**Course Name: Secure Software Engineering**

**Course Code: COMP SCI 4412 & COMP SCI 7412**

**Assessment Component: Exercise 3 (5%) – Individual Assessment**

**Release Date: 15/08/2019**

**Due Date: 13/10/2019 by 11:55pm.**

**Submission: MyUni Canvas.**

**The list of tasks for exercise 3**

1. Clone the GitHub repositories associated with 3 of your vulnerabilities (<https://bit.ly/SV_SSE2019>) to your computer. **In case you encounter any problem cloning the repositories, let us know as soon as possible**. You do not have to submit this part. Hint: git clone
2. The main task of exercise 3 is to identify the Vulnerability-Contributing Commits (VCCs) from fixing commits of 3 vulnerabilities you have found in exercise 1. **If the fixing commit provided was different from the one you found, then you should specify which one you choose and why.**
3. To identify VCC, for each affected (deleted, added) line of each file in the fixing commit of each vulnerability, perform the following steps and **write down what you have done for this task in the report with detailed explanation and screenshot**. If a commit has more than 5 affected files, please identify at most 5 files that you think are the most relevant ones for **fixing** the current vulnerability only.
   1. If a line was deleted, identify the latest commit that modified that line and record that commit. Hint: git blame
   2. If a line is added, first identify the smallest enclosing scope (e.g., class/interface, function, for, while, if, else, switch, try, catch, finally). In case there are also other relevant pieces of code that you think can also lead to the current added line, please specify and explain them in details. Then, identify the latest commit that modified the selected lines and record that commit. Hint: git blame and regular expression or abstract syntax tree parsing
   3. Then, select the most frequently identified commit as the VCC for the current fixing commit. In case there is more than one such commit with the same frequency, select all of them.
4. Develop an automated script to identify VCC (e.g., using GitPython and relevant libraries as demonstrated in the tutorial in Working Session 1). **Explain how you implement it in the report and attach a link to your code in your GitHub repository** (make sure it is public).
5. Study and repeat task 3 with different parameters of git blame: -w, -wM, -wC, -wCC. –wCCC. Give your thoughts about whether there is any change in the VCCs. **If the parameters produce different VCC,** **please mention which parameter you think give you the most satisfactory VCC(s).** **Write in the report your findings.**
6. Analyze each of your identified Vulnerability-Contributing Commit (VCC) in Tasks 4 and 5 and answer the following questions similarly to exercise 2. If a commit has more than 5 affected files, please identify at most 5 files that you think are the most relevant ones for **introducing** the current vulnerability only. The Git command for answering each question can be found in Hint.
   1. What was the message and title of the VCC? Was there any mention of fixing another bug or vulnerability? Hint: git diff or git show
   2. How many total files were affected in the current VCC? Hint: git diff or git show
   3. How many total directories were affected in the current VCC? For example, if a file path is: abc/def/File.java, then its directory is abc/def. Hint: git diff or git show
   4. How many total lines of code (**including** comments and blank lines) were deleted? Hint: git diff or git show
   5. How many total lines of code (**including** comments and blank lines) were added? Hint: git diff or git show
   6. How many total lines of code (**excluding** comments and blank lines) were deleted? Hint: git diff or git show
   7. How many total lines of code (**excluding** comments and blank lines) were added? Hint: git diff or git show
   8. How many days were between the current VCC and the previous commit of each affected file? Hint: git log
   9. How many time has each affected file of the current VCC been modified in the past since their creation (including rename of the file)? Hint: git log
   10. Which developers have modified each affected file since its creation? Hint: git log
   11. For each developer identified, how many commits have each of them submitted? From your observation, are the involving developers experienced (with many commits) or new ones (with few commits) or both? Hint: git log or git shortlog
7. Develop an automated script to answer the above questions (e.g., using GitPython and relevant libraries as demonstrated in the tutorial in Working Session 1). **Explain how you implement it in the report and attach a link to your code in your GitHub repository** (make sure it is public). In case you have already implemented this part in exercise 2, you can reuse it here, but please still include the link to your script.
8. **In the report, give your detailed analysis about the similarities and differences of each fixing commit with its corresponding VCC(s) for each of your 3 vulnerabilities** in terms of what you found in Tasks 4 and 5.
9. **Put the results in the Excel format** we have given to you besides this file. **Do not forget to change file name to your name and your student ID**. For some of the fields, you may need to compute the statistics (average or sum).
10. Write a report to summarize your findings for the above tasks.

The report of this exercise should be in A4-size page with Times New Roman or similar font size 12. The first page of the report should include your full name and student ID along with the list of three vulnerabilities we have assigned to you. **Name the file with your name and student ID.**

**Tips about how I would go about doing this exercise:** Before doing this Exercise, I will first check whether the links of repositories and the fixing commits of the 3 assigned vulnerabilities are valid. I will also have a close look at the tutorial about Git given by Triet in Working Session 1, and also recall the knowledge I have gained about Git when doing Exercises 1 and 2. If I did not install Git, I would do it. Then, I will start to clone/download all repositories to my computer. After that, I will move on to study each of the Git command mentioned in the Hints of tasks 3 and 5 to find out which parameter it accepts and which one suits my need. I will need to repeat the same process for every (up to the 5 most relevant) affected file of each fixing commit of the 3 vulnerabilities. For task 3, I will need to consider deleted (-) and added (+) lines separately. For deleted lines, I will use git blame I have studied earlier to find the latest commit that modified such lines. For added lines, I will find the closest enclosing scope as described as well as use any other code snippet that can lead to such addition. I will then utilize git blame to find the latest commits of all those relevant lines. For step 5, I will learn about the listed parameters to see and explain how they are different from the default settings (no parameters) of git blame. After that, I will run each of them to see whether they produce different VCCs for me. In case they give me different VCCs, and I will explain which one I think give me the most satisfactory results. I will also look at steps 4 and 7 to see how I can use some programming libraries (e.g., GitPython in Python) to automate the use of Git commands. Then, I will describe and explain in details each question for each vulnerability in the report. I will also need to compare my findings of VCCs with the ones I obtained from exercise 2 of fixing commits to analyze their similarities and differences. I also fill in the Excel file the information I found. Finally, I will submit the 2 files to Canvas.

**Please note that answer without explanation would not receive any point.**

***How to Submit:*** *The exercise solution will be submitted via Canvas as there is an upload facility created for this exercise on Canvas.*

This exercise is designed to help you to achieve the learning outcomes # 5 and 6 in the course outline.