International Journal of Control Theory and Applications

ISSN : 0974–5572

© International Science Press

Volume 9 • Number 40 • 2016

A Taxonomical Approach for Cross Site Scripting Attacks

Anusha Kadambari Shankar[a] and G. Usha[b]

[a]*UG Student, Dept. of Software Engineering, SRM University Kattankulathur Kanchipuram Dt.*

E-mail: anushishankar@gmail.com

[b]*Assistant Professor(Sr.G), Dept. of Software Engineering SRM University Kattankulathur Kanchipuram Dt.*

E-mail: usha.g@ktr.srmuniv.ac.in

Abstract: Nowadays, the Internet has become the preferred platform for users to carry out several activities of their day to day lives, including activities that involve sensitive information such as E-commerce, E-governance, E-banking, Shopping Portals and more. Web Applications have become pervasive in all aspects of life because of the ease of remote accessibility for its users. But as the usage of web increases every day, it has also brought into light the dangerous side of html. Security has, therefore become one of the major concerns regarding cyberspace. In this paper, we focus on the specific problem of cross site scripting (xss) attacks. We present a taxonomy study on cross site scripting attacks. We have also discussed various types of vulnerabilities present and risks produced for this attack. We have also produced a survey on the existing methods of defense against such attacks.

Keywords: *Cross site scripting, injection attack,server side scripting, client side scripting.*

1. INTRODUCTION

Cross-Site Scripting (XSS)[1] assaults are a kind of injection attack, in which malevolent scripts are infused into generally harmless and trusted sites. XSS assaults happen when an assailant uses a web application to send malicious code, usually as a browser side script, to an alternate end client. Faults that permit these assaults to succeed are very far reaching and occur anyplace a web application utilizes contribution from a client for an output it produces without validating or encoding it.

An assailant can utilize XSS to send a malignant script to a clueless client. The client's[2] program has no real way to realize that the script ought not to be trusted, and will execute the script. Since it assumes that the script originated from a trust-able source, the malicious script gains access to cookies, session tokens, or other delicate data held by the browser and utilized with that site. These scripts can even revise the substance of the HTML page.

In the event that the cookies, which usually contain session identifier data, can be perused by the JavaScript, the assailants can utilize them on their own browser and login to the web application under the identity of the victim. On the off chance that that does not work, the attacker can in any case read private data from the pages, for example, read CSRF tokens and make requests whilst masquerading as the client. Usually, XSS vulnerabilities require some sort of interaction on the client's side to trigger the vulnerability, either by means of social engineering, or visiting a particular page. That is the reason it's frequently not considered important by engineers, but rather if left unpatched, can be exceptionally hazardous.

The organization of the paper is as follows: In Section II we present the taxonomy of XSS attacks. Section III concentrates on how the XSS attacks will affect the Android mobiles. Section IV we describe about the risks and threats of XSS attacks. Section V discusses about the solutions offered in literature to defend against these attacks. Finally, we conclude our paper with future work.

2. TAXONOMY OF XSS ATTACKS

Now we describe about the various taxonomical solutions against cross site scripting attacks. Fig 1 explains about the Taxonomical classification of proposed technique.

2.1. Detection model for XSS attacks

In [3] the proposed framework is an identification method for XSS assault comprising of both Persistent and non-Persistent cross-site scripting assault. The proposed method has distinctive engineering for customer and server side. The Persistent XSS assault comprises of five modules: Input Sanitizer, Filtering Module, Filtered Output, Attack Rule Library and Attack Repository. The recognition of Non-Persistent XSS assault comprises of five squares: Input Checker, Prevention utilizing CSP (Content Security Policy), Notify Client, Attack Rule Library and Attack Repository.

2.2. Pixy: tool for detecting web application vulnerabilities

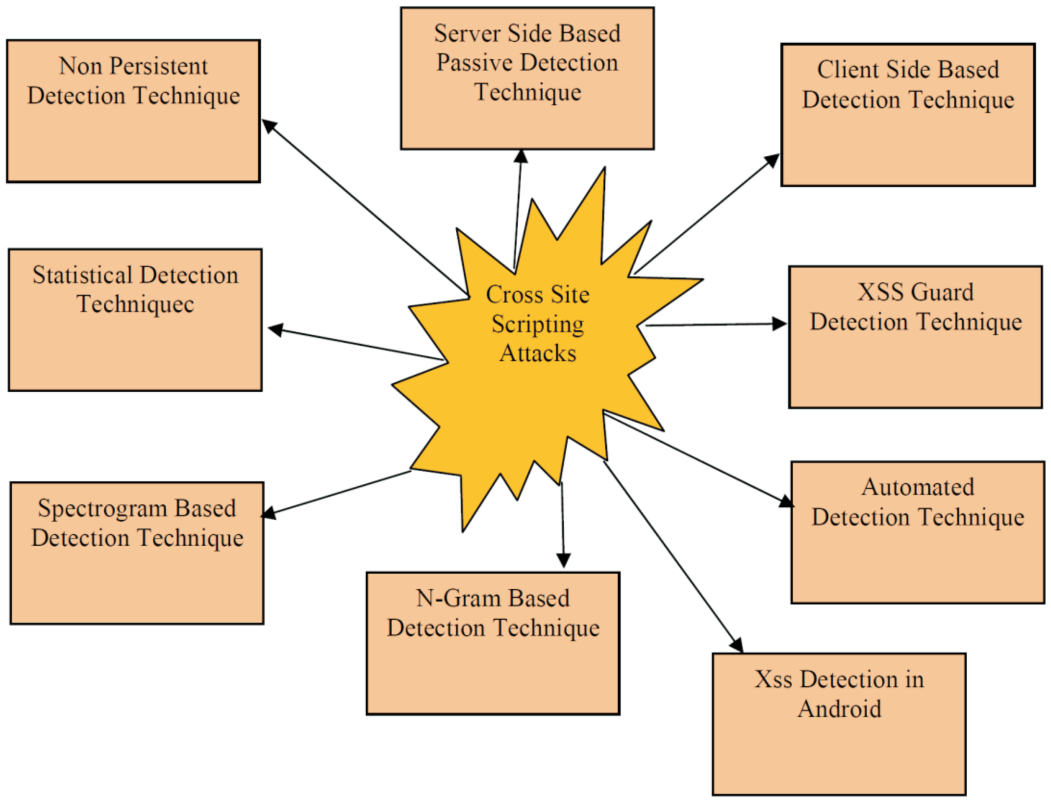
Non Persistent Detection Technique
Server Side Based Passive Detection Technique
Client Side Based Detection Technique
Statistical Detection Techniquec
Cross Site Scripting Attacks
XSS Guard Detection Technique
Spectrogram Based Detection Technique
Automated Detection Technique
N-Gram Based Detection Technique
Xss Detection in Android

Figure 1: Taxonomy of Cross Site Scripting Attacks

In[4] they address the issue of susceptible Web applications by methods for static source code examination. All the more unequivocally, they utilize stream delicate, between procedural and setting touchy dataflow investigation to find helpless focuses in a program. Likewise, pseudonym and strict investigation are utilized to enhance the rightness and accuracy of the outcomes. The exhibited ideas are focused at the general class of pollute style vulnerabilities and can be connected to the recognition of helplessness sorts, for example, SQL infusion, cross-site scripting, or summon infusion. Imp, the open source model execution of our ideas, is focused at recognizing cross-site scripting vulnerabilities in PHP scripts. Utilizing the instrument, 15 already obscure vulnerabilities in three Web applications are distinguished, and reproduced 36 known vulnerabilities in three other Web applications. The watched false positive rate is at around half (i.e., one false positive for every weakness) and subsequently, sufficiently low to allow compelling security reviews.

2.3. Spetcrogram:Markov Chain model for web anomaly detection

In [5], they displayed another model and sensor structure that offers a great adjust under this imperative and exhibits change over some current methodologies. Spectrogram is a system arranged sensor that progressively collects bundles to recreate content streams and figures out how to perceive true blue web-layer script input. They portray a proficient model for this errand as a blend of Markov's chains and infer the comparing preparing calculation. The assessments demonstrate huge recognition comes about on a variety of certifiable web layer assaults, looking at positively against other AD approaches.

2.4. XSSDS: Server side detection

In[6], they proposed a uninvolved recognition framework to recognize fruitful XSS assaults. In light of a prototypical execution, we inspect our approach's precision and check its identification abilities. They ordered an informational index of 500.000 individual HTTP ask for/reaction sets from 95 well known web applications for this, in blend with both genuine word and physically made XSS-abuses; The discovery approach brings about an aggregate of zero false negatives for all tests, while keeping up a phenomenal false positive rate for over 80% of the inspected web applications.

2.5 Noxes: A client side solution

In [7] it presents Noxes, which is, to the best of our insight, the principal customer side answer for relieve cross-site scripting assaults. Noxes goes about as a web intermediary and utilizations both manual and consequently created principles to moderate conceivable cross-website scripting endeavors. Noxes viably shields against data spillage from the client's condition while requiring negligible client communication and customization exertion.

2.6. XSS Guard: Precise dynamic Prevention of XSS attacks

In [8], the creators proposed XSS-Guard, another system that is intended to be an anticipation component against XSS assaults on the server side. XSS-Guard works by progressively taking in the arrangement of scripts that a web application plans to make for any HTML ask. This approach additionally incorporates a powerful system for recognizing scripts at the server side and evacuates any script in the yield that is not proposed by the web application. They talk about broad test comes about that show the strength of XSS-Guard in keeping various genuine XSS misuses.

2.7. An automaton based approach for cross site scripting

In [9], the creators proposed a straight based robot approach called XSS Chaser which keeps us applications from XSS assaults. This approach performs string examination to produce powerless examples to counteract XSS. These examples are created utilizing forward a regressive understanding. The test result demonstrates that our approach gives better reaction time contrasted with existing Techniques

2.8. Improved N-gram approach for detecting XSS attacks

In [10], they recognize a gathering of elements from pages and utilize them to create classifiers for XSS discovery. Furthermore, they exhibit an enhanced n-gram show (a model got from n-gram display) worked from the components to order site pages. Thirdly, they propose an approach in view of the mix of classifiers and the enhanced n-gram model to recognize XSS in OSN. At last, a technique is proposed to reenact XSS worm spread in OSN to get more precise investigation information. Our analysis comes about exhibit that our approach is viable in OSN's XSS location.

Until 2005, two kinds of XSS attacks were distinguished, Stored XSS and Reflected XSS. In 2005, Amit Klein characterized a third sort of XSS, which he named DOM Based XSS.

2.8.1. Store XSS AKA Persistant type XSS

Stored XSS usually happens when client information is stored on the objective server, for example, in a database, in a message discussion, guest log, remark field, and so forth. And afterward a victim can recover the stored information from the web application without that information being made safe to render in the program. With the appearance of HTML5, and other browser innovations, we can imagine the assault payload being forever stored in the victim's browser, for example, a HTML5 database, and failing to be sent to the server at all.

2.8.2. Reflected AKA non-Persistant type XSS

Reflected XSS happens when client info is instantly returned by a web application in an error message, query output, or whatever other response that incorporates a few or the majority of the information given by the client as a feature of the request, without that information being made safe for the browser, and without permanently putting away the client given information. At times, the user provided information may never at any point leave the browser.

2.8.3. DOM based XSS

As defined by Amit Klein, who published the first article about this type of attack, DOM Based XSS is a type of XSS where the entire corrupted information flow from source to sink occurs in the browser, i.e., the wellspring of the information is in the DOM, additionally the sink too is in the DOM, and the information stream never leaves the browser. For instance, the source (where malignant script is read) could be the URL of the page (e.g., document.location.href), or it could be a component of the HTML, and the sink is a delicate technique call that causes the execution of the malicious information (e.g., document.write).

2.8.4. Broader classification

For a long time, most people thought these (Stored, Reflected, DOM) are three distinct sorts of XSS, however actually, they often overlap. You can have both Stored and Reflected DOM Based XSS. You can likewise have Stored and Reflected Non-DOM Based XSS as well, however that is confounding, so to help elucidate things, beginning in mid-2012, the research community proposed and began utilizing two new terms to help distinguish the types of XSS that can happen:

1. Server XSS happens when untrusted client-provided information is incorporated into a HTML response created by the server. The source of this information could be from the request, or from a stored area. In that capacity, you can have both Reflected Server XSS and Stored Server XSS. For this situation, the whole weakness is in server-side code, and the browser is just rendering the reaction and executing any legitimate script inserted in it.

2. Client XSS happens when untrusted client-given information is utilized to refresh the DOM with a risky JavaScript call. A JavaScript call is viewed as hazardous on the off chance that it can be utilized to bring valid JavaScript into the DOM. The source of this information could be from the DOM, or it could have been sent by the server (by means of an AJAX call, or a page load). A definitive source of the information could have been from a request, or from a stored area on the customer or the server. In that capacity, you can have both Reflected Client XSS and Stored Client XSS. With these new definitions, the meaning of DOM Based XSS doesn't change. DOM Based XSS is just a subset of Client XSS, where the wellspring of the information is some place in the DOM, instead of from the Server. Next we discuss how XSS attacks affect Android devices in detail.

3. XSS IN ANDROID

Over the last few years, smart phones, and in particular Androids have become our constant companion devices, having all but melded with our hands as they have become integrated into almost every aspect of our daily lives. Smart phones have evolved to be so much more than just machines used to make calls. They are our remote, portable access to the internet and cyberspace. You can access mails, other accounts, websites, even making online purchases through your phone now. Moreover, social networking and smart phones are practically hand in hand as of now. Owing to the fact that smart phones are such a large, constant presence in every small part of our day to day activities, an attacker who gains access to your smart phone in any way can monitor much of your life and sensitive data. Cross site scripting attacks can be carried out with our smart phones as targets as well. The following are a few of the attacks aimed at smart phones using XSS.

3.1. Local cross site scripting against Android phones

In [10] they exhibit a XSS assault that objectives Google's most up to date Android form 2.3.4 (can be run additionally on the Android Tablet adaptation 3.01), which was discharged in April 2011. They expect that the assault can likewise be mounted on every single past form of Android. They have built up a proof-of-idea application that, without requiring any authorizations, is able to do: Stealing treats put away in the web program for Web locales of the aggressor's decision; consequently introducing subjective applications from the Android Market without client assent.

3.2. Code Injection attacks on HTML5 based mobile apps

In [11], they found another type of code infusion assault, which acquires the basic reason for Cross-Site Scripting assault (XSS), however it utilizes numerous a bigger number of channels to infuse code than XSS. These channels, special to cell phones, incorporate Contact, SMS, Barcode, MP3, and so on. To survey the commonness of the code infusion weakness in HTML5-based portable applications, they have built up a defenselessness recognition apparatus to dissect 15,510 PhoneGap applications gathered from Google Play. 478 applications are hailed as helpless, with just 2.30% false-positive rate. They have likewise actualized a model called NoInjection as a Patch to PhoneGap in Android to protect against the assault.

3.3. Cross site scripting attacks on Android WebView

In [12], Cross-website scripting assaults or XSS assaults particular to Android WebView are talked about. Cross-website scripting (XSS) is a kind of powerlessness regularly found in web applications. This defenselessness makes it feasible for assailants to run malevolent code into casualty's WebView, through HttpClient APIs. Utilizing this pernicious code, the aggressors can take the casualty's accreditations, for example, treats. The get to control arrangements (i.e., a similar beginning strategy) utilized by the program to secure those accreditations can be avoided by misusing the XSS defenselessness.

3.4. Attacks on Android clipboard

Clipboard information control may prompt normal code infusion assaults, as JavaScript infusion and charge infusion. Moreover, it can likewise bring about phishing assaults, including web phishing and application phishing. Information taking happens when delicate information replicated into the clipboard is gotten to by vindictive applications. In [13] for every classification of assault, they have investigated an expansive number of competitor applications and demonstrate various contextual analyses to show its attainability. Likewise, their application examination process is defined to profit future application advancement and helplessness discovery. Next we talk about the dangers and dangers are created by XSS assaults.

4. RISKS AND THREATS OF XSS

XSS assaults represent a danger not to individual clients alone whose browsers are misdirected, additionally to companies and associations whose sites might be contaminated, giving aggressors a chance to take private client data. For organizations, this can mean both direct expenses and harming hits to their reputation. As per the Hosted Application Scanning Management team at IBM, 17 percent of somewhere in the range of 900 dynamic Web application scan demonstrated a weakness to XSS. However, this information originated from associations with the robust and developed security practices. A review by White Hat Security found that about a half all sites (47.9 percent) are helpless against XSS assaults.

1. **Account hijacking:** A standout amongst the most well-known XSS assault vectors is to seize legitimate client accounts by taking their session cookies. This permits aggressors to mimic victims and get to any delicate data or functionality under their identity.
2. **Stealing credentials:** A functional assault vector for XSS is to utilize HTML and JavaScript with a specific end goal to take client qualifications, rather than their cookies. This should be possible by cloning the login page of the web application and after that utilizing the XSS weakness to present it to the victims.
3. **Sensitive information:** Another capable assault vector for XSS is to utilize it so as to exfiltrate sensitive information (for instance individual identifiable data or cardholder information) or to perform unapproved operations, for example, siphoning reserves.

5. SOLUTION METHODS FOR XSS

In this section we discuss about the solutions provided by to detect XSS attacks.

5.1. Basic Detection and Defenses

To reiterate, an XSS assault is a sort of code injection: client information is erroneously deciphered as malevolent program code. So as to prevent this sort of code injection, data needs to be handled in a secure manner. For a web engineer, there are two basic methods of performing secure information handling: Encoding and Validation.

5.1.1. Filtering

The least difficult and arguably the most straight forward type of XSS defense is to pass every piece of external data through a filter which will eliminate risky keywords, for example, the notorious <SCRIPT> tag, JavaScript commands, CSS styles and different hazardous HTML markup, (for example, those that contain event handlers.) Many web engineers execute their own filtering mechanisms; they generally compose server-side code (in PHP, ASP, or some other web-enabled dialect) to look for catchphrases and supplant them with purge strings. This procedure is in itself not a bad one, however unfortunately the programmers more often than not have more experience than the web developers, and frequently figure out how to bypass straightforward channels by utilizing strategies, for example, hex encoding, Unicode character varieties, line breaks and invalid characters in strings. These procedures should all be handled and that is the reason it is prescribed to utilize some kind of library that has been attempted and tested by the community. The unfortunate side-reaction with these filtering systems is that genuine content is regularly expelled in light of the fact that it hits at least one of the illegal catchphrases.

5.1.2. Disabling

When disabling an XSS attack you are essentially telling the browser that the information you are sending ought to be dealt with as information and ought not to be translated in some other way. On the off chance that an assailant figures out how to put a script on your page, the target won't be influenced considering that the browser won't execute the script in the event that it is appropriately disabled. In HTML you can escape hazardous characters by utilizing &# before its character code. Getting away HTML is relatively simple, however with a specific end goal to appropriately shield yourself from all XSS assaults you have to be able to escape JavaScript, Cascading Style Sheets, and in some cases XML data. There are likewise numerous pitfalls on the off chance that you attempt to do all the disabling on your own. This is the place an Escaping Library comes helpful. The two most mainstream getting away libraries accessible are the ESAPI given by OWASP and AntiXSS that accommodates Microsoft. Both libraries are continually updated to stay aware of the most recent hacker tactics and are kept up by industry specialists who comprehend changing strategies and developing innovations, for example, HTML5.

6. CONCLUSION

Cross-site scripting is a dangerous problem to have. Technology is constantly moving forward and continuously changing, and hacker strategies are getting more complex.

Therefore in order to overcome XSS, all data entered should be considered untrustworthy and needs to be encoded, or transformed into the sort of text that doesn't render in a browser, before being added to your database. Websites need to be scanned regularly for XSS vulnerabilities which may be an unprotected comment-box or other hidden ones. This paper surveys the dangers of XSS attacks, and the existing methods to tackle it. It must be enunciated that defense against this kind of attack is the need of the hour, and exhaustive research must be conducted towards it in order to arrive at a comprehensive solution to prevent such attacks.

REFERENCES

[1] Engin Kirda, Christopher Kruegel, Giovanni Vigna, and Nenad Jovanovic. Noxes: A client -side solution for mitigating cross site scripting attacks. In Proceedings of the 21st ACM Symposium on Applied Computing (SAC), 2006.

[2] 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.

[3] N. Jovanovic; C. Kruegel; E. Kirda. Pixy: a static analysis tool for detecting Web application vulnerabilities.

[4] Yingbo Song, Angelos D. Keromytis and Salvatore J. Stolfo. Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic. Network and Distributed Security Symposium, 2009 San Diego, California

[5] Martin Johns; Björn Engelmann; Joachim Posegga. XSSDS: Server-Side Detection of Cross-Site Scripting Attacks. Computer Security Applications Conference, 2008. ACSAC 2008. Annual

[6] Bisht P., Venkatakrishnan V.N. (2008) XSS-GUARD: Precise Dynamic Prevention of Cross-Site Scripting Attacks. In: Zamboni D. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2008. Lecture Notes in Computer Science, vol 5137. Springer, Berlin, Heidelberg.

[7] D.Arul Suju; G.Meera Gandhi. An Automaton based Approach for forestalling Cross Site Scripting attacks in web application. Advanced Computing (ICoAC), 2015 Seventh International Conference on Dec. 2015

[8] Rui Wang; Xiaoqi Jia, Qinlei Li; Daojuan Zhang. Improved N-gram Approach for Cross-site Scripting Detection in Online Social Network. Science and Information Conference (SAI), 2015

[9] Michael Backes; Sebastian Gerling; Philipp von Styp-Rekowsky. A Local Cross-Site Scripting Attack against Android Phones. IJCSN International Journal of Computer Science and Network, Volume 3, Issue 3,June 2014

[10] Xing Jin, Xunchao Hu, Kailiang Ying, Wenliang Du, Heng Yin and Gautam Nagesh Peri . Code Injection Attacks on HTML5-based Mobile Apps: Characterization, Detection and Mitigation. Department of Electrical Engineering & Computer Science, Syracuse University, Syracuse, New York, USA

[11] Bhavani A B . Cross-site Scripting Attacks on Android WebView. IJCSN International Journal of Computer Science and Network, Vol 2, Issue 2, April 2013

[12] Zhang X., Du W. (2014) Attacks on Android Clipboard. In: Dietrich S. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2014. Lecture Notes in Computer Science, vol 8550.Springer,Cham.