Advances in Science, Technology and Engineering Systems Journal Vol. 4, No. 5, 409-418 (2019)
www.astesj.com
Special Issue on Multidisciplinary Sciences and Engineering

ASTESJ
ISSN: 2415-6698

A Big Data Security Layer Meta-Model Proposition

Allae Erraissi[*], Abdessamad Belangour

Laboratory of Information Technology and Modeling, Hassan II University, Faculty of sciences Ben M'Sik, Casablanca, Morocc.

ARTICLE INFO

Article history:
Received: 11 July, 2019
Accepted: 09 October, 2019
Online: 28 October, 2019

Keywords:
Meta-model
Model Driven Engineering
Big Data
Security layer
Management layer

ABSTRACT

In Big Data, several solution providers offer distributions to handle this large amount of data. Given the variety of these solutions, we are working to provide universal meta-modeling for all layers of a Big Data system in order to address the issue of interoperability and portability between these solutions. As part of our continuous efforts to standardize concepts in Big Data world, we apply in this paper techniques related to Model-Driven Engineering "MDE" to propose a meta-model for the security layer in Big Data. This meta-model with the others that we have already proposed for the other layers of the Big Data system, will be used as a platform-independent according to Model-Driven Architecture pattern, which describes the structures of Big Data layers independently of any specific platform.

1. Introduction

The issues of the expression and application of security needs are present in all information systems. With the emergence of new types of environments, such as cloud computing and big data, this problem is becoming more complex. Hence, it is necessary to take into account their heterogeneity and the different levels of such an architecture to deal with this problem.

At the Big Data level, several distribution providers have proposed solutions to manage this huge amount of data, among these distributions, we found HortonWorks, Cloudera, MapR, InfoSphere BigInsghits IBM, Pivotal HD, etc. To put it another way, this massive amount of heterogeneous data led to the emergence of a large number of big data systems and technologies that share similar architectures but with different implementations. In essence, the common architecture of these data systems is composed of many components: Data sources, Ingestion, Hadoop Storage, Hadoop Platform management, Visualization, Monitoring, and Security Layers [1]. In our way for a unified abstract implementation, we proposed, in previous works, meta-models for data sources, ingestion [2], storage [3], management [4] and visualization layers [5]. We also relied on our previous comparatives studies to define key concepts of security layer in Big Data [6].

In this paper, we shall first present the security properties that express the security requirements of the software architecture. These properties are then combined to form a security policy: different models and languages of expression of security and insurance policies will be detailed. Then, we shall present the meta-model that we proposed for the security layer. This meta-model with the others already proposed for the other layers of the Big Data system will be used like a platform-independent according to Model-Driven Architecture pattern [7], which describes the structures of Big Data layers independently of any specific platform.

2. Related Work

Several research studies have been developed to standardize the security layer concepts at the Big Data systems level. Among the foremost relevant approaches in security modeling are UMLSec [8] and SecureUML [9]. Yet, both approaches use UML extensions to specify security requirements. UMLSec was created to verify the formal security requirements specifications within the system design. SecureUML was used to illustrate the MDS approach. SecureUML is specific to role-based access control infrastructures. It shows how an MDE policy could be applied to code generation for any aspect of security.

In the literature, several MDS approaches have been proposed. For example, on a middleware platform that integrates the implementation of the Corba component model with the OpenPMF security framework, Reznik et al. [10] present an MDS solution for

[*]Allae Erraissi, Laboratory of Information Technology and Modeling, Hassan II University, Faculty of sciences Ben M'Sik, Casablanca, Morocco, Email: erraissi.allae@gmail.com

developing secure applications. In the MDS approach, another UML profile was developed to model access control policies. Lang and Schreiner [11] show how OpenPMF architecture can be used to translate a high-level security-related regulatory requirement into enforceable authorization rules.

Our proposal is different from other existing MDS approaches, which use templates to represent design decisions and implementation details for target platforms. The application of Model Driven Engineering techniques that provide meta-modeling to the security layer in Big Data will advocate a systematic MDS application process. This application separates security requirements specifications and design decisions related to their implementation. Thus, the reuse of models is facilitated by this separation.

However, ModelSec proposes an intermediate model conforming to a target platform meta-model to reduce the semantic gap between the security requirements and the software generated. This intermediate step favors the reuse of the transformations. Unlike the other approaches that create UML profiles, a DSL is defined to express the security requirements. It has been implemented by applying meta-modeling techniques. Besides, ModelSec is a generic approach, whereas most MDS approaches are specific and they focus on access control policies.

Yet, other approaches manage security requirements. These approaches are not aligned with MDS, and they focus primarily on eliciting security requirements rather than generating software objects from them. Among these proposals are Secure Tropos [12] and [13], which introduce the concept of antigoal. Yu et al. [14] use ontologies and Haley et al. [15] present a framework consisting of a set of activities designed to meet security requirements.

3. Security Properties and Policies

In this section, we describe methods for formalizing the different security needs. We first present the usual security properties and those that can be derived from them. Then, we shall take existing models to apply these properties, security policies and languages to define them.

3.1. Security properties

Security properties are the basis for expressing security requirements. The set of security properties is commonly seen as a set derived from three main properties: confidentiality, integrity, and availability (CIA: Confidentiality, Integrity, Availability). The exact interpretation of what these three properties imply depends on the context of use. However, their definition and application is an essential part of the safety assessment criteria, at both European [16] and international [17]. Several definitions of these properties exist in the literature [16,17,18]. We present a synthesis here.

3.1.1. Confidentiality

Confidentiality is about preventing unauthorized disclosure of information. It aims to prohibit unauthorized access to information. Accordingly, the property of confidentiality implies that the information is accessible only by certain entities, that is to say, that certain entities must not be able to obtain the information. This property is often used in sensitive environments, such as defense. It concerns both direct access and information transfer.

3.1.2. Integrity

Integrity means the fact that information cannot be unauthorized modified or deleted, whether during processing, storage or transfer of information. This property does not only concern voluntary modifications and deletions, but also accidental acts. Just as in the case of confidentiality, the integrity property specifies all the entities authorized to modify or delete information. By default, other entities cannot alter the information.

3.1.3. Availability

The availability property expresses the ability to access information or a resource. It is related to the reliability of a system since an unavailable system is a failing system. The temporal notion of access is relative to the field of application: access to information or service on a critical system (for example, in the medical field) must be done more quickly than on a non-critical system (for example, a website).

3.1.4. Insurance

The property triple of confidentiality, integrity, and availability is sometimes extended with other properties [19], such as the insurance property. The purpose of this is to provide evidence that the other properties have been applied and that they have the desired effect. The assurance of a property is the verification that this property has been properly applied. The level of insurance required depends on the system in question and must, therefore, be adapted according to the criticality of the system. In sum, the purpose of insurance property is to verify that the other properties of the policy have been applied and that this application has had the desired effect.

3.1.5. Derived properties

Privacy, integrity, and availability properties are the basic concepts of security. They can be used to define derived properties which are special cases, subsets or combinations of these basic properties. In this section, we describe some of these derived properties.

- PROCESS CONFINEMENT

Process containment has been defined by Lampson [20]. The problem of containment concerns the prevention of the disclosure by a service or a process of information considered as confidential by the users of this service. According to Lampson, one of the features needed for a process is that it should not disclose information and it must not store information. Indeed, if a process stores information and a user can observe this process, then there is a risk that the user can access the information. If the process does not store the information, then it cannot be disclosed. This process containment property can thus be seen as the isolation of the process from the rest of the system.

- AUTHENTICATION

The authentication property is a property that allows or denies access to information or service to entities. Authentication is the process of establishing trust in the identity of an entity [21]. This authentication property is essential to apply the properties seen previously. Indeed, it makes it possible to establish the identity of a user, a service or a system, which is necessary to give him the appropriate rights.

3.2. Security Policies

A security policy is a set of properties that express the security needs of a system or a set of systems. Most security properties are based on resource access control. As a result, the concepts introduced in access control models can be transposed into a language to formally express the properties of the previous section. In this section, we shall first describe the main models of historical access control. Then, we shall present security policy and insurance policy expression languages, which express and apply security properties.

3.2.1. Historical models

As defined above, some security properties can be applied by access control mechanisms. An access control system is usually modeled using the following three elements:

- A set of topics that are the active entities of the system (for example, processes);
- A set of objects that are the passive entities of the system, on which the subjects can perform actions (files, sockets, etc.);
- A set of permissions that represent the authorized actions between a subject and an object (reading, writing, etc.), or between two subjects (sending a signal).

▪ **Control of Discretionary Access**

Discretionary Access Control (DAC) is the default historical model present on most operating systems. In this model, the management of access rights to a resource is left to the discretion of the owner of that resource. For example, on UNIX, the owner of a file can set the read, write, and execute rights for itself, for the members of the group that owns the file, and for all other users in the system.

An access control model can be represented as a matrix, where a line represents a subject, a column represents an object or a subject, and each element of the matrix represents a set of permissions of the subject on the object (or on the second subject). This model was formalized by Lampson [22, 23] using capacity lists and Access Control Lists (ACLs). Therefore, it proposes to indicate, in an array A, the set D of the protection domains (representing program execution contexts, that is to say, the subjects) on the lines, and the set X of the objects on the columns. Hence, Lampson defines the lists of capacities (definition 1) which establish the permissions of a domain d on all the objects o of the system. This is the set of actions allowed for each domain d.

Definition 1: List of Capabilities

Given a d∈D domain, the list of capabilities for domain d is all couples:

$$(\mathbf{o}, \mathbf{A}[\mathbf{d}, \mathbf{o}]), \forall \mathbf{o} \in \mathbf{X}. \qquad (1)$$

Afterworlds, Lampson defines access control lists (definition 2) that specify the set of permissions granted on an object for each domain in the system.

Definition 2: Access Control List (ACL)

Given an o∈X object, the access control list (ACL) for object o is the set of pairs:

$$(\mathbf{d}, \mathbf{A}[\mathbf{d}, \mathbf{o}]), \forall\ \mathbf{d} \in \mathbf{D}. \qquad (2)$$

However, this model is complex to update. For example, when adding a new user to the system, a complete row must be added to the A matrix. The Lampson model has therefore evolved to the HRU model. In the HRU model [24], discretionary access control is modeled from a matrix P containing all the subjects' rights to the objects. In this template, subjects can edit the access control matrix to create or delete topics or objects, as well as edit existing permissions. The HRU model models protection using the following elements:

- An access control matrix P;
- A set S of subjects and a set O of objects;
- A set R of generic rights (reading, writing, execution, possession, etc.);
- A finite set C of commands representing all the operations provided by the operating system (creation of files, modification of rights, etc.);
- A set E of elementary operations: enter and delete (add and remove rights), create subject and create object, destroy subject and destroy object (destruction of subjects and objects).

A triplet (S, O, P) represents the configuration of the system protection. In order to study the problem of the safety of a protection system, the authors of the model HRU are interested in the transfer of privilege (right) occurring when a command inserts a right r in the matrix P. The safety problem can be defined in the following way: given an initial configuration of the security policy, a system is considered safe for a right r if none of the commands of this system causes the transfer of the right r. The authors then showed that if the commands contain only one elementary action, the safety problem is decidable but its verification algorithm is NP-complete. Nevertheless, in the general case (the commands contain several elementary actions), the problem is undecidable. In the case of an operating system, the commands are not mono-operational and the problem of the safety of the protection system is therefore undecidable. This shows that it is impossible to guarantee security properties with a discretionary access control model.

Other DAC models have been defined to extend the HRU model, including TAM (Typed Access Matrix) [25] and DTAM (Dynamic Typed Access Matrix) [26]. TAM extends the HRU model by incorporating a strong typing concept [27] which corresponds to the association of immutable security types to all the subjects and objects of the system. DTAM extends the TAM model by adding the ability to dynamically modify object types. The different models of discretionary access control are historical models. They are mainly used for system rights management.

▪ **Mandatory access control**

Discretionary access control leaves the management of resource permissions to their owners, that is, users of the system.

In practice, this system is limited since many attacks against systems aim to obtain privileged access (root, or super-user). Such an attack, so-called elevation of privilege, is intended to obtain rights greater than those possessed. Thus, it allows the user to gain full access to the system and its resources and to overcome discretionary control. Besides, studies [28,29] have shown that DAC models are vulnerable, particularly because of the need for users to correctly define the set of resource permissions. Any definition error can create a security vulnerability that can be exploited to gain privileges. Mandatory Access Control (MAC) is consequently intended to address this problem by imposing a security policy on system users. Anderson proposes the use of a reference monitor [30] to control the interactions between subjects and objects and to determine which ones are valid (that is, the ones allowed by the policy). This section shall present the main models of mandatory access control. These models explain the concepts of security properties (integrity, confidentiality) in order to apply them.

The Bell-LaPadula (BLP) model [31] is based on the confidentiality needs of the military and is accordingly intended to prevent disclosures. This model extends the HRU model by adding the notion of a label associated with each subject and object of the system. A label corresponds to a security level and is composed of two security identifiers: the first identifier, hierarchical, indicates the level of classification (for objects) or authorization (for subjects), e.g., secret or top secret. The second identifier, called category, specifies the organizations using the information, for example, military or private.

In addition to the classic rules which are defined by an access control matrix, two new rules are defined:

- **ss-property** (simple security property): for reading access to be authorized, the subject requesting it must have a level of authorization greater than or equal to that of the object;
- ***-property** (star property): Information can only be transferred from a lower classification object to a higher classification object.

These two rules make it possible to ensure the confidentiality of the information and its non-disclosure. However, the existence of hidden channels can cause information flows that cannot be controlled. For this reason, a more restrictive version of BLP has been proposed with the following rules:

- No Read Up: A subject requesting read access to an object must have a security level greater than or equal to the object;
- No Write Down: A subject requesting write-only access (adding data) to an object must have a security level that is less than or equal to the object.

Consequently, a subject requesting read and write access to an object must have the same level of security as the object. This model is sometimes called MLS (Multi-Level Security), and thus refers to the level system used to define security rules.

A dual model at BLP, Biba [32] has been defined to meet integrity needs. The Domain and Type Enforcement (DTE) model [33] is a high-level mandatory access control model. Unlike models such as BLP or Biba, it is not intended to apply a specific security property, but rather to define the allowed access between different entities in the system. Nonetheless, this model can serve as a basis for implementing security property models, such as BLP, Biba, or process containment. The DTE replaces notions of topics and objects with those of domains and types. Thus, each object has a type and each subject runs in a domain. Access rights on types and domains are then defined for each domain. Consequently, this model aims to restrict the resources accessible by a process (even for privileged processes by the principle of least privilege) and to control which processes have access to sensitive resources. The different models of mandatory access control can be applied at the same time at the system level, in distributed environments such as the case of military infrastructures, or even be integrated within a software.

3.2.2. *Expression languages of security policies*

The access control models that were presented above are the historical models that allowed the definition of security policy expression models and languages. A security policy consists of a set of rules that may have the purpose of applying properties, possibly following one of the described models or reusing certain concepts. In this section, we shall describe the main models and languages of a policy expression.

RBAC [Role Based Access Control] is a role-based access control model that simplifies the writing and management of a security policy. The administration of a mandatory access control policy involves the management of multiple access rules. Yet, this process is time-consuming and subject to errors. With RBAC, rules can be simplified by expressing them according to the role of the subject and not his or her identity.

Thus, within an organization, roles are defined from the functions of the different positions and permissions are assigned to these roles. Users have roles that allow them to obtain permission. Considering that the roles are not directly assigned to the users, their management is facilitated. For example, when adding a user, it is necessary to assign the corresponding roles.

Different versions of RBAC have been defined [34,35]:

- Core RBAC understands the basic concepts of RBAC and specifies that user-role and role-permission associations are of many-to-many types;
- Hierarchical RBAC adds inheritance support between roles: for example, a junior employee has a junior role with permissions and the senior role inherits permissions from the junior role;
- Constrained RBAC introduces constraints to apply the privilege separation principle. Since 2004, RBAC is a NIST standard [36].

OrBAC [37] (Organization Based Access Control) is an access control model based on the concept of organization. OrBAC abstracts the notions of subject, action, and object by those of role (as in the case of RBAC), activity and view. Subsequently, this makes it possible to group entities according to the security rules that concern them, and thus simplify the expression of the policy. In order to facilitate the expression of dynamic policies, OrBAC uses contexts. Three types of rules are possible: First, permissions which allow a role to perform an activity on a view, in a given context. Second, bans when an activity is prohibited and finally, obligations when a role must perform an activity on a view in a given context. An OrBAC rule can be expressed as follows: A role can have permission, prohibition or obligation to perform an activity on a given view when the associated context is verified.

ABAC [38] (Attribute Based Access Control) is an access control model in which rights are given to users based on attributes. Attributes can be seen as features of system elements. An attribute consists of a type (for example, a role, a project, a sensitivity level, etc.) and a value that can be single-valued or multi-valued. For example, a role identifier or the value of the sensitivity level, etc.). These attributes can be compared to each other or to fixed values, which allows defining the rules of policy. For example, a rule evaluates one or more attributes of a subject to decide whether to allow or deny access to an object. Note that ABAC can be used with the DAC, MAC and RBAC models [39].

XACML [40] (eXtensible Access Control Markup Language) is a standard of the OASIS consortium [41]. It is both an access control policy definition language and a template for interpreting (allowing or disallowing) access requests based on this policy. XACML is based on the XML language. A set of XACML policies consists of policies, which are themselves composed of a set of elements: target, rules, a combination algorithm, and obligations. The target element indicates whether a policy should be applied to a given query. The rules specify the allowed accesses: they consist of a condition that is to say a Boolean expression. If this condition is verified, it causes an effect (authorization or prohibition) on the access. The rule combination algorithm defines how the policy should be interpreted when multiple rules may apply, for example, "permission overrides prohibition". Obligations are optional and allow you to perform an action when a rule is encountered, for instance, generate an alert. However, because of the flexibility and expressiveness of XACML, the definition of security policies can be complex [42]. XACML cannot easily allow the cloud user to express his own security needs.

Ponder [43] is a specification language for security and administration policies for distributed systems. It is a declarative and object-oriented language that supports different types of policies: authorization policies (specifying allowed or forbidden access for a subject), obligation (actions that a subject must perform in response to an event), restriction (actions that a subject should not perform), delegation (actions that a subject may delegate to another subject), constraint (to limit the application of other policies, for example in function of time). Meta-policies can also be defined to manage interactions between policies. Ponder2 [44] reuses Ponder's concepts by adapting them to autonomous systems. Hence, Ponder2 is based on a decentralized architecture, made up of self-managed components (SMCs) capable of interpreting and applying the policy on system objects. The objects considered by Ponder2 are Java objects that can communicate with SMCs. These objects must be adapted to be managed by Ponder2.

3.2.3. Expression language of insurance policies

Security policies may be supplemented by insurance policies. An insurance policy must express methods for assessing the level of protection of a system.

XCCDF [45] is an XML-based standard for specifying security checks and performance tests. Hence, it is, a language used for defining an insurance and verification policy. Besides, XCCDF can be used to automate vulnerability checks and responses when these vulnerabilities are detected. XCCDF is used by SCAP [46] (Security Content Automation Protocol), which is a set of specifications defined by NIST to standardize the format in which software vulnerabilities and security configurations are expressed. A distributed version of XCCDF, called DXCCDF [47], is used to express distributed vulnerabilities.

OVAL [48] is another standard aimed at unifying the expression of insurance policies. OVAL uses XML to evaluate the state of a system by performing a series of tests on the machine. The evaluation of the state of the system with OVAL is broken down into three stages: the representation of the system information, the description of the different states and the transmission of the results of the evaluation. The distributed version of OVAL is named DOVAL [49] and can handle distributed vulnerabilities.

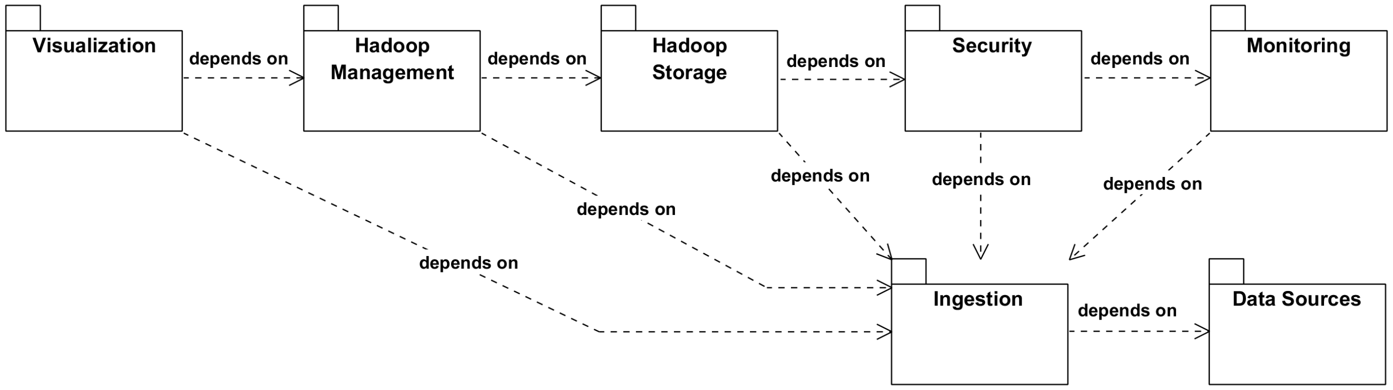
A-PPL [50] is a policy language for expressing accountability obligations. A-PPL can be used for privacy management policies, access control, and usage control policies. Besides, A-PPL manages elements specific to the insurance: it makes it possible to define alerts in the event of a detected error and determine rules of localization of the data. It can also specify which are the characteristics wanted for the audit and the storing error messages.

4. Meta-model for Security layer:

Big Data solutions are more and more used. Indeed, many providers of Big Data solutions have already proposed distributions like HortonWorks, Cloudera, MapR, etc. Although each of these distributions has its vision for a Big Data system, they all share their necessary needs of the security layer. Accordingly, Security is a crucial element for both customers and service providers.

However, before presenting the Security layer, we deem it necessary to talk firstly about our latest contribution of meta-models. In fact, in our previous work, we have proposed meta-models for layers: Data Sources, Ingestion, Hadoop Storage [51], Hadoop Platform Management [52], and visualization of Big Data architecture. Accordingly, we realized that there is a direct relationship between the security layer and the other layers in Big Data. In order to show the link between the layers of the Big Data system, we shall present in figure 1 the following meta-package diagram. It clearly indicates the meta-packages: IngestionPkg, DataSourcesPkg, HadoopPlatformManagementPkg, VizualisationPkg, SecurityPkg, and MonitoringPkg and the dependency relationship that links them.

We present in the figure 2 the meta-model that we proposed for the Security layer in Big Data. This meta-model defines specific security concepts. To separate the access control mechanisms from the rest of the concepts, we have devised our meta-model to two separate parts: the meta-model of security concepts and the meta-model that defines access control mechanisms. The purpose of this separation is to offer the possibility of extending the meta-model of access control mechanisms in our future work. It should be mentioned that this extension of the meta-model will not affect the rest of the meta-model. The figure 2 shows the meta-model of the general concepts of the security layer.



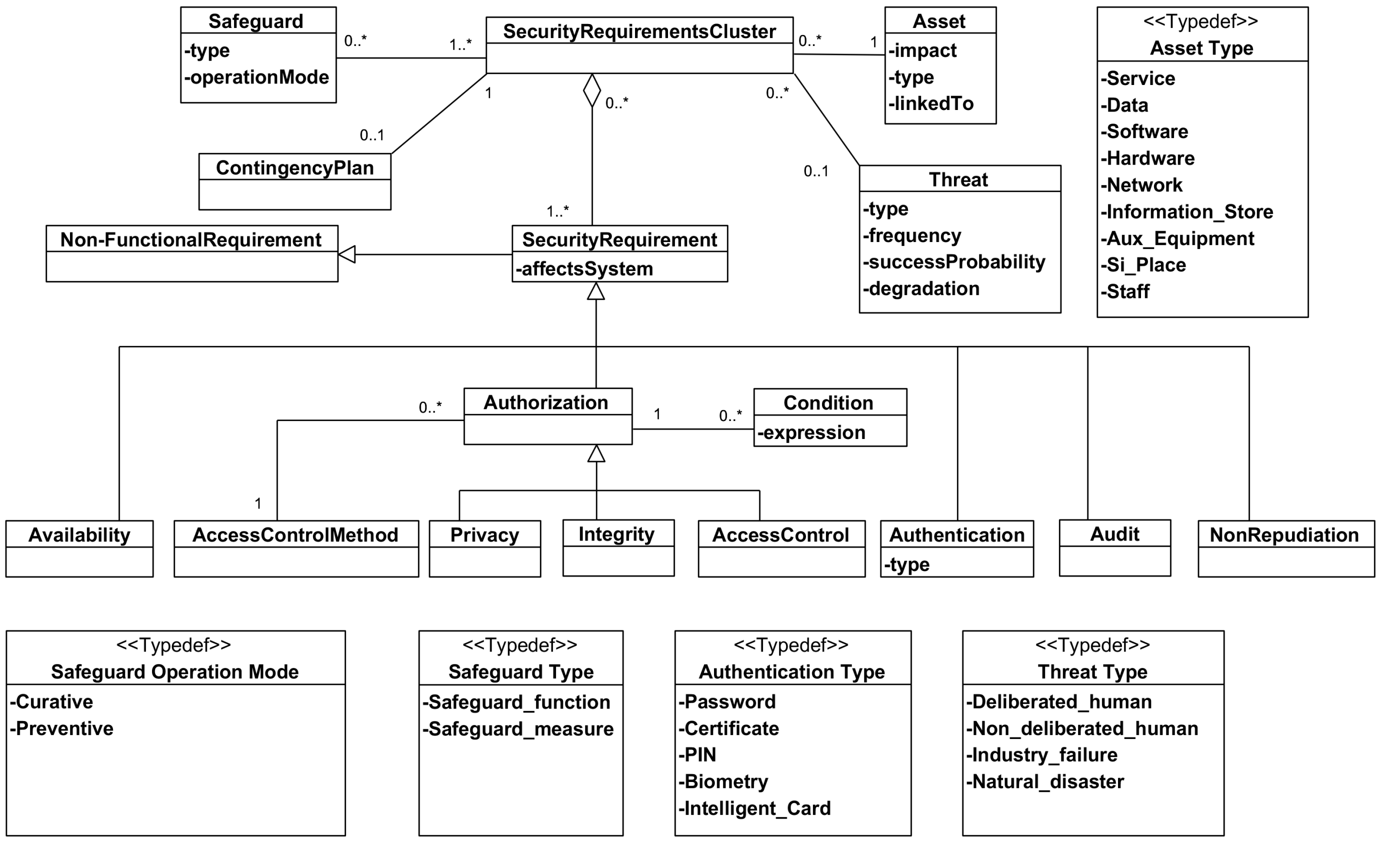


Figure. 2. Meta-model of general security concepts.

A threat can damage the assets. A threat has the following properties: type, frequency (modeled as an annual rate), probability of actual success, and degradation (that is, the level of damage to an asset if a threat reaches its goal).). Safeguards are a barrier to risk to reduce it. As indicated in the Safeguards type attribute, we can distinguish between Safeguard functions and Safeguard measures. Safeguards functions are actions that reduce risk while Safeguards' measures are physical or logical devices or processes that reduce risk. Safeguards can act in two different modes, healing if they act on damaged assets or preventative if they act before a threat has appeared. A detailed emergency plan which consists of a set of safeguards is recommended to reduce a threat that may cause harm.

There is no standard classification for security requirements. According to [53], seven categories were taken into account (confidentiality, access control, integrity, authentication, non-repudiation, availability, and audit). Integrity is the guarantee that the information remains complete and correct. Access control is employed to force authorized users to access an asset. Confidentiality is to ensure that only authorized information can be read by those who are authorized. These three types of requirements are related to authentication and may have a condition that is defined as an expression. Authentication is a procedure by which a computer system certifies the identity of a person or a computer. The purpose of this procedure is to allow the person to access certain secure resources. It will compare the information of authorized users stored in a database (locally or on an authentication server) to those provided. Access will only be allowed if the information is the same. It is the administrator of the information system who grants the rights and sets the access. The user with an access account (ID + password) will only have access to the resources he is allowed to see. Availability maintains the proper functioning of the information system and ensures access to a service or resources. Finally, the audit assesses the IT risks of physical security, logical security, change management, emergency plan, etc. It also assesses a set of IT processes - which is usually the case - to respond to a specific customer request. All of these kinds of requirements can affect particular assets or the entire system.

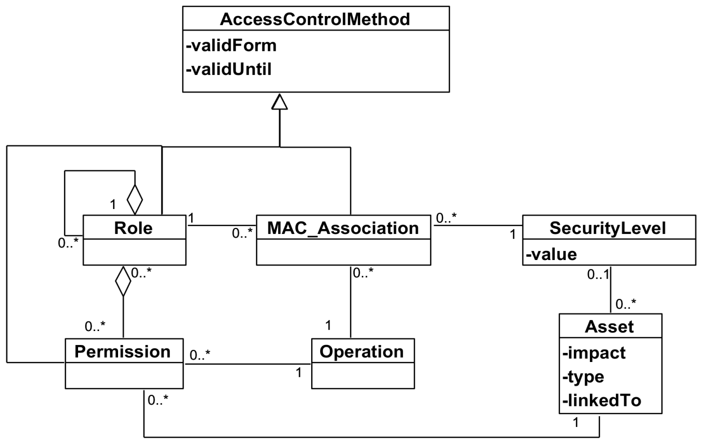


Figure. 3. Meta-model of access control mechanisms.

These two meta-models that we have proposed show the most important attributes, concepts, and data types that a Big Data system would need. Indeed, the meta-class of non-functional requirements is the extension point of the main meta-model and has the ability to add new non-functional requirements. The security requirement is a non-functional requirement specialization intended to be the root of meta-classes representing security concepts.

5. Transformations

The meta-models that we have defined for the different Big Data layers represent the platform-independent model (PIM) level, which means a model-independent of the platform. The goal of these meta-models is to standardize concepts at the Big Data level and to create an independent meta-modeling of platforms and solutions. Following that, we made some transformations using the Atlas Transformation Language (ATL). This transformation language will allow us to move from the PIM model to the Platform Specific Model (PSM). PSMs can use domain-specific languages or general languages like Java, C #, Python, and so on. The techniques used in the MDA approach are thus mainly modeling techniques and model transformation techniques. The figure below shows the transformations that we have made in this paper to make the transition from our meta-model of the security layer at the level of Big Data to PSM:

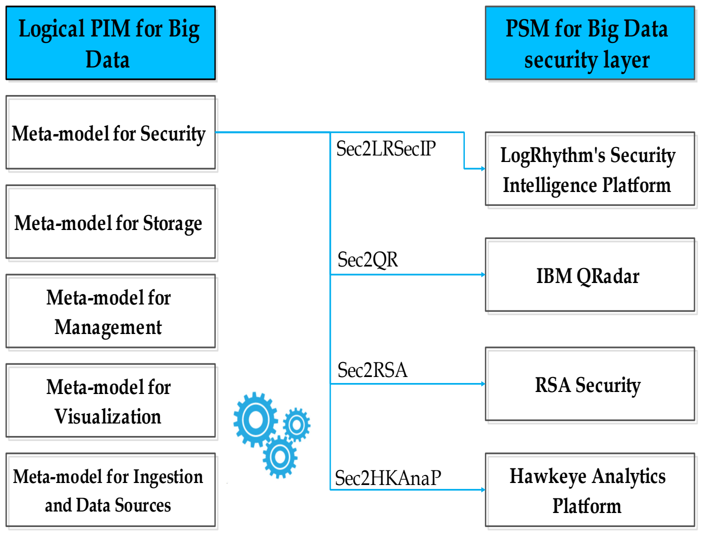

Logical PIM for Big Data
PSM for Big Data security layer
Meta-model for Security
Meta-model for Storage
Meta-model for Management
Meta-model for Visualization
Meta-model for Ingestion and Data Sources
Sec2LRSecIP
Sec2QR
Sec2RSA
Sec2HKAnaP
LogRhythm's Security Intelligence Platform
IBM QRadar
RSA Security
Hawkeye Analytics Platform

Figure. 4. Transformation of security meta-model to PSM.

6.1. Configuration

In this section, we shall discuss the techniques that we have used to implement the approach presented in Figure 4. Version 4.12 of the Eclipse IDE was used, with the addition of the Eclipse Modeling Framework (EMF) to draw the proposed meta-models. We also used version 4.1 of the ATL transformation language on

Eclipse 4.12, to define the transformation rules and to create our four transformations: Sec2LRSecIP, Sec2QR, Sec2RSA, and Sec2HKAnaP. The switch to the PSM did not take into consideration the version of the chosen solutions since the meta-model we proposed is a standard for the security layer. After applying the transformation rules, the result found will be used on all versions of LogRhythm's Security Intelligence Platform, IBM QRadar, RSA Security, and Hawkeye Analytics Platform.

6.2. Experiences

While performing our tests, we used two datasets to better measure the transformation execution time to the four chosen solutions: LogRhythm's Security Intelligence Platform, IBM QRadar, RSA Security, and Hawkeye Analytics Platform. The results are shown in Figures 5 and 6.

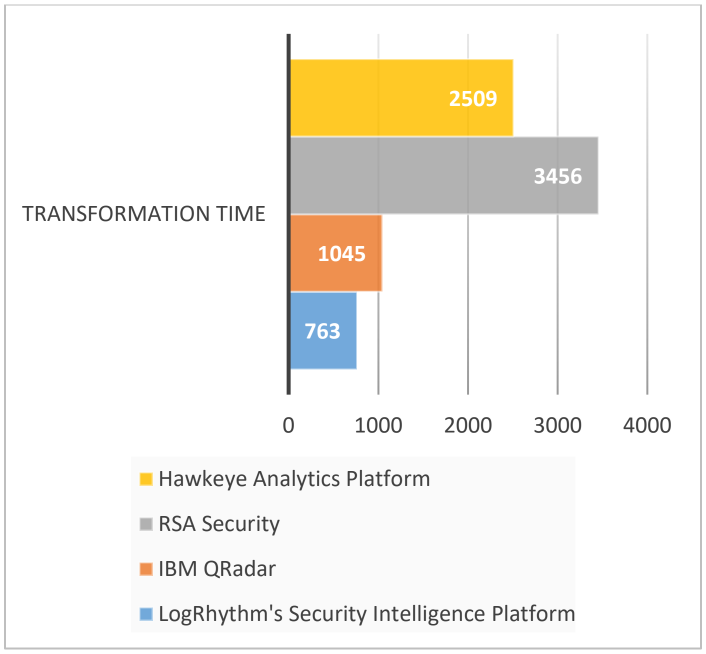


Figure. 5. Transformations time.

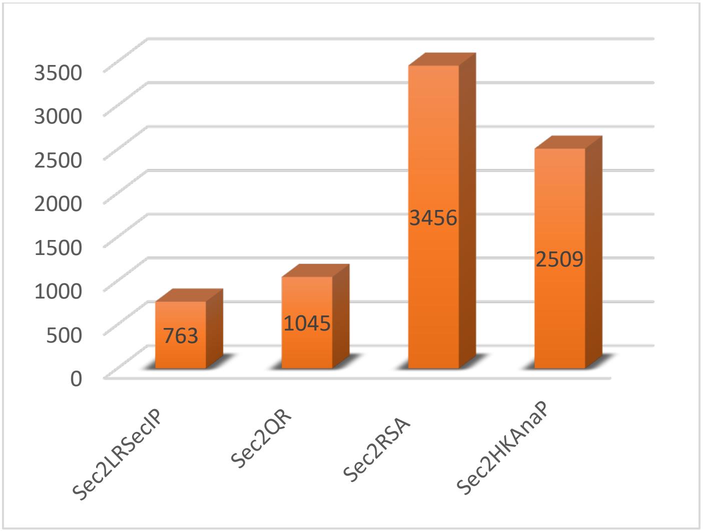


Figure. 6. Execution time for ATL transformations.

Table 1: Transformations runtime.

Security solution	LogRhythm's Security Intelligence Platform	IBM QRadar	RSA Security	Hawkeye Analytics Platform
Transformation time	763	1045	3456	2509

7. Discussion and Perspectives

The security properties that were presented in the first section are textual and abstract. They allow a user literally to express his security needs. Although these properties are high-level requirements, they cannot be directly applied to the system. This is why different models and security languages are designed to express these properties so that they can be applied to a system.

In the second section, we presented historical models for access control policies and security and insurance policy expression languages. Indeed, Historical access control models introduce concepts. These concepts can be generalized to security policies that are not dedicated to access control. For example, modeling in subject-permission-object triplet form is not necessarily specific to access control. Besides, access control has shown the value of a mandatory security policy compared to a discretionary policy. In fact, a discretionary policy gives the possibility to guarantee security properties [56,57]. Existing policy models and languages also make it possible to introduce essential notions into the expression languages of security and insurance policies. Thus, the notion of role introduced in RBAC is used in other models of security policies, such as OrBAC and XACML. It can simplify the expression and administration of policy. These languages allow the definition of security needs in the form of properties. However, these policies are often restricted to the expression of a property and support only a subset of properties. Moreover, existing languages are mostly not suitable for defining security and assurance properties for cloud and big data environments.

As we have mentioned before, Big Data has become essential in today's world. Indeed, our analysis of the subject leads us to find that there are several solutions to manage Big Data in the market (HortonWorks, Pivotal HD, IBM InfoSphere BigInsights, etc.). Each of these solutions manages Big Data in its own way without relying on standards like meta-models. This of course results in the diversity of solutions and the non-interoperability between the different solutions. During our research project, we have worked on the meta-modeling of the different layers of a Big Data system [58]. In this paper, we rely on comparative studies that we have already done to draw the key concepts of the security layer. The meta-model that we proposed for this layer and the other layers is platform-independent according to Model-Driven Architecture pattern. By using the ATL transformation language, for example, we can move to the other platform-specific models (PSM) [59,60]. The PSM we have chosen in this work are LogRhythm's Security Intelligence Platform, IBM QRadar, RSA Security, and Hawkeye Analytics Platform. After the creation of these meta-models, in the next step, we shall work on the creation of models respecting these meta-models. Then we shall define the transformation rules between these meta-models using the transformation language ATL (Atlas Transformation Language). These meta-models are platform-independent according to Model Driven Architecture pattern, which describes the structures of Big Data layers independently from any specific platform.

8. Conclusion

Big data refers to the collection and aggregation of large amounts of data from different sources to extract information through statistical, descriptive and predictive analysis. Several new solutions exist in the IT market that can handle this huge amount of data. This big data trend makes security an essential but complex point to address. The definition of security needs can indeed be a difficult task, especially since the user of the service does not necessarily know these needs. However, we have seen that there are methods of risk analysis that allow a user to determine their security needs. Thus, we have presented in this paper the different basic security properties that may correspond to the needs of a user, as well as several models of policies to express these needs. After that, we have defined a standard meta-model for the security layer that represents the PIM level. Finally, we made transformations to move from our meta-model to four PSM.

References

[1] N. Sawant and H. (Software engineer) Shah, Big data application architecture Q & A a problem-solution approach. Apress, 2013.

[2] Erraissi, A., & Belangour, A. Data sources and ingestion big data layers: meta-modeling of key concepts and features. International Journal of Engineering & Technology, 7(4), 3607–3612, 2018. https://doi.org/10.14419/ijet.v7i4.21742

[3] Erraissi, Allae, and Abdessamad Belangour. « Hadoop Storage Big Data Layer: Meta-Modeling of Key Concepts and Features ». International Journal of Advanced Trends in Computer Science and Engineering 8, nº 3, 646-53, 2019. https://doi.org/10.30534/ijatcse/2019/49832019

[4] Erraissi, Allae, and Abdessamad Belangour. « Meta-Modeling of Big Data Management Layer ». International Journal of Emerging Trends in Engineering Research 7, no 7, 36-43, 2019. https://doi.org/10.30534/ijeter/2019/01772019.

[5] Erraissi, Allae, and Abdessamad Belangour. « Meta-Modeling of Big Data Visualization Layer Using On-Line Analytical Processing (OLAP) ». International Journal of Advanced Trends in Computer Science and Engineering 8, nº 4, 990-98, 2019. https://doi.org/10.30534/ijatcse/2019/02842019.

[6] Allae Erraissi, Abdessamad Belangour, and Abderrahim Tragha, "Digging into Hadoop-based Big Data Architectures," Int. J. Comput. Sci. Issues IJCSI, vol. 14, no. 6, pp. 52–59, Nov. 2017. https://doi.org/10.20943/01201706.5259

[7] Kleppe, A.G., Warmer, J., Warmer, J.B. and Bast, W., 2003. MDA explained: the model driven architecture: practice and promise. Addison-Wesley Professional, 2003.

[8] Jurjens, J. Secure Systems Development with UML. Springer Verlag, 2003.

[9] Basin, D., Doser, J., and Lodderstedt, T. Model driven security: From UML models to access control infrastructures. ACM Trans. Softw. Eng. Methodol., 15(1):39–91, 2006. https://doi.org/10.1145/1125808.1125810

[10] Reznik, J., Ritter, T., Schreiner, R., and Lang, U. Model driven development of security aspects. ENCTS, 163(2):65–79, 2007. https://doi.org/10.1016/j.entcs.2006.10.016

[11] Lang, U. and Schreiner, R. Managing business compliance using model-driven security management. In Securing Electronic Business Processes, pages 1–11, 2008. https://doi.org/10.1007/978-3-8348-9283-6_24

[12] Bresciani, P., Mouratidis, H., and Zanone, N. Modelling security and trust with secure tropos. Integrating Security and Software Engineering: Advances and Future Visions, pages 160–189, 2007. https://doi.org/10.4018/978-1-59904-147-6.ch008

[13] Van Lamsweerde, A. Elaborating security requirements by construction of intentional anti-models. In ICSE'04: 26th Int. Conf. on Software Engineering, pages 148–157. IEEE Computer Society, 2004. https://doi.org/10.1109/ICSE.2004.1317437

[14] Yu, E., Liu, L., and Mylopoulos, J. A social ontology for integrating security and software engineering. Integrating Security and Software Engineering: Advances and Future Visions, pages 70–109, 2007. https://doi.org/10.4018/9781605660608.ch048

[15] Haley, C., Laney, R., Moffett, J., and Nuseibeh, B. Security requirements engineering: A framework for representation and analysis. IEEE Trans. Software Eng., 34(1):133–153, 2008. https://doi.org/10.1109/TSE.2007.70754

[16] C.Jahl, ITSEC. Information Technology Security Evaluation Criteria (ITSEC) v1.2. Technical report, 1991. https://doi.org/10.1109/ICSE.1991.130656

[17] Tcsec, D. Trusted computer system evaluation criteria. DoD 5200.28-STD, 83, 1985.

[18] Bishop, M. What is computer security? Security & Privacy, IEEE, 1(1):67–69, 2003. https://doi.org/10.1109/MSECP.2003.1176998

[19] Stoneburner, G. Underlying technical models for information technology security: recommendation of the National Institute of Standards and Technology. US Department of Commerce, Computer Security Division, Information Technology, National Institute of Standards and Technology, 2001.

[20] Lampson, B. W. A note on the confinement problem. Communications of the ACM, 16(10):613–615, 1973. https://doi.org/10.1145/362375.362389

[21] Burr, W. E., Dodson, D. F. et Polk, W. T. Electronic authentication guideline: Recommendations of the national institute of standards and technology. NIST Special Publication, pages 800–63, 2013. https://doi.org/10.6028/NIST.SP.800-63-2

[22] Lampson, B. W. Dynamic protection structures. In Proceedings of the November 18-20, 1969, fall joint computer conference, pages 27–38. ACM, 1969. https://doi.org/10.1145/1478559.1478563

[23] Lampson, B. Protection. In Proc. 5th Princeton Conf. on Information Sciences and Systems, pages 18–24. Princeton, 1971. https://doi.org/10.1145/775265.775268

[24] Harrison, M. A., Ruzzo, W. L. et Ullman, J. D. Protection in operating systems. Communications of the ACM, 19(8):461–471, 1976. https://doi.org/10.1145/360303.360333

[25] Sandhu, R. S. The typed access matrix model. In Research in Security and Privacy. Proceedings. 1992 IEEE Computer Society Symposium on, pages 122–136. IEEE, 1992. https://doi.org/10.1109/RISP.1992.213266

[26] Soshi, M., Maekawa, M. et Okamoto, E. The dynamictyped access matrix model and decidability of the safety problem. IEICE transactions on fundamentals of electronics, communications and computer sciences, 87(1):190–203, 2004. https://doi.org/10.1007/10722599_7

[27] Sandhu, R. S. The schematic protection model: its definition and analysis for acyclic attenuating schemes. Journal of the ACM (JACM), 35(2):404–432, 1988. https://doi.org/10.1145/42282.42286

[28] Ferraiolo, D. et Kuhn, R. Role-based access control. In In 15th NIST-NCSC National Computer Security Conference, (1992). https://doi.org/10.1016/S0065-2458(08)60206-5

[29] Loscocco, P. A., Smalley, S. D., Muckelbauer, P. A., Taylor, R. C., Turner, S. J. et Farrell, J. F. The inevitability of failure: The flawed assumption of security in modern computing environments. In Proceedings of the 21st National Information Systems Security Conference, volume 10, pages 303–314, 1998.

[30] Anderson, J. P. Computer security threat monitoring and surveillance. Rapport technique, Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, 1980.

[31] Bell, D. E. et LaPadula, L. J. Secure computer systems: Mathematical foundations. Rapport technique, DTIC Document, 1973.

[32] Biba, K. J. Integrity considerations for secure computer systems. Rapport technique, DTIC Document, 1977.

[33] Boebert, W. E. et Kain, R. Y. A practical alternative to hierarchical integrity policies. NIST SPECIAL PUBLICATION SP, pages A–10, 1989.

[34] Sandhu, R. S., Coyne, E. J., Feinstein, H. L. et Youman, C. E. Role-based access control models. Computer, (2):38–47, 1996. https://doi.org/10.1109/2.485845

[35] Sandhu, R., Ferraiolo, D. et Kuhn, R. The nist model for role-based access control: towards a unified standard. In ACM workshop on Role-based access control, volume 2000. https://doi.org/10.1145/344287.344301

[36] Standard, R. Incits 359-2004. ANSI INCITS, pages 359–2004, 2004.

[37] Kalam, A. A. E., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C. et Trouessin, G. Organization based access control. In Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on, pages 120–131. IEEE, 2003. https://doi.org/10.1109/POLICY.2003.1206966

[38] Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R. et Scarfone, K. Guide to attribute based access control (abac) definition and considerations. NIST Special Publication, 800:162, 2014. https://doi.org/10.6028/NIST.SP.800-162

[39] Jin, X., Krishnan, R. et Sandhu, R. S. A unified attributebased access control model covering dac, mac and rbac. DBSec, 12:41–55, 2012. https://doi.org/10.1007/978-3-642-31540-4_4

[40] Moses, T. et al. Extensible access control markup language (xacml) version 2.0. Oasis Standard, 200502, 2005.

[41] OASIS. OASIS. https://www.oasis-open.org/. 2015.

[42] Hu, V. C., Martin, E., Hwang, J. et Xie, T. Conformance checking of access control policies specified in xacml. In Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International, volume 2, pages 275–280. IEEE, 2007. https://doi.org/10.1109/COMPSAC.2007.96

[43] Damianou, N., Dulay, N., Lupu, E. et Sloman, M. A language for specifying security and management policies for distributed systems. London: Department of Computing, Imperial College, Tech. Rep, 2000.

[44] Twidle, K., Dulay, N., Lupu, E. et Sloman, M. Ponder2: A policy system for autonomous pervasive environments. In Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on, pages 330–335. IEEE, 2009. https://doi.org/10.1109/ICAS.2009.42

[45] Waltermire, D., Schmidt, C., Scarfone, K. et Ziring, N. Specification for the extensible configuration checklist description format (xccdf) version 1.2 (draft). (2011b).

[46] Waltermire, D., Quinn, S., Scarfone, K. et Halbardier, A. The technical specification for the Security Content Automation Protocol (SCAP). NIST Special Publication, 800:126. (2011a).

[47] Barrère, M., Badonnel, R. et Festor, O. Collaborative remediation of configuration vulnerabilities in autonomic networks and systems. In Proceedings of the 8th International Conference on Network and Service Management, pages 357–363. International Federation for Information Processing. (2012a).

[48] OVAL Oval language. Http://oval.mitre.org/. 2014.

[49] Barrère, M., Badonnel, R. et Festor, O. Towards the assessment of distributed vulnerabilities in autonomic networks and systems. In Network Operations and Management Symposium (NOMS), 2012 IEEE, pages 335–342. IEEE. (2012b). https://doi.org/10.1109/NOMS.2012.6211916

[50] Azraoui, M., Elkhiyaoui, K., Önen, M., Bernsmed, K., De Oliveira, A. S. et Sendor, J. A-ppl : An accountability policy language. In Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance, pages 319–326. Springer, 2015. https://doi.org/10.1007/978-3-319-17016-9

[51] A., Erraissi A., Belangour A. Capturing Hadoop Storage Big Data Layer Meta-Concepts. In: Ezziyyani M. (eds) Advanced Intelligent Systems for Sustainable Development (AI2SD'2018). AI2SD 2018. Advances in Intelligent Systems and Computing, vol 915. Springer, Cham, 2019. https://doi.org/10.1007/978-3-030-11928-7_37

[52] A. Erraissi and A. Belangour, "Meta-modeling of Zookeeper and MapReduce processing," 2018 International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), Kenitra, Morocco, pp. 1-5, 2018. https://doi.org/10.1109/ICECOCS.2018.8610630

[53] Rodriguez, A., Fern´andez-Medina, E., and Piattini, M. A bpmn extension for the modeling of security requirements in business processes. IEICE Transactions, 90-D(4):745–752, 2007. https://doi.org/10.1093/ietisy/e90-d.4.745

[54] Mellado, D., Fernandez-Medina, E., and Piattini, M. A common criteria based security requirements engineering process for the development of secure information systems. Comput. Stand. Interfaces, 29(2):244–253, 2007. https://doi.org/10.1016/j.csi.2006.04.002

[55] Samarati, P. and Capitani, S. D. Access control: Policies, models, and mechanisms. In FOSAD, pages 137–196, 2000. https://doi.org/10.1007/3-540-45608-2_3

[56] Ferraiolo, D. et Kuhn, R. Role-based access control. In In 15th NIST-NCSC National Computer Security Conference, 1992.

[57] Loscocco, P. A., Smalley, S. D., Muckelbauer, P. A., Taylor, R. C., Turner, S. J. et Farrell, J. F. (1998). The inevitability of failure: The flawed assumption of security in modern computing environments. In Proceedings of the 21st National Information Systems Security Conference, volume 10, pages 303–314.

[58] A. Erraissi, B. Mouad and A. Belangour, "A Big Data visualization layer meta-model proposition," 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO), Manama, Bahrain, 2019, pp. 1-5. doi: 10.1109/ICMSAO.2019.8880276

[59] M. Banane, A. Erraissi and A. Belangour, "SPARQL2Hive: An approach to processing SPARQL queries on Hive based on meta-models," 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO), Manama, Bahrain, 2019, pp. 1-5. doi: 10.1109/ICMSAO.2019.8880393

[60] Fatima Kalna, Allae Erraissi, Mouad Banane, Belangour "A Scalable Business Intelligence Decision-Making System in The Era of Big Data" International Journal of Innovative Technology and Exploring Engineering 2019. https://doi.org/10.35940/ijitee.L3251.1081219