Risk Client Server Application

Test case report

Ryan Silver

Version 1.0

[1 purpose](#_u46o6eh9tuvs)

[2 Requirements for Test](#_rcllwmc1exli)

[2.1 database testing](#_ng6b5f85lxiw)

[2.2 function and security testing client server](#_u2o0ccq2p49f)

[2.3 Configuration testing](#_iun9i7a9ypoq)

[3 Test Strategy](#_8lb5hv5f8g2g)

[3.1 database testing](#_sdkrpzys6brq)

[3.1.1 Verify the client server application has access to the database and all of it’s tables](#_92rj01ebu2mx)

[3.1.2 Verify correct insertion and retrieval of database data](#_5hezgi8a0lhu)

[3.2 function and security testing client server](#_il631i2clo41)

[3.2.1 The user should be able to see that game as it plays out.](#_dg8ljfmarshb)

[3.2.2 The user should be able to view replays of games using the website](#_pxwqppvt02f2)

[3.2.3 The user should be able to download source code for the client on website](#_9d4c0q86jese)

[3.2.4 The user should be able to connect over the network to the server.](#_1g20doli3fxg)

[3.2.5 The server should not lock up if the user disconnects](#_ur00p6ldfilq)

[3.2.6 The server should not accept incorrect transactions from players](#_lc3vsd787trb)

[3.2.7 The server should not be vulnerable to attacks on the database](#_xp5vpbysl0dg)

[3.3 Configuration testing](#_s3vzhvsgpp04)

[3.3.1 The server should read from the database as written in riskDbLocation.CFG](#_k2q0fasd62t0)

[4 testing process](#_trpfi2e0mkme)

[4.1 database testing](#_3wch2k2tnhmr)

[4.1.1 Verify the client server application has access to the database and all of it’s tables](#_js05e5dm158)

[4.1.2 Verify correct insertion and retrieval of database data](#_e5a31ysbeth1)

[4.2 function and security testing client server](#_bnallsgmtxd2)

[4.2.1 The user should be able to see that game as it plays out.](#_9i981f63dvm)

[4.2.2 The user should be able to view replays of games using the website](#_7wl76zluzcqq)

[4.2.3 The user should be able to download source code for the client on website](#_x356er1z7lpg)

[4.2.4 The user should be able to connect over the network to the server.](#_pm2tvpgql9vk)

[4.2.5 The server should not lock up if the user disconnects](#_jbxv3ta0uw5k)

[4.2.6 The server should not accept incorrect transactions from players](#_pw6hgt1k7h0t)

[4.2.7 The server should not be vulnerable to attacks on the database](#_z7lgeg3hd5uo)

[4.3 Configuration testing](#_lvy8frr30i9y)

[4.3.1 The server should read from the database as written in riskDbLocation.CFG](#_3fqdtzfslokr)

# 1 purpose

This document describes the plan, process and results for testing the Risk Client Server Application.

# 2 **Requirements for Test**

The section below details requirements for functional and nonfunctional requirements

## 2.1 database testing

Verify the java application has access to the database and all of it’s tables

Verify correct insertion and retrieval of database data

## 2.2 function and security testing client server

The user should be able to see that game as it plays out

The user should be able to view replays of games using the website

The user should be able to download source code for the client on website

The user should be able to connect over the network to the server

The server should not lock up if the user disconnects

The server should not accept incorrect transactions from players

The server should not be vulnerable to attacks on the database

## 2.3 Configuration testing

The server should read from the database as written in riskDbLocation.CFG

# 3 Test Strategy

## 3.1 database testing

##### 3.1.1 Verify the client server application has access to the database and all of it’s tables

Access database with DBConn class to verify connection to database

##### 3.1.2 Verify correct insertion and retrieval of database data

Access database with DBConn class to insert into all tables. Retrieve data to verify integrity.

## 3.2 function and security testing client server

##### 3.2.1 The user should be able to see that game as it plays out.

Start a game with Client UI on and check to see if it updates properly

##### 3.2.2 The user should be able to view replays of games using the website

Start a game and let it finish, after completion check the website to see if the new new game number appears and watch the game to verify consistency.

##### 3.2.3 The user should be able to download source code for the client on website

Access the tutorials page of the website and attempt to download the source code

##### 3.2.4 The user should be able to connect over the network to the server.

Start the server on a separate computer and verify that the client has access to it by connecting.

##### 3.2.5 The server should not lock up if the user disconnects

Download a version of the source code for the client and edit it so that it disconnects from the server at a certain point in the transaction. Do this for all transactions. Verify that the server has not hard locked by reconnecting to it in a regular client.

##### 3.2.6 The server should not accept incorrect transactions from players

Download a version of the source code for the client and edit it so that it

Incorrectly sends transactions with the wrong shortnames or wrong count of armies. Verify that the server disconnects after these unsafe transactions

##### 3.2.7 The server should not be vulnerable to attacks on the database

Download a version of the source code for the client and edit it so that it sends an sql injection string to the database via the username. Verify that the database is not breached in this attack.

## 3.3 Configuration testing

##### 3.3.1 The server should read from the database as written in riskDbLocation.CFG

Write a new path to the database into the CFG file and check that it accessed that data

# 4 testing process

## 4.1 database testing

##### 4.1.1 Verify the client server application has access to the database and all of it’s tables

Access database with DBConn class to verify connection to database

Steps:

1. Start main method in DBConn class

2. Wait for completion of main method

3. After completion access the database through the console to verify data is inserted

##### 4.1.2 Verify correct insertion and retrieval of database data

Access database with DBConn class to insert into all tables. Retrieve data to verify integrity.

Steps:

1. Start main method in DBConn class

2. Wait for completion of main method

3. After completion access the database through the console to verify data is inserted

## 4.2 function and security testing client server

##### 4.2.1 The user should be able to see that game as it plays out.

Start a game with Client UI on and check to see if it updates properly

Steps:

* Open an instance of the server and the client both in display mode
* Connect to the server.
* Watch the game to ensure the client’s board matches that of the server.

##### 4.2.2 The user should be able to view replays of games using the website

Start a game and let it finish, after completion check the website to see if the new new game number appears and watch the game to verify consistency.

Steps:

* Start a game and let it finish same as 4.2.1
* Connect to the website and login to user mentioned in code of client
* View game and verify consistency

##### 4.2.3 The user should be able to download source code for the client on website

Access the tutorials page of the website and attempt to download the source code

Steps:

* Access website
* Navigate to tutorials page
* Click download hotlink for source code

##### 4.2.4 The user should be able to connect over the network to the server.

Start the server on a separate computer and verify that the client has access to it by connecting.

Steps:

* Start the server on one computer
* Start the client on another computer outside of the subnet
* Verify connection between the 2 computers

##### 4.2.5 The server should not lock up if the user disconnects

Download a version of the source code for the client and edit it so that it disconnects from the server at a certain point in the transaction. Do this for all transactions. Verify that the server has not hard locked by reconnecting to it in a regular client.

Steps:

* Edit the source code of a copy of client so that it disconnects from the server mid transaction for all transactions
* Connect each separate version to the server one at a time.
* Verify after connection is cut that the server’s socket has timed out.
* Attempt to connect with a regular random client.
* Test is passed if for all of the modified versions a regular client can run after the disconnect is performed on the same port

##### 4.2.6 The server should not accept incorrect transactions from players

Download a version of the source code for the client and edit it so that it

Incorrectly sends transactions with the wrong shortnames or wrong count of armies. Verify that the server disconnects after these unsafe transactions

Steps:

* Edit the source code of a copy of client so that it replies incorrectly to transactions.
* Verify that the server disconnects from the client when these bad transactions occur.
* Verify that the server is still accessible after the bad transaction on the same port

##### 4.2.7 The server should not be vulnerable to attacks on the database

Download a version of the source code for the client and edit it so that it sends an sql injection string to the database via the username. Verify that the database is not breached in this attack.

Steps:

* Edit the source code of a copy of client so that it sends an SQL injection delete to the database copy.
* Verify that the SQL injection did not occur by checking database state.

## 4.3 Configuration testing

##### 4.3.1 The server should read from the database as written in riskDbLocation.CFG

Write a new path to the database into the CFG file and check that it accessed that data

Steps:

* Start DBConn’s main with a new database referred to in the riskDbLocation.cfg file
* Verify the database’s creation upon execution of the main method