International Journal of Advance Research in Science and Engineering
Vol. No. 5, Issue No. 07 , July 2016
www.ijarse.com
IJARSE
ISSN 2319 - 8354

SURVEY ON CROSS-SITE SCRIPTING (XSS) ATTACKS AND COUNTER MEASURES

[1]Anusree K, [2]Midhun T P

[1]Computer Science and Engineering,
Vimal Jyothi Engineering College, Kannur, Kerala, (India)
[2]Asst.Professor, Computer Science and Engineering,
Vimal Jyothi Engineering College, Kannur, Kerala, (India)

ABSTRACT

Web application is a client-server software applicationin which the client runs in web browser. Most web applicationshave critical bugs influencing their security, which makesthem defenseless against attacks by hackers and organized crime.Cross-site scripting(XSS) is among the most serious and commonthreats in Web applications today. Attackers inject maliciouscode via inputs, thereby causing unintended script executions byclient's browsers. There should needed a mechanism to mitigateXSS attacks. There are multiple XSS solutions ranging fromsimple static analysis to complex runtime protection mechanisms.This survey discuss XSS attacks and various solutions to mitigateXSS attacks.

I INTRODUCTION

Web applications are one of the most prevalent platformsfor information and service delivery over the Internettoday.Web applications are used to perform most major tasks or websitefunctions. Web applications are popular due to the ubiquityof web browsers and convenience of using web browser as aclient to update and maintain web applications easily. Webapplication security is the process of securing confidentialdata stored online from unauthorized access and modification.Most web applications have critical bugs. Security threadscan compromise the data by an organization is hackers withmalicious intention try to gain access to malicious information.The most well-known dangers in the web environment is cross-sitescripting (XSS).

Cross Site Scripting (XSS) is a vulnerability in webapplications that permit an attacker to inject malicious code,typically including JavaScript code, into a web page. If userinput is not validated correctly, it could contain code thatkeeps along with server code in a clients browser. XSS usuallyaffects victims web browser on the client-side. There shouldneeded a mechanism to mitigate XSS attacks.

There are various XSS countermeasures are existing today.This survey discuss major security issues related to webapplications mainly for XSS attack and various solutions tomitigate XSS attacks.

II OVERVIEW OF CROSS-SITE SCRIPTING ATTACK

Cross-site Scripting (otherwise called XSS) is one ofthe dangerous and most common application layer hackingtechnique. In 2010 XSS positioned first in the Miter CommonWeakness Enumeration (CWE)/SANS Institute list of Top 25Most Dangerous Software Errors and second in the Open WebApplication Security Project (OWASP) Top 10 list of securityrisks[1]. Several major websites including Facebook, Twitter,Myspace, eBay, Google, and McAfee have been the targetsof XSS exploits. XSS flaws exist in Web applications writtenin different programming languages such as PHP, Java, and.NET where application webpages reference unrestricted userinputs. Attackers infuse malicious code by means of theseinputs, in this way bringing on unintended script executionsby client's browsers. Researchers have proposed multiple XSSsolutions ranging from simple static analysis to complex runtimeprotection mechanisms. However, vulnerabilities continueto exist in many Web applications due to developer's lackof understanding of the problem and their unfamiliarity withcurrent resistance's qualities and restrictions [1].

2.1 XSS Attack Issues

Some of the common issues of XSS attacks are givenbelow

- Session hijacking: One of the most common issue of XSS attack is session hijacking. Credentials stored in cookie can be stolen by attacker. Then attacker can easily steal user's identity and access his confidential information. In the case of a normal user, their personal data, credit card information, bank account information can be misused. Session hijackingbecome dangerous for users with high privileges like administrator.If their accounts are stolen via XSS, then entire websitecan be controlled by attacker.
- Misinformation: Another critical threat from XSS is misinformation. XSS attacks may be include malicious code and it can spy on user's history of site visited and what are the previous user click etc. It results loss of privacy.
- Denial of Service: In view of an enterprise, it is essential feature that their Web applications are should be accessible all the time. However, malicious script can lead to loss of availability. For example, it may redirect user's browser to other web site.
- Browser exploitation: Malicious script can redirect client browsers to an attacker's site, so that the attacker is able to take advantage of specific security hole in web browsers to control user's computer by executing arbitrary commands. Example: install Trojan horse programs on the client, etc.

2.2 XSS Attack Mechanism

Cross-site scripting is one of the most common applicationlayer attack. Cross Site Scripting attacks exploits the user's trust in website [2]. XSS defined as the injection ofmalicious script into website, then cause a unintended scriptexecution by client's browser and perform some maliciousactivity. The main key feature of this attack is that,

thismistrusted content is not easily recognizable and take actionagainst it. So that this attack exploit users trust in website.

Figure 1 shows a simple XSS attack through URL. Firstconsider a simple web form that will simply accept a querystring from user and display the name of the user on page.In the normal case there is no problem. There is no script isinjected and no XSS attack is performed. But if an attackerplace or try to passes a script in the query string then itwill make a problem. It will run the script and the html tagsand images could possibly wipe out the original page andshow something else entirely. Such an exploitation techniqueis called a XSS attack.

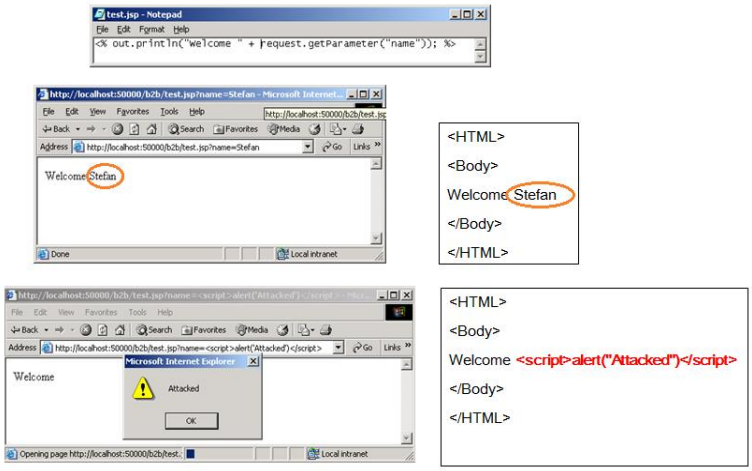

Fig.1.A typical XSS attack

2.3 Types of XSS Attacks

Depending on the ways HTML pages reference user inputs,XSS exploits can be broadly classified as reflected,stored, orDOM-based [3].

- Reflected XSS Attacks: This attack is also known as non persistent XSS attack. This kind of attack occurred due to malicious input given by the user and that is processed by server-side scripts without properly verifying or sanitizing the input. And the server should reflect this input along with some other data as HTTP response. A very common example is a site search engine. If the input given is a script then the same script is reflected by the server resulting in a reflected XSS. When an attacker inject his mischievous script into a search query, a search box, or the end of an url, it's called Reflected XSS Attack as the script gets executed and is reflected along with the server response.
- Stored XSS Attacks: This attack is also known as persistent XSS attack. This attack occur, when a server program stores user input containing injected code in a persistent data store such as a database and then references it in a webpage [4]. Attacks against social networking sites commonly exploit this type of XSS flaw. Stored XSS attack is the most destructive of all cross site scripting attacks. Message boards that allow users to post messages provide breeding ground for persistent XSS. An hacker just has to submit XSS exploit code which is designed in a way to cause harm to an area of a web site that is likely to be visited by

other users. These areas could be blog comments, user reviews, message board posts, chat rooms, HTML e-mail and numerous other locations. At the point when a client visits the tainted website page, the code is naturally executed.

- DOM Based XSS Attacks: DOM based XSS attack is also known as, type 0 XSS attack. DOM based XSS means XSS vulnerability that appers in DOM. The code is executed when a client side script access the URL to modify the DOM. The prerequisite is for the defenseless site to have an HTML page that uses data from the "document.location" or "document.URL" or "document.referrer" in an insecure manner.

III XSS Defenses

There exist so many traditional network security techniques,like firewalls and cryptography-based mechanisms. But Attackers have managed to continue exploiting XSS attacksacross Internet web applications. The use of specific securedevelopment techniques can help to mitigate the problem, butit is not always enough [5]. XSS attack prevention mechanismsidentify and block the XSS attacks. Most of the XSSprevention techniques are done at the point of deployment. Sothis techniques are deployed on either the server side or clientside. For XSS detection, testing tools are commonly used.

Tools are used for identifying XSS vulnerabilities. But most ofthe cases XSS vulnerability remains undetected. So additionalsafeguard is needed. Detection and prevention techniques aredesigned for different purpose, but their technologies aresimilar. Detection is best used in situations where is a need toexplain what happened in an attack, whereas prevention stopsattacks. Different XSS prevention techniques are discussedbelow:

3.1 BEEP : Browser Enforced Embedded Policies

A simple mechanism for preventing XSS is Browser Enforced Embedded Policies (BEEP). Here security policiesare embedded with in web page and enforced by browser.The idea is that a web site can embed a policy in its pages that specifies which scripts are allowed to run. The browser,which knows exactly when it will run a script, can enforcethis policy perfectly. The main idea behind BEEP is that, thesecurity policy is expressed as a trusted JavaScript functionthat the web site embeds in the pages it serves. This functioncan be call as security hook. A browser with BEEP passeseach script it detects to the security hook during parsing andwill only execute the script if the hook approves it. BEEP hasseveral advantages also. One main feature of BEEP is policies. There are two kinds of policies.

First policy is a whitelist, in which the hook functionincludes a one-way hash of each legitimate script appearing inthe page. The main advantage of whitelists is, it provide bettersecurity. But the disadvantage is hashing overhead. Secondpolicy is a DOM sandboxing. Here, the web applicationstructures its pages to identify content that might includemalicious scripts. It is just reverse of whitelists. It will identifythe corresponding blacklist. The coming scripts are comparedwith blacklists, if match found,the corresponding scriptwill be blocked. Policies can be easily modified over time.

DOM sandboxing provide another level of security. But themain problem is finding the set of all possible malicious stringis very complex task[6].The Limitations are this method requires modificationin server software as well as in the client browser. Thatis, it needs to be implemented by users, but most of usersare unaware of damage due to XSS and some of themare unwilling to do additional effort for security of theirsystems. Another problem is runtime overhead increases. Thatis, comparing with blacklist and whitelist cause the problemof time consumption.

3.2 SWAP : Secure Web Application Proxy

SWAP is a server-side solution for detecting and preventingcross-site scripting attacks [7]. BEEP technology requirebrowser modification. SWAP will reduces this problem. SWAPis a reverse proxy, which relays all traffic between the Webserver that should be protected and its visitors. The main twocomponents of SWAP are,

- Reverseproxy(rproxy)
- JavaScript Detection Component(jstester)

The proxy forwards each Web response, before sending itback to the client browser, to a JavaScript detection component,in order to identify JavaScript content. Reverseproxy installedin front of webserver and intercept all HTML responsefrom the sever and subjects them to analysis by the JavaScriptdetection component. JavaScript detection component, firstidentity all the legitimate javascript and convert this javascriptto scriptid.

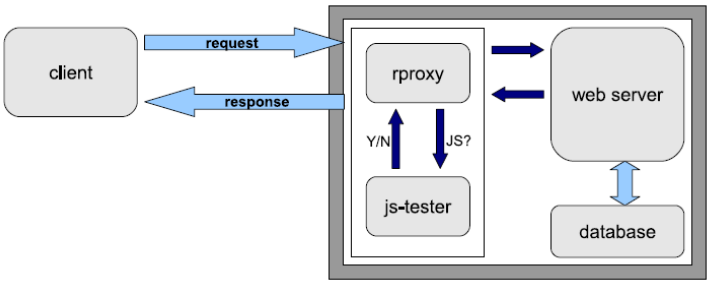

Fig.2. Scheme of SWAP setup

Some limitations are associated with SWAP. That isSWAP introduces a performance overhead as it uses a full sizedweb browser for JavaScript detection. Some scripts areconsidered as benign for one class of browser but for other itmay be malicious.

3.3 Pattern filtering approach

This is a mechanism to secure java web applications fromby applying a framework based on pattern matching approach.In computer science, pattern matching is the act of checkinga given sequence of tokens for the presence of constitutes ofsome pattern. Pattern matching can be used to filter data ofcertain structure. In the case of XSS

prevention, validation isthe act of filtering user input so that all malicious part of itare removed. Two classification strategies are,

- Blacklisting
- Whitelisting

Validation outcomes are,

- Rejection : The input is simply rejected
- Sanitisation : All invalid parts of the inputs are removed.

This framework consist Request/Response Analyser andModifier modules. The first interaction of request is to RequestAnalyser/Modifier Module which decides about request ismalicious or not and takes decision accordingly. Responseanalyser and Modifier module deals with the data to bereturned the client, it modifies the malicious response toharmless data. Attack Recorder and Response Rejecter Modulerecords the malicious Request/Response for future use. JavaRegex has been used for pattern generation and matching themalicious attack signatures. Patterns are generated by usingregular expressions [8].

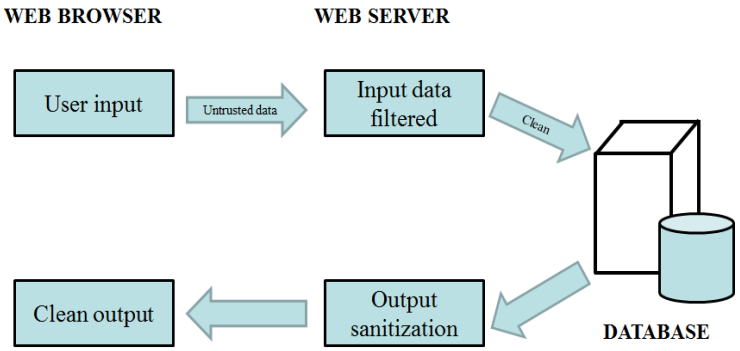

Fig.3. Process of filtering

To defend against persistent Cross-Site Scripting attacks,a simple task has to be performed for input filtering: Any datafrom the input must be transformed or filtered in a way thatit is not executed by a browser if sent to it. To avoid XSS,developers must sanitize the user's input before storing it inthe database.

3.4 Noxes

Noxes acts as a personal firewall that allows or blocksconnections to websites on the basis of filter rules, which arebasically user-specified URL white-lists and blacklists [9].It will be run as a background service on the desktop of auser. All the incoming and outgoing connections in that localmachine go through this noxes. The main idea is that, toblock and detect malware and protect users against remotelyexploitable vulnerabilities. In personal firewall, connectionrequest detected will match with firewall rules and then userwill decided to block the connection, allow it or create apermanent rule that specifies, whether such request is detectedagain in future.

For preventing XSS attack, Noxes allows the user to createfilter rules for web requests. There are three ways of creatingrules:

- Manual creation
- Firewall prompts
- Snapshot mode.

In manual creation, user will create a rule database andenter a set of rules. In firewall prompts, user can interactivelycreate a rule whenever a connection request is made that doesnot match any existing rule. Atlast, in snapshot mode, usercan use special snapshot mode integrated into Noxes to createa browsing profile to automatically generate a set of permitrules.

When Noxes receives a request to fetch a page, it goesthrough several steps to decide if the request should beallowed. It first uses a simple technique to determine if arequest for a resource is a local link. If the domains arefound to be identical, the request is allowed. If a request beingfetched is not in the local domain, Noxes then checks to seeif there is a temporary filter rule for the request. If there is atemporary rule, the request is allowed. If not, Noxes checks itslist of permanent rules to find a matching rule. If no rules arefound matching the request, the user is prompted for actionand can decide manually if the request should be allowed orblocked [10].

Limitation are, Noxes is a client-side web-proxy thatconveys allWeb traffic and acts as an application-level firewall.However, in comparison to SWAP, Noxes needs user-specific settings and also it requires user interaction when any newevent occurs that does not matches with the current firewallrules. Such user awareness is not always guarantied.

3.5 Content security policy

Content security policy is an added layer of security thathelps to detect and mitigate certain types of attack includingXSS attack. A newly developed web application can useCSP to mitigate XSS attack by allowing particular scriptsfor execution at client side that are specified in policy andblocking inline JavaScripts. CSP is backward compatible. CSPis used to constrain the browser viewing your page so that itcan only use resources downloaded from trusted source. CSPoffers with white-list based content resolution mechanism.

CSP contains some disadvantages. CSP is just a additionallayer of security applied at client side. It is not a replacementof traditional mechanism of validation and escaping of inputand output on the server-side. It also requires manual changesto be done in each and every page of website. Applying CSPmanually is tedious task for large web application because theweb administrators have to change server-side code to identifywhich codes and resources are used by web pages. And alsothese scripts need to be isolated from web page.

Content security policy (CSP) is a content restrictionmechanism, now supported by all major browsers, that offersthorough protection against XSS. Unfortunately, simply enablingCSP for a web application would affect the application'sbehavior and likely disrupt its functionality. AUTOCSP is anautomated technique for retrofitting CSP to web applications.

Main steps of AUTOCSP is :

- Dynamic taint tracking

- Firewall prompts

AUTOCSP leverages dynamic taint analysis to identifywhich content should be allowed to load on the dynamicallygeneratedHTML pages of a web application and automaticallymodifies the serverside code to generate such pages with theright permissions [11]. Given a web application, AUTOCSPoperates in four main phases. First, it marks as trusted allknown values in the web application's server-side code and exercisesthe web application while performing positive dynamictaint tracking. The result of this phase is a set of dynamicallygenerated HTML pages whose content is annotated with taintinformation. Second, it analyzes the annotated HTML pages toidentify which elements of these pages are trusted. Basically,the tainted elements are those that come only from trustedsources. Third, AUTOCSP uses the results of the previousanalysis to infer a policy that would block potentially untrustedelements while allowing trusted elements to be loaded. Fourth,AUTOCSP automatically modifies the server-side code ofthe web application so that it generates web pages with theappropriate CSP. But this fourth step is very difficult to do.Server side code modification is one of the difficult task.

3.6 Blueprint

Blueprint is a new XSS defense strategy designed to beeffective in widely deployed existing web browsers, despiteanomalous browser behavior. This approach try to minimizetrust placed on browsers for interpreting untrusted content[5].Blueprint enables a web application to effectively take controlof parsing decisions. This approach employs techniques tocarefully transport and reproduce this blueprint in the browserexactly as intended by the web application, despite anomalousbrowser parsing behavior. HTML code is processed by thebrowser's HTML lexerand parser, which produces a parsetree. This tree is supplied as input to the document generationstage, and HTML parsing activity is complete once we enterthis stage. The implicit goal of XSS prevention is to safelycommunicate untrusted content to the browser's documentgeneration stage, such that the browser-generated parse treeis free of script nodes. Here content filtering based defensesutilizes and attempt to anticipate the behavior of browserparsers to ensure script nodes are not created. However, aftersupplying untrusted HTML, the web application has no further

control over the resulting parse tree. The web applicationtherefore cannot ensure the resulting parse tree is free of scriptnodes because browser parser behavior cannot be reliablypredicted due to parsing quirks. The main idea of this approachis to eliminate any dependence on the browser's parser forbuilding untrusted HTML parse trees. The main two stepsperformed are: 1) On the application server, a parse tree isgenerated from untrusted HTML with precautions taken toensure the absence of dynamic content nodes in the tree. 2)On the client browser, the generated parse tree is conveyed tothe browser's document generator without taking vulnerablepaths. This two-step process ensures untrusted content generatedby the browser is consistent with the web application'sunderstanding of the content. The generated document reflectsthe application's intention that the untrusted content does notcontain scripts, therefore all unauthorized script execution isprevented.

IV CONCLUSION

Security is important factor we need to keep informationas confidential.There are many security and privacy issues in many web applications. The most well known dangers inthe web environment is cross-site scripting (XSS). This papersurveyed about the main defensive mechanisms against XSSattack. The first part of this survey deals with cross-site attackmechanisms and types in detail. And the last section discussesdifferent approaches to mitigate XSS attacks.

REFERENCES

[1]. Open web application security project, XSS (cross- site scripting), prevention cheat sheet
[2]. R. Hansen A. Rager P.D Petkov S. Fogie, J. Grossman, Xss attacks: Cross site scripting exploits and defense.
[3]. C. Malarvizhi V. Nithya, S. LakshmanaPandian, A survey on detection and prevention of cross-site scripting attack, 2015
[4]. Dr. RenuDhirJyotiSnehi, Web client and web server approaches to prevent xss attacks, 2013
[5]. A. Kiezun, Automatic creation of sql injection and cross-site scripting attacks, 2009
[6]. N. Swamy T. Jim and M. Hicks, Defeating script injection attacks with browser enforced embedded policies, 2007
[7]. Peter Wurzinger, Christian Platzer, Christian Ludl, EnginKirda, and Christopher Kruegel SWAP: Mitigating XSS Attacks using a Reverse Proxy.
[8]. Imran Yusof, Al-Sakib Khan Pathan, Preventing Persistent Cross-Site Scripting (XSS) Attack By Applying Pattern Filtering Approach, 2009
[9]. EnginKirdaa, NenadJovanovic, Christopher Kruegel, Giovanni Vigna, Client-side cross-site scripting protection, 2009
[10]. MattiaFazzini, PrateekSaxena, Alessandro Orso, AutoCSP: Automatically Retrofitting CSP to Web Applications, 2013
[11]. Mike TerLouw, V.N. Venkatakrishnan, BLUEPRINT: Robust Prevention of Cross-site Scripting Attacks for Existing Browsers, 2009