Apply fuzzing techniques

Lab Manual

Objectives

- Explain the concept of “fuzzing”
- Distinguish bug classes
- write code to implement a buffer overflows
- Use integer overflows
- Use format string attacks
- Create several types of fuzzers

Table of Contents

1.

Background Reading

Textbook:

Grey Hat Python, chapter 8

Other resources:

https://www.sans.org/reading-room/whitepapers/malicious/basic-reverse-engineering-immunity-debugger-36982
https://sgros-students.blogspot.ca/2014/05/immunity-debugger-basics-part-1.html

Notes common to all lab and home assignment problems

For every lab and home assignment, all work should go into your personal repository, subdirectory named mXX, where XX stands for the module number. For each problem, carefully name the program as described. The programs are extracted from your repository by a Python script, and errors in the program name will result in the instructor never seeing your program, and your mark for it will be ZERO!

Anything to record and report in this lab is to be written in plain text file (Word document is not a plain text file). There shall be one file per problem, and it MUST be named **`mXXpYYreport.txt`**, where the XX is module number, and YY is problem number. Mis-naming this file, or not having it in the proper location will result in mark ZERO for anything to be recorded or reported.

There are always many ways how to solve a programming problem, and usually one or two ways which are fast, compact and elegant.

Make sure to push your work to the server often, and have pushed the working version of the program by the deadline specified. The script extracting your programs from your repository will be run at any time after the deadline.

Lab specific intro

In this lab, we will explore the concept of “fuzzing”. We will generate malformed data, send it to the application under attack, and observe the behaviour of the application. With some luck, the application under attack will crash repeatedly, and we will learn what data patterns make it crash. We will explore several bug classes: buffer overflows, integer overflows, and format string attacks. We will create program **`file_fuzzer.py`** to run as a simple fuzzer.

Problem 1

In module 7, we implemented a buffer overflow on stack in C, and then exploited that code to start

calc.exe. Write a code in C which will generate heap overflow, name it **`heapover.c`**. Run the code under debugger control, and observe where it crashes. Observe what caused the crash, and report annotated screenshots of the debugger. Save the screenshots in files named heapoverXXX.png, where XXX is the sequence number of the screenshot.

Problem 2

Read the chapter 8.1.2 in Grey Hat Python book, describe the sequence of events which will lead to integer overflow and compromising the heap.

Problem 3

Write C program, name it **`m10p03.c`**, which will allocate 3 blocks of heap of 100 bytes in size, then overflow the middle block. Print the allocated adresses, and inspect the heap before and after the overflow. Describe what you found in file **`m10p03report.txt`**

Problem 4

Study the file fuzzer in chapter 8.2. Demonstrate your understanding of its functionality in your own words in file **`m10p04report.txt.`** Reference the line numbers in the attached printout of file_fuzzer.py in your description.

Problem 5, bonus mark .2

Have a 10 minute talk ready on how you solved the highest of the problems. Show good understanding of the problem and of your attempted solution. Be prepared to answer questions from the class and the instructor.