ENHANCEMENT OF TASK ORIENTED MAINTENANCE MODEL USING SECURE SOFTWARE DESIGN MAINTENANCE

ESSA ZAKI ABDULRAZZAK

A project submitted in partial fulfillment of the
requirements for the award of the degree of
Master of Computer Science (Information Security)

Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia

JANUARY 2013

This project is dedicated to my family for their endless support and encouragement.

ACKNOWLEDGEMENT

First and foremost, I would like to express heartfelt gratitude to my supervisor **Dr. Imran Ghani** for his constant support during my study at UTM. He inspired me greatly to work in this project. His willingness to motivate me contributed tremendously to our project. I have learned a lot from him and I am fortunate to have him as my mentor and supervisor

Besides, I would like to thank the authority of Universiti Teknologi Malaysia (UTM) for providing me with a good environment and facilities such as employees in CICT help me to validate the project enhanced model and gave me some information which I need during validation process.

Last but not the least, I would like to thank my family especially my parents and my wife, for encouraging me to complete my postgraduate studying of master degree and supporting me spiritually throughout my life.

ABSTRACT

Most of the software today are not secure and contain security vulnerabilities that can be exploited by people with malicious intend to cause financial and physical damage. One of the reasons is that most research efforts have been put into the general development and maintenance processes with the implementation of some models. One such model for maintenance of software is task oriented maintenance model. This maintenance model does not focus on how to maintain secure software. Thus, this project identifies software design issues that need to be addressed in maintenance stage. In order to do this, we enhance the task oriented maintenance model to task oriented security maintenance (TOSiM) model. The proposed enhanced TOSiM model aspired to avoid design vulnerabilities by considering security features. In order to study the concept suitability of the model, two case studies have been conducted with software industry experts and the results are analyzed. The analysis shows that the enhanced model can be used to guide software designers/architects that fulfill their needs for how to maintain secure software design with less vulnerability.

.

ABSTRAK

Kebanyakkan perisian pada masa kini adalah tidak selamat dan mengandungi kelemahan-kelemahan yang boleh diguna oleh orang-orang yang berniat jahat dan akan menyebabkan kerosakan dari segi kewangan dan juga fizikal. Antara faktor-faktornya ialah kebanyakan usaha-usaha penyelidikan telah digunakan untuk penambahbaikan dalam proses-proses pembangunan-pembangunan umum dan juga proses-proses penyelenggaraan. Salah satu model ialah model tugas yang berorientasikan penyelenggaraan. Model ini tidak fokus dalam bagaimana untuk mengekalkan keselamatan perisian. Oleh itu, matlamat kajian ini ialah untuk mengenal pasti isu-isu reka bentuk perisian yang perlu dipertengahkan dalam peringkat penyelenggaraan dan untuk meningkatkan model penyelenggaraan ini. Peningkatan model yang dicadangkan ini berhasrat untuk mengelak kelemahan-kelemahan pada reka bentuk dengan mempertimbangkan ciri-ciri keselamatan. Metodologi pembangunan perisian yang selamat menyediakan cara-cara untuk mengintegrasikan keselamatan dalam perisian semasa pembangunannya. Pembangunan perisian yang selamat berkemungkinan ialah keselamatan keperluan proses, keselamatan proses reka bentuk, suatu set garis panduan dan prinsip-prinsip keselamatan. Semasa menjalankan penyelanggaraan, dua kajian-kajian case telah dijalankan dengan pakar-pakar industri perisian dan keputusannya dikaji. Maklum balas menunjukkan bahawa model yang dipertingkatkan boleh digunakan untuk membimbing pereka bentuk/arkitek perisian dalam mengekalkan reka bentuk perisian yang selamat dan kurang kelemahan.

TABLE OF CONTENT

LIST OF FIGUERS

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Software is one of the main components of computer system, it is operating all hardware parts of computer machine even though operating all computerize machine in the current time. Therefore software industry expands greatly with increasing of computer users because of all universities, governments and business workers created demand for software. Software development organization implements process for constructing software and used standard technique to write the software. Software development does not stop when the system is delivered to the client but continue for life time of the system. Most large companies spend a lot of money to use of software for many years to get back on it is investment but business change and change of user expectation and operational environment generate new requirement for existing software. The software traceability ensures design maintenance traceable to keep the design component for software link to reflect good requirement for user and adapted with new change. There are many secure software maintenance efforts in the directions of building secure software design and one of these efforts using traceability process to trace the designing of the software from requirements and throughout building secure software design.

Software maintenance changes should not expose the system to threats to its confidentiality, integrity and availability. The effect of maintenance change in the security of the software should first be evaluated when the change is occurred in the design and later during the verification process. In this project will propose a model for improving traceability in the design maintenance. The model can be evaluated using case study.

1.2 Problem Background

Software maintenance activity is performed by giving feedback and defects report to the vendor and asking for corrections. The corrections correct the defects or security vulnerabilities. It is important for building, maintaining and reuse software to improve the functionality of software design, and accurate traceability need to be resilience to change as possible. The traceability links remain true even when the model change (Yu, Jurjens et al. 2008). The resilience change in software property can help to reduce the effort in maintenance modification. Most refactoring steps are used to improve the understandability of maintenance process. Software design is the most important process of software maintenance activity, so that how to trace the security issues in the design phase in the early stage of software development and what are the security vulnerabilities in the design that threaten software especially in web application.

This study is concentrated for enhancing one established model called task oriented maintenance model and the enhancing collaborate with some phases which are related to requirement phase and design phase. Another objective of this study is to validate the model in the real life.

1.3 Software Maintenance

Apache http server and Mozilla web browser have been studied by (Koponen and Hotti 2005) of two large projects and came out with the conclusion that maintenance process in software is alike to the common vision of the maintenance process defined in the standards ISO/IEC 12207 (1995) and ISO/IEC 14764 (1999) (See Figure 1.1).

In the picture maintenance process is containing problem and modification analysis, maintenance review/acceptance and modification implementation which is connected with cyclic relationship. After these processes, software will entering retirement and migration phase which is can be the end of software life cycle.

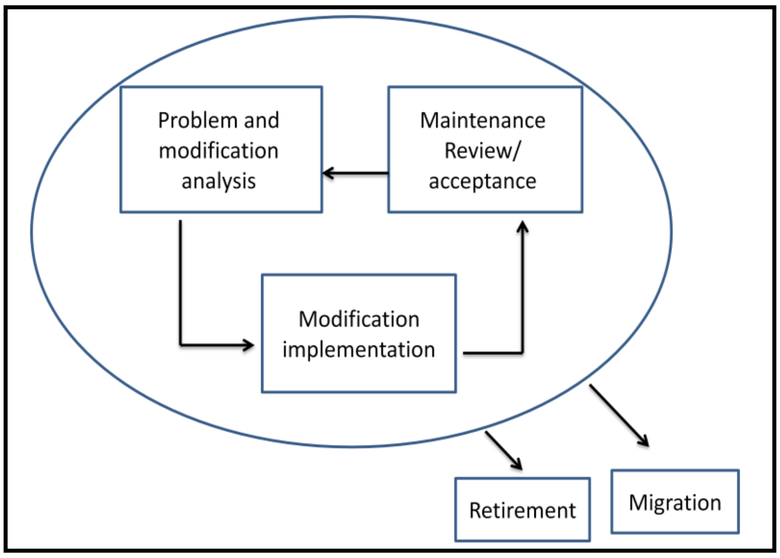
Problem and modification analysis
Maintenance Review/ acceptance
Modification implementation
Retirement
Migration

Figure 1.1 ISO/IEC Maintenance Process Activities

1.4 Problem Statement

The research highlights security issues in the design for secure software maintenance. The maintenance process used to trace the development of software from requirement through design process to vet the software functionality and find any missing security requirements that is not allocate through design process. Traceability process ensure that design satisfies the security requirements and the implementation does not digress from secure design. The previous studied maintenance model has been developed, but did not focus on the vulnerabilities in the design for secure software maintenance as mention it in problem background.

Indeed, this study will improve software design architecture during maintenance process and strive to reduce or mitigate security flaws that designer may overlook it's in the design for secure software. Web application attacks nowadays exploits design flaws with malicious intend to abnormal use software systems and breaks security protection. In addition, most of users not worry about security principles during collection of requirements. But during analyses process the user remembered that do not concentrate on security requirement which is significantly used to reduce software threats and security vulnerabilities in the software design. Ultimately, this study enhances task oriented maintenance model by utilizing security principles and guidelines in the design of secure software maintenance. The model highlights the following questions:

- What are the design issues that are occurred during maintenance of secure software design?
- What are the security enhancements can be proposed to reduce software vulnerabilities in secure software design maintenance?
- Does the software maintenance functionality work as it is suppose to do?

1.5 Project Objectives

1. To identify software design issues that needs to be addressed in maintenance process.

2. To propose security enhancement for task oriented maintenance model using secure software design maintenance.
3. To analyzed and validate the enhanced model by conducting case study via software industry experts.

1.6 Project Scope

1) The study focus on enhancing security traceability in the design of software maintenance model.
2) The proposed model will be evaluated using case study can be conducted at UTM CICT for software development.

1.7 Project Organization

This thesis is divided into six chapters. The first chapter presents the introduction of study which contains the explanation of background, problem statement, scope, objectives and the contribution. The study of literature will be reviewed in Chapter two. This chapter is concentrated in reviewing current design issues of software maintenance, Software maintenance models, select one maintenance model to enhance, traceability and some explanation about security requirements and design phase. Chapter three discuss about methodology of the research. The implementation of enhancements for selecting TOSiM model, proposed enhancement processes, maintenance process in design, target system and validation of TOSiM model using survey all can be described in Chapter four while Chapter five present implementation of survey and data collection for TOSiM Model. Chapter six explains about the conclusion and future work.

1.8 Conclusion

Security threats in software systems are a major dangerous that threatens computer software. In the early stages of software development ,these issues need to handle , so that this project propose security enhancements in one maintenance model by using secure design features and analysis, while security threats modeled by use and misuse case ,attack tree and attack pattern. Security design analyses method give a good guidance in detailed design validation of the system implementation. This method assists software designers/architects to discover security vulnerabilities in the early stage of design and to utilize security mitigation techniques to reduce it.

The project used some secure design practices and guidelines that are reported in chapter two. It is expected that these practices aid the developers to respect security techniques and grow the number of security issues that are encountered.

REFERENCES

April, A. and A. Abran (2009). "A Software Maintenance Maturity Model (< i> S3M</i>): Measurement Practices at Maturity Levels 3 and 4." Electronic Notes in Theoretical Computer Science **233**: 73-87.

April, A., J. Huffman Hayes, et al. (2005). "Software maintenance maturity model (smmm): the software maintenance process model." Journal of software maintenance and evolution: Research and Practice **17**(3): 197-223.

Association, I. S. (1998). IEEE Std 1219-1998 IEEE Standard for Software Maintenance, IEEE–SA.

Banker, R. D., S. M. Datar, et al. (1993). "Software complexity and maintenance costs." Communications of the ACM **36**(11): 81-94.

Barnum, S. (2008). "Common attack pattern enumeration and classification (CAPEC) schema description." Cigital Inc, http://capec. mitre. org/documents/documentation/CAPEC_Schema_Descr iption_v1 **3**.

Bishop, M. (2004). Introduction to computer security, Addison-Wesley Professional.

Boehm, B. W. (1987). "Industrial software metrics top 10 list." IEEE SOFTWARE **4**(5): 84-85.

Brooks, F. P. (1987). "No silver bullet: Essence and accidents of software engineering." IEEE computer **20**(4): 10-19.

Buyens, K., B. De Win, et al. (2007). Empirical and statistical analysis of risk analysis-driven techniques for threat management. Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on, IEEE.

Canfora, G., J. Czeranski, et al. (2000). Revisiting the delta IC approach to component recovery. Reverse Engineering, 2000. Proceedings. Seventh Working Conference on, IEEE.

Dekleva, S. (1992). Delphi study of software maintenance problems. Software Maintenance, 1992. Proceerdings., Conference on, IEEE.

Deraman, A. (1995). "Requirement For A Software Maintenance Process Model: A Review." Malaysian Journal of Computer Science **8**(2): 174-202.

Dupuis, R. (2004). "Software Engineering Body of Knowledge."

Erdil, K., E. Finn, et al. (2003). "Software maintenance as part of the software life cycle." Comp180: Software Engineering Project.

Goertzel, K. M., T. Winograd, et al. (2007). Software Security Assurance: A State-of-Art Report (SAR), DTIC Document.

Gregoire, J., K. Buyens, et al. (2007). On the secure software development process: CLASP and SDL compared. Proceedings of the Third International Workshop on Software Engineering for Secure Systems, IEEE Computer Society.

Grubb, P. and A. A. Takang (2003). Software maintenance: concepts and practice, World Scientific Publishing Company Incorporated.

Hadawi, M. (2007). Vulnerability Prevention in Software Development Process. Proceedings of the 10th International Conference on Computer & Information Technology (ICCIT'07).

Haley, C. B., R. Laney, et al. (2008). "Security requirements engineering: A framework for representation and analysis." Software Engineering, IEEE Transactions on **34**(1): 133-153.

Hightower, R., M. K. Brady, et al. (2002). "Investigating the role of the physical environment in hedonic service consumption: an exploratory study of sporting events." Journal of Business Research **55**(9): 697-707.

Howard, M. (2007). "Lessons learned from five years of building more secure software." MSDN MAGAZINE **22**(11): 56.

Howard, M. and D. LeBlanc Writing Secure Code. 2003, Microsoft Press International.

Kajko-Mattsson, M. (2001). Motivating the corrective maintenance maturity model (CM< sup> 3</sup>). Engineering of Complex Computer Systems, 2001. Proceedings. Seventh IEEE International Conference on, IEEE.

Khan, M. K., M. A. Rashid, et al. (1996). "A task-oriented software maintenance model." Malaysian Journal of Computer Science **9**(2): 36-42.

Khan, M. U. A. and M. Zulkernine (2008). Quantifying Security in Secure Software Development Phases. Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International, IEEE.

Koponen, T. and V. Hotti (2005). Open source software maintenance process framework. ACM SIGSOFT Software Engineering Notes, ACM.

Kozlov, D., J. Koskinen, et al. (2008). "Assessing maintainability change over multiple software releases." Journal of software maintenance and evolution: Research and Practice **20**(1): 31-58.

Lewis, W. E. (2004). Software testing and continuous quality improvement, Auerbach publications.

Lientz, B. P. and E. B. Swanson (1980). "Software maintenance management: a study of the maintenance of computer application software in 487 data processing organizations."

Mamone, S. (1994). "The IEEE standard for software maintenance." ACM SIGSOFT Software Engineering Notes **19**(1): 75-76.

McGraw, G. (2004). "Software security." Security & Privacy, IEEE **2**(2): 80-83.

Moore, J. W. (1998). Software engineering standards, Wiley Online Library.

Mouratidis, H., P. Giorgini, et al. (2003). Integrating security and systems engineering: Towards the modelling of secure information systems. Advanced Information Systems Engineering, Springer.

Osborne, W. M. and E. J. Chikofsky (1990). "Guest Editors' Introduction: Fitting Pieces to the Maintenance Puzzle." IEEE SOFTWARE: 11-12.

Palvia, P., J. T. Nosek, et al. (1990). An empirical evaluation of system development methodologies. Managing information resources in the 1990s: proceedings of 1990 Information Resources Management Association international conference.

Peine, H. (2008). Rules of thumb for developing secure software: Analyzing and consolidating two proposed sets of rules. Availability, Reliability and Security, 2008. ARES 08. Third International Conference on, IEEE.

Pigoski, T. M. (2001). "Software maintenance." Encyclopedia of Software Engineering.

Richardson, T. and C. N. Thies (2012). Secure Software Design, Jones & Bartlett Learning.

Saini, V., Q. Duan, et al. (2008). "Threat modeling using attack trees." Journal of Computing Sciences in Colleges **23**(4): 124-131.

Saltzer, J. H. and M. D. Schroeder (1975). "The protection of information in computer systems." Proceedings of the IEEE **63**(9): 1278-1308.

Schneidewind, N. F. (1987). "The state of software maintenance." Software Engineering, IEEE Transactions on(3): 303-310.

Schneier, B. (1999). "Attack trees." Dr. Dobb’s journal **24**(12): 21-29.

Server, L. (2008). "Product Overview." Citrix Systems, available at: http://www. citrix. com/site/ps/products. asp.

Sherman, S. and I. Hadar (2012). Identifying the need for a sustainable architecture maintenance process. Cooperative and Human Aspects of Software Engineering (CHASE), 2012 5th International Workshop on, IEEE.

Sindre, G. and A. L. Opdahl (2008). "Misuse Cases for Identifying System Dependability Threats." Journal of Information Privacy and Security **4**(2): 3-22.

Singh, Y. and B. Goel (2007). "A step towards software preventive maintenance." ACM SIGSOFT Software Engineering Notes **32**(4): 10.

Smith, J. (2009). "Wpf apps with the model-view-viewmodel design pattern." MSDN MAGAZINE.

Sodiya, A. S., S. A. Onashoga, et al. (2006). "Towards building secure software systems." Issues in Informing Science and Information Technology **3**.

Team, C. P. (2002). "Capability Maturity Model® Integration (CMMI SM), Version 1.1." Software Engineering Institute, Carnegie Mellon University/SEI-2002-TR-012. Pittsburg, PA.

Thayer, R. H. and E. Yourdon (1997). "Software engineering project management." Software Engineering Project Management: 72.

Van Lamsweerde, A. (2004). Elaborating security requirements by construction of intentional anti-models. Proceedings of the 26th International Conference on Software Engineering, IEEE Computer Society.

Van Vliet, H. (2008). Software architecture knowledge management. Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on, IEEE.

Vehvilainen, R. (2000). What is preventive software maintenance? Software Maintenance, 2000. Proceedings. International Conference on, IEEE.

Yu, Y., J. Jurjens, et al. (2008). Traceability for the maintenance of secure software, IEEE.

Zitouni, M., A. Abran, et al. (1995). "Élaboration d'un outil d'évaluation et d'amélioration du processus de la maintenance des logiciels: une piste de recherche." Actes des Huitièmes Journées Internationales: Le Génie Logiciel et ses Applications.