EPiC Series in Computing

Volume 45, 2017, Pages 120–132

SCSS 2017. The 8th International Symposium on Symbolic Computation in Software Science 2017

A Comprehensive Formal Solution for Access Control Policies Management: Defect Detection, Analysis and Risk Assessment

Faouzi Jaidi[1], Faten Labbene Ayachi[2], and Adel Bouhoula[3]

[1] faouzi.jaidi@gmail.com
[2] faten.labbene@supcom.rnu.tn
[3] adel.bouhoula@supcom.rnu.tn

Abstract

Nowadays, the access control is becoming increasingly important for open, ubiquitous and critical systems. Nonetheless, efficient Administration, Management, Safety analysis and Risk assessment (AMSR) are recognized as fundamental and crucial challenges in todays access control infrastructures. In untrustworthy environment, the administration of an access control policy, which is a main security aspect, generally raises a critical analysis problem when the administration is distributed and/or potentially un-trusted users contribute to this process. Consequently, collusions attempts and inner threats may take place to generate crucial and invisible breaches to circumvent the policy. To address this issue, we introduce a rigorous and comprehensive solution for an efficient and secure management of access control policies. Our proposal gives a high visibility on the development process of an access control policy and allows in an elegant manner to detect, analyze and assess the risk associated to the policy defects. The strength of our proposal is that it relies on logic-like formalisms to ensure a high surety by verifying the correctness and the completeness of our formal reasoning. We rely on an example to illustrate the relevance of the proposal.

1 Introduction

As Information and Communication Technologies (ICT) are becoming increasingly pervasive, ubiquitous and embedded in everyday object, issues pertaining to the deployment and operation of systems and information security techniques are becoming increasingly important. ICT developments have a direct and wide impact on information and systems security since there is more data and complex assets to be protected in more connected, mobile, open and critical infrastructures. Access control as a typical solution is well adopted to ensure confidential and authorized interactions between components of Information Systems (IS). The importance given to access control for securing IS has been widely studied and justified in literature. It is commonly agreed that defining and setting up an efficient and reliable access control policy is a main requirement for securing critical infrastructures. Nevertheless, setting up a trustworthiness environment of access control and monitoring its compliance and coherence have emerged

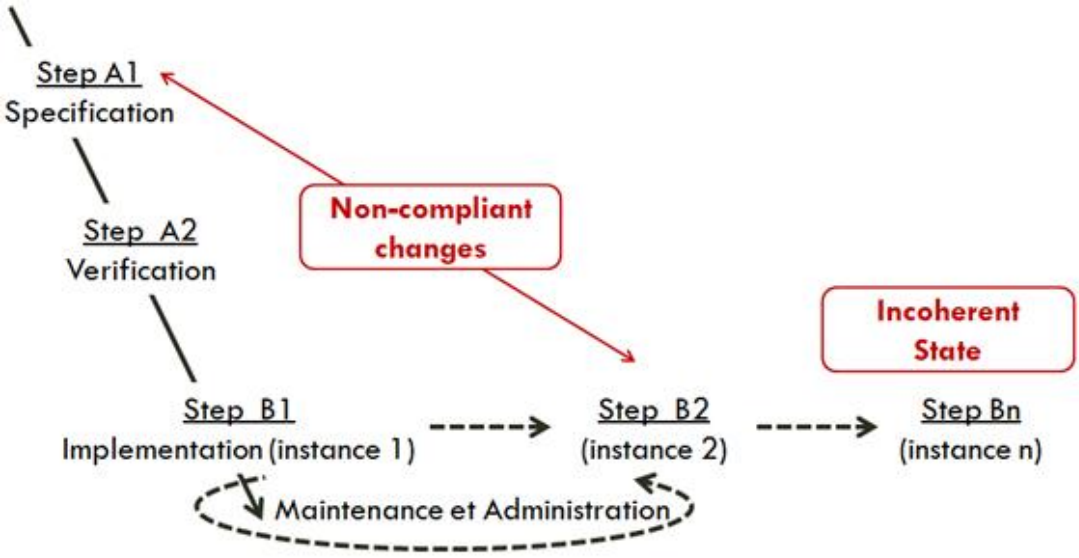

Figure 1: Traditional life-cycle of access control policies.

as complicated and confusing tasks. More, in todays access control infrastructures, efficient administration, management, safety analysis and risk assessment (AMSR) are recognized as fundamental challenges. However, mastering the AMSR tasks is crucial as it would address the urgent desire to ensure a higher security of enterprises IS.

The traditional life-cycle of an access control policy illustrated in figure 1 define three main phases: the specification, the verification and the implementation of the policy. Then the policy evolves with reference to maintenance and administrative tasks and following the evolution of security needs. Throughout its life cycle, the policy can undergo confused alterations: (i) it may record illegal updates and non-compliant changes with regard to its original specification. This generally occurs following an intrusion attempt or an illegal delegation of rights. (ii) It may contain incoherent and conflicting access control rules. This generally occurs following inner threats, collusions attempts and particularly in case the policy is defined by using more than a unique model of access control that lead to redundancy, inconsistency and contradiction in the expression of the policy.

In large scale, open and untrustworthy environments, the administration and management of an access control policy (considered as main security aspects) generally raise a critical analysis problem in case of a distributed administration of the policy and/or potentially un-trusted users (in most cases represent malicious administrators) contribute to the administration process. As a consequence, collusions attempts and inner threats may take place to generate crucial and invisible breaches to circumvent the policy. In a Data Base Management System (DBMS) context, we easily check that as business and private data is exposed to several security threats and attacks, an access control policy is also subject to the same dangers [8]. According to Imperva Application Defense Center reports in 2013 and 2015, Excessive and Unused Privileges and Privilege Abuse are identified as most critical threats in top ten database security threats. Moreover, in the context of healthcare and e-healthcare systems (as a typical critical infrastructures), access control solutions should be rigorous to ensure a higher protection and flexible to treat emergency cases. We check that the simultaneous coupling of two necessary but

contradictory objectives (robustness and flexibility) has a direct influence and a wide impact on the compliance of the deployed access control policy [13].

To address this crucial problem, we introduce in the current paper a reliable and comprehensive solution for an efficient and secure management of access control policies. Our goal is to enhance the integrity of low-level access control policies. To reach this purpose, we aim to: (i) identify and classify attacks and alterations that may corrupt a role based access control (RBAC [21]) policy; (ii) define a formal framework that disposes of complete mechanisms for detecting different alterations; and (iii) evaluate the risk associated to the detected anomalies. Our proposal gives a high visibility on the development process of an access control policy and allows in an elegant manner to detect, analyze and assess the risk associated to the policy defects. Therefore, it helps to monitor the compliance between the low and the high levels of the access control policy in a simple and efficient manner. The strength of the proposal is that it relies on logic-like formalisms to ensure a high surety by verifying the correctness and the completeness of our formal reasoning.

The rest of the paper is organized as follows. We introduce and discuss related works in section 2. In section 3, we present a summary of alterations that may corrupt the compliance of an access control policy. In section 4, we introduce our solution for an efficient deployment and management of reliable and trusted access control policies. We highlight the relevance of our contribution based on an illustrative example. Finally, we conclude the paper and present ongoing works in section 5.

2 Related Works

We classify related works defined in literature to address the thematic of verification and validation (V&V) of access control policies in two main categories: the V&V of the specification and the V&V of the implementation of access control policies.

Several approaches and research works have been well defined to address the verification of the specification of RBAC policies with a main objective is to check the exactitude of a specified policy before proceeding to its implementation. In [1], authors chose to specify access control policies with the security modeling language SecureUML and to verify the specified diagrams based on OCL requests by using the SecureMOVA tool. Authors in [3] adopted a formal approach that consists of specifying the access control policy with SecureUML, encoding the specified diagrams in the Z language and formally analyzing the policy by animating the specification with the animator Jaza tool. Authors in [25] opted for a formal verification of the correctness of the specifications of RBAC policies. They chose to specify the policy with SecureUML and to encode the specifications in the B notation by using the B4Msecure tool. Then, they formally analyze the policy via the animator ProB tool. In [14], authors opted for structuring the set of application roles in a graph of roles that captures different variants of RBAC models. This approach can benefit from well-established results in graph transformation systems [17] and from issues addressed in [18], [20].

Concerning the validation of the implementations of access control policies, the main goal of related works is to verify the correctness of concrete policies regarding the defined security constraints. Existing approaches proposed to represent the set of application roles in different formalisms allowing the analysis, the validation or the optimization of the policies. The proposed contributions deal essentially with the following features: (i) validation of a concrete instance of an access control policy regarding the set of security constraints defined around that policy by using a finite model checker [7]; (ii) detecting by using the formalism of graph of roles existing anomalies of redundancy and inconsistency in the schema of the policy [4]. In [2],

the author proposed to model the access control policy as a graph of roles and to detect illicit transfer of privileges by using graph theory algorithms or a specific LDAP directory schema. Authors in [22] focused primarily on how to enforce and check security constraints and proposed a logical framework to enforce the integrity of access control policies in the context of relational databases.

As for the risk assessment, integrating risk awareness in RBAC systems deals mainly with three main concepts: trust, risk mitigation and risk quantification. Indeed, several approaches proposed to integrate trust relationships into the RBAC model like in [23], [6]. Others focused on constraints-based risk mitigation approach and many attempts proposed to specify SSoD and DSoD constraints like in [5], [24] in RBAC systems. Finally, several works focused on risk quantification approaches and proposed frameworks to quantify the risk associated to access requests such as [19], [15], [16].

Most of discussed works treated the thematic of V&V of access control policies either to check the exactitude of the specifications or to verify the correctness of the implementations regarding defined security constraints. Nonetheless, no complete and rigorous solution is defined for checking the correspondence and monitoring the conformity between high-level and low-level policies. This issue is not addressed sufficiently in literature and needs more and more attention. We address this problematic and we propose a rigorous and comprehensive solution for detecting, analyzing and evaluating the risk associated the policies defects.

3 Compliance Defects in Access Control Policies

We focus in this section on the identification of the defects that may characterize the progress and evolution of an access control policy. We introduce in Table 1 a summary of possible alterations that may corrupt the compliance of a concrete RBAC policy. We present, for each case, a classification of the defect, a qualification of its origin and a theoretical analysis of the associated risk.

We derive four types of anomalies [12]. (i) Anomalies of inconsistency associated to new access control rules defined in the concrete instance and not initially foreseen during the specification of the policy. (ii) Contradiction anomalies are caused by access control rules that become invalid and introduce conflicts to the access control process. (iii) Anomalies of redundancy are linked to access control rules that infer with other rules (which are basically implemented or recently added). (iv) Anomalies related to a partial implementation of the specifications that falsifies the global behavior of the access control process. We qualify the discussed anomalies in two manners: conceptual and optimization problems that may not influence the access control process in a risky manner but should be treated rapidly to avoid any further exploitation and extension; and security problems that have a wide security impact on the access control process and recognized as risky anomalies that should be treated immediately.

4 Formal Framework for Access Control Policies Management

Our approach to address issues related to the deployment and management of access control policies extends the traditional life cycle of access control policies with pertinent phases that we consider as necessary activities for ensuring the trustworthiness and the compliance of security policies. To ensure an efficient and secure deployment and management of reliable access

$N\circ$	**Access Control Policies Defects**			
	Cases	***Classification***	***Qualification***	***Risk***
1	A permission is not assigned to any role	Redundancy / Contradiction	Conceptual/ Optimization Problems	Not risky (Minor to Low risk)
2	A role without permissions	Redundancy / Contradiction		
3	A user without roles and consequently without permissions	Redundancy / Contradiction		
4	Direct assignment of permissions to users	Redundancy / Contradiction		
5	Multiple assignment of the same permissions to a role	Redundancy		
6	Two roles or more share the same permissions	Redundancy		
7	Hidden Users: New users created and assigned roles and permissions bypassing what was initially specified	Inconsistency	Security Problems	Risky (Moderate to High risk)
8	Hidden Roles: New roles created and assigned permissions bypassing what was initially predefined	Inconsistency		
9	Missed Users: removed, missed or deactivated users due to a malicious use of granted rights or a partial implementation of the policy	Partial implementation		
10	Missed Roles: removed, missed or deactivated roles due to a malicious use of granted rights or a partial implementation of the policy	Partial implementation		
11	Renamed Users: users renamed one time or more in order to avoid an audit or a system investigation	Inconsistency		
12	Renamed Roles: roles renamed one time or more in order to avoid an audit or a system investigation	Inconsistency		
13	Hidden access flow : generating a new potential access flow invisible from the outside of the database by means of illegal delegations of rights to users or illegitimate assignments of users to roles, roles to roles or permissions to roles	Inconsistency		
14	Missed access flow : ignoring or removing an authorized access flow via omitting or revoking a set of legal assignments of users to roles, roles to roles or permissions to roles due to an unintentional/intentional erroneous use of priveleges or a partial implementation of the policy	Partial implementation		

Table 1: Analysis of access control policies defects.

control policies, we cover four key security aspects like illustrated in figure 2. (i) The specification, verification and implementation of the policy invariants; (ii) the validation of a concrete (implemented) instance of the policy regarding its original specification; (iii) the assessment of the risk associated to the policy defects; and (iv) the adjustment and optimization of the access control policy schema. In fact, the goal during the specification phase is to capture the maximum of security needs and to distinguish the invariants that must meet any concrete instance of the access control policy. *Security architects* dispose, during this phase, of security modeling languages that extend classical application modeling languages. Verifying the exactitude of the specification and adopting a *Model Driven Architecture* (MDA) approach is highly interesting in systems development and allows especially reaching the implementation via successive refinements of the verified specification. During the validation phase, the reference stage (the specification) and the concrete instance are facing in a logical framework allowing formal reasoning and compliance demonstration. To do so, two preliminary phases are necessary: a reverse engineering phase that allows generating the schema of the implemented policy and a formalization phase for representing the extracted policy in our formal framework. Evaluating the risk associated to the detected defects is highly important that allows to qualify the impact of different anomalies on the system and to respond in an automatic and autonomous manner to critical situations. The optimization phase corrects the redundancy anomalies and helps to check the properties of the graph of roles, to calculate the power of a role, etc. Obtained results allow the adjustment and the up to date of the corresponding policy.

4.1 Formal Detection of the Policy Defects

We formally represent a role based access control policy as follows: **U** (users); **R** (roles); **O** (objects / resources); **A** (access modes); **P** (permissions defined as possible actions on objects, $P \subseteq R \times O$); **AUR** (users-roles assignments, $AUR \subseteq U \times R$); **ARR** (hierarchy of roles, $ARR \subseteq R \times R$) and **APR** (permissions-roles assignments, $APR \subseteq P \times R$). Our validation process requires putting in duality two different notations. Hence, we note $ACP = (U, R, P, AUR, ARR, APR)$ the formal representation of the specified policy and $ACP' = (U', R', P', AUR', ARR', APR')$ the formal representation of the concrete instance of that policy. We developed a set of inference systems that allow checking and validating the compliance between the low-level (concrete instance, ACP') and the high-level (specified instance, ACP) of the RBAC policy and calculating differences between the two versions of the policy. We prove the correctness and the completeness of our formal reasoning.

Our formal reasoning allows detecting the following anomalies: hidden users (**HU**); missed users (**MU**); renamed users (**RU**); hidden roles (**HR**); missed roles (**MR**); renamed roles (**RR**); hidden access flow (**HAF**) that comprises hidden assignments of users to roles (**HAUR**), hidden assignments of roles to roles (**HARR**) and hidden assignments of permissions to roles (**HAPR**); missed access flow (**MAF**) that comprises missed assignments of users to roles (**MAUR**), missed assignments of roles to roles (**MARR**) and missed assignments of permissions to roles (**MAPR**); elementary redundancy (**RED**) and redundancy associated to the DAC Model (**DACRED**) [9].

We present in figure 3, as an example, the inference system that allows detecting the set of hidden assignments of roles to users. We refer to the following theorems to prove the correctness and the completeness of our system.

Theorem 1 [correctness]: the system is correct if for all sets of specified assignments of roles to users (AUR) and implemented assignments of roles to users (AUR') the following assertions are always verified:

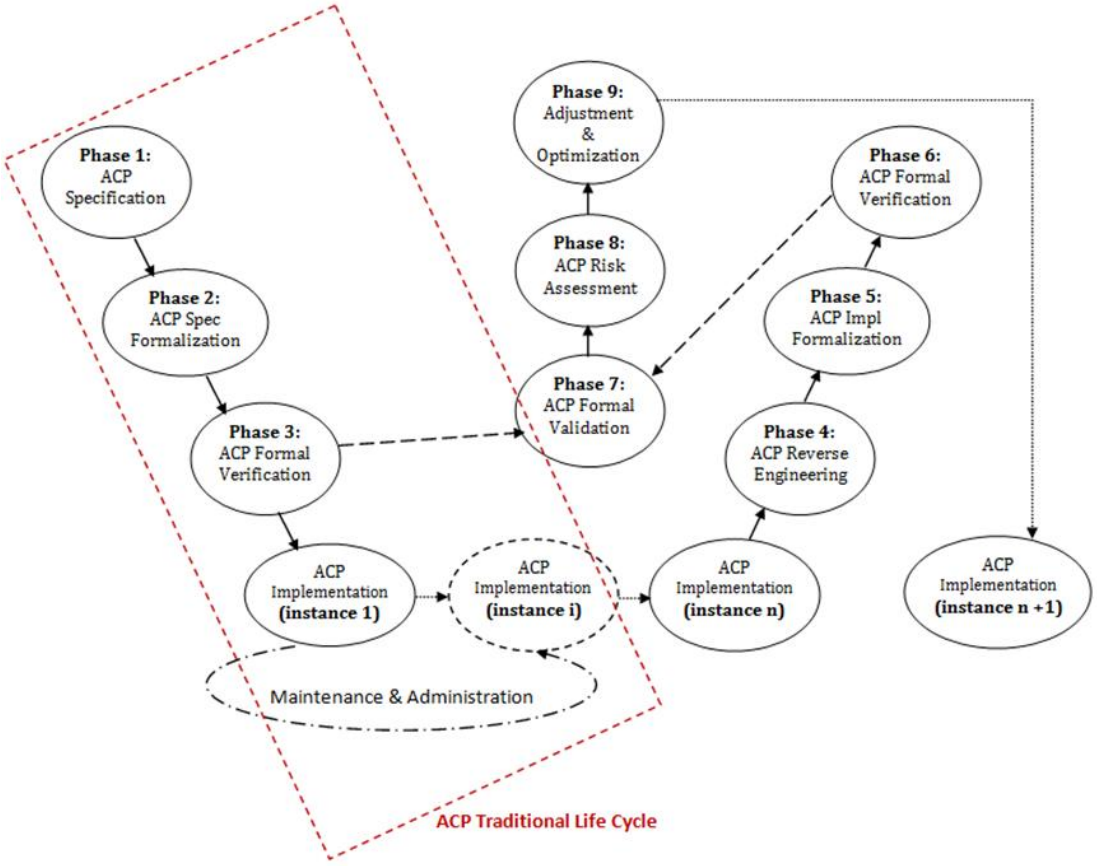

Figure 2: Approach of deployment and management of access control policies.

(i) Compliance: **if** $(U', R', AUR, AUR', \emptyset) \vdash^* Success$ **then** $AUR = AUR'$.

(ii) Non-compliance: **if** $(U', R', AUR, AUR', \emptyset) \vdash^* (U', R', AUR, \emptyset, HAUR)$ with $HAUR \neq \emptyset$ **then** $AUR' - AUR \neq \emptyset$.

Proof:

(i) If $(U', R', AUR, AUR', \emptyset) \vdash^* Success$ then we have an iterative application of the third rule *(Non-hidden assignment)* that implies that $\forall (u, r) \in AUR' \Rightarrow (u, r) \in AUR$. Hence, we have $AUR' = AUR$.

(ii) If $(U', R', AUR, AUR', \emptyset) \vdash^* (U', R', AUR, \emptyset, HAUR)$ with $HAUR \neq \emptyset$ then we have at least one time application of the second rule *(Hidden assignment)* that implies that $\exists u, r | u \in U' \wedge r \in R' \wedge (u, r) \in AUR' \wedge (u, r) \notin AUR$. Hence, we have $AUR' - AUR \neq \emptyset$. Therefore, the defined reasoning is correct.

Theorem 2 [completeness]: the system is complete if for all sets of specified assignments of roles to users (AUR) and implemented assignments of roles to users (AUR') the following assertions are always verified:

Init	$\frac{}{(U', R', AUR, AUR', \emptyset)}$	
Hidden assignment	$\frac{(U', R', AUR, AUR' \cup \{(u,r)\}, HAUR)}{(U', R', AUR, AUR', HAUR \cup \{(u,r)\})}$	$if (u \in U' \wedge r \in R' \wedge (u,r) \notin AUR)$
Non-hidden assignment	$\frac{(U', R', AUR, AUR' \cup \{(u,r)\}, HAUR)}{(U', R', AUR, AUR', HAUR)}$	$if (u \in U' \wedge r \in R' \wedge (u,r) \in AUR)$
Conformity	$\frac{(U', R', AUR, \emptyset, \emptyset)}{Success}$	
Non-conformity	$\frac{(U', R', AUR, \emptyset, HAUR)}{HAUR}$	$if\ HAUR \neq \emptyset$

Figure 3: Formal detection of hidden users-roles assignments.

(i) Compliance: **if** $AUR' - AUR = \emptyset$ **then** $(U', R', AUR, AUR', \emptyset) \vdash Success$.

(ii) Non-compliance: **if** $AUR' - AUR \neq \emptyset$ **then** $(U', R', AUR, AUR', \emptyset) \vdash HAUR$ with $HAUR \neq \emptyset$.

Proof:

(i) If $AUR' - AUR = \emptyset$ then $AUR' = AUR$. Then, $\forall u \in U', r \in R' | (u,r) \in AUR' \Rightarrow (u,r) \in AUR$. Then, by applying the third rule (*Non-hidden assignment*) we have $\forall u \in U', r \in R'$, $(U', R', AUR, AUR' \cup \{(u,r)\}, \emptyset) \vdash (U', R', AUR, AUR', \emptyset)$. Then, via an iterative application of the same rule until $AUR' = \emptyset$, we obtain $(U', R', AUR, AUR' \cup \{(u,r)\}, \emptyset) \vdash (U', R', AUR, \emptyset, \emptyset)$. Hence, by applying the forth rule we have $(U', R', AUR, AUR', \emptyset) \vdash Success$.

(ii) If $AUR' - AUR \neq \emptyset$ then $\exists u, r | (u,r) \in AUR' \wedge (u,r) \notin AUR$. Then, by applying the second rule (*Hidden assignment*), we have $(U', R', AUR, AUR' \cup \{(u,r)\}, \emptyset) \vdash (U', R', AUR, AUR', \emptyset \cup \{(u,r)\})$ that implies that $(U', R', AUR, AUR', \emptyset) \vdash (U', R', AUR, AUR', HAUR)$ with $HAUR \neq \emptyset$. Then, after an iterative application of both second and third rules, we obtain $(U', R', AUR, AUR', \emptyset) \vdash (U', R', AUR, \emptyset, HAUR)$ with $HAUR \neq \emptyset$. Hence, by applying the fifth rule we have $(U', R', AUR, AUR', \emptyset) \vdash HAUR$ with $HAUR \neq \emptyset$. Therefore, the defined reasoning is complete.

4.2 Risk assessment of the Policy Defects

To consider the risk associated to detected anomalies during the check of the compliance of the access control policy, we introduce the necessary formulas that allow computing the risk values of the policy components as well as the policy defects. More, we proceed to classify the risk values associated to the policy defects based on a dynamic risk rating and thresholds [11]. The risk assessment engine is in charge of estimating and re-estimating a risk threshold or a risk rating for each component based on predefined risk factors such as history events, contextual or situational factors, etc. We define an initial risk rating [Minor ($\geq 0\%$ and $< 20\%$); Low ($\geq 20\%$ and $< 40\%$); Moderate ($\geq 40\%$ and $< 60\%$); High ($\geq 60\%$ and $< 80\%$); Extremely High ($\geq 80\%$)] that will be automatically updated based on the evolution of the risk factors.

We evaluate in (1) the risk of a permission $R(P_i)$ as the sum of the probabilities $Pr(k)$ of occurrence of malicious usages $k; k = 1, ..., m$; multiplied by the cost associated to each malicious usage $C(k)$.

$$R(P_i) = \sum_{(k=1)}^{m} Pr(k) * C(k). \tag{1}$$

We compute the risk of the role R_j like illustrated in (2) as the sum of the risk values of all permissions $R(P_i); i = 0, ..., n$; assigned to it.

$$R(R_j) = \sum_{(i=0)}^{n} R(P_i) | P_i \in APR(R_j). \tag{2}$$

We evaluate the risk of the user $R(U_i)$ as shown in (3) as the sum of the risk values of all roles $R_j; j = 0, ..., n$; assigned to it.

$$R(U_i) = \sum_{(j=0)}^{n} R(R_j) | R_j \in AUR(U_i). \tag{3}$$

We consider the risk of an association as the ratio between the risk values of the members of the association. For example, the risk value of the user-role assignment relation $AUR(k)$ that attributes the role R_j to the user U_i is evaluated like defined in (4) as the ratio between the risk of the role and the risk of the user.

$$R(AUR(k)) = \frac{R(R_j)}{R(U_i)}. \tag{4}$$

As for the risk assessment of the policy defects, we seek to determine the impact of each anomaly on the system, (ie) we probe to quantify the influence and the effect of the associated security breaches on the system. From this perspective, we evaluate the risk of an anomaly, like presented in (5), as the ratio between the risk values of the elements of this anomaly and the risk values of the system elements of the same type.

$$R(Anomaly) = \frac{\sum x \mid x \in Anomaly}{\sum y \mid y \in System} * 100\%. \tag{5}$$

For example, the risk of the set Hidden Roles is evaluated in (6) as the sum of the risk values of all hidden roles $R_j; j = 0, ..., n$; divided by the sum of the risk values of all maintained roles. Maintained roles are defined as the intersection between specified and implemented roles.

$$R(HiddenRoles) = \frac{\sum_{(j=0)}^{n} R(R_j) | R_j \in HiddenRoles}{\sum_{(l=0)}^{m} R(R_l) | R_l \in (ROLES\_IMP \cap ROLES)} * 100\%. \tag{6}$$

4.3 Illustrative Example

To highlight the relevance of our proposal, we consider the meeting scheduler as an illustrative example. This system defines four principal actors. A *system user* is able to create/modify/-cancel meetings, add participants to a meeting, and notify the participants about the meeting. The *system administrator* is responsible of managing (creating, modifying related information and deleting) persons. The *supervisor* is a special system user, who has the privilege to modify

or cancel meetings he doesnt own. The *director* is both a user and an administrator. A predefined security property requires that a meeting may only be modified/canceled by its owner and supervisors can notify or cancel meetings they don't own.

The specification of this IS considers the defined actors as application roles that users can perform. It defines also the assignments of the users (Alice, Bob, Charles and David) to their corresponding roles, the hierarchy between roles (Director SystemAdministrator; Director SystemUser; and Supervisor SystemUser) as well as the assignment of permissions to roles. The formalization in the B notation of the specified policy is defined as follows:

```
...
SETS
USERS = {Bob, David, Alice, Charles};
ROLES={Director, Supervisor, SystemAdministrator, SystemUser};
OBJECTS={Meeting, Person, MeetingNotify, MeetingCancel, MeetingModifyStart, MeetingModifyDuration,
     PersonModifyName};
ACTIONS={read, create, modify, delete, fullAccess, execute};
VARIABLES
UsersRolesAssig, RolesHierarchy, PermissionsRolesAssig,
INVARIANT
UsersRolesAssg : USERS --> POW(ROLES) &
RolesHierarchy : ROLES <-> ROLES &
PermissionsRolesAssig : ROLES --> (OBJECTS * POW(ACTIONS))&
INITIALISATION
UsersRolesAssg := {(Bob|-> {Director, SystemUser}), (Charles|-> {SystemUser}), (David|->
     {SystemAdministrator}), (Alice|-> {Supervisor, SystmUser})} ||
RolesHierarchy := {(Director|-> SystemAdministrator), (Director|-> SystemUser), (Supervisor|-> SystemUser)}
     ||
PermissionsRolesAssig := {(SystemUser|-> (Meeting|-> {create, read})), (SystemUser|-> (Meeting|-> {delete,
     modify})), (SystemAdministrator|-> (Meeting|-> {read})), (SystemAdministrator|-> (Person|->
     {fullAccess})), (Supervisor |-> (Meeting|-> {create, read})), (Supervisor|-> (Meeting|-> {delete,
     modify})), (Supervisor|-> (MeetingCancel |-> {execute})), (Supervisor|-> (MeetingNotify|->
     {execute})),(Director|-> (Meeting |-> {create, read})), (Director|-> (Meeting|-> {delete,
     modify})),(Director|-> (Meeting |-> {read})), (Director|-> (Person|-> {fullAccess})) } ||
...
```

For the next, let suppose that after a period of time from the implementation of the system, the policy has evolved to a new state where significant changes are introduced. The encoding of the concrete instance of the policy in the target B notation is defined based on an appropriate SQL-B mapping [10]. The formal representation (in the B notation) of the concrete policy is as follows.

```
...
SETS
USERS_IMP = {Alice,Bob, Charles, Marie, Paul};
ROLES_IMP = {Director, Supervisor, SystemAdministrator, SystemUser, Cosupervisor};
OBJECTS_IMP = {Meeting, Person, MeetingNotify, MeetingCancel, MeetingModifyStart, MeetingModifyDuration,
     PersonModifyName};
ACTIONS_IMP = {read, create, modify, delete, fullAccess, execute};
VARIABLES
UsersRolesAssig_IMP,RolesHierarchy_IMP,PermissionsRolesAssig_IMP, PermissionsUsersAssig_IMP,
INVARIANT
UsersRolesAssg_IMP: USERS_IMP --> POW(ROLES_IMP) &
RolesHierarchy_IMP : ROLES_IMP <-> ROLES_IMP &
PermissionsRolesAssig_IMP : ROLES_IMP --> (OBJECTS_IMP * POW(ACTIONS_IMP)) &
PermissionsUsersAssig_IMP : USERS_IMP --> (OBJECTS_IMP * POW(ACTIONS_IMP)) &
INITIALISATION
UsersRolesAssg_IMP:= {(Bob|-> {Director,SystemUser}), Marie|-> {SystemAdministrator}),(Alice|->
     {Supervisor,SystemUser}), (Charles|-> {SystemUser}), (Paul|-> {Cosupervisor})} ||
RolesHierarchy_IMP := {Director|-> SystemAdministrator), (Director|-> SystemUser), (Supervisor|->
     SystemUser), (Cosupervisor|-> Supervisor)} ||
PermissionsRolesAssig_IMP:={(SystemUser|->(Meeting|-> {create, read, delete, modify})),
     (SystemAdministrator|->(Meeting|-> {read})), (SystemAdministrator|-> (Person|-> {fullAccess})),
     (Supervisor|-> (Meeting|-> {create, read, delete, modify})), (Supervisor |-> (MeetingCancel|->
     {execute}),(Supervisor|-> (MeetingNotify|-> {execute}), (Cosupervisor|->(Meeting|-> {create, read,
     delete, modify})), (cosupervisor|-> (MeetingCancel|-> {execute}), (cosupervisor|-> (MeetingNotify|->
```

```
     {execute})),(Director|-> (Meeting|-> {create, read, delete, modify})), (Director|-> (Meeting|->
     {read})), (Director|-> (Person|-> {fullAccess})) } ||
PermissionsUsersAssig_IMP:= {(Bob|->(Person|->{read}))}
...
```

The formal validation of the compliance between the two instances of the access control policy detects the following anomalies: ***Hidden Users*** = { Marie, Paul }; ***Missed Users*** = { David }; ***Renamed Users*** = $\emptyset$; ***Hidden Roles*** = { Cosupervisor }; ***Missed Roles*** = $\emptyset$; ***Renamed Roles*** = $\emptyset$; ***HiddenACFlow*** = [***Hidden ARR*** = { (Cosupervisor |− > Supervisor)}; ***Hidden AUR*** = { (Marie |− > {SystemAdministrator }), (Paul |− > { Cosupervisor })}; ***Hidden APR*** = { (Cosupervisor |− > (Meeting |− > { create, read, delete, modify })), (Cosupervisor |− > (MeetingCancel |− > { execute })), (Cosupervisor |− > (MeetingNotify |− > { execute }))}]; ***MissedACFlow*** = [***Missed ARR*** = $\emptyset$; ***Missed AUR*** = { (David |− > { SystemAdministrator }) }; ***Missed APR*** = $\emptyset$]; ***DacRedundancy*** = { (Bob |− > Director) * ((Person |− > { fullaccess }) |− > Director) * (Bob |− > (Person |− > { read })) }; ***Redundancy*** = { (Bob |− > Director) * (Bob |− > SystemUser), (Alice |− > Supervisor) * (Alice |− > SystemUser) }.

In order to simplify the evaluation of the risk values associated to the detected anomalies, we consider (as an hypothesis) that the risk of the permissions *execute the operation MeetingCancel* and *execute the operation MeetingNotify* are evaluated to 5 et 4 in a scale that varies from 0 to 5, while the risk values of the rest of the permissions are evaluated to 1 in the same scale.

According to our risk assessment process, we evaluate and classify the risk values associated to the identified anomalies as follows: R(Hidden Users) = 54.54% (***Medium risk***); R(Missed Users) = 15.15% (***Minor risk***); R(Renamed Users) = 0% (***Minor risk***); R(Hidden Roles) = 43.33% (***Medium risk***); R(Missed Roles) = 0% (***Minor risk***); R(Renamed Roles) = 0% (***Minor risk***); R(Hidden ARR) = 71.42% (***High risk***); R(Missed ARR) = 0% (***Minor risk***); R(Hidden AUR) = 66.66% (***High risk***); R(Missed AUR) = 33.33% (***Low risk***); R(Hidden APR) = 25% (***Low risk***); R(Missed APR) = 0% (***Minor risk***).

5 Conclusion

We address in this paper deficiencies and challenges related to access control policies management. We propose a comprehensive solution for deploying reliable and trusted access control infrastructures and for an efficient and secure management of security policies. We define formal mechanisms for detecting, analyzing and assessing the risk associated to concrete RBAC policies defects. The strength of our proposal regarding other approaches is its robustness and completeness. Robustness is justified by formally proving the efficiency of our solution in detecting compliance defects in RBAC policies. Completeness is illustrated by the complete treatment of this problematic that covers the hole life-cycle of access control policies.

Ongoing works address mainly the extension of our risk assessment process towards the definition of a reliable formal framework for enhancing risk management in access control systems.

References

[1] Basin D. A., Clavel M., Doser J., and Egea M. Automated analysis of security-design models. Inf. and Softw. Technology, 51(5): 815-831, 2009.

[2] Ghadi A. Modèle hiérarchique de contrôle d'accès d'unix basé sur un graphe de rôles (in french). PhD, 2010.

[3] Idani A., Ledru Y., Richier J., Labiadh M. A., Qamar N., Gervais F., Laleau R., Milhau J., and Frappier M. Principles of the coupling between uml and formal notations. ANR-08-SEGI-018, 2011.

[4] Huang C., Sun J., Wang X., and Si Y. Security policy management for systems employing role based access control model. Information Technology Journal, 8, 726-734, 2009.

[5] Gligor V. D., Serban I. G., and Ferraiolo D. On the formal definition of separation-of-duty policies and their composition. In Security and Privacy, 1998, 1998.

[6] Feng F., Lin C., Peng D., and Li J. A trust and context based access control model for distributed systems. In Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications, HPCC '08, pp. 629-634, USA, 2008.

[7] Hansen F. and Oleshchuk V. Conformance checking of rbac policy and its implementation. In 1st Information Security Practice and Experience Conference, 144155., 2005.

[8] Jaidi F. and Labbene Ayachi F. An approach to formally validate and verify the compliance of low level access control policies. In Proceedings of 2014 IEEE 17th International Conference on Computational Science and Engineering (CSE), 1550-1557, 2014.

[9] Jaidi F. and Labbene Ayachi F. A formal approach based on verification and validation techniques for enhancing the integrity of concrete role based access control policies. In International Joint Conference, Advances in Intelligent Systems and Computing, 369: 53-64, 2015.

[10] Jaidi F. and Labbene Ayachi F. A reverse engineering and model transformation approach for rbac-administered databases. In 13th International Conference on High Performance Computing and Simulation, HPCS 2015, 2015.

[11] Jaidi F. and Labbene Ayachi F. A risk awareness approach for monitoring the compliance of rbac-based policies. In Proceedings of the 12th International Conference on Security and Cryptography, SECRYPT 2015, pp. 454-459, 2015.

[12] Jaidi F. and Labbene Ayachi F. To summarize the problem of non-conformity in concrete rbac-based policies: Synthesis, system proposal and future directives. In NNGT International Journal of Information Security, 2: 1-12, 2015.

[13] Jaidi F., Labbene Ayachi F., and Bouhoula A. Advanced techniques for deploying reliable and efficient access control: Application to e-healthcare. Journal of Medical Systems, 40: 262, 2016.

[14] Rozenberg G. Handbook of graph grammars and computing by graph transformations. Foundations, 1997;World Scientific, ED, 1997.

[15] Molloy I., Dickens L., Morisset C., Cheng P.-C., Lobo J., and Russo A. Risk-based security decisions under uncertainty. CODASPY'12, 2012.

[16] Ma J., Adi K., Mejri M., and Logrippo L. Risk analysis in access control systems. In Eighth Annual International Conference on Privacy Security and Trust (PST), pp. 160-166, 2010.

[17] Koch M., Mancini L. V., and Parisi-Presicce F. A graph-based formalism for rbac. ACM Transactions on Information and System Security, vol. (5):3, pp. 332-335, 2002.

[18] Nyanchama M. and Osborn S. The role graph model and conflict of interest. ACM Transactions on Information and System Security (TISSEC), 1(2): 333, 1999.

[19] Ni Q., Bertino E., and Lobo J. Risk-based access control systems built on fuzzy inferences. ASIACCS'10, pp. 250-260, USA, 2010.

[20] Baldwin R. Naming and grouping privileges to simplify security management in large databases. In Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, 116132. Los Alamitos, California, USA, 1990.

[21] Sandhu R., Coynek E. J., Feinsteink H. L., and Youmank C. E. Role-based access control models. IEEE Computer, 29(2): 38-47, 1996.

[22] Thion R. and Coulondre S. A relational database integrity framework for access control policies. Journal of Intelligent Information Systems, 38(1): 131-159, 2012.

[23] Chakraborty S. and Ray I. Trustbac: integrating trust relationships into the rbac model for access

control in open systems. In Proceedings of the 11th ACM symposium on Access control models and technologies, SACMAT '06, pp. 49-58, USA, 2006.

[24] Jaeger T. On the increasing importance of constraints. In fourth ACM workshop on Role-based access control, pp. 3342., 1999.

[25] Ledru Y., Idani A., Milhau J., Qamar N., Laleau R., Richier J., and Labiadh M. A. Taking into account functional models in the validation of is security policies. Advanced Information Systems Engineering (CAiSE) Workshops, 83: 592-606., 2011.