Modeling Authorization in Enterprise-wide Contexts

Matus Korman*, Robert Lagerström, Mathias Ekstedt

KTH Royal Institute of Technology
100 44 Stockholm, Sweden
{matusk,robertl,mathiase}@ics.kth.se

Summary. Authorization and its enforcement, access control, has stood at the beginning of the art and science of information security, and remains being a crucial pillar of secure operation of IT. Dozens of different models of access control have been proposed. Although enterprise architecture as a discipline strives to support the management of IT, support for modeling authorization in enterprises is lacking, both in terms of supporting the variety of individual models nowadays used, and in terms of providing a unified metamodel capable of flexibly expressing configurations of all or most of the models. This study summarizes a number of existing models of access control, proposes an unified metamodel mapped to ArchiMate, and illustrates its use on a selection of simple cases.

1 Introduction and related work

Authorization and its enforcement (access control) has been a crucially important pillar of enterprise IT security, both on technical levels (in computer systems, databases, networks etc.) and organizational levels (access policy and its human enforcement). Yet, major enterprise architecture (EA) modeling languages such as ArchiMate [1] do not currently support modeling access control, nor provide extensions, which would enable practitioners to do so in an elegant, defined and generic manner.

A plethora of different access control models have been proposed (a subset is listed in table 1). Several of them have become widely adopted in a variety of IT systems. For example, discretionary access control (DAC) implemented using access control lists (ACLs) and role based access control (RBAC) resound most, their origin dating back to 70's and 80's, respectively. While these and a few more models have been employed extensively, there are some fresh candidates on the verge of larger-scale adoption, such as the attribute based access control (ABAC), not to mention their more recent and sophisticated risk-adaptive variants.

Access control models are typically modeled formally (e.g., [3, 7, 8, 11, 13]), and a subset of them even freely conceptually (e.g., [10, 11, 13, 17]). However, the analysis of access control in enterprise IT landscapes calls for a middle way

* This study has been financed by SweGrids, the *Swedish Centre for Smart Grids and Energy Storage* (www.swegrids.se).

S. España J. Ralyté, P. Soffer, J. Zdravkovic and Ó. Pastor (Eds.):
PoEM 2015 Short and Doctoral Consortium Papers, pp. 81-90, 2015.

Table 1. Summary of access control models, policies and mechanisms studied

Name	Character	Property	Domains	Policy	References
Access matrix	Mechanism	*any*	*any*	DAC*	[2]
Access control list (ACL)	Mechanism	*any*	*any*	DAC*	[3] (p. 35)
Protection bits	Mechanism	*any*	OS	DAC*	[4] (p. 14)
Capability ticket	Mechanism	*any*	*any*	DAC*	[5] (p. 134)
Protection ring/domain	Mechanism	*any*	*any*	MAC*	[3]
Lattice model	Model	*any*	*any*	MAC	[3, 6]
Bell LaPadula (BLP)	Model, policy	Conf:ty	*any*	MAC	[7]
Biba	Model, policy	Integrity	*any*	MAC	[3]
Brewer-Nash (Chinese wall)	Model, policy	Integrity	*any*	MAC	[8, 9]
Role-based access control (RBAC)	Model	*any*	*any*	*any*	[10, 11]
Attribute-based access control (ABAC)	Model	*any*	*any*	*any*	[12–15]
Usage control model (UCON)	Model	*any*	*any*	*any*	[16]
Risk-adaptive access control (RAdAC)	Model	*any*	*any*	*any*	[17, 18]
Token-based access control (TBAC)	Model	*any*	*any*	*any*	[19]

* *denotes typically, however not exclusively*; OS *denotes operating systems*

between these approaches – conceptual modeling according to a defined, unified language.

This study addresses the challenge of flexibly modeling scenarios of authorization according to the most well-known access control models, in terms of EA. The purpose is to enable EA practitioners easily capturing authorization relations in enterprise architectures. The study presents a number of existing access control models in terms of conceptual modeling, and proposes a unified metamodel (seen as an ontology or modeling grammar) for describing their configurations. The proposed unified metamodel is formed as a prospective extension of the popular EA modeling language ArchiMate. Subsequently, four illustrative examples are presented, to exemplify several different ways of modeling authorization, and to demonstrate the applicability of the metamodel. Similar approach has also been adopted by Basin et al. [20], proposing an approach titled "model driven security", building on an extended metamodel of RBAC [10] called SecureUML [21], and providing a semantically well-founded modeling language and code generation process. Somewhat similarly, the work of Gaaloul & Proper [22] and Gaaloul et al. [23] propose an access control model for use in EA modeling. However, the approaches are exclusively based on RBAC, which makes them inapplicable for modeling a number of other commonly used models. Slimani et al. [24] and Muñante et al. [25] propose approaches for modeling access control in a more generic manner, however, both fall short of being able to express an arbitrary ABAC configuration, not to mention the more recent models. This study treats the most well-known and widely adopted models of access control, as well as

Table 2. Common vocabulary of access control

Term	Description
Subject/ requestor	An entity capable of performing actions in a system under consideration (SUC). For example, a program running on an operating system.
Object/ resource	An entity within a SUC, which is in need of protection from unauthorized access. For example, an object can be a document or a system operation.
Mode of access	The way, in which a subject can access an object within a SUC. Examples are read, write, execute, delete, create, search, or list contents.
Access rule/ permission/ prohibition/ access right	A rule specifying a specific mode of access for a subject to an object – either permitting it (more common), or by prohibiting it. In a yet more generic sense, a single access rule may also specify multiple modes of access for multiple subjects to multiple objects.
User	A user can be a subject, often having the privilege to further create subjects in a SUC (e.g., run programs). A non-subject user might only be allowed to manipulate subjects, however, not itself access objects directly.
Session	A temporally constrained window of usage, typically authenticated (e.g., by a log-in procedure), in which a user can act within a SUC via subjects.
Classification	A security designation of an object (e.g., a document), which indicates e.g. the highest secrecy of information contained therein, according to a predefined scheme (e.g., a mathematical lattice defining a partially ordered set of security labels, or simpler, a full order such as in figure 3b.
Clearance	A security designation of the eligibility of a subject to access object having a certain level of classification, in a certain access mode. Specifics depend on the model of access control under consideration.
Security label	A mark associated with an object or a subject, which carries a specific security meaning. A security label typically denotes a specific classification (of an object) or clearance (of an object).
Attribute	A characteristic of an entity such as a subject (e.g., organizational affiliation or business role), an object (e.g., minimum amount of credits needed for access or classification), or the environment (e.g., time of day, threat level or other environmental condition). It can be seen as a function that takes as input an entity (e.g., a subject, object or the environment), and returns a specific value based on the properties/state of the entity.
Token	An attribute extended through its possible dependence on volatile, dynamic properties or items such as cryptographic tokens (e.g., a Kerberos token), devices (e.g., a smart card), biometric tokens, or risk tokens, which change based on subject behavior and/or other conditions.

some prospectively powerful newcomers, and proposes a unifying metamodel mapped to ArchiMate.

2 Access control: concepts and models

This section introduces the terms specific to access control used throughout the paper (table 2), and briefly describes the models of access control treated.

A distinction between an access control model, policy, and mechanism should be made. While the first describes an access control system, the second describes

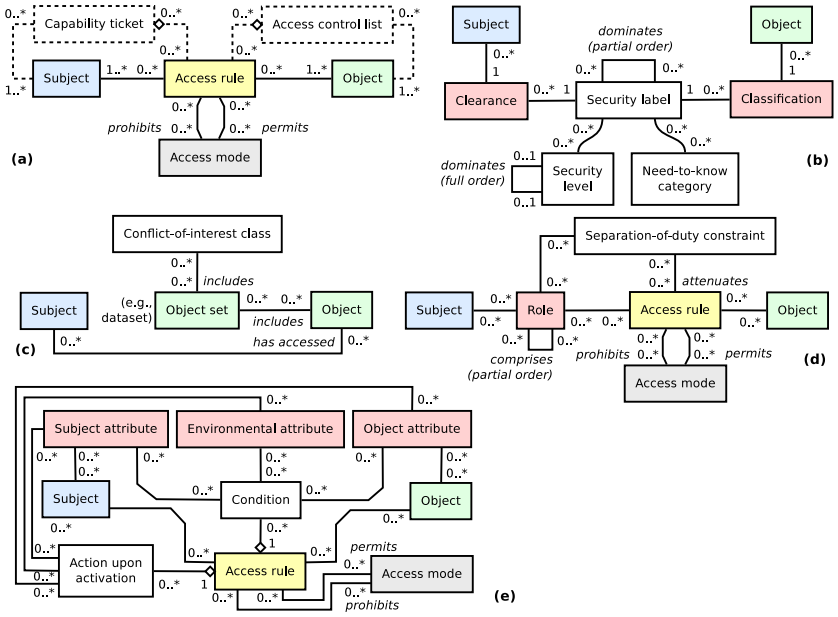

Fig. 1. Generic metamodels for expressing configurations of (a) DAC; (b) BLP and Biba; (c) Brewer-Nash (Chinese wall); (d) $\text{RBAC}_{0,1,2,3}$; and (e) ABAC. In (a), the dashed items describe implementation aspects/alternatives.

a set of requirements for the system, and the third describes a part of an implementation of the system. Table 1 summarizes the access control models, policies and mechanisms treated within this study.

Discretionary access control (DAC, figure 1a) models are based on the identity of subjects and access rules stating what the subject can and/or shall not do. Subjects can decide over other subjects' permissions (access rules) [5]. DAC is likely the most prevalent access control model today, thanks to its simplicity and extensive legacy. An example of DAC can be found in a typical Windows or UNIX filesystem. The most common representation of a DAC configuration is an access control matrix [2], which is in practice typically represented by multiple access control lists (ACL) (see [3], p. 35) or capability tickets [5].

Mandatory access control (MAC, figure 1b) models have largely become synonymous with the term *lattice-based access control* [6], the security levels of which are structured as a lattice. The Bell-LaPadula (BLP) model [7] and Biba model [3] use need-to-know categories (e.g, project numbers) for regulating access to objects in a DAC-like fashion, and security labels denoting security levels for classification of objects and clearance of subjects. Both models consider two modes of access – reading and modification. Biba additionally considers invocation (i.e., calling upon another subject) which can consequentially be viewed as modification under the invoked subject's clearance. However, while BLP addresses confidentiality, Biba addresses integrity. In BLP, reading an object is

allowed to a subject if the subject's clearance is equal or higher than the object's classification, and writing it allowed if it is equal or lower. In Biba, reading an object is allowed if the subject's clearance is equal or lower than the object's classification, and writing (and invocation) is allowed if it is equal or higher. Although the difference between BLP and Biba makes the two policies conflicting, they can be combined given separate labels for confidentiality and integrity [6]. The Brewer-Nash model [8] (Chinese wall, figure 1c) differs in that its configuration changes dynamically according to the history of each subject's access. The model defines a term *conflict-of-interest class*, which groups datasets, or rather *object sets* (e.g., data of different banks), and regulates access as follows. A subject can read an object only if the object is in the same object set as an object already accessed by the subject, or if the object belongs to an entirely different conflict-of-interest class. A subject can write [to] an object only if it also can read the object, and if no such object can be read that is in a different object set from the one for which write access is requested and which at the same time contains unsanitized (i.e., not anonymized) information.

Role-based access control (RBAC, figure 1d) [10,11] is technically a non-discretionary model, in which subjects are granted access based on the roles they take on themselves for a specific session (e.g., Jane can take on herself the role of a system administrator, a financial analyst, or a teller). Several types of RBAC have been identified [11] according to their features. RBAC_0 denotes a minimal version, in which a subject can only take on itself a single role for a session, and there are no constraints for separation of duty. RBAC_1 augments RBAC_0 with hierarchies of role inclusion, in form of a partially ordered set. RBAC_2 augments RBAC_0 with constraints (e.g., expressing that a subject must not be assigned two specific roles at the same time). RBAC_3 combines RBAC_1 and RBAC_2, which also enables constraining for dynamic separation of duty (e.g., a subject must not take on itself two specific roles within a single session).

Attribute-based access control (ABAC, figure 1e) [10, 13] is one of the more recent models, which, although being the fastest growing one [14] and seemingly on the verge of a large-scale adoption [15, 26], is not yet as widely known as RBAC. Its major advantages over DAC, MAC and RBAC, are far greater expressiveness, richness, greater precision and flexibility. In fact, ABAC no longer requires specifying individual relationships between subjects and objects [14]. On top of ABAC, UCON [16] proposes *mutable attributes* (changeable as a consequence of access in addition to administrative actions), predicates that have to be evaluated prior to a usage decision (*authorizations*), and predicates that verify mandatory requirements for access (*obligations*). The invention of ABAC has been preceded by numerous extensions to RBAC (e.g., by spatial, temporal, task-, organization- and decision-related aspects), however, this study does not treat them in favor of the more generic ABAC.

Risk-adaptive and token-based access control (RAdAC [17,18], TBAC [19]) have been proposed in the recent years. On top of ABAC, RAdAC considers measures of risk related to access decisions, which can be arbitrary (e.g., based on subjects' behavior and trust; ways, probabilities and consequences of misusing

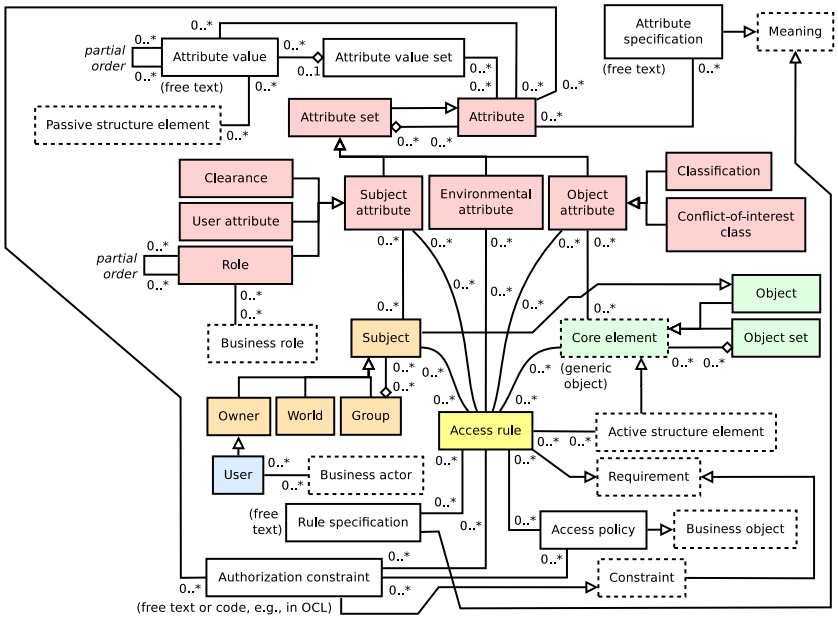

Fig. 2. Proposed unified metamodel for modeling authorization. The dashed entities are defined by ArchiMate [1].

objects; environmental conditions). TBAC, although not having received academic attention, broadens the perspectives of application and implementation of ABAC and RAdAC in a way highly relevant for EA practice and modeling. Example tokens are listed in [19].

3 Metamodel for modeling authorization

The unified metamodel for modeling authorization is depicted in figure 2. Below, this section first describes the metamodel, motivating certain features of its design, and later provides a set of illustrative examples, each showing the use of the metamodel through a corresponding model of a concrete configuration of a certain access control model.

Structurally and syntactically, the unified metamodel mostly resembles that of ABAC (cf. figure 1e), thank to ABAC's ability to encompass or emulate most of the other access control models' function. For structural simplicity however, the entity `Attribute` semantically comprises both *attribute* from ABAC and *token* as used in TBAC. Also, items such as *role* or *clearance* can be modeled simply as an *attribute*. For more clarity however, the unified metamodel retains a number of such entities, namely `Role`, `User attribute`, `Clearance`, `Classification` and `Conflict-of-interest class`, since those are expected to occur commonly. Less generally common such entities (e.g., *predetermined*

explicit authorization or *location*) can be instantiated from the closest fitting child class of `Attribute` rather than having a separate class. `Role`, unlike other children of `Attribute`, allows the modeler to define arbitrary partial orders, to capture configurations of $\text{RBAC}_{1,3}$ [11]. Since the name of a modeled attribute might not suffice to capture its full nature and its range of values, the modeler can further specify attributes textually (e.g., by free text or references), using `Attribute specification`. At the same time, the modeler can specify partial orders (e.g., lattices) of attribute values using `Attribute value` and group them into sets (e.g., for security levels and need-to-know categories), using `Attribute value set`. Moreover, attribute values can be linked to instances of ArchiMate's `Passive structure element`, to denote values that might already be modeled using ArchiMate. An `Object` can group arbitrary sets of ArchiMate's `Core element`s. `Subject` figures as a child of `Object`, since a subject itself can be an object. `Subject`, much like `Attribute`, is also further categorized into the commonly occurring `Owner`, `Group`, `World` (i.e., anyone), and even `User` denoting a an intelligent actor (e.g., human), for the case its distinction from `Subject` is desirable to model. `Access rule` can connect to a `Subject` and an `Object`, although it is not necessary, e.g., in case of ABAC. `Access rule` can relate to `Attribute`s of any kind, also multiple ones. It can also relate to ArchiMate's `Active structure element`, e.g., to denote dependency on a system that realizes its enforcement etc. As with `Attribute specification`, `Rule specification` can help further specify an `Access rule`. Finally, an `Access rule` might be a part of a specific `Access policy`. Various authorization constraints (e.g., cf. RBAC_2 [11]) might need to be modeled, using `Authorization constraint`. Similarly to `Access rule`, the modeler can also relate an `Authorization constraint` to an `Access policy`. Finally, three patterns occur repeatedly in the design of the proposed unified metamodel. First, a specific form of grouping is used at `Attribute`, `Subject` and `Object` represented by ArchiMate's `Core element`: The grouping entity (titled a *-set* or *-group*), inherits from its immediate base entity, and aggregates a set of its instances. This allows arbitrary tree-like grouping under the name of the base entity (e.g., `Subject`). Second, relations of partial order allow the modeler to create arbitrary lattice-like hierarchies. Third, multiplicities of relations are highly permissive, and in most cases allow *0..** rather than the more constraining *0..1* or *1..**, to provide higher flexibility. The proposed metamodel includes bindings to ArchiMate entities, in figure 2 distinguished from others using dashed lines. Following, four illustrative examples of the metamodel's usage are presented.

Example of DAC: File system (figure 3a). Let us have a school computer file system, one teacher and two students. The students, belonging to a group called "Students", are allowed to read contents of the course study directory, and execute a program for exam submission. The teacher, belonging to a group called "Teachers", is allowed to read and write grade records, and read the contents of an exam directory, which stores exams submitted by students.

Example of MAC: BLP multilevel security (figure 3b). Let us have an environment with multilevel security policy according to the Bell-LaPadula

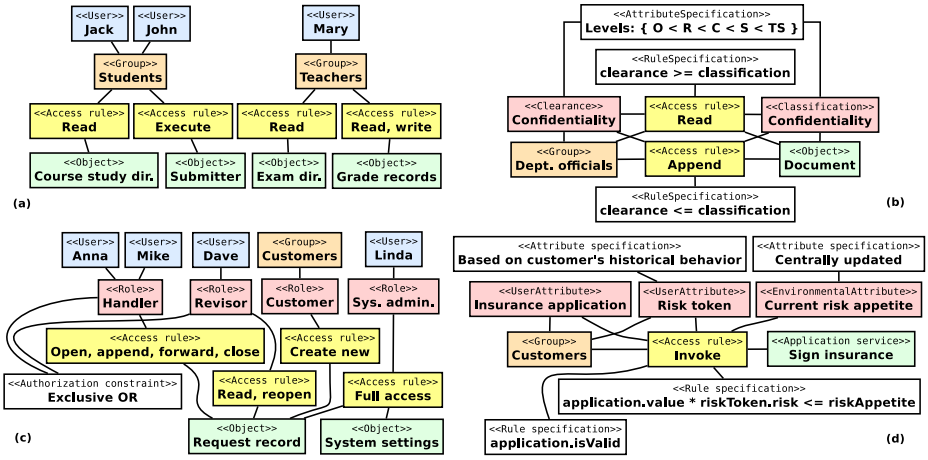

Fig. 3. Illustrative example of a (a) DAC; (b) Bell-LaPadula; (c) RBAC; and (d) ABAC/RAdAC/TBAC model configuration.

model [7]. Let us only consider a single group of users called "Department officials" and their authorizations to read and append to protected documents.

Example of RBAC: Request tracking system (figure 3c). Let us have a request tracking system with two types of objects – system settings and request records, a few users, a group representing customers, and four roles, each having different access: system administrator, customer, request handler and revisor. Further, let it be forbidden to combine the roles of handler and revisor.

Example of ABAC/RAdAC/TBAC: Insurance application system (figure 3d). Let us have an automated processing of insurance applications in an insurance company. Let the company use a risk token, which calculates risk value for each customer based on the customer's history; and let there be a risk appetite setpoint providing a threshold for how risky a deal the company can sign at a given moment. Let the system register the customer's insurance request as an user attribute automatically upon the customer applying through a web-based form. Finally, let us only allow the system to invoke an insurance signage service if the insurance application is valid, and if the risk that the signed deal would pose to the company does not exceed the risk appetite.

4 Discussion and conclusions

The proposed metamodel offers a high degree of modeling flexibility, which emerges mainly from the presence of four features: (1) broad possibility to group items or present them as groups/sets (e.g., attributes, subjects, objects) with the possibility of introducing further detail; (2) the possibility to arbitrarily textually specify attributes, attribute values, access rules, policies and constraints; (3) the conceptual redundancy provided (e.g., the modeler can model a DAC or a

RBAC model both as an ABAC model, or entirely avoiding the use of attributes in the former case while making use of `Role` in the latter case; (4) the possibility to exploit the permissive cardinality constraints to make abstractions similar to grouping, and so to reduce the number of modeled instances and connections.

Although the generalizability of the metamodel to all existing models of access control is difficult to evaluate, the consideration of well-known and highly generic models of access control such as ABAC provides outlook for a high degree of generalizability of the proposal. Similar concern relates to how applicable will the metamodel remain over time, which depends on the amount of innovation taking place within the domain of access control.

Although the proposed metamodel of this study shares many conceptual likenesses with the results of Gaaloul & Proper [22], Gaaloul et al. [23], Basin et al. [20], Slimani et al. [24] and Muñante et al. [25], it surpasses these works in terms of the breadth of coverage of the different existing models of access control. Additionally, this study shares much likeness in terms of its ArchiMate mapping compared to that proposed in Gaaloul & Proper [22]. However, the latter is more direct and constraining (e.g., the entity `User` inherits from ArchiMate's `Business actor` and `Role` inherits from ArchiMate's `Business role`, rather than associating with them), which leads to lesser modeling flexibility in comparison to the mapping proposed in this study.

In terms of conceptual modeling, this study has summarized a number of relevant models of access control including a few recent ones, presented an ArchiMate-mapped unified metamodel capable of expressing configurations of all the individual models of access control treated, and finally provided four illustrative examples of using the metamodel in distinct scenarios.

In the future, enriching the unified metamodel with automated analysis is intended, enabling the metamodel to warn about risky patterns of configuration, or deviations from best practice. Additionally, the metamodel could analyze attributes related to a given access control implementation and configuration, enterprise needs and maintenance processes (e.g., cost, amount of maintenance, modifiability or security through the likelihood of being in a state of misconfiguration), and so help enterprises optimize their architecture.

References

1. The Open Group: ArchiMate 2.1 Specification, Technical Standard. Van Haren Publishing, Zaltbommel, `http://www.opengroup.org/archimate/` (2013)
2. Lampson, B.W.: Protection. ACM SIGOPS Operating Systems Review **8**(1) (1974) 18–24
3. Biba, K.J.: Integrity considerations for secure computer systems. Technical report, DTIC Document (1977)
4. Hu, V.C., Ferraiolo, D., Kuhn, D.R.: Assessment of access control systems. US Department of Commerce, National Institute of Standards and Technology (2006)
5. Stallings, W., Brown, L.: Computer Security: Principles and Practice. 2 edn. Pearson Education (2012)
6. Sandhu, R.S.: Lattice-based access control models. Computer **26**(11) (1993) 9–19

7. Bell, D.E., LaPadula, L.J.: Secure computer systems: Mathematical foundations. Technical report, DTIC Document (1973)
8. Brewer, D.F., Nash, M.J.: The chinese wall security policy. In: Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on, IEEE (1989) 206–214
9. Sandhu, R.S.: Lattice-based enforcement of chinese walls. Computers & Security **11**(8) (1992) 753–763
10. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed nist standard for role-based access control. ACM Transactions on Information and System Security (TISSEC) **4**(3) (2001) 224–274
11. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **29**(2) (1996) 38–47
12. Hu, V.C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to attribute based access control (abac) definition and considerations. NIST Special Publication **800** (2014) 162
13. Jin, X., Krishnan, R., Sandhu, R.: A unified attribute-based access control model covering dac, mac and rbac. In: Data and applications security and privacy XXVI. Springer (2012) 41–55
14. Hu, V.C., Kuhn, D.R., Ferraiolo, D.F.: Attribute-based access control. Computer **48**(2) (February 2015) 85–88
15. Wagner, R.: Identity and access management 2020. `http://www.issa.org/resource/resmgr/JournalPDFs/feature0614.pdf` (2014)
16. Park, J., Sandhu, R.: The ucon abc usage control model. ACM Transactions on Information and System Security (TISSEC) **7**(1) (2004) 128–174
17. McGraw, R.W.: Risk-adaptable access control (radac). In: Privilege (Access) Management Workshop. NIST–National Institute of Standards and Technology–Information Technology Laboratory. (2009)
18. Shaikh, R.A., Adi, K., Logrippo, L.: Dynamic risk-based decision methods for access control systems. Computers & Security **31**(4) (2012) 447–464
19. Radhakrishnan, R.: The fifth and final frontier of access control model. `http://www.isaca-washdc.org/presentations/2012/201211-session3_article.pdf` (2012)
20. Basin, D., Doser, J., Lodderstedt, T.: Model driven security: From uml models to access control infrastructures. ACM Transactions on Software Engineering and Methodology (TOSEM) **15**(1) (2006) 39–91
21. Lodderstedt, T., Basin, D., Doser, J.: Secureuml: A uml-based modeling language for model-driven security. In: UML 2002 - The Unified Modeling Language. Springer (2002) 426–441
22. Gaaloul, K., Proper, H.: An access control model for organisational management in enterprise architecture. In: Semantics, Knowledge and Grids (SKG), 2013 Ninth International Conference on, IEEE (2013) 37–43
23. Gaaloul, K., Guerreiro, S., Proper, H.A.: Modeling access control transactions in enterprise architecture. In: Business Informatics (CBI), 2014 IEEE 16th Conference on. Volume 1., IEEE (2014) 127–134
24. Slimani, N., Khambhammettu, H., Adi, K., Logrippo, L.: Uacml: Unified access control modeling language. In: New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on, IEEE (2011) 1–8
25. Munante, D., Gallon, L., Aniorté, P.: An approach based on model-driven engineering to define security policies using orbac. In: Availability, Reliability and Security (ARES), 2013 Eighth International Conference on, IEEE (2013) 324–332
26. Sandhu, R.: The future of access control: Attributes, automation and adaptation. `http://profsandhu.com/miscppt/coimbatore_131219.pdf` (2013)