Linux中构造基于ACE的安全高效通信平台

王春枝,冯旭刚
(湖北工业大学 计算机学院,湖北 武汉 430068)

摘 要:当把Linux操作系统作为大型安全管理系统的服务器的时候,构建安全的通信平台显的尤为重要。自适配通信环境(ACE)是一种面向对象(OO)的工具包,它实现了通信软件的许多基本的设计模式。文中介绍了ACE和整个系统模型的设计,然后重点阐述了如何利用SSL实现安全通信,以及数据传输过程中的效率问题的解决方案。实践中经过sniffer抓包分析和压力测试,可以达到预期要求。

关键词:ACE;SSL网络;安全
中图分类号:TP311 **文献标识码**:A **文章编号**:1005-3751(2006)04-0228-02

Construct Secure and Efficient Communication Platform Based on ACE in Linux

WANG Chun-zhi, FENG Xu-gang
(Department of Computer Science, Hubei Technical University, Wuhan 430068, China)

Abstract: Constructing a safe and efficient communication platform is the most important when using Linux OS as the server of a large network security management system. ACE(Adaptive Communication Environment) is a kind of OO(Object Oriented) tool kit. It implements many basic designing pattern of communication software. In this thesis, introduce the ACE and the design of the whole system simply. And then describes how to implement secure communication by SSL and solution of increasing efficiency of the data transferring. In practice, using sniffer to capture packet and pressure test, the communication system can work correctly.

Key words: ACE; SSL network; security

1 ACE概述

Adaptive Communication Environment (ACE)[1]是一种免费开放源代码的面向对象框架结构,该结构实现了许多并行通信软件的核心设计模式。ACE提供丰富的C++ wrapper facades,以及可跨平台执行通信软件的基本任务的框架对象。ACE提供的基本任务包括事件分离与事件处理的分发、信号量处理、服务初始化、进程间通信、共享内存管理、消息路由、分布式服务的动态配置、并发执行与同步。

ACE的主要优点有:高可移植性,增强软件的质量,高效率与可预见性,可容易地整合为高级的中间件。

ACE OS适配层:并发和同步,进程间通信(IPC)共享内存,事件多路分离,显式动态链接,文件系统机制。

OS接口的C++ Wrapper Facade:并发和同步组件,IPC和文件系统组件,内存管理组件。ACE含有一个高级的网络编程框架,集成并增强了较低层次的C++ Wrapper Facade。该框架支持并发分布式动态配置的应用。ACE的框架部分包含以下组件[2]:

* 事件多路分离组件:ACE Reactor(反应器)和Proactor(前摄器)是可扩展的面向对象多路分离器,它们分派应用特有的处理器,以响应多种类型的基于I/O、定时器、信号和同步的事件。

* 服务初始化组件:ACE Acceptor(接受器)和Connector(连接器)组件分别使主动和被动的初始化任务与初始化一旦完成后通信服务所执行的应用特有的任务去耦合。

* 服务配置组件:ACE Service Configurator(服务配置器)支持应用的配置,这些应用的服务可在安装时和/或运行时动态装配。

* 分层的流组件:ACE Stream组件简化了像用户级协议栈这样的由分层服务组成的通信软件应用的开发。

* ORB适配器组件:通过ORB适配器,ACE可以与单线程和多线程CORBA实现进行无缝集成。

2 系统模型设计

当今的大型内部安全管理系统的设计的主流思想都是控制台与代理端不直接通信,过去的安全管理软件一般都设计成C/S模式,这样作为Server的一端一般就是处理一些来自Client的连接。这种设计思想的主要优点是

收稿日期:2005-07-08
作者简介:王春枝(1958-),女,湖北武汉人,教授,研究方向为计算机网络安全。

思想简单明了,效率较高,程序设计相对来说也是比较简单。但是这种方法,由于控制台和数据库都在同一个操作系统上,给系统的稳定性带来了很大的隐患。在大型系统中,这个缺点可能是致命的。为了解决这个问题,人们提出了 CSA 模式[3],如图 1 所示。

console⇔server⇔agent

图 1　系统结构模式图

这样控制台和数据库分开,同时,控制台不跟代理端直接通信。虽然增加了整个系统的复杂度,但是结构更加清晰,整个系统更加稳定。

鉴于 Linux 系统被越来越多地作为系统服务器使用,且表现出色,因此把 Linux 作为这里 Server 端的操作系统。

下面就如何通过 ACE 在该系统上建立稳定安全和高效的通信和数据库平台进行详细描述。

3　具体实现

3.1　编程的基本过程

首先,在 Linux 系统的自启动文档中添加两项,一个是服务程序的启动,一个是数据库的自启动。

对于服务程序,首先,建立 SSL 通信,然后对所有的节点生成一个节点链表。具体方法是生成一个节点链表类的对象。然后对反应堆进行实例化 ACE_Reactor::instance(reactor),同时建立读和写类的对象,以及心跳包(用来定时检测 Agent 是否在线),同时开 3 个端口:2001,2002,2004,其中 2001 端口用来监听来自 Agent 端的连接,而 2002 和 2004 分别用来与 Console 端和 Agent 端传送文件。最后执行 ACE_Reactor::run_event_loop (),也就是进入无限循环。这样服务器的主要功能就是两点,第一就是数据的存储,把来自 Agent 端和 Console 端的数据保存到数据库;第二就是数据的转发,比如,把来自 Console 的命令传递给 Agent,只是作简单的数据转发。

3.2　SSL 通信的实现

因为整个系统本身是内部安全管理系统,所以系统自身的通信安全必须得到保证。故系统各端之间的通信必须是加密的。为了保证通信过程能够顺利进行,服务程序最开始从证书认证管理中心 CA 系统得到证书,用自己的私钥解密 Console 端和 Agent 端发送的数据,同时用 Console 端和 Agent 端的公钥加密传送出去的数据。这样就可以保证整个系统的安全通信,防止局域网内被截获明文。

实现部分的部分源码如下:

```
ACE_SSL_Context * SSL_Context = ACE_SSL_Context::instance ();
SSL_Context -> set_mode (ACE_SSL_Context::TLSv1_server);
...
SSL_Context -> load_trusted_ca (ROOT_CERT, CERT_DIR); //加载认证中心证书
SSL_Context -> certificate (SERVER_CERT, SSL_FILE_TYPE_PEM)//加载用户证书
SSL_Context -> private_key (SERVER_KEY, SSL_FILE_TYPE_PEM))//加载私钥
SSL_Context -> verify_private_key (); //验证私钥
```

以上就是 SSL 通信过程的初始化。

在通信的过程中,例如对于命令由 Console 向 Agent 传输的过程中,

class iGuard_command_done: public ACE_Task < ACE_MT_SYNCH>

编写该类的构造函数如下:

```
iGuard_command_done::iGuard_command_done (void)
{
  iGuard_TRACE ("iGuard_command_done::iGuard_command_done");
  command_ = this;
  this->open();
  this->activate (THR_NEW_LWP);
  this->water_marks (ACE_IO_Cntl_Msg::SET_LWM, COMMAND_MQ_LEN);//加载数字水印
  this->water_marks (ACE_IO_Cntl_Msg::SET_HWM, COMMAND_MQ_LEN);
}
```

同时,在通信过程中的 I/O 句柄是:ACE_SSL_SOCK_Stream * stream_;这样整个通信过程就可以在保密的情况下正常进行。

3.3　数据传输过程中的效率解决

数据传输的过程中,有两种形式的数据,一种是简单数据的获取以及命令的转发,由于这两类数据都比较小,因此,传输过程中,以数据报文的形式传递,可以达到理想的效果。当传输的是文件的时候,尤其是当文件比较大的时候,如果继续采用数据报的形式,系统效率将会非常低。因此,在实际系统中,采用了数据流的方式。但是,这个文件传输进程将会长期占用所分配的连接,而当又有新的文件传输请求连接的时候,这个连接请求将会被阻塞,造成应该上传数据的丢失。

为了解决这个问题,有两种方法可以选择,一种是通过设置标志位,然后保存一个队列,队列的每个节点保存所要传送文件的相关信息,当有新的文件需要传输的时候,首先判断该标志位,如果该标志位为空,则建立连接,然后开始传送,如果标志位不为零,表示已经有文件在等待,就给队列添加节点。另外一种方法是[4],使用多线程的方法,这种方法与第一种方法相比有明显的优点,那就是效率更高,但实现起来比较麻烦。可是,由于该系统对效率要求比较高,而且相对来说,系统资源比较充足,所以选择了第二种方法。

如下是写过程类的构造函数:

(下转第 233 页)

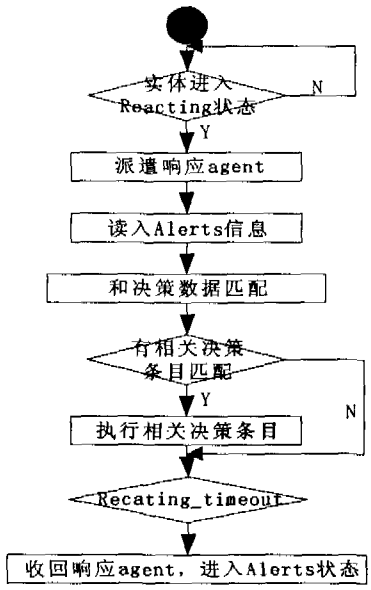

图 4　响应 agent 执行框图

一旦有特殊的事件使得实体进入 Reacting 状态,响应 Agent 就在决策数据库中搜索相关的条目并更新网络域的配置。根据事件的属性、攻击的形式和攻击的目标的匹配来执行相关的决策条目,如数据流的阻止、整形等。决策条目所对应的 action 的执行时间在实施中是由 Reacting_timeout 配置参量决定的。每当实体处于 Alerted 状态时,响应 agent 还要将信息传送给管理控制台,以便子网管理员做出及时响应。

2　结　论

系统所采用的移动 agent 和分布式协同机制为抵御 DDoS 攻击带来了很多优势:

(1)基于移动 agent 的分布式协同体系结构,减轻了网络的负载,提高了工作性能;

(2)系统采用的 Aglets 系统和 Java 语言实现,使得系统能够跨平台在异构的环境中工作;

(3)检测 agent 的可移动性,增强了系统的抵抗 DDoS 攻击的能力;

(4)Alerts 和 Heartbeat 的通信格式能够确保系统通信的有效性;

(5)在各个网络域在本地进行检测并做出响应,使系统能够最小化检测和响应的时间并不需要回溯机制。

当然,系统仍需要进一步的完善,以增进移动 agent 间通信的安全性,更好地抵御未来的 DDoS 攻击。

参考文献:

[1] 徐　恪,徐明伟,吴建平. 分布式拒绝服务攻击研究综述[J]. 小型微型计算机系统,2004,25(3):337－346.

[2] 严　毅,宁　葵,李陶深. 分布式拒绝服务攻击手段及其防范技术研究[J]. 微机发展,2004,14(9):81－83.

[3] Kolaczek G, Pieczynska－Kuchtiak A. A Mobile Agent Approach to Intrusion Detection in Network Systems [Z]. Springer－Verlag GmbH. Lecture Notes in Computer Science. /2005.842 － 849.

[4] Vigna G, Cassell B, Fayram D. An Intrusion Detection System for Aglets[Z]. Springer－ Verlag GmbH. Lecture Notes in Computer Science. 2002.64－77.

[5] 肖建华,张建忠. 基于移动 agent 的分布式入侵检测系统 MAIDS 的设计与实现[J]. 计算机工程与应用, 2003,17:164－165.

(上接第 229 页)

```
iGuard_read::iGuard_read()
{
  iGuard_TRACE ("iGuard_read::iGuard_read");
  data_block_ =0;
  stream_ =0;
  this→activate (THR_NEW_LWP); //开新的线程
}
```

这样,当在第一个连接正在传送文件的时候,假如又来了一个文件传输的请求,系统将开一个新的线程用来处理这个请求,因此多个传输过程可以同时进行[5]。

4　结束语

整个系统经过快一年的时间,终于开发成功。其中,对于 Server 端的通信平台也达到了预期的效果,利用 Sniffer 把接入的一个网卡设置成混杂模式,用 hub 组网,分析截获的数据包,均为密文,而且,在拥有近 200 台电脑的网吧进行压力测试,网吧电脑自由上网一个月后,服务器运行依然正常。

总之,在 Linux 环境下,利用 ACE 合理地搭建通信平台,能够做到安全、高效。

参考文献:

[1] 何　青. ACE 在开发健壮可靠的 C++ 系统中的应用研究[J]. 微机发展,2005,15(5):43－45.

[2] Schmidt D C, Huston S D. C++ 网络编程·卷 1:运用 ACE 和模式消除复杂性[M]. 於春景译. 武汉:华中科技大学出版社,2003.

[3] Buschmann F, Meunier R, Rohnert H, et al. Pattern－Oriented Software Architecture － A System of Patterns[M]. USA:Wiley and Sons,2003.

[4] Huston S D, Johnson J C E. ACE 程序员指南:网络与系统编程的实用设计模式[M]. 马维达译. 北京:清华大学出版社,2004.

[5] Schmidt D C. Acceptor and Connector: Design Patterns for Initializing Communication Services[A]. in Martin R, Buschmann F, Riehle D. Pattern Languages of Program Design [C]. Reading, MA: Addison－Wesley, 2002.