The Journal of Systems and Software 106 (2015) 102–116

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Automated analysis of security requirements through risk-based argumentation

Yijun Yu [a,*], Virginia N.L. Franqueira [b], Thein Than Tun [a], Roel J. Wieringa [c], Bashar Nuseibeh [a,d]

[a] The Open University, Milton Keynes, UK
[b] University of Derby, Derby, UK
[c] University of Twente, Enschede, The Netherlands
[d] Lero the Irish Software Engineering Research Centre, University of Limerick, Ireland

ARTICLE INFO

Article history:
Received 14 July 2014
Revised 13 April 2015
Accepted 17 April 2015
Available online 24 April 2015

Keywords:
Structured argumentation
Risk assessment
Security analysis

ABSTRACT

Computer-based systems are increasingly being exposed to evolving security threats, which often reveal new vulnerabilities. A formal analysis of the evolving threats is difficult due to a number of practical considerations such as incomplete knowledge about the design, limited information about attacks, and constraints on organisational resources. In our earlier work on RISA (RIsk assessment in Security Argumentation), we showed that informal risk assessment can complement the formal analysis of security requirements. In this paper, we integrate the formal and informal assessment of security by proposing a unified meta-model and an automated tool for supporting security argumentation called OpenRISA. Using a uniform representation of risks and arguments, our automated checking of formal arguments can identify relevant risks as rebuttals to those arguments, and identify mitigations from publicly available security catalogues when possible. As a result, security engineers are able to make informed and traceable decisions about the security of their computer-based systems. The application of OpenRISA is illustrated with examples from a PIN Entry Device case study.

1. Introduction

Security risks evolve in software-intensive systems. Attackers exploit increasing number of vulnerabilities, ranging from cryptographic protocols to human subjects. Introducing new technologies to such systems often imposes security risks with higher likelihood to do harm to the assets. In practice, security is not perfect due to limited resources available to security engineers, uncertainties about the attackers' skills and commitment, and incomplete knowledge about evolving threats and vulnerabilities.

Recent years found structured argumentation approaches effective to build safety cases (Kelly, 1998) and to reason about both formal and informal descriptions of software systems, to demonstrate compliance to laws and regulations (Burgemeestre et al., 2010; Cyra and Górski, 2007), to trace and justify software design decisions (Potts and Bruns, 1988), to establish confidence in software development (Graydon and Knight, 2008), and to build dependability cases to assure compliance in software development (Huhn and Zechner, 2010).

Extending the work on security argumentation (Haley et al., 2008), we have developed a framework for reasoning about security requirements of the system where abstract properties are important. For instance, it is possible to formally prove that an access control model will deny access to the Human Resource (HR) database by those who do not work in the HR department.

However, real-life phenomena could defy generalisation and abstraction, there the framework needs to support the uses of informal arguments. For instance, many HR employees could share a common password, and when one of the employees leaves the department and the common password is not changed, thus access becomes available to someone who is no longer a member of the HR department.

Through the use of RIsk assessment in Security Argumentation (RISA) method (Franqueira et al., 2011), we have shown how risk assessments iteratively challenge the satisfaction of security requirements. The main limitation of our previous work lies in that the separate models for formal arguments and risk-based arguments, which hinders the automated tool support.

In this work, this limitation is addressed by the means of three contributions of the RISA method. First, we introduce an integrated modelling language to represent risk assessment and arguments uniformly. Second, the OpenRISA tool support extends the OpenArgue (Yu et al., 2011) argumentation tool to perform automated checking

* Corresponding author. Tel.: +44 1908655562.
E-mail address: y.yu@open.ac.uk (Y. Yu).

http://dx.doi.org/10.1016/j.jss.2015.04.065
0164-1212/

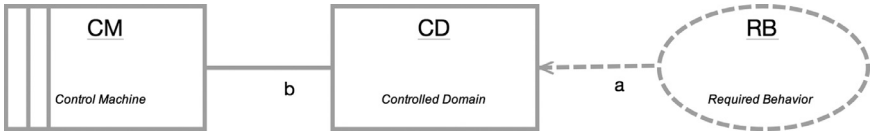

Fig. 1. Syntactical elements of a problem diagram are illustrated by the "Required Behaviour" problem frame. A requirement is shown as a dotted oval, and a specification is shown as a rectangle with double vertical strips on the left. Problem world domains are shown as rectangles. The letters on solid edges indicate the interfaces between the domain nodes. A dotted arrow indicates the phenomena of the domain constrained by the requirement. A dotted line (not shown in this diagram) indicates the phenomena of the domain referenced by the requirement. Abbreviations of the name of the nodes are optional.

of the formal arguments. Third, we incorporate an automated search functionality to match catalogues of security vulnerabilities such as CAPEC (Common Attack Pattern Enumeration and Classification patterns[1]) and CWE (Common Weakness Enumeration[2]) with the keywords derived from the arguments. Compared to the previously *ad hoc* search, the new tool supports a complete coverage of these public catalogues of security expertise.

The OpenRISA approach has presented a research contribution to represent and reason about risks associated with software security requirements. The argumentation part of the work has been evaluated with an industry evaluator at DeepBlue (Yu et al., 2011).

The remainder of the paper is organised as follows. Section 2 reviews relevant background on the satisfaction of security requirements and security arguments, whilst Section 3 reviews related work. Section 4 provides an overview of the RISA method, Section 5 describes the corresponding OpenRISA tool support. Section 6 demonstrates the tool supported method with a PIN Entry Device (PED) example. Section 7 discusses and points to future research and development work. Finally, Section 8 concludes.

2. Background

The RISA method builds on the notions of *satisfaction of security requirements*, and *outer* and *inner* arguments, introduced by Haley et al. (2008).

2.1. Requirements satisfaction arguments about the problem depicted in a problem diagram

Following the Problem Frames approach in requirements engineering (Jackson, 2001), software system artefacts are separated into *S*, *W*, and *R*, where *S* represents the specification of a software system, *W* represents a description of the world in which the software system is to be used (*i.e.*, the context), and *R* represents a description of the requirements. The software within the system context should satisfy the requirements. This semantics of a *requirements problem* can be described by the following entailment relationship:

$$W, S \vdash R \quad (1)$$

The world context *W* consists of *domains* (short for *problem world domains*); elements of the world can be either physical, such as people and hardware, or logical, such as software and data structure. Typically, *W* also contains assumptions made about these domains.

Using the Problem Frames approach (Jackson, 2001), the analysis of a requirement problem follows the principles of divide-and-conquer (*i.e.* decomposition) and unite-and-rule (*i.e.* composition). First of all, the knowledge of the physical domains in the context directly referenced and/or constrained by the requirement statements *R* are analysed, in order to bring in indirectly related domains to the analysis until a machine specification *S* is found. Around *S*, the collection of the domains in the physical world *W* are depicted in a diagram where nodes represent the domains and edges represent the phenomena shared between the domains. A shared phenomenon could be an event controlled by one domain and observed by another. As such, the so-called problem context diagram, illustrated in Fig. 1, captures the knowledge of high-level causality about the behaviour between these domains. The OpenRISA tool includes the components of OpenPF which support the creation of context diagrams.

2.2. Toulmin-structured arguments and argumentation processes

According to Haley et al. (2008), arguments for requirements satisfaction are in two kinds: the first argument constructed, called the outer argument, is a formal proof that relies on certain assumptions about the system context and the specification. The outer argument is formal in the sense that it typically uses a mathematical logic such as the prepositional logic. The assumptions made in the outer arguments are expanded in and supported by the informal inner argument which uses the structured natural language because those assumptions cannot be described in the formal logic.

Effective argumentation establishes a *claim* that one wishes to convince the world of. In other words, claim is the object of an argument, a statement that one wishes to convince an audience to accept. In an effective argument, ground truths or facts need to provide the underlying support for an argument, *e.g.*, evidence, facts, and common knowledge. In other words, *ground* is a piece of evidence, a fact, a theory, a phenomenon considered to be true. In addition, *warrants* connect and establish relevancy between the grounds and the claim. A warrant explains how the grounds relate to the claim, but not the validity of the grounds. Structurally, warrant is a statement that links a ground and a claim, showing how a claim is justified by ground(s).

Rebuttals express conditions under which an argument does not hold by invalidating any of the grounds and associated warrants, thus undercutting the support to the claim. Grounds and warrants may need to be argued, making them claims of nested arguments, therefore grounds and warrants can also be attacked by the rebuttals. Here, rebuttal is a counterargument which undermines the support for the claim. Specifically in the case of security-related argumentation, rebuttals represent risks. Rebuttals can be mitigated in order to restore the confidence that the claim of the rebutted argument still holds. A mitigation, while negating a rebuttal, introduces additional knowledge to show that the rebuttal, *i.e.* a risk, can somehow be tolerated. In doing so, a mitigation can also introduce new risks, leading to new iterations of rebuttals and mitigations in argumentation.

Mitigating a rebuttal requires an iterative process, which introduces additional arguments incrementally. The notion of *round* denotes the number of iterations from the beginning. Using the round numbers, cyclic arguments can be avoided by eliminating the redundant facts from the increments at different rounds. Specific to security satisfaction arguments, there are two types of rebuttals, risks and mitigations. In general the argumentation structure iteratively relates inner arguments of logical rebuttals to outer arguments of boundary expansions. Starting from the initial ground about the software system in question, every round of argumentation may introduce additional facts and/or enclose more elements into the system boundary, further the scope of the knowledge.

[1] http://capec.mitre.org/.
[2] http://cwe.mitre.org/.