



Universidade do Porto
Faculdade de Engenharia
FEUP



João Manuel de Vila Fernandes Orvalho

Security and Authorization Mechanisms in a Distributed Digital Asset Management System

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Departamento de Engenharia Electrotécnica e de Computadores

Security and Authorization Mechanisms in a Distributed Digital Asset Management System

João Manuel de Vila Fernandes Orvalho

Dissertação submetida para satisfação parcial dos
requisitos do grau de mestre
em
Redes e Serviços de Comunicação

Dissertação realizada sob a supervisão do
Professor Doutor José António Ruela Simões Fernandes,
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

Porto, Junho de 2003

UNIVERSIDADE DO PORTO
Faculdade de Engenharia
BIBLIOTECA M
N.º 79581
CDU 204(043)
Data 17 12 / 20 03

Security and Authorization Mechanisms in a Distributed Digital Asset Management System

João Manuel de Vila Fernandes Orvalho

Dedication

To my father Manuel, my mother Luisa, my sister Ana and my wife Verito.

Abstract

The television world has been in recent years evolving at a considerable rate caused by the increasing synergies between the computing and broadcasting technologies, which presents new requirements to the broadcasters.

Instituto de Engenharia de Sistemas e Computadores do Porto (INESC Porto), as part of its research and development activity in the digital television integrated system domain, has jointly worked with other companies on presenting new solutions, mostly based on information technology, to respond to these necessities.

First the ATLANTIC project, funded by the European Commission under the ACTS framework, defined, developed and integrated the essential building blocks of an architecture based on standard IT equipment and protocols of a digital TV post-production studio

This work was further improved in the ORBIT project, a joint effort between INESC Porto and British Broadcasting Corporation Research and Development Department (BBC R&D), which developed a distributed digital asset management architecture, called Middleware for the Broadcast Environment (MBE).

This dissertation will propose the integration of Security and Authorization Mechanisms in the MBE architecture, and delineate an implementation strategy, considered essential to the enhancement and valorization of this new architecture.

We present the design of a Role Based Access Control (RBAC) model using the access control mechanisms provided by the CORBA Common Secure Interoperability version 2 (CSiV2) architecture. The proposed system can be perceived as a CORBA based application authorization service.

Resumo

O mundo da televisão tem evoluído nos últimos anos a uma taxa considerável causada pelo crescimento das sinergias entre as tecnologias de computadores e difusão, apresentando novos requisitos aos operadores de televisão.

O Instituto de Engenharia de Sistemas e Computadores do Porto (INESC Porto), como parte da sua actividade de pesquisa e desenvolvimento no domínio de sistemas de televisão digital integrados, trabalhou conjuntamente com outras empresas para apresentar novas soluções, maioritariamente baseadas em tecnologias de informação, para responder a essas necessidades.

Primeiro o projecto ATLANTIC, financiada pelo Comissão europeia através do programa ACTS, definiu, desenvolveu e integrou os componentes essenciais de uma arquitectura de um estúdio pós-produção de TV digital baseada em equipamento IT e protocolos standard/ normalizados.

Este trabalho foi melhorado ainda mais no projecto ORBIT, um esforço conjunto entre o INESC Porto e a British Broadcasting Corporation Research and Development Department (BBC R&D), que desenvolveu uma arquitectura distribuída de gestão de material audiovisual digital, chamada Middleware for the Broadcast Environment (MBE).

Esta dissertação irá propor a integração de mecanismos de Autenticação e Segurança na arquitectura MBE, e delinear um estratégia de implementação, considerada essencial para a valorização e melhoria desta nova arquitectura.

Apresentamos o projecto de um modelo de Controlo de Acesso Baseado em Papéis (Role Based Access Control - RBAC) que utiliza os mecanismos de controlo de acesso fornecidos pela arquitectura Commom Secure Interoperability versão 2 (CSIv2) do CORBA. O sistema proposto pode ser interpretado como um serviço de autorização de aplicações baseado em CORBA.

Preface

During recent years, the evolution of television technology created a new set of requirements to the broadcasters.

To stay competitive the broadcasters have to reduce costs, minimize production times, adapt its offerings to new distribution media like the Internet and high definition television and maximize the power of available technology.

These needs led to develop solutions that were based on commodity IT equipment and integrated the handling of essence and metadata in the same network, as the ones proposed by the ATLANTIC and ORBIT projects.

The integration of IT based solutions on TV studios will inevitably affect the way people work and impact the whole broadcasting industry. For these solutions to be completely embraced and accepted, it is necessary to be responsive to the needs of the broadcasters and provide them functionalities that ease the transition and support familiar concepts.

Issues of access control and accountability arise when a system is distributed, because the professionals still want to maintain some of the “old feeling” of being in charge of their working environment.

This dissertation contributes to this effort by proposing the improvement of MBE (Middleware for the Broadcast Environment) distributed digital asset management architecture with security and authorization mechanisms, which assure that the resources are used only by authorized personnel.

Organization

The first chapter introduces the reader to the objectives of the work and is followed by an overview of the MBE Architecture.

The third chapter explains the main security concepts and terms used throughout the dissertation. The fourth deals with to security in the context of our system: it contains a threat analysis of the architecture and a selection of security components to be used.

The fifth and the sixth chapters discuss in detail the analysis, design processes and the implementation strategy of our work.

Finally, in the seventh chapter, we conclude by going over the main points and discussing the achieved results, as well as outlining what should still be done in the problem area to continue improving the system.

Acknowledgements

First I would like to thank INESC Porto for giving me the opportunity and the privilege of collaborating on very interesting projects, and giving me all the conditions to develop my work

I am especially grateful to Prof. José Ruela for all the useful advices and insightful comments concerning my dissertation, which made it much more comprehensible and valuable.

I also would like to express my gratitude to all the members of INESC Porto's Telecommunications and Multimedia Unit, and made coming to work such a delight for a period of more than three years. A special thanks goes to Pedro Cardoso, for pushing me to always do better and leading our group with unlimited strength, and to Pedro Ferreira, Ernesto Santos and Vítor Teixeira. I hope everything goes well in this new stage of their lives as founders of a new start-up company. I would also like to mention Jaime Cardoso, Vítor Rodrigues (thanks for your help with understanding MBE), Luís Barral, Orlando Ribeiro and Mário Cordeiro that besides being excellent engineers are even better friends.

I would also like to thank my family, for supporting me in every step of my life, and Verito, for always give me strength to continue and making me believe in a better future.

This thesis was funded by a grant of Fundação para a Ciência e a Tecnologia (FCT).

Reader Notes

The font used throughout the text is Times New Roman. Computer Programme code uses the Courier New font.

In the beginning of each chapter there is a small summary of the contents.

As usual in a technical text, several acronyms are used. To improve the reading process and in order not to clutter the text with definitions, a list of acronyms is provided in the beginning of the text, to which the reader can refer easily.

Bibliographic references are listed at the end of the document. Whenever referred to on the text, a number between square brackets identifies them.

Table of Contents

DEDICATION.....	III
ABSTRACT.....	IV
RESUMO.....	V
PREFACE.....	VI
ORGANIZATION.....	VI
ACKNOWLEDGEMENTS.....	VII
READER NOTES.....	VII
TABLE OF CONTENTS.....	IX
LIST OF FIGURES.....	XI
LIST OF TABLES.....	XII
ACRONYMS.....	XIII
CHAPTER 1 - INTRODUCTION.....	1
1.1 - OBJECTIVE OF THE WORK.....	2
1.2 - SUMMARY OF CONTENTS AND RESULTS.....	4
1.3 - LIMITATIONS.....	5
CHAPTER 2 - MBE ARCHITECTURE.....	6
2.1 - BACKGROUND.....	7
2.1.1 - ATLANTIC.....	7
2.1.2 - VIDION.....	10
2.2 - ORBIT OVERVIEW.....	13
2.3 - MBE IMPLEMENTATION DETAILS.....	17
2.3.1 - DIMICC.....	21
2.4 - SECURITY REQUIREMENTS.....	25
CHAPTER 3 - SECURITY BASICS.....	29
3.1 - CONCEPTS.....	30
3.2 - DISTRIBUTED SECURITY.....	35
3.3 - AUTHORIZATION MODELS.....	37
3.3.1 - Design Approaches.....	39
3.3.2 - Access Control Models.....	41
CHAPTER 4 - SECURITY IN MBE.....	47
4.1 - THREAT ANALYSIS.....	48
4.2 - SECURITY COMPONENTS AND TECHNOLOGIES.....	53
4.2.1 - CORBA.....	53
4.2.2 - CORBA Security Service.....	56
4.2.3 - Common Secure Interoperability version 2.....	60
4.2.4 - Authorization Token Layer Authentication Layer.....	63
4.2.5 - CORBA Component Model.....	64
4.2.6 - Kerberos.....	65
4.2.7 - Secure Socket Layer / Transport Layer Security.....	68
4.3 - SELECTION CRITERIA.....	70
CHAPTER 5 - ANALYSIS AND DESIGN.....	72
5.1 - BASIC DESIGN.....	73
5.2 - RBAC BASED IN CORBA SECURITY.....	78
5.3 - RBAC BASED IN CSIV2.....	82

5.4 - ROLE DEFINITION PROCESS	89
5.5 - CONCLUDING REMARKS	94
CHAPTER 6 – IMPLEMENTATION STRATEGY.....	95
6.1 - USED TOOLS.....	96
6.1.1 - ACE	96
6.1.2 - TAO	98
6.1.3 - ORBacus.....	100
6.1.4 - OpennCCM.....	101
6.1.5 - OpenSSL.....	102
6.1.6 - ORBASec SL3.....	103
6.2 - IMPLEMENTATION APPROACHES	105
6.2.1 – <i>Securing Essence Handling Connections</i>	110
CHAPTER 7 – CONCLUSIONS	113
7.1 - CONCLUSIONS	114
7.2 - FUTURE WORK	116
BIBLIOGRAPHY	117
BOOKS.....	117
PAPERS.....	118
PAPERS.....	118
STANDARDS	120
WEB PAGES.....	121
MISCELLANEOUS.....	122
INDEX.....	123

List of Figures

Figure 2-1: The ATLANTIC Studio Reference Model (source [12]).....	8
Figure 2-2: VIDION Architecture for Digital Studio Archive (source [13]).....	11
Figure 2-3: MBE Architectural Layers	16
Figure 2-4: Component Installation (source [66])	19
Figure 2-5: Handling XML Data in MBE.....	20
Figure 2-6: Production Workflow on a Typical TV Studio (source [15])	26
Figure 2-7: ORBIT Architecture (source [71]).....	27
Figure 3-1: Conventional Access Control, with Reference Monitor (source [18]).	37
Figure 3-2: Core RBAC Model.....	44
Figure 4-1: AND-decomposition	49
Figure 4-2: OR-decomposition	49
Figure 4-3: A Request Passing from a Client to a CORBA Object Implementation...	54
Figure 4-4: Enforcement of Policies in CORBA Security (source [26])	58
Figure 4-5: CSIV2 Architecture	61
Figure 4-6: CORBA Component (source [65])	65
Figure 5-1: Interactions between Client, Object and RAD Service (source [29])	76
Figure 5-2: Interactions between Objects and Services	87
Figure 5-3: Content Creation Process (source [43])	89
Figure 5-4: Content Creation Process in MBE (source [66])	91
Figure 6-1: ACE Class Hierarchy (source: [60])	97
Figure 6-2: TAO Block Diagram (source: [61]).....	99
Figure 6-3: Essence Transfer Plane Protocol Stack (source [65]).....	111

List of Tables

Table 5-1: Required Rights Matrix 79

Table 5-2: Granted Rights Matrix 79

Table 5-3: Elements of a Role Authorization Statement 84

Table 5-4: RAD Service's Permission Matrix..... 85

Table 6-1: Example of the RA Table that Associates Roles with Users 108

Acronyms

AC	Access Control
ACE	Adaptive Communication Environment
ACTS	Advanced Communications Technologies and Services
API	Application Programmer Interface
ATLANTIC	Advanced Television at Low Bitrates and Networked Transmission over Integrated Communication systems
ATLAS	Authorization Token Layer Acquisition Service
ATM	Asynchronous Transfer Mode
BBC	British Broadcasting Corporation
CA	Certificate Authority
CAL	Client Authentication Layer
CCM	CORBA Component Model
CORBA	Common Object Request Broker Architecture
CS	CORBA Security
CSlv2	Common Secure Interoperability version 2
DCOM	Distributed Component Object Model
DETAIL	DIMICC Essence Transfer Already Implemented Library
DIMICC	Distributed Middleware for Multimedia Command and Control
DLL	Dynamically Linked Library
DoD	Department of Defense (USA)
DOM	Document Object Model
DSD	Dynamic Separation of Duty
EJB	Enterprise Java Beans

GIOP	Generic Inter-ORB Protocol
GoP	Group of Pictures
GRM	Granted Rights Matrix
GUI	Graphical User Interface
IAL	Identity Assertion Layer
IDL	CORBA Interface Definition Language
IETF	Internet Engineering Task Force
IFR	Interface Repository
IIOP	Internet Inter-ORB Protocol
INESC	Instituto de Engenharia de Sistemas e Computadores
IP	Internet Protocol
IT	Information Technology
KDC	Key Distribution Server
MBE	Middleware for the Broadcast Environment
MPEG	Moving Picture Experts Group
MXF	Material eXchange Format
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
ORB	Object Request Broker
ORBIT	Object Reconfigurable Broadcast using IT / Infrastructure Trial
PC	Personal Computer
PES	Packetized Elementary Streams
PM	Permission Matrix
RAD	Resource Access Decision
RAID	Redundant Array of Inexpensive Disks

RBAC	Role Based Access Control
RMI	Remote Method Invocation
RRM	Required Rights Matrix
RTP	Ráadio Televisão Portuguesa
SAS	Secure Attribute Service
SDK	Software Development Kit
SECIOP	Secure Inter-ORB Protocol
SMEF	Standard Media Exchange Framework
SQL	Structured Query Language
SSD	Static Separation of Duty
SSL	Secure Socket Layer
SSLIOP	Secure Socket Layer Inter-ORB Protocol
TAO	The ACE ORB
TCP	Transport Control Protocol
TCSEC	Trusted Computer Security Evaluation Criteria
TLS	Transport Layer Security
UMID	Unique Material Identifier
VIDION	Video DIgital ON Line
VTR	Video Tape Recorder
XML	eXtensible Markup Language

RBAC	Role Based Access Control
RMI	Remote Method Invocation
RRM	Required Rights Matrix
RTP	Rádio Televisão Portuguesa
SAS	Secure Attribute Service
SDK	Software Development Kit
SECIOP	Secure Inter-ORB Protocol
SMEF	Standard Media Exchange Framework
SQL	Structured Query Language
SSD	Static Separation of Duty
SSL	Secure Socket Layer
SSLIOP	Secure Socket Layer Inter-ORB Protocol
TAO	The ACE ORB
TCP	Transport Control Protocol
TCSEC	Trusted Computer Security Evaluation Criteria
TLS	Transport Layer Security
UMID	Unique Material Identifier
VIDION	Video Digital ON Line
VTR	Video Tape Recorder
XML	eXtensible Markup Language

Chapter 1 - Introduction

This chapter introduces the main objectives of the work described in the thesis.

It presents a summarized version of the contents of the thesis, as well as of the results obtained.

Finally, a brief analysis of the limitations of the work is made.

1.1 - Objective of the Work

The work described in this thesis is part of the research and development activity that is being carried out for several years in the digital television integrated system domain at Instituto de Engenharia de Sistemas e Computadores do Porto (INESC Porto).

It is based on an existing and functioning system developed in the ORBIT (Object Reconfigurable Broadcast using IT) project. This project, a contract between INESC Porto and BBC Research and Development, was developed as a pilot demonstrator of the deployment and network integration of low cost, mass produced and open IT equipment in a distributed digital television environment.

The architecture of the system developed in the ORBIT project is called Middleware for the Broadcast Environment (MBE). It is a middleware object-oriented solution aimed at broadcasting environments, to manage multimedia assets, such as video and audio (often called essence), and metadata (data describing features on those assets).

The MBE framework allows the wide scale deployment of objects in diverse network environments, through extensive use of CORBA, XML and Java technologies. During the implementation phase proven technology and standards were adopted, whenever they existed.

This project provides a working system, which is able to show to the broadcasters, the full potential of IT and its associated benefits: significant cost reduction in production infrastructure and equipment, integrated access and management of essence and metadata, better scalability and system reconfiguration ability and compatibility with new distribution methods.

In this thesis we will describe the process of adding security-related functionality to an already existing architecture (MBE), considered essential to its enhancement and valorization.

It is our aim to reuse this working and installed base, and introducing significant changes to the architecture or the implementation falls outside the scope of this work.

Specifically, we will propose Authorization and Security mechanisms to be integrated to the MBE distributed digital asset management platform, and delineate an implementation strategy.

1.2 - Summary of Contents and Results

To protect distributed applications and their resources it is necessary to use the security provided by middleware access control (AC) mechanisms.

After careful analysis and balancing of the available technologies and techniques and our limitations and constraints, we have chosen to design a Role Based Access Control (RBAC) system in the MBE architecture.

This AC system is based on CORBA Common Secure Interoperability version 2 (CSIv2) architecture and uses a Resource Access Decision (RAD) service, to centralize access control enforcement and decoupling of authorization and application logic.

Authorization, or AC, depends on authentication. To solve the problem of authentication of users to the system we chose a design around the Kerberos distributed protocol.

On a lower level, we have decided to protect and secure the essence handling connections provided by the DIMMIC module (responsible for the media handling in the MBE architecture), with SSL protocol.

The proposed system will also have many benefic side-effects on the overall architecture that can be explored at a later stage. Mainly, it eases and simplifies the administration management of the whole MBE platform.

1.3 - Limitations

There are some limitations associated with this work, and that should be noted upfront.

In our case, security goal “Design for security” [1]; this is the notion that security should be considered during all phases of the development cycle and greatly influence a system's security.

In MBE, Security was not designed into the system from the beginning, which made our task considerably more difficult. As Bruce Schneier says [2], “Security is a process, not a product”: it is not recommended to add security to the system after the fact and expect it to work. It is vital to understand the real threats to the system, design a security policy countermeasure with those threats, and build in appropriate security countermeasures from the beginning. But this is a slightly pessimistic view.

Security engineering is also a discipline focused on adapting existing systems as their environment evolves [3]. This is our selected approach; we have a system that exists and is already in (limited) use, and we have to adapt it to remain dependable in the face of malice, error or mischance.

Another severe limitation was the availability of secure software/products for the existing platform. As we will see later, the choice made was conditioned by the discontinuation of the development and support of a package that provided an implementation of CORBA Security Service for the ORB that is on the basis of the MBE architecture. This had a great influence on the design stage of the work, because it changed some of the initial conditions and assumptions, causing a deep alteration in the design and implementation strategy.

We decided, with some pragmatism, to reuse part of the investigated concepts and adopt them to a new environment. The solution presented allows us to demonstrate the validity of the architectural concepts, although with some limitation because it is based on a non standard technology.

Chapter 2 - MBE Architecture

This chapter will present an overview of the MBE Architecture developed at INESC Porto.

The first section presents some background information and projects that led to the development of the ORBIT project.

Then we will give a technical description of the MBE architecture and some important implementation details.

Finally we will identify the need to introduce security mechanisms in this architecture.

2.1 - Background

This work is part of the research and development that is being carried out for several years in the digital television integrated system domain at INESC Porto.

To completely understand how the ORBIT project came to be, we first need to present the research projects that preceded it.

We will focus our presentation on the ATLANTIC and VIDION projects.

2.1.1 - ATLANTIC

The ATLANTIC (Advanced Television at Low Bitrates and Networked Transmission over Integrated Communication systems) project started in March 1995, with the objective of developing technology for the implementation of a TV production studio totally based on MPEG-2 [10]. This project had a duration of approximately three years and was partially funded by the European Community, under the ACTS framework.

ATLANTIC was run by a large consortium integrating the following European companies: BBC R&D, CSELT, EPFL, ENST, FhG, INESC and Snell & Wilcox.

An important factor in containing the costs and maintaining the technical quality TV production is the ability to handle and process the video and audio signals as much as possible in compressed form. But the technical quality of the television sound and picture signals that would normally result from the use of bit-rate reduction, and in particular from decoding to PCM form and recoding, was very bad.

To overcome this problem, one of ATLANTIC's main goals was to develop and demonstrate the technology, which enabled the signal to be kept in compressed MPEG2 format throughout the complete broadcast chain. This was achieved by avoiding decoding/recoding operations wherever possible and by reusing the original coding decisions: a technique known as MOLETM was developed by one of the ATLANTIC partners for this purpose [11]. The result was the achievement of cascading with zero loss between compressed and uncompressed signals.

The project developed a prototype news studio composed of various pieces of equipment interconnected via an ATM network. This was done in order to accomplish another objective of ATLANTIC: to prove the possibility of using standard IT equipment and protocols as building blocks of a digital TV post-production studio that performs tasks usually done by high cost, proprietary and specific equipment.

ATM was chosen as the network technology for the ATLANTIC studio for its unique characteristics - flexibility, scalability, provision of bandwidth on demand and the ability to support a wide range of quality of service requirements. At that stage switched LAN technologies were not yet mature and widely available.

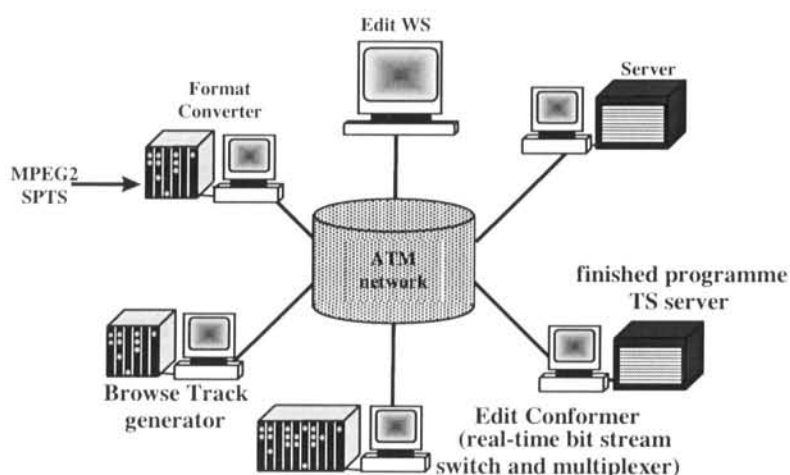


Figure 2-1: The ATLANTIC Studio Reference Model (source [12])

The various components of the ATLANTIC post-production facility are shown in the reference model on Figure 2-1.

The Format Converter will serve as the gateway of the ATLANTIC studio, since it interfaces the ATLANTIC studio to the outside world, receiving all incoming audio-visual material in the format of MPEG2 Single Program Transport Streams. The role of the Format Converter is to recover the elementary components of incoming MPEG2 TV programmes and to store them as separate streams of PES (Packetized Elementary Streams) packets in the Main Server.

The Server stores all essence files along with index files that facilitate the conversion of timestamps into byte offsets, thus making it possible the time-based random access into the streams.

The ATLANTIC post production suite will allow the user to edit video sequences with an accuracy of one frame. Thus, instead of working with the full-quality MPEG2 video bit streams, the editing workstation will generate the Edit Decision List (EDL), and will be able to preview the final result, by using the browse versions of the selected material. Then the Browse Track Generator will access MPEG2 source material stored on the Main Server and will generate a second bit stream with only I-frames (browse version) to be used in rough-editing operations.

The Journalist Editing Workstation (Edit WS) provides editing facilities for post production. It consists of a PC with a graphical user interface (GUI) and MPEG2 commercial decoder boards.

The Edit WS lets users produce their own programmes by creating an EDL (a list of the decisions which describe a series of edits); then EDL's will be passed to the Edit Conformer, which is responsible for using the EDL and the MPEG-2 streams in order to generate the final programme and store it as a MPEG Transport.

2.1.2 - VIDION

VIDION (Digital Video On-Line) a project funded by a Portuguese RD programme involving RTP (the Portuguese public TV broadcaster) and Europarque (responsible for a science centre park).

The main goal of VIDION was to propose a strategy for the evolution towards the digital domain of the complete RTP archive [13]. This archive presented many problems due to the different support formats (various analogue and digital formats) amounting to more than 300 000 hours of recorded material, the requirements put by the amount of information to store and transmit, the complexity of the database and the whole architecture involved.

To fulfill this goal and prove the effectiveness of the proposed solution, VIDION developed a small-scale archive to be used by the news service of RTP.

This system was composed of several elements: two different kinds of servers (broadcaster and browse quality servers), workstations for searching, selecting and previewing video sequences, digitising stations and the required communication infrastructure. In addition it was also developed a text-based intelligent search engine, which is able to assist the indexing and search processes.

The system developed within VIDION is composed of a number of elements as shown in Figure 2-2.

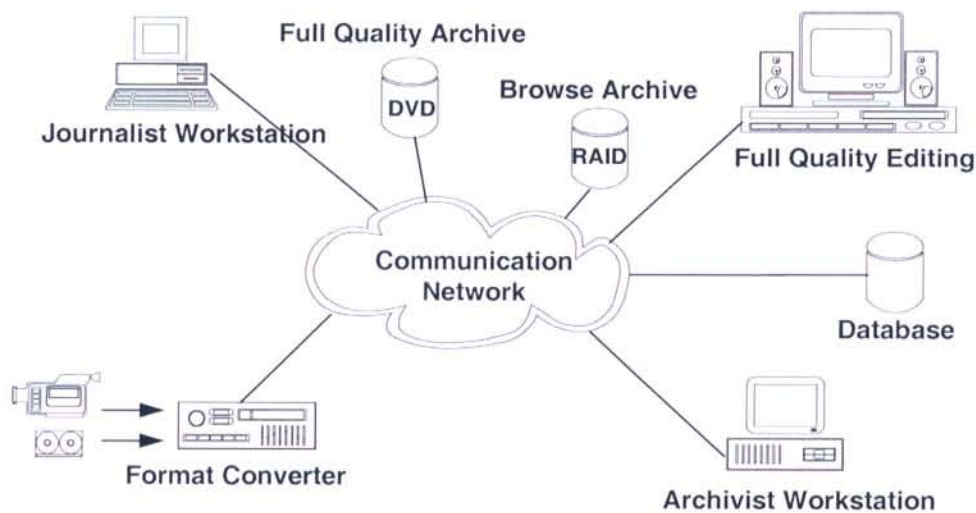


Figure 2-2: VIDION Architecture for Digital Studio Archive (source [13])

The first step on building a digital archive is the creation of two digital versions of the same source but with different size and quality: browse and full quality (broadcaster). The browse version is available for searching, previewing, selecting and pre-editing in the journalist workstation. The full quality archive provides the information in broadcasting quality to be used in the final edition of the programme.

By supporting different quality levels it allows the reduction of storage, transmission and processing costs while still maintaining references to each frame and its original time code.

One of the main concerns of the project was on choosing standard video formats (MPEG1 at around 1.5Mbps for the browse quality video, and MPEG2 for the full quality server), long last storage devices (such as RAID storage) and on finding solutions for easy and efficient access to the information (system architecture builds on generic and open protocols and interfaces). These three aspects will allow the real preservation of this valuable archive.

The system is now available through 4VDO [58], a company that spun-off from INESC Porto to commercialize and assist in the deployment of this media asset management system.

VIDION provides a tool for the efficient management of video archives by associating essence with metadata and making both available through a network, integrating the process of media ingest, annotation, archiving, searching and editing.

2.2 - ORBIT Overview

Following the success of ATLANTIC, BBC R&D and INESC Porto decided to launch the Object Reconfigurable Broadcasting using IT (ORBIT) project to develop a trial implementation of distributed networked system based on MPEG-2 and IT equipment.

This was necessary in order to test and prove the reliability and suitability of the solutions developed in the ATLANTIC project, before they could be widely deployed in real broadcasting environment [14]. The pilot also provided the means for the operational staff to use the system to gain experience and to feedback their comments and criticisms.

The main objectives of ORBIT were to identify best practices in several areas [15]:

- Migration of content handling away from specialist broadcast equipment towards economical mass-market IT hardware.
- Integration of media asset management and content handling tools.
- Suitability of architecture for use on a corporate-wide scale.
- Ready access to content at the desktop.
- Automation of the essence and metadata handling process.
- Reconfigurability for different content formats, including any necessary conversion.
- Reconfigurability to cope with various production processes and programme genres.
- Pursue standardization of the techniques developed in ATLANTIC.
- Demonstrate the advantages of using middleware in order to maximize flexibility and scalability of television facilities.
- Ability to adopt new technologies and standards quickly.

These goals can be summarized in one sentence: re-engineering of production processes to capture and retain the metadata, keeping it linked with the essence

through the whole operational chain, and having them both available anytime and anywhere they are requested [69].

Broadcasters have always had metadata but, in many cases, that information is hard to access and often lost.

As the amount of network traffic that is generated by metadata is relatively small when compared with essence, it is easy to achieve distributed access to metadata across the whole organization, using IT infrastructure (generic networks and hardware), databases and several standards: SQL (searching), SMEF (defining data models), XML (to define and move around metadata), UMID (linking metadata and essence), MPEG-7 (multimedia concept description interface) and MPEG-21 (multimedia framework concepts).

But this is clearly not enough. We have to also handle essence in the same network, to allow us to keep an exact map with the corresponding metadata [70].

There were several proprietary solutions to handle the essence, but they were expensive and did not merge well with the IT based metadata network. And after all, we must always have present that the savings presented by a news solution must be greater than the costs of deploying it.

ORBIT puts the essence and the metadata in one IT network, by having:

- Efficient essence compression: different qualities of compressed signal are used. ORBIT chose MPEG-2 Long GoP for production quality signals, giving a three to one improvement in storage efficiency, and MPEG-1 (I frame only) for a range of browse quality versions adapted to different working methods.
- Essence storage and transport: besides the use of compressed signals and taking advantage of the continuous performance boost of computer hardware, ORBIT developed a set of network and resource management utilities and load sharing techniques.

But, the ORBIT project aimed at pushing further the ATLANTIC expectations implementing a generalized middleware layer between the broadcast equipment and the operational workstations.

The middleware solutions available at the time had several intrinsic limitations that prevented them from being used in the project: they were designed for relatively small volumes of data and were not suitable for video handling, which has massive requirements.

Therefore, INESC Porto and BBC R&D developed the Middleware for Broadcast Environment (MBE) architecture: a platform based on CORBA and XML that provides a unified middleware solution to essence and metadata handling in television environments.

CORBA provides a suitable middleware foundation that is vendor, platform and language independent and is a standardized and open technology (CORBA will be further explained in a later chapter).

XML provides the information (metadata) support and allows the data and its schema (structure definition) to be carried together; it is a standardized and fast evolving technology, which is used and understood throughout the industry (having several tools available to use) making it easy to incorporate work coming out from current metadata related efforts. It provides data model independence and allows evolution in accordance with changes in organization.

We can identify two essential areas in this platform:

- Essence handling: it is provided by the DIMICC (Distributed Middleware for Multimedia Command and Control) layer. This is implemented as a separate module that manages the hardware devices and the software services, instantiation and distribution of services, the essence (production and “browse” versions) transfers, location of essence in the network and the setup and break down of connections as required.
- Business rules definition: it brings together the essence and the devices, using DIMICC, and the descriptive metadata, using XML. Business rules define the process rules for a given utilization scenario (i.e., the who, the where and the when in an organization), and is provided by the Business Objects layer.

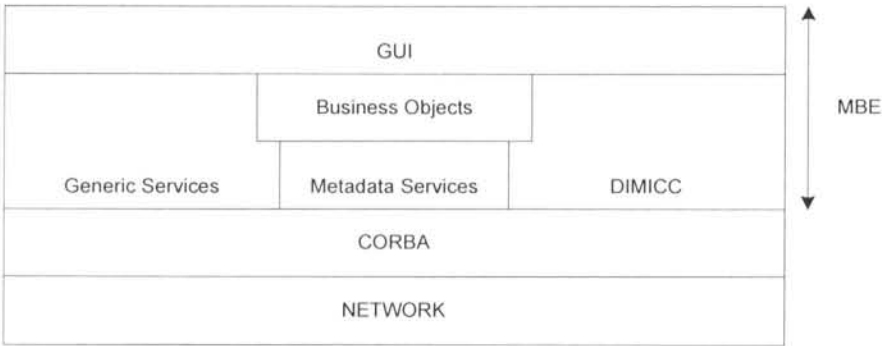


Figure 2-3: MBE Architectural Layers

Figure 2-3 shows the MBE layered architecture. It provides a look at the system from the client perspective hiding the underlying technologies, storing the data transparently and giving an abstract API that describes the behaviour that the system is able to assume.

It is important to notice that the interfaces between the different layers use existing standards where possible, to provide easier integration with third party systems.

As a conclusion, the ORBIT project developed distributed digital asset management architecture we call MBE. It is an efficient, scalable and affordable broadcast infrastructure, which is based on commodity IT equipment to handle essence and metadata in the same network.

It uses standard, best of the breed technology whenever possible: XML, CORBA, SQL, MPEG-7 and MPEG-21, to provide a rich and open API for application developers.

2.3 - MBE Implementation Details

After introducing ORBIT, we will give some important implementation details of the MBE architecture, and of its essence handling module, DIMICC.

MBE architecture is a framework for the development of re-configurable objects that may be used to provide solutions for Broadcasting. The definition of a core set of software components with carefully designed interfaces and clear definition of functionality, results in a pool of tools that may be deployed in different configurations providing solutions for different situations [65].

As Figure 2-3 showed the MBE architecture uses a layered approach.

Business Rules, data sets and graphic interface to users are frequently packaged together in very complex software GUIs, which brings problems of development, maintainability and repetition of code.

This is where the Middleware approach comes in. By deploying a Business Object Layer that may be shared by several other software clients, one is able to gather the rules of a system in a way that is easier to maintain, control and reuse.

So, we now have GUIs presenting the data according to the rules defined in the Business Object layer. Another problem arises: how to handle the data in a flexible way?

The solution is to provide a lower layer of software that is able to handle data for developers in the most appropriate way.

If these low level components are generic, they need not depend on any data model, especially environments where data models are bound to evolve over time and vary from organization to organization.

Also it must be easy to update the rules of the system and that means not only data but also behaviour. This includes for instance connection to third party software components.

The Services (metadata related and others) are all software modules that have no knowledge of business or organization rules and are therefore generic and reusable in different scenarios [66].

There are also external services used to integrate all the modules in the system, like the OMG defined Naming Service (it allows modules to register themselves by name), Trading Service (it allows modules to register themselves by service type) and Notification Service (it provides the Notification Channels).

In MBE, whenever possible, existent standard solutions are used to eliminate vendor dependency, and facilitate integration with third party systems. The interfaces between the Services and Business Objects and GUIs and even between services use existent standards, as well.

The different modules from every layer (except DIMICC) were implemented using the following technologies:

- Java for implementing the modules and objects.
- CORBA Component Model (CCM) to control object configuration and deployment.
- XML for data handling.

Java [46] was chosen as the implementation language. It brought several advantages: portability, good object-oriented programming support, large collection of libraries (as JDBC to handle database connectivity and Swing for GUI implementation) and simple memory management (due to Garbage Collection and absence of pointers). In fact, Java was responsible for the quickness of implementation of the MBE system, especially of the GUI components.

Early in the development phase it became apparent that some way to automatically configure and deploy the software modules was needed. This is particularly important if we consider the size of an installation of the MBE system at a broadcaster.

CCM provides the answer to this. This specification goes beyond the previous CORBA specifications that stopped at the interface definition. With this new specification, standard mechanisms are now in place to describe implementation details in a standard way allowing the use of tools from different vendors to automatically generate parts of the implementations of the modules, which we will call from now on CORBA Components.

The CCM also addresses deployment and configuration of the CORBA Components. The idea is to have a component server, which is a software process that runs on a host machine, provides a component installation interface and an environment where the Components can run.

The GUIs in MBE are also CORBA Components so instead of having all the system installed from the start, one may deploy a Component repository and have the GUIs installed on the clients on demand. This process is shown on Figure 2-4.

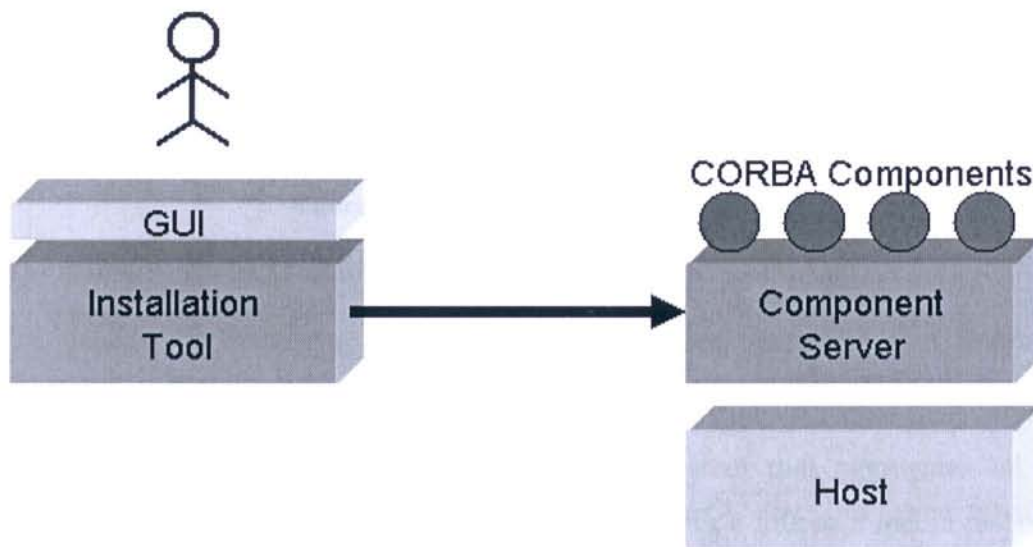


Figure 2-4: Component Installation (source [66])

Because the installation API is CORBA based, installation of software can be remote which has obvious advantages when it comes to large-scale installation. It means that software upgrades may be done remotely as well, as well.

For data handling the solution chosen was to use the XML object model, because defining data structures statically through the use of IDL interfaces would cause compile time dependencies on the clients, which is not flexible enough.

Applications use a software component called a parser that transforms the XML text data into a tree structure, usually in memory, that the application may manipulate. The API provided to manipulate that tree follows the object model defined in the DOM recommendation. XML Schema was chosen to define the structure of the data, allowing interoperability between modules.

Figure 2-5 shows how MBE applications handle XML data processing.

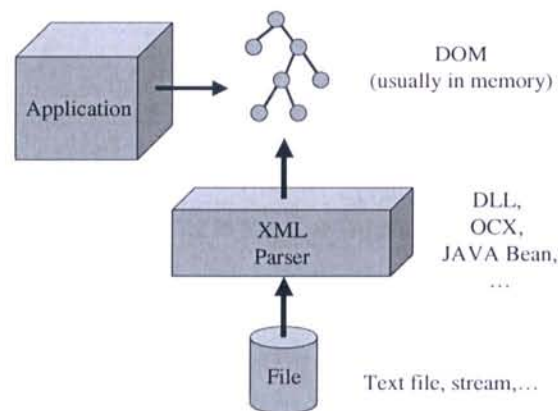


Figure 2-5: Handling XML Data in MBE

For a general understanding of the MBE architecture, there are two important concepts that are the building blocks of the system:

- Media asset: an entity (resource) on that system that aggregates all the required metadata (technical and descriptive) with a link to a media resource. These types of flexible wrapping formats are a good way to insure inter-working between applications from different manufacturers. MBE uses MPEG-21's "Digital Item Declaration" to support essence and metadata linking and transfer.
- Service: a small reusable processing module used in the media handling context. In MBE any piece of software has a number of input and output objects plus a control interface, thus we can create a processing chain by having several services connected with each other.

2.3.1 - DIMICC

The DIMICC [15] module is responsible for handling the essence in the MBE architecture.

DIMICC defines an object model and a set of protocols for the usage of and the interaction between the different components of a television production studio. Since a studio is composed of several hardware entities (e.g. computers or specialist hardware), CORBA is prescribed as the middleware to use for the communication. Thus, the objects in the object model are indeed CORBA objects [67].

The generic DIMICC entity can be defined as a CORBA object that exports a control interface and aggregates several input (or sink) and output (or source) objects; it is called a Service. The main advantage of sinks and sources being first-class CORBA objects is that it allows references to them to be passed freely, allowing sources and sinks to refer to each other in a location transparent fashion.

An essence transfer is obtained by connecting a Source to a given number of Sinks (Multicast).

DIMICC is a CORBA based scalable architecture that allows an easy development and seamless integration of the different components of a Digital Television Studio. It manages the hardware devices and the location of media in the network, and is also responsible for the transfer of the production and “browse” essence [70].

DIMICC presents a consistent view for the control of essence and devices.

The design of DIMICC fulfilled several requirements:

- It should be based on distributed object-oriented techniques, which led to the selection of CORBA, justified by our own experience and its record as an open, vendor independent, highly available technology.
- It should be as much as possible built upon existing *de jure* or industry standards.
- It should be based on a full-blown object model that treated as first-class objects both tangible things, such as devices, and intangible items, such as essence.

- This model should be completely extensible, capable of allowing the non-disruptive addition of new classes of objects, in order to support new devices or items.
- It should leverage CORBA abilities such as location transparency and encapsulation, i.e., users should only care with value provided by the objects, and never, unless explicitly required, with their physical location or the way they are implemented.
- It should promote componentization, that is, the construction of a system based on small building blocks that could easily be tested and reused without changes on the implementation.
- It should also be able to use several data transport mechanisms, in a way that such mechanisms would be transparent to DIMICC users. Besides, there should be provision for the definition of the required quality of service (QoS) on a per-connection basis and for the use of multicast.
- Achieve integration of metadata and essence, keeping the relationships and navigability between each other.
- Support for management applications should also be intrinsic in the system.
- Support should exist for both legacy and inexpensive devices that do not have networking capabilities and for legacy networks such as the Serial Digital Interface (SDI).

The DIMICC architecture can be divided into four planes, each one being responsible for dealing with separate aspects of the whole system: system plane, control plane, essence transfer plane and metadata plane.

The system plane is in charge of the system-wide services that provide the users with a unified view of the whole system.

The control plane deals with issues related to controlling devices and essence, and with the reporting of important events and alarms.

The metadata plane is accountable for the linking between metadata and essence. Since other layers and modules in MBE handle metadata, the model only includes the necessary mechanisms for navigation between the metadata and the essence worlds.

Metadata management systems use (or at least will use in the near future) UMIDs to identify pieces of essence, whereas in DIMICC object references are used for that purpose.

To perform the link between the two namespaces, there will be a system service responsible for the mapping between a UMID and a CORBA reference, the Essence Locator Service (ELS).

Whenever essence is ingested into the system, or is created within it, it will register with the ELS. Clients, upon getting a UMID from the metadata management system, will query the ELS and retrieve the object reference of the desired piece of essence. Navigability on the other direction will be provided by the ability of querying an essence item for its UMID.

The essence transfer plane is responsible for transferring essence between components and for the management of the network connections required for that task.

Due to CORBA's poor performance when used to stream large amounts of data, CORBA will not be used for essence transfer. This would make it difficult to fulfil the requirement of supporting several network technologies and protocols.

The solution to all this is to define a protocol for the negotiation and establishment of connections between Sources and Sinks, which, whenever instructed by a client to do so, will transparently set-up the data connections.

In this case, each endpoint (source or sink) of a connection has two views: a CORBA and a C++ interface.

In DIMICC, whenever a client wants to interconnect a Source and a Sink, it will invoke a method in the Source and pass it a reference to the target Sink.

The Sink will then return a list of the protocols supported and the respective service access point (SAP) addresses. Comparing with its own list, the Source will decide which protocol to use and then establish the connection, using the appropriate signalling.

The `Sink` will then return a list of the protocols supported and the respective service access point (SAP) addresses. Comparing with its own list, the `Source` will decide which protocol to use and then establish the connection, using the appropriate signalling.

Finally, the `Source` will notify the `Sink` that the establishment has succeeded (or otherwise) and pass it a connection identifier.

When a connection is to be terminated, the client will instruct the `Source` to do so, causing the latter to inform the `Sink` and to release all the resources associated with the connection.

XDIMICC [68] is an implementation of servers, clients, business objects, ActiveX controls and libraries that follow the interfaces and protocols defined by DIMICC. XDIMICC is implemented on C++ programming language, contrary to the rest of MBE.

Unlike most servers that are implemented as executable files, all XDIMICC servers are implemented as dynamically linked libraries (DLLs). The rationale for this option stems from the need to select the desired concurrency model at deployment time rather than at implementation time.

For instance, it is possible to instruct one server to run as an independent process, or to co-locate several servers in the same process.

2.4 - Security Requirements

The projects described before represent a deep change in the television world, which has been in recent years evolving at a considerable rate caused by the increasing synergies between the computing and broadcasting technologies.

This evolution will inevitably affect how people work and impact the whole broadcasting industry. Reinforcing this change is also the need to reduce costs in order to be competitive in a market populated with mergers and strategic partnerships.

To keep a leading role in the broadcasting business it is required a complete change in the minds of most people who manage and work in TV broadcasters and this will cause changes to the usual workflow of these organizations.

It is inside the television studio that we can see some of the most profound changes happening. The integration of IT based solutions will, in many cases, cause the disruption of many deep-rooted practices and processes.

For these solutions to be completely embraced and accepted on a sometimes conservative and avert to change medium, we have to be responsive to the needs of the professionals and provide them functionalities that assist the transition and support familiar concepts.

In a television environment we can identify three big functional areas:

- **Playout:** programmes from the production area are mixed with live signals and are broadcasted to over different transport networks.
- **Production:** where the acquisition of material (intake) and its subsequent edition into programmes takes place. Insertion of metadata and packaging (building the final version of the content to be distributed) are also made in this area.
- **Archive:** archives all broadcasted programmes, as well as edited programmes and extra materials that come from the production area.

All of these areas will change with the evolution that has been outlined in the previous discussion.

The diagram on Figure 2-6 shows the typical workflow during the production of a TV programme.

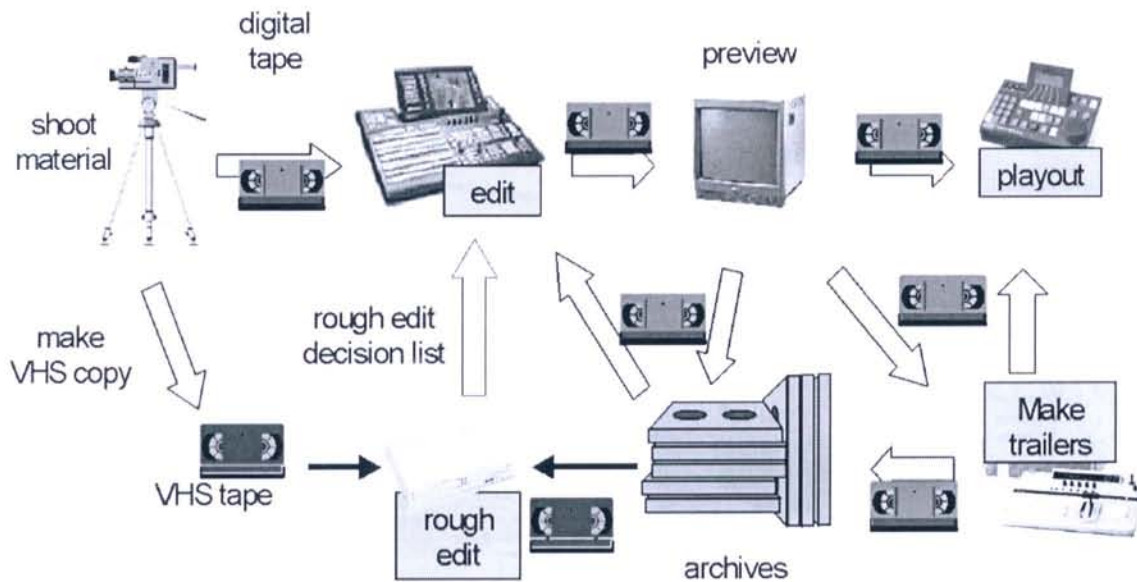


Figure 2-6: Production Workflow on a Typical TV Studio (source [15])

Along the several steps a programme has to go through, tapes are exchanged, giving the technicians the complete control of the material as well as of the dedicated TV equipments (VCR, Editor...).

In the case of ORBIT, the possible new architecture of a television studio is shown in Figure 2-7.

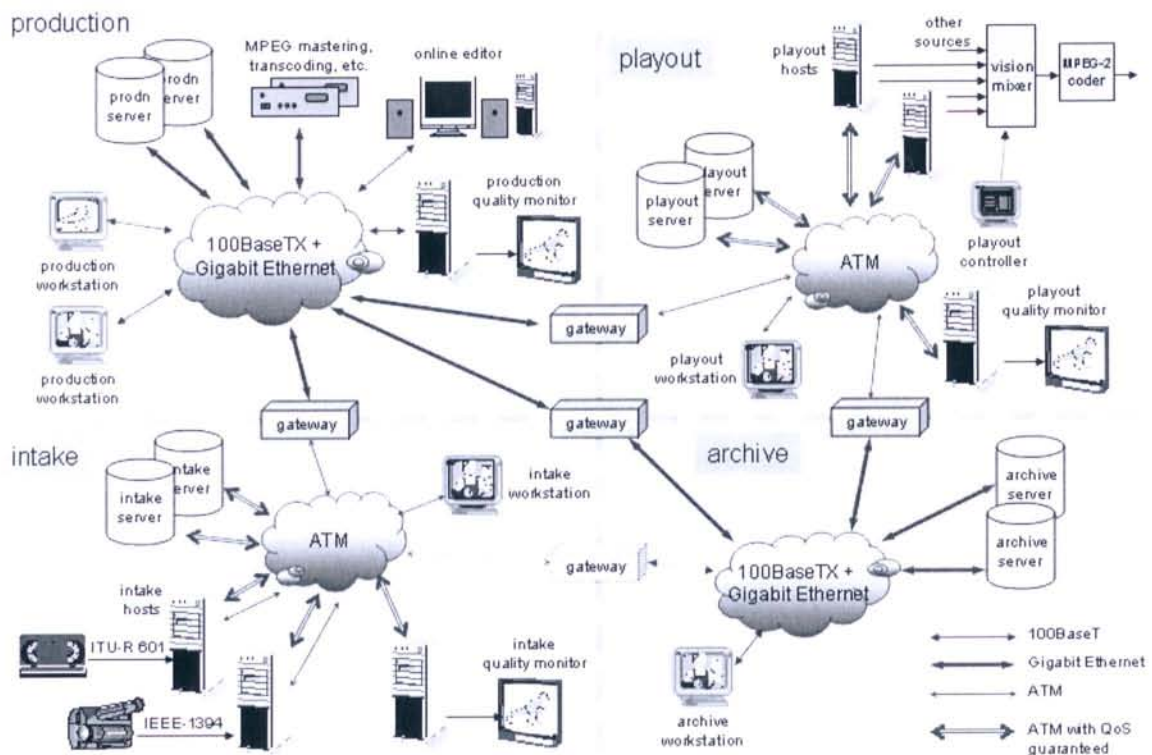


Figure 2-7: ORBIT Architecture (source [71])

In this new architecture, a network that spans the whole television environment (and sometimes other departments) connects everything giving (almost) instant access to essence and associated metadata.

This concept is completely new to most television personnel. Besides having to prove the reliability of an IT based solution, we have to give some assurances that their work will not be disrupted by overlapping needs and requests or by unauthorized users.

This calls for the integration of security and authorization mechanisms into the MBE distributed architecture. There was a general security goal that was initially defined, but never really was designed.

Our work aimed at satisfying this need and had three main objectives:

- Authentication: identify users and prevent unauthorized user from disrupting the system.

- Authorization: control the access to the system resources.
- Integrity: guarantee that the programmes and other materials are not tampered or changed without consent.

Some of these concepts will be further developed in the next chapter.

When the system is distributed, people still want to maintain some control on their working environment.

Before, control of a tape meant the control of the programme they were working on or material they wanted. With the advent of new architectures we have to introduce mechanisms and solutions that respond to the needs of the broadcasters and assure a smooth transition into a new era.

Chapter 3 - Security Basics

Up until now, we have talked about security and authentication in a few occasions, but stayed away from a clear definition of these terms.

This chapter will provide a concise characterization of key security concepts and terminology.

We begin by specifying the meaning of security and related concepts, and follow this by exploring the problem of security in distributed systems.

The problem of authorization or access control is discussed in the last section. We also specify several design approaches and access control models, especially focusing on the RBAC model.

Before continuing our analysis we should first define what we are really talking about. Most people have some kind of understanding on what security means, especially when it refers to their systems. But normally it is based on some kind of horror story involving a (sometimes teenager) hacker (more correctly, we should call it cracker [54]) or magic antidote revolving around cryptography.

But this is far from a useful definition on which we can base our discussion of security problems on a complex distributed system. This chapter attempts to provide a concise characterization of key security concepts and terminology.

The following text was largely based on some books [1] [2] [3] [4], and on two extremely useful glossaries [23] [64]. These are highly recommended sources for a deeper understanding of these issues.

Let's begin with some useful definitions.

3.1 - Concepts

What do we mean when talking about security?

Security is about the protection of assets (tangible, like network hosts, or intangible, like information). The owner associates a value with the asset and wants to restrict manipulation and access to it.

Because assets have a value it is more appropriate to quantify the loss cost associated with potential attacks. The loss cost can be defined as the negative impact (replacement cost, reputational damage ...) when an asset is compromised.

So we can say that the purpose of security is to minimize the overall loss associated with potential attacks. But total security is unattainable, because a system must react to user actions and therefore be vulnerable in some way or another. Therefore, the security enforcement is about finding an appropriate trade-off between the cost of implementing a security measure and the loss cost of a compromise. This is called risk analysis, which is responsible for determining and comparing all potential risks to security goals with the effectiveness of the corresponding countermeasures.

It should be noted, that in some situations (like CORBA for instance) security is mainly concerned with information security, because other types of security (such as

physical security), cannot be achieved through mechanisms that the technology provides. This is our case. Information Security can be defined as the task of preventing information assets from being compromised.

When securing information we must distinguish a number of different aspects. Protection of information assets from unauthorized access or interference is often defined as having the following goals (as defined in the Common Criteria standard [38]), also called the three pillars of security:

- Confidentiality: the prevention of unauthorized disclosure of information.
- Integrity: the prevention of unauthorized modification of information.
- Availability: the prevention of unauthorized withholding of information or resources.

These goals can have different priorities depending on the actual application or particular environment. We can also add some goals to this list such as accountability and dependability.

The terms security and confidentiality were, historically, closely related, and as a result of this, research in computer security was mainly concentrated on this topic. Sometimes these terms were used as synonyms, despite that other aspects are involved in securing a system. In our context, Confidentiality denotes the protection of requests and replies from unauthorized reading, as well as restriction of access to target operations, which return confidential information, so that only authorized callers can invoke them.

It is not easy to give a concise and precise definition of integrity. In a narrow sense, it deals with the prevention of unauthorized writing, and can be viewed as the dual of confidentiality. In fact, similar security mechanisms can be used to achieve both goals. In the context of communications security, integrity refers to the detection and correlation of modification, insertion, deletion, or replay (repetition) of transmitted data, including both intentional manipulations and random transmission errors. In the case of our distributed system we can also include in this definition the access to target operations that can modify protected information needs.

Availability is often defined as the property of being accessible and useable on demand by an authorized entity. Usually, an availability compromise is the prevention of authorized access to resources or delaying of time-critical operations, and is denominated by denial of service attack. In our case, we cannot fully provide protection from denial of service attacks, because availability is very much a concern beyond the boundaries of information security, involving other areas and techniques like real-time and fault tolerant computing.

The three goals of information security described before can also be interpreted as three different aspects of the same goal: to prevent unwelcome events by controlling access to resources. Still, in the real-world things can (and will) go wrong, no matter the effectiveness of access control methods, so we should add another aspect to the definition of security: accountability. It can be described as the property that enables activities on a system to be traced to individuals (or entities), who may then be held responsible for their actions. There are two important functional components to achieve this goal: security audit, which logs security relevant events, and non-repudiation, which generates evidence of user (or entity) actions.

The term dependability is concerned with the reaction of an IT system to failures, i.e., situations in which the proper performance is required in the face of adverse conditions. Dependability is a desirable property for any security system, because it will increase users' confidence on the system they have to rely on. However, in our situation, addressing dependability is hard because it heavily depends on layers and mechanisms outside the scope of our work.

Security is a difficult technical problem, but it is much more than that. Security management is also very important when we are dealing at an organizational level, because many times the real cause of a security problem is the management process and social inadequacies. It is often people that introduce vulnerabilities on a technically sane solution, because they are untrained, unaware or just don't care to follow the rules.

Good security management has several objectives:

- Definition of an enterprise wide security policy.

- Identification, implementation and documentation of appropriate countermeasures through a process known as Risk Analysis.
- Evaluation of the effectiveness of the designed countermeasures based on the feedback of the audit and analysis process.

Besides the search and deployment of technical solutions to the security of the system, it is equally important to define a global policy to be followed by everyone within the organization.

There are three key concepts that appear in every discussion about security: threats, vulnerabilities and countermeasures.

A threat is a possible danger to a system (might be a person, a thing or an event) that might exploit a vulnerability of the system. Threats are a potential system misuse that could lead to a failure in achieving the security goals, and they fall into three main categories: environmental (fires, floods...), accidental misuse (caused by ignorance and insufficient training of users), and intentional attacks (by outsiders or insiders).

Vulnerability is a point where a system is susceptible to attack, i.e., is a weakness in the security system that might be exploited by a threat. A threat that exploits a vulnerability is said to perpetrate an attack on the system. Vulnerabilities are often the result of trade-offs made in the system design and implementation, usually to achieve increased performance or additional functionality.

To successfully protect the resources of the system we have to reduce vulnerabilities. Countermeasures are methods that allow doing just that. An effective set of countermeasures is constituted by three different but equally important, parts:

- Protection: prevents assets from being damaged before the attack happens. An example of this would be locks on the doors or a safe.
- Detection: warns when an asset gets compromised and can give information on how it happened and who is responsible. Alarms are the most common examples of this kind of countermeasures.
- Reaction: allows recovering the compromised assets or recovering from damage to assets after they have occurred. For example, police or security

guards running towards the compromised area (usually after detection has been made).

In practice, information security is concerned with identifying threats and vulnerabilities in systems, as well as protecting against threats to those systems by devising and applying appropriate countermeasures.

To conclude this section we once again return to security definition.

Security is a collection of safeguards that ensure the confidentiality of information, protect the systems or networks used to process it, and control access to them. Security typically encompasses the concepts of confidentiality, integrity and availability.

Managing security effectively is hard. One has to approach security from a wide perspective, including both organizational and technical aspects.

And as Bruce Schneier says [1], Security is not a product (static, finished and complete), it's a process (dynamic, ever-evolving and fluid).

3.2 - Distributed Security

You know you have a distributed system when the crash of a computer you have never heard of stops you from getting any work done

– Leslie Lamport

Distributed systems, such as CORBA based middleware systems we are considering, are more vulnerable to security breaches than more traditional host-centric systems, because there are more places where attacks can be launched against the system.

When systems become large, problems do not scale up linearly. There is often a qualitative change in complexity and some things that are trivial to deal in a stand alone system or a small set of machines and principals, will suddenly become a big problem. Principal is a human user or system entity that is registered in and authentic to the system [44].

Obviously the primary point of attack is the network, which is often available to anyone, or at least to a large number of people and systems. Several attacks exist in such an environment, because system interactions are often unpredictable and very complex: communications can be eavesdropped upon (passive attack) and attackers can masquerade as legitimate authorized participants (active attacks).

In addition to the increased surface area susceptible to threats and vulnerabilities, there are also other issues that further complicate the security enforcement and administration in distributed systems:

- Mutual Distrust: in distributed systems, a single object can be client for one request and server for another, and this complicates trust relationships. Add to this the fact that most of the times the client or the network cannot be trusted to ensure the protection of the server's resources from unauthorized access.
- Naming Problems: objects in distributed CORBA systems do not have a unique identifier, because this characteristic was rejected by the designers due to being perceived as an obstacle to scalability (this is an example of a trade-off). Objects are located through their object reference, which changes from time to time, and objects can have more than one identifier (or none at all).

- Dynamic Interaction: in some distributed object based systems (like those based on CORBA) it is hard to know who the other party is, because there are many flexible interactions between objects: clients can become servers, some objects can delegate at run-time parts of its implementation to other objects, and firewalls and proxies complicate even further the topology of the network communication path.
- Scalability: by definition and by design distributed systems can scale (almost) without limit and, consequently, it is difficult to enforce security in very large systems (e.g., thousands of objects and nodes). Large distributed systems are hard and cumbersome to administer (especially when they are geographically distributed and administrators do not trust each other), and often have several security policy domains (with different security requirements).
- Layers of Abstraction: in a middleware based distributed system, because of the transparency it provides, there is always a lot of behind the scenes activity, making it very difficult to administer and even to understand the interactions occurring. CORBA-like middleware architectures are highly layered, with complex conversions at layer boundaries, complicating even further the already complex layered security architecture. This is a major problem in distributed systems because it is difficult to have a clear and precise picture of the whole systems and of the object interactions.

3.3 - Authorization Models

Central to our work is the problem of authorization or access control. We will now analyze it in more detail.

Authorization mechanisms ensure that the rules governing the use of system resources and services are enforced [18].

Authorization relies on authentication to determine the subject's identity. This identity verification can be made by checking: what the user knows (e.g., password), what the user has (e.g., token, certificate) or what the user is (e.g., eye, fingerprint).

Access control mechanisms allow a system owner to enforce those rules when rule check and enforcement are possible. When checking and enforcement of the rules are not possible, data protection mechanisms, such as data encryption, are used.

The structure of traditional access control mechanisms can be viewed using the conceptual model of reference monitor.

A reference monitor is a part of the security subsystem, responsible for mediating access by subjects to system resources (traditionally called objects), as illustrated in Figure 3-1.

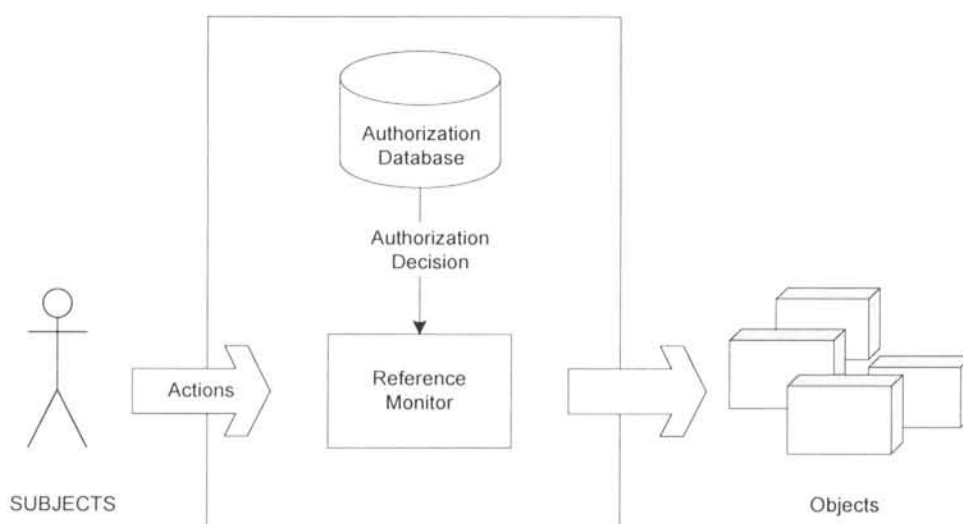


Figure 3-1: Conventional Access Control, with Reference Monitor (source [18]).

The mediation consists of making authorization decisions, by checking access requests against a set of authorization rules (policy) from the authorization database (storage of such rules) and enforcing them. Authorization rules commonly specify which subjects can perform what actions on what objects, and permitted actions are called access rights.

The reference monitor, when making an authorization decision, uses the authorization rules and three groups of information: the access request, the subject who made the request and the object to be accessed.

There are several problems associated with access control in distributed systems. First when designing a system we must address several concerns:

- A distributed system may consist of a large set of interconnected applications. It is difficult to enforce organization-wide security policies across these applications and provide ways to ensure the consistency of enforcement across application boundaries.
- Usually the environment of the system changes frequently (insertion or deletion of applications, changes in business processes or hardware/software platforms).
- It must be implemented and operated at reasonable cost.

3.3.1 - Design Approaches

There are three main approaches described in the literature of controlling access to resources in distributed and heterogeneous application systems: policy agents, interface proxies and interceptors, and authorization servers.

The central idea behind policy agents is centralized access control management via rule translation into languages supported by local access control mechanisms, and distribution of the rules across the systems, achieved with the help of policy agents that reside on computers hosting applications.

The concept of policy agents has the great advantage of accommodating the existing applications and technologies already deployed, and this is its main motivation. It is also a highly fault tolerant approach.

On the other hand, it is hard to achieve the consistency of the policy across different applications, and also to automate the mapping of the global policy into the languages of the various access control mechanisms.

For the second approach, interface proxies and interceptors, the access to an application is controlled externally; authorization decisions are made and enforced before an application gains control. This is the design alternative used in CORBA, where invocations are intercepted in the middleware layer. The reference monitor is implemented externally to the application system.

The main advantage of the interceptors approach is that it does not require changes to the application system. However, because there are as many instances of access control proxies as applications, consistency of enforced policy and coherence authorization data is difficult to ensure.

There is also another limitation that should be noticed: variables, whose values become available at some point after the method is invoked but before an access control decision is made, can not be used.

When using authorization servers in the design of access control systems the reference monitor is embedded in the application, enabling it to enforce its own access control policies.

In some cases, the access control decision is performed by a dedicated authorization server external to the application system, allowing the decoupling of the reference monitor decision part from an application.

Due to its logical centralization of access control rules, it brings several important advantages: the authorization policies enforced are inherently consistent and coherent; policy change and update is extremely easy; can significantly reduce the cost of access control administration; and since authorization logic is centralized and decoupled from the application logic, it is possible to change policy type without affecting application systems.

This centralization approach also has some limitations: the authorization server has to be carefully designed so that it does not become a performance bottleneck; in case of server failure the system would be rendered unoperational.

3.3.2 - Access Control Models

Also important when discussing access control is the model used.

Most of the research and development efforts done in access control technology are supported by the defense sector, especially the American Department of Defense (DoD).

This research has resulted in two fundamental types of access control: Discretionary Access Control (DAC) and Mandatory Access Control (MAC). DAC enforces access control to an object on the basis of permissions or denials or both configured by an individual user, typically the object's owner. MAC enforces access controls on the basis of security labels attached to users, more precisely, to subjects and objects.

DAC permits the granting and revoking of access control privileges (to any object) to be left to the discretion of the individual users. As such, users are said to be the owners of the objects under their control. For some organizations, the corporation is the actual owner of system objects as well as the programmes that process them. Access priorities are controlled by the organization and are often based on employee functions rather than data ownership.

MAC, as defined in the DoD's Trusted Computer Security Evaluation Criteria (TCSEC) [36], is "A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity."

With this mechanism it is the system that enforces the security policy over the wishes of the object owner. It is mainly used in classified military environments.

These policies for access control are not particularly well suited to the requirements of government and industry organizations that process unclassified but sensitive information. Such environments usually require the ability to control actions of individuals beyond just an individual's ability to access information according to how that information is labeled based on its sensitivity.

There is also a third model that has been receiving a lot of attention on the last decade: RBAC. It enables the security policies be specified in a way that maps more naturally to an organization's structure.

We will now center our analysis on RBAC, our model of choice to be integrated in the MBE architecture.

3.3.2.1 - RBAC

The basic concept of RBAC [20] [59] is that users are assigned to roles, permissions are assigned to roles, and users acquire permissions by being members of roles. A major purpose of RBAC is to facilitate security administration and review [21].

With role-based access control, access decisions are based on the roles that individual users have as part of an organization. Access rights are grouped by role name, and the use of resources is restricted to individuals authorized to assume the associated role. The process of defining roles should be based on a thorough analysis of how an organization operates and should include input from a wide spectrum of users in an organization.

Roles are created for the various job functions in an organization and users are assigned roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles as needed.

This simplifies the administration and management of privileges; roles can be updated without updating the privileges for every user on an individual basis.

In fact one huge advantage of the RBAC model resides in the more stable nature of role-permission association. A NIST study indicates that permissions assigned to roles tend to change relatively slowly compared to changes in user membership of roles.

The particular combination of users and permissions brought together by a role tend to change over time. The permissions associated with a role on the other hand are more stable (they tend to change less often than the people who fill the job function

that role represents). Therefore, basing security administration on roles rather than on permissions is simpler.

Once an RBAC framework is established for an organization, the principal administrative actions are the granting and revoking of users into and out of roles. This is in contrast to the more conventional and less intuitive process of attempting to administer lower-level access control mechanisms directly (e.g., access control lists) on an object-by-object basis.

This access control method was first introduced by David Ferraiolo and Richard Kuhn [19], but some significant divergences in models and implementations led to a standardization effort, involving NIST.

The NIST RBAC model is defined in terms of four model components: Core RBAC, Hierarchical RBAC, Static Separation of Duty Relations, and Dynamic Separation of Duty Relations [22].

Core RBAC defines a minimum collection of RBAC elements, element sets, and relations in order to completely achieve a role-based access control system. This includes user-role assignment and permission-role assignment relations, considered fundamental in any RBAC system. Core RBAC includes requirements that user-role and permission-role assignment can be many-to-many. Thus the same user can be assigned to many roles and a single role can have many users. Similarly, for permissions, a single permission can be assigned to many roles and a single role can be assigned to many permissions.

The Hierarchical RBAC component adds relations for supporting role hierarchies. A hierarchy is mathematically a partial order defining a seniority relation between roles, whereby senior roles acquire the permissions of their juniors and junior roles acquire users of their seniors.

A third model component, Static Separation of Duty (SSD) Relations, adds exclusivity relations among roles with respect to user assignments. Conflict of interest in a role-based system may arise as a result of a user gaining authorization for permissions associated with conflicting roles. Because of the potential for inconsistencies with respect to SSD Relations and inheritance relations of a role

hierarchy, the SSD relations model component defines relations in both the presence and absence of role hierarchies.

The fourth model component, Dynamic Separation of Duty (DSD) Relations, defines exclusivity relations with respect to roles that are activated as part of a user's session. DSD requirements limit the availability of the permissions by placing constraints on the roles that can be activated within or across a user's sessions.

Of interest to our work is the Core RBAC, which was known before as the RBAC₀ reference model [21]. Now, we will do a brief description of this model, as well of associated definitions.

Core RBAC includes sets of five basic data elements called users (USERS), roles (ROLES), objects (OBS), operations (OPS), and permissions (PRMS). The RBAC model as a whole is fundamentally defined in terms of individual users being assigned to roles and permissions being assigned to roles. As such, a role is a means for naming many-to-many relationships among individual users and permissions. In addition, the Core RBAC model includes a set of sessions (SESSIONS). Session is a mapping between a user and an activated subset of roles that are assigned to the user.

Core RBAC model element sets and relations are defined in Figure 3-2.

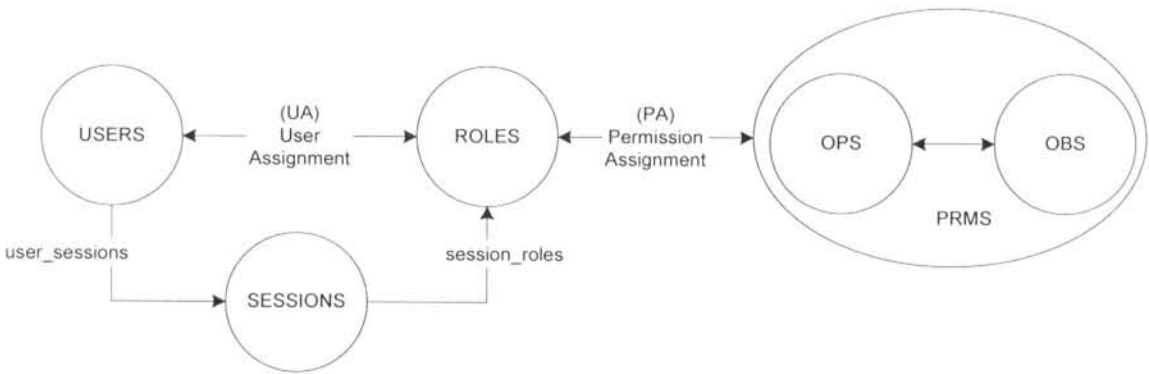


Figure 3-2: Core RBAC Model.

A role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role. Permission is an approval to perform an operation on one or more RBAC protected objects. An operation is an executable image of a programme, which upon invocation executes some function for the user.

Central to RBAC is the concept of role relations, around which a role is a semantic construct for formulating policy. Figure 3-2 illustrates user assignment (UA) and permission assignment (PA) relations. The arrows indicate a many-to-many relationship (e.g., a user can be assigned to one or more roles, and a role can be assigned to one or more users).

Without these conveniences there is an enhanced danger that a user may be granted more access to resources than is needed because of limited control over the type of access that can be associated with users and resources. Any increase in the flexibility of controlling access to resources also strengthens the application of the principle of least privilege.

Although RBAC is policy neutral (it results from the configuration of the relationships between users, roles and permissions), it directly supports three well-known security principles: least privilege, separation of duties, and data abstraction:

- Least privilege is supported because RBAC can be configured so only those permissions required for the tasks conducted by members of the role are assigned to the role.
- Separation of duties is achieved by ensuring that mutually exclusive roles must be invoked to complete a sensitive task, such as requiring an accounting clerk and account manager to participate in issuing a check.
- Data abstraction is supported by means of abstract permissions such as credit and debit for an account object, rather than the read, write, execute (RWX) permissions typically provided by the operating system.

Summarizing, RBAC is a family of reference models in which permissions are associated with roles and users are assigned to appropriate roles. A role can represent

competency, authority, responsibility or specific duty assignments. Some variations of RBAC include the capability to establish relations between roles, between permissions and roles, and between users and roles. A major purpose of RBAC is to facilitate access control administration and review.

Chapter 4 - Security in MBE

After the general introduction to security concepts and the detailed discussion of the RBAC model, we will come back to the MBE architecture and examine it in terms of security.

Fundamental to any secure design is the identification of threats and vulnerabilities of a system. We shall begin by doing a threat analysis of the MBE system.

Next we describe several security components that can be used in the selection process of our design stage.

Finally, we will discuss the most important selection criteria and the constraints found during this process.

4.1 - Threat Analysis

To really secure a system we first need to know things like “Secure from whom?” or “Secure for how long?” [1].

We need a way to model the various threats against the system we are trying to protect. To design and implement effective countermeasures to thwart the attacks, we first need to understand the different ways a system can be attacked [24].

To facilitate this process of threat modeling we can use attack trees. Attack trees have been used in various forms, and under various names, for many years, but have most recently been defined as a systematic method to describe and characterize system security based on varying attacks.

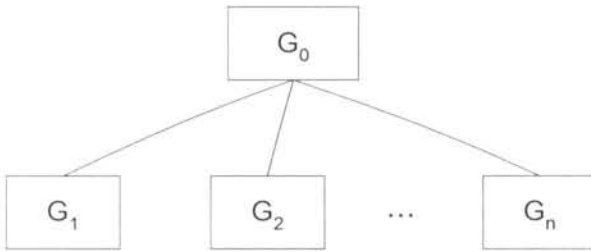
Using this method, attacks against the system are represented in a tree structure, with the goal as the root node and different ways of achieving the goal as leaf nodes. Attack trees refine information about attacks by identifying the compromise of enterprise security or survivability as the root of the tree [25]. We can have a set of attack trees, forming a forest, with the root node of each tree representing an event that could severely harm the system.

The attack tree is also called an and/or tree, because we decompose a node either as:

- a set of attack sub-goals, all of which must be achieved for the attack to succeed; this is an AND-decomposition, or
- a set of attack sub-goals, any of which must be achieved for the attack to succeed; this is an OR-decomposition.

Attack trees can be represented graphically or textually. AND-decomposition is represented as follows:

Graphical:

Textual: **Goal** G_0 **AND** G_1 G_2

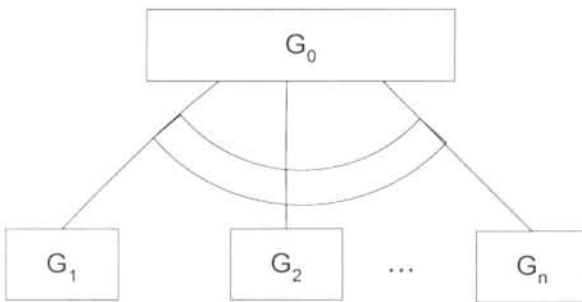
...

 G_n

Figure 4-1: AND-decomposition

We represent the OR-Decomposition as follows:

Graphical:

Textual: **Goal** G_0 **OR** G_1 G_2

...

 G_n

Figure 4-2: OR-decomposition

We prefer the textual representation for the rest of our analysis.

To generate an individual intrusion, i.e., a unique attack on the system, we transverse the tree in a depth-first manner.

Then by assigning values to the nodes, we can make some basic calculations with the tree to make statements about different attacks against the goal. The values assigned to a node can be any yes/no value, like “possible” versus “impossible”, “easy” versus “difficult”, “special equipment required” versus “no special equipment”. We can also directly assign a cost to each node to show exactly how expensive an attack can be.

It should be stressed that the values assigned will change over time, as attacks become easier and we gather more exact information on the values.

After creating the attack tree we use it to take security decisions. We can determine if the system is vulnerable to a particular type of attack (by looking at the root node) or calculate the impact of a system modification or the discovery of a new vulnerability.

One important characteristic of the attack trees is that they capture knowledge in a reusable way. Once we have completed an attack tree on a specific system we can use it in every new occasion where that system is deployed, becoming in fact part of a larger attack tree. This scalability allows us to build a library of attack trees against particular systems and enables us to not be an expert in everything; all we need to know are the values of the root node.

For our threat analysis we will only focus on problems that are specific to the MBE and other middleware based architecture and leave out all security problems generally related with the network, such as viruses, outside intrusions, password management problems, failure to patch software packages, etc. For instance, if the underling network is vulnerable to DOS attacks, then our system cannot preserve availability.

The MBE architecture is based on CORBA technology, and thus presents many (if not all) of the security problems associated with this technology.

Next we present our attack tree. The goal to be achieved is unauthorized access to information in the system, be it essence or metadata.

Goal Unauthorized access to media assets (essence and/or metadata)

OR 1. Impersonation: user masquerading as someone else

OR 1. Discovering someone else's authentication token (password ...) - this falls outside the scope of protection of our system.

2. Unwanted delegation: a user may delegate his rights to other objects, so they can act on his behalf, and this adds the threat of rights being delegated to widely.

2. Eavesdropping: "listening" on a communication line.

OR 1. Inside threat: employee monitors connections on intranet (normally on shared media as Ethernet).

2. Outside threat: Gain privileged access to machines on intranet connected via Internet - this falls outside the scope of protection of our system.

3. Get trusted process to "leak" or collect sensitive information: it can then forward it to an outside attacker, or be recovered by a trusted insider.

3. Authorized user of the system gaining access to information he is not allowed, and that should be hidden from him.

4. Direct access to objects and interfaces: bypassing controls and the way the system was designed to be used.

OR 1. Using pointers (C or C++) to directly access objects.

2. Querying naming service to get CORBA reference - this falls outside the scope of protection of our system.

5. Tampering: modifying, inserting or deleting items and message on communication between objects.

We have to implement several countermeasures to these security threats.

First, we have to assure transport message protection, e.g., using cryptographic techniques. This involves:

- Protection of the normal ORB message flow related to each invocation.
- Protection of the low-level essence connections.

This will protect from eavesdropping and tampering.

Second we have to control the access to an interface (that is, the set of all implementations of an IDL interface), as well as to the interface operations. This will decrease vulnerabilities 3 (Authorized user gaining access to prohibited information) and 4 (Direct access to objects and interfaces).

Next, to deny impersonation and in part vulnerabilities 3 and 4, it is necessary to make the identification and authentication of:

- The user to the CORBA system.
- The client to the target.
- The target to the client.

It is also necessary to have Delegation mechanisms implemented in our security system, to avoid uncontrolled delegation.

There is one important problem that we will not address: accountability.

The more appropriate countermeasure would be to implement a security audit log mechanism; this is by itself a complicated matter, because it would have to be integrated with an already existent log mechanism, and we deferred this problem to a subsequent phase.

4.2 - Security Components and Technologies

We should now select the security components that we will use in the design of the system. First, there is a description of the technologies under consideration, which is limited by what is already in use in the available implementation of the MBE architecture.

Technologies such as DCOM and Java RMI were discarded in the development process of the MBE architecture because of several inherent limitations. Instead, CORBA was the distributed object-oriented computing technology chosen as the basis of the MBE architecture.

Other security technologies will also be briefly discussed because they are the support of some of the tools used in our system.

In the next section we will discuss the most important selection criteria and the constraints found during this process.

4.2.1 - CORBA

The Common Object Request Broker Architecture (CORBA) is an industry standard developed and enhanced since the early nineties by OMG, a not-for-profit consortium with over 800 members, which develops specifications for object-oriented systems.

The CORBA technology provides a general-purpose infrastructure for developing and deploying distributed object based systems in a broad range of specialized application domains.

CORBA has several important strengths that should be mentioned:

- Openness: CORBA is completely free and is developed by an open membership association (OMG).
- Platform and Language Independence: CORBA has the ability to run in (almost) any platform (from the big iron servers to embedded space) and to be used in conjunction with every major programming language (there are bindings for such distinct languages as C++, Java, Python and TCL)
- Common Object Services (COS): a useful set of already specified services that implement commonly needed features, simplifying the development process.

CORBA is built around three basic blocks: the OMG Interface Definition Language (IDL) [39], the Object Request Broker (ORB) and the Generic Inter-ORB Protocol (GIOP).

IDL is a neutral declarative intermediate language that is used to specify an object's interface with any potential client. It is the key to CORBA interoperability because it strictly defines an object's interface and separates it from implementation details.

The ORB is responsible for almost all of the under-the-hood work in CORBA. Every CORBA object is identified by an opaque handle known as Interoperable Object Reference (IOR). It is the ORB that manages almost everything when a client invokes a method on the object.

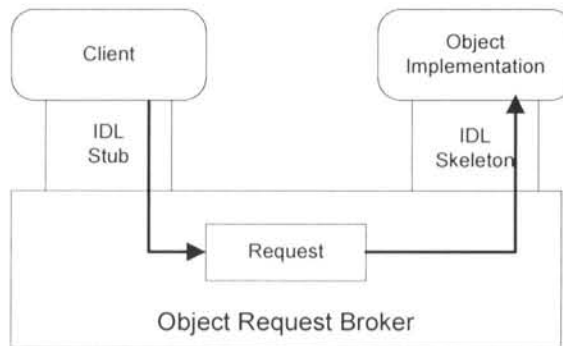


Figure 4-3: A Request Passing from a Client to a CORBA Object Implementation

Figure 4-3 shows how an invocation occurs in CORBA. The stub and the skeleton are code adaptation layers written in the same language as the object implementation, and generated by the IDL compiler.

The GIOP defines a set of rules that are made concrete on actual protocols. It is the basic language ORBs use to communicate. They can use derivatives of GIOP, but all of them must support the basic IIOP, or Internet Inter-ORB Protocol.

We can now briefly summarize some CORBA concepts.

All entities in the CORBA computing model are identified with interfaces defined in IDL. A CORBA interface is a collection of three things: operations, attributes, and exceptions. Object functionality is exposed to other CORBA based applications only

through the corresponding interfaces, completely separate and independent from their particular implementation.

An implementation of a CORBA interface is called a CORBA object. It is possible to have several different implementations for the same interface. Objects have object references by which they can be referenced. An object reference is a handle through which one requests operations on the corresponding object.

A quick explanation is needed: when the meaning does not cause confusion and it is clear by the context, we will use the term “CORBA object” or just “object” to mean “implementation of a CORBA interface”.

Summarizing, CORBA is a technology that provides a middleware foundation that:

- Allows the choice of the implementation language, operating system and network using a definition language
- Separates interface from implementation enabled by OMG IDL:
 - The interface to each object is defined very strictly
 - The implementation of an object - its running code, and its data - is hidden from the rest of the system
- Provides physical location abstraction

4.2.2 - CORBA Security Service

The CORBA Security (CS) standard [40] was developed by OMG to meet all the different security policy requirements of distributed and interoperable systems, and is a part of the Common Object Services.

The CORBA Security Service, or CORBASec, defines several functionalities that are visible to application developers and security administrators: identification and authentication, authorization and AC, auditing, message integrity and confidentiality protection, authentication of clients and target objects, optional non-repudiation, administration of security policies and related information [26].

It should be noted that one of the main objectives of CORBASec is to be totally unobtrusive to application developers. This service is transparent to security-unaware objects, which are able to run securely on a secure ORB without any modification to their implementation. On the other hand, it must be possible for security-aware objects to exercise even stricter security policies than those that are implemented in CS. In the CORBASec model, all object invocations are mediated by the appropriate security functions in order to enforce various security policies such as AC.

As with any security model, CS has the concept of entities that are “first causes” of security relevant events: principals [27]. Associated with it are a set of attributes (identity and privileges) that represent the information the service requires to the decision making process on what the principal is allowed to do. This information is encapsulated within a Credentials object and constitutes the identity of the new subject, who initiates requests on CORBA objects on behalf of the user. The authentication of a user results in a Credential object.

CS establishes secure associations between the client and an object which provide identification and authentication of the client to the object and vice-versa, protected transfer of credentials between the client and the object, and a way to negotiate acceptable transport message protection. The mechanisms used to provide this protection are outside the scope of CS, and must be implemented externally using public or secret key cryptography. CS assumes direct network connection between client and server and security policies are enforced end-to-end.

In any moderately complex system managing every security related object becomes extremely complicated, and so various techniques were developed to simplify the

administrator's job by grouping similar objects. CS architecture achieves performance and administration scalability by means of policies and policy domains, where any security policy is associated with a policy domain (or just domain). Policies of more than one type could be associated with the same domain and each object can belong to more than one policy domain.

To further simplify the managing of a system, domains could be organized in federations, hierarchies or be completely unrelated. CORBA objects residing on different computers can be associated with the same domains.

Since access control decision can be specific to each object, the model scales very well without losing fine granularity.

The CS specification only provides an object invocation access control service, i.e., access controls that the ORB enforces between the object interfaces and the implementations of these interfaces, each time an object makes an invocation on another. AC can be enforced completely outside of an application system because the enforcement occurs at the ORB level. Everything, including obtaining information necessary for making authorization decisions, is done before the method invocation is dispatched to the target object.

Figure 4-4 shows that when a message from client application to a target object is passed through the ORB, the policy enforcement code is executed inside of CS enforcement sub-system. This enforcement uses three sources of information for making decisions, and is performed both at the client and the target ORB. The first source of information is the policy of the domain(s) which the target belongs to. Then it verifies the credentials of the client. Finally the message (a request to invoke an operation on the target object) itself used in the access control enforcement is a request to invoke an operation on the target object.

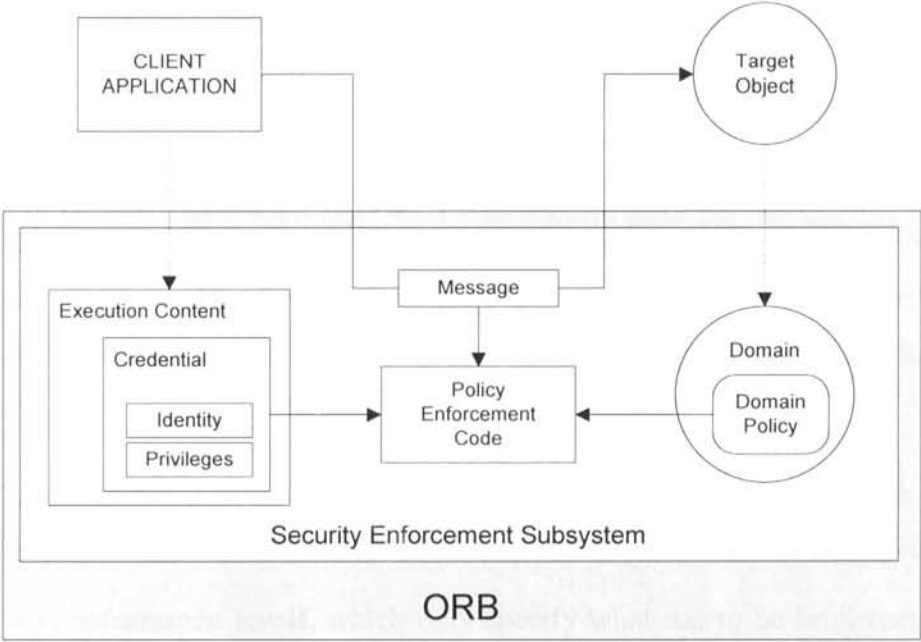


Figure 4-4: Enforcement of Policies in CORBA Security (source [26])

In CS, objects are placed in policy domains, which allows having the same access policy governing all the domain members. It is important to notice that CS allows stating policies in terms of operations implemented by the subject and object. These operations are grouped via rights required for invoking them. Access Control policies control which subjects can invoke what operations on what objects in the domain the policies are defined on. For a subject to be granted the rights to execute an operation his privilege attributes must match the required rights of the operation.

Delegation complicates access control in CORBA. Sometimes, upon receiving an invocation, a target object must become a client to other objects to form a response to the original request. To the second (or higher) order request, this object could need a set of privileges from the first request. This is called privilege delegation where the initiating principal's privilege attributes are delegated to intermediate objects until they reach the final target. All access control protocols will use these delegate privileges to make their decision.

There are several types of delegation policies defined in the CS specification: no delegation (intermediate object uses only its own privilege attributes), simple delegation or impersonation (intermediate objects simply use the initiating principal's

privileges), combined privileges delegation (intermediate objects combine in an undistinguishable manner their and the initiating principal's privileges), composite delegation (intermediate objects pass to the target their and the initiating principal's privileges, who then checks both against its access policy) and trace delegation (extension of the former, where every intermediate object does a composite delegation, which enables the target to trace the entire delegation sequence). These policies were defined to support a broad range of specific organizational needs.

The CS specification is long and sometimes complex, and most vendors were not able or willing to implement it completely. So, two levels of conformance were specified.

These two conformance levels, which only specify what has to be implemented and not how, have different characteristics and features [4]:

- Level 1: it is mandatory to provide a specified set of security features during object invocation for security-unaware applications and to allow security-aware applications to enforce their own access control and auditing;
- Level 2: supports a wider range of security facilities on top of level 1 functionality.

CS standard has many important characteristics that make it suitable to use in enterprise scalable distributed environments: user grouping via privilege attributes, object grouping via policy domains, and method grouping via the concept of required rights.

But, still, some applications need to have a finer-grained control policy, and should implement it themselves. A security-aware application can do so with the help of CORBA Security interfaces defined by OMG and based on the two conformance levels. As an example, for an application to be able to enforce access control policies, it needs to know who wants to access, what protected resource, and in what way.

Interface `SecurityLevel1::Current`, available in the CORBASec API, defines the method `get_attributes()` for obtaining subject security attributes, and provides to an application a means to find out “who”.

So we can conclude that in the CS model access control policies can be enforced either by the ORB (security-unaware object) or by the application (security-aware).

4.2.3 - Common Secure Interoperability version 2

CSIV2 is a standardized security architecture for CORBA that has been recently adopted by OMG [39]. It is also important to notice that it is endorsed by the Enterprise Java Beans specification for EJB security. This ensures interoperability of CORBA with EJB, as well as between CORBA systems.

The CSI architecture is organized into two different layers [53]:

- Transport Layer (CSIV1): SSL or SECIOP;
- Authentication Layer: SAS Protocol.

Both of these layers interact in such a way to give security contexts that can last over connections or just for one request.

The Transport Layer is basically Common Security Interoperability Version 1. Version 1 handled security down at the transport layer, i.e. closely wrapped around TCP/IP. It covers the secure establishment of the connection from client to server.

The normal transport layer ORB protocol is commonly known as IIOP (GIOP over TCP/IP). CSIV1 supports different transport layer protocols: SSLIOP (required) and SECIOP (optional, Kerberos based). SSLIOP defines the transport layer as using SSL over TCP/IP. So, basically SSLIOP is GIOP over SSL over TCP/IP. SECIOP defines the transport layer as using SECP, an OMG defined transport Protocol. Before messages start flowing freely over the channel, the secure SSL, or SECP endpoints negotiate the channel protection, such as encryption and integrity protection.

This Authentication layer is implemented using the CORBA Secure Attribute Service (SAS) Protocol (chapter 24 of [39]), running over a secure transport layer. SAS is a secure interoperable wire protocol that tightly defines the messages and the IDL, and supports client authentication, delegation and privileges.

The SAS protocol is divided in two layers:

- Supplemental client authentication layer: used to perform additional client authentication where sufficient authentication could not be accomplished in the transport.
- Security attribute layer: provides identity assertion, enables pushed security privilege attributes and proxy endorsement.

The identity assertion capability is the foundation of a general-purpose impersonation mechanism that makes it possible for an intermediate to act on behalf of some identity other than itself.

This layer also enables a client to push (i.e., deliver) security attributes (identity and privilege) to a target where they may be applied in access control decisions. It normally uses the standard CORBA ATLAS service (described later) to acquire authorization tokens.

The different layers and functionalities of the CSIv2 architecture can be seen in Figure 4-5.

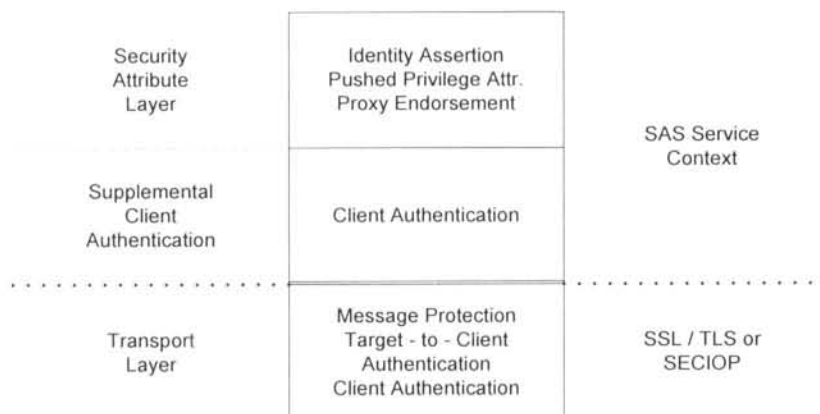


Figure 4-5: CSIv2 Architecture

The SAS protocol supports the establishment of both transient and reusable security contexts. Transient contexts, also known as stateless contexts, exist only for the

duration of the GIOP request that was used to establish the context. Reusable contexts, also known as stateful contexts, endure until they are discarded, and can be referenced for use with subsequent requests.

CSIV2 is a very flexible architecture, also offering multiple forms of delegation (both simplified and extended), degrees of authorization (username or extended attributes for authorization) and federation.

Finally, three conformance levels (0, 1, 2) were defined to allow an evolving support from the implementations of this architecture. Level 0 defines the basis of secure interoperability that all implementations are required to support. There is also another differentiating fact among implementations: stateful conformance, i.e., whether or not they support stateful security contexts.

4.2.4 - Authorization Token Layer Authentication Layer

ATLAS is defined by OMG [42] as a part of Common Object Services already mentioned. It is a service needed to acquire authorization tokens to access a target system using the CSIV2 protocol.

ATLAS is a service that defines a single interface by which a client's security service (CSS) acquires an authorization token to deliver to a target's security service (TSS).

CSS needs to deliver an authorization token (information processed by a TSS for security purposes) that contains privilege information that is comprehensible to the target and within the target's privilege scope.

To facilitate secure interoperability, the TSS indicates to a client the location of a specific ATLAS that defines the target's privilege scopes (thus, acting as privilege authorities).

The CSS retrieves from that ATLAS an authorization token (AT), and the CSS is guaranteed that the token is understood by TSS.

An ATLAS server only delivers AT for one privilege scope, but many different targets may belong to the same privilege scope. This privilege scope represents ATLAS authorization domains.

ATLAS solves the problem of acquiring the privileges (authorization token) needed for a client to acquire a set of privileges (particular custom-made token) the target will understand, because it relieves the client from having the obligation of understanding the format of the token retrieved from the ATLAS service.

ATLAS is in fact a service that allows to plug-in personalized authorization tokens into the CSIV2 protocol.

4.2.5 - CORBA Component Model

CCM provides reusability of software components, automatic deployment and configuration of components and the description of implementation details in a standard way that enables automatic code generation of parts of the component.

The current CCM specification defines several models, among which we specify the following:

- The execution model: defines containers, runtime environments for component instances and their homes (Home is a CORBA Object that is created at installation time and serves as a factory of Component instances of that type). Several containers could be hosted by a common component server. An interesting feature of component servers is the ability to download archives from repositories. Thus, downloading packages can be done at any time, and more precisely only when a component is required.
- The abstract model: defines interfaces and properties of components. A component can offer multiple interfaces (as shown on Figure 4-6), each one defining a particular point of view to interact with the component. Two interaction modes are provided: facets for synchronous invocations and event sinks for asynchronous notifications. Moreover, a component can define its required interfaces, which define how the component interacts with others: receptacles for synchronous invocations, and event sources for asynchronous notifications.

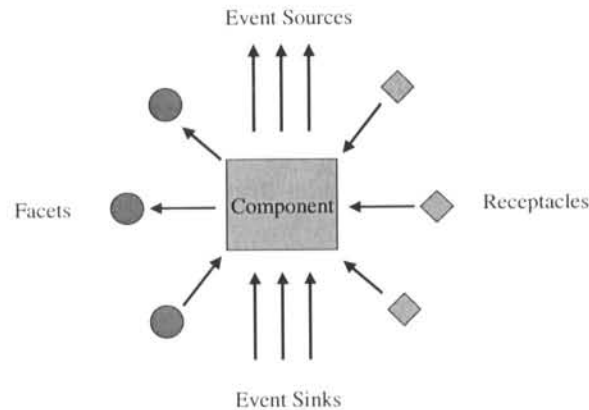


Figure 4-6: CORBA Component (source [65])

4.2.6 - Kerberos

Kerberos [55] is a distributed authentication protocol that originated from the Athena Project at MIT [28]. The protocol identifies users on a TCP/IP network and then authenticates them to application servers. Kerberos is also the name of the free software programme that implements the protocol, but is also available in many commercial products as well. The software and the protocol are both named after the three-headed dog that guards the gates of Hell in Greek mythology. Kerberos is currently at version 5. Version 4 was the first to be published - versions 1, 2 and 3 were test-beds.

The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server use Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

The idea is to allow users to log in to a single computer, and for all other necessary authentication to occur transparently. It works surprisingly well, and an awful of lot happens behind the scenes to make it so. The idea of single sign-on has recently re-surfaced in MS Passport. Microsoft wisely adopted a slightly modified version of Kerberos for Windows 2000, and it is now the default mechanism for network authentication.

The principles behind Kerberos are fairly simple - the protocol only consists of 5 basic messages - but making it work in practice can quickly become complicated.

The main components of this network authentication protocol are:

- A Kerberos server contains records of all clients.
- A client is generally a user or a programme that needs access to an application server.
- An application server is any computer that runs services (such as telnet or FTP) that require Kerberos authentication from a key distribution center before they can be used.
- A key distribution server (or KDC) can use information given to the client by the Kerberos server to authenticate the client to various application servers.

The Kerberos server shares secret keys with all the clients, key distribution centers, and application servers. When the client first logs on, it sends its identity to the Kerberos server, along with the identity of the key distribution server that the client uses. The Kerberos server looks up the client's identity in its own database.

If it finds the identity of the client in its database, it generates a ticket-granting ticket (TGT) for that client. Tickets are Kerberos data structures that can be safely sent across a network that, when valid, prove to its recipient the authorization of the client to use the recipients' services. All tickets in Kerberos are encrypted with the key of the recipient, and adhere to a standard structure which among other things contains: the client identity and session key.

The TGT is used by the client to request tickets from the key distribution centers. The ticket generated by the server is encrypted with the shared secret key of the key distribution center and the client session key is copied and encrypted with the shared key of the client. Both the ticket and the encrypted client session key are then sent to the client.

If the client has the correct password, then it may decrypt the session key. No further communication is required with the Kerberos server. Subsequent communication takes place with the KDC, and must be encrypted with this session key.

The next step is normally the client requesting access to the services controlled by the KDC. To gain access to the services, the client must possess a ticket from the KDC. To get this ticket, the client creates an authenticator to send to the KDC.

Both the ticket-granting ticket and the authenticator are then sent to the KDC. If the received authenticator decrypts successfully the KDC generates a new ticket. The recipient of this ticket is the service that the client wishes to access, and as such the ticket is encrypted with the key of the service. A new session key is generated for the ticket by the KDC.

The KDC encrypts the new session key with the old session key, and sends this and new ticket to the client. The client composes a new authenticator, this time for the service it wishes to access. This authenticator is encrypted with the new session key.

The client then sends this authenticator and the ticket it received from the KDC to the service it wishes to access.

The service will perform a check on the authenticator the client sent him. If the information is verified, the server is certain the client was identified by the Kerberos server and the client was authorized by the KDC. And thus, our client is allowed access.

In summary, Kerberos provides the tools of authentication and strong cryptography over the network to help secure an information system across an entire enterprise.

4.2.7 - Secure Socket Layer / Transport Layer Security

SSL [56] was first developed by Netscape for sending encrypted information over the World Wide Web in order to secure financial transactions. TLS [57] is the IETF standardized version of SSL. The TLS Working Group was established in 1996 to standardize a transport layer security protocol based in SSL version 3.0, and in 1999 the protocol was specified in RFC 2246 [41].

SSL enables two parties to establish a secure connection over an insecure channel. Any normal TCP based protocol can be used over an encrypted SSL channel.

SSL is a layer between the raw TCP/IP protocol suite and the application layer [5]. While the standard TCP/IP protocol simply sends an unauthenticated stream of information between two computers (or two processes running on the same machine), SSL adds numerous features to that stream, including:

- Authentication of the server, using digital signatures.
- Authentication of the client, using digital signatures.
- Data confidentiality through the use of encryption.
- Data integrity through the use of message authentication codes.

Client and servers negotiate a common secret called “master secret”. All other key data is taken from this secret and passed through a carefully designed Pseudo Random Function.

SSL is an extensible and adaptive protocol, which enables the protocol to adapt in the fast moving field of cryptography, and to incorporate new cryptographic protocols (because these will not work unless both parties of the communication use the same algorithm).

Unlike to Kerberos that provides a number of security functions through an interface, the SSL/TLS protocol is a secure transport mechanism that uses network sockets directly as its interface. SSL is set by the above layers and then automatically enforces security on all traffic that goes through a particular socket.

The integration of SSL/TLS into the CORBA security architecture is done by the concept of pluggable protocols. A new SSLIOP transport mechanism was introduced as a secure alternative to IIOP. The security architecture is responsible for the set up of an SSL connection and then uses SSLIOP to communicate on top of it, thus replacing the transport layer of the CORBA architecture.

4.3 - Selection Criteria

After presenting the relevant technology, we will discuss the selection criteria we adopted. They were defined early on and they can be quickly summarized in a list, ordered by decrease of importance:

- Integration with existing work: this is clearly the most important criterion. We have an installed base, a working system that already implemented the MBE Architecture, which we have to reuse. It was outside the scope and objectives of this work to introduce significant changes to the architecture or the implementation.
- Availability of implementation: the technology chosen must be implemented, tested and supported by third parties. This is important because we opted not to pursue the difficult task of in-house development of an implementation of a desirable technology.
- Openness of the technology: to follow the objectives present in the MBE system design, whenever possible we should use standard technologies (like CORBA), and when choosing implementations free software packages should be preferred over proprietary ones.

Due to the limitations imposed by an already implemented and tested MBE architecture we quickly narrowed down our choices to two options, and as expected were both CORBA based:

- CORBA Security Service
- Common Secure Interoperability version 2

Our first decision was to base our work on the CORBA Security Service. This seemed the most logical and straightforward way of adding security and authentication to MBE.

In fact we designed a large part of the system around CORBASec (this design will be thoroughly explained in the next chapter) and with some success.

But then the initial assumptions have changed. The software package that implemented the CORBA Security Service on top of the ORB used in the MBE

architecture (ORBacus) was discontinued by the company that developed it. As this was proprietary software, it meant that the package was unsupported from that moment onward, and therefore we had to reverse part of the design process.

The same company that discontinued the CS software introduced a new product that was an implementation of the OMG defined CSiv2. Because there was a rather restrict availability of ORBacus security-related support, we were almost forced to choose this package.

The constraints found during this selection process, especially related to the first two criteria we choose Common Secure Interoperability version 2 as the technological base of our work.

Chapter 5 - Analysis and Design

Following the identification of threats to the system and selection of the technology to be used (CSIV2) we come to the subsequent stage: analysis and design.

We will begin by outlining the basic design decisions concerning the access control model and identifying the main objectives of our system.

Next we will discuss the ability of CORBA Security Service to support the Core RBAC reference model.

Then, we show that the CSIV2 architecture, along with the ATLAS service, can support the RBAC model, and proceed to discuss with in detail the main elements of the system design.

Finally we will describe the role of definition process, which is a very important step of the design of an RBAC system.

This chapter gives an overview of the major decisions taken when designing the system.

In the design process, we first studied a solution based on CORBA Security, as this seemed the “natural” solution to our problem.

We will show the adequacy of an RBAC model design based in CORBA Security technology.

As explained in the last chapter, several constraints, especially the inexistence of a software package that implemented CORBA Security, led to the dismissal of the first solution and to the selection of CSiv2 to design and implement a secure distributed environment.

Although there are some problems with this choice, mainly basing our system on non-standard interfaces and services, we opted for a pragmatic approach, in which we tried to reuse what we had investigated and designed and adapt it to a new environment, with the goal of fulfilling our objectives and prove the validity of the concepts.

We will begin by describing the major decisions and approaches followed in the analysis and design stage, which are valid for the two technologies that were referred to: CORBASec and CSiv2.

5.1 - Basic Design

Even if we had a constrained choice on the technological level, there were also other (almost) independent choices we had to make.

Among those, the most important on a high level design was the choice of an access control model. In fact authorization is central to the security problem of our system.

The model we chose to implement was RBAC, due to several reasons:

- Stable nature of roles: in a television studio roles within the organization are clearly defined and permissions associated with those roles change infrequently. On the other hand, the assignment of users to a particular task (role) within a studio can happen regularly.

- Ease of administration: the main administrative action, after the RBAC framework is established in a system, consists in adding or removing users from the various roles. In fact the major purpose of this model is to facilitate access control administration, because it avoids attempts to directly control low-level mechanisms.
- Well adapted to a distributed system: it enables analysis and design at the enterprise or organization level.

Of the four different RBAC reference models, only the most basic - Core RBAC - was selected, because it defines the minimum set of conditions to achieve a role based access control system.

It was decided to leave the other reference models to a later stage, if and when they became necessary.

The MBE architecture has two important areas that must be secured in order to have a globally secure system, which brings additional problems to our design. First we have the bulk of the platform that defines the business rules and manages metadata, which is implemented in Java programming language. Then there is the essence handling module, DIMICC, which is implemented in C++ and utilizes a different ORB (the specific software packages that are used in the implementation of the system are described in the beginning of the next chapter).

Our design will have to address these characteristics, and thus has the following objectives:

- Implement a role based access control system in the MBE platform, based on CORBA Common Secure Interoperability version 2 architecture.
- Secure lower level connections that handle essence transfers. This will be integrated in the DIMICC module.

The first objective can be clearly included in the business rules of the system (definition of roles and permissions), and thus must be integrated into the Business Objects layer of the MBE architecture.

To implement the authorization logic we chose to use the Resource Access Decision (RAD) architecture [29], which was recently the target of an OMG specification [45].

The design can be perceived as an application authorization service, which will be responsible for guaranteeing the access control on the system. It is a simple design that introduces minimal modifications to the MBE architecture.

We will use RAD Service to obtain centralized AC enforcement on the MBE distributed architecture. It is an approach that separates authorization from application logic for distributed systems, when AC is used at the application-level.

The main property of this architecture - decoupling of authorization and application logic - is achieved because applications (or in our case CORBA objects) delegate authorization decisions to a RAD server.

The architecture can support any level of resource protection granularity (in CORBA, the granularity of AC mechanisms is at the level of operations on objects) because of the generic data structure representing a resource name.

Resource name is an abstraction which is used by the applications for referring to the protected entities, and hides application dependent semantics and syntax. A resource name can be associated with any valuable asset or object, of an application owner, to which access must be controlled. The way to associate a resource name to the protected entity varies from application to application, and there is not a general algorithm to perform such an association. The resource name, as well as operations names and client (principal) security attributes, are represented in simple data structures.

The authorization logic of the system is encapsulated into an authorization service (RAD server) external to applications. An application, in order to perform AC, requires an authorization decision from the RAD server, and then enforces that decision. The RAD server needs some data before making a decision: identification of the resource name and the access operation name in association with the client's identification.

To guarantee easier administration and centralization of the AC rules, the decisions are obtained from a single RAD.

Figure 5-1 shows a simplified representation of the interactions between client, target object (which we use to represent the application) and RAD service.

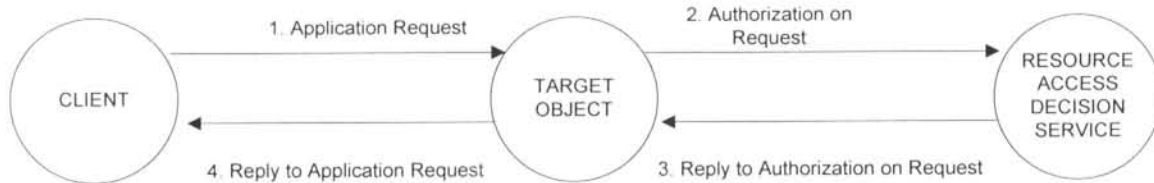


Figure 5-1: Interactions between Client, Object and RAD Service (source [29])

This sequence of interactions has the following steps:

1. Client invokes an operation on the target object.
2. Target object requests an authorization decision from the RAD on the request made by the client.
3. RAD makes an authorization decision, based on the AC policy, and returns it to the target object.
4. Target Object enforces the authorization decision. If access was granted, the Target Object returns the results of the client invocation. Otherwise, it either raises an exception, or returns just partial results.

The RAD architecture is simple (there are simple interfaces between application and authorization service and all required information is represented in simple structures), generic (resources are abstracted by names, can be used in various domains), policy neutral (can be used to implement RBAC policies) and flexible.

The RAD architecture allows externalization of a reference monitor from the application.

We can characterize our access control design approach as being based on authorization server. In fact, the best way to implement a reference monitor is making it external to the application system, guaranteeing that authorization decisions are made and enforced before an application gains control. This approach has the great advantage of not requiring many changes to an already existing large system.

We now come to the second objective our design will have to address: securing lower level connections that handle essence transfers.

The current implementation of the DIMICC module, XDIMICC only supports TCP/IP based essence connections. Therefore, we chose SSL to establish secure connections, because any normal TCP based protocol can be used over an encrypted SSL channel.

This will provide transport message protection, and provide the necessary protection against eavesdropping and tampering.

These choices represent the general features of the target system and were the starting point of the system design process. In the next sections we will discuss the specifics of the design.

5.2 - RBAC based in CORBA Security

The first solution we designed was a RBAC system based in CORBA Security.

CORBA access control mechanisms have great expressiveness. They were analyzed and shown to support mandatory access control [30]. Later, it was also shown that it is possible to configure CORBA access control mechanisms to support RBAC models [31].

In our work we intend to implement only the Core RBAC reference model, the most basic one. It requires only that a system has notions of users, roles, permissions and sessions and no constraints are imposed on the assignment of permissions to roles and users to roles.

In order to show that is possible to support the Core RBAC reference model in a CORBA Security based system, we have to describe the CORBA access control model in more detail.

CS uses the notion of principal, which completely represents the user in the model.. From the model point of view, a principal (represented by a Credential object after authentication has occurred) is an unordered collection of authenticated (and typed) security attributes.

The only function a principal has within the model is to invoke operations on objects. For every operation defined in the object interface there is a set of required rights that allow the principal to invoke it. This can be modeled as a Required Rights Matrix (RRM), with a row for every interface operation and with two columns: the first specifies the set or required rights and the second a combinator (all or any).

Operations	Required Rights	Combinator
op1	R1	all
op2	r2, r3	any
op3	R1, r2, r3	all

Table 5-1: Required Rights Matrix

Table 5-1 shows an example of RRM. Each operation will be uniquely identified by the interface it is defined on. We can interpret the third row of the table to have the following meaning: Operation op2 can be invoked by any principal who is granted either right r2 or r3. The last row specifies that the principal must be granted both r1, r2, and r3 rights to be able to invoke the operation op3.

For each CS domain (which is just a collection of objects), the principal is granted different rights according to the attributes it possesses. In fact, we can model this as a Granted Rights Matrix (GRM), with a row for every attribute and a column for every domain. This matrix allows the definition of the access control policy in each domain.

Table 5-2 shows an example of GRM, and the mappings between principal attributes and granted rights.

Attributes	Domains	
	d1	d2
a1	r1	r1,r2
a2	-	r3

Table 5-2: Granted Rights Matrix

RRM and GRM define access permissions. When a principal wants to invoke an operation on an object (that is, the implementation of some interface), the CS mechanisms check in the GRM which rights are granted to that principal on the

domain that the object belongs to. Then it tries to match those rights with the ones represented in the RRM, which defines the required rights for operation invocation to succeed. If the match happens then access is granted, otherwise invocation fails because access to the object is denied.

As we have seen before, the Core RBAC model defines a set of basic data elements: users, roles, sessions, objects, operations and permissions. There is also the concept of role relations: user assignment (UA) and permission assignment (PA) relations.

To show the adequacy of CORBA security mechanisms to support RBAC models we will map elements in CS model to elements in RBAC model.

To support the Core RBAC reference model we have the following identities [31]:

- Users in RBAC are mapped to users (represented by principals) in CS.
- Roles are represented by a set of privilege attributes.
- Permissions are equivalent to a set of rights.
- Sessions are equivalent to principals (set of security attributes).
- Operations are equivalent to operations in CORBA interfaces.
- Objects are equivalent to CORBA objects or domains (group of objects), which allows easier administration.
- PA relation is specified by the GRM.
- UA relation is managed by user administrator in CS; this management functionality is not provided by CS specification, and therefore will be implementation specific. The user to role assignment relation defines which set of attributes (which define a role) is assigned to a user.

So we have determined that for our system to support Core RBAC it has to be based on a CS implementation that complies with the standard and externally provides a way to administer the UA relation.

Since we did not have a CORBA security service implementation that ran on our ORB, we were forced to give up this option.

Fortunately we found another way to solve our problem, which we describe in the next section.

5.3 - RBAC Based in CSiv2

Several constraints led to the dismissal of the first solution and to the selection of CSiv2 to design and implement a secure distributed environment.

This proved to be a more difficult task, because CSiv2 is a newer technology with less penetration in the market than CORBA Security.

This design has some problems, which will be outlined in the discussion.

One that we have already identified is that the system will be based on non-standard interfaces. This will be explained with more detail in the next chapter.

The first thing we have to determine is if the CSiv2 architecture, along with the ATLAS service, can in fact support the RBAC reference models, and more specifically, the Core RBAC model.

As we have seen in the previous chapter, CSiv2 defines the SAS protocol that provides these two services, which will be important to our design:

- Identity assertion: allows a client to claim making a CORBA request on behalf of some identity other than itself.
- Authorization service: transfers a client's authorization data to a target, i.e., enables a client to push security attributes to a target where they may be applied in access control decisions.

Similar to what we have done with CORBA Security, we will try to map the elements in CSiv2 model to elements in the RBAC model. Some of the mapping will be provided by features outside the scope of the CSiv2 specification and, therefore will be implementation specific. In fact, unlike CS, the CSiv2 specification does not define an API, which will lead to more implementation specific solutions.

To support Core RBAC reference model we have the following identities in our design:

- Users in RBAC are mapped to users in the system
- Roles are represented by security attribute of type role. In fact roles are just names (strings) on role certificates.
- Permissions are represented by a set of rights. These rights are defined in RAD's Permission Matrix.
- Sessions are represented by principals.
- Operations are equivalent to operations in CORBA interfaces.
- Objects are equivalent to CORBA objects. We could also define ATLAS authorization domains that grouped several objects with similar access restrictions, in order to enable easier administration. This option would cause us to use one ATLAS service for each domain.
- Permission Assignment relation is defined by the RAD service, which encapsulates authorization logic. This is a feature external to the CSIv2 architecture.
- User Assignment relation is managed by the Role Authority (RA) service. An ATLAS service is used to deliver certificates. This management functionality is not provided by CSIv2 specification, and therefore will be implementation specific.

We have determined that our system can support Core RBAC. Next we will discuss the details of the design of our system.

First there is the authentication problem. How will the system verify the users' identity?

We will authenticate the users using a system based on the Kerberos protocol. This will allow users to login in single point and be authenticated throughout the whole MBE system. After authentication, a user will be completely represented by a principal within the system.

The network administrator will be responsible for user management in the Kerberos server.

Second, we have to describe the RA service. This service is responsible for managing the assignment of roles to authenticated users.

RA will issue a role certificate if the principal is allowed to activate a specific role. Roles are granted to principals using role certificates.

In our design, a single ATLAS service will be responsible for delivering role certificates to principals, and therefore can be interpreted as custom made authorization tokens.

A role certificate is a digitally signed statement that binds a role to a user [32].

These role statements can be represented as simple data structure of pairs <type, value>. The structures can have a different number of fields, among which we can emphasize:

- User: this field contains the name of the user to which the role is granted.
- Role: this field contains the name of the role activated. In fact, roles in the system are represented by just names, and will also be used in authorization decisions by the RAD service. We can interpret names as security attributes of the Principal.
- Validity: time period during which the certificate is valid.

This is a simple and easily extendible mechanism; more fields can be defined, if their existence is deemed necessary. Table 5-3 shows an example of elements of a role authorization statement.

Type	Value
User	Charles S.
Role	Reporter
Validity	3 hours
...	...

Table 5-3: Elements of a Role Authorization Statement

Third, we have the RAD service.

The RAD service encapsulates authorization logic, defining which system resources a role is permitted to access.

To make security decisions, the RAD service needs, as already mentioned, identification of the resource name, the access operation name and the principal's identification. We need to associate resource names with the protected entity.

In our system the resource names are just CORBA objects and operation names are simply operations within CORBA interfaces.

The principal is identified by the role he has activated. Principals use CSIv2's identity assertion service to activate a role.

The RAD service is responsible for PA relation. In our system, we want to protect CORBA objects, so for every operation defined in the object's interface the RAD will have to decide if the principal is allowed to invoke it. The best way this can be modeled is through a Permission Matrix (PM), with a row for each role and a column for every interface operation. If an operation is not represented on the matrix, the decision is to deny the request.

Roles	Resources (operations)		
	op1	op2	op3
r1	Allow	allow	deny
r2	Deny	allow	allow
r3	Deny	deny	allow

Table 5-4: RAD Service's Permission Matrix

Table 5-4 shows an example of a permission matrix, and the mappings between roles and interface operations. The access decision depends solely on the role the user has activated.

The RA and RAD are in fact interfaces, where operations can be invoked.

They were described separately but can be implemented in the same object. As both use role names in their internal logic, this approach would ease administration of the system. We could design an Authorization Manager that encapsulates all the logic necessary to implement a RBAC system

Now that we have seen the main elements of the design let us describe how the access control process takes place.

After the authentication, the user is represented in the system by a Principal, which we can also interpret as a client object acting on the Principal's behalf. The Principal has a reference to a Target Object where he wishes to invoke an operation. The CSIv2 part of the IOR of the Target Object contains a reference to the ATLAS that define the Target Object's privilege scope.

From this point on, the sequence of interactions, is shown on Figure 5-2, and has the following steps:

1. The Principal contacts the ATLAS service and asserts it is in a role, using identity assertion service from SAS protocol.
2. ATLAS gets a role certificate from RA. If the Principal does not have permissions to assume that role, RA will not issue a certificate.
3. ATLAS delivers role certificate to the Principal.
4. The Principal invokes an operation on the Target Object, using normal CORBA mechanisms. It also delivers the role certificate to the Target Object, using authorization service of CSIv2 SAS protocol.
5. The Target Object requires authorization decision from the RAD. Target Object delivers client role certificate and operation information to the RAD, using SAS' authorization service. The message (a request to invoke an operation on the target object) itself used in the access control enforcement.

- 6. The RAD server makes authorization decision, after consulting the Permission Matrix, and replies to the Target Object.
- 7. The Target Object enforces the authorization decision. If access was granted, the Target Object returns the results of the client invocation. Otherwise, it will raise an exception.

All these interactions must be done over a secure association, provided by the CSIV2 architecture. Security association is the shared security state information, which permits secure communication between two entities [64]. The CORBA Security Context object encapsulates the shared state information representing a security association and also exists in the non-standard API defined by the software package that implements CSIV2.

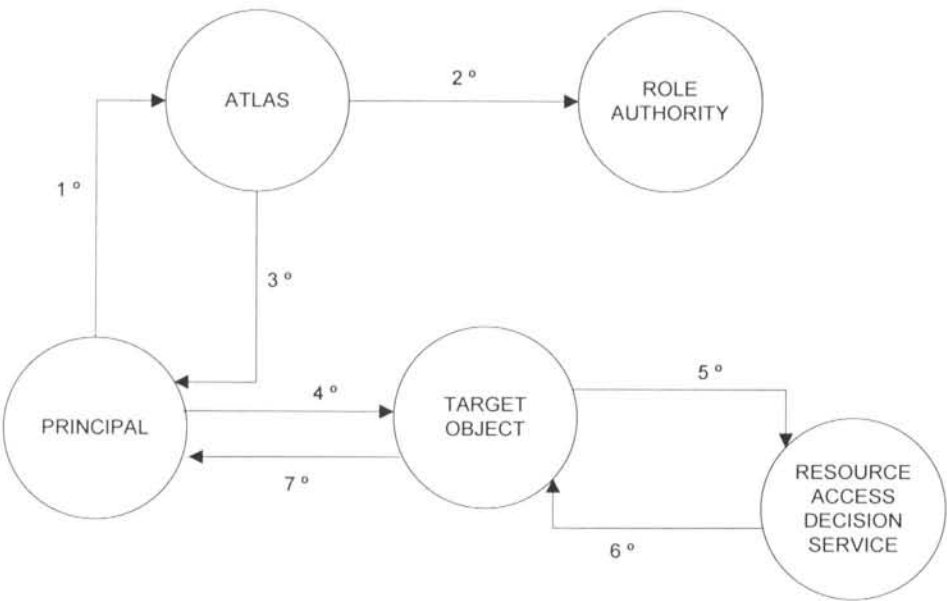


Figure 5-2: Interactions between Objects and Services

There are some problems with this solution.

The first one has to do with role activation: the principal must remain in the same role throughout an established security association with the target object. To use a different role, the principal has to establish a new security association.

RA issuing different role certificates, one for each role, can enable a principal to achieve simultaneous activation of roles.

Then, our design is not as reusable as we initially wished, because it has many implementation specific parts that provide features outside the scope of the CSIV2 specification.

Third, we will have to carefully design our Authorization Manager (RA + RAD) in order to avoid making it a performance bottleneck on the whole system. It is also important to achieve a reliable implementation of the manager, because in case of failure of the authorization server the system becomes unoperational, with all requests being denied by default.

5.4 - Role Definition Process

A very important step on the design of an RBAC system is the process of defining the suitable roles to be represented. Roles can represent competency, authority, responsibility or specific duty assignments.

The process of defining roles should be based on a thorough analysis of how an organization operates and should include input from a wide spectrum of users in an organization.

In the case of the broadcasting world, the content creation process is known and has been clearly described. Figure 5-3 represents this process.

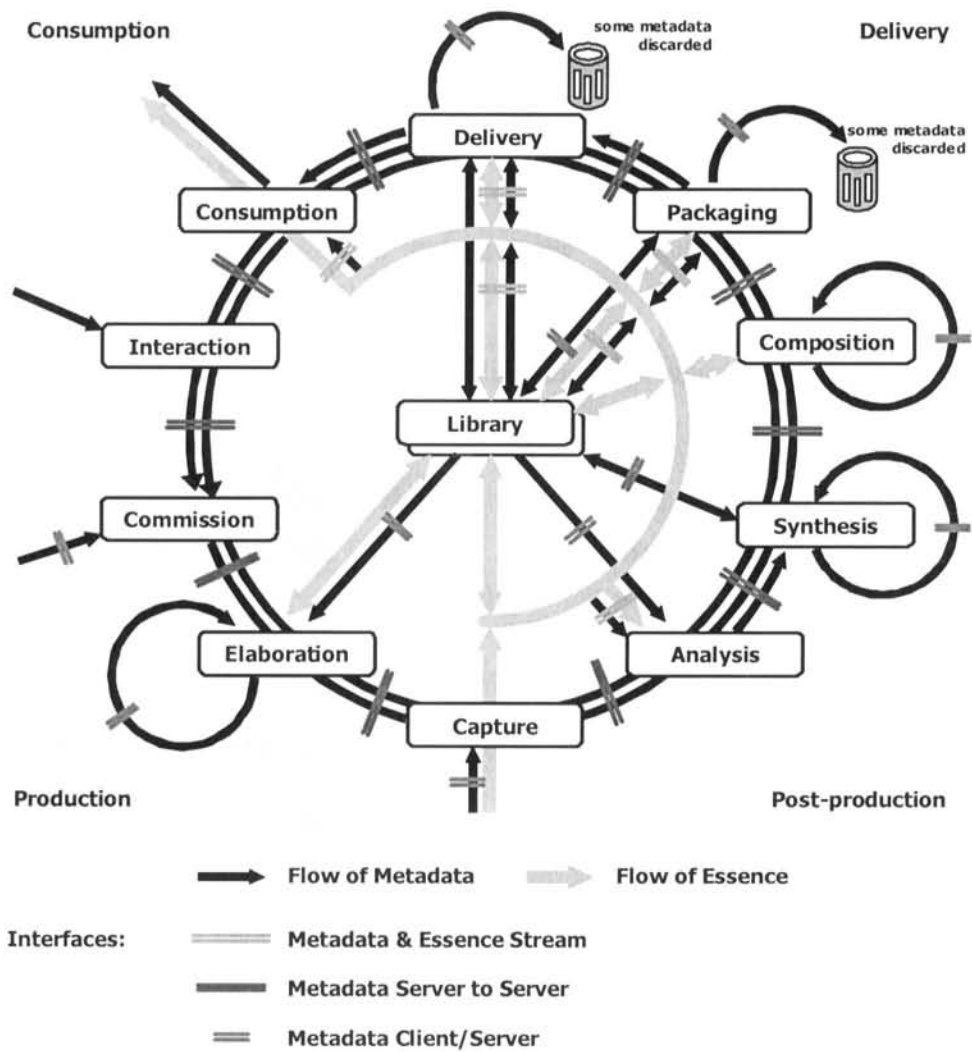


Figure 5-3: Content Creation Process (source [43])

The most important steps of the content creation process [43] are:

- Commission: the commissioning of the content by the producer begins the content creation cycle.
- Elaboration: an iterative process that defines the concept of the content, and is performed during pre-production.
- Capture: during this stage, the raw essence is brought into the content database. Capture includes shooting, graphic origination, archive research and acquisition. Note that Content may exist prior to either Commission or Elaboration.
- Analysis: includes the processes of feature extraction, motion analysis, and content analysis of the raw essence.
- Synthesis: is an iterative process in which extracted descriptions are merged with each other to synthesize useful descriptions of the raw essence.
- Composition: it is the first phase of editing or cutting down the raw essence to form the intended content, and uses the metadata derived from all earlier phases.
- Packaging: it is the phase of building the final version of the content to be distributed.
- Delivery: this phase includes the transfer of content from the production form to the delivery vehicle – content server, web server, packaged media, or broadcast.
- Consumption: it is the phase of viewing or receiving the content.
- Interaction: it is the phase in which the consumer interacts with the content.
- Library: library functions include deep Archive (also known as long term storage), retrieval (with an emphasis on cataloguing Metadata) and Library access (where the emphasis is on search and retrieval).

Some of these steps are external to the work done on the Television studio, and therefore are outside the scope of the MBE architecture.

Several applications (GUI's) were developed during the ORBIT project that corresponded to the needs of studio personnel, in order to demonstrate the suitability

of the MBE platform to broadcasting environments and to serve as examples to future development of applications.

The GUI's can be seen as MBE workstations, where a user sitting at a PC can control the activities associated with that GUI. They can be mapped to the several steps of the content creation process.

Figure 5-4 [66] represents the mapping between the content creation process and:

- the processes (GUI's) in MBE, in light green;
- external processes to each MBE may integrate with.

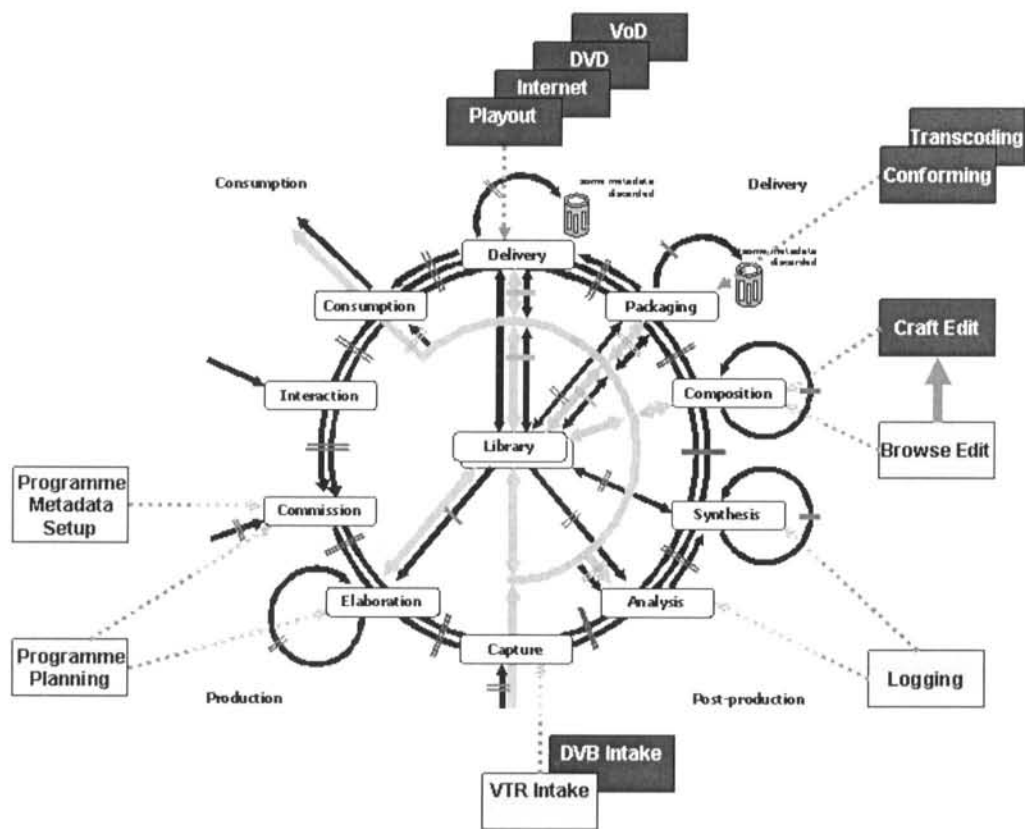


Figure 5-4: Content Creation Process in MBE (source [66])

At the Programme Metadata Setup stage, a new Programme is created in MBE: resources are allocated to the production of the new programme, descriptive fields are annotated and default data fields may be set.

Then, at the Programme Planning process (which includes elements of the elaboration and commission stage), a member of a team producing a Programme will be able to specify details on the programme such as: the scenes that compose the programme, the scheduling of material capture and the preparation of editing sessions.

VTR Intake GUI allows the control of a VTR, an encoder board, a transcoder (to generate lower quality versions of the essence) and an Essence Server to allow the capture of material in a tape.

Logging is the process of documenting the captured essence using different kinds of mechanisms like automatic shot detection and manual segment identification and documentation

At the Browse Edit stage the operator may build an initial version of an edit decision list for a programme or part of a programme. This application allows a rough edition of the programme that can then be exported to a Craft Editor for finer treatment. The operator uses browse quality essence at this point, but all the timecode marks are valid when applied to full quality essence.

There is also a Search GUI since we have to store and manage huge amounts of information. This is the process of specifying filtering conditions to obtain available material to be used in new programmes, or parts of a programme. It is not shown in the diagram above.

All these processes and applications represent an approximation of the tasks involved in a television studio.

We recommend that these tasks be reflected in roles in the access control mechanism of the MBE system.

Many times these roles will represent only a part of the assignments and duties a user inside the Television studio. In this case a user will be granted membership to several roles, in order to fulfill all of its obligations.

This fact will have two great advantages:

- Roles will not be too general, and therefore roles will not be granted more access to resources than is needed - Least privilege principle.

- It enables a more flexible duty assignment and variety of jobs inside the system. Users will not have to be adapted to predefined and rigid jobs, but can have different responsibilities reflected through different role membership.

Of course the role definition will have to be adapted to the specifics for the broadcasting organization where the MBE system is deployed.

For instance, on a news organization a Reporter could be granted membership to the following roles:

- Search: to allow him to search background material and even archive footage if needed.
- Elaboration: to allow him to specify details on the news story. It should be noticed that the commissioning of a new programme (resource allocation) would be done by the News or Information Director, and not by the reporter himself.
- Browse edition: this is part of composition process, and would be needed if the reporter has to do the initial rough edition of his own news piece. In any case, the final edition (Craft Edition) should be done by a specialized Editor with a different role within the system.

5.5 - Concluding Remarks

The analysis and design stage found several problems and constraints, the main being the inexistence of an implementation of the CORBA Security Service for the ORB used in the MBE architecture.

As such, we had to find other technologies to enable adding security and authorization to MBE. We found that technology on CSIV2, standardized by OMG.

As this technology is still in its infancy (i.e., it is not widely adopted and deployed), we had to demonstrate the adequacy of CSIV2, in conjunction with ATLAS service, to support the RBAC access control model, and more specifically the basic Core RBAC reference model.

Next we proceeded to design a secure system, using the concept of RAD service, to decouple authorization from application logic and enable the centralization of AC decisions for better system administration.

We finished the chapter with a description of the very important process of role definition that is the basis of the implementation of an RBAC system in an enterprise, which must be adapted to the specifics of the broadcasting organization where a MBE system is deployed.

In the next chapter we will focus on delineating a strategy for implementing the system we designed on the existing MBE architecture. The implementation stage was not included in this work due to time restrictions, technological constraints and availability of resources within the project and the partners.

Chapter 6 – Implementation Strategy

In the previous chapter we proposed a system architecture that integrates Authorization and Security mechanisms in the MBE distributed digital asset management platform.

Now we will delineate an implementation strategy for that system.

This chapter starts by describing the software packages that were used in the MBE implementation, as well as the ones that will be purposely introduced to support security in our system.

Then we will recommend some approaches to be used in the implementation of the different modules identified in the previous chapter, as well as a discussion concerning the securing of essence connections.

6.1 - Used Tools

We will describe several important software packages that were used in the MBE architecture, as well as the ones that will be used in our improvements and extensions to the system.

Whenever possible, Free Software [47] / Open Source [48] packages were selected due to their inherent advantages. They provide an open and evolutionary development process that allows us to participate as equals.

This is very important because it assures that we will not get locked-in with end-of-the-line unsupported software (see ORBASec SL2 case): we can always improve it ourselves, and certainly will be joined by many through out the Internet.

6.1.1 - ACE

One of the requirements for the implementation of the MBE architecture was that it should be portable; specifically, it should run on Windows and GNU/Linux machines.

After careful search and evaluation of frameworks that provided a portable API and shielded the code from the differences of platforms, the ADAPTATIVE Communication Environment (ACE) [33] [60] was adopted to build the DIMICC module.

ACE is a freely available, open-source object-oriented (OO) framework that implements many core patterns for concurrent communication software. ACE provides a rich set of reusable C++ wrapper facades and framework components that perform common communication software tasks across a range of OS platforms.

ACE was initially developed by Douglas Schmidt at The Distributed Object Computing group at Washington University, in St. Louis, Missouri. Today ACE continues to improve with the contributions of hundreds of users and programmers, including some patches by INESC Porto. ACE is supported commercially via the Riverace Corporation using an open-source business model.

The ACE platform provides several important mechanisms, which include event demultiplexing and event handler dispatching, signal handling, service initialization, interprocess communication, shared memory management, message routing, dynamic

(re)configuration of distributed services, concurrent execution and synchronization. Figure 6-1 shows ACE's class hierarchy.

The use of ACE has many associated benefits, which were important to the success of the implementation. Besides an increased portability, it also increased software quality, reliability and efficiency of the architecture as a whole.

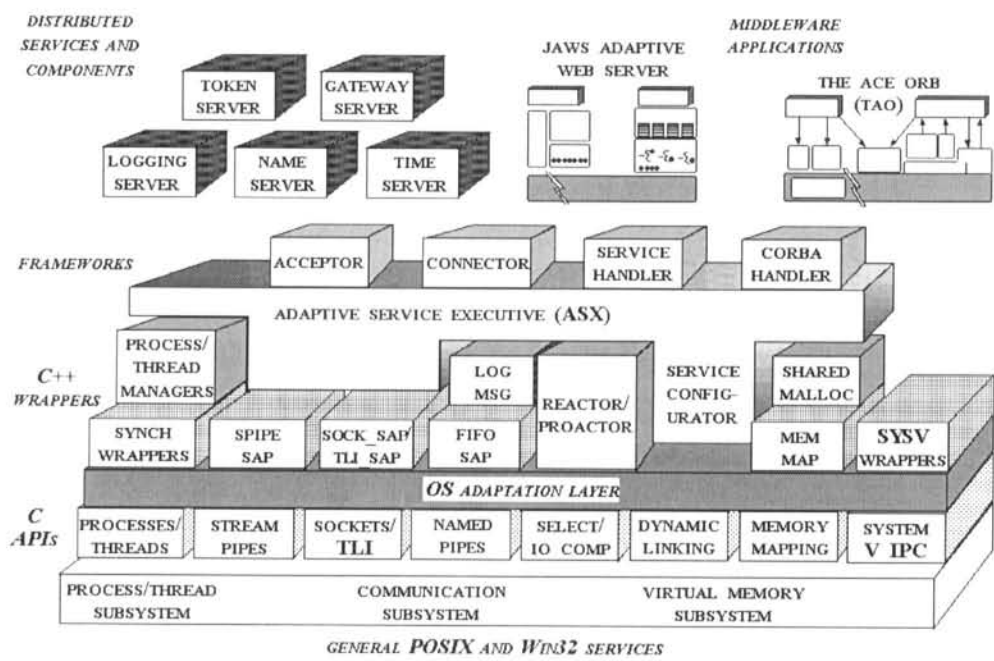


Figure 6-1: ACE Class Hierarchy (source: [60])

6.1.2 - TAO

The ACE ORB, or TAO [61], is an ORB built on top of ACE, compliant with most of the features and services defined in CORBA 3.0 specification. It is used as the ORB of the DIMICC module in the MBE architecture.

TAO is a freely available, open-source, and standards-compliant real-time implementation of CORBA that provides efficient, predictable, and scalable end-to-end quality of service (QoS). TAO comes with many CORBA services, like Naming Service, Notification Service, Property Service, Time Service, Trading Service and a partially implemented Security Service. Figure 6-2 shows TAO's block diagram.

TAO started as a research project at the Washington University, but as a successful open source project, was improved by a large community of contributors, and this development model is considered to be responsible for its success. TAO is supported commercially by OCI and PrismTechnologies.

The group responsible for the design of TAO was deeply involved with OMG work on specifying real-time ORB requirements and specifications. TAO applies the best software practices and patterns to automate the delivery of high-performance and real-time QoS to distributed applications.

TAO was chosen because of its close relation with ACE and also its real-time capabilities, which are very important in QoS stringent environments such as those in the television world.

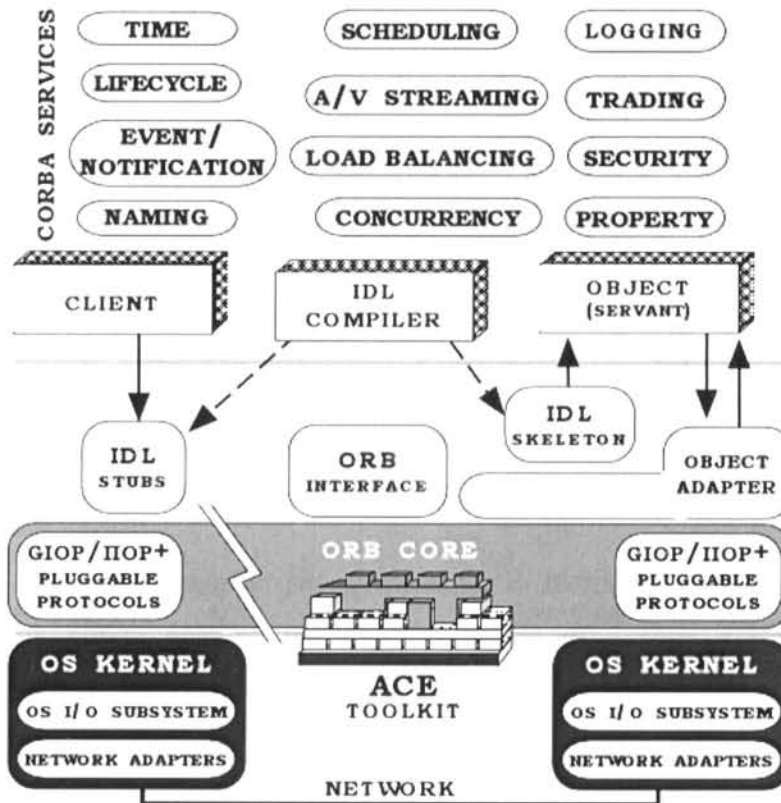


Figure 6-2: TAO Block Diagram (source: [61])

6.1.3 - ORBacus

ORBacus [62] is a multi platform, fully CORBA 2.4 compliant ORB distributed by IONA Technologies.

This is the ORB used in most of MBE architecture, due to its good Java language mapping. ORBacus is available in both Java and C++ version, which are totally interoperable with each other as well as with other products that use IIOP.

ORBacus has excellent support for multithreading and has a small memory footprint. It also provides support for portable interceptors, interface and implementation repositories as well as the pluggable protocol framework that allows the use of various network protocols.

One very important characteristic of this ORB is its simple configuration and bootstrapping, which enabled the implementation of a semi automated network spanning mechanism for the MBE architecture.

ORBacus also comes with the implementation of several standard CORBA services: Naming Service, Event Service, Property Service, Notification Service, Trading Service and Time Service.

It is distributed with source-code, but it is not an open source product. In fact, with the 4.1.x release, the license was once again changed, introducing a limited trial license to the IDL-to-C++ and IDL-to-Java translator.

6.1.4 - OpennCCM

OpenCCM [49] stands for the Open CORBA Component Model Platform, and is an open source implementation of the CORBA Component Model [37] mechanism, defined by OMG [50].

In fact, it was the first publicly available open source implementation of CCM. This is the main reason for its selection. When it was decided to use CCM technology in the MBE infrastructure, this was the only implementation available and we are using it since the 0.1 release.

Still, OpenCCM is under development and it does not include all expected CCM features. At present it consists of an open development tool chain (a compiler for the OMG Component Implementation Definition Language, a generator to visit the OMG IDL3 Repository, etc.), a deployment infrastructure (e.g., CCM XML descriptors.) and an open container runtime (a component server to deploy and to execute Java components with support for transactional deployment).

The Open CCM project, developed at Lille University, in France, and with support of a large community, allows us to design, implement, compile, package, assemble, deploy, and execute distributed applications compliant with the CCM.

All the software packages described earlier were already in use in the MBE architecture.

Now we will talk about the tools that will be purposely introduced to support security in our system.

6.1.5 - OpenSSL

The OpenSSL[51] project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library.

It is used by the ACE framework to provide secure sockets needed by our proposed modifications to XDIMICC essence handling module.

OpenSSL is based on the excellent SSLeay library developed by Eric A. Young and Tim J. Hudson, and is managed by a worldwide community of volunteers who use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation.

It is a cross-platform, portable, quality library that is highly regarded in the security world, and represents a very important contribution from the open source community, and simultaneously a validation of this kind of development process.

6.1.6 - ORBAsec SL3

Then we have the software packages whose changes significantly influenced the direction of the work. It caused us to redesign our solution and move away from the first proposal.

ORBAsec is a proprietary software product from Adiron, LLC [106]. When first surveyed, ORBAsec SL2 implemented on top of the ORBacus ORB (version 3.3.x) a compliant CORBA Security Service.

But then Adiron decided to stop the development of ORBAsec SL2 and released a totally different product, which they feel is more robust and dependable, and branded it ORBAsec SL3 [53]. This was a crucial decision, because it was the only available implementation of the CORBA Security Service that runs on the ORB already in use in the majority of the MBE architecture - ORBacus.

ORBAsec SL3 adds authentication and encryption to IONA's ORBacus ORB. ORBAsec SL3 is an implementation of the standard CORBA CSiv2 architecture with proprietary and non-standard `SecurityLevel3` and `TransportSecurity` interfaces. This is because CSiv2 specification does not define an API, unlike the CS protocol, which forced Adiron to design and develop new interfaces for the product.

ORBAsec SL3 abstracts the CORBA CSiv2 protocol using `Credentials` interfaces, which represent the principal completely within the system, as was the case with CORBA Security. This concise set of interfaces allows you to control the complexities of the CSiv2 protocol.

ORBAsec SL3 supports Kerberos, and the Kerberos User-to-User protocol, which allows single sign-on capability to CORBA clients and servers, as well as Windows 2000 authentication servers.

ORBAsec also has support for the use of the ATLAS, a service that allows to plug-in personalized authorization tokens into the CSiv2 protocol.

As a whole, ORBAsec SL3 is quite different from its predecessor, ORBAsec SL2, in that it departs from the CORBA Security Level 2 API. It gives a brand new API to Distributed Security, which is based on a mathematical model of principals [34],

because it was asserted that CORBA security level 2 API was insufficient for enabling the use of CSiv2.

This poses several hindering problems to our work.

First, this product has non-standard interfaces. Unlike CS that was (and still is) standardized by OMG after a long process of peer-review (which is the right way to approach a security process), ORBAsec SL3 presents with a new and still unproven set of interfaces that were developed mostly internally by Adiron . This prevents us from delivering a solution based on open products and standard technologies.

Second, but closely related, this software has many external dependencies. These packages are also proprietary, and in most cases do not even allow free non-commercial use.

6.2 - Implementation Approaches

In the previous chapter we proposed a solution to implement RBAC on the MBE architecture. It uses a RAD Service to obtain centralized AC enforcement, and it is based on the CSIV2 architecture.

In this section, we will recommend ways of implementing the different modules and techniques that were identified on the analysis and design stage.

Our implementation strategy will be based on the ORBAsec SL3 software package, which was described in the previous section. Its interfaces, albeit non-standard, present a rich variety of functionalities that can help the implementation of our solution.

First we have authentication.

The proposed authentication system is based on the Kerberos protocol. As we have seen before, ORBAsec SL3 supports Kerberos.

We have to install an external Kerberos server that contains records of all clients [55], or integrate with Windows 2000 authentication servers using Kerberos [35], to manage distributed authentication. ORBAsec SL3 also has the ability to authenticate using a Username/Password scheme. A human network manager will be responsible for administering the server.

As we have explained before a principal is represented by a Credential object after authentication has occurred within the security model implemented by ORBAsec SL3. To acquire this Credential object, there is a long procedure that must be followed:

1. Get reference to `SecurityManager` (SL3 service object that is retrieved from ORB using `resolve_initial_reference()` call).
2. Resolve `CredentialsCurator` object from the `SecurityManager`. The `CredentialsCurator` is an object that creates credentials and maintains a list of active credentials for the ORB.
3. Use `ArgumentFactory` interface to build the acquisition argument that is passed to `CredentialsCurator`, through the use of `ArgBuilder`

objects. This interface was created to assist in the task of acquiring credentials, because there are many parameters and options that can come into play.

4. Finally, the credentials are stored in a `OwnCredentials` object. This is in fact the representation of the principal within the model.

At the end of this process, the user is fully authenticated within the system, and represented by a credential object (`OwnCredentials`), which can begin making the requests.

Second, we must talk about the RAD server that will define which system resources a role is allowed to access.

The main implementation problem we should focus on is the Permission Matrix, which associates roles with interfaces. There are several items we need to address:

- The matrix will be a structure that resides in memory, to allow faster resolution of authorization decision requests by the objects.
- Storage: for handling data of the matrix we can use the XML object model. This has several advantages, the main being that XML is already used throughout MBE, and therefore there are already many modules and components to conveniently handle XML. Another thing we can do is store the contents of the XML in a database, which is the procedure used in MBE to obtain persistence. The contents of a XML document is mapped into rows of tables in the database, which were dynamically generated by analyzing the schema associated with the data. It also should be noted that we must add a refresh method to the service, to force the reload of the matrix from the database every time there is a modification.
- Representation: the roles will be just names that must be first defined in the RA server. It is important to assure that the role names coincide, because every request will be denied, if the role is not on the PM. Validity checks should be implemented between RA and RAD to ensure this condition. But, what about interface operations? We can make the dynamic discovery and enumeration process automatic, and therefore maintain an update collection of every available operation in every interface in the whole system, by using the

Interface Repository (IFR) [63], which is provided by the CORBA standard. IFR is a distributed service that allows applications to store and retrieve IDL definitions to and from an instance of the service. The IFR holds metadata about objects separately from the objects themselves, and therefore requires that, in order to use in dynamic CORBA applications, the service be populated with the IDL definitions. ORBacus comes with IFR, but currently it is not used in MBE. To discover the operations and interfaces available we can access the IFR through `ORB::resolve_initial_reference("InterfaceRepository")` method, which returns an object reference to a `CORBA::Repository`. This interface can then be used to look up and navigate through all type definitions stored in the repository. We should also assure that the RAD server is in synchronism with the system, by including a method to force the updating of the set of operations, every time an IDL interface is added, removed or modified in the system.

This RAD Server will be very important for the operation of the all system, because it will be requested very often to make authorization decisions, so we must insure its robustness, fastness and reliability.

Third, we must describe the RA server and how to implement role certificates in MBE architecture.

The RA server will be the place where the roles will be defined, and assigned to users, after a thorough analysis of the organization or enterprise.

Roles will be represented by just names stored in a table, where each role is associated with zero, one or more users, which are allowed to assert (zero, for the cases where a role was defined but nobody was yet assigned to it). Table 6-1 shows an example of such table: Commissioner is a role with no users associated, and Search role can be asserted by three users.

Role	Users
Editor	John Doe, Jane Wilson
Commissioner	
Search	John Doe, Jane Wilson, Charles Smith
...	...

Table 6-1: Example of the RA Table that Associates Roles with Users

This table can also be mapped to an XML document, which can then be stored in a table of the underlying database. It would then be loaded into memory by the server during system initialization, and each time there is a modification to the contents of the table (this could be implemented by a method that forced the reload of data).

It is important to assure that the role names are unique, so we can make them the primary key of the database table, which will be responsible to assure referential integrity.

We have also to deal with role certificates, which associate a role to a user through a digitally signed statement.

The certificates are implemented with cryptographic techniques, using public key techniques where there are two keys: one is public the other is private. Cerificates are generated by a Certificate Authority (CA), by signing a statement with its private key. Then every “client” can verify that the statement came from the CA by decrypting it with the public key. In reality, what is signed is a message digest (a short, fixed length representation of longer variable-length messages) and not the whole certificate.

In our system RA will be the certificate authority, responsible for signing role certificates.

The SSL protocol allows the creation of CA, as well as signing, requesting and using certificateds. The OpenSSL software package implements everything that is needed to implement support for certificates within MBE system.

We can use the package directly at the command line to generate initially the key pair (public/private) and to establish the CA, or do it inside the ACE framework, using the class `ACE_SSL_Context` that provides a wrapper to some OpenSSL functions.

In the design stage we advanced the possibility of an Authorization Manager that implemented both the RAD and RA services, in the same software module. This could be a viable decision that presented advantages beyond ease administration of the system.

It must be noticed that only the RAD and RA service need to understand role certificates, which are in fact personalized authorization tokens. Also, both RAD and RA process roles and thus consistency between the two must be maintained. This would be simpler if these two services were implemented in just one software component: the Authorization Manager.

The Authorization Manager can be implemented as a CORBA component, using the Java programming language, as in the majority of the MBE system. We could add a GUI to the manager, also implemented as a CORBA component, that could be run as a standalone application, as the rest of the GUIs developed on top of MBE.

6.2.1 – Securing Essence Handling Connections

To secure essence handling connections we will have to modify XDIMICC.

The objective is to add SSL/TLS support to the essence connections, to guarantee authentication of the two endpoints, essence data confidentiality and integrity.

XDIMICC is implemented on top of the ACE framework. ACE with SSL support is an addition to the core classes, which allow the use of the Secure Socket Layer with the ACE framework. Those classes are the following [60]:

- `ACE_SSL_Context` contains the interface to the SSL library. The library used is OpenSSL, which must be installed previous to utilization.
- `ACE_SSL SOCK_Stream` uses an interface and behavior nearly identical to `ACE SOCK_Stream`, only with the additional SSL layer functionality.
- `ACE_SSL SOCK_Acceptor` and `ACE_SSL SOCK_Connector` are again similar to their `ACE SOCK_Acceptor` and `ACE SOCK_Connector` counterparts.

The changes needed to add SSL support, will be made in two parts of the XDIMICC module:

- DIMICC IDL interfaces.
- DETAIL (DIMICC Essence Transfer Already Implemented Library).

To DIMMIC's IDL interfaces, we will add support for secure sockets by creating an SSL protocol module. This will allow us to have a secure socket as one of SAP that Sinks and Sources support.

DETAIL is the layer that shields users from dealing with DIMICC's connection management protocol and with the network, providing general implementations of Sources and Sinks. The DETAIL API defines an interface between programmers and

DETAIL that is common across different OSs, and is thus fundamental for code portability [15].

Figure 6-3 shows the DETAIL layer in the Essence Transfer plane protocol stack.

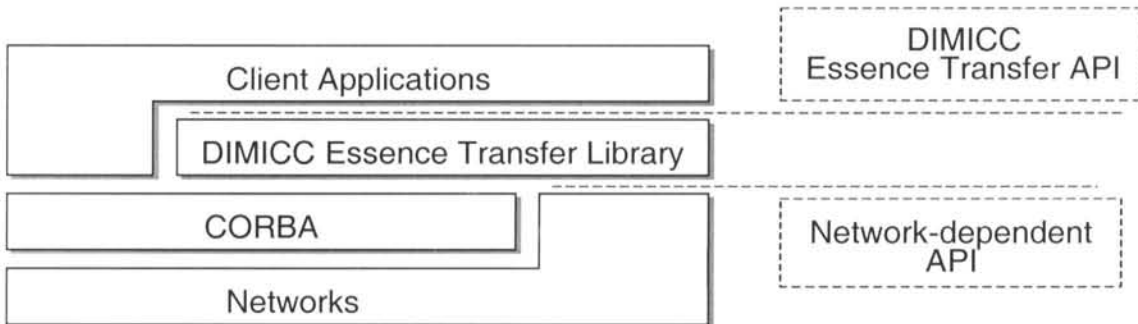


Figure 6-3: Essence Transfer Plane Protocol Stack (source [65])

XDETAIL is the current implementation of the DIMICC Essence Transfer protocol, providing a facility for the transfer of data in a flexible and location transparent fashion. It provides a means for the creation of *Source* and *Sink* endpoints that can be used to send and receive data, respectively [68].

We will also have to implement support of SSL protocol into XDETAIL, which will be very similar to the TCP protocol. We will have to replace, among others, the following classes:

- o `ACE SOCK_Stream` with `ACE_SSL SOCK_Stream`.
- o `ACE SOCK_Acceptor` with `ACE_SSL SOCK_Acceptor` in the `Source` class.
- o `ACE SOCK_Connector` with `ACE_SSL SOCK_Connector` in the `Sink` class.

This is facilitated because ACE SSL based classes' interface and behavior nearly identical to non-secure TCP ones.

The code will also have to handle certificates and cryptographic keys that are associated with a SSL connection, with the help of the `ACE_SSL_Context` class.

Chapter 7 – Conclusions

We come to end of the dissertation after discussing some implementation strategies, and hopefully it has succeed in presenting a way to secure a distributed middleware based architectures, like MBE.

This chapter starts by going over the main points of our work.

Finally it outlines what should still be done in the problem area to continue improving the system.

7.1 - Conclusions

Television world has been evolving at a fast pace in recent times, and broadcasters need to adapt to the surrounding conditions of extreme competition, new distributions media and reduced time to market.

Several solutions have been proposed that maximize the power of the available technology. We can find proposals of IT based network-distributed architecture, developed by projects like ATLANTIC and ORBIT.

Among these we have MBE distributed digital asset management system, a middleware object-oriented solution aimed at managing multimedia assets in broadcasting environments.

To protect distributed applications and their resources it is necessary to use the security provided by middleware access control mechanisms. We found that the AC model that was best suited to distributed environments was RBAC.

In this dissertation we first showed the adequacy of the CSiv2 architecture to support the Core RABC reference model. This was achieved by employing external mechanisms, not defined on the standard. This result is important, because CSiv2 is still a new technology and unproven on some fields.

Next we designed the authorization of the system based on the RAD architecture, which can be perceived as a CORBA based application authorization service. We used the Kerberos distributed authentication protocol to identify and certify users to the system.

To complete the integration of security mechanisms into MBE architecture, we had to protect the low level essence handling connections that is provided by the DIMMIC module. To respond to this vulnerability of the system we chose to secure the connections by adding the SSL protocol.

Finally we also defined the general lines of an implementation strategy, which utilizes and depends on several mechanisms available at the current MBE architecture, and minimizes changes to what already is implemented, therefore providing a good integration with the existing platform.

By defining a mapping between CSiv2 architecture and the Core RBAC reference model, and designing an application level authorization system based on RAD approach, we obtained an extensible and general authorization platform, which is central to obtaining secure distributed middleware based architectures.

Despite having focused on defining an implementation strategy, and the existence of time, technology and resource limitations, some developments were made on the proposed system architecture.

The RAD service module was partially implemented, but not integrated with the RA service. The XML document format was created and mapped into tables of the database, the deployment of IFR in MBE system was initiated, allowing to dynamically discover interface operations, and some experimental roles were defined, in order to test the RAD server. This limited implementation, and some basic tests, allowed us to confirm the suitability of parts of the proposed design.

In order to fully validate the entire architectural concept, a complete implementation of the system architecture would be needed.

The system proposed was mainly developed to meet broadcasters requirements of security and respond to the professional needs to have a familiar model, but it will also have other important influences on the overall architecture. The integration of security mechanisms will improve the management possibilities of the MBE architecture and simplify its administration.

7.2 - Future Work

There is still much to be done on the MBE architecture, and several tasks could be based on or complete the work described in this dissertation. They constitute interesting open question for future research.

First, the system designed on this dissertation could be fully implemented and integrated into MBE, if the necessary resources were allocated within the two partners of the ORBIT project.

Second, the system could be extended to support more RBAC reference models: Hierarchical RBAC, Static Separation of Duty Relations, and Dynamic Separation of Duty Relations. This would require some extensive research, because it is not a straightforward evolution and the support from CORBA is not present in some cases, requiring the implementation of external functionalities.

As one of the major advantages of RBAC is the ease of administration and management, this access control model could be used as the basis for an architecture-wide management system that would simplify the administration of MBE. This would be very important because the current implementation of MBE is not very administrator and user-friendly.

Closely related to this would be the design of a system to allocate resources. Since several physical devices (VTR's, tapes...) must be exclusively used, it is important to guarantee the allocation of resources for some important tasks, such as the Delivery phase, when a programme must be broadcasted. This system could also implement a priority mechanism, where for instance the playout area would have the highest priority, and archive related jobs the lowest.

These proposals can be important to the improvement of the MBE architecture, and could lead to an easier adoption of this new distributed digital asset management solution in the broadcasting world.

Bibliography

Books

- [1] Viega, J. and McGraw, G., *Building Secure Software: How to Avoid Security Problems the Right Way*, Addison-Wesley, 2002.
- [2] Schneier, B., *Secrets & Lies, Digital Security in a Networked World*, John Wiley & Sons, 2000.
- [3] Anderson, R., *Security Engineering, A Guide to Building Dependable Distributed Systems*, John Wiley & Sons, 2001.
- [4] Lang, U. and Schreiner, R., *Developing Secure Distributed Systems with CORBA*, Artech House, 2002.
- [5] Garfinkel, S., Spafford, G., *Web Security, Privacy & Commerce*, 2nd ed, O'Reilly, 2002
- [6] Henning, M. and Vinoski, S., *Advanced CORBA programming in C++*, Addison-Wesley, 1999.
- [7] Siegel, J., *CORBA 3, Fundamentals and Programming*, 2nd Ed, OMG Press / John Wiley & Sons, 2000.
- [8] Schmidt, D. et al., *Pattern-Oriented Software Architecture, Volume 2*, John Wiley & Sons, 2000.
- [9] Gamma, E. et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995

Papers

- [10] Storey, R., Alves, A., Ruela, J., Teixeira, L., Andrade, T., *The ATLANTIC news studio: reference model and field trial*, European Conference on Multimedia Applications Services and Techniques (ECMAST), 1997.
- [11] Knee, M., Wells, N., *Seamless concatenation - a 21st century dream*, ITS Montreux, 1997
- [12] Ferreira, P., Teixeira, V., Cardoso, P., Ruela, J., *An Open Architecture to Support Distributed Services in a Digital TV Studio*, 107th AES Convention, 1999.
- [13] Silva, U., Viana, P, Silva, U, *VIDION – An On-Line Archive for Video*, 6th DELOS Workshop, pp. 67-70, Tomar, Portugal, 1998,????.
- [14] Cardoso, P., *ORBIT - Object Reconfigurable Broadcast Using IT*, 2000.
- [15] Ferreira, P., *An Open Architecture to Support Distributed Services In a Digital Television Studio Environment*", MsC Dissertation, Faculdade de Engenharia da Universidade do Porto, Portugal, September 2000.
- [16] Santos, E., Veiga, D., *Middleware for the Broadcast Environment*, 2001.
- [17] White, D., *Distributed Systems Security*, DBMS, November, 1997.
- [18] Beznosov, K., Deng, Y., *Engineering Access Control In Distributed Systems*, 1999.
- [19] Ferraiolo, D., Kuhn, D., *Role Based Access Control*, 15th National Computer Security Conference, 1992.
- [20] Ferraiolo, D., Cugini, J., Kuhn, D., *Role Based Access Control: Features and Motivations*, Computer Security Applications Conference, 1995.
- [21] Sandhu, R. et al., "Role-Based Access Control Models", IEEE Computer 29(2): 38-47, IEEE Press, 1996.
- [22] Ferraiolo, D. et al., *A Proposed Standard for Role Based Access Control*, ACM Transactions on Information and System Security, vol. 4, no. 3, August, 2001.
- [23] Slade, R., *Glossary of Communications, Computer, Data and Information*

Security Terms, v 0.09, 2002.

- [24] Schneier, B., *Attack Trees*, Dr. Dobbs Journal, December 1999.
- [25] Moore, A. et al., *Attack Modeling for Information Security and Survivability*, CERT Report CMU/SEI-2001-TN-001, March 2001.
- [26] Beznosov, K., *Engineering Access Control for Distributed Enterprise Applications*, PHD Dissertation, Florida International University, Miami, Florida, USA, 2000.
- [27] Chizmadia, D., *A quick Tour Of the CORBA Security Service*, 1998.
- [28] Steiner, J., Neuman, C. et al, *Kerberos: An Authentication Service for Open Network Systems*, 1988.
- [29] Beznosov, K. et al., *A Resource Access Decision Server for CORBA-based Distributed Systems*, In Proceedings of the Annual Computer security Applications Conference, 1999.
- [30] Karjoth, G., *Authorization in CORBA Security*, Journal of Computer Security, 8, pp. 89-100, 2000.
- [31] Beznosov, K., Deng, Y., *A Framework for Implementing Role Based Access Control Using CORBA Security Service*, Proceedings of the Fourth ACM Workshop on Role-Based Access Control, pages 19-30, Fairfax, Virginia, USA, 1999.
- [32] Kuo, C. and Humenn, P., *Dynamically Authorized Role-Based Access Control for Secure Distributed Computation*, 2002
- [33] Schmidt, D., *An Architectural Overview of the ACE Framework: A Case-study of Successful Cross-platform Systems Software Reuse*, USENIX login magazine, Tools special issue, November 1998.
- [34] Abadi, M., Lampson, B. et al., *Authentication in Distributed Systems: Theory and Practice*, ACM Trans. Computer Systems 10, 4, pp 265-310, 1992.
- [35] Schmidt, D., Vinosky, S., *Object Interconnections: Dynamic CORBA, Part 4: The Interface Repository*, C/C++ Users Journal, January, 2003

Standards

- [36] Departement of Defense, *Trusted Computer Secuity Evaluation Criteria*, DoD 5200.28-STD, 1985.
- [37] Object Management Group, *CORBA Components Specification*, v 3.0, 2002.
- [38] International Organization for Standardization, *ISO IS 15408 Common Criteria Version 2.1*, CCIB, ISO/IEC, 2000.
- [39] Object Management Group, *Common Object Request Broker Architecture (CORBA) Core Specification*, v. 3.0.2, 2002.
- [40] Object Management Group, *CORBA Security Service specification*, v 1.7, 2001.
- [41] IETF, RFC 2246, *The TLS Protocol Version 1.0*, 1999.
- [42] Object Management Group, *Authorization Token Layer Acquisition Service (ATLAS)*, v 1.0, 2002.
- [43] ISO/IEC, *Multimedia Content Description Interface - Systems*, IS 15938-1, 2000
- [44] Object Management Group, *CORBAservices: Common Object Services*, 1998.
- [45] Object Management Group, *Resource Access Decision (RAD) Specification*, v1.0, 2001.

Web Pages

- [46] Java web site, <http://java.sun.com>
- [47] Free Software Foundation web site, <http://www.fsf.org/>
- [48] Open Source Initiative web site, <http://www.opensource.org/>
- [49] OpenCCM web site, <http://openccm.objectweb.org/>
- [50] OMG web site, <http://www.omg.org/>
- [51] Open SSL web site, <http://www.openssl.org/>
- [52] Adiron LLC web site, <http://www.adiron.com/>
- [53] ORBAsec web site, <http://www.adiron.com/ORBAsec3.html>
- [54] The Jargon File web site, <http://www.catb.org/~esr/jargon/>
- [55] Kerberos web site, <http://web.mit.edu/kerberos/www/>
- [56] Netscape's SSL web site, <http://wp.netscape.com/eng/ssl3/>
- [57] TLS web site, <http://ietf.org/html.charters/tls-charter.html>
- [58] 4VDO web site, <http://www.4vdo.pt/>
- [59] NIST RBAC web site, <http://waltz.ncsl.nist.gov/rbac/>
- [60] ACE web site, <http://www.cs.wustl.edu/~schmidt/ACE.html>
- [61] TAO web site, <http://www.cs.wustl.edu/~schmidt/TAO.html>
- [62] ORBacus web site, http://www.iona.com/products/orbacus_home.htm
- [63] Microsot Windows 2000 Kerberos Server web site,
<http://www.microsoft.com/windows2000/techinfo/howitworks/security/kerberos>

Miscellaneous

- [64] OASIS security committee, *Glossary for the OASIS Security Assertion Markup Language (SAML)*, draft-sstc-glossary-02, 2002
- [65] Santos, E., *MBE Architecture Notes*, ORBIT/INESC/059, ORBIT project internal document, 2002
- [66] Santos, E., *MBE Overview*, ORBIT/INESC/065, ORBIT project internal document, 2002
- [67] Ferreira, P., *DIMICC Overview*, ORBIT/INESC/066, ORBIT project internal document, 2002
- [68] Ferreira, P., *XDIMICC Overview*, ORBIT/INESC/067, ORBIT project internal document, 2002
- [69] Cardoso, P., *ORBIT*, DASE Symposium presentation, 2001
- [70] Rodrigues, V., *ORBIT*, ASSET meeting presentation, 2002
- [71] Brightwell, P. *ORBIT: Possible Architecture*, pjb025-possible-arch, internal document, 2000

INDEX

4

4VDO..... 11, 121

A

Abstraction..... 36, 45, 55, 75
 Accessiv, xiii, xv, 4, 37, 41, 42, 58, 74, 77,
 118, 119, 121
 Access Control iv, vi, xi, xiii, xv, 4, 29, 32,
 37, 38, 39, 40, 41, 42, 43, 46, 56, 57,
 58, 59, 61, 72, 73, 74, 75, 76, 77, 78,
 79, 81, 82, 86, 88, 92, 94, 105, 114, 116
 Accountability..... vi, 31, 32, 52, 53
 xi, xiii, xv, 96, 97, 98, 102, 109, 110,
 111, 112, 119, 121
 ACTS iv, v, xiii, xv, 7
 Adiron..... 103, 104, 121
 Analysis 33, 48, 72, 90
 Architecture xiii, xv, 6, 11, 53, 70, 117,
 118, 119, 120
 xi, 25, 90, 118, 119
 Asset ... iv, vi, 3, 11, 13, 16, 20, 30, 33, 75,
 95, 114, 116
 ATLANTIC iv, v, vi, xi, xiii, xv, 7, 8, 9,
 13, 14, 114, 118, 119
 ATLAS xiii, xv, 61, 63, 72, 82, 83, 84, 86,
 94, 103, 120
 Attack Tree 48, 50
 Attack Trees..... 119
 i, xiii, xv, 4, 29, 37, 51, 52, 56, 60, 61,
 63, 65, 66, 67, 68, 70, 78, 83, 86, 88,
 103, 105, 110, 114, 119
 Authorization .iv, vi, xiii, xv, 4, 27, 29, 37,
 38, 39, 40, 41, 43, 56, 57, 61, 62, 63,
 66, 73, 75, 76, 77, 81, 82, 83, 84, 85,
 86, 87, 88, 94, 103, 106, 107, 109, 114,
 115, 119, 120
 Servers 39, 40
 xv
 Availability ... 5, 31, 32, 34, 44, 50, 70, 71,
 77, 94

B

BBC iv, v, xiii, 2, 3, 7, 13, 15
 Business Rules 74

C

Capture..... 90
 CCM..... xiii, xv, 64, 101, 102, 119
 certificate..... 37, 84, 86, 108
 Certificate Authority xiii, 108, 109
 xi, xiii, xv, 16, 23, 24, 35, 52, 54, 56,
 57, 58, 60, 61, 63, 65, 66, 67, 68, 75,
 76, 77, 82, 86, 87, 108
 Client Authentication Layer..... xiii, xv
 Commission iv, 90
 Composition..... 90
 Confidentiality 31, 34, 56, 68, 88, 110
 Consumption 90
 CORBA.....iv, v, xi, xiii, xiv, xv, 2, 3, 4, 5,
 15, 16, 18, 19, 21, 22, 23, 24, 30, 35,
 36, 39, 50, 51, 52, 53, 54, 55, 56, 57,
 58, 59, 60, 61, 64, 65, 69, 70, 72, 73,
 74, 75, 77, 78, 80, 81, 82, 83, 85, 86,
 87, 88, 94, 98, 100, 101, 103, 107, 109,
 114, 116, 117, 119, 120, 121
 Component Model xi, xiii, xv, 18, 19, 64, 101,
 102, 119
 Security .xi, xiii, xv, 5, 56, 57, 58, 59, 70, 71, 72,
 73, 77, 78, 79, 80, 81, 82, 87, 88, 94, 103,
 104, 105, 119, 120, 121
 CORBA Security xiii, xv, 5, 56, 57, 58, 59,
 70, 71, 77, 78, 81, 88, 103, 104, 105,
 119, 120, 121
 CORBASec 5, 56, 77, 78, 81, 103, 104,
 119, 120, 121
 Countermeasures.. 5, 30, 33, 34, 48, 52, 88
 CSI 60

D

DAC 41, 81
 Data Abstraction 45
 Database. 10, 18, 38, 66, 90, 106, 108, 115
 DCOM..... xiii, xv, 53
 Delegation..... 51, 52, 58, 60, 62
 Delivery..... 90, 116
 Dependability 31, 32
 DETAIL xiii, xv, 110, 111
 DIMICC xi, xiii, xv, 17, 18, 21, 22, 23,
 24, 74, 77, 88, 96, 98, 110, 111, 122
 Discretionary Access Control 41, 81

Dynamic Interaction 36

E

Editing..... 9, 11, 12, 90, 92
 Elaboration..... 90, 93
 Enterprise Java Beans xiii, xv, 60
 Essence vi, xi, xiii, xv, 2, 3, 4, 9, 12, 13,
 14, 15, 16, 17, 20, 21, 22, 23, 27, 50,
 51, 52, 74, 77, 88, 90, 92, 95, 102, 110,
 114
 Essence Locator Service xv, 23

F

Free Software..... 96, 121

G

GIOP..... xiv, xv, 54, 60, 62
 GUI..... xiv, xv, 9, 18, 90, 91, 92, 109

I

Identity Assertion Layer xiv, xv
 IIOP..... xiv, xv, 54, 60, 69, 88, 100
 INESC Porto iv, v, vii, 2, 3, 6, 7, 11, 13,
 15, 96, 119, 121
 Information Security..... 31, 118, 119
 Integrity..... 31, 34, 56, 60, 65, 68, 88, 108,
 110
 Interaction 36, 90
 Interceptors 39, 77, 100
 Interface Repository..... xiv, 107, 115, 119

J

Java xiii, xv, 2, 3, 18, 53, 60, 74, 100, 101,
 102, 105, 109, 121

K

Kerberos. xi, 4, 60, 65, 66, 67, 68, 83, 103,
 105, 114, 119, 121
 Key Distribution Server..... xiv, xv, 66, 67

L

Least Privilege 45
 Level 1 conformance 59
 Level 2 conformance 59, 103
 Library xiii, xv, 90, 110
 Log..... 52, 53, 65, 116

Loss Cost..... 30

M

MAC 41, 81
 Mandatory Access Control..... 41, 81
 MBE...iv, v, vi, vii, xi, xiv, xv, 2, 3, 4, 5, 6,
 15, 16, 17, 18, 19, 20, 21, 23, 24, 27,
 42, 47, 50, 53, 70, 74, 75, 77, 83, 90,
 91, 92, 93, 94, 95, 96, 98, 100, 101,
 102, 103, 105, 106, 107, 108, 109, 113,
 114, 115, 116, 122
 Metadata..... vi, 2, 3, 12, 13, 14, 15, 16, 17,
 20, 22, 23, 25, 27, 50, 51, 74, 90, 107
 MOLE 7
 MPEG xiv, xv, 7, 9, 13, 14, 16, 20
 MPEG-1 14
 MPEG-2..... 13, 14, 16, 20
 MPEG-21..... 14, 16, 20
 MPEG-7..... 14, 16
 Mutual Distrust 35

N

Naming Service..... 18, 98, 100
 NIST..... 42, 43, 121
 Notification Service 18, 98, 100

O

OASIS xiv, xv, 122
 Object..... 2, 3, 96, 114
 xi, xiii, xiv, xv, 2, 3, 17, 18, 19, 20, 21,
 22, 23, 24, 35, 36, 37, 38, 41, 43, 44,
 45, 51, 52, 53, 54, 55, 56, 57, 58, 59,
 63, 75, 76, 78, 79, 80, 83, 85, 86, 87,
 88, 96, 105, 106, 107, 114, 118, 119,
 120
 Object-Oriented..... 117
 OMG xiv, xv, 18, 53, 54, 55, 56, 59, 60,
 63, 71, 75, 94, 98, 101, 102, 104, 117,
 121
 OpenCCM 101, 102, 121
 Operation... 7, 9, 31, 32, 44, 45, 48, 52, 54,
 55, 57, 58, 75, 76, 78, 79, 80, 83, 85,
 86, 106, 107, 115
 ORB . xiv, xv, 5, 52, 54, 56, 57, 59, 60, 70,
 74, 77, 81, 88, 94, 98, 100, 103, 105,
 107
 ORBacus xi, 71, 100, 103, 107, 121
 ORBAsec xi, 96, 103
 ORBIT...iv, v, vi, xi, xiv, xv, 2, 3, 6, 7, 13,
 14, 15, 16, 17, 26, 27, 90, 114, 116,
 118, 119, 122

P

Packaging.....	90
Permission... 41, 42, 43, 44, 45, 73, 74, 77, 78, 79, 80, 85, 86	
Assignment.....	45, 80, 83, 85
Playout.....	25
Policy 5, 32, 33, 36, 38, 39, 40, 41, 45, 56, 57, 58, 59, 76, 79	
Agents.....	39
Private Key	108
xi, 25, 26	
Programme.... 9, 10, 11, 13, 26, 28, 45, 65, 66, 91, 92, 93, 116	
Property Service.....	98, 100
Pseudo Random	68
Public Key	108

R

RBAC ... iv, v, xi, xv, 4, 29, 42, 43, 44, 45, 47, 72, 73, 74, 76, 77, 78, 80, 81, 82, 83, 86, 89, 94, 105, 114, 115, 116, 118, 119, 121	
Core xi, 43, 44, 72, 74, 78, 80, 82, 83, 94, 115	
Dynamic Separation of Duty.....	xiii, xv, 43, 44, 116
Hierarchical.....	43, 116
Static Separation of Duty.....	xv, 43, 116
Reference Monitor... xi, 37, 38, 39, 40, 76, 77	
Resource	14, 20, 59, 75, 85, 93, 115
Resource Access Decision... xiv, 4, 75, 76, 83, 84, 85, 86, 87, 88, 94, 105, 106, 107, 109, 114, 115, 119, 120	
Risk.....	30, 33
Analysis	33
RMI.....	xv, 53
Role... 8, 25, 42, 43, 44, 45, 72, 73, 74, 77, 78, 80, 81, 83, 84, 85, 86, 87, 88, 89, 92, 93, 94, 106, 107, 108, 109, 115, 116	
Role Authority 83, 84, 86, 88, 106, 107, 108, 109, 114, 115	
RTP.....	xv, 10

S

Scalability	2, 3, 8, 13, 35, 36, 50, 57
Scalable.....	16, 21, 59, 98
Schneier, Bruce.....	5, 34, 117, 119, 121
SECIOP.....	xv, 60
Secure Attribute Service	xv, 60, 61, 82, 86
Security i, ii, iv, xi, xiii, xv, 3, 5, 25, 29, 30, 31, 32, 34, 35, 41, 47, 53, 56, 58, 59, 60, 61, 68, 70, 73, 77, 78, 81, 87,	

95, 98, 102, 103, 105, 117, 118, 119, 120, 121, 122	
Management.....	32
Security-Aware	59
Security-Unaware	59
Security Management	32
Separation of Duties.....	45
Separation of Duty	
Dynamic.....	xiii, xv, 43, 44, 116
Service iv, 8, 10, 18, 20, 22, 23, 32, 51, 52, 56, 57, 61, 63, 67, 72, 73, 75, 76, 77, 81, 82, 83, 84, 85, 86, 88, 94, 96, 98, 103, 105, 106, 107, 109, 114, 115	
Session	44, 66, 67, 78, 80, 92
SMEF.....	xv, 14
SQL.....	xv, 14, 16
SSL xi, xv, 4, 60, 68, 69, 77, 88, 102, 105, 108, 109, 110, 111, 112, 114, 115, 121	
SSLIOP	xv, 60, 69, 88
Standard iv, v, 8, 11, 16, 18, 31, 53, 56, 59, 61, 64, 66, 68, 70, 73, 80, 82, 87, 100, 102, 103, 104, 105, 107, 114, 118, 119	
Synthesis	90

T

TAO	xi, xv, 98, 105, 121
TCP/IP.....	24, 60, 65, 68, 77, 88
Threat Model.....	xi, 48
Threats.....	33
Time Service	98, 100
Trading Service	18, 98, 100
xi, xv, 60, 68, 102	

U

UMID.....	xv, 14, 23
User4, 9, 17, 22, 27, 30, 32, 33, 35, 37, 41, 42, 43, 44, 45, 51, 52, 56, 59, 65, 66, 67, 73, 74, 77, 78, 80, 83, 84, 85, 86, 89, 91, 92, 96, 106, 107, 108, 110, 114, 116	
Assignment	43, 45, 80, 83

V

VIDION	xi, xv, 7, 10, 11, 12, 118, 119
Vulnerabilities....	32, 33, 34, 35, 47, 52, 88

X

XDIMICC xi, 24, 77, 88, 96, 98, 102, 105, 110, 115, 122	
XML..... xi, xv, 2, 3, 14, 15, 16, 18, 19, 20, 101, 102, 106, 108, 115	





FACULDADE DE ENGENHARIA

UNIVERSIDADE DO PORTO

BIBLIOTECA



0000079581