Journal of Software Engineering

ISSN 1819-4311

Journal of Software Engineering 6 (1): 10-15, 2012
ISSN 1819-4311 / DOI: 10.3923/jse.2012.10.15
© 2012 Academic Journals Inc.

Secure Spiral: A Secure Software Development Model

[1]Daljit Kaur, [2]Parminder Kaur and [2]Hardeep Singh
[1]Department of Computer Science, Lyallpur Khalsa College, Jalandhar, India
[2]Department of Computer Science and Engineering, G.N.D. University, Amritsar-143005, India

Corresponding Author: Parminder Kaur, Department of Computer Science and Engineering, G.N.D. University, Amritsar-143005, India

ABSTRACT
Secure software is the demand of time in this connected world. Security needs to be given high priority in software development life cycle. Considering security as a non-functional requirement in software and giving a side thought to it after development of software only results a software with vulnerabilities. Software engineering is still lagging in secure software development processes. Due to it's critically, security should be integrated in software life cycle process from the very beginning of development of software. Current research implements the secure development phases in spiral model and proposes a new security aware spiral.

Key words: Software security, spiral model, secure software development life cycle

INTRODUCTION
Security is considered very critical issue in software systems. But often it is given a thought after software development. After the completion of software development, organizations try to incorporate security as a patch (McGraw, 2004). Moreover, securing operational environment using firewall and antivirus programs does not work and organizations continue to incur heavily losses due to exploitation of vulnerabilities or software flaws (Ahmed, 2007). Security is an emergent property of a complete system, not a feature (McGraw, 2004). But, giving only a post development consideration to security means that software is still developed with vulnerabilities inside (Shirazi, 2009).

In last few years of advancements in software engineering and the development of tools, progress in improving security of software is still lagging (Gillam *et al.*, 2003). A life cycle process that includes security assurance is needed for improving security of software.

According to security in software life cycle-secure software cannot be intentionally subverted or forced to fail. It is, in short software that remains correct and predictable in spite of intentional efforts to compromise that dependability (McGraw, 2006). Current research indicates that good software engineering approach is to consider security from the early stages of life cycle (McGraw, 2006).

This paper mainly focuses on secure life cycle of software that requires a thorough consideration that includes security in requirement analysis, design, implementation and testing phase. Main focus of this technique is to identify security risks and managing those risks. Based on spiral model and its risk perception, a new model has been proposed. The new model is a blend of spiral with security and is named as SaS (Security aware Spiral). This new model focuses on security risk management and review after each cycle.

SECURE SOFTWARE DEVELOPMENT

Software that is developed with security in mind is typically more resistant to both intentional attack and unintentional failures (Allen *et al.*, 2008). Also, it has been seen that if security is implemented right from the inception of software, it saves the economy billions of dollars. The section below discusses various security actions to be taken at each phase of Software Development Life Cycle (SDLC) in software engineering.

Security in requirements phase: Inadequate requirement analysis can cause many problems as problems at this phase result in poor quality applications.

Suggestions:

- **Awareness:** Software engineers must educate themselves on general security concepts. Cases on previous security attacks should be presented to them in order to understand the need for proper protection of software (Sodiya *et al.*, 2006)
- Requirements specification review to find security errors by possibly using a checklist of requirement specification security errors (McGraw, 1998)
- Build abuse cases to describe system's behavior under attack (Hans, 2010)

Security in design phase: At this phase system must be designed with security and design should identify possible attacks.

Suggestions:

- Secure design guidelines and pattern should be followed while developing initial design (Khan and Zulkernine, 2009)
- The system should be divided into sub parts so that amount of damage is minimal that can be done to system when a unit is compromised (Chen, 2004)
- External review is often necessary to identify security errors (McGraw, 2002)
- Risk analysis should be undertaken before code is committed (Hans, 2010)

Security in coding phase: Secure programming language should be selected to minimize security errors. Coding standards and guidelines should be followed.

Suggestions:

- Use of unsafe functions should be avoided. Look for buffer overflow, array out of bound errors, integer underflow and overflow (Hans, 2010)
- All inputs and outputs must be validated. Don't relay on client side validation. Validations should be applied for type, length and range (Hans, 2010)
- Static and dynamic analysis tools can be used

Security in testing phase: In this phase, different tests are conducted to check security of developed software.

Suggestions:

- Risk-based security testing based on attack patterns and threat models (McGraw, 1998)
- Fuzz testing should be done that provides invalid, unexpected or random data to the inputs of the program (McGraw, 1998)
- All risks can emerge up during all phases of SDLC, so a constant risk analysis with continual risk tracking and monitoring activities is recommended (McGraw, 2006)

SECURITY RISK MANAGEMENT

Risk is defined as an exposure to chance of injury or loss (Kontio, 1994). That means risk implies that there is possibility that something negative may happen. In the context of software projects, negative implies that there is adverse affect on cost, quality and schedule (Jalote, 1997). But in case of secure software development, risk on security needs to be considered which means risk management in secure software must deal with risks on security too. Thus, in secure software, risks are broadly divided into four main categories:

- Security risk
- Cost risk
- Performance risk
- Schedule risk

Software security unnoticed during early phases of life cycle is inherited to later phase; i.e., one phase transfers its vulnerabilities to other phase (Daud, 2010). Risk Management is the area which tries to ensure that negative impact of risk is minimal. The focus of evaluation is risk perception of the project which reflects the chances that objectives of the project may not met. Risk management is basically to avoid disasters or heavy losses. Large number of software developed do not meet security standards as they lead to denial of service, non-functioning or many other undesired behaviors when their vulnerabilities get exposed by attackers. It is now suggested that many of these failures can be avoided if the security risks are properly identified and managed (Geontzel *et al.*, 2007). Risk management can be considered as dealing with the possibility and actual occurrence of those events that are not regular or commonly expected. It deals with events that are infrequent, somewhat out of control of the project management and are large enough to justify special attention (Kontio, 1994). Security is also a risk that is commonly ignored, so it needs a special attention too. Security risk management deals with identifying the undesirable events (attacks, misuse of data, security breaches etc.) that can occur, the probability of their occurring and the loss if such events occur. Once this is known, strategy can be formulated to avoid or reduce the probability of the risk. Therefore, like other risk management, security risk also revolves around risk assessment and risk control. Security management activities are presented in Fig. 1.

Security risk is the possibility that system is vulnerable to attack and its assessment must start from very beginning of the project i.e., project planning and should be done throughout the project as early identification provides better management and saves lot of time and cost. In this assessment activity, security risks must be identified, analyzed and priority must be assigned. Risks are project dependant and identification of the risks is necessary before any management can be done (Jalote, 1997). From the survey results and security properties, there is a list of top security risk items proposed by us. Table 1 shows the list of these items.

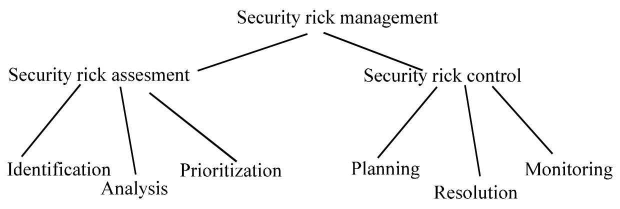

Fig. 1: Security management activities

Table 1: Security risk items

S. No.	Security risk items
1	Environment variables
2	Buffer overflows
3	Data as instructions or script injections
4	Nnmeric overflows
5	Race conditions
6	Network and information exposures
7	Operational misuse
8	Default settings
9	Programmer backdoors
10	Skilled security personnel

Items in software security checklist can be used to identify risks. This is just one approach likely to satisfy some projects. The other methods can be misuse cases, using security assessment tools. Now for assigning priority, one approach is through the concept of Risk Exposure (RE) (Bitz *et al.*, 2008). And security risk exposure SRE is defined as:

$$\text{SRE} = \text{Prob (Attack)*Loss (Attack)}$$

where, Prob (Attack) is the probability of occurrence of attack and Loss (Attack) is loss due to attack. Higher the value of SRE, the higher priority of security risk and also the total risk of security in project is the sum of all security exposures.

$$\text{Project security risk} = \Sigma \text{ Prob (Attack)*Loss (Attack)}$$

SECURITY AWARE SPIRAL (SAS)- A MODEL

Software is vulnerable to attack when some security lapses are overlooked during software life cycle. SaS considers security from the very beginning of the software life cycle. Security itself is a complete life cycle of software development (Daud, 2010). And, spiral model is an organized approach for developing software in which activities are organized like a spiral i.e., it has many cycles. Spiral model is based on risk perception for project (Jalote, 1997). An important feature of spiral model is that each cycle in spiral is completed by review that covers all products developed during that cycle, including plans for the next cycle. And in secure software development review is necessary to identify security errors (McGraw, 2002). The activities in secure spiral model are organized very much like traditional spiral that has many cycles. The model is shown in Fig. 2. Each cycle begins with identification of objectives, constraints, alternatives possible for achieving

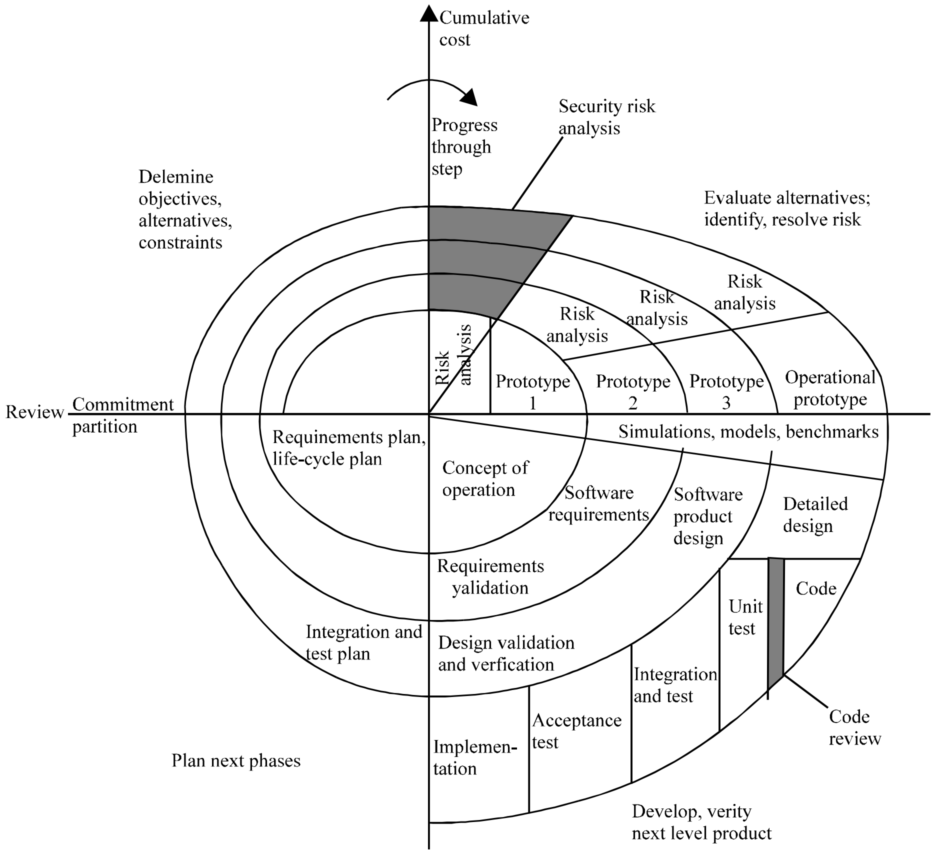

Fig. 2: Secure spiral model

the objectives for that cycle in its first quadrant (upper-left quadrant). In next step of the cycle, different alternatives are evaluated based on objectives and constraints.

Area shaded in grey in Fig. 2, shows the security risk analysis before beginning of each cycle in spiral. The focus of evaluation is risk perception of the project which reflects the chances that objectives of the project may not met. The list of risk items related to security is shown in Table 1. After identifying the risk, next step is to develop strategy to resolve the uncertainty and risks. This can be done with simulation, benchmarking, use cases, abuse cases and prototyping. Next, the software is developed with the implementation of suggestions for secure software development and keeping in mind the risks. And after the code is written, code must be reviewed by the programmer himself considering the risks. Code can also be reviewed by the external peer team so that holes/flaws can be removed from the code which could cause security breach.

CONCLUSIONS AND FUTURE WORK

It is true that security is the requirement of every organization. But outside attacks happen and succeed due to vulnerabilities inside. Different software engineering approaches are followed for the design and development of the software that includes iterative approach, agile methods but security is neglected and needs special consideration. Therefore, all approaches need security blend to make secure software development.

This paper explains a secure approach blend with spiral model. This can be used as guide for any security developer. Model explains can be further extended, whereas all activities of cycle can be modeled. Moreover, this model is yet to be implemented and results have to be analyzed.

REFERENCES

Ahmed, S.R., 2007. Secure software development-identification of security activities and their integration in software development life cycle. Master Thesis, Blekinge Institute of Technology.

Allen, J.H., S. Barnum and R.J. Ellison, 2008. Software Security Engineering: A Guide for Project Managers. Addison-Wesley Professional, Reading, MA, ISBN: 9780321509178, Pages: 334.

Bitz, G., J. Cochran, M. Coles, D. Dhillon and C. Fagan, 2008. Fundamental practices for secure software development: A guide to most effective secure practices today. Safe Code Software Forum for Excellence in Code, October 2008.

Chen, J., 2004. Security engineering for software. CS996-CISM. http://isis.poly.edu/courses/cs996-management/Lectures/SES.pdf

Daud, M.I., 2010. Secure software development model: A guide for secure software life cycle. Proceedings of the International Multi Conference of Engineers and Computer Scientists, March 17-19, 2010, Kowloon, Hong Kong.

Geontzel, K.M., T. Winograd, H. Mckin, L. Oh and M. Colon *et al.*, 2007. Software security assurance. A state-of-the Art Report (SOAR), Information Assurance Technology Analysis Center, Hemdon, VA. July, 2007, Data and Analysis Center for Software (DACS), http://iac.dtic.mil/iatac/download/securit

Gillam, D.P., T.L. Wolfe and J.S. Sherif, 2003. Software security checklist for the software life cycle. Proceedings of the 12th IEEE International Workshop on Enabling Technologies: Infrastructure for Colaborative Enterprises, June 9-11, 2003, Linz, Austria, pp: 243-248.

Hans, K., 2010. Cutting edge practices for secure software engineering. Int. J. Comput. Sci. Security, 4: 403-408.

Jalote, P., 1997. An Integrated Approach to Software Engineering. 2nd Edn., Springer, Berlin, Heidelberg, New York, ISBN: 9780387948997, Pages: 497.

Khan, M.U.A. and M. Zulkernine, 2009. A survey on requirements and designmethods for secure software development. Technical Report No. 2009-562, School of Computing, Queen's University, Kingston, Ontario, Canada, http://techreports.cs.queensu.ca/files/2009-562.pdf

Kontio, J., 1994. Software engineering risk management. A Technology Review Report, Nokia Research Center Report, Process Improvement Project deliverable PI_4.1, 1994. Helsinki, Finland.

McGraw, G., 1998. Testing for security during development: Why we Should scrap penetrate-and-Patch. IEEE Aerospace Electron. Syst. Magaz., 13: 13-15.

McGraw, G., 2002. Building secure software: Better than protecting bad software. IEEE Software, 19: 57-58.

McGraw, G., 2004. Managing software security risks. IEEE Security Privacy, 2: 80-83.

McGraw, G., 2006. Software Security: Building Security in. Addison-Wesley Professional, Reading, MA, ISBN: 9780321356703, Pages: 408.

Shirazi, H.M., 2009. A new model for secure software development. Int. J. Intellig. Inform. Technol. Appl., 2: 136-143.

Sodiya, A.S., S.A. Onashoga and O.B. Ajayi, 2006. Towards building secure software systems. Proceedings of the Issues in Informing Science and Information Technology, June 25-28, 2006, Salford, Greater Manchester.