CSC 495/583 Fall 2017 Lab 2

Dr. Si Chen

Due on: 10/22/2017

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: Launch Return-oriented programming (ROP) Attack

The provided C code (lab2.c) contains a stack buffer overflow vulnerability. Please create a ROP gadget chain (by modifying data.txt). 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. We do not recommend using your own VM image. Our VM's image link can be found on our course website: https://www.cs.wcupa.edu/schen/security/

We suggest using wget to ensure that you've downloaded the file correctly -
wget -c https://www.cs.wcupa.edu/schen/security/download/arch32.ova

Steps:

0). Turn off ASLR. (check step 1-3 in lab1 walkthrough)

https://www.cs.wcupa.edu/schen/security/lab1.html

1) Download the provided C code from our course website inside virtual machine, open it with any code editor.

2) Compile the provided C code (which you will be exploiting):

```
gcc -m32 -fno-stack-protector lab2.c -o lab2
```

3) To run this program, create a new file called data.txt and put some string literals in it, and execute lab1 by:

```
./lab2 < data.txt
```

4) When you put a very long list of characters in data.txt, you will notice lab1 crashes with memory segfault, this is because the return address has been overwritten by your data.

5) Now you can craft your shellcode in data.txt. 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 data.txt) should work outside of GDB. Just running

```
./lab2 < data.txt
```

You should get a new shell (e.g. executing a linux command '/bin/bash') if success.

6) Provide a screenshot of you exploiting sort.

7) Have fun.

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

More...

A detailed guidance will be available 2 days before the due date.

Submission

- The project is due by 10/22/2017. Late submission will not be accepted;
- 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.