THE ASIAN BULLETIN OF BIG DATA MANAGMENT
Vol. 5. Issue 1 (2025)
https://doi.org/10.62019/abbdm.v5i1.320

ASIAN BULLETIN OF BIG DATA MANAGEMENT
http://abbdm.com/
ISSN (Print): 2959-0795
ISSN (online): 2959-0809

Insecure Software Development and Threat Mapping via Security Frameworks

Shaiqa Nadeem, *Ahthasham Sajid

Chronicle

Article history
Received: 01 Feb, 2025
Received in the revised format: 30th Feb, 2025
Accepted: 15th March, 2025
Available online: 30 March, 2025

Shaiqa Nadeem & Ahthasham Sajid are currently affiliated with Department of Information Security and Data Science, Riphah Institute of Systems Engineering, Riphah International University, Islamabad, Pakistan.
Email: shaiqanadem@gmail.com
Email: ahthasham.sajid@riphah.edu.pk

Corresponding Author*

Abstract

Integrating security in the software development life cycle has been a significant concern for researchers, security professionals and software developers. Security frameworks help to improve security in SDLC and mitigate threats by promoting the use of best practices. While in the presence of such best practices, security is somehow considered to be an afterthought and often leaves us with insecure software. Insecure web development makes web applications vulnerable to security threats like injection attacks, data breaches, privilege escalation, CSRF and other threats. This research based on mixed methods approach aims to provide valuable insights for security professionals and web application developers regarding the use of security frameworks to map threats for secure web development. Security frameworks including NIST SSDF, OWASP top 10, OWASP SAMM, SAFECode 3rd edition and BSIMM13 are involved for this purpose. The goal is to address the gap by leveraging these security frameworks to systematically map threats in web development environment. The research will aim to provide a comprehensive methodology for identifying potential security risks, analysing their impact and recommending security measures tailored to specifically web development environment. To achieve this, a comparative study of security frameworks, testing of web applications has been conducted to achieve the results.

Keywords: Insecure Web Development, Code Security, Secure Web Development, Threat mapping, Software Development lifecycle SDLC, Security frameworks.

INTRODUCTION

Basie Von Solms discussed the difference between cyber security and information security. In his research he defines cyber security as protecting information in cyber space only. Whereas, he defines information security to be the protection of information everywhere. (von Solms & von Solms, 2018). Secure software is resistant to security threats. Implementation of secure software development practices must be used to make software secure and resilient to attacks. Insecure software is developed due to insufficient knowledge of developers about secure software development practices. (Kanniah & Mahrin, 2024). Brown and Paller discussed in their research the reason, why developers write vulnerable code? In their study, the reason is knowledge gap of developers to secure coding practices as well as unawareness to what constitutes to secure coding. (Brown & Paller, 2008). Most common vulnerabilities in web are SQL injection, XSS, HTTP response splitting and path traversal. (Backman, 2018). Developers need to know security threats to make their code resistant to those flaws. (Brown & Paller, 2008). Stats shows that the first ever website was developed in 1991, which reached to 1.88 billion websites by 2018. (Internet Live Stats, 2024). Most significant data breach happened world-wide was experienced by "Yahoo" in 2017. This data breach compromised account information of 3 billion users. Another security attack was reported by Alibaba in July 2022 that exposed 1.1 billion users information. Similarly, LinkedIn reported data breach in June 2021 that compromised 700 million users' information. (Statista, 2021). Huge development around web security is needed with this growth rate of something that is in use to such an extent. (Sundqvist, 2018). The amount of data breached in these incidents reveals the growing need for web security.

The importance of developer's knowledge about web security and how to prevent common vulnerabilities cannot be neglected. (Sundqvist, 2018). Mitre analysed security attack report and revealed that most of the attacks exploited common web vulnerabilities. Web developers who lack web security knowledge are the most affected ones by data breaches. (Sundqvist, 2018). In industry, training developers about software security consumes more time and money. It is well known that during software development insecure coding practices are used by developers that make applications vulnerable to security threats. Educating software developers about how to write code that is free of security flaws and resilient to security threats is needed. (Gasiba et al., 2020). Developers having insufficient awareness about code security, mostly rely on code snippets available on stack overflow to solve their development problems. These insecure code snippets oftentimes provide functionality but threaten software security. (Fischer et al., 2017).

Developers must ensure that the software they are creating does not fulfil just functionality but is also secure. Emerging new threats reveals the fact that weaknesses exist somewhere in the software itself. Such weaknesses can easily be exploited by hackers. The Department of Homeland Security reports that 90% of weaknesses in software are exploited due to vulnerable code. Many security frameworks provide best practices to integrate security in SDLC by reducing vulnerabilities in software. (Ramirez et al., 2020). Best practices given by security frameworks are ignored by developers due to extensive workload of development as well as pressure to meet strict deadlines. (Backman, 2018). Many standards and frameworks might not address all security threats for secure software development when used alone. (Sundqvist, 2018). Common web vulnerabilities are SQL injection, cross-site scripting, remote file injection and broken access control. (Statista, 2021). Despite advancements in web development technologies and security protocols, many web applications remain susceptible to a range of threats like cross site scripting (XSS), sql injection, authentication bypass and distributed denial of service (DDos) attacks.

Developers often face difficulties in identifying, prioritizing and addressing vulnerabilities in early stages of development. Current security tools and frameworks, while helpful, do not always offer a holistic and proactive approach to threat identification and prevention. Additionally, many frameworks do not provide clear guidance on mapping security threats to corresponding mitigations within the context of a particular development environment or web development environment. The figure 1 shown below provides illustration of the concept behind the objective of the proposed study. This research focuses to acknowledge web security threats that makes a web application insecure, suggesting their relevant mitigation best practices and to understand the effectiveness of security frameworks in addressing those threats for web development. The purpose of this study is to suggest web developers the use of security frameworks for specific threats and to enable developers to develop secure web applications even if they lack knowledge or experience. The study focuses to introduce security in development phase of SDLC for web development. The underlying objectives for this study are to explore the benefits of using static testing and dynamic testing in threat mapping to address vulnerabilities that might be missed by either method individually. To identify specific web applications vulnerabilities addressed by these security frameworks.

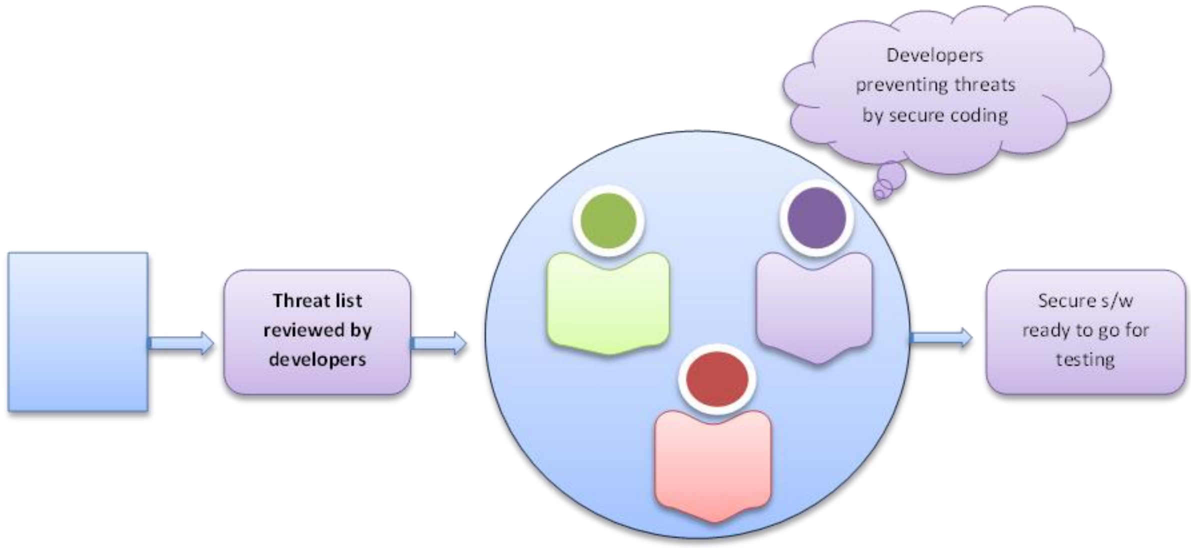
Developers preventing threats by secure coding
Threat list reviewed by developers
Secure s/w ready to go for testing

Figure 1.
Relationhip between Proposed Study and Development Team

To compare the strengths and weaknesses of security frameworks in addressing web application vulnerabilities. To examine the key challenges faced by web developers in integrating the security frameworks in SDLC.

METHODOLOGY

A comparative study of security frameworks against the threats identified from the static and dynamic testing of some open-source web applications has been adopted in the methodology. The methodology consists of several key steps shown below.

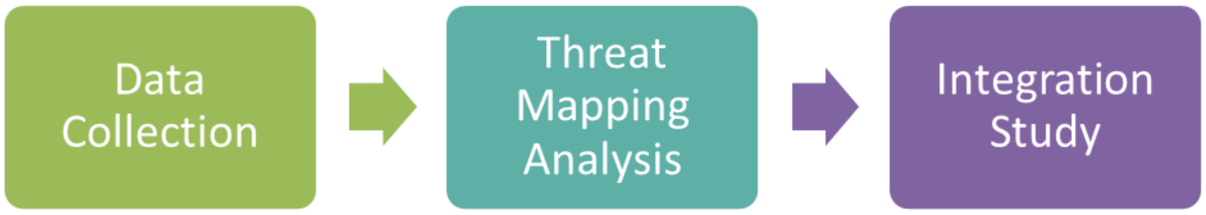

Figure 2.
Proposed Methodology Diagram

Data Collection

The first step involves collection of primary data sources for the study.

- **Selection of Web Applications** For static and dynamic testing of applications, 52 open-source web applications have been selected available on Github. The selected web applications include e-commerce, communication and content management applications developed using Javascript, Node.js, Python, Django, Flask, PHP and C# programming languages.
- **Static Testing of Web Applications**

Static testing is used to identify vulnerabilities and code errors by reviewing source code without executing the application. (Aydos & Aldan, 2022). In this study SNYK

Code has been selected to perform static testing of selected web applications. The most popular and used SAST tool by developers is SNYK Code (So Now You Know). SNYK Code is an effective and reliable tool in identifying code level vulnerabilities and is suggested to perform SAST of web applications. (Ali & Ammar, 2024).
In figure 3 shown below, threats detected in static testing by SNYK Code of all web applications are shown in the graph.

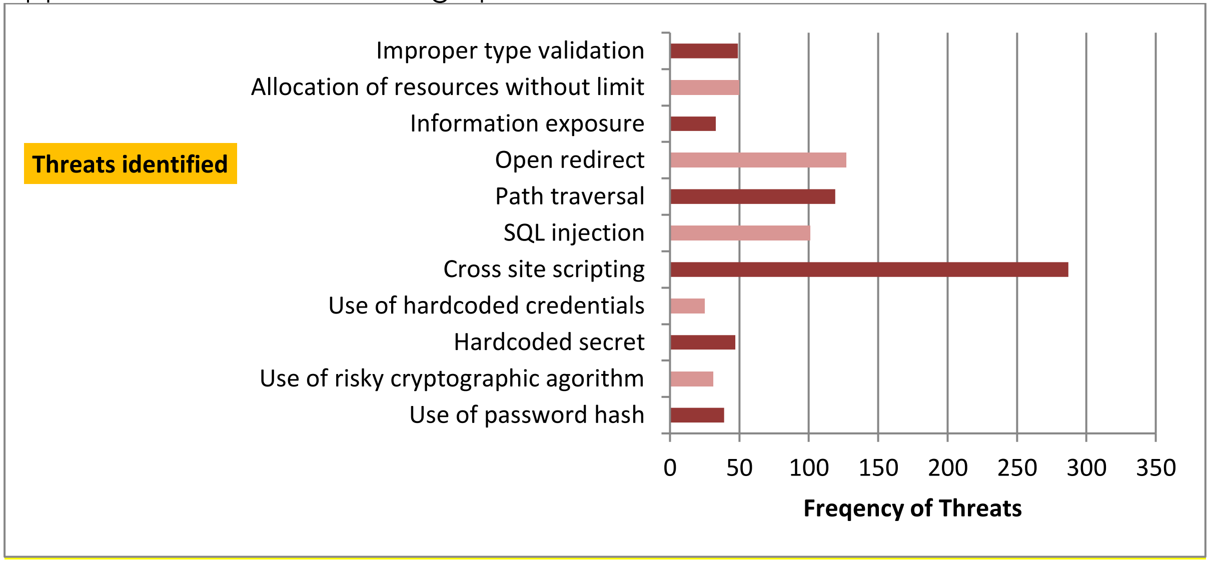

Figure 3.
Highly Frequent Threats Detected via Static Testing

In figure 4 shown below, low frequent threats identified by SNYK Code in static testing of all web applications are mentioned.

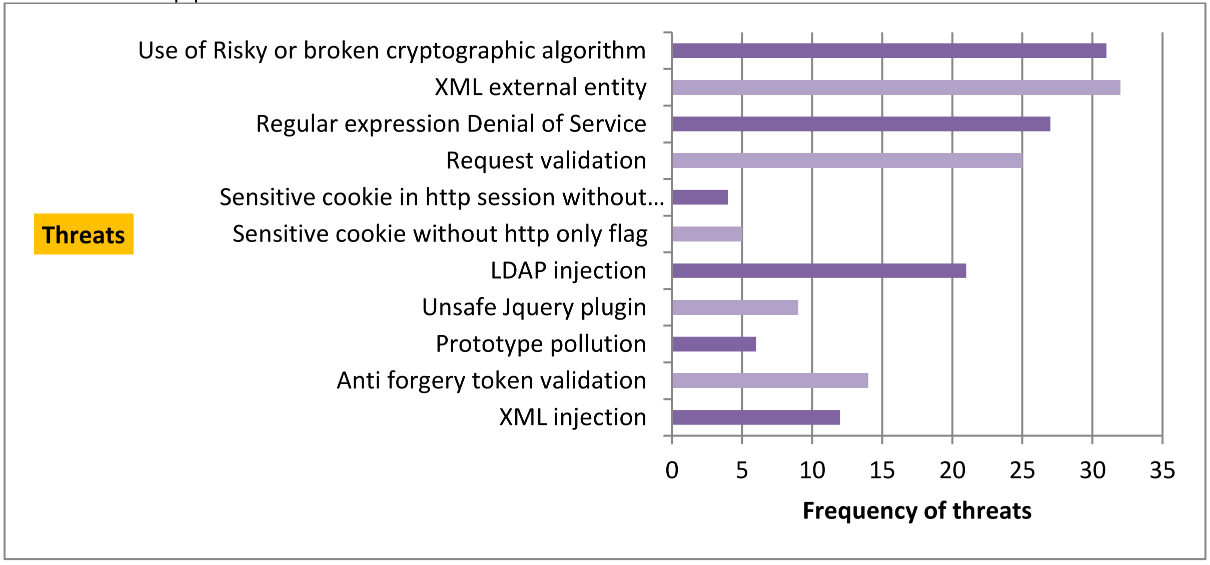

Figure 4.
Low Frequent Threats Detected via Static Testing

Dynamic Testing of Web Applications:

Dynamic testing involves the execution of application to detect runtime errors and security or performance issues. (Aydos & Aldan, 2022). In this study dynamic testing of selected web applications has been carried out by using Owasp Zap. OWASP ZAP version 2.12.0 is a reliable version and is suggested to be used for dynamic testing of web applications. (Potti et al., 2025). In the figure 5 shown below, different high and low priority threats found by Zap in dynamic testing of all the selected web applications have been shown.

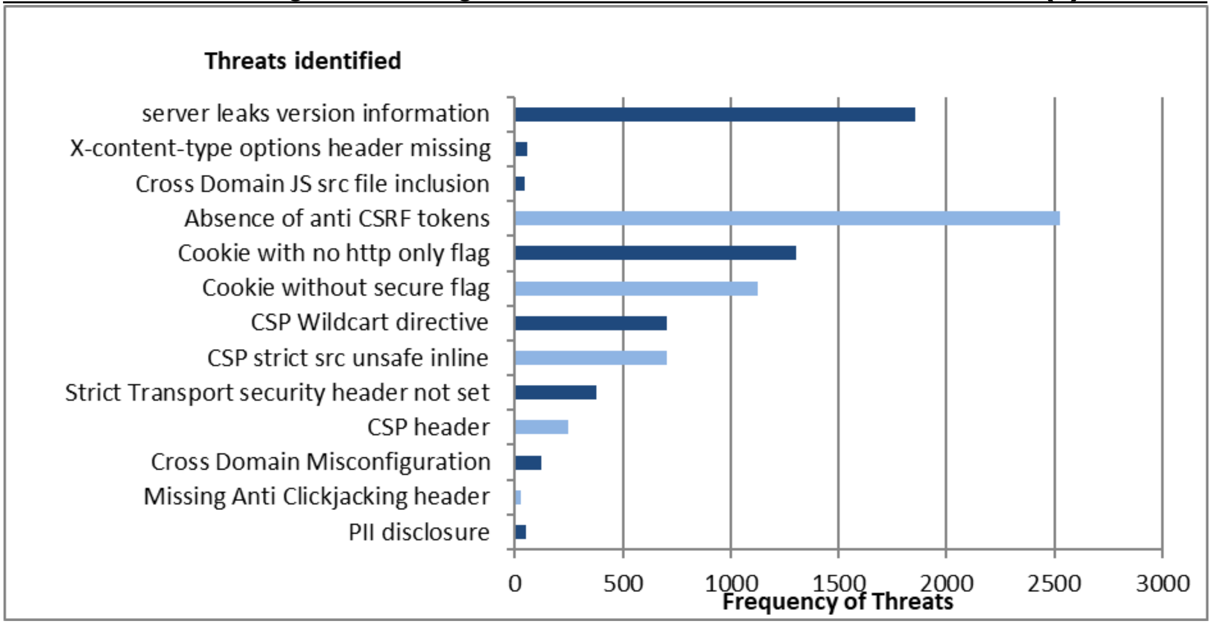

Figure 5.
Threats Identified Using Dynamic Testing of all web projects

Threat mapping Analysis

The second step involves comparative study of security frameworks to map threats against mitigation guidelines mentioned in the frameworks. The purpose of doing so is to evaluate the number of threats for which frameworks provide exactly accurate preventions, number of threats for which frameworks provide neutral or fuzzy mitigations and number of threats which are left uncovered by these frameworks. In this step strengths and weaknesses of each framework within the context to mitigate identified web security threats will become prominent.

Comparative Study of Security Frameworks with Reference to Identified Threats

Selected security frameworks for this study are Owasp Samm, Owasp Top Ten, Nist SSDF, Safecode and BSIMM. The factor that is common among all of these security frameworks is that these frameworks guide the development process of secure software. These frameworks have been studied thoroughly. The purpose of this comparative study is to analyse how well and to what extent these frameworks help in addressing and reducing web application vulnerabilities.

Integration Study

The third step involves the survey of web developers to evaluate developer adoption rate for each framework and to identify key challenges.

Survey of Web Developers

The purpose to conduct survey is to identify key challenges that are faced by web developers while integrating security frameworks in the development phase of web development.

RESULTS

Comparative Analysis of Security Frameworks

To observe the performance of security frameworks for the prevention of identified threats, a comparison of strengths and weaknesses of security frameworks is needed. Whereas strengths and weaknesses of frameworks with reference to identified threats depend on three factors mentioned below.

Covered Threats The figure 6 shown below represents percentage of threats for which security frameworks provide clear or accurate mitigation guidelines.

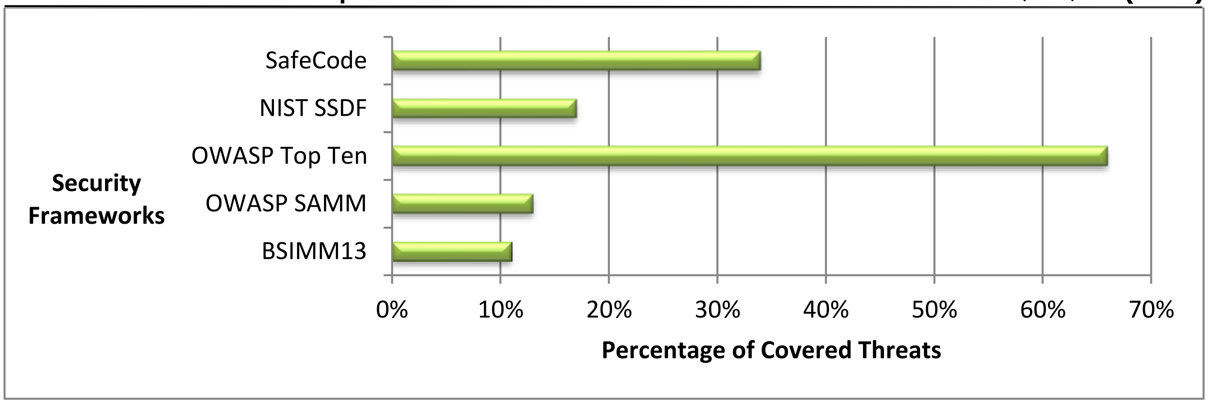

Figure 6.
Percentage of Covered Threats

Neutralized Threats

The **figure 7** shown below represents percentage of identified threats for which security frameworks do not provide clear or comprehensive mitigation strategies. BSIMM13 do not neutralizes any threat, that is why there is no bar showing its progress to neutralize threats.

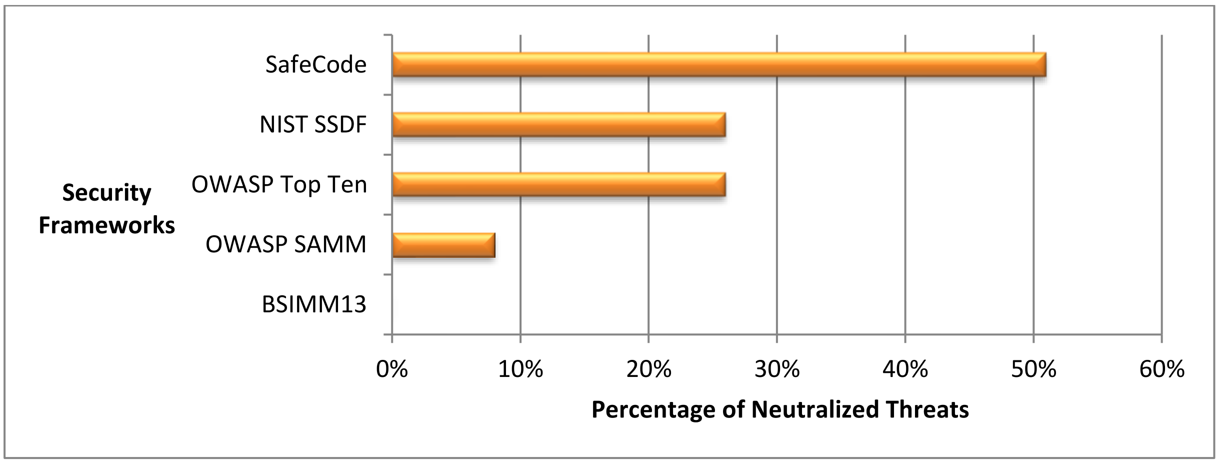

Figure 7.
Percentage of Neutralized Threats

Uncovered Threats

The figure 8 shown below represents the percentage of threats for which security frameworks do not provide mitigation strategy.

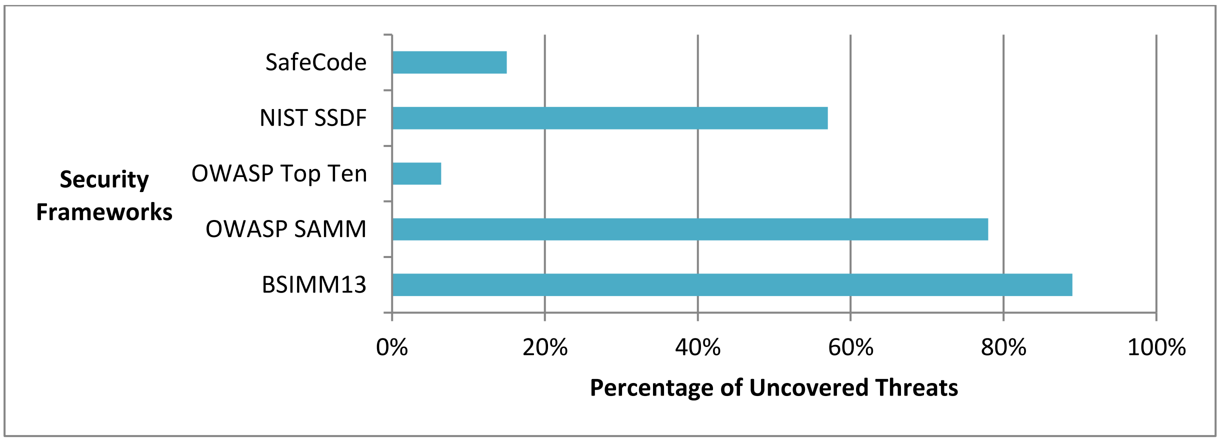

Figure 8.
Percentage of Uncovered Threats

Survey

The survey of 52 web developers shows developer adoption rate for all security frameworks included in this study in the figure 9 below. Owasp top ten is adopted by most of the web developers for web development with a developer adoption rate of 65%, for BSIMM13 is 11.5%, for Owasp Samm is 19.2%, for Nist SSDf is 51.9 and for SafeCode 3rd edition is 53.8%.

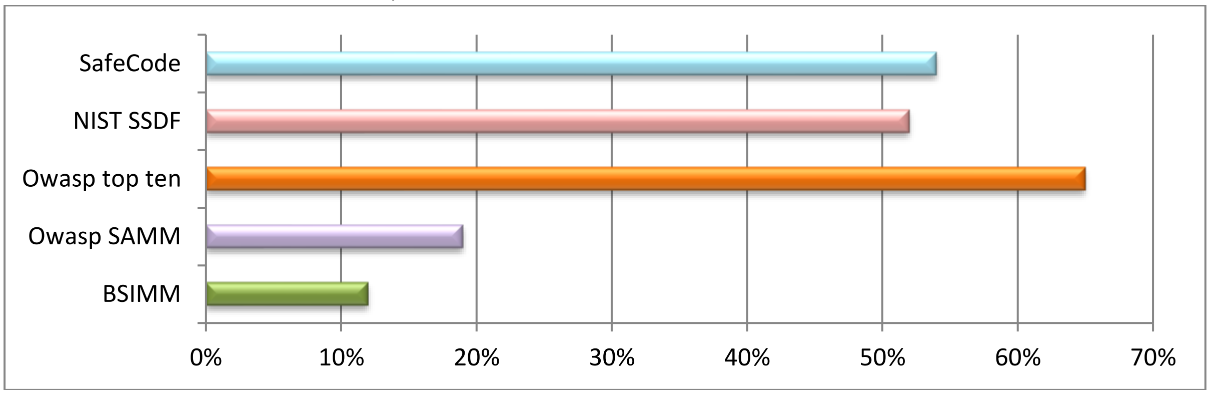

Figure 9.
Developer Adoption Rate of each Framework

When the web developers were asked that do you know which specific threats can be mitigated by these frameworks included in this study. The figure 10 shown below represents that not a single web developer had any knowledge about this fact.

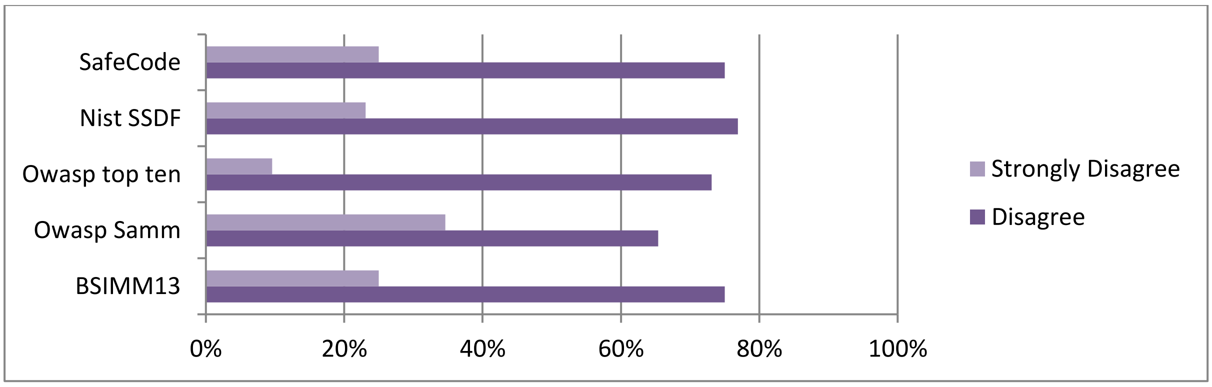

Figure 10.
Knowledge of Developers about Threats mitigated by the Frameworks

The **figure 11** shown below presents the challenges faced by developers while implementing the security frameworks.

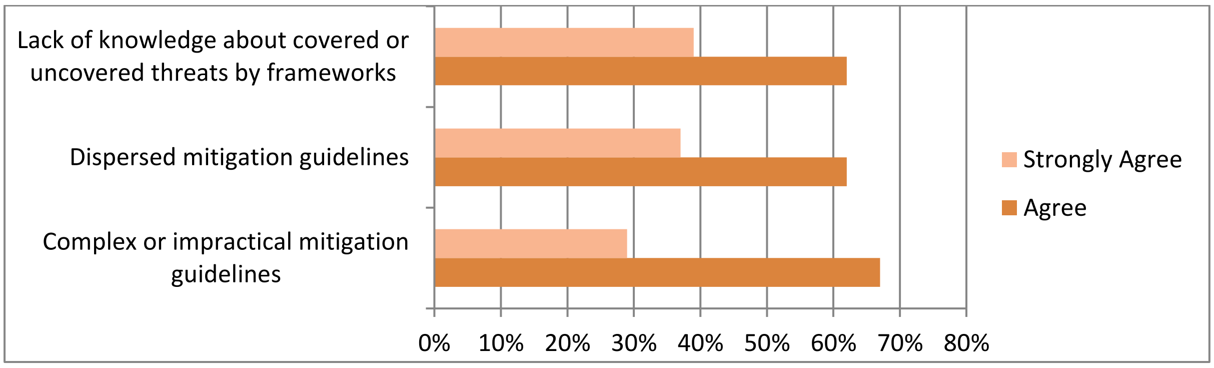

Figure 11.
Challenges faced by Web Developers while implementing Frameworks

DISCUSSION OF SURVEY

The findings indicate that while most web developers are familiar to security frameworks such as BSIMM, Owasp Samm, Owasp top ten, Nist SSDF and SafeCode for web development. However, they do not consistently integrate the security practices prescribed by these frameworks into their daily development workflow. Developers acknowledge the presence of several challenges that hinder the effective implementation of these security practices. Among the primary obstacles are fragmented mitigation guidelines, the complexity to understand these guidelines and lack of awareness regarding the extent to which each framework addresses, neutralizes or leaves security threats unresolved. Beyond these specific challenges, developers also contend with broader constraints such as time management, strict project deadlines and organizational policies. These general challenges, however, are common across the development community. Additionally, a significant number of developers recognize the importance of threat mapping and consider it a crucial aspect of software development. To enhance the adoption of security frameworks, developers suggest consolidating mitigation controls for prevalent web security threats into a unified reference, along with increasing awareness of which threats each framework effectively prevents, mitigate or leaves unresolved. This, in turn, would facilitate informed decision making in selecting appropriate framework for web development and ultimately improve the overall adoption rate of security frameworks within the industry.

Table 1.
Efficacy of each Framework

Sr #	Security Framework	%age of Threat Model Coverage	Covered Threats	Neutralized Threats
1.	BSIMM13	10%	10%	0%
2.	OWASP SAMM	12%	12%	8%
3.	OWASP Top Ten	63%	63%	24.5%
4.	NIST SSDF	16%	16%	24.5%
5.	SAFECode	33%	33%	49%

The above represented data show that BSIMM13 provides best practices to prevent 10% of threats out of all identified threats in this study. While OWASP SAMM provides best practices to prevent 12% threats, OWASP Top Ten provides best practices to prevent 63% threats, NIST SSDF provides best practices to prevent 16% threats and finally SAFECode provides best practices to prevent 33% threats out of all identified threats during static and dynamic testing of all web applications in this study. Whereas threat model consists of all threats identified during static and dynamic testing of web applications and threat model coverage represents percentage of threats identified by the framework from the threat model.

RESULTS DISCUSSION

This study identified 49 threats in static and dynamic testing of web applications. The results of this study shows that OWASP top Ten is the most effective whereas BSIMM is the least effective framework in providing preventive guidelines for web security threats that can be prevented in the development phase of web development. This shows that OWASP Top Ten highlights most of the web application security threats and provides mitigation guidelines to prevent most of the web security vulnerabilities. Therefore, Owasp Top Ten is suggested to be followed during the software development life cycle to enhance the security of the web application or website that is to be developed. The study increases the knowledge of web developers about the web security threats and efficiency of security frameworks in addressing those threats. The results of the study not just include identification of web security threats but also provide controls for the identified threats that have adherence to security frameworks. This methodology makes the web application developers aware of the threats that can be prevented by them in implementation phase.

CONCLUSION

This study concludes that merely adhering to a security framework in web development does not inherently ensure the comprehensive mitigation of all security threats, nor does it guarantee the absolute security of web applications. Rather, the crux of secure development lies in developers' understanding of the efficacy of these frameworks in mitigating security vulnerabilities and fortifying software against cyber threats. It is imperative for web developers to discern which web security threats can be effectively neutralized and which remain unaddressed despite the application of security frameworks. Such insights can significantly alleviate the complexities associated with secure software development. The results of this study stem from an analysis of the identified challenges of developers that include fragmented mitigation, complex security guidelines and lack of transparency regarding the threat landscape. For web developers, it is crucial to recognize the threats that are mitigated, neutralized or left unresolved by these security frameworks. Ultimately, this study provides security controls for identified web security threats thus enable and equip web developers with actionable insights to enhance the security posture of their applications.

LIMITATIONS

This study intends to provide valuable insights regarding identification and mitigation of web security threats. Most of the identified threats can be prevented by web developers by using secure coding guidelines. The study also aims to evaluate the effectiveness of security frameworks for the prevention of identified threats. The limitations of this study include identification of security threats that can be prevented by web developers in the field of web application development. Additionally, the evolving nature of security threats may limit the comprehensiveness of the findings of this study.

FUTURE WORK

In future work, the authors of this study propose to include more threats specific to web environment and refine control selection by considering specific preventive measures. Moreover, the authors also aim to employ different penetration testing techniques to web applications developed by following security frameworks in order to identify how the software behave towards multiple pen testing techniques. Instead of mapping threats to best practices given by frameworks, different attack techniques can help in analysing the behaviour of the software developed by following security frameworks against multiple attacks.

DECLARATIONS

Acknowledgement: We appreciate the generous support from all the supervisors and their different affiliations.
Funding: No funding body in the public, private, or nonprofit sectors provided a particular grant for this research.
Availability of data and material: In the approach, the data sources for the variables are stated.
Authors' contributions: Each author participated equally to the creation of this work.
Conflicts of Interests: The authors declare no conflict of interest.
Consent to Participate: Yes

Consent for publication and Ethical approval: Because this study does not include human or animal data, ethical approval is not required for publication. All authors have given their consent.

REFERENCES

Ali, E. A., & Ammar, F. M. (2024). SAST tools and manual testing to improve the methodology of vulnerability detection in web applications. *The International Journal of Engineering and Information Technology, 12*(1).

Aydos, M., & Aldan, K. (2022). Security testing of web applications: A systematic mapping of the literature. *Journal of King Saud University - Computer and Information Sciences, 34*(10), 7633-7650. https://www.sciencedirect.com/science/article/pii/S131915782100269X

Backman, L. (2018). *En jämförande studie mellan utvecklares medvetenhet kring mjukvarusäkerhet och existerande sårbarheter i deras mjukvara* (Why is security still an issue?-A study comparing developers' software security awareness to existing vulnerabilities in software applications) [Master's thesis, Linköping University]. Linköping University Electronic Press.

Brown, M., & Paller, A. (2008). Secure software development: Why the development world awoke to the challenge. *Information Security Technical Report, 13*(1), 40-43. https://doi.org/10.1016/j.istr.2008.03.001

Fischer, F., Böttinger, K., Xiao, H., Stransky, C., Acar, Y., Backes, M., & Fahl, S. (2017). Stack Overflow considered harmful? The impact of copy & paste on Android application security. *Proceedings of the IEEE Symposium on Security and Privacy*, 121–136. https://doi.org/10.1109/SP.2017.31

Gasiba, T., Lechner, U., Cuellar, J., & Zouitni, A. (2020). Ranking secure coding guidelines for software developer awareness training in the industry. *OASIcs.ICPEC, 2020(11)*. https://doi.org/10.4230/OASIcs.ICPEC.2020.11

Internet Live Stats. (2024). *Total number of websites*. Internet Live Stats. Retrieved June 2, 2024, from https://www.internetlivestats.com/total-number-of-websites/

Kanniah, L., & Mahrin, M. N. (2024). A review on factors influencing implementation of secure software development practices. *International Journal of Computer Systems Engineering*. Retrieved from https://d1wqtxts1xzle7.cloudfront.net/...

Larochelle, D., & Evans, D. (2002). Improving security using extensible lightweight static analysis. *IEEE Software, 19(1)*, 42–51. https://doi.org/10.1109/52.976940

Potti, U.-S., Huang, H.-S., Chen, H.-T., & Sun, H.-M. (2025, January 10). Security testing framework for web applications: Benchmarking ZAP V2.12.0 and V2.13.0 by OWASP as an example. *arXiv*. https://doi.org/10.48550/arXiv.2501.05907

Ramirez, A., Aiello, A. J., & Lincke, S. J. (2020). A survey and comparison of secure software development standards. *2020 13th CMI Conference on Cybersecurity and Privacy (CMI)*, 1–6. https://doi.org/10.1109/CMI51275.2020.9322704

Statista. (2021). *Biggest online data breaches worldwide as of March 2021, by number of records exposed*. Statista. Retrieved June 2, 2024, from https://www.statista.com/statistics/290525/cyber-crime-biggest-online-data-breaches-worldwide/

Sundqvist, J. (2018). *Reasons for lacking web security: An investigation into the knowledge of web developers* [Bachelor's thesis, Blekinge Institute of Technology]. URN: nbn:se:bth-17008

von Solms, B., & von Solms, R. (2018). Cybersecurity and information security – What goes where? *Information and Computer Security, 26*(1), 2-9. https://doi.org/10.1108/ICS-04-2017-0025