ASSIGNMENT-II

SECURE SOFTWARE ENGINEERING

VULNERABILITIES

1. SQL Injection (CWE-89) Slims9_bulian
2. CRLF Injection (CWE-93) Scm-Manager
3. OS Command Injection (CWE-78) OpenWB
4. Cross-Site Request Forgery (CWE-352) Jenkins

Submitted by,

Kishaiyan Vellaichamy Thangaraj,

A1819309

PART-1

Vulnerability-I

SQL Injection

Link to the file-
https://github.com/slims/slims9_bulian/admin/modules/reporting/customs/staff_act.php

Link to the commit-
https://github.com/slims/slims9_bulian/commit/761159a38b93e9cc803d94968cae2aa756393860

Name of the file- staff_act.php

The programming language used is PHP.

Name of the repository- slims9_bulian

Number of repository stars-145

Number of contributors in the repositories-14

Type of Vulnerability CWE-89

The vulnerability in the file staff_act.php is SQL injection. The input value start date and until date was not checking for any SQL queries that could be given as the input and if it were not rectified could have led to data breach with SQL queries. escape_string() method in PHP is used to sanitize the user input. It ensures that the special characters are properly escaped before being incorporated into a SQL query. Lines 122 and 126 of the file were accepting the input from the user without sanitizing it. The fix for this vulnerability is the use of escape_string() method with an instance of dbs which is an instance for database connection that provides various method to interact with the database. escape_string() method in PHP is a standardized mitigation technique for SQL Injection.

Vulnerability-II

CRLF Injection

Link to the file-https://github.com/scm-manager/scm-manager/blob/develop/scm-core/src/main/java/sonia/scm/util/HttpUtil.java

Link to the commit-https://github.com/scm-manager/scm-manager/commit/500a082a3fa49f7ffa9811f8a46449275c09f9d4

Name of the file- HttpUtil.java

The programming language used is java.

Name of the repository- scm-manager

Number of repository stars-117

Number of contributors in the repositories-38

Type of Vulnerability CWE-93

The vulnerability in the file HttpUtil.java is CRLF Injection. The software did not check or sanitize the user input for any CRLF character that could be a security vulnerability. The code does not check for the special character in the user input. The fix for this vulnerability should be proper sanitation of the user input which is the standard mitigation technique for this vulnerability. But the Fix in the given commit is implementing a separate function with string as parameters and that checks for any CRLF characters using CharMatcher method from google package. Although, it is not the standard mitigation technique they have fixed it using CharMatcher.

Vulnerability-III

OS Command Injection

Link to the file- https://github.com/snaptec/openWB/blob/master/web/settings/setPassword.php

Link to the commit- https://github.com/snaptec/openWB/commit/dfc47c5c7c8b18d4ecc580a8687ec546856410e1

Name of the file- setPassword.php

The programming language used is PHP.

Name of the repository- openWB

Number of repository stars-315

Number of contributors in the repositories-90

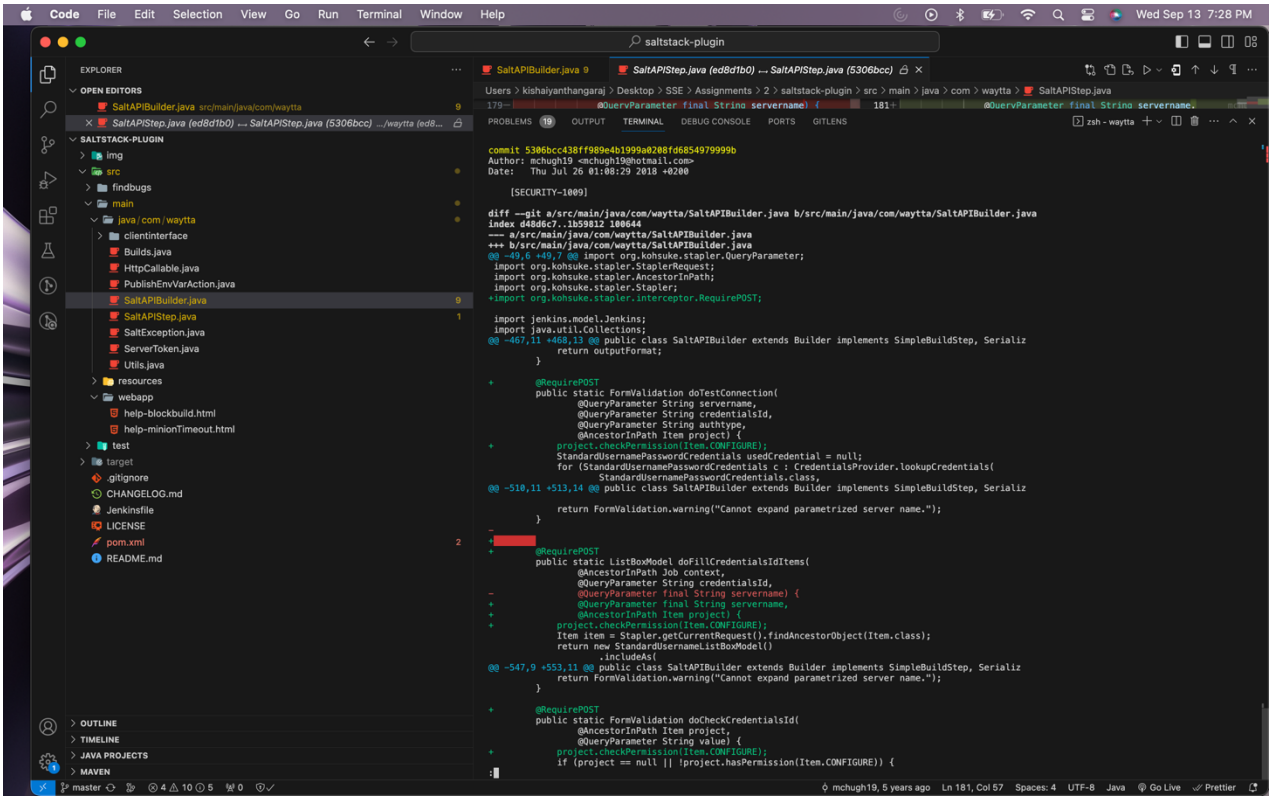
Type of Vulnerability CWE-78

The Vulnerability in this file setPassword.php is OS Command Injection. The software did not properly escape the username thus making it vulnerable to command injection. The escapeshellarg() is a PHP function for escaping a string that makes it safe to be used within a shell command. Line 63 in the file did not properly escape the username parameter which could lead to Command Injection vulnerability. The proper use of escapeshellarg() to escape the string username

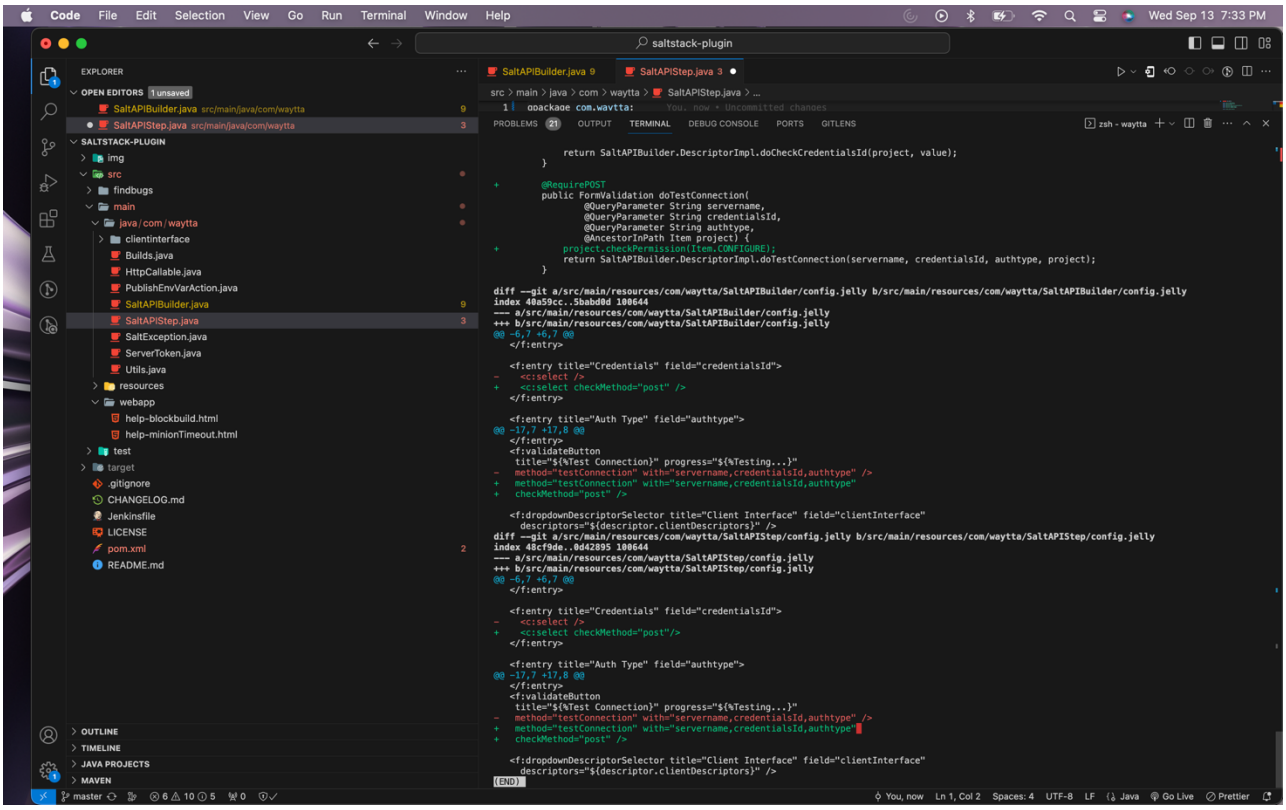
to sanitize the string fixes the vulnerability. The escapeshellarg() is a standard mitigation technique for Command Injection vulnerability.

PART-2

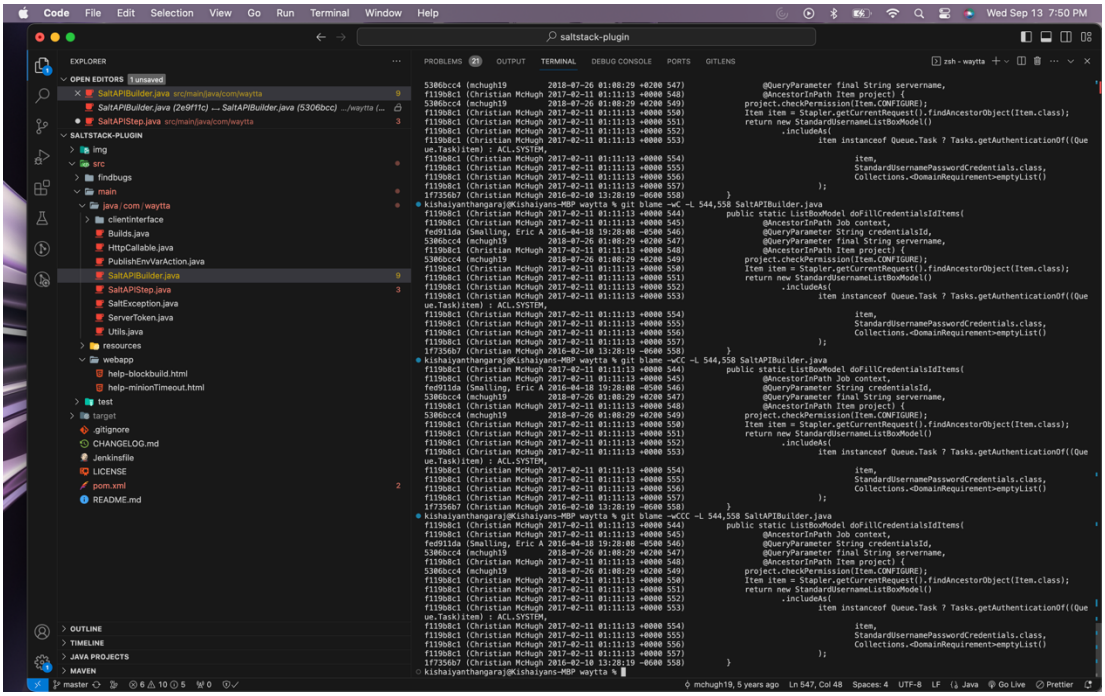
The deleted lines from the code on the fixing commit from the Vulnerability contributing commit (VCC) can be seen using *git log –p* command to show all the commits and the deleted lines for each commit. To see for the commit given to me, I used git show <commit-hash> command to show the exact lines deleted in a file.



The added lines in the fixing commit are inside the class and there is no other relevant code that led to the current line.



There are no changes in VCCs while using git blame with different parameters such as –w, -wM ,-wC , -wCC and –wCCC



5.

a. The Title and the message of VCC were Creds3(#54) and convert to credentials 2.1 api Allow for folder-based credentials. There was not a mention of a vulnerability or a bug fix.
b. There are 5 files affected by the current VCC.
c. There was only one affected directory src/main/java/com/waytta
d. There were 82 deletions including comments and blank spaces.
e. There were 92 additions including comments and blank spaces.
f. There were 79 deletions excluding comments and blank spaces.
g. There were 87 additions excluding comments and blank spaces.
h. There was a gap of 10 days (about 1 and a half weeks) between the VCC and the previous commit.
i. File 1: 89 modifications
j. File 2: 1 modification
k. There were totally 8 developers modified each affected files from the initial commit
l. From observing the log, the developers submitting the commits were a mix of experienced and new developers.

6.

a. The VCC is not the initial commit.
b. From the observation, the author of VCC and the fixing commit are the same.
c. The difference between VCC and its corresponding fix is 530 days (about 1 and a half years) in total.
d. The current VCC was not fixed the same day. Although, there were a lot of bugs fixed between the VCC and the Fixing commit, the reason for the long gap between the VCC and the fixing commit is that the vulnerability was not diagnosed anytime soon and therefore it took long time to fix the problem.