Secure Programming Laboratory 1: Introduction

Arthur Chan and David Aspinall, Informatics @ Edinburgh

7th February 2017

Orientation

This is the first Laboratory Session for **Secure Programming**

It is convened by **Arthur Chan**, **Rui Li** and **David Aspinall**.

Please take a copy of the **handout**. It is also available online via the course web page.

What is this lab about?

There are 5 exercises showing exploits which result from **data corruption** as we have studied in Lectures 3-6.

- **Exercise 0.** Effective user id, basic stack overflows (Basic stuff)
- **Exercise 1.** Stack overflows, corrupting memory.
- **Exercise 2.** Corrupting a network protocol.
- **Exercise 3.** A more advanced buffer overflow.
- **Exercise 4.** A real-life data handling flaw in a version of OpenSSL.

The exercises are based on executable and source files provided in a Virtual Machine running Linux.

What do we hope you will learn?

- Understanding of some basics for how low-level attacks work, including manipulating executable code and memory.
- Practice thinking about program flaws in C and Java, and how they may be exploited.
- Use of some simple tools to help investigate or find flaws.
- Experience repairing or preventing some attacks, and checking to see that they have been stopped.

Resources

- Use **anything**! You are encouraged to search on the web for help, tutorials, manuals, etc.
- You can get plenty of help this way. But it is probably more rewarding to try to solve the exercises for yourself first. Make sure to spend time experimenting, not only reading.
- **Warning**: experiment with care! If you download sample exploits, generation tools, etc, install and run these in the Virtual Machine, **not on the host DICE environment**. The VM already has several interesting tools provided.
- **Ask us!** We are here to help, as much as we can.
- **Ask each other!** There may be expert x86 programmers, C hackers, exploit developers(?) among you. . .

Timing

You probably won’t have time to complete all exercises in this lab session.

- Don’t worry!
- Exercise 1, especially, could take a long time depending on how much you know already and how much you do.
- Exercise 3 is optional and advanced, only recommended if you found Exercise 1 easy.
- Exercise 4 involves a large software package, it will be difficult to understand the context.
- Of course, you can spend more of your own time later if you are interested, but completing the lab is not essential. At least, try to look at each exercise a little bit, and review the solutions when they are released. The important thing is to understand the concepts well.

Discussion

We use Piazza for discussion and follow-up questions on-line after the lab, but during the lab we will provide individual help and guidance, and also make announcements during the lab with hints and tips.

Feedback on your work

Besides using Piazza, discussing with us, we will give additional brief feedback *at the next lab* based on **submitted work**.

Submission is optional.

Each exercise has a series of **Checkpoints** which are questions that you can provide brief answers to. A plain text file checkpoints.md is provided, please fill this in and submit it electronically. The main point of the checkpoints is for you to check your own understanding, so please **don't spent time writing long or very polished answers!**.

Personalised feedback will only be available on submissions made by the deadline of **Monday 13th Feb, 4pm**. This is to discourage you from spending too much time.

Useful pointers

For Exercise 1:

- If you don't already know GDB, try out a tutorial or two on the web, such as unknownroad.com's examples.
- You can also find tutorials on crafting stack overflow exploits, for example here and, the original Aleph One article. Beware that older articles like these may not work exactly as written.

For all examples:

- Study the lecture slides from Lectures 3-6.

Good Luck!

We hope you enjoy the lab.