A Software Requirement Tool for Capturing Implied Security Requirements

Annette Tetmeyer, University of Kansas
Hossein Saiedian, University of Kansas

Abstract. While expressing software requirements and needs, many clients, especially the non-technical ones, will indirectly imply concerns and expectations that are security related. One way to capture such implied concerns (i.e., security requirements that are not explicitly stated) is to use a parsing tool and look for terminology and keywords that indirectly (and perhaps sometime very directly) imply security requirements, constraints, or expectation. Such keywords will be tagged and further refined into formally specified software requirements and incorporated into the final requirements document. We introduce such a tool and a mini process for utilizing it. Our approach has the advantage of steadily incorporating security requirements engineering into existing software development processes with minimal disruption while adding value to the software development process.

Introduction

Scan and carefully review any software requirements specification artifact for security related terms such as "password", "encryption", "authorization", "integrity", "hacking", "accountability", "monitoring", "controlling", "event log", or even "security". Are these terms likely to be found within the artifact? Yes. Are these terms associated with a security specific requirement? Possibly. What does this imply? In many cases, security is implied within software requirements but may not be specifically considered a security requirement. In addition, the requirement that contains security terms may be vague and open to interpretation depending on the viewpoint of the reader. Implied security requirements may be creeping into software requirements due to increasing awareness of security needs from novice end-users to security experts alike. Data breaches, privacy issues and security concerns related to software are increasingly headline news and are raising the security awareness of a broad spectrum of the population. Software users increasingly expect security even if they cannot clearly define what security means. Software developers are responding by implementing security features to mitigate risk, but often this is on an ad hoc basis. Legislators at the state and federal level are also enacting regulation in response to security events. All of these responses are primarily reactionary in nature. The question is what should be done to improve the integration of security into requirements engineering from the start rather than reacting later?

Security requirements engineering (SRE) [1] is receiving more attention as a not only a valid, but increasingly necessary part of the software development process. Building security into software, much like building quality into products from the start, can improve the overall security of software and move from a reactionary to a proactive approach to secure software [2,3]. In order to build secure software, processes and tools are needed that address software security in the early stages of development as a part of the normal software engineering process. Regardless of the type of software development life cycle model (waterfall, agile, spiral, etc.) used, all include a requirements phase in the earliest stage of development. During development, functional and non-functional requirements are elicited, analyzed and developed into software requirements with the key goal being that these requirements are as final as possible. Requirements changes during development are the bane of software project managers due to the amount of effort needed to address these changes (particularly if changed during the late stages of development). It is highly desirable to develop stable requirements as early as possible to reduce the detrimental impact on cost and time as a project progresses.

One type of software requirement that can be overlooked is security requirements. Even the most casual of software users may expect a minimum level of software security even if they have a hard time defining security expectations. Defining security requirements can be difficult due to a lack of common ground among stakeholders in terms of security knowledge, skill, and even vocabulary. If stakeholders cannot define and understand security needs, then it is unlikely that security requirements will be properly elicited, captured and defined. Even if security requirements are defined, they are often seen as constraints on functional requirements and can be at odds with the goals of stakeholders. The cost of developing and implementing security requirements may also be a difficult sell to those signing off on the cost of the project. This leaves members of the software development team in a position in which they must anticipate the security goals of client and develop the appropriate security requirements as early as possible in development. Finally, security requirements must be justified based on a risk-reward analysis.

Security requirements engineering must become a prioritized part of the software development process and tools must be developed to aid in developing security requirements.

Best practices, enumerations, methodologies, models and elicitation techniques have been proposed that are intended to improve the integration of security requirements into early phases of development. A key factor in each of these is the focus on the first stage of software development or requirements development. Software developers who have previously not emphasized the development of security requirements must start including them in their software development processes. However, jump-starting an SRE initiative can be daunting and if initially unsuccessful, can be a detriment to future inclusion of security requirements. An alternative method, particularly for a small software development team, is to capture security requirements during the normal requirements process and build to a comprehensive security requirements engineering process. Stakeholders often have difficulty expressing security related needs but use terminology that implies a security need. By capturing these implied security needs, further elicitation activities can be undertaken to refine security needs into security requirements.

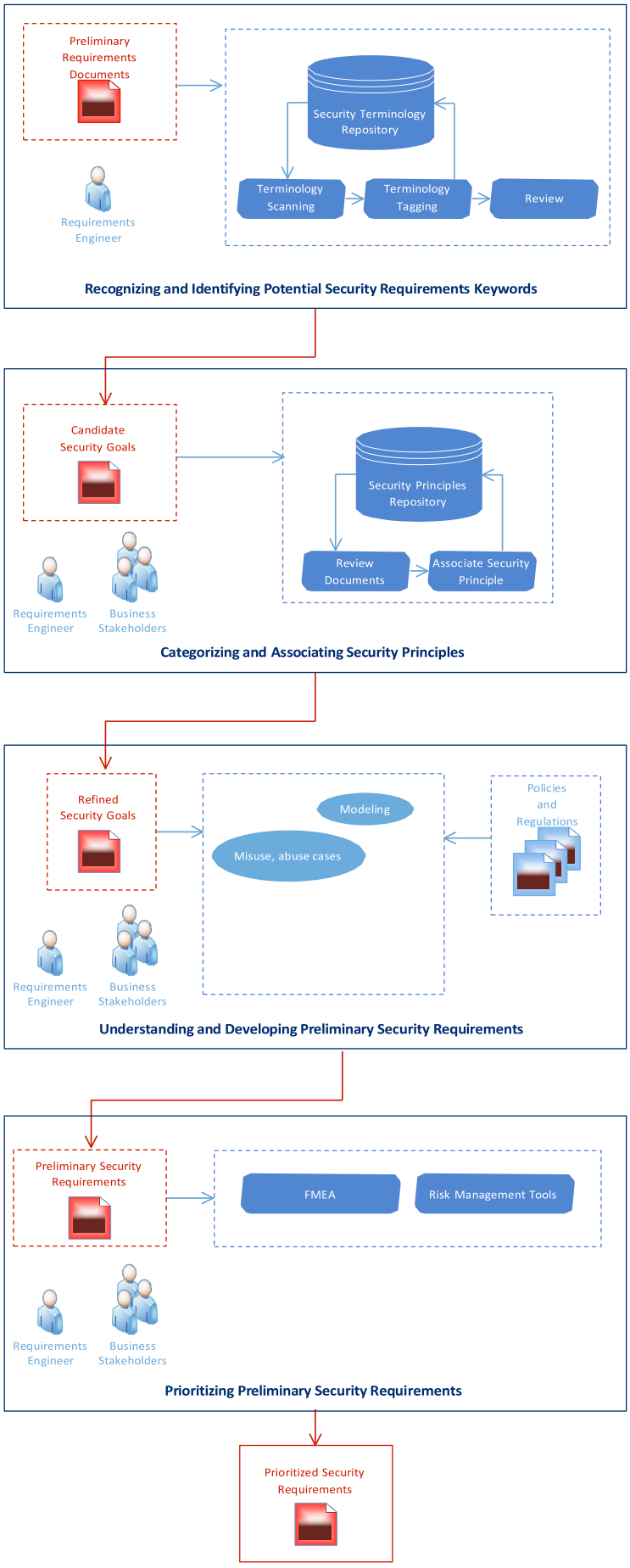

Figure 1: A Tool for Capturing Security Requirements

A Requirements Tool for Capturing Security Requirements

How can security requirements be inferred from general software requirements? One method is to parse the natural language that stakeholders use when defining requirements to extract security implied terms. During requirements elicitation, security related terms may even be included in general requirements without identifying them as security specific requirements. Identifying and extracting phrases based on these terms is the first step to understanding security goals. During elicitation activities, the requirements engineer can further extract security needs from stakeholders and lead prioritization activities in order to convert security goals into security requirements. A tool to identify, categorize, understand and prioritize security goals that is integrated into requirements elicitation activities can be the starting point for a security requirements engineering initiative. Figure 1 illustrates the software requirements artifacts input and output, stakeholder roles and the processes defined by the security requirements capturing tool.

Recognizing and Identifying Potential Security Requirements Keywords

The requirements engineer starts by identifying potential security goals based on security terms and phrases appearing in preliminary requirements artifacts. Preliminary requirements artifacts can be formal software requirements specifications, user stories, business process documents, or other documents related to requirements specifications. The type of requirements artifacts and software life cycle model used is not a constraining factor; the point is to identify implied security requirements based on terminology. The identification of these terms and location within the requirements artifacts are the starting point for security requirements elicitation.

Scanning can be manual for small artifact sets, but an automated scanning tool is desirable for larger artifact sets. An automated scanning tool can be easily implemented for common types documents (such as Word or text documents) and should identify the frequency and tag the location of security terminology passages in artifacts for further analysis. Prior to scanning, the requirements engineer would define a set of security terms or phrases in a security terminology repository. The starting set of terms can be based on the security knowledge of the requirements engineer or by using a common dictionary of security terms. Over time, the security terms and phrases would be refined by the requirements engineer for reuse on other projects. After initial scanning, the requirements engineer with review the tagged terminology to determine an initial set of candidate security goals for further requirements elicitation.

Categorizing and Associating Security Principles

Categorizing security goals and associating security common security principles will aid in gaining a deeper understanding of the stakeholders security requirements. The requirements engineer should be knowledgeable about general security and key security principles such as confidentiality, integrity, and availability principles (also referred to as CIA). A starting set of security principles will be defined and used to categorize and associate with

each candidate security goal. Working with business stakeholders, the requirements engineer will be able to extract a deeper understanding of needs while also building a common ground of security understanding with the business stakeholders. Tagged security passages should be refined from general or vague to more exacting language that clearly defines the security need. Categorizing based on a common set of security principles and including this language within the security goals will also aid in understanding. After an initial review among all stakeholders, any tagged passages containing security terms that are not deemed as candidate security requirements should be discarded from further review.

Understanding and Developing Preliminary Security Requirements

Using the refined security goals as input, the requirements engineer will continue elicitation activities with business stakeholders to develop preliminary security requirements. Structured or unstructured elicitation activities such as face-to-face meetings or review sessions will aid in communication and yield further understanding of the stakeholder's needs. Activities specifically related to developing security requirements such as modeling activities, developing misuse/abuse cases, or building attack trees can be undertaken at this time. Business artifacts such as policies and regulations should be used as input at this time. The goal of these elicitation activities is to gain a deeper understanding of the implied security goals and develop them into preliminary security requirements.

Prioritizing Preliminary Security Requirements

Next, candidate security requirements should be prioritized. There is always a trade-off to be made when determining which requirements are feasible to be implemented regardless of the type of requirement. Functional requirements are often cut from consideration if they are deemed too costly or do not meet a return on investment (ROI) threshold. Security requirements are not immune to analysis to determine if they are feasible. Attaining 100% secure software is not feasible. The software development team and business stakeholders should perform ROI or risk analysis to determine which candidate security requirements should be implemented. These activities will further enhance communication and foster familiarity with software security among all stakeholders. In the absence of a preferred analysis technique, Failure Modes and Effects Analysis (FMEA) can be utilized.

FMEA is an analysis and decision-making tool often associated with quality and Six Sigma methodologies [4]. A failure mode is the manner in which something might fail. Effects analysis is the study of the consequences of these failures. FMEA is used to identify, estimate, prioritize, and reduce the risk of failure. As a software engineering tool, FMEA is not widely used, but has advantages over other analysis tools in that it is easy to implement and can be used by a broad audience. A requirements engineer can use FMEA to elicit security related information from stakeholders, prioritize the data, and present an analysis of the risks associated. The prioritized risks allow for informed decision making to choose which actions to consider.

This approach is very useful to communicate and clarify the impact of technical materials in an easy to understand format.

Analysis requires creating severity, occurrence and detection rankings in order to determine a risk priority number (RPN). A standard scale for severity, occurrence and detection can be adopted as a starting point for FMEA analysis but experienced FMEA users may wish to develop more refined rankings scales. A standard scale ranges from a low of 1 for unlikely to a high of 10 for very high. The RPN is calculated as the product of the risk rankings:

RPN = (severity ranking)(occurrence ranking)(detection ranking)

The requirements engineer could generate a preliminary ranking of candidate security requirements and follow-up with business stakeholder or all stakeholders could be involved at the start of analysis. Rankings for severity, occurrence and detection are determined by the stakeholders and the RPN is calculated. The resulting RPN generates a prioritized list of potential security requirements. Using the FMEA results, requirements engineer and business stakeholders will refine the preliminary security requirements until a list of final security requirements has been generated.

Scenario to Demonstrate the Capturing Security Requirements Tool

The following scenario demonstrates the capturing security requirements tool. A software developer has been contracted to develop a software application for a small organization. The software developer embraces agile software development methodologies (face-to-face customer interaction, lean techniques and minimal documentation) and is accustomed to fast-paced project schedules. Preliminary requirements artifacts have been developed using standard word processing tools and the software developer's requirements document template. Scanning and tagging of the preliminary requirements artifacts have revealed an implied security need to limit the impact of "unauthorized" users. Working with business stakeholders, the elicitation activities associate the security principles of confidentiality and integrity with the use of the term "unauthorized" in the requirements artifacts. In this case, requested data needs to be protected against unauthorized disclosure (confidentiality) and as well as against unauthorized modification or destruction (integrity). The refined security goals are further understood and developed by reviewing relevant regulation as well as by developing misuse and abuse cases with the business stakeholders. Both the requirements engineer and business stakeholders are beginning to fully appreciate the need to refine these security goals into preliminary security requirements. A new set of preliminary security requirements is added to the requirements artifact document to address the impact of data requests by unauthorized users. The final step is to prioritize the preliminary security requirements use FMEA analysis. The requirements engineer and business stakeholders identify three failure modes and effects for analysis (see Table 1). Severity, occurrence and detection rankings are agreed upon and RPN's are calculated. The resulting FMEA analysis reveals that both data being viewed or stolen by an unauthorized user represents a strong risk to the business and need to be represented in the security requirements. Data corruption by an unauthorized user

Table 1: FMEA Analysis of Security Requirements

Failure	Effect	Severity	Occurrence	Detection	RPN
data request by an unauthorized user	data viewed	3	7	9	189
data request by an unauthorized user	data stolen	9	4	9	324
data request by an unauthorized user	data corrupted	5	4	4	80
Standard Impact and Rating Scale for Severity, Occurrence or Detection Very High (9-10), High (8-7), Moderate (4-6), Low (3-2), Unlikely (1)					

represents a lower risk and the business determines that this requirement does not need to be represented in the security requirements. Therefore, a new security requirement is written to address the impact of data requests by unauthorized users and is included in the final software requirements artifact.

A Brief Analysis of the Security Requirements Capturing Tool

In order to be successful, the overall approach (process and tools) must be both measurable and feasible. Measurability is a key to determining the success of any process or tool. Scanning and tagging of security terms and phrases provides the basis for benchmarking the use and importance to security requirements within existing documents. Over time, statistics can be gathered to refine the process of determining the importance of implied security terms within requirements artifacts. Risk analysis activities, such as FMEA, also provide the ability to analyze security requirements activities. Implementing an SRE process can quickly become overwhelming due to the complexity of software security, resources required and need for security expertise. All projects need to balance cost and resources in order to deliver on time and on budget. This approach combines both process and tools to feasibly meet project goals. The process of integrating security requirements into an existing requirements engineering process of capturing implied security needs by tagging security terms can be a feasible addition to existing processes. Additional resources from the development team (such as a security expert) are not compulsory and activities are integrated into existing processes. Existing requirements artifacts are used as the basis for SRE activities. An automated scanning tool to tag security terms is desirable, but can be built or acquired at minimal cost. Minor training may be required to implement FMEA or other analysis activities. In general, all of these activities can be implemented at any scale and can grow and mature with the needs of the organization. Other security frameworks, best practices and models can coexist side-by-side with these activities with minimal disruption. Therefore, the proposed approach is a feasible alternative to beginning and building a SRE initiative for any organization.

Summary

SRE is one part of a solution for secure software development. Security requirements can be captured by scanning and tagging security terminology within requirements artifacts to identify security goals. Elicitation activities further refine the goals into security requirements by associating each with security principles and developing a deeper understanding of stakeholders needs. Risk analysis prioritizes preliminary security requirements to determine a final list of security specific requirements. These newly identified security requirements are integrated into the final software requirements document. The development of security requirements is integrated into existing software development processes in order to build a SRE initiative.

Further efforts can be continually refined to build the basis for secure software development.

ABOUT THE AUTHORS

Annette Tetmeyer is a Ph.D. candidate in computer science at the University of Kansas. Her research interests include security requirements engineering, HCI, and engineering education. In addition to experience in private industry, she has taught a variety of undergraduate and graduate engineering courses at the University of Kansas. She received her MS in Computer Science from the University of Kansas (2013) and BS in Mechanical Engineering from Iowa State University (1993).

Phone: 785-864-8812
E-mail: tetmeyer@ku.edu

Hossein Saiedian (Ph.D., Kansas State University, 1989) is currently an associate chair, director of IT degree programs, and a professor of computing and information technology at the Department of Electrical Engineering and Computer Science at the University of Kansas (KU) and a member of the KU Information and Telecommunication Technology Center (ITTC). Professor Saiedian has over 150 publications in a variety of topics in software engineering, computer science, information security, and information technology. His research in the past has been supported by the NSF as well as other national and regional foundations.

Phone: 785-864-8812
E-mail: saiedian@ku.edu

REFERENCES

1. J. H. Allen, S. Barnum, R. J. Ellison, G. McGraw, and N. R. Mead, Software Security Engineering: A Guide for Project Managers: Addison-Wesley, 2008.
2. G. McGraw, S. Migues, and J. West. (2012). The Building Security In Maturity Model.
Available: <http://www.bsimm.com/>
3. G. McGraw, "The Security Lifecycle-The 7 Touchpoints of Secure Software-Just as you can't test quality into software, you can't bolt security features onto code and expect it to become hack-proof Security," Software Development, vol. 13, pp. 42-43, 2005.
4. ASQ. Failure Mode Effects Analysis (FMEA). Retrieved 11/21/13, from <http://asq.org/learn-about-quality/process-analysis-tools/overview/fmea.html>