

Website Vulnerability to Session Fixation Attacks

Bhavna C.K. Nathani Erwin Adi*
School of Computer Science, Binus International, Bina Nusantara University, Jl. Hang Lekir 1 No. 6, Senayan, Jakarta 10270, Indonesia
* E-mail of the corresponding author: eadi@binus.edu

Abstract
Session fixation is a vulnerability of web applications where a malicious attacker gains full control of a victim's web account without having to use the victim's credentials such as username and password. Extant defensive techniques and procedures are not completely effective against such attacks. The authors found that some 48% of Indonesian websites are vulnerable to such attacks because, contrary to best software engineering practices, many use default session management IDs generated by their development platforms. This paper presents procedures for identifying vulnerable websites and the results.
Keywords: web application security; session fixation; session hijacking

1. Introduction

In http communication, various TCP connections are used to access websites. To recognize every user's connection, a web server creates a session ID and sends this to the user's browser in various ways: as a cookie in the HTTP response body, or as a string of characters in the URL or in any part of the HTTP response header.

A malicious attacker can impersonate the user by sending a request to the web server that contains this session ID. This technique, known as session hijacking, is ranked 3rd among the world's most critical security risk, up from its previous rank of 7th in 2007 (OWASP, 2010)

One subset of session hijacking is to make the user access a web application using a session ID created by the attacker. This procedure, called session fixation (Kolsek, 2002) works as follows: first, the attacker interacts with a web server to obtain a session ID. Through phishing or some other means, the attacker tailors a message for a user to click. By doing so, a user unknowingly submits a session ID to the server on which a web application session has been owned by the attacker. Thus the user becomes a victim.

One report suggests that such attacks can be avoided by using web applications that change the session ID after each login (Takamatsu, Kosuga, & Kono, 2010). However, our investigation found that such attacks succeed despite post-login change of session IDs. Curiously, the investigation also found websites that are immune to session fixation attacks yet use the same session IDs.

This paper contributes to the existing body of knowledge with (a) an identification protocol for potential website vulnerability to session fixation attacks, (b) testing and measurement protocols for actual website vulnerability to session fixation attacks, and (c) procedures to help technical and nontechnical users against such attacks.

This paper includes five (5) sections: Section I presents the research background and relevance, Section II describes the research procedure and limitations, Section III presents the results and analysis, Section IV contains the authors' recommendations, and Section V suggests areas for future research

2. Research Procedure and Limitations

This research was implemented using a five-step process.

- One hundred and twenty-five (125) websites on servers under Indonesia domain name were randomly selected via a Google search command using the search term site:id.
- The websites were examined and initially categorized. (a) Each website with a URL and/or cookie containing a string of characters that can represent a session ID was classified as vulnerable. (b) When such a pattern was not found, the website was classified as immune.
- A procedure was created to test website vulnerability. One challenge, which the authors met, was to design an ethical procedure which: (a) does not break into any third-party website account, (b) does not penetrate the website being tested and, (c) ensures that all parties involved do not experience loss or damage of any kind.
- The websites were subjected to session fixation attacks using the designed procedure.

- Each website that succumbed to the attack was classified as vulnerable, while websites impervious to the attack were classified immune. The initial and final classifications were compared. A successful attack pattern was identified in two ways: (a) a server response containing a tailored message (i.e., "123456789") or (b) a server-generated text that is absent prior to a login (i.e., having a link to "Logout").

This research was implemented with two limitations. One is the exclusion of hidden fields which, although not the best way for managing sessions, remains a legitimate technique. The exclusion is for ethical reasons: the launching of a program that modifies values of hidden fields in third-party websites is unethical. Secondly, this investigation is limited to websites on Indonesian servers because:

- Indonesian laws do not specifically ban invasive testing procedures that penetrate websites without permission (Allister, Michael, & Lim, 2009) and
- It is commonly perceived that web programmers in Indonesia focus more on meeting deadlines and budgets than on website security. In addition, frequent reports on compromised Indonesian websites (Wardi, 2010) indicate that many web programmers in Indonesia do not prioritize website security. These two assumptions were not tested.

3. Results and Discussion

This study found six (6) categories of website vulnerability to session fixation attacks (See Table 1). A local attack refers to one that is done by modifying cookies of the victim. A remote attack is done through phishing or cross-site scripting that induces a user to click on a malicious message.
The study identified seven (7) categories of website immunity to session fixation attacks (See Table 2). SID means session ID.

Results of the investigation indicate that almost half (48%) of the tested websites are vulnerable to session fixation attacks. Most of the vulnerability (47.2%) is caused by reusing the same session IDs. This supports the notion that a significant number of programmers in Indonesia do not prioritize website security design.

3.1 Session ID Renewal

The investigation found that, contrary to popular belief (OWASP, 2010) replacing session IDs after logins do not immunize websites to session fixation attacks (category vuln-3). At least 16.8% of the tested websites are immune although these do not renew session IDs (category immune-3 and immune-7).

These immune websites use information in the *referer* header to track the page where the user had just came from and, using that information, block access to particular pages. This indicates that programming websites against session fixation attacks can be done by using the *referer* header.
However, this approach is not recommended because character strings in *referer* headers are limited in length. The Cookie State Management Approach, which allows access through session management, is recommended because it has no limitations on string lengths and, in addition, is scalable.

3.2 Post-Authentication Pattern

Expiring each session ID after log out is a mechanism that is theoretically effective against session fixation attacks. Curiously, 4% of the tested websites that follow this protocol fall under two vulnerability categories (vuln-2 and vuln-4). The authors assume that the programmers of these websites did not use session IDs as part of authentication control.

3.3 Naming Pattern

This investigation found that more than half of the tested websites use default naming to name session IDs (see Figure.1). The names, mostly PHPSESSID (55.83% used PHP), and ASPSESSION ID (60% used ASP) also include common naming conventions such as osCsid (10.83%) and zenid (6.67%).

Since osCsid ID names are generated by the osCommerce software while zenid ID names are generated by zen cart (Zen Ventures, LLC, 2010) an open source ecommerce shopping cart software, this data indicates that the website programmers cut corners and use the default session management functions of the programming platforms they use. This practice allows malicious attackers to access the websites; it may explain the numerous reports of compromised websites in Indonesia.

Since the strength of each session ID determines the success of each session fixation attack, and since session ID regeneration is the programmer's domain, the programmer is responsible for changing the ID for every page in each session (Figure 1).

4. Conclusions and Recommendations

For most people, session management procedures are too complex to easily understand. To help more people learn how to nullify session fixation attacks, this research offers two types of recommendations:

one for programmers, one for technically-skilled users as well as universal users. Session ID names in PHP-based web applications.

4.1 Recommendations for Programmers

Ideally, web servers generate a sequence of session IDs in such a way that the pattern is difficult to predict. This would deter a cracker to predict the subsequent IDs. However session fixation attack does not depend on the randomness of the session ID for the attack to succeed. Instead of relying on this mechanism, the web application programmers must implement techniques to prevent session fixation attacks (Stuttard & Pinto, 2008). There are several ways:

Disallow logins to a chosen session ID: A common strategy of session fixation attacks is to allow users to login using a session ID chosen by the attacker. A program to issue a new session ID after each authentication procedure prevents this. The program must display the association between the old and the new IDs, which an attacker should not be able to read.

Implement session timeout: An attacker may find a pattern of subsequently-generated session IDs. To prevent this, programmers must implement session timeouts by storing a session variable containing the time stamp of the last access of each session ID (Wikipedia contributors, 2010). When the same session ID is used again, the current timestamp (e.g., in PHP use the time () function call) is compared with the timestamp stored in the session. If the difference is more than a certain value, say 2 minutes, the session variable is destroyed or updated to a new timestamp.

There is a race between the session duration and the time it takes for an attacker to find the subsequently generated session ID. Hence the strength of the ID should exceed the time it takes to find the ID's next sequence.

Store the session ID in cookies: Although storing the session ID in cookies is not the best solution to prevent session fixation attacks; this mechanism is more difficult to attack than protocols that store the ID in the URL.

Destroy session if the *referer* seems suspicious: When a user clicks on a web page, most browsers embed information in the *referer* header. This information contains the link that the user followed to arrive at this page. A user request to access a page (i.e., welcome.php) must activate a web application that allows access to the page only if the *referer* header shows that the page is being accessed from a valid page (i.e., login.php).

Accept server-generated session IDs only: One way to ensure security is to accept server-generated session IDs only. The following code can be used for such a purpose (Wikipedia contributors, 2010)

```
if (!isset($_SESSION['SERVER_GENERATED_SID'])) {
   session_destroy();
}
session_regenerate_id();
$_SESSION['SERVER_GENERATED_SID'] = true;
```

4.2 Recommendations for Users

Technically knowledgeable users should check for a session ID within an internet link before clicking on it. When such a link is present, exercise caution about inadvertent submission of credentials by clicking on the links.

Universal users include web users who cannot be bothered to delete cookies, who do not know about session IDs is or how to recognize strings in a URL. Hence, there is a challenge to create detailed "how-to's" that help web users avoid false negatives. For these types of users, the authors recommend that they constantly update their web browsers with the latest patch.

4.3 Recommendations for Future Research

This paper recommends two directions for future research: (1) how a session fixation attacks succeed despite new session IDs, and (2) a cost-effective program design to effectively thwart any session fixation attack.

Despite session ID renewal after logins, session fixation attacks can be launched by predicting succeeding IDs. The recommended research focus is on observing degrees of difficulty in predicting session IDs by devising a metric to measure the difficulty. This difficulty largely depends on the degree of randomness of the selected session IDs. A tool designed by the National Institute of Standards and Technology is a statistical test suite for degrees of randomness, available at http://csrc.nist.gov/groups/ST/toolkit/rng/index.html. Tools such as WebScarab and Burp Suite allows

session ID collection and randomness analysis (Hope & Walther, 2008). These tools, however, do not allow assessment of randomness from combined session management techniques such as hidden fields. To lower the success rate of session fixation attacks, Stuttard & Pinto 2008 suggest a conservative approach: issuing a new session ID for every generated page. However, we would question the scalability and performance of this technique, which serves as the basis of our second recommendation for future research. Szydlowski, Kruegel, & Kirda (2007) describe a technique to bind sensitive information to confirmation tokens to transfer data securely through a compromised host. The authors suggest using this technique to generate a secure session ID, with the end goal of formulating a metric that measures complexity and efficiency.

5. References

Allister, D., Michael, F., & Lim, C. (2009). Eastern cyberlaw exposed: The port scanning way. *ICT Development for The Knowledge Based Society.* Bandung, Indonesia.

Hacker ikut bantu Susno! Rusak 60 situs. (2010, May 17). Retrieved from resep.web.id: http://www.resep.web.id/berita/hacker-ikut-bantu-susno-rusak-60-situs.htm

Hope, P., & Walther, B. (2008). *Web security testing cookbook.* Sebastopol: O'Reilly Media, Inc.

Kolsek, M. (2002, December). *Session fixation vulnerability in web-based application.* Retrieved March 24, 2010, from ACROS Security: http://www.acrossecurity.com/papers/session_fixation.pdf

OWASP. (2010, April). *OWASP top 10.* Retrieved May 1, 2010, from OWASP: http://www.owasp.org

OWASP. (2010, February). *Testing for session fixation (OWASP-SM-003).* Retrieved from OWASP: The Open Web Applications Security Project: http://www.owasp.org/index.php?title=Testing_for_Session_Fixation_%28OWASP-SM-003%29&oldid=78834

Stuttard, D., & Pinto, M. (2008). *The web application hacker's handbook.* Indianapolis: Wiley Publishing, Inc.

Szydlowski, M., Kruegel, C., & Kirda, E. (2007). Secure input for web applications. *23rd Annual Computer Security Applications Conference* (pp. 375-384). Miami Beach: The IEEE Computer Society Digital Library.

Takamatsu, Y., Kosuga, Y., & Kono, K. (2010). Automated detection of session fixation vulnerabilities. *WWW2010* (pp. 1191-1192). Raleigh, USA: The ACM Digital Library.

Wardi. (2010, June 7). Retrieved from maswardi.com: http://www.maswardi.com/2010/06/website-pemda-lampung-tengah-kena-hack.html

Wikipedia contributors. (2010, July). Retrieved from Session fixation: http://en.wikipedia.org/w/index.php?title=Session_fixation&oldid=374643499

Zen Ventures, LLC. (2010). Retrieved from zen cart: the art of e-commerce: http://www.zen-cart.com/

Table 1. Categories of vulnerable websites

Category	**SID sent through**		**After login**		**SID expires after**		**Type of attack**		**Population**
	Cookie	*URL*	*Renew ID*	*New cookie*	*Logout*	*Browser closed*	*Local*	*Remote*	
Vuln-1	√					√	√		40.0%
Vuln-2	√						√		2.4%
Vuln-3	√		√		√	√	√		0.8%
Vuln-4	√			√			√		1.6%
Vuln-5	√			√	√		√		2.4%
Vuln-6	√	√					√	√	0.8%
								Total	48.0%

Table 2. Categories of immune websites

Category	**SID sent through**		**SID sent**	**Renew SID**			**Population**
	Cookie	*URL*	**before login**	*Before Login*	*After Login*	*More Cookies*	
Immune-1	√				√		15.2%
Immune-2	√				√		3.2%
Immune-3	√			√			16.0%
Immune-4	√		√		√		6.4%
Immune-5	√	√			√		4.0%
Immune-6	√					√	6.4%
Immune-7		√					0.8%
						Total	52.0%

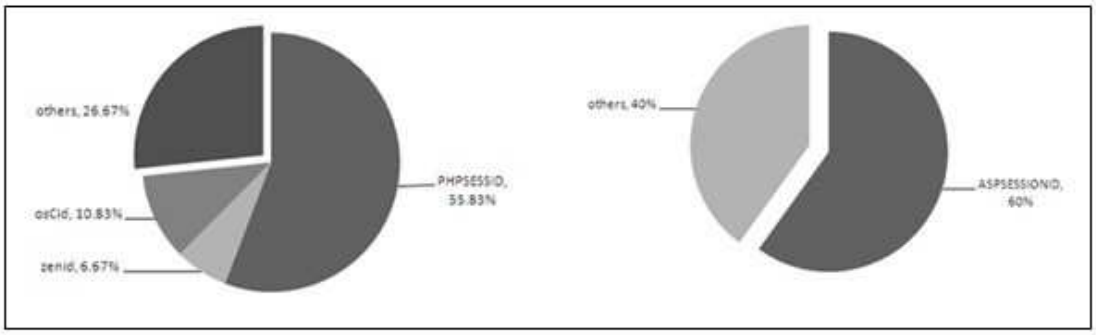

Figure 1. Session ID names in PHP-based web applications (left) and in ASP-based web applications (right)