Formal description of HOrBAC model

Martin Benoît Azanguezet Quimatio

University of Dschang, Faculty of Sciences

TSOGNONG FIDELE (✉ tsognong.fidele@gmail.com)

University of Dschang, Faculty of Sciences <https://orcid.org/0000-0002-7685-872X>

Marcellin Julius Nkenlifack

University of Dschang, Faculty of Sciences

Research Article

Keywords: Security, Access control, Organizational hierarchy, HORBAC, Redundant authentication

Posted Date: January 10th, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1246591/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Formal description of HOrBAC model

Martin Benoît Azanguezet Quimatio · Fidèle Tsognong · Marcellin Julius Nkenlifack

the date of receipt and acceptance should be inserted later

Abstract Companies' information systems are regularly exposed to internal attacks perpetrated by users who have been granted access to the system. Discretionary, mandatory, role-based and organization-based access control models do not guarantee optimal protection against these attacks because these models trust in users. Therefore, they are unable to protect the system against attacks carried out by authenticated users, especially the super user who can carry out any type of internal attack on information system's data. The objective of this paper is to propose a model that excludes any trust in users. To do so, our model extends the OrBAC (Organization Based Access Control) model by integrating two concepts: the organizational hierarchy and the redundant authentication. The model thus implemented offers a hierarchical and redundant access control to data and processing in an information system based on zero trust in users.

HOrBAC is short name for Hierarchical organisation Based Access Control

B. M. Azanguezet

Research unit in fundamental computer science, engineering and applications(URIFIA), Department of Mathematics and Computer Science, University of Dschang , Cameroon
Tel.: +237-677691096
E-mail: azanguezet@gmail.com

F. Tsognong

Research unit in fundamental computer science, engineering and applications(URIFIA), Department of Mathematics and Computer Science, University of Dschang , Cameroon
Tel.: +237-691929495
E-mail: tsognong.fidele@gmail.com

M. J. Nkenlifack

Research unit in fundamental computer science, engineering and applications(URIFIA), Department of Mathematics and Computer Science, University of Dschang , Cameroon
E-mail: marcellin.nkenlifack@gmail.com

Keywords Security · Access control · Organizational hierarchy · HORBAC · Redundant authentication

1 Introduction

Access control to data and processing in information systems is one of the unsolved problems to date with regard to the internal attacks identified in companies [1]. It is the process of mediating every request to resources and data maintained by a system and determining whether the request should be granted or denied [2, 3]. . The discretionary access model (DAC)[4] limits access to objects based on the identity of their owners. Each user can transmit (indirectly or directly) his rights to any other user. This poses a problem of uncontrollable propagation of rights. The mandatory access control model (MAC) [5] is based on the hierarchy defined by security levels. Security levels are assigned to users and objects, and users can only access objects with a security level lower or equal to their own in the hierarchy. In the MAC model, access is strictly controlled by the administrator. Although this mechanism allows to define a robust security system, the administration of this model becomes very cumbersome when scaling up. To solve this problem, the role-based access control model (RBAC) [6] assigns permissions to roles, users become members of the appropriate roles and inherit the permissions granted to those roles. However, RBAC has been criticized for being static and for its hierarchy not conforming to the real hierarchy of organizations. The Organization-Based Access Control model (OrBAC) [7] takes the concepts of RBAC, incorporates the concept of organization and context, and extends the notion of permissions to include obligations, recommendations and prohibitions. Numerous extensions of RBAC and

OrBAC models have been proposed to address other access control issues [7, 8, 9]. But none of these extensions protects the information system (IS) against internal attacks, especially those carried out by the super user (administrator) of the system. Indeed, all these models trust in users, and particularly the super user who has all the privileges on the whole IS. The latter can use his rights to carry out malicious activities that are very difficult to counter in real time. This is why the super user account is a prime target for malicious users looking to perpetrate attacks.

Some works propose user activity monitoring (UAM) techniques [10, 11]. These techniques involve frequently auditing employee activities in the IS and analyzing the audit data to detect possible attacks. But like all previous models, they detect attacks after the fact because they are unable to monitor the suspicious actions of authenticated users in real time. The access control model based on organization's hierarchy and redundant authentication (HOrBAC) that we formalize aims to control access to data and processing for all users in the IS, including the super user or the DBA (database administrator), in order to prevent attacks and not to observe them. This model is based on two fundamental concepts:

- The organizational hierarchy. This model uses a descending n-ary tree to represent the structure of an organization. This tree is structured in three parts. The root, the internal nodes and the leaves. The root represents the administrative unit with all rights over the rest of the organization. The internal nodes represent the administrative units that have control and supervision rights over the activities of the leaves nodes, which represent the operational units. These operational units model the employees in the organization. This new structuring of the organization reveals two types of roles, the administrative role and the operational role.
- Redundant authentication. This concept models the real-time interactive control of all users' actions, by imposing an authentication to each of these actions. To achieve this, we consider each user action as a request to the hierarchy, waiting for validation. To implement this concept of request validation, two algorithms are proposed and grouped in the concept of *digital parafer*.

These concepts allow to propose an interactive access control model based on zero trust in users and which models the real organizational structure and the administration of the organizations' information systems. This paper proposes a formal description of our model using UML diagrams and first order logic. The

formal language we propose focuses only on permissions in order to avoid conflicts related to the existence of prohibition policies [12, 13].

This paper is organized as follows: the second section presents related works, the third section presents our access control model based on organizational hierarchy and redundant authentication. The fourth section provides the formal description of the proposed model and the specification of security rules. The fifth section presents a case study and the sixth section explains the prototype project.

2 Related Works

Following RBAC [6], many models have been proposed to address limitations such as access contexts, fine-grained access control, interoperability between organizations. Anas Abou El Kalam et al [7] proposed the organization-based access control (OrBAC) to specify contextual rules related to permissions, prohibitions, obligations and recommendations. OrBAC extends the role-based access control model by adding the concepts of organization, activity, view and context to express security rules independently of their implementation. In the same logic, many OrBAC extensions have been proposed with the organization as a central concept. Anas Abou El Kalam et al. [14] proposed the multi-organization-based access control model (Multi-OrBAC) for collaborative, heterogeneous and distributive systems. Frédéric Cuppens et al. [15] proposed the Organization-to-Organization model (O2O) based on OrBAC to manage security policy interoperability.

Vincent C. Hu et al. [16] proposed the attribute-based access control model (ABAC) to provide fine-grained access control. Some researches combined ABAC and RBAC models to have more flexibility [17, 18, 19].

History-Based Access Control (HBAC) grants or declines access by evaluating the history of activities of the inquiring party that includes behaviour, time between requests and content of requests. A.N Ravari et al. [20] proposed a Generalized Temporal History Based Access Control (GTHBAC) which enhances the specification of user-defined authorization rules by constraining time interval and temporal expression over users' history of accesses.

Wenrong Zeng et al. [21] proposed Content-Based Access control model (CBAC) for content-centric information sharing in order to prevent misuse of the privileges by users. CBAC is expected to be deployed on top of RBAC or Multi-level Security (MLS), in the application scenarios where RBAC and MLS will give excessive access rights. The access decision is based on the content similarity.

But none of these models take into account the hierarchy of organizations. These models consider the role hierarchy instead of the organizational hierarchy. Therefore, the role of "Director" in a hospital, for example, will inherit all the rights and privileges of the subordinate roles, whereas operationally the Director does not play all these roles. That is why root users or database administrators (DBAs) inherit all rights in an information system, which technically places them above their superiors in the central administration.

3 Hierarchical organization-based access control model - HOrBAC

The HOrBAC model we formalize is based on organizational hierarchy and excludes the principle of trusting in users (zero trust). In addition to the concepts of organization, activity, context and view defined in OrBAC, we introduce the concepts of administrative unit, operational unit, employees, request, resource and digital parafer to model the organizational hierarchy and redundant authentication. The notion of permission is divided into administrative and operational permissions. Similarly, we add a new "type" attribute to the role concept to distinguish administrative roles from operational roles in an organization.

3.1 Model concepts

3.1.1 Administrative Unit

It models the set of controlling or supervising activities that are grouped into a role and used to administer operational units or other administrative units. It is represented by an internal node in the hierarchical tree of an organization (Figure 1).

3.1.2 Operational Unit

It models the set of related IS state transformation activities grouped into a role and allowing employees to perform basic actions on IS resources. It is represented by a leaf node in the hierarchical tree of an organization (Figure 1).

3.1.3 Employee

It models all physical personnel with an operational role in the organization. It replaces the concept of subject in previous models, which expressed both real subjects (employees) and virtual subjects (users existing only in the organization's database, with no real and unique

relationship with a physical employee). Our model implements a bijection between the employees and the users created in the IS.

3.1.4 Resource

It models the passive entities of the organization (file, folder, database, ...). It replaces the object entity in the previous models, because the concept of object creates confusion between subjects and the resources they manipulate.

3.2 Relations between concepts

In this section, we describe the relationships (mapping) added or modified on the OrBAC model.

3.2.1 Employs

One operational unit employs several employees. We use the relationship *Employs* to model this concept (Figure 2).

3.2.2 Appoints

One employee is appoints to an administrative unit. We use the relationship *Appoints* to model this concept (Figure 3).

3.2.3 Administrate

Several units (operational, administrative) are administered by one administrative unit. We will use the relationship *Administrate* to model this concept (Figure 4).

3.2.4 Uses

A resource can be used in several views of an organization. We use the *Uses* relationship to model this concept (Figure 5).

3.2.5 Defines

We reformulate the definition of the context as described in OrBAC to take into account the concepts of Employee and Resource that we have introduced in our model (Figure 6).

Fig. 1 hierarchical tree Illustration

Fig. 2 Employs relation

Fig. 3 Appoints relation

Fig. 4 Administrate relation

Fig. 5 Uses relation

Fig. 6 Defines relation

3.3 High-level permissions

High-level permissions are applied to abstract entities: Organization, Administrative Unit, Operational Unit, Activity, View, Context. In particular, this model is composed of two high-level permissions: operational permissions and administrative permissions. It is at this level that the traitement modes must be specified.

3.3.1 Assignment of operational permissions

We use the relationship *Operational-Permission* to assign permissions to an operational unit allowing it to perform activities on views (Figure 7).

Fig. 7 Operational Permission

3.3.2 Assignment of administrative permissions

We use the relationship *Administrative-Permission* to assign permissions to an administrative unit allowing it to control the activities of subordinate units, as shown in Figure 8.

3.4 Concrete permissions

3.4.1 Request

It models the set of actions of an employee on one or more resources of the organization in a given context.

Fig. 8 Assigning administrative permissions

A request may or may not require control and approval from the hierarchy. We use the treatment mode property which can be real time for requests requiring no hierarchical control and deferred time for requests requiring control and approval from the hierarchical chain. When the treatment mode associated with a request is deferred time, its processing works according to the Four Eyes principle [22] with several levels of validation. This process has the particularity of being transparent to the issuing employee and the associated action is only executed if the last superior gives a favourable opinion.

Status of a request: A request can be in progress, processed or archived. The diagram in the figure (9) models a request.

Fig. 9 Request model

Concrete permissions are low-level permissions that are applied to concrete entities, namely: Employee, Resource and Action. We define two main types of concrete permissions: the permission to initiate a request and the permission to manage a request.

3.4.2 Permission to initiate a request

The concrete permission for an employee to initiate an action on a resource is noted as *CanInitRequest* (Figure 10). Such a permission can only directly change the state of the IS in a real-time mode.

Fig. 10 Permission to initiate a request

3.4.3 Permission to manage a request

The concrete permission allowing a hierarchical superior to manage a request emit by an employee on a resource is noted *CanManageRequest* (Figure 11). This permission gives its owner the right to validate or reject a request.

Fig. 11 Permission to manage a request

3.5 Security model

Figure 12 summarizes our security model. In particular, we do not have a prohibition security type as in OrBAC since we consider systems in which unauthorized actions are prohibited.

3.6 Redundant authentication

We distinguish two types of access: access to the system and access to resources. By redundant authentication¹, we mean that users are authenticated when accessing the system and each time they want to perform critical actions. The second authentication is performed by the users hierarchical superiors, who check whether the actions they are performing are legitimate. We use this principle to defeat attacks that rely on vectors (emails, shared files, downloaded files from unknown sources, infected removable disks, etc.) to use the privileges of the current user to attack the target machine. These attacks act undercover users and abuse the accesses they have received in order to perform illegitimate actions in the system. These are usually critical actions that users are not used to perform in daily life, such as creating a new user account, encrypting data, deleting important data, stealing data (copying and transferring to

¹ In the field of databases, redundancy consists in having several copies of the same data in the database. In security, this concept is used to secure access to data at several levels in order to reduce the scope of attacks [23]. The invention [24] uses this concept to secure access to cloud resources by combining several identity providers.

Fig. 12 Security Model

an unknown organization), locking data, etc. In reality, these attacks have malicious programs that try to perform bad actions during the user's session, which assumes that the user is already logged in. Therefore, the first authentication to the system is thwarted. By authenticating the user at each critical action (a kind of redundancy since the user is already authenticated for the first time) we can defeat these types of attacks. Nowadays, these attacks are numerous and cause a lot of damage to companies, in particular: Ransomwares [25, 26].

3.7 Digital parafer

This section describes the concept of *digital parafer* that we propose and that works as a reference monitor on the whole organizational tree in order to control the emission of requests and hierarchical treatments. This concept assumes that all the activities of the employees are considered as requests. It consists of two processes: the emission process and the treatment one. The figure 13 gives an overview of the structure of the digital parafer system.

3.7.1 Initializing the digital parafer

This phase consists of initializing the security store by creating organizational units, registering employees in their respective units, creating views and resources, creating activities and actions, defining administrative and operational permissions with their respective contexts and treatment modes.

Fig. 13 Components of the digital parafer

3.7.2 Request emission process

It controls the emission phase of requests initiated by an employee (Figure 14). This process puts all requests whose treatment mode is deferred on hold.

3.7.3 Request treatment process

This process allows line managers to validate or reject a request issued by an employee. In the case of validation, the request is executed and changes the state of the IS, otherwise it is cancelled (Figure 15).

3.8 Life cycle of a request

As defined in our previous article [27], a request is an action issued by an employee on a resource in a given con-

Fig. 14 Request emission process

text. When a request is issued, if the associated treatment mode is real time, it is directly executed in the system. If the associated treatment mode is deferred, it triggers a hierarchical treatment chain. When the last superior in this chain approves (it is assumed that the last hierarchical superior has the last decision), the re-

quest is executed, if he rejects the request, it is cancelled. The treatment done by the intermediate superiors are only transitory treatments allowing the last superior of this chain to make good decisions. Each request has the identifier and the opinion of each superior who processed it so that at each level its history can be

Fig. 15 Request treatment process

Fig. 16 Life cycle of a request

traced. The figure 16 shows the life cycle of a request. The implementation at the database level is done using triggers on the tables that one wants to secure.

4 Logical description of our model

In this section we give the logical formalism of our model. The language described here extends the one proposed for OrBAC formalism using our concepts.

4.1 Proposed Language

The formalism we propose is based on the first-order language L. From this language we can easily express the entities and relationships of HOrBAC model described in Figure (12). Each expression of L contains symbols from a particular vocabulary classified into four groups: constant symbols, individual variables, relationship symbols, and function symbols.

4.1.1 Constant symbols

The constant symbols correspond to the instances of the classes in Figure 12 such as organisation, Work Employee, Resource, Action, Operational Unit, Administrative Unit, View, Activity, Request, and Context. For example CBC Bank is a constant of type organisation, Jacques is constant of type WorkEmployee.

4.1.2 Individual variables and relationship symbols

In the L language, variables are noted by words beginning with lower-case letters. There are individual variables for each entity class. The relationship symbols in L will be denoted by words beginning with uppercase letters corresponding to the relationships in our meta-model diagram.

Each relationship symbol P is considered as a relationship type. According to our classes diagram we have the following relationship symbols:

- *Employs* is a relationship symbol of type (organisation, Work Employee, Operational Unit), which

maps an employee to his operational unit among an organisation.

- *Appoints* is a relationship symbol of type (organisation, Work Employee, Administrative Unit), which maps an employee to his administrative unit among an organisation.
- *Administrates* is a relationship symbol of type (Organisation, Unit, Administrative Unit); this means that in an organization, a unit (operational or administrative) is administered by an administrative unit.
- *Uses* is a relationship symbol of type (organisation, Resource, View); which means that in an organisation a resource is uses in a view.
- *Considers* is a relationship symbol of type (organisation, Action, Activity); which means that in an organisation an action is considered as part of an activity. Actions are basics operations like read, write, select, update, get, put, clone, push, commit, depending of the system which is implemented. As stated in OrBAC, activities are concepts used to categorize actions which have the same objectives. Activity can be modify, consult, activate, cancel, create, delete, erase, download, load, etc and they are independent of the implementation and files systems.
- *Defines* is a relationship symbol of type (organisation, Work Employee, Resource, Action, Context); which means that in an organisation a context is defined for an employee to operate on a resource.
- *Operational Permission* is a relationship symbol of type (organisation, Operational Unit, Activity, View, Context, Treatment Mode); which means that an organisation grants a permission to an operational unit to perform an activity on a view with respect to a given context and treatment mode.
- *Administrative-Permission* is a relationship symbol of type (organisation, Administrative Unit, Activity, View, Context, Treatment Mode); which means that an organisation grants a permission to an administrative unit to approve an activity performed by an operational unit with respect to a given context an treatment mode.

4.1.3 Functions

A function symbol permits to derive informations about entities. Let consider the given sets: E the set of employees, U the set of organisational units (administrative and operational), R the set of resources, V the set of views, A the set of activities and O the set of actions (operations). We have particular functions in our language:

- $unit : E \rightarrow U$, return the unit of an employee;
- $type : U \rightarrow \{Operational, Administrative\}$, return the type of a unit;
- $superior : E \rightarrow E$, return the direct superior of an employee;
- $hierarchy : U \rightarrow U$, return the direct hierarchy of a unit;
- $subunits : U \rightarrow 2^U$, return the children a of unit;
- $employees : U \rightarrow 2^E$, return the employees of a unit;
- $resources : V \rightarrow 2^R$, return the resources of a view;
- $actions : A \rightarrow 2^O$, return the actions which are part of an activity;
- $depth : U \rightarrow N$, return the depth of the sub-tree rooted by the given unit.

4.1.4 Atomic Formulas

Atomic formulas represent facts. They are formed using relationship symbols, variables, constants and functions. Given the following variables: org: Organisation, e: Employee, u: Unit, ou: Operational unit, au: Administrative Unit, q: Activity, a: Action, m: processing mode, c: context. Examples of atomic formulas are given below:

- $Employs(org, e, ou)$
- $Appoints(org, e, au)$
- $Uses(org, r, v)$
- $Consider(org, a, q)$
- $Defines(UDS, e, a, r, c)$
- $Operational-Permission(org, ou, q, v, c, m)$
- $Administrative-Permission(org, au, ou, q, v, c, m)$

4.2 Specifying Policies in HOrBAC

We describe below how operating rules can be specified via our language.

4.2.1 Hierarchy order

In OrBAC the notion of hierarchy is reduced to the inheritance of objects and materialized by prefixed predicates of the keyword *Sub*. We go beyond this inheritance between entities and defines a hierarchy of control over employees and organisational units. The notion of sub-organisation does not exist in our model, but we consider sub-views and sub-activities as defined in OrBAC model as they contribute to reduce administration workload.

The hierarchical order in an organisation is a hierarchy of superiority between units and between employees. To simplify we will use the up arrow symbol ($\uparrow$)

to represent hierarchy between units(resp employees), $u1 \uparrow u2$ means u1 units is superior to u2. For employees $e1 \uparrow e2$ means e1 is the hierarchical superior of e2.

In our language the hierarchical order between the units of an organisation is represented by the predicate *isSuperior*. Through polymorphism we use the same predicate to characterize the hierarchical superiority relationship between employees.

Between employees, the relation $e1 \uparrow e2$ is deduced by the following rule:

$$\begin{aligned} &\forall \text{org: Organisation, } e1, e2: \text{Employee} \\ &isSuperior(org, e1, e2) \leftarrow \\ &type(unit(e1)) = ADMINISTRATIVE \wedge \\ &Administrate(org, unit(e2), unit(e1)). \end{aligned} \quad (1)$$

This means that $e1 \uparrow e2$ is true in any organization org if e2 works in a unit administered by the e1's unit.

The hierarchy order ($\uparrow$) relation is asymmetric, not transitive and not reflexive:

4.2.2 Context and constraints

The context allows one to know the concrete circumstance in which a permission is granted to an employee [28]. The mostly used contexts are time-based, location-based and devices-based. Contexts can be declared in our language using *Define* relation. For instance: Defines(CBC Bank, HR Manager, salary-file, write, at-office), mean that the human resource manager of CBC bank can only write to the salary-file at office.

Constraints are expressed in our model by restrictions on cardinalities and attribute domains:

- There is a single employee appointed to the head of an administrative unit;
- An operational unit is placed under one and only one administrative unit;
- An Administrative Unit is subordinate to another administrative Unit;
- There is only one non-subordinated administrative unit, which is the root unit; The head of the organisation works in this root unit;
- The emitter of a request cannot be the approver;
- In some organisations, multiple job-holding is prohibited and each employee is assigned to only one organisational unit.

4.2.3 Security rules specification

A safety rule reflects how the security status is related to the different permissions that exist in the system.

There are two low level types of permissions in our model: *can suggest* and *can treat* which can also be understood as *can emit a request* and *can approve a request*. In the proposed language, these two concrete permissions can be derived as follows: if an organisation org, in the context c, grants the operational permission to the unit ou to initiate the activity q in the view v, with the treatment mode m, if org Employs the work employee e to unit ou, if org Uses the resource r in the view v, if org Considers that the action a is part of the activity q and if within the organisation org the context c is true between e, a and r, then the employee e can perform the action a on the resource r with or without superior approval (depending on the associated treatment mode m). This result in the following rule:

$$\begin{aligned}
 &\forall e: \text{Employee}, a: \text{Action}, r: \text{Resource} \\
 &\text{CanInitRequest}(e, a, r) \leftarrow \\
 &(\exists \text{org} : \text{Organisation}, m : \text{TreatmentMode}, \\
 &q: \text{Activity}, v: \text{View}, c: \text{Context}) \\
 &\text{Op} - \text{permission}(\text{org}, \text{unit}(e), q, v, c, m) \wedge \\
 &\text{Uses}(\text{org}, r, v) \wedge \\
 &\text{Consider}(\text{org}, a, q) \wedge \text{Defines}(\text{org}, e, a, r, c).
 \end{aligned} \tag{2}$$

If the organisation org, in the context c, grants administrative permission to the administrative unit au to approve activity q performs by organisational unit ou, on view v, with the treatment mode m, if $e \uparrow \text{emittere}'$, if org uses the resource r in view v, if org considers the action a as part of the activity q and if within the organisation org the context c is true between e, a, r, then employee e can approve the request to perform action a on resource r issued by the emitter e' with or without further hierarchical approval (depending on the associated treatment mode m). So we have the following rule:

$$\begin{aligned}
 &\forall e: \text{Employee}, a: \text{Action}, r: \text{Resource}, \\
 &e': \text{Emitter}, c: \text{Context} \\
 &\text{CanManageRequest}(e, e', a, r) \leftarrow \\
 &(\exists \text{org} : \text{Organisation}, q: \text{Activit}, v: \text{Vue}, \\
 &c: \text{Context}, m: \text{Treatment Mode}) \\
 &\text{Ad} - \text{permission}(\text{org}, \text{unit}(e), q, \text{unit}(e'), v, c, m) \wedge \\
 &\text{Uses}(\text{org}, r, v) \wedge \\
 &\text{Considers}(\text{org}, a, q) \wedge \\
 &\text{Defines}(\text{org}, e', a, r, c)).
 \end{aligned} \tag{3}$$

4.2.4 Synthesis

Given an organization org, let's consider the following sets: E the finite set of employees, O the set of actions (operations), R the set of resources, OU the set of operational units, AU the set of administrative units, A the set of activities of the organization, V the set of views, C the set of contexts and $M = \{ \text{Real Time}, \text{Deferred Time} \}$ the treatment modes.

A HOrBAC policy consists of the tuple:

$$\langle E, O, R, A, C, M, V, OA, RV, EC, OU, AU, OP, AP, EU, POU, PAU, \uparrow \rangle \tag{4}$$

Where,

- $OA \subseteq O \times A$: assignment of actions to activities,
- $RV \subseteq R \times V$: assignment of resources to views,
- $EC \subseteq E \times O \times R \times 2^C$: assignment of contexts,
- $OP \subseteq A \times V \times 2^C \times M$: set of operational permissions,
- $AP \subseteq OU \times A \times V \times 2^C \times M$: set of administrative permissions,
- $EU \subseteq E \times (OU \cup AU)$: assignment of employees to units,
- $POU \subseteq OP \times OU$: assignment of permissions to operational units,
- $PAU \subseteq AP \times AU$: assignment of permissions to administrative units,
- $\uparrow$ is a partial order relation on the units, called hierarchical control relation. $\forall u1, u2 : \text{Unit}, u1$ can control the activities of $u2$ if $u1 \uparrow u2$.

By analogy to the analysis work done by [29, 30, 31], for all these relationships, the ones that change constantly due to the evolution of an organization are the assignment of employees to units and the assignment of access contexts to employees. Other relationships such as hierarchical order, assignment of actions to activities, assignment of resources to views, and permissions rarely change.

Let consider the access request $ar_i = (e_i, \alpha_i, r_i, c_i) \in E \times A \times R \times 2^C$ and $\sigma(ar_i)$ the function to evaluate if the employee e_i is allowed to emit the request ar_i . $\sigma(ar_i)$ is true if the conditions (a), (b), (c), (d) and (e) are met simultaneously.

$$\sigma(ar_i) = \bigwedge \left(\begin{array}{l} (a) \exists ou_i \in OU \wedge (e_i, ou_i) \in EU \\ (b) \exists v_i \in V \wedge (r_i, v_i) \in RV \\ (c) \exists a_i \in A \wedge (\alpha_i, a_i) \in OA \\ (d) ar_i = (e_i, \alpha_i, r_i, c_i) \in EC \\ (e) \exists m_i \in M \wedge \exists op_i = (a_i, v_i, c_i, m_i) \in OP \\ \quad \wedge (ou_i, op_i) \in POU \end{array} \right)$$

(5)

The evaluation complexity of $\sigma(ar_i)$ is given by:

$$\begin{aligned} \tau_{\sigma(ar_i)} &= \sum \begin{pmatrix} |EU| \\ |RV| \\ |OA| \\ |EC| \\ |POU| \end{pmatrix} \\ &= \sum \begin{pmatrix} |E| \times (|OU| + |AU|) \\ |R| \times |V| \\ |O| \times |A| \\ |E| \times |O| \times |R| \times |M| \times 2^{|C|} \\ |OU| \times |A| \times |V| \times 2^{|C|} \times |M| \end{pmatrix} \end{aligned} \quad (6)$$

The set of access requests is given by:

$$Q_a = E \times A \times R \times 2^C \quad (7)$$

The total number of access requests that can be emitted in the system is:

$$\mathcal{T}_{ar} = |E| \times |A| \times |R| \times 2^{|C|} \quad (8)$$

To evaluate the completeness of the model it is necessary to verify that the system covers each $ar_i \in Q_a$, since any non-permitted action is forbidden, for that it is necessary to evaluate the following formula:

$$\forall ar_i = (e_i, \alpha_i, r_i, c_i) \in Q_a, \sigma(ar_i) \in \{T, F\} \quad (9)$$

The evaluation time for access requests coverage is given by:

$$\begin{aligned} \mathcal{T}_{ar} &= \sum_{ar_i \in Q_a} \tau_{\sigma(ar_i)} \\ &= \sum_{ar_i \in Q_a} \left(\sum \begin{pmatrix} |E| \times (|OU| + |AU|) \\ |R| \times |V| \\ |O| \times |A| \\ |E| \times |O| \times |R| \times |M| \times 2^{|C|} \\ |OU| \times |A| \times |V| \times 2^{|C|} \times |M| \end{pmatrix} \right) \\ &= |Q_a| \times \tau_{\sigma} \end{aligned} \quad (10)$$

We consider τ_{σ} as the evaluation time for a single access request ar_i since it doesn't depend on i .

The Binary Decision Diagram (BDD) of $\sigma(ar_i)$ is given on figure(17).

Let consider the treatment request

$tr_i = (e'_i, e_i, \alpha_i, r_i, c_i) \in E^2 \times A \times R \times 2^C$ sent by the employee e'_i and $\sigma'(tr_i)$ the function to evaluate if the employee e'_i is allowed to do so. $\sigma'(tr_i)$ is true if

Fig. 17 BDD of $\sigma(ar_i)$ function

the conditions (a), (b), (c), (d), (e), (f), (g) and (h) are met simultaneously.

$$\sigma'(tr_i) = \bigwedge \begin{pmatrix} (a) \exists au_i \in AU \wedge (e'_i, au_i) \in EU \\ (b) \exists ou_i \in OU \wedge (e_i, ou_i) \in EU \\ (c) (ou_i, au_i) \in OU \times AU \\ (d) \exists v_i \in V \wedge (r_i, v_i) \in RV \\ (e) \exists a_i \in A \wedge (\alpha_i, a_i) \in OA \\ (f) (e'_i, \alpha_i, r_i, c_i) \in EC \\ (g) \exists m_i \in M \wedge \\ \exists ap_i = (ou_i, a_i, v_i, c_i, m_i) \in AP \\ (h) (au_i, ap_i) \in PAU \end{pmatrix} \quad (11)$$

The evaluation complexity of $\sigma'(tr_i)$ is given by:

$$\begin{aligned} \tau_{\sigma'(tr_i)} &= \sum \begin{pmatrix} 2|EU| \\ |OU| \times |AU| \\ |RV| \\ |OA| \\ |EC| \\ |PAU| \end{pmatrix} \\ &= \sum \begin{pmatrix} 2|E| \times (|OU| + |AU|) \\ |OU| \times |AU| \\ |R| \times |V| \\ |O| \times |A| \\ |E| \times |O| \times |R| \times |M| \times 2^{|C|} \\ |AU| \times |OU| \times |A| \times |V| \times 2^{|C|} \times |M| \end{pmatrix} \end{aligned} \quad (12)$$

The Binary Decision Diagram (BDD) of $\sigma'(ar_i)$ is given on figure(18).

The treatment chain of an access request ar_i is the set of treatment requests tr_j made by the sender's superiors in sequence, ordered by time. Let consider $\delta(ar_i)$ the treatment chain of the access request ar_i and Q_t the set of treatment requests.

$$\delta(ar_i) = \left\{ \begin{aligned} &tr_j^t = (e'_j, e_j, \alpha_j, r_j, c_j) \in Q_t \mid \\ &\bigwedge \begin{pmatrix} e_i = employee(ar_i) \\ \alpha_i = action(ar_i) \\ r_i = resource(ar_i) \end{pmatrix} \end{aligned} \right\}^{<_t} \quad (13)$$

Fig. 18 BDD of $\sigma'(ar_i)$ function

We use tr_j^t to specify the j^{th} treatment request with it associated time t , meaning that the j^{th} has been sent at t .

The evaluation time of the treatment chain of an access request ar_i is the sum of the evaluation time of each treatment request belonging to the chain. Let $\tau_{\delta(ar_i)}$ the evaluation time of the treatment chain of an access request ar_i .

$$\tau_{\delta(ar_i)} = \sum_{tr_j \in \delta(ar_i)} \sigma'(tr_j) \quad (14)$$

When the treatment mode of an access request ar_i is deferred, it becomes mandatory to evaluate its final status before execution in the system. The final status of an access request (outcome of the request) depends on its treatment mode and the outcome of each treatment request belonging to $\delta(ar_i)$. Let ρ be the algorithm used to combine the outcomes in order to determine the final status, $\gamma(ar_i)$ the final status of the access request ar_i .

$$\gamma(ar_i) = \rho \left(\left\{ \gamma(tr_j) \mid tr_j \in \delta(ar_i) \right\} \right) \quad (15)$$

The set of request outcomes is:

$$\mathcal{O} = \begin{pmatrix} \mathcal{V} = Validated \\ \mathcal{R} = Rejected \\ \mathcal{U} = Undecided \end{pmatrix} \quad (16)$$

The set of combining algorithms is:

$$\mathcal{A} = \begin{pmatrix} \mathcal{LHD} = LastHierarchicalDecision \\ \mathcal{MHD} = MaximumHierarchicalDecision \end{pmatrix}$$

(17)

Combining algorithms are used to avoid outcomes conflicts, the semantic is given below:

- (a) With *lhd* the outcome of the access request to consider is the outcome of the last treatment request in the chain.

$$\gamma_{lhd}(ar_i) = \left(\left\{ \gamma(tr_j^{t'}) \mid tr_j \in \delta(ar_i) \wedge t' = \max(t) \right\} \right) \quad (18)$$

Ex. $\gamma_{lhd}(\mathcal{V}, \mathcal{V}, \mathcal{U}, \mathcal{R})^{<t} = \mathcal{R}$

- (b) With *mhd*: the outcome of an access request is the overriding decision for its treatment requests.

$$\gamma_{mhd}(ar_i) = \left(\left(\vartheta \mid \vartheta \in \max_{\theta \in \mathcal{O}} \left(\left\{ \gamma(tr_j) \mid tr_j \in \delta(ar_i) \wedge \gamma(tr_j) = \theta \right\} \right) \right) \right) \quad (19)$$

Ex. $\gamma_{mhd}(\mathcal{V}, \mathcal{V}, \mathcal{U}, \mathcal{R})^{<t} = \mathcal{V}$

We only give these two algorithms, however the combining algorithms allow for voting decisions and may vary depending on the applications.

4.3 Overview of the administration model

The administration model defines who and how to manage units, permissions and the hierarchy between units. Contrary to ARBAC (Administration model of RBAC) [32], AdOrBAC (Administration model of OrBAC) [33] and the administration model of ABAC [34] where the super-user has all the rights, we propose a cooperative administration which will be a negotiation between the administrator and his superior. The administration of our model works according to the same principles as our model: any action is carried out in the system on the basis of request. Our idea is to reduce the powers of the super-administrator which is a point of failure for the previous administration models. We define below the prerequisites.

Theorem 1 *Let org be an organisation and E the set of its employees, to perform the hierarchical access control in org, E must respect the following constraint: $|E| \geq 2$*

Theorem 2 *The permissions to emit and process a request are mutually exclusives.*

As defined for ARBAC and AdOrBAC, administering a model means managing administration views. In our case we have the following views: AEU (assignment of employees to units), APU (assign permissions to units), AUH (Inter-Unit Hierarchy Management). For implementation within an organization, to ensure control over the requests of users, when they send requests after having authenticated themselves, we check if the sender is the only employee in the organisation, if so the only possible action he can perform in the system is to create the root unit and appoint an employee who will be his superior, this is done directly without the need for hierarchical validation. If the administrator is not the only employee in the organization, the request is executed following the HOrBAC model.

5 Case study

In this section, we show how our model is used to express access control in a sample organization. In this case we consider a bank organization and few resources as shown by Figures 19 and 20. Here we translate the L language given above in Prolog.

5.1 Initialization

5.1.1 Units to units mapping

The following facts define units hierarchy.

Listing 1 map units to units

```
Administrate(bank, brand-manager,
             direction).
Administrate(bank, hr, brand-manager).
Administrate(bank, fr, brand-manager).
Administrate(bank, it, brand-manager).

Administrate(bank, employees-officers,
             hr).
Administrate(bank, customers-officers,
             hr).
Administrate(bank, cashiers, fr).
Administrate(bank, system-admin, it).
```

5.1.2 Employees to units assignment

Employees are assigned to their units using the following facts.

Listing 2 assign employees to units

```
appoints(bank, cfo, direction).
```

```
appoints(bank, bm, brand-manager).
appoints(bank, dhr, hr).
appoints(bank, dfr, fr).
appoints(bank, dit, it).

employs(bank, employees-agent1,
         employees-officers).
employs(bank, customer-agent1,
         customers-officers).
employs(bank, cashier1, cashiers).
employs(bank, superadmin, system-admin)
.
```

5.1.3 Resources to views mapping

Here views are databases, bank-DB-tables, security-DB-tables and Applications. Resources to views mapping are given below.

Listing 3 map resources to views

```
uses(bank, bank_DB, databases).
uses(bank, security_DB, databases).

uses(bank, customer_table,
      bank_db_tables).
uses(bank, employee_table,
      bank_db_tables).
uses(bank, service_table,
      bank_db_tables).
uses(bank, salary_table, bank_db_tables
      ).
uses(bank, user_account_table,
      security_db_tables).
uses(bank, incident-table,
      security_db_tables).
uses(bank, audit-table,
      security_db_tables).
uses(bank, employees_app, applications)
.
uses(bank, customers_app, applications)
.
uses(bank, accounts_app, applications).
```

5.1.4 Actions to activities mapping

Here activities are add, display, delete, activate, modify, save, view, open, authenticate. The following facts map actions to activities. We give few actions in order to simplify.

Listing 4 map actions to organisation activities

Fig. 19 Illustration of bank's structure

Fig. 20 Illustration of resources

```

consider(bank, select, display).
consider(bank, update, modify).
consider(bank, backup, save).
consider(bank, insert, add).
consider(bank, activate, update).
consider(bank, delete, delete).
consider(bank, execute, open).
consider(bank, login, authenticate).

```

5.1.5 Contexts mapping

Here we consider three contexts as example: at-office, working-time and owner. The following facts define contexts.

Listing 5 map access contexts to employees

```

define(bank, employees_agent1, execute,
        employees_app, working_time).
define(bank, superadmin, backup,
        security_db, at_office).
define(bank, -, insert, accounts_app,
        at_office).
define(bank, -, update,
        user_account_table, owner).

```

5.1.6 High level permissions assignment

Sample permissions assignment are given below.

Listing 6 assign operational and administrative permissions

```

treatment_mode(real_time).
treatment_mode(deferred_time).
operational_permission(bank,
    employees_officers, open,
    employees_app,

```

```

    at_office, real_time).
operational_permission(bank,
    customers_officers, open,
    customers_app, -, real_time).
operational_permission(bank,
    customers_officers, add,
    customers_table, working_time,
    real_time).
operational_permission(bank,
    customers_officers, delete,
    customers_table, working_time,
    deferred_time).
operational_permission(bank,
    employees_officers, add,
    employees_table, at_office,
    deferred_time).
operational_permission(bank,
    employees_officers, modify,
    salary_table, working_time,
    deferred_time).
operational_permission(bank,
    system_admin, activate,
    user_account_table, at_office,
    deferred_time).
operational_permission(bank, -,
    authenticate, accounts_app,
    at_home, deferred_time).
operational_permission(bank,
    system_admin, delete,
    user_account_table, working_time,
    deferred).
operational_permission(bank,
    system_admin, add,
    user_account_table, -, deferred_time).

```



```

admin_permission(bank, hr,
    employees_officers, delete,
    customers_table, -, deferred_time).
admin_permission(bank, brand_manager,
    employees_officers,
    delete, customers_table, -,
    deferred_time).
admin_permission(bank, director,
    employees_officers, delete,
    customers_table, -, real_time).
admin_permission(bank, it, system_admin,
    add, user_account_table,
    at_office, real_time).

admin_permission(bank, brand_manager, -,
    auhtenticate,
    accounts_app, -, rela_time).

```

5.2 Interrogation

5.2.1 Hierarchy order between units

The rules below check the hierarchy order relation between units.

Listing 7 check units hierarchy order

```

isSuperior(U1, U2):- administrate(bank,
    U2, U1).
isSuperior(director, brand_manager). //
    True
isSuperior(fr, it). //False
isSuperior(hr, X). // X =[
    employees_officers,
    customers_officers]

```

5.2.2 Hierarchy order between employees

The rules below check the hierarchy order between employees.

Listing 8 check employees hierarchy order

```

unit(E, U):- employs(bank, U, E) ;
    appoints(bank, U, E).
isSuperior(bank, E1, E2):- unit(E1, U1)
    ,
    unit(E2, U2), isSuperior(U1, U2).
isSuperior(bank, superadmin, cfo). //
    False
isSuperior(bank, cfo, S). // S = bm

```

5.2.3 Policies interrogation

The rules below derived low levels permissions from high level ones. We add context C and treatment mode M for logic purpose.

Listing 9 check access permissions

```

canInitRequest(E, A, R, C, M):- employs
    (bank, E, U),
    uses(bank, R, V), consider(bank, Q, A),
    defines(bank, U, Q, R, C),
    operational_permission(bank, U, Q, V, C,
        M).
canManageRequest(E1, E2, A, R, C, M):-
    isSuperior(bank, E1, E2),
    canInitRequest(E2, A, R, C, M).

canInitRequest(customers_agent1, insert
    ,
    customers_table, working_time, M).
//M = real-time
canInitRequest(customers_agent1, delete
    ,
    customers_table, working_time, M).
//M = deferred_time
canInitRequest(superadmin, update,
    user_account_table, break_time, M).
// M is empty which means access denied

canManageRequest(bm, customers_agent1,
    delete, customers_table, -, M).
//M = deferred_time
canManageRequest(cfo, customers_agent1,
    delete, customers_table, -, M).
//M = real-time

```

6 Implementation

MotOrBAC [35] has been developed to administrate and simulate security policies based on OrBAC model. Our model has been implemented as a part of an open source identity provider project name's *horbac-idp* which is described and managed throw Jira plateforme ². The main features of the project are : authentication, manage accounts, manage organizations, manage units, manage employees, manage actions and activities, manage resources and views, manage contexts and permissions. For testing purpose we implement a *permissions helper* feature to check and troubleshoot the can-init-request and can-manage-request low level permissions. The server

² <https://psoes.atlassian.net/horbac>

part of the project is implemented as restful apis in java using Spring Boot framework and the web interface is implemented in javascript using angular technologies. The code source is available³ and we have deployed a test version⁴. The figure 21 illustrate the web interface of our prototype. The tree view of units is shown by figure 22. The requests simulation interface is presented by the figure 23.

Fig. 21 Illustration of web interface of our prototype

Fig. 22 Illustration of units tree view

7 Discussion

Our model can be applied in two cases: authentication and authorization, unlike previous models that are only dedicated to authorization. One of the fundamental differences between our model and the previous models is based on the redundant authentication that we apply to secure access to resources. Indeed, these models rely totally on the proof of identity provided by the user

³ The server and the web interface codes are available at <https://www.github.com/orgs/psoes/horbac-client>, <https://www.github.com/orgs/psoes/horbac-idp>

⁴ The Test version can be seen at <https://www.horbac-test.herokuapp.com>

Fig. 23 Requests simulation interface

to access the organization's resources, no matter who is acting behind this identity. And so a user once authenticated to the system can directly perform actions on the system resources, including critical actions. Although this is within the limit of his privileges, a malicious program (or person) can exploit these same permissions to act as the user and destroy the system. To achieve this we assume that there is a direct bijection between the organization's directory and the system's users. To perform critical actions the employee is authenticated a second time by his superiors within the organization.

In addition to this redundancy which reinforces the security of resources, our model respects the organization chart of companies, the notion of hierarchy is oriented by the functions of the company and the authorizations follow this same logic.

However, the involvement of hierarchical superiors (humans) in the access control process can cause saturation of access requests in the system when the system scales up. We addressed this issue in a previous article [27], by adding a new authentication layer based on the analysis of user habits in the system. And this new layer filters out the requests that finally need to be authenticated by the superiors. Furthermore our model can only work in an organisation with at least two employees (the super-user and the owner of the organization).

8 Conclusion

The aim of this article was to propose a formal description of the HOrBAC model. To do so, we first presented the concepts of this model through UML diagrams. Then we used the first order language to express the objects of the model and the relations between them. From this language we defined constants, variables and functions to specify a security policy based on HOrBAC and proposed an example. This model excludes the notion of trusting employees by performing redundant authentication and can be an alternative to fight against impersonation, brute force and ransomware attacks that cause huge losses to companies. One of the

current challenges is to reduce the response time to an access request between the different authentication layers. We have used the first order logic for its expressiveness, but this logic is semi-undecidable. Although some works have tried to propose decidable fragments [36, 37], this language still remains abstract and does not take into account implementation details hence the need to complete this description with an implementation level language. Finally, the open source identity provider project based on this model lacks an integration part with client applications, we plan to complete this implementation in order to have a complete prototype.

Declarations

Funding:

No funding was received for conducting this study.

Competing interests:

The authors have no competing interests to declare that are relevant to the content of this article.

References

1. ENISA, (2020). DOI 10.2824/552242
2. S.D.C. di Vimercati, S. Paraboschi, P. Samarati, *Softw. Pract. Exp.* **33**(5), 397 (2003). DOI 10.1002/spe.513. URL <https://doi.org/10.1002/spe.513>
3. J. Andress, *The basics of information security - understanding the fundamentals of InfoSec in theory and practice, Second edition* (Syngress, 2014). URL <https://www.worldcat.org/oclc/880706587>
4. G. Ahn, in *Encyclopedia of Database Systems, Second Edition*, ed. by L. Liu, M.T. Özsu (Springer, 2018). DOI 10.1007/978-1-4614-8265-9_135. URL https://doi.org/10.1007/978-1-4614-8265-9_135
5. B.M. Thuraisingham, in *Encyclopedia of Database Systems, Second Edition*, ed. by L. Liu, M.T. Özsu (Springer, 2018). DOI 10.1007/978-1-4614-8265-9_214. URL https://doi.org/10.1007/978-1-4614-8265-9_214
6. R. Sandhu, E. Coyne, H. Feinstein, C. Youman, *Computer* **29**(2), 38 (1996). DOI 10.1109/2.485845
7. A.A.E. Kalam, S. Benferhat, A. Miège, R.E. Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, G. Trouessin, in *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003)*, 4-6 June 2003, Lake Como, Italy (2003), p. 120. DOI 10.1109/POLICY.2003.1206966. URL <https://doi.org/10.1109/POLICY.2003.1206966>
8. M. Ben-Ghorbel-Talbi, F. Cuppens, N. Cuppens-Boulahia, A. Bouhoula, *Int. J. Inf. Sec.* **9**(3), 209 (2010). DOI 10.1007/s10207-010-0104-3. URL <https://doi.org/10.1007/s10207-010-0104-3>
9. A.A.E. Hassani, A.A.E. Kalam, A. Bouhoula, R. Abassi, A.A. Ouahman, *Int. J. Inf. Sec.* **14**(4), 367 (2015). DOI 10.1007/s10207-014-0254-9. URL <https://doi.org/10.1007/s10207-014-0254-9>
10. R. Venkatesh, K.R. Sekhar, *Asian Journal of Information Technology* **15** (2016)
11. H.K. Sharma, I. Khanchi, N. Agarwal, P. Seth, P. Ahlawat, *International Journal of Engineering and Advanced Technology* **9** (2019)
12. K. Adi, Y. Bouzida, I. Hattak, L. Logrippo, S. Mankovskii, (2009), vol. 26, pp. 212–226. DOI 10.1007/978-3-642-01187-0_17\$
13. L. Fang, L. Yin, Y. Guo, Z. Wang, F. Li, in *2018 IEEE Military Communications Conference, MILCOM 2018, Los Angeles, CA, USA, October 29-31, 2018* (2018), pp. 1–6. DOI 10.1109/MILCOM.2018.8599819. URL <https://doi.org/10.1109/MILCOM.2018.8599819>
14. A. Abou El Kalam, Y. Deswarte, (ESAT, 2006), pp. 67 – 85. URL <https://hal.archives-ouvertes.fr/hal-00086523>
15. F. Cuppens, N. Cuppens-Boulahia, C. Coma, in *Information Systems Security, Second International Conference, ICISS 2006, Kolkata, India, December 19-21, 2006, Proceedings, Lecture Notes in Computer Science*, vol. 4332, ed. by A. Bagchi, V. Atluri (Springer, 2006), *Lecture Notes in Computer Science*, vol. 4332, pp. 101–115. DOI 10.1007/11961635_7. URL https://doi.org/10.1007/11961635_7
16. V.C. Hu, D.R. Kuhn, D.F. Ferraiolo, *Computer* **48**, 85 (2015)
17. X. Jin, R.S. Sandhu, R. Krishnan, in *Computer Network Security - 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2012, St. Petersburg, Russia, October 17-19, 2012. Proceedings, Lecture Notes in Computer Science*, vol. 7531, ed. by I.V. Kottenko, V.A. Skormin (Springer, 2012), *Lecture Notes in Computer Science*, vol. 7531, pp. 84–96. DOI 10.1007/978-3-642-33704-8_8. URL https://doi.org/10.1007/978-3-642-33704-8_8

18. H. Qi, X. Luo, X. Di, J. Li, H. Yang, Z. Jiang, in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2016, Chengdu, China, October 13-15, 2016*, ed. by B. Xie, X. Xu (IEEE, 2016), pp. 66–71. DOI 10.1109/CyberC.2016.21. URL <https://doi.org/10.1109/CyberC.2016.21>
19. H. Qi, X. Di, J. Li, J. Inf. Secur. Appl. **43**, 53 (2018). DOI 10.1016/j.jisa.2018.09.001. URL <https://doi.org/10.1016/j.jisa.2018.09.001>
20. A.N. Ravari, J.H. Jafarian, M. Amini, R. Jalili, Telecommun. Syst. **45**(2-3), 111 (2010). DOI 10.1007/s11235-009-9239-9. URL <https://doi.org/10.1007/s11235-009-9239-9>
21. W. Zeng, Y. Yang, B. Luo, in *2014 IEEE International Conference on Big Data (IEEE BigData 2014), Washington, DC, USA, October 27-30, 2014*, ed. by J. Lin, J. Pei, X. Hu, W. Chang, R. Nambiar, C.C. Aggarwal, N. Cercone, V.G. Honavar, J. Huan, B. Mobasher, S. Pyne (IEEE Computer Society, 2014), pp. 701–710. DOI 10.1109/BigData.2014.7004294. URL <https://doi.org/10.1109/BigData.2014.7004294>
22. M. OSACI, A.D. CRISTEA, D. GHIUZAN, D.A. BERDIE, ANNALS of Faculty Engineering Hunedoara-International Journal of Engineering (2018)
23. J.P. Boyer, R. Hasan, L.E. Olson, N. Borisov, C.A. Gunter, D. Raila, in *Proceedings of the 2007 ACM workshop on Computer Security Architecture, CSAW 2007, Fairfax, VA, USA, November 2, 2007*, ed. by P. Ning, V. Atluri (ACM, 2007), pp. 54–62. DOI 10.1145/1314466.1314475. URL <https://doi.org/10.1145/1314466.1314475>
24. D.C. Hardt. Redundant multifactor authentication in an identity management system (U.S. Patent US20100132019A1, May 27, 2010)
25. A. Sana, A. Muhammad, I.M. Azhar, M.A. Islam, Journal of Information Assurance and Security **6**(2), 48 (2017). DOI 10.1109/ISDFS52919.2021.9486348
26. D. Farhat, M.S. Awan, in *2021 9th International Symposium on Digital Forensics and Security (ISDFS) (2021)*, pp. 1–6
27. B.M.A. Quimatio, F. Tsognong, SN Comput. Sci. **2**(2), 121 (2021). DOI 10.1007/s42979-021-00515-w. URL <https://doi.org/10.1007/s42979-021-00515-w>
28. F. Cuppens, A. Miège, in *19th Annual Computer Security Applications Conference (ACSAC 2003), 8-12 December 2003, Las Vegas, NV, USA (2003)*, pp. 416–425. DOI 10.1109/CSAC.2003.1254346. URL <https://doi.org/10.1109/CSAC.2003.1254346>
29. A.L. Ferrara, P. Madhusudan, G. Parlato, in *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, ed. by S. Chong (IEEE Computer Society, 2012), pp. 113–125. DOI 10.1109/CSF.2012.28. URL <https://doi.org/10.1109/CSF.2012.28>
30. A.L. Ferrara, P. Madhusudan, T.L. Nguyen, G. Parlato, in *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings, Lecture Notes in Computer Science*, vol. 8559, ed. by A. Biere, R. Bloem (Springer, 2014), *Lecture Notes in Computer Science*, vol. 8559, pp. 184–191. DOI 10.1007/978-3-319-08867-9_12. URL https://doi.org/10.1007/978-3-319-08867-9_12
31. K.K.Q. Dinh, T.D. Tran, A. Truong, in *Future Data and Security Engineering - 4th International Conference, FDSE 2017, Ho Chi Minh City, Vietnam, November 29 - December 1, 2017, Proceedings, Lecture Notes in Computer Science*, vol. 10646, ed. by T.K. Dang, R.R. Wagner, J. Küng, N. Thoai, M. Takizawa, E.J. Neuhold (Springer, 2017), *Lecture Notes in Computer Science*, vol. 10646, pp. 243–261. DOI 10.1007/978-3-319-70004-5_17. URL https://doi.org/10.1007/978-3-319-70004-5_17
32. Y. Zhang, J.B.D. Joshi, in *Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2007, 13-15 August 2007, Las Vegas, Nevada, USA (IEEE Systems, Man, and Cybernetics Society, 2007)*, pp. 196–202. DOI 10.1109/IRI.2007.4296620. URL <https://doi.org/10.1109/IRI.2007.4296620>
33. F. Cuppens, A. Miège, Comput. Syst. Sci. Eng. **19**(3) (2004)
34. S. Jha, S. Sural, V. Atluri, J. Vaidya, IET Inf. Secur. **13**(2), 96 (2019). DOI 10.1049/iet-ifs.2018.5010. URL <https://doi.org/10.1049/iet-ifs.2018.5010>
35. F. Autrel, F. Cuppens, N. Cuppens-Boulahia, C. Coma, in *3rd Conference on Security in Network Architectures and Information Systems (SAR-SSI 2008), Loctudy, France (2008)*, pp. 273–288
36. M. Voigt, Decidable fragments of first-order logic and of first-order linear arithmetic with uninterpreted predicates. Ph.D. thesis, Saarland University, Saarbrücken, Germany (2019). URL <https://publikationen.sulb.uni-saarland.de/handle/20.500.11880/27767>
37. M. Voigt, CoRR **abs/1706.03949** (2017). URL <http://arxiv.org/abs/1706.03949>