Presented at the Second International Workshop on Secure Software Engineering, in conjunction with ARES 2008, Barcelona, Spain, March 2008

Covering Your Assets in Software Engineering

Martin Gilje Jaatun and Inger Anne Tøndel
SINTEF Information and Communication Technology
Department of Software Engineering, Safety and Security
{martin.g.jaatun, inger.a.tondel}@sintef.no

Abstract

Many security requirements elicitation techniques implicitly assume that assets are identified on beforehand, but few actually describe how this should be done. In this paper we suggest one specific method that can be used to identify and prioritize assets in any software engineering project.

1 Introduction

The concept of "assets" is central to the very idea of computer security – we need security because we have something that needs protection. This "something" is what we collectively refer to as our assets. Thus, asset identification is a crucial component of the requirements phase – specifically, security requirements are primarily needed in order to protect our assets, and this will obviously be impossible to do properly unless we know what these assets are.

We have previously [1] argued the need for secure software engineering methods that are suitable for the "average" developer. This remains our focus also in this paper, and our proposed method for asset identification is intended for "normal" software engineering projects, not projects that are particularly security critical. This implies that the method must be easy and inexpensive to use, also for developers who are not security experts. The goal of the method is to discover all the assets that are relevant[1] for the system being developed, and facilitate a prioritization process in order to identify which assets have a higher (or lower) priority with respect to information security.

Before asset identification takes place, the main security objectives of the software to be developed should be identified. By security objectives we mean high level security requirements or goals identified by customers, and any security requirements coming from standards, policies or legislation. The results of asset identification should be used as a basis for identifying threats, where attack trees [2] or similar are created based on the most important assets identified. Security requirements are then elicited based on threat analysis.

[1] Strictly speaking, our primary concern is to identify the assets that are *most important* with respect to information security – if we overlook assets that *don't* need protection, we can still sleep at night.

2 Related work

We have in our previous survey [1] shown that asset identification is considered important in several major security requirements engineering initiatives [3–6]. Details on how to perform asset identification in the context of software engineering are however rare. CLASP [3] states that it is important to take a resource-centric approach: "Functional security requirements should show how the basic security services are addressed for each resource in the system, and preferably on each capability on each resource." Exactly how one should go about identifying a system's resources is not explained beyond providing a list of sample resources. Haley et al. [5] place a lot of focus on the importance of identifying assets; but advice on how to do this is limited to showing in a figure that assets are elicited from application business goals, in addition to a description of what assets are: "In general assets consist of all the information resources stored in or accessed by the system-to-be and tangible resources such as the computers themselves. Assets can be composed of other assets; backup tapes would be a good example." Boström et al. [6], who focus on security requirements engineering in development projects using eXtreme Programming (XP), suggest "Identification of critical assets" as an important step, but actual advice is limited to a definition of assets, one example ("Confidential negotiation proposals") and the following description: "The XP team, led by its security engineer, collaborates with the customer to identify relevant assets and their value."

Of the above mentioned initiatives, the most detailed suggestions on how to perform asset identification come from Microsoft. Lipner and Howard [4] describe asset identification as part of threat modeling: "Using a structured methodology, the component team identifies the assets that the software must manage and the interfaces by which

those assets can be accessed." More details are not given as to how this structured methodology works. The threat modeling methodology of Microsoft is further described by Swidersky and Snyder [7]. Here "Determining which assets are of interest" is one of the steps suggested. Most of the section related to assets focuses on describing different types of assets, but some tips are also given on how to identify assets: "Many assets are identified when discussing system functionality, use scenarios, and other background information. Questions to ask include: Does the system have access to any resources that an external entity could not normally access? Which aspects of the system are critical to proper functionality? What is the purpose of this system?" Which information to collect on each asset is detailed, including numerical ID, name, description[2] and trust levels[3].

An additional initiative not mentioned in Tøndel et al. [1] is the AEGIS (Appropriate and Effective Guidance for Information Security) process for building secure and usable software [8]. Assets are a main focus of AEGIS, and the following categories of assets are considered: Operatives (e.g. users, administrators, developers), hardware (e.g. network link, computer) and data (e.g. application, information). In addition, security measures are introduced as an asset category that can consist of any of the other asset types. Several different types of system stakeholders are involved in the AEGIS process, including decision makers, developers, users and facilitators. There is little detail on how assets are identified, but an UML asset model is provided, and a case study with students explains that assets were identified by "one student drawing the asset model onto a whiteboard while the group of students as a whole asked detailed questions about the architecture that would inform the diagram." Participants then further judge the identified assets importance when it comes to confidentiality, integrity and availability in a qualitative way based on natural language. Scenarios, for instance in the form of abuse cases, is also suggested to improve the understanding of what the security properties means for an asset. AEGIS then continues with risk analysis followed by design activities.

Asset identification is also a common activity in risk analysis. An important example of an asset driven risk evaluation approach is OCTAVE[4] (Operationally Critical Threat, Asset, and Vulnerability Evaluation[5]). OCTAVE is targeted at organizational risk and comes in three main versions: the OCTAVE method that is centered towards large organizations with 300 or more employees; OCTAVE-S which is centered towards organizations of about 100 people or less; and finally OCTAVE Allegro that focuses primarily on information assets, how they are used, and threats towards these assets. OCTAVE Allegro [9] is the most relevant OCTAVE version for our setting. In OCTAVE Allegro it is suggested that assets are identified by brainstorming activities, and experience shows that users of the methodology have experienced little difficulty in identifying assets. More focus is therefore placed on how to identify the most important assets, and the use of critical success factor analysis [10] is suggested as a way to improve this. For each selected asset it is then registered why it was selected, who is the owner of the asset, and what its main security requirements are. Further activities include identifying and investigating areas of concern for each asset.

[2]Including why the asset needs protection.

[3]Trust levels are explained as access categories. Should state which trust levels are normally allowed to access or interact with the asset.

[4]http://www.cert.org/octave/methods.html. OCTAVE® is registered in the United States Patent and Trademark Office by Carnegie Mellon University

[5]Operationally Critical Threat, Asset and Vulnerability Evaluation is a service mark of Carnegie Mellon University

3 Asset identification

Our asset identification method helps to establish an overview of the assets of a system and their different requirements for protection. This information makes it easer to prioritize security requirements later on. We suggest to look at the value of the assets both from the customer's and the system owner's point of view, but also from the view of an attacker. To identify assets one can use functional requirements of the system in combination with brainstorming techniques. For each asset one should then make a judgment regarding the different stakeholders' priority of the confidentiality, integrity and availability of this asset. When assigning priorities one should use predefined categories, for instance high, medium and low.

When an asset identification has been performed one has a view of which properties of what assets are most important to protect. This information should be used to prioritize requirements, and also to identify where to focus the effort when it comes to identifying more detailed requirements.

3.1 Key contributors

The asset identification process should be performed by a group composed of:

- Requirements team
- Other developers
- Security experts (if available)[6]
- Customers and/or end-users (if applicable)

[6]Security expertise will be a bonus in this process, but our method is designed to also work without it.

Generally, anyone that could contribute with ideas on assets may contribute, but the total group should not exceed 8 persons (plus facilitator).

This method can be used in any project. For large projects one may however need to use more coarse assets than in smaller projects, or execute the method several times on a subset of the application domain.

3.2 Step 1: Brainstorming

We suggest to use the brainstorming techniques described by Dybå et al. [11]. Note that this may be considered an amalgamation of traditional brainstorming and "brainwriting" [12] [13]. The purpose of brainstorming is generally to generate ideas on a given topic; in our case the topic is restricted to answering the question "what are our assets". By using this technique one is able to involve different types of persons in a creative idea-generating process, without putting too many restrictions on the end result.

As preparation for the brainstorming session, one must make available Post-it Notes[7] and pens for all participants, and somewhere to put the Post-it Notes during brainstorming (e.g. a large sheet of paper to put on the wall).

1. Present the process and the rules to follow while brainstorming.
   - Everybody shall participate
   - No discussion/criticism during the brainstorming
   - One should build on ideas presented by others
2. Give out Post-it Notes and pens to all participants
3. Decide on a time limit for individual brainstorming, e.g. 5 minutes.
4. Formulate a question on which to brainstorm (e.g., "What are our assets?"), write it down and place it somewhere visible to everybody in the group.
5. All participants write down their ideas – one idea per post-it note.
6. When time is up, everybody presents their ideas to the group by placing their post-it notes on for instance a piece of paper on the wall. Often it is advantageous to do this in a structured way: Everybody takes turns presenting their ideas. One is only allowed to present a limited number of ideas at the time, and the remaining ideas must wait until the next round.
7. Group the ideas and eliminate duplicates. Everybody should participate in this step.
8. Document the result, e.g. with a camera.

Note that although Wilson [13] warns that a trained facilitator is necessary to ensure success of a brainstorming session, we have found that even with just "normal" participants (and appointing one facilitator), there is tangible value to be had from brainstorming. Furthermore, this is also to a great extent *learning by doing* – if a development organization frequently employs brainstorming in relevant situations, the individual participants will in time become reasonably confident (if not to say skilled) as facilitators.

In a small group where the participants know each other well, it may be more cost-effective with respect to time to let each participant express their ideas orally (one at a time), and assign one participant (or facilitator) to write the ideas on a whiteboard as they are presented. In this case, duplicates are avoided, and it may be easier (quicker) to get new ideas regarding assets based on the assets that are being presented.

3.3 Step 2: Assets from existing documentation

Once the brainstorming session is finished, it is a good idea to examine any available functional requirements or functional descriptions of the system to determine if any important assets have been overlooked. In some cases, this may inspire a second round of brainstorming.

3.4 Step 3: Categorization and prioritization

Once a list of assets is available, the assets must be categorized and prioritized with respect to security. This should be performed by the same group that participated in the asset identification, possibly augmented by management participation[8].

We recommend assigning priorities from three stakeholders' perspectives:

- The Customer/system user
- The System developer/owner
- The Attacker

As can be seen from Table 1, the different stakeholders' priority of the assets is described with three letters:

C: Confidentiality

I: Integrity

[7] *Post-it Notes*® is a registered trademark of 3M (http://www.3m.com). We have found that generic-brand sticky notes also work.

[8] Management should possibly *not* be invited to the brainstorming session itself, since one of the pitfalls identified by Wilson [13] is inviting "anyone [...] who is feared by the other members".

ASSETS	STAKEHOLDERS' PRIORITY		
	Focus: protection. What is most important to protect from stakeholders' point of view?		**Focus: attacks.** What is most interesting/valuable for an attacker?
Description	**Customer/ system user**	**System developer/ owner**	**Attacker**
<asset 1>	<C-? I-? A-?>	<C-? I-? A-?>	<C-? I-? A-?>
<asset 2>	<C-? I-? A-?>	<C-? I-? A-?>	<C-? I-? A-?>
...	...	...	...

Table 1. Asset Prioritization Table

A: Availability

These letters can then get assigned a number indicating the importance of confidentiality, integrity or availability for this asset[9].

1. Decide which categories to use to represent priorities. In most cases it will be adequate with three (qualitative) levels:

   1: High

   2: Medium

   3: Low

2. For each asset, make a judgment regarding the different stakeholders' priority of the confidentiality, integrity and availability of this asset. If e.g. confidentiality is not an issue for an asset, it is not necessary to include this property and assign a level to it (the same goes for integrity and availability).

3. If many assets have been identified, consider pruning the assets with low priorities.

If management has not been involved up to this point, they should be given the opportunity to comment on the asset tables. For small systems where a limited number of assets are identified, the final prioritization may be performed manually. With a large number of assets, we suggest adding together all priorities for each asset (a missing priority counts as a "4"), and ranking the asset with the lowest sum as the highest overall priority. This implicitly gives more weight to customers' and owners' priorities over attackers', but this seems reasonable since the latter are the most imprecise.

[9]We maintain that the traditional CIA triad is enough for this purpose - additional properties like non-repudiation will represent special cases that should be treated separately.

4 Experiences

In the spirit of how compilers traditionally were developed [14][10], we have employed our asset identification method within the project where we defined it; the SODA web application *SODAweb* [15]. The SODA project defined two versions of a web application (the two versions where defined one year apart), where different developers where involved. In the following we describe our experiences identifying assets in the two different versions of the application.

4.1 First version

The idea behind SODAweb is to provide a tool for system developers that will help improve the information security properties of everyday programs. It is basically a publicly accessible web resource with open content, but where users can log on to receive more personalized service. SODAweb shall assist the users in selecting techniques by offering a set of questions about the user's preferences and create a profile based on the answers.

SODAweb shall be an independent and self-contained application – meaning that it does not depend on any other systems to be useful to its users. However, SODAweb may contain recommendations of other products that are useful for applying the individual techniques.

Security is a major concern for SODAweb. As the product is to supposed to promote secure software development, it is important that the product is secure, i.e. that it is resistant to malicious attacks.

4.1.1 User interfaces

SODAweb has two main user interfaces: one for administrative use (content and user administration) and one main interface for using and interacting with the system for regular users. Both these user interfaces shall be web-browser-

[10]Where C-compilers were written in C, and subsequently compiled with themselves.

ASSETS	STAKEHOLDERS' PRIORITY		
	Focus: protection. What is most important to protect from stakeholders' point of view?		**Focus: attacks.** What is most interesting/valuable for an attacker?
Description	**Customer/ system user**	**System developer/ owner**	**Attacker**
Username	I-1 A-3	I-1 A-2	I-2 A-2
Password	C-1 I-2 A-3	C-1 I-2 A-2	C-1 I-2 A-2
Technique descriptions	I-1 A-1	I-1 A-1	I-1 A-1
Personal profile	C-2 I-1 A-3	C-2 I-1 A-2	C-3 I-2 A-2
Personal notes	C-1 I-1 A-3	C-2 I-1 A-2	C-2 I-3 A-3
Content structure	I-1 A-1	I-1 A-1	I-1 A-1
Questions (for profile)	I-1 A-3	I-1 A-2	I-1 A-3
Authorization data	C-2 I-1 A-3	C-2 I-1 A-2	C-1 I-1 A-3

Table 2. Asset prioritization table for first application version

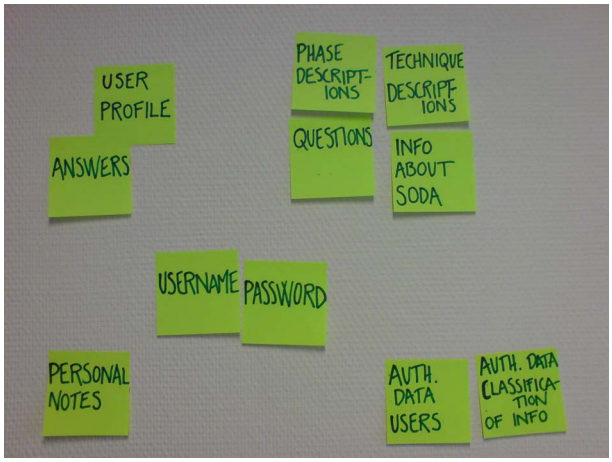

Figure 1. Result of the brainstorming session

based. The regular user interface shall map closely to the phases defined in the SODA lifecycle model [15].

4.1.2 Assets identified

We performed a brainstorming session as described in section 3.2, producing the assets depicted in Figure 1. We then proceeded immediately to classification and prioritization.

Table 2 shows the main assets and to what extent they need protection. As can be seen, in this application availability of the personalized service is not considered of high importance, since the open content (technique descriptions) will still be available. Confidentiality is also not considered relevant for the open content and for the usernames.

4.2 Second version

The second version of the SODAweb application that was specified was significantly less ambitious than the first. This time, the goal was simply to define a mostly read-only web application (little more than a static web page hierarchy). In this case, we also specifically stated that the application would be running on a publicly accessible server in SINTEF's domain (http://www.sintef.no). The only interactive component was decided to be a feedback mechanism where users can report experiences with the application etc.

The security objectives that provided the context for the asset identification were:

- Integrity of the application
- SINTEF's IT regulative and security policy

The identified assets and the results of the prioritization process are shown in Table 3. Reputation was also identified as an asset, but it proved difficult to fit this into the mold, and it was therefore left out of the table. However, it was agreed that damage to the integrity of the application would also damage our reputation (as system owner).

In contrast with the first version, we did not leave out criteria considered less important, but instead assigned a value of 3 to these. This does not conform to our method as described in section 3, and in hindsight we realize that not excluding (e.g.) confidentiality when it is considered unimportant effectively compresses the scale from four to three levels. This is unfortunate, and we thus recommend that the original method is adhered to.

4.3 Process evaluation

The brainstorming technique was productive in identifying assets for both versions of the application. Though

ASSETS	STAKEHOLDERS' PRIORITY		
	Focus: protection. What is most important to protect from stakeholders' point of view?		**Focus: attacks.** What is most interesting/valuable for an attacker?
Description	**Customer/ system user**	**System developer/ owner**	**Attacker**
Information	C-3 I-1 A-2	C-3 I-1 A-2	C-3 I-2 A-2
Web server	C-3 I-2 A-2	C-2 I-1 A-2	C-3 I-1 A-2
Database	C-3 I-1 A-2	C-2 I-1 A-2	C-3 I-1 A-2
Our internal network	C-3 I-3 A-3	C-1 I-1 A-1	C-3 I-2 A-3
Administrator account	C-3 I-3 A-3	C-1 I-1 A-2	C-2 I-1 A-3
Feedback information	C-3 I-1 A-3	C-2 I-2 A-3	C-3 I-2 A-3

Table 3. Asset prioritization table for second application version

the composition of the teams for the different versions varied slightly, both authors were present in each process. In each case, the same persons who were defining functional requirements were also part of the asset identification, and they thus knew each other on beforehand.

The asset identification process was fairly efficient in both cases, taking approximately an hour[11]. The result provided a useful overview and a starting point for prioritizing which assets to focus on in the ensuing requirements elicitation process (misuse cases, attack trees, etc.). It is important to take the time to actually perform the prioritization; our stated goal is to identify the most important assets. The extent and coverage will always depend on how important security is considered in any given project.

As in any collaborative process, there may be considerable discussion and disagreement with respect to what value or prioritization to put on any given asset. The actual arguments and rationale used for categorizing a given asset is not documented in our method – this may lead to difficulties in comparing asset identification sessions, even if the reasoning is reasonably consistent within a given session. On the whole, however, we feel that we have struck the right balance between documentation rigor and ease-of-use.

Our two examples were small enough for the final ranking of assets to be done manually. For illustrative purposes, we have nonetheless calculated the "priority sum" and the resulting ranking for the second example in table 4.

At first glance, the two asset tables seem quite different, but this is mainly due to different granularity with respect to different aspects of the two cases. E.g., "Technique descriptions", "Personal profile" and "Content structure" in the first version approximately map to "Information" in the second version, "Personal notes" maps to "Feedback information", and "Authorization data" maps roughly to "Administrator account". The assets "Web server", "Database" and "Our internal network" were not identified in the first version primarily since we then pictured a "stand-alone" server outside our own network. The absence of "Username", "Password", etc., in the second version is of course due to changing requirements as described in section 4.2.

[11] although we did not actually time the process.

5 Discussion

In the introduction we stated that we need a method for asset identification that is easy and inexpensive to use, also for developers who are not security experts. At the same time we need to be able to discover relevant assets and prioritize them. Our experiences in using the method suggested in this paper show that this method has potential when it comes to reaching this goal. It is easy to use, and the time needed should be affordable to most development projects. The time needed will also be reduced with training, and it should be possible to reuse part of the result in similar development projects later on. The method does not require specific security expertise, though it will undoubtedly benefit from participants that are able to think of the system from the viewpoint of an attacker. The method is able to discover important assets and prioritize these, though it is too early to say whether all relevant assets are discovered.

The method can be said to meet our main requirements, but still has some limitations. The success of the method is very much dependent on the individuals participating in asset identification, as the types of assets identified will depend on their competence and main focus. This can be shown by the differences in the types of assets that were identified in the two sessions described in this paper. Only one session identified assets like web server and database. One likely reason for this is that the method is based on brainstorming, which is not a structured method. To improve this it is possible to utilize checklists or predefined questions in the brainstorming activity, e.g. the questions suggested for asset identification by Swidersky and Snyder [7]. In spite of the limitations, brainstorming and other unstructured methods for identifying assets are also recom-

mended in established methods such as OCTAVE Allegro and in research initiatives like AEGIS, and we still believe this is a fruitful method for asset identification.

Using functional requirements as a starting point comes with the risk of not covering assets such as reputation. As pointed out by Swidersky and Snyder [7] assets can be abstract, like the safety of employees, the company's reputation and availability and connectivity of resources. We however believe that for this lightweight approach, most of the assets of this type can be indirectly covered by looking at the different actors' value of the assets. When stating the value of an asset from the owner's point of view, reputation should be part of the evaluation.

In our approach, information on an asset is limited to values representing the importance of the confidentiality, integrity and availability of this asset. This is done to keep the method lightweight, and it is what is needed for prioritization. The reasons behind these values and the criteria used are however lost. This may reduce the possibility to reuse the results later, and makes it harder to compare results of different sessions since the criteria may be different.

One of the main innovations in our approach compared to similar initiatives is the inclusion of the attacker's perspective. By only focusing on assets important for users or system owners, system developers may fail to cover assets that are attractive to attackers, since different actors' view of an asset are not directly related [5]. One may argue that attackers are a diverse group and that it is hard to know to what extent they have interest in specific assets. We still feel that it is useful to put oneself in the attackers place and think about who has interest in this asset, something which will influence the likelihood of attack. The attacker's perspective is then further elaborated in the threat analysis following asset identification.

Another advantage of including the attacker's perspective is that this indirectly covers some assets that are otherwise easily overlooked (or difficult to relate to). A specific example of this is the "Reputation" asset: In the past, it may not have mattered[12] to a company if someone uses their file servers without permission for storing data, as long as they behave themselves and do not create problems for legitimate users. However, this changes dramatically if the company risks being exposed in the tabloids as a haven for file-sharers and other deviants – suddenly protecting the "File Server" asset becomes much more important, implicitly because of the "Reputation" asset.

ASSETS	Ranking sum
Database	17
Web server	18
Information	19
Administrator account	19
Our internal network	20
Feedback information	22

Table 4. Calculated asset ranking for second application version

As pointed out in section 2, details on how to identify assets in a software development context are rare. One of the most concrete recommendations we are aware of are those described by Swidersky and Snyder [7] as part of threat modeling. Their recommendation of using discussions based on background information, e.g. system functionality, can be said to be concretized in our method, and the questions they suggest can be utilized in the brainstorming process. The information they recommend to register for each asset is however different from what is registered in our method. We have suggested to look at the value of the assets from the customer's, the system owner's, and the attacker's point of view. These three views are not taken by Swidersky and Snyder. On the other hand, they focus on information like descriptions and trust levels, something that if included in our approach can reduce one of the limitations of our method, namely that the criteria used and the reasons behind the judgment are lost. Including more information will however increase the time needed for asset identification.

Our approach is also in many ways similar to the AEGIS approach in the use of informal group discussions and assessment of the importance of confidentiality, integrity and availability for each asset. OCTAVE Allegro, although focusing on organizational risk, also utilizes brainstorming, and considers confidentiality, integrity and availability requirements for each asset. Compared to AEGIS our approach however lacks a view of dependencies between assets. Our experience shows that at least for smaller applications such dependencies may reveal themselves as part of creating attack trees for the major assets. It should however be considered to what extent dependencies can be identified as part of our method without making the method unnecessarily complex.

It may be argued that our ranking sum is an overly simplistic approach; more experience with applying our method to projects with a larger number of assets is required to determine the usefulness of this heuristic. Other possibilities include calculating separate sums for C, I and A for each asset, but this again comes at a cost of increased complexity.

[12] The example is somewhat construed, since I cannot imagine a system administrator ever "not caring" about illegitimate users on her system; however, this effectively would have been the reaction of the local police force should the incident have been reported: "Sooo…nothing was *actually* stolen …?"

6 Conclusion and further work

We have presented a step-by-step method for asset identification and classification that is suitable for most software engineering projects.

We have applied this method in the requirements phase of the SODA research project [15], but we intend to perform further testing in national and European projects in order to validate and subsequently improve the method we have described in this paper.

Acknowledgments

The authors wish to thank Lillian Røstad of SINTEF ICT, who led our first brainstorming session for asset identification. We would also like to thank the other participants in the SODA project, Per Håkon Meland, Jostein Jensen and Maria B. Line for fruitful collaboration. We are furthermore grateful for the influence from the Software Process Improvement group at our department, represented by Tore Dybå, Torgeir Dingsøyr and Nils Brede Moe.

Thanks also to the anonymous reviewers who provided many useful comments for the improvement of our paper.

The title of this paper is inspired by a footnote in Marcus Ranum's article "Thinking About Firewalls" [16].

References

[1] I. A. Tøndel, M. G. Jaatun, and P. H. Meland, "Security requirements for the rest of us: A survey," *IEEE Software*, vol. 25, no. 1, 2008.

[2] B. Schneier, "Attack Trees - Modeling security threats," *Dr. Dobb's Journal*, July 2001. [Online]. Available: http://www.ddj.com/184411129

[3] Secure Software Inc. (2005) The CLASP Application Security Process. Secure Software Inc. [Online]. Available: http://www.securesoftware.com/solutions/clasp.html

[4] S. Lipner and M. Howard. (2005) The Trustworthy Computing Security Development Lifecycle. Microsoft. [Online]. Available: http://msdn2.microsoft.com/en-us/library/ms995349.aspx

[5] C. B. Haley, R. Laney, J. D. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *IEEE Transactions on Software Engineering*, vol. (to appear), 2007.

[6] G. Boström, J. Wäyrynen, M. Bodén, K. Beznosov, and P. Kruchten, "Extending XP practices to support security requirements engineering," in *SESS '06: Proceedings of the 2006 international workshop on Software engineering for secure systems*. New York, NY, USA: ACM Press, 2006, pp. 11–18.

[7] F. Swidersky and W. Snyder, *Threat Modeling*. Microsoft Professional, 2004.

[8] I. Flechais, C. Mascolo, and M. A. Sasse, "Integrating security and usability into the requirements and design process," *International Journal of Electronic Security and Digital Forensics*, vol. 1, no. 1, pp. 12–26, 2007. [Online]. Available: http://inderscience.metapress.com/link.asp?id=j32v167864556552

[9] R. A. Caralli, J. F. Stevens, L. R. Young, and W. R. Wilson, "Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process," CMU/SEI, Tech. Rep. CMU/SEI-2007-TR-012, 2007. [Online]. Available: http://www.cert.org/archive/pdf/07tr012.pdf

[10] R. A. Caralli, "The Critical Success Factor Method: Establishing a Foundation for Enterprise Security Management," CMU/SEI, Tech. Rep. CMU/SEI-2004-TR-010, 2004. [Online]. Available: http://www.sei.cmu.edu/publications/documents/04.reports/04tr010.html

[11] T. Dybå, T. Dingsøyr, and N. B. Moe, *Praktisk prosessforbedring - En håndbok for IT-bedrifter*. Fagbokforlaget, 2002.

[12] M. Aiken, M. Vanjani, and J. Paolillo, "A comparison of two electronic idea generation techniques," *Information & Management*, vol. 30, no. 2, pp. 91–99, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/B6VD0-3VVVRD2-5/2/e3a232f84f07347c8c8bd7ae65314dcf

[13] C. E. Wilson, "Brainstorming pitfalls and best practices," *interactions*, vol. 13, no. 5, pp. 50–63, 2006.

[14] D. M. Richie, "The Development of the C Language," in *Second ACM History of Programming Languages Conference*, April 1993.

[15] (2007) SODA – a Security-Oriented Software Development Framework. SINTEF ICT. [Online]. Available: http://www.sintef.no/soda

[16] M. J. Ranum, "Thinking about firewalls," in *Proceedings of Second International Conference on Systems and Network Security and Management (SANS-II)*, April 1994.