Security Requirements Engineering – A Strategic Approach

Chandrabose A
Research Scholar
Madurai Kamaraj University
Madurai, India

Dr Alagarsamy K
Associate Professor
Madurai Kamaraj University
Madurai, India

ABSTRACT
Although security requirements engineering has recently attracted increasing attention, it has lacked a context in which to operate. A number of papers have described how security requirements may be violated, but apart from a few hints in the general literature, none have described satisfactorily what security requirements are.

This paper proposes a strategy which unifies the concepts of the two disciplines of requirements engineering and security engineering. From requirements engineering it takes the concept of functional goals, which are operationalized into functional requirements, with appropriate constraints. From security engineering it takes the concept of assets, together with threats of harm to those assets. Security goals aim to protect from those threats, and are operationalized into security requirements, which take the form of constraints on the functional requirements.

In addition we explore the consequences of the fact that security is concerned with the protection of assets, while computers only provide interfaces. We show how to specify the relationship between security requirements and the specification of software behavior, using Jackson's Problem Frames approach.

Keywords
Security Requirements, Problem frames, Requirements Engineering, Non-functional Requirements

1 INTRODUCTION

1.1 Motivation
Although security requirements engineering has recently attracted increasing attention, it has lacked a context in which to operate. This lack was pointed out in [1], where he presents three generations of security design methods: checklists; mechanistic engineering methods; and integrated design. Unfortunately he was unable to point to any examples of integrated design methods that were used in practice. His comments apply to design, but it is also true today that there is no satisfactory integration of security requirements engineering into requirements engineering as a whole. In this section we review existing literature, in order to show the truth of this statement, and then motivate the remainder of the paper by showing why it matters.

1.2 Previous Work on Security Requirements
Extensive work has been carried out on security requirements during the last few years. [2] point out the importance of considering security requirements in the development life cycle, but does not show how to integrate them with other requirements. [3] shows how the SCR method can be used to specify and analyze security properties, without giving the criteria for distinguishing them from other system properties.

A number of papers have focused on security requirements by describing how they may be violated. For example, [4], followed independently by [5] and elaborated by [6], describe abuse and misuse cases, extending the use case paradigm to undesired behavior. [7] describe a method of analyzing possible illicit use of a system, but omit the important initial step of identifying the security requirements of the system before attempting to identify their violations.

[8] use the concept of security goals, and describe obstacles in the KAOS method, which prevent security goals from being met, but they do not then take the further step of defining security requirements at the same level as operationalized KAOS functions. [9] use the GBRAM method to operationalize security goals for the generation of security policies and requirements, but also do not define security requirements.

None of the above defines what security requirements are. On the other hand, when discussing non-functional requirements (NFRs), [10] defines NFRs as "restrictions or constraints" [on system services] and similar definitions can be found in other text books. Security requirements are an instance of NFR, and our view is identical to that of Kotonya; they are requirements for constraints on system functions.

[11] appears to take a similar view, stating "security requirements mostly concern what must not happen".

1.3 The Importance of Security Requirements
It is important to know what security requirements are, because the issue of their definition in actual applications is not trivial. Consider the description of a clinical information system in [12]. The report presents a view of the security goals of a Clinical Information System from the point of view of the doctors. It makes explicit assumptions that the doctors should have control of the system, while the administrators should be subordinate. It is well known that, in many health services, there is a power struggle between doctors and administrators. In a hypothetical system in which that power struggle has not been resolved, we can consider two hypothetical sets of candidate security requirements. In set 1, proposed by the doctors, some actions are considered legitimate for doctors, but prohibited for administrators. In set 2, proposed by the administrators, the situation is reversed; some actions that would have been legitimate by the standards of report 1 are security violations, and vice versa. It cannot be left to the implementers to resolve conflicts between points of view; a requirements document must state unambiguously what is to be allowed or prohibited to whom; i.e. what are the constraints that are to be imposed on the use of functions of

the system. Only then can we analyze the requirements for misuses or abuses.

1.4 Outline of Approach

We propose a strategy which unifies the concepts of the two disciplines of requirements engineering and security engineering. From requirements engineering it takes the concept of functional goals, which are operationalized into functional requirements, with appropriate constraints. From security engineering it takes the concept of assets, together with threats of harm to those assets. Security goals aim to protect assets from those threats, and are operationalized into security requirements, which take the form of (a subset of) the constraints on the functional requirements.

2. THE STRATEGY

We illustrate our strategy in figure 1, which is a UML class diagram showing its structure. The strategy is a meta-model, in the spirit of the KAOS conceptual model [13].

2.1 Requirements Engineering Concepts

On the left-hand side of figure 1 we represent, in simplified terms, some generally accepted concepts of requirements engineering. A business has goals, some of which can naturally be described as functional goals, i.e. to carry out some business task. There are also nonfunctional goals (NFGs) as described in, e.g. [10]. Examples are reliability, usability, safety and, of course, security. We mention non-security NFGs for completeness, but do not attempt to treat them in this paper.

The right- hand side of the diagram is best described in terms of a security risk analysis and management process. See [14] for more detail on security risk analysis and management processes.

In this section we outline existing requirements engineering concepts, which we find useful in our discussion. We have felt free to approximate where precision is not critical for our argument; for example, we use the terms function, operation and system service interchangeably.

2.1.1 Goals and Requirements

Any organization has a number of goals, which drive and control the business, and are stated at a high level. Some of these, e.g. the articles of association of a limited company, define the general activities of the organization, and there are usually more detailed goals stated in policy documents, which describe these activities more precisely. We call these functional goals.

Functional goals are refined, using an approach such as KAOS, until they are made operational, as functions to be performed by agents. We call these functional requirements. These functional requirements may have constraints on them, where a constraint is a limitation of the freedom of performance of a function. In addition to the examples of constraints that support security, which we will give below, other examples of constraint on a function are: a requirement to complete a function within a specified time, in support of a performance requirement; and a requirement to present a function through a GUI, in support of a usability requirement.

2.1.2 Twin Peaks

The concept of "twin peaks", described in [15], emphasizes that requirement and design cannot be separated. Analysis of a requirements specification will lead to a design proposal, and analysis of the design will show the need for further requirements. Thus, as the development process progresses, what started as a small amount of detail within the requirements and design specifications, will broaden out as detail is added.

2.1.3 A Multi-domain Approach

There is ample evidence that security has to be considered in every relevant domain. It is not by chance that Kevin Mitnick, the arch-hacker of recent times, whose exploitation of IP spoofing and other weaknesses in the TCP/IP protocols has given rise to a whole new generation of technical attacks, has written a book on Social Engineering [16].

His attacks illustrate the exploitation of vulnerabilities arising from a combination of the properties of human (procedural), physical and software domains.

Michael Jackson's work on Problem Frames [17] has enabled us to articulate a multi-domain approach. Requirements are about what happens in the world, while software specifications only deal with interfaces. As we emphasize below, security is about protecting real-world assets, while many security techniques are expressed entirely in terms of the behavior of software. So problem frames are an essential element in our exposition of security requirements.

In one respect we differ from Jackson, not because we believe that his approach to his chosen problem area is wrong, but because our concerns are different. He explicitly regards the Machine as the optative3 target of specification, and all other domains as indicative. As we shall show in our discussion of our simple example, we do not take this approach. All kinds of security constraint – physical, procedural and software-based – need to be considered, and probably used in combination.

A consequence of this is that we use Jackon's biddable domains (usually, people) in their true dictionary meaning: "docile; obedient". We accept that they lack "positive predictable causality … the most that can be done is to issue instructions to be followed", but in the security world this is true of computers as much as it is of people. Both computers and people can be programmed or "trained to follow stipulated procedures and can be expected to do so". Both computers and people may fail to follow the procedures and we must allow for this in our security design. This principle is already well established in system safety engineering, see e.g. [18] where a combination of physical, procedural and software safety measures is used, taking into account the likelihood of failure of any of them.

2.1.4 Multiple Domains and Security Principles

There are two principles (see, e.g. [19]) that should be obeyed when designing for secure systems:

- Defense in Depth: it should always be assumed that a constraint is fallible, so if one fails, another should still prevent a successful attack on an asset.
- Diversity of Defense: Defense in Depth is more likely to be successful if the defences that are used are diverse in nature.

It is therefore desirable, whenever possible, to supplement security measures of one kind with those of another; a combination of physical, procedural and software security is likely to be most effective. These principles reinforce the need to take a multi-domain approach.

3. SECURITY RISK ANALYSIS AND MANAGEMENT

Security risk analysis and management are well-established techniques, which were first developed in the 1960s, and have since been refined and developed. Security risk analysis is concerned with identifying and evaluating the risks to a system, and security risk management then makes decisions on appropriate security measures. It is illustrated in Figure 2.

The meanings of terms in this area are not universally agreed. We will use the following

- *Threat*: Harm that can happen to an asset
- *Impact*: A measure of the seriousness of a threat
- *Attack* : A threatening event
- *Attacker*: The agent causing an attack (not necessarily human)
- *Vulnerability*: a weakness in the system that makes an attack more likely to succeed

3.1 Asset Identification

Security is about protecting assets from threats. This is an important point, which has been taken on board more in the area of safety than security. It has been observed that, in safety critical systems, the computer never harms anyone; it is the system as a whole, including an embedded computer, whose malfunction causes harm. It is significant that Nancy Leveson called her book [18] "Safeware", not "Safe Software", even though a large part of it is concerned with how computers can contribute to, or prevent, accidents.

Similarly, in security, the computer in the abstract can never cause any harm. The assets that are affected by a computer only have value in a real world context; a bit stream that is leaked from a computer only does harm when it is information in the hands of a human competitor or enemy, or if it causes an ATM to put my currency notes into the hand of a thief. So, in the same sense that there is no unsafe software, there is no insecure software; there is only software which, together with the environment in which it is embedded, protects assets or exposes them to attacks.

The first step of a security risk analysis process, after deciding on its scope, is the identification of all relevant assets, with the aim of evaluating them. There are many types of asset controlled by a system, including:

- Information
- Money
- Intangibles, such as an organization's confidence and public reputation.

The intangibles may be at least as important as the direct assets; organizations have been destroyed by the loss of public confidence that they have suffered from security incidents.

After performing asset identification, we have the Assets that are at the top of the right-hand column of Figure 1. Our aim in this process is to value, as well as to identify, assets, but we cannot perform the valuation until we have identified the relevant threats.

Figure 2 – Security Risk Analysis and Management

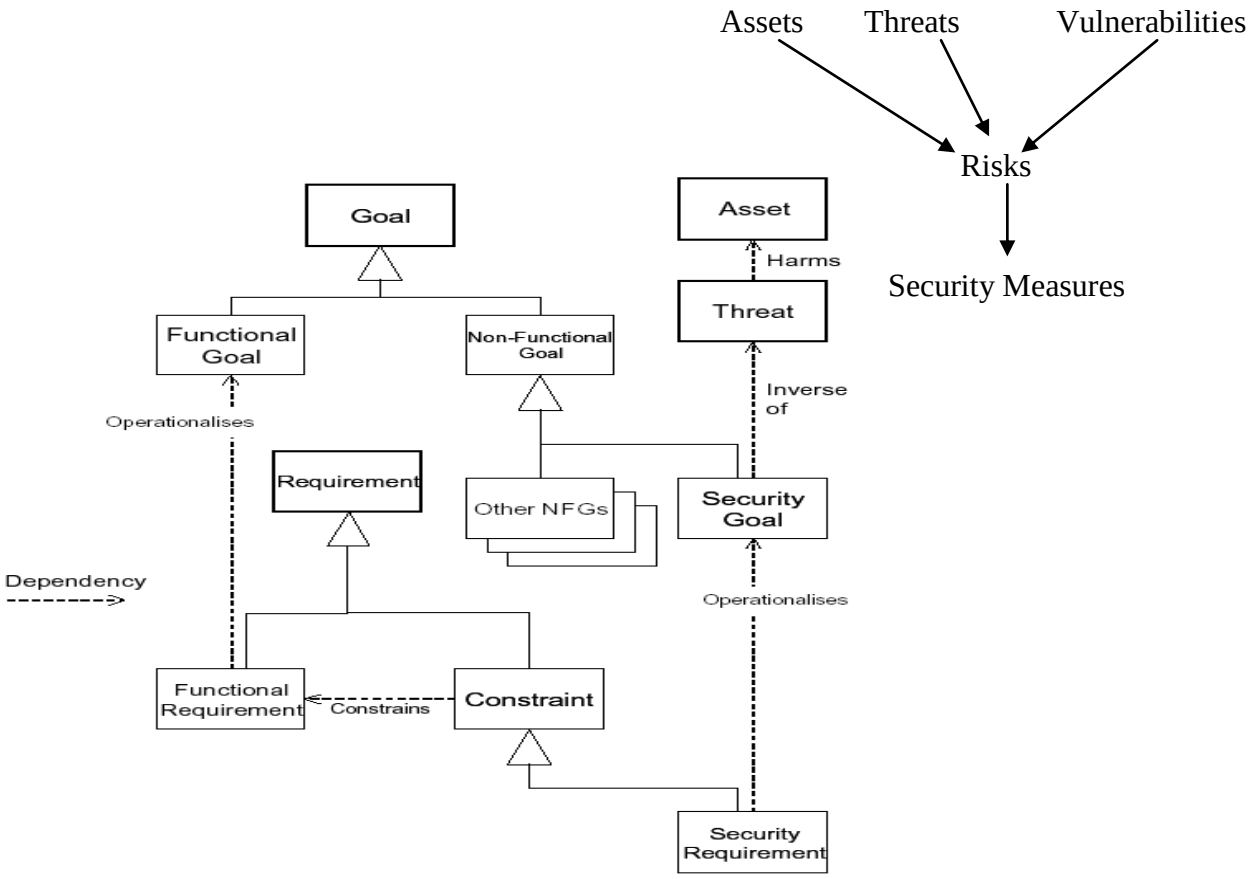

Figure 1 – The Strategy

3.2 Threat Identification & Asset Valuation

In security risk analysis, asset valuation is not a simple matter of determining its market value. It is valued in negative terms, by the amount of harm that would be caused if a particular threat comes to pass. For each asset type, it is necessary to identify the threats that apply. Thus for stored information there are the following possibilities:

- Unauthorized exposure
- Unauthorized alteration
- Loss of availability.

How much impact (harm) will the business suffer if each of these threats comes to pass? The size of the impact will depend upon the nature of the business and the particular item of information; the balance between exposure of information and loss of availability of information will differ widely between a commercial business system and a research organization, for example.

For other asset types the only threat is loss, e.g. money or reputation. It is clear that for many asset types, threat valuation is a very approximate exercise and, although ideally we wish to value in precise money terms, in practice it is necessary to categorise, e.g. does this threat have a Low, Medium or High impact on the organisation?

After performing threat identification and asset valuation, we have the Threats, and the quantified Harm that they can cause to Assets, as shown in Figure 1.

3.3 Priorities Vs. asset values

There is a need to reconcile the risk analysis approach of asset valuation with requirements priorities. The requirements will usually be classified by Priority, into classes such as Essential, Useful and Nice-to-Have. By contrast, asset values (with respect to security threats) have a different scale. The two scales may need to be reconciled, but this creates no difficulty in principle.

3.4 Vulnerability Analysis, Risk Assessment and Security Measures

The remaining steps of the risk analysis and management process are mainly, but not entirely, concerned with the design and implementation stages of the life cycle. We summarize them here.

3.4.1 Vulnerability Analysis

In this step we analyze a baseline system in order to identify its vulnerabilities. For an existing system the baseline will be that system, with its known security measures and weaknesses. For development of a new system the baseline will take into account the security facilities of the envisaged infrastructure, and standard good practice. For each threat, the baseline is analyzed in order to identify the vulnerabilities, i.e. the means of exploiting a threat successfully. We assess levels of likelihood of attempting, and of succeeding in an attack, and combine them into an assessment of the importance of the vulnerability.

3.4.2 Risk Assessment

Risk assessment combines the results of vulnerability analysis with the impact valuation of threats to assets, and reaches an overall conclusion about the level of risk to an asset.

3.4.3 Security Measures

There are several possible responses to risk

- Avoid it completely by withdrawing from an activity
- Accept it and do nothing, if the risk level is trivial
- Reduce it with security measures.

There are a number of possible security measures that reduce vulnerability

- Reduce likelihood of attempt, e.g. publicize security measures in order to deter attackers
- Reduce likelihood of success by preventive measures, e.g. access control, encryption, firewalls
- Reduce impact, e.g. use fire extinguisher / firewall
- Recovery measures, e.g. restoration from backup
- Risk Management identifies the possible security measures, and decides which to choose, based on two main principles:
- Ensure complete coverage
- The expenditure on security measures, and their benefits, should be commensurate with the risks; low risks do not justify high expenditure.

3.4.4 Comment on the Security Risk Analysis and Management Process

The process that is outlined above has served several generations of security professionals well, but has been overtaken by advances in software engineering. In particular, the stages of Vulnerability Analysis, Risk Assessment and Security Measures can be distributed across several stages of the development life cycle, as discussed below. However, the Asset and Threat stages remain valid and useful.

4. SECURITY GOALS

The traditional security risk analysis and management process took us satisfactorily as far as the identification of assets and their valuation against threats but, surprisingly, it does not include the concept of security goals. The next step in our strategy is the identification of security goals, in order to define, at a high level, what we are aiming at to achieve security.

This is a simple step; each threat needs to be inverted to become a goal, by inserting "protection from":

- The threat of unauthorized exposure is converted to the goal of protection from unauthorized exposure, commonly known as Confidentiality
- The threat of unauthorized alteration is converted to the goal of protection from unauthorized alteration, commonly known as Integrity
- The threat of loss of availability is converted to the goal of Availability.

This is a simple step, but it is important because it converts our security concerns into a form that is compatible with the other requirements of an organization, which is an essential step if we are to integrate security requirements into the mainstream process.

We now have the third box in the right hand column of Figure.1.

4.1 Other Security Goals

There are of course other security goals, such as aspects of authentication, which derive directly from other threat types, but are not discussed here because they are not needed for our main argument. More important, in our meta-model we are representing security goals as the inverse of threats. We are leaving it as an open question at present whether there are other security goals, which are not the inverse of threats. For example, is Anonymity a separate goal, or is it better regarded as a means of achieving a goal such as Confidentiality? Work

on understanding the goals of Anonymity will be required to answer that question.

4.2 The Source of Security Goals

Where do security goals come from? The obvious source is as described above, from threats to an organization's assets, e.g. a bank's goal not to lose its own money.

However, they may arise indirectly from other goals. An example of this is a bank's goal to maintain a good reputation. A necessary condition is that it should be able to demonstrate its commitment to protect its customers' money, in addition to its own. Therefore there is a derived goal, which depends upon the bank's Reputation goal, of protecting customers' money. So, eliciting security goals cannot be done from a narrow security perspective; all of the organization's main goals have to be taken into account.

4.3 Characteristics of Security Goals

An organization's security goals have some characteristics which make them harder to manage than functional goals:

- They cannot be immediately discharged by the specification of requirements, but have to be re-interpreted at each iteration of the design.
- They may interact with each other.

4.3.1 Security Goals are not discharged by Security Requirements

At every iteration between requirements and design, whenever a new functional requirement is introduced it must be evaluated against the security goals and appropriate security constraints introduced.

For example, although some constraint may be necessary to achieve the Confidentiality goal, it is not by itself sufficient for the purpose. We will assume that there is also an Availability goal to be achieved, and that one of the means of achieving Availability is to perform regular data backups. A backup functional requirement will be introduced at a later iteration of our requirement. This implies in practice that a copy of the information exists that can be read using a function that has not been defined in our requirements. Unconstrained use of this function can violate the Confidentiality goal, and therefore there will need to be a security requirement that constrains it. At still later a level, engineer's access to the Machine for maintenance purposes can also provide access to the information, using yet another function, which generates yet further security requirements.

The original security goal has not changed, but at each iteration of the requirements, as additional functions are introduced, additional security requirements to constrain the use of those functions are added.

4.3.2 Security Goals Interact

Security goals interact. For example, it might be decided to introduce an encryption function in order to achieve Confidentiality. This is a new function for the system, and its use must be evaluated against all the security goals. One of those goals is Availability, and analysis shows that Availability is threatened by the loss of a secret key. Therefore further measures need to be taken to ensure that the Availability goal is still met, either by ensuring that the secret key is always available or by reconsidering the design decision.

5. SECURITY REQUIREMENTS

We define security requirements to be the constraints, on functional requirements, that are derived from security goals. A simple example is:

The system shall not display salary information except to members of Human Resources Dept.

There may also be temporal constraints:

The system shall not display salary information outside normal office hours; and complex constraints on traces:

The system shall not display information about an organization to any person who has previously accessed information about a competitor organization (the Chinese Wall Security Policy, [20]).

Availability goals will need constraints on response time:

The system shall display salary information within 5 seconds for 99% of requests.

This paper does not claim to provide a complete taxonomy of constraints nor, since this is a strategy rather than a process or method, does it attempt to mandate a specification language.

5.1 Why Define Security Requirements as Constraints?

Note that we do not claim to be *correct* in defining security requirements as constraints on functional requirements; we are proposing a software engineering approach, not carrying out scientific research. Our reasoning for proposing this as a *useful* definition is as follows:

- Requirements specifications, in general, describe the functions (or operations or services) to be provided by a system.
- It is clearly desirable for the specification to describe security requirements in a way that enables them immediately to be related to the functions.
- Constraints upon functions are a natural way to do this.

Other candidate forms for security requirements are:

- *Security goals*. Security goals are necessary as a starting point, but they are more abstract than functional requirements and it would be necessary for the designer to carry out further work, possibly dependent upon requirements domain knowledge that he does not possess, in order to decide how the security goals should constrain the functions.
- *Security functions*. A security function such as encryption is part of the solution, and the specification of security requirements in terms of security functions may lead to a non-optimal design.

It appears to us that, in order to ensure that requirements engineers and system designers each work within their appropriate limits, the appropriate boundary between security requirements engineering and security design is provided by our proposal.

5.2 The Scope of Security Requirements

The scope of security constraints on functional requirements must be global. They are not to be interpreted as constraining any one statement of functional requirement, but of all instances of that function. To illustrate this using a simple example from the case study, assume that there are two functional requirements, derived from two separate business

goals: for members of HR Dept to Display salary information; and for Auditors to Display salary information. Since they are derived from separate goals, the two functional requirements must be kept separate, to enable traceability. However, an additional prohibition, that staff who are suspended are prohibited from Displaying salary information, must clearly apply to both requirements, and to any other requirement for this function.

5.3 How Are Security Requirements Elicited?

In order to derive security requirements, each relevant security goal needs to be examined for possible relevance, and then operationalized constraints must be derived from it. We could do this informally, but goal refinement methods such as KAOS [13] provide a more methodical approach.

5.4 Are there any other Security Requirements?

It is often stated that security is only as strong as its weaknesses, and it is therefore important for it to be complete. We must therefore ask whether, by specifying the constraints on system functions, we have produced a complete set of security requirements?

If we assume that we have a complete statement of the organization's security goals, and have taken them all into account in deriving the constraints, then the answer is Yes. It would be tempting to include another requirement for an application: "and nothing else must happen" so as to ensure that the designers do not assume that they need do nothing else to ensure a secure system.

However, we have no means of expressing what we mean by "nothing else", so we are expressing a general goal, rather than providing a specification, and there should be no statement to this effect as part of the specification. However, we should recognize that the security goals have not been discharged by the specification of constraints on system functions; that is a necessary, but not sufficient, condition for the achievement of the security goals.

This is a proper separation of concerns. To take an example from the case study, the organization has a security goal of Confidentiality of Personnel Information. If it is to achieve this goal, then it will have to state security requirements on a number of activities and domains, including securing the engineer's hardware interface and communications infrastructure. By proposing a new application we have introduced some additional functions by which the security goal could be breached and the security requirements for the new system is properly and completely expressed as appropriate constraints on the functions. When the system requirement results in a design, then the implementation of that design may result in additional functions, adding ways in which security goals could be violated, e.g. through the engineer's hardware interface or through a hacker intercepting communications.

In order to achieve the organization's security goals, additional operational security requirements will need to be derived, from the security goal, for the engineer's system and the communications infrastructure. So, our conclusion is that, if the analysis has been done thoroughly, the security constraints do constitute the complete set of security requirements for the application as far as it is understood at that point. However, the organization's security goals are never discharged until there is an implemented system, and the security goals must be revisited whenever additional functionality is proposed during the course of development.

6. ANALYZING SECURITY REQUIREMENTS

6.1 Internal Analysis of Requirements (Verification)

Security requirements are simply one kind of constraint, and are therefore subject to the same kind of internal analysis as any other kind of constraint. Taken as a whole are the functional requirements and their associated constraints complete and mutually consistent? For example, security constraints can conflict with safety constraints.

This verification activity is not special to security requirements, and we do not discuss it further.

6.2 External Analysis of Security Requirements (Validation)

Even if internal analysis of the requirements has verified that they are consistent, it is still necessary to validate them against the organization's security goals. In particular, will the security constraints actually achieve the security goals?

We give a very simple example of the kind of analysis that could be done at this stage. There is a security requirement that People who are not members of HR Dept are prohibited from displaying salary information. Analysis (in this case informal) shows that this constraint does not prohibit a member of HR Dept who is currently suspended, possibly because of allegations of dishonesty, from displaying salary information. One could argue that this contradicts the security goal of Confidentiality. An additional security constraint, that People who are suspended are prohibited from displaying salary information, could therefore be added.

7. SECURITY REQUIREMENTS AND SECURITY PROPERTIES

Security "properties" are often referred to, especially in formal specifications, and we need to consider how they fit into this strategy. Most security properties are expressed in terms of constraints on traces of the behavior of a system, and this fits in very well with our own view of security requirements as constraints on the operations of a system. It emphasizes that realistic security requirements are likely to be far more complex than the simple constraints that we have used in this paper.

Some security properties may, of course be expressed at a lower level than system requirements, and it will only be possible to discuss them at that lower level.

There are security properties, such as "no covert channels", which do not conform to the constraint model. They are like the "and nothing else must happen" requirement discussed above, and we take the same view, that they are not a concern for system security requirements, but must be addressed at a design or implementation level.

8. CONSTRAINTS AND SECURITY REQUIREMENTS

A set of requirements can contain many constraints on functions, derived from a variety of goals, e.g. constraints arising from all the other NFGs that are relevant to a system, such as performance and reliability. If we examine a constraint, such as the following, how do we know that it is a security requirement?

The machine shall not display Salary Information except to members of HR Dept. The answer is, we cannot identify this as a security requirement from its contents alone. Why not? Consider a hypothetical Payroll Information Display System in an environment in which the honesty and discretion of all users has never been in any possible doubt, so that the organization has no need of any security goals at all. However, it has a goal of

Comprehensibility, and the Payroll Information is so difficult to understand that it is considered essential for all information to be interpreted by members of HR Dept, rather than being directly available to all users. Then, although there is no Confidentiality goal, the constraint has been derived in order to satisfy the Comprehensibility goal, and it would be reasonable to call it a comprehensibility requirement, not a security requirement. From this we conclude that any particular constraint is identified as a security requirement by the source goal from which it is derived, and not from its contents.

9. SOFTWARE SECURITY SPECIFICATIONS

It will be apparent from our argument so far that we do not believe that there can be a software (or machine) security specification independent of the software specification as a whole. We take the view that security requirements are simply one of many constraints on the functions of a system. The functional aspects of a system requirements specification consist of definitions of required behavior and constraints on that behavior (plus traceability information). It is the job of a designer to determine the optative properties of a software machine and possibly of other domains, given his assumptions about the indicative properties of relevant domains, in order to satisfy all the requirements. Some of the properties of the software will be specified to meet security requirements, but that will not necessarily be apparent from the specification itself. However, we can discover that a particular property derives from a security requirement, by using traceability information which provides the rationale for the design.

9.1 Security Functions

A security strategy discussion would not be complete without a mention of security functions. Where do functions such as access control, authentication, encryption, etc, fit in? Our answer is that they are functions (full stop). We use the same argument as for constraints discussed above. If the designer includes a function in order to satisfy a security requirement (i.e. derived from a security goal), then we could reasonably describe it as a security function, but if that same function is used to satisfy some other kind of goal, that is a different matter.

Pursuing the example discussed above, if we have a Comprehensibility requirement that only members of HR Dept are permitted to read Payroll Information, and we decide to implement that using authentication and access control functions, then these functions should be described as Comprehensibility functions. On the other hand, if they are used in their more common role of supporting security requirements, then we will call them security functions.

10. CONCLUSIONS AND FUTURE WORK

This paper has set out a strategy for dealing with security requirements, which has several features:

- Security requirements are derived from security goals, and take the form of constraints on the functions of a system.
- Security requirements are therefore automatically integrated with the system's functional requirements and constraints derived from other sources. For example, if Safety requirements are also defined in terms of constraints on operations, security and safety constraints are expressed in identical terms and the analysis of their interaction is directly possible.
- It is essential to define security requirements in terms of the real- world assets of a system. Their realization in software (and by physical and procedural means also) must then be shown to satisfy these requirements.

We claim that this strategic approach will help requirements and security engineers to understand the place of the various synthetic and analytical activities that have previously been carried out in isolation. The strategy has raised a number of issues, mentioned in the discussion, but we believe that it provides a way forward to the effective co-operation of the two disciplines of requirements and security.

As part of our future work, we plan to implement the strategy in a case study and validate the claims made as a part of the discussion.

11. REFERENCES

[1] Baskerville, R. (1993). "Information Systems Security Design Methods: Implications for Information Systems Development." ACM Computing Surveys **25**(4): 375-414.

[2] Lee, Y., J. Lee, et al. (2002). "Integrating Software Lifecycle Process Standards with Seurity Engineering." Computers & Security **21**(4): 345-355.

[3] Heitmeyer, C. (2001). Applying `Practical' Formal Methods to the Specification and Analysis of Security Properties. Information Assurance in Computer Networks (MMM-ACNS 2001), St. Petersburg, Russia, Springer-Verlag.

[4] McDermott, J. and C. Fox (1999). Using Abuse Case Models for Security Requirements Analysis. Annual Computer Security Applications Conference, Phoenix, Arizona.

[5] Sindre, G. and A. L. Opdahl (2000). Eliciting Security Requirements by Misuse Cases. 37th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-PACIFIC 2000), IEEE Computer Society Press.

[6] Alexander, I. (2002). "Misuse Cases in Systems Engineering." Computing and Control Engineering Journal **13**(6): 289-297.

[7] Liu, L., E. Yu, et al. (2003). Security and Privacy Requirements Analysis within a Social Setting. RE'03 - 11th IEEE International Requirements Engineering Conference, Monterey Bay, CA, USA.

[8] van Lamsweerde, A. and E. Letier (2000). "Handling Obstacles in Goal-Oriented Requirements Engineering." IEEE Transactions on Software Engineering **26**(10): 978-1005.

[9] Antón, A. I. and J. B. Earp (2001). Strategies for Developing Policies and Requirements for Secure E-Commerce Systems. Recent Advances in E-Commerce Security and Privacy. A. K. Ghosh, Kluwer Academic Publishers**:** 29-46.

[10] Kotonya, G. and I. Sommerville (1998). Requirements Engineering - Processes and Techniques, John Wiley. ISBN 0 471 97208 8.

[11] Rushby, J. (2001). Security Requirements Specifications: How and What? Symposium on Requirements Engineering for Information Security (SREIS), Indianapolis.

[12] Anderson, R. (1996). Security in Clinical Information Systems. IEEE Symposium on Security and Privacy, Oakland, CA.

[13] Dardenne, A., A. van Lamsweerde, et al. (1993). "Goal-directed Requirements Acquisition." Science of Computer Programming **20**: 3-50.

[14] Peltier, T. (2001). Information Security Risk Analysis, Auerbach. ISBN 0-8493-0880-1.

[15] Nuseibeh, B. A. (2001). "Weaving Together Requirements and Architectures." IEEE Computer **34**(3): 115-117.

[16] Mitnick, K. (2002). The Art of Deception: Controlling the Human Element of Security, John Wiley & Sons Inc. ISBN 0471237124.

[17] Jackson, M. (2000). Problem Frames: Analysing and Structuring Software DevelopmentProblems, Addison Wesley. ISBN 020159627X.

[18] Leveson, N. G. (1995). Safeware: System Safety and Computers, Addison Wesley. ISBN 02011 19722.

[19] Zwicky, E. D., S. Cooper, et al. (2000). Building Internet Firewalls, O'Reilly UK. ISBN 1565928717.

[20] Brewer, D. F. C. and M. J. Nash (1989). The Chinese Wall Security Policy. IEEE Symposium on Security and Privacy, Oakland, CA, IEEE Computer Society Press.