Sonar Qube - Community Edition Version 9.2 (build 49834)

1. **Kategorija: Cross-Site Request Forgery (CSRF)**
2. ***Make sure disabling Spring Security's CSRF protection is safe here.***

**Status:** True positive

**Komentar:** U okviru koda je namerno ugašena zaštita Spring framework-a za CSRF i to je potrebno ispraviti.

**Kod:**

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeRequests()

.antMatchers("/login").permitAll()

.antMatchers("/\*\*").authenticated()

.and()

.formLogin()

1. **Kategorija**: **SQL Injection**
2. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Nebezbedan način za egzekuciju SQL komandi, preko executeQuery metode koja kao parametar prima string koji se šalje kao upit, uz korišćenje string-a id za formiranje upita ka bazi podataka. Iako je id za restoran u okviru baze integer, to svakako ne menja stanje, budući da napadač može da izvrši SQL Injection.

**Kod:**

|  |
| --- |
| public Object getRestaurant(String id) { |
|  |

|  |
| --- |
| String query = "SELECT r.id, r.name, r.address, rt.name FROM restaurant AS r JOIN restaurant\_type AS rt ON r.typeId = rt.id WHERE r.id=" + id; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(query)) { |
|  |

|  |
| --- |
| if (rs.next()) { |
|  |

|  |
| --- |
| return createRestaurant(rs); |
|  |

|  |
| --- |
| } |
|  |
|  |

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** False positive

**Komentar:** Generalno nebezbedan način za brisanje iz baze podataka uz korišćenje executeUpdate metode sa string parametrom koji sadrži upit, međutim ovde to nije problem, budući da se za formiranje upita koristi integer, pa SQL Injection ne može da se desi. Trebalo bi ispraviti kod u svakom slučaju, iako trenutno nije rizik.

**Kod:**

|  |
| --- |
| public void deleteRestaurant(int id) { |
|  |

|  |
| --- |
| String query = "DELETE FROM restaurant WHERE id=" + id; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement() |
|  |

|  |
| --- |
| ) { |
|  |

|  |
| --- |
| statement.executeUpdate(query); |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

|  |
| --- |
| } |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeUpdate sa string parametrom, ali u ovom slučaju je SQL Injection moguće odraditi, budući da klasa RestaurantUpdate sadrži stringove za polja. Potrebno je ispraviti kod da se koristi PreparedStatement klasa.

**Kod:**

|  |
| --- |
| public void updateRestaurant(RestaurantUpdate restaurantUpdate) { |
|  |

|  |
| --- |
| String query = "UPDATE restaurant SET name = '" + restaurantUpdate.getName() + "', address='" + restaurantUpdate.getAddress() + "', typeId =" + restaurantUpdate.getRestaurantType() + " WHERE id =" + restaurantUpdate.getId(); |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement() |
|  |

|  |
| --- |
| ) { |
|  |

|  |
| --- |
| statement.executeUpdate(query); |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć id-a koji je string, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public Customer updateCustomerAddress  (String id) { |
|  |

|  |
| --- |
| String sqlQuery = "SELECT id, username, password FROM users WHERE id=" + id; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery)) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if (rs.next()) { |
|  |

|  |
| --- |
| return createCustomerWithPassword(rs); |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeUpdate sa string parametrom, gde se upit formira uz pomoć id-a koji je string, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public void deleteCustomer(String id) { |
|  |

|  |
| --- |
| String query = "DELETE FROM users WHERE id=" + id; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement() |
|  |

|  |
| --- |
| ) { |
|  |

|  |
| --- |
| statement.executeUpdate(query); |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |
|  |

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeUpdate sa string parametrom, gde se upit formira uz pomoć polja iz CustomerUpdate klase, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public void updateCustomer(CustomerUpdate customerUpdate) { |
|  |

|  |
| --- |
| String query = "UPDATE users SET username = '" + customerUpdate.getUsername() + "', password='" + customerUpdate.getPassword() + "' WHERE id =" + customerUpdate.getId(); |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement() |
|  |

|  |
| --- |
| ) { |
|  |

|  |
| --- |
| statement.executeUpdate(query); |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

|  |
| --- |
| } |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć string-a id, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public List<Address> getAddresses(String id) { |
|  |

|  |
| --- |
| String sqlQuery = "SELECT id, name FROM address WHERE userId=" + id; |
|  |

|  |
| --- |
| List<Address> addresses = new ArrayList<Address>(); |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery)) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| while (rs.next()) { |
|  |

|  |
| --- |
| addresses.add(createAddress(rs)); |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** False positive

**Komentar:** Ponovo se koristi metoda executeUpdate sa string parametrom, gde se upit formira uz pomoć integer-a id, pa SQLi nije moguće odraditi.

**Kod:**

|  |
| --- |
| public void deleteCustomerAddress(int id) { |
|  |

|  |
| --- |
| String query = "DELETE FROM address WHERE id=" + id; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement() |
|  |

|  |
| --- |
| ) { |
|  |

|  |
| --- |
| statement.executeUpdate(query); |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |
|  |

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeUpdate sa string parametrom, gde se upit formira uz pomoć integer-a id i stringa name, iz klase Address, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public void updateCustomerAddress(Address address) { |
|  |

|  |
| --- |
| String query = "UPDATE address SET name = '" + address.getName() + "' WHERE id =" + address.getId(); |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement() |
|  |

|  |
| --- |
| ) { |
|  |

|  |
| --- |
| statement.executeUpdate(query); |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

|  |
| --- |
| } |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeUpdate sa string parametrom, gde se upit formira uz pomoć name string-a i userId integer-a, iz klase NewAddress, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public void putCustomerAddress(NewAddress newAddress) { |
|  |

|  |
| --- |
| String query = "INSERT INTO address (name, userId) VALUES ('"+newAddress.getName()+"' , "+newAddress.getUserId()+")"; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement() |
|  |

|  |
| --- |
| ) { |
|  |

|  |
| --- |
| statement.executeUpdate(query); |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć string-a id, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public ViewableDelivery getDelivery(String id) { |
|  |

|  |
| --- |
| String sqlQuery = "SELECT d.id, d.isDone, d.date, d.comment, u.username, r.name, rt.name, a.name FROM delivery AS d JOIN users AS u ON d.userId = u.id JOIN restaurant as r ON d.restaurantId = r.id JOIN address AS a ON d.addressId = a.id JOIN restaurant\_type AS rt ON r.typeId= rt.id WHERE d.id = " + id; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery)) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if (rs.next()) { |
|  |

|  |
| --- |
| return createDelivery(rs); |
|  |

|  |
| --- |
| } |
|  |
|  |

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć string-a id, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| List<DeliveryDetail> details = new ArrayList<>(); |
|  |

|  |
| --- |
| String sqlQuery = "SELECT di.id, di.amount, f.name, f.price FROM delivery\_item AS di JOIN food AS f ON di.foodId = f.id WHERE deliveryId = " + id; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery)) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| while (rs.next()) { |
|  |

|  |
| --- |
| details.add(createDetail(rs)); |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć string-a searchQuery, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| + "OR UPPER(r.name) LIKE UPPER('%" + searchQuery + "%')" |
|  |

|  |
| --- |
| + "OR UPPER(rt.name) LIKE UPPER('%" + searchQuery + "%')" |
|  |

|  |
| --- |
| + "OR UPPER(a.name) LIKE UPPER('%" + searchQuery + "%')"; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery)) { |
|  |

|  |
| --- |
| while (rs.next()) { |
|  |

|  |
| --- |
| cars.add(createDelivery(rs)); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

return cars;

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć string-a username, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public HashedUser findUser(String username) { |
|  |

|  |
| --- |
| String sqlQuery = "select passwordHash, salt, totpKey from hashedUsers where username = '" + username + "'"; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery)) { |
|  |

|  |
| --- |
| if (rs.next()) { |
|  |

|  |
| --- |
| String passwordHash = rs.getString(1); |
|  |

|  |
| --- |
| String salt = rs.getString(2); |
|  |

|  |
| --- |
| String totpKey = rs.getString(3); |
|  |

return new HashedUser(username, passwordHash, salt, totpKey);

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** False positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć integer-a id, pa SQLi nije moguće odraditi.

**Kod:**

|  |
| --- |
| public List<Food> getMenu(int id) { |
|  |

|  |
| --- |
| List<Food> menu = new ArrayList<>(); |
|  |

|  |
| --- |
| String sqlQuery = "SELECT id, name FROM food WHERE restaurantId=" + id; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery)) { |
|  |

|  |
| --- |
| while (rs.next()) { |
|  |

|  |
| --- |
| menu.add(createFood(rs)); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

} catch (SQLException e) {

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeUpdate sa string parametrom, gde se upit formira uz pomoć integer-a address i restaurantid iz NewOrder klase, međutim, budući da se koristi i string komentar, moguće je odraditi SQLi.

**Kod:**

|  |
| --- |
| "values (FALSE, " + userId + ", " + newOrder.getRestaurantId() + ", " + newOrder.getAddress() + "," + |
|  |

|  |
| --- |
| "'" + date.getYear() + "-" + date.getMonthValue() + "-" + date.getDayOfMonth() + "', '" + newOrder.getComment() + "')"; |
|  |

|  |
| --- |
| try { |
|  |

|  |
| --- |
| Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| statement.executeUpdate(sqlQuery); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| sqlQuery = "SELECT MAX(id) FROM delivery"; |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery); |
|  |

|  |
| --- |
|  |
|  |

if (rs.next()) {

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** False positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira bez ulaznih podataka, pa SQLi nije moguće odraditi.

**Kod:**

|  |
| --- |
| Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| statement.executeUpdate(sqlQuery); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| sqlQuery = "SELECT MAX(id) FROM delivery"; |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if (rs.next()) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int deliveryId = rs.getInt(1); |
|  |

sqlQuery = "INSERT INTO delivery\_item (amount, foodId, deliveryId)" +

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** False positive

**Komentar:** Ponovo se koristi metoda executeUpdate sa string parametrom, gde se upit formira uz pomoć integer-a, pa SQLi nije moguće odraditi.

**Kod:**

|  |
| --- |
| } |
|  |

|  |
| --- |
| deliveryItem += "(" + item.getAmount() + ", " + item.getFoodId() + ", " + deliveryId + ")"; |
|  |

|  |
| --- |
| sqlQuery += deliveryItem; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| System.out.println(sqlQuery); |
|  |

|  |
| --- |
| statement.executeUpdate(sqlQuery); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** False positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć integer-a, pa SQLi nije moguće odraditi.

**Kod:**

|  |
| --- |
| public Object getAddresses(int userId) { |
|  |

|  |
| --- |
| List<Address> addresses = new ArrayList<>(); |
|  |

|  |
| --- |
| String sqlQuery = "SELECT id, name FROM address WHERE userId=" + userId; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(sqlQuery)) { |
|  |

|  |
| --- |
| while (rs.next()) { |
|  |

|  |
| --- |
| addresses.add(createAddress(rs)); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

} catch (SQLException e) {

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** False positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć integer-a, pa SQLi nije moguće odraditi.

**Kod:**

|  |
| --- |
| public List<Permission> findByRoleId(int roleId) { |
|  |

|  |
| --- |
| List<Permission> permissions = new ArrayList<>(); |
|  |

|  |
| --- |
| String query = "SELECT id, name FROM permissions WHERE id IN (SELECT permissionId FROM role\_to\_permissions WHERE roleId=" + roleId + ")"; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(query)) { |
|  |

|  |
| --- |
| while (rs.next()) { |
|  |

|  |
| --- |
| int id = rs.getInt(1); |
|  |

|  |
| --- |
| String name = rs.getString(2); |
|  |

|  |
| --- |
| permissions.add(new Permission(id, name)); |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** False positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć integer-a, pa SQLi nije moguće odraditi.

**Kod:**

|  |
| --- |
| public List<Role> findByUserId(int userId) { |
|  |

|  |
| --- |
| List<Role> roles = new ArrayList<>(); |
|  |

|  |
| --- |
| String query = "SELECT id, name FROM roles WHERE id IN (SELECT roleId FROM user\_to\_roles WHERE userId=" + userId + ")"; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(query)) { |
|  |

|  |
| --- |
| while (rs.next()) { |
|  |

|  |
| --- |
| int id = rs.getInt(1); |
|  |

|  |
| --- |
| String name = rs.getString(2); |
|  |

|  |
| --- |
| roles.add(new Role(id, name)); |
|  |

}

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć string-a username, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public User findUser(String username) { |
|  |

|  |
| --- |
| String query = "SELECT id, username, password FROM users WHERE username='" + username + "'"; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(query)) { |
|  |

|  |
| --- |
| if (rs.next()) { |
|  |

|  |
| --- |
| int id = rs.getInt(1); |
|  |

|  |
| --- |
| String username1 = rs.getString(2); |
|  |

|  |
| --- |
| String password = rs.getString(3); |
|  |

return new User(id, username1, password);

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** True positive

**Komentar:** Ponovo se koristi metoda executeQuery sa string parametrom, gde se upit formira uz pomoć string-ova username i password, pa je SQLi moguće odraditi.

**Kod:**

|  |
| --- |
| public boolean validCredentials(String username, String password) { |
|  |

|  |
| --- |
| String query = "SELECT username FROM users WHERE username='" + username + "' AND password='" + password + "'"; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ResultSet rs = statement.executeQuery(query)) { |
|  |

|  |
| --- |
| return rs.next(); |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

|  |
| --- |
| } |
|  |

return false;

1. ***Make sure using a dynamically formatted SQL query is safe here.***

**Status:** False positive

**Komentar:** Ponovo se koristi metoda executeUpdate sa string parametrom, gde se upit formira uz pomoć integer-a userId, pa SQLi nije moguće odraditi.

**Kod:**

|  |
| --- |
| public void delete(int userId) { |
|  |

|  |
| --- |
| String query = "DELETE FROM users WHERE id = " + userId; |
|  |

|  |
| --- |
| try (Connection connection = dataSource.getConnection(); |
|  |

|  |
| --- |
| Statement statement = connection.createStatement(); |
|  |

|  |
| --- |
| ) { |
|  |

|  |
| --- |
| statement.executeUpdate(query); |
|  |

|  |
| --- |
| } catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

}

1. **Kategorija: Insecure Configuration (CSRF)**

**Broj sigurnosnih problema:** 25

**Status:** True positive

**Komentar:** U okviru posmatranog projekta u 25 slučajeva se javlja hvatanje izuzetka tipa SQLException, pri čemu se radi printStackTrace() za izuzetke, što predstavlja informacije za debagovanje. Ostavljanje ovakvih informacija u okviru koda koji se nalazi u produkciji je nesigurno jer napadaču daje dodatne informacije i uvid u deo programskog sistema. Trebalo bi izbegavati ostavljanje ovakvog koda u produkciji. Problematičan deo koda dat je u nastavku:

|  |
| --- |
| catch (SQLException e) { |
|  |

|  |
| --- |
| e.printStackTrace(); |
|  |

}