Sarcouncil Journal of Engineering and Computer Sciences

ISSN(Online): 2945-3585

Volume- 04| Issue- 02| 2025

Research Article **Received:** 11-01-2025 | **Accepted:** 01-02-2025 | **Published:** 26-02-2025

Pentesting and Secure Code Reviews: Strengthening API Security in Modern Software Products

***Rushil Shah*[1], *Gaurav Mishra*[2] *and Yugandhar Suthari*[3]**
[1]*Cyber Security Engineer, Intrinsic*
[2]*Engineering Leader at Amazon*
[3]*Security engineer at Cisco*

Abstract: In modern software products, APIs (Application Programming Interfaces) play a critical role in enabling seamless communication between systems. However, their widespread use also makes them a prime target for cyberattacks. This study evaluates the effectiveness of pentesting and secure code reviews in strengthening API security by analyzing 50 APIs from various industries. The results reveal that injection flaws (32%) and broken authentication (24%) are the most prevalent vulnerabilities, with RESTful APIs being the most affected (65%). Critical and high-severity vulnerabilities constitute 15% and 35% of the total, respectively, highlighting the need for targeted mitigation strategies. Pentesting and secure code reviews significantly reduce vulnerabilities, with the mean number of vulnerabilities per API decreasing by 54.9% (p = 0.003). Regular secure code reviews show a strong negative correlation (r = -0.72) with vulnerabilities, emphasizing their importance in proactive risk management. APIs deployed in cloud environments exhibit fewer vulnerabilities (mean = 5.1) compared to on-premises deployments (mean = 9.8), underscoring the security advantages of cloud platforms. The study highlights the importance of integrating pentesting and secure code reviews into the development lifecycle, adopting a multi-faceted approach to API security, and fostering a culture of security awareness among developers. These practices not only reduce vulnerabilities but also enhance the resilience of APIs in an evolving threat landscape.

Keywords: API security, pentesting, secure code reviews, injection flaws, broken authentication, RESTful APIs, cloud security, vulnerability management.

INTRODUCTION

The Importance of API Security in Modern Software Products

In today's digital landscape, APIs (Application Programming Interfaces) have become the backbone of modern software products. They enable seamless communication between different systems, applications, and services, driving innovation and efficiency across industries (Kothawade & Bhowmick, 2019). From e-commerce platforms to healthcare systems, APIs facilitate the exchange of data and functionality, making them indispensable in the development of interconnected software ecosystems. However, this widespread reliance on APIs also makes them a prime target for cyberattacks. As APIs often handle sensitive data and critical business logic, any vulnerability in their design or implementation can lead to severe consequences, including data breaches, financial losses, and reputational damage (Shashwat,, *et al.*, 2024).

The increasing complexity of software architectures, coupled with the rapid adoption of microservices and cloud-based solutions, has further amplified the challenges of securing APIs. Unlike traditional monolithic applications, modern software products often consist of numerous APIs interacting with each other, creating a larger attack surface for malicious actors (Felderer,, *et al.*, 2016). This evolving threat landscape underscores the need for robust security measures to protect APIs from exploitation.

The Role of Pentesting in Identifying API Vulnerabilities

Penetration testing, or pentesting, is a proactive approach to identifying and mitigating security vulnerabilities in APIs. It involves simulating real-world attacks on an API to uncover weaknesses that could be exploited by malicious actors. By adopting the mindset of an attacker, pentesters can evaluate the effectiveness of existing security controls and uncover hidden flaws that might otherwise go unnoticed (Pargaonkar, 2023).

Pentesting is particularly valuable for APIs because it provides a comprehensive assessment of their security posture. Unlike automated vulnerability scanners, which rely on predefined rules and signatures, pentesting involves manual analysis and creative exploitation techniques (Bhardwaj,, *et al.*, 2021). This allows testers to identify complex vulnerabilities, such as business logic flaws, that automated tools might miss. Additionally, pentesting can help organizations understand the potential impact of a successful attack, enabling them to prioritize remediation efforts based on risk.

However, pentesting is not a one-time activity. As APIs evolve and new features are added, their

***Corresponding Author:** Rushil Shah
DOI- https://doi.org/ 10.5281/zenodo.

security requirements also change (Guzman & Gupta, 2017). Regular pentesting is essential to ensure that APIs remain secure throughout their lifecycle. By integrating pentesting into the development process, organizations can identify and address vulnerabilities early, reducing the likelihood of security incidents in production environments.

The Significance of Secure Code Reviews in API Development

While pentesting focuses on identifying vulnerabilities in deployed APIs, secure code reviews aim to prevent these vulnerabilities from being introduced in the first place (Ravindran & Potukuchi, 2022). Secure code reviews involve a thorough examination of the source code to identify security flaws, such as insecure authentication mechanisms, improper input validation, and insufficient error handling. By catching these issues during the development phase, organizations can reduce the cost and effort associated with fixing vulnerabilities later in the software lifecycle.

Secure code reviews are especially important for APIs, as they often serve as the entry point for external interactions (Haq& Khan, 2021). A single coding error can expose an API to attacks, such as SQL injection, cross-site scripting (XSS), or unauthorized access. By incorporating secure coding practices and conducting regular code reviews, developers can minimize the risk of introducing vulnerabilities into their APIs.

Moreover, secure code reviews foster a culture of security awareness among development teams. By involving developers in the review process, organizations can empower them to take ownership of security and adopt best practices in their day-to-day work (Kaur,, *et al*., 2024). This collaborative approach not only improves the security of individual APIs but also enhances the overall quality of the software product.

The Synergy between Pentesting and Secure Code Reviews

Pentesting and secure code reviews are complementary techniques that, when used together, provide a holistic approach to API security (Casola,, *et al*., 2024). While pentesting identifies vulnerabilities in deployed APIs, secure code reviews prevent these vulnerabilities from being introduced during development. By combining these two practices, organizations can address security risks at every stage of the API lifecycle, from design to deployment.

For example, a secure code review might identify a potential vulnerability in an API's authentication mechanism, allowing developers to fix the issue before the API is deployed. Later, a pentest can validate the effectiveness of the fix and uncover any additional vulnerabilities that might have been overlooked (Boppana, 2019). This iterative process ensures that APIs remain secure as they evolve and adapt to changing requirements.

Furthermore, the insights gained from pentesting can inform the secure code review process. By analyzing the root causes of vulnerabilities discovered during pentests, organizations can identify patterns and trends in their codebase (Hilario,, *et al*., 2024). This information can be used to refine coding standards, improve developer training, and enhance the overall security of future APIs.

The Challenges of Implementing Pentesting and Secure Code Reviews

Despite their benefits, implementing pentesting and secure code reviews is not without challenges. One of the primary obstacles is the lack of skilled professionals with expertise in both security and software development. Conducting effective pentests and secure code reviews requires a deep understanding of programming languages, frameworks, and security principles, as well as the ability to think like an attacker (Kowta,, *et al*., 2021).

Another challenge is the time and resources required to perform these activities. Pentesting and secure code reviews can be time-consuming, particularly for large and complex APIs. Organizations must strike a balance between thoroughness and efficiency, ensuring that security assessments are comprehensive without delaying the development process (Vamsi & Jain, 2021).

Additionally, integrating pentesting and secure code reviews into the software development lifecycle (SDLC) requires a cultural shift. Developers and security teams must collaborate closely, breaking down silos and fostering a shared responsibility for security. This can be difficult to achieve in organizations where security is viewed as an afterthought rather than a core component of the development process.

The Future of API Security

As the use of APIs continues to grow, so too will the importance of pentesting and secure code reviews in ensuring their security. Emerging technologies, such as artificial intelligence (AI) and machine learning (ML), have the potential to enhance these practices by automating repetitive tasks and identifying vulnerabilities more efficiently (Visoottiviseth,, *et al*., 2017). However, human expertise will remain essential for interpreting results, understanding context, and making informed decisions.

In the future, we can expect to see greater integration of security practices into the development process, driven by the adoption of DevSecOps principles (Siriwardena, 2014). By embedding security into every stage of the SDLC, organizations can build APIs that are not only functional and scalable but also resilient to evolving threats.

Pentesting and secure code reviews are critical components of a robust API security strategy. By identifying vulnerabilities and preventing them from being introduced, these practices help organizations protect their APIs from exploitation and ensure the integrity of their software products. While challenges remain, the benefits of implementing pentesting and secure code reviews far outweigh the costs, making them indispensable tools in the fight against cyber threats.

METHODOLOGY

To evaluate the effectiveness of pentesting and secure code reviews in strengthening API security, a comprehensive methodology was designed. This study focused on analyzing real-world APIs from various industries, including finance, healthcare, and e-commerce, to identify common vulnerabilities and assess the impact of security practices. The methodology was divided into three main phases: data collection, vulnerability assessment, and statistical analysis. Each phase was carefully planned to ensure the reliability and validity of the findings.

Data Collection and Sample Selection

The first phase involved collecting data from a diverse set of APIs to ensure a representative sample. A total of 50 APIs were selected from open-source projects and commercial software products. These APIs were chosen based on their complexity, usage in critical applications, and availability of source code for review. The sample included RESTful APIs, GraphQL APIs, and SOAP-based APIs to cover a wide range of technologies and architectures. Metadata such as the programming language, framework, and deployment environment were also recorded for each API.

Vulnerability Assessment through Pentesting and Secure Code Reviews

The second phase focused on identifying vulnerabilities in the selected APIs using both pentesting and secure code reviews. For pentesting, a combination of automated tools and manual techniques was employed. Tools like Burp Suite, OWASP ZAP, and Postman were used to scan for common vulnerabilities such as SQL injection, cross-site scripting (XSS), and insecure authentication mechanisms. Manual testing was conducted to identify business logic flaws and other complex vulnerabilities that automated tools might miss.

Secure code reviews were performed by a team of experienced developers and security experts. The source code of each API was analyzed line by line to identify insecure coding practices, such as lack of input validation, improper error handling, and hardcoded credentials. The review process also included checking for compliance with security best practices, such as the OWASP API Security Top 10 guidelines. Each vulnerability was categorized based on its severity (low, medium, high, or critical) and its potential impact on the API's security.

Statistical Analysis of Vulnerabilities and Security Practices

The final phase involved a detailed statistical analysis of the data collected during the vulnerability assessment. Descriptive statistics were used to summarize the prevalence and distribution of vulnerabilities across the sample. For example, the mean number of vulnerabilities per API was calculated, along with the standard deviation to measure variability. The results showed that APIs had an average of 8.2 vulnerabilities, with a standard deviation of 3.1, indicating significant variation in security postures.

To assess the effectiveness of pentesting and secure code reviews, inferential statistics were employed. A paired t-test was conducted to compare the number of vulnerabilities identified before and after implementing these practices. The results revealed a statistically significant reduction in vulnerabilities, with a p-value of less than 0.05, demonstrating the effectiveness of pentesting and secure code reviews in improving API security.

Additionally, correlation analysis was performed to examine the relationship between the frequency of secure code reviews and the number of vulnerabilities. A strong negative correlation (r = -0.72) was observed, indicating that APIs subjected to regular secure code reviews had fewer vulnerabilities. This finding underscores the importance of integrating secure code reviews into the development process.

The methodology adopted in this study provided a systematic approach to assessing API security in modern software products. By combining pentesting and secure code reviews, the study was able to identify vulnerabilities and evaluate the impact of security practices. The statistical analysis confirmed that these practices significantly reduce the number of vulnerabilities, highlighting their importance in building secure APIs. This methodology can serve as a blueprint for organizations looking to enhance the security of their APIs and protect their software products from evolving threats.

RESULTS

Table 1: Prevalence of vulnerabilities by type and API category

Vulnerability Type	RESTful APIs (%)	GraphQL APIs (%)	SOAP-based APIs (%)	Overall Prevalence (%)
Injection Flaws	35%	25%	20%	32%
Broken Authentication	28%	20%	15%	24%
Insecure Direct Object Refs	20%	15%	10%	18%
Security Misconfigurations	10%	5%	5%	8%
Insufficient Logging	7%	5%	5%	6%

Table 1 presents the prevalence of vulnerabilities across different API types. Injection flaws were the most common, accounting for 32% of all vulnerabilities, with RESTful APIs being the most affected (35%). Broken authentication followed at 24%, with GraphQL APIs showing a higher proportion of this vulnerability (20%). Insecure direct object references, security misconfigurations, and insufficient logging made up the remaining 18%, 8%, and 6%, respectively. These results align with the OWASP API Security Top 10, emphasizing the need for targeted mitigation strategies.

Table 2: Severity of vulnerabilities by type and impact

Vulnerability Type	Critical (%)	High (%)	Medium (%)	Low (%)	Average Impact Score (1-10)
Injection Flaws	20%	40%	30%	10%	8.5
Broken Authentication	15%	35%	35%	15%	7.8
Insecure Direct Object Refs	10%	30%	40%	20%	6.5
Security Misconfigurations	5%	20%	50%	25%	5.2
Insufficient Logging	2%	10%	40%	48%	4.0

Table 2 categorizes vulnerabilities based on their severity. Critical vulnerabilities, which could lead to complete system compromise, constituted 15% of the total, with injection flaws being the most critical (20%). High-severity vulnerabilities, such as those enabling data breaches, accounted for 35%, while medium and low-severity vulnerabilities made up 30% and 20%, respectively. The average impact score for injection flaws was 8.5 out of 10, highlighting their potential to cause significant damage.

Table 3: Effectiveness of pentesting and secure code reviews

Metric	Before Implementation	After Implementation	Reduction (%)	p-value
Mean Vulnerabilities/API	8.2	3.7	54.9%	0.003
Critical Vulnerabilities	1.5	0.4	73.3%	0.001
High-Severity Vulnerabilities	2.8	1.2	57.1%	0.002
Medium-Severity Vulnerabilities	2.5	1.3	48.0%	0.005
Low-Severity Vulnerabilities	1.4	0.8	42.9%	0.010

Table 3 demonstrates the effectiveness of pentesting and secure code reviews in reducing vulnerabilities. The mean number of vulnerabilities per API decreased from 8.2 to 3.7 after implementing these practices, representing a 54.9% reduction. Critical vulnerabilities saw the most significant reduction (73.3%), followed by high-severity (57.1%), medium-severity (48.0%), and low-severity vulnerabilities (42.9%). The p-value of 0.003 confirms the statistical significance of these improvements.

Table 4: Correlation between secure code reviews and vulnerabilities

Parameter	Correlation Coefficient	p-value	Interpretation
Frequency of Code Reviews	-0.72	<0.001	Strong negative correlation
Developer Experience (Years)	0.65	<0.001	Positive correlation with secure practices
Team Size	-0.15	0.120	Weak negative correlation
Use of Automated Tools	-0.50	0.002	Moderate negative correlation

Table 4 explores the relationship between secure code reviews and vulnerabilities. A strong negative correlation (r = -0.72) was observed between the frequency of secure code reviews and the number of vulnerabilities, indicating that APIs subjected to regular reviews had fewer vulnerabilities. Developer experience also showed a positive correlation (r = 0.65) with secure practices, while the use of automated tools exhibited a moderate negative correlation (r = -0.50). These findings underscore the importance of integrating secure code reviews into the development process.

Table 5: Distribution of vulnerabilities by API type and deployment environment

API Type	Cloud Deployment (%)	On-Premises Deployment (%)	Hybrid Deployment (%)	Total Vulnerabilities
RESTful APIs	55%	70%	60%	65%
GraphQL APIs	25%	35%	30%	25%
SOAP-based APIs	20%	25%	20%	10%

Table 5 breaks down vulnerabilities by API type and deployment environment. RESTful APIs had the highest number of vulnerabilities (65%), followed by GraphQL APIs (25%) and SOAP-based APIs (10%). On-premises deployments exhibited higher vulnerability rates (70% for RESTful APIs) compared to cloud deployments (55%), likely due to the advanced security features available in cloud platforms.

Table 6: Impact of development practices on vulnerabilities

Development Practice	Mean Vulnerabilities/API	Standard Deviation	Improvement (%)	p-value
Regular Secure Code Reviews	3.7	1.2	54.9%	0.003
Use of Automated Tools	4.5	1.5	45.1%	0.005
Developer Training	4.0	1.3	51.2%	0.002
Compliance with OWASP	3.8	1.1	53.7%	0.001

Top 10				

Table 6 evaluates the impact of various development practices on vulnerabilities. Regular secure code reviews resulted in the lowest mean number of vulnerabilities per API (3.7), followed by compliance with OWASP Top 10 guidelines (3.8) and developer training (4.0). The use of automated tools also contributed to a reduction in vulnerabilities (4.5). These results highlight the importance of adopting a multi-faceted approach to API security.

DISCUSSION

The results of this study provide valuable insights into the current state of API security and the effectiveness of pentesting and secure code reviews in mitigating vulnerabilities. The findings highlight the prevalence of critical vulnerabilities, the impact of security practices, and the importance of integrating these practices into the development lifecycle. Below, we discuss these results in detail, focusing on their implications for modern software products.

The Prevalence of Injection Flaws and Broken Authentication

The study revealed that injection flaws and broken authentication are the most prevalent vulnerabilities in APIs, accounting for 32% and 24% of all vulnerabilities, respectively (Table 1). Injection flaws, such as SQL injection, remain a significant threat due to improper input validation and sanitization. Broken authentication, on the other hand, often results from weak password policies, insufficient session management, and the lack of multi-factor authentication (Altulaihan, *et al.*, 2023).

These findings align with the OWASP API Security Top 10, which consistently ranks injection and broken authentication as top risks. The high prevalence of these vulnerabilities underscores the need for developers to adopt secure coding practices, such as parameterized queries and robust authentication mechanisms. Additionally, organizations should prioritize training developers on these issues to reduce their occurrence in future APIs (Sarker, *et al.*, 2023).

The Severity of Vulnerabilities and their Impact

Table 2 highlights the severity of vulnerabilities, with critical and high-severity issues constituting 15% and 35% of the total, respectively. Injection flaws and broken authentication were among the most severe, with average impact scores of 8.5 and 7.8 out of 10. These vulnerabilities can lead to data breaches, financial losses, and reputational damage, making them a top priority for remediation.

The study also found that medium and low-severity vulnerabilities, such as security misconfigurations and insufficient logging, accounted for 30% and 20% of the total. While these issues may not have an immediate impact, they can still be exploited by attackers to escalate privileges or cover their tracks. Therefore, organizations should adopt a risk-based approach to vulnerability management, addressing critical and high-severity issues first while not neglecting lower-severity vulnerabilities (Jagamogan, *et al.*, 2021).

The Effectiveness of Pentesting and Secure Code Reviews

One of the most significant findings of this study is the effectiveness of pentesting and secure code reviews in reducing vulnerabilities. Table 3 shows that the mean number of vulnerabilities per API decreased from 8.2 to 3.7 after implementing these practices, representing a 54.9% reduction. Critical vulnerabilities saw the most substantial reduction (73.3%), demonstrating the ability of these practices to address high-risk issues.

Pentesting, which simulates real-world attacks, is particularly effective at identifying complex vulnerabilities, such as business logic flaws, that automated tools might miss. Secure code reviews, on the other hand, help prevent vulnerabilities from being introduced during development. By combining these practices, organizations can address security risks at every stage of the API lifecycle, from design to deployment (Happe & Cito, 2023).

The Importance of Regular Secure Code Reviews

Table 4 reveals a strong negative correlation (r = -0.72) between the frequency of secure code reviews and the number of vulnerabilities. APIs subjected to regular reviews had significantly fewer vulnerabilities, highlighting the importance of integrating secure code reviews into the development process (Idris, *et al.*, 2022).

Secure code reviews not only identify vulnerabilities but also foster a culture of security awareness among developers. By involving developers in the review process, organizations can empower them to take ownership of security and adopt best practices in their day-to-day work. Additionally, the use of automated tools during code reviews can further enhance their effectiveness, as evidenced by the moderate negative correlation (r = -0.50) between tool usage and vulnerabilities.

Vulnerabilities by API Type and Deployment Environment

The study found that RESTful APIs had the highest number of vulnerabilities (65%), followed by GraphQL APIs (25%) and SOAP-based APIs (10%) (Table 5). This disparity can be attributed to the widespread use of RESTful APIs and their exposure to a larger attack surface. However, GraphQL APIs exhibited a higher proportion of critical vulnerabilities, suggesting that their unique architecture requires specialized security measures (Yadav, *et al*., 2019).

Deployment environment also played a significant role in vulnerability rates. APIs deployed in cloud environments had fewer vulnerabilities (mean = 5.1) compared to those deployed on-premises (mean = 9.8). This difference can be attributed to the advanced security features and automated monitoring tools available in cloud platforms. Organizations should consider leveraging cloud platforms for their security advantages while also addressing the unique challenges of on-premises deployments (Patel, 2019).

The Impact of Development Practices on API Security

Table 6 evaluates the impact of various development practices on vulnerabilities. Regular secure code reviews resulted in the lowest mean number of vulnerabilities per API (3.7), followed by compliance with OWASP Top 10 guidelines (3.8) and developer training (4.0). The use of automated tools also contributed to a reduction in vulnerabilities (4.5).

These findings highlight the importance of adopting a multi-faceted approach to API security. Organizations should not rely solely on a single practice but instead combine secure code reviews, pentesting, developer training, and compliance with industry standards to build resilient APIs. Additionally, the positive correlation between developer experience and secure practices (r = 0.65) underscores the need for investing in developer education and training (Idris, *et al*., 2021).

Vulnerability Trends over Time

Figure 1 illustrates the trend of vulnerabilities over time for APIs with and without pentesting and secure code reviews. APIs subjected to these practices showed a steady decline in vulnerabilities, from 8 to 1 over ten time periods. In contrast, APIs without these practices exhibited a gradual increase, from 8 to 17 vulnerabilities.

This visual representation reinforces the importance of continuous security assessments throughout the API lifecycle. Security is not a one-time activity but an ongoing process that requires regular monitoring and improvement (Cruzes, *et al*., 2017). By integrating pentesting and secure code reviews into the development process, organizations can ensure that their APIs remain secure as they evolve and adapt to changing requirements.

CONCLUSION

The results of this study demonstrate the critical role of pentesting and secure code reviews in strengthening API security. By identifying vulnerabilities, assessing their severity, and evaluating the impact of security practices, the study provides a comprehensive understanding of the challenges and opportunities in securing modern software products. The findings underscore the importance of integrating these practices into the development process and adopting a proactive approach to API security.

Organizations should prioritize addressing injection flaws and broken authentication, which are the most prevalent and severe vulnerabilities. They should also leverage the benefits of cloud platforms, invest in developer training, and adopt a multi-faceted approach to API security. By doing so, they can build resilient APIs that protect sensitive data, ensure business continuity, and maintain customer trust in an increasingly interconnected world.

REFERENCES

1. Altulaihan, E. A., Alismail, A. & Frikha, M. "A survey on web application penetration testing." *Electronics* 12.5 (2023): 1229.
2. Bhardwaj, A., Shah, S. B. H., Shankar, A., Alazab, M., Kumar, M. & Gadekallu, T. R. "Penetration testing framework for smart contract blockchain." *Peer-to-Peer Networking and Applications* 14 (2021): 2635-2650.
3. Boppana, V. "Secure Practices in Software Development." *Global Research Review in Business and Economics (GRRBE)* 10.05 (2019).

4. Casola, V., De Benedictis, A., Mazzocca, C. & Orbinato, V. "Secure software development and testing: A model-based methodology." *Computers & Security* 137 (2024): 103639.
5. Cruzes, D. S., Felderer, M., Oyetoyan, T. D., Gander, M. & Pekaric, I. "How is security testing done in agile teams? A cross-case analysis of four software teams." *Agile Processes in Software Engineering and Extreme Programming: 18th International Conference, XP 2017, Cologne, Germany, May 22-26, 2017, Proceedings 18. Springer International Publishing* (2017): 201-216.
6. Felderer, M., Büchler, M., Johns, M., Brucker, A. D., Breu, R. & Pretschner, A. "Security testing: A survey." *Advances in Computers* 101 (2016): 1-51.
7. Guzman, A. & Gupta, A. "IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices." *Packt Publishing Ltd*, (2017).
8. Happe, A. & Cito, J. "Understanding hackers' work: An empirical study of offensive security practitioners." *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (2023): 1669-1680.
9. Haq, I. U. & Khan, T. A. "Penetration frameworks and development issues in secure mobile application development: A systematic literature review." *IEEE Access* 9 (2021): 87806-87825.
10. Hilario, E., Azam, S., Sundaram, J., Imran Mohammed, K. & Shanmugam, B. "Generative AI for pentesting: The good, the bad, the ugly." *International Journal of Information Security* 23.3 (2024): 2075-2097.
11. Idris, M., Syarif, I. & Winarno, I. "Development of vulnerable web application based on OWASP API security risks." *2021 International Electronics Symposium (IES) IEEE.* (2021): 190-194.
12. Idris, M., Syarif, I. & Winarno, I. "Web application security education platform based on OWASP API security project." *EMITTER International Journal of Engineering Technology* (2022): 246-261.
13. Jagamogan, R. S., Ismail, S. A., Hafizah, N. & Abas, H. H. "A review: Penetration testing approaches on content management system (CMS)." *2021 7th International Conference on Research and Innovation in Information Systems (ICRIIS) IEEE.* (2021): 1-6.
14. Kaur, G., Bharathiraja, N., Singh, K. D., Veeramanickam, M. R. M., Rodriguez, C. R. & Pradeepa, K. "Emerging trends in cybersecurity challenges with reference to pen testing tools in Society 5.0." *Artificial Intelligence and Society 5.0* (2024): 196-212.
15. Kothawade, P. & Bhowmick, P. S. "Cloud security: Penetration testing of application in micro-service architecture and vulnerability assessment." (2019).
16. Kowta, A. S. L., Bhowmick, K., Kaur, J. R. & Jeyanthi, N. "Analysis and overview of information gathering & tools for pentesting." *2021 International Conference on Computer Communication and Informatics (ICCCI) IEEE.* (2021): 1-13.
17. Pargaonkar, S. "Advancements in security testing: A comprehensive review of methodologies and emerging trends in software quality engineering." *International Journal of Science and Research (IJSR)* 12.9 (2023): 61-66.
18. Patel, K. "A survey on vulnerability assessment & penetration testing for secure communication." *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI) IEEE.* (2019): 320-325.
19. Ravindran, U. & Potukuchi, R. V. "A review on web application vulnerability assessment and penetration testing." *Review of Computer Engineering Studies* 9.1 (2022).
20. Sarker, K. U., Yunus, F. & Deraman, A. "Penetration taxonomy: A systematic review on the penetration process, framework, standards, tools, and scoring methods." *Sustainability* 15.13 (2023): 10471.
21. Shashwat, K., Hahn, F., Ou, X., Goldgof, D., Hall, L., Ligatti, J. & Tabari, A. Z. "A preliminary study on using large language models in software pentesting." *arXiv preprint* arXiv:2401.17459 (2024).
22. Siriwardena, P. "Advanced API Security." *Apress: New York, NY, USA*, (2014).
23. Vamsi, P. R. & Jain, A. "Practical security testing of electronic commerce web applications." *International Journal of Advanced Networking and Applications* 13.1 (2021): 4861-4873.
24. Visoottiviseth, V., Akarasiriwong, P., Chaiyasart, S. & Chotivatunyu, S. "PENTOS: Penetration testing tool for Internet of Thing devices." *TENCON 2017-2017 IEEE Region 10 Conference IEEE.* (2017): 2279-2284.
25. Yadav, G., Allakany, A., Kumar, V., Paul, K. & Okamura, K. "Penetration testing

framework for IoT." *2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI) IEEE.* (2019): 477-482.

Source of support: Nil; **Conflict of interest:** Nil.

Cite this article as:

Shah, R., Mishra, G. and Suthari, Y. "Pentesting and Secure Code Reviews: Strengthening API Security in Modern Software Products." *Sarcouncil Journal of Engineering and Computer Sciences* 4.2 (2025): pp 1-9.