Chapter 22
Software Security Engineering – Part I:
Security Requirements and Risk Analysis

Issa Traore
University of Victoria, Canada

Isaac Woungang
Ryerson University, Canada

ABSTRACT

It has been reported in the literature that about twenty new software vulnerabilities are reported weekly. This situation has increased the security awareness in the software community. Nowadays, software services are expected not only to satisfy functional requirements but also to resist malicious attacks. As demand for more trustworthy systems is increasing, the software industry is adjusting itself to security standards and practices by increasing security assessment and testing effort. Even though there is a consensus that better software engineering is to improve software quality in the early stage of software development, so far, various approaches that have been proposed to analyze and quantitatively measure the software security target, primarily show the finished software products in their operational life. There are few achievements on how to reduce or effectively mitigate the security risks faced by software products during the development process. In this chapter, the authors introduce a novel model-driven perspective on secure software engineering, which integrates seamlessly software security analysis with traditional software development activities. A systematic security engineering process that starts in the early stages of the software development process and spans the entire software lifecycle is presented. Fundamental software security concepts and analysis techniques are also introduced, and several illustrative examples are presented, with focus on security requirements and risk analysis.

INTRODUCTION

Over the last decades, software quality attributes such as maintainability, reliability, and performance have been widely studied. In contrast, less attention has been paid to the field of software security, sometime due to the complex and multifaceted nature of the notion of security or for economical reasons such as the need to shorten "time to market." As of today, the study of software security still remains immature. Current approaches to software security engineering

DOI: 10.4018/978-1-4666-8111-8.ch022

referred to as "*penetrate and patch*" consist mostly of fixing security flaws after they have been exploited. "*Penetrate and patch*" is a fictitious solution which deals only with the symptoms and not the deep causes of the problem. This can be worrisome because software applications are often deployed in malicious environments in which security attacks and intrusions happen all the time. The computer security technology business is a fast growing sector, with increasing marketing opportunities. The CSI/FBI annual survey is an insightful reference of how often computer crimes occur and how expensive they can be. In their survey for the year 2003, it was reported that the total annual financial losses were $201,797,340 (Richardson, 2003).

This observation could actually be worse since only 251 out of the 530 participants (47%) reported their losses. The survey also shows other compelling statistics: 92% of the respondents detected attacks during the last 12 months while 75% of the respondents acknowledged financial losses due to security breaches. As mentioned above, only 47% reported their losses though. Therefore, the question is: how do organizations cope with such attacks?

Many organizations address security from three different perspectives: prevention, detection, and reaction. According to the 2003 CSI/FBI survey (Richardson, 2003), 99% of the respondents use a mixture of various technologies pertaining to those perspectives. For example, more than 90% use prevention technologies such as firewall, access control, and physical security.

Firewall technology has been used so far to protect and isolate segments of networks against untrusted networks, by filtering out harmful traffic. However, there are several limitations on firewall technologies, which make them insufficient to achieve strong network protection. There are several widely publicized exploits that allow hackers to access sensitive data by tunneling through authorized protocols. In order to provide a stronger level of security, most organizations combine firewalls with a range of security monitoring tools named intrusion detection systems (IDS); 73% of the company surveyed by CSI/FBI use intrusion detection systems. The role of IDS is to monitor and detect computer and network intrusions, in order to take appropriate measures that would prevent or avoid the consequences. Intrusion detection systems, however, are severely limited by their ineffectiveness in detecting new forms of attacks. This is unfortunate, since the Internet is a wild zone, where new forms of security attacks are developed and executed daily.

A large number of these attacks succeed because of the existence of software flaws such as buffer overflow, Trojan horse, or race conditions. These flaws may be directly related to the security mechanisms themselves, which in a large number of cases are implemented as software (e.g. firewall, IDS, encryption scheme etc.). They may also be related to other software applications whose primary purpose is not security related.

As a matter of fact, it is commonly agreed (Mc Graw, 1998) that software carries the biggest security challenge of today's systems: about 20 new software vulnerabilities are reported weekly. Unfortunately software security engineering is still in its infancy. Current approaches to software security engineering referred to as "*penetrate and patch*" (Mc Graw, 1998) consists mostly of fixing security flaws after they have been exploited. "*Penetrate and patch*" is a fictitious solution which deals only with the symptoms and not the deep causes of the problem.

A more effective approach to software security consists of treating it from a quality assurance perspective and integrating security concerns in the entire software life cycle from the requirements definition to the production stage. Following a well-defined process allows us to clearly define the actual security issues faced by the system, to select and implement appropriate protection mechanisms, and to make sure that these mechanisms work when they are needed. Our focus in this Chapter is on security requirements and risk