CORBAmed Security White Paper

By Wayne Wilson <<mailto:wwilson@umich.edu>wwilson@umich.edu>
University of Michigan Medical Center,
and
Konstantin Beznosov
<<mailto:beznosov@baptisthealth.net>beznosov@baptisthealth.net>
<http://www.baptisthealth.net>Baptist Health Systems of South Florida
OMG document number: <ftp://ftp.omg.org/pub/docs/corbamed/97-11-03.html>corbamed/97-11-03
November 7, 1997

Introduction

The issue of security in healthcare has been discussed from a variety of perspectives at many CORBAMED meetings. This report focuses on the practical topic of how CORBAMED RFP's for services can go forward while accounting for security requirements. As we have found out, the exact nature of what is encompassed in the use of the word security can vary from person to person. Even if we start with a specific definition of security, requirements will also vary across a spectrum of viewpoints.

The paper consists of the following sections: an overview of the CORBA Security Service, a discussion of the CORBA Security Service from the standpoint of the healthcare vertical domain ("The Healthcare Perspective"), a summary of the paper results, and references. We provide a terminology section with both a summary and detailed explanations in the appendix.

How To Read This Paper

Those people who are very comfortable with the design of the CORBA Security service can brows the "CORBA Security Service Overview" just to see how security requirements for healthcare services are attempted to be resolved by using bare CORBA Security environment. Otherwise, they can go directly to the discussion of CORBA security from the standpoint of the healthcare vertical domain. Other people are strongly advised to read and try to understand the material presented in the "CORBA Security Overview" section. If during reading of the overview and/or the discussion section, you realize that you are confused about the meaning of such terms as "authentication", "authorization", "confidentiality", "integrity", "principal" and others, we advise you to read the terminology appendix. It provides definitions, examples, discussions and CORBA related descriptions of the key functionalities of distributed systems security.

The CORBA Security Service Overview

[This section is a reprint of "Guidelines for Security and CORBA implementation of the PIDS Access Policy Model" by Polar Humenn <polar@blackwatch.com>, sent to PIDS proposal mailing list the other day. If you do not know ideas behind PIDS, think about PIDS as one more healthcare domain CORBA service.]

Guidelines for Security and CORBA implementation of the PIDS Access Policy Model
by Polar Humenn <polar@blackwatch.com>

It is a requirement of the PIDS to provide <#Confidentiality>confidentiality of information that is stored about an individual. This requirement fuels the need for fine grained <#Authorization>access control on trait information that is associated with a PIDS identifier.

The PIDS access policy model governs the access to information associated with a single identifier. The PidAccess module defines this Identity interface that allows a client to gain an object reference containing information pertaining to only one particular identifier. Creating an CORBA object reference as a single accessor to the information yields benefits in controlling the access to that information. It is a single point to which access of this information must flow. Since CORBA security services can automatically deny requests as a result of an access policy, access to information behind this interface can be controlled by that mechanism.

Access to trait information is determined by a list of security attributes associated with each trait. These security attributes tag each trait with required client authentication credentials, such as a sensitivity level, role, and identity. The AccessPolicy interface is supplied in the PidAccess module to manage the <#Authorization>access control policy for an individual identifier. This interface manages the lists of security attributes associated with each trait .

Security Requirements

For the PIDS to be secure in its possible dissemination of information it needs to adhere to several requirements:

* The PIDS needs to authenticate a client's principal identity, role, and sensitivity level.
*
* The PIDS needs to transmit information confidentially and with <#Integrity>integrity.
*
* The PIDS needs to be consistent in its access policy.
* The first requirement states that the entire PIDS interface implementations must be able to identify a potential client. If it cannot authenticate a client, then the client may be severely limited in the particular requests that the PIDS can service.

The second requirement provides for the <#Confidentiality>confidentiality of the information. The client must communicate with the PIDS using not only encryption to protect data, but signature as well, so as not to have data tampered with during communication. There is no sense in putting a Sensitivity level of "OwnerOnly" on a trait and have its value transmitted to the owner in the clear.

The third requirement is the most important. The PidAccess::AccessPolicy interface maps easily into the PidAccess::Identity interface. However, although the AccessPolicy interface applies only to one identifier, the PidService::PersonLookup interface should not ignore the access policy model that is implied by the AccessPolicy interface. The PersonLookup interface should not match with or deliver up profiles of identifiers that contain traits tagged with security attributes to which a client should not have access. This requirement may bring up a multitude of issues regarding correlating domain managers, and even the PidChangeMgmt service.

The problem is, How does one get CORBA to support this access policy model?

CORBA Security Overview

In an effort to keep the PIDS interfaces security unaware, i.e. no extra visible security relevant parameters in methods, access

policy must be adhered to from behind the interfaces. The CORBA security model offers several ways to apply security policy
to method invocations.

However, the CORBA Security Specification [document number??] is not a cookbook for using CORBA security in building
applications. It is a general framework with which ORB vendors and application vendors can build a multitude of different
security policy models. The Security Specification also gives the interfaces for which implementations of applications can access
those security services that are supplied with a secure ORB.

A secure PIDS implementation must be aware of the security services offered by the ORB. Also, secure PIDS implementation
may have to be specific with respect to the ORB implementation and the security services it offers. This caveat also means that
a client of the PIDS service may have to know the kind of ORB and the security services that is used by the PIDS.

The CORBA security specification outlines a general security policy model. Although the specification is vague about which
approach should be taken, it is specific enough to be able to choose from a couple of models that can be supported.

The CORBA security model bases itself on credentials and security domains. Credentials are data objects that contain
attributes such as privileges, capabilities, and sensitivity levels, amongst others. Security domains are mappings from credentials
to access rights. Credentials can be encrypted and signed to prevent tampering and achieve a level of trust between client and
server. CORBA credentials get passed with requests beneath the visible level of the interface. CORBA security services give
the clients and servers the ability to authenticate/verify credentials to implement policies in security domains.

Many different schemes, algorithms, services, and vendor implementations exist to provide implementation of security policy,
and many different implementations of those schemes may be integrated into a CORBA compliant ORB. It is not the purpose
of this specification to dictate the specific implementation of an ORB and security services that should be used, but to outline
the external requirements for the PIDS implementation. These requirements and guidelines aid in selecting a secure ORB and
the level security functionality needed to implement the PIDS access policy model.

Secure Interoperability Concerns

CORBA has built the communication bridge between distributed objects creating a interoperable environment that spans
heterogeneous platforms and implementations. However, security adds another layer of complexity to the issue of
interoperability. ORB implementations are required neither to include security services nor provide an interoperable mechanism
of security services. However, a specification does exist for the target object to advertise, via the IOR, the security services that it supports and the services it requires from the client. Both the client and server ORBs must use compatible mechanisms of the
same security technology.

The CORBA Common Secure Interoperability (CSI) Specification [document number??] defines 3 levels of security
functionality that ORBs may support. The levels are named, CSI Level 0, CSI Level 1, and CSI Level 2. Each level has
increasing degrees of security functionality.

The CSI Level 0 supports identity based policies only and provides mechanisms for identity authentication and message protection with no privilege delegation. The CSI Level 1 adds unrestricted delegation. The CSI Level 2 is the full gamut that can implement the entire CORBA Security Specification at Security Level 2.

Each CSI level is parameterized by mechanisms that can support the level of security functionality, such as SPKM for CSI Level 0, GSS Kerberos for CIS Level 0 or CIS Level 1, and CSI_ECMA for CSI Level 2. Future developments in security functionality and mechanism are not restricted, and mechanisms can be added to each level.

The ORB implementations may use different security technology with differing capabilities and underlying mechanisms, such as SSL, DCE, Kerberos, Sesame, or other standards. Choosing the ORB and its underlying security services will be critical to protecting PIDS, and it will influence the implementation of the access policy that a secure PIDS implementation must support.

For example, an ORB that only supports SPKM, i.e. CSI Level 0, can only authenticate clients and provide <#Confidentiality>confidentiality and <#Integrity>integrity of communication. It cannot support definition and use of security attributes beyond an access identifier. Support for security attributes beyond an access identifier require CSI Level 2. Therefore, using an ORB that only provides CSI Level 0 will require the PIDS to maintain its own information on the credentials of clients.

However, even if an ORB's security technology supports the definition of security attributes that can be delivered to the PIDS, i.e. CSI Level 2, there are still concerns involving the trust between the client and the PIDS.

Trust Models

The available trust models for the PIDS is simplistic. Since the PIDS is an end point and does not require to make requests on other services on a client's behalf, a delegation trust model is not needed. This simplifies the model and eliminates an absolute need for a CSI Level 1 or CSI Level 2 secure ORB. However, this does not restrict the implementation of a secure PIDS to use a CSI Level 0 secure ORB.

There are two basic trust models for the PIDS. If the PIDS and its client are implemented using CSI Level 0 or CSI Level 1 ORBs, only the first trust model can be supported. If a CSI Level 2 ORB is used, both trust models can be supported. The trust models are:

* The client's identity can and is trusted to be authenticated. However, the client is unable or untrusted to deliver the valid credentials.
*
* The client is trusted to deliver the correct credentials.
* In the first model, the client ORB is required to authenticate its principal and provide authentication information to the server ORB. The methods used to accomplish principal authentication is specific to the mechanisms that the selected ORB supports. Management of those identities is also specific to the mechanism. The server ORB must have a compatible mechanism that verifies the authentication information and carries out mutual authentication with the client.

With this trust model, a secure PIDS implementation must maintain and manage a map of identities to privilege attributes. A

CSI Level 0 ORB is able to support this trust model.

However, even if the ORB has CSI Level 2 functionality, it may be a local policy that a PIDS does not trust the credentials brought forth from an authenticated client. In that case, the PIDS must maintain the map or use a default set of security attributes for requests from clients it does not trust.

In the second model, the client ORB is required to authenticate its principal and acquire its valid credentials. The methods used to accomplish principal authentication and acquisition of privilege attributes are specific to the mechanism that the selected ORB supports, such as DCE and Sesame. Management of those identities and attributes are also mechanism specific. A secure PIDS installation using this trust model must take a careful look at that management scheme and operation, evaluate it, and decide to trust it. In such a scenario, the server ORB, which has CSI Level 2 functionality, automatically verifies the credentials on invocation.

A secure PIDS built to the second model leaves management of identities and their attributes to the security services policy management system used by the ORB. The PIDS only manages security attributes for the data itself. This management is done with the PidAccess::AccessPolicy interface.

A secure PIDS built to the first model will have to have some scheme to manage trusted identities and their credentials. There is no interface or plan in the PIDS to specify this kind of management.

CORBA Credentials

To adhere to the credential model that most supports the PIDS's AccessPolicy interface, a set of credentials must contain privilege attributes such as the identity of the client, the role in which the client is actively representing, and the sensitivity level of information to which the client is allowed access. It will be the responsibility of a PIDS implementation to advertise to potential client vendors the specifics of these attributes and how to represent them externally. A client implementation needs to ascertain certain credentials and must pass them to the PIDS. An external representation of those credentials is needed so that credentials can be passed between client and server within the CORBA security services. The CORBA Security module defines the structure for this representation.

```
module Security {

    const SecurityAttributeType AccessId = 2;
    const SecurityAttributeType Role = 5;
    const SecurityAttributeType Clearance = 7;

    struct SecAttribute {
        AttributeType attribute_type;
        Opaque defining_authority;
        Opaque value;
    };

    typedef sequence<SecAttribute> AttributeList;
}
```

Listed above are the relevant pieces of the specification from the Security module that apply to externalizing credential information. The three security attributes of the PidAccess::AccessPolicy map isomorphically to three security attributes already defined in the CORBA Security module.

The Security::AccessId security attribute type should represent the owner of the PidAccess::Identity. In constructing the value of a Security::SecAttribute of this type, the defining authority part should be the name of the PIDS identifier domain manager, and the value part can be the identifier within that domain. However, if the ORB uses an underlying scheme where the value of the AccessId security attribute is supplied by some security services, such as a DCE name, a map to the PIDS identifier may be needed.

The Security::Role security attribute type should represent the mandatory role. Assuming that role names are defined by a PIDS, the same approach should be taken in defining a value for the SecAttribute. The defining authority part can take the name of the PIDS identifier domain that specifies the role, and the value can be the identifier within that domain.

The Security::Clearance security attribute type can be used to represent the Sensitivity Level. The values can be represented by the strings, "OwnerOnly", "LevelA", "LevelB", "LevelC", "None", and "Undefined". The defining authority can be "iso.org.omg.corbamed".

CORBA Security Domain Access Policy

In addition to a credential based scheme, CORBA defines security domains. The purpose of this section is to explain and illustrate the use of the standard CORBA security policy domain and the way in which it may be used to implement a security policy for the PIDS. This section offers a recommendation to a PIDS implementor in order to give a feel for the kinds of security policy a PIDS implementation may need to support. It should also guide the implementor in evaluating a secure ORB and available security services.

A security domain governs security (access) policy for objects that are managed within that domain. In order to make scalable administration of security policy, these domains map sets of security credentials to certain sets of rights. A right is a sort of an internal security credential.

CORBA defines a standard set of rights that are granted to principals within a security domain. A security domain administrator manages that map through the SecurityAdministration module's DomainAccessPolicy interface. Access decision then can be based on a set of required rights and the rights granted to the client by the domain access policy, by virtue of the client's credentials.

ORB security services vendors will supply a security policy management infrastructure that implements the standard CORBA rights scheme. The PIDS must use security services that can place different required rights on the PIDS interfaces. Some ORB security services may allow a security domain to create special rights. However, CORBA defines a standard set of rights: get, set, and manage. This right set will suffice to handle the PIDS.

Any number of domain models can be used, such as a separate security domain for each PIDS component. However, in this model, there is one security domain for all of the PIDS components. The CORBA rights families scheme within a single security policy domain suffices to differentiate the security nature of the methods.

The PIDS interfaces are divided up so that for most of the interfaces one right can apply to all methods of each interface. The table below recommends the required rights for each of the PIDS interfaces. An asterisk implies that the listed right applies to

all methods in the interface.

Interface	Required Rights
PidService::PersonLookup::*()	corba:g
PidService::IdDomainMgr::*()	corba:s
PidChangeMgmt::ChangeMgr::*()	corba:s
PidChangeMgmt::ChangeListener::*()	corba:s
PidCorrelation::CorrelatingMgr::*()	corba:m
PidCorrelation::QuerySet::*()	corba:m
PidAccess::Identity::get_trait()	corba:g
PidAccess::Identity::get_profile()	corba:g
PidAccess::Identity::update_trait()	corba:s
PidAccess::Identity::remove_trait()	corba:s
PidAccess::Identity::replace_profile()	corba:s
PidAccess::AccessPolicy::*()	corba:m
PidAccess::IdentityAccess::get_identity()	corba:g

Every method in the PersonLookup interface can be considered a "get" method. The domain access policy for the security
domain should grant authenticated clients with the proper access credentials, i.e. access id and role, with the get (corba:g) right.

All the methods in the IdDomainMgr interface and the PidChangeMgmt module can be considered "set" methods. They change information, and therefore these methods have a different security function other than the PersonLookup interface. A client that
is granted the right "get" should not necessarily allowed access to method that can change information. Clients that are allowed
to change information in the PIDS should be granted the set (corba:s) right.

The Correlating Domain manager module performs management of id domain managers, and therefore it is recommended that
access to this object should be more limited. This can be done a couple different ways, such as putting the correlating domain
manager in its own security domain. However, the manage (corba:m) right may be sufficient to separate this duty from the
others in a single domain.

The Identity interface is somewhat different. Access to its methods have the obvious get and set connotations. However, it is
unclear whether Identity objects created should remain in the same domain. This is an implementation of security policy issue.

Request Content Based Policy

The CORBA standard domain access policy scheme only protects methods from invocation at the target and without regard to
content of the request. The PIDS needs a more fine grained
<#Authorization>access control in order to implement the content based access policy required. The PIDS implementations must be made security aware to implement an access policy based on the value of
arguments in a request. There are multiple ways to implement this policy using a secure CORBA implementation.

The CORBA Security Specification supplies two different schemes represented by an interface hierarchy, which are Security
Level 1 and Security Level 2. These interfaces describe the level of security functionality that is available to security aware
implementations.

Security Level 1

For the PIDS to take advantage of CORBA security in order to implement its access policy model, the ORB must at least
implement the CORBA Security Level 1 interfaces. A Security Level 1 compliant ORB supplies an interface to access the

attributes of the credentials received from the client.

Using the Security Level 1 interfaces, which is simplistic, gives the implementation of the PIDS interfaces the ability to examine the client's credentials and compare them to the <#Authorization>access control information that is managed by the AccessPolicy interface. However, the implementation of the PIDS must be security aware.

```
module SecurityLevel1 {

    Current get_current();

    interface Current {
        Security::AttributeList get_attributes(
           in Security::AttributeTypeList attributes
           );
    };
}
```

Using the Security Level 1 interfaces, each implementation of a PIDS query interface must call the get_attributes() function on the Current pseudo object, examine the attributes, compare to the access policy information, and make the access decision. The implementation should raise the PidService::CannotAccess exception if access is determined to be denied.

It is the responsibility of the client's ORB to acquire the proper credentials securely. It is the responsibility of the server's ORB to authenticate credentials received from the client, extract the security attributes from them, and make them available to the implementation through the Current::get_attributes() method.

Security Level 2

Using an ORB which supplies the Security Level 2 interfaces, the implementation can be somewhat free of making the access control decision in the implementation of the query interfaces. Having an implementation that is security unaware is attractive in CORBA, because security policy decisions can be made underneath the functionality, and they have the ability to be changed without retooling the application.

As with any framework, there are several interpretations about the way in which to use the Security Level 2 interfaces. One approach could be to implement a replaceable security service for the access decision. Security Level 2 describes a method in which security can be enforced by the creation of an Access Decision object. The AccessDecision object would interact with a DomainAccessPolicy object to get effective rights and compare those to rights returned from the RequiredRights interface.

Some secure ORB implementations may allow the installation of specialized Access Decision objects to be used in conjunction with specialized DomainAccessPolicy objects. In the Security Level 2 interfaces, the specification implies <#Authorization>access control only on the invocation of a method and not the contents of the request.

```
module SecurityReplaceable {

    interface AccessDecision {
        boolean access_allowed (
           in SecuirytLvel2::CredentialList red_list,
           in CORBA::Object target,
           in CORBA::Identifier operation_name,
           in CORBA::Identifier interface_name
```

```
        );
    };
}
```

Currently, the AccessDecision object specified in the SecurityReplaceable module does not take the invocation Request as an argument. It only makes an access decision based on the credentials received from the client, the target object reference and operation name, and the target's interface name. This criteria is insufficient to implement the content based access policy needed by the PIDS to be automatically performed by the ORB.

Since the PIDS requires <#Authorization>access control on the contents of the method invocation (such as asking for the value of the HomePhone trait), this scheme of replacing these Security Level 2 components cannot be used. Also, ORB security services that use the standard CORBA domain access policy may use implementations for these components. This standard domain access policy functionality gives a the PIDS a high level of invocation protection that is orthogonal to the content based access policy. The PIDS needs the the standard domain access policy functionality in addition to a content based access policy. Therefore, another approach must be taken.

A content based access policy can be implemented in an Security Level 2 ORB by using an interceptor. A request level interceptor takes the Request as an argument and therefore, it can examine the content of the invocation arguments.

```
module CORBA {

    interface Interceptor { ... };

    interface RequestLevelInterceptor : Interceptor {
        void client_invoke( inout Request request );
        void target_invoke( inout Request request );
    };
}
```

Installing an interceptor on an ORB is ORB implementation specific, and each ORB vendor may have their own flavor of interceptors (Orbix has "Filters"). The ORB calls the request level interceptor just before the invocation gets passed to the server implementation by using the target_invoke() operation. The interceptor uses the Dynamic Skeleton Interface (DSI) to examine values of the arguments of the invocation and make access decisions. These access decisions are also based on the credentials received from the client and the access policy implied by the PIDS AccessPolicy interface. The interceptor will deny access to the invocation by raising the PidService::CannotAccess exception. The server's ORB will transmit this exception back to the client.

The use of the interceptor scheme frees the implementation of the PIDS interfaces from the implementation of the access decision policy. If the access policy model changes, then the interceptor can be changed out without retooling the PIDS implementation.

As awareness of the need for more powerful and flexible security policy management grows, more tools to create, manage, and analyze policy will come into existence. A PIDS implementation relying on interceptors to implement its security policy may be

able, with relative ease, to switch to using more robust policy services as they become developed. [End of Extract]

The Healthcare Perspective

CORBAmed's interests in security consist of the following: authentication of principals, <#Authorization>authorization of principles, <#Confidentiality>confidentiality of messages, confidentiality of information, and <#Audit>audit of state changes of any of the preceding items. We will also provide a definition of each technical term we use as hot links within the document. It has been our experience that much discussion can take place over these definitions, so we make no claims for their universality. We only claim that when we use them in this report, that is what they mean.

CORBA has a security service specification. One of our primary tasks will be to explain how CORBA security services can be used and what they will accomplish according to our interests. We will both explain security functions and discuss the implications from the point of view of the healthcare domain.

Our conclusions are that base CORBA security services are not extensive enough for the healtcare domain. There are several courses of action open. We believe that CORBAmed must decide how to proceed on these before completing any specifications. Briefly, CORBA security is complete enough with the exception of the granularity of security policies, namely fine <#Authorization>authorization capabilities. If the application becomes security aware it could make it's own authorization decisions and this facility is supported.CORBA security is restricted to authorization of method invocations, ignorant of method parameters. >From a design viewpoint, this means that if we build all services such that the authorization requirements fall out on methods, or in other words, if we design our applications such that authorization is provided along functional dimensions, then CORBA security is all we need. If, however, authorization is dependent not only on the functional capabilities of the interface, but also on data content or method parameters, CORBA security is not enough.

As we see it, the options for providing content within function authorization are the following:

* **Security Aware applications (provided for by CORBA security)**
*
* **Third party add-on's to CORBA security**
*
* **New RFP issued for Healthcare Security Framework**
*
* **Agreement on application security interfaces among vendors**
*

We recommend the third option. Both Security aware and add-on's suffer from inter-operability problems. Inter-operability is most impaired with application awareness as each implementor of an application module must use the same security model and implement the security functions correctly. If one imagines a large scale Healthcare enterprise with thousands of modules implemented by dozens of vendors, the expectation that they would all have used the same underlying (invisible to the functions of the interface) model and implemented it correctly in a thousand different places is unrealistic. With add-on's, the problem is greatly reduced as the add-on is applied outside of the interfaces and their implementations. But, how add-on's perform their functions or even define their functions will be a vendor dependency. That means the federation of services in a rapidly changing business climate will depend upon each player having chosen the same add-on vendor. True, the swap out of one vendor's add-on for another only involves rebuilding the security policy tables and touching each point of service in the infra-structure without changing application codes, but this is still a considerable amount of work.

We believe that along the range of possibilities for content within function <#Authorization>authorization, a fairly simple "80%" solution can be achieved. The work of preparing an RFP would involve specifying security functionality in terms that leave implementation choices. We ask that CORBAMED initiate work to that end.

We believe that access to healthcare information will need to be controlled across the following dimensions:

* **Method input argument values**
*
* **Return result values (returned data content)**
*
* **Time of service request**
*
* **Location of service requester**
*
* **Principal ID**
*
* **Principal role**
*
* **Method invocation**
*

In the subsection below, we provide several examples from healthcare service specifications. The specifications are in CORBAmed adoptation process. The examples illustrate our opinion that <#Authorization>access control has to be excersised on the level of method input argument values and return result values.

Examples of Fine Grain Access Control

This section provides a few examples of interfaces to some CORBAmed specifications that are either adopted or are in the process of adoption by the task force. The examples illustrate a need in the healthcare vertical domain to exercise access control on the level of method input argument values and/or return values.

All current examples are brought from <#[PIDS,4]>Person Identification Service (PIDS), Initial Submission Revision 4 (19 October 1997). The interfaces are from module org/omg/PersonIdService.

Access Control on Input Argument Value Level

These queries require <#Authorization>access control on the level of input argument values.

```
interface DemographicAccess :
        IdentificationComponent
{
  Profile get_profile (in PersonId person_id, in SpecifiedTraits
specified_traits)
      raises ( InvalidID, UknownTraits, DuplicateTraits );

  void update_traits( in PersonId id, in Profile the_profile )
        raises ( InvalidId, UnknownTraits, ReadOnlyTraits, DuplicateTraits );
};
```

"The get_profile() operation returns the profile or subset of the profile that the service knows about the person. The passed in traits indicate what subset of the profile that is being sought by the client." <#[PIDS,4]>[PIDS,4] The service is supposed to control what <#Principal>principal can access what traits of what person_id.

The update_traits() "is used to modify the profile of an already existing ID by adding in new traits and over writing any values for existing traits that are also passed in. Any traits already stored for that person but not mentioned in

the passed in profile are left intact." <#[PIDS,4]>[PIDS,4] The service is supposed to know what principal can update what the_profile of what person id.

Access Control on Return Value Level

These queries require access control on the level of return values. Note that a method can return results via return value as well as via out and inout arguments.

```
interface SequentialAccess :
    IdentificationComponent
{
 ProfileList get_all_ids(in TraitNameSeq traits_requested, in IdStateSeq
states_of_interest)
      raises ( TooMany, UnknownTraits, DuplicateTraits, InvalidStates );
};
```

Operation get_all_ids() "returns profiles for all patients the service knows about that match one of the states passed in. The returned profiles only contain the traits indicated by the passed in parameter." <#[PIDS,4]>[PIDS,4] The service is not supposed to return those profiles that the principal does not have access to.

```
interface IdentifyPerson :
    IdentificationComponent
{
    void find_candidates( in ProfileSelector profile_selector, in Constraint
constr,
        in IdStateSeq states_of_interest, in unsigned long max_sequence,
        in unsigned long max_iterator, out CandidateSeq id_sequence,
        out CandidateIterator id_iterator )
   raises ( TooMany, UnknownTraits, DuplicateTraits, InvalidAlgorithms,
           InvalidStates, InvalidConstraint, InvalidWeight );
};
```

"Knowing some identifying information about a person (or group of people with common traits)[,] a client can ask the service [by invoking find_candidates()] to find the candidate persons that the service thinks may match those traits." <#[PIDS,4]>[PIDS,4] The service is not supposed to return those IDs that the principal does not have access to.

Conclusions

In this paper, we discussed the CORBA Security service from the perspective of the healthcare vertical domain. CORBA security does not suit the needs of the healthcare domain perfectly. Lack of specified finer granularity security policies would lead vendors to proprietary solutions on top of secure ORBs. CORBAmed will better suit healthcare's needs if it proactively standardizes the ways such policies are enforced and managed in the CORBA-based healthcare services. The first step is to issue a corresponding RFP.

References

[ISO 7498 - 2] Information Processing Systems, Open System Interconnection, Basic Reference Model, Part 2: Security Architecture [Karjoth, 1996] Gnter Karjoth, Analysis of Authorization in CORBA Security, IBM Research Division, Zurich Research Laboratory, 1996 [MMS, 1996] Massachusetts Medical Society House of Delegates, Massachusetts Medical Society Policy: Patient Privacy and Confidentiality, <http://www.massmed.org/whatsnew/privacy/>http://www.massmed.org/whatsnew/privacy/,1996 [OMG, 1996] Object Management Group, CORBAservices: Common Object Services Specification, Security Service Specification, 1996 [PIDS,4] OMG CORBAmed DTF, Person Identification Service (PIDS), Initial Submission - Revision 4, <http://www.protocol.com/engineering/corbamed/rev4/pids.pdf>http://www.protocol.com/engineering/corbamed/rev4/pids.pdf, 1997

Appendix

Concepts and Terminology

This section gives background on distributed systems security that is necessary to understand the paper. It also defines terminology used in the paper to ensure adequate understanding of terms. We provide everyday definitions of terms to get the idea across as well as formal definitions for those who are more comfortable with pedantic style. We also give examples to support definitions as well as synonyms of defined terms for those who are used to slightly different terminology.

Since the paper's intended audience has diverse background in the distributed system security field, we provide various means to comprehend terminology. For each key term we list informal as well as formal definitions, examples, synonyms. Also, we discuss relevant issues (like common confusion with other terms, etc.) and relate CORBA Security specification (CORBASEC) with the terms. The section can be read in consequent order for those who want to refresh memory on security concepts and terminology, or it can be used for finding definitions of particular terms.

Security can mean different things for different people and in various situations. For example, airport security service is mostly concerned with protecting buildings and planes from terrorist actions. Here, we will consider only information security and services security of distributed computer systems, or "distributed systems" for short. Security of such distributed systems should mean the same no matter what enterprise those systems are deployed at.

Distributed Systems Security

Goals

In distributed systems, security protects an information system from unauthorized attempts to access information or interfere with its operation. Security services are concerned with the information system's:

* ***Confidentiality***. *Information is disclosed only to users authorized to access it.*
*
* ***Integrity***. *Information is modified only by users who have the right to do so, and only in authorized ways. It is transferred only between intended users and in intended ways.*
*
* ***Accountability***. *Users are accountable for their security-relevant actions. A particular case of this is non-repudiation, where responsibility for an action cannot be denied.*
*
* ***Availability***. *Use of the system cannot be maliciously denied to authorized users.*
*

<#[OMG,>[OMG, 1996]

General security goals above include more specific concerns. Confidentiality, for instance, is concerned not only with communication confidentiality but also with such threat as memory reuse in operating system environment. It happens when an application frees memory which contains sensitive data and another application gets access to the data by allocating memory that happens to have the data. Another threat that confidentiality is concerned with is the failure of some applications to clean disk cash with sensitive data.

Functionalities

Now we will describe the main functionalities, which are to be present in CORBA security services to achieve the goals listed in the previous section.

Authentication

Definition: Authentication functionality of system security is responsible for making sure that a user or a service is who they claim to be.
Formal Definition: "The corroboration that an entity is the one claimed" <#[ISO>[ISO 7498-2]
Examples:

* When Jeff logs into a distributed system, the authentication functions of it are responsible for making Jeff to prove that he is Jeff.
*
* When you access a web service somewhere on the Internet, you sometimes want to make sure it is the server it claims to be.
* Synonyms: Sometimes, the word "identification" is used instead of "authentication" to mean the same.

Discussion: Authentication part of security deals only with user or service identities. It is not responsible for access control, confidentiality or any other security functionalities. Though, it might use confidentiality and integrity to protect information exchanged between the system and say the user during the authentication phase.
CORBASEC: Alone with other security models, CORBASEC uses the term **principal** to refer to a human or system entity that is registered in and authenticatable to a distributed system.

The specification introduces an entity called "user sponsor." The user sponsor is responsible for receiving authentication data from a user and for calling on the Principal Authenticator object, which authenticates the user and obtains user Credentials containing authenticated identity and privileges (see discussion on CORBASEC and Authorization below for information about privileges). User sponsor is not specified in CORBASEC since it is considered to be an implementation-dependent part of the service.

Authorization
Definition: Authorization functionality is responsible for making decisions about what users and what services can access what system services and for endorsing those decisions.
Formal Definition: "The granting of rights, which includes the granting of access based on access rights." <#[ISO>[ISO 7498-2]
Example:

* Authorization functions of a system are responsible for making decisions whether to give user *jeff* rights to print a document on the printer located in the mail room on the third floor.
*
* Authorization is responsible for enforcing the following hypothetical policy: only enterprise services can access company directory that contains employee information.
* Synonyms: "Authorization" and "access control" are used interchangeably to mean the same.

Discussion: Authorization cannot be enforced without reliable authenticating functionality of a system. Before access rights decisions can be made, it is critical to identify a user or a service. Authorization decisions are based on access control policies. Such policies can be very rudimentary ("grant access to anyone") or very complex ("Give access to HIV information of patient X only to a user who has status of 'Attending physician for patient X' when such a user is located at her hospital office and only if the patient X gave a consent to disclose her HIV information and when it is before 2 weeks after the patient X was discharged").

Access control policies are expressed in the form of access control rules. A set of access control rules constitute access control language that allows mapping of the application system business model into the access control model supported by the particular distributed system authorization services.

Unix access control rules are a good example of a basic access control language. In Unix, each resource including processes, files and devices is owned by some user (owner) and group. Unix access rules specify what type of

access right (read, write, execute) is granted to the resource owner, group, and the rest of the world in regard to this resource. In order for a user to perform access operations granted to its owner, the user has to have the same identity as the resource owner; to perform group access operations, the user has to be a member of the same group as the resource; to perform operations allowed to "the rest of the world" the user does not have to have any special rights. Unix access rules are therefore simple:
If you are the *owner* this what access operations you can invoke, otherwise
if you are a *member of the same group*, this is what access operations you can invoke, otherwise
you have the same access rights as anyone else in the system.

CORBASEC: The CORBASEC access control model is based on:

* **Privilege Attributes** that users or services can posses and that are used by security services to compute,
*
* **Granted Rights** for a given user or service in the context of particular
*
* **Access Control Domains** that are used to partition objects, and
*
* Rights required to invoke particular method on a target implementing a particular interface in the given domain (**Required Rights**).
*

CORBASEC access rules can be roughly formalized to the following language: For a given **domain** D, and a **delegation state** d

if you have a **privilege attribute** a_1 with value v1in a **delegation state** d then you are assigned **granted rights** $R_{11},R_{12}...R_{1P-1},R_{1P}$, and
....
If you have a **privilege attribute** a_N with value vN in a **delegation state** d then you are assigned **granted rights** $R_{N1},R_{N2}...R_{NT-1},R_{NT}$.
You can invoke **method** M on any instance of **interface** I in this **domain** D if and only if you have **any** or **all** (but not mixed) of the following **required rights** -- R1,R2...RQ, .

Where:

* Where delegation state d can have one of two values: "initiator" or "delegate."
*
Privilege attribute type can be one of the following CORBASEC standard privilege attributes: Public, AccessId, PrimaryGroupId, GroupID, Role, AttributeSet, Clearance, Capability. There are also AuditId, AccountingId, NonRepudiationId attribute types.
* Right Rxy is either OMG defined right or some other right. OMG standardized the following three rights: "get", "set", and "manage" of family "corba".
*

As you can see, there are three separate multidimensional discrete spaces:

* Privilege attributes
*
* Granted Rights
*
* Required Rights
*

Authorization services of CORBASEC maps user or service privilege attributes obtained during the authentication process into a space of granted rights for a given domain and a given delegation state. Then, granted rights are matched via the "and" or "or" function with required rights for a given interface.method entity. If the match is successful, the user/service is granted access to invoke that method on that interface.
More formal analysis of CORBASEC access control can be found in <#[Karjoth,>[Karjoth, 1996].

Information Integrity
Definition: Integrity service is responsible for providing the protection of data from unauthorized modifications.
Formal Definition: "The property that data has not been altered or destroyed in an unauthorized manner." <#[ISO>[ISO 7498 - 2]
Example: When we send a message via electronic mail service, we want our message not to be altered on its way to the destination. In other words, we want guaranteed integrity of information sent over e-mail.
Synonyms: Term "integrity" does not have synonyms.
Discussion: Since it is almost impossible to enforce access control over information traveling through multiple intermediate hops in inherently insecure networks, integrity becomes a very important asset of secure communications in distributed systems. In most of traditional systems that provide secure communications, integrity is achieved by "signing" messages digitally. The idea of digital signatures comes from check-sum computation in communication protocols. The main difference between check-sums and digital signatures is the ability to ensure that the signature was generated by the original sender.
CORBASEC: The specification hides underlying technology that provides integrity functionality from application developer. It provides the notion of Quality of Protection (QoP). Integrity in QoP can be enforced via corresponding policy or by security aware application.

Information Confidentiality
Definition: Information confidentiality functionality is responsible for protecting information from unauthorized disclosure.
Formal Definition: "The property that information is not made available or disclosed to unauthorized individuals, entities or processes." <#[ISO>[ISO 7498 - 2]
Examples:
* Sending letters in sealed envelopes as opposed to postcards is a well known computer unrelated example of confidentiality services. By enclosing your letter in an envelope, you protect its contents from being accessed by anyone else but its intended reader.
*
* In computer communications, confidentiality is usually achieved by encrypting information and making only sender in a position of decrypting the received data.
*
* Making sure that information left in a system after an application is not read by any other application is also responsibility of confidentiality services. CORBASEC does not address such issues directly since they are under control of applications themselves or operating systems where those applications are executed.
*

Synonyms: Terms "confidentiality functionality", "confidentiality service" and "confidentiality" are used in this document interchangeably to mean security environment function responsible for information confidentiality. The latter is defined in this subsection.
Discussion: Confidentiality function of security services is sometimes confused with authorization function since its definition mentions "protection from unauthorized disclosure." We would argue that confidentiality begins to play an important role when authorization functions cannot serve their purpose. For example, when data is sent over insecure networks, there is no way to enforce access control to the data. This is when confidentiality functions of a system are employed.

Another confusion is between terms "confidentiality" and "confidential." They are not equivalent! "Confidential" is used to express the sensitivity level of particular information.

Many with the healthcare background use terms "privacy" and "confidentiality" interchangeably. It is due to the fact that the healthcare domain puts very

different meaning in the word "confidentiality" than the technically oriented computer security world. The Massachusetts Medical Society Policy on Patient Privacy and Confidentiality explains the meaning of "confidentiality" in the healthcare domain and its difference from "privacy" well: [Confidentiality means] *the right to rely on the trust or discretion of another; the right of an individual to control access to and disclosure of private information entrusted to another. Although the words privacy and confidentiality often are used interchangeably, they are related but not synonymous terms. Privacy derives from the concepts of personal freedom and autonomy, and involves the ability of an individual to control the release or dissemination of information that relates to him/herself. Confidentiality, on the other hand, arises in a relationship, when an individual gives private information to another on the condition of or with the understanding that the other will not further disclose it, or will disclose it only to the extent that the individual directs.*
<#[MMS,>[MMS, 1996]
In this paper, we will use the term "confidentiality" only in the context of distributed system security. To avoid any confusion, we do NOT use the terms "privacy" or "confidentiality" in the healthcare domain meanings.
CORBASEC : The specification hides underlying technology that provides confidentiality functionality from application developer. Confidentiality in Quality of Protection (QoP) can be enforced via corresponding policy or by security aware applications. An administrative policy or an application can enforce confidentiality, integrity, or both, or nothing for communications. CORBASEC does not allow specifying HOW confidentiality or integrity is enforced. It provides an interface independent from underlying technology.

Accountability
Definition: Accountability functionality is responsible for making users accountable for their security-relevant actions.
Formal Definition: "The property that ensures that the actions of an entity can be traced." <#[ISO>[ISO 7498 - 2]
Examples:

* Non-computer example of accountability service is registered (certified?) mail when a receiver signs a slip upon reception of a mail item and the signed slip is mailed back to the sender. Registered mail allows senders to protect from receivers denying the fact of receiving a particular mail item.
*
* There is an analogy of registered mail in electronic mail when a sender receives confirmation that a receiver got the messages in their incoming mailbox.
*

Synonyms: There are no terms used interchangeably with the term "accountability."
Discussion: Accountability service is an important part of any security system since it provides virtually the only way to monitor security activities in the system and to detect security breaches as well as to provide proof that a particular action was requested and/or a particular message was sent/received later in court.
Accountability requires authentication to have reliable information about identity of involved parties.
CORBASEC: Accountability in CORBA Security services is achieved via security audit and non-repudiation functionalities.

* Security audit goal is to facilitate "an independent review and examination of system records and activities in order to test for adequacy of system controls, to ensure compliance with established polices and operational procedures, to detect security breaches and to recommend any indicated changes in control policy and procedures." <#[ISO>[ISO 7498 - 2]
*
* Non-repudiation functionality is to protect against originator of a message or action denying that it originated the message or the action as well as

against the recipient of a message or action denying that he or she has received the message or was requested action.
*

Interface to security audit facilities is provided by level 2 compliant implementations. Security aware applications can find out whether an audit is needed according to current audit policies and they can write into abstract audit channel, which isolates an application from a particular audit technology.
Non-repudiation functionality is optional for CORBA compliant implementations of security services.

Availability
Definition: Availability functionality of security is responsible for ensuring that the system is available to authorized users.
Formal Definition: "The property of being accessible and useable upon demand by an authorized entity." <#[ISO>[ISO 7498 - 2]
Examples:
* Protecting components of a distributed system from denial of service attacks.
*
* Introducing redundancy into a system increases its availability.
*

Synonyms: Term "availability" does not have synonyms.
Discussion: Availability function is sometimes confused with reliability. The latter is a measure of how few failures happen to the system components. Naturally, the more reliable system components are, the higher availability of the system. The reverse is not always true because availability can be (and usually is) achieved by other means than increase of reliability. Availability is a functionality of system security because most of security breaches potentially decrease overall system availability.
CORBASEC: The specification does not specifically address availability functionality of CORBA Security services because it is the current specification authors' opinion that "availability is often the responsibility of other OMA components such as archive/restore services, or of underlying network or operating systems services." <#[OMG,>[OMG, 1996]