Enhancing MongoDB with fine-grained context-aware access control

Pietro Colombo    Elena Ferrari

Department of Theoretical and Applied Sciences,
University of Insubria, Varese – Italy

{pietro.colombo,elena.ferrari}@uninusbria.it
NoSQL datastores

• Designed to meet data management requirements of modern application
  • Large volumes of data
  • Structured, semi-structured, and unstructured data
  • Flexible and easy to use programming
  • Dynamic schema
• Auto sharding and automatic replication
• More scalable than RDBMSs and superior performance
• Data management support for several Big Data scenarios
NoSQL datastores

• Several proposals
different data models and query languages
no de facto standard

“The rise of NoSQL databases marks the end of the era of RDBMS
dominance.”

M. Fowler
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Oracle</td>
<td>Relational DBMS</td>
<td>1442.10</td>
<td>-4.03</td>
<td>-60.64</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>MySQL</td>
<td>Relational DBMS</td>
<td>1294.26</td>
<td>+9.68</td>
<td>-14.83</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>Microsoft SQL Server</td>
<td>Relational DBMS</td>
<td>1131.03</td>
<td>-18.09</td>
<td>-76.77</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td>5</td>
<td>MongoDB</td>
<td>Document store</td>
<td>277.32</td>
<td>-1.27</td>
<td>+52.70</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td>4</td>
<td>PostgreSQL</td>
<td>Relational DBMS</td>
<td>273.52</td>
<td>+5.20</td>
<td>+32.88</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>DB2</td>
<td>Relational DBMS</td>
<td>201.04</td>
<td>+3.40</td>
<td>+14.57</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>Microsoft Access</td>
<td>Relational DBMS</td>
<td>145.58</td>
<td>+3.39</td>
<td>+0.22</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td>9</td>
<td>Cassandra</td>
<td>Wide column store</td>
<td>106.55</td>
<td>+1.66</td>
<td>+24.82</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
<td>8</td>
<td>SQLite</td>
<td>Relational DBMS</td>
<td>105.16</td>
<td>+2.86</td>
<td>+15.87</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>13</td>
<td>13</td>
<td>Redis</td>
<td>Key-value store</td>
<td>94.73</td>
<td>+0.17</td>
<td>+32.69</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td></td>
<td>10</td>
<td>SAP Adaptive Server</td>
<td>Relational DBMS</td>
<td>85.49</td>
<td>-1.20</td>
<td>+5.48</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>11</td>
<td></td>
<td>Solr</td>
<td>Search engine</td>
<td>82.93</td>
<td>+0.92</td>
<td>+15.77</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td></td>
<td>12</td>
<td>Teradata</td>
<td>Relational DBMS</td>
<td>70.13</td>
<td>-0.13</td>
<td>+4.60</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td></td>
<td>17</td>
<td>Elasticsearch</td>
<td>Search engine</td>
<td>64.83</td>
<td>+0.17</td>
<td>+32.77</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td></td>
<td>HBase</td>
<td>Wide column store</td>
<td>61.70</td>
<td>+0.57</td>
<td>+21.44</td>
</tr>
</tbody>
</table>
Addressing security issues

• Current situation:
  • Lack of standard data protection tools
  • Basic form of access control, and no support for:
    • fine grained AC
    • context awareness

• Goal: systematic, general approach to enhance the access control capabilities of NoSQL datastores
  • No ad-hoc implementation at application level
Roadmap

- The goal: building context aware fine grained access control on top of NoSQL datastores
  - Enhancement, no substitution
- A very ambitious goal due to variety of existing datastores
  - Different system architectures, query languages data models
- An incremental approach focusing on one datastore at a time
  - Generalization of enforcement mechanisms
MongoDB

• Our first choice: the most popular NoSQL datastore
• A document oriented DB → record = document
• Documents serialized as JSON objects
  • fields may specify a primitive value
  • or include other documents, arrays, and arrays of documents
```json
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```


MongoDB RBAC

• RBAC at collection-level
  • administrators grant privileges scoped to specific collections to given roles
  • privileges are granted to users through role assignments

```
privileges: [
  { resource: { db: "products", collection: "inventory" },
    actions: [ "find", "update", "insert" ] },
  { resource: { db: "products", collection: "orders" },
    actions: [ "find" ] }
]
```
Drawbacks of MongoDB RBAC

• *Issue*: MongoDB RBAC collection level granularity is inappropriate
  • documents can refer to multiple subjects
• *Desiderata*: access control at document level

Ex. 1, Enron emails: employees can specify different access control preferences for their emails

• *Issue*: each document in a collection could have a structure different from the other documents
• *Desiderata*: Access control mechanisms applicable to documents with specific structural characteristics

Ex. 2, emails can have zero to multiple attachments, and different policies could be defined based on the presence of these elements
Drawbacks of MongoDB RBAC

• *Issue*: no context aware mechanism
  • Static management of access privileges through role assignments

• *Desiderata*: context aware access control

Ex. 3, given emails can be accessed only if the MAC address of the device used for requesting the access belongs to an authorized list

Ex. 4, given emails can be accessed only within specific time intervals of a working day
A step forward

• MongoDB clients and servers interact using a protocol called Wire

• Development of the MongoDB enforcement monitor (Mem), as a Wire interpreter and MongoDB proxy

• Mem intercepts, analyzes, and possibly rewrites the MongoDB Wire messages that are exchanged by the clients and the server:
  • Independence from specific versions of MongoDB
  • Relies on the security mechanisms natively provided by MongoDB
Mem

• Wire based client-server interaction:
  • MongoDB client request
    • E.g., db command execution requests, query execution requests, result set analysis requests...
  • MongoDB server response
  • Server responses are only issued to answer client requests

• Mem behaves like:
  • A client for the server
  • A server for the clients
Assumptions

- Context model specified as a data record
  - Ex: a context property specifies the MAC address of devices hosting a MongoDB client, the time at which a client request is issued, etc.

- Finest policy granularity: MongoDB document

- Context based policies at document level are encoded within a dedicated field `policy` of the protected documents
  - E.g., an array of strings each specifying a MAC address
Monitoring of target messages

• Authentication

• Whenever a user \( u \) starts the authentication process
  1. Mem lets the client wait for the authentication response
  2. Injects commands to profile \( u \)
     • Derive the values of target reference contextual properties for \( u \)
       *E.g. the list of devices authorized for \( u \)*
  3. Forwards the authentication response to the client
### Early experiments: Enron dataset

<table>
<thead>
<tr>
<th>Id</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>Count messages issued in a given date range</td>
</tr>
<tr>
<td>q2</td>
<td>Select a single message</td>
</tr>
<tr>
<td>q3</td>
<td>Select emails sent in a specific time interval and sort them by date</td>
</tr>
<tr>
<td>q4</td>
<td>List all distinct receivers</td>
</tr>
<tr>
<td>q5</td>
<td>List all distinct senders</td>
</tr>
<tr>
<td>q6</td>
<td>Return senders in common with receivers</td>
</tr>
<tr>
<td>q7</td>
<td>Return senders who did not receive any email</td>
</tr>
<tr>
<td>q8</td>
<td>Return senders who has also received an email (To/Cc/Bcc)</td>
</tr>
<tr>
<td>q9</td>
<td>Return senders of emails who were Enron employees</td>
</tr>
<tr>
<td>q10</td>
<td>Count the number of messages received by given email addresses</td>
</tr>
<tr>
<td>q11</td>
<td>Derive the recipient lists of each sender</td>
</tr>
<tr>
<td>q12</td>
<td>Derive the number of emails received by a user grouping them by message sender</td>
</tr>
</tbody>
</table>
Conclusions

• A roadmap to integrate context aware FGAC into NoSQL datastores
• A step forward by focusing on MongoDB
  • Encouraging results
• Future work
  • short term
    • finer granularity -> field level
    • analysis of enforcement overhead
  • long term
    • Generalization of the approach for other NoSQL datastores
Questions?

HOW TO WRITE A CV

DO YOU HAVE ANY EXPERTISE IN SQL?

NO

Leverage the NoSQL boom

DNE'ST MATTER. WRITE: "EXPERT IN NO SQL"