An algorithm for Prevention and Detection of Cross Site Scripting Attacks

Aqsa Afroz[#1], Dr Mohsin Ali Memon[*2], Salahuddin Saddar[#3] , Muhammad Haris Khan[#4]

Software Engineer’s & Department of Software Engineering

Mehran UET, Jamshoro, Pakistan

Abstract:
Currently, we live in an era of information and communication technology (ICT) in which humans are globally connected with each other through Internet. With the advent of World Wide Web (WWW), Internet has enabled numerous useful applications for the benefit of people around the world. These include online shopping, e-learning, internet banking, social interactions, etc. However, security of web applications has always remain a major concern of its users in general and prevention from hacking attacks in particular. Although, an adversary might attack on web applications by exploiting several hacking techniques, but in recent years Cross-Site Scripting (XSS) and Cross-site Request Forgery (XSRF) attacks has got significant attention from the researchers. According to Open Web Application Security Project (OWASP), XSS attack is amongst the top ten web application vulnerabilities (Mahindrakar, 2014; Cross-site Scripting, 2015). XSS might result in several types of threats, such as phishing, pop-up flooding, session hijacking, etc. The focus of this research is analysis, detection and/or prevention of XSS attacks. In contrast to earlier work on XSS attacks, this research provides a solution that is browser compatible and web development language independent. And our approach will provide zero code modification of already running web applications, equally beneficial for providing prevention to legacy systems.

Keywords- *Cross Site Scripting, Algorithm ,Scripting Attacks, Vulnerabilities, Prevention and Detection, SQL Injection, Security Misconfiguration, Maliciuos Attacks, Broken Authentication and Session Management, Cross Site Request forgery*

I. Introduction

Internet has turned this world into a global village. The advent of World Wide Web further reduced the distance between service providers and consumers. Every day almost each of us uses some type of web application (i.e. e-shopping, e-banking etc) in some way that makes trust and security as the essential requirements of web applications. As most of the applications are not developed on complete measure of security and to harm such web application are much easy. If attacker targets famous web applications a lots of users will become victim and result will be horrible. For such pre developed web applications there should be an additional mechanism required for making use of such application secure of any kind of malicious code attacks.

According to Open Web Application Security Project (OWASP) latest report some web applications attacks are listed below

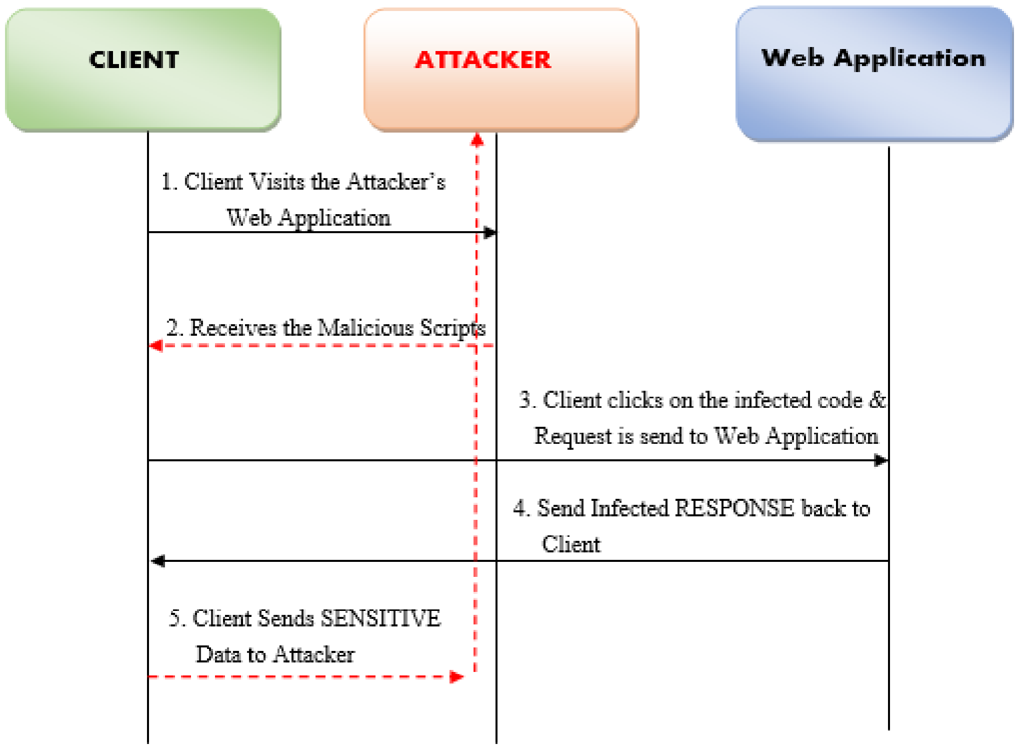
- Cross Site Scripting Attack (XSS)
- SQL injection
- Cross Site Request Forgery
- Malicious File Execution
- Insecure Direct Object Reference
- Security Misconfiguration
- Broken Authentication and Session Management

I precede my research with Cross Site Scripting Attacks, a kind of attack through which attacker injects malicious code into the web application which may harm to permissible user.

Cross-site scripting (XSS) is a hacking technique that exploits vulnerabilities in the code of a web application and steals sensitive data from the victim's web application. By using XSS technique, an adversary could easily insert malicious code (such as HTML, VBScript, ActiveX, JavaScript, etc) into a vulnerable dynamic page (Shar and Tan, 2012). As a result, the adversary might get sensitive or confidential information, steal cookies, create fake requests or execute malicious code on the victim’s system, which has several other severe consequences, such as redirecting the users towards a fake web page (phishing threat), changing the visual appearance of the victim's web page (defacing web site threat), adding fake information, e.g. "for more details, email to xyz@fake.com" (misinformation threat), adding web scripting language's code, which forwards victim's cookies to the adversary (session hijacking threat), etc (Nithya1 et al.,2015). In general, XSS attacks are classified into three categories: reflected, stored and document object model (DOM) based. Brief description of these attacks are given below

A. Reflected XSS

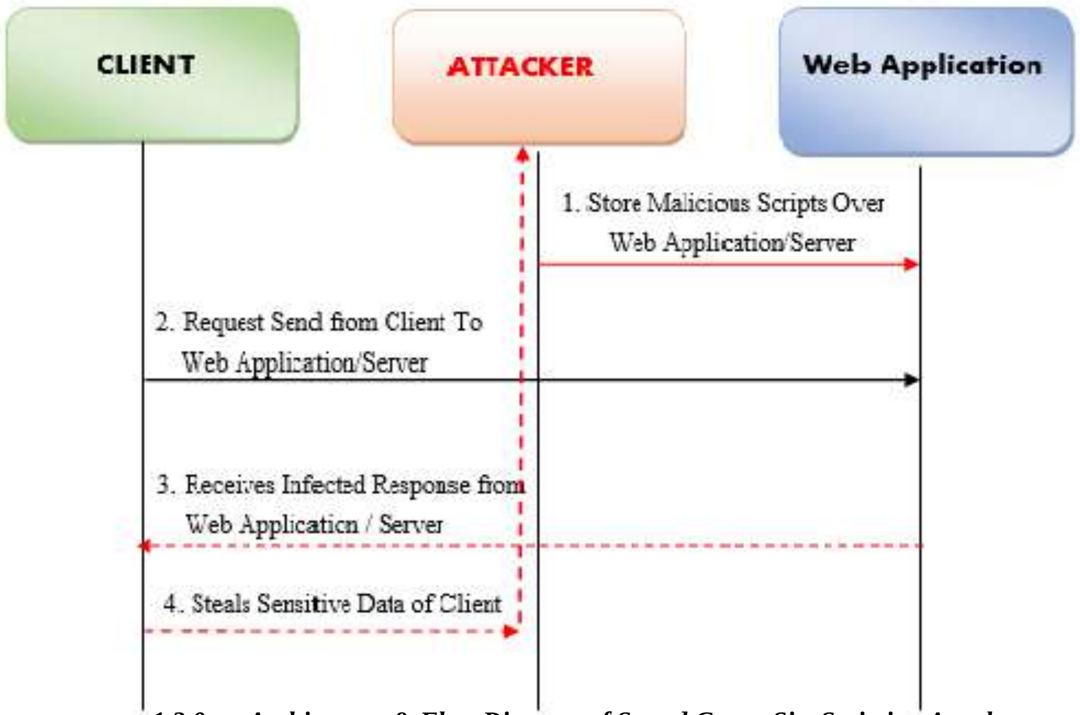
Also known as non-persistent cross site scripting attack, in this type the malicious code is non-permanently stored on web application itself but immediately reflect back to the victim of maliciously crafted link. Attacker do this attack by sending a link to the victim through email or other resource, when user clicks on that link the malicious script will run and become part of information which is send back to the user's web browser where this malicious code will executed. Given diagram shows the complete steps involve in this kind of XSS attack.


CLIENT
ATTACKER
Web Application
1. Client Visits the Attacker's Web Application
2. Receives the Malicious Scripts
3. Client clicks on the infected code & Request is send to Web Application
4. Send Infected RESPONSE back to Client
5. Client Sends SENSITIVE Data to Attacker

1.1.0 Architecture & Flow of Reflected Cross-Site Scripting Attack.

B. Stored XSS

In Stored XSS attack, an attacker inject malicious code into the web application permanently and this scripted code will be stored over the targeted server as html text. Stored XSS attack takes victims data, stores it in a database, a file or any other back end system and this data later on shows to the victim. If user visits the web page which contains XSS malicious code, this code will execute over victim's browser, which in return sends sensitive data of user to attacker. This type of attacking is extremely dangerous when we are talking about blogs, message posting on forums, CMS etc. where large number of comments are posted from other individuals. Stored Cross Site Scripting is also called as Persistent XSS attack. This type is much more dangerous than Reflected XSS vulnerability, because it constantly attack the users unless until administration of web application will remove it. Given figure shows the architecture of Stored XSS.

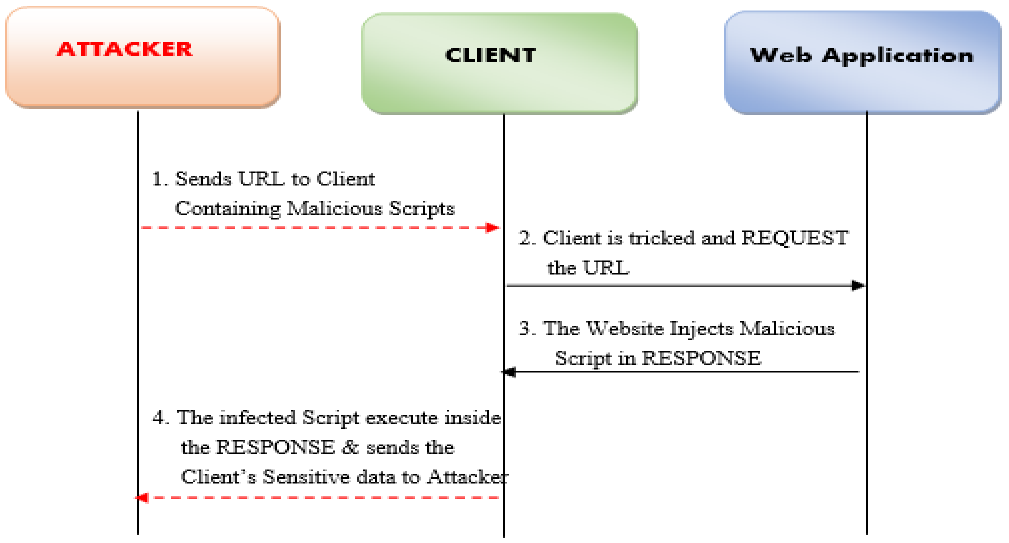

CLIENT
ATTACKER
Web Application
1. Store Malicious Scripts Over Web Application/Server
2. Request Send from Client To Web Application/Server
3. Receives Infected Response from Web Application / Server
4. Steals Sensitive Data of Client

1.2.0 Architecture & Flow Diagram of Stored Cross- Site Scripting Attack.

C. DOM Based XSS

DOM based is totally different from previous two types, DOM based attack is performed by modifying the DOM environment in client side rather than sending infected code to the sever. It allows the script to modify XML or HTML documents.

DOM based XSS will not inject malicious code into the web application. Like other types of XSS, DOM based XSS is also used to steal confidential user information or for hijacking user's account.


ATTACKER
CLIENT
Web Application
1. Sends URL to Client Containing Malicious Scripts
2. Client is tricked and REQUEST the URL
3. The Website Injects Malicious Script in RESPONSE
4. The infected Script execute inside the RESPONSE & sends the Client's Sensitive data to Attacker

1.3.0 Architecture & Flow Diagram of DOM-based Cross- Site Scripting Attack.

A reflected attack can be launched while a user from its browser tries to access a web application through re-directing the user's request to a fake web server. Whereas in stored XSS attack, the adversary stores malicious code on server in a message, comment field or in a database. Both reflected XSS and Stored XSS attacks are results of weak or without sanitization on server. In contrast, DOM-based XSS attack is directly launched on web browser in order to steal the sensitive data, such as cookies, passwords, etc. (Hayes and Offutt, 2006). Researcher have proposed many solutions (Shahriar and Zulkernine, 2009; Jim et at., 2007; Liu and Tan, 2009) to encounter XSS attacks, however they lack the support of cross browser compatibility as well as their proposed solutions are not web development language independent.

II. Background and Related Work

To prevent existing web applications from such attacks and to entertain users with security is currently a big issue which almost every application on internet is facing in these days. Many applications were introduced and many were implemented but none of them are much accurate to take guarantee of absolute level of providing security over web application. A major reason of this shortcoming is, for implementing these security shields one needs to modify their source code which is not an easy task for existing web applications. According to the latest survey of 2015 many approaches and provided solutions were discussed along with their advantages and disadvantages i.e. static analysis approach, dynamic analysis approach , client side security, server based security etc. are highlighted.

There were many solutions had been propose to provide XSS attacking detection and prevention. Some of them are discussed here. Scott D & Sharp R [10] proposed web proxy which worked between web application and user as a firewall, making sure that web application is stick to pre-described security policies. The main drawback of this approach is the creation of security policies over network and there management task is quite difficult and error- prone. Another tool was also developed by YW Huang [11] named as Webs SARI (Web Application Security By Static Analysis and Runtime Inspection). In his approach he used to combine runtime inspection and analysis of static code features to find vulnerabilities. He implemented intra procedural approach and lattice model for finding out vulnerabilities. The main drawback of his tool is that it provides large number of positive and negative results because of intra procedural technique based analysis. Webs SARI also took result from user's filter, hence there is a chance of missing the actual vulnerability, it is also possible that the actual malicious code may not be detected by the designed filter functionality. Webs SARI is language dependent i.e. only prove security to PHP based web applications.AppShield is a commercial product, a web application proxy firewall proposed and developed by Robert, Xuhua and Yueqiang [12]. This product claims that apparently it doesn't need security policies and it automatically detects web threats i.e. XSS attacks. One cannot verify this claim because the AppShield is a closed source. Furthermore it is a plugin which can perform only simple checks and can provide limited security. Wassermann and Z. Su used Untrusted List approach [13] in which a list of harmful scripts is used to detect malicious data. In this method provided data is checked from provided blacklist and if the result was positive the further action was taken. Searching and replacing of few characters is consider as weak practice and can be attacked easily. There are numerous ways for XSS attack which can easily bypass this blacklist validation. Another approach was used known as BEEP (Browser-Enforced Embedded Policies) approach, in which T. Jim, Swamy and Hicks introduced a white list of all scripts which was provided by web application to the browser so that it can protect web applications from XSS attack [8]. Its drawbacks are every browser has its own mechanism to parse data and secondly modification required in all web browsers. Since every user needs to have an updated version of browser on their machines. Many other projects like AppScan [14], WebInspect [15] and ScanDo [16] follow approach to identify errors in development cycle and they are unable to provide any prevention to running web application. Moving forward there is another tool named as Noxes. Noxes is a web based proxy provides protection against stealing of sensitive information from user's site to a third party site [17]. This is an application level firewall which monitors every connection coming or leaving the client machine, Noxes prompts user to decide whether the connection is allowed or blocked. Similar type of approach has been adopted in [10 and 18]. It is not enough to black list links in order to provide security from cross site scripting attack. Proxy based fails to find errors and requires watchful configuration. Moreover many other tools are also present as open source HDIV is one of them. HDIV provide security against XSS attack and protects web application from all

OWASP top 10 risks. HDIV offers no change of code but it requires configuration in XML files of the web application. Another drawback is, it works only with some specific frameworks of java, it has Spring dependency and it is unable to provide security to legacy systems.

TOOLS	ADVANTAGES	DISADVANTAGES
ModSecurity	Open source, easy to install apache module	Has no support for different encodings and filters are not effective
PHPids	Easily catches all injections of basic level, open source, codebase support.	PHP dependent, CPU consumption etc.
HDIV	Open source, eliminates or mitigates web security risks, easy to install	Works only with some specific frameworks of java, spring dependent, needs to modify source code i.e. modification in web.xml.
NOXES	Provide protection against cross site scripting attack	Closed source, client side solution working only on Microsoft windows, have to run on client's personal computer as a firewall.
WebSSARI	Provide protection against cross site scripting attacks	Only for PHP based web application. Gives large number of positive and negative analysis.
XSSFilter	Provide protection against cross site scripting attack, Language Independent, Deployment Environment independent, can run on any operating system.	Currently working only for Cross-Site Scripting Attacks. Remaining 9 of OWASP Vulnerabilities are remaining.

III. Need For XSS Filter Application/ Motivation / Contribution

In present days existing web applications are facing XSS attacks. Malicious code can easily be injected to harm web applications i.e. its vendor and users equally. The attacker sniffed the request and he tried to inject his malicious code inside web application, unfortunately because of security lack he successfully hijacked users authorized account and enjoying all access of victim's account. A victim can be any web application over the World Wide Web which is facing security issue and carrying sensitive data i.e. online banking, shopping carts, government websites etc. are common examples. Through XSS attack, attacker get control over dynamic content in HTTP response i.e. JavaScript, CSS, HTML etc. by injecting code like

"">< img src = 'i:i' onerror='alert(0)'>

"><iframe src='JavaScript: alert(0)'>

Some common mistakes which leads XSS attack are weak input validation, no security prevention results in account hijacking, cookie theft, changing of user's account settings. Currently many tools are present in market but all of them have some dependency or need code to modify and are not beneficial for legacy systems. One major reason of this shortcoming is lack of complete methodology for evaluation in term of performance or modification of source code is needed which is an overhead for existing systems. A system is required which will be easy to deploy and provide a better performance to detect and prevent web application from Cross Site scripting attacks.

By keeping in mind the need of time we proposed a XSS filter which totally works independently prior to the development language, by using which web application is designed and developed, have no tension on which environment the web application is running as well as there is no need of any code modification for binding XSSFilter which detects and prevent from XSS attacks. Proposed approach will not only detects harmful XSS attacks over web application but also remove such malicious code from the request before it hits the actual web application. This XSS Filter is equally beneficial for any language based web application or even for legacy systems too.

Present tools also provide security from XSS attacks but they are limited either to technology or environment. Some are browser specific or some required changing's on code level. Some are purchased and other needs typical installations for providing security against XSS attacks. In contrast with these tools, application and browser based plug-in XSS Filter over the web applications, such as shopping carts, online banking and other web

applications containing very sensitive data. With the help of proposed web tool vendors of the web applications will be able to add further security to their existing applications without changing the source code and runtime environment.
There are four main goals of the proposed tool/application which are usability, compatibility, independency and security as well. From usability we mean the tool is simple enough to use and easy to understand so that every developer can use it. By compatibility point of view, the application should be easily integrated with any language based web application i.e. JSF, PHP, VB.net etc. By independency means tool/application should not dependent on the programming language through which the targeted web application is developed, not thinking about the environment over which the application is running. From Security point of view, the XSS Filter tool will make sure that it is capable of establishing secure and separate session for each individual and also assure that all communication between web application and user; and between several entities of the web application must be safe and secure. The Major goal of XSS Filter is to provide security to existing web application without making any change in their code. In last we make sure that this application is light weight and providing 100% security to any web application of any programming language accessed by using any web browser deployed on any operating system.

XSS Filter acts as a web application and works on all possible patterns to mitigate credible Cross Site Scripting Attempts. XSS Filter effectively works against leakage of information from user's account, requiring no customization effort.

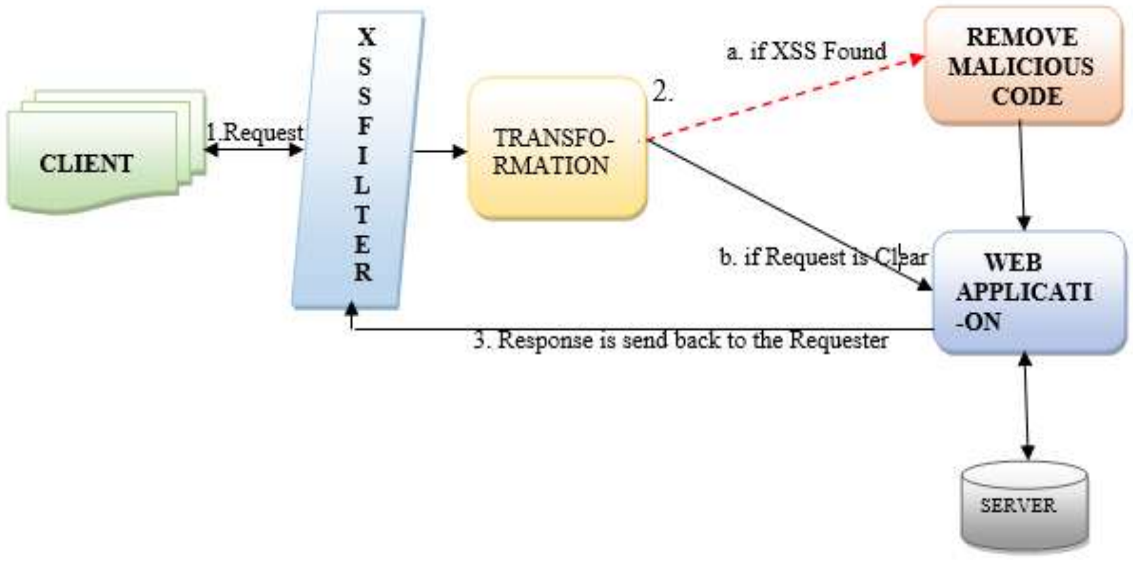
IV. Proposed Approach

Cross-site scripting (XSS) is a hacking technique that exploits vulnerabilities in the code of a web application and steals sensitive data from the victim's web application. By using XSS technique, an adversary could easily insert malicious code (such as HTML, VBScript, ActiveX, JavaScript, etc) into a vulnerable dynamic page (Shar and Tan, 2012). As a result, the adversary might get sensitive or confidential information, steal cookies, create fake requests or execute malicious code on the victim's system, which has several other severe consequences, such as redirecting the users towards a fake web page (phishing threat), changing the visual appearance of the victim's web page (defacing web site threat), adding fake information, e.g. "for more details, email to xyz@fake.com" (misinformation threat), adding web scripting language's code, which forwards victim's cookies to the adversary (session hijacking threat), etc (Nithya1 et al.,2015). In general, XSS attacks are classified into three categories: reflected, stored and document object model (DOM) based. A reflected attack can be launched while a user from its browser tries to access a web application through re-directing the user's request to a fake web server. Whereas in stored XSS attack, the adversary stores malicious code on server in a message, comment field or in a database. Both reflected XSS and Stored XSS attacks are results of weak or without sanitization on server. In contrast, DOM-based XSS attack is directly launched on web browser in order to steal the sensitive data, such as cookies, passwords, etc. (Hayes and Offutt, 2006). Researcher have proposed many solutions (Shahriar and Zulkernine, 2009; Jim et at., 2007; Liu and Tan, 2009) to encounter XSS attacks, however they lack the support of cross browser compatibility as well as their proposed solutions are not web development language independent. The main motto of our approach is to provide prevention from XSS malicious code attacking from client side over web application. We aim to detect every vulnerable content that may go through any input validation mechanism inside a web application. In order to achieve the desired results, and overcome the deficiencies of previously developed application/ tools, we'll develop a filter application that will work on the client side over the web in between user's request and web application in order to protect the web applications from XSS attacks.

Our approach is not only for finding Cross Site Scripting vulnerabilities due to unchecked or insufficiently check malicious data but also providing zero code modification facility. Below mention table show XSS attacks tricks.

User Input	XSS Attack
var name = “Smith”	var name=”Smith”; “<script> alert(‘evil’); </script>”
var msg= “hello Smith ”	var msg= “Hello \n <script> var i = new Image();i.src='http://attack.com ?msg='+document.cookie</script>”
alert(“HI Smith”);	String.fromCharCode(97,108, ……………….n);

The approach has two parts: (1) wrap a web application into XSS Filter so that any client’s request passes through it to get requested data and (2) if any malicious attack is found first transform the request before hitting the real web application, we remove malicious code through prelisted regular expression scheme of our own. Given figure shows the working of XSS Filter.



A. Working Diagram of XSS Filter

In contrast to prior work, the proposed solution will be web development language independent, deployment environment, cross browser compatible and needs no ZERO code modification. Our proposed solutions are equally beneficial for existing web applications as well as for legacy systems.

This research makes these main contributions.

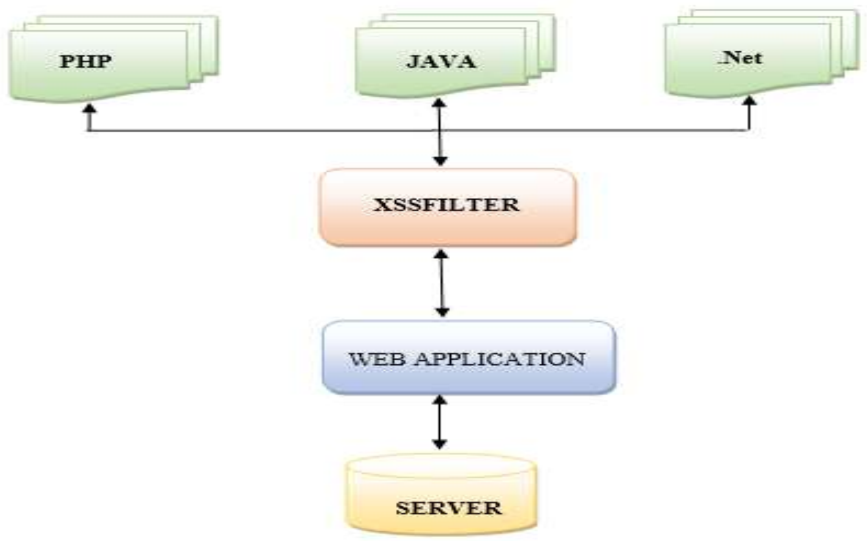
- Proposes an approach for detecting XSS attack cause because of weak input validation.
- Through this prevention of Legacy Systems becomes possible.
- It gives an actual running example of single application which provides security against XSS to any language based web application.

V. Implementation

To test our approach we implemented a system and named it as XSS Filter. Our approach is totally different from existing approaches. We work on transformation of request rather than on modification of existing web application.

We implement our approach as a JAVA based Web Application Framework names as XSS Filter which provides detection and prevention against Cross Site Scripting attacks. The flow of information between user and web application is controlled by XSS Filter. It extends the behavior of existing web applications with respect to security against XSS malicious code injection. This implies that we can use XSS Filter with web application developed by using any programming language in transparent way without any simple or complex addition to the web application. An illustration of XSS Filter’s architecture is presented in Fig 5.0.The XSS Filter consists of five phases; Request Receiver, URL Transformation, XSS Detector and XSS

Remover, Response to User. It works on front of web application. The request receiver receives user input as original web application is receiving and forward that request to URL Transformation Module. This phase after getting the request transform it according to the rule we made and getting its request parameters and enters into XSS Detector phase. In XSS detection phase the requesting parameters are processed through the XSS detection scheme, if any clue of XSS malicious code traced the request enters in to XSS Remover phase. This phase removes all the vulnerable data through our define custom methods, produces clean parameter towards transformation phase after that request is forwarded to the original web application which process this request and sends response to Response to User phase, this final phase is responsible for out to user and for the session management for each individual of the web application.



5.0 Architecture & Flow Diagram of XSS Filter.

VI. Evaluation

In order to evaluate the proposed system and for providing support to our approach we tested our XSS Filter over 4 web applications, 3 are open source and one is custom made (for testing purpose). Through this evaluation, we tried to get answers of given questions: Does our approach compromises over the required output of the web application? Is the approach really effective in removing malicious XSS scripts from web applications of real world? Does this XSS Filter web application provide security to legacy web application? Is ZERO code modification statement true?

Our setup for evaluation of our approach consisted of a machine (4GB RAM, dual core processor) using Windows Operating System. We deployed the actual web application and XSS Filter over two separate Tomcat 7 servers. Then we divide our experiments into two parts; discussed in detail below

A. Evaluation on Real World XSS Attacks

Our first objective was the evaluation of the effectiveness of our proposed approach XSS Filter against the real world XSS attacks. Since our XSS Filter targeted towards legacy web applications; we go through the CVE repository [19] and choose such famous applications that had reported in 2015. MOODLE, X-cartare taken from CEV report and source code is taken from their official web sites; EOBI and MVR were taken from departments on request.

Now we discuss these exploits and experience of

APPLICATION	VERSION	LANGUAGE	XSS ATTACK DESCRIPTION
EOBI	3.0	JEE	via form input fields
MOODLE	2.8.5	PHP	via input fields
X-cart	4.5	PHP	via arguments pass through URL
MVR	2.0.1	JEE	via HTTP argument in URL

evaluating XSS Filter against these application's vulnerabilities.

EOBI: This is an intranet web based application developed on JEE Struts1 framework. Stands for Employee Old Age Benefit Income. This application basically works for money saving as employee can get benefit after retirement especially in private sectors. This intranet web application has many forms and input fields. XSS code can be injected through these fields easily.

MOODLE*:* Stands for Modular Object Oriented Dynamic Learning Environment; focus on online teaching and learning of courses. Most of the universities and educational institutes use this open source application to interact with students. Unfortunately the input fields in MOODEL can be attacked through persistent XSS. We tested XSS attack on MOODEL by editing our profile and put some JavaScript code i.e. <script>alert("HELLO There"); </script>, web application accepts our input and displayed our injected alert message.

X-cart: It is an open source web application, used for online shopping. Code is developed on PHP. One can use code from this web site and customize it according to requirement, host it over purchase domain. Unluckily X-cart accepts XSS attack embedded in URL

i.e.xcart.com/product.php?pro='>&lt;script&gt;alert(document.cookie)&lt;/script&gt;

MVR*:* MVR stands for motor vehicle registration system. This is an intranet application used to register, transfer vehicles. This web application was developed on JEE Struts2 framework. We attack on this application by adding vulnerable code into request, send through URL and found that application is vulnerable.

Such typical XSS vulnerabilities are not checked by input validators and here is the need of some external safeguard like our XSS Filter. Proposed solution successfully provides defense against all 4 XSS exploits mention in above table. This evaluation shows that our XSS Filter can be user to provide protection to real world application successfully.

B. Comprehensive Evaluation

In this part we evaluate resilience of our XSSFilter, we select OWASP XSS Filter Evasion Cheat Sheet [21], a collection of numerous XSS attacking hacks. In our evaluation we focused on 20 XSS attacks, test using two different browsers namely Mozilla Firefox and Google Chrome. These vulnerabilities are classified into various categories few of which are mention below.

- **XSS Case Insensitive Attack** is the attack by adding case insensitive data to the JavaScript code to bypass XSS sensitive check e.g. < IMG src= JavaSCripT:alert('Attack')>. XSSFilter disallow it.
- **Grave-Accent Obfuscation** used both double and single quotes to encapsulate the malicious JavaScript's code. Most of the XSS filters do not provide prevention against grave accent. Our approach detects and removes this vulnerability from input code.
- **No Semicolon and No Quotes** is the trick to inject malicious code without adding any quote or semicolon to the script. XSS Filter also provides security against such kind of attack.
- **Using the JavaScript Directives** image XSS can be done. XSS Filter can fight against this vulnerability too.
- **Decimal HTML Character** is used to add vulnerability when there is no quote of any kind allowed. This can be done by calling String.fromCharCode ( ) method using script. Our solution works efficiently against this attack too.
- **Hexadecimal HTML Character** is a tricky kind of XSS attack. This attack is done by converting malicious JavaScript code into Hexadecimal format. Most of the filters assumes that there is a numeric value with a pound symbol. Our XSSFilter never miss such tricky attack of XSS, detect it and remove it from request generated by user.

We used our custom application to check all these vulnerabilities of Cross Site Scripting. After that we provide protection to our custom web application by wrapping it under XSSFilter, which is able to defend all. The successful safeguard against several attacks proves that XSSFilter is highly resilient.

VII. Conclusion

The developed application will work as session hijacking filter over the existing web applications like financial sites, web application for hospitals,shopping sites etc. Any change made to the filter will be independent of the web application and won't affect its architecture.

With the help of this software vendor of web applications can add security to their running applications without changing the existing code. This anti XSS Software will detect and prevent web applications from the attackers attack. Moreover this filtering software can be added to the newly developing website in order to provide security from session hijacking. This software will work independently for any web application, developed by using any programming language (PHP, JSP, ASP.net, JSF and etc.), and provide cross browser compatibility too

VIII. Future Work

Although XSSFilter is fully functional. At first we think to make XSSFilter tool available as purchased utility. But at the moment we provide it on demand with our custom support and integration support. Through XSSFilter we only covered CROSS Site Scripting vulnerability among top ten security concerns of OWASP. We are thinking to extend our XSSFilter Tool so that it covers all remaining 9 OWASP. We would like to provide these functionality as soon as possible.

Acknowledgment

First and foremost, praises and thanks to the ALLAH, the Almighty, for His showers of blessings throughout my research work to complete the research successfully. I would like to express my deep and sincere gratitude to my research supervisor, Dr Mohsin Ali Memon and co- supervisor, Salahuddin Saddar for giving me the opportunity to do research and providing invaluable guidance throughout this research. I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. I am very much thankful to my Husband Muhammad Haris Khan for his love, understanding, prayers and continuing support to complete this research work. Also I express my thanks to my sisters, brothers, and brother in law for their support and valuable prayers. My Special thanks goes to my friend's and my Project Manager Bushra Hasnat for the keen interest shown to complete this research successfully.

Finally, my thanks go to all the people who have supported me to complete the research work directly or indirectly.

References

[1] A. S. Christensen, A. Mooler and M. I. Schwartzbach, "*Precise analysis of string expression*", In proceedings of the 10th international static analysis symposium, LNCS, Springer-Verlag, vol. 2694, pp. 1-18. S

[2] Y. W Huang, F. Yu, C. Hang, C. H. Tsai, D. Lee and S. Y. Kuo, "*Verifying Web Application using Bounded Model Checking*," In Proceedings of the International Conference on Dependable Systems and Networks.

[3] Cross-site Scripting (XSS). https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[4] H. Liu, H.B.K. Tan (2009). "*Covering Code Behavior on Input Validation in Functional Testing*". Information & Software Technology. Vol. 51,No. (02)

[5] H. Shahriar, M. Zulkernine (2009). "*MUTEC: mutation-based testing of cross site scripting*". In Proceedings of the 5th International Workshop on Software Engineering for Secure Systems (SESS'09).

[6] I. Hydara, Abu Bakar Md. Sultan, Hazura Zulzalil, Novia Admodisastro (2015). "*Current State of research on cross-site scripting (XSS) – A systematic literature review*", International Journal of Information & Software Technology, Vol. 58, Pages: 170-186.

[7] J.H. Hayes, A.J. Offutt (2006). "*Input Validation Analysis and Testing*", Empirical Software Eng. Vol. 11 No. (04).

[8] L. Khin Shar, Hee Beng Kuan Tan (2012). "*Automated removal of cross site scripting vulnerabilities in web applications*". Journal of Information & Software Technology, Vol. 54, No. (5). Pages: 467-478.

[9] Manisha S. Mahindrakar (2014), *"Prevention to Cross-site Scripting Attacks: A Survey*". International Journal of Science and Research (IJSR), Vol. 3, Issue 7.

[10] T. Jim, N. Swamy, M. Hicks (2007). "*Defeating Script Injection Attacks with Browser Enforced Embedded Policies (BEEP)".* In Proceedings of the 16th International Conference on World Wide Web.

[11] V. Nithya1, S. LakshmanaPandian and C. Malarvizhi (2015). "*A Survey on Detection and Prevention of Cross-Site Scripting Attack*". International Journal of Security and Its Applications, Vol. 9, No. 3.

[12] David Scott and Richard Sharp (2002). "*Abstracting Application-Level Web Security*". In Proceeding the 11th International World Wide Web Conference.

[13] Yao-Wen Huang, Fang Yu, Christian Hang, Chung-Hung Tsai, D.T. Lee and Sy-Yen Kuo (2004). "*Securing Web Application Code by Static Analysis and Runtime Protection*". In Proceedings of the 13th International World Wide Web Conference.

[14] Yueqiang Cheng, Xuhua Ding, Robert H. Deng (2013). "*AppShield: Protecting Applications Against Untrusted Operating System*". School of Information Systems, Singapore Management University.

[15] G. Wassermann and Z. Su (2008). "*Static detection of cross-site Scripting vulnerabilities*". In Proceeding of the 30thInternational Conference on Software Engineering.

[16] Sanctum Inc. "*Web Application Security Testing—AppScan*" 3.5.http://www.sanctuminc.com

[17] SPI Dynamics (2003). "*Web Application Security Assessment*". SPI Dynamics Whitepaper.

[18] Kavado, Inc (2003). InterDo Version 3.0. Kavado Whitepaper.

[19] E. Kirda, C. Kruegel, G. Vigna and N. Jovanovic (2006). "*Noxes: A client-side solution for mitigating cross site scripting attacks*". In Proceedings of the 21stACM symposium on Applied computing, ACM pp. 330-337.

[20] N. Jovanovic, C. Kruegel and E. Kirda (2006). "*Pixy: A static analysis tool for detecting web application*

vulnerabilities (short paper)”. In IEEE Symposium on Security and Privacy, Oakland, CA.

[21] MITRE. Common Vulnerabilities and Exposure List. http://cve.mitre.org.

[22] Sourceforge, Open source websiteW.

[23] Monali Sachin Kawalkar, Dr. P. K. Butey "*An Approach for Detecting and Preventing SQL Injection and Cross Site Scripting Attacks using Query sanitization with regular expression*". International Journal of Computer Trends and Technology (IJCTT) V49(4), 2017.

[24] OWASP XSS Filter Evasion Cheat Sheet.

[25] https://www.owasp.org/index.php .