IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 3, May 2014.
www.ijiset.com
ISSN 2348 - 7968

XSS Preventer - Server-Side Prevention of Cross site Scripting in RESTful Web Services

Angel. N[1], Chandrasekar. A[2]

Abstract – Nowadays, Restful Web Services play an important role in providing the services to the browsers and to exchange the sensitive data. This paper presents the novel idea for preventing the XSS attacks from the server side in RESTful web service. Attacks work by injecting malicious code in to HTTP request for RESTful web service which is usually a java script. In the proposed technique, all input for RESTful web service will be transformed to unreadable format using the XSS preventer. The converted code will be stored in database.When the user invokes the web services, the actual malicious data which is in the form of unreadable format will be sent to the browser, but the malicious script will not get executed on the browser.

Keywords: *Cross Site Scripting, HTTP Request, Information Security, malicious code, Restful Web Service*

I. Introduction

Apart from the standards such as the Common Request Broker Architecture (CORBA), Remote Method Invocation (RMI) and Distributed Component Object Model (DCOM),Web Services [8] is a set of standards and a programming methods for sharing data between different software applications, moreover Web services is a standardized way to distribute services on the Internet.

More and more websites are opening their services to third-party developers as RESTful Application Programming Interfaces (RESTful APIs)[9]. These APIs are introduced to open up a channel where third-party applications can interact with those websites for data and resources. RESTful Web service interoperability goals are to provide automatic connections from one software application to another.

Cross site scripting[1] attacks exploit weak or lack of user input validation mechanisms in place in a web application, which serves the user-entered data as it is back to browsers. An attacker can exploit this weakness in the web application by injecting malicious script or browser executable content as part of the input. The injected code is served back to victim's browser by the unsuspecting web application as part of its generated response. When this malicious response is viewed by victim, it gets executed with the privileges of that web application in the victim's browser, causing the damage.

I.1. Problem Scope

XSS is rated as the 3rd most popular web application vulnerability by OWASP Top 10 2013 Project[2]. It was ranked at 2 in OWASP Top 10 2010 list, and 4 in CWE/SANS top 25 most dangerous software errors[3].

Types of XSS

There are three well known variations of XSS exploits that are known, namely: Local or DOM XSS (Type 0), Non-Persistent or Reflected XSS (Type 1), Persistent XSS
(Type 2)[10].

Reflected XSS

Reflected XSS vulnerability is exploited by attackers by crafting a URL to a vulnerable web application with malicious script via URL query or form parameters.

Persistent XSS

In persistent XSS, a malicious script injected by an attacker as part of input to a vulnerable web application is stored in the server side storage space (typically database or file system) without any validation. Later, when a victim makes a service request to that vulnerable web application, the web application generates a response that contains the previously stored malicious script verbatim.

Cross Site Scripting exposed in RESTful web services are very wild. According to analysis in 2012 via HP Fortify on Demand[4], cross-site scripting attacks remain a major threat when attempting to secure Web applications. Cross-site scripting (XSS) [5] continuously leads the most wide-spread Web application vulnerabilities lists. Estimates in [11] suggest that 87 percent of all current Web sites are vulnerable to XSS.

Server side solution supports a Cross Site Scripting mitigation mode that significantly reduces the number of connection alert prompts while, at the same time, it provides protection against Cross Site Scripting attacks where the attackers may target sensitive information such as cookies and session IDs. We propose a mechanism that limits the amount of information that can be stolen by any single Cross Site Scripting attack.

An unprotected site providing a forum where users must identify with user name and password is the ideal environment for stored Cross-Site Scripting attacks[6]. In case of forums where cookies are used to provide successful authentication, then the attack is performed as follows:

<SCRIPT>document.location("http://evil.org/steal.cgi?c=+escape(document.cookie);")</SCRIPT>

The simplicity and adoptability of the REST-based web services, however, come with a price. A REST implementation provides no pre-defined security protection methods. So this paper presents the novel idea for avoiding the Cross-site Scripting in RESTful web service.

ISSN 2348 - 7968

I.2. Contributions

This paper has the following contributions:

II.We propose a novel notion that will avoid the Cross site scripting in any type of browsers.

III. We describes how this approach solves the different type of malicious script without requiring the change in existing application server and browser infrastructure.

IV. We provide the case study of how the proposed component is being installed in every application server.

V. Proposed Architecture

Compared to SOAP-Based Web services, a REST-based approach to Web services is much easier to implement since its design simply relies on the HTTP protocol. It uses (a) URIs (Uniform Resource Identifiers) to identify resources; and (b) the GET, PUT, POST and DELETE actions to retrieve, update, create, and delete the resources remotely through Web servers.

A scenario is presented here to explain security threats encountered in an enterprise system that is built upon REST-based Web services. Consider a customer-facing system that allows users to register as customers, browse a catalog, place an order, enter shipment instructions, and pay for the order on the Web. At this time a hacker can act like a legitimate user and they will send the illegal script to the server. When that data is being accessed from server to client, that illegal script will get executed. The illegal script may be of type of executable script.

Example:

<script>
window.location=
'http://attacker/?cookie='+document.cookie
</script>

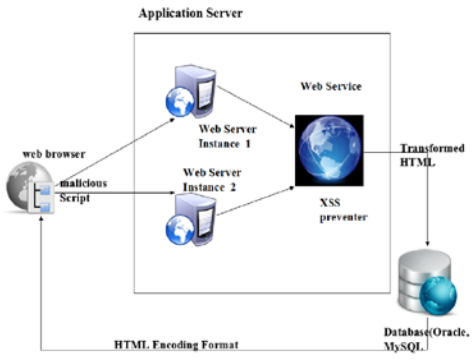

Fig. 1. System Architecture for XSS Preventer

The main purpose of Server side solution is that it is effectively reduces Cross Site Scripting attacks in RESTful web service from side. This architecture is suitable for any type browser and also for any of the application server.

According to the Fig 1, the main components of the proposed architecture are

VI. XSS Preventer - This component is available in the application server. When the request for the service has come this component will make a transformation of the content into unreadable format.

VII. Application server - It is a program that handles all application operations between users and an organization's backend business applications or databases.Ii is used to monitor for high-availability, high-performance distributed application services and support for complex database access.

VIII. Database : By using data access objects instead of accessing the data source directly, the type and implementation of the actual data source is decoupled from its usage. This allows moving from one data source to a different data source without having to change the business logic.

IX. Web Browser: A web browser is a software application for retrieving, presenting and traversing information resources on the World Wide Web.

IX.1. Algorithm for XSS Preventer

Step 1: The attacker injects malicious code into the Restful web service, whose identity is authenticated.
Step 2: When the malicious code is being sent to the Restful web service, that will be stored as transformed content.
Step 3 : When the vulnerable web service is being invoked, the tainted data will be sent to service invoker from the server
Step 4: The data with malicious code send to the web browser
Step 5: XSS preventer will avoid executing the malicious code in the browser

The above algorithm mainly used for preventing stored XSS (Type 1). Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information.

The following JSP code segment queries a database for an employee with a given ID and prints the corresponding employee's name.

```
<%...
 Statement stmt = conn.createStatement();
 ResultSet rs = stmt.executeQuery("select * from emp
where id="+eid);
  if (rs != null) {
   rs.next();
   String name = rs.getString("name");
%>
    Employee Name: <%= name %>
```

ISSN 2348 - 7968

The RESTful application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Stored XSS exploits occur when an attacker injects dangerous content into a data store that is later read and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

The Stored XSS is avoided by XSS Preventer, by using the following code sample

Sample Code:

```
char[] chars = data.toCharArray();

 for (int i = 0; i < chars.length; i++){
  buf.append("*"+(int)chars[i]);
 }
System.out.println(buf.toString())
```

The typical XSS attack and its steps is being depicted in the following Fig 2

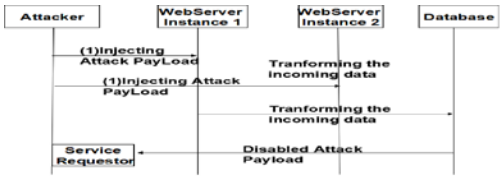

Fig 2. A typical XSS attack in RESTful web Service

In the above Fig 2 describes about how the attacker inject the malicious payload into the database and it tells about how it is being avoided during the service request or invocation.

X. Implementation and Discussion

This proposed system(XSS Preventer) has been developed in Netbeans 7.1 and JBOSS 7.1.0. The XSS preventer can be packaged as a component which can be installed in any different application server or web server. The component can be act as a plug in and it can be loaded and tested in various web server like JBoss, Apache Tomcat, IIS Server.

Also this will work well without any modification in web browsers like mozilla firefox, Google Chrome and Internet Explorer etc

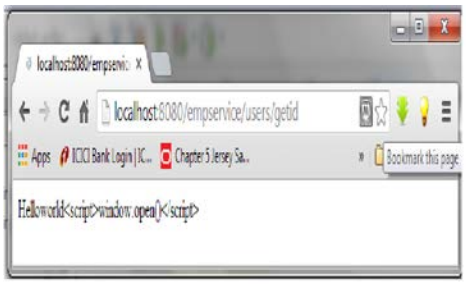

Fig 3. Output Sample with XSS Preventer

In Fig 3, the proposed system has given the output that avoids the running of malicious script (e.g.<script>window.open()</script>).

Fig 4. Transformation of malicious content into unreadable format

The Fig 4, says that the malicious script which is stored in the database has been nullified, so that we are not getting any unwanted web pages running in HTTP request and HTTP response.

TABLE I
EVAULATION OF XSS PREVENTER

XSS Preventer	Mozilla	Chrome	Internet Explorer	Opera
JBOSS	√	√	√	√
Apache	√	√	√	√
Oracle	√	√	√	√
WebLogic	√	√	√	√

From the TABLE 1, it shows that the proposed system works successfully in all types of application servers and web browsers.

XI. Abbreviation and Acronyms

REST - Representational state transfer
XSS - Cross - Site Scripting
HTTP - Hyper Text Transfer Protocol
SOAP - Simple Object Access Protocol

ISSN 2348 - 7968

V. Conclusion

We have presented the strategy for providing the security in RESTful Web service. The usefulness of this method XSS Preventer has been tested and evaluated in different web browsers and application servers. Since the component XSS Preventer is having less code, it can be attached as a plug-in in server without any modification. Our approach works well in multilevel community infrastructure.

References

[1]http://en.wikipedia.org/wiki/Crosssite_scripting
[2]"2013 OWASP top 10 list" - https:// www. owa s p . o r g / i n d e x . p h p / Top_10_2013-Top_10
[3] "2011 CWE/SANS top 25 most dangerous software errors" - http://cwe.mitre.org/top25/#Listing
[4]Sahba Kazerooni, Takeaki Chijiiwa, Subramanian Ramanathan, and Jevonne Peters, *HP 2012 Cyber Risk Report*, Feb 2013.
[5]CERT "*Advisory CA-2000-2002- Malicious HTML Tags Embedded in Client Web Requests*" http://www.cert. org/advisories/CA-2000-02.html, 2000.
[6]A. Duraisamy, M.Sathiyamoorthy, S.Chandrasekar, "*A Server Side Solution for Protection of Web Applications from Cross-Site Scripting Attacks*", International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278 - 3075, Volume-2, Issue-4, March 2013.
[7]W3C Web Services Activity http://www.w3.org/2002/ws/.
[8] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine*, 2000.
[9] Krishna Chaitanya Telikicherla and Harigopal K B Ponnapalli, "*Anatomy of Cross Site Scripting (XSS) Vulnerabilities*", CSI Communications, Nov 2013
[10]WhiteHat Security. Website Security Statistics Report. http://www.whitehatsec.com/home/ resource/stats.html, 2008.
[11]Gudgin M., Hadley M., Mendelsohn N., Moreau J., and Nielsen H., "*SOAP Version 1.2 Part 1: Messaging Framework*," W3C Recommendation,June 2003

Authors' information

Angel N, received her MCA degree from Standard Fireworks College for Women affiliated to Madurai Kamaraj University, M.E(CSE) degree from Sathyabama University.

She is currently working as a Associate Professor in the Department of Master of Computer Applications in St.Joseph's College of Engineering, Chennai. Her area of interest includes Web Security, Advanced Java Programming, Component Based Technology.

Chandra Sekar A, received his B.E(CSE) degree from Angala Amman College of Engineering and Technology affiliated to Bharathidasan University, M.E(CSE) degree from A.K. College of Engineering affiliated to Madurai Kamaraj University, and Ph.D in Information and Communication Faculty (Computer science & Engineering) from Anna University, Chennai, India.
He is currently working as a Professor in the Department of Computer Science & Engineering in St.Joseph's College of Engineering, Chennai. His area of interest includes Network Security and Analysis of Algorithms.