# RADAR-AD User Authentication Test Procedures

# Introduction

This document presents an integration test for user authentication via JWT tokens. The client side is supposed to be a mobile application and server side is a java-based web application. The test environment utilizes the technologies below

1. Docker
2. Spring boot
3. JWT auth token
4. Javascript & JQuery
5. Curl

The server side consists of spring boot and JWT auth library and client side consists of curl and/or javascript.

**Requirements**

The following items are required to proceed the steps in this document.

1. Docker
2. Wget
3. 5gb space in the host machine.

Docker will download an image with 2GB size, so there should be enough space in the host machine.

## Set up codebase

The following command copy the GitHub repository.

|  |
| --- |
| git clone https://github.com/halilagin/rest-jwt-auth.git |

The directory container-programs should have two program installed in 1) JDK 2) MAVEN. The git repository does not contain these programs and you need to download and locate those programs in container-programs directory.

The filename **${root directory}/container-programs/jdk** should point to JDK home and **${root directory}/container-programs/mvn** home.

**Download JDK**

|  |
| --- |
| sh download-java.sh |

**Download MAVEN**

Wget program needs to be installed in the host machine.

|  |
| --- |
| wget http://www.mirrorservice.org/sites/ftp.apache.org/maven/maven-3/3.6.2/binaries/apache-maven-3.6.2-bin.tar.gz |

**Setting up a maven repository**

The docker-compose configuration file (docker-compose.yml) points to a maven repository directory. The default is **/Users/halil/.m2** in docker-compose.yml. This path should be changed accordingly so that maven could compile the codebase properly.

## Creating a docker virtual network

|  |
| --- |
| sh create-radaradnet.sh |

## Starting the Docker Container

|  |
| --- |
| sh dcup.sh |

## Connecting to the Docker Container via a terminal

|  |
| --- |
| sh dcterminal.sh |

# Running the Auth Server

|  |
| --- |
| cd /home/ruser/workdir/jwt-spring-security-demo  mvn clean install  mvn spring-boot:run |

After running the commands above the server should listen at port 8080 and docker redirect the port 8080 to 18051 for the host machine.

# Testing

There are two ways to test JWT auth mechanism 1) CURL 2) Javascript&JQuery based web page.

## Testing via Curl

Following steps shows how auth test is achieved via curl command.

**Testing the login operation**

|  |
| --- |
| cd /home/ruser/workdir/jwt-spring-security-demo/test  sh test.login.sh |

The command above sends a POST request to the server and stores the token received in token\_received.txt

**Testing the login operation**

|  |
| --- |
| cd /home/ruser/workdir/jwt-spring-security-demo/test  sh test.get\_userinfo.sh |

The command above sends a POST request to the server and stores the user information received in userinfo\_received.txt.

Each POST request should be performed with having authentication token in the headers, where the token is stored in token\_received after first request done as in the section “**Testing the login operation**

” above.

## Testing via browser

The server’s url is <http://localhost:18051>.

The port address of the server’s url is defined in docker-compose.yml. If it is different, be aware of that.

Visit the page is <http://localhost:18051>.

The auth server serves a web page who looks like the one in Figure 1.

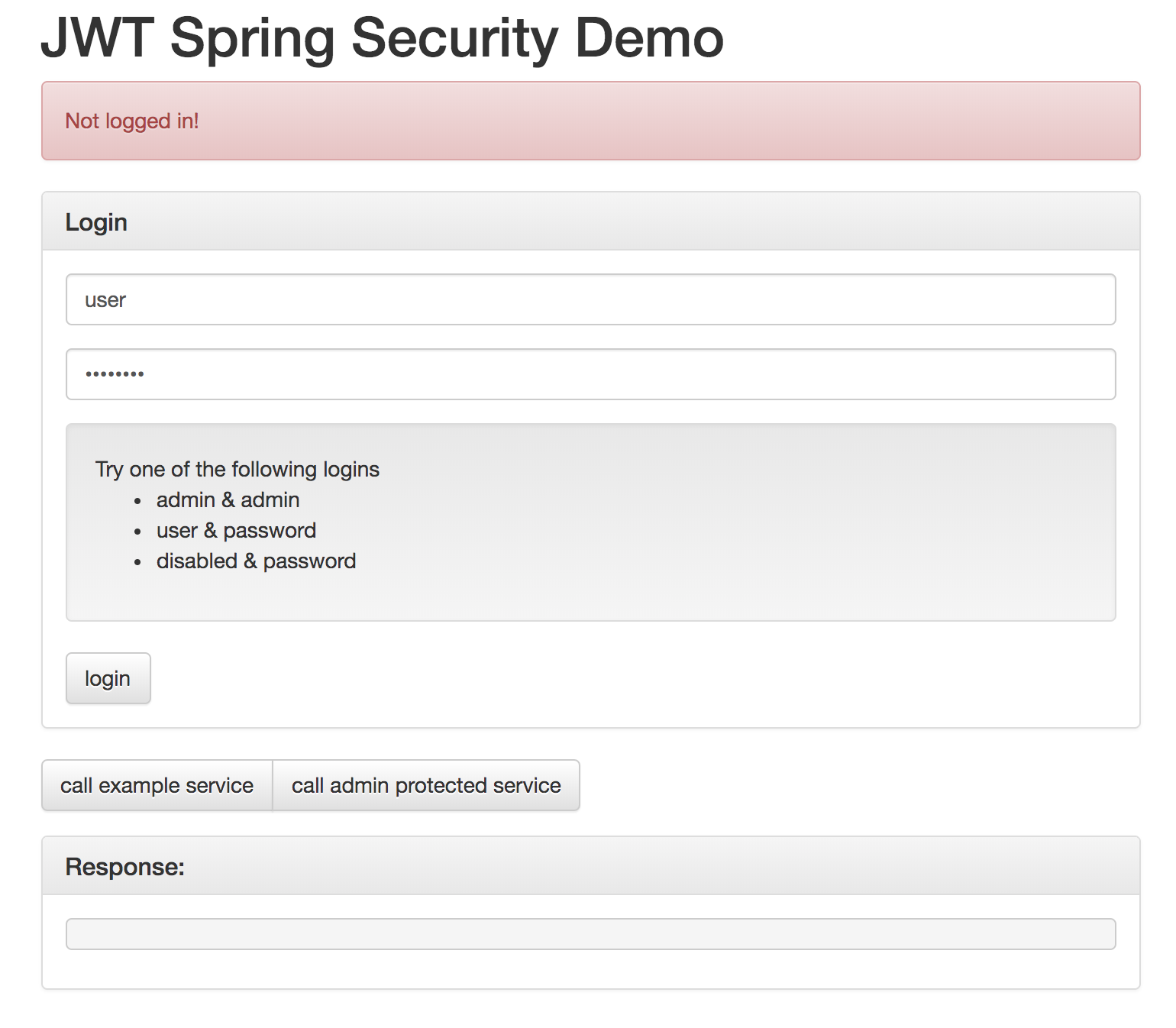


Figure 1 The home page of the auth test server

The test can be achieved by entering the credentials below.

1. Username:admin password: admin
2. Username:user password: password
3. Username:disabled password: password

The HTTP request and response can be observed in the debug screen of the browser.

**Notes**

<https://github.com/okode/ios-qr-auth>

<https://hub.docker.com/r/okode/ios-qr-auth>