# IPv6下基于病毒过滤防火墙的设计与实现

张玉芳[1] 熊忠阳[2,3] 赖 苏[1] 张 科[1] 刘 君[1] 王银辉[1]
(重庆大学计算机学院[1] 电气工程博士后流动站[2] 信息与网络管理中心[3] 重庆 400030)

**摘 要** 包过滤防火墙无法检测出网络病毒,因此对其研究很有必要。设计的防火墙屏蔽了Linux自身的TCP/IP协议栈,重新构建了适合防火墙专用的TCP/IP协议栈,完成了防火墙上TCP协议的连接保持、数据包确认、文件传输等功能。防火墙主要考虑了HTTP协议下的文件过滤,使得内网主机在通过HTTP协议下载文件时自动过滤病毒文件,保证内网主机的安全;防火墙以Linux可加载内核模块形式实现,可以过滤链路层以上的各层;为提高病毒检测速度,提出了将病毒检测软件运行在核心态的方法。实验结果表明:设计的防火墙在性能和功能上都达到了预期目的。

**关键词** IPv6,防火墙,透明模式,病毒过滤
**中图法分类号** TP309 **文献标识码** A

## Design and Implementation of IPv6 Transparent Firewall with Virus Filtering

ZHANG Yu-fang[1] XIONG Zhong-yang[2,3] LAI Su[1] ZHANG Ke[1] LIU Jun[1] WANG Yin-hui[1]
(Department of Computer Science[1], Post-Doctorial Research Station of Electrical Engineering[2], Network and Information Management Center[3], Chongqing University, Chongqing 400030, China)

**Abstract** The existing packet filtering firewall is helpless for the spread of network virus, making a study on the designed of anti-virus firewall is very necessary. This paper designed and realized an experimental anti-virus firewall in IPv6. The firewall was realized in the form of kernel loadable module, and can be used to filter virus above link layer. The special TCP/IP used for firewall is rebuilt because transparent mode firewall cannot be run in the original TCP/IP of Linux. The firewall can get and resolve IPv6 packets and the IPv6 in IPv4 tunnel packet. It can also traverse every extension header until higher protocol layers. The firewall uses open source software to filter the virus in the network and ensures network security. For working in the kernel mode, the detecting speed was improved. The experimental results show that the performance and function of the firewall achieve the desired objectives.

**Keywords** IPv6, Firewall, Transparent mode, Virus filter

防火墙仍然是防御网络入侵者最有效的机制。随着IPv6协议的日渐完善,其安全问题也随之提上日程。IPv6防火墙的研究和产品大多都只针对包过滤问题,没有考虑病毒问题;而目前影响网络最大的安全因素就是网络病毒,所以对IPv6下的病毒防火墙的研究有着重要意义。

本文结合当前的病毒检测技术,对IPv6下集成病毒过滤防火墙的设计和实现进行了探讨,并在Linux环境下实现了一个防火墙系统。后续内容安排如下:第2节概述IPv6下的防火墙系统,防火墙各个模块的实现方法在第3节介绍,实验安排在第4节,最后是小结。

## 1 防火墙系统概述

### 1.1 防火墙系统功能概述

IPv6环境下的防火墙需要实现两个功能。一是透明模式;二是病毒检测功能。本文只探讨和实现了HTTP协议下的网络文件扫描和转发。内网主机用户通过HTTP协议进行文件下载时,必须通过防火墙的病毒扫描,从而保护内网主机。

由于本文主要针对HTTP协议实现病毒过滤,防火墙对不同类型的TCP进行不同处理:不含应用层数据的数据包只是传输层TCP协议的交互,所以防火墙对其只进行包过滤、状态检测过滤;只含应用层交互数据的数据包是HTTP客户和服务器进行交互的数据包,防火墙也对其进行相同的处理;而对于含应用层数据的数据包,不仅要对其进行包检测、状态检测过滤,对其中的文件内容也要提取,随后进行文件内容的检测。

### 1.2 防火墙系统功能模块组成

防火墙的处理对象就是在网络上传输的各类报文。由于每种报文都有它的功能、特点,防火墙对每种报文的处理应该有所区别,本文正是依此划分了防火墙的功能模块,主要功能模块如图1所示。

到稿日期:2008-10-19 本文受教育部留学回国人员启动基金(教外司留[2007]1108-10)和中国博士后科学基金(20070420711)资助。
张玉芳(1965-),女,博士,副教授,主要研究方向为网络与信息系统,E-mail:zhangyf@cqu.edu.cn;熊忠阳(1962-),男,博士,教授,主要研究方向为并行处理与网络;赖 苏(1973-),男,博士生,主要研究方向为网络;张 科(1982-),硕士,主要研究方向为嵌入式系统。

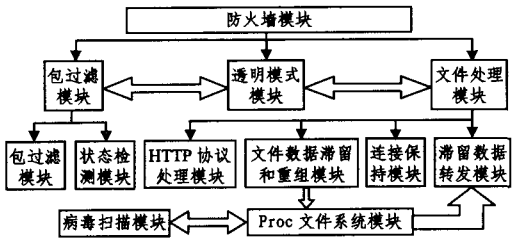


图 1　系统功能模块图

图中空心箭头指示两个模块间的通讯方向。系统通过透明模式模块获得通过防火墙的所有数据包，得到的数据包再交付给包过滤模块进行包过滤和状态检测模块进行过滤。检测合法后，如果该数据包中包含有应用层内容(实验中只考虑 HTTP 协议)，则将该包发送给 HTTP 协议处理模块进行解析，解析出数据包中 HTTP 协议数据，判断 HTTP 数据是客户服务器之间的交互数据还是文件内容。对于交互数据进行 HTTP 状态的安全性检查；对于含有文件内容的数据包，HTTP 协议处理模块将其交付给文件处理模块。文件处理模块首先滞留数据包，如果文件内容没有传输完毕，则等待；否则利用病毒扫描模块对文件内容进行病毒检测并将结果传递给 Proc 文件系统以通知防火墙该文件是否被病毒感染。直接丢弃被感染的文件；而不含病毒的文件，则需要调用滞留数据转发模块将文件内容转发给客户端。

## 2　系统设计实现

### 2.1　透明模式实现

防火墙有 3 种透明模式的实现方法：ARP 透明代理、网桥模式、网卡置于混杂模式。3 种方式各有优劣，ARP 方式较为复杂，对所有要解析的内网中的 IP 都用与内网相连的防火墙端口的 MAC 地址来回答，路由器如果认为是网络欺骗的话，那么防火墙则无法正确执行；另外防火墙的 MAC 地址对外暴露，留下安全隐患。网桥模式靠内部端口管理软件和网桥协议实体共同完成，程序实现复杂，与防火墙的网络操作系统、整体构架结合不紧凑，影响执行效率。网卡置于混杂模式实现简单，但涉及底层编程较多。

本文采用将网卡置于混杂模式，使网卡可以处理接收到经由本网卡的网络数据包。具体利用 Linux 的特点[1]，保留各层头部信息，从而不用再重新查找目的 MAC 地址。网卡置于混杂模式在 Linux 内核态下的实现方法描述如下。

(1)将网卡置于混杂模式：利用内核提供的函数 dev_set_promiscuity 将网卡置于混杂模式；

(2)数据包的接收和发送

由于防火墙运行在透明模式，防火墙不具备 IP 地址，不能使用原防火墙的 TCP/IP 协议，需要屏蔽原有协议栈对数据包的处理，使防火墙成为唯一处理 IP，ARP 数据包的程序。具体包括数据包的接收、转发等操作。

文献[2]运行在系统 Socket 层，通过函数调用来获得数据包。此方式工作在用户态，系统性能不高。本文在核心态下处理数据包，提高了系统性能。

核心态下转发数据包的工作相对简单，只要保存原来的数据包内容，使数据包在经过防火墙后内容不发生改变。

数据包的接收则稍显复杂，其过程描述如下。Linux 通过网络核心层函数从网络接口层接收发送来的数据，再交给软中断处理函数处理。

当系统收到一个网络数据包时调用 netif_receive_skb()函数，遍历该函数中对此协议的数据链表，如果找到合适的处理函数，则使用该上层协议处理函数进行处理，相当于网络数据包向上层传递。虽然此时防火墙处理函数可以处理 IP 数据包，但由于 netif_receive_skb()将遍历该链表，系统原来的处理函数 ip_rcv()仍然有效，因此需要屏蔽。实验先利用 dev_add_pack()将防火墙的处理函数插入链表头，接着将下一结构指针设置为空，从而达到屏蔽系统原来处理函数的目的。

因此本文的防火墙处理函数成为唯一的数据包处理函数，保证了自己加载上去的协议处理函数能接收到相应的协议数据包，而系统加载上去的协议处理函数收不到数据包。为保证防火墙程序退出后系统仍然能够接收任何协议的数据包，在编程时设置一个全局变量初始赋值为截断后的链表，当防火墙模块卸载时将该链表加入对应位置。这样模块卸载后系统仍然可以继续接收所需要处理的包。

采用网卡置于混杂模式来实现透明模式，并利用 Linux 核心栈数据结构的特点保留了各种头部信息，使得防火墙的数据判断、转发不需要重新获得 MAC 地址，使得系统在内核态保证防火墙加载上去的协议处理函数成为唯一能收到相应数据包的程序，从而彻底屏蔽了操作系统对数据链路层以上的数据包处理流程，消除了操作系统所固有的协议栈指纹，为 TCP/IP 协议栈重构打下了基础。

### 2.2　HTTP 协议处理

跟踪客户端发出的 HTTP 报文确定文件内容的开始部分。如果是 Request 报文，则对服务器返回的第一个响应报文进行分析，如果返回状态为 OK，说明文件存在并出现在后续报文中。当然，文件内容的开始部分也可能由第一个响应报文的实体主体部分来传送，这时文件内容就在连续两个“回车”(CR)和“换行”(LF)之后，否则就是在第一个响应报文之后的报文中。

文件内容的结束有两种情况：一是用户主动请求访问该文件时，服务器会在响应报文的首部提供文件大小；二是非用户主动请求访问，服务器返回给客户的文件信息。此时服务器并不告诉客户端文件的大小，而是通过主动关闭连接通知客户端文件是否传输完毕。

### 2.3　TCP/IP 协议栈的重构

#### 2.3.1　数据包的滞留

正常情况下，文件传输双方在建立 TCP 连接后，客户端向服务器发送一个文件请求报文，服务器在本地找到该文件，就向客户端发送该文件内容。一般的内容过滤防火墙仅检测传输的一两个报文，而病毒检测功能需要防火墙对传输的文件进行病毒扫描，这就涉及到数据包滞留和连接保持问题。

本文主要讨论用 HTTP 协议进行文件传输。通过对每个经过的数据包进行检测，滞留包含文件数据，并去除包括 MAC，IP，TCP 等头部信息，甚至包括 HTTP 的响应头部，得到该数据包中包含的文件内容。

#### 2.3.2　连接保持

对于不包含文件内容的数据包，防火墙只对单个数据包进行包过滤检测就转发给目的主机。由此造成的网络延时客户和服务器基本不会察觉，防火墙也不必考虑对两者 TCP 连

接的影响。但是加入病毒检测功能之后，防火墙需要滞留所有含有文件内容的数据包进行检查，通过后方可交与目的主机。此时防火墙完全阻隔了客户和服务器之间的联系，防火墙必须代替客户和服务器向对方发送确认信息，否则双方会因为长时间得不到对方响应而断开连接，导致文件传输失败。

要解决 TCP 的连接维持问题必然涉及 TCP 协议的规则内容。TCP 协议依靠数据发送方和数据包接收方之间的确认序号，终端主机才能确认对于接收到的报文是接受还是放弃。因此报文的序号在连接的维持与继续中有着非常重要的作用。

在文件传输过程中，客户的发送序号一直不变，只是对服务器发送序号进行确认；服务器因为没有收到客户端的应用层数据，其确认序号也一直不变，而发送序号随着文件内容的发送而变化。文件传输完毕，两者就发送结束标志关闭 TCP 连接。本文的解决方案是：防火墙在滞留数据包时，可以代替客户端向其发送确认报文，其发送序号不变；而在文件检测通过以前，防火墙只是利用客户向服务器发送的最后一个报文序号进行确认，即一直对最后一个报文确认，而不进行文件传输。在文件检测通过后，再将文件内容转发给客户。

防火墙在滞留文件的同时，分别伪装成客户和服务器向对方发送不含数据的、不占用发送序号，只有确认序号置位的无应用数据的 TCP 报文。由于防火墙为客户和服务器都提供了确认报文，这个回应对应用层是不可见的（TCP 报文中无应用数据），但是又符合通信客户端、服务器双方 TCP 协议处理流程，于是双方就维持连接。

这里巧妙利用 TCP 中的确认报文既维持了连接，又延缓了时间，使得防火墙得以完成文件数据的重组和检查。防火墙对重组后的文件数据进行检查，继续客户端、服务器之间的连接，将合法的原始 TCP 报文发出。

#### 2.3.3 数据包的转发

在确认文件没有病毒时，需要传递给客户端文件内容，此时可以关闭和服务器的连接。HTTP 协议使用 TCP 协议进行传输，所以必须考虑数据包的丢失、重发、超时重传、滑动窗口等内容。实验中考虑到实验的复杂性和局域网的高速率、高可靠性，仅使用了一个定时器每隔一段时间将数据包传输给客户端，对客户端发送过滤的确认包进行确认，对丢包重新发送。

## 3 文件的病毒检测

### 3.1 病毒检测软件

系统采用了开源项目 Clamav[3] 作为病毒扫描工具，对文件进行病毒扫描时，将文件内容和病毒库中的特征码进行比较，如果有配对的记录，则说明该文件被病毒感染。文中将 Clamav 运行在核心态进行通讯。

### 3.2 现有运行在核心态的解决方案

文献[5]通过修改 Linux 内核的装载程序将用户进程的段选择子直接赋值为核心状态，从而使一个普通的用户程序完全工作在核心模式下，该解决方案需要修改 Linux 核心源码，不适合一般应用。

文献[6]在系统调用处理程序中将用户进程的段选择子修改为核心状态，从而使用户进程在系统调用返回后运行在核心模式下。这种方法使得用户进程可以自由地在核心模式和用户模式下相互切换，通过核心模块也很容易实现对用户进程的安全检测；但会引发一些问题。

### 3.3 本文的解决方案

通过将用户程序的段选择子修改为核心态，可以非常方便地将用户进程运行在核心模式下，但这种方法同时会引起一些问题。如当前进程指针存取失败，会使很多系统调用失败，甚至会因为访问非法地址导致系统崩溃。以下给出引发问题的具体解决方案。

#### 3.3.1 核心堆栈过小问题

在 Linux 中一个进程同时拥有用户堆栈和核心堆栈[7]。在标准的 Linux 配置中，内核堆栈只有 4K 或 8K 的空间。为了避免内核堆栈的溢出，可以利用内核分配更大的一块连续内存空间作为用户进程新的核心堆栈。在 Linux 内核中，通过系统调用获得多个连续页面作为新的内核态堆栈；在用户进程返回用户态后，核心堆栈恢复为原来的堆栈。

#### 3.3.2 用户进程调用内核函数

由于链接程序不能正确识别解析核心函数，用户程序不能简单地通过函数名称对其调用，必须先获得函数的实际地址。文献[5,6]通过顺序读取 System. map 文件来获得，这种方法由于涉及文件操作，因此效率不高，且增加了编程人员的工作量。

在系统启动后，Linux 内核映像被加载到内存的内核区域，并锁定在该区域中。Linux 在编译内核时将未导出内核符号的地址存放在 System. map 文件中，因此对于已经在文件 System. map 中有记录的未导出内核符号，其内存地址通过在文件中查找就可得到；对于文件 System. map 中没有记录的内核符号，可以通过查看内核代码，查找与其相邻的符号、通过指针访问他的其他变量、通过已知其值的情况下搜索也可以找到其地址。

### 3.4 ClamAv 核心态运行实验

实验中 ClamAv 分别运行在用户态、一般的核心模式和本文改进后的核心模式下，对相同的文件进行病毒扫描。实验平台为 CPU Duron 1. 1MHz，内存 256M，Linux 内核 2. 6. 10，结果如表 1 所列。

表 1 3 种方法实验结果的对比

| 工作方式 \ 文件规模 | 文件一 (54kB) | 文件二 (1. 47M) | 文件三 (44. 2M) | find 命令查找同一文件 |
|---|---|---|---|---|
| 用户模式 | 1. 60μs | 956ms | 65. 4s | 189ms |
| 一般的核心模式 | 1. 48μs | 903ms | 62. 8s | 164ms |
| 改进的核心模式 | 1. 13μs | 826ms | 59. 1s | 146ms |

从实验结果可以看出，将用户程序运行在核心模式下确实能提高用户程序的性能，而本文的方法避免了调用核心函数时执行 INT 0x80 软中断，性能有了进一步的提高，从而验证了上述解决方案的可行性和有效性。

## 4 实验及结果分析

本系统由软件实现，涉及 TCP/IP 协议重建、防火墙与病毒扫描引擎的集成等。选用的开发工具是 Linux 下的 C 语言，编译器 GCC，调试工具 kgdb[8]，病毒扫描采用开源项目 ClamAv。

### 4.1 透明模式功能的测试

实验中，内网主机分别在没有防火墙和加入防火墙的情

况下对网络进行访问，并在该主机上使用 ethereal 对网络数据包进行抓包分析，以查看防火墙对网络访问的影响。结果表明，内网主机在没有任何配置的情况下，可以自由进行各种网络活动，内网主机可以进行如浏览网页（使用 HTTP 协议）、FTP 文件下载、Telenet 远程终端、SMTP 收发邮件等。

### 4.2 病毒过滤功能的模拟测试

防火墙病毒过滤功能是在内网主机向外部服务器请求文件时，根据病毒检测工具返回的结果进行处理。如果是正常文件，防火墙如实转发，内网主机正确得到文件；如果文件被病毒感染，防火墙将文件内容丢弃，内网主机得不到文件。

实验中，内网主机使用重庆大学互联网技术实验室安装 IPv6 协议的主机对外部 Web 服务器进行访问。因为文件内容存放在外部 Web 服务器上，上面不存在病毒文件，所以对于病毒过滤功能只能模拟实现。

分别进行了两个实验：一是正常文件的处理，即当内网主机访问文件时，防火墙能如实交付文件内容，内网主机得到文件。使用一台主机通过浏览器访问上海交通大学 IPv6 实验站上 telnet. exe 文件，内网主机如实得到了该文件内容。二是访问病毒文件模拟实现，因为实验环境中不具备访问病毒文件的条件，在调用病毒检测 API 函数后，人为地将其病毒检测结果设置为病毒文件来模拟病毒文件的环境。实验中，主机访问上海交通大学 IPv6 实验站上的一个可执行文件，病毒检测工具返回结果为病毒。内网主机最终没有得到文件内容，并因为超时而断开了网络连接。

### 4.3 性能测试

从两方面对设计的防火墙进行性能测试：一是测试不包含文件数据的数据包的网络延迟，另一对包含文件数据的数据包网络延迟进行测试。对于第一种测试，通过多次访问中国下一代教育科研网 Cernet2（http://www.cernet2.edu.cn/）、重庆大学 IPv6 网站（www.cqu6.edu.cn）、上海交通大学 IPv6 实验站（ipv6.sjut.edu.cn）、日本 KAME 计划网站（www.kame.net），分别对内网主机发出和得到的不包含文件数据网络数据包延迟进行统计。实验数据采用多次访问的平均值，实验结果如图 2 所示。

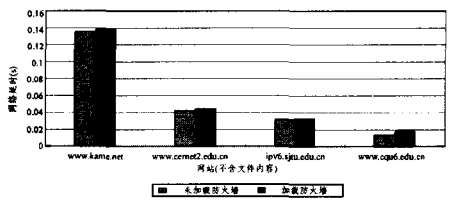


图 2 不包含文件数据的网络延迟

对于第二种测试，通过对上述各个网站上提供的不同大小文件进行访问，测试内网主机得到数据包的网络延迟，实验结果如图 3 所示。

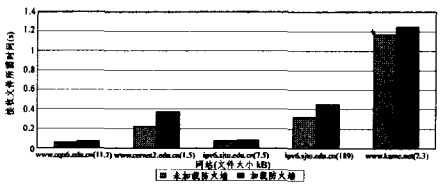


图 3 包含文件数据的网络延迟

从两个图中可以看出，对于不包含文件数据的数据包，防火墙只是对其进行包检测，引入的网络延迟并不明显。对于包含文件数据的数据包，由于防火墙需要对整个文件内容进行滞留，网络延迟明显增加。防火墙带来的额外的网络延迟与网络状况和请求的文件大小有关。网络状况越差、请求的文件越大，从文件请求到防火墙得到全部文件内容需要的时间越长，防火墙引入的网络延迟越大。但由于防火墙中加入了连接保持模块，内网主机仍然可以正确得到文件数据。

从实验结果可以看出，本文设计的防火墙实现了透明模式和病毒过滤功能，内网主机在不需要任何设置的情况下可以进行各种合法的网络访问，防火墙能够将其访问的病毒文件过滤，保证了内网主机的安全。同时防火墙引入的网络延迟也在合理的范围之内，基本实现了预期设计目标。

**结束语** 本文提出的防火墙以可加载模块化设计，实现了防火墙的透明模式，使得防火墙可以对链路层以上的各层进行过滤。防火墙屏蔽了 Linux 操作系统自身的 TCP/IP 协议栈，构建了适合防火墙的专用 TCP/IP 协议栈，完成了防火墙上 TCP 协议的连接保持、数据包确认、文件传输等功能。防火墙主要考虑了 HTTP 协议下的文件过滤，使得内网主机在通过 HTTP 协议下载文件时自动过滤病毒文件，保证内网主机的安全。并提出了将病毒检测软件运行在核心态的方法，提高了病毒检测软件的检测速度。实验结果表明，设计的防火墙在功能和性能上都达到了预期的目标。

本文的防火墙只考虑到 HTTP 协议传输文件的过滤问题，尚没有考虑其他协议，这些可以通过设计一个框架结构，在需要处理新的协议时，将该协议处理过程加入即可。同时没有考虑 IPv6 下 IPSec 协议。IPSec 的加密功能提供的是端到端的保护，并且可以任选加密算法，密钥是不公开的。防火墙不能解密就无从知道 TCP/UDP 端口号，给防火墙提出了很严峻的问题，这些都是值得今后进一步展开研究的。

## 参 考 文 献


[1] Benvenuti C. Understanding Linux Network Internals. O'Reilly, 2005, 12

[2] 申凤兰. 防火墙中透明模式和流过滤技术的研究与实现. 暨南大学, 2003, 4

[3] http://www.clamav.org/

[4] Jones M T. Access the Linux kernel using the /proc filesystem. http://www-128.ibm.com/developerworks/linux/library/l-proc.html? S_TACT=105AGX52&S_CMP=cn-a-l. 2006.3

[5] Maeda T. Safe Execution of User Programs in Kernel Mode Using Typed Assembly Language [D]. Tokyo: The Graduate School of the University of Tokyo, 2002

[6] 姜新，等. Linux 核心模式下的用户进程研究[J]. 计算机工程与应用，2004，4：118-120

[7] 陈莉君，冯锐，等. 深入理解 Linux 内核[M]. 北京：中国电力出版社，2004

[8] LinSysSoft Technologies. KGDB Document. http://kgdb.linsyssoft.com/index.html. 2006