International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-2 Number-3 Issue-5 September-2012

Prevention of Cross-Site Scripting Vulnerabilities using Dynamic Hash Generation Technique on the Server Side

Shashank Gupta[1], Lalitsen Sharma[2], Manu Gupta[3], Simi Gupta[4]
Lecturer in Department of Information Technology, MIET, Jammu (J&K)[1,4]
Associate Professor in Department of Computer Science and IT, Jammu University (J&K)[2]
Assistant Professor in Department of Information Technology, MIET, Jammu (J&K)[3]

Abstract

Cookies are a means to provide stateful communication over the HTTP. In the World Wide Web (WWW), once the user using web browser has been successfully authenticated by the web server of the web application, then the web server will generate and transfer the cookie to the web browser. Now each time, if the user again wants to send a request to the web server as a part of the active connection, the user has to include the corresponding cookie in its request, so that the web server associates the cookie to the corresponding user. Cookies are the mechanisms that maintain an authentication state between the user and web application. Therefore cookies are the possible targets for the attackers. Cross Site Scripting (XSS) attack is one of such attacks against the web applications in which a user has to compromise its browser's resources (e.g. cookies etc.). In this paper, a novel technique called Dynamic Hash Generation Technique is introduced whose aim is to make cookies worthless for the attackers. This technique is implemented on the server side whose main task is to generate a hash of the value of name attribute in the cookie and send this hash value to the web browser. With this technique, the hash value of name attribute in the cookie which is stored on the browser's database is not valid for the attackers to exploit the vulnerabilities of XSS attacks.

Keywords

Cookies, HTTP, Cross-Site Scripting Attacks, Hash function.

1. Introduction

Normally, users through web browsers request the resources from the web server of the web application, and the web server respond with the resources through HTTP protocol [1] in which no sessions are retained. Therefore, web applications generally use cookies to provide a mechanism for creating stateful HTTP sessions. Cookies are often used to store the session ids [2] for the web applications that require authentication. Since the cookies can both identify and authenticate the users [3], this makes the cookies a very interesting target for the attackers. Now-a-days, Cross-Site Scripting (XSS) attack is a common vulnerability which is being exploited in modern web applications through the injection of advanced HTML tags and Java Script functions. A weak input validation on the web application causes the stealing of cookies from the web browser's database. In many cases, the attacker who can obtain the valid cookies from XSS attack can directly hijack the user's session.

Cross-Site Scripting attack continuously leads the most wide spread web application vulnerabilities lists (e.g. OWASP [4] etc.). XSS are broadly classified into two main attacks which are Persistent and Non-Persistent Attacks [5] [6]. 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). The rest of the paper is organized as follows. Section II discusses the area related to proposed technique: Background of cookies, architecture of exploiting the XSS attack and recent work related to exploitation, detection and prevention of XSS attacks. Section III discusses our proposed technique. Section IV discusses the corresponding results and analysis part of our proposed technique. Finally we conclude and brief the future work in section V.

2. Background and Related Work

Cookies

The cookies are generally used to store the session IDs or personal sensitive information in the web applications. The cookies are sent by the web applications as a part of the response message using Set-Cookie or SetCookie2 header. The Set-Cookie header is used by the version 0 cookies and Set Cookie2 header is used by the version 1 cookies. The web browser stores the cookies in its repository and includes the cookies with every subsequent request to the web application. In general, there are two different versions of cookie specifications in use [1]: Version 0 cookie (Netscape Cookie), and Version 1 cookie (RFC 2965).

The version 0 cookie is the most widely used version. In such type of cookies, the cookies are identified by the combination of following attributes: Name, Domain and Path. The web server can use an arbitrary string as the value of name attribute. The domain and path attributes inform the web browser that the cookie must be sent back to the server when requesting URL of a given domain and path.

The version 1 cookie is an extended version of version 0 cookie. In addition to identifying the cookies by name, domain and path attributes as in the version 0, the version 1 adds an additional ability to identify the cookie by the port attribute as well. In this type of cookie, the web server must set the cookie using Set-Cookie2 header instead of using Set-Cookie header. The browser still returns the cookie using the Cookie header as the version 0 header but uses a different format [1]. Almost all the modern browsers do not support the version 1 cookie except opera browser [7] [8].

So, in this paper, we have performed the experiments on modern web browsers which support version 0 cookies except opera browser.

Architecture of Exploiting the XSS Vulnerability

XSS attacks are those attacks against the web applications which is often used to steal the cookies from a web browser's database. The following figure 1 is an architecture which shows the sequence of steps of exploiting the XSS vulnerability by a malicious attacker.

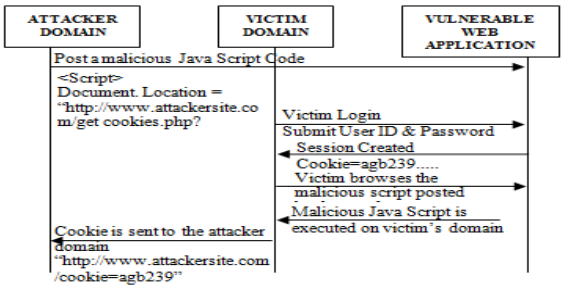

Figure 1: Architecture of Exploiting the XSS Vulnerability

The above architecture contains three useful commodities i.e. Attacker Domain, Victim Domain and Vulnerable Web Application. Here are some sequences of steps which will explain the above architecture of exploiting the XSS attack:-

- Firstly the attacker has found that the corresponding web application is vulnerable to Cross-Site Scripting attack. After this, attacker will post a malicious Java Script Code on the Vulnerable Web Application whose function is to steal cookies of the victim's account session.
- Secondly, the victim logs into the vulnerable web application by giving the user-id and password. As a result, the web server of web application will generate and transfer the cookie of that particular session to victim's web browser.
- In the third step, the victim browses the malicious Java Script Code and gets executed on its browser.
- In the fourth step, the Java Script Interpreter of the victim's browser gets invoked and transfers the cookies of the victim's session to the attacker's domain.
- Now lastly, these cookies will be utilized by the attacker to get into the account of victim.

In this way, XSS attack gets exploited on the victim's domain. The related work on XSS attacks has been surveyed focusing on some issues related to XSS attacks. The survey has been divided into three categories namely Exploitation, Detection and Prevention of XSS attacks.

Exploitation of XSS Vulnerability

Recently, researchers have shown some basic ways to demonstrate how XSS attacks can be used to control and modify the functionality of a web page. Various types of platforms (like Acunetix [9] etc.) are

available online to test or exploit some vulnerabilities of XSS attack. Acunetix test website offers the platform to a user who wants to exploit the vulnerabilities of XSS attack. It is a way of limiting security testing to only systems that we own, or have permission to work with.

A. Detection of XSS Vulnerability

In static detection of XSS, testing is generally performed by source code analysis. On the other hand, in dynamic testing of XSS, known attacks are executed against web applications. Recently, researchers have proposed various detection techniques to discover the XSS Attacks. In [10], a Webmail XSS fuzzer called L-WMxD (Lexical based Webmail XSS Discoverer), which works on a lexical based mutation engine is an active defence system to discover XSS before the webmail application is online for service. The researchers have run the L-WMxD on over 26 real-world Webmail applications and found vulnerabilities in 21 Webmail services, including some of the most widely used Yahoo-Mail. In [11], a static analysis for finding XSS vulnerabilities has been put forward that directly addresses weak input validation. This approach combines work on tainted information flow with string analysis.

Pixy [12] is a tool that performs data flow analysis on PHP code to detect reflected XSS vulnerabilities. Various prototype tools which are based on Pixy have been implemented by the researchers and test on the real world PHP programs. Similar approaches have been adopted by commercial products like AppScan [13], Nessus[14] and so on.

B. Prevention of XSS Vulnerability

Cross-site Scripting (XSS), the top most vulnerability in the web applications, demands an efficient approach on the server side as well as client side to protect the users of the web application. In [15], an application-level firewall is suggested, which is located on a security gateway between client and server and which applies all the security relevant checks and transformations. Some server side prevention approaches require the collaboration of web browsers. One such example is BEEP (Browser-Enforced Embedded Policies) [16], a mechanism that modifies the browser so that it cannot execute the malicious scripts. Security policies dictate what the server sends to BEEP-enabled-browsers. Apart from this, the researchers developed the WebSSARI (Web Security via Static Analysis and Runtime Inspection) tool [17], which performs type-based static analysis to identify potentially vulnerable code sections and instrument them with runtime guards.

On the client side, researchers have developed the Noxes [18] which acts as a personal firewall that allows or blocks the connections to websites on the basis of filter rules, which are generally user-specified URL white-list and blacklist websites. When the browser sends a HTTP request to an unknown website, Noxes immediately alerts the client, who chooses to permit or deny the connection, and remembers the client's action for future use. Another client side approach is presented in [19], which aims to identify the information leakage using tainting of input data in the browser. In [20], a mechanism for detecting malicious java script is proposed. The system consists of a browser-embedded script auditing component, and IDS that processes the audit logs and compares them to signatures of known malicious behavior or attacks. Several server-side countermeasures do exists, but such techniques have not been universally applied because of their deployment overhead. On the other hand, existing client side solutions degrade the performance of client's system resulting in poor web surfing experience. The necessity to install updates or additional components on each user's web browser or workstation also degrade the performance of client side solutions.

The solutions mentioned above regarding prevention of XSS attack cannot prevent the XSS attack completely. So, in the next section we have proposed a new technique which is implemented on the server side, whose aim is not to protect the cookies from XSS attack, but make the cookies worthless for the attackers.

3. Proposed Method

In this section, we present a novel procedure called Dynamic Hash Generation Technique, whose main objective is to make the cookies useless for the attackers. This approach is easily implemented on the web server without any changes required on the web browser. With this technique, the web server will produce a hash of value of name attribute in the cookie and send this hash value to the browser, so the browser will keep the hash value of cookie in its database rather than the original value. Now each time, if the browser wants to reconnect as a part of active connection, the browser has to include the hash cookie value into its corresponding request so that the web server will also rewrite this hash cookie value to

the original value, which is generated by the web server. Rewriting of hash value to original value is necessary to be done at the server side, so that the user at the browser side will get authenticated by the web server. As the browser stores the hash value of cookies, so even the XSS attack can steal the cookies from browser's database, the cookies cannot be used later to hijack or take off the user's session.

We have conducted the experiments on version 0 cookies in which three attributes (name, domain and path) are specified for the identifying the cookies uniquely. Following Table 1 is used for storing the cookies of version 0 on the server side.

Table 1: Original and Hash Value of Cookie

Name	Domain	Path	Original Value	Hash Value
Cookie 1	www.ace.gov.in	/loc1	789pqrs	+#2g&)@
Cookie 2	www.lal.ac.in	/loc2	432frtg	*$67+e#

In this paper, we have used the Dynamic Hashing Generation Technique on the server side, which is used to generate the hash of value of name attribute in the cookie. All the other attributes (i.e. domain and path attributes as shown in the Table 1) will remain same. Following are some of the steps which are used to explain the Dynamic Hashing Generation Technique:-

- The user on the web browser side submits the user-id and password to the web server of the web application.
- The web server submits the corresponding information from the browser and generates a cookie.
- Now the web server will dynamically generate the hash of value of the name attribute in the cookie and store both these values (original as well as hash value) in the form of a table on the server side.
- Subsequently, the web server will send the hash value of the name attribute in the cookie to the web browser.
- The web browser will store this hash value into its repository.

Since the cookies (hash version) at the browser's database now are not valid for the web applications. Therefore XSS attack will not be able to impersonate the user using stolen cookies which are converted into its hash form. Now if the browser wants to reconnect to the web server as a part of the active connection, it has to include cookie (hash value) with its corresponding request to the web server. The web server will use the information in the table to rewrite back the values of name attribute in the cookie (sent by the web browser) to the original value generated by the web server as shown in the Figure 2.

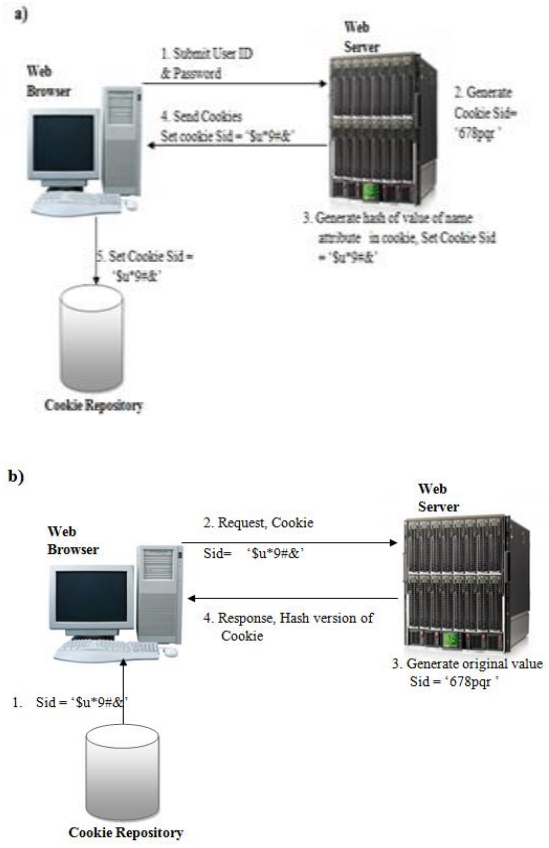

Figure 2: Dynamic Hash Generation Technique

4. Results and Analysis

We have tested our proposed technique on four modern web browsers like Google Chrome, Firefox, IE v8 and Safari by using the services of XAMPP server. We have evaluated our approach with the version 0 cookies using all these browsers on the following test cases:

- Set-Cookie: SID = pqrs123
- Set-Cookie: SID = abcd456; Domain = .trial.com
- Set-Cookie: SID = stuv890; Domain = .trial.com; Path = /area1

- Set-Cookie: SID = lmno678; Domain = .trial.com; Path = /area2
- Set-Cookie: SID = ijkl456; Domain = .trial.com; Path = /area1

We have also evaluated our approach with the real time web applications on the internet and the results showed that our proposed technique worked well. We have seen in our experiments that the web server successfully generate the hash of the value of name attribute in cookie and browser stores and returns this value to the web server on every subsequent request.
We have also tried to steal the cookies from a browser's database by exploiting the vulnerabilities of XSS attacks on the browser side and the results showed that stolen cookie in the form of hash value cannot impersonate the user.

The efficiency of our proposed technique and their corresponding results has been described in the form of associated strengths and weaknesses which is elaborated as below:

C. Strengths

- The proposed technique described above does not affect the performance of client side web browser resulting in superior web surfing experience.
- This technique does not suffer from single point of failure. As the web server fails for some time, in any case, it will not send the original cookies to the browser.
- Even if attacker will perform the XSS attack to steal the cookies from the browser's repository, the attacker will get the hash version of the cookies, which are not appropriate to impersonate the user.
- This approach is compatible with all the modern web browsers like Google Chrome v6, Firefox v3, Internet Explorer v8, Safari v4 etc.
- The proposed technique works well with the version 0 cookies and does not have any side effects on the HTTP protocol.
- Since XSS vulnerability exists in all types of platforms, so we have tried to make our proposed technique as platform independent and it has been implemented on various platform independent browsers, so it can be used with other operating systems with fewer changes.

D. Weaknesses

- The results have shown that this server side solution degrades the performance of the whole system.
- Generating the hash code of cookie on the server side adds more latency and increases the response time.
- Our approach does not work well with the version 1 cookie as this version adds an additional attribute (i.e. port number) for the identification of cookies.
- Our proposed technique does not able to intercept the HTTPs and SSL connections.
- Our approach has faced the problems with some of the real time websites while producing the hash value of the cookie in the HTTP header.

5. Conclusion and Future Work

This paper has presented the Dynamic Hash Generation technique, whose main purpose is to make the cookies worthless for the attackers even if the attacker successfully exploits the vulnerabilities of XSS attacks on the victim's web browser. This technique has been implemented on the web server and the results showed that our technique worked well with the Version 0 cookies on all the modern web browsers except opera.

Currently we are working on how our proposed technique works with the Version 1 cookies on the real world websites. In future, we would also like to develop an analysis on the web browser side to discover the set of strings that can cause their java script interpreter to be invoked.

References

[1] D. Gourley, B. Totty, M. Sayer, S. Reddy, and A. Aggarwal, HTTP The Definitive Guide, 1st ed., O'Reilly Media, US, 2002.
[2] D. Kristol, "HTTP State Management Mechanism,", in Internet Society, 2000. Available: http:// www.ietf.org/rfc/rfc2965.txt
[3] "Cross Site Scripting Techniques and mitigation,", GovCertUK, revision 1.0, October 2009. Available: www.govcertuk.gov.uk.
[4] Open Web Application Security Project, "The ten most critical web application security vulnerabilities", 2007, www.owasp.org/index.php/OWASP_Top_Ten_Project
[5] J. Garcia-Alfaro and G. Navarro-Arribas, "Prevention of Cross-Site Scripting Attacks on

Current Web Applications,", Available: http://hacks-galore.org/guille/pubs/is-otm-07.pdf
[6] S. Saha, "Consideration Points: Detecting Cross-Site Scripting," (IJCSIS) International Journal of Computer Science and Information Security,Vol. 4, No. 1 & 2, 2009.
[7] UW Staff Web server [Online]. Available: http://staff.washington.edu/fmf/2009/06/19/setting-cookies/.
[8] Wikipedia website [Online]. Available: http://en.wikipedia.org/wiki/Talk%3AHTTP cookie.
[9] Acunetix, "http:// www.acunetix.com/".
[10] Zhushou Tang, Haojin Zhu, Zhenfu Cao, Shuai Zhao, L-WMxD: Lexical Based Webmail XSS Discoverer, IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 976-981, 2011.
[11] Gary Wassermann, Zhendong Su, Static Detection of Cross-Site Scripting Vulnerabilities, ACM/IEEE 30th International Conference on Software Engineering (ICSE), pp. 171-180, 2008.
[12] N. Jovanovic, C. Kruegel and E. Kirda, Precise alias analysis for static detection of web application vulnerabilities. In: ACMSIGPLAN Workshop on Programming languages and Analysis for Security, Ottawa, Canada, 2006.
[13] AppScan,http://www01.ibm.com/software/awdtools/appscan/.
[14] Nessus, http://www.nessus.org/.
[15] D. Scott and R. Sharp. Abstracting Application-level Web Security. In 11th World Wide Web Conference, 2002.
[16] M. T. Louw and V. N. Venkatakrishnan, "Blueprint: Robust Prevention of Cross-Site Scripting Attacks for Existing Browsers", Proc. 30th IEEE Symp. Security and Privacy (SP 09), IEEE CS, pp. 331-346, 2009.
[17] W. Halfond, A. Orso, and P. Manolios, "WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation", IEEE Trans. Software Eng., pp. 65-81, Jan. 2008.
[18] E. Kirda et al., "Client-Side Cross-Site Scripting Protection," Computers & Security, pp. 592-604, Oct. 2009.
[19] P. Vogt, F. Nentwich, N. Jovanovic, C. Kruegel, E. Kirda, and G. Vigna. Cross site scripting prevention with dynamic data tainting and static analysis. In 14th Annual Network and Distributed System Security Symposium (NDSS), 2007.
[20] O. Hallaraker and G. Vigna, Detecting Malicious JavaScript Code in Mozilla. In Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), 2005.

Shashank Gupta was born in Jammu on 25th September, 1987. He has completed his graduation in B.E. (I.T.) from Pune University, Pune. He has also qualified the GATE 2010 with 94 percentile. He has obtained his M.Tech. (CSE) degree, specialization in Information Security from Central University of Rajasthan, Ajmer. He is currently working as a Lecturer in the department of Information Technology in Model Institute of Engineering and Technology, Jammu. His area of interest includes Cryptography, Network Security, Theory of Computation and Database.

Lalit Sen Sharma was born in the district of Bilaspur, Himachal Pradesh, India, on April 12, 1969. He has obtained Master of Science in Mathematics and MCA from Guru Nanak Dev University, Amritsar (India). He has also obtained Doctorate of Philosophy (PhD) from Guru Nanak Dev University in 2008. Currently, he is working as an Associate Professor in the department of Computer Science and Information Technology in University of Jammu, India. He has been teaching to postgraduate students in computer applications for fifteen years. He is a life member of Indian Science Congress Association, Institute of Electronics and Communication Engineer, India and National HRD network, India. His area of interest includes data communication and networking, internet and web services. He has also organized workshops and acted as a member of organizing committee to organize national conferences.

Manu Gupta was born in Jammu on 1st October, 1982. He has completed its B.tech. (I.T.) from U.P. Technical University in 2004. He has obtained M.Tech. (I.T.) from Punjab Technical University (PTU) in 2011. Currently he is working as an Assistant Professor in Department of I.T. in M.I.E.T. Jammu. His area of interest includes Cryptography and Network Security.

Simi Gupta was born on 14th January, 1983 in Jammu. She has completed its B.tech. (I.T.) from Model Institute of Engineering and Technology, Jammu. She is also pursuing the M.Tech.(CSE). Currently she is working as a Lecturer in M.I.E.T., Jammu. Her area of Interest includes Digital Logic, Web technologies.