CS 6238: Project 2

Secure Shared Store (S3) Report

Division of Labor

I worked on this project alone and implemented all of the project by myself.

Protocol

This project uses Protocol Buffers to define and send messages over the encrypted socket. The socket encryption is handled by Java's SSLSocket class and performs mutual authentication and encryption as long as the server runs on localhost.

There are 11 different messages that can be sent:

CheckinRequest = 1;
CheckinResponse = 2;
CheckoutRequest = 3;
CheckoutResponse = 4;
LoginRequest = 5;
LoginResponse = 6;
DelegationRequest = 7;
DeleteRequest = 9;
DeleteResponse = 10;
CloseRequest = 11;

Each of these messages is embedded in an S3Message object and transmitted to the other end. The definition of the messages can be found in `src/s3.proto`.

Threat Models

The socket is encrypted using SSLSocket with the certificates in the Java keystore. Thus, an attacker cannot eavesdrop on the conversation. Additionally, the client and server both perform an additional validation of each others' certificate against the CA.

The files on disk are stored in plaintext if the security is set to NONE. When set to ALL, both confidentiality and integrity are maintained. Confidentiality will encrypt the file using a file-specific 128-bit AES key, and store the key encrypted using the server's public key. Integrity will generate a signature object using the server's public key.

Code Analysis

FindBugs found 24 bugs overall that were not resolved. All but three exist in the non-generated code. The use of System.exit() by the client application was deemed to be erroneous; however, I expect the program to exit in the cases where the server could not be reached or the keys could not be verified. Upon running PMD, most violations were resolved and the ones remaining are exclusive to the code generated by the Protocol Buffer compiler.

Performance

Given an ideal network (i.e, localhost), the performance of the connection under SSL encryption and a fully encrypted and signed document is approximately one second. When the document's security flag is set to NONE, the duration is approximately 0.65s. Signing adds about 0.1s and encryption adds about 0.1s, individually. When combined, the combination adds additional overhead on the algorithm. Without the encryption overhead on the connection, the performance is approximately 0.4s.

The test file is the following:

```
#include <stdio.h>

void swap(int*, int*);

int main(int argc, char const *argv[])
{
      int x = 100;
        int y = 36;
          printf("%d\t%d\n", x, y);
            swap(&x, &y);
              printf("%d\t%d\n", x, y);
                return 0;
}

inline void swap(int* a, int* b) {
      *a ^= *b ^= *a ^= *b;
}
```