SRSC - P1

Cláudio Pereira(51717) and Diogo Cruz(57058)

Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa

1 Introduction

In this work we developed a multicast based secure messaging protocol (SMCP) and subsequent demonstration application.

Our application was built based on the teacher sample code, but due to a port to JavaFX it ended up diverging. Albeit the same methods are present, they call a different library and are split across several files. Our application is structured with an view-controller structure, as JavaFX requires.

For additional ease of development and dependency management, we used Maven. Maven is totally optional as it is only used to fetch JavaFX and make it available for the code to compile. If JavaFX is installed system-wide it becomes unneeded.

2 Build

To build the application, the Maven command `mvn clean javafx:run` automatically resolves the JavaFX dependencies. If JavaFX is installed system wide a regular `javac` build will work assuming that JavaFX is in the classpath.

The main entry point is `view.ChatApplication`.

3 JAR trustworthiness

To ensure the trustworthiness of the resulting program, we signed the JAR using a trust store. Two different JAR files are present in the repository, one of them unsigned and the other signed with a sample trust store.

The sign command is: `jarsigner -storetype JCEKS -storepass " " -keystore ./truststore P1.jar sign-cert`

To verify the signature one can issue the command: `jarsigner --verify -storetype JCEKS -storepass " " -keystore ./truststore P1.jar sign-cert`

Both commands must be issued with the jar folder as the working directory. The demo truststore password is ” ” (a space).

4 Final SMCP protocol format

SMCP messages have the following format:

$vID||sID||SMCPMsgType||SAttributes||SPayloadLen||SPayload||FastSMCheck$

Where the fields are:

vID Protocol version.
sID Session ID (the ip:port pair).
SMCPMsgType Type of SMCP message.
SAttributes Endpoint security parameters for validation with the format:

$$sID||sName||Cipher||Mode||Padding||IntHash||FastHMAC$$

Each attribute is the *SHA-256* of the attribute string representation.
sID
sName The plaintext name of the chat session.
Cipher The cipher algorithm (eg. AES, DES, ...)
Mode The cipher mode (eg. ECB, CBS, ...).
Padding The employed padding scheme (eg. PKCS#5)
IntHash The integrity hash algorithm, used in *IntegrityCheck*.
FastHMAC The authenticity HMAC algorithm, used in *FastSMCheck*.

SPayloadLen Integer with the payload length.
SPayload Encrypted data, defined as follows:

$$Message||SeqNr||Nounce||IntegrityCheck$$

Message The plaintext message serialized.
SeqNr The sequence number of this message.
Nounce Randomly generated nouce to prevent replay attacks.
IntegrityCheck An (ideally strong) hash that ensures *Message* integrity.

FastSMCheck (Ideally performant) HMAC that allows for a fast message authenticity check, to distinguish forged messages.