Role-Based Access Control in Retrospect

Virginia N. L. Franqueira, VF InfoSec Consulting
Roel Wieringa, University of Twente

***Abstract*—Role-Based Access Control (RBAC) has been a success in terms of the amount of research that went into it, its uptake in international standards, and its adoption by major software vendors. Yet, RBAC remains complex to implement in user organizations. In this paper we review the state of the art of RBAC in terms of RBAC features, assumptions, strengths and possible weaknesses, and review current developments to mitigate these weaknesses. This review helps practitioners to assess the applicability of RBAC to their organization and also indicates where more research is needed to improve RBAC.**

***Index Terms*—Access Control, Identity and Access Management (IAM), RBAC, role engineering, role management, security management.**

I. Introduction

Since first introduced in the 90's, the Role-Based Access Control (RBAC) model evolved into probably the most discussed and researched access control model in academia [1, Page 145]. A 2010 economic analysis of RBAC estimated that "[in 2010] just over 50% of users at organizations with more than 500 employees are expected to have at least some of their permissions managed via roles" [2, Page ES-5]). It has become the basis for hundreds of theoretical studies, research prototypes, and textbooks. Especially after the RBAC standard was officially approved by the American National Standards Institute (ANSI) in 2004, RBAC features also gained a lot of attention from high profile commercial vendors. For example, the Microsoft Authorization Manager module provides RBAC capabilities for Windows Server 2008 and 2003, SELinux allows a RBAC layer of abstraction between the user and the underlying type-enforcement model, and the NetWeaver Identity Management module, which can work integrated with the SAP Business Suite, provides RBAC features. Finally, the use of RBAC has also been recommended by security standards, such as the Health Insurance Portability and Accountability Act (HIPAA).

Despite this success, the RBAC model is also target of critique. Some security experts suggest that most user organizations don't implement RBAC [3]. Triggered by this and other anecdotal comments, we aim to review what is known about experience with using RBAC in practice. We structure our review by means of the following questions.

- Q1: What are the basic features of the RBAC model?
- Q2: Which assumptions are implicit in the RBAC model?
- Q3: What are the claimed strengths of the RBAC model?
- Q4: Which phenomena, and deriving problems, observed in practice limit the strengths of the RBAC model?
- Q5: How are the elements uncovered by Q1–Q4 related to each other?

First, we sketch a conceptual framework of the RBAC life cycle.

II. RBAC Life Cycle

Roles can be used to control access to information in at least four types of applications: (i) *support applications*, with coarse-grained Operating System-specific roles, (ii) *stand-alone business applications*, with application-specific roles, (iii) *enterprise-wide applications*, with roles shared among several applications, and (iv) *cross-enterprise applications*, with roles shared among several organizations. In any of these types of applications, the life cycle of a role-based application consists of the following three phases [4]:

Customization
This phase basically involves planning, software customization, and **role engineering**: (i) design of roles, and (ii) design of the role structure.
The design of roles can take a top-down approach from the business point-of-view, a bottom-up approach from the point-of-view of existing permissions, or a hybrid approach that combines both [5]. There are commercial products available for mining roles in a bottom-up approach such as the SAM Role Miner [5].

Implementation
Implementation consists of (i) set-up of users' need-to-know policies, and (ii) assignments of users to roles and of roles to permissions. It also involves activities related to launching a role-based application in production.

Operation
The day-to-day operation of an RBAC application, called **role management**, consists of keeping the role structure, and assignments user-role and permission-role up-to-date. According to Gallaher et al. [4], this is the phase where the strengths of RBAC translate to economic benefits.

We answer Q1–Q4 in Sections III–VI and only return to role engineering and role management when we review phenomena in Section VI.

III. Basic RBAC Features

We collected the following list of features of the RBAC model from the RBAC ANSI/INCITS 359:2004 standard.

Core RBAC

Feature F1: Permissions are assigned only to roles, never directly to users.
Feature F2: There is a many-to-many relationship between users and roles.

Feature F3: There is a many-to-many relationship between roles and permissions.

Feature F4: Users do not need to have all their roles always activated.

Feature F5: Users can have more than one role activated at the same time.

Review functions for Core RBAC

Feature F6: It is possible to have an overview of all users assigned to a specific role.

Feature F7: It is possible to have an overview of all roles assigned to a specific user.

Hierarchical RBAC

Feature F8: Roles can be organized in hierarchies, allowing inheritance of permissions.

For references see sidebars *Core RBAC* and *Advanced RBAC*.

IV. RBAC Assumptions

RBAC relies on the principle that users should not acquire permissions because of individual attributes. It assumes that users share profiles depending on, e.g., responsibilities, duties, job functions, qualifications, or authority in accordance with the organizational structure. These roles determine the set of permissions that users should have. Therefore, changing the permissions of a role impacts a (possibly large) set of users, and assigning a new user to roles automatically grants this user a complete set of needed permissions. User turnover and function changes become a matter of removing assignments or performing reassignments user-role. One implicit assumption of RBAC is that the role structure and role-permission assignments in an organization are mature, and therefore stable, and that the set of users and user-role assignments are very dynamic. Seta's marketing survey [6] confirms that the benefits of RBAC increase if there is a large number of users, a high turnover rate, little change of roles, and a stable organizational structure.

RBAC is flexible in terms of role semantics. This represents a strength if assumed that those semantics are well understood and agreed on among those involved with role engineering and management [7]. However, such assumption is particularly uncertain in enterprise-wide applications and in cross-enterprise applications (see Section II).

Furthermore, RBAC assumes there are no unknowns involved in granting access. However, this is not always true. For instance, in Web-based applications the identity of users is usually not known in advance. In this case, the Web server must first establish trust in the requesting user before finding the mapping from the user to her authorized roles [8]. The same may happen with permissions; they may be unknown until the actual need arises (see Section VI).

Summarizing, RBAC makes the following assumptions.

Assumption A1: Users should not acquire permissions because of individual attributes; they share profiles which determine their roles.

Assumption A2: The number of roles is much smaller than the number of users to be granted permissions.

Assumption A3: The role structure and the set of permissions assigned to each role are stable; what changes a lot is the set of users and their assignments to roles.

Assumption A4: There is agreement about the semantics of roles between those people involved with their role engineering and management.

Assumption A5: Users' identity and permissions are known in advance, before the RBAC system decides either to grant or deny access.

V. RBAC Strengths

In cases where the assumptions listed in the previous section are true, RBAC has several strengths.

1) **Efficient Access Management**
   If A1–A3 are true, administering permissions in terms of roles (included executing review features F6 and F7) is efficient in terms of time and effort. Role management basically involves frequent deletion of assignments or reassignments user-role, eventual updates to assignments permission-role and rare updates to the role structure. This efficiency directly translates to economic benefits [2], [4].
2) **Effective and Efficient Enforcement of Need-to-know**
   An important internal control principle used to assure a satisfactory level of security is the principle of least privilege, also known as *need-to-know*. It aims at assuring that users are neither under-entitled nor over-entitled. This means that users should not have less permissions than they need to perform their duties (otherwise this can affect productivity and encourage users to circumvent the problem, causing security risks [2]) but no more than that. RBAC is a effective and efficient way to enforce the need-to-know principle since the assignment of users to roles (the *need* part) and the assignment of roles to permissions (the *know* part) jointly establish such connection.
3) **Simplified Regulatory Compliance**
   RBAC facilitates auditing; it is a very convenient way to manage and document users' permissions since it provides visibility of all permissions that a user has by means of the roles assigned to that user. As a consequence, RBAC is effective to demonstrate compliance to security requirements established by regulations such as the HIPAA, the Gramm-Leach-Bliley Act, the Sarbanes-Oxley Act, and to industrial standards such as the Payment Card Industry Data Security Standard (PCI DSS). While HIPAA explicitly recommends the use of RBAC, others require organizations to show that adequate internal controls are enforced or that access control policies are in place to safeguard data [4]. Although it is always possible to comply with these requirements with, e.g., Access Control Lists and groups, this tends to be more time consuming and error prone compared to the RBAC model. This happens because in such cases permissions are assigned directly to users, and are not necessarily

constrained within groups.

4) **Scalable Inheritance of Permissions via Role Hierarchy**
In the absence of role hierarchy, there are two alternatives for the assignment of many-to-many permissions to users.
The first alternative is to duplicate permissions assignments among roles. For example, if managers should have the permissions of programmers, this is represented in a flat structure of roles by assigning to the role "manager" permissions specific to managers *plus* permissions specific to programmers. Apart from a substantial increase on the number of role-permission assignments, inconsistencies may arise.
The second alternative is to not duplicate permissions across roles, but let users accumulate permissions via assignment to several roles. In this strategy, managers should be assigned *both* to roles "manager" and "programmer" to perform their duties. This increases the number of user-role assignments and becomes specially problematic to avoid violations of need-to-know.
In an hierarchical RBAC model by contrast, the inheritance of permissions achieved via role hierarchy turns the assignment of permissions to users a scalable task, potentially representing a major strength of RBAC.

5) **Flexible Semantics of Roles and Permissions**
No restrictions are imposed by the RBAC model regarding the semantics of roles and permissions. Therefore, semantics must be defined in the process of role engineering, and differs by type of applications (Section II). For example, in case RBAC is applied to Operating Systems, roles tend to be coarse-grained and usually refer to classes of users with semantics agreed among network administrators. In an university context, roles at this level could be "academic staff", "administration staff" and "students". The flexibility of roles becomes increasingly apparent when RBAC is applied to standalone business applications, enterprise-wide applications and cross-enterprise applications. In such cases, roles can be used to assign a set of permissions shared, e.g., by all users of a specific location, department or organization. Roles can also be defined at a lower level of abstraction, if created at the level of tasks. Even more, roles can represent business or technical roles. This strength only materializes, however, if roles are organized in hierarchies, and if consensus is reached about the semantics of roles.
The lack of predefined semantics for permissions also provides flexibility. For instance, permissions related to roles regarding database are typically fine grained and involve operations like "insert", "delete" and "append" applied to objects like records and tables. While permissions related to roles regarding business applications may involve operations like "place" and "manage" applied to objects like purchase orders.

Summarizing, we found the following strengths claimed for RBAC.

Strength S1: Efficient management of large scale users' permissions, both in terms of time and effort.
Strength S2: Effective and efficient enforcement of the need-to-know access control principle, achievable by the assignment of users to roles and by the assignment of roles to permissions.
Strength S3: Simplified auditing of users' permissions for regulatory compliance.
Strength S4: Scalable assignment of permissions via inheritance of permissions in role hierarchies.
Strength S5: Flexible semantics of roles and permissions.

VI. Phenomena Limiting RBAC Strengths

This section reviews phenomena observed in practice, as reported in the literature, which may limit the strengths of RBAC.

1) **Role Explosion** (a role engineering problem)
There are two main causes of role explosion, described next.
Complexity of Users Attributes.:
Individuality.: Different users belonging to a same functional role may have different sets of permissions, i.e. may need to perform different types of operations on different or even the same objects, depending on specific circumstances. The only way to accommodate such individualities in RBAC is to create one role for each set of permissions. For example, role "teller-at-probation" has more restrictive permissions than role "teller-not-at-probation" [9], and role "part-time-health-care-assistant" has more restrictive permissions than role "full-time-health-care-assistant [10].

Locality.: Different users belonging to the same functional role may have different sets of permissions depending on their geographic location. For example, a role "teller" may have different permissions for tellers located on A and B [9]; potentially giving rise to roles "teller-A" and "teller-B". Using hierarchy does not solve the problem, since role "teller" would have permissions common to A and B, inherited by roles "teller-A" and "teller-B" which would need extra permissions to cope with specifics of locality.

Particularity.: Different users belonging to a same functional role may be permitted to perform the same set of operations but on different objects. This again means that several roles must be created to accommodate different sets of permissions. For instance, doctors should only have permissions to perform operations on their own patients' data. This means that doctors cannot be given permissions to the whole container of patients data, otherwise they become over-entitled, violating the need-to-know principle. Such situation in a pure implementation of RBAC may give rise to a number of roles of the type *doctor-X* with permission to perform

operations on objects of the type *patients-of-X*.

Dynamic context-dependent constraints.: Constrained RBAC (see sidebar *Advanced RBAC*) recognizes the need to impose constraints either on the assignment of users to roles (to deal with static separation of duties) or of sessions to roles (to deal with dynamic separation of duties). However, there are dynamic context-dependent constraints that are important when granting permissions to users, apart from roles, such as the need for temporal constraints [9]. In practice, we often see situations in which a user should have permissions only during a certain period of time. This means that the activation of roles during a session should comply with time-based restrictions.

Factors like individuality, location, particularities and dynamic context-dependent constraints can increase the number of roles to a point where there is an impractical number of roles and, in extreme cases, more roles than users. The fact that all assignments of users to permissions need to be granted via roles can be both a strength and a limitation because the number of roles easily explodes.

2) **Unexpected Side-effects of Role Hierarchy** (a role engineering & management problem)
Although according to the RBAC standard a role hierarchy could support arbitrary hierarchies, *any* hierarchy should comply with the bottom-up rule of inheritance (see sidebar *Advanced RBAC*). This means that structuring and maintaining hierarchies require a clear understanding of the consequences of inheritance, otherwise side-effects of over-entitlement or under-entitlement may happen [11]. Since real-world hierarchies are complex, their management requires expertise beyond the knowledge base of existing staff in many companies, compared with less sophisticated mechanisms such as Access Control Lists [4]. This has economic implications, since these more skilled staff are likely to be more expensive.
Furthermore, in practice, exceptions to the RBAC model bottom-up inheritance rule happen. For instance, at first glance it is quite straightforward that role "project manager" should be higher in the role hierarchy than role "programmer", therefore managers would inherit programmers' permissions. However, a project manager may not need to have update permission over the production directory, as a programmer does, because managers may not have the technical skills to deploy executable code live [9].

3) **Interoperability Issues** (a role engineering & management problem)
Ambiguous Semantics.: As already mentioned, the RBAC standard does not impose any specific semantics on roles. In order to fill this semantic gap, a consensus must be reached among parties involved in terms of: (i) terminology (e.g., what will be the meaning of role "manager"?), and (ii) permissions (e.g., what should be the set of permissions assigned to role "manager"?). However, reaching such agreement is not trivial in practice [7], and may lead to errors in granting access [11].

Multiple Interpretations of the RBAC Model.: The RBAC standard is itself complex and raises debates about its interpretation. It is under scrutiny by the research community [12] and may evolve over time. In contrast, the concept of role is intuitive and it becomes easy to adopt *the idea of RBAC in general*, without being adherent to the standard, i.e., without fulfilling the core features F1–F5 of RBAC. Therefore, in practice, we often find numerous interpretations of the RBAC model, resulting in no interoperability between them.

4) **Rigidity in the Face of Modern Business Dynamics** (a role management problem)
The dynamics of current enterprises may turn the task of role management, specially in the presence of role hierarchies, overwhelming. Thus, business changes, such as merges, splits, outsourcing and business partnerships, may all affect the role structure and the assignments user-role and role-permission.

Need-to-know Versus Need-to-share.: Enterprises are facing two conflicting requirements: the *need-to-know*, from the security and compliance perspectives, and the *need-to-share*, from the business perspective [3]. Therefore, on the one hand, there is a push to restrict access to information to the minimum necessary for the business to meet growing security constraints and, on the other hand, there is a push to increase access to information to the maximum possible for the business to meet growing economic constraints.
The need-to-share increases the spectrum and diversity of permissions to be granted. For example, it is not unusual that users from third-parties need permissions to perform operations on objects in a similar way as employees need. However, it is harder to fit them into roles since their responsibilities, job functions and qualifications, most of the times, are not visible to the organization granting these permissions. As a result, either roles have to be created to accommodate these users on an almost one-to-one basis or these users have to fit into an average set of roles that may grant them more permissions than needed to perform their duties.
The dynamics of current organizations make need-to-share hard to manage. As a consequence, it may not be known upfront which permissions users *should* have until the need actually arises [3]. When this need arises, the situation falls under the same circumstances as discussed above, i.e., either new roles are created or there is a risk of over-entitlement via assignment to existing roles.

Rigid model.: The RBAC model results in two possible access decisions: *allow access* or *deny access*. However,

this binary decision does not cope with unforeseen situations (such as emergencies) or temporary responsibilities (such as in downsizing periods [11]), and presupposes that permissions are known in advance. In practice, specially in some domains like health care, these assumptions do not always hold and a way to securely apply exceptions becomes important.

Summarizing, each item in this list presents a phenomenon and the problem that it causes.

Role Explosion

Phenomenon P1: Aspects such as users' individuality, locality and particularity give rise to roles with a few members; coping with this contributes to role explosion.

Phenomenon P2: There may be many dynamic context-specific attributes which affect users' permissions; coping with this contributes to role explosion.

Unexpected Side-effects of Role Hierarchy

Phenomenon P3: Structuring and managing role hierarchies require a clear understanding of the inheritance of permissions; lack of this understanding causes unexpected side-effects resulting in under-entitlement or over-entitlement of users.

Interoperability Issues

Phenomenon P4: The meaning of roles (in terms of terminology and permissions) across different departments, branches, or business partners has to be shared for RBAC to be effective; reaching agreements about the semantics of roles across departments and enterprises may not be trivial, giving rise to interoperability problems.

Phenomenon P5: The RBAC standard is complex and evolves. In contrast, the concept of role is intuitive and the idea of RBAC in general becomes easy to adopt without compliance to the standard (i.e., with at least the core features F1–F5). This results in numerous interpretations of the RBAC model in practice, causing interoperability problems.

Rigidity in the Face of Modern Business Dynamics

Phenomenon P6: Growing need-to-share information in B2B relationships creates a situation where changes affecting users' permissions happen frequently; access management based on roles may become either an overwhelming task or may lead to violations of need-to-know policies.

Phenomenon P7: It may not be known in advance which permissions users should have until the need actually arises, and there are emergency situations or temporary responsibilities which fall outside users' normal roles; RBAC cannot deal with such dynamics.

VII. Conclusion

Taking the perspective of the ANSI/INCITS 359:2004 RBAC standard, our analysis provides a thorough, yet concise, overview of the relationship between the strengths of RBAC which will only materialize if the assumptions about its context of use are satisfied. This is illustrated in Figure 1(a). Our analysis also connects phenomena that can happen in the RBAC context of use to problems that are expected to arise, as illustrated in Figure 1(b).

These two diagrams are useful for practitioners when implementing RBAC in organizations in two ways: (1) they can check if assumptions and phenomena of RBAC context of use apply to their organizations, therefore anticipating strengths and problems to expect, and (2) they can diagnose possible reasons for unsuccessful use of RBAC. We complement this analysis in sidebar *Addressing Phenomena P1–P7* by pointing out initiatives that aim to avoid or reduce phenomena P1–P7.

References

[1] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 1st ed. John Wiley & Sons, Inc., 2001.

[2] A. C. O'Connor and R. J. Loomis, "2010 Economic Analysis of Role-Based Access Control," NIST, Tech. Rep. RTI Project Number 0211876, December 2010.

[3] B. Schneier and M. Ranum, "Schneier-Ranum Face-Off: Is Perfect Access Control Possible?" 03 September 2009, Information Security Magazine, http://searchsecurity.techtarget.com/magazineFeature/0,296894,sid14_gci1365957_mem1,00.html, accessed Jan 2010.

[4] M. P. Gallaher, A. C. O'Connor, and B. Kropp, "The Economic Impact of Role-Based Access Control," NIST, Tech. Rep. RTI Project Number 07007.012, March 2002.

[5] M. Kuhlmann, D. Shohat, and G. Schimpf, "Role Mining - Revealing Business Roles for Security Administration using Data Mining Technology," in *SACMAT'03: Proc. of the 8th ACM Symposium on Access Control Models and Technologies*. ACM Press, 2003, pp. 179–186.

[6] C. L. Smith, E. J. Coyne, C. E. Youman, and S. Ganta, "A Marketing Survey of Civil Federal Government Organizations to Determine the Need for RBAC Security Product," Report prepared by Seta Corporation for NIST, http://csrc.nist.gov/groups/SNS/rbac/documents/cost_benefits/seta.ps, accessed Oct 2011, July 1996.

[7] A. H. Karp, H. Haury, and M. H. Davis, "From ABAC to ZBAC: The Evolution of Access Control Models," *Information Systems Security Association Journal*, vol. 8, no. 4, pp. 22–30, April 2010.

[8] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli, *Role-Based Access Control*. Norwood, MA, USA: Artech House, Inc., 2003.

[9] B. Hilchenbach, "Observations on the Real-world implementation of Role-based Access Control," in *Proc. of the 20th National Information Systems Security Conference*, October 1997, pp. 341–352.

[10] AHIMA/HIMSS HIE Privacy & Security Joint Work Group, "The Privacy and Security Gaps in Health Information Exchange," Healthcare Information and Management System Society (HIMSS), http://www.himss.org/content/files/201106_AHIMA_HIMSS.pdf. Accessed 1 Nov 2011., April 2011.

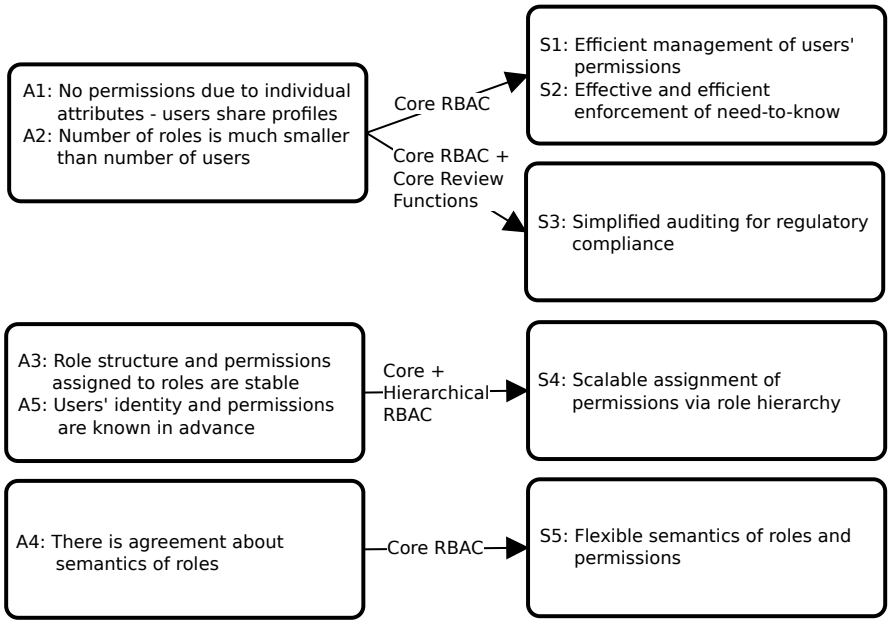
[11] K. D. Gordon, J. E. Michelman, B. Waldrup, and R. D. Slater, "Accounting Data Security at JEA," 2011, presented at the American Accounting Association Annual Meeting on 10 August in Denver, Colorado/U.S. Abstract can be found at http://aaahq.org/AM2011/abstract.cfm?submissionID=2382.

[12] N. Li, J.-W. Byun, and E. Bertino, "A Critique of the ANSI Standard on Role-Based Access Control," *IEEE Security and Privacy*, vol. 5, no. 6, pp. 41–49, 2007.

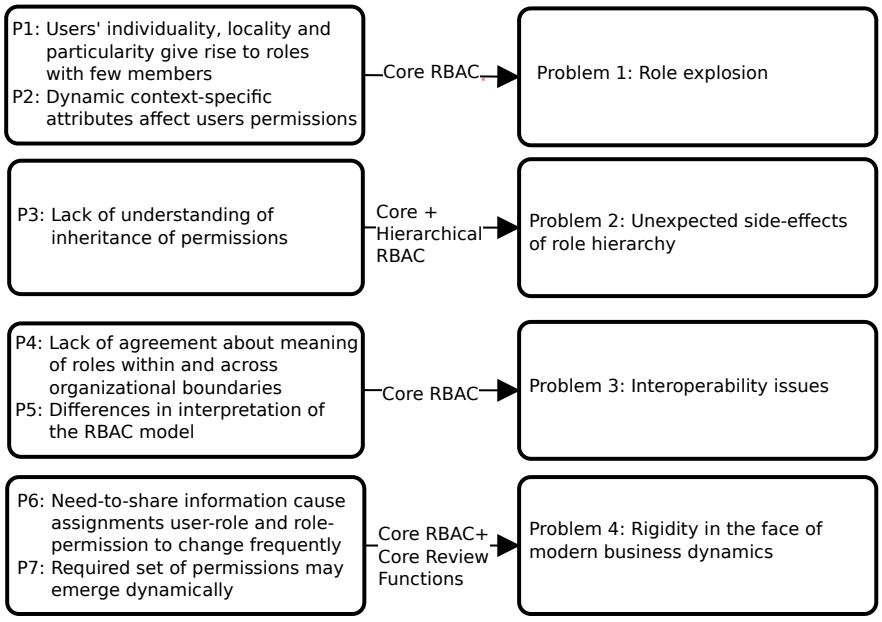
About the Authors

Virginia Nunes Leal Franqueira received her Ph.D. in Computer Science from the University of Twente (The Netherlands), where she also held a postdoc position. Currently, she is an independent researcher, based in the UK, providing consultancy related to Information Security.
Contact her at *virginia.franqueira@vf-infosec.com*.

Roel J. Wieringa is chair of the Information Systems group and Head of the Computer Science Department at the University of Twente, the Netherlands.
Contact him at *r.j.wieringa@ewi.utwente.nl*.



(a) Assumptions under which the claimed strengths of RBAC materialize.



(b) Phenomena of RBAC context of use that cause problems.

Fig. 1. Core RBAC consists of features F1–F5, core review functions consists of features F6–F7 and hierarchical RBAC consists of feature F8 (see Section III).

CORE RBAC

The Role-Based Access Control (RBAC) model, introduced by seminal work by Ferraiolo and Kuhn [1] and by Sandhu et al. [2], became an official access control model when the standard proposed by Sandhu, Ferraiolo and Kuhn [3] evolved into a proposal [4] that was subsequently modified to become the ANSI/INCITS 359:2004 RBAC standard [5].

Central to the RBAC model is the concept of **role**. "A role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role" [5]. The fundamental rationale of RBAC is the fact that a *role* is an intermediate element between *users* and *permissions*. Therefore, users are directly assigned to roles (many-to-many assignments), permissions are also directly assigned to roles (many-to-many assignments) and, this way, users get indirectly assigned to permissions.

According to the standard [5], "the permissions available to the user are the permissions assigned to the roles that are currently active across all the users sessions". A **user** is regarded as a human being although it could be a process, a machine, or a network. A **permission** is an approval to perform an **operation** on **objects**, where *operation* is an action, function or task that can be invoked by an user, and *object* can either refer to information containers (files, directories, database tables) or resources (printers, network drivers, computers).

A **session** is a mapping between a user and a set of roles assigned to this user during a session (one-to-many user-session and many-to-many session-role assignments). This means that RBAC requires the ability for a user to actually activate multiple roles simultaneously in a single session [3], although not all roles need to be activated [6]. Sessions represent a fundamental mechanism to implement the core RBAC model which support many-to-many user-permission assignments [6]. Also part of the core RBAC component are two functions to review: (i) the set of users assigned to a given role, and (ii) the set of roles assigned to a given user. Other review functions are "advanced", implying that they are not mandatory [5].

REFERENCES

[1] D. F. Ferraiolo and D. R. Kuhn, "Role-Based Access Controls," in *Proc. of the 15th NIST-NCSC National Computer Security Conference*, 1992, pp. 554–563.

[2] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[3] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST Model for Role-Based access control: towards a unified standard," in *RBAC'00: Proc. of the Fifth ACM Workshop on Role-based Access Control*. ACM Press, 2000, pp. 47–63.

[4] D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," *Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.

[5] ANSI/INCITS_359:2004, "Information Technology - Role Based Access Control," American National Standards Institute (ANSI), International Committee for Information Technology Standards (INCITS), February 2004.

[6] D. Ferraiolo, R. Kuhn, and R. Sandhu, "RBAC Standard Rationale: Comments on "A Critique of the ANSI Standard on Role-Based Access Control"," *IEEE Security and Privacy*, vol. 5, no. 6, pp. 51–53, 2007.

ADVANCED RBAC

Hierarchical RBAC adds the concept of hierarchy of roles to Core RBAC. "Hierarchies are a natural means of structuring roles to reflect an organization's lines of authority and responsibility" [1].

A hierarchy of roles supports inheritance of permissions, avoiding duplication of permissions that are common to two or more roles. Permissions are inherited bottom-up. This means that towards the top of the hierarchy we have more senior roles and towards the bottom we have more junior roles. Hierarchy in the RBAC model comes in two flavors: (i) limited role hierarchies, which allows single inheritance, and (ii) general role hierarchies, which allow multiple inheritance too. In general, tree-shape hierarchies allow aggregation of permissions, while inverted-tree-shape hierarchies allow sharing of permission [2].

Constrained RBAC is another addition to Core RBAC, compatible with Hierarchical RBAC, which addresses conflicts of interest among roles via static and dynamic separation of duty.

Static separation of duty (SSoD) constrains user-role assignments to ensure that a user do not acquire permissions to perform operations over protected objects which exceed her need-to-know, facilitating security breaches [2]. Determining SSoD conflicts require a pairwise analysis of roles from the role set [1]. If roles are hierarchically structured, possible inherited permissions should be taken into account to determine conflicting roles.

Dynamic separation of duty (DSoD) is much more flexible than SSoD. It applies to session-role assignments, restricting the activation of roles within or across a user's sessions [1]. Thus, instead of restricting the entire roles' space of permissions, it restricts the users' space of permissions. This way, a user can be assigned to conflicting roles but cannot activate them simultaneously. DSoD also applies when roles are hierarchically structured.

REFERENCES

[1] ANSI/INCITS_359:2004, "Information Technology - Role Based Access Control," American National Standards Institute (ANSI), International Committee for Information Technology Standards (INCITS), February 2004.

[2] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST Model for Role-Based access control: towards a unified standard," in *RBAC'00: Proc. of the Fifth ACM Workshop on Role-based Access Control*. ACM Press, 2000, pp. 47–63.

ADDRESSING PHENOMENA P1–P7

The Cyber Security committee (CS1.1) from the InterNational Committee for Information Technology Standards (INCITS) is a Task Group addressing public comments on the Draft INCITS 459 RBAC Implementation and Interoperability Standard (RIIS) [1], studying updates to the ANSI/INCITS 359:2004 RBAC standard, and launching a next generation RBAC standard (http://csrc.nist.gov/groups/SNS/rbac/, expected to be released in 2012).

RIIS defines RBAC *interaction functions* for syntactic and semantic exchange of RBAC information, allowing system-to-system offline interoperability [2]; this approach, to be incorporated to the new RBAC standard, addresses **phenomena P4 and P6**. Furthermore, RIIS specifies "how to design RBAC products to conform to INCITS 359" [2], aiming an increase in consistency among RBAC implementations, therefore, contributing to address **phenomenon P5**.

A change the CS1.1 committee is currently implementing to the RBAC standard is described in [3]. The new RBAC model will incorporate dynamic attributes (related to phenomenon P2) which will constrain the role structure, expressing a set of more static attributes (related to phenomenon P1). Access will be granted based on the intersection of P and R [3], where P is the set of permissions from roles active in a session (RBAC), and R is the set of permissions from attribute-based rules (ABAC [4]). This change to the RBAC standard aims to address **phenomena P1–P2** and, indirectly, **phenomenon P7**. Another approach to address these phenomena is the use of dynamic roles [5]. In this case, access is granted based on object and user profiles, including static roles and attributes, and based on environmental status, both used to determine the set of permissions of a user.

Moreover, Ferreira et al. [6] propose dealing with **phenomenon P7** by an RBAC model that allows users to have permissions granted on an exceptional basis, given that a justification is provided and logged.

Phenomenon P3 materializes in implementations of RBAC and, therefore, is dealt with (to some extent) in different ways. For instance, in the SAP NetWeaver Identity Management module, role hierarchy is established by means of single and derived roles, and authorization checks are available to adjust these derived roles. This phenomenon has also been dealt with in the context of implementing RBAC with the OASIS standard XACML language to promote portability of access policies and interoperability among different applications, therefore, addressing **phenomenon P4**. Stepien et al. [7] propose a rule inheritance mechanism in their non-technical XACML notation that simplifies assessing the effect of role inheritance.

Future experiences will learn to which extent all this work will effectively address phenomena P1–P7.

REFERENCES

[1] "Role-Based Access Control Implementation Standard," Int'l Committee for Information Technology Standards (INCITS), proposed standard, 2007, draft INCITS 459 RBAC, http://csrc.nist.gov/rbacdraft-rbac-implementation-std-v01.pdf, accessed Oct 2011.

[2] E. Coyne and T. Weil, "An RBAC Implementation and Interoperability Standard: The INCITS Cyber Security 1.1 Model," *IEEE Security and Privacy*, vol. 6, pp. 84–87, 2008.

[3] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding Attributes to Role-Based Access Control," *Computer*, vol. 43, pp. 79–81, 2010.

[4] A. H. Karp, H. Haury, and M. H. Davis, "From ABAC to ZBAC: The Evolution of Access Control Models," *Information Systems Security Association Journal*, vol. 8, no. 4, pp. 22–30, April 2010.

[5] R. Fernandez, "Enterprise Dynamic Access Control Version 2 Overview," Prepared for NIST, 2006, http://csrc.nist.gov/rbac/EDACv2overview.pdf, accessed Oct 2011.

[6] A. Ferreira, D. Chadwick, P. Farinha, R. Correia, G. Zao, R. Chilro, and L. Antunes, "How to Securely Break into RBAC: The BTG-RBAC Model," in *ACSAC'09: Proc. of the 2009 Annual Computer Security Applications Conference*. IEEE Press, 2009, pp. 23–31.

[7] B. Stepien, S. Matwin, and A. P. Felty, "Advantages of a Non-Technical XACML Notation in Role-Based Models," in *PST'11: 9th Annual Conf. on Privacy, Security and Trust*. IEEE Press, 2011, pp. 193–200.