Course Name: Secure Software Engineering
Course Code: COMP SCI 4412/7412
Assessment Component: Assignment 2 (15%) – Individual Assessment
Release Date: 30/08/2023
Due Date: 13/09/2023 by 11:55pm.
Submission: MyUni

The list of tasks for assignment 2

Part 1 (5 points):
Please visit the Assignment 2 Part 1 link to register your repos on MyUni and input your identified vulnerable source code files **as soon as possible** after you find them. **The student who submits earlier will claim the authorship of the source code file and the later ones must choose a different file to work on.**

1. Study about ***SQL Injection, OS Command Injection and CRLF Injection vulnerabilities*** on Common Weakness Enumeration and related websites. You do not have to submit this part.
2. Identify 3 source code files in open-source GitHub repositories. Each type of vulnerability must have at least one source code file. The projects must satisfy the following conditions:
   - The programming languages must be either ***Java, JavaScript, or PHP***
   - The repository has more 100 stars and 10 contributors on GitHub
3. Include the following artifacts about each file you have found in the report:
   - Link to the file
   - Link to the commit that fixes the vulnerable file
   - Name of the file
   - The programming language used in the file
   - Name of the repository
   - Number of repository stars
   - Number of contributors in the repositories
   - Type of vulnerability (CWE)
4. Pinpoint the code lines within the source code files you have identified that contain the vulnerabilities you found.
5. Also enter the information you have found in tasks 3 and 4 into the Assignment 2 Part 1 link along with your name and student ID to avoid duplicate submission.
6. Explain how the vulnerable lines correlate to the definition or causes of the vulnerability you have studied
7. Show how to fix the vulnerability and explain in details. It is not mandatory that the fix has to be executable, but the explanation must be reasonable. If there is already a fix available, explain how this fix complies with the standard mitigation techniques for the vulnerability.
8. Write your findings in the report.
9. Please visit the Assignment 2 Part 1 link to input your identified vulnerable source code files **as soon as possible** after you find them. You can do the analyses and put your findings in the report later (but still before the deadline). The student

who submits earlier will claim the authorship of the source code file and the later ones must choose a different file to work on. In case you accidentally select the same source code file, there will be a red flag to notify you.

Part 2 (10 points):

1. Clone the GitHub repositories associated with your vulnerabilities (See the Assignment 2 Part 2 link on MyUni) 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 assignment 2 part 2 is to identify the Vulnerability-Contributing Commits (VCCs) from fixing commits of the vulnerability you have found in assignment 1 part 2 and analyze these commits. **If the fixing commit provided was different from the one you found, then you should specify which one you choose and why.**
   Changing the assigned repository is not allowed. If you would like to change to another repository, please contact your teaching assistants explaining your situation.
3. To identify VCC, for each affected (deleted, added) line of each file in the fixing commit of the vulnerability, perform the following steps and **write down what you have done for this task (e.g., The complete commands you use and how you extract useful information from returned results of the commands) in the report and screenshot**. If a commit has more than 1 affected files, please identify 1 or 2 files that you think are the most relevant ones for **fixing** the current vulnerability only.
   a. If a line was deleted, identify the latest commit that modified that line and record that commit. Hint: git blame
   b. 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 detail. 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
   c. Then, select the most frequently identified commit as the VCC for the current fixing commit. If there is more than one such commit with the same frequency, select all of them.
4. 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 gives you the most satisfactory VCC(s). Write in the report your findings.**
5. Analyze your identified Vulnerability-Contributing Commit (VCC) and answer the following questions. Perform the following steps and **write down what you have done for this task (e.g., The complete commands you use and how you extract useful information from returned results of the commands) in the report and screenshot**. If a commit has more than 1 affected files, please

identify 1 or 2 files that you think are the most relevant ones for **introducing** the current vulnerability only. The Git command that you can consider for answering each question can be found in Hint.

a. 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
b. How many total files were affected in the current VCC? Hint: git diff or git show
c. 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
d. How many total lines of code (**including** comments and blank lines) were deleted? Hint: git diff or git show
e. How many total lines of code (**including** comments and blank lines) were added? Hint: git diff or git show
f. How many total lines of code (**excluding** comments and blank lines) were deleted? Hint: git diff or git show
g. How many total lines of code (**excluding** comments and blank lines) were added? Hint: git diff or git show
h. How many days were between the current VCC and the previous commit of each affected file? Hint: git log
i. 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
j. Which developers have modified each affected file since its creation? Hint: git log
k. 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

6. After you gathered your VCC and fixing commit of the vulnerability, you continue to analyze the events happening between them by **answering the following questions with explanation for each pair**:
   a. Is the current VCC an initial commit? Hint: git show
   b. Is the developer of the current fixing commit and its corresponding VCC the same? If not, give your reflections on the difference in their experience. Hint: git show
   c. How many days were between the current fixing commit and its corresponding VCC? Is the current VCC fixed immediately (within the same day)? Hint: git log
   d. If the current VCC is not fixed immediately (within the same day), was there any other vulnerability, bug or special request happening between the current fixing commit and VCC that might require higher

priority? **Include your detailed reflections on why it had remained unfixed**. Hint: git log with string matching or regular expression.

7. **Put the results in an Excel sheet**. **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).
8. 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 vulnerability we have assigned to you. **Name the file with your name and student ID.**

Tips about how I would go about doing this exercise:

Part 1:

I will first study about the vulnerabilities mentioned in the task on Common Weakness Enumeration website. Google is also always worth a try if I want to explore more. Then, I will try to use the name of the vulnerability and search it on GitHub. After I find the repositories, I will filter them using the above criteria. Then, I will focus on the vulnerable files and analyze them line-by-line or use existing tool. If there is already a fix for that vulnerability, I will include it in my report. Otherwise, I will try to see how I can fix it using the mitigation techniques I have learned for the vulnerability. I will explain how my findings match with the materials I have learned for that vulnerability. Most importantly, I will try to input my identified source code files as soon as I find them to avoid duplication and having to search again.

Part 2:

Before doing this Assignment, I will first check whether the links of repositories and the fixing commits of the assigned vulnerabilities are valid. I will also have a close look at the tutorial about Git in Working Session 1, and also recall the knowledge I have gained about Git when doing Assignment 1. If I did not install Git, I would do it. Then, I will start to clone/download the repository to my computer. After that, I will move on to study each of the Git command mentioned in the Hints of task 3 and task 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 2 most relevant) affected file of fixing commit of the vulnerability. 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 task 4, 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. For task 6, I will focus on the fixing commit and its VCC(s) using git log and git show to answer each question. For point d of task 6, I will try to look for the important keywords/phrases that I have observed that they usually go together with a vulnerability/bug or special request to identify the special events that might have higher priority and resulted in a delay in the fix.Then, I will describe and explain each task in the report. 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 MyUni as there is an upload facility created for this exercise on MyUni.*

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