Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

DS2OS - Deliverable A
Project Overview and Functional Requirements

Giovanni Schmid

RT-ICAR-NA-2010-02 **Novembre 2010**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

DS2OS - Deliverable A
Project Overview and Functional Requirements

Giovanni Schmid[1]

Rapporto Tecnico N.:
RT-ICAR-NA-2010-02

Data:
Novembre 2010

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli

DS2OS - Deliverable A
Project Overview and Functional Requirements

Giovanni Schmid

November 10, 2010

Abstract

In the next decade, the Internet web will became more and more integrated in our physical and social environments, and both high-speed and wireless connection will proliferate, causing an increasing demand for IT architectures which are targeted to manage the interactions of many people among them and with different Institutions, Organizations and Enterprises in order to get fun, services and information at a faster rate and lower costs.

A key point for these environments and virtual communities will be the ability to manage very complex access control and resource usage scenarios, with no loss in performance, reliability and security. Because of interoperability and scalability constraints, such architectures cannot rely upon centralized trust and authorization models, but - conversely - they must implement distributed access control models which enable users to share effectively and efficiently computing resources and facilities, applications and data.

The **Distributed Security-Oriented Operating Systems** (DS2OS) Project concerns the design of an open access control infrastructure upon which next-generation distributed computing environments can be effectively and safely built. Specifically, this project aims to implement a such kind of architecture for the SSH system entry service, deploying such implementation in the context of a "proof-of-concept" testbed.

This document, named **Deliverable A**, represents the first of a series of project's deliverables describing the the overall DS2OS architecture with its functional requirements and design specification. After illustrating reasons and inspiration which led to DS2OS Project, **Deliverable A** aims to describe DS2OS goals and specify its functional requirements. Moreover, it gives an overview of DS2OS architecture, sketching its main components and their mutual relationships. As such, this document is also intended as a "reading key" for other project's deliverables, illustrating the overall organization of DS2OS specification.

1 Introduction

Current Internet web usage scenarios (e.g. social networks, file sharing communities, virtual computing organizations, open software development communities) are rapidly changing scopes and usages of networking, announcing the era when Internet will result in a collection of service-oriented, user-centric, large-scale distributed and ubiquitous computing environments which can evolve and adapt their features not only because of their management staff decisions, but also in function of the users' needs and the behaviours of (some of) the items managed in such environments. Indeed, as pointed out in [11], many networks based on Internet technologies are experiencing the so-called "network effect", where an increasing

number of on-line services attract increasing numbers of users, attracting further online availability of information and services. The meaning of the term "on-line service" itself is quickly changing: only few years agò, we used it to denote just a (single) computing facility offered by a single host on a network (e.g a server web, an FTP repository, an SSH connection); now, it is more often employed to refer to a coordinate set of computing activities performed on different hosts of a network with the scope of realizing a comprehensive and useful task from the viewpoint of an end-user (user-friendliness here plays a crucial role). *Web services* [10] represent particular instance of the above concept (paradigm), where the focus is on implementation technologies (i.e. web technologies); *grid services* and *cloud services* are other examples, focused at the servicing hardware and software infrastructure, instead. In any case, we refer to distributed computations which could require the access and/or the management of computing resources and facilities scattered across a network. The resources involved in such computations can be both logical and physical, are not application-specific, and are in general very different in nature; they might be executables, user profiles, devices, data, cpu cycles, disk space, etc. And a potentially high complex mix of those resources might have to be managed during a computation.

Ideally, we would rely on architectures that can allow those computations to be efficiently and securely run in open networking environments, in a seamless and transparent way both for their providers and consumers. These requirements drive forcibly toward **Access control frameworks (ACF)** which are far more open, scalable, dynamic, user-driven and fine-grained than the ones deployed so far.

1.1 Drawbacks of current ACFs

The huge research and technical efforts which have been developed in the last decade in the field of distributed systems, culminating in Grids and Clouds deployments, resulted in some ACFs that can effectively manage very complex policies, scaling well to a huge number of users and a consistent set of services [1, 2, 4]. However, these systems exhibit one or more of the following characteristics which, in our opinion, adversely affect their adoption in the evolution for the Internet sketched in Section 1.

- **Enterprise-centricity**: nowdays ACFs for distributed computing environments are targeted to centralized, enterprise-level organizations (e.g.Cloud computing platforms) or at most a federation of such organizations, like in computational Grids. In a federated organization settlement, a loosely coupled set of cooperating entities set up cross-organizational trust allowing one participant organization to directly provide services to entities registered at another organization member of the underlying federation. However, there is still a lot of centralism here, since user collaborations are defined and deployed mostly at the federation level, whilst user-centric, ad-hoc collaborations are hardly supported.

- **Application or organization specificity**: actual frameworks are implemented at the application layer or at the organization layer, as middleware. One of the main motivation of these approaches is the need of having OS-platform independent systems, which is a must because of the different OS platforms on the market. However, these solutions are designed and implemented as service- or environment-oriented framewoks rather than general-purpose, infrastructural software. This inhibits the convergence toward worldwide accepted open standards, and results in component solutions that do not form an integrated infrastructure, undermining global interoperability and acceptance.

- **Coarse-grain resource control**: Although with significative differences among them, actual systems are generally not designed to provide the highest fine-grain access control to resources, and there is often a gap in accessing a resource through these systems instead of accessing it via the access control monitor built into the application or the operating system which directly manages it.

  Many frameworks, likewise previous generation operating systems, do not properly separate the identities of resource requesters from their authorizations. This results in an "all-or-nothing" and "context-unaware" policy, in which an access to a given resource is granted or denied only on the basis of the identity of the resource requester. Confusing identities with authorizations was a major historical misconception in access control theory that has adversely affected the evolution of systems for a long time, with remarkable consequences on many actual implementations.

  More advanced systems discriminate between identities and authorizations, resulting in a much better and flexible control. However, such systems usually realize a trade-off in matching with the different local access control monitors they have to interact. Thus,they are usually unable to reach the same level of control which can be achieved by some new generation applications and operating environments.

1.2 A possible alternative

The root cause of the above adversing features is that nowdays ACFs were designed focusing on environments built from legacy operating systems and legacy Internet protocols and services, by just adding ad-hoc, distributed-oriented application software and middleware. This approach has originated by the fact that complex distributed environments are never constructed from scratch, but are obtained as a coordinated resource sharing among existing collections of individuals, institutions, and resources. After all, one of the key factors in the worldwide adoption of the Internet web is that it builds on existing computing infrastructures.

However, this reasoning fails to catch two other key factors which have made possible the convergence toward Internet as *the* global computer network:

1. an *infrastructural approach* to its design, where the focus has been on the evolving specification - in a comprehensive international standardization process - of a set of communications models and technologies, which are *platform-independent* and *application-unaware*;

2. a *layered architecture* for its implementation, in which a *tight interfacing between the infrastructures at (1) and host operating systems* has been realized as OS kernel components and userland libraries.

As a result of this approach, standalone operating systems have over time evolved into networking-enabled platforms which integrate the link, internet and transport layers of the **Internet protocol suite**[1] in their kernels, offer some core client-side network applications as APIs (e.g. the client-side of DNS[2]. Other major information services (e.g. NIS, NIS+)

[1] A.k.a. (somewhat improperly) **TCP/IP suite**, a set of IETF standard protocols for communicating over the Internet, whose basic requirements for host system implementations are described in [14, 15]. The link, internet and transport layers are described in the first of such documents.

[2] **Domain Name System (DNS)**, a globally hierarchical and distributed naming service whose client-server style protocol is an IETF standard [25, 26]. Although DNS is positioned at the application layer in the TCP/IP stack, DNS clients are actually implemented as the *resolver* library in all POSIX-compliant OSes

or directory services (e.g. LDAP) are also tightly coupled with some modern operating systems, because of the relevance these services have played (and, in same cases, - as we are going to show - will play) for networking and distributed access control.

A design driven by both points (1) and (2) as guideline principles is - in our opinion - the right way for approaching and successfully getting in the Internet computing era. The latter is indeed just a next step in the long evolutive roadmap of the Internet; so, why should it not be driven by the same approach and strategies successfully adopted so far?

In order to solve the resource's access issues posed by next generation computing environments we do not need some more gridware, cloudware or whatever else environment-specific new software to "glue together" legacy OS's access control mechanisms at the network layer. This inevitably results in poorly interoperable, somewhat outdated frameworks.

What we need is - given any operating system - the best, closest matching between its local access control subsystem and a general-purpose, open-standard access control framework which fits in with the actual set of Internet's infrastructure technologies and augments them, in a similar way the DNS did for the first generation Internet.

Conversely than the"legacy software plus ad-hoc middleware" approach, an infrastructural one - if designed in a modular fashion - can provide for a solution to the access control problem which does not rely upon specific network topologies or usage scenarios, and will comply with any future evolution of the Internet.

2 Technologies background

If we look at the core architecture supporting Internet web, then we found that it is just composed of the following three basic elements[3]: a set of protocols for raw data transmission over different (physical and logical) kind of networks, the DNS service, and the **Hypertext transfer protocol (HTTP)**[4].

Since the Web is what end users are looking at, in the last years HTTP is acting as a main attractor, a sort of "center of gravity" in the field of networking and distributed computing. With this trend, which many expert in the field are indicating as the advent of what they call the *Web-computing Era*[5], it is not strange that nowdays access-control research efforts and technical developments are almost exclusively focusing on web services and their backend stack of software, pointing once again to the application layers and forgetting the importance that operating system security mechanisms play in supporting security at higher levels.

About twelve years agò, researchers at the National Security Agency recognized that "security efforts suffer from the flawed assumption that adequate security can be provided in application with the existing security mechanisms of [that time] mainstream operating systems" [23]. That work urged the need for more advanced security mechanisms in operating systems, spawning a renewed interest in the field.

From that time, operating systems technology has greatly evolved, and nowdays we dispose on the market of products with very sophisticated protection subsystems, which can

[3] Of course, the nowdays so-rich web experience would not be possible without a plethora of other protocols and services; however the truly infrastructural elements are the ones cited here

[4] An application-layer protocol for distributed, collaborative, hypermedia information systems which at the present time is a IETF draft standard [6].

[5] The reader should have noticed that in our referring to next generation distributed computing we have preferred the term "Internet computing" to "Web-computing". And indeed, for reasons which should be more clear as a consequence of the following discussion, we believe like P. Vixie that, though the Web has changed the rules, "...the Internet was here before the Web and will be here after the Web and is much larger than the Web..."[34].

enforce sinergically different access control policies and offer much more fine-grain access control to resources than previous systems. Indeed, the majority of general-purpose operating systems dispose now of mechanisms to enforce a *role-based* (RBAC) policy over the conventional *discretionary access control* (DAC) mode. Some systems support *mandatory-style* access control (MAC) policies as alternatives to DAC for highly protected environments. Finally, there are very advanced systems in which a user security context is built not only on the basis of a set of user-related identifiers, but thanks to an additional set of authorizations which affect the way applications and processes are run by that user [24]. In these systems authorizations can be enforced not only through permissions and ownerships of files, allowing for an unprecedent control over user's actions and resources.

In the meantime, we have assisted to a strong evolution and deployment of technologies for the sharing of access control related information over networks, too.

As it should be clear, the previously sketched basic infrastructure of the Internet does not encompass nor authentication neither authorization. Indeed, Internet was born when networking was restricted in the context of local area networks (LANs) only, and consisted basically of remote logins and file transfers. At that time, the usage of Internet itself was limited to email and a primitive form of what we are now experiencing as "web browsing". Authentication information was stored locally, in the systems who managed the resource under request, or at most in LAN information servers, using NIS (later, although with not so much success, NIS+ was proposed as the NIS successor); whilst authorization information was directly coded into file systems, in the form of file access control modes.

Later, the increasing both in number and sophistication of networking applications, and the simultaneous emerging of database systems, gave rise to application-specific access control monitors. The main motivation for this approach was decoupling access control to specific services from that to the operating system itself, a suitable mean to get more secure and efficient service's platforms when the hardware costs and the absence of "off-the-shelf" virtualization technologies make the paradigm "one OS for many applications" the rule of thumb.

In the meantime, the financial and commercial demand pushed the emerging of international open standards for cross-organization authentication and data protection. Large-scale, remote authentication had indeed became technically possible by asymmetric cryptography and the introduction of public-key certificates (PKCs) [12], which allow authentication information to be available and checked through public directory servers and protocols[6].

In 1988 was published edition 1st of ITU-T X.509 [16] (formerly CCITT X.509), also published as ISO/IEC Standard 9594-8, as part of X.500 series of computer networking standards covering electronic directory services. It defined *public key infrastructure* (PKI) as a strict hierarchical system of certification authorities for issuing public-key certificates, in contrast to the web trust model - already implemented at that time in PGP [37] - where any one can attest the validity of other's public key certificates. Although the X.500 system has never been fully accepted nor implemented, over the last two decades X.509 has been continuously revised and extended to support the constant increasing demand in identity management features driven by the evolution of the Internet and its usage scenarios[7]. In 1997 was issued X.509 edition 3rd [17], which redefined the PKI framework in a more flexible

[6] A **Directory** is a map of suitable identifiers (*names*) with related values, and a **Directory service** is a system that stores, organizes and provides access to information in a directory, allowing the look up of values given a name in a similar way it is realized by using a dictionary.

[7] The main obstacle in the full adoption of X.500 standard is perhaps the idea that managed entities must have globally (worldwide) unique public names. As noticed in [5], many enterprises and organizations consider valuable or even confidential their own collections of directory entries, so they do not intend to release them to the the world in the form of an X.500 directory sub-tree.

way, in order to support other topologies like bridges and meshes, used in some common communication scenarios, including the web of trust model. This edition also defined the versions for PKCs and *certificate revocation lists* (CRL) as they are currently used[8].

Basically, PKIs allow for conveying identity based access control decisions outside OS- or application-enforced security domains. Indeed, the wide adoption of PKIs during this first decade of Internet computing is because the large majority of actual distributed environments rely upon identity-based access control decisions. In these systems PKCs are used to gain corroborate evidence of the identity of a requester, by checking that she has access to the private key that corresponds to the public one in the PKC.

However, as already recognized in the context of applications and mainstream OSes, it was realized that distributed computing scenarios could benefit from advanced access control policies such as *rule-based*, *role-based*, and *rank-based* ones. These forms of access control decisions require additional information that cannot normally be included in a PKC. Although PKCs have optional fields designed to contain authorization information, the lifetime of such information is generally much shorter than the lifetime of the public-private key pair. Moreover, authorization information is often contex-dependent, whereas authentication information is not.

To overcome this main limitations, edition 4th of X.509 [18] introduced *attribute certificates* (AC) and *privilege management infrastructures* (PMI). An AC is a structure similar to a PKC, but instead of realizing a binding between a public key and identity information of the certificate's holder, it associates such identity information with attributes that may specify group membership, role, security clearance, or other authorization information. The AC format allows any additional information to be bound to a PKC by including, in a digitally signed data structure, a reference back to one specific PKC or to multiple PKCs, useful when the subject has the same identity in multiple PKCs. Additionally, the AC can be constructed in such a way that it is only useful at one or more particular contexts or targets (see Deliverable C for a detailed analysis of these issues).

Summarizing, the research efforts and technological developments observed so far in the field of access control have resulted in a two-fold achievement, one concerning OS protection systems and the other directory-based distributed architectures for conveying public credentials over insecure communication channels with integrity protection. In both cases, the achievement has derived by realizing that many access control scenarios require a sharp distinction between authentication and authorization information, conveying and managing them by mean of different data structures and flows.

However, no tight interoperability is currently provided between the security contexts which can be enforced, at the local level, through modern OSes and, at the network level, through directory services. Indeed, until now, directory-based services have usually been employed to export and manage just basic POSIX-compliant login access information; that is, usernames and passwords.

3 Problem Statement and project goals

At the heart of this project, named **Distributed Security-Oriented Operating Systems (DS2OS)**, there is the idea that ACFs suitable for the Internet computing Era have to be obtained from the confluence of state-of-the-art OS access-control related technologies with the emerging Internet standards concerning, broadly speaking, information services.

[8] The IETF's **Public-Key Infrastructure (X.509)**, or **PKIX**, working group has adapted the standard to the more flexible organization of the Internet. In fact, the term X.509 certificate usually refers to the IETF's PKIX Certificate and CRL Profile of the X.509 standard, as specified in [3].

That fulfil the two factors which in Section 1.2 we recognized as driving forces for the worldwide deployment of the Internet, and aims to fill the gaps discussed in the previous section among ACFs at the network and OS layers.

In order to comprehensively illustrate the DS2OS project we need both a precise formulation of the problem it aims to solve, and the way it intends to do that. These issues are discussed in the following sections.

3.1 DS2OS computing environments

In the following we will consider *networked* operating systems, briefly called *hosts*. The term "networked" has here a specific meaning, referring to the fact that the targeted OSes have built in all the kernel components and the APIs required for basic TCP/IP internetworking (datagram and packet transmission, client-side name resolution, and client-side directory service)[9].

We will consider arbitrary sets of such hosts, interconnected by a network of arbitrary topology. Likewise grid environments [8], we will moreover assume that for such a network:

1. A **Resource** is any (logical or physical) item which - directly or indirectly (that is, via an application layer) - falls under the control of the protection subsystem of a given host in the network. Resources are thus scattered through hosts constituting the network, in such a way that each resource is managed locally by a given host, being subjected to authentication and authorization mechanisms and policies which - with the exception of specific cases - are locally defined and have significance just in the context of multiple, local *trust domains* enforced through such hosts.

2. Generally speaking, a **Computation** is any computing task which can be performed using a pool of resources and hardware/software facilities of the network. However, we are specially interested in (truly distributed) computations that, throughout their lifetimes, are composed of dynamic groups of processes running on different hosts of the network. Processes can be created remotely and on demand by other processes, by using different computing schema such as remote invocations, process migration or cloning. A computation could thus be the result of many processes which run concurrently and/or sequentially on different hosts, communicate using dynamically created TCP/IP connections, and dynamically acquire or release resources such as data, CPU time, devices, and so on.

3. In order to perform seamlessly and trasparently computations as stated at point (2) - a **Global security policy** must integrate a heterogeneous collection of locally administered users and resources in different trust domains. The focus of the global security policy is on controlling interdomain interactions and the mapping of interdomain operations into local security policies. The integration of the global security policy with those enforced locally must be such that:

   (a) operations that are confined to a single trust domain are subject to local security policy only;

[9]The Internet protocol suite has been greatly refined and extended over time in order to overcome some limitations and encompass new services, and actually composes of more than 200 standard protocols. It is implemented, although at different extents and refinements, in most operating systems in use today, including all consumer-targeted systems. What we aim to catch with the previous definition is a minimum common denominator of protocols allowing for the mutual Internet-style communication and interaction of a wide spectrum of eterogeneous, network-enabled devices

(b) both global and local subjects exist;

(c) all access control decisions are made locally on the basis of the local security policy.

Conversely than in grids, we do not assume to have a *connectivity layer* realized in middleware and common to all the hosts in the network [7] upon which a network-wide ACF can be built. The only building blocks upon we rely to realize such ACF are the single ACFs enforced on each host by its own OS, the X.509 directory services infrastructures (PKI and PMI) and a directory service protocol (LDAP).

In analogy with clouds environments[33], we think of our environment as a megacomputer - composed by a mix of infrastructural hardware e software, virtualization technologies, web services, etc. - that can be accessed through any networked host. Differently than in clouds, however, we do not focus on infrastructure providers, and we assume that resources can be in many cases shared between users.

What detailed until now ultimately represents both a model and a framework for (distributed) computations; we will refer to it in the following as **DS2OS computing environment**.

3.2 Principals and their security profiles

According a classic definition given in [28], a *principal* is any"entity in a computer system to which authorization is granted; thus the unit of accountability in a computer system". More concisely, [31, 27] define a principal as "an entity whose identity can be authenticated". We are concerned with networks of computer systems, and we are interested in network-wide entities which can perform actions and acquire or release resources on multiple hosts in those networks. Thus we need to refine in some way the above definition. We will denote with the term **Principal** any physical entity (a human being, a software, a device) which is uniquely identifiable in the context of a DS2OS environment and whose identity can be - directly or indirectly - authenticated on each host of the environment on which it can perform actions and/or utilize resources.

Principals are therefore global entities which can be related to more than one computer system. Examples of principals are the (single) entities represented by an individual who has account profiles on multiple hosts in the environment, or by a software system composed of different modules distributed among different hosts[10].

We will suppose without loss of generality that, given an host, a principal p can have no more than a single account on it. Let $u_H(p)$ be the **Username** of principal p on host H, a platform-dependent-form string which represents the identifying coordinate of the unique account of p on H. We will denote by $a_H(p)$ the authentication token which represents the validating coordinate for the above account; the access to H as $u_H(p)$ is granted only if the accessing request presented for $u_H(p)$ is corroborated in some way by the fact that p "owns" $a_H(p)$. In the majority of cases, $a_H(p)$ is a *password*, an alphanumerical string which is supposed to be known only by p. The logical entity of H univocally represented by $u_H(p)$ is called a **User** of H. In general-purpose operating systems users are generally associated to one ore more **Groups**, collective entities that represents set of users which share access control settings.

An host H associates to any principal p successfully authenticated as $u = u_H(p)$, that is to any of its users, a set $I = I_H(u)$ of *identifiers* plus, possibly, a set $A = A_H(u)$ of *security*

[10] Multiple, identical copies of the same software must be considered different principals, however.

attributes. We will call the joint set

$$C = C_H(p) = C_H(u) = I_H(u) \cup A_H(u)$$

the **Set of credentials**, or concisely the *credentials*, of user u (or, equivalently, principal p) in H, since - as we will soon show - C constitutes u's designated entitlements for services and computing resources from H. In such way, the binding

$$p \longrightarrow (u, a) \longrightarrow C = I \cup A$$

adequately catches the intuitive notion of p's **Security profile** on computer system H.

In general, both I and A have significance just in the operating environment in which they were issued, and are platform-dependent data. In Unix-like systems, I consists in the *user identifier* (UID) and a bounce of *group identifiers* (GID). These are integer values used by the kernel to assign credentials to userland processes and to define file ownership. As consequence of the authentication process, a user is mapped univocally to a single UID and to as many GIDs as the groups to which she belongs. Then, each process spawned by that user inherits her UID and her current primary GID as its credentials to get access to system resources. Actually, this is true only for processes that are generated by running non set-ID programs: a *set-UID* (*set-GID*) program runs with the UID (GID) of the (group) owner of the program executable file. The set-ID mechanism was introduced to allow non-sysadmin users to perform some operations that, although necessary or useful in the context of their standard login sessions, requires different credentials[11]. In any case, the UID and GID inherited by a process are matched against those indicating the owner and the group owner for any file the process makes an access request for, and access is granted or denied depending on the permission settings for such owner and group owner (plus the file permission settings encompassing all users in the system).

That sketched above is the only access control mechanism implemented in legacy operating systems. Thus, for such systems A is the void set, p's profile becames:

$$p \longrightarrow (u, a) \longrightarrow \{UID, GID_1, ..., GID_n\} \ ,$$

and the *security context* for such profile can be expressed at any time by p's *capability*, a list indicating filenames and p-related file permissions for all the files in H.
Most advanced operating systems provides for sets A filled with special types of credentials, such as Solaris *authorizations*, *privileges* and execution attributes[30]. See Deliverable B for an overview of these concepts and their implementation.

3.3 Access control in DS2OS environments

As already stated, this project aims to solve the access control problem in DS2OS environments. That informally consists in defining an ACF which allows for what is established at point (3) in Section 3.1. In [11], the authors observe that nowdays networks are increasingly facing with two main scale-related problems:

i) boundary access controllers (as those implemented through perimeter firewalls) cannot easily enforce fine-grained policies, and became a bottleneck as the level of replication increases in an attempt to meet increased demand in network bandwith and processing;

[11] For example, the `man` utility - a program for accessing the documentation released with any Unix-like distribution - requires to update some configuration files which are ownen by the "man" pseudo-user and must not be modified by any other user

ii) the increasing size and complexity strains the ability of administrators to effectively manage their systems.

On such basis, they argue that access control in modern networks "must became an end-to-end consideration similar to authentication and confidentiality", outlining architectural-level requirements that favor credential-based policy management and the decentralization of both policy specification and policy enforcement.

With respect to DS2OS computing environments (see Section 3.1), we believe that a correct reformulation of that general end-to-end principle turns out in the following *access control policy requirements*. They are inspired by the guideline principles introduced in Section 1.2, and are devoted to overcome the limitations illustrated in Section 1.1.

Requirement Set 1 (AC Policy Requirements) *An access control policy should be:*

AP1 *enforced on each host via the host operating system protection mechanisms;*

AP2 *specified and administered directly by resource owners, in such a way that an overall (i.e. a network-wide) policy could result from the proper mutual integration of many locally specified policies; and,*

AP3 *stored and managed via a distributed information service obtained by integrating host repositories (files, local databases) with credential-based directory services.*

AP1 roots in the circumstance that the resources that have to be managed in our environments are not application-specific and ultimately fall under the control of an operating system. Moreover, an OS's protection subsystem is the closest layer through which this set of resources can be accessed and controlled[12]. Thus **AP1** allows for the finest-grained resource control and, at the same time, gets free of application or organization specificity. Moreover, it offers the highest (architectural) scalability, since poses policy enforcement points at their mimimum possible distance from resources.

As observed in [11], "the traditional way of handling scale at the human level has been decentralization of management and delegation of authority". **AP2** is the utmost incarnation of such paradigm, since applies separation of administration duties at the maximum possible extent, that of resource owners. Albeit the particular meaning of the term "resource owner" is specific of the computing environment being considered, in many cases, and especially in emerging computing environments, it has to be distinguished by that of "system administrator". In many cases, it is more appropriate to consider as resource owner a service/application or a standard user, instead of a system administrator. Other than addressing precisely which entity/subsystem manages the resources under consideration, this allows for an easier application of the least privilege principle.

Both **AP1** and **AP2** requirements are encompassed by what we called *dynamic delegation* (see Section 4.1). Informally speaking, dynamic delegation is an access control model which allows standard users on a given host H to authorize remote users or applications to use the resources they own as a result of their accounting profiles on H. Dynamic delegation represents an extension of *direct delegation*, which in turn was introduced in grid environments to support highly-dynamic, ad-hoc workgroups [13].

AP3 concerns directly neither problem (i) nor (ii), but is intended to give some more insight on how requirements **AP1** and **AP2** should be realized. It establishes that credential-based directory services are used to export at the network layer what is of concern for that

[12]Of course, single database records are not encompassed here; however, they falls under the control of a database system which, in turn, falls under the control of an operating system.

layer of a principal security profile, leaving in local repositories (that is, at the host layer) what is host-specific. Since a security profile (see Section 3.2) is the joint set of entity authentication coordinates and authorizations, the idea here is to extend to authorizations the way hostnames and authentication information are resolved in modern networked environments.
In order to illustrate this, let us consider just the basic case in which a principal p requires a network service s (i.e. processing by another principal) provided by a remote host H[13]. We will assume that s has to check p's profile on H

$$p \longrightarrow (u, a) \longrightarrow C_H(p) = C_H(u) = I_H(u) \cup A_H(u) \tag{1}$$

for some credentials which cannot be directly desumed by p identity (i.e. they fall in the set $A_H(u)$). Then, **AP3** states that s has to resolve such authorization information starting from the authorization-related repositories provided and managed by H for the local access-control policy specification, and then checking some more (optional) directory services for p authorizations. There is a main difference here w.r.t. hostnames or principal identities resolution, since in this case the resolution process is *incremental* in its nature and requires *consistency*, too. Suppose that authorization information for p is stored in repositories R_1 and R_2. The resulting p authorization profile should be the set of authorizations a_i retrieved from both R_1 and R_2, but with the constraint that, if a_i clashes with a_j, then the authorization rule stored in a local repository takes precedence over the remote one. More generally, the precedence order of (possibly more than two) authorizazion repositories can be arranged to reflect the hierarchy security policy domains deployed over a network as a result of the existence of multiple organizations and their afferences to virtual ones.
This features can be easily implemented, coherently with **AP1** and **AP2**, by assuming that the repositories of authorization information can be managed through the *name service switch* framework. This is indeed the case for our development platform OpenSolaris, and Deliverables C and D illustrate such implementation.

4 Functional requirements

From Section 3.3 it follows that our solution of the access control problem in DS2OS environments results in a ACF that adopts the *dynamic delegation* access control model, and makes a combined use of directory services and OS protection mechanisms to enforce that model in such a way to accomplish requisites **AP1**-**AP3**.

In the present project, the implementation of the above ACF is restricted to the case of the SSH system entry service, as a "proof-of-concept" testbed. The following sections details about a precise formulation of dynamic delegation, and the functional requirements involved by our design at various levels.

4.1 Dynamic delegation

Given a networked environment as detailed in Section 3.1, and a host H, we will say that a principal s has a **sponsor profile** (or simply is a **sponsor**) on H if the profile of s on

[13]That does not represent a loss in generality, since the general case of a computation issued by p which might consist of different services provided by a set of remote hosts decouples in a number of the above basic cases, each concerning a single couple "service s at host H". Indeed, because of **AP2** we assumed no centralized access-control decision points; moreover, resource/service lookup and discovery, process sinchronization and other relevant issues in the field of distributed computations are not of concern from our strictly "access-control viewpoint".

H (see (1)) includes the credential for allowing other principals g, that have not a regular account on H, to operate on H with a set of credentials $C_H(g)$ such that

$$I_H(g) = \{UID(g) = \phi(g, UID(s)), GID(g) = GIDG\} \ , \quad A_H(g) \subseteq A_H(s) \ ,$$

where $GIDG$ is a special GID value reserved to guest profiles, and ϕ is a suitable function which allows to uniquely determine g given s and $UID(g)$. Principals g are said to have a **guest profile** (or simply to be guests) on H (see requirement **DD3** below).

Dynamic delegation (DD) is an access control model such that any principal p in a DS2OS environment - thanks to a *single sign-on* - is allowed to perform actions and to run computations on the set $\{H_i\}$ of hosts composing a such environment according to p's sponsor, guest and/or standard profiles on each H_i and the security policies enforced on those hosts.

It is assumed that:

Requirement Set 2 (Dynamic Delegation Requirements) *:*

DD1 *the identification of a principal p is unique in the context of a DS2OS environment, possibly encompassing the entire Internet, althought p might have a local identifier (other than the UID) on some or all the H_i;*

DD2 *both authentication and authorization of p on H_i could rely on asymmetric-cryptography-based tokens that are managed through public directory services;*

DD3 *a guest g on host H is not a regular user of H, in the sense that its profile is given by*

$$g \longrightarrow C_H(g) = I_H(g) \cup A_H(g) \ . \tag{2}$$

Expression (2) establishes that g is not supposed to have a password, neither - at least by default - a username, a shell and a home directory on H. The set I of identifiers for g is derived at runtime thanks to the function ϕ and the special GID of value $GIDG$[14]*.*

DD4 *a guest g on host H can exercise its sponsorship only for specified periods of time and/or as a consequence of suitable conditions, as defined by its sponsor s according to the security policy of H;*

DD5 *a single user on H can act as sponsor s for different guests g_j, and each g_j must have its own separate authorization profile and its own separate running environment;*

DD6 *a single principal p can be the guest of different sponsors s_j on the same host H. In this case, p has different profiles on H, one for each s_j, and these profiles must be kept separate and not-joinable*[15]*.*

[14] Thus, a guest profile on a Unix-like system is not managed through the standard local databases `passwd` and `shadow` (or their remote counterparts), but thanks to the certificates provided in **DD2** and the repositories used to specify the sponsor profile. As we shall see in Deliverables C and D, there is however one main exception: when g is allowed to log in H. In this case, for backward compatibility with the legacy system entry service workflow, it is advisable for g to have username, shell and home directory, and that such information is managed - as usual - through `passwd`

[15] In case of an entry service, it is allowed that p gets the same home directory and shell. However, ps set of credentials on H varies with s_j and these sets cannot be joined in some way to get a more powerful profile on H.

4.2 Operating system requirements

This project has been carried out using OpenSolaris as OS development and deployment platform. OpenSolaris implements extended security attributes for users and support a right-based security policy (see Deliverable B). Both these features are key factors for the implementation of direct delegation at the OS level, as shown in Deliverable D. Obviously, a great care has to take in order to assure that the above implementation does not introduce security flaws or bugs. The requirements at the OS layer can be summarized as follows:

Requirement Set 3 (Operating System Requirements) *:*

OS1 *the target OS must implement user security profiles that allow for the enforcement of not ID-based credentials. In other words, the set A_H in (1) must be not empty;*

OS2 *the OS databases storing the credentials in A_H must be managed through the name service switch framework;*

OS3 *the filesystem must support advanced access control modes and customization at the single user level. Moreover, in case a guest is allowed to have its own home directory, the system must support virtualization technologies in order to keep separate and under special control the guest environment from those of standard users;*

OS4 *the new access control facilities and their implementations must be backward compatible with any* discretionary*,* role-based *or* mandatory *access control models and security policies and mechanisms implemented in the target operating system;*

OS5 *the extended model should neither introduce new architecture security flaws, nor substantially augment the "threath-exposition" surface for the access control subsystem.*

4.3 Directory service requirements

In this project, we propose *Lightweight Directory Access Protocol* (LDAP) [36] to realize the X.509-compliant PKI and PMI that support our environments. LDAP, which was created to replace DAP directory service protocol, has been object in recent years of extensive researches and contributions, and is the de-facto standard for storing and querying X.509 PKCs.

Our design introduces a new certificates and LDAP structures in order to support direct delegation, as described in detail in Deliverable C. The requirements at this layer can be summarized as follows:

Requirement Set 4 (Directory Service Requirements) *:*

DS1 *a specific attribute certificate is required to manage at the network layer the credentials stored in A_H*

DS2 *the directory service must implement suitable structures to reflect the existence of guest and sponsor profiles;*

DS3 *The authorization information from the local databases and the directory must be integrated in an incremental and consistent way, as described at the end of Section 3.3. This requires a special handling during the resolution process performed through the name service switch framework.*

4.4 Entry service requirements

Our (prototipal) implementation of an ACF for DS2OS environment is restricted to the *secure shell* (SSH) service. A first, PMI-compliant SSH implementation is described in [22]. SSH is an Internet standards track protocol for secure remote login and other secure network services, which is widely adopted to operate on a remote host through a command-line shell in a secure way. Indeed, SSH encrypts communication sessions after the mutual authentication of the client and the server, thus allowing integrity and confidentiality protection for both user login coordinates, and subsequent transmitted data.

Currently two major versions (SSHv1 and SSHv2) of the protocol exist, but SSHv1 has some security weaknesses and its support in modern implementations is provided only to help sites with existing SSHv1 clients and servers to transition to SSHv2. For that reason, we will refer to SSHv2 only.

SSH host-based authentication relies on asymmetric cryptography, user authentication can either use it optionally, and both are designed to support public-key certificates. Nevertheless, the protocol makes no assumptions for an infrastructure or means for distributing the public keys of hosts, allowing communication without prior communication keys or certification, both from the server-side and the client-side[16].

Apart from that, all the interactions provided by SSH are possible only if the requester (i.e., the client agent) has a regular account on the remote host which offers the service, or otherwise she knows the authentication credentials of another user on that host. This is a severe limitation, which is incompatible even with grid basic functional requirements, leading the Globus designers to realize a grid-aware SSH server, that is supporting Globus *proxy-certificates* [32].

In our design, we required the following features for the SSH service and, more generally, for any other system entry service:

Requirement Set 5 (Entry Service Requirements) *Given a principal p and a host H with an entry service ϵ, then:*

ES1 *p can log in H through ϵ if p has been granted a suitable guest profile on H;*

ES2 *its sponsor s entitles p to get a complete operating environment (e.g. username, home directory, shell, credentials, environment variables, etc.) in such a way to satisfy, in particular, requirements* **DD3** *and* **OS3**.

[16] This choice was made with the consciousness that it exposes to *Man-in-the-Middle* attacks, but in order to make the protocol much more usable during the transition time until a widely deployed PKI will be available on the Internet [35].

References

[1] Chadwick D.W., Otenko O.: The Permis X.509 Role Based Privilege Management Infrastructure. Future Gener. Comput. Syst. **19** (2003).

[2] Chadwick D.W.: Authorization in Grid Computing. Information Security Technical Report **10** (2005).

[3] Cooper D. et al.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, *RFC 5280* (2008).

[4] DACS Distributed Access Control System `http://dacs.dss.ca/`

[5] Ellison C. et al.: SPKI certificate theory, *RFC 2693* (1999).

[6] Fielding R. et al.: Hypertext Transfer Protocol - HTTP/1.1, *RFC 2616* (1999).

[7] Foster I. et al.: The Anatomy of the Grid, *J. Supercomputer Applications* (2001).

[8] Foster I. et al.: A Security Architecture for Computational Grids, *Proceedings of the 5th ACM Conference on Computer and Communication Security* (2003).

[9] Globus Toolkit 4.2.1 Security Documentation
`http://www.globus.org/toolkit/docs/4.2/4.2.1/security/#security`

[10] Gunzer H., Introduction to Web Services, *Borland white paper* (2002). `http://archive.devx.com/javasr/whitepapers/borland/12728JB6webservwp.pdf`

[11] Keromytis A.D., Smith J.M.: Requirements for Scalable Access Control and Security Management Architectures, *ACM Trans. on Internet Technology*, vol. 7, n. 2 (2007).

[12] Konfelder, L.M.: *Towards a Practical Public Key Cryptosystem.*, MIT BS Thesis (1978).

[13] Lorch M. , D. Kafura: Supporting Secure ad hoc User Collaborations in Grid Environments, *Grid Computing - GRID 2002 3rd International Workshop, LNCS 2536* (2002)

[14] Internet Engineering Task Force NWG: Requirements for Internet Hosts - Communication Layers, *RFC 1122* (1989).

[15] Internet Engineering Task Force NWG: Requirements for Internet Hosts - Application and Support, *RFC 1123* (1989).

[16] International Telecommunication Union: The Directory - Authentication Framework. *ITU-T Rec. X.509* (1988).

[17] International Telecommunication Union: The Directory - Authentication Framework. *ITU-T Rec. X.509* (1997).

[18] International Telecommunication Union: The Directory - Authentication Framework. *ITU-T Rec. X.509* (2001).

[19] International Telecommunication Union: The Directory - Authentication Framework. *ITU-T Rec. X.509* (2005).

[20] International Telecommunication Union: Security Frameworks for Open Systems - Access Control *ITU-T Rec. X.812* (1995).

[21] Laccetti G., Schmid G.: A Framework Model for Grid Security, *Future Gener. Comput. Syst.* **23**(2007)

[22] Laccetti G., Schmid G.: A PMI-Aware Extension for the SSH Service, *Lec. Not. Comp. Sci.* **4967** (2008)

[23] Loscocco P. A. et al.: The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. *Proc. of the 21st National Information Systems Security Conference* (1998)

[24] Mauro J., McDougall R.: *Solaris Internals* (2nd ed.), Sun Microsystem Press (2005)

[25] Mockapetris P.: DOMAIN NAMES - CONCEPTS AND FACILITIES, *RFC 1034* (1987)

[26] Mockapetris P.: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION, *RFC 1035* (1987)

[27] National Institute of Standards: *Glossary of Key Information Security Terms*, 1st Rev. (2010)

[28] Saltzer J., Schroeder M.: The Protection of Information in Computer Systems, *Proceedings of the IEEE* (1975).

[29] Samar V., C. Lai: Making Login Services Independent of Authentication Technologies, *Proceedings of the SunSoft Developers Conference* (1996).

[30] Sun Microsystems Security Engineers: *Solaris 10 Security Essentials*, Sun Microsystem Press (2009).

[31] U.S. Dept. of Commerce: FIPS Pubs 196 - *Entity Authentication Using Publi Key Cryptography* (1997).

[32] Tuecke S. et al.: Internet X.509 Public-Key Infrastructure (PKI) Proxy Certificate Profile, *RFC 3820* (2004).

[33] Vaquero, L. M. et al.: A Break in the Clouds: Toward a Cloud Definition, *ACM SIGCOMM* (2009).

[34] Vixie P.: What DNS Is Not, *Communications of the ACM* col. 52 n. 12 (2009).

[35] Ylonen T.: The Secure Shell (SSH) Protocol Architecture. *RFC 4251* (2006).

[36] Zeilenga K.: Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates *RFC 4523* (2006).

[37] Zimmermann P.R.: *The Official PGP User's Guide* (1995).