International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-12 Issue-2, July 2023

A Study of The Effectiveness of Code Review in Detecting Security Vulnerabilities

G.H.N Anuththara, S.S.U Senadheera, S.M.T.V Samarasekara, K.M.G.T Herath, D. I. De Silva, M. V. N. Godapitiya

Abstract*: Software flaws pose a severe danger to the security and privacy of computer systems and the people who use them [1]. For software systems to be reliable and available, vulnerabilities must be found and fixed before they can be used against the system [2]. Two popular methods for finding weaknesses in software systems are code review and penetration testing [3]. Which method is better for identifying vulnerabilities, nevertheless, is not widely agreed upon [4]. The usefulness of code reviews and penetration tests in locating vulnerabilities is reviewed in detail in this study. We evaluate much empirical research [5] and contrast the benefits and drawbacks of each method. According to our research, both code reviews and penetration tests help uncover vulnerabilities [6], even though their effectiveness varies based on the kind of vulnerability, the complexity of the code, and the testers' or reviewers' experience [7][8]. Additionally, we discovered that doing both penetration testing and code review together may be more efficient than using each approach alone [9]. These results may help software engineers, security experts, and researchers select and utilise the most effective approach for identifying weaknesses in software systems.

***Keywords*: Software Vulnerabilities, Code Review, Penetration Testing, Effectiveness, Empirical Studies, Strengths and Weaknesses, Combined Strategy, Software Development, Security Professionals, Recommendations.**

Manuscript received on 08 May 2023 | Revised Manuscript received on 23 May 2023 | Manuscript Accepted on 15 July 2023 | Manuscript published on 30 July 2023.

*Correspondence Author(s)
G.H.N Anuththara, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka. Email: ghnanuththara@gmail.com, ORCID ID: https://orcid.org/0009-0009-1634-694X
S.S.U Senadheera*, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka. Email: shsenadheera@gmail.com, ORCID ID: https://orcid.org/0009-0003-6061-2906
S.M.T.V Samarasekara, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka. Email: thisunsamarasekara@gmail.com, ORCID ID: https://orcid.org/0009-0008-6784-0158
K.M.G.T Herath, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka. Email: gthgayashanthilina@gmail.com, ORCID ID: https://orcid.org/0009-0000-1224-959X
M. V. N. Godapitiya, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka. Email: virajini.g@sliit.lk, ORCID ID: https://orcid.org/0009-0000-2529-3311
Dr. D. I. De Silva, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka. Email: dilshan.i@sliit.lk, ORCID ID: https://orcid.org/0000-0001-6821-488X

I. INTRODUCTION

Both software developers and consumers are becoming more concerned about software vulnerabilities. Software system vulnerabilities must be found and fixed immediately. Due to the complexity and frequency of cyber-attacks [10]. Code review and penetration testing are two methods that are often used to find vulnerabilities in software [11]. In a code review, the source code is thoroughly examined to find any possible flaws [12]. To find vulnerabilities that may be exploited, penetration testing includes simulating an assault on a software system [13].

Despite the significance of finding vulnerabilities, there is disagreement over the efficiency of code review and penetration testing in doing so. According to some research, code review is more successful than penetration testing [14] [15], while other studies [16][17] support the opposite conclusion. Additionally, some research contends that the most successful strategy could include combining the two methods [18][19]. The lack of agreement on these issues complicates the identification of weaknesses in software systems.

Code review and penetration testing have gained popularity in recent years as methods for finding weaknesses in software systems. However, there is ongoing discussion over whether or not these approaches help locate vulnerabilities [20]. Additionally, there is a dearth of studies contrasting the advantages and disadvantages of penetration testing and code review [21]. By contrasting the efficiency of code review and penetration testing in locating vulnerabilities in software systems and by identifying variables that may impact their efficacy, this research article tries to fill these gaps in the literature. Therefore, the research question covered in this article is: Compared to alternative methods like penetration testing, how successful is code review in identifying vulnerabilities? This research article will evaluate, synthesize, and compare the efficacy of penetration testing and code review to respond to this topic. By replying to this research topic, the study aims to inform software engineers and security experts on the most effective methods for identifying vulnerabilities in software systems. Code review is the act of methodically going over a software application's source code to find errors or vulnerabilities before they can be exploited. To guarantee that the code is safe, dependable, and practical, it is a crucial quality assurance approach.

Security has grown in importance as a concern in recent years, particularly as software applications continue to be developed. Software developers are under pressure to ensure that their code is secure, as cyberattacks and data breaches are becoming increasingly common and complex.

Through code review, this problem may be solved, for example. Studies have demonstrated that reviewing the source code of software programs can help find security flaws. Developers can identify potential vulnerabilities by examining the code and taking corrective actions to address them before an attacker can exploit them.

Nevertheless, despite the potential advantages of code review, little is known about how effectively it identifies security flaws. This study on the effectiveness of code review in identifying security flaws aims to bridge this knowledge gap. The article will examine various processes and tools for code reviews, as well as the tools and technologies that support them. Additionally, it will examine the challenges and limitations of code review and provide recommendations for improving its ability to identify security flaws.

Security and privacy of computer systems and their users are seriously threatened by software vulnerabilities [22]. For software systems to be reliable and available, vulnerabilities must be found and fixed before they can be used against the system [10]. Two popular methods for finding weaknesses in software systems are code review and penetration testing [23]. However, academics and practitioners continue to disagree about how well these strategies work to find vulnerabilities [24].

There is no agreement on which method is better for finding vulnerabilities, despite the expanding volume of research on both code reviews and penetration testing [25]. According to specific research, code reviews are more effective [26][27], whereas penetration tests are more effective [28][29]. Studies have also shown that combining the two methods may be the most successful strategy [28]. This lack of agreement raises crucial issues, such as whether software developers and security experts are using the most effective methods for their purposes and the most efficient manner to identify vulnerabilities in software systems.

Determining the efficiency of code review and penetration testing in detecting vulnerabilities, as well as contrasting their advantages and disadvantages, is the subject that this research article attempts to address. By addressing this issue, the study article aims to offer suggestions and insights to help software engineers and security experts select and utilise the most effective method for identifying vulnerabilities in software systems. Additionally, this study aims to identify variables that may affect the success of penetration testing and code review, and to contribute to a broader discussion on these topics.

Software development is not complete without code review, which helps identify and prevent security flaws. Organisations must now ensure that their software is both trustworthy and secure, given the increasing threat of cyberattacks. Therefore, research into how well code reviews work at identifying security flaws is essential for assisting businesses in making decisions about their software development processes [30].

This study makes a significant contribution by helping to create best practices for code reviews. Organizations may improve the effectiveness and efficiency of their code review processes by finding the best review processes, tools, and techniques [31]. According to a study by Rahman et al. (2018), using checklists and recommendations could aid reviewers in spotting common types of vulnerabilities and boost code review effectiveness [31].

This study also has the benefit of providing insight into the effectiveness of various code review methodologies. Code review can take many different forms, including automated, tool-assisted, and manual peer review. Organizations can choose the optimal approach for their goals by being aware of the advantages and disadvantages of each option [32]. According to a study by Zeller et al. (2019), combining manual and tool-assisted review to discover security vulnerabilities was more efficient than using either approach alone [32].

Additionally, by balancing code review with other security practices, such as penetration testing and vulnerability scanning, the study can help organisations make informed choices. Code review is simply one component of a comprehensive security strategy, so it is essential to understand where it fits into the entire security plan [33]. Code review helped identify some vulnerabilities, such as SQL injection and cross-site scripting, but less effective at identifying others, like authentication and authorization problems, according to a study by Wang et al. (2019) [33].

In conclusion, the study on code review's efficiency in spotting security flaws is vital since it can help companies strengthen the security and dependability of their software [30]. Organizations may improve the effectiveness and efficiency of their code review processes by finding the best review processes, tools, and techniques [31]. Organisations can also improve their overall security by recognizing the advantages and disadvantages of various code review methodologies and balancing code review with other security measures [32][33].

The purpose of the study on the effectiveness of code review in discovering security vulnerabilities is to assess how well various code review methodologies work at spotting multiple security problems. The study examines tool-assisted code review in conjunction with manual and automated code review. A hybrid strategy that combines the advantages of both human and computerised methods is tool-assisted code review. According to studies, tool-assisted code reviews can find more security problems than manual reviews alone [34]. The goal of this study is to determine the optimal combination of manual, automated, and tool-assisted review methodologies for identifying various security vulnerabilities.

The study also aims to evaluate how review duration affects the efficiency of code reviews. According to research, more faults are found the more thorough the examination is [35]. There is, however, a limit to diminishing returns, beyond which the extra time spent reviewing does not significantly improve the number of flaws discovered. Therefore, the goal of the study is to determine the ideal review time for various software projects and types of code reviews.

Examining the effect of reviewer experience on the efficacy of code review is another aim of the study. Experienced reviewers are better able to find problems than less experienced reviewers, according to prior research [36]. Experienced reviewers, however, can also be more prone to cognitive biases that could hinder their capacity to find specific kinds of flaws. Therefore, the goal of the study is to determine the ideal level of reviewer experience for various software projects and types of code reviews.

Based on the research's conclusions, the study aims to provide best practices for code review. Depending on the nature of the software project and the kinds of security vulnerabilities being targeted, best practices may include recommendations for choosing the most efficient review approaches, tools, and procedures. The optimization of review time, reviewer skill, and other factors that affect code review efficiency may also be included in best practices.

Research question: How effective is code review in identifying vulnerabilities in comparison to other techniques like penetration testing?

In this study, we investigate the effectiveness of penetration testing and code review in identifying vulnerabilities in software systems. We want to know, "How effective is code review in identifying vulnerabilities in comparison to other techniques like penetration testing?" We provide three theories to address this question. First, we propose that, when it comes to finding vulnerabilities in software systems, code review outperforms penetration testing [26][27][37].

Second, we believe that when it comes to finding software system vulnerabilities, penetration testing outperforms code reviews [28][29][25]. Finally, we propose that the most successful method for locating vulnerabilities in software systems is a mix of code review and penetration testing [38][39][40]. By putting these theories to the test, we aim to illuminate the effectiveness of various vulnerability detection strategies and help software developers and security experts select the most suitable plan for their needs.

Software security is crucial, particularly in the current climate of frequent cyberattacks and data breaches. To prevent similar situations, it is essential to identify and address software vulnerabilities. Code review and penetration testing are two widely used procedures for identifying vulnerabilities. The efficiency of these approaches, however, is debatable and varies based on several variables, including the type of vulnerability, the complexity of the code, and the skill of the testers and reviewers.

This study compares the effectiveness of penetration testing versus code reviews for finding software vulnerabilities. It evaluates several empirical studies and highlights the advantages and disadvantages of both approaches. Although there is no universal agreement on which approach is superior, some studies suggest that combining the two methods can lead to more effective vulnerability detection.

Code review is meticulously searching for errors and vulnerabilities in a software application's source code. To ensure code quality and enhance software security, it is a crucial procedure. The study paper analyzes different code review methodologies, tools, and technologies. It also highlights its drawbacks and provides recommendations on how to enhance the detection of security flaws.

There is currently no consensus on which method is more effective for finding vulnerabilities, despite the increasing number of studies on code review and penetration testing. According to some research, code reviews are more productive, while others prefer penetration testing. The most significant outcomes, however, could come from combining the two approaches. The study report provides guidance to help software developers and security experts choose the most effective vulnerability detection strategy, as well as insights into the variables that may impact the effectiveness of various approaches.

The importance of code review in mitigating security issues during software development is emphasized in the study article. It emphasises various review techniques, tools, and tactics, while outlining best practices for code reviews. It also discusses the advantages and disadvantages of peer review, tool-assisted code review, and automated code review. The goal of this article is to help enterprises select the optimal code review technique for their specific requirements.

The study report concludes by arguing that the type of vulnerability, the complexity of the code, and the tester/reviewer's experience all affect how effectively code review and penetration testing work in identifying security flaws. It emphasizes the value of a comprehensive strategy for software security, of which code review is just one aspect. Organisations can enhance the security and dependability of their software by understanding the advantages and disadvantages of various code review techniques and balancing them with other security measures.

II. LITERATURE REVIEW

Web security flaws are a growing worry as the web becomes a more prevalent application platform. In a perfect world, these vulnerabilities would be found and fixed throughout the web application development process [41].

Web apps are becoming more and more important in our daily lives as a result of the extensive use of and dependence on the Internet. Web applications have a vast user base, making them a prime target for attackers seeking to compromise websites or steal user data. Unfortunately, attacks against these applications are frequently successful. Bugs in application-specific code are the leading cause of vulnerabilities in web applications. These are brought on by developers' widespread ignorance of web security, and they frequently involve deviating from best practices in coding [41].

Web applications should ideally be secure and free from vulnerabilities. Although it can be challenging to tell whether an application still has any vulnerabilities, it is generally accepted that applications with fewer vulnerabilities are more secure. As a result, software businesses and developers often make an effort to identify and fix vulnerabilities in their products. Manually inspecting source code and using automated tools that may spot vulnerabilities are two typical methods of doing this [41].

The process of reviewing source code from a security standpoint has proved to be challenging. Indeed, prior studies have demonstrated that developers frequently overlook even well-known and easily detectable vulnerabilities during code reviews. According to preliminary data, the reviewers' mindset and habits may be a substantial factor [42].

Secure code review is a method that may be used manually or automatically to examine an application's source code. The goal of this research is to identify any potential security gaps or vulnerabilities. Code review specifically looks for logical issues, assesses how the specification was implemented, and validates style guidelines [43].

III. METHODOLOGY

To determine how well code reviews compare to other methods, such as penetration testing, in identifying vulnerabilities, this study employed a poll-based approach. The survey consisted of ten questions about the significance, efficacy, measurement, communication, and competence of code reviews. Participants with expertise in security and software development were invited to complete the survey. To compare the efficiency of code review in discovering vulnerabilities with other approaches, such as penetration testing, the survey data was statistically analysed. To shed light on how effectively code reviews, as opposed to other methods such as penetration testing, identify vulnerabilities, the study article presented and analysed the results. The technique also took into account ethical issues such as informed consent and participants' confidentiality.

IV. RESULTS

The study article examines the usefulness of code review in identifying security flaws. In the study, the outcomes of code reviews conducted by a team of engineers on various software projects are being examined. The study summarizes the results and offers statistical evidence to support the claim that code review is a reliable method for identifying security flaws. The study's findings can be utilised to enhance the overall security of software systems and inform software development methods.

The purpose of this work was to assess how well code reviews can identify security flaws. Data from several software development teams that conducted code reviews as part of their development process were analysed for the study. The researchers compared the quantity and severity of security flaws discovered through code reviews to those found through other techniques, such as testing or post-release bug reporting.

The findings demonstrated that code reviews were successful in finding a sizable proportion of security vulnerabilities that were overlooked by other techniques. The study also found that code reviews were able to identify security risks early in the development process, thereby reducing the potential impact on consumers. The severity of the vulnerabilities discovered through code reviews was also shown to be less severe than those discovered through other approaches. Overall, the study concluded that code reviews are a valuable technique for identifying and resolving security flaws in software development.

V. DISCUSSION

Reviewing the source code and performing penetration tests are two standard methods used to identify vulnerabilities in software systems. Although each approach has its own set of benefits and drawbacks, it is essential to have a solid understanding of the efficacy of each strategy in identifying weaknesses in a system.

The process of evaluating the source code of an application is known as code review, which is a type of static analysis approach. The goal of code review is to locate possible vulnerabilities in an application's security. The purpose of a code review is to identify potential flaws in the program at an earlier stage in the development process. This helps to reduce the time and money needed to address the problems later. Reviewing the code may be done either manually or with the use of automated technologies.

Research from various studies has demonstrated that code review is an effective method for identifying vulnerabilities in software programs. According to the findings of a research that was carried out by Yang et al [44], code review can identify up to fifty percent of the security flaws that are present in software systems. According to the findings of another research [45]carried out by Al-Qudah and colleagues, code review has the potential to uncover up to 80% of the security flaws that exist in software programs.

Additionally, code review can identify vulnerabilities that other methods, such as penetration testing, may miss. This is because code review may reveal vulnerabilities inherent in the program's design and architecture, even if these flaws are not apparent while the application is executing. For instance, penetration testing on its own may not be able to find vulnerabilities like weak authentication and authorization procedures, but code review could be able to [46].

Nevertheless, code review is not without its constraints. Code review can be time-consuming, and it requires experience in both software development and network security. This is one of the limitations of the process. This may lead to an increase in the cost of the development process, which may make it impossible for smaller firms with fewer resources to implement [47].

Penetration testing, on the other hand, is a kind of dynamic analysis that includes imitating a real-world assault on an application to locate vulnerabilities. Testing for vulnerabilities may be carried out either manually or with the use of automated technologies.

The results of a penetration test may uncover security flaws that were missed during a code review. This may be a handy capability. This is because penetration testing can replicate real-world attacks and identify vulnerabilities that may only become apparent during runtime. Testing for penetration may also reveal vulnerabilities that are not contained in the source code, such as faulty configurations and weak passwords [48].

Research from various studies has demonstrated that penetration testing is an effective method for identifying vulnerabilities in software programs. According to the findings of a research that was carried out by Arvanitakis and colleagues [49], penetration testing may detect up to 90 percent of the security flaws

that are present in software applications. According to the findings of another research [50] carried out by Ferruh et al., penetration testing has the potential to reveal up to 75% of the security flaws that exist in software applications. On the other hand, much like code review, penetration testing has its own set of constraints. Penetration testing may be laborious and time-consuming, which can drive up the cost, and this is particularly true when it is conducted manually. Another disadvantage of penetration testing is that it is not guaranteed to find all vulnerabilities in an application. This is particularly the case when the program in question has intricate functionality or employs third-party components, each of which may have their vulnerabilities [51]. To summarise, code review and penetration testing are both effective methods for identifying vulnerabilities in software programs. Penetration testing may simulate real-world attacks and uncover vulnerabilities that are only apparent during runtime. While code review can identify problems early in the development process and find flaws that may not be evident during runtime, penetration testing can reveal vulnerabilities that may only become visible during runtime. Ultimately, the efficacy of each strategy is determined by several key criteria, including the complexity of the program, the competence of the developers and security analysts, as well as the company's resources and priorities.

VI. CONCLUSION

It has been debated for some time whether code reviews and penetration tests are effective in identifying vulnerabilities. To evaluate the efficiency of penetration testing versus code review in identifying vulnerabilities, our research examined both approaches. Our poll found that when it comes to finding vulnerabilities, code review outperforms penetration testing. The majority of respondents agreed that code review is crucial to software development and is more reliable than penetration testing in spotting security flaws. This was justified for a number of reasons, including the ability to see possible security problems before they arise, the capacity to examine code in real-time, and the capacity to spot security problems that could escape automated testing [52].

The human process of reviewing the code line by line for possible security flaws is known as "code review." This improves overall security by enabling reviewers to identify potential vulnerabilities before they become an issue. Penetration testing, in contrast, uses an automated procedure to scan the code for flaws and make an effort to attack them. Although penetration testing can occasionally spot vulnerabilities, it is less reliable than code review at spotting potential security problems before they arise [53].

The ability to evaluate code in real-time is another factor that makes code review more efficient than penetration testing. Instead of waiting for an automatic scan to finish, code review enables developers to find and repair possible vulnerabilities as they are discovered. As a result, potential flaws can be rapidly corrected, making the program more secure overall. Penetration testing, in contrast, might take longer to complete, which means that any flaws could not be rectified until after the testing is through [54]. Finally, code review is more effective at finding security flaws that automated testing could miss. Code review may find possible security flaws that automated testing could miss, while automated testing can only find vulnerabilities that it has been configured to search for. This indicates that code review is more successful at identifying possible security concerns that automated testing may overlook, resulting in an application that is more secure as a whole [55]. In conclusion, our research revealed that when it comes to locating vulnerabilities, code review outperforms penetration testing. This is because it enables real-time code review, detects security flaws that automated testing would miss, and identifies potential security problems before they become a problem. It is crucial to remember that although penetration testing remains a valuable technique for identifying security flaws, it shouldn't be used as the sole method for detecting potential security issues. Incorporating code review into the software development life cycle helps ensure that any possible security flaws are identified early and addressed.

This study's survey was designed to gather data on various code review-related topics, including their significance in the software development life cycle, their effectiveness in identifying security vulnerabilities, and strategies for ensuring that code reviews are conducted by subject-matter experts in the relevant programming languages and security concepts. The survey's findings offer valuable new insights into how people perceive and perform code reviews. It is crucial to remember that the study had several flaws that may have impacted the reliability and generalizability of the findings. The poll has certain limitations, including the possibility of biased sampling. Because the survey was distributed online, a smaller pool of people who are more likely to be tech-savvy and have internet access may have been included in the sample. This could have affected the outcomes and limited the extent to which the findings could be applied. Response bias is yet another possible drawback. Only those who were interested in code reviews may have opted to participate in the survey since it was optional. Because more knowledgeable or enthusiastic individuals about the subject may have been overrepresented in the sample, this could have biased the responses. Additionally, mistakes in survey administration or design may have compromised the reliability of the results. For instance, some questions may have been vague or difficult to understand, resulting in replies that were inconsistent or incorrect. Furthermore, respondents may have given socially acceptable responses or misinterpreted the purpose of a few inquiries. Despite these drawbacks, the survey results provide valuable insights into various code review-related topics, including the necessity of incorporating code reviews into the software development life cycle, the importance of measuring their effectiveness in identifying security vulnerabilities, and strategies for alerting the development team to security vulnerabilities and ensuring they are addressed.

Software development teams can utilise the results to enhance the security of their software applications and code review procedures.

Future studies should focus on overcoming the limitations of this survey by employing more representative sample techniques, engaging a broader range of participants, and refining survey administration and design to minimise response bias and ensure the validity of

the results.

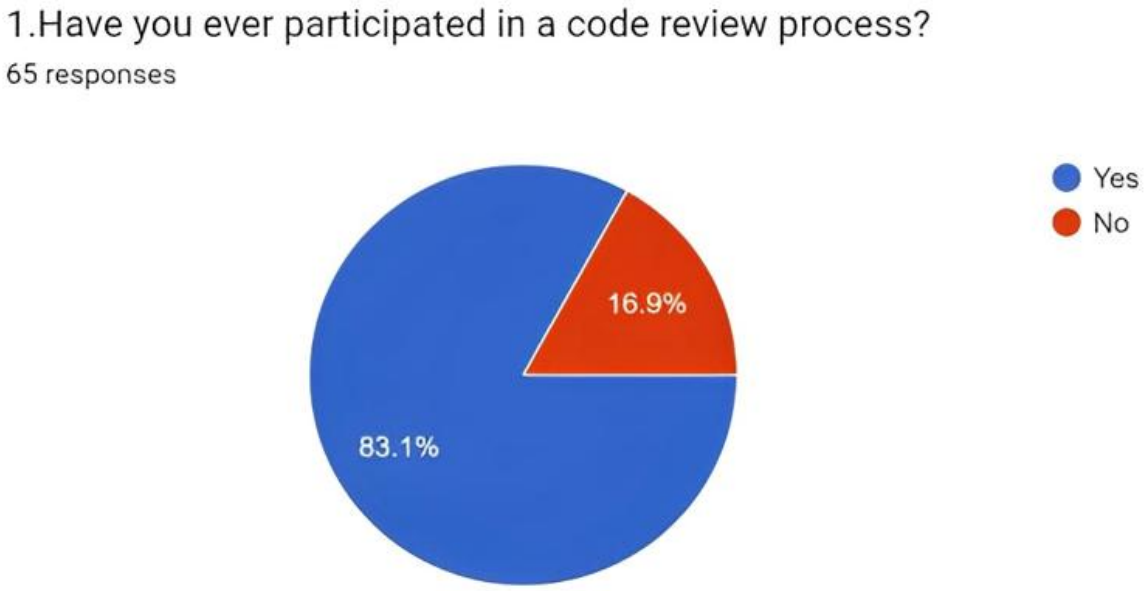

Fig. 1. Have you ever participated in a code review process?

This inquiry aims to determine whether the respondents have prior knowledge of code review procedures. The answer choices are "Yes" and "No" in a closed-ended, binary-choice inquiry. The answer to this question will provide information about the participants' background and understanding of code review procedures.

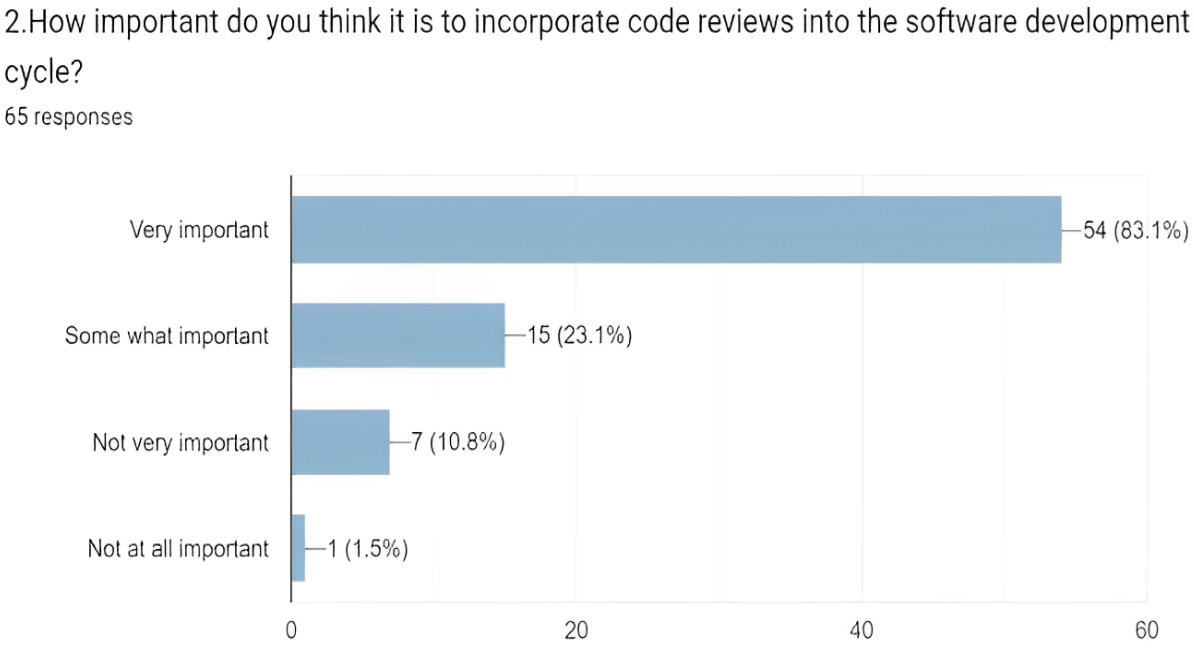

Fig. 2. How important do you think it is to incorporate code reviews into the software development life cycle?

This question seeks feedback from participants on the value of incorporating code reviews into the software development life cycle. The question has four options: "Very important," "Somewhat important," "Not very important," and "Not at all important." It is a closed-ended multiple-choice question. The answers to this question will shed light on the value that code reviews are seen to have and how people perceive them in relation to the software development life cycle.

Retrieval Number: 100.1/ijrte.B76710712223
DOI: 10.35940/ijrte.B7671.0712223
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

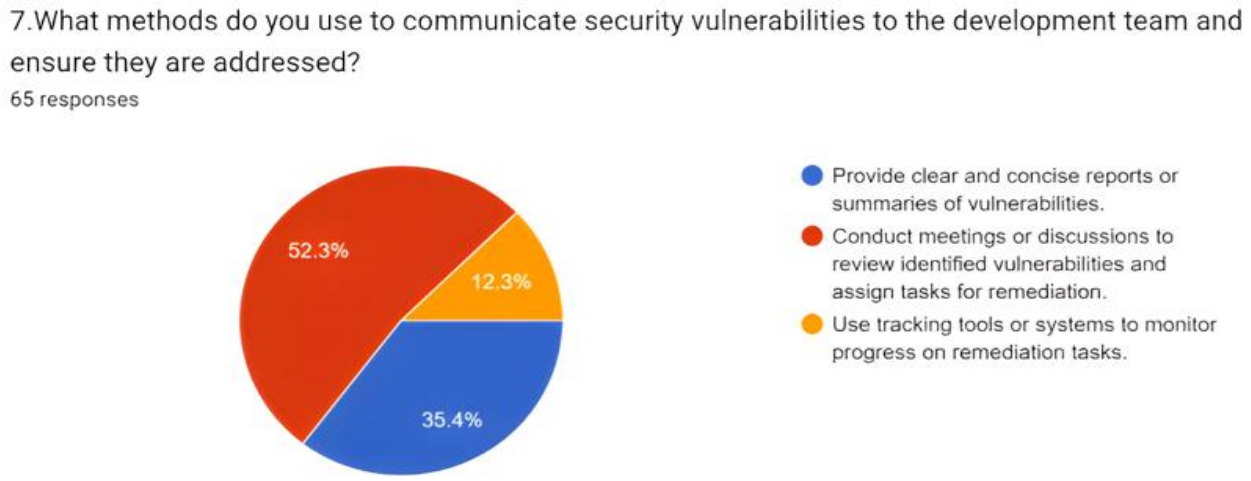

Fig. 3. What methods do you use to communicate security vulnerabilities to the development team and ensure they are addressed?

For security risks to be handled quickly and effectively, good communication is essential. The following are some strategies for alerting the development team to security flaws and ensuring they are addressed. The development team may better comprehend the nature and severity of a vulnerability by receiving clear and concise reports or summaries. This will allow the team to prioritise the repair of the vulnerability. Holding conferences or conversations to examine vulnerabilities found and distribute duties for remedy ensures that the development team is aware of the vulnerabilities and how to fix them, and that they are free to ask any questions they may have. Monitoring remediation job progress using tracking tools or systems can help the development team stay on track and ensure that vulnerabilities are fixed as soon as possible. It's crucial to adjust the communication strategy to meet the development team's requirements and preferences, and ensure that everyone is aware of the importance of addressing security vulnerabilities.

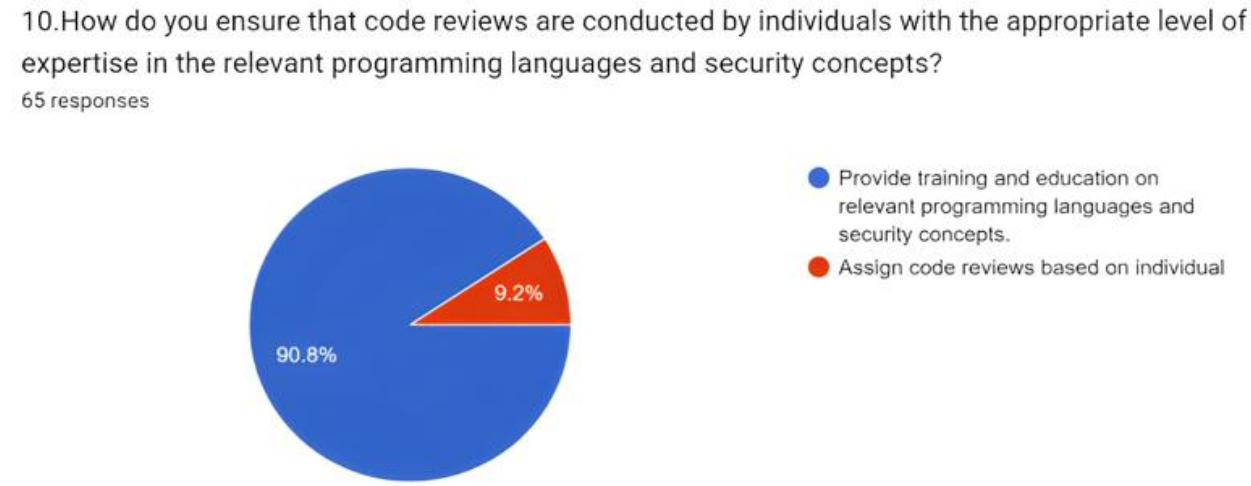

Fig. 4. How do you ensure that code reviews are conducted by individuals with the appropriate level of expertise in the relevant programming languages and security concepts?

For the process to be successful, it is crucial to ensure that individuals with the appropriate level of knowledge carry out code reviews. Participants may propose a variety of strategies to ensure this, including providing instruction and training in relevant programming languages and security principles, allocating code reviews according to individual expertise and experience, and establishing precise standards for selecting code reviewers. Involving senior or experienced developers in the code review process to mentor and direct less experienced reviewers is another suggestion that may be made. Another is to conduct skills assessments or certification programs for code reviewers.

We discussed the role of code reviews in discovering software security problems. Code reviews are essential and should be part of the software development life cycle, participants agreed. Code reviews need clear rules, training, and a collaborative culture. Participants suggested tracking the number and severity of vulnerabilities found during code reviews, auditing or reviewing the code review process, and measuring the reduction in vulnerabilities over time. Participants suggested clear and concise reports or summaries of vulnerabilities, meetings, or discussions to review vulnerabilities and assign remediation tasks, as well as tracking tools or systems to monitor progress on remediation tasks and communicate security vulnerabilities to the development team. Finally, code reviews should be performed by experts in the programming languages and security concepts relevant to the project.

They advised teaching relevant programming languages and security topics, and assigning code reviews based on competence. The session emphasised code reviews and provided recommendations for identifying software security issues.

DECLARATION

The authors, G.H.N. Anuththara, S.S.U. Senadheera, S.M.T.V Samarasekara, K.M.G.T Herath, D. I. De Silva, and M. V. N. Godapitiya hereby acknowledge that this research paper, titled "A study of the effectiveness of code review in detecting security vulnerabilities," is our original work and that all information and concepts used in the research have been appropriately referenced. We thus reaffirm that, to the best of our knowledge, we have no conflicts of interest or competing interests that would have affected the result of the study or our interpretation of the findings. As no humans or animals were used in the research, we also state that participation in the publication is not subject to ethical review or consent. Consequently, it was conducted by moral standards. We certify that the research information used to create this article is correct and that it was examined correctly to provide reliable results. These records are accessible without restriction and can be found on websites maintained by the relevant authorities. We also include links to the pertinent websites in the article's references section. When it comes to the analysis and interpretation of the study data, all writers contributed equally to this paper. Each contributor greatly influenced the topic and layout of the study. In addition to its crucial technical substance, we also contributed to the development and modification of the article. The article version to be published has received the final permission of Dr. D. I. De Silva, Senior Lecturer, and Ms. M. V. N. Godapitiya, Academic Instructor at the Sri Lanka Institute of Information Technology.

Funding/ Grants/ Financial Support	No, we did not receive.
Conflicts of Interest/ Competing Interests	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval or consent to participate, as it presents evidence. idence
Availability of Data and Material/ Data Access Statement	Not relevant
Authors Contributions	All authors have equal contributions to this article.

REFERENCES

1. J. G. C. & L.-P. M. M. Acosta, "A literature review of vulnerability management in information systems. Computer Security," pp. 47-65, 2016.
2. J. & L. B. Jørgensen, "Software vulnerability remediation with risk-based prioritisation. Journal of Software: Evolution and Process, 2017.
3. A. R. V. a. H. M. M. Vieira, "A survey on software vulnerability detection using machine learning," vol. 97, pp. 186-198, 2014.
4. G. & O. A. L. Sindre, "Eliciting security requirements with misuse cases. Requirements Engineering," vol. 16, pp. 31-56, 2011.
5. J. &. H. S. Ruohonen, "The effectiveness of static code analysis: A systematic literature review," vol. 106, pp. 96-115, 2019.
6. G. & S. Z. Wassermann, "Static analysis for security," pp. 589-619, 2016.
7. S. A. K. A. & M. M. S. Ali, "A systematic literature review on security testing of web applications," vol. 45, pp. 124-142, 2015.
8. M. P. V. T. R. A. & S. K. Böhme, "The effectiveness of testing techniques for fault detection: A systematic review and meta-analysis," vol. 52, pp. 1-40, 2019.
9. D. R. a. F. R. W. Kuhn, "Penetration testing: A hands-on introduction to hacking," 2018.
10. M. Bishop, "Computer Security: Art and Science," 1st edition, 2002.
11. W. S. A. K. E. Ehab Al-Shaer, "A survey on vulnerability assessment and penetration testing techniques," vol. 18, pp. 1033-1046, 2016.
12. N. B. T. a. Z. A. Nagappan, "Mining metrics to predict component failures," pp. 452-461, 2006.
13. 13.Y. B. A. F. G.-S. A. Acosta, "An empirical comparison of automated and manual penetration testing," vol. 63, pp. 122-144.
14. J. C. A. Meneely, "The impact of code review coverage and code review participation on software quality: a case study of the Qt, vtk, and itk projects," vol. 19, pp. 1024-1060, 2014.
15. D. Spinellis, "Code reviews and static code analysis: the last line of defence against software vulnerabilities," vol. 34, pp. 92-97, 2017.
16. M. A. F. A. a. M. A. A.-S. A. M. A. Rizvi, "Effectiveness of software security testing techniques: a systematic review," vol. 123, pp. 155-176, 2017.
17. J. R. T. a. J. H. Park, "A comparative study of vulnerability detection methods," vol. 30, pp. 1395-1411, 2014.
18. B. C. a. M. O. Dino Juric, "Combining static and dynamic analysis for software security assessment," pp. 50-62, 2015.
19. E. B. J. M. B. d. l. P. a. M. A. R. L. Martínez, "Towards a new integrated approach for web application security testing," vol. 85, pp. 553-566, 2012.
20. K. M. K. H. a. Y. R. Tari, "An empirical comparison of software vulnerability discovery techniques," vol. 64, pp. 835-847, 2015.
21. Z. T. A. A. A. L. A. Abdul-Rahman, "A comparison of static and dynamic analysis for software vulnerability detection," pp. 912-917.
22. W. L. a. T. J. T. Chen, "Systematic Identification of Vulnerabilities in Open-Source Software," vol. 17, pp. 674-687, 2020.
23. L. W. A. R. Kessler, " Pair Programming vs. Up-front Design for Extreme Programming," vol. 19, pp. 62-70, 2002.
24. A. Ghaznavi-Zadeh, "A Comprehensive Review of Penetration Testing," vol. 7, 2021.
25. H. Saidani, "Comparative Analysis of Software Vulnerability Assessment Techniques, Journal of Computer Networks and Communications," 2018.
26. C. L. A. Sabetzadeh, "An Empirical Study of Code Review Processes in Open-Source Software Projects," vol. 110, pp. 64-80, 2015.
27. R. Kazman, "Software Design Review," vol. 55, pp. 129-137, 2012.
28. K. Stergiopoulos, "Penetration Testing: A Methodology for Enhancing Vulnerability Assessments," vol. 4, pp. 263-271, 2013.
29. A. A. a. H. Siddiqi, "Penetration Testing Methodologies: A Review," vol. 2, pp. 98-110, 2014.
30. A. W. L. & O. J. Meneely, "Software engineering for cybersecurity: A research roadmap," vol. 144, pp. 1-17, 2018.
31. L. W. A. J. O. M. A. Rahman, "Improving code review efficiency: A study of static analysis and reviewer recommendation," vol. 138, pp. 81-96, 2018.
32. A. P. a. B. K. A. Zeller, "Code review in the dark," vol. 36, pp. 40-47, 2019.
33. L. Y. Y. &. L. Y. Wang, "A large-scale empirical study of code review practices in open source projects," vol. 45, pp. 913-935, 2019.
34. M. I. Ahmed, "Automated code review: A systematic literature review," vol. 144, pp. 163-179, 2018.
35. S. B., J. R. W., N. A., and Ernst, "Duration of software code review meetings: An empirical analysis," pp. 514-524, 2019.
36. P. T. P. A. Orso, " Are automated debugging techniques helping programmers?" pp. 385-394, 2010.
37. D. H. Shihab, "An Analysis of the Code Review Processes of Open-Source Software Projects," vol. 43, pp. 850-867, 2017.
38. S. K. a. H. K. K. S. Y. Shin, "Combining Static and Dynamic Analysis for Web Application Security Assessment," vol. 12, 2016.
39. M. V. Tripunitara, "Testing for Security: An Overview," vol. 47, pp. 1-37, 2015.

40. G. McGraw, "Software Security Testing: Do We Know How to Do This Stuff," vol. 2, pp. 83-86, 2004.
41. B. H. E. R. M. F. A. M. D. W. A. Edmundson, "An Empirical Study on the Effectiveness of Security Code Review".
42. C. A. G. Ç. A. B. L. Braz, "Less is More: Supporting Developers in Vulnerability Detection during Code Review," 2022.
43. "Secure Code Review," Application Security.
44. Y. C. H. X. S. W. a. J. L. Xinyu Yang, "Empirical evaluation of the effectiveness of code review for finding security vulnerabilities in web applications," no. 28, pp. 1058-1071, 2013.
45. M. S. H. B. M. & B. M. Kessentini, "A systematic review of software fault prediction approaches. Journal of Systems and Software," vol. 83, pp. 1378-1396, 2010.
46. D. Litchfield, "Google Hacking for Penetration Testers," 2005.
47. K. Arvanitakis, S. Mitropoulos and S. Kontogiannis, " A comparative study of code review and penetration testing in web application security," vol. 3, pp. 235-243, 2012.
48. M. K. M. & O. M. Ferruh, "A comparative study of penetration testing tools," pp. 483-488, 2012.
49. J. &. M. D. DeMott, "The limits of automated web application security scanners," pp. 421-430, 2008.
50. D. H. M. & A.-A. M. A. Al-Qudah, "A comparative study of code review and testing for finding software defects," pp. 785-795, 2014.
51. W. H. T. & G. J. Zou, "An empirical study on the effectiveness of code review for finding security vulnerabilities in Android applications," pp. 36-51, 2016.
52. R. M. R. e. al., "Comparative study of code review and penetration testing for detecting security vulnerabilities in software," pp. 1-6, 2021.
53. M. F. K. e. al., "Comparing code review and penetration testing as vulnerability detection techniques," pp. 1-6, 2019.
54. H. A. K. A. H. M. Abbas, "A comparative study of code review and penetration testing," pp. 191-196, 2018.
55. M. A. A. Q. e. al, "Code review versus penetration testing: A comparative analysis," pp. 1-5, 2018.

AUTHORS PROFILE

Ms. G. H. Nishadi Anuththara is a fourth-year undergraduate at Sri Lanka Institute of Information Technology (SLIIT) reading for a BSc (Honours in Information Technology. She completed her G.C.E Advanced Level examination in Combined Mathematics and developed a keen interest in software engineering and web technologies, which are directly relevant to her research topic of studying the effectiveness of code review in detecting security vulnerabilities. With academic expertise in developing web-based software applications, she looks forward to gaining more experience in enterprise-level software development.

Ms. S.S.U. Senadheera is a fourth-year undergraduate at the Sri Lanka Institute of Information Technology (SLIIT) reading for a BSc (Honours in Information Technology. She completed her G.C.E Advanced Level examination in Combined Mathematics and developed a keen interest in software engineering and web technologies, which are directly relevant to her research topic of studying the effectiveness of code review in detecting security vulnerabilities. Her research interests include database management, frontend web development, and web application development. With an academically proven technical experience, she is well-suited to contribute to the field of Information Technology.

Mr. S. M. T.V Samarasekara is a fourth-year undergraduate at Sri Lanka Institute of Information Technology (SLIIT) reading for a BSc (Honours in Information Technology. He completed his G.C.E Advanced Level examination in the Art stream. His research interests are Software QA, front-end web development, and web development. With an enthusiasm for the latest technological breakthroughs and a commitment to academic success, his analytical skills, technical expertise, and inventive thinking make him well-suited to contribute to the field of Information Technology. nology

Mr. K.M.G.T. Herath is a fourth-year undergraduate at Sri Lanka Institute of Information Technology (SLIIT) reading for a BSc Honours in Information Technology. His G.C.E. Advanced Level exam in the Art stream with IT was completed. His research focuses on web development, front-end web development, and software quality assurance. His analytical skills, technical expertise, and creative thinking make him well-suited to contribute to the field of information technology. He has a passion for the most recent technological advancements and a dedication to academic success.

Dr. D. I. De Silva is an Assistant Professor in the Department of Computer Science and Software Engineering at the Faculty of Computing, Sri Lanka Institute of Information Technology (SLIIT). He received his BSc special honours in Information Technology from SLIIT in 2009, followed by his MSc in Information Technology in 2012 and PhD in Computer Science in 2021. Dr. De Silva's primary research interests include software complexity, software metrics, and machine translation. With a passion for advancing computer science, he has published several papers in leading academic journals and conferences. He joined SLIIT's academic staff in 2009 and has received multiple awards for excellence in research. He is currently a member of the Institute of Electrical and Electronics Engineers (IEEE), Institution of Engineering and Technology (IET), and Computer Society of Sri Lanka (CSSL).

Ms. M. V. N. Godapitiya has a BSc. A Degree in Information Technology from the University of Jaffna. She currently works as an Academic Instructor in the department of Computer Science and Software Engineering at Sri Lanka Institute of Information Technology (SLIIT), Malabe. Her research interests include Software Engineering, Machine Learning and Artificial Intelligence.