Chapter 23
Security Testing Framework for Web Applications

Layla Mohammed Alrawais
Prince Sultan University, Riyadh, Saudi Arabia

Mamdouh Alenezi
https://orcid.org/0000-0001-6852-1206
Prince Sultan University, Riyadh, Saudi Arabia

Mohammad Akour
Yarmouk University, Irbid, Jordan

ABSTRACT

The growth of web-based applications has increased tremendously from last two decades. While these applications bring huge benefits to society, yet they suffer from various security threats. Although there exist various techniques to ensure the security of web applications, still a large number of applications suffer from a wide variety of attacks and result in financial loses. In this article, a security-testing framework for web applications is proposed with an argument that security of an application should be tested at every stage of software development life cycle (SDLC). Security testing is initiated from the requirement engineering phase using a keyword-analysis phase. The output of the first phase serves as input to the next phase. Different case study applications indicate that the framework assists in early detection of security threats and applying appropriate security measures. The results obtained from the implementation of the proposed framework demonstrated a high detection ratio with a less false-positive rate.

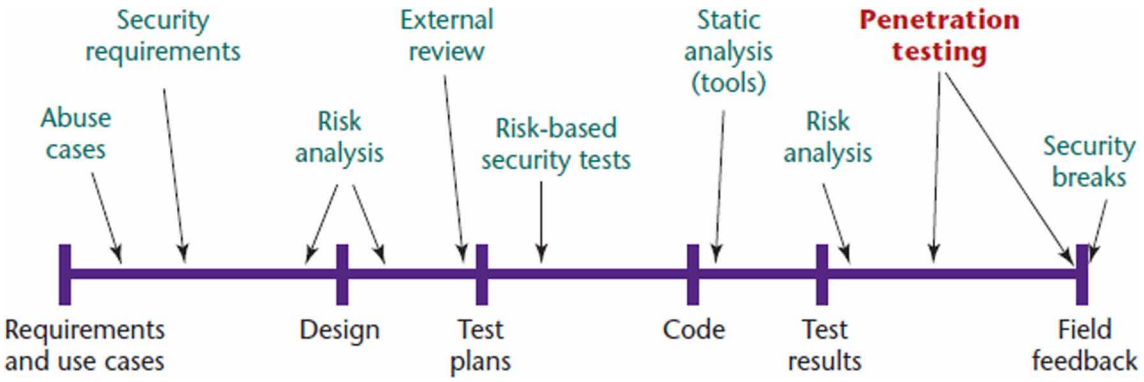
1. INTRODUCTION

The tremendous increase in the development of software applications naturally leads to thinking about the security aspects associated with these applications. To ensure security, testing is an important mechanism both to identify defects and assure that the software is working as expected (Da Mota Silveira Neto, Do Carmo MacHado, McGregor, De Almeida, & De Lemos Meira, 2011). Software testing is an important

DOI: 10.4018/978-1-6684-3702-5.ch023

and costly activity in the software development life cycle. Furthermore, inadequate software testing usually leads to major risks and consequences (Garousi & Zhi, 2013). The software testing activity continues throughout the software development life cycle (SDLC) unlike the thought that the activity of testing is performed at the end of software development. Similar to SDLC the Security Testing Life Cycle (STLC) is shown in Figure 1:

Figure 1. Security throughout the SDLC (PCI Security Standards Council, 2015)



The software from the organizations often suffers from systematic faults at different levels of SDLC. The reason is the failure of following standard security practices (PCI Security Standards Council, 2015) throughout the life cycle of software. Acceptance testing and penetration testing are considered too late in the identification of bugs and at this stage, it may be possible that time and budget constraints do not allow fixing things.

Software engineering faces several challenges from several domains in order to prevent malicious attacks and adopt security measures. Testing is a widespread validation approach in the industry. Meanwhile, this approach is expensive, ad-hoc and unpredictable in effectiveness. Indeed, software testing is a broad term encompassing a variety of activities along the development cycle and beyond, aimed at different goals (Bertolino, Bertolino, & Faedo, 2007). Hence, a lot of challenges are faced by software testing (Gao, Bai, & Tsai, 2015; Harman, Jia, & Zhang, 2015). A consistent roadmap of the most relevant challenges to be addressed is required to be proposed.

Security testing is usually considered to be done at the end of software development and working software is tested using penetration testing. One major limitation of this approach is considering software testing at the end of software development activity which could be too late to tack a problem if exists (Arkin, Stender, & McGraw, 2005).

Acceptance testing and penetration testing are thought to be effective in discovering bugs/errors and issues. The issue with both of these testing types is that (1) penetration testing covers only certain bugs and a certain segment of functionality and (2) late discovery of the bugs and defects may leave the software in a state where fixing these may be prohibitively expensive in terms of time and cost.

To improve the security of software application, security models in the software development life cycle (Howard & Lipner, 2006; McGraw, 2006; Chandra, n.d.) are proposed in decades. In this research,