ADV MATH SCI JOURNAL
Advances in Mathematics: Scientific Journal **9** (2020), no.7, 4513–4521
ISSN: 1857-8365 (printed); 1857-8438 (electronic)
https://doi.org/10.37418/amsj.9.7.20 Spec. Issue on RDESTM-2020

A NOVEL AND EFFICIENT TECHNIQUE FOR PREVENTION OF XSS ATTACKS USING KNAPSACK BASED CRYPTOGRAPHY

DEEPAK DEMBLA, YOGESH CHABA[1], KULDEEP YADAV, MRIDUL CHABA, AND ASHISH KUMAR

ABSTRACT. The problem is to protect cookies from cross-site scripting attacks (XSS). The major challenges with the cookies are user information, no randomness, no secure transportation and size etc. There is always very easy chance for attackers to attack. There are many XSS detection and prevention techniques available but have their own limitations. In this paper we propose a client side solution to protect the cookies from the XSS attacks. We propose a knapsack cryptographic local proxy with encryption and decryption capability. This approach encrypts the cookies value (session-ID) attribute in the cookie at cryptographic local proxy before sending it to the browser, Further the requests of client with encrypted cookie is sent to cryptographic local proxy where it decrypts these requests and forward them to the web server. This technique will let Browser store the encrypted cookie which will not be useful for the attacker even if it is stolen.

1. INTRODUCTION

HTTP is a stateless protocol used by browser. Browser does not maintain a connection during accessing a page. During access of the page browser may repeat request to same server after sometime or never. So we need a way to tell a server which browser is this. Therefore, both sever and browser maintain

[1]*corresponding author*

2010 *Mathematics Subject Classification.* 94A60, 68P25.

Key words and phrases. Cross-site Scripting, Cookie, Session-id, Encryption, Cryptography, Local proxy.

a state. State stored in browser is called a cookie, similarly the state stored in the server is called a session. When a server is interacting with many different browser at the same time, the server need to know from which browser a request came. Without state all browser look same, hence a lot of problem may arise. Without cookie, each web page is an individual entity. Since the cookies are used for identifying and authenticating the user, which makes cookies a soft target for intruders. According to survey XSS attack is one of the most common attacks for a web application to steal the cookie using malicious script. Using these cookies, the attacker can impersonate identity and can then act as that user or hijack the user's session [1], [6].

Cookie are small amount of information selected by a server which is then send to user's browser. It will be stored in user's hard disk. The data in cookie can depend upon the server or type of application. For e.g. cookie can be used for authentication are different from cookie used during accessing shopping application. It is then stored in the users hard disk, the location and name of file depends upon operating systems respectively. Further browser send a GET or POST requests to the server with the cookie set by the server [5]. There are 2 different types of cookies :

Persistent Cookie: Persistent cookies are permanently stored on user hard disk until user delete or it expire. How long a cookie store into the browser depends on the web server and that particular website.

Session Cookie: These are stored temporally into the web browser and are deleted once user close his browser at the end of users current session. A session cookie or session token is a piece of textual information that is used in communication to identify a session for related message interchanges. When stateless protocol like HTTP is used then Session identifiers are required.
There are two versions of cookies in use now a days.

Version 0 cookies (NetScape cookies): The version 0 cookie is the most widely used and is basic most version.

Version 1 cookies (RFC 2965): The higher version of version 0 or Netscape cookie was launched as version 1 cookie. In version 1 cookie an attribute port number is added. Port no identifies the port on which application is running. Cookie header is same but format is different. All modern browsers normally do not use cookies of version 1 (except Opera), so these cookies are not widely used by web developers.

XSS is security threats that exists in a web applications. The target of attacker is the client browser to execute malicious scripting commands like Java,VB and any other type of script. It occurs due to improper input validation. XSS can steal the web browser session cookie, and he can hijack the user's session by using stolen cookie [7].
There are three types of XSS:

1. *Non-Persistent or Reflected XSS:*
   It is the most common type of XSS. In this first the attacker sends victim the link to with XSS payload. Then victim click on the link. Further XSS payload execute and steals cookie from victim browser.
2. *Persistent or Stored XSS:*
   This type XSS vulnerability mostly occurs when first attacker stored some malicious script at server, and then when any user access those page it will send to user's location and execute on client browser and cookie will be redirected to attacker location.
3. DOM based XSS:
   DOM is Document Object Model which makes all the component of a web page accessible. Object can be created, updated and deleted with the help of java script. In order to change the flow of the code, the attacker inject DOM element along with the code. The DOM element are executed by the server without the knowledge that what the sever is executing.

2. PRELIMINARIES

There are preventive measures set by Open web application security project (OWASP). There are certain rules to be followed by web developer that can simply prevent the XSS at basic level. But while developing the web application if we don't adhere by the rule given by OWASP then there remains vulnerability for XSS attacks [3], [10]. XSS could be prevented either at client side or at server side. Following are the existing methods to prevent the XSS attacks but each having their own limitations. Kirda et al. [2] in 2009 has given a client side mechanism to detect and prevent XSS attack. It has used a window based firewall called Noxes. Noxes filters all the request and will inform the user by

alarming system, when no rule will be found in rule set for a domain. Here when a information send to unknown location, alarm will raise. So it will stop XSS but still there is a chance of denial of service attack that does not transfer cookie or any other important information as given by Kristal in 2001 [2]. The concept Http-only cookies were used by Microsoft for web Internet Explorer 6. The browser will deny scripting languages to access those cookies. The Http-only cookie alone can not prevent all types of XSS attack [6], [8]. Dacosta et al. in 2012 [4] has given the statement that the client and the web servers only send the cookies via the SSL connections. Thus eavesdropping can be protected, whenever they send packet. The other client side solution proposed for detection malicious javascript code. In this solution, Mozilla Firefox web browser use an auditing system for detection misuse and anomaly. This technique based on the monitoring of javascript execution and comparison of predefine rule and policy to identify behaviour. For each malicious behaviour rules should be defined to identify cross site script [12]. Many approaches have been proposed for XSS attacks but very less work has been done using cryptographic approach.

3. Proposed Approach

In this model we propose a client side solution in which we design a new cryptography local proxy with specific browser whose cookie we want to prevent against XSS attack as shown in figure 1. This paper uses Knapsack based cryptographic technique [9] for encryption. This cryptography local proxy intercept HTTP request and HTTP response [11] and subsequently encrypt the cookie's attribute (session-id) from HTTP header and decrypt the encrypted cookie as well. In figure 2, we want to encrypt the cookie for Browser-2 so we configure the cryptographic local proxy with browser-2 only.

This model is efficient because to implement this system we don't need the modification at the server side or some vast modification at browser, just simply configure our crypto- graphic local proxy and change the proxy setting at browser. We can configure this tool with specific protocol so that it can not negatively effect the speed.

In knapsack system input is (**x**, *m*) where $\mathbf{a} = (x_1, x_2, ..., x_n)$ is a vector of positive integers and *m* is a positive integer, represented in some base. Here *m* is written as sum of different attributes of **x**.. Here we have to select bits

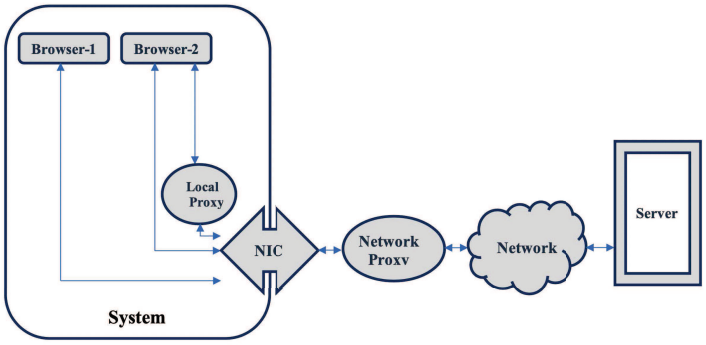

FIGURE 1. System model with local proxy for HTTP and HTTPS traffic

$z_1, z_2, ..., z_n$ such that

$$\sum_{j=1}^{n} z_j x_j = m\,.$$

Process of KNAPSACK encryption is done by treating message symbols as bits where *n* indicates length of message block. A message block $w = y_1 y_2 .... y_n$ (bit sequence) is encrypted as the number

$$z = e_k(w) = \sum_{j=1}^{n} y_j x_j\,.$$

Here bits are message symbols and the length of the message block is *n*. A message sequence $w = y_1 y_2 .... y_n$ is encrypted as the number z. The public key *k* is **x**. Finally this kind of encrypting is available in *P*.

We configured this approach with specific protocol i.e. HTTP and HTTPS because application are accessed through these protocol. The proposed method will be easily compatible with any browser and any platform.

Initially, the browser send its first GET request to a server. The server then checks whether it has a cookie for this request or not. If not, then server assume this request as the first request. The server sets a cookie with some name (session id) and sends it back along with the rest of the response. The cookie send by the server is encrypted at proxy i.e built in with the browser for HTTP and HTTPS request and response. And the encrypted cookie is send to the browser. Now browser will store the encrypted cookie. Further browser send a GET or POST requests to the server with the encrypted cookie that is first decrypt at proxy then forward to the server .

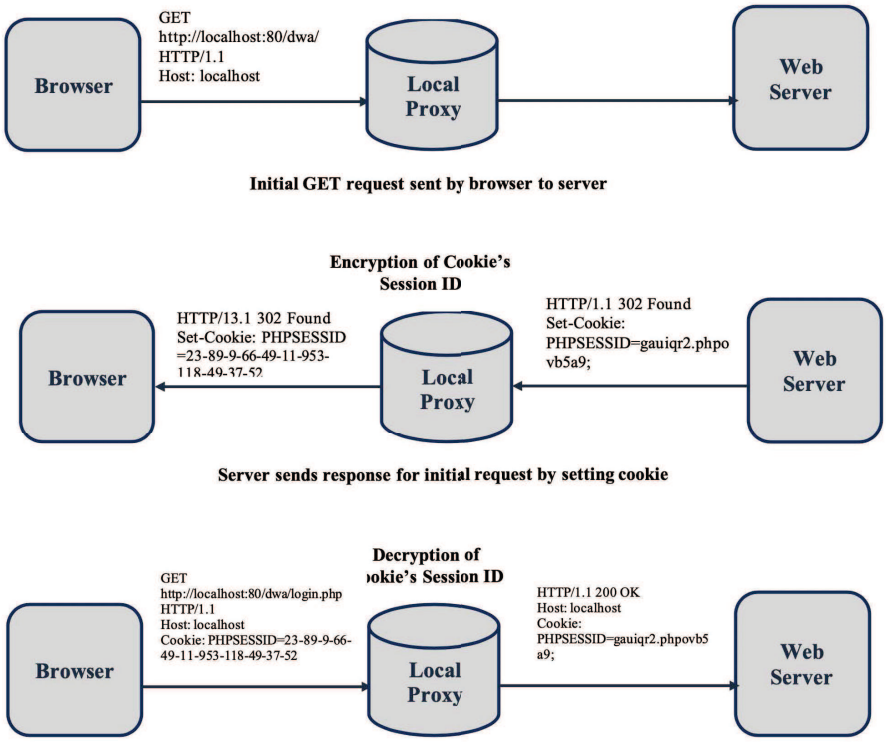

FIGURE 2. Proposed Technique

4. RESULTS AND DISCUSSION

Experimental setup is described as follows:

- *Platform:* Windows 7, processor Intel i5 3rd generation, 2GB RAM.
- *Sever:* XAMPP (Apache web server, MySQL)
- *Browser:* Mozilla Firefox
- *Tool:* Cryptographic local proxy
- *Implemented On:* Damn Vulnerable web application (DVWA)

The cookie which is sent or received via HTTP request and HTTP response header is passed through the cryptographic local proxy and get encrypted.This encrypted cookie need a memory space to be stored. Implementation limits related to this memory space are given in (RFC-2109) [5]. These cookies are store only till one transaction(one request and one response) get completed. There are two scenarios. The first scenarios of the First browser accessing web application without our approach is illustrated in Table 1 and the second scenario of the

First browser accessing web application with our approach is illustrated in Table 2 Performance: By deploying cryptographic local proxy, definitely increases the response time but with the latest hardware platform there will not be significant performance degradation. Using the proposed technique for prevention of XSS , their chances of attacks reduces by 95% in websites. Further following improvements are also observed in our approach:

Small Storage space: In our proposed technique, we require small storage space at cryptographic local proxy to modify http header.

Compatibility: Proposed approach modifies the values of the cookies in http header, it does not lead to invalid the http and be able to work well and transparently with all websites on the internet. Although most websites on the internet use only the version 0 cookies, however our approach follows both version 0 and version 1 cookie specifications in order to be able to manage properly all cookies sent by all websites.

Security : Knapsack algorithm is used for encryption in the proposed techniques which makes it secured and robust. The cookies at the browser are encrypted and at the server needs decryption with the same algorithm and dynamically gets changed for each session. Hence, using this proposed technique, the whole process becomes robust.

Cost effective: Since, there is no extra cost of any hardware (like Firewall etc.) is involved, the proposed technique is very cost effective.

Efficiency: The proposed model is efficient because to implement it, we don't need the modification at the server side or some vast modification at browser, just simply configure our crypto- graphic local proxy and change the proxy setting at browser.

Complexity : The proposed technique rejects the need of setting up any filtering engine or sanitization solution as only a little modification in browser setting is required for configuring the secured proxy. Hence, reduces complexity at client and the server levels both.

Response Time: Although there is a little increase in the response time, the proposed techniques has increased the security to a larger extent.

S. No.	Cookie at web server	Cryptographic Local proxy	Cookie at Browser
1	gauiqr2.phpovb5a9	Not configured	gauiqr2.phpovb5a9
2	mp17f7o9m8ksqkgt	Not configured	mp17f7o9m8ksqkgt

TABLE 1. First browser accessing web application without our approach

S. No.	Cookie at Web Server	Cryptographic Local proxy	Cookie at Browser
1	gauiqr2.phpovb5a9	conured	23-89-9-66-49-11-953-118-49-37-52
2	mp17f7o9m8ksqkgt	configured	121-37-105-46-30-13-123-511-125-80-19-33-5-122-39

TABLE 2. First browser accessing web application with our approach

5. CONCLUSION

XSS is a method used to exploit communications between users and applications. It is security threats that exists in a web applications which need to be reduced. We have developed an approach which protects the cookie from the cross site script (XSS) attacks. This technique is implemented with the help of Knapsack based cryptographic local proxy without any change required on both web browser and web server. The technique encrypts value of (session ID) attribute of the cookie at the cryptographic local proxy before sending it to the browser, now the browser will store the encrypted cookie which will not be useful for the attacker if it steals the using XSS. In this research paper, we have implemented a novel method of protecting the cross site scripting attacks by use of encrypted cookies at both client and the server using Knapsack algorithm, cookies are encrypted each a new user log in the browser and are dynamic in nature. Performance improvement has been observed in terms of Storage space, Compatibility, Security, Cost effective, Efficiency and complexity.

REFERENCES

[1] A. V. STOCK, B. GLAS, N. SMITHLINE, T. GIGLER: *The Ten Most Critical Web Application Security Risks* , OWASP Top 10, 2013.

[2] E. KIRDA, N. JOVANOVIC, C. KRUEGEL, G. VIGNA: *Client-side cross-site scripting protection* , Computers and Security, **28** (2009), 592-604.

[3] L. K. SHAR, H. B. K. TAN: *Automated removal of cross site scripting vulnerabilities in web applications*, Information and Software Technology, **54**(5) (1970), 467-478.

[4] I. DACOSTA, S. CHAKRADEO, M. AHAMAD, P. TRAYNOR: *One-Time Cookies: Pre- venting Session Hijacking Attacks with Stateless Authentication Tokens*, ACM Transactions on Internet Technology, **12**(1) (2012), 1-24.

[5] V. KHU-SMITH, C. MITCHELL: *Enhancing the security of cookies*, ICICS-LNCS (2002), 132-145.

[6] N. NIKIFORAKIS, W. MEERT, Y. YOUNAN, M. JOHNS, W. JOOSEN: *SessionShield: Lightweight protection against session hijacking*, 3rd International Symposium on Engineering Secure Software and Systems (ESSoS 2011), **6542** (2011), 87-100.

[7] I. HYDARA, A. B. M. SULTAN, H. ZULZALIL, H. N. ADMODISASTRO: *Current state of research on cross-site scripting (XSS) - A systematic literature review*, Information and Software Technology, **58** (2015), 170-186.

[8] OWASP: *Introduction of HTTP-only Cookies*, Open Web Application Security Project.

[9] W. STALLINGS : *Cryptography and Network Security: Principles and Practices*, 4th ed., Pearson Education, Inc Pvt. Ltd, 2005.

[10] J. GROSSMAN, R. HANSEN, P. PETKOV, A. RAGER: *XSS Attacks: Cross site scripting exploits and defence*, Syngres Publishing Elsevier Inc., 2007.

[11] D. M. KRISTOL: *HTTP Cookies: Standards,Privacy,and Politics*, ACM Transactions on Internet Technology, **1**(2) (2001), 151-198.

[12] R. PUTTHACHAROEN, P. BUNYATNOPARAT: *Protecting cookies from cross site script attacks using dynamic cookies rewriting technique*, 13th International Conference on Advanced Communication Technology (ICACT) (2011), 1090-1094.

JECRC UNIVERSITY JAIPUR, INDIA

Email address: dembla.deepak@gmail.com

GURU JAMBHESHWAR UNIVERSITY OF SCIENCE AND TECHNOLOGY, HISAR, INDIA
Email address: yogeshchaba@yahoo.com

DIRECTORATE OF EDUCATION, NEW DELHI, INDIA
Email address: kdresearch2015@gmail.com

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY, NEW DELHI, INDIA
Email address: mridul.chaba@gmail.com

MANIPAL UNIVERSITY JAIPUR, INDIA
Email address: kumar.ashish@jaipur.manipal.edu