文章编号: 1007-5321(2006)02-0118-05

支持 IDS 的高速网络信息获取体系结构

张兆心[1], 方滨兴[2], 胡铭曾[1]
(1. 哈尔滨工业大学 计算机网络与信息安全技术研究中心, 哈尔滨 150001;
2. 国家计算机网络与信息安全管理中心, 北京 100029)

摘要: 提出了一种针对高速网络环境的信息获取体系结构. 可扩展的网络探测模型较好地解决了不同网络带宽的适应性问题; 高带宽数据流实时捕获技术、高效的多线程 TCP/IP(传输控制协议/互联网络协议)协议栈, 以及基于插件的 PLUGINs 协议还原平台, 使信息的捕获与还原问题得到了根本性解决. 综合以上技术所构建的体系结构可以应用于多线路、高速网络环境下, 解决检测范围大、数据流量高等问题, 保证数据信息的及时性、安全性与准确性.

关 键 词: 零拷贝; 并行协议栈; 协议还原
中图分类号: TP393 文献标识码: A

An IDS-Supported Architecture of Information Capture for High-Speed Networks

ZHANG Zhao-xin[1], FANG Bin-xing[2], HU Ming-zeng[1]
(1. Research Center of Computer Network and Information Security Technology, Harbin Institute of Technology, Harbin 150001, China;
2. National Computer Network and Information Security Administrative Center, Beijing 100029, China)

Abstract: The architecture of information capture for the high-speed network environment was proposed. The different network bandwidth adaptive problem was well solved by the extensible network detection model. It has been found that real-time data stream capture from high-bandwidth networks, high-performance multi-thread TCP/IP (transmission control protocol/Internet protocol) protocol stack, and protocol analysis platform based on PLUGINs have captured and assembled the information wonderfully. The architecture integrated these technologies effectively will solve the problems such as the large-scale detection, high-speed data stream, ensuring the in-time, security and veracity of the information on multi-road and high-speed networks.

Key words: zero-copy; multi-thread transmission control protocol/Internet protocol stack; protocol analysis

0 引 言

近些年互联网在国际范围内得到了突飞猛进的发展, 已成为人们日常生活中不可或缺的一部分. 互联网在对社会、经济、文化和科技带来巨大推动和冲击的同时, 其自身的安全问题也变得日益突出. 互联网自身的安全, 已经开始制约互联网的进一步广泛应用, 因此, 如何保障互联网的安全运行, 已成为关注的焦点.

保障网络安全的 PDRR(protect detect react re-

收稿日期: 2005-02-25
基金项目: 863-917 专项; 2004 研 4-A A-01
作者简介: 张兆心(1979—), 男, 博士生, E-mail: heart@hit.edu.cn.

store)防范模型, 从防护、检测、响应、恢复这几个角度全面保障互联网的安全. 其中, 检测技术所依赖的核心技术就是互联网信息的获取. 通过对网络信息的获取, 进一步判定是否出现了网络入侵事件. 换句话说, 互联网的信息获取是 IDS(入侵检测系统)[1,2] 的重要环节. 尤其是目前网络带宽越来越高, 解决高速网络环境下的信息获取变得十分重要. 高速网络定义为骨干网络上有 1 Gbit/s 以上流量的网络, 在这样的网络环境下的信息获取面临如下的难题:

1) 网络数据处理瓶颈. 随着百兆以太网、ATM、千兆以太网广泛应用, 现在主机对网络数据包的处理逐渐成为一个瓶颈, 往往需要机群来捕获处理大量的网络数据.

2) 要获取的信息种类层出不穷. 由于网络安全问题的类型很多, 所面对的网络信息类型也很多, 从而使得为网络安全监控而需要的网络信息的获取变得复杂.

3) 各种协议层出不穷, 导致系统变动频繁. 网络信息获取系统需要面对不同的协议给出不同的分析结果, 如何解决协议处理的通用性成为了挑战性问题.

通过以上分析, 本文提出了一种在高速网络环境下的信息获取体系结构. 该结构采用可扩展的网络探测模型、高带宽数据流实时捕获[3-5]、高效的多线程 TCP/IP 协议栈、基于可扩展的 PLUGINs 的协议还原、多模匹配算法[6] 等关键技术, 有效地解决了高速网络环境下信息捕获面临的难题. 为检验该体系结构的功能, 设计了针对该体系结构的实验系统, 并进行了测试.

1　信息获取体系结构

1.1　系统结构

信息获取体系结构如图 1 所示. 该体系结构由 1 组(n 个)原始信号耦合器(C_i)、汇总/分流子系统(GD)、1 组(m 个)处理子系统(P_j)、系统预警中心(SWC)和汇总/处理子系统(GP)组成. 首先利用原始 C_i 复制数据信号, 传给 GD; 然后由 GD 将数据流(DS_k)按照一定的策略分配给 P_j; 接着 P_j 对分到的数据流进行捕包、还原、分析等操作, 并将处理后的信息传给 GP; 最后由 GP 将最终的结果汇总和分析. SWC 实时监视系统的运行情况.

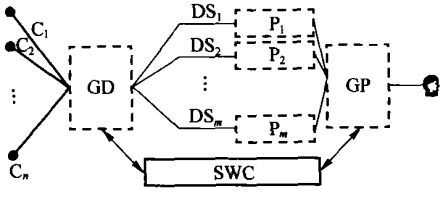

图 1　信息获取体系结构

1.2　主要子系统简介

1.2.1　原始信号耦合器

原始信号耦合器是指网络原始数据输出机构, 如 HUB 的监听口、交换机的镜像口等. 通过这些端口将线路上的数据信号完整地复制出来, 送到下一级进行初级处理. 原始信号的耦合技术根据不同的监听机制来设计. 考虑到在高速网络环境下进行信息获取, 所以采用光耦合监听. 采用光耦合器件可以将光信号复制出 1 份或多份, 因其是无源设备, 故而运行稳定, 只是双向数据需要分别输出. 这种方式适用于 POS、GE、10/100BASE-FX 等光接口.

1.2.2　汇总/分流子系统

汇总/分流子系统对接入的数据流进行均衡的划分, 这是实现宽带网络数据高性能处理的关键. 该系统利用"异或"操作,

$$XOR(Sip, Dip, Sport, Dport) \bmod n$$

(n 为集群系统的节点数)将数据流的各个连接均匀地分配到集群系统的各个节点机上, 保证了经过同一过滤点的同一连接的 2 个方向数据都能进入同一节点机中.

1.2.3　处理子系统

为解决在不同网络带宽下的适应性问题, 每个处理器子系统采用基于单控制流多数据流(SPMD, single program multiple data)计算模式的集群式体系结构, 通过对集群系统进行可伸缩性设计, 达到仅通过增加处理节点而不修改程序即可成倍扩展处理器子系统处理能力的目的. 其结构如图 2 所示. 图中的插件为各种应用层服务, 目前已实现 HTTP、TELNET、FTP、SMTP、POP3、FREENET、IMAP、

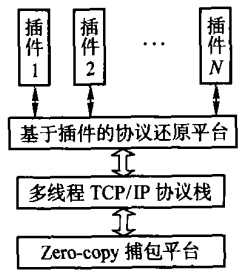

图 2　处理子系统结构

FREEGATE 和 TRIBOY，可以根据实际需求进行添加或删除，不会影响其他插件的正常工作.

2 关键技术

2.1 Zero-copy 捕包平台

目前，在 Unix/Linux 操作系统下的网络监听系统普遍建立在 Libpcap 捕包平台上，这是一个可用于捕获经过网络接口数据包的 C 函数接口系统库. 由于 Libpcap 在捕包过程中经过多次函数调用和内核拷贝，以及频繁的中断处理和进程切换，导致捕包效率不高.

Zero-copy 是指在某节点的报文收发过程中不会出现任何内存间的拷贝，即在发送和接收过程中，数据包直接在应用程序的用户缓冲区和网络接口间传递，避免了中间的数据拷贝过程，故称为“真正的零拷贝”. 要实现真正的零拷贝需解决用户和内核共享内存、地址翻译[7,8]、保护机制和控制传递方式4个问题.

首先，利用 Linux 操作系统提供的一种内存映射[9]机制，即将某一块内核空间映射到虚地址空间，用该空间内的虚地址访问相应的物理页面，实现用户和内核共享内存.

其次，采用内核和网卡协作. 在内核创建一个虚拟设备模块(Dum)用于存储用户空间缓冲区块的索引，并维护一个虚地址到物理地址的地址表. 上层用户和网卡通过访问这个虚拟设备得到自己需要的目的地址，解决地址翻译问题.

再次，采用允许多个用户访问网络接口和数据包的缓冲区的方式，但只允许一个进程对网络接口设置的保护机制.

最后，采用中断与轮询的综合处理方式，给中断加一个定时器，修改网卡驱动程序，每隔一定时间发一次中断，每次同时处理多个数据包.

通常情况下，内核接收网络数据包也存在一定的浪费. 对以太网而言，当一个数据包到达网卡时，向内核发一个收包中断(硬中断)，内核调用中断处理程序，将该数据包拷贝到内核的 SK_Buffer 缓冲区，并剥去以太包头，然后标志 NET_BH 标志位，返回，等待中断的下半部分处理. 此过程约耗费20%的CPU. 当CPU空闲时，调用中断的下半部分处理程序，将 SK_Buffer 中的 IP 报文投入内核协议栈，进行还原. 该过程发生多次系统调用与内存拷贝，耗费约20%的CPU. 最后内核将还原的数据由内存的内核空间拷贝到用户空间，提供给用户处理使用，该过程约需要50%的CPU.

综合以上的分析设计出基于 Zero-copy 思想的捕包平台，如图3所示. 在用户层(User)，分配1个大的用户缓冲区(UB)，存放接收和发送的数据包，并被分成每块大小为2 KB* Block，每小块存放1个数据包. 在内核区(Kernel)创建1个 Dum，作为用户程序与网络接口进行交互的中介，并成为上层应用操作网络的句柄. Dum 为用户缓冲区的每一小块建立1个描述符 Item，并将每个描述符挂入 Dum 不同的队列里. 共建立了发送队列 SendQ、接收队列 ReceiveQ、空闲队列 FreeQ 3个环形队列. 待发送的数据包的描述符挂入 SendQ，接收到的数据包的描述符挂入 ReceiveQ，没有数据的块的描述符挂入 FreeQ. 同时，Dum 设备还维护1张虚地址——物理地址对应表(PAT). 用户缓冲区有多少虚地址页，这张表就有多少项，每一项存放该虚地址页对应的物理页面的首地址. Dum 设备本身在内核中的缓冲区空间被映射到用户区，和 UserBuffer 一起成为一块由内核和用户共享的区域. 这块区域包括 UserBuffer、3个环形队列和物理地址表. 这样网卡和用户都可以自由地访问这块共享区域的数据. 修改网卡的驱动程序，将网卡设备的私有域填上 Dum 设备的信息，网卡进行直接存储器存取(DMA)传输时，直接根据 Dum 设备中的描述符找到真正的位于用户区的缓冲区，然后直接将数据包从网卡传到用户区，数据包不再进入内核的协议栈.

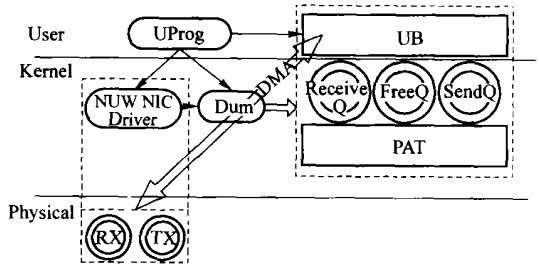

图3 基于 Zero-copy 的捕包平台

在相同的实验环境且每个包大小为64 B时，Libpcap 与 Zero-copy 捕包效果的比较如图4所示，

* 因为目前路由器的 MTU 为1 500 B，加上以太包头的14 B，所以传送到用户空间的包字节数不会超过1 514 B，这样2 KB大小的块刚好能满足包大小的要求. 又因为在 Linux 2.4 内核下，内存每一页面的大小为4 KB，这样 Block 的大小刚好为页面大小的一半，便于后面进行页操作.

从图中可以清楚地看出，Zero-copy 的捕包能力远优于 Libpcap.

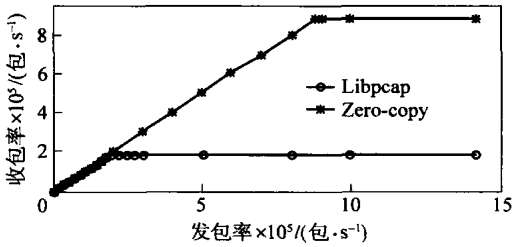

图 4　Libpcap 与 Zero-copy 捕包性能比较

2.2　多线程 TCP/IP 协议栈

与传统的 TCP/IP 协议栈不同，用于网络信息获取的 TCP/IP 协议栈不需要输出数据，而只需对输入的数据进行处理. 但这种协议栈却需要同时处理网络链路 2 个方向上的数据，即从用户层输入，通过协议栈从链路层输出到通信线路上，与从通信线路上输入通过协议栈传递到用户层. 而用于网络信息获取的 TCP/IP 协议栈的 2 个方向的数据都是从通信线路上输入，通过协议栈的层层还原，还原出应用层的通信原始数据流，再提交给具体的应用分析插件进行分析.

通过以上分析可知，传统协议栈的单线程机制并不能充分发挥通用的共享存储器多处理器 SMP* 体系结构的高性能服务器的计算能力，因此采用多线程共享内存的多处理器平台(SMP)[10] 完成连接级别的 TCP/IP 协议栈的并行处理方法. 多线程连接级并行的 TCP/IP 协议栈本质上是同时运行多个逻辑上独立的 TCP/IP 协议栈，协议栈结构框图如图 5 所示.

并行协议栈通过 1 个数据分发器将捕包系统捕获的网络数据包按照 IP 包头的源地址和目的地址对分发到相应的线程进行处理，并通过 1 个空闲单元收集器完成空闲单元的释放. 注意，这里的空闲单元释放并未将相应的数据块存储单元空间释放，而是将存储单元退还给捕包系统，让捕包系统重复利用这些单元，实现捕包到分析的 Zero-copy.

每个 TCP/IP 协议栈线程均有 1 个私有的协议栈状态表和 2 个数据队列索引，其中协议栈状态表是协议栈在进行 IP 协议和 TCP 协议还原时用来保存上下文信息和暂存数据；而 2 个数据队列索引则分别用来存储待分析的数据块和已分析块. 这样，任何一协议栈线程在进行数据操作时都不会和其他协议栈线程共享数据块，避免协议栈间的临界锁.

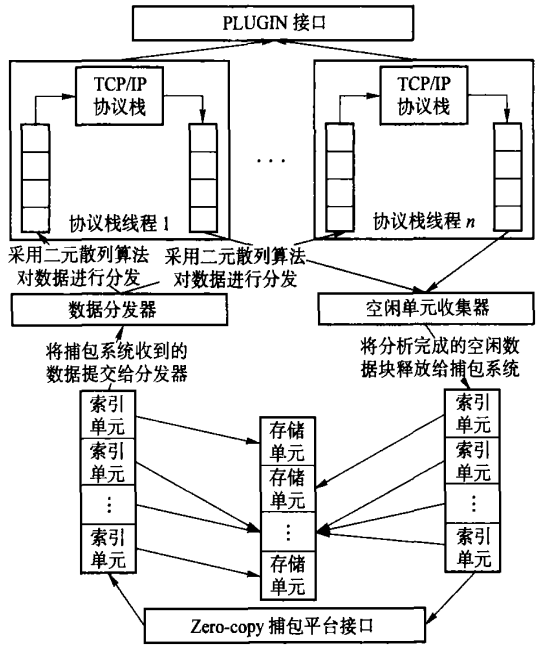

图 5　多线程 TCP/IP 协议栈结构框图

3　系统测试

为验证该信息获取体系结构的工作能力，设计了如图 6 所示的实验环境，用于测试系统的整体性能. 测试中所有机器的硬件配置为内存 4 GB，CPU

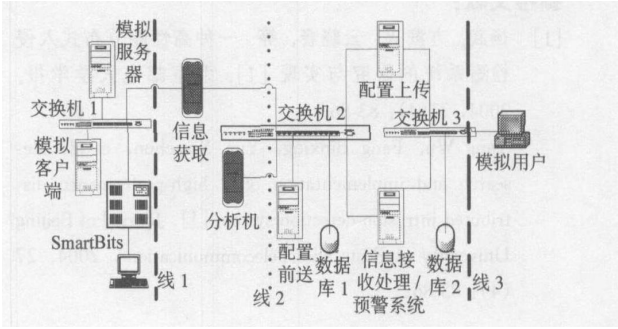

图 6　测试环境

为 2.0 GB，硬盘为 18 GB，Intel(R)PRO/1 000；软件配置 Linux 机器为 Red Hat 2.4.18-5smp，Windows 机器为 Windows2000. 在 SmartBits 的背景流量下，通过客户端对服务器访问. 在此实验环境下，进行网页地址、邮件地址、网页内容、邮件内容和 IP 地址监视的测试，实验结果如图 7 所示. 从图中可见，信息获取的效果非常理想，但这与试验环境有相当大的关系. 首先，SmartBits 发的每个包都一样；其次，

* 该架构有较高的灵活性，可以通过适当增加处理器的数量适应光纤网络传输能力的提高.

在局域网中测试,不存在网络延迟大、堵塞等网络问题.

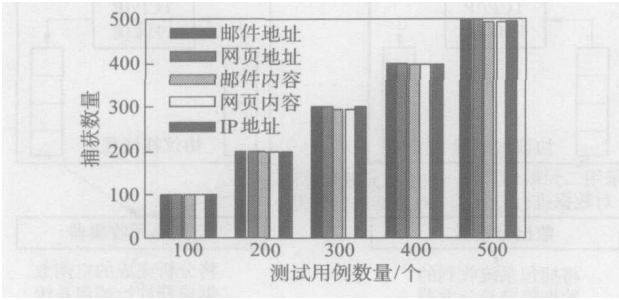

图7 信息监测测试

4 结 论

本文所提系统采用的可扩展的探测模型能适应各种网络环境;高带宽数据流实时捕获技术在SmartBits作背景流量,每个包为64 B时捕包率达到9×10^5包/s,尤其当包大小为128 B时捕包达到了线性;高带宽复杂网络环境下数据流协议还原技术大大提高了协议还原的能力,当有4个线程以上的协议栈时,其吞吐量可达到2 Gbit/s. 采用上述技术所实现的高速网络环境下的信息获取体系结构,很好地解决了目前高速网络信息获取中的问题.

参考文献:

[1] 杨武, 方滨兴, 云晓春, 等. 一种高性能分布式入侵检测系统的研究与实现[J]. 北京邮电大学学报, 2004, 27(4): 83-86.
Yang Wu, Fang Binxing, Yun Xiaochun, et al. Research and implementation of a high-performance distributed intrusion detection system[J]. Journal of Beijing University of Posts and Telecommunications, 2004, 27(4): 83-86.

[2] 柴平, 龚向阳, 程时端. 分布式入侵检测技术的研究[J]. 北京邮电大学学报, 2002, 25(2): 68-73.
Chai Pingxuan, Gong Xiangyang, Cheng Shiduan. Research on distributed intrusion detection[J]. Journal of Beijing University of Posts and Telecommunications, 2002, 25(2): 68-73.

[3] Kurmann A, Rauch F, Stricker T. Speculative defragmentation-leading gigabit Ethernet to True zero-copy communication[J]. Cluster Computing, 2001, 4(1): 7-18.

[4] Rubini A, Corbet J. LINUX device drivers[M]. Sebastopol: O'Reilly & Associates, 2002.

[5] 胡希明, 毛德操. LINUX内核源代码情景分析[M]. 杭州: 浙江大学出版社, 2001.
Hu Ximing, Mao Decao. Analysis the nuclear code of LINUX[M]. Hangzhou: Zhejiang University Press, 2001.

[6] Charras C, Lecroq T T. Exact string matching algorithms[M]. London: King's College London Publications, 2004.

[7] Welsh M, Basu A, Eicken T. Incorporating memory management into user-level network interfaces[D]. NY: Cornell University Ithaca, 1997.

[8] Jacob B, Mudge T. Software-managed address translation[C]//In Proceedings of the 3rd International Symposium on High Performance Computer Architecture. Texas: San Antonio, 1997: 156-157.

[9] Denning P J. Virtual memory[J]. ACM Computing Surveys (CSUR), 1970, 2(3): 153-189.

[10] Bjorkman M, Gunningberg P. Performance modeling of multiprocessor implementations of protocols[J]. IEEE/ACM Transactions on Networking, 1998, 6(3): 262-273.