Ontology Based Log Content Extraction Engine for a posteriori Security Control

Hanieh Azkia, Nora Cuppens-Boulahia, Frédéric Cuppens, Gouenou Coatrieux

Ontology Based Log Content Extraction Engine for *a posteriori* Security Control

Hanieh AZKIA[a,1], Nora CUPPENS-BOULAHIA[a], Frédéric CUPPENS[a], Gouenou COATRIEUX
[a]*IT/Telecom Bretagne, 2 Rue de la Châtaigneraie 35576 Cesson Sévigné, France*

Abstract. In *a posteriori* access control, users are accountable for actions they performed and must provide evidence, when required by some legal authorities for instance, to prove that these actions were legitimate. Generally, log files contain the needed data to achieve this goal. This logged data can be recorded in several formats; we consider here IHE-ATNA (Integrating the healthcare enterprise-Audit Trail and Node Authentication) as log format. The difficulty lies in extracting useful information regardless of the log format. *A posteriori* access control frameworks often include a log filtering engine that provides this extraction function. In this paper we define and enforce this function by building an IHE-ATNA based ontology model, which we query using SPARQL, and show how the *a posteriori* security controls are made effective and easier based on this function.

Keywords. Access Control, Log, IHE-ATNA, Ontology, SPARQL

Introduction

In information systems like Healthcare systems, where environment is supposed to be trustworthy, policy makers prefer to specify minimum security checks during the operations. Security controls are made later. This kind of control is called *a posteriori* security control. It lets users continue their work, without blocking them due to some security policy rules. However, policies are checked after granting access by a security controller. According to *a posteriori* access control approaches [1, 2], this kind of control requires three components: (1) a log process records the history of actions that have been executed, (2) a log analysis identifies abnormal actions through the verification of logs, and (3) an accountability procedure takes as input abnormal actions and verifies whether the dysfunction was authorized or not. In this paper, we focus on the log process and particularly on the log content extraction engine. The log process takes as input a log format, here IHE-ATNA, and rewrites it in a common knowledge representation format. This transformation makes easier the investigation of different logs independently of their original log formats. The log content extraction engine, using a query language, takes the logged data in this rewritten format and generates as output some useful data. The analysis process, based on these log extraction results, determines whether the logged actions are consistent with the security policy.

In our study we focus on healthcare systems and use IHE-ATNA (Integrating the Health care Enterprise [3] - Audit Trail and Node authentication [4]), a log structure

[1] Corresponding Author. Hanieh Azkia is funded by a grant from the Britany region and the ANR SELKIS project, email : hanieh.azkia@telecom-bretagne.eu

standard in medical information systems. The approach is based on building ontology of logs. This avoids dealing with specific (logical or physical) information system design. So the logged data can be easily compared to the attributes of the security policy by exporting, translating and querying them and thus checking if the logged data are compliant with the security policy. In this paper, for the sake of clarity, the security policy is represented by a set of quadruples <subject; action; object; interval> which represent a subject is permitted to execute an action on an object on a given time interval. Since the security policy is checked using an *a posteriori* process, the time interval refers to time when the permission was active.

The reminder of this paper is organized as follows. Section 1 explains how we build ontology of logs. Section 2 presents an ontology example based on the IHE-ATNA standard. Section 3 presents the log engine enforcement and extraction processes, and shows how to query logged data for security investigation purpose. Finally, section 4 concludes and presents future works.

1. Security and logs

The core idea of our approach is to perform extraction and filtering of data that can be mapped onto or abstracted into security policy concepts and attributes. As the log structure and policy structure are generally different, we need to reconcile the log and policy concepts. The kind of data must be generated in the logs are configured either by an administrator or governed by some norms and standards. So to define our ontological representation of logs, we need the following necessary inputs:

- *Terminology used in the target domain*: most of the ontology classes are derived from this terminology to represent explicitly and assign consistent meaning to the target domain concepts. In a medical context, such significant concepts are for instance: Physician, Patients Records, etc.
- *Log structure of the target system*: this second source of concepts enriches the ontology classes and relations. It might be based on standards, standards *de facto* or simply built-in structures, like the structure of log provided by IHE-ATNA. Each ATNA log record is a triple: (1) the original "Host", a specific hardware, software where the action occurred, (2) A "TimeStamp", which determines the time when an action in an auditable scenario occurred, and (3) a fixed number of "Auditable events" with parameters (e.g. PatientRecord event which generates logs whenever a patient record is created, modified or accessed.). From these three elements, we derive the main classes, object properties and data properties to be added to the log ontology.
- *Security policy model*: we draw on the language and concepts on which this model rests. Existing security models like RBAC [5] or OrBAC [6, 7] have their own concepts which are not similar to those of ATNA logs. For instance, the RBAC model is based on the concept of role whereas OrBAC introduces other concepts like organization and context. However, all these models have in common that they provide means to derive if a given subject is permitted to execute a given action on a given object at a given time. For the sake of clarity and conciseness, we use in this paper a less expressive security model than RBAC or OrBAC, close to the discretionary model where we cover the concepts of subject, action and object plus a temporal context.

2. Ontology example

Using the three aforementioned inputs, figure 1 shows a part of the log ontology related to one of the most important ATNA Auditable event named, “BeginStoringInstances”. This log record is generated whenever a system begins to transfer a set of medical studies from one node to another node.

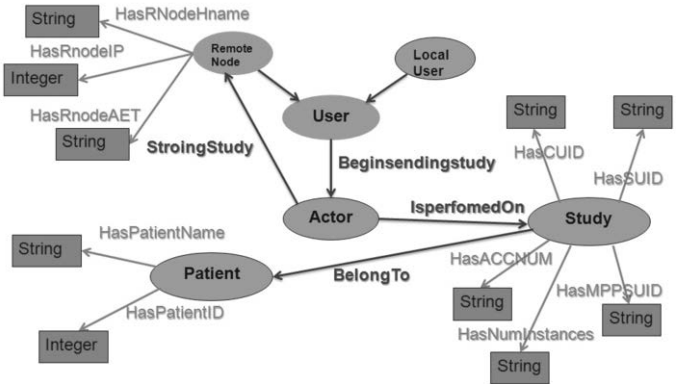

Figure 1. An excerpt of the whole ontology: “BeginStoringInstances”ontology

Once the log ontology is built, it can be queried to obtain information needed to make decision or to check any dysfunction in the information system. The ontology querying process may be triggered periodically or not. In periodic case, the query will be activated every day, week, month, etc. Otherwise, some causes will trigger the query, for example, an auditor request due to a security problem. This process is synchronized with the analysis process. In periodic case the scope of investigation is wide and the analysis must consider all the logged events and actions since the last audit. But in non periodic case, the analysis investigates only the logs related to a specific subject, action or object concerned by the activated event.

3. Querying the log for security investigation

A general schema for “Log Engine Model” is depicted in figure 2. The main process, after building the log ontology, is querying the log. This step uses SPARQL queries that are sent to a middleware called “adapter” which evaluates them. The main objective of this adapter is to perform a mapping between logged data which are stored in some built-in or native log format and which will be rewritten in our ontology model. In other words, the adapter takes a SPARQL query as input regardless compatibility between logged data format and the query language, evaluates it and returns the result in a desired format. There are four types of SPARQL query that we can use to query logged data: (1) “SELECT”, the adapter returns a set of tuples, (2) “ASK”, the result is a Boolean and (3) “CONSTRUCT” or (4) “DESCRIBE” it returns an RDF graph. We now illustrate some examples of “SELECT” queries and we discuss how each of them helps auditor to find an effective or potential violation in logged data.

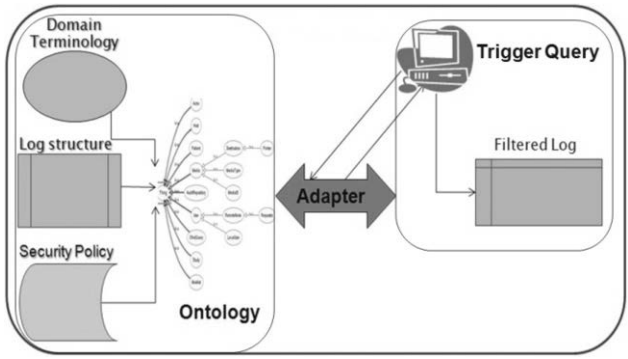

Figure 2. Log Engine Model

Query for monitoring purpose: this kind of query covers the wide investigation objective. It is usually used for security and quality controls. In the case of the "BeginStoringInstances" scenario, this query returns all users who perform an action on an object in this ATNA auditable event. The result obtained is displayed in triple format <subject; action; object>, see Figure 3:

subject	action	object
RNode1	http://www.semanticweb.org/ontologies/2011/5/Ontology1307868394438.owl#AccessAction	56678
RNode3	http://www.semanticweb.org/ontologies/2011/5/Ontology1307868394438.owl#CreateAction	23R5467
RNode2	http://www.semanticweb.org/ontologies/2011/5/Ontology1307868394438.owl#AccessAction	16562010
Ted	http://www.semanticweb.org/ontologies/2011/5/Ontology1307868394438.owl#DeleteAction	6573E24
Bob	http://www.semanticweb.org/ontologies/2011/5/Ontology1307868394438.owl#ModifyAction	12234
Jean	http://www.semanticweb.org/ontologies/2011/5/Ontology1307868394438.owl#DeleteAction	16562010
Alice	http://www.semanticweb.org/ontologies/2011/5/Ontology1307868394438.owl#ModifyAction	12234

Figure 3. Results of a monitoring query

In this case, there is not a security problem but the auditor verifies query results to find a potential violation or a system dysfunction. For example, the object number "12234" has been modified by "Bob" and "Alice". The auditor will verify the authorized and legal accesses of both users in the security policy.

Query in the case of dysfunction: this kind of queries has a specific scope of investigation. It is issued by the auditor for a particular security reason, for instance creation of an unknown patient record. In this case, the auditor must find subjects who performed "create action" of this patient record and verify the consistency between the query result and policy rules. Thus, the auditor checks if these subjects were allowed to perform this action. In Figure 4, an "order Filler" subject has created a patient record "05KAHNA". This action seems abnormal because "order Filler" is an application that manages the working list and accepts or rejects the scheduled work orders.

subject	action	object	log
Order Placer	http://www.atna.fr/ontologies.owl#PatientRecordModify	23HANAZ	http://www.atna.fr/ontologies.owl#Log6
Order Filler	http://www.atna.fr/ontologies.owl#PatientRecordCreat	05KAHNA	http://www.atna.fr/ontologies.owl#Log5
ADT Registration	http://www.atna.fr/ontologies.owl#PatientRecordAccess	23HA1279	http://www.atna.fr/ontologies.owl#Log4

Figure 4. Results of query about dysfunction

Query in the case of violation: let Ted be a patient who complains for an unexpected modification in his study. In this case, the query interrogates which subjects and actions have accessed to Ted's study. In Figure 5, there are three actions performed by three

different users on Ted's study. The auditor verifies in the set of security rules, who was authorized to modify Ted's study.

subject	action	log
Bob	http://www.atna.fr/ontologies.owl#ModifyAction	http://www.atna.fr/ontologies.owl#Log3
Helia	http://www.atna.fr/ontologies.owl#ModifyAction	http://www.atna.fr/ontologies.owl#Log2
Alice	http://www.atna.fr/ontologies.owl#AccessAction	http://www.atna.fr/ontologies.owl#Log1

Figure 5. Results of query to detect violations

4. Conclusion

Formalizing logged data and querying them provides efficient means to perform *a posteriori* access control. An ontology modelling of logs brings several benefits with regards to flexibility, adaptability and work efficiency. It lets us interrogate logged data using query languages such as SPARQL and detect potential policy violations. There are several interesting perspectives to this work. One of them is to use logged data to manage changes in more advanced security policy. For example, we can use logged data to check policy updates, detect creation of user sessions, activation of roles in these sessions or verify context changes. Since standard log formats were not initially designed to audit such security related actions, this is not a trivial extension that will be addressed in a forthcoming paper.

References

[1] Corin R, Etalle S, J.den Hartog, Lenzini G, Staicu I. Logic for auditing accountability in decentralized systems, Formal Aspects in Security and Trust Springer. Berlin 2004; 173: 187-202

[2] Azkia H, Cuppens-Boulahia N, Cuppens F, Coatrieux G. Reconciling IHE- ATNA Profile with aposteriori Contextual Access and Usage Control Policy in Healthcare Environment, 6th IEEE International Conference on Information Assurance and Security. Atlanta, USA 2010; 197-203

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

[4] Integrating the Healthcare Enterprise, IHE IT Infrastructure Technical Framework Volume I (ITI TF-1) Integration Profiles, August 2009. www.ihe.net/Technical_Framework/index.cfm#IT

[5] Ferraiolo DF, Sandhu R, Gavrila S, Kuhn DR, Chandramouli R. Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security, 2001; 4(3):224-274

[6] Abou El Kalam A, El Baida R, Balbiani P, Benferhat S, Cuppens F, Deswarte Y, Miège A, Saurel C, Trouessin G. Organization Based Access Control Policy 03 Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks. June 2003; 120-131

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