Secure Programming Coursework Part 1

Arthur Chan and David Aspinall

School of Informatics, University of Edinburgh

This is an **individual** assessed practical exercise. It is the only assessed coursework for the Secure Programming course. It consists of two parts. Part 1 is issued first and covers the earlier portion of the course. Part 2 will be issued later and covers lectures and lab exercises yet to come. Provided you have attended the relevant lectures and lab sessions in the course, the work for both parts should take about 30 hours. The practical will be awarded a mark out of 100. The single deadline for submission (both parts) is **5pm, Fri 15th November 2019**. The *recommended* deadline to complete this part is **Fri 25th October 2019**. The final page summarises the submission instructions.

Virtual machinery

We provide a virtual machine for you to use. The VM has two users, `user` and `root`. The **passwords are the the same as their usernames**.

To install the VM, you should use a virtual disk file stored on a local disk on your machine. If you are working in the Appleton Tower Lab on the DICE machines, you can use the directory `/tmp/sNNNNNNN` if there is enough space. Configure VirtualBox to use the right disk area by setting `File → Preferences → General → Default Machine Folder`. Next, **import the appliance** (`File → Import Appliance`) from the file:

```
/afs/inf.ed.ac.uk/group/teaching/module-sp/SecureProgramming-Coursework.ova
```

If you are working remotely or on your own machine, we recommend taking a copy of the `.ova` file first rather than importing over directly from AFS. The file is about 1G. It may be more convenient to use a USB stick than download it over the Internet.

Important: make sure that your VM disk files are stored in a directory which is only readable by you. Beware that `/tmp` are disk areas which are not backed up. So if you are using a lab machine (and anyway, for safety), **back up your work** by saving any work that you do inside the virtual machine (edited source files, etc) in your home directory.

Using the machine

The machine is set to use NAT. Once started you can either use the console window, or (recommended): SSH in via your local machine over port 2222, with: `ssh -p 2222 user@localhost`.

We've supplied some tools to make things easier but feel free to install additional software in the VM. In your answers, please describe **all tools you have used**, including Linux packages, browser plugins used in your host machine, etc.

1. RIDL and Fallout (25 marks)

In 2018, two critical vulnerabilities were discovered in modern processors after the Meltdown and Spectre incident. They are named as RIDL and Fallout and details in four CVE entries. This question asks you to consider these CVEs, CVE-2019-11091, CVE-2018-12126, CVE-2018-12127 and CVE-2018-12130. You should put your answers for this question in **answers1.pdf**.

Remark: Some researchers named the vulnerabilities as ZombieLoad or Microarchitectural Data Sampling (MDS)

1. Security researchers consider RIDL and Fallout inspired by previous speculative execution attacks, including Meltdown and Spectre. Study the CVE entries and related documentations for RIDL and Fallout, in your own words *briefly* describe what is a speculative execution attack and the similarities and differences of RIDL and Fallout comparing to Meltdown and Spectre. (4 marks)

2. There are four CVE entry describing RIDL and Fallout, please identify which of them is (are) describing RIDL and which of them is (are) describing Fallout. (2 marks)

3. Identify the Common Weaknesses (using CWEs) for these two vulnerabilities. Also identify which scope in the CIA triad has been violated by the common weaknesses and why. (4 marks)

4. Identify the possible consequences of these vulnerabilities and how an attacker can make use of the consequences. (4 marks)

5. These two vulnerabilities are considered to be "hardware" vulnerabilities. Briefly discuss the difference between hardware vulnerabilities and software vulnerabilities and how you might draw a distinction. To help your explanation, give an example of one software vulnerability and one other hardware vulnerability that are listed in the NVD (https://nvd.nist.gov) and occurred in the last two years. (4 marks)

   Remark: No marks will be given if you give an example of RIDL/Fallout/Meldown/Spectre.

> *"RIDL and Fallout are so dangerous. After I heard about these vulnerabilities, I did a full scan of my computer by the latest anti-virus software and installed all application updates. The anti-virus protection does not find any problems. I have also installed a personal firewall in my computer. So I strongly believe that my computer should be secure enough to defend RIDL and Fallout attacks."*
>
> *By Mr. Super Secure*

6. State your opinion and discuss whether you agree or disagree with Mr. Super Secure's words. Provide some reasons to support your opinion, explaining the role of PC security software mentioned. (5 marks)

7. Apart from installing the above mentioned security software, please describe two possible measures to protect your computer from RIDL and Fallout attacks. (2 marks)

2. Secure Coding (25 marks)

Code for this question is in the `/home/user/exploit` folder on the VM. You are given a program called **vulnerable** which is compiled from **vulnerable.c** (to recompile the program, simply use the **make** command). The program takes a user name and message from the command line and writes into a message board file. There are two message board files, **board1** and **board2** in the directory. By default, the program will store the message in **board1**, and **board2** is a private message board that is reserved for private use. Unfortunately, the program contains a vulnerability that allows an attacker to write to the private **board2**.

Please put all your written text-based answers in **answers2.pdf** for this question.

1. Identify the vulnerability in the program **vulnerable** and give the name of a CWE which categorises the vulnerability type most closely. Briefly explain the vulnerability and identify in general how and why can an attacker abuse it. (3 marks)

2. A successful attack should allow an attacker to store the message in **board2** instead of **board1**.

   Create an exploit script called **exploit** that takes the path of the program as its first argument and launches a successful attack. We will run your program by executing:

   ```
   ./exploit ./vulnerable
   ```

   Remark: We will only execute your exploit by normal user account

   It must not output anything other than the output produced by the vulnerable program. You may use any scripting language to write your exploit, provided it runs as described. If you use a high level language, please also attach your original source code in **answers2.pdf**. If your exploit cannot be run as described (files missing or execution errors), no marks will be awarded for this part. We will execute your code in a fresh VM copy of the machine imported into Virtual Box. (7 marks)

3. Please briefly explain your **exploit** script created for the last question and describe how it abuses the **vulnerable** program to force it to store your message in **board2** instead of **board1**. If you use some hard-coded values, please also explain how you got those values. (2 marks)

4. Provide a patch file that fixes the vulnerability of **vulnerable.c**. The patch file should be named as **question2a.diff**. (1 marks)

5. There is another program **vulnerable2.c** with multiple vulnerabilities. Perform a code review and report up to three *different* vulnerabilities in this second program.

   For each vulnerability, describe what the problem is, how it might be exploited, and what the possible consequences of an exploit might be. Finally, give a correction to the code to show how it may be fixed.

   You should provide your description and answers in **answers2.pdf** and provide a patch file that fixes your three reported vulnerabilities of **vulnerable2.c**. The patch file should be named as **question2b.diff**. (12 marks)

Note

Patch files can be created with the command

```
diff -c oldfile newfile  > question2x.diff
```

Keep a copy of the original file so you can make the patch file!

Reminder warning: *Never execute your exploits on a real machine* unless you have the express permission of the owners to do such testing.

Submission instructions (Part 1)

Remark: Submission instructions for part 2 will be released later.

You should submit your answers electronically with the command:

`submit sp cw` *filename*

or if you want to submit multiple files:

`submit sp cw` *filename filename* ...

Where *filename* is:

answers1.pdf A PDF document containing the answers to Question 1.

answers2.pdf A PDF document containing the answers to Question 2.

exploit The script required to exploit the program for Question 2.2.

question2a.diff The patch file generated for Question 2.4.

question2b.diff The patch file generated for Question 2.5.

Wrong *filename* arguments will not be accepted. The PDF documents should be well-formatted printable A4 PDFs, you may generate them with whatever program you want. Text answers should be brief and to-the-point.

Repeated submission of the same *filename* will overwrite the previous submission. Take care: the submission does not keep a history of submitted files, we will mark the most recent files and their submission timestamps must be before the deadline to avoid standard lateness penalties.

You must submit by the **final deadline 5pm, Fri 15th November 2019**.

The coursework is separated into two parts and released in stages, but both parts have the same final deadline. However, you are **strongly encouraged** to submit your answer for the first part before the second part is released, to help you manage your time. So:

It is recommended to submit for this part by **Friday 25th October 2019**.

You're reminded that late coursework is not allowed without "good reason", see the Informatics policy on coursework deadlines[1] for details, and the procedure to follow if you must submit late. In particular, if you have a good reason to submit late, use the Student Support Team contact button to make a request rather than asking us.

[1] http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursework-extension-requests