# **Test Plan for broken-hashserve**

Prepared by Tom Foster

## INTRODUCTION

The purpose of this document is to provide the blueprint for testing the broken-hashserve application. The application provides three endpoints to the user that allow a given password to be hashed, retrieved, and provide basic server stats.

## SCOPE

### In Scope

The scope of this testing will be to verify the functional specification as laid out in the QA Assignment document. The primary goals will be the following:

* Validate the new endpoints for correct functionality and protection against incorrect usage
* Validate the startup and port utilization
* Validate the proper hashing of a password request
* Validate the proper shutdown of the application utilizing the shutdown call
* Validate the handling of multiple connections

### Out of Scope

The following items will be considered out of scope as their specifications have not yet been defined:

* HTTPS – The application currently only supports HTTP.
* Stability and Performance testing will be incorporated in a later sprint when this functionality is integrated into its final product.

## SUPPORTED PLATFORMS

This application has been designed to work on the following operating systems:

* Ubuntu 16.04
* Mac OS X – Sierra, High Sierra
* Windows 10

All functionality should be verified on each platform.

## ASSUMPTIONS

The following assumptions have been made:

* The application should accept passwords of any length with a basic character set (A-Z, 0-9, special characters). No testing of foreign character sets has been planned.
* Endpoints should be protected against unplanned usage. For example, a POST should not be allowed when only a GET is expected.

## RISKS

The following risks and mitigations have been identified for this new feature:

* QA is currently unable to test on MAC OS X due to hardware and licensing issues. This could be mitigated if the company were to provide the proper equipment for this test.
* Performance requirements have yet to be defined, so performance tests will not be performed. Once these specifications are in place, proper testing can be scheduled. Functional testing will include basic load testing to verify that the application will process multiple connections simultaneously.
* Stability requirements have yet to be defined, so stability tests will not be performed. Once these specifications are in place, proper testing can be scheduled.
* Security testing will be performed once HTTPS has been implemented in the future.

## TEST ENVIRONMENT AND TOOLS

The following environments have been prepared for testing version 0c3d817 of this application:

* Windows 10 Home with an OS build of 17134.706
* Ubuntu 16.04.6 running in Oracle Virtual Box version 6.06

Please note that QA is currently waiting on a MAC OS X system and as identified in the risks, the application will not be verified on that platform.

For both environments, CURL will be used to send requests to the application endpoints via native terminal clients within each operating system. Shell scripts will be written to speed up the process of sending multiple requests for each endpoint and to assist in providing timing of when messages were sent.

jMeter will be utilized for simple load testing and to provide a random series of requests to the application to simulate a potential workload.

## TEST APPROACH

This application will be tested with a series of manual test cases due to strict timelines imposed by this documents author and due to a relatively low need of having to execute these again during the length of this project.

I have broken down testing for this application into the following categories: startup, POST hash, GET Hash, GET Stats, shutdown, and load. I will cover each strategy individually.

#### Startup

The proper startup of the application will be verified to ensure that it launches correctly and utilized the PORT environment variable correctly.

#### POST Hash

Foremost, the basic functionality should be checked. Properly formatted messages will be sent to the server and a job identifier will be expected immediately, while the hash itself should be computed 5 seconds later.

Once the basic functionality is verified, various keyword:value inputs will be checked to ensure that application can handle things like special characters, case, numbers, etc.

Requests will be posted from various sessions to verify that multiple requests are handled simultaneously.

#### GET Hash

After a password request has been posted, we can test that the application will allow us to fetch the base64 encoded password. For basic verification, a POST to /hash will be sent and then that specific job will be retrieved via the GET from /hash.

The endpoint will also be checked to verify functionality if an unknown job number was sent to the application.

#### GET Stats

The stats endpoint will be checked to ensure it is functioning properly. The application will be started, and a known quantity of requests will be sent in and compared to the output.

The endpoint will also be checked to ensure that it is only active for GET requests and that POST, PUT, etc. are blocked.

#### Hash Validation

The correct hashing and encoding of the passwords will be verified to ensure that the algorithms are correct in the application. A POST message will be sent and then a GET on that job will be performed. The messaged received will then be used to verify that the hashing algorithm worked correctly and did in fact return the correct base64 encoded SHA512 hash. A known hash generator will be used to check this.

#### Shutdown

Graceful shutdown of the application will be verified by sending various requests to the server before and after a shutdown was ordered. These requests will be sent from various sessions to ensure that multiple connections are handled properly during shutdown. Any requests that were actively executing will be allowed to complete, while new requests should be denied.

#### Load Testing

Basic traffic will be generated using jMeter to ensure that the system can handle multiple requests from various user sessions and that the application doesn’t crash in this manner. Specifications for expected user load and performance have not been defined yet, so basic load testing is all that will be executed for now.

## RESOURCES and ESTIMATES

Our entire QA department will drop all that they are doing and jump on this immediately. The test plan should be executed over the period of 2 afternoons.

## DELIVERABLES

The following items will be provided upon execution of this plan:

* Test cases written in everyone’s favorite test case management software; Microsoft Excel!
* Test case results matrix
* Any defects will be logged in our defect tracking system