Using classification for role-based access control management

Nazia Badar, Jaideep Vaidya* and Vijayalakshmi Atluri*

CIMIC and MS/IS Department, Rutgers University, 1 Washington Park, Newark, NJ 07102, USA
Email: nbadar@scarletmail.rutgers.edu
Email: jsvaidya@rbs.rutgers.edu
Email: atluri@rutgers.edu
*Corresponding authors

Nino Vincenzo Verde
Department of Mathematics Italy, Roma Tre University, Largo S. Leonardo Murialdo, 1 Room 005 00146 Rome, Italy
Email: nverde@mat.uniroma3.it

Janice Warner
School of Business, Georgian Court University, 900 Lakewood Ave. Lakewood, NJ 08701-2697, USA
Email: warnerj@georgian.edu

Abstract: Access control is based on the specification of rights to resources. Role-based access control (RBAC) has emerged as one of the most robust security models which significantly simplifies administrative overheads. Despite all the compelling benefits that RBAC offers, it still lacks the ability to handle dynamic environment aspect and handling any unforeseen situations. Manual intervention becomes necessary when a user who is not previously defined in the system requests an access. For a system administrator, it becomes challenging to decide whether a submitted request should be honoured or not and how a new user can be added to the existing system in a secure manner. These issues have significantly increased the demand for new access control solutions that provide flexible, yet secure access. In this paper, we present an approach to facilitate automatic enforcement of access control policies when a new user is added to an existing access control system. Our approach is based on classification method. To evaluate the effectiveness of our approach we performed extensive experiments on both real and synthetic datasets. We compare the performance of our approach to another well-known approach that was proposed earlier to handle a similar
problem. Experimental results show that our approach performs very well. Moreover, we have found that our approach is relatively easier to implement.

**Keywords:** access control; classification; coalitions; collaboration; semantics.


**Biographical notes:** Nazia Badar is currently pursuing her PhD from the MSIS Department at Rutgers Business School, Rutgers University. She completed her undergraduate and Masters from the University of Karachi. Her research interests include information privacy and security, optimisation, data mining and analysis. Her current research is based on using data mining methods for enhancing the security aspect of deployed access control systems. She is always interested in problems that have real-world impact, solutions that are simple, but are based on practical foundations.

Jaideep Vaidya is an Associate Professor in the MSIS department at the Rutgers University. He completed his PhD in Computer Science from the Purdue University. His general area of research is in data mining, data management, security, and privacy. He has published over 100 technical papers in peer-reviewed journals and conference proceedings, and has received four best paper awards from leading conferences in data mining, databases, digital government, and informatics.

Vijayalakshmi Atluri is a Professor in the Computer Information Systems from the MSIS Department, and research director for the Center for Information Management, Integration and Connectivity (CIMIC) at the Rutgers University. Her research interests include information security, privacy, databases, workflow management, spatial databases and distributed systems. She has published over 150 technical papers in such journals and conferences as the IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Knowledge and Data Engineering, ACM Transactions on Information Systems Security, *The VLDB Journal, Distributed and Parallel Databases: An International Journal*, IEEE Symposium on Security and Privacy, IEEE Conference on Data Engineering and ACM Conference on Computer and Communication Security.

Nino Vincenzo Verde is a Post-Doc Researcher at the Department of Computer Science of Sapienza University of Rome. He completed his PhD in Mathematics from the University of Roma Tre in 2011, and his Masters in Computer Science from the Sapienza University of Rome in 2007. Nino’s main interests include digital forensic, critical infrastructure protection, information warfare, role based access control (RBAC), data mining and machine learning. Nino is affiliated with the Research Center of Cyber Intelligence and Information Security (CIS) of Sapienza.

Janice Warner works as the Dean of the School of Business at Gregorian Court University. She completed her undergraduation and Masters in
Using classification for role-based access control management

Electrical Engineering from the Columbia University led her to a career in Telecommunication and Information Networking with companies including Bellcore, Siemens AG, and Telcordia Technologies. She completed her MBA and Doctorate in Management from Rutgers University School of Business where she focused on Information Technology, Business Analytics, and Electronic Commerce. Her research interests include management of technology and higher education pedagogy for business.

1 Introduction

Access control is based on the specification of rights to resources (also known as objects) by subjects (users or processes), by groups or by roles. To date, several access control models and their extensions have been proposed in the literature of information security. However, role-based access control (RBAC) has emerged as one of the most robust security models to meet diverse access control requirements. Direct mapping of individual users to access control lists attached to resources is burdensome task for security administrator. RBAC, on the other hand, significantly simplifies administrative overheads by assigning group of permissions to a user via role.

Management of user authorisations is a critical aspect of maintaining a secure system. In a static environment, the resources, subjects and roles are well known and can be administered. In a dynamic or a multi-domain environment, these fundamental building blocks of access control are not so well established. In particular, resources may be requested by describing what is needed (rather than by specific name) and users may be completely new to a system (without prior access history, a user id, group id, or role id). Despite all the compelling benefits that RBAC offers, it still lack the ability to handle dynamic environments aspect and handling any unforeseen situations. For example, consider a security environment where an RBAC is already deployed and all the user-role assignments are defined in the system. In such system, when a new user requests an access and the resources they need may vary, traditional access control mechanisms lacks ability to handle these kind of requests effectively. In this situation, it should not be necessary to entirely change the established local access control mechanisms for a new user. Nor should it be necessary to centrally administer the policies that apply for handling these type of changes. Instead, to support dynamic environments, it should be possible to automate the generation and enforcement of access control policies to meet the new user needs, while the existing mechanism remain intact.

Usually, when a new user is added to an existing system, manual intervention of security administrator becomes necessary to decide whether a user should gain access to a requested resource or not. In an RBAC-based security environment, if prior access permissions of a new user are not known, or they are not relevant from requested access’s perspective, it becomes difficult for the security administrator to determine a suitable role for a new user through which a user can acquire access privileges that are required to perform a particular job. From security perspective, it is important to ensure that granted access to a new user is consistent with the organisational access control policy. However, a security administrator can make a mistake in assigning appropriate
role to a new user. If an assigned role to a new user has lesser permissions than the
required permissions for completing a certain job, then this causes user’s failure to
perform the assigned duties. Therefore, when an RBAC is in place, determining a
suitable role for a new user is a critical yet challenging task. Moreover, the prior
authorisations data of a new user does not exist in the system and the permission
data of external user is not always relevant in determining a locally suitable role for
a user.

Attribute-based access control (ABAC) is based on the idea of making access
tool control decisions on the basis of attributes of a user who submits an access request
(Frikken, Atallah and Li, 2006; Jin, Sandhu and Krishnan, 2012). Using attributes
simplifies administration because new users do not have prior access privilege history
available in the system. For such users, access rights can be determined purely on
the basis of attributes and these attributes can apply to many different users, who
would present their attributes through the use of credentials. In traditional systems,
when access control decisions are made for a new user, usually the extreme actions
are taken. First extreme may be to require the new user to produce credentials that
correspond to all credentials held by individuals who have rights to the requested
object. This preserves the organisation’s resource security to the highest degree but
limits the access permissiveness needed for a new user. At the opposite extreme,
there could be no credentials required to enable maximum permissiveness but this
leads to little security. Determining a fine balance between those two extremes is
challenging task. Previously, in Warner, Atluri and Mukkamala (2005b), an approach
based on matching of user attributes to that of requested object was given to
eable dynamic sharing of organisational resources with users which are not defined
within the implemented system. While a part of the solution, ABAC and credential
discovery are not sufficient. It is a useful mechanism, but the problem remains of
determining what attributes will be considered significant and are acceptable by an
organisation.

Significant attributes for a role are mainly the distinguishing attributes that are
common to users in the role but rare outside. In this paper, we present an approach
based on classification method to automate the process of identifying significant
attributes for a role and then using those attributes to determine whether a user qualifies
for a particular role or not. The goal of this approach is to ensure that the access
control decision reflects the inherent local access control policy of the organisation.
Our approach is aimed to discover a balanced set of credential requirements that provide
an optimal level of security and permissiveness based on the need of a new user without
defeating the purpose of implemented access control system. Our approach is based on
first building classification model for each role in the system. The model is then used
for evaluating whether an access request submitted by a user can be honoured or not
and how it can be done in a secure manner.

When an authorisation is requested in a quick manner, immediate analysis of
policies becomes issue of vital importance. Manually performing analysis of policies
is quite daunting task for security administrators. However, we believe that the
organisation’s local policies that governs access to its resources are inherent in
access history of users. We, therefore, use this data to discover hidden yet important
information to extract access control policies for a requested resource. Our approach
relies on using classification method for generating access control policy model. Model
is based on required attributes to access a requested object and then using that model
for testing credentials of new user against the model to determine whether a user qualifies for a requested permission.

Our work is closely related to the prior work given in Warner et al. (2007). We performed extensive experiments using both synthetic and real datasets for comparing performance of our approach to the one given in Warner et al. (2007). Throughout this paper, we refer to their approach of identifying significant attributes for a role, as Semantics-based method. In this paper, we also present the results of our experiments. In addition to the problem of determining suitable role for a new user, our approach can also be used to facilitate coalition-based access control. Moreover, our approach can be used to facilitate the process of automating the process of misconfiguration detection and its removal from the deployed RBAC systems.

Remaining paper is organised as follows. Framework and preliminaries are discussed in Section 2. Our approach is described in Section 4. Semantics-based method is briefly in Section 4.3. Related work is discussed in Section 7. Experimental evaluation is discussed in Section 6, while in Section 8 we discuss future research directions and conclude the paper.

2 Framework

This section introduces the preliminaries. In particular, we review role-based access control, and user and object attributes.

2.1 Role-based access control (RBAC)

Since our framework assumes that RBAC is in place within each organisation participating in a coalition, we review the definition of RBAC. We further assume that the roles defined under the RBAC system are based on job functions. This is a necessary assumption for enabling semantic matching of information resources to groups of users who perform a certain function.

Definition 1: [RBAC] RBAC has been formally defined by NIST (Ferraiolo et al., 2001) as follows:

- $U, R, OPS$ and $O$ are sets of users, roles, operations and objects, respectively.
- $UA \subseteq U \times R$ is a many to many user to role assignment relation.
- $P$ (the set of permissions) $\subseteq \{(op, o)|op \in OPS \land o \in O\}$.
- $PA \subseteq P \times R$ is a many to many permission to role assignment relation.
- $assigned\_users(r) = \{u \in U|(u, r) \in UA\}$, the mapping of role $r$ onto a set of users.
- $assigned\_permissions(r) = \{r \in R|(u, r) \in UA\}$, the mapping of role $r$ onto a set of permissions.
- $assigned\_objects\_per\_permission (r, p) \rightarrow \{o \in O |p = (op, o)\}$, the permission-to-object mapping which gives the set of objects associated with permission $p$ for a given role.
assigned_objects(r) \rightarrow \bigcup \{ assigned_objects_per_permission(r, p) | p \in assigned_permissions(r) \}, the permission-to-object mapping which gives the set of all objects associated with any of the permissions assigned to role r.

- RH \subseteq R \times R is a partial order on R called the role hierarchy or role dominance relation.

2.2 Attributes

We assume that both users and objects are associated with specific attributes. An attribute consists of a name and a value pair, \((a_i, v_i)\) and is referred to by its name, \(a_i\). The set of all attributes defined for an organisation is \(\Lambda\).

2.2.1 User attributes

User attributes that may be semantically relevant to objects describe what the user is capable of doing, has done or is assigned to do. Some of these attributes may be drawn from certifiable credentials (Housley et al., 2002) possessed by the users indicating, for example, that a user has completed a degree or a training program. Others may be explicitly assigned internally by the organisations to indicate, for example, experience the user has had or their current assignments. The set of all user attributes is denoted as the user attribute base \(UAB\). We use \(ua_i = \{(a_i : v_i), (a_j : v_j), \ldots\}\) to denote all the attributes associated with a specific user \(u_i\).

Example 1: Tom, who works at LM, has the following attribute set: \(ua_{To m} = \{(\text{hasDegree: bachelors}), (\text{performsJob: software}), (\text{assignedTo: projectBlue}), (\text{hasExpertiseIn: java}), (\text{officeLoc: NVC1})\}\).

2.2.2 Object attributes

Objects attributes are either explicitly defined or automatically derived using text analysis as in Sanderson and Croft (1999). Object attributes might be classified by keywords or content, in terms of their type (e.g., executable, spreadsheet), attributes of their author/owner, or in relationship with each other. Object attributes include object id \(o_i\), attribute name \(a_i\), and attribute value \(v_i\). The set of all object attributes is denoted as the object attribute base \(OAB\). We use \(oa_i = \{(a_i : v_i), (a_j : v_j), \ldots\}\) to denote the set of object attributes associated with a particular object \(o_i\).

Example 2: LM has an object ‘Component 1 software’ \(C1\) that has the following attribute set: \(oa_{C1} = \{(\text{hasContent: java}), (\text{hasContent: financial}), (\text{hasContent: software}), (\text{createdUnderProject: Blue})\}\).

3 Classification

The problem of classification has long been recognised as one of the most important data mining and machine learning problems. It is applicable in a wide range of science and technology domains. The goal of classification is to build a model based on important structural properties of data with defined class labels and using that model
Using classification for role-based access control management
to predict the class label for new examples or previously unseen data instances. From
the perspective of access control problem that we are addressing in this paper, we
are employing classification method to build a model for predicting role membership
status of new users. The classification model for any role is constructed by using
the information about credentials of existing users who have that role, and also the
credentials of existing users who do not have that role. In other words, labelled training
data (supervised learning) for roles are first used for model building process and later
those models are used to classify a user for a potential role. Figure 1 is an example
that shows how we are employing classification method. The input for building the
classification model is the labelled access control data of users. In the given example,
there are eight users and five attributes. Attributes are also called decision variables.
It is mentioned earlier in Section 1, role membership status and credentials are stored
in form of a binary matrix. Each row represents a user in Figure 1. For columns
representing attributes, ‘1’ in any cell represents that a user has that attribute and ‘0’
represents that a user does not have that attribute. For columns representing roles, ‘1’
in any cell represents that a corresponding user has that role and ‘0’ represents that a
user does not have that role. New user can become a member of a role if his credentials
qualify for a role. This is done by using the classification model for a particular role.
For example, in Figure 1 it can be seen that the users u10 and u11 are new users
and their role membership is unknown. Therefore, attributes of users u10 and u11
are checked against the classification model for the requested role. Role membership
status ‘0’ is predicted for u11 which means that his credentials do not qualify for the
requested role. However, role membership status ‘1’ is predicted for user u11 which
means credentials of this user qualifies for the requested role.

Figure 1 The process of classification (see online version for colours)

Some other situations where the problem of classification is applicable include: loan
granting decisions, credit card fraud detection, network intrusion detection (NIDS),
target marketing, medical diagnosis, speech recognition, handwriting recognition,
and document classification. In document classification problem, content within the
document is analysed to classify by topic. Internet search engines and organisational
intranets are popular examples of environments where document classification
approaches are widely used. Target marketing is another example of classification problem that strongly depends on underlying classification models used by firms to identify potential customer segment that can be reached for successful promotional activities and business profitability. The purpose of the classification model in this setting is to predict buying decision of potential customers on the basis of their past buying behaviours, interests, age group, ethnicity, etc. Consider an example from the Healthcare industry. This industry has huge amounts of data. That data can be of great benefit if it is put to an optimal use. For example, a medical history along with other clinical variables about a new patient can be used by healthcare providers for monitoring health conditions of the patient and for classifying that patient into a known set of diseases or any other related health problems. Predictive models for such classifications are generated by utilising patient data with known classification outcomes. In a banking application, when a bank receives a new loan request, credentials of a requesting party are evaluated by using predictor variables, such as education, age, income, and many more predictor variables to decide, whether a loan should be granted or not.

Several classification methods have been proposed in the literature of data mining and machine learning. For example: ID3 Quinlan (1986), C4.5 Quinlan (1993), algorithms are based on decision tree model for classification. SVM Burges (1998), on the other hand, is based on linear classification model. PRISM Cendrowska (1987) and Decision Tables Fisher (1996) use rule based model for classification. In this work we are employing three different classification methods. We are using J.48 Quinlan (1993) decision trees, Random forest Breiman (2001), and a meta-learner Support Vector Machine Burges (1998) algorithms for generating classification model for the organizational roles. Below, we briefly discuss each of these classification approaches.

3.1 Decision trees

Decision tree is commonly used as a decision support tool that uses a tree-like graph or model of decisions and possible outcome of those decisions for the decision analysis purpose. Tree-like structure is grown in such a way that the leaf node of a tree represents outcome or a class label, whereas internal nodes represent test on attributes. Number of outgoing edged from each internal node represents number of possible outcomes that test can have. Classical implementation algorithms for decision tree construction are: ID3, C4.5, and C.5 (Quinlan, 1993, 1996). These algorithms are generally based on divide and conquer strategy where problem of learning classification structure from known set of observation leads to a systematic construction of a decision tree.

3.2 Random forest

Ensemble-based learning method for classification uses multiple models to obtain better predictive performance than could be obtained from any of the constituent models. Recently, these approaches have attracted significant interest from data mining and machine learning community. Random forest (Breiman, 2001) is also an ensemble-based learning method for classification. Moreover, it is a supervised learning technique. At the time of training a model, large number of decision trees are constructed. The outcome or class label of the model is determined from the mean or a mode outcome of the individual trees. One of the main strengths of this classifier is that
the performance of Random forest classifier remains consistent even in the presence of noise. Thus making it suitable for real-world datasets containing many outliers, missing values, and even other errors.

3.3 Support vector machines (SVM)

In classification, the goal of a SVM-based classifier is to establish a two-dimensional mapping space by introducing a boundary line (hyperplane) that separate data points of a particular class from data points of another class. Suppose there is a two-class classification problem. Labelled data points (supervised learning) are first used to build SVM-based model and when a class label for a new example or a data point has to be determined, its attributes are checked against that model in order to map a new data point to the correct side of the hyperplane.

4 Identifying required role attributes using classification

As discussed in Section 1, our approach to identify significant attributes for a role membership relies on classification method. Classification model constructed from the existing data is later used for testing credentials of a new user for a role membership by matching them against the classification model of a role. In real life situations, security administrator does not have understanding of various aspects of role configuration. Therefore, it is highly probable that she may pick ill-suited role for a new user for granting any requested permission in RBAC security environment. An ill-suited role is a role through which a user may get lesser than necessary permissions or extra permissions than necessary permissions. When more than necessary permissions are given to a user, security configuration errors are introduced in the deployed system leaving the security loop holes open for attack by malicious users. On the other hand, if less than necessary permissions are given to a user, the task assigned to a user cannot be completed. Our approach based on classification is to ensure that the addition of a new user to an existing system is in compliance with the access control policies that are governing the deployed system.

In RBAC-based security system, user acquires a permission to access a resource via role (Frikken, Atallah and Li, 2006). In our approach to facilitate coalition-based access control, we are assuming that for ensuring the enforcement of security policies, RBAC is already deployed in the resource owning organisation. For identifying the set of required user credentials for a role, we first build classification model for each role by using user attribute base ($UAB$), and role assignment data ($UA$). $UAB$ and $UA$ are discussed in Section 2.2.1 and in Section 1. Each role’s model is computed by using the $UAB$ of all users regardless of their role membership status. Note that our classification-based approach is not sensitive to the choice of classification algorithm. Once the classification model for each role is constructed, these models are then stored and are later used for determining whether a new user qualifies for a particular role or not.

Credentials of a new user are tested against the classification model for a role through which a user can acquire a requested authorisation. Generally, to prevent misuse of any organisational resources, established local access control policies allow only those users, to access a particular resource who possess attributes comparable to
the attributes of a resource. When a new user is added as a result of coalition formation, organisation wants similar policies to be exercised while sharing resources with external users takes place. Therefore, it is important to check that external user also possesses required credentials to access a requested resource.

The overall process to add a new user to existing system involves following three steps:

**Step 1. Building classification model for each role.** This step is executed to build separate classifier for each role $r_m \in R$, where $R$ represents set of roles that are locally deployed within the organisation. This step can be executed just once, as shown in Algorithm 1. All models are then stored for later use. Note that if any changes in the configuration of any role are observed, then the model for that role can be updated either incrementally or by rebuilding a model.

**Step 2. Selecting roles having requested permission.** When an access request is received from a new user to access particular object, or set of objects, this step is executed to identify set of all roles through which a permission can be granted. Note that we are assuming that RBAC is in place, therefore permissions are acquired by a user through role. All those roles through which a user may acquire a permission to access requested object are stored in $\text{candidateRoles}$ list, as shown in Algorithm 2.

**Step 3. Classifying external user’s attributes for roles.** Though there could be multiple roles through which a permission to access requested object can be granted to a new user, but it is important to ensure that the user should be assigned to a role that he qualifies for. To determine a secure role for a new user, his credentials are checked against the model of each role retrieved in $\text{candidateRoles}$ list. If there is any role in the list of $\text{candidateRoles}$ for which he qualifies on the basis of his credentials then that role is assigned to new user. However, if he does not qualify for any role in a list of $\text{candidateRoles}$, access request would be denied.

The details of three steps are given in following subsections.

### 4.1 Building classification models to identify required attributes for roles

This step is carried out just once. Essentially, we need attributes classification model for each role to begin with. Classification model for all roles is generated and stored for future use at this step. Algorithm 1 shows how this step is executed.

Given a role $r_m \in R$, goal of attributes classification model is to identify combination of attributes which are important for categorising user into the role $r_m$. Having an RBAC system in place, local user acquires permission to access any object through role. Thus, if $PermO_j$ is requested by a user, to access object $O_j$, we assume that there would be atleast one role that has permission to access object $O_j$. Generally, in RBAC-based environment, it is possible to have roles with overlapping permissions. Therefore, if credentials of an external user do not match with required credentials for one particular role then they are tested for remaining $\text{candidateRoles}$. By $\text{candidateRoles}$, we mean those roles which contains permission to access requested object.
4.2 Selecting candidate roles

When an access request of an external user is received, all roles through which a requested permission could be assigned are stored in \textit{candidateRoles} list. However, out of all \textit{candidateRoles}, user is assigned to a role that has a match with attributes of that user. This matching is done at next step. Recall that \textit{candidateRoles} has only those roles which contains permission requested, therefore if any role from \textit{candidateRoles} is assigned to a user automatically a user will acquire requested permission through it. Algorithm 2 shows how this selection is done.

4.3 Evaluation of credentials of an external user

Once the set of \textit{candidateRoles} is identified, attribute set of external user has to be categorised into a role. If a given request is $au_x, \text{Permobj}_y$, where $au_x$ represents credentials of an external user $u_x$ and $\text{Permobj}_y$ is the permission requested by a user $u_x$ to access $\text{obj}_y$, user $u_x$’s credentials $au_x$ are checked against the attribute classification models of roles containing permission $\text{Permobj}_y$. If $au_x$ qualifies for membership into any role that has permission to access $\text{obj}_y$, then that role is assigned to an external user $u_x$. Essentially, $u_x$ would get $\text{Permobj}_y$ via assigned role. However, if $au_x$ do not qualify for any of the roles having \textit{candidateRoles}, then the request is denied.

\begin{algorithm}
\caption{Selecting Candidate Roles for New User}
\begin{algorithmic}[1]
    \Require $\text{Perm}(O_j)$, represents requested permission to access object $j$.
    \Require $\text{PA}$, represents permission role assignments.
    \State $\text{candidateRoles} \leftarrow \phi$
    \State \{to discover all roles having permission $\text{PermO}_j$ to access object $O_j$\}
    \For {each role $r$ in $\text{PA}$}
        \If {$r$ has $\text{PermO}_j$}
            \State $\text{candidateRoles} \leftarrow r$
        \EndIf
    \EndFor
    \State \textbf{return} \text{candidateRoles}
\end{algorithmic}
\end{algorithm}

5 Identifying required attribute set using threshold value

In this section, we present the basic framework and description of an approach to which we are comparing our work with. This approach uses object and user attribute
semantics for determining user attribute-value pairs or candidate attributes of users (caur_{significant}) that characterise a role. In this paper, we refer to this approach as semantics-based approach.

5.1 Framework for semantics-based approach

In this section we present some preliminary concepts that are unique to semantics-based approach.

5.1.1 Concept hierarchy

Semantics-based approach employs concept hierarchies to link the different attributes and compare their values. A concept hierarchy is a graphical notation for representing knowledge using interconnected nodes and arcs. In a concept hierarchy, a more specific concept(s) is represented as a descendant(s) of its more general concept(s).

A concept hierarchy, $C_i$, consists of a partially ordered ($\preceq$) set of concepts. Given two concepts $c_1, c_2 \in C_i$, the following four possible relationships are considered between them: subClass($c_1$) = $c_2$ for the relationship where $c_2$ is a more specialised concept; eq($c_1$) = $c_2$ where $c_1$ and $c_2$ are equivalent concepts (synonyms); sup($c_1$) = $c_2$ where $c_2$ is the next more general concept than $c_1$ (i.e., separated by only one link in the concept hierarchy); and com($c_1$) = ($c_2$) where sup($c_1$) = sup($c_2$). This latter relationship means that $c_1$ and $c_2$ are compatible. Note that several other relationships among concepts are defined in Owl web ontology language guide (Horridge et al., 2004). Discussed approach utilises only the relationships specified above. Note that the concepts can be either the attribute names or their values.

Figure 2 shows an example concept hierarchy for the general concept software. From the figure, sup(Financial) = Functions and eq(Financial) = Fiscal are examples of relationships. Note that when two concepts are synonyms they are enclosed in a box. Finally, com(Financial) = Display indicating that these two concepts representing have a common parent and thus represent more specific aspects of the same general concept.

Figure 2  Example concept hierarchy for concept software
Definition 2: We say an attribute \( a_i \) is associated with (or belongs to) a concept hierarchy \( C_j \), denoted as \( a_i \rightarrow C_j \) if its value \( v_i \) is an included concept in \( C_j \).

It is assumed that every attribute in \( \Lambda \) with a non-numerical attribute value must be associated with one or more concept hierarchies. For example, the values for user attributes \( \text{performsJob} \) and \( \text{hasExpertiseIn} \) are concepts in the concept hierarchy for the concept \( \text{software} \). Likewise, the values for the object attributes \( \text{hasContent} \) or \( \text{hasKeyword} \) are also concepts in the concept hierarchy for the concept \( \text{software} \). Note that the degree of specialisation of a node increases with its distance from the root. For example, in Figure 2, \( \text{Java Code} \) is more specialised than \( \text{Code} \) which in turn is more specialised than \( \text{software} \).

5.1.2 Attribute linking

For the comparison of two attributes and their associated values, semantics-based approach needs to know if the attributes use the same vocabulary. It is achieved by associating attribute names with concept hierarchies where the association indicates the concept hierarchy from which the attribute values are drawn. Attributes are comparable if they are associated with attribute values that are from the same concept hierarchy. Therefore, if any attributes are comparable attributes they are also linked.

Semantics-based approach requires two types of linking:

1. **referential linking** that links a user attribute name and an object attribute name when associated with the same concept hierarchy;
2. **synonymous linking** that links two user attribute names that are synonyms.

Whether it is referential or synonymous linking, the approach uses \( T(a_i, a_j) \) to represent the linking between two attribute names \( a_i \) and \( a_j \). The linking relationship is transitive. That is, if \( T(a_i, a_j) \) and \( T(a_j, a_k) \), then \( T(a_i, a_k) \).

**Definition 3:** \( T(a_i, a_j) \) is a referential linking if \( a_i \) is a user attribute and \( a_j \) is an object attribute such that \( a_i \rightarrow C_i \land a_j \rightarrow C_j \).

**Definition 4:** \( T(a_i, a_j) \) is a synonymous linking if both \( a_i \) and \( a_j \) are user attributes such that \( eq(a_i) = a_j \).

Referential linking is used to compare user and object attribute values to extract the necessary attribute-value pairs for a role.

Synonymous linking is used to link a user attribute-value requirement to an attribute-value received from a new user or external user. This is done when the attribute name in the credential is not exactly that used by the resource owning organisation but is a synonym of an attribute name that is used.

5.2 Generating required user attributes for roles

This approach assumes that the user should be authorised to access only those objects within the organisation which matches semantically with the user attributes. Semantic match is defined as:
Definition 5 [Semantic match] [User-object attribute semantic match]: There exists a
semantic match between a user attribute $a_i$ and an object attribute $a_j$ iff
$(\top(a_i, a_j)) \land (eq(v_i) = v_j) \lor (\text{subClass}(v_j) = v_i))$.

Example 3: For example, the object ‘Component 1 Software’ has the attribute
`hasContent` which draws its values from the concept hierarchy for `software`
shown in Figure 2. A user, Elise, has an attribute `hasExpertiseIn` whose
values are also drawn from the concept hierarchy for `software`. Thus, the object
attribute and the user attribute are linked. Now suppose the value of the object
attribute, `hasContent`, is `functions`. Suppose also that the value of Elise’s
`hasExpertiseIn` attribute is `financial`, then there is a semantic match between
the two attributes because Elise’s attribute value is a subclass of the Component
1 Software’s attribute value in the software concept hierarchy. Note that if Elise’s
attribute value for `hasExpertiseIn` had also been `functions`, there would have
also been a semantic match between the two attributes. However, if Elise’s attribute
value for `hasExpertiseIn` had been `software`, there would be no semantic match
because `subClass(function) \neq software`. The inclusion of `subClass` is to allow,
for example, users with `hasExpertiseIn` in Java code to access objects with
attributes `Code` or `Software`.

However, there is more to the process than simply performing semantic matches. To
determine attribute requirements for a role, the following four step process is executed.
This overall process is briefly defined below.

**Step 1. Discovering user attributes that are semantically related to object attributes:**
For each user who is currently a member of the role, consider each of their attribute-
value pairs contained in the UAB. This is the candidate attribute-value pair set for
that user. For each attribute-value pair, determine if there is a semantic match between
it and an attribute of any of the objects to which the role has privileges. If there is no
semantic match, remove the attribute-value pair from consideration by taking it out
out of the candidate-value pair set for that user.

**Step 2. Merging candidate attribute-value pair sets:** This part of the process involves
combining user candidate attribute-value pair sets that are the same or similar. The
similarity is determined by finding pairs that have the same attribute names but
different attribute values. The values are compared and if there is a relationship as per
the concept hierarchy associated with the values, they may be merged.

**Step 3. Pruning attribute-value pair sets by assessing significance of attribute-value
pairs:** For each candidate attribute value pair set, determine if it is significant for that
role. An attribute-value pair is significant if it is a characteristic of a high percentage
of members of the role and is a characteristic for only a small number of non-members
of the role. Only those attribute-value pairs that are significant for the role are kept
for further consideration. The result is a set of candidate attribute-value pair sets that
are significant to the role.

**Step 4. Checking attribute requirements across roles:** Finally, the candidate attribute-
value pair sets of all roles are tested if they obey certain rules when considering
the role hierarchy, which involves ensuring that related roles meet certain attribute
requirements rules. This is a semi-automated process. Note that new attribute-value pairs may be added during this step.

The details of the four steps are presented in the following subsections.

5.3 Semantic matching of user attributes to object attributes

This step involves extracting and semantically matching user attributes (and their values) for role members to object attributes (and their values) for objects for which the role has some permission. To derive the attribute requirements for membership in role \( r \), we need the set of objects for which permissions are assigned to the role, such objects \( \text{assigned}_\text{objects}(r) \) and the set of users assigned to the role, \( \text{assigned}_\text{users}(r) \) as per Definition 1.

**Definition 6** [Candidate object attributes]: The candidate object attribute set for a role \( r \), \( cao^r \) = \( \{ \cup_{o_i \in \text{assigned}_\text{objects}(r)} a_{o_i} \} \).

Essentially, for each of the object \( o_i \in \text{assigned}_\text{objects}(r) \), all the attributes associated with that object \( a_{o_i} \) in OAB are extracted. The union of all the attributes for all the objects in \( \text{assigned}_\text{objects}(r) \) make up the candidate object attribute set \( cao^r \) for role \( r \).

**Definition 7** [Candidate user attributes]: The candidate user attribute set for user \( u_i \) of role \( r \), \( cau^r_i \) = \( \{ a_j | a_j \in au_i \land u_i \in \text{assigned}_\text{user}(r) \land a_j \text{ has a semantic match to some } o_k \in cao^r \} \).

For each of the users \( u_i \in \text{assigned}_\text{users}(r) \), extract user attribute set, \( au_i \). For each of the attributes \( a_j \in au_i \), check whether there is a semantic match (as per Definition 5) between them and any of the object attributes in \( cao^r \). If there is a semantic match, the attribute \( a_j \) and the value \( v_j \) of the matching object attribute are added to the candidate user attribute requirement set for user \( u_i \) under role \( r \), \( cau^r_i \). If a user has no semantic matches to any of the object attributes, that user is flagged since they may be incorrectly assigned to the role. Likewise, if there is an object whose attributes have no semantic matches to any of the users assigned to the role, it is also flagged as an inappropriate object for the role.

**Definition 8** [Candidate role attributes]: The candidate role attribute set for role \( r \), \( cau^r \) = \( \{ \cup_{cau^r_i} | u_i \in \text{assigned}_\text{users}(r) \} \).

**Example 4:** Figure 3 shows a role, Component 1 Software Developer, which is a specialisation of the role Software Developer (see Figure 4). This role has three members, Tom, Elise, and Harry and is assigned permission for all files for Component 1 code. The attributes of Tom, Elise, and Harry as well as the Component 1 Code object are shown in the figure. When comparing Tom’s attributes to those of Component 1 Code, there are referential links between the user attribute performsJob and the object attribute hasContent, the user attribute hasExpertiseIn and the object attribute hasContent, and the user attribute assignedTo and the createdUnderProject object attribute. Therefore, the candidate attribute set
when just looking at Tom as the first user assigned to the software developer role would be $\text{cau}^{\text{SD}} = \{(\text{performsJob:software}), (\text{assignedTo:Blue}), (\text{hasExpertiseIn:java})\}$. Tom’s attributes, hasDegree:Computer Science and hasExpertiseIn:C++, were removed from consideration since there is no semantic match between those attributes and attributes of the objects.

For Elise, there is a match between her attribute performsJob and the object attribute hasContent, the user attribute hasExpertiseIn and the object attribute hasContent, the user attribute assignedTo and the createdUnderProject object attribute.

Harry has no attributes which match the attributes of the objects. This should be a concern since it appears that Harry may not belong to the role. Harry’s lack of semantically matched attributes should be flagged for further analysis. The empty set is not added as a candidate.

Therefore, the candidate attribute user sets for the Component 1 Software Developer role ($\text{C1SD}$) can be expressed as: $\text{cau}^{\text{C1SD}} = \{(\text{performsJob:software}), (\text{assignedTo:Blue}), (\text{hasExpertiseIn:java})\}, \{(\text{performsJob:software}), (\text{hasExpertiseIn:financial functions}), (\text{assignedTo:Blue})\}, \{(\text{performsJob:software}), (\text{assignedTo:Blue})\}$. 

Figure 3 Component 1 software developer role, members, and permissions (see online version for colours)

Figure 4 Partial role hierarchy for LM systems
Blue}). Note that the ‘;’ represents ‘or’. A set of users in the role have either of the sets of attribute-value pairs.

Now suppose the role Component 1 Software Developer also has access to an ethics statement, also shown in the figure. This file has no attributes that are linked to the user attributes of Tom, Elise, or Harry. As such, it does not add or detract from the candidate attribute requirements for the role. However, it can be flagged as an object that may not belong to the role since it does not appear to be semantically related.

5.4 Merging candidate attribute-value pairs

The second step for deriving the user attribute requirements for a role involves merging the sets of unique attribute-value pairs, $as(i)$, contained in $cau_{all}$ if there is more than one set of attribute-value pairs. If all the users in the role had exactly the same attribute-value pairs, there would only be one unique set. However, it is very likely that different users have different attribute-value pairs. In Step 1, we have a set of attribute-value pairs of every user who is a member of a role. These sets are reduced to the number of sets such that only members of a role who were assigned for distinct reasons are represented by different sets. This is done through merging of the candidate attribute-value pairs when users have the same attributes but different values for these attributes.

Merger begins by comparing every pair of sets within $cau_{all}$, $as(i)$ and $as(j)$, that have exactly the same attribute names but different attribute values. Let the unequal values be $v_i$ and $v_j$. If $v_i = sup(v_j)$ or $v_j = sup(v_i)$ then the sets can be merged as one set with the equal pairs being included along with either $v_i$ or $v_j$, respectively. If $com(v_i) = v_j$, and if $v_i$ and $v_j$ represent all the children of $sup(v_i)$, then the value of $a_k$ is replaced with $sup(v_i)$. If $v_i$ and $v_j$ do not represent all the children of $sup(v_i)$, then the value of $a_k$ is replaced with a concatenation of the values $v_i$ and $v_j$, $(a_k : [v_i, v_j])$ which represents that either $v_i$ or $v_j$ is acceptable values for attribute $a_k$.

Example 5: An example would be $as(1)$ and $as(3)$ where $as(1) = (a_1 : v_3, a_3 : v_5)$ and $as(3) = (a_1 : v_1, a_3 : v_3)$. Both $as(1)$ and $as(3)$ have the same attributes ($a_1$ and $a_3$) but the values of $a_1$ are different. The merger proceeds pairwise between sets of this type. Let us say that $com(v_5) = v_1$, then $as(1)$ and $as(3)$ can be merged with a new set that includes the multiple values for $a_1$ that are acceptable. For example, $as(3)$ is removed from $cau$ and $as(1)$ is replaced by $as(1) = (a_1 : [v_1, v_3], a_3 : v_5)$.

For example, suppose we have a Project Manager role for Project Blue, a software project. Some members might be part of that role because they have a college degree and project management experience while others might be part of the role because they have project management certification and have worked on software projects. The attribute and value pairs, (hasDegree: bachelors) and (hasExpertiseIn: project management) are different from (hasCertificate: Project Management Institute) and (hasExpertiseIn: software) and cannot be combined or merged.

Example 6: For LM, the Component 1 Software Developer role, after merger, would have the following attribute sets, $cau_{1SD} = \{(\text{performsJob: software})\}$. 

The general software developer role (see the partial role hierarchy in Figure 4) has the attribute set, \( \text{cau}_{SD} = \{(\text{hasDegree: [bachelors, masters]}), \ (\text{performsJob: software}), \ (\text{assignedTo: [Blue, Gold, Red]}), \ (\text{hasExpertiseIn: [UML, Code]})\} \).

### 5.5 Pruning the required candidate user attributes by assessing their significance

To assess the significance of required candidate user attributes, the significance of each attribute-value set as a whole is considered and each attribute-value pair individually in order to judge what combination of attributes would be the most likely to represent the users who belong in the role and not those who do not have the qualifications of the role. The significance of an attribute-value set \( \text{as}(i) \in \text{cau}_{\text{unique}} \) can be measured by using the number of users in role \( r \) that possess all the attributes and values in \( \text{as}(i) \) versus the number of users that are not in \( r \) but still possess the attributes and values in \( \text{as}(i) \).

The attribute-value set significance factor \( \phi_c \) as \( \text{as}(i) \in \text{cau}_r \), is computed as follows:

\[
\phi_c = \frac{|\psi^r_{\text{as}(i)}|/|\text{assigned}_\text{users}(r)|}{|\psi^r_{\text{as}(i)}|/(U - |\text{assigned}_\text{users}(r)|)},
\]

where \( \psi^r_{\text{as}(i)} = \{u_j | u_j \in (\text{assigned}_\text{users}(r)) \land \forall (a_i : v_i) \in \text{as}(i), a_i \text{ is an attribute of } u_j \} \), and \( \psi^r_{\text{as}(i)} = \{u_j | u_j \in (U - (\text{assigned}_\text{users}(r))) \land \forall a_i \in \text{as}(i) \mid (a_i : v_i) \text{ is an attribute of } u_j \} \).

**Example 7:** For the software developer role, let us say that our final merged candidate attribute set is: \( \text{cau}_{SD} = \{(\text{performsJob: software}), \ (\text{assignedTo: [Blue, Gold, Red]}), \ (\text{hasExpertiseIn: [UML, Code]})\} \). LM has 500 employees and 20 are currently assigned to the software developer role. All of those assigned to the role have attributes that match the candidate attribute-value pair set and 4/480 users who are not assigned to the software developer role have the attributes and their required values. The significance factor is thus \((20/20)/(4/480) = 12\), as shown in Table 1. If we set the threshold value of \( \phi_c \) to 100 (it is a 100 times more likely for a software developer to have the attributes than for someone other than a software developer to have those attributes), then the attribute set is relevant.

**Table 1** The significance factor

<table>
<thead>
<tr>
<th>Attribute and its values</th>
<th>( \psi^r_i )</th>
<th>( \psi^{\tilde{r}}_i )</th>
<th>( \phi_a )</th>
</tr>
</thead>
<tbody>
<tr>
<td>performsJob: software</td>
<td>20/20</td>
<td>52/480</td>
<td>9.23</td>
</tr>
<tr>
<td>hasExpertiseIn: [UML, Code]</td>
<td>20/20</td>
<td>55/480</td>
<td>8.73</td>
</tr>
<tr>
<td>assignedTo: [Blue, Gold, Red]</td>
<td>20/20</td>
<td>130/480</td>
<td>3.69</td>
</tr>
</tbody>
</table>
5.6 Checking requirements across roles

The final step is a post-processing step that evaluates the usability of the required candidate user attributes, \( cau^r_{\text{significant}} \). This step is to ensure that attribute-value pairs are semantically relevant and significant for the role.

Given a role hierarchy, \( RH \), if two roles, \( r_i \) and \( r_j \) are such that \( r_i \) is a more specialised role than \( r_j \), then

1. \( cau^r_{\text{unique},i} \neq cau^r_{\text{unique},j} \)
2. the values of attributes held in common in \( cau^r_{\text{unique},i} \) and \( cau^r_{\text{unique},j} \) must have the relationship \( \text{SubClass}(v_j) = v_i \)

where \( v_i \) is an attribute value of a required user attribute for \( r_i \) and \( v_j \) is an attribute value of a required user attribute for \( r_j \).

The above is a formal way of expressing the following. A member of a more specialised role should have a more specific range of values for an attribute. For example, if an electrical engineering degree is needed to have the role \textit{electrical engineer}, then a stricter requirement that can be imposed for the role \textit{senior electrical engineer} is that a person has a masters degree in electrical engineering.

\textbf{Example 8:}  Figure 5 shows how the concepts from this step are applied. The more specialised \textit{C1SD} role has more stringent set of required user attributes than that of the \textit{Software Developer – Project Blue} role (i.e., the C1SD Role must have expertise in a specific type of Code – java or financial). The \textit{SW Developer – Project Blue} role, likewise, has more stringent required user attributes than the general \textit{SW Developer} role (i.e., the \textit{SDBlue} role requires assignment to Project Blue). Thus hierarchically, the attribute requirements are appropriate. However, the two sibling roles, \textit{C1SD} and \textit{C2SD} require reexamination. This is because, the required user attributes for \textit{C2SD} are a subset of those for \textit{C1SD}. Unless it is deemed appropriate that anyone who is in the \textit{C1SD} role with java expertise can also gain access to the \textit{C2SD} role, additional required user attributes must be added for \textit{C2SD}. Let us say that a new attribute is defined to make the \textit{C2SD} the required user attribute set unique. The new required attribute is \textit{hasExpertiseIn:display} because \textit{component 2} includes display functionality. The members of \textit{C2SD} are then rechecked to see if it is appropriate to assign them this attribute and if so the attribute and value are added to the \textit{UAB} for those users.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5.png}
\caption{Role attribute requirement comparisons}
\end{figure}
5.7 Evaluation of access requests from a new or external users

Once required attributes are determined, they can be used to decide whether to grant or disallow access to requested objects. Access decisions are made purely on the basis of the submitted attribute-value pairs.

**Example 9:** Suppose Lara Werner from HU is sending a request that she would like access to resources of type UML that are associated with Project Blue. Her request would be presented as follows: ⟨⟨HU433⟩⟨(objectContains:UML), (project:Blue)⟩⟨lwerner, (title:software engineer), (project:Blue), (expertise:UML)⟩⟩.

There are several steps in processing a request:

**Step 1.** If the request is made in terms of object attributes, a select query is made to the OAB with the specified attributes and values to determine the objects (if any) whose object attributes match the request. Requested objects, \( r_o \), are the set of all objects that meet the request. An object meets the request by having equivalent attribute names to those in the request and associated attribute values which are equal, equivalent, or where the requested value has the subclass relationship with the object value. The roles that have permission to access the object(s) are then identified by locating the objects in \( \text{assigned} \_ \text{objects}(r) \).

**Example 10:** In this example, the object is described by attributes: \{ (objectContains:UML), (project:Blue) \}. Objects with those attributes are \( o_{526} \) and \( o_{989} \). Assuming the only operation allowed on these objects is ‘read’, we consult the permission to role assignment relationship, \( \text{PA} \) per Definition 1, to discover which roles have access to the object(s) that match the request. Both objects \( o_{526} \) and \( o_{989} \) are assigned to the \( \text{SDBlue} \) role.

**Step 2.** Next, the requester’s user attributes are examined and compared to the attribute requirements for the role(s), \( au^r \), that have permission to access the requested object(s).

If there are no other roles to test, a denial message is returned to the requester with (optionally) a reason given that the request did not include sufficient credentials.

**Example 11:** For our example, the role \( \text{SDBlue} \) has the following attribute requirements: \{performsJob:software, assignedTo:Blue, hasExpertiseIn:[Code,UML]\}. Our requester has the attributes \{title:software engineer, project:Blue, expertise:UML\}.

If the requester does not submit appropriate credentials covering all attribute requirements for access to the requested object, it is assumed that the access is simply denied.

6 Experimental evaluation

In order to compare the performance of both approaches, we conducted an experimental study. This section discusses the details of experimental study. Validation methods that we have employed to study the performance of models are also discussed.
Using classification for role-based access control management

in this section. Performance evaluation is done by using both real data sets and the
synthetic datasets. Experimental results are evaluated by considering the F-measure
and Lift of the resulting models. We conclude this section with the summary and
analysis of results.

The purpose of conducting experimental study was to answer the following
questions:

• What is the the overall performance of each of our approaches?
• What is the impact of choosing different threshold values on the performance of
our first approach?
• What is the impact of different classification algorithms on the performance of
our second approach?
• What is the impact of changing organisational parameters, such as number of
users, number of attributes, and number of roles on the performance of our
approaches?

All experiments were ran on an Intel P-IV machine with 4 GB memory and 2 GHz
dual processor CPU. In order to test the performance of classification-based method
for facilitating coalition-based access control, we are using following three well known
classification algorithms:

1. J.48 Decision tree, which is an extension of C4.0 (Quinlan, 1993),
2. Support vector machines (SVMs) (Burges, 1998), and

The code for these algorithms is adapted from the Weka machine learning open source
repository (Witten and Frank, 1999). Weka is an open source software suite developed
at University of Waikato. For SVM algorithm, we are using libSVM which is a wrapper
class of Weka (Chang and Lin, 2011).

Semantics-based approach is implemented in Java. This approach requires tuning
of threshold value for identifying necessary attributes for a role. We ran our
experiments with several threshold values. However, in this paper, we are including
results obtained from models that are built with two threshold values: 0.2 and 0.7. The
reason for not including results of additional models is that we are getting more or less
the same range of results.

Validation methods: To study the quality of produced models, we are using two
performance evaluating metrics (Powers, 2007):

1. F-measure and
2. Lift.

Brief discussion about these measures is done below.

F-Measure. F-measure is a harmonic mean of precision and recall (Powers, 2011).
Precision corresponds to the number of true positive instances retrieved by the model
out of total true instances. Equation 1 shows how the precision is computed. Total true
instances correspond to the sum of true positive and false positive instances. True
positive instances are those instances which are correctly classified as belonging to the positive class. On the other hand, false positive instances are those instances which are incorrectly classified as belonging to the positive class.

\[
\text{Precision} = \frac{Rtp}{Atp + Afp}
\]

(1)

Where,
\(Rtp\) = Retrieved number of true positive instances,
\(Atp\) = Actual number of true positive instances,
\(Afp\) = Actual number of false positive instances.

Recall, on the other hand is the proportion of instances which are actual positives to the number of instances which are correctly identified by the model as positive instances, as given in Equation 2.

\[
\text{Recall} = \frac{Rtp}{Atp + Afn}
\]

(2)

Where,
\(Rtp\) = Retrieved number of true positive instances,
\(Atp\) = Actual number of true positive instances,
\(Afn\) = Actual number of false negative instances.

**Lift.** Lift represents the performance of a particular model in identifying the targeted data instances (Witten and Frank, 1999). It is a ratio between the number of targeted data instances identified when a particular model is employed vs the number of targeted data instances identified when no model is employed. High lift is an indicator of good performance of a model. Lift is computed by first sorting all data instances into descending order of their probability score. The probability score represents degree of confidence with which an instance is assigned to a particular class. Arranged data are then evenly divided into certain number of chunks (usually 10 chunks). Lift value is then computed for each chunk. Lift value of a chunk is obtained by dividing the number of positive cases within that chunk to the total number of positive cases within the dataset. Cumulative lift curve serves as a useful tool for visualising the performance of a model and is constructed by using probability scores. It is generally used to estimate how well the model is at predicting the class than using random predictions alone on the same dataset. On the graph, the x-axis has cumulative number of data instances and y-axis has cumulative number of true positives instances. True positive instances are those instances which are correctly classified by the predictive model.

**Real datasets:** We used four real datasets. Important characteristics of the real datasets are given in Table 2.

**Synthetic datasets:** The synthetic datasets were created using the test data generator from Vaidya, Atluri and Warner (2006). The test data generator performs as follows: First, a set of roles are created. For each role, a random number of permissions up to a certain maximum are chosen to form the role. The maximum number of permissions to be associated with a role is set as a parameter of the algorithm. Next, the users are created. For each user, a random number of roles are chosen. Again, the maximum number of concurrent roles a user can have is set as a parameter of the algorithm. Finally, the user permissions are set according to the roles to which the user has been assigned. Tables 2(a–c) give the characteristics of the datasets created. Since the effect
of large number of users, permissions, and varying densities has already been studied with the real datasets, the synthetic datasets were created with a limited size to enable focused testing of the effect of noise, and of the role risk. As the test data creator algorithm is randomised, five datasets for each combination of parameters are created, and the results are averaged. Overall, we performed our experiments using 15 synthetic datasets.

Table 2  Characteristics of real data sets

<table>
<thead>
<tr>
<th>numRoles</th>
<th>numUsers</th>
<th>numAttribs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Dataset 1</td>
<td>121</td>
<td>130</td>
</tr>
<tr>
<td>Real Dataset 2</td>
<td>241</td>
<td>141</td>
</tr>
<tr>
<td>Real Dataset 3</td>
<td>593</td>
<td>358</td>
</tr>
<tr>
<td>Real Dataset 4</td>
<td>158</td>
<td>104</td>
</tr>
</tbody>
</table>

(a) Parameters for synthetic datasets when varying number of users, keeping everything else constant

<table>
<thead>
<tr>
<th>numRoles</th>
<th>numUsers</th>
<th>numAttribs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic Dataset 1</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>Synthetic Dataset 2</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td>Synthetic Dataset 3</td>
<td>200</td>
<td>1000</td>
</tr>
<tr>
<td>Synthetic Dataset 4</td>
<td>200</td>
<td>2000</td>
</tr>
<tr>
<td>Synthetic Dataset 5</td>
<td>200</td>
<td>5000</td>
</tr>
</tbody>
</table>

(b) Parameters for synthetic datasets when varying number of attributes, keeping everything else constant

<table>
<thead>
<tr>
<th>numRoles</th>
<th>numUsers</th>
<th>numAttribs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic Dataset 1</td>
<td>150</td>
<td>1500</td>
</tr>
<tr>
<td>Synthetic Dataset 2</td>
<td>150</td>
<td>1500</td>
</tr>
<tr>
<td>Synthetic Dataset 3</td>
<td>150</td>
<td>1500</td>
</tr>
<tr>
<td>Synthetic Dataset 4</td>
<td>150</td>
<td>1500</td>
</tr>
<tr>
<td>Synthetic Dataset 5</td>
<td>150</td>
<td>1500</td>
</tr>
</tbody>
</table>

(c) Parameters for synthetic datasets when varying number of roles, keeping everything else constant

<table>
<thead>
<tr>
<th>numRoles</th>
<th>numUsers</th>
<th>numAttribs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic Dataset 1</td>
<td>80</td>
<td>1500</td>
</tr>
<tr>
<td>Synthetic Dataset 2</td>
<td>150</td>
<td>1500</td>
</tr>
<tr>
<td>Synthetic Dataset 3</td>
<td>250</td>
<td>1500</td>
</tr>
<tr>
<td>Synthetic Dataset 4</td>
<td>400</td>
<td>1500</td>
</tr>
<tr>
<td>Synthetic Dataset 5</td>
<td>500</td>
<td>1500</td>
</tr>
</tbody>
</table>

Methodology for creating synthetic data sets. For creating synthetic datasets, first, a set of roles are created. For each role, a random number of attributes up to a certain
maximum are chosen to form the role. The maximum number of attributes to be
associated with a role is set as a parameter to the algorithm. Next, the users are created.
For each user, a random number of roles are chosen. Again, the maximum number
of concurrent roles a user can have is set as a parameter to the algorithm. Finally, the
user attributes are set according to the roles the user has. In some cases, the number
of roles randomly chosen is 0 – indicating that the user has no roles, and therefore, no
attributes.

Since the test data creator algorithm is randomised, we ran it five times on each
particular set of parameters to generate the datasets. Both approaches were tested
on each of the created data sets. All results reported for a specific parameter set are
averaged over the five runs.

- **Varying number of users with fixed number of attributes and roles:** In the first
  set of experiments, we kept the number of attributes and roles constant, while
  changing the number of users. It is important to note that number of attributes
  and roles are kept constant to ensure that results of each synthetic dataset within
  this set of experiment are comparable. Characteristics of data sets for set1 are
given in Table 2(a).

- **Varying number of attributes with fixed number of users and roles:** In the
  second set of experiments, we kept the number of users and roles constant while
  varying the number of attributes (and correspondingly, the number of attributes
  per role). Again, the number of users and roles are kept constant to ensure that
  results of each synthetic dataset within set2 are comparable. Characteristics of
data sets for set2 of experiment are given in Table 2(b).

- **Varying number of roles with fixed number of users and attributes:** In the third
  set of experiments, we kept the number of users and number of attributes
  constant while varying number of roles. Now, the number of users and attributes
  are kept constant to ensure that results of each synthetic dataset within set3 are
  comparable. Characteristics of datasets for set3 are given in Table 2(c).

**Results on real datasets.** Datasets are partitioned into training and testing data.
Training data have 2/3rd of the data instances and test data consist of remaining
1/3rd of the data instances. Training data are used to build predictive models whereas,
test data are used to evaluate the performance of the model. For our first approach,
two predictive models are built through approach discussed in Section 4.3 using two
different threshold values: 0.7 and 2.0. For testing the performance of classification-
based approach, we are using following three classification algorithms: SVMs, Decision
trees, and Random forests, as discussed in Section 3.

1. **Performance in terms of F-Measure:** Figure 6 shows performance of obtained
models. Each model has a separate performance curve on the graph. Performance
is shown in terms of F-measures. We observe that the classification-based method
using Random forest outperforms all other models overall with the minimum
F-measure 0.13 and maximum 0.4.

Semantics-based method has poor performance on real datasets with no F-measure
greater than 0.03. Also, we observe that performance of Semantics-based method is
actually worst for the larger datasets, whereas it slightly improves for relatively smaller
datasets. It is also interesting to note that for real datasets, change in the threshold value used for Semantics-based method has little or no impact on the F-measures. On the other hand, for classification-based method, choice of classification algorithm has significant effect on the performance. For example, SVM has worst performance. In fact, SVM performs better than semantics-based method for just one dataset, but for the remaining three datasets its performance is no better than that of Semantics-based method. However, the performance of decision trees is close to that of random forests.

Figure 6 Predictive performance in terms of F-measure for real datasets (see online version for colours)

Performance in terms of Lift: Figures 7(b–d) show performance of our approaches in terms of lift for real dataset1, real dataset2, and real dataset3, respectively. The area between the baseline curve and the lift curve for any model shows how much a predictive performance can be improved by using that model as opposed to when no model is used. We have separate graph for each real dataset. The lift curves on each graph show that classification-based method using Random forest algorithm outperforms all other models in terms of lift. Lift curve of a decision tree shows that it is the second best model. The performance of SVM is actually worst among all three classification algorithms that we used.

Lift curve for semantics-based models shows that these models perform poorly. Their lift curves are also showing that the model that is created with lower threshold value performs slightly better than the one built with higher threshold value.

Results on synthetic datasets

Performance in terms of F-Measure: Figures 7(e–g) shows performance of models on synthetic datasets in terms of F-measure.
On synthetic datasets, decision tree has overall best performance in terms of F-Measure. Variation in number of users and number of attributes has little or no effect on the performance on decision tree. However, the change in number of roles affects the performance. The increase in number of roles while keeping other parameters constant improves the performance of decision tree.

**Figure 7** Performance of models on both real and synthetic datasets (see online version for colours)

(a) Predictive performance in terms of lift for Real Dataset1
(b) Predictive performance in terms of lift for Real Dataset2
(c) Predictive performance in terms of lift for Real Dataset3
(d) Predictive performance in terms of lift for Real Dataset4
(e) F-measures, synthetic Data set1
(f) F-measures, synthetic Data set2
(g) F-measures, synthetic Data set3
(h) Lift curves and baseline for synthetic data 1
(i) Lift curves and baseline for synthetic data 2
(j) Lift curves and baseline for synthetic data 3

In case of synthetic datasets, Random forest also performs well, and is the second best model in terms of F-measure. Its performance improves with the increase in number of users, while keeping other parameters constant. Its performance also improves with increase in number of attributes while keeping other parameters constant.
Finally, increasing the number of roles while keeping everything else constant has a positive impact on the performance of decision tree.

F-measures for semantics-based approach are lowest which means it performs poorly on synthetic data sets. Performance of the model built with threshold 0.7 declines significantly when the number of attributes are increased while other parameters are kept constant. Performance of the model built with threshold 2.0 shows no change in response to change in number of users. We also observe that the performance of model built with threshold 2.0 is better than that of SVM-based model when number of attributes and number of roles are changed, as seen in 7(f) and 7(g).

2 Performance in terms of lift: Figures 7(h–j) show performance of models in terms of lift. Decision tree outperforms all other models in terms of lift. Random forest also performs well and is the second best model. SVM has the worst performance among all the classification algorithms used. Varying parameters has little or no impact on the performance of classification-based models.

Lift curves for semantics-based approach models show that these models do not perform well. In fact, their lift curve is same as baseline curve which means that results based on these models are no better than results obtained from random predictions.

Summary and analysis of results. Based on the experimental results on both real datasets and synthetic datasets, we demonstrated performance of two methods discussed in this paper. The main goal of both approaches is to identify the set of necessary attributes of a user for a role. Results were evaluated in terms of both F-measure and lift. For the semantic-based method, we used two different threshold values to create models based on it. For the classification-based method, we used three well-known classification algorithms: Support Vector Machines (SVMs), decision tree, and Random forests for building classification models.

Experimental results indicate that the performance of classification-based method is significantly better and consistent as compared to the performance of semantics based approach. Specifically, Random forest and Decision tree models are two best performing models to address the problem of coalition-based access control. It is interesting to note that decision tree performs better than random forest when synthetic datasets are used, whereas random forest performs better when real data sets are used. Overall the performance of Random forest algorithm is quite robust in presence of noise, and decision tree performs better when the data for building classification model is clean.

Determining significant attributes through semantic-based approach is quite complex and lengthy process. It involves steps like discovering user attributes that are semantically related to object attributes, forming referential and synonymous linking, and merging candidate attribute-value pair sets. Moreover, the tuning of threshold value is not easy. If the threshold is set to some higher value then very few attributes would be included in the set of necessary attributes for a role. When this identified set of attributes is used to classify new user for a role, false positive errors in the system would increase significantly because many necessary attributes would not be added to a set of significant attributes. Whereas, if the threshold value is set to lower value then many unnecessary attributes would be included in the set of required attributes even when they are not actually significant for a role membership. When this identified set of attributes (which may also include unnecessary attributes) is used to classify new user
for a role, false negative errors in the system would increase because many unnecessary attributes would be required from a user to become a member of a role.

Classification-based method, on the other hand, is relatively straightforward and easier to implement. Simply, $UAB$ and $UA$ matrices are used to generate the classification models. No linking of attributes, merging pairs, or tuning of threshold value are required. More importantly, semantic-based approach is rigid in a sense that it returns a fixed set of required attributes for a role. User should have all the required attributes to become a member of a role. On the other hand, Classification model allows flexibility and is based on alternative sets of required attributes. Moreover, the significance value of each attribute is used to predict the membership status of a new user. If few attributes of lesser significance are missing from new user’s attributes set and most of the attributes with higher significance value are present in user’s attributes set, then the role membership is given through classification method. We believe that this is a realistic way of assessing user credentials for assigning a particular role.

7 Related work

Access control research in the area of dynamic coalitions was first introduced by Philips, Ting and Demurjian (2002); Philips et al. (2002) by providing motivating scenarios in defense and disaster recovery settings. Cohen et al. (2002) proposed a model that captures the entities involved in coalition resource sharing and identifies the interrelationships among them. Two sets of researchers (Bharadwaj and Baras, 2003; Khurana et al., 2003) addressed the issue of automating policy negotiation and Yu, Winslett and Seamos (2003) addressed the issue of building trust. In our prior work (Atluri and Warner, 2004; Warner, Atluri and Mukkamala, 2005a,b), we have proposed a coalition-based access control (CBAC) model that facilitates automatic translation of coalition level policies to the implementation level policies, and vice versa. This primarily employs credential attributes in accomplishing the translation, but does not exploit semantics. Access control via attributes simplifies administration because specific users do not have to be given identities. Instead access rights are determined purely on the basis of attributes and these attributes can apply to many different users, who would present their attributes through the use of credentials. Bacon, Moody and Yao (2002) proposed a similar RBAC architecture called OASIS where they mapped users via credentials issues by a third party. However, they did not address how the credential requirements are determined, which is the main focus of this paper. Wang, Wijesekera and Jajodia (2004) presented ABAC but they also did not give a mechanism for determining which attributes should be used. Li, Mitchell and Winsborough (2002) proposed a coalition-based access control (CBAC) model that facilitates automatic translation of coalition level policies to the implementation level policies, and vice versa. This primarily employs credential attributes in accomplishing the translation, but does not exploit semantics. Access control via attributes simplifies administration because specific users do not have to be given identities. Instead access rights are determined purely on the basis of attributes and these attributes can apply to many different users, who would present their attributes through the use of credentials. Bacon, Moody and Yao (2002) proposed a similar RBAC architecture called OASIS where they mapped users via credentials issues by a third party. However, they did not address how the credential requirements are determined, which is the main focus of this paper. Wang, Wijesekera and Jajodia (2004) presented ABAC but they also did not give a mechanism for determining which attributes should be used. Li, Mitchell and Winsborough (2002) and Li, Winsborough and Mitchell (2003) address the problem of present a role-based trust management (RT) framework that addresses the issue of discovering credentials needed to map to a role. However, their algorithm does not address how to select attribute requirements based on existing RBAC policies.

Krishnan et al. (2011) presented an approach group-centric secure information sharing (g-SIS), to facilitate secure information sharing and collaboration. In their work, they used the notion of group which refer to users and objects from multiple organisations. Basically, group represents a set of users from multiple organisation who shares the same set of objects available to their respective group. g-SIS-based approach
is based on first-order linear temporal logic (FOTL). g-SIS works as a complimentary model for infrastructure of security across organisational boundaries when coalitions are formed. User who becomes a member of a particular group can access all objects available to the remaining users of the group. However, their model does not take into account the various authorisation semantics. Chun, Warner and Keromytis (2013) presented a framework of privacy-preserving data sharing and integration for mashup service. Mashup content is created by extracting and combining data from multiple data sources.

When using the CBAC framework, one can think of two extreme cases: If a resource provider (P) requires all the credentials to be possessed by the resource requester (R) to access its resources, P’s policies are ensured with the highest level of security. However, from the perspective of R it is highly restrictive (or less permissive). At the other extreme, if P does not require any credentials at all for R, then there are no guarantees on the security. However, R has the highest permissiveness to acquire the resource. Many variations in between these extremes may exist. Wang, Wijesekera and Jajodia (2004) and Warner, Atluri and Mukkamala (2005a) presented ABAC but did not give a mechanism for determining which attributes should be used. Atluri and Warner (2004) computes the union of all credential attributes of the users playing a role as the required set to facilitate coalition we assumed that a user would send all possessed credentials with every request. Also, it was assumed that the required credentials would match a union of all credentials that anyone who had access to the object possessed. There are two obvious drawbacks of the approach. First, it required far more credentials than a typical user is likely to have and second it required processing of many irrelevant credentials. under emergencies among entities of heterogeneous nature. Chen and He (2011) also presented a framework based on XACML to facilitate secure sharing of resources. Again, automated discovery of required credentials of role remains unaddressed. Warner, Atluri and Mukkamala (2005b) proposes a graph pruning method to reduce the required credential attribute set. The set of required attributes is reduced using frequency counts and sets of attributes held in common by users assigned to roles. The attribute significance introduced in this paper is a similar concept, but computed differently. Soluade (2013) proposed an approach based on classification to automate the process of identifying the most critical features of software.

Further, Alsulaiman, Miège and El-Saddik (2007) presented a threshold based Collaborative access control (T-CAC). In (T-CAC), there is a threshold associated to each permission. Role of a user is used to compute the threshold value. Lu, Zhang and Sun (2009) gives task-activity based access control (TABAC) model for collaborative environment where authorization rules are linked by the relationship between tasks and activities. This model cannot directly work in the RBAC environment. Toahchoodee et al. (2009) discusses problems associated with insecure sharing of information with unauthorized entities. Brucker and Hutter (2010) presented an approach to support formation of coalition between organizations where Liaison Officer creates new roles in an ad-hoc manner and assign users from home organization to those roles. Limitation of their approach is that it is based on an assumption that the Liaison Officer understands the role configuration of both home organization and host organization very well. In many real world examples, it is seen that this assumption is too optimistic. Usually, when coalition is formed in quick and ad hoc manner, the internal role configurations are not shared with any entity from external organization.
Carminati, Ferrari and Guglielmi (2011) discussed their concept of granting temporary access control when the access to a request is denied by the regular access control system. Temporary access control policies only become active during emergencies. In their work, contextual access control integrated with the regular system to inform the regular system about the emergencies and as soon as the emergency is detected, the temporary access control policies becomes active. Temporary access control allows access. Therefore, it not the user who decides whether to override regular access control policies or not, but it is the access control system that decides whether overriding of regular policies should be allowed or not. The system given in this paper considers that emergency situations and corresponding actions for handling that emergency are pre-defined in the system. As a matter of fact, it is not always possible to define the emergency situations before they occur, therefore, knowing the actions needed to handle those critical and unseen emergencies is complex task and cannot be described accurately in advance. In Jonker and Petkovic (2012), a language to capture complex event patterns and critical emergency scenarios is proposed. In this paper, the problem of dynamically configuring the access control at the time of emergencies is addressed, however work presented in this paper does not consider the aspect of secure sharing of information.

Another line of work which is closely related to this work is in the area of Break-glass approaches. Mainly, break-glass approaches were introduced to handle shortcomings of traditional access control systems. Traditional access control system runs under the assumption that if some object is requested by a subject and there is no policy defined in the system for such request then simply the access is denied. However, in real world, there are numerous situations where this assumption is overly rigid and therefore creates a bottle neck in the system. To add flexibility property in traditional access control systems, break-glass approaches were proposed. These approaches are based on the idea of overriding the implemented access control policy in case of emergency. Under break-glass model, when the overriding is allowed, a user who exercise these privileges has to comply with the obligations associated with the access granted through implemented break-glass model. Brucker and Petritsch (2009) presented an approach to integrate break-glass strategies into the deployed access control system. The difference between the regular policy and the required new policy is used to make decisions under this model. However, it is not defined how the distances are computed in real world access control environment. Recently, Marinovic et al. (2011) presented RUMPOLE model to facilitate decision making in the situations where break-glass policy has to be activated. Rumpole is based on belnap four-valued logic that to enable controlled and informed decision making. In addition to above mentioned approaches, some other approaches such as Ferreira et al. (2006, 2009) were also proposed to facilitate break-glass policies where break-glass models are added as an additional layer to the existing access control system and this layer is activated whenever an exceptional situation occurs.

Break-glass models are not panacea for access control systems. On one hand, traditional models are too rigid in preventing access to requested objects, whereas break-glass models are too flexible in granting access to subjects. Therefore, a fine mix of security and flexibility was needed. To address this requirement, risk-based access control systems were introduced. The idea behind risk-based access control is that the access is granted to a requesting user if the underlying risk associated with the request is relatively low as compared to the benefit of allowing such access. Several efforts have
been made to design frameworks and develop approaches to support risk based access control. Some of the notable work includes (but is not limited to) Kandala, Sandhu and Bhamidipati (2011), Zhang, Brodsky and Jajodia (2006), Ni, Bertino and Lobo (2010), Dimmock et al. (2004), Molloy et al. (2011), Nisanke and Khayat (2004), Chen and Crampton (2011), Sarma et al. (2012). However, most of the works in this area suffer limitations in terms of effective and reliable means of quantifying risk, computing risk values at run-time according to the situation, combining multiple policies from different organisations to compute risk. Badar et al. (2013) used classification methods for quantifying the risk associated with the requested permission. In their work, historic data of permissions are used to create the models. Our work is similar to their work, but they have used permission-based data for building the models. Whereas, we are using user attribute base for building classification model. Moreover, in their work, problem of coalition-based access control is not taken into an account.

8 Conclusions

In RBAC security environment, authorisations are granted to a user via role. Traditionally, to handle any authorisation request submitted by a new user, system administrator has to decide whether a request should be honoured or not and how it should be honoured. For that purpose, thorough analysis of existing assignments is done in order to determine why a specific role is assigned to certain users. Manual analysis of large datasets is quite complex. By simply looking at the user-role assignment (UA) matrix and permission-role (PA) matrix, a system administrator cannot decide whether an authorisation should be granted to a new user or not. Therefore, automated process for analysing the access request is needed.

In this paper, we presented an approach to facilitate automatic enforcement of access control policies when a new user is added to an existing access control system. Our approach is based on using classification method for building model to classify user credentials for a role. We have assumed that RBAC is already deployed in the organisation. We have also assumed that the user credentials are stored in form of a binary matrix locally within the organisation. We exploit user attributes data and user-role assignment data for our approach. We compared our approach to another work which was proposed earlier for addressing the similar problem. Experiments were conducted to compare the performance of both approaches. Both real and synthetic datasets were used to evaluate the performance. Results of experimental study showed that our approach based on classification method performs very well on both synthetic and real datasets. Moreover, from the perspective of implementation, our proposed approach is much easier to implement.

There are several possible future directions of this work such as: how well these approaches can work in collaborative environments where requesting users may belong to an inter domain organisation. For now, we plan to extend these approaches to address the problem of misconfiguration in access control.

References


