International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 05 | May 2022 www.irjet.net p-ISSN: 2395-0072

Security Introspection for Software Reuse

Sheleshma Shukla[1], Dr.Dhirendra Pandey[2]

[1, 2] *Department of Information Technology, Babasaheb Bhimrao Ambedkar University, Lucknow, Uttar Pradesh*

Abstract - *Both scholars and practitioners agree that software reuse is a good idea. By depending on established relationships, a system can become more secure, or more insecure, by exposing a wider security vulnerabilities via susceptible repositories. In order to build on a previous study and shed further light on the topic, we look into the link between reusability and security flaws. We utilize a multiple-case research to examine 1244 open-source projects in order to investigate and debate the distribution of security vulnerabilities in code generated by a project team as well as code reproduced through dependents. For this, we take into account both possible vulnerabilities discovered through static analysis and publicly published flaws. The results indicate that the amount of possible vulnerabilities in both native and reused code is linked to the scale of a development. Furthermore, we noticed that the number of dependent and the number of vulnerability are closely related. According to our research, source code reuse is neither a panacea for addressing vulnerabilities nor a terrifying werewolf that entails an excessive number of them*.

Key Words: Vulnerabilities, Flexibility, Security Hazardous, Software Reuse etc.

1. INTRODUCTION

Modern component-based and service-oriented systems make use of reuse to provide multiple productivity and cost-cutting benefits. Instead of building entire applications and systems from the ground up, they are instead assembled from existing and newly developed components and services, reducing money and time to market. However, when such systems contain safety and security features, these gains are outweighed by a number of problems. (Although safety and security are two distinct domains, we will treat them together in this work wherever possible.) The link between software reuse and safety and security has at least two major difficulties. The attainment of flexibility is directly linked to reuse (see, for example, [1] for an economic analysis of flexibility in reuse). Traditional security and safety assurance concepts for monolithic systems, which rely on fixed, inflexible structures ideal for the types of analyses employed in privacy and protection assessment, approval, and accreditation, are irreconcilable with mobility.

Because both safety and security are emergent aspects of a system [2], ensuring individual components of the system is extremely challenging. Assurance is usually performed at the system level. The "local" nature of reusability vs the "global" nature of safety and security raises many challenges. The "local" nature of reusable components vs the "global" aspect of protection and wellbeing raises many challenges. The inventor of a component is difficult to know ahead of time the precise safety or security context in which it would be used, finding it challenging to develop well with appropriate attributes.

A software engineer is more likely to create a component than just a cyber-security or risk specialist. It's not always a smart idea to incorporate security features into elements since it blends functional requirement attributes, making repurposing more difficult. We've taken an as double approach to the problem. Prototype techniques for designing independent security rules and safety continuation; separation of responsibilities principle for decoupling non-functional features.

2. LITERATURE REVIEW

2.1 Security Concern:

Reusing software isn't a panacea. Some of its flaws are described as "hazardous" rather than "concerning," in the sense that one of the most significant adverse effects is the potential for security problems. In a study involving Kula et al. [1], it was found that, although more open-source software systems exist, over 80% of the systems relied on out-of-date external libraries, and 69% of the systems. Any security concerns presented were unknown to the developers questioned. Furthermore, in the state of New York, Snyk discusses the troubling findings of the Open-Source Security report.The number of disclosed cases increased by 88% between 2017 and 2019. Open-source libraries have vulnerabilities.

2.2 Detecting Vulnerable code:

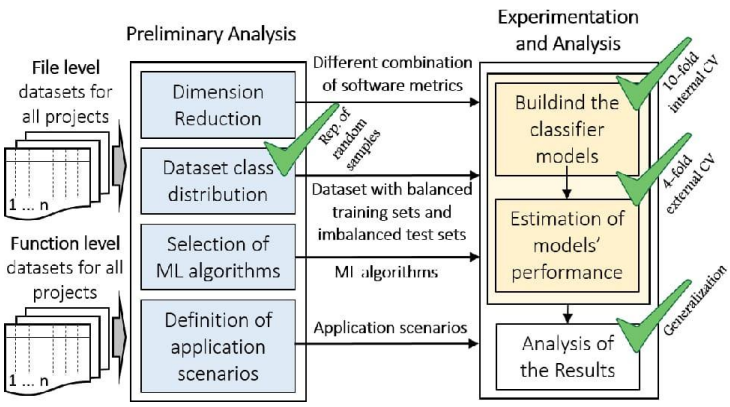
Pham et al. [3] suggested to the automated identification of non-useful of vulnerable codes in the area of finding susceptible codes. The authors introduced SecureSync, an open-source programme that analyses previously revealed vulnerabilities and they recommend systems and builds models to recognize similar suspicious trends in the data from different systems.

2.3 Policies and Requirements:

Security, like beauty, is a subjective concept. A public library's perspective on computer security will certainly differ from that of a major clearing house for interbank transactions. The specific security needs of a particular installation can only be determined after a thorough examination of the business environment, user preferences, and/or defence posture. The set of laws, policies, and practices that regulate how an organization manages, secures, and distributes sensitive information," according to the TCSEC [3] Glossary. The application of a high-level organizational policy to particular system needs is referred to as a security aspect. To match current usage, we'll use the phrase "security policy" to refer to both "policy" and "requirement" in the following sections.

2.4 Bringing Security and System models together:

Software engineers use models initiation is the first phase to improve the performance of artefacts like requirements papers. Early in a project's life cycle (e.g., requirement, design), attention to quality leads to defect discovery and avoid ability. It is commonly recognized that undiagnosed flaws can propagate downstream, dramatically increasing the cost of identification and eradication. Initiation is the first phase, the use of high-level, entity models (such as UML) to aid requirement problems and design processes has become widespread[4]. The ontology of the application domain is thoroughly examined in contemporary constraints modelling and entity design approaches. The rest of the requirements process is driven by the creation of a domain ontology model [5, 6, 7]. As evidenced by the Fu-sion [8] technique, this method was found to be helpful in practice and is frequently used in enterprise, notably in the implementation of the information systems. Reverse engineering can also benefit from modeling. Tools have been developed to extract models from an existing system [10, 9]. These kind of models can be useful for upkeep and re-engineering.



3. CONCLUSIONS

Software reuse is seen as a method to increase the efficiency and quality of software development. There is a lot of study going on in the field of software reuse. Lifecycle of software development. This paper is about security of reuse have been highlighted. The issue of reuse must be addressed. Where it satisfies the requirements of industry, as well as client requirements.

ACKNOWLEDGEMENT

I am grateful to my guide Dr. Dhirendra Pandey for their support in bringing out this paper successfully.

REFERENCES

[1]. Favaro, J., Favaro, K., Favaro, P.: Value Based Software Reuse Investment, Annals of Software Engineering (5), 1998, pp. 5-52.

[2]. Leveson, N.: Safeware: system safety and computers, ACM Press, New York, NY, 1995

[3] N. H. Pham, T. T. Nguyen, H. A. Nguyen, X. Wang, A. T. Nguyen,T. N. Nguyen, Detecting Recurring and Similar Software Vulnerabilities, in:40 Proc. 32nd ACM/IEEE Int. Conf. Software Engineering (ICSE '10), ACM,875 Cape Town, South Africa, 2010, pp. 227–230. doi:10.1145/1810295.1810336.

[4] I. Jacobson, M. Griss, and P. Jonsson. Software Reuse : Architecture Process and Organization for Business Success. Addison Wesley, 1997.

[5] A. Borgida, S. J. Greenspan, and J. Mylopoulos. Knowledge representation as the basis for require- ments specifications. IEEE Computer, 18(4):82-91, 1985.

[6] J. Mylopoulos, A. Borgida, M. Jarke, and M. K-oubarakis.Telos: Representing knowledge about in-formation systems. A CM Transactions on Office In-formation Systems, 8(4):325-362, October 1990.

[7] B. Nuseibeh and S. Easterbrook. Requirements engi-neering: a roadmap. In A. Finkelstein, editor, "The Future of Software Engineering", Special Volume pub- lished in conjunction with ICSE 2000, 2000.

[8] D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, and P. Jeremaes. Object- Oriented Development: The Fusion Method. Prentice- Hall, 1994.

[9] P. Devanbu and W. Frakes.Extracting formal domain models from existing code for generative reuse. ACM Applied Computing Review, 1997.

[10] D. Jackson and A. Waingold. Lightweight extraction of object models from bytecode. In Proceedings of the 1999 international conference on Software engineer- ing, Los Angeles, CA, May 1999.

BIOGRAPHIES

Sheleshma Shukla is a M.Tech Scholar of Department of Information Technology in Babasaheb Bhimrao Ambedkar University, Lucknow. Her research areas are software engineering and software security.

Dr. Dhirendra Pandey is working in the Department of Information Technology of Babasaheb Bhimrao Ambedkar University, Lucknow and he has obtained MPhil Degree in Computer Science from M. K. University, Madurai in 2008 and received PhD Degree in Computer Science from Devi Ahilya University, Indore in 2012. He has more than 15 years of Teaching and Research experience. Dr. Pandey has successfully executed Major Research Project from Council of Science and Technology of Uttar Pradesh under Young Scientist Scheme. He has published more than 75 research papers in various reputed International Journals and authored 06 Books and 07 Book Chapters. Dr. Pandey's research areas are Software Engineering, Theoretical Computer Science, Software Security and Data Mining and Warehousing.