ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

Prevention of CSRF and XSS Security Attacks in Web Based Applications

D.Kavitha[1], M.R.Akshaya[2], M.Karthick[3], K.Baghya[4], K.Gomathi Raja Eswari[5]

Assistant Professor, Department of CSE,Valliammai Engineering College, Potheri, TN, India[1]

UG Student, Department of CSE, Valliammai Engineering College, Potheri, TN, India[2,3,4,5]

ABSTRACT:CSRF (Cross Site Request Forgery) attacks targets an end user to execute malicious action on a web application and it is currently authenticated. It mainly focuses on target state changing request not theft on data. The changing requests like transferring of fund, changing email id, etc. The CSRF attack inherits the privilege of the victim to perform an undesired function on behalf of the victim. The prevention measures like using a secret cookie or only accepting POST request should not prevent the attack. The XSS attack injects malicious script in trusted websites whereas it occurs when an attacker uses a website to send malicious code generally in the form of browser side scripting. The HTML code filtering technique filters only the scripting tags and no the special characters. The proposed work focuses on preventing CSRF and XSS attacks. CSRF attacks are prevented by generating multiple fresh tokens to various sensitive actions in a session. The generated token id is unique so that the actions are prevented. The security of the token is incorporated by means of using hash function through MD5 algorithm. The XSS attack is prevented by filtering both HTML tags and special characters so that the attack cannot include malicious tag in a trusted web application.

KEYWORDS:CSRF attack, XSS attack, security attack, vulnerabilities, filtering.

I. INTRODUCTION

In today's modern world, everyone uses internet. Thus, it is important to have secured access over the web applications [3]. As there exist a number of attacks over web applications, we are proposing a system for preventing two types of attacks, namely CSRF (Cross Site Request Forgery) and XSS (Cross Site Scripting)[12], [2].
CSRF is a relatively common problem in web applications. CSRF attack takes place in a vulnerable web application [9], [4]. CSRF is also known as XSRF or hostile linking or session riding [11] or one click attack [6]. CSRF attacks occur when a malicious web site causes a user's web browser to perform an unwanted action on a trusted site. There are many existing defence plots to prevent or block CSRF attacks. In many previous papers, they adopt the scheme of CSRF tokens to prevent CSRF attacks. By adding those tokens to that request, the defence system can check whether each page request contains a valid token. If the attacker has or can guess victim's tokens, the web server will pass attacker's forgery requests and then tokens become meaningless. In this paper, multiple tokens are used in order to prevent the attacker from guessing the victim's token. Additionally we add a feature to the server to verify the MAC address of the victim. Even if the attackers guess the pattern of the tokens, he will not be able to perform the attack because the MAC address mismatches. XSS attack is similar to SQL injection attack [12], [4]. XSS attacks have become very common now a day, due to unsecured code written in PHP web applications [8], [7], [2]. This attack is performed by entering a malicious code in the text box of a web page. Such malicious code provides the victim's secret information to the attacker. In previous papers, these attacks were prevented by filtering techniques [5]. But still we don't have proper filtration on XSS attack. In this paper we are introducing an advanced filtration technique.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

II. RELATED WORK

Year	Title of the paper	Author	Techniques used	Advantages
2014	By passing XSS auditor: taking advantage of badly written PHP code	Anastasio Stasinopoulos, et.al	Proper code writing rules to build secure web along with XSS auditor	XSS auditor can easily identify which parts of the response are being treated as script
2013	Light-weight CSRF protection by labeling user created contents	Yin-chang sung, et.al	Content box usage in order to distinguish malicious request	Build in method and build in properties to reduce the computation overhead
2011	A study of the effectiveness of CSRF guard	Bayan chen, et.al	CSRF guard to protect CSRF vulnerabilities	For the use of URL for sensitive data transaction, it injects a token to protect the resources
2009	An HTTP extension for secure transfer of confidential data	Masaru takesue, et.al	Secured authentication mechanism in browser while submitting a request	Not only security but also performance overhead deploy ability
2009	Thread modeling fort CSRF attacks	Ron Ruhl, et.al	New CSRF attack pre-modeling for protection features in realistic web applications	Used for exploring, understanding, validating security protection features in realistic web application
2006	Preventing CSRF attacks	Nenad jovanovic, et.al	Protection against XSRF attacks using server side proxy	Minimal manual effort required to implement

III. SYSTEM OVERALL ARCHITECTURE:

3.1. CSRF ARCHITECTURE

Figure 1 describes that if the user sends the web request to the server through browser. The server will generate the token and the token is encrypted. This encrypted token and the response from the server sends back to the user. Using that token the user will continue the application.

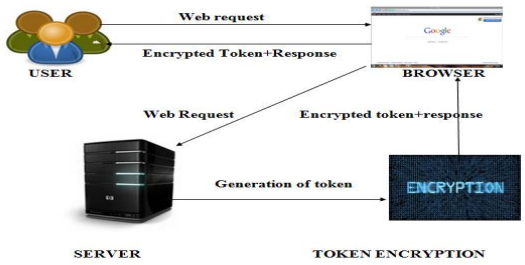

FIG 1: OVERALL ARCHITECTURE FOR PREVENTING CSRF ATTACK

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

3.2. XSS ARCHITECTURE:

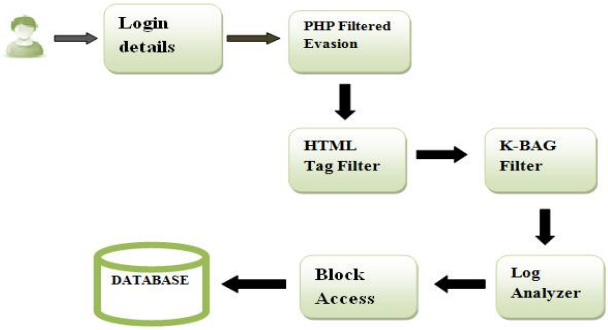

Fig 2: Overall architecture for preventing XSS attack.

Figure 2 describes that when the user enters the login details then the system start checking the details given by the user for tags and special characters. At once input passes through K-BAG filter, the entered text will be free from malicious code.

IV. SYSTEM DESCRIPTION AND DESIGN

4.1. USER LOGIN TO ACCESS SERVER

HTML is mark-up language use to create the web page. HTML uses HTTP protocol. HTTP protocol is a REQUEST-RESPONSE protocol that is used to communicate with hosts. This protocol is a stateless protocol that is it treats each request as an independent transaction. When the user access the application through browser, the request from the user is send to the server in the form of HTTP request. A typical format for HTTP request will be get/URL HTTP/1.1. The get method used when the browser request for a page. The URL (Uniform Resource Locator) specifies this request is for this particular URL. HTTP/1.1 specifies the version of the HTTP protocol. After receiving the request from the user, the server will verify whether the user has the rights to access the page. If the user has those rights then the server will respond the user in form of HTTP response[10]. A typical format for a HTTP response will be HTTP/1.1 200 OK. This format specifies that request from the user accepted successfully. The code 200 specifies success. If the format contains 4XX and 5XX then it means client error and server error respectively. If the users have not those rights to access the page then the server will not send the respond to the user. We are going to create the login page using HTML. The HTML contains pre-defined tags. The HTML documents begins with <!DOCTYPE html> to specify the type declaration. Then the HTML documents starts with <HTML> and ends with </HTML>. Inside the HTML tag mainly we have two tags namely head and body tag. The head tag is used to specify the title of the page. The body tag is used to specify the visible part of the page. A basic syntax for HTML is

```
<!DOCTYPE html>
<html>
<head> …….. </head>
<body> …….. </body>
</html>
```

In order to get an input from the user, the HTML forms are used. The <form> tag is used in HTML forms. The <form> tag contains many types of input elements. A basic syntax for HTML form is

```
<form>
<input type=” text” name=”username”>
</form>
```

The login button is created as follows

```
<input type=”submit” value=”login”>
```

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

When the user click the login button, the request from the user is sent to the server through HTTP protocol.

Fig 3: Request for a web page

Figure 3 describes that if the user wants to access the server, then the user will send the web request to the server through the browser.

4.2. TOKEN GENERATION FROM SERVER WITH ENCRYPTION

At once the user sends a request to get a web page in the form of HTTP request packets, the server generates a Token[1][6]. A token is nothing but a small 8-bit value generated for specific user identification. Token is generated in server which is encrypted with values using MD5 algorithm, where MD5 is of 128-bit encryption algorithm. Now a day, there exist more secured algorithms than MD5 like SHA, etc. But usage MD5 is better because other algorithms uses more effective security algorithms which takes more time than MD5, thus MD5 takes less time such that it creates session timeout when attacker tries to change or alter the packets. MD5 uses hash function to generate a random hash value to encrypt the token, where hash function converts a variable size data to a fixed size. In this system, tokens are generated for every single access of the web page with encrypted format. Hence the attacker gets confused while capturing the packets of the user which have different encrypted token values.

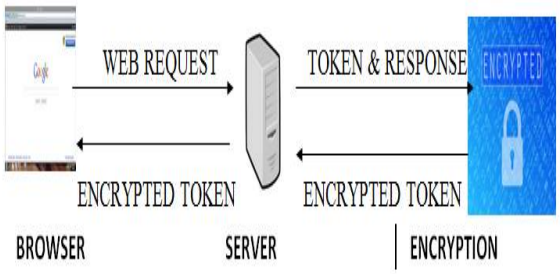

Fig 4: Generation of encrypted token

Figure 4 describes that if the user sends the web request to the server through browser, then the server will send response and encrypted token.

4.3. RESPONSE FROM SERVER TO CONTINUE APPLICATION

After the encryption of the token, the encrypted token along with the response is sent back to the server. Then the user will send the next request along with the encrypted token in the form of cookies to the server. Every time while using the server, it generates new encrypted tokens. The server will check the token to verify whether the user is authenticated or not. If the token is mismatched then the web access will be denied.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

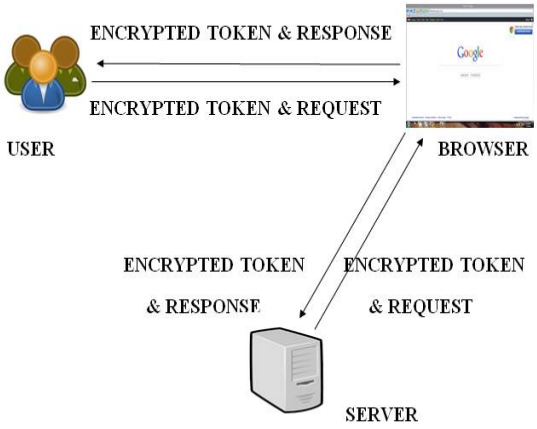

Fig 5: Flow of request response packets

Figure 5 describes that if once the user got encrypted token and response from the server, then the user will continue the application by sending web request along with that encrypted token.

4.4. FILTERING OF TAGS AND SPECIAL CHARACTERS:

XSS attack is performed by injecting malicious code in the text box of the web page. To prevent this attack we are going to filter the tags and special characters. The input given by the user must verify before it reaches the server. First the input given by the user must passes through the HTML tag filter which is used to filter the HTML tags. For example: If the user tries to inject the malicious code like <script>alert (1) </script> which enter into the HTML tag filter, it filters the script tags. So the output from the HTML tag filter is alert (1). In order to remove the special characters, the alert (1) must pass through K-BAG filter which is used to filter the following characters like (,), @, &, : etc. Final output from the K-BAG filter contains no HTML tags and special characters. Then the output must pass through pattern matching which verifies whether the input is in the correct pattern or not. If the pattern is mismatch then the access is blocked. Otherwise the access will be provided.

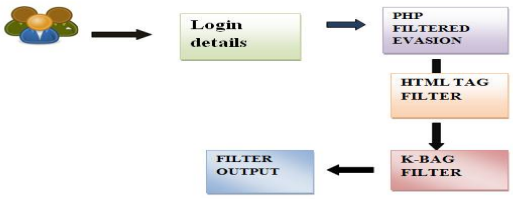

Fig 6: Script tag filter

Figure6 describes that the input given by the user will pass through the filters in order to filter the malicious code.

4.5.VERIFICATION THROUGH LOG ANALYZER

After the output from the filter, we have to verify whether the output is correct or not. This verification is done through log analyzer. Database contains the pattern for all types of input so the log analyzer compares the input and database. If the input is in wrong pattern then the access is denied.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 3, March 2016

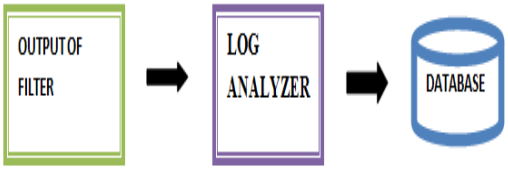

Fig 7: Verification of tags

Figure 7 describes that the output of the filter is verified through the log analyzer.

V. CONCLUSION

In this paper we discussed about the web attacks like CSRF and XSS security attack whereas the background of those attacks are discussed. The proposed model focuses on generating unique tokens for the session states and it is secured by using encrypting algorithm. Anew feature is added to the server focusing on MAC address and it is protected by means of providing access control rights to the owner, user and groups. The proposed security model focuses on XSS attack whereas it filters both special characters and tags in order to prevent the injection of malicious code to the program.

REFERENCES

[1] Light-weight CSRF protection by labeling user-created contents by Yin- chung sung, in 2013 7th International conference on software security and reliability.
[2] Bypassing XSS Auditor: taking advantage of badly written PHP code by Anastasios Stasinopoulos in 2014 IEEE transaction.
[3] Protecting websites from attack with secure delivery networks by David Gillman in 2015.
[4] Analysis of field date on web security vulnerabilities by jose fonseca in IEEE transaction on dependable and secure computing vol 11 NO:2 March/April 2014.
[5] A study of XSS worm propagation and detection mechanisms in online social networks by mohammad reza faghani in November 2013 IEEE.
[6] A study of effectiveness of CSRF Guard by Boyan Chen in 2013 IEEE.
[7] Defending against cross site scripting attacks by Lwin Khin Shar in march 2012 IEEE.
[8] Security vulnerabilities in the same- origin policy: Implications and alternatives by Hossein Saiedian in September 2011 IEEE.
[9] Threat modeling for CSRF attacks by Xiaoli Lin in 2009 IEEE.
[10] An HTTP Extension for secure transfer of confidential data by Masaru Takesue in 2009 IEEE.
[11] Preventing cross site request forgery attacks by Nenad Jovanovic in 2006 IEEE.
[12] New threats and attacks on the world wide web by Thorsten Holz in 2006 IEEE.