Chapter XVI
Software Specification and Attack Languages

Mohammed Hussein
Queen's University, Canada

Mohammad Raihan
Queen's University, Canada

Mohammad Zulkernine
Queen's University, Canada

ABSTRACT

General-purpose software specification languages are introduced to model software by providing a better understanding of their characteristics. Nevertheless, these languages may fail to model some nonfunctional requirements such as security and safety. The necessity for simplifying the specification of nonfunctional requirements led to the development of domain-specific languages (e.g., attack description languages). Attack languages are employed to specify intrusion detection related aspects like intrusion signatures, normal behavior, alert correlation, and so forth. They provide language constructs and libraries that simplify the specification of the aforementioned intrusion detection aspects. Attack languages are used heavily due to the rapid growth of computer intrusions. The current trend in software development is to develop the core functionalities of the software based on the requirements expressed in general-purpose software specification languages. Then, attack languages and other security mechanisms are used to deal with security requirements. However, using two sets of languages may result in several disadvantages such as redundant and conflicting requirements (e.g., usability vs. security). Moreover, incorporating security at the latter stages of a software life cycle is more difficult and time consuming. Many research works propose the unification and reconciliation of software engineering and security engineering in various directions. These research efforts aim to enable developers to use the current software engineering tools and techniques to specify security requirements. In this chapter, we present a study on the classification of software specification languages and discuss the current state of the art *regarding attack languages. Specification languages are categorized based on their features and their main purposes. A detailed comparison among attack languages is provided. We show the example extensions of the two software specification languages to include some features of the attack languages. We believe that extending certain types of software specification languages to express security aspects like attack descriptions is a major step towards unifying software and security engineering.*

INTRODUCTION

Software requirements specification is an intermediate step between requirements elicitation and implementation. Software specification languages (SSL) are used to model software systems to gain better understanding of the software. The outcome of such modeling is a design which can be verified against the user requirements. Specifications can also help in code and test case generation. Several studies exist in the literature that compare a number of SSL based on different properties (Clements, 1996; Ostroff, 1992; Jin & Nahrstedt, 2004; Wieringa, 1998). Although SSL are powerful, developers face difficulties when using them to model nonfunctional requirements. Complex models, lack of language constructs, and libraries are examples of such difficulties. These difficulties lead to the presence of domain-specific languages (e.g., attack languages). Attack languages are employed to specify intrusion detection related aspects like intrusion signature, normal behavior, and alert correlation.

Software play a key role in every strata of our life. For example, they are being used in financial institutions, government agencies, health sectors, and power-control systems to store and process security critical information. As the world is becoming more dependent on such software systems, the need for developing secure software is becoming more evident. Computer attacks or intrusions are increasing since the knowledge required for launching them is becoming more available. According to Computer Emergency Response Team (CERT) statistics (CERT, 2005), the number of incidents reported to CERT in 2001 was 52,658, while in 2002, the number was 82,094. In 2003, 137,529 incidents were reported. which is more than the number in both 2001 and 2002 together. The current trend in the software industry is to wait till the main functionalities of the software are implemented, and then to add security aspects by using special languages for security.

However, using two sets of languages may cause several disadvantages. First, designing the system without considering security at the early stages of the software development life cycle (SDLC) increases the number of vulnerabilities that must be dealt with in the latter stages. Second, incorporating security aspects within an existing design leads to redundant and conflicting development efforts. Third, delaying security issues imposes higher development cost and produces less maintainable software.

Unifying software and security engineering, on the other hand, is a promising solution to the above mentioned problems. The unified engineering approach is called Software Security Engineering (Zulkernine & Ahamed, 2005). It provides developers with a more concrete view of security requirements which helps in avoiding conflicting design decisions due to the requirements. Moreover, the unification enables the developers who are not expert in security engineering to build secure software. Software security engineering also helps in anticipating the cost of developing security aspects of the system at the early stages of the SDLC. Many research directions have been explored for the purpose of unification such as new frameworks for software