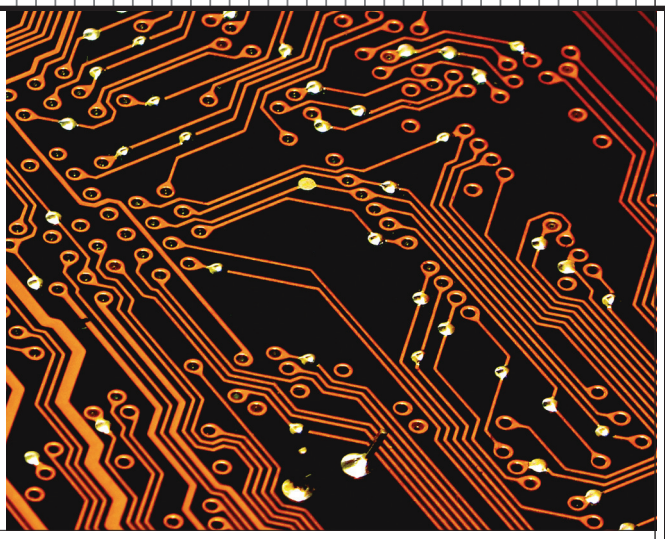
Role-Based Access Control in Retrospect

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



A review of the state of the art of role-based access control can help practitioners assess RBAC's applicability to their organization and indicates where more research is needed to improve the RBAC model.

Since its introduction in the 1990s, role-based access control (RBAC) has evolved into one of the most discussed and researched access-control models in academia.[1] A 2010 economic analysis estimated that "just over 50 percent of users at organizations with more than 500 employees are expected to have at least some of their permissions managed via roles."[2] RBAC has become the basis for hundreds of theoretical studies, research prototypes, and textbooks.

After the American National Standards Institute (ANSI) approved RBAC as a standard in 2004, the model's features were incorporated in many high-profile commercial offerings. For example, the Microsoft Authorization Manager module provides RBAC capabilities for Windows Server 2008 and 2003, SELinux allows an RBAC layer of abstraction between the user and the underlying type-enforcement model, and the SAP NetWeaver Identity Management module, which can integrate with the SAP Business Suite, offers RBAC features. Legislation such as the Health Insurance Portability and Accountability Act (HIPAA) also recommends using RBAC.

Nevertheless, some security experts have observed that most enterprises do not implement the RBAC model.[3] Such critiques and other negative anecdotal comments prompted a review of the state of the art of RBAC with the goals of helping practitioners assess its applicability to their organization and indicating where more research is needed to improve the model.

RBAC LIFE CYCLE

The RBAC model can be used to control access to information in at least four types of applications:

- support applications, with coarse-grained operating-system-specific roles;
- stand-alone business applications, with application-specific roles;
- enterprise-wide applications, with roles shared among several applications; and
- cross-enterprise applications, with roles shared among several organizations.

The life cycle of any role-based application consists of three phases: *customization*, *implementation*, and *operation*.[4]

Customization involves planning, software modification, and *role engineering*—the design of roles and 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] Commercial products are available for mining roles using a bottom-up approach such as SAM Role Miner.[5]

Implementation consists of setting up users' need-to-know policies, and assigning users to roles and roles to permissions. Implementation also involves activities related to launching a role-based application in production.

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

BASIC RBAC FEATURES

As defined by the ANSI/INCITS (InterNational Committee for Information Technology Standards) 359-2004 standard,

0018-9162/12/$31.00 © 2012 IEEE Published by the IEEE Computer Society

RBAC has eight basic features. These features are broken down into three categories.

Mandatory features for core RBAC

- F1: Permissions are assigned only to roles, never directly to users.
- F2: There is a many-to-many relationship between users and roles.
- F3: There is a many-to-many relationship between roles and permissions.
- F4: Users do not need to have all their roles always activated.
- F5: Users can have more than one role activated at the same time.

Mandatory review functions for core RBAC

- F6: It is possible to have an overview of all users assigned to a specific role.
- F7: It is possible to have an overview of all roles assigned to a specific user.

CORE AND ADVANCED RBAC

Role-based access control was originally developed by David Ferraiolo and Rick Kuhn,[1] and by Ravi Sandhu and colleagues.[2] After further modifications,[3,4] RBAC became an official standard as ANSI/INCITS 359-2004.[5]

CORE RBAC

The concept of *role* is central to RBAC. As defined by the standard, "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] RBAC's fundamental rationale is that a role is an intermediate element between users and permissions. An RBAC implementation directly assigns users to roles (many-to-many assignments) and permissions to roles (many-to-many assignments), and thus indirectly assigns users to permissions.

According to the standard, "the permissions available to the user are the permissions assigned to the roles that are currently active across all the user's sessions." A *user* is normally considered to be a human being, but it could also be a process, machine, or network. A *permission* is an approval to perform an *operation* on an object—that is, an action, function, or task that a user can invoke. The term *object* can refer either to information containers (such as files, directories, or database tables) or resources (such as printers, network drivers, or computers).

A *session* is a mapping between a user and a set of assigned roles, with one-to-many user-session assignments and many-to-many session-role assignments. RBAC allows a user to activate multiple roles simultaneously in a single session,[3] although it is not necessary to activate all roles.[6] Sessions implement the core RBAC model, which supports many-to-many user-permission assignments.[6] Also part of the core RBAC model are two functions to review the set of users assigned to a given role and the set of roles assigned to a given user. Other review functions in the standard are advanced, implying they are not mandatory.

ADVANCED RBAC

Hierarchical RBAC adds the concept of *role hierarchy* to core RBAC. As the standard indicates, "hierarchies are a natural means of structuring roles to reflect an organization's lines of authority and responsibility."[5]

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, which means that roles become more senior moving up the hierarchy. RBAC hierarchy comes in two flavors: *limited* role hierarchies, which allow single-role inheritance, and *general* role hierarchies, which allow multiple-role inheritance. Broadly, tree-shaped hierarchies allow aggregation of permissions, while inverted-tree-shaped hierarchies allow sharing of permissions.[3]

Another addition to core RBAC, *constrained RBAC* addresses conflicts of interest among roles via static and dynamic separation of duties. Constrained RBAC is compatible with hierarchical RBAC.

Static separation of duty (SSoD) constrains user-role assignments to ensure that users do not acquire permissions to perform operations on protected objects that exceed their need-to-know, which facilitates security breaches.[3] Identifying SSoD conflicts requires a pair-wise analysis of roles from the role set.[5] If roles are hierarchically structured, the RBAC implementation should take into account possible inherited permissions 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.[5] Thus, instead of restricting the entire roles' space of permissions, DSoD restricts the users' space of permissions. A user can be assigned to conflicting roles but cannot activate them simultaneously. DSoD also applies when roles are hierarchically structured.

References

1. D.F. Ferraiolo and D.R. Kuhn, "Role-Based Access Controls," *Proc. 15th Nat'l Computer Security Conf.*, Nat'l Inst. of Standards and Technology, 1992, pp. 554-563; http://csrc.nist.gov/rbac/ferraiolo-kuhn-92.pdf.
2. R.S. Sandhu et al., "Role-Based Access Control Models," *Computer*, Feb. 1996, pp. 38-47.
3. R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST Model for Role-Based Access Control: Towards a Unified Standard," *Proc. 5th ACM Workshop Role-Based Access Control* (RBAC 00), ACM, 2000, pp. 47-63.
4. D.F. Ferraiolo et al., "Proposed NIST Standard for Role-Based Access Control," *ACM Trans. Information and System Security*, Aug. 2001, pp. 224-274.
5. ANSI/INCITS 359-2004, *Information Technology—Role-Based Access Control*, American Nat'l Standards Inst./Int'l Committee for Information Technology Standards, 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*, Nov./Dec. 2007, pp. 51-53.

Nonmandatory feature for hierarchical RBAC

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

The "Core and Advanced RBAC" sidebar provides additional information about the standard and the model's main features.

RBAC ASSUMPTIONS

The RBAC model makes the following assumptions.

- A1: Users should not acquire permissions because of individual attributes; they share profiles that determine their roles.
- A2: The number of roles is much smaller than the number of users to be granted permissions.
- A3: The role structure and the set of permissions assigned to each role are stable; what changes frequently are the set of users and their assignments to roles.
- A4: There is agreement about the semantics of roles among those people involved in role engineering and management.
- A5: Users' identity and permissions are known in advance, before the RBAC system decides either to grant or deny access.

RBAC relies on the principle that users should not acquire permissions because of individual attributes. It assumes that users share profiles depending on, for example, 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 a role's permissions 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 user-role reassignments. One implicit RBAC assumption 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 dynamic. The benefits of RBAC increase if there are a large number of users, a high turnover rate, little change of roles, and a stable organizational structure.[6]

RBAC is flexible in terms of role semantics. This constitutes a strength if those involved in role engineering and management understand and agree on those semantics,[7] but such an assumption is particularly uncertain in enterprise-wide and cross-enterprise applications.

RBAC assumes there are no unknowns involved in granting access, but this is not always true. In Web-based applications, for example, the identity of users is usually not known in advance. In this case, the webserver must first establish trust in the requesting user before finding the mapping from the user to the authorized roles.[8] The same is true of permissions—they might be unknown until the actual need arises.

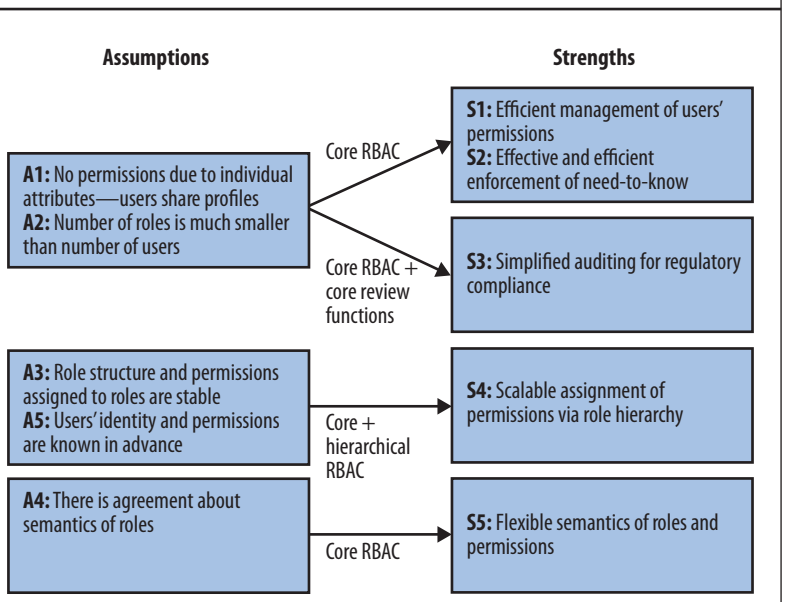


Figure 1. In cases where the assumptions implicit in role-based access control are true, RBAC has several strengths.

RBAC STRENGTHS

In cases where the assumptions are true, the RBAC model has several strengths:

- S1: Efficient management of large-scale users' permissions, in terms of both time and effort.
- 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.
- S3: Simplified auditing of users' permissions for regulatory compliance.
- S4: Scalable assignment of permissions via inheritance of permissions in role hierarchies.
- S5: Flexible semantics of roles and permissions.

Figure 1 shows how strengths S1-S5 follow from assumptions A1-A5 in the context of core RBAC and hierarchical RBAC.

Efficient access management

If A1-A3 are true, administering role permissions (including review features F6 and F7) is efficient in terms of time and effort. Role management involves frequent deletion of user-role assignments or reassignments, eventual updates to permission-role assignments, and rare updates to the role structure. This efficiency directly translates to economic benefits.[2,4]

Effective and efficient enforcement of need-to-know

An important internal control principle used to assure a satisfactory level of security is that of least privilege, or need-to-know. This aims to guarantee that users are neither under- nor overentitled—that is, users should have the permissions 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 an effective and efficient way to enforce the need-to-know principle because the assignment of users to roles (the *need* part) and the assignment of roles to permissions (the *know* part) jointly establish such connections.

Role hierarchy makes the assignment of permissions to users via inheritance a scalable task, representing a major potential strength of RBAC.

Simplified regulatory compliance

RBAC facilitates auditing; it is a convenient way to manage and document users' permissions because it makes visible all permissions that a user has by means of the roles assigned to that user. Consequently, RBAC effectively demonstrates compliance with security requirements established by legislation such as HIPAA, the Gramm-Leach-Bliley Act, and the Sarbanes-Oxley Act, and to industrial standards such as the Payment Card Industry Data Security Standard (PCI DSS).

While HIPAA explicitly recommends using RBAC, other regulations and standards 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 possible to comply with these requirements with other access control models, alternatives to RBAC tend to be more time-consuming and error-prone. For example, access control lists (ACLs) assign permissions directly to users, and these permissions are not necessarily constrained within groups.

Scalable inheritance of permissions via role hierarchy

In the absence of role hierarchy, there are two options for assigning many-to-many permissions to users.

The first is to duplicate permission assignments among roles. For example, if managers should have programmers' permissions, assign to the role "manager" permissions specific to both managers and programmers. However, this can substantially increase the number of role-permission assignments as well as lead to inconsistencies.

The second option is to not duplicate permissions across roles but let users accumulate permissions via assignment to several roles. For example, assign managers to both the "manager" and "programmer" roles to perform their duties. However, this will increase the number of user-role assignments and can lead to need-to-know violations.

In contrast, role hierarchy makes the assignment of permissions to users via inheritance a scalable task, representing a major potential strength of RBAC.

Flexible semantics of roles and permissions

RBAC imposes no restrictions on the semantics of roles and permissions. Therefore, semantics must be defined in the process of role engineering, and differ by type of application. For example, when RBAC is applied to an OS, roles tend to be coarse-grained and usually refer to classes of users, with network administrators agreeing on the semantics. In a university context, roles at this level could be "academic staff," "administrative staff," and "student."

The flexibility of roles becomes increasingly apparent when RBAC is applied to stand-alone business, enterprise-wide, and cross-enterprise applications. In such cases, roles can be used to assign any set of permissions that are shared by users, at any level of abstraction. For example, roles can be defined for business roles or technical roles, but also for users at a specific location, or for users performing a particular task.

The lack of predefined semantics for permissions in RBAC also provides flexibility. For example, permissions related to roles regarding databases are typically fine-grained and involve operations like insert, delete, or append when applied to objects like records and tables. Permissions related to roles for business applications can involve operations such as place and manage when applied to objects like purchase orders.

RBAC LIMITATIONS

Several phenomena observed in practice can limit RBAC's strengths.

Role explosion

- P1: Users' individuality, locality, and particularity give rise to roles with only a few members, contributing to role explosion.
- P2: There might be many dynamic context-specific attributes that affect users' permissions, contributing to role explosion.

Unexpected side effects of role hierarchy

- P3: Structuring and managing role hierarchies require a clear understanding of the inheritance of permissions to prevent unexpected side effects resulting in user under- or overentitlement.

Interoperability issues

- P4: For RBAC to be effective, the meaning of roles in terms of terminology and permissions must be shared across different departments, branches, or business partners; reaching consensus might not be trivial, giving rise to interoperability problems.
- P5: While the concept of role is intuitive, the RBAC standard is complex and evolving. This results in numerous interpretations of the RBAC model in practice, causing interoperability problems.

Rigidity in the face of modern business dynamics

- P6: The increasing need to share information in business-to-business relationships can lead to frequent changes affecting users' permissions; RBAC might become either unmanageable or lead to need-to-know violations.
- P7: An enterprise might not know which permissions users should have until the need actually arises, especially in emergency situations or when users have temporary responsibilities outside their normal roles; RBAC cannot deal with such dynamics.

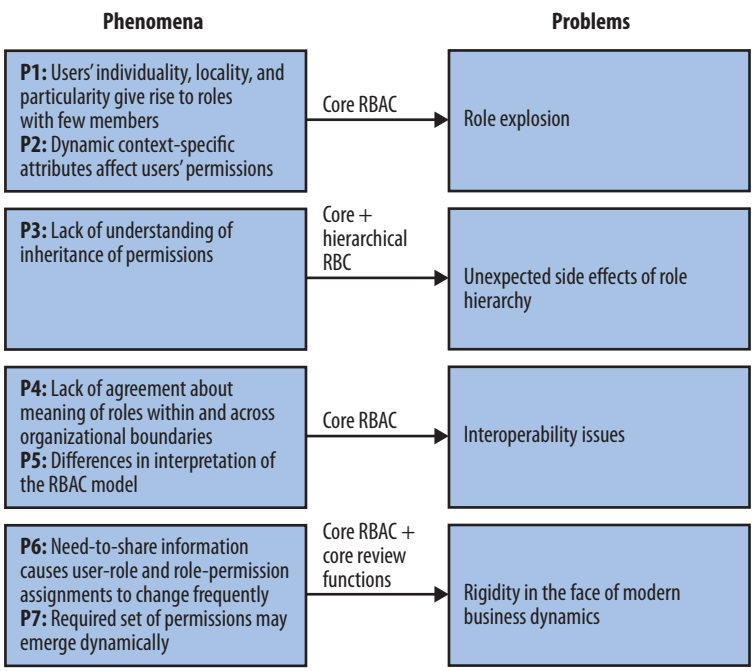


Figure 2. In cases where phenomena observed in the RBAC context of use are true, RBAC can cause problems.

Figure 2 lists the phenomena that limit RBAC strengths in the context of core RBAC (features F1-F5), core review functions (features F6-F7), and hierarchical RBAC (feature F8).

Role explosion

Assigning permissions via roles only can be a limitation as well as a strength because the number of roles can easily explode. There are two main causes of role explosion, which is a role engineering problem. Complex user attributes—including individuality, locality, and particularity—and dynamic context-dependent constraints can result in an impractical number of roles and, in extreme cases, more roles than users.

Complexity of user attributes. Different users with the same functional role can have different sets of permissions—they might need to perform different types of operations on different or even the same objects, depending on specific circumstances. The only way to accommodate *individuality* in RBAC is to create one role for each set of permissions. For example, the role "teller-at-probation" has more restrictive permissions than "teller-not-at-probation,"[9] and "part-time-healthcare-assistant" has more restrictive permissions than "full-time-healthcare-assistant."[10]

Different users with the same functional role can also have different sets of permissions depending on their *locality*. For example, the role "teller" might have different permissions for tellers located at places A and B, potentially giving rise to roles "teller-A" and "teller-B."[9] Role hierarchy does not solve the problem, as "teller" would have permissions common to A and B inherited by "teller-A" and "teller-B," which would need extra permissions to cope with location specifics.

Finally, different users with the same functional role might be permitted to perform the same set of operations but on different objects. This *particularity* requires creating several roles to accommodate different sets of permissions. For example, doctors should have permission to access only their own patients' data; giving them permission to access all patient data would make them overentitled, violating the need-to-know principle. Such a situation in a pure RBAC implementation could 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. As the "Core and Advanced RBAC" sidebar describes, *constrained RBAC* recognizes the need to impose restrictions either on the assignment of users to roles (to deal with static separation of duties) or on the assignment of sessions to roles (to deal with dynamic separation of duties). However, there are important dynamic context-dependent constraints when granting permissions to users apart from roles. For example, in practice it is common to allow users to exercise their permissions only during a certain period of time, such as during office hours: in these situations, the activation of roles during a session must comply with temporal constraints.[9]

Unexpected side effects of role hierarchy

This is both a role engineering and a role management problem.

According to the RBAC standard, a role hierarchy could support arbitrary hierarchies, but any hierarchy should comply with the bottom-up rule of inheritance. Structuring and maintaining hierarchies thus necessitate a clear understanding of the consequences of inheritance to avoid the side effects of over- or underentitlement.[11] Real-world hierarchies are complex, and their management requires expertise beyond the knowledge base of existing staff in many companies, compared with less sophisticated mechanisms such as ACLs.[4] This has economic implications given that more skilled staff are likely to be more expensive.

Ambiguous semantics in RBAC and multiple interpretations of the model can cause interoperability issues.

Furthermore, in practice, exceptions to the bottom-up inheritance rule occur. For example, at first glance it seems straightforward to place the role "project manager" higher in the role hierarchy than the role "programmer," in which case managers would inherit programmers' permissions. However, a project manager who lacks the technical skills to deploy executable code live probably should not have update permission over the production directory.[9]

Interoperability issues

Ambiguous semantics in RBAC and multiple interpretations of the model can cause interoperability issues, which is also both a role engineering and a role management problem.

Ambiguous semantics. Because RBAC does not impose any specific semantics on roles, stakeholders involved in implementing an RBAC application must reach consensus in terms of terminology (for example, what will the role "manager" mean?), and permissions (for example, what set of permissions should be assigned to the role "manager"?). However, reaching such agreement is not trivial in practice[7] and can lead to errors in granting access.[11]

Multiple interpretations of the RBAC model. While the concept of role is intuitive, the RBAC standard itself is complex and has been subject to various interpretations by practitioners, as well as by the research community.[12] Adopting the idea of RBAC in general without adhering to the specific RBAC standard is common, resulting in numerous implementations that are not interoperable.

Rigidity in the face of modern business dynamics

Modern business dynamics can make role management, especially in the presence of role hierarchies, challenging. Mergers, splits, outsourcing, and business partnerships can all affect the role structure and user-role and role-permission assignments.

Need-to-know versus need-to-share. Enterprises face two conflicting requirements: on one hand they seek to minimize access to information to meet growing security and compliance constraints (need-to-know), while on the other hand they want to maximize access to information to meet growing economic needs (need-to-share).[3]

The dynamics of current organizations make need-to-share hard to manage. A need-to-share policy increases the spectrum of permissions to be granted, and creates the problem of which permissions users should have until the need actually arises.[3] For example, third parties' users often need to perform operations on objects in a similar way to employees. However, it is harder to fit such users into roles because their responsibilities, job functions, and qualifications are usually not visible to the enterprise that grants permissions. Consequently, either new roles must be created to accommodate these users on an almost one-to-one basis or these users must be assigned roles that could grant them more permissions than they need to perform their duties.

Dynamic ad hoc access decisions. An RBAC implementation assumes that the conditions under which permissions can be granted to users are known in advance, leading to the decision to either allow or deny access. However, this binary access decision does not cope with unforeseen situations (such as emergencies) or temporary responsibilities (such as in downsizing periods[11]). In practice, especially in domains like healthcare, these assumptions do not always hold, and having a way to securely apply exceptions becomes important.

SOLUTIONS

INCITS' Cyber Security committee formed CS1.1, the Role-Based Access Control Working Group, to collect requirements for a next-generation RBAC standard, expected

to be released in late 2012. These requirements for the implementation and interoperability of RBAC include possible updates to the RBAC model, as defined by ANSI/INCITS 359-2004, and have been formally standardized as ANSI/INCITS 459-2011.[13]

ANSI/INCITS 459:2011 defines RBAC interaction functions for the syntactic and semantic exchange of RBAC information, allowing system-to-system offline interoperability;[14] this approach, to be incorporated in the new RBAC standard, addresses phenomena P4 and P6. Furthermore, ANSI/INCITS 459:2011 specifies "how to design RBAC products to conform to INCITS 359,"[14] which aims to address phenomenon P5 by increasing consistency among RBAC implementations.

The new RBAC model will incorporate dynamic attributes that constrain the role structure's expressing a set of more static attributes. The system will grant access based on the intersection of *P* and *R*,[15] 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).[7] This change directly addresses phenomena P1 and P2 and, indirectly, phenomenon P7. Another proposed solution to these phenomena is the use of dynamic roles. With this approach, the system uses object and user profiles—including static roles and attributes—and environmental status to determine the set of a user's permissions.[16] Moreover, Ana Ferreira and colleagues have suggested dealing with phenomenon P7 by letting the system grant users access on an exceptional basis, given that a justification is provided and logged.[17]

Phenomenon P3 materializes in RBAC implementations and therefore can be addressed in different ways. For instance, the SAP NetWeaver Identity Management module establishes role hierarchy by means of single and derived roles, with authorization checks to adjust derived roles. Bernard Stepien, Stan Matwin, and Amy Felty have proposed another way to deal with phenomenon P3 in the form of a mechanism in Extensible Access Control Markup Language (XACML), an OASIS standard, to more easily assess the effect of role inheritance.[18] In fact, implementing RBAC with XACML notation promotes portability of access policies and interoperability among different applications, thereby helping to address phenomenon P4.

Our analysis takes the perspective of the current ANSI/INCITS 359-2004 standard, and provides a thorough, yet concise, overview of the strengths of the RBAC model that follow from certain assumptions, and of phenomena we have observed in practice that might limit these strengths. We have also outlined proposed solutions to these limitations. Figures 1 and 2 can help practitioners determine whether the assumptions implicit in RBAC are true for their organization and whether they have observed the same phenomena, enabling them to anticipate RBAC strengths, identify problems, and diagnose possible reasons for unsuccessful RBAC implementations. C

References

1. R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed., John Wiley & Sons, 2008.
2. A.C. O'Connor and R.J. Loomis, *2010 Economic Analysis of Role-Based Access Control*, RTI project no. 0211876, Nat'l Inst. of Standards and Technology, 2010; http://csrc.nist.gov/groups/SNS/rbac/documents/20101219_RBAC2_Final_Report.pdf.
3. B. Schneier and M. Ranum, "Schneier-Ranum Face-Off: Is Perfect Access Control Possible?," *Information Security*, Sept. 2009, http://searchsecurity.techtarget.com/magazineContent/Schneier-Ranum-Face-Off-Is-Perfect-Access-Control-Possible.
4. M.P. Gallaher, A.C. O'Connor, and B. Kropp, *The Economic Impact of Role-Based Access Control*, RTI project no. 07007.012, Nat'l Inst. of Standards and Technology, 2002; www.nist.gov/director/planning/upload/report02-1.pdf.
5. M. Kuhlmann, D. Shohat, and G. Schimpf, "Role Mining—Revealing Business Roles for Security Administration using Data Mining Technology," *Proc. 8th ACM Symp. Access Control Models and Technologies* (SACMAT 03), ACM, 2003, pp. 179-186.
6. C.L. Smith et al., *A Marketing Survey of Civil Federal Government Organizations to Determine the Need for a Role-Based Access Control (RBAC) Security Product*, SETA Corp., 1996; http://csrc.nist.gov/groups/SNS/rbac/documents/cost_benefits/seta.ps.
7. A.H. Karp, H. Haury, and M.H. Davis, "From ABAC to ZBAC: The Evolution of Access Control Models," *Information Systems Security Assoc. J.*, Apr. 2010, pp. 22-30.
8. D.F. Ferraiolo, D.R. Kuhn, and R. Chandramouli, *Role-Based Access Control*, Artech House, 2003.
9. B. Hilchenbach, "Observations on the Real-World Implementation of Role-Based Access Control," *Proc. 20th Nat'l Information Systems Security Conf.*, Nat'l Inst. of Standards and Technology, 1997, pp. 341-352; http://csrc.nist.gov/nissc/1997/proceedings/341.pdf.
10. AHIMA/HIMSS HIE Privacy & Security Joint Work Group, "The Privacy and Security Gaps in Health Information Exchange," white paper, American Health Information Management Assoc./Healthcare Information and Management System Soc., 2011; www.himss.org/content/files/201106_AHIMA_HIMSS.pdf.
11. K.D. Gordon et al., "Accounting Data Security at JEA," presentation, American Accounting Assoc. Ann. Meeting, 2011; 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*, Nov./Dec. 2007, pp. 41-49.
13. ANSI/INCITS 459-2011, *Information Technology—Requirements for the Implementation and Interoperability of Role-Based Access Control*, American Nat'l Standards Inst./Int'l Committee for Information Technology Standards, 2011.
14. E. Coyne and T. Weil, "An RBAC Implementation and Interoperability Standard: The INCITS Cyber Security 1.1 Model," *IEEE Security and Privacy*, Jan./Feb. 2008, pp. 84-87.

15. D.R. Kuhn, E.J. Coyne, and T.R. Weil, "Adding Attributes to Role-Based Access Control," *Computer*, June 2010, pp. 79-81.
16. R. Fernandez, *Enterprise Dynamic Access Control Version 2 Overview*, US Navy, 2006; http://csrc.nist.gov/rbac/EDACv2overview.pdf.
17. A. Ferreira et al., "How to Securely Break into RBAC: The BTG-RBAC Model," *Proc. Ann. Computer Security Applications Conf.* (ACSAC 09), IEEE, 2009, pp. 23-31.
18. B. Stepien, S. Matwin, and A. Felty, "Advantages of a Non-Technical XACML Notation in Role-Based Models," *Proc. 9th Ann. Int'l Conf. Privacy, Security, and Trust* (PST 11), IEEE, 2011, pp. 193-200.

Virginia N.L. Franqueira *is the founder of VF InfoSec Consulting, an information security consultancy based in the UK. She received a PhD in computer science from the University of Twente, the Netherlands. Her research interests include risk assessment, security engineering, and network security. Franqueira carried out the research described in this article, which was funded by Sentinels (www.sentinels.nl), as a postdoctoral researcher at the University of Twente. 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. His research interests include requirements engineering, risk assessment, and design science methodology. Wieringa received a PhD in computer science from VU University, Amsterdam. Contact him at r.j.wieringa@utwente.nl.*