Evaluating the comprehensive complexity of authorization-based access control policies using quantitative metrics

Malek Belhaouane, Joaquin Garcia-Alfaro, Hervé Debar

Evaluating the Comprehensive Complexity of Authorization-based Access Control Policies Using Quantitative Metrics

Malek Belhaouane, Joaquin Garcia-Alfaro and Hervé Debar
Institut Mines-Telecom, Télécom SudParis, CNRS Samovar UMR 5157, Evry, France
{*malek.belhaouane, joaquin.garcia_alfaro, herve.debar*}*@telecom-sudparis.com*

Keywords: ICT Security, Authorization, Access Control, Quantitative Security, Security Assurance, Security Metrics.

Abstract: Access control models allow flexible authoring and management of security policies, using high-level statements. They enable the expression of structured and expressive policies. However, they have an impact on the policy characteristics. The complexity of such policies is one of the affected characteristics. We propose a series of quantitative metrics to assess comprehensive complexity of policies. By comprehensive, we mean the difficulty of understanding a policy by administrators. We formalize the concepts of authorization-based access control models, to propose general metrics regardless of the model. We also show the application of the proposed metrics through a content management system (CMS) policy example. We outline a proof-of-concept to evaluate the feasibility of our proposal, based on SELinux policies for a general-purpose CMS.

1 INTRODUCTION

Access control is a security fundamental concern. It covers a wide area of applications. An access control rule can be defined as a decision (e.g., allow or deny) for an access request (a subject who wants to perform an action on an object). As a consequence, an access control policy can simply be modeled as a set of quadruplets <*decision, subject, action, object*>. This is a really simple modeling of access control policies, called identity-based access control (IBAC) modeling. Using IBAC models (Lampson, 1969; Lampson, 1974; Harrison et al., 1976), the number of rules in a policy can dramatically increase. With many rules, the task of generating and managing an access control policy may become difficult and error-prone (Garcia-Alfaro et al., 2006).

To simplify the management of access control rules, modern access control models allow a flexible and easy expression of access control policies by the use of high-level statements (Samarati and De Capitani di Vimercati, 2000). Access control models such as Role-Based Access Control (RBAC) (Sandhu et al., 1996), Attribute-Based Access Control (ABAC) (Yuan and Tong, 2005) or Organization-Based Access Control (OrBAC) (Abou-El-Kalam et al., 2003) attempt to reduce the system administrators effort by introducing a level of abstraction that enables the aggregation of statements. In addition, these models allow the definition of constraints to structure the policy management process. Even when using these models, the tasks of designing and managing access control policies are still not trivial. Abstraction and constraints have an impact on policy characteristics such as security and usability (Beckerle and Martucci, 2013), policy coverage (Martin et al., 2006), and both conflict (Cuppens et al., 2007) and exception management (Garcia-Alfaro et al., 2007).

In this paper, we focus on the comprehensive complexity of a policy as its core characteristic. We mean by comprehensive complexity, the the reasoning effort needed to understand a policy. Complex access control policies are difficult to maintain and tend to have inconsistencies and errors, such as redundant or conflicting rules (Garcia-Alfaro et al., 2008; Garcia-Alfaro et al., 2013). There is a need for tools to measure and evaluate access control characteristics as well as to optimize the policy management process. Our goal is to provide means to evaluate the comprehensive complexity of authorization-based access control policies. By comprehensive, we mean the difficulty of understanding a policy by administrators. We formalize the basics of authorization-based access control models and define quantitative metrics. The metrics are applied to a content management system (CMS) policy example. We also present a proof-of-concept implementation, based on SELinux policies, for a general-purpose CMS to validate the feasibility of our approach.

Section 2 presents our motivation and illustrates some of the basic concepts with a practical example. Section 3 formalizes the concepts of authorization-based access control models. Section 4 defines the comprehensive complexity of policies and present our metrics to measure this characteristic of access control policies. Section 5 describes a practical application of our proposal. Section 6 provides a comparison with related work. Section 7 closes the paper.

2 MOTIVATION

The tasks of authoring and managing access control policies are not necessarily handled by specialists in computer security but frequently delegated, e.g., to human resources, or performed in several places by different people. It is sure that the abstraction introduced in access control models (Sandhu et al., 1996; Yuan and Tong, 2005; Abou-El-Kalam et al., 2003) helps in reducing the effort of administrators, but it has an impact on the policy characteristics such as the complexity of understanding the policy by administrators. In addition, constraints introduced in these models help administrators to design structured policies and to respect functional restrictions. However, such constraints are challenging from two perspectives. First, they require additional checks when administrating the policy. Second, constraints make it more difficult for human to manage and correctly understanding a policy. Managing complex policies is an error-prone task for administrators.

In order to assess strategies to reduce the complexity of policies, we propose metrics to measure their comprehensive complexity. In the sequel, we describe a concrete example about the complexity of access control policies.

Complexity Example

Our example describes the access control policy of a content management system (CMS) providing an electronic newspaper (e.g., based on solutions such as Wordpress (Wordpress Web Site, 2015) or Drupal (The Drupal project, 2015)). We present a simplified version of roles and actions. For simplicity reasons, we omit as well the administration tasks of the assumed CMS.

The scenario defines the type of access a subject has on a post. Subjects are classified into three categories: contributors, publishers and subscribers. We distinguish three types of subscribers: general news subscribers, economic news subscribers and sports news subscribers. The general news subscription is a default subscription for all subscribers. The two other subscriptions are optional. In addition, a subject can not be a contributor and a publisher at the same time. The posts are also classified into three types: general news posts, sports news posts and economic news posts. Each post can only be assigned to one category. Contributors have the right to write any type of posts. Publishers have the permission to modify (write) and publish posts. Subscribers have the permission to only read posts allowed by their subscriptions. Finally, we model the scenario as follows:

- A table of subjects and their attributes (cf. Table 1a).
- A table of objects and their attributes (cf. Table 1b).
- A table describing the policy (cf. Table 2).

In our scenario, we assign to each subject attributes (A1, A2, A3, A4, A5) that reflect its role in the CMS (cf. Table 1a). We also assign to each post attributes (A6, A7, A8) that indicate its type (see Table 1b).

In our scenario, attributes are not the conventional attributes described in the ABAC model. In the ABAC model, an attribute can take different possible values and the attribute value is either a simple value or a set of values. Attributes in our scenario are equivalent to domains and types in the Domain and Type Enforcement model (Badger et al., 1996). For instance, subjects in a DTE environment are equivalent to processes, so that to each subject we can assign a domain. Using domain transition, each subject can be associated to different domains. Objects in a DTE environment are resources to which we can assign only one type. Thus, attributes in our scenario are equivalent to domains for subjects and to types for objects.

Table 2 shows the permissions assigned to particular subjects for reading, writing, or publishing the objects shown in the first column. For instance, the first line expresses that subjects s1 to s225 (225 subjects) are allowed to read objects o1 to o100 (100 objects). This represents $225 \cdot 100 = 22500$ concrete rules. In addition, subjects s201 to s225 (25 subjects) are allowed to write objects o1 to o100 (100 objects) which represent $25 \cdot 100 = 2500$ concrete rules; and subjects s221 to s225 (five subjects) are allowed to publish objects o1 to o100 which represent $5 \cdot 100 = 500$ concrete rules. Following the same reasoning, we count 5250 rules for the second line, and 6250 rules for the third line. This amounts to having 37000 concrete rules in total.

Due to the difficulty in managing a policy with 225 subjects, 200 objects and 37000 rules, we stress

Table 1: Assignment of attributes

(a) Subject-Attribute relationship

Subject	Attributes				
	A1	*A2*	*A3*	*A4*	*A5*
[s_1,s_{100}]	×				
[s_{101},s_{130}]	×	×			
[s_{131},s_{180}]	×		×		
[s_{181},s_{200}]	×	×	×		
[s_{201},s_{220}]				×	
[s_{221},s_{225}]					×

A1: General news subscriber
A2: Sports news subscriber
A3: Economic news subscriber
A4: Contributor
A5: Publisher

(b) Object-Attribute relationship

Object	Attributes		
	A6	*A7*	*A8*
[o_1, o_{100}]	×		
[o_{101}, o_{150}]		×	
[o_{151}, o_{200}]			×

A6: General news post
A7: Sports news post
A8: Economic news post

the need of quantitative metrics to assess the comprehensive complexity of such a policy.

To define metrics, regardless of the model, we formalize in the next section a set of assumptions that summarize the basic concepts of authorization-based access control models.

Table 2: Concrete security rules

Object	Actions		
	read	*write*	*publish*
[o_1, o_{100}]	[s_1, s_{225}]	[s_{201}, s_{225}]	[s_{221}, s_{225}]
[o_{101}, o_{150}]	[s_{101},s_{130}], [s_{181},s_{225}]	[s_{201}, s_{225}]	[s_{221}, s_{225}]
[o_{151}, o_{200}]	[s_{131}, s_{225}]	[s_{201}, s_{225}]	[s_{221}, s_{225}]

3 BASICS OF AUTHORIZATION-BASED ACCESS CONTROL MODELS

Access control mechanisms are used for controlling access rights to resources in a given organization. Concretely, an access control rule indicates if a given subject has the permission to perform an action on a given object. Thus, an access control rule is composed of a set of conditions (subject, action, object) and a decision (e.g., allow or deny the action). Definitions 1 to 3 present common concepts of access control mechanisms.

Definition 1. *Entities.*

A subject is an entity (e.g., a person, a process, a user account) to whom access to an object may be granted. Let S be the set of subjects.

An object is a resource to which access may be granted (e.g., a file or a directory). Let O be the set of objects.

An action is an operation performed by a subject over an object (e.g., read, write, execute). Let A be the set of actions.

The above defined entities are also called concrete entities because they concretely interact with the access control mechanisms.

Definition 2. *Access control decision.*

There are two basic decisions for an access control request: allow or deny. We refer to the set of access control decision as $Z = \{-1, 1\}$*, where* -1 *means DENY and* 1 *means ALLOW.*

In some authorization-based access control models, we find more than two access control decisions. Nevertheless, the set of access control decisions is a finite set.

Definition 3. *Access control concrete rule.*

At the concrete level, an access control rule describes relations among subjects, actions, objects and access control decisions. For instance, the rule $(s, \alpha, o, 1)$ *states that a subject s is allowed to perform an action* α *on an object o.*

We define, $R_c \subseteq S \times A \times O$ *as the set of concrete access control rules in an access control model.* $n_{Rc} = |R_c|$ *is the number of concrete rules.*

Since the number of concrete rules, n_{Rc}, might increase exponentially, the rule set becomes difficult to manage and maintain. To solve this problem, access control models introduce high-level statements to express access control rules. These statements consist of an abstraction of conditions. This generally reduces the number of expressed access control rules.

Definitions 4 to 8 define common high-level concepts used in authorization-based access control models.

Definition 4. *Abstraction of conditions.*

In an authorization-based access control model, there exists at least one of the following abstractions for concrete entities:

1. *Abstraction of subjects: subjects are abstracted using one or more of their attributes or using the role they are playing in an organization. We call Role [1] such an abstraction of subjects element. $\mathcal{R}$ denotes the set of roles.*
2. *Abstraction of actions: actions are abstracted using one or more of their attributes or using the role they are playing in the access control policy. We call Activity [1] such an abstraction of actions element. $\mathcal{A}$ denotes the set of activities.*
3. *Abstraction of objects: objects are abstracted using one or more attributes or using the role they are playing in the access control policy. We call View [1] such an abstraction of objects element. $\mathcal{V}$ denotes the set of views.*

Each of the above defined abstract entities are associated with one or more concrete elements.

After defining abstract entities, relations between abstract and concrete entities are implicitly or explicitly expressed in the model.

Definition 5. *Assignment relations.*

In an authorization-based access control model, the association between concrete entities and abstract entities is characterized by:

- $\mathcal{SR}_a \subseteq S \times \mathcal{R}$, *a many-to-many mapping of subject-to-role assignment relation.*
- $\mathcal{AA}_a \subseteq A \times \mathcal{A}$, *a many-to-many mapping of action-to-activity assignment relation.*
- $\mathcal{OV}_a \subseteq O \times \mathcal{V}$, *a many-to-many mapping of object-to-view assignment relation.*

Definition 6. *Assigned entities.*

Let $r \in \mathcal{R}$, then $assigned\_subjects(r) = \{s \in S | (s,r) \in \mathcal{SR}_a\}$, is the set of subjects assigned to the role r.

Let $a \in \mathcal{A}$, then $assigned\_actions(a) = \{\alpha \in A | (\alpha,a) \in \mathcal{AA}_a\}$, is the set of actions assigned to the activity a.

Let $v \in \mathcal{V}$, then $assigned\_objects(v) = \{o \in O | (o,v) \in \mathcal{OV}_a\}$, is the set of objects assigned to the view v.

Using the abstraction of conditions, access control rules are expressed differently. High-level statements are translated to rules, but only conditions are abstracted. Access control decisions are still unchanged but they are applied to the abstract elements.

Definition 7. *Abstract access control rules.*

At the abstract level, an access control rule describes the relation between abstract elements and access control decisions.

Let $r \in \mathcal{R}, a \in \mathcal{A}, v \in \mathcal{V}$ then the rule $(r,a,v,1)$ states that subjects assigned to the role r are allowed to perform actions abstracted by the activity a on objects assigned to the view v.

We define $\mathcal{R}_a$ as the set of abstract rules in an access control model, where $n_{\mathcal{R}_a} = |\mathcal{R}_a|$ is the number of abstract rules.

In authorization-based access control models, abstract elements can be organized in a flat structure (e.g., flat ABAC) or in a hierarchical structure (e.g., RBAC or OrBAC). In case of hierarchical structures of abstract elements, we define the following definitions.

Definition 8. *Hierarchy relations.*

In an access control model, hierarchy relations among abstract elements is characterized by:

- $\mathcal{H}_\mathcal{R} \subseteq \mathcal{R} \times \mathcal{R}$, *a many-to-many mapping of role-to-role hierarchy relation.*
- $\mathcal{H}_\mathcal{A} \subseteq \mathcal{A} \times \mathcal{A}$, *a many-to-many mapping of activity-to-activity hierarchy relation.*
- $\mathcal{H}_\mathcal{V} \subseteq \mathcal{V} \times \mathcal{V}$, *a many-to-many mapping of view-to-view hierarchy relation.*

Hierarchy structures are defined in authorization-based access control models to simplify the policy definition and management task by introducing the concept of inheritance. We present this concept in the following definition.

Definition 9. *Inheritance of rules.*

Let $(r_1,r_2) \in \mathcal{H}_\mathcal{R}$, then r_2 inherits all abstract rules of r_1. r_1 is called a super-role while r_2 is called a sub-role.

Let $(a_1,a_2) \in \mathcal{H}_\mathcal{A}$, then a_2 inherits all abstract rules of a_1. a_1 is called a super-activity while a_2 is called a sub-activity.

Let $(v_1,v_2) \in \mathcal{H}_\mathcal{V}$, then v_2 inherits all abstract rules of v_1. v_1 is called a super-view while v_2 is called a sub-view.

In an access control model in which abstract elements are hierarchically structured, access control rules are divided in two groups: local rules and inherited rules.

Definition 10. *Sets.*

$\mathcal{R}_L$ denotes the set of local abstract access control rules. $\mathcal{R}_L \subseteq \mathcal{R}_a$.

$\mathcal{R}_i$ denotes the set of inherited abstract access control rules. $\mathcal{R}_i \subseteq \mathcal{R}_a$.

In authorization-based access control models, constraints are a powerful mechanism for laying out a higher-level organizational policy. Constraints allow to regulate the policy management process by respecting some functional restrictions when writing or modifying the policy.

[1] The terms *Role*, *Activity* and *View* are used in the OrBAC model (Abou-El-Kalam et al., 2003).

Definition 11. *Constraints.*

Constraints are special rules over the entities and relations of an authorization-based access control model, that have to be respected when a security policy is designed.

Let C be the set of constraints.

Let $c \in C$, then, $related\_entities(c)$ is the set of entities affected by the constraint.

Most common constraints expressed in authorization-based access control models are separation constraints (e.g., roles *R1* and *R2* are separated, meaning that a subject assigned to *R1* can not be assigned to *R2*, and vice versa), prerequisite constraints, and cardinality constraints (e.g., only four subjects can be assigned to role *R1*).

4 OUR PROPOSED METRICS

In this section, we define a consistent set of metrics and measurements to evaluate the comprehensive complexity of authorization-based access control policies. Our metrics are constructions designed to facilitate the decision making process for security administrators by giving them information about the comprehensive complexity of an access control policy. They guide the administrators in the task of optimizing the complexity of their access control policies.

4.1 Complexity Definition

Before defining metrics to evaluate the complexity of policies, we need to identify a useful definition for complexity. This policy characteristic is ambiguous and can be studied in several ways. The first type of complexity is the computational complexity of a policy. For instance, it can be denoted as the cost for a policy decision point to transform a policy into a configuration file. The second type is the complexity of policy execution. The complexity of executing the policy can be seen as the complexity for a policy enforcement point to enforce a given policy. The third type is the comprehensive complexity or the complexity of understanding a policy by administrators. The latter is the main focus of our study.

We define the comprehensive complexity of an access control policy as the reasoning effort needed to understand the policy by administrators in the purpose of modifying it. If administrators understand well the structure of the organization (entities, hierarchy relations), the set of access control rules and the constraints that regulate the policy, then, presumably they should understand the effect of making a change (Jaeger, 2001).

Comprehensiveness is improved with the abstraction of entities and the creation of hierarchy relations. They reduce the number of entities and rules (some rules and entities are inherited) and structure the policy elements. The use of hierarchy relations has two effects. On the one hand, the use of hierarchy relations will reduce the number of local abstract rules. This will reduce the comprehensive complexity (less rules to understand). On the other hand, hierarchy relations may also create additional reasoning effort when analyzing a policy. We have to find the optimum number of hierarchy relations to decrease the complexity level of the access control policy.

Other structural aspect that could be considered to assess the comprehensive complexity of authorization-based access control models are constraints. Depending on the type of the constraint, it may affect all relations in the access control model. The comprehensiveness of a constraint depends of the number of policy elements affected by the constraint. The complexity may also depend on the number of functions used in the expression of the constraints (e.g., cardinality and comparison). Thus, the complexity measured for an access control policy may not simply be the number of entities, rules and relationships among entities, but also it depends on hierarchy relations and constraints that regulate the policy.

4.2 Measurements

Based on the access control concepts formalized in Section 3, we define a set of measurements that will be used in our metric computation processes.

In the following, $|X|$ denotes the cardinality of the set X.

Definition 12. *Number of concrete entities.*

$n_S = |\bigcup_{\forall r \in \mathcal{R}} assigned\_subjects(r)|$, *represents the number of all assigned subjects in an access control policy.*

$n_A = |\bigcup_{\forall a \in \mathcal{A}} assigned\_actions(a)|$, *represents the number of all assigned actions in an access control policy.*

$n_O = |\bigcup_{\forall v \in \mathcal{V}} assigned\_objects(v)|$, *represents the number of all assigned objects in an access control policy.*

$n_{cE} = n_S + n_A + n_O$, *represents the number of concrete entities in an access control policy.*

Definition 13. *Number of abstract entities.*

Let $A_s = 1$, if the access control model implements an abstraction of subjects.

$n_{\mathcal{R}} = |\mathcal{R}|$, *represents the number of roles in an access*

control policy.
If the model does not implement an abstraction of subjects, then $A_s = 0$ and $n_{\mathcal{R}} = 0$.

Let $A_\alpha = 1$, if the access control model implements an abstraction of actions.
$n_{\mathcal{A}} = |\mathcal{A}|$, represents the number of activities in an access control policy.
If the model does not implement an abstraction of actions, then $A_\alpha = 0$ and $n_{\mathcal{A}} = 0$.

Let $A_o = 1$, if the access control model implements an abstraction of objects.
$n_{\mathcal{V}} = |\mathcal{V}|$, represents the number of views in an access control policy.
If the model does not implement an abstraction of objects, then $A_o = 0$ and $n_O = 0$.

$n_{aE} = n_{\mathcal{R}} + n_{\mathcal{A}} + n_{\mathcal{V}}$, represents the number of abstract elements in an access control policy.

$n_H = |\mathcal{H}_{\mathcal{R}}| + |\mathcal{H}_{\mathcal{A}}| + |\mathcal{H}_{\mathcal{V}}|$, represents the number of hierarchy relations in an access control policy.

$n_G = |\mathcal{SR}_a| + |\mathcal{AA}_a| + |\mathcal{OV}_a|$, represents the number of assignment relations in access control policy.

Definition 14. *Number of rules.*

$n_{Rc} = |R_c|$, represents the number of concrete rules in an access control policy.

$n_{Ra} = |\mathcal{R}_a|$, represents the number of abstract rules in an access control policy.

$n_{RL} = |\mathcal{R}_L|$, represents the number of local abstract rules in an access control policy.

$n_{Ri} = |\mathcal{R}_i|$, represents the number of inherited abstract rules in an access control policy.

$$n_{Ra} = n_{RL} + n_{Ri}$$

Definition 15. *Measurements for constraints.*

$n_C = |C|$, represents the number of constraints in access control policy.

Let $c \in \mathcal{C}$, then, $n_{rC}(c) = |related\_entities(c)|$ represents the number of entities affected by the constraint c and $n_{fC}(c)$ represents the number of functions used in the expression of the constraints c.

4.3 Abstraction Metrics

After defining different measurements related to an access control policy, we propose a set of metrics to evaluate the complexity of a policy.

Metric 1: abstraction of rules

An access control model adds high-level statements to reduce the number of rules. The higher the number of security rules, the higher the complexity of the policy. We propose to calculate the ratio between abstract rules and concrete rules as an indicator for policy complexity.

$$M_1 = \frac{n_{Ra}}{n_{Rc}} \tag{1}$$

- if $M_1 > 1$, then the number of abstract rules n_{Ra} exceeds the number of concrete rules n_{Rc}. The abstraction is not well defined. The access control model increases the complexity of the policy.
- if $M_1 \to 1$, then the number of abstract rules n_{Ra} is approximately equal to the number of concrete rules n_{Rc}. The abstraction does not reduce the number of rules. The policy is complex or trivial (if the n_{Rc} is small).
- if $M_1 \to 0$, then the abstraction reduces the number of rules. The policy is easy to understand. It is more expressive.

Metric 2: abstraction of entities

An access control model, by introducing an abstraction for conditions, reduces the complexity of the policy. An access control policy expressed with less entities is easier to understand for administrators. We propose to calculate the ratio between abstract entities and concrete entities as an indicator for policy complexity.

$$M_2 = \frac{A_s \cdot n_{\mathcal{R}} + A_\alpha \cdot n_{\mathcal{A}} + A_o \cdot n_{\mathcal{V}}}{A_s \cdot n_S + A_\alpha \cdot n_A + A_o \cdot n_O} \tag{2}$$

- if $M_2 > 1$, then the number of abstract entities exceeds the number of concrete entities. The abstraction is not well defined. This may be due to the existence shadowed roles, activities or views.
- if $M_2 \to 1$, then the number of abstract entities is approximately equal to the number of concrete entities. The abstraction does not reduce the number of entities. The policy is complex (existence of shadowed roles, activities or views) or trivial (if the number of concrete entities n_{cE} is small).
- if $M_2 \to 0$, then the abstraction reduces the number of entities. The policy is easy to understand. Abstract entities are expressive.

4.4 Complexity Metric

To understand the policy, an administrator has to understand each entity, each access control rule, each hierarchy relation, each assignment relation, and each constraint. In (Jaeger, 2001), the complexity of policy elements is expected to depend on how many types of information must be understood.

Before defining the comprehensive complexity of a policy, we propose a metric to assess the complexity of constraints. In (Jaeger, 2001) the complexity of a constraint depends not only on number of policy elements affected by the constraint but also on the number of functions that must be computed to verify the constraint. Thus, a large number of related entities

and used functions may indicate an increased reasoning effort for adding new policy elements because the consistency checks are more complex. We introduce $M_C(c)$ as the comprehensive complexity for a given constraint $c \in \mathcal{C}$:

$$M_C(c) = n_{rC}(c) + n_{fC}(c)$$

For each policy element, we propose to affect a model-dependent weight which would reflect the comprehensive complexity of the element:

- w_E: the complexity weight of each concrete or abstract entity.
- w_L: the complexity weight of each local access rule.
- w_i: the complexity weight of each inherited access control rule.
- w_H: the complexity weight of each hierarchy relation.
- w_A: the complexity weight of each assignment relation.

To compute the complexity weights, we leverage from (Miller, 1956), and assume that people think effectively in terms of chunks of information. For instance, in programing the use of code blocks enable programmers to convert programs in to high level concepts such as functions that represent their semantic intents. As the complexity of a particular program increases, the programmers need to create useful chunks to reduce it. Thus, they must re-learn complex each time that they need to use these chunks. This way, for authorization based access control models, the elementary chunks of information are the elements of the policy: concrete entities, abstract entities, rules, hierarchy relations and assignment relations. However, we suppose that the complexity of policy elements are not always equal. Given that different information are stored in each policy element, we assume that their complexity depends on how many types of information must be understood from the element to perform an operation.

Based on the aforementioned assumptions, we evaluate the policy complexity by a weighted sum of policy elements, as expressed in Equation 3.

$$M_3 = w_E \cdot n_{cE} + w_E \cdot n_{aE} + w_L \cdot n_{RL} + w_i \cdot n_{Ri} + w_H \cdot n_H + w_A \cdot n_G + \sum_{c \in C} M_C(c) \quad (3)$$

5 EVALUATION

5.1 Applying the Metrics

In order to evaluate our metrics, we compute the complexity metrics for the example defined in Section 2. We propose to use high-level access control models in order to express the scenario of the example. In the following, we express the policy using the RBAC model and the ABAC model. For each access control model, we determine the complexity weights used in the complexity metric, M_3 (cf. Section 4.4). Finally, we compute the metrics proposed in Section 4 for the resulting policies.

5.1.1 RBAC model

The RBAC model uses an abstraction of subjects, based on the role each subject is playing in the organization. In our scenario, we group actors by the role they have in the CMS. Thus, we have five possible roles in our scenario: General news subscribers (R1), Sports news subscribers (R2), Economic news subscribers (R3), Contributors (R4) and Publishers (R5).

Table 3: RBAC

(a) Subject-Role assignments

Subject	Roles				
	R1	*R2*	*R3*	*R4*	*R5*
[s_1,s_{100}]	×				
[s_{101},s_{130}]		×			
[s_{131},s_{180}]			×		
[s_{181},s_{200}]		×	×		
[s_{201},s_{220}]				×	
[s_{221},s_{225}]					×

R1: General news subscriber role
R2: Sports news subscriber role
R3: Economic news subscriber role
R4: Contributor role
R5: Publisher role

(b) RBAC abstract rules

Object	Actions		
	read	*write*	*publish*
[o_1, o_{100}]	R1,R2,R3,R4,R5	R4,R5	R5
[o_{101}, o_{150}]	R2,R4,R5	R4,R5	R5
[o_{151}, o_{200}]	R3,R4,R5	R4,R5	R5

Table 3a shows the subject-role assignments. Each cell of the table represents the assignment of a group of subject to a specific role. In the first cell (first

line, first column) of Table 3a subject s_1 to s_{100} (100 subjects) are assigned to role R1. The first cell of the table represents $1 \cdot 100 = 100$ assignment relations. Following the same reasoning, the first line expresses 100 assignment relations, the second line 30, the third line 50, the fourth line 40, the fifth 20 and the sixth 5. This amounts to having a total of 245 assignment relations for this RBAC policy ($n_G = 245$).

Using the abstraction presented in the RBAC model, the permissions will be assigned to roles. Then, a subject assigned to a role will have all the permissions assigned to such role.

In the description of the scenario (cf. Section 2), we indicate that a subject could not be a contributor and a publisher at the same time. This functional restriction could be translated to a separation constraint (C1) over the roles R4 (Contributors) and R5 (Publishers).

Table 3b presents the RBAC access control rules. Each cell of the table represents a specific access performed on a given set of posts. When we place a role in a cell, it means that the role has the permission to perform the action (given by the column) on objects (given by the line).

In the first cell (first line, first column) of Table 3b, roles R1 to R5 (five roles) are allowed to read objects o_1 to o_{100} (100 objects). The first cell of the table represents $5 \cdot 100 = 500$ abstract rules. Following the same reasoning, the first line expresses 800 abstract rules, the second line 300, and the third line another 300. This amounts to having a total of 1400 abstract rules for this RBAC policy ($n_{Ra} = 1400$). This number will not increase if we increase the number of subjects. However, it will increase if we add new posts (objects) to the CMS.

The metric M3 applied to the RBAC model will have the following weights:

- $w_E = 1$. For an entity, we have to understand only the information given by the node.
- $w_L = 4$. For an abstract rule in the RBAC model, we have to understand the three elements (role, action, object) and the conclusion that compose the rule.
- $w_i = 0$ and $w_H = 0$. Hierarchy relations are not defined in the RBAC model.
- $w_A = 2$. For and assignment relation, we have to understand two elements (abstract entity, concrete entity).

If we compute the metrics proposed in Section 4 for this RBAC policy, we have the following values:

- $n_{aE} = 5$, $n_{cE} = 428$, $n_{Ra} = n_{RL} = 1400$, $n_{Ri} = 0$, $n_{Rc} = 37000$, $n_H = 0, n_G = 425$
- $M_C(C1) = 2 + 1 = 3$, to understand the constraint C1, an administrator has to understand the two roles (R4 and R5) and the function that verifies if the two roles are separated (comparison of sets function).
- $M1_{RBAC} = \frac{n_{Ra}}{n_{Rc}} = \frac{1400}{37000} = 0.038$.
- $M2_{RBAC} = \frac{n_{\mathcal{R}_i}}{n_S} = \frac{5}{225} = 0.022$.
- $M3_{RBAC} = 1 \cdot n_{cE} + 1 \cdot n_{aE} + 4 \cdot n_{RL} + 2 \cdot n_G + M_C(C1) = 6526$

$M1_{RBAC} \to 0$, $M2_{RBAC} \to 0$ The abstraction is well defined. The policy is easy to understand.

5.1.2 Hierarchical RBAC model

As described in the scenario, any subject subscribed to the Sports news or Economic news is automatically subscribed to General news. In other words, the Sports news subscribers and the Economic news subscribers inherit the access of General news subscribers. Therefore, we can introduce a hierarchy relation in terms of the RBAC model that reflects this situation. General news subscribers (R1) is a super-role for Sports news subscribers (R2) and Economic news subscribers (R3). This means that R2 and R3 inherit all the permissions of R1.

In addition, Publishers (R5) have the same rights as Contributors (R4), plus an additional right which is to publish posts. We can, then, introduce a hierarchy relation to model this situation. R4 is a super-role for R5, which means that R5 inherits all the permissions of R4.

With the introduction of the hierarchy concept, the abstract rule set changes. Table 4 describes the new abstract access control rules in an RBAC-Hierarchical model.

Table 4: RBAC-Hierarchical abstract rules table

Object	Actions		
	read	*write*	*publish*
$[o_1, o_{100}]$	R1,R2*,R3*,R4,R5*	R4,R5*	R5
$[o_{101}, o_{150}]$	R2,R4,R5*	R4,R5*	R5
$[o_{151}, o_{200}]$	R3,R4,R5*	R4,R5*	R5

*: For inherited rules

Table 4 presents the RBAC access control rules after the introduction of the hierarchy relation. Each cell of the table represents a specific access performed on a given set of posts. When we place a role in a cell, it means that the role has the permission to perform the action (given by the column) on objects (given by line). We add the asterisk symbol (i.e., symbol *) for inherited permissions.

In the first cell (first line, first column) of Table 4, roles R1 and R4 (2 roles) are allowed to read objects o_1 to o_{100} (100 objects). Roles R2, R3 and R5 (3 roles) inherit the permission to read objects o_1 to o_{100} (100 objects). The first cell of the table represents $2 \cdot 100 = 200$ local abstract rules and $3 \cdot 100 = 300$ inherited rules. Following the same reasoning, we count 1400 abstract rules for this Hierarchical-RBAC policy ($n_{Ra} = 1400$). The abstract rules are composed of 800 local rules ($n_{RL} = 800$) and 600 inherited rules($n_{Ri} = 600$).

The metric M3 applied to the Hierarchical-RBAC model will have the following weights:

- $w_E = 1$ (cf. Section 5.1.1 for more details).
- $w_L = 4$ (cf. Section 5.1.1 for more details).
- $w_i = 1$. An inherited rule has the same elements as those associated to the local rule except for one element (the role). Thus, to understand an inherited rule, it suffices to understand such an element.
- $w_H = 2$. Each role hierarchy relation involves two roles. Understanding a role hierarchy relation relies on understanding the two entities involved in such hierarchy relations.
- $w_A = 2$ (cf. Section 5.1.1 for more details).

If we compute the metrics proposed in Section 4 for this example, we get the following values for the Hierarchical-RBAC policy summarized in Table 4:

- $n_{aE} = 5, n_{cE} = 428, n_{Ra} = 1400, n_{RL} = 800, n_{Ri} = 600, n_{Rc} = 37000, n_H = 3, n_G = 245$
- $M_C(C1) = 2 + 1 = 3$ (cf. Section 5.1.1 for more details).
- $M1_{H-RBAC} = \frac{n_{Ra}}{n_{Rc}} = \frac{1400}{37000} = 0.038$
- $M2_{H-RBAC} = \frac{n_R}{n_S} = \frac{5}{225} = 0.022$
- $M3_{H-RBAC} = 1 \cdot n_{cE} + 1 \cdot n_{aE} + 4 \cdot n_{RL} + 1 \cdot n_{Ri} + 2 \cdot n_G + 2 \cdot n_H + M_C(C1) = 4732$

We note that $M3_{H-RBAC} < M3_{RBAC}$. We conclude that $M3$ is a relative metric. By introducing hierarchy relations, the complexity has been reduced from 6526 to 4732.

Indeed, the RBAC model reduces the number of rules with the introduction of the subject abstraction. The results can be improved if we abstract, as well as, the object entities. In the sequel, we show such an improvement.

5.1.3 ABAC model

The ABAC model uses an abstraction of subjects and objects, based on their attributes. The access control rules are written using attributes. Each abstract rule states that all subjects that have a given subject attribute are allowed to perform an action α on those objects that have a given object attribute. For instance, in our scenario, we could state that those subjects with attribute A5 (Publisher) are allowed to publish objects with attribute A6 (General news post).

In the description of the scenario (section 2), we indicate that a subject could not be a contributor and a publisher at the same time. This functional restriction could be translated to a separation constraint(C1) over the attributes A4 (Contributors) and A5 (Publishers). Another functional restriction is that a post could only have one attribute (A6, A7 or A8). This latter could be translated to a separation constraint over attributes A6, A7 and A8.

Table 5 presents the ABAC access control rules. Each cell of the table represents a specific access performed on posts having the attribute given by the line. When we place a subject attribute in a cell, it means that the subjects having this attribute have the permission to perform the action (given by the columns) on objects having the attribute given by the line.

Table 5: ABAC abstract rules

Object	Actions		
	read	*write*	*publish*
A6	A1,A2,A3,A4,A5	A4,A5	A5
A7	A2,A4,A5	A4,A5	A5
A8	A3,A4,A5	A4,A5	A5

In the first cell (first line, first column) of Table 5, subjects with attributes A1 to A5 (five attributes) are allowed to read objects having the attribute A6 (one attribute). The first cell of the table represents $5 \cdot 1 = 5$ abstract rules. Following the same reasoning, the first line expresses a total of eight abstract rules. The second line expresses six abstract rule. The third line expresses another six abstract rules. This amounts to having a total of abstract rules for this ABAC policy ($n_{Ra} = 20$). We conclude that with an abstraction of both subjects and objects, the number of abstract rules has dramatically decreased compared to the RBAC policies.

The metric M3 applied to the ABAC model will have the following weights:

- $w_E = 1$ (cf. Section 5.1.1 for more details).
- $w_L = 4$. For an abstract rule in the ABAC model, we have to understand the three elements (subject attribute, action, object attribute) and the conclusion that compose the rule.
- $w_i = 0$ and $w_H = 0$ (cf. Section 5.1.1 for more details).
- $w_A = 2$ (cf. Section 5.1.1 for more details).

The metrics proposed in Section 4 calculated for this ABAC policy, give the following values:

- $n_{aE} = 8,\ n_{cE} = 428,\ n_{Ra} = n_{RL} = 20,\ n_{Ri} = 0,\ n_{Rc} = 37000, n_H = 0.$
- $n_G = 545$, the number of assignment relations is computed using the same reasoning adopted in Section 5.1.1 over the elements of Table 1a and Table 1b.
- $M_C(C1) = 2 + 1 = 3$ (cf. Section 5.1.1 for more details).
- $M_C(C2) = 3 + 1 = 4$, to understand the constraint C2, an administrator has to understand the three attributes (A6, A7 and A8) and the function that verifies if the three attributes are separated (comparison of sets function).
- $M1_{ABAC} = \frac{n_{Ra}}{n_{Rc}} = \frac{20}{37000} = 5.4 \cdot 10^{-4}$
- $M2_{ABAC} = \frac{n_{\mathcal{R}}+n_{\mathcal{V}}}{n_S+n_O} = \frac{5+3}{225+200} = 0.019$
- $M3_{ABAC} = 1 \cdot n_{cE} + 1 \cdot n_{aE} + 4 \cdot n_{RL} + 2 \cdot n_G + M_C(C1) + M_C(C2) = 1613$

Notice that $M3_{ABAC} \ll M3_{RBAC}$. Therefore, we conclude that the more abstraction level we add, the more we reduce the complexity of the policy.

5.2 Discussion

The values of the proposed metrics computed in Section 5.1 confirm the definition of complexity presented in Section 4.1. For the RBAC policy and the hierarchical RBAC policy, we observe that hierarchy relations reduce the complexity of policies ($M3_{H-RBAC} < M3_{RBAC}$). We also note that M_1 and M_2 do not take into consideration the hierarchy relations. In other words, their values do not change. For the ABAC policy, the values of M_1 and M_2 were improved because the ABAC model introduces an abstraction of objects in addition to the abstraction of subjects used in the RBAC model.

The results are confirmed in practice. Indeed, to add a new subject to the ABAC or RBAC policy, it is equivalent to creating the subject and assigning it to a role (for RBAC) or an attribute (for ABAC). Then, the ABAC and RBAC cases would have the same complexity. The process is not the same for objects. Adding a new object in the RBAC policy is equivalent to (1) creating the object and (2) adding all the needed permissions to those roles that shall manipulate it. On the contrary, in the ABAC policy, administrators only need to create the object and assign to it the attributes. Therefore, it is fair to say that the complexity is not the same for the ABAC and RBAC policies.

To reinforce the above remarks, we can also compute the metrics upon the same scenario, but changing the number of subjects and number of objects. Assume, for instance, the following two newspaper examples: in the first one, the number of subscribers largely exceeds the number of contributors; in the second one, subjects are at the same time subscribers and contributors. In the first case, the computed complexity metrics would give the following values for the RBAC policy:

- $n_{aE} = 5,\ n_S = 10^5,\ n_A = 3,\ n_O = 200,\ n_{cE} = 100203,\ n_{Ra} = n_{RL} = 1400,\ n_{Ri} = 0,\ n_{Rc} \approx 10^7,\ n_H = 0, M_C(C1) = 3$
- $M1_{RBAC} = \frac{n_{Ra}}{n_{Rc}} = \frac{1400}{10^7} = 0.14 \cdot 10^{-5}$
- $M2_{RBAC} = \frac{n_{\mathcal{R}}}{n_S} = \frac{5}{10^5} = 5 \cdot 10^{-5}$
- $M3_{RBAC} = 1 \cdot n_{cE} + 1 \cdot n_{aE} + 4 \cdot n_{RL} + 2 \cdot n_G + M_C(C1) = 105811$

We notice that the abstraction is well defined ($M_1 \to 0,\ M_2 \to 0$). We also notice that the complexity increases, since M_3 takes into consideration the number of concrete entities.

In the second case, the computed complexity metrics, give the following values for the RBAC policy:

- $n_{aE} = 5,\ n_S = 1000,\ n_A = 3,\ n_O = 200,\ n_{cE} = 1203,\ n_{Ra} = n_{RL} = 1400,\ n_{Ri} = 0,\ n_{Rc} \approx 10^5,\ n_H = 0, M_C(C1) = 3$
- $M1_{RBAC} = \frac{n_{Ra}}{n_{Rc}} = \frac{1400}{10^5} = 0.14 \cdot 10^{-5}$
- $M2_{RBAC} = \frac{n_{\mathcal{R}}}{n_S} = \frac{5}{1203} = 0.41 \cdot 10^{-2}$
- $M3_{RBAC} = 1 \cdot n_{cE} + 1 \cdot n_{aE} + 4 \cdot n_{RL} + 2 \cdot n_G + M_C(C1) = 6811$

We notice that the abstraction is well defined ($M_1 \to 0,\ M_2 \to 0$). We also notice that the complexity M_3 decreases compared to the first case, since there are less subjects.

5.3 Testbed based on SELinux Policies

To validate our approach, a proof-of-concept has been developed based on Lil CMS (Lil CMS — The Easiest Content Management System, 2014). Lil CMS is a simple content management system that allows users to remotely change the content of a Web server. The contents in Lil CMS are stored in plain text files. Lil CMS is easy to integrate with any traditional LAMP (Linux, Apache, MySQL and PHP) Web server. Our proof-of-concept, available at (Belhaouane et al., 2015), allows us computing the set of metrics described in this paper, upon our adapted version of Lil CMS (assumed to be administered via

SELinux policies on a Web server). Our proof-of-concept provides some representative SELinux (*Security Enhanced Linux* (McCarty, 2004; Mayer et al., 2006)) policies, described according to the different SELinux access control models, and with the purpose of putting in practice the metrics defined in Section 4.

The SELinux access control mechanism is primarily based on the TE (*type enforcement*) (Badger et al., 1995) model. TE associates each process with a domain and each non-process object with a type. Permissions are encoded as access vectors, which specify the operations that processes of a domain are authorized to perform on objects of a given type, such as files. Thus, an object's type can be considered as an abstraction element for objects and a domain as an abstraction element of related processes. The TE access control mechanism implemented in SELinux can be compared to an ABAC model but with the constraint that entities (e.g., processes and files) can have only one attribute (type). SELinux can also implement a second access control model, called SELinux *role-based access control* (RBAC). The SELinux RBAC feature is, in turn, built upon TE. Indeed, the access control in SELinux will be conducted via TE.

Using the aforementioned concepts on SELinux, we provide in our proof-of-concept some descriptive policy examples for our adapted version of Lil CMS — expected to be controlled by SELinux modules in a LAMP Web server. We evaluated and tested similar policies as those described in Section 5.1. Two main types of SELinux policies are defined. The first type uses only TE access control mechanisms to implement the CMS policies. The second type uses RBAC upon TE to implement the CMS policies. The aim of our proof-of-concept is the evaluation of such policies, in terms of their understanding complexity by SELinux administrators. At the same time, the series of policies defined in our proof-of-concept are also of value in order to evaluate other kinds of complexity measures (e.g., the computational complexity of the policy or the complexity of executing such policies by the Web server).

6 RELATED WORK

Several propositions in the complexity of access control policies have been presented. Research in this field includes computational complexity (Colantonio et al., 2010), size and structure of policies (Vaidya et al., 2010; Beckerle and Martucci, 2013), performance measure for evaluating the complexity of policy execution (Kateb et al., 2012), or comprehensive complexity by evaluating maintenance effort (Jaeger, 2001). The work in (Colantonio et al., 2010) studies the effort for the computation of an RBAC model by examining the complexity of role mining algorithms. The approach in (Beckerle and Martucci, 2013) highlights that complexity is a major problem in the manageability of access control policies. They confirm that the size of an access control policy impact the complexity. They proposed, for instance, the number of policy elements as a metric to evaluate the complexity. Furthermore, results in (Kateb et al., 2012) confirm that the increasing number of an access control model elements can cause performance issues and impacts the execution complexity. In contrast, the work in (Jaeger, 2001) defines the complexity of access control models as the complexity of understanding the model by administrators in order to create and maintain access control policies. Similar to Jaeger, we define the complexity of authorization-based access control models as the effort required to understand the model.

Our work is related to previous contributions from the area of complexity measurement for authorization-based access control models. This way, complexity measurement may help to optimize and refactor access control policies to simplify the policy administration process (Beckerle and Martucci, 2013; Kateb et al., 2012). The previous work only focuses on the size and structure of an access control model to evaluate the complexity. Our work complements this stream of research with a set of metrics that can be used in policy optimization process with respect to reasoning effort.

7 CONCLUSION

We introduced a series of metrics that evaluate the comprehensive complexity of understanding an access control policy. We started by describing a concrete example about the policy complexity concept. We then formalized basic concepts of authorization-based access control models, and presented our definition of policy complexity. Based on our definition, we provided a set of metrics to evaluate such a characteristic. The provided metrics were validated via a content management system (CMS) policy example. The approach has also been validated on a real environment using a general-purpose CMS controlled by SELinux (*Security Enhanced Linux*) policies. The results indicate that our metrics give a real estimation of the complexity of policies. Our metrics provide a tool for evaluating the complexity of understanding a policy by administrators. In addition, our metrics can be used in the optimization

processes for generating simpler and error-free access control policies. Future work aims at improving the proposed metrics and extend them for contextual access control policies. Another objective is to propose strategies for analyzing alternative policies using metrics.

Acknowledgements — Authors acknowledge support from the EC Framework Program, under the PANOPTESEC project (GA 610416), as well as Spanish Ministry of Science and Innovation (project TIN2011-27076-C03-02 CO-PRIVACY).

REFERENCES

Abou-El-Kalam, A., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miège, A., Saurel, C., and Trouessin, G. (2003). Organization Based Access Control. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003)*, pages 120–131. IEEE Computer Society.

Badger, L., Sterne, D. F., Sherman, D. L., and Walker, K. M. (1996). A domain and type enforcement UNIX prototype. *Computing Systems*, 9(1):47–83.

Badger, L., Sterne, D. F., Sherman, D. L., Walker, K. M., and Haghighat, S. A. (1995). Practical domain and type enforcement for UNIX. In *Security and Privacy, 1995. Proceedings., 1995 IEEE Symposium on*, pages 66–77. IEEE.

Beckerle, M. and Martucci, L. A. (2013). Formal Definitions for Usable Access Control Rule Sets From Goals to Metrics. In *Ninth Symposium on Usable Privacy and Security (SOUPS 2013)*, pages 1–11. ACM.

Belhaouane, M., Debar, H., and Garcia-Alfaro, J. (Last Access: 2015). Evaluating the Complexity of Access Control Policies Using Quantitative Metrics – SELinux Testbed Repository (Appendix). [On-line]. Available at `https://github.com/met-testbeds/selinux`.

Colantonio, A., Pietro, R. D., Ocello, A., and Verde, N. V. (2010). Taming role mining complexity in RBAC. *Computers & Security*, 29(5):548–564.

Cuppens, F., Cuppens-Boulahia, N., and Ben Ghorbel, M. (2007). High Level Conflict Management Strategies in Advanced Access Control Models. *Electronic Notes in Theoretical Computer Science*, 186:3–26.

Garcia-Alfaro, J., Boulahia-Cuppens, N., and Cuppens, F. (2008). Complete analysis of configuration rules to guarantee reliable network security policies. *Int. J. Inf. Sec.*, 7(2):103–122.

Garcia-Alfaro, J., Cuppens, F., and Cuppens-Boulahia, N. (2006). Analysis of policy anomalies on distributed network security setups. In *Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006, Proceedings*, pages 496–511.

Garcia-Alfaro, J., Cuppens, F., and Cuppens-Boulahia, N. (2007). Management of Exceptions on Access Control Policies. In *22nd IFIP TC-11 International Information Security Conference (IFIP SEC 2007)*, pages 97–108.

Garcia-Alfaro, J., Cuppens, F., Cuppens-Boulahia, N., Martínez Perez, S., and Cabot, J. (2013). Management of stateful firewall misconfiguration. *Computers & Security*, 39:64–85.

Harrison, M. A., Ruzzo, W. L., and Ullman, J. D. (1976). Protection in operating systems. *Commun. ACM*, 19(8):461–471.

Jaeger, T. (2001). Managing Access Control Complexity Using Metrics. In *Sixth ACM Symposium on Access Control Models and Technologies (SACMAT-01)*, pages 131–152.

Kateb, D. E., Mouelhi, T., Traon, Y. L., Hwang, J., and Xie, T. (2012). Refactoring access control policies for performance improvement. In *Third Joint WOSP/SIPEW International Conference on Performance Engineering, ICPE'12, Boston, MA, USA - April 22 - 25, 2012*, pages 323–334.

Lampson, B. W. (1969). Dynamic protection structures. In *AFIPS Fall Joint Computing Conference*, pages 27–38.

Lampson, B. W. (1974). Protection. *Operating Systems Review*, 8(1):18–24.

Lil CMS — The Easiest Content Management System (Last Access: 2014). Available at `http://www.lilcms.com/`.

Martin, E., Xie, T., and Yu, T. (2006). Defining and Measuring Policy Coverage in Testing Access Control Policies. In *2006 International Conference on Information and Communications Security (ICICS '06)*, pages 139–158. Springer.

Mayer, F., MacMillan, K., and Caplan, D. (2006). *SELinux by Example: Using Security Enhanced Linux (Prentice Hall Open Source Software Development Series)*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

McCarty, B. (2004). *SELinux: NSA's Open Source Security Enhanced Linux*. O'Reilly Media, Inc.

Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63:81–97.

Samarati, P. and De Capitani di Vimercati, S. (2000). Access control: Policies, models, and mechanisms. In *FOSAD*, pages 137–196.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, 29(2):38–47.

The Drupal project (Last Access: 2015). Available at `http://drupal.org/`.

Vaidya, J., Atluri, V., and Guo, Q. (2010). The role mining problem: A formal perspective. *ACM Trans. Inf. Syst. Secur.*, 13(3).

Wordpress Web Site (Last Access: 2015). Available at `http://wordpress.com/`.

Yuan, E. and Tong, J. (2005). Attributed Based Access Control (ABAC) for Web Services. In *2005 IEEE International Conference on Web Services*, pages 561–569.