Protecting the Hosted Application Server

Paola Dotti, Owen Rees
Extended Enterprise Laboratory
HP Laboratories Bristol
HPL-1999-54
April, 1999

E-mail: {Paola_Dotti,Owen_Rees}@hpl.hp.com

application server, CORBA, security, firewall, outsourcing

Internet applications, are evolving from the web server to the more powerful and dynamic application server in order to support the deployment of complex applications integrated with the organization's back end systems.

A key element of the application server architecture is CORBA, the Common Object Request Broker that allows applications to communicate in a transparent and interoperable manner. For this new architecture to succeed, it must guarantee a secure processing environment to the organisations.

This paper explains why a conventional firewall can not be used to secure CORBA applications. It describes the architecture of CORBAgate, a gateway based on the concept of proxies and object key re-mapping. The paper compares the CORBAgate solution with the OMG specification for CORBA firewall security. Finally, the paper discusses how these elements combine to enable outsourced application services.

Internal Accession Date Only

Protecting the Hosted Application Server

Paola Dotti, Owen Rees
Hewlett-Packard Laboratories,
Filton Road, Stoke Gifford,
Bristol, BS34 8QZ,
United Kingdom.
Paola_Dotti@hpl.hp.com, Owen_Rees@hpl.hp.com

Abstract

Internet applications, are evolving from the web server to the more powerful and dynamic application server in order to support the deployment of complex applications integrated with the organization's back end systems.
A key element of the application server architecture is CORBA, the Common Object Request Broker that allows applications to communicate in a transparent and interoperable manner. For this new architecture to succeed, it must guarantee a secure processing environment to the organisations.
This paper explains why a conventional firewall can not be used to secure CORBA applications. It describes the architecture of CORBAgate, a gateway based on the concept of proxies and object key re-mapping. The paper compares the CORBAgate solution with the OMG specification for CORBA firewall security. Finally, the paper discusses how these elements combine to enable outsourced application services.

Introduction

Application server has become one of the hottest new Internet product categories. Despite the numerous definitions currently in circulation, the application server can be seen as a concrete sign of the technological evolution of the Web Server, core of today's internet applications, towards a more intelligent, complex and dynamic entity. Generally, the application server is a set of middleware and application level components which allow application processing to be moved out of the back end system into a middle tier component. Figure 1 shows the transition from the old Web Sever Model to the new Application Server Model.

There are currently various application server solutions provided by different vendors. Services provided by the application server usually include support for components to allow rapid deployment of applications, support for a middleware communication for a more scalable integration among applications and the back end systems, and support for complex transactions to give the required predictability. Most vendors offer Enterprise Java Beans as the component model, and CORBA[1] as the middleware communication infrastructure.

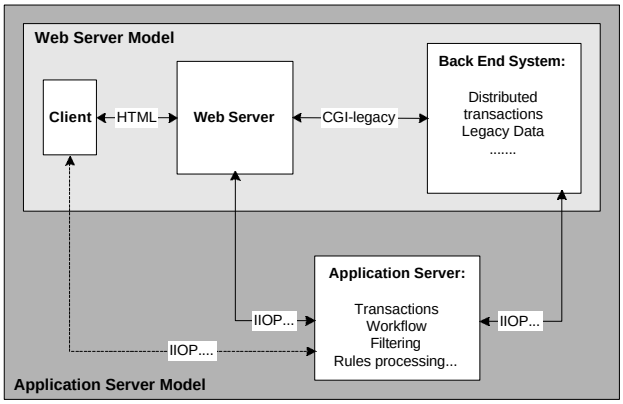

Figure 1. Web Server versus Application Server Model

This new architectural model will not be successful if it does not guarantee a secure processing environment for the applications. Security is a vital element for the success of any Internet application, and its value grows with the level of interaction an organisation allows between the clients and its back end systems. Application server is about breaking the barriers between clients and servers. It is about replacing the Web Server model – a basic GUI window on the back end data – with a multi-tier model by opening up the back end system and moving some intelligence out of it into a middle tier.

With appropriate security, the middle tier can be hosted separately from the client and from the back-end

data. This opens up the opportunity for outsourcing application services.

Where the different elements are hosted by different organizations, the interactions must pass through the firewalls that protect each organization's network. Conventional firewall mechanisms are not well matched to the needs of CORBA, as used by many application servers. CORBAgate enables the CORBA interactions without compromising security.

CORBA and Firewalls

CORBA – Common Object Request Broker – is an infrastructure that allows applications, written in different languages, to communicate and collaborate with each other without knowing where they run. An application wanting to use the services provided by another application, needs only to know its code name. This unique code name is called Interoperable Object Reference (IOR).

The beauty of CORBA is that the programmer has to know little about how the communication actually happens and who is listening on the other side. Server applications can be launched on demand and an application can be simultaneously act as a client or a server. References to objects can be passed around among objects allowing a very dynamic global interaction. The dynamism and the transparency of CORBA make it difficult to control access to CORBA applications through a firewall. To understand the issues it is necessary to analyze how IORs pass between objects.

An object that has a reference to a second object can pass this reference to a third object. The CORBA infrastructure must arrange that the third object can then use the reference it receives to invoke the second object.

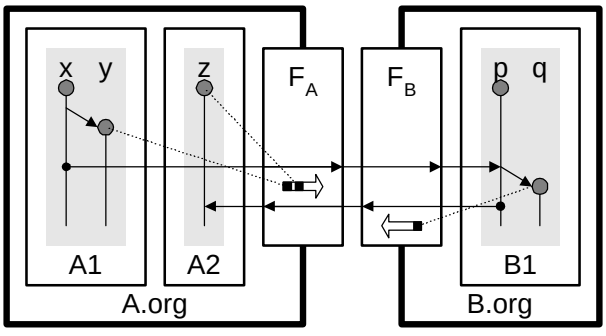

Figure 2. Object reference passing

Figure 2 illustrates the passing of references to both new and existing objects. In this example, an object *x* in a process on host A1 has a reference to an existing object *z* which is in a process on host A2, and an existing object *p* which is in a process on host B1. Object *x* creates a new object *y*, and then invokes *p* passing references to both *y* and *z*. The object *p* then creates a new object *q* and uses the reference it just received to invoke *z* passing a reference to *q*. The responses (which are not shown) may also carry object references as the result, as out parameters, or in an exception.

The example shows the objects as belonging to different organizations, A and B, both of which have firewalls protecting their networks. CORBA through firewalls presents two main unresolved issues. First of all, the passing of object references in CORBA means that the set of hosts and ports that must be accessible through the firewall changes dynamically, and this does not match current firewall configuration mechanisms.

A further issue is that an ORB will typically support many objects through the same port on a host. If the mechanism at the firewall grants access at the granularity of a port, then if one object on an ORB is made visible then so are some, if not all, other objects on that ORB.

The CORBAgate object gateway

Although CORBA presents a problem for conventional firewall mechanisms, it provides a solution by enabling the creation of an application level object gateway that can be inserted transparently between application objects. An application object cannot tell that it has been given a reference to a proxy object in the gateway, rather than a reference to the target object. The target object cannot tell that it has been invoked by the proxy, rather than the original client object. This transparency also makes it possible for invocations to cross multiple gateways without any additional mechanisms. The key to this transparent gateway insertion is that the gateway replaces references in invocation parameters and results as they pass through. Once an initial reference has been replaced by a reference to a proxy, this mechanism ensures that exactly the necessary set of proxies is created.

A gateway based on proxies allows access control to be enforced at the level of individual objects. A proxy is for a specific target object and does not grant access to other objects that happen to be hosted by the same ORB instance.

A standard proxy can be replaced by an application specific proxy to provide special controls. This makes the proxy solution modular.

A CORBA interoperable object reference (IOR) for an object that supports the Internet Inter-ORB Protocol (IIOP) includes a host and port to which the client object will connect, and an object key which identifies the

particular object. IORs will be written here as IOR[h:p:k] to identify these significant elements.

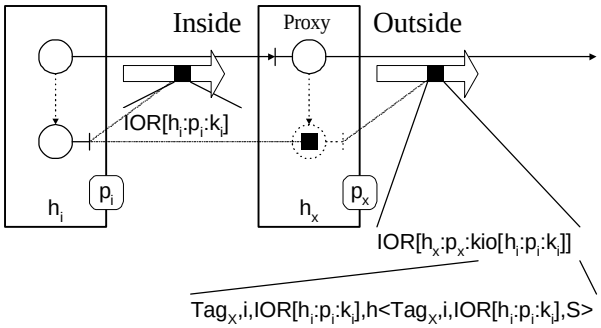

Figure 3. Reference embedding

Figure illustrates the reference mapping that occurs when references are replaced. The client object invokes the proxy object, passing an IOR. The proxy object replaces the IOR in the request with an IOR that refers to another proxy object in the gateway. The substitute IOR contains the host and port of the gateway, and an object key. That object key includes the original IOR, a flag to indicate that the proxy will be for incoming invocations, a tag that is used for access control, and a secure hash of these elements and a secret known only to the gateway.

The information included in the object key is sufficient to create the proxy. Since this information will be included in an incoming invocation, the gateway need not retain any state. The activator mechanism of the ORB supporting the proxy objects is used to create new proxies when they are needed.

The tag in the object key identifies how the reference came to be created. Each proxy contains a tag, and, by default, inserts the same tag into references it creates. The tag identifies a set of proxies for access control. Before creating a proxy in response to an incoming invocation, the tag is checked against the current access policy. If the check fails, the proxy is not created, and the invocation is rejected. The tag is also used to shut down active proxies so that access can be revoked immediately if required.

If a new tag is created as the starting point of some series of interactions then all interactions initiated from that starting point can be disabled when the tag is revoked. This combines access control enforcement at the granularity of individual objects with management of access in terms of the application logic.

The secure hash with a private secret protects against forgery of the external IORs. It is easy enough to discover the host and port of the gateway, and a valid tag. The target IOR of an internal service may be discovered or guessed. The hash with a secret prevents an attacker from combining these elements into an object key that will trigger the creation of a proxy. Without this defense, every object would have to defend itself against access from external clients as if there were no firewall.

The interactions of clients and servers with the gateway depend only on facilities defined in the CORBA 2.0 specifications. This means that CORBAgate can be used with application objects implemented on existing interoperability compliant ORBs.

CORBAgate supports a modular approach to security. It allows organizations to enforce their security policies by changing the application proxies, allows fine grained access control at the object level, uses an anti forgery mechanism to prevent security attacks, complies fully with the CORBA specifications and is completely transparent to the end user and the applications.

Figure 4 shows how CORBAgate can be deployed to protect the application server from rogue clients, and also to protect the back-end system

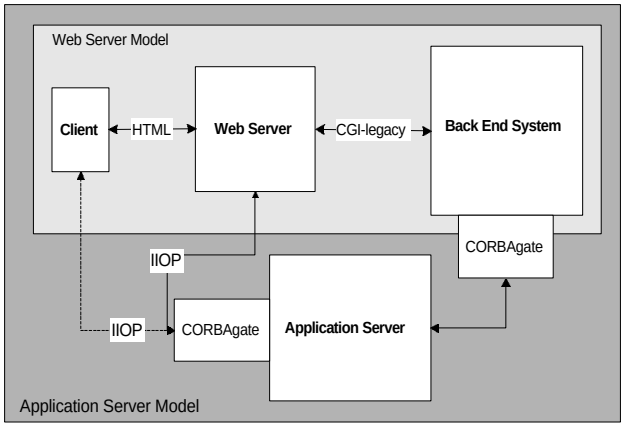

Figure 4 CORBAgate and the Application Server

The OMG specification

The Object Management Group has recently adopted a CORBA firewall security specification. The primary focus of this new specification is to define how to change CORBA so as to allow interactions across current conventional firewall components, although it also describes a new proxy firewall component.

CORBAgate was designed to achieve a different goal, providing a firewall component that can be used with existing CORBA infrastructure and applications.

The OMG specification describes changes to IORs to include additional data, as well as changes to the GIOP protocol, and the Portable Object Adapter; these are fundamental parts of the CORBA core. This requires changes to ORB implementations, and objects built on

current implementations will not be able to interact across the firewall mechanisms described in the new specification.

Unlike CORBAgate, the OMG specification is based upon having each object, or rather the ORB supporting it, create IORs containing additional components that describe all of the inbound firewalls that need to be traversed to reach the object. These references are transmitted unchanged rather than being mapped as they pass the firewall.

This strategy means that every ORB supporting server objects must be configured to know where it is relative to the firewalls; the issues of managing the configuration are not discussed in the OMG specification.

Since any object may create references to be used externally, there is no protected environment in which these references can have forgery protection added. Since the references and the object keys in them are generated away from the firewall, checking at the firewall would require digital signatures on the relevant parts of the reference. Every ORB instance would need a signing key, and would need to keep it secret.

CORBAgate does not depend upon any of the changes introduced in the new specification. The prototype has been built on an ORB that has none of the new features, and it interacts with applications built on various currently available ORBs. CORBAgate also concentrates all of the security mechanisms and administration into one place rather than spreading them over a large number of hosts that will typically be continually changing.

Hosting the Application Server

Application servers not only provide a framework for rapid deployment of new business processes, they also separate the business logic from the back end data. This makes it possible to outsource the application server and the processing of the middle tier business logic to an application service provider.

The fine-grained but unobtrusive access control provided by CORBAgate will be essential to allow the application service provider to access the back end data.

An application service provider will want to offer its hosting service to many clients, and it will be essential to keep the application components and data belonging to different clients separate. Fine-grained access control in front of the application server is required.

In a service provider environment, using separate machines for different clients becomes a burden due to the physical space required. The solution is to use fewer, but more powerful machines, but hosting different clients on the same machine requires special security precautions in order to preserve the separation. An effective solution is to use a trusted operating system of the kind originally designed to handle classified data. Such systems can be configured to enforce the required separation.

CORBAgate has been tested on a system of this kind, the HP Praesidium VirtualVault.

Conclusions

CORBAgate is an application level gateway for the IIOP protocol which allows fine grained access control to CORBA applications, provides high security through its anti-forgery mechanism, complies fully with the CORBA standard and is completely transparent to the applications. Its superior security approach make it the ideal component in the application server architecture for any organization concerned about opening up its back end systems either to an outsourced middle tier or directly to their internet clients.

References

[1] “Common Object Request Broker: Architecture and Specification”, Object Management Group, Framingham, MA, USA, February 1998.