CSC 497/583 Fall 2019 Lab 2

Dr. Si Chen

September 23, 2019

Stack Overflow

The goals of this lab:

- Understanding the concepts of stack overflow
- Exploiting a stack buffer overflow vulnerability

Students should be able to clearly explain: 1) what is buffer overflow; 2) why buffer overflow is dangerous; 3) how to exploit a buffer overflow. With the knowledge about buffer overflow, students are expected to launch an attack that exploits a stack buffer overflow vulnerability in a provided toy program.

Our course webpage: https://www.cs.wcupa.edu/schen/ss2019/

Objectives and Targets

Targets 1: Understanding Buffer Overflow (40%)

Note: For this task, you may use online resources to show a program with buffer overflow vulnerability, but please cite these online sources. The diagrams should be your own (not copied from the online resources).

Write a testing program (not lab2.c from Targets 2) that contains a stack buffer overflow vulnerability. Show what the stack layout looks like and explain how to exploit it. In particular, please include in your diagram:

1. The order of parameters (if applicable), return address, saved registers (if applicable), and local variable(s),
2. Their sizes in bytes,
3. Size of the overflowing buffer to reach return address,
4. The overflow direction in the stack.

You are not required to write the real exploit code, but you may want to use some figures to make your description clear and concise.

Deliverable: a pdf file containing your vulnerable programs (paste your code into the pdf directly) and your explanations. Submit to D2L

Targets 2: Buffer Overflow Demo (60%)

The provided C code (lab2.c) contains a stack buffer overflow vulnerability. Please write an exploit (by modifying data.txt) to output **"hacked by YOUNAME!!!!" (replace YOUNAME by your name)** on Linux. The high level idea is to overwrite the return address with the address of function hacked(). Once the return instruction is executed, this function will be called and output the string.

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/ss2019/

We suggest using wget to ensure that you've downloaded the file correctly -

```
wget -c https://www.cs.wcupa.edu/schen/ss2019/download/M64.ova
```

Steps:

1. Import the OVA file to VirtualBox.
2. Download the provided C code from our course website inside virtual machine, open it with any code editor, make changes
   - Please set the array (char array[]) size **equal to the last two digits of your student ID** (e.g. 0861339 $->$ array size should set to 39)
   - Change the string literal YOUNAME inside "hacked" function to your full name (e.g. Si Chen).
3. Turn off ASLR (check slides – ch07.pptx)
4. Compile the provided C code (which you will be exploiting):

   ```
   gcc lab2.c -o lab2 -m32 -fno-stack-protector -zexecstack -no-pie
   ```

5. When you input a very long list of character, you will notice lab2 crashes with memory segfault, this is because the return address has been overwritten by your data.
6. Now you can craft your shellcode in exploit.py. Again, your goal is to overwrite the return address with the address of function **hacked()**. GDB can be used to find these library addresses and test/debug your exploit.
7. Provide a screenshot of you exploiting sort.
8. Have fun.

Deliverables: A screenshot of the exploit script (exploit.py) and a screenshot of the result after executing your script. The two screenshots should be put into the PDF file (the same from Targets 1).

More...

Check ch06.pptx, ch07.pptx (on our course website) for more details.

Submission

- The project due date is on our course website. 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.