

一种网络信息监管系统的设计与实现

蔡俊朝, 蔡皖东, 胡润东

(西北工业大学 计算机学院, 陕西 西安 710072)

摘 要: 基于 Linux 的 Netfilter 框架, 根据特定的策略实现对网络数据的捕获或者封堵. 系统主要由两部分组成: 一个可加载的内核模块和一个用户空间的应用程序. 分别采用 netlink 和内存映射技术实现核心态和用户态之间的命令传递和数据交互. 可以利用该系统实现基于内容的网络过滤、深度数据包检测、或者采用特定模式匹配算法的入侵检测系统. 最后, 以封堵 BitTorrent 流量为例对系统进行测试, 并取得了较好的实验结果.

关键词: 网络监管; 内容过滤; Netfilter; 可加载内核模块; 内存映射

中图分类号: TP393.09

文献标识码: A

文章编号: 1000-7180(2010)10-0140-05

Design and Implementation of a Network Information Monitoring System

CAI Jun-zhao, CAI Wan-dong, HU Run-dong

(College of Computer, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: A method based on the Netfilter framework of Linux system to capture or block the network traffic under rules is brought up in this paper. The system consists two parts: a loadable kernel module and a user-space application. Netlink socket is adopted to transmit orders between kernel-space and user-space, and memory mapping technology is used for the data exchange between both of them. This system can be used in network content-based filtering, DPI (deep packet inspection), or IDS (intrusion detection system) use a specific pattern matching algorithm. Finally, blocking BitTorrent traffic as an example to test the system is shown, and achieves good results.

Key words: network monitoring; content filtering; Netfilter; LKM; memory mapping

1 引言

由于互联网自身的开放性和匿名性等特点, 使得网络信息良莠不齐, 一些反动言论、暴力、色情等不良信息已经危害到国家的利益和社会的稳定, 因此对网络信息进行必要的监管也显得越来越重要. 网络内容安全成为继防火墙、网络防毒、入侵检测之后, 网络安全领域又一个重要的组成部分. 在进行网络信息监管时, 底层网络数据采集的效率和处理的速度是保证实时性的关键因素.

在 Linux 下常用的数据包捕获的实现方法是利用 Libpcap^[1]或 Socket 等. 它们都是采用系统调用

的方式来实现的. 而每一次的系统调用都会带来一次上下文的切换, 并且对于每一个到来的数据包, 都会重复一次系统调用和从内核空间到用户空间的内存拷贝, 这样所带来的系统开销在高速网络下是很难承受的. 另外, Libpcap 只能用于对网络数据进行捕获, 而对于实时检测到的非法网络连接, 并不能立即对其进行封堵.

针对 Libpcap 的不足, 文中基于 Linux 的 Netfilter 框架提出了一种数据包捕获和封堵的方法. 不但减少了系统调用, 而且通过采用内存映射技术来代替内核态与用户态之间的内存拷贝, 有效地降低了系统的开销. 此外, 文中还解决了用户态和内核态进

程间通信,以及基于策略实时地识别和封堵非法网络连接等关键问题.

2 网络信息监管系统框架

网络信息监管系统采用的是基于网关的实现方式.系统运行在一个拥有两块千兆网卡的 Linux 主机上.如图 1 所示,整个系统作为一个独立的控制设备,串接在内部网和 Internet 之间,所有进出局域网的数据都要流经网络信息监管系统.

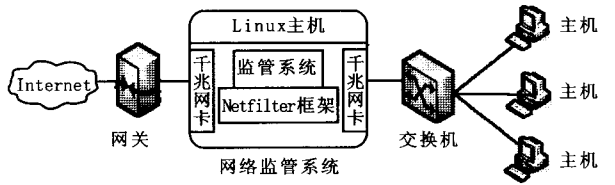


图 1 网络信息监管系统框架

网络信息监管系统所在的 Linux 主机此时的功能类似于一个静态路由,网络数据通过两块网卡进行转发.要想实现两块网卡间的数据转发功能,就需要对 Linux 主机进行一些必要的设置:首先,开启内核的数据包转发功能,可以通过命令 `echo "1" > /proc/sys/net/ipv4/ip_forward` 实现.要想让监管系统两边的局域网和外部网能够通信,还需要一些设置.接下来有两种方法:一是通过 iptables 的相关命令将该主机配置为 NAT 服务器,这样就可以实现内网 IP 和外网 IP 的转换了;另外一种解决方案是创建一个网桥,然后分别将两块网卡的网络连接添加到这个网桥,最后启动网桥即可.按上面的步骤设置完后,网络信息监管系统两边的两个网络之间就可以自由通信了,然后监管系统就可以基于 Netfilter 框架实现对流经该主机的网络数据进行基于策略的监控和封堵了.

系统由一个可装载的内核模块和一个用户空间的应用程序两部分组成.通过用户进程用户可以设置监控策略和封堵策略.所谓监控策略,即用户用来指定哪些类型的数据包会被捕获,被捕获的数据包将被保存到硬盘文件中.封堵策略是用来设定需要被封堵的数据包类型.两种策略格式相同,主要由网络协议类型(应用层协议如 HTTP、SMTP 等;传输层协议如 UDP 或 TCP)、网络地址(IP 地址或端口)以及数据包载荷中的关键字这三部分组成.内核模块利用 Netfilter 框架获取流经系统的网络数据包,然后根据用户进程所设置的监控策略和封堵策略对数据包进行相应处理.

3 Netfilter 框架

Netfilter 框架是一个抽象的通用化的包过滤框架,它是内核内一系列基于协议栈的钩子函数的集合.这些钩子函数会在数据包流经协议栈的几个关键点时被调用^[2].这些关键点称为钩子点,在钩子点处 IP 层与内核模块间切换对数据包的处理权. Netfilter 框架的钩子点分布及数据包通过过程如图 2 所示.表 1 中显示了数据包通过系统协议栈中各个钩子点的时机.

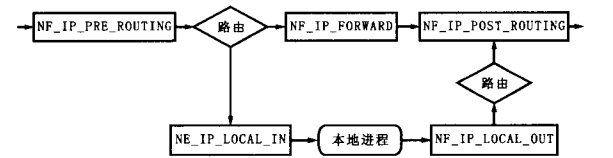


图 2 Netfilter 框架的钩子点分布

表 1 数据包通过每个钩子点的时机

钩子名称	数据包通过的时机
NF_IP_PRE_ROUTING	刚进入网络层的数据包,路由前
NF_IP_LOCAL_IN	路由查找后,送往本机的数据包
NF_IP_FORWARDING	路由查找后,要转发的数据包
NF_IP_LOCAL_OUT	本地进程产生的数据包,路由前
NF_IP_POST_ROUTING	即将通过网络设备发出的数据包

内核模块如果需要对数据包进行处理时,可以通过调用 Netfilter 的钩子注册函数将模块自己的包处理函数挂载到相应的钩子点上.当数据包流经某个钩子点时,系统会检查相应的钩子函数链表中是否有钩子函数被注册,如果存在则调用钩子函数.这样,内核模块就得到了数据包的处理权. Netfilter 最后会根据内核模块的包处理函数返回的结果对数据包做下一步的处理.内核模块的包处理函数需要向 Netfilter 返回的值及其对应的意义如表 2 所示.

表 2 内核模块包处理函数返回值及其含义

NF_DROP	丢弃该数据包
NF_ACCEPT	保留该数据包
NF_STOLEN	忘掉该数据包,即不再处理
NF_QUEUE	将该数据包插入到用户空间
NF_REPEAT	再次调用该钩子函数

4 网络监管系统设计

系统的框架设计如图 3 所示,程序主要分为用户管理进程和一个可加载的内核模块.两部分之间

的数据交换主要采用三种方式: netlink、/proc 文件系统和内存映射. Netlink 的使用方法非常类似于常用的网络套接字, 所以它非常便于用来在内核态和用户态之间传递一些命令消息. 系统中用户管理进程使用 netlink 给内核模块发送各种控制命令. /proc 文件系统主要是用来传递用户管理进程制定或修改后的监控和封堵策略, 需要重新下发策略时, 用户管理进程还会通过 netlink 发送一个消息通知内核模块更新相应的策略. 与监控策略匹配成功的数据包会被内核模块保存到内核缓存中, 用户进程通过内存映射的方式来访问内核缓存, 然后负责将捕获的数据包写入到硬盘文件. 因为监控捕获包的数据量比较大, 所以这里相对于 netlink 和 /proc 文件系统这两种方式来讲, 采用内存映射的方式是最高效的.

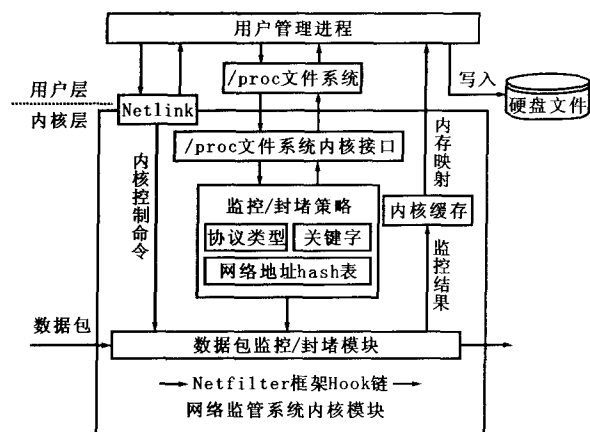


图3 系统结构图

总的来讲, 用户管理进程的主要任务有: (1) 管理内核模块: 通过 netlink 给内核模块发送各种控制命令, 如激活内核模块、停止内核模块、命令内核模块断开网络连接或者通知内核模块更新监控/封堵策略等. (2) 管理监控/封堵策略: 用户管理程序采用 Qt/KDE 开发, 用户可以利用窗口界面直接定制或修改两种策略, 然后用户管理进程负责根据用户的设置生成相应的策略文件. 如果用户设置的策略中包含多个网络地址(IP 地址和端口号), 用户进程会将它们生成一张 Hash 表, 目的是提高内核模块匹配策略时候的查找效率. (3) 将监控结果写入硬盘: 内核模块将与监控规则匹配的数据包写入到内核缓存中, 然后用户进程通过内存映射的方式访问内核缓存, 并将监控捕获的数据包写入到硬盘文件.

内核模块将数据包处理函数挂载到 Netfilter 框架的 NF_IP_PRE_ROUTING 钩子点上, 即所有

进入网络层的数据包, 不管是发往本机的还是需要通过本机转发的, Netfilter 都会在其到达时交由内核模块的包处理函数进行处理. 如果用户进程将内核模块置于激活状态, 则内核模块会将数据包与此时所设置的监控和封堵策略进行匹配. 若与监控策略匹配成功, 则内核数据包处理函数会将其保存到内核缓存中, 然后给 Netfilter 框架返回值 NF_ACCEPT, Netfilter 会继续转发该数据包, 用户进程通过对内核缓存进行地址映射, 直接从中读取捕获的数据包并写入到硬盘文件供用户查看, 避免了内核态向用户空间的内存拷贝. 若数据包与封堵策略匹配成功, 则内核数据包处理函数给 Netfilter 框架返回值 NF_DROP, 则 Netfilter 框架会将该数据包丢弃不再转发, 从而达到封堵的目的.

5 网络监管系统实现

5.1 内存映射技术

Linux 的内存空间被分为内核空间和用户空间两部分. 内核进程和用户进程分别存放和运行在内核空间和用户空间, 它们不能访问彼此的地址空间. 要将内核模块捕获的数据包提交给用户进程进行分析和保存, 需要利用内存映射技术建立用户态和核心态之间的接口. 这样做的目的是为了减少系统内核向用户的内存拷贝, 降低系统开销. 下面介绍下内核映射的具体实现过程^[3]. 在下面的例子中内核模块往缓存中写入一个字符串, 然后由用户空间进程读取.

(1) 在内核模块中分配缓冲区

在 Linux 中, 每个内存页的大小是 4K, 使用函数 `_get_free_page(GFP_KERNEL)` 来分配一个内存页. 另外, 还可以使用函数 `_get_free_pages(GFP_KERNEL, Order)` 来一次分配多个连续的内存页, 参数 Order 是 2 的指数. 内核支持的最大可分配连续缓存是 2M, 也就是 Order 最大为 9. 要想分配更大的内核缓存, 就必须多次分配非连续的缓存, 然后用数组或链表来对这些缓存进行管理.

```
static unsigned long saddr; //用来保存分配空间的首地址
```

```
saddr = _get_free_pages(GFP_KERNEL, 9); //分配 2M 内核缓存
```

```
mem_map_reverse(virt_to_page((char *) saddr)); //常驻内存
```

```
strcpy(((char *) saddr), "this is data from kernel"); //向内核缓存写入数据
```

(2) 将内核缓存首地址传递给用户进程

不能直接将 `saddr` 传递给用户进程,因为内核申请的缓存地址 `saddr` 是虚拟地址,而用户进程需要得到的是物理地址,所以传递 `saddr` 前必须将其转换成物理地址,具体方法是使用函数 `char * addr = (char *)_pa(saddr)`,然后传递 `addr`. 地址转换 `_pa(saddr)`,其实是 `saddr - PAGE_OFFSET, PAGE_OFFSET` 等于 3G,即将虚拟地址减去 3G 就可以得到对应的物理地址.

将内核缓存首地址传递给用户进程,常用的方法有采用 `/proc` 文件系统,或者使用 `netlink` 套接字,这里采用后者. 内核模块把捕获的数据包写入到内核缓存,当缓存写满后,内核模块分配新的缓存,并通过 `netlink` 发送消息给用户进程通知其进行处理,这时缓存首地址作为消息的一部分发送给用户进程,用户进程处理完后再发送 `netlink` 消息通知内核模块释放该内核缓存. `netlink` 的实现方法放在下一小节进行介绍.

(3) 用户态程序访问共享内存

用户进程通过 `netlink` 套接字接收到内核缓存的物理地址 `addr` 后,就可把其映射到用户空间了,具体方法是:在用户空间打开 `/dev/mem` 设备,它是一种特殊的文件,通过它可以访问整个系统的物理内存,而物理内存的地址就是读写该文件时的偏移量.最后使用 `mmap` 把内核态下分配的缓存映射到自己的进程空间即可.具体实现如下:

```
fd = open("dev/mem", O_RDWR);
```

```
char * uaddr = mmap(NULL, len, PROT_READ|PROT_WRITE, MAP_SHARED, fd, addr);
```

其中参数 `len` 用来指定要被映射的缓存大小.这样就把内核分配的缓存映射到指针 `uaddr` 指向的缓存,使用 `printf("%s\n", uaddr)` 输出,就可以看到在内核模块写入的消息 "this is data form kernel" 了.

5.2 核心态和用户态进程间通信

5.2.1 netlink 套接字通信的实现

`netlink` 是一种很好的内核模块与用户空间进程之间进行双向数据传递的方式,用户态中使用 `netlink` 就像使用普通的 `socket API` 一样,不同的是套接字所绑定的地址不再是 IP 地址和端口号这样的网络地址,这里使用的是其进程号,而内核模块则需要专门的 `netlink API`. `netlink` 通信的实现过程如图 4 所示^[4].

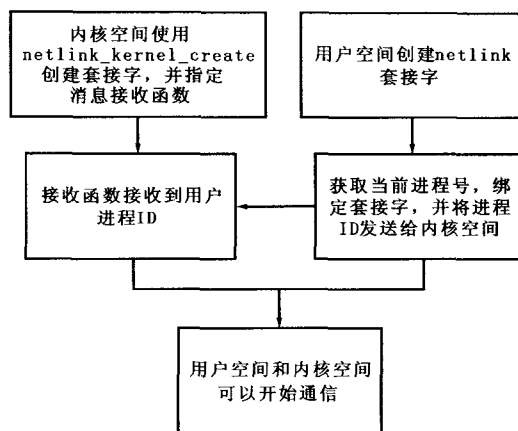


图 4 netlink 套接字通信实现过程

5.2.2 使用 `/proc` 文件系统传递监控策略

`/proc` 文件系统是 Linux 的一个虚拟文件系统,系统使用它来传递监控和封堵策略.它的具体实现过程如下:

(1) 创建 `/proc` 虚拟文件

内核模块使用函数 `create_proc_entry` 创建一个 `/proc` 虚拟文件 `proc_dir_entry * proc_entry = create_proc_entry("FilterRule", 0644, NULL)`;第一个参数是虚拟文件的名字.函数返回的是一个 `proc_dir_entry` 指针变量,可以通过它来设置这个虚拟文件的一些参数.结构体 `proc_dir_entry` 最重要的两个成员变量是 `read_proc_t * read_proc` 和 `write_proc_t * write_proc`,这是两个函数指针,分别用来实现从用户空间读取和写入该虚拟文件的方法.当卸载内核模块时,可以使用函数 `remove_proc_entry` 来释放所创建的 `/proc` 虚拟文件.

(2) 实现从用户空间读写该虚拟文件的方法

从用户空间读取文件的方法 `read_proc` 容易实现,函数 `read_proc` 的第一个参数 `char * page` 表示内核空间的一段缓冲区,内核模块只需要把要给用户空间进程读取的数据写入到这段缓存即可. `write_proc` 提供的是用户空间进程写该虚拟文件的方法,该函数的第二个参数 `const char * user * buff` 表示的是用户空间要写入数据的缓存地址,但是不能直接把它的数据拷贝到内核缓存中,因为 `buff` 是在用户地址空间而不是内核地址空间,从 `buff` 中拷贝数据要使用函数 `copy_from_user` 函数来完成.

(3) 用户空间进程操作 `/proc` 文件

用户空间对于 `/proc` 文件的操作就和使用普通的文件一样.例如打开文件的操作: `fd = open("/proc/FilterRule", O_RDWR)`.

5.3 注册 Netfilter 钩子函数

加载内核模块时,系统会调用模块的 `module_init()` 函数来初始化内核模块.卸载模块时,内核模块的 `module_exit()` 函数会被系统调用.在 `module_init()` 函数中调用 `int nf_register_hook(struct nf_hook_ops * req)` 函数来注册 Netfilter 钩子函数.在卸载模块时调用的 `module_exit()` 函数中使用函数 `void nf_unregister_hook(struct nf_hook_ops * req)` 函数来注销挂载的钩子函数.结构体 `nf_hook_ops` 的成员变量 `nf_hookfn * hook` 是一个函数指针,用来指定要挂载的钩子函数.本系统在该钩子函数中实现数据包规则匹配检查和相应处理.另外,可以利用 `nf_hook_ops` 结构体的成员变量 `int hooknum` 和 `int priority` 来分别指定要挂载的钩子点和钩子函数的优先级^[5].

分别使用系统命令 `insmod` 和 `rmmod` 来加载和卸载所生成的内核模块,或者通过在 `/etc/modules.conf` 文件中添加相应的条目来实现内核模块随系统开机自动加载.

6 实验结果

以封堵网络内的 BitTorrent 流量为例,来对系统进行测试.实验环境为局域网,网络信息监管系统运行在局域网的出口,如图 1 所示.根据 BT 的协议特征,Peer 节点之间建立连接时握手消息的 TCP 载荷的第一个字节是 19(0x13),接下来的 19 个字节是字符串“BitTorrent Protocol”.所以,网络监控系统以这个特征为关键字添加到封堵策略中,对 Peer 节点之间的握手数据包进行封堵,Peer 节点之间无法正常握手建立连接,也就无法进行文件传输了.图 5 是对 BT 数据流封堵前后整个局域网流出的数据量对比.另外,测试过程中,除 BT 下载无法正常进行外,其他应用访问网络的行为并未受到明显的影响,系统运行正常.

络信息监管系统,可以根据所设定的策略对局域网内用户的上网行为进行监管,实现方便可行,具有实用价值.另外,还可以在本系统的基础上做进一步的应用开发,如对于网络内容进行监管时不再仅仅基于关键字,而是采用 SVM 或者潜在语义索引法等来提高内容过滤的查全率和查准率.

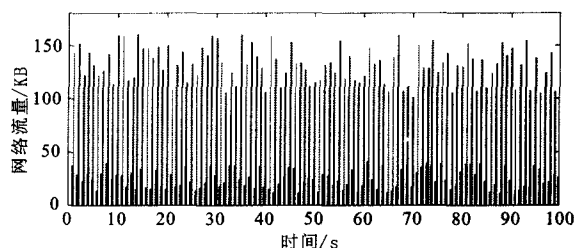


图 5 对 BT 数据流封堵前后流出网络数据量对比

参考文献:

- [1] Gerry Zaugg. A lightweight packet capturer for high-speed links[EB/OL]. [2009-11-15]. <ftp://ftp.tik.ee.ethz.ch/pub/students/2003-2004-WiSA-2004-08.pdf>.
- [2] 博嘉科技. Linux 防火墙技术探秘[M]. 北京:国防工业出版社,2002.
- [3] 梁健. 零拷贝技术研究 with 实现[EB/OL]. [2009-11-12]. <http://www.xfocus.net/articles/200311/642.html>.
- [4] 章小明,杜春燕. Netlink 消息通信机制的研究和在 IPsec VPN 中的应用[J]. 微电子学与计算机,2007,24(2):135-138.
- [5] 李健,王玲. 基于 Linux 的网络入侵检测与防火墙集成系统的设计与实现[J]. 微电子学与计算机,2006,23(4):18-22.

作者简介:

蔡俊朝 男,(1986-),硕士研究生.研究方向为网络与信息安全.

蔡皖东 男,(1955-),教授,博士生导师.研究方向为网络与信息安全.

7 结束语

文中基于 Linux 的 Netfilter 框架提出了一种网