Specification: Designing a Virtual Security Layer for Cloud Content

Lukas Klingsbo

October 8, 2015

Contents

1 Background

This work will be done at the company Uprise (Dragarbrunnsgatan 36C, Uppsala). The goal of this project is to design and develop a local access control layer for files and views in a content distribution network. This should make handling of content safer and make it easier to have control over the security settings of groups of files and will also make it possible to enforce local (as in not on the CDN) auth and audit logging.

2 Description

The project should design and possibly develop a system where it is possible to set different types of access to different clusters of virtual files. For example, the access control could be divided into three groups:

- Very secret - Only accessible from white listed IP addresses
- Secret - Only accessible from authenticated users with the correct access
- Public - Accessible by anybody

The security rules put in place should not be hard coded, but possible to update and extend. When a file is set to a belong to one of these groups it should not be static, it should be possible to, at a later stage, assign it to another group.

A virtual file, as described in the beginning of the description, is a form of pointer connected to some meta-data that points at an actual file in the filesystem or remote service. One "physical" file can be referenced by many virtual files.

It should also be possible to create so called views which a group of files are added to, which then have their own security settings. A view is a subset of files, and a view can be a view of another view and then only contain files which is in contained in the parent view. A file can be a part of several views that then in their turn have different security settings. The system should then be able to determine what the correct security setting for the actual file is or if it needs to create several versions of the file.

For example, which security rules should take precedence if a file, lets call it "Test.jpg", is uploaded to a CDN once, but is contained in two views with different security settings. If then Test.jpg is changed in one of the views, but is not supposed to be changed in the other, how is that handled and how is it determined when a change should propagate through the whole system or when it should just create a new version of it in the current virtual view.

More things to examine:

- A way to dynamically switch a single or a group of files security settings has to be examined as the storage method of the files usually will be distributed.
- To make sure you are handling the correct file previews has to be shown in the system.
- Access to the system should be configured in a modular sense, one module could be LDAP.

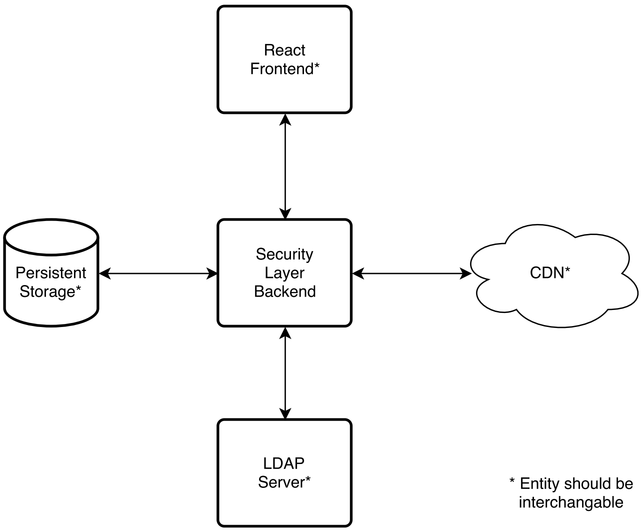

Figure 1: System Overview

If looking at Figure 1, the component that this work is supposed to specifically design is the Security Layer in the middle, everything else should be interchangeable. The persistent storage will most likely be a database which keeps track of the files and security settings. The CDN could be any service with a rougher security system which needs to be more fine grained for assets. The LDAP server in the figure can be any form of authentication service so that wont need to be handled in the security layer. The React front-end is for handling files and security settings in the back-end, it should be possible to write any front-end for it.

3 Methods

- The design will be well evaluated in the sense of security and technical usability
- The prototype/system will be developed in Scala and React
- A literature study will be done to find more relevant litterature once the specification is accepted
- The proof should be an informal paper proof or if time is sufficient a proper proof should be done in a model checker
- Results from the informal proof will be evaluated and documented
- The implementation will be evaluated with existing security tools, it needs to be researched which these tools will be
- Results from these tools and that are relevant to the report will be added in an appendix
- Everything relevant in the work process, from problems that occured to final results, will be documented in the report
- The code will have its own documentation separated from the report
- The prototype/system will be released under an open source license
- A work station and desk space will be provided by Uprise

4 Relevant Courses

- Computer Networks I (1DT052)
- Computer Networks II (1DT074)
- Distributed Systems (1DT064)
- Secure Computer Systems I (1DT072)
- Secure Computer Systems II (1DT073)
- Process Oriented Programming (1DT083)

5 Delimitations

If the project is easier than expected it could be expanded by looking into automatically doing simpler image analysis to find images that are about to be assigned to the incorrect category, for example a “Very secret” image that is about to be published as “Public”.

It is possible to do delimitation of the core research question as the project proceeds, but it will most likely not be needed as the research question is quite concise.

6 Time Plan

Note that this is just a rough time plan that will change during the course of the project when the circumstances change. There will rarely be one task at a time, but rather a lot of the tasks will be overlapping.

- A few days - Properly setting up the work station
- 2 weeks - Literature study and obtaining relevant books etc
- 20 weeks - Writing the report should be done pretty much throughout the whole project
- 1 week - Analyze the current system for managing the CDN assets
- 1 week - Analyze what has been done earlier regarding the subject and determine if anything is useful for this project
- 3 weeks - Define system mechanism and model
- 3 weeks - Analyse system mechanism and model (this item will overlap with the definition)
- 3 weeks - Informal proof (or formal if time is sufficient) of the systems mechanism and model
- 8 weeks - Implement a prototype/system

  1 week - defining and developing backend API

  2 weeks - define a basic set of extendable security rules

  <1 week - investigate and implement "physical" file handler

  <1 week - implement simple persistant storage for meta data

  1-2 weeks - implement authentication interface

  <1 week - implement CDN interface

  2 weeks - implement user interface
- 1 weeks - Test security of implementation, if only a prototype is developed the security should only be analysed in theory

The interfaces does not have to have anything real connected to them, it just needs to be possible to hook up services to them.

7 References

One possibility to connect the security layer to:

`http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-signed-cookies.html`