ROLE-BASED ACCESS CONTROL ON THE WEB USING LDAP

Joon S. Park
Information and Software Engineering Department
George Mason University
jpark@itd.nrl.navy.mil

Gail-Joon Ahn
College of Information Technology
University of North Carolina at Charlotte
gahn@uncc.edu

Ravi Sandhu
Information and Software Engineering Department
George Mason University
sandhu@gmu.edu

Abstract This paper gives a framework for how to leverage Lightweight Directory Access Protocol (LDAP) to implement Role-based Access Control (RBAC) on the Web in the *server-pull* architecture. LDAP-based directory services have recently received much attention because they can support object-oriented hierarchies of entries in which we can easily search and modify attributes over TCP/IP. To implement RBAC on the Web, we use an LDAP directory server as a role server that contains users' role information. The role information in the role server is referred to by Web servers for access control purposes through LDAP in a secure manner (over SSL). We provide a comparison of this work to our previous work, RBAC on the Web in the *user-pull* architecture.

Keywords: Access Control, LDAP, RBAC, Web Security

1. Introduction

The emergence of Web technology has changed life styles and the world economy. It enables us to access various information regardless of the distance between service providers and requesters. Web technology has continued its rapid evolution, most recently towards increasingly secure transactions on enterprise-wide infrastructure. However, information security problems have become more important as the world uses more information systems, since the security of information systems directly or indirectly affect organizations, which rely on information technology. An organization's enterprise intranet security depends on largely on access control management. RBAC (Role-based Access Control [SCFY96]) has been accepted as a proven technology for strong and efficient access control. As the vehicle of access control management, RBAC can be used to provide the services that focus on how users' role information can be used instead of users' identities.

Park and Sandhu identified two different approaches for obtaining user's attributes[1] on the Web: *user-pull* and *server-pull* architectures [PS99]. They classified the architectures based on "Who pulls the user's attributes?" In the user-pull architecture, the user pulls her attributes from the attribute server and then presents them to the Web servers, which use those attributes for their purposes. In the server-pull architecture, each Web server pulls user's attributes from the attribute server as needed and uses them for their purposes. To show the feasibility of those approaches, RBAC (as one example of secure attribute services) on the Web has been implemented in the user-pull architecture and described in our previous papers [PS99b, PSG99, ASKP00]. In this paper, we describe how we implement RBAC on the Web in the server-pull architecture using commercial off-the-shelf (COTS) technologies, such as LDAP (Lightweight Directory Access Protocol [HSG99]) and SSL (Secure Socket Layer [WS96, DA99]). Furthermore, we compare this work with our previous work, based-on our hands-on experiences.

This paper is organized as follows. Section 2 briefly introduces RBAC and the existing technologies that we use for our server-pull implementation in this paper. In Section 3, we describe two different operational architectures of RBAC on the Web. Section 4 describes how we implement RBAC on the Web in the server-pull architecture using existing well-known Web technologies. Section 5 compares the server-pull and user-pull RBAC architectures based on our hands-on experiences. Finally, Section 6 concludes this paper.

[1]An attribute is a particular property of an entity, such as a role, access identity, group, or clearance.

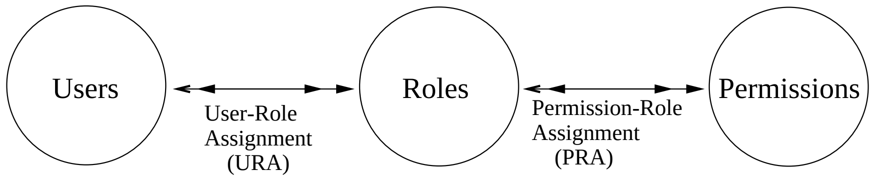


Figure 1. A Simplified RBAC Model

2. Related Technologies

2.1. Role-based Access Control (RBAC)

Role-based Access Control (RBAC [SCFY96]) has rapidly emerged in the 1990s as a technology for managing and enforcing security in large-scale enterprise-wide systems. The basic notion of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles. Figure 1 shows a simplified RBAC model. RBAC ensures that only authorized users are given access to certain data or resources. This simplifies security management. Intuitively, a user is a human being or an autonomous agent, a role is a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role. Permissions are simply treated as abstract token. A user can be a member of many roles and a role can have many users (denoted by double arrows in Figure 1). Similarly, a role can have many permissions and the same permissions can be assigned to many roles.

System administrators can create roles, grant permissions to those roles, and then assign users to the roles on the basis of their specific job responsibilities and policy. Therefore, role-permission relationships can be predefined, making it simple to assign users to the corresponding permissions through the roles. Without RBAC, it is difficult (especially, in a large enterprise system) to determine and change what permissions have been authorized for what users.

2.2. Lightweight Directory Access Protocol (LDAP)

User information is often fragmented across the enterprise, leading to data that is redundant, inconsistent, and expensive to manage. Directories are being viewed as one of the best mechanisms to make enterprise information available to multiple different systems within an organization. Directories also make it possible for organizations to access infor-

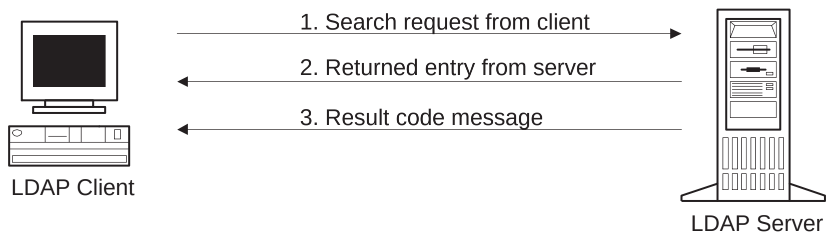


Figure 2. LDAP Operation

mation over the Internet. The most common information stored in a directory service is information about users on a network; this can include user id, passwords, assigned groups, or network access rights. In order to retrieve the information, a directory access protocol is used to convey the entries from directory-oriented server. The trend towards directories has been accelerated by the recent growth of LDAP (Lightweight Directory Access Protocol [HSG99]).

LDAP is a protocol that enables X.500 based directories to be read through Internet clients. It was developed by the University of Michigan and the Internet Engineering Task Force (IETF) as a set of network services to provide object management and access over TCP/IP networks. LDAP is also a message-oriented protocol. When an LDAP client needs a specific entry in an LDAP server, the LDAP client generates an LDAP message containing a request and sends this message to the LDAP server. The server retrieves the entry from its database and sends it to the client in an LDAP message. It also returns a result code to the client in a separate LDAP message to terminate the session. Figure 2 shows this interaction between the LDAP server and client. With this feature of directory services, we use an LDAP server and client for our server-pull RBAC implementation.

Although we use LDAP between Web servers and role server for RBAC in the server-pull architecture, it is also possible to use LDAP for the user-pull architecture, where clients can retrieve their roles from the role server via LDAP and present them to the Web servers.

2.3. Secure Socket Layer (SSL)

SSL [WS96, DA99] was introduced with the Netscape Navigator browser in 1994, and rapidly became the predominant security protocol on the Web. Since the protocol operates at the transport layer, any program that uses TCP (Transmission Control Protocol) is ready to use SSL connections. The SSL protocol provides a secure means for establishing an encrypted communication between Web servers and browsers. SSL also supports the authentication service between Web servers and browsers.

SSL uses X.509 certificates. Server certificates provide a way for users to authenticate the identity of a Web server. The Web browser uses the server's public key to negotiate a secure TCP connection with the Web server. Optionally, the Web server can authenticate users by verifying the contents of the client certificates.

Even though SSL provides secure communications between Web servers and browsers on the Web, it cannot protect against end-system threats. For instance, after a user, Alice, receives her attributes from a Web server over SSL, it does not mean that the attributes are still safe. In other words, once the user receives some attributes from the server over SSL, an attacker is able to change the attributes in the user's machine, because SSL does not support security in the user's end system. However, as we will see later in this paper, SSL can be used as part of our implementation to protect information on the Web.

3. Operational Architectures

Park and Sandhu identified two different approaches for obtaining a user's attributes on the Web, especially with respect to the user-pull and server-pull architectures, in which each architecture has user-based and host-based modes [PS99]. In this section, we embody those general approaches for RBAC on the Web with specific components and relationships. Basically, there are three components in both architectures: client, Web server, and role server. These components are already being used on the Web. Clients connect to Web servers via HTTP using browsers. The role server is maintained by an administrator and assigns users to the roles in the domain [SP98]. Detailed technologies (such as authentication, role transfer and protection, and verification) to support these architectures depend on the applications that are used.

3.1. RBAC in User-Pull Architecture

In the user-pull architecture, a user, say Alice, pulls her roles from the role server and then presents them to the Web servers, as depicted in a UML (Unified Modeling Language [BJR98]) collaborational diagram in Figure 3. We call this a user-pull architecture, since the user pulls her roles from the role server, in which roles are assigned to the users in the domain. HTTP is used for the user-server interactions with standard Web browsers and Web servers.

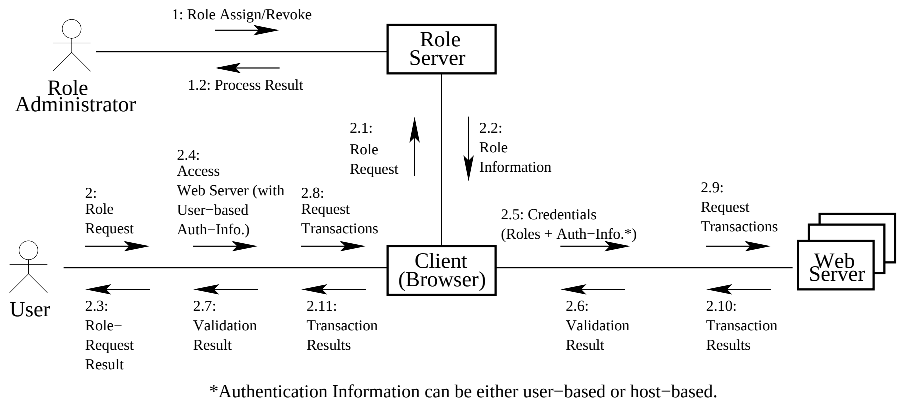

1: Role Assign/Revoke
1.2: Process Result
Role Administrator
Role Server
2.1: Role Request
2.2: Role Information
2: Role Request
2.4: Access Web Server (with User−based Auth−Info.)
2.8: Request Transactions
2.5: Credentials (Roles + Auth−Info.*)
2.9: Request Transactions
User
Client (Browser)
Web Server
2.3: Role−Request Result
2.7: Validation Result
2.11: Transaction Results
2.6: Validation Result
2.10: Transaction Results
*Authentication Information can be either user−based or host−based.

Figure 3. A Collaborational Diagram for the User-Pull Architecture

In the user-pull-host-based mode, the user needs to download her roles from the role server and store them in her machine, which has her host-based authentication information, such as IP numbers[2]. Later, when the user wants to access the Web server, which requires proper authentication information and roles, her machine presents that information to the Web server. After client authentication and role verification, the Web server uses the roles for RBAC. However, since this mode is host-based, it cannot support high user mobility, while it may support more convenient services than the user-based mode, which requires user's co-operation (e.g., typing in passwords).

On the other hand, the user-pull-user-based mode supports high user mobility. The user can download her roles to her current machine from the role server. Then, she presents those roles to the Web server along with her user-based authentication information, such as her passwords. After user authentication and role verification, the Web server uses the roles for RBAC. In this mode, the user can use any machines, which support HTTP, as long as she has right user-based authentication information (e.g., passwords).

In this user-pull architecture, we must support the binding of roles and identification for each user. For instance, if Alice presents Bob's

[2]Address-based authentication is a convenient authentication mechanism because the authentication process is transparent to users, but such a method is not always desirable. For example, if the user's IP address is dynamically assigned to her computer whenever she connects to the Internet, or the user's domain uses a proxy server, which provides the same IP numbers to the users in the domain, this is not a proper authentication technique. In addition, we cannot avoid IP spoofing, which is a technique for gaining unauthorized access by sending messages to a computer with a trusted IP address.

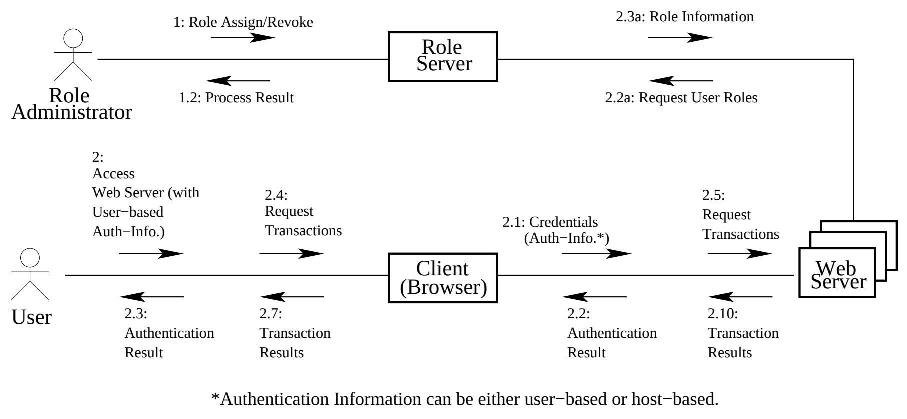

1: Role Assign/Revoke
1.2: Process Result
Role Administrator
Role Server
2.3a: Role Information
2.2a: Request User Roles
2: Access Web Server (with User−based Auth−Info.)
2.4: Request Transactions
2.1: Credentials (Auth−Info.*)
2.5: Request Transactions
User
Client (Browser)
Web Server
2.3: Authentication Result
2.7: Transaction Results
2.2: Authentication Result
2.10: Transaction Results
*Authentication Information can be either user−based or host−based.

Figure 4. A Collaborational Diagram for the Server-Pull Architecture

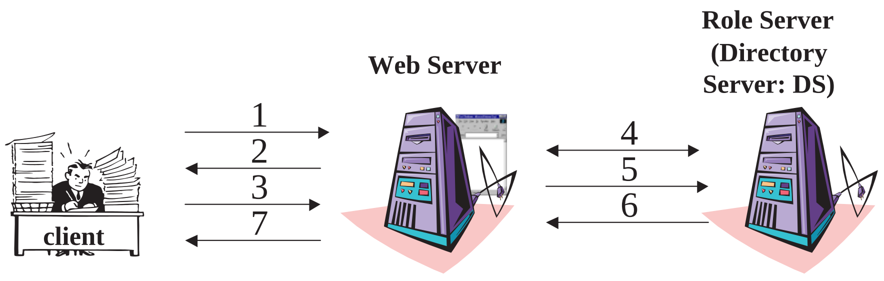
roles with her authentication information to the Web server, she must be rejected. General approaches for binding user attributes and their identities are discussed by Park and Sandhu in [PS00].

3.2. RBAC in Server-Pull Architecture

In the server-pull architecture, each Web server pulls the user's roles from the role server as needed and uses them for RBAC as depicted in a UML collaborational diagram in Figure 4. We call this a server-pull architecture, since the server pulls the user's roles from the role server. HTTP is used for the user-server interactions with standard Web browsers and Web servers. If the role server provides users' roles securely, the Web server can trust those roles and uses them for RBAC.

In this architecture, the user does not need access to her roles. Instead, she needs only her authentication information. In the server-pull-host-based mode, she presents her host-based authentication information (e.g., IP numbers) to the Web server. Role obtaining mechanism is transparent to the user, while limiting the portability. However, in the server-pull-user-based mode, Alice presents her user-based authentication information (e.g., passwords) to the Web server. This supports high portability, while it requires the user's cooperation (e.g., typing in passwords). After user authentication, the Web server downloads the user's roles from the role server and uses them for RBAC.

We describe an implementation of this architecture in both user and host-based modes in Section 4. A detailed comparison of the user-pull and server-pull architectures is discussed in Section 5.



1 Authenticate client (using password or client certificate)
2 Display the initial page
3 Request resources by clicking a link
4 Establish SSL between DS and Web server
5 LDAP over SSL; Request client ’s role information to DS
6 LDAP over SSL; Return client ’s role information to Web server
7 Display appropriate resources after authorization check
based on client ’s roles

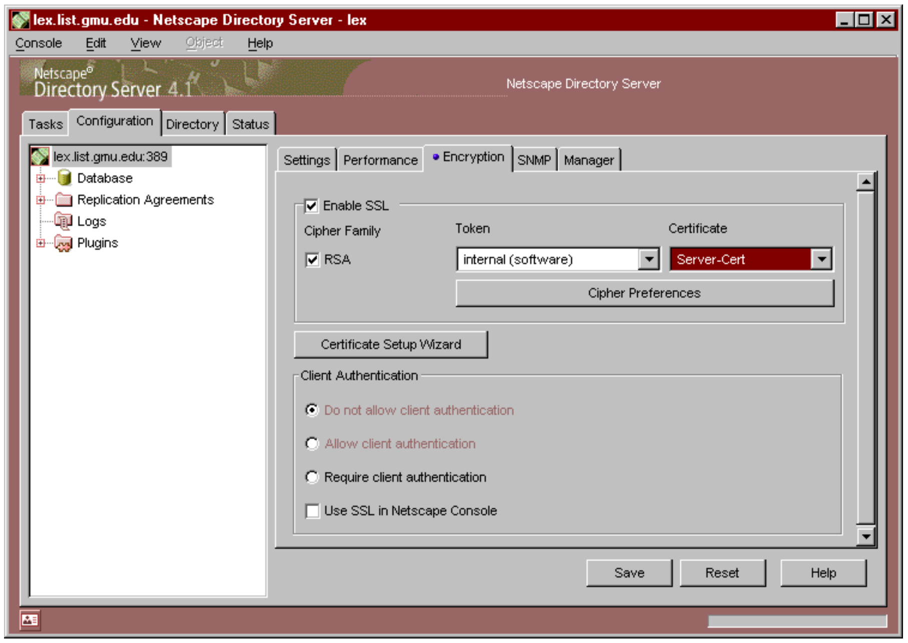
Figure 5. Transaction Procedures for RBAC in Server-Pull Architecture

4. Implementing RBAC on the Web in the Server-Pull Architecture

We use LDAP to communicate between the directory-oriented role server and Web servers. In our implementation, we use Netscape Directory Server as an LDAP-supporting role server and use its group objects as users' role attributes. This directory-oriented role server contains users' role information to be used for access control by Web servers.

The basic scenario of our implementation is that a client presents her authentication information to a Web server and then, after a successful authentication process, the Web server gets the client's role information from the role server via LDAP to use those roles for RBAC services in the Web server.

For this purpose, we set up a computing environment based on the server-pull architecture (see Section 3.2) as shown in Figure 4. Figure 5 shows the transaction procedures of our experiment. We have three major components: Web server, role server, and client. The role server keeps URA (User-Role Assignment) information. The Web server contains resources, which require particular roles to be accessed. The Web server also contains a PRA (Permission-Role Assignment) table, which



[✓] Enable SSL

Cipher Family	Token	Certificate
[✓] RSA	internal (software)	Server-Cert

Client Authentication

[✓ Do not allow client authentication]
Allow client authentication
Require client authentication
[ ] Use SSL in Netscape Console

Figure 6. Directory Server: Enabling SSL

specifies the required roles for particular resources in the Web server. This table is referenced to check if the user has proper roles to access particular resources in the Web server. Clients use Web browsers to connect Web servers over HTTP or HTTPS.

The detailed transaction procedures are as follows. A client presents her authentication information to a Web server. We can use username/passwords, IP numbers, client certificates[3], or other authentication techniques for this purpose. The Web server authenticates the user using proper authentication mechanism. Once a user is authenticated by the Web server successfully (otherwise, the user gets an error message), the Web server triggers the CGI (Common Gateway Interface) scripts that call an LDAP client software. This LDAP client software sends a search query to the LDAP server, which retrieves the user's roles from the directory server through SSL. The retrieved roles are sent back to the

[3]To use client certificates, which is an optional operation of SSL, we need to configure the Web server to accept and understand particular client certificates. Note that this SSL channel is optional and independent from the SSL connection between Web servers and the role server.

LDAP client in the Web server during the same SSL session. When the user requests an access to the resources, which requires particular roles to be accessed, by clicking the corresponding link in the initial page, the Web server compares the user's roles (that it pulled from the role server) with the ones in its PRA table. If the user has corresponding roles in the table, the Web server allows the user to access the resources.

Figure 6 is a snapshot showing the directory server (role server in our case) configuration. This directory server is running a process called *slapd* (LDAP server daemon) to allow requests from LDAP clients. We configure this server to have two network ports: one is for regular port and the other is for secure communications. For the secure communications, we need to enable SSL establishment between the LDAP client (installed in the Web server in our implementation) and directory server, more specifically the LDAP server.

5. Discussion

In this section, based on our hands-on experiences, we discuss the tradeoffs between the user-pull (implemented and described in [PS99b, PSG99, ASKP00]) and the server-pull (implemented and described in Section 4) architectures using different technologies for RBAC on the Web. In the user-pull architecture, the user pulls her roles from the role server, and then presents the role information to the Web servers along with her authentication information. In the server-pull architecture, the user presents her authentication information to the Web servers, which pulls the user's role information from the role server for RBAC after a successful authentication.

The user-pull architecture requires a user's cooperation to obtain her roles, but it enhances the Web server performance. Once the user obtains her roles, she can use them in many different sessions even in different Web servers until the roles expire. This increases the reusability. By this feature, the user-pull architecture is a good solution for the applications, especially, where users' convenience is required for maintaining and using their roles frequently in diverse Web sites, such as pay-per-access. However, the longevity of the roles decreases the freshness of the roles. For instance, if the user already pulled her roles, the updated version in the role server would not become effective instantly. Namely, an additional synchronization process is required. Thus, when the dynamic role update is critical, the role server should push the status change of users' roles, such as role revocation, to the Web servers for updated information.

	User-Pull Architecture	Server-Pull Architecture
User's Convenience	Low	High
Performance	High	Low
Reusability	High	Low
Role Freshness	Low	High
Single-Point Failure	Low	High

Table 1. A Comparison of User-Pull and Server-Pull Architectures

The server-pull architecture requires the Web server's cooperation for obtaining the user's role information - which decreases the Web server performance - from the role server. In this architecture, the Web server retrieves the user's role information from the role server for each session. This increases the freshness of the roles, so the information update (e.g., role revocation) is more efficient than the user-pull architecture, because all the roles are stored in the role server and pulled by the Web servers on demand. By this feature, the server-pull architecture is a good solution for the applications, especially, where dynamic role update is critical, such as stop-payment in electronic transactions. However, it decreases reusability and increases the single-point failure vulnerability because every session requires an access to the role server. We summarize the comparison of the user-pull and server pull architectures in Table 1.

6. Conclusion

We have developed the system architectures and mechanisms for achieving RBAC on the Web. We have also described several proof-of-concept implementations to demonstrate the feasibility of our approaches. After we had practical experiences in both the user-pull and server-pull architectures of RBAC on the Web, we compared the tradeoffs of both approaches. A comprehensive description and analysis for RBAC on the Web using different technologies in different architectures is available in [PSA01]. We believe that our work can be an important step towards offering strong and efficient security management based on users' roles on the Web.

References

[ASKP00] Gail-Joon Ahn, Ravi Sandhu, Myong Kang, and Joon Park. Injecting RBAC to Secure a Web-based Workflow System. In *Proceedings of 5th ACM Workshop on Role-based Access Control*, pages 1–10, Berlin, Germany, July 2000.

[BJR98] Grady Booch, Ivar Jacobson, and James Rumbaugh. *The Unified Modeling Language User Guide*. Addison-Wesley, 1998.

[DA99] T. Dierks and C. Allen. *The TLS (Transport Layer Security) Protocol.* RFC 246, January 1999.

[HSG99] Timoth Howes, Mark Smith, and Gordon Good. *Understanding and Deploying LDAP Directory Services.* Macmillan Technical Publishing, 1999.

[Neu94] B. Clifford Neuman. Using Kerberos for Authentication on Computer Networks. *IEEE Communications*, 32(9), 1994.

[Net99] Netscape Communications Corporation. Netscape Directory Server 4.1 Deployment Guide. *http://developer.netscape.com/docs/manuals/directory/dir40/de/contents.htm*, 1999.

[PS00] Joon S. Park and Ravi Sandhu. Binding Identities and Attributes Using Digitally Signed Certificates. In *Proceedings of 16th Annual Computer Security Applications Conference (ACSAC)*, New Orleans, Louisiana, December 2000.

[PS00b] Joon S. Park and Ravi Sandhu. Secure Cookies on the Web. *IEEE Internet Computing*, 4(4), 36-44, July-August 2000.

[PS99] Joon S. Park and Ravi Sandhu. Smart Certificates: Extending X.509 for Secure Attribute Services on the Web. In *Proceedings of 22nd National Information Systems Security Conference (NISSC)*, Crystal City, Virginia, October 1999.

[PS99b] Joon S. Park and Ravi Sandhu. RBAC on the Web by Smart Certificates. In *Proceedings of 4th ACM Workshop on Role-based Access Control*, pages 1–9, Fairfax, Virginia, October 1999.

[PSA01] Joon S. Park, Ravi Sandhu, and Gail-Joon Ahn. RBAC on the Web. *ACM Transactions on Information and Systems Security*, 4(1), February 2001.

[PSG99] Joon S. Park, Ravi Sandhu, and SreeLatha Ghanta. RBAC on the Web by Secure Cookies. In *Proceedings of 13th Annual IFIP11.3 Conference on Database Security*, Seattle, Washington, July 1999.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based Access Control Models. *IEEE Computer*, 29(2):38–47, February 1996.

[SP98] Ravi Sandhu and Joon S. Park. Decentralized User-Role Assignment for Web-based Intranets. In *Proceedings of 3rd ACM Workshop on Role-based Access Control*, pages 1–12, Fairfax, Virginia, October 1998.

[WS96] D. Wagner and B. Schneier. Analysis of the SSL 3.0 Protocol. In *Proceedings of 2nd USENIX Workshop on Electronic Commerce*, Oakland, California, November 1996.

[YLS96] Nicholas Yialelis, Emil Lupu, and Morris Sloman. Role-based Security for Distributed Object Systems. In *Proceedings of IEEE Fifth Workshops on Enabling Technology: Infrastructure for Collaborative Enterprise.* 1996.