Chapter 1

Graphical Modeling of Security Goals and Software Vulnerabilities

David Byers
Linköping University, Sweden

Nahid Shahmehri
Linköping University, Sweden

ABSTRACT

Security has become recognized as a critical aspect of software development, leading to the development of various security-enhancing techniques, many of which use some kind of custom modeling language. Models in different languages cannot readily be related to each other, which is an obstacle to using several techniques together. The sheer number of languages is, in itself, also an obstacle to adoption by developers. The authors have developed a modeling language that can be used in place of four existing modeling languages: attack trees, vulnerability cause graphs, security activity graphs, and security goal indicator trees. Models in the new language can be transformed to and from the earlier language, and a precise definition of model semantics enables an even wider range of applications, such as testing and static analysis. This chapter explores this new language.

INTRODUCTION

Modern society has rapidly become dependent on computers, and by extension dependent on computer software. As a result, the impact of software failure can be tremendous. Over the years we have seen software failures with consequences ranging from the amusingly absurd (Grisogono, 1999)1, to the terrifyingly lethal (Schmitt, 1991)2. While most software failures are caused by flaws in the software being triggered unintentionally, some failures are the result of vulnerabilities being intentionally exploited. Over the last several decades the economic impact of the most publicized IT security incidents has been estimated to be tens of billions of dollars, worldwide (Hoy et al., 1989; Rhodes, 2001; Computer Economics, 2002, 2003). This does not include costs incurred

DOI: 10.4018/978-1-4666-6359-6.ch001
by individuals or institutions due to e.g. identity theft or lost business. The total cost of cybercrime and cyber espionage was recently estimated at $100 billion per year (Center for Strategic and International Studies, 2013). Security problems are not limited to mainstream computing. Serious vulnerabilities have been reported in industrial control systems, healthcare systems, automotive systems, and other kinds of embedded systems. The impact of vulnerabilities coupled with their prevalence in all kinds of software, clearly demonstrates that software security is a critical issue.

Our work concentrates on improving the ability of developers using conventional methods to address typical software security issues. Typical software security issues include the prevention of known vulnerabilities and the identification and fulfillment of common security goals. Known vulnerabilities account for nearly all publicly reported vulnerabilities, and failure to implement common security goals for nearly all design flaws we have observed.

We have developed a process improvement methodology, called S3P, that is based on detailed analysis of vulnerability causes (Byers, Ardi, Shahmehri, & Duma, 2006; Ardi, Byers, & Shahmehri 2006; Byers & Shahmehri, 2007, 2008, 2009; Byers, 2013). The S3P uses models to describe both vulnerability causes and mitigating activities. This work is complemented by the SHIELDS EU project (SHIELDS, n.d.), which developed a shared repository for security models, and tied together multiple model-based activities for secure software development.

In this chapter we present a graphical modeling language, the security goal model (SGM) language, that can be used in place of attack trees, security activity graphs (SAG), vulnerability cause graphs (VCG), and security goal indicator trees (SGIT). Table 1 summarizes these languages. An SGM shows how a given security goal can be fulfilled, and can be used for purposes as diverse as process improvement, automatic testing, static analysis, and manual inspection. Models in the traditional languages can be transformed to SGMs, and SGMs can be viewed using any of the traditional notations. This means that developers familiar with the older notations need not learn the SGM language unless they need the improvements the new language provides (Byers & Shahmehri, 2010).

The SGM language offers several benefits over using the older languages. A minor benefit is that a single language is easier to learn, use, and build tool support for than it is for multiple languages. More importantly, models in the four languages we cover are often closely related and may contain similar or identical elements. When using conventional languages, information cannot be shared or reused between models of different types. Using SGMs, models that would previously have used different languages can share elements,

### Table 1. Graphical modeling languages covered by SGMs

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack trees</td>
<td>Attack trees are used to model how to perform attacks. In an attack tree, the root is a successful attack, and other vertices are sub-attacks. Sub-attacks may be combined with <em>and</em> and <em>or</em>. Attack trees are used in risk analysis.</td>
</tr>
<tr>
<td>Vulnerability cause graphs</td>
<td>Vulnerability cause graphs (VCGs) model how vulnerabilities are caused. The original purpose of VCGs was software process improvement. The SGM language is a direct successor of the VCG language.</td>
</tr>
<tr>
<td>Security activity graphs</td>
<td>Security activity graphs (SAGs) model how to perform security-related activities. SAGs were designed to be used in conjunction with VCGs to help developers find the best way to prevent vulnerabilities.</td>
</tr>
<tr>
<td>Security goal indicator trees</td>
<td>Security goal indicator trees (SGITs) model how to perform goal-directed manual inspection of software development artifacts.</td>
</tr>
</tbody>
</table>