Review of Security Requirements Engineering Processes

By Brendan Cervin

October 26, 2004

Review of Security Requirements Engineering Processes

Deriving security requirements during the early stages of design and development is important because security requirements can affect both non-functional requirements like network configuration and functional requirements such as user access rights. A large amount of research has been done on software requirements engineering and only recently has this work been filtering through to security requirement engineering.

This paper looks at a selection of security requirement methods in an attempt to review and explain each of their properties, weaknesses and strengths. Specifically this paper will focus on methods that derive requirements from the use of a malicious user and discuss them in regard to there usability for developers without much information security experience.

1 Introduction

Research into the area of security requirements engineering has become more popular in recent years. This has been due to a lack of methods to derive accurate security requirements. Several methods are now being researched to enable accurate creation of security requirements.

It has been common practice to design a system based on the functional requirements which the users require. Consideration of security is often an after thought or is considered during implementation when developers have a better understanding of their design in terms of security. This can lead to security vulnerabilities within the software. Vulnerabilities are weaknesses in the software system which can be attacked by a malicious user to cause harm.

It is becoming common for clients to expect/require secure systems and many software developers without software security experience are relied upon to produce secure software systems. These methods are aimed at equipping those developers with tools to create accurate security requirements during the design and analysis phase of software development.

I will review and compare several methods that aim to produce accurate security requirements. They vary in complexity but all have one similar property. All the reviewed methods make use of a style which requires a malicious user to act upon the system and from these actions security requirements are derived.

This paper initially describes some of the simpler methods in section 2 such as problem frames, threat descriptions, trust assumptions, aspects and subproblems. These methods are then discussed in terms of more elaborate security requirement engineering processes in section 3 and 4 which make use of multiple simpler methods to create a careful process to analyse an entire system and create more complete security requirements. Section 5 will compare these methods and to conclude a summary of the main points will be given with thoughts on how applicable these methods are in practice.

2 Methods used in creating security requirements

2.1 Problem Frames

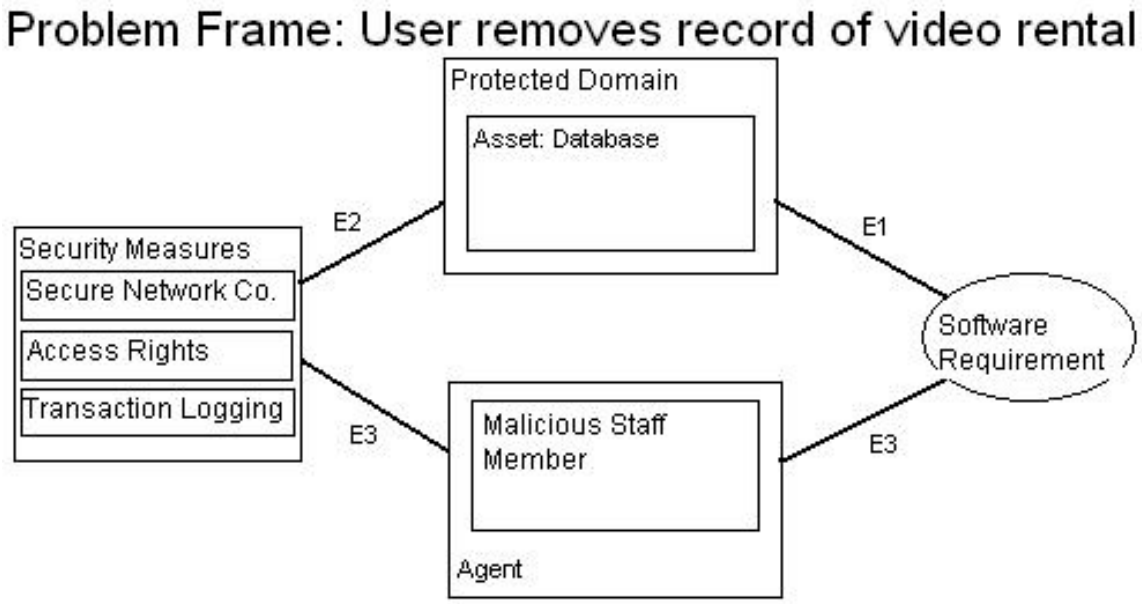
An established and well referenced approach to creating security requirements is through the use of problem frames [1]. Developed by M. Jackson they involve the use of a malicious user. A problem frame has four parts, the security requirement, the malicious user, the protected domain, and the security measures. The security requirement must describe the intention/goal of the system. The protected domains are the identified assets the malicious user is attacking. These could be database records, backup files or network devices. The next step is describing security measures such that the security requirement is fulfilled in the final system.

An advantage of problem frames is their ability to create a scope to the security requirements. If privacy is a security issue then the security requirement must apply to certain parts of the software system, the problem frame requires the developer to identify assets which are concerned about privacy and to describe the means in which they will be protected.

Below is an example fig 1 of a problem frame being applied to a software system for a video store. The problem is the removal of records showing a video was hired by a member, this results in lost property of the video store. So the security requirement states only administrators can delete records from the video store database. So in the situation where a video store staff member tries to delete a record, the database is the asset being protected. The database is protected by secure network communication protocols, access rights allowing only administrators to delete records and transaction logging so that accountability can be assigned when

discrepancies arise.

Figure 1: Problem frame describing how security requirements are met



An issue has been raised regarding creation of security requirements in general. Security requirements describe what is wanted and how it is achieved. Security requirements can impact other requirements and can become quite broad. This can over look the issue of what the security problem actually is and what the security measure is attempting to protect.

It is felt that by not describing the problem the security requirements are less accurate and less-related to other functional and non-functional requirements (cross-cutting, abuse frame). By explicitly identify the problem developers can more carefully identify what other functional requirements the problem impacts.

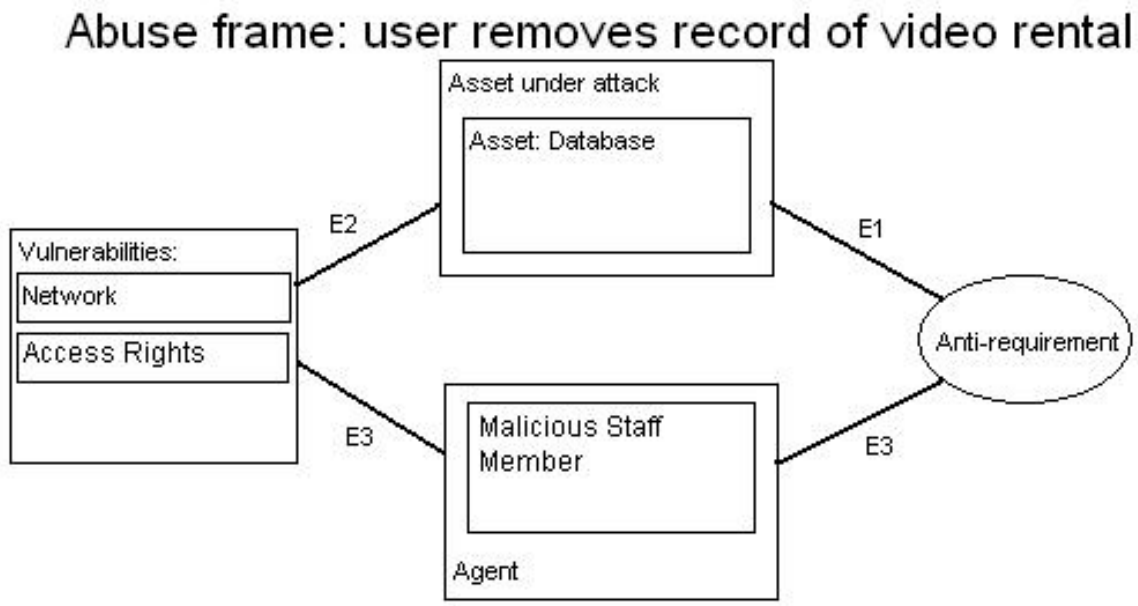
2.2 Abuse Frames

Abuse frames [2] [3] are an adaptation of problem frames, they retain the same structure but replace security requirement with anti-requirement and security measures with security vulnerabilities. They aim to identify functional requirements that security is concerned with. Abuse frames identify the problem and then work towards a solution instead of describing the security requirement initially as done by problem frames.

The anti-requirement is the requirement of the malicious user to subvert an existing requirement [2]. The security vulnerabilities describe the means by which the malicious user will fulfil the anti-requirement. This adaptation of problem frames creates a different emphasis for the developer. Once the functional requirements are made, abuse frames attempt to break them, requirements can be re-written or created in order to stop the vulnerabilities of the abuse frame being exploited.

Below an example of an abuse frame is given in 2. The problem frame example is modified to be considered within the abuse frame. So the malicious user is a video store staff member, and their anti-requirement is to delete a record of a video hire in order to steel a video. So the domain under attack still contains the database asset, and now the vulnerabilities are identified as network communication and access rights to the database. So the software requirements are then made to protect the vulnerabilities by prescribing the use of secure network communication and access rights that don't allow staff members to delete hire records from the database.

Figure 2: Abuse frame identifying vulnerabilities on assets



2.3 Aspect Oriented Architecture

Within software requirements engineering research is being carried out in the area of aspects [4]. Software programs use Object Oriented Design (OOD) to describe the design of a system using the Unified Modelling Language (UML). These OODs

are similar to the blue print of a building, they describe the structure. Blue prints have many angles or aspects from which they look at a building, each aspect show's different information important to the design of a building. Prof. John Grundy is researching the use of aspects to further describe OODs. Aspects identify where different parts of a system connect and relate to each other.

This description of software systems can be used to identify points of interaction between assets. This is of particular interest in security engineering. By displaying the system from different aspects, security concerns can be explored in terms of their scope and helps in identify all assets within the system which are part of the security concern. This relates to identifying all the assets related to vulnerability, failure to identify effected assets will limit the effectiveness of security requirements.

2.4 SubProblems

The security issues that are of concern in most software systems are CIA, confidentiality, integrity, and availability [5]. These must be broken down into subproblems [6] that can be managed effectively. Subproblem is a term which describes a problem that is broken down to the size of a problem frame. For example providing confidentiality to all areas of a software system which the security requirements must describe will involve many assets communicating with each other in various ways. There may be a number of unrelated security concerns that must be dealt with independently from each other. So the problem of confidentiality must be solved by the sum of its subproblems.

This idea can be used to provide another visual tool. If a problem is the root of a tree called a problem tree, then a developer can work down the tree creating subproblems which branch into different areas of concern. When a subproblem can no longer be broken down, it is believed that the branch adequately describes the security requirements. An assumption is made that the security requirement is met. It may not be worth breaking a subproblem down any further due to effort or difficulty. This idea will be further discussed later in regard to more complex security requirement processes.

2.5 Trust Assumptions

At some point an assumption must be made with security. The assumption that well known secure communication protocols like SSL or HTTPS will perform as expected by encrypting the communication over a network, means a security requirement is placing trust in the ability of the protocol. These assumptions are known as trust assumptions [6]. The developer must specify what level of trust they are placing in the ability of a domain. At the end of every leaf on a problem tree a trust assumption must be made, explaining the level of trust in the properties of a domain. This can then be reviewed and justified in order to create realistic security requirements.

Trust assumptions can help in making decisions regarding the level of security built and cost in a system. If a system doesn't require a great deal of security or the cost of high security is not worth the damage that could be incurred if it is broken, then a greater level of trust can be placed in a domain and no further requirements need be made to ensure security. If security is paramount and the cost of security being broken is very high, then very few assumptions may be made in an attempt to ensure a high level of security.

3 Crosscutting Threat Descriptions

This is a more complex process of deriving security requirements as described in [6]. This process combines several of the ideas explained in the above methods. The aim of this process is to create a formal method that identifies all security requirements and the effect that they have on other functional requirements.

This process does not aim to produce a list of security requirements. Instead it is to constrain already specified functional requirements so that vulnerabilities are removed. The focus is to alter the functional requirements rather than to create abstract security requirements. Through a technique called crosscutting joins are located within the software system. These are the areas of interest where assets and functional requirements join, they are points of attack. The threats and vulnerabilities at these joins are analysed by the developer.

This technique makes use of aspects to identify assets in the system and what threats exist on those assets. Depending on an assets location within the system and the function it performs the threats will vary greatly. For example a web server

will expose a database to the internet; this requirement creates a variety of threats that wouldn't apply to another database kept within the firewall. Without different aspects to show the network layout of a software system it would be hard to see from a class diagram if a database was exposed to the internet.

Problem frames are applied in the standard way to joins in order to identify security requirements. But this technique doesn't believe a security requirement is adequate. It gives definitions of security requirements as a required outcome or goal. What is more important is how the goal will be accomplished. Security requirements may state that only the system administrator may access backup files on the file server. Whereas the specification would state only the administrator access's backup files by only by logging on to the file server machine physically. This removes question over network communication threats since no access is provided over the network.

Crosscutting threat descriptions are based around threat descriptions. These are stated as "performing action X on/to asset Y could cause harm Z" [6] such as deleting a hard drive could cause loss of work and productivity. The statement above describes the relation (threat, asset, subproblem), this relation is how security requirements are identified and resolved. For a subproblem there are threats, the assets which the threat effects must be protected with a security requirement, the requirement alters the functional requirements of the software system. Once the functional requirements are altered they must be analyzed since other vulnerabilities may have been exposed.

The crosscutting technique uses problem frames which are an established method. The use of threat descriptions is a new way of summarizing the information held within a problem frame, which may be simpler but still requires the use of problem frames. The use of threat descriptions seems to create a duplication of the statement of a subproblem. Both the problem frame and the threat description overlap in the specifying of threats, this duplication may lead to confusion during specification.

The most interesting idea the author used within this technique was the alteration of existing functional requirements from security requirements. By altering functional requirements a real sense of implementing security is found within the functional requirements, rather than an abstract list of security requirements that are not directly linked to functional requirements.

The Deriving Security Requirements from Crosscutting Threat Descriptions pa-

per used many examples to depict the process of building security requirements into functional requirements. The extensive use and referencing of other research in this area helped create a tone of authority. There was no statistical or anecdotal data that showed the use of this technique in an actual software system, however because many of the ideas and methods used by this technique have already been researched elsewhere it can be believed that the combination of these ideas is effective is trustworthy.

4 Anti-models and Anti-Goals

This technique aims to create security requirements at the application level. The paper "Elaborating Security Requirements by Construction of Intentional Anti-Models" cites research showing that a major source of security vulnerabilities is at the application level. The lower levels of the OSI model such as network and transport have defined and accepted security methods such as secure crypto technologies, so these are not a focus of this paper.

Goals are created to represent requirements; this is extended to represented anti-goals which are the goals of malicious users. Both these goals and anti-goals are placed at the top of models and anti-models respectively. Both models are constructed at once with anti-goals added to the anti-model to break goals created in the model, this is an iterative process which reviews and modifies requirements as they evolve.

These anti-goals are broken down into subproblems, the subproblems form the nodes on threat trees. A threat tree exists for each goal and the leaf nodes, the smallest subproblems, represent the fine grained security requirements which the system will implement.

Threat trees create a useful visual aid in the systematic creation of requirements. The trees contain branches which can be AND/OR branches. When a subproblem can be solved by different security requirements then all possible solutions are placed on the tree using an OR join and a choice can be made to implement the most suitable requirement. If multiple requirements are required to solve a subproblem then an AND join is placed on the tree specifying that all the security requirements under it are required in order to solve the subproblem.

Once requirements are on the end of all subproblems such that all subproblems

are solved the goal is reached and the security requirements can be constructed from the goal. This is an ordered approach that uses a simple diagram to keep track of all issues concerned with the goal.

Anit-models use mathematical formalisation to carefully specify security requirements. These can become difficult to use for a developer who is unfamiliar with this technique. The paper describes a process that works in parallel with the building of functional requirements using models and goals. There are already many development processes that have been widely published such as waterfall model, Rational Unified Process [7], some of which some of which do not use goals or models during analyses and design, making this security requirement process less appealing to developers who already conform to another development process. The use of a visual threat tree can help developers see the requirements they are building and track any subproblems which have yet to be resolved.

5 Discussion

Methods which completely describe the creation of security requirements or secure functional requirements are still being researched [6][8]. While problem frames can be used to effectively describe a problem there is no systematic approach to finding all such problems within a software system at the design and analysis phase. Abuse frames alter the perspective of problem frames in an attempt to create emphasis on the problem rather than the solution so that the developer may investigate several solutions before choosing one; this helps to keep an open mind during analysis.

Aspect oriented architecture is a method that by itself provides little help to describing security issues, but can be very useful in the identification of security issues by providing more information about a software system than a traditional OOD would.

Subproblems are a method by which larger problems are refined towards a manageable smaller problem for which security requirements can be created. The problems faced within software tend to be confidentiality, privacy, integrity and availability. Subproblems can be derived from these towards the creation of security requirements.

Trust assumptions must be used in association with other methods as a means of completely describing security requirements. They help to create a value for security

requirements by specifying the level of trust placed in the requirement.

Crosscutting Threat Description technique combines problem frames, threat descriptions, subproblems, trust assumptions, and aspect oriented architecture to create security constraints on functional requirements rather than creating separate security requirements. This technique makes use of several similar methods, these are problem frames, subproblems and threat descriptions all of which are related to each other in terms of what they attempt to describe. These overlapping methods need to be clearly understood and defined in order follow the process of identification and analysis of threats right through to constraining and iterating over functional requirements until all threats have been nuetralised.

The use of anti-models with anti-goals provides a simple and useful visual aid in describing security requirements. Working from the top goals of security such as CIA the problems are broken down in a threat tree identifying all threats to these goals. This is a simple approach which is supplemented by a more complex mathematical formalism which can create difficulties for less mathematically inclined developers. This technique could be implemented without the mathematical formalism but this could cause the lose of completeness or accuracy.

6 Conclusion

Security requirements engineering is an emerging area and a number of ideas have become widely accepted as good practice. The use of threats and malicious users is widely used as a means of describing security problems. The use of a standardized set of terminology will allow inexperienced security developers to quickly grasp concepts across a range of security requirements methods.

In conclusion several of the security requirement methods discussed in this paper have already become accepted standards and processes which use these methods to create complete security requirements are still being researched and haven't been widely adopted yet. There are some effective tools available in security requirements engineering that a developer with little information security experience can apply when designing software systems.

References

[1] M. Jackson, *Problem Frames.* Addison Wesley, 2003.

[2] L. Lin, B. Nuseibeh, D. Ince, M. Jackson, and J. Moffett, " Introducing Abuse Frames for Analysing Security Requirements ," in *11th IEEE International Requirements Engineering Conference*, IEEE Computer Society Press, 2003.

[3] L. Lin, B. Nuseibeh, D. Ince, and M. Jackson, " Using Abuse Frames to Bound the Scope of Security Problems ," in *Proceedings of the 12th IEEE International Requirements Engineering Conference*, IEEE Computer Society Press, 2004.

[4] J. Grundy, " Aspect-Oriented Requirements Engineering for Component-Based Software Systems ," in *Fourth IEEE International Symposium on Requirements Engineering*, IEEE Computer Society Press, 1999.

[5] C. Pfleeger and S. Pfleeger, *Security in Computing.* Prentice Hall, 2002.

[6] C. H. amd R.C. Laney and B. Nuseibeh, " Deriving Security Requirements from Crosscutting Threat Descriptions ," in *Proceedings of the 3rd international conference on Aspect-oriented software development*, pp. 112–121, ACM Press, 2004.

[7] P. Kruchten, "Tutorial: introduction to the rational unified process," in *Proceedings of the 24th international conference on Software engineering*, pp. 703–703, ACM Press, 2002.

[8] A. van Lamsweerde, " Elaborating Security Requirements by Construction of Intentional Anti-Models ," in *Proceedings of the 26th International Conference on Software Engineering*, IEEE Computer Society Press, 2004.