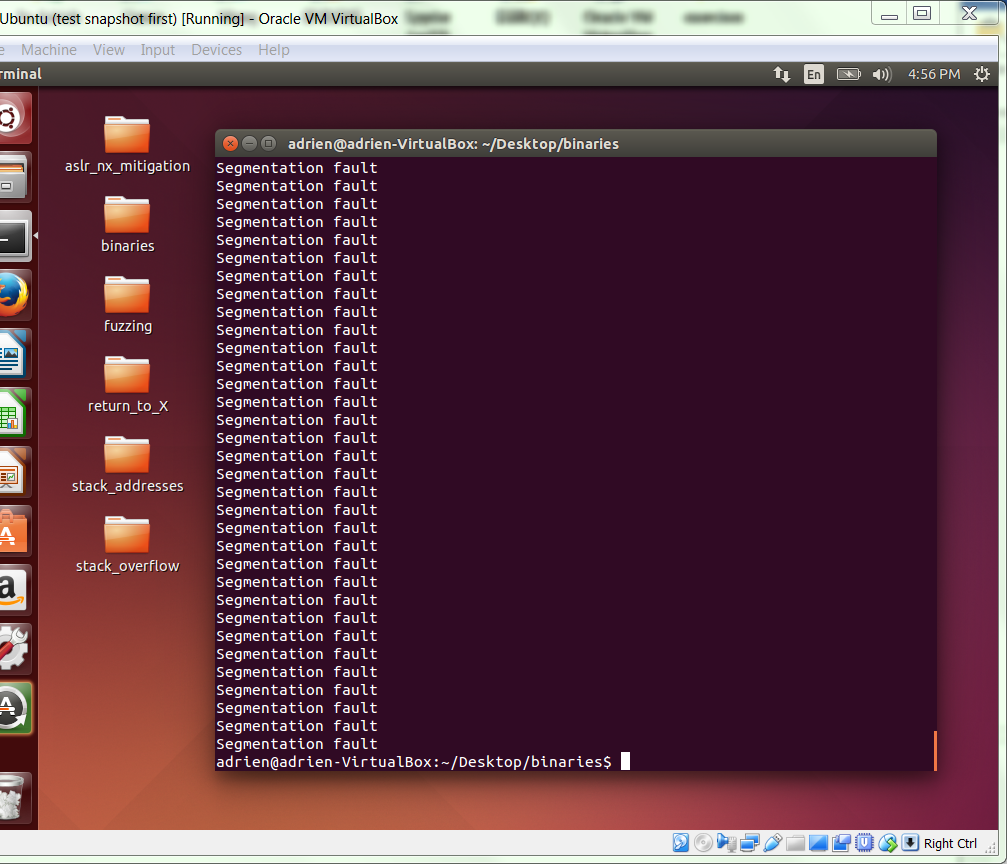
print

Store the text above in a file called fiz\_beta.py. You can invoke the script as follows.

python fiz\_beta.py > output.txt

Can you see how this script might be used to automate program runs, to discover crash inputs? Change the script so that it causes ans\_check5 to crash at least once. Include the lines that you changed below.

Pattern\_max = 10 to pattern\_max = 100



**Gate 3**

Create a new version of fiz\_beta.py, called fiz.py, that accepts program, arg\_pattern, and pattern\_max parameters as command line parameters. Include your new source code below.

import subprocess, os, sys

# The following variables control the command line

program = sys.argv[1]

arg\_pattern = sys.argv[2]

pattern\_max = int(sys.argv[3])

for i in range(pattern\_max):

print "Trying input with length", i

cs = program + " " + arg\_pattern\*i

print "Command: %s" % cs

print "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

proc = subprocess.Popen([cs], shell=True,

stdin=subprocess.PIPE,

stdout=subprocess.PIPE)

print proc.communicate()[0]

print "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

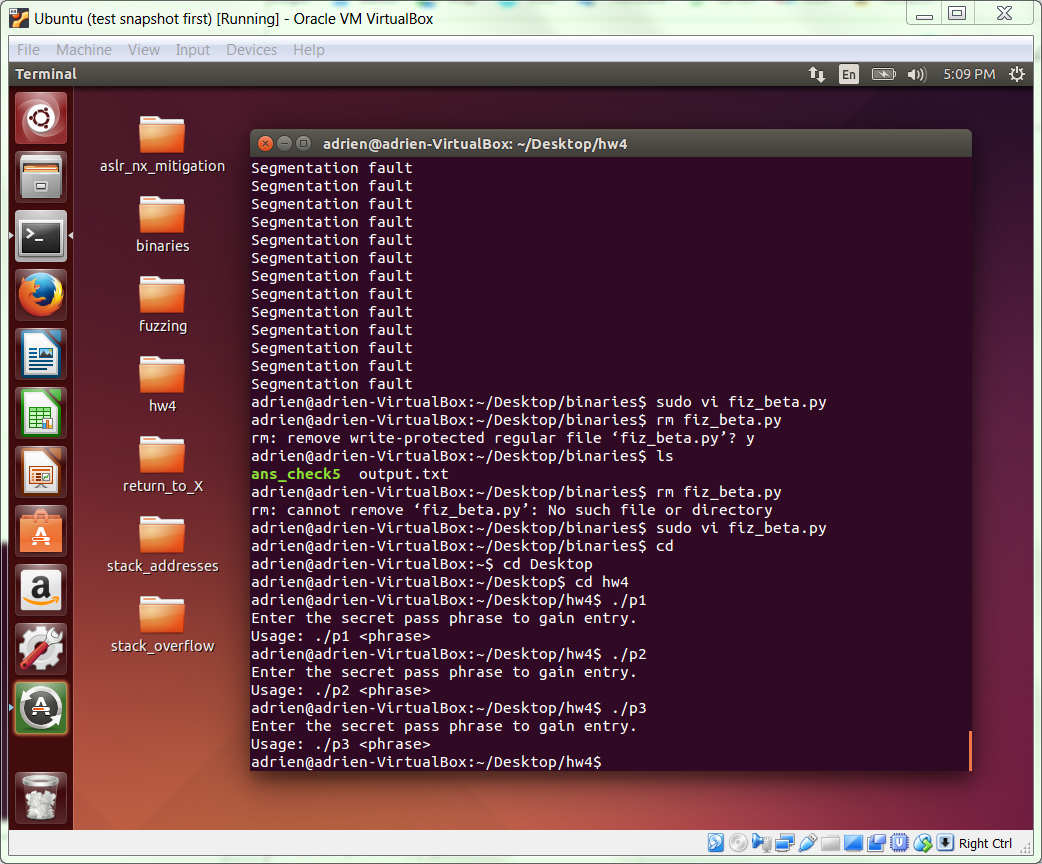
print "Return value: %i, %s" % (proc.returncode,

os.strerror(proc.returncode))

print

**Gate 4**

Open [hw4](https://drive.google.com/file/d/1wO-DDjk4lQQCiSMB8t-Ten33JmbCj6Ky/view?usp=sharing) and, following the instructions in that document, make sure you can download and run the 3 binary programs that are part of the assignment. This is at the top of the “Exploit Instructions” section of the document. Provide a transcript below that shows you invoking each binary without arguments.



**Gate 5**

At this point, you should be set up for both sections of Homework 4. Get started early! It is the largest homework of the semester.

**COMPLETE**