Secure Programming for Application Development:

Library Management System

Security Audit Report

for

Prof. Irina Tal

by

Rohan Bhangale (x18147119)

Contents

Abstract

In this era of digital revolution, there has been a massive rise in demand for software applications, while emphasis is given to creating softwares which are user friendly and have catchy visuals rather than security for the software application developed. These developed software applications lack the required security measures and are exposed to numerous threats and vulnerabilities. It is of critical importance that these factors should be addressed during the development and testing of the application. In this study, various programming paradigms and security testing models/methodologies are discussed. Furthermore, a Java based application is audited for security.

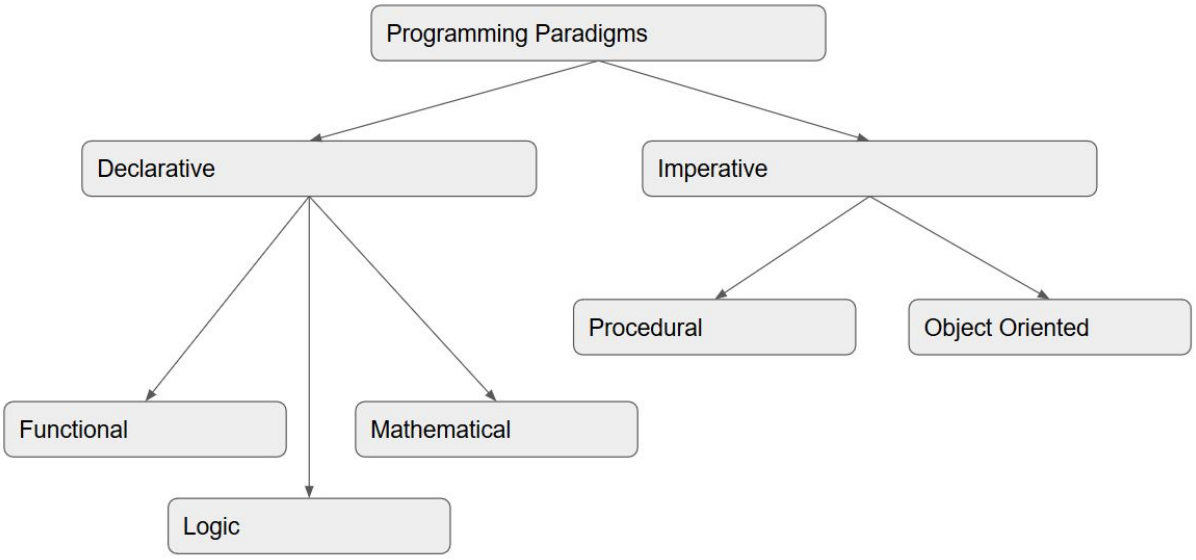
Introduction

It is of critical importance to secure applications in order to protect data and sensitive information from the attackers. Any vulnerability present in the application can be exploited by an attacker via various attack vectors such as social engineering, IP addresses and Access Point etc. Application should be tested for such threats and attacks while security should be integrated in SDLC. Intent of security testing is to find loopholes primarily covering areas such as CIA triad, Authorization, Authentication and Non-repudiation.[**1**] In-depth penetration testing should be carried out along with risk and code analysis.Use of databases(CVE, OWASP, OSINT) with commonly known information security vulnerabilities and exposures should be made use of. Analysis of the application should be performed in a non-runtime environment and while the program is in operation for studying the behavior of the application and monitoring its effect on the system resources[**2**].Various security testing tools such as SonarQube, SQLMap, ZAP, OllyDbg and Burp Suite [**3**]can be used for identifying vulnerabilities like CSRF, XSS, SQL Injection, BufferOverflow. Apart from using automated tools for performing static or dynamic code analysis, manual code analysis gives a perspective beyond the automated analysis in terms of design and architecture flaws.
For illustrating, a vulnerable application based on Java is audited for security using various above mentioned techniques.

Programming Paradigms and Security

As the literal meaning of *Paradigm*, a pattern or model while *Programming Paradigm* refers to style of programming. Programming Paradigm is an approach in which a programming language solves a problem[4]. The concept of Programming Paradigm was formally proposed by Robert W. Floyd. Programming Paradigms differ in

concepts, methods(control flow, events, objects, variables, etc.) and each paradigm's abstraction level. The paradigm shift can be observed by historically looking at programing languages, starting with Machine Language were the programming was done in binary using opcode ie use of the same instruction set as used by the computer which was then followed by assembly language which uses ergonomics for making it easier for instructors with acronyms such as ADD,MOV etc thereafter for the successor languages(PASCAL, FORTRAN, BASIC, C) which make use of functions like for loops, if statements. It can be observed that for the further languages more level of abstraction is seen as lesser interaction with physical hardware is involved. As these high level languages focus on the problem and how to solve it. Languages like SQL which have a higher level of abstraction as very high level of language is used where instructions such as SELECT X from Y. Machine Learning demonstrates the latest programing paradigm were it can be observed that getting computers to do tasks for which humans can not possibly express the instructions. Common programming paradigms can be categorised into Imperative and Declarative. Imperative specifies the procedure which gives the computer instructions on how to do. Declarative specifies the computer instructions on how to do.



The best performing programming languages in industry and academia are Java, Kotlin and C++. It is observed that use of Java has grown popular as it is one of the top programming languages with a larger developer community and market share. The reason being platform independent(runs on Windows,Linux and Mac) and the ability to write once and run anywhere with the use of JVM(Java Virtual Machine) the program is converted into a class file(bytecode), which is interpreted by JVM

anywhere[5]. Java is a multi-paradigm programming language[6] There exist four programming paradigms in Java, namely,

Imperative

Based on the model of computation used in Turing Machine.Imperative programming paradigm changes the state of program with the use of statements. It holds commands for the computer to execute. It works on describing how to find the solution.
Languages: Assembler, C, Fortran
[7]

Procedural

This programming paradigm focus on procedures, also considered as routines, which consist of subroutines. Routines are built of a series of computational steps to be executed. During execution, call can be given to any procedure at any point by the procedure itself or other procedures. It has the ability of reusing the code.
Languages: C, C++, ColdFusion

Object Oriented

The OO programming paradigm focuses on classes and objects which communicate while the execution of program. The basic unit is called object. Data is given importance over the procedure. It follows bottom-up approach for designing and are reusable.Interaction among the objects achieved with the help of functions. Some major benefits include Abstraction, Flexibility and Inheritance.

Object Oriented Programming has four classes,

Abstraction

In OO programming paradigm, abstraction deals with hiding the implementation from the user, while the user is only made available with functionality. Users only have information on what the object does instead of how it does. Use of Abstract classes and interface is made to achieve abstraction.
[8]

Encapsulation

One of the important OO concepts out of the four, encapsulation is the mechanism of bundling of data and code together to function as methods, as a single unit. It enables data hiding, that is variable inside one class are hidden from other classes and only accessible by current class methods. Class is declared as private for variables to achieve encapsulation, while public methods are implemented to carry out modification and viewing of the variable.

Polymorphism

Another OO concept which enables co working of objects by defining methods which take other objects as parameter. More cooperation and efficiency is achieved by keeping the objects under one super class. Polymorphism is the ability to take many forms. Java offers polymorphism at two stages at run time and compile time. Use of reference variable is made in order to access an object.

Inheritance

The OO concept that enables objects to work together. It defines the relations between the various classes . In Java all classes are descendent of java.lang.Object and implement its methods. Observing the class hierarchy, all subclasses have inherited default API libraries and it adds its own set of fields, methods while also inheriting from the superclasses. The properties from the parent class are carried in the further sub class.
[8]

Declarative

Declarative programming paradigm is more abstract and opposite to Imperative, it focuses on what the program is supposed to do rather than how the computer is supposed to do it.The sequence of execution of the code is not crucial. Declarative is divided in three types of programming paradigms further. Declarative programming paradigm emphasis on the logic of the computation rather than control flow. [9]

Functional

Functional Programming Paradigm has its foundational roots in mathematics and is independent of languages. Key focus of this paradigm is the execution of mathematical functions. Implementations of the function are hidden while data and functions are loosely coupled. Functions are replaceable with their values without changing program meaning.
[10]
Runtime Application Self-Protection Technology (RASP) is a newer approach in security when it comes to Java Application, it monitors, detects and mitigates against real time security threats against the application. It works as an plug-in to JVM. Applications are among the top attack vectors, applications with vulnerabilities end up in production because of reasons such as deployment deadlines often trump security and vulnerable built of applications are deployed, vulnerabilities are introduced in updates of application, use of 3rd party as that can't be mitigated. RASP contributes to heightened level of security, it has knowledge of logic, event and data flow of the program.[11][12]

Security Testing

Before stressing on security testing, it is equally important to create code which is clean and does not redundant and unused functions,as these functions can be later be exploited by a malicious user. It is also troublesome when the code is reviewed or performed with dynamic analysis for security testing. In order to avoid loss or impact and for finding loopholes and weaknesses in the system, security testing is conducted to find the vulnerabilities, threats and risks involved with the application. Objective of security testing is to identify the vulnerabilities which got developed during the application development process, so that these flaws can be addressed and fixed by the developers.

Secure Development Lifecycle

With time and experience it is of critical importance to have security be tested during the Software Development Lifecycle, implementing Secure Development Lifecycle addresses the issue. SDL is used alongside the traditional Software Development Lifecycle with security introduced at every stage of the development lifecycle.

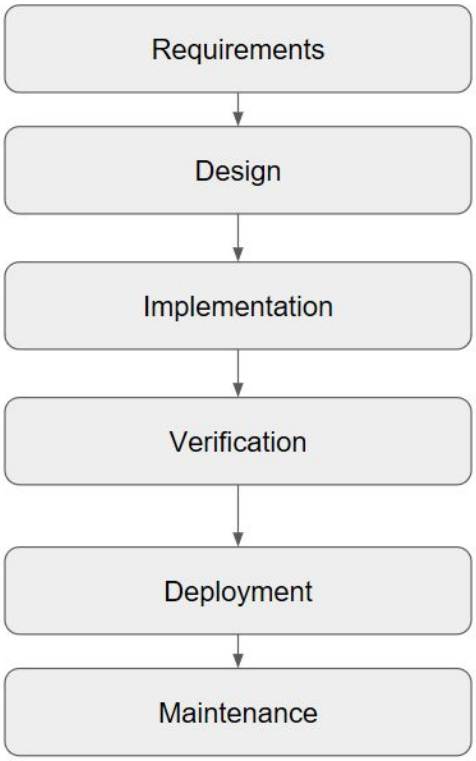


Figure: Secure Development Lifecycle

SDL comprises of 6 steps namely,

Requirements

Phase involves gathering information on the application ie its clients, user experiences. Analysis of all the security issues to be addressed(CIA Triad). Documentation of various system interfaces and inconsistencies and feasibility. It is the most important step as being planned and prepared serves as the most secure manner to tackle cyber incident with the application. Development of security requirements and risk analysis ie identifying data assets and the threats revolving them with the help of threat modelling and ranking the threats and vulnerabilities based on their severity

Design

At this phase design selections such as which software architecture, software components, programming languages and Interfaces are to be applied and used in the development and are documented. For placing security, multiple layers of defenses are implemented ie Defense in Depth.
Use of Secure Design Principles such as
[13]

Principle of Least Privilege
– Compartmentalization
– Isolation
Defense-in-Depth
Secure the Weakest Link
Fail-Safe
Secure by Default
Keep it Simple
Avoid Security by Obscurity
Reluctance to Trust
Privacy
Use Proven Technologies

Figure: Secure Design Principles

Implementation

Post Coding phase, involves implementing secure coding standards, builds and configuration and making use of good programming practices and identify and avoiding known vulnerabilities and awareness of malicious logic. Design and implementation of security phases.

Verification

Involves testing of the developed code, conducting code reviews and documenting all the findings. Running the application against various security tests.
[14]

Deployment

Once the application is released in the production environment, it is important that time to time upgrades and security patches should be released for zero day exploits and new vulnerabilities learnt.

Maintenance

Customer feedback should be addressed and security incidents and reports should be studied and analysed for incidents in order to fix and mitigate them and be ready for such future incidents. While upgrades should be aired keeping up with the technology and mitigations towards the newly discovered attacks.

Application Testing (Library Management System)

Well-Known Approaches

Static Code Testing

In Static Code Testing, the code is manually tested along with requirement and design documents for finding the errors. Objective of static testing is to find errors at the initial stages of development lifecycle

Dynamic Code Analysis

In Dynamic Code Testing, the testing is done during the execution of code. Here the behavior of the application is monitored for its impact on memory, cpu and performance of the system. The main purpose is to test the application for its

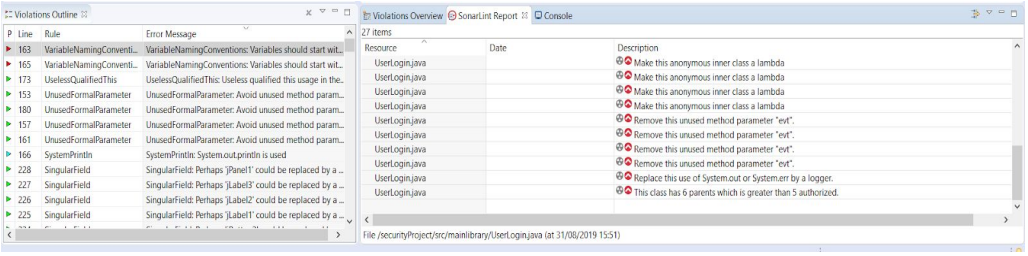
readiness for production. It gives the bugs and bottleneck observed in the application.

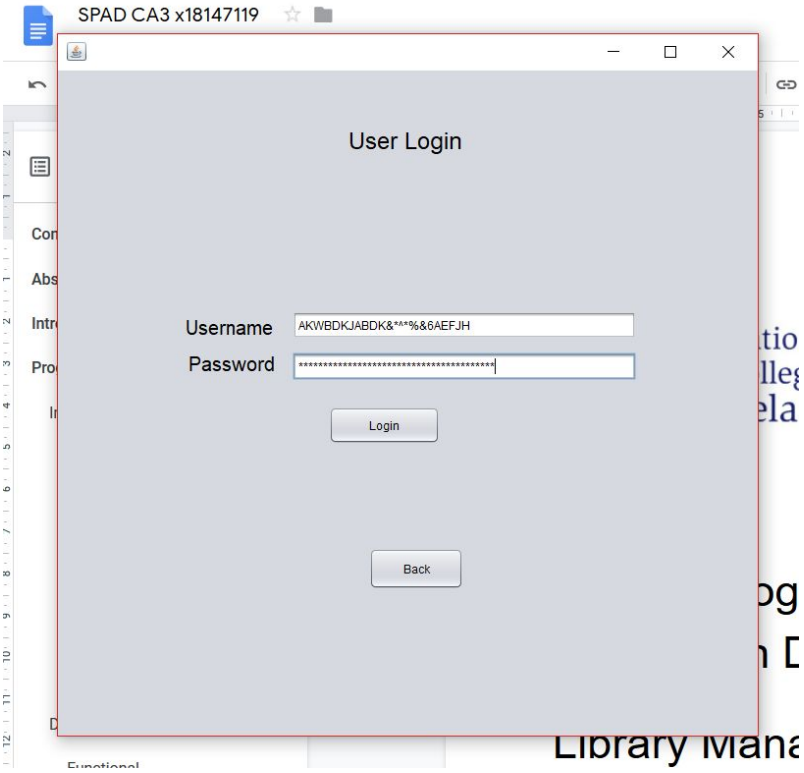
Automated Testing

In Automated Testing, a software is used to test application/ software for vulnerabilities and its outcome to various defined case scenario, these vulnerabilities are checked for security flaws with an existing threat and vulnerabilities database. Approaches such as GUI based and API test driven are used[15].

Manual Code Review

01 UserLogin.java



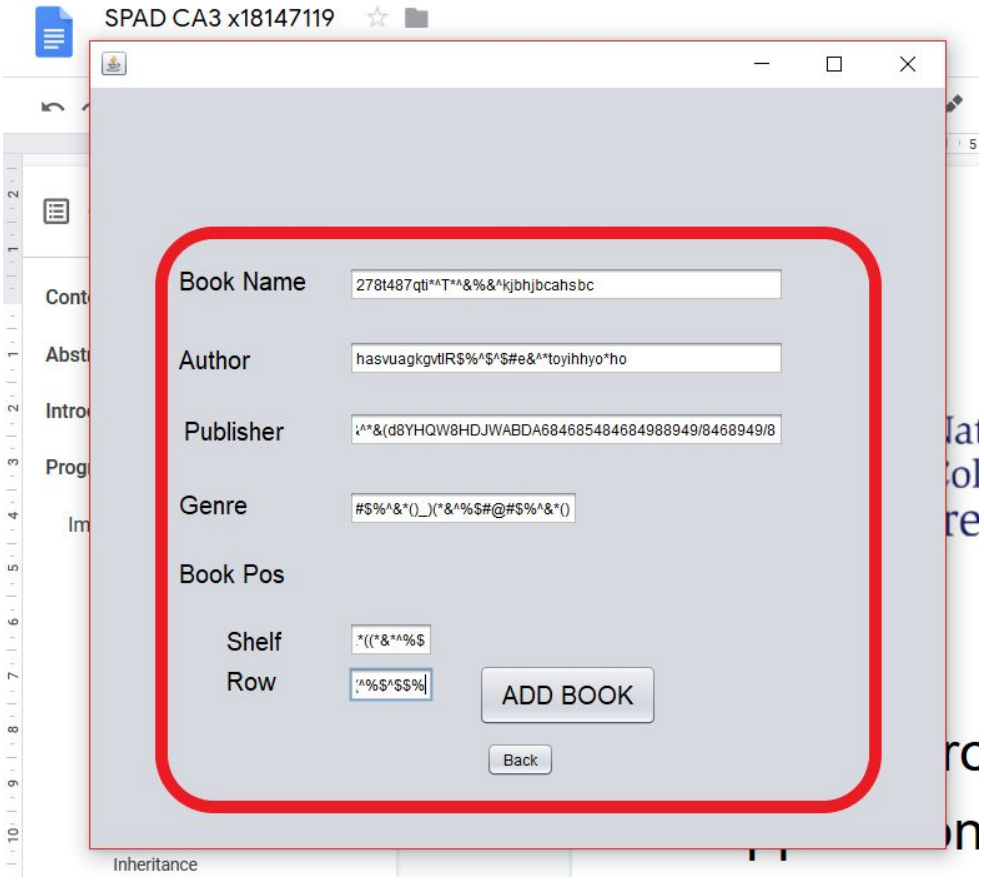



```
UserLogin.java
155     }//GEN-LAST:event_usernameActionPerformed
156
157     private void passwordActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_passwordActionPerformed
158         // TODO add your handling code here:
159     }//GEN-LAST:event_passwordActionPerformed
160
161     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
162         // TODO add your handling code here:
163         String User;
164         User = username.getText();
165         String Pass = String.valueOf(password.getPassword());
166         System.out.println(User + " " + Pass);
167         UsersDao.validate(User, Pass);
168         if (UsersDao.validate(User, Pass)) {
169             this.dispose();
170             UserLoginSuccess.main(new String[]{User, Pass});
171
172         } else {
173             JOptionPane.showMessageDialog(UserLogin.this, "Sorry, Username or Password Error", "Login Error!", JOptionPane.ERROR_
174             username.setText("");
175             password.setText("");
176         }
177
178     }//GEN-LAST:event_jButton1ActionPerformed
```

Logs are not being maintained, also leak of credentials is observed. No option tp reset password.

02 BookForm.java

```
Output × Action Items
securityProject (run) × securityProject (run) #2 × securityProject (run) #3 ×
ant -f "C:\\Users\\lonehawk\\Desktop\\SPW Project\\trainingProject\\securityProject" -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\lonehawk\Desktop\SPW Project\trainingProject\securityProject\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\lonehawk\Desktop\SPW Project\trainingProject\securityProject\build\built-jar.properties
compile:
run:
encosier 1234
Sat Aug 31 16:44:50 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements
Sat Aug 31 16:44:50 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements
Enco Sier 1 enco.cs.doc@gmail.com
Sat Aug 31 16:45:32 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements
Sat Aug 31 16:45:32 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements
Sat Aug 31 16:45:32 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements
com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax
```



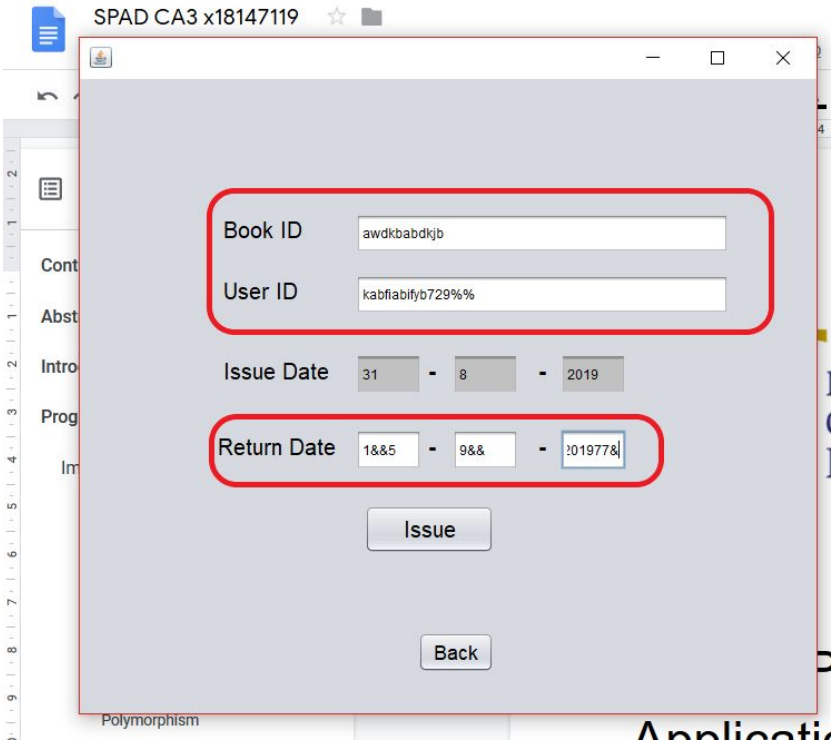

```
DB.java  BookForm.java
232         // TODO add your handling code here:
233         this.dispose();
234         ThisLogined.setVisible(true);
235     }//GEN-LAST:event_jButton2ActionPerformed
236
237     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
238         // TODO add your handling code here:
239         String BookN = BookName.getText();
240         String AuthorN = Author.getText();
241         String PublisherN = Publisher.getText();
242         String ShelfN = Shelf.getText();
243         String RowN = Row.getText();
244         String GenreN = Genre.getText();
245
246         if (BookDao.PublisherValidate(PublisherN)) {
247
248         } else {
249             if (BookDao.AddPublisher(PublisherN) != 0) {
250                 ; // JOptionPane.showMessageDialog(BookForm.this, "Sorry, Publisher can't be added!","Publisher Error!", JOption
251             }
252         }
253         if (BookDao.SaveBook(BookN, AuthorN, PublisherN, ShelfN, RowN, GenreN) != 0) {
254             JOptionPane.showMessageDialog(BookForm.this, "The Book is added!", "Book Added!", JOptionPane.ERROR_MESSAGE);
```

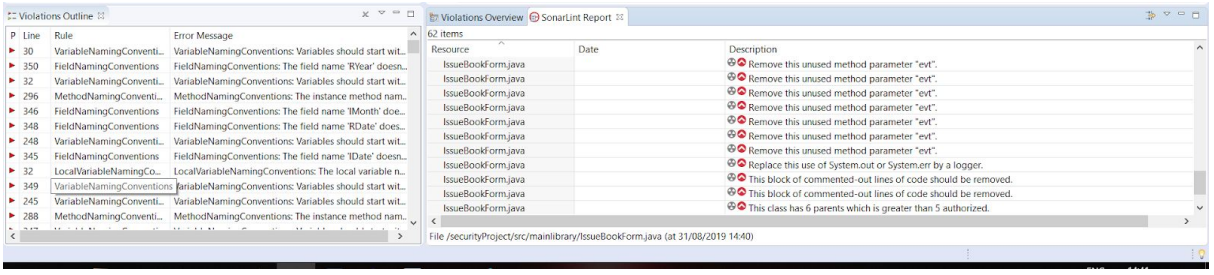
Improper SQL insert query

03 IssueBookForm.java

Result Grid | Filter Rows: | Export:

BookID	UserID	IssueDate	ReturnDate
5	1	2016-11-17	2016-12-02
12	2	2016-11-17	2016-12-02
5	1	2019-08-31	1993-04-08





IssueBookForm.java

```java
        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        int BookIDV;
        BookIDV = Integer.parseInt(BookID.getText());
        int UserIDV;
        UserIDV = Integer.parseInt(UserID.getText());

        String IFDate = IYear.getText() + "-" + IMonth.getText() + "-" + IDate.getText();
        String RFDate = RYear.getText() + "-" + RMonth.getText() + "-" + RDate.getText();
        System.out.println(IFDate);
```

```java
                    JOptionPane.showMessageDialog(IssueBookForm.this, "Unable to Issue Book!", "Issuing Book Error!", JOptionPan
                }
            }

        } else {
            if (TransBookDao.UserValidate(UserID.getText())) {
                JOptionPane.showMessageDialog(IssueBookForm.this, "The Book  is NOT available in Library Database!", "Issuing Boo
            } else if (TransBookDao.BookValidate(BookID.getText())) {
                JOptionPane.showMessageDialog(IssueBookForm.this, "The User is NOT available in Library Database!", "Issuing Book
            } else {
                JOptionPane.showMessageDialog(IssueBookForm.this, "The Book  and User are NOT available in Library Database!", "
            }

        }
        //  if(IssueBookDao.)
```

Date Validation is not present

04 LibrarianDao.java

Result Grid | Filter Rows: | Edit: | Export/Import:

LibrarianID	FullName	UserName	Password	Email
1	Enco Sier	Encosier	1234	enco.cs.doc@gmail.com
2	Leloush Britannia	Zero	9876	leloush.zero@bitannia.com
NULL	NULL	NULL	NULL	NULL

UserLogin.java | UserLoginSuccess.java | UsersDao.java | LibrarianDao.java

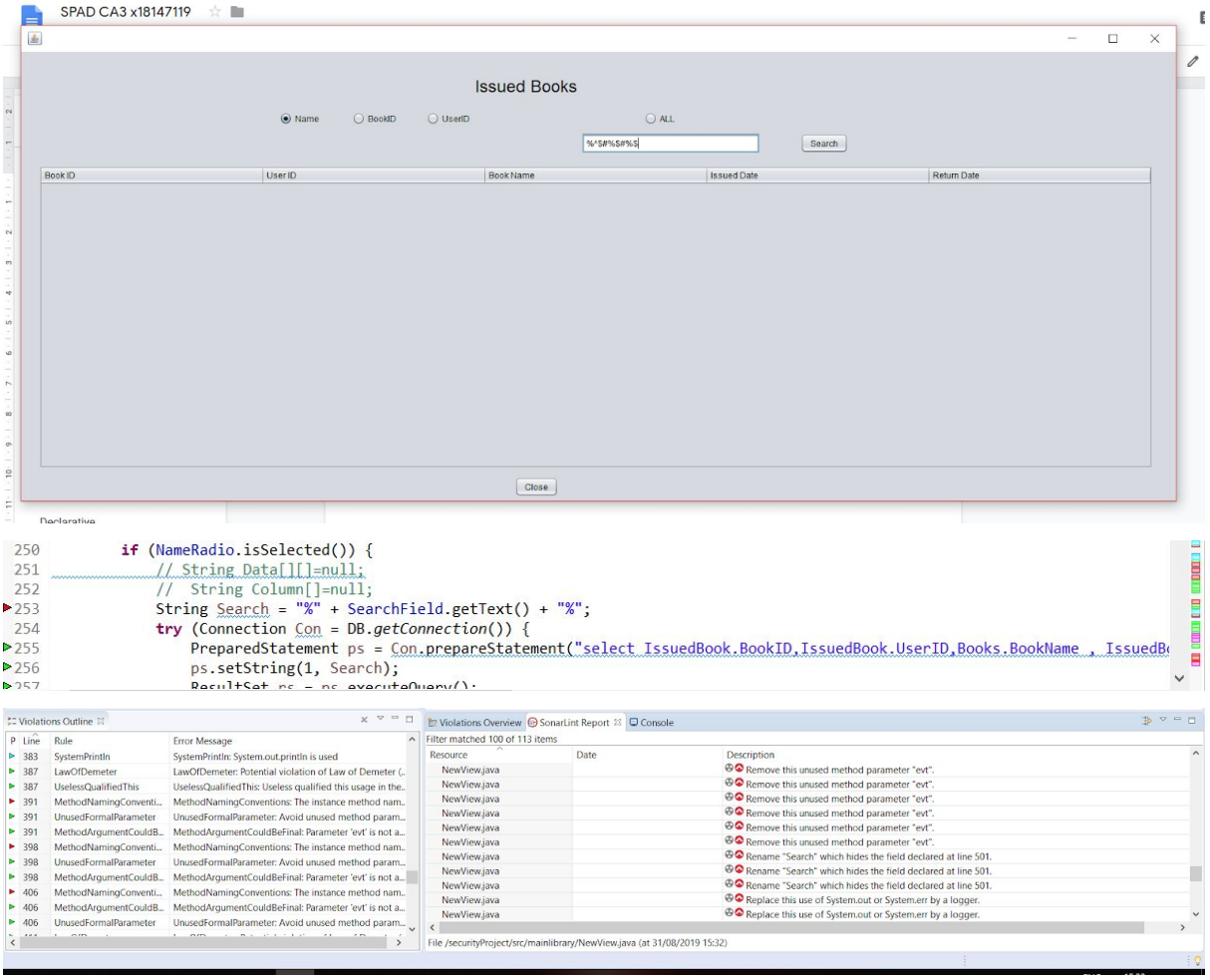
```
33            status = ps.executeUpdate();
34            con.close();
35        } catch (Exception e) {
36            System.out.println(e);
37        }
38        return status;
39    }
40
41    public static boolean validate(String name, String password) {
42        boolean status = false;
43        try {
44            Connection con = DB.getConnection();
45            String select = "select * from Librarian where UserName= '" + name + "' and Password='"+ password +"'";
46            Statement selectStatement = con.createStatement();
47            ResultSet rs = selectStatement.executeQuery(select);
48
49            status = rs.next();
50            con.close();
51        } catch (Exception e) {
52            System.out.println(e);
53        }
54        return status;
55    }
```

UserLogin.java | UserLoginSuccess.java | UsersDao.java | LibrarianDao.java

```
3  import java.sql.*;
4
5  public class LibrarianDao {
6
7      public static int save(String name, String password, String email, String address, String city, String contact) {
8          int status = 0;
9          try {
10
11             Connection con = DB.getConnection();
12             PreparedStatement ps = con.prepareStatement("insert into librarian(name,password,email,address,city,contact) values(?
13             ps.setString(1, name);
14             ps.setString(2, password);
15             ps.setString(3, email);
16             ps.setString(4, address);
17             ps.setString(5, city);
18             ps.setString(6, contact);
19             status = ps.executeUpdate();
20             con.close();
21         } catch (Exception e) {
22             System.out.println(e);
23         }
24         return status;
25     }
```

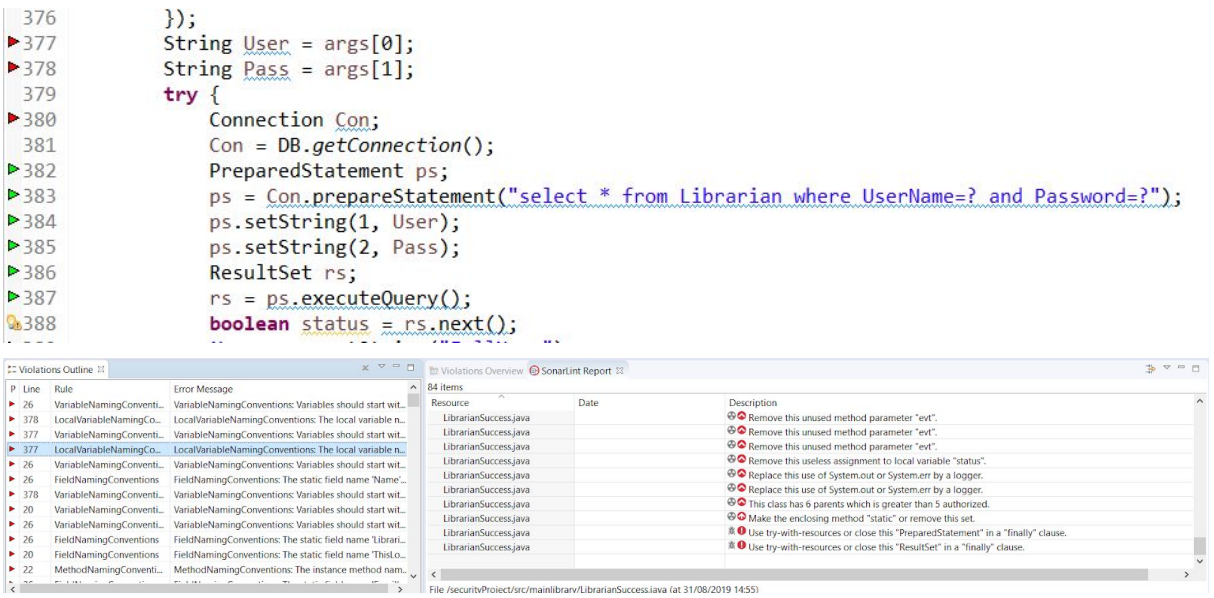
SQL Injection, Select * leads to resource consumption

05 NewView.java



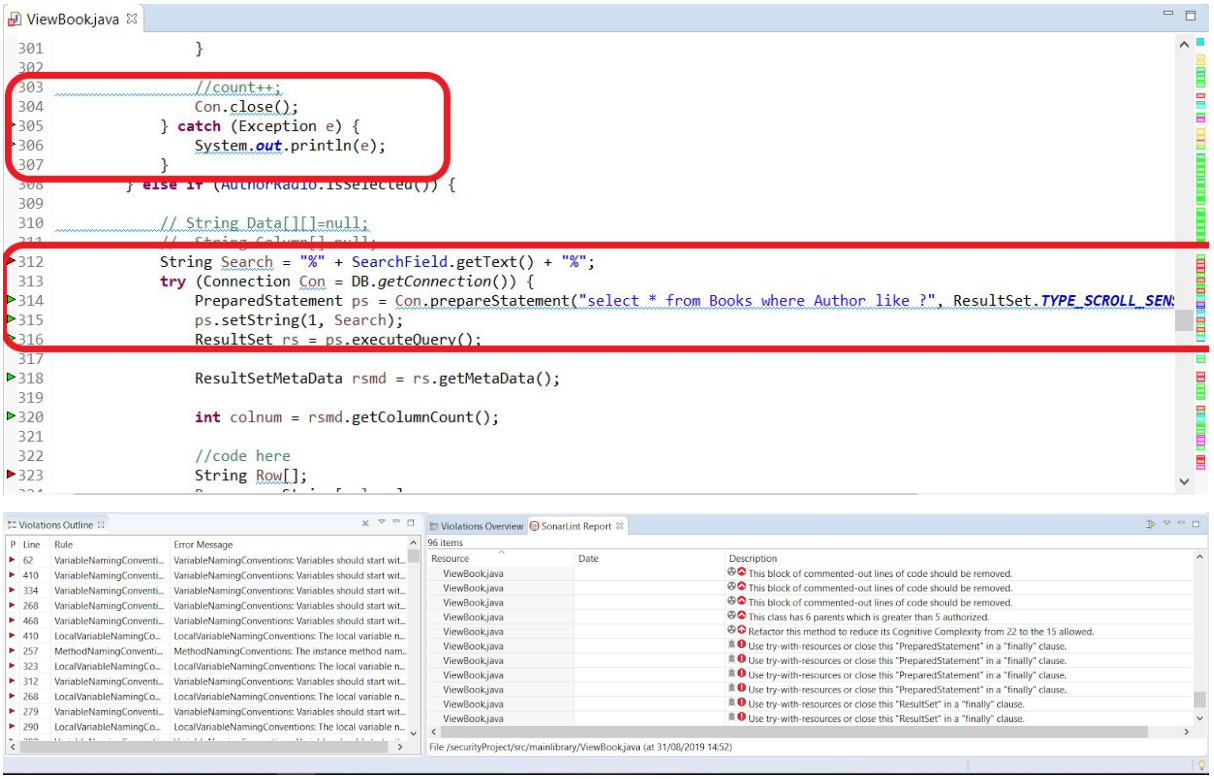
Poor GUI, no validation for text box

06 LibrarianSuccess.java



Use Select* which leads to resources consumption

07 ViewBook.java



Improper Exception Handling

08 UserDao.java

UserLogin.java | UserLoginSuccess.java | UsersDao.java

```java
    }

    public static int AddUser(String User, String UserPass, String UserEmail, String Date) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Te

        int status = 0;
        try {

            Connection con = DB.getConnection();
            PreparedStatement ps = con.prepareStatement("insert into Users(UserPass,RegDate,UserName,Email) values(?,?,?,?)");
            ps.setString(1, UserPass);
            ps.setString(2, Date);
            ps.setString(3, User);
            ps.setString(4, UserEmail);
            status = ps.executeUpdate();
            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
        return status;

    }

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

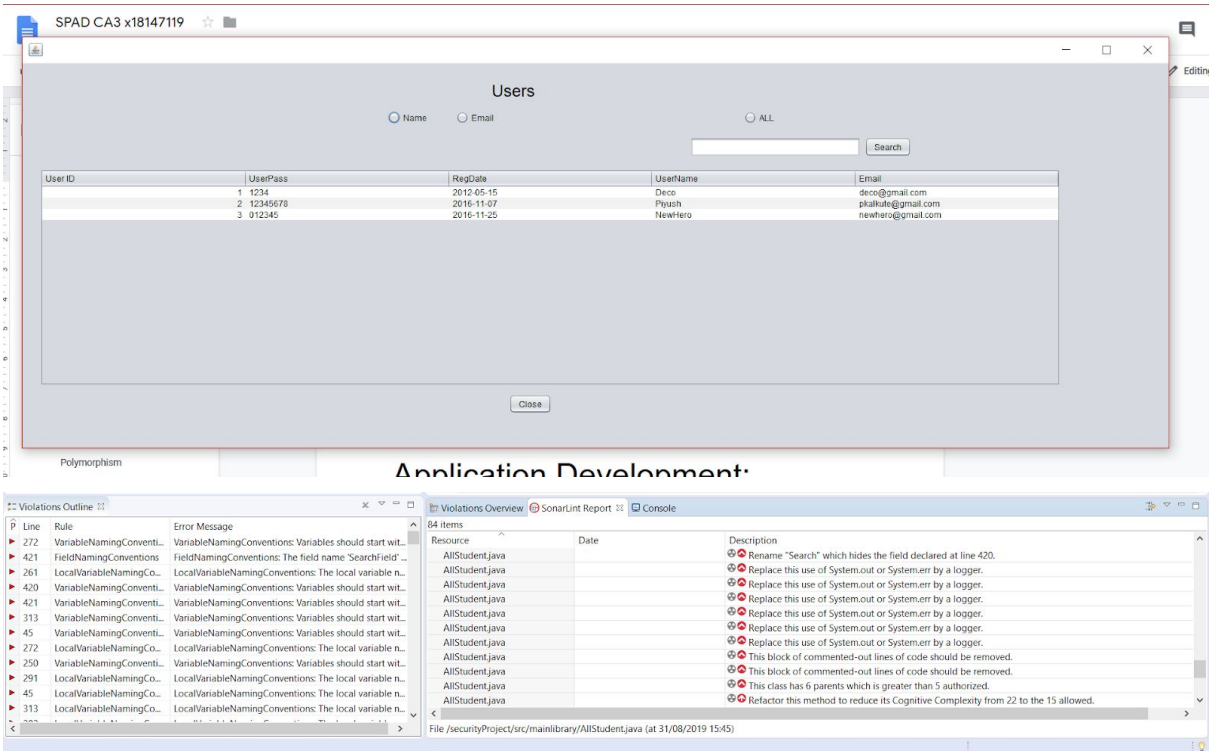
UserID	UserPass	RegDate	UserName	Email
1	1234	2012-05-15	Deco	deco@gmail.com
2	12345678	2016-11-07	Piyush	pkalkute@gmail.com
3	012345	2016-11-25	NewHero	newhero@gmail.com
4	%&^	2019-08-31	awklknfofh	$%%&^'1
NULL	NULL	NULL	NULL	NULL

UserLogin.java | UserLoginSuccess.java | UsersDao.java

```java
/**
 *
 * @author bikash
 */
public class UsersDao {

    public static boolean validate(String name, String password) {
        boolean status = false;
        try {
            Connection con = DB.getConnection();
            String select = "select * from Users where UserName= '" + name + "' and UserPass='"+ password +"'";
            Statement selectStatement = con.createStatement();
            ResultSet rs = selectStatement.executeQuery(select);
            status = rs.next();
            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
        return status;
    }

    public static boolean CheckIfAlready(String UserName) {
        boolean status = false;
        try {
```

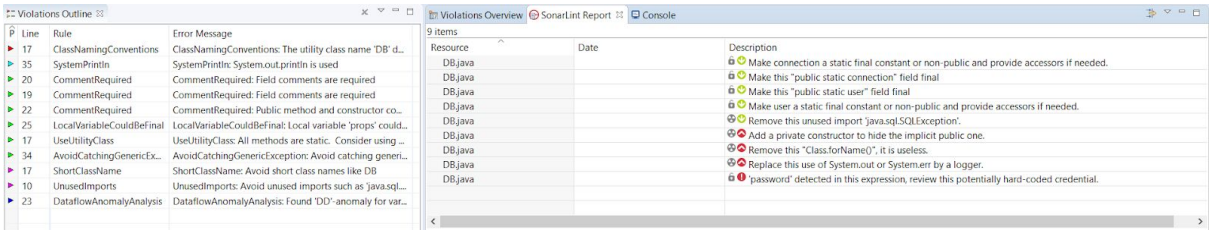
No checks for data field formats, no hashing of password

09 AllStudent.java



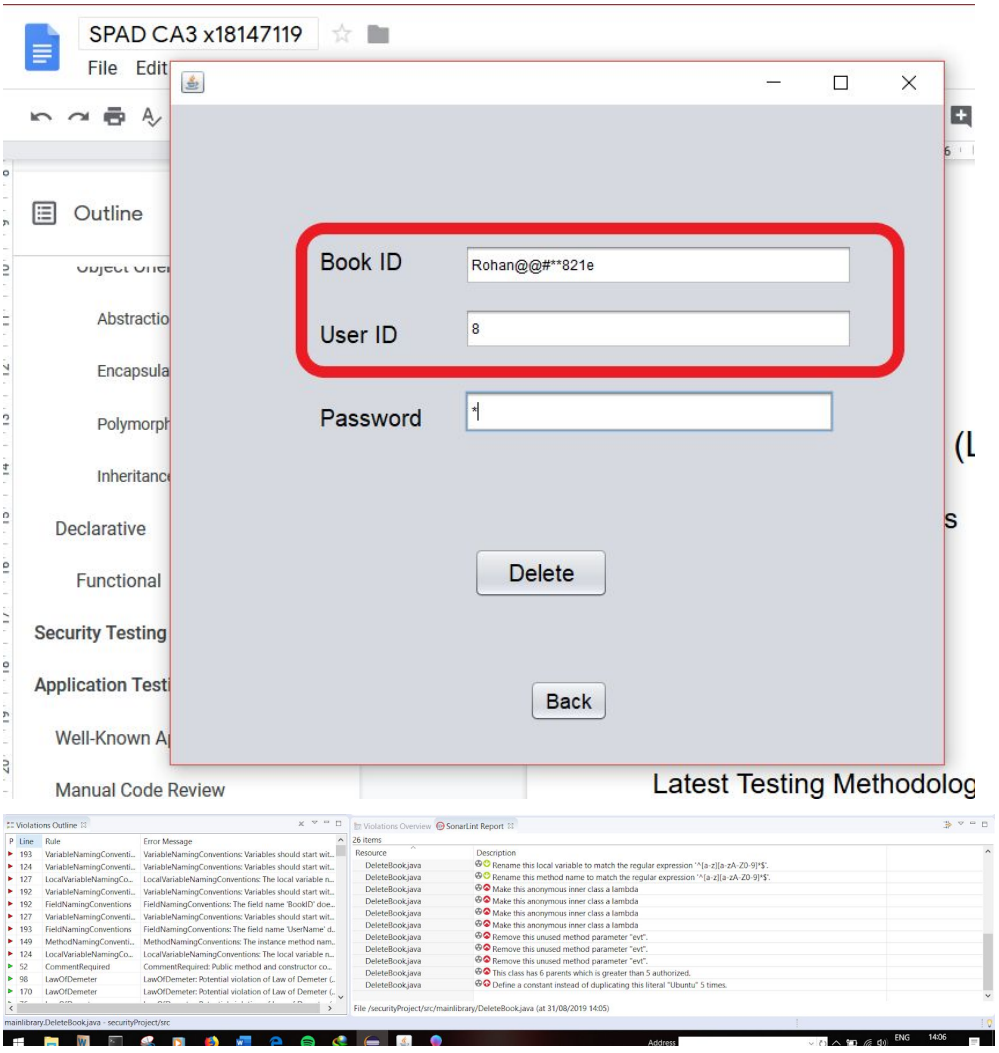
Password visible in plaintext in database

10 DB.java



Direct root access is granted

11 DeleteBookForm.java



Redundant USER ID field, Input Validation not present for numerical fields

12 LibrarianLogin.java

```
Output | Action Items
securityProject (run) | securityProject (run) #2
ant -f "C:\\Users\\lonehawk\\Desktop\\SPW Project\\trainingProject\\securityProject" -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\lonehawk\Desktop\SPW Project\trainingProject\securityProject\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\lonehawk\Desktop\SPW Project\trainingProject\securityProject\build\built-jar.properties
compile:
run:
Deco 1234
Sat Aug 31 16:40:45 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements
Sat Aug 31 16:40:45 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements
Sat Aug 31 16:40:45 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements
```

```
LibrarianLogin.java
        */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(LibrarianLogin.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(LibrarianLogin.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(LibrarianLogin.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(LibrarianLogin.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        LibrarianLogin thiswin;
```

Violations Outline

P	Line	Rule	Error Message
	185	UnusedFormalParameter	UnusedFormalParameter: Avoid unused method param...
	185	MethodArgumentCouldB...	MethodArgumentCouldBeFinal: Parameter 'evt' is not a...
	194	MethodArgumentCouldB...	MethodArgumentCouldBeFinal: Parameter 'args' is not ...
	194	DataflowAnomalyAnalysis	DataflowAnomalyAnalysis: Found 'UR'-anomaly for vari...
	194	DataflowAnomalyAnalysis	DataflowAnomalyAnalysis: Found 'UR'-anomaly for vari...
	196	CommentSize	CommentSize: Comment is too large: Line too long
	197	CommentSize	CommentSize: Comment is too large: Line too long
	198	CommentSize	CommentSize: Comment is too large: Line too long
	201	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'info' could ...
	201	LawOfDemeter	LawOfDemeter: Potential violation of Law of Demeter (...
	203	LawOfDemeter	LawOfDemeter: Potential violation of Law of Demeter (...
	208	LawOfDemeter	LawOfDemeter: Potential violation of Law of Demeter (...

Violations Overview | SonarLint Report

29 items

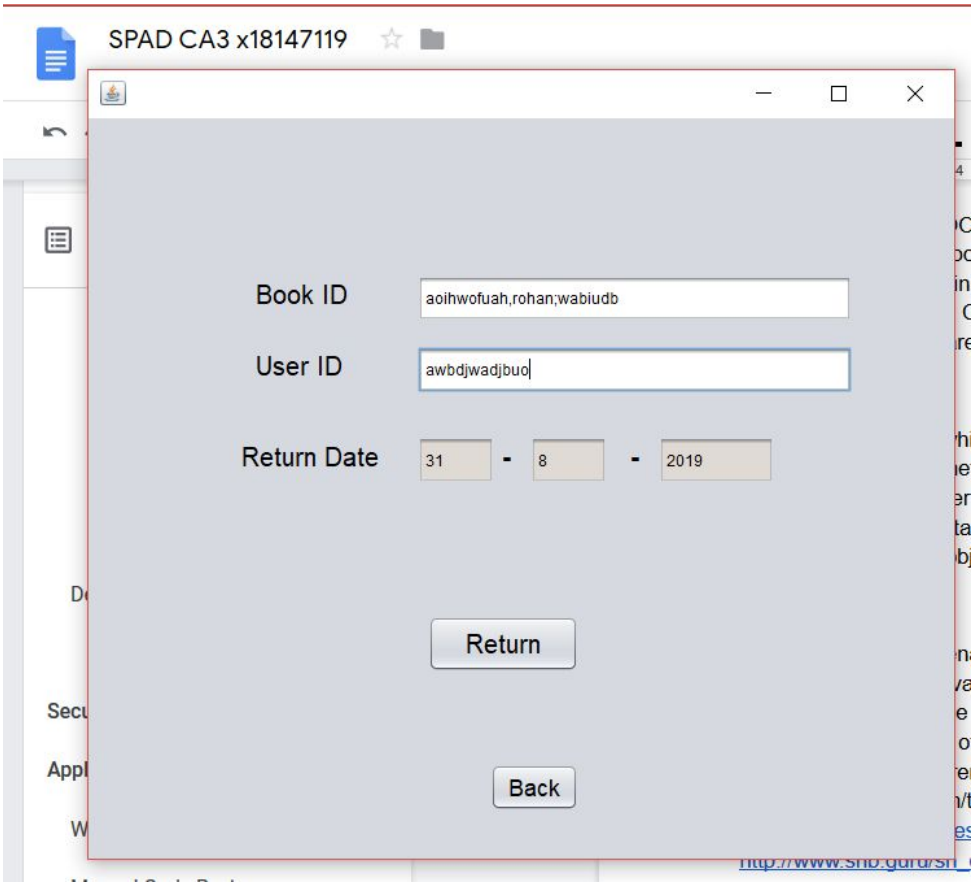
Resource	Date	Description
LibrarianLogin.java		Move the array designator from the variable to the type.
LibrarianLogin.java		Remove the "jButton1" field and declare it as a local variable in the relevant methods.
LibrarianLogin.java		Remove the "jButton2" field and declare it as a local variable in the relevant methods.
LibrarianLogin.java		Remove the "jLabel1" field and declare it as a local variable in the relevant methods.
LibrarianLogin.java		Remove the "jLabel2" field and declare it as a local variable in the relevant methods.
LibrarianLogin.java		Remove the "jLabel3" field and declare it as a local variable in the relevant methods.
LibrarianLogin.java		Remove the "jPanel1" field and declare it as a local variable in the relevant methods.
LibrarianLogin.java		Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
LibrarianLogin.java		Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
LibrarianLogin.java		Make this anonymous inner class a lambda
LibrarianLogin.java		Make this anonymous inner class a lambda

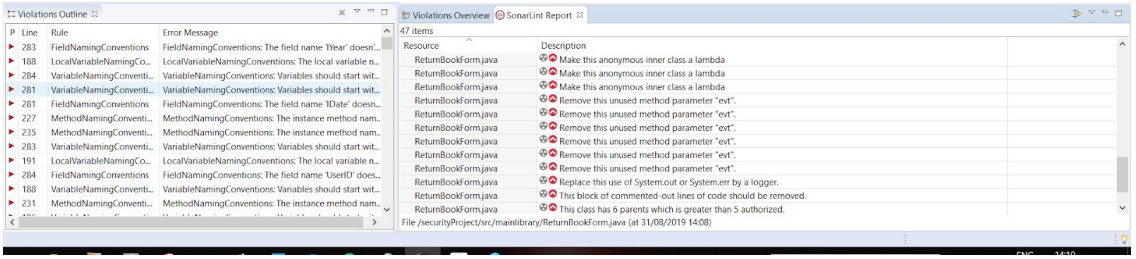
File /securityProject/src/mainlibrary/LibrarianLogin.java (at 31/08/2019 15:05)

```
import javax.swing.JOptionPane;

/**
 *
 * @author bikash
 */
public class LlibrarianLogin extends javax.swing.JFrame {

    private static void setvisible(boolean b) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Temp
    }

```

Multiple lines of same code.

13 ReturnBookForm.java





Result Grid | Filter Rows: | Export: | Wrap Cell Content:

BookID	UserID	IssueDate	ReturnDate
5	1	2016-11-17	2016-12-02
12	2	2016-11-17	2016-12-02
6	2	2016-11-17	2016-12-02

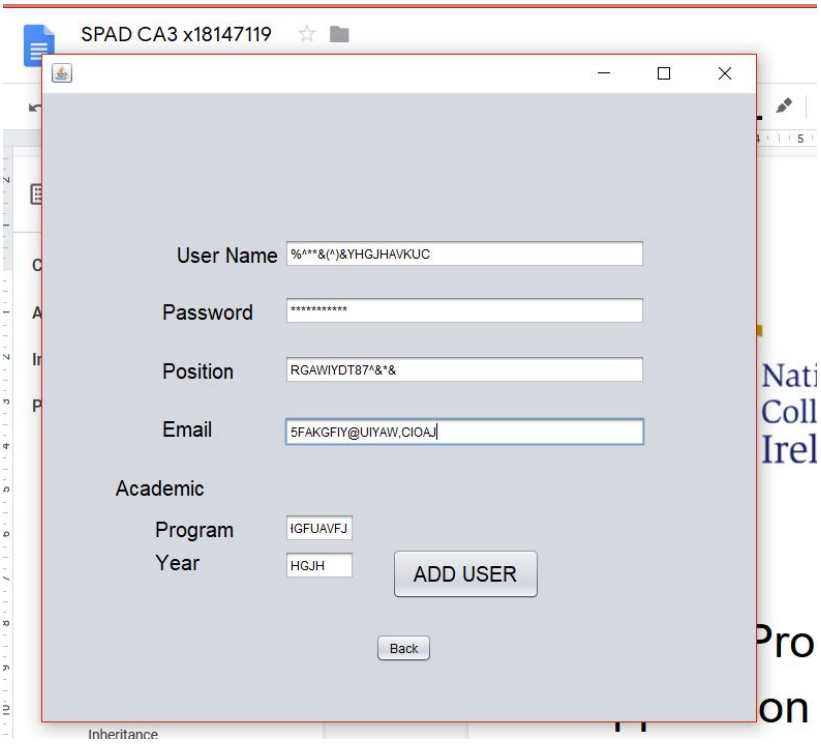
Result Grid | Filter Rows: | Export: | W

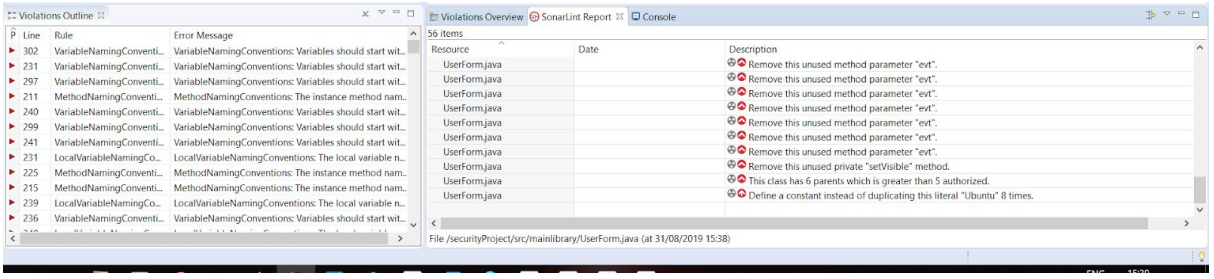
BookID	UserID	IssueDate	ReturnDate
5	1	2016-11-17	2016-12-02
12	2	2016-11-17	2016-12-02

\

No record keeping is practiced.

14 UserForm.java





No validations for email and numerical fields

15 UserLoginSuccess.java

```
UserLogin.java    UserLoginSuccess.java
249                 }
250         });
251
252         String User = args[0];
253         String Pass = args[1];
254         try {
255             Connection Con;
256             Con = DB.getConnection();
257             PreparedStatement ps;
258             ps = Con.prepareStatement("select * from Users where UserName=? and UserPass=?");
259             ps.setString(1, User);
260             ps.setString(2, Pass);
261             ResultSet rs;
262             rs = ps.executeQuery();
263             boolean status = rs.next();
264             GetName = User;
265             GetRegDate = rs.getString("RegDate");
266             GetEmail = rs.getString("Email");
267             GetUserID = rs.getString("UserID");
268             Con.close();
269
270         } catch (Exception f) {
271             System.out.println(f);
272         }
```

SQL Injection Vulnerability

Proposed Solutions

File	Line	Issue	Solution
UserLogin.java	10,165	No encryption for sensitive data such as user passwords	Encryption or Hashing with Salt should be implemented Use of PB2KDF or BCrypt
BookForm.java	17	Class BookForm is public which gives malicious user to extend the class for improper use.	Use of Public final modifier should be used, as per OWASP recommendations
IssueBookForm.java	17	Class IssueBookForm is public which gives malicious user to extend the class for improper use.	Use of Public final modifier should be used, as per OWASP recommendations

LibrarianDao.java	14	No hashing is done before it is passed into insert statement	Passwords should be hashed
NewView.java	18	Access modifier is set to public, data exposed	Set access modifier from public to protected
LibrarianSuccess.java	18	Catch Exception generic, which can expose content specific information in the error messages	Exceptions should be used accordingly to the use case Use of SQLException should be made
ViewBook.java	73,305	Improper exception handling, no input validation	Catch should there for handling SQLException
UserDao.java	17, 57,64	Class UserDao is public which gives malicious user to extend the class for improper use. No validation on data, improper exception handling, no hashing	Use of Public final modifier should be used, as per OWASP recommendations, implement input validation and use of hashing algorithm and exception handling
AllStudent.java	30	Data is exposed, missing encapsulation	Constructor should be private
DB.java	26	SSL error messages visible	Connection variable to SSL should be set to True
DeleteBookForm.java	28	Missing input validation in numerical fields	Input Validation should be implemented
LibrarianLogin.java	170	Missing encryption	Hashing with salt should be used.
ReturnBookForm.java	15	Class BookForm is public which gives malicious user to extend the class for improper use, input	Use of Public final modifier should be used, as per OWASP recommendations,

		data validation missing, no logs/records maintained	logging should be done and data validation should be done
UserForm.java	248	Missing encryption, sensitive information not encrypted	Use of Encryption or Hashing should be done
UserLoginSuccess.java	270	Input data fields are not sanitized, use of credentials to fetch user information. Generic Catch Exception	Input data validation and sanitization should be used. Catch should be declared for exception

Conclusion

It was observed that, security should be an integral part of the software development process and code reviews and static/dynamic analysis iterations should be carried out at each phase. Not only good coding standards or tested or proven technologies usage is necessary but also design architecture and flow is equally important.

Youtube:https://youtu.be/sdeNcgEIJ9o

References

[1]
Ward Solutions. (2019). *Penetration Testing Experts - Ward Solutions*. [online] Available at: https://www.ward.ie/cyber-security-testing/ [Accessed 21 Aug. 2019].

[2]
“Top 10 Open Source Security Testing Tools for Web Applications (Updated),” *Hackr.io Blog*. 28-Nov-2018 [Online]. Available: https://hackr.io/blog/top-10-open-source-security-testing-tools-for-web-applications. [Accessed: 23-Aug-2019]

[3]
“Static Testing vs. Dynamic Testing,” *Veracode*, 03-Dec-2013. [Online]. Available: https://www.veracode.com/blog/2013/12/static-testing-vs-dynamic-testing. [Accessed: 29-Aug-2019]

[4]
“Source Code Analysis Tools - OWASP.” [Online]. Available: https://www.owasp.org/index.php/Source_Code_Analysis_Tools. [Accessed: 28-Aug-2019]

[5]
“Introduction of Programming Paradigms,” *GeeksforGeeks*. 12-Oct-2018 [Online]. Available: https://www.geeksforgeeks.org/introduction-of-programming-paradigms/. [Accessed: 21-Aug-2019]

[6]
“The JVM Tool Interface (JVM TI): How VM Agents Work.” [Online]. Available: https://www.oracle.com/technetwork/articles/javase/index-140680.html. [Accessed: 28-Aug-2019]

[7]
T. Singh, “New Learning Methodology for Student of Java Programming Language” [Online]. Available: https://www.academia.edu/11868737/New_Learning_Methodology_for_Student_of_Java_Programming_Language. [Accessed: 22-Aug-2019]

[8]
J. Edwards, *Imperative to Functional Programming Essentials*. USA: CreateSpace Independent Publishing Platform, 2016.

[9]
“Abstract Methods and Classes (The Java™ Tutorials > Learning the Java Language > Interfaces and Inheritance).” [Online]. Available: https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html. [Accessed: 21-Aug-2019]

[10]
“Lesson 8: Object-Oriented Programming.” [Online]. Available: https://www.oracle.com/technetwork/java/oo-140949.html. [Accessed: 23-Aug-2019]

[11]
M. O. Schwager, “Programming paradigms reloaded,” *Phoenix IT MOS - blog*, 29-May-2016. [Online]. Available: https://phoenix-it-mos.net/2016/05/programming-paradigms-reloaded/. [Accessed: 24-Aug-2019]

[12]
DAT10603 Programming Principle. PediaPress.

[13]
P. Čisar and S. M. Čisar, “The framework of runtime application self-protection technology,” in *2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2016, pp. 000081–000086.

[14]
“What Is RASP | Veracode.” [Online]. Available: https://info.veracode.com/what-is-rasp-infosheet-resource.html. [Accessed: 28-Aug-2019]

[15]
“Security by Design Principles - OWASP.” [Online]. Available: https://www.owasp.org/index.php/Security_by_Design_Principles. [Accessed: 24-Aug-2019]

[16]
“OWASP Secure Software Development Lifecycle Project - OWASP.” [Online]. Available: https://www.owasp.org/index.php/OWASP_Secure_Software_Development_Lifecycle_Project. [Accessed: 25-Aug-2019]

[17]
“Secure Development Practices Aren’t Enough,” *SAFECode*, 04-Apr-2019. [Online]. Available: https://safecode.org/secure-development-practices-arent-enough/. [Accessed: 30-Aug-2019]

[18]

Van Roy, Peter. (2012). Programming Paradigms for Dummies: What Every Programmer Should Know. [Accessed: 30-Aug-2019]