**SECURITY ISSUES IN PHP WEB APPLICATIONS/SOFTWARE**

**ABSTRACT**

This study conducts a survey among Nigerian IT professionals regarding security flaws in PHP applications. We specifically targeted 170 IT professionals working for different Nigerian ICT companies and institutions. The questionnaire served as the primary data collection tool. A stratified random technique was employed to choose participants from different strata. A total of 170 specialists were chosen at random from the sample, comprising 20 ICT lecturers, 55 MIS department employees from higher education, and 95 ICT organization employees. Basic frequencies and percentages were used to analyze the questionnaire data that was gathered. A greater proportion of respondents were found to be familiar with and to regularly use PHP. The majority of respondents also highlighted SQL injection and source code disclosure as the two main causes of security vulnerabilities. A significant proportion of the participants reported experiencing difficulties with database hacking. They recommended that developers implement all necessary security measures and functions to safeguard the application, such as turning off global directives, utilizing escape functions, and so on. In comparison to other applications, PHP scores highly in terms of portability and accessibility, security, and ease of use (Oyemade, O., Odiagbe, J. O., & Buhari, B. A. 2015).

PHP web applications must be secured in order to avoid common vulnerabilities. A number of problems, including SQL injection, XSS, CSRF, and insecure file uploads, can jeopardize user privacy and data integrity. Prepared statements, input validation, and the use of security features like anti-CSRF tokens and secure session handling are all part of mitigation. Dependencies must also be taken care of, strong password management must be ensured, and appropriate logging and monitoring must be put in place. PHP web applications are kept trustworthy and legitimate by conducting regular code audits, adhering to security best practices, and using tools like web application firewalls to build a strong defense against changing threats.

**INTRODUCTION**

Web applications, in particular, are used today as front ends to many security critical systems (e.g., home banking and e-commerce), but due to their high exposure, they are particularly vulnerable to being heavily attacked or becoming attractive targets for attackers due to the large degree of authority they possess, their large user populations, and the prevalence of vulnerabilities they contain. This means that they require special attention to ensure their security and resilience to these threats. PHP is one of the two most popular development frameworks for dynamic web sites. Popular web development tools like WordPress, Joomla, and Drupal are built on top of it as well. Therefore, there is an urgent need to secure PHP web applications. 60% of vulnerabilities that are discovered actually impact web applications. In order to improve the security of these banking and e-commerce systems, such as security enhanced online registration prepaid scratch payment approach, a lot of attention in the security research community has been paid to eliminating or rationalizing the impact of these vulnerabilities. Certain web applications have security flaws that could lead to the theft of private information, compromise data integrity, or interfere with the functionality of the web application. In this study, IT professionals in Nigeria are surveyed regarding security flaws in PHP applications. In different ICT organizations and institutions throughout Nigeria, we aim to reach 170 IT professionals. Questionnaires were the primary tool used to gather data. Via the use of these questionnaires, these IT specialists were able to identify the root causes of security flaws in PHP as well as the difficulties they are having with security and the methods they are using to solve them. In addition, to rank the PHP in relation to other applications, security, ease of use, and portability and accessibility. The study employed a stratified random method to select participants from different strata. The sample consisted of 170 specialists, of which 95 were randomly selected from ICT organizations, 55 were selected from higher education institutions' MIS departments, and 20 were selected as ICT lecturers. Simple frequencies and percentages were used to analyze the questionnaire data.

Web applications, in particular, are being used today as front ends to many security critical systems (e.g., home banking and e-commerce), but due to their high exposure, they are particularly vulnerable to being heavily attacked or becoming attractive targets for attackers due to the large degree of authority they possess, their large user populations, and the prevalence of vulnerabilities they contain. This means that they require special attention to ensure their security and resilience against these threats. PHP is one of the two primary development frameworks for these dynamic websites. It serves as the foundation for popular web development applications such as Drupal, Joomla, and WordPress. Therefore, it is imperative that PHP web applications be secured. Indeed, web apps are impacted by 60% of vulnerabilities that are discovered. Because of this, the security research community has given a lot of attention to mitigating the impact of these vulnerabilities and strengthening the security of these banking and e-commerce systems, such as through the use of security-enhanced online registration and prepaid scratch payment methods (Gupta, B. B., & Quamara, M. 2019).

It is impossible to exaggerate the importance of PHP in web programming. With the help of its server-side scripting features, developers can produce dynamic and engaging online content. Unfortunately, a number of security issues are brought up by this very dynamism. The power and potential of PHP contrasted with the vulnerabilities that require careful attention is best captured in the first chapter, which also highlights this dual nature. After recognizing PHP's widespread use in the digital world, the story proceeds. Its widespread adoption can be attributed to its open-source nature and the vibrant community surrounding it. But this popularity also draws malicious actors looking to take advantage of vulnerabilities for their own evil intentions, in addition to eager developers. Setting the scene, the introduction acknowledges that PHP's widespread use has two sides: it can be a great thing for innovation but also a potential liability when it comes to security (Snyder, C., & Southwell, M. 2005).

The identification of the main security issues that PHP web applications face is a central theme of the introduction. Every challenge is presented as a crucial aspect requiring investigation, ranging from SQL injection and Cross-Site Scripting to the subtle nuances of session management and file upload vulnerabilities. For the reader, this list acts as a road map as they navigate the complex web of security issues that come with developing PHP applications. The introduction's ethical considerations are interwoven throughout, emphasizing the duty developers have to protect user data privacy and confidentiality. The moral obligation to safeguard against unauthorized entry and possible breaches emerges as a recurrent theme, highlighting the relationship between moral behavior and security (Chatterjee, S., et al., (2015).

Furthermore, the introduction emerges as an exhortation to keep learning new things constantly. Staying up to date with the latest advancements, security best practices, and emerging attack vectors is imperative due to the constantly changing nature of cybersecurity threats. The focus on continuous learning gives the introduction's story a forward-looking aspect by placing the investigation of PHP web application security in the framework of a journey that is dynamic and ever-changing (Abrahams, T. O., *et al.,* (2024).

The introduction discusses the delicate balance that must exist between user experience and strict security measures in its pursuit of balance. A thoughtful approach to the trade-offs and considerations that developers must make in the pursuit of a comprehensive online application is prompted by the recognition of the tension between smooth functionality and strong security as an inherent challenge.

Cross site scripting (XSS) vulnerabilities are among the most prevalent security issues in web applications because they allow malicious scripts to be injected into them. Because it opens the door for other sorts of attacks like session hijacking and Cross Site Request Forgery (CSRF), XSS is selected as the main threat for web applications. Users and website owners alike may suffer consequences from cross-site scripting (XSS). Although it is simple to exploit, it is challenging to counter. Numerous approaches have been put forth to detect them. Still, there is a problem with XSS vulnerabilities in web apps. Most researchers have used hybrid, static, and dynamic studies to find XSS vulnerabilities. But their results are tainted by the false positive rate and the different difficulties in identifying XSS vulnerability. Thus, genetic algorithms made their foray into the field of software testing by producing test cases to examine software security. XSS vulnerability can be identified with a reduced false positive rate using this type of technique. The whole XSS vulnerability in the Java web application framework is found by the genetic algorithm, with no false positive rate in the results. In contrast, several false positive rates are displayed in the PHP web application after the algorithm has been run. The reason for the significant number of false positive results is that the researchers did not exclude the impractical pathways, which would not function at all in the CFG (Juarez, M., et al. (2014).

**THEORETICAL BACKGROUND**

In order to protect PHP web applications from potential risks, it is necessary to comprehend and put important theoretical concepts into practice. Guarding against SQL injection is essential because it prevents unauthorized database queries from being executed by attackers who take advantage of weaknesses in user inputs. By using prepared statements and parameterized queries, this is lessened and user input is thoroughly verified and cleaned (Chen, F., & Roşu, G. (2007).

Another issue is the injection of malicious scripts into online pages, known as Cross-Site Scripting (XSS). In order to mitigate this, programmers employ tools such as htmlspecialchars() to cleanse user input and stop malicious scripts from running. An additional line of security against XSS attacks is provided by the implementation of Content Security Policy (CSP) headers.

Forms can be made to prevent Cross-Site Request Forgery (CSRF) by using anti-CSRF tokens. By verifying that requests come from the anticipated source, these tokens stop users from acting in an unlawful manner. A strong defense against session-related assaults is also bolstered by secure session management techniques like session regeneration and suitable timeout settings.

The risk posed by file upload vulnerabilities is the possibility of arbitrary file execution or attack entry points. Developers store uploaded files outside of the site root, evaluate file sizes, and restrict file kinds. Two other ways to improve security are file type verification and secure file permissions (McDonald, M. (2020).

When sufficient authorization checks are not performed, Insecure Direct Object References (IDOR) may happen. For sensitive data to be protected from illegal access or alteration, strong access controls, rigorous user authorization management, and comprehensive validation are essential. Reliability issues must be addressed. A safe environment necessitates the maintenance of current versions of third-party libraries, which can be introduced by using obsolete or insecure ones. Tools such as Composer can help with this. One such potential weakness is poor password management. By imposing complexity restrictions, employing strong password hashing algorithms (like bcrypt), and teaching users on how to create and manage secure passwords, secure practices are implemented.

The identification of security incidents may be hampered by inadequate recordkeeping and monitoring. The timely identification of odd actions and the establishment of regular monitoring procedures facilitate the implementation of complete logging of security-relevant events and enable the rapid reaction to potential threats. Validation and sanitization of data are also essential components of a theoretical framework. It is imperative to verify and cleanse all incoming data from external sources, including as cookies, query parameters, and form inputs, in order to avert injection attacks and other security flaws.

All things considered, a comprehensive strategy is needed to secure PHP web applications based on their theoretical foundation. The proactive defense against potential threats includes doing routine code audits, keeping up with emerging security best practices, and using tools like web application firewalls (WAFs). In the always changing field of web security, this multipronged approach guarantees the availability, confidentiality, and integrity of PHP web applications (Scambray, J., Shema, M., & Sima, C. (2006).

**RELATED WORKS**

One way to classify the XSS vulnerability as one of the top web application vulnerabilities is to look at the Open Web Application Security Project's (OWASP) 10 leading vulnerabilities rankings. Shar and Tan used Java web apps to test the static analysis methodology. They discovered an XSS vulnerability that produced a lot of false positives. Many times, using static analysis has a significant false positive rate. Static analysis was used by Shar et al. to address the nodes, and dynamic analysis was used to identify the vulnerable nodes but the lack of accuracy in the dynamic analysis results and the false positive rate of the static analysis taint their hybrid methodology. The genetic algorithm was used by Hydara et al. to create test cases for the static analysis. Finding the Java source code's concrete XSS vulnerability was the goal. They were able to identify all of the real vulnerable routes in the Java framework and their methods lowered the false positive rate. Andrea and Mariano suggested a method to find the reflected XSS vulnerability in the PHP web application without eliminating it.

In order to identify all three types of XSS vulnerabilities, Moataz improved this process even more. Without making the most of the genetic mutation operator, Andrea and Mariano's research seeks to identify just reflected XSS vulnerabilities. However, by using the database of XSS patterns to identify the likely XSS vulnerabilities—stored, reflected, and DOM-based XSS—the moataz and Fakhreldin methodology improved upon the one provided by Andrea and Mariano. They only used reflected and stored XSS vulnerabilities in their research, though. Its technique is also limited because some of the CFG's paths are considered unfeasible because they perform poorly or not at all.

As well as best practices and mitigation techniques, the following linked works, articles, and resources go into further detail about security vulnerabilities in PHP web applications:

1. **OWASP PHP Security Cheat Sheet:** The OWASP PHP Security Cheat Sheet is a valuable resource provided by the Open Web Application Security Project (OWASP). It serves as a comprehensive guide for developers to secure their PHP applications. The cheat sheet covers a wide range of topics, including input validation, output encoding, authentication, session management, and more. Each section provides detailed explanations, examples, and recommended best practices. The OWASP PHP Security Cheat Sheet emphasizes the importance of secure coding practices to mitigate common vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). By following the guidelines outlined in this cheat sheet, developers can significantly enhance the security posture of their PHP applications.
2. **PHP: The Right Way - Security Section:** "PHP: The Right Way" is a community-driven guide that aims to provide best practices and up-to-date information on various aspects of PHP development. The security section of this guide focuses on essential security considerations for PHP developers. It covers topics like data validation, secure password handling, secure file uploads, and more. The guide encourages developers to follow industry best practices, adopt secure coding techniques, and stay informed about security updates in the PHP ecosystem. It emphasizes the importance of understanding and mitigating security risks throughout the development lifecycle.
3. **Securing PHP: A Guide for the Insecure:** Authored by Chris Shiflett, "Securing PHP: A Guide for the Insecure" offers practical insights into securing PHP applications. The guide addresses common security vulnerabilities and provides actionable recommendations for developers. Topics covered include input validation, session management, and secure coding practices. Chris Shiflett's guide is particularly beneficial for developers seeking hands-on advice and real-world examples. By following the recommendations in this guide, developers can enhance the security robustness of their PHP applications and reduce the risk of exploitation.
4. **Web Application Security Best Practices (PHP):** The DigitalOcean community tutorial on web application security best practices, with a focus on PHP, provides a comprehensive overview of security considerations for PHP developers. It covers topics such as secure coding practices, input validation, and protection against common web vulnerabilities. This resource is well-suited for developers who want a holistic understanding of web application security, with practical tips and guidance specific to PHP development. It encourages developers to implement security measures from the early stages of application design and development.
5. **PHP Security Guide by SANS Institute:** The SANS Institute's PHP Security Guide offers in-depth insights into securing PHP applications. The guide covers a wide range of security topics, including secure coding practices, input validation, and common pitfalls to avoid. SANS Institute is a reputable organization in the field of information security, and its PHP Security Guide provides authoritative guidance for developers aiming to strengthen the security posture of their PHP applications.
6. **PHP Security Checklist:** The PHP Security Checklist, available on GitHub, serves as a practical resource for developers to ensure they cover essential security measures in their PHP applications. The checklist includes items related to input validation, secure coding practices, and configuration security. Developers can use this checklist as a reference guide during the development and deployment phases to identify and address potential security issues systematically.

**METHODOLOGY**

Descriptive survey research design was used in this study. It is a technique for gathering data in which a sample of people are given a questionnaire. The purpose of descriptive surveys is to gather data regarding a phenomenon's present state or to provide answers to inquiries such as where, what, how, why, when, and who. Its goal is to generate statistical data regarding a field of study.

**1.1 Target Population**

It is crucial to specify the population that will be the subject of the study because this aids in the selection of resources and sampling strategies by the researchers. 170 IT professionals from various ICT institutions and organizations in Nigeria were the target audience for this study. Based on their area of expertise in software development, the personnel are divided into six categories.

**1.2 Sampling Size and Sampling Procedure**

To prove that the sample is representative enough for generalization, it is crucial to state the sample size and the sampling methodology. This may be the result of a number of issues that make population-wide research difficult. In addition, strategies or processes for choosing a sample from a target population are referred to as sampling procedures. To make sure all categories were fairly represented in the sample, personnel from various areas of specialization were chosen using stratified random sampling. A straightforward random technique was then employed to choose responders from different strata. A total of 150 specialists (88.23%) were included in the sample; of these, 95 were from ICT organizations and 55 were from higher education institutions' MIS departments. A total of 170 respondents, or 25 ICT lecturers from the sampled institutions, were chosen at random.

**1.3 Research Instruments**

The primary tool used to gather data was the questionnaire. This is due to the widespread use of questionnaires to swiftly and accurately gather data about current circumstances and practices as well as to ask questions about attitudes and opinions.

**1.4 Validity and Reliability of Research Instruments**

Reliability is the extent to which a test consistently measures whatever it measures, and validity is the extent to which a test measures what it is intended to measure. In this study, the reliability of the questionnaires was established through the test-and-retest method during the pilot study, while the validity of the questionnaires was guaranteed by the assessment of experts in software development, teaching, learning, and research techniques.

**1.5 Data Analysis**

In order to calculate a number, a percentage, etc., data analysis entails modifying data that has been gathered via the use of statistical tools. Following the use of a questionnaire to gather data, basic frequencies and percentages will be used to analyze the results.

**1.6 Response Rate**

Individuating the respondents' response rate is crucial for determining the comprehensiveness of the data gathered. One hundred and twenty (or 70.6%) of the 170 questionnaires that were distributed were correctly completed and returned. During analysis, fifty (29.4%) were discarded because they were not correctly filled out or returned. More than 50% returns are considered acceptable in any research. A hundred and twenty (70.6%) was deemed sufficient for analysis in this study's return.

**REFLECTION**

The security of PHP-built web applications is becoming more and more important as the digital landscape changes. A closer look at the complexities involved in protecting sensitive data, user privacy, and the overall integrity of web-based systems is prompted by thinking about the many security challenges that developers encounter in this field.

**The Complexity of Security in PHP**

PHP web applications require a sophisticated understanding of the language's features and how they interact with other components in order to navigate the complex web of potential security pitfalls. When common vulnerabilities like SQL Injection (SQLi) and Cross-Site Scripting (XSS) are taken into account, the depth of this complexity becomes clear.

One potential drawback and strength of PHP is its versatility when managing databases and user input. SQL injection attacks are commonplace, which emphasizes the importance of thorough validation and sanitization of user input. Given that even a small mistake could leave the entire system vulnerable to attack, thoughtful developers have to carefully weigh the pros and cons of flexibility versus security. In the same way, XSS poses a constant threat. The responsibility to validate and sanitize user inputs becomes even more important when considering the dynamic nature of PHP applications. The reflective practitioner considers not only technical solutions to these challenges, but also the educational efforts required to embed secure coding practices throughout development teams.

**Ethical Imperatives in Security**

Thinking about ethical considerations is necessary when exploring the security landscape of PHP web apps. It is not just a technological but also a moral duty to protect user data. It is difficult for developers to ignore concerns about consent, privacy, and the possible consequences of a security breach. Strict user authentication procedures and the deployment of secure session management acquire moral significance. It is evident to the thoughtful developer that decisions made during the system's design have an immediate effect on user confidence. It strengthens the sense of ethical responsibility ingrained in the development process to acknowledge the possible repercussions of security lapses.

**Continuous Learning Amidst Evolving Threats**

There are always new threats to contend with in the dynamic landscape of security. When considering the security risks associated with PHP web applications, as a developer, it is crucial to remain committed to lifelong learning. It becomes more than just a professional duty to stay up to date on the most recent security best practices, threat intelligence, and evolving attack vectors—it becomes a personal quest for mastery. Because security threats are constantly changing, it is necessary to adopt a constant state of alertness. The thoughtful practitioner understands full well that security is the opposite of complacency. Updating dependencies on a regular basis, adopting new security standards, and engaging with the larger security community are not merely chores on a to-do list but rather a dedication to the art of creating secure, resilient systems.

**Balancing User Experience with Security**

One of the perennial challenges in web development is striking the delicate balance between a seamless user experience and robust security measures. The reflective developer grapples with the tension between implementing stringent security controls and ensuring that these controls do not hinder the usability of the application. For instance, the decision to implement CAPTCHA mechanisms as a defense against Denial of Service (DoS) attacks prompts a reflection on user experience. Striking the right balance requires an understanding that security measures, while necessary, should not impede the intended functionality of the application. This delicate equilibrium is a constant consideration in the ongoing development and refinement of PHP web applications.

**CONCLUSION**

This research formulated the security testing for XSS vulnerabilities in a search optimization approach, with an objective of eliminating the threat arising from XSS vulnerability in PHP web application. The proposed approach is based on static analysis and genetic algorithm that will be able to detect the XSS vulnerability from PHP source code. Therefore, it was imperative that the present work improved the previous approaches on XSS detection in PHP web application by removing the infeasible paths. The resulting outcome of the present research demonstrated the approach contained zero false positive rates. Furthermore, there was experimentation of detecting the Reflected and Stored XSS vulnerability in the PHP source code, while the approach herein was able to detect the DOM-based XSS attacks based on the self-developed XSS database. However, there were no previous literatures covering experiments on Dom-based XSS. The results demonstrated that the proposed approach achieved better results compared to the previous studies on detection of reflected and stored XSS vulnerability in PHP web applications. It is worth noting here that the proposed approach need to conduct experiments on DOM-based XSS as well, and the proposed approach still need to conduct different experiments on reflected and stored XSS, in a way to reaffirm the proposed approach to detect the XSS vulnerability.

It surveys the security vulnerabilities in PHP Application among IT professionals in Nigeria. We target 170 IT Professionals in various ICT organizations and institutions in Nigeria. The main instrument for data collection was questionnaire. Stratified random method was then used to select respondents from various strata in which there were a total of 170 specialists in the sample which included 95 from ICT organization, 55 from MIS department of higher institutions and 20 ICT lecturers were randomly selected. The data collected from questionnaire were analyzed using simple frequencies and percentages. Out of 170 questionnaires distributed 120 (70.6%) were appropriately filled and returned. 50 (29.4%) were not appropriately filled or returned and therefore were discarded during analysis. Considering the profile of the respondents, majority of the respondent are male (83.33%), aged 36 – 65 years old (54.16%) followed by 18 – 35 years old (37.5050, having over 8 years PHP experience (66.66%) followed by 6 – 8 years of experience (16.67%), application developers (40.00%) followed by Designers and Programmer with (25.00%) each and are Ph.D. Holders (41.67%) followed by Master’s Degree (33.33%). In addition, higher numbers of respondent are conversant with PHP (90.00%) and using PHP always (87.50%). Furthermore, two causes of the security vulnerabilities, SQL injection and source code revelation are being highlighted by the majority of the respondents 75 (62.50%). Majority of respondents 80(66.67%) were recorded to be having database hacking challenges. And 95.83% of the respondents suggested that developer should use all the appropriate security measures and function to secure the application i.e. using ESCAPE FUNCTION, Turned OFF register\_globals directive etc. PHP has also been rated in terms of security, ease of operation and compared to other applications. So, 91.66% rated PHP applications as one of the most portability software development application, 90.84% rated PHP application as much secured application, 83.33% rated PHP as a user friendly application, 92.50% reviewed and rated PHP as best application compared to other applications and 58.33% were recorded not to have limitation with PHP. Lastly, this research can be improved to cover IT professionals worldwide. This may be accomplished by creating an online questionnaire for all IT professionals worldwide to give their opinions on the security vulnerabilities in PHP applications.

**REFERENCES**

1. Oyemade, O., Odiagbe, J. O., & Buhari, B. A. (2015). A survey of security vulnerabilities in PHP applications among IT professionals in nigeria. *African Journal of Computing & ICT*, *8*(3).
2. Gupta, B. B., & Quamara, M. (2019). *Smart Card Security: Applications, Attacks, and Countermeasures*. CRC Press.
3. Snyder, C., & Southwell, M. (2005). *Pro PHP security*. Apress.
4. Chatterjee, S., Sarker, S., & Valacich, J. S. (2015). The behavioral roots of information systems security: Exploring key factors related to unethical IT use. *Journal of Management Information Systems*, *31*(4), 49-87
5. Abrahams, T. O., Ewuga, S. K., Kaggwa, S., Uwaoma, P. U., Hassan, A. O., & Dawodu, S. O. (2024). Mastering compliance: a comprehensive review of regulatory frameworks in accounting and cybersecurity. *Computer Science & IT Research Journal*, *5*(1), 120-140.
6. Juarez, M., Afroz, S., Acar, G., Diaz, C., & Greenstadt, R. (2014, November). A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (pp. 263-274)
7. Chen, F., & Roşu, G. (2007, October). Mop: an efficient and generic runtime verification framework. In *Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems, languages and applications* (pp. 569-588)
8. McDonald, M. (2020). *Web security for developers: real threats, practical defense*. No Starch Press
9. Scambray, J., Shema, M., & Sima, C. (2006). *Hacking exposed: Web applications*. New York: McGraw-Hill.