2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops

Access Control in Social Enterprise Applications: An Empirical Evaluation

Rafae Bhatti [1], Camille Gaspard [2], Cristina Nita-Rotaru [3]
[1]PwC
[2]Cisco Systems
[3]Purdue University

***Abstract*—The social enterprise is reported as one of the biggest IT trends, and is only increasing in popularity. Many enterprises are adopting social media communication channels such as Yammer, Chatter, and Jive for collaboration amongst employees. One key concern however is the lack of user-level access control mechanisms in these applications. In particular, introducing social media applications in government, healthcare and financial sectors requires strict controls on which employees can access or share what kinds of company data based on various federal and state regulations. The existing vendor solutions do not provide fine-grained access control policies to support these requirements, and the impact of adding such policies to these applications have not been explored yet.**

In this work we provide an empirical evaluation of embedding fine-grained access control policies in Group Communication Systems (GCS) which serve as a mechanism for message exchange in social media applications. Our evaluation is based on a proposed framework for Role-Based Access Control for GCS in wide area networks (WAN) scenarios where the access control policies are specified and enforced using the X-RBAC policy framework. The main focus of this work is to evaluate the performance of our proposed framework and demonstrate that adding the access control mechanisms to an existing GCS incurs minimal overhead, looking especially at the challenges in WAN scenarios that are relevant to message exchange between geographically distributed employees in the enterprise. We show that with the use of caching, the proposed framework adds minimal overhead in WAN environments, while still providing the advantages of having such a framework built in the GCS's interface to enable access control for the social enterprise.

I. Introduction

The social enterprise is reported as one of the biggest IT trends, and is only increasing in popularity [1]. Many enterprises are adopting social media communication channels such as Yammer, Chatter, and Jive for collaboration amongst employees. One key concern however is the lack of user-level access control mechanisms in these applications. In particular, introducing social media applications in government, healthcare and financial sectors requires strict controls on which employees can access or share what kinds of company data based on various federal and state regulations ([2], [3], [4]).

An example of such application for a healthcare enterprise dealing with sensitive personal data is Doximity [5], which connects physicians for exchanging healthcare information. Access control plays an important role in this setting since physicians in Doximity can create a private profile to share only with the colleagues whom they have confirmed, and also elect to do secure or insecure group messaging among confirmed participants. Any inadvertent release of this data to unintended recipients will result in a significant privacy breach, and hence access controls must be in place to ensure that messages are always sent securely by authorized users.

Similar trends are observed in the financial sector, where the growth of social media applications is slow due to the strict laws around data sharing between employees. For example, employees in a central branch of a financial institution for a region may communicate and share deal data across individual branches in that region, but employees in individual branches may not have mutual access to each other's data. In government sector too, there are strict requirements on the information flows that are permitted between employees at certain levels. Social media applications designed for the enterprise must meet these requirements in order to become widely adopted.

Yet, the existing vendor solutions do not provide fine-grained access control policies to support these requirements, and neither has the impact of adding such policies to these applications been explored yet. Social media applications are dynamic and distributed by their nature, and major challenges for introducing them in an enterprise are reconciling flexibility with scalability, and enforcing access control while supporting process failures and network partitions. With these considerations in mind, in this work we provide an empirical evaluation of embedding fine-grained access control policies in Group Communication Systems (GCS) which serve as a mechanism for message exchange in social media applications.

Our evaluation is based on a proposed framework for Role-Based Access Control [6], where the access control policies are specified and enforced using the X-RBAC policy framework [7] in the Spread[8] group communication system. The main focus of this work is to evaluate the performance of our proposed framework and make sure that adding the access control mechanisms to an existing group communication system incurs minimal overhead, looking especially at the challenges in WAN scenarios that are relevant to message exchange between geographically distributed employees in the enterprise. We aim at showing in this work that with the use of caching, the proposed framework adds minimal overhead in WAN environments, while still providing the advantages of having such a framework built in the GCS's interface to enable access control in the social enterprise.

The remainder of the paper is organized as follows. We first

978-0-7695-5023-7/13 $25.00 © 2013 IEEE
DOI 10.1109/ICDCSW.2013.4

describe Spread[8], the group communication system we use in this work, and then describe X-RBAC, the access control policy framework. We then describe the integration of X-RBAC with Spread to evaluate the impact of adding access control policies. We then provide the details of the access control evaluation, and finally conclude the paper and provide some directions for future work.

II. Spread: A Group Communication System

Spread [8] is a group communication system for wide- and local-area networks. It provides reliability and message ordering (FIFO, causal, agreed/total ordering) as well as a membership service. Spread consists of a server and a library linked with the application. The process and server memberships correspond to the model of light-weight and heavy-weight groups. This approach amortizes the cost of expensive distributed protocols, since these protocols are executed only by a relatively small number of servers, as opposed to (a much larger number of) all clients. Spread operates in a many-to-many communication paradigm, where each member of the group can be both a sender and a receiver. Although designed to support small- to medium-size groups, Spread can accommodate a large number of different collaboration sessions, each spanning the Internet. Spread scales well with the number of groups used by the application without imposing any overhead on net- work routers. The Spread toolkit is publicly available and is being used by several organizations for both research and practical projects. The toolkit supports cross-platform applications and has been ported to several Unix platforms as well as Windows and Java environments.

III. XRBAC: An RBAC policy framework

To implement our RBAC GCS proposal [6] and integrate access control in Spread, we use an XML-based RBAC policy specification language, X-RBAC, outlined in [7] that incorporates the salient features of the RBAC model [9]. The specification language for X-RBAC aims at modeling the basic RBAC elements (users, roles, permissions) and their associated set-relations, namely user-to-role and permission-to-role assignments. [1] The following are the core components of the policy language.

Credentials: The user, role, and permission credentials in X-RBAC comprises of attributes which are relevant for role assignment. An example of user and role credential is included in an XML User Sheet (XUS) and an XML Role Sheet (XRS) in Table I. An example of permission credential is included in an XML Permission Sheet (XPS) in Table II.

Assignment Rules: An integral component of RBAC polices in X-RBAC is the specification of rules for user-to-role and permission-to-role assignments. The assignment policies are specified in an XML User to Role Assignment Sheet (XURAS) and XML Permission to Role Assignment Sheet (XPRAS), and an instance of each is illustrated in Table II.

[1] X-RBAC and its extended versions also support advanced features such as integrity and contextual constraints, which we will not review for the purposes of this work.

An assignment rule consists of an assignment constraint, which comprises of multiple assignment conditions. Each assignment condition contains a set of logical expressions to encode rules on a given credential type, and may be combined using Boolean connectives AND (all rules must be true), OR (at least one rule must be true), and NOT (no rule must be true). An assignment condition is satisfied if all of its included rules encoded using logical expressions are satisfied according to the respective mode. Role assignment occurs as a consequence of an assignment constraint being satisfied.

IV. Integration

In this section, we describe how the access control policy framework provided by X-RBAC is integrated within Spread.

There is no default Access Control Monitor (ACM) in Spread that can enforce anything except an ALLOW_ALL policy that permits all requests by users to join/leave group and send/receive messages. Spread, however, is designed with a pluggable mechanism with the capability to integrate specific security services [10]. The integration of an RBAC policy in Spread required for us to plug X-RBAC as an ACM for all access requests received by the system. It also required for us to provide a mechanism to associate the access control policy with a specific operation (such as join, leave, send, receive). This means that we used the RBAC policy specified in Section III to instantiate the ACM, and modify the relevant operations in Spread to request access decisions from X-RBAC policy engine in order to enforce the policy.

A. Spread Operations

The following group operations are currently supported by Spread to which we applied access control:

- **join_group**
- **leave_group**
- **send**

The Spread distribution comes with a simple client, named *spuser*, which allows a user to request one of the above operations. We extended this client to allow us to measure the time for these operations in order to compare the impact of adding access control.

B. X-RBAC APIs

The X-RBAC framework has been implemented as a set of Java and C APIs. For this work, we use the C APIs to integrate it within Spread system. The following are the key APIs used:

- **IsUserInRole(Policy P, User x, Role y):** For Policy P, returns True if user x is assigned to role y, else returns False
- **IsPermInRole(Policy P, Permission p, Role y):** For Policy P, returns True if permission x is assigned to role y, else returns False

The above mentioned APIs were invoked when the relevant operation is requested by a client application (for evaluation purposes we used the *spuser* client provided with the Spread toolkit), and a decision is returned by the policy engine. If the request is allowed by the policy, it is sent to the Spread server, otherwise it is aborted.

```
<XUS>
 <user user_id="john">
   <cred_type type_name=NetworkUser>
     <cred_expr>
        <domnain>engineering</domain>
        <level>secret</level>
     </cred_expr>
   </cred_type>
  </user>
</XUS>
(a)
```

```
<XRS>
 <roles>
  <role role_id="NetworkRole">
    <senior>AdminRole</senior>
    <cardinality>20</cardinality>
  </role>
 </roles>
</XRS>
(b)
```

TABLE I
XML INSTANCES OF (A) XUS (B) XRS

```
<XURAS>
 <ura ura_id="Network">
  <role_id>NetworkRole</role_id>
  <users>
   <user user_id="john">
    <cred_conditions>
     <cred_condition>
      <type>NetworkUser</type>
      <logical_expr>
       <predicate>
         <operator>eq</operator>
         <name>domain</name>
         <value>engineering</value>
       </predicate>
      </logical_expr>
     </cred_condition>
    </cred_conditions>
   </user>
  </users>
 </ura>
</XURAS>
(a)
```

```
<XPS>
 <permission perm_id="send">
  <object_type>message</object_type>
  <operation>send</operation>
 </permission>
</XPS>
(b)
```

```
<XPRAS>
 <pra pra_id="SendMsg">
    <role_name>NetworkRole</role_name>
    <permissions>
     <perm_id>send</perm_id>
    </permissions>
 </pra>
</XPRAS>
(c)
```

TABLE II
XML INSTANCES OF (A) XURAS (B) XPS (C)XPRAS

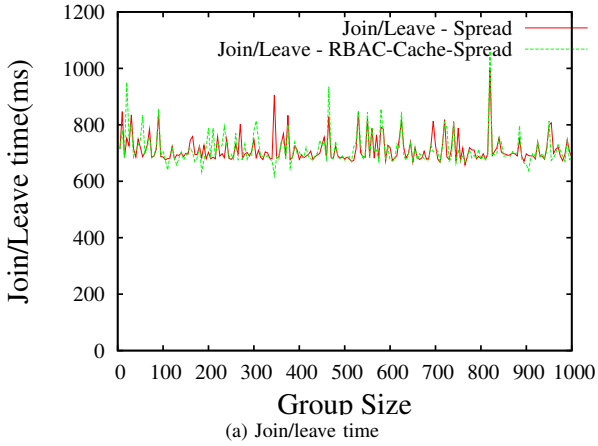
V. EVALUATION

This section aims at evaluation of Role-Based Access Control for GCS [6] using Spread [8] as the GCS integrated with the X-RBAC [7] policy framework. More specifically, we are interested in measuring the performance of the system after adding the role-based access control mechanism. This is to demonstrate, through a prototype implementation, the effectiveness of using the RBAC framework in Wide Area Network (WAN) scenarios. This evaluation allows us to assess the impact of embedding access control in the applications for the social enterprise that are built using this mechanism.

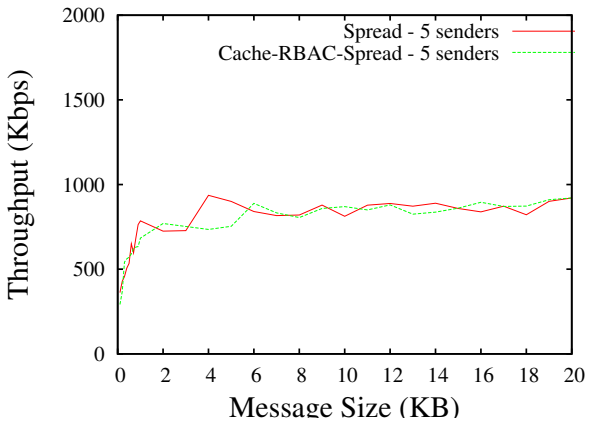
We aim at evaluating the integrated framework by measuring the performance of two classes of operations: join/leave (latency) and send/receive (throughput) operations. These two classes of operations represent the main functionalities provided by the GCS, and it follows naturally that their performance has a direct effect on the performance of applications built on top of GCSs. This is why it is very important to evaluate them in different realistic scenarios in WAN setup to make sure that by making the access control mechanism an integrated part of the GCS, the performance of such systems does not degrade in a way that affects the application.

We use the following metrics when evaluating our proposed framework:

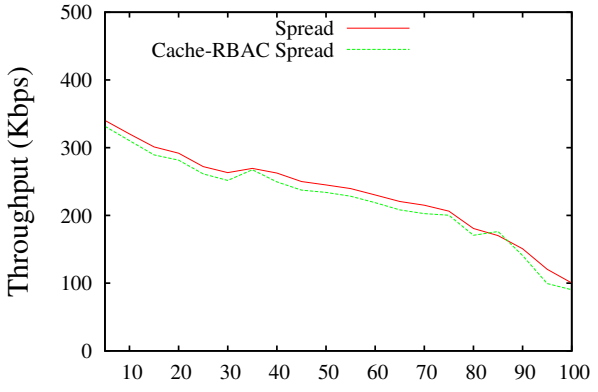
- **Latency** is the time between when a client performs a join/leave, till the time when all nodes (including the client performing the action) in the group perceives this action.
- **Throughput** is the rate of bytes per second that can be sent in the system in a certain scenario without getting a send error.
- **Overhead** is the extra amount of processing and memory caused by the proposed framework over a standard implementation of the studied group communication system.



(a) Join/leave time



(b) Throughput as function of message size with a fixed group size of 20



(c) Throughput as function of group size with a fixed message size of 5,000 bytes and one sender

TABLE III
WAN JOIN/LEAVE AND THROUGHPUT RESULTS

A. Wide Area Network Setup

Testbed and Experiment Setup: For WAN scenarios, we conducted our experiments on the PlanetLab [11] Internet testbed. We selected 20 nodes that are geographically separated and have a low clock drift since our experiment is sensitive to such drifts. We did try to compensate for clock drifts by estimating the drift between the clocks while running our WAN experiments. Each of these nodes would run a Spread server and one or more clients. On each client, we used an RBAC policy similar to the one in Tables I and II. The permission to send a message was granted to a role, and all authorized users were granted this role. We deliberately kept the assignment policy simple to focus on the performance evaluation. All our results are averaged over 5 runs.

Join/Leave: The join/leave time is defined by the time taken since a peer initiates a join, till all peers in the group (including the peer itself) receive a membership message confirming the agreed new view of the group. To measure the join/leave latency, we record these times perceived by all alive peers and average them to get a data point representing the current group size. We used the Network Time Protocol ntp [12] to synchronize the clocks.

The results of the join/leave experiment for the wide area network scenario are presented in Figure (a) in Table III. We conducted the join/leave experiment in two configurations: standard group communication system (or original Spread), and RBAC with caching. The results show that adding RBAC does not incur any noticeable overhead on the join/leave performance perceived by all members of a group in WAN scenarios.

Note that in our WAN setup we do not see a linear relationship between the number of clients and the latency. The latency of join/leave in WAN starts from a high value and maintains that high value for an increasing number of clients reaching a 1000. This can be understood by the fact that usually in a such WAN scenarios (while having nodes spread all around the world) at least one of these nodes would have a slow response confirming receiving the leave/join event to other nodes and thus the latency would be capped by at least one slow responding machine. This can be a result of the machine being overloaded, the network segment being fully used, or both. In either case the response coming back from that machine will be extremely slower than the others.

Throughput: As in the case of join/leave, we aim here at measuring the throughput of the system with, and without, the access control framework. The throughput is defined here by how many bytes can be sent per second in the system without getting sending errors. The throughput has major importance in specially bandwidth-demanding applications. In order to give a realistic estimate, we need to evaluate the throughput of the system under different scenarios. More precisely we need to vary: number of users in the group; number of simultaneously sending peers; and size of messages. By varying these three factors we create different scenarios and get an understanding of the performance of the system in most realistic scenarios. For each scenario, a certain number of simultaneous senders send messages of a specific size (1400 and 14,000 bytes for message sizes are used) to the group. We log the time $t0$ when the senders start sending data, and the time $t1_i$ when member

i got all data sent. We divide the number of bytes sent in the system by the time difference $t1_i$ - $t0$, we then average over i members to get the throughput of the whole system.

The results of the two throughput experiments for the wide area network scenario are presented in Figure (b) and (c) in Table III. In the first experiment (b) we fixed the number of nodes in the system to 20 and we measured the throughput as function of message size. In the second case (c) we wanted to measure the effect of increase in clients in a group to the throughput perceived by each client. In order to do so, we fixed the size of the message sent to 5000 bytes. We have picked this size for the message because it is the value that plateaued the throughput in Figure (b). We conducted each experiment in two configurations: standard group communication system (or original Spread), and RBAC with caching. The results show that adding RBAC does not incur any noticeable overhead on the join/leave performance perceived by all members of a group in WAN scenarios.

B. Overhead

The overhead added is due to CPU time and memory. The major overhead is the memory space needed to cache the policy ($O(n)$ where n is the number of policy entries.) Luckily, each policy entry is represented in a sequence of zeros and ones, representing the role-based access control matrix of actions and subjects. Actions can be: join, send, receive, etc.. Thus, if we have a policy of 100,000 entries, we will need roughly 100 KB - 200 KB to represent this policy in memory; we maintain the policy in memory in the form of a hash table. The other major overhead that is caused by adding RBAC is parsing the policy (written in XML format.) This can take a considerable time when adding a fresh new policy to the system. Fortunately, this is not usually the case in real life scenarios, when usually few updates on a policy are done at a time. This will make the process of updating the cache in memory an easier task. With the careful design of a caching mechanism for our RBAC framework, the overhead of updating the cache as a result of changes in the policy can be kept to a minimal level. In addition, since a hash table lookup is made for every request in the system, it is important for the lookup procedure not to impose much latency on the system. In a previous study we conducted, a hash table of size of 100,000 entries, showed only a less than 30 micro seconds lookup time.

VI. Conclusion

Based on our observation, adding RBAC functionality to a group communication system did not affect the performance of applications in a WAN scenario. This was proven by showing that the performance of the main functionalities of a GCS was not noticeably affected by the added access control mechanisms. In conclusion, we have shown that with the use of caching, the proposed RBAC framework adds minimal overhead to a GCS, while still providing the advantages of having such a framework to enable access control in social networking applications.

Acknowledgements

We would like to thank David W. Bettis for the implementation of the C library of X-RBAC that was used in this work.

References

[1] "Social 2013: The Enterprise Strikes Back," http://goo.gl/WBULv.

[2] "Financial Industry Gets New Guidance on the Use of Social Media," http://goo.gl/vbTYn.

[3] "Social Media in Healthcare: Privacy and Security Considerations," http://www.himss.org/content/files/SocialMediaWSHIMA042012LG.pdf.

[4] "Social Media Brings New Challenges to Government Security Administrators," http://soa.sys-con.com/node/2308264.

[5] "Doximity," http://www.doximity.com.

[6] C. Nita-Rotaru and N. Li, "A framework for role-based access control in group communication systems," in *ISCA PDCS*, D. A. Bader and A. A. Khokhar, Eds. ISCA, 2004, pp. 522–529.

[7] R. Bhatti, J. Joshi, E. Bertino, and A. Ghafoor, "Xml-based specification for web services document security," in *IEEE Computer, Vol 37, Issue 4*, 2004.

[8] "The Spread Toolkit," http://www.spread.org.

[9] D. Ferraiolo, R. Sandhu, S. Gavrila, R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," in *ACM TISSEC, Vol 4, Issue 3*, 2001.

[10] Y. Amir, C. Nita-Rotaru, and J. Stanton, "Framework for authentication and access control of client-server group communication systems," in *Proceedings of the Third International Workshop on Networked Group Communication, London, UK*, 2001.

[11] "PlanetLab," http://www.planet-lab.org.

[12] "Network Time Protocol," http://www.ntp.org.