Extended Request-oriented Role Access Control Model for Web Applications

Dmitry KONONOV* and Sergey ISAEV
ICM SB RAS, Akademgorodok 50/44, Krasnoyarsk, 660036, Russia

*Corresponding author

Keywords: Web, Security, Access control, Roles, RBAC.

Abstract. This work is dedicated to developing security access control model for web applications. Our work is focused on RBAC model described by Ferraiolo and Kuhn [1992] and Sandhu [1998]. This article describes the new request-oriented RBAC model, which allows flexible access control using web request path and parameters. The new model is a development of our previous extended path-based RBAC model and provides additional access control capabilities. Applying this new model allows reducing security risks for web applications.

Introduction

Today modern Web applications and services are affected by several security issues. Computer security is becoming increasingly important area of research. According to Symantec security research [1], there are high security risks for web applications. Web applications security is a complex problem with several aspects. Security policy allows specifying necessary restrictions for access control. Access control is accomplished by security model restrictions. Applying appropriate security model leads to reducing risks of successful attacks.

Widely known security models include discretionary, mandatory, and role-based [2, 3]. In this work we describe security model based on Role-based access control model (RBAC) [4]. Role access control represents access rights control from subjects to objects grouped by some characteristics named roles. Original role-based access control model does not take into account web applications features [5], particularly hierarchic requests. Also assigning permissions is limited to roles only. This work describes request-oriented RBAC model, which improves original RBAC model and allows flexible access control using request path (URI) and path parameters.

Security Models

Currently, there are several security access control models. Some of them include access control accomplished by discretionary matrix, mandatory levels, and role-based.

Discretionary security models are based on access control from subjects to objects by using access control lists or access matrix. This family include security model such as Harrison-Ruzzo-Ulman [6], typed access matrix [7], Take-Grant [8].

Mandatory access control—access control from subjects to objects based on assigned confidentiality label for information contained in the objects and permission entities to access information with such level of confidentiality. An example of the mandatory model is Bell-LaPadula [2]. Classic Bell-LaPadula model describes conditions under which the computer system cannot initiate information flows from objects with high level of confidentiality to objects with lower level of confidentiality.

Role-based access control [4] is a further development of discretionary access control policy: permissions to system objects are grouped according to certain characteristics, forming roles. Roles are intended to manage access control rules in a more simple way. Modern web applications and web services use role-based access control models because of flexibility [9, 10]. Some authors use hybrid models combining access control and information flow control [11]. Other researchers describe
automated verification methods [12]. There is a temporal RBAC modification with time-based constraints [13].

These models do not take into account the web applications features, in particular, hierarchical structure of requests and web-specific protocols parameters. The paper describes the adapted request-oriented role-based security model that eliminates these problems. This work is based on our previous security model that allows access control using request path [14].

**Role-based Access Control**

The original role-based access control model [4] defines a set of elements:

\(<U, R, P, S, UA(U), PA(R), user(S), roles(S)\),

where:

- \(U\) – set of users;
- \(R\) – set of roles;
- \(P\) – set of access permissions;
- \(S\) – set of user sessions;

\(UA: U \rightarrow 2^R\) – function assigning for each user a variety of roles to which he can be authorized;

\(PA: R \rightarrow 2^P\) – function assigning for each role set of access permissions, while \(\forall p \in P, \exists r \in R\) such that \(p \in PA(r)\);

\(user: S \rightarrow U\) – function defining for each user session, on whose behalf it is authorized;

\(roles: S \rightarrow 2^R\) – function defining for user a variety of roles for which he is authorized with current session; at the same time \(\forall s \in S\) satisfies the condition \(roles(s) \subseteq UA(user(s))\).

The model \(RBAC_1\) is defined as \(RBAC_0\), at the same time introducing a role hierarchy (RH).

**Request-oriented Role-based Access Control**

To existing \(RBAC_1\) model's elements "user", "role", "permission", "session" we added new elements taking into account web application features: "token", "request", "request parameter". The new model defines a set of elements:

\(<U, R, P, S, Tk, Rq, RqP, UA(U), PA(R), RqA(P), RqPA(P), user(S), roles(S), token(Tk), requests(S), params(Rq)\>.

**Definition 1.** Token (Tk) is a set of user attributes that allow him to carry out authentication in a system. Token is a pair <name, password>, or pair <public key, private key>.

We define a function \(token()\) mapping token to user it belongs: \(token: T \rightarrow U\).

**Definition 2.** Request (Rq) is a set of information sent by the client to HTTP server. The request contains a set of headers, a unique resource identifier (URI), a set of parameters name/value, and a request payload (body).

Let’s define a function \(requests()\) mapping a session to requests allowed in this session: \(requests: S \rightarrow 2^{|S|}\).

Request belongs to the session, one session can handle multiple requests. Requests and permissions are tied by many-to-many relationship. On top of requests \(Rq\) inclusion relation is defined.

**Definition 3.** Request A includes a request B (\(B \leq A\)), if the path of unique resource identifier (URI) of request A contains the path of unique identifier of the resource request B with the initial position in the line within the same namespace, with \(len(B_{path}) \leq len(A_{path})\), where \(len(x)\) – length of the string x.

Inclusion relation has the following properties:

- Reflexive: \(\forall rq \in Rq: rq \leq rq\),
- Antisymmetric: \(\forall rq, rq' \in Rq: ((rq \leq rq') \& (rq' \leq rq)) \rightarrow rq = rq'\),
- Transitive: \(\forall rq_1, rq_2, rq_3 \in Rq: ((rq_1 \leq rq_2) \& (rq_2 \leq rq_3)) \rightarrow rq_1 \leq rq_3\).
Thus, the inclusion relation on top of requests set \( Rq \) defines non-strict partial order.

Next, we define a function \( RqA() \) mapping permissions to multiple requests \( RqA : P \rightarrow 2^{Rq} \).

**Definition 4.** Requests hierarchy (\( RqH \)) is an inclusion relation defined on top of requests \( Rq \). For any \( p \in P \) the following condition is true: if \( rq, rq' \in Rq, rq \in RqA(p) \), and \( rq \leq rq' \), then \( rq' \in RqA(p) \).

Thus, the definition 4 makes it possible a flexible access control for individual requests and all children requests.

**Definition 5.** Request parameter (\( RqP \)) is a set of pairs key/value of HTTP request, which belongs to request \( Rq \) and used to transfer additional data to a specified location.

Here and after, we assume "request parameter" and "parameter" are the same. Parameter belongs to the request, one request can have multiple parameters. We define two new functions:

- \( RqPA() \) mapping permissions to multiple parameters: \( RqPA : P \rightarrow 2^{RqP} \);
- \( params() \) mapping requests to multiple parameters: \( params : Rq \rightarrow 2^{RqP} \).

Additional parameters restrictions can be made using explicit function mapping and special filtering language capabilities.

Figure 1 shows elements of new request-oriented role-based access control security model.

![Figure 1. Elements of request-oriented RBAC model.](image)

**Application of Model**

Security model described in this paper can be used in a wide range of applications. To apply this security model the system must meet the following requirements:

- centralized access control – access control is carried out only in a single module without delegating to other units or systems;
- principle of least privilege – provide the user only minimal set of privileges necessary for his work;
- separation of duties support – tasks processed in the system may require multiple users to process one operation;
- possibility of decomposition the system into separate components, which can be accessed using a variety of URIs, which are unique within the system.

When the above requirements are met, the system may be divided into different parts, each of them is uniquely identified by a URI. URI paths are described as access control elements and used to define a set of requests \( R_q \). This set includes all client requests and API calls provided by the system. Using \( R_q \) request hierarchy is created that reflects the interaction and dependence between components.

In order to apply security model it is necessary to create a set of roles \( R \). The role defines a set of permissions for users. Examples of roles: "user", "moderator", "registrar", "administrator".

Next, the system should have defined set of permissions \( P \). Permissions define specific action or operation in the system, for example, "create new user", "delete the document," etc. One role can have many permissions. One permission can be assigned to many roles. Assigning permissions for roles is performed by function \( PA() \). Permissions are non-overlapping and consistent. Elements from permissions \( P \) map to a subset of requests \( R_q \) using function \( R_qA() \). One permission can have multiple requests. One request can be assigned to many permissions. Also elements from permissions \( P \) map to a subset of request parameters \( R_qP \) using function \( R_qPA() \). One permission can have multiple parameters. One parameter can be assigned to many permissions.

To maintain users a set of users \( U \) is created, which have assigned permissions from \( P \) using mapping function \( UA() \). One user can have multiple roles. One role can be assigned to multiple users.

To identify a user the model includes a set of tokens \( T_k \). The elements of the set are pairs of <username, password>, or <public key, private key>. Each user can have multiple tokens. The token belongs to one user.

Once authenticated, authorized work of users is carried out by sessions. A session is a set of authorized user data, including a set of roles to which the user is authorized. Users can create multiple sessions. A session belongs to one user. Sessions can have additional data related to authorization or specific operation.

As an example, we describe application of extended request-oriented RBAC for publication system. The system allows users to create articles for public reading. Registered users can create and edit their articles. Editors have the ability to edit articles created by users. The administrator has access to all sections, including user administration. The system has two users, who write articles: Alice and Bob. John is an editor, and Martin is a system administrator. In addition, Martin can work as editor. The system provides a special role for anonymous users for public reading without editing articles.

Users \( U \): {Anonymous, Alice, Bob, John, Martin}
Roles \( R \): {Viewer, User, Editor, Administrator}
Permissions \( P \): {"view article", "create article", "edit own articles", "edit all articles", "user management", "access control", "system maintenance"}
Requests \( R_q \): {
  /articles/list,
  /articles/view,
  /manage/articles/list,
  /manage/articles/create,
  /manage/articles/edit,
  /manage/users/list,
  /manage/users/create,
  /manage/users/edit,
  /manage/permissions/roles,
  /manage/permissions/acl,
  /manage/system/settings,
  /manage/system/maintenance\}
Request parameters \( R_qP \): {
  articleID,
}

72
Roles assignment for users:
- UA(Anonymous) → {Viewer}
- UA(Alice) → {User}
- UA(Bob) → {User}
- UA(John) → {Editor}
- UA(Martin) → {Editor, Administrator}

Permissions assignment for roles:
- PA(Viewer) → {"view article"}
- PA(User) → {"view article", "create article", "edit own article"}
- PA(Editor) → {"view article", "create article", "edit all articles"}
- PA(Administrator) → {"user management", "access control", "system maintenance"}

Requests assignment for permissions:
- RqA("view article") → {/articles/list, /articles/view}
- RqA("create article") → {/manage/articles/create}
- RqA("edit own article") → {/manage/articles/edit}
- RqA("edit all article") → {/manage/articles/edit}
- RqA("user management") → {/manage/users}
- RqA("access control") → {/manage/permissions}
- RqA("system maintenance") → {/manage/system}

Request parameters assignment for permissions:
- RqPA("view article") → {articleID}
- RqPA("create article") → {articleTitle, articleContent}
- RqPA("edit own article") → {articleID, articleTitle, articleContent}
- RqPA("edit all article") → {articleID, articleTitle, articleContent}
- RqPA("user management") → {userID}
- RqPA("access control") → {userID, permissionID}
- RqPA("system maintenance") → {action}

As shown in the example above, the new model offer flexibility and simplicity for access control restriction using request path and parameters.

Conclusion

The paper describes extended request-oriented role access control model, which takes into account web applications features. New elements were defined: token, request, request parameter, inclusion relation. The model created allows flexible access control for modern web applications.

Authors created guidelines to apply model for real-world web applications. Developing web applications according to these guidelines allows reducing security risks. The new model can be used to enhance security in real-world web applications.

Reference

[1] The 2016 Internet Security Threat Report, Symantec Corp, 2016. Available at: https://www.symantec.com/security-center/threat-report (Accessed September 9, 2016).

[2] Bell D. E., LaPadula L. J. Secure Computer Systems: Unified Exposition and Multics Interpretation. – Bedford, Mass.: MITRE Corp., 1976. – MTR–2997 Rev. 1.

[3] M. Bishop. "Introduction to Computer Security", Published by Addison-Wesley, 2005, pp. 784.
[4] R. Sandhu. Role-based Access Control // Advanced computers, vol. 46, 1998, pp. 237–286.

[5] R. Bhatti, E. Bertino, and A. Ghafoor. A Trust-based Context-Aware Access Control Model for Web Services // Distributed and Parallel Databases Archive, vol. 18, no. 1, July 2005, pp. 83–105.

[6] Harrison M., Ruzzo W. Monotonic protection systems/In DeMillo R., Dobkin D., Jones A., Lipton R., editors//Foundation of Secure Computation. – New York: Academic Press, 1978, pp. 337–365.

[7] Sandhu R. The typed access matrix model // Proceeding of the IEEE Symposium on Research in Security and Privacy. – Oakland, CA, May 1992, pp. 122–136.

[8] R. J. Lipton, L. Snyder. A Linear Time Algorithm for Deciding Subject Security // Journal of the ACM, Published by Addison-Wesley, vol. 24. no. 3, 1977, pp. 455–464.

[9] Fr. Li, Ht. Wu. Design and Implementation of Authorization System Based on RBAC // 7th International conference on intelligent human-machine systems and cybernetics IHMSC 2015, Vol I, 2015, pp. 502-504.

[10] D. Yan, J. Huang, et al. Policy Conflict Detection in Composite Web Services with RBAC // IEEE 21st international conference on web services (ICWS 2014), 2014, pp. 534-541.

[11] S. Kedjar, A. Tari, P. Bertok. Access Control and Information Flow Control for Web Services Security // International journal of information technology and web engineering, vol. 11/1, 2016, pp. 44-76.

[12] M. H. Alalfi, J. R. Cordy, T. R. Dean. Automated Verification of Role-based Access Control Security Models Recovered from Dynamic Web Applications // 14th IEEE International Symposium on Web Systems Evolution (WSE), 2012, pp. 1-10.

[13] J.B.D. Joshi, E. Bertino, U. Latif, A. Ghafoor. A generalized temporal role-based access control model // IEEE Transactions on Knowledge and Data Engineering, vol. 17, iss. 1, 2005, pp. 4-23

[14] D. Kononov, S. Isaev. Development of extended path-based role access control model for web applications // Proceedings of International Conference Mathematical and Information Technologies (MIT 2016), Vrnjacka Banja, Serbia, 2016, pp. 166-173.