End-to-end Secure Chat App

Terry Henning

February 10, 2017

1 Motivation and Problem Statement

The purpose of this project is to better understand cryptographic principles by applying them to a cross-platform chat client. The goal is to implement a secure, end-to-end encyption mechanism, including an Ephemeral Diffie-Hellman key exchange, that can possibly be extended to multiple clients. This project will enhance my ability to create secure web applications in the workforce and promote an increased awareness of client-server web security (with respect to confidentiality and integrity) and the libraries available.

2 Proposed Solution/Approach

The ideal approach is to create the web application using 1. an Amazon Web Service (AWS) LAMP server, utilizing PHP for server-side scripting, and 2. JavaScript (jQuery framework) for cross-platform (mobile and web) client-side functionality. If I recall, OpenSSL uses MAC-then-encrypt, so I will need to use a function from the library to ensure that encrypted messages are authentic. I would like to implement a Psuedorandom Function, HMAC with SHA-256, for instance, for integrity and AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode for confidentiality, especially since I'll be using **https** for the web server. Overall, I'll be following the project outlined via Beachboard.

3 Details of Implementation

3.1 Platform

This chat client will be web-based.

3.2 Programming Language

This application will include JavaScript (jQuery), PHP, and MySQL.

3.3 Components Involved

If time permits, I will implement group chat functionality, so multiple clients will be utilized, specifically browsers (i.e. Chrome, Firefox, IE).

3.4 Hardware Requirements

Any IoT device, PC, or Desktop that has a web browser installed, available, and supported by jQuery 3.0+.

3.5 Time-line

I will be implementing this project exactly as Dr. Aliasgari stated with 6 phases:

1. Phase 1: **Setup your AWS LAMP Ubuntu server**
   I plan to complete this from February 11 to February 12 2017
2. Phase 2: **A simple HTTPS Server**
   Completed from February 13 to February 14 2017

3. Phase 3: **JWT and RESTful**
   Completed from February 20 to February 24 2017

4. Phase 4: **Project requirements and design documentation**
   Completed from March 1 to March 31 2017

5. Phase 5: **Key Exchange**
   Completed from April 1 to April 8 2017

6. Phase 6: **Client side**
   Completed from April 9 to April 21 2017