Secure Programming Coursework Part 1

Robert Flood 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 **12 noon, Fri 18th November 2022**. The *recommended* deadline to complete this part is **Fri 28th October 2022**. The final page summarises the submission instructions.

1. Log4Shell and Insecure Input (12 marks)

You should put your answers for this question in **answers1.pdf**.

1. The following code snippet contains a vulnerability. Please describe a) the base CWE number for this vulnerability and b) how this vulnerability may be exploited. (3 marks)

```
from flask import Flask, request

app = Flask(__name__)

CONFIG_DATA = {
        'NAME': 'TestScoreLog',
        'PASSWORD': 'my_s3cr3t_p4ssw0rd'
        }
# Process GET request
@app.route('/submit')
def log_score():
    coursework = request.args.get('coursework', '')
    student = request.args.get('student', '')
    score = request.args.get('score', '')
    response = "In {coursework}, {student} received mark of {score}".format(coursework=coursework,
                student=student, score=score) # Format output
    return format_response("[*] {CONFIG[NAME]} : " + response)

def format_response(log):
    return log.format(CONFIG=CONFIG_DATA)
```

2. Log4j is an open-source logging framework used by developers to log data from their applications. In 2021, Chen Zhaojun uncovered Log4Shell (CVE-2021-44228) a zero-day vulnerability in Log4j. Exploitation of Log4Shell relies on untrusted user input as well as insecure deserialisation. Study the CVE entry for Log4shell and, in your own words, describe the features of Log4j that caused this vulnerability as well as the attack itself. (3 marks)

3. The Log4shell vulnerability was introduced in in 2013, after a user requested that additional functionality be added to Log4j. Discuss the (potentially) competing interests of developers and security engineers when developing software (2 marks)

4. Log4shell was given a CVSS rating of 10, the highest score possible. What factors contributed to this high score? Make explicit reference to the CIA triad in your answer as well as the potential impact of an attack. (2 marks)

> "*As soon as I heard about Log4shell, I immediately checked if anyone has exploited it on my system. Using grep, I searched all of my logs for the Log4shell exploit string. Thankfully, I didn't find anything, so that means my system is secure and wasn't attacked.*"
>
> *By Mr. Super Secure*

5. In your opinion, is Mr. Super Secure's assessment accurate? Support your opinion with at least 2 points. In addition, describe two possible mitigations for Log4shell. (2 marks)

2. Secure Coding (12 marks)

You are given two programs called **vulnerable** and **vulnerable2** which are compiled from **vulnerable.c** and **vulnerable2.c** respectively (to recompile the programs, use the **make** command). These programs have been tested on DICE.

Please put your written answers in **answers2.pdf** for this question.

1. There is a program vulnerable.c, a toy program for parsing commandline arguments. It contains a major insecure coding practice.

   Identify the insecure coding practice in the program *vulnerable*, with reference to its CWE. Provide a short script called exploit that demonstrates that this program is problematic. We will run your script as: ./exploit. **Remark: Your script does *not* have to actually exploit the underlying vulnerability**. We are merely looking for a short (~1 line of code) script that crashes the program, indicating that similar code could be vulnerable. (2 marks)

2. Provide a patch file that fixes the vulnerability of **vulnerable.c**. The patch file should be named as question2a.diff. It must apply cleanly against the original **vulnerable2.c** with no superfluous flags. We will test your patch file by running

   ```
   patch < question2a.diff
   ```

   (1 mark)

3. Assuming that the insecure coding practice in **vulnerable.c** existed in a larger, feature-rich program, describe how it may be exploited. Make explicit reference to what the impact of an attack may be. Support your speculation with reference to at least one CVE(s) with the same CWE as you identified in Q2.1. (2 marks)

4. There is another program **vulnerable2.c**, a toy program for printing random arrays according to some conditions. It contains a major insecure coding practice.

   Identify the insecure coding practice in the program *vulnerable2*, with reference to its CWE, describing its root cause in detail. Provide a short script called exploit2 that demonstrates that this program is problematic. We will run your script as: ./exploit2. **Remark: Your script does *not* have to actually exploit the underlying vulnerability**. We are merely looking for a short (~1 line of code) script that crashes the program, indicating that similar code could be vulnerable. (2 marks)

5. Provide a patch file that fixes the insecure coding practice of **vulnerable2.c**. The patch file should be named as question2b.diff. It must apply cleanly against the original **vulnerable2.c** with no superfluous flags. We will test your patch file by running

   ```
   patch < question2b.diff
   ```

   (1 mark)

6. Assuming that the insecure coding practice in **vulnerable2.c** existed in a larger, feature-rich program, describe how it may be exploited. Make explicit reference to what the impact of an attack may be. Support your speculation with detailed reference to at least one CVE(s) with the same CWE as you identified in Q2.4. (2 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!

Submission instructions (Part 1)

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

Go to the SP Learn course and select "Assessment" from the left hand menu. Select the "Assignment Submission" folder and then the "Coursework (parts 1 and 2) folder. Click on the link "Submit via Gradescope". This will take you to the Gradescope interface. For anyone who has sat an online exam over the last two years, this should look familiar to you. From here, you can drag and drop your file(s) to submit.

Please name your files as follows:

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 for Question 2.1.

exploit2 The script required for Question 2.4.

question2a.diff The patch file generated for Question 2.2.

question2b.diff The patch file generated for Question 2.5.

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.

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 12 noon, Fri 18th November 2022**.

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 28th October 2022**.

You're reminded that late coursework is only allowed if approved by the University's central ESC Team, see the Informatics advice page for more details and how to apply.[1]

[1] https://web.inf.ed.ac.uk/infweb/student-services/taught-students/information-for-students/information-for-all-students/your-studies/late-coursework-extension-requests