From Access Control Models to Access Control Metamodels: A Survey

Chapter · January 2020

DOI: 10.1007/978-3-030-12385-7_61

3 authors, including:

Mehdi Adda
Université du Québec à Rimouski UQAR
52 PUBLICATIONS **184** CITATIONS

SEE PROFILE

From Access Control Models to Access Control Metamodels: A Survey

Nadine Kashmar[1], Mehdi Adda[2] and Mirna Atieh[3]

[1,2] Université du Québec à Rimouski, Rimouski QC G5L 3A1, Canada
kasn0002@uqar.ca[1], mehdi_adda@uqar.ca[2]
[3] Lebanese University, Hadat, Lebanon
matieh@ul.edu.lb

Abstract. Access control (AC) is a computer security requirement used to control, in a computing environment, what the user can access, when and how. Policy administration is an essential feature of an AC system. As the number of computers are in hundreds of millions, and due to the different organization requirements, applications and needs, various AC models are presented in literature, such as: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC), etc. These models are used to implement organizational policies that prevent the unauthorized disclosure of sensitive data, protecting the data integrity, and enabling secure access and sharing of information. Each AC model has its own methods for making AC decisions and policy enforcement. However, due to the diversity of AC models and the various concerns and restrictions, its essential to find AC metamodels with higher level of abstraction. Access control metamodels serve as a unifying framework for specifying any AC policy and should ease the migration from an AC model to another. This study reviews existing works on metamodels descriptions and representations. But, are the presented metamodels sufficient to handle the needed target of controlling access especially in the presence of the current information technologies? Do they encompass all features of other AC models? In this paper we are presenting a survey on AC metamodels.

Keywords: Metamodel, model, access control, policy, security.

1 Introduction

As long as there is an increase and development in the use for internet services, there is also an increase and a critical need for computer security solutions [1, 2]. Computer security and the related topics were and still the main issues in the world of Information Technology (IT).

Over the years until today, IT security and privacy are critical concerns for academic, economic, social, industrial, and governmental organizations. Access Control (AC) is one of the most important and critical aspects of IT security. For this purpose, different AC models and policies are developed. In literature, various AC

models are proposed, such as: Discretionary Access Control (DAC) [3, 4], Mandatory Access Control (MAC) [3, 5, 6], Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC) [3, 4, 5, 7], and Organization Based Access Control (OrBAC) [5]. Each AC model is developed either to overcome limitations found in previous models or as a solution for a specific use case and application.

Finding a unified AC model becomes a significant issue due to the need to include all features offered by the existing AC models, which in some cases, are incompatible and irreconcilable. Also, due to the widespread and heterogeneity of interconnected networks and distributed systems, heterogeneity of platforms and applications, and due to diversity of users, the necessity to design a well coherent AC architecture for enterprises becomes a must. In this context, different AC metamodels, to address these issues, are presented in literature to serve as a unifying framework for specifying and enforcing different AC policies. In the following sections we will summarize existing AC metamodels. Nevertheless, before describing and comparing these metamodels, we have to explore some basic concepts related to AC.

The remaining of this paper is organized as follows: Section II summarizes the existing AC models, their advantages and limitations. The AC usage in different system levels are presented in section III. Section IV describes the concept of AC metamodels, the metamodeling tools, and provides a comprehensive overview about the existing AC metamodels. Section V presents the potential research issues. Section VI concludes this paper.

2 Access Control Models

2.1 A Brief Overview

Access control is defined as an essential security requirement in the field of IT. Each organization has its own information system where a set of policies is defined based on conditions where users can access all or some system resources. Implementing these policies is essential to protect resources. AC methods are carried out at different IT infrastructure levels. They are used in operating systems, databases, networks, and information systems. The goal is to protect files, directories, regulate access to database objects and fields, and protect applications' information (payroll processing, e-health…). etc. However, the primary objective of access controls is the fulfillment of the defined AC policies [1, 2].

Generally speaking [3, 4, 5], AC models and mechanisms are defined in terms of subjects, objects and access rights. The subject concept usually refers to a user or program; the object concept refers to an entity a user wants to have access to such as a file, a table or a class. However, a subject may or may not have an access right to an object. Access right means that a subject is able or perform an operation on an object. The operation may be read, write, execute, etc. In other words, a user (subject) can perform an operation (read, write…) on an object (file, class…) if he has a permission to do so. To carry out an operation, access rights are required. To manage who and how operations must be carried out, privileges or AC must be defined. Data resources are protected under different access policies. A model is the projection of the scope of

policies and the needed behavior between subject and object entities. Policies are a set of guidelines, which are generalized, abstracted, formally, or semi-formally described [3]. Several research surveys are presented in literature with detailed descriptions about AC models. In the following sections we summarize the concept of each model.

2.2 Discretionary Access Control (DAC)

In late 1960s, Discretionary Access Control (DAC) model was first introduced by Lampson, a member of a curriculum design team. In DAC, the system protection notion includes three major components: a set of objects, a set of domains, and a matrix. Lampson's work was then extended by Graham and Denning, where the term "subject" was included instead the domain. Then, the extended Lampson's work was developed by Harrison, Ruzzo and Ullman (HRU) to find a formal proof that tracking privilege propagation was undecidable in general [3].

DAC is defined as a user-centric AC model in the sense that a file owner determines the permissions that are assigned to other users requiring access to the file [4]. DAC mechanism allows users control the access rights (read, write…) to their files without the need of a pre-specified set of rules. The access rights are specified by Access Control Matrix (ACM), where AC rights of subject(s) over object(s) are specified. Other variations of implementing AC matrix include Capability Lists (CLs) and Access Control Lists (ACLs). In the concept of CLs the user access rights are stored by rows, whereas in ACLs the access rights for various users on a file are stored by columns. Lampson and Harrison Ruzzo Ullman (HRU) are two DAC models [3].

DAC model is very flexible to assign access rights between subjects and objects. But it also has limitations where the system maintenance and verification of security principles are extremely difficult, since users control access rights to their owned objects. Also, the possible attacks for Trojan horses [3, 5].

2.3 Mandatory Access Model (MAC)

In 1970s, Mandatory Access Control (MAC) protection was presented to include the use of a security kernel. In 1987, a paper was published in IEEE Symposium of Security and Privacy, where crucial differences between commercial and military security requirements were presented by Clark and Wilson [3].

In MAC users cannot define AC rights by themselves. The AC policy is managed in a centralized manner. MAC model is based on the concept of security levels associated with each subject and object, where permissions and actions are derived. Security classes have hierarchical and nonhierarchical components. The hierarchical components include types: unclassified (U), confidential (C), secret (S), and top-secret (TS) where $TS \geq S \geq C \geq U$, to classify subjects and objects into levels of trust and sensitivity. For objects a security level is called the classification level and for subjects it is called clearance level. The nonhierarchical component is represented by a set of categories. Security labels indicate security levels for classification of objects and clearance of subjects, and uses two security properties, simple security property and *-property. Bell and LaPadula (BLP) of multi-level security and BIBA are two

MAC variants. In BLP, a subject is allowed to read an object if the subject's clearance is ≥ than the object's classification, and to write if it is ≤. In BIBA, a subject is allowed to read an object if the subject's clearance is ≤ than the object's classification, and to write if it is ≥ [3, 5].

This model is presented to overcome the limitations of DAC model, which is the Trojan Horse attacks, by centralizing access control. In [6], it is mentioned that MAC model is relatively straightforward and is considered to be a good model for commercial systems that operate in hostile environments such as web servers and financial institutions where the risk of attack is very high. Also, it has limitations, since it is difficult to implement due to the dependence on trusted components, and the necessity for applications to be rewritten to adhere to MAC labels and properties. Similarly, the assignment of security levels by the system places limits on user actions which prevents dynamic modification of the original policies.

2.4 Role-Based Access Control (RBAC)

Based on historical practices, Role-Based Access Control (RBAC) was defined as a job a user performs, and he/she can be assigned one or more roles to indirectly associate permissions with users [3].

RBAC model is considered as an alternative approach to MAC and DAC. In RBAC, users can be assigned several roles and a role can be associated with several users [4]. A role means a collection of permissions to use objects to perform a job function that combines the authority and responsibility assigned to a subject who plays this role, e.g. accountant, director, engineer, etc. each role is associated with privileges or permissions [3]. The aim of RBAC is to facilitate the administration of the AC policy. It governs the access of a user to information through roles for which the user is authorized to perform. RBAC model is based on several entities, which are, users, roles, permissions, actions, operations, and objects. Each role can have many permissions, and permissions may be assigned to many roles. A subject can operate or play many roles and a role can be performed by different subjects [5]. Several RBAC models are proposed in the literature, Flat RBAC (RBAC0), Hierarchical RBAC (RBAC1), Constrained RBAC (RBAC2), and Symmetric RBAC (RBAC3) [5, 7].

This model has many benefits, it has central administration of role memberships and ACs. It may also be applied in distributed areas because it is based on the concept of constraints and inherence [5, 6]. In RBAC, role hierarchy specifies which roles and permissions are available to subjects based on different inheritance mechanisms. Role hierarchies and permission inheritance in RBAC models are explained in [8, 9]. Also, in distributed areas where different resources are shared among users, RBAC has powerful means of specifying AC decisions [10]. Conversely, it also has some drawbacks. It is frequently criticized for the difficulty of setting up an initial role structure and for inflexibility in rapidly changing IT technologies. For example, RBAC provide poor support for dynamic attributes such as time of day, which might be needed when determining user permission [11]. Another drawback is reflected in large systems, where role inheritance and the need for customized privileges make administration potentially heavy [6].

2.5 Organization Based Access Control (OrBAC)

Organization Based Access Control (OrBAC) is first presented in 2003. The aim of this model is to solve some problems in the previous AC models (DAC, MAC and RBAC), to find a more abstract control policy. It is designed to address the subject, object and action, in such a way that the policy determines what subject(s) has some action(s) to access some object(s). Each organization (clinic, banks, hospitals…) is comprised of a structured group of subjects having certain roles, or entities. This model exceeds the concept of only granting permissions to subjects, it also addresses the concept of prohibitions, obligations and recommendations [5]. A role may have a permission, prohibition or obligation to do some activity on some view given an associated context. In this model seven entities are defined, a) the abstract level or organizational (1- Role, 2- Activity, 3- View) and b) the concrete level (4- Subject, 5- Action, 6- Object). The seventh entity is Context lies between the two levels to have a correspondence between the elements of each level. The context is presented in OrBAC to express dynamic rules for relations between entities, for example, Permission, Prohibition, Isprohibited, Recommendation, Ispermitted, Isobligatory, Isrecommended, Obligation [5, 12].

The OrBAC has an advantage in eliminating conflicts between security rules. It also has some vulnerabilities to some kinds of attacks, e.g. covert channels [5].

2.6 Attribute-Based Access Control (ABAC)

Attribute Based Access Control (ABAC) concepts have paralleled that of RBAC. It has some advantages over RBAC, because of its benefits in authorization management and its ability to support dynamic attributes. The main idea behind ABAC is to grant or deny user requests based on arbitrary attributes of the user and selected attributes of the object that may be globally recognized [3, 11]. It enables precise AC which allows a higher number of discrete inputs into an AC decision. This provides a larger set of possible combinations of variables to reflect a larger set of possible rules to express policies [8]. Recently, this model gained attention from businesses, academia and organizations due to the limitations in RBAC model. Two standards that widely address the ABAC framework are: The Extensible AC Markup Language (XACML) and Next Generation AC (NGAC) with AC facility for applications and other important features [3].

In ABAC, subjects are enabled to access a wider range of objects without specifying individual relationships between each subject and each object. As well as, there are three types of attributes: subject, object and environmental attributes. The first two are common to all ABAC models. The third type used in some models that depend on the availability of system sensors that can detect and report values, they may include the current time or day of the week. Despite the benefits of ABAC model over the other models and its flexibility to assign policies and security features, it has a number of limitations such as [2]:

- The problem of determining the current set of permissions available to all users.
- Its implementations require significant time to run.

- It is often not possible to compute the set of users that may have access to a given resource.
- It is difficult to efficiently calculate the resulting set of permissions for a given user as all objects would need to be checked against all relevant policies.

The historical AC models are summarized to provide an overview of their concepts, benefits and limitations. This overview gives a conception about creating new AC models, combining some of their features, or finding models with higher level of abstraction (metamodels).

3 Access Control Usage

This section explains how AC models are generally integrated in any Information System (IS). In general, the IS is composed of six components: hardware, software, data, procedures, people and communication. IT security concepts are related to each component of any IS. For a secure environment, security must be carefully managed. Moreover, any connected computer to the internet is vulnerable to attacks. Thus, the IS components are also under threat. Subsequently, various AC models and principles are deployed to protect the IS environment. Their function is to control which object have access to which subject in the system (files to read, programs to execute, data to share, etc.). Fig. 1 shows AC at different levels in a system. Applications may be written on top of middleware, such as a database management system. The middleware use services provided by the underlying operating system. Also, the operating system of ACs usually relies on hardware features provided by the processor or by associated memory management hardware [13].

The IT security requirements are widely explained in literature. Some examples of these requirements are: protection from improper access, user authentication, data integrity, etc. In the following sections we will present AC usage in operating systems, databases, networks, cloud computing, and Internet of Things (IoT).

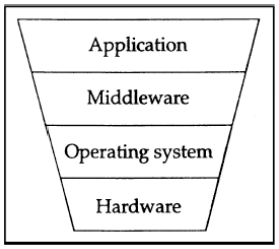

Fig. 1. Access control levels in a system [13].

3.1 Access Control in Operating Systems

Access control mechanisms are provided with Operating Systems (OSs) to authenticate system administrators and users using some procedures, for example, passwords. After authentication, AC play an important role in allowing users to access files, communications ports, and other system resources. User permissions are modeled as a

matrix of access permissions, where columns represent files/folders and rows represent users. In this context, a file owner determines the permissions that are assigned to other users requiring access to the file. However, only these users may have some permissions (privileges), on the file, to read, write, execute, etc. An ACM is used to show the users' access rights on a system and the files in it [4]. Although ACMs can be used to implement protection mechanisms, they don't scale well with many users. For example, in an institution with a large number of users and applications, a plenty of entries will occur and this will cause a performance problem and administrative mistakes. Two practical ways to overcome those issues are presented in [13]. Firstly, by using groups or roles to manage the privileges of large sets of users simultaneously. Secondly, by storing the ACM either by columns (ACL) or rows (CLs).

3.2 Access Control in Databases

Today, it is common for institutions to have databases (DBs) with critical data, such as: bank accounts, employment records, hospital reports, etc. Thus, the security needs become very critical, especially for online users. DB management systems (Oracle, SQL…) have their control mechanisms and users should have different types of permissions based on their job functions. In this context, OS play an important role to separate the DB from other applications running on the same computer by identifying users. Besides, ACLs and CLs are mixed to provide the needed mechanisms for DBs [13]. Database users should have different types of permissions based on their job functions. Each user should only have the permissions which are granted for him after his login to the system, which is known as authorization. These permissions are assigned to determine the user actions within the DB.

Different DB tasks and maintenance procedures must be occurred, these tasks must be implemented by DB administrators. These tasks include creating DBs, removing unneeded DBs, managing disk space allocation, monitoring performance, and performing backup and recovery operations. DB platforms allow default system administrators to perform such tasks and delegate permissions to other users [14].

3.3 Access Control in Networks

Network (NW) security is also an essential part of IS. In this domain, different issues must be taken into consideration, which are: the NW design, NW device security, firewalls, virtual private NWs, Intrusion Detection and Prevention Systems (IDPS), etc. The section below describes and summarizes these considerations [15].

The requirements of NW security and budgets are specified based on NW design. For this, different aspects must be taken into consideration in designing NWs, such as: availability, cost, performance, number of users, etc. In security, it is important to enable effective and secure links to other NWs, provide a platform that is helpful for securing sensitive NW assets, and identify critical security controls and understand the consequences of a failure of these controls. Also, NW device security concern is how to use routers and switches to increase the security of the NW. The internetworking protocol in use today is known as Transmission Control Protocol/Internet Protocol (TCP/IP). It is a suite of protocols and applications that have discrete functions that map to the Open Systems Interconnection (OSI) model.

Each connected device on a NW has two NW addresses: The Media Access Control (MAC) address, and the IP address. However, there are several configuration steps to configure the device (router, switch…) for increased security. The various steps are: switch security practices, ACLs, administrative practices, Internet Control Message Protocol (ICMP), logging to routers, etc. Furthermore, firewalls are defined as the first line of defense between the internal NW and untrusted NWs like the Internet. They play an important role in controlling application communication and other functions, such as: Network Address Translation (NAT), antivirus, e-mail (spam) filtering, IDPS, etc. Virtual Private Networks (VPNs) are created by establishing a virtual connection using dedicated connections is to provide a secured communication channel over public NWs. To secure VPNs, different issues must be addressed, e.g. the authentication process, client configuration, etc. A security administrator always checks the system and security log files looking for something abnormal. Audit tools are used by administrators to detect a wide range of rogue events, such as: unauthorized registry changes, protocol attacks, Denial of service (DoS) attacks, etc. Also, for website security different methods are handled, e.g. prevention of SQL injection, using complex passwords, management of cookies, and others.

The AC models are developed to match the security needs in all aspects of IT. In the presence of new technologies, such as: Cloud Computing and IoT, it is worth presenting some AC mechanisms in such fields.

3.4 Access Control in Cloud Computing

Cloud Computing (CC) is an emerging technology. Due to the huge amount of data which are generated from different end users' applications and information systems, CC is considered as an efficient solution for easier and faster storage retrieval of data. In CC, users can access computer services via the internet and this makes their data vulnerable to attacks. For this reason, different AC methods for CC are presented in literature. This section presents some of AC in CC methods.

Onankunju, in [16], introduces a method for providing secure AC in CC after presenting the possible CC attacks, e.g. DoS and authentication attacks. Hence, a hierarchical structure using a clock is presented to upload, download, and delete files to and from the cloud. The root of the hierarchical structure is the trusted authority which authorizes the top-level domain authorities. Also, domain authorities authorize cloud users. The system is composed of 4 parts: cloud owner, untrusted cloud, clock and cloud users. The user, with a key, encrypts his data before uploading it to the untrusted cloud. For the user, to access his data, he should send a request to the cloud owner which in turn sends him back a key. The key remains available for a certain period, then it becomes invalid after the clock stops counting. The user should access his data within the time limit. Another method is proposed in [17]. The method avoids using static passwords, it uses a one-time password and one day password. Whereas, the first password expires in two minutes, and the second one after twenty-four hours. The user receives passwords with encryption via e-mail for each login session.

3.5 Access Control in IoT

In [18], IoT is defined as "a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols". Which means a huge number of heterogeneous objects, with different technologies and platforms, are communicating together via the internet. Hence, all devices connected to the internet are vulnerable to attacks, and this is the case for IoT devices. In this context, various authentication and AC methods in IoT are also presented in literature, to integrate security issues with this technology.

Liu et al. in [19] propose a model to find a secure communication between things by a certain procedure. The main idea of this procedure is based on verifying identities between two IoT devices. The method is based on implementing authentication protocol in the presentation layer, where identification key establishment occurs. They adopt the concept of authorization in RBAC model, and for secure key establishment they implement Elliptic Curve Cryptosystem (ECC). Moreover, authors in [20] propose a "smart contract-based framework to implement distributed and trustworthy access control". Their aim is "to apply the smart contract-enabled blockchain technology to achieve distributed and trustworthy AC for the IoT". This framework contains multiple Access Control Contracts (ACCs), one Judge Contract (JC), and one Register Contract (RC). ACCs are implemented between subjects and objects for AC. JC is used for judging the unpleasant behavior of the subject during AC. RC is used to manage the ACCs and JC. To demonstrate the framework feasibility, case studies are addressed.

Different other AC proposals are presented in this field, which reflects the evolutionary stage of CC, IoT, and security concerns due to the presence of attacks.

4 Access Control Metamodels

The need to use AC methods in different system levels and technologies, imposes the necessity of finding AC models with combined features from two or more models. Due to the continuous increase and upgrade of information technology features, the presence of security threats also increases. The technological environment which is open to all types of users is a crucial concern, because it is also open to various types of attacks. This makes security enforcement, through AC models, an urgent need. So, many AC models are presented in literature with combined features from two or more AC models based on research motivations and needs. For example, we can find many models with combined features from both RBAC and ABAC. Some hybrid RBAC and ABAC models and others are presented for this purpose.

In [11] and due to RBAC's difficulty to set up an initial role structure in rapidly changing environments, and because RBAC does not support dynamic attributes, the idea of adding attributes to RBAC is addressed. The aim is to find a model that supports dynamic attributes specially in organizations. These features are presented to handle relationship between roles and attributes to provide better AC features in dynamic environments. Also, authors in [21] target the idea of enhancing features from both RBAC and ABAC, because both have complimentary features to each

other. Hence, Attribute Enhanced RBAC model (AERBAC) is presented. The model "retains the flexibility offered by ABAC, yet it maintains RBAC's advantages of easier administration, policy analysis and review of user permissions [21]".

However, AC models must consider the continuous developments and changes. The new technologies (CC, IoT...), the variety of platforms and applications, users' types, etc. comprise a complex fact in controlling secure and private accesses to the needed resources in different areas. All this makes AC models and even combining some features of them are insufficient to handle the needed target. This fact forces the need to find models with higher level of abstraction, which is called AC metamodels. The aim of AC metamodels is to serve as a unifying framework for specifying and enforcing any AC policy. For this purpose, different research works are present and still conducting for finding an AC metamodels. The following sections present some existing AC metamodels.

4.1 Metamodel Definition

Before exploring the existing AC metamodels, it is worth mentioning some essential definitions and ideas about metamodeling. Metamodel in [22] is defined as a textual, graphical/visual, or formal representation of concepts and how they are linked together. In other words, it is a structure of a collection of concepts in a certain domain. These concepts might be terms, rules, guidelines, etc. for an institution or organization. In [23], metamodeling is defined as modeling of a model, where they should describe the permitted structure to which models must adhere. Furthermore, models and metamodels need adaptable supporting tools due to changing requirements and policies. As mentioned earlier, metamodels can be illustrated using textual, graphical or formal representations. Though, different metamodeling tools and languages are presented in [22, 23] such as: Unified Modeling Language (UML), Meta-Object Facility (MOF), Eclipse Modeling Framework (EMF), MetaEdit, Conceptbase, and other tools. Fig. 2 shows the metamodel abstraction levels. M_3 is an instance of a model, it defines a specific information domain. M_2 is an instance of metamodel, it defines a language to describe an information domain. M_1 is an instance of a Meta-metamodel, it defines the language for specifying a model. M_0 is the infrastructure for a metamodeling architecture, it defines the language for specifying metamodels [22].

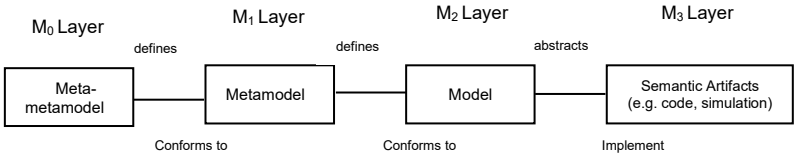

Fig. 2. The Four-layer metamodelling Architecture [23].

4.2 State of Art

Korman et al. in [24] propose a unified metamodel designed using Enterprise Architecture (EA) modeling language, the ArchiMate, and explain its use on many

scenarios and two business cases. Their aim is to combine different AC models in a single EA model, and to propose an extension to an established EA modeling language. Their model targets the challenge of flexibly modeling policies of authorization according to the most well-known AC models: DAC, MAC (BPL, BIBA, and Chinese Wall), RBAC, and ABAC, in terms of EA. The authors summarize some of the existing AC models, then define the vocabulary of these models, such as: subjects, objects, sessions, etc. to use them later in their presented metamodel. Then, they represent the conceptual models for the most basic common terms of AC, which are: subject, object, and access mode for the same purpose of later use. Also, the configurations of DAC, BLP, BIBA, Chinese Wall (CW), $RBAC_{0,1,2,3}$, and ABAC are represented as metamodels, e.g. in Fig. 3 they represent the metamodel for ABAC. Finally, these predefined steps are used as an introduction before presenting their unified metamodel in Fig. 4. The model mostly builds on the conceptual model of ABAC for its ability to emulate most functions of the other AC models.

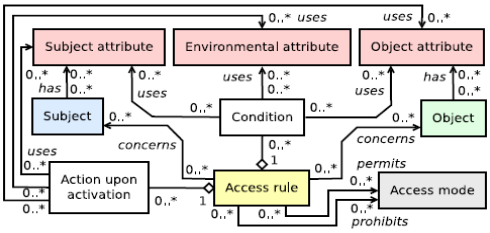

Fig. 3. Metamodel for expressing configurations of ABAC [24]

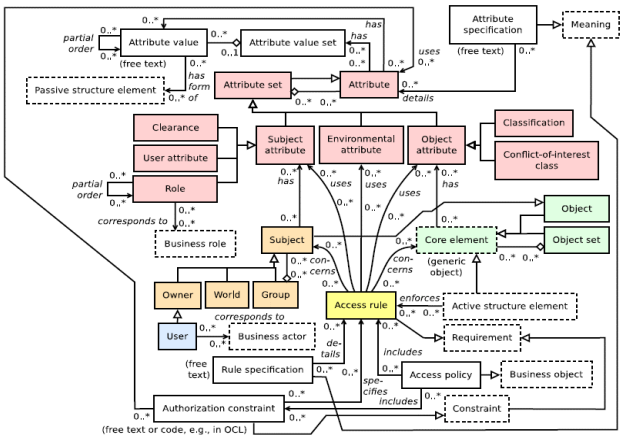

Fig. 4. Unified metamodel for modeling authorization [24]

Authors in [25] propose an AC metamodel to concurrently handle multiple AC models. Their metamodel consider four models, which are: CW, BLP, BIBA, and RBAC models. They start from the general concepts of metamodels; the object, subject, access mode, role, and the instances of associations between metamodeling elements (subject and role). They present the decision concept which relies behind

applying logical rules that are expressed in terms of AC metamodel elements. As proposed, each AC metamodel has a special element called Decision Handler. Fig. 5 illustrates the initial concept of their metamodel. Then, kernel AC elements are presented to later illustrate the associations between metamodel elements. These elements are: Object, Objects Group, Subject, Subjects Group, Access Mode, Additional Attribute, Query, Environmental Attribute, and Decision. Objects Group and Subject Group respectively represent sets of objects and subjects sharing some properties. The AccessMode represents, for example, some actions like: read, write, execute, etc. Query is the access request on an object by a subject. Environmental Attribute is used to hold information related to access request events such as time, place, temperature, etc. Additional Attribute represents a construct associated to a metamodel element to support the specification of some property of that element. Decision (e.g. permit, deny, indeterminate, NotApplicable…) is where the response is issued by the AC request. The metamodel is implemented with the four AC models (CW, BLP, BIBA, and RBAC) as shown in Fig. 5. ACmetaModelElement is a generalization of any element of the AC metamodels other than DecisionHandler. The proposed AC metamodels relies on a subset of UML but without determining how an instance of the metamodel returns an AC decision. To do so, First Order Logic (FOL) mapping is used for relating entities to their types, specifying relationships between entities, and for expressing a decision logic based on relations. To specify the integration of several AC metamodels of a hybrid AC policy, they presented an example of Integration Metamodel (IM) based on Ascending Decisions Tree (ADT). Hybrid AC policies mean applying multiple AC specifications and policies. Fig. 6 shows the ADT metamodel example which concludes with only one AC decision as output, in response to a set of multiple AC decisions as input. ADT nodes are, DecisionHandler instances and nodes applying Combining Algorithms (ComAls). DecisionHandler instance issues its AC decision based on its metamodel decision logic (explained in [25]). ComAlNode issues its decision by applying its ComAl on the decisions of its direct children nodes. It has a unique root which returns the decision of the whole tree with the hybrid AC policy. Finally, the proposed IM metamodel is illustrated in Fig. 7, which encompasses the whole above concepts.

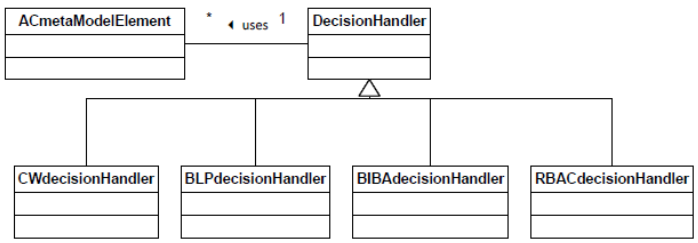

Fig. 5. DecisionHandler specializations in access control metamodels [25]

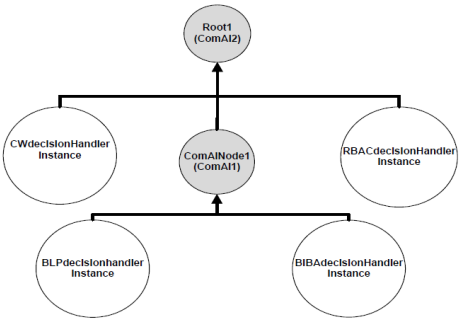

Fig. 6. An ADT integrating multiple AC metamodels instances [25]

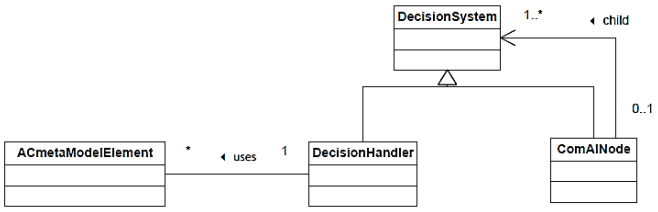

Fig. 7. The main view of IM metamodel [25]

Besides, designing AC metamodels for distributed environments (consisting of several sites) are also taken into account. This is due to the importance of finding dynamic or collaborative policies, to consider system changes or collaborate with other policies of several sites. This concept is covered in [26], where authors present the importance of finding a formal specification language to define AC models and policies in such environments. For this purpose, the concept of rewriting techniques (for security policies and protocols) is suggested to provide semantics for distributed AC mechanisms. Also, they mention some several available rewrite-based programming languages for fast prototyping, such as, Muade. Their proposed rewriting techniques are defined as an instance of a metamodel based on Distributed Event Based AC model (DEBAC). However, the authors first explain the advantages of rewriting systems, then explore the existing AC models to introduce the concept of a unifying metamodel for AC. Also, they describe the main features of the AC metamodel and define the extension of the metamodel for distributed environments. Second, they propose a formal specification of the distributed metamodel in a rewrite-based language, constructed on core concepts of AC models, and focus on the modular properties of the system. In this context, the federation notion is considered, like database systems where several systems are integrated by a federated system and each system preserves its autonomy. Third, general policy combining operators are well-defined to define combinations of policies. Algebraic terms are used in all the steps to define and rewrite policies, properties, operational semantics of the distributed metamodel, and integrating combination operators in the distributed metamodel. In [26], detailed explanations about the used expressions are also provided. Similarly, a metamodel extension mechanism is proposed as a solution in

the context of MoNoGe French collaborative project [27]. This project is based on a textual Domain Specific Language (DSL). The aim is to face up current limitations and the lack of standard solutions in the existing project. In addition to building a generic lightweight metamodel extension approach for the industrial environment where rapid and efficient adaptations of the used modeling tools are needed. Authors begin by defining the concept of modeling in real industrial projects, which deal with different models and metamodels, in addition to supporting tools. Hence, they present the main industrial use case of MoNoGe, which comes from DCNS (a world-leading company in naval defense and energy that especially develops Combat Management Systems, CMS, for ships). DCNS use two separate modeling tools: DoDAF (U.S. Department of Defense Architecture Framework) standard, and Modelio supporting software design and development. Then, a metamodel extension operators and a DSL are introduced to easily use them. Also, two different implementations of their proposed extension mechanism, based on Eclipse/EMF and the Modelio modeling environment, are also described. Fig. 8 shows a sample of the grammar of their proposed metamodel extension textual DSL.

```
Model:'define' extensionName=ID 'extending' metamodel+=Metamodel ':'
    prefix+=Prefix ("," metamodel+=Metamodel ':'  prefix+=Prefix)*
    '{' extensions += Extension* '}';
Extension: Create | Refine | Generalize | ModifyClass | FilterClass;
Metamodel:  name=ID;
Prefix: name=ID;
Create: 'add class' class=ID;
Refine: 'add class' classNew=ID 'specializing' prefix=[Prefix] '.'
    classOriginal=ID;
Generalize: 'add class' classNew=ID 'supertyping' prefix+=[Prefix]
    '.' class+=ID ("," prefix+=[Prefix] '.' class+=ID)*;
ModifyClass:
    'modify class' prefix=[Prefix] '.' class=ID '{'
        modifyOperators += ModifyOperator*
    '}';
ModifyOperator: AddProperty | ModifyProperty | FilterProperty |
    AddConstraint | FilterConstraint;
AddProperty:    'add property' property=ID 'type' type=ID;
ModifyProperty: 'modify property' property=ID value+=ValueAssignment
    ("," value+=ValueAssignment)*;
ValueAssignment:    attribute=ID '=' value=EString;
FilterProperty: 'filter property' property=ID;
FilterClass:    'filter class' prefix=[Prefix] '.' class=ID;
AddConstraint:  'add constraint' constraint=ID value=EString;
FilterConstraint:   'filter constraint' constraint=EString;
```

Fig. 8. The Grammar metamodel extension textual DSL [27]

Furthermore, AC metamodels also consider the Web Content Management Systems (WCMSs). WCMSs are frameworks that are widely used for web applications development for enterprises, e.g. Drupal, Wordpress, and Joomla. Users with little technical knowledge can fully develop technical systems due to their integrated environment. This environment provides design definition, layout, content management and organization of the application. In this context, Martínez et al. in [28] highlight the importance of security requirements, since WCMSs may contain sensitive information. Authors propose a metamodel to the representation of WCMS AC policies, to ease the analysis and manipulation of security requirements by abstracting them from vendor-specific details. They focus on the idea of facilitating WCMS configuration, to minimize the possibility of vulnerabilities because users often lack depth technical and security knowledge. Besides, authors enumerate some law-level security aspects, for example, management of cookies, and prevention of

SQL injection vulnerabilities. Although AC techniques are integrated in most WCMS systems, some limitations still exist in such systems, e.g. knowing the level of protection of the implemented access policy in a WCMSs is complex and error-prone task. For this purpose, authors propose to raise the level of abstraction of the AC implementation, to be represented according to a vendor-independent metamodel. Fig. 9 represents the proposed WCMS metamodel sample, which is inspired by RBAC.

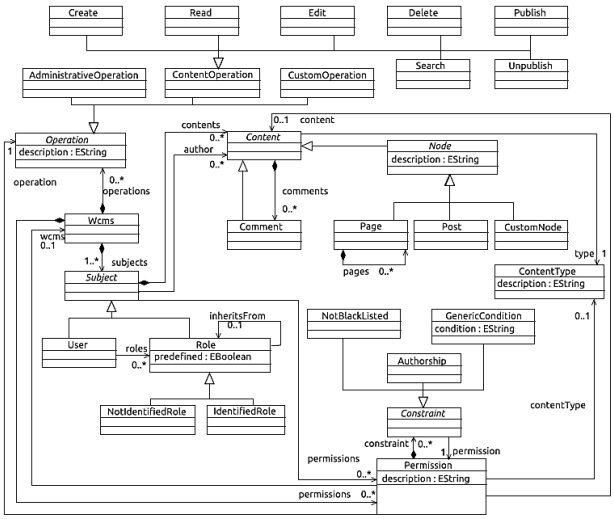

Fig. 9. WCMS metamodel sample [28]

The four metamodel basic elements functionalities explained in [28] are: content, actions, permissions and subjects. The idea of their process is to automatically extract the AC information in the domain of WCMSs. Moreover, they mention that even their metamodel could be manually filled by investigating AC information using WCMS administration tools, it should also be filled by an automatic reverse engineering approach. Thus, they present an automatic process for Durpal in Fig. 10, where Durpal contents with AC information are stored in the backend database. SQL queries over the database are injected to obtain a model that conform their proposed WCMS metamodel. This process allows the possibility of defining extra AC rules or modifying them programmatically. The abstract representation of the WCMS AC model is developed using Model Driven Engineering (MDE) where the relation between metamodel elements can be easily realized. Concerning WCMS migration, they present using their metamodel as a pivot representation. This illustration is to represent the AC information of the old WCMS in a way corresponding to their metamodel, to facilitate its analysis. Another web service metamodel is proposed in [29] to handle the verification of authorization in Web Service Oriented Architecture (WSOA). The aim of this metamodel is to improve the existing AC models for better features to match the requirements of WSOA. The proposed metamodel is depicted in Fig. 11. The metamodel is an enhancement for hierarchical RBAC and ABAC. Conceptual UML modeling is used to present the metamodel and define the sets and

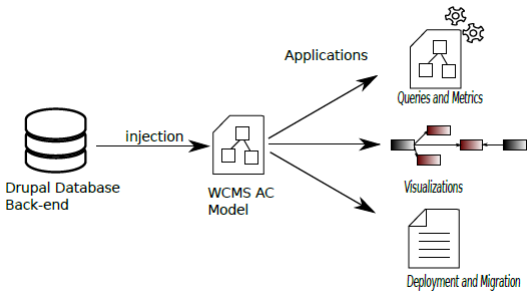

Fig. 10. Drupal access control extraction [28]

relations. In this metamodel the commonly used type of operations, e.g. read, write, which are carried between permission and object elements are removed. Instead, the focus on placing the input parameters of web service operation. As shown in Fig. 11, there are two relations from permission to an object: 1) indirect relation via policy, 2) direct relation to the input parameter, where "a parameter does not need to know if its value is evaluated for access control [29]". Web services Composition "consists of multiple invocations of other Web Service Operations in a specific order [29]". This element plays an important role to stop execution in case of missing authorization at an early stage. The proposed metamodel is mapped to an authorization verification service, which is part of Identity Management (IdM) architecture. Then it is linked with the core concern of WSOA, and the feasibility of this approach is illustrated in a case study. Also, Web Services Description Language (WSDL) is used for service interface definition. Likewise, addressing network security is also a critical concern. Martínez et al. in [30] propose a model driven approach to extract network AC policies enforced by firewalls within a network system. Their concept tackles the problem of filtering the traffic of a network with the presence of a number of filtering rules. based on their analysis, "the network topology, that may include several firewalls, may impose the necessity of splitting the enforcement of the global security policy among several elements. [30]". In this context, their aim also is "to raise the

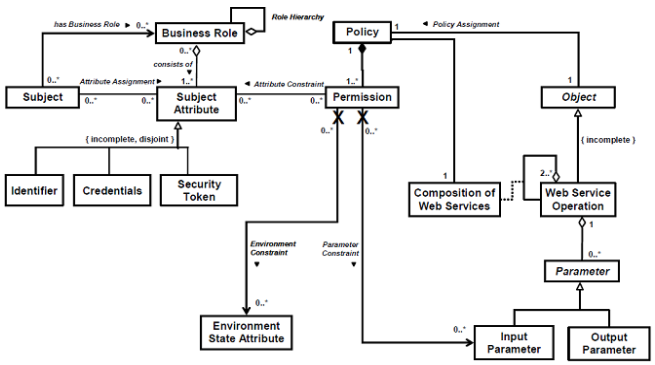

Fig 11. Metamodel for AC in WSOA [29]

level of abstraction of the information contained in the firewall configurations files so that the AC policy they implement is easier to understand, analyze, and manipulate [30]". However, Eclipse tool (Xtext) is used to extract AC information out of the net-filter iptables language. The features of RBAC and OrBAC AC models are implemented in this approach. Fig. 12 shows the network connection metamodel. The metamodel consists of two entities, host and connection. The former represents a network host, e.g. IP address. The latter represents connections between hosts, where the port and the protocol are specified to establish connections and specify if the connection is allowed or denied.

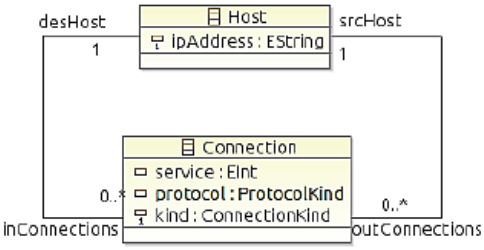

Fig 12. Network connection metamodel [30]

5 Potential Research Issues

Developing AC metamodel, that covers the features of all other AC models, is a challenging issue especially if the aim is to find a metamodel that encompasses all existing AC models. The dynamic requirement for enforcing security issues and the rapid propagation of technology makes it an urgent need.

The presented metamodels come with some advantages, and many case studies are addressed to handle each metamodel design. But we notice that each metamodel is itself a case and does not encompass a general base concept. Table I summarizes the presented AC metamodels, which depicts different metamodels in IT systems. Also, it indicates that metamodeling is a recent research field, especially in the last few years. As we notice, all the presented metamodels are designed for dedicated case scenarios or projects based on some features of AC models. Furthermore, in addition to the achieved progressions, there are still issues to be addressed. In fact, despite the advantages of the presented metamodel in [24], authors present some of its limitations. For example, it misses the concept of logging, in addition to the difficulty for a potential implementation of automated analytical capabilities of the unified metamodel. So is the case for the other metamodels, they are not generic enough to include all AC models features. As we can see some combined features from some models to cure some existing deficiencies in some projects or enhancing some service features. In addition to the concept of applying the same complex process in assigning relationships between model elements in some metamodels.

Table 1. Summary of Presented Access Control Metamodels.

Metamodel Features					
Ref.	Publication	Designed for	Type	Used Models	Modeling tools
[24]	2016	Enterprise architecture	Unified	DAC, MAC RBAC, ABAC.	ArchiMate
[25]	2015	Enterprise	Hybrid	CW, BLP, BIBA, RBAC	UML, FOL
[26]	2014	Distributed Environment	Metamodel Extension	DEBAC	Rewrite semantics
[27]	2015	Industrial project	Metamodel Extension	MoNoGe project	DSL, EMF, Modelio
[28]	2013	WCMSs	Metamodel Extraction	RBAC	MDE
[29]	2007	Web Service	Metamodel Integration	$RBAC_1$, ABAC	UML
[30]	2012	Network firewalls	Metamodel Extraction	RBAC, OrBAC	Xtext

In this paper, we try to spot on the idea of metamodels, the existing metamodels in literature, and to look into future with some raised questions concerning this matter. Subsequently, in addition to the existing concerns and metamodels, many questions are raised, such as: is it possible to find a more general concept of metamodels? Is it possible to implement easier and general unified structures of metamodels, and visualize their elements more readily? Are the existing metamodels handle the feature of flexibility for any new extensions or transformations? Or, are the current metamodeling frameworks flexible and dynamic enough for any changes? If so, what are the possible ways, steps or strategies to merge metamodel elements? Although there are many metamodels are built, based on many AC models, do they overcome the existing limitations of these AC models? Last but not least, is the existing metamodeling tools and languages enough to answer all the above questions or some of them?

Additionally, as presented in this survey we can see that metamodels are implemented for different scenarios: AC models, WCMSs, and distributed environments. Thus, is there any opportunity to find a metamodel design or plan that encompasses the different scenarios? Or is it more efficient to find a unified metamodel for each scenario? As a result, currently we may not have answers to the above inquires, but at least we know that metamodels introduce a new era of enforcing policies and controlling access in IT world.

Another interesting feature, that is missing in current AC metamodels, is the ease of the migration from an AC model to another. In fact, having a metamodel, should make it possible to translate an existing AC policy between the different AC models covered by the metamodel.

6 Conclusion

We covered in this survey existing AC metamodels, and the AC models they generally cover (DAC, MAC, RBAC…). We also presented a brief explanation about how these AC models are used in different fields e.g. Databases, operating systems, IoT, etc. Moreover, Metamodels are proposed in literature to concurrently handle multiple AC models, also in distributed environments and web systems. The main goal is to develop a metamodel that is general enough to instantiate all existing AC models and that may also help organizations to easily migrate from an AC model to another.

Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number 06351].

References

1. Matt, B., *Introduction to computer security*. 2006: Pearson Education India.
2. De Capitani di Vimercati, S., S. Paraboschi, and P. Samarati, *Access control: principles and solutions.* Software: Practice and Experience, 2003. **33**(5): p. 397-421.
3. Hu, V.C., D.R. Kuhn, and D.F. Ferraiolo, *Attribute-Based Access Control*. 2018, Norwood: Artech Hous
4. Kayem, A.V., S.G. Akl, and P. Martin, A presentation of access control methods, in Adaptive Cryptographic Access Control. 2010, Springer. p. 11-40.
5. Ennahbaoui, M. and S. ELHAJJI. Study of access control models. in Proceedings of the World Congress on Engineering. 2013
6. Ausanka-Crues, R., Methods for access control: advances and limitations. Harvey Mudd College, 2001. 301: p. 20.
7. Sandhu, R., D. Ferraiolo, and R. Kuhn. The NIST model for role-based access control: towards a unified standard. in ACM workshop on Role-based access control. 2000
8. Crampton, J. On permissions, inheritance and role hierarchies. in Proceedings of the 10th ACM conference on Computer and communications security. 2003. ACM
9. Belokosztolszki, A., Role-based access control policy administration. 2004, University of Cambridge, Computer Laboratory .
10. Zhang, C.N. and C. Yang, Designing a complete model of role-based access control system for distributed networks. J. Inf. Sci. Eng., 2002. 18(6): p. 871-889.
11. Kuhn, D.R., E.J. Coyne, and T.R. Weil, Adding attributes to role-based access control. Computer, 2010. 43(6): p. 79-81.
12. OrBAC: Organization Based Access Control. 2010; Available from: http://orbac.org/?page_id=21 .
13. Anderson, R., Security engineering. 2008: John Wiley & Sons
14. Rhodes-Ousley, M., Information security: the complete reference. 2013: McGraw Hill Education
15. Rajpoot, Q.M., C.D. Jensen, and R. Krishnan. Attributes enhanced role-based access control model. in International Conference on Trust and Privacy in Digital Business. 2015. Springer.

16. Onankunju, B.K., *Access control in cloud computing.* International Journal of Scientific and Research Publications, 2013. **3**(9): p. 1.
17. Hussain, S., *Access Control in Cloud Computing Environment.* International Journal of Advanced Networking and Applications, 2014. **5**(4): p. 2011.
18. Atzori, L., A. Iera, and G. Morabito, *The internet of things: A survey.* Computer networks, 2010. **54**(15): p. 2787-2805.
19. Liu, J., Y. Xiao, and C.P. Chen. Authentication and access control in the internet of things. in Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on. 2012. IEEE.
20. Zhang, Y., Kasahara, S., Shen, Y., Jiang, X. and Wan, J., Smart Contract-Based Access Control for the Internet of Things. arXiv preprint arXiv:1802.04410, 2018.
21. Rajpoot, Q.M., C.D. Jensen, and R. Krishnan. Integrating attributes into role-based access control. in IFIP Annual Conference on Data and Applications Security and Privacy. 2015. Springer.
22. Assar, S., Meta-modeling: concepts, tools and applications, in IEEE 9th International Conference on Research Challenges in Information Science, IEEE RCIS 2015. 2015: Athens, Greece; Available from: https://www.computer.org/cms/ComputingNow/education/said-assar-metamodeling-tutorial.pdf
23. Sprinkle, J., Rumpe, B., Vangheluwe, H. and Karsai, G., 3 Metamodelling, in Model-Based Engineering of Embedded Real-Time Systems. 2010, Springer. p. 57-76.
24. Korman, M., R. Lagerström, and M. Ekstedt, Modeling enterprise authorization: a unified metamodel and initial validation. Complex Systems Informatics and Modeling Quarterly, 2016(7): p. 1-24.
25. Abd-Ali, J., K. El Guemhioui, and L. Logrippo, *A Metamodel for Hybrid Access Control Policies.* JSW, 2015. **10**(7): p. 784-797.
26. Bertolissi, C. and M. Fernández, A metamodel of access control for distributed environments: Applications and properties. Information and Computation, 2014. **238**: p. 187-207.
27. Bruneliere, H., Garcia, J., Desfray, P., Khelladi, D.E., Hebig, R., Bendraou, R. and Cabot, J. On lightweight metamodel extension to support modeling tools agility. in European Conference on Modelling Foundations and Applications. 2015. Springer.
28. Martínez, S., Garcia-Alfaro, J., Cuppens, F., Cuppens-Boulahia, N. and Cabot, J. Towards an access-control metamodel for web content management systems. in International Conference on Web Engineering. 2013. Springer.
29. Emig, C., Brandt, F., Abeck, S., Biermann, J. and Klarl, H., An access control metamodel for web service-oriented architecture. 2007.
30. Martínez, S., Cabot, J., Garcia-Alfaro, J., Cuppens, F., & Cuppens-Boulahia, N. A model-driven approach for the extraction of network access-control policies. in Proceedings of the Workshop on Model-Driven Security. 2012. ACM