A Schema for Field Level, Attribute Based Access Control (ABAC) in a Key Value Datastore

David Wilson

Project Description:

Organizations around the world are producing, recording, and storing more information than ever. The rate of this data production and storage is increasing as well. Spending on big data technologies is expected to hit $187 billion this year. [Olavsrud, 2016] In order to make better use of this data, organizations are storing information in fewer and larger systems. This colocation enables new insights but it also comes with a new set of security challenges.

One challenge organizations often run into is fast, direct access to specific pieces of data based on some identifier in the data. In order to quickly lookup information across vast amounts of data, many organizations are using databases that fall under the category of NoSQL. A specific subset of which are key-value stores, also known as key-value databases. These off direct lookup to a “value” when a user asks for specific “key”.

A second challenge in big data systems is adequately controlling data access while not excessively reducing performance of that data access. This is difficult because each individual set of data in a system can have it’s own access control roles, policies, etc.. Within this dataset, separate columns, rows, or even fields can have specific access controls that must be enforced. As the number of datasets held in a system increases, the number and complexity of access control rules tends to increase. Processing these rules can be extremely costly. The National Institute of Standards and Technology (NIST) currently recommends attribute based access control over other models like mandatory access control, discretionary access control, and role based access control [NIST, 2014].

This project will describe and implement a schema for enforcing field level ABAC on a key value datastore while adding little performance overhead. The implementation will use Apache Accumulo as the datastore technology. Accumulo is an implementation of Google’s Bigtable as described in, *Bigtable: A Distributed Storage System for Structured Data* [Chang, 2008]. It includes extensions to this design including a mechanism for server side programming and cell-based access control.

Preliminary Literature Search:

ABAC in big data is a relatively well studied topic. As mentioned earlier, NIST has produced topics on the subject including a general guide for implementation of ABAC. It also has

published papers specifically focused on Big Data including *An Access Control Scheme for Big Data Processing* [NIST, 2014].

Specific research on ABAC in KeyValue datastores is much more limited. There was a presentation at Accumulo Summit 2015 conference [Garcek, 2015] which discussed how to support ABAC in Accumulo using XACML. This work attempts to solve the same problem but in a different manner. It relies completely on Accumulo's visibility column which requires data to be stored at the field level even if each field does not need to be queryable. If an entire record needs to be returned when a key is found, then the whole record must be reconstructed from many key value pairs.

Approach:

Table Schema

The table schema below will be used. Advantages and disadvantages will be explored including performance, storage cost, and utility.

Row	Column Family	Column Qualifier	Visibility	Timestamp	Value
Query Term	Index Name		Term Attribute	Timestamp	Record

Data Model

A simple data model will be developed which will allow for field level attribute definition. This data model will be required only in testing. The schema and access control mechanisms will be data agnostic and allow for pluggable data models.

Ingest Mechanism

An ingest mechanism will be created using Accumulo's BatchWriter interface which will read the proposed data model and write rows into a table using the proposed schema.

Access Control Mechanisms

There will be two access control mechanisms developed:

- A KeyValue store agnostic external filter which reads the results of the query and returns only records the end user is allowed to receive.
- An Accumulo specific implementation which takes advantage of two features; visibility filtering in order to determine if the user can access the term being queried and the server side programming extension, Iterators. This will provide data access control before the data is ever transported over the network.

Evaluation:

Evaluating the schema and filtering mechanism will be two-fold:

- Accuracy: Most important is the ability to accurately and consistently enforce ABAC at the field-level
- Performance: Query performance will be measured using the two proposed control mechanisms, Accumulo and non-Accumulo specific. Additionally, the

Time Plan:

Week 1: Test Environment

A test environment image will be built for use with Amazon EC2. This will enable quick build up and tear down of a reusable environment for development, testing, and evaluation. It will run Centos 7 and include Apache Accumulo 1.8.1, Java 8, Hadoop 2.7.3, and Zookeeper 3.4.9.

Week 2: Data Model

The data model will be defined. This will include a library for read, write, and generating records.

Week 3: Table Schema

A table schema will be finalized and trialed for use with Accumulo's visibility filtering capability.

Week 4-5: Ingest API

An API will be written for batch writing data to Accumulo tables. This will include an implementation for the data model devised in week 2.

Week 6-7: Query API

A simple API for querying terms in the database will be written. It will include a way to tag the query with attributes to use during access control decisions.

Week 8: External Filter

The external filtering mechanism will be written which provides access control based on the attributes associated with the query.

Week 9: Accumulo Specific Filter

Term attributes will be input to the visibility column and included in the ingest API. Server side access control will be enforced at the field level using an Accumulo Iterator.

Week 10-12:

Project will be evaluated based on accuracy and performance. This will require production of mock datasets, numerous attributes to apply to fields within the data, and a set of users with varying attributes.

References

Olavsrud, T. (2016, May 24). Big data and analytics spending to hit $187 billion. Retrieved from https://www.cio.com/article/3074238/analytics/big-data-and-analytics-spending-to-hit-187-billion.html

NIST. 2014. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. (Special Publication (NIST SP) - 800-162). Retrieved from: http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., . . . Gruber, R. E. (2008, 06). Bigtable. *ACM Transactions on Computer Systems, 26*(2), 1-26. doi:10.1145/1365815.1365816

NIST. 2014. An Access Control Scheme for Big Data Processing. Retrieved from: https://csrc.nist.gov/csrc/media/projects/access-control-policy-and-implementation-guides/documents/big_data_control_access_7-10-2014.pdf

Garcek. 2015. Extending Accumulo to Support ABAC using XACML. Retrieved from: http://accumulosummit.com/program/talks/extending-accumulo-to-support-abac-using-xacml/