Extended Role-Based Access Control with Context-Based Role Filtering

Gang Liu[1], Runnan Zhang[1], Bo Wan[1*], Shaomin Ji[1] and Yumin Tian[1]
[1]School of Computer Science and Technology, XIDIAN University
Xi'an, 710071 - China
[e-mail: gliu@xidian.edu.cn]
*Corresponding author: Bo Wan

Received July 7, 2019; revised October 27, 2019; accepted January 6, 2020; published March 31, 2020

Abstract

Activating appropriate roles for a session in the role-based access control (RBAC) model has become challenging because of the so-called role explosion. In this paper, factors and issues related to user-driven role management are analysed, and a session role activation (SRA) problem based on reasonable assumptions is proposed to describe the problem of such role management. To solve the SRA problem, we propose an extended RBAC model with context-based role filtering. When a session is created, context conditions are used to filter roles that do not need to be activated for the session. This significantly reduces the candidate roles that need to be reviewed by the user, and aids the user in rapidly activating the appropriate roles. Simulations are carried out, and the results show that the extended RBAC model is effective in filtering the roles that are unnecessary for a session by using predefined context conditions. The extended RBAC model is also implemented in the Apache Shiro framework, and the modifications to Shiro are described in detail.

Keywords: Role-based access control, role activation, context condition, candidate role, role explosion

A preliminary version of this paper appeared in the 3rd International Conference on Information and Network Technologies, 24-26th of May 2018, Osaka, Japan. This version includes a concrete analysis and definition of the problem of activating roles, along with an implementation of our model. This work was partly supported by the the National Science Basic Research Plan in Shaanxi Province of China (Program No. 2018JM6034), the Shaanxi Key R&D Program (Grant No. 2019ZDLGY13-01) and the Science and Technology Projects of Xi'an, China (Grant number: 201809170CX11JC12).

http://doi.org/10.3837/tiis.2020.03.019 **ISSN : 1976-7277**

1. Introduction

Role-based access control (RBAC) has become the most popular access control model, and is simple to administer and reviewer because it reflects an organisational structure [1]. In the RBAC model, users are assigned roles, and roles are assigned permissions; thus, users are granted permissions through role activation. This simple RBAC user authorisation model is sufficient for well-organised systems (for instance, a company), where there are generally few roles.

However, because of the development of networks, the so-called role explosion problem (an excess of roles assigned to a single user) has become serious, leading to increasing difficulties in managing and using roles [2]. Currently, researchers focus on facilitating role management for administrators by extending the RBAC model, for example, administrative role-based access control (ARBAC) [10] and the context-aware access control model [9],[12],[12]. However, in the context of role explosion, we believe that the problem of user-driven role management, which means that the user manages session-activated roles, is a crucial one. The significant difference between a normal user and a professional security administrator is the level of knowledge of system security. Security administrators have a comprehensive knowledge of security requirements, policies, and the authorisation framework. In other words, an administrator has a comprehensive understanding of his role and experience in its management. Obviously, users do not have the same security knowledge as security administrators. Thus, user-driven role management is a difficult and urgent problem to be solved. Moreover, the user understanding of security issues directly affects the gravity of role management issues. The better the user understands security, the smaller the management issue, and the reverse is also true.

Zhang et al., “defined and addressed the user authorisation query (UAQ) problem [3], and it has been widely studied [4-8]. The UAQ problem is defined as follows: in the web environment, where the number of roles in a system is large, users do not know which roles they are assigned, and they only request a set of permissions without activating roles. According to the set of permissions requested by a user, the system determines the role set to activate meeting certain conditions [3]. The UAQ problem assumes that the user knows the permissions needed for the session. However, in real environments, knowledge of security can be very different among users. The assumption of the UAQ problem may not be realistic, and it places higher requirements on user security knowledge. In fact, users often do not fully understand their assigned permissions or roles. Therefore, user-driven role management cannot always be translated into a UAQ problem.

Thus, the fundamental assumptions of the UAQ problem are not sensible. This paper proposes the following assumptions on user knowledge of security:

(1) Users have an approximate understanding of the session goals and the tasks to be performed. Although they cannot formally express the goal of the session, they can assign reasonable permissions to it when they know the meaning of the permissions.

(2) Users have a certain degree of understanding of the role. Although a user cannot accurately remember each role and permission, they can correctly use a role and a permission after obtaining information, such as its name and description.

Based on the above assumptions, the problem of user-driven role management can be summarised as the session role activation (SRA) problem: in a complex system, users know the tasks for a session, but they cannot describe the required permissions accurately, which leads to a difficulty in activating roles for the session. Obviously, the SRA is a smaller problem for user

security knowledge than UAQ. The SRA problem is a new, reasonable, and practical problem of user-driven role management.

To solve the SRA problem, this paper proposes an extended role-based access control model (referred to as E-RBAC in the following) which uses context information to filter roles and obtain candidate roles strongly related to the session. The number of candidate roles is significantly reduced compared to the user role set, meaning that the extended RBAC model can significantly reduce user workload on role management.

This paper provides the following three contributions:

(1) The relationship between user security knowledge and the problem of role management is analysed. A more reasonable assumption on the user's mastery of security knowledge is proposed, and the SRA problem is defined according to this assumption.

(2) We propose an E-RBAC model with a context-based role filtering function to solve the SRA problem. It uses context conditions to select a session that is more strongly related to user roles, significantly reducing the number of candidate roles. The E-RBAC model helps users who lack security knowledge to activate appropriate roles for a session. Furthermore, dynamic management of the session-role relationship is realised, complying with the principle of least privilege.

(3) We implement the E-RBAC model in a Shiro framework [13].

2. Related Work

2.1 Context

Context is an elusive concept that has different meanings to different people and communities, and even researchers (such as Dey et al., "[14] and Ryan et al., "[16]) have different definitions of context. The common point is that context is any information that can be used to characterise the interactions between users and applications. Additionally, it is believed that the three important aspects of context are where you are, who you are with, and what resources are nearby.

Context has been applied to different areas, such as the Internet of Things [18], vehicles [19], mobile computing [20], and healthcare social networks [21]. Furthermore, using context to improve the security of different applications has become an important task for researchers [22,23,25-32].

Many researchers have introduced context information into RBAC. Because roles cannot be organised in static hierarchies, Kayes et al., "introduced both formal and ontology-based approaches to model dynamic contextual roles and to specify context-aware access control policies by activating such dynamic roles at runtime [25]. Abdella et al., "proposed a context-aware RBAC model for the Android permission system which can provide dynamic permission granting and revoking while keeping the number of polices low. In the model, Android applications assign roles corresponding to a set of permissions, which in turn are associated with contexts. Permissions for the corresponding role are granted and revoked according to the associated contexts [10]. Sadat Emami et al., "proposed a context-aware dynamic RBAC model for pervasive computing environments where contexts are divided into long-term and short-term ones. At the beginning of a session, the model dynamically assigns roles to a user considering only long-term contexts. However, during the session, an active permission set for the assigned roles is determined according to the short-term context conditions [26]. Rathod et al., "introduced a semantically rich context-sensitive access control system for OpenStack by incorporating the current context attributes of a user (such as location and time). They integrated such context information in their own access control system to express and enforce the contextual-situation policies in OpenStack [28]. Zhong et al., "presented an application layer multicast system for context-aware dynamic role assignment, which adjusts

user permissions in real time according to context information. Roles and permissions are assigned dynamically, according to the context information collected on the user and the resources, to avoid real-time patching of user permissions [29]. Oluwatimi et al., "extended the context-aware system to secure enterprise content (CASSEC) by incorporating a proximity-like structure. Using the information obtained from analysing the context, the system can make inferential decisions [30].

All these studies focus on managing roles and permissions according to context. However, their constraints on roles and permissions are simple and rigid. In fact, they assume that the goals or tasks of the constrained roles/permissions can be formally described through the context. Thus, the general context constraint method clearly does not solve the SRA problem.

2.2. RBAC Authorisation Process

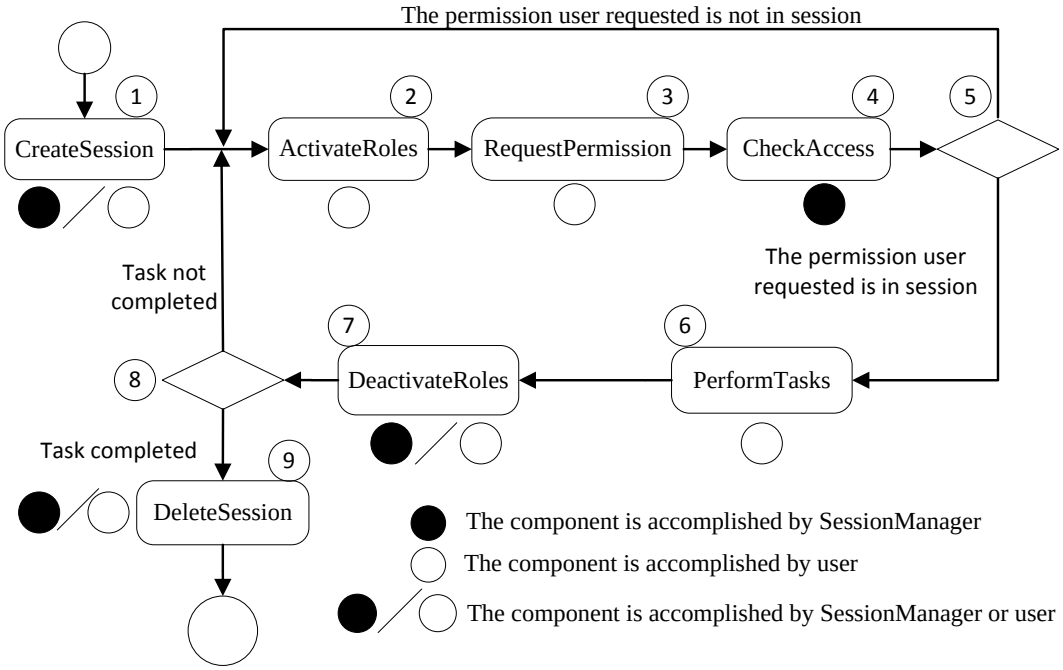

Fig. 1. Authorisation process in RBAC

Fig. 1 depicts the authorisation process in the NIST RBAC [24], and the operations are detailed as follows:

➢ *CreateSession*. Each session is defined as the mapping of one user to many possible roles; that is, after the user establishes a session, they maintain a subset of the roles assigned to them for the session.

➢ *ActivateRoles*. The *ActivateRoles* operation is a key feature of RBAC that distinguishes it from other access control models. The user selects the appropriate role to activate, and obtains the permissions assigned to the role. *ActivateRoles* ensures that only part of the role is activated, ensuring the least privilege principle and increasing system security. Because in well-organised systems (for example in a company) a user is typically assigned a small number of roles, they know which roles should be activated before requesting permissions.

➢ *RequestPermission* and *CheckAccess*. Similar to other access control models, *RequestPermission* and *CheckAccess* are required in RBAC. After the user activates the roles, they request permissions to perform a task, and the system determines whether the permissions correspond to the activated roles.

➢ *PerformTasks* and *DeactivateRoles*. If the permissions requested by the user are granted in the current session, the user starts performing their task. If the permissions have not been granted, the user returns to *ActivateRoles*. Because the permissions required for a session change with the execution of the task, during the session lifetime the user continuously returns to *ActivateRoles* or *DeactivateRoles* to maintain the session's permissions, in accordance with the least privilege principle.

➢ *DeleteSession.* This releases the resources that were used in the session, including the roles that the user did not deactivate.

Role explosion is defined as the problem of users being assigned too many roles by the system. In the RBAC model, users have no methods of effectively managing roles for a session, which makes the problem of user-driven role management evolve into the SRA problem.

3. E-RBAC MODEL

This section first defines context constraints, and then describes the proposed E-RBAC model, demonstrating how it filters roles using these constraints.

3.1. Context Constraints

According to[15], a context constraint is defined as a means of considering the context information in access control decisions. A context constraint is categorised according to different properties, such as being dynamic/static, coercive/non-coercive, or being an assignment or authorisation constraint.

In particular, an authorisation constraint is a constraint on role activation. When a user needs to activate a role, the system determines whether the authorisation constraint is met. This constraint needs to be met for the role to be activated. Obviously, the context considered by the authorisation constraint needs to be detected during the session lifetime. The context constraints used in this paper can be classified as authorisation constraints.

Before applying them to the extended RBAC model, a formal description of the context constraints used in role activation is given. A context constraint is defined according to terms including context attribute, context function, and context condition, as follows:

Definition 1. A **context attribute** *ATTR* represents a certain property of the context; for example, *TIME*, *LOCATION*, and *NAME* are context attributes. The actual value of a context attribute may change dynamically (such as *TIME*), or it may vary for different instances of the same abstract entity (such as *LOCATION*, *BIRTHDAY*, or *NATIONALITY*).

Definition 2. A **context function** is a mechanism to obtain the current value of a context attribute. We use *ATTR(p)* to represent the context function of context attribute *ATTR*, being *p* a set of input parameters, which can be empty. For example, *DATE()* could be defined to return the current value of context attribute *DATE* and *BIRTHDAY(id)* to return the birthday of a person having identifier *id*.

Definition 3. A **context condition** *COND* is a binary predicate that compares the current value of a context attribute with a predefined constant (such as *DATE()*=“2016.10.01”) or with the value of a different context attribute (such as *SCORE()>AVERAGESCORE()*). Thus, a context condition can be represented in the form (*ATTR(p)* $\perp$ *x*) or (*ATTRi(p)* $\perp$ *ATTRj(q)*), where *x* is a value and $\perp$ is a comparison operator in the set {≤,<,=,>,≥} used in this work. The range of a context condition is {true, false}.

Definition 4. A **context constraint** *CONS* contains a combination of one or more *COND*s, that is:

$$CONS = COND_1 \wedge COND_2 \wedge ... \wedge COND_n.$$

It returns true when each context condition $COND_i$ $(i=1..n)$ is true; otherwise, it returns false.

3.2. E-RBAC Model

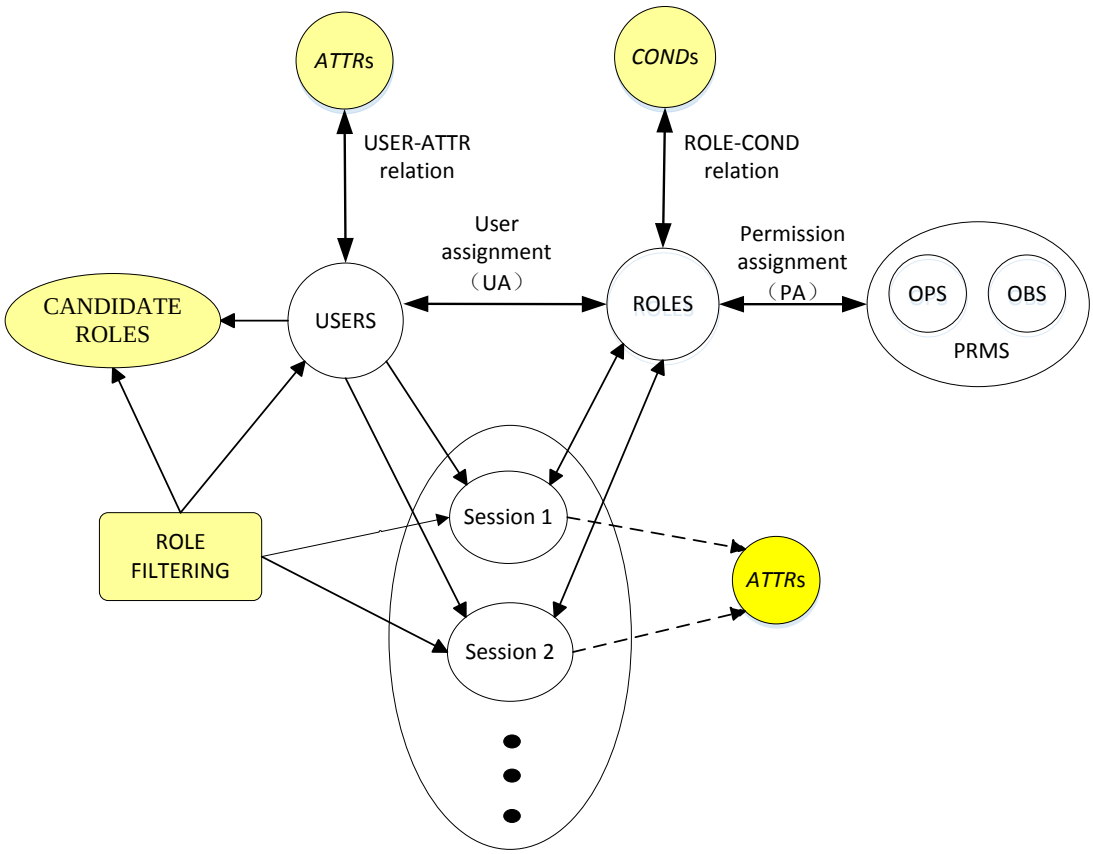

Fig. 2. E-RBAC model

The elements of the E-RBAC model and their relations are described in **Fig. 2**. The model includes the basic RBAC model and several new elements and relations, detailed in the following:

(1) CANDIDATE ROLES stores the roles selected by the ROLE FILTERING module. These roles can be viewed and activated by the user for a session.

(2) ROLE FILTERING is a module filtering roles, using context information to select those that may be appropriate for the user to activate for the session. The selected roles are inserted into CANDIDATE ROLES.

(3) *ATTR*s are context attributes that are assigned to the user according to the roles assigned to the user and that can be used to monitor the user and session context.

"USER-ATTR relation" is a many-to-many relationship, that is, a user can be assigned one or more *ATTR*s and an *ATTR* can be assigned to one or more users.

(4) *COND*s is used to represent the conditions for activating a role. In general, *COND*s and ROLEs exhibit a many-to-many assignment relationship. The *COND*s assigned to a role constitute a *CONS*. A role can only be activated when the user context meets all of the *COND*s assigned to the role, that is, the result of evaluating *CONS* must be true.

When creating roles, the security administrator establishes ROLE-COND relations according to security requirements. The USER-ATTR relations are established automatically when assigning a role to the user.

*ATTR*s are used to monitor the context information of user and session, and *ATTR*(·) obtains the value of *ATTR* of a user, session or environment. Both *COND* and *CONS* are truth-expressions associated with *ATTR*. Each *CONS* is mapped to its role, and ROLE FILTERING evaluates whether the role is added to CANDIDATE ROLES according to the value of the related *CONS*.

According to the above description, the relationship between constraints, users, sessions, and roles in E-RBAC is formalised as follows:

$$ATTR \rightarrow \text{USER}$$
$$ATTR \rightarrow \text{Session}$$
$$CONS \rightarrow \wedge COND$$
$$COND = ATTR(\bullet) \perp constant$$
$$ATTR(\bullet) = ATTR$$
$$CONS \rightarrow \text{ROLE}$$

3.3. Authorisation in E-RBAC

In E-RBAC, ROLE FILTERING and CANDIDATE ROLES are related to the authorisation process. **Fig. 3** illustrates the functions of the two components and demonstrates how the E-RBAC filters roles to solve the SRA problem. Compared to the RBAC authorisation process, four steps are added and *ActivateRoles* is modified. The E-RBAC authorisation process is detailed as follows.

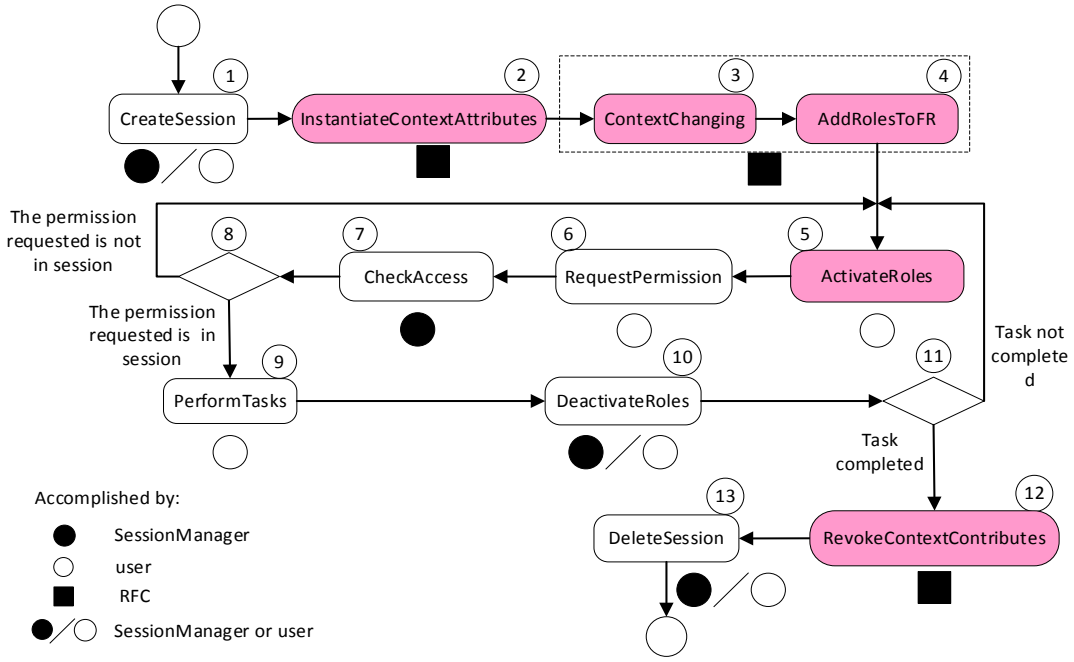

Fig. 3. Authorisation process in E-RBAC

➢ *InstantiateContextAttributes*. When a user creates a session, ROLE FILTERING instantiates the *ATTR*s assigned to the user, where instantiation means that all *ATTR*s assigned to the user are monitored in real time.

➢ *ContextChanging* and *AddRolesToFR*. *ContextChanging* is a not-real module that needs to be executed by *SessionManager*, the user, or ROLE FILTERING. In general, the user performs the task while changes occur in the context, particularly in the interconnected and collaborative environment. When ROLE FILTERING detects a change in the *ATTR*s, it evaluates the *CONS* assigned to the role according to the context, and if the *CONS* is met and it is user- or environment-dependent, the role is added to CANDIDATE ROLES. Although the modules are depicted at the beginning, this process is repeated during the session.

➢ *ActivateRoles*. One of the main differences between E-RBAC and RBAC in terms of role activation is that in the E-RBAC model the user or system only needs to find roles in CANDIDATE ROLES, and the number of roles here is significantly lower than that assigned to the user by the system. This is because the number of roles satisfying *CONS* is very small in the current environment. Thus, the user or system workload for activating the role is substantially reduced and the SRA problem is solved.

➢ *RevokeContextContributes*. At the end of the session, not only do the remaining activated roles need to be deactivated, but also the instantiated *ATTR*s need to be revoked; in other words, ROLE FILTERING no longer monitors user context in real time.

The authorisation process above described is for E-RBAC: it uses *CONS* to filter the roles, either adding the roles satisfying the *CONS* to CANDIDATE ROLES or activating them automatically. The difference between E-RBAC and RBAC is that CANDIDATE ROLES is added, which makes E-RBAC add four role-filtering-relevant steps to the RBAC authorisation process. In this way, the time required for a user to query or activate a role is highly reduced, thereby solving the SRA problem.

3.4. Simulation

To illustrate the effectiveness of E-RBAC, we carried out a simulation. Because there is currently no other study in the literature on the SRA problem, we cannot compare our results with similar works; thus, the results of the simulation are compared with those of native RBAC to illustrate the validity of E-RBAC. An evaluation using policies (including role data) from real organisations would be ideal, but we are not aware of any suitable and publicly available policy of this kind. Therefore, we evaluated our model using synthetic policies.

The number of roles assigned to the user is selected from a uniform distribution which range is 1 and the total number of roles in the simulation. For the simulation, we define generic attributes *attr1*, *attr2*, etc. The range of user attributes is [0,9]. A context condition is a binary predicate in the form of $attr \in [min,max)$, in which $min \in [-10,9)$ and $max \in (min,19]$. The scope of a *COND* exceeds the range of user attributes, so that our simulation can express terms such as $attr<3$ by using $attr \in [-5,4)$, for example. A *CONS* contains a combination of one or more *COND*s. The user attributes *min* and *max* are integers, and their values are selected from a uniform distribution in their respective ranges. In the simulation, the attribute used by the role *COND* matches the user attribute.

To explain the simulation method clearly, we provide the example shown in **Tables 1** and **2**. The example only includes three users (*U1*, *U2* and *U3*) and three roles (*R1*, *R2* and *R3*). Each role has two *COND*s to form its *CONS*. Each user has two corresponding *ATTR*s. *U1* is assigned *R2*, and the evaluation of its *CONS* is as follows:

$CONS_{R2}(U1)$ = (-1≤*ATTR1*(*U1*)<5)^(-5≤*ATTR2*(*U1*)<6) = (-1≤4<5) ^(-5≤5<6) = *True*

Therefore, *R2* is added to the CANDIDATE ROLES of *U1*. *U3* is assigned *R3*, and the evaluation of its *CONS* is:

$CONS_{R3}(U3)$ = (5≤*ATTR1*(*U3*)<15)^(-3≤*ATTR2*(*U3*)<12) = (5≤2<15)^(-3≤0<12) = *False*

R3 cannot be added to the CANDIDATE ROLES of *U3*. Similarly, the data of CANDIDATE ROLES in **Table 1** can be calculated.

Table 1. Information of Users

User id	Assigned roles	CANDIDATE ROLES	*ATTR1*	*ATTR2*
U1	*R2*	*R2*	4	5
U2	*R1, R3*	∅	4	3
U3	*R1, R2, R3*	*R1, R2*	2	0

Table 2. Information of Roles

Role id	*COND1*	*COND2*
R1	2≤*ATTR1*(•)<3	0≤*ATTR2*(•)<18
R2	-1≤*ATTR1*(•)<5	-5≤*ATTR2*(•)<6
R3	5≤*ATTR1*(•)<15	-3≤*ATTR2*(•)<12

As seen in **Fig. 4**, the statistics change as the number of users increases. In our simulation, there are 5,000 users, 500 roles, and 6 *COND*s. After counting the data of 2,000 users, the statistics no longer exhibit substantial changes. For this reason, we used 2,000 users in other simulations. **Table 3** displays the results for different numbers of *ATTR*s and roles.

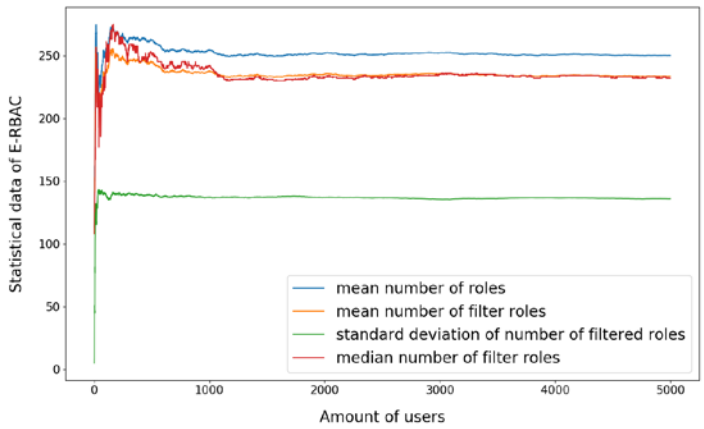

Fig. 4. Simulation for 5,000 users, 500 roles, and 6 *COND*s

In **Fig. 4**, because a uniform distribution was used to create the *ATTR*s and *COND*s, the mean and median of filtered roles are nearly equal after counting the data of 2,000 users. Moreover, at this user count, the difference between the mean number of roles assigned to the user and of those filtered is stable. This demonstrates that the E-RBAC model can stably filter out certain roles and address the SRA problem.

Table 3. Assignment of ATTRs

Roles	CONDs	Roles	Filtered roles			μ_{fr}/μ_r
		μ_r	μ_{fr}	σ	Median	
100	2	51.620	33.106	26.287	34.0	64.1%
	4	48.546	41.153	25.644	40.0	84.8%
	6	50.691	47.712	27.192	48.0	94.3%
200	2	101.414	61.275	53.449	62.0	60.4%
	4	101.895	87.866	52.178	89.0	86.2%
	6	101.812	94.960	54.123	96.0	93.3%
500	2	252.083	153.058	132.190	153.0	62.7%
	4	247.993	214.252	130.055	218.0	86.4%
	6	250.102	233.515	135.849	232.0	93.4%

Obviously, the number of *COND*s has a significant effect on the number of filtered roles. As seen from **Table 3**, columns 2 and 4, the mean and median number of filtered roles increase significantly when the number of *COND*s increases. Additionally, the mean and median number of roles and filtered roles increase as the number of roles increases. For groups with the same number of roles and different numbers of *COND*s, the standard deviation is essentially the same.

The three following results can be obtained from **Table 3**:

(1) Additional roles are filtered by using a higher number of *COND*s; therefore, a refined context condition should be used.

(2) The number of filtered roles increases as does the total number of roles. An increase in the number of roles contributes to the severity of the SRA problem.

(3) The standard deviation of the number of filtered roles oscillates following the number of roles.

In summary, the E-RBAC model can address the SRA problem. Better results are provided by refining the *COND*s.

Assume the mean number of roles assigned to a user is *n*. Assume the average number of *COND*s assigned to a role is *m*. In the simulation, when a user creates a session, E-RBAC sweeps through the user role set and calculates the CONDs of each role. Therefore, the time complexity for E-RBAC to filter roles is *O(nm)*, and its space complexity is *O(1)*.

Importantly, in the simulation, the number of candidate (unfiltered) roles is significantly reduced (recall, especially, **Table 3**) with respect to native RBAC, which has no role filtering. This implies that E-RBAC greatly reduces the user's workload on roles management.

4. Implementation

Apache Shiro is a powerful and flexible open-source security framework that cleanly handles authentication, authorisation, enterprise session management, and cryptography. The Shiro framework has been applied extensively because of its ease of use and understanding, particularly in network environments. It provides a clean and intuitive API, simplifying task of securing applications for developers. The authorisation module in this framework is the realisation of the standard RBAC model. We modified the Shiro security framework and added a ROLE FILTERING module in order to address the SRA problem. The whole code and data are available at [33].

4.1. Architecture

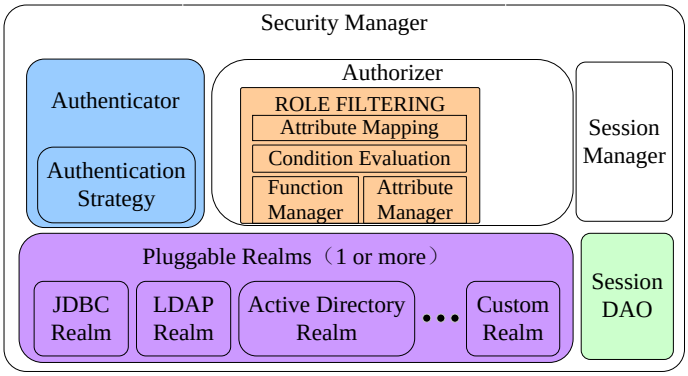

Fig. 5. Modified Shiro architecture

The model architecture is illustrated in **Fig. 5**. The original Shiro framework authorisation provides a simple access control interface: provided the user/role and role/permissions are assigned, it uses the *hasRole* method to determine whether the user has a specific role and whether to grant permissions to the user, rather than using the user-activated role set. This authorisation process is simple, but not sufficiently safe and flexible, because it renounces the session of the RBAC model. In ROLE FILTERING, we added the session and implemented the role filtering function, whereby the function manager and attribute manager are used for managing the supported functions and *ATTR*s, while attribute mapping and condition evaluation are used for *CONS* evaluation.

4.2. Static Architecture

To maintain its simplicity and ease of use, we implemented very few changes in the Shiro framework and made the new ROLE FILTERING an optional module that can be applied according to developer decisions. As illustrated in **Fig. 6**, in this paper we introduce the novel *Subject* and *Session* classes in Shiro and present the entire ROLE FILTERING module, without covering the whole implementation of the framework.

The *hasRole* method of the *Subject* class is the implementation of the access control method in Shiro. We added the *ChangeAttributeValue* and *AddActiveRole* methods, based on the *hasRole* method, to the *Subject* class. The *Session* class in Shiro is the implementation of the network session function, and we use this class to implement the session functionality in the RBAC model. The *Session* class has three properties: *ActivatedRoles*, *Attributes*, and *FilteredRoles*. *ActivatedRoles* is used to store the roles activated by the user, *Attributes* is used to store the set of *ATTR*s assigned to the user, and *FilteredRoles* is used to store the roles that meet the *CONS*s. Using the *Session* class, we changed the original *hasRole* method to determine whether *ActivatedRoles* contains a specific role, rather than whether the user has that role assigned; that is, it is used to determine whether the user has activated the specific role. The new *ChangeAttributeValue* method is used to change the value of the attributes stored in *Attributes*. The *AddActiveRole* method, operated by the user to activate a role, places the role in the *ActivatedRoles* member.

The ROLE FILTERING module evaluates the *CONS* assigned to a role and stores the role satisfying the *CONS* in the *FilteredRoles* member of the *Session* class. It includes the *SecurityManager*, *Policy*, *Target*, *Condition*, *Match*, *Apply*, *ConfigManager*, *FunctionFactory*, and *AttributeFactory* classes. The *SecurityManager* class hides the internal structure of ROLE FILTERING from the other components that use it. Thus, an external component uses ROLE

FILTERING through the methods provided by *SecurityManager*. The *SecurityManager* class has a private member *PolicyFileLocation*, which contains the address of the policies stored in the *PolicyLibrary* member. *PolicyLibrary* is a collection of *Policy* classes, and a *Policy* consists of the *COND*s assigned to a role. The *Policy* class encapsulates two private member classes, the *Target* and *Condition* classes, which are used to implement attribute mapping and condition evaluation, respectively. Regarding attribute mapping, we evaluate whether the value of an attribute variable is equal to the value of the corresponding attribute constant, taking into account that the compared attributes must have the same attribute ID and type. In this implementation, we support attribute types such as *string*, *Boolean*, *integer*, *time*, and *URI*, among others. The *Target* class encapsulates all the attribute constants to be matched by the user context, and we use the *Match* class to encapsulate individual attribute constants.

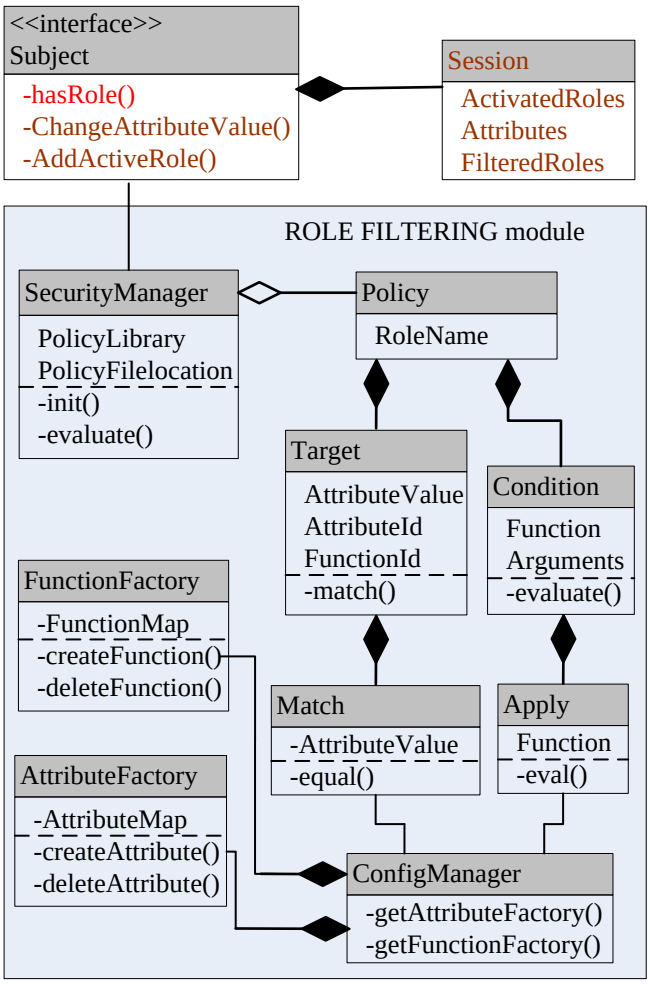

Fig. 6. Class relations

Regarding condition evaluation, we evaluate the relationship between two *ATTR* variables, such as *SCORE()> AVERAGESCORE()*, where the *SCORE* attribute refers to the subject score and the *AVERAGESCORE* attribute refers to the average score. This can be achieved by using the *Condition* class. The *Apply* class encapsulates a function that represents a particular operator that is used to compare the *ATTR*s, and the *Condition* class encapsulates all of the applied objects. Our implementation supports arithmetic, logical, numeric comparison, date and time arithmetic,

non-numeric comparison, and string conversion functions, among others. The *Match* and *Apply* classes contain context attributes and function members; in the initialisation, they need to use *AttributeFactory* and *FunctionFactory* to obtain a specific *ATTR* and function, respectively. The two classes contain the basic *ATTR* and function types, which can be extended by developers according to their actual needs.

4.3. Access Control Decisions

The *hasRole* method also provides access control functions in the modified Shiro framework. However, as mentioned previously, it no longer simply detects whether a role is assigned to the user, but rather detects whether the role is in *ActivatedRoles*. In the new framework, we use *ChangeAttributeValue* to implement the context-changing function. When the value of an attribute stored in *Attributes* is changed by the *ChangeAttributeValue* function, *SecurityManager* immediately re-evaluates the *CONS*s of all roles, adding those that satisfy them to *FilteredRoles*, excluding the other roles, and automatically activating roles according to the different constraint types. Eventually, the user activates a role contained in *FilteredRoles*, instead of depending on those assigned to them. The parameters passed to the *SecurityManager* class include the role name and the set of attributes assigned to the user. The attribute set can be retrieved from the *Attributes* member in the *Session* class. **Fig. 7** describes a *SecurityManager* instance called to evaluate the message sequence chart (MSC) method.

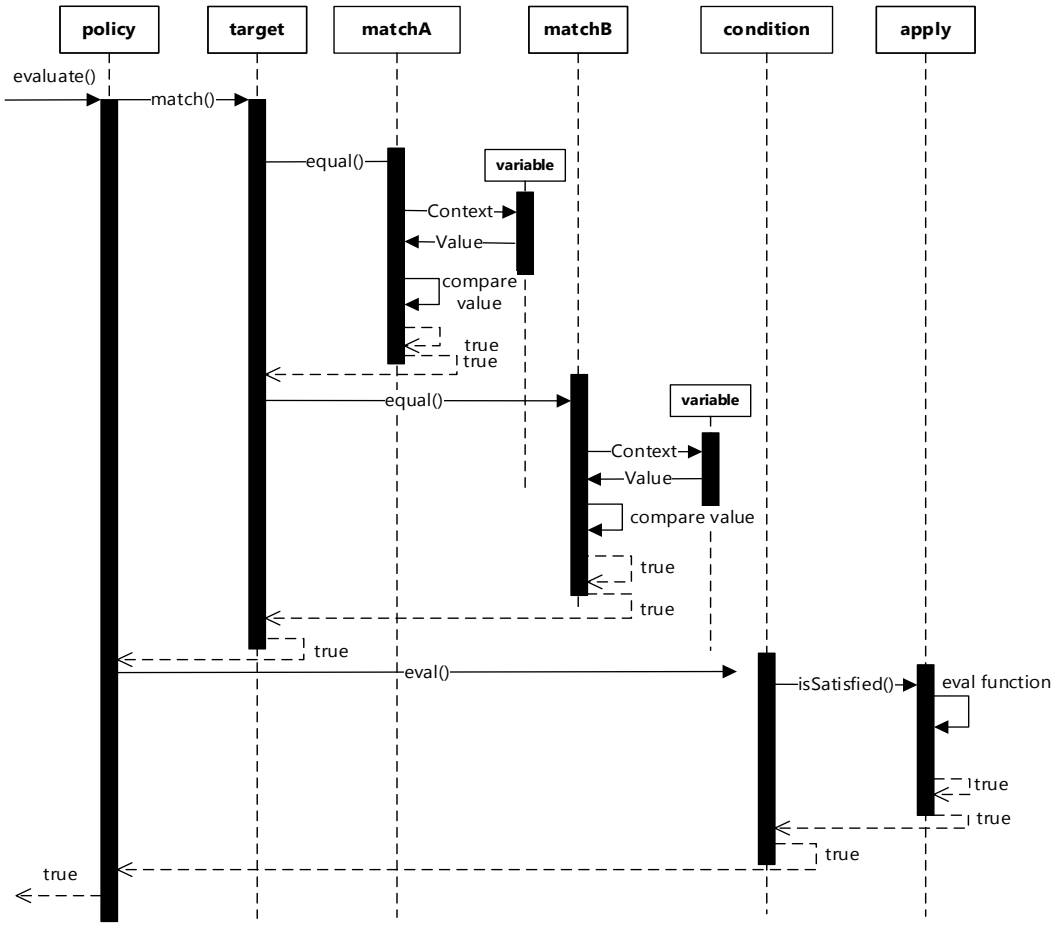

Fig. 7. Evaluation of MSC returning “true”

The *evaluate* method passes the call to the *policy* object, which is an instance of the *Policy* class. Then, the *policy* object uses the *match* method to pass the call to the *target* object, which is an instance of the *Target* class. In this example, there are two constant values that the user *ATTR*s must match: matchA and matchB. The *target* object calls the *equal* method of matchA and matchB. This method compares *ATTR* with the attribute constant; if they are equal, it returns to the *target* object. If all of the objects to be matched return true, the *target* object passes the call back to the *policy* object.

If *target* returns true, the *policy* object calls the *eval* method, whereby *policy* passes the call to the *condition* instance of the *Condition* class. Then, the *condition* object passes the call to the *apply* object, which uses its member function to detect whether *ATTR*s meet the required relationship. If true, the result is returned to the *condition* object, and the *condition* object is returned to the *policy* object; thus, the condition evaluation is over. For simplicity, the *condition* object has only one call to the *apply* object in this example. When the *policy* object returns the result to the *SecurityManager* class, the policy evaluation process is completed.

If any of methods or *isSatisfied* return false, the entire process ends immediately and the *SecurityManager* class returns false to the *hasRole* method. Otherwise, if a role's *CONS* is true, the role is stored in the *ActivatedRoles* member of the *Session* class, which means that the role is activated and its *CONS* does not need to be evaluated again. It should be noted that when the value of an attribute stored in the *Session* class changes, all of the active roles associated with that attribute are immediately deactivated, that is, removed from *ActivatedRoles*.

5. Conclusion

Because of the development of networks, the problem of user-driven role management (meaning users managing session-activated roles) has become increasingly serious. In this paper, the SRA problem, which is a new, practical, user-driven role management problem, was first defined based on reasonable assumptions. Then, to address this problem, we proposed an E-RBAC model that filters roles (reducing the number of candidate roles) by monitoring contexts in real time. The formal definition, framework, and workflow of E-RBAC were introduced. The simulations and implementation of E-RBAC in the Shiro framework were also presented. From the results of the simulation, the following conclusions were drawn: the session-context mechanism of E-RBAC reasonably selects candidate roles by automatically sensing changes in the session, and E-RBAC effectively reduces the number of candidate roles by filtering them.

It should be noted that although our study focused on filtering roles by using context information, it still has limitations in application environments. Here, apart from using context information as a constraint, additional information related to the user or session can be used to filter roles. Despite these limitations, we believe that the E-RBAC model using context constraints is an effective solution to the SRA problem. In future work, we will focus on role filtering by combining user behaviour records and practical context information.

References

[1] Loomis R J, O'Connor A C, "2010 Economic analysis of role-based access control. Final report," *Nist*, 2010.

[2] Elliott A, Knight S., "Role Explosion: Acknowledging the Problem," *Software Engineering Research and Practice*, 349-355, 2010.

[3] Zhang Yue and J B D Joshi, "UAQ: a framework for user authorization query processing in RBAC extended with hybrid hierarchy and constraints," in *Proc. of ACM Symposium on Access Control MODELS and Technologies ACM*, pp. 83-92, 2008. Article (CrossRef Link)
[4] Lu J, Joshi J B, Jin L, & Liu Y, "Towards complexity analysis of User Authorization Query problem in RBAC," *Computers & Security*, vol. 48, pp. 116-130, Feb 2015. Article (CrossRef Link)
[5] Lu J, Xin Y, Zhang Z, et al., "Supporting user authorization queries in RBAC systems by role–permission reassignment," *Future Generation Computer Systems*, 88, 707-717, 2018. Article (CrossRef Link)
[6] Lu J, Wang Z, Xu D, et al., "Towards an Efficient Approximate Solution for the Weighted User Authorization Query Problem," *IEICE TRANSACTIONS on Information and Systems*, 100(8), 1762-1769, 2017. Article (CrossRef Link)
[7] Wickramaarachchi G T, Qardaji W H, & Li N, "An efficient framework for user authorization queries in RBAC systems," in *Proc. of ACM symposium on Access control models and technologies*, pp. 23-32, June 2009. Article (CrossRef Link)
[8] Lu J, Wang Z, Xu D, Tang C, & Han J, "Towards an Efficient Approximate Solution for the Weighted User Authorization Query Problem," *IEICE TRANSACTIONS on Information and Systems*, vol. E100.D(8), pp. 1762-1769, 2017. Article (CrossRef Link)
[9] Samuel A, Sarfraz M I, Haseeb H, Basalamah S, & Ghafoor A, "A framework for composition and enforcement of privacy-aware and context-driven authorization mechanism for multimedia big data," *IEEE Transactions on Multimedia*, vol. 17, pp. 1484-1494, July 2015. Article (CrossRef Link)
[10] Abdella J, Özuysal M, Tomur E, "CA‐ARBAC: privacy preserving using context‐aware role‐based access control on Android permission system," *Security and Communication Networks*, 9(18), 5977-5995, 2016. Article (CrossRef Link)
[11] Kulkarni D, & Tripathi A, "Context-aware role-based access control in pervasive computing systems," in *Proc. of ACM symposium on Access control models and technologies*, pp. 113-122, June 2008.
[12] Hosseinzadeh S, Virtanen S, Díaz-Rodríguez N, & Lilius J, "A semantic security framework and context-aware role-based access control ontology for smart spaces," in *Proc. of International Workshop on Semantic Big Data*, June 2011.
[13] Apache Shiro Reference Documentation. .online, available on https://shiro.apache.org/reference.html#apache-shiro-reference-documentation
[14] Abowd G D, Dey A K, Brown P J, Davies N, Smith M, & Steggles P, "Towards a better understanding of context and context-awareness," in *Proc. of HUC '99 Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pp. 304-307, Sep 1999. Article (CrossRef Link)
[15] Neumann G, & Strembeck M, "An approach to engineer and enforce context constraints in an RBAC environment," in *Proc. of ACM symposium on Access control models and technologies*, pp. 65-79, June 2003. Article (CrossRef Link)
[16] Ryan, Nick, Jason Pascoe, and David Morse, "Enhanced reality fieldwork: the context aware archaeological assistant," *Bar International Series*, pp. 269-274, 1998. Article (CrossRef Link)
[17] Schilit B, Adams N, & Want R, "Context-aware computing applications," in *Proc. of WMCSA '94 Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, pp. 85-90, Dec 1994 Article (CrossRef Link)
[18] Li F, Han Y, Jin C, "Practical access control for sensor networks in the context of the Internet of Things," *Computer Communications*, 89-90, 154-164, 2016. Article (CrossRef Link)
[19] Gansel S, Schnitzer S, Gilbeau-Hammoud A, et al., "Context-aware access control in novel automotive HMI systems," in *Proc. of International Conference on Information Systems Security, Springer, Cham*, 118-138, 2015. Article (CrossRef Link)
[20] Das P K, Joshi A, Finin T, "Personalizing context-aware access control on mobile platforms," in *Proc. of 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC). IEEE*, 107-116, 2017. Article (CrossRef Link)

[21] Qin Z, Sun J, Chen D, et al., "Flexible and Lightweight Access Control for Online Healthcare Social Networks in the Context of the Internet of Things," *Mobile Information Systems*, 1-15, 2017. Article (CrossRef Link)
[22] Kayes A S M, Rahayu W, Dillon T, et al., "Accessing data from multiple sources through context-aware access control," in *Proc. of 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE*, 551-559, 2018. Article (CrossRef Link)
[23] Das P K, Joshi A, Finin T, "Personalizing context-aware access control on mobile platforms," in *Proc. of 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC). IEEE*, 107-116, 2017. Article (CrossRef Link)
[24] Ferraiolo D F, Sandhu R., Gavrila S, Kuhn D R, & Chandramouli R, "Proposed NIST standard for role based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4(3), pp. 224-274, Aug 2001. Article (CrossRef Link)
[25] Kayes A S M, Rahayu W, Dillon T, "An ontology-based approach to dynamic contextual role for pervasive access control," in *Proc. of 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA). IEEE*, 601-608, 2018. Article (CrossRef Link)
[26] Sadat Emami S, Zokaei S, "A context-sensitive dynamic role-based access control model for pervasive computing environments," *The ISC International Journal of Information Security*, 2(1), 47-66, 2010. Article (CrossRef Link)
[27] Luna V, Quintero R, Torres M, et al., "An ontology-based approach for representing the interaction process between user profile and its context for collaborative learning environments," *Computers in Human Behavior*, 51, 1387-1394, 2015. Article (CrossRef Link)
[28] Rathod V, Narayanan S, Mittal S, et al., "Semantically Rich, Context Aware Access Control for Openstack," in *Proc. of 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC). IEEE*, 460-465, 2018. Article (CrossRef Link)
[29] Zhong FJ(Zhong FuJian), Liu YJ(Liu YuJiao), "Context-Aware Dynamic RBAC Model for Application Layer Multicast," in *Proc. of International Conference on Mechatronics, Electronic, Industrial and Control Engineering (MEIC)*, pp. 506-510, APR 2015. Article (CrossRef Link)
[30] Oluwatimi, O (Oluwatimi, Oyindamola), Damiani, ML (Damiani, Maria Luisa), Bertino, E (Bertino, Elisa), "A context-aware system to secure enterprise content: Incorporating reliability specifiers," *COMPUTERS & SECURITY*, vol. 77, pp. 162-178, AUG 2018. Article (CrossRef Link)
[31] Kayes A S M, Rahayu W, Dillon T, et al., "Context-aware access control with imprecise context characterization for cloud-based data resources," *Future Generation Computer Systems*, 93, 237-255, 2019. Article (CrossRef Link)
[32] Milosavljevic G, Sladic G, Milosavljevic B, et al., "Context-sensitive constraints for access control of business processes," *Comput. Sci. Inf. Syst.*, 15(1), 1-30, 2018. Article (CrossRef Link)
[33] https://github.com/wukong19920405/ExtendedRBACInShiro

Gang Liu was born in Yueyang, Hunan, China. He received the M.S and Ph.D. degrees in computer science and technology from Xian Jiaotong University, Xi'an, China in 2001 and 2004, respectively. From 2005 to 2006, he was a associate professor with Guangdong University of Petrochemical Technology, Guangdong, China. His research interests include high-speed computer network and multimedia technology. Since 2007, he has been a faculty member of the School of Computer Science and Technology at Xidian University. His major research interests include embedded system, information security and trusted computing.

Runnan Zhang received B.S. degree from Zhongbei University, Taiyuan, Shanxi, China in 2014. Now he is studying for a Ph.D. degree at Xidian University. His current research interests include research on access control models and quantitative evaluation, conflict detection and resolution, and the optimization of security policies.

Bo Wan was born in Leshan, Sichuan Province, China in 1976. He received the B.S. MS, and Ph.D. degrees from Xidian University, Xi'an, Shaanxi province. He is now an associate professor in the School of Computer Science and Technology at Xidian University, China. His current research interests include input and output technologies and systems, human-computer interaction.

Shaomin Ji received the B.S. degree in computer science and technology from Xi'an University of Technology, Xi'an, Shaanxi province, in 2015 and the M.S degree in computer science and technology from Xidian University in 2018. Now he is studying for a Ph.D. Degree at Xidian University. His research interest is access control model.

Yumin Tian was born in 1964. She received the BS. and MS. degrees in Computer Science and Technology from Xidian University, Xi'an, China. She is now a professor in the School of Computer Science and Technology at Xidian University, China. Her current research interests include input and output technologies and systems, image and video processing and understanding.