Extracting Attribute-Based Access Control Rules From Business Process Event Logs

Amani Abou Rida
Computer Science Department
Lebanese University - Faculty of sciences
Beirut, Lebanon
amani.abourida96@gmail.com

Nour Assy
Computer Science Department
Lebanese International University
Beirut, Lebanon
nour.assy@liu.edu.lb

Walid Gaaloul
Computer Science Department
Telecom sudParis
Paris, France
walid.gaaloul@telecom-sudparis.eu

***Abstract*—Protecting sensitive information from unauthorized access is recognized as a crucial issue for today's organizations. Identity and Access Management is one of the best practices techniques that ensure that the right people have access to the right systems at the right time. In particular, Attribute-Based Access Control (ABAC) models have recently gained popularity because of their capability to provide fine-grained and contextual access control that is not based on the user but on the attributes of every component in the system. Despite the benefits of adopting ABAC, it is commonly agreed that deploying an ABAC system is a complicated, time-consuming and challenging task. This is because all attributes of the system must be defined, and acess rules must not only be created, but also regularly monitored and reviewed. In this paper, we propose an automated approach to extract ABAC rules from event logs which record the actual execution of business processes. Event logs capture which tasks are performed by whom and at what point in time, and what data are taken as input and output. Therefore, they provide rich information on task and data access policies. Concretely, we propose to use (i) process mining techniques in order to analyze the event log and extract useful attributes and (ii) data mining techniques in order to learn the ABAC rules. To validate our approach, we (i) developed a java application, and (ii) performed experiments on a real-life event log. Experimental results show that our approach is efficient and feasible.**

***Index Terms*—Process mining, Attribute Based Access Control Model, Event logs, Association Rule Mining.**

I. Introduction

Organizations are increasingly becoming dependent on information technology to perform their business operations, thereby meeting their business objectives. In order to ensure reliable execution of business processes and protect data from unauthorized access, security measures need to be implemented. Identity and Access Management (a.k.a access control) is considered as a crucial security measure and a strong driving force for protecting the data, employees, and property of an organization. Roughly speaking, the purpose of access management is to grant authorized users access to appropriate data and deny access to unauthorized users.

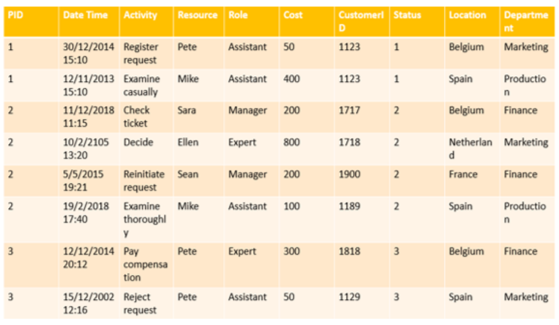
Recently, there has been a growing interest in Attribute-Based Access Control (ABAC) models [7] which define access control rules based on the attributes of every component in an information system. In an ABAC system, any type of attribute such as user attributes (a.k.a subject attributes), resource attributes and other relevant contextual attributes, such as date-time, are used to determine access. For instance, the rule "*Permit managers to access financial data provided they are from finance department*" would allow users with attributes of "Role=Manager" and "Department=Finance" to access data with the attributes of "Category=Financial". This makes ABAC a flexible and fine-grained strategy to manage users' access operations.

In order to deploy ABAC, one has to manually define all the attributes of the system, assign attributes to each system component and create and maintain policies (a.k.a rules) according to the security requirements. This makes ABAC systems complicated and hard to deploy. This paper addresses the aforementioned issue by proposing an automated approach to learn ABAC rules from business process event logs. Event logs [1] record the actual execution of business processes in an organization. They capture which tasks are performed by whom and at what point in time, and what data are taken as input and output. Therefore, they provide rich information on task and data access policies. The extracted ABAC rules depict the "current state" permissions within an organization. In case of an existing ABAC model, our extracted rules can be used to monitor and review the granted permissions. In the other case, they serve as starting point for further refinement and customization to derive "target state" ABAC models that represent the tailored permissions to execute business processes. Concretely, our approach is divided into two main steps. In the first step, we automatically extract from the event log the attributes and relations that are required for mining the ABAC rules. In the second step we use Association Rule Mining to learn ABAC rules from the event log.

The remainder of the paper is organized as follows. In Section II, we present an example that will be used throughout the paper to illustrate our approach. Section III formalizes some concepts and definitions that are required for our approach. In Section IV, we detail our approach for extracting ABAC rules from event logs. Our implementation and experimental results are reported in Section V. In Section VI, we discuss some related works before we conclude in Section VII.

II. Running Example

We consider a scenario that describes a process for handling a request for ticket compensation within an airline. To handle this ticket, the process should be secured so that each worker should know his work and if he can access the request or not. In general this process is shared between different process users. It will be accessed according to specific rules. These rules are complicated in the process if a company decided to do them manually. For that we should improve information sharing by maintaining control of that information. In order to ease the process security experience, the process provider decides to provide attribute based access control rules that relies upon the evaluation of attributes of the subject, attributes of the object, environment attributes, and the formal relationship or access control rule or policy that can define the allowable operations for subject-object attribute combinations. These attributes needed in an ABAC model are to be extracted from the event logs. For example consider an event log where every row in it corresponds to an event.These events have different properties, for example we can see in the first column the PID which is the case ID. The second column refers to the time-stamp in which the activity is being executed by the resource. The third column, refers to the activity that is being executed. Then there is a column referring to the resource which is the person executing the corresponding activity. And we can have all kinds of other columns with other data like role, cost, department, location, and status. Using the event log shown in figure 1 we can extract a rule with A resource.role == ”Manager” AND resource.status == “2” AND environmental.dateTime == “10/2/2105 13:20” if object.cost == “8” can do action == ”decide” AND object.customerID == ”1718” will lead for Ellen to have an access for the object while Sara can’t have access to it. For this reason the above event log should be analyzed in order to (i) classify the attributes according to the ABAC model and (ii) infer the set of ABAC rules. However, doing this manually is an incredibly tedious and error-prone task. Therefore, we propose in this work an automated approach for extracting ABAC rules from an event log.



PID	Date Time	Activity	Resource	Role	Cost	CustomerID	Status	Location	Department
1	30/12/2014 15:10	Register request	Pete	Assistant	50	1123	1	Belgium	Marketing
1	12/11/2013 15:10	Examine casually	Mike	Assistant	400	1123	1	Spain	Production
2	11/12/2018 11:15	Check ticket	Sara	Manager	200	1717	2	Belgium	Finance
2	10/2/2105 13:20	Decide	Ellen	Expert	800	1718	2	Netherland	Marketing
2	5/5/2015 19:21	Reinitiate request	Sean	Manager	200	1900	2	France	Finance
2	19/2/2018 17:40	Examine thoroughly	Mike	Assistant	100	1189	2	Spain	Production
3	12/12/2014 20:12	Pay compensation	Pete	Expert	300	1818	3	Belgium	Finance
3	15/12/2002 12:16	Reject request	Pete	Assistant	50	1129	3	Spain	Marketing

Figure 1. Shows an excerpt of the event log

III. Preliminaries

This section presents two main ingredients of our approach: event logs which are used as input of our approach (detailed in Section III-A) and attribute-based access control models which represent the output of our approach (detailed in Section III-B).

A. Event Logs

An event log contains the execution data of a business process and is recorded by the information system. Event logs are used by process mining techniques to discover process models, to check the conformance of a-priori process models, to detect execution errors or to observe social behaviors [1]. The structure of an event log is defined by the XES standard which defines an event log as a set of traces [5].

Figure 2 shows the class diagram of an event log [1]. A log consists of traces and a trace consists of events. The events within a case are ordered and they can also have attributes. Examples of typical attribute names are activity, Contextual Attribute (e.g. time), Object Data (e.g. costs), and resource. An event log accumulates the execution history of one process. A log case corresponds to one process instance execution. The list of the most common attributes in event logs are:

- Case ID: which is the process instance id of the event.
- Activity: name of the action performed in the event.
- Event Type: which refers to the event state such as started, paused, resumed, and completed.
- Time-stamp: date and the time at which the event has been executed, establishing an order between the events.
- Resource: name of the resource that initiates the event. Data: data attribute related to the event.

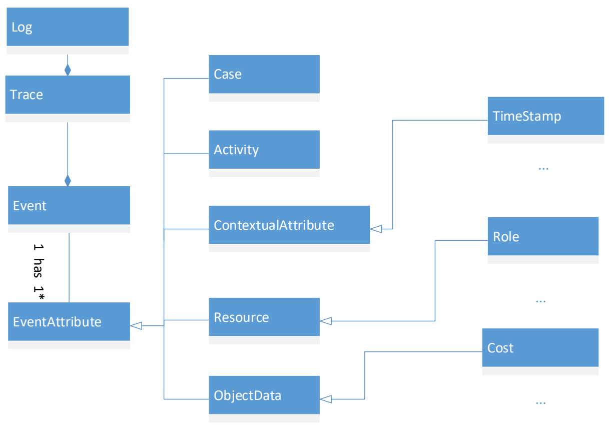


Figure 2. Class diagram for an event log

Definition 1 (Trace, Event log, Event Attribute) [11]. Let E be the event universe, i.e., the set of all possible event identifiers [11]. An event log is a set of traces and each trace contains number of events, a trace $< e1, e2, . . ., en > \in T$ is a sequence of events.
Events may be characterized by various attributes, e.g., an event may have a time-stamp, that refers to an activity, which is performed by a specific person, has associated costs, etc.

[1]This diagram is a slightly modified version of the diagram presented in [1]

Let AN be a set of attribute names. For any event e $\in$ E and name n $\in$ AN, #n(e) is the value of attribute n in event e. If event e does not have an attribute named n, then #n(e) = $\perp$ (null value) [11].
Each event ei $\in$ E has event attribute. These event attributes corresponds to activity #activity (e) = a $\in$ A, and resource #resource (e) = r $\in$ R, and #time(e) = t $\in$ D is the time-stamp of event e. For convenience we assume the following standard attributes:

- #activity (e) is the activity associated to event e.
- #time (e) is the time-stamp of event e.
- #resource (e) is the resource (user) associated to event e.

We can also have another attributes that can be classified according to contextual attributes (time, location ...) and object attributes (Data type, cost, status ...). Let CA be set of contextual attributes where CA = D,L ... and OA be a set of object attributes where OA= DT, S, CS.

B. Attribute Based Access Control Model

ABAC model leads to grant or deny user requests according to the arbitrary conditions of the user, attributes of the object, and environment conditions that can be recognized and more similar to the policies at hand. As owners of the objects, they have the permission to establish a policy that can relates what operations may be performed upon those objects, by whom, and in what context those subjects may perform those operations.

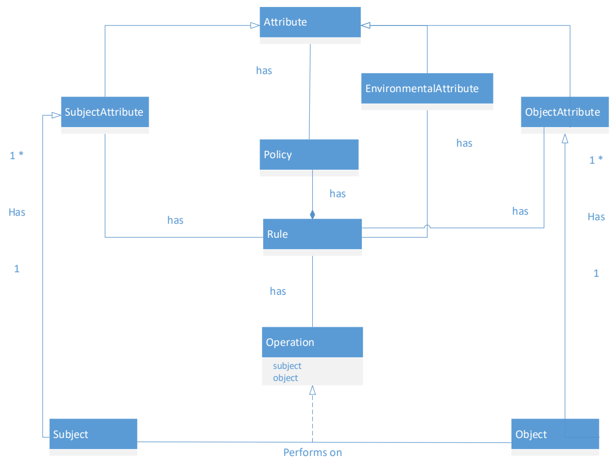


Figure 3. Class diagram for attribute based access control model

The main elements in ABAC model is the attributes that can be about anything and anyone. These attributes are likely to fall into 4 different categories or functions (as in grammatical function). Figure 3 shows a class diagram for ABAC model that contains main categories. These categories are the following:

- Subject attributes: That describes the attributes in which they express the user who wants to access e.g. age, status, role, job title...
- Operation attributes: They present the action or activity being done e.g. read, delete, view, approve...
- Object attributes: attributes that express the object (or resource) being accessed e.g. the object type, the department, the location, the cost...
- Contextual (environment) attributes: attributes that deal with time, location or dynamic aspects of the access control scenario.

ABAC is also concerned with the policy and rules. These rules are based on the privileges of subjects and how resources or objects are to be protected under which environmental conditions in order to determine if access is allowed or not.

Definition 2 (ABAC relation) We also define the following relation: A subject - operation - object SOPO permission tuple is a tuple $<$s, op, o $>$ containing a subject s $\in$ S, an operation op $\in$ OP, and an object o $\in$ O. This tuple means that subject s has permission to perform operation op on object o. For convenience we assume the following standard attributes:

- #SubjectAttribute (s) is a function that returns a set of pair (attribute name, value) of the subject.
- #ObjectAttribute (o) is a function that returns a set of pair (attribute name, value) of the object.
- #EnvironmentalAttribute ($<$s, op, o $>$) is a function that returns a set of pair (attribute name, value) of the environmental attributes.

Definition 3 (ABAC rule) A rule ABAC-R is a tuple $<$ #SubjectAttribute (s), SOPO , #ObjectAttribute (o), #EnvironmentalAttribute ($<$s, op, o $>$) $>$. The ABAC rule is defined as follow: #SubjectAttribute(s) $\wedge$ SOPO $\wedge$ #EnvironmentalAttribute ($<$s, op, o $>$) $\Rightarrow$ $\#ObjectAttribute(e)$.

IV. Mining ABAC rules

A. Approach overview

In this section, we present an overview of our automated approach for extracting ABAC rules from event logs. The input here is an event log that contains attributes defined in Definition 1. The output of this algorithm generate ABAC rules needed for an ABAC model.

Algorithm 1 Building an attribute based access control guidance model

```
1) Input: E
2) Output: ABAC-R
3) for e ← E
4)     E-attributes = get-Attributes (e)
5)     E-relations = get-Relations (e, E-attributes)
6)     ABAC-attributes = get-Attributes (e)
7)     ABAC-relations = get-Attributes (e)
8)     Mapping (E-attributes, E-relations, ABAC-attributes,
       ABAC-relations)
9)     E-saved = Save-To-ARFF (E-attributes)
10) end for
11) ABAC-R = Apriori (E-saved, minS, minC)
```

The algorithm proceeds in four main steps. In the first step, the event log is analyzed to extract attributes and relations needed in our model that are stored in E-attributes and E-relations respectively (lines 4, 5). Then, in the second step, the attribute based access control model is also analyzed to extract main attributes and relations to be stored in ABAC-attributes and ABAC-relations respectively (lines 6,7). The third step consists of mapping between the attributes and relations of event logs to the attributes and relations of attribute based access control (step 8). Finally, the last step consists of generating the set of attribute based access control rules (step 11).

B. Extracting attribute based access control rules from business process event logs

In this section we present the approach in two main steps as following: In the first step, we show a mapping between event logs and ABAC models by analyzing the event log and mapping the event logs' attributes to ABAC models' attributes. In the second step, we applied apriori algorithm to extract ABAC rules.

C. Mapping between event logs and ABAC models:

In this section, we present two main parts to have the mapping between an event log and an ABAC model. In the first part, we analyze the event log to extract the attributes needed. In the second part, we map the attributes extracted from the event log to ABAC model attributes.

1) Analyzing the event log:
   In this section we analyzed the event log given as input. In this step, we aim to show how we can benefit from the event log to extract the attributes needed for creating an ABAC model. The class diagram in figure 3 summarizes the analysis for any event log. We extended the original event log so that we can benefit from the attributes in creating ABAC model. An event log is decomposed of traces that contain a number of events. Each event has a set of attributes and we modified some of them as follow:
   - Resource: name of the resource that initiates the event. It can also contain the role of the resource, age, and the status.
   - Contextual attributes: can be: Time-stamp: date and the time at which the event has been executed, establishing an order between the events. It can also contain location, and department ...
   - Object Data: data attributes related to the event. It can contain cost, and type ...

   After analyzing the event log we can realize that any attribute can be classified into these categories. The main thing is to extract attributes that match the above elements so that any event should have at least the following attributes (activity, object data, contextual attribute, resource).

Table I
SHOWS MAPPING BETWEEN EVENT LOGS' ATTRIBUTES AND ABAC MODELS' ATTRIBUTES

Event Log	Attribute Based Access Control
Resources	Subject Attribute
Object Data	Object Attribute
Contextual Attribute	Environmental Attribute
Activity	Operation
Resource-Activity-Object Data (RAO)	Subject-Operation-Object (SOPO)

   Moreover, we can infer a Resource-Activity-ObjectData relation from these attributes which is defined as following:
   Definition 4 (Event relation) Resource-Activity-ObjectData (RAO) assignments: The relation RAO is defined as RAO = $\{$ (r, a, od) $\in$ R $\times$ A x OD $\exists$ e $\in$ E , #resource(e) = r $\wedge$ #activity(e) = a $\wedge$ #objectData(e) = od $\}$. RAO relation holds if at least one event e $\in$ E with resource r $\in$ R that executes activity a $\in$ A and ObjectData od $\in$ OD is recorded in the event log. We say that we assign resource r to activity a that can be accessed by object data od .
2) Mapping event logs' attributes to ABAC models' attributes:
   Table 1 shows a mapping where a resource is said to be subject attribute, an object data is the object attribute, contextual attribute is the environmental attribute, and the activity is the operation.

D. Apriori-based approach for extracting ABAC rules:

The goal here is to find associations of items that occur together more often than one would expect from a random sampling of all possibilities. Apriori starts by selecting the frequent single attribute, then generates the candidate pairs of attributes in the event from the frequent singles and so on, until it finds all possible attributes according to all the events in the event log. It uses Support, a well-known metric to compute the frequency of a set of correlated attributes in the event log. The support is defined as the fraction of correlated attributes of each event in the event log in which they always appear together.

$$S = \frac{\mid Ae \mid}{\mid E \mid} \tag{1}$$

Where $\mid Ae \mid$ is the number of correlated attributes in an event and $\mid E \mid$ is the number of events in an event log. A support is equal to 1 if all the events in the event log repository contain the correlated attributes. A support is equal to 0 if none of the events in the event log contain the corresponding attributes together. A set of correlated attributes is frequent if its support is above a given threshold mSupp.

In the second step of the algorithm, the set of relevant attribute based access control rules in the form of LHS $\Rightarrow$ RHS are derived from the frequent correlated attributes. In order to keep only relevant rules, the confidence metric is computed to evaluate the probability of occurrence of a rule. The confidence of an attribute based access control rule tt:

LHS $\Rightarrow$ RHS is defined as the probability of occurrence of the attributes in the right-hand side RHS given that the configurations in the left-hand side LHS are selected. In order to have ABAC rule as in definition 3 we should apply filtering on the attributes to the derived rules. In this definition we should be sure that the attributes in the left side contains the subject attribute, environmental attribute, and the operation that the user is doing whereas the right side should contain the object attribute that describe the data that is accessed. Rules: #SubjectAttribute(s) $\wedge$ SOPO $\wedge$ #EnvironmentalAttribute(<s, op, o >) $\Rightarrow$ #ObjectAttribute (o) .

$$C = \frac{Sup(RHS \cup LHS)}{SupRHS} \tag{2}$$

Where Sup(RHS LHS) is the support of the attributes (Subject Attributes, Environmental Attribute, Object Attributes, and Operation) in the right-hand and left- hand sides of Rule and SupRHS is the support of the attributes in the right-hand side (Object Attribute).

When applying appriori algorithm on these events, we can recognize in-frequent rules since the time in these rules contain numerical values and each event contains different time. The discretization on time-stamp is defined in our work is the ability to specify that certain event needs to be executed at a specified date. Time might be absolute or relative and the granularity needs to be considered.

Definition 4 (Absolute interval time-stamp) it is a time-stamp pattern constraint that defines a punctual temporal structure and refer to start and finish times of an activity [8]:

1) Must Start On (MSO), Must Finish On (MFO): indicates the exact time, in which an activity must be scheduled to begin or complete;
2) Start No Earlier Than (SNET), Finish No Earlier Than (FNET): indicate the earliest possible time that an activity can begin or complete;
3) Start No Later Than (SNLT), Finish No Later Than (FNLT): indicate the latest possible time that this activity is to begin or complete
4) Start No Earlier Than (SNET), Finish No Later Than (FNLT): indicate the earliest possible time that this activity is to begin and the latest possible time that this activity is to complete.

In our approach we decided to use the concept of (4) where SNET indicates the minimum time and FNLT indicates the maximum time. Each event e $\in$ E has event attributes #timestamp(e), #resource(e), #activity(e), and #object(e). We compare #timestamp(e) in each event e that have the same #resource(e), #activity(e), and #object(e) so that we can have the min Time-stamp(e) between all the events. We then change all the #timestamp(e) to min Time-stamp(e) . This is similarly done to calculate the max Time-stamp(e).

Definition 5 (Relative time-stamp) It is a time-stamp pattern constraint that refers to a dependency between the activities. This dependency is a relationship between two activities, aq and al where q is not equal to l, in which one activity depends on the start or finish of another in order to begin or end [6]. In our approach this relationship depends on the start event of each trace compared with the other events in the same trace. This is defined by the following: For each trace T $\in$ E in event log and each event e $\in$ T in the trace we obtain start event se $\in$ T which is the first event in each trace. The time-stamp here is splitted to years, day, hours, minutes, and seconds. We compare #timestamp(se) - #timestamp(e) in each event e in same trace T. We then see the events e that have same #resource(e), #activity(e), and #object(e) so that we can have the min Years(e), Day(e), Hours(e) , Minutes(e) , Seconds(e) between all the events. We then change all the #timestamp(e) to min Years(e), Day(e), Hours(e) , Minutes(e) , Seconds(e) that have same #resource(e), #activity(e), and #object(e).

V. Evaluation and Validation

We implemented our approach as a java application to generate ABAC rules. We integrated ProM libraries [2] in order to import and extract information from an event log. We also used “filter events” in ProM to reduce the noise impact on the data. ProM is an open source framework for implementing process mining techniques. Moreover, we integrated Weka software that contains a group of visualization tools and algorithms for data analysis and predictive modeling to generate rules using apriori algorithm [9]. The user can choose different values of the support and confidence to apply apriori algorithm so that we can generate the ABAC rules.

To evaluate our approach, we used a real life event log provided from BPI challenge 2018 [12]. The data set consists of real business processes for EU direct payments for German farmers. The event log consists of 43809 traces over a period of three years. Tables II and III show some statistics about the number of traces, events, and the number of values in each attribute before and after filtering the event log from the noise respectively. The attributes that are used in the event log are Resource, Activity, Document Type, Time-stamp. These are classified into ABAC attributes as following:

- Subject Attribute: Resource
- Object Attribute: Document Type
- Environmental Attributes: Time-stamp

The Activity is the operation used to perform an access by the subject on the object according to the following attributes. Moreover, Table III shows the attributes Absolute Time-stamp, years, minutes, seconds, days, and hours are observed after applying both Relative and Absolute Time-stamp that are defined in the approach.

In the first experiment, we evaluate the quality of the approach by calculating the complexity of the extracted rules. This is studied by counting the number of rules that are derived before and after filtering the attributes to derive ABAC rules and by analyzing the impact of the Apriori support and confidence values (detailed in Section V-A). In the second experiment, we evaluate the quality of the approach

[2]http://www.promtools.org/

Table II
STATISTICS ON THE BPI CHALLENGE 2018 EVENT LOG

Traces	43809
Events	2514266
values in Activity	41
values in Resource	165
values in Document Type	8

Table III
STATISTICS ON THE BPI CHALLENGE 2018 EVENT LOG AFTER FILTERING

Traces	108
Events	6183
values in Activity	34
values in Resource	102
values in Document Type	8
values in Absolute Time-stamp	25
values in Years	3
values in Minutes	38
values in seconds	47
values in Days	7
values in Hours	21

by measuring the completeness of the extracted rules. This is studied by counting the average number of values extracted for the different attributes (detailed in Section V-B).

A. *Complexity of the rules and parameter impact*

In this experiment, we study the complexity of our approach which is expressed in term of the number of extracted rules. The higher the number of extracted rules, the more the complexity increases. Our objective is to study the impact of the different support and confidence values on the number of extracted rules. To do so, we compute (1) the number of extracted rules before filtering which includes all the rules that are generated by applying the apriori algorithm and (2) the number of extracted rules after filtering which includes the rules that contain the selected attributes for the ABAC. The average number of extracted configuration rules with different support and confidence values are shown in Figure 4.

First, the results clearly show that the number of rules decrease when the support values increase. The same applies for the confidence values. However, the effect of the support values is much more noticeable than that of the confidence values. This means that frequency plays an import role in the context of access control. Highly frequent rules may refer to access control rules that are respected in an organization while less frequent rules may indicate a violation of permissions and therefore need a closer inspection by experts. Second, the number of rules greatly decreases after filtering the attributes. This is because an ABAC rule is valid only if it includes the following mandatory attributes: the name attribute of the activity, at least one attribute of the subject (e.g. name, role, etc.), the timestamp attribute of the environmental attributes and at least one attribute of the object. An ABAC rule should also follow the following format (Subject attributes, Activity attributes, Environmental attributes $\Rightarrow$ Object attributes). By imposing such constraints, we can decrease the complexity of our approach by decreasing the high number of rules that are generated by Apriori and that are less interesting in the context of access control.

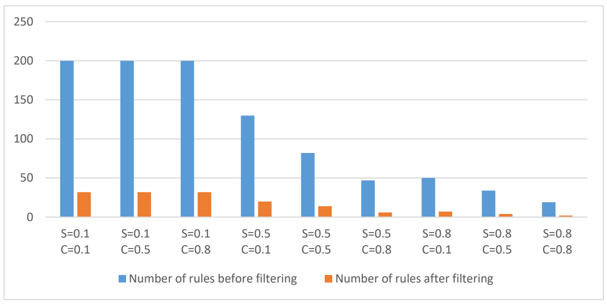


Figure 4. Bar graph that shows the number of rules before and after filtering the attributes according to different support and confidence values

B. *Completeness of the rules and parameter impacts*

In the second experiment, we target to study the completeness of our approach which is expressed in terms of the number of attribute values extracted in the rules compared with the total number of attribute values in the event log. On the one hand, a high number of attribute values per rule means that we are able to cover the association between the attributes of all ABAC elements from the event log. On the other hand, a high percentage of retrieved values per attribute means that our rules cover all possible elements in the event log choices and that we are able to find all possible ABAC rules. Our objective is to study the impact of the different support and confidence values on the number of extracted attribute values. To do so, We study (1) the number of extracted attribute values per rule before filtering which includes all the rules that are generated by applying the apriori algorithm and (2) the number of extracted attribute values per rule after filtering which includes the rules that contain the selected attributes for the ABAC. Moreover, we study the relation between the complexity and the completeness of the extracted attribute based access control rules. We study the average number of values of each attribute found in the extracted rules compared to the total number of attribute values in the event log that can be seen in table III. Figure 5 shows the average number of attribute values after performing filtering to the attributes so that the rules contains the selected attributes for the ABAC.

First, the results clearly show that the number of attribute values in the rules before and after increases when the support values decreases. The same applied for the confidence. However, the effect of the support values is much more noticeable than that of the confidence values. This can be explained by the fact that, in high frequency the rule is not shown. This leads to the conclusion that the complexity of the ABAC rules is positively correlated with their completeness (i.e. when the complexity increases the completeness increases) while both the completeness and complexity are negatively correlated with the support threshold value (i.e. when the support decreases, the completeness and complexity increase).

Therefore, one has to choose or to find a compromise between the complexity and the completeness of the results.

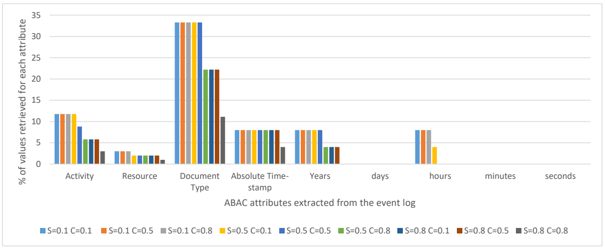


Figure 5. Bar graph that shows the average number of values retrieved per attribute according to different support and confidence values

As a conclusion the Experimental results showed that the extracted attribute based access control rules are of good quality in the mean of complexity and completeness.

VI. Related Work

Our work is related to two research areas: access control models (detailed in Section VI-A) and automated extraction of access control models from event logs (detailed in Section VI-B).

A. Access Control Model

Some access control mechanisms that the Computer security architects and administrators have developed to protect their object by mediating a request from subjects are defined in this section. Each mechanism has its own advantages and limitations but it is important to note the evolution of these models to fully appreciate the flexibility and applicability of the ABAC model. Some of these models are the following:

- MAC: Mandatory access control provides only the owner and guardian management of the access controls. This means the end user has no control over any settings that provide any privileges to anyone [4].
- DAC: According to [13] the Discretionary access control do not supply a high security assurance for two reasons: First, the granting access is transitive. Second, DAC policies are vulnerable to Trojan Horse attacks. A Trojan Horse program is the one that looks to be doing one thing on the surface but literally does something more underneath without the cognizance of the user.
- IBAC: Identity Based access control captures the identity of the subjects that want to access an object. This way is considered hard to be managed.
- RBAC: Role Based Access Control model defines roles that carry specific set of privilege that a subject request and by the access for object owner when determining the privilege associated with each role [3].The big issue with this access control model is that if one requires access to other files that he doesn't has permission to, he has to find another way to do it since the roles are only associated with the position; otherwise, security managers from other organizations could possibly get access to files they are unauthorized for [2].

B. Extracting Access Control Models from Event Logs

Looking at the previous studies related to our topic, one can see a closely related approach which is extracting Role-Based Access Control (RBAC) from event logs [10]. This approach consists of three main steps where the authors aim to derive an RBAC model from an event log. In the first step, called analysis, a process mining technique is used to extract process-related data from an event log in XES format. In the next step, extracted data can be transformed into an in-memory RBAC model. Before that, minor adjustments could be made to the extracted data, so that the data and relationships would reflect actual settings of the business process. In the final step, an RBAC model is exported to the XML-based format in order to support the data exchange between different applications, e.g., information systems could implement the RBAC model or access policy management systems could be used to enhance the RBAC model. ABAC models can be seen as a generalization of RBAC models, i.e. the core element of RBAC is "role" which becomes an attribute in ABAC.

VII. Conclusion and Future Work

In this paper, we proposed an automated approach to extract ABAC models from event logs which record the actual execution of business process. Given an event log as an input, we proposed to extract the attributes needed for an ABAC model and to learn the ABAC rules automatically. We used process mining and data mining techniques. The output of our result is a set of ABAC rules in the form of SubjectAttribute (s) $\wedge$ SOPO $\wedge$ EnvironmentalAttribute ($<$s, op, o $>$) $\Rightarrow$ ObjectAttribute (o).

We validated our approach by implementing it as a java application. We also performed experiments on a real life event log from BPI Challenge 2018. Experimental results showed that our approach is feasible and that the frequency is an important factor in the context of access control. Highly frequent rules may indicate that the right permissions are respected in an organization. In case our approach is used to create a new access control model, these rules help experts to set up their access control model. On the other hand, infrequent rules may indicate violation of permissions and need to be closely inspected by experts.

We aim to extend our work in two different directions. First, we plan to perform more experiments with real-life event logs to validate and generalize our results. Secondly, recently there has been a growing interest in cloud-based process mining solutions. This has raised issues about privacy and the call for cryptography techniques to secure event logs. Therefore, we plan to extend our approach to extract ABAC rules from secured event logs in order to check if the *intended* security is *actually* respected.

References

[1] Wil M. P. van der Aalst. *Process Mining: Data Science in Action*. 2nd ed. Heidelberg: Springer, 2016. ISBN: 978-3-662-49850-7. DOI: 10.1007/978-3-662-49851-4.

[2] Mark Ciampa. *Security+ guide to network security fundamentals*. Cengage Learning, 2012.

[3] DF Ferraiolo and DR Kuhn. “Natl Institute of Standards and Tech., Dept. of Commerce, Maryland, Role-Based Access Control”. In: *Proceedings of 15th Natl Computer Security Conference*. 1992.

[4] Virginia Nunes Leal Franqueira. “Access control from an intrusion detection perspective”. In: (2006).

[5] Christian W Günther and Eric Verbeek. “Standard definition”. In: *Fluxicon Process Laboratories, XES Version* 1 (2012).

[6] Rania Ben Halima et al. “Scheduling Business Process Activities for Time-Aware Cloud Resource Allocation”. In: *OTM Confederated International Conferences” On the Move to Meaningful Internet Systems”*. Springer. 2018, pp. 445–462.

[7] Vincent C Hu et al. “Guide to attribute based access control (ABAC) definition and considerations (draft)”. In: *NIST special publication* 800.162 (2013).

[8] Cosmina Cristina Niculae. “Time patterns in workflow management systems”. In: *BPM Center Report BPM-11-04, available online at http://bpmcenter. org/wp-content/uploads/reports/2011/BPM-l l-04. pdf* (2011).

[9] Weka Team. *Use WEKA in your Java code*. 2016.

[10] Taivo Teder. “Extracting Role-Based Access Control Models from Business Process Event Logs”. In: ().

[11] Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Vol. 2. Springer, 2011.

[12] Van Dongen, B.F. (Boudewijn) and Borchert, F. (Florian). *BPI Challenge 2018*. 2018. DOI: 10.4121/UUID:3301445F-95E8-4FF0-98A4-901F1F204972.

[13] Younis A Younis, Kashif Kifayat, and Madjid Merabti. “An access control model for cloud computing”. In: *Journal of Information Security and Applications* 19.1 (2014), pp. 45–60.