

Linux 平台双协议栈主机网络管控系统设计与实现

贾 锐, 蔡皖东

(西北工业大学 计算机学院, 陕西 西安 710129)

摘 要: 为了应对基于 Linux 内核国产操作系统的日益普及, 同时网络安全应用又相对匮乏所造成的安全隐患, 分析 IPv4 向 IPv6 过渡技术中的双协议栈技术, 研究 Linux 内核网络栈结构, 采用 Linux 内核中 Netfilter 框架, 结合白名单策略, 设计并实现主机网络管控系统, 达到对双协议栈主机的网络访问数据进行管控的目的. 实验表明, 该系统可“无区别”地对两种网络数据流进行管控, 为 Linux 平台网络安全管理提供便利, 进而提高网络安全性.

关键词: Linux; IPv4/IPv6 双协议栈; Netfilter 框架; 白名单

中图分类号: TP311

文献标识码: A

文章编号: 1000—7180(2016)06—0014—04

The Design and Implementation of the Network Management System of the Dual Stack Based on Linux Platform

JIA Rui, CAI Wan-dong

(College of Computer, Northwestern Polytechnical University, Xi'an 710129, China)

Abstract: In response to security concerns caused by the increasing popularity of the Linux kernel domestic operating systems, while the network security applications lacked, analyse the dual stack technology of IPv4 to IPv6 transition technology, study the Linux kernel network stack structure, use the Linux kernel Netfilter framework, combine with the white list strategy, design and implement a host network management system. This system achieve the purpose to management and control the dual stack host network data. Experiments show that the system can be nondistinctivet control the two kinds network data flow, to facilitate the network security management of Linux host and thus improve the network security.

Key words: Linux; IPv4/IPv6 dual stack; netfilter framework; white List

1 引言

自从计算机网络技术兴起以来, 网络安全一直都是研究的热点问题. 国产操作系统主要是依托 Linux 内核进行二次开发, 因其市场化程度不高, 导致用于安全管理方面的应用软件相对匮乏. 由于网络泄密更多是在网络行信息交互中进行的, 如果能够对主机的网络行为进行管控, 就可以在很大程度上降低因违规的网络操作造成的泄密风险^[1]. 研究 Linux 主机网络行为的管控, 对推进以 Linux 内核为基础的国产操作系统的部署, 加强网络信息安全有着重要的现实意义.

IPv6 作为下一代互联网技术革新的核心, 已经从当初的实验网逐步演变为当前网络不可或缺的重要组成部分. 由于 IPv4 网络目前的支柱作用, IPv6 与 IPv4 将在相当长的时间内共存^[2]. Linux 内核对 IPv4 和 IPv6 两种协议栈的良好支持以及稳定优良的网络性能, 使其在局域网环境中占比呈上升趋势, 因而对 Linux 主机双协议栈网络行为管控开展研究是必要的.

2 研究现状及方案概述

从 1.1 内核开始, Linux 系统就已经具有了包过滤功能, 在 2.4 以后内核中开始采用全新的 Net-

filter 框架^[3]. 处于内核态的 Netfilter 与处于用户态的 IPTables 命令行工具相互协作实现数据包审查管控等防火墙的功能. 该命令行工具根据 IPv4 和 IPv6 数据包的不同分为 IPTables 和 IP6Tables 两种版本,必须以管理员身份通过人工操作输入一系列命令完成所需的功能,可视性及可操作性较差.

本文在研究 Linux 内核 Netfilter 框架的基础上,提出在 Linux 内核态将 Linux 主机网络行为所产生的 IPv4 和 IPv6 两种出栈数据包在网络层进行“无区别”的分析处理,结合白名单策略的实施,对本机进程流出网络栈的数据包作出接受 (ACCEPT) 或者丢弃 (DROP) 的选择,从而实现对 Linux 双协议栈主机网络行为的管控,加强局域网环境下 Linux 平台双协议栈主机网络行为的安全性.

3 关键技术研究

3.1 IPv4/IPv6 双协议栈技术

双协议栈技术 (Dual Stack) 是指在网络节点中同时具有 IPv4 和 IPv6 两个协议栈,它既可以接收、处理、收发 IPv4 的分组,也可以接收、处理、收发 IPv6 的分组. 对于 Linux 主机来讲,其内核默认编译了 IPv4 和 IPv6 两种协议栈,这使得 Linux 平台主机能顺利地接入 IPv4/IPv6 共存的网络环境,实现“即插即用”的良好用户体验^[4]. Linux 平台 IPv4/IPv6 双协议栈结构可以根据需要对外层应用产生的数据进行 IPv4 或者 IPv6 封装,而这种操作与上层应用及下层数据链路不具有严格的关联性,使得该技术能简单便捷的应用于 IPv4/IPv6 网络环境中. 具体如图 1 所示.

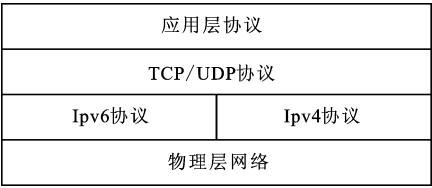


图 1 主机 IPv4/IPv6 双栈

当双协议栈主机发送数据包时,采用 IPv4 还是 IPv6 协议栈处理数据包取决于上层应用 (如 Ping 或 Ping6),其反映在报文里面就是目的地址是 32 位的 IPv4 地址还是 128 位的 IPv6 地址. 根据上层所使用的地址类型对双协议栈的工作方式作出如下约定:

(1)若应用程序 (如 Ping) 使用的目的地址是 IPv4 地址,则在网络层使用 IPv4 协议;

(2)若应用程序 (如 Ping6) 使用的目的地址是 IPv6 地址,且为本地在线网络,则在网络层使用 IPv6 协议;

(3)若应用程序使用的目的地址是 IPv4 兼容的 IPv6 地址,并且非本地在线网络,则使用 IPv4 协议,此时的 IPv6 将封装在 IPv4 中;

(4)若应用程序使用的目的地址是非 IPv4 兼容的 IPv6 地址,不是本地在线网络,就使用 IPv6 协议;

(5)若应用程序使用域名作为目标地址,则首先从域名服务器 (DNS) 解析得到对应的 IPv4/IPv6 地址,之后再根据地址的具体情况进行相应处理.

3.2 Netfilter 框架原理

对主机网络的管控,在网络层就是对流经内核网络栈的数据包进行干预. 对 Linux 网络数据包的干预处理依靠 Linux 内核的 Netfilter 框架实现^[5]. 该框架是 Linux 内核防火墙的实现框架,通过该框架在网络层实现对数据包的处理,使其和数据包本身完全剥离,进而在数据处理方面更具优势.

针对每种网络协议 (如 IPv4、IPv6、DECnet 等),Netfilter 框架定义一套钩子函数,并以链表的形式组织起来,这些钩子函数在数据包流经网络协议栈的几个关键点时依照之前预置的优先级被依次调用,并返回相应的结果^[6]. 数据包通过 Netfilter 框架的流程如图 2 所示.

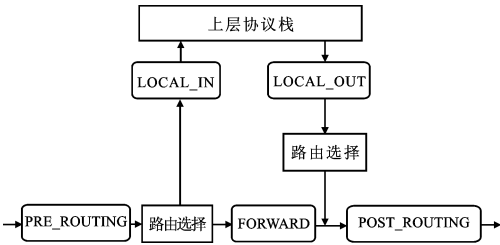


图 2 数据包流程图

数据包首先经过 PRE_ROUTING,经处理后经路由选择判断数据包是发往本机还是需要转发,如需要转发则由 FORWARD 点的钩子函数处理,再经由 POST_ROUTING 函数处理后传输到网络上. 如目的地址是本机则由 LOCAL_IN 函数处理后发往本地程序进行处理,本地进程 (Local Process) 产生的数据包则首先由 LOCAL_OUT 函数处理,经路由选择后再由 POST_ROUTING 函数进行处理后传输到网络上.

针对 IPv4 和 IPv6 分别注册一个或者多个钩子函数,当数据包被传递进 Netfilter 框架时,内核就

检查是否有模块对 IPv4 和 IPv6 协议注册了钩子函数,这样模块中分属不同协议的多个钩子函数就按照注册函数 `nf_hooks[协议][函数标号]` 注册钩子函数时所规定的优先级(priority)依次对数据包进行审查干预,而上述每个钩子函数必须返回下列预定义的 Netfilter 返回值 `NF_DROP`、`NF_STOP`、`NF_ACCEPT`、`NF_STOLEN`、`NF_QUEUE` 和 `NF_REPEAT` 中的一个。

重点关注 `NF_ACCEPT`、`NF_STOLEN` 和 `NF_DROP` 这三个返回值。返回值 `NF_ACCEPT` 意味着到目前为止,该数据包还是被接受的,且应当被递交到网络栈的下一阶段;返回值 `NF_STOLEN` 告诉 Netfilter“忘掉”该数据包,即该 Hook 函数将从开始对该数据包的处理,同时该数据包的资源未被释放,这个数据包及其 `SK_BUFF` 数据结构依然有效;返回值 `NF_DROP` 则表示该数据包被完全丢弃,所有为它分配的资源都被释放。

3.3 白名单策略文件

该系统对进入网络栈的数据包采用“一切未被允许的都被禁止”的白名单策略,即凡列入白名单的网络资源(目的 IP、协议、端口等)都是允许访问的,除此之外均禁止访问。这样可以将主机的网络行为限制在最小的范围,从而最大限度地保证自身和所处局域网其他资源的安全运行。但白名单策略来会给网络使用者造成一些不便,因此在安全性与便利性之间需要权衡。

为保障策略文件在未授权的情况下被修改,策略文件以文本加密的形式保存。策略文件的明文存储格式如图 3 所示。

分隔符\$	IP地址	分隔符@	协议类型	分隔符*	端口
-------	------	------	------	------	----

图 3 策略文件存储格式

策略信息逐条存放在文本文件中,目的 IP 地址、协议类型以及端口用“\$”、“@”及“*”特殊字符分割开。当模块加载时在内核中单独启动一个线程,以固定时间间隔(如 50 ms)从用户空间读取加密后的策略文件进行解密,逐条解析还原,存储在策略信息结构体 `Struct strategy_info` 中,以备数据包审查模块根据需要调用全部或者部分策略信息要素。策略信息结构体如下:

```
static struct strategy_info
{
    char a[]; // (源) 目的地址
    unsigned short p; // 协议
```

```
    unsigned short int t; // 端口
} strategy;
```

该结构体实际上针对挂载点上每个钩子函数形成了一个规则链,数据包进入钩子函数进行处理时,遍历比对规则链,若符合其中一条规则,就返回 `NF_ACCEPT`,至此该钩子函数结束;如果钩子函数链表还注册有其他钩子函数,则根据优先级次序,数据包将再次进入钩子函数处理环节;若未能与规则链中任何一条匹配成功,数据包则根据白名单策略中的默认规则丢弃(`NF_DROP`)。数据包在规则链处理流程如图 4 所示。

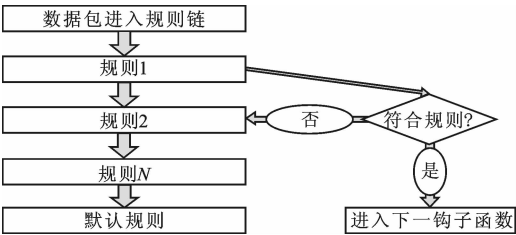


图 4 规则链流程

4 系统设计与实现

4.1 系统模型

Linux 平台双协议栈主机网络管控系统根据承担任务的不同,分为内核态和用户态两个部分。内核态部分以内核模块方式加载到内核,依托内核中 Netfilter 框架,在 `NF_IP_LOCAL_OUT` 和 `NF_IP6_LOCAL_OUT` 钩子点分别挂载钩子函数 `hook_func_ipv4` 和 `hook_func_ipv6`,并为上述钩子函数赋以最高的优先权 `NF_IP_PRI_FIRST` 和 `NF_IP6_PRI_FIRST`,确保钩子函数在数据包进入钩子点之后首先执行,实现在网络层对流出网络栈的 IPv4 和 IPv6 数据进行抓取、分析和审查的功能;用户态部分则以应用程序的方式完成策略文件获取并记录日志信息。上述两部分相互配合完成 Linux 平台 IPv4/IPv6 双协议栈主机网络行为管控。此外策略文件的解密和解析也在内核态完成,即该模块在插入内核进行初始化时启动一个线程 `kthread_run` (`read_strategy`, `dir_strategy`, "readstrategy"),实现从用户空间读取策略文件经解密解析后存储在策略信息结构体 `static struct strategy_info` 中。策略文件源于本地生成或者远端的控制平台传入。整个系统模型如图 5 所示。

4.2 双栈网络数据包审查

根据上述系统模型,为达到对 Linux 平台主机网络行为进行有效管控,实现对 Linux 平台双栈主

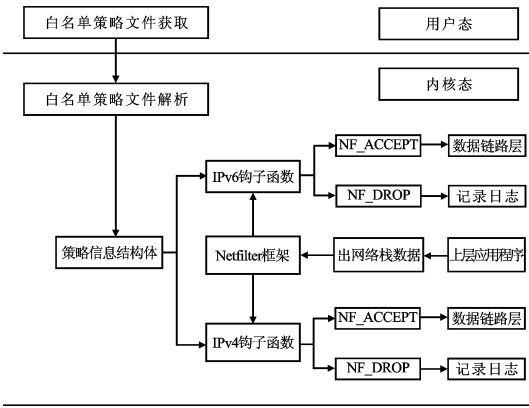


图 5 系统模型

机的安全防护,同时保证局域网网络资源的正常有序访问,需要在 Linux 平台双栈主机网络栈的“出口”设置“检查哨”,即针对从上层应用进入网络栈的 IPv4 和 IPv6 两种数据包,将钩子函数 hook_func_ipv4 和 hook_func_ipv6 分别部署在 NF_IP_LOCAL_OUT 和 NF_IP6_LOCAL_OUT 钩子点上.本地进程(Local Process)产生的数据包经传输层 TCP 或者 UDP 封装后,进入网络层进行封装,网络层会根据应用层程序产生的数据包版本属性选择封装 IPv4 包头还是 IPv6 包头,之后该数据包进入网络栈的 Netfilter 框架进行处理,流程如下:

- (1)获取包头数据分析其中的 IP 数据包版本字段,若为 4,数据包进入 NF_IP_LOCAL_OUT 钩子点等待处理;若为 6,则数据包进入 NF_IP6_LOCAL_OUT 钩子点等待处理;
- (2)在 NF_IP_LOCAL_OUT 钩子点和 NF_IP6_LOCAL_OUT 钩子点按照事先预置的钩子函数优先级(priority)分别调用钩子函数 hook_func_ipv4 和 hook_func_ipv6 对数据包进行处理;
- (3)上述钩子点的两个钩子函数在获取数据包中有用的要素后(如 IP 地址、传输层协议类型、端口等),与白名单规则链进行逐条比对;
- (4)根据要素与白名单规则链匹配的结果,钩子函数 hook_func_ipv4 和 hook_func_ipv6 返回 NF_ACCEPT(匹配成功)或者 NF_DROP(匹配失败),或者经默认规则丢弃该数据包;
- (5)再次进入(1),如此循环往复处理每个流出网络栈的数据包.

由于 NF_IP_LOCAL_OUT 和 NF_IP6_LOCAL_OUT 钩子点是本地网络数据流出网络协议栈进入网络的“必经之路”,因此在上述钩子点部署钩子函数对主机的网络数据进行审查比较好.

将钩子函数注册到 Netfilter 框架,需要对 nf_hook_ops 结构体实例化,之后使用 nf_register_hooks(nfho,ARRAY_SIZE(nfho)) 注册进 Netfilter 框架,即将 nfho 结构添加到 nf_hooks 数组的相应的成员链表中.struct list_head nf_hooks[NPROTO][NF_MAX_HOOKS]是一个二维数组,该二维数组的每一个成员均是一个链表.每一个链表的所有节点都代表一个协议在一个钩子点上的所有的钩子函数成员,其中 NPROTO(协议)有 32 个,而钩子点有 8 个.在同一个链表上,节点是按照优先级顺序排列的,优先级值最小的钩子函数变量存在链表的最前面,优先执行.

5 系统测试

校园网具备 IPv4 和 IPv6 两种网络环境,因此系统测试在校园网上进行,测试所用网络拓扑如图 6 所示.

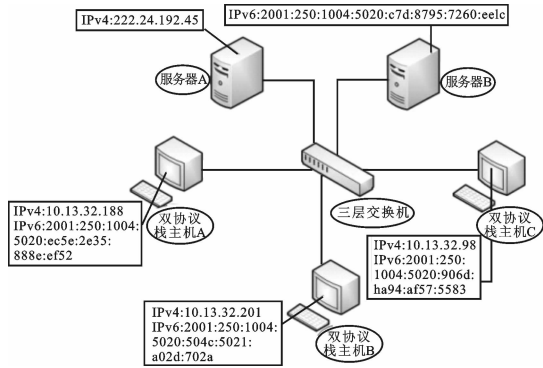


图 6 拓扑图

编辑策略文件 IP_Strategy.TXT 内容为
\$ 10.13.32.98@ICMP * 0
\$ 2001:250:1004:5020:906d:ba94:af57:5583@ICMPV6 * 0
\$ 222.24.192.45@ALL * 0
\$ 2001:250:1004:5020:c7d:8795:7260:eelc@ICMPV6 * 0

(注:策略信息中的“ALL”表示所有协议,端口“0”表示所有端口)

模块 IP_Filter.ko 在双协议栈主机 A 上以 root 权限用命令 insmod IP_Filter.ko 加载入内核运行.因为服务器 A、服务器 B 和双协议栈主机 C 的地址等信息在白名单策略文件中,因此双协议栈主机 A 可以使用 Ping 和 Ping6 命令分别与服务器 A、服务器 B 以及双协议栈主机 C 连通;但因双协议
(下转第 22 页)

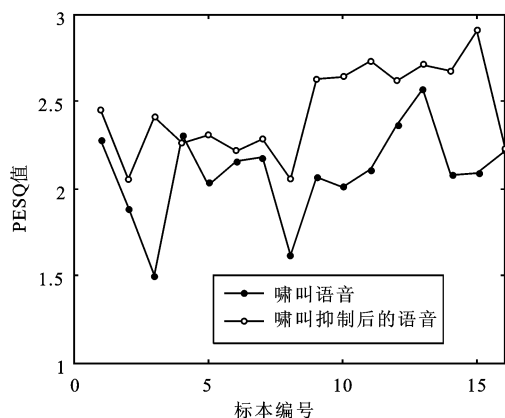


图9 啸叫语音和啸叫抑制后语音的 PESQ 值

参考文献:

- [1] Thomas D, Jayan A R. Automated suppression of howling noise using sinusoidal model based analysis/synthesis[C]// Advance Computing Conference (IACC), 2014 IEEE International. Miami, IEEE, 2014: 761-765.
- [2] 王建伟. 基于 SHARC 系列 DSP 的自适应声反馈抑制器的研制[D]. 杭州:浙江大学, 2014.
- [3] Liu L, Li H, Hu X. Digital filter algorithm based on whistle sound suppression system design[C]// Computer Science & Service System (CSSS), 2012 International Conference on. China, Nanjing, IEEE, 2012:

1559-1562.

- [4] Kia S H, Henao H, Capolino G. Efficient digital signal processing techniques for induction machines fault diagnosis[C]// Electrical Machines Design Control and Diagnosis (WEMDCD), 2013 IEEE Workshop on. France, Paris, IEEE, 2013: 232-246.
- [5] 尚海涛, 薛红喜. 基于带通滤波器的 ZoomFFT 算法研究[J]. 自动化仪表, 2011, 32(4): 10-13.
- [6] 肖启洋, 张忠慧, 方元. 基于 DSP 的声反馈抑制系统的研究与实现[J]. 电子技术应用, 2013, 39(1): 13-16.
- [7] 贾瑞, 李冬梅. 实时的 Gammatone 听感知滤波器组的 FPGA 实现[J]. 微电子学与计算机, 2015, 32(1): 35-39.

作者简介:

刘红玉 女, (1991-), 硕士研究生. 研究方向为多媒体信息处理. E-mail: lhy@whu.edu.cn.

易本顺 男, (1965-), 博士, 教授. 研究方向为多媒体信息处理、网络通信.

谢秋莹 女, (1991-), 硕士研究生. 研究方向为多媒体信息处理.

邱波 男, (1992-), 硕士研究生. 研究方向为多媒体信息处理.

(上接第 17 页)

栈主机 B 不在白名单策略文件中, 因此其与双协议栈主机 A 无法通信; 若将

```
$ 10.13.32.201@ICMP * 0
```

```
$ 2001:250:1004:5020:504c:5021:a02d:702a@ICMPV6 * 0
```

添加进策略文件 IP_Strategy.TXT, 则双协议栈主机 A 即可与双协议栈主机 B 进行连通.

通过上述简单测试可以看到, 该系统能根据白名单所列的地址等相关信息对双协议栈主机的访问行为进行管控, 实现预期功能, 相比命令行的操作形式具有一定的便利性.

6 结束语

本文针对 Linux 平台 IPv4/IPv6 双协议栈主机, 利用 Linux 内核提供的 Netfilter 框架结构, 采用内核编程方式设计并实现了主机网络管控系统. 该系统可最大限度确保 Linux 平台 IPv4/IPv6 双协议栈主机网络行为的可控性, 提高主机自身和局域网网络资源的安全性. 由于在网络栈上逐个检查数

据包势必会影响网络的性能, 下一步需要研究效率高更的匹配算法, 以及如何在此平台上针对应用程序和域名信息进行更细粒度和更高效的管控.

参考文献:

- [1] 余青霓. Linux 防火墙[M]. 北京: 人民邮电出版社, 2000.
- [2] 谢希仁. 计算机网络[M]. 北京: 电子工业出版社, 2008.
- [3] 姚晓宇, 赵晨. Linux 内核防火墙 Netfilter 实现与研究应用[J]. 计算机工程, 2003, 29(8): 112-115.
- [4] 王以伍, 任宇, 舒晖. 一种基于双栈的双效防火墙设计[J]. 电脑知识与技术, 2009, 33(15): 67-68.
- [5] 杨刚, 陈蜀宇. Linux 中基于 Netfilter/Iptables 的防火墙研究[J]. 计算机工程与设计, 2007, 28(17): 108-112.
- [6] 袁方方. 基于 Netfilter 的内容过滤系统的设计与实现[D]. 北京: 北京邮电大学, 2012.

作者简介:

贾锐 男, (1981-), 硕士研究生. 研究方向为网络信息安全. E-mail: 408164460@qq.com.

蔡皖东 男, (1955-), 教授, 博士生导师. 研究方向为网络信息安全.