Information and Software Technology 56 (2014) 294–308

Contents lists available at ScienceDirect

Information and Software Technology

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

Comparing attack trees and misuse cases in an industrial setting

Peter Karpati [a,1], Yonathan Redda [a,2], Andreas L. Opdahl [b,*], Guttorm Sindre [a]

[a] Dept. of Computer and Information Science, Norwegian University of Science and Technology, Sem Sælands vei 7-9, NO-7491 Trondheim, Norway
[b] Dept. of Information Science and Media Studies, University of Bergen, P.O. Box 7802, NO-5020 Bergen, Norway

ARTICLE INFO

Article history:
Received 19 April 2013
Received in revised form 28 October 2013
Accepted 28 October 2013
Available online 5 November 2013

Keywords:
Security requirements
Requirements modelling
Misuse cases
Attack trees
Industrial experiment

ABSTRACT

The last decade has seen an increasing focus on addressing security already during the earliest stages of system development, such as requirements determination. *Attack trees* and *misuse cases* are established techniques for representing security threats along with their potential mitigations. Previous work has compared attack trees and misuse cases in two experiments with students. The present paper instead presents an experiment where industrial practitioners perform the experimental tasks in their workplace. The industrial experiment confirms a central finding from the student experiments: that attack trees tend to help identifying more threats than misuse cases. It also presents a new result: that misuse cases tend to encourage identification of threats associated with earlier development stages than attack trees. The two techniques should therefore be considered complementary and should be used together in practical requirements work.

1. Introduction

The last decade has seen an increasing focus on addressing security already in the earliest stages of system development [50,22], among other things because delaying security engineering until later stages can often cause costly redesign [35]. Several techniques have been proposed for analysis of security threats to support requirements elicitation. Two of the more well-known are *attack trees* [60] and *misuse cases* [67], both of which can be used in either textual and diagrammatic forms. They have several things in common. Both are fairly simple and thus potentially usable by stakeholders with limited modelling competence, and they both focus on *attacking behaviours*, i.e., on modelling what the attacker is trying to achieve, not only on the *security behaviours* that prevent attacks from happening. But there are also notable differences between them. Attack trees represent potential attacks (or *threats*) through *AND*- and *OR-decomposition* into more detailed attack steps, but do not focus on regular behaviour. Misuse cases represent the attacking behaviours (or *misuse cases*) of a *misuser* together with the system's normal functions (the *regular use cases*) [34], but focus less on hierarchical decomposition of attacks (or threats) – unless a lot of *include*, *extend* and *generalization* relationships are used between the misuse cases [69]. Attack trees and misuse cases can therefore be seen as competing techniques, i.e., as alternative ways of satisfying the same modelling need. But they can also be seen as complementary, i.e., as techniques that can and should be used together to effectively provide a good picture of important threats and potential mitigations for a proposed system.

We therefore need a better understanding of the relative strengths and weaknesses of attack trees and misuse cases, in order to determine whether they are competing or complementary and to guide practical use and further research. Experimental comparisons of attack trees and misuse cases using students as participants have already been reported in Opdahl and Sindre [53], concluding that participants using attack trees found more security threats than the ones using misuse cases, in particular for authorisation and confidentiality threats, but that participant perception of the two techniques was similar. However, the experiments left several open questions, in particular whether the results of the comparison would have been similar in an industrial setting.

This paper therefore presents an experiment that compares attack trees and misuse cases with industrial practitioners working on security requirements for real systems. The purpose is manifold: Firstly, we want to enhance the general knowledge about the two techniques by providing new data from industry. Secondly, we want to support the industrial uptake of attack trees and misuse cases by investigating their use in an industrial setting. Thirdly, we want to investigate whether an industrial experiment like the one to be presented here produces results similar to the earlier stu-

* Corresponding author. Tel.: +47 5558 4140; fax: +47 5558 9149.
E-mail addresses: Peter.Karpati@hrp.no (P. Karpati), Yonathan.Redda@gmail.com (Y. Redda), Andreas.Opdahl@uib.no (A.L. Opdahl), Guttorm.Sindre@idi.ntnu.no (G. Sindre).
[1] Present address: Institute for Energy Technology, P.O. Box 173, NO-1751 Halden, Norway.
[2] Present address: P.O. Box 5364, Bethel, Addis Ababa, Ethiopia.

0950-5849/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2013.10.004

dent experiments in [53]. Compared to the earlier experiments, the present one (1) involves real system developers as subjects, (2) uses a real system as task instead of a simple paper task, (3) allows more time for solving the tasks and (4) characterises different types of threats in a new way. These and other differences will be elaborated in Section 3.10.

The new experiment retains the three main research questions from the earlier student experiments [53]:

- Which technique aids participants in finding *more threats* and *mitigations*? (RQ1)
- Which *types of threats* does each technique encourage participants to find? (RQ2)
- Which of the two techniques do the participants *perceive* most favourably? (RQ3)

In addition, the new experiment also asks:

- Will an experiment with practitioners working on a real system *give results similar to* the earlier experiments with students working on simple paper tasks? (RQ4)

The rest of the paper is organised as follows: Section 2 reviews related work. Section 3 explains the research method, before Section 4 presents the experiment results. Section 5 discusses the results and their implications for practice and further research, as well as threats to validity. Finally, section 6 concludes the paper.

2. Related work

2.1. Attack trees

Attack trees (AT) offer a structured way for investigating and describing a security attack or threat [61]. The high-level attack is the root node of the tree and is recursively decomposed through AND/OR branches into increasingly fine-grained attacking steps, certain combinations of which must succeed to for the high-level attack to succeed. A diagrammatic notation is provided for visualising attack trees. For an example of an AT diagram, see [60].

Nodes in the AT can have values in order to answer questions like "Which is the cheapest attack?" or "Which is the best low-risk, low-skill attack?" [60]. They can be used to evaluate proposed designs but are also applicable at an early requirements stage. Attack trees are intuitive and many extensions have appeared, such as defense trees [5], protection trees [19], attack response trees [79], attack countermeasure trees [59] and unified parameterisable attack trees [75]. Gegick and Williams [25] presents a case study indicating that attack trees could be useful to achieve early focus on security concerns. The technique has also been used in a number of experiments [41,43,9], but not all of them focussed specifically on evaluating the AT technique.

2.2. Misuse cases

Misuse cases (MUC) adapt regular use cases (UC) for security purposes by extending them with misusers, misuse cases and mitigation use cases [67]. Specifically, *misusers* are added in addition to regular use-case actors to represent attackers, i.e., people, organisations or software agents that may try to violate the security of the system-to-be. Misusers try to effect *misuse cases*, which *threaten* regular use cases with unwanted system behaviours. Misuse cases can in turn be *mitigated* by other use cases [1], which can be called *security use cases* [22]. A diagrammatic notation is provided for misuse cases based on use-case diagrams, using inverted (filled, as opposed to open) symbols for misusers and misuse cases. For an example of a misuse-case diagram, see [67].

The EU project CORAS [18] combined misuse cases with UML-based techniques into a comprehensive method for secure systems development. Herrmann et al. [29] proposes RiskREP, which extends misuse case-based methods with ICT-architecture based risk assessment. By linking countermeasures (mitigations) to both business goals and costs, the countermeasures can then be prioritised and selected by cost-effectiveness. MUCs have also been investigated for safety [66,71,72] and other system dependability threats [68]. *Abuse cases* introduce similar concepts to misuse cases, but do not show use and misuse cases in the same diagram [50]. Abuse and misuse cases both represent negative behaviours that potential attackers want to perform using the system, whereas security use cases represent countermeasures to avoid or repel these attacks [22].

2.3. Comparisons of AT and MUC

Several authors have proposed to combine AT with MUC on the assumption that the two have complementing features, such as Suraksha [49], Hybrid Technique [23], SHIELDS [74] and HARM [36]. [17] compares AT, Common Criteria (CC) and MUC by using all three techniques on a wireless hotspot case, finding that AT and MUC were easier to learn and use than CC; that AT and CC provided clearer models, perhaps in particular for large systems; and that CC and MUC were easier to analyse than AT. The authors suggest that combining the three techniques might therefore be a good idea.

Opdahl and Sindre [53] compares attack trees and misuse cases in two controlled student experiments with 28 and 35 participants, in order to investigate (1) which technique aids participants in finding *more threats* and *mitigations*, (2) which *types of threats* and *mitigations* does each technique aid participants in finding and (3) which of the two techniques do the participants *perceive as preferable*? In each experiment, the participants solved two threat identification tasks individually by means of the two techniques, using a crossover Latin-Squares design to control for technique and task order [27]. (We will explain this type of experimental setup in more detail in Section3.3.) Three types of dependent variables were measured: *effectiveness of the techniques* was measured as the number of threats and mitigations found; *coverage of the techniques* was measured in terms of the types of threats found (e.g., whether *authorisation*, *integrity*, *confidentiality*, *availability* and *physical*); and *perception of the techniques* was measured through a post-task questionnaire inspired by the Technology Acceptance Model [14]. The only difference was that, in the second experiment, but not in the first, a pre-drawn use-case diagram was provided to the participants to use as a starting point for solving the tasks. The main finding was that attack trees were more effective for finding threats, in particular when there was no pre-drawn use-case diagram and, in particular, authorisation and confidentiality threats. However, the participants had similar opinions of the two techniques, and perception of a technique was not correlated with performance with that technique. The data also suggested that the two techniques might encourage modellers to identify *different types* of threats and mitigations, a difference we will explore further in the present paper.

2.4. Other security requirements techniques

Several authors have proposed extensions to the Unified Modeling Language (UML) [52] for modelling security-related concerns. *UMLsec* [35] defines stereotypes for integrated security related information in UML specifications, and *SecureUML* [46] supports development of secure, distributed systems based on role-based access control supported by authorisation constraints. UML can also be used to represent *security patterns*, which describe recommended