

# Study on the application layer security in E-Commerce websites

Zhenlong Li<sup>1</sup>, Wenping Guo<sup>1</sup>, Xiaoming Zhao<sup>1,2</sup>

<sup>1</sup>Department of Computer Science, Taizhou University, Linhai, China  
Email: {li\_zhenlong@163.com, guo\_wp@126.com }

<sup>2</sup>Information Engineering Institute, Zhejiang Industrial University, Hangzhou, China  
Email: tzxyzxm@yahoo.com.cn

**Abstract**—Most of studies about E-Commerce Security focus on the data confidentiality issue. Although security mechanisms, such as SSL (secure socket layer) or SET (secure electronic transaction), have been adopted in websites, catastrophic events that confidential data in E-Commerce are revealed happened more than once. The essential reason for this is that there exist potential security vulnerabilities in the E-Commerce applications themselves. The origins of these vulnerabilities are mainly from the lack of reliable input validation that can prevent E-commerce application from attacks. SQL Injection, Cross-site Scripting (XSS) and Price Changing Attack are mainly known security threats to E-Commerce applications. These attacks and the protecting ways by using XML validation technology have been discussed, and a framework that prevents E-Commerce applications from attacks has been proposed in this paper.

**Index Terms**—E-Commerce Security; SQL Injection; XML Schema; Input validation

## I. INTRODUCTION

Because security is the basic precondition of successful E-Commerce, most of studies about E-Commerce Security focus on the data confidentiality issue. In order to prevent important information for users such as credit card number and transaction information from stealing and modifying on the internet, the security mechanisms like SSL or SET have been adopted in present E-Commerce websites [1]. In spite of this, catastrophic events that confidential data in E-Commerce was revealed happened more than once. The essential reason is that there exist potential security vulnerabilities in E-Commerce applications themselves and not that Encryption mechanism security was insufficient. The origins of these vulnerabilities are mainly from the lack of the reliable input validation that can prevent E-commerce application from attacks [2]. The attacker can get the chance to steal or modify the transaction information using input page. SQL Injection is a typical example of attacks on the vulnerabilities. Cross-site Scripting and Price Changing Attack are other known security threats to E-Commerce applications.

Although the security vulnerabilities mentioned above have been discovered, the studies that can offer the effective protecting methods are less. Because the design

logic of application, the purpose of transport information among pages, as well as the parameter type and the length are different, the validation programs that can be used to prevent the E-commerce application from attacks for specific webpage should be different. It is a challenge to develop a unified method to validate web inputs for all web applications. In this paper, the application layer security threats and protecting ways based on XML validation technology will be discussed and a framework that prevents E-Commerce applications from attacks will be proposed.

## II. THE APPLICATION LAYER SECURITY THREATS AND COUNTERMEASURE

There are mainly three kinds of methods which can be used to attack the applications in the website, i.e. SQL Injection, Cross-site Scripting (XSS) and Price Changing Attack, respectively

### A SQL Injection

SQL Injection is an attack method using the application vulnerability. Because the present encryption protection only can guarantee the security of data transmitting on the internet, but can not check the data content any input data in the page form which was filled by the user, can be send to the web server. If the attacker has filled the data that include the vicious SQL query instruction in the web page form, these query instruction together with HTML file will drill through the firewall and arrive to web server. Once they are run at web server, the important information will be revealed or modified.

In order to keep away SQL Injection, the input string in the form must be filtered. The filter type and reasons were shown as in Tab. 1.

TABLE I. SQL INJECTION FILTER TYPE AND REASONS

Filter type	Reason
'	For include vicious code, end up the string
Select, create, update, insert, drop, union,...	SQL instruction
;	For add vicious SQL, end up SQL ahead
--	even if SQL grammar error, attack go on

the National Natural Science Foundation of China and Foundation of Scientific and Technological Study of Education Ministry, China

### B Cross-site Scripting (XSS)

Different from SQL Injection, Cross-site Scripting attack take place at the client. When the user browse and download the pages designed by the attacker, the script that embedded in these pages will be run [3]. The user's information was thieved or destroyed. For example, if attacker has embedded the following script in his page, the user's information that stored in the cookie will be transport to the website (www.hacker.com) designed by the attacker. Once having got the user information and authority, the attacker can carry out further attacks.

```
<Script>
Window.location=http://www.hacker.com/steal.cgi?ck+=document.cookie;
</ Script >
```

The attacker usually embeds the script in the public website using its form input function. Cross-site Scripting, using IE vulnerability, JavaScript and ActiveX, is so dangerous that all browsers that have not been protected will be attacked.

The attacker usually embeds the script in the public website using its form input function. Cross-site Scripting, using IE vulnerability, JavaScript and ActiveX, is so dangerous that all browsers that have not been protected will be attacked.

In order to avoid the Cross-site Scripting attack, the application in the website must filter the words and symbol that may appear in script and replace these symbol for corresponding ASCII code. The filter type and reasons are shown as Tab. 2.

TABLE II. CROSS-SITE SCRIPTING (XSS) FILTER TYPE AND REASONS

Filter type	Reason
< >	Begin tag End tag
“ “ & ? space tab	Add up parameter
/	Redirection vicious page URL
%	Code mark
Non-ASCII symbol	Not in ISO symbol set

### C Price Changing Attack

Price Changing Attack is an easy thing to be done. Firstly, attacker stores the page, in which the form containing the price column and is being used before check-out, to his own computer. Secondly, attacker edits the price column that is not able to be changed in the original page using the common editor software. Finally, attacker open the page that has been edited, and run the check-out program. The inevitable result is that the attacker can buy the goods with the cost that is lower than its marked price.

Because the attacker receive data from the website firstly, and then modify them, finally return them to website, Price Changing Attack can be taken as an attack that breaks the data integrity.

For protecting the data integrity, we adopt the way that adds up the Message Authentication Code (MAC) to the information which needs to protect. HMAC (Hash Message Authentication Code) Algorithm was used to create the MAC. Because the HMAC Algorithm creates the MAC by using a one-way Hush function and using the protecting information and the secret key as parameters [4], the attacker who don't know the secret key can not create the suitable MAC. However, the attacker may modify the price column using the Replaying Attack. For preventing the Replaying Attack, we adopt the HMAC Algorithm using the protecting information, secret key and goods number as parameters.

### III. VALIDATION BASE ON THE XML SCHEMA

We take the XML Schema as Security Policy Description Language. For each page, we create a XML Schema file to describe the constraint (type, range and style) of the input data in the form. Once the page input has been translated to a XML file, we can validate the XML file using the corresponding XML Schema. As the result of this, the page input data has been validated.

#### A Translate input page to XML file

The data accepted by application may be from three kinds of input ways, such as Form, QueryString or Cookie. No matter any way, the input data appear in the form of "name=value", such as address: http://www.ebuy.com/list.cgi?music=pop&page=3, in which include music=pop and page=3. We can use this property to define the rules responsible for turning the page to XML file. According to the input way, the defined data are laid into the corresponding block of XML file (see Fig. 1). For being consistent with the XML grammar, the form of "name=value" has to be turned to the form of "<name>value</name>"[5]. For example, "music=pop" has become "<music>pop</music>".

```
<?xml version="1.0" encoding="big5"?>
<Http
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
Xsi:noNamespaceSchemaLocation="framework.xsd" >
  <Form>
    <price MAC='y'>0577</price>
    <amount>3</amount>
    <isMember>Y</isMember>
    <mac>3a53fe1d995a23</mac>
    <time>13eaf49b</time>
  </Form>
  <QueryString>
    <music>POP</music>
    <page>3</page>
  </QueryString>
  <Cookie>
    <sessionID>8988991225</sessionID>
  </Cookie>
</Http>
```

Figure 1. The post-turn XML file

#### B Generate XML Schema file

After the page has been translated to XML file, in order to define the constraint to each parameter in XML file, we can generate the XML Schema file for each XML

file. For example, each “name=value” generates a following XML Schema element inform:

```
<xs:element name="name" type="nameType"/>
```

It means that the type of the parameter name is the nameType and the corresponding constraint to the name can be defined by the nameType. It will be presented as follows how to generate the XML Schema file for different input constraint.

#### 1) Length constraint

If the value type of parameter “passwd” is a string whose length is an integer between 8 and 15, the corresponding XML Schema file can be shown as following:

```
<xs:simpleType name="passwdType">
  <xs:restriction base="xs:string">
    <xs:minLength value="8"/>
    <xs:maxLength value="15"/>
  </xs:restriction>
</xs:simpleType>
```

#### 2) Range constraint

If the value of parameter “age” is an integer between 0 and 100, the corresponding XML Schema file can be shown as following:

```
<xs:simpleType name="ageType">
  <xs:restriction base="xs:int">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="100"/>
  </xs:restriction>
</xs:simpleType>
```

#### 3) Enumeration constraint

If the value of parameter “ccard” is one of strings including “Visa”, “Master” and “JCB”, the corresponding XML Schema file can be shown as following:

```
<xs:simpleType name="ccardType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Visa"/>
    <xs:enumeration value="Master"/>
    <xs:enumeration value="JCB"/>
  </xs:restriction>
</xs:simpleType>
```

#### 4) Special style constraint

If parameter “ID” expresses the number of identification card, it can be limited by regular expression. The corresponding XML Schema file can be shown as following:

```
<xs:simpleType name="IDType">
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-Z]d{9}"/>
  </xs:restriction>
</xs:simpleType>
```

#### 5) Avoiding the SQL Injection and Cross-site Scripting

In order to arrest the symbols that maybe appears in SQL Injection and Cross-site Scripting, the following XML Schema file can be used.

```
<xs:simpleType name="nameType">
  <xs:restriction base="xs:string">
    <xs:pattern value="^[^";<|%]"/>
  </xs:restriction>
</xs:simpleType>
```

#### 6) Avoiding the price changing

For the parameter “price” that can not be modified at web client, the predefined MAC attribution has been used in the corresponding XML Schema file, which can be shown as following.

```
<xs:complexType name="priceType">
  <xs:simpleContent>
    <xs:extension base="xs:int">
      <xs:attribute name="MAC" type="macType" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

### IV. THE FRAMEWORK FOR APPLICATION LAYER SECURITY OF E-COMMERCE WEBSITE

Base on what have been mentioned above, we propose a framework for application layer security of E-commerce website, which has been shown as Fig. 2. In this framework, the protecting program, which is independent from application, has been located between the E-commerce server and the web applications. Using the filter and transport function offered by web server, the input page be sent to the page filter for validation, only if there is no the word that threatens the web application, the input information that come from the web page can be push to the web applications. There are four modules in this framework, such as Page Filter, XML Parser, MAC Processor and XML Schema Generator. They will be described in the following, respectively.

#### A Page Filter

Page Filter, which is the core module, will fulfill following tasks.

- (1) Extract all parameter and create a XML file.
- (2) Push XML file to the XML Parser for the input validation. If pass validation, then go to (3), else send the error information to client computer and end up.
- (3) Push the XML file to MAC Processor for the data integrity checking. If pass checking, then go to (4), else send the error information to client computer and end up.
- (4) Push the input data to the application.
- (5) Accept the Cookie and HTML that come from the application. Extract the URL and form data and send them to the MAC Processor, which generate the MAC for them.
- (6) Accept the MAC and add them to the Cookie and HTML. Then push them to client computer.

#### B XML Parser

XML Parser accepts the XML file that come from the Page Filter and validates it using the corresponding XML Schema file. Because all the software that supports the XML includes the XML Parse, this module do not needs to be developed by ourselves.

#### C MAC Processor

MAC Processor is responsible for the data integrity protecting. Once the MAC Processor receive the data from Page Filter, it will find out the parameter that possesses MAC attribution by consulting XML Schema

file firstly, then generate a MAC for this parameter using the HMAC algorithm; finally compare it with the MAC from the web client. The compared result will indicate whether the parameter has been modified at web client. Similarly, for the HTML information which is sent to the web client, MAC Processor finds out the information that possesses MAC attribution and generates a MAC and pushes the MAC to the web client together with the information.

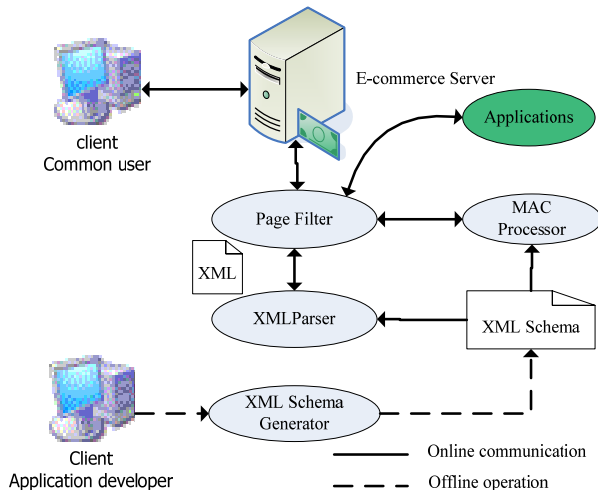


Figure 2. The framework for E-commerce application

#### D XML Schema Generator

XML Schema Generator is an offline auxiliary tool that helps the developer to generate the corresponding XML Schema file for each input web page. In our framework, it is an arduous task that prepare and maintains one XML Schema file for each input web page. Based on this, we develop a XML Schema Generator module in order to facilitate using of system. The XML Schema Generator can automatically generate a XML Schema file for one appointed input web page according to Developer's choice and setting in web page.

#### V. CONCLUSION

We use the standard XML schema as a security policy description language (SPDL). Developers can use XML schema to specify the constraints of web inputs. In the

proposed framework, which has been located between the web server and web applications, web inputs are first encapsulated in an XML document generated on the fly, the XML document is then validated by using XML schema. If no errors are found after the XML validation, the web inputs are valid for web applications. Hence, web applications can be protected effectively. Compared with previous approaches [6, 7], our framework uses the standardized XML schema as the SPDL for web applications. Therefore, no any particular compiler is required. In addition, because no any network configuration is needed in our framework, Existed applications of E-commerce can also be protected without any modifications. In summary, our framework provides a simpler and more effective mechanism for securing applications of E-commerce than the previous approaches.

#### ACKNOWLEDGMENT

The authors wish to thank Xiwen Li (Associate Professor) for providing technical assistance. This work was funded by the National Natural Science Foundation of China (No. 60473024) and by Foundation of Scientific and Technological Study of Education Ministry, China (No.031280).

#### REFERENCES

- [1] D.Scott., and R. Sharp, "Abstracting Application-Level Web Security," Proc. 11th Int'l World Wide Web Conf., ACM Press, New York, May 2002, pp:396-407
- [2] OWASP, "The Ten Most Critical Web Application Security Vulnerabilities," <http://unc.dl.sourceforge.net/sourceforge/owasp/OWASPWebApplicationSecurityTopTen-Version1.pdf>, January 13, 2003
- [3] iDEFENSE, "Evolution of Cross-Site Scripting Attacks," <http://www.odefense.com/XSS.html>, May 2002
- [4] H. Krawczyk, M.Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," Internet Request For Comments 2104, Feb. 1997
- [5] W3C, Extensible Markup Language, <http://www.w3.org/XML>, Nov.2006
- [6] Apache XML Project Home, <http://xml.apache.org>, Nov 2007
- [7] Apache Software Foundation, "Filters - Apache HTTP Server," [http://httpd.apache.org/docs-2.1 /filter.html](http://httpd.apache.org/docs-2.1/filter.html), Nov. 2004.