Object-Tagged RBAC Model for the Hadoop Ecosystem

Maanak Gupta, Farhan Patwa, Ravi Sandhu

▶ To cite this version:

Maanak Gupta, Farhan Patwa, Ravi Sandhu. Object-Tagged RBAC Model for the Hadoop Ecosystem. 31th IFIP Annual Conference on Data and Applications Security and Privacy (DBSEC), Jul 2017, Philadelphia, PA, United States. pp.63-81, 10.1007/978-3-319-61176-1_4 . hal-01684349

HAL Id: hal-01684349

https://hal.inria.fr/hal-01684349

Submitted on 15 Jan 2018

Object-Tagged RBAC Model for the Hadoop Ecosystem

Maanak Gupta[✉], Farhan Patwa and Ravi Sandhu

Institute for Cyber Security and Department of Computer Science
University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA
gmaanakg@yahoo.com, farhan.patwa@utsa.edu, ravi.sandhu@utsa.edu

Abstract. Hadoop ecosystem provides a highly scalable, fault-tolerant and cost-effective platform for storing and analyzing variety of data formats. Apache Ranger and Apache Sentry are two predominant frameworks used to provide authorization capabilities in Hadoop ecosystem. In this paper we present a formal multi-layer access control model (called HeAC) for Hadoop ecosystem, as an academic-style abstraction of Ranger, Sentry and native Apache Hadoop access-control capabilities. We further extend HeAC base model to provide a cohesive object-tagged role-based access control (OT-RBAC) model, consistent with generally accepted academic concepts of RBAC. Besides inheriting advantages of RBAC, OT-RBAC offers a novel method for combining RBAC with attributes (beyond NIST proposed strategies). Additionally, a proposed implementation approach for OT-RBAC in Apache Ranger, is presented. We further outline attribute-based extensions to OT-RBAC.

Keywords: Access Control, Hadoop Ecosystem, Big Data, Data Lake, Role Based, Attributes, Groups Hierarchy, Object Tags

1 Introduction

Over the last few years, enterprises have started harvesting data from 'anything' to discover business and customer needs. It is estimated that 163 zettabytes of data will be generated annually by year 2025 as quoted by IDC [5]. Such massive and varied collections of data, referred to as Big Data, are considered 21st century gold for data miners. Enterprises gain useful insights from analysis to offer targeted marketing, fraud detection, accident forecasting, traffic patterns and even strong love matching. With volume, variety and velocity of data burgeoning, massive storage and compute clusters are required for analysis.

Apache Hadoop [1] has established itself as an important open-source framework for cost-efficient, distributed storage and computing of data in timely fashion. The platform offers resilient infrastructure for sophisticated analytical and pattern recognition techniques for multi-structured data. Hadoop ecosystem includes several open-source and commercial tools (Apache Hive, Apache Storm, Apache HBase, Apache Ambari etc.) built to leverage the full capabilities of Hadoop framework. These tools along with Apache Hadoop 2.x core modules (Hadoop Common, Hadoop Distributed File System (HDFS), YARN and MapReduce) empower users to harness the potential of data assets.

As Hadoop is widely used in government and private sector, its security has been a major concern and widely studied. Multi-tenant Data Lake offered by Hadoop, stores and processes sensitive information from several critical sources, such as banking and intelligence agencies, which should only be accessed by legitimate users and applications. Threats—including denial of resources, malicious user killing YARN applications, masquerading Hadoop services like NameNode, DataNode etc.—can have serious ramifications on confidentiality and integrity of data and ecosystem resources. The distributed nature and platform scale makes it more difficult to protect the infrastructure assets.

Apache Ranger [3] and Apache Sentry [4] are two important software systems used to provide fine-grained access across several Hadoop ecosystem components. In this paper we present the multi-layer access control model for Hadoop ecosystem (referred as HeAC), formalizing the authorization model in Apache Ranger (release 0.6) and Sentry (release 1.7.0) in addition to access controls capabilities in core Hadoop 2.x. We further propose an Object-Tagged Role Based Access Control (OT-RBAC) model which leverages the merits of RBAC and provides a novel approach of adding object attribute values (called object tags) in RBAC model. We also outline extensions to OT-RBAC to incorporate NIST proposed strategies [27] for adding attributes in RBAC. To our knowledge this is the first work to consider formal authorization models specific to Hadoop ecosystem.

The remainder of this paper is as follows. Section 2 discusses current authorization capabilities in Hadoop ecosystem. In Section 3, we present a formal Hadoop ecosystem access control model called HeAC. We introduce the Object-Tagged Role Based Access Control (OT-RBAC) model in Section 4, followed by proposed implementation in Section 5. In Section 6, we present attributes-based authorization extensions to OT-RBAC. Section 7 reviews previous related work, followed by Section 8 which gives our conclusions.

2 Multi-layer Authorization in Hadoop Ecosystem

The most critical assets required to be secured in Hadoop ecosystem involve services, data and service objects, applications and cluster infrastructure resources. In this section we discuss the multi-layer authorization capabilities provided in Hadoop ecosystem in line with Apache Hadoop 2.x, along with access control features offered by Apache Ranger, Apache Sentry and Apache Knox.

Service Access: The first layer of defense is provided by service level authorization which checks if a user or application is allowed to access the Hadoop ecosystem services and Hadoop core daemons. This check is done before data and service objects permissions are evaluated, thereby preventing unauthorized access early in the access request lifecycle. ACLs (Access Control Lists) are specified with users and groups to restrict access to services. For example, ACL `security.job.client.protocol.acl` is checked to allow a user to communicate with YARN ResourceManager for job submission or application status inquiry.

This layer also restricts cross-service communication to prevent malicious processes interaction with Hadoop daemon services (NameNode, ResourceManager etc.). Another ACL `security.datanode.protocol.acl` is checked for interaction between DataNodes and NameNode for heartbeat or task updates. A user making API requests to individual ecosystem services like Apache Hive, HDFS, Apache Storm etc., is restricted by implementing single gateway (e.g Apache Knox [2]) access point—which enforces policies to allow or deny users to access ecosystem services before operating on underlying objects.

Data and Service Objects Access: Hadoop Distributed File System (HDFS) enforces POSIX style model and ACLs for setting permissions on files and directories holding data. Multiple other ecosystem services require different objects to be secured. For example, Apache Hive requires table and columns, whereas Apache Kafka secures topic objects from unauthorized operations by users. Some services like Apache Hive or Apache HBase also have native access control capabilities to secure different data objects. Security frameworks like Apache Ranger or Sentry provide plugins for individual ecosystem services, where centralized policies are set for different data and service objects. In Apache Ranger, authorization policies can also be formulated on Tags, which are attribute values associated with objects. For example, a tag PII can be associated with table SSN and a policy is created for tag PII. Such tag-based policy will then control access to table SSN. Tags allow controlling access to resources along several services without need to create separate policies for individual services. It should be noted that data access allowed at one service may be restricted by permissions at underlying HDFS, thereby requiring user to have multiple object permissions at different services.

Application and Cluster Resources Access: Multi-tenant Hadoop cluster requires sharing of finite resources among several users, controlled by Apache YARN capacity (or fair) scheduler queues in Hadoop 2.x. Queue level authorization enables designated users to submit or administer applications in different queues. This restricts user from submitting applications in cluster and prevents rogue users from deleting or modifying other user applications. Further, cluster resources are not consumed by certain applications requiring more resources as queues have limited resources allocated. It should be noted that application owner and queue administrator can always kill or modify jobs in queue. These queues support hierarchical structure where permissions to parent queues descend to all child queues. Hadoop implements these authorization configurations using ACLs. Configuration file can also be associated with applications to specify users who can modify or kill an application. Access to cluster nodes can be restricted by assigning node labels. Each queue can be associated with node labels to restrict nodes where applications submitted to queues can run.

Figure 1 reflects multi-layer authorization architecture provided in Hadoop ecosystem. An authenticated user passes through several access control mechanisms to operate on objects and services in Hadoop cluster. Gateway (such as Apache Knox) offers single access point to all REST APIs and provides first layer of access control to check if services inside the cluster are allowed access by out-

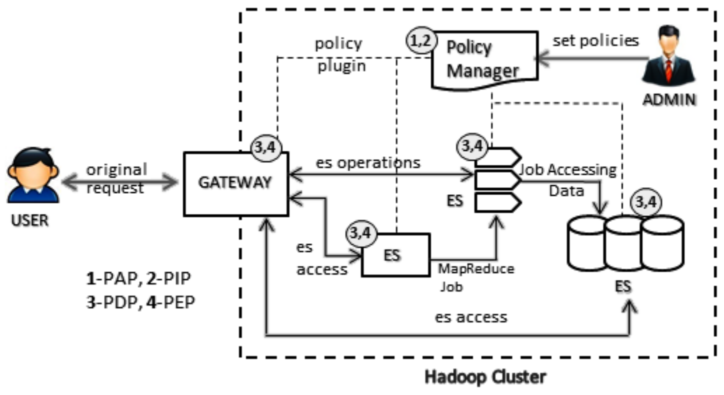


Fig. 1. Example Hadoop Ecosystem Authorization Architecture

side users. Once user is approved through the gateway policy plugin, ecosystem services apply policies cached from central policy manager to validate requests of user. User trying to access objects (files, tables etc.) in ecosystem services like HDFS or Apache Hive (shown as ES in Figure 1) is checked by policy plugins attached to the services to enforce access decisions. A user wanting to submit an application or to get submitted application status should be allowed through gateway policies to communicate with YARN ResourceManager. Apache YARN queue permissions are then checked and enforced by plugin to know if a user is allowed to submit or administer application in queues. Cross-services access (between Hadoop daemons) for information passing or task status update is mainly enforced using core Hadoop service ACLs.

As shown in Figure 1, security frameworks like Apache Ranger provides centralized Policy adminstration (1-PAP) and information point (2-PIP). Enforcement and decision points (3-PEP, 4-PDP) are plugins attached to each service which cache policies periodically from central server and enforce access decisions.

3 Hadoop Ecosystem Access Control Model

In this section we present the formal multi-layer access control model (HeAC) for Hadoop ecosystem based on Apache Hadoop 2.x. The model also covers access capabilities provided by two predominant Apache projects, Ranger (release 0.6) and Sentry (release 1.7.0). Apache Ranger supports permissions through users and groups, while Sentry assigns permissions to roles which are assigned to groups and via groups to member users. We will now discuss the formal definitions of HeAC model as specified in Table 1 and shown in Figure 2.

The basic components of HeAC include: Users (U), Groups (G), Roles (R), Subjects (S), Hadoop Services (HS), Operations (OP_{HS}) on Hadoop Services, Ecosystem Services (ES), Data and Service Objects (OB) belonging to Ecosystem Services, Operations (OP) on objects, and Object Tags (Tag).

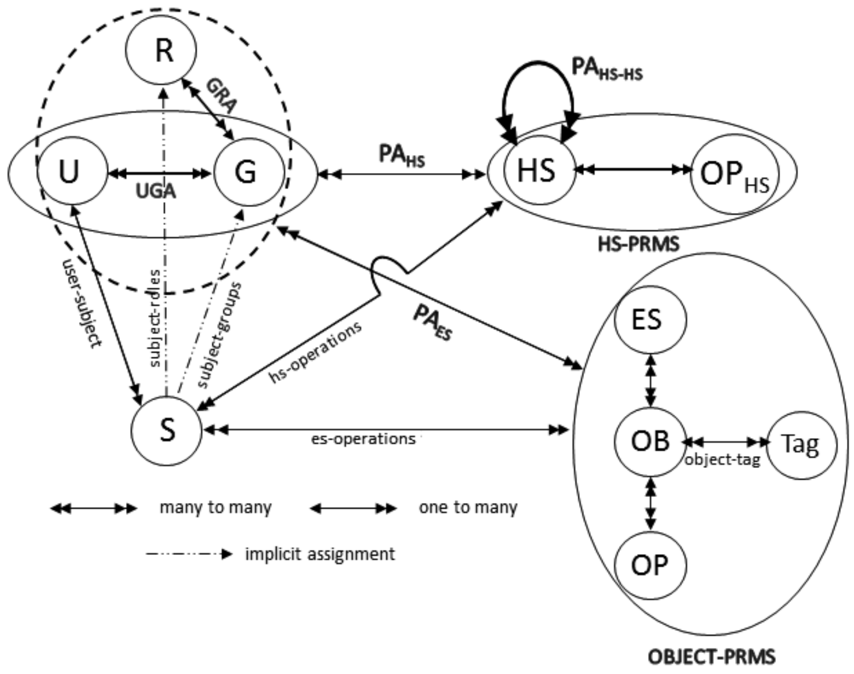


Fig. 2. A Conceptual Model of HeAC

Users, Groups, Roles and Subjects: A user is a human who interacts with computer to access services and objects inside the Hadoop ecosystem. A group is a collection of users in the system with similar organizational requirements. A role is a collection of permissions which can be assigned to different entities in the system. Permissions are assigned to users, groups or roles. In the current model roles can only be assigned to groups, thereby giving permissions to member users of groups indirectly. A subject is an application running on behalf of the user to perform operations in the Hadoop ecosystem. In HeAC model subjects always run with full permissions of the creator user.

Hadoop Services: These services are background daemon processes, like HDFS NameNode, DataNode, YARN ResourceManager, ApplicationMaster etc., which provide core functionalities in Hadoop 2.x framework. User access these services to submit applications, data block recovery or application status updates. Besides interaction with end user, these daemon services also communicate with each other for resource monitoring or task updates. It should be noted that these services do not have objects associated with them.

Operations on Hadoop Services: These are actions allowed on Hadoop services. In most cases, the general action allowed is to access a service. For example, ACL `security.client.protocol.acl` is used to determine which user is allowed to access HDFS NameNode service. These ACLs are part of Hadoop native access control capabilities (referred as service level authorization).

Ecosystem Services: Data and objects inside the Hadoop ecosystem are accessed through different platforms which we consider as Ecosystem Services. Example of such services include Apache HDFS, Apache Hive, Apache HBase, Apache Storm, Apache Kafka etc. These services can either have data objects (tables, columns) or other type of resources (queues, topics) which they support. Access to the ecosystem services is first required before operation on supported objects. We consider Data Services as one instance of Ecosystem Services.

Data and Service Objects: Ecosystem services manage different types of resources (objects) inside the cluster. For example, Apache HDFS supports files and directories, while Apache HBase has data objects like column-family, cells etc. YARN manages queue objects and Apache Solr has collections. These are resources which are protected from unauthorized operations from users.

Operations on objects: Multiple data and service objects support different operations to perform actions on them. Apache Hive has select, create, drop, alter for tables and columns while Apache HBase data objects (column family, column) support read, write, create etc. YARN queues have operations to submit applications or administer the queue.

Object Tags: Objects inside ecosystem can be assigned attributes based on sensitivity, content or expiration date. Such classification is done using attribute values called Tags. An object can have multiple tags associated with it and vice versa. For example, PII tag can be attached to sensitive data table SSN.

As shown in Table 1, a user can be assigned to multiple groups defined by directUG function. Groups are also assigned to multiple roles as reflected by function directGR. Relation object-tag denotes a many-to-many relation between objects and associated attribute values called tags. Hadoop ecosystem has two different sets of permissions to perform actions on services and objects. OBJECT-PRMS is the set of data and service object permissions which is power set of the cross product of ecosystem services (ES), objects (OB) or object tags (Tag), and operations (OP). Here permissions can be set either on object or object tags, and policies can allow or deny operations on the object based on its associated tags or the object itself. OBJECT-PRMS also include ecosystem service as part of permission thereby taking into account the requirement of service access before object operations. Another set of permissions called Hadoop service permissions (HS-PRMS) is the power set of the cross product of HS and $\mathrm{OP}_{\mathrm{HS}}$. These are required for application submission or other non-data or object operations. Depending on the type of operations to be performed, a user may require either one or both type of permissions.

A many-to-many relation $\mathrm{PA}_{\mathrm{HS}}$ specifies the assignment of HS-PRMS to users or groups. In this way a user can be assigned HS-PRMS directly or through group membership. OBJECT-PRMS can be assigned to users, groups or roles (shown by $\mathrm{PA}_{\mathrm{ES}}$). A group can get the object permissions directly or through roles, which will then enable it to the member users. It should be noted that a user may need multiple data object permissions across several data services to operate on a data object. For example, in case of Apache Hive table, besides having permission on the table, a user may be required to have permissions on

Table 1. Hadoop Ecosystem Access Control (HeAC) Model Definitions

Basic Sets and Functions

– U, G, R, S (finite set of users, groups, roles and subjects respectively)
– HS, OP_{HS} (finite set of Hadoop services and operations respectively)
– ES, OB (finite set of ecosystem services and objects respectively)
– OP, Tag (finite set of object operations and object tags respectively)
– directUG : $U \rightarrow 2^{G}$, mapping each user to a set of groups, equivalently UGA $\subseteq U \times G$
– directGR : $G \rightarrow 2^{R}$, mapping each group to a set of roles, equivalently GRA $\subseteq G \times R$
– object-tag $\subseteq$ OB$\times$Tag, relation between object and object tags
– OBJECT-PRMS = $2^{ES\times(OB \cup Tag)\times OP}$, set of data and service object permissions
– HS-PRMS = $2^{HS\times OP_{HS}}$, set of Hadoop services permissions

Permission Assignments

– $PA_{HS} \subseteq (U \cup G)\times$HS-PRMS, mapping entities to Hadoop service permissions. Alternatively, $hs_{prms} : (x) \rightarrow 2^{\text{HS-PRMS}}$, defined as $hs_{prms}(x) = \{p \mid (x,p) \in PA_{HS},\ x \in (U \cup G)\}$
– $PA_{ES} \subseteq (U \cup G \cup R)\times$OBJECT-PRMS, mapping entities to object permissions. Alternatively, $es_{prms} : (x) \rightarrow 2^{\text{OBJECT-PRMS}}$, defined as $es_{prms}(x) = \{p \mid (x,p) \in PA_{ES},\ x \in (U \cup G \cup R)\}$

Hadoop Cross Services Access

– $PA_{HS\text{-}HS} \subseteq HS \times$ HS-PRMS, mapping Hadoop service to Hadoop service access.
Alternatively, $hs\text{-}hs_{prms} : (hs{:}HS) \rightarrow 2^{\text{HS-PRMS}}$, defined as
$hs\text{-}hs_{prms}(hs) = \{p \mid (hs,p) \in PA_{HS\text{-}HS}\}$

Effective Roles of Users (Derived Functions)

- effectiveR : $U \rightarrow 2^{R}$, defined as $\text{effectiveR}(u) = \bigcup_{\forall g \in \text{directUG}(u)} \text{directGR}(g)$

Effective Permissions of User

- $\text{effectiveHS}_{prms} : U \rightarrow 2^{\text{HS-PRMS}}$, defined as
$\text{effectiveHS}_{prms}(u) = hs_{prms}(u) \cup \bigcup_{\forall g \in \{\text{directUG}(u)\}} hs_{prms}(g)$
- $\text{effectiveES}_{prms} : U \rightarrow 2^{\text{OBJECT-PRMS}}$, defined as
$\text{effectiveES}_{prms}(u) = es_{prms}(u) \cup \bigcup_{\forall x \in \{\text{directUG}(u)\ \cup\ \text{effectiveR}(u)\}} es_{prms}(x)$

User Subject

- userSub : S$\rightarrow$ U, mapping each subject to its creator user, where the subject gets all the permissions of the creator user.

Ecosystem Service Object Operation Decision

A subject s $\in$ S is allowed to perform an operation op $\in$ OP on an object ob $\in$ OB in ecosystem service es $\in$ ES if the effective object permissions of userSub(s) include permissions for object ob or for tag t $\in$ Tag associated with object ob. Formally,
$(es,ob,op) \in \text{effectiveES}_{prms}(\text{userSub}(s)) \vee$
$(\exists\ t)\ [(ob,t) \in \text{object-tag} \wedge (es,t,op) \in \text{effectiveES}_{prms}(\text{userSub}(s))]$

the underlying data file in HDFS. $PA_{HS\text{-}HS}$ encapsulates the access requirement between several Hadoop services inside the cluster for task updates or resource monitoring (e.g. communication between DataNodes and NameNode). The effective roles of user are covered by effectiveR which is union of roles assigned to all member groups. The effective permissions on Hadoop services attained by user (reflected by effectiveHS$_{\text{prms}}$) is the direct permissions on HS and permissions inherited through group membership. The final set of ES object permissions for a user is union of direct permission and permissions assigned through group membership and effective roles as shown in effectiveES$_{\text{prms}}$.

A subject is created by a user as expressed by userSub. It inherits all the permissions assumed by the user to perform actions. In last section of Table 1, a subject is allowed to perform operations on objects in ES service depending on either direct permission on objects or permission on tags associated with objects.

It should be noted that Apache Ranger provides context enrichers, which are used to add contextual information to user request based on location, IP address or other attribute. We treat such information as environment attributes and include these in attribute-based model in Section 6. It should also be mentioned that data ingestion into Hadoop cluster is beyond the scope of this paper and for the access control points discussed, we assume data already present inside the cluster. Further, we ignore deny access request, for the sake of simplicity.

4 Object-Tagged RBAC for Hadoop Ecosystem

In this section we propose Object-Tagged Role-Based Access Control model for the Hadoop Ecosystem, which we denote as OT-RBAC. With respect to HeAC model, this model assigns both objects and Hadoop service permissions to users only through roles, consistent with the basic principle of RBAC. The model presents a novel approach for combining attributes and RBAC [33] besides NIST proposed approaches (i.e., Dynamic Roles, Attribute-Centric and Role Centric) [27]. Hence the convenient administrative benefits of RBAC, along with a finer-grained attributes authorization, are incorporated in this model.

The conceptual model for OT-RBAC is shown in Figure 3 followed by formal definitions in Table 2. The remainder of this section discusses the new and modified components introduced in OT-RBAC model (marked ** and †† respectively) with respect to HeAC model. In OT-RBAC model users are directly assigned to multiple roles specified by function directUR. Group hierarchy (GH) is introduced into the system, defined by a partial order relation on G and written as $\succeq_g$. The inheritance of roles is from low to high, i.e., $g_1 \succeq_g g_2$ means g_1 inherits roles from g_2. In such cases, we say g_1 is the senior group and g_2 is the junior group. The HS-PRMS and OBJECT-PRMS permissions are assigned to roles only, specified by many-to-many relations PA$_{\text{HS}}$ and PA$_{\text{ES}}$ respectively. This is modified with respect to the original HeAC, where HS-PRMS were assigned to users or groups and OBJECT-PRMS to user, groups or roles also. This reflects the advantage of RBAC model where permissions are allotted or removed from users by granting or revoking their roles. Both OBJECT-PRMS and HS-PRMS can be assigned to same role in the Hadoop ecosystem. With group hierarchy (GH), the effective roles of a group (expressed by effectiveGR) is the union of direct roles assigned to group and effective roles of all its junior groups. It should

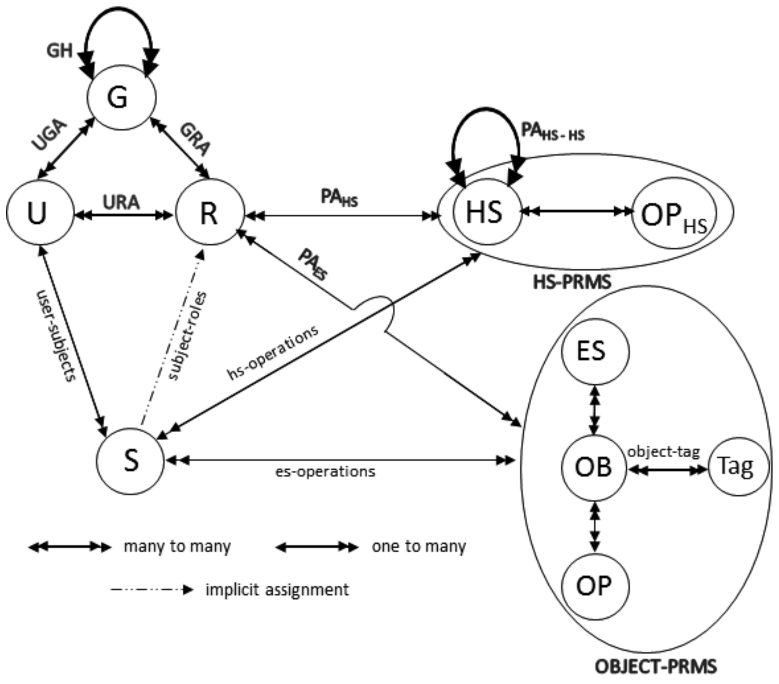


Fig. 3. Conceptual OT-RBAC Model for Hadoop Ecosystem

be noted that this definition is recursive where the junior-most groups have same direct and effective roles. The effective roles of the user (defined by effectiveR) is then the union of direct user roles and effective roles of the groups to which the user is directly assigned. For example, assuming group Grader is assigned roles Student and Graduate and a senior group TA is assigned to role Doctoral. Then the effective roles of group TA would be Student, Graduate and Doctoral. A user u_1 can be directly assigned to role Staff. If u_1 also becomes a member of group TA, u_1 has the effective roles of Student, Graduate, Doctoral and Staff. The important advantage of user group membership is convenient assignment and removal of multiple roles from users with single administrative operation.

A subject S (similar to sessions in RBAC [17]) created by the user can have some or all of the effective roles of the creator user. The effective permissions available to a subject (expressed by $\text{effectiveES}_{\text{prms}}$ and $\text{effectiveHS}_{\text{prms}}$) will then be the object and Hadoop service permissions assigned to all the effective roles activated by the subject. A subject might need to have multiple permissions to access different services or objects inside Hadoop ecosystem which may result in requiring multiple roles. The prime advantage of OT-RBAC model over HeAC model is the assignment of permissions only to roles instead of assigning directly to users and groups. Further it introduces the concept of group hierarchy which results in roles inheritance and eases administrative responsibilities of the security administrator. Also including group hierarchy makes OT-RBAC model

Table 2. Formal OT-RBAC Model Definitions

Basic Sets and Functions

– U, G, R, S (finite set of users, groups, roles and subjects respectively)
– HS, OP_{HS} (finite set of Hadoop services and operations respectively)
– ES, OB (finite set of ecosystem services and objects respectively)
– OP, Tag (finite set of object operations and object tags respectively)
– directUG : $U \rightarrow 2^{G}$, mapping each user to a set of groups, equivalently UGA $\subseteq$ U $\times$ G
– **directUR : $U \rightarrow 2^{R}$, mapping each user to a set of roles, equivalently URA $\subseteq$ U $\times$ R
– directGR : $G \rightarrow 2^{R}$, mapping each group to a set of roles, equivalently GRA $\subseteq$ G $\times$ R
– **GH $\subseteq$ G$\times$G, a partial order relation $\succeq_g$ on G
– object-tag $\subseteq$ OB$\times$Tag, relation between object and object tags
– OBJECT-PRMS = $2^{ES\times(OB \cup Tag)\times OP}$, set of data and service object permissions
– HS-PRMS = $2^{HS\times OP_{HS}}$, set of Hadoop services permissions

††**Role Permission Assignments**

– $PA_{HS} \subseteq$ R$\times$HS-PRMS, mapping roles to Hadoop service permissions. Alternatively, hs_{prms} : (r:R) $\rightarrow 2^{HS\text{-}PRMS}$, defined as $hs_{prms}(r) = \{p \mid (r,p) \in PA_{HS}\}$
– $PA_{ES} \subseteq$ R$\times$OBJECT-PRMS, mapping roles to object permissions. Alternatively, es_{prms} : (r:R) $\rightarrow 2^{OBJECT\text{-}PRMS}$, defined as $es_{prms}(r) = \{p \mid (r,p) \in PA_{ES}\}$

Hadoop Cross Services Access

– $PA_{HS\text{-}HS} \subseteq$ HS $\times$ HS-PRMS, mapping Hadoop service to Hadoop service access. Alternatively, hs-hs_{prms} : (hs:HS) $\rightarrow 2^{HS\text{-}PRMS}$, defined as hs-$hs_{prms}(hs) = \{p \mid (hs,p) \in PA_{HS\text{-}HS}\}$

††**Effective Roles of Users (Derived Functions)**

- effectiveGR : $G \rightarrow 2^{R}$, defined as
  $\text{effectiveGR}(g_i) = \text{directGR}(g_i) \cup (\bigcup_{\forall g \in \{g_j \mid g_i \succeq_g g_j\}} \text{effectiveGR}(g))$
- effectiveR : $U \rightarrow 2^{R}$, defined as
  $\text{effectiveR}(u) = \text{directUR}(u) \cup (\bigcup_{\forall g \in \text{directUG}(u)} \text{effectiveGR}(g))$

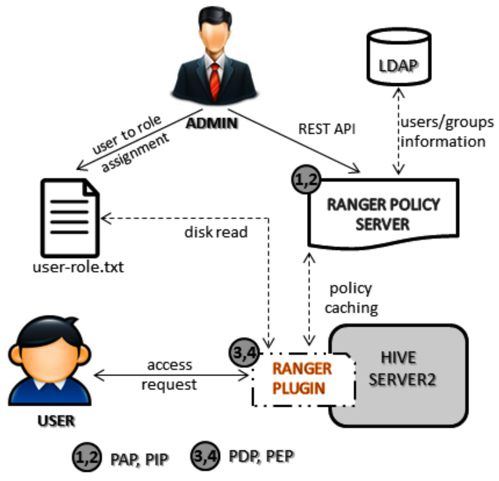
††**Effective Roles and Permissions of Subjects**

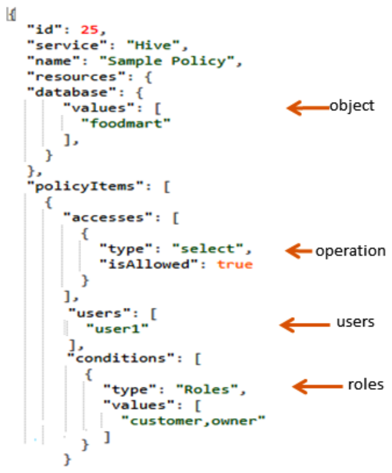
- userSub : S$\rightarrow$ U, mapping each subject to its creator user
- effectiveR : $S \rightarrow 2^{R}$, mapping of subject s to a set of roles. It is required that : effectiveR(s) $\subseteq$ effectiveR(userSub(s))
- $\text{effectiveHS}_{prms}$: $S \rightarrow 2^{HS\text{-}PRMS}$, defined as $\text{effectiveHS}_{prms}(s) = \bigcup_{\forall r \in \text{effectiveR}(s)} hs_{prms}(r)$
- $\text{effectiveES}_{prms}$: $S \rightarrow 2^{OBJECT\text{-}PRMS}$, defined as $\text{effectiveES}_{prms}(s) = \bigcup_{\forall r \in \text{effectiveR}(s)} es_{prms}(r)$

Ecosystem Service Object Operation Decision

A subject s $\in$ S is allowed to perform an operation op $\in$ OP on an object ob $\in$ OB in ecosystem service es $\in$ ES if the effective object permissions of subject s include permissions to object ob or to tag t $\in$ Tag associated with object ob. Formally,
$(es,ob,op) \in \text{effectiveES}_{prms}(s) \vee$
$(\exists\, t)\ [(ob,t) \in \text{object-tag} \wedge (es,t,op) \in \text{effectiveES}_{prms}(s)]$

** and †† highlight new and modified components respectively with respect to HeAC





```
{
  "id": 25,
  "service": "Hive",
  "name": "Sample Policy",
  "resources": {
  "database": {
      "values": [
        "foodmart"
      ],
    }
  },
  "policyItems": [
    {
      "accesses": [
        {
          "type": "select",
          "isAllowed": true
        }
      ],
       "users": [
         "user1"
       ],
      "conditions": [
        {
          "type": "Roles",
          "values": [
            "customer,owner"
          ]
        }
      }
```

Fig. 4. Proposed Implementation in Apache Ranger and Sample JSON Policy

easier to fit into attributes based models where role is one of the other attributes. In such case group hierarchy can be very useful in attributes inheritance offering convenient administration by assigning or removing multiple attributes to users with single administrative operation [35].

The proposed OT-RBAC model presents a novel approach for adding attributes to RBAC (besides NIST strategies [27]), by introducing object tags. The model represents object permissions (OBJECT-PRMS) as union of permissions on attribute values (reflected as tags) associated with objects and regular permissions as discussed in RBAC [33]. In the following section, we propose an implementation approach for OT-RBAC using open-source Apache Ranger.

5 Proposed Implementation

One approach to implement OT-RBAC model is by extending open-source Apache Ranger which provides centralized security administration to multiple Hadoop ecosystem services. It offers REST API to create security policies which are enforced using plugins appended to each secured service. These plugins intercept a user access request, and check against policies cached sporadically from policy server to make access decisions. Apache Ranger 0.5 and above provide extensible framework to add new authorization functionalities by offering context enricher and condition evaluator hooks. Context enricher is a Java class which appends user access request with additional information used for policy evaluation. Condition evaluator enables a security architect to add custom conditions to policies. These hooks can be used to extend plugins to enforce OT-RBAC.

Proposed Apache Ranger architecture for Hive service authorization is shown in Figure 4. Users and groups are stored in LDAP, which are synced to Ranger policy manager to create policies. A text file is added which stores current users

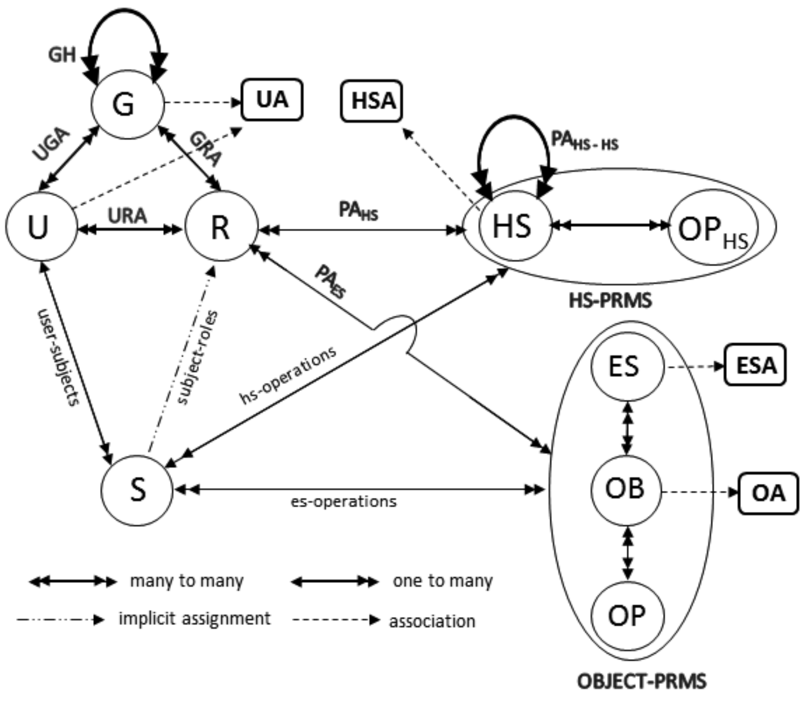
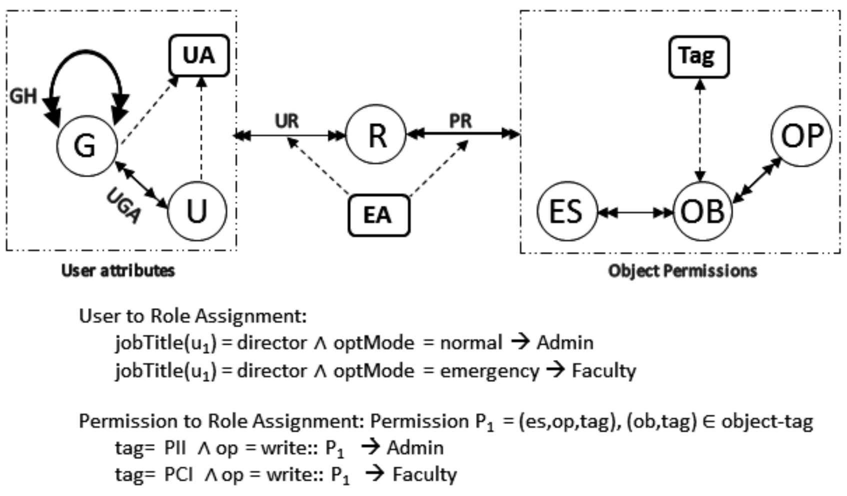


Fig. 5. Adding Attributes to OT-RBAC Model

to roles assignment. This file is used by context enricher implemented, to add roles of user to access request along with objects and actions. A condition evaluator should also be implemented to include roles in policy used for evaluation. A sample policy in JSON format is shown in Figure 4. This policy includes roles in condition which specifies the roles allowed to perform select operation on table foodmart. Hive service definition should be updated with new context and condition hooks information using REST API. Access decision and enforcement is done in Ranger plugin embedded with Hive service whereas policy administration and information is through central policy server as shown in Figure 4. Similar implementation approach can be adopted in other ecosystem services also. This proposed implementation requires roles addition at two places, one in text file and other in policy conditions which requires extra effort by administrator.

6 Attributes Based Extensions to OT-RBAC

We outline some approaches for adding attributes in OT-RBAC model to achieve finer-grained access control. OT-RBAC model incorporates tags for objects, which is further generalized by introducing set of object attributes along with attributes for other entities. As shown in Figure 5, UA is a set of attributes for users and groups, and OA is a set of attributes for data and service objects. HSA



User to Role Assignment:
  jobTitle(u_1) = director ∧ optMode = normal → Admin
  jobTitle(u_1) = director ∧ optMode = emergency → Faculty

Permission to Role Assignment: Permission P_1 = (es,op,tag), (ob,tag) ∈ object-tag
  tag= PII ∧ op = write:: P_1 → Admin
  tag= PCI ∧ op = write:: P_1 → Faculty

Fig. 6. Dynamic Roles and Object Permissions in OT-RBAC

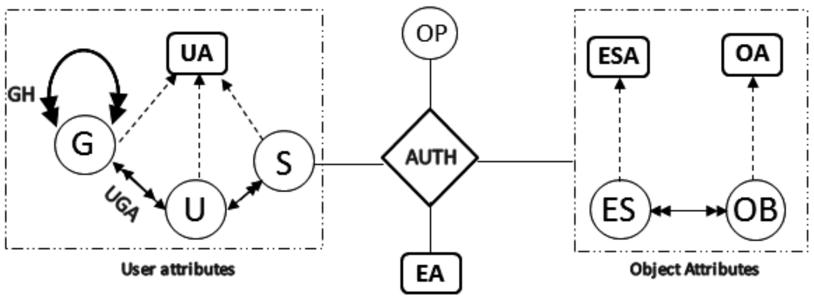
and ESA are set of attributes for HS and ES. An attribute is a function which takes as input an entity and returns values from a specified range [24]. Attribute-based authorization policies are used to determine access permissions of users on services and objects. With group hierarchy, senior groups inherit attributes from junior groups [20], and a user assigned to senior groups gets all attributes of group besides its direct attributes. A set of environment attributes is also added to incorporate contextual information (like access time, threat level) in policies.

We outline how an attribute enhanced OT-RBAC model, along the lines of Figure 5, can incorporate NIST proposed strategies [27] for adding attributes in RBAC, i.e., Dynamic Roles, Attribute Centric and Role Centric. We discuss these in context of objects permissions assignment. These approaches can be similarly applied to Hadoop services permissions assignment also.

6.1 Dynamic Roles

Dynamic Roles approach considers user and environment attributes to determine roles of a user. This automated approach require rules defined using a policy language [8] composed of attributes and resulting roles. The roles of the user will change based on the user's current attributes as well as current environment attributes. As shown in Figure 6, OT-RBAC model can be configured to achieve dynamic roles assignment to users based on the direct or inherited attributes through group memberships [20]. We can further extend the use of attributes for dynamic permissions assignment to roles based on object tags, environment attribute values and operations.

As in Figure 6, user u_1 with attribute jobTitle value director and environment attribute optMode value normal can be assigned Admin role, which can change to



Authorization$_{write}$ (s:S, es:ES, ob:OB) :: effective$_{jobTitle}$(s) = director ∧ access(s,es) = True ∧ name(es) = hdfs ∧ tag(ob)= PII ∧ optMode = normal

Authorization$_{read}$ (s:S, es:ES, ob:OB) :: effective$_{jobTitle}$(s) = professor ∧ access(s,es) = True ∧ name(es) = hdfs ∧ tag(ob)= PCI ∧ optMode = emergency.

Fig. 7. Attribute Centric approach in OT-RBAC

role Faculty when attribute optMode changes to emergency. Similarly, permission containing operation write on object ob with tag value PII can be assigned to role Admin which can change to role Faculty when tag changes to PCI.

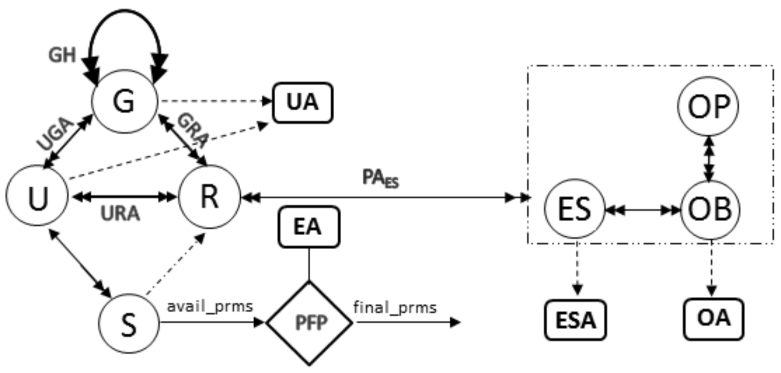
6.2 Attribute Centric

In this approach, access decision is based on attributes of entities (role is also an attribute) where authorization policies comprise attributes of subjects, objects or environment [23, 24, 42]. To configure OT-RBAC with attribute centric strategy, boolean authorization functions are defined using propositional logic formula for each operation in OP which specify policy if subject s can perform operation op on object ob in ecosystem service es under some environment attributes.

As shown in Figure 7, authorization policy is defined stating that subject s with effective attribute jobTitle value director is allowed to perform write on object ob with attribute tag value PII in ecosystem service es with name hdfs when environment attribute optMode is normal. It should be noted that object ob must belong to ecosystem service es and subject must be allowed to access es (expressed by access(s,es)) before performing any operation on object in es. Similar authorization policy for read operation can be defined by administrators.

6.3 Role Centric

In this approach the maximum permissions (avail_prms) are assigned to user through roles assignment (similar to RBAC [33]) but the final set of permissions (final_prms) is dependent on the attributes of entities. Permission Filtering boolean functions are defined based on the attributes, which are checked for each permission in avail_prms set available to users via roles, to determine the final_prms set assigned to the users as discussed in [25].



FAdmin1(s:S, es:ES, ob:OB, write) ::
jobTitle(subUser(s)) = director ∧ optMode = normal

FAdmin2(s:S, es:ES, ob:OB, read) ::
jobTitle(subUser(s)) = faculty ∧ tag(ob) = PCI

Fig. 8. Role Centric approach in OT-RBAC

Assume user u_1 assigned to role Admin then u_1 gets permissions (avail_prms) of writing to hdfs service file customer and reading a file having PII tag. These permissions are checked against filter functions selected using target functions discussed in [25]. As shown in Figure 8, filter function FAdmin1 is invoked to check if first permission is in final_prms set. The function checks if creator user of s has jobTitle attribute with value director and optMode is normal to avail this permission. If it returns true, the permission will be included in final set (final_prms). Similar filter function can be called for other permissions also.

7 Related Work

Several papers [6, 7, 14, 19, 32, 36, 43] discuss security threats and solutions in Hadoop ecosystem. Recently, Gupta et al [18] presented a multi-layer authorization framework for Hadoop ecosystem, which covers several access control enforcement points and demonstrates their application using Apache Ranger. Access control using cryptography based on proxy re-encryption [31] provides approach for delegated access to Hadoop cluster. A security model for G-Hadoop framework using public key and SSL is presented in [46]. Security and privacy concerns of MapReduce applications are discussed in [15]. Ulusoy et al [39, 40] proposed approaches for fine grained access control for MapReduce systems. Privacy issues in Big Data are addressed in [13, 29, 37, 38].

Risk aware information disclosure in [9] can be used for Hadoop Data lake. Secure information access model via data services [11] can be applied for Hadoop data services. HDFS can use data access protection using data distribution and swapping in [16]. Vimercati et al [41] discuss confidentiality of outsourced data.

Colombo et al [12] also proposed fine-grained context-aware access control features for MongoDB NoSQL datastore.

Risk based access using classification [10] studies role assignment based on risk factors. Contextual attributes in location aware ABAC in [21] can be applied in Hadoop. Classification of data object based on content is presented in [44]. Policy engineering for ABAC [26] can be used to define values based on risk or context. Another promising approach in attribute based data sharing has been presented in [45]. Use of role mining in [28] can be extended to determine roles of users based on attributes. A research roadmap on trust and Big Data is presented in [34]. Trust based Data ingestion or processing can use models in [30].

Hu et al [22] presented a general access control model for big data processing frameworks. The paper introduces chain of trust among several entities to authorize access request. The work provides a preliminary document which can be conceptualized to specific systems like Hadoop. However, the authors do not address details particular to the Hadoop ecosystem.

8 Conclusion and Future Work

In this paper we present first formalized access control model called HeAC for Hadoop ecosystem. Besides the regular permissions including objects and operations, this model also includes object attribute values (represented as tags) in object permissions. We further extended HeAC model to propose Object-Tagged RBAC model (OT-RBAC) which preserves role based permission assignment and presents a novel approach for adding object attributes to RBAC. We proposed an implementation approach for introducing roles in open-source Apache Ranger using context enricher and condition evaluators. We additionally draft some extensions to OT-RBAC by adding attributes to provide fine grained access policies. We outline OT-RBAC model to support NIST strategies for including attributes using Dynamic Roles, Attribute Centric and Role Centric.

For future work, we plan to develop pure attribute based access control models for fine grained access to Hadoop ecosystem resources. Also, since the Hadoop data lake is used by multiple tenants it would be interesting to introduce data ingestion security into the system to secure data at HDFS data nodes level.

Acknowledgement

Sincere gratitude is extended to James Benson, Technology Research Analyst at Institute for Cyber Security, UTSA, for his useful comments. This research is partially supported by NSF Grants CNS-1111925, CNS-1423481, CNS-1538418, DoD ARL Grant W911NF-15-1-0518 and by The Texas Sustainable Energy Research Institute at University of Texas at San Antonio.

References

1. Apache Hadoop, http://hadoop.apache.org/
2. Apache Knox, https://knox.apache.org/
3. Apache Ranger, http://ranger.apache.org/
4. Apache Sentry, https://sentry.apache.org/
5. Data Age 2025: The Evolution of Data to Life-Critical, https://www.idc.com/
6. Big Data: Securing Intel IT's Apache Hadoop Platform (2016), http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/big-data-securing-intel-it-apache-hadoop-platform-paper.pdf
7. Securing Hadoop: Security Recommendations for Hadoop Environments (2016), https://securosis.com/assets/library/reports/Securing_Hadoop_Final_V2.pdf
8. Al-Kahtani, M.A., Sandhu, R.: A model for attribute-based user-role assignment. In: Proc. of IEEE ACSAC. pp. 353–362 (2002)
9. Armando, A., Bezzi, M., Metoui, N., Sabetta, A.: Risk-based privacy-aware information disclosure. IJSSE 6(2), 70–89 (2015)
10. Badar, N., Vaidya, J., Atluri, V., Shafiq, B.: Risk based access control using classification. In: Automated Security Management, pp. 79–95. Springer (2013)
11. Barhamgi, M., Benslimane, D., Oulmakhzoune, S., Cuppens-Boulahia, N., Cuppens, F., Mrissa, M., Taktak, H.: Secure and privacy-preserving execution model for data services. In: Proc. of CAISE. pp. 35–50. Springer (2013)
12. Colombo, P., Ferrari, E.: Complementing mongodb with advanced access control features: Concepts and research challenges. In: Proc. of SEBD 2015 (2015)
13. Colombo, P., Ferrari, E.: Privacy aware access control for big data: a research roadmap. Big Data Research 2(4), 145–154 (2015)
14. Das, D., OMalley, O., Radia, S., Zhang, K.: Adding security to Apache Hadoop. Hortonworks, IBM (2011)
15. Derbeko, P., Dolev, S., Gudes, E., Sharma, S.: Security and privacy aspects in mapreduce on clouds: A survey. Computer Science Review 20, 1–28 (2016)
16. Di Vimercati, S.D.C., Foresti, S., Paraboschi, S., Pelosi, G., Samarati, P.: Protecting access confidentiality with data distribution and swapping. In: Proc. of IEEE BdCloud. pp. 167–174 (2014)
17. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM TISSEC 4(3), 224–274 (2001)
18. Gupta, M., Patwa, F., Benson, J., Sandhu, R.: Multi-Layer Authorization Framework for a Representative Hadoop Ecosystem Deployment. In: Proc. of ACM SACMAT (To appear). 8 Pages. (2017)
19. Gupta, M., Patwa, F., Sandhu, R.: POSTER: Access Control Model for the Hadoop Ecosystem. In: Proc. of ACM SACMAT (To appear). 3 Pages. (2017)
20. Gupta, M., Sandhu, R.: The GURA$_G$ administrative model for user and group attribute assignment. In: Proc. of NSS. pp. 318–332. Springer (2016)
21. Hsu, A.C., Ray, I.: Specification and enforcement of location-aware attribute-based access control for online social networks. In: Proc. of ACM ABAC'16. pp. 25–34 (2016)
22. Hu, V.C., Grance, T., Ferraiolo, D.F., Kuhn, D.R.: An access control scheme for big data processing. In: Proc. of IEEE CollaborateCom. pp. 1–7 (2014)
23. Hu, V.C., Kuhn, D.R., Ferraiolo, D.F.: Attribute-based access control. IEEE Computer (2), 85–88 (2015)
24. Jin, X., Krishnan, R., Sandhu, R.: A unified attribute-based access control model covering DAC, MAC and RBAC. In: Proc. of IFIP DBSec. pp. 41–55. Springer (2012)

25. Jin, X., Sandhu, R., Krishnan, R.: RABAC: role-centric attribute-based access control. In: Proc. of MMM-ACNS. pp. 84–96. Springer (2012)
26. Krautsevich, L., Lazouski, A., Martinelli, F., Yautsiukhin, A.: Towards attribute-based access control policy engineering using risk. In: Proc. of RISK. pp. 80–90. Springer (2013)
27. Kuhn, D.R., Coyne, E.J., Weil, T.R.: Adding attributes to role-based access control. IEEE Computer 43(6), 79–81 (2010)
28. Lu, H., Hong, Y., Yang, Y., Duan, L., Badar, N.: Towards user-oriented RBAC model. Journal of Computer Security 23(1), 107–129 (2015)
29. Lu, R., Zhu, H., Liu, X., Liu, J.K., Shao, J.: Toward efficient and privacy-preserving computing in big data era. IEEE Network 28(4), 46–50 (2014)
30. Moyano, F., Fernandez-Gago, C., Lopez, J.: A conceptual framework for trust models. In: Proc. of TrustBus. pp. 93–104. Springer (2012)
31. Nunez, D., Agudo, I., Lopez, J.: Delegated access for hadoop clusters in the cloud. In: Proc. of IEEE CloudCom. pp. 374–379 (2014)
32. OMalley, O., Zhang, K., Radia, S., Marti, R., Harrell, C.: Hadoop security design. Yahoo, Inc., Tech. Rep (2009)
33. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
34. Sänger, J., Richthammer, C., Hassan, S., Pernul, G.: Trust and big data: A roadmap for research. In: Proc. of IEEE DEXA. pp. 278–282. IEEE (2014)
35. Servos, D., Osborn, S.L.: HGABAC: Towards a Formal Model of Hierarchical Attribute-Based Access Control. In: International Symposium on Foundations and Practice of Security. pp. 187–204. Springer (2014)
36. Sharma, P.P., Navdeti, C.P.: Securing big data Hadoop: a review of security issues, threats and solution. IJCSIT 5 (2014)
37. Soria-Comas, J., Domingo-Ferrer, J.: Big data privacy: challenges to privacy principles and models. Data Science and Engineering 1(1), 21–28 (2016)
38. Tene, O., Polonetsky, J.: Big data for all: Privacy and user control in the age of analytics. Nw. J. Tech. & Intell. Prop. 11, xxvii (2012)
39. Ulusoy, H., Colombo, P., Ferrari, E., Kantarcioglu, M., Pattuk, E.: GuardMR: Fine-grained security policy enforcement for MapReduce systems. In: Proc. of ACM ASIACCS. pp. 285–296 (2015)
40. Ulusoy, H., Kantarcioglu, M., Pattuk, E., Hamlen, K.: Vigiles: Fine-grained access control for mapreduce systems. In: Proc. of IEEE Big Data Congress. pp. 40–47 (2014)
41. Vimercati, S.D.C.D., Foresti, S., Paraboschi, S., Pelosi, G., Samarati, P.: Shuffle index: efficient and private access to outsourced data. ACM TOS 11(4), 19 (2015)
42. Wang, L., Wijesekera, D., Jajodia, S.: A logic-based framework for attribute based access control. In: Proc. of ACM FMSE. pp. 45–55 (2004)
43. White, T.: Hadoop: The Definitive Guide. O'Reilly Media, Inc. (2012)
44. Wrona, K., Oudkerk, S., Armando, A., Ranise, S., Traverso, R., Ferrari, L., McEvoy, R.: Assisted content-based labelling and classification of documents. In: Proc. of IEEE ICMCIS. pp. 1–7 (2016)
45. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: Proc. of ACM ASIACCS. pp. 261–270 (2010)
46. Zhao, J., Wang, L., Tao, J., Chen, J., Sun, W., Ranjan, R., Kołodziej, J., Streit, A., Georgakopoulos, D.: A security framework in G-Hadoop for big data computing across distributed cloud data centres. JCSS 80(5), 994–1007 (2014)