*Available online at*www.scholarsresearchlibrary.com

Scholars Research Library

**Archives of Applied Science Research, 2015, 7 (2):9-14**
*(http://scholarsresearchlibrary.com/archive.html)*

**ISSN 0975-508X**
**CODEN (USA) AASRC9**

# Preventing cross-site scripting attacks on the client side

**Anjaneyulu G. S. G. N.*, Shivani Singh[1] and Bhawna Kumari[2]**

*Department of Computer Application, SITE, VIT University, Vellore, India*

---

**ABSTRACT**

*Regular and frequent use of web applications is introducing different security issues. Cross –site scripting is one of them. Cross site scripting is an attack via browser side scripting language (generally JavaScript).Active sessions are frequently target of XSS attacks. It can be anything, from your facebook password to bank-account login session. Session is the mechanism used by web applications to track whether the logged in user is authenticated or not. In every session ,session cookie contains a string known as session token, which is stored in a cookie at user's browser and sent to the web server every time user make request. By means of these cookies only, user can be authenticated once logged in. if active session cookies are stolen by the attacker, he may perform any action as the user. The site will not be able to find difference easily. Attackers do so, by including malicious codes in application's output. In this research paper, we will be giving solution to secure this active sessions.*

**Key words:** active session, cookies, xss, malicious code

---

## INTRODUCTION

Many applications want to track the state of a particular user. For the security purpose, a user should be authenticated. Once a user logged in, in its account, he/she is authenticated for a particular session (till he/she logged out).This is the authentication of the user. Without the concept of sessions, user needs to be authenticated again and again on every request that he make. A session consists of two sections, a hash value and a session id. The role of session id is to determine the hash value uniquely for a particular session. The cookies, sent to the client's browser from the server includes the session id, and same way if a client make any request, browser at client side send cookies to the server. Mostly, web application provides user with authentication user id and password. These user_IDs are checked by the web application, and stored in hash value after log in, from now a session is active for the particular client. The session_id identifies the session, once a user logged in; it loads the requested page till the session is active.

It is clear now; a cookie serves as temporary authentication for a web application. The active sessions are the threat to user's confidential information if stolen. Following are the techniques by virtue of which, cookies can be hijacked:-

~Sniffing of the cookies in an insecure network, (like, in an unencrypted LAN, it is easy to track the traffic of all clients connected to the network).
~Cross-site scripting attacks at client side, aims to steal the cookies at user's browser.
In our proposed algorithm we will be using hard coded physical address (MAC address) of the device from which user will be login in encrypted form.

**2. RELATED WORK AND EXISTING SYSTEM**

2.1 The solution on the client side as well as the server side affects the performance as the solution on the server side degrades the web application and are often unreliable. Also the solution on client side result with the inefficient web browsing with the need for a client solution which do not affect the performance the proposed model introduces us with a solution which provides security with optimal web browsing. This includes three step process

2.1.1. Attackers Abilities- With an assumption that the attacker has following abilities
a. Attacker has his own site.
b. User visit attackers website.
c. The target website is letting the attacker to inject his code into the entity body of one of its http response.

2.1.2 Vulnerability coverage-the client side should prevent against all the vulnerabilities However with infeasible implementation focus is laid upon a certain vulnerable class or stream. Example-only rejected vulnerability where byte code appears in http request retrieve the resource.

2.1.3. Attackers goal-The goal of attacker is using user's browser run the arbitrary script with the target website. The attacker can always run the script whenever it can induce user to take arbitrary action. The paper considers the attackers with an intension to achieve their goal with single click interaction or zero interaction.

These are the Microsoft windows based personal firewall application which act as a background service on the desktop of the user .It is inspired by the windows firewall used today in most of our pc's and tabs. A fine control over the incoming connections that our local machine is receiving and the outgoing connections that the currently running applications are making is provided by the personal firewall. These blocks and detects malwares like worms, spywares etc and also prevents from the remotely exploitable vulnerabilities.The personal firewall prompts the user for the action whenever a connection request is detected and is not matched with the firewall rules .then this is upon the user to either block the connection, allow or create a permanent rule that specifies what should be done if such a request is detected again in future .However the personal firewall provides with the protection against a number of threats, but are inefficient in case of the client side attacks, like xss attacks. The reason behind is that the personal firewall will allow the browser of the user to make outgoing connection to any IP address with the destination port.

Noxes introduces an additional layer for the protection which the existing personal firewall do not provides .The idea behind is to allow he user to exert control over the connection that the browser is making just as personal firewall allows user to have a control over the connection that are being received or originated by the running processes on local machines. Noxes operates much like a web proxy fetching request on the behalf of the browser .Hence all web connections of the browser pass through noxes and can either be blocked or allowed depending upon the currently existing security policies.Unlike personal firewall noxes allows user to create filter rules (also can be called as firewall rules) for web results .There are three ways o creating these firewall noxes.

2.2. Manual creation-As the name itself suggests we manually can add rules in the rules database manually with the possibilities of including wildcards and can also choose to permit or deny request matching the rules.

2.2.1 Firewall prompts-with an interactive feature we can create a rule whenever connection request is made that does not matches any existing rule like provided in personal firewall .The user can also specifies if the rule could be permanent or temporary of the current browsing session preventing the base rule to grow too large.

2.2.2 Snapshot Mode-The user can use the snapshot mode that is integrated into noxes to create a "Browsing Profile" and to automatically generate a set of permit rules.First the snapshot mode is activated by the user and then surfing. After it start noxes tracks and collects the domains visited by the browser .Then user can automatically set the permanent filter rules based on the list of domains collected during a specific session .They can be reset by the user. A personal firewall, in theory helps to lessen the xss attacks as the attack from now will not be able to send the sensitive information to the server under his control without informing the user.

One of the security problem faced today is of xss attacks (i.e.)the cross site scripting in which the attacker encrypts a malicious script into the output of the application that is to be sent to the users web browser .the solution presented by our paper stops the xss attack on the browser client side by tracking the sensitive information being used in the

java engine of the web browser .if there is any transfer of sensitive information to take place then it prompts the user by asking to either allow or not for the transfer to take place .the existing solution lessens the performance of the client system resulting in the poor web surfing this solution provides a step by step approach without effecting the web browsing speed.

The solution allows the malicious script to do everything in context of the web browser (Mozilla firefox) our system tracks if any processing of sensitive information taking place and it sends this information to the user and asks for the transfer to take place. Then its up to user to either allow it or not with every access of sensitive information (i.e., cookies) results in the mark data in the java script program the processing of the marked sensitive information is tracked. The below table represents table of browser sensitive data objects with key properties ,these objects are DOM with its help the critical information can be accessible .all such objects contain the sensitive information and should therefore b marked as sensitive . In our proposed solution browser elements are treated as sensitive in DOM structure and there corresponding elements. All the marked elements are not equally sensitive. The most important is cookies

It has been implemented as the prototype version of noxes as windos.net application in c# .The application has small footprint and consisting of about 5,400 lines of code. The .NET platform for implementation being used as an important proportion of interrupt user surf under the windows as the conceptual and library similarity exists between the java and c#, the code is expected to be portable to java without any difficulty.

JSP,PHP ,ASP, etc are some of the languages which are used for the development of the web applications developed for increasing the customer's base .It was found that there is a need for a solution that could aim to facilitate with the independent services with derived interfaces that can be called to perform their task in a standard way, without having the pre knowledge about the calling application and without the application having or needing knowledge of how the services actually performs its task.We know that the web application is built for various purposes and each with different requirements for performance, security mechanism, internationalization and scalability to serve its customers.

Here in this project much of the stress was laid upon a specific case of xss attack against the security of the web application in browser side. This attack relays on the injection of the malicious code into the web application .If it succeeds then he could by pass.The proposed solution introduces a personal web firewall that helps to mitigate the cross site scripting attack .The main advantage with this solution is that it provides us with a client side solution .That provides the cross site scripting protection effectively without depending upon the application providers.

The client side solution supports a cross site scripting mitigation mode that significantly reduces the number of connection alert prompts that appears .This paper proposed the solution that limits the amount of information can be stolen by any single cross site scripting attack.

**3. PROPOSED SYSTEM - ALGORITHM**

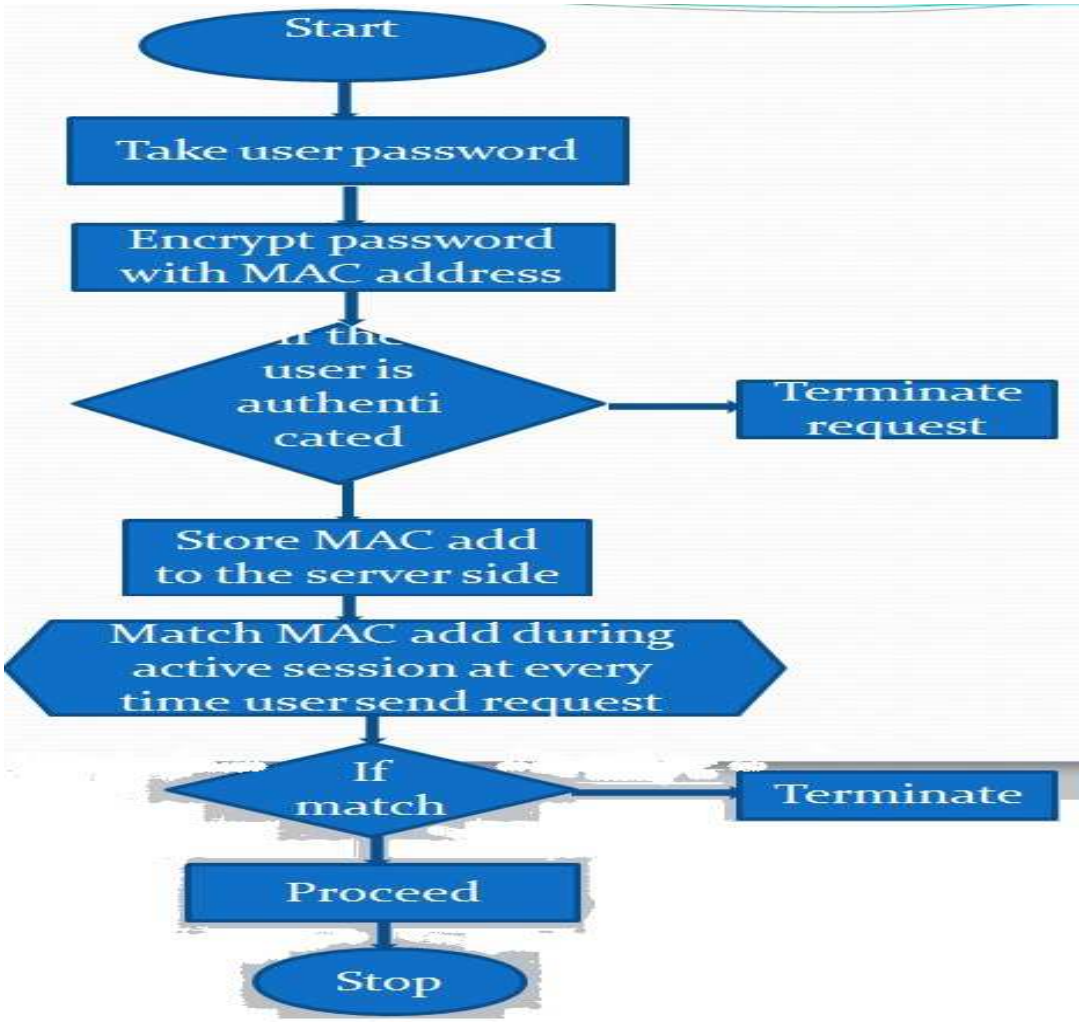
In our proposed solution, we are intend to use a MAC address in encrypted form so that session during which user is active on a particular system can be saved. We first take password from the user and authenticate him/her. Now attach encrypted MAC address with the password and store the MAC address on the server machine. While making connection to the client machine server tally the encrypted MAC address every time during one active session. If a hacker tries to hack the active session , server machine can figure it out because MAC address of his machine will differ the MAC address of the user's machine. Because every time user requests the server checks for his MAC address and matches it from the stored MAC address. Now once user logged out, the stored MAC address during last active session will be dropped, so that if the user wants to login from different device , he will be able to do so.

Start
-Take password from user and authenticate it.
-Encrypt MAC address and attach it with the password
-Store MAC address to the server
-during active session:

Repeat {
For every request{

If(MAC add of device==stored MAC add)
Entertain request
Else
Terminate connection
}
} till connection terminate or user logged out

**4. FLOW CHART OF THE PROPOSED ALGORITHM**




**5. IMPLEMENTATION**

We can implement the functionality of the algorithm by using visual studio. By running it on local host machine. We can use any of the provided language in visual studio for the implementation. Or we can actually implement by using client server architecture by making one machine as a client and another as a server. Future Scope is that we can propose individual algorithms for encrypting MAC address at client machine and store this MAC address at server machine in decrypted form. We can propose the working model of the algorithm and demonstrate it functionality.

**CONCLUSION**

We are currently having technologies like noxes in which filtering of data takes place, mitigation in which code replacement takes place etc., but when it comes to parallel processing of these techniques are somehow not up to the

mark. Where as, we have proposed an ideology by virtue of which we can actually have a watch on whether the is made by user or not. The idea of involving physical address of the device during connection to the server machine reduce the threat of hacking of active session.

**Figure 2: Existing Structure**

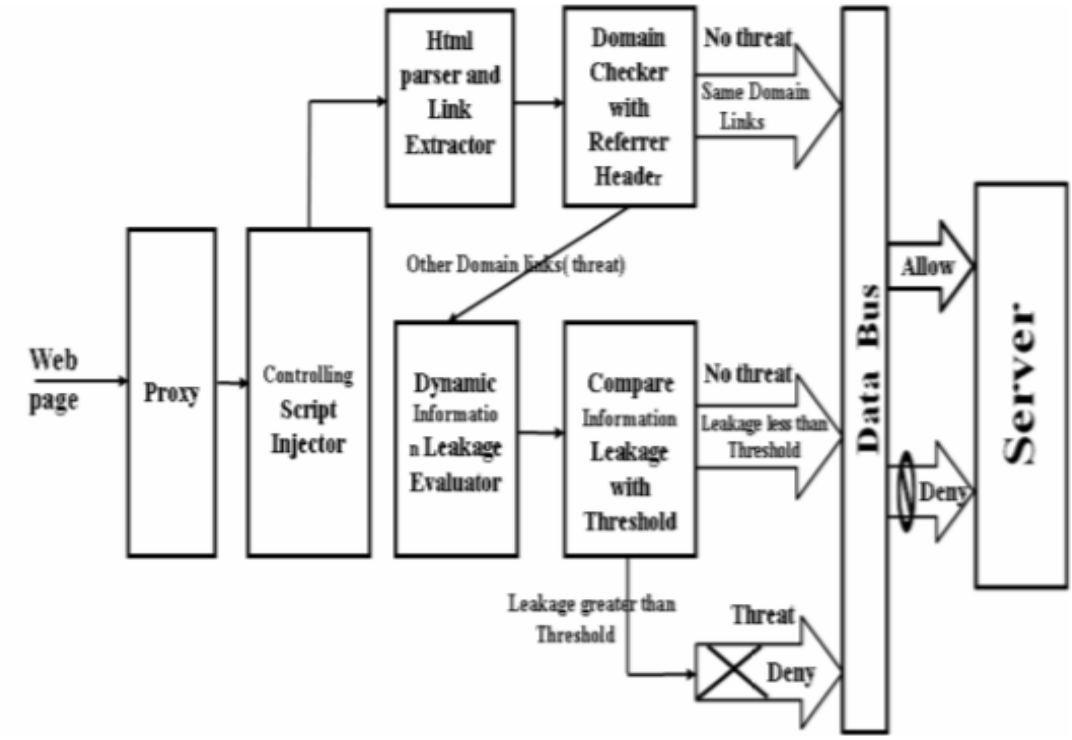



Table 1.1Browser Sensitive Data Objects with Key Properties

| Object | Properties |
|---|---|
| Document | cookie, domain, forms, lastModified, links, referrer, title, URL |
| Form | Action |
| Any form Input element | Checked default Checked, default Value, name, selectedIndex, toString, value, History current, next, previous |
| History | current, next, previous, toString |
| Select option | defaultSelected, selected, text, value |
| Location and Link | hash, host, hostname, href, pathname, port,protocol, search, toString |
| Window | defaultStatus, status |

**Abbreviations and Acronyms:** XSS (Cross Site Scripting) , LAN (Local Area Network) , MAC address(Media Access Control address)

## REFERENCES

[1] http://www.ijirset.com/upload/**2014**/april/77_ACritical.pdf
[2] http://www.cse.iitb.ac.in/~ritubala/Analysis_of_XSS_attack_Mitigation_techniques.pdf
[3] http://www.cs.uic.edu/~venkat/research/papers/blueprint-oakland09.pdf
[4] http://homes.cs.washington.edu/~mernst/pubs/create-attacks-tr054.pdf
[5] http://www.iosrjournals.org/iosr-jce/papers/Vol12-issue5/D01251923.pdf
[6] http://www-bcf.usc.edu/~halfond/papers/halfond11stvr.pdf
[7] http://ijcsi.org/papers/IJCSI-8-4-1-650-654.pdf
[8] http://www.ijpret.com/publishedarticle/**2014**/3/IJPRET%20-%20CSE%201491.pdf
[9] https://www.ccsl.carleton.ca/people/theses/Raman_Master_Thesis_08.pdf
[10] http://www.ijarcsse.com/docs/papers/Volume_4/3_**March2014**/V4I3-0644.pdf