An Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem

Maanak Gupta, Farhan Patwa and Ravi Sandhu
Institute for Cyber Security (ICS),
Center for Security and Privacy Enhanced Cloud Computing (C-SPECC),
Department of Computer Science, University of Texas at San Antonio
Email: gmaanakg@yahoo.com, farhan.patwa@utsa.edu, ravi.sandhu@utsa.edu

ABSTRACT

Apache Hadoop is a predominant software framework for distributed compute and storage with capability to handle huge amounts of data, usually referred to as Big Data. This data collected from different enterprises and government agencies often includes private and sensitive information, which needs to be secured from unauthorized access. This paper proposes extensions to the current authorization capabilities offered by Hadoop core and other ecosystem projects, specifically Apache Ranger and Apache Sentry. We present a fine-grained attribute-based access control model, referred as HeABAC, catering to the security and privacy needs of multi-tenant Hadoop ecosystem. The paper reviews the current multi-layered access control model used primarily in Hadoop core (2.x), Apache Ranger (version 0.6) and Sentry (version 1.7.0), as well as a previously proposed RBAC extension (OT-RBAC). It then presents a formal attribute-based access control model for Hadoop ecosystem, including the novel concept of cross Hadoop services trust. It further highlights different trust scenarios, presents an implementation approach for HeABAC using Apache Ranger and, discusses the administration requirements of HeABAC operational model. Some comprehensive, real-world use cases are also discussed to reflect the application and enforcement of the proposed HeABAC model in Hadoop ecosystem.

CCS CONCEPTS

• **Security and privacy → Access control**; **Authorization**; **Security requirements**; *Formal security models*;

KEYWORDS

Access Control; Hadoop Ecosystem; Big Data; Data Lake; Role Based; Attributes Based; Authorization; Trust;

ACM Reference Format:
Maanak Gupta, Farhan Patwa and Ravi Sandhu. 2018. An Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem. In *ABAC'18: 3rd ACM Workshop on Attribute-Based Access Control, March 19–21, 2018, Tempe, AZ, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3180457.3180463

ABAC'18, March 19–21, 2018, Tempe, AZ, USA

ACM ISBN 978-1-4503-5633-6/18/03…$15.00
https://doi.org/10.1145/3180457.3180463

1 INTRODUCTION

Big Data has become an essential asset for enterprises, which are harnessing its potential for generating additional revenue, offering better customer experience, and developing insights into their business models. The data generated from diverse and varied sources including Internet of Things, social platforms, health-care, system logs, bio-informatics, etc. contribute and define the ethos of Big Data which is volume, velocity and variety. Data lake formed by the amalgamation of data from these sources requires powerful, scalable and resilient, storage and processing platforms to reveal the true value hidden inside this data mine.

Over the last several years, Apache Hadoop has emerged as a predominant platform for handling Big Data. Along with core Hadoop 2.x components including Hadoop Common, MapReduce, Hadoop Distributed File System (HDFS) and Apache YARN, several projects have contributed to make Hadoop ecosystem the prime choice as a robust, resilient and fault-tolerant Big Data processing system. Open source projects like Apache HBase, Apache Hive, Apache Knox, Apache Storm, Spark etc. have made this framework available and usable to business and non-technical users also, making it ubiquitous in enterprises, academia and elsewhere. Such wide acceptance of this platform compels researchers and scientists to make it more secure, considering the fact that it handles the most precious asset of any enterprise, i.e. Data. In year 2017 alone several instances of data breaches were brought to the notice of the world [11] which amplifies and emphasises the need for better cyber security and privacy mechanisms.

Hadoop framework security is very challenging considering its distributed nature and broad attack surface. This multi-tenant platform must be secure to prevent unauthorized access to sensitive information and cluster resources used inside this system. Since multiple users would be running different applications and jobs on this platform, it is important that no data breach occurs and relevant data is only revealed to authorized users. The confidentiality and integrity of information and resources can be compromised if attacks like Hadoop service daemons (HDFS NameNode, DataNode, YARN ResourceManager etc.) impersonation, denial of cluster resources, killing or modifying of user applications by malicious user, unauthorized data access in HDFS etc. are orchestrated. For example, in case of Hadoop daemons masquerading, once a malicious service is registered as a part of the Hadoop cluster, unauthorized users can access data blocks residing on data nodes or even consume all cluster resources by running high resource demanding jobs, therefore, preventing other users to use the cluster. Such attacks can be organized from inside and outside of the organization, which makes it more difficult to detect and prevent them.

Researchers have understood the need to embed security into the Hadoop system and have proposed several security measures to prevent cyber attacks. Access control plays a pivotal role to restrict unauthorized access to data, services and resources in the Hadoop cluster. Apache Hadoop core along with other security projects like Apache Ranger [5] and Apache Sentry [6] offer multi layer access controls across multiple Hadoop ecosystem projects including Apache Hive, HBase, YARN etc. using Access Control Lists (ACLs). These current authorization features (we refer to as HeAC) have been presented and illustrated in our previous work [21, 22]. In this paper, we extend the current access control capabilities and propose a fine-grained attribute-based access control model for Hadoop ecosystem, which we call as HeABAC. We further introduce the novel concept of cross Hadoop service trust to secure inter-Hadoop services interaction in the framework. Attribute-based access control (ABAC) [23, 30, 32] has received significant attention in recent years because of the flexibility it offers in multi-user distributed environment, which inspired us to use it in Hadoop ecosystem authorization security controls, which is still primarily based on ACLs. This work is a successor to our previous work [21] which also proposed an extension to HeAC, the Object-Tagged Role based access control model for the Hadoop ecosystem (OT-RBAC) consistent with widely accepted RBAC [19, 38] definitions and incorporating object attribute values called tags. To our knowledge, this paper is the first work to define and formalize a pure attribute-based access control model for Hadoop ecosystem with the concept of cross Hadoop services Trust.

The remaining paper is organised as follows. Section 2 provides an overview of multi-layer access control requirements and features in Hadoop ecosystem. This section also briefly discusses the current Hadoop ecosystem access control model (HeAC) and its OT-RBAC extension. Section 3 proposes and presents the formal attribute-based access control model for Hadoop ecosystem (referred as HeABAC) and introduces the concept of cross Hadoop service trust. This section also presents an implementation approach of HeABAC using Apache Ranger and provides an overview of its administrative requirements. Section 4 presents comprehensive real-world use-cases to demonstrate the usage and application of HeABAC. Section 5 reviews some relevant previous works, followed by summary of the paper in Section 6.

2 ACCESS CONTROL IN HADOOP ECOSYSTEM

The authorization capabilities provided by Apache Hadoop and relevant security projects embody the important security principle of 'Defense in Depth' to secure the Hadoop cluster. The multi-layer access controls offered restrict users and applications at different levels including Hadoop daemon services, ecosystem services, data or service objects and the cluster resources. In this section, we discuss the authorization requirements and various access control mechanisms provided in core Hadoop 2.x, Apache Ranger (version 0.6), Apache Sentry (version 1.7.0) and Apache Knox. We will further briefly outline the Hadoop ecosystem access control model (HeAC) and Object-Tagged Role based access control model (OT-RBAC) which were discussed in our previous work [20–22] and are predecessor to our work in this paper.

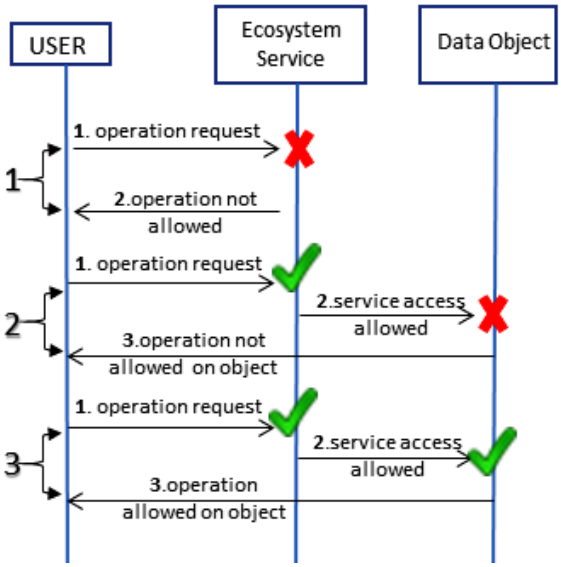

Figure 1: Data Object Access Flow in Hadoop Ecosystem

2.1 Multi-layer Authorization

The first line of check an authenticated user has to pass is the perimeter security. Provided by the service layer authorization, it checks if a user or its application is allowed to access ecosystem services like Apache Hive, HBase etc. or Hadoop daemon services like HDFS NameNode, DataNode, YARN ResourceManager etc. in the cluster. Some capabilities to protect these daemon services are offered by Hadoop core 2.x using Access Control Lists (ACLs), which specify users or groups allowed to perform operations like submitting applications, query application status or simply accessing a service. Besides users, services also have controlled access to other daemons services necessary for task updates or cluster resource status. For example, ACL `security.resourcetracker.protocol.acl` is used to restrict unauthorized communication between YARN ResourceManager and NodeManager. Apache Knox also provides perimeter security using a single access point gateway which sets policies to protect ecosystem services from unauthorized users making API calls. This layer checks and prevents users to access these services much before their underlying data or service objects are accessed, and as such is the earliest access control layer.

Once the user is authorized to access the ecosystem service, data or service objects access control is considered. Hadoop Distributed File System (HDFS) is responsible to store large files in a distributed manner across different data nodes. HDFS mainly uses extended ACLs and POSIX style permissions on files and directories to protect from unauthorized access. With Hadoop 2.x and Apache YARN, various Hadoop ecosystem services can access the same data residing in HDFS in different formats based on the data model supported. For example, Apache Hive represents data in tables and columns, while HBase supports column families etc. These data models support different operations like select or drop in Apache Hive and read or write in HBase. Apache Ranger and Apache Sentry provide plugins which are attached to these services to enforce policy-based access

controls. Therefore, when data is accessed in ecosystem service like Apache Hive, the authorization is checked at both Apache Hive and HDFS objects besides making a prior perimeter check of service access. As shown in Figure 1, when a user tries to access a data object (like table) in an ecosystem service (say Apache Hive), access control policy is first checked for user permission for service. If that is denied, no further access is allowed; in case it is allowed, the next check is at the object itself where the policy is defined for the particular table. Therefore, two level check is done to access objects in Hadoop ecosystem. Apache Ranger also introduces the notion of object tags, where data objects are assigned certain attribute values and access control is defined based on these tags. In such a case, tag based policies are set where the value of tag associated with the object will determine if a user is allowed to access the object. Apache Ranger also allows policies to enable column masking and row filtering in Apache Hive where certain data sets are concealed or filtered from the users which are not allowed to access them.

In addition to protecting services and objects, user applications and cluster resources are also needed to be protected from nefarious users. Unauthorized user must not be able to delete or know the status of any other user. Further, users must not be allowed to use and consume all the resources of the cluster by running bulky jobs to orchestrate denial of resource attacks. Apache YARN offers capacity and fair scheduler queues which enable sharing of resources along multiple users in the cluster. This prevents any malicious user from using all the cluster resources, since each user will only have a limited quota of resources assigned to the queue. These queues have Access Control Lists (ACLs) attached which determine the users who are allowed to submit applications in the queue and who are allowed to modify or delete applications submitted to the queue by the users. In general, each queue have administrators who can kill or modify the jobs submitted in that particular queue. These queues are hierarchial in nature where users allowed to submit applications in parent queues are also allowed to submit applications in children queues but not vice-versa. Hadoop also support cluster node labels where users are only allowed to submit jobs on limited cluster nodes with set labels based on the queues to which they submit the job.

As mentioned above, a layered approach is applied to restrict users from accessing resources, data and services inside the cluster where a user is passed through multiple Policy Decision (PDP) and Policy Enforcement Points (PEP) to ensure authorized access.

2.2 Access Control Models

This subsection reviews the current formal access control model for Hadoop ecosystem, referred as HeAC. It also reviews the Object-Tagged RBAC model [21] which is an extension to HeAC and a precursor to the work in this paper. These models are primarily developed considering the authorization features provided in Hadoop 2.x and two dominant security projects, Apache Ranger (version 0.6) and Apache Sentry (version 1.7.0). Apache Ranger provides permissions to the users either directly or through group memberships, whereas Apache Sentry uses roles which are assigned to groups to which users are made members to get permissions. We will now discuss the components of HeAC and OT-RBAC models as shown in Figure 2 and Figure 3 respectively.

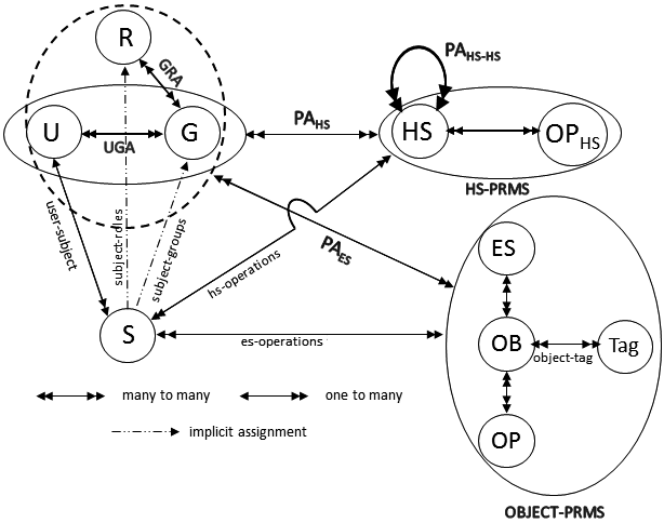

Figure 2: The Conceptual HeAC Model [21]

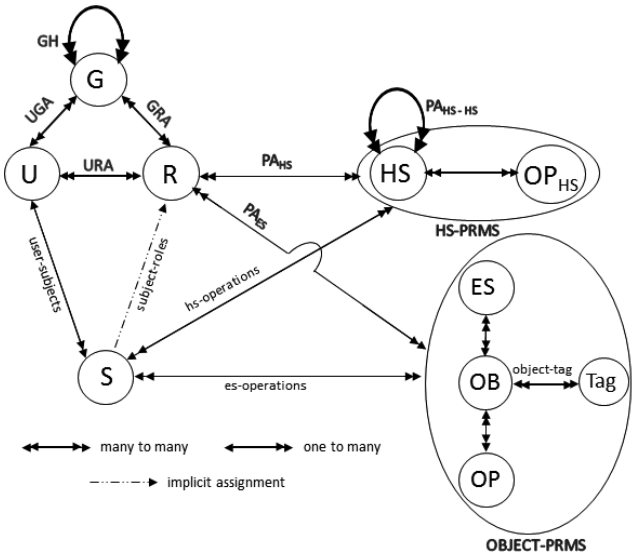

Figure 3: The Conceptual OT-RBAC Model [21]

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

- **Users, Groups, Roles and Subjects:** A user is human interacting directly with the computer system to access data objects and services in a Hadoop cluster. Users can have similar characteristic and requirements which are bundled to form groups. A role as defined in [38] is a collection of permissions which are assigned to various users in the system. A subject represents an application running on behalf

of the user to perform actual operations on the objects and services in Hadoop ecosystem.

- **Hadoop Services:** These are daemon services running in the background to provide the core functionalities in Hadoop 2.x platform. Some examples of these services are Hadoop NameNode, DataNode, YARN ResourceManager, ApplicationMaster, Secondary NameNode etc. Users or applications interact with these daemons to submit application or to view the status of running jobs. These services also need interaction among themselves for operations including task updates or resource monitoring. These Hadoop daemon services do not have any objects associated with them.
- **Operations on Hadoop Services:** These are the set of actions allowed on the Hadoop daemon services, predominantly including access operation performed by users or other daemon services. For example, NameNode accessing DataNode, or ResourceManager accessing ApplicationMaster are common Hadoop requirements in background.
- **Ecosystem Services:** Hadoop ecosystem support several projects or services (Apache HBase, Apache Hive, Apache Kafka etc.) which have underlying data or service objects operated upon by users. Any user or application must be allowed to access the service first before its object can be accessed. For example, Apache Hive ecosystem service must be first allowed to be accessed by user before any operation on its table object is performed.
- **Data and Service Objects:** Different ecosystem services support various data and service objects which are operated upon by the users or users' subjects. These are the actual resources which are secured from unauthorized access in the Hadoop cluster besides computational resources. For example, YARN has queue objects, Apache Kafka has topics, Apache HBase support column family and Apache Hive has tables etc.
- **Operations on Objects:** Different objects in ecosystem services support various operations inside the Hadoop cluster based on data model supported. Operations vary from select or drop in Apache Hive, read or write in HDFS, and submit or administer on YARN queue objects etc.
- **Object Tags:** These are attribute values attached to various objects in the system. Such values can help defining tag-based policies where a user is permitted or denied to perform operation on a resource based on the tag associated with the resource. Each object can have multiple tags associated with it which is done primarily under Apache Atlas service.

In both HeAC and OT-RBAC models, as shown in Figure 2 and Figure 3 respectively, two types of permissions exist in the system — Hadoop service permissions (HS-PRMS) and Object permissions (OBJECT-PRMS). HS-PRMS defines the operations on Hadoop services and is formally stated as a power set of the cross product of HS and OP_{HS}. OBJECT-PRMS define the set of permissions to perform operations (OP) on the objects (OB) in ecosystem services (ES). These permissions either include the object or the tag (Tag) associated with the objects (shown by object-tag) to create tag-based policies. It should be noted that access to ecosystem service is first required before performing operation on its objects, which is incorporated in OBJECT-PRMS by using ecosystem service (ES) as its component. It may be required for a user to have one or both the types of permissions to perform operations in cluster. Cross Hadoop service communication requirements is expressed by $PA_{HS\text{-}HS}$ permission assignment (shown as self-loop).

HeAC model and OT-RBAC models are primarily different in terms of permission assignments. In HeAC, HS-PRMS are assigned to both users and groups (shown as many to many PA_{HS}), whereby a user can get permission directly or through group membership. OBJECT-PRMS are assigned to users, groups and roles also (reflected by dotted circle in Figure 2 and shown as PA_{ES}). These assignments are very different in OT-RBAC model which proposes to assign HS-PRMS and OBJECT-PRMS only to roles, thereby inheriting the benefits of role based access control model [38]. Also with the notion of group hierarchy in OT-RBAC, shown as GH in Figure 3, roles are inherited from junior to senior groups. Therefore, a user assigned to a senior group g_1 gets all the roles of its junior groups besides the roles of group g_1. This group hierarchy offers the advantage of easy administration where several roles can be assigned or removed from a user just by assigning or removing it from groups. The importance of group hierarchy will be further reflected in the next section which will describe its use in attribute based models. It should be noted that a user will need multiple PA_{ES} and PA_{HS} assignments based on the type of operation requested. Finally, a subject created by user will inherit all or some of the permissions assigned to its creator user to perform operations in Hadoop cluster.

In the following section, we will discuss the proposed attribute-based access control model for Hadoop Ecosystem, referred as HeABAC. Both the aforementioned HeAC and OT-RBAC models are precursors to the eventual and desired fine grained HeABAC model for multi-tenant Hadoop environment.

3 ABAC MODEL FOR HADOOP ECOSYSTEM

Attribute based access control is known to offer flexible fine grained authorization capabilities by introducing the characteristics of subjects, objects, environment or context in access control decision. Such mechanisms are required in complex distributed systems like Hadoop where multi-tenant data lake is being accessed at varied data granularity levels by multiple users at different time, locations and conditions. Real-world use-cases like a user allowed to access data only from a particular location or IP address or at a specific time are very common and most conveniently addressed by attribute based systems.

3.1 The HeABAC Model

We will now define the attribute based access control model for Hadoop ecosystem, referred as HeABAC and shown in Figure 4. The complete formalization of HeABAC model is given in Table 1.

The basic sets of HeABAC model involve the previously defined access control components — Users (U), Groups (G), Subjects (S), Hadoop Services (HS), Ecosystem Services (ES), Data and service objects (OB), Operations on objects in Ecosystem Services (OP), Operations on Hadoop Services (OP_{HS}), as elaborated in Section 2 and stated in Table 1. Some of these entities have characteristics which are used in access control decision and are expressed as

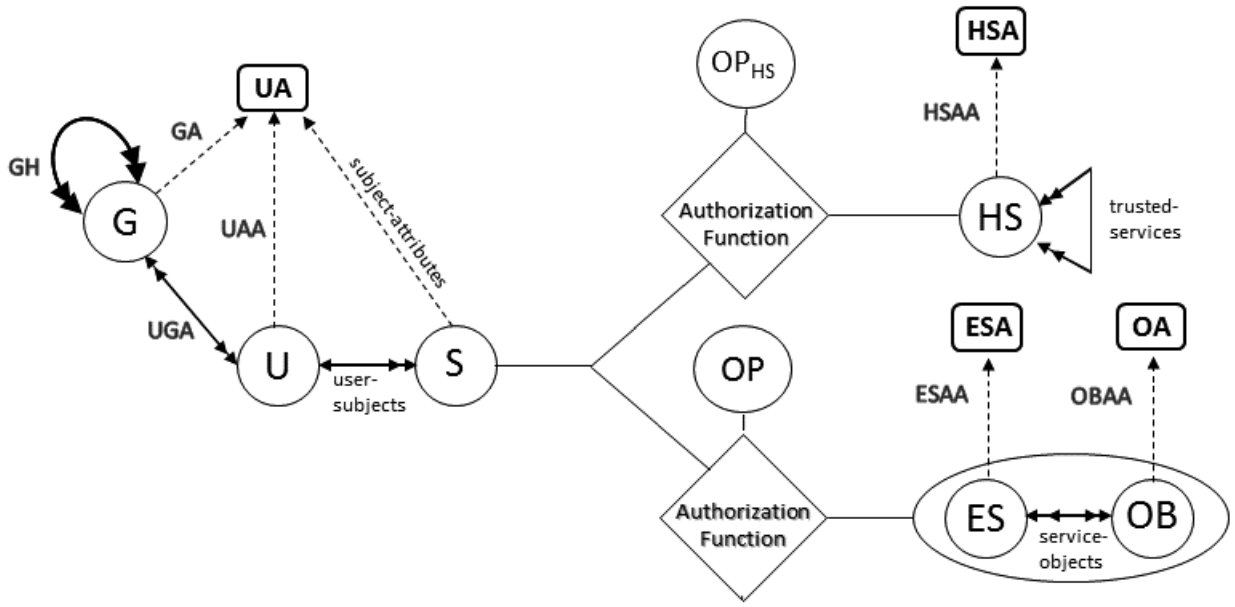

Figure 4: The Conceptual Attribute Based Access Control Model for Hadoop Ecosystem (HeABAC)

their attributes. User attributes (UA) is the set of user attributes for users, groups and subjects. Object attributes (OA) is the set of object attributes assigned to data and service objects (OB). Ecosystem service attribute (ESA) and Hadoop service attribute (HSA) are the set of attribute functions which can be assigned to Ecosystem services (ES) and Hadoop Services (HS) respectively. Users, groups, Hadoop or ecosystem services and objects can be assigned attribute values directly for an attribute function att (in their respective sets) from the set of atomic values in the range, denoted by Range(att). Attribute functions in UA are required to be only set valued whereas for other sets OA, ESA and HSA both set and atomic valued functions are allowed. Each attribute function in UA, denoted by att_u, will map a user or group to a set of values in power set of Range(att_u). Similarly, attribute functions in OA, ESA and HSA map OB, ES and HS respectively to one or subset of attribute values from the range depending on atomic or set valued attribute type as shown in Table 1.

Users are assigned to multiple groups (defined by many to many function directUG) to achieve simplified administration of attributes. When a user is made member to a group, the user inherits all attributes of the group, whereby multiple attributes can be assigned or removed from a user by just a single administrative action. Further, group hierarchy (GH) also exists in the system (shown as self loop on G), defined using a partial order relation on G and denoted by $\succeq_g$, where $g_1 \succeq_g g_2$ signifies g_1 is senior to g_2 and g_1 will inherit all the attributes of g_2. Therefore, for attribute att_u, the effective values for group g_1 is the union of values directly assigned to g_1 for att_u and the effective values of att_u for all the junior groups to g_1, as defined by effective$G_{att_u}(g_1)$. The effective attribute values of a user for attribute att_u will then be the directly assigned values to user for att_u and the effective attribute values of attribute att_u for all the groups to which user is directly assigned. For example, if group g_1 has attribute role with value Chair, and a junior group g_2 has role with value Faculty, then the effective values of attribute role for group g_1 will be Chair and Faculty. Further, when a user is assigned to group g_1, it will inherit all values of attribute role (i.e. Chair and Faculty) besides the values directly assigned to user, as further elaborated in [23, 41]. A subject which is created by the user (denoted by function userSub) inherits subset or all the values of effective attributes of the creator user as stated by effective$U_{att_u}(s) \subseteq$ effectiveU_{att_u}(userSub(s)). These values can change with time but must not exceed values of the creator user.

In Hadoop ecosystem, several Hadoop services interact or access other Hadoop services for task updates or cluster resource status (like HDFS NameNode and DataNode or YARN ResourceManager and ApplicationMaster). We refer to this type of interaction as Cross Hadoop Service Trust in Table 1 (stated as trusted-services), determining which Hadoop services are allowed to access other services. In this cross service relation, we introduce the notion of Cross Hadoop service trust as a many to many relation where $HS_A \trianglelefteq HS_B$ denotes that HS_B is a trusted service by HS_A and therefore, HS_B is allowed to access or interact with HS_A. In this case HS_B is a trustee service and HS_A is a trustor service, and trust relation existence is controlled by HS_A. This trust relation obviates the need to specify ACLs as done in HeAC model. For example, a service level authorization ACL `security.datanode.protocol.acl` controls which DataNodes are allowed to communicate with NameNodes. In such cases, a DataNode running as datanode1 can access service NameNode namenode1, if cross service trust relation is established between them i.e. namenode1 $\trianglelefteq$ datanode1. Different types of cross Hadoop service trust relations can exist in the system which are discussed further in the next subsection.

Table 1: Formal ABAC Model Definitions for Hadoop Ecosystem

Basic Sets and Functions

– U, G, S are finite sets of users, groups and subjects respectively.
– HS, ES are finite sets of Hadoop services and ecosystem services respectively.
– OB, OP are finite sets of objects and object operations respectively.
– OP_{HS} is a finite set of operations on Hadoop services.
– UA, OA are finite sets of user and object attribute functions respectively.
– ESA, HSA are finite sets of ecosystem and Hadoop service attribute functions respectively.
– For each attribute att in UA ∪ OA ∪ ESA ∪ HSA, Range(att) is a finite set of atomic values.
– attType: UA = {set}, defines user attributes to be set valued only.
– attType: OA ∪ ESA ∪ HSA = {set, atomic}, defines remaining attributes to be set or atomic valued.
– For each attribute att_u in UA, $att_u : U \cup G \rightarrow 2^{Range(att_u)}$ mapping each user or group to a set of attribute values in $Range(att_u)$.
– Each attribute att_{ob} in OA maps objects in OB to attribute values. Formally, $att_{ob} : OB \rightarrow \begin{cases} Range(att_{ob}) & \text{if attType}(att_{ob}) = \text{atomic} \\ 2^{Range(att_{ob})} & \text{if attType}(att_{ob}) = \text{set} \end{cases}$
– Each attribute att_{es} in ESA maps services in ES to attribute values. Formally, $att_{es} : ES \rightarrow \begin{cases} Range(att_{es}) & \text{if attType}(att_{es}) = \text{atomic} \\ 2^{Range(att_{es})} & \text{if attType}(att_{es}) = \text{set} \end{cases}$
– Each attribute att_{hs} in HSA maps services in HS to attribute values. Formally, $att_{hs} : HS \rightarrow \begin{cases} Range(att_{hs}) & \text{if attType}(att_{hs}) = \text{atomic} \\ 2^{Range(att_{hs})} & \text{if attType}(att_{hs}) = \text{set} \end{cases}$
– $directUG : U \rightarrow 2^G$, mapping each user to a set of groups, equivalently UGA ⊆ U × G
– GH ⊆ G×G, a partial order relation $\succeq_g$ on G

Effective Attributes of Users (Derived Functions)

– For each attribute att_u in UA,
  • $effectiveG_{att_u} : G \rightarrow 2^{Range(att_u)}$, defined as $effectiveG_{att_u}(g_i) = att_u(g_i) \cup \left(\bigcup_{g \in \{g_j | g_i \succeq_g g_j\}} effectiveG_{att_u}(g) \right)$.
– For each attribute att_u in UA,
  • $effectiveU_{att_u} : U \rightarrow 2^{Range(att_u)}$, defined as $effectiveU_{att_u}(u) = att_u(u) \cup \left(\bigcup_{g \in directUG(u)} effectiveG_{att_u}(g) \right)$.

Effective Attributes of Subjects

– userSub : S⟶ U, mapping each subject to its creator user.
– For each attribute att_u in UA, $effectiveU_{att_u} : S \rightarrow 2^{Range(att_u)}$, mapping of subject s to a set of values for its effective attribute att_u. It is required that : $effectiveU_{att_u}(s) \subseteq effectiveU_{att_u}(userSub(s))$.

Cross Hadoop Services Trust

– trusted-services : $HS \rightarrow 2^{HS}$ is a required function to map Hadoop services to a set of trusted Hadoop services. Equivalently, relation service trust written as $\trianglelefteq \subseteq HS \times HS$, where $hs_a \trianglelefteq hs_b$ iff $hs_b \in$ trusted-services(hs_a), meaning trustee service hs_b is trusted by trustor service hs_a and Hadoop service hs_b can access service hs_a.

Authorization Functions

– Service Authorization Function: For each op ∈ OP_{HS}, $Authorization_{op}(s : S, sr : HS \cup ES)$ is a propositional logic formula returning true or false, which is defined using the following policy language:
  • $\alpha ::= \alpha \wedge \alpha \mid \alpha \vee \alpha \mid (\alpha) \mid \neg\alpha \mid \exists\, x \in set.\alpha \mid \forall\, x \in set.\alpha \mid set \triangle set \mid \text{ atomic} \in set \mid atomic \notin set$
  • $\triangle ::= \subset \mid \subseteq \mid \nsubseteq \mid \cap \mid \cup$
  • $set ::= effective_{att_u}(s) \mid att_{sr}(sr)$ for $att_u \in UA$, $sr \in ES \cup HS$, attType(att) = set
  • $atomic ::= att_{sr}(sr) \mid value$ $sr \in ES \cup HS$, attType(att) = atomic
– Data or Service Objects Authorization Function: For each op ∈ OP on objects ob ∈ OB belonging to ecosystem service es ∈ ES, $Authorization_{op}(s : S, es : ES, ob : OB)$ is a propositional logic formula returning true or false, which is defined using the policy language stated above with following changes
  • $set ::= effective_{att_u}(s) \mid att_{es}(es) \mid att_{ob}(ob)$ for $att_u \in UA$, attType(att) = set
  • $atomic ::= att_{es}(es) \mid att_{ob}(ob) \mid value$ attType(att) = atomic

Access Operation Decision

– A subject s ∈ S is allowed to perform an operation op ∈ OP_{HS} on a service sr ∈ ES ∪ HS if the effective attributes of subject s and attributes of services satisfy policies stated in $Authorization_{op}(s : S, sr : HS \cup ES)$.
Formally, $Authorization_{op}(s : S, sr : HS \cup ES)$ = True.
– A subject s ∈ S is allowed to perform an operation op ∈ OP on an object ob ∈ OB in ecosystem service es ∈ ES if subject is allowed to access services es and the effective attributes of subject s, the attributes of object ob and service es satisfy policies in $Authorization_{op}(s : S, es : ES, ob : OB)$.
Formally, $Authorization_{access}(s : S, es : ES)$ = True ∧ $Authorization_{op}(s : S, es : ES, ob : OB)$ = True.

We specify two separate authorization functions, one for Hadoop or ecosystem service authorization and other for ecosystem data or service objects authorization, for operations in OP_{HS} and OP respectively. A common policy language is defined for both authorization functions using propositional logic formula, which define the conditions to approve or deny operation on an object by a subject, as stated in Table 1. In case of service authorization, each operation op $\in OP_{HS}$ has an authorization function $Authorization_{op}(s : S, sr : HS \cup ES)$ specifying the conditions under which subject $s \in S$ is allowed to perform operation op $\in OP_{HS}$ on any service sr in Hadoop service HS or Ecosystem service ES in the Hadoop cluster. Here the effective attributes of the user and the direct attributes of the service in question are used to make the access control decision. Another authorization function is defined for data or service objects for each operation op $\in$ OP as stated by $Authorization_{op}(s : S, es : ES, ob : OB)$. This authorization function specifies the conditions under which subject $s \in S$ is allowed to perform an operation op $\in$ OP on an object ob $\in$ OB in service es $\in$ ES. In this function, the policy conditions are allowed to use the effective attributes of user but only the direct attributes assigned to the service and objects requested for access. Since a subject must first be checked if allowed to access the service before its object is allowed to operate, therefore, a subject $s \in S$ is allowed to perform operation op $\in$ OP on an object ob $\in$ OB in service es $\in$ ES if and only if both $Authorization_{access}(s : S, sr : ES)$ and $Authorization_{op}(s : S, es : ES, ob : OB)$ evaluate to True.

The proposed HeABAC model presents a pure attribute based access control model for Hadoop ecosystem to offer flexible and fine grained access to different objects in the cluster. Since the cluster and Hadoop data lake is multi-tenant system used by several entities with different access requirements, ABAC will play an important role to maintain confidentiality and integrity of data and cluster resources. Other contextual and environment attributes can be also introduced into the system to offer more complex access control policies, but are considered out of scope of this paper to solely present the prime essence, i.e. the use of attributes for access control in Hadoop ecosystem. It should be noted that data ingestion security is out of scope and we assume data is already present in the system before access control mechanisms come into play.

3.2 Concept of Cross Hadoop Services Trust

Cross Hadoop service trust determines which two Hadoop services can interact with each other. Our definition of trust relations are primarily unidirectional and involves only two Hadoop services, a trustor and trustee. We assert that trust is established unilaterally by the trustor, and can only be revoked or modified by the trustor.

The cross Hadoop trust relation $\trianglelefteq$ is a binary relation established between trustor and trustee services. A service can be a trustor in one relation and trustee in another. This relation has the following defining properties, for Hadoop services hs_a, hs_b, $hs_c \in HS$:

- **Reflexive:** A Hadoop service must always trust itself, meaning $hs_a \trianglelefteq hs_a$.
- **Non Transitive:** The Cross Hadoop service trust relation is always defined by the trustor and cannot be inferred from any indirect combinations of other trust relationships i.e. $hs_a \trianglelefteq hs_b \wedge hs_b \trianglelefteq hs_c \not\Rightarrow hs_a \trianglelefteq hs_c$.
- **Non Symmetric and Non Asymmetric:** This characteristic states that the trust relation is always unidirectional and is also independent in each direction i.e. $hs_a \trianglelefteq hs_b \not\Rightarrow hs_b \trianglelefteq hs_a$ and $hs_a \trianglelefteq hs_b \wedge hs_b \trianglelefteq hs_a \not\Rightarrow hs_a \equiv hs_b$.

We will now identify and discuss four potential types of trust relations to enable cross Hadoop services access control. The type of relation is determined by who controls the existence of trust relation and who controls access to the service. These types of relations are analogous and adapted from the trust relation types discussed in [36, 45–47].

Type-α. In this relation, trustor grants access to trustee and the relation is controlled (exists or created) by the trustor. For example, if $hs_a \trianglelefteq_\alpha hs_b$, then Hadoop service hs_a authorizes service hs_b to access hs_a, and the relation is controlled by hs_a and service access is also controlled by hs_a. This type of relation are most intuitive and for simplicity, is used in our HeABAC model.

Type-β. In this relation, trustee grants access to trustor and the relation is still controlled by trustor. For example, if $hs_a \trianglelefteq_\beta hs_b$, then Hadoop service hs_b authorizes service hs_a to access hs_b, and the relation is controlled by hs_a and service access is also controlled by hs_a without the consent from service hs_b.

Type-γ. In this relation, trustee controls access of trustor and the relation is still controlled by trustor. For example, if $hs_a \trianglelefteq_\gamma hs_b$, then Hadoop service hs_b authorizes service hs_a to access hs_b. Here the relation is controlled by hs_a but service access is controlled by Hadoop service hs_b.

Type-δ. In this relation, trustee takes access from trustor by approving or denying access. For example, if $hs_a \trianglelefteq_\delta hs_b$, then Hadoop service hs_a authorizes service hs_b to access hs_a, and the relation existence is controlled by hs_a but the service access is controlled by the consent of hs_b.

3.3 HeABAC Implementation Approach

Apache Ranger and Apache Sentry are two dominant open-source security projects in Hadoop ecosystem which are focussed in offering authorization and access control capabilities in several ecosystem projects including Apache Hive, HBase, Kafka etc. Both Apache Ranger and Sentry provides security plugins which are attached to different ecosystem services which needs to be protected. Every access request by a user is intercepted by these plugins which check the security policies defined by the administrator using REST API or user interface, to decide and enforce access control decisions. Apache Ranger currently offers some fine grained extensions where attributes of a user are embedded into the access request using context enricher. These context enrichers are Java class which enriches the request of the user with extra information which is used in the security policy conditions to approve or deny request. These conditions are evaluated using condition evaluators which are also defined in Apache Ranger. For example, is a user Alice wants to access an object obj1 but the policy specifies that user Alice can only access resource obj1 if time is after 10 pm, then, the context enricher will add on the current time into the access request of the user Alice and the condition evaluator will check if the access request complies with the time condition specified in the policy.

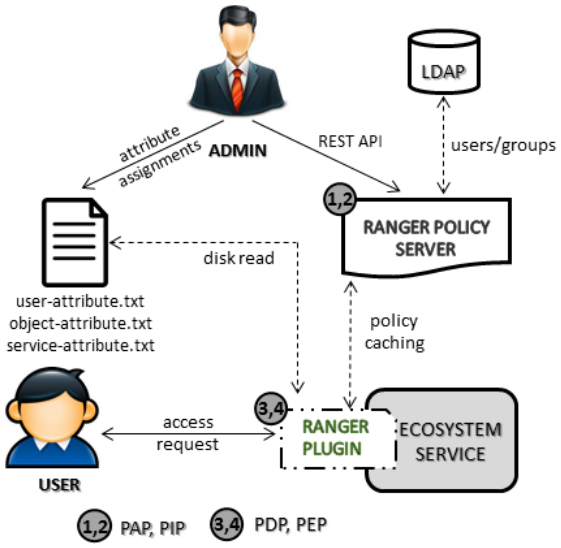

Figure 5: Proposed HeABAC Implementation

Our proposed implementation of HeABAC extends the current context enricher and condition evaluators in Apache Ranger. We propose that context enricher will not only be used for enriching user information but also for services and objects in the access request. As shown in Figure 5, the security administrator will add text files for different users, objects and services into the system with their relevant attributes. These files will then be used by context enricher implemented, which will add attributes of users, services and objects in the access request. Similarly, condition evaluators also need to be extended to incorporate the attributes of objects and services in policies, which will be also evaluated when the enriched access request with attributes is checked against a defined security policy. Here, the administration of policies is done through the central policy server while the decisions and enforcement are made by Apache Ranger security plugins attached with the individual services as shown in Figure 5.

3.4 Administrative Realms of HeABAC

The intrinsic character of HeABAC model involves attributes, which determine the permissions granted or denied to the user to operate or access objects and services in the Hadoop system. The assignment of attributes to these entities before the implementation and enforcement of HeABAC model is very important. Also, the establishment and agreement of cross Hadoop service trust relations are required to allow inter Hadoop service communication. In this subsection, we will briefly highlight the administrative models necessary to manage the HeABAC operational model.

As shown in Figure 4, the user side of HeABAC requires some administrative models to manage user attribute assignment (UAA), user group assignment (UGA), group attribute assignment (GA) and group hierarchy (GH). The GURA_G [23] and GURA [29] administrative models proposed (inspired by [37]) discuss the administration of these attributes via different administrative roles and relations. These roles will be able to assign values to various attributes of user and groups based on different policy conditions specified in the administrative models. Further, the assignment of user to groups will also be managed by these administrative roles depending on the current group membership of user or its attributes. Group Hierarchy (GH) is assumed to be static in our model but can be dynamic based on the changes in the attribute values of the groups. Similarly, the addition or deletion of attributes for Hadoop services (HSAA), Ecosystem service assignment (ESAA) and Objects attribute assignment (OBAA) can be also managed. The creation and changes in security policies for various authorization functions is the responsibility of security architect of the Hadoop data lake provider and must be done in a diligent manner.

The administrative operations necessary for cross Hadoop service trust involves initialization, acceptance and revocation of trust among relevant Hadoop services. Since, these Hadoop services are running under some service user account, the establishment and revocation of the trust must be initialized and accepted by relevant service users of the Hadoop services depending on the type of trust relation created.

4 USE-CASES AND HeABAC APPLICATION

In this section we will illustrate some important use-cases to emphasise the use and benefits of fine grained and flexible attribute based access control in Hadoop ecosystem. These use-cases will reflect real world scenarios and will cover different access control requirements as discussed in earlier sections of this paper. In these use-cases we consider that users have already been authenticated by some external mechanisms and data is already ingested into the Hadoop system before access control comes into enforcement.

Internet of Things is a growing buzz among different businesses, and several enterprises are harnessing the potential it offers. It has spread itself to different spheres of our life including health, smart homes and more recently to smart city and transportation. Vehicular Internet of Things is the future where vehicles and road infrastructure will be communicating with each other. These vehicles will be generating lot of data ranging from car sensor readings, road conditions or even driver health vitals to be analysed by different stake-holders for better and life saving services to the customers. Let us suppose, a connected vehicle from car manufacturer Toyota is running on the road and is continuously generating data, which is stored in multi-tenant Hadoop data lake. This stored data can be used by various entities, including the car dealer or manufacturer for diagnostic services, by transportation or police department for over-speeding check, by insurance company to understand driver driving behaviour or by a doctor who is continuously monitoring the heart-rate of the patient driver. Each of these users must have different levels of access to data in the Hadoop data lake and are only required to know what they should need to know to perform their functions following the principle of least privileges, and without compromising the integrity and privacy of data. Further, for analysis purposes these users will be also running some jobs or applications in the Hadoop cluster, which must cater the needs of all the users without unwarranted resource constraints.

Figure 6 illustrates the use cases to reflect the importance of attribute based system in such distributed and multi-tenant environment like Hadoop. In this case, a user Alice is assigned to a

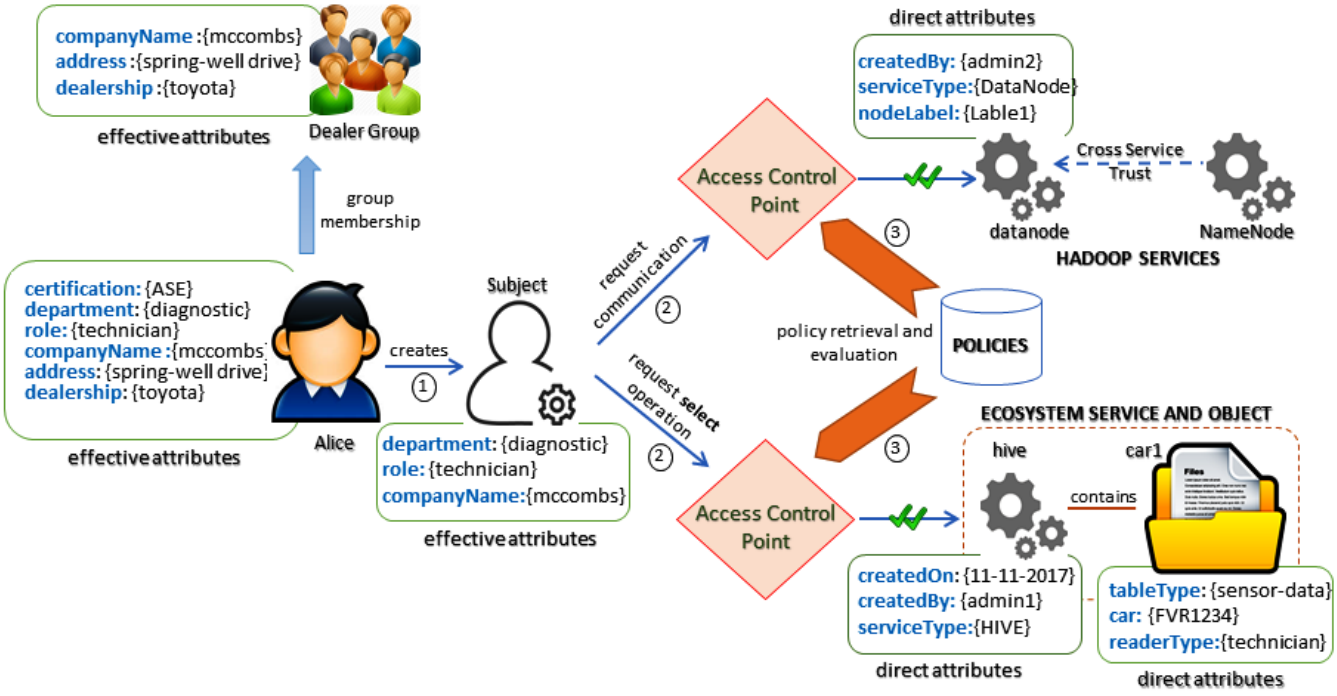

Figure 6: IoT Use-Case Illustrating ABAC Access Control in Hadoop Ecosystem

Dealer group, which makes it inherit the attributes of Dealer group, yielding the effective attributes for Alice. Here, the attributes of Dealer group (`companyName: mccombs, address: spring-well drive, dealership: Toyota`) are added to the direct attributes of Alice user (`certification: ASE, department: diagnostic, role: technician`). The benefits of user to group assignment are evident, since with single administrative operation all the attributes of Dealer group are assigned to Alice. Further, when Alice creates a subject, the subject inherits subset of the effective attributes of Alice. Also, other entities shown such as Hadoop services (datanode, NameNode), Ecosystem service (hive) and object (Table car1) in the system, are also assigned direct attributes by a security administrator. Security policies are defined by the architect and stored in the central policy server. Cross Hadoop services trust relation is also established which will be discussed more in the following part of this section. The numbers in the figure define the sequence of access control process where the subject is first created, which initiates requests to perform different operations on objects and services. These requests are intercepted by the access control decision and enforcement point (shown as rhombus), which will retrieve polices from the central server to make an access control decision.

Let us assume that the following security policy (referred as policy 1) is created by an administrator in the system to control access to some Ecosystem service:

$$\texttt{Authorization}_{\texttt{access}}\texttt{(s:S, es:ES)} \equiv \texttt{diagnostic} \in \texttt{effective}_{\texttt{department}}\texttt{(s)} \wedge \texttt{technician} \in \texttt{effective}_{\texttt{role}}\texttt{(s)} \wedge \texttt{serviceType(es) = HIVE} \wedge \texttt{createdBy(es) = admin1}$$

This policy states that a user (or subject) belonging to diagnostic department with technician role can access ecosystem service of type HIVE which was created by admin1. Clearly, if subject s : S created by a user and an ecosystem service es : ES satisfy the stated policy condition (i.e. evaluates to True), then access operation will be granted on service es to subject s. This policy can be enforced by Apache Knox [4], which offers a single point gateway to multiple services inside Hadoop ecosystem. Another security policy (referred as policy 2) is created to determine if select operation is allowed by a subject s : S on an object ob : OB in ecosystem service es : ES:

$$\texttt{Authorization}_{\texttt{select}}\texttt{(s:S, es:ES, ob:OB)} \equiv \texttt{Authorization}_{\texttt{access}}\texttt{(s:S, es:ES) = True} \wedge \texttt{diagnostic} \in \texttt{effective}_{\texttt{department}}\texttt{(s)} \wedge \texttt{effective}_{\texttt{role}}\texttt{(s)} \in \texttt{readerType(ob)} \wedge \texttt{tableType(ob) = sensor-data} \wedge \texttt{car(ob) = FVR1234}$$

This policy requires a subject to perform select operation on an object ob belonging to Ecosystem service es if the user belongs to diagnostic department and the effective roles of the user belong to readerType attribute of the object, tableType attribute of object having value sensor-data and car attribute with value FVR1234. It should be noted that, this authorization function has a condition $\texttt{Authorization}_{\texttt{access}}\texttt{(s:S, es:ES) = True}$, stating that subject s must be first allowed to access ecosystem service es before its underlying object ob is allowed to be operated by subject s.

Let us say, in our use-case a user Alice from a car dealer wants to read the data of a car which is stored in Hadoop data lake. As a security requirement, Alice can only access data through Apache Hive ecosystem service with no direct access to data at HDFS level. Alice has some attributes which are its own, but some are also inherited from the car dealer as being a part of its employee. For this access to authorize, Alice must first be allowed to access Apache

Hive ecosystem service and then allowed to read the table inside it. Looking at the effective attributes of user Alice, the subject created by Alice and the attributes of service hive and object table car1, it can be well understood that the policy 1 and policy 2 are satisfied by subject of user Alice. Therefore, select operation by Alice on table car1 is allowed by the defined security policies. Let us suppose another user Bob from the same car dealer but in a different department (say sales) tries to perform select operation on the same object Table car1. The operation will not be allowed as the value for department attribute for Bob will be sales, which will not satisfy the above stated policies. Similar set of policies can be defined in the system to cater various other use-cases and security requirements in Hadoop data lake. For example, some user may only be allowed to access HDFS files directly without access through Apache Hive, or some may be allowed to submit YARN applications only. Policies can also be defined to prevent denial of resource attacks where specific users are only allowed to submit jobs to YARN capacity or fair scheduler queues which have limited set of resources attached to them. A sample security policy to restrict submitting YARN applications to only specific users can be defined for YARN capacity scheduler queues as:

$$\texttt{Authorization}_{\texttt{submit}}\texttt{(s:S, es:ES, ob:OB)} \equiv \texttt{Authorization}_{\texttt{access}}\texttt{(s:S, es:ES)} = \text{True} \wedge \texttt{diagnostic} \in \texttt{effective}_{\texttt{department}}\texttt{(s)} \wedge \texttt{technician} = \texttt{effective}_{\texttt{role}}\texttt{(s)} \wedge \texttt{queueType(ob)} = \texttt{dept-diagnostic} \wedge \texttt{queueAdmin(ob)} = \texttt{admin2}$$

In this case, $\texttt{Authorization}_{\texttt{access}}\texttt{(s:S, es:ES)} = \text{True}$ signifies that a user with certain attributes must be allowed to access YARN ResourceManager (approved by a separate policy) before allowed to submit applications to its queues which have attributes queuetype having value dept-diagnostic and queueAdmin with value admin2.

Similar attribute based access control policies can be also defined for controlling user access to Hadoop daemon services like HDFS DataNode, NameNode etc. In Figure 6, we created a Hadoop service DataNode 'datanode' which has a set of attributes directly assigned to it. An access control list (ACL) `security.client.datanode.protocol.acl` is currently defined in Hadoop to control the communication between user clients and DataNodes to retrieve data blocks. The following security policy can be used in place of ACL to control this access:

$$\texttt{Authorization}_{\texttt{access}}\texttt{(s:S, hs:HS)} \equiv \texttt{diagnostic} \in \texttt{effective}_{\texttt{department}}\texttt{(s)} \wedge \texttt{technician} \in \texttt{effective}_{\texttt{role}}\texttt{(s)} \wedge \texttt{serviceType(hs)} = \texttt{DataNode} \wedge \texttt{createdBy(hs)} = \texttt{admin2}$$

Clearly, if user Alice subject tries to access DataNode service 'datanode', the aforementioned policy will allow Alice to access service 'datanode' which will serve the purpose of defining `security.client.datanode.protocol.acl` ACL. Similar attribute based policies can be stated for other service level authorization ACLs also.

Another Hadoop service NameNode has also been shown in Figure 6 which trusts DataNode service 'datanode'. Cross Hadoop service trust relation is needed to control cross Hadoop services communication which is currently controlled by ACLs. For example, `security.datanode.protocol.acl` controls communication between DataNode and NameNode. These can be replaced by defining trust relations between Hadoop services. As shown in Figure 6, DataNode 'datanode' has a cross service trust with NameNode, meaning datanode can access NameNode where NameNode is trustor and 'datanode' is trustee service. To state this requirement, we assume to have a cross Hadoop service Type-α trust type, written as NameNode $\trianglelefteq_\alpha$ datanode, where the trust is initiated by NameNode and service access to NameNode is also controlled by NameNode service. Other trust types can also be considered depending on use-cases requirements but for simplicity, we restrict to Type-α trust type in HeABAC model.

As noted, these real world use-cases illustrate how attribute-based access controls can be enforced into this dynamic and distributed environment, where users have different access needs. Fine grained requirements of a multi-tenant Hadoop data lake, where a user can access one service but not other, or two users having different levels of access to the same object can be truly catered by this HeABAC authorization model. Further, the use of trust in cross Hadoop service communication obviates the need to define service level authorization ACLs in the system.

5 RELATED WORK

Security and Privacy of Big Data has always been a concern in the academic and scientist community. Several works have been published to protect the confidentiality, integrity and availability of these valuable resources. The first paper we read about Big Data access control security is from Hu et al. These researchers proposed an access control scheme for Big data processing in [27] which offers the concepts security agreement, trust list, access control policy and trust chain to authorize access to authenticated user. The paper offers a more general approach for access control in Big data processing systems with some examples of incorporation in Hadoop. However, the details pertinent to Hadoop ecosystem are not well discussed in this paper. Our work is very inspired by this important paper from National Institute for Standards and Technology (NIST) researcher. NIST Big Data Public Working Group has been formed to create Big Data interoperability framework in consensus with academia, industry and government. The draft version 2 of special publication [40] offers aspects of security and privacy with respect to Big Data, reviews security and privacy use cases, and proposes security and privacy taxonomies. Several other white papers and enterprise technical reports have been published to emphasis the security requirements and techniques for protecting Big data in Hadoop including [1, 2, 17, 35, 44, 52]. Articles and reports have also been published online to illustrate several security features and their incorporation in Hadoop ecosystem. Some important best practices in relation to securing HDFS and Apache Hive [3] are discussed in [24, 25] respectively. Hortonworks [8] and Cloudera [7] have been key players in offering secure Hadoop deployments which have provided several important security features in their proprietary Hadoop suite offerings.

A demonstration paper in a representative Hadoop ecosystem presenting multiple access control mechanisms using Apache Ranger is an important cohesive document [20] to understand multi-layer access control features in Hadoop. Sharma et al also presented security issues and requirements in Hadoop system in [42]. Zeng et al proposed content based access control model for content based

data sharing in [54]. A cryptographically enforced access control system based on proxy re-encryption for Big data processing in cloud in discussed in [34]. Security and privacy of data including the MapReduce job security in cloud, authentication of mapper and reducer, and other security challenges are well addressed in [18]. A security model for G-Hadoop based on public key cryptography, SSL protocol and a single sign-on approach is discussed in [55]. Ulusoy et al have presented important fine grained access control system for MapReduce systems namely, GuardMR and Vigiles in [49, 50] respectively. Colombo et al presented a research road-map for Big Data security in [15]. Same authors have also discussed research challenges and fine grained context aware access controls in MongoDB NoSQL database in [14]. Privacy preserving Big data computing and challenges are elaborated extensively in [33]. Some other important work related to Big Data privacy and security are discussed in [43, 48]. Trust and Big Data inter-dependencies and a research road-map is presented in [39].

Attribute based access control (ABAC) have received significant attention and several researchers have proposed important models. Xin et al presented a unified ABAC model configuring DAC, MAC and RBAC in [30]. Yuan et al [53] presented an authorization architecture and policy formulation for ABAC in web services. Wang et al [51] provided framework using logic based programming to model ABAC. XACML [10] is well used as a policy definition language and a processing architecture whereas SAML [9] is used for exchanging information. Hu et al also presented ABAC in [28] which provides a guide to ABAC definitions and considerations in [26]. NIST also proposed strategies to incorporate attributes in RBAC is presented in [32]. A significant work by Servos et al [41] introduced the notion of groups to ease the administration of attributes into the system, which inspired us to use groups in our work. Some other relevant access control models [12, 13, 16, 23, 31, 37, 38] in attributes based, role based and its administration can be found which have significantly influenced and inspired our contribution in this work.

6 CONCLUSION

In this paper we propose the first formalized attribute-based access control model for the Hadoop ecosystem, referred to as HeABAC. This model is an extension to the already existing Hadoop access control model (HeAC) which includes the authorization capabilities of core Hadoop and two important security projects, Apache Ranger and Sentry. The paper also outlines OT-RBAC model which is an RBAC extension to HeAC and predecessor to our work in this paper. Attribute based access control offers fine grained and flexible authorization which is an important requirement in multi-tenant Hadoop data lake. This model offers such capabilities as been illustrated using examples and use-cases in the paper. The novel concept of cross Hadoop services trust has also been introduced where inter Hadoop service interaction is primarily controlled using the trust relations as defined in the system. We have also proposed an implementation approach for HeABAC model using open-source Apache Ranger using context enricher and condition evaluators, and highlighted some administrative requirements in HeABAC. Some future extensions in this line of work may involve data ingestion security at HDFS data nodes level.

ACKNOWLEDGMENTS

Sincere gratitude is extended to James Benson, Technology Research Analyst at Institute for Cyber Security, UTSA, for his valuable comments and suggestions in this research paper. This work is partially supported by NSF CREST Grant HRD-1736209, NSF grants CNS-1111925, CNS-1423481, CNS-1538418 and DoD ARL Grant W911NF-15-1-0518.

REFERENCES

[1] 2016. Big Data: Securing Intel IT's Apache Hadoop Platform. http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/big-data-securing-intel-it-apache-hadoop-platform-paper.pdf
[2] 2016. Securing Hadoop: Security Recommendations for Hadoop Environments. https://securosis.com/assets/library/reports/Securing$_{H}$adoop$_{F}$inal$_{V}$2.pdf
[3] 2017. *Apache Hive.* https://hive.apache.org/
[4] 2017. *Apache Knox.* https://knox.apache.org/.
[5] 2017. *Apache Ranger.* http://ranger.apache.org/.
[6] 2017. *Apache Sentry.* http://sentry.apache.org/.
[7] 2017. *Cloudera.* http://www.cloudera.com/products/apache-hadoop.html
[8] 2017. *Hortonworks.* https://www.hortonworks.com/
[9] 2017. *SAML.* https://www.oasis-open.org/committees/tc$_{h}$ome.php?wg$_{a}$bbrev=security
[10] 2017. *XACML.* https://www.oasis-open.org/committees/tc$_{h}$ome.php?wg$_{a}$bbrev=xacml
[11] 2017 (Accessed: November 14, 2017). *IdentityForce.* https://www.identityforce.com/blog/2017-data-breaches.
[12] Mohammad A Al-Kahtani and Ravi Sandhu. 2002. A model for attribute-based user-role assignment. In *Proc. of ACSAC*. IEEE, 353–362.
[13] Smriti Bhatt, Farhan Patwa, and Ravi Sandhu. 2017. ABAC with Group Attributes and Attribute Hierarchies Utilizing the Policy Machine. In *Proc. of ABAC Workshop*. ACM, 17–28.
[14] Pietro Colombo and Elena Ferrari. 2015. Complementing mongodb with advanced access control features: Concepts and research challenges. In *Proc. of SEBD 2015*.
[15] Pietro Colombo and Elena Ferrari. 2015. Privacy aware access control for Big Data: a research roadmap. *Big Data Research* 2, 4 (2015), 145–154.
[16] Jason Crampton and George Loizou. 2003. Administrative scope: A foundation for role-based administrative models. *ACM Transactions on Information and System Security (TISSEC)* 6, 2 (2003), 201–231.
[17] Devaraj Das, Owen O'Malley, Sanjay Radia, and Kan Zhang. 2011. Adding security to Apache Hadoop. *Hortonworks, IBM* (2011).
[18] Philip Derbeko, Shlomi Dolev, Ehud Gudes, and Shantanu Sharma. 2016. Security and privacy aspects in MapReduce on clouds: A survey. *Computer Science Review* 20 (2016), 1–28.
[19] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. 2001. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)* 4, 3 (2001), 224–274.
[20] Maanak Gupta, Farhan Patwa, James Benson, and Ravi Sandhu. 2017. Multi-Layer Authorization Framework for a Representative Hadoop Ecosystem Deployment. In *Proc. of the 22nd ACM on Symposium on Access Control Models and Technologies (SACMAT)*. ACM, New York, NY, USA, 183–190. https://doi.org/10.1145/3078861.3084173
[21] Maanak Gupta, Farhan Patwa, and Ravi Sandhu. 2017. Object-Tagged RBAC Model for the Hadoop Ecosystem. In *Proc. of Data and Applications Security and Privacy XXXI: DBSec 2017, Philadelphia, PA, USA, July 19-21, 2017*. Springer, 63–81. https://doi.org/10.1007/978-3-319-61176-1$_{4}$
[22] Maanak Gupta, Farhan Patwa, and Ravi Sandhu. 2017. POSTER: Access Control Model for the Hadoop Ecosystem. In *Proc. of the 22Nd ACM on Symposium on Access Control Models and Technologies (SACMAT)*. ACM, New York, NY, USA, 125–127. https://doi.org/10.1145/3078861.3084164
[23] Maanak Gupta and Ravi Sandhu. 2016. The GURA$_{G}$ Administrative Model for User and Group Attribute Assignment. In *Proc. of NSS*. Springer, 318–332.
[24] Robert Hryniewicz. 2016. Best Practices in HDFS Autorization with Apache Ranger. https://hortonworks.com/blog/best-practices-in-hdfs-authorization-with-apache-ranger/. (2016).
[25] Robert Hryniewicz. 2016. Best Practices in Hive Autorization with Apache Ranger. https://hortonworks.com/blog/best-practices-for-hive-authorization-using-apache-ranger -in-hdp-2-2/. (2016).
[26] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. 2014. Guide to attribute based access control (ABAC) definition and considerations. *NIST Special Publication* 800, 162 (2014).
[27] Vincent C Hu, Tim Grance, David F Ferraiolo, and D Rick Kuhn. 2014. An access control scheme for big data processing. In *Proc. of CollaborateCom*. IEEE, 1–7.

[28] Vincent C Hu, D Richard Kuhn, and David F Ferraiolo. 2015. Attribute-based access control. *IEEE Computer* 2 (2015), 85–88.

[29] Xin Jin, Ram Krishnan, and Ravi Sandhu. 2012. A role-based administration model for attributes. In *Proc. of the First International Workshop on Secure and Resilient Architectures and Systems*. ACM, 7–12.

[30] Xin Jin, Ram Krishnan, and Ravi Sandhu. 2012. A unified attribute-based access control model covering DAC, MAC and RBAC. In *Proc. of IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 41–55.

[31] Xin Jin, Ravi Sandhu, and Ram Krishnan. 2012. RABAC: role-centric attribute-based access control. In *Proc. of MMM-ACNS*. Springer, 84–96.

[32] D Richard Kuhn, Edward J Coyne, and Timothy R Weil. 2010. Adding attributes to role-based access control. *Computer* 43, 6 (2010), 79–81.

[33] Rongxing Lu, Hui Zhu, Ximeng Liu, Joseph K Liu, and Jun Shao. 2014. Toward efficient and privacy-preserving computing in big data era. *IEEE Network* 28, 4 (2014), 46–50.

[34] David Nunez, Isaac Agudo, and Javier Lopez. 2014. Delegated Access for Hadoop Clusters in the Cloud. In *Proc. of CloudCom*. IEEE, 374–379.

[35] Owen O'Malley, Kan Zhang, Sanjay Radia, Ram Marti, and Christopher Harrell. 2009. Hadoop security design. *Yahoo, Inc., Tech. Rep* (2009).

[36] Navid Pustchi, Ram Krishnan, and Ravi Sandhu. 2015. Authorization federation in IaaS multi cloud. In *Proc. of the 3rd International Workshop on Security in Cloud Computing*. ACM, 63–71.

[37] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. 1999. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security (TISSEC)* 2, 1 (1999), 105–135.

[38] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. 1996. Role-based access control models. *IEEE Computer* 29, 2 (1996), 38–47.

[39] Johannes Sänger, Christian Richthammer, Sabri Hassan, and Günther Pernul. 2014. Trust and big data: A roadmap for research. In *Proc. of DEXA*. IEEE, 278–282.

[40] NIST Big Data Public Working Group, Security and Privacy Subgroup. 2017. DRAFT: NIST Big Data Interoperability Framework: Volume 4, Security and Privacy. *NIST Special Publication* 1500, 4 (2017).

[41] Daniel Servos and Sylvia L Osborn. 2014. HGABAC: Towards a formal model of hierarchical attribute-based access control. In *Proc. of International Symposium on Foundations and Practice of Security*. Springer, 187–204.

[42] Priya P Sharma and Chandrakant P Navdeti. 2014. Securing big data Hadoop: a review of security issues, threats and solution. *IJCSIT* 5 (2014).

[43] Jordi Soria-Comas and Josep Domingo-Ferrer. 2016. Big data privacy: challenges to privacy principles and models. *Data Science and Engineering* 1, 1 (2016), 21–28.

[44] Ben Spivey and Joey Echeverria. 2015. *Hadoop Security. Protecting your Platform.* " O'Reilly Media, Inc.".

[45] Bo Tang and Ravi Sandhu. 2013. Cross-tenant trust models in cloud computing. In *Proc. of 14th International Conference on Information Reuse and Integration (IRI)*. IEEE, 129–136.

[46] Bo Tang and Ravi Sandhu. 2014. Extending openstack access control with domain trust. In *Proc. of International Conference on Network and System Security*. Springer, 54–69.

[47] Bo Tang, Ravi Sandhu, and Qi Li. 2015. Multi-tenancy authorization models for collaborative cloud services. *Concurrency and Computation: Practice and Experience* 27, 11 (2015), 2851–2868.

[48] Omer Tene and Jules Polonetsky. 2012. Privacy in the age of big data: a time for big decisions. *Stanford Law Review Online* 64 (2012), 63.

[49] Huseyin Ulusoy, Pietro Colombo, Elena Ferrari, Murat Kantarcioglu, and Erman Pattuk. 2015. GuardMR: Fine-grained security policy enforcement for MapReduce systems. In *Proc. of ASIACCS*. ACM, 285–296.

[50] Huseyin Ulusoy, Murat Kantarcioglu, Erman Pattuk, and Kevin Hamlen. 2014. Vigiles: Fine-grained access control for mapreduce systems. In *Proc. of Big Data Congress*. IEEE, 40–47.

[51] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. 2004. A logic-based framework for attribute based access control. In *Proc. of Workshop on Formal methods in security engineering*. ACM, 45–55.

[52] Tom White. 2012. *Hadoop: The Definitive Guide.* O'Reilly Media, Inc.

[53] Eric Yuan and Jin Tong. 2005. Attributed based access control (ABAC) for web services. In *Proc. of International Conference on Web Services*. IEEE.

[54] Wenrong Zeng, Yuhao Yang, and Bo Luo. 2013. Access control for Big Data using data content. In *Proc. of International Conference on Big Data*. IEEE, 45–47.

[55] Jiaqi Zhao, Lizhe Wang, Jie Tao, Jinjun Chen, Weiye Sun, Rajiv Ranjan, Joanna Kołodziej, Achim Streit, and Dimitrios Georgakopoulos. 2014. A security framework in G-Hadoop for big data computing across distributed cloud data centres. *JCSS* 80, 5 (2014), 994–1007.