Exercise 2
Secure Software Engineering - 2019 Semester 2

Chris Crouch - a1015970
==

Task 2

The 3 fixing commits that I am analyzing are:
https://github.com/apache/camel
    235036d2396ae45b6809b72a1983dee33b5ba32
https://github.com/jenkinsci/junit-plugin
    15f39fc49d9f25bca872badb48e708a8bb815ea7
https://github.com/FasterXML/jackson-databind
    7487cf7eb14be2f65a1eb108e8629c07ef45e0a
==

Task 3 - camel
a. What was the message and title of the fixing commit? Was there any mention of fixing a bug or vulnerability?
CAMEL-10567: Camel-Jackson: Add an option to allow the UnmarshallType header use
This reference the JIRA ticket raised for the vulnerability.

b. How many total files were affected in the fixing commit?
3

c. How many total directories were affected in the fixing commit?
2

d. How many total lines of code (including comments and blank lines) were deleted?
73

e. How many total lines of code (including comments and blank lines) were added?
1

f. How many total lines of code (excluding comments and blank lines) were deleted?
39

g. How many total lines of code (excluding comments and blank lines) were added?
1

h. How many days were between the current fixing commit and the previous commit of each affected file?
components/camel-jackson/src/main/java/org/apache/camel/component/jackson/JacksonDataFormat.java: 417.37 days
components/camel-jackson/src/test/java/org/apache/camel/component/jackson/JacksonMarshalUnmarshalTypeHeaderNotAllowedTest.java: N/A new file
components/camel-jackson/src/test/java/org/apache/camel/component/jackson/JacksonMarshalUnmarshalTypeHeaderTest.java: 870.508 days

i. How many times has each affected file of the current fixing commit been modified in the past since their creation (including rename of the file)?
components/camel-jackson/src/main/java/org/apache/camel/component/jackson/JacksonDataFormat.java: 21
components/camel-jackson/src/test/java/org/apache/camel/component/jackson/JacksonMarshalUnmarshalTypeHeaderNotAllowedTest.java: 17
components/camel-jackson/src/test/java/org/apache/camel/component/jackson/JacksonMarshalUnmarshalTypeHeaderTest.java: 17

j. Which developers have modified each affected file since its creation?
components/camel-jackson/src/main/java/org/apache/camel/component/jackson/JacksonDataFormat.java
  Andrea Cosentino
  Claus Ibsen
  Daniel Kulp
  Richard Kettelerij
  Willem Jiang
components/camel-jackson/src/test/java/org/apache/camel/component/jackson/JacksonMarshalUnmarshalTypeHeaderNotAllowedTest.java
  Andrea Cosentino
components/camel-jackson/src/test/java/org/apache/camel/component/jackson/JacksonMarshalUnmarshalTypeHeaderTest.java
  Andrea Cosentino
  Claus Ibsen
  Willem Jiang

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?
Andrea Cosentino : 445
Claus Ibsen : 10380
Daniel Kulp : 252
Richard Kettelerij : 30

Willem Jiang : 1250

This list includes the two most frequent developers (note that Willem Jiang has used at least 3 different accounts), two fairly frequent developers, and even the least appearing is in the top 50.

==

Task 3 - junit-plugin

a. What was the message and title of the fixing commit? Was there any mention of fixing a bug or vulnerability?
[SECURITY-521] Avoid XXE when parsing test results
This directly references a security advisory, from which the vulnerability can be traced.

b. How many total files were affected in the fixing commit?
5

c. How many total directories were affected in the fixing commit?
3

d. How many total lines of code (including comments and blank lines) were deleted?
125

e. How many total lines of code (including comments and blank lines) were added?
1

f. How many total lines of code (excluding comments and blank lines) were deleted?
96

g. How many total lines of code (excluding comments and blank lines) were added?
1

h. How many days were between the current fixing commit and the previous commit of each affected file?

src/main/java/hudson/tasks/junit/SuiteResult.java:	78.8846 days
src/main/java/hudson/tasks/junit/XMLEntityResolver.java:	322.746 days
src/test/java/hudson/tasks/junit/JUnitResultArchiverTest.java:	106.123 days
src/test/resources/hudson/tasks/junit/testXxe-oob.xml:	N/A (new file)
src/test/resources/hudson/tasks/junit/testXxe-xxe.xml:	N/A (new file)

i. How many times has each affected file of the current fixing commit been modified in the past since their creation (including rename of the file)?

src/main/java/hudson/tasks/junit/SuiteResult.java:	63
src/main/java/hudson/tasks/junit/XMLEntityResolver.java:	14
src/test/java/hudson/tasks/junit/JUnitResultArchiverTest.java:	29
src/test/resources/hudson/tasks/junit/testXxe-oob.xml:	1
src/test/resources/hudson/tasks/junit/testXxe-xxe.xml:	1

j. Which developers have modified each affected file since its creation?

src/main/java/hudson/tasks/junit/SuiteResult.java:
  Andres Rodriguez
  Andrew Bayer
  David Gageot
  Haindrich Zoltán (kirk)
  Jared Arnold
  Jaromir Hamala
  Jesse Glick
  Mirko Friedenhagen
  Oliver Gondža
  Wadeck Follonier
  Zoltan Haindrich
src/main/java/hudson/tasks/junit/XMLEntityResolver.java:
  Jaromir Hamala
  Jesse Glick
  Wadeck Follonier
src/test/java/hudson/tasks/junit/JUnitResultArchiverTest.java:
  Andres Rodriguez
  Andrew Bayer
  Evaristo Gutiérrez
  Jesse Glick
  Oliver Gondža
  Per Böhlin
  Wadeck Follonier
src/test/resources/hudson/tasks/junit/testXxe-oob.xml:
  Wadeck Follonier
src/test/resources/hudson/tasks/junit/testXxe-xxe.xml:
  Wadeck Follonier

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?

Andres Rodriguez : 7

Andrew Bayer : 80

David Gageot : 1
Evaristo Gutiérrez : 1
Haindrich Zoltán (kirk) : 1
Jared Arnold : 1
Jaromir Hamala : 1
Jesse Glick : 127
Mirko Friedenhagen : 1
Oliver Gondža : 38
Per Böhlin : 2
Wadeck Follonier : 1
Zoltan Haindrich : 1

This is a mixture of 3 of the top 5 most common authors, along with a lot of inexperienced developers.
==

Task 3 - jackson-databind
a. What was the message and title of the fixing commit? Was there any mention of fixing a bug or vulnerability?
Preliminary blocks for #2052: not actually sure if there is vuln via gadgets, but they seem suspicious enough to block tentatively
This references a GitHub issue, which details the vulnerability.

b. How many total files were affected in the fixing commit?
1

c. How many total directories were affected in the fixing commit?
1

d. How many total lines of code (including comments and blank lines) were deleted?
6

e. How many total lines of code (including comments and blank lines) were added?
0

f. How many total lines of code (excluding comments and blank lines) were deleted?
3

g. How many total lines of code (excluding comments and blank lines) were added?
0

h. How many days were between the current fixing commit and the previous commit of each affected file?
src/main/java/com/fasterxml/jackson/databind/jsontype/impl/SubTypeValidator.java: 21.1563 days

i. How many times has each affected file of the current fixing commit been modified in the past since their creation (including rename of the file)?
src/main/java/com/fasterxml/jackson/databind/jsontype/impl/SubTypeValidator.java: 8

j. Which developers have modified each affected file since its creation?
src/main/java/com/fasterxml/jackson/databind/jsontype/impl/SubTypeValidator.java
  Casasola Marcos
  Tatu Saloranta

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?
Casasola Marcos : 1
Tatu Saloranta : 2735

Tatu Saloranta has used at least 6 different names that make up the 6 most common contributors; the next most common has made 28 commits. So this is almost a one-person project. The other contributor, as can be seen, only made this single contribution.

==

Task 4

I actually performed task 3 with a Bash script that can be found here:
https://github.com/a1015970/SSE_Exercise2/blob/master/analyze_repository.sh
This script is not particularly robust however, so for this task I have also automated it in Python.

These Python scripts can be found in https://github.com/a1015970/SSE_Exercise2
The file https://github.com/a1015970/SSE_Exercise2/blob/master/analyze_git_commit.py has a function analyze_git_commit(repo_path, fixing_commit) that performs the analysis on a given repository at a given commit.
The file https://github.com/a1015970/SSE_Exercise2/blob/master/script_main.py runs this function on my assigned vulnerabilities. To run it, simply execute "python script_main.py"

I have developed and tested these running under Windows 10. The code should be portable to other platforms, but I have not tested it.

The Python script is very similar to the Bash script. It calls repo.git to directly call the Git binary, so it can use all the same options. Where filtering is required it uses regular expressions, similar to the Bash script.
It should be noted that it uses simple regular expressions to remove comments, and that these are not particularly exhaustive (indeed automatically stripping comments from any file written in any language would be an enormous undertaking). These filters were designed to strip out the comments that I found in the diffs for my assigned commits, and nothing else.

A few details on how each step does its job:

A. git log -1 --pretty=%B - gets just the commit message from the first log entry
B. git diff --name-only HEAD HEAD^ - lists files modified
C. git diff --dirstat HEAD HEAD^ - lists directories modified
D. git diff HEAD HEAD^ - shows diff. This can then be filtered using regex
E. as D.
F. as D.
G. as D.
H. for each file as in B., “git log -2 --pretty=%ct $file” gives the modification time in seconds since the epoch for the last two commits.
I. for each file as in B., “git log --follow --pretty=oneline $file” has a single line entry for each commit, following renames.
J. for each file as in B., “git log --pretty=" %aN" $file” lists the authors, which then needs to be reduced to the unique list.
K. create the author list as in J., then search the list of “git log --pretty=%aN” and count the occurrences of each author.

=======================================
Notes on Excel Spreadsheet:

This spreadsheet had several columns not explained fully in the exercise description, and I had to make some choices about how to fill it out.

Average number of days between fixing commit and previous commit - I chose to treat new files as having 0 days between commits rather than treating them as missing values.

Average time of each affected file being modified in the past - new files - 1 modification.

Average number of developers who have modified the affected files - I treated this as average per file.

Maximum number of commits made by an involving developer - I did exact string matching on names, not trying to match alternate forms of names.