In Proceedings of the *9th International Conference on Telecommunication Systems — Modeling and Analysis*, pages 21–32, ATSMA, IFIP, 2001.

Object-Oriented Security Analysis and Modeling[*]

Peter Herrmann, Heiko Krumm

Universität Dortmund, FB Informatik, LS4, 44221 Dortmund, Germany
Email: {Peter.Herrmann|krumm}@cs.uni-dortmund.de

Abstract

The security analysis and protection of modern distributed information systems has to deal with the complexity, heterogeneity and broad interconnectivity of the systems. With respect to that our approach employs object-oriented modeling techniques in order to facilitate the analysis and to assure its quality even in case of extensive systems. The analysis efforts can concentrate on the creation of a model of the existing system, while threat and weakness identification, risk assessment, and countermeasure planning are substantially supported by automated tool-assistance. The tool moreover adopts conceptions of object-oriented design tools like the utilization of predefined class libraries and the use of graphical UML-based class and instance diagrams. Therefore the tool already supports the comfortable model creation. The following tasks correspond to model analysis, refinement and augmentation. They are supported by automated tool functions which apply enhanced object-oriented techniques like multiple class hierarchies, object patterns, and graph rewrite system based transformation rules. We report on the principles of the approach and clarify its application by means of an example.

Key Words: security analysis, risk analysis, risk assessment

1 Introduction

The gravity of the information system security issue is rapidly growing. On the one hand there is an increasing deployment of information technology. Thus, institutions as well as persons more and more depend on the secure and reliable operation of information systems. On the other hand the vulnerability of the systems is growing due to their increasing size and complexity. That trend is strengthened in particular by the internet-based distribution, inter-system operation, and remote accessibility of modern systems. In fact, the network technology enables new and important applications (e.g., e-commerce applications). Nevertheless it is accompanied with various new threats. So, the distribution of functions and the interoperation of systems result in a wider spectrum of points of attack and in a broader range of possible attack effects since there may be a large series of connected components and even locally restricted attacks can potentially influence all parts of a system as well as other cooperating systems. Moreover the internet accessibility of systems may be utilized by a growing community of attackers.

Therefore an urgent needs for the security analysis and protection of network-based distributed information systems exists with respect to the audit of existing systems, to the introduction of countermeasures, and to the design of secure new systems. The analysis, however, is expensive and laborious due to the heterogeneity and complexity of the systems. Well-educated specialists have to analyze the systems in detail under consideration of extensive recommendations and standards (e.g., in order to establish baseline protection [7] or to certify the state of security [14, 20]). Moreover they have to be aware of current developments and consult rapidly growing information bases (e.g., incident notes, vulnerability notes, and advisories, cf. [10]).

Many procedures for the analysis and design of secure systems were proposed already [3]. Typically one passes — possibly under iterations — through a series of phases which are devoted to following subtasks:

- Identification of the system, of its structure and its components,
- valuation of the assets contained in the system and definition of the security objectives,
- identification of weaknesses and threats,
- assessment of resulting risks,
- planning, design, and evaluation of suitable countermeasures.

Our approach also supports these tasks. In particular, it facilitates the security analysis, reduces its expenses, and assures its quality by means of

[*]This work was funded by the German research foundation DFG

object-oriented modeling techniques. We apply graphical system models in form of object class and object instance diagrams as they are defined in the well-known Unified Modeling Language approach [4]. A special interactive tool provides modeling and analysis assistance.

In our approach expensive human efforts can concentrate on the first two phases, on the system identification and on the definition of the security objectives. For both purposes one develops and augments a graphically defined object-oriented system model. The development is supported by interactive tool-functions. Moreover, predefined class libraries exist. They supply a suitable conception for the architecture of models and provide for the detailed definitions of the object types needed.

The other three phases are performed under substantial support of automated tool functions where the automation is enabled by the detailed structure of the object-oriented system model. The functions evaluate the class memberships, attribute values, and association structures of the object instances of the model. Based on that information the functions can identify possible weaknesses and threats and introduce object representations of them into the model. In the next phase then these augmentations of the model enable automated risk assessment support. Due to the associations between threats and assets, the tool identifies the risks automatically and guides their interactive valuation. In the last phase the tool automatically detects where countermeasures are of interest. It proposes countermeasures and their location. Moreover it guides the interactive valuation of their costs. Further iterations of the phases can be employed for the refinement of the model and for the analysis of residual risks.

In more detail the automated tool functions are based on enhanced object-oriented techniques which are employed in addition to graphical model descriptions and class libraries:

- Object classes define security relevant attributes and associations. Moreover they define methods for automated class-specific analysis and evaluation.

- The class libraries supply multiple class hierarchies where each hierarchy is devoted to the modeling of special aspects. In combination with multiple inheritance that supports comfortable modeling under separation of concerns.

- By means of object configuration patterns recurring scenarios are defined, which are used as starting points for special automated analysis and evaluation procedures.

- For the automated refinement and augmentation of models the approach of graph rewriting systems is applied (e.g., [1]). A combination of an object configuration pattern with an applicability condition serves as enabling condition of a transformation rule. A second object configuration pattern defines the result.

- Libraries of scenarios and rewrite rules are supplied in addition to class libraries. They support the comfortable integration of known threat scenarios and countermeasures.

Moreover a set of interesting perspectives supplements the advantages of our approach. So, general modeling conceptions as they are defined in security certification standards (e.g., Common Criteria [20] and ITSEC [14]) can directly be supported by means of corresponding class libraries. Likewise, one can represent incident notes, vulnerability notes, and advisories by object pattern and rewrite rule libraries thus enabling their automated tool-assisted consideration during the analysis of special systems. Furthermore we study the introduction of a more abstract model layer which serves for the representation of abstract security policies. Finally we should mention that object-oriented system models can also be utilized to support the second major task in the provision of secure systems, namely the proper operation and management of the security services. With respect to this, [26] reports on an approach for the model-based management of firewall configurations. The approach can be extended to the general management of security services. It can share tool functions and system models with our approach. Therefore management functions may efficiently reuse analysis information. An integration of both approaches can result in a technique for the analysis-based design of management procedures.

In the sequel we firstly give a short overview of related work, in particular of other security analysis approaches. Thereafter we concentrate on the principles of our new approach and successively enter into the different phases of object-oriented security analysis. For each phase we discuss its objectives and functions. We outline the corresponding modeling and tool support. Moreover we elucidate the application by means of the example of the small enterprise IT system proposed in [31].

2 Security Analysis

Evaluating the assets of computer systems, data, and networks, determining the risks of malicious attacks on these assets, as well as suggesting suitable countermeasures against the attacks form an important field of research since the early seventies. Baskerville delineates three generations of security analysis methods [3]. The first generation are methods based on checklists. Here, a system is scrutinized for the existence of every conceivable countermeasure by means of a checklist. If a countermeasure is not available, its necessity is examined by means of a risk analysis where the risk for an asset is calculated from the asset's value and the likelihood of a successful attack (e.g. [13]). Examples of the checklist-based method are SAFE [22], the Computer Security Handbook [19], and AFIPS [5]. Tools based on this method comprise [2, 6, 8, 17, 18, 30, 33]. The main drawback of this generation is the informal and non-structured way of analysis which is hardly scalable to more complex computer systems.

This weakness is addressed by the second generation of security analysis methods which is called mechanistic engineering method [3]. This method focus on identifying and solving detailed function system requirements facilitating the reduction of a complex system analysis into easier manageable system requirement examinations. In particular, system assets, threats on the assets, and countermeasures against these threats are identified and evaluated consecutively. Thereafter a risk analysis is performed depending on which suitable countermeasures are selected and implemented. This method was introduced by Parker [28] and Fisher [16]. A well-known tool is CRAMM (e.g., [11]) provided by the UK Government. Here, in a first step examiners scrutinize a computer system for its assets by means of structured checklist-based interviews with the system owners. Based on the results CRAMM develops further questionnaires for determining the threats on the assets which have to be filled by interviewing the system owners, too. In a third step, the existing countermeasures of a system are delineated by further checklists. Based on the results, CRAMM finally suggests suitable countermeasures. Other tools based on mechanistic engineering are RISKPAC [12], BDSS [27], and CBISA [15]. While these tools are useful even for very complex systems, the complexity of the analysis process urges for expensive security expert teams performing the examinations. Moreover, due to the isolation of the security-based system analysis from the functional design process, each major system modification calls for a complete new security analysis.

The third generation of so-called logical transformational systems intends to overcome these shortcomings by introducing abstract modeling of systems and security-related requirements. An abstract model forms the basis for the introduction of suitable problem-solutions which are elaborated by model modifications. Finally, the abstract solutions are refined to implementable countermeasures. The extension SSADM of the tool CRAMM [9] is an early solution of this idea. Here, abstract specifications of a system, its problems, the security requirements, and possible technical options are developed in parallel to the CRAMM interviews. The reviewing process and the creation of questionnaires are guided by the specifications. Another approach is Baskerville's logical control design method [2] where relevant assets of a system and the threats to them are modeled in a process-like way and collected in a dictionary. Depending on the threats for each process cross-references to special processes modeling countermeasures are set which are guiding the system implementation. More recent approaches concentrate on formal modeling of processes and requirements. For instance, Kienzle and Wulf propose the use of hierarchical organized trees which are called Methodically Organized Argument Trees (MOAT) as a method to assess security of computer systems [21]. In a first step security requirements are defined which form MOAT roots. In subsequent steps the leaves of a MOAT describing unjustified assumptions are refined resp. decomposed into subgoals or alternatives. Thereafter the leaves are justified either by formal verification or by informal plausibility checks according to the risk for the modeled requirement. A similar method is the harmonizer approach of Leiwo and Zheng [25]. Here, the security requirements of a company are modeled formally by tuples each stating a pair of organization elements, a communication between the elements, rules for the messages via the channel, an algorithm protecting the messages, and parameters governing the algorithm. Moreover, the system contains so-called harmonization functions which create, delete, or modify a tuple depending on the existing set of tuples. The security analysis is performed by iterative application of harmonization functions on an initial set of requirement tuples.

A major drawback of these approaches is that they root in abstract requirement descriptions. Thus they support the development of secure systems but are hardly suitable to the analysis of existing systems. This weakness is addressed by the Risk Data Repository (RDR) approach of Kwok and Longley [24] which centers on supporting security officers to maintain existing systems. The RDR consists of various domains describing relevant elements of a computer system, mappings between domains, and countermeasure diagrams. An existent system is modeled by parameterizing RDR entries based on a hypertext system [23]. Thereafter the countermeasure diagrams are used to delineate threats on systems elements and to suggest suitable countermeasures.

Our approach of object-oriented security analysis can be classified as a third generation system since it applies abstract modeling, model-based analysis, and logical transformation. While the existing approaches are based on classical data base and information system techniques like dictionaries, data repositories, relation tables, and decision trees, we apply explicit object-oriented modeling and enhanced object-oriented techniques. The tool support adopts the conceptions of typical object-oriented design tools which are well-established in the field of computer-aided software engineering and support the comfortable interactive design of graphical model definitions (c.f. e.g., the Argo tool [32]). In fact, our tool reuses open-source modules of the Argo project. While we did not find references to other object-oriented security analysis approaches, we have to mention, that the general idea of using object-oriented modeling and graphical system diagrams is not new. Particularly the concluding remarks of [23] end with the statement "The object-oriented paradigm will be more suitable for the security models of large organizations".

3 System identification

UML-based object-oriented modeling [4] is performed in a series of steps. Under utilization of UML class diagrams, at first the modular structure of a system is modeled by design of a hierarchy of classes and links between the classes like associations, aggregations, or inheritance. These models form the basis for specifying objects and object relations which are described by UML object diagrams. Objects are instantiated from the classes and the relations between objects comply with the class links. In subsequent steps, the system behavior, i.e., the internal objects behavior and the interactions between objects are modeled using mainly UML state chart, sequence, and collaboration diagrams. Since a security analysis focuses mainly on the structure of a system, our approach concentrates on describing the system classes, objects, and their links.

In order to facilitate the description of systems, we already provide a set of classes describing relevant system parts (i.e., the UML class diagram in Fig. 1). A basic class *Asset* reflects that every part of a computer system is a potential asset itself. From *Asset* currently four subclasses are inherited

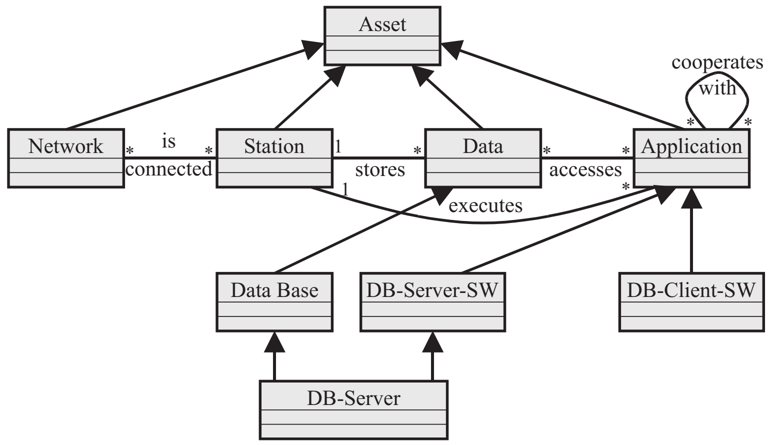


Figure 1: UML class diagram of basic system classes

describing computer networks, stations, data, and applications. Associations between these classes state their relations. A station *stores* data, *executes* applications, and *is connected* with a network while an application *accesses* data and *cooperates with* other applications. From these subclasses more specialized subclasses are inherited. For instance, database clients and servers may be described by the classes *DB-Client-SW* and *DB-Server-SW* which are inherited from *Application*. Since class links are inherited due to the inheritance relation, these two classes can be linked by the association *cooperates with*. Moreover, one can use multiple inheritance for definition of subclasses which own properties of various classes. For instance, the class *DB-Server* models a database server consisting of both, the data storage and the server software. Therefore this class inherits the classes *Application* and *Data*.

A system to be analyzed is modeled by objects instantiated from these classes and object relations based on the class associations. As an example we use a computer system typically used by small enterprises as delineated in [31]. The system consists of a small number of PC-Clients, a PC-Server, and a printer which are connected by an ethernet LAN. The PC-Server also has a direct link to the Internet. The business functions of this system include various database related actions like the storage and processing of customer sales, administration, and financial affairs. The communication

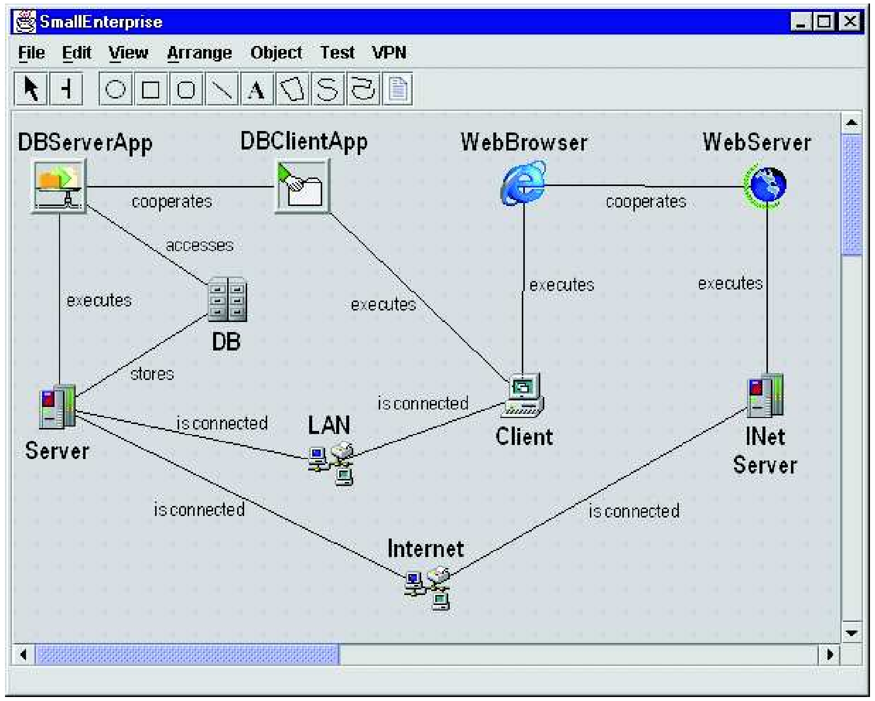


Figure 2: UML object diagram of a small enterprise subsystem

related functions consist of local client server cooperation as well as internet access. In particular, the system provides a company web server used by customers for e-commerce.

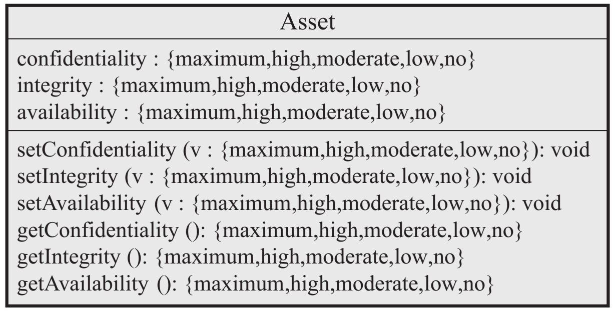
The specification of the small enterprise computer system was developed using the graphical drag-and-drop functions of the ARGO-based tool. Fig. 2 is a screenshot outlining the object diagram of an example subsystem kept small for the sake of simplicity. It contains a PC-server and a PC-Client which are modeled by the objects *Server* and *Client*. These objects are instantiated from descendants of class *Station*. The ethernet LAN connecting the two stations is specified by an object *LAN* of the class *Network*. The subsystem also contains a database system consisting of the set of data as well as the server and client applications. These elements are specified by the objects *DB* (class *Data Base*), *DBServerApp* (class *DB-Server-SW*), and *DBClientApp* (class *DB-Client-SW*). The associations *executes* and *stores* link the elements to particular stations while their internal relations are modeled by the associations *cooperates* and *accesses*. Moreover, the internet access of the server is specified as well. Besides the object *Internet* of class *Network* the subsystem description contains a remote Internet Station *INetServer* and a WWW-Server *WebServer*. Furthermore, the client executes a *WebBrowser* which cooperates directly with the remote WWW-Server.

4 Valuation of assets

After modeling relevant system parts, the protection requirements for these assets have to be determined. Two different methods to evaluate an asset for its degree of protection are common. At first, one can decide the costs to repair an asset attacked successfully. At second, one can define security levels depending on the extent of damage by an attack. For instance, in [7] four security levels *maximum*, *high*, *moderate*, and *low* are defined. The level *maximum* shall be assigned to an asset if its failure “leads to total collapse of the institution or has serious consequences for large part of the society or industry”. Failures of assets rated to level *high* lead to malfunctioning of central areas of the attacked institution which results in a considerable disruption of the institution itself or third parties. The use of the security level *moderate* is recommended if the damage causes normal disruption while attacks on assets of level *low* result only in a minor disruption of an institution. Moreover, for each security level [7] outlines recommendations with regard to the general aspects of data secrecy, correctness of information, and downtime of systems. Our security analysis approach currently uses these four security levels.

Nevertheless, the assignment of a single valuation to an asset is not sufficient since the degree of damage depends on the kind of attack (cf. [24]). For instance, if an enterprise secures crucial data by a capable back-up system but fails to store them encrypted, the removal from the data is only of minor harm since they can easily be restored. Wiretapping, however, may cause significant damage since the thief has no problem to interpret the information contained in the data. Therefore, like [7] our approach uses separate security levels for the confidentiality, integrity, and availability of an asset.

In the object-oriented system model, security levels are assigned by initializing object attributes. As sketched in Fig. 3, the basic class *Asset* contains three attributes *confidentiality*, *integrity*, and *availability* each specifying a security level. The possible values of the attributes are *maximum*,



Asset

confidentiality : {maximum,high,moderate,low,no}
integrity : {maximum,high,moderate,low,no}
availability : {maximum,high,moderate,low,no}

setConfidentiality (v : {maximum,high,moderate,low,no}): void
setIntegrity (v : {maximum,high,moderate,low,no}): void
setAvailability (v : {maximum,high,moderate,low,no}): void
getConfidentiality (): {maximum,high,moderate,low,no}
getIntegrity (): {maximum,high,moderate,low,no}
getAvailability (): {maximum,high,moderate,low,no}

Figure 3: Security level related attributes and methods of basic class *Asset*

high, *moderate*, and *low* stating the selected security level while the value *no* is assigned if no protection requirement is intended. Besides these attributes, *Asset* contains methods for writing and reading them. Since the class is an ancestor of all other system element classes, the attributes and methods are available in every object modeling a system asset.

In the subsystem of the small enterprise example (cf. Fig. 2) we evaluated the station *Server* which stores and maintains the relevant business data as crucial hardware component of the company. As a major damage of this system causes significant disruption, we rate the security level *high* to the three security aspects confidentiality, integrity, and availability. In contrast, the *Client* and the *LAN* are less important for the enterprise. Thus, they are assigned with the level *moderate*. The Internet and the remote internet host are not evaluated in this analysis and therefore rated with the level *no*. Since the data base *DB* is assumed to be important for the enterprise, we evaluated the three security aspects each with the level *high*. The data base client and server applications as well as the web browser are standard software products which can be restored easily. Since, moreover, it is only available as machine code, its confidentiality and availability levels are rated *low*. With respect to integrity, however, these assets are more vulnerable. For instance, a virus infection may threat significant system parts as the server or the data base and also customer computers. Thus, the integrity of these systems are rated with the security level *high*.

5 Identification of weaknesses and threats

The preceding two phases have identified the system with respect to its direct constituents, their values and their relations. Now we want to complement the system model by representations of the existing threats since organizational shortcomings, human failures, technical failures, deliberate acts, and force majeur may have negative impacts on assets of the system. We have to consider that persons and groups of persons — so-called principals — exist who can influence the system. In particular, besides of users, asset owners, and administrators there are attackers who originate deliberate act based threats.

The object-oriented modeling reflects that by means of two object class hierarchies. Subclasses of class *Principal* correspond to the different types of persons, subclasses of *Threat* represent the relevant types of shortcomings, failures, attacks, and incidents. Moreover associations between principals and assets are introduced modeling e.g., that an owner *owns* an asset, that a user *accesses* a resource, or that an attacker *misuses* an application.

In that setting the threat identification shall result in an augmentation of the system model by threat objects, principal objects, and associations which model relevant threats. In general not all threats are relevant to all assets of a system. Rather a threat is relevant only if a weakness exists which provides for a suitable point of attack. Therefore threat identification coincides with weakness identification and we can utilize the observation that weaknesses correspond to the occurrence of certain subsystems within the system. In the simplest case already the existence of one object of a certain type implies a weakness, e.g., the occurrence of an open network comes along with direct eavesdropping opportunities for network based attackers. In the more complex cases some possible interrelations between different objects of a subsystem form the weaknesses, e.g., a network-based attacker may misuse a network in order to gain control over a station. He then may misuse this station for an integrity attack on data stored in a connected asset.

To enable tool-assisted automated threat identification we represent weaknesses by those object configuration patterns which apply to the corresponding subsystem configurations. The patterns are represented by object instance diagrams. With respect to our example, Fig. 4 depicts a pattern where one principal has a *misuse station* association with a station. The station executes an application which cooperates with another application installed on a second station. The pattern implies the weakness that the principal who misuses the first station may also misuse the second station since he may find a way to gain control on the second station via the two cooperating applications. The scenario which directly represents the resulting misuse association between the principal and the second station

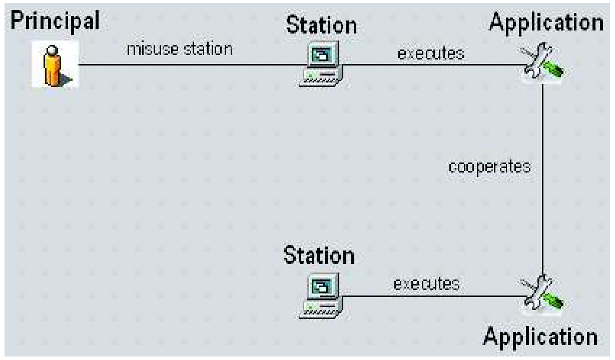


Figure 4: Weakness pattern

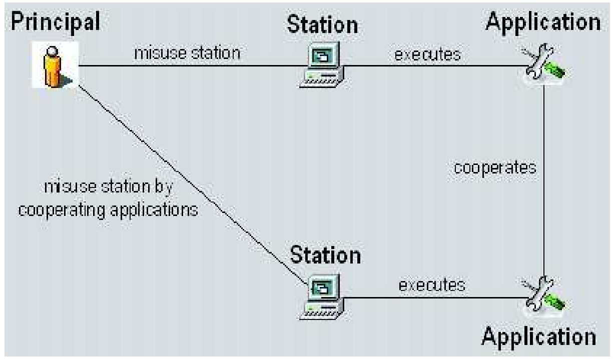


Figure 5: Result pattern

again can be represented by means of an object configuration pattern. The pattern is shown in Fig. 4. We note that both patterns form a pair in the sense that the first pattern triggers the detection of a threat and the second pattern documents a possible effect of this threat. Moreover we note that the second pattern can contain subsystems which can contribute to further threats. So, in the example of Fig. 5 we find the principal now misusing the second station directly. If for instance an application on this station cooperates with a further application on a third station, again the pattern of Fig. 4 can apply. In that way threat identification and effect documentation can be performed repeatedly in order to grasp all indirect threats, too.

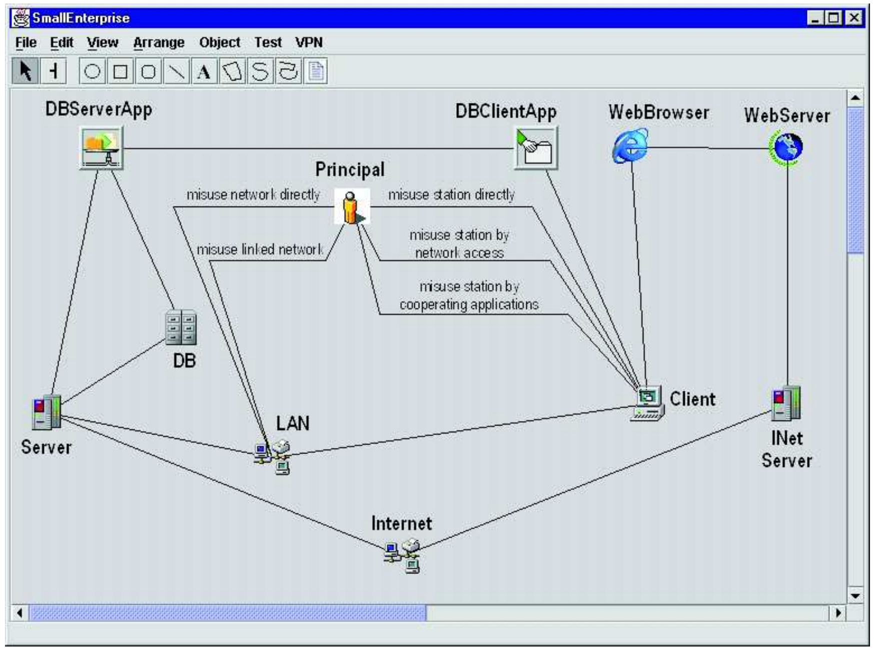


Figure 6: Result of the detection of LAN-based threats

The procedure of repeatedly identifying triggering pattern occurrences in a model and transforming the model in accordance with result patterns, can be implemented by means of a graph transformation system (cf. [1]). A transformation system is defined by a set of transformation rules. Each rule consists of a pattern pair, i.e., a triggering pattern and a result pattern. Moreover a rule can be augmented by an application condition and by effect functions. The application condition refers to properties of the input pattern occurrence. It guards the transformation which can only be applied if the condition is evaluated to true. The effect functions define how attribute values in the resulting pattern occurrence depend on values of the input pattern occurrence. In that way one can represent even subtle triggering configurations and result augmentations.

Applying these principles we developed libraries of transformation rule collections which correspond with the different types of threats. Furthermore the analysis tool implements functions for the inclusion of transformation rule collections and for the repeated application of the rules. Thus, after inclusion of the necessary transformation rules the tool automatically looks for possible rule applications and - under execution of the transformations - augments the model by threat representations. Fig. 6 exemplifies the augmentation. It depicts an intermediate state of the threat analysis of the small enterprise system shown in Fig. 2. The analysis has detected some threats which may be originated by company internal attackers or malicious internet users. Fig. 6 documents that attackers may misuse the LAN, the client station and client-side applications. These misuses cause various confidentiality, integrity, and availability threats on the client and server data resp. applications. For instance, the asset *DB* of the small enterprise subsystem is vulnerable to an eavesdropping attack from an internet-based intruder.

Graph rewriting is also used to assess the likelihood of misusing or attacking assets. In order to add attributes to association links, the UML offers special association classes which are linked to associations by a special link attribute (cf. [4]). We introduce the association class *Threat* which can be linked to associations describing threats on assets. Class *Misuse* is a descendant of *Threat* and is linked to associations stating a misuse of a system part. Both classes contain an attribute *likelihood* describing the probability of an attack on or misuse of an asset using the five values *maximum*, *high*, *moderate*, *low*, and *no*. Some likelihoods of threats have to be assessed manually by the analyst. Others, however, can be calculated by the tool. For instance, the probability of an attack on an asset protected by a safeguard depends on the level of protection (introduced below in Sec. 7) provided by the safeguard.

6 Risk assessment

According to Courtney [13] a risk to an asset depends on the valuation of the asset and the likelihood of an attack on it. In our approach this corresponds to the security level of the asset (cf. Sec. 4) and the level of likelihood of a malicious attack (cf. Sec. 5). A risk to an asset is modeled by an object of the class *Risk*. As delineated in Fig. 7, this class contains an attribute *value* describing the risk level as well as methods for providing the access to this attribute. For the sake of simplicity we use the values *maximum*, *high*, *moderate*, *low*, and *no* for describing risk values, too. *Risk* is associated with three classes. The first class is *Asset* modeling the assessed asset. A second class is the association class *Threat* which describes the threat on the asset and the third class *Misuse* specifies the kind of misuse the threat is

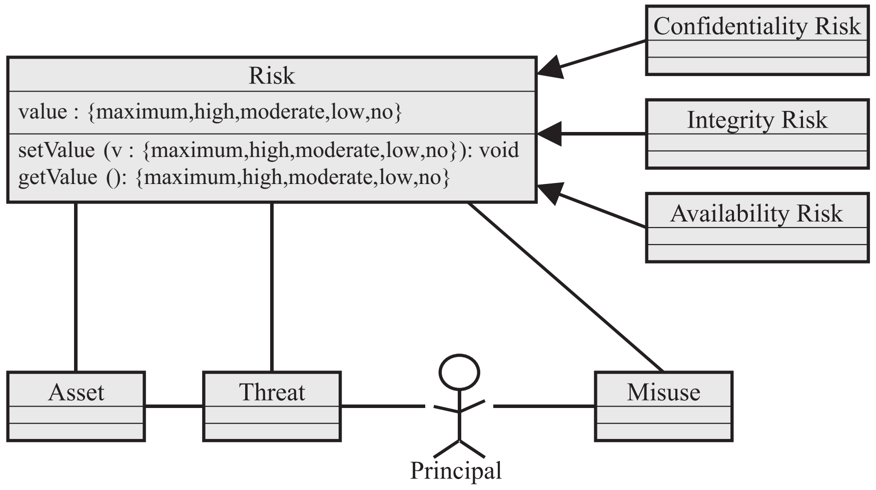

Confidentiality Risk
Risk
value : {maximum,high,moderate,low,no}
setValue (v : {maximum,high,moderate,low,no}): void
getValue (): {maximum,high,moderate,low,no}
Integrity Risk
Availability Risk
Asset
Threat
Misuse
Principal

Figure 7: UML class diagram of the class *Risk* and its links

based on. Since the risk values may differ depending on the kind of threats, three subclasses *Confidentiality Risk*, *Integrity Risk*, and *Availability Risk* are inherited from *Risk*.

In a first assessment step our tool adds instances of the risk subclasses to the system description by graph rewriting. If, for instance, a pattern of an asset object, a threat object instantiated from a descendant of *Confidentiality Threat*, a malicious principal, and a misuse object are linked, an instance of *Confidentiality Risk* is created and associations to the pattern objects are generated. Thereafter the tool determines the risk value based on the security level and the likelihoods of the threat and misuse objects.

	Overall likelihood of attack				
Security level	maximum	high	moderate	low	no
maximum	maximum	maximum	high	moderate	no
high	maximum	high	moderate	low	no
moderate	high	moderate	low	low	no
low	moderate	low	low	no	no
no	no	no	no	no	no

Table 1: Matrix for calculating risk values

At first, the overall likelihood of attack is calculated as the minimum of the threat and misuse likelihoods. This reflects that an intruder must be able to misuse a system part for an attack as well as to attack the asset after gaining access to the system part. Afterwards, the risk level is calculated from the overall likelihood and the security level of the asset according to the matrix outlined in Tab. 1. Risk objects with the value *no* are removed. Finally, the security analyst examines the risk objects and deletes the objects if the risks are considered acceptable. The tool supports also strategies to remove risk objects automatically. For instance, the analyst may request the tool to remove any confidentiality risk if the risk level is not higher than *moderate* and the risk is based on eavesdropping attacks on data. Strategy depending discards are performed by graph rewriting as well. If all risk objects are deleted after the assessment and examination process, the security analysis is finished as the risk to the assets is assumed acceptable and the system is considered as sufficiently secure.

In our subsystem example the object *DB* modeling a data base is linked with three confidentiality risk objects since the model contains an object of class *Eavesdrop Threat* and three misuse objects. The station *Server* can be misused for an attack on *DB* either by direct access, by abusing a cooperating application, or by remote access from the LAN or Internet. The confidentiality security level of *DB* is *high* (cf. Sec. 4). The likelihood of eavesdropping is considered *maximum* since data is not protected by an authentication system and therefore is vulnerable to every station user. Due to a certain amount of trust in the employees and visitors of the enterprise the likelihood of misusing the server directly is considered *moderate*. Since the data base retrieval software, however, enables the transmission of data to *Client* which due to its internet access is an easier hostage, the likelihood of misuse by a cooperating application is assumed as *high*. Due to the internet connection the chance of a remote station misuse is also rated *high*. Thus, two confidentiality risk objects are assigned to the risk level *high* and one to the level *moderate*. As the high risk of eavesdropping essential business data is not acceptable, the analysis has to be continued.

7 Countermeasure introduction

After risk assessment the security analysis proceeds to the selection of suitable countermeasures in order to reduce the risk of attacks against relevant assets. In our approach safeguards are modeled by objects of the class *Countermeasure* (cf. Fig. 8). This class contains attributes to describe the

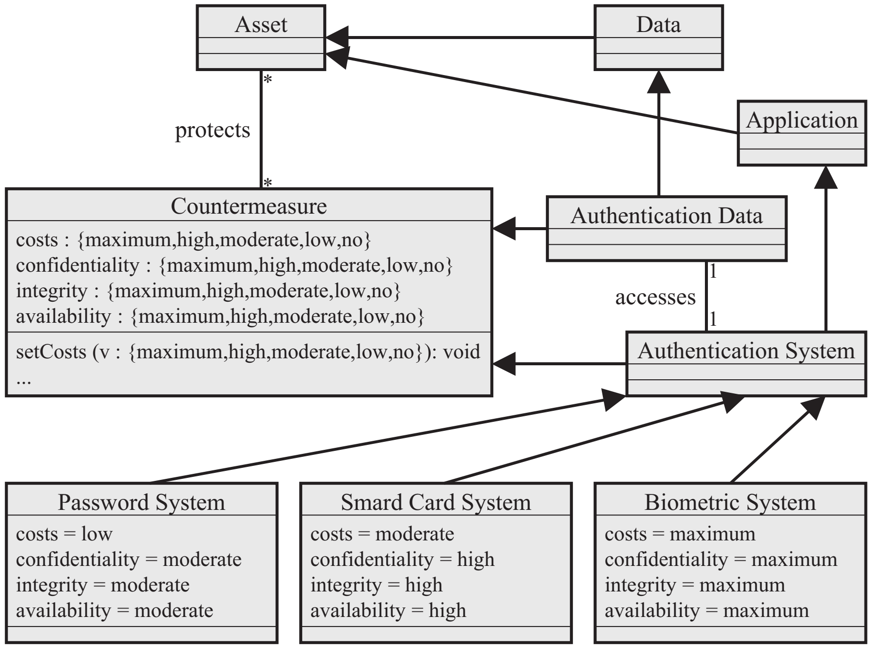


Figure 8: UML class diagram of the class *Countermeasure* and some descendants

cost of deploying the countermeasure and the levels of protection the safeguard provides with respect to confidentiality, integrity, and availability. Like other class attributes, they use the values *maximum*, *high*, *moderate*, *low*, and *no*. An association *protects* links countermeasure objects to the protected asset. *Countermeasure* is an ancestor of several classes describing specific safeguards like a password authentication system. Using multiple inheritance these classes are also descendants of subclasses of *Asset* like *Application* or *Data*. Moreover, the cost and protection level attributes are already assigned with initial values.

The tool provides graph rewriting rules in order to suggest countermeasures against threats. At first, for each safeguard which is appropriate against a threat the corresponding object is created and linked to the protected asset. Thereafter the tool compares the safeguards with respect to their degrees of protection and their costs. All countermeasures are discarded the protect values of which are lower than the risk levels of the asset. If no countermeasures remain, the corresponding asset cannot be guarded accordingly. If there are more than one countermeasure left, the least expensive is selected and the others are removed. As countermeasures are also descendants of the class *Asset*, they own Attributes describing their confidentiality, integrity, and availability related security levels (cf. Sec. 4). In a third step these attributes are initialized in dependence on the risk levels, protection values, and the security levels of the protected asset.

After selecting countermeasures, the extended system has to be analyzed as well since a safeguard itself may be a target of a malicious attack which consequently leads to a severe risk of the protected asset, too. Therefore a further iteration of the security analysis will be performed (cf. Sec. 5).

In our example subsystem the risk of an eavesdropping attack on the data base *DB* was calculated as *high*. A suitable countermeasure is the introduction of an access control system and an authentication system. The authentication system secures that a principal is the person he pretends to be. The access control system limits the access on data or applications only to a certain group of persons. As possible authentication systems the tool suggests a password system, a smart card based system, and a biometric authentication system. The password system offers only *moderate* protection since passwords can be cracked quite easily. In contrast, a smart card based system with PINs (and possibly TANs) can hardly be deceived and therefore is rated *high*. Finally, suitable biometric systems offer *maximum* protection. At first, the tool deletes the password system since its protection with respect to confidentiality is lower than the confidentiality risk level of the asset *DB*. Thereafter the smart card based system is selected since it is cheaper than the biometric system. Likewise, for access control a discretionary access control system is chosen. Fig. 9 outlines the extended subsystem containing the authentication and access control systems together with associated data files. Finally, the security levels of the four new components are selected. Since they protect a data base of security level *high*, their own security levels are also rated with *high*.

8 Conclusion

The preceding description of our approach and in particular the example used there applies to medium grain system models. They represent the major building blocks and asset aggregations of distributed information systems. The size of the model grains reflects the needs of the practical security analysis of current systems which consist of relatively large components. In fact many future information systems will be composed from

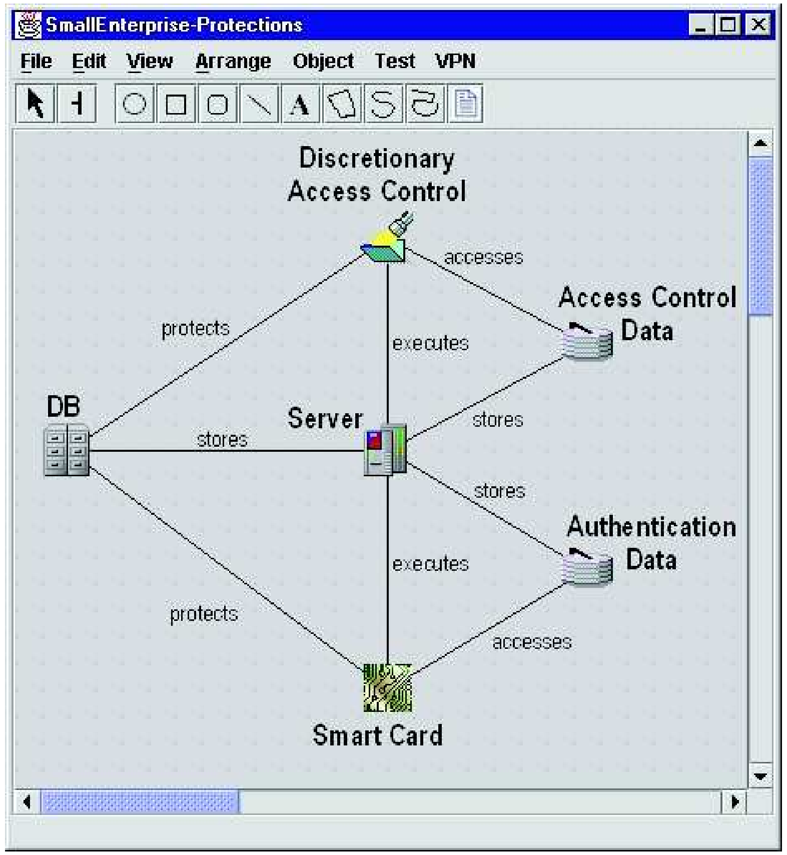


Figure 9: Small enterprise subsystem with authentication and access control systems

so-called software components as they are supported by component models and application server platforms (e.g., CORBA 3 [29]). The software components are of different size, many of them will be of significantly smaller size than current major building blocks. The components will be supplied by different vendors and be offered on an open component market. The information systems will employ a dynamically changing set of software components and various relevant interrelations between components exist. With respect to those software component based systems therefore the careful security analysis has to be based on more fine-grained models. They have to model the system structure as well as the information flow and access control properties in more detail. Moreover they have to deal with new types of countermeasures like policy enforcement modules, component wrappers, and component security information services. Currently we are extending our approach into this direction. First examples show that component systems correspond to highly structured and very extensive models. Therefore we investigate the introduction of abstraction layers into our approach. We expect that the resulting combination of object-orientation, tool-assistance, and subsystem abstraction will contribute to the practical feasibility of careful security analysis procedures for future application server based applications.

References

[1] R. Bardohl, G. Taentzer, M. Minas, and A. Schürr, "Application of graph transformation to visual languages", in **Handbook on Graph Grammars and Computing by Graph Transformation, Volume 2: Applications, Languages and Tools**, chapter 1. World Scientific (1999).

[2] R. Baskerville, "Designing Information Systems Security", Wiley & Sons, Chichester (1988).

[3] R. Baskerville, "Information Systems Design Methods: Implications for Information Systems Development", *ACM Computing Surveys*, vol. 25, no. 4, 375-414 (1993).

[4] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language User Guide", Addison-Wesley Longman (1999).

[5] P. Browne, "Security: Checklist for Computer Center Self-Audits", AFIPS Press, Arlington, (1979).

[6] T. Bui and T. Sivasankaran, "Cost-Effectiveness Modeling for a Decision Support System in Computer Security", *Computer Security*, vol. 6, no. 2, 139-151 (1987).

[7] Bundesamt für Sicherheit in der Informationstechnik, "IT Baseline Protection Manual", www.bsi.de (1999).

[8] J. Carroll and W. MacIver, "Towards an Expert System for Computer Facility Certification", in: **Computer Security A Global Challenge**, North-Holland, Amsterdam, 293-306 (1984).

[9] CCTA, "SSADM-CRAMM Subject Guide for SSADM Version 3 and CRAMM Version 2", London (1991).

[10] CERT, www.cert.org.

[11] W. R. Chisnall, "Applying Risk Analysis Methods to University Systems", in: **Proceedings of the EUNIS 97 Congress**, Grenoble (1997).

[12] Computer Security Consultants, "Using Decision Analysis to Estimate Computer Security Risk", Ridgefield (1988).

[13] R. Courtney, "Security Risk Assessment in Electronic Data Processing", in: **AFIPS Conference Proceedings of the National Computer Conference 46**, AFIPS, Arlington, 97-104 (1977)..

[14] EC advisory group SOGIS (Senior Officials Group — Information Systems Security), "Information Technology Security Evaluation Criteria (ITSEC)" (1991).

[15] T. Finne, "Computer Support for Information Security Analysis in a Small Business Environment", in: **Proceedings of the IFIP TC11 WG 11.2 on Small Systems Security**, ed. J. H. P. Eloff, Samos, 73-88 (1996).

[16] R. Fisher, "Information Systems Security", Prentice-Hall, Englewood Cliffs (1984).

[17] S. Guarro, "Principles and Procedures of the LRAM Approach to Information Systems Risk Analysis and Management", *Computer Security*, vol. 6, no. 6, 493-504 (1987).

[18] L. Hoffman, E. Michelman, and D. Clements, "SECURATE — Security Evaluation and Analysis using Fuzzy Metrics", in: **AFIPS Conference Proceedings of the National Computer Conference 47**, AFIPS, Arlington, 531-540 (1978)

[19] D. Hoyt, "Computer Security Handbook", Macmillan, New York (1973).

[20] ISO/IEC, "Common Criteria for Information Technology Security Evaluation", **International Standard ISO/IEC 15408** (1998).

[21] D. M. Kienzle and W. A. Wulf, "A Practical Approach to Security Assessment", in: **Proceedings of the Workshop New Security Paradigms '97**, Lake District, 5-16 (1997).

[22] L. Krauss, "SAFE: Security Audit and Field Evaluation for Computer Facilities and Information Systems", Amacon, New York (1972).

[23] L. F. Kwok, "Hypertext Information Security Model for Organizations", *Information Management & Computer Security*, vol. 5, no. 4, 138-148 (1997).

[24] L. F. Kwok and D. Longley, "A Security Officer's Workbench", *Computers & Security*, vol. 15, no. 8, 695-705 (1996).

[25] J. Leiwo, C. Gamage, and Y. Zheng, "Harmonizer — A Tool for Processing Information Security Requirements in Organization", in: **Proceedings of the 3rd Nordic Workshop on Secure Computer Systems (NORDSEC'98)**, Trondheim (1998).

[26] I. Lück, C. Schäfer, and H. Krumm, "Model-based Tool-Assitance for Packet-Filter Design", in: **Proceedings of the International Workshop on Policies for Distributed Systems and Networks (Policy 2001)**, eds., M. Sloman, J. Lobo, and E.C. Lupu, Bristol, Springer-Verlag, LNCS 1995, IEEE (2001).

[27] W. Ozier, "Risk Quantification Problems and Bayesian Decision Support System Solutions", *Information Age*, vol. 11, no. 4, 229-234 (1989).

[28] D. Parker, "Computer Security Management", Reston (1981).

[29] J. Siegel, "An Overview of CORBA 3", in: **Proceedings of the Second IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems (DAIS 99)**, eds. L. Kutvonen, M. Tienari, and H. König, Helsinki, Kluwer Academic Publisher, 119-132 (1999).

[30] S. Smith and J. Lim, "An Automated Method for Assessing the Effectiveness of Computer Security Safeguards", in: **Computer Security A Global Challenge**, North-Holland, Amsterdam, 321-328 (1984).

[31] D. Spinellis, S. Kokolakis, and S. Gritzalis, "Security requirements, risks, and recommendations for small enterprise and home-office environments", *Information Management & Computer Security*, vol. 7, no. 3, 121-128 (1999).

[32] Tigris, "ArgoUML Vision", argouml.tigris.org/vision.html.

[33] M. Zviran, J. Hoge, and V. Micucci, "SPAN — a DSS for Security Plan Analysis", *Computer Security*, vol. 9, no. 2, 153-160 (1990).