Test Cases with Results

1. Confirm waiting connection
2. Run ./broken\_hashserve\_linux and view the shell to confirm that it is waiting for a connection

* **Result = Pass**

1. Execute command:

curl -X POST -H "application/json" -d

'{"password":"angrymonkey"}' <http://127.0.0.1:8088/hash>” > 1

Confirm that it accepted connection and return value by confirming that 1 is in the directory

* **Result = Pass**

1. Confirm it answers to PORT specified in the PORT environmental variable

Execute command:

$ export PORT=8088

$./broken\_hashserve\_linux

$curl -X POST -H "application/json" -d

'{"password":"angrymonkey"}' <http://127.0.0.1:8088/hash>” > 2

Confirm that it returned a value by validating that 2 is in the directory

* **Result = Pass**

1. Confirm it answers to PORT specified in the PORT environmental variable

Execute command:

$ export PORT=7388

$./broken\_hashserve\_linux

$curl -X POST -H "application/json" -d

'{"password":"angrymonkey"}' <http://127.0.0.1:8088/hash>” > 2

Confirm that the commands did not execute

* **Result = Pass**

1. Confirm it does not answer to other ports:

Execute command:

curl -X POST -H "application/json" -d

'{"password":"angrymonkey"}' <http://127.0.0.1:8085/hash>” > 3

Confirm that it does not respond to an unsupported port by confirming 3 is not int the directory

* **Result = Pass**

1. Post to /hash accepts “password”

Execute command:

curl -X POST -H "application/json" -d

'{"password":"angrymonkey"}' <http://127.0.0.1:8085/hash>” > 4

Confirm no error is reported

* **Result = Pass**

Confirm password is accepted by validating file checking that there is a value in stored location

Execute command:

curl -H "application/json"

http://127.0.0.1:8088/hash/4

* **Result = Pass**

1. Post to /hash does not accept anything but “password”

Execute command:

curl -X POST -H "application/json" -d

'{"other":"nomonkey"}' <http://127.0.0.1:8085/hash>” > 5

* **Result = Fail – Accepted “other” and returned a value**

1. Post to /hash returns job identifier immediately

Execute command:

curl -X POST -H "application/json" -d '{"password":"angrymonkey"}' <http://127.0.0.1:8085/hash>” > 6

* **Result = Fail –** **Fail did not return job identifier**

1. Post to /hash waits 5 seconds to compute hash

Execute command:

curl -X POST -H "application/json" -d '{"password":"angrymonkey"}' <http://127.0.0.1:8085/hash>” > 7

* **Result = Pass**

1. Post to /hash algorithm is SHA512

Execute command:

curl -H "application/json" <http://127.0.0.1:8088/hash/7>

Compare that to the value from the output of running:

$SHA512 angrymonkey

* **Result = Fail, values do not match, does not return password in SHA512 checksum**

1. Get to /hash accepts job identifier

Execute:

Get <http://127.0.0.1:8088/hash/1>,

* **Pass**

1. Get to /hash returns base64 encoded password for the corresponding POST request

Execute script to compare base64 result to hash result:

./test3.sh

Expected Results = equal

* **Result = Fail, returned ‘not equal’. Get to /hash not returning base64 encoded password.**

1. Get /Stats should not accept data

Execute command:

curl -X POST -H "application/json" -d '{"password":"angrymonkey"}' <http://127.0.0.1:8085/stats>”

* **Result = Fail. Command is accepted and no error message is reported. Entering this type of command should be blocked and return error message.**

1. Get /Stats should return a JSON data structure for the total hash requests since the server started

Execute:

curl <http://127.0.0.1:8088/stats>

* **Result = Pass**

1. Get /Stats should return the average time of a hash request in milliseconds

Execute script to exercise program:

./test1.sh

Run Command:

curl http://127.0.0.1:8088/stats

* **Fail, does not return in the correct time format**

1. Confirm that the software should be able to process multiple connections simultaneously.

Execute script to run multiple commands at once:

./test4.sh

* **Result = Pass**

1. Graceful shutdown request should allow any in-flight password hashing to complete

Execute script to run multiple password hashing, shutdown command and then some more password hasing:

./test5.sh

* **Result = Pass**

1. Graceful shutdown request rejects any new requests

Execute script to run multiple password hashing, shutdown command and then some more password hasing:

./test5.sh

* **Result = Pass**

1. Graceful shutdown request responds with a *200 –*

Execute:

curl -X POST -d ‘shutdown’ *http://127.0.0.1:8088/hash*

* **Fail, does not return 200**

1. Graceful shutdown request does shutdown

Execute:

curl -X POST -d ‘shutdown’ *http://127.0.0.1:8088/hash*

* Result - Pass

1. No additional password requests should be allowed when shutdown is pending

Execute:

curl -X POST -d ‘shutdown’ *http://127.0.0.1:8088/hash*

* Result - Pass