Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 6 (2011) pp. 389–397

A Posteriori Access and Usage Control Policy in Healthcare Environment

Hanieh Azkia[1], Nora Cuppens-Boulahia[1], Frédéric Cuppens[1] and Gouenou Coatrieux[2]

[1]IT/Telecom Bretagne,
2, Rue de la Châtaigneraie, 35576 Cesson Sévigné, France
firstname.lastname@telecom-bretagne.eu

[2] IT/Telecom Bretagne
Technopôle Brest-Iroise, CS 83818, 29238 Brest, France
firstname.lastname@telecom-bretagne.eu

Abstract: **Traditional access control mechanisms prevent illegal access by controlling access right before executing an action; they belong to a class of *a priori* security solutions and, from this point of view, they have some limitations, like inflexibility in unanticipated circumstances. By contrast, *a posteriori* mechanisms enforce policies not by preventing unauthorized access, but rather by deterring it. Such access control needs evidence to prove violations. Evidence is derived from log records, which trace each user's actions. Efficiency of violation detection mostly depends on the compliance of log records with the access and usage control policy.**

In order to develop an efficient method for finding these violations, we propose restructuring log records according to a security policy model. We illustrate our methodology by applying it to the healthcare domain, taking care of the Integrating the Healthcare Enterprise (IHE) framework, particularly its basic security profile, ATNA (Audit Trail and Node Authentication). This profile defines log records established on the analysis of common health practice scenarios. We analyze and establish how ATNA log records can be refined in order to be integrated into an *a posteriori* access and usage control process, based on an expressive and contextual security policy like the OrBAC (Organization Based Access Control) policy.

Keywords: Access control model, IHE-ATNA, Audit.

I. Introduction

Controlling access to data is an important issue in information security. It must be considered in various domains, such as banking, healthcare, and so on while being applied to different systems client/server architectures within distributed or centralized environments. In order to define access control rules, the system administrator has to specify which data can be invoked, by which user and for which operation. By this way, one can regulate data confidentiality and privacy, as security services. Confidentiality and privacy are both important issues especially in collaborative environment and distributed system where multiple users and systems access and exchange data.

A lot of works have been devoted to security models, and a number of policy models have been proposed to express users' privileges, e.g. DAC (Discretionary Access Control) [1], MAC (Mandatory Access Control) [2], RBAC (Role Based Access Control) [3] or OrBAC (Organization Based Access Control) [4]. In these models, a user's request is checked immediately before granting or denying data access and usage. Thus, such models refer to *a priori* protection mechanisms, an *a priori* point of view which has some limits. Actually, deploying such traditional security models is restricting in contexts where users act in unforeseen and exceptional circumstances, such as emergency situations in hospital. So this kind of access control is not flexible and not appropriate in such environment. For example, it will deny access to unauthorized hospital staff while they have to handle a serious situation. At the same time, this approach creates a delay to verify authorization of each user's requests before granting access. So in some contexts, like emergencies, where time is vital, *a priori* access control is not always appropriate.

To solve these problems, *a posteriori* access control approaches have been proposed [5],[6] (see section IV) where policies are checked after granting access. This means that the user must account for his or her actions at a later time by providing proof that he or she was allowed to conduct these actions. This security check process performed afterwards appears more suitable in healthcare environments. For instance, regarding medical emergency situations, access will be given to each health professional who requests access. In other words, health professionals can continue their work, regardless of security policy rules. But if *a posteriori* access control detects an unauthorized action without justification, the perpetrator can be held accountable for such actions.

In order to deploy *a posteriori* access control, three components are required: (1) Log process which records the history of actions executed by the system's actors. Logged actions, referred to as logs, constitute evidence that can be used to demonstrate either an actor was allowed to perform a particular action or not. (2) Log analysis (auditing), which is triggered to identify abnormal actions through the verification of logs, led by a set of security rules. (3) Accountability component, which takes as input abnormal actions and detects

either that the misbehavior was authorized, i.e. the actor was allowed to carry out the action (depending on the context), or not. In the latter case, a violation occurred and the actor can be submitted to penalties. The effectiveness of this approach lies on how to identify misbehavior and violations.
In this paper, we focus on the log analysis component. This one contains three modules: (1) the "Security Policy" container includes two types of policy rules: *a priori* and *a posteriori* access and usage control rules; (2) the "Policy-Oriented Log Processor" reconciles security policy with logged actions, generated automatically by the system. This transformation makes our violation detection process easier and more efficient, and; (3) the "Decision Module" checks whether those Policy-Oriented Logs are consistent with the organizational security policy or not. By contrast with previously proposed frameworks, our approach is not intrusive, as we use traces generated by the existing log module of the target system. This log module can be compliant with *d*e facto standard like SYSLOG [7, 8] or with standard related to a specific domain like IHE-ATNA [9]. Moreover, the security policy used by the violation detection process is specified in accordance with an access and usage control model to provide low-level policies abstraction and interpretation ease by a human administrator.
In this study, we consider the IHE (Integrating the Healthcare Enterprise) framework [10] which is recommended for the integration of medical information system. This standard defines different profiles to which an information system can be compliant or not. The IHE-Audit Trail and Node Authentication (ATNA) is one of these integration profiles which ensure tractability in medical environment. It provides privacy and security rules and mechanisms, such as an audit trail. An audit trail allows monitoring activities related to security, patient privacy, user authentication and access and usage authorization in distributed applications. ATNA defines one structure for the contents of the audit trail, and this structure is used as log format in this work.
We also consider the OrBAC model to specify security policies. Our motivation to choose this model is that OrBAC is an expressive security model that makes it possible to specify a large number of security requirements, sometimes complex, which can be found in the medical domain. It uses native structural concepts (like "organization" and "context") and includes different security modalities (permission, prohibition, obligation and dispensation). This makes it more suitable to deal with new requirements in comparison with other models. The central notion in OrBAC is Organization. An organization can be seen as an entity that is responsible for managing security policy (e.g. hospital, cancerology department, firewall). Another interesting notion in this model is context. Contexts are used to define conditions to activate security rules (e.g. emergency, patient consent). They also increase or reduce user privileges depending on various events or states (e.g. temporal, spatial, provisional, etc.) and thus the obtained security policy is more dynamic.
The remainder of this paper is organized as follows. Section II defines and describes the audit profiles and the security model used in our framework. Section III presents the framework we suggest to enforce *a posteriori* access and usage control, in particular the transformation process of IHE-ATNA logs into a format compliant with an access and usage analysis led by an OrBAC policy. Section IV recalls related works. Finally, section V concludes the paper.

II. Audit profile and security model

A. IHE-ATNA

IHE is an initiative of healthcare professionals and industry, designed to stimulate the integration of information systems in the healthcare domain. Its objective is to ensure that all required patient information is both reliable and available to healthcare professionals. IHE is defined for some domain which has several profiles. These profiles describe the solution to a specific integration problem. To specify the technical details of the integration profile, IHE provides common technical frameworks. These frameworks identify a set of functional components of the healthcare institution called "IHE Actors", and specifies actor interactions referred to as "standards based transactions".
IHE Actors or system roles are responsible for acting on information in the context of an IHE Profile. Each Profile assigns specific requirements to specific actors. The same actor might be used in many profiles. IHE transaction is the interactions between actors that transfer the required information and belong to common working scenarios, i.e. when handling medical information system and data in daily medical practice. The IHE actors and transactions are abstractions of the real world healthcare information system environment.
ATNA is one of the integration profiles that have been defined for two domains, namely IT infrastructure and Radiology. While ATNA in radiology section establishes radiology specific audit trail messages and expresses the basic security measures to protect patient confidentiality, privacy and ensure traceability, in IT Infrastructure domain ATNA describes users and nodes authentication using certificates and the PHI (Protected Health Information) related audit events transmission to an audit record server. PHI is any information about demographic data that relates to the individual's past, present or future physical or mental health or condition, any part of a patient's medical record or payment history.
Actually, according to recent legislation on privacy and personal data security, it is essential to trace all actions that apply to subjects' personal data. We selected IHE ATNA as one solution to ensure traceability and security of both communications and data. It has some characteristics as below: (1) User Authentication, which requires that each user must log in before running an application in ATNA environment, after authentication, user's work can be traced over the network. (2) Node Authentication, which verifies that only authorized nodes, can have access to patient data; ATNA for this process provides a type of authentication based on mutual exchange of digital certificates. (3) Secure Communication, ATNA proposes that all nodes use encrypted tunnel between nodes. And finally, (4) Audit Record Generation, which keep track of all actions and communications that take place within a secure ATNA environment, this part requires an audit record server.
Here, our interest focuses on ATNA-Audit Record Generation, which provides the aforementioned evidence, i.e. the audit trail, for the security analysis process. Generation of

an audit trail relies on two main steps: 1) identifying events, based on transactions and 2) recording the audit messages. Each transaction performed by an actor could trigger specific events and thus cause creation of an audit record. These records will be stored for purposes of analysis, in an audit record server (referred to as an Audit Record Repository in ATNA), where they can be monitored in order to allow detection of abnormal and improper behavior.
ATNA specifies the use of Reliable Syslog [8] as the transportation mechanism for logging audit record messages to the central audit record server. It also makes it possible to use BSD (Berkeley Software Distribution) Syslog [7] knowing that it has several limitations. Thus, the audit record server will support both audit Reliable and BSD mechanisms. An audit record in ATNA is a record of actions performed by users on protected health information. This action can be creation, deletion, modification, query or view. An audit record is created by IHE actors when an IHE auditable event occurs. Auditable events are attached either to a technical context, or to patient data access. Most of these events have been identified considering the field of radiology.
IHE defines several audit record message formats and an IHE actor can use one or more formats, all of them use XML encoding and are defined by XML schema. There are two main types of format: 1) the IHE Audit Message Format, which is a combination of several standards (e.g. DICOM [11], IHE extensions) and which also extends the basic vocabulary provided by RFC-3881 [12] and 2) the IHE Provisional Audit Message Format, which is a provisional XML schema of audit record contents and is generated by the IHE radiology actors. The provisional format is less flexible than the IHE Audit Trail Format, which is the preferred format, but it is suitable for reporting in radiology and other diagnostic and treatment activities. It can also be converted into an equivalent IHE Audit Trail Format to reduce the burden on audit repositories.

B. OrBAC model

The purpose of a security model is to restrict an action on some objects to authorized subjects only, by defining security rules. OrBAC is one of these models. It is expressive enough to handle different and complex security requirements like those related to the healthcare domain. Specifically, this model is context aware, meaning that it allows us to define fine grained, dynamic and flexible security policies. Moreover, this model can potentially cooperate with other security models to allow interoperability as well as to allow natively some specific administration features like delegation. The OrBAC model (and its refinements like O2O [13] for interoperability and AdorBAC [14] for administration) is the chosen model for policy specification in our framework, since it meets our requirements.
As said before, the concept of organization is central in OrBAC. Instead of defining security rules that directly apply to subject, action and object, access control is defined at the "Organizational" level. For this purpose, subject, action and object are respectively abstracted into role, activity and view in the appropriate organization. Each organization can then define security rules that specify that some roles are permitted, prohibited or obliged to carry out particular activities on particular views. Security rules do not apply statically but dynamically, as they may depend on contextual conditions. For this purpose, the concept of context is explicitly introduced. OrBAC defines four predicates:

- *empower(org,s,r)*: in organization *org* subject *s* is empowered in role *r*.
- *consider(org,α,a)*: in organization *org* action α implements the activity *a*.
- *use(org,o,v)*: in organization *org* object *o* is used in view *v*.
- *hold(org,s,α,o,c)*: in organization *org*, context *c* is true between subject *s*, action α and object *o*.

Abstract security rules in OrBAC are modelled as a predicate with the five aforementioned parameters:
security-policy-type(org, role, activity, view, context) where security policy type belongs to {permission, prohibition, obligation, dispensation}
For instance, the organizational security rule: *permission(hospital CHU, Dr.Helia, consult, Alice_medical_record, emergency)* means that, in organization CHU, Dr. Helia is permitted to consult Alice's medical record in the context of emergency.
Concrete permissions, prohibitions, obligations or dispensations, that apply to triples $\langle subject, action, object\rangle$ are modeled using the predicate *concrete-security-policy-type(subject,action, object)* and logically derived from organizational security rules. The general derivation rule is defined as follows:
$$\begin{aligned}&\textit{security-policy-type}(Org, R, A, V, C) \wedge\\ &empower(Org, Subject, R) \wedge\\ &consider(Org, Action, A) \wedge\\ &use(Org, Object, V) \wedge\\ &hold(Org, Subject, Action, Object, C)\\ \rightarrow\quad &\textit{concrete-security-type}(Subject, Action, Object)\end{aligned}$$
where *concrete-security-type* belongs to $\{Is\text{-}permitted, Is\text{-}prohibited, Is\text{-}obliged, Is\text{-}dispensated\}$.

Typically, the parameters of these concrete rules correspond to a kind of information that we can find in a log records.
Other useful features are also defined in the OrBAC model, such as separation constraints and hierarchy of roles, views, activities and contexts (interested readers can refer to [4] for more details).
In the following section, our approach demonstrates how ATNA audit records can be reformulated or restructured making use of OrBAC predicates. The objective is to distinguish and extract OrBAC elements from ATNA event scenarios. Having a formal log structure based on a policy model facilitates checking the restructured log with policy rules, as well as improving the effectiveness of the analyzing process that detects non-compliant behavior (improper creation, access, modification and deletion of PHI).

III. Our proposal

The core idea of our approach is to structure the logs in order to bring them close to the security policy's relevant concepts, without modifying either the way the logs are generated or

their format. In this way, the violation detection process comes down to a mere verification. Thus, our *a posteriori* access and usage control framework (APAUC) requires three components: a log engine, an analyzing engine and an accountability/sanction module (see Figure 1).

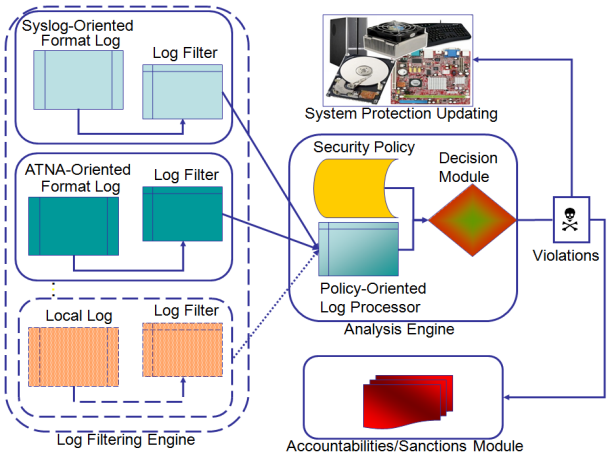

Figure. 1: APAUC Framework

This framework depicts the four steps of the APAUC process. The first step is conducted by the "Log Filtering Engine". The objective is first to obtain logs from various Information Systems (e.g. hospital's IS, analytical laboratory's IS) which collaborate, contain and exchange data that must be protected (e.g. patient record). Each of these systems may use special log format (syslog, ATNA, Common Logfile Format, etc.). For this reason, log record formats are not always similar. Moreover, they are independent of the security policies of the associated IS. In fact, elements that must be generated in the logs are configured by the administrator (e.g. time, date, IP, etc.). They are generally based on the administrator's expertise and on what appears useful to be logged for him.
Then, these logs are filtered to extract triples $\langle subject\_i, action\_i, object\_i \rangle$, their associated relevant events, states and metadata (IP of the host used by $subject\_i$ to perform $action\_i$, timestamp of $action\_i$, backup of the system performed before $subject\_i$ performs $action\_i$ on $object\_i$ if any, session number, and so on). The result of the log filtering must ease the construction of workflows or system's history related to some $subject\_i$, $action\_i$ or $object\_i$ from some specified point to another.
The second step consists in rewriting the filtered logs obtained during the first step in a format compatible with the policy model implemented in the analysis engine. This policy-aware reconciliation of log records facilitates violations detection. During this step, we perform an abstraction process of the logged elementary entities, starting with the triples $\langle subject\_i, action\_i, object\_i \rangle$. The ultimate goal is to meet the key security organizational concepts of the chosen access and usage control model. For instance, if we consider the case of the RBAC model with a given set of metadata like a session number, a date and a logged action of $subject\_i$, the process will retrieve all the roles that have been assigned to this subject in these circumstances. This is handled by the Policy-Oriented Log Processor.
In the third step, we trigger a compliance verification process of events, actions and transactions that occurred and that have been logged. For instance, for a logged action a_i performed by a subject s_i during some session, the process verifies that there exists an abstract security rule which authorizes some role R_i to do some activity Ac_i where s_i was assigned to R_i and a_i implements Ac_i. If it is not the case, APAUC raises a violation alert.
Finally, in the fourth step, the Accountability/Sanction module is used to qualify the raised violations, proving or discarding them, based on particular contexts. Norms like HIPAA (Health Insurance Portability and Accountability Act of 1996, issued by the U.S. Department HHS) can be used as input to assist this qualification task. If the violation is proved, responsibilities and appropriate penalties, if any, must be specified. When the violation has not been carried out by a user, and so discarded while an external problem has been raised (e.g. wrong medicine prescription), the security policy and the protection mechanisms used to deploy it must be checked (external attack, viruses, policy inconsistency).

A. *Policy-Aware Restructuring of IHE-ATNA Logs*

As our approach is applied to healthcare systems, we use IHE-ATNA profile for logging format. The reformatting procedure maps the relevant structures and contents of these standards to organizational security relevant concepts of the selected security model, OrBAC in our case. Here, we use The Radiology Audit Trail Option defended in [15] which details the required audit event for each IHE Radiology transaction and IHE Radiology actors. This option deals largely with the details of the Record Audit Event transaction in the IHE ITI Technical Framework. This later is used by the all IHE actors that support the Audit Trail and Node Authentication Integration Profile to create an audit record and to communicate with the Audit Record Repository actors.
Some trigger events defined in ATNA are not applicable to Radiology actors and transactions while they are applicable for the others. So we consider the common trigger events in our study. Here, We choose IHE ITI technical framework supplement 2004-2005 as reference. This framework describes the "IHE Provisional Audit Message format". It contains two structural elements: "auditable events" and "audit record elements" which correspond to trigger events, and audit record objects respectively. IHEYr4 is presented in this version as a complete audit record payload (see figure 2). IHEYr4 is a triple composed of the original host, the event occurrence time and a fixed number of auditable event types with related elements details. Some of these auditable events are not related to an IHE transaction and describe technical context (e.g Actor Start Stop) and the others are IHE transaction related and describe patient information (e.g. Patient Record Event).
In each transaction scenario there are some relevant actors. For each actor, we verify if it depends on some system's users. These actors pertain to one of the following categories: 1) User dependent, 2) User semi-dependent and 3) User independent. The reason of defining these categories is to detect and fix properly responsibilities within scenarios. We present them below:

- Actors depending on users' actions, like "Order placer" and "Order filler". These actors cannot act or interact without human intervention. Transactions related to this

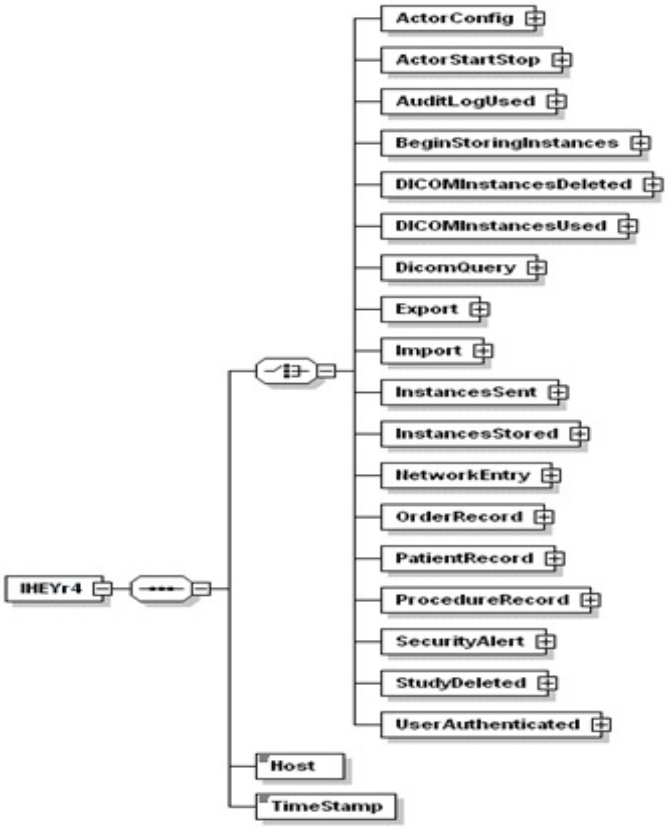

Figure. 2: IHEYr4 Audit Record

kind of actors must be validated by some system's user. For example, "Placer Order Management" transaction which is used by the "Order Placer" to place a new order or cancel an order with the "Order Filler", must be validated by an operator [16].

- Actors depending on the system users only at the beginning or the end of each transaction. An example of this category is "Acquisition Modality" actor. In the "Modality Images Stored" transaction, the "Acquisition Modality" sends the acquired images to the "Image Archive" automatically, but the start or shut down of the "Acquisition Modality" is triggered by a user[16].

- Actors which have not any explicit dependency on the system's users for performing a transaction. However, a user can intervene after the transaction execution, by sending a retrieve request for instance. Example of this category of actors is viewing a medical image on "Image Display". Images are sent from "Image Archive" to "Image Display" automatically. It is used to view image objects. But the user can also request and retrieve images from an "Image Archive" via "Image Display" [16].

Now, to restructure the audit records according to the policy model security concepts. We first have to consider a derivation and abstraction procedure for each auditable event type. Then, for each corresponding type of audit message, the procedure analyzes and extracts elements according to the OrBAC predicates (Organization, Role, Activities, Views and Contexts). In our general algorithm, we assume the "Message" as input data, the "Mapping function" as instruction and OrBAC parameters as output data, where:
Message: is a triple <Auditable Event, Host, TimeStamp >
Map: is a mapping function between auditable event and its security policy concepts extraction algorithm.
Let event be the Auditable Event of the given message and Extract-algorithm be the corresponding concrete security concepts extraction algorithm, then OrBAC elements are the outputs of the General algorithm (depicted in a pseudocode hereafter).

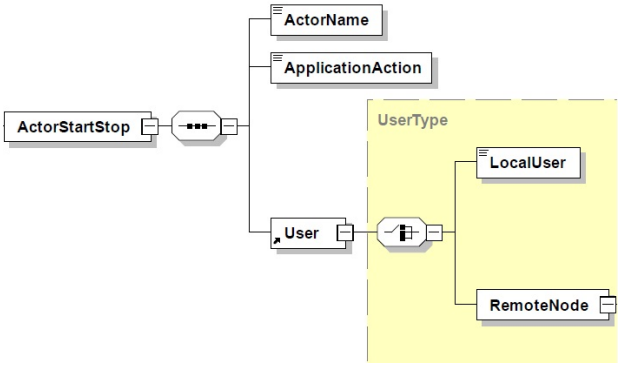

Figure. 3: ActorStartStop

Algorithm 1 General algorithm

Require: Map<Auditable Event, Extract-algorithm>
input: message (auditable event, host, timeStamp)
event: Auditable Event ← message.getEvent()
algo: Extract-algorithm ← Map.get(event)
output : OrBAC elements = algo(message)

For instance, let us consider the ActorStartStop audit record (see figure 3) which is the second event type in IHEYr4. The trigger event of this audit event is not transaction based because ActorStartStop is an application activity. This record is generated at the startup or shutdown of any application or actor and it has three elements:

- ActorName: is an arbitrary name which need be unique on the host. The combination of host and ActorName may be used to identify specific hardware, software, or usage.

- ApplicationAction: describes the running status of an application or actor for the event ActorStartStop. And its valid values are Start or Stop.

- User: identifies a user that performs activities which generate audit records. The user identification is either a username for LocalUser or, if a remote machine acts as a user, it is the RemoteNode identification.

As we see, all the elements needed for extracting OrBAC predicates explicitly appear in audit message. We can consider the "System User" as the subject of the scenario, "ApplicationAction" as the scenario's action on the object "ActorName". Temporal and spatial contexts related to this audit message can also be derived. Algorithm 2 below called algoACTSS extracts the OrBAC elements from the log message related to ActorStartStop event.

Algorithm 2 algo-ACTSS

Require: message (ActorStartStop, host, timestamp)
Subject ← value of ActorStartStop.User
Action ← value of ActorStartStop.ApplicationAction
Object ← value of ActorStartStop.ActorName
Context ← type of ActorStartStop.User(Local,Remote)
& value of timestamp
return (subject,action,object,context)

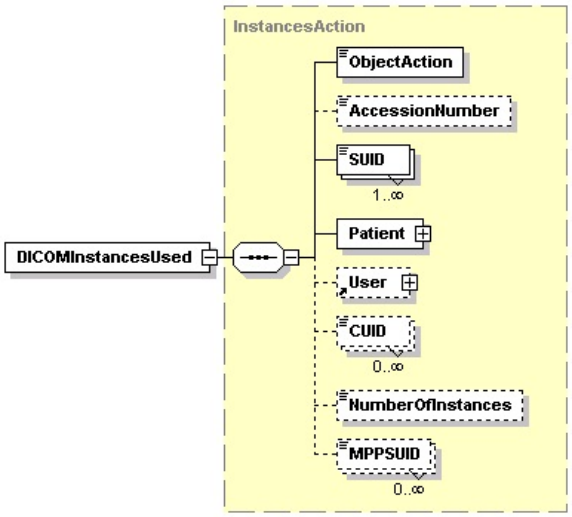

Figure. 4: DICOMInstancesUsed

As another example, let us consider the DICOMInstancesUsed event (see figure 4) which is the 6th event type in IHEYr4. This audit record is generated whenever locally stored information are created, modified, or accessed. This audit record contains eight elements which some of them are optional in audit message.

- ObjectAction: describes the action performed on an imaging object. The valid values are Create, Modify, Access and Delete.
- Accession Number: a string which presents a DICOM accession number.
- SUID (Study Unique IDentifier): is a single identification number which identifies a whole examination, in time and place. The used objects are always identified by its SUID.
- Patient: a text that describes a patient.
- User: identifies a user who performs activities that generate audit records.
- CUID (Class Unique IDentifier): identifies the type of service for which the image is intended.
- Number of instance: it should be unique inside the specific series (identified by the Series Instance UID).
- MPPSUID (Modality Performed Procedure Step Unique IDentifier): it is used for sending all study information such as acquisition information, patient details and image information, to the radiology server upon completion of the examination.

In this event, contrarily to the ActorStartStop event, extracting the OrBAC elements is not straightforward. It contains "ObjectAction" which corresponds to the action value,"Patient" which corresponds to object values. But to extract the subject value, the actor name must be fixed. That is because for each auditable event there is one corresponding trigger event, but each trigger event might be used in different transactions (as it is the case in the radiology workflow for instance)[17]. In this case, one way to help the subject value extraction process is to use the sequence number of the transaction which is mapped to the related actor. So, we consider a stack of transaction/actor value pairs for the concerned trigger event and use it as an input for the algorithm extracting the actor name. This algorithm is defined as below:

Algorithm 3 GetActorName

Require: Map<Auditable Event, Trigger Event>
input: auditable event
let t be a trigger event corresponding to auditable event on the given map
let St be a stack of transaction/actor pairs of t, then
x ←St.pop
output: x.actorname

According to the General algorithm and GetActorName function, the DICOMInstancesUsed OrBAC elements extraction algorithm, called algo-DIIUsed, is defined as below:

Algorithm 4 algo-DIIUsed

Require: message(DICOMInstancesUsed, host, timestamp)
Subject ← getactorname(auditable event)
Action ← value of DICOMInstancesUsed.ObjectAction
Object ← value of DICOMInstancesUsed.Patient
Context ←
typeOfDICOMInstancesUsed.User(Local,Remote)
& value of timestamp
return (subject,action,object,context)

Afterwards, once the extraction and mapping procedure is achieved, we trigger the abstraction process using, in that case, the logged administration actions, principally: role assignment, activity assignment and view assignment actions. Actually, the matter is to retrieve *empower(org,s,r)*, *consider(org,α,a)* and *use(org,o,v)* seen in section II-B. By doing so, we obtain an abstract version of the logs where, instead of a record log specifying, for instance, "Helia modifies PH010 on Wednesday 30th august 2009 ", we retrieve its abstraction version which specify; a doctor (his role) performs an updating (the activity implemented by the modifying action) of prescription (the view object PH010 belongs to) and the doctor was on call when performing the updating (which is the active context when the doctor performs the updating activity). The analysis process then has to check essentially, but not only, if there exists such a permission.

B. Analysis Process Activation

The analysis process to detect violations may be activated by a temporal or a non temporal event. If the activating event

is temporal, analysis is triggered every day, week, month, etc. The security administrator fixes such periodicity. In this case, the scope of investigation is wide and the analysis must consider all the logged events and actions since the last audit. This is closer to an analysis and risk assessment than an *a posteriori* access and usage control as we have in mind; the philosophy is different, though we can reach the objective.
Examples of non-temporal events include: complaint, medical error, medicine poisoning, deletion of a patient's record, and so on. In this case, the analysis investigates only the logs related to a specific subject, action or object concerned by the activating event. As the healthcare staff is composed of trusted subjects, the analysis is really performed either to give evidence that 1) the concerned subject has not violated the policy; or 2) he or she did violate the policy but there are reasons making this violation null and void, so he or she is accountable for his action but he or she is not sanctionable; or 3) he or she did violate the policy but there are no mitigating circumstances and in this case he or she is both accountable and sanctionable.

IV. Related Work

For a long time, and it is still mostly the case, computer systems used logs to restore databases and file systems to consistent states after crashes. They also used them for security purposes, and the auditing of these logs has been used to enforce security since 1980. Providing a policy language whereby an auditor may prove that an obligation has been fulfilled by analyzing a log of actions has been proposed by Corin *et al* [5] and Cederquist *et al*[6]. In these works, a policy language semantics, a logging mechanism and an audit procedure were defined. In [5], the authors went beyond suggesting a global framework, essentially developing the core notions of data and agent accountability. In [6], policy language was extended to allow using variables and quantifiers. This makes it possible to define a fundamental rule that gives the ability to refine policies. Using the same principle, Cederquist *et al*[18] presented a distributed framework that uses an *a posteriori* auditing approach. For auditing, they introduce three functions, namely observability, conclusions and proof obligations. They also provide implementation of the proof checker and a proof finder in the Twelf logical framework. The former allowing the auditor to check the justification given by agents while the latter allowing agents to justify an action they have performed to answer the auditor's request. Dekker *et al* [19] implemented *a posteriori* access control in the healthcare domain. They outlined the full architecture needed for audit-based access control of electronic healthcare record systems and discussed the advantages and limitations of this approach. Etalle *et al* [20] presented an approach, similar to that of Cederquist *et al* for *a posteriori* compliance based control. They combine logic and trust management language. This approach is less specific with regard to the expressive power of the policy rules, but it is more precise with regard to how the policies appear in the system, namely as sticky policies attached to the data items. Intrusion detection that focuses on anomalous behavior has also driven research in auditing and logging. More often it is mistaken for *a posteriori* access and usage control. Though the general objective is the same, the difference is essentially due to the following: 1) the environment where the later is deployed is trustworthy and generally reliable, and 2) the objective of the former is to stop attacks whereas the objective of the latter is rather to gather evidence and avoid sanctions. Our work is not intrusion detection-oriented, as our objective is not to stop ongoing attacks; we consider trustworthy environments where *a posteriori* control makes sense and where the notion of attack is almost asemantic. Contrary to the aforementioned works, 1) our main concern is to rely on a security policy model, flexible and expressive enough to allow us to manage different security requirements, like those encountered in the healthcare domain; 2) this model is also human-interpreted, so the detected violations and evidence are readable; and 3) our approach is not intrusive, in that it does not introduce any constraints regarding subjects concerned by the *a posteriori* control, so they do not have to log evidence themselves, and we do not impose any log format.

V. Conclusion

In this paper, we present a framework and processes to enforce and manage *a posteriori* access and usage control. Contrary to previous work that mainly focuses on security language, our security control process is based on a contextual security model having an appropriate level of abstraction. Thus, evidence of violation or not is human-interpreted. This is an important point, as performing *a posteriori* security control in a trustworthy environment targets essentially the organizational system, which will hopefully have a positive impact on the information system regarding security enforcement. The significance of this paper lies in its ability to converge logging data and policy structural concepts. This eases the detection violation process. We were careful not to introduce any constraints due to logs' format as on the manner they are generated in order to satisfy the principle of compliancy attached to the *a posteriori* access and usage control philosophy. The deployment of our solution is ongoing in the information system of the CHU of BREST, a French university hospital in Brittany.

Acknowledgments

This work presented in this paper is supported by a grant from The Britany Region, France, and by funding from SELKIS project.

References

[1] Department of Defense Trusted Computer System Evaluation Criteria, CSC-STD-011-83, Fort Meade, MD August 1983.

[2] D. Bell and L. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation, MITRE, Bedford, Mass, 1975.

[3] D. Ferraiolo and R. Kuhn. Role-Based Access Controls, 15th NIST-NCSC National Computer Security Conf, Baltimore, MD, 1992.

[4] F. Cuppens and N. Cuppens-Boulahia. Modeling contextual security policies, In International Journal of Information Security, 7(4): 285-305, 2008.

[5] R. Corin, S. Etalle, J. den Hartog, G. Lenzini and I. Staicu. A logic for auditing accountability in decentralized systems, Vol 173, Springer, pp. 187-202, Berlin 2004.

[6] J.G. Cederquist, R. Corin, M.A.C. Dekker, S. Etalle, J. den Hartog. An audit logic for accountability, IEEE Computer Society, pp. 34-43, 2005.

[7] C. Lonvick. The bsd syslog protocol, RFC 3164, August 2001.

[8] D. New and M. Rose. Reliable delivery for syslog, RFC 3195, November 2001.

[9] Integrating the Healthcare Enterprise, IHE IT Infrastructure Technical Framework Supplement 2004-2005 Audit Trail and Node Authentication Profile (ATNA), August 2004.

[10] Integrating the Healthcare Enterprise, IHE IT Infrastructure Technical Framework Volume I (ITI TF-1) Integration Profiles, August 2009.

[11] DICOM Standards Committee, Working Group 14, Digital Imaging and Communications in Medicine (DICOM) Supplement 95: Audit Trail Messages, Virginia USA, June 2004.

[12] G. Marshall. Security Audit and Access Accountability Message XML, RFC 3881, September 2004.

[13] F. Cuppens, N. Cuppens-Boulahia, C. Coma. O2O: Virtual Private Organizations to Manage Security Policy Interoperability, ICISS, pp. 101-115, 17-21 December 2006.

[14] F. Cuppens, A. Miège. Administration model for Or-BAC, Computer Systems Science and Engineering (CSSE04), 19(3), May 2004.

[15] Integrating the Healthcare Enterprise, IHE Radiology Technical Framework Volume I (RAD TF-1) Integration Profiles, June 2008.

[16] Integrating the Healthcare Enterprise, IHE Radiology Technical Framework Volume II (RAD TF-2) Integration Profiles, June 2008.

[17] Integrating the Healthcare Enterprise, IHE Radiology Technical Framework Volume III (RAD TF-3) Integration Profiles, June 2008.

[18] J.G. Cederquist, R. Corin, M.A.C. Dekker, S. Etalle, J. den Hartog. The audit logic-Policy Compliance in Distributed Systems, Technical Report TR-CTIT-06-33, 2006.

[19] M.A.C Dekker and S. Etalle. Audit-based access control for electronic health records, Electronic Notes in Theoretical Computer Science, 168:221-236, 2007.

[20] S. Etalle, and W. H. Winsborough. A posteriori compliance control, Proceedings of the 12th ACM symposium on Access control models and technologies, pp. 11-20, New York, USA, 2007.

[21] TL. Tsai, ML. Pan and DM. Liou. Implementation of an IHE ATNA-Based Electronic Health Record System, Institute of Biomedical Informatics, National Yang-Ming University.

Author Biographies

Hanieh Azkia is currently Phd student in information system security at TELECOM Bretagne LUSSI department since November 2009. She received an engineering degree in software engineering and master degrees in information system security. Her research domain is the access control a posteriori. She prepares her thesis under supervision of Prof. Frédéric Cuppens and Prof. Nora Cuppens.

Nora Cuppens-Boulahia is a teacher/researcher at the TELECOM Bretagne LUSSI department. She holds an engineering degree in computer science and a PhD from SupAero and an HDR from University Rennes 1. Her research interest includes formalization of security properties and policies, cryptographic protocol analysis, formal validation of security properties and thread and reaction risk assessment. She has published more than 60 technical papers in refereed journals and conference proceedings. She has been member of several international program committees in information security system domain and the Programme Committee Chair of Setop 2008, Setop2009, SAR-SSI 2008, CRiSIS 2010 and the co-general chair of ESORICS 2009. She is the French representative of IFIP TC11 "Information Security" and she is co-responsible of the information system security axis of SEE.

Frédéric Cuppens is a full professor at the Telecom Bretagne LUSSI department. He holds an engineering degree in computer science, a PhD and an HDR. He has been working for more 20 years on various topics of computer security including definition of formal models of security policies, access control to network and information systems, intrusion detection, reaction and counter-measures, and formal techniques to refine security policies and prove security properties. He has published more than 150 technical papers in refereed journals and conference proceedings. He served on several conference program committees and was the Programme Committee Chair of ESORICS 2000, IFIP SEC 2004, of SARSSI 2006 and general chair of ESORICS 2009.

Gouenou Coatrieux received the PhD degree in signal processing and telecommunication from the University of Rennes I, Rennes, France, in collaboration with the Ecole Nationale Suprieure des Tlcommunications, Paris, France, in 2002. His PhD focused on watermarking in medical imaging. He is currently an Associate Professor in the Information and Image Processing Department, Institut TELECOMTELECOM Bretagne, Brest, France, and his research is

conducted in the LaTIM Laboratory, INSERM U650, Brest. His primary research interests concern medical information system security, watermarking, electronic patient records, and healthcare knowledge management.