Project no. IST-033576

XtreemOS

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

Trust Management in the XtreemOS Operating System

XtreemOS Technical Report # 5

Benjamin Aziz[a], Alvaro Arenas[b], Ian Johnson[c], Erica Yang[d], Matej Artač[e], Aleš Černivec[f], Philip Robinson[g], Yvon Jegou[h]

Report Registration Date: June 24, 2010

Version 0.1 / Last edited by Benjamin Aziz / June 24, 2010

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)	

[a]benjamin.aziz@stfc.ac.uk
[b]alvaro.arenas@stfc.ac.uk
[c]ian.johnson@stfc.ac.uk
[d]erica.yang@stfc.ac.uk
[e]matej.artac@xlab.si
[f]ales.cernivec@xlab.si
[g]philip.robinson@sap.com
[h]Yvon.Jegou@inria.fr

Revision history:

Version	Date	Authors	Institution	Section affected, comments
0.1	24/06/10	Benjamin Aziz	STFC	New document

1 Abstract

Trust management in large-scale distributed systems is an important enabling factor in the operation of these systems. In this paper, we discuss trust management in a Grid operating system, called XtreemOS. The paper presents the trust model underlying XtreemOS and discusses its underlying elements. We demonstrate how the trust services can be used to establish trust in a XtreemOS-based Grid and propose a couple of solutions for terminating trust. We also discuss the different alternative designs for the model and pros and cons of each alternative. Finally, we provide an overview of the design, implementation and performance evaluation of the trust services.

2 Introduction

In large-scale distributed systems, the management of information sharing requires, beside the functional solutions, solutions that can ensure that the security and dependability of information and computations are maintained. In this paper, we discuss trust management in XtreemOS (1), a large-scale distributed operating system based on the Linux kernel that aims at providing for Grids what a traditional operating system offers for a single computer: abstraction from the hardware and secure resource sharing between different users. In such a Grid-wide environment, where users and resources belong to different organisations managing different administrative domains, trust is essential in facilitating the creation and operation of Virtual Organisations (VOs) combining users and resources from various administrative domains.

According to Grandison and Sloman (2), *trust* is one aspect of belief in the competence of an entity to act dependably, securely and reliably within a specified context. Trust can be categorised into several classes among which are the *service provision trust* and the *certification trust*. Service provision trust denotes the reliance of a user on the functionality of a service, which is an essential aspect of Grid-based applications. Certification trust, on the other hand, refers to trust built on a set of certified attributes.

The XtreemOS trust model is grounded on three main aspects. First, trust is perceived as an administrative separation of resource and service ownership and management. Different organisations, resource owners, users and core XtreemOS service managers are allocated their own domains, which define their own boundaries of trust. This notion is based on the notion of service provision trust. Second, trust is asserted in special tokens created based on cryptographic mechanisms such as digital certificates and other credentials, which are then verified by their consumers. These tokens convey verifiable attributes of entities that allow them to establish trust with other entities existent in other domains. This aspect is similar to certification trust. Finally, the transmission of trust tokens is achieved via trustworthy communication channels and protocols.

The main contributions of this paper are: (i) to introduce a trust-oriented architecture for a Grid-based operating system such as XtreemOS; (ii) to present protocols and services for managing trust, combining service provision trust with certification trust; and (iii) to describe an implementation of such trust management system.

3 An Overview of the XtreemOS Grid Operating System

The XtreemOS Grid operating system results from a novel approach where the underlying operating system is extended for enabling and facilitating Grid computing. XtreemOS is based on Linux, extended as needed to support VOs and to provide appropriate interfaces for Grid OS services.

As illustrated in Figure 1, XtreemOS is composed of two parts: the XtreemOS foundation, called XtreemOS-F, and high-level Grid services, called XtreemOS-G. XtreemOS-F is a modified Linux kernel embedding VO support. XtreemOS-F is a modified Linux kernel embedding VO support mechanisms and providing kernel level process checkpoint/restart functionalities. XtreemOS-G comprises several Grid OS distributed services to deal with resource and application management in VOs, and it is implemented on top of XtreemOS-F at user level.

XtreemOS targets scalable and flexible management of dynamic VOs (3). XtreemOS Grids spans multiple administrative domains on different sites, comprising heterogeneous resources that can be shared by the participating organisations. A Grid member can create a VO, for which he becomes the VO owner. Any Grid member can request his registration in a given VO, subject to the VO owner approval. Resources can be registered in VOs as well. The VO owner defines policies stating permissions and usage rules for VO resources. Grid administrator also defines policies regulating what a Grid member can do (for example, permission to create a VO). Resources owners in the different administrative domains may also define local policies for resource usage. Grid, VO and local policies are enforce by the XtreemOS system.

The Application Execution Management services are in charge of discovering, selecting and allocating resources for job execution, as well as starting, controlling and monitoring jobs. Data management in XtreemOS is achieved with the XtreemFS Grid file system. XtreemFS federates multiple data stores located in different administrative domains and provides secure access to stored files to VO members, whatever their location.

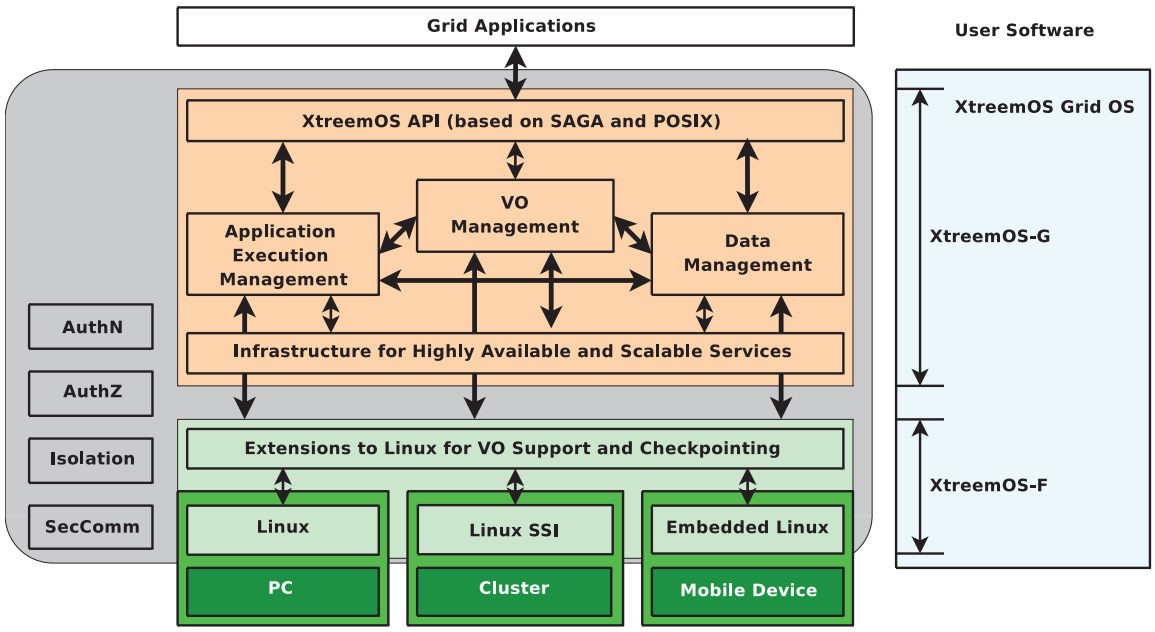

Figure 1: XtreemOS Software Architecture

4 The XtreemOS Trust Model

In the following sections, we shall discuss the administrative divisions in XtreemOS, called *trust domains*, and the different elements constituting the *XtreemOS trust model*.

4.1 Trust Domains

Domains (or sites) refer to the separation in the ownership and management of software and hardware resources as well as user membership. We use the term *trust domain* to indicate the level of assurance that each domain provides the designers, administrators and users of the XtreemOS operating system with.

In its broad definition, XtreemOS consists of three main trust domains: *Core sites*, *Resource sites* and *User sites*. Assuming that $S \rightarrow S'$ is an *assurance ordering relation* taken from some lattice of assurance levels (e.g. (4)) to indicate that S has a higher assurance level (and is therefore more trustworthy) than S', then the ordering among the three trust domains in XtreemOS is as follows: *Core Site* $\rightarrow$ *Resource Site*, *Core Site* $\rightarrow$ *User Site*. From this relation, the Core site is required to be more trusted than either the User or the Resource sites. However, no ordering exists between the User and Resource sites, since neither of these two is assumed to be more trusted than the other in a manner that can be rendered comparable.

The Core site domain is the domain in which all the core security and VO management services in XtreemOS run. A detailed description of these services can be found in (5). As a result, this domain constitutes the *root of trust* from which all other domains can be bootstrapped. Therefore, it is an important requirement for the domain to have the highest level of assurance in any XtreemOS-based Grid system.

The Core site domain may be split into two domains: The *Core site (offline)* and the *Core site (online)*, with the ordering *Core site (offline)* $\rightarrow$ *Core site (online)*. Essentially, the two domains represent two possible operation modes of the Core site. In the online mode, the domain is networked to other domains and so services and applications running remotely can access the domain and its core services. This has a lower level of assurance than in the offline case, although in practice, it is still a requirement to maintain high levels of assurance by adopting strong security protection measures. In the offline case, the domain has no network connections to any other domains, it utilises strict security measures and its services are only accessible via the interactions of authorised administrators. Therefore, the domain is considered to have the highest assurance level among all other domains.

The Resource site domain represents any domains in which resources (machines, services, software) are hosted and are connected to the Grid, therefore making them available to any VOs formed out of the Grid. The level of assurance of a resource site cannot be guaranteed.

Finally, the User site domain is any domain hosting users of the Grid, which may apply to join VOs and avail of the VO resources. Like Resource sites, User sites have no guarantees regarding their assurance levels.

4.2 Elements of the Trust Model

We now turn our attention to the main elements constituting the XtreemOS trust model. These are shown in Figure 2, and can be classified into four main categories: *Certification Authorities*, *Credentials*, *Users* and *Resources*.

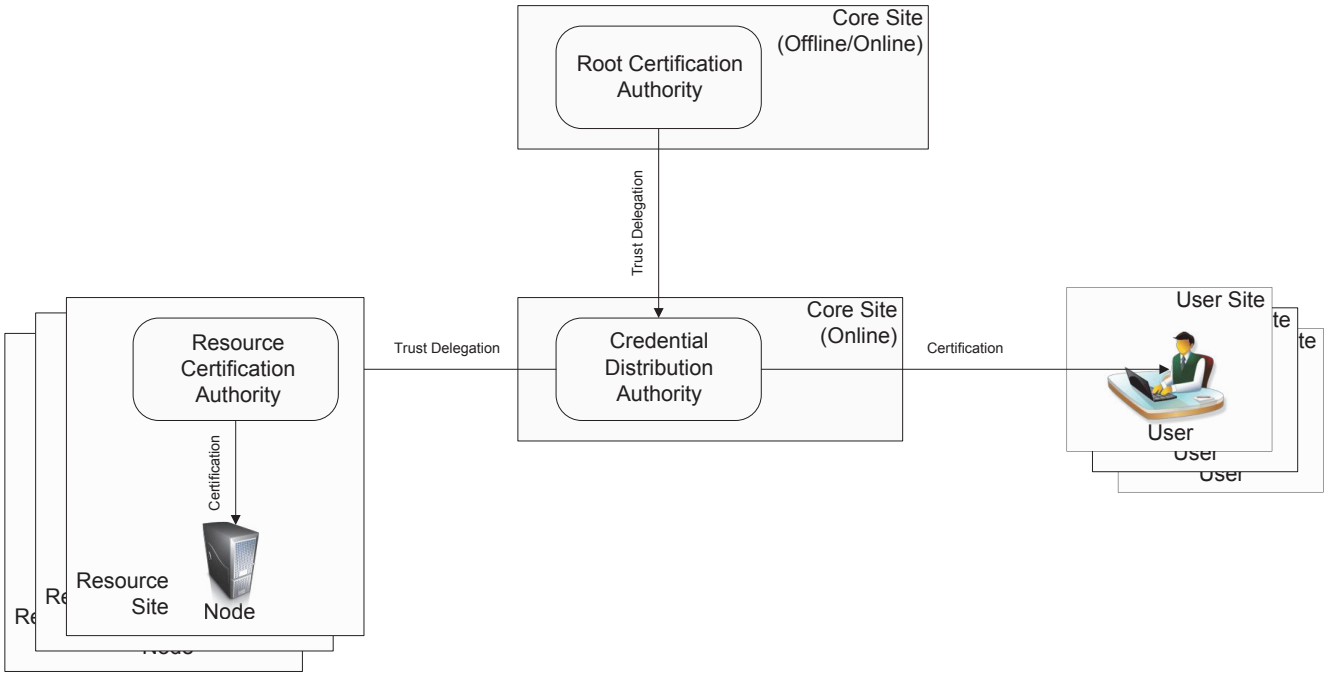

Figure 2: The XtreemOS Trust Model.

4.2.1 Certification Authorities.

Certification authorities represent points of trust from which users and resources can obtain credentials to certify their identities and/or their attributes. We define three such authorities: the *Root Certification Authority* (Root CA), the *Credential Distribution Authority* (CDA) and the *Resource Certification Authority* (RCA). These authorities are organised in a hierarchy as shown in Figure 2, where trust is *delegated* from the Root CA to the CDA and then again to the RCA. This trust delegation implies that the CDA has its public key signed by the Root CA and that the RCA has its public key signed by the CDA. Therefore, any entity that trusts the Root CA will also trust the CDA and similarly then it will trust the RCA. The Root CA itself is a self-signing authority meaning that it will sign its own public key. An important advantage of adopting separate authorities for users (CDAs) and resources (RCAs) is that one can achieve cleanly a separation of concerns between user and resource credential management. This implies that the CDAs and the RCAs can easily be maintained (e.g. upgraded or exchanged) in an independent manner as long as the format of their certificates remains compatible with each other.

The Root CA is the *trust anchor* for any XtreemOS-based Grid system, which issues identity certificates to core XtreemOS services. It also delegates trust to the CDA for the purpose of certifying users and resource sites. The fact that the Root CA operates as the trust anchor means that it provides a point of reference to the system whenever other services are compromised and their certificates need to be reissued. Therefore, it is important to ensure that the Root CA is highly protected from unauthorised accesses and is running on a highly secured machine. The Root CA operates in an offline mode. This means that any trust delegation (to the CDA) is carried out via offline means (e.g. emails, telephones, administrators' direct access using command line programs). No network connection is provided to any services or programs running on different machines. This is considered to provide a higher assurance level than if the Root CA provided online access, which would increase the risk of the Root CA being compromised (e.g. by the theft of the Root CA private key).

The CDA is a subordinate of the Root CA; the Root CA delegates trust to the CDA. This delegation of trust means that the CDA can certify the public keys of users and core services such that any entity consuming the resulting certificates will be able to trace the chain of trust up to the Root CA. The user certificate that the CDA issues also contains the user's VO attributes, such as their VO membership. The CDA serves several purposes: First, tt acts as the online certificate distribution frontend to the offline Root CA. Therefore, one can achieve a separation of concerns between the management of the Root CA's security and its online certificate distribution functionality. The CDA also permits the separation of functionality between user certification, and resource certification as performed by the RCA. To this end, the CDA delegates trust to the RCA by signing the RCA's public key. The CDA also acts as the authority enabling the formation of VOs out of users and RCAs who have been certified by a common CDA.

Finally, the RCA is a subordinate of the CDA where the CDA delegates trust to each RCA for the purpose of managing resource certification in its resource domain. The RCA certifies the public keys of resources belonging to its site. Additionally, it also issues attribute certificates required for those resources, such as certificates stating the storage capacity, speed of processors and assurance and QoS levels of the resource. The RCA facilitates the management of the resource certification process within each resource site in the Grid. It also relieves the CDA from the task of certifying each individual resource in the Grid, which would require information about the resource only available to its local administrators.

4.2.2 Credentials.

Digital credentials are pieces of data held by the different entities that provide some information about them. Credentials are usually cryptographic in the form of X.509v3 certificates (6). We call XtreemOS user certificates simply *XOS-certificates*, certificates issued to XtreemOS services *service certificates*, and certificates issued by the RCA, *resource certificates*. The certificates in XtreemOS take two forms:

Identity Certificates. These are certificates that enable their consumer to cryptographically validate the binding between the identity of the certificate's holder and its public key. This binding is important as it allows the consumer in the future to validate the authenticity of any information signed by the certificate holder. In XtreemOS, identity certificates are issued to all entities in the Grid and therefore, they constitute the most essential trust mechanism without which entities cannot participate in VOs.

Attribute Certificates. Technically, these are similar to the identity certificates except that they have an additional purpose. Instead of just binding the entity to its public key, attribute certificates have a field enumerating all the attributes of the entity. Such attributes may include the role of the user in the VO or the computational power of the resource. In XtreemOS, attribute certificates carry the attributes in extension fields to the X.509v3 certificate format (6).

4.2.3 Users.

Users are either humans or software that interact with the XtreemOS system and utilise the Grid resources within well-defined VOs.

4.2.4 Resources.

These are the individual machines, or nodes, that offer services such as computational cycles and storage space to the Grid users. A resources is managed by a *resource administrator*, who could also be the *site administrator*.

5 Setting-up Trust

Setting-up trust in XtreemOS consists of a few processes leading up to an operational XtreemOS Grid from which VOs can be formed. These various processes are described in the following sections.

5.1 Setting-up the Root CA certificate

The Root CA is run on an offline non-networked machine part of the Core site domain. The Grid administrator will run a command line program to generate the Root CA's private key and self-signed Root CA public key certificate. the Root CA's public certificate is then transferred to a networked machine for distribution to all nodes in this Grid. The Root CA will also create a certificate for the CDA.

5.2 Setting-up Core Services

The CDA creates service certificates for the core services. A CDA client program running on a core service generates the private key for that service and then uses it to sign a Certificate Signing Request (CSR), which contains the service's public key and other requested attributes, such as the type of the service and some descriptive text. The type of the service is defined by a constrained value representing any of the XtreemOS services and is checked by the service's clients during a Secure Sockets Layer (SSL) (7) handshake. There is an option for the CDA server to not automatically process all or any requests for service certificates. In this case, pending CSRs are stored in a special database for later manual processing by the Grid administrator, using the CDA's private key to create the service certificates. This option allows the Grid administrator to apply different levels of trust to CSRs on the basis of the identity of the requestor.

5.3 The XtreemOS User Registration Process

This process represents the entry point for users who wish to use the resources offered by a XtreemOS-enabled Grid. It is, in some sense, a pre-authentication step to what will follow. XtreemOS users will normally apply for an account in an XtreemOS Grid through a Web interface. This allows the applicant to enter their account details (username and password), and contact details (such as organisation and e-mail address). The following steps describe the process:

- The user accesses the Web registration page through their Web browser. To protect the confidentiality of user account details, this should only be accessed over an *https* connection secured with SSL[1]. The user submits an application to join the Grid.
- The Grid administrator views the application and may wish to obtain more details about the user in order to verify the authenticity of their request. This communication between the Grid administrator and the applicant may be offline in the form of email exchanges, phone calls, or even face-to-face meetings.
- If the Grid administrator approves the request, the user is informed via email and can then start using their account in the Grid. Otherwise, the request is rejected and the user is notified.

The process above is similar to the vetting required to join an organisation. The manual vetting of applicants ensures that they are trustworthy to starting using this Grid. The Grid administrator has the option, when considering a registration request, of applying a level of scrutiny to the registration applications appropriate to the level of security and assurance required in their Grid. Once the user has had their application approved, they can use the Web interface to join existing VOs or to create their own VOs. The user can then request an XOS-certificate certifying their identity and containing their VO attributes.

5.4 Obtaining a user XOS-Certificate

The main aim behind this process is to allow the users to be certified by the CDA, which will allow them to start creating VOs and to use VO resources. More concisely, it allows the users to enter the operational mode of a VO.

In this process, the user can obtain their XOS-certificate from either the Web interface or from the command-line CDA client program. In both cases, the underlying protocol used is the same and takes place over a SSL-encrypted and authenticated channel, where we denote by $[A \longrightarrow B]$ secure communications from A to B:
1. $[U \longrightarrow C]$: *username, password*
2. $[C \longrightarrow U]$: *status*
3. $[U \longrightarrow C]$: CSR_U
4. $[C \longrightarrow U]$: $(< \mathit{cert}_U >_{SK_C}, < \mathit{cert}_C >_{SK_R})$

In step 1, the user, U, sends their username and password to the CDA server, C. The CDA server then returns in step 2 the status corresponding to the authentication of the user. If this authentication status indicates that the user has been authenticated, then the user proceeds by sending a Certificate Signing Request (CSR_U) to the CDA server in step 3. This CSR contains the user's public key and some optional request attributes, all signed with the user's private key. The CDA server authenticates the CSR by checking the signature, then creates an XOS-certificate containing the user's public key and his VO attributes in extension fields. In step 4, the CDA server sends back to the user a certificate chain consisting of the XOS certificate of the user, $< \mathit{cert}_U >_{SK_C}$, signed by the CDA's private key, SK_C, and the CDA's own certificate, $< \mathit{cert}_C >_{SK_R}$, signed by the Root CA's private key, SK_R. At the end of the protocol, U can demonstrate in a verifiable manner its own identity certified by the Root CA via the CDA.

5.5 The Resource Certificate Distribution Process

During this process, the RCA, R, aims at obtaining a root certificate and an identity certificate from the CDA, C, through the following steps:
1. $[R \longrightarrow C]$: CSR_R
2. $[C \longrightarrow R]$: $(< \mathit{cert}_R >_{SK_C}, < \mathit{cert}_C >_{SK_C})$

where in the first step, CSR_R is a request for certificate signing sent from the RCA to the CDA, $< \mathit{cert}_R >_{SK_C}$ is the RCA's identity certificate signed by the private key of C, and $< \mathit{cert}_C >_{SK_C}$ is a self-signed root certificate issued and signed by C.

5.6 Machine Certification by Local RCAs

In general, machines need to register with at least one local RCA securely. Because machines are operated within the same administrative (trust) domain as their RCA, the problem of establishing a secure channel between a machine and its RCA is

[1] We mandate the use of the most secure TLS protocol available. Although the term "SSL" connection is used, we not use the obsolete SSL protocol (8).

resolved locally within the domain. This will depend on the level of security and assurance adopted in the domain. Therefore, we do not describe here how a secure channel (if needed) is obtained between a machine and its RCA, since this is a local issue.

6 Termination of Trust

The termination of trust in XtreemOS takes the form of certificate revocation. This feature will be included in the next release of the XtreemOS software.

One classical solution to revocation consist in using certificate validation protocols such as the *Online Certificate Status Protocol* (OCSP) (9), which has advantages over classical Certificate Revocation Lists (CRLs), as it provides timely information on the status of the certificates of different users and resources. However, since it is a server-based solution, it can create a central point of failure. Therefore, we plan to implement a more scalable solution.

XtreemOS provides a highly scalable and available Publish/Subscribe (Pub/Sub) service, used by XtreemOS services to notify other services and users about important, possibly time-critical, events within the XtreemOS operating system. Such events may include the termination of user jobs, updates in the file system and availability of new VO resources. Our solution is based on using the Pub/Sub service to distribute events about the validity of certificates. The Pub/Sub service will publish a topic called *certificate status* and any service or user interested in the topic will receive timely updates on the status of certificates.

7 Trust Model Flexibility

One of the main features of the XtreemOS trust model is its flexibility regarding the configuration of the various services and the relatios among them. This means that several alternative settings are possible, which we discuss a few of them below.

In the first alternative setting, multiple CDAs under a single Root CA are set-up. The main advantage of this model is that it permits the division of the Grid into multiple domains each with its own CDA. One possibility is that each domain corresponds to a single VO. The CDA domain will also be responsible for the certification of its own users, RCAs and resources. This will facilitate the scalability of trust and improve fault tolerance. The management of CDAs, for example upgrading them, will not result in the blocking of the certification of RCAs and users belonging to other different CDAs. Moreover, the model encourages elastic provisioning of resources since it is possible for users certified by one CDA to utilise resources certified by another CDA on demand since both obtain trust delegation from a single Root CA. Nonetheless, as a result of the presence of multiple CDAs, a man-in-the-middle attack can be mounted by a malicious user who has already compromised a CDA. In this attack, the malicious user will simply set-up and run a fake CDA using the stolen private key, which can then start certifying his own group of users and possibly other fake RCAs. This may lead to attacks on users and resources in other CDA domains. This scenario is not possible with a single CDA since it would be altogether disallowed.

The second alternative depicts a Grid divided into multiple Grid domains each running their own root of trust. This multiplicity at the root requires the definition of the relationship among the different Root CAs. In this case, the model provides the possibility of one Grid to trust the root of another by simply allowing the CDA certificates of the former to be signed by the Root CA of the latter. This model has the advantage of providing more granularity at the Grid level regarding the management of trust, and at the same time, it is possible to allow users from one Grid to utilise resources belonging to another. It also means that the upgrading of one Root CA will not impact trust in the domains of other Root CAs. However, similar to the man-in-the-middle attack in the previous model, in this model, a malicious user who's able to compromise the Root CA of some Grid domain will be able to set-up a fake Root CA and masquerade the Grid domain belonging to that Root CA. This will trick users and resources in other Grids to believe that a Grid exists whose root of trust is the malicious user.

The last alternative model we discuss here is one in which there are multiple Root CAs where some of them have *cross-certified* one another. Cross-certification means that a Root CA will certify (by signing the public key of) other Root CAs it trusts, and vice versa. The model has similar advantages and disadvantages as the previous one, but in addition, as noted in (10), cross-certification can have negative implications in the compatibility of levels of assurance in the cross-certified domains. For example, a "high level of assurance" classification may have different meanings in different domains, which are not necessarily compatible. Also, cross-certifying a low-assurance domain with a high-assurance one leads to the reduction of the level of assurance of the latter.

8 Implementation and Evaluation of the Trust Services

Here, we provide an overview of the design and evaluation of the CDA and the RCA.

8.1 The CDA Design

We describe first the classes constituting the CDA and the interactions among those classes. The CDA design comprises clien-side and service-side software. The client-side software is used by users or RCAs to interact with the CDA server. The CDA client consists of two main classes: the *CDAClient* and the *PeerChecker* classes The CDAClient class has two main purposes:

First is to authenticate a user and second to allow that user to send certificate requests to a CDA server. The *PeerChecker* class is used to allow the CDA client to authenticate the CDA server it is connecting to.

On the other hand, the CDA server consists of two classes: the main CDA engine class called *Engine*, and a helper class called *VOService*. The *Engine* class is the main CDA engine, which is responsible for issuing XOS certificates, whereas the *VOService* class is used to authenticate users before they can use the engine.

8.2 The RCA Design

The design of the RCA follows along the same lines as the design of the CDA above. The RCA consists of client-side and server-side software. The client-side software is represented by the class *RCAClient*, which provides an interface with the RCA server. It encapsulates the class *RCAClientProcessor*, which implements the functionality of storing, installing and retrieving the appropriate machine's certificates, generating the machine's key pair, and saving the result obtained through the front-end from the RCA.

On the other hand, the RCAServer is the main class at the server-side and it encapsulates methods for the RCA's functionality, implemented by the depending classes *RCAServerProcessor* and *ResourceRegistration*. The *RCAServerProcessor* class represents the functionality of storing and manipulating the resource registration entries and query result retrieval. It also maintains the list of VOs that an RCA is contributing to. Finally, *RCAServerProcessor* implements the core functionality of the RCA, i.e. the capability to compose certificates for the resources and to sign them using the RCA's service private key. It provides a method for each type of the certificates that might be requested for signing by the resource (machine identity and attribute certificates).

8.3 Evaluation of the CDA Service

The main performance bottleneck in the XtreemOS model is the CDA, since it is the most centralised point in the model. Therefore, a performance test was carried out to understand how effective this service is. Figure 3. This figure shows the CPU

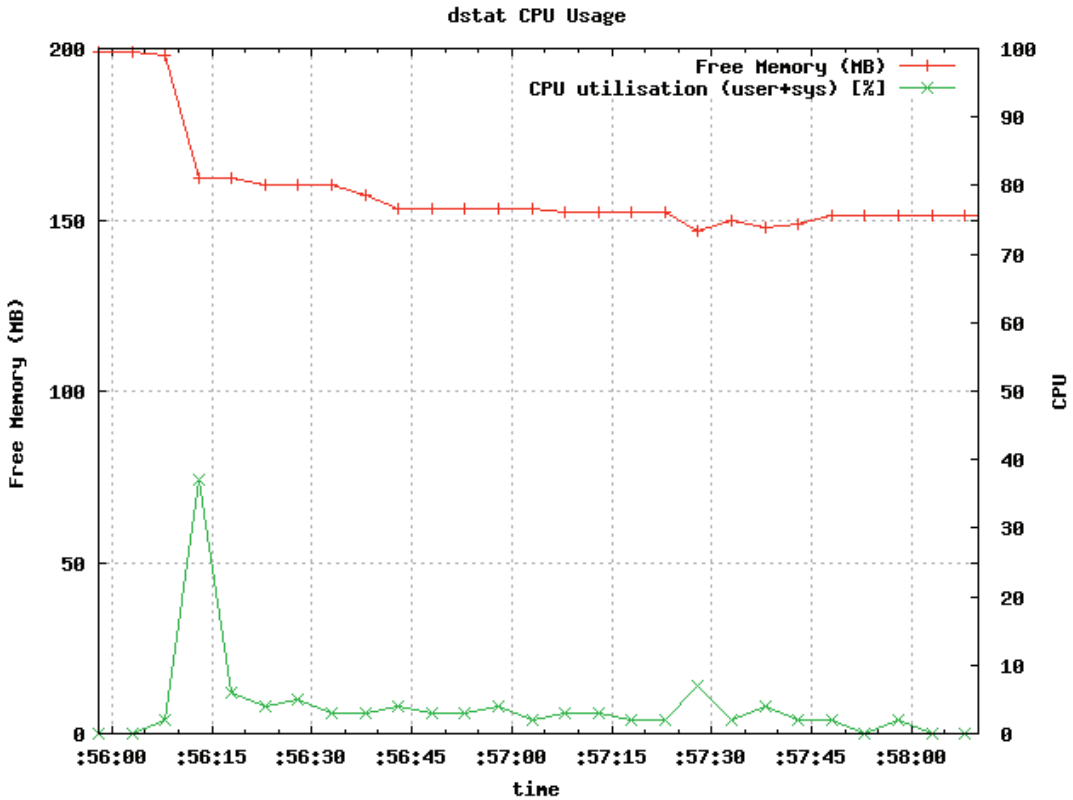

Figure 3: The CDA Test Results.

load (right axis) and reduction in free memory (left axis) while the CDA server is processing 100 client requests. Before the test was started, CPU utilisation was around 3%. The CPU load while the test was running was around 8%, apart from an initial spike of around 40%, which may be associated with the CDA server servicing the first request (involving loading classes and making the initial database connections). This represents an additional CPU load of 5% due to the CDA server processing a single stream of requests. On the other hand, The amount of free memory before the test was around 200MB. During the test, this reduced to around 150 MB. Hence, operation of the CDA server during this test consumed about 50MB of extra memory.

9 Related Work

The main example of certification trust in Grids is provided by the Grid Security Infrastructure (GSI) (11), which defines a standard service-oriented architecture for securing Grids based on a PKI and leveraging from Web services security. In GSI, trust roots include CAs responsible for binding names to public keys in signed certificates, enabling secure (authenticated, private) communication between named parties. CAs are responsible for certifying both people and services, and the set of trusted CAs determines which certificates for those people and services can be trusted. To manage certificates and trust root configurations, GSI used the myProxy credential management service (12). GSI has strongly influenced the design of our trust management, but there are some differences. Some of the CDA functionality is comparable to myProxy, but XtreemOS caters for the possibility of having multiple CDAs and potential replication of them. In order to achieve modularity and scalability, XtreemOS separates management of resource certificates, achieved via RCAs, from certificate management for users and VOs, achieved via CDA.

The GridTrust project implements trust bootstrapping in Grids via the notion of Virtual Breeding Environments (VBE) (13), infrastructures facilitating the creation of VOs from clusters of organisations willing to collaborate with each other. Each VBE has associated a CA, which caters for certificate management for resources and users, both within the VBE and the participating VOs. As mentioned before, XtreemOS separates certificate management for local resources from certificate management for users and VOs.

Globe is a wide-area object-based middleware that includes a plugglable trust management module (14). Third trusted parties are used to certify the association between the symbolic object name, which identifies the service provider to the user, and the object ID, which represents the object in the digital world. Associated with each Globe distributed shared object, there is a public/private key pair cryptographically bounded to the object identifier, similar to the self-certifying file names. Once trust has been established between the users Globe runtime and the objects key, this can be used to bootstrap trust relationships. This is similar to XtreemOS PKI, but it assumes that CAs can be third trusted parties.

10 Conclusion and Future Work

In this paper, we presented the trust model for XtreemOS; a large-scale European Grid operating system based on the Linux kernal. We discussed the different processes involved in setting-up trust among XtreemOS services, users and resource providers. We also highlighted options for terminating trust using certificate validation protocols and pub/sub services. Finally, we presented a few alternatives to the model and discussed their pros and cons.

One of the major future research direction is related to the applicability of XtreemOS in the area of Cloud computing as envisaged by (1). This new playground for XtreemOS will pose new challenges to the trust model, mechanisms and services. Among these will be the managing credentials in Cloud federations and ensuring secure and trustworthy isolation among users and resources belonging to different Cloud domains.

Acknowledgements

This work was funded by the European FP6 project XtreemOS under the EC contract number IST-033576.

References

[1] Morin, C., Jégou, Y., Gallard, J., Riteau, P.: Clouds, a new playground for the xtreemos grid operating system. Parallel Processing Letters (PPL) **19**(3) (2009) 435–449

[2] Grandison, T., Sloman, M.: A Survey of Trust in Internet Applications. IEEE Communications Surveys and Tutorials **3**(4) (September 2000)

[3] Coppola, M., Jégou, Y., Matthews, B., Morin, C., Prieto, L.P., Sánchez, O.D., Yang, E., Yu, H.: Virtual Organization Support within a Grid-wide Operating System. IEEE Internet Computing **12**(2) (March 2008) 20–28

[4] Denning, D.: A lattice model of secure information flow. ACM Transactions on Programming Languages and Systems **19(5)** (May 1976) 236–243

[5] XtreemOS Consortium: Fourth specification, design and architecture of the security and vo management services. In: XtreemOS public deliverables - D3.5.13, Work Package 3.5 (December 2009)

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

[7] Dierks, T., Rescorla, E.: Rfc 5246: The transport layer security (tls) protocol version 1.2 (August 2008)

[8] Freier, A.O., Karlton, P., Kocher, P.C.: The ssl protocol version 3.0 (November 1996)

[9] Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: Rfc 2560: X.509 internet public key infrastructure online certificate status protocol - ocsp (June 1999)

[10] Canadian Institute for Health Information: Cross-Certification Guidelines: National PKI Framework for Health (2001)

[11] Welch, V., Siebenlist, F., Foster, I.T., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., Tuecke, S.: Security for Grid Services. In: 12th International Symposium on High-Performance Distributed Computing (HPDC-12 2003). (2003) 48–57

[12] Basney, J., Humphrey, M., Welch, V.: The myproxy online credential repository. Software Practice and Experience **35**(9) (2005) 801–816

[13] Naqvi, S., Massonet, P., Aziz, B., Arenas, A.E., Martinelli, F., Mori, P., Blasi, L., Cortese, G.: Fine-grained Continuous Usage Control of Service-based Grids – The GridTrust Approach. In P. Mahonen, K.P., Priol, T., eds.: ServiceWave 2008. (2008)

[14] Popescu, B.C., Crispo, B., Tanenbaum, A.S., Bakker, A.: Design and Implementation of a Secure Wide-Area Object Middleware. Computer Networks **51**(10) (2007) 2484–2513