|  |  |  |  |
| --- | --- | --- | --- |
| Server | | | |
| Inputs: | Random characters without number |  |
| Expected output: | “Unavailable Port number” message |
| Actual output: | “Unavailable Port number” message |

White boxing test:

Exception Type Testing

1. Number Format Exception

|  |  |
| --- | --- |
| Client (servers.num) | |
| Config File |  |
| Inputs: | Random characters without number as the value |
| Expected Output: | Ignore the invalid value and access the largest valid value |
| Actual Output: | The same as the expected output |

|  |  |
| --- | --- |
| Client(server.port) | |
| Config File |  |
| Inputs: | Random characters as the port |
| Expected Output: | “Server <index> has an unavailable port” message |
| Actual Output: | “Server <index> has an unavailable port” message |

1. Unknown Host Exception

|  |  |
| --- | --- |
| Client (server.host) | |
| Config. File |  |
| Input: | A series of invalid format hostname. |
| Expected Output: | Set the invalid server as null and access the valid server |
| Actual Output: | The whole server information become null when keep verifying invalid hostname |
| Reason: | The whole data has been reset after the newest data size is set up  Therefore, the output is still the expected since the invalid hostname would not be allowed to add into the server data list |

1. Illegal Argument Exception

|  |  |  |
| --- | --- | --- |
| Server (port out of range) | | |
| Input: | Exceeded range of port number |  |
| Expected Output: | “Port Number Out of Range” message |
| Actual Output: | “Port Number Out of Range” message |

1. (Socket Time Out Exception)/ Socket Exception

|  |  |
| --- | --- |
| Client (Communication to the server with socket)  Connection + send message + receive response | |
| Input: | 1. The connection is more than 2 seconds 2. The sending message is more than 2 seconds 3. the receiving message is more than s seconds |
| Expected Output: | “Server is not available” message |
| Actual Output: | “Server is not available” message |

1. Interrupted Exception

|  |  |
| --- | --- |
| Client (cast) | |
| Input: | Interrupt threads by calling “thread.interrupt()” |
| Expected Output: | “Unavailable thread” message |
| Actual Output: | “Unavailable thread” message |

1. FileNotFound Exception / and IOException

|  |  |  |
| --- | --- | --- |
| Client | | |
| Inputs: | input without any config. File  input Non-exist File |  |
| Expected Output: | “file unavailable” message and then quit the client |
| Actual Output: | No file inputed would be the same output as the expected, but input non-exist file would shows “file unavailable” message and continue the client |
| Reason: | Since the file does not exist, the client system just initialize a new empty server information list for user to store the server data in. Therefore, the Output is totally fine since we can run the system perfectly |

1. Socket Refuse Connection Exception/ Socket Connection reset Exception:

|  |  |  |
| --- | --- | --- |
| Server | | |
| Inputs: | Running the client and send request to a non available server |  |
| Expected Output: | “Server is not available” message |
| Actual Output: | “Server is not available” message |
| Reason: | The Server is not listening to the socket connection since the server is not available, or the firewall blocking the socket connection |

Grey boxing test:

Server/Client attacked by other users

Server:

|  |  |
| --- | --- |
| Sending 10 thousand “tx” tansaction commands to attack the server system | |
| Inputs: | 10 thousand “tx” transaction commands |
| Expected Output: | Server strongly handles the 10 thousand commands and still run and print out the result perfectly |
| Actual Output: | The same as the expected output |

|  |  |
| --- | --- |
| Sending the package, which contains 10 million random command data to attack the server | |
| Inputs: | The package contains 10 million random command data |
| Expected Output: | “Scoket Connection reset” message  Attacker’s client system would be forced quit |
| Actual Output: | The server system refuse to connect with the hacker’s client and force hacker to quit the client system. |

|  |  |
| --- | --- |
| Continuously sending 10 thousand “tx” tansaction and “pb” commands to attack the server system  (DOS attack) | |
| Inputs: | The 10 thousand “tx” transaction and “pb” commands |
| Expected Output: | The server would response all of the “pb” and “tx” requests and send the responses back to client immediately |
| Actual Output: | The server is freezing once client send 10 thousands pb and tx requests concurrently. |
| Reason: | In the java socket connection, the maximum queue length for incoming connection indications is set to 50. Therefore, the threads can not handle more than 50 concurrent clients. Once the connection indication arrives if the queue length is too full, the server system would refuse the connection or taking too long to produce responses for the large amount of requests. |
| Solutions: | There is no right solution for preventing DOS attack since there might be more than 10 million clients connect to the same server. The server would directly cause extremely slow response or just directly freezing in the worst case. The only way to prevent from DOS attack is to make server allow to handle huge amount of requests concurrently |

Client:

|  |  |
| --- | --- |
| Initializing a huge duplicated server data to attack the client | |
| Input: | Huge amount of duplicated server information |
| Expected Output: | Initialize the config. File in the normal flow |
| Actual Output: | The system takes longer to initialize the config. data |
| Reason: | The System needs to verify each of the server data line by line. Since we input a huge duplicated server information into the system, the system would take longer time to initialize the data, but it would still working in the normal flow. |

The attacking config. File:

This part has been duplicated 10 times: The last part of the data :



+

Black boxing test:

Random Users testing:

User A:

|  |  |  |
| --- | --- | --- |
| Server data verification in config. file in BlockchainClient.java | | |
| Inputs: | 1. Unknown hostname (not in any format) 2. Duplicated the server data. | duplicated more than 10 times  (The config. File created by UserA) |
| Expected Output: | 1. Catch “unknownhostname” Exception and set the value as null at the server key. 2. Verify all of the data sets and successfully store the valid server information |
| Actual Output: | The same as the expected output |
| Reason: | Catching the exception by using InetAddress.getbyName(“hostname”) to check the format of ipv4, ipv6 and IP address. |

User B:

|  |  |  |
| --- | --- | --- |
| Update the server value at the key which is out of the range of the server list.  (In the BlockchainClient.java) | | |
| Inputs: | Update the server data at the key which is out of range of the server list  Example: list.size() = 3, but do “up|10|localhost|5444” | (The command is tested by User B) |
| Expected Output: | Ignore the requested update value and shows “update out of range” message |
| Actual Output: | The same as the expected output |
| Reason: | Since the server data set list’s size has been limited, the system would catch “Index out of Bound” exception once add the value at the out-of-range index |