CORBA 防火墙技术研究

A Study of CORBA Firewall

李天宁 李 冀 龚丽敏 黄 皓 谢 立

(南京大学计算机科学与技术系 南京大学软件工程中心 南京210093)

Abstract As a mainstream of middleware technology, CORBA has integral model of distributed architecture, highly abstract capability and good transparency. However, CORBA confronts the same security problem as other distributed models. The emergence of CORBA Firewall has solved the problem. In this paper we present the analysis of several models of CORBA Firewall, and discuss some key issues about it.

Keywords CORBA, Firewall, Proxy, Security

1. 概述

随着网络及分布式应用的日益普及,作为分布式应用的重要模型,由 OMG 组织制定的 CORBA(Common Object Request Broker Architecture)也逐步走向了完善。它不仅具有分布式系统健壮性好、易于维护、负载平衡等特点,还充分利用了当今最为流行的面向对象技术、软件总线技术和部件技术,使已有的应用得到了有效的保护,并实现了应用各部件的灵活配置,为构造新的应用提供了便捷的方法。较之同类型的 DCOM 和 RMI,CORBA 更具有跨平台、跨语言、结构规范、实施灵活的巨大优势。

但是同其它分布式应用模型一样,CORBA 系统也面临着诸如非法访问 CORBA 对象、恶意攻击等安全性问题,因此需要进行访问控制、身份认证、授权管理、鉴定,并保护现有资源不受侵害、数据通信不被侦听。为此 OMG 提出了 CORBA Firewall 的概念。它将 CORBA 与 Firewall 有机地结合起来,并针对 CORBA 系统本身的特点,加入了新的变化。

本文通过对 CORBA Firewall 几种模型的介绍,分析了各自的优缺点及其适用性。此后就 CORBA 与 Firewall 结合带来的几方面的变化进行了探讨。

2. CORBA 防火墙模型

CORBA 防火墙[1]是介于有组织的网络环境和它所处的外部网络(通常是 Internet 或 Extranet)之间的网关:我们利用它可以将重要的资源分离出来加以保护,并严格控制对内部网络的访问,防火墙的安全方法可以利用网络协议控制网络流量。如图1所示,Client 端与 Server 端分别处于不同的网络并被防火墙隔离,Client 端 ORB 与 Server 端 ORB 的数据交换必须通过各自所在网络的防火墙。

图1 CORBA 防火墙基本结构

如图2所示,我们将数据或请求从外部网络进入内部网络所经过的防火墙称为 InBound 防火墙,反之称为 OutBound 防火墙。因此,InBound 防火墙实现了外部网络对内部网络资源的存取控制,而 OutBound 防火墙控制了内部网络对外部资源的访问。不过,这种分类方式只是逻辑上的概念,在具体实现过程中并不严格地将两者区分开来。

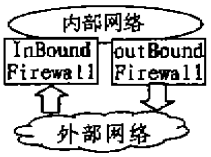


图2 内部网络防火墙的配置

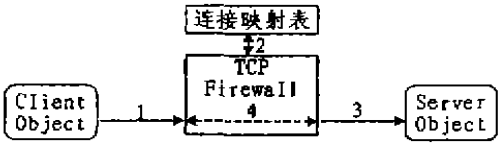
目前,GIOP 是实现 CORBA 互操作性的通讯协

李天宁 硕士研究生,主要从事分布式系统、对象技术方面的研究。李 冀 硕士研究生,研究方向为分布式系统、并行处理。龚丽敏 硕士研究生,主要从事安全操作系统方面的研究。黄 皓 博士,主要研究方向为信息安全技术、分布式处理。谢 立 教授、博士生导师,主要研究方向为分布式计算、并行处理、先进操作系统。

议，它在 Internet 上定义了向 TCP/IP 的映射，称为 I-IOP（Internet inter-ORB Protocol）。CORBA Firewall[5]中定义了三种防火墙：TCP Firewall、SOCKS 和 GIOP Proxy。前两者属于运输层防火墙，而后者属于应用层防火墙。

2.1 TCP Firewall

如图3所示，TCP Firewall 在其提供 IIOP 服务的端口接收到 Client Object 从〈clienthost，clientport〉发出的连接请求，分析 TCP 报文中源主机地址，根据连接映射表中的规定，判断是否是允许的主机。如果满足条件，TCP Firewall 就建立两条 Socket 连接：一个是〈clienthost，clientport〉与防火墙的连接，另一个由 TCP Firewall 代替 Client Object 建立一个在连接映射表中与 clienthost 相对应的〈serverhost，serverport〉连接，并在防火墙内部建立两个连接的一个映射，用于转发随后的报文。由于 TCP Firewall 不允许 Client 与 Server 直接通讯，当 Server Object 将 IOR（Interoperable Object Reference）返回时，TCP Firewall 应将 IOR 中 Server Object 的 serverhost 和 serverport 替换为与 Server Object 相联系的防火墙主机名及端口号，以保证 Client Object 能够正确地寻址。



1. Client Object 向 TCP Firewall 发送连接请求
2. TCP Firewall 根据请求报文和连接映射表确定是否允许提供服务
3. TCP Firewall 根据端口连接配置与 Server 建立相应连接
4. TCP Firewall 转发报文

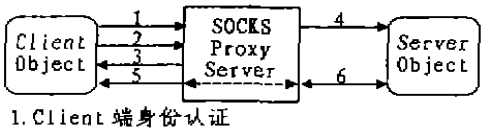
图3 TCP 防火墙工作规程

TCP Firewall 在 Client 和 Server 之间仅仅是一个简单的代理。对于 GIOP/IIOP 报文而言，防火墙是透明的。TCP Firewall 的结构简单，配置和使用都十分方便：只要在连接映射表中预先定义允许通信的主机名/IP 地址和可供调用的 Server Object 所在的主机名及端口号，实际上它就是一个简单的包过滤防火墙。TCP Firewall 将 Client 与 Server 完全隔离。如果 Client 需要调用 Server 提供的服务，只能从 TCP Firewall 的 IIOP 端口获得。因此从 Client 的角度看，TCP Firewall 的 IIOP 端口就等于 Server 提供服务的端口，从而较好地屏蔽了网络的具体结构。

但是 TCP Firewall 的安全控制粒度是主机名/IP 地址，因此很容易受到伪造 IP 地址主机的攻击。其次，即使 Client 具有合法的主机名/IP 地址，仅仅利用规则库无法限制它使用 Server 所在主机提供的其他服务。第三，TCP Firewall 只能理解 TCP 报文，而无法理解更高层次的语义，也就无法防范因垃圾报文拥塞导致的服务瘫痪攻击。第四，TCP Firewall 中所提供的连接是静态配置的，即需要预先规定哪些 Client 需要与哪些 Server 及其端口建立连接，所以它只适合于作为主动建立连接的 Client 端的防火墙，而不适合于被动建立连接的 Server 端。

2.2 SOCKS

SOCKS 由 SOCKS Proxy Server 和 Client 端的 SOCKS 运行库组成。Client 通过调用 SOCKS 库的例程与 Server 进行间接通讯。如图4所示，Client 向 Server 发出请求时，先与 SOCKS Proxy Server 进行有关身份认证的协商。此后 SOCKS Proxy Server 为 Client 创建一个 SOCKS Proxy，这个 Proxy 从请求报文中获取目的地址信息后，代替 Client 建立一条与 Server 相联系的连接，并在 Proxy 内部将这两条连接联系起来。



1. Client 端身份认证
2. Client 端连接请求
3. SOCKS 返回连接确认，并创建 SOCKS Proxy
4. SOCKS Proxy 建立与 Server Object 的连接
5, 6. Client 与 Server 进行通讯

图4 SOCKS 工作规程

SOCKS 实际上位于运输层与应用层之间，属于虚电路级的防火墙，较之高层防火墙响应速度更快。与 TCP Firewall 相比，SOCKS Proxy Server 提供了 Client 与 Server 之间的动态映射，更加灵活，IOR 中也不必进行地址映射，因为 Client 对 Server 的调用还是通过 SOCKS 完成。我们可以把它看作一个通用的 TCP Firewall，而且 SOCKS 具有更为多样化的认证机制，它包括对主机地址的检测、对用户名及其口令的检测和对调用方法的检测。此外在运输层还可以利用 SSL（Security Socket Layer）提高 SOCKS 的安全性。

SOCKS 适合部署在 Client 端，而不适用于作为 Server 端的防火墙，因为所有的连接请求均由 Client Object 发起，Server 无法预先得知与之建立连接的 Client；而且无法防止恶意的 TCP 数据流入 Server Object；当数据报过长、报文不符合 GIOP 的格式要求均可能导致 CORBA 对象出现拒绝服务的情况。

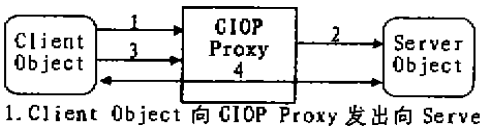
2.3 GIOP Proxy

CIOP Proxy 属于应用层防火墙，较之前两种防火墙，GIOP Proxy 的层次更高，它不仅能够转发报文，还能够理解 GIOP/IIOP 报文的语义，因此它可以为

CORBA 系统提供更高级别的保护，ORB 将对象引用、方法名、参数、结果等经过一定的分解，以请求报文的方式发送给 GIOP 代理。GIOP 代理可根据从报文头中获取的有关请求方式、对象标识等信息进行过滤；然后在每个报文前加上自己的 GIOP 报文头，交由下层通讯设施进行传送处理。

CORBA Firewall[5]中根据 GIOP Proxy 在连接过程中参与的程度不同，提出两种模式：normal 方式和 passthrough 方式。

如图5所示，Passthorugh 方式只是将 Client Object 与 Server Object 的连接在 GIOP Proxy 内部建立一个简单的映射，而没有其他额外的处理。当参与通讯的两个对象对 GIOP Proxy 不足够信任时，就可以采用这种方式。



1. Client Object 向 GIOP Proxy 发出向 Server Object 连接请求
2. Server Object 与 GIOP Proxy 建立 TCP 连接
3. GIOP Proxy 回复 Client Object 连接建立
4. Client Object 与 Server Object 通过 Proxy 进行通讯

图5 passthorugh 方式的 GIOP Proxy

如图6所示，normal 方式的 GIOP Proxy 不仅将两条连接联系起来，而且还具有向两个对象提供异常处理、参与 GIOP 对话、进行安全检查等功能。它将 Client 与 Server 完全隔离，并提供较高安全等级的服务。由于 CORBA 防火墙系统本身尽可能地考虑了安全性问题，Server Object 和 Client Object 可以完全信任 GIOP Proxy，因此 OMG 推荐使用 normal 方式。

1. Client Object 向 GIOP Proxy 发出向 Server Object 连接请求
2. GIOP Proxy 与 Client Object 建立 TCP 连接
3. Server Object 与 GIOP Proxy 建立 TCP 连接

图6 normal 方式的 GIOP Proxy

GIOP Proxy 以服务/Object 作为安全控制的基本单位，比主机、端口的粒度更小。此外利用这种防火墙代理，我们还可以对 GIOP 得到的报文进行审查、日志记录，并进行相应的语法语义检查，使之符合自己的安全策略。它有效地过滤了可能导致 ORB 系统瘫痪、CORBA 对象出现异常的垃圾报文，这正是前两种防火墙所不具备的。因此，这种防火墙更适合于在 Server 端配置。

3. CORBA 防火墙带来的变化

由于防火墙的出现，使得原有 CORBA 系统必须做出相应的变动，以符合自身的安全策略并保证通讯尽可能不受影响。这些变化主要体现在动态特性、回调机制、定址方式和数据加密等方面。

3.1 动态特性

通常见到的 HTTP、FTP deamon 总是在开机时就启动，并随时监听对应的固定端口传来的请求，但是这种方式不适用于 CORBA 防火墙，因为只有 Server ORB 的实现库(Implement Repository)在 CORBA 系统启动时在预先规定的端口启动，而其它 CORBA 对象只有在首次调用时才被激活，且它们对应端口不固定，所以防火墙不可能以静态方式确定侦听端口，更不可能侦听所有可能的端口，这样无疑会极大地加重防火墙的负担，只能采取动态的方法。此外防火墙还应对 CORBA 对象进行全生命周期的监控，以确定针对该 CORBA 对象的请求做出何种动作。TCP Firewall 的动态特性不太好，因为它只支持连接的静态配置；而 SOSKS 就具有一定的动态特征，它能够动态地为 Client/Server 创建 Proxy，并为各自的连接建立映射关系；由于 GIOP 标准对连接映射的规定不断变化，GIOP Proxy 的动态特性依赖于具体实现。良好的动态特性便于系统的扩充，可以减少人工管理的工作量，同时也提高了系统的透明性。目前 CORBA Firewall 实现动态特征的方法多是将 IOR 中的目的地址替换成 Proxy/Firewall 的地址，Client 在调用时，再由 Firewall 对调用请求进行路由。这要求对现有的 CORBA 系统进行改造，包括对 BOA/POA、GIOOP 协议等重要组成部分进行改造，这也意味着目前的系统可能无法集成到 Firewall 保护的网络环境中。我们认为这里可以采取一种折衷的解决办法：不将 IOR 中真实的目的地址替换或重定向，而是利用数据报封装的思想，将原地址写入 IOR 的某个 tag 中，同时在变更过的 IOR 中加入当前 Firewall 的路由信息。这样就保证了 Firewall 相对于 Server Object 的透明性。虽然新的 IOR 可能会比较长，但相对于巨大的改造工作量而言还是合算的。

3.2 回调机制与双向 GIOP

回调(callback)是 CORBA 中一种重要的机制，能够实现对象之间的异步通讯，提高 CORBA 系统的并行度。这种方式由 Client 将供回调的对象的引用传送给 Server；当 Server 完成操作时，就利用这个对象的引用将结果返回或通知 Client，如果 Client 与 Server 之间存在防火墙，并且拒绝两者直接通信，则 Server 在调用 Client 对象时会被 InBound/OutBound 防火墙

拒绝而导致 CallBack 失败，目前有两种解决办法，一种是在 Client 端对称配置 GIOP 代理，使用方法类似于 Server 端的 GIOP 代理。由于调用服务的对象远远多于提供服务的对象，因此这种对称配置方式工作量极大，在实际操作中很难推行。另一种方法是建立第二条 TCP 连接，并利用 TCP 代理对 Server 端进行判定：当且仅当至少存有一个 Client 所在主机到 Server 所在主机的 IIOP 连接时才允许回调。这种方法也不够好，因为 Client 所在主机与 Server 所在主机之间有 IIOP 连接只是允许建立第二条 TCP 连接的必要条件，而非充分条件。它不能保证反向连接的安全性，更无法保证不影响其它 CORBA 对象的安全。在 CORBA 2.3 中提出了双向 GIOP(bi-directional GIOP)的概念，但是为了安全起见，只有在 Server 和 Client 在同一台机器上的情况下才允许。

在 CORBA 3.0[2]中，GIOP 代理中规定了双向 GIOP，即指只利用一条 GIOP 连接就可以进行数据/调用的双向传输，而无须建立第二条 TCP 反向连接。这使得分布式环境下的回调和 CORBA 异步通讯成为可行。使用双向 GIOP，能够保证两个方向上连接的生命周期相同，杜绝了额外配置 GIOP Proxy 而产生的安全隐患。但是需要注意的是，如果需要建立回调连接时，Client 端的 InBound 防火墙不能拒绝 Server 端发出的请求报文，否则回调无法实现。另外，Client 端还可能将其它 CORBA 对象暴露给 Server，应当对 Server 的行为加以限制。除此以外，由于 Client 和 Server 可以互相调用，这可能会因程序设计不当导致系统死锁。因此我们认为需要进程进行监控，这可以利用超时机制、调用关系图判定等方法解决。

3.3 定址方式

为了使报文从内部网络穿越防火墙正确地到达目的地，应对它们的路由进行调整。如果 Client 与 Server 之间至少存在一个防火墙，则应在请求报文/IOR 中加入一定的标记，以保证防火墙能够对其正确地转发。在使用 TCP Firewall 时，Client Object 只能与防火墙建立联系，所以请求报文中不会出现 Server Object 的地址及端口号；而 Server Object 为了使 Client Object 能够正确地调用 IOR，将 IOR 中的 Server 地址及端口替换成防火墙的相应地址及端口。使用 SOCKS 时，Client 只要使用 SOCKS 的运行库，SOCKS 会自动建立它与 Server 的连接；而 SOCKS 对于 Server 是透明的，因此 IOR 中无需对地址/端口进行映射。如果 GIOP Proxy 是 OutBound 防火墙，ORB 需要获取它的 IOR；如果 Proxy 是 InBound 防火墙，Server Object 的 IOR 中应带有 Proxy Object 的 IOR。当然，如果 Server 与 Client 同在一个防火墙保护的网络内部，IOR 则可以直接传至 Client，而无须加上标记部件；Client 也可以直接与 Server 通信。

我们认为透明 Firewall 将是 Firewall 的发展方向，定址的主要工作应放在 Firewall 内部和 ORB 内部进行；而 Client/Server 现实调用时，应尽可能少地涉及具体的 Firewall。所以 TCP Firewall 应在保留自身优点的同时，融入 SOCKS 的思想，将安全策略的实现从具体服务调用中分离出来，形成模块化。这样既减轻了用户的负担，也有利于安全控制的灵活配置。

3.4 数据加密

为了防止数据在不安全的网络上被监听以及身份认证的需要，需要对传输的数据进行加密，防火墙必须具备这样的能力。CORBA 防火墙 RFP[1]中指出，防火墙可以采用 IIOP/SSL 或 SecIOP 方式，以保证数据传输的安全性。前者是一种 TCP/IP 上的网络传输安全协议，为运输层防火墙提供安全功能；后者处于 GIOP 下层，但独立于 GIOP，它在 GIOP 基础上定义了消息的一种安全传输格式，针对图1的防火墙结构，目前有两种加密模型：一种是在两个防火墙之间传输的数据进行加密，这种模型适用于 Server 与 Client 各自所处的网络环境安全性良好、而两者之间连接不安全的场合，即 normal 模式。如果 Client/Server 对 GIOP Proxy 不十分信任时，可以采用 passthrough 模式，即在 Client 和 Server 之间的整个通讯链路上加密。我们可以利用 GIOP Proxy 所提供的安全策略实现网络上的安全数据传输，而 Client 与 Server 之间应具有一定的认证机制，GIOP Proxy 只提供数据的转发而不对报文过滤。

结束语 CORBA 防火墙的规范、实现和完善，为分布式应用提供了一把保护伞，而文中介绍的 CORBA Firewall 特点各异，提供的服务层次也不尽相同，所以不能对其优劣一概而论，更不能把它们作为唯一的安全保护手段。只有合理地综合利用多种防火墙，充分发挥它们的优点，并改善软件部件布局的合理性，加强管理安全防范意识，充分发挥其他安全策略的作用，才能提高分布式系统的安全可靠性。

参考文献

1 OMG. CORBA Firewall Security RFP. Object Management Group, November 1997
2 Siegel J. A Preview of CORBA 3. Computer, 1999(May): 114～116
3 Otte R, Patrick P, Roy M. Understanding CORBA. Prentice Hall PTR 1996
4 汪芸. CORBA 技术及其应用. 东南大学出版社, 1999
5 Schreiner R. Whitepapers & Experiments. Available at: http://www.technosec.com/.
6 Amoroso E, Sharp R. Intranet and Internet Firewall Strategies. Ziff-Davis Press, 1996
7 OMG. CORBA Firewall Security + Errata. Object Management Group, July 1998
8 OMG. The Common Object Request Broker: Architecture and Specification, Revision 2.2. Object Management Group, February 1995