**Secure Shared Data Repository**

**Implementation Report**

CS6238

December 4, 2015

Project Team:

Kyle Koza – [kyle.koza@security.gatech.edu](mailto:kyle.koza@security.gatech.edu)

Anant Lummis – [jimmy.lummis@security.gatech.edu](mailto:jimmy.lummis@security.gatech.edu)

# Background and Purpose

For the second project in the Fall 2015 Secure Computer Systems course, we were assigned a project to develop a secure shared data repository.

The purpose of the secure shared data repository is to have a site for storing and sharing access to files with both confidentiality and integrity in mind.

The purpose of this paper is to document specific implementation details from our project, as well as to document the challenges we faced and any design decisions made based on those challenges.

# Implementation Details

## Key Classes

There are four main components we developed including:

* a database to securely store documents
* a server application to manage interactions with the document database
* a client application for users to interact with the server
* a certificate authority to manage key exchange between clients and the server

We elected to use Python to implement the server, the client, and the certificate authority. We also chose to use SQLite to implement the S2DR database.

To perform the confidentiality and integrity functions we used Fernet’s encryption scheme, which is an amalgamation of AES Encryption and HMAC using SHA256.

We chose to implement the following pieces of functionality slightly differently than what was specified in the assignment:

1. Security Flags – Our program records the flags set by users, but we chose to MAC and encrypt all documents regardless of the flag since there is minimal additional overhead to perform this function and it increases security overall.
2. Safe Delete – Our database stores the encryption key and the actual file blob in a single row with pragma of secure delete set. Secure delete is handled at the database layer with a row overwrite of all zeros.
3. Terminate Session – There is no terminate session function since each connection is built up and torn down automatically when completed
4. File checkout – We chose not to allow file checkout without MAC verification

The only portion of this project that we did not completely implement as prescribed was the client application. We took the time to implement unit testing for this program and can demonstrate most of the functionality of the S2DR. We also wrote a lightweight client application that can demonstrate the limited features we were not able to demonstrate through the unit testing.

# Security Analysis

One of the requirements for this project was to analyze our source code with a static code analysis tool in order to identify any potential security vulnerabilities within our code. Unfortunately we were not able to complete this step, as there are no free or cheap static code analysis tools for Python. In addition, Python does not readily lend itself to being analyzed by static code analysis tools. We did manually analyze our source to ensure we minimized the use of unsafe function calls.