Project no. IST-033576

XtreemOS

Integrated Project
BUILDING AND PROMOTING A LINUX-BASED OPERATING SYSTEM TO SUPPORT VIRTUAL ORGANIZATIONS FOR NEXT GENERATION GRIDS

Security Services prototype month 36

D3.5.12

Due date of deliverable: 30/05/2009
Actual submission date: 29/06/2009

Start date of project: June 1^{st} 2006

Type: Deliverable
WP number: 3.5
Task number: T3.5.2

Responsible institution: Rutherford Appleton Laboratory,
Science & Technology Facilities Council,
Harwell Science and Innovation Campus,
Didcot, Oxon OX11 0QX, United Kingdom
Editor & and editor's address: Ian Johnson

Version 1.0 / Last edited by Ian Johnson / 29/06/2009

Project co-funded by the European Commission within the Sixth Framework Programme		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision history:

Version	Date	Authors	Institution	Section affected, comments
0.0	05/05/09	Ian Johnson	STFC	Created outline
0.1	08/05/09	Matej Artač	XLAB	Added initial RCA text
0.2	04/05/09	Chengchun Shu	ICT	Description of VOWeb front-end
0.3	15/05/09	Aleš Černivec	XLAB	Added initial VOPS text
0.4	29/05/09	Ian Johnson	STFC	Description of root CA, CDA, architecture
0.5	25/06/09	Alvaro Arenas	STFC	Inclusion of reviewers comments; general revision
1.0	29/06/09	Ian Johnson	STFC	Expanded description of CDA and general revision

Reviewers:
Corina Stratan (VUA) and Santiago Prieto (TID)

Tasks related to this deliverable:

Task No.	Task description	Partners involved°
T3.5.2	Specification of XtreemOS Security Services	STFC*, XLAB, ICT

°This task list may not be equivalent to the list of partners contributing as authors to the deliverable
*Task leader

1 Executive Summary

This document describes the second prototype of XtreemOS security services, implementing the services specified in D3.5.4, the second specification of security services. The services produced by XtreemOS WP3.5 are the fundamental mechanisms to prove a user's identity for authentication purposes in the Grid. These user credentials can then be evaluated in Grid-level VO policies to enable authorization decisions to be made concerning actions requested by subsystems on behalf of the user, such as resource matching in AEM job submission.

In comparison with the first prototype, deliverable D3.5.5, this prototype includes the full functionality of the Resource Certification Authority (RCA) and adding flexible database management to the functionality of VO Management, extending the registration process by allowing users to register with a Grid, to create/join VOs, and to obtain XOS credentials; furthermore, extending the VO management process by operations to create a VO, add users/groups/roles to a VO, and managing applications to join a VO.

Contents

1 Executive Summary **1**

Glossary **4**

2 Introduction **5**
2.1 Enhancements over First Prototype 6
2.2 Standards and Profiles used 7

3 Prototype Description **8**
3.1 Prototype Outline Description 8
3.1.1 Brief Introduction 8
3.1.2 Prototype Functionality 8
3.1.3 Prototype Architecture 8
3.2 Prototype Integration 8

4 Security Services and Interfaces **9**
4.1 X-VOMS 9
4.1.1 Overview 9
4.1.2 The XtreemOS Root Certification Authority 9
4.1.3 The X-VOMS Database 10
4.1.4 The Credential Distribution Authority 10
4.1.5 The CDA Client 10
4.1.6 The CDA Server 11
4.2 RCA 12
4.2.1 Brief Introduction 12
4.2.2 Major Components 13
4.3 VOPS server 14
4.3.1 Brief Introduction 14
4.3.2 Major Components 15
4.4 VOLife 17
4.5 Interaction between Security Services 18

5 Installation and Configuration of Security Services **19**
5.1 X-VOMS Root Certification Authority 19
5.2 Configuring X-VOMS database 20
5.3 Configuring and Running a Credential Distribution Authority (CDA) Server 23
5.4 RCA 26
5.5 VOPS 28
5.6 VOLife 31

6 User Guide for Security Services **33**
6.1 Using X-VOMS 33
6.1.1 Introduction 33
6.1.2 CDA Client 33
6.1.3 X-VOMS Guide for VO Administrators 36
6.1.4 Operating the Root Certification Authority 36
6.2 RCA 36
6.2.1 User guide for Resource administrators 38
6.2.2 RCA guide for Site administrators 44
6.2.3 RCA guide for VO administrators 45
6.3 VOPS 46
6.3.1 Console commands for VOPS 46
6.3.2 VOPS managed through GUI 47
6.4 VOLife 48
6.4.1 Manage VOLife users 48
6.4.2 Manage identity 48
6.4.3 Manage VO 48
6.4.4 Manage Resource 49

7 Conclusion and Future Work **50**
7.1 Future Work 50

Glossary

AEM	Application Execution Management
CDA	Credential Distribution Authority
CA	Certification Authority
CSR	Certificate Signing Request
DIXI	Distributed XtreemOS Infrastructure
NLP	Node Level Policy
PDP	Policy Decision Point
PKI	Public Key Infrastructure
RCA	Resource Certification Authority
VOM	Virtual Organization Management
VOPS	Virtual Organization Policy Service
VOLife	Virtual Organization Lifecycle service
XtreemFS	XtreemOS File System
X-VOMS	XtreemOS Virtual Organization Management Service
XOSD	XtreemOS Daemon

2 Introduction

The second prototype of the services for security and VO management implements the services specified in D3.5.4, Second Specification of Security Services [2] .

The services produced by XtreemOS WP3.5 are the fundamental mechanisms to prove a user's identity for authentication purposes in the Grid. These user credentials can then be evaluated in Grid-level VO policies to enable authorization decisions to be made concerning actions requested by subsystems on behalf of the user, such as resource matching in AEM job submission.

The credentials are contained in an XOS Certificate identifying a user. This certificate is submitted along with, for example, an AEM job request to authenticate the job requester. The user's VO attributes (such as their Global ID) are contained in the certificate and will be used by the VO Policy Service in making VO-level policy decisions, and in mapping to local UIDs/GIDs for node-level access control.

The principle of establishing trust between XtreemOS users and XtreemOS services is to use a Trusted Third Party - the Credential Distribution Authority - to enable clients to trust servers.

The security services use Public Key Infrastructure (PKI) standards and OGF profiles defining public key certificates to reflect best practice and to allow for interoperability with other Grid middleware systems.

VO policies are expressed in the XACML language, allowing flexibility and the possibility of tool support for creating policies in the future.

The specification of Secure Virtual Organization Management in D3.5.4 defines the following VO services:

- *VO Management Service* to allow creation of VOs, management of VOs, and checking of VO membership
- *VO Policy Service* A Policy Decision Engine for VO-level policies
- *Credential Distribution Authority* (CDA) to issue XOS Certificates to XtreemOS users
- *Resource Certification Authority* (RCA) to offer resources to a Grid and issue resource certificates to hosts

In the second prototype, we group the VO Management Services and the Credential Distribution Authority under the logical unit X-VOMS, the XtreemOS VO Managment Service.

The Second Prototype also provides the following web applications:

- *VOWeb front-end* Allow user to register with a Grid, create/join VOs, obtain XOS credentials
- *RCAWeb front-end* Register a RCA with a Grid, and add resources to an RCA

These web interfaces are the main usability feature added in the second prototype. They allow a user to conveniently interact with the XtreemOS security services from a web browser.

2.1 Enhancements over First Prototype

The RCA service, implemented shortly after M18, was not available for the First Prototype. The RCA was released in the XtreemOS First Sofware Release at M30, and has since been additionally enhanced.

The functionality of the VO Management service is backed by a flexible database interface, storing details of users and their registration status, details of VOs, groups and roles.

The command-line CDA client has the following refinements:

- Allow selection of primary group, as well as primary VO;
- Request specific lifetime for an XOS certificate;
- Use an existing private key for the certificate request, rather than creating a new private key.

Users can now interact with X-VOMS in the following ways:

- *Registration* Allow user to register with a Grid, and provide a means for a Grid administator to approve or reject applications.
- *Identity Management*Create a private key and XOS-Certicate via the VOWeb front-end, in addition to the command-line CDA client.
- *VO Management* Create a VO, and add groups/roles to a VO. Apply to join a VO. VO owners can approve/reject such applications.

2.2 Standards and Profiles used

To lay the foundations for interoperability with other Grid middleware system, XtreemOS uses well-established Grid standards. The fundamental principles and standards for Public Key Infrastructure are described in IETF RFC3280 [6]. The XOS Certificate conforms to the Public Key Certificate profile defined in RFC3280, and also to the Proxy Certificate Profile defined in RFC3820[9]. In addition, XOS Certificate conforms to the most relevant recommendations in the OGF Grid Certificate Profile [5].

The VO Policy Service defines policies in the XACML 1.1 language [4]. The CDA, VOPS and RCA are first defined in [1], First Specification of Security Services, and are further refined in [2], Second Specification of Security Services.

3 Prototype Description

3.1 Prototype Outline Description

3.1.1 Brief Introduction

The second prototype of the security and VO Managment services builds on the experience gained by use of the first public release of the XtreemOS software, and also adds extra tools and interfaces to make use of these services easier.

3.1.2 Prototype Functionality

The second prototype of the security and VO manager services provides the means of establishing the user's identity, their authentication credentials and their VO attributes. Resources on a resource node can be certified by the Resource Certification Authority, and offered to a VO for user by the RCA client program.

3.1.3 Prototype Architecture

The VO model in XtreemOS supports multiple VOs in a single XtreemOS Grid. The second prototype of the XtreemOS Security Services assumes the following minimal configuration:

Service	Instance count	Placement
X-VOMS	single	core node
VOWeb	single	core node
RCAWeb	single	code node
VOPS	single	core node
RCA	1 instance per administrative domain	core node

Placement of components The X-VOMS and VOWeb components access the same database, and are only supported running on the same host as the database server.

3.2 Prototype Integration

As there is minimal interaction between the services, the integration of the component services into the prototype is focussed on ensuring non-duplication of configuration information.

4 Security Services and Interfaces

This section describes in more detail the functionality provided by the following services: XtreemOS VO Management Service (X-VOMS), the Resource Certification Authority, and VO Policy Service (VOPS). The web front-ends VOFrontEnd and RCAFrontEnd are also described here.

4.1 X-VOMS

4.1.1 Overview

The XtreemOS VO Management System comprises the following components and services:

- The XtreemOS Root Certification Authority
- The X-VOMS database
- The Credential Distribution Authority.

X-VOMS provides the following functionality:

- Establishing the root of trust in an instance of an XtreemOS Grid
- Creating service certificates to protect client-server communications
- Accepting user enrollment requests
- Creating and distributing user XOS Certificates.

4.1.2 The XtreemOS Root Certification Authority

The XtreemOS Root CA holds the root of trust in the system, the Root CA private key and public key certificate. The Root CA private key is used to sign service certificates issued for XtreemOS Grid services; this allows a client connecting to such a service to authenticate the service whilst establishing an SSL connection. The client node has access to a local copy of the XtreemOS Root CA certificate, which contains the public key matching the Root CA private key.

The Root CA credentials are created once, when an XtreemOS Grid is first installed. The Root CA public key certificate needs to be installed on every node in an XtreemOS Grid. When a new grid service is installed, the Root CA is used to convert a certificate signing request (CSR) for this service into a service certificate.

4.1.3 The X-VOMS Database

The X-VOMS database holds registration information about users (such as their username, real names, expiry date, host organisation etc), and information on VOs (such as VO names and identifiers, roles and groups).

Database entities The X-VOMS database uses the Hibernate Object-Relational Mapping library to allow a flexible VO model. Entities in a VO are identified by system Globally Unique Identifiers [1] (GUIDs), and also short and long descriptions for convenience in displaying information to the user.

4.1.4 The Credential Distribution Authority

The Credential Distribution Authority provides a means for users to obtain an XOS-Certificate which contains their VO identity, and their VO attributes defining which VOs and groups they belong to, and which roles they hold. The CDA is implemented by a server which accesses the X-VOMS database.

CDA Protocol The CDA server processes requests from the corresponding CDA client, the command-line program `get-xos-cert`. This takes the form of a protocol with two commands. The user's username and password are sent with the `AUTHENTICATE user,password` command. The response to this is `AUTHOK` if the details are verified. The request for a certificate is sent with the command: `CERTREQUEST voName,groupName <CSR>`. Here, the voName and groupName are sent, along with a CSR structure, which contains the user's public key and some optional X.509 request extensions. The response to to this is the user's XOS Certificate.

4.1.5 The CDA Client

The CDA client program, `get-xos-cert`, is used whenever the XtreemOS user needs to obtain an XOS-Certificate. This contains their VO attributes, and normally lasts for 30 days.

Establishing an authenticated connection between the CDA client and CDA server The CDA client establishes a secure connection to the CDA server using SSL. The server sends the client a certificate chain consisting of the CDA's service certicate and the XtreemOS Root certificate. The client verifies the connection by checking that the hostname encoded in the CDA's service certificate is the same

[1] http://en.wikipedia.org/wiki/Globally_Unique_Identifier

as the host it is connecting to, and that the CDA's service certificate has been signed by the private key corresponding to the public key contained in the service certificates. Finally, the CDA client checks that the Authority Key Identifier field in the CDA service certificate corresponds to the client node's local copy of the XtreemOS Root CA certificate. If all these tests succeed, the CDA client can be confident that it is connecting to the authentic CDA server for this XtreemOS Grid. This is important, as the CDA client needs to protect the user's username and password, which are sent in the next step.

Authenticating the user to the CDA server The CDA client prompts the user for their username and password, and sends the `AUTHENTICATE` command to the CDA server. If the user is authenticated successfully, the next step is executed, otherwise the CDA client prints an error message to the user and exits.

Creating a private key and Certificate Signing Request (CSR) in the CDA Client The CDA client can create a new public/private keypair for the user, or use an existing keypair. (The latter option is more efficient for resource-limited nodes). If a new keypair is created, the user is prompted for a secret pass-phrase to protect the keypair, and the keypair is stored in a file readable only by the user. If an exisiting keypair is specified, the user is prompted for the passphrase protecting the keypair.

Sending the CSR to the CDA server The CDA client creates a Certificate Signing Request, and the public part of the key is added to it. If the user specifies a desired duration for the XOS-Certificate, this is encoded as an extension field in the CSR. The CSR is then signed by the user's private key, which allows the CDA server to establish the authenticity of the public key that is contains. Finally, the CDA client sends the names of the primary VO and primary group along with the CSR structure to the CDA server in the `CERTREQUEST` command.

Receiving the XOS-Certificate When the CDA client receives the XOS-Certificate in response to the `CERTREQUEST` command, it verifies that the creator of the certificate is the CDA by checking that the signature on the XOS-Certificate is valid. If this succeeds, the CDA client stores the XOS-Certificate in the user's filestore.

4.1.6 The CDA Server

Processing a connection request When the CDA client opes a connection to the CDA server, the CDA server sends a certificate chain consisting of the CDA service certificate and the XtreemOS root certificate to the client.

Authentication The CDA server authenticates a user `AUTHENTICATE` request by checking that the username and password supplied match those stored in the X-VOMS database, and that the user's registration has been approved and has not expired. If any of these details do not match, the user request is rejected.

Handling certificate requests The CDA server processes a `CERTREQUEST` by firstly checking that the user is a member of the VO specified as the primary VO. If this is the case, the user's public key is extracted from the CSR structure and used to initialise a new XOS-Certificate for the user. If the user has requested a specific validity period for the certificate (lower than the default validity period), the CDA server sets the certificate `notAfter` date appropriately.

Encoding VO Attributes in the XOS-Certificate Every user in the system has a GUID for their registration ID. This used as the value of the certificate Subject:CN field and for the VO attribute GlobalUserID. The VO ID relating to the primary VO name is used for the attribute GlobalPrimaryVOName. The CDA server nexts retrieves the list of VOs that the user belongs to from the X-VOMS database. Any VO IDs apart from the primary VO are put in a list of secondary VOs. Similarly, all groups that the user belongs to apart from the primary group are put into a list of secondary groups. The values of the GlobalPrimaryName, GlobalVOName, GlobalPrimaryGroup and lists of secondary VOs and groups are then used as the values for certificate extension fields.

Recording issued certificates The CDA server writes a record of each XOS-Certificate it issues, consisting of the certificate's serial number, the Subject:CN field, and the expiry date of the certificate.

4.2 RCA

4.2.1 Brief Introduction

In XtreemOS, the computers as computational capabilities need a means to present their identity in the process of communication with services and clients. The Application Execution Management (AEM) services enable their exploitation, and checking the trustworthiness of the nodes in a decentralised way is essential for a system that supports a growing and dynamic grid of resources. Using the PKI paradigm, the trusted resources can request signed certificates, and the trust checks can then occur without having to consult a central authority.

The **Resource Certification Authority (RCA)**, developed in the first release, is a security service that provides a base to bootstrap the trust of resource nodes

in XtreemOS. However, a network with the XtreemOS system also requires other nodes, that do not necessarily provide their computation to the jobs. Therefore we have extended the RCA's usage to extend the trust to the core nodes and the services that run on these nodes.

4.2.2 Major Components

In logical terms, the RCA is a core-level service which issues machine certificates upon request. However, the life cycle of a machine certificate and the whole mechanism of gaining the possibility to have a certificate issued lead to implementing RCA as an **RCA Server** service, **RCA database**, and an **RCA client** service.

RCA Server. This is the service that provides the main functionality of the resource certification. The service runs core-level, and its main purpose is to receive clients' requests for issuing certificates, check the validity of the requestee, and sign the certificates. The RCA Server comprises of the following:

- The RCA Server logic.
- The front-end for both the RCA server logic and the RCA database implemented for DIXI [3].
- The service certificate signed by the organisation's root certification authority or another authority with the organisation's root CA in the signature chain.

RCA DB. This is an implementational unit which stores the following iformation:

- A list of VO IDs. These IDs represent the VOs that the RCA is a member of.
- A list of pending resources. This list contains the details of those resources that the resource administrators have published and could be used for the VOs.
- A list of registered resources. This list contains the details of the resources that a site administrator has decided that they can be used for the VOs.

We exposed the RCA DB functionality using the RCA Server' DIXI service front-end.

RCA Client. The RCA Server is a stand-alone unit, and in principle the users could interact with it manually or using small programs. We provide the RCA Client as a DIXI service that runs on each node and eases the administration steps required when using the RCA. Its functionality includes the following:

- Creation of unique private keys and certification requests.
- Obtaining the details of the node from AEM's Resource Monitor service. This saves the resource administrator the tedious collection and entering the resource's details.
- Communication with the RCA Server service to send the registration application, registration confirmations, and reception of the sign certification.
- Saving, examining and installing the machine certificates.

4.3 VOPS server

4.3.1 Brief Introduction

VOPS is a **core-level service**, primarily intended serving decisions on user access to VO resources while performing some actions (e.g. job submission) and digitally signs its decisions before forwarding responses back to services. In other words, it provides coordinated access control over VO resources by offering VO level policy decision point (PDP). It also provides Policy Information Point by creating policy filters taking into account information provided by caller service. VOPS also provides mechanism to administer policies through its API (Policy Administration Point) and provided libraries.

VO-level policies and node-level policies form a hierarchical access control framework that can be tuned to achieve various degrees of control to resource usage within a VO. Target section of the policy defines the set of decision requests of a rule, policy or a policy set which are identified by definitions for resource, subject and action. Target of the policy therefore defines to whom this policy will apply. There are three types of policies managed through VOPS:

- user policies,
- VO policies,
- node policies.

User policies are managed by users themselves and users must be able to freely manage policies or rules which belong to them. Each user has his own policy defined by a target's subject section of the policy.

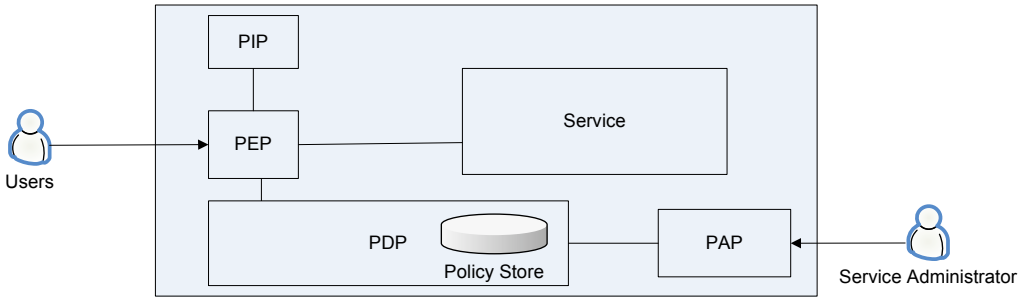

Figure 1: A generic policy model describing the relationship between PIP, PDP, PEP, and PAP

VO policies are targeted at managing broad range of users and resource nodes at a time. So in these policies VO administrator can manage access control of one specific user or group of users over one or more resource nodes. These types of policies are most general.

Node policies are policies which belong to a specific node. Target section defines a node to which rules of this policy will apply.

These types are also reflected inside the policy storage. There they are separated to make administration easier and more intuitive for users/administrators.

4.3.2 Major Components

This section describes management components which are provided by the VOPS. As illustrated in Figure 1, there are four components related to a general policy model:

- `PEP - Policy Enforcement Point:` this is where a user initiates a request, which contains information about the user, such as certificates, attributes, and context, targetted resource(s), and actions that the user wants to perform over the resource(s).
- `PDP - Policy Decision Point:` a user request is subject to the decision made by PDP, which can be a integral part of a service (as it currently stands in XtreemOS) or an independent trusted third party outside a

service providing general policy decisions for all services. A PDP is often attached to a policy store which provides all the policies that the PDP is used to make the decisions.

- PIP – Policy Information Point: upon receiving a user request, PDP can refer to PIP for further information, such as additional attributes about the user, to make decisions.

- PAP – Policy Administration Point: this is a point where a service administrator adds/modified the policy store.

VOPS comprises of next major components:

- VOPS Server front-end,

- VOPS core,

- VOPS storage.

VOPS Server. The service runs core-level and it enforces user requests against user, VO level and resource policies for gaining access to specific resource nodes. This component provides access point for

- policy administration,

- decision point for resource management services,

- filter policies.

PIP constructs request comprised of upper elements in a key-value pairs. Values are obtained from certificates or JSDL specification. Attributes with prefix *Extensions* are obtained from X509 certificate attribute extensions.

PDP is an entity which acts as a decision point, where applicable policies are evaluated and authorization decision is made. Input to the PDP are attributes and policies which apply to subject requesting a decision. Decision point is called by other services like Application Execution Manager's ResMng.

PAP is used as an administration point, where policies can be initially created, maintained and eventually removed. PAP acts as a source of policies which are used in PDP when evaluating a request.

VOPS policy storage Currently, policies are stored as plaintext XML files under the location provided by the VOPS configuration file. In the final release, VOPS will incorporate **eXist** [7] as an XML database engine. Replacing our own code for parsing XML and string manipulation with a tested engine, we have increased the flexibility of policy languages and the usage of the overall service.

EXist has also simple backup/restore procedures which can be easily triggered from VOPS. During backup, eXist exports the contents of its database (as standard XML files) to a hierarchy of directories on the hard drive. This hierarchy is organized according to the organization of collections in the database. This way the whole structure of XACML policies will be backed up and could be restored when administrator wishes to run maintenance procedures (or if automatically maintenance will be scheduled to run periodically).

4.4 VOLife

In XtreemOS, VO related services X-VOMS, RCA and VOPS are interacted with using small programs or client services(e.g. RCA client), with which aside from knowledges of the services themselves a user needs explicitly be engaged in tedious tasks such as manually acquiring the certificates, and setting up service bootstrap configurations. In the cases, the high knowledge requirement and learning curve impede the usability of core services and thereby limit their widespread acceptance. Besides the client programs of the services, VOLife is implemented as a web fronted to these core services, providing a one-site client service that exposes the VO-related functionalities, as well as simplifies the users' requirements of accessing the core services via a web interface.

Logically VOLife comprises three major components: identity management, VO management, and resource management.

Identity management. This is the interface to identity-related services which provides functionalities including grid user sign up, PKI key pair generation and download, and XOS certificate signing and maintenance.

VO management. This is a essential part that enables a user to create, control its own VOs, and request to join/leave other VOs without ownership in order to provide or share resources.

RCA management. provides a user capabilities of resource management, including registering Resource Certificate Authorities (RCAs), adding/deleting resources to RCA and VOs, approving resource adding requests, and acquiring ma-

chine certificates. RCA management is a user-friendly interface to the RCA core service.

4.5 Interaction between Security Services

During the service configuration phase, the Root CA is involved in processing requests for service certificate (Certificate Signing Requests). The human operator of the Root CA verifies the authenticity of the requestor, and uses the process-csr command to create the service certificate from the CSR file.

Once configured, the security services are loosely coupled. There is currently no inter-service communication.

The VOWeb, RCAWeb and CDA services access the X-VOMS database with the following access modes:

Component	VO Web	RCA Web	CDA Server
Database Access Mode	Read/Write	Read/Write	Read Only

Currently, the only deployment option supported is to run these services on the same machine as the X-VOMS database server.

5 Installation and Configuration of Security Services

This chapter describes how to install the constituent security services and how to configure them.

5.1 X-VOMS Root Certification Authority

The Root CA is the top level of the trust mechanism in XtreemOS. It is a critical part in the XtreemOS Public Key Infrastructure (PKI). To achieve and maintain the level of trust required by users of an XtreemOS Grid, the Root CA must be operated only on one machine. This host must be a physically-secure core node to avoid compromise of the Root CA private key, which would destroy any trust placed on the Root CA. Some organisations may choose to run the Root CA on a machine which isn't connected to a network, to eliminate any risk of intrusion.

The Root CA comprises root entity credentials which are trusted by all participants in an XtreemOS Grid, and a mechanism to create service certificates that identify other XtreemOS core services.

The package `rootca-config` contains the configuration files for creating a Root CA, and for creating service certificates from Certificate Signing Requests.

Install the `rootca-config package`. This places configuration files in `/etc/xos/config/openssl`. Decide on a directory to hold the files related to the Root CA, for example, `/opt/xtreemosca`.

The Root CA certificate is configured by properties in the file `/etc/xos/config/openssl/create-rootca-creds.conf`. The section `[ root_ca_distinguished_name ]` can be modified to change the certificate fields `commonName`, `organizationName` and `organizationalUnitName` as required. The `[ req ] section` Âăcontains the property `default_days` to set the duration of the certificate's validity, and `default_bits` to set the size of the Root CA private key.

To create the Root CA directory, the private key and public certificate, run the command presented in Figure 2.

```
Root # create-rootca /opt/xtreemosca
```

Figure 2: Creating the XtreemOS Root CA.

You will be prompted for a passphrase - this protects the private key, and is required when using the Root CA to create service certificates from Certificate Signing Requests (CSRs). This passphrase must be kept secret to prevent use of the private key by anyone other than the operator of the Root CA.

The private key is created in the sub-directory `private` under the Root CA (in this case, `/opt/xtreemosca/private/xtreemos.key`). The public key certificate of the Root CA is the XtreemOS 'root certificate'. It is created in the sub-directory `public` of the Root CA directory (in this example, `/opt/xtreemosca/public/xtreemos.crt`). The XtreemOS root certificate needs to be installed on all machines in this XtreemOS Grid. The certificate can be placed in
`/etc/xos/truststore/certs/xtreemos.crt` on these machines.

The Root CA is now ready for its operational role. This consists of processing Certificate Signing Requests (CSRs) from administrators of core node (for applications such as CDA, RCA and VOPS servers, and XtreemFS client and servers). This is described in Section 6.1.4, 'Operating the Root Certificate Authority'.

5.2 Configuring X-VOMS database

X-VOMS (XtreemOS Virtual Organization Management Service) is an advanced Virtual Organisation (VO) management service for supporting secure and flexible collaborations and resource sharing among people, projects and organisations. It is written in Java and back by a (Hibernate-based) X-VOMS database schema. Like other VO management software packages, X-VOMS provides a set of APIs for managing identity, attributes, and VO membership of users and resources.

X-VOMS can be used as a backend of different presentation frontends: a web application (allowing the access via a web browser), and a OS daemon service (allowing the access via a OS command line console, or directly from a user application). In *the current release*, X-VOMS is not a standalone service. It attaches to the VOLife web frontend to provide (part-of) its VO management capabilities to end users. In the future releases, the daemon frontend of X-VOMS will be offered so that applications can directly utilize X-VOMS functionalities.

X-VOMS manages, but does not distribute, credentials. It can be used with a Certification Authority (CA), such as the Credential Distribution Authority (CDA) service developed by the XtreemOS project, or a third-party attribute authority, to disseminate credentials.

X-VOMS also supports home volume creation for users of XtreemFS, a Grid file system being developed in the XtreemOS project.

This instruction assumes you know how to use MySQL (e.g. how to add a user in MySQL). For user management in MySQL, please read:
`http://dev.mysql.com/doc/refman/5.0/en/adding-users.html`

Software prerequisites The current X-VOMS implementation relies on the following software:

- Hibernate 3.0 [2] (or newer)
- MySQL 5.1.6[3] (or newer)

Major files and their location The steps needed to create the X-VOMS database and load it with data are encapsulated in the script xvoms_prepare_database.sh:

Configure the X-VOMS database, as illustrated in Figure 3.

```
Root # /usr/share/xvoms/bin/xvoms_prepare_database.sh
```

Figure 3: Configuring the X-VOMS database.

Running this script is sufficient to allow the following steps 'Installing the CDA' (5.3), 'Installing VOLife' (5.6) etc to be performed.

The following files described below are merely described for reference purposes.

The configuration files are located at: `/usr/share/xvoms/`. The X-VOMS library (xvoms-version.jar) is located at: `/usr/share/java/`.

- /usr/share/xvoms/scripts/setup.sql

  This script sets up the basic X-VOMS table schema. It is very important that you perform this step before starting to test any X-VOMS functionalities. Without setting up the tables, some security features (such as X-VOMS access control and authentication) cannot be demonstrated or your requests will be automatically denied (Figure 4).

```
Root # mysql -u some_user --password=some_pass < setup.sql
```

Figure 4: Initialising the X-VOMS database.

  This line populates three tables: rules, actors, and actions, which are essential for X-VOMS.

- `/usr/share/xvoms/data/xvoms.txt`

  a sample xvoms database file, including both schema and some sample data (users, vos, vo attributes). To use this file to setup a sample xvoms database, perform the steps presented in Figure 5.

[2] http://www.hibernate.org/

[3] http://www.mysql.com

```
Root # mysql -u root --password=xxxxx
mysql> create database xvoms;
mysql> quit;
mysql -u root --password=xxxxx xvoms < \
/usr/share/xvoms/data/xvoms.txt
```

Figure 5: Populating the X-VOMS database.

An example accounts `xtreemos-vouser`, is included in the file `xvoms.txt` to allow testing X-VOMS. The password for this account is 'xtreemos'.

To refresh the sample file, you can do as shown in Figure 6.

```
Root # mysqldump xvoms -u root --password=xxxxx  -r \
/usr/share/xvoms/data/xvoms.txt
```

Figure 6: Refreshing the X-VOMS database.

- `/usr/share/xvoms/xsl/junit-noframes.xsl`

  xslt for transforming junit test reports (in XML) to html. This file should be used with the *source distribution* of X-VOMS, which contains `build.xml` to allow generation of junit test reports for X-VOMS functionalities.

- `/usr/share/xvoms/hibernate.cfg.xml` a hibernate configuration file for setting hibernate connection properties. The most notable settings are:

  ```
  <property name="connection.url">jdbc:mysql://host:3366/xvoms
  </property>
  <property name="connection.username">volifecycle</property>
  <property name="connection.password">xosvo</property>
  ```

- `/usr/share/xvoms/log4j.properties` a log4j configuration file for setting hibernate logging properties. The most notable settings are:

  ```
  log4j.logger.org.hibernate=fatal
  log4j.logger.org.hibernate.SQL=fatal
  ```

- `/usr/share/xvoms/MRC.properties` a MRC/XtreemFS home volume configuration file for setting MRC server properties. The most notable settings are:

```
mrc.host=localhost
mrc.port=32636
```

The `mrc.host` property should be set to the location of the machine running the MRC server.

Other notes Apart from MySQL, you are free to choose any other JDBC-compliant database engines that Hibernate supports, including Oracle, DB2, Sybase, MS SQL Server, PostgreSQL, MySQL, HypersonicSQL, Mckoi SQL, SAP DB, Interbase, Pointbase, Progress, FrontBase, Ingres, Informix, and Firebird.
See `http://www.hibernate.org/344.html` for more details.

5.3 Configuring and Running a Credential Distribution Authority (CDA) Server

This sub-section is for a Grid administrator running a CDA server.

The Credential Distribution Authority is implemented in the **cdaserver** package. For the first release of XtreemOS, the CDA server runs on only one core node in an XtreemOS Grid.

The standalone CDA client program can be used to obtain user VO credentials from the CDA, and is provided by the **cdaclient** package.

The CDA server issues XOS certificates to users. The server needs an service certificate issued by the Root CA to authenticate itself to the corresponding CDA client. This service certificate can be obtained by the procedure described in section 5.3. This procedure also produces a private key, which should be placed into `/etc/xos/truststore/private/cda.key`. The service certificate contains the service's public key, and should be placed in `/etc/xos/truststore/certs/cda.crt`.

As root, install the CDA server as shown in Figure 7.

```
Root # urpmi cdaserver
```

Figure 7: Installing the CDA server.

The following aspects of the CDA server are configurable by setting values in the file `/etc/xos/config/cdaserver/cdaserver.properties`:

- **cdaserver.keyFilename** — private key of CDA server - must be kept secure, readable only by owner.
- **cdaserver.keyPassphrase** — the private key is secured by a passphrase, the longer the better.

- **cdaserver.certFilename** — public key certificate of CDA server.
- **xtreemos.rootCertificate** — public key certificate of root CA.
- **cdaserver.sslAlgorithm** — cipher used by SSL.
- **cdaserver.sslHandshakeCipher** — the cipher used in initial SSL key exchange.
- **cdaserver.signatureAlgorithm** — algorithm used to sign the XOS-certificate returned to user.
- **cdaserver.validityDays** — number of days that certificate is valid for
- **cdaserver.validityHours** — number of hours that certificate is valid for
- **cdaserver.validityMinutes** — number of minutes that certificate is valid for

The default validity of a certificate is calculated as (cdaserver.validityDays) days + (cdaserver.validityHours) hours + (cdaserver.validityMinutes) minutes. Any two of these values can be zero. Hence, the lifetime of certificates issued by the CDA server can be set on a fine basis, if required. The CDA server will create a certificate with a shorter lifetime than the default if the CDA client program *get-xos-cert* is invoked with the '-D days' argument.

Other aspects of the CDA server operation are:

Connection to X-VOMS database - this is set in hibernate.cfg.xml

The level of logging, log file location, etc, are defined in log4j.properties.

Once configured, the server is started by issuing the command shown in Figure 8.

```
Root # /sbin/service cdaserver start
```

Figure 8: Installing the CDA server.

The server writes its log files in `/var/log/cdaserver/cdaserver.log` by default.

Most service certificates are used to authenticate core services to client programs. For mounting XtreemFS filesystems, one mode of use is to use the service certificate for the `xtfs_mount` application to authenticate the client host to the XtreemFS server. Alternatively, the XtreemFS mount client can also use an XOS-certificate if the client is being run on behalf of a single user.

Prerequisite for installing any core service application. The following conditions apply:
Before installing any server, the XtreemOS Root Certificate Authority must be active in your XtreemOS Grid. See Sections 5.1 and 6.1.4 for details.
The `create-csr` package must be installed; it contains the **create-csr** command and an OpenSSL configuration file to create a certificate signing request (CSR) file for an application. The requestor must then send the CSR to the operator of the Root CA to obtain the service certificate.

The steps involved are shown below.

The **create-csr** command creates a Certificate Signing Request (CSR) file. The arguments to this command are:

- the host name — this is encoded in the subjectAltName extension field of the certificate, and as part of the Subject CN field. The Fully-Qualified Domain Name for the host is required, not its IP address. Some client programs, such as the CDA client, will check this field during the SSL handshake against the FQDN of the server they are attempting to connect to.
- the name of your organisation.
- the name of the application. This is incorporate into the Subject CN field as <fqhn>/<application>. E.g. for a CDA server at host, the Subject field would include `CN=host/cda`.
  Legitimate values for the application argument are:
  - `cda` The Credential Distribution Authority server
  - `rca` The Resource Certification Authority server
  - `vops` The VO Policy Service server
  - `mrc` The XtreemFS Metadata and Replica Catalogue Server
  - `dir` The XtreemFS Directory Service
  - `osd` The XtreemFS Object Storage Device server
  - `xtfs_mount` The XtreemFS mount client

An example, creating a request for a service certificate, where `host` is replaced with either the Fully-Qualified Domain Name for the host, or its IP address. The last argument to this command identifies the type of service/client that this certificate will be used by (Figure 9).

The command in Figure 9 produces a private key for the application in `host-cda.key`, and a CSR in `host-cda.csr`. Send this CSR file to the administrator of the Root CA in your organization to get an service certificate (e.g. `host-cda.crt`)

```
create-csr host "My Organization" cda
```

Figure 9: Creating a request for a CDA service certificate.

in return. Install this service certificate in `/etc/xos/truststore/certs/cda.crt` and the private key in
`/etc/xos/truststore/private/cda.key`.

The passphrase protecting the key can be specified in the properties/configuration file of the server it is to be used with. In this case, you must ensure that the file containing the passphrase is only readable by the owner of the service itself, e.g. for the CDA server, the properties file should only be readable by '`cdauser`'.

Connecting the CDA server to the X-VOMS database The CDA server uses the Hibernate ORM library to retrieve VO attributes from the X-VOMS. Hibernate uses a JDBC connection that is specified by the parameters in the Hibernate configuration file, `hibernate.cfg.xml`. The settings that may need to be changed here are `connection.username` and `connection.password`.

5.4 RCA

The Resource Certification Authority services run as DIXI services.

RCA comes in two packages:

- **vom-rca-node** — This package contains the node level service which should run on each node capable of executing jobs.
- **vom-rca-server** — This package contains the core-side service which usually runs on one node within a physical organisation.

Both packages depend on the package **rcalib**, which contains the service's logic library.

To install the necessary software, simply use `urpmi` with the name of the package. In order to actually run either RCA client or the RCA server, the DIXI daemon's configuration file (**XOSdConfig.conf** by default) needs to have its handler enabled. The configuration files used by the RCA are placed into **/etc/xos/-config/** by default.

The RCA Client does not depend on any AEM services, but it can take advantage of the AEM's Resource Monitor to learn the details of the local resource.

Enabling services in DIXI daemon's configuration. To have one or both services start with the local DIXI daemon, place an empty file with proper file name into `/etc/xos/config/xosd_stages`, named after the class that starts the stage. The names are as follows:

- **RCA server**: eu.xtreemos.xosd.security.rca.server.service.RCAServerHandler.stage
- **RCA client**: eu.xtreemos.xosd.security.rca.client.service.RCAClientHandler.stage

Configuring core-level RCA service. The RCA server service creates and uses the **RCAServerConfig.conf** to obtain the configuration:

- **certDNLocation** — the location of the organisation covered by the RCA server. The value is a part of the distinguished name (DN) of a certified resource.
- **certDNCountry** — the country of the organisation covered by the RCA server. The value is a part of the distinguished name (DN) of a certified resource.
- **certDNOrganisation** — the name of the organisation covered by the RCA server. The value is a part of the distinguished name (DN) of a certified resource.
- **certDNOrganisationUnit** — the name of the organisation unit covered by the RCA server. The value is a part of the distinguished name (DN) of a certified resource.
- **daysCertValidity** — The number of days the certificate will be valid, starting from the day of certification and expiring this number of days later.
- **privateKey** — The path to the server's certificate authority's private key.
- **certificateFileName** — The path to the server's certificate authority's public key/certificate.
- **cdaPassword** — The server's certificate authority's public key's passphrase.
- **keyPassword** — The server's certificate authority's private key's passphrase.
- **rcaDBFile** — The path to the file containing the RCA DB.
- **attributeType** — the type of the attribute certificates. Use V2 for attribute certificates, or V3 for certificates with attributes stored in extensions. The default value is V3, and it is a recommended value for compatiblity with openssl libraries.

The RCA server requires a private key and a certificate signed by a certification authority that is trusted by the nodes in the XtreemOS. The RCA server will use the certificate signed by a commonly trusted root authority to sign the machine certificate requests. The steps for creating the certificate for the RCA are similar to those described in Section 5.3 for the CDA server. The location of the private key and the certificate are defined by **privateKey** and **certificateFileName** of the **RCAServerConfig.conf**, respectively.

Configuring node-level RCA service. The RCA client service creates and uses the **RCAClientConfig.conf** to obtain the configuration:

- **cdaCertificateFileName** — the path to the RCA server's certificate authority's public key/certificate.
- **resPrivateKeyFileName** — the path to the resource's private key.
- **resIdentityCertFileName** — the path to the resource's identity certificate (public key).
- **resAttributeCertFileName** — the path to the resource's attribute certificate (attribute certificate).
- **resAttributeCertExtFileName** — the path to the resource's attribute certificate (attributes stored in an extension).
- **resVOAttributeCertIncoming** — the path to the folder that will store the attribute certificates pushed from the RCA Server.

5.5 VOPS

VOPS installation and setup VOPS is a **core-level service** which, due to usage of the DIXI framework, runs as a service using DIXI communication stages. VOPS has to be started in a way like other XOS daemons are: using *xosd* script provided in a bundle containing VOPS package. First, administrator has to set up **XOSdConfig.conf** and **VOPSConfig.conf** appropriately. **ResMng.conf** (on server, where ResMng service is running) has to be configured appropriately to use VOPS, see also figure 11. VOPS is a server primarily intended serving requests and forwarding answers from/to resource discovery services and therefore it needs private key and public certificate to be able to digitally sign its decisions before forwarding them to services. Services querying VOPS should have access to VOPS public certificate to be able to check authenticity of its answers.

To be able to run VOPS server using DIXI framework, place an empty file with proper file name into `/etc/xos/config/xosd_stages`, named after the class that starts the stage:

`eu.xtreemos.xosd.security.vops.service.VOPSHandler.stage`

If VOPSConfig.conf does not exist yet, you can run xosd and stop it. This way VOPSConfig.conf is automatically generated under /etc/xos/config, where you can edit it manually (see figure 10).

```
enableAccessControl=true

VOAdminRoles.size=15
VOAdminRoles.0=role_get_VOAttributes4

ResourceAdminRoles.size=15
ResourceAdminRoles.0=res_role_get_VOAttributes

serviceKey=/etc/xos/truststore/private/vopsserver.pem
policyStorage=/usr/share/dixi/VOPS/files/policy/testStorage
keyPassword=xtreemos
```

Figure 10: A sample VOPS configuration file.

- **globalVOPS.port** and **globalVOPS.host** legacy settings that are not used, so they can be safely comment out or ignored.
- **enableAccessControl** enables or disables access control: if enabled, extension (role) from user certificate is checked whether it is one from roles listed under **VOAdminRoles** or **ResourceAdminRoles**.
- **VOAdminRoles.size** is the size of array defining VO administrator roles.
- **VOAdminRoles** are roles of users which are permitted to manipulate with XACML policies. These roles must be same as roles specified in certificates (VO administrator roles).
- **ResourceAdminRoles.size** is the size of array defining resource administrator roles.
- **ResourceAdminRoles** are roles of users which are permitted to manipulate with XACML policies. These roles must be same as roles specified in certificates (resource administrator roles).

- **serviceKey** is VOPS's private key used to sign responses.

- entry **policyStorage** points to storage (XML files) which contains user policies and resource policies defining access control to users over these resources.

```
#Properties File for the client application
#Thu Jun 26 13:08:14 CEST 2008
VOPSPubCert=/etc/xos/truststore/certs/vopsserver.pem
testVOPS=true
```

Figure 11: A sample ResMng configuration file.

While resource discovery services have to check authenticity of the VOPS's answers, the node running **ResMng** service has to include next lines in its configuration files. List of entries under **ResMng** configuration file:

- **VOPSPubCert** is path to public certificate of the vops server.

- **testVOPS** enables or disables calls to VOPS service.

It is important that if VOPS is to enforce policies over user queries, RCA client must run on resource node which is considered in query. VOPS needs to access RCA client service to obtain resource certificates from which attributes are considered in the query.

Packages:

- **vom-vops** —The VOPS service provides means to store and manage VO-level policies, to obtain the policy filters and the policy decisions on the VO level.

eXist installation and setup In the second prototype eXist [7] version 1.2 of eXist database is used. EXist is installed under /usr/share/exist directory where additional scripts for configuration, running, shutting down and configuration are provided and are going to be called when installing VOPS on the system so the installation is transparent for the user.

5.6 VOLife

VOlife runs as a web application deployed in Tomcat container. The requirements to install Volife include Java Development Kit 1.5 above, Apache Tomcat 5.0 above and MySQL 5.0.45 or above.

VOLife is implemented in two parts: backend and frontend.

- **backend** — This part contains the VOLife database setup sql statements, and java library (i.e. JAR files) that interfaces to the VO-related core services by the VOLife web interfaces.

- **frontend** — This package forms the core of web application VOLife which exposes the VO related functionalities for nontechnical users with limited computer expertise.

VOLife installation comprises of the following steps:

1) Deploy the frontend part as a web application running in tomcat container: put the frontend part under directory tomcatdir/webapps/volifecycle (Figure 12).

```
Root # cp volife/frontend /usr/share/tomcat5/webapps/volifecycle
```

Figure 12: Deployment of the frontend.

2) Create VOLife database. This is done by importing the database schema and table content into MySQL using the database setup sql statements provided in the backend part. This is database will also be prepared by xvoms, so the creation can be skipped if the xvoms has been installed (Figure 13).

```
# /usr/share/xvoms/bin/xvoms_prepare_database.sh
```

Figure 13: Creation of VOLife database.

3) Install java libraries for VOLife web application. The java libraries to install are backend library and their dependent libraries, which should be put into directories WEB-INF/classes and WEB-INF/lib respectively. By default the step can be omitted as the java libraries has been included in the deployable frontend part (Figure 14).

4) Configure the web application. One configuration file that may need modification is for sign up notification email.

```
# cd volife/backend;
# ant   //build the backend classes,
//assuming the dependent java jar files are under
//volife/frontend/WEB-INF/lib/
# cd ../../
# cp -af volife/backend/build/org
/usr/share/tomcat5/webapps/volifecycle/WEB-INF/classes/.
# cp volife/backend/build/eu
/usr/share/tomcat5/webapps/volifecycle/WEB-INF/classes/.
```

Figure 14: Installation of Java libraries for VOLife.

Configuring sign up notification email. The configuration file can be found under directory tomcatdir/conf/mail.conf. The file configures the email account that sends out sign up notification to a grid user when he has signed up. The format of the configuration file is compatible with JavaMail config file. The common configuration items are given as follows:

- **mail.smtp.host** — the hostname or IP address of the SMTP server the sends out the notitificaiton email.
- **mail.smtp.auth** — the flag that indicates whether the email sending through the smtp server should be authenticated, with two possible values *true* and *false*.
- **mail.smtp.port** — the port number of the smtp server.
- **mail.smtp.user** — the username of the email account, acting as the sender of the notification email.
- **mail.smtp.password** — the password for the email account that authenticates to the smtp server.

An example configuration is presented in Figure 15.

```
mail.smtp.host=smtp.gmail.com
mail.smtp.auth=true
mail.smtp.port=465
mail.smtp.socketFactory.port=465
mail.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory
mail.smtp.socketFactory.fallback=false
mail.smtp.password=PASSWD
mail.smtp.user=MAILACCOUNT
```

Figure 15: Example configuration sign up notification email.

6 User Guide for Security Services

6.1 Using X-VOMS

6.1.1 Introduction

The XtreemOS user can access the following features in X-VOMS:

- Identity Management - Store details of user identity, secure their access to system
- VO Managment - Allow creating/joining VOs
- Credential Management - generate private keys and XOS certificate containing user's public key and VO attributes

The interface to of these features is available via web front-ends and is described in Section 6.4. In addition, there is a command-line tool for credential management, the CDA client.

6.1.2 CDA Client

The CDA client can be used to generate a user's private key and obtain their XOS Certificate containing their public key and their VO attributes. Optionally, the CDA client can use an existing private key file. The CDA client allows the user to request a duration for the validity of their credentials.

The CDA client is invoked by the command `get-xos-cert`. The example in Figure 16 shows how to request a certificate for the user `xtreemos-vouser` belonging to the VO `vo1`.

```
$ get-xos-cert host:6730 vo1 -g group1
Enter your username: xtreemos-vouser
Enter password: <not echoed>

Passphrase to protect private key
(at least 8 characters long): not echoed
Type passphrase again to confirm: not echoed

Saving user certificate in
/home/user/.xos/truststore/certs/user.crt,
saving private key file in
/home/user/.xos/truststore/private/user.key.
```

Figure 16: Example user requesting certificate.

The example in Figure 16 retrieves the user's VO attributes for the VO *vo1* and primary group *group1*. The username supplied is that of the pre-defined user *xtreemos-vouser* (password *xtreemos*).

The values of the user's VO attributes can be viewed with the command *view-xos-cert*, as shown in Figure 17.

```
$ view-xos-cert /home/user/.xos/truststore/certs/user.crt
Certificate:
   Data:
       Version: 3 (0x2)
       Serial Number:
           01:21:43:39:3c:16
       Signature Algorithm: sha256WithRSAEncryption
       Issuer: O=XtreemOS, OU=cda, CN=host/cda
       Validity
           Not Before: May 15 07:37:53 2009 GMT
           Not After : Jun  4 07:47:53 2009 GMT
       Subject: CN=ea9a7366-e34f-4a99-9e31-277430366475
       X509v3 extensions:
           X509v3 Basic Constraints: critical
               CA:FALSE
           X509v3 Key Usage: critical
               Digital Signature, Key Encipherment
           X509v3 Extended Key Usage: critical
               TLS Web Client Authentication
... (Details excluded)

       XtreemOS VO Attributes:

           GlobalPrimaryVOName:
               2c0e8cb2-4453-46fe-85b7-74874e76e7c2
           GlobalSecondaryVONames:
               null
           GlobalUserID:
               ea9a7366-e34f-4a99-9e31-277430366475
           GlobalPrimaryGroupName:
               ae88816f-9f5c-48f9-ad7d-f71a64977904
           GlobalSecondaryGroupNames:\
41ef52b1-8c03-4305-bbfb-9f07245702cd,\
9d481237-26c9-4854-a1e2-e23d1a61059c
           Role:
               null
           Group:
               group1
           Subgroup:
               null
```

Figure 17: Viewing a XOS certificate.

6.1.3 X-VOMS Guide for VO Administrators

Administrating X-VOMS consists of operating the Root CA to provide service certificates, and running the constituent services.

6.1.4 Operating the Root Certification Authority

The main operating mode of the XtreemOS Root CA is to take Certificate Signing Requests (CSR files) for applications and convert them to service certificates. The CSR file can be sent to the administrator of the root CA in an email message or by other means. The operator of the root CA should save the CSR file and examine it to check its validity, and contact the originator of the request if necessary (to verify its source).

For example, figure 18 shows the contents of a request for the CDA server on the host `host` (the values in your CSR files will, of course, be different).

Note that the Subject Alternative Name extension field contains the FQHN (Fully-Qualified Host Name) of the machine that this certificate will be used on. This hostname, along with the name of the application, is also encoded in the Subject CN field.

If you trust the originator of this request, and you have validated the request by examining it as shown above, you are now in a position to generate an service certificate from the request. Figure 19 shows the step needed to create an service certificate from a CSR file, in this case for the CDA server at `host`.

The service certificate is created in the file `host-cda.crt` in the current working directory. A copy of the service certificate is also stored in the 'certs' subdirectory of the root CA. In the example above, this would be `/opt/xtreemosca/certs`), named `<NN>.pem`. A record of the newly issued service certificate is made in `/opt/xtreemosca/index.txt`, including the certificate's expiry date, revocation status, serial number and CN.

6.2 RCA

The RCA issues machine certificates, encrypted using a so-called machine password. This means that in principle the resource administrator is never prompted for the password needed to access the private key part of the machine certificate. The cycle of the initial sending the resource's details, registring the resource, requesting for certificate and storing it could therefore be done automatically. If an admin would want a higher level of security, she would need to enter the password at each boot time or service start-up. However, this would not ensure trust and security in any way. Therefore we want to have people with authority taking part within this loop:

```
$ view-csr host-cda.csr
Certificate Request:
 Data:
  Version: 0 (0x0)
  Subject: CN=host/cda, O=XtreemOS Project,OU=cda
  Subject Public Key Info:
   Public Key Algorithm: rsaEncryption
   RSA Public Key: (1024 bit)
    Modulus (1024 bit):
     00:d0:ec:ed:89:93:2e:c3:23:47:7c:30:1e:de:fb:
     40:bb:9a:6f:bd:77:35:54:28:24:b2:62:9b:cc:9e:
     dd:f5:1b:19:55:be:fe:0b:9f:2d:56:a6:98:bd:77:
     53:1c:da:38:3a:ba:60:03:90:f9:bc:a4:af:ec:5a:
     c1:ec:80:34:cb:bd:fa:18:93:af:c1:84:5d:16:72:
     ed:94:e8:eb:59:13:1f:99:6b:ac:93:d3:e5:07:e1:
     54:77:e8:8f:44:4c:4a:0b:31:5c:26:af:19:c3:f6:
     6c:71:22:cb:0c:56:47:99:f3:14:ab:1b:43:de:e9:
     13:48:17:00:f0:da:0c:de:e1
    Exponent: 65537 (0x10001)
   Attributes:
   Requested Extensions:
    X509v3 Basic Constraints:
     CA:TRUE
    X509v3 Subject Alternative Name:
     DNS:host.org.domain
    Signature Algorithm: sha1WithRSAEncryption
      5a:c2:4e:aa:8f:bc:e2:c5:b2:0a:12:20:92:d5:90:de:fb:96:
      bb:d7:c3:5f:6d:67:89:5e:b1:6c:e1:9e:c6:e7:8e:e9:42:ea:
      0e:65:f1:3d:e9:73:31:44:95:6c:d3:3c:b4:b1:cc:bb:e6:1c:
      3c:a2:c1:99:a7:2e:d4:24:10:06:49:99:51:94:53:81:d9:8e:
      4d:a2:f5:1c:ac:df:19:e0:f4:bb:96:19:5f:74:88:57:e3:82:
      01:e7:93:a1:45:cc:3e:ef:54:28:bb:8a:1e:c0:3a:a9:dd:85:
      00:2f:ac:c7:b8:5c:c8:94:99:2f:a7:04:76:d4:74:84:f4:5d:
      a7:92
```

Figure 18: Examining Certificate Signing Request for a service certificate.

```
Root # process-csr /opt/xtreemosca host-cda.csr
```

Figure 19: Signing a host CSR file, creating an service certificate.

- Resource administrators install the XtreemOS services and provide the environment for the computation. The administrators that own and prepare nodes to run core XtreemOS services belong to this category as well.
- Site administrators decide which resources are trustworthy and usable for XtreemOS. These actors ensure that it is not possible to install any computer with potentially compromised or malicious software, and have it run alongside the other XtreemOS nodes on the site.
- VO administrators include an RCA in their VOs, or decide an RCA and its underlying resources would no longer be involved in the VO.

Shell commands provide the functionality for each type of users.

6.2.1 User guide for Resource administrators

The object of the RCA's operation are the certificates of various purposes. In order to provide them to the daemons and services, should be placed in the system's directory where the services expect them. The specific directories depend on the configuration of the service or the daemon. However, as a convention the XtreemOS services use the default configurations which expect their public keys and certificates to be placed in

```
/etc/xos/truststore/certs
```

and the private keys to be located in

```
/etc/xos/truststore/private
```

Registering a resource. A resorce or a node that has a fresh installation of the XtreemOS system, has to be first registered before it can obtain any certificates from the RCA. This process involves sending the details on the node to the RCA server, putting them to the list of the candidate resources. The details needed include

- the node's address and end-point (IP, port),
- the node's static metrics, such as the number of CPUs, the CPU speed, the memory size etc.,
- the list of services that will run on the node.

These details are collected automatically and sent to the RCA server by calling

```
$ rca_apply
```

The process of the node registration needs to be completed by the site administrator before the resource administrator can proceed.

OID	Label	Explanation
1.34.5.0.15.1	CPUSpeed	The clock of the CPU, in Hz
1.34.5.0.15.2	CPUCount	The number of computational units (CPU cores or CPUs)
1.34.5.0.15.3	MemorySize	The size of the physical memory, in B
1.34.5.0.15.4	Service	Services supported by the node

Table 1: Attributes in the attribute certificate.

Machine identity certificate. Before the node's services can communicate with other services, they need a valid machine certificate. A resource administrator on a registered node that has no machine identity certificate or has an expired one, needs to request for a new certificate. This can be done by calling:

```
$ rca_request
```

The script has the RCA client create a new private key, sends the corresponding public key for signing to RCA server and, if properly registered in RCA DB, lists the obtained certificate contents. It creates and replaces the files containing the machine certificate and machine's private key on the node. By default they are named `/etc/xos/truststore/certs/resource.crt` for the public key and machine identity certificate, and `/etc/xos/truststore/private/resource.key` for the private key. The private key's permission to read should be applied only for the system administrator (root user).

To see the contents of the certificate, the administrator can then invoke the command shown in Figure 6.2.1. This certificate identifies the resource through the **Subject**, which describes the resource's location and organisation. The CN field of the **Subject** represents the resource's ID.

Attribute certificate. Along with the machine certificate, the services can also use an attribute certificate. This certificate is placed in the public certificate folder, and is named either `attrextcert.crt` (X.509 V3 certificate) or `attrcert.crt` (X.509 V2 certificate), as shown in Figure 21.

This certificate shares the Subject and the public key with the resource identity certificate. The extensions of the V3 certificate or the attributes of the V2 attribute certificate contain the items shown in Table 1.

VO certificates. A node can provide its resources to the jobs from any number of VOs. For each VO, the node needs to have a corresponding certificate installed. This certificate contains the machine identity's public key, and an attribute in its extention states the ID of the VO that can utilise the resource.

```
$ openssl x509 -text -noout -in \
/etc/xos/truststore/certs/resource.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:21:1a:1a:1f:f0
        Signature Algorithm: sha256WithRSAEncryption
Issuer: O=XLAB, OU=rca, CN=xtreemtej0.xlab.si/rca
        Validity
            Not Before: May  7 07:59:28 2009 GMT
            Not After : Jun  6 08:09:28 2009 GMT
Subject: C=SL, L=Ljubljana, OU=Research, O=XLAB,
CN=Address=[://172.16.117.196:60000(172.16.117.196)]
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
[ Details excluded ]
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Key Usage: critical
Digital Signature,Key Encipherment,Data Encipherment
            X509v3 Extended Key Usage: critical
                2.5.29.37.0

[ Details excluded ]
```

Figure 20: Examining a resource certificate.

There are several means of obtaining a VO certificate for a specific VO:

- The VO administrator can add the RCA to a VO. If the node is registered with this RCA, the certificate will be pushed to the node.
- The VO administrator can individually add the resource to a VO. The certificate will be pushed to the node as well.
- The resource administrator can request the certificate from the RCA Server.

Pushing the VO certificate involves the RCA server to notify the node's running RCA Client service that it can obtain the VO certificate for a particular VO.

```
$ openssl x509 -text -noout -in \
/etc/xos/truststore/certs/attrextcert.crt
Certificate:
[ Details excluded ]
Signature Algorithm: sha256WithRSAEncryption
Issuer: O=XLAB, OU=rca, CN=xtreemtej0.xlab.si/rca
        Validity
            Not Before: May  7 07:59:28 2009 GMT
            Not After : Jun  6 08:09:28 2009 GMT
Subject: C=SL, L=Ljubljana, OU=Research, O=XLAB,
CN=Address=[://172.16.117.196:60000(172.16.117.196)]
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
[ Details excluded ]

                serial:05

            1.34.5.0.15.1:
                ..3.49700096E9
            1.34.5.0.15.2:
                ..1
            1.34.5.0.15.3:
                ..5.27433728E8
            1.34.5.0.15.4:
eu.xtreemos.system.communication.\
redirector.ServiceCallRedirector,\
eu.xtreemos.xosd.daemon.Daemon,\
eu.xtreemos.xosd.xmlextractor.XMLExtractor,\
eu.xtreemos.xosd.resourcemonitor.ResourceMonitor,\
eu.xtreemos.xosd.jobDirectory.JobDirectory,\
eu.xtreemos.xosd.resmng.ResMng,\
eu.xtreemos.xosd.security.rca.client.RCAClient,
eu.xtreemos.xosd.security.vops.VOPS,
eu.xtreemos.xosd.jobmng.JobMng
[ Details excluded ]
```

Figure 21: Examining an attribute certificate.

The RCA Client service automatically sends the VO certificate request, passing its public key. When it obtains the VO certificate, it stores it into

`/etc/xos/truststore/certs/incoming/`

with the following naming scheme:

- V2 certificates: `attrcert`*VOID*`ext.crt`
- V3 certificates: `attrcert`*VOID*`.crt`

where *VOID* stands for the ID of the VO.

The client places the pushed certificates into an ”incoming” directory in order to not have it installed without the resource administator's intervention. Therefore, for the pushed VO certificates, the resource administrator has to move the required VO's certificates into the certificate directory (`/etc/xos/truststore/certs/` by default).

To make an explicit request for a new VO certificate for a VO with ID *VOID*, the resource administrator can invoke

```
$ rca_resource_vo c <VOID>
```

For example, for VO with ID `65335f10-d113-4f7b-96a8-4d955d5d9cd2` that the resource is a member of, the resource administrator can invoke the command shown in Figure 22.

```
$ rca_resource_vo c 65335f10-d113-4f7b-96a8-4d955d5d9cd2
The RCA client received the certificate for VO
65335f10-d113-4f7b-96a8-4d955d5d9cd2. Please
check /etc/xos/truststore/certs/.
```

Figure 22: Requesting a new VO certificate.

As the command's shell output shows, in this case the RCA Client obtains and installs the certificate (Figure 23).

```
$ openssl x509 -text -noout -in \
  /etc/xos/truststore/certs/incoming/\
  attrcert65335f10-d113-4f7b-96a8-4d955d5d9cd2ext.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:21:3e:ce:7d:24
Signature Algorithm: sha256WithRSAEncryption
Issuer: O=XLAB, OU=rca, CN=xtreemtej0.xlab.si/rca
        Validity
            Not Before: May 14 11:02:48 2009 GMT
            Not After : Jun 13 11:12:48 2009 GMT
Subject: C=SL, L=Slovenia, OU=Research, O=XLAB,
CN=Address=[://172.16.117.196:60000(172.16.117.196)]
[ Details excluded ]
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Key Usage: critical
Digital Signature,Key Encipherment,Data Encipherment
            X509v3 Extended Key Usage: critical
                2.5.29.37.0
[ Details excluded ]
                serial:05

            1.34.5.0.15.1:
3.494903808E9
            1.34.5.0.15.2:
                1
            1.34.5.0.15.3:
                5.27433728E8
            1.34.5.0.15.4:
eu.xtreemos.system.communication.redirector.\
ServiceCallRedirector,
eu.xtreemos.xosd.daemon.Daemon,
eu.xtreemos.xosd.security.rca.server.RCAServer,
eu.xtreemos.xosd.xmlextractor.XMLExtractor,
eu.xtreemos.xosd.resourcemonitor.ResourceMonitor,
eu.xtreemos.xosd.jobDirectory.JobDirectory,
eu.xtreemos.xosd.resmng.ResMng,
eu.xtreemos.xosd.security.rca.client.RCAClient,
eu.xtreemos.xosd.security.vops.VOPS,
eu.xtreemos.xosd.jobmng.JobMng
            1.34.5.0.15.5:
                65335f10-d113-4f7b-96a8-4d955d5d9cd2
[ Details exluded ]
```


Figure 23: Examing an attribute certificate.

OID	Label	Explanation
1.34.5.0.15.5	VO	The ID of the VO the resource is in

Table 2: Attributes in the VO certificate.

In addition to the contents of the attribute certificate, the certificate VO certificate also contains a single VO ID (Table 2).

6.2.2 RCA guide for Site administrators

The site administrators are the authorities within an organisation who ensure that only the trustworthy nodes are active in the XtreemOS system. In principle, they need to have no access to the actual resources, but they are the contact of the resource administrators. The trusworthiness of the resources is thus related to the amount of trust the individual resource administrator enjoys.

At any point, the site administrator can examine the list of resources pending for the registration using the command illustrated in Figure 24.

```
$ rca_list_pending
Listing pending resources:
ResourceID = [IP=172.16.117.196:60000]: [hostIP={Address =
  [://172.16.117.196:60000(172.16.117.196)]},
  hostUniqueID={172.16.117.196}, operatingSystemName={Linux},
  processorArchitecture={x86}, CPUCount={1.0},
  RAMSize={5.27433728E8}, cpuLoadLast15Min=0,
  cpuLoadLast5Min=1, cpuLoadLast1Min=0]
ResourceID = [IP=172.16.117.156:60000]: [hostIP={Address =
  [://172.16.117.156:60000(172.16.117.156)]},
  hostUniqueID={172.16.117.156}, operatingSystemName={Linux},
  processorArchitecture={x86}, CPUCount={1.0},
  RAMSize={5.27433728E8}, cpuLoadLast15Min=10,
  cpuLoadLast5Min=13, cpuLoadLast1Min=17]
```

Figure 24: Examining the list of resources pending for the registration.

In the example in Figure 24, the output of the command was a list containing two nodes that are waiting for the site administrator's confirmation. The nodes' addresses are `172.16.117.196:60000` and `172.16.117.156:60000`, respectively. Each node's entry also contains a brief overview of the node's properties.

To confirm one of the resources thus listed, the site administrator has to use the resource's address:

`$ rca_confirm` *Resource_Address*
For example:

```
$ rca_confirm 172.16.117.196:60000
```

This puts the node to the list of the registered nodes, the list of which can be obtained usign the command shown in Figure 25.

```
$ rca_list_registered
Listing registered resources:
ResourceID = [IP=172.16.117.196:60000]: [hostIP={Address =
[://172.16.117.196:60000(172.16.117.196)]},
  hostUniqueID={172.16.117.196}, operatingSystemName={Linux},
  processorArchitecture={x86}, CPUCount={1.0},
  RAMSize={5.27433728E8}, cpuLoadLast15Min=0,
  cpuLoadLast5Min=1, cpuLoadLast1Min=0]
```

Figure 25: Confirming RCA Registration

At this point, the resource administrator of the node located at `172.16.117.196:60000` can request a machine certificate for the node.

6.2.3 RCA guide for VO administrators

Each RCA can be a member of any number of VOs. The RCA's membership in the VO means that the resources covered by the RCA can individually become the members of the VO. The following commands are available to the VO administrators:

- `rca_vo a` *VO_id* — add the RCA to the VO with the ID *VO_id*. This automatically adds all the registered resources into the VO.
- `rca_vo r` *VO_id* — remove the RCA from the VO with the ID *VO_id*. This automatically removes all the registered resources from the VO.
- `rca_vo l` — list the IDs of the VOs that the RCA is a member of.
- `rca_resource_vo a` *VO_id Resource_Address* — set the node denoted with *Resource_Address* in the form of *IP*:*Port* to be a member of the VO with *VO_id* for its id. If the target node is online, the result of the script invocation will be a new VO-related attribute certificate placed into the node's incoming folder.

- `rca_resource_vo r` *VO_id Resource_Address* — remove the node denoted with *Resource_Address* in the form of *IP*:*Port* from a membership in the VO with *VO_id* for its id.

6.3 VOPS

The VOPS server has a twofold purpose. In most cases, it provides other services a point for consulting or deciding regarding the policies. For instance, a service such as AEM requests for a policy related to a specific VO, user group and action in the VO. Similarly, the ADS/RSS service needs a policy filter that improves the resource selection. On the other hand, it provides the administrators a way to add and modify policies and their related rules.

In the first release, the user's interaction with VOPS was through using the shell command line, and the policies and rules needed to be passed as text structured into XACML [8]. While this is still possible, the upcoming release will support a more user-friendly mechanism, enabling also a GUI front-end.

6.3.1 Console commands for VOPS

The console commands use the API exposed through DIXI framework's API. The user needs to have rights to access the policy administration point, regulated through the user's role as stored in the user certificate. The commands are available in the XtreemOS Linux's **dixi-xati** package, and can be executed from the **xconsole_dixi** shell:

- **xlistPolicy -pid** $\langle policyId \rangle$ lists policy with specified **policyId**, which is stored in the VOPS policy storage. Policy is listed in XACML format.

- **xlistPolicies** lists all policies stored in the VOPS policy storage. Policies are listed in XACML format.

- **xaddRuleToPolicy -rule** $\langle xacmlRule \rangle$ **-pid** $\langle policyId \rangle$ adds rule which is passed as XML string in XACML format (**xacmlRule** is a path to XACML file containing XACML rule) to the policy residing in VOPS policy storage and identified by **policyId** argument. Returns rule created as a string object.

- **xremoveRuleFromPolicy -rid** $\langle ruleId \rangle$ **-pid** $\langle policyId \rangle$ removes rule identified by **ruleId** from policy with **policyId**.

- **xremovePolicy -pid** $\langle policyId \rangle$ Removes policy identified by **policyId** from policy storage

- **xaddPolicy -policy** $\langle xacmlPolicy \rangle$ adds policy residing localy on the path **xacmlPolicy** into VOPS policy storage.

- **xreloadVOPS** reloads VOPS policy storage.

- **xwritebackVOPS** writesback VOPS policy storage to directory where **policyStorage** entry in VOPS config file points to.

6.3.2 VOPS managed through GUI

Since policy management from console is not easy to perform and not user-friendly considering policies can be really complex, we have decided to support policy management from VOLife using web application (Figure 26).

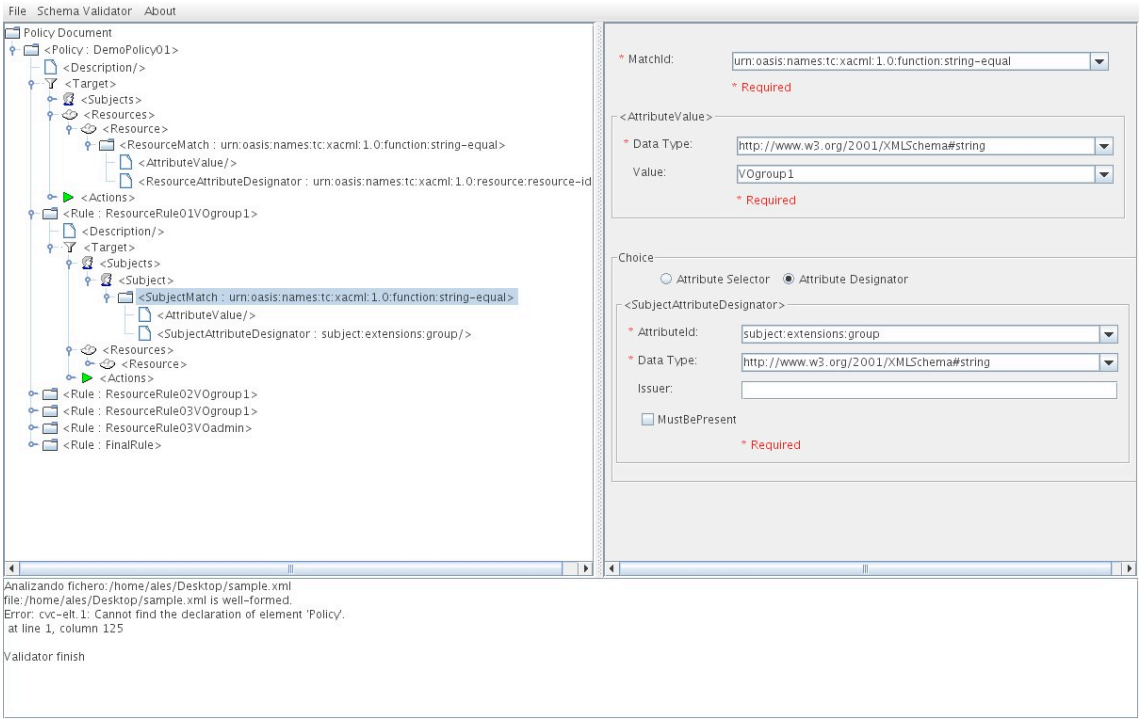

Figure 26: XACML policy editor.

In the third prototype we will provide a tool similar to the one presented in the Figure 26. VOPS front-end will provide the policy editor content of the policy which will be edited using tools provided by the VOPS libraries.

6.4 VOLife

VOLife contains two types of users: VOLife administrator and ordinary users. Aside from the functionality of ordinary users, the administrator is responsible for approving/declining the user signup requests in VOLife.

- VoLife administrator has username *admin*, and is created by the first user signup with username *admin*. The administrator's signup request is automatically approved so he can login VOLife immediately after the signup. The administrator is the first user accessing VOLife as no ordinary users can do before they are approved by the administrator in VOLife.
- Ordinary users are those with username other than *admin*. The users submit user signup requests, but are not able to use VOLife until their requests are approved by the administrator.

6.4.1 Manage VOLife users

Only the administrator can manages the VOLife users. A table listing all VOLife users is shown when the administrator selects VOLife's *Manage Users* menu. Each line of the table displays a user's detail including username, realname, status, affiliation and email. The users with pending status are those have submitted the signup requests but are not approved yet. The adminstrator can either approve or decline the requests. The approving action changes the user status to approved and thereby the user can access VOLife, while the declining action deletes the signup request and the VOLife user.

6.4.2 Manage identity

The administrator and ordinary users with approved status are able to interact with VOLife's Identity Management, which consists of the following functionalities:

- **Get an XOS-Cert** — obtains a certificate signed by a VO owner.
- **Generate new keypair** — Generates a PKI key pair and downloads the private key.
- **Change Password** — Changes the password of VOLife for current user.

6.4.3 Manage VO

The administrator and ordinary users with approved status can create and manage VOs, and send requests to join and leave other VOs. The detailed functionalities are given as follows:

- **Create a VO** — Creates a user's own VO.
- **Join a VO** — Send requests to join/leave others' VOs.
- **My Pending Requests** — Lists the current user's pending VO requests.
- **My Owned VOs** — Lists,creates and deletes the current user's owned VOs.
- **Approve Requests** — Lists, approves and declines the pending requests for joining the current user's owned VOs.
- **Manage Groups/roles** — Lists, creates, deletes groups/roles/users in the current user's owned VOs.
- **Manage Policies** — Manage VO-related policies (To be implemented).

6.4.4 Manage Resource

Resource management is available to the administrator and ordinary users with approved status. The exposed functionalities from RCA services are list as follows:

- **Register a RCA** — Creates, removes the current user's Resource Certificate Authorities.
- **Add a Resources** — Adds, removes resources to the owned RCAs, and sends requests to add the resources to the joined VOs.
- **My Pending Requests** — Lists the current user's pending VO requests.
- **Approve Resource** — Lists,approves and declines the resource requests of the current user's owned VOs.
- **Get Machine Certificates** — Obtains the machine certificates(To be implemented).

7 Conclusion and Future Work

Conclusions The second prototype of the security and VO management services provides a set of services to cater for the needs of the Grid administrator, resource administrator, and Grid end-user. It provides the minimum facilities needed to set up the root of trust (the offline Root Certificate Authority), the online CDA and RCA, the VOPS and VOLife web application.

7.1 Future Work

Reducing functionality replicated in VOWeb and CDA server Both the VOWeb front-end and the CDA server can be used to generate a user's XOS-Certificate. Currently, VOWeb calls the certificate generator library directly to create an XOS-Certificate. This requires the VOWeb configuration to include the filenames of the CDA public key certificate and private key, and also the passphrase for the CDA private key, in addition to the VOWeb owner (user 'tomcat') having read access to the CDA private key - these requirements weaken the protection of the CDA private key. Another disadvantage is that VOWeb does not record user requests for XOS-Certificates, whereas the CDA server logs the IP addresses of incoming requests, the usernames involved, and details of the certificates issued.

VOWeb could be changed to request an XOS-Certificate the CDA server using the CDA client protocol. This would have the the benefit of logging requests, and not needing to share the CDA server's private key. This change would create a new interaction between VOWeb and the CDA server.

References

[1] XtreemOS Consortium. First specification of security services. http://bit.ly/XtreemOS-D353, May 2007.

[2] XtreemOS Consortium. Second specification of security services. http://bit.ly/XtreemOS-D354, December 2007.

[3] XtreemOS Consortium. Revised system architecture. http://bit.ly/XtreemOS-D317, December 2008.

[4] Simon Godik and Tim Moses. extensible access control markup language (xacml) version 1.0. http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf, February 2003.

[5] David L Groep, Michael Helm, Jens Jensen, Milan Sova, Scott Rea, Reimer Karlsen-Masur, Ursula Epting, and Mike Jones. Grid certificate profile (draft). http://purl.oclc.org/NET/OGF-CAOPS-draft-GridCertificateProfile-v25.pdf, November 2007.

[6] R. Housley, W. Polk, W. Ford, and D. Solo. Rfc 3280 - internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. http://www.ietf.org/rfc/rfc3280.txt, April 2002.

[7] http://exist.sourceforge.net/. exist-db, an open source database management system.

[8] Commitee Specification. Oasis standard, extensible access control markup language, Avgust 2003.

[9] S. Tuecke, W. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet x.509 public key infrastructure (pki) proxy certificate profile. http://www.ietf.org/rfc/rfc3820.txt, June 2004.