Regulating Exceptions in Healthcare using Policy Spaces

C.A. Ardagna[1] S. De Capitani di Vimercati[1] T. Grandison[2]
S. Jajodia[3] P. Samarati[1]

[1] University of Milan, Italy
[2] IBM Almaden Research Center, USA
[3] George Mason University, USA

Abstract. One truth holds for the healthcare industry - nothing should interfere with the delivery of care. Given this fact, the access control mechanisms used in healthcare to regulate and restrict the disclosure of data are often bypassed. This *"break the glass"* phenomenon is an established pattern in healthcare organizations and, though quite useful and mandatory in emergency situations, it represents a serious system weakness.
In this paper, we propose an access control solution aimed at a better management of exceptions that occur in healthcare. Our solution is based on the definition of different policy spaces regulating access to patient data and used to balance the rigorous nature of traditional access control systems with the prioritization of care delivery.

1 Introduction

Healthcare systems support interactions among patients, medical practitioners, insurance companies, and pharmacies. The very sensitive nature of the information managed by these systems requires the balance between two contrasting needs: the need for data, to guarantee a proper delivery of care, and the need for keeping data secure, to properly protect the privacy of patients. Access control is the base mechanism that healthcare systems adopt for protecting medical data. Traditional access control models and policies are based on the assumption that the authorizations regulating access are known in advance. However, since in healthcare systems an important requirement is that "nothing interferes with the delivery of care" [12], access control restrictions may need to be bypassed in case of emergencies and care delivery, especially when there is a risk for the patient's health. For instance, in case of emergency, a nurse may require (and should be granted access to) data that during the "normal" working she cannot access. This phenomenon is usually referred to as "break the glass". While useful and mandatory in the name of

care delivery, such a situation may represent a weak point in the system that can easily be the target of abuses, for example, if breaking the glass becomes the norm [19].

The access control system should then be designed to be flexible and extensible, and should not be limited to a particular model or language (depending on the context, different solutions might be utilized). Also, the access control system should minimize the uncertainty by limit those cases in which no regulation applies and the break the glass principle is used. Finally, the access control system should protect the privacy of the patients, and should not allow exchange of identity data that violates government legislations, such as, the Health Insurance Portability and Accountability Act (HIPAA) [15] in the United States.

In this paper, we address the need for developing a flexible and powerful access control system for healthcare scenario by proposing an access control model that attempts to balance, on the one hand, the rigorous nature of traditional access control models and, on the other hand, the priority of care delivery in healthcare. We introduce the definition of *policy spaces* regulating access to medical data and we describe how policies are specified and enforced within each space and how their combination works. Our approach regulates the whole set of accesses, which would otherwise fall into a possible "break the glass" policy, and is aimed at a better treatment of "unusual" access requests.

The remainder of this paper is organized as follows. Section 2 introduces policy spaces and describes the properties that an access control system for healthcare should satisfy. Section 3 illustrates the considered scenario and how the policies in the different spaces are defined. Section 4 describes the policy evaluation process and illustrates a possible use case. Section 5 discusses related work. Finally, Section 6 presents our concluding remarks.

2 Exception-Aware Access Control Spaces for Healthcare

Traditional access control models and languages (see Figure 1(a)) are based on the definition of two spaces: *authorized accesses* ($\mathcal{P}^+$), regulating common practice requests; and *unplanned exceptions* ($\mathcal{E}^U$), regulating all requests that are not managed by $\mathcal{P}^+$. Since nothing should interfere with the delivery of care, space $\mathcal{P}^+$ may be bypassed especially when a patient's health is at risk. In such situations, an access request that falls into space $\mathcal{E}^U$ is authorized although the requester was not previously allowed to access what she requests, thus enforcing the break the glass

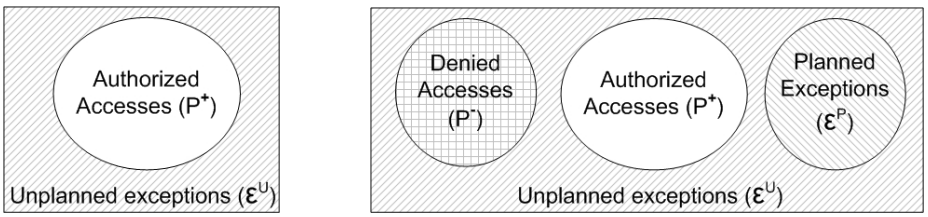

Fig. 1. Traditional access control (a); access control based on policy spaces (b)

principle. This makes the system vulnerable to malicious users that may exploit the break the glass principle for breaching the patient's privacy also when it is not strictly necessary.

To limit the possible damages caused by the break the glass, we put forward the idea of defining a solution based on the following extended set of *policy spaces* (see Figure 1(b)).

Authorized Accesses ($\mathcal{P}^+$). It corresponds to traditional access control policies. Intuitively, $\mathcal{P}^+$ includes the authorizations regulating *'common practice'*.

Denied Accesses ($\mathcal{P}^-$). It corresponds to access control policies that are used to prevent abuses. Denials are meant to be strictly enforced and do not allow any exception. They can be specified *a priori* to eliminate accesses that should never be authorized (i.e., accesses that cannot be bypassed by the break the glass) or inserted *a posteriori* because of observed abuses.

Planned Exceptions ($\mathcal{E}^P$). It corresponds to policies regulating access requests that do not fall into the normal routine as well as activities that should not be normally allowed. Policies in $\mathcal{E}^P$ are evaluated if and only if there are no applicable policies in $\mathcal{P}^+$ and $\mathcal{P}^-$, or applicable policies have no effect. This space regulates access requests that are managed as exceptions and can be foreseen, for example, according to past observations. Traditionally, space $\mathcal{E}^P$ is included in space $\mathcal{E}^U$.

Unplanned Exceptions ($\mathcal{E}^U$). It corresponds to policies regulating all access requests not covered by the previous policy spaces ($\mathcal{P}^+, \mathcal{P}^-$, and $\mathcal{E}^P$).

Accesses regulated by $\mathcal{E}^U$ are inserted into an auditing log for subsequent analysis and integration into the other spaces.

An important characteristic of these spaces is their *modularity*, since they are not limited to a particular access control model, language, or implementation. As a consequence, our solution allows the incorporation of policy languages that better suit the requirements of each particular situation. Furthermore, our spaces are fully compatible with traditional approaches and can be incrementally populated by analyzing accesses in $\mathcal{E}^U$ through an auditing process. In particular, the auditing process can show access requests that: *i)* correspond to common practice and should be explicitly permitted by appropriate policies in $\mathcal{P}^+$; *ii)* should be never admitted and should be explicitly denied by defining appropriate policies in $\mathcal{P}^-$; *iii)* are frequent but not common and should be captured by appropriate exceptions in $\mathcal{E}^P$.

As a result, the number of access requests granted by breaking the glass is considerably reduced and the probability of abuses decreases. Also, the modularity of our solution and the independence from the specific adopted language allow *backward compatibility* of the policy spaces definitions, meaning that our solution can immediately take the place of traditional access control models with limited effort.

3 Policy Spaces Language

We consider a scenario where *users* can connect to the system and make access requests of the form $\langle$*user_id*, *action*, *object*, *purposes*$\rangle$, where *user_id* is the unique identifier characterizing the requester, *action* is the action that is being requested, *object* is the object on which the requester wishes to perform the action and may correspond to the personal information of patients, and *purposes* is the purpose or a group thereof for which the object is requested. We assume that the personal information of patients is collected for the *purpose* of providing patient care. Also, in addition to the user id, each user is characterized by other properties (e.g., name, address) that are collected and stored into *profiles* associated with each user. A profile can therefore be seen as a container of pairs of the form $\langle$*attribute_name*,*attribute_value*$\rangle$, where *attribute_name* is the name of the attribute and *attribute_value* is its value. Such a user-related information is both *static* and *dynamic* in nature. Static information includes information that does not change or that does not change frequently (e.g., name, address, date of birth). Dynamic information includes context information that may depend on the specific user session. For instance, in

healthcare systems based on the role-based access control model [20], the roles activated during a user session are an example of dynamic information that is stored within the corresponding user's profile.

Medical data to be protected are referred to as *datasets*. Each dataset is characterized by a unique *object identifier* (object id, for short). Datasets can be organized in classes containing groups of datasets that can be collectively referred to with a given name and are associated with *metadata* that provide additional contextual information.

We finally assume that for each of the spaces introduced, the policy evaluation can result in three outcomes: *i) true,* positive evaluation; *ii) false,* negative evaluation; *iii) unknown,* no applicable policy is found. Policy evaluations are then combined to grant or deny the access (we will discuss the policy evaluation process in Section 4).

3.1 Policies for Spaces $\mathcal{P}^+$ and $\mathcal{P}^-$

Policies in $\mathcal{P}^+$ regulate normal accesses and correspond to positive authorizations managed by traditional access control systems. For instance, an authorization can state that a user can access the medical data of her patients when she activates the Doctor or Nurse role.

Policies in space $\mathcal{P}^-$ correspond to access control policies that are used to prevent abuses. These policies represent negative authorizations and identify accesses to be denied. Denials are strictly enforced and do not allow exceptions, meaning that exception policies in $\mathcal{E}^P$ and $\mathcal{E}^U$ are evaluated if and only if negative authorizations in $\mathcal{P}^-$ do not apply or their evaluation is 'false'. The main goal of $\mathcal{P}^-$ policies is to limit the number of access requests evaluated in the exception spaces $\mathcal{E}^P$ and $\mathcal{E}^U$. Negative authorizations are specified *a priori* for those unwanted accesses that can be foreseen at system setup, or inserted *a posteriori* because of abuses observed through the analysis of audit logs produced by accesses in $\mathcal{E}^U$.

Our approach is orthogonal to the specific policy adopted for regulating access and allows for incorporating any policy model and language in spaces $\mathcal{P}^+$ and $\mathcal{P}^-$ [10, 16].

3.2 Exception-based Policies for Space $\mathcal{E}^P$

Space $\mathcal{E}^P$ allows the definition of fine-grained policies used to regulate requests that cannot be considered 'normal routine' and that usually fall in $\mathcal{E}^U$ (e.g., a nurse on duty can access medical data of patients entering the trauma ward). Among them, we consider:

– *emergency requests,* which include all accesses necessary to preserve the health of patients;
– *on demand requests,* which include all accesses requiring an interaction with patients.

By taking a look at these types of access requests, we note that there are situations where the application of the break the glass principle is constrained by the fact that some conditions *must* be satisfied for an access to be granted and that, if at least one condition is not satisfied, the access should not be granted. For instance, suppose that even in case of emergency, access to medical data can be allowed *only* to the medical staff. It is then easy to see that such a restriction cannot be simply represented as a rule stating that the medical staff can be authorized. In fact, while the single rule presents the desired behavior, its combination with other rules specified for the same subject, object, and action may not, since typically an access request is permitted when there is at least a rule that allows the request, thus violating the *only* constraint. From what we said, it is clear that an approach of specifying exceptions as positive permissions for the access is not sufficient. Consequently, even if we do not make any assumption on the language for specifying policies $\mathcal{E}^P$, we propose a language inspired by the work in [4, 5] that supports the definition of two sets of rules: a set of *restrictions*, denoted $\mathcal{R}_E$, and a set of *authorizations*, denoted $\mathcal{A}_E$. Intuitively, restrictions are useful to specify requirements of the *only if* form stated above; while authorizations specify requirements in the traditional positive *if* form.

A *restriction* rule specifies requirements that are *necessary* (but not sufficient) to have the request satisfied and is defined as follow.

Definition 1 (Restriction rule). *A restriction rule has the form* ⟨*subject_id*⟩ [*WITH* ⟨*subject_expression*⟩] *CAN* ⟨*actions*⟩ *FOR* ⟨*purposes*⟩ *ON* ⟨*object_id*⟩ [*WITH* ⟨*object_expression*⟩] [*ONLYIF* ⟨*conditions*⟩] [*FOLLOW* ⟨*consequences*⟩]*, where:*

– subject_id *and* object_id *are the identifiers of a user and object, respectively;*
– subject_expression *is a boolean formula of terms that allows referring to a set of subjects depending on whether they satisfy given conditions that can be evaluated on the subject's profile;*
– actions *is the action (or class of actions) to which the restriction refers;*

– object expression *is a boolean formula of terms that allows referring to a set of objects depending on whether they satisfy given conditions that can be evaluated on the object's metadata;*
– conditions *is a boolean formula of conditions that every request to which the restriction applies must satisfy;*
– consequences (*or obligations [2, 8, 9]*) *is a boolean formula of actions that must be either performed after an access has been granted or that need to be performed in the future, based on the occurrence of well defined events (e.g., time-based or context-based events);*
– purposes *denotes the purposes for which the information can be collected and used. In other words, purposes represent the reason for which an access is requested. Abstractions can be defined within the domain of purposes to refer to purposes with common characteristics and to refer to a whole group with a name.*

Lack to satisfy any of the restriction rules that apply to a given request implies the request will be denied. By definition, these rules are equivalent to negative authorizations, where *conditions* are negated, and are ANDed. In this work, we rely on restriction rules for planned exceptions, since they are usually more intuitive and easy to define than negative authorizations. Also, restriction rules give a clear separation between the expressions (i.e., *subject_expression* and *object_expression*) that are evaluated to identify the applicable rules, and the necessary conditions that the request have to satisfy. As an example, notice the difference between rules like "users can read data only if they are Doctors and fill in a form" and "Doctors can read data only if they fill in a form". The first rule prohibits access to non-doctor since the state of being a doctor is defined in the only if part of the rule and then represents a necessary condition to gain the access; the second rule instead uses the condition to be a doctor only as a condition of the applicability of the rule.

An *authorization* rule specifies permission to be satisfied to have the access granted and is defined as follow.

Definition 2 (Authorization rule). *An authorization rule has the form:*
⟨*subject_id*⟩ [*WITH* ⟨*subject_expression*⟩] *CAN* ⟨*actions*⟩ *FOR* ⟨*purposes*⟩ *ON* ⟨*object_id*⟩ [*WITH* ⟨*object_expression*⟩] [*IF* ⟨*conditions*⟩] [*FOLLOW* ⟨*consequences*⟩], *where* subject_id, object_id, subject_expression, actions, object_expression, consequences, *and* purposes *have the same syntax and semantics as in restrictions, and* conditions *is a boolean expression of conditions whose satisfaction authorizes the access.*

Authorization rules are similar to positive authorizations managed by traditional access control systems. In case multiple authorization rules are applicable to a given request (i.e., rules where the conditions before the IF part are satisfied), they are all evaluated and the results are ORed. Unlike for restrictions, lack of satisfaction of a condition in an authorization simply makes the authorization inapplicable but it does not imply that the access will be denied. In particular, access can be authorized if there is at least one authorization that applies to it for which the conditions are satisfied.

Syntactically, subject expressions, object expressions, conditions, and consequences are always represented as boolean formulas of terms of the form `predicate_name(`*arguments*`)`, where *arguments* is a list, possible empty, of constants or attributes. Single attributes appearing in profiles of the users and metadata associated with objects are referenced via the usual dot notation. For instance, `Alice.Address` indicates that `Alice` is the user id (and therefore the identifier for the corresponding profile), and `Address` is the property. Also, to refer to the requester (i.e., the subject) and the target (i.e., the object) of the request being evaluated without the need of introducing variables in the language, we use keywords **user** and **object**, respectively, whose appearances in a conditional expression are intended to be substituted with actual request parameters during run-time evaluation of the access control policy. Keyword *any* is used to refer to any subject id and object id, and function `meta(`*object_id*`)` is used to refer to the metadata associated with *object_id*.

The conditions specified in the *conditions* element of restriction and authorization rules can be classified into two main categories: *static* conditions, which are similar to subject expressions and evaluate conditions on users' profile; *dynamic* conditions, which can be brought to satisfaction at run-time processing of the request. For instance, `fill_in_form(`*privacyform*`)` is a dynamic condition that is evaluated to true if the privacy form has been filled in by the requester. Table 1 illustrates an example of authorization and restriction rules in $\mathcal{E}^P$.

Evaluation of planned exceptions. Restrictions and authorizations are evaluated separately. The set of applicable restriction rules $\mathcal{R}_E$ is first calculated. The applicable restriction rules are those rules for which their *subject_id*, *actions*, *object_id*, and *purposes* elements include the *user_id*, *action*, *object*, and *purposes* specified in the access request, and the profile associated with the *user_id* and the metadata of *object* satisfy all conditions specified in *subject_expression* and *object_expression*, respectively. If

	Rule	Description
A1	*any* WITH equal(**user**.*role*,'*Nurse*') CAN *read* FOR *emergency* ON MedicalData WITH notequal(meta(**object**).*nurseId*, **user**.*id*) IF fill_in_form(*privacyform*)	A Nurse can read the Medical Data of patients not under her responsibility in case of emergencies after filling in a privacy form
A2	*any* WITH equal(**user**.*role*,'*Doctor*') CAN {*read*, *write*} FOR *emergency* ON MedicalData WITH notequal(meta(**object**).*doctorId*, **user**.*id*)	A Doctor can read or write the Medical Data of patients not under her responsibility in case of emergencies
A3	*any* WITH equal(**user**.*role*,'*PoliceMan*') CAN *read* FOR *investigation* ON MedicalData IF in(*time*, **user**.*startDuty*, **user**.*endDuty*) FOLLOW notify(meta(**object**).*dataOwner*)	A Police Man on duty can read the Medical Data of each patient in case of criminal investigation, notifying the data owner
R1	*any* WITH equal(**user**.*role*,'*Nurse*') CAN *read* FOR *emergency* ON MedicalData WITH notequal(meta(**object**).*nurseId*, **user**.*id*) ONLYIF in(*time*, **user**.*startDuty*, **user**.*endDuty*) FOLLOW {notify(meta(**object**).*dataOwner*), audit()}	A Nurse can read Medical Data of patients not under her responsibility in case of emergencies *only if* she is on duty, and notifying the data owner. Access must be audited
R2	*any* WITH equal(**user**.*role*,'*Doctor*') CAN {*read*, *write*} FOR *emergency* ON MedicalData WITH notequal(meta(**object**).*doctorId*, **user**.*id*) ONLYIF in(*time*, **user**.*startDuty*, **user**.*endDuty*)	A Doctor can read or write Medical Data of patients not under her responsibility in case of emergencies *only if* she is on duty
R3	*any* WITH equal(**user**.*affiliated*,'*yes*') CAN *read* FOR *emergency* ON MedicalData WITH notequal(meta(**object**).*doctorId*, **user**.*id*) ONLYIF notin(*time*, meta(**object**).*doctorId.startDuty*, meta(**object**).*doctorId.endDuty*) FOLLOW notify(meta(**object**).*doctorId*)	A user affiliated with the hospital can read Medical Data of patients not under her responsibility in case of emergencies *only if* the doctor responsible for the patient is not on duty and then notifying the doctor responsible for the patient

Table 1. An example of authorization and restriction rules in $\mathcal{E}^P$

at least one applicable restriction rule in $\mathcal{R}_E$ is such that the *conditions* element evaluates to false, the access is denied. Otherwise, the applicable authorization rules in $\mathcal{A}_E$ are selected. If at least one applicable authorization rule is such that *conditions* element evaluates to true, the access is granted; if $\mathcal{A}_E$ is empty or all the applicable rules evaluate to false, the access request is redirected to $\mathcal{E}^U$.

3.3 Exception-based Policies for Space $\mathcal{E}^U$

The space of unplanned exceptions $\mathcal{E}^U$ regulates those access requests that do not fall or cannot be evaluated in the spaces just introduced (i.e., $\mathcal{P}^+$, $\mathcal{P}^-$, and $\mathcal{E}^P$). Policies in $\mathcal{E}^U$ should be simple and must always grant access according to the break the glass principle since the promptness in reacting against exceptions is fundamental for preserving patients health. As a consequence, we adopt a solution different from the work in [17] where the concept of access request redirection is introduced to allow denied accesses in case of emergencies. In particular, space $\mathcal{E}^U$ regulates access by providing post-incident capabilities (i.e., auditing) to be used subsequently to better redistribute policies and requests among the spaces. The auditing process provides then logging facilities [3] that can be used a posteriori for overseeing the access requests in a given domain (e.g., a hospital department). To this aim, objects and classes of objects

are associated with supervisors that are responsible for their management. Such supervisors monitor all the access requests that fall in $\mathcal{E}^U$ and directed to a set of objects in a given domain. Also, based on log files, they can take countermeasures for misbehaving subjects or formalizes common behavior by defining additional policies in spaces $\mathcal{P}^-$, and $\mathcal{P}^+$ or $\mathcal{E}^P$, respectively. Cross-domain activities will be managed by the collaboration of different supervisors. This allows the mitigation of the risk of malicious supervisors and incorrect policy definitions.

To better clarify the concept, suppose that an employee of the hospital responsible for cleaning the surgical equipment reads the type of patient disease to prepare the suitable cleaning protocol (as each cleaning protocol is different for each infectious diseases). For each request submitted by this employee and allowed in $\mathcal{E}^U$, an auditing process must be performed and the access is logged. Since, the request is perfectly admittable and should be always allowed, a policy should be defined in $\mathcal{E}^P$ by the supervisor to regulate this scenario. By contrast, suppose that a malicious employee, in addition to the type of patient disease, also accesses the personal data of the patient to sell them to an insurance company. In this case, the supervisor is able to apply remedies, for example, initiating termination procedures and defining additional policies in $\mathcal{P}^-$ to avoid other future, similar unauthorized accesses.

4 Policy Evaluation and Enforcement

When an access request is received the policies in the different spaces are evaluated and enforced in sequence. Figure 2 shows the policy evaluation flow, where each policy space is modeled as a box that receives as input an access request and returns as output an evaluation response (i.e., true, false, or unknown). Based on the response the access request is granted, denied or forwarded to another policy space.

First, policies in $\mathcal{P}^+$, which govern *'normal access'* to medical data, are evaluated against the access request. If the evaluation result is 'true', the access is granted. Otherwise, if there are no applicable policies ('unknown' evaluation) or the policies have no effect ('false' evaluation), the request is redirected and evaluated in space $\mathcal{P}^-$. If the result of the evaluation of policies in $\mathcal{P}^-$ is 'true', the access is denied. Otherwise, if there are no applicable policies ('unknown' evaluation) or the policies have no effect ('false' evaluation), the evaluation proceeds by evaluating exceptions in space $\mathcal{E}^P$.

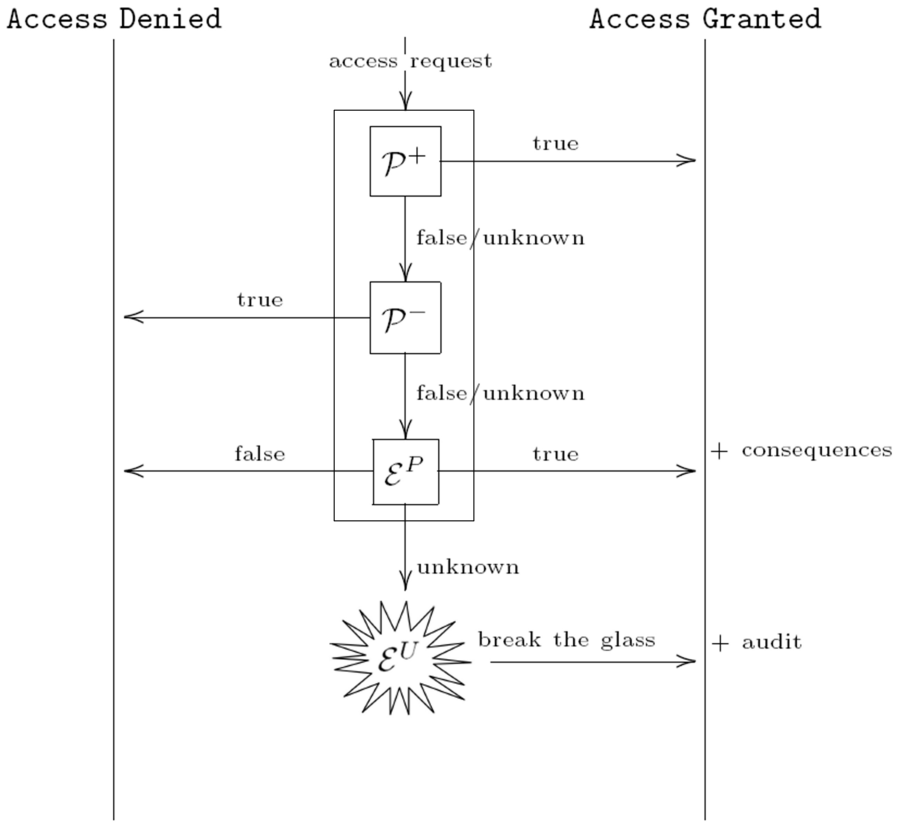

Fig. 2. Policy evaluation flow

Since policies in $\mathcal{E}^P$ are either authorization rules or restriction rules, the evaluation process in this space is more complex (see Section 3.2 for more details). If at least one applicable restriction in $\mathcal{R}_E$ is not satisfied, the evaluation result is 'false' and the request is denied. Otherwise, if all the applicable restrictions in $\mathcal{R}_E$ are satisfied (or $\mathcal{R}_E$ is empty) and at least one applicable authorization in $\mathcal{A}_E$ is satisfied, the evaluation result is 'true', the access request is granted and the possible involved *consequences* are enforced. If also the applicable authorizations are not satisfied (or $\mathcal{A}_E$ is empty) the request must be forwarded to space $\mathcal{E}^U$.

Here, the access request is inserted into a log file and is granted. The supervisor involved in the access request is then able to perform a subsequent analysis to possibly individuate abuses or access requests that

should be regulated by the definition of a proper set of rules in spaces $\mathcal{P}^+$, $\mathcal{P}^-$, or $\mathcal{E}^P$.

4.1 Use Case

We consider the case of Timothy, who is four years old, and currently being examined at Mount Cedar (MC) Hospital. Timothy was brought into MC's first aid clinic by his mother, Eva, late Wednesday evening. The admitting staff observed that Timothy suffered from several contusions all over his body, a fractured rib and a distorted shoulder. There are main stakeholders in this scenario:

- *the child* who might have been abused;
- *the child's mother* who brought the child to the hospital;
- *the child's doctor* who is responsible for providing care to the child;
- *the child's nurse* who is responsible for helping the doctor;
- *the child's social worker* who is responsible for helping the child address the trauma or abuse;
- *the police man* who is responsible for investigating the case and establishing possible criminal charges and responsibilities.

The policies that govern access at MC's computer system are defined in Table 1. Let's walk through the events that would occur in this all too common situation.

Initially, Timothy's doctor in the first aid clinic, Dr Murthy, takes a history, fills in the electronic patient record, assigns the patient to a care team, and orders a series of examinations. When the results return, Dr Murthy suspects child abuse and initiates the protocol defined for cases of *suspected child abuse*.

As a consequence, the police and social services are informed of Timothy's situation. Now, the police officer responsible for the criminal investigation, Lt. Starke, requires access to Timothy's medical information. As this kind of request is neither *normal practice* nor an *access abuse*, the evaluation of policies $\mathcal{P}^+$ and $\mathcal{P}^-$ results in 'false' and the request is redirected to space $\mathcal{E}^P$ where policies in Table 1 are evaluated. No restriction rule is selected (unknown evaluation) and authorization rule A3 is selected and evaluates to true. According to the discussion in Section 3, access is allowed and Lt. Starke informs MC of the access.

At the same time, the social worker responsible for helping abused kids reported by MC's staff, Miss Woodrow, requests access to Timothy's health record. As was the case with Lt. Starke, neither the $\mathcal{P}^+$ nor the

$\mathcal{P}^-$ policies evaluate to true and the request gets redirected to space $\mathcal{E}^P$. However, this time the $\mathcal{E}^P$ policy evaluation results in 'unknown', since no authorization and no restriction rules are applicable, which means the request is redirected to space $\mathcal{E}^U$. Here, access is granted to Miss Woodrow and because she has *broken the glass*, the supervisor is awakened and the audit process starts.

Since, Miss Woodrow's access is common in this case, the supervisor may define an additional policy in $\mathcal{E}^P$ to manage such a request in future and avoid further *break the glass* accesses.

Let's suppose that Timothy's health deteriorates and Dr Murthy is not on duty. If the current doctor on duty, Dr Wright, submits an access request to read Timothy's medical data, under emergency circumstances, then the request falls in $\mathcal{E}^P$ and the applicable rules are selected. If Dr Wright satisfies all applicable restriction rules (i.e., rules R2 and R3) and all applicable authorization rules (i.e., rule A2) in Table 1, she is given access to the data and Dr Murthy is informed of the access.

5 Related Work

A number of projects and research works about access control models and languages have been presented in the last few years [1, 7, 10, 16], although they do not address the issues and requirements that characterize the healthcare scenario. Jajodia et al. [16] define the *Flexible Authorization Framework* (FAF), a powerful framework that addresses many traditional access control policies as well as many protection requirements existing in real world applications. FAF allows one to specify accesses to be granted or denied in a declarative manner by defining expressive logic-based rules. The FAF language allows the definition of both positive and negative authorizations. The framework incorporates notions of authorization derivation, conflict resolution, and decision strategies, which rely on the hierarchical organization of objects, users, groups, and roles. Bonatti et al. [6] provide an algebra for combining security policies, which are defined as expressions of the algebra, with its formal semantics. The authors define a policy composition framework allowing the integration of different component policies while retaining their independence. The authors define a translation of algebra expressions into equivalent logic programs, which represent the basis for the implementation of the language.

In this work, we do not aim at providing new access control languages, but rather we are focused on formalizing a language-independent

model and infrastructure, which regulate the management of exceptions in healthcare. Below we describe some approaches for regulating exceptions. Gert [11] discusses how emergencies in a healthcare scenario are different from other situations. The author argues that same moral rules apply to all situations, although some actions are permissible in emergencies situations only. Reichert and Dadam [18] study the problem of applying workflow management systems to dynamic business processes. The authors present a formal foundation for the support of dynamic structural changes of running workflow instances and criteria to identify and handling the possible exceptions resulting from a workflow change. Han et al. [14] analyze the problem of managing medical workflow exceptions by giving an overview of past works in such a field and by proposing future research issues and solutions. The paper is focused on three main topics that would improve the quality of the medical procedures: representing, handling, and analyzing exceptions. Rostad and Odsberg [19] claim that exception mechanisms used to ensure access in critical situations increase the threats to patient privacy. Also, they provide an analysis of access log that results in a study of access control requirements for healthcare systems. In particular, using audit trails of access logs of Central Norway Health Region (CNHR), they aim at uncover information about the real user needs to provide a better access control mechanism for healthcare. Gupta et al. [13] provide a criticality-aware access control model for pervasive applications based on the calculation of a criticality level depending on critical events that happen in the system. Based on such a criticality level, they define two access control modes: a normal mode and a criticality-aware access control mode, which exploits a *promoterole* function improving the role privileges of the users to manage such critical cases. With respect to our solution, we do not provide different access control modes. Rather we provide a traditional positive authorization space together with several spaces managing access control exceptions. These spaces manage critical situations through the definition of policies supporting context-based conditions. Also our solution never changes users classifications or privileges for managing emergencies or critical events, but it defines policies to be considered in requests that do not satisfy traditional access control policies. Keppler et al. [17] discuss the problem of managing requests that are denied, by providing a range of other possible actions to use in emergencies situations. The framework extends the Flexible Authorization Framework (FAF) [16] with a sharing policy language for request and data redirection. Finally, Bhatti and Grandison [3] provide PRIMA, a system that improves privacy policies satisfaction in

healthcare scenario. The implemented solution is based on the concepts of policy coverage and refinement. In particular, the coverage between the ideal state of the system, that is, the policy defined by system administrator, privacy officer, and the like, and the real state of the system, that is, the policies logically tied to the audit logs, is calculated and then a refinement of policies is provided aimed at maximizing policy coverage itself.

6 Conclusions

We presented an exception-based access control solution whose main goal is to better control the break the glass attempts in healthcare systems to reduce possible breaches in the patients' privacy. We introduced the definition of policy spaces that balance the rigorous nature of traditional access control systems with the prioritization of care delivery. We illustrated how policies are specified and enforced within each space, and how these policy spaces are combined.

Acknowledgments

The research leading to these results has received funding from the European Union within the 6FP project PRIME under contract n^o IST-2002-507591, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^o 216483, and the Italian MIUR within PRIN 2006 under project n^o 2006099978.

References

1. C. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati. A privacy-aware access control system. *Journal of Computer Security (JCS)*, 2008. (to appear).
2. C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekera. Provisions and obligations in policy management and security applications. In *Proc. of the 28th Conference Very Large Data Bases (VLDB 2002)*, Hong Kong, China, August 2002.
3. R. Bhatti and T. Grandison. Towards improved privacy policy coverage in healthcare using policy refinement. In *Proc. of the 4th VLDB Workshop on Secure Data Management 2007*, Vienna, Austria, September 2007.
4. P. Bonatti, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. An access control system for data archives. In *Proc. of the 16th International Conference on Information Security*, Paris, France, June 2001.
5. P. Bonatti, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. A component-based architecture for secure data publication. In *Proc. of the 17th Annual Computer Security Applications Conference (ACSAC 2001)*, New Orleans, Louisiana, USA, December 2001.

6. P. Bonatti, S. De Capitani di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security*, 5(1):1–35, February 2002.
7. P. Bonatti and P. Samarati. A unified framework for regulating access and information release on the web. *Journal of Computer Security (JCS)*, 10(3):241–272, 2002.
8. M. Casassa Mont. Dealing with privacy obligations: Important aspects and technical approaches. In *Proc. of the 1st International Conference on Trust, Privacy and Security in Digital Business 2004 (TrustBus 2004)*, Zaragoza, Spain, August - September 2004.
9. M. Casassa Mont and F. Beato. On parametric obligation policies: Enabling privacy-aware information lifecycle management in enterprises. In *Proc. of the 8th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2007)*, Bologna, Italy, June 2007.
10. *eXtensible Access Control Markup Language (XACML) Version 2.0*, February 2005. `http://docs.oasis-open.org/xacml/2.0/access\_control-xacml-2.0-core-spec-os.pdf`.
11. H. Gert. How are emergencies different from other medical situations? *The Mount Sinai Journal OF Medicine - Issues in Medical Ethics Conference on "Special Challenges of Emergency Medicine"*, 72(4):216–220, July 2005.
12. T. Grandison and J. Davis. The impact of industry constraints on model-driven data disclosure controls. In *Proc. of the 1st International Workshop on Model-Based Trustworthy Health Information Systems (MOTHIS) 2007*, Nashville, Tennessee, USA, September 2007.
13. S. Gupta, T. Mukherjee, and K. Venkatasubramanian. Criticality aware access control model for pervasive applications. In *Proc. of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM 2006)*, Pisa, Italy, March 2006.
14. M. Han, T. Thiery, and X. Song. Managing exceptions in the medical workflow systems. In *Proc. of the 28th international conference on Software engineering (ICSE 2006)*, Shanghai, China, May 2006.
15. *Health Insurance Portability and Accountability Act*. `http://www.dhhs.gov/ocr/hipaa/`.
16. S. Jajodia, P. Samarati, M. Sapino, and V. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.
17. D. Keppler, V. Swarup, and S. Jajodia. Redirection policies for mission-based information sharing. In *Proc. of the ACM Symposium on Access control Models and Technologies (SACMAT 2006)*, Lake Tahoe, California, USA, June 2006.
18. M. Reichert and P. Dadam. Adeptflex-supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems (JIIS)*, 10(2):93–129, March/April 1998.
19. L. Rostad and O. Edsberg. A study of access control requirements for healthcare systems based on audit trails from access logs. In *Proc. of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference (ACSAC 2006)*, Miami Beach, Florida, USA, December 2006.
20. R. Sandhu, D. Ferraiolo, and D. Kuhn. The NIST model for role based access control: Towards a unified standard. In *Proc. of the 5th ACM Workshop on Role Based Access Control*, Berlin, Germany, July 2000.