Comprehensive Analysis on Secure Software Development Practices

Zaki Ahmad Rathore Computer Science University of Central Florida Orlando, Florida, United States za535945@ucf.edu

Shishir Singh Computer Science University of Central Florida Orlando, Florida, USA Shishir.singh@ucf.edu

Vigneshwar Sundararajan Cybersecurity and Privacy University of Central Florida Orlando, Florida, United States vi126747@ucf.edu

ABSTRACT

In today's interconnected world, where digital systems underpin almost every aspect of our lives, the security of software has never been more critical.

Data leaks, financial losses, and even threats to national security can result from software flaws. The techniques used by bad actors to exploit vulnerabilities evolve along with technology. Organizations must establish and continuously enhance safe software development processes to combat this changing threat landscape. In this proposal, a thorough study effort that will survey, examine, and improve secure software development processes across several businesses is described.

PROBLEM STATEMENT

When we talk about Software Security, we have to keep in mind that SDLC is a process that consists of many stages. Programming, in reality, is not even half the total part of the life cycle in question. So it is imperative to make sure that secure practices are followed along the way for most of the stages, if not all. The biggest problem in today's software development is the pace at which new frameworks, languages and tools are getting introduced.

With rapid pace of these tools being readily available, it is getting harder and harder to do your due diligence when it comes to vetting what is best for the company and the project. 30 years ago, it was hardly one or two main programming frameworks used for web development, while the introduction of so many tools and resources does help developers, it is often counterproductive for security.

SOFTWARE SECURITY IN SDLC

The best strategy to maintain software is to ensure security at all times while the product is being developed. There are many factors that can influence security during the SDLC phase and we will discuss these phases next.

I. Requirement Analysis

In this phase, we need to make sure that we do threat modelling along with noting down all of the security requirements as well. As much of it is documented, the better it is for all the stakeholders involved in the project. This also sets the right expectations from the start.

II. Design

We need to make sure that the designs are good and the cohesiveness between different objects and components of the software do not leave any unchecked vulnerability open to be attacked.

III. Implementation

We should also Follow coding standards that emphasize security, such as OWASP's Secure Coding Practices. We should also make sure we incorporate different security techniques alongside the coding sprints like Static Analysis.

IV. Code Reviews and Testing

Code reviews using dynamic testing tools and the most important human code review is imperative to the SDLC lifecycle. We need to make sure that we follow all best practices and do not leave any unchecked code blocks.

V. Deployment and Maintenance

Deployment now a days is a big thing. We should make sure that we go for as much managed servers as possible. That is why cloud is a very secure option in comparison to spending the time to set up your own servers and managing all nodes and resources. There are many options in the cloud that take the guaranteed deployment and zero down time deployments which are good for secure and scalable applications.

We should also make sure that our apps are regularly patched with the latest software updates and in the event of a new vulnerability, we can react to that fix very quickly.

You can ensure that security is a fundamental component of the product and not just an afterthought by integrating security principles throughout the SDLC. This "shift-left" strategy aids in early vulnerability detection and correction, lowering the overall risk and potential expense of addressing them in the future.

REGULATIONS AND COMPLIANCE

Modern software development has changed a lot. When it comes to new software, there are numerous new regulations and compliance policies introduced with them to ensure the security and privacy of its users and the software itself. Governments throughout the world have developed legislation and compliance standards to set the bar for software development and data protection in the age of data breaches, cyberattacks, and growing privacy concerns.

I. **GDPR or General Data Protection Regulation** was enacted by the European union and mandates the software companies to make sure that there is no data breach of the users data. There are also fail safe mechanisms that need to be set within the software in case of a breach.

II. **CCPA or California Consumer Privacy Act** is a regulation by USA that is pivotal in providing rights to the consumers of goods and services. They give their personal information to these organizations and the organizations can not distribute their data without asking for consent.

III. **HIPAA** is also one of the biggest regulations out there that makes sure that private health information always stays private. No other doctor or human should be able to access anyone's medical records unless specifically authorized by the patient.

While these regulations are important, they also introduce a lot of complexity within the software development plan and also limit innovation. That is why we have to go above and beyond to incorporate practices that will not shadow our creativity and give us a better output.

RESEARCH FINDINGS

We have gone through a series of literature and surveys done on this topic and have found accurate input and things that work for modern software development while encouraging innovation.

I. **The security development lifecycle by howard and lipner** presents microsofts security development lifecycle (SDLC) as a comprehensive framework for integrating security into software development processes. This paper advocates for the adoption of secure coding standards and emphasizes the importance of integrating security throughout the entire software development lifecycle.

II. **Classifying software changes clean or buggy? By kim, whitehead, and zhang** introduces a classification approach to distinguish between clean and buggy changes in software code, providing a means to identify potential vulnerabilities introduced during code modifications. This research sheds light on the importance of continuous monitoring and assessment of code changes.

III. **Vccfinder: finding potential vulnerabilities in open-source projects to assist code audits proposes vccfinder**, a tool for automatically identifying potential vulnerabilities in open-source projects. This tool streamlines code audits and contributes to the reduction of security weaknesses in open-source software.

IV. **An Approach for Secure Software Development Life Cycle in Small Software Firms by Prakash Chandra Behera** talks about the issues with development lifecyle and the security vulnerabilities that come with smaller working teams. It touches upon the topic of unrealistic deadlines and lack of training of new resources involved with this type of working model.

V. **Secure Software Development Lifecycle: A case for Adoption in software SMES by Wisdom Umeugo** talks about the three principles of software security which are Confidentiality, Integrity and Availability. It also talks about malicious attacks in different phases of a software's life. They introduce various frameworks like Comprehensive Lightweight Application Security Process and Security Assurance Maturity Model (SAMM). It also talks about Building Security in Maturity Model (BSIMM) and Microsoft Security Development Lifecycle (and Agile) introduced by Microsoft.

VI. **Security in the Software Development Lifecycle by Hala Assal and Sonia Chiasson** goes beyond the case study and actually conducts live interviews with developers to gather insightful data that involves software development. They also discuss how team size and other variables like project deadlines and software model (agile, waterfall etc) influence the importance given to security in general.

VII. **How to Manage Security & Third Party/Open Source Code in the SDLC by Drew Brown** mentions the importance of security issues when using 3rd party tools or open source code in your project. It talks about different types of scans like OS, IAST and DAST scans and how imperative they are when using open source code to make sure there are no vulnerabilities in the version we are using.

ENHANCING RESEARCH THROUGH NEW TECHNIQUES

This analysis also serves as a platform to explore how new techniques and methodologies can enhance the works of these papers. The continuous evolution of the software development landscape, along with emerging threat vectors, necessitates novel approaches to secure software development. By leveraging the findings from these papers, we can identify areas where innovation can lead to improved research outcomes:

I. Incorporating Machine Learning and AI

New techniques, such as the integration of machine learning and artificial intelligence, can bolster the classification system proposed by kim et al. This can enable automated detection of potential vulnerabilities in code changes, improving the efficiency of their methodology.

II. Security Automation and DevSecOps

Building on the foundations of the security development lifecycle, embracing devsecops principles and security automation can further streamline the integration of security controls throughout the software development process. This can create a more agile and responsive approach to security

III. Advanced Code Analysis Tools

The research presented in vccfinder can be enhanced through the development of more advanced and accurate code analysis tools. The incorporation of advanced static and dynamic analysis techniques, along with improved vulnerability databases, can result in more effective vulnerability detection and mitigation.

IV. Implementing CI/CD Pipelines

One of the innovative techniques introduced is the implementation of CI/CD (continuous integration/ continuous deployment) pipelines, especially for code qualification with stringent security measures. This approach promises to enhance the existing coding standards by automating security checks at every stage of development, thereby improving overall code quality and security.

V. Holistic Security Integration

Beyond the immediate research findings, these innovations can infuse security throughout the entire software development lifecycle. They bolster not only the coding stage but also extend their influence to the critical phases of analysis, design, planning, implementation, and maintenance, thereby providing a proactive approach to software security that permeates every facet of the development process.

VI. Preventive Measures in Every Phase

The application of innovative techniques should encompass a proactive stance in each phase of the software development lifecycle, fortifying the software against potential vulnerabilities and security threats before they arise. This approach shifts the focus from reactive measures to an ongoing, anticipatory strategy that safeguards the software from the very outset.

Figure 1: Implementation of Security in SDLC

This analysis aims to not only provide a comprehensive understanding of the existing research but also to highlight the potential for innovation and improvement in the realm of secure software development practices. It serves as a foundation for exploring how new techniques and approaches can further advance the state of the art and contribute to more secure software development in a rapidly evolving threat landscape.

CONCLUSION AND INITIAL RESULTS

In conclusion, this analysis has undertaken a thorough exploration of eleven pivotal research papers in the realm of secure software development. We have dissected their primary objectives, highlighting the significance of adopting secure coding standards, integrating security controls across the software development lifecycle, identifying prevalent security weaknesses, and the pivotal role of security awareness and training programs.

Notably, we have ventured beyond the confines of these seminal papers to envision how innovative techniques can enhance and extend their research outcomes. The introduction of machine learning, ai, security automation, advanced code analysis tools, and ci/cd pipelines has the potential to revolutionize the field of secure software development. These techniques not only improve coding practices but also cast a protective net over the entire software development lifecycle, from initial analysis to ongoing maintenance.

Furthermore, we emphasized the importance of taking a proactive stance in every phase of software development, fortifying applications against potential vulnerabilities and security threats before they materialize. This anticipatory approach shifts the paradigm from reactive measures to a continuous and preemptive strategy that safeguards software assets from inception to end-of-life.

As the digital landscape continues to evolve, the importance of secure software development practices cannot be overstated. The

insights from these research papers, coupled with the innovative techniques proposed, provide a roadmap for organizations and developers to navigate the intricacies of an ever-changing threat landscape. By embracing these strategies and practices, software development teams can ensure that security remains at the forefront of their operations, guarding against vulnerabilities and threats at every turn.

WORK DISTRIBUTION

Until this point of the project, we have already finished nearly half the amount of research papers we wish to complete. We are going through every paper individually first and trying to cross reference ideas and techniques that can better the software security principles being used in today's era.

The group member leading the project is Zaki Ahmad Rathore. We have divided the papers in equal parts, and we have weekly checkpoint meetings in which we get together to discuss any finished papers and take notes about what we can include in our project report. Up until now, we have finished 11 papers and we have divided them into 4-4-3 denominations. Shishir and Vigneshwar completed 4 papers each and sent the findings to Zaki. Zaki completed the rest of the papers and wrote the checkpoint report. The report was then further discussed within the team for errors and possible corrections that could be done. The number of total hours spent individually are very hard to track but a good guess would be 40 hours for each group member involved.

REFERENCES

[1] Behera, P. C., Dash, C., & Yadav, S. K. (2021, May 03). An Approach for Secure Software Development Life Cycle in Small Software Firms. Indian Journal of Natural Sciences. Retrieved from https://www.researchgate.net/publication/353306538_An_Approach_for_Secure_Software_Development_Life_Cycle_in_Small_Software_Firms

[2] Umeugo, W. (2023). Secure Software Development Lifecycle: A Case for Adoption in Software SMEs. International Journal of Advanced Research in Computer Science. Retrieved from https://www.researchgate.net/publication/368737283_SECURE_SOFTWARE_DEVELOPMENT_LIFECYCLE_A_CASE_FOR_ADOPTION_IN_SOFTWARE_SMES

[3] Assal, H., & Chiasson, S. (2018, August 12–14). Security in the Software Development Lifecycle. The Advanced Computing Systems Association. Retrieved from https://www.usenix.org/system/files/conference/soups2018/soups2018-assal.pdf

[4] Davis, N. (2013, July). Secure Software Development Life Cycle Processes. Software Engineering Institute, Carnegie Mellon University. Retrieved from https://insights.sei.cmu.edu/documents/430/2013_019_001_297287.pdf

[5] Roshaidie, M. D., Liang, W. P. H., Jun, C. G. K., Yew, K. H., & Zahra, F.-t.-Z. (2020, December). Importance of Secure Software Development Processes and Tools for Developers. Taylor's University Selangor, Malaysia. Retrieved from https://www.researchgate.net/publication/348078670_Importance_of_Secure_Software_Development_Processes_and_Tools_for_Developers#fullTextFileContent

[6] Brown, D. (2019, August 15). How to Manage Security & Third Party/Open Source Code in the SDLC. CISO Platform. Retrieved from https://www.cisoplatform.com/profiles/blogs/how-to-manage-security-amp-third-party-open-source-code-in-the

[7] Software Development Life Cycle - SDLC. (2018, November 07). Retrieved from https://devqa.io/software-development-life-cycle-sdlc-phases/

[8] Howard, M., & Lipner, S. (2006, May). The Security Development Lifecycle. Microsoft Press. ISBN: 0735622140. Retrieved from https://www.researchgate.net/publication/234792172_The_Security_Development_Lifecycle

[9] Perl, H., Dechand, S., Smith, M., Arp, D., Yamaguchi, F., Rieck, K., Fahl, S., Acar, Y. (2015). VCCFinder: Finding Potential Vulnerabilities in Open-Source Projects to Assist Code Audits. Fraunhofer FKIE, Germany; University of Bonn, Germany; University of Göttingen, Germany; Saarland University, Germany. Retrieved from https://saschafahl.de/static/paper/vccfinder2015.pdf

[10] Chang, R.-Y., Podgurski, A., & Yang, J. (2008, September). Discovering Neglected Conditions in Software by Mining Dependence Graphs. IEEE Transactions on Software Engineering, 34. doi: 10.1109/TSE.2008.24. Retrieved from https://www.researchgate.net/publication/220070307_Discovering_Neglected_Conditions_in_Software_by_Mining_Dependence_Graphs

[11] Kim, S., Whitehead, E. J. Jr., & Zhang, Y. (2008). Classifying Software Changes: Clean or Buggy? IEEE Transactions on Software Engineering, 34(2). Retrieved from https://groups.csail.mit.edu/pag/pubs/kim-tse-2008.pdf

[12] Citation: Figure1: https://intellisoft.io/what-is-a-secure-software-development-life-cycle-stages-methodologies-and-best-practices/