Information Modeling for Automated Risk Analysis

Howard Chivers

Department of Information Systems, Cranfield University,
Defence Academy of the United Kingdom, Shrivenham, Swindon, SN6 8LA, UK
hrchivers@iee.org

Abstract. Systematic security risk analysis requires an information model which integrates the system design, the security environment (the attackers, security goals etc) and proposed security requirements. Such a model must be scalable to accommodate large systems, and support the efficient discovery of threat paths and the production of risk-based metrics; the modeling approach must balance complexity, scalability and expressiveness. This paper describes such a model; novel features include combining formal information modeling with informal requirements traceability to support the specification of security requirements on incompletely specified services, and the typing of information flow to quantify path exploitability and model communications security.

Keywords: security, risk, model, information, threat, service-oriented, communication.

1 Introduction

Security Risk analysis provides a criterion for the value of security in the business and social context of a system. It is the only viable cost benefit justification for security controls, and is the established basis for information security management standards [1] and methods [2]. There are a range of problems in applying systematic risk analysis to large distributed service-oriented systems, one of which is developing an analytic model which is able to support automated threat analysis.

The SEDAN (Security Design Analysis) framework has been developed to support the risk analysis and security design of large distributed systems. At the core of the framework is an information model, which integrates a system design, usually expressed in UML, with a specification of the security environment, including attackers and security objectives. The primary function of this model is to support automated threat path analysis - finding paths from attackers to critical assets - which is at the heart of risk analysis.

The design of this information model is a compromise between the need for efficiency and scalability, and the need to accurately model a diverse range of security objectives and requirements. The information model described here efficiently interprets the information flow in a system as a graph; however, the needs of risk analysis and requirement modeling have resulted in novel features in how the graph is constructed and used. These include combining a generic model of information-flow with informal requirements traceability, allowing the specification of security requirements on incompletely specified sub-systems, or services, and typing of

H. Leitold and E. Markatos (Eds.): CMS 2006, LNCS 4237, pp. 228 – 239, 2006.

information within the model, to distinguish threat path exploitability and allow the specification of communications security requirements.

The contribution of this paper is that it describes an approach to information modeling specifically designed to support risk analysis. In order to ensure scalability and allow the specification of a wide range of security requirements the model has a number of novel features, including combining formal modeling with informal requirements traceability, and the typing of information flow to distinguish path exploitability, and model communications security.

The information model described in this paper has already been applied in practice, by the production of supporting tools, and their use in the analysis of a complex industrial distributed system [3]. For reasons of space only the information model is described here; the specification of security requirements in the SEDAN framework is published separately [4], together with a worked example. A detailed account of the framework and its application is also available [5], which includes a formal account of the model described in this paper.

This paper is organized as follows: Following a brief description of related work, section 3 describes the information model, how it is related to a system design, and the motivation for combining formal information modeling with informal requirements traceability. Section 4 describes an important extension to the basic model: information typing. Section 5 discusses possible limitations in graph-based modeling and how they are overcome, and section 6 concludes the paper.

Definitions

In this paper a security objective or protection objective is an unwanted outcome for a particular asset, also known as an asset concern; such objectives are the goals of threat path analysis. A security requirement, or control requirement is an operationalized requirement (e.g. an access control), that is part of the specification for a system component, usually in the form of a functional constraint.

This paper is concerned with system protection; security goals also require functions, such as intrusion detection, but these are beyond the scope of the paper.

2 Related Work

Risk analysis approaches were reviewed by Baskerville [6] over a decade ago, and his analysis is still relevant today. He characterizes current methods (including tools, such as CRAMM [7]) as engineering based, but failing to integrate risk analysis with the engineering models of the systems they support. He identifies the need to use abstract models of systems, which allow a combination of security design and stakeholder analysis, independent of physical implementation. This characterization of the problem is one of the motivations for the SEDAN framework, since it identifies the scope for abstract modeling to make a fundamental contribution to risk analysis.

Recent work on risk is typified by the European CORAS research project [8]; this has sought to integrate risk and engineering models by providing process metamodels and threat stereotypes for UML, and by investigating how various methods from the safety and risk community (e.g. failure mode analysis, fault tree analysis) can be utilized in e-commerce risk assessment. Essentially it provides a documentation base

for risk analysis, but no new modeling, and while there are many proponents of threat modeling or analysis (e.g. [9]), these describe good practice and principles, but not abstract models that allow systematic tool support.

Related work, such as UMLSec [10] builds on the formal semantics of UML to allow the complete specification of security problems, which can then be subject to formal proof, or exported to a model-checker. This work is typical of a wide range of formal approaches to security; it is promising for small hard security problems, such as protocol analysis, but there is little evidence that this approach will scale to large-scale practical systems, or that it can accommodate risk as a criterion.

Security risk management is essentially a form of requirements engineering, and the goal-refinement community are active in developing new requirements management models [11], some of which are tool supported. However, this work has yet to accommodate risk metrics, or threat analysis.

In summary, the creation of an effective analytic model for risk analysis is an important open question; the approach described in this paper is unique, since it systematically combines formal modeling and informal requirements traceability.

3 Modeling the Information in a System Design

The purpose of the information model described in this paper is to enable the systematic, automated, discovery of threat paths, and the calculation of other risk-based metrics, in a high-level service-oriented system design. The starting point for the information model is therefore what is represented in such a design:

- the structure of the system: its services (or sub-systems) and data structures;
- interfaces to these services, including the messages that they support; and,
- communication between services: which services are invoked by others.

The following sections describe the basic information model, including how it is mapped to a system, the need to represent security requirements, and resolving the problem of incompletely specified system behavior by combining a generic model of service behavior with traceability to informal security objectives.

3.1 Mapping the System to an Information Flow Graph

Threat path discovery is a form of model checking: it is necessary to expand the information model into a graph, determine paths that correspond to threats, and relate these results back to the system. Following paths in a graph also corresponds to an intuitive model of threat analysis, so the information model is formulated as a graph which represents the information flow in the system.

The system is divided into information carriers and behaviors, which are mapped to graph vertices and directed edges, respectively. Information carriers are data, messages, or events in the functional model; behaviors include system functions or services. The graph is directed, and information paths in the graph may be cyclic. Users (strictly, user roles) are modeled as sources or sinks of information. For example, consider the simple system presented in the UML model in fig 1.

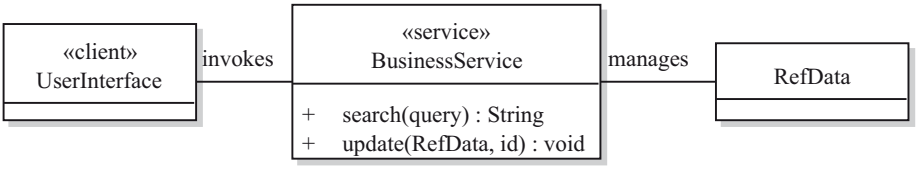

Fig. 1. An Illustrative System

In fig 1 there are two services, the first, stereotyped <<*client*>> is directly accessible by users, and the second (*BusinessService*) provides operations to search and update business data assets (*RefData*). In this system the information carriers are the messages between the two services, that invoke or return data from their operations, the data asset, and messages that flow directly to users.

The corresponding information graph is shown in fig 2, in which the services (*s1*, *s2*) encapsulate graph edges (internal to the services and not shown), and the vertices (*va ... vg*) represent information carriers that are identifiable in the system. The figure is annotated to show how the graph is related to the system in fig 1.

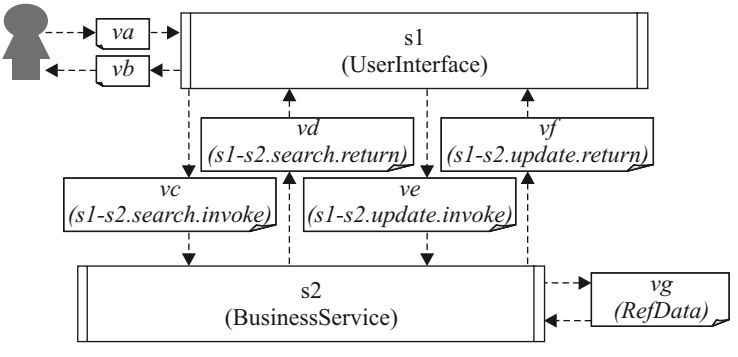

Fig. 2. The Information Graph: vertices represent information carriers such as messages or events; service behavior is characterized by edges that carry information between vertices

Fig. 2 illustrates the mapping between a system and the information model:

- vertices represent information carriers, and have exactly one service as their input and one service as their output;
- the edges of the graph are partitioned between the services of the system. The only information flow between services is via vertices; and
- users are represented as active subjects, similar to services, in that they can invoke operations in client services. However, unlike services, they are sources or sinks of information, rather than graph edges.

Graph vertices are derived directly from information carriers identified in the system model. For example, *vc* and *vd* model the call and return of the *search* operation in the *BusinessService* service (*s2)*, invoked by the *UserInterface* service (*s1)*. Note that

from an information perspective there is usually no need to distinguish the fine grain data structure within a message in information terms (e.g. multiple parameters in the *update*()), but there is a need to represent events that carry no data, such as the *void* return from *update()*; this is discussed further in section 4. This mapping does not imply that messages to and from services are necessarily synchronous.

Vertices can equally represent information exchanged with system users (*va, vb*), or data that is part of a service's state (*vf*); from a threat path discovery perspective, all assets of concern are mapped to graph vertices.

The edges of the graph capture system behavior or functionality, but as noted above, the behavior of services in a system design may be unspecified. Unless a service is constrained by a security requirement, it potentially routes information from all its inputs to all its outputs. For example, in service *s2* it is possible to distinguish nine distinct behaviors (and hence, graph edges) that represent information flow between {*vc, ve, vg*} and {*vd, vf, vg*}. In the absence of security requirements that restrict this behavior, this generic model of a service is used to define the graph edges.

A consequence of this strict division between information carriers and behavior is that a service is never mapped to a graph vertex, which suggests that it can never be the target of a threat path. This is discussed further in section 5.3.

3.2 Modeling Security Requirements

This section describes how the information model supports security requirements; a more detailed explanation of how requirements are specified is published separately [4], together with a worked example.

An essential part of risk management is evaluating a proposed protection profile (a set of security, or control, requirements) to identify residual threats. Some security requirements can be represented in terms of a system design, and some are more difficult. For example, access controls are, in principle, straightforward to specify and model, since they constrain messages that can be identified in the system. However, the specification of constraints on the behavior of services is not as straightforward.

For example, consider the system information graph shown in fig 3; the same symbols are used to denote services and data as in fig 2, but vertices that are not important for the discussion are unlabeled, solid arrowed lines show invocation, dashed lines show other relevant information flow.

In fig. 3, an attacker has access to two services (*sa, sb*) that invoke further services (*sc,sd,se*) to update a data asset (*va*). The security objective is to prevent unauthorized modification of the data asset. There are several paths between the attacker and the asset, so there are a number of options for how the resulting threats can be defended. It is obvious by inspection that unless more is known about the behavior of the services, a single security control is not sufficient to protect all the possible threat paths; at least two are needed, for example *ra* and *re*.

These two control requirements are different in type. For example, service *sa* may be a management interface which normal users (including this attacker) do not need to access: *ra* is an access control. Requirement *re* is unlikely to be as straightforward, since given the system configuration, it is unlikely that all accesses between *sb* and *se* can be prohibited. The requirement *re* must constrain the operation invoked by *sb*, rather than prevent it; perhaps by allowing read-only access to *va*.

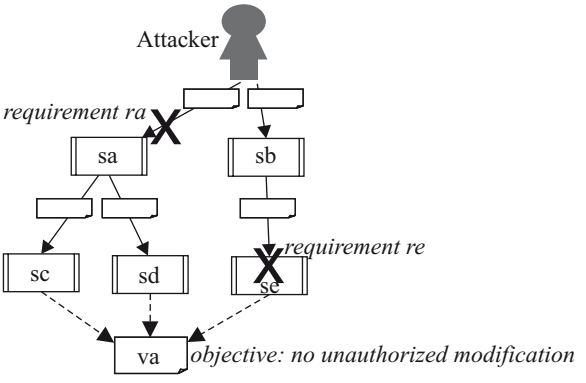

Fig. 3. Attacking a Data Asset via Services

This example highlights the difference between requirements that constrain specific elements of the system model (e.g. access controls, to restrict possible messages) and constraints on elements of the system that are not fully specified[1]: the behavior of services. This also underlines the difference between security and functional requirements: it may have been the intention of the designer that service *se* provides read-only access to *va*, but a systematic analysis will ensure that this is identified and documented as an essential security requirement.

One approach to specifying control requirements on service behavior is to first fully specify the behavior of each service, but this anticipates the design process, reduces the options available to an implementer, and may suffer from the scalability difficulties associated with the use of formal methods in large systems [12]. This paper describes an alternative: the use of generic information-flow constraints, complimented by traceability to informal security objectives. This is the approach described in the next section.

3.3 Deferred Requirements

In fig. 3, requirement *re* has three main components:

service *se* ... must protect *va* from *sb* ... to prevent unauthorized modification

The function of service *se* is not specified, so this requirement cannot be formalized within the domain of discourse provided by the system design. A generalized information flow constraint captures the first two parts of the requirement (constrain information flow between *sb* and *va* in service *se*); however, the security objective (prevent modification) is an important clarification of the requirement.

This type of requirement is described as deferred, because the semantics of the protection objective are informal, and can be properly interpreted only in terms of a detailed functional design or implementation. Deferred requirements have three parts:

[1] In a system design the *interface* to a service (i.e. the messages it receives) is specified, but this is distinct from its *behavior* (i.e. what it actually does).

- the service which must implement the requirement;
- the information context: which messages at the interface to the service, or which assets within the service's state, are constrained; and
- informal semantics, which are specified by reference to the security objective for the associated asset[2].

This approach is a compromise between a formal model of security, and informal requirements management. The former requires a system to be modeled in sufficient detail to allow the specification of any functional constraints, the latter does not benefit from a fully systematic analysis. Essentially, the formal information model defines where requirements are placed in the system, but traceability to informal security objectives are used to clarify what the requirements must achieve.

3.4 Graph Sets

The semantics of deferred requirements are not fully defined in the information model, since part of their specification is informal. As a result, deferred requirements associated with different security objectives are not necessarily comparable. In other words, an information flow constraint traceable to one security objective does not necessarily protect another. For this reason the SEDAN information model is not a single information graph, but a set of graphs, one for each security objective.

This feature is a technical issue for the implementation of the associated model-checker, but does not essentially change the underlying efficiency of the graph-based modeling approach.

3.5 Diverse Security Objectives

The combination of information modeling and informal semantics is able to accommodate a wide range of security objectives. Confidentiality can be interpreted directly in information-flow terms, but most other security objectives are not as easily expressed; for example, integrity has a wide range of different interpretations [13], including no unauthorized changes, or maintaining provenance, or consistency.

These different types of integrity can be treated in a uniform way: the information model allows the discovery of threat paths from attackers to related assets, and security requirements can be placed on these threat paths to protect the security objective; this resolves the problem in information terms, but does not distinguish between different integrity objectives. The implementer is able to determine the detailed protection requirement by traceability to the informal security objective.

This pattern of formal path discovery, and informal requirements traceability, therefore supports a wide range of different security objectives. (See also section 5.2)

4 Information Typing and Communications Security

One security requirement that could be used to preserve integrity (see previous section) is authentication: the source of data is accredited, preventing an attacker from

[2] This does not imply that each asset has distinct security objective; security objectives may apply to groups of assets, or be traceable to higher level goals that specify their purpose.

injecting or substituting false data. However, if the objective were availability, then data authenticity would not protect against a denial of service attack involving high volumes of invalid data.

This is an example of a threat which can be transmitted via system events, or traffic flow; in risk analysis it is necessary to distinguish traffic flow from information carried by data, since they support different threats. Vertices in the information model are therefore typed to characterize the threat paths that may be supported by the vertex. The base types are *data* or *void*[3], corresponding to information carriers that support data or traffic, respectively.

However, the value of information typing extends beyond the need to characterize threat paths; it also allows communications security requirements to be represented in the information model, and this is described in the remainder of this section.

Modeling Communications Security

Attacks via implementation mechanisms (e.g. buffer overflow, or direct access) are common and important, and one purpose of communications security[4] is to protect against such attacks. The associated threat paths can be determined by evaluating the impact of an attacker with direct access to a graph vertex. This requires the type of a vertex to be further qualified by its accessibility to an external attacker. Three additional types are needed: *Confidentiality*, *Integrity*, and *Virtual*. Fig. 4 shows how they are used to model confidentiality; data flow is shown dashed, and traffic flow dotted.

In fig.4, vertex *Vx* represents an encrypted message, resulting in traffic flow from service *s1* to *Vx* (1). An attacker with direct access to the message, for example by wiretapping, is able to extract only traffic information (2). However, if the attacker injects data (3) then this may be inadvertently accepted by service *s2* (4). A *Confidentiality* vertex therefore supports a traffic flow from its source to an external attacker, and a data flow from the attacker to the destination service. The reverse pattern applies to an *Integrity* vertex: an external attacker is able to obtain data, but not misrepresent it.

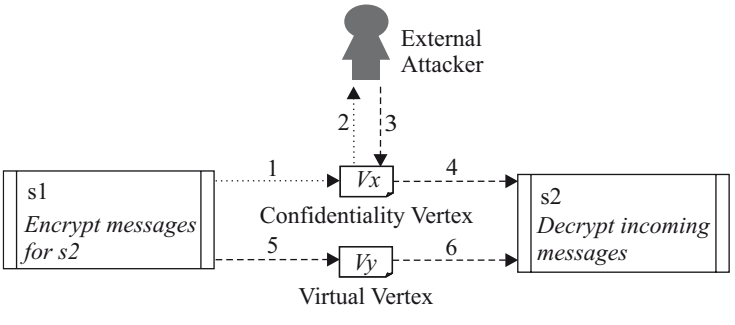

Fig. 4. Vertex Types used to Model Confidentiality

[3] *Void* is named because many related system events are messages with void parameter sets. For example, vertex *vf* in fig 2.

[4] Communications security is also used to protect end-to-end messages from intermediate services; this is also accommodated by the model described here.

The *Confidentiality* type characterizes the communication level message from *s1* to *s2*; however, *s2* obtains data from this message by decryption, unlike an external attacker or intermediate service (not shown), so it is also necessary to add a direct auxiliary information path (5-6 via *Vy*) to model the recovery of message content. This vertex has a special type, *Virtual*, which corresponds to the message layer in the system model, conveys data, and is inaccessible by an external observer.

This describes the main features of information typing: it allows the information model to distinguish between traffic and data flow in the system, and hence allows the modeling of communications and message security in an information graph.

5 Potential Limitations

The interpretation of a system as a set of information graphs provides an efficient basis for threat path analysis on large systems; however, it makes some explicit assumptions about the implementation, and is less expressive that other possible formal expressions of information flow (e.g. as a set of traces). The following sections review the key assumptions and possible limitations.

5.1 Critical Modeling Assumptions

The structure of the information model embodies critical assumptions that must be maintained in an implementation if the risk analysis is to remain valid; these are that:

- a system can be decomposed into services;
- the only information flow between services is via identified information carriers; and
- the structure of services and information carriers is consistent between the system design and its implementation.

These assumptions are appropriate to service-oriented systems that comprise services communicating via messages; individual services may be deployed to physically separate servers, but however they are deployed the implementation is likely to preserve the isolation between services. However, if deployed services are able to communicate via a mechanism that is not present in the design, then this introduces a behavior that was not anticipated in the analysis, with undefined consequences for security.

This is a special case of the general principle that information flow in an implementation must be a subset of that analyzed [14]. These implementation constraints are therefore normal for security analysis and modeling generally, and the system structure described here is well adapted to service-oriented systems.

5.2 Information Representation Limitations

The information model is a graph which is mapped to information carriers in the system design. Such a graph can be made as expressive as necessary, by expanding the number of vertices to enumerate properties of interest; however, in practice there is a need to balance scalability and expressiveness, so the properties enumerated are limited: vertices are distinguished by source, destination or ownership. As a consequence the

information graph does not directly model sequence or time, and this potentially limits the security objectives that can be analyzed.

Consider such an objective: two different users are required to perform an action in sequence; one originates a purchase order, and the second must approve it. The security objective is to avoid an incorrect transaction sequence, because the organization wishes to prevent an approver writing 'blank cheques'.

One or more security requirements are needed in the paths between these users and the payment. The information model can be used to show that a security requirement is correctly placed, because threat paths can be identified between the users and the payment; the system implementer is able to determine that this requires 'correct sequence', because the requirement is traceable to the informal security objective.

The essence of this example is that the information model is able to solve the critical problem in systematic design – the placement of security requirements – without necessarily needing to fully interpret the requirement. This approach is just as effective for temporal constraints as for other constraints on behavior, and mitigates the absence of formal temporal modeling.

5.3 The Representation of Services

A service is modeled as pure behavior: a set graph edges that convey information flow (see section 3.1). It is the vertices of the graph that represent identifiable data items in the system, and are the potential targets of attack. This strict division between information carriers and behavior seems to suggest that a service can never be the target of a threat path.

This potential issue can simply be avoided by explicitly modeling a service's state as data, and identifying that data as a security-critical asset. This approach is occasionally necessary in practice; for example, if the algorithm or software used to perform a service is itself confidential. However, experience suggests that security objectives that system stakeholders wish to assign to services are often misplaced. The most common example is availability: intuitively 'availability of a service' is an appealing security objective; however, what is usually required is availability of the results of the service to the user. From a modeling perspective it is preferable to identify the result as the asset of concern, since the whole information path, including the service, is then the subject of analysis. An abstract model should clarify important aspects of the target system, and this strict mapping of services to behavior prompts the user to identify unambiguously the targets of protection.

6 Conclusions

This paper describes an information model that has been developed to support security risk analysis. The essential structure of the model is an information graph, in which vertices correspond to identifiable information carriers in the system (e.g. messages) and edges represent service behavior. This approach supports a direct mapping between the system design and the information model, and allows the efficient and intuitive analysis of threat paths.

Compared to fully formal system modeling, this approach offers considerable scalability and efficiency, but is potentially less expressive; this problem is overcome by two novel features: the use of informal objectives to clarify security requirements, and information typing.

The need for informal semantics arises because the behavior of services is not defined in a high-level system model. Security requirements on the behavior of such services are specified by a combination of a generic information-flow constraints, and are traceable to the security objective which they support. In effect, the information model determines threat paths and specifies the position of security requirements, while the specific form of protection is clarified by the informal security objective.

This mixture of formal and informal requirements management accommodates a wide range of different security objectives, and also has technical consequences: the information model is a set of graphs, one for each security objective.

Information typing distinguishes between data and traffic flow in the system; this characterizes the exploitability of different threat paths, and allows the modeling of communications, and message-based security.

The information model described in this paper has been used to support practical risk-analysis tooling, and the analysis of a substantial industrial system [3]. The remaining open questions are not concerned with the underlying model, as described here, but with suitable models for established security requirements (similar to fig 3), which are often patterns within the information model.

Acknowledgements

This work was carried out as part of a Royal Academy of Engineering Senior Research Fellowship. We are also grateful to the anonymous referees for their perceptive and constructive contribution to this paper.

References

1. Information Security Management Part 2 Specification for information security management systems, British Standards Institution, BS 7799-2:1999.
2. Risk Management Guide for Information Technology Systems, National Institute of Standards and Technology (NIST), SP 800-30. January 2002. http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf (accessed January 2006)
3. Chivers, H. and Fletcher, M., Applying Security Design Analysis to a Service Based System. Software Practice and Experience: Special Issue on Grid Security, 2005. **35**(9). 873-897.
4. Chivers, H. and Jacob, J. Specifying Information-Flow Controls, Proceedings of the Second International Workshop on Security in Distributed Computing Systems (SDCS) (ICDCSW'05), Columbus, Ohio, USA. IEEE Computer Society, 2005; 114-120.
5. Chivers, H., Security Design Analysis, Thesis at Department of Computer Science, The University of York, York, UK, available on-line at http://www.cs.york.ac.uk/ftpdir/reports/YCST-2006-06.pdf, (accessed July 2006). p. 484. 2006
6. Baskerville, R., Information Systems Security Design Methods: Implications for Information Systems Development. ACM Computing Surveys, 1993. **25**(4). 375-414.

7. CRAMM Risk Assessment Tool Overview, Insight Consulting Limited, available at http://www.cramm.com/riskassesment.htm (accessed May 2005)
8. Dimitrakos, T., Raptis, D., Ritchie, B., and Stølen, K. Model-Based Security Risk Analysis for Web Applications: The CORAS approach, Proceedings of the EuroWeb 2002, St Anne's College, Oxford, UK. (Electronic Workshops in Computing). British Computer Society, available on-line at http://ewic.bcs.org/conferences/2002/euroweb/index.htm (accessed January 2006), 2002.
9. Swiderski, F. and Snyder, W., Threat Modelling. Microsoft Professional. 2004: Microsoft Press.
10. Jürjens, J. Towards Development of Secure Systems Using UMLsec, Proceedings of the Fundamental Approaches to Software Engineering : 4th International Conference, FASE 2001 : Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001, Genova, Italy. (Lecture Notes in Computer Science vol 2029). Springer-Verlag, 2001.
11. Kalloniatis, C. Security Requirements Engineering for e-Government Applications: Analysis of Current Frameworks, Proceedings of the Electronic Government: Third International Conference, EGOV 2004, Zaragoza, Spain. (Lecture Notes in Computer Science vol 3183 / 2004). Springer-Verlag, 2004; 66-71.
12. Schaefer, M. Symbol Security Condition Considered Harmful, Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA. IEEE Computer Society, 1989; 20-46.
13. Mayfield, T., Roskos, J. E., Welke, S. R., and Boone, J. M., Integrity in Automated Information Systems, National Computer Security Center (NCSC), Technical Report 79-91. http://www.radium.ncsc.mil/tpep/library/rainbow/C-TR-79-91.txt (accessed January 2006)
14. Jacob, J. L. On The Derivation of Secure Components, Proceedings of the 1989 IEEE Symposium on Security and Privacy. IEEE Computer Society, 1989; 242-247.