Network Security

Project 3

Buffer Overflow

Instructor: Shiuhpyng Shieh

TA: Po-Hsing Wu & Te-Yu Chang

Introduction

This project will introduce you to control-flow hijacking vulnerabilities in application software, including buffer overflows. We have provided source code, binary executable, and the assembly code. Try to find vulnerabilities in these files and exploit it to get the secret flag.

Objectives

• Be able to identify and avoid buffer overflow vulnerabilities in native code.
• Understand the severity of buffer overflows and the necessity of standard defenses.
• Gain familiarity with machine architecture and assembly language.

Read this First

This project asks you to launch attacks on a remote server we control. Attempting the same kinds of attacks against others' systems without authorization is prohibited by law and university policies and may result in fines, expulsion, and jail time**. You MUST NOT attack anyone else's system without authorization!** Per the course ethics policy, you are required to respect the privacy and property rights of others at all times, or else you will fail in the course.

Getting Files

You can obtain your files from the following url

`140.113.194.78:8080/<student_id>.zip`

where <student_id> is your student ID. For example, if your student ID is 0656001, your url would be `140.113.194.78:8080/0656001.zip`

The files in 0656001.zip include:

`a.out:`	An executable program(32bit ELF) vulnerable to buffer overflow attacks
`vulnerable.c:`	The source code of your `a.out`
`vulnerable.asm:`	The assembly code of your `a.out`

Project Description

`a.out` takes inputs from stdin and exits immediately. Your job is to provide inputs that makes it output the flag. Your input will need to overwrite the return address so that the function `main()` transfers control to `magic()` when it returns.

Target Machine

We run your `a.out` on `140.113.194.78:`xxxxx. Please find your own xxxxx in `std_port_list.txt`. We are going to refer to this vulnerable service as "target". After you figure out how to overwrite the return address on your local machine, try to send your payload to the target machine. Once your payload successfully changes the control flow, you will receive the secret flag from the target machine, which is the answer of this project. The flag is in the following format

```
FLAG{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}
```

Resources

GDB

You will make extensive use of the GDB debugger. Useful commands that you may not know are "`disassemble`", "`info reg`", "`x`", and setting breakpoints. See the GDB help for details, and don't be afraid to experiment! This quick reference may also be useful: http://csapp.cs.cmu.edu/2e/docs/gdbnotes-ia32.pdf

x86 Assembly

These are many good references for Intel assembly language, but note that this project targets the 32-bit x86 ISA. The stack is organized differently in x86 and x86_64. If you are reading any online documentation, ensure that it is based on the x86 architecture, not x86_64. Here is one reference to the syntax that we are using: https://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html#s3

Deliverables

Each student must work individually and submit a .zip file, named by "<STUDENT_ID>.zip", for example "0656001.zip", containing:

1. `flag.txt` – a text file contains the secret flag you got.
2. `Writeup.pdf` – a report in PDF format about how to get the flag.