Generation of Transmission Control Rules Compliant with Existing Access Control Policies

Yoann Bertrand(✉), Mireille Blay-Fornarino, Karima Boudaoud, and Michel Riveill

University of Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
{bertrand,blay,boudaoud,riveill}@i3s.unice.fr

Abstract. Access Control (AC) is a well known mechanism that allows access restriction to resources. Nevertheless, it does not provide notification when a resource is retransmitted to an unauthorized third party. To overcome this issue, one can use mechanisms such as Data Loss/Leak Prevention (DLP) or Transmission Control (TC). These mechanisms are based on policies that are defined by security experts. Unfortunately, these policies can contradict existing AC rules, leading to security leakage (i.e. a legitimate user is allowed to send a resource to someone who has no access rights in the AC).

In this article, we aim at creating TC policies that are compliant with existing AC policies. To do so, we use a mapping mechanism that generates TC rules directly from existing AC policies. Thanks to the generated rules, our solution can make inferences to improve existing AC and enhance security knowledge between infrastructures.

Keywords: Security · Access Control · Security policies · Transmission Control · Transmission security · Data Loss Prevention · Data Leak Prevention · Data leakage

1 Introduction

To add security to an infrastructure, one can start by controlling access to certain resources by using Access Control (AC) mechanisms. Unfortunately, traditional AC mechanisms are not useful to notify and manage what can happen to the resource once it is accessed. Indeed, a legitimate user can access then retransmit (legitimately or not) the resource to an unauthorized third party. If the third party does not have access to the resource in the AC, this retransmission can be seen as a violation of AC, and thus, as a data leakage. To tackle this problem, one can use Data Loss/Leak Prevention (DLP) or other Transmission Control mechanisms (TC). Such mechanisms are based on policies that aim at monitoring and notifying unauthorized resource transmission. DLP / TC are often used on top of AC, leading security experts to manage both paradigms. This double management can lead to data leakage.

© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2015
B. Thuraisingham et al. (Eds.): SecureComm 2015, LNICST 164, pp. 438–455, 2015.
DOI: 10.1007/978-3-319-28865-9_24

Let us take an example. Imagine an AC policy containing a rule mentioning that *"user Chris can access the resource docA.pdf"* and a TC rule saying that *"Chris can send all pdf files"*. If Chris accesses docA.pdf and wants to send it to Ana (who does not have access to docA.pdf in the AC), several remarks can be made. First of all, the fact that Chris can access docA.pdf does not violate the AC policy. Secondly, the fact that Chris sends docA.pdf to Ana does not violate the TC policy. Nevertheless, the transmission will cause a violation of the AC policy because Ana does not have access to docA.pdf.

This simple, but yet explicit example, shows that even if both AC and TC policies are correct, TC rules can violate existing AC policies and consequently lead to data leakage.

Our objective is to work with Transmission Control (TC) policies that do not contradict the existing Access Control (AC) policies.

To do so, we propose a mechanism that generates TC policies based on existing AC policies. Thanks to the generated TC policies, our solution offers mechanisms that :

- help improving existing AC policies (M1);
- help integration and enhancement of security knowledge between infrastructures or companies (M2).

The rest of the paper is organized as follows. Section 2 presents the main works related to access and transmission control. Section 3 describes the vocabulary we are using. Section 4 details our solution. Section 5 presents the results of our evaluation while section 6 concludes the paper and outlines future works.

2 Related Works

This section presents the main Access Control models and Data Loss/Leak Prevention (DLP) notions. The last part of the section presents existing solutions that aim at linking both AC and TC paradigms in common models or frameworks.

2.1 Access Control Models (AC)

Access Control (AC) encompasses sets of controls to restrict access to certain resources. Several contributions have been made to create efficient and fine-grained AC mechanisms. The following subsections present the main AC models.

Mandatory Access Control (MAC). MAC is a type of access control that secures resources by assigning sensitivity labels on resources and comparing these labels to the accreditation level a user is operating at. These levels are defined and controlled by the system, independently of user operations and choices. MAC is often used in confidential and military infrastructures. Famous models, such as Bell-LaPadula [1] or Biba [2] are based on MAC principles.

Discretionary Access Control (DAC). DAC allows users to determine and set the permissions over all the resources they own. The main DAC models are Access Control Lists (ACL) and Capability-based access control. Access Control Lists [3] represent resource rights as a table of subjects mapped to their individual rights over the resource. ACLs are data-oriented and provide a straightforward and rather simple way of granting or denying access.

Capability-based [4] access is more subject-oriented. A capability is an unforgeable token used to access a resource. It can be represented as a pair (x, r) where x is the name of a resource and r is a set of access rights. Thus, subject's capabilities are stored with the subject. Systems such as Plessey System 250 [5] are based on capabilities.

Role Based Access Control (RBAC). The paradigm behind RBAC [6] is based on the notion of role. A role is a set of users that share common attributes (for instance, a role "Network-Staff" containing all the network engineers of a company). In this model, users are members of one or several groups and this membership gives them access to certain resources.

Attributes Based Access Control (ABAC). NIST defines ABAC as "*An access control method where subjects requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions*" [7]. Attributes can represent various things about a subject (age, sex, etc.) or an object (resource security level, type, etc.). Thus, ABAC can be seen as an extension of RBAC.

Policy Based Access Control (PBAC). Policy Based Access Control [8] allows access rules to be defined and updated in a policy-oriented fashion. Policies are sets of rules that can be combined to determine if an access is authorized or not, depending on various attributes regarding the subject, object or environment. For these reasons, PBAC can be viewed as a standardization of ABAC for companies or other governance oriented structures.

Traditional AC models offer an easy way to restrict access to resources. Nevertheless, they do not tackle retransmission problems. To overcome this issue, solutions have been proposed. These solutions include Data Loss/Leak Prevention.

2.2 Data Loss/Leak Prevention (DLP)

This subsection presents the main notions of Data Loss/Leak Prevention[1].

[1] DLPs have been described in various terms, including Information Leak Detection and Prevention (ILDP), Information Leak Prevention (ILP) or Content Monitoring and Filtering (CMF). Nevertheless, DLP is the most commonly used name.

Definition and Classification. DLPs have been described as "*systems that monitors and enforce policies on fingerprinted data that are at rest (i.e. in storage), in-motion (i.e. across a network) or in-use (i.e. during an operation) on public or private computer/network.*" [9].

Policy Definition. DLPs are based on policies. These policies can help security experts defining fine-grained rules that help the DLP to detect and prevent leakage (for instance: *"deny the transmission if data x is sent to user U1"*). Industrial DLPs, such as the one provided by Symantec[2] or RSA[3], offer graphical user interfaces to generate these rules.

DLP can provide efficient TC mechanisms thanks to policies. As stated in the introduction, such policies can be in contradiction with an existing AC, leading to AC policy violations. To overcome this issue, one solution can be to combine both Access Control and Transmission Control in an unified paradigm and define both aspects at the same time. Such solutions are presented in the next subsections.

2.3 Unifying AC and TC

Several works have been proposed to unify AC and TC in common formalisms or frameworks. By doing so, a security expert can define at the same time both AC and TC policies, reducing the risk of contradiction. This subsection presents the main works in the domain.

Usage Control (UCON). UCON [10] has proposed to add the notion of ongoing usage to AC. Based on the notions of Authorizations, Obligations and Conditions, UCON offers a unified framework that covers traditional AC models and enhance them to tackle prerequisites within network-connected environment. UCON has been followed by many works, tackling policy definition [11], decentralized systems [12] or existing company mechanisms enforcement [13].

Organization Based Access Control (OrBAC). OrBAC defines a conceptual and industrial framework to meet the needs of information security. It covers a lot of issues such as conflict detection [14] or interoperability and deployment in companies Workflows [15]. In [16], OrBAC has been enhance to tackle information flow control problematic.

eXtensible Access Control Markup Language - Data Loss Prevention (XACML-DLP). XACML is a XML standard that defines a declarative AC. In October 2014, a new version of XACML has been implemented. This version,

[2] https://www.symantec.com/data-leak-prevention/
[3] http://www.emc.com/security/rsa-data-loss-prevention.htm?fromGlobalSelector

named XACML-DLP[4], embeds both Access Control and Transmission Control in a same formalism.

Linking AC and TC in a common formalism allows security experts to define at the same time both paradigms, reducing the risk of contradiction between them. Nevertheless, these solutions do not use the existing AC. Moreover, they cannot help enhancing the existing AC policies (M1) and ease the integration between companies thanks to inferences (M2). Following sections present a solution providing such mechanisms.

3 Context and Vocabulary

This section gives information about the concepts and vocabulary we are using. It first describes the scope of our study. Then, it presents a generic AC formalism and sets some working hypothesis regarding the existing AC policies.

3.1 Scope of the Study

In this paper, we have considered only 3 actions: Read, Read-Write, and no access. Also, we have decided to represent subjects as individuals instead of roles or groups. In term, we intend to broaden our solution to encompass more sophisticated actions and subjects representation.

Concerning the validation of the generated TC policies, a checking mechanism has been implemented to verify that generated TC rules are coherent with the existing AC policies. In this article, we focus on the generation process itself, for that reason, details about this validation mechanism are intentionally omitted.

3.2 Generic Access Control Model

To take into account the main AC models of the literature, we have been inspired by [17] and more specifically [18] and [19] to represent a generic AC as a set of rules (1). A rule is always composed of three fundamental things; Subject, Action and Resource (2).

$$GenericAC =< \sigma_1, \sigma_2, ..., \sigma_n >, \forall \sigma \in Rules \quad (1)$$

$$\sigma =< s, a, r >, s \in Subject, a \in Action, r \in Resource \quad (2)$$

Subjects, Actions and Resources are subsets of Entity (3). An entity can be formalized has a unique identifier (for instance a name) and a set of parameters (i.e. attributes) (4). A parameter can represent for instance a role (ex: role="manager") or an accreditation level (ex: "accreditationLevel = "3"). The main properties of the identifier is that it cannot be empty (5) and it must be unique (6).

[4] http://docs.oasis-open.org/xacml/xacml-3.0-dlp-nac/v1.0/csprd01/xacml-3.0-dlp-nac-v1.0-csprd01.html

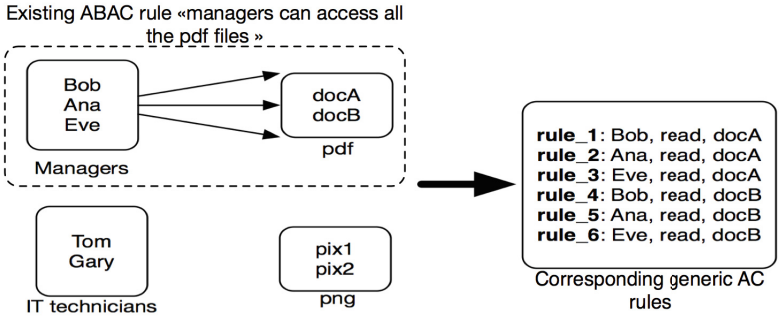


Fig. 1. Correspondences between the ABAC rule "Managers can read all pdf files" and the generic AC rules.

$$\{Subject, Action, Resource\} \subset Entity \tag{3}$$

$$entity = \{identifier, < p_1, p_2, ..., p_n >\} \tag{4}$$

$$\forall e \in Entity, e(identifier) \neq \emptyset \tag{5}$$

$$\forall e_i, e_j \in Entity, e_i(identifier) \neq e_j(identifier) \tag{6}$$

A parameter is a pair of key/value (7). Both key and value cannot be empty (8). For a particular entity (ex: subject "Bob"), two parameters cannot have the same key (9).

$$parameter =< key, value > \tag{7}$$

$$\forall p \in Parameter, p(key) \neq p(value) \neq \emptyset \tag{8}$$

$$\forall (p_i, p_j) \in P^2, p_i(key) \neq p_j(key) \tag{9}$$

Thanks to this formalism, we consider that traditional AC models can be represented. For instance, in the case of ABAC, a rule such as *"every Manager can access all the pdf files in read mode"* is equivalent to an enumeration of the rules (i.e. Cartesian product) among set "Manager" and set "pdf" (see Fig. 1.). We underline that such transformation can generate a huge amount of rules. Nevertheless, conducting tests in Section 5 show that our model is efficient for quite large sets of rules.

We make some hypothesis concerning the original AC. First of all, we consider that the original AC does not contain contradictory rules (for instance, a subject has both access and no access to a particular resource). Secondly, we consider that the correspondence between the original AC and the generic AC rules is not destructive, meaning that the semantic is conserved (no information is added, modified or removed). Finally, we consider that the corresponding generic AC can contain duplicate rules.

In this section, we have presented the generic model used by our solution. This generic model has been defined in order to take into account several AC models such as traditional ACL, RBAC or ABAC. Working hypothesis have

been made concerning the correspondence with generic AC rules. The following section describes our contribution in detail.

4 Contribution

In this section, we present our model. The first subsections describe our Transmission Control paradigm (4.1) and representation (4.2). Then, we present the generation mechanism that transforms Access Control policies into Transmission Control policies (4.3). The last subsections present a mechanism that notifies possible AC improvement (4.4) and propose an example to illustrate that our solution can ease integration and improve security knowledge between infrastructures (4.5).

4.1 Transmission Control List

Transmission Control model aims at answering the question: *"who can send what to whom?"*. To do so, we have defined a Transmission Control List (TCL), formalized as a set of transmissions regarding a specific resource (10). Thus, a specific TCL cannot describe transmission rights of more than one resource (11). A transmission embeds the following elements: a source subject (i.e. the sender), a destination subject (i.e. the receiver), the actions of the sender and the receiver and a transmission type (12). A transmission type represents if a transmission is authorized (TRANSMISSION_AUTH) or if the transmission is denied (TRANSMISSION_DEN).

$$\forall tcl \in TCL, tcl = \{resource, < \tau_1, ..., \tau_n >\} \tag{10}$$

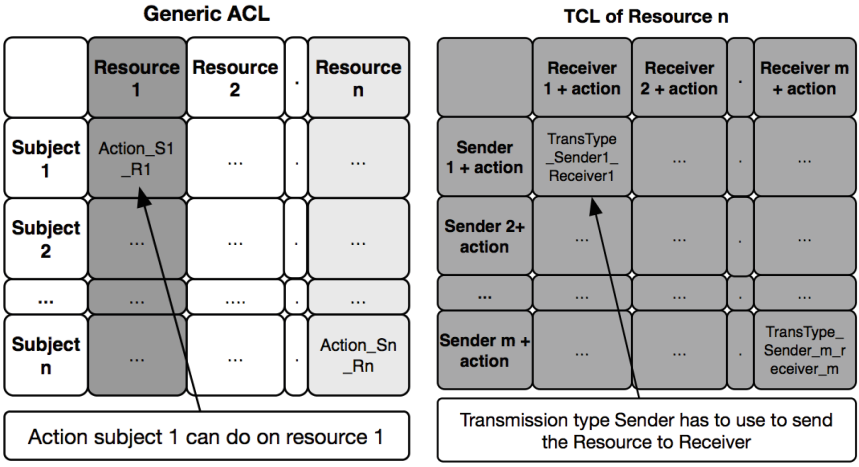
$$\forall tcl_1, tcl_2 \in TCL, tcl_1(resource) \neq tcl_2(resource) \tag{11}$$

$$\begin{aligned} \tau =< sender, receiver, senderAction, receiverAction, type >, \\ sender, receiver \in Subject, \\ senderAction, receiverAction \in Action, type \in TransmissionType \end{aligned} \tag{12}$$

4.2 Representation

For the sake of understanding, we represent ACL and TCL as matrices. ACL can be represented as a two-dimensional matrix, where columns represent resources, rows represent subjects, and intersections represent the action that the corresponding subject can perform on the corresponding resource (Fig. 2.a).

For a specific resource, the corresponding TCL can be represented as a two-dimensional matrix, where rows represent senders, columns represent receivers, and intersections represent the transmission type (ex: TRANSMISSION_AUTH) between the sender and the receiver (Fig. 2.b). We underline that actions of senders and receivers are also conserved in the TCL, due to the TCL formalism defined in 4.1. We also underline that the only subjects that are present in the



(2.a) (2.b)

Fig. 2. Graphical representation of a generic ACL (2.a) and a corresponding TCL (2.b)

corresponding TCL are the subjects with an explicit access right to this resource (we call such subjects "**marked subjects**"). Thus, the size of the TCL depends on the number of marked subjects. Indeed, a resource with many access rights in the ACL will generate a bigger TCL than a resource that can be accessed by fewer subjects.

4.3 Generation Mechanisms

After having described both ACL and TCL, we now discuss the generation mechanisms that aim at transforming existing ACL into TCLs. This subsection presents the main parts of the generation mechanism.

Creation of the TCL Structure. To create the general structure of the TCLs, the mechanism starts by retrieving all resources of the ACL. For each resource, the mechanism retrieves every marked subjects. Then, the mechanism creates the general structure of the matrices (one matrix per resource) by adding for each row and column the marked subjects as senders and receivers.

Mapping Rules Concept. Once the TCLs have been generated, they must be filled. A naive approach could be to fill every intersection with TRANSMISSION_AUTH, because every subject in a specific TCL is a marked subject and thus, has access to the resource in the original AC (see Fig. 3). Nevertheless, allowing every single transmission between marked subjects can be too permissive. In order to restrict transmissions in certain cases, we have defined Mapping

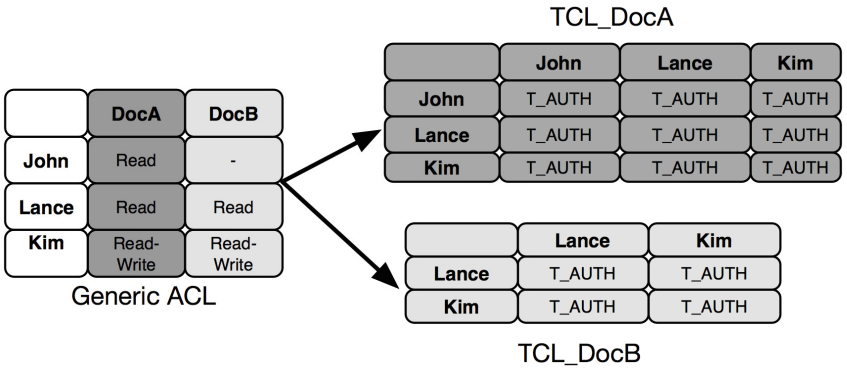


Fig. 3. Illustration of the creation of TCLs structure. For every resource in the ACL, a corresponding TCL is created. The size of the TCL depends on the number of marked subjects. Moreover, each marked subject is both sender and receiver.

Rules (MR). A MR can be represented as a function that takes parameters of a sender, actions, receiver and resource and returns a transmission type (13).

$$\begin{aligned} f(sender, senderAction, receiver, receiverAction, resource) \rightarrow type \\ type \in TransmissionType \end{aligned} \tag{13}$$

For each element of the matrix (i.e. each row/column intersection), the mechanism retrieves all the parameters concerning the sender, the receiver, the action of the sender, the action of the receiver and the resource, then output a transmission type.

To define how the transmission type is chosen depending on the entry parameters, we have defined a syntax. Details about this syntax are given below.

Mapping Rules Syntax. We have defined a syntax called Mapping Rules Syntax (MRS). MRS is based of three different things: targets, operators and inputs. A target can be formalized as an entity and an element (14). An entity can be a sender, a receiver, an action of the sender, an action of the receiver or a resource (15). An element can be an entity identifier (ex: "John"), a parameter key (ex: "role"), or a parameter value (ex: "manager")(16).

$$target = (entity, element) \tag{14}$$

$$entity = \{sender, receiver, senderAction, receiverAction, resource\} \tag{15}$$

$$element = \{identifier, parameter(key), parameter(value)\} \tag{16}$$

MRS uses two types of operators: arithmetic operators and logical operators (17):

$$\begin{aligned} arithmeticOperator : \{=, \neq, <, >, \geq, \leq\} \\ logicalOperator : \{\vee, \wedge\} \end{aligned} \tag{17}$$

Finally, the last component of MRS is the input, which is just a String (i.e. any word in the alphabet $\mathcal{A}$) (18).

$$input \in \mathcal{A}^* \tag{18}$$

Generic Rules. Thanks to previous definitions, generic rules can be defined and applied. A generic rule is defined by a target, an arithmetic operator, another target and a transmission type (19):

$$\begin{aligned} genericRule = targetA, arithmeticOperator, targetB \rightarrow type \\ targetA, targetB \in Target \\ type \in TransmissionType \end{aligned} \tag{19}$$

Generic rules can provide predefined and generic patterns to security experts. For instance, a generic rule such as "*you cannot send any resource to someone with an accreditation level lower than yours*" can be defined. Considering that subjects have a parameter "level" describing such accreditations, the previous generic rule will be:

rule1: (sender, level) > (receiver, level) → TRANSMISSION_DEN

With this formalism, we aim at providing general patterns that can be automatically applied to every row/column intersection. Such mechanism can then easily transform ACLs into TCLs.

Nevertheless, a security expert might want to define particular rules, adapted to her/his business or infrastructure. To do so, we have defined specific rules.

Specific Rules. Our model defines a specific rule as a target, an arithmetic operator and an input (20):

$$specificRule = target, arithmeticOperator, input \rightarrow type \tag{20}$$

Specific rules are used to define specific conditions on parameter values. A specific rule such as "*if the receiver does not have read and write permission, the transmission is denied*" will be defined by:

rule2: (receiverAction,identifier) ≠ "Read-write" → TRANSMISSION_DEN

To express even more complex rules, conditions and rules can be combined with logical operators. For instance, a set of conditions can be used to define the rule "*if John sends docA.pdf to a manager, the transmission is authorized*". This rule will be formalized as follows:

rule3:
(sender, identifier) = "John" ∧ (resource, identifier) = "docA.pdf" ∧
(receiver, identifier) = "manager" → TRANSMISSION_AUTH

Thanks to generic and specific rules, conditions on entities and parameters can be defined and applied. Generic rules provide a toolkit that can automatically be applied while specific rules formalism can be used by a security expert to express specific conditions, depending on her/his infrastructure and security concerns.

Confidential Transmission. Because TRANSMISSION_AUTH does not modify the medium, we empathize that sending a resource in cleartext is not secured and can be viewed as sending a resource to everyone. Thus, we have added another type of transmission, called TRANSMISSION_CONF, which allows a security expert to express confidentiality. This transmission type can be used with generic or specific rules. For instance, the rule "*resource 'docX.pdf' needs to be sent with confidentiality*" will be expressed as follows:

rule4: (resource, identifier) = "docX.pdf" → TRANSMISSION_CONF

Conflict Detection. Our model is able to express generic and specific rules. Nevertheless, definition and combinaison of these rules can lead to conflicts. Indeed, imagine for instance that a security expert defines and combines two different rules **r1** and **r2**, where **r1** defines *"When managers are sending a resource, the transmission must have confidentiality property"* and **r2** defines *"John cannot send docA.pdf"* (even if he has access to it in the ACL). Imagine now that John is a manager. The mapping mechanism will have issues deciding which transmission type to apply for every element in the row "John" for the TCL of docA.pdf.

Indeed, for this resource, the system will not be able to determine if the resource can be sent (**r1**) or not (**r2**). To overcome this issue, we have defined several mechanisms.

The first one is to notify the security expert of the inconsistency and ask her/him for an answer. She/he can chose the transmission type of her/his choice, or implement an ad hoc rule.

To avoid multiple notifications, another mechanism that we have defined is the decision strategies (DS). To use decision strategies, a security expert first needs to set levels for transmission type. For our example, we have considered that a denied transmission is more secure than the other types of transmission. Thus, we have chosen the following order:

Level_1: TRANSMISSION_AUTH < Level_2 : TRANSMISSION_CONF <
Level_3: TRANSMISSION_DEN

Once the levels have been defined, the security expert can use one of the following decision strategies:

- HIGHEST: apply the transmission type with the highest level
- LOWEST: apply the transmission type with the lowest level
- MOST PRESENT: apply the transmission type which is the most present in the sequence of rules
- DEFAULT: apply the default transmission type

In our example, the following rule will be applied automatically, depending on the strategy:

Strategy	**Applied rule**
HIGHEST	r2
LOWEST	r1
MOST PRESENT	cannot answer, DEFAULT is applied
DEFAULT	TRANSMISSION_DEN

To transform AC policies into TC policies, we have defined a specific syntax, called Mapping Rules Syntax (MRS). MRS can express generic and specific rules. Generic rules provide a toolkit that applies generic security policies while Specific rules can be use to define ad hoc rules. Thanks to this syntax, an ACL (which can be represented as a two-dimensional matrix) can be transformed into many TCLs matrices. Each TCL represents all the transmissions marked subjects can/cannot do for a specific resource.

4.4 Inference Mechanisms to Enhance Existing AC Policies (M1)

One of our objective is to provide a solution that is capable of improving an existing AC model. To do so, we use inference mechanisms. This subsection presents the main inferences that our solution is able to make.

Similarities Between Subjects. In the same TCL, if two couples of row/column are identical, it means that for a particular resource, two subjects have the same transmission behavior (i.e. they can send and receive the resource in the same way). If this reasoning is generalized for all TCLs, it means that these two subjects have exactly the same transmission rights for all the resources they have been marked for. Such inference mechanism is able to notify security experts that two or more subjects are similar in terms of transmission rights.

The model is also able to determine if these similar subjects have common parameters (such as "role" or "group"). If the existing ACL is based on RBAC or ABAC, the exact original classification is detected. However, if the original AC was a model without roles or attributes, notifications can help a security expert to have a better understanding of her/his ACL. With this knowledge, the security expert can decide to migrate her/his original ACL to a RBAC or ABAC model, using the notification to create roles or categories. Fig. 4. gives an example of the similarities between subjects.

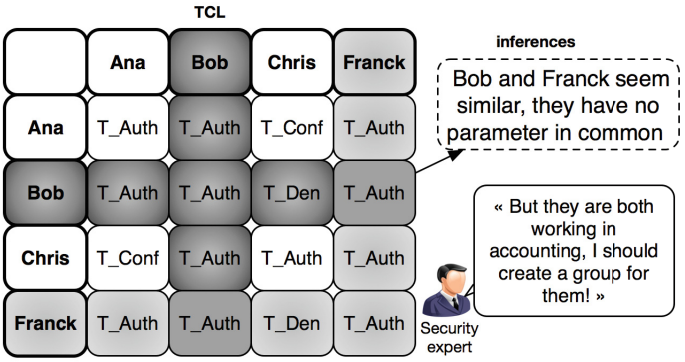


Fig. 4. Representation of the subject similarities mechanism for a single resource. In this case, security expert will be notified that Bob and Franck have the same behavior.

Similarities Between Resources. The same inferences can be applied to resources. In this case, if two TCLs are strictly equivalent, it means that for two different resources, the same subjects have strictly the same transmission behavior. The same assumption can be made for more than two resources. Thus, the same reasoning as subject similarities can be applied, meaning that the model is able to determine if similar resources (in term of transmission behavior) have properties in common (for instance, type="pdf"). Once again, these notifications can help security experts to have a better understanding of the original AC.

4.5 Inference Mechanisms to Help Integration and Enhance Security Knowledge (M2)

Inference mechanisms can also be used to detect security indulgences between two infrastructures regarding the same resources. Indeed, imagine that a company A wants to buy another company B. Both companies can be very different in terms of hierarchy, policies and sensibilities toward security. After the buy-out, security experts from company A might have some issues equalizing the two environments. Our solution can be interesting in such case.

Indeed, imagine that documentX.pdf is both used by company A and B. By applying our model in both companies, two different TCLs, TCL_A and TCL_B, will be generated. TCL_A (resp. TCL_B) will represent the transmission behavior of documentX.pdf inside company A (resp. company B). Inferences mechanisms presented previously cannot be applied, mainly because the two companies do not share the same subjects. Nevertheless, it is possible to compare the "tendencies" of the two TCLs. By tendencies, we mean general statistics such as the total number of subjects who have access to the resource (i.e. the size of the TCL) or the transmission types distribution (i.e. the percentage of each transmission type). To give an example, Fig. 5. represents TCL_A as a small matrix filled with confidential transmissions, while TCL_B is represented as a

very big TCL with a lot of non-confidential transmissions. Tendencies underline that there is a difference of security level regarding documentX.pdf. With such comparison, our model is able to notify security experts that company B has an indulgent security policy concerning documentX.pdf. Security experts can then modify mapping rules to generate a less permissive TCL_B, or tackle the problem at its root and modify the AC of company B in order to reduce the size of TCL_B.

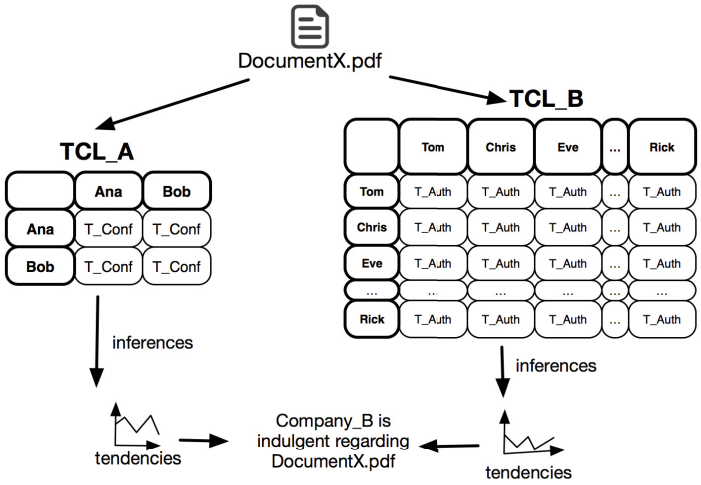


Fig. 5. Illustration of the tendency inferences, applied to a buyout example. Based on tendencies, notification can underline that company B is more indulgent regarding documentX.pdf security.

5 Evaluations

This section presents the results of several tests conducted in order to show that the proposed solution can be applied in real-life scenarios. For these tests, we aim to answer the following three questions:

- **Q1**: Are generation and inference mechanisms time-consuming?
- **Q2**: Is our solution suitable for small and medium-sized companies?
- **Q3**: Do specific ACL characteristics have any effects on the computational time?

5.1 Implementation

To generate ACLs, we have implemented an automatic rules generator. For the tests, several ACLs have been generated in order to simulate small and medium-sized company in term of subjects and resources. Information about these ACLs

are given in Table 1. The last column of this table (Ratio) is the proportion between the number of subjects and the number of resources (Ratio = NbResources/NbSubjects). A ratio greater than 1 is more likely to be found in a company. Indeed, the total amount of resources is often bigger than the total amount of subjects. We have based our sets on this assumption and have generated ACLs with different size and ratio. These ACLs try to simulate the amount of subjects and resources that can be found in small and middle-sized companies. We empathize, however, that these ACLs are purely speculative. Future works will focus on asking companies for insight about realistic volumetry and ratios (to that end, an online survey can be found here: http://goo.gl/forms/l0VIKDYBGt).

From a technical point of view, we have used a MacBook Pro Retina (Intel Core i7, 2,4 GHz, 16GB RAM, 256 GB SSD hard drive) and Java 7. Java Virtual Machine has been tweaked with a heap size of 4096 bytes.

We have used two mapping rules. The first one was "*you cannot send a resource to someone with a lower accreditation level than you*" while the second was: "*confidential resources need to be sent with confidentiality property*". Thus, we have created subjects with a parameter "accreditationLevel" and resources with a parameter "securityLevel". In order not to distort results with human interactions, we have used Decision Strategy "STRONGEST" with the security level described in 4.3. Thus, in case of conflict, the first rule was applied automatically.

5.2 Generation Tests

In these tests, we have measured the time-consumption of the process that allows our model to generate TCLs based on ACL. To do so, we have measured the time between the loading operation of an ACL and the end of the process (i.e. when all TCLs have been generated and saved as serializable objects in the hard drive). Results in Fig. 6.A show the generation process results. For very little set such as ACL1, it takes less than 1 second to compute. For sets ACL2 / ACL3 and ACL4 / ACL5, we can notice that the ratio slightly influences the computation. It can be explained by the fact that for ACL4, the maximum size of a TCL would be 1000 rows and columns (if everyone has access to the corresponding resource), whereas the maximum size of a TCL generated with ACL5 would

Table 1. Access Control Lists used for the tests.

ID	Rules	Subjects	Actions	Resources	Ratio
ACL1	50	10	3	40	4
ACL2	1500	250	3	1000	4
ACL3	1500	50	3	1250	25
ACL4	5000	1000	3	4000	4
ACL5	5000	200	3	4000	20
ACL6	10000	200	3	7000	35

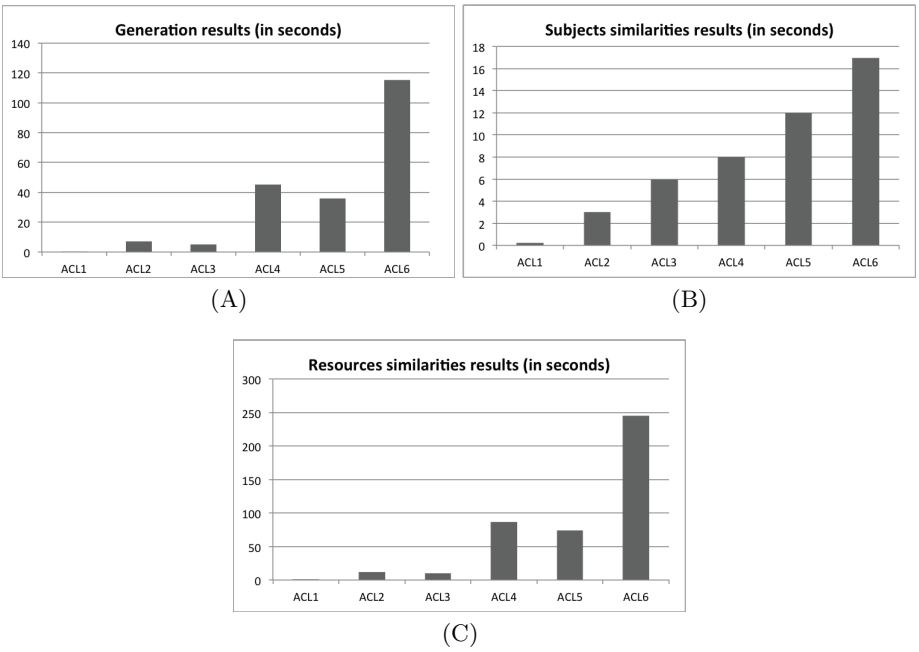


Fig. 6. Results for the generation process tests (A), subjects similarities tests (B) and resources similarities tests (C).

be "only" 200. Thus, these results show that it is algorithmically easier to do operations on a lot of smaller TCLs, rather than manipulating fewer, but bigger ones. Moreover, the saving process that consists of storing the generated TCLs is quite time-consuming, especially for big serialized objects.

Finally, ACL6 results show that our model can compute medium-size companies sets in less than 2 minutes. We consider these results has acceptable, especially for a process that needs to be done several times a day to be up-to-date.

5.3 Inferences Tests

We have tested the subjects and resources inferences mechanisms. Results in figure 6.B show that the subjects similarities computational time depends on the size of the ACL, with no significant impact regarding the ratio. Results in figure 6.C, however, show that ratio has a little impact for resource similarities. Indeed, even if ACL4 has 400 more resources, results shows that resources similarities process in ACL4 is faster than in ACL3. This results can be explained by the fact that once again, it is easier to compare many smaller Java objects.

Thanks to the conducted tests, questions **Q1**, **Q2** and **Q3** have been answered. Results show that the mechanisms involved in our model (i.e. generation and inferences) are quite fast, even with ACLs that embeds hundred

of subjects and thousands of resources (**Q1**). Despite the fact that these ACLs generate thousands of TCLs, results have shown that our model is scalable and can be used for small and medium-size companies volumetry (**Q2**). Finally, tests have shown that ratio between subjects and resources can have an impact on the processing time (**Q3**). Indeed, bigger ratio reduces the computational time for some mechanisms. Fortunately, this kind of ratio is more likely to be found in a real life scenario.

6 Conclusion

Over the years, Access Control (AC) mechanisms have been proposed to control access of resources. Unfortunately, traditional AC do not provide notification mechanisms when a resource is retransmitted to an unauthorized third party. To overcome this issue, Data Loss/Leak Prevention or other Transmission Control (TC) mechanisms can be implemented on top of AC. Nevertheless, TC policies can contradict existing AC policies, leading to potential data leaks. One solution can be to link both paradigms in a common formalism, allowing a security expert to define both policies at the same time. Nevertheless, proposed solutions do not always provide TC rules that are compliant with existing AC policies. Moreover, they do not offer notification mechanisms that can help enhancing the existing AC policies and facilitate the integration between companies and infrastructures. To cover these drawbacks, we have defined a new transformation mechanism that takes existing AC policies and generates TC policies. Thanks to the generated TC policies, two notification mechanisms have been implemented.

The first mechanism can help enhancing the existing AC (M1). This mechanism has been implemented thanks to resources and subjects similarities features. Such features can help a security expert to have a better understanding of her/his existing AC, by detecting resources and subjects with the same transmission behavior (Section 4.4). The second mechanism can be used to ease the integration between infrastructure and increase security knowledge (M2). Once again, we have used generated TCLs to infer tendencies that fulfill such purpose. To give an example, we have proposed a simple case study of a company buyout (Section 4.5). Finally, we have tested our solution with various ACLs. Results show that the model is interesting in terms of computational time for small and medium-sized companies.

In the future, we intend to modify our model to take into account more sophisticated entities by using capability-based security. Moreover, we will aim at reasoning with clusters of entities rather than single entities to simplify policy management and reduce the number of generated files. Secondly, we aim at creating realistic ACLs (in term of subjects and resources ratio) by asking companies about the volumetry of their ACLs. Finally, we would like to offer a formalized approach of the complexity of our mechanisms.

References

1. Bell, D.E., La Padula, L.J.: Secure computer systems: Mathematical foundations (No. MTR-2547-VOL-1). MITRE Corp., Bedford (1973)
2. Biba, K.J.: Integrity considerations for secure computer systems. No. MTR-3153-REV-1. MITRE Corp., Bedford (1977)
3. Saltzer, J.H., Schroeder, M.D.: The protection of information in computer systems. Proceedings of the IEEE **63**(9), 1278–1308 (1975). doi:10.1109/PROC.1975.9939
4. Levy, H.M.: Capability-Based Computer System. Butterworth-Heinemann, Newton (1984)
5. Fabry, R.S.: Capability-based addressing. Communications of the ACM **17**(7), 403–412 (1974)
6. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **2**, 38–47 (1996)
7. Hu, V.C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to attribute based access control (ABAC) definition and considerations. NIST Special Publication **800**, 162 (2014)
8. Han, W., Lei, C.: A survey on policy languages in network and security management. Computer Networks **56**(1), 477–489 (2012)
9. Shabtai, A., Elovici, Y., Rokach, L.: A survey of data leakage detection and prevention solutions. Springer Science & Business Media (2012)
10. Park, J., Sandhu, R.S.: The UCON ABC usage control model. ACM Transactions on Information and System Security (TISSEC) **7**(1), 128–174 (2004)
11. Hilty, M., Pretschner, A., Basin, D., Schaefer, C., Walter, T.: A policy language for distributed usage control. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 531–546. Springer, Heidelberg (2007)
12. Kelbert, F., Pretschner, A.: Decentralized distributed data usage control. In: Kiayias, A., Askoxylakis, I., Gritzalis, D. (eds.) CANS 2014. LNCS, vol. 8813, pp. 353–369. Springer, Heidelberg (2014)
13. Gheorghe, G., Mori, P., Crispo, B., Martinelli, F.: Enforcing UCON policies on the enterprise service bus. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6427, pp. 876–893. Springer, Heidelberg (2010)
14. Cuppens, F., Cuppens-Boulahia, N., Ghorbel, M.B.: High level conflict management strategies in advanced access control models. Electronic Notes in Theoretical Computer Science **186**, 3–26 (2007)
15. Ayed, S., Cuppens-Boulahia, N., Cuppens, F.: Deploying security policy in intra and inter workflow management systems. In: International Conference on Availability, Reliability and Security, ARES 2009, pp. 58–65. IEEE (2009)
16. Ayed, S., Cuppens-Boulahia, N., Cuppens, F.: An integrated model for access control and information flow requirements. In: Cervesato, I. (ed.) ASIAN 2007. LNCS, vol. 4846, pp. 111–125. Springer, Heidelberg (2007)
17. Barker, S.: Logical approaches to authorization policies. In: Artikis, A., Craven, R., Kesim Çiçekli, N., Sadighi, B., Stathis, K. (eds.) Sergot Festschrift 2012. LNCS, vol. 7360, pp. 349–373. Springer, Heidelberg (2012)
18. Slimani, N., Khambhammettu, H., Adi, K., Logrippo, L.: UACML: unified access control modeling language. In: 2011 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–8. IEEE (2011)
19. Khamadja, S., Adi, K., Logrippo, L.: An access control framework for hybrid policies. In: Proceedings of the 6th International Conference on Security of Information and Networks, pp. 282–286. ACM (2013)