|  |
| --- |
| Peer to peer systems and security project |
| Initial Approach Report |
| Report#1 |
| Group08, Assigned to Testing |
| **Makan Tayebi Gholamzadeh – Dakota Biggs** |
| **5/16/2015** |

# Participants’ Background

Makan

Programming Languages: Java, C, Backend & Frontend web development… (Open to new ones)

Project Management & Design: BPMN, UML, MS Project

Tools: Linux bash, Windows command line, WireShark…

Dakota

Programming Languages: Java, C#

Tools: Windows command line, office, …

# Tools

We already both are experienced Windows users and so far, no exclusive need of a linux System has been seen. Software like Mininet also are supported with Windows versions, and if an absolute requirement of a linux is seen in the future, it is not necessary to move a lot around; An installation of a Virtual Machine would do. So there is no need to put up with the effort of installing Linux right now. As for the programming language, we believe Java would deliver every means needed for building network related desktop applications. Java is a language that you use, when you are not struggling to enhance performance bit by bit. It is language for labs and testing applications. simply because of two important factors: automatic memory management (Garbage collector), and the fact that it supports concurrent programming, which is an absolute necessity when dealing with listening to ports. Of course there are multiple options that are comparable to Java from these angles, but the common expertise of both participants has a major effect. Java is an imperative Turing Complete language with an impressive set of libraries almost in any field that comes to mind; from parser generators like Antlr, to file handlers, and stream handlers. That aside, the duties of this test application are clear. It is to listen on ports, record, edit and manage messages and deliver them. Java has Serializers to save and load objects, Streams to deal with files, it has built in methods to listen on ports, and so on… Hence, to the best of our knowledge, there is not a Java library that we will need! Every task that we have looked upon, can be done easily using Java. In terms of network providing, we have so far agreed on Mininet, since it seems to be convenient enough, not relying on real networks and addressed to real uncontrollable delays like Planetlab would be. It is on your laptop, so it is always available when you need it.

# Planning

We believe that the implication of using terms like Scrum, Agile, etc. could involve more paper work and make the process sluggish. This is a two person team with limited resources and experience. We think pragmatically. We would be using one of the free software available for managing the process. A simple list can be found [here](http://blog.capterra.com/free-open-source-project-management-software/). It does not matter which or none will be picked later, or this will be handled simply on a text file. What does matter for us is that we will apply the Work Breakdown Structure, schedule, and update it when necessary. A little more detailed premature WBS follows:

In our meeting, it was already modeled that we need to break the work into Proxies. Practically that is true. But we need to come up with scenarios. Each scenario may involve a number of proxies. So the following steps are anticipated:

1. Devise a scenario for testing. It should be either about Validation or Qualification.
2. Design a number of needed Proxies
3. Implement each Proxy.
4. Provide dummy test-cases for a validator proxy

More about step 4

Since there are no programs to test yet, we will come up with dummy KX, DHL, VOIP modules that create random messages. With the following Ideas:

1. They might provide a false ID for the message
2. Provide Wrong message size
3. Generally, each field should have random/valid data tested into it.
4. Modules that don’t answer at all
5. Modules that send a message multiple times
6. If we get technical into the implementation of the sockets, we think that there would be a number of rooky mistakes that needs to be covered.

## Workload

The workload distribution the following: Choosing the list of scenarios is absolutely a group work. We believe sometimes it should even be consulted with the project supervisor, since it is a delicate task. This happens in sessions. The list and description of proxies needed for the scenarios are created also in the same session. Implementing proxies however, starts out as a pair-programming task. Then, when the mutual understanding of the modules and the routine has been developed, and the Generic Pattern has evolved, then it is practical, to divide the responsibilities of each task. Reporting of the project, where involves details about proxies, the person responsible, writes it. Then there will be an overall team-review of the report for the scenario. Reports about the result of tests are done together. Any extra Documentation about proxies is the responsibility of the person related to that proxy.

# Design

If we were to show an intuition of what design of application looks like, we’d say that we need a generic proxy with configurable logic layer. The programming language (java) and the usual convention dictate an object oriented approach. However, there is always the workflow of the program to think about. Since a proxy has to correspond to messages, (events) this is obviously a loop that processes the next message: [eventloop](http://en.wikipedia.org/wiki/Event_loop).

# Quality Control

In the quality related scenarios we will put a top-down, requirement-to-parameter approach. We consider the requirements that the user expects, like bandwidth and delay, availability, etc. Then we find the related parameters that result into meeting each requirement. In terms of code quality, there is lengthy list on static code analyzers just like Cppcheck [here](http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis). Each is for a language or multi-lingual. We have to see if applying them is necessary. There are also dynamic code analyzers that we are aware of. In three cases we believe these analyzers are handy: (this can easily be updated as we have more experience)

1. Module consumes too much time
2. Module consumes too much processing power or memory
3. Module needs debugging

# License and Reusability

As the test module of the Project, is one to measure and control the quality of working modules, Hence, it is not the core of ideas and integrity of the P2P VOIP, We would like to announce that this product will be issued under MIT license. Besides we realize that if an academic project has the potential to create start-ups and Excitements, It would be nice to go along with it.

Thank you

Makan Tayebi Gholamzadeh

Dakota Biggs