An extended misuse case notation: Including vulnerabilities and the insider threat

Lillian Røstad[1]

Norwegian University of Science and Technology, Trondheim, Norway
Lillian.Rostad@idi.ntnu.no

Abstract. Misuse cases are a useful technique for eliciting and modelling security requirements and threats. In addition they may be very useful in a risk analysis process, particularly as part of the system development process. The original misuse case notation adds inverted use cases to model threats and inverted actors to represent attackers. However, an attack is usually performed by exploiting a vulnerability in a system and it would be useful to be able to represent vulnerable functions in a model. In addition, it should be possible to discern between insiders and outside attackers in a model, as they have very different abilities and potential for attacking a system. This paper therefore proposes an extended misuse case notation that includes the ability to represent vulnerabilities and the insider threat, and discusses the use of this extended notation in the system development and risk analysis processes.

1 Introduction

Security is being increasingly recognized as an important quality of IT-systems. Much of the reason for this can be explained by the evolution of IT-systems towards what Gary McGraw in [13] defines as the *trinity of trouble*: connectivity, extensibility and complexity. While these three properties typically improves the possibilities of what a system can do, they also significantly increases the risks. Being secure means having control and being able to keep the bad guys out - but the more complex a system is the harder it is to manage, and the possibility of third-party extensions only adds to the complexity. Connectivity is seductive as it greatly increases the potential use of a system, but it also greatly increases the number of attackers that can have a go at breaking into or otherwise harm the system. In some systems, like health care, defence and banking, security has always been considered an important property. But as the system's operational environment changes, so does the threat scenarios and need for defence mechanisms. Where isolation previously has been considered an appropriate defence, this is no longer an option.

An excellent example of this, and the original motivation for the work presented here, is access control in healthcare systems. In healthcare systems protecting the patient's privacy is a major concern - however it always has to be balanced against the need for access to information to make sound medical decisions and provide the best possible care. The current state-of-the art is Role-Based

Access Control (RBAC) [8] and a role in existing systems is typically a rather static structure combined of a user's profession (doctor, nurse etc), place of work (ward) and where the patient is currently admitted (ward). There is currently a move towards making the systems more dynamic and user centric and enabling information sharing. As patients are able to select hospital or place of care more freely there is a need to be able to make a patient's medical information available to those providing care. This significantly adds to the complexity and changes the requirements for the access control mechanisms - static structures are no longer sufficient. Also, most existing systems include mechanisms that allow a user to override the access control mechanism in emergency situations. In such situations there is no time to register the patient at the correct ward to enable the normal access control mechanism to function. Emergency access control effectively constitutes a vulnerability in the system that may be exploited by insiders - that is; legitimate system users that may misuse the functionality. As systems become connected the user bases grow, thereby increasing the potential risk for exploitation.

To be able to design secure solutions in a changing threat scenario one needs to be able to perform risk analysis [21] based on system requirements and design [13]. UML use cases [1] have become a widely used technique for elicitation of functional requirements [7] when designing software systems. One of the main advantages of use cases is that they are easy to understand with only limited introduction to the notation, and therefore are a very well-suited tool for communicating and discussing requirements with system stakeholders. A use case model illustrates required usage of a system - i.e. expected functionality. In risk analysis it is equally important how one should *not* be able to use a system - i.e. potential threats and exploitation. Misuse cases [18] have been proposed as an approach to identifying threats and required countermeasures. The notation is very simple and complements the UML use case notation. However, the usability of the notation or the ability to give a more complete risk overview could be significantly improved by adding some minor extensions enabling the specification of vulnerabilities and the insider threat in misuse case models. The remainder of this paper presents such an extended misuse case notation and discusses potential use in system development and risk analysis.

2 Related work

The notation proposed here builds upon work done on how to utilize use cases as a tool for eliciting and modelling security requirements. John McDermott [11] and Chris Fox [12] used the term *abuse cases* in their approach where they explored how threats and countermeasures could be expressed using the standard UML use case notation. In their approach they kept the abuse cases in separate models.

Later, in a series of papers [15], [16], [17], [19], [14], [18], Guttorm Sindre and Andreas L. Opdahl have proposed, and elaborated on, the concept of *misuse cases* including both graphic and textual description. Misuse cases [18] extends

the UML use case notation by adding inverted use cases to model misuse and inverted actors to model attackers. Sindre and Opdahl [18] define *misuse cases* and *misusers* as:

- *Misuse case* - a sequence of actions, including variants, that a system or other entity can perform, interacting with misusers of the entity and causing harm to some stakeholder if the sequence is allowed to complete.
- *Misuser* - an actor that initiates misuse cases, either intentionally or inadvertently.

Misuse cases are created by extending a use case model and thus provide the ability to regard system functions and possible attacks in one coherent view. In the initial work on misuse cases two additional relationships were defined [15]: *prevents* and *detects* and it was pointed out that the UML use case relationships *include* and *extend* may also be used to connect misuse cases. They also pointed out that the *include*-relationship may be used between a misuse case and use case to illustrate that an attack utilizes system functionality. This in fact corresponds to exploiting a vulnerability, but they did not provide a tailored notation for this.

Ian Alexander has written several papers discussing misuse cases as a tool [6] [5] and experiences from application of misuse cases [2]. He has also discussed misuse cases in relation to goal-oriented requirements engineering [3] [4]. In this case Alexander stays true to the graphic notation of inverted use cases proposed by Sindre and Opdahl, but he defines four different relationships: *threatens*, *mitigates*, *aggravates* and *conflicts with*. It is interesting to note that in their latest (at the time of writing this paper) [18] publication on misuse cases, Sindre and Opdahl have refined the relationships in the misuse case notation adopting *threaten* and *mitigate* as suggested by Ian Alexander. By their definition a *use case mitigates misuse case* and *misuse case threaten use case*. Exchanging the *prevents* and *detects* with the softer *mitigate* makes sense as it is unlikely that any countermeasure applied will entirely eliminate a threat.

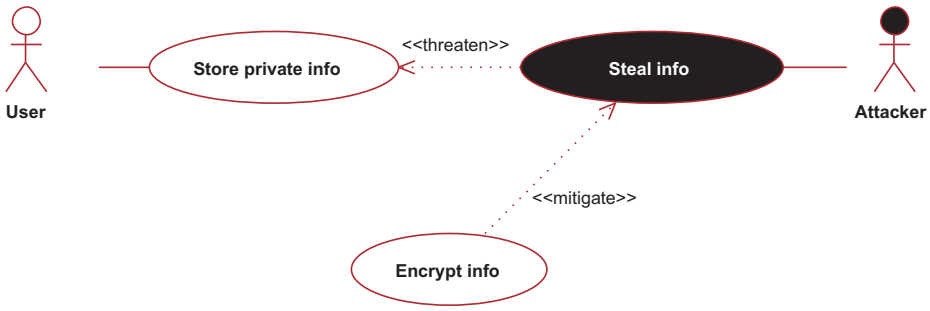

Fig. 1. Simple use and misuse case illustrating the notation

Donald G. Firesmith has discussed the concept of *security use cases* [9] where a security use case represents functionality needed to protect a systems assets

from identified threats. The idea of security use cases as a way of representing specific security functionality, or countermeasures, has been adopted by Sindre and Opdahl [18] and linked directly to the *mitigate* relationships. Security uses cases have not been given a specific graphical notation, but are represented as ordinary use cases in the models.

Figure 1 depicts a very simple misuse case that illustrates the current notation. In this figure *encrypt info* is a security use case added to protect against the threat (*steal info*) identified as a potential misuse case.

3 Extended misuse case notation

This paper proposes an extended misuse case notation to enable visualisation of vulnerabilities and the insider threat. The original misuse case notation only defines outside attackers [18]. However, inside attackers also pose a serious threat. An insider, to an organisation or a system, usually has much easier access to a system and thereby may perform other attacks and exploit other weaknesses than an outside attacker. As such it is useful to be able to model insiders as a separate actor type in order to get a comprehensive and complete overview of possible threats and attacks. In the original misuse case notation misuse cases are linked directly to use cases that they threaten. In other words attacks are linked to system functionality that may be disabled or otherwise damaged as a consequence of a successful attack. However it would be useful to be able to visualize what vulnerabilities are exploited to perform that attack. Threats towards a system may only be realized in an attack if the system contains vulnerabilities that can be exploited. It is important to be able to illustrate vulnerabilities to be able to identify all possible threats and attacks. We define an *insider* and a *vulnerability* as:

- *Insider* - a misuser that is also member of an authorized group for the entity being attacked - e.g. an authorized user of a system, a member of the development team, an employee of an organization.
- *Vulnerability* - a weakness that may be exploited by misusers.

Figure 2 presents a combined overview of the notation for use cases and extended misuse cases. In addition to actors representing insiders and misuse cases representing vulnerabilities an additional relationship *exploit* is defined. The *exploit* relationship is used to link a threat to a vulnerability. Insiders and vulnerabilities have been given the same grey colour in this extended notation. This choice of colour indicates that both represent weaknesses in a system that may or may not be exploited. Either way it is important to have knowledge about the weak spots of a system as this constitutes the systems *attack surface* that may be exploited. The remainder of this section presents examples of how to use the extended notation. We have included three examples that illustrates different situations and systems where the notation will be useful.

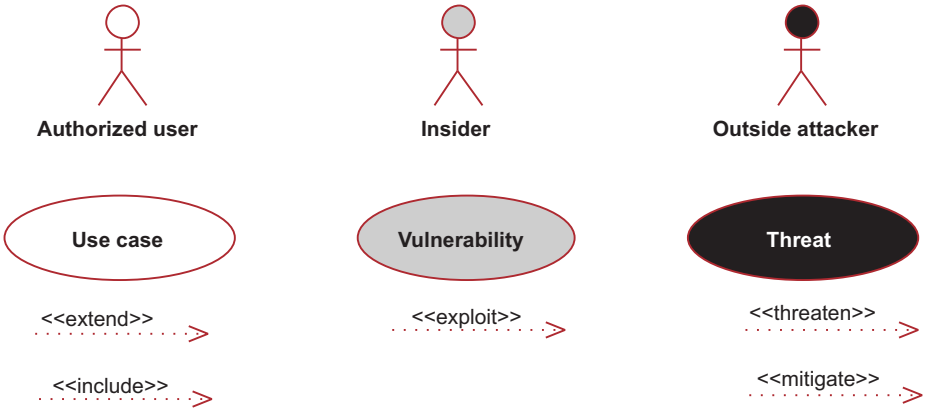

Fig. 2. Extended misuse case legend

3.1 Examples of use of the extended notation

Emergency access control in healthcare systems Figure 3 depicts an example misuse case model using the extended notation proposed in this paper. The model illustrates use and misuse of the access control mechanism in an Electronic Patient Record (EPR) system. As explained in the introduction, such healthcare systems often have emergency access control mechanisms designed to be able to override the standard access control mechanisms in situations where access to information is of vital importance but there is no time to register the patient in the system and link him/her to a specific ward - which is necessary for the standard access control to function properly. In these situations healthcare personnel are authorized, by their organization and the law, to use the emergency access control mechanism to gain access to information that they have a legitimate need and right to view. However, for such an emergency mechanism to be useful, it has to be available at all times. This effectively leads to a backdoor into the system that may be misused by insiders to snoop around when they should not. Most system users will not attempt misusing this mechanism although it is possible. But, it is important to be able to consider the possibility and map out potential consequences and apply proper countermeasures if the consequences are grave. And that is the reason why this addition to the misuse case notation is important. You cannot get a complete overview of potential risks and threats towards a system if you do not consider the complete picture. By identifying emergency access as a vulnerability we are also able to consider proper countermeasuers to apply in order to minimize the risk for misuse - in this case auditing (enables traceability and detection of misuse) and awareness training (e.g. making sure that system users are aware of the consequences of misuse - and what is considered misuse).

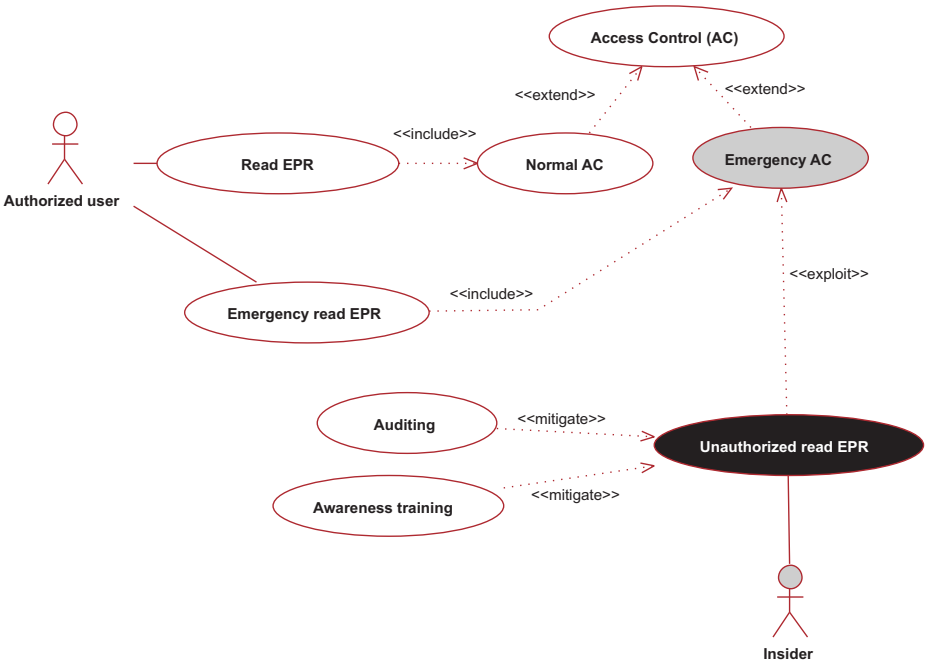

Fig. 3. Extended misuse case example: access control

User input in web-enabled systems In an IT-system all input, from users or other systems, should be handled with caution. Figure 4 illustrates a generic login procedure for a web application - the user has to enter a username and password to log in. Identified attacks include (but are definitely not limited to):

- Injection - for instance sql-injections to tamper with database content or override password check.
- Overflow - entering unexpected or large quantities of data in the input fields to observe system reaction or possibly take control over the system.

Input validation is identified as a countermeasure that helps mitigate these threats. This model illustrates how the extended notation helps highlight vulnerabilites that may be exploited. An insider is not inlcuded because these attacks are typically performed by outside attackers. Highlighting vulnearbilities in this way may be particularly helpful in a risk analysis process, where the customers are involved. By visualizing vulnerabilities, attacks and what may happen it will hopefully be easier to get acceptance and resources to apply security measures.

An insider on the system development team This example illustrates how the extended notation may be used not only on a system level, but also on a business- or organizational level. An insider may exist inside a development

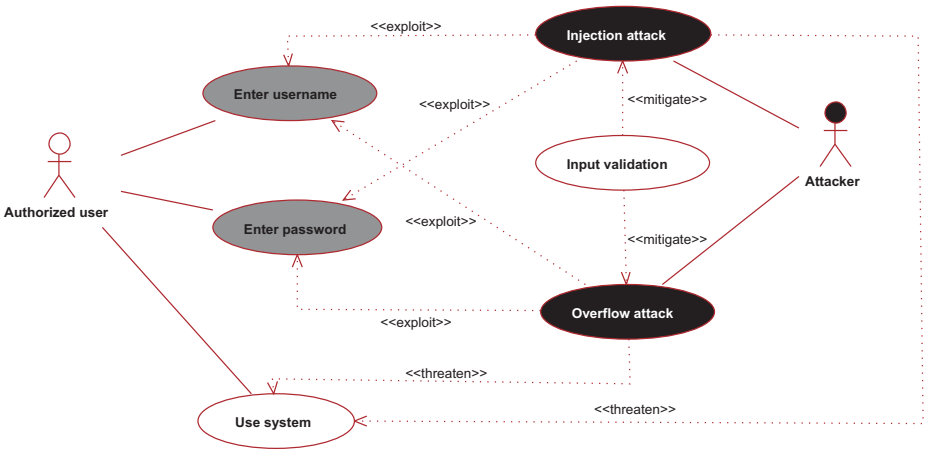

Fig. 4. Extended misuse case example: user input

team or an organization. For example a disgruntled employee working on a development project may inject code into a system that opens up a backdoor that attackers may exploit like Figure 5 illustrates.

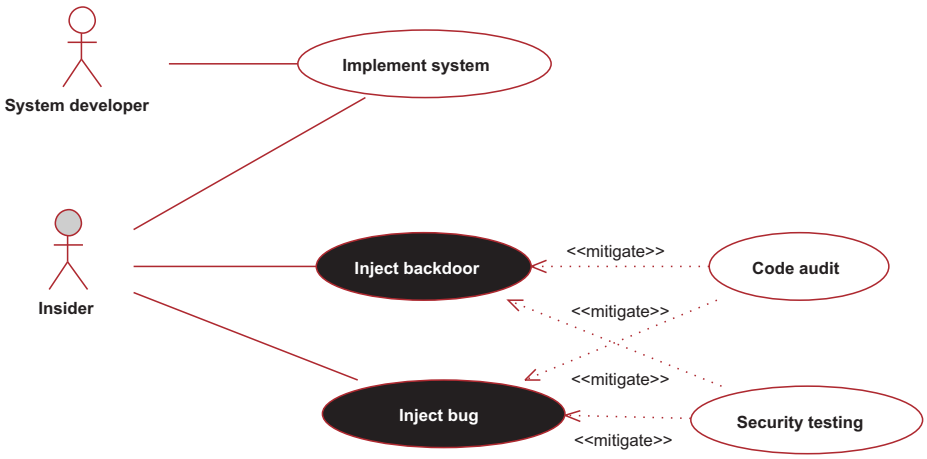

Fig. 5. Extended misuse case example: insider in development team

3.2 A step-by-step approach: how to apply the extended notation

In [15] Sindre and Opdahl propose guidelines, a set of steps, to perform when using misuse cases to elicit threats and countermeasures. The approach described

here for applying extended misuse cases is based on their guidelines, but refined to include the necessary activities to consider the insider threat and uncover vulnerabilities.

1. **Identify actors and use cases for the target system.** Only with an overview of what the system is supposed to do, and who will use it, is it possible to start identifying potential attackers, weak spots and threats. Create UML use case models. These will serve as input to the subsequent steps.
2. **Identify potential outside attackers.** With knowledge about the system discovered in step 1 one should be able to identify who might target the system for attack purposes from the outside. This may include a wide range of persons with a wide variety of motivations - from the unskilled hacker using downloaded tools to jam the system using a DoS[1]-attack, to highly skilled industrial spies. At this point it is important to create as complete a list as possible of potential attackers. Later, in the risk analysis process one can eliminate attackers that are deemed unlikely or that will probably not be able to cause any harm.
3. **Identify potential insiders.** As with outside attackers there may be a variety of insiders. For example there are the people developing a system - who may intentionally inject backdoors - and there are different kinds of users of the operational system that have different rights and thereby differ in the harm they are capable of inflicting on the system. On this point as well, the main concern is to generate as complete a picture as possible.
4. **Identify threats.** Having identified potential misusers who may harm the system, the next step is to identify what types of harm they may want to inflict on the system - i.e. potential threats and attacks. To be able to do this one should consider what might be the goal of the identified misusers - what would they want to achieve?
5. **Identify vulnerabilities.** This step means analysing how threats and attacks may be performed. Given the identified threats - how may an outside attacker, or insider, do this? This means examining the systems functionality identified in step 1 and consider each use case carefully to decide if it may be exploited for malicious purposes. When a potential vulnerability is identified it should be labeled accordingly in conformance with the extended misuse case notation.
6. **Identify security requirements.** Having identified misusers, threats and vulnerabilities - in this step the focus is on countermeasures. This is done by adding security use cases (as earlier mentioned these use the notation of ordinary use cases) to the models and adding the *mitigate* relationship to the threats or vulnerabilities they protect against.
7. **Revise findings so far.** This is of course an iterative process that may be carried out several times before one is satisfied that the result is reasonably sound and complete. Creating a 100% complete overview of all risks is infeasible but applying a structured risk-based approach and using the right

[1] Denial of Service

people with the required knowledge [13] should help ensure the best possible result.

Note that the steps need not necessarily always be carried out in exactly this order. Specifically steps 2 through 5 may be intertwined as it may be hard or possibly not beneficial to completely separate these steps.

4 Relation to risk management in system development

Risk analysis, and risk managed development processes, is a well known technique for making decisions in many engineering fields. Building secure systems is about managing risks. It is not possible to build a system that is absolutely secure against all attacks, known in the present or that may be invented in the future [22]. Risk managed system development is about creating systems that are reasonably protected against known attacks and with a robust build using design principles that will hopefully make the system able to withstand future attacks. What is reasonable protection and what risks should be handled is for the system stakeholders to decide - i.e. the customer. To decide what risks to handle one needs to rank the identified risks and this requires assigning a value. A risk value is calculated as:

$$Risk = Probability \times Consequence \tag{1}$$

Misuse cases provide an overview of information that is very useful in a risk analysis process [10]. However, misuse cases only provides an overview and should be a starting point for creating attack trees [22] and doing threat modelling [20] to get a complete view of the threats and vulnerabilities in a system. Adding notation for expressing vulnerabilities and the insider threat makes the misuse case notation richer and adds more detail which should provide a better starting point for the continuing risk management process.

5 Discussion

Although not all vulnerabilities may be represented in a use case or misuse case model, it is important when considering adding functionality to a system to examine if it represents a vulnerability that can be exploited. Only then is it possible to make a risk-based decision whether to not include that functionality or apply the necessary countermeasures to ensure protection. The possibly greatest power of use and misuse cases is that they are so graphical and easy to understand, and work very well as a basis for discussion with system stakeholders. Typically customers are not eager to spend money on security as it does not directly add to the value of the product. Misuse cases can help convince customers that security is important. Extending the misuse case notation helps this process as it enables:

– Visualisation of effects of adding functionality that might seem desirable, but actually represents vulnerabilities. The extended misuse case notation enables explicitly stating how vulnerable functions may be exploited.
– The insider threat should not be neglected. Insider attackers have very different possibilities from outsider attackers and by using a separate notation for insiders one is able to emphasize this.

The extensions proposed here are simple, in accordance with the original misuse case notation. The idea is to keep close to the UML use case notation and only add what is needed to include security concerns, while keeping the models very easy to understand.

6 Conclusion and further work

This paper has presented and shown examples of an extended misuse case notation including notation for expressing vulnerabilities and insider attackers. This adds to the expressiveness of misuse cases while still keeping the notation very straightforward and easy to understand. The extended notation enables expressing a richer and more complete picture of security threat considerations for a system which is useful when using misuse cases in risk analysis. To further investigate the ideas presented here, it would be useful to create a textual representation of extended misuse cases. Also, security functionality is currently represented as ordinary use cases. It might be useful to create a specific notation for security functionality, or countermeasures that have been added to mitigate vulnerabilities and threats.

7 Acknowledgements

The ideas described in this paper was inspired by the project on case studies of access control in healthcare performed by Julie-Marie Foss and Nina Ingvaldsen at the Norwegian University of Science and Technology (NTNU) in the fall of 2004 which I had the pleasure of supervising. Misuse cases was used in that project to model findings and some alterations, initiated by very useful comments and suggestions from Guttorm Sindre, to the notation had to be made to be able to express all findings. These alterations were the starting point of the extended notation described in this paper.

References

[1] Unified modeling language: Superstructure. Technical report, Object Management Group (OMG), August 2005. http://www.omg.org.
[2] I. Alexander. Initial industrial experience of misuse cases in trade-off analysis. In *IEEE Joint International Conference on Requiremenst Engineering*, Essen, Germany, 2002. IEEE.

[3] I. Alexander. Modelling the interplay of conflicting goals with use and misuse cases. In *Goal-Oriented Business-Process Modeling (GBMP) 2002*, volume 109, London, UK, 2002. CEUR Workshop Proceedings.

[4] I. Alexander. Modelling the interplay of conflicting goals with use and misuse cases. In *International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ) 2002*, Essen, Germany, 2002.

[5] I. Alexander. Misuse cases help to elicit non-functional requirements. *Computing & Control Engineering Journal*, 14(1):40–45, 2003.

[6] I. Alexander. Misuse cases: Use cases with hostile intent. *IEEE Software*, 20(1):58–66, 2003.

[7] I. Alexander and N. Maiden. *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*. John Wiley & Sons, 2004. ISBN: 0470861940.

[8] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-Based Access Control*. Computer Security Series. Artech House Publishers, Boston, 1 edition, 2003. ISBN: 1580533701.

[9] D. G. Firesmith. Security use cases. *Journal of Object Technology*, 2(3):53–64, 2003.

[10] P. Hope, G. McGraw, and A. I. Anton. Misuse and abuse cases: Getting past the positive. *IEEE Security & Privacy*, 2(3):90–92, May/June 2004.

[11] J. McDermott. Abuse case models for security requirements analysis. In *Symposium on Requirements Engineering for Information Security (SREIS)*, Indianapolis, USA, 2001.

[12] J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In *Annual Computer Security Applications Conference*, Phoenix, Arizona, 1999.

[13] G. McGraw. *Software Security - Building Security In*. Addison-Wesley Software Security Series. Addison-Wesley (Pearson Education), Boston, 1 edition, 2006. ISBN: 0321356705.

[14] G. Sindre, D. G. Firesmith, and A. L. Opdahl. A reuse-based approach to determining security requirements. In *9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03)*, Klagenfurt/Velden, Austria, 2003.

[15] G. Sindre and A. L. Opdahl. Eliciting security requirements by misuse cases. In *37th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-Pacific 2000)*, pages 120–131, Sydney, Australia, 2000.

[16] G. Sindre and A. L. Opdahl. Capturing security requirements through misuse cases. In *Norsk Informatikkonferanse (NIK)*, Tromsø, Norway, 2001.

[17] G. Sindre and A. L. Opdahl. Templates for misuse case description. In *Seventh International Workshop on Requirements Engineering: Foundation of Software Quality (REFSQ'2001)*, Interlaken, Switzerland, 2001.

[18] G. Sindre and A. L. Opdahl. Eliciting security requirments with misuse cases. *Requirements Engineering*, 10(1):34–44, 2005.

[19] G. Sindre, A. L. Opdahl, and G. F. Brevik. Generalization/specialization as a structuring mechanism for misuse cases. In *2nd Symposium on Requirements Engineering for Information Security (SREIS'02),*, Raleigh, NC, USA, 2002.

[20] F. Swiderski. *Threat Modeling*. Microsoft Press U.S., 2004. ISBN: 0735619913.

[21] D. Verdon and G. McGraw. Risk analysis in software design. *IEEE Security & Privacy*, 2(4):79–84, 2004.

[22] J. Viega and G. McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison Wesley, 2001. ISBN: 020172152X.