The Challenges of CORBA Security

A. Alireza[1], U. Lang[2,3], M. Padelis[4], R. Schreiner[3], and M. Schumacher[4]

[1] T-Nova Deutsche Telekom Innovationsgesellschaft mbH - Technologiezentrum
[2] University of Cambridge
[3] technoSec Ltd.
[4] Darmstadt University of Technology
ameneh.alireza@telekom.de
ulrich.lang@cl.cam.ac.uk
padelis@ito.tu-darmstadt.de
ras@technosec.com
schumacher@ito.tu-darmstadt.de

Abstract Large, distributed applications play an increasingly central role in today's IT environment. The diversity and openness of these systems have given rise to questions of trust and security. It is the aim of the project *Secure TINA* to examine exactly these questions and try to find possible solutions. The focus lies on OMG's *Common Object Request Broker Architecture* (CORBA) as a basis technology for developing distributed systems and on the Security Service specified for it, since this seems to be the most promising technology in the field. The followed approach is thereby twofold. At first, a thorough analysis of the specification itself and known implementations thereof is performed, based also on experiences in the broader area of distributed systems security. At a second, more practical stage, the attempt to develop an own, prototypical implementation of CORBA Security is undertaken, with the main objective of gaining as much practical experience as possible and experimenting with possible alternatives to find a solution to the problems encountered.

1 Introduction

Today, many applications are re-engineered to use Internet technologies. Much attention has been devoted recently to security issues and it is apparent that a high level of security is a fundamental prerequisite for Internet-based transactions, especially in the business-to-business area.

In our collaborative research project *Secure TINA* we address security aspects of distributed, heterogenous systems, especially for the *Telecommunications Information Networking Architecture* (TINA) platform. The basic idea of TINA is to logically separate applications and the communication infrastructure from each other. Another very important issue is to integrate management and control functions into a unified, logical software architecture supported by a single distributed computing platform [Lap98].

As CORBA is the distributed computing platform of choice, we decided to focus on the CORBA Security Service (*CORBAsec*). One of the expected results of this bottom-up approach is, to answer whether or not TINA components can use CORBAsec and to show how CORBAsec can be integrated into TINA.

In this paper we begin with a brief introduction on CORBA and CORBAsec. In the next section we discuss problems that we identified during the analysis of the specification and given CORBAsec products. With the introduction of the project *Mico Security* we describe our approach to filling the gaps between theory (the specification) and real world implementation, as well as gathering practical experience with the application and the implementation of CORBAsec.

2 CORBA and the CORBA Security Service

The technological advances in recent years have led to a situation where large, distributed applications that cooperate with each other are becoming an essential part of IT technology. As a consequence, the need for standard architectures and frameworks for developing such applications has arisen. The Object Management Group (OMG, [OMG00a]) has specified the OMA (Object Management Architecture) in response to these needs; at the heart of OMA lies the CORBA specification ([OMG99], [HV99]).

2.1 CORBA in a Nutshell

The CORBA specification allows programmers to design and implement distributed applications in a standardized manner that guarantees portability and interoperability, following the object-oriented paradigm. The central component in the CORBA architecture is the Object Request Broker (ORB), as depicted in figure 1.

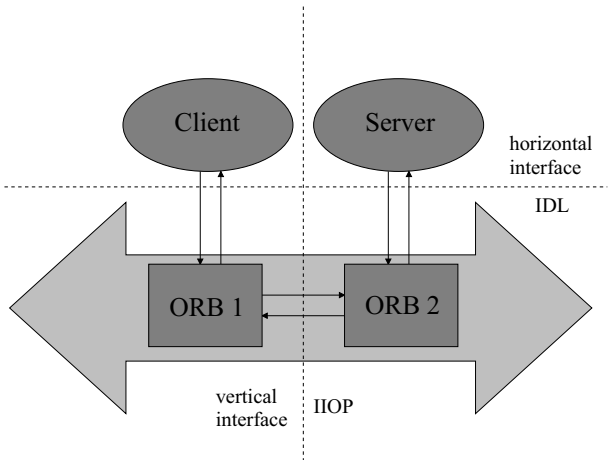

Figure1. Interfaces of the CORBA Architecture

We can distinguish between two distinct interfaces in this architecture. The first is the horizontal interface, which is defined between the application and the ORB. At this

interface the application interacts with the underlying CORBA infrastructure. On the server side, the application implements objects, which provide services to clients in the network. These services are specified in a programming language independent fashion, using the CORBA-specific Interface Definition Language (IDL). On the client side, the application can use the services provided by a server by accessing the appropriate (application-specific) stub. Any programming language can be used for the implementation, as long as an appropriate language mapping from IDL exists. Furthermore, any platform (operating system) can be used for which an ORB system is available. This way, CORBA achieves portability of the application code and reusability for legacy systems.

The second interface is the vertical interface, which is the interaction point of ORB systems installed at different nodes in the network. At this point, a standardized protocol is used by any compliant ORB product. For the normal case of TCP/IP-based networks, this is the Internet Inter-ORB Protocol (IIOP). In this way, interoperability between installations of CORBA products of various vendors (corresponding to independent technological domains) is guaranteed. In a nutshell one could describe the ORB as a software bus, that enables and manages the communication between clients and servers.

The ORB specification represents the backbone of the CORBA architecture and forms the foundation for providing portability, reusability and interoperability. Nevertheless, in many cases it does not provide enough support for developers of distributed applications. When developing distributed applications, one has to cope with several common tasks, such as finding services, guaranteeing transactional processing and/or providing asynchronous communication. These are not done inside the ORB, as this was designed to provide only the core functionality. For this reason, the OMG has specified so-called object services, which reside outside of the ORB core. Examples for CORBA services are the naming service, the trading service, the transaction service, the event service, and the security service; this latter service (i.e. CORBAsec) is the topic of the next chapter.

2.2 Overview of CORBA Security

As described above, the objective of the CORBAsec ([OMG00c], [Bla00]) is to provide security in the ORB environment in the form of an object service. The focus lies hereby on confidentiality, integrity, and accountability. Technically these services are provided through the specification (by the OMG) and the implementation (by a CORBAsec product vendor) of objects, which exhibit a series of interfaces described in CORBA IDL that define precisely the necessary functionality. A high-level overview of these interfaces/ functionality together with the underlying design principles will be given in this chapter.

Terminology and Main Components. In CORBAsec, actors (the users of the system) are described with the term *principals*. Each principal is associated with a *credential*, which collects his *security attributes* (e.g. privileges). Another important term in CORBAsec is that of a *domain*. The most security relevant type of domains in CORBAsec are the *security domains*, which denote a part of the system, where a specific set of *security policies* (rules) is valid.

CORBAsec includes interfaces that define services for the following well-known areas of computer security: *Authentication*, *Message Protection* (including encryption for guaranteeing confidentiality as well as integrity), *Access Control*, *Auditing*, and *Non-repudiation*[1]. The latter two provide means to achieve some degree of accountability. In addition, CORBAsec describes interfaces that can be used for coping with the tedious, but very crucial task of security management/administration (e.g. assigning policies to domains). Furthermore they take care of specific problems with object-oriented (security) systems, such as the delegation of rights.

Design Principles. In order to meet its objectives, CORBAsec's design is based on some general principles. The most important ones are *transparency*, *scalability*, *flexibility*, and *interoperability*, presented in the following.

The functionality of CORBAsec is provided in three distinct *levels* of increasing functionality. The bottom Level 0, which is not part of the specification in the beginning, but was added later on, just integrates SSL into CORBA. Level 1, provides all the above-mentioned security measures except for non-repudiation. This is done in such a way that the applications running on top of the CORBA system are not aware of the security features provided underneath. Level 1 thus covers the *security unaware* applications. On the contrary Level 2 deals with those applications that are *security aware* and that are consequently in a position to interact with CORBAsec in order to specify the exact security features used. For these security aware applications non-repudiation is also relevant.

Note that CORBAsec itself does not provide any security mechanisms (e.g. cryptography). Instead, CORBAsec provides only a standardized architecture and interfaces that can be used to integrate security in a CORBA distributed objects scenario. The concrete security mechanisms (implementations) must be provided separately. Security mechanisms that have been considered so far are Kerberos, SPKM, SESAME, and SSL. It is considered beyond the scope of this paper to give an introduction to all these technologies; the interested reader may refer to the extensive literature in this area, cf.[Gol99] for an introduction. Of course, the actual functionality provided through CORBAsec is mostly determined by the underlying technology. For example, SSL, although extremely popular, is not very powerful, as discussed in more detail in the next chapter. In some cases, as with digital signatures (used for non-repudiation), a more elaborated security infrastructure is needed, such as a Public Key Infrastructure (PKI).

Interoperability is one of the most important design principles of CORBAsec. First is the issue of interoperability between ORB products and CORBAsec products, often described as *security replaceability* and achieved through a well-defined and standardized interface between ORB and CORBAsec. In the current stage of development this interface is defined by *interceptors*, but the work in this field is not completed yet. Second is the interoperability between different CORBAsec implementations. This is similar to the case of interoperability between different ORBs. The solution is also similar: a special standardized communication protocol, SECIOP (Secure Common Inter-ORB Protocol) in our case, is defined. This whole area is covered by the *Common Secure*

[1] This is optional and available only for Level 2 (see below), as non-repudiation makes sense only for security-aware applications.

Interoperability (CSI) part of the specification. In addition to these major notions of interoperability there are a range of other aspects of interoperability, such as interoperability between security domains or underlying security mechanisms.

3 Problems and Weaknesses of CORBA Security

In this section we discuss major problems that have been identified in the current phase of *Secure TINA*. Based on an extensive analysis[2] of the CORBAsec specification and detailed investigations of available implementations of CORBAsec, we found that those issues are not only of a technical, but also of a non-technical nature.

3.1 Application Layer of CORBA Security

Architecture. CORBAsec was initially developed for static applications in restricted environments and cannot be easily adapted to new requirements and trust relationships of Internet-based applications, for example because the code base of CORBAsec is too big, and because firewalls block messages passed between objects. The OMG firewall draft [OMG00d] and the (however proprietary) integration of SSL into CORBA are first attempts to bring CORBAsec to the Internet.

In theory the architecture of CORBAsec follows a *layered approach*, the security functionality can be achieved at the level of the ORB or at the application layer. However, often it is unclear where and how a given security feature should be implemented. For example encryption can often be done most efficiently outside of the CORBA system, if there are no additional security requirements.

More specific weaknesses of the architecture are *covert channels*, e.g. via object references (IORs), which can contain security sensitive data like object keys or names of internal servers. Beside that it is possible to scan a network for unknown objects as an "access denied" exception explicitly discloses the existence of a server.

Authentication and Authorization. Before client and server can exchange messages in a secure way, they must know the identity and other security attributes of the communication peer. The predefined attributes of CORBAsec are limited and cannot describe all properties of a principal, and many security mechanisms do not provide sufficient security attributes.

The authorization model of CORBAsec has several weaknesses [Lan99], such as the predefined access rights that have only limited validity and do not fit to all application scenarios. Thus, many vendors introduce their own access control models, which are not interoperable by nature. Therefore, the only reasonable way to achieve access control seems to be to implement it within the applications.

Note that some problems or conflicts described in this paper are not CORBAsec related, but are a consequence of inherent difficulties and trade-offs in distributed, object-oriented systems; for example, authentication versus delegation, or access control versus inheritance.

[2] Carried out by technoSec Ltd.

Security Audits. The audit service of CORBAsec can be the basis for a reasonable audit mechanism if the following weaknesses are addressed and eliminated: firstly, the specification of the audit functionality is not complete, e.g. security for audit records, effective filtering means, or interfaces for an appropriate analysis are missing. Furthermore, there is no standardized format for audit records or audit trails, so it is impossible to achieve a centralized audit processing. For portability only the attributes of the corresponding CSI level should be used. Unfortunately, this affects the flexibility of the audit service dramatically.

Non-Repudiation. The specification of the non-repudiation service in CORBAsec is also incomplete, requiring components like *delivery authority*, *adjucator* and *secure storage*. The specification refers to other services and standards, but does not mention details of service integration and standards availability explicitly. Even worse, there are no products that implement the *ISO Non-Repudiation Framework* [ISO97] which is the basis for the CORBAsec non-repudiation service, nor a prototype implementation for the referred standard is available.

Besides that, the supported interfaces offer only a subset of the different kinds of evidence. It is only possible to prove that a message was sent or received, but not successfully executed; only a trusted non-repudiation service on the ORB-level can achieve this.

As there is no standard format for non-repudiation evidence tokens, it is impossible to achieve interoperability. The main reason for this may be that there is no proposal for an agreed upon technology.

The non-repudiation service has been announced as a *stand-alone replaceable service*, but in fact it is highly dependent on other components, such as the CORBA TimeService. Both asymmetric and symmetric cryptography can be used, i.e. these methods are replaceable, not the service itself.

Apart from that, there are several other reasons why products that implement the CORBAsec non-repudiation service are not available today. Firstly the CORBAsec non-repudiation service relies on CORBA security level 2 and there are only a few implementations on the market. Secondly it is not mandatory to implement non-repudiation, as it is only declared as an optional extension. Finally, severe legal restrictions prevent the usage of non-repudiation.

Policy Management. The *SecurityAdmin* interfaces provide a rudimentary definition of how policy objects can be accessed and how they can be used. The actual version of the specification does not address many important issues like management of underlying security mechanisms, management of policies at the application layer, support for conflicting policy rules, policy federations, etc. For more sophisticated management of policies the specification refers to *management services* that are part of the CORBA Common Facilities. Again, there is no agreed upon standard, but only an "initial submission" for a *Security Domain Membership Management Service* [OMG00f].

Assurance. In contrast to stand-alone systems, it is very difficult to establish trust in distributed systems, as there are many different components and mechanisms involved.

In addition, trust relationships change frequently, which makes an analysis for assurance very difficult. The CORBAsec specification introduces the notion of a *Distributed Trusted Computing Base* that includes all security-critical components, such as application objects that enforce security, the ORB kernel, object adapters, security interceptors, security mechanisms, hardware, etc. In practice, often only a partial analysis of all components is possible as many components are not under control of the service provider, e.g. client side software.

3.2 Implementation Layer

Interoperability. One of CORBA's most important strengths is its platform and programming language independence, where distributed applications can ideally run on different computers, even on top of different ORBs. In theory the same should be true for secure CORBA. However, practical experience shows that there are obstacles that prevent the interoperability of CORBAsec solutions. As written in [Sch00b] it is up to the ORB vendors to push interoperability, as only they have the chance to move CORBAsec in that direction.

Portable Integration. A portable integration of CORBAsec can be achieved in several ways, through either additional ORB source code, pluggable protocols, or interceptors.

The first approach has the advantage of a straightforward integration, as all required information of the ORB can be accessed directly. On the other hand this always requires the source code of the ORB, therefore third party implementations for most commercial products are not very likely.

Pluggable protocols [KOSP99] represent an abstraction of the ORB's transport mechanisms. This allows for a comfortable way of replacing the transport mechanisms used by the ORB, so that SSL for SSLIOP instead of TCP for IIOP can be plugged in. Pluggable protocols are a convenient way to implement security features like message protection, but unfortunately they are currently not standardized, i.e. each ORB that provides pluggable protocols implements proprietary interfaces.

Interceptors play an important role with regard to an integration of CORBAsec into an ORB, especially if there is no source code available.They are also required to implement other services that have to manipulate individual messages, such as the Transaction Service. In the past, each ORB manufacturer implemented proprietary interceptor interfaces. As this resulted in poor interoperabilty capabilities, the OMG addressed this with a Request for Proposals (RFP) for *Portable Interceptors* [OMG00e]. Unfortunately the agreed submission does not define message level interceptors, which are (amongst other things) required to implement message protection. As a consequence the situation with interoperabilty has not changed at all.

Using any of these approaches, SECIOP can be implemented[3], which is responsible for establishing the *security context* between client and server, as well as protection of subsequent messages between them. It is a very powerful protocol, for example it makes it possible to have multiple secure associations over a single TCP connection,

[3] Alternatively DCEIOP or SSLIOP.

but consequently it is complex and difficult to implement. Besides that it is very demanding on its environment, e.g. both client and server have to be multi-threaded. On the other hand, most of SECIOP's functionality is already provided on other layers of the CORBA protocol stack, and therefore, compared to the complexity of SECIOP, the benefits seem to be minimal. This may be the reason why only a few vendors offer an implementation of SECIOP. As a consequence, a more lightweight approach has been proposed to the OMG [OMG00b].

External Dependencies. CORBAsec cannot be seen in isolation, because it is dependent on external services and additional security infrastructure, such as *NamingService*, *EventService* or *Persistent Object Service*. In the field of CORBA Security, some concepts are quite new and therefore stable interfaces to external components cannot be expected immediately. Besides that it is not always easy to determine how new proposals should fit into the existing CORBAsec architecture, such as the CORBA PKI draft [DST00] and the CORBA firewall draft [OMG00d].

3.3 Non-technical Issues

Misleading Advertisement. An often neglected problem of CORBAsec comes from misleading discussions about the objectives and features of the architecture that are discussed below.

Just as OSF's Distributed Computing Environment (DCE) [Ope00], the predecessor of the CORBAsec architecture (ICL's DAIS Security) was originally developed for huge company intranets that are static by nature.

Since around 1997 CORBAsec has been applied to Internet applications that have different security requirements and trust relationships than intranet applications. Very soon the limitations of the architecture in this new environment were revealed, for example by the lack of support for firewalls or mobile code. In other words, various important features were missing.

Maybe it is an unrealistic concept to define and implement a security service for all possible application scenarios. The approach to introduce different conformance levels does not help, as it was intended for a phase-by-phase adoption of security and not for the support of different types of applications. It could be useful to have subsets of CORBAsec for specific application domains, such as company intranets, electronic commerce, telecommunications or health-care. The current discussions in the OMG on *CORBAsec Light* indicate that this may be the right way to go.

CORBAsec should by no means be seen as a panacea for all security problems, as it essentially does not define any new security functionality. CORBAsec also cannot solve the fundamental difficulties associated with distributed systems security, the architecture suffers under the same problems and conflicting objectives as other solutions for distributed object-oriented systems, such as the conflict of flexibility and guaranteed security.

The usage of CORBAsec per se does not result in a secure system - both the developers and operators of a distributed application have to know what they are doing, and security specialists have to analyze on a case-to-case basis to ensure the effectiveness of

the security enforcement. In summary, it is a more realistic viewpoint to consider CORBAsec as a powerful toolkit for secure, distributed applications rather than a plug-in that automatically secures CORBA systems.

Lack of Experience. CORBAsec was the first security system for object-oriented middleware. It has been developed from scratch with very little previous experience to draw on, and as a result the specification is not mature at this time of writing.

One of the major issues is that the architects of CORBAsec had a wrong estimation of the market trends with respect to future security technologies. During the development of CORBAsec, SESAME [Cla00], the most powerful security mechanism available at that time, has served as a paragon, therefore the basic concepts of SESAME are found in CORBAsec. Unexpectedly the weaker SSL became more widely used than SESAME. Consequently, CORBAsec on top of SSL is not as powerful as CORBAsec on SESAME, and many features simply no longer match.

Based on the fact that there are only a few implementations of CORBAsec that offer the full functionality, most CORBA developers lack experience with regard to the utilization of most advanced CORBAsec features. The only way to solve this problem is to gather practical experience and to adapt CORBAsec to today's requirements.

4 Project: MICO Security

Consequently of the lack of usable CORBAsec products, it was decided to implement a CORBAsec prototype for MICO [PR00] as part of *Secure TINA*. The main goal of MICOsec (MICO Security Service) was to gain practical experience with securing CORBA applications, in particular already existing applications which need to be secured a posteriori.

4.1 Why MICO?

The main objective of MICO (MICO Is CORBA) is to provide a freely available and fully compliant implementation of the CORBA standard. The clear micro-kernel based architecture allows for extensibility and customization for different environments. Note that MICO has been branded as CORBA compliant by the OpenGroup, thus demonstrating that open-source software can indeed produce industrial strength software.

As outlined in [Sch00a], open-source software is a good way to achieve reliability and secure IT systems supporting the business needs of many companies. In fact, a source code analysis of MICO with ITS4 [VBKM00] revealed that MICO does not contain very security critical code.

As MICO provides only a C++ language mapping, we decided to show interoperability of the MICOsec prototype with another CORBAsec product. Adiron's ORBAsec SL2 [Adi00], which is based on Java, was chosen as the peer security service implementation to which interoperability should be established as part of the project. Both implementations are based on SSL and Adiron has also signaled an interest in participating in such an experiment.

4.2 Prototype Implementation

The long-term goal of the MICOsec prototype project is to implement as much of the CORBAsec Level 2 functionality as possible to gain experience with this new technology and to identify potential pitfalls. An incremental approach to implementing the various parts of the Level 2 functionality was chosen, and at the current stage of the project the basic functionality for authentication and message protection is implemented. SSL was chosen as the basic security mechanism, firstly because various open-source SSL implementations are available, and secondly because most other CORBAsec products are based on it.

The IDL interfaces used for MICOsec are currently based on the CORBAsec 1.7 draft [OMG00c] which also closely resembles the ORBAsec SL2 interfaces. The findings of an extensive CORBAsec analysis carried out at an earlier stage of *Secure TINA* are also taken into account during the implementation, in particular with respect to the access control model and the representation of security attributes.

As a proof of concept, our MICOsec prototype will be used to secure an online auctioning system that will make use of all implemented parts of the security service. The auctioning application, which is implemented in C++, will be accessible through many platforms supported by MICO, such as Tcl/Tk based clients written for Windows or Linux. In addition, it is planned to integrate mobile devices like the Palmpilot [Pud00] at a later stage of the project,.

At the time of this writing, the basic authentication and message protection functionality has been implemented and tested. Subject to the successful completion of *Secure TINA*, the full MICOsec implementation is anticipated to be completed around mid-2001[4] and will include some of the following security functionality: CORBAsec Level 2 access control and auditing, a PKI interface both at the ORB and application layers based on the upcoming OMG PKI standard, as well as a limited non-repudiation service.

5 Conclusions

Despite all the problems mentioned in this paper, we believe that CORBAsec has a lot of potential, especially when looked at its features in a more realistic way. It is important to understand that CORBAsec is only a (powerful) security toolbox and not the solution to all security problems. In order to move the specification into the right direction, the various inherent problems of its architecture have to be further analyzed.

We recommend the following approach to securing CORBA systems with the current version of CORBAsec:

- Take into account the security of the entire system, not just the CORBAsec components. It is always necessary to look at the system as a whole and at the interplay of its various components.
- Detect and solve weaknesses of CORBAsec, e.g. the management of users or domains.

[4] Please contact the authors if you are interested in the MICOsec distribution.

– Develop creative solutions when needed, such as making the firewall ORB-friendly when it isn't.
– Ignore absurd issues in the specification, such as the predefinition of the TCP ports for IIOP/SSLIOP[5].

Competing technologies like DCOM/COM+, DCE, and EJB don't provide the functionality of CORBA and its independence from languages and platforms. They are more immature than CORBA, especially in the field of security. There is no alternative to CORBA for large-scale, distributed and heterogenous applications.

References

[Adi00] Adiron. Orbasec SL2 and Control. http://www.adiron.com, 2000.

[Bla00] Bob Blakley. *CORBA Security: An Introduction to Safe Computing with Objects*. Addison Wesley, 2000.

[Cla00] Joris Claessens. A Secure European System for Applications in a Multi-vendor Environment. https://www.cosic.esat.kuleuven.ac.be/sesame/, 2000.

[DST00] DSTC. Public Key Infrastructure RFP. ftp://ftp.omg.org/pub/docs/ec/99-12-03.pdf, 2000.

[Gol99] Dieter Gollmann. *Computer Security*. Wiley, 1999.

[HV99] Michi Henning and Steve Vinoski. *Advanced CORBA Programming with C++*. Addison Wesley, 1999.

[ISO97] ISO. Iso 10181-4: Information Technology - Security Frameworks for open Systems: Non-repudiation Framework, 04 1997.

[KOSP99] Fred Kuhns, Carlos O'Ryan, Douglas C. Schmidt, and Jeff Parsons. The Design and Performance of a Pluggable Protocols Framework for Object Request Broker Middleware. http://www.cs.wustl.edu/ schmidt/PfHSN.ps.gz, 1999.

[Lan99] Ulrich Lang. Distributed Access Control. http://www.cl.cam.ac.uk/ ul201/proposal.pdf, 1999.

[Lap98] Martine Lapierre. *TINA*. Prentice Hall Europe (Academic), 1998.

[OMG99] Object Management Group. CORBA/IIOP 2.3.1 specification. http://sisyphus.omg.org/technology/documents/corba2formal.htm, 1999.

[OMG00a] Object Management Group. Website. http://www.omg.org/, 2000.

[OMG00b] OMG. Common Secure Interoperability V2 RFP. http://www.omg.org/ /techprocess/meetings/schedule/Common_Secure_Interop._V2_RFP.html, 2000.

[OMG00c] OMG. Corba Security Service Specification v1.7 (Draft). ftp://ftp.omg.org/pub/docs/security/99-12-02.pdf, 2000.

[OMG00d] OMG. Joint Revised Submission CORBA/Firewall Security. ftp://ftp.omg.org/pub/docs/orbos/98-05-04.pdf, 2000.

[OMG00e] OMG. Portable Interceptors RFP. http://www.omg.org/techprocess/meetings/schedule/Portable_Interceptors_RFP.html, 2000.

[OMG00f] OMG. Security Domain Membership RFP. http://www.omg.org/ /techprocess/meetings/schedule/Security_Domain_Membership_RFP.html, 2000.

[Ope00] Opengroup. DCE Portal. http://www.opennc.org/dce/, 2000.

[PR00] Arno Puder and Kay Römer. *MICO: An Open Source CORBA Implementation*. Morgan Kaufmann Publishers, 2000.

[5] As they are assigned to privileged ports, a talented attacker could easily gain root access.

[Pud00] Arno Puder. Mico for the Palmpilot. http://www.mico.org/pilot/index.html, 2000.

[Sch00a] Rudolf Schreiner. Open Source Software Security. http://www.technosec.com/ /whitepapers/open_source/open_source_security.html, 2000.

[Sch00b] Rudolf Schreiner. Sicherheitsbedürfnis. *iX - Magazin für professionelle Informationstechnik*, page 14, June 2000.

[VBKM00] John Viega, J.T. Bloch, Tadayoshi Kohno, and Gary McGraw. ITS4 : A Static Vulnerability Scanner for C and C++ Code. *ftp://ftp.rstcorp.com/pub/papers/its4.pdf*, 2000.