CSC 472 Fall 2024 Lab 2

Dr. Si Chen

September 27, 2024

Stack Overflow

Lab Objectives:

- Understand the fundamentals of stack overflows
- Learn how to exploit a stack buffer overflow vulnerability

Upon completing this lab, students should be capable of articulating:

1. The nature of stack overflows
2. The risks associated with stack overflows
3. The methodology for exploiting a stack overflow vulnerability

Equipped with this understanding, students are expected to successfully execute an attack that exploits a stack buffer overflow vulnerability in a given toy program.

Course Webpage: https://www.cs.wcupa.edu/schen/ss2024/

Return Hijack Attack (10 points)

The C code provided (‘lab2.c‘) contains a stack buffer overflow vulnerability. Your task is to write an exploit script in Python using the pwntools library. The script should output **“hacked by [Your First + Last Name]!”** when run on a Linux system. The overarching strategy is to overwrite the return address with the address of the ‘hacked()‘ function. Upon execution of the return instruction, this function will be invoked, producing the specified output.

Please use our Badger CTF platform to complete this assignment.

Steps:

1. Log in to Badger CTF using your personal account.

2. Download the provided C code from the course website to the folder '/workdir/'. To copy the source code after successfully logging in to Badger CTF, execute the following command:

```
cp /workdir/ss2024/lab2/lab2.c /workdir/
```

3. Open the source code file 'lab2.c' using a command-line text editor (e.g., Vim, nano) and implement the following changes:
   - Set the size of the array ('char array[]') to **the last two digits of your student ID**. For example, if your student ID is '0861339', set the array size to '39'. **Note:** If the last two digits of your ID are less than 10 (e.g., '05'), use the last three digits of your student ID as the array size.
   - Replace the string literal 'YOUNAME' in the 'hacked' function with your actual name (e.g., 'Si Chen').
4. Compile the C code using the following command:

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

5. After running the program and inputting a long list of characters, you may notice that lab2 crashes due to a memory segmentation fault. This occurs because the return address has been overwritten by your data.
6. Craft your shellcode using Python and the pwntools library. A script template is available on our course website for your convenience. Your objective is to overwrite the return address with the address of the 'hacked()' function. You can use GDB to find these library addresses and to test or debug your exploit.
7. Provide a screenshot demonstrating your successful exploitation of the program.
8. Have fun.

Deliverables: Submit a detailed lab report including a screenshot of your exploit script (named 'exploit.py') along with a screenshot displaying the results of successful script execution. Additionally, provide an explanation detailing the method used to determine the **Magic Number**—the length of the dummy-character string required to successfully overwrite the return address.

Return to Shellcode Attack (Bonus: 2 points)

For this bonus challenge, attempt to execute a return-to-shellcode attack similar to the class example provided in "Overflow2.c". Apply this attack to your "lab2" program. Upon successfully gaining shell access, execute the commands `whoami` and `date`.

Deliverables for this Section: Embed a screenshot within your Python script to document the successful exploit. Additionally, capture a screenshot showcasing the shell interface after successfully obtaining shell access and executing the specified commands.

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. [Report Format Requirement]
- **No copy or cheating is tolerated**. If your work is based on others' or AI, please give clear attribution. Otherwise, you **WILL FAIL** this course.