Chapter XXXVI
Threat Modeling and Secure Software Engineering Process

Wm. Arthur Conklin
University of Houston, USA

ABSTRACT

Software defects lead to security vulnerabilities, which cost businesses millions of dollars each year and threaten the security of both individuals and the nation. Changes to the software engineering process can help to reduce the number of defects, improving the quality of the process. This chapter introduces the concept of threat modeling to include security in the process of developing software. Adding threat modeling to the software development process will improve the quality of the process. The majority of software coding errors are preventable using a process designed to avoid a series of common errors. Increasing the visibility of common errors will enable software engineers to produce code with substantially fewer security errors. Threat modeling provides insight into the risks facing the software at design time, increasing the software engineering team’s opportunity to avoid errors during coding.

INTRODUCTION

Software engineering is a relatively new discipline and has grown significantly in the past 30 years. The development of software is a complex task involving many parties working to build a software solution to a particular task. As the level of complexity of the software projects has increased, so has the importance of using a process based model to manage the complexity. With the rise of the Internet, with its inherent connectedness, and the increase in e-commerce, a new threat has emerged, that of e-crime. E-crime is a form of fraud that uses the interconnected nature of the Internet to attack victims. Attacks on software began with simple worms and viruses, and have now progressed to complicated attacks against Web applications.

Attacks on computer systems are a significant issue for enterprises with Web-facing applications (CSI 2006). The increase in business desire to develop Web-based, customer driven applications has opened the door to even greater exposure to threats and risk. The majority of the security efforts in an enterprise are focused on defending the enterprise against attacks. Software flaws, coupled with the test and patch doctrine, act as a financial burden to both software users and developers. Addressing software flaws in a preventative fashion, before they place the enterprise at risk is highly desired by developers and customers. Preventing and removing vulnerabilities from internally generated applications will significantly...
Threat Modeling and Secure Software Engineering Process

strengthen the security posture of an organization and reduce operational costs. Just as the use of software development lifecycle models has reduced the complexity of development, introduction of security into the development lifecycle process will provide a solution to the complexities of securing code.

A very powerful tool that can be used to inject security into any software development lifecycle model is threat modeling. Threat modeling begins acting during the design phase and interacts through testing, helping to build quality into the software. Threat modeling is process agnostic, it can be added to any existing software development process. Threat modeling can act as a comprehensive security communication plan throughout the software development process.

Expertise to achieve the development of a comprehensive threat model and keep the development team apprised of the changing security landscape has led to the need for security experts on the development team. Just as manufacturing needed quality experts to assist in the focus of quality in manufacturing, software production has the same need. In fact, security can be seen to be a lot like quality in manufacturing. Quality in manufacturing has been defined as conformance to requirements, and security can be viewed in the same light.

BACKGROUND

Whether we listen to the news, read the newspaper or online news feeds, or monitor reports from the US CERT, we learn of cyber vulnerabilities and exploitations. These exploits and vulnerabilities are occurring at an alarming rate, costing businesses millions of dollars each year and threatening the security of the nation and of individuals (CSI 2006). Most cyber vulnerabilities can be traced to defects in software that are caused by bad design and poor development practices (Howard & LeBlanc, 2002; Howard, LeBlanc, & Viega, 2005). Defective software is largely the result of poor software engineering techniques that often “add on” security considerations rather than include them from the beginning. Yet the efficacy of bolt-on, after the fact security measures has been called into question (Conklin & Dietrich, 2005).

The Software Engineering Institute (SEI) at Carnegie Mellon University has analyzed programs written by thousands of programmers and concluded that even experienced professionals unintentionally make errors in design and programming. These errors result occur in the determination of requirements, designing, developing, and testing the software (Davis & Mullaney, 2003). Errors occur in new code and in code corrections, with one defect for every seven to 10 lines of new or changed code being a normal rate. In another benchmark study, analyzing hundreds of software projects, Jones found that released software contains from one to seven defects per thousand lines of new or changed code (Jones, 2000). Regardless of the actual error rate, these errors manifest themselves in the form of vulnerabilities and the need for costly patches.

The cause of errors and defects are known issues. According to a study done by the SEI, over 90 percent of security vulnerabilities are the result of known software defect patterns. This study demonstrated that 10 common defects accounted for about 75 percent of all security vulnerabilities (Davis & Mullaney, 2003). Another study of 45 e-business applications showed that 70 percent of security defects were caused by poor software design (Jacquith, 2002). For the most part, software design and development errors that lead to security breaches include a basic lack of security knowledge, declaration errors, failure to validate input, buffer overflows, and logic errors. The director of research at the SANS Institute observed that all of the conditions listed on the SANS Institute Top 20 Internet Security vulnerabilities are a result of poor coding, testing, and bad software engineering practices (SANS Institute, 2005). All of these defects are preventable. In fact, books have been written about the common causes (Howard et al., 2005). Previous academic research into the subject has examined causes, but has not produced significant reductions in flaws (Baskerville 1993; Wang & Wang 2003).

The cause of the problem has been identified as coming from multiple sources: education of programmers, business practices of software development firms, and application deployment practices of end-users (Bishop & Engle, 2006). This chapter tackles the concept of adding security principles to a software development process, resulting in a secure lifecycle development process (Howard & Lipner, 2006). This chapter proposes an actionable methodology to reduce the number of flaws in software.

Security in software development is a term with different meaning to different people. This chapter examines the concept of applying security related changes to the development process with the overall goal of affecting the output quality. Software quality can be measured using bug counts and normalized in
Related Content

A Game Theoretic Approach to Optimize Identity Exposure in Pervasive Computing Environments
[www.igi-global.com/article/game-theoretic-approach-optimize-identity/50494?camid=4v1a](www.igi-global.com/article/game-theoretic-approach-optimize-identity/50494?camid=4v1a)

Electronic Medical Records, HIPAA, and Patient Privacy
[www.igi-global.com/article/electronic-medical-records-hipaa-patient/2486?camid=4v1a](www.igi-global.com/article/electronic-medical-records-hipaa-patient/2486?camid=4v1a)

Investigation of Credit Risk based on Indian Firm Performance

Short Online/Off-line Signature Scheme for Wireless Sensor Networks