
Hanif Mohaddes Deylami a,b, Iman Tabatabaei Ardekani a
Ravie Chandren Muniyandi b, Abdolhossein Sarrafzadeh a

a Department of Computing and Information Technology, Unitec Institute of Technology, Private Bag 92025, Victoria Street West, Auckland 1142, New Zealand.
b School of Computer Science, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Malaysia.

hmdeilami@gmail.com , iardekani@unitec.ac.nz
ravie@ukm.edu.my , hsarrafzadeh@unitec.ac.nz

Abstract

Security is a significant issue in the software development life cycle, and may become much more problematic in the future. In the current state there is no simple solution to the software security issue. Moreover, software developers have to be able to deal with huge numbers of defects in software. Security must be integrated into the development process from the beginning and continue through the design process to improve the security of the released software. This paper discusses software security challenges that one might face when developing enterprise software applications. As an essential concept in software engineering, software security is the process of protecting data and resources in order to achieve a more secure and reliable design and implementation. The main contribution of the paper is to present, in a coherent manner, major existing approaches and to emphasize description and method guidelines in particular. Moreover, this paper shows how to plan and conduct software development projects for creating secure and reliable products. Finally, it reviews the software security attribute requirements.

Keywords

1. Introduction

Today, web applications are very popular and essential tools for organizations and businesses because they can create business value by interconnecting a business with their customers and prospects. Businesses can be jeopardized, essential customers’ information and data could be critically affected by hackers who seek out loopholes in web applications. Traditionally, security testing is performed when the Software Development Life Cycle (SDLC) approaches to its final phases. In this situation, security testing can be considered as a barrier against the completion of the project in a timely manner, because fixing found-late bugs or issues usually requires unexpected resources. Particularly when the bugs do not cause any problem for the functionality of the software but creates a flaw in its security.

To solve this problem, secure SDLC initiatives can be adopted, where security deliverables are included in all SDLC phases. The positive results of the above actions are (1) Reduced security issues (2) Quicker defect rectification (3) Highlighting risks in first step of SDLC, and (4)
Reduced costs [1]. Generally hackers can penetrate systems by taking advantage of defects [2]. The Internet enables hackers to collect data and information through security holes in software. In fact there are always some security vulnerability in a piece of software. Moving towards any fundamental improvement cannot be achieved unless government, industry, academia, and communities work together for developing solutions to keep cyberspace safe and prosperous.

IT groups have been confronted with such problems and face serious obstacles when designing robust and foolproof systems. Academically computer science courses taught to train good programmers and application developers but lack the ability and tough subject to protect data. As a result, developers today are unaware of the different ways they can introduce security problems into their code. There is also currently a misalignment between stakeholders across the software development life cycle [3]:

*Misaligned Priorities:* IT developers are more focused into producing innovative products for the business needs. Vulnerabilities arising from defective codes are seen as critical issues and not on-time delivery and functionalities. Quality Assurance (QA) teams are involved with defective software and attempt to please customers. Security groups are more tuned to working towards protecting and producing critical data. They are also accountable for protecting localized and commercial software’s and to answer to hacking of codes after they are implemented.

*Misaligned Process:* Security audits and QA testing usually occur at the last phase of the development of the software and it is at this point that it is costly to fix problems when the company is more focused on releasing the product to the market and working on the next product. Therefore, teams leave the assessment plans to the tail end of the development process to preempt security teams from assessing the product for defects. Moreover, since security assessment is not part of the regular operation defects highlighted during the development are not likely to be addressed during testing due to time and cost constraints. Hence security problems highlighted late will tend to cause problems in the decision making to market and ensue security at the highest level.

*Misaligned Tools:* Developers’ mindset is such they do not want interference to their flow of work and therefore do not wish to use methodologies used by security personnel since these methodologies entail detailed procedures and contribute little by way of direct rectification of defects. It does not seem appropriate for developers to dabble with security testing. They will tend to oversee the results of such assessment since the unessential data will distract the actual solution to issues.

Software security have to be implement the best practices to enhance above-average technical methodologies and consider security assessment at the very beginning of the software life cycle, identifying and comprehending normal threat, designing for a secure lifetime, and engaging all software dimensions to in-depth, real time risk assessment and testing [2]. In addition, software security is about building secure software that means making software with a secure structure and ensuring all those involved in the development process are educated to consider security in their part of the process [2].

The rest of this paper is organized as follows; firstly, we explain the basic concepts of software security. Secondly, present guidelines for how to describe software security cases in detail by considering: Risk management framework, Touchpoints, and knowledge and method guidelines for eliciting security requirements. Thirdly, review a software security framework (SSF) practical use and discuss the strengths and weaknesses of software security, and compare them to related work. Finally, in the conclusion of this paper enumerate strategies for more research. The essential result of this research is to offer a clear, logical presentation, important current methodologies and to underline particular description and methodologies involved.
2. Software Security Fundamentals

The science of software security concerns constructing secure software. This entails providing the software with a structure and design that ensures it is secure. This also involves teaching software developers, constructors and end users the way to construct secure software [2]. Details are in the following discussion:

Three pillars of software security:
A. Risk Management Framework
B. Software Security Touchpoints
C. Software Security Knowledge

A. Risk Management Framework (RMF):
The management of risks is significant and a main key to achieving a secure framework to provide secure artifacts. In fact, it gives effective frame and structure to support secure software. In this part to make software security we should use technical methods and security standards (e.g. PCI Security Standards Council), to help governments, companies, and developers to determine what they should do about their framework. Figure 1 represents general steps needed to develop a risk management framework.

![Figure 1: Risk management framework](#)

Details are in the following discussion:
1. A business scenario wishes to convey an existing business issue and to provide the advantage and extent of the project [2].
2. The risks inherent in a business will provide the basis to comprehend the software risks in a business environment. Technical risks are a scenario that negates the design or implementation of a system under construction [2].
3. Questions such as ‘What needs to be done in the face of the current situation’ and ‘To reduce the risks what sort of resources should be planned’, should be addressed by creativity and prioritizing [2].
4. The business should dictate the direction to be taken to reduce the risks associated with the business in terms of finance, consolidation and comprehension [2].
5. To ensure that artifact quality is consistent and methodology should be put in place that can be consistently evaluated [2].
B. Software Security Touchpoints

Software security Touchpoints are based on good software engineering and involve explicitly considering security throughout the software life cycle. This means knowing and understanding common risks, designing for security and subjecting all software artifacts to thorough, objective risk analyses and test.

Figure 2: Security development life cycle [9]

The concept of Touchpoints are describing as a bellow:

1. Code Review: Using of Static Analysis tools such as: Fortify SCA, RATS, and ITS4 in order to monitoring source codes for finding regular malicious behaviour.
2. Architectural Risk Analysis: It’s required in design stage (class hierarchy) and also in the characteristic architecture stage.
3. Penetration Testing: Testing the internal structure of software in real environment by using some useful methods like: White-box testing, or Black-box testing in order to explicit knowledge of software structure.
4. Risk Based Security Testing: Use architectural risk analysis results to drive scenario based testing.
5. Abuse Cases: It’s highlight the systems behaviours under stress. Using some useful tools in order to simulate the situation for developer like working as a hacker.
6. Security Requirements: In this requirement security must be comprehensively covered. These should encompass functional security (e.g. cryptography) and recently adopted scenario (to detect abuse and hacker patterns).
7. Security Operations: Experience accumulated in studying hackers and data obtained should be used in software design and development.

Figure 2 specifies one set of the best practices and represents how software practitioners can apply them to the various software artifacts produced during software development. That means software security best practices applied to various software artifacts. Although the artifacts are laid out according to a traditional waterfall model in this illustration, most organizations follow an iterative approach today, which means that best practices will be cycled through more than once as the software evolves [4].

C. Software Security Knowledge

Software security knowledge can apply in the various stages throughout the entire software development life cycle (Figure 3). An effective method is to implement software security ‘best practices’ via the following:

1. Prescriptive Knowledge
   a. Principles: High-level architectural principles, e.g. the principle of least privilege.
b. Guidelines: Mid-level guidelines e.g. make all Java objects and classes final unless prevented.

c. Rules: Tactical code-level rules, e.g. avoid the use of the library function `gets()` in C.

2. Diagnostic Knowledge

a. Vulnerabilities: Descriptions of software vulnerabilities experienced and reported in real systems.
b. Exploits: Descriptions how instances of vulnerabilities are used to compromise the security of particular systems.
c. Attack patterns: Descriptions of common sets of exploits in a more abstract form that can be applied across multiple systems.

3. Historical Knowledge

Historical Risks: Data collected and analyzed from actual software development must reflect its influence on the business. This database assists in influencing similar problems in later software development without ‘re-inventing the wheel’ [6].

![Figure 3: Software security unified knowledge architecture][9]

3. Software Security Framework (SSF)

Nowadays software developers and other IT personnel realize that an understanding of software security and issues is important. There are many aspects to software security. The SSF permits an individual to hold a discussion without going into too many details. Table 1 shows the SSF 12 practices, which categorized into 4 domains.

<table>
<thead>
<tr>
<th>Governance</th>
<th>Intelligence</th>
<th>SDL Touchpoints</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategy and Metrics</td>
<td>Attack Models</td>
<td>Architecture Analysis</td>
<td>Penetration Testing</td>
</tr>
<tr>
<td>Compliance and Policy</td>
<td>Security Features and Design</td>
<td>Code Review</td>
<td>Software Environment</td>
</tr>
<tr>
<td>Training</td>
<td>Standards and Requirements</td>
<td>Security Testing</td>
<td>Configuration and Vulnerability Management</td>
</tr>
</tbody>
</table>

Table 1: A software security framework [2]
The more details of the domains are elaborated as a below:

1. **Governance**: Exercises that assist in organizing managers and evaluating a software security proposal. This includes staff skills enhancement [2].

2. **Intelligence**: Exercises that help to collate corporate information used in implementing security initiatives all over the organization. Collation should include spontaneous security action and organizational threat modeling [2].

3. **SDL Touchpoints**: Initiatives aligned to assessment and assurance of specific software development artifacts and procedures. All software security procedures are inclusive of these procedures [2].

4. **Deployment**: Exercises that collaborate with previous generation security and software monitoring departments. Software identification, monitoring and other external issues have a direct influence on software security [2].

### 4. Return on Security Investment (ROSI)

Nowadays the security issue comprises a major part of the software life cycle from network security, application security, and system security in order to develop secure software [6]. Organizations and companies still spend budget and more time to analyze and measure the effectiveness of those methods they are using to provide secure software. That’s why they need to measure the budget usage and also provide supportive strong reasons to claim budget for next step of the software development cycle in order to justify their cost effectiveness and cost of their information security issue.

In an end-to-end software life cycle, security issue is not an investment to provide profit. To comply with security needs, policies, training developers and security procedures require big budgets. Two questions confronting management are “What is most cost effective method to safeguard information?” and “What system is used to assess return on security investment?”. Organizations and its managers need to plan and also bring a sufficient justification for cost and their requirement for software security methodologies to deal with software security problems.

People who purchase security packages for their software will contemplate on how much benefits they will obtain on their investment (ROSI). In the past ROSI computations were complex due to the complexity of the method. Scientists and researchers have attempted to wade through several requirements to compute ROSI and eventually decided to categorize security management outcomes into 3 parts each part designed to provide a certain increase towards ROSI:

- **Effective Security**
  The paramount requirement of a set of software security solutions is its effectiveness in use. Combining the various advantages into a compact and complete software programming would be the best defense mechanism against malwares. But to prove that these security effectiveness methods work is not easy. When faced with the dilemma as to whether an attack did not take place or the installed software actually blocked the invader, to prove its effectiveness. Since this response id negative it is indeed a difficult task for management to prove that a software security solution is required and worthy for investing by organization in order to achieve to the high level of security.

- **Risk Reduction**
  The Risk reduction is another advantage that is provided by software security initiatives. In order that business activities should proceed uninterrupted management should install state of the art security systems. In the absence of effective software security system businesses could face downtimes and short-term monetary disadvantages generally affecting the profitability.

- **Efficiency**
  Software security solutions are now looked upon as an integral part of any organization dealing with clients’ data to reach high level of security. It is no more an add-on or an afterthought. It is
designed into the company’s operating portfolio. The source code and architecture ensure that the system performance is not compromised. Efficiency is a requirement for organizations that are in an above average production environment such as algorithmic or transaction processing where performance and scalability are critical. If source code efficiency and scalability were analyzed, it would appear that hidden threats and the potential degradation in response time could affect the customer’s satisfaction.

The amount of funding required by an organization for security is dictated by ROSI, which also affords cost effective options. Moreover, it provides an essential data for the managers’ decision making in the areas of “excessive security expenditure”, “The monetary impact on productivity, due to an absence of security”, “adequacy of security”, and “is the current security portfolio providing the required benefits?” [19].

The risk of investment is evaluating by combination of cost of implementation of security and quantitative risk assessment, which called ROSI. It is defined as below [19]:

\[
\text{ROSI} = \frac{\text{Monetary loss reduction} - \text{Cost of the solution}}{\text{Cost of the solution}}
\]

The negative financial variance can be minimized due to ALE. This is an annual negative financial variance that can be anticipated from a particular risk on a particular asset, it is defined as, Annual Loss Expectancy (ALE) = Annual Rate of Occurrence (ARO) * Single Loss Expectancy (SLE). Therefore, ROSI is defined as the difference of the ALE without the installation of the security solution. This is compared to the modified ALE (mALE) required for security solutions [19].

\[
\text{ROSI} = \frac{\text{ALE} - \text{mALE} - \text{Cost of the solution}}{\text{Cost of the solution}}
\]

It is also equals to the mitigation ratio of the solution applied to the ALE:

\[
\text{ROSI} = \frac{\text{ALE} \times \text{mitigation ratio} - \text{Cost of the solution}}{\text{Cost of the solution}}
\]

It is also simplified as a below:

\[
\text{ROSI} = \frac{[(\text{ALE} \times \% \text{ Risk Mitigation}) - \text{Scost}]}{\text{Scost}}
\]

Table 2: Different type of result of ROSI formula

<table>
<thead>
<tr>
<th><strong>Negative</strong></th>
<th>Investment not justifiable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Null</strong></td>
<td>No return on investment</td>
</tr>
<tr>
<td><strong>Positive</strong></td>
<td>Justifiable as compared with other solutions</td>
</tr>
</tbody>
</table>

If we look at Table 2 the result of the ROSI for a virus scanner one can estimate how the equation functions. G Data Anti virus has previously been infected. It is computed that the average loss due to damage in productivity loss was $25,000. Currently, G Data gets four of these viruses per year. G Data expects to catch at least three of the four viruses per year by implementing a $25,000 virus scanner.
Risk Exposure: $25,000, 4x per year = $100,000
Risk Mitigated: 75%
Solution Cost: $25,000

\[
\text{ROSI} = \frac{[($100,000 \times 75\%) - $25,000]}{$25,000} = 200\%
\]

The virus scanner seems to have paid for itself. Since we assume that the cost of the debacle is $25,000 and that the scanner will trap 75% of the intruders and also the actual cost of the scanner is $25000. In actual fact it is highly improbable that these numbers are close to actual. What would the scenario be if three of the four viruses cost $5000 in damages and the last one amounted to $85000? The average cost is still $25,000. Which one of those four viruses is going to get past the scanner? If it is a $5,000 one, the ROSI increases to nearly 300% - but if it is the expensive one, the ROSI becomes negative.

Generally, sensible numbers for the ROSI equation is quite difficult to say the least. Currently there is no acceptable model to estimate the monetary risk connected with security incidents. Concurrently there is an absence of a program or model that can evaluate the risk reducing power of a security solution. Estimating the cost of solution could be grossly divergent. In some instances all factors are included in the computation whereas in others some items are excluded such as overheads, software, hardware, long-term influence on productivity.

5. Discussion

This research focused on assisting organizations to well-defined software security process by considering RMF, Touchpoints, software security knowledge, SSF, and ROSI benefits. Our main goal was to how integrate software security steps as a part of the software development and maintenance life cycles, for the purposes of assuring the security performance of the system.

Based on our research, we are facing the variety of security requirements at the practical level in the higher level of abstraction. It is usually easy to create some high level objectives in all possible situations; one can even set an objective for a system to be secure in its entirety. In practice IT security and its objectives could be enumerated from the point of view of setting up an acceptable level of protection. Therefore, it is quite obvious that all aspects of computing the cost of protection such as usability, cost and security can rarely be inclusive.

While researching for this paper it is to be taken into account that some aspects can be included in the systems while others may not be precisely differentiated from the security information data outside the system purview and environment. Generally, it is more difficult to specifify objective to a system, which already exists than to determine them in the early stage of the software development. In practice though, when the system is already in use one tends to have expectations towards it, what it can do and it does not need to do, based on its past operations. At that case it is hard to stay neutral and define objectives without thinking beforehand if the system is able to meet them or not. We also noted that one can not accurately specify the information security requirement specification and the practical security vulnerabilities. Security assurance process should result in more realistic and concrete requirements than is the case at the moment.

6. Conclusions and Future Work

Today, software companies are faced with the rapid changes of security issues in order to produce secure software. By considering security throughout the software development life cycle and also through education and use of best practice software project managers and developers can develop more secure software. Software security requires an emphasis on
security issue during software life cycle such as requirement gathering, risk analysis, design and development, penetration testing and integration (operation and maintenance), and finally development.

This study is preliminary due to lack of extensive evaluation. As future work we would like to implement a framework to support our research and perform extensive evaluation to verify the usefulness of our framework.

Acknowledgments
This research is outcome of a research collaboration opportunity funded by Unitec’s Department of Computing, and also Research and Enterprise Office of Unitec Institute of Technology.

References
[15]. “Introduction to secure software development life cycle”, (2013), February 01.
URL: http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle/
[16]. “Securing the software development life cycle”, (2015), posted by Ispitzer, April 05.
URL: http://www.securingthehuman.org/blog/2015/04/05/securing-the-software-development-lifecycle#
URL: http://software-security.sans.org/blog/2012/07/09/what-appsec-can-learn-from-devops
URL: http://software-security.sans.org/blog/2012/02/22/agile-development-teams-can-build-secure-software