Increasing Trustworthiness Through Security Testing Support

Jose Romero-Mariona
University of California, Irvine
Donald Bren School of Information and Computer Sciences
jromerom@uci.edu

Hadar Ziv
University of California, Irvine
Donald Bren School of Information and Computer Sciences
ziv@ics.uci.edu

Debra Richardson
University of California, Irvine
Donald Bren School of Information and Computer Sciences
djr@ics.uci.edu

Trustworthiness is an essential and sometimes life-critical concern in software-intensive systems. Furthermore, supporting the proper testing of these systems can often times prove complicated. Within trustworthiness, security and privacy play key roles. Considering both security and privacy issues early in development are necessary to increase the trustworthiness of systems. In this paper we concentrate on the security aspect of trustworthiness. We believe that proper capture of software security starts at requirements and carries forward throughout system development, especially into testing. While a variety of techniques for specifying security requirements exist, there is a tangible lack of support for making security requirements useful during testing. In this paper we present testing capabilities of a new security requirements engineering technique called SURE, Secure and Usable Requirements Engineering. SURE focuses on the mapping of security requirements into testing artifacts using a specialized syntax. We also present results from a usability study of our technique's testing capabilities; the study confirmed that participants were able to map testing artifacts from security requirements with increased understanding and confidence, which could potentially lead to improved trustworthiness.

Keywords—security requirements; security requirements-based testing; usable requirements

I. INTRODUCTION

The past decade has witnessed unprecedented advances in software and applications, for example Internet commerce, Web and mobile applications, and cyber infrastructure. With the progress and innovation brought forth by these advances, there are new areas of concern specifically with regard to potential breaches in trustworthiness a system offers its end-users. Trustworthiness is a software quality that must be addressed properly to achieve trustworthy computing (TC). Trustworthy computing is an important area of concern for both academia and industry, requiring further research. The 1999 National Academy of Sciences study “Trust in Cyberspace”, which indirectly led to NSF’s Trustworthy Computing program, documents trustworthiness (or lack thereof) as potentially compromising the security and privacy, physical safety, and economic well-being of individual citizens as well as the information security of government and industry [1]. Likewise, in 2002, Bill Gates announced that trustworthy computing would be Microsoft’s top priority as they moved forward [2].

For the purpose and context of this paper, software development, we adopt the definition of trustworthiness proposed by [3] as being composed of security and privacy. Security is a software property that has as its primary concern the preservation of confidentiality, integrity, and availability of information [4]. Privacy, in software development context, refers to the right of an individual to determine what information about themselves should be known to others [5]. While we recognize the importance of both security and privacy with respect to trustworthiness, this paper focuses primarily on the security aspect.

Software security is increasingly becoming an essential and often life-critical concern in developing quality software. Adding security concerns to a system late in development often leads to significant rework later or, worse yet, reduced overall security. Security could benefit tremendously from methodologies and tools that allow users to address security concerns during requirements specification, as well as after requirements into subsequent developments steps. It is also important to evaluate and confirm the usefulness of such methodologies and tools.

In this paper we argue for security requirements-based testing as means of improving (or at least impacting) the trustworthiness of a system. Current security requirements engineering (SRE) approaches often lack support for security requirements that are testable and useful during testing. To address this we developed Secure and Usable Requirements Engineering (SURE), a new technique that supports the specification of security requirements and their mapping into testing artifacts. We also discuss ASSURE (Automated Support for Secure and Usable Requirements Engineering), a software system that implements SURE as well as aids in managing all the artifacts produced. Following a brief introduction of the specification capabilities of SURE, this paper focuses on its support for requirements-based testing.

The main contribution of this paper is the introduction of a new SRE technique for specifying security requirements then mapping them into testing. In addition, we describe results from an evaluation study where Ph.D. candidates carried out a variety of testing-related tasks using SURE. We discuss both quantitative and qualitative responses obtained from study participants. Our evaluation indicates that SURE...
provides strong and useful support for mapping security requirements into test artifacts. The paper is organized as follows: Section 2 discusses the concept of security requirements-based testing and related works. Section 3 provides background on both SURE and ASSURE. Section 4 describes the process involved in mapping testing artifacts with SURE. Section 5 presents results from the usability study on ASSURE’s testing support. Section 6 concludes the paper as well as presents our future directions.

II. TRUSTWORTHINESS AND RELATED WORKS

Trustworthy computing is a complex area, due in part to its non-functional nature [6]. As suggested by [7], the lack of focus on testability, usability, and ultimately trustworthiness throughout software development, impede progress and improvements in trustworthy computing. Moreover, trustworthiness is an elusive quality. For instance, despite extensive research from Microsoft over a period of seven years, a 2009 report [8] showed that very little had been accomplished. One of the main reasons for this was the fact that trustworthiness was approached as a patch or add-on, rather than a concern that was developed along with the system.

Trustworthiness must be treated as a first-class quality attribute, meaning that it is given the same importance as functionality from the beginning and resources are allocated and expended throughout development. Attention must be given to trustworthiness during requirements engineering (RE) as the first major stage of software development. It is during this stage, that the customer and developers come to an agreement as to what constitutes the system to be developed, and should explicitly address what quality attributes are priorities. RE is a critical stage of development because anything that is (or is not) resolved will be carried throughout the rest of development. Good requirements engineering is therefore essential for trustworthy computing [9].

In addition to good RE, we argue that proper testing support for these requirements is just as important. The problem that developers are often times faced with is the lack of support for testing, particularly when it comes to non-functional requirements like security.

A. SRBT

We define requirements-based testing as a process that deals with two major issues, namely the validation of requirements followed by the design of test cases from those requirements [10]. In this paper we introduce the concept of Security Requirements-Based Testing (SRBT). Similar to requirements-based testing, SRBT looks to validate security requirements to ensure their correctness, completeness, unambiguity, consistency, and level of security. Once these security requirements are validated, test cases are designed and derived from them. SURE aims to support SRBT through its requirements and testing process as well as the tool support provided by ASSURE.

Within requirements-based testing we are interested in model-based testing and specifications-based testing. Model-based testing refers to deriving test cases from a model that represents a system’s behavior [11, 12]. Ulrich classifies three different types of models, test models, design models, and models constructed from source code [13]. Model-based testing could be used to improve quality attributes [14] on a variety of systems. Furthermore, Williams et al. express that it is beneficial to support the automatic generation of test cases from these models [15].

In Specifications-based testing, tests tend to be generated from an abstract system specification. These tests are then used to verify a system’s implementation [16]. Wimmel and Jürjens [17] use specification-based testing to test an implementation. They compute test sequences from a security model that covers possible violations of the original security requirements. Even though specification-based testing is done mostly manually, [18] describes the benefits of shifting to automatic specification-based testing.

Similar to model-based and specification-based testing, SURE’s SRBT capabilities are black box-based. Different from MBT and SBT, however, SURE focuses directly on original requirements, not any model created from them nor specifications derived from models. Similarly, ASSURE does not automatically generate test cases from security requirements, but rather supports the manual mapping of requirements into test artifacts.

III. BACKGROUND

Traditionally, requirements and testing are different and distinct stages of software development, often separated in many ways. This is especially true with non-functional requirements, including security, where testing and test artifacts are often not considered until after functional requirements are designed and implemented. SURE aims to bring security requirements and testing closer together. Consider, for example, a simple login system in a domain such as online banking, e-commerce, or similar; one of its security requirements could be that “the system shall enforce 128-bit encryption on every username.” SURE helps map such security requirements into testing artifacts early in the development process. One possible testing artifact that can be created from this requirement is that “a username lacking 128-bit encryption is registered in the system; the system should detect this and warn users about the lack of security.” We will revisit this example throughout this paper, and detail the ways in which SURE supports the mapping of security requirements into testing artifacts.

This paper is greatly motivated by an earlier literature survey [19] of current approaches to SRE; this survey also identifies work relevant and related to the method and tool described here. The survey compared the level of support offered by current SRE methods across five areas of security requirements engineering, namely their elicitation, analysis,
specification, maintenance and support for later development stages. Out of 30 possible SRE approaches, 12 were selected for surveying based on their maturity, relevance to both industry and academia, and the amount of information and documentation available regarding each approach. They are:

- Misuse Cases [20]
- Abuser Stories [21]
- Secure TROPOS [22]
- Security Problem Frames [23]
- Anti-Models [24]
- i* Security Requirements [25]
- Common Criteria [26]
- SQUARE [27]
- OCTAVE [28]
- Attack Trees [29]
- USeR Method [30]
- CLASP [31]

The results of our literature survey indicated that there is a tangible need to further support and benefit from security requirements beyond specification. Existing SRE approaches lack significantly in their support for creating useful, usable, and understandable security requirements. This weakness, in turn, motivated our creation of a new SRE approach – SURE.

A. SURE

The SURE technique enables users to elicit, analyze, and specify security requirements as well as document misuse scenarios and possible threats to the system. At a high level, some of the major benefits and distinctive features of SURE include:

- Supporting non-security experts, such as customers and developers, in the specification process
- Following a well-defined process that helps specify security from different perspectives
- Providing guidance for mapping of testing artifacts from security requirements

A key feature and focus of SURE, distinguishing it from other SRE approaches, is that it brings together a variety of stakeholders. Since no one person or group of individuals can fully specify security requirements, SURE supports the following three types of stakeholders: customers, developers, and security experts.

SURE guides users through the creation of two types of specification artifacts, namely requirements artifacts and misuse cases artifacts.

1) Requirements Artifacts

SURE uses a three-step refinement process for creating security requirements. At each step a new artifact is created from the previous one; the three types of artifacts describe important aspects of the system, namely security statements, security needs, and security requirements. Users start the process by specifying security statements (SS); a security statement is a high-level idea that describes some security aspect about the system at hand. Security statements are then refined into security needs (SN); there can be one or more needs per security statement. A security need extracts important security information from each security statement. Security needs are then refined into actual security requirements (SR); there can also be multiple security requirements that satisfy each need. A security requirement specifies the resources needed to secure each need. The SURE process produces and results in requirements artifacts of the three types mentioned above.

2) Misuse Cases Artifacts

Misuse cases are use cases from the point of view of an attacker. They provide means for thinking about details of the project within a proper setting; this in turn facilitates uncovering details that are hard to consider in stand along security statements. Misuse cases describe scenarios where the system could be misused or mishandled. For each misuse case the SURE technique aids users in extracting misuse case consequences and misuse case threats. The first type of scenarios specifies repercussions that the system could suffer if the misuse case actually happens. The second associates a variety of known security threats to the misuse case; this helps in considering early in the development process some of the possible threats that the system could face once deployed.

B. ASSURE

ASSURE is a system implemented to support users in the artifact creation process defined by SURE. Additionally, ASSURE extends its SRE support to also aid with project management. ASSURE includes user and project tracking utilities, which, for example, make sure that every stakeholder is associated with a particular project and is able to offer his or her feedback. Furthermore, ASSURE provides project progress tracking, evaluating stakeholder security skill levels, offering a messaging system, and supporting the view of security artifacts in the context of keywords; this not only reaffirms the artifact relationships, but also promotes security requirements reuse. Furthermore, ASSURE links all requirement artifacts to a test case artifact and keeps track of which requirements do not yet have a test case associated with them. In a nutshell, ASSURE provides an interface to the process described by the SURE technique, allowing for users and projects to be managed.

ASSURE implements the SURE technique by enabling stakeholders to create the various requirement and testing artifacts as outlined by the technique. Some of the support provided by ASSURE in this area includes:

- The implementation of SURE’s process for security requirements specification
- The implementation of SURE’s process for mapping testing artifacts from security requirements
- The implementation of SURE’s syntax for requirements and testing artifacts

C. Running Example

In order to describe SURE’s support for requirements-based testing, we introduce the following example along with artifacts specified for this example. Assume you have been asked to specify the requirements for an online banking application. The first module to be specified is the login system that confirms users identities. Given that this is a security-sensitive system, security requirements must be specified.

IV. TESTING WITH SURE

This paper focuses on SURE’s support for requirements-based testing. Below we explain the syntax proposed for test artifacts.

A. Testing Artifacts Types

The SURE process produces six types of requirements artifacts. In order to consolidate various artifacts produced during the testing stage, only three types of requirements artifacts are mapped into testing artifacts. The three types of testing artifacts, derived from requirements artifacts, are:

- Requirement Test Case (RTC): Directly derived from a unique SURE security requirement.
- Misuse Test Case (MTC): Derived from a unique SURE misuse case.
- Threat Test Case (TTC): Derived from a unique threat identified using SURE.

B. Testing Artifacts Syntax

We propose our own syntax for recording testing artifacts using the SURE technique. The syntax proposed makes it easier to track which artifacts have been mapped (and which still need to be) and understand their content. Each testing artifact is comprised of the following fields,

- Test Artifact ID: Unique ID that links requirements
- Test Artifact Name: This field also links the requirements artifact from which the current testing artifact is being derived
- Steps to Test: Steps that need to be followed in order to conduct the test
- Input: Types of input the testing artifact will expect
- Expected Output: Types of output that this testing artifact should produce

Based on this syntax, for our example system, a security requirement (SS1-SN1-SR1) would map to the corresponding test case shown in Figure 1 below,

![Sample Requirements Test Case](Figure 1)

Test cases for misuse and threats would look similar to Figure 1, but reflect the content specific for each type of artifact as specified during the requirements stage.

V. EVALUATION

In this section, we report on a study conducted to evaluate SURE’s testing support.

A. Study Design

In the evaluation study, participants were tasked with mapping security requirements, similar to those in the running example, into testing artifacts, as follows.

1) Study Tasks.

Evaluation participants were asked to complete three different tasks, labeled Task A – Task C. These tasks dealt with the creation of testing artifacts given a set of security requirements artifacts. Below are the three tasks participants were asked to complete,

- Task A: Create a new requirements test case
- Task B: Create a new misuse test case
- Task C: Create a new threat test case

2) Study Logistics.

Participants were asked to come to our research lab and allow one hour for the study. We conducted the study with two participants at a time, each using a laptop computer with ASSURE running on it. Participants were given a list of tasks and directions to accomplish each task. As participants completed tasks they had a “rating” sheet to record the support provided by ASSURE, similar to that in Figure 6.
3) **Study Population.**

Five computer science Ph.D. candidates were selected to participate in the study. All of the students have developed test cases in the past, in both academic and industry settings, but none have worked with security cases. These students represent typical users of ASSURE.

4) **Evaluated Aspects.**

For each task, we evaluated the support provided by ASSURE along the following three aspects:

- Aspect 1: Overall Support- What amount of support ASSURE provided to complete task
- Aspect 2: Artifact Understandability- How understandable is/are the artifact(s) created
- Aspect 3: Artifact Confidence- How confident are you with the artifact(s) created

Participants were asked to rate the support ASSURE provided for each aspect on a scale of 1 (Extremely Low), 2 (Low), 3 (Medium), 4 (High), 5 (Extremely High).

These are aspects that will also aid in understanding the support offered for trustworthiness. The understandability and confidence aspects are related to our interpretation of trustworthiness, therefore the ratings received for them can be used to hypothesize how our approach would improve trustworthiness.

5) **Threats to Study’s Validity.**

It is important to discuss possible threats to the validity of the evaluation study. First, the population is small as well as homogeneous, consisting only of Ph.D. students. While the students are valid representatives of the target audience, we would have liked to conduct the study with a larger and more diverse user population, including industry users. Second, we would have liked them to carry out a larger set of tasks.

B. **Evaluation Results**

Once the rating for each task was collected from all five participants, we tabulated the results. Figure 7 shows the average response from all five participants for each task.

![Figure 3. Average User Response per Task](image)

The overall support that ASSURE provided for each aspect (shown in the blue bar) was at least “High” for all three tasks completed by participants. The understandability of the artifacts created during each task was also ranked “high” (shown in the red bar). Lastly, the confidence that participants have in artifacts is essential, for they are non-security experts. On average participants felt “highly” confident (shown in the green bar) in the artifacts created with ASSURE.

C. **Evaluation Observations**

In spite of the positive responses from participants, the ratings obtained in the study show areas for improvements. Below we present these areas from the perspective of numerical ratings and free-response areas of the evaluation.

1) **Numerical Ratings.**

The average response to each aspect across all tasks was very positive. The average response for overall support was 4.1 (out of 5); average response for artifact understandability was 3.6 (out of 5); and the average response for artifact confidence was 4.4 (out of 5). This tells us that participants were highly confident in that the artifacts created were done correctly, as well as they were highly understandable. We consider these results as indications that our approach could definitely improve the trustworthiness of a system based on the artifacts created for it.

Figure 7 shows that, of the three aspects, participants found artifact understandability to be the least supported by ASSURE. This requires further investigation, but may be due to the fact that participants were not familiar enough with ASSURE’s representation of artifacts.

2) **Free-response.**

From comments participants made after rating each task (in the “notes” area as shown in Figure 6) for overall support the common suggestion was that ASSURE provides pre-filled test cases that can then be customized based on each requirement artifact. This is a comment that we believe will have a direct impact on artifact understandability. By providing examples of what artifacts should look like, users will be able to familiarize themselves with the way ASSURE stores and displays artifacts. For the aspect of artifact understandability participants suggested that the naming convention for the testing artifacts be adjusted; they felt that there was too much information in each box (as the title in the box traces the origin of each artifact). Adjusting the naming conventions will not only free-up space in each artifact, but prevent information overload. For increasing the confidence in the artifacts created participants suggested that we incorporate a new interface that allows security experts to revise the testing artifacts created. This is an excellent suggestion that validates our initial idea of providing security expert support at this level.

In addition to the notes section of each task’s aspect, we also asked participants to answer two questions at the end of the evaluation. We asked participants which features they liked in ASSURE and what are some overall improvements that could be made. For the first question, the majority of
participants conveyed that the way the interface organizes and displays all of the artifacts (colors and hierarchy) really helped in visualizing what has been completed so far and what needs attention. In addition, a good number of participants also felt like SURE and ASSURE could aid in the specification and mapping of privacy requirements. This is extremely promising, as it would enable testing that would directly affect trustworthiness. For the latter, most participants agreed that it would be helpful to include examples of what each testing artifact should look like, so that they can learn by example, and then model their own input after the examples.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we argued for better support for testing of security requirements in order to improve trustworthiness. We described the testing support offered by a new technique (SURE) when it comes to mapping security requirements into testing artifacts. SURE uses a specific syntax to aid in the mapping of security requirements. In addition, ASSURE provides support for not only the specification of security requirements but, more importantly, their mapping into testing artifacts and artifact management.

We also presented results from a preliminary evaluation study where five Ph.D. candidates completed a variety of tasks using our technique. The overall response was very positive for all aspects of each task.

Future work will focus on understanding how our work on security can be extended/adapted to also support privacy. Improvements to the evaluation will be made, where a bigger emphasis can be put into trustworthiness-related aspects. We also plan to improve ASSURE itself based on the comments received during the evaluation, for example by adding features and capabilities for security experts.

REFERENCES