Chapter 3

Architectural Framework Description

Test Template

We choose a template that is readable to the terminal but is also easily read by a person for easy understanding. It includes the component and function that is being tested, a description of what is being tested, the name of the driver, the arguments that must be passed to the driver, and the expected outcome.

Compiling tests

Since the components that we testing have a lot of dependencies and also need some environment variables that are set during compilation, we decided to compile our drivers in the general git compilation. It adds a huge overhead for the first time that you compile the project (around 2 minutes). However, since it is compiled using a makefile, after the first compilation, only the files that have been changed are recompiled and linked with the others. Furthermore, we think that it is the most convenient approach since it is how native tests are compiled on git.

Test Script

The script copies all drivers from /testCasesExecutables to the project folder (project/git) and compiles all the project. As pointed above, it takes around 2 minutes for the first compilation, but in later executions it will compile only the files that have changed. After compile the project, the script takes all test cases. For it test case, it identifies the driver name, the arguments, and run the driver. Then, the output is stored in a variable and compared with the expected result to define wether the test passed or not.

How To Run Script

First, all libraries that are used by git should be installed:

- sudo apt-get install libcurl4-openssl-dev zlib1g-dev openssl libexpat1-dev libiconv*

After install the libraries, go to the top-level folder of the test framework and run:

- ./scripts/runAllScripts.sh

Test Cases

testCase1

id "001"
module "credential.c"
function "credential_match"
it "compares the protocol"
driver "test-credentialMatchDriver"
arguments "https example.com foo.git bob http example.com foo.git bob"
expected "1"
end

testCase2

id "002"
module "credential.c"
function "credential_match"
it "compares the host"
driver "test-credentialMatchDriver"
arguments "https example.com foo.git bob https other Example.com foo.git bob"
expected "0"
end

testCase3

id "003"
module "credential.c"
function "credential_match"
it "compares the path"
driver "test-credentialMatchDriver"
arguments "https example.com foo.git bob https example.com foo.git bob"
expected "0"
end

testCase4

id "004"
module "credential.c"
function "credential_match"
it "compares the username"
driver "test-credentialMatchDriver"
arguments "https example.com foo.git bob https example.co foo.git mary"
expected "0"
end

testCase5

```
id "005"
module "credential.c"
function "credential_match"
it "returns true when every credential field matches"
driver "test-credentialMatchDriver"
arguments "https example.com foo.git bob https example.com foo.git bob"
expected "1"
end
```

Experience:

In completing deliverable 3 we had more experience working together as a team. We started off by defining what we needed to get done and separating the work amongst ourselves. Our divide and conquer approach helped knock out one task after another in a nice pace. When a team member got stuck or had issues with code they could get help when we meet together or by using slack. Slack is messaging app that let us communicate between the three of us. By having team meetings it makes it easy to get help with making the different sections work together while keeping us on the same page. Starting was hard because it was hard to figure out how we wanted to go about setting up the drivers and how to compile them with low overhead.