WEB ROUTING

SECURITY MANUAL

SLIPPERY ROCK UNIVERSITY

Thomas Haley – [tjh1019@sru.edu](mailto:tjh1019@sru.edu)

Beth Orgovan – [bro0700@sru.edu](mailto:bro0700@sru.edu)

Dakota Myers – [drm1022@sru.edu](mailto:drm1022@sru.edu)

Sinchana Kori – [ssk1002@sru.edu](mailto:ssk1002@sru.edu)

Contents

[1. Synopsis 3](#_Toc133064184)

[2. Roles 3](#_Toc133064185)

[3.Access Limitations 3](#_Toc133064186)

[3.1 PErmissions based denials 3](#_Toc133064187)

[3.2 persistence of permissions denials in html 4](#_Toc133064188)

[3.3 ENSURING DENIALS VIA WEB SECURITY CONFIGURATION 6](#_Toc133064189)

[4. Password Security 6](#_Toc133064190)

[4.1 PASSWORD HASHING 6](#_Toc133064191)

[5. User entity requirements 7](#_Toc133064192)

[5.1 Username requirements 7](#_Toc133064193)

[5.2 PASSWORD REQUIREMENTS 7](#_Toc133064194)

[5.3 EMAIL REQUIREMENTS 7](#_Toc133064195)

[6. Handling Errors 8](#_Toc133064196)

[6.1 Login error Handling 8](#_Toc133064197)

[6.2 REGISTRATION ERROR HANDLING 8](#_Toc133064198)

[6.3 Data Type Error Handling 9](#_Toc133064199)

[6.4 DATABASE ERROR HANDLING 10](#_Toc133064200)

[7. Denying SQL INJECTION 10](#_Toc133064201)

[8. READING FILES 10](#_Toc133064202)

[10. LOGGING INCIDENTS 10](#_Toc133064203)

[11. AVOIDING BOTS 11](#_Toc133064204)

[12. FORGOTTEN PASSWORD 11](#_Toc133064205)

[13. EMAIL VALIDATION 12](#_Toc133064206)

[14. DATA LIMITATIONS 12](#_Toc133064207)

[15. FIGURES 12](#_Toc133064208)

# 1. Synopsis

This document is a complete overview of the Web Routing Auctioning System’s security measures and what is implemented to improve user experience and application security.

# 2. Roles

The Web Routing Auctioning System implements Role-Based Access Controls within the entirety of the application. This is to ensure that security is persisted throughout all aspects of the application. These mechanisms of security ensure that Users must be assigned roles to access different parts of the application and depending on the assigned roles defines what users can and cannot access.

# 3.Access Limitations

This section will highlight the implementation of security mechanisms to deny access to users based on their roles and how this is persisted in other aspects of the application as well as how this denies elevation of privilege.

## 3.1 PErmissions based denials

In the Web Routing Auctioning System there are four different roles implemented to fit different needs of the user, and the security of the application.

**Admin:** The Admin role’s identifier in the program and in the database is “**ADMIN**”. Admins can create new users and can change any user’s account information except the password. They can enable and disable any user account as well as enable and disable a user’s access to the auction.

**Shipper:** The Shippers role’s identifier in the program and in the database is “**SHIPPER**”. Accounts with a Shipper role have access to the shipments page. Here they can create a shipment and then push it to auction or directly assign it to a Carrier. If a shipment is in auction, the Shipper can accept bids from bidders.

**Carrier:** The Carrier role is identified as “**CARRIER**” and has high functionality in the system. Carriers create and therefore, have access to the following fields that are centered around the Vehicle that is to carry the shipment:

* Contacts
* Drivers
* Locations
* Maintenance
* Routes
* Shipments
* Technicians
* Vehicles
* Vehicle Types

The contacts that are added to the system can become Drivers or Technicians (or both currently) of Vehicles. The Vehicle Type needs to be added before a vehicle of that type can be added. A Vehicle resides at a Location and can be assigned Shipments. When Vehicle Maintenance orders are created, they are assigned to Technicians.

A Carrier user has access to all Shipments and Routes that are in the auction and the ones directly assigned to them.

**Auctioneer:** The Auctioneer role of the program is the **Master List**role known as the “**MASTERLIST**”.

Auctioneer role has complete control over the auction and can do the following:

* Push shipments to auction.
* Remove shipments from auction.
* Force End an auction.
* Direct Assign shipments to carriers
* Freeze and Unfreeze Shipments.
* View, edit, and delete bids.
* Edit and Delete Shipments in any status.
* View Shipment details including routes.

Auctioneer role also has the access to Simulations.

**Shadow Admin:** The Shadow Admin role’s identifier in the program and in the database is **“SHADOWADMIN”**. The Shadow Admin oversees the admin role and keeps track of the system logs for all users.

Logs include details such as:

* User logs in/logs out.
* Add, edit, or delete a user or object.
* Unsuccessful attempts to add, edit, or delete users or objects.
* Excel Upload errors
* Auction activities
* Shipment activities

The logs can be filtered by:

* User
* Start Date
* End Date
* Level – INFO and ERROR

Filtered and unfiltered logs can be exported to an Excel file.

## 3.2 persistence of permissions denials in html

In the application Role Based permissions are softly implemented in the HTML pages. All HTML pages have a simple Authority statement (Figure 1).

Diagram

Description automatically generated

Figure 1 – Activity Diagram illustrating HasAuthority method.

As shown in Figure 1 we can see that if the user’s role is not permitted to the page, then the page is hidden from their view, inversely if the user’s role is privileged to access the page, then the page will be made available to them.

This is important as it satisfies confidentiality in the program. If the user does not have a role with privilege to the page, then they do not need the option to access it. As well as this keeps malicious users from identifying possible targets to attack and manipulate the system. Figures 2 illustrates an example of how this is reflected in the system.

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 2 – Shipper and Admin Role example of the HasAuthority function

## 3.3 ENSURING DENIALS VIA WEB SECURITY CONFIGURATION

The previous sub-section implements Role Based permissions softly. This section addresses the hard implementation of role-based permissions.

In the Web Routing Auctioning system, Spring boot is used as a framework for the application. In this framework the availability of configuring security is high and application uses it to implement security by implementing the **WebSecurityConfigurerAdapter.** With this, the application is allowed to implicitly approve and deny access to users based on their roles.

The previous section talks about satisfying confidentiality. If a malicious user were to gain information on pages not accessible by their account, without the **WebSecurityConfigurerAdapter** in place**,** they would be able to possibly elevate their privilege or gain access to data they are not permitted to.

By implementing this into the application, confidentiality, availability, and integrity persists, as well as denying elevation of privileges. Figure 6 shows an example of this.

Shape

Description automatically generated

Blank Page

Figure 3 – Shipper Account Trying to Access Admin Page

# 4. Password Security

This section highlights the use of Password security implemented into the Web Routing Auctioning system and how this satisfies the security requirements of confidentiality, integrity, and availability of the application’s passwords.

## 4.1 PASSWORD HASHING

In the Web Routing Auctioning system passwords are stored in the database. When a user enters their password, the backend of the program utilizes a package called the “**BCryptPasswordEncoder*”*** which is a password hashing (Figure 4).

|  |  |
| --- | --- |
| Password | Password |
| Hashed Password | '$2a$10$EVBsfb2HGqaMlI9z443kR.zM.Tn66fT.7nbvsVhDeqAw.fc4HyXOG' |

Figure 4 – Example of BCryptPasswordEncoder

This security mechanism helps keep the user’s password confidential and keeps the integrity of the passwords as the actual password is not changed for the user. Furthermore, the only person who can access this hashed password would be the Data base administrator as no role in the system has direct access to a user’s password except for the user. The user may update their password, which will then be hashed and changed in the database, the admin role may reset a user’s password, but this will send a reset password link to the user’s email.

# 5. User entity requirements

This section addresses User entity requirements. Which is user defined inputs and requirements implemented on those inputs. This is achieved through a custom **User Validator**.

## 5.1 Username requirements

In the Web Routing Auctioning system, accounts are created and identified with a username.

The requirement of the username is as follows:

* Username **must** be unique.
* Username **must** be six characters long minimally and cannot exceed thirty-two characters.

The user is also confined to this username, and they cannot change it.

## 5.2 PASSWORD REQUIREMENTS

The Web Routing Auctioning system also implements password requirements through the **User Validator**

The requirement of the password is as follows:

* Password **must** be between eight and cannot exceed thirty-two characters.

The user may change their password by logging in and updating it, or by following the steps of forgetting their password.

## 5.3 EMAIL REQUIREMENTS

In this application Email requirements are also implemented upon the user. Using the **User Validator** class, the requirements for emails are as follows:

* A user’s email must be between 3 and 64 characters long.
* A user’s email must contain an @ character.
* A user’s email cannot be the same as another user’s email.
* A user’s email must be a real email account (See Section 13)

# 6. Handling Errors

Errors should always be handled during Application run times and prevented. This section of the security manual will highlight areas where errors are handled at.

## 6.1 Login error Handling

During the login process if a user enters an incorrect username or password a message is displayed to the user indicating that a field is incorrect (Figure 5).

Graphical user interface

Description automatically generated

Figure 5 – Login Errors on Page

It is important to note that the error message does not explicitly tell you what is wrong, this is important since malicious actors do not know which one is wrong with this message and ensures that confidentiality in security is maintained.

## 6.2 REGISTRATION ERROR HANDLING

Error handling is also handled very similarly to login in the registration, this error handling is a bit more implicit as it needs to be so the user knows what field to change if the error occurs (See Figure 6).

Graphical user interface, application

Description automatically generated

Figure 6 – Registration Errors on Page

## 6.3 Data Type Error Handling

Data type errors are handled throughout the Web Routing Auctioning system. Errors will be thrown for users that try to input a datatype that the input does not take when adding a new entry manually or through Excel upload.

Dependencies exist throughout the Web Routing Auctioning system. Entities are checked throughout the application. If an entity is required due to a dependency, then the entity cannot be removed, and an error message is thrown to the user.

The system handles the following errors:

* Attempting to add a duplicate entry.
* When attempting to add an item that has dependencies that don’t exist in the system yet.
* If the Excel sheet attempting to be uploaded is incorrectly formatted.
* Attempting to delete an entry that has a dependency.

Some examples of error handling messages shown in the following figures(7, 8, 9).

Graphical user interface, text, email

Description automatically generated

Figure 7 – Error message for uploading an entry that is dependent on another entry not yet added.

A picture containing text

Description automatically generated

Figure 8 – Error message for entry already existing in the system.

Graphical user interface, text, application, email

Description automatically generated

Figure 9 – Popup alert for dependency conflict

## 6.4 DATABASE ERROR HANDLING

Pertinent to System Administration, when the application is run, default data values are attempted to be added. If the default data is already in the application, then the errors are caught, and an explanation is provided (Figure 10).

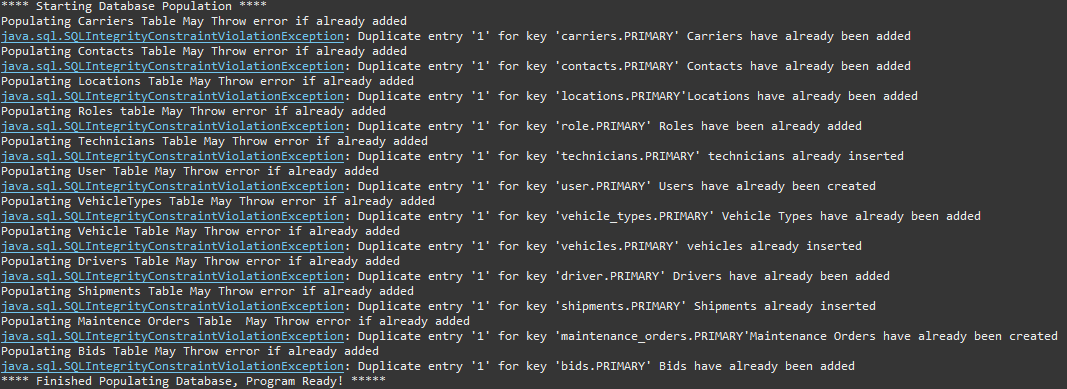


Figure 10 – Database Error Handling

# 7. Denying SQL INJECTION

SQL Injection is a major issue for web applications and can seriously impact the services offered by them. The Web Routing Auctioning system handles SQL injection attacks via implementation of the CRUD repository and Hibernate Library. This compiles into an anti-SQL injection framework that satisfies confidentiality, integrity, and availability to users.

# 8. READING FILES

Reading files is a part of the Web Routing Auctioning system solely in the Shipments Controller class. The files are being iterated through and added to the database. If errors occurred, then errors are logged for admin use.

# 10. LOGGING INCIDENTS

A logging class is implemented throughout the application where logging is pertinent to gathering information. Information needs to be logged for admin use when errors occur. This is so admins can understand errors encountered by users and attempt to mitigate these problems when encountered.

As mentioned in section 3, the Shadow Admin account can view and retrieve logs. Figure 11 shows an example of filtered logs by start data, end date, level, and user.

Table

Description automatically generated

Figure 11 – Filtered logs

# 11. AVOIDING BOTS

Bots can be a detrimental issue to online service applications. They can pollute applications and provide false information to users. The Google ReCAPTCHA is implemented during sign-up to detect and determine bots, if a bot is detected through the Google API, then their request to prove they are human is denied, and they will not be able to create an account.

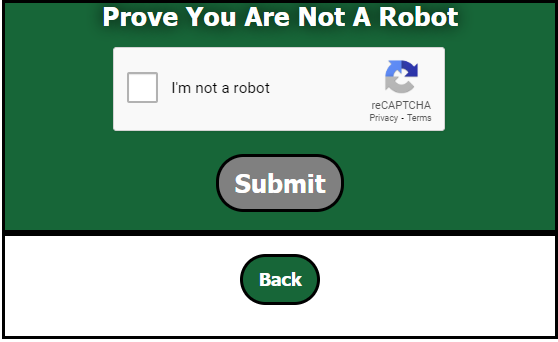


Figure 12 – Google ReCAPTCHA

# 12. FORGOTTEN PASSWORD

A forgot password is implemented into the Web Routing Auctioning system, this allows users to reset their password by following the button and entering their email address. A malicious actor could use this to try and reset another user’s password, However, an email is sent to the user in-case of this. As well as this does not deny account access. The user may change their account password or may not.

# 13. EMAIL VALIDATION

Upon account creation a user is denied access to their account until they follow through with the email verification. Once their email is verified, they are enabled in the database and gain access to the application. This enforces users to have a real email address as well as prove that it is real.

# 14. DATA LIMITATIONS

Data limitations are integrated into the application. This ensures that certain data stays in the standards of real data. These limitations are persisted into user inputs allowing for users not to by-pass data constraints in the database.

# 15. FIGURES

Figure 1 – Activity Diagram illustrating HasAuthority method 5

Figure 2 – Shipper and Admin Role example of the HasAuthority function 5

Figure 3 – Shipper Account Trying to Access Admin Page 6

Figure 4 – Example of BCryptPasswordEncoder 6

Figure 5 – Figure 5 – Login Errors on Page 8

Figure 6 – Registration errors on page. 8

Figure 7 – Error message for uploading an entry that is dependent on another entry not yet added 9

Figure 8 – Error message for entry already existing in the system. 9

Figure 9 – Popup alert for dependency conflict 9

Figure 10 – Database Error Handling 10

Figure 11 – Filtered logs 10

Figure 12 – Google ReCAPTCHA 11