Distributed Object Systems Security

Prof. Vijay Varadharajan

Dept. of Computing Science, University of Western Sydney, Nepean,

P.O.Box 10, Kingswood, NSW 2747, Australia.

Email : vijay@st.nepean.uws.edu.au

January 9, 1995

Abstract

In this paper, we consider a distributed object environment, and describe how security services and mechanisms can be provided. We also discuss how these security measures can be achieved within the OMG/CORBA model.

1 Introduction

Security plays a vital role in the design, development and practical use of the distributed computing environment, for greater availability and access to information in turn imply that distributed systems may be more prone to attacks. The widespread publicity of attacks has also helped to create a greater customer awareness of security issues, and the need for protection in the technology they buy.

A distributed system architecture comprises networks, users (people), information storage resources, computing resources, and peripherals. One may have various levels of interaction and various degrees of sharing between these entities.

At one level, we may have interaction between different resources : such as host-host, host-peripheral, and host-storage resource. At another level, one may have interactions between applications in different entities. At the outer level, we can have interactions between users. Furthermore, the degree of interaction and sharing may also vary. At one level, there may be just transfer of information (e.g. email, edi). At another level, one may have sharing of computing resources (e.g. processors) and sharing of information storage resources (e.g. disks). At the next level, one entity (e.g. an object) may act upon another entity (e.g. object) to obtain a service from the latter. A still greater degree of cooperation occurs when entities jointly work together to perform tasks.

In each of these activities various attacks can occur and hence the need for security measures become significant. Furthermore, in practice, organizational structure and boundaries are superimposed on the distributed computing environment. The diagram shown in Figure 1 illustrates several organizations and several departments within each organization. This has impact on who has responsibility and authority over which parts of the distributed system. This in turn has implications on trust which is crucial for the placement and operation of security functions, and the interactions between them.

The paper is organized as follows: Section 2 very briefly describes the security threats and attacks faced in a distributed environment, and the types of security services and security

Figure 1: A Simple Distributed System Environment

mechanisms that are required to counteract these threats. We also mention the security management functions required. In Section 3, we consider a distributed object environment and describe how the security services and mechanisms can be provided, and some of the trust dependencies involved. We discuss how this can be achieved within the proposed OMG/CORBA model. Finally, Section 4 concludes by extracting out some of the key issues that we need to consider when designing secure distributed systems.

2 A Security Model Overview

2.1 Security Threats

This section describes a list of security threats that can arise in a general distributed environment.

- *Masquerading* : This can happen at different levels in a distributed system : within a computer system, a user or process might masquerade as another to gain access to a file or memory to which it is not authorized, while over a network, a masquerading user or host may deceive the receiver about its real identity.
- *Unauthorized use of Resources* : This includes unauthorized access to both resources on the networks as well as within a system. For instance, within a computer system, this threat corresponds to users or processes accessing files, memory or processor without authorization. Over a network, the threat may be in the form of accessing a network resource. This may be a simple network component such as a printer or a terminal, or a more complex one such as a database, or some applications within the database. Thus unauthorized use of resources may lead to theft of computing and communications resources, or to the unauthorized destruction, modification, disclosure of information related to the business.
- *Unauthorized Disclosure and Flow of Information* : This threat involves unauthorized disclosure and illegal flow of information stored, processed or transferred in a distributed system, both internal and external to the user organizations. Within a system, such an attack may occur in the form of unauthorized reading of stored information, while over the network, the means of attack might be wiretapping or traffic analysis.
- *Unauthorized Alteration of Resources and Information* : Unauthorized alteration of information may occur both within a system (by writing into memory) and over the network (through active wire-tapping). The latter attack may be used in combination with other attacks such as replay whereby a message or part of a message is repeated intentionally to produce an unauthorized effect. This threat may also involve unauthorized introduction (removal) of resources into (from) a distributed system.
- *Repudiation of Actions* : This is a threat against accountability in organizations. For instance, a repudiation attack can occur whereby the sender (or the receiver) of a message denies having sent (or received) the information. For instance, a customer engages in a transaction with a bank to debit a certain amount from his account, but later denies having sent the message. A similar attack can occur at the receiving end; for instance, a firm denying the receipt of a particular bid offer for the tender even though it actually did receive that offer.

- *Unauthorized Denial of Service* : Here, the attacker acts to deny resources or services to entities which are authorized to use them. For instance, within a computer system an entity may lock a file thereby denying access to other authorized entities. In the case of the network, the attack may involve blocking the access to the network by continuous deletion or generation of messages so that the target is either depleted or saturated with meaningless messages.

2.2 Security Services and Mechanisms

A security model should be applicable for a wide range of systems and applications, and consequently it is intended that it should include a wide range of security services that can be used and combined in different ways to meet different security policies. [2]. In particular, a distributed environment is likely to have multiple security policies and different authorities responsible for various parts of the system. We consider the aspect of multiple authorities later in Section 2.3.

In developing a security service, we need to address at least the following questions :

- what are the security information/attributes used by the service?
- what mechanisms can be used to provide the service and what are the associated rules of operation?
- what are the authorities that are involved in the management of the service and its associated mechanisms?

Let us now consider the security services.

Identification and Authentication : This service provides the confidence that at the time of request, an entity is not attempting a masquerade or to mount a replay attack [3].

Authentication Information : Attributes such as password, keys, smart card, retinal scan, biometrics.

Authentication Mechanisms : Possible authentication mechanisms include :

- * Claimant presents the authentication information (such as a password) to the verifier who then authenticates it.
- * Protected exchange of authentication information, e.g. using cryptographic techniques or one-way functions.
- * Challenge-response techniques.

Either one-way or mutual authentication may be provided. In a general situation, when two parties wish to authenticate each other, they may need to involve one or more third parties. A simple model is when we have a single trusted third party. The nature of trust between each party and the third party is an important issue in determining the assurance of the service. Examples of trusted third parties include authentication servers, key management servers and certification authorities.

Access Control and Authorization : This service provides the ability to limit and control access to host systems, applications and information, and to limit what resources might do with the information contained, e.g. in applications, files. In an access control scheme, we have certain entities (initiators) attempting to access other entities (targets). [4]

Access Control Information : Access control information used in the decision process includes the following:

- * individual identities of initiators and targets,
- * group identities of initiators and targets,
- * security labels of initiators (e.g. clearances) and targets (e.g. classifications),
- * roles (e.g system administrator, manager) of initiators and targets,
- * the actions or operations that can be allowed to be performed on the target and
- * other contextual information, which may include time periods (over which the access is granted or denied), routing information (thereby granting an access request for only some specific routes), and location information (thereby granting access only to initiators at specific end systems, workstations or terminals).

Access Control Rules: An access control policy essentially specifies a set of rules that define the conditions under which initiators may access targets. The decision to grant or deny a particular request is determined using access control rules and access control information associated with the request.

Delegation: An important requirement that commonly arises in a cooperating environment and is associated with access control is that of delegation. The principle behind delegation is that one entity can authorize another entity "to act on its behalf". Delegation allows us to have a more flexible and dynamic form of access control. The security model can provide generic mechanisms to allow for delegation. The mechanisms allow for the following :

- * From the originator, or the delegator point of view, it may wish to give only a part of its overall rights, or even just a single right. Furthermore, it may only want to grant these rights for just a limited duration. Also it should be able to identify each of its delegations so that it may at some stage attempt to revoke one of these delegations.

- * The entity which has been delegated must be able to prove its delegation to an end point. It may be required for the delegated object to know the scope of its privileges, in terms of what and who they are for, and how long they may last. This can be done by passing a delegation token. It may also be necessary to set further delegations or to request access to an end point using the rights delegated to it (unless the rights have been revoked).

- * The end point receiving a request from an entity claiming some delegated rights must be able to read, authenticate and verify the tokens specifying these rights and duration. The end point can then make an access decision based on knowledge of the requestor's rights together with the delegation token and the access control information.

One can provide suitable delegation schemes using the authentication and access control information and mechanisms described in [7].

Information Confidentiality: Confidentiality service provides for the protection of information from unauthorized disclosure.

Confidentiality Mechanisms : The mechanisms that are typically used to provide confidentiality are based on cryptographic techniques. In an operating system environment, it may be sufficient to protect the confidentiality of information just by using access control mechanisms. In a network environment, we can provide link-to-link or end-to-end encryption.

Confidentiality Attributes : These include:

- * Secret keys
- * Public and private keys

Information Integrity: Integrity service provides for the protection of information from unauthorized modification. The unauthorized modification may involve alteration, insertion, or deletion of information.

Mechanisms: The provision of the integrity service involves:

- (1) generation of integrity checks (at the originating end).
- (2) verification of integrity checks (at the receiving end).

Integrity mechanisms employ cryptographic techniques to produce integrity checksums which can be used to determine whether there has been any insertion, deletion or reordering of information. This is done using feedback chaining techniques. E.g. using the cipher block chaining (CBC) technique to generate the Message Authentication Code (MAC).

Non-Repudiation: The non-repudiation service provides the proof of the origin or delivery of information. This protects the sender against the threat of false denial by the recipient (that the information has been received) and protects the recipient against the threat of false denial by the sender (that the information has been sent).

The common mechanism for providing non-repudiation relies on the use of digital signatures. Typically the digital signature is calculated using the asymmetric (public key) algorithm described above. A hash function is needed when it is necessary to sign long messages.

Auditing and Accountability: The security audit service is complementary to all the security services described above in that it is not directly involved in the prevention of security violations but assists in their detection. The security audit can in turn be used to test the adequacy of the security controls and the conformance of the system with the security policy, and to assist in the formulation of any modification to the security policy.

Auditing services are crucial both within a system and over the networks. Following audit analysis, an entity may be held accountable for its actions so that violations or attempted violations of system security may be traced uniquely to it.

Auditing is provided by first defining the security related events and generating security audit and/or alarms. The security audit is then analyzed to evaluate the correctness and strength of the security policy. For secure reporting of audit information, auditing service may use other services such as authentication, integrity, confidentiality and non-repudiation.

Availability and Prevention of Denial of Service: Denial of a service can be regarded as an extreme case of information modification in which the information transfer is either blocked or drastically delayed.

A measure against such an attack is provided by arranging periodic exchange of information between entities to ensure that an open path exists between them. The greater the frequency of such a request response mechanism, the shorter the time period during which the denial of service attack will remain undetected. However the disadvantage is that this reduces the effective bandwidth of the network.

2.3 Security Management

When describing a security model, we also need to consider how to manage these services and how changes in policy and its enforcement can take place. [8]. For instance, in the case of confidentiality and integrity services, it is necessary to manage the keys used in the encryption and decryption processes. In the case of access control service, we need to manage the access control information such as access control lists and the access rules. Similarly in the case of authentication, authentication information, e.g. passwords and keys, needs to be managed. In the case of auditing, the management of audit trails and audit analysis is necessary.

In distributed systems, it is likely that there is no single authority that controls the entire environment. For instance, in an organization there may be several security managers

responsible for a subset of users, objects and operations. This does not mean that it is not possible to control security in a distributed environment centrally. However even central security authorities end up trusting that the authorities responsible for local systems have implemented appropriate security.

In practice, there may be several authorities performing different aspects of these security management functions.

- * Access Control and Authorization Management (Service: Access Control and Authorization)

- * Authentication Management (Services: Identification and Authentication, Non-repudiation, Information Integrity)

- * Key Management (Services: Information Confidentiality, Information Integrity)

- * Audit Management (Service: Auditing and Accountability)

2.4 Security Domains

One way of managing a large distributed environment is to partition it in terms of separate security domains. A security domain may be thought of as a set of elements under a given security policy administered by a single authority for some specific security relevant activities. Examples of elements of a security domain in an open systems environment include logical or physical elements such as application processes, relays, and human users. One can have different relationships between security domains. For instance, domains may have a peer-to-peer relationships in the case of two company security networks. Domains may have subdomains.

In order for interactions to take place between domains, it is necessary that there be common aspects in the security policies of the domains. Depending on the relationship between the domains, the secure interaction may come about as follows :

- * For secure interaction between two independent security domains to be feasible, the security policies must enable common services and mechanisms to be selected, possibly through negotiation, and security attributes in each security domain to be related to each other, possibly through mapping.

- * In the case of security subdomains within the same security domain, secure interaction between subdomains can be established by the security domain authority.

- * Secure interaction between domains via a common third party domain can be established by the common security domain for the sets of activities for which it is given jurisdiction.

In general, when developing such a multi-domain system, one needs to identify the activities involved in each domain, the role and functions of the authorities involved and the interaction between these authorities. Typically, this may involve the need for additional third parties responsible for holding some security relevant information and acting as an arbiter in resolving conflicts.

2.5 Trust

Trusted mechanisms are essential to ensure the enforcement of security controls and in the provision of the necessary assurance that the correct operation of the security measures are maintained. Whatever the combination of security services and mechanisms have been chosen to meet the security requirements and to implement the security policy, it is necessary to trust some collection of agents - whether human, software or hardware - to carry out the

security functions correctly. In particular, the security management authorities mentioned above need to be trusted to maintain the security of the system.

In a multi-domain system, trust is based on mutual assurance which can be obtained either by something a domain is told or by something a domain knows. For instance, a subdomain could be told whom to trust by a superdomain. Or a domain may trust another as a result of negotiated policies.

3 A Simple Distributed Object Environment

3.1 Object System

The system comprises a number of objects interacting and cooperating with each other via some network medium. An object is an identifiable, encapsulated entity that provides one or more services to clients. A client is a program or a process that issues requests to an object. In particular, an object could itself be a client. The objects could be of different levels of granularity. Typically one can envisage conventional client server systems to consist of large entities such as machines, whereas objects in an object system could be of fine granularity such as a document, a chapter in a book, a page or even an integer. We will refer to the machine objects as simply machines. Each machine may support a number of objects, and at any instant, each object is supported on a particular machine.

The objects communicate with each other by passing messages. So a client object requests a certain service on an object by sending an appropriate message. The request is an event, and it has such information as the object reference or identifier, the name of the operation, zero or more parameters and possibly a context. When a client issues a request, a method at the target object corresponding to the operation requested is called. The method is activated and the code corresponding to the method is executed and any results produced are passed back to the requesting client.

In order for the system to work in a distributed environment, we need some form of a location service which keeps track of all the objects in the system, and is able to route the client request to the target object. Furthermore suitable naming schemes will be necessary for assigning object identifiers and references. In the Object Management Group's Common Object Request Broker Architecture (CORBA), the Object Request Broker (ORB) is responsible for all of the mechanisms required to find the object implementation for the request, and to communicate the data making up the request.

Consider the following typical scenario : an user A logs onto a machine W and carries out certain tasks. As part of this computation, an object O1 may require some service from another object O2 residing on a remote machine S.

Figure 2 shows an example of such a situation. Before considering the security services, let us first identify the security threats in such an environment. In fact, most of the security threats mentioned earlier are applicable here.

3.2 Security Threats

Typical security threats that can occur in the above scenario include the following :

1. Masquerading.
 - (a) a user A' pretending to be user A either to the local machine or to a remote machine.

Figure 2: A Simplified Distributed Object Environment

- (b) an object O1' pretending to be the object O1 in sending requests.
 - (c) an object O1 pretending that it is acting on behalf of user A'.
2. Unauthorized access to an object, e.g. an object O1 requesting a service from the object O2 for which it has no authorization.
 3. Threats to communication security
 - (a) modifying the messages between machines W and S, thereby affecting the contents of the service request from O1 to O2, and the returned results from O2 to O1.
 - (b) eavesdropping on the messages between machines W and S.
 4. Ensuring accountability of users' actions.

4 Security Services

We will assume the existence of a Security Facility (SF). For instance, this may be part of the Object Management Facility (OMF) which is typically responsible for message delivery, object location and resource management. In terms of the OMG model [12], the SF can be seen as an extension to the ORB core. The important point is that we will assume that a part of ORB extended with the SF resides on the client machine. Similarly, part of the ORB extended with SF is assumed to exist within the server machine. The SF is trusted and provides a trusted set of services to a collection of objects. This should become clear as we describe the security services and how they are provided. In particular, SF can be regarded part of the Network Trusted Computing Base (NTCB) [1].

The security services required to counteract the above threats are as follows :

1. Authentication Service to counteract masquerading
2. Access Control and Authorization Service
3. To counteract communication threats, we need
 - (a) Integrity Service
 - (b) Confidentiality Service
4. Auditing Service

4.1 Authentication Service

When a user logs onto a machine, his identity needs to be verified. The SF needs to be involved in this process. With regard to how and where the authentication process can be carried out, we have the following :

The SF interface provides the authentication service to the objects and users of the client, with the assistance of an Authentication Server (AS). AS is a trusted entity on the network. One such entity per security domain is envisaged. The interface offered by the AS to the SFs should be independent of the mechanisms used to provide authentication.

The type of authentication information used may vary from the knowledge of a password (or PIN), a key, to possession of an intelligent token containing a detailed set of user related information, e.g. in the form of a smart card.

Different authentication mechanisms are possible to meet this requirement : for illustration purposes, consider the use of a challenge-response mechanism.

Assume that users and clients and targets have agreed on a publicly available one-way function to be used for authentication, and the AS maintains a table of user related authentication information, such as keys and passwords in a secure manner.

When a user tries to login, a challenge is issued to him in the form of a fresh random number. For instance, a protocol could be along the following lines : The SF interface passes the client user's login request to the AS and the AS issues the challenge through the SF. The user calculates the response by applying the publicly known function and his key to the random fresh value and the result is returned to the AS for verification.

Such challenge-response schemes can be implemented using either the symmetric or the public key approaches.

- *Symmetric key approach* : Here the AS has the secret keys of the users. The user employs his secret key in calculating the response to the challenge. The secret key can either be stored in a smart card or may be derived from the user's password. The AS uses the alleged user's stored secret key in its verification process and compares it with the result received from the user.
- *Public key approach* : In this case, only the public keys of the users are kept by the the AS, whereas the users' private keys are held only by the users or by their smart cards. The user now applies his private key to the random challenge value and the AS uses the public key of the alleged user in checking the value.

Note that Kerberos [9] provides a symmetric key approach, whereas DASS [10] uses public keys. Both these use timestamps and not the challenge-response approach mentioned above. Once the authentication is achieved, the AS provides an Authentication Token to the client SF which it can use to authenticate itself in the future. The Token should atleast have the following : Identity of the user validated by the AS, Timestamp and validity periods over which the identity is valid, and Keys associated with the user such as the Public key of the user or protected symmetric keys that the user can employ in subsequent communications. We will not consider the precise structure of the token at this stage. The structure will also be affected whether public key or symmetric key technology is used. As SF is trusted, we will assume that the Token is in fact stored by the SF on user's behalf.

In the scheme described above, the user must have the means to be able to perform the calculation on the issued challenge. This may be achieved using a smart card, given the appropriate interface to the workstation. The smart card can contain the one-way function with the user key bound in, so that it can return the response for a given challenge. (In fact, one can make use of the smart card to store additional information, such as the privileges of the user.)

In OMG model, the authentication service can be seen as an extension to the current ORB services. The SF interface can be seen as an extension of the ORB interface.

4.2 Access Control and Authorization Service

The main role of the access control service in this example is to apply control over messages between objects. In providing the access control service, we need to decide the following :

- What access control information is used in the decision making process? In the implementation, this information may be implicitly stored or may be derived.

- What access control rules are required?
- What authorities are there in the implementation?

Returning to the scenario, we have an object O1 in a machine W with SF1 wishing to make a request on another object O2 on a machine S with SF2.

Access Control Information and Access Rules :

The access control information used in the specification of access rules can be based around some attributes of the initiating object (O1), the target object (O2), and the action being attempted.

Typical attributes of the initiator and the target objects that can be used include : the identity of the object, the class of the object, the owner of the object, the membership of the owner or object in some domain, his or her role, privileges associated with the objects, and sensitivity/integrity labels associated with the objects. As for the action being attempted, two approaches are possible. First, the decision may be based simply on the message type. Secondly, the decision may be based on more fundamental properties: the familiar “know-about”, “read”, “modify”, “move” and “delete”, and so on.

As objects are created, they must somehow have access control information associated with them, by a general default, or on the grounds of their creation in a certain context, such as a containing object. Thus, the correspondence between message type and creation is important. Furthermore in system operation, there may be a need to change the access control information and rules associated with them. These depend on the access control policy. For instance, the policy might permit some users to change the access control information to certain objects, whereas for some other objects, this may be totally under the control of the SF.

Using the above access control information, in this example, one can for instance specify identity-based, group and role based, label based access or capability based access control policy rules.

Also the chain of messages which cause the message and the attributes of the intermediary objects might be important. Where there is a notion of binding, where one access control decision corresponds to several interactions, this is practicable. However, checking chains of arbitrary length may be too heavy for a scheme based on decisions for every call. If this is the case, perhaps it may make sense to restrict the decision either to the immediate originator of the message or to a combination of the first and last in the chain.

Feedback to callers needs to be designed with “know-about” rights in mind. If a user tries to access an object on the remote machine, five things may happen: the object might not exist; the object might exist but fall outside the restricted view granted to the user by the owner of the remote machine; the object may not support the chosen message type; the object may not be allowed to respond the chosen message type under the restricted view; the object may be allowed to respond to a supported message type. It is desirable that the responses to the remote user in the first and second cases, and in the third and fourth cases, be the same. For the third case, this may require some discipline on the part of the developer.

Authorities

The access mediation process consists of two parts: First the access request by an object for an action on another object needs to be evaluated using the relevant attributes of the objects in question and the specified access control rules. We will refer to the entity which performs

this function as *Access Decision Facility* (ADF) [4]. Secondly, the decision is enforced thereby either allowing the access request or denying it; this is carried out by an *Access Enforcing Facility* (AEF) [4]. Now the question to consider is how many such AEF and ADFs do we have for our application and how are they distributed? Several alternatives are possible, as we describe below.

First, note that as there is no notion of binding between objects, it seems necessary that access control checks need to be performed call by call. Secondly, we will assume for this example that there is no central access control authority which decides who can access what servers. Thus there are three possible locations for authority: the initiator SF, the target SF, and the target object implementation itself. In the case of the OMG model, the target SF can be seen as part of the extended ORB residing at the server machine, or as part of the Basic Object Adapter (in fact, part of the Skeleton).

The initiator SF can enforce control over outgoing messages so that an initiator may not perform an action which is not in conformance with the policy for the initiator. The target SF can enforce control over the incoming messages so that the target object may not receive a request that is not in conformance with the access control policy for the target. The target object itself may perform additional controls to decide whether to grant the request or not. Enforcing control at the initiator will help to eliminate some calls at the local end, thereby increasing efficiency. Enforcement at the target may be adequate for many situations whereas there may be certain circumstances where at least some of the enforcement needs to be performed by the destination object implementation itself. Note that there is a trust issue here with respect to leaving the access decision and enforcement at the object itself.

There is also the case when understanding of the semantics of the message is required in making the access decision. For instance, a given message may result in processing which creates, deletes or moves objects, under an arbitrary message type. If the access policy at the SF is described in terms of read, modify, destroy and so on, the SF will not be able to interpret the message in these terms. One possibility may be for the SF to pass on access information to objects in the form of hints, expressed in terms of read, modify, destroy and so on, and let the object to make the decision based on these. However, this will place extra trust assumptions on the object.

Alternatively, access control rules can be defined in terms of the message types. Creation of objects requires their entry in the access control scheme. This can be taken care of using the primitives provided by the SF. Given that the SF is always involved in object creation, the involvement of the local SF in access control decision can help solve the problem of the correspondence between message types and object creation.

Related to the decision of ADF and AEF placement is the problem of the location of information representing the state of access control. It may make sense to associate information on restrictions on access to an object with the object itself. However it is appropriate to store such information in the SF which may be protected and trusted, whereas an object may not be.

4.3 Proxy

An important requirement that commonly arises in cooperating object-oriented systems and is associated with access control is that of delegation. The principle behind delegation is that one object can authorize another object "to act on its behalf". Delegation allows a more flexible form of access control.

Returning to our scenario, the delegation situation occurs when the object O1 authorizes an

object O3 to make a request on his behalf to object O2. Delegation is secured by verifying that an object that claims to be acting on another's behalf, is indeed authorised to act on its behalf. In practice, this also involves secure communications between objects. We have considered the problem of delegation in distributed systems in detail in [7]. In particular, the nested or linked token protocol for delegation can be used here. Once again we can implement these using either public key or symmetric key cryptosystems, as described in [7]. We will not describe these schemes in this paper.

4.4 Integrity Service

Recall that the authentication process would have resulted in the establishment of key(s) associated with the user. This could be in the form of a public key associated with a user validated by the AS. Or it could be in the form of symmetric keys provided by the AS in a secure manner to the user/client SF, which are shared by other SFs in some target machines. In either case, these keys can be used to establish a secret conversation key between the client objects and the target objects.

The role of the integrity service is to ensure the integrity of messages transferred between the objects. For instance, considering our scenario, we need to ensure that the integrity of the request from the object O1 from the machine W to object O2 in the machine S is protected.

Depending on whether symmetric key or public key method had been used, we have different possibilities.

Symmetric key approach : First the message whose integrity needs to be protected is first hashed and then the resulting hashed (smaller) message is "signed" (encrypted) using the sending client SF key to produce a checksum. The SF at the receiving end can then calculate the checksum and see if it matches with the received one. This would not only ensure message integrity but also provide origin authentication as the sending SF's key has been used. In practice, one may include other parameters within the message such as a sequence number and a timestamp.

Public key approach : With the public key system, the hashed message can be signed using the sending SF's secret key of the PK system to produce the checksum. The receiving SF can verify the checksum using the public key of the sending SF which it knows. Once again this provides message origin authentication and integrity. In fact, this scheme also provides non-repudiation, as no other SF could have generated the checksum. This is not so for the symmetric scheme as the receiving SF also has the ability to generate the checksum.

4.5 Confidentiality Service

This service provides confidentiality of messages between the machines, for instance between W and S in our scenario. Once again, following the SF authentication process, this would have resulted in the exchange of keys between the SFs, and we can use these keys in the provision of the confidentiality service. Computational requirements may constrain us to use a symmetric key approach in the provision of this service. The messages will be encrypted by the sending SF, which can be decrypted by the receiving SF.

4.6 Auditing Service

The main role of the audit service is to collect security relevant information from system operation that can be used to test the adequacy of the security controls.

From above we can see that SF in the machine plays an important role in the operation and management of security services. SF is responsible for enforcing security controls in the machine. So it is natural that SF should be involved in the auditing process.

The security policy defines those events that are security relevant. Some typical security relevant events in this example may include : request for some specific actions such as creation of new objects, deletion of objects, requests from some specific objects/machines, requests to some specific objects/machines, usage of some special objects, failure of requests, events causing changes to access control rules and access control information. For each recorded event, the audit record can identify : date and time of the event, user/object identity, type of event and success or failure of event. For creation or deletion of objects, the record can include the name of the object.

The security audit record can then be analyzed locally, or a suitably filtered version of the record can be sent to a trusted authority for further analysis. Note that this trusted authority could be another server on the network offering its interface to the SFs.

4.7 Trust Dependencies

Some of the trust dependencies in this example include the following:

- The SF is trusted to operate correctly in the provision of all the security services.
- The SF is trusted to convey the requests of objects it is managing accurately.
- Information stored in the SF is trusted to be secure and the SF is trusted to manage the information that it holds.
- The SFs trust each other to operate correctly.
- It is not possible to for a user/object to be able tamper with the the resources of the SF itself.
- An extension of the proposed solution may allow different SFs to be trusted to different degrees. Here a service request can have the name of the originating SF marked within it, which can then be used to decide whether a service request should be granted.
- The user trusts the objects which are acting on his behalf not to make requests which he does not authorize.
- The Authentication Server is trusted to operate correctly. Information stored in the Authentication Server (e.g. keys) is trusted to be secure and the Server is trusted to manage them.

5 Discussion

Having considered the security services required in a distributed environment and looked at the options that are available, we conclude this paper by extracting out some key issues that need to be addressed in designing secure distributed systems.

First major issue is that of trust. Trust in a distributed system is not a monolith but that different entities/authorities are trusted for different functions and tasks. An authentication authority might be trusted with respect to authentication functions but may not be

trusted with respect to ascertaining privileges to an authenticated entity. In particular in a distributed system, it is likely that there are different in fact distinct authorities responsible for different functions. A target in making the overall decision such as whether to grant a service request or not to a client, needs to trust certain authorities and these authorities in turn trust each other for their specified responsibilities and the information they provide.

Secondly, the distribution, verification of security information in a distributed system needs to be clearly understood. With respect to distribution, in general, there are two possible schemes that represent two ends of the spectrum. One approach is where the client presents all the necessary security information to the target. The other approach is that the target gets hold of at least some of the security information necessary to make the decision. The former is sometimes referred to as the “push” approach whereas the latter is seen as a “pull” one. Let us for instance apply this to the authentication information. Typically, a client can pass its authentication attributes to the target for it to verify, or the target can get hold of additional attributes such as a certificate from a trusted authority which it can use to verify the information it receives from the client. Further note that the verification itself can be done by the target or with the assistance of a trusted third party. An example could be the need for public key capability for verification and that the target did not have such a capability. The choice is even more clear with respect to the authorization information. Consider, for instance, the credit privilege that a client can have. Whereas the authentication information may be pushed to the target by the client, the target might decide to get the credit privilege attribute of the client itself from a trusted third party prior to making the decision and at the time the decision needs to be made. This implies that certain type of information may need to be pulled whereas other information can be pushed. In fact, we can go further to say that in general “static, generic” information can be and perhaps should be pushed whereas “state dependent, dynamic” information may be pulled. Authentication information such as the identity or the certificate fall into the first category whereas the specific privileges dependent on state such as “the ability to spend 10000 dollars per week” may need to be pulled to check how much has already been used up at the time of a given transaction.

Finally, we mention the issue of distribution of trusted authorities themselves. In fact, this point is linked with the two issues mentioned above. Recall that in the first point above, we mentioned that it is likely that we will have different authorities trusted for different tasks in a distributed system. Here we are concerned about which authority “is responsible” for what. In general, we can envisage the following authorities in a distributed system : Authentication, Authorization, Notary, and Credit Authorities. Consider for instance a Consumer-Provider transaction. A client C wishes to purchase some information from a provider P. The authentication information for the client C may be provided by the Authentication Authority of the organization that C is a member of such as his company (X1), whereas the credit information relating to C’s ability to pay may be provided by another organization such as a bank (Y). So in this case, X1 is responsible and trusted for the authentication information that it provides on X1 members, whereas Y is responsible and trusted for the credit information for members from a number of such X organizations : X1,...,XN. This issue of distribution of trusted authorities is particularly important to understand when designing security protocols in large distributed environment such as the Internet.

References

- [1] National Computer Security Centre, Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, NCSC-TG-005, Vers.1, July 1987.
- [2] International Organization for Standardization (ISO), ISO/IEC JTC 1/SC 21/WG1: Security Framework Overview, Working Draft, 1991.
- [3] International Organization for Standardization (ISO), ISO/IEC JTC 1/SC 21/WG1: Authentication Framework, Draft International Standard DIS 10181-1.
- [4] International Organization for Standardization (ISO), ISO/IEC JTC 1/SC 21/WG1: Access Control Framework, Draft International Standard.
- [5] European Computer Manufacturers' Association (ECMA), Authentication and Privilege Attribute Security Application, Working Draft 1992.
- [6] M.Gasser, A.Goldstein, C.Kaufman, B.Lampson, *The Digital distributed security architecture*, Proc. of the 12th National Computer Security Conference, 1989.
- [7] Vijay Varadharajan, Phillip Allen, Stewart Black, *An Analysis of the Proxy Problem in Distributed Systems*, Proc. of the 1991 IEEE Symposium on Research in Security and Privacy, 1991.
- [8] Vijay Varadharajan, *A Security Architectural Framework for a Distributed Environment*, Proc. of the International Computer Security Conference, COMPSEC'91, London, Oct.1991.
- [9] J.Kohl, B.C.Neumann, J.Steiner, *Kerberos*, Version 5, Draft RFC, Project Athena, MIT, Dec.1990.
- [10] Charles Kaufman, DASS : Distributed Authentication Security Service, Network Working Group, Internet Draft, 1991.
- [11] Open Software Foundation, Distributed Computing Environment Overview, 1990.
- [12] OMG, The Common Object Request Broker (CORBA): Architecture and Specification, OMG Doc.No.91.12.1