Initial Report

16 Onions

Josef Stark Charlie Groh

May 19, 2017

1 General

We are the team "16 Onions" consisting of Josef Stark and Charlie Groh, and our goal is to develop a prototype implementation of the Onion module.

2 Programming Language and Operating System

Since the anonymity of a person using the VoIP application depends on the number of other users, it is important that it is as simple as possible to port the software to nearly every environment. Therefore we decided to use Java 7 as programming language as there is no need to change or even only recompile Java source code to run it on different platforms. As operating system we use Linux, since both team members are fairly familiar with it. We will not test it the module on other systems than Linux, since this is the most important one for us and the program behavior shouldn't change on other platforms running the JVM. Another advantage of Java are the built in safety mechanisms preventing common memory corruption vulnerabilities. Java is often quoted as being slower than e.g. C or C++, but we are convinced that for our project and with todays hardware, this will not be a concern; far more important would be the efficiency of our implementation in terms of e.g. network usage.

3 Build System

As development environment we will use Eclipse on a Debian GNU/Linux system. For writing the reports, LaTeX will be utilized. The exact versions will

vary among the different workstations of the team members. Eclipse does a good job at handling dependencies and incremental builds for reduced build times, so most of the time we will just rely on the IDE to do the compiling. But in order to also be able to build the project from the command line (e.g. building on a server with no graphical interface available), we will regularly generate an ant build file out of the Eclipse project. This file, which will reside in the project root folder, can then be interpreted by the command line tool ant independently of Eclipse.

We think that the Unit-Test-Framework of Java will do a good job in testing single modules for protocol-conformance. For testing API conformance we will use `https://gitlab.lrz.de/voidphone/testing`, which is provided by the course instructors. For testing the behaviour of many instances of the module the network virtualizer Mininet will be used. In comparison to more low-level languages like C, Java has improved safety quite a bit by e.g. removing the possibility of direct memory access and introducing automatic bounds checking and memory management with garbage collection. This eliminates the need to use memory checkers.
To avoid other types of bugs and ensure good quality code, we might use eCobertura, JLint and/or FindBugs for code coverage, data flow and statical code analysis.

4 Libraries

Java 7 already ships a pretty big standard library, which will probably provide most of the functionality required for implementing the Onion module. For advanced functionality we might have to use 3rd party libraries, but whenever possible it would be preferable to avoid this as it reduces portability and ease of use. The following functional requirements have been identified so far, along with libraries providing it:

- TCP communication: java.net.Socket (integrated)
- UDP communication: java.net.DatagramSocket (integrated)
- INI config file handling: May be possible with java.util.Properties (integrated), if not, we could use ini4j (3rd party) or implement it ourselves.
- Transferring data/objects: java.io.Serializable (integrated),
  X-Stream (3rd party but human-readable and more compact), Gson (same with JSON) or implementing our own protocol (more effort but probably more efficient)

`http://ecobertura.johoop.de`
`https://sourceforge.net/projects/jlinteclipse/`
`http://findbugs.sourceforge.net/`
`http://ini4j.sourceforge.net`
`https://x-stream.github.io`
`https://github.com/google/gson`

- Cryptography: java.security, javax.crypto (integrated) or Bouncy Castle Crypto APIs (3rd party, more functionality)

5 License

Our programming work will be released under the GNU Public License, because we believe it is necessary to release privacy/anonymity related applications as open source software. Firstly it should be simple for external researchers to audit the software in order to find possible security holes. Secondly users should be able to verify that there are no backdoors or information leakages. Thirdly we hope that our development can help others with their own projects, and therefore we decided to use the widespread GPL license.

Due to the same reasons we will place every documentation and report under the GNU Free Documentation License Version 1 or Creative Commons Attribution-ShareAlike 4.0 International License depending on the use case of the document.

6 Previous Programming Experience

Both team members learned Java as their first programming language at school and used it in several bigger projects. Therefore we are familiar in using internal and external libraries, sending and receiving data in various protocols by using sockets and testing using unit tests.

7 Work Sharing Strategies

We will share the programming work by dividing it in smaller pieces and weighting it with the expected workload. Then the pieces will be assigned to the team members ensuring that every member gets a similar workload. Additionally we will meet at least two times per week to discuss implementation and integration problems and wrongly weighted pieces.

The reports will be written in the same manner as the programming is done. We will try to assign every report piece to that team member who programmed the respective software piece.

8 Issues and Complains

Until now we did not discover any issues or complaints concerning the project and hope this will last.

`https://www.bouncycastle.org`