Introducing Abuse Frames for Analysing Security Requirements

Luncheng Lin Bashar Nuseibeh Darrel Ince Michael Jackson
Department of Computing
The Open University
Walton Hall
Milton Keynes MK7 6AA
E-mail: {L.C.Lin, B.A.Nuseibeh, D.C.Ince, M.Jackson}@open.ac.uk

Jonathan Moffett
Department of Computer Science
University of York
Heslington
York YO1 5DD
Email: jdm@cs.york.ac.uk

Abstract

We are developing an approach using Jackson's Problem Frames to analyse security problems in order to determine security vulnerabilities. We introduce the notion of an anti-requirement *as the requirement of a malicious user that can subvert an existing requirement. We incorporate anti-requirements into so-called* abuse frames *to represent the notion of a security threat imposed by malicious users in a particular problem context. We suggest how abuse frames can provide a means for bounding the scope of security problems in order to analyse security threats and derive security requirements.*

1. Introduction and background

The increased use of computers has meant that valuable business and mission critical assets are increasingly stored and manipulated by computer-based systems. The incidence of misuse of those assets has also increased because of the worldwide accessibility of the Internet and the automation of systems.

The security engineering community has developed a variety of techniques for managing and protecting computer-based information; these have focused primarily on design and implementation issues, such as security mechanisms for detecting attacks and counter measures for reacting to security breaches. However, what the security community has identified as important, but still lacking, is a precise notion of security requirements, a means of analysing them, and a systematic approach to defining suitable problem boundaries in order to provide a focus for early security threat analysis.

Traditional information security development methods have focused on the notion of threats. They are usually driven by risk-analyses carried out at the later stages of the development process life-cycle; and often the results are unsatisfactory. Security requirements are often specified in abstract statements for which the satisfaction criteria are unclear, or in formal global assertions that are too restrictive and unintuitive to use. This has limited the support for effective reasoning about security objectives during the early stages of the development process. Moreover, without a clear understanding of the functional behaviour and the security objectives of an envisioned system, the boundaries of the associated security problems cannot be defined clearly and effective security measures cannot be identified.

To address these problems, the requirements engineering community has started to investigate systematic approaches to analysing security threats and security requirements. By analysing and elaborating security requirements, requirements engineers should be able to identify the scope of protection, reason about security threats, and evaluate the trade-offs among different design decisions, systematically and rationally. However, current techniques for analysing and reasoning about security requirements often lack the notion of security threat posed by a malicious user. Without including this notion in security requirements analysis, many security threats cannot be expressed explicitly and security measures cannot be determined effectively. A well-defined system boundary enables software developers to focus on the characteristics of problem domains and their interactions. Non-trivial security vulnerabilities can be uncovered more easily and security threats can be reduced by selecting appropriate security measures.

In order to address these issues, our goal is to explore the role of security requirements and threats in the early phases of a system development lifecycle, and to devise a suitable representation that captures such requirements.

We are currently developing an approach using Jackson's Problem Frames [2] to analyse security threats and derive security requirements. Our approach introduces two conceptual tools – *anti-requirements* and *abuse frames* – and deploys these systematically to explore security problems arising from timing issues at the requirements level.

2. Anti-requirements and abuse frames

A threat indicates the potential for abuse of assets. A threat is characterised in terms of a threat agent, a presumed attack method, any vulnerabilities that are the foundation for the attack, and identification of the assets under attack. A vulnerability is defined as the condition of a system that, in conjunction with an internal or external threat, can lead to a security failure. A security failure is a state of the system that is inconsistent with a system's security requirements.

Our approach uses problem frames as a means of defining system boundaries to provide a focus for early security threat analysis. We exploit the notion of *anti-requirement* as the intention of a malicious user. An anti-requirement (AR) is the requirement of a *malicious* user that subverts an existing requirement [1]. That is, an anti-requirement defines a set of undesirable phenomena imposed by the malicious user that will ultimately cause the system to reach a state that is inconsistent with the system's requirements.

A security threat is then represented by an *abuse frame* (figure 1). Abuse frames share the same notation as the normal problem frames, but each domain is now associated with a different meaning:

- The Machine domain contains the vulnerabilities that the malicious user is exploiting to achieve the attack (although, of course, other kinds of attack on other domains are possible).
- The victim domain identifies the asset under attack.
- The malicious user domain and an anti-requirement define the threat agent.

The phenomenon E1 describes the undesirable phenomenon in the victim domain during an attack.

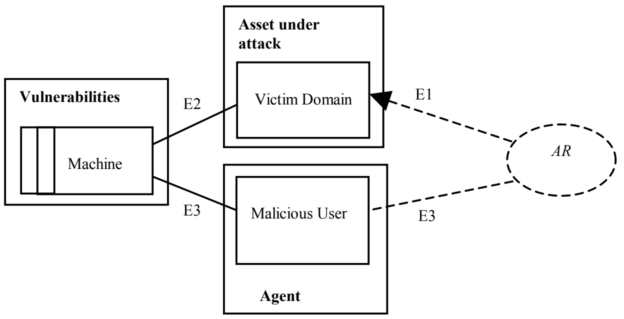

Figure 1: A threat described by a generic abuse frame diagram.

Each abuse frame is also associated with a set of *abuse frame concerns* that need to be addressed for an attack to succeed. The purpose of addressing the abuse frame concerns is to uncover vulnerabilities and provide a counterexample to the claimed security properties of a machine specification; e.g., an attack scenario that exploits the vulnerabilities in the machine and violates a security property specified by the requirements.

Abuse frames provide an abstract model of the threat imposed by a malicious user within a defined system boundary. Separation of concerns for different threats is achieved by expressing each threat in a different abuse frame diagram. In this way the security analysis focuses on the domains and the shared phenomena shown in each abuse frame diagram.

Figure 2 shows a generic problem frame diagram representing a security requirement. We make no distinction between a problem frame expressing a security requirement (SR) and those expressing other kinds of requirements. This is because we represent a security requirement as a requirement in a problem frame diagram that requires the machine to result in some phenomena in the world in order to protect the domains of assets. In figure 2:

- The phenomenon *E3* identifies the potential shared phenomena between the machine and a malicious user. *E3* can be regarded as the phenomenon that describes the attacks from the malicious user.
- The machine is the machine to be built that incorporates the security measures to counteract the attacks from the malicious user.
- *E1* identifies the desirable phenomenon of the protected domain in the presence of an attack.

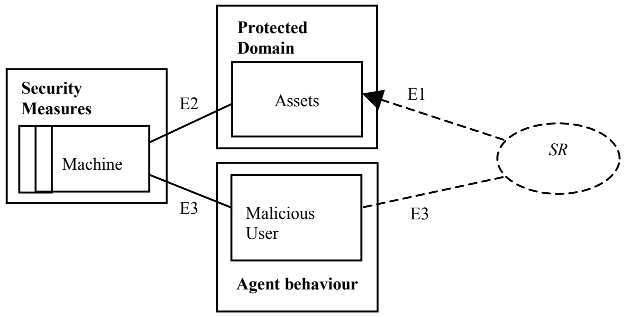

Figure 2: A security requirement expressed in Problem Frames.

Figure 2 is only one example of a security problem frame diagram. Other security requirements for different types of problem domains can be expressed using different classes of problem frames.

References

[1] R. Crook, D. Ince, L. Lin, B. Nuseibeh, "Security Requirements Engineering: When Anti-requirements Hit the Fan", *RE'02*, Germany, 9-13 Sept 02, IEEE CS Press.

[2] M. Jackson, *Problem Frames: Analyzing and structuring software development problems*, Addison-Wesley, 2001.