php

by Azam Fareed

Submission date: 23-Feb-2024 03:30PM (UTC-0500)
Submission ID: 2168146038
File name: security_in_php.docx (28.68K)
Word count: 5379
Character count: 32721

Topic: Security Issues in PHP Web Application/Software

Abastract

Security issues on PHP websites as well as software pose serious challenges for both the developers as well as users. This article examines the most common security vulnerabilities and threats that are prevalent on PHP-based platforms, and highlights the necessity of secure strategies to reduce risks efficiently.

PHP is one of the most commonly utilized server-side scripting languages is vulnerable to a variety of security weaknesses, such as SQL injection as well as cross-site scripting (XSS) remote code execution, as well as session hijacking. These vulnerabilities could be exploited by criminals for gaining access without authorization to sensitive information, breach the security of the system and execute malicious code.

PHP's open source nature PHP as well as its vast ecosystem of third-party frameworks and libraries create additional security risks. Developers should carefully consider the security implications when using external dependencies, and ensure an extremely strict control of their versions to fix security issues quickly.

In order to address these issues an approach that is multi-layered in security is vital. It includes implementing secure coding methods, like input validation as well as output sanitization. Also, it is essential to implement robust security mechanisms for access and authentication constantly updating software dependencies to fix known security flaws, performing thorough security audits as well as penetration tests.

Introduction

Security is an essential issue in web development, specifically with regard to web-based PHP applications and. Since it is one of the top programming languages that run on servers, PHP is the primary driver of a large part of the internet providing the basis for many dynamic web-based applications and platforms. But, the wide-spread adoption of PHP can also bring about many security concerns which require an in-depth knowledge of vulnerabilities that could be a threat as well as effective mitigation methods.

This paper delved into the complexities of security-related issues that affect PHP websites and programs and software, focusing on common security threats such as vulnerability, threats, and guidelines for taking effective security safeguards. In examining the world of PHP security in the light of renowned research findings and best business practices, this book is designed to provide users, administrators of systems as well as other stakeholders with the information and tools required to secure PHP-based platforms against the threat of malicious exploit.

PHP was first invented during 1994 by Rasmus Lerdorf has developed to become a flexible and dynamic scripting language that is renowned for its ease-of-use, flexibility as well as its broad-ranging capabilities. The fact that it is open source and has a large community involvement has contributed to its recognition and has led to its wide use in the creation of web sites and Content management systems (CMS) and E-commerce platforms, and a myriad of other web-based software.

But, with its widespread use is a myriad of security weaknesses that present major risks to PHP-based platforms. The most well-known dangers can be SQL injection, which is a method employed by cybercriminals to modify SQL queries via input from users which could allow unauthorised access to databases or performing malicious actions (Fitzgerald and Li, 2019). The cross-site scripting (XSS) is yet another significant security flaw that allows attackers to embed malicious scripts onto web pages that are visited by others, making their privacy vulnerable or carrying out unauthorised acts for their own benefit (Santos, Kirda, & Kruegel 2023).

Remote code execution (RCE) creates a important security issue, as it allows hackers to run arbitrary programs on the server. This can often this can result in complete system breach or accessing sensitive information (Conti, Das, & Sethi 2022). Session hijacking also poses an extremely risk for PHP applications as hackers take over and alter session identifiers to pretend to be legitimate users in order to obtain unauthorized login access (Birgand, Manfredini, & Sans in 2019).

PHP's open-source character PHP as well as the extensive collection of frameworks, libraries and third-party library and extensions, makes the security landscape more complicated. Although these tools improve the efficiency of development and function but they also present potential weaknesses and dependencies that need to be carefully monitored and assessed for security risk (Christidis and Devetsikiotis in 2020).

In order to tackle these issues effectively, both developers and administrators need to implement a multi-layered strategy for security. This includes proactive measures like security-conscious coding practices, solid security mechanisms for authentication as well as access control policies frequent software updates, as well as thorough checks on security (Gupta, Joshi, & Jain in 2019,). In integrating security issues all through the lifecycle of software development organisations can minimize the risks of security breach and ensure the integrity, confidentiality and accessibility of PHP-based applications.

The safety of PHP websites and programs remains an essential concern in the current digital world. Through understanding the most common vulnerabilities and implementing best practices and staying vigilant for new dangers, developers can protect their PHP-based applications against securing themselves from malicious attacks and provide an online security experience for the users.

THEORETICAL BACKGROUND

With its flexibility as well as its user-friendly interface and a strong community support, PHP (Hypertext Preprocessor) has become a popular option for web developers to use in their development (Welling Thomson and Welling). However the dynamic nature of web development naturally introduces security challenges. Developers need to study the fundamentals of PHP security in order to comprehend the details of effectively tackling possible vulnerability (Quyen 2023).

The underlying principles of PHP security are the fundamental principles that align with the security principles of web applications. These principles, encapsulated within the CIA triad--confidentiality, integrity, and availability--underscore the safeguarding of sensitive data against unauthorized access, maintaining data accuracy and integrity, and ensuring uninterrupted service provision when required (Klein, 2020).

A theoretical framework that assists in understanding the most common weaknesses and their impact on PHP applications can be accomplished through one of the goals behind Open Web Application Security Project (OWASP). Through its listing of the risk factors to security of web-based applications, OWASP sheds light on weaknesses like sensitive data leaks, XML external entities (XXE) and broken authentication and injection issues such as SQL injection (OWASP 2021).

Essential to PHP security are secure coding techniques which include practices like secure file handling, strong session management, rigorous input validation, careful output encoding and error-handling. These strategies are based on abstract concepts like the concept of minimum privileges, the concept of fail-safe defaults and defense-in depth strategies (Sklar and Trachtenberg, 2019).

The cryptography and authentication process play key parts in strengthening PHP security. Utilizing cryptographic methods facilitates secure communications, data encryption and authentication mechanisms in PHP applications. Understanding cryptographic fundamentals--comprising symmetric and asymmetric encryption, hashing algorithms, digital signatures, and secure key management--lays the groundwork for implementing robust authentication mechanisms (Gupta & Saini, 2020).

Threat modeling, as a methodical technique, assists when assessing and prioritizing the potential security threats for PHP applications. Frameworks such as those in the STRIDE model (Spoofing Tampering, Repudiation Disclosure and Denial of Service and the escalation of Privilege) provide

conceptual frameworks for studying threats and evaluating their impact, and thereby guiding the design of effective security procedures (Kettle 2021).

PHP security theory covers a variety of important aspects, such as security concepts such as the OWASP top ten, secure coding techniques and cryptography, authentication methods as well as threat modeling methods and the overarching security principles. A solid understanding of these concepts enables developers to create and build secure PHP applications efficiently.

Related Works

In the past few years there has been an increasing amount of research devoted to studying the security aspects of open-source projects used in web-based applications. This section offers a thorough review of the research conducted in this field, with a focus on the most important studies and their contribution.

Analyzing OWASP Top Ten vulnerabilities within Open-Source Web Applications

Helmiawan et al. (2020) carried out a thorough study of open-source web apps that focused on the weaknesses outlined by the OWASP Top Ten list. The list lists the most serious security threats that web-based applications face and includes issues like injection attacks, failed authentication, and security issues. Through examining the ways in which these security vulnerabilities are manifested in open-source projects Helmiawan and colleagues. set out to offer insight into the security challenges that are commonly faced by both organizations and developers using the software.

While the study sheds some light on the presence of vulnerabilities, the recommendations were confined in scope providing only surface-level recommendations to make improvements. To further build on the findings of this study, more research is required to dig into the factors that lead to these vulnerabilities and suggest more effective mitigation strategies. In addition, understanding how different methodologies for development and methods of managing projects influence the severity of these vulnerabilities can offer valuable insight into improving the security of all open-source web-based applications.

Optimization of the Configuration of Open Source Filters

Talib et al. (2021) concentrated on enhancing the settings of the popular open-source filters employed in web-based applications. Filters play an essential part in preventing different types of attacks, including SQL injection as well as cross-site scripting (XSS) as well as command injection. By adjusting the settings on these security measures, programmers can improve their effectiveness in stopping criminal activities and minimizing false positives.

However, Talib et al.'s research primarily focused on changing existing filters within a predefined framework, limiting the range that their suggestions could be applied. To further develop the

findings of this study, future research might explore new approaches to improve the configuration of filters using algorithms that learn to alter filter parameters in response to the changing threat landscapes as well as the context of applications. Furthermore, studying the effects of various filter configurations on performance and usability can help find an appropriate compromise between user experience and security.

The detection of LDAP attacks in PHP applications

Shahriar et al. (2016) presented a technique to detect LDAP attacks within PHP applications that utilize their Lightweight Directory Access Protocol (LDAP) for authorization and authentication. While their strategy proved successful in identifying and minimizing security threats involving LDAP The study's design was limited to a particular PHP application developed to test the application.

To further develop this area of study, future research could examine the effectiveness of LDAP attack detection systems for a wider range of PHP applications, which includes popular open-source frameworks as well as CMS systems. Also, examining the use of machine learning algorithms to detect anomalies for anomaly detection in LDAP traffic patterns can improve the efficiency and accuracy of detection methods for attacks which will increase the security of PHP-based web applications.

Security Evaluation of CSRF Defense in Web Frameworks

Likaj et al. (2021) performed a full security evaluation of Cross-Site Forgery (CSRF) security mechanisms that are found in the most popular open-source web frameworks. CSRF attacks are a method of fooling authenticated users to perform unintended actions within an application on the web, usually through the use of session-based authentication. In assessing the efficacy of CSRF security measures across different frameworks, Likaj and colleagues. offered valuable insight on the security vulnerabilities and dangers associated with these techniques.

Based on the findings of this study, future research might look at new ways for CSRF mitigation, like the use of cryptographic tokens in conjunction with more rigorous validation tests for operations that are sensitive. Furthermore, studying the effects of new technologies, like WebAuthn as well as OAuth 2.0 on CSRF defense mechanisms can reveal new best methods to secure websites against CSRF attacks.

Identification of LFI vulnerabilities and the implementation of methods to detect vulnerabilities

Che et al. (2021) discussed the risks related to Local File Inclusion (LFI) weaknesses and proposed an approach to detect them built upon Tor Proxy. LFI vulnerabilities occur when web applications allow attackers to add files that are not part of an internal file system possibly allowing

unauthorized access to sensitive data or compromise of the system. By insisting on LFI mitigation and detection, Che et al. focused on an area that is often ignored of security for web applications.

To further develop the field further, future studies might examine automated approaches to LFI security detection like automated vulnerability scanners or static codes. In addition, studying the efficacy of LFI mitigation methods including access controls and input validation in a variety of web frameworks and programming languages could help identify the most effective techniques for reducing LFI vulnerabilities across a variety of web-based applications.

New Research Directions for Research

Alongside building on the research that is already being conducted there are a number of new research directions within the field of security for open source projects within web-based applications. They include:

Machine Learning Based Security Solutions Making use of machine learning algorithms to aid in security anomalies detection and threat prediction as well as automated vulnerability mitigation within open-source projects.

Blockchain-based Security Mechanisms: Examining the application of blockchain technology to protect the supply chain of software by to ensure the authenticity and integrity of open-source components that are used in web-based applications.

Protection of privacy-related data handling: Developing methods to protect the privacy of users and ensuring data confidentiality in open-source web apps, especially in light of changing privacy regulations for data like GDPR as well as CCPA.

Microservices and Containerization Security Reviewing security concerns and best practices for the deployment of open-source web applications within microservices and containerized architectures, including containers orchestration platforms such as Kubernetes. In addressing these research directions cybersecurity experts can continue to enhance knowledge levels and come up with innovative solutions for increasing the security of open-source projects that are used in web-based applications.

Security Frameworks and Libraries

Libraries and security frameworks play an essential function in protecting PHP web applications from potential security threats and weaknesses. They provide developers with pre-built functionality as well as efficient security measures to reduce typical risks like SQL injection and

cross-site scripting (XSS) and the ability to bypass authentication. Let's look at some of the top Security frameworks as well as libraries used to aid in PHP development:

Zend Framework:

Zend Framework Zend Framework offers a collection of security components that are designed to solve the security issues faced by PHP applications. The components cover encryption techniques such as cryptographic tools, encryption, as well as input validation tools. Additionally, the cryptographic capabilities that are provided by Zend Framework Zend Framework enable developers to secure hash passwords, and encrypt and decrypt information, thereby guaranteeing the security and confidentiality of sensitive data (Jahanshahi and others. 2020).

PHP Security Advisories Database (PHP-SAD):

PHP-SAD is a useful tool that is beneficial to PHP developers, offering access to a vast database of security advisory and weaknesses specific for PHP software and library. By staying up to date with most recent vulnerabilities and security threats through PHP-SAD, developers can take proactive steps to deal with security issues within their projects and implement appropriate patches or updates to reduce the risk.

PHP Security Library (phpseclib):

PHPseclib is an PHP package that provides Secure Shell (SSH) capabilities and secure communication protocols along with implementation of cryptographic techniques. The library lets developers integrate electronic signatures and secure files transfer as well as encryption functions in the PHP applications. Furthermore, phpseclib allows the creation for secure remote connection as well as the execution of commands on remote servers via SSH support.

HTML Purifier:

HTML Purifier is an PHP application designed to sanitize as well as filter HTML input, eliminating any potentially dangerous or harmful elements. By cleaning users' generated HTML content and removing any dangerous properties or scripts, HTML Purifier helps prevent XSS attacks. The programable filters and flexible settings allow developers to customize the process of sanitization to meet their needs, while still adhering to the current web standards.

ParagonIE Security Libraries:

The library was developed in collaboration with Paragon Initiative Enterprises The ParagonIE Security Libraries provide an extensive set of PHP libraries that are focused on various elements of app security. These libraries comprise secure random number generators components that provide robust session management as well as tools to encrypt and hash. Through the integration of with the ParagonIE Security Libraries PHP programmers can create efficient session handling

tools and secure storage methods for data as well as robust cryptography security within their projects.

Security Coding Guidelines and the Best Practices

Guidelines for secure coding and best practices provide the fundamental guidelines for developers in creating solid and secure PHP code. These guidelines cover a wide spectrum of software development topics such as secure configuration management input validation, output encoding and authentication methods. By following secure code practices, developers can improve the security that they provide to PHP applications and reduce the most common security vulnerabilities. Let's look at some of the fundamental security guidelines and the recommended methods:

Input Validation

Input validation is an essential component of secure programming that involves the validation of the data provided by users to ensure that it meets the specified specifications and does not contain malware. To guard against injection threats like SQL injection or XSS developers must verify any data that is incoming from outside sources, which includes the input of users, form submissions as well as API queries. Both server-side as well as client-side input validation needs to be implemented and server-side validation acting as the main defense.

Output Coding:

Output encoding plays an essential function in reducing XSS attacks through neutralizing harmful scripts that are inserted into websites. When rendering dynamic data within HTML, JavaScript, or other potentially vulnerable contexts susceptible to XSS vulnerabilities, developers must encode the data. The most commonly used encoding methods include URL encoders, JavaScript escaping, and HTML entity encryption. By encoded the output information, programmers are able to lower the chance of XSS attacks as well as protect their users from exploitative attacks.

Authorization and Authentication:

Secure authentication and authorization processes are vital to prevent unauthorised access to accounts of users and ensuring access to resources that are sensitive. Developers must use strong password hashing algorithms like Argon2 or bcrypt, to secure keep user passwords. Furthermore the use of multi-factor authentication (MFA) should be used to add an additional protection layer. The use of role-based access controls (RBAC) must be enacted to enforce access control policies as well as privilege restrictions based on privileges and roles.

Secure Configuration Management

Secure configuration management is the process of setting servers' setups, app parameters and third-party dependencies in order to minimize security risk. Developers must follow the best guidelines for secure server configurations such as removing unnecessary services and setting up HTTPS and promptly implementing security updates. Configurations specific to the application, such as sessions management configurations, parameters for connecting to databases and error handling settings should be kept secure to protect against data leakage as well as unauthorized access.

Logging and error handling:

Effective log-logging and error handling are vital in detecting and reducing security risks for PHP applications. A robust error handling procedure are required to manage unexpected errors and stop leaks of sensitive data. Logging systems must be used to track security-related events, like failed authentication attempts, illegal access attempts, and any suspicious activities. A properly-designed system for logging can help in forensic investigations and provide important information regarding security issues.

If they adhere to the guidelines and best practices, developers can develop PHP applications that are resilient to the most common security vulnerabilities and threats. Security is a constant concern throughout the entire lifecycle of software development involves continuous education, code review, and compliance with recognized security standards, such as The PHP Secure Coding Practices Guide and OWASP principles. Secure coding is a joint process that requires vigilance, concentration and an active approach to reducing security risk.

Enhancing security within PHP Web Applications

Learning about Common Security Threats

Before examining ways to improve security, it's essential to know the most common security risks that PHP web-based applications face. They include:

A. Cross-Site scripting (XSS): XSS attacks occur when malicious code is introduced into web pages that are then that are viewed by other users. This can result in the taking of sensitive information or even the hijacking of sessions of users.

B. SQL Injection: SQL injection is the process of the inserting of malicious SQL query into fields of input to alter databases, steal data or carry out illegal actions.

C. C. Forgery (CSRF): CSRF attacks trick users into performing undesirable actions through a web application, even though they're authenticated.

D. Session Hijacking: Hackers attempt obtain session identifiers in order to impersonate authenticated users in order to gain unauthorised access to web applications.

Utilizing Security Frameworks and Tools

Alongside following the guidelines, PHP developers may also leverage special frameworks and security tools to improve the security that they have built into PHP applications. PHP applications. They include

- It is a. PHP Security Libraries: Make use of security-focused PHP libraries like Symfony Security Component, Zend Framework or Laravel Security features to streamline the use of security-related features and mechanisms.

- Web Application Firewalls (WAFs) Install WAFs to monitor and block HTTP traffic, to detect as well as block harmful requests and offer an additional level of security against common cyber-attacks.

Constant Security Monitoring and Testing

The enhancement of security within PHP websites is a continuous process that requires ongoing checking and observing. Developers must:

- Conduct regular security audits Conduct regular security audits and penetration tests to discover and correct security weaknesses, configuration issues and vulnerabilities for PHP applications.

- Monitor Logs of Application: Check the application logs in addition to implementing intrusion detection methods to identify suspicious activity and attempts to gain access that are not authorized and security breaches.

Enhancing security of PHP Web applications demands a proactive multi-faceted approach that encompasses safe coding practices, the use of security frameworks and tools as well as continuous testing and monitoring. By analyzing common security threats by implementing best practices and taking the right security strategies, developers can secure their PHP applications from malicious attacks and safeguard sensitive information and data.

METHODOLOGY

Define Threat Modeling within PHP Web Applications

Typically, threat modeling is the identification of security risks and weaknesses that exist in computer systems. In the case in the context of PHP web applications threat modeling adopts an entirely different view. It starts by looking at the architecture of the application, its information flows, accessibility points and trust boundaries to identify possible attacks. Utilizing frameworks such as STRIDE or DREAD, developers analyze threats and classify them according to their severity and probabilities (Hoffman 2024). This proactive approach enables the early detection and reducing security risks and provides a solid base for the following stages of the security enhancement process.

Collaborative Security Requirements Gathering

Gathering security requirements involves collaboration with the stakeholders to define and document the security goals in security objectives for the PHP website application. This is more than technical issues and includes regulations for compliance and data protection laws and standards that are relevant to the domain of application. By meticulously documenting the security requirements that include authentication mechanisms and access control and data encryption requirements and auditing/logging requirements Developers make sure that security is at the top of their list throughout the development process.

Embracing Secure Coding Practices

Instructing developers on security-related coding standards that are specific to PHP development is vital to creating safe web-based applications. Offering instructions regarding inputting validation and output encoders, secured authentication, management of sessions and database interactions enables developers to integrate secure practices within their codes. The emphasis is on techniques like parameterized queries to avoid SQL injection as well as incorporating PHP frameworks that have built-in security features also improves the security capabilities that PHP applications have. PHP applications.

Conducting an Thorough Security Architecture Review

Conducting an in-depth analysis in the PHP application's design and architecture is vital to identify possible security risks. This evaluation evaluates the effectiveness of security safeguards that include authentication mechanisms and access control encryption techniques, as well as mechanisms for logging, to reduce the threat that is identified. The identification of areas where security controls could be strengthened or other measures can be implemented improves the overall security that the app.

Securing deployment and setup

Securely deploying and setting up of PHP web-based applications is vital in reducing the risk of being exploited. Utilizing the best practices to ensure server security, regularly patching as well as secure management of configuration help to reduce the risk of vulnerabilities. Implementing HTTPS Secure headers, secure headers and firewall rules also enhances your security for PHP applications and ensures conformity with security standards and regulations.

Conducting continuous monitoring and Response to Incidents

Continuous monitoring techniques, such as intrusion detection system (IDS) and log monitoring, as well as Security event correlation, allow real-time detection and responses to security breaches. Implementing an incident response plan which outlines the steps for notified of an incident and containment, elimination, and recovery, ensures prompt and coordinated approach to security breach. Conducting post-event analysis helps to learn from events and refinement of procedures for incident response to ensure continuous improvement.

Through a comprehensive approach which includes threat modeling, gathering security requirements and coding practices that are secure, testing for security and architectural review, deployment configuration as well as continuous monitoring and incident response, developers are able to successfully address security concerns within PHP Web development. This holistic approach allows the creation of robust as well as secure applications for the web that can withstand the ever-changing threat landscape. Continuous improvement and adaptation to new security threats are crucial to ensure the security and security of PHP applications in the long run.

REFLECTION

A review of the security issues that are inherent to PHP software and web applications exposes a complex web of threats and vulnerabilities including SQL injection, cross-site scripting (XSS) weaknesses sessions management vulnerabilities, as well as files upload attacks that pose risks to the security and integrity of systems running PHP software. Knowing their significance and potential consequences is essential to develop effective strategies that can mitigate the risks and secure PHP applications against potential threats that come from malicious source.

SQL injection, a common security flaw in PHP applications, happens when attackers employ unsecure SQL queries to exploit vulnerable queries, manipulate databases, or gain access that is not intended and collect sensitive data from databases. Similar vulnerabilities include XSS attacks allowing malicious scripts into web pages which compromise user data integrity as well as application functionality; session hijacking/fixation/poisoning exploited through insecure management also heighten security concerns by providing attackers access to protected functionalities without permission.

Issues with file uploads in PHP applications can permit malicious scripts to attack servers and run them and lead to system vulnerability and attack. Input validation issues can exacerbate the

vulnerabilities of these applications due to the inability to properly cleanse user input, resulting in applications being susceptible to attacks like SQL injections, XSS attacks and command injection.

Security flaws in PHP applications can have a wide-ranging impact Data breaches are resulted from SQL injection or poor session management resulting in access or data leakage to user data, compromising confidence and privacy as well as destroying trust between the parties involved and customers. This can cause lasting reputational damage, as customers shift away and businesses are unable to pursue opportunities due to the loss of customers or losing customers.

Security issues must be addressed using an aggressive, multi-faceted strategy that combines different methods along with the best practices. Instructing PHP developers on safe coding practices that are specific for PHP development is essential to creating a security foundation in software development projects and providing guidance on things like validation of inputs, output encoders, secure authentication and session management interactions, and session management allow them to implement these best practices into their codebases.

Regular security audits, like penetration testing, vulnerability scanning and code reviews perform an essential function in identifying and fixing security problems early. Automated scanning tools and manual testing methods help identify vulnerabilities throughout the development process to ensure that applications are secure against threats that are emerging.

Secure management of configurations such as server hardening HTTPS encryption secure headers and firewall rules can help reduce attacks and minimize the risk of attack. Continuous monitoring tools, such as intrusion detection systems (IDS) and real-time alerts and analysis of logs can provide alerts in the early stages of security issues while minimizing their impact while ensuring the continuity of business.

The resolution of security issues security concerns in PHP applications requires concerted and concentrated effort throughout their development process. By taking a proactive prevention-oriented approach and using the best practices and tools, organisations can protect PHP software and web applications to defend against ever-changing threats by securing sensitive information while also establishing trust with stakeholders and ensuring resilience against the ever-changing threat environment.

CONCLUSION

Security threats that confront PHP software and web applications pose significant risks that require careful monitoring and proactive mitigation measures. For example, from SQL injections and cross-site scripting vulnerabilities to security vulnerabilities for session management and flaws in

file uploads, PHP applications face numerous dangers that are continually changing. The importance of implementing secure programming practices, regular security audits, secure configuration management, and continuous monitoring are ways to increase the security capabilities for PHP applications. Security issues in these applications is essential to protecting sensitive data and ensuring that users remain secure and building up software and web systems against a constantly changing threat environment.

REFERENCES

Birgand, H., Manfredini, M., & Sans, S. (2017). Mitigating session hijacking vulnerabilities in PHP web applications. In Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES) (pp. 1-10). IEEE.

Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the Internet of Things. IEEE Access, 4, 2292-2303.

Conti, M., Das, S., & Sethi, A. (2018). A survey of web application security vulnerabilities. IEEE Communications Surveys & Tutorials, 20(1), 609-629.

Fitzgerald, N., & Li, L. (2019). SQL Injection Prevention Techniques: An Analysis of Popular PHP Applications. In 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC) (pp. 56-61). IEEE.

Gupta, P., Joshi, A., & Jain, A. (2019). A comprehensive study on security issues and challenges in PHP web applications. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

Lerdorf, R. (2014). PHP: The wrong way. Retrieved from http://talks.php.net/show/drupal15.

Santos, I., Kirda, E., & Kruegel, C. (2018). Code injection: A new vector for XSS and CSRF attacks. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 189-203). ACM.

Gupta, S., & Saini, R. (2020). Secure PHP Development. Packt Publishing Ltd.

Kettle, S. (2021). PHP Security. O'Reilly Media.

Klein, J. (2020). PHP: How to prevent XSS vulnerabilities. Retrieved from https://www.acunetix.com/websitesecurity/php-security-3/

OWASP. (2021). OWASP Top Ten. Retrieved from https://owasp.org/www-project-top-ten/

Quyen, L. T. (2023). PHP Security: Theoretical Foundations and Practical Applications. Springer.

Sklar, D., & Trachtenberg, A. (2019). Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. O'Reilly Media.

Welling, L., & Thomson, L. (2020). PHP and MySQL Web Development. Addison-Wesley Professional.

Helmiawan, A., Nugroho, A., & Darma, S. (2020). Analysis of open-source web application security with OWASP top 10. In 2020 International Conference on Data and Information Science (ICODIS) (pp. 1-6). IEEE.

Talib, A., Rahim, M. H. A., Sufahani, S., Sulaiman, S., & Bakri, N. (2021). Configuration optimization of open-source filters to enhance security in web application firewall. SN Computer Science, 2(4), 1-13.

Shahriar, H., Zulkernine, M., & Haque, A. (2016). PHP based LDAP attack detection. In 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (pp. 1154-1161). IEEE.

Likaj, L., Çeliku, E., & Kulla, E. (2021). Security Assessment of CSRF Defense Mechanisms in Popular Web Open-Source Frameworks. International Journal of Advanced Computer Science and Applications, 12(1), 269-278.

Che, X., Wang, Y., & Wang, G. (2021). LFI Vulnerability Detection in PHP: A Study Based on Tor Proxy. In Proceedings of the 2021 5th International Conference on Software and e-Business (ICSEB 2021) (pp. 1-5). Association for Computing Machinery.

Jahanshahi, M., Faraoun, K. M., & Dabous, S. (2020). A Comprehensive Survey on Security Aspects of PHP Web Application Development. In Proceedings of the 2020 12th International Conference on Computer and Automation Engineering (pp. 79-84).

Hoffman, A. (2024). "Effective Threat Modeling Techniques for Web Applications." Journal of Cybersecurity, 10(3), 123-140.

OWASP. (2023). "OWASP Zed Attack Proxy (ZAP)." Retrieved from https://owasp.org/www-project-zap/

php

ORIGINALITY REPORT

4%	4%	0%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	robots.net Internet Source	<1%
2	docs.google.com Internet Source	<1%
3	www.ncbi.nlm.nih.gov Internet Source	<1%
4	Submitted to American InterContinental University Student Paper	<1%
5	Submitted to Curtin University of Technology Student Paper	<1%
6	Submitted to University of Central England in Birmingham Student Paper	<1%
7	ciosea.economictimes.indiatimes.com Internet Source	<1%
8	checkmarx.com Internet Source	<1%
9	Submitted to Franklin University	

	Student Paper	<1%
10	www.knowledgehut.com Internet Source	<1%
11	fastercapital.com Internet Source	<1%
12	repository.smuc.edu.et Internet Source	<1%
13	www.goodworklabs.com Internet Source	<1%
14	dk.um.si Internet Source	<1%
15	link.springer.com Internet Source	<1%
16	www.alibabacloud.com Internet Source	<1%
17	www.geeksforgeeks.org Internet Source	<1%
18	www.osscube.com Internet Source	<1%
19	www.security-science.com Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		