SecTag: A Multi-Policy Supported Secure Web Tag Framework

Ruixuan Li, Meng Dong, Bin Liu, Jianfeng Lu, Xiaopu Ma, Kai Li

School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, Hubei 430074, China
rxli@hust.edu.cn, {mengdong, bliu9, lujianfeng, xpma}@smail.hust.edu.cn, likai@hust.edu.cn

ABSTRACT

Traditional web application development often encounters tight coupling problem between access control logic and business logic. It is hard to configure and modify access control policies after a system has been deployed. In this demonstration, we present SecTag, a multi-policy supported secure web tag framework, to address this problem. We define a series of general-purpose secure attributes that meet the demand of fine-grained access control in web presentation layer. We also design a set of high interactive secure tags, which encapsulate secure features to provide reusable secure components for web development. A running example of SecTag is presented to demonstrate the effectiveness of the proposed framework.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*; D.2.9 [**Software Engineering**]: Management—*Software configuration management*

General Terms

Security, Management

1. INTRODUCTION

Web-based enterprise-level applications have gained a tremendous growth in the last decade. While business groups and individuals enjoy the rich data and information services on the web through all kinds of web applications, fine-grained access control requirements for protecting the sensitive contents and important resources within web pages have become more and more urgent [1]. Traditional access control module development in web systems often suffers problems, such as lack of fine-grained and multi-policy support, tight coupling of the access control logic with the business logic. It is hard to reconfigure or modify the access control policies after the system has been deployed. Hence, it is ultimately important to provide a development framework considering the features of multi-policy authorization, component reusability, multi-views and high interaction support.

In this demonstration, we present a multi-policy supported secure web tag framework, SecTag, to create secure and fine-grained web resource protection. The framework is designed upon the core secure tag library that implements tag-based fine-grained access control model in applications. SecTag provides the features of easy configuration and multi-policy support for secure tag components. They can be configured through the visual interface of secure tag management service in web presentation layer. Our main contributions are as follows.

(1) We develop a reusable framework to help developers create secure fine-grained and multi-policy support web resource protection.

(2) We define a series of general-purpose secure attributes that meets the demand of fine-grained access control in web presentation-layer.

(3) We create a set of high interactive secure tags, which encapsulate secure features to provide reusable secure components for web development.

The source code and program guide for SecTag can be downloaded at SecTag@Google.code[1].

2. DESIGN OF SECTAG

2.1 Framework of SecTag

Our goal is to design and implement a light-weight basic framework of SecTag, which can be easily satisfied with the traditional model-view-controller (MVC) mode. By using SecTag, fine-grained access control policies can be easily configured through the visual interface without any modification of codes. Most of the general access control logics can be encapsulated in the form of secure tags. Thus, the development and maintenance workload are greatly reduced. Figure 1 shows the architecture of SecTag framework.

The core of SecTag includes Business/Model Dispatcher (BMD), Secure View Proxy (SVP) and Secure Tag Library (STL). BMD implements the request/response scheduling logic of web framework. The ultimate response is generated through SVP. Secure tags deployed in the web pages will be bound with secure data according to the attributes of users, which makes the same web pages present different result views according to the security configuration.

As shown in Figure 1, when an initial request reaches the Servlet container, it will be passed to a standard filter chain. The filter chain includes the preprocessing filter that is used to initialize the secure context. When a user is not authenticated, a login page will be returned if the

[1]http://code.google.com/p/sectag/

CCS'10, October 4–8, 2010, Chicago, Illinois, USA.
ACM 978-1-4503-0244-9/10/10.

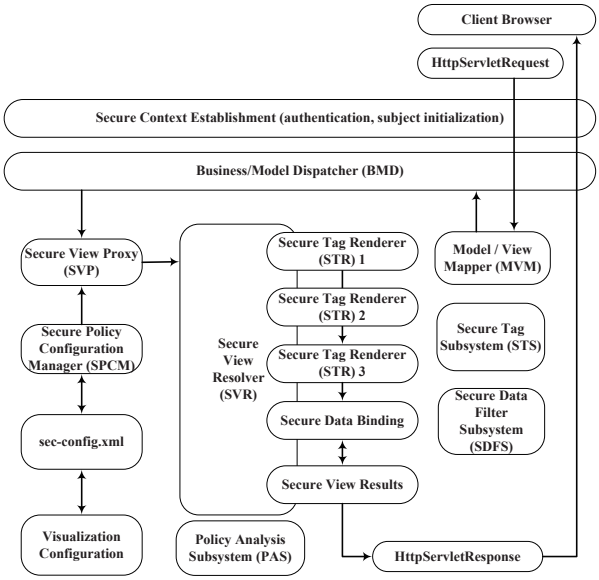

Figure 1: Framework of SecTag.

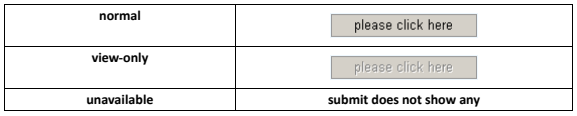

Figure 2: Three rendering states of *submit* tag.

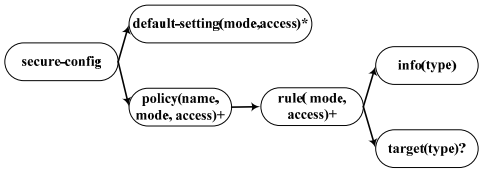

Figure 3: Structure of policy configuration.

Table 1: Common tags with secure requirements

Tags of UI display	<a>; <submit>; <button>; <textfield>; <textarea>
Tags of data access control	<select>; <radio>; <checkbox>

user's information can not be obtained from cookies or sessions. After the user provides legal identity information, the preprocessing filter will initialize user's secure context according to the secure policy configuration, including the attributes of authorized subject and secure decision point (SDP) for each policy. Then, BMD will be called, which issues polling requests to Model/View Mapper (MVM) to determine whether the Secure View Resolver (SVR) should be called by this request.

If an SVR is to be called by MVM, BMD will assign its control permissions to SVP and transmit the initial data model and view to SVP. SVP will get the secure policy information from Secure Policy Configuration Manager (SPCM) that reads secure configuration information from sec-config.xml. Then, an SVR is created by SVP for secure view rendering. The SVR traverses the document tree of the entire view and calls Secure Tag Renderer (STR) for each secure tag protected by fine-grained access control policy. STR will call Secure Tag Subsystem (STS) to determine the final presentation and render the local secure view according to authorization attributes of the secure context and secure policies defined in SPCM. Before binding the data model, SVP will dynamically filter out the data that the user wants to shield according to the secure configuration. Finally, the view encapsulating secure data with all rendered secure tags will be returned to the client browser.

2.2 Design of Policy Configuration

Document Object Model (DOM) has been widely used in XML-oriented fine-grained access control model [2]. SecTag adopts XML to describe the policy rules, called sec-config.xml, in which all of these fine-grained permissions are configured intensively. Based on actual secure applications, tag rendering that responds to user's requests can be divided into three states: normal, view-only and unavailable, taking *submit* tag as an example shown in Figure 2.

Meanwhile, tags can be divided into two specific categories, as shown in Table 1, according to the objects they control.

(1) Tags of user interface (UI) display: mainly includes the visibility and availability of control, such as whether the button to display or not, and whether the text is editable or the editing operation is locked on the specific user.

(2) Tags of data access control: mainly includes the dynamic rendering of the data list. For example, the cascading-select tag will filter out the data that the user is unable to access, and only show those data the user has permissions. Each data item in the data list can also be divided into three states: normal, view-only and unavailable.

The structure of policy configuration is shown in Figure 3. In the figure, "default-setting" specifies the default authorization settings. Attribute "mode" represents the default access control model supporting multi-policies, such as RBAC (role-based access control), MAC (mandatory access control) and DAC (discretionary access control). Attribute "access" represents the default display status, such as normal, view-only and unavailable. Attribute "name" of policy is policy's unique identification with a unique value. Each policy can set attribute "mode" and "access", and contains at least one specific rule. "rule" is used to describe the specific access control information, and its attribute "access" specifies the access permission to the rule. The sub-node "info" sets the user collection who owns the permissions specified by "access". The sub-node "target" indicates the display status corresponding to the data items of the data list that the tag receives. We use "n/v/u" to denote the different values of "target": normal/view-only/unavailable.

Figure 4 gives an example of policy configuration for policy p. As shown in Figure 4, we can inject a structured query language (SQL) statement or a method of a class in sec-config.xml when "type" is defined as "SQL" or "Method". In these cases, Policy Analysis Subsystem (PAS) will automatically load the dynamic data to complete the policy analysis.

2.3 Design of Secure Tags

We extend traditional web tags to support secure attributes through binding secure policies that are configured in sec-config.xml. We use FreeMarker template engine to dynamically render tags, which makes different users see different views according to their requests and permissions. The architecture of Secure Tag Subsystem (STS) is shown in Figure

```
<sec-config>
  <default-setting access="view-only" mode="RBAC"/>
  <policy name="p">
    <rule access="view-only" mode="MAC">
      <info>L3</info>
    </rule>
    <rule access="unavailable" mode="DAC">
      <info>John,Mark</info>
      <target>n,n,n,n,v,v,v,v</target>
    </rule>
    <rule access="normal" mode="RBAC">
      <info type="SQL">SELECT user FROM UserRole
        WHERE role="SuperAdmin" OR role="Admin"</info>
      <target type="Method">
        cn.edu.hust.idc.sectag.demo.getTargetList()</target>
    </rule>
  </policy>
</sec-config>
```

Figure 4: An example of policy *p* configured in sec-config.xml.

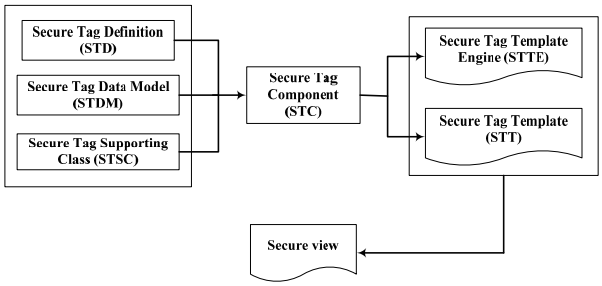

Figure 5: Architecture of Secure Tag Subsystem.

5. These secure tags are based on the development guidelines of JSP 2.0. Secure tags in SecTag includes five main components.

(1) Secure Tag Definition (STD). The definition of secure tags is described in sectag.tld, which includes the names of all secure tags, supporting classes, and the description of attributes. Attributes are divided into two types: traditional HTML tag attributes and the extended secure attribute, named "policy", which is used to bind an access control policy in sec-config.xml.

(2) Secure Tag Component (STC). The JSP page with secure tag references will call corresponding tag resolver, namely Secure Tag Supporting Class (STSC), to analyze the tags according to the definitions in sectag.tld. The function of STSC is to get values of the attributes recorded in tags and initialize STC object simultaneously. STCO will call Secure View Renderer (SVR) and commission Secure Decision Point (SDP) to make an authorization decision feedback for current access request according to the relevant secure attributes configured in sec-config.xml. STC is bound to Secure Tag Data Model (STDM) according to user's permissions after obtaining the authorization decision-making, and then chooses the right Secure Tag Template Engine (STTE) that meets current authorization. Finally, STC assigns the task of rendering secure result view in HTML to Secure Tag Template (STT) and STTE.

(3) Secure Tag Template Engine (STTE). We use FreeMarker template engine as STTE to create different result views according to the access requests and permissions.

(4) Secure Tag Template (STT). The advantage of STT used in SecTag is the separation of application and view which can draw different HTML views. SecTag components will dynamically call different templates to show different result views of the same tag according to access permissions and data models.

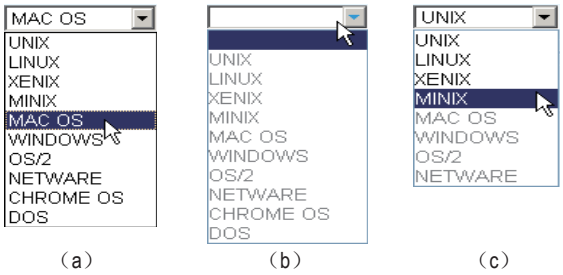

Figure 6: Rendering of secure tag *select*.

(5) Secure Tag Data Model (STDM). STDM is defined to describe the tag data with access control. Before rendering into HTML, the original data list will be filtered by Secure Data Filter Subsystem (SDFS) according to the policy it binds. Hence, a secure data view is generated. The features of SDFS make the data model no longer rely on the filtering code that may be coded repeatedly. Therefore, the cost of security code development and maintenance can be reduced.

3. AN EXAMPLE OF SECTAG

Taking the tag *select* that receives data as an example, we bind attribute "policy" of *select* to policy *p* that is configured in sec-config.xml shown in Figure 4, and describe the user identity in the form of (username, role, level). Suppose there are 6 user profiles: (Tom, SuperAdmin, L6), (Mary, Admin, L5), (Lucy, User, L4), (Lily, User, L3), (John, Casualuser, L2), (Mark, Casualuser, L1). The rendering result views are shown in Figure 6 (a), (b) and (c) with the login of Tom, Lucy and John respectively. All the data items could be selected for Tom, but view-only for Lucy. John can select items 1∼4 normally, view-only for items 5∼8 and unavailable for the rest items.

4. CONCLUSIONS

In this demonstration, we develop a reusable secure development framework SecTag to solve the problem of tight coupling between access control and business logic. It assists developers to achieve fine-grained and multi-policy authorization management for web resource protection.

Acknowledgements

This work is supported by National Natural Science Foundation of China under Grant 60873225, 60773191 and 70771043.

5. REFERENCES

[1] G. Hsieh, K. Foster, G. Emamali, G. Patrick, L. Marvel: Using XACML for Embedded and Fine-Grained Access Control Policy. International Conference on Availability, Reliability and Security, 462-468, 2009.

[2] E. Damiani, S. Paraboschi, S. Vimercati, P. Samarati: A Fine-Grained Access Control System for XML Documents. ACM Trans. on Information and System Security, 5(2): 169-202, 2002.