Towards Usable Cyber Security Requirements

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

Dennis Bystritsky
University of California, Irvine Donald Bren School of Information and Computer Sciences
dbystrit@uci.edu

ABSTRACT
Security has become a primary and prevalent concern for software systems. The past decade has witnessed a tremendous increase in not only the sheer number of attacks but also the ease with which attacks can be performed on systems. In this paper we exemplify the usage of a novel technique for developing security requirements, by demonstrating each step in the technique when applied to an example usage scenario. Furthermore, this new technique also provides support for deriving testing artifacts from the specified security requirements. We believe that in order to protect a system against harm (intended or not), attention must be given to its requirements. Similar to other system properties and quality attributes, security must be considered at the requirements.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/Specifications

General Terms
Management, Documentation, Security

Keywords
Security, Requirements, Testing, Security Requirements Based Testing, Later Stages Support, Usable, Cyber Security

1. INTRODUCTION
The need for software security, especially cyber security, is increasing in its prevalence and therefore in its importance. Cyber attacks, such as denial of service, eavesdropping, and evil twin, are more frequent, widespread and harmful than ever before. While many approaches to combat and cope with cyber security breaches have been proposed, few have allowed for highly usable and testable specification of cyber security requirements [1].

While security risks continue to evolve daily, security requirements engineering (SRE) techniques that specify the software that will be under risk struggle to cope with this evolution. As supported by [2], good requirements specifications are necessary for addressing security. Cyber security requirements would benefit tremendously from a proper SRE approach that supports them beyond specification [3].

CSIIRW '09, April 13-15, Oak Ridge, Tennessee, USA

In this paper we lead the reader through an example usage-scenario for an online-banking system, called ROBS (Requirements for Online Banking), that demonstrates a novel approach to SRE. Our approach is called SURE (Secure and Usable Requirements Engineering); it increases the usability of security requirements specifications by supporting the derivation of testing artifacts from them.

1.1 Prior Work
Cyber security needs arise when stakeholders determine that some resource belonging to a software system, tangible (e.g. money) or intangible (e.g. confidential information), is valuable to the organization. Such resources are called assets [4, 5], and SRE is focused on the protection of these valuable assets from the requirements perspective. During previous research [1], we investigated over 30 approaches to SRE and examined in detail 12. Results from this research showed us that there is a need for a new SRE approach that supports not only the specification of security requirements, but also aids in increasing their usability during later stages of development.

2. ROBS IN DETAIL
Our example begins with Developer Dennis, who decides to use the SURE technique to specify the security requirements for the document access and sharing feature of the online banking system his company is developing. Developer Dennis is,

- Familiar with traditional RE techniques like narrative text and shall statements
- Security specifications novices
- Knowledgeable of basic understanding of security terms

The characteristics described above represent not only Developer Dennis, but also the target group of users that we aim to support with the SURE technique.

Furthermore, SURE supports a specific type of cyber security, Virtual Enterprise Security. This type of security refers to the measures an organization has in place in order to secure processes, transactions, and access to information over a network [6]. In order to increase the usability of cyber security requirements, you have to ensure that they are developed so that other artifacts can be extended, derived, and/or mapped from them. SURE aids in developing security requirements so that testing artifacts can be easily derived from them.

Below we describe the steps Developer Dennis takes in order to specify ROBS using the SURE technique; there are two main areas of support, security requirements and security testing. Figure 1 shows the overall series of steps involved in this process.

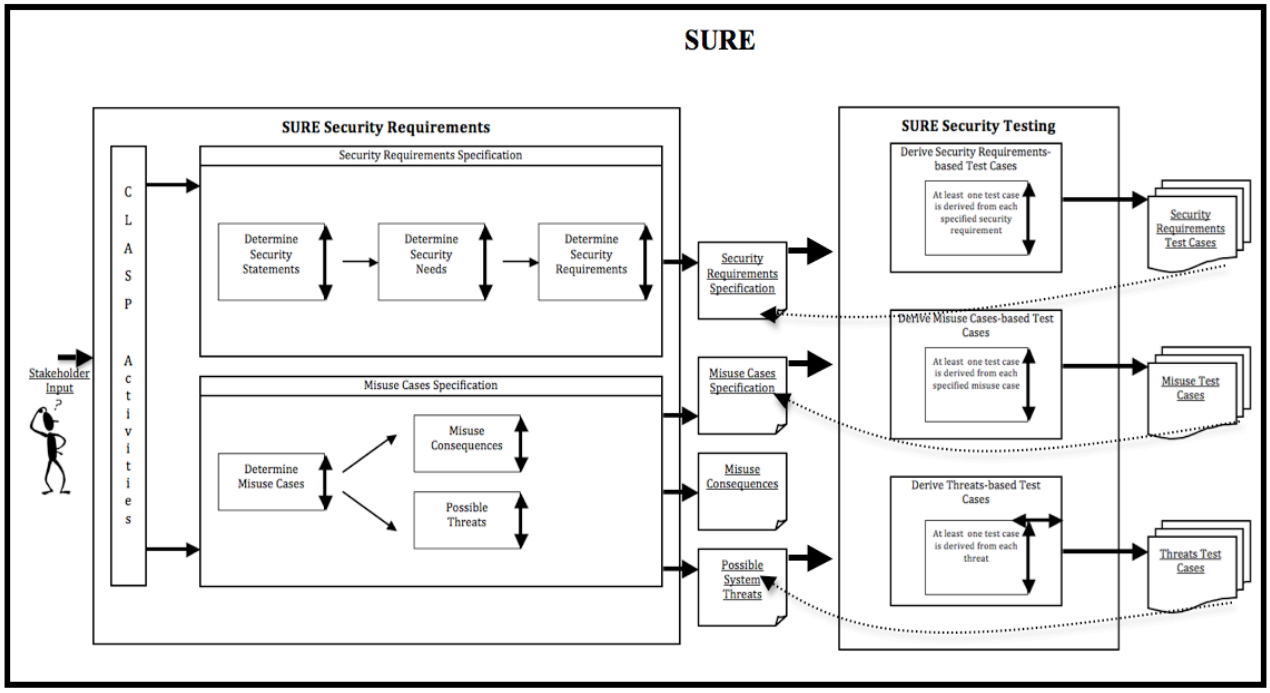


Figure 1. Steps in the ROBS Scenario

2.1 Security Requirements

The first area Developer Dennis needs to consider is developing a security requirements specification. SURE supports this development as a three-step process that produces a variety of artifacts. Each step is influenced by each of the approaches that SURE combines CLASP [7], USeR [8], and Misuse Cases [9]

2.1.1 CLASP

From the CLASP approach SURE makes usage of the concept of activities. A CLASP Activity is like a step that is necessary for that specific project. There are 30 possible CLASP activities that SURE customizes depending on a specific project. The SURE technique provides Developer Dennis with a questionnaire that helps him determine the activities needed. Some of the CLASP activities that Developer Dennis chooses for his project are shown in Figure 2.

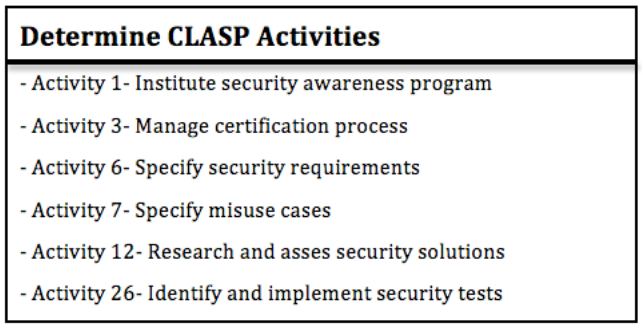


Figure 2. Step 1 of ROBS Scenario

At this point Developer Dennis has determined the CLASP activities necessary to successfully specify his security requirements; as mentioned, these activities will serve as a checklist for tasks that need to be accomplished. The next step is to specify the actual security requirements.

2.1.2 USeR

For this step we use the idea of refinement as a three-step process as shown in Figure 3 below.

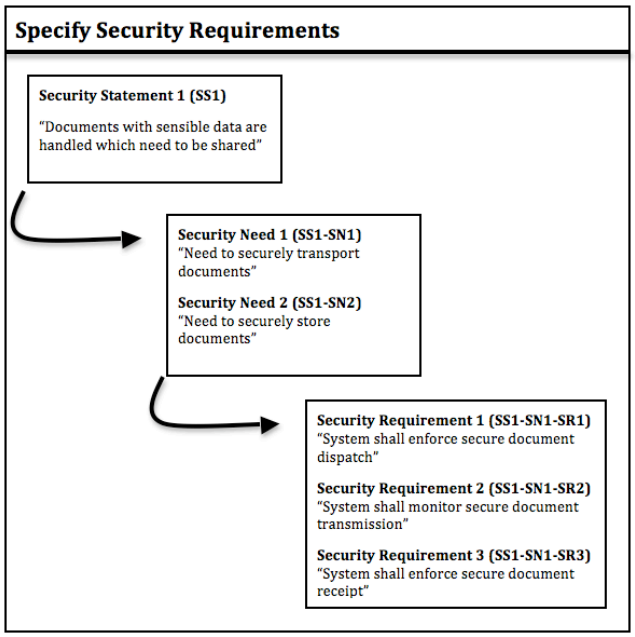


Figure 3. Step 2 of ROBS Scenario

The USeR method proposed the concept of "security statements," which are high-level ideas of the security intensive aspects of the system. Security statements facilitate the specification of security requirements for developers, as they do not need to know details of the security aspects, nor be experts in security, to initiate the specification. One of Developer Dennis' security statements is "documents containing sensible data need to be handled and shared", let's call this SS1. Figure 3 below shows the refinement process for this specific security statement.

Security statements are further refined it into security needs; there can be one or more security needs per statement. In the case of SS1 some of the security needs we can refine are SN1 "need to securely transport documents" and SN2 "need to securely store documents." The last step of the security specification is to extract security requirements from the security needs one or more security requirements can fulfill each security need.

Let's take SN1, some of the security requirements that can be extracted from it include SR1 "system shall enforce secure document dispatch," SR2 "system shall enforce secure document transmission," and SR3 "system shall enforce secure document receipt." As in traditional RE, at this stage we are not concerned with "how" each security requirement will be implemented. At the end of the security requirements specification we have a set of specifications, which have been refined from the original security statements that were inputted by the developer as well as cross-checked by both the expert and the customer. The resulting artifact for the second step in the development process is a set of security requirements that have been refined from high-level security statements.

2.1.3 Misuse Cases

The last step in the security requirements part of SURE is the specification of Misuse Cases. Misuse Cases are like traditional Use Cases except that they are created from the point of view of the attacker. SURE supports Developer Dennis in specifying situations where the system can be misused/abused through a three-step process. This three-step process is shown in Figure 4.

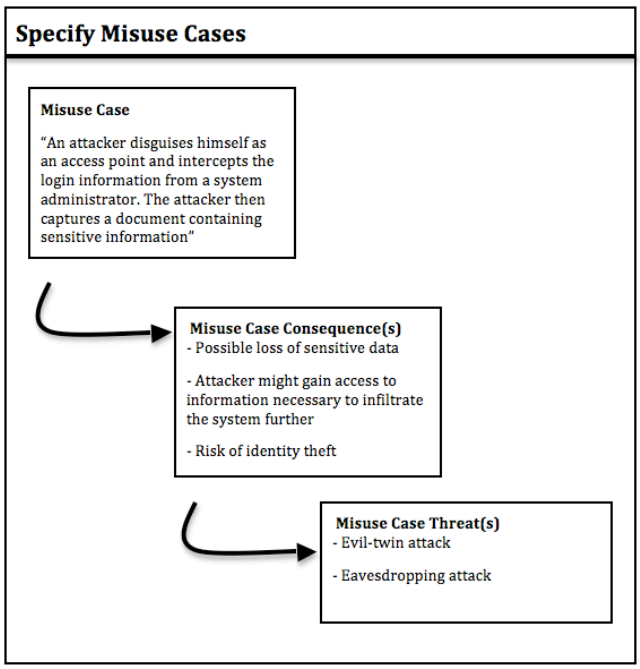


Figure 4. Step 3 of ROBS Scenario

The first step is to state the misuse cases; again since misuse cases are derived from use cases, they are fairly easy to be stated by the developer. In the case of Developer Dennis, one of his misuse cases states "An attacker disguises himself as an access point and intercepts the login information from a system administrator. The attacker then captures a document containing sensitive information," let's call it MC1.

The next step is to derive possible consequences from the stated misuse cases; these consequences refer to the "worst case" scenarios that the system as a whole could face if the misuse case was to come true. This helps the stakeholders involved in evaluating early on the consequences that certain parts of the system could produce if they are not secured properly. There can be one or more consequences to each misuse case; for MC1 some of the consequences include MCC1 "Possible loss of sensitive data," MCC2 "Attacker might gain access to information necessary to infiltrate the system further." The last step of the misuse case specification is to determine some of the possible attacks/threats that the system could face given the stated misuse cases. We offer seven types of threats that developers can choose from, Exploits, Eavesdropping, Denial of Service, Social Engineering, Indirect Attacks, Backdoor Attacks, Direct Access Attacks. At the end of the Misuse Case Specification step we have three types of artifacts, misuse cases, misuse case consequences, and possible threats to the system.

In addition to supporting the specification of different situations where the system can be misused, Developer Dennis now also has consequences that derive from this misuse as well as possible threats that the system can encounter once deployed.

At the end of the security requirements stage, Developer Dennis has produced a variety of artifacts ranging from security requirements to possible system threats. The support that the SURE technique offers does not stop here though, it supports these artifacts into testing as we will see in the following section.

2.2 Security Testing

Developer Dennis determines that while specifying ROBS has been done successfully, he is now interested in going beyond just the specification. The second area that the SURE technique support is security testing. This support increases the usability of the security requirements artifacts produced in the first stage. During the testing stage, we make use of the different artifacts developed during the security requirements process (i.e. security requirements specification, misuse cases, and threats) in order to derive test cases from them. Below we describe and exemplify using Developer Dennis how a variety of test cases can be derived from security requirements artifacts using the SURE technique.

2.2.1 Security Requirements Test Cases

The first type of test cases that the SURE technique helps Developer Dennis derive from the security requirements artifacts are test cases that come directly from the security requirements themselves. The SURE technique requires that at least one test case (in reality more than one) be derived from each of the security requirements specified. In the case of Developer Dennis, figure 3 below shows four test cases that can be derived from the original security requirements specified.

Security Requirements-Based Test Cases	
ID: SRT1 **Title**: Document Dispatch **Steps**: Select a document/Select recipient/ Click send **Input**: Predefined document **E. Output**: Successful sending of document **Sec. Requ.**: SR1	**ID**: SRT2 **Title**: Document Dispatch-Unauthorized recipient **Steps**: Select a document /Select recipient with incorrect security level/ Click send **Input**: Predefined document **E. Output**: System error; recipient selected has no sufficient security privileges
ID: SRT3 **Title**: Document Transmission **Steps**: Select a document/Select recipient/ Click send/ Inspect received document **Input**: Predefined document **E. Output**: Document transmitted has no changes **Sec. Requ.**: SR2	**ID**: SRT4 **Title**: Document Receipt **Steps**: Open document inbox/ Select document/ Open document **Input**: Previously sent document **E. Output**: Document asks for password before displaying **Sec. Requ.**: SR3

Figure 3. Security Requirements-based Test Cases

2.2.2 Misuse Cases Test Cases

The SURE technique also assists in deriving test cases from the specified Misuse Cases. The technique helps Developer Dennis break down the Misuse Case into smaller pieces using keywords; these smaller pieces then become individual test cases. Figure 4 shows some test cases that Developer Dennis can derive from the specified Misuse Cases during the requirements stage

Misuse Case-Based Test Cases	
ID: MCT1 **Title**: Encryption Quality **Steps**: From inbox select a document/Open document/Determine encryption level **Input**: Predefined document **E. Output**: Document encryption level that meets or exceeds requirements **MC.**: MC1	**ID**: MCT2 **Title**: Login Algorithm **Steps**: Select preferences/Open login options/Select run algorithm **Input**: Login system algorithm **E. Output**: Login encryption level that meets or exceeds requirements **MC.**: MC1

Figure 4. Misuse Case-based Test Cases

2.2.3 Threat Test Cases

The last type of test case that the SURE technique aids in deriving is threat-based test cases. We provide developers with a collection of not only possible attacks/threats but also typical test cases that are geared towards those specific threats/attacks. In the case of Developer Dennis, the SURE technique assists him in deriving the test cases shown in figure 5 from the original threats.

Threat-Based Test Cases	
ID: TT1 **Title**: Password renewal timing **Steps**: Click preferences/Select settings/Click login **Input**: None **E. Output**: Password renewal timing should be less than or equal to 15 days **Threat**: Evil-twin Attack	**ID**: TT2 **Title**: Valid user connectivity **Steps**: Click preferences/Select settings/Click login **Input**: Security question **E. Output**: Valid users will respond to the question asked by the system to validate their connection **Threat**: Eavesdropping Attack

Figure 5. Threat-based Test Cases

3. CONCLUSIONS AND FUTURE WORK

Security has become a system property that is not only sought after more than before, but also truly needed. In this paper we exemplified SURE, a new technique for specifying security requirements as well as deriving testing artifacts from them. We showed how Developer Dennis was able to, step by step, specify security requirements artifacts for his specific ROBS project. Once he had specified the security requirements, Developer Dennis was able to use SURE's support in deriving testing artifacts from the specified security requirements.

The current state of SURE is that the technique itself is being refined and the implementation of the SURE system, which is a web-based application, is about 70% completed. Future work will include the refinement of the technique as well as completion of the implementation so that we can start the evaluation and validation; these two will be done through a task analysis as well as a comparative study to determine the benefits of SURE to the SRE community.

4. REFERENCES

[1] Romero-Mariona, J., Ziv, H., Richardson, D.: Security Requirements Engineering: A Survey. Technical Report UCI-ISR-08-2. University of California, Irvine. 2008

[2] Hassan, R., Bohner, S., El-Kassas, S.: Formal Derivation of Security Design Specifications from Security Requirements. Workshop on Cyber security and information intelligence research. 2008

[3] Redwine, S. et al.: Processes to Produce Secure Software: Towards More Secure Software. National Cyber Security Summit. 2004

[4] Chivers, H. and Fletcher, M.: Applying Security Design Analysis to a service based system. Software: Practice and Experience, vol. 35 no. 9. 2005

[5] ISO/IEC.: Information Technology - Security Techniques - Evaluation Criteria for IT Security - Part 1: Introduction and General Model. ISO/IEC. International Standard 15408-1. 1999

[6] Allen, J.: Governing for Enterprise Security. Technical Note CMU/SEI-2005-TN-023. 2005

[7] Viega, J.: Building Security Requirements with CLASP. Proceedings of the Workshop on Software Engineering for Secure Systems (SESS). 2005

[8] Hallberg, N., Hallberg, J.: The Usage-Centric Security Requirements Engineering (USeR) Method. Information Assurance Workshop. 2006

[9] Jacobson, I. et al.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley. 1992

Towards Usable Cyber Security Requirements

Jose Romero-Mariona- jromerom@uci.edu
Hadar Ziv- ziv@ics.uci.edu
Debra J. Richardson- djr@ics.uci.edu
Dennis Bystritsky- dbystrit@uci.edu

Donald Bren School of Information and Computer Sciences
University of California, Irvine

Outline

- Motivation
- Background
- ROBS in Detail
- SURE Technique
- SURE Requirements
- SURE Testing
- Conclusions
- Future Work

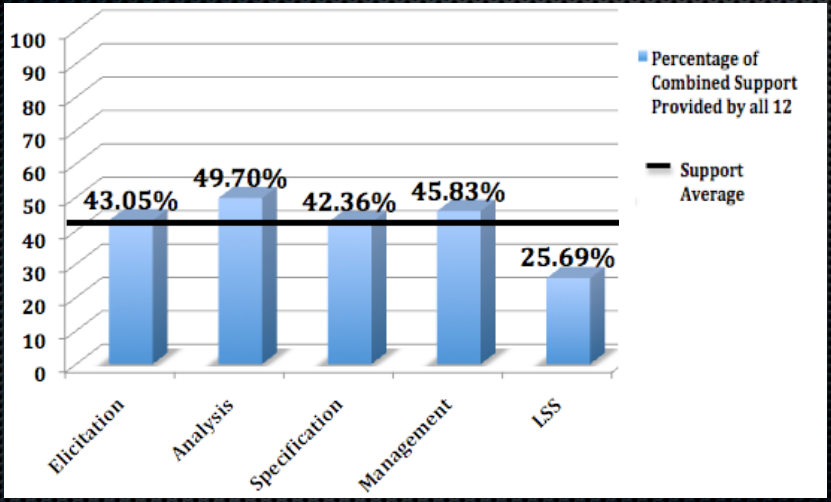
Motivation

- Security risks are increasingly more frequent, widespread, and volatile
- Often, the need for security is realized too late in the software development process
- Traditional Requirements Engineering (RE) approaches are not well suited for security-focused systems
- Specialized approaches to Security Requirements Engineering (SRE) are still in their infancy
- Support for later stages (LSS) of development is especially lacking

Background

- Literature Survey:
  - **12 approaches**
  - **5 SRE phases**
  - **34 questions total**
- Results:
  - Need for a SRE approach that provides **Later Stages Support**
  - Current approaches lack meaningful **testing support**
  - Top approaches
    - Original: CLASP and USeR
    - Derived: Misuse Cases

Background



ROBS in Detail

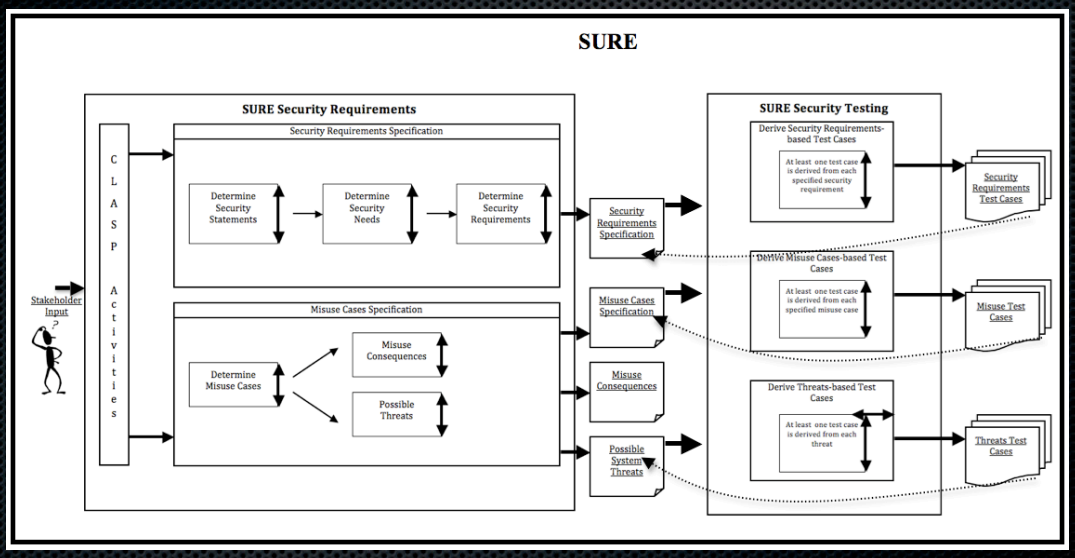
- ROBS- Requirements for an Online Banking System
- Example scenario for application of SURE
  - Secure and Usable Requirements Engineering
  - New SRE approach
  - Aids in deriving testing artifacts from security requirements

ROBS in Detail

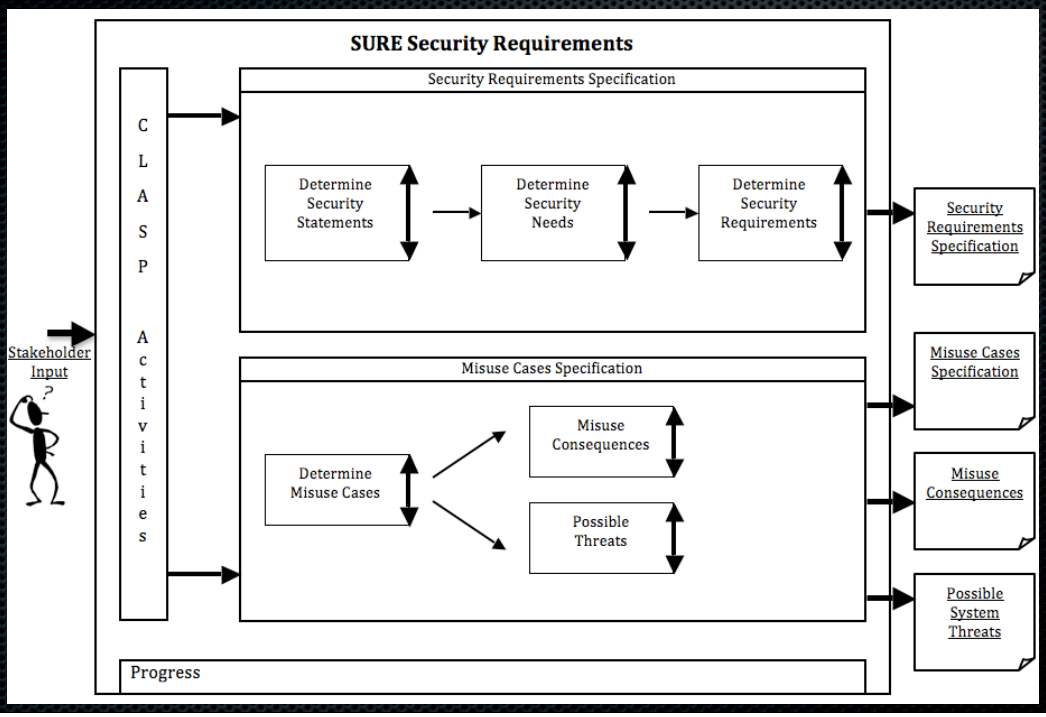
SURE- User Analysis

- Software Developer- Target user
- Assumptions
  - Familiar with traditional RE methods like narrative text, shall statements, and use cases
  - Security specification novice- Some knowledge about specifications in general, but needs support
  - Familiar with basic security concepts like threats and attacks
  - Software testing basics- test plan, test cases, etc.

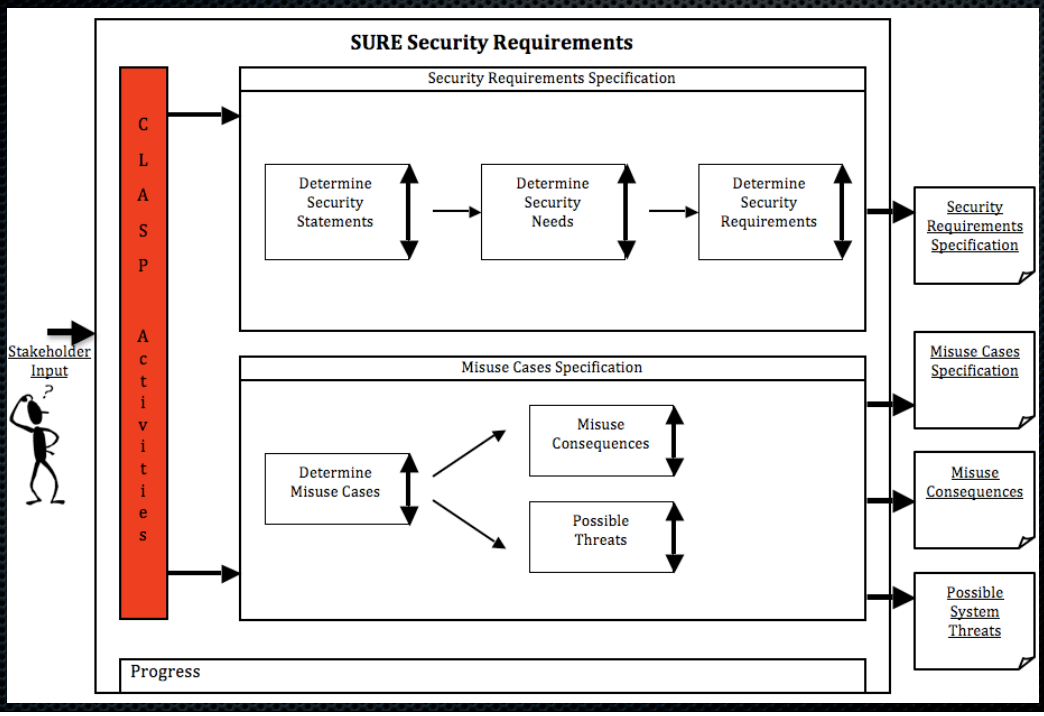
SURE Technique



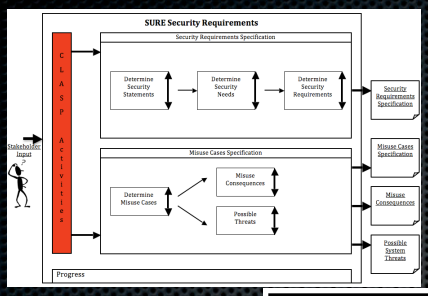
SURE Technique- Requirements



SURE Technique- Requirements



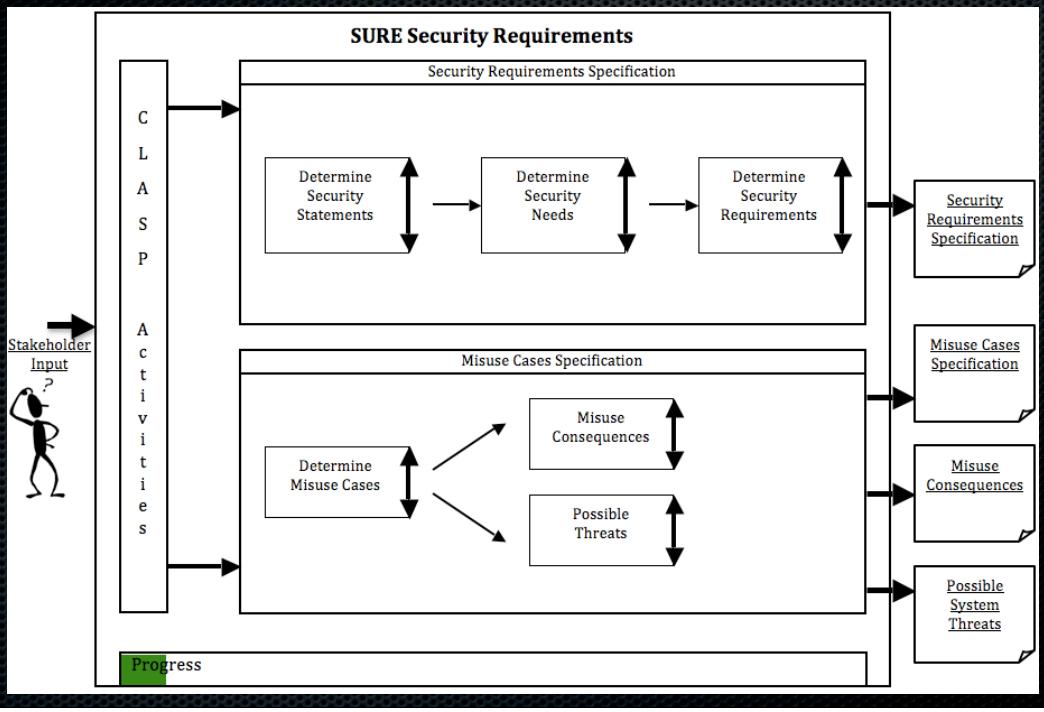
SURE Technique- Requirements



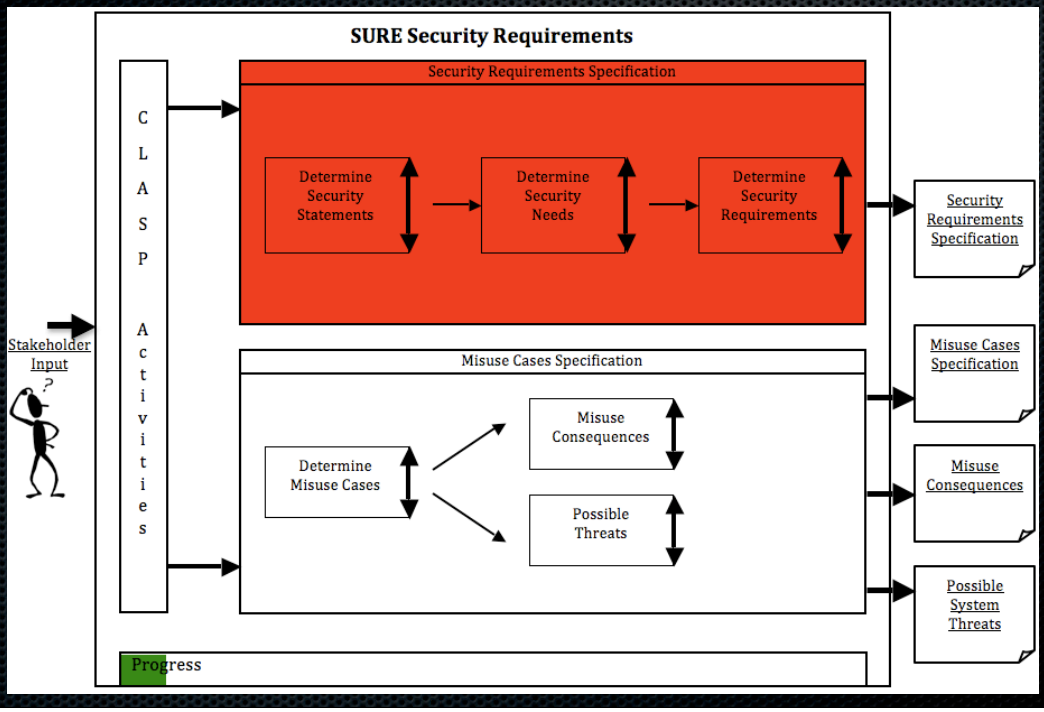
Determine CLASP Activities

- Activity 1- Institute security awareness program
- Activity 3- Manage certification process
- Activity 6- Specify security requirements
- Activity 7- Specify misuse cases
- Activity 12- Research and asses security solutions
- Activity 26- Identify and implement security tests

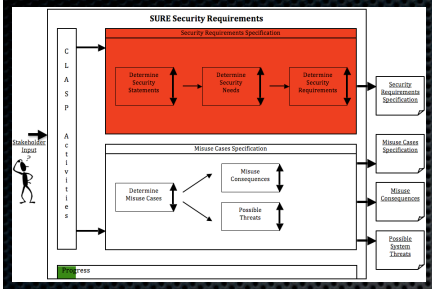
SURE Technique- Requirements



SURE Technique- Requirements



SURE Technique- Requirements



Specify Security Requirements

Security Statement 1 (SS1)

"Documents with sensible data are handled which need to be shared"

Security Need 1 (SS1-SN1)
"Need to securely transport documents"

Security Need 2 (SS1-SN2)
"Need to securely store documents"

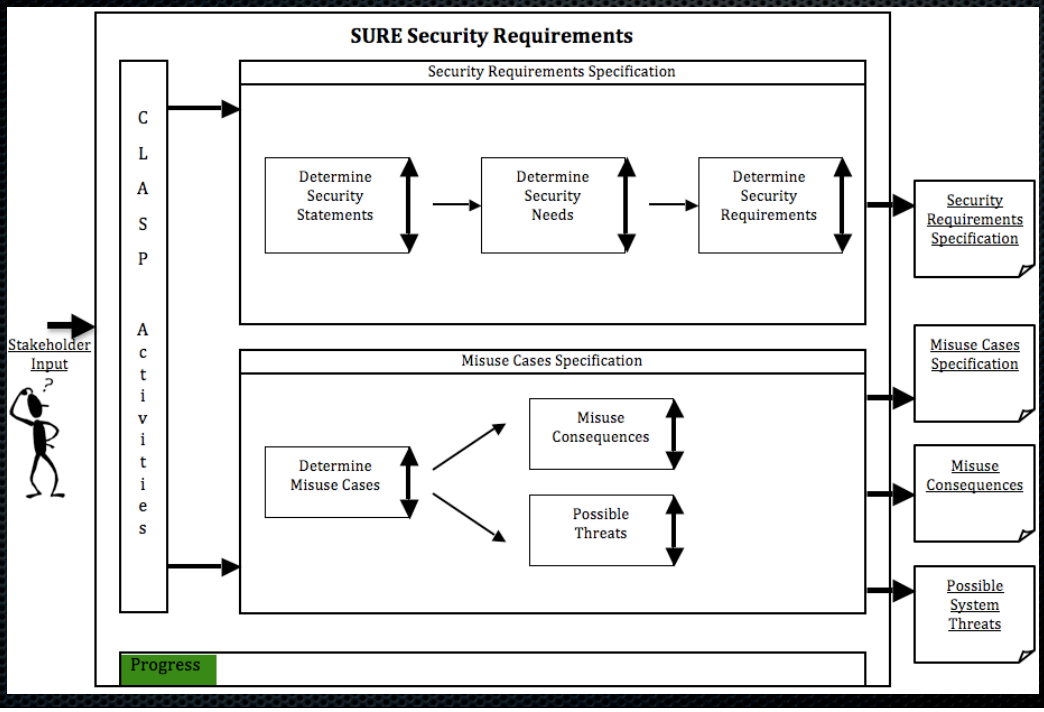
Security Requirement 1 (SS1-SN1-SR1)
"System shall enforce secure document dispatch"

Security Requirement 2 (SS1-SN1-SR2)
"System shall monitor secure document transmission"

Security Requirement 3 (SS1-SN1-SR3)
"System shall enforce secure document receipt"

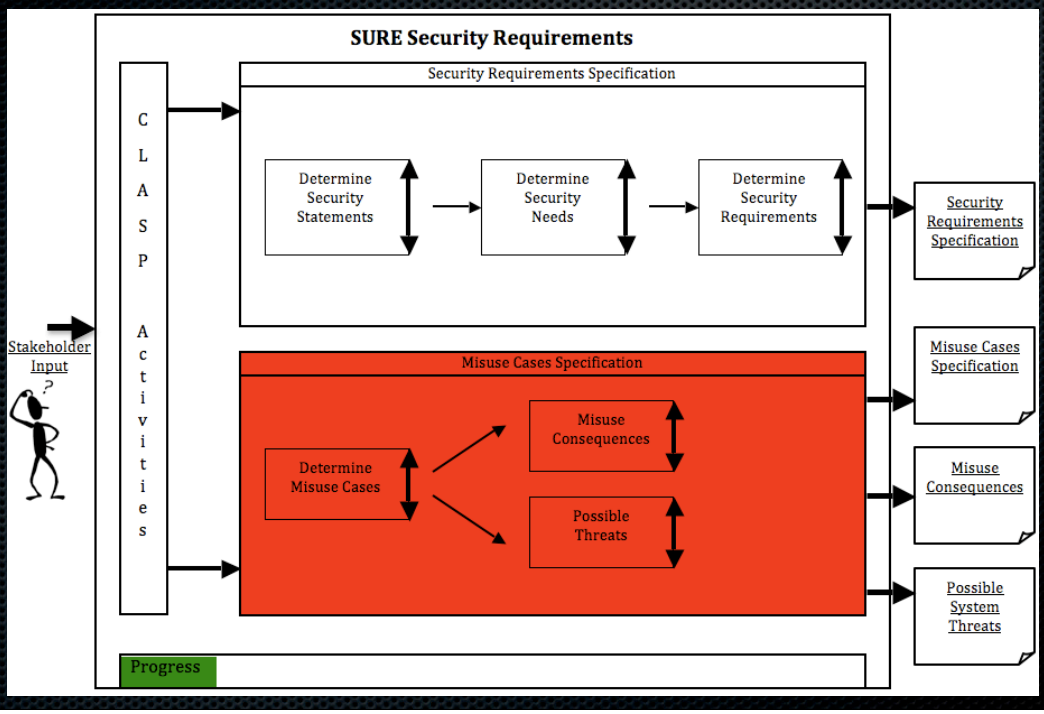
8

SURE Technique- Requirements

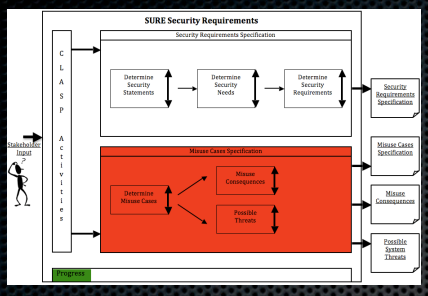


8

SURE Technique- Requirements



SURE Technique- Requirements



Specify Misuse Cases

Misuse Case

"An attacker disguises himself as an access point and intercepts the login information from a system administrator. The attacker then captures a document containing sensitive information"

Misuse Case Consequence(s)
- Possible loss of sensitive data
- Attacker might gain access to information necessary to infiltrate the system further
- Risk of identity theft

Misuse Case Threat(s)
- Evil-twin attack
- Eavesdropping attack

SURE Technique- Testing

- SRBT foundation
  - Set of 4 security requirements based testing principles
  - 12-step process for SRBT
- Supports
  - Test completion criteria
  - Test case design
  - Test case specification

Conclusions and Future Work

- SURE- New SRE approach
- Supports security requirements specification
- Facilitates test artifact derivation from security requirements
- Implementation- SURE System
- Technique refinement
- Technique/System evaluation