

# Comprehensive secure code review analysis of web application security vulnerabilities

Azlinda Abdul Aziz<sup>1</sup>, Nur Razia Mohd Suradi<sup>2</sup>, Rahayu Handan<sup>2</sup>, Mohd Noor Rizal Arbain<sup>2</sup>

<sup>1</sup>Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM) Melaka, Campus Jasin, Melaka, Malaysia

<sup>2</sup>Faculty of Communication, Visual Arts and Computer, Universiti Selangor, Selangor, Malaysia

## Article Info

### Article history:

Received Nov 11, 2024

Revised Apr 7, 2025

Accepted Jul 2, 2025

### Keywords:

Cyber thread

Risk

Secure code

Security

Vulnerabilities

Web application

## ABSTRACT

A secure code review is a process of software development involves systematic examination of application code. However, web applications evolving of cyber threats makes it challenging to conduct adequate security. Therefore, this paper conducts a comprehensive secure code review analysis to protect any crucial aspect of web security from potential threats and vulnerabilities. The application code is scanned for security issues during the real review, and the results are classified according to the areas of vulnerability. As a result, the application code risk level and list of risk categories were defined. This result assists in prioritizing issues for resolution, beginning with the most critical problems to lower risk levels. Next, a list of risk categories that give the most significant security vulnerabilities affecting application codes is defined. SQL injection, weak password handling, insecure direct object reference, information exposure, improper session management, missing input validation, deprecated functions, and lack of comments are defined as a risk category. Moreover, the result of application code weakness in the security of the application code is determined based on the level of risk and categories. Thus, the analysis result offers the developers a clear perspective on protecting the web applications from threats and vulnerabilities.

This is an open access article under the [CC BY-SA](#) license.



## Corresponding Author:

Azlinda Abdul Aziz

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM) Melaka

Campus Jasin, Melaka, Malaysia

Email: azlindaaziz@uitm.edu.my

## 1. INTRODUCTION

Web applications play a critical role in modern digital ecosystems, supporting essential services across various industries. Strict measures must be taken at every level of the software development lifecycle because of the widespread use of these devices, which often makes them the target of security attacks. One crucial approach to safeguarding web applications is secure code review, a process in which developers analyze application code to identify and mitigate security vulnerabilities, enhance code quality and ensure compliance with security standards [1]. This process is essential for safeguarding web applications, especially those handling sensitive data, from potential cyber threats and attacks [2]. However, the complexity of modern web applications and the constantly evolving nature of cyber threats make conducting effective security reviews challenging [3]. Furthermore, ongoing training helps developers stay informed about the latest security threats and best practices. Incorporating security measures early in the development lifecycle helps establish a strong foundation for secure coding [4].

One of the main objectives of secure code reviews is to identify and fix security vulnerabilities before deployment. Therefore, this process helps prevent common threats such as SQL injection, cross-site scripting

(XSS), and cross-site request forgery (CSRF), which could compromise sensitive data and system security [5]. Optimizing secure code review techniques for real-world use is still difficult, despite prior research emphasizing the need of early security implementation and ongoing developer training. This study explores secure code review, evaluating their effectiveness and suggesting improvements to enhance web application security. By implementing secure code review practices, developers can adopt best security measures, such as enforcing secure authentication mechanisms, validating user inputs, and managing sessions securely, all of which contribute to strengthening web application security. Conducting thorough secure code reviews is important to keep web applications strong and safe, especially as cyber threats continue to evolve [6].

## 2. LITERATURE REVIEW

Code review is an integral part of software development [7]. The developers examine source code in a systematic approach to find bugs, improve code quality, and make sure standards are followed. Reviews of the code are an essential part of web security because they keep applications safe from threats and holes [8]. Web applications are easy targets for hackers because they handle a lot of private data and are used by many people. Therefore, code reviews are important for finding vulnerabilities before they happen [9]. The main goal of code reviews is to find and fix security holes before the code is deployed. Moreover, adopting a proactive approach helps mitigate risks such as SQL injection, XSS, CSRF, and other common vulnerabilities that could enable hackers to compromise data or breach security [10].

Code reviews significantly improve web application security by ensuring developers follow the best practices for security, such as using secure authentication methods, validating input, and managing sessions securely [11]. Code reviews also improve code quality by encouraging code to be straightforward, easy to update, and work well. Moreover, it encourages team members with the developer to work together and share what they know, which helps everyone understand security principles and practices better [12]. A structured review method looks at all the code, gives helpful feedback on security issues, and checks the fixes to ensure the problems are fixed correctly.

The complexity of current web applications and the constantly changing cyber threats make it hard to do effective security. Security practices early in the development lifecycle help to set the stage for safe code. Previous studies have mostly examined automatic and manual code reviews independently, each with unique advantages and limitations. By examining big codebases, automated technologies effectively find typical vulnerabilities like SQL injection and XSS [13]. Nevertheless, they frequently produce false positives and might fail to notice context-specific security vulnerabilities. Conversely, manual reviews enable skilled developers to properly examine the code, identifying logical errors and business logic weaknesses that automated methods could overlook. Therefore, developers must stay updated on the latest security threats and protection measures by participating in regular training sessions [14]. To overcome these problems, a mix of automated tools for quick identification and hand inspection for more in-depth analysis [15]. Thus, the study conducts a comprehensive code review analysis of web security by integrating various techniques, providing a solid foundation for web application security [16].

## 3. OBJECTIVES

There are two main objectives to be achieved in this study. Here are the objectives:

- a) To identify the risk level and categories from the secure code review affect the lack of security
- b) To examine how application code development affects the resistance to cyber threats

## 4. METHOD

The secure code review process follows a structured approach to ensure a systematic evaluation of web applications before deployment. This study involves a combination of manual and automated techniques to identify security vulnerabilities effectively. The methodology consists of several key stages, including the selection of multiple web applications developed by the developer to ensure diversity in codebases and security implementations. Applications with different architectures, programming languages, and frameworks are included to enhance the study's generalizability. The secure code review is to identify security weaknesses resulting from missing or inadequate security controls, which may compromise the application's resistance to cyber threats [17]. Rather than attempting to identify every possible flaw, the review prioritizes the detection of key vulnerabilities that indicate broader security risks. The methodology adopts a comprehensive classification of security threats, which helps in creating a more systematic approach to secure coding. Security analysts manually examine the source code to detect security flaws in authentication mechanisms, data validation, and session management [18]. Identified security risks are documented in

dedicated spreadsheets, categorized based on severity, and assigned risk ratings. Common vulnerabilities are verified through cross-checking, while less frequent security issues undergo group reviews for validation. This process directly supports the objective of identifying risk levels and categories that contribute to the lack of security in web applications.

The combination of manual and automated techniques ensures comprehensive vulnerability detection [19]. Manual reviews help identify logic-based security issues that automated tools might overlook, while automated tools enhance coverage and efficiency. Each identified security issue is assessed based on predefined risk criteria, considering factors such as exploitability, impact, and remediation complexity. Risk reports include severity ratings and recommended mitigations to address identified issues. By systematically identifying and mitigating potential security vulnerabilities, the secure code review strengthens web application security [20]. Furthermore, it enhances developers' awareness of security issues, contributing to the development of more resilient applications. Security risks are greatly decreased by best practices such as strong data validation, secure authentication, and appropriate session management. Ensuring adherence to these principles protects user data and maintains the credibility of web applications. Finally, the review directly addresses the objectives of the study by evaluating the effectiveness of security implementations in web applications and making a comprehensive classification of security threats, which helps in creating a more systematic approach to secure coding [21]. This approach aligns with the study's aims to identify vulnerabilities and improving application security resilience. The methodology stage consists of defining objectives, outlining the review process, and analyzing the study outcomes, as illustrated in Figure 1.

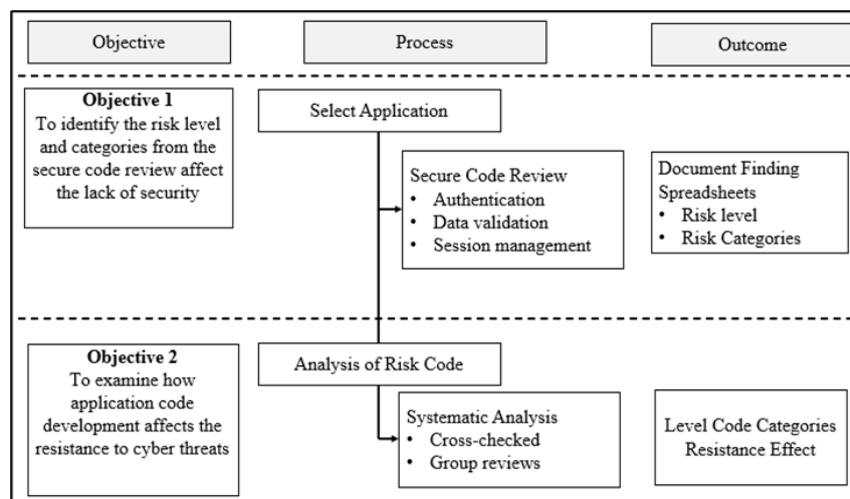


Figure 1. Methodology stage

## 5. RESULTS AND DISCUSSION

The analysis starts by viewing the code created by the developer and examining the code's lack of security, which can cause a cyber threat. The analysis of the code risk levels is divided into four categories: high, medium, low, and informational, as illustrated in Figure 2. The findings show high risk consists of ten findings with the most prominent effect and needs quick attention to stop significant security breaches and system failures. The medium risk showed nine results that essential problems must be fixed immediately. Low Risk has one finding that points to minor bugs in the code that need to be fixed to keep the code quality high. Two findings for informational research are not vulnerabilities but helpful observations for best practices ideas. Furthermore, this result helps prioritize problem fixes, starting with the most critical issues and moving to those with lower risk levels [22].

Moreover, Figure 3 shows the list of risk category findings and highlights the most significant security vulnerabilities identified during the secure code review. The most common are and missing input validation, which are shown in seven cases [23]. Therefore, it shows that bigger problems must be fixed because attackers could easily run flaws at any SQL command and take advantage of the fact that input is not being appropriately checked, which would be a significant security breach [24]. Weak password handling and improper session management were found two times, which shows that keeping user sessions and private information safe is not strong enough. Weak password handling, insecure direct object reference, information exposure, deprecated functions, and lack of comments are some less common but still important results. Although these problems are less frequent, they still pose significant risks that must be addressed [25].

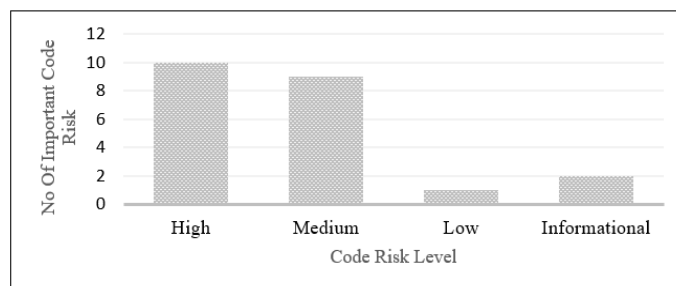


Figure 2. Risk level finding

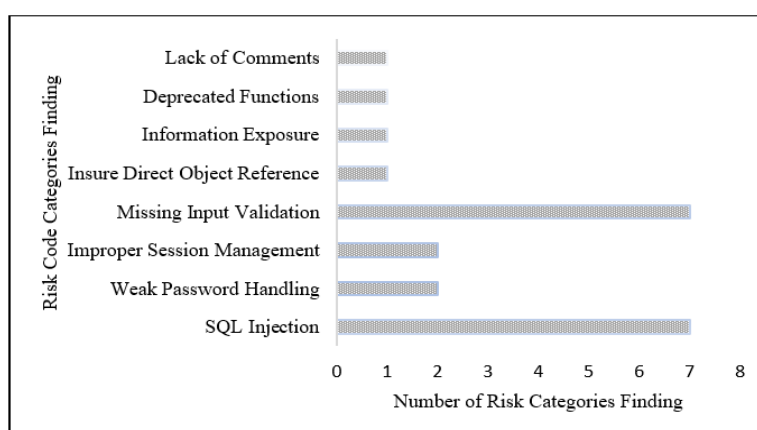


Figure 3. List risk categories

Moreover, the application codes were analyzed by the secure code reviewer, and the weak security of the application code development was defined. Based on the analysis, the higher risk code categories of resistance effects are shown in Table 1. The study revealed SQL injection is the weakness of the software, which constructs SQL commands using externally influenced input without proper neutralization of unique elements that could modify the intended SQL command. Moreover, weak password handling must be stressed to avoid cryptographic failures for plaintext password storage. The weakness is that the software stores sensitive data, such as passwords, using a reversible encoding rather than a one-way hash [25].

Furthermore, in insecure direct object reference, broken access control occurs in broken access control, which is authorization bypass through a user-controlled key. The weakness of the application is that it does not correctly verify if a user is authorized to access the resource it is requesting. Moreover, information exposure affects sensitive data exposure for information exposure. The weakness is the software does not adequately protect sensitive data from unauthorized access during its lifetime.

Moreover, Table 2 shows the analysis result of the resistance effects of the medium-risk code categories, which consist of improper session management and missing input validation. The improper session management in cryptographic failures shows the weakness in plaintext storage of a password where the software stores sensitive data such as passwords using reversible encoding rather than a one-way hash. Moreover, missing input validation shows that the code created is not an improper input validation where the weakness of the software does not properly validate inputs, which can affect and control the flow of the program.

Moreover, Table 3 shows the resistance effects of the informational code categories, which consist of deprecated caused by the security logging and monitoring failures, which are the effect of potentially dangerous functions where the application uses deprecated functions that of deprecated functions and a lack of comments. From the analysis shown the deprecated functions are caused by security logging and monitoring failures, which are the effect of potentially dangerous functions where the application uses deprecated tasks that might not be supported in future programming language versions and may have known vulnerabilities [15]. Furthermore, the lack of comments affects the security logging and monitoring failures for suspicious comments. It involves the application's lack of comments explaining complex code sections, which can lead to difficulties in maintenance and debugging.

Table 1. High-risk code categories resistance effect

Categories	Application code	Description	Risk
SQL injection	<code>\$sql = "INSERT INTO 'transaction'('item','amount','type','email','date')VALUES ('\$item','\$amount','\$type','\$temp','\$date')";</code>	User input variables are directly concatenated into the SQL query without proper sanitization or use of prepared statement	High
Weak password handling	<code>\$sql = "UPDATE 'user' SET 'password'=' \$password' WHERE 'email'=' \$id'"</code>	The new password is stored in plaintext in the database	High
Insecure direct object reference	<code>\$d = \$ GET('id') \$sql = "DELETE FROM 'transaction' WHERE id = "\$id";</code>	The transaction ID from the URL parameter is used directly in an SQL DELETE query without verifying user permissions	High
Information exposure	<code>if (\$row ==1) { \$ _SESSION['sess'] = \$email; \$ _SESSION['notify'] = '1'; header ("location: ../ dashboard.php"); } else ( \$ _SESSION['notify'] = '2'; header ("location: ../ login.php"); }</code>	Error messages reveal sensitive information	High

Table 2. Medium-risk code categories resistance effects

Categories	Application code	Description	Risk
Improper session management	<code>SESSION_START();</code>	The session is started without regenerating the session ID upon successful login	Medium
Missing input validation	<code>\$type= \$ _POST['type']; \$item= \$ _POST['item'] \$amount= \$ _POST['amount']</code>	The session is started without regenerating the session ID upon successful login	Medium

Table 3. Informational risk code categories resistance effects

Categories	Application code	Description	Risk
Deprecated functions	<code>\$conn= mysql_connect ('localhost', 'root', 'mt');</code>	The mysql_connect function is deprecated	Informational
Lack of comment	<code>\$sql = "UPDATE 'user' SET 'name' = "\$name", 'email' = '\$email' WHERE 'email' = '\$temp', if (\$conn-&gt; query (\$sql) === TRUE);</code>	The code performs a series of operations without any comments explaining the logic	Informational

The findings of this study highlight the critical importance of secure coding practices in mitigating security vulnerabilities before deployment. The analysis confirms that proper input validation, the use of structured SQL queries, password hashing, session ID regeneration, and continuous code updates are essential in preventing security threats. A key finding is that strict input validation effectively prevents XSS and SQL injection attacks, reinforcing the necessity of implementing prepared statements to ensure user input is treated as data rather than part of SQL commands [19]. Additionally, the study underscores that adding a unique salt to passwords before hashing enhances security, ensuring that even if a hash is compromised, the original password remains protected. These findings align with well-established security guidelines, demonstrating that adhering to best coding practices significantly reduces security risks in web applications.

In contrast to earlier studies, which frequently offers broad security advice, this study provides a systematic and categorized way to find and classify vulnerabilities according to their relevance. This allows developers to effectively allocate resources toward the most critical security risks. One of the study's strengths is its comprehensive classification of security threats, which helps in creating a more systematic approach to secure coding. However, a notable limitation is its reliance on static code analysis, which may overlook runtime vulnerabilities and emerging threats [22]. The results also show that poor session management and inadequate password handling are more common than expected, indicating that even basic security precautions are frequently disregarded in practical applications. Furthermore, despite current development norms restricting such practices, the continued use of obsolete functions and the absence of adequate code documentation point to ongoing security oversights. These unexpected results highlight the need for greater awareness and enforcement of security best practices among developers. By integrating both automated and manual review techniques, this study enhances the accuracy and efficiency of secure code analysis. Automated tools allow for quick detection of common vulnerabilities, while manual reviews provide deeper insight into logic-based security flaws [25]. This combination ensures a more comprehensive approach to securing web applications, reinforcing best coding practices and strengthening cybersecurity defenses.

This study successfully achieves its goal of emphasizing the importance of secure code reviews in reducing cybersecurity risks and strengthening web application security. It provides an effective structure for developers, organizations, and cybersecurity experts to enhance their security strategies through proper input validation, secure session management, and strong authentication mechanisms. To enhance the effectiveness and precision of secure code reviews, future enhancement need explores artificial intelligence (AI)-enhanced tools and dynamic security testing approaches. By continuously refining security techniques, the industry can stay ahead of evolving cyber threats and ensure the long-term security and resilience of web applications.

## 6. CONCLUSION

In conclusion, carefully reading and looking over the application code developed by the developer by hand during the review is essential. The secure code review revealed several security holes requiring immediate attention to improve security and code quality. Identifying and addressing these vulnerabilities at an early stage helps prevent potential cyber threats that could exploit weaknesses in the system. Using the analysis requires immediate attention to improve. From that, code quality can enhance by encouraging straightforward, easy-to-update, and well-functioning code. Consequently, the developer needs to review all the code and security methods to ensure the security problems are fixed correctly. By integrating automated tools alongside manual reviews, developers can enhance accuracy and efficiency in detecting security flaws, leading to more secure and resilient applications. A strong secure coding culture within development teams ensures long-term security benefits and adopts accountability among developers. Therefore, secure code reviews are very important for keeping users' trust and protecting private data because they find bugs before they happen, follow best practices, and encourage a culture of constant improvement. Thus, developers can make their web applications much safer by following these rules. These rules will protect against standard security holes and give developers a strong defense against threats. Additionally, organizations should invest in regular security training programs to keep developers updated on the latest threats and protection strategies. Lastly, continuous monitoring, regular updates, and thorough testing are necessary to maintain the security and integrity of an application in the long term. A proactive approach to security not only protects applications from cyberattacks but also strengthens the overall digital ecosystem by reducing security risks at a larger scale.

## ACKNOWLEDGEMENTS

The appreciation for the team member that has collaboration in completed for this research and contribute throughout the process of writing paper. Their expertise and support have greatly enhanced the quality of this work.

## FUNDING INFORMATION

This research was conducted independently by the authors without any external financial support.

## AUTHOR CONTRIBUTIONS STATEMENT

All authors contributed meaningfully to the conception, design, execution, and analysis of this study, as described in the following table:

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Azlinda Abdul Aziz	✓	✓	✓	✓	✓		✓	✓	✓	✓			✓	
Nur Razia Mohd Suradi		✓	✓	✓				✓	✓	✓				
Rahayu Handan	✓		✓	✓			✓			✓				
Mohd Noor Rizal Arbain	✓	✓			✓		✓							

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nterpretation

R : **R**esources

D : **D**ata Curation

O : **O**riginal Draft

E : **E**diting

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

## CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest related to this research.




## DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request. Due to privacy and confidentiality considerations, the data are not publicly available.




## REFERENCES

- [1] X. D. Hoang, T. H. Nguyen, and H. D. Pham, "A novel model for detecting web defacement attacks transformer using plain text features," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 37, no. 1, pp. 232-240, Jan. 2025, doi: 10.11591/ijeecs.v37.i1.pp232-240.
- [2] A. A. Almutairi, S. Mishra, and M. AlShehri, "Web security: emerging threats and defense," *Computer Systems Science and Engineering*, vol. 40, no. 3, pp. 1233-1248, 2021, doi: 10.32604/CSSE.2022.019427.
- [3] S. Calzavara, R. Focardi, M. Squarcina, and M. Tempesta, "Surviving the web: a journey into web session security," in *The Web Conference 2018 - Companion of the World Wide Web Conference, WWW 2018*, New York, New York, USA: ACM Press, 2018, pp. 451-455. doi: 10.1145/3184558.3186232.
- [4] V. Abdullayev and A. S. Chauhan, "SQL injection attack: quick view," *Mesopotamian Journal of CyberSecurity*, vol. 2023, pp. 30-34, Feb. 2023, doi: 10.58496/MJCS/2023/006.
- [5] Y. Ashibani and Q. H. Mahmoud, "Cyber physical systems security: analysis, challenges and solutions," *Computers and Security*, vol. 68, pp. 81-97, Jul. 2017, doi: 10.1016/j.cose.2017.04.005.
- [6] H. Lamsellak and M. G. Belkasmi, "Global software development agile planning model: challenges and current trends," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 32, no. 3, pp. 1774-1784, Dec. 2023, doi: 10.11591/IJECS.V32.I3.PP1774-1784.
- [7] R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, "Systematic literature review on security risks and its practices in secure software development," *IEEE Access*, vol. 10, pp. 5456-5481, 2022, doi: 10.1109/ACCESS.2022.3140181.
- [8] Q. Wang *et al.*, "Break the wall from bottom: automated discovery of protocol-level evasion vulnerabilities in web application firewalls," in *Proceedings - IEEE Symposium on Security and Privacy*, IEEE, May 2024, pp. 185-202. doi: 10.1109/SP54263.2024.00129.
- [9] J. Heino, A. Hakala, and S. Virtanen, "Study of methods for endpoint aware inspection in a next generation firewall," *Cybersecurity*, vol. 5, no. 1, p. 25, Sep. 2022, doi: 10.1186/s42400-022-00127-8.
- [10] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, "Software security patch management - a systematic literature review of challenges, approaches, tools and practices," *Information and Software Technology*, vol. 144, p. 106771, Apr. 2022, doi: 10.1016/j.infsof.2021.106771.
- [11] M. Alsaffar *et al.*, "Detection of web cross-site scripting (XSS) attacks," *Electronics*, vol. 11, no. 14, p. 2212, Jul. 2022, doi: 10.3390/electronics11142212.
- [12] L. Braz, E. Fregnan, G. Calikli, and A. Bacchelli, "Why don't developers detect improper input validation? "; DROP TABLE papers; --," in *Proceedings - International Conference on Software Engineering*, IEEE, May 2021, pp. 499-511. doi: 10.1109/ICSE43902.2021.00054.
- [13] A. O. Agbakwuru and D. O. Njoku, "SQL injection attack on web base application: vulnerability assessments and detection technique," *International Research Journal of Engineering and Technology*, pp. 243-252, 2021.
- [14] W. P. K. Fernando, D. A. N. P. Dissanayake, S. G. V. D. Dushmantha, D. L. C. P. Liyanage, and C. Karunatilake, "Challenges and opportunities in password management: a review of current solutions," *Sri Lanka Journal of Social Sciences and Humanities*, vol. 3, no. 2, pp. 9-20, Aug. 2023, doi: 10.4038/sljssh.v3i2.96.
- [15] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yılmaz, and E. Akin, "A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions," *Electronics*, vol. 12, no. 6, p. 1333, Mar. 2023, doi: 10.3390/electronics12061333.
- [16] R. L. Alaoui and E. H. Nfaoui, "Deep learning for vulnerability and attack detection on web applications: a systematic literature review," *Future Internet*, vol. 14, no. 4, p. 118, Apr. 2022, doi: 10.3390/fi14040118.
- [17] W. Charoenwet, P. Thongtanunam, V. T. Pham, and C. Treude, "Toward effective secure code reviews: an empirical study of security-related coding weaknesses," *Empirical Software Engineering*, vol. 29, no. 4, p. 88, Jul. 2024, doi: 10.1007/s10664-024-10496-y.
- [18] L. Braz, C. Aeberhard, G. Calikli, and A. Bacchelli, "Less is more: supporting developers in vulnerability detection during code review," in *Proceedings - International Conference on Software Engineering*, New York, NY, USA: ACM, May 2022, pp. 1317-1329. doi: 10.1145/3510003.3511560.
- [19] G. Deepa and P. S. Thilagam, "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," *Information and Software Technology*, vol. 74, pp. 160-180, Jun. 2016, doi: 10.1016/j.infsof.2016.02.005.
- [20] R. A. Putra, I. A. Kautsar, H. Hindarto, and S. Sumarno, "Detection and prevention of insecure direct object references (IDOR) in website-based applications," *Procedia of Engineering and Life Science*, vol. 4, Jul. 2023, doi: 10.21070/pels.v4i0.1435.
- [21] L. Braz and A. Bacchelli, "Software security during modern code review: the developer's perspective," in *ESEC/FSE 2022 - Proceedings of the 30th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, NY, USA: ACM, Nov. 2022, pp. 810-821. doi: 10.1145/3540250.3549135.
- [22] S. Alazmi and D. C. De Leon, "A systematic literature review on the characteristics and effectiveness of web application vulnerability scanners," *IEEE Access*, vol. 10, pp. 33200-33219, 2022, doi: 10.1109/ACCESS.2022.3161522.
- [23] Z. Li *et al.*, "Automating code review activities by large-scale pre-training," in *ESEC/FSE 2022 - Proceedings of the 30th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, NY, USA: ACM, Nov. 2022, pp. 1035-1047. doi: 10.1145/3540250.3549081.
- [24] V. Casola, A. De Benedictis, C. Mazzocca, and V. Orbinato, "Secure software development and testing: A model-based methodology," *Computers and Security*, vol. 137, p. 103639, Feb. 2024, doi: 10.1016/j.cose.2023.103639.
- [25] S. Kartunov, "Protection of user credentials in web application," in *Thematic conference proceedings of international significance [Elektronski izvor]/International Scientific Conference*, 2020, pp. 604-610. [Online]. Available: <http://jakov.kpu.edu.rs/handle/123456789/1334>




**BIOGRAPHIES OF AUTHORS**

**Azlinda Abdul Aziz**    received her Ph.D. in computing from Universiti Selangor (UNISEL), Malaysia, in 2022, an M.Sc. in computer science (distributed computing) from Universiti Putra Malaysia in 2003, and a Bachelor of Science in computer science from Universiti Putra Malaysia in 2000. Her research interests include IoT and cloud computing, cybersecurity and web security, computer and network security, as well as telecommunication and network forensics. She can be contacted at email: [azlindaaziz@uitm.edu.my](mailto:azlindaaziz@uitm.edu.my).






**Nur Razia Binti Mohd Suradi**    received her Ph.D. in computing from Universiti Selangor (UNISEL) in 2023, Master's in computer science (software engineering) in 2011 and B.Sc. Degree in computer science from Universiti Sains Malaysia in 1990. Her fields of interest are software engineering, web programming, software quality assurance and software testing. She can be contacted at email: [razia@unisel.edu.my](mailto:razia@unisel.edu.my).



**Rahayu Handan**    received her BBA in Management Information Systems (MIS) and International Business from Drexel University, PA, USA, in 1999, and her M.Sc. in Computer Science (IT) from UiTM, Selangor, Malaysia, in 2012. Her research interests include technopreneurship, information systems, and human-computer interaction. She can be contacted at email: [raha@unisel.edu.my](mailto:raha@unisel.edu.my).



**Mohd Noor Rizal Bin Arbain**    received his Diploma in electronic engineering (majoring computer) from Polytechnic Ungku Omar (PUO) in 2001, Bachelor's degree in Information Technology (Majoring Networking) from University Utara Malaysia (UUM) in 2004 and Master's in information technology (majoring management information system) from University Selangor (UNISEL) in 2010. Currently pursuing a Ph.D. in information technology (research in field cybersecurity and deep learning) from University Kuala Lumpur (UNIKL), Malaysia. His field interest are computer networking, cybersecurity, and deep learning. He can be contacted at email: [rizal-it@unisel.edu.my](mailto:rizal-it@unisel.edu.my).