security issues in php

by Azam Fareed

Submission date: 24-Feb-2024 03:21PM (UTC-0500)
Submission ID: 2175085810
File name: SECURUTY_ISSUES_IN_PHP.docx (21.19K)
Word count: 1785
Character count: 10491

THE FEDERAL POLYTECHNIC, ILARO, OGUN STATE

NAME

KOMOLAFE SAMSON IBUKUNOLUWA

MATRIC NO:

H/CS/22/1034

TOPIC

SECURITY ISSUES IN PHP WEB APPLICATIONS/SOFTWARE

LECTURER'S NAME

MR. AKINODE

FEBRUARY, 2024

ABSTRACT

7

One of the most popular languages for web development is PHP. However, the business logic process, web presentations, and data access code were all mixed together because of the unstructured development style. It perplexes and creates issues for web developers. The PHP framework, on the other hand, offers a fundamental foundation for creating web applications, which aids developers in fostering rapid application development (RAD). Time is saved, redundant code is reduced, and the program can be constructed more steadily. This is because the framework addresses the drawbacks of simple PHP by utilizing the Model View Controller (MVC) approach. The business world has shifted its focus to the Internet in recent years, and as a result, web applications now pose a significant threat to security. The state-of-the-art bug discovery tools for web-application development languages like PHP suffer from a relatively high false-positive rate and low coverage of real errors; this is primarily due to the tools' path-insensitivity and imprecise modeling of dynamic features. In this paper, I will be highlighting security problems in websites and software made with PHP.

1.0 INTRODUCTION

"While PHP and Java are both popular languages for open-source web applications, they possess different security reputations. In this paper, we investigate whether the variation in vulnerability density is greater between different applications written in PHP or between different languages. I compare fourteen open-source web applications written in PHP, including both PHP 4 and PHP 5 code. Despite differences in security reputations, PHP remains a prevalent choice for web development, with more than twice as many open-source web applications written in PHP compared to Java. PHP's security reputation has been influenced by default language features present in earlier versions, such as the 'register globals' feature, which automatically created program variables from HTTP parameters (1). However, these features have gradually been disabled or removed from the language over time. To measure security, we utilize a static analysis tool to identify common secure programming errors in the source code of PHP applications. Static analysis tools offer a repeatable and objective method of evaluating code, making them suitable for comparing different codebases. I also compute code size, complexity metrics, and a security resources indicator metric to examine the source of differences between projects. The structure of the paper includes: Theoretical Background, Related Work, Methodology, Reflection, Conclusion and References

2.0 THEORETICAL BACKGROUND

This chapter presents a few attack scenarios that are pertinent to security problems with PHP online applications and software. These examples cover a range of attack types, including LDAP attacks, PHP and Java's LFI (Local File Inclusion) vulnerabilities, and SQL injection.

2.1 Cross-Site Scripting (XSS)

Three general types of cross-site scripting (XSS) attacks can be distinguished: Stored Cross-Site (XSS), DOM-Based (XSS) and Reflected (XSS).

- **Stored Cross-Site Scripting (XSS):** This kind of attack is inserting malicious code into a website, usually via input fields such as discussion boards or comment sections. The malicious code in this content runs in other users' browsers when they see it, giving the attacker access to their personal data (2).
- **DOM-Based (XSS):** This particular kind of reflected XSS takes use of holes in the browser's Document Object Model (DOM).
- **Reflected (XSS):** This kind of attack involves the attacker sending the server a malicious request that includes JavaScript code. The user's browser then runs this

code after receiving a reflection from the server. This technique can be used by the attacker to take sensitive data, including cookies.

2.2 SQL Injection

SQL injection is a type of attack that targets databases specifically. General web applications utilize databases that are often more likely to be exploited by attackers than XSS vulnerabilities because of their involvement with data storage (3). Alert Logic Cloud Security recently stated that SQL injection accounted for 55% of the detected attacks. The general test method for this vulnerability involves the construction of simple test code in the username and password input boxes, such as:

Username' or '1' = '1

Password' or '1' = '1

This method is often used on online login pages by attackers. Alternatively, vulnerabilities can be mined by adding &id = 1 and id = '1' after the label in the URL bar. The existence of vulnerabilities of this type is conformed if no 404 or 403 warning is displayed and the new page's content is identical to that of the old page. In addition, Boolean blind injection, timestamp injection, truncated injection and other methods can be used to determine the effectiveness of SQL injection on web applications

2.3 LFI

LFI primarily affects web applications written in PHP . When a file is imported through a PHP function, the incoming file name or path is not properly processed. Checking or operating unintended files may lead to accidental file leakage or even malicious code injection (4). The vulnerable functions in PHP are usually include(), require(), include_once(), and require_once(). In JSP/servlet, they are ava.io.File() and java.io.FileReader() functions instead.

2.4 LDAP

LDAP is a lightweight online directory access protocol; its information storage form is depicted. The usage of web applications has increased manifold recently, and the resources and data of these applications are distributed and stored in directories. Usually, different applications involve directories, called proprietary directories, dedicated to their related data. An increase in the number of proprietary directories leads to the formation of information islands (i.e., information systems that cannot interoperate or coordinate work with each other).

This complicates the sharing and management of systems and resources. In this case, some web programs utilize LDAP to facilitate data query and processing, which may make them vulnerable to LDAP attacks (5). LDAP injection is similar to SQL injection—parameters introduced by the user are used to generate LDAP queries, which can be divided into ‘and’ injection and ‘or’ injection, as depicted below:

(&(parameter1 = value1)(parameter2 = value2))

(//(parameter1 = value1)(parameter2 = value2))

If a server opens any of the ports 389, 636, or 3269, it is likely to exhibit LDAP vulnerability. An attacker can exploit this vulnerability to perform blind LDAP injection on test directory attributes and obtain document information (6)

3.0 RELATED WORK

Using a variety of approaches and criteria to evaluate vulnerability density and defect prediction, numerous research have examined security vulnerabilities in PHP websites, applications, and software (7).

Static analysis technologies have been used by Coverity and Fortify to measure the vulnerability density in open-source projects. Fortify examined a broader range of Java programs, whereas Coverity concentrated on C and C++ projects. These studies offer insightful information on the frequency of security flaws in various programming languages and project kinds.

In the context of PHP security vulnerabilities, Nagappan's work on defect density prediction using static analysis methods is pertinent. Even if defect density and vulnerability density might not directly correspond, Nagappan's research emphasizes how crucial it is to evaluate software security using empirical data and metrics (8).

Ozment, Schechter, and Li looked at how security flaws changed over time in different software projects. Their findings underscore the dynamic nature of software security and the necessity of ongoing monitoring and improvement initiatives by implying that the quantity of security concerns may vary.

Shin and Nagappan investigated relationships between vulnerabilities and software complexity metrics like cyclomatic complexity. Their investigations illuminated potential elements impacting the security of PHP websites and applications, even if their results were conflicting and several projects showed only minor correlations.

Dependencies and vulnerabilities in Red Hat Linux packages were examined by Neuhaus and Zimmerman (9). Their research emphasizes how crucial it is to take external dependencies into account when evaluating the security of PHP software because flaws in third-party libraries can seriously jeopardize the system's security posture.

4.0 METHODOLOGY

A multi-layered technique is used to address security vulnerabilities in PHP online applications and software (10). This entails using secure coding techniques while developing new software, carrying out exhaustive security evaluations and penetration tests, routinely applying security patches and updates, keeping an eye out for questionable activity in web application logs, and teaching users and developers about common security risks and best practices.

5.0 REFLECTION

The continuous difficulties and dynamic character of security in PHP online applications and software are brought to light by the careful examination of academic publications (11). Even while efforts to detect and mitigate vulnerabilities have progressed, new threats are always emerging, necessitating ongoing awareness and preventative actions to safeguard against cyberattacks (12). The section on reflection delves into the consequences of the results for developers, security experts, and establishments responsible for safeguarding PHP-based systems.

6.0 CONCLUSION

In conclusion, security for PHP online apps and software is still of utmost importance. This essay has examined related work, discussed approaches for addressing security challenges, identified common security issues, explored theoretical foundations, critically analyzed recent scholarly research, reflected on the findings, and offered recommendations for improving PHP-based system security (13). Maintaining the availability, integrity, and confidentiality of PHP web applications and software requires ongoing study, cooperation, and security measure investments.

REFERENCES

Artzi et al. *Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking*. IEEE Trans. on Soft. Eng., 36(4), 2010.

Balakrishnan, G. Sankaranarayanan, S. Ivancic, F. Wei, O. and Gupta. A. Slr: *Path-sensitive analysis through infeasible-path detection and syntactic language refinement. In Static Analysis, LNCS*. *Springer*, 2008.

Balzarotti, D. et al. Saner: *Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications*. S&P'2008

Biggar, P. and Gregg, D. *Static analysis of dynamic scripting languages,* 2009. Common weakness enumeration. http://cwe.mitre.org/top25/.

Das, M., Lerner, S. and Seigle, M. Esp: *path-sensitive program verification in polynomial time*. In PLDI '02. ACM, 2002.

Dhurjati, D. Das, M. and Yang. Y. *Path-sensitive dataflow analysis with iterative refinement*. In Static Analysis, LNCS. Springer, 2006.

Dillig, I., Dillig, T. and Aiken, A. *Sound, complete and scalable pathsensitive analysis*. *In PLDI '08*. ACM, 2008.

G. Snelting, T. Robschink, and J. Krinke. *Efficient path conditions in dependence graphs for software safety analysis*. ACM Trans. Softw. Eng. Methodol., 2006.

J. Fonseca, M. Vieira, and H. Madeira. *Testing and comparing web vulnerability scanning tools for sql injection and xss attacks*. In PRDC'07. IEEE CS, 2007.

M. Taghdiri, G. Snelting, and C. Sinz. *Information flow analysis via path condition refinement*. In FAST 2010. Springer-Verlag, 2010.

N. Jovanovic, C. Kruegel, and E. Kirda. Pixy: *a static analysis tool for detecting Web application vulnerabilities*. In S&P'06. IEEE, 2006.

Y. Minamide. *Static approximation of dynamically generated web pages*. In WWW '05. ACM, 2005. [13] PHP—Personal Home Pages. http://www.php.net.

Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo. *Securing web application code by static analysis and runtime protection*. In WWW '04, 2004.

security issues in php

ORIGINALITY REPORT

16%	13%	14%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Zhen Yin, Scott Uk-Jin Lee. "Security Analysis of Web Open-Source Projects Based on Java and PHP", Electronics, 2023 Publication	5%
2	www.readkong.com Internet Source	5%
3	d3s.mff.cuni.cz Internet Source	1%
4	James Walden, Maureen Doyle, Robert Lenhof, John Murray. "Chapter 5 Idea: Java vs. PHP: Security Implications of Language Choice for Web Applications", Springer Science and Business Media LLC, 2010 Publication	1%
5	Submitted to Global College of Engineering and technology, Oman Student Paper	1%
6	coek.info Internet Source	1%
7	docplayer.net Internet Source	

		1 %
8	scholarworks.utep.edu Internet Source	1 %
9	publikationen.bibliothek.kit.edu Internet Source	<1 %
10	Lecture Notes in Computer Science, 2010. Publication	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		