Roles-based Access Control Modeling and Testing for Web Applications

Bo Song[1], Shengbo Chen[2]
[1] College of Information Science and Technology, Qingdao University of Science & Technology, 266061 Qingdao, China
[2] School of Computer Engineering and Science, Shanghai University, 200072 Shanghai, China
Email: {songbo, schen}@shu.edu.cn

***Abstract*—Web applications are widely used in people's everyday life. They have permeated financial sectors, banking sectors, e-business and online shopping. Usually, different users have different permissions on these applications. Additionally, role-based access control (RBAC) mechanisms have been wildly integrated into web applications. The security and correctness of web applications are the most fundamental, crucial aspects to the success of business and organizations. In existing research work on modeling of RBAC, the user's roles and permissions are fixed and static, and do not consider that with the evolution and running of the system, the roles and permissions are dynamic. To the best of our knowledge, research work on role-based access control modeling and testing for web application has been seldom done. In this paper, taking the dynamic feature of roles and permissions into account, we propose an approach to modeling and testing web applications with role-based access control. We give out an algorithm to capture and compute the dynamicity of roles and permissions in running time. The FSM is employed to model the behavior of web applications, and then the augmented FSM (AFSM) is plied as a tool to model role-based access control. Finally, using the construction algorithm, the tests are generated automatically which satisfy the corresponding test coverage criteria.**

Keywords-Software testing; access control;RBAC; FSM; web applications; tests generation

I. INTRODUCTION

Nowadays, web applications are very prevalent around the world, while their correctness, security and reliability are often crucial to the success of business and organizations. Web applications are widely used in the financial fields, banking sectors, e-commerce, e-government, online shopping, BBS, etc. Generally speaking, different users have different roles and grant permissions or access authorization. For example, in the same web applications, the administrator has the permissions to manage the users and resource, while the regular user, e.g., Bronze user only has the permissions to view the post, create a new post, view resource; Silver user has a *Download* resource permission more than that of Bronze user. Gold user has the same permissions as Bronze user, but his/her experience points increase more quickly than the other users. When the experience points reach to a certain level, his/her role and permissions will change dynamically in the running time of the system. The structures and behaviors of web applications are complicated. At present, there are no systematic methods and tools that are employed to test Web applications efficiently.

Today's web applications integrate role-based access control (RBAC) [1] to manage the users' roles and permissions. These aggravate the complex of web applications. In existing work on modeling role-based access control, the user's role and permissions are fixed and static which is not suitable to use to modeling the real web applications. How to model and test web applications with role-based access control to ensure their security and correctness presents a challenge in software engineering community.

In this paper, we take the dynamic feature of the users' roles and permissions into account, and propose an approach to modeling and testing web applications with role-based access control. The FSM is used as a formal tool to model the behavior of web application. An algorithm is presented to capture and compute the roles and permissions of the users dynamically in the running time of the system. An augmented FSM (AFSM) is employed to model role-based access control. Based on AFSM models, using the construction algorithm, finally, the tests generation is carried out automatically.

As for the basic knowledge about role-based access control, you can refer to the paper [1].

II. A MOTIVATING EXAMPLE

In order to address our role-based web application modeling and testing, in this section, we give out a small web application *Post View and Resource Download* (PVRS) as an example to illustrate our approach, as shown in Fig. 1. Under normal conditions, a user must log in a web application to view, reply and create a new post, and to download some resource which the user needs. First and foremost, the user needs to log in the system. So he/she types the web application's URL in *Address* field of web browser. After the *Enter* key is pressed, the *MainPage* will display in the web browser and show to the user. In the *MainPage*, there is a link *login*. When the user clicks the *login* link, the web page *Login* will display to the user. The user inputs his/her username and password, and clicks the *submit* button. Upon this clicking, the username and password are sent to the web server for authentication. If the username and password are correct, and the server identifies that the user is a common web user, the user will enter the private page *UserView*. On the other hand, if the username and password are correct, and the server identifies that the

978-0-7695-4863-0/12 $26.00 © 2012 IEEE
DOI 10.1109/WCSE.2012.19

user is a web administrator, the user will enter the private and administrating page *AdmView*.

In the *UserView* page, there are three links: *viewPost*, *newPost* and *viewResource* which used to view some posts, create a new post and view resource, respectively. After the user clicks the *viewPost* link, the *PostView* page will show to the user. When the user wants to reply this post, so he clicks the link *replyPost* in the *PostView* page, then the web page *PostReply* will appear, the user write down his/her content and press *submit* button to finish replying. And when the user wants to create a new post, he/she clicks the *newPost* link in the *UserView* page, then the PostNew page will display, so he/she input what he want to create and with the clicking of the *submit* button, the user finishes creating a new post. When the user wants to view and download some resource, he/she can click the *viewResource* link from the *UserView* page, then the *ResourceView* page will display. The user clicks a resource link which he/she wants to download which depends on his/her the number of Experience Points (E). If his/her experience point is bigger than the permitted number *M*, the next page will be *DownloadView* page, otherwise, will be *PointsLimited* page which does not permit the user to download the corresponding resource.

In the *AdmView* page, there are two links: *managePost* and *manageUser* which is employed to manage the posts and manage the web users, respectively. When the administrator clicks the *managePost* link, the *PostMangement* page will appear. The administrator can delete some invalid posts and shield some unhealthy posts. When the administrator find an invalid post and he can choose the post and clicks the *delPost* links, the he will enter the *PostDelete* page, and with the clicking of *confirmDelP* button, the corresponding post will be permanently deleted. In the same manner, the administrator can shield the unhealthy posts. When the administrator wants to manage the users, he/she can click the *manageUser* link in the *AdmView* page. Then the *UserMangement* page will appear. So, taking the same approach, the administrator can freeze some inactive users or can delete some users.

In this paper, R = {*Administrator*, *Bronze User*, *Silver User*, *Gold User*} or R = {A, B, S, G} where A denotes *Administrator*, B means *Bronze* user, S means *Silver* user, and G is *Gold* user.

III. MODELING ROLE-BASED WEB APPLICATIONS

Finite state machines (FSM) provide a convenient way to model software behavior in a way that avoids issues associated with the implementation. Several methods for deriving tests from FSMs have also been proposed [2, 3, 4]. Theoretically, web applications can be completely modeled with FSMs, however, even simple Web pages can suffer from the state space explosion problem. Consequently, in this paper, we focus on the role-based access control behavior of web application. Without considering the details of web pages, we can use FSM to model our web applications. So, firstly, we give out the definition of FSM.

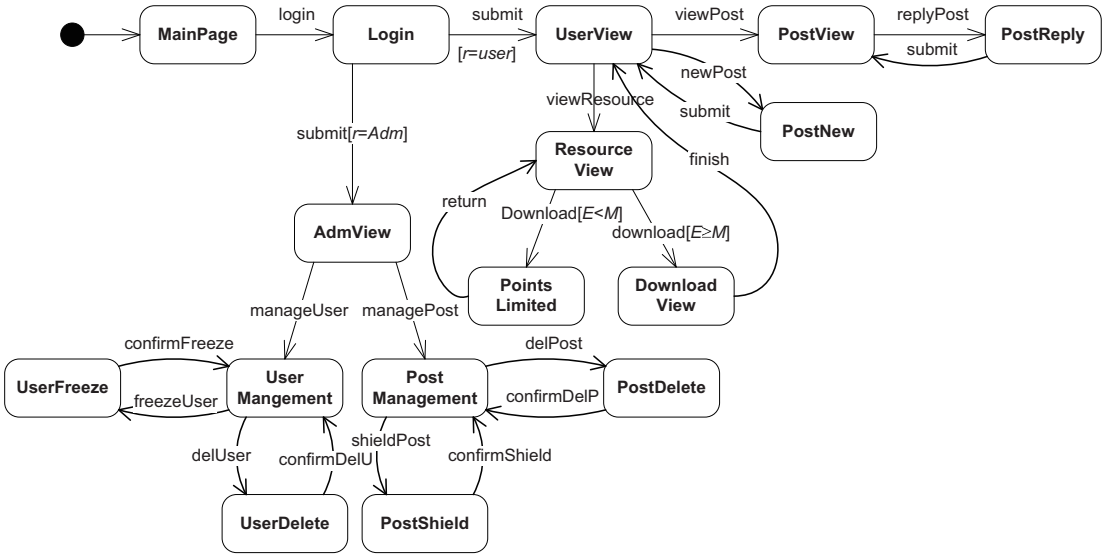

Figure 1. Example of web applications.

Definition 7. A Finite state machine (FSM) [5] M is a five-tuple of the form: $M = (S, S_0, Act, T, S_m)$, where:

- S is a finite, non-empty set of states to denote the set of pages in our example. Each page can be looked as a state.
- $S_0 \subseteq S$ is a non-empty set of initial states; in our example, the initial state is *MainPage*;
- Act is a nonempty finite set of actions; for example, clicking the links or buttons;
- $T \subseteq S \times Act \times S$ is a finite set of labeled transition, usually called transition function, and can also denote as $T : S \times Act \rightarrow S$. In our example, the user clicks the

buttons or links to trigger the state transitions;

– $S_m \subseteq S$ is the set of terminal states.

Consequently, according to the FSM definition above, we can employ the FSM to model the web application as shown in Fig.1. But, from the Fig. 1, we can not see some information about role-based access control. Next, in order to fully model the web application with role-based access control, the information of role-based access control should be taken into account. First of all, we give out the requirements on role-based access control for our example as follows.

Requirements of Role-Based Access Control (R-RBAC) for web application: in our motivating example, there exist two categories of web users: one is web *administrator*; the other is the regular web *users*. Different user's category can play different roles and have different permissions according to his/her number of experience points *E*. And usually, there have different grade role of the regular web users. For example, in our motivating example, there exist three grades depending on the corresponding experience points *E*: *Bronze User* ($0 \le E \le 50$), *Silver User* ($50 < E \le 100$), and *Gold User* ($100 < E$). Additionally, different grade users have different permissions and have diverse and various growth of experience points. If number of experience point of the regular web user among $0 \le E \le 50$, that to say, the user is a *Bronze User*, the corresponding permissions include: *Reply a post* and *New a post*. And at this grade, it does not permit the user to download some resource. When he finishes replying a post, his *E* will incremented by 1 unit. And when he creates a new post, his *E* will increase by 2 units. Besides, when one of his posts is replied by someone, his *E* will add by 1 unit. If his *E* is beyond 50 and less than 100 ($50 < E \le 100$), he will be a *Silver User*, at this time, the corresponding permissions include: *Reply a post*, *New a post* and *Download* resources. When he finishes replying a post, his *E* will add by 2 units, creates a new post, his *E* will increase 3 units. And also when one of his posts is replied by someone, his *E* will add by 2 units, while downloading a resource will decrease by 2 units. When the number of his experience points is larger than 100 ($E > 100$), he will become a *Gold User*. At the moment, with replying a post, he will get by 2 units; creating a post gets by 4 units, being replied will increase by 2 units, while downloading a resource will cut down by 2 units.

From the above description of requirements of role-based access control for web application, we can see that the role and permission is dynamic. For example, suppose that the number of a user's experience points is 49 ($E = 49$), he is just a *Bronze User*, and can not have the permission to download a resource. But when he finishes creating a post ($E = E + 2$), his *E* will be $49 + 2 = 51$ which is among the range ($50 < E \le 100$). At this moment, he becomes a Silver User (*role*) and his permissions also change to add a *Download* functionality. When he finishes downloading a resource, $E = E - 2 = 51 - 2 = 49$, he will become a *Bronze User* again and without a *Download* permission. Additionally, during the running of the web application, someone replies one user's post, and corresponding *E* will change. Forasmuch, by this way, we can see that the user's role and permissions are changed dynamically. Fig.2 shows how to compute the user's Experience Points *E* and the corresponding roles. And the corresponding algorithm is shown in Algorithm 1.

$$(E) = \begin{cases} Administrator, \quad E = NULL; \\ Users \begin{cases} Bronze\ (0 \le E \le 50) \begin{cases} Reply\ a\ post, \to E = E + 1; \\ New\ a\ post, \quad \to E = E + 2; \\ Be\ replied, \quad \to E = E + 1; \end{cases} \\ Silver\ (50 < E \le 100) \begin{cases} Reply\ a\ post, \to E = E + 2; \\ New\ a\ post, \quad \to E = E + 3; \\ Be\ replied, \quad \to E = E + 2; \\ Download, \quad \to E = E - 2; \end{cases} \\ Gold\ (100 < E) \begin{cases} Reply\ a\ post, \to E = E + 2; \\ New\ a\ post, \quad \to E = E + 4; \\ Be\ replied, \quad \to E = E + 2; \\ Download, \quad \to E = E - 2; \end{cases} \end{cases} \end{cases}$$

Figure 2. Computations of Experience Points (E) and Roles.

As mentioned above, the number of the user's experience points *E* and the user's role is dynamically changed, so the user's permission assignment is also dynamically regulated. It is hard to directly use the traditional FSM to model and test the web application with role-based access control. FSM provides a convenient way to generate test cases. Therefore, in order to facilitate testing of web application with role-based access control, in this paper, we give an augmented FSM (AFSM) to model web applications and based on AFSM, we give out the tests generating algorithm to generate the tests.

Definition 8. A web application with role-based access control is described as an augmented FSM, *AFSM* = (*S*, S_0, *Act*, *Tr*) where

- ✧ *S* is a finite set of states. Each state consists of a page, the user's role and his/her permissions on this page, used to model each state. That is to say, *state* = (*page*, *role*, *permission*). All the states are listed in Table 1, where NIL denote empty; P0-*MainPage*; P1-*Login*; P2-*UserView*; P3-*PostView*; P4-*PostReply*; P5-*PostNew*; P6-*ResourceView*; P7-*PointsLimited*; P8-*Download*; P9-*AdmView*; P10-*UserManagement*; P11-*PostManagement*; P12-*PostDelete*; P13-*PostShield*; P14-*UserDelete*; P15-*UserFreeze*; ViP-*ViewPost*; ViR-*ViewResource*; Re-*ReplyPost*; Ne-*NewPost*; Be-*BeReplied*; Do-*Download*;
- ✧ $S_0 \subseteq S$ is the set of initial states; In our example PVRS, the initial state is (P0, NIL, NIL, login);
- ✧ *Act* is a finite set of actions; for example, clicking the links or the buttons on web page.
- ✧ $Tr \subseteq S \times Act \times S$ is a finite set of labeled transitions, for every state $s \in S$ there is a state $s' \in S$ such that $Tr(s, s')$;

Algorithm 1 User's Role Assignment

```
1: R ← {Bronze, Silver, Gold}
2: for all r ∈ R do
3:   for 0 ≤ E ≤ 50 do
4:     r ← Bronze;
5:     for ReplyPost() = TRUE do
6:       E = E + 1;
7:     end for
8:     for NewPost() = TRUE do
9:       E = E + 2;
10:    end for
11:    for BeReplied() = TRUE do
12:      E = E + 1;
13:    end for
14:  end for
15:  for 50 < E ≤ 100 do
16:    r ← Silver;
17:    for ReplyPost() = TRUE do
18:      E = E + 2;
19:    end for
20:    for NewPost() = TRUE do
21:      E = E + 3;
22:    end for
23:    for BeReplied() = TRUE do
24:      E = E + 2;
25:    end for
26:    for Download() = TRUE do
27:      E = E − 2;
28:    end for
29:  end for
30:  for 100 < E do
31:    r ← Gold;
32:    for ReplyPost() = TRUE do
33:      E = E + 2;
34:    end for
35:    for NewPost() = TRUE do
36:      E = E + 4;
37:    end for
38:    for BeReplied() = TRUE do
39:      E = E + 2;
40:    end for
41:    for Download() = TRUE do
42:      E = E − 2;
43:    end for
44:  end for
45: end for
```

As for the pages which *Administrator* accessed, for example, *AdmView*, *UserFreeze*, *UserManagement*, *PostManagement*, *PostDelete*, *PostShield* and *UserDelete* are only related to the *Administrator* role, and have no role and permission change. When modeling and testing web applications which logged as an *Administrator*, we can separate *Administrator* from regular users and model and test each part. Fig.3 gives out the AFSM model of web application with role-based access control which the *Administrator* logs on the system. Fig.4 gives out the AFSM of web application and also takes role-based access control into account.

Table 1. The state of each page with RBAC.

Page	State(s)	No.
P0	(P0, NIL, NIL, login);	S0
P1	(P1, Adm, NIL, submit);	S1
	(P1, User, NIL, submit);	S2
P2	(P2, B, (ViP, ViR, Re, Ne, Be), (ViP, ViR, Re, Ne, Be));	S3
	(P2, S, (ViP, ViR, Re, Ne, Be, Do), (ViP, ViR, Re, Ne, Be));	S4
	(P2, G, (ViP, ViR, Re, Ne, Be, Do), (ViP, ViR, Re, Ne, Be));	S5
P3	(P3, B, (ViP, ViR, Re, Ne, Be), Re);	S6
	(P3, S, (ViP, ViR, Re, Ne, Be, Do), Re);	S7
	(P3, G, (ViP, ViR, Re, Ne, Be, Do), Re);	S8
P4	(P4, B, (ViP, ViR, Re, Ne, Be), submit);	S9
	(P4, S, (ViP, ViR, Re, Ne, Be, Do), submit);	S10
	(P4, G, (ViP, ViR, Re, Ne, Be, Do), submit);	S11
P5	(P5, B, (ViP, ViR, Re, Ne, Be), submit);	S12
	(P5, S, (ViP, ViR, Re, Ne, Be, Do), submit);	S13
	(P5, G, (ViP, ViR, Re, Ne, Be, Do), submit);	S14
P6	(P6, B, (ViP, ViR, Re, Ne, Be), NIL);	S15
	(P6, S, (ViP, ViR, Re, Ne, Be, Do), Do);	S16
	(P6, G, (ViP, ViR, Re, Ne, Be, Do), Do);	S17
P7	(P7, B, (ViP, ViR, Re, Ne, Be), return);	S18
P8	(P8, S, (ViP, ViR, Re, Ne, Be, Do), NIL);	S19
	(P8, G, (ViP, ViR, Re, Ne, Be, Do), NIL);	S20
P9	(P9, Adm, (MaU, MaP, FrU, DeU, ShP, DeP), (manageUser, managePost));	S21
P10	(P10, Adm, (MaU, MaP, FrU, DeU, ShP, DeP), (delUser, freezeUser));	S22
P11	(P11, Adm, (MaU, MaP, FrU, DeU, ShP, DeP), (delPost, shieldPost));	S23
P12	(P12, Adm, (MaU, MaP, FrU, DeU, ShP, DeP), confirmDelP);	S24
P13	(P13, Adm, (MaU, MaP, FrU, DeU, ShP, DeP), confirmShield);	S25
P14	(P14, Adm, (MaU, MaP, FrU, DeU, ShP, DeP), confirmDelU);	S26
P15	(P15, Adm, (MaU, MaP, FrU, DeU, ShP, DeP), confirmFreeze);	S27

As for the regular users, things become complicated. From Fig.2 and the Algorithm 1, we can see that the user's roles and permissions are dynamically changed. For example, suppose the number of experience points is E = 49, 49 ∈ [0, 50]. So the current logged user is a *Bronze User* without the *download* permission. When he create a new post, E = E + 2 = 49 + 2 = 51, 51∈ (50, 100], he becomes a *Silver User* with add a *download* permission. And the value of E can be changed only when the user replies a post, new a post, or download a resource. So, the value of *E* can be added when replying a post, creating a new post, while the value will be reduced when downloading a resource (as shown in blue lines in Fig.4). These situations can complex the behaviors of web application with role-based access control. Fig.4 shows the AFSM of web application logged as regular users, and the changes of roles and permissions are also considered.

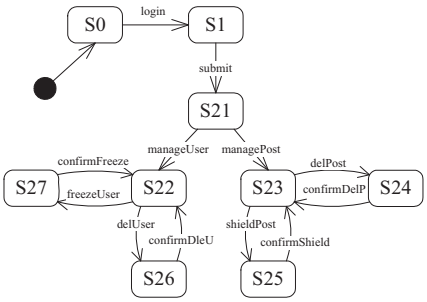

Figure 3. AFEM for logging as Administrator.

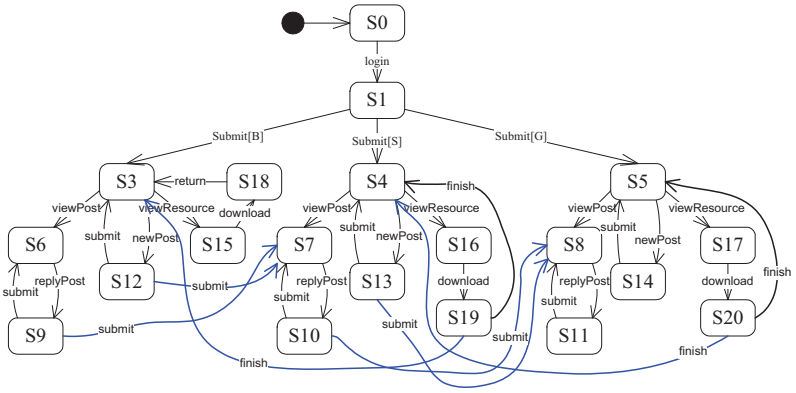

Figure 4. AFSM for different roles of regular users.

As mentioned above, the behavior model of web application with role-based access control can be formally illustrated by the AFSM. In order to test role-based access control of web applications, next, this paper will provide a method to generate tests which satisfies the state coverage and transition coverage.

IV. GENERATING TESTS

The structure of a Web application is often complicated and difficult to maintain. Especially, role-based access control is used to aggravate this dilemma. As the AFSM (see Fig.3 and 4) has many more cycles, it is hard to use to generated tests without redundancy effectively and directly. So, based on AFSM, using the construction algorithm [13], a FSM Test-Tree is constructed for each AFSM model. Fig. 5 and 6 show the corresponding FSM test-trees for the AFSM models in Fig.3 and 4, respectively. From the test-tree, we just get 4+15=19 test sequences which satisfying states coverage and transitions coverage.

Each sequence from the root to the leaf corresponds to a test sequence. So, according to the Fig.5, we can get 4 test sequences as below:

1. S0- «login» - S1 - «submit» - S21 - «manageUser» - S22 - «freezeUser» - S27 - «confirmFreeze» - S22;
2. S0 - «login» - S1 - «submit» - S21 - «manageUser» -S22 - «delUser» -S26-«confirmDelU»-S22;
3. S0 - «login» - S1 - «submit» - S21 - «managePost» - S23 - «shieldPost»-S25-«confirmShield»-S23;
4. S0 - «login» - S1 - «submit» - S21 - «managePost» - S23 - «delPost»-S24-«confirmDelP»-S23;

According to the Fig.6, we can get 4 test sequences as below:

1. S0-«login»-S1-«submit[B]»-S3-«viewPost»-S6-«replyPost»-S9-«submit»-S6;
2. S0-«login»-S1-«submit[B]»-S3-«viewPost»-S6-«replyPost»-S9-«submit»-S7;
3. S0-«login»-S1-«submit[B]»-S3-«newPost»-S12-«submit»-S3;
4. S0-«login»-S1-«submit[B]»-S3-«newPost»-S12-«submit»-S7;
5. S0-«login»-S1-«submit[B]»-S3-«viewResource»-S15-«download»-S18-«return»-S3;
6. S0-«login»-S1-«submit[S]»-S4-«viewPost»-S7-«replyPost»-S10-«submit»-S7;
7. S0-«login»-S1-«submit[S]»-S4-«viewPost»-S7-«replyPost»-S10-«submit»-S8;
8. S0-«login»-S1-«submit[S]»-S4-«newPost»-S13-«submit»-S4;
9. S0-«login»-S1-«submit[S]»-S4-«newPost»-S13-«submit»-S8;
10. S0-«login»-S1-«submit[S]»-S4-«viewResource»-S16-«download»-S19-«finish»-S3;
11. S0-«login»-S1-«submit[S]»-S4-«viewResource»-S16-«download»-S19-«finish»-S4;
12. S0-«login»-S1-«submit[G]»-S5-«viewPost»-S8-«replyPost»-S11-«submit»-S8;
13. S0-«login»-S1-«submit[G]»-S5-«newPost»-S14-«submit»-S5;
14. S0-«login»-S1-«submit[G]»-S5-«viewResource»-S17-«download»-S20-«finish»-S4;
15. S0-«login»-S1-«submit[G]»-S5-«viewResource»-S17-«download»-S20-«finish»-S5;

V. RELATED WORK

A lot of work has been done for role-based access control, each of which has different origins and provides different resolutions for access controls. The papers [6, 7] focus on role-based access control modeling in database management systems. The article [8] gives out a framework of four reference models to let the developers better understand RBAC. The authors' framework separates the administration of RBAC from its access control functions. Yan et al. [9] consider the large and complex multi-user applications, and proposes an object-oriented model in UML supporting application-level access control based on user's roles.

Raman and Sharma [10] use active rules to enforce role-based access control standard, analyze different components of active rules and their mapping mechanisms.

In Internet and web applications, Kurt [11] proposes an approach that uses role-based access controls (RBACs) and web session management to protect against network security breaches. Kun Wang [12] gives out an access control judgment algorithm based on ANN to quickly get the user's permissions in wireless networks.

As mentioned above, many researchers focus their attention on role-based access control. To our knowledge, research work on role-based access control modeling and testing for web application has been seldom done.

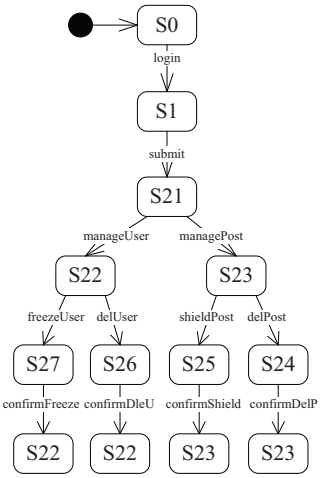

Figure 5. FSM-TT derived from Fig.3.

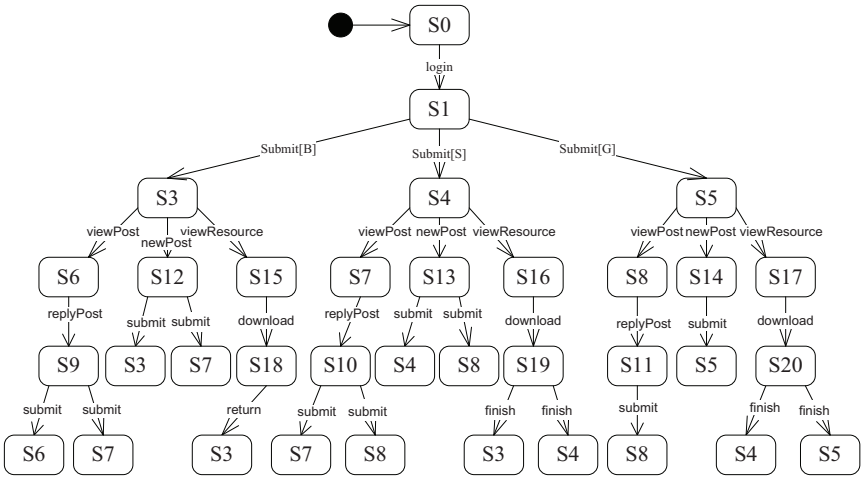

Figure 6. FFSM-TT derived from Fig.4.

VI. CONCLUSIONS

Web applications are widely used in people's everyday life. They have permeated financial sectors, banking sectors, e-business and online shopping. Usually, different users have different permissions on these applications. Additionally, role-based access control mechanisms have been integrated into web applications. The security and correctness of web applications are the most fundamental, the most crucial aspects to success their process. In this paper, taking the dynamic feature of roles and permissions into account, we propose an approach to modeling and testing web applications with role-based access control. We give out an algorithm to capture and compute the dynamicity of roles and permissions in running time. In this paper, we construct FSM test-tree derived from the AFSM models of web applications with role-based access control, which used to generate test sequences to satisfy the corresponding coverage and the test sequences are effective and not redundant.

ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation of Shandong Province (No.ZR2011FL010), the National Natural Science Foundation of China (NSFC) (60970007, 61073050, 61170044), Shanghai Leading Academic Discipline Project (J50103), Key Laboratory of Science and Technology Commission of Shanghai Municipality (09DZ2272600).

REFERENCES

[1] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman, "Role-Based Access Control Models," IEEE Computer, vol. 29, no.2, February 1996, pp. 38-47.

[2] T. Chow, "Testing software designs modeled by finite-state machines," IEEE Transactions on Software Engineering, SE-4(3): May 1978, pp.178-187.

[3] S. Fujiwara, G. Bochmann, F. Khendek, M. Amalou, and A. Ghedasmi, "Test selection based on finite state models," IEEE Transactions on Software Engineering, 17(6), June 1991, pp.591-603.

[4] Jeff Offutt, Shaoying Liu, Aynur Abdurazik, and Paul Ammann, "Generating test data from state-based specifications," The Journal of Software Testing, Verification, and Reliability, 13(1), March 2003, pp. 25-53.

[5] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman and Monica S. Lam. Compilers: Principles, Techniques, and Tools (second edition). Addison-Wesley, Inc. 2006.

[6] Milivoje Petrovic, Michael Grossniklaus, and Moira C. Norrie, "Role-based modelling of interactions in database applications," In Proceedings of the 18th international conference on Advanced Information Systems Engineering (CAiSE'06), 2006, pp. 63-77.

[7] Yingjun Zhang, Yang Zhang, and Kai Chen, "A map-layer-based access control model," In Proceedings of the 12th international conference on Information Security Applications (WISA'11), Springer-Verlag, Berlin, Heidelberg, 2011, pp. 157-170.

[8] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman, "Role-Based Access Control Models," IEEE Computer, vol.29, no.2, February 1996, pp. 38-47.

[9] Yan Han, Liu Fengyu, and Zhang Hong, "An object-oriented model of access control based on role," SIGSOFT Softw. Eng. Notes, vol. 25, no.2, March 2000, pp. 64-68.

[10] Raman Adaikkalavan and Sharma Chakravarthy, "Access control using active rules," In Proceedings of the 27th British national conference on Data Security and Security Data (BNCOD'10), Springer-Verlag, 2010, pp. 12-24.

[11] Kurt Gutzmann, "Access Control and Session Management in the HTTP Environment," IEEE Internet Computing, vol. 5, no.1 January 2001, pp. 26-35.

[12] Kun Wang, Zhenguo Ding, and Lihua Zhou, "Efficient Access Control in Wireless Network," In Proceedings of the IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology (WI-IATW'06), 2006, pp. 85-88.

[13] Bo Song and Huaikou Miao, "Modeling Web Applications and Generating Tests: A Combination and Interactions-guided Approach," In Proceedings of the Third IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE '09). IEEE Computer Society, 2009, pp. 174-181.