

Programming Project #2

Due: Wednesday, March 10th, 2010, 11:59 pm

1 Overview

1.1 Introduction

For programming project 2, you will implement a *man-in-the-middle (MITM)* attack on SSL¹, using an SSL proxy server. You will also implement a simple (command-line) administrative interface for the proxy that will make use of password authentication.

1.2 Background

Recall that an eavesdropper on an SSL connection has little power because of the encryption being used, but if an attacker is able to trick the user into using the attacker's public key rather than the intended recipient's, this security is lost. While a real attacker would likely intercept and manipulate the network packets directly to implement this attack, you will have to make use of an SSL proxy. After a client (i.e. a web browser) is configured to make use of an SSL proxy, all client SSL requests are intercepted by the proxy and relayed to the intended remote webserver.

After an initial plaintext proxy *CONNECT* request by the client, normally the proxy just forwards the encrypted data to the server. However, instead of forwarding the initial request to the remote server, your proxy will setup its own connection with the remote server and setup a connection to the client using its own certificate. Then all traffic between the client ↔ proxy and the proxy ↔ web-server is SSL encrypted, but with different keys. This means that the proxy has access to the plaintext data sent and received by the client. Having the proxy use a single, fixed SSL server certificate is not ideal, though, because modern web browsers check the *common name (CN)* field of the certificate against the domain name of the remote server. So, to mount a more transparent MITM attack, the proxy will have to generate new server certificates on the fly, for each new client request. Web browsers will still complain **once** that the certificate is not trusted, but if the user clicks past this warning, then the attacker wins.

You will be learning :

- **keytool (command line utility)** to generate and manage keys and certificates.
- **IAIK-JCE APIs** to create and sign certificates programmatically.
- **JSSE (Java Secure Socket Extension)** to do secure networking.

¹We emphasize that this project is for educational purposes only, and should never be used outside of this class.

1.3 Requirements

We will provide you with code for a basic SSL proxy, and you will need to do the following :

- Build and use a public key infrastructure using X509 certificates.
- Modify the SSL proxy to dynamically generate new SSL server certificates, based on the domain name of the requested remote web server.
- Implement password authentication, over an SSL connection, for a simple administrative interface.
- Implement a challenge/response based user authentication scheme. (Extra Credit)

We will examine each of these features in detail below. Since we have not yet covered in the lectures all of the topics explored by this project, you may wish to start first on those aspects of the project that you can do immediately and save the other parts for later.

2 Description

2.1 Secure communication

You will be working with network sockets. The JCE provides an abstraction for secure sockets in the `javax.net.ssl` package and this relieves us from explicitly performing the key exchange, encryption and integrity of the messages transferred over these sockets.

2.2 Public Key Infrastructure

2.2.1 Offline Key Generation

The SSL proxy has a public/private key pair which is generated offline using `keytool`. The `keytool` is used to generate a *keystore* for each entity in the system. Before the system is bootstrapped, you will have to generate a public/private key pair for the SSL proxy. The public key of the proxy is self-signed.

2.2.2 Generating new server certificates

After connecting to a remote webserver, the proxy will have to create a new server certificate which has the same common name (CN) field as the remote webserver's certificate. This new certificate will then be presented to the client, for use in an SSL session. You will use classes from the LAIK library to create and sign these new server certificates.

2.3 Password Authentication

In addition to implementing the MITM attack with the proxy server, you will implement a simple remote administrative interface for the proxy server, which uses password authentication. This will allow the hacker to remotely log into the server and issue commands. In order to ensure only those users the attacker has authorized can log in, the interface will use password authentication. To connect to the proxy server, the administrative program will setup an SSL connection to the proxy

server and transmit the hacker's username, password, and command. The proxy server maintains an encrypted password file, which contains a list of authorized usernames and passwords, stored salted and hashed. When the proxy receives a log in request, it should compare the hash of the received password with the stored hash from the appropriate user, allowing the user to proceed if they match, otherwise closing the connection. Once the admin client is authenticated, the appropriate command should be executed.

You will need to implement the following commands:

- **shutdown:** shutdown the MITM proxy server
- **stats:** List how many requests were proxied

2.4 Challenge/Response Authentication (Extra Credit)

For *extra credit* you may implement a more sophisticated challenge/response authentication method along with the password authentication described above. In this the proxy server will issue a challenge to the user, which they must then answer with a response that proves their identity. Several such methods exist, and we leave it to you to decide on the precise details of the method. If you do choose to implement such a system, you should provide a brief explanation of it, and why it provides a secure authentication mechanism.

2.5 System setup

There are four main types of entities in our system: the user's web browser, the remote webserver the user is attempting to connect to, the proxy server intercepting the connection, and the administrative client to the proxy server. Once the proxy server is started and begins listening for connections, the user's web browser should be configured to use an SSL proxy with the hostname and port used by the MITMProxyServer. All SSL connections by the web browser will then be routed through the proxy server.

When the browser attempts to make an SSL connection, the proxy will parse the `CONNECT` request and make its own SSL connection to the requested server. The proxy will use the connection to obtain the remote server's certificate. The proxy will then create a forged certificate which copies the entries in the remote server's certificate (e.g. its Common Name). The proxy then signs this generated certificate with its self signed CA certificate (loaded from a keystore specified at startup). It then passes this generated certificate back to the web browser, setting up an SSL connection between itself and the browser. The proxy then passes data, which it of course sees in the clear, between the two connections.

In addition to listening for connections from web browsers, the proxy server listens for connections from the admin client on a separate port. When the admin client wants to connect to the proxy server, it opens an SSL connection to the proxy on its hostname and admin port and transmits a username and password. The proxy server then consults its password file (specified at startup and stored on disk using authenticated encryption) and authenticates the user.

3 Implementation

As with the first programming project, we have provided you with starter code. The starter code illustrates the basic socket and thread programming. See the following section for links of tutorials