IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 2, July 2023

Application Security and Secure Coding Practices

Nilam Nagesh Lokhande
Student, Department of Masters of Computer Applications
Late Bhausaheb Hiray S. S. Trust's Hiray Institute of Computer Application, Mumbai, India
nilamlokhande2303@gmail.com

Abstract: *The concept of security in web applications is not new. However, it is often ignored in the development stages of the applications. Moreover, developers are more inclined to implement features and often do not practice secure coding. Therefore, countless web applications are launched with security vulnerabilities like cross-site scripting, injection attacks and resource alterations. As software applications are used more often across a range of industries, maintaining their security has grown to be a top priority. Web applications comprise a large proportion of the contemporary Internet with many of them dealing with sensitive information and handling critical operations whose compromise could result in large monetary and privacy costs. Naturally, the security of web applications has become an increasingly important issue as web technologies are utilized more and more. Without practicing secure coding and having an integrity verification system in place, it is difficult to defend security attacks. To that end, the incorporation of security controls throughout the software development lifecycle (SDLC) has emerged as the most prominent solution for detecting security defects early and fixing them with minimal cost and overhead. This research paper gives an in-depth analysis of secure coding techniques and application security. The study finishes by summarizing the main conclusions and highlighting the value of application security and secure encryption procedures to lower risk and safeguard sensitive data*

Keywords: Vulnerabilities, Security, guidelines, confidentiality, mitigate, SDLC

I. INTRODUCTION

Application security refers to the discipline of protecting software applications from threats, vulnerabilities, and attacks. It involves implementing measures and practices to identify, mitigate, and prevent security risks throughout the entire lifecycle of an application. It is reported that most vulnerabilities originate in the source code of the application. Specifically, the survey by Positive Technologies reports a whopping 82% of vulnerabilities being located in the application code. The main objective of application security is to ensure the confidentiality, integrity, and availability of both the application itself and the data it processes.

Secure coding practices are a set of techniques and guidelines that developers follow to write code that is resistant to security vulnerabilities and exploits. By incorporating secure coding practices into the software development process, developers can build applications that are more robust and less susceptible to attacks. These practices address various aspects of coding, including input validation, authentication, access control, secure communication, error handling, and more. The matter of addressing security in application development calls for integration of security controls throughout the software development lifecycle.By incorporating these practices into the software development process, organizations can build more secure and resilient applications, reducing the risk of data breaches and ensuring the protection of sensitive information.

II. LITERATURE REVIEW

Application security and secure coding practices are critical aspects of software development that aim to protect applications from vulnerabilities and potential threats. This literature review provides an overview of the current research and best practices in the field of application security and secure coding.A systematic literature review (SLR) was selected as the research method for this study. "An SLR is a type of secondary study in which primary studies are examined impartially and iteratively to define,

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 2, July 2023

interpret, and discuss evidence relevant to the research questions”

"Secure Coding: Principles and Practices" by Mark G. Graff and Kenneth R. van Wyk:
This book offers comprehensive coverage of secure coding principles and best practices. It covers various programming languages and provides practical examples, code snippets, and case studies. The authors emphasize secure coding techniques to prevent common vulnerabilities like buffer overflows, injection attacks, and cross-site scripting.

"The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws" by Dafydd Stuttard and Marcus Pin to:
Focused on web application security, this book explores the techniques used by attackers and provides insights into securing web applications. It covers topics such as input validation, authentication, session management, and secure communication. The authors also delve into common vulnerabilities and attack vectors, making it a valuable resource for understanding application security risks.

"OWASP Testing Guide" by The Open Web Application Security Project (OWASP):
OWASP is a well-known organization dedicated to improving application security. Their Testing Guide is a comprehensive resource that outlines various security testing techniques, methodologies, and tools. It covers all stages of the software development lifecycle, including threat modeling, code review, and penetration testing.

"Secure Programming Cookbook for C and C++" by John Viega and Matt Messier:
This book focuses on secure coding practices specific to C and C++ programming languages. It provides practical recipes for addressing common vulnerabilities and demonstrates how to write secure code using the language-specific features and libraries. The book covers memory management, input validation, cryptographic functions, and secure communication.

"Secure Development for Mobile Apps: How to Design and Code Secure Mobile Applications with PHP and JavaScript" by J. D. Glaser.:
With the increasing prevalence of mobile applications, this book provides insights into secure development practices specifically for mobile platforms. It covers topics such as secure data storage, user authentication, handling push notifications securely, and secure network communication. The book includes real-world examples and code snippets.
"Building Secure Software: How to Avoid Security Problems the Right Way" by John Viega and Gary McGraw:
In this book, the authors emphasize the importance of integrating security practices into the software development process from the beginning. It covers security requirements, threat modelling, secure design principles, and secure coding techniques. The book provides actionable recommendations and case studies to help developers build more secure software.

III. PROBLEM DEFINITION

The problem addressed in this research paper is the need for effective application security and the implementation of secure coding practices in software development. The objective is to identify the specific challenges and issues faced in ensuring the security of applications and to propose solutions for integrating secure coding practices into the development process
Key problem areas include:

- Lack of Security Awareness and Education: Many developers may not have sufficient knowledge and awareness of secure coding practices and the importance of application security.
- Vulnerability Management: Applications are susceptible to various vulnerabilities, such as input validation flaws, insecure session management, and inadequate authentication mechanisms. The problem is to effectively identify, assess, and mitigate these vulnerabilities throughout the application.
- Integration of Security into Software Development Lifecycle: Integrating security practices into the software development lifecycle can be challenging. The problem is to find effective ways to seamlessly integrate security activities, such as secure coding reviews, vulnerability assessments, and security testing, into the software development process.
- Emerging Threat Landscape: The threat landscape is constantly evolving, with new attack vectors and techniques emerging regularly.
- Compliance with Security Standards and Regulations: Organizations need to comply with industry-specific security standards and

regulations to protect user data and ensure the privacy and security of sensitive information.

- Secure Integration of Third-Party Components: Many applications rely on third-party libraries, frameworks, and components. The problem is to address the security risks associated with third-party components and establish best practices for their secure integration and ongoing management.

By addressing these problems, organizations can enhance the security of their applications, protect sensitive data, and mitigate the risk of security breaches. The proposed research aims to identify effective solutions, best practices, and frameworks for application security and secure coding practices, ultimately contributing to the development of more secure software systems.

IV. OBJECTIVE / SCOPE

The objective of this research paper is to investigate and analyze the various aspects of application security and secure coding practices. The paper aims to explore the importance, challenges, best practices, and emerging trends in the field of application security and secure coding. It seeks to provide insights, recommendations, and guidelines to enhance the security of software applications and promote the adoption of secure coding practices.

4.1 Scope

The research paper will focus on the following key areas related to application security and secure coding practices:

- Overview of Application Security
- Secure Coding Practices
- Vulnerability Analysis and Mitigation
- Integration of Application Security in Software Development Lifecycle (SDLC)
- Emerging Trends and Technologies
- Compliance and Regulatory Requirements

The scope of the research on application security and secure coding practices will encompass various dimensions and areas of focus including:

- Software Development Lifecycle (SDLC):It examines how security measures can be integrated at each stage to ensure secure software development.
- Programming Languages and Frameworks: The research focuses on specific programming languages or frameworks commonly used in application development.
- Secure Coding Guidelines and Standards: The research evaluates and propose enhancements to existing secure coding guidelines and standards such as OWASP Top Ten, CERT Secure Coding Standards, or SANS Secure Coding.
- Secure Development Tools and Technologies: The research explores the effectiveness of various tools and technologies used for secure application development.
- Emerging Technologies and Security Challenges: The scope can extend to emerging technologies such as cloud computing, Internet of Things (IoT), blockchain, or artificial intelligence (AI), and their associated security challenges.
- Human Factors and Education: The research investigates the role of human factors in application security, including developer awareness, training, and secure coding education.

V. RESEARCH METHODOLOGY

The main purpose of this literature review is to study the current challenges and gaps in application security. It involves systematic and structured approach to gather, analyze, and interpret data. Phases involved in carrying out this research are as follows:

- Planning
- Conducting
- Reporting

A. Research Questions

1.What are the security risks that should be avoided while designing secure software applications?
2.What are the best practices to follow when designing secure software applications?
3.What What are the challenges, limitations, and gaps related to application security?

B. Code Analysis

Analyzed code samples or projects usingstatic analysis tools, manual reviews, or code scanning techniques to identify security vulnerabilities, adherence to secure coding practices, and common pitfalls.

Following steps are part of the research methodology:

- Data Collection
- Data Analysis
- Framework and Model Development
- Case Studies and Experiments
- Recommendations and Guidelines

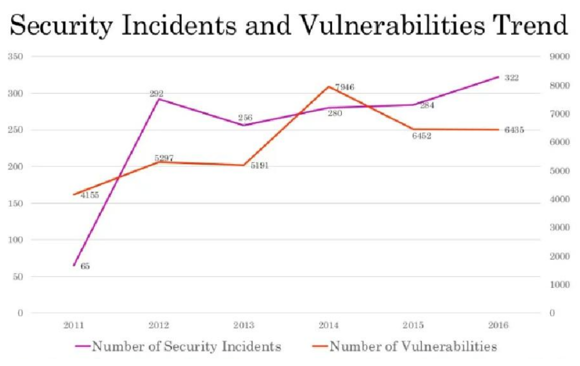
- Validation and Peer Review
- Conclusion and Limitations

By following this methodology, research paper was produced to provide valuable insights, practical recommendations, and guidelines to enhance application security and promote the adoption of secure coding practices in software development.

VI. ANALYSIS & FINDINGS

Securing critical software resources is more important than ever as the focus of attackers has steadily moved toward the application layer. Study found that attacks against web applications constitute more than 60% of the total attack attempts observed on the Internet.

This report provides coding practices that can be translated into coding requirements without the need for the developer to have an in depth understanding of security vulnerabilities and exploits. However, other members of the development team should have the responsibility, adequate training, tools and resources to validate that the design and implementation of the entire system is secure.



Below are set of general software security coding practices, that can be integrated into the software development lifecycle. Implementation of these practices will mitigate most common software vulnerabilities.

Input Validation:

- Conduct all input validation on a trusted system (server side not client side)
- Identify all data sources and classify them into trusted and untrusted
- Validate all data from untrusted sources (databases, file streams, etc)
- Use a centralized input validation routine for the whole application
- Specify character sets, such as UTF-8, for all input sources (canonicalization)

Output Encoding:

- Conduct all output encoding on a trusted system (server side not client side)
- Utilize a standard, tested routine for each type of outbound encoding
- Specify character sets, such as UTF-8, for all outputs
- Contextually output encode all data returned to the client from untrusted sources
- Ensure the output encoding is safe for all target systems

Authentication and password management

- Require authentication for all pages and resources, except those specifically intended to be public
- All authentication controls must be enforced on a trusted system
- Establish and utilize standard, tested, authentication services whenever possible
- Use a centralized implementation for all authentication controls, including libraries that call external authentication services
- If your application manages a credential store, use cryptographically strong one-way salted hashes

Session management

- Use the server or framework's session management controls. The application should recognize only these session identifiers as valid
- Session identifier creation must always be done on a trusted system (server side not client side)
- Session management controls should use well vetted algorithms that ensure sufficiently random session identifiers

Access control

- Use only trusted system objects, e.g. server side session objects, for making access authorization decisions
- Enforce authorization controls on every request, including those made by server side scripts
- Segregate privileged logic from other application code
- Restrict access to files or other resources, including those outside the application's direct control, to only authorized users

Cryptographic practices

- All cryptographic functions used to protect secrets from the application user must be implemented on a trusted system

Error handling and logging

- Do not disclose sensitive information in error responses, including system details, session identifiers or account information
- Use error handlers that do not display debugging or stack trace information

Data protection

- Implement least privilege, restrict users to only the functionality, data and system information that is required to perform their tasks

Database security

- o Use strongly typed parameterized queries
- o Utilize input validation and output encoding and be sure to address meta characters. If these fail, do not run the database command
- o Use secure credentials for database access

General coding practices

- o Utilize task specific built-in APIs to conduct operating system tasks. Do not allow the application to issue commands directly to the Operating System, especially through the use of application initiated command shells
- o Utilize locking to prevent multiple simultaneous requests or use a synchronization mechanism to prevent race conditions
- o Do not pass user supplied data to any dynamic execution function
- o Review all secondary applications, third party code and libraries to determine business necessity and validate safe functionality

By implementing effective security measures and following secure coding practices, organizations can significantly reduce the risk of security breaches and safeguard sensitive data.

Key findings and analysis include:

- Common Vulnerabilities: Through research and analysis, it has been found that certain vulnerabilities, such as injection attacks (e.g., SQL injection), cross-site scripting (XSS), and authentication bypass, are prevalent in many applications. These vulnerabilities can lead to unauthorized access, data breaches, and other security incidents.
- Importance of Secure Coding: Secure coding practices play a crucial role in preventing security vulnerabilities. By following guidelines and best practices for secure coding, developers can minimize the risk of introducing vulnerabilities during the software development process.
- Integration of Security in SDLC: Integrating security activities throughout the software development lifecycle (SDLC) is crucial. This includes incorporating security requirements analysis, threat modelling, security testing, and code reviews at various stages of development. Organizations that adopt a secure SDLC approach tend to have more robust and secure applications.
- Secure Third-Party Component Usage: Many applications rely on third-party libraries and components. However, it has been observed that inadequate vetting and management of these components can introduce vulnerabilities. Organizations should prioritize the evaluation and continuous monitoring of third-party components to ensure their security.
- Emerging Threat Landscape: The threat landscape is constantly evolving, with new attack vectors and techniques emerging regularly. The research in this area highlights the importance of staying updated on the latest threats and vulnerabilities to effectively counteract them. Techniques such as threat intelligence and proactive vulnerability scanning can aid in identifying and addressing emerging threats.
- Training and Awareness: Promoting security education and awareness among developers and stakeholders is essential. Research emphasizes the significance of providing training, resources, and regular knowledge sharing sessions to ensure that individuals involved in software development understand the importance of application security and follow secure coding practices.

VII. LIMITATIONS & FUTURE SCOPE

7.1 Limitations:

- Human Error: Despite following secure coding practices, human error can still occur. Developers may inadvertently introduce vulnerabilities or overlook certain security considerations

- Lack of Standardization: There is a lack of universal standards and guidelines for secure coding practices. While organizations can follow established frameworks like OWASP Top 10 or CERT Secure Coding Standards.
- Rapid Technological Advancements: The rapid evolution of technologies, frameworks, and programming languages introduces new security challenges. Secure coding practices must adapt to these advancements, requiring continuous learning and updates to address emerging vulnerabilities.
- Time and Resource Constraints: Implementing robust application security and following secure coding practices can be time-consuming and resource-intensive.
- Legacy Systems and Codebases: Organizations often have legacy systems and codebases that were developed without adequate security considerations. Retrofitting security measures into these systems can be challenging and may require significant time and effort.

7.2 Future Scope

- Secure DevOps Integration: The integration of security practices into DevOps methodologies is gaining prominence. Future research can focus on effective ways to seamlessly integrate security controls and processes throughout the DevOps lifecycle, ensuring security is not an afterthought but an inherent part of the development process.
- Automation and Tooling: Advancements in automation and security tooling can enhance secure coding practices. Future research can explore the development of advanced static and dynamic analysis tools, code scanners, and automated security testing techniques to identify vulnerabilities and enforce secure coding practices more efficiently.
- Secure Coding for Emerging Technologies: As new technologies emerge, such as blockchain, IoT, and AI, there is a need to develop secure coding practices specific to these domains.
- Metrics and Evaluation: Developing metrics and evaluation frameworks for measuring the effectiveness of secure coding practices can provide valuable insights. Future research can explore methodologies for assessing the impact of secure coding practices on application security, quantifying the reduction in vulnerabilities, and evaluating the return on investment in security measures.
- Collaborative Efforts and Knowledge Sharing: Encouraging collaboration and knowledge sharing among developers, security professionals, and researchers is crucial. Future research can focus on fostering communities, platforms, and forums for sharing best practices, case studies, and lessons learned to collectively enhance application security and secure coding practices.

VIII. CONCLUSION

In conclusion, the research paper highlights the importance of incorporating robust security measures and adhering to secure coding practices in software development. The findings emphasize that application security is a critical concern, given the evolving threat landscape and potential vulnerabilities that can be exploited by malicious actors.

The analysis reveals that secure coding practices, such as input validation, output encoding, access control, secure communication, and error handling, significantly contribute to minimizing security risks. Integration of security activities throughout the software development lifecycle (SDLC) and proper management of third-party components are crucial for building secure applications.

The research paper identifies limitations, including human error, lack of standardization, resource constraints, and the challenges of securing legacy systems. Future research should focus on integrating security practices into DevOps methodologies, leveraging automation and tooling, developing secure coding practices for emerging technologies, establishing metrics and evaluation frameworks, and promoting collaborative efforts and knowledge sharing.

Ultimately, the research paper emphasizes the need for continuous training, awareness, and adaptation to address emerging threats. By embracing secure coding practices and implementing effective application security measures, organizations can enhance their resilience to security breaches, safeguard sensitive data, and build trustworthy software applications.

REFERENCES

[1]. Secure Programming Cookbook for C and C++ by John Viega and Matt Messier

[2]. The Art of Software Security Assessment: Identifying and Preventing Software

Vulnerabilities by Mark Dowd, John McDonald, and Justin Schuh
[3]. Secure Coding in Java: Best Practices for Secure Java Development by Robert C. Seacord
[4]. Threat Modeling: Designing for Security by Adam Shostack
[5]. Secure Development for Mobile Apps: How to Design and Code Secure Mobile Applications with PHP and JavaScript by J.D. Glaser
[6]. Secure Coding Guidelines for the Java Programming Language by Oracle
[7]. Common Weakness Enumeration (CWE) - MITRE Corporation
[8]. The Building Security In Maturity Model (BSIMM) by Cigital, Inc.
[9]. Security Development Lifecycle (SDL) Implementation Guide by Microsoft
[10]. ISO/IEC 27034: Application Security
[11]. NIST SP 800-64: Security Considerations in the System Development Life Cycle
[12]. CWE (Common Weakness Enumeration) and CERT Secure Coding Standards
[13]. www.sans.org/reading-room/topics/secure-coding
[14]. www.nist.gov/topics/software-assurance
[15]. www.computer.org/technical-committees/center-for-secure-design