3.1 Test Coverage

3.1.1 Tested Items

List all tested items, along with the test cases that were applied on this item. For each test item, explain why it was necessary to test it. For instance, all features listed as requirements for each build is a mandatory test item. In addition, identify at least five units (i.e. classes/methods) and explain why they require unit testing due to their importance in the implementation through their frequency of use and/or the severity of the impact of their misbehavior. You can categorize your tested items, e.g. “Requirements”, “Units”, etc.

Unit Testing:

Requirements Testing:

Stress Testing:

Stress testing is used to test an application’s performance under extreme conditions of usage, such as a high volume of users or a large data load. It is done in order to determine the system’s robustness and reliability, and helps to identify issues that may only arise under stressful conditions.

There are a number of tools available for stress testing a web application that simulate extreme conditions. Here, we used Apache JMeter, a Java application that works by creating multiple threads that simultaneously access a given web server via HTTP requests. We tested stress conditions for both the login and schedule generation functions of the system.

Security Testing:

Security testing is crucial for any software system to ensure that the system can protect itself against malicious attacks. We used two tools to test security. The first was SQL Inject Me, a Firefox plugin that sends SQL injection strings through the forms of a webpage and checks for any database errors that may occur as a result. The second was Nikto, a web server scanner that uses a Perl script to perform multiple tests against a given server, including checking for dangerous files and checking for outdated versions.

3.1.2 Untested Items of Interest

List all untested items that you find would necessitate testing. Explain how it could be tested, and why it would be important to test.

3.2 Test Cases

3.2.1 Unit Testing

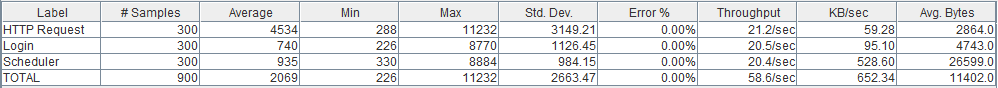
In your system, identify 2 substantive, mid-level testable units (classes, modules or subsystems)

For each of these two units, include a list of test cases and the code for the stubs and drivers used. Show the results of the testing. Where possible, a test suite should be used.. Explain what techniques were used to derive these tests.

3.2.3 Stress Testing

Describe potential extreme situations of system usage. Describe the design of tests that would verify system performance under these extreme conditions. Perform the tests and show the results.

We ran the JMeter tool with 300 simultaneous users and a 5-second ramp-up period (time to get all threads started). There were zero errors, and we obtained the following summary report:



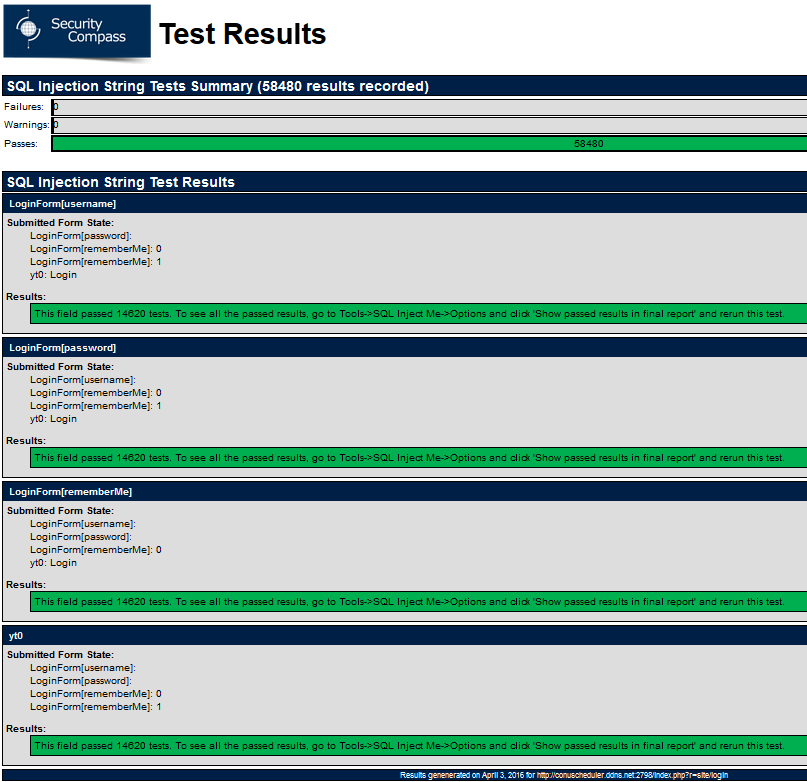
3.2.4 Security Testing

*Your project should be tested for security, Including resistance to SQL injection attacks, against at least two automated tools (e.g Nikko and Skipfish).*

SQL Inject Me results:

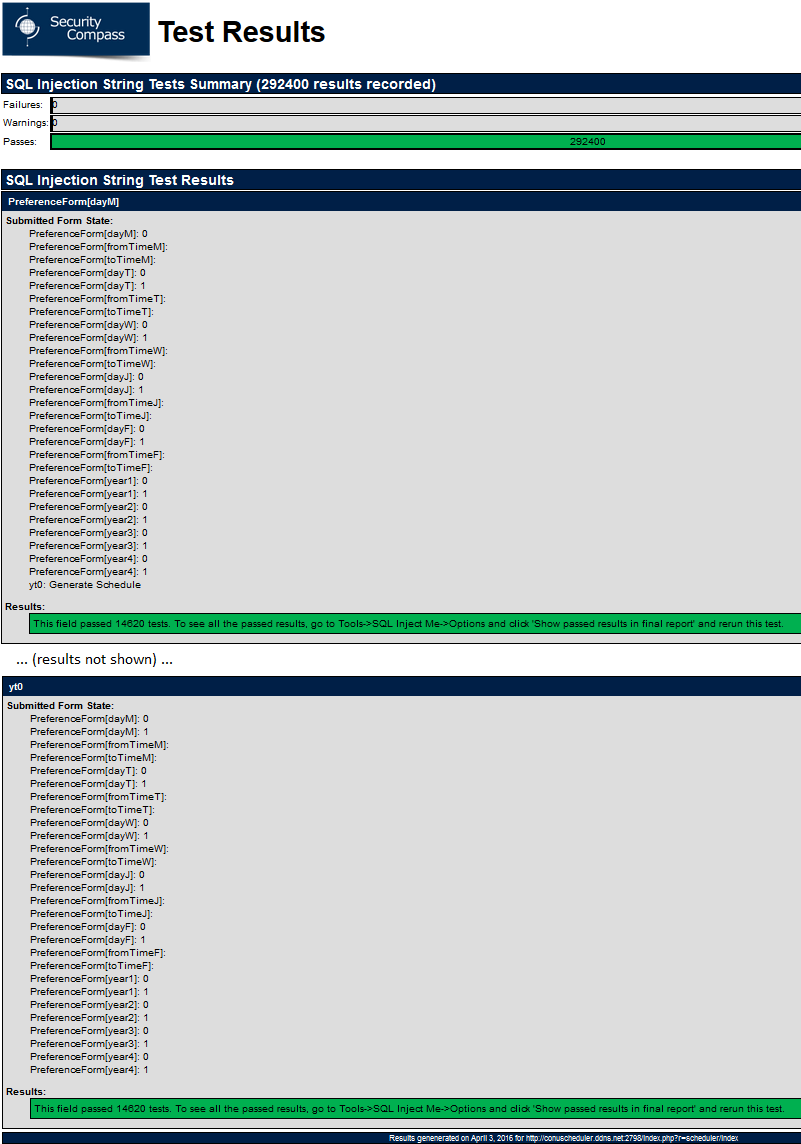
LoginForm:

The LoginForm passed all 58480 tests, shown in the screenshot below:



PreferenceForm:

The PreferenceForm passed all 292400 tests, shown in the screenshot below:



Nikto results:

