An XML-based Solution to Web Applications Security

Teng Lv$^{1,2}$ and Ping Yan$^1$

$^1$ College of Mathematics and System Science, Xinjiang University, Urumqi 830046, China
lt0410@163.com
$^2$ Teaching and Research Section of Computer, Artillery Academy, Hefei 230031, China
yanp2002cn@163.com

Abstract. This paper analyzes the most common security problems of web application level. Two XML-based languages ACPDL (Access Control Policy Description Language) and SPDL (Security Policy Description Language) are proposed to specify access control policies and security policies for Web application level security, respectively. Based on ACPDL and SPDL, a system framework WALSG (Web Application Level Security Gateway) is presented to provide web application level security, which can be used as a secure tool to define access control policies and security policies with the development of web site.

1 Introduction

Web applications are popular at today’s web sites. Traditional methods, such as firewalls, are not capable of providing security defense for web application level any longer. The number of security incidents is increasing exponentially and the percentage of web-based attacks now represents approximately 70 percent of all attack incidents \cite{1}. Many application level attacks require no particular hacker techniques at all \cite{2}. Traditional methods of protecting application level security are mainly focused on how to write secure code of a particular program \cite{3}. However, it is very difficult, even impossible, to require all code of all web applications to be written in secure way considering the cost of program development, the code mistakes, and the security knowledge of programmers.

In order to deal with these challenges, we propose a new XML-based solution called WALSG (Web Application Level Security Gateway) to web application level security in this paper. We define Access Control Policy Description Language (ACPD$L$) and Security Policy Description Language (SPDL) by XML Schema \cite{4} for security requirements of web applications. Thus, we can use XML \cite{5} files to describe specific access control policies and security policies for URLs and cookies. The corresponding access control policy files and security policy files are stored in Access Control Library (ACLib) and Security Policy Library (SPLib), respectively. Furthermore, a Request Processor (RP) is designed to parse a request from a client and carry out the corresponding access control policies and security policies for the request. WALSG can dynamically provide not only the security protection on web application level but also a useful mechanism to maintain web application security with the devel-
An XML-based Solution to Web Applications Security

Related work. An application level security system, AppShield [6], developed by Sanctum Inc. provides web application level security dynamically. But it does not provide administrators with mechanism to specify security settings according to different HTML pages. This will inevitably restrict the usability and flexibility in practical environment. Our solution WALSG provides administrators with means to describe specific security settings for specific HTML page, which makes WALSG more flexible and applicable in practical situations. A security policy description language is presented in Ref.[7], which is described by XML DTD [8], Standard ML [9] and a pre-defined library, but it has poor consistency and requires different programs to understand the syntax of a single security policy file. Our solution WALSG employs uniform XML Schema to specify security settings for URLs to reduce the inconsistency of syntax of security policy file which can be fully understood by any XML parser. We also propose a solution to web application level security in Ref. [10], but with little considerations of access control policies for different users accessing different URLs. In this paper, WALSG provides detailed access control policies, which is a significant improvement over our previous solution. A related access control policy called RBAC (Roll-Base Access Control) is proposed in Ref.[11], whose access decisions are based on the roles that individual users have as part of an organization. But our access control policies are based on object-subject-action-condition model which are specified by administrators. Ponder proposed in Ref. [12] is an object-oriented policy language for distributed systems management, which has the similar functionality as WALSG, but we use XML as description language for security policy and access control policy, which are more flexible and maintainable for web sites.

Organization. The rest of this paper is organized as follows: Section 2 gives a brief analysis of web application level security attack. Section 3 describes the structure, components, and technical aspects of our solution WALSG. And finally, Section 4 concludes the paper and points out the future directions of this work.

2 Web Application Level Attack Analysis

We list some common techniques frequently used by a hacker or someone else to attack web applications.

1. Parameter modification method. There are often many problems when an invalid or a prohibitive parameter of a URL request is transferred to web applications. If these problems are not handled properly in advance, some credential information or other data which are not intended to show to users may be exposed to users directly. This attack includes: to modify the input fields of a HTTP form; to provide special designed invalid input for a SQL sentence; etc.

2. Cookie modification method. More and more web applications rely on cookies to identify, establish, and maintain valid connection to unique user. The unauthorized users can easily establish a connection with the server by modifying the contents of the authorized user’s cookie.

©GESTS-Oct.2005
3. **Directory traversal method.** This includes: to traverse directories of a web site to explore the logical structure of a web site; to truncate a given directory path to find some useful information for further attack; to search hidden web paths to find the access to restricted areas of web applications; etc.

3 An XML-Based Solution to Web Application Level Security

From web application level security attack techniques mentioned in Section 2, we can see that they have a common characteristic, i.e., they modify the parameters of web forms, or cookies of a web site, or access directories of a web site. Based on the common characteristic, we propose a new XML-based solution WALSG to web application level security shown in Figure 1. The basic idea of WALSG is to check and verify the parameters of web forms, cookies of a web site, and access directories of a web site according to the security policies and access control policies specified by administrators. WALSG is resided between firewalls and web servers and automatically protects the whole web applications on the web servers. As a URL request is transferred from clients to servers, WALSG automatically allows or denies the request according to the corresponding security policies and access control policies defined for each HTML page.

![Fig. 1. Overview of WALSG in a Web application system](image)

3.1 Structure of WALSG

Figure 2 shows the overall structure of WALSG. We can see that a URL request generated from clients is transferred not to the web servers directly, but to Request Processor (RP) of WALSG. Then RP analyzes the request according to the access control policies and security policies which are stored in Access Control Library (ACLib) and Security Policy Library (SPL), respectively. If the request conforms both to the access control policies and the security policies, it will be transferred to the web servers. Otherwise, a security error message will be returned to the client. And if necessary,
the message may contain some explanations for the error, which give the client useful
divices for correct form of URL requests.

Furthermore, for security administrators of web servers, WALSG also provides a
useful mechanism to maintain the security policies and access control policies with the
development of web sites. If a URL is added into or deleted from the web site, the
URL Engine will be triggered to give administrators the mechanism to define the
access control policies and the security policies for the changed URL.

3.2 Access Control Policy

Access control policies define which URLs are allowed to be accessed by specific
clients. We give the Access Control Policy Description Language (ACPDL) based on
XML Access Control Language (XACL)\textsuperscript{[13]}.

As ACPDL is based on XACL, it can specify object-subject-action-condition ori-
teered access control policies in web application level. We give a brief description of
ACPDL:
1. A subject is the user or client, which is specified by \textit{uid} (user ID), role-set and
group-set, where role-set and group-set are a set of role names and group names to
which the user belongs, respectively.
2. An object is a URL to be controlled by access control policies.
3. An action is a kind of operation that can be operated on an object by a subject. The
permission attribute is used to indicate whether an access is granted or denied. The
name attribute is used to specify the action name ("GET", "POST" or "GETand-
POST"). Provisional actions are associated with an action, which define some ad-
ditional actions should be executed before or after the action. A provisional action
can have zero or more input parameters. A parameter element can have attributes
name and value which indicate the name and value of the parameter, respectively.
4. A condition is a Boolean formula such that the access is granted if the formula holds true. The sub-element predicate represents a Boolean function that returns true or false and may have one or more parameter elements as child elements.

5. Element ACLpolicy specifies the access control policies, which consists of sub-elements xacl and property. Element xacl defines access control policies for specified objects. Element acl defines the specified actions for the specified subjects if the specified condition is satisfied. Element property defines the propagation policies, which consists of sub-elements ACLpolicy_definition and action_definition. Element ACLpolicy_definition defines propagation policies in which the propagation (of objects, roles, and groups), conflict resolution, and default policies for the action are specified, which are indicated by elements propagation_along_oh, propagation_along_rh, propagation_along_gh, conflict_resolution, and default. Element action_definition specifies the propagation policies for a given action. The attribute policy of element action_definition is used to refer to an ACLpolicy_definition element.

Access Control Library (ACLib) stores and manages the access control policy files and associated information for URLs controlled by access control policies. For each access control policy file, the file name, URL specified by the file, creator, created date, the last modified date, size, and so on, are stored in ACLib. This will facilitate the retrieval and management of a specified file by different attributes.

**Example 1.** The following XML file specifies an access control policy for user Tom (who belongs to group UserGroup and whose role is User) to access URL http://www.demo.org.cn. When Tom accesses http://www.demo.org.cn, the system checks the uid of Tom, and records it after the access. The propagation action of uid Tom, role User, and group UserGroup is override and upward, and conflict resolution for them is dtp (i.e., to denial take precedence).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ACLpolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="E:\spdl.xsd">
<xacl id="ID000005" precedence="0">
<object URL="http://www.demo.org.cn"/>
<acl id="ID000001" precedence="102">
<subject>
<uid>Tom</uid>
<role>User</role>
<group>UserGroup</group>
</subject>
<action permission="grant" name="Get">
<provisional_action timing="after">
<parameter name="uid" value="Tom"/>
</provisional_action>
</action>
<condition operation="and">
<predicate name="Check">
<parameter name="uid" value="Tom"/>
</predicate>
</condition>
</acl>
</xacl>
</ACLpolicy>
```
3.3 Security Policy

Security policies define the security requirements for URLs and cookies. We give the Security Policy Description Language (SPDL) based on Ref.[10].

Some explanations of SPDL:

1. A policy element may consist of a list of sub-elements URL and cookie. Each URL element consists of a list of parameter sub-elements which has attributes name, MAC and method, and sub-elements otherConstraints and transformationRules. Each cookie element has attributes name and MAC, and sub-elements otherConstraints and transformationRules.

2. The element URL defines a URL address requested by clients. The attribute prefix defines a part of URL such as http://www.demo. The name attribute defines the name of a parameter or a cookie. The MAC attribute indicates whether a parameter or a cookie must be encoded by MAC [14] generated on the server side. The method attribute indicates a URL address is requested from a client (method="GET"), submitted to the server (method="POST"), or both (method="GETandPOST").

3. It is important that we use elements otherConstraints and transformationRules to specify more complicated constraints for a parameter or cookie element.

Security Policy Library (SPLib) stores and manages the security policy files for URLs or cookies and associated information. For each security policy file, the file name, URL or cookie name specified by the file, creator, created date, the last modified date, size, and so on, are stored in SPLib. This will facilitate the retrieval and management of a specified file by different attributes.

Example 2. The following XML file specifies a security policy for URLs whose prefix is http://www.demo such that parameter ProdNum must be encoded by MAC (indicated by attribute MAC) and conforms to XML Schema productnumber.xsd.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<policy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="C:\policy.xsd" xsi:type="PolicyType">
  <URL prefix="http://www.demo">
    <property>
      <action_definition ACLpolicy="ID000006">
        <ACLpolicy_definition id="ID000006">
          <propagation_along_oh permission="grant" name="override" direction="upward"/>
          <propagation_along_rh permission="grant" name="override" direction="upward"/>
          <propagation_along_gh permission="grant" name="override" direction="upward"/>
          <conflict_resolution name="dtp"/>
          <default permission="grant"/>
        </ACLpolicy_definition>
      </property>
    </property>
  </URL>
</policy>
```
XML Schema file productnumber.xsd constraints the product number ProdNum to be the form of either 3 characters followed by one of letter A to Z or just 7 digits only as shown in following:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xsd:simpleType name="ProdNumType">
        <xsd:annotation>
            <xsd:documentation>a Product number</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\d{3}[A-Z]|\d{7}"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

3.4 Request Processor

Figure 3 shows the workflow of Request Processor (RP) of WALSG. The brief description of this workflow is:
1. A URL request from a client is transferred to RP.
2. RP verifies that the requested URL is stored in Access Control Library (ACLib) and confirms to the access control policies of the URL. If passed, then go to step 3; otherwise, go to step 5.
3. RP collects the information of requested URL and verifies the constraints according to the security policy stored in Security Policy Library (SPLib). If passed, then go to step 4; otherwise, go to step 5.
4. The request is transferred to web servers and a response is returned to the client. Go to step 6.
5. The requested URL is not allowed for the client and RP shows an error message to the client.
6. The process of request and response terminates.
3.5 URL Engine

URL Engine (UE) provides a useful mechanism to maintain access control policies and security policies for security administrators with the development of web sites. UE is started by administrators as needed or automatically triggered by any new created URL or deleted URL in web sites. For each new created URL, UE requires the administrator to specify the new corresponding access control policies and security policies. For each deleted URL, UE requires the administrator to delete the corresponding access control policies and security policies. All these changes of access control policies and security policies must be immediately updated in SPLib and ACLib, respectively. This mechanism of incremental management of UE provides more flexibility and security for dynamically developing web sites.

3.6 Discussion of Efficiency of WALSG

For the problem of efficiency of WALSG, as each secured URL page is checked and verified by WALSG, the overall throughput of a web server will decrease with the complexity of security policies and access control policies of web sites. As most security policies and access control policies are not complex and specified by administrators according to practical situations, it is possible and practical to obtain an average high throughput of a web site.
3.7 Discussion of Languages ACPDL and SPDL

In this sub-section, we give some brief explanations for ACPDL and SPDL proposed in Sub-section 3.2 and 3.3.

1. It is easy and enough to express object-subject-action-condition oriented access control policies in web application level by ACPDL.
2. It is convenient to constraint the types and patterns for Web forms, parameters, URLs, and cookies by SPDL.
3. ACPDL and SPDL are extensible for new types of policy that may arise in the future and this can be achieved by defining new element types or including other XML Schema files in specific XML files.
4. ACPDL and SPDL are easy to use, understand, maintain, and implement by different applications as they are all based on XML, which is a de facto standard for data exchange over World Wide Web.

4 Conclusions and Future Work

This paper presented an XML-based solution WALSG to web application level security. It is useful for the maintenance of an existing web site and the development of a new web site in the aspect of application level security. Although our solution can not resolve all application level security problems, it indeed can effectively resolve most practical and common web application level security problems such as those mentioned in Section 2 at the present time.

More and more attack techniques targeted on application level are evolved as the time elapses. We plan to add more functions (such as XML encryption, signature, etc.) into WALSG to face the evolving attack challenges.

Acknowledgements

This work is supported by National Natural Science Foundation (NSFC) of China under Grant No. 60563001 and Foundations for Young Teachers of Xinjiang University under Grant No. QN040101.

References


Biography

Name: Teng Lv
Address: College of Mathematics and System Science, Xinjiang University, Urumqi, 830046, P. R. China
Education & Work experience: He did postdoctoral research in Xinjiang University from October 2003 to October 2005. He got his Ph. D degree from Fudan University in June 2003.
Tel: 86-991-8582013
E-mail: lt0410@163.com

Name: Ping Yan
Address: College of Mathematics and System Science, Xinjiang University, Urumqi, 830046, P. R. China
Education & Work experience: She is a professor of Xinjiang University now. She did postdoctoral research in Xinjiang University from October 2002 to October 2004. She got her Ph. D degree from Fudan University in July 2002.
Tel: 86-991-8582013
E-mail: yanp2002cn@163.com

©GESTS-Oct.2005