TOOL-ASSISTED SECURITY ASSESSMENT OF DISTRIBUTED APPLICATIONS

Peter Herrmann, Lars Wiebusch, and Heiko Krumm
Universität Dortmund, FB Informatik, LS IV
D - 44221 Dortmund, Germany

Abstract The CORBA security services support the flexible provision of security features. Their employment, however, has to be tailored to the assets and threats of a system. We relate the corresponding analysis and design of CORBA systems with traditional security analysis, risk assessment, and countermeasure planning as it is in the scope of information system security standards. Since security analysis tends to be difficult and error-prone, we combine that proposal with our object-oriented security analysis and modeling approach. It employs object-oriented modeling techniques and tool-assistance in order to facilitate the analysis and assure its quality even in case of extensive systems.

Keywords: Security analysis, risk assessment, Common Criteria, CORBA security services, object-oriented security analysis

1. INTRODUCTION

The growing importance of the CORBA platform (OMG, 1997) and the existing needs for application security analysis and design plead for the provision of an approach which is devoted to the efficient security analysis of CORBA-based applications. In particular the objectives expressed in the appendix E of the CORBA security services specification (OMG, 2000) aim to the assurance of trustworthiness of information systems. They should be linked with the general international security certification standard conceptions, in particular with the notions and procedures of the Common Criteria (ISO/IEC, 1998). Analysis and design of secure systems, however, is usually expensive and laborious since well-educated specialists have to analyze the systems in detail under consideration of recommendations and standards (e.g., baseline protection cf. BSI, 1999 or certified levels cf. SOGIS, 1991; ISO/IEC, 1998).

Moreover they have continuously to consult the rapidly growing security information bases (e.g., CERT, 2001).

Meanwhile many procedures for the corresponding analysis and design of secure systems were proposed (cf. overview in Baskerville, 1993). They comprise series of phases devoted to system identification, asset valuation and security objective definition, weakness and threat identification, assessment of risks, and finally planning, design, and evaluation of suitable countermeasures. In Herrmann and Krumm, 2001 we proposed the new approach of object-oriented security analysis. While existing approaches are based on classical data base and information system techniques like dictionaries, data repositories, and decision trees, we apply explicit object-oriented modeling and enhanced object-oriented techniques. Our tool adopts the conceptions of object-oriented design tools of computer-aided software engineering and supports the comfortable interactive design of graphical model definitions. In fact, our tool re-uses open-source modules of the Argo project (TIGRIS, 2000). We represent system and security objectives by means of an object instance diagram. Special support results from libraries of predefined object classes which model system component types and define basic methods for automated class-specific analysis and evaluation.

The automated threat analysis and countermeasure introduction of the tool is mainly based on the conception of graph rewrite systems (cf. Bardohl et al., 1999). A rewrite system consists of a set of rewrite rules where a rule is a pair of graph patterns, a pre-pattern and a post-pattern. Moreover an application condition and effect functions can belong to a rule. A rule can be applied to a graph if an instance of its pre-pattern can be found in the graph. The application replaces the instance of the pre-pattern by a corresponding instance of the post-pattern. The patterns are object configuration patterns. So, pre-patterns can correspond to scenarios which come along with vulnerabilities and post-patterns can augment those scenarios with representations of threats.

In the sequel we report on the adaption of class libraries and rewrite rules to the security analysis of CORBA based distributed applications.

2. OBJECT-ORIENTED ANALYSIS

ISO/IEC published a set of common criteria (CC) to standardize security evaluations of IT systems (ISO/IEC, 1998). Moreover it provides a methodology for detecting vulnerabilities and for developing countermeasures in order to reduce the risks. Fig. 1 delineates the main notions of the CC. Computer systems and system components are assets for their owners. The assets are exposed to various security risks since they con-

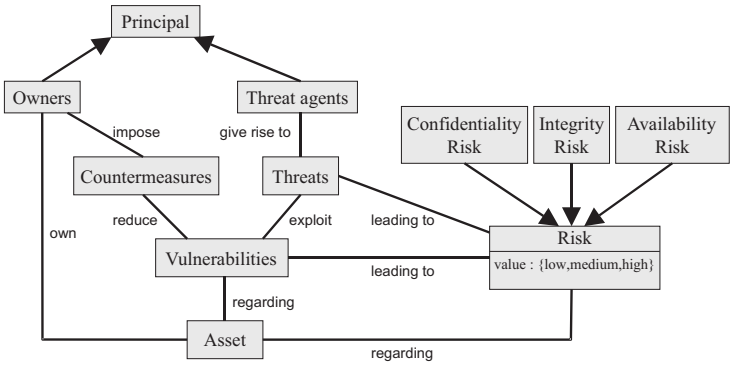

Figure 1 Security classes and associations

tain vulnerabilities which can be utilized by malicious threat agents. To minimize the risks, asset owners impose countermeasures reducing vulnerabilities. The countermeasures, however, may contain vulnerabilities themselves which have to be reduced by other countermeasures.

Our approach supports the CC compliant system identification by a library of asset classes like networks, stations, applications, and data as well as associations between the classes. The tool supports the creation of asset and association instances resulting in a UML object diagram of the system. Thereafter the assets have to be evaluated in order to define the requirements for protection. According to the potential damage each asset is assigned with either one of the four security levels of baseline protection *maximum*, *high*, *moderate*, and *low* or with one of seven evaluation assurance levels introduced in the CC. One can assign different security levels for an asset with respect to each of the three security objectives confidentiality, integrity, and availability.

The next phase identifies firstly vulnerabilities and thereafter threats on the assets. It is supported by libraries of vulnerability classes and threat classes. Moreover two rewrite rule libraries exist. One detects and documents vulnerabilities, the other adds corresponding threats. The interactive valuation of the seriousness of threats concludes the phase.

Then risks are modeled by objects augmenting pairs of assets and vulnerabilities. Moreover, the value of a risk is calculated from the security level of the asset and the seriousness of the vulnerability (cf. Courtney, 1977). That risk assessment is again supported by a class and a rewrite rule library. An interactive classification of risks follows and decides if a risk is not acceptable and must be reduced by countermeasures.

A countermeasure class library and a countermeasure introduction rewrite rule set support the last phase. Attributes of countermeasure objects describe protection levels and prizes which influence the auto-

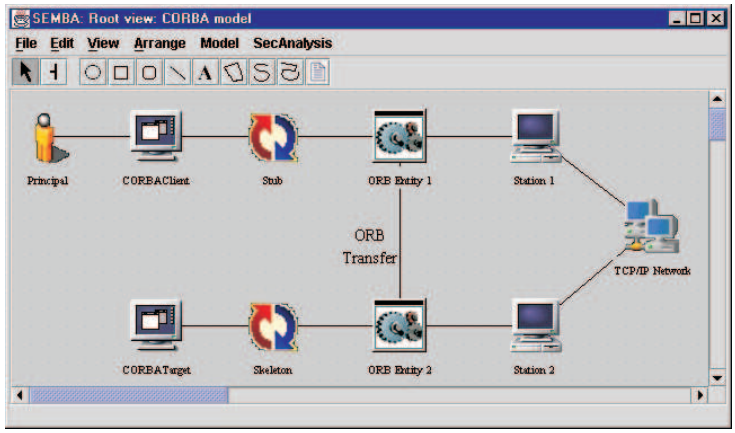

Figure 2 Object model of the CORBA architecture

mated selection of measures and the computation of achieved security levels. Since countermeasures may have vulnerabilities themselves, the analysis has to iterate the phases in order to check the extended system.

3. ANALYSIS OF CORBA APPLICATIONS

In order to enable security analysis according to the CORBA security services specification (OMG, 2000), we extended the set of system asset classes by CORBA-specific components like client and target objects, stubs, skeletons, and ORB components. Moreover classes were added specifying countermeasures like principal-authenticator objects or credential objects. Fig. 2 depicts an UML object diagram of a CORBA-based system. Here, the CORBA client and target objects and the principal accessing the client are described by the objects *Client*, *Target*, and *Principal* while the objects *Stub* and *Skeleton* model the access points of the IDL-based interface. The ORB specification consists of the underlying hardware platform (*Station 1*, *Station 2*, *Network*) and of the ORB software components (*ORB Entity 1*, *ORB Entity 2*).

Next, we will sketch the security analysis of a small part of the system. In a first step, we evaluate the client object and assume that a damage of the client object caused by a malicious attack may lead to a considerable disruption of the institution. Therefore, according to BSI, 1999, we evaluate it with the security level *High*.

Thereafter the tool identifies vulnerabilities of the client object which may cause threats. With respect to user access, objects may be vulnerable in two manners: At first, an object is not able to recognize the true identity of a principal which may be exploited by masquerade based

attacks. At second, an object has to enable access to system resources in order to fulfill given privileges. That may be utilized for an extension of the granted privileges. The tool therefore assigns the two vulnerability objects *Inability to recognize true identity* and *Utilization of not granted privileges* to *Client*. Likewise threat objects are added describing threats based on wrong identities or exceeded privileges. The seriousness of the vulnerabilities is rated to *Maximum* since *Client* is not protected by countermeasures yet.

In the next step the tool assesses the risks for *Client*. The confidentiality, integrity, and availability risks are calculated based on the security level *High* of the object *Client* and the seriousness values *Maximum* of the vulnerabilities. By using a special risk matrix (cf. Herrmann and Krumm, 2001), the tool calculates the value *High* for all three risks of the client. Since these risks are unbearable, suitable countermeasures have to be imposed. Therefore the security analysis is continued.

The CORBA security specification (OMG, 2000) intends the use of a principal-authenticator object and credential objects to provide authentication and access control. The principal-authenticator object authenticates principals requesting access to a CORBA system resource by authentication methods like passwords, smart cards, or biometric systems. Moreover, it maintains an access control list describing the privileges of the particular principal. The principal-authenticator object creates credential objects which can be forwarded to objects the principal wants to access. A credential object testifies that the principal was authenticated by the principal-authenticator object and contains information enabling the accessed object to check the principal's privileges for compliance with its own security policy. If an object does not contain security related functions, the checks are performed by the ORB. In our example, the tool introduces a principal-authenticator object and credential objects to reduce the two vulnerabilities of the client object. Moreover, it has to decide which authentication method to impose. Since password-based authentication systems offer too weak protection for an asset facing a high risk, the tool selects a smart card system which is the least expensive of the remaining alternatives. Finally, the tool assigns the security level *High* to the principal-authenticator object and to the credential objects since these objects guard an asset also rated with *High*.

Since the principal-authenticator and credential objects carry vulnerabilities themselves which intruders may use to reduce their protection, a second iteration of the security analysis is performed for the extended system consisting of the original CORBA example and the added safeguards. This iteration leads to the introduction of unique signatures

and timestamps for credentials in order to protect them against attacks during network transfer.

4. CONCLUDING REMARKS

We considered the needs for standard compliant security assessment of distributed CORBA-based applications and reported on the principles of a new approach for that purpose. It combines object-oriented modeling of CORBA security issues and Common Criteria. Moreover it focuses on tool-assistance and automation. Current work is devoted to the enhancement of the libraries and to practical experiments. Future work aims to the integration of support for the second major task in the provision of secure systems, namely proper operation and management of the security components. It will utilize results from current work on general model-based security management (cf. Lück et al., 2001).

References

Bardohl, R., Taentzer, G., Minas, M., and Schürr, A. (1999). Application of graph transformation to visual languages. In *Handbook on Graph Grammars, Volume 2*, Chapter 1. World Scientific.

Baskerville, R. (1993). Information Systems Design Methods: Implications for Information Systems Development. *ACM Computing Surveys*, 25(4):375–414.

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

BSI (1999). *IT Baseline Protection Manual*. Bundesamt für Sicherheit in der Informationstechnik, www.bsi.de.

CERT (2001). Current information bases and advisories. www.cert.org.

Courtney, R. (1977). Security Risk Assessment in Electronic Data Processing. In *AFIPS Conf. Proc. of the National Computer Conference 46*, pages 97–104, Arlington. AFIPS.

Herrmann, P. and Krumm, H. (2001). Object-oriented security analysis and modeling. In *Proc. 9th International Conference on Telecommunication Systems*, pages 21–32, Dallas. ATSMA, IFIP.

ISO/IEC (1998). *Common Criteria for Information Technology Security Evaluation*. ISO/IEC. International Standard ISO/IEC 15408.

Lück, I., Schäfer, C., and Krumm, H. (2001). Model-based Tool-Assistance for Packet-Filter Design. In *Policies for Distributed Systems and Networks*, LNCS 1995, pages 120–136, Bristol. IEEE, Springer-Verlag.

OMG (1997). *A Discussion of the Object Management Architecture*. Object Management Group (OMG).

OMG (2000). *Security Services Specification, Version 1.5*. Object Management Group (OMG), CORBA.

SOGIS (1991). *Information Technology Security Evaluation Criteria (ITSEC)*. EC advisory group SOGIS.

TIGRIS (2000). *ArgoUML Vision*. Tigris. argouml.tigris.org/vision.html.