A Cryptographically Secure Departmental Resource Server

Chris Watson

2190594

Project outline

Motivation

Sharing resources across organisations or departments is a difficult and daunting task. Security is extremely important for resources and determining access control is an equally complex and error-prone task. Common methods include the Role-based access control (RBAC) approach, however even incorporating permissions matrices does not often provide a suitably fine-grained system for precise or modular access. This problem is easily solved with the use of Attribute-based access control (ABAC) instead. Wherein, users are granted specific attributes instead of roles, allowing for any user to be given an extremely precise and even unique set of attributes. As such the likelihood of granting too much access to a user is greatly reduced compared with RBAC. In the case of this project's resource server, an ABAC system was determined to be inadequate as resources stored on the server would be left vulnerable to attacks or leaks. Consequently, an Attribute-based encryption (ABE) system should be built into the server to handle the secure storage and transmission of resources through advanced encryption. Like ABAC, the ABE system would utilise policies of attributes for users to determine if a resource should be decrypted by a given user.

Aims

This project will develop a full resource server product that meets the definition of "cryptographically secure". The product will feature 2 servers: an online 'dumb' resource server to store the ABE encrypted resources and an offline 'cold storage' master key server. The online server will store all uploaded encrypted files, but will not be aware of the contents of said resources, while supporting uploading and downloading of resources. The master key server will be tasked with the distribution of keys for users, with keys containing the signed attributes of said user. The end user will also have a locally running web client for communication with the resource server, allowing for simple encryption & decryption of files, along with uploading & downloading. Additionally, a CLI tool will be supported in order for user's to encrypt and decrypt resources downloaded from the online server. The ABE system

will run the cipher text policy (CP) scheme and rely on the OpenABE library - and associated PyOpenABE package - from Zeutro LLC (found here). With the web server using Python's Django framework and an undetermined Python ABAC library such as VAKT (found here).

Progress

- *Language chosen: project will be implemented in Python with libraries that wrap/bind to lower-level C libraries.*
- *ABE library chosen: project will utilise the OpenABE library from Zeutro LLC, as mentioned above.*
- *Web server framework chosen: project will make use of Flask to build a simple and quick-to-deploy web server.*
- *Background research conducted on ABE technologies.*
- *Component diagram for software and network architecture outlined.*
- *Sequence diagrams designed to describe the flow of information within the product.*
- *OpenABE library downloaded, compiled and tested.*
- *Python bindings for OpenABE (PyOpenABE) compiled and tested.*
- *Example policies for ABE designed, along with associated Use Cases.*
- *Universe of attributes for ABE defined, based off of the associated Use Cases.*
- *Initial prototype for master key server developed.*
- *Initial prototype for resource server developed.*
- *Simple tests for encrypting and uploading resources to the server carried out successfully.*
- *Similar tests for downloading then attempting to decrypt resources carried out successfully with decryption success correctly relying on the user's attributes.*

Problems and risks

Problems

- *Defining the universe of attributes for the ABE system proved more complex and time consuming than anticipated due to lack of prior knowledge on the structure of the School of Computing Science.*
- *Similarly, the defining of the related policies also proved difficult due to a lack of experience with policy language.*
- *The OpenABE library is both new and relatively inactive, and as such the available documentation can be unclear with regards to certain problems.*
- *OpenABE itself proved troublesome to correctly compile on the development system.*
- *Further, the PyOpenABE bindings were extremely difficult to compile due to compatibility*

issues with macOS 10.14.1 and the OpenSSL library.
- *Lastly, files encrypted by PyOpenABE do not, at present, appear to be compatible with OpenABE CLI tools.*

Risks

- *On inspection it seems that the policy metadata attached to resources is not extractable, at least as it is currently understood. Mitigation: on encryption of a resource, save a copy of the policy to a separate metadata structure.*
- *Priority for this project lies with the resource server, master key server and CLI tool interactions for encrypting, uploading, downloading and decrypting. Depending on progress, a full-featured web server with GUI and ABAC support may be too time consuming and affect the key elements of the product. Mitigation: cut (or simplify) the web server GUI to ensure resource server is working correctly.*

Plan

Semester 2

- *Week 1-2: continue to develop and improve web server and related GUI, regarding the underlying resource server.*
  - Deliverable: *complete web server with support for ABAC system to determine which resources are visible to a user.*
- *Week 3-5: implement and test designed Use Cases with product.*
  - Deliverable: *tested policies and scenarios of the designed Use Cases with the product, test results show that for the Use Cases the servers are working as expected.*
- *Week 6: evaluate extensibility of final system.*
  - Deliverable: *discussion of how extensible the system is, regarding new policies and attribute universes, determine future steps to improve portability of the system.*
- *Week 7-9: final implementation and improvements to web server and specifically GUI.*
  - Deliverable: *polished software ready, passing basic tests, ready for evaluation stage.*
- *Week 9: evaluation of system with further tests.*
  - Deliverable: *testing for usability and effectiveness of the system, evaluate the success of the system in direct relation to the defined Use Cases*
- *Week 8-10: Write up.*
  - Deliverable: *first draft submitted to supervisor two weeks before final deadline.*