CORBA Firewall Security: Increasing the Security of CORBA Applications[1]

Habtamu Abie
Norwegian Computing Center
P. O. Box 114 Blindern, 0314 Oslo, Norway
abie@nr.no, http://www.nr.no/~abie
January 2000

Abstract

Traditional network firewalls prevent unauthorised access and attacks by protecting the points of entry into the network. Currently, however, there is no standard mechanism by which a firewall identifies and controls the flow of Internet Inter-ORB Protocol (IIOP), that has become the de-facto standard interoperability protocol for Internet providing "out-of-the-box" interoperation with ORBs, and is based on vendor-neutral transport layer. The OMG's intention in proposing its CORBA Firewall Security is to provide a standard approach to the control of IIOP traffic through network firewalls, allowing controlled outside access to CORBA objects, thus increasing their accessibility and security. This article describes and analyses the OMG's CORBA Firewall Security, paying special attention to such issues as the specific problems associated with it, how current firewall techniques are used to control CORBA based communication, their potential limitations and how these might be overcome, and the various aspects of firewall traversal. In addition, a possible CORBA firewall application scenario is presented. Some CORBA Firewall compliant products are emerging on the market, and this current trend in the implementation of CORBA firewall products will also be described.

Keywords: CORBA Firewall Security, Object Access Control, Computer Network Security.

1 Introduction

Nowadays networks are subject to continual change and modification as they are adapted to changing circumstances and new situations brought about by reorganisations, acquisitions, outsourcing, mergers, joint ventures and strategic partnerships. In addition, networks are increasingly connected to the Internet. Due to these developments, the maintenance of security has become a far more complicated matter than hitherto. Common Object Request Broker Architecture (CORBA) has become the de-facto standard. Its extensive infrastructure supports all the features required by new business situations of the type mentioned above, and its increasing use in open systems necessitates the development of sophisticated security technologies at the interface between networks of different security domains such as between Intranet and Internet or Extranet. The best way of ensuring interface security is the use of a firewall.

[1] Telektronikk Volume 96 No. 3-2000, pp.53-64

Conventional network firewalls (see [1] for an overview of firewall technologies) prevent unauthorised access and attacks by protecting the points of entry into the network. Currently, however, there is no standard mechanism by which a firewall identifies and controls the flow of IIOP protocol, since IIOP has become the de-facto standard interoperability protocol for Internet providing "out-of-the-box" interoperation with Object Request Brokers (ORBs). The Object Management Group (OMG) [11], a non-profit consortium with a current membership exceeding 840 organisations, whose purpose is to promote the theory and practice of object technology in distributed computing systems, is the body responsible for setting standards and specifications for CORBA Firewall Security. The purpose of the OMG's CORBA Firewall Security is to provide a standard approach to control IIOP traffic through network firewalls, allowing outside access to CORBA objects, thereby increasing their security. This article discusses CORBA Firewall Security with the emphasis on such issues as the specific problems associated with it, how CORBA communication can easily and securely be handled by firewalls, how current firewall techniques are used to control CORBA based communications and their potential limitations, and how to overcome such potential limitations. It also describes various aspects of firewall traversal, IIOP/SSL, callbacks, desired proxy behaviour, chaining or pass-through mode for IIOP/SSL, and CORBA interworking through firewalls.

In addition, this article assesses the CORBA Firewall implementation technologies available on the market, such as WonderWall of IONA, Inprise's Gateway, ObjectWall of Technosec, NAI's ORB Gateway, etc. This discussion is essential to an understanding of current trends in the development of CORBA firewall products.

2 CORBA Firewall Security - an Overview

CORBA is widely available, provides an extensive and mature infrastructure, and plays a crucial role in integrating many existing enterprises, and has thus become today's most important system integration technology for distributed applications. The increasing use of CORBA in open systems requires sophisticated security technologies to isolate networks or sub-networks that are in different security domains. A security domain here means a network or sub-network under common administrative control, with a common security policy and security level. The domain boundary may be between Intranet and Internet or Extranet. The appropriate means to enforce security policies at the boundaries between security domains are firewalls.

The aim of OMG's CORBA firewall is to improve accessibility to CORBA application servers when there is a firewall separating a server from a client. It makes it easier to enable and control client-firewall-server communication under a broader range of circumstances with significantly reduced administrative burdens. Interoperable CORBA communication is via the General Inter-ORB Protocol (GIOP), which on the Internet is implemented by IIOP (a mapping of GIOP to TCP transport). Because firewalls control IP networking communication, and because ORBs communicate via IIOP, a large part of the activity of the CORBA Firewall is dedicated to the various operations involved in handling IIOP traffic through a firewall [9].

The main function of the CORBA Firewall is thus to recognise an IIOP message, process it, and then allow it to pass through providing fine-grained access control. CORBA firewall technology is applied to both inbound[2] and outbound[3] protections. It processes requests from objects outside the firewall wishing to invoke operations on objects inside the firewall, and requests from client objects inside the firewall wishing to use CORBA-based applications outside the firewall on the Internet or Extranet.

As already pointed out, in a CORBA environment, firewalls protect objects from client objects in other networks or sub-networks. A firewall will either grant access from another network to a particular object or deny it. When it grants access, it can do so at different levels of granularity. For example, access to selected objects may be granted, but not to others, and, similarly, access to selected operations within a given object may be granted, but not to others. Firewalls have two distinct functions: inbound and outbound protections. Outbound protection involves allowing authorised client objects to initiate communication with objects outside the enclave (see below), while preventing those not authorised to do so from doing so. Inbound protection involves granting authorised outside client objects access to inside objects while denying unauthorised outside objects access. With no outbound protection, clients would be able to access any outside resource, and with no inbound protection the contents of an enclave would be totally unprotected against the (frequently malicious and destructive) machinations of the world at large.

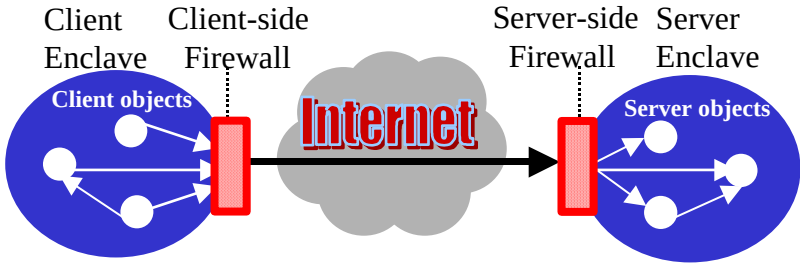

Figure 2-1: Object invocations through firewalls over the Internet

Figure 2-1 shows the basic concept of CORBA object invocations through firewalls in an Internet environment. An enclave, as shown in the figure, is a group of CORBA objects protected by the common firewall which controls all network communication between them and the outside world. Enclaves can be nested, so that an enclave may contain other enclaves arranged hierarchically on the basis of different access policies. The figure shows a client object calling an operation on a server object in another enclave. The client object can communicate directly with all objects in its own enclave, but must communicate through the firewall with objects outside.

It should be noted that outbound protection involves granting or denying inside objects access to the outside world, but in most cases does not involve restricting which operations they are permitted to invoke on which outside objects. On the other hand, inbound protection involves not only denying or granting outside objects access to inside objects in the enclave, but also restricting which operations they are permitted to invoke on which inside objects.

[2] *Inbound protection* is used to control external access to internal resources.
[3] *Outbound protection* is used to limit the outside resources to be accessed from within the enclave.

3 Specific Problems Associated with CORBA Firewalls

Unlike traditional firewalls, CORBA Firewall Security addresses some of the specific problems associated with addressing, callbacks, encryption, cascading of firewalls, transparency, and integration into security systems [9, 13].

3.1 Addressing

Today's standard firewalls with static configurations are not well suited for dynamic CORBA applications because they work on the assumption that a client will always use a fixed, designated port on the basis that a given type of server always listens at that particular port. An HTTP server, for example, always listens at port 80. A CORBA server object, however, does not listen at a specific, designated port. While it is possible to bind a CORBA object to a specific port in the case of simple applications, this is not possible in most cases, because CORBA applications and ORBs generally use objects launched at arbitrarily selected ports.

Since it is not usually possible to predict which hosts and ports will be used for inter-enclave CORBA communication, it is difficult to configure firewalls in this situation. While the Interoperable Object Reference (IOR) provided by the server, containing host/port addressing information, is sufficient within the enclave, since no firewalls are involved, it does not help client objects from outside as they are unable to reach the server object directly. Instead of the actual address of the server, the client needs a proxified address, the address of the firewall. The firewall will process the request and forward it either to the server or to the next firewall. The address of the outbound firewall on the client side can be configured manually, but the IOR containing the address of the inbound firewall on the server side must be supplied by the server.

3.2 Callbacks

Traditionally in a client/server system, there is a very clear and sharp distinction between client and server. The server accepts connections from the client, but not vice versa.

This is not the case in a CORBA object system, a system of communication objects. In many cases it is desirable for a CORBA object server to contact a client object, for example, to facilitate asynchronous information flow. This is achieved by the client creating a callback object, the reference to which, an IOR containing the address of the callback object's inbound firewall, is passed by the client to the server. The server can then contact the callback object through its own outbound firewall and the callback object's inbound firewall.

It should be noted that this is not possible where the client-side ORBs have been downloaded as Java applets since these applets are not permitted to accept inbound connections or to create objects in the client space, and neither do they have any knowledge of the inbound firewall.

3.3 Encryption

Encryption technology is the ideal protection technology, especially for protecting communication over the Internet or other insecure networks against tapping and tampering. In CORBA communication, if the CORBA requests contain confidential information the use of encryption is the only way of protecting it.

Where firewalls are involved, there are three different patterns of encrypted connection: 1) between client and server, 2) between firewall and firewall, and 3) between client and firewall and firewall and server. This means that CORBA firewalls must be able to handle any combination of these three patterns.

3.4 Cascading of Firewalls

In many cases an invocation of an operation passes from client to server through more than one firewall. Figure. 3-1 shows such a situation. In the figure we see how an invocation passes from a client object in the enclave of a group, for example the Research Lab, over the Internet, to a server object in a server enclave. It passes from the client object to the firewall of the Research Lab, then to the firewall of the department, then to the firewall of the organisation, then over the Internet to the server side firewall, and finally to the server object. Such cascading presupposes the ability of invocations to pass through a series of firewalls even when the latter do not use the same technology.

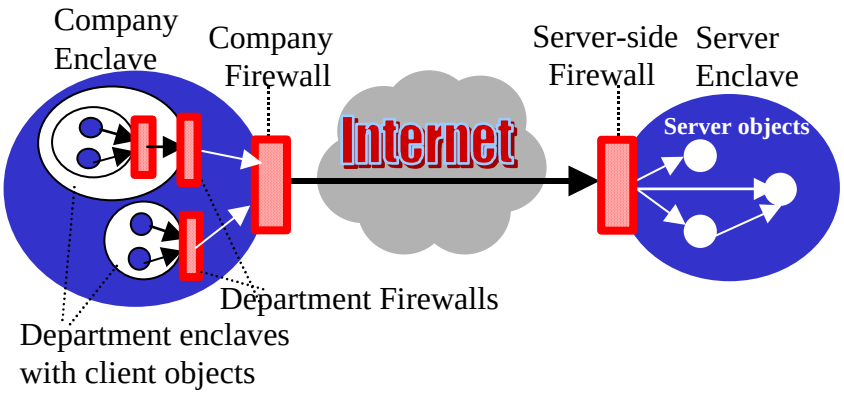

Figure 3-1: Cascading of Firewalls in Nested Enclaves

3.5 Transparency

A CORBA firewall should be transparent to users (application programmers, administrators and end-users). Schreiner [13] has proposed a transparent firewall that allows a client object to use the normal IOR of the server object instead of a proxified address. The client attempts to connect to the server object as though no firewalls were present. The firewall intercepts the request and launches a proxy object, which relays the request to the server object.

In Schreiner's opinion this will make things simpler for programmers and administrators since it obviates the need for proxification and will only work if the incoming side uses an officially assigned IP address. It occurs to me, however, that the use of officially assigned IP addresses will limit the use of dynamic IP addresses.

Will the use of interceptor and smart agent technologies be the trend in the development of the new generation of firewall technologies?

3.6 Interoperability

As described earlier, firewalls have two distinct duties, inbound protections that are used to control external access to internal resources, and outbound protections that are used to limit the outside resources that should be accessed from within the enclave. In order for an ORB to establish a connection to an object in another ORB, outbound and inbound firewalls that need to be traversed must be known.

Information about outbound firewalls is specified in the client side ORB, and information about inbound firewalls may also be specified in the case of Intranet and Extranet configurations. In general, however, the client side knows nothing about the server side, so that the only interoperable way of giving the client access to the information about inbound firewalls is to have this information included in the IORs provided by the server.

3.7 Management

A CORBA firewall needs a secure interface through which administrators can configure it remotely and manage security policies. The functionality of this interface should permit the security policy to be configured dynamically. Preferably the firewall should be auto-configurable. Such functionalities are of paramount importance in the dynamic world of the Internet, since closing down a site is inconvenient and can be costly. The management functionality must also support key management interfaces if the firewall is to be involved in the encryption and decryption process as described above. CORBA firewall management should also be integrated into the organisation's overall security management systems, especially into CORBA Security Services [10] management.

4 The OMG Firewall Proposal

The Joint Revised Submission on CORBA Firewall Security (submitted in response to the OMG Firewall RFP (Request for Proposal)) [9] specifies the use of IIOP in network firewalls for controlling access to CORBA-based applications from Internet, Intranet or Extranet, and specifies and describes how inter-ORB interoperability through such firewalls can be achieved. According to the Joint Revised Submission, firewalls are broadly speaking of two kinds (in contrast to Figure 4-1), transport level and application level firewalls. The former permit or deny access to different resources using different application level protocols on the basis of addressing information in the headers of transport packets, and thus on the basis of where things are coming from or going to, and not what is being accessed. The latter, on the other hand, are, in addition, restricted to granting or denying access to particular application level protocols, such as IIOP or HTTP, and to those resources known to the application protocol. Consequently, they can grant or deny access on the basis of both addressing information and specific resources.

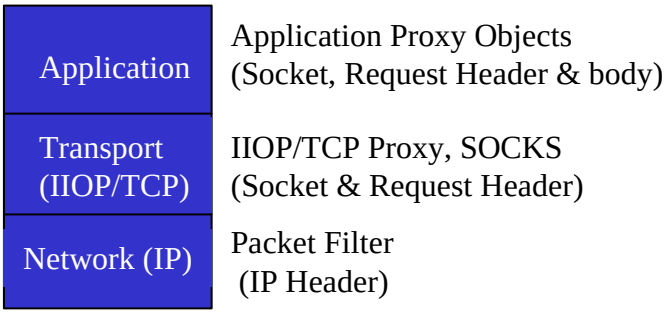

Figure 4-1: 3-Levels of Different Firewalls

Figure 4-1 shows the operation of three different types of firewall. Each type operates on the basis of information available at a specific level, application level, transport level or network level. At the transport and network levels, IIOP can be handled like any other TCP/IP-based application protocol. This means that well-known firewall techniques like packet filters and transport-level proxies can be used.

On transmission, the message body at the application level is encoded in CDR (Common Data Representation). CDR then translates IDL (Interface Definition Language) data types into a byte-ordering independent octet string. For such an octet string to be decoded back to IDL data type, the interface definition of the object is needed. This information, however, is not as a rule at the firewall with the result that the firewall is unable to decode the message body. This means that an IIOP proxy cannot base filtering decisions on the request body [13].

Since the mechanisms involved in interaction with a firewall will vary with the type of firewall, it is necessary to have a precise definition of which types of firewall are supported in CORBA if ORB interoperability is to be achieved. In this connection, the current joint revised firewall submission proposes three types of firewall (see below) for use in different situations [9, 13], a TCP firewall for simple and static applications, a SOCKSv5 [18] proxy for client-side firewalls, and a GIOP application level proxy for enforcing fine-grained security policies (especially on the server-side).

4.1 TCP Firewalls

A TCP Firewall is a simple firewall that operates at the transport level, basing its access control decisions solely on information in TCP headers and IP addresses. When a connection request is received on a given port of the firewall, the firewall establishes a connection to a particular host and port. In the course of this process the firewall uses the combination of host address and port (<host, port>) to authenticate the client on the basis of the IP address alone. Having established the connection, the firewall will allow GIOP messages to pass through uninterrupted. In other words ORB protocols are of no significance to a TCP firewall.

A simple form of ORB interoperability through TCP firewalls can be achieved without any modifications to CORBA. TCP firewalls must be statically configured with host/port address information in order to process access requests. The server can then be configured to replace its own host/port address with that of the firewall in its IOR for use outside the enclave. How this is implemented varies with the situation.

One method is to proxify the IOR manually. The client thus receives an IOR containing the address of the firewall rather than that of the server, and sends GIOP messages to the firewall (which forwards them to the server) under the impression that that is where the server is.

Due to the tradition of TCP/IP using one port per service, it is common practice to identify a TCP service by the port number used for the server. As a result, most of today's firewalls make low-level access control decisions on the basis of port used. Since there is no well-known "IIOP port", this practice does not facilitate ORB interoperability through TCP firewalls. As part of its proposed solutions, the OMG has defined a recommended “well-known IIOP port” and a “well-known IIOP/SSL port”. This will enable client enclaves with TCP firewalls to permit access to IIOP servers by enabling access to this port through their firewalls.

The OMG's firewall RFP points out that, while these ports are not mandatory since IIOP servers can be set up to offer service through other ports, the ports serve as a basic guideline for server or TCP, SOCKS or GIOP proxy deployment, and make it possible for client enclaves to identify or filter immediately the traffic as IIOP without processing.

4.2 SOCKS Firewalls

The SOCKS [18] protocol is an open Internet standard (IETF RFC1928) which performs network proxying at the transport layer, mainly on the client-side. SOCKS creates a proxy which is transparent to either party, and which serves as a data channel between clients and servers based on TCP or UDP. In this case a client can have control over which server it wishes to connect to, in contrast to the situation when normal static mapping of TCP connections by a TCP proxy is used. A SOCKS firewall could then reasonably be referred to as a “dynamic TCP proxy”.

SOCKS consists of a SOCKS proxy server on the firewall and a client-side library. In the client program the normal network calls of the socket-interface have to be replaced by the corresponding calls of this SOCKS library, and the process is called 'socksification'. The server is unchanged. The socksified client calls the corresponding functions of the SOCKS library. These SOCKS functions communicate transparently with the client and server over the SOCKS proxy server on the firewall.

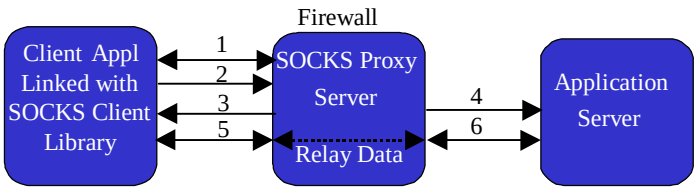

Figure 4-2: SOCKS Proxy Firewall Traversal Scenario

Figure 4-2 shows a typical scenario of a client communicating with an application server through a SOCKS firewall. There are six phases to the communication process: In the first the client authenticates itself to the SOCKS proxy server, and if

successful, the process proceeds to phase two, in which the client requests a connection to an application server. In the third phase, the SOCKS proxy grants the client's request and, in the fourth, creates a connection to the application server. In phases five and six client and server exchange data transparently over the SOCKS proxy. In practice the SOCKS proxy is relaying data between them. From the server's point of view, the SOCKS proxy server is the client, and from the client's point of view, it is the server. SOCKS also supports authenticated traversal of multiple SOCKS proxy servers.

SOCKS supports strong authentication of clients using GSS-API compliant security mechanisms such as User/Password, Kerberos, SESAME [17], or SSL [19]. The client and SOCKS server can enter into an authentication-method-specific sub-negotiation. If SSL is deployed, the client's certificates can be passed through the connections to allow the SOCKS server and the application server to authenticate the client directly. A SOCKS proxy can base transparent access control on both IP address information and user information stored in its server's and the client's configuration files.

From the perspective of SOCKS, IIOP is simply an example of a TCP-based application protocol. Thus, SOCKS is already capable of serving as a proxy mechanism for IIOP, enabling IIOP traffic to traverse firewalls. So, to handle the simple case of a CORBA client invoking an operation on a CORBA object across a firewall (a special case of Figure 4-2), the only requirements are that the CORBA client must be linked with a SOCKSified TCP library (that provides an identical API for sending TCP/UDP traffic and reimplements these functions to interact with a SOCKS firewall), and that the firewall must support SOCKS (which most existing firewalls do). An additional change is that the client host must be configured to route SOCKS requests to the appropriate proxy server. This is controlled by the client-side configuration file [9].

The information on the security provided by SOCKS firewalls, gleaned from recent research and experiment [15], is as follows: "As a server side firewall, it doesn't protect the server ORB from malicious traffic in the TCP octet stream, and doesn't allow a fine-grained enforcement of security. These can both be added to the SOCKS server, but today's SOCKS doesn't support it. On the client side an outbound firewall normally doesn't need this feature, so this would be a reasonable application of the SOCKS firewall."

4.3 GIOP Proxy Firewalls

The two firewall techniques described above work on the transport level with no knowledge of the content of the TCP connection between the client and server (with the reservation that SOCKS can be extended to act as an application level firewall). Neither of them is able to check whether the content of the TCP stream is valid IIOP. Hence, neither of them provides any real defence against the malicious client or allows fine-grained enforcement of security policies.

A GIOP Proxy is an application level firewall that understands GIOP messages and the specific transport level Inter-ORB Protocol supported (i.e. a TCP GIOP Proxy understands IIOP messages). A GIOP Proxy Object is a fully-fledged CORBA Object which provides operations for firewall navigation. Note that this CORBA

Object does not require a full ORB to be implemented in the firewall, as long as it behaves in a way that is consistent with the semantics of a CORBA Object, and understands the GIOP protocol and a transport mapping (such as IIOP) [9].

A GIOP Proxy firewall relays GIOP messages between clients and server objects. A GIOP message consists of a GIOP header, a message header and a message body. The message header, which is important for the GIOP proxy, contains the operation to be called, the object key to identify the target object, and the requesting client. The GIOP proxy makes access control decisions or fine-grained filtering decisions based on the information in the message header. For example, it could block requests to an object with a particular object key, or it could block requests for a particular operation on a specific object.

To establish a connection to a server, a client establishes a connection to the GIOP proxy.

If the GIOP Proxy is an outbound one, the ORB should be configured manually with the IOR of the proxy object. If the GIOP Proxy is an inbound one, the IOR provided by server to the client should contain the IOR of the proxy object on the firewall. The server places this information in a tagged component, which it then includes in the IOR it provides.

The GIOP proxy first authenticates the client, and then, if successful, connects to the server. Now the client sends a GIOP message to the proxy. The proxy examines this message to see whether it conforms to the security policy, and, if it does, sends it on to the server object. The proxy can, in addition, log the request and the communication if this is desired.

4.3.1 Connection Styles

There are two styles of connection through a GIOP Proxy: normal and passthrough.

- A **Normal** connection is one in which the client connects to the firewall, which in turn connects to the server. The client perceives the firewall as the server, and the server perceives the firewall as the client, neither being aware that it is connected to a mediator. It is the firewall's job to ensure that both the connections are correctly maintained, and to raise the right exception and inform the client in the event of a request being blocked or denied.

  A GIOP proxy in this mode can examine the GIOP message and do fine-grained filtering as mentioned above. This gives rise to two security issues. Firstly, the client may not trust a GIOP proxy, and hence would not want the proxy to examine the traffic. Secondly, the client and server may be using a particular authentication and/or encryption mechanism that is unknown to the proxy. Both of these problems can be solved by the concept of a passthrough connection.

- A **Passthrough** connection is one in which the GIOP proxy does not terminate the connection at the GIOP level. The difference between a TCP proxy and a GIOP proxy operating in this mode is that the latter provides security enforcement at the CORBA object level rather than at the transport level. Like the former, it simply forwards GIOP messages without processing or examining them, or raising exceptions, once the connection has been established.

The passthrough mode is mainly used for encrypted communication.

An OMG compliant GIOP proxy has to support both normal and passthrough connections, but may reject the latter if the security policy dictates so.

4.3.2 Callbacks

The OMG firewall RFP also proposes two solutions for handling callbacks over a GIOP firewall: Bi-directional GIOP and GIOP Proxy object.

The proposed **Bi-directional GIOP** solution involves allowing a server to reuse a client's connection to send GIOP request messages. This is a very simple approach because if the client can contact the server, callbacks are possible without further measures on the client side and only works if the server object and the object making the callback to the client are on the same host.

The second proposed solution, more generic and secure than the first, involves the use of a **GIOP Proxy object** at the client side too, making a callback similar to a normal request (but in the opposite direction) from the server to the client, allowing security enforcement at the client side.

4.3.3 IIOP/SSL

The standard protocol in use today for the encryption of requests sent over the Internet is SSL. SSL supports strong authentication based on asymmetric cryptography and standard X.509 certificates, while symmetric encryption is used on the IIOP message itself. ORB interoperability is frequently based on IIOP/SSL. Therefore GIOP Proxy firewalls that forward IIOP requests must support the use of SSL as a transport mechanism for secure invocations, and the proxy administrator must have an interface to the proxy in order to be able to specify different levels of access control for different users, client objects and target objects. The proxy should include client and server side authentication for proxified connections, access to client and server X.509 certificates, and access control to proxies.

For proxy firewalls to support the use of SSL a number of requirements must be met. The first is that the certificate of the client must be accessible at each link in the proxy chain, and at the server. The second is that each (inbound) proxy in the chain must be able to impose its own access policy on the traffic passing through it. The third is that Certificate Authorities (CAs) must be known to both parties and trusted by them. In addition, either the certificate policies at both sides must be compatible, or it must be possible for the parties to negotiate a common certificate policy acceptable to both.

Proxies that support the use of SSL fall into two categories: trusted and untrusted.

An untrusted proxy can forward a message from a client by pass-through connection. This means that the proxy has no access to the encrypted message. While this ensures the integrity of the communication between client and server (which is necessary when one or both of the objects are sceptical of the proxy), it gives the proxy little scope for access control, since it is unable to apply its access control list fully.

A trusted proxy, on the other hand, can, in addition to forwarding messages using this same pass-through mechanism, also forward messages by establishing a separate connection to the server. This enables a trusted proxy to apply full access control, which means that when a trusted proxy is used, access control can be applied at the server, or at the proxy on a per operation basis.

5 CORBA Firewall Traversal

5.1 Firewall Tag Components

In a CORBA-based system, client objects connect to server objects by using an IOR. An IOR contains the address of the target object, such as a host/port pair. For an object to be able to pass through firewalls, the IOR must contain access information for these inbound firewalls. In a situation where there are multiple enclaves, i.e. cascaded firewalls, it may be necessary for the IOR to contain access information for all the firewalls to be traversed, although, according to the OMG's firewall RFP, it is strictly speaking only necessary for the IOR to contain access information for the first inbound firewall to be encountered by the object, i.e. the outermost inbound firewall.

The **TAG_FIREWALL_TRANS** component, contained in an IOR, designates a single point of entry into the network of the target object, and may appear several times, once, or not at all in an IOR. The presence of multiple firewall components in an IOR indicates that there are multiple points of entry into the target's network, through any one of which the client can reach the target object. The **TAG_FIREWALL_TRANS** component is encoded as an encapsulated sequence of **FirewallMechanism** structures. This sequence is important since it describes the chain of known inbound firewalls to be traversed, and therefore dictates the order in which they must be traversed. Each firewall mechanism in the sequence contains a **FirewallProfileId** and a sequence of firewall profile data. The latter contains information about the type of firewall supported. The OMG’s firewall RFP currently defines three types of firewall, TCP, SOCKS and GIOP proxy.

5.2 Firewall Traversal Algorithm

CORBA Firewall Traversal enables CORBA objects to communicate with each other across different security domains and different combinations of different types of firewall, and hence achieves ORB interoperability through firewalls. Since each type of firewall has its own specific mechanisms for allowing connections through it, it is necessary to be acquainted with these mechanisms, and to know how firewalls of different types work in combination. The rules necessary for the traversal of any combination of the above mentioned types of firewall are laid down in the OMG’s firewall RFP [9].

A client object will determine whether it needs to traverse a firewall in order to call a target object, and will do this by examining the IOR it is assumed to possess. If the client object is in the same domain as the target object it wishes to call, it can make a direct invocation. If the two objects are not in the same domain, this is not possible. If the client object has in its configuration information about an outbound firewall to be traversed, it will send the request to that firewall. In the absence of such

information it chooses the first **FirewallMechanism** in the **TAG_FIREWALL_TRANS** field of any firewall component in the IOR.

Having determined which is the first firewall to be traversed, the behaviour the client object exhibits will be dependent upon the type of firewall involved. For further information on the traversal algorithm for each of the three OMG compliant types of firewall, see [9].

5.3 HTTP Tunnelling

HTTP tunnelling is a mechanism for traversing client-side firewalls [4] by encapsulating IIOP packets in HTTP. In the Web environment firewalls are normally configured to pass HTTP traffic, and to block the passage of messages using other protocols, including IIOP. One way of allowing an IIOP message to pass through a firewall not configured to pass IIOP traffic, is to encapsulate, or tunnel, the IIOP message in HTTP. This makes possible communication between CORBA objects through the firewall without any reconfiguration of the firewall.

The client ORB encapsulates the IIOP request in HTTP (encodes it into HTTP), which allows it to pass through the HTTP proxy at the client side firewall. At the server side there is an HTTP-to-IIOP-gateway which decodes the request from HTTP to IIOP, and forwards it to the target object. When the target object replies, the gateway encodes it into HTTP and sends it back to the client [13].

There are, of course, additional processing overheads associated with this technique due to the encoding of the IIOP message into HTTP, and the decoding of the HTTP message into IIOP. An additional disadvantage is that it does not support callbacks.

6 CORBA Firewall Application

One of the benefits of CORBA is the ease with which it is able to integrate with legacy systems through object wrappers which define object-oriented interfaces for legacy applications to enable them to interoperate with other objects in a distributed object computing environment. This means that a CORBA firewall can enable legacy systems behind the firewall of an enterprise to interact safely with application objects running on systems outside the firewall. For example, users on the Web can access services of objects that are part of the internal system of an enterprise, and an external third party can access objects for the purpose of remotely monitoring and controlling their activity on behalf of the enterprise.

One of the most important characteristics of CORBA is its wide availability and its provision of an extensive and mature infrastructure, which enables it to play the crucial role it does in the integration of distributed systems. CORBA firewalls can therefore fulfil the function of enforcing different security policies at the boundaries between integrated domains.

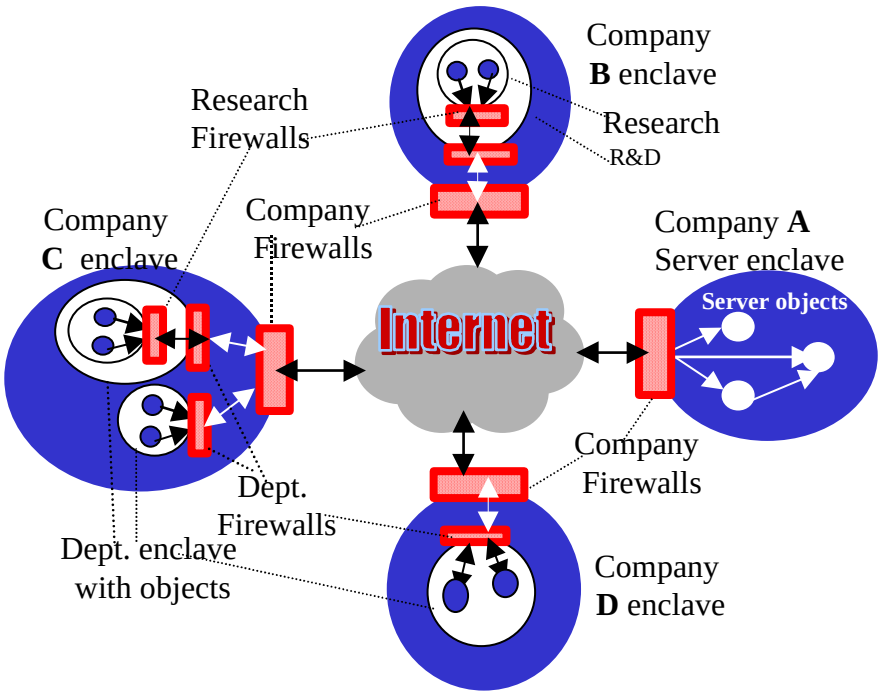

Figure 6-1: CORBA Firewall Applications over the Internet

Figure 6-1 shows a situation in which four different organisations engaging in joint research activities communicate over the Internet using CORBA firewalls. Companies B, C and D use cascaded firewall access, which enables them to enforce different access policies for different departments, while company A uses single firewall access for its server objects, allowing it to enforce access policy at only one single point of entry for the entire company. This means that the research department, for example, of company B can be permitted to access the domain of the research department of company C, while other departments of the same company C cannot. Thus a relationship of limited trust can be established between partner companies. Those organisations using CORBA-based solutions in their information systems will benefit from the use of CORBA firewalls. These include, to mention but a few, healthcare, telecommunications, financial, manufacturing and government entities.

One may reasonably wonder why it should be necessary to have such defence-in-depth consisting of so many cascading internal firewalls when the company's network is already adequately protected by the outermost inbound firewall. The explanation is quite simple. In spite of the traditional assumption that all those behind the firewall are friendly, and all those outside it are at least potentially hostile, more than half of all breaches of security are perpetrated by legitimate users behind the firewall, according to recent risk surveys [21, 20, 12, 2].

7 Emerging CORBA Firewall Implementations

CORBA Firewall Security compliant products are emerging on the market and this section gives an overview of these. This discussion is essential to an understanding of current developments in CORBA Firewall Security trends.

7.1 WonderWall of IONA

WonderWall is an IIOP proxy with a bastion host as its basis, whose job is to decide which objects are to be permitted to communicate with which objects across which security domains [4].

It filters all messages arriving on the server's well-known port on the basis of request message header, and provides fine-grained control security. WonderWall supports the exchange of encrypted IIOP messages using Orbix transformer mechanism, and has a facility for logging messages, which allows the tracing of the history of suspicious message exchanges and provides a useful debugging and monitoring facility. A general feature of WonderWall's security practice is that all messages are blocked unless specifically allowed. It uses proxies, reinforced with special security features, to foil sophisticated attacks on the network. In addition WonderWall supports HTTP tunnelling of IIOP.

7.2 Visibroker Gatekeeper of Inprise

Traditionally, to safeguard network security, Java applets have not been permitted to communicate with objects other than those in their own enclave of origin. Gatekeeper is a gateway through which Java applets can communicate across Intranets or the Internet with CORBA server objects outside their own enclave of origin without compromising network security.

Gatekeeper uses an IIOP proxy server, and sends all traffic through a single port [3]. It supports SSL, which it uses to secure the Internet communication between a client object or a Java applet, and the firewall. Gatekeeper supports callbacks, HTTP tunnelling and GUI-based configuration, and provides location transparency.

7.3 Custom Firewall Solutions

These may take the form of TCP proxy firewalls; for example SOCKS or TIS Gauntlet's generic plug proxy. They have the advantage of using IIOP/SSL and providing application transparency, and the disadvantage of lacking application level filtering, having a low level of security and requiring firewall configuration [13].

TIS Plug-gw as a CORBA Firewall

The plug-gw of the TIS Firewall Toolkit can be used as a very simple transport level IIOP firewall proxy. It needs proxified IORs, which can be hard if one doesn't have the ORB's source code and the environment is dynamic. It does not support secure callbacks [14].

SOCKS as a CORBA Firewall

As pointed out earlier, SOCKSv5 [18] is one of the current OMG Firewall Joint Submission's suggested firewall mechanisms. In recent experiments SOCKS has been successfully socksified [15].

7.4 ObjectWall of TechnoSec

ObjectWall is a tool kit to be used in combination with other firewall tools, such as SOCKS, to construct secure firewalls for CORBA based applications. It supports transparent proxies at inbound and outbound sides, callbacks, central policy management (i. e. it only grants access to an object if the security policy defined by the administrator allows it) and firewalls with several levels of defence, and provides proxies and packet filters. It also supports the dynamic launching of proxies at runtime.

ObjectWall consists of two parts, Enforcement Modules and Policy Manager. The former are standard firewall tools, packet filter, NAT, and TCP level proxies with CORBA interface. The latter, the Policy Manager, has the task of checking whether the requests are in accordance with security policy [16].

7.5 ORB Gateway of NAI

The ORB Gateway functions like a firewall proxy in that it controls the access of CORBA operations to an enclave, but does so with a higher degree of granularity in its control over the CORBA access policy than is typical of a proxy. The access policy is expressed in DTEL++, like OO-DTE, and each request is categorised according to this policy. The domains to which an object is granted access are determined by the category in which the object has been placed on the basis of the degree of trustworthiness of the authentication mechanism. An unauthenticated object may be granted access to a limited domain with few privileges, while strongly authenticated objects will be allowed much freer access.

The ORB Gateway currently supports SSL as an authentication mechanism, but in the future, according to NAI, DCE, IPSec and Kerberos will be used.

The ORB Gateway is a single point of external access to the object services of an enclave which vets access requests on the basis of the nature of the request and of the attributes of the object from which the request comes, and connotes control of traffic between ORBs of multiple enclaves rather than gateway or proxy control of traffic with the outside world [6].

7.6 OO-DTE of NAI

Object-Oriented Domain and Type Enforcement (OO-DTE) [7] provides scalable, role based access control for CORBA systems, and is an object oriented extension of DTE, under which each resource is assigned a type and each process runs under a domain. What types of resource the processes of a given domain are permitted to read and write is specified by the DTE policy. Analogously in OO-DTE a CORBA operation is assigned a type, and the client object and server object (called processes by NAI) each runs in its own domain. The client object can invoke an operation if it is permitted to invoke an operation of that type, and similarly, the server object can implement that operation if it is permitted to implement an operation of that type.

As mentioned above, the language for expressing OO-DTE policy is called DTEL++.

There are two versions of OO-DTE already implemented, a DTE-kernel based system that provides non-bypassable access for CORBA systems, and an above-kernel OO-DTE system that performs access control in an Orbix filter without the non-bypassability feature. The two versions are, however, interoperable and use the same access control policy.

NAI state that they are currently working on the addition of SSL to above-kernel OO-DTE and the improvement of security policy administration, and that the policy distribution and the synchronisation tools presently under development will allow a centrally administered DTEL++ policy to be automatically distributed to CORBA systems within the enclave.

7.7 Gauntlet 5.0 of NAI

Gauntlet 5.0 for UNIX (Solaris & HP-UX) from NAI [8] is an IIOP proxy for Gauntlet. This proxy forwards messages sent between CORBA objects across Gauntlet firewalls, only after ascertaining that they meet CORBA's IIOP standard, thus ensuring that only valid IIOP messages travel across the firewall. This protects the network by ensuring that IIOP-designated ports are used solely for IIOP traffic, and by reducing the exposure of CORBA objects to invalid messages that could disable them.

Gauntlet 5.0 is compatible with IIOP 1.1. It accepts only well formed IIOP messages, which it passes unchanged to both clients and servers in accordance with the policy expressed in its administrative tools and netperm table, and, in the event of communication failure or a message being rejected, it generates an appropriate error message. Gauntlet operates transparently for both inbound and outbound connections and supports callback requests from servers to clients using bi-directional IIOP, and the "NORMAL" mode of IIOP proxy connection, but the current version does not support the "PASSTHRU" mode.

According to NAI information, Gauntlet has been tested for correct interoperation with client and server applications based on the ORBs Orbix v2.3, OrbixWeb, and VisiBroker v3.3 (C++ and Java).

7.8 MPOG of NAI

The Multi-Protocol Object Gateway (MPOG) [5] is an application proxy server that has been installed on Network Associates' Gauntlet firewall. It provides access control for operations on distributed objects, and its architecture allows reuse of policy data base and access control techniques for multiple distributed object technologies such as DCOM, CORBA and Java RMI. It has facilities for message routing based on the CORBA, Java RMI or DCOM interface requested.

For handling security protocols, MPOG currently has two protocol handlers, a non-secure handler and an SSL handler. The non-secure handler uses an unencrypted communication channel which can be used in an environment where encryption of messaging and authentication of users may not be necessary. The SSL handler supports SSL for authentication between client and MPOG, and between MPOG and server, and for negotiating the supported cryptographic parameters between client and server, in situations where security is an important consideration.

As stated in [5], the MPOG access control mechanism separates the access control decision from the distributed object message handling, and supports OO-DTE domain derivation, trust management, and per-object and role-based access control.

8 Conclusions

Because of CORBA's popularity as a distributed object technology supporting cross-language and cross-platform interoperability and integration of enterprise-wide distributed applications, it is being increasingly used for the development of applications over the Internet, Intranet and Extranet in specialised market areas, such as healthcare, telecommunications, manufacturing and financial services.

The OMG recognises the need to safeguard the security of CORBA objects, and the fact that firewalls are still a powerful protective mechanism that will continue to play an important and central role in the maintenance of Internet security for some years yet. As a result they have specified CORBA Firewall Security with a view to bringing firewall technology to the world of distributed object computing technology, enabling firewalls to identify and control the flow of IIOP traffic, and thus enabling CORBA application objects behind a firewall to interact safely with objects outside the firewall. CORBA firewalls can also make fine-grained access decisions based on the attributes and operations of objects.

Firewalls continue to change and develop, and new features are regularly added as the need arises. If this development follows the present trend, CORBA firewalls will continue to combine configurable access control and authentication mechanisms with their already existing functions, thus providing more powerful and flexible protection mechanisms for the Internet, Intranet and Extranet. In the future it is probable that they may also handle moving agents, and may perhaps be able to provide migration transparency.

OMG CORBA firewall compliant products are emerging on the market, and will continue to do so.

References

1. H. Abie, An Overview of Firewall Technologies, Tlektronikk Volume 96 No. 3-2000, pp. 47-52, January 2000
2. G. Dalton, Acceptable Risks, a survey by PricewaterhouseCoopers and InformationWeek, August 31, 1998, http://www.informationweek.com/698/98iursk.htm
3. Inprise's VisiBroker Gatekeeper, http://www.borland.com/visibroker/gatekeeper/solsheet/
4. IONA Technologies PLC, WonderWall Administrator's Guide, December 1997, http://www.iona.com/
5. G. Lamperillo, Architecture of Distributed Object Firewall Proxy, NAI Labs, December 30, 1999.
6. NAI, Network Associates Labs, an ORB Gateway, http://www.nai.com/
7. NAI, Network Associates Labs, Object-Oriented Domain Type Enforcement (OO-DTE), http://www.nai.com/nai_labs/asp_set/applied/arse_corba.asp

8. NAI, Network Associates Labs, Gauntlet 5.0 for Unix, an IIOP proxy for Gauntlet, http://www.nai.com/asp_set/products/tns/gauntletunix_intro.asp
9. OMG, Joint Revised Submission, CORBA/Firewall Security+Errata, OMG Document, ftp://ftp.omg.org/pub/docs/orbos/98-07-03.pdf, July 6, 1998
10. OMG, The CORBA Security Service Specification (Revision 1.2), ftp://ftp.omg.org/pub/docs/ptc/98-01-02.pdf, January 1998
11. OMG, The Object Management Group, http://www.omg.org/
12. J. L. Phipps, Hackers: Can You Stop Them?, http://www.mediainfo.com:81/ephome/news/newshtm/minfocom/1198a.htm
13. R. Schreiner, CORBA Firewall White Papers, TechnoSec Ltd., 1998 http://www.technosec.com/whitepapers/corba/fw/main.html
14. R. Schreiner, TIS plug-gw as a CORBA firewall White Papers, http://www.technosec.com/whitepapers/corba/tis_plug_gw/cfwexp1.html
15. R. Schreiner, SOCKS as a CORBA firewall White Papers, http://www.technosec.com/whitepapers/corba/socks/socks_exp.html
16. R. Schreiner, ObjectWall: Integrated Protection of Dynamic CORBA Applications, http://www.technosec.com/products/Objectwall.htm
17. SESAME, A Secure European System for Applications in a Multi-vendor Environment, http://www.esat.kuleuven.ac.be/cosic/sesame/
18. SOCKS V5, http://www.socks.nec.com/
19. SSL3.0 Spec: http://home.netscape.com/eng/ssl3/3-SPEC.HTM#2
20. M. J. Thompson, Corporate Network Security, September 21, 1998, http://www.thestandard.com.au/metrics/display/0,1283,750,00.html
21. Warroom Study: 1996, Security in Cyberspace Hearings before the Permanent Subcommittee on Investigations of the Committee on Governmental Affairs, United States Senate, 104th Congress, 2nd Session. ISBN 0-16-053913-7, http://www.warroomresearch.com/researchcollabor/infosecuritysurvey.htm