Network Security

Project 3

Buffer Overflow

Instructor: Shiuhpyng Shieh

TA: Wei-Ti Su & Xin-Yu Wang

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 vulnerability in these files and exploit it to get the secret flag. There are two flags: flag1 & flag2. If you get flag1, you will score 100, flag2 is just the bonus flag. Have fun!

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 the course.

Project Description

This program create some structure and provide you some functionality to manipulate these structure. For example, you can view and edit the structure. But the program generate a secret random number to protect the structure, so you need to know it to edit the structure. Your goal is to edit the structure to overwrite the return address and hijack the program flow.

Server

Server is running on 140.113.194.66, port 8787.

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. flag1.txt (30%) – a text file contains the secret flag you got.
2. Writeup.pdf (70%) – a report in PDF format about how to get the flag.
3. flag2.txt (10%) - a text file contains flag2.