CSC 495/583 Fall 2019 Lab 3

Dr. Si Chen

October 13, 2019

Return-oriented programming (ROP)

The goals of this lab:

- Understanding the concepts of Return-oriented programming (ROP)
- Exploiting a Return-oriented programming (ROP) vulnerability

Objectives and Targets

Target 1: Understanding ROP (20%)

Please **draw a diagrams to illustrate the structure of your ROP shellcode for lab3. Explain why your shellcode will work and how it forms the ROP chain.** In particular, please include the following in your diagram:

- The order of dummy characters, parameters (if applicable), return address, and ROP gadgets
- Length of the dummy characters to reach return address

Target 2: Launch Return-oriented programming (ROP) Attack (80%)

The provided C code (lab3.c) contains a stack buffer overflow vulnerability. Please create a ROP gadget chain. The high level idea is to overwrite the return address with the address of function add_bin() and then create ROP chain to execute add_bash() and exec_string() sequentially. You should get a new shell (e.g. executing a linux command '/bin/bash') if success.

We have provided you with a virtual machine image for this project, use the latest version of VirtualBox. Our VM's image link can be found on our course website:
https://www.cs.wcupa.edu/schen/ss2019/

Steps:

1. Turn off ASLR.
2. Download the provided C code from our course website inside virtual machine, open it with any code editor.
3. Compile the provided C code (which you will be exploiting):

   ```
   gcc -m32 -fno-stack-protector -no-pie lab3.c -o lab3
   ```

4. To run this program, type the following command:

   ```
   ./lab3
   ```

5. When you type a very long list of characters, you will notice lab1 crashes with memory segfault, this is because the return address has been overwritten by your data.
6. Now you can craft your shellcode using the provided skeleton Python script. Again, your goal is to overwrite the return address with the address of function **add_bin()** and then create ROP chain to execute **add_bash()** and **exec_string()**. **You also need to prepare proper arguments and store them in the stack before calling target function**.

   GDB can be used to find these library addresses and test/debug your exploit. However, it should be noted that your final exploit (i.e., the final version of your Python script) should work outside of GDB. Just running

   ```
   python lab3_exp.py
   ```

   You should get a new shell (e.g. executing a linux command '/bin/bash') if success.
7. You can use **ROPgadget** to help you find the gadget that needed.

   ```
   ROPgadget --binary lab3
   ```

8. Provide a screenshot of the execution result.
9. Have fun.

Deliverables: the Python script you crafted and a screenshot of the exploit. The string of your shellcode and the screenshot should be put into the PDF file.

Submission

- The assignment should be submitted to D2L directly.
- Your submission should include: A **detailed project report in PDF format** to describe what you have done, including screenshots and code snippets.
- **No copy or cheating is tolerated**. If your work is based on others', please give clear attribution. Otherwise, you **WILL FAIL** this course.