ISBN 978-981-11-3671-9
Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)
Beijing, 25-27 June, 2017, pp. 454-459
doi: 10.18178/wcse.2017.06.078

Security Benefits for Agile Software Development

S. Hassan Adelyar [+] and Alex Norta

Institute of Informatics, Tallinn University, Tallinn, Estonia, adelyar@tlu.ee
Department of Informatics, Tallinn University of Technology, Tallinn, Estonia, alex.norta.phd@ieee.org

Abstract-Agile methodologies such as scrum, Extreme Programming (XP) and Feature-Driven Development (FDD) are efficient development processes by accepting changes at any phase and delivering software quickly to customers. However, these methodologies have been criticized because of the unavailability of security as an important quality goal of software systems. Although there are pre-existing research results on this topic, there is no pure approach for identifying security benefits of XP practices that relate to the core “embrace-changes” principle of agile. Specifically, we analyze XP practices to find the security benefits in customer- and developer activities.

Keywords-Extreme-programming; Embrace-changes; Software-security; Security-benefits; Security-principles

1. Introduction

The aim of this paper is to analyze XP practices in order to identify security benefits during software-development processes throughout the customer- and developer activities. XP is an iterative and incremental software development approach and each iteration involves the team to go through a full development cycle [1]. XP produces working software quickly while maintaining a high rate of customer satisfaction [2], [3], [4]. However, XP methodology do not pay attention to security features because the working software and iterative delivery are the primary measure of success [5]. At the same time, considering the current attacking landscape, security is an important non-functional requirement of software products. According to [6], software is vulnerable to threats that may occur during software-development processes and inadequate practices of software development can lead to insecure software [7].

One way to incorporate security into development team is by identifying security benefits of XP practices. We define security benefits from security principles described in Section 3-C. According to security principles, secure system can be studied by assessing four main characteristics which are: separation, restriction, simplicity and awareness [8]. Separation supports the accountability attribute of security. Restriction supports the confidentiality attributes of security. Simplicity assures that the development- team activities are valid and correct. Software- developer attention and awareness is required to supports confidentiality, integrity, availability and accountability. These benefits improve security of software by incorporating security principles [9] into XP features such as customer- and developer interaction, short iterations and responses to changes. Therefore, our aim is to improve XP methodology for producing secure software by considering the security benefits of XP. This is accomplished by analyzing XP practices to identify what activities of customers and developers are most beneficial for secure software development. This paper is a continuation of our previous work [10] in which we analyzed XP practices in order to identify security challenges based on the security principles.

[+] Corresponding author.
E-mail address: *adelyar@tlu.ee*

We conduct a case-study based research [11] about the development process of applications that follows XP practices to analyze the relationship between security principles [9] and security benefits of XP practices. For data collection, we use both interviews and focus-group methods.

The rest of the paper is organized as follows. Section II provides a summary of existing literature. Section III contains additional information relevant for XP practices, software security and security principles. Section IV presents a brief overview of our case-study approach. In Section V, we present the results of our research. Finally, Section VI concludes this paper by summarizing the research work, giving the contributions achieved and showing directions for future work.

2. Related Work

There exist many publications that criticize XP methodology because of unavailability of security elements in its development phases. On the other hand, in response to the increasing rate of security issues caused by security vulnerabilities in software products, many researchers publish about security integration with XP practices [12], [13], [14]. Researchers also found that many of the XP practices comply for building secure software [15], [16], [17].

A group of researcher study XP practices and discuss its potential benefits for secure software development [18], [19], [5]. Other researchers study XP methodology for integrating security into a specific practice, such as refactoring [20], [21]. However, based on literature, we found a gap pertaining to a holistic approach for identifying security benefits in XP practices based on a set of security principles [9]. Since XP focuses on communication, self-organization and the collaboration between developers and customers, therefore applying security principles on developer- and customer activities helps the developer team to understand security concern.

3. Background

In this section we briefly explain the concepts that are useful in understanding our approach. XP practices, described in Section 3-A, software security, described in Section 3-B and security principles, described in Section 3-C, are important elements in our research.

3.1. Agile Software Development

Agile is a dominant approach for software development and it is based on the concept of agility. In general, agility is the ability of the process to successfully cope with changes in requirements and delivering of working software in short duration [22]. XP is one of the wide-used methodology of agile and the cornerstone of XP methodology is the practices that help to produce software quickly. The twelve practices of XP are: planning-game, on-site customer, metaphor, small-releases, simple-design, pair-programming, collective-ownership, coding standards, 40-hour-week, continuous-integration, refactoring and testing.

3.2. Software Security

Security is a quality aspect of a system property that reflects the ability to protect itself from accidental or deliberate attacks. Security is a composite of the attributes confidentiality, integrity, availability and accountability [23], [24]. Confidentiality is defined as the prevention of unauthorized exposure of software code and execution. Integrity is the preventions of software code and execution from unauthorized alterations, amendment or deletion. Availability is the ability of software to be available when needed, executed in a predictable way and delivers results in a predictable time frame. Accountability is the availability and integrity of the identity of the person who performs an operation.

3.3. Security Principles

Security principles are defined by [9] and guide a software design and implementation without security flaws. These principles are concepts or guidance that can be followed to develop secure systems during a software development stage. Applying security principles on the software development process also helps non-security expert developers to understand security concerns. Security principles are: Separation of Privileges, Least Common Mechanism, Least Privileges, Complete Mediation, Fail-safe Defaults, Economy of Mechanism, Psychological Acceptability, and Open Design [9].

4. Case Study Design

We choose a case-study based research method [11] and our aim is to identify security benefits in developer- and customer activities of XP practices based on the security principles as listed in Section 3-C. Our research attempts to answer the following research question: How to identify XP security benefits during software development? To answer the main research question, we have three sub-questions: What are security benefits of response-to-changes based on security principles? What are the tasks that improve security benefits in XP software development? Which XP practices have more security benefits?

We answer these questions with our case-study data collection and analysis. The case for our study is a software development process using XP practices described in Section 3-A [11]. We select three different software development teams in Kabul city for interviews and one group of six developers as a focus-group. The subject for our study is security benefits in XP practices.

We use the same set of questions for conducting the interview and focus-group. Since our aim is to identify security benefits based on the security principles, therefore we derive the interview and focus-group questions based on the concept of security principles. As we mentioned in Section 1, according to the security principles, a secure system can be studied by assessing four main characteristics which are: separation, restriction, simplicity and awareness [8].

For data validity, we carefully design our study implementing the qualitative investigation measures in all phases of our case study. For ensuring construct validity, we use iterative questioning method during the focus-group and interview sessions. To avoid the effect of external factors such as the lower security awareness of developer, we carefully infer the interview and focus-group questions from security principles [9]. We use both focus-group and interview as two separate data collection methods. During the analysis the collected data we code and use themes to categorize and generalize the findings.

4.1. Data Collection Procedures

For answering our research questions we use two direct data collection methods that are focus-group discussions and interviews. We consider a focus-group comprising six developers and conduct interviews with 10 software developers. All the interviewees and focus-group members use an XP software development methodology and each member of the team has at least experience from three software-development projects. The interview and focus-group questions are derived from security principles listed in Section 3-C. The same questions are asked for the three main phases of XP practices, planning-game, pair-programming and continuous-integration. The mentioned three phases are collaborative and the activities of developers and customers in these practices are interdependent.

4.2. Analysis procedure

The main goal of analysis is to generalize the findings and find answers for our research questions. To achieve this goal, we analyze the collected data with the following steps:

1) First we formulate a set of themes that group the related codes.
2) We read all the texts in the collected data and mark where the codes fit into the themes.
3) Results of the coding are analyzed per theme and presented.

Table I shows our predefined themes and a brief description from which a corresponding theme is derived.

Table I: Themes and Themes description

Theme	Theme Description
Separation of privileges	To see how continuous changes to software make the process of separation of privilege easier to implement.
Restriction of privileges	Restrict privileges, check every access and deny access during mistakes.
Software simplicity	Make the design of software simple, small and easy.
Improve attention	To see how continuous changes to software improve the developer attention.

These themes are deduced from security principles that relate to our research questions. During software development, if the activities of developers and customers are in compliance with the security principles, then it reduces security flaws and vulnerabilities in the software. All codes are presented in Table II and Table III of our technical report [25].

For answering our research questions we analyze Table II and Table III of our technical report [25] to better make the coded data for identifying security benefits, the related tasks and XP practices that contain more benefits. Therefore, we combined Table II and Table III into Table IV, during combination we derived the benefits and related tasks from theme / code column of Table II and Table III. The practices column of Table IV is derived from the phase column of Table II and Table III. For each theme we select the three top ranked codes. The result of this process is shown in Table IV of our technical report [25].

5. Results and Discussion

As mentioned earlier, our goal is to analyze XP practices for identifying security benefits in customer- and developer activities based on security principles. For achieving this goal, we define the themes from security principles. Through the coding session, data collected from the focus-group and interview is organized into corresponding respective themes. After analyzing Table IV in our technical report [25] we found the following results:

Our First theme belongs to the security principles of "Separation of Privileges" and "Least Common Mechanisms". Based on these principles, the secure software development process must verify the identity of developers based on their responsibilities and minimize common mechanisms to more than one developer. From Table IV in our technical report [25] the benefits of clarity of customer & developer responsibilities, elimination of future disputes and identification of problem-source are the benefits to support this theme. All these benefits supports the "accountability attribute" of security, described in Section III-B. The tasks that improve these benefits are customer feedback, incremental development and privileges assignment. The most important practices for these benefits and tasks are planning game, pair programming, continuous integration, refactoring, on-site customer, small release and collective ownership.

Next, our second Theme belongs to "Least Privileges", "Complete Mediation" and "Fail-safe Default" security principles. Based on these principles, security attributes have a system-wide nature and the protection and authorization mechanisms for developing secure software, requires the restriction of privileges. From Table IV in our technical report [25] the benefits of system-wide view, understanding the software, knowing all parts of the software, verification of code, prevention of unwanted changes and prevention of errors are the benefits to support this theme. All these benefits supports the "confidentiality and availability attributes" of security, described in Section III-B. The tasks that improve these benefits are incremental development, customer feedback, iterative work and pair working. The most important practices for these benefits and tasks are planning game, continuous integration, refactoring, on-site customer, small release and collective ownership.

The third theme we derive from "Economy of Mechanism", "Open Design" and "Psychological Acceptability" security principles. Based on these principles, the design must be simple and small since techniques such as line-by-line inspection are necessary for finding security flaws in the code of software. For such techniques to be successful, a small and simple design is essential [9]. The focus-group and interviewees pointed to the simplicity of software development, software- quality improvement, and ease of changes as the benefits that improve integrity. The tasks of working on one task at a time, working on each part separately, discussion of pairs, privileges determination, customer feedback and incremental development are tasks that improve the simplicity of software. The most important practices for these benefits and tasks are continuous integration, small release, simple design, refactoring, planning game and coding standards.

Finally, the last Theme of "Attention and Awareness", we derive from the security principle of "Fail-safe Defaults". This principle emphasizes security mechanisms that require high attention of developers during the whole software development process. From Table IV in our technical report [25] the benefits of understanding & attention improvement, concentration & competition, improvement of developer focus on the software and reduction of errors are the benefits to support this theme. These benefits supports the confidentiality, integrity, availability and accountability attributes of security, described in Section 3-B. The tasks that improve these benefits are iterative work, working in pairs, and on-time customer feedback. The most important practices for these benefits and tasks are pair programming, continuous integration, coding standards, refactoring, planning game and small release.

6. Conclusion and Future Work

In this paper, we conduct a case study to identify and explain security benefits of XP software development by evaluating developer- and customer activities. The result of our study shows that adequate tasks and activities of developers and customer, reduce security flaws and vulnerabilities from the developed software. Table V in our technical report shows the tasks (customer- and developer activities), its security benefits and XP practices for security benefits. Table V shows that many tasks in developer- and customer activities of XP methodology improve software security through “Separation of privileges”, “Restriction of privileges”, “Software simplicity”, and “Developer attention”.

As a limitation of this research, the interviewed developers have lower security knowledge and we are not able to be a member of the development team in order to observe the developer activities. Further studies and future work for introducing visual methods such as agent-oriented modelling (AOM) [26] can be studied for raising security awareness of agile software developers.

7. References

[1] C. Lan and B. Ramesh, "Agile Requirements Engineering Practices: An Empirical Study," 2008.

[2] H. John, "Agile Software Construction.," 2005.

[3] B. Gabrielle, "Rolling out agile in a large enterprise.," in 41st Hawaii International Conference on System Science. , 2008.

[4] I. Ghani and I. Yasin, "Software Security Engineering in Extreme programming Methodology: A Systematic Literature Review," ISSN 1013-5316; CODEN: SINTE, pp. 215-221, 2013.

[5] J. Wayrynen, M. Boden and G. Bostrom, "Security Engineering and eXtreme Programming: An Impossible Marriage".Comminications Security Lab, Ericsson Research.

[6] Cappelli, Dawn, Trzeciak and Randall, Insider Threats in the SDLC. Presentation at SEPG, 2006.

[7] C. Mann, "Why Software is so Bad?," in Technology Review, 2002.

[8] Y. Michael and T. Issa, "Properties for Security Measures of Software Products.," 2007.

[9] J. H. Saltzer and M. D. Schroeder, "The Protection of Information in Computer Systems," pp. 1278 - 1308, 1975.

[10] H. Adelyar and A. Norta, "Towards a Secure Agile Software Development Process," pp. 101-106, 2016.

[11] P. Runeson, M. Host and A. Rainer, Case Study Research in Software Engineering, New Jersey, USA: John Wiley, 2012.

[12] M. Siponen, R. Baskerville and T. Kuivalainen, "Integrating security into agile development methods., 2005.," in In Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005.

[13] V. Kongsli, "Towards Agile Security in Web Applications. In the Proceedings of OOPSLA," in OOPSLA, Oregon, USA, 2006.

[14] G. Bostrom and B. Konstantin, "Extending XP Practices to Support Security Requirements Engineering," in ICSE, University of British Columbia, Canada, 2006.

[15] P. Amey and R. Chapman, "Static Verification and Extreme Programming.," in Proceedings of the ACM SIGAda Annual International Conference., 2003.

[16] O. Murro, R. Deias and G. Mugheddo, "Assessing XP at an European Internet Company.," 2003.

[17] J. Shore, "Continuous Design. IEEE Software, Vol. 21 (1).," 2004.

[18] B. Konstantin, "Extreme Security Engineering: On Employing XP Practices to Achieve Good Enouht Security," First ACM Workshop on Business Driven Security Engineering, p. 7, 2003.

[19] C. Wood and G. Knox, "Guidelines for Agile Security Requirements Engineering".

[20] E. Aydal, R. Paige, H. Chivers and P. Brooke, "Security Planning and Refactoring in Extreme Programming," Springer Link, vol. 4044, pp. 154-163, 2006.

[21] Sonia, A. Singhal and J. Balwani, "Analysing Security and Software Requirements using Multi-Layered Iterative Model," IJCSIT International Journal of Computer Science and Information Technology, vol. 5, no. 2, pp. 1283-1287, 2014.

[22] T. Kurian, Agility in IT: Creating Software Through Agile Techniques., Bangalore, India: Proceedings of the IEI International Conference on Agility Design & Manufacturing (ADEMSE-05), 2005.

[23] A. Avizienis, J.-C. Laprice, B. Randell and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," IEEE Transaction on Dependable and Secure Computing, vol. 1, pp. 11-33, 2004.

[24] C. Pfleeger and S. Lawrence, Security in Computing, New Jersey, USA: PRENTICE HALL, 2003.

[25] A. Hassan and N. Alex, "Security Benefits for Agile Software Development," in https://digi.lib.ttu.ee/i/?7405, 2017.

[26] S. S. Leon and T. Kuldar, The Art of Agent-Oriented Modelling, London: Massachusetts Institute of Technology, 2009.