Software Security Project Part 4: Architectural Design Principles, Usable Security Principles, Protection Poker, and Bug Fixes

Name	Unity ID
Abhash Jain	ajain28
Arjun Sharma	asharm33
Sachin Kumar	skumar26
Aishwarya Sundararajan	asundar2

1. Architectural Design Principles

i) **The following test case violates the principle of "Separation of Privilege"** which states that: "Software should not grant access to a resource, or take a security-relevant action, based on a single condition."

Test Case ID	Test_1
Test Name	**Separation of super user privilege to multiple users**
Steps for test case	i)Go to Admin Home Page using Admin credential. (URL: http://152.46.16.55:8082/openmrs-standalone/referenceapplication/home.page) : Username: admin, PW: Admin123 ii) Click on System Administration on the home page and then click on "Manage Accounts". iii) On the next page Select "Add new Account". Add a user Family Name: "Jain" Given Name: "Anuj",Male. And Select Add User account checkbox. Enter Username "januj" and password as "Ncsu1234", privilege as "full". Uncheck force password change. In capability only select "Administer System" and save the detail. And new user is created. iv) Logout from this user and login with new user credential. UN: "januj" and PW: "Ncsu1234" InPatient Ward. v) On Home Page select "System Administrator" and then Manage Account module on next page. vi) Click on "Add New Account". Add a user Family Name: "Jain" Given Name: "Sanskar",Male. And Select Add User account checkbox. Enter Username "sjain" and password as "Ncsu1234", privilege as "full". Uncheck force password change. In capability only select "Has Super User Privileges" and save the details. And new user is created. vii) Logout from this user account. viii) Login to new user account with new sjain's credential. You are able to successfully able to create a new super user.
Expected Result	New User "sjain" should have not given the Super user privilege at the moment. There should be a pipeline mechanism until all user in that pipeline don't approve user should not be given very elevated privilege like "Super user"
Test Case Status	Failed
Mitigation to be taken by OpenMRS	Various users of the application should be authorized to approve new user request such as super user. Until all of them don't Approve the request the new user creation request will be in

	Pending state. Once they approve the request new user will be Created. This is how we can separate the sensitive operation to Take place with multi user intervention.

ii) The following test case violates the principle of "Complete Mediation" that states:
Software should validate every access to object to ensure that access is allowed

Test Case ID	Test_2
Test Name	**Elevation of privilege of standard user**
Steps for test case	i)Go to Admin Home Page using Admin credential. (URL:http://152.46.16.55:8082/openmrs-standalone/login.htm) ii) Click on System Administration on the home page and then click on"Advanced Administration". iii) On the next page Select "Manage Users" Anchor link. iv) From the Role drop-down menu select "Organizational: Nurse" as roles. It will list Nurse user Select the hyperlink under System-ID 4-2.that brings to next page. v) Change the username and password from the input item User password's and confirm it in the next password box the same Password. I have selected the password "Ncsu1234". And select the save user button to save the information. vi) Now login as nurse,where on the dashboard you don't have access to System administration option. vii) Now open manage users link which is available to admin user, http://152.46.16.55:8082/openmrs-standalone/admin/users/users.list?name=admin&role=&action=Search. viii) Now type admin in find user or name field, in search result admin records will appear, click on admin link appearing in first column of the record of the search result. viiii) Now in the page that appear, we can change the admin's password and other details,so let's change the password from Admin123 to Admin1234 and save it ix) Now logout and login with username as admin and password as Admin123, it will fail and now login with password Admin1234. x) So here , a standard user with not having access to manage the users did changed the admin account details.
Expected Result	User 'nurse' not having access to admin pages of manage users, shouldn't have been able to access it and ability to also modify the data over there

Test Case Status	Failed
Mitigation to be taken by OpenMRS	Various level of users of the application should be authorized to access different parts of the application after authentication.

iii) The following test case violates the principle of "Fail Safe defaults" that states:
The initial state should be to deny access unless access is explicitly required. Then, unless software is given explicit access to an object, it should be denied access to that object and the protection state of the system should remain unchanged.

By default, weak passwords should not be allowed

Test Case ID	**Test_3**
Test Name	**Weak password acceptance in Register Patient**
Steps for test case	i) Go to http://152.46.16.55:8082/openmrs-standalone/index.htm, there i was logged in as admin ,then click on admin dropdown top bar of that page,then Myaccount button will appear,so click on it. ii) Next click on Change password button. iii) Now in the page that appears, provide the old password, and provide a weak new password such as **'Test1234'**, and then save it. iv) Now logout and login again with new password
Expected Result	The application should not have allowed to set a weak password while changing the password, as it makes system vulnerable to dictionary and brute-force attacks.
Test Case Status	Failed
Mitigation to be taken by OpenMRS team	The password rules for authentication should be strong and not easily open to dictionary and brute-force attacks. A combination of upper and lower case characters with a mix of alpha-numeric characters that don't usually form a sequence is a good password rule.

2. Usable Security Principles:

2.1 Appropriate Boundaries:

OpenMRS violates Appropriate boundaries principle for User Interaction Design in Secure Systems. As per the principle, the interface should distinguish objects and actions along boundaries that relate to important issues such as “need to know” or “least privilege”. In OpenMRS doctors can register a patient given that he is aware of the register patient request link.

Test Case ID	USP1
Description	Low privilege boundary between doctor and admin.
Steps:	1) Login to OpenMRS using admin credentials and navigate to “Register Patient” section and copy the url from browser which is http://localhost:8081/openmrs-standalone/registrationapp/registerPatient.page?appId=referenceapplication.registrationapp.registerPatient 2) Now log out from admin and login using doctor credentials. 3) You will find only three options on the home menu viz “Find Patient Record”, “Active Visits”, “Appointment Scheduling”. 4) Paste the link copied from “Register Patient” section and enter. 5) You’ll be redirected to “Register Patient” section even though you’re logged in as doctor.
Expected Results	Doctor should not able to access “Register Patient” section/module.
Actual Results	Doctor is able to access “Register Patient” section/module.
Result	FAIL

2.2 Explicit Authorization:

OpenMRS violates Explicit Authorization principle for User Interaction Design in Secure Systems. Explicit Authorization means the user’s authorities must only be provided to other actors as a result of an explicit action that is understood by the user to imply granting. As mentioned in the above test scenario the doctor is able to access the admin console by just manually inserting the link in the browser. Then the doctor is able to create a new user and provide full privileges. During this process the user should have been authorized while providing full privileges to the newly created user.

Test Case ID	USP2

Description	Doctor able to create user by accessing admin console.
Steps:	1) Login to OpenMRS using Doctor credentials and visit the below link by entering it in browser search. (http://localhost:8081/openmrs-standalone/admin/users/users.list) 2) Click on Add user and create a new user by entering user information and selecting Roles as “Privilege Level Full” and save. 3) Log out from doctor’s account and login as the created user. 4) Try login using newly created user’s credentials.
Expected Results	Doctor shouldn’t have the access to create new users and the newly created user shouldn’t be allowed to login.
Actual Results	The newly created user was able to login.
Result	FAIL

2.3 Revocability + Expected Ability:

OpenMRS violates Revocability and Expected Ability principle for User Interaction Design in Secure Systems. Revocability principle states the admin interface should allow the user to easily revoke authorities that the user has granted and Expected Ability principle states the interface must not generate the impression that it is possible to do something that cannot actually be done. If a user is already logged in the system and admin downgraded or changed the user privileges/roles, it doesn’t reflect in the current session and the user is under wrong impression that it’s still possible to perform activities.

Test Case ID	USP3
Description	Doctor able to perform activities in current session even after privileges being revoked.
Steps:	1) Login to OpenMRS via browser 1 say Firefox using system admin credentials and login to OpenMRS via browser 2 say Google Chrome using nurse credentials. 2) In browser 1 (for admin) navigate to System Administration → Advanced Administration → Manage Users → Search for nurse “Jane Smith” and change nurse’s role from Organizational: Nurse to Organizational: Doctor. 3) In browser 2 (for nurse) refresh the page, the nurse is able to navigate to “Capture Vitals” module and search.
Expected Results	Nurse user should not be able to view/access “Capture Vitals” module after the her role has been changed.
Actual Results	Nurse user is able to view/access “Capture Vitals” module.
Result	FAIL

3. Protection Poker

FUNCTIONAL REQUIREMENTS:

1) **Addition of Payment records with Patient's Past Visits :** Patient payment records can be maintained as a separate task managed by admin and integrated with the patient past visits
2) **Maintaining Patient immunization history :** Patient immunization history could be maintained along with other patient details
3) **Saving Patient Insurance records :** Patient health insurance records should be saved in the database and should be used along with the appointments functionality as the same insurance is likely to be used for a long period of time.
4) **Addition of emergency ward location :** Emergency ward could be integrated as another location in the OpenMRS database
5) **Maintaining Patient referrals records :** New patient referrals records could be maintained

Table 1 : Database tables value points

Database table	Value points	Usage in Requirement No.
Visit	Sachin: 13 Abhash: 13 Arjun: 8 Aishwarya: 13 **Team:** 13	1,2,3,5
Patient	Sachin: 20 Abhash: 8 Arjun: 8 Aishwarya: **Team:** 20	1,3,5
Visit_type	Sachin: 8 Abhash: 8 Arjun: 8 Aishwarya: 5 **Team:** 8	2
Visit_attribute	Sachin: 8 Abhash: 8 Arjun: 8 Aishwarya: 8 **Team:** 8	2
Visit_attribute_type	Sachin: 8 Abhash: 5	2

	Arjun: 8 Aishwarya: 5 **Team:** 8	
form	Sachin: 13 Abhash: 13 Arjun: 8 Aishwarya: 13 **Team:** 13	2,3
form_field	Sachin: 13 Abhash: 13 Arjun: 8 Aishwarya: 8 **Team:** 13	2,3
field_answer	Sachin: 13 Abhash: 13 Arjun: 13 Aishwarya: 13 **Team:** 13	2,3
location	Sachin: 5 Abhash: 5 Arjun: 5 Aishwarya: 5 **Team:** 5	4
location_tag	Sachin: 5 Abhash: 5 Arjun: 3 Aishwarya: 3 **Team:** 5	4
location_attribute	Sachin: 5 Abhash: 5 Arjun: 3 Aishwarya: 5 **Team:** 5	4
Patient_identifier	Sachin: 13 Abhash: 13 Arjun: 8 Aishwarya: 5 **Team:** 13	5

Table 2: Database tables used by requirements

Requirement	**Table used**	**Value points of Table**	**Max value point**

1	Visit Patient	13 20	20
2	Visit Visit_type Visit_attribute Visit_attribute_type form form_field field_answer	13 8 8 8 13 13 13	13
3	Visit Patient form form_field field_answer	13 20 13 13 13	20
4	location location_tag location_attribute	5 5 5	5
5	Visit Patient Patient_identifier	13 20 13	20

Table 3 : Security risk
(Security risk = Ease points x Value points)

Requirement	Ease of attack points	Value points	Security risk	Rank
1	Sachin: 8 Abhash: 8 Arjun: 8 Aishwarya: 5 **Team:** 8	20	160	2
2	Sachin: 3 Abhash: 3 Arjun: 3 Aishwarya: 3 **Team:** 3	13	39	5
3	Sachin: 5 Abhash: 5 Arjun: 8 Aishwarya: 8 **Team:** 5	20	100	3

4	Sachin: 13 Abhash: 13 Arjun: 8 Aishwarya: 13 **Team:** 13	5	65	4
5	Sachin: 13 Abhash: 13 Arjun: 13 Aishwarya: 8 **Team:** 13	20	260	1

Evidence and commentary on rankings:

With protection poker, in order to assign some value points we did found out database tables for corresponding requirements , since the number of tables and their strategic importance will in turn determine their value and ease points, and as a corollary will impact overall security risk and rankings.

Also for some of the features there had been many tables with interdependent relationships which does make complexity of those relationships a major factor to assign value or ease points.

Now since security risk was dependent on value points and ease points, which in turn depends on number of tables and their associated data composition and relational complexity, it also determined the complexity of feature.
Feature 5 is more complex relative to rest of the features, due to its higher value points attributing to important tables being used for it and also relatively higher ease points with those tables being relatively easy to target. Feature 1 is relatively simpler as compared to feature 5 attributing to lesser ease points with tables comparatively harder to access a compared to feature 5.Similarly, requirement 3 although having same value points does have lesser ease points thus less security risk as compared to requirement 5, because of the tables used in the requirement are harder to access because of its complex interrelationships.Moreover for requirement 4, although there is higher ease points, there is less value to these records comparatively, which overall makes its security risk lower than requirement 3. Lastly, requirement 2 has relatively lower ease points and value points than requirement 4, which did make it least risky requirement.

META Plan :

Requirement No.	Applicable META	Remarks
1.	M	Payment records are extremely sensitive records and its disclosure can result in malicious usage resulting in monetary loss to the patients and also hospitals if patient sues them for the monetary compensation for it.

		This risk can be mitigated with stronger encryption and hashing of sensitive payment data stored in database.Also these records , while presented for view on UI can be hidden to a extent to not reveal all the PII(Personally Identifiable Information).
2.	M	Immunization records can be encrypted in the database. Also ,strong access control lists can be maintained for roles based access to the tables containing immunization records data. Moreover, for the encryption algorithm, we will use a standard and secure algorithm like RSA or AES-256 , instead of reinventing the wheel with our own encryption algorithm and in turn compromising the system security.
3.	M	Patient insurance records are very sensitive piece of information, and its revelation will result in non-compliance of HIPAA regulations ,which in turn compromises the organisation's integrity and patient details. To mitigate this strong encryption algorithm and Access Control Lists can be maintained to limit the access of insurance information availability to the number of application roles and users, thus limiting the risk of disclosure and security failure.
4.	M	Locations impact patients visit data indirectly, as if patient visits are incorrectly linked to a non-desired location, then it will impact the integrity and consistency of patient medical records. This risk can be mitigated by limiting this privilege to admin access user, so that limited number of users have access to it and thus minimise the risk involved in it.
5.	M	Patient referral records are important in context of patient medical history ,often adhering to HIPAA regulations,its disclosure can result in invasion of privacy. These records can be made more secure with sensitive data encrypted in database and limiting the access to these records to a specific set of users,thus limiting the risk with enforcement of stronger access control lists.

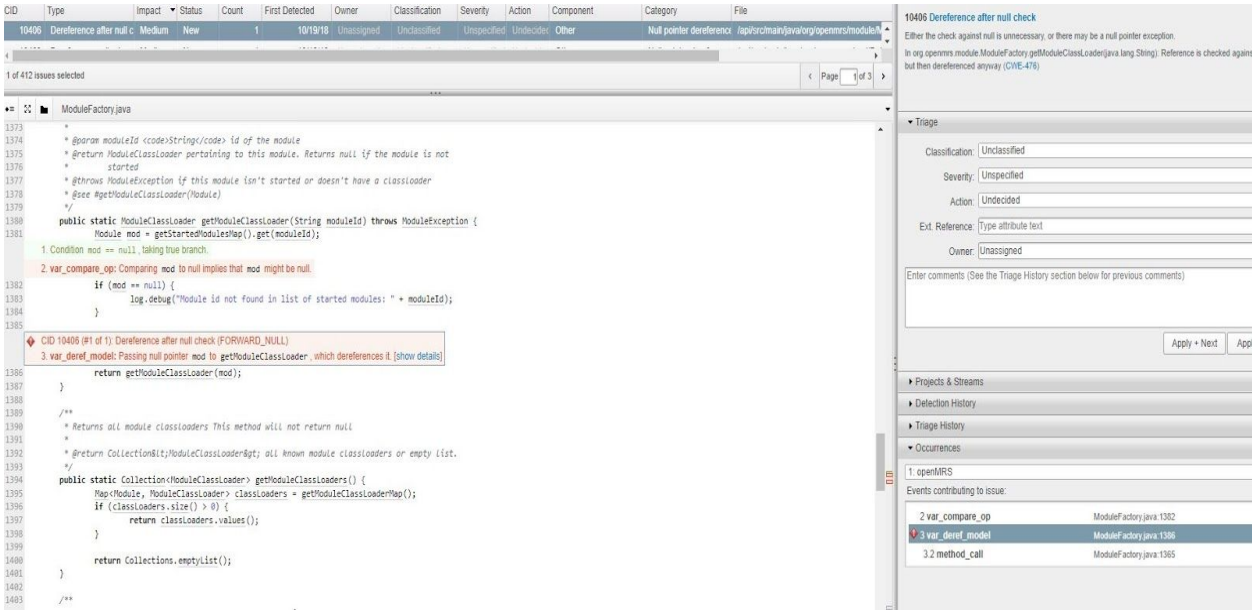
4. Bug Fixes:

For Project Part 4, here are the bug fixes we had developed:

i) ID : Bug_fix_1

Test Case : Dereference after null check (Check against null is unnecessary or there may be NULL pointer exception)

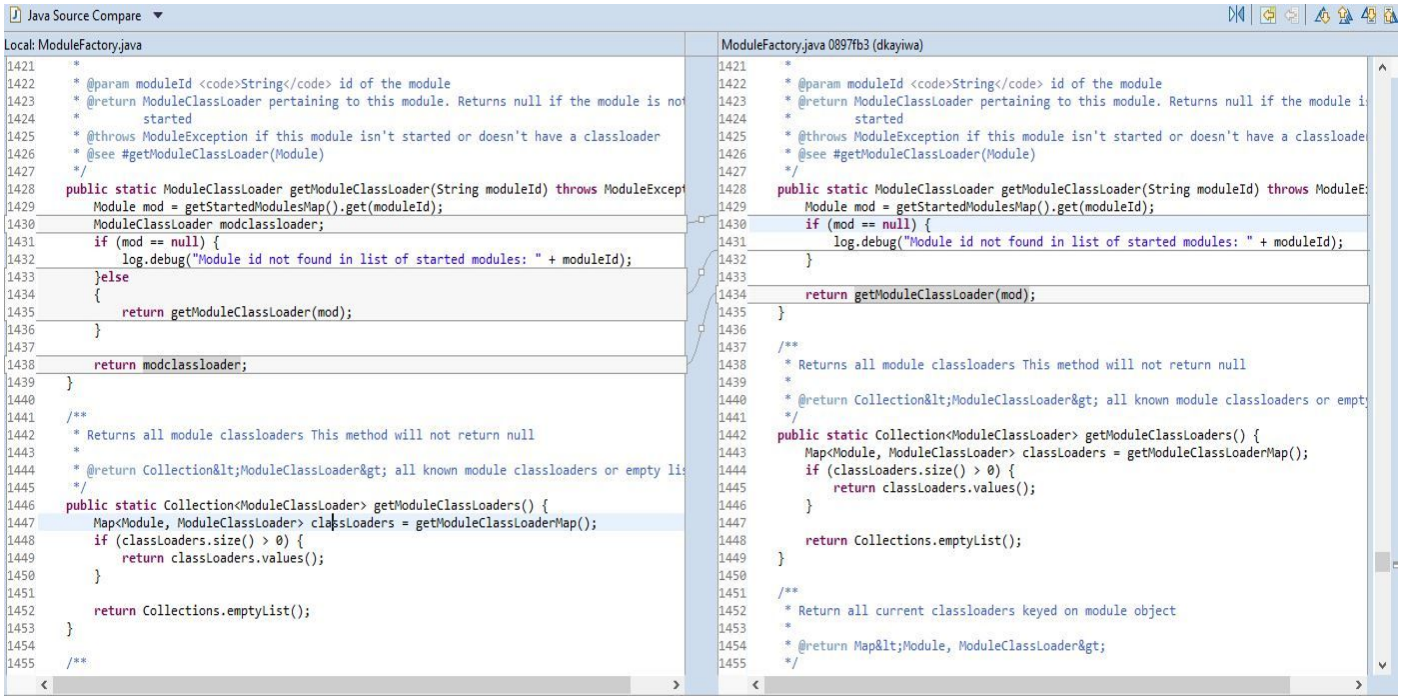
Reference : Test case 8 , in Report 3
CID: 10406 in Coverity report



Tool used : Coverity

Diff of code :

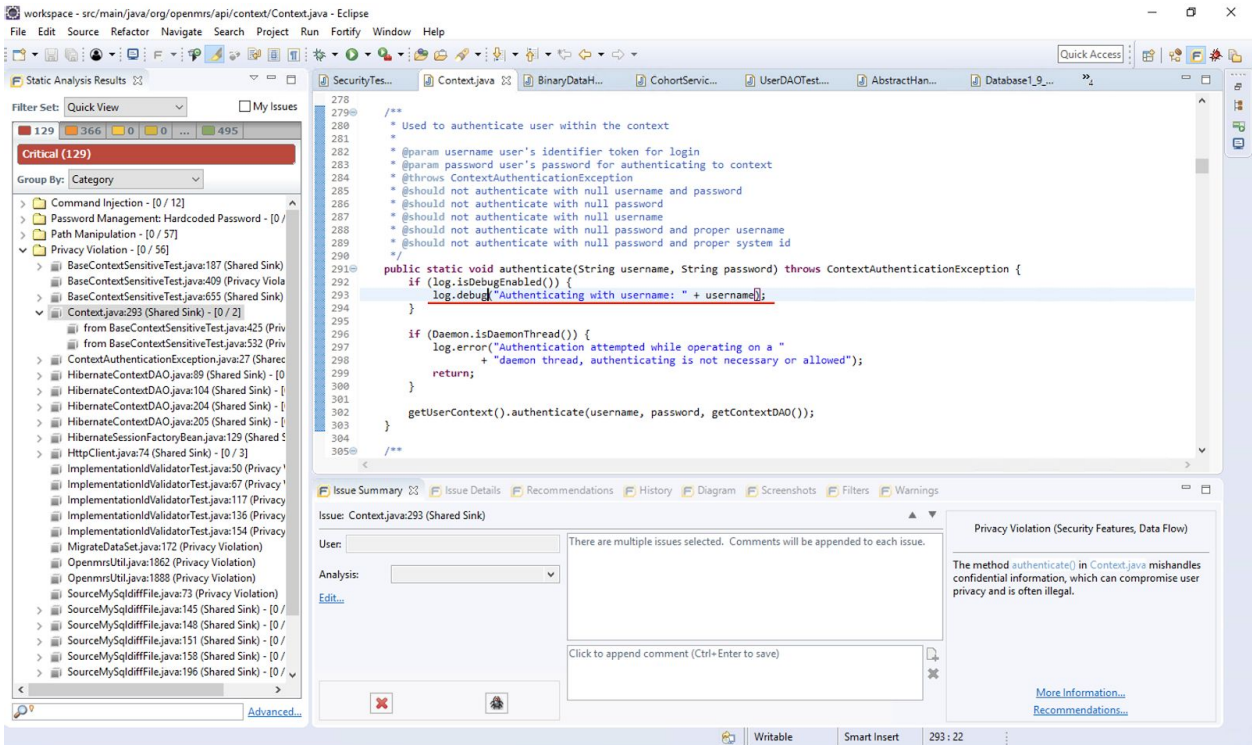
Fixed code: **Buggy Code:**



ii) ID : Bug_fix_2

Test Case : Privacy violation: (This method mishandles the confidential information which can compromise the user privacy)

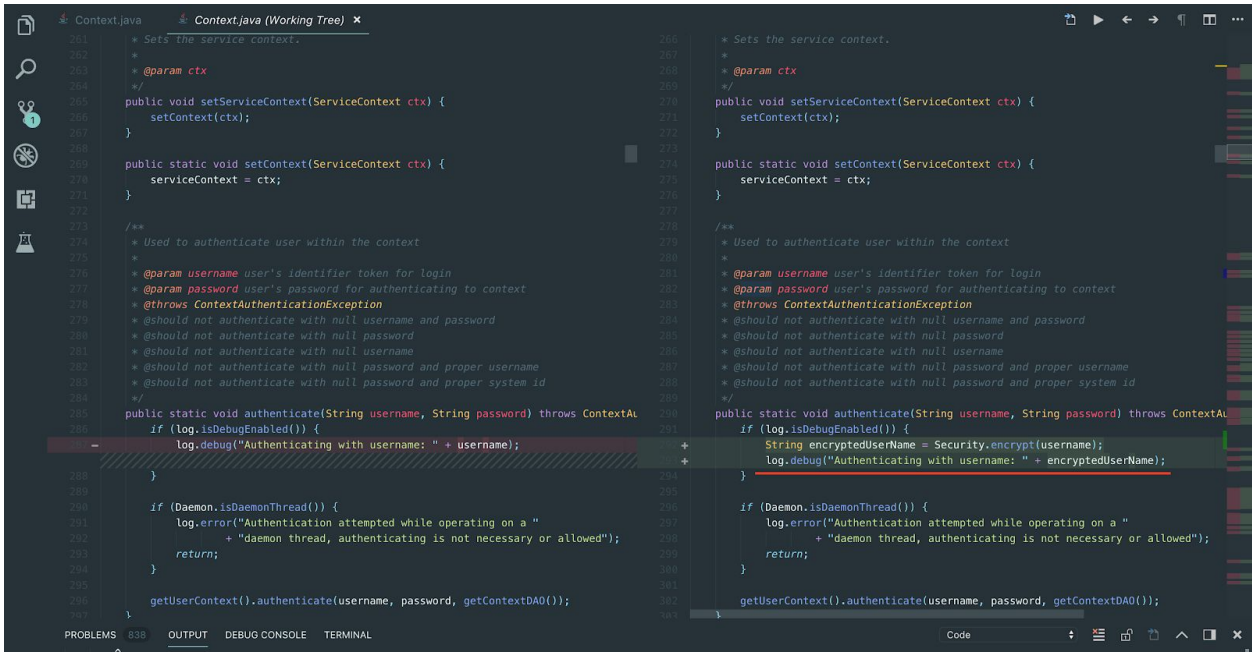
Reference : Privacy Violation in Context.java as found in Fortify report



Tool used : Fortify

Diff of code :

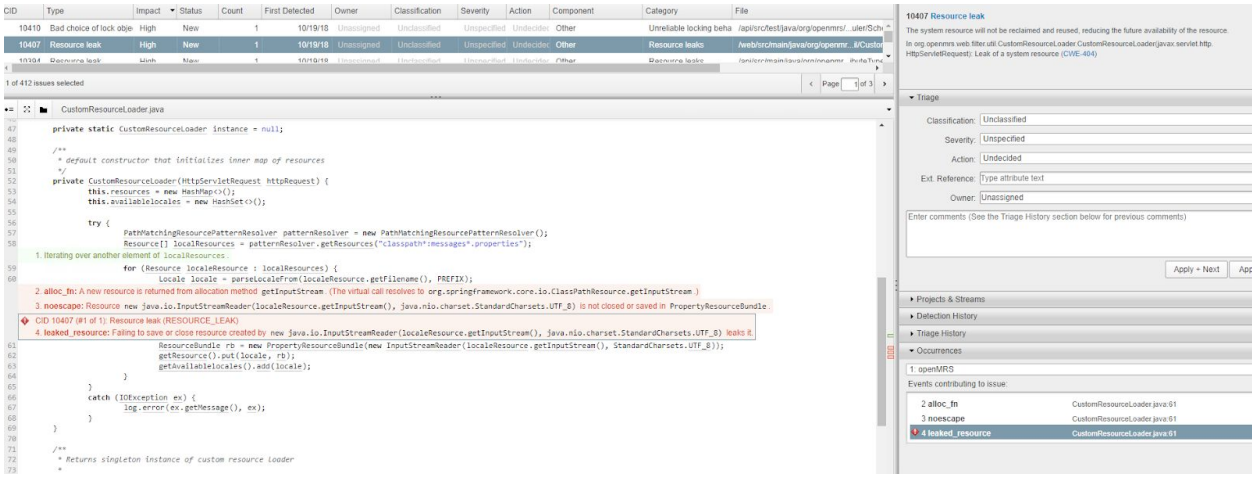
Buggy Code: **Fixed code:**



iii) ID : Bug_fix_3

Test Case : Resource leak (System resource is not reclaimed and reused reducing the future availability of the resource)

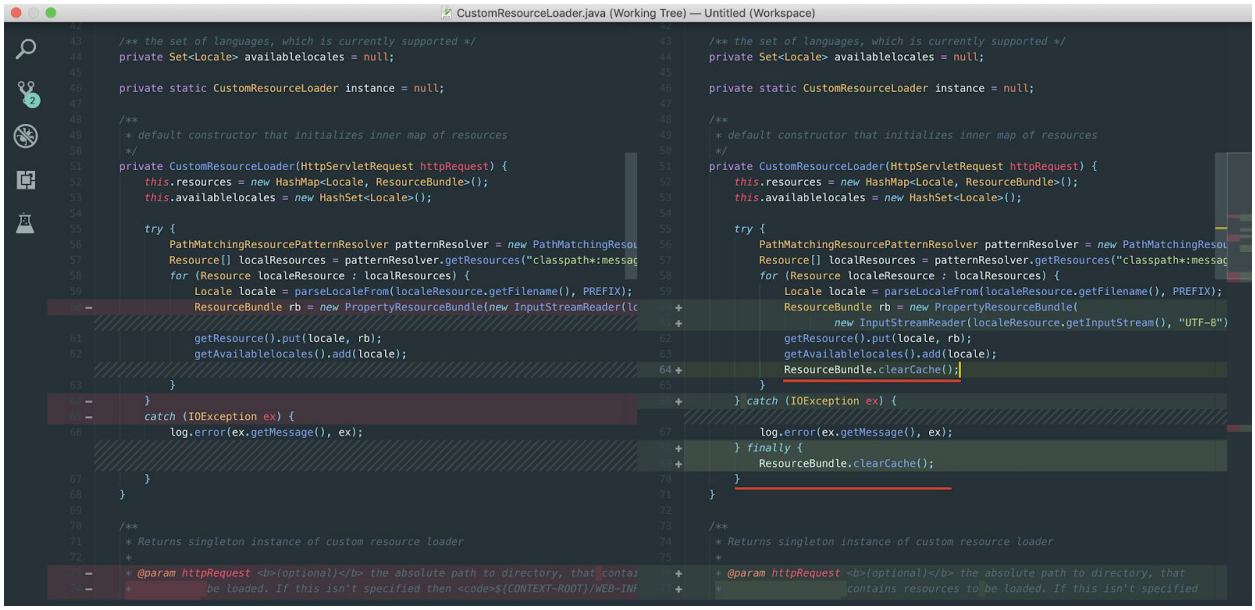
Reference : Test case 2 , in Coverity Bugs sections of Report 3
CID: 10407 in Coverity report



Tool used : Coverity

Diff of code :

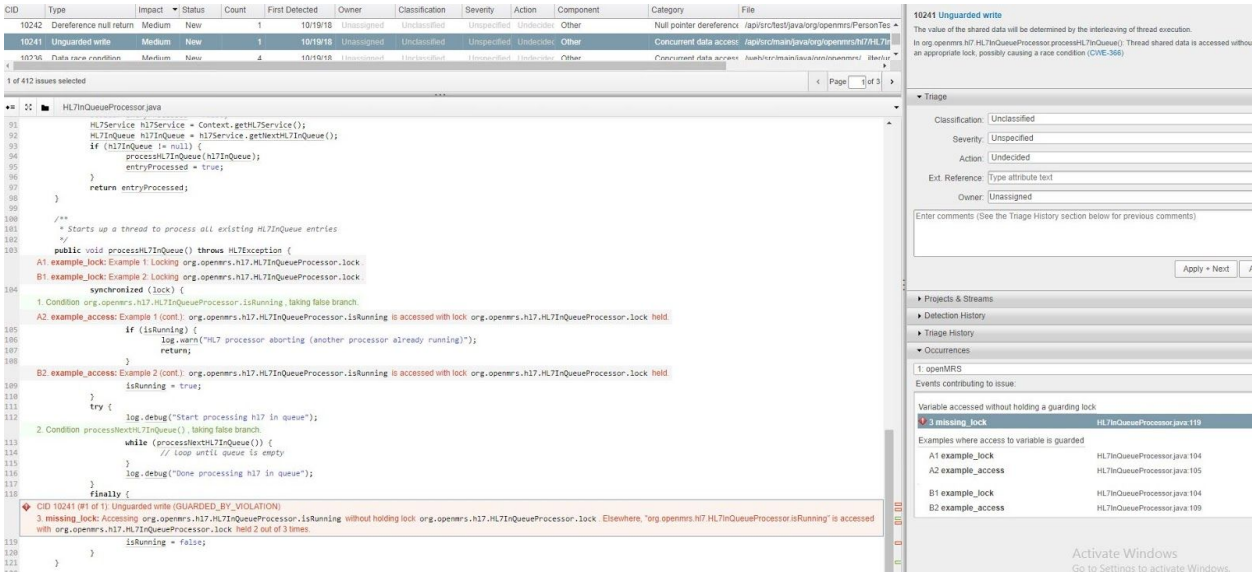
Buggy Code: **Fixed code:**



iv) ID : Bug_fix_4

Test Case : Unguarded write (Value of the shared data is determined by the interleaving of the thread execution, which can cause a Race condition which might lead to a deadlock state.)

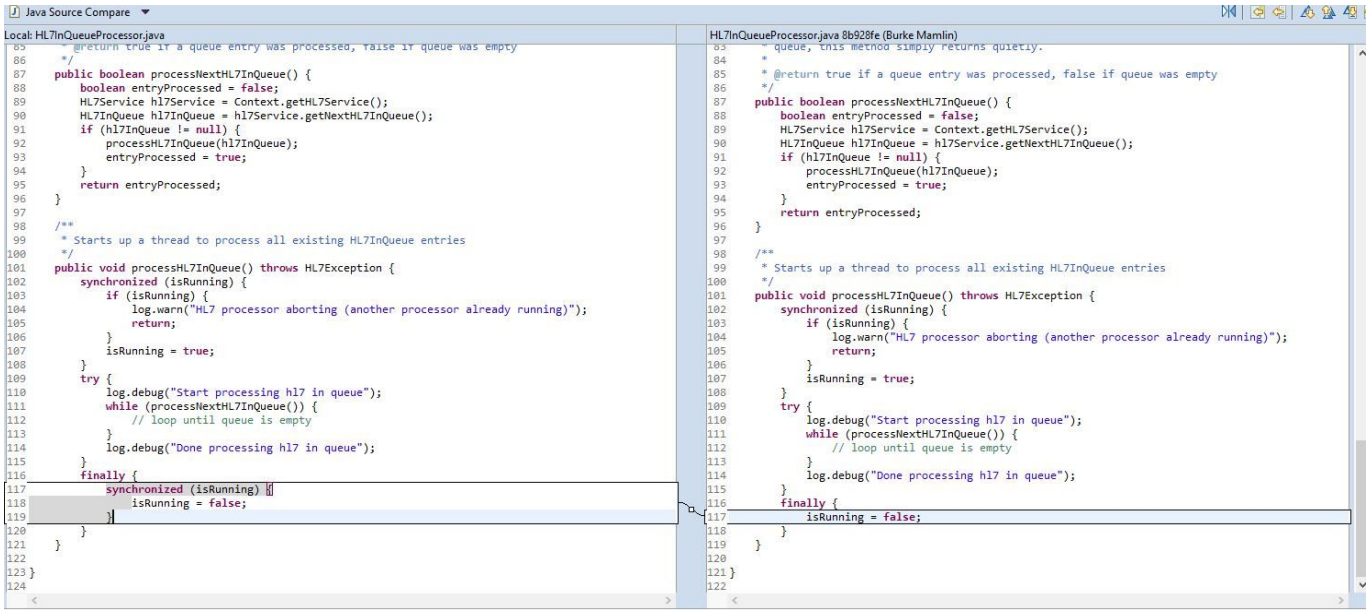
Reference : CID: 10241 in Coverity report



Tool used : Coverity

Diff of code :

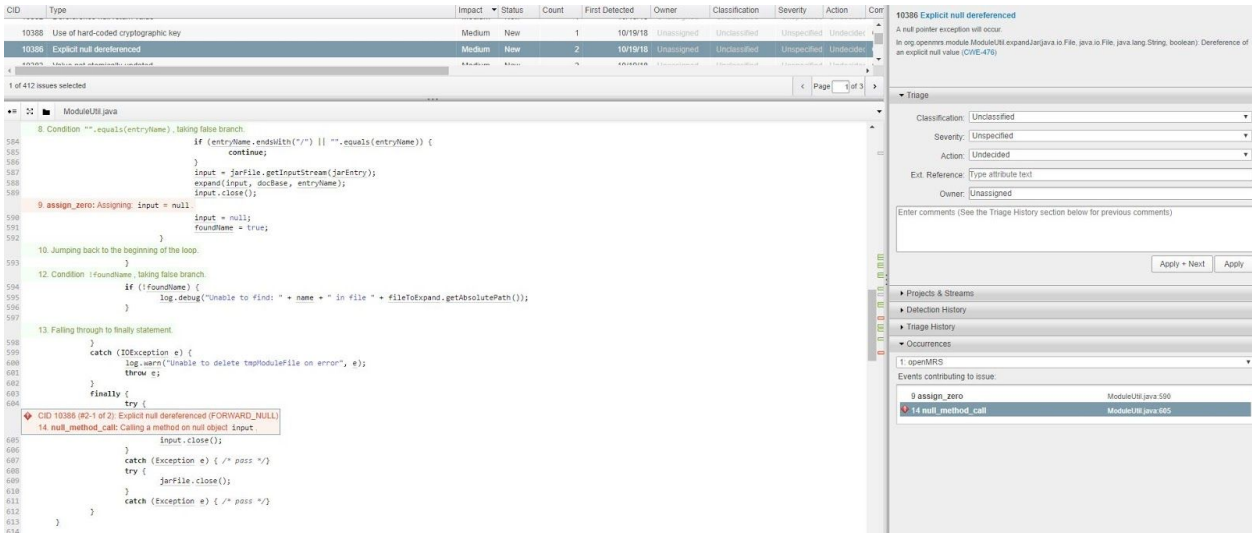
Fixed code: **Buggy Code:**



v) ID : Bug_fix_5

Test Case : Explicit null dereferenced (A null pointer exception will occur due to rerefereing of an explicit null value)

Reference : CID: 10386 in Coverity report



Tool used : Coverity

Diff of code :

Fixed code: **Buggy Code:**