A Tagging Approach to Extract Security Requirements in Non-Traditional Software Development Processes

Annette Tetmeyer, Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, USA

Daniel Hein, Automotive OEM, Garmin International, Olathe, KS, USA

Hossein Saiedian, Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, USA

ABSTRACT

While software security has become an expectation, stakeholders often have difficulty expressing such expectations. Elaborate (and expensive) frameworks to identify, analyze, validate and incorporate security requirements for large software systems (and organizations) have been proposed, however, small organizations working within short development lifecycles and minimal resources cannot justify such frameworks and often need a light and practical approach to security requirements engineering that can be easily integrated into their existing development processes. This work presents an approach for eliciting, analyzing, prioritizing and developing security requirements which can be integrated into existing software development lifecycles for small organizations. The approach is based on identifying candidate security goals using part of speech (POS) tagging, categorizing security goals based on canonical security definitions, and understanding the stakeholder goals to develop preliminary security requirements and to prioritize them. It uses a case study to validate the feasibility and effectiveness of the proposed approach.

Keywords: Application Development, Part Of Speech (POS) Tagging, Requirement Engineering Process, Security Requirements, Software Security

1. INTRODUCTION

Software security is a complex, evolving problem that can be significantly improved by integrating security requirements into the early stages of software development rather than correcting security flaws after release (Allen, Barnum, Ellison, McGraw, & Mead, 2008). However, traditional software development life cycle (SDLC) processes tend to focus attention on functional requirements leaving non-functional requirements, such as security, as an aside or afterthought. This results in security requirements that are added later (McGraw, 2005) in the development cycle or

DOI: 10.4018/ijsse.2014100102
worse, after the product has been released in response to security events, market response, or regulatory demands.

There are several reasons for the reactionary response to software security. First, software engineers have a difficulty in building secure software due to a lack of software security awareness, training and education (Viega, 2005). Additionally, decisions about security may simply have been made based on the technology and capabilities available at the time the system was developed (i.e., early infrastructure systems). Finally, project cost constraints may focus resources on delivering functional requirements over non-functional requirements, such as security. Regardless of the reasons for this reactionary response to security, both software engineers and business stakeholders are becoming increasingly aware of software security needs.

Recent news reports of highly publicized data breaches have increased general awareness of the need to integrate security into software during development. In response, legislation at the state and federal level has also been increasing as the need for privacy and security becomes apparent. Such increased legislation is evidenced by the fact that nearly all states have enacted either security, or data breach notification legislation. In general, software engineers have reacted to increased public awareness and legislative pressures by adding security mechanisms to existing systems on an ad hoc basis to mitigate risk. While subsequent mechanism based mitigation is a useful (and sometimes necessary) approach when addressing new, evolving, or previously unknown security risks, the approach often results in isolated, add-on countermeasures that are not cohesively integrated into the resulting system and its design. Money and time may be lost as software engineers work to prevent additional bugs as they retrofit existing systems with added security mechanisms.

However, if the security risk existed prior to development, then a security requirement could have addressed the risk by building security into the system from the start. In this case, implementing security mechanisms does not address the core problem that security requirements need to be integrated into software from the start, not mitigated later. The emerging field of security requirements engineering (SRE) seeks to address the special needs of integrating security requirements into the software development process.

1.1. Security Requirements Engineering

When it comes to specifying security requirements, stakeholders have different expectations. Integrating software security into development requires eliciting security requirements along with other requirements. From the business stakeholder viewpoint, software requirements are an extension of business goals that stakeholders wish to implement in a software product. Business goals generally represent desired functionality, but may also imply general security needs based on limited security knowledge that business stakeholders possess. Using these goals, the software requirements engineer must elicit functional and non-functional requirements. However, a hurdle to eliciting security requirements is the difficulty that stakeholders and software engineers have with explicitly expressing security needs. Security may again be expected, but verbalizing specific security requirements may be difficult. The combination of vague security goals and limited security specific resources often results in a requirements specification which may not include security specific requirements.

To remedy the lack of software security, requirements engineering approaches have been proposed to aid in the development of security requirements. Security requirements should be elicited and developed along with functional requirements and should be included as part of the software requirements specification. Best practices, enumerations, methodologies (Hallberg & Hallberg, 2006; Romero-Mariona, 2009; Mead, Hough, & Stehney, 2005), models (Luckey, Baumann, Méndez, & Wagner, 2010; McGraw, Migues, & West, 2012) and elicitation techniques (Ingoldsby, 2009) have been
Individual Improvisation in Information Systems Development
www.igi-global.com/chapter/individual-improvisation-information-systems-development/21064?camid=4v1a

Measuring Models
www.igi-global.com/chapter/measuring-models/26829?camid=4v1a