Cookie Injection Attack

Research & Development Project

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
by
Astha Jada & Achala Bhati
(153050027 & 153050056)

Supervisor:
Bernard L. Menezes

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai 400076 (India)

29 December 2016

Table of Contents

List of Figures

Chapter 1

Problem Statement

This document describes the Cookie Injection Attack which is the recent attack in the field of web security. This attack is possible due to the vulnerabilities present in the current browsers like Chrome, Safari, Firefox etc. When the attack was first discovered, even Google and Bank of America were vulnerable to the attack.

Chapter 2

Background

2.1 Vulnerablity for the attack

The same origin policy of cookies separates the content of one domain from another. However, this works in unusual way in the case of cookies. The isolation of domain of cookies is weak. The secure flag of cookies ensure that the cookie should be sent over HTTPS connection only however the cookie can be set with the value of secure flag true from HTTP connection also. Hence cookies can be sent over HTTPS connection that are set by HTTP connection.For example, a cautious user might only visit news websites at open wireless networks. But an attacker can inject malicious cookies to poison her browser, and compromise her bank account when she later logs on to her bank site at home.

There is [5]an unusual behaviour of cookies. Cookies are set and stored as a name/domain/path to value attributes mapping, but only name-value pairs are presented to both Javascript and web servers. This asymmetry allows cookies with the same name but different domain and/or path scopes to be written into browser; a subsequent reader can read out all same name cookies together, yet cannot distinguish them because the other attributes such as path are not presented in the reading process. Also, an origin for a given URL is defined by a 3-tuple: scheme, e.g. HTTP or HTTPS, domain and port. However, the security policy of cookies does not provide separation based on either scheme or port but only on domain.In addition, a website can set cookies with flexible domain scopes: 1) not shared (i.e., host-only), 2) shared with its subdomains, or 3) shared with its sibling domains.

Because of these reasons,the integrity of the cookie from an active Man In The Middle (MITM) or [1] a malicious related domain that shares some cookie domain scope with it cannot be preserved.

2.2 Types of Cookie Injections

1. **Cookie Overwriting:** If a cookie shares the domain scope with a related domain, it can be directly overwritten by that domain using another cookie with the exactly same name/domain/path.

2. **Cookie Shadowing:** An attacker with the control of a related domain can intentionally shadow a cookie by injecting another one that has the same name, but different domain/path scope.

Chapter 3

Experiments

3.1 Cookie Over writing :

3.1.1 steps :

- Created two web server using Xampp server. Both Server have same suffix domain.
- One is attacker website and other is valid server.
- Now user visited valid website, this website set a cookie.
- user visited attacker website and this website over write cookie set by the valid server.

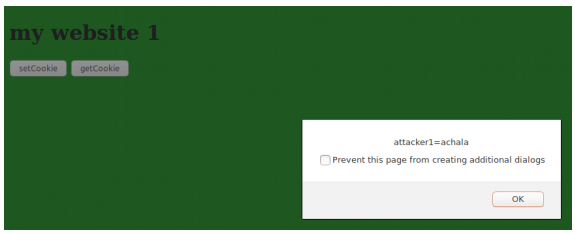

Figure 3.1: Cookie set by the valid Server

Figure 3.2: Cookie set by the malicious Server

Figure 3.3: Cookie set by the valid Server

3.2 Malicious Cookie Insertion :

If we have two website with same domain name, we can insert malicious cookie in user system so that whenever user visited valid server these malicious cookie will be sent with it. We have inserted these cookie using simple java script language.

3.2.1 steps :

- First we have created a dummy server which have same domain as sharelatex.
- Now whenever user visited that server, this server [3] inserts cookie with domain name sharelatex.com.

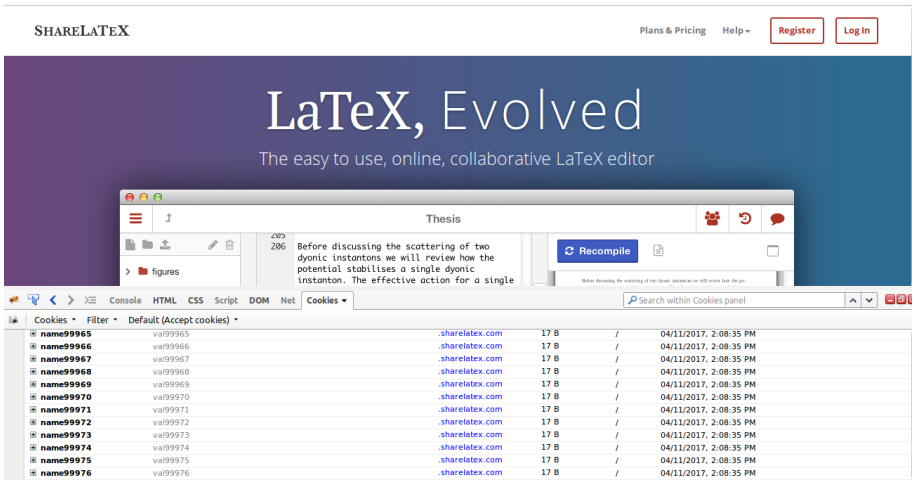

Figure 3.4: Malicious Cookie insertion in the valid server

3.3 Denial of service using Cookie Attack :

We have implement of this attack on two real website www.sharelatex.com and gpo.iitb.ac.in.

3.3.1 Idea :

Idea was based upon [4]the fact that whenever user send request to a server then request header will contain all cookie with requested domain scope. If we increase the cookie size or number of cookie to be inserted then request header size become greater than the maximum allowable request header size. So request send by the user get reject, Hence although user is a valid and the server to which user is sending request is valid , but user is not able get the service from the server.

3.3.2 steps :

- First we have created server with same domain (here sharelatex.com and gpo.iitb.ac.in).
- Now we have created [2]lots of cookie whenever user visited these dummy website.
- After creating lots of cookie with large size , now user visited the valid server www.sharelatex.com and gpo.iitb.ac.in.
- After sending request the request header become large and this request gets rejected at the server side.

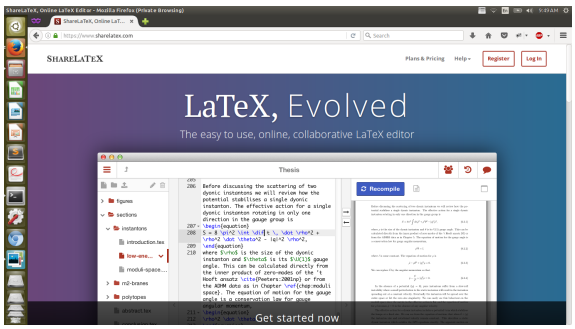

Figure 3.5: Before Inserting cookie, response from www.sharelatex.com

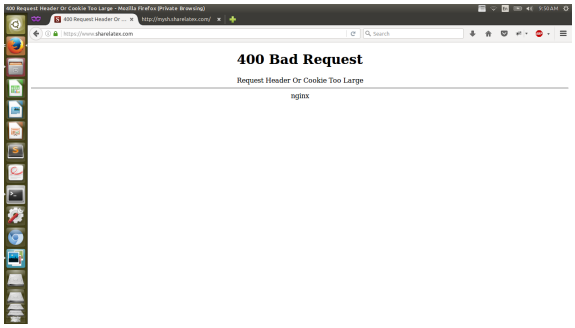

Figure 3.6: After Inserting cookie, response from www.sharelatex.com

Figure 3.7: Before Inserting cookie, response from gpo.iitb.ac.in

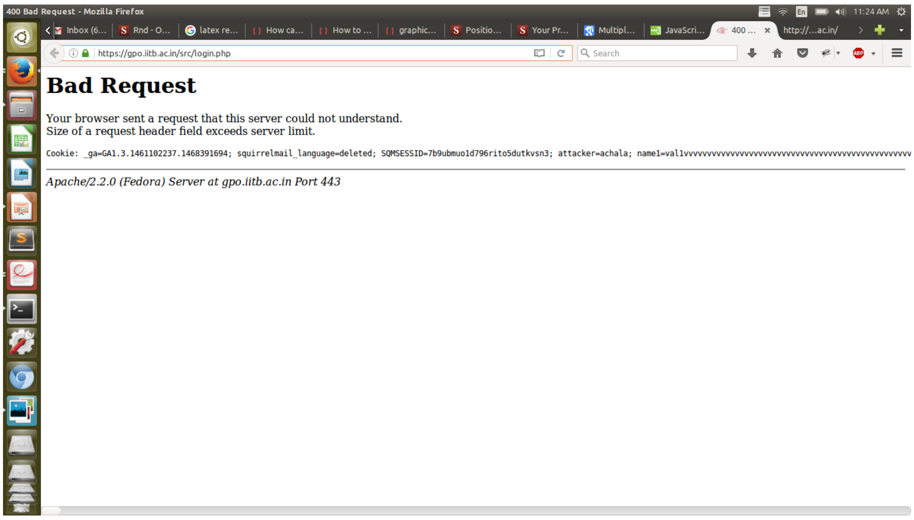

Figure 3.8: After Inserting cookie, response from gpo.iitb.ac.in

Chapter 4

Mitigation

4.1 HSTS

HSTS (HTTP Strict Transport Security) allows a server to inform a client to only initiate communications over HTTPS. After receiving an HSTS header, a conforming browser ensures that all subsequent connections to that domain always take place over HTTPS until the policy expires. The websites should deploy full HSTS to prevent cookie injection from active network attackers, as this provides complete protection once a site is pinned by a user visit.

4.2 Anomaly Detection

Websites should consider detecting same name cookies in the cookie header.This is reasonable because same name cookies should not be considered a legitimate use according to both the specification and the inconsistent implementations.

References

[1] Bortz, A., Barth, A., and Czeskis, A., 2011, "Origin cookies: Session integrity for web applications," *Web 2.0 Security and Privacy (W2SP)*

[2] Cookie, Insertion and overwriting, `http://www.w3schools.com/js/js_cookies.asp`

[3] Cookie, E. T., Cookie Handling, `http://www.editthiscookie.com/`

[4] Johnston, P., and MOORE, R., 2004, "Multiple browser cookie injection vulnerabilities," *Westpoint Security Advisory, http://www. westpoint. ltd. uk/advisories/wp-04-0001. txt*

[5] Zheng, X., Jiang, J., Liang, J., Duan, H., Chen, S., Wan, T., and Weaver, N., 2015, "Cookies lack integrity: Real-world implications," in *24th USENIX Security Symposium (USENIX Security 15)*, pp. 707–721.