Development of Extended Path-based Role Access Control Model for Web Applications

Dmitry Kononov and Sergey Isaev

Institute of Computational Modeling of SB RAS
Akademgorodok 50/44, Krasnoyarsk, Russia
{ddk,si}@icm.krasn.ru
http://icm.krasn.ru

Abstract. Web applications security is a complex problem with several aspects. One aspect is access control according to specified security policy. Access control is accomplished by security model restrictions. This research is dedicated to developing security access control model for web applications. This work describes path-based RBAC model, which improves RBAC and allows flexible access control using request path (URI). Authors created guidelines to apply model's elements for real-world web applications. Developing web applications with model described allows reducing security risks.

Keywords: security models, access control, web applications.

1 Introduction

Today modern Web applications and services are affected by several security issues. Computer security is becoming increasingly important and actual. According to Symantec security research [1], in spite of security means development there are high security risks for web applications. Web applications security is a complex problem with several aspects. One aspect is access control according to specified security policy. Access control is accomplished by security model restrictions. Choosing and applying appropriate security model is able to reduce risks of successful attacks.

Widely known security models include discretionary, mandatory, and role-based [2, 3]. In our work, we research and develop security model built 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 path-based RBAC model, which improves RBAC and allows flexible access control using request path (URI).

2 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 analyzes conditions under which the computer system cannot initiate information flows from the objects with a high level of confidentiality to objects with a lower level of confidentiality.

Role-based access control is a further development of the discretionary access control policy: permissions to system objects are grouped according to certain characteristics, forming role. Roles are intended to manage access control rules in a more simple way. These models do not take into account the specifics of web applications, in particular, the hierarchical organization of requests and links. The paper describes the adapted role-based security model that eliminates these problems.

3 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: $U \to 2^R$ – function assigning for each user a variety of roles to which he can be authorized;
PA: $R \to 2^P$ – function assigning for each role set of access permissions, while $\forall p \in P, \exists r \in R$ such that $p \in PA(r)$;
$user$: $S \to U$ – function defining for each user session, on whose behalf it is authorized;
$roles$: $S \to 2^R$ – 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).

4 Adapting the model for web applications

To existing $RBAC_1$ model elements "user", "role", "permission", "session" we added new elements taking into account web application features: "token", "request".

Definition 1. *Token (Tk)* – 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>.

Definition 2. *Request (Rq)* – 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).

A request belongs to the session, one session can handle multiple requests. Request and permission 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 a unique resource identifier (URI) of request A contains the path of the unique identifier of the resource request B with the initial position in the line within the same namespace, with len $(\mathrm{B}_{\mathrm{path}}) \leq$ len $(\mathrm{A}_{\mathrm{path}})$, where len(x) – length of the string x.

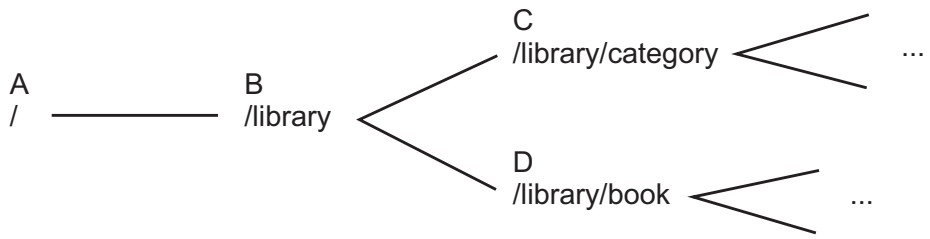

Fig. 1. An example of the inclusion relations on top of requests

Figure 1 shows the requests A, B, C, and D, which satisfy the following: A $\leq$ B, B $\leq$ C, B $\leq$ D, A $\leq$ C, A $\leq$ D.

Inclusion relation has the following properties:

1. reflexivity:
$$\forall rq \in Rq : rq \leq rq,$$
2. antisymmetry:
$$\forall rq, rq' \in Rq : ((rq \leq rq')\&(rq' \leq rq)) \rightarrow rq = rq',$$
3. 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: $P \to 2^{Rq}$.

Definition 4. *Requests hierarchy (RqH)* – 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.

Figure 2 shows a diagram of adapted security model elements. The model name is path-based role-based access control security model.

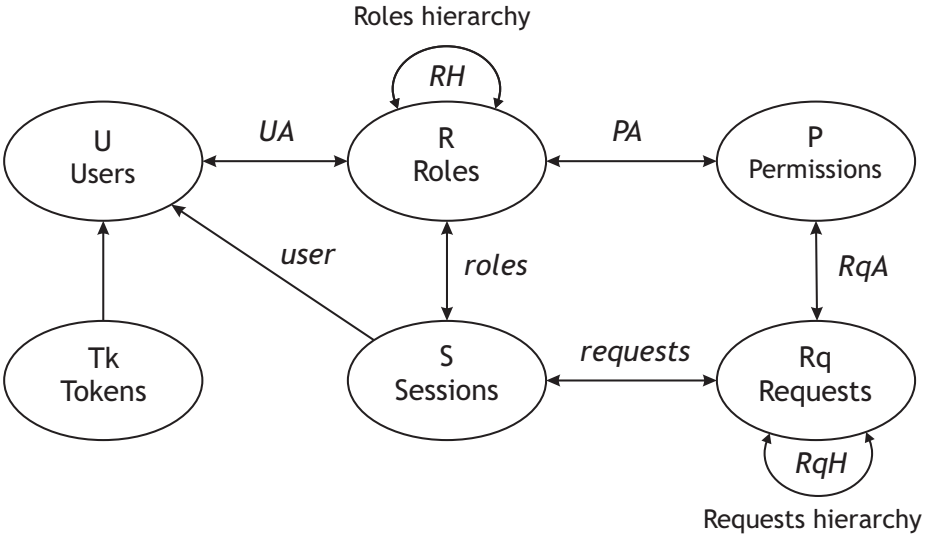

Fig. 2. Elements of path-based RBAC model

5 Adapting mandatory access model

In [9] for role-based security model authors describe use of mandatory access control designed to protect against threats to information confidentiality. Within defined terminology, we describe mandatory role-based access model to web applications. In addition to defined above elements, the following elements were added:

Rq – set of requests;

$(L, \leq)$ – confidentiality levels lattice;

$c : U \to L$ – function of user access levels;

$c : Rq \to L$ – function of confidentiality levels for requests;

$A = \{\text{read}, \text{write}\}$ – access types;
$R = \{x\_read | x \in L\} \cup \{x\_write | x \in L\}$ – set of roles;
$P = \{(rq, read) | rq \in Rq\} \cup \{(rq, write) | rq \in Rq\}$ – set of permissions.

Using the definitions 5.20 and 5.22 [10], according to requirements of liberal mandatory access control for set of requests Rq we define a hierarchy on top of roles R and restriction functions $UA()$, $roles()$, and $PA()$.

As a part of the mandatory access control, information flow is defined.

Definition 5. We assume that there is an *information flow* from request $rq \in Rq$ to request $rq' \in Rq$ if and only if there are roles $r, r' \in R$, and session $s \in S$, such that $(rq, read) \in PA(r)$, $(rq', write) \in PA(r')$, and $r, r' \in roles(s)$.

Let's formulate a proposition about the impossibility of forbidden information flows from the request with a higher confidentiality level to requests with a lower confidentiality level.

Proposition 1. If a role-based access model complies with liberal mandatory access control requirements, then for any requests $rq, rq' \in Rq$, such that $c(rq) > c(rq')$, it is impossible to initiate information flow from rq to rq'.

The proof is similar to theorem 5.1 [10].

Thus, the model described is safe in terms of information flows for requests with different confidentiality levels.

6 Application of models

Security models described can be used in a wide range of applications. To apply these security models 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 Rq. This set includes all client requests and API calls provided by the system. Using Rq 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 that a user can perform. 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 Rq using function $RqA()$. One permission can have multiple requests. One request can be assigned to many permissions.

To maintain users a set of users U is created, each of which have assigned roles from R 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 Tk. 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 path-based 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 Rq: {
/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}

Roles assignment for users:

UA(Anonymous) $\rightarrow$ {Viewer}

UA(Alice) $\rightarrow$ {User}

UA(Bob) $\rightarrow$ {User}

UA(John) $\rightarrow$ {Editor}

UA(Martin) $\rightarrow$ {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}

As you can see from the example above, the developed models offer flexibility and simplicity for access control restriction that can be used in real-world web applications.

7 Conclusion

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

Also extended path-based mandatory role-based access control model was created with additional sets: requests, security levels lattice, access types, roles, and permissions. Impossibility of forbidden information flows from higher to lower security levels was proven.

Authors created guidelines to apply model's elements for real-world web applications. Developing web applications according to these guidelines allows reducing security risks. We use the model developed to enhance security in our web applications [11].

References

1. The 2016 Internet Security Threat Report, Symantec Corp, 2016. Available at: `https://www.symantec.com/security-center/threat-report` (Accessed April 13, 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. Bishop M. "Introduction to Computer Security", Published by Addison-Wesley, 2005, pp. 27–35.
4. Sandhu R., Coyne E. J., Feinstein H. L. and Youman C. E. Role-based Access Control Models // IEEE Computer (IEEE Press), vol. 29, no. 2, 1996.

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. Sandhu R. Role-based Access Control // Advanced computers, vol. 46, 1998, pp. 237–286.
10. Devyanin P. N. Security models for computer systems. – M.: Publishing center "Academy", 2005. 144 p.
11. Kononov D.D., Isaev S.V. The security model of cross-platform web services for municipal procurement support // Prikladnaya diskretnaya matematika [Applied Discrete Mathematics], 2011, no. 4, pp 48–50.