基于Linux高速报文捕获平台的DDoS入侵检测系统的研究

黎忠文[1] 吴成宾[2] 许晓晨[3]
(成都大学信息科学与技术学院 成都 610106)[1] (成都大学现代教育技术中心 成都 610106)[2]
(厦门大学计算机系 厦门 361005)[3]

摘 要 如何在高速网络环境下实现线速的报文捕获以及上层的安全应用,一直是研究的热点。前期用内存映射和零拷贝等方法实现了基于千兆网卡的高速报文捕获平台NACP,在此基础上,通过使用IP地址的分布与系统资源的使用情况等作为检测参数,在snort工具上实现了防DDoS攻击的入侵检测系统。在NACP上的实验表明,改进的DDoS入侵检测工具snort与高速报文捕获平台兼容性良好,发生DDoS时能迅速检测到并且做出恰当的回应。由于使用了高速报文捕获平台,DDoS检测占用系统资源明显减少,很大程度上提高了系统的效率,系统可以在入侵检测的同时处理其他的事务。

关键词 千兆,高速报文捕获,NAPI,分布式拒绝服务攻击,入侵检测系统
中图法分类号 TP393 **文献标识码** A

Research on DDoS Intrusion Detection System Based on Linux High Speed Packet Capturing Platform

LI Zhong-wen[1] WU Cheng-bin[2] XU Xiao-chen[3]
(School of Information Science and Technology, Chengdu University, Chengdu 610106, China)[1]
(Modern Educational Technology Center, Chengdu University, Chengdu 610106, China)[2]
(Computer Department, Xiamen University, Xiamen 361005, China)[3]

Abstract It has always been a hot research aspect to achieve wire-speed packet capturing and the upper security applications in gigabit network environment. In previous work, we created the high-speed Gigabit Ethernet packet capture platform NACP by using memory mapping and other methods. On this basis, by using the distribution of IP addresses and the use of system resources as detection parameters, we achieved the anti-DDoS attacks intrusion detection system based on snort tool. The experiments on NACP show that improved DDoS intrusion detection tool Snort is compatible with the high-speed packet capturing platform, and the event of DDoS can be quickly detected and get appropriate response in NACP. Because of the use of high-speed packet capturing platform, the system resources occupied by DDoS detection are significantly reduced, so it greatly improves the system efficiency, and the system can handle other affairs during the intrusion detection.

Keywords Gigabit, High-speed packet capture, NAPI, Distributed denial of service attacks, Intrusion detection system

1 引言

百兆环境下使用的Linux系统自带Libpcap[1]报文捕获函数库已经不能继续满足人们的需求,因为其"每包驱动"的特性在千兆级以上的高速环境下将产生大量中断,造成中断淹没和丢包率急剧上升,引起系统瓶颈,若系统不改进,则会影响其他网络应用(如入侵检测系统、网络协议分析、防火墙)的效率,成为高速网络普及的阻碍。因此如何在高速网络环境下实现线速的报文捕获以及上层的安全应用,一直是研究的热点。前期针对线速报文捕获问题,采用Linux环境,用零拷贝和内存映射[2]等方法设计和实现了一个高速报文捕获平台NACP,它具有线速报文捕获、CPU占用率低和与Libpcap兼容等特点。该平台使用零拷贝和轮询机制结合的方式省去了数据包在内核空间传递过程中的拷贝、缓存以及系统调用过程,从而使得网络报文的处理性能有很大的提升;另外,通过在数据包中加入标志位,解决了高速数据包接收过程中由于内核空间和用户进程空间同时对数据包进行读取而带来的同步问题。实验表明,平台在主流PC机上实现且数据包尺寸小于等于256字节时,CPU使用率及数据包接收速率较Libpcap都有很大的提高;而当数据包大于256字节时,接收速率达到网卡线速,且CPU使用率也很低(小于10%),使系统有足够的时间处理别的任务,例如对数据包进行深度检测。

在此基础上,设计了一个基于网络特征的入侵检测算法,在分析了系统遭受DDoS(Distributed Denial of Service,分布

到稿日期:2013-06-04 返修日期:2013-08-09 本文受国家自然科学基金(60903160),中央高校基金(11D11209),四川省科技支撑计划基金项目(2013GZ0016),四川省教育厅重点项目(13ZA0296)资助。
黎忠文(1970—),女,博士,教授,主要研究方向为计算机网络、物联网等,E-mail:lizwxmu@gmail.com;吴成宾(1971—),男,硕士,高级工程师,主要研究方向为计算机网络、中间件等,E-mail:wpacer@cdu.edu.cn(通信作者);许晓晨(1987—),女,硕士,主要研究方向为计算机网络安全。

式拒绝服务)攻击时的特征后,选取了 IP 地址的分布和系统资源(包括内存和 CPU)使用率作为参数来判断系统是否遭受 DDoS。在 snort 工具上实现了该算法,并将其移植到上述高速报文捕获平台。在系统资源使用方面,由于 snort 原先的数据包捕获是通过 Libpcap 平台,因此也存在高速报文捕获时的系统瓶颈问题,移植到新平台上后,可以有效减少在进行数据捕获时所占用的系统资源;另外,由于采取了两项参数来进行入侵检测的判断,因此提高了检测的准确性和效率,并能及时对入侵作出反应。

2 高速报文捕获平台

高速报文捕获平台 NACP 由两部分组成,一部分是位于内核空间的改进的网卡驱动模块,它通过在内核、用户空间对同一段物理地址建立映射关系,使得两个空间可以共享这段内存,达到了消除内存拷贝所产生的额外开销之目的,换句话说,它提供了一个从网卡到用户空间应用程序的直接通道;另一部分是用户空间的兼容 Libpcap 的接口,上层零拷贝应用程序只需简单地调用此接口就可对封包进行处理。为了提高报文处理效率,平台采用了 NAPI 机制[3-6],NAPI 的特点在于:在一个繁忙的网络中,每次有数据包到达时,不需要都引发中断,因为高频率的中断可能会影响系统的整体效率。NAPI 在低流量时使用中断接收数据包,而在高流量时候则使用基于轮询的方式来接收,其好处是:1)中断缓和(Interrupt mitigation),在高流量下,网卡产生的中断可能达到每秒几千次,而如果每次中断都需要系统来处理,是一个很大的压力,而 NAPI 使用轮询时禁止了网卡的接收中断,从而释放了系统压力;2)数据包节流(Packet throttling),通过将接收到的超额的需要丢掉的数据包尽可能早地丢弃,根本不扔给内核去处理,减少了内核压力。

3 一种改进的基于网络特征的防 DDoS 检测算法

对于 DDoS[7-9]攻击检测而言,最重要的是把收取到数据包中的正常数据和攻击数据分离开。正常数据的来源 IP 一般没有规律,而对于攻击者,可能有两种情况:一种是直接使用同一个源 IP 进行攻击,这样的攻击比较低级,很容易被系统识别;另一种是随机 IP 伪造,算法就是对这种随机伪造的 IP 地址引起的报警进行统计和判断来推断出是否产生了 DDoS 攻击。

以下给出 snort 规则定义。

T:单位时间。

N:T 时间内入侵检测系统日志中的各种报警事件的总数。

K:抽样密度。

Q_{ip}:危险 IP 队列。这个队列中存放被过滤出来的可能为攻击源的 IP 地址。

R_{fak}:无回应报警率。随机抽样得到 N/K 个报警事件作为样本。其中没有回应包的报警包数为 A_{fak}、正常报警包数为 A_{nor}。则

$$R_{fak}=\frac{A_{fak}}{A_{fak}+A_{nor}}$$

R_{thr}:无回应报警率阈值(下限),如果 $R_{fak}>R_{thr}$,认为有可能发生 DDoS 攻击,系统转为防御处理阶段。

Q_R:R_{thr} 值队列,为确保 R_{thr} 更能反映最真实的网络状况,此队列实时更新。

P_{thr}:丢包数阈值。由于系统转为处理阶段之后会将可疑数据包丢弃,设定阈值协助判断 DDoS 攻击结束与否。

$count$:丢包计数器。计算一定时间内丢弃数据包数量。

$timer_R$:启动计时器,R_{fak} 开始计算时启动,用来给出时间获取数据源。

$timer_Q$:超时计时器,Q_{ip} 队列会向防火墙提供数据,若数据队列满,则计时器不起作用;若队列一直不满,计时器超时后会强制 Q_{ip} 队列提交数据。

$timer_C$:$count$ 的计时器,到达设定时间会将 $count$ 与 P_{thr} 做比较。

T:$timer_R$ 的阈值。

T_Q:$timer_Q$ 的阈值

T_C:$timer_C$ 的阈值。

M_{thr}:内存资源可用率阈值,即 DDoS 发生时内存最低可用率。

C_{thr}:CPU 剩余利用率阈值,即 DDoS 发生时 CPU 最低利用率。

M_R:实际内存可用率。

C_R:实际 CPU 可用率。

N_{max}:每个 IP 最大的半连接数。DDoS 攻击可能由固定 IP 发出,如果某 IP 地址的半连接数大于 N_{max},则认为是发生固定 IP 的 DDoS 攻击。

U:报警在协议连接时的分布。这个变量是相对于 N_{max} 设置的,它反映的是 DDoS 攻击由随机伪造的 IP 产生时的判断标准,当 U 值比较小时,说明报警的分布比较平均,则发生 DDoS 攻击的可能性较大。随机抽样得到 N/K 个报警作为样本,样本内连接总数为 M,在第 i 次连接上的报警数为 N_i,则

$$U=\sum_{i=0}(N/KM-N_i)^2$$

L:计算出的取样数据是否发生 DDoS 攻击标准值,$L=\frac{N}{U}$。

Q_L:长度为 N 的 L 值队列。元素为 L_{ddos1},L_{ddos2},…,L_{ddosN},代表从最近发生的 DDoS 攻击中随机取样 N 次计算出的 L 值。

Q_{nor}:长度为 N 的 L 值队列。元素为 L_{nor1},L_{nor2},…L_{norN},代表从最近没有发生 DDoS 攻击中随机取样 N 次计算出的 L 值。

L_{thr}:用于判断 DDoS 攻击是不是达到 L 阈值。当 $L>L_{thr}$ 时,表示 DDoS 攻击发生,需要转换到防御处理阶段。

$$L_{thr}=(\sum_{i=1}^{N}L_{nor_i}-\sum_{i=1}^{N}L_{ddos_i})/2N$$

算法主要结合了系统资源和捕获包的 IP 特性两个方面来进行 DDoS 攻击的判断。首先判断系统资源是否超过阈值,若未超过,则认为无 DDoS 攻击发生;若超过,则进行 IP 检查,也分为两个部分,首先检查是否是固定 IP 攻击,继而检查是否是随机伪造 IP 的 DDoS 攻击。

算法伪码如下:

Procedure Detection( )

```
{  //初始化;
 if(QL 为空) /* 初始化 QL 队列 */
 {
   Lthr初始化;
   QL 中所有值=Lthr;
 }
 if (Qnor为空) /* 初始化 Qnor队列 */
   Qnor中所有值=Qnor;
 if (Rthr为空)//初始化 Rthr
   Qthr中所有值=Lthr;
 if (QR 为空)/* 初始化 Qn 队列 */
   QR 中所有值=Rthr;
   /* 以下代码用于判断进入防御处理阶段的时机,通过实时调整
    Lthr值来保证真实反映网络当前的情况 */
    While (1)
    {
      if (入侵检测系统未开启)
        break;
      //检测内存资源和 CPU 资源是否超过阈值,若未超过则系统
        继续处于一般检测状态,否则继续判断,直到内存使用率和
        CPU 使用率降到规定值以下
        else
        {
          while (1)
            if ((MR<Mthr) or (Ch<Cthr))
              break;
        }
    /* 判断某个 IP 地址的半连接数是否超过 Nmax,若超过,则认为
     此 IP 为 DDoS 攻击源 */
      if (Nip>Nmax)
    { 清空 QL,清空 Qnor;进入防御处理阶段;break;}
     //计算 U;计算 L;计算 Lthr;比较 L 值,确认是否有可能发生随
       机源 IP 的 DDoS 攻击,再用无回应报警率来结合判断是否
       发生 DDoS 攻击
     if (L>Lthr)
     { L 加入队列 QL;启动 timerR;
       /* 系统在此段时间内获取数据计算 Rfak */
       while (timerR<nT){};
       /* 查看半连接率是否比较高,若高,说明可能发生攻击 */
     if (Rfak>Rthr)
     { timerR=0;Rfak放入队列 QR;启动计时器 timerC;
       do
       { while (timerC< TQ)
       { 提取系统日志记录中的 IP 加入队列 Qip;
           count++;
           if(Qip队列满)
           break;
       }
       清空 Qip;
       timerC=0;
       }
       进入防御处理阶段;
       while (count > Pthr)
         解除 IP 限制;
       }
       else
           退出防御处理阶段进入检测阶段
     }
   else L 加入队列 Qnor;
     } }
```

4 入侵检测平台和报文捕获平台兼容性处理

snort 分析数据有两个来源:一是自己捕获包,另一是读取已保存的.pcap 文件。在未修改的系统中,snort 启动时调用 open_pcap()函数使得网卡工作于混杂模式[10],它便开始对捕获的所有数据包进行监控,然后调用 Libpcap 的 pcap_loop()、pcap_read()等函数来获取数据包。由此可见,要使 snort 能从所设计的平台捕获数据包,只要将数据包捕获工作从 Libpcap 转移到修改后的网卡驱动上即可,具体为:修改 snort 配置,使用 config interface:eth1 命令将 snort 的网络接口设置为数据捕获用高速网卡,这样在 snort 初始化时就调用该网卡的数据包捕获模块进行监控。

5 系统测试与分析

5.1 系统环境

硬件环境:

发包机配置:P5 xeon 3.0G,1G 内存,Intel82541GI 千兆电口网卡。

收包机配置:P5 xeon 3.0G,1G 内存,Intel82541GI 千兆电口网卡。

交换机:思科 WS-C3750-48TS-S 千兆交换机。

软件环境:Linux RHEL4.0[4]操作系统,内核版本为 2.6 以上。

测试环境的拓扑结构如图 1 所示。

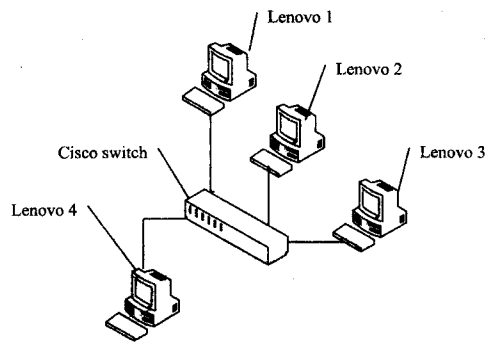

图 1 测试环境拓扑结构

5.2 发包程序

国内外在零拷贝平台方面的研究中,有使用专用的网络数据包硬件发送器,也有使用软件发包平台 TCPREPLAY。硬件发包平台可以任意调整数据包的大小、频率。TCPREPLAY 虽是较通用的发包平台,但是其数据统计都是在用户空间,数据包经过用户空间传入内核空间,内核空间再通过网卡将数据包发送出去,这将增加延时,若发送队列由于已满而无法满足发送条件,则会更加增大延迟。为了减小发送延迟造成的统计误差,开发了专用于 Intel E1000 网卡的发包程序,它可根据需要发送任意大小的 UDP[5]数据包,其原理是将要发送的数据包直接放在网卡硬件发送队列里,这样基本上不用延时网卡就会将数据包发送出去。由于发包程序本身

的缺点，使得只能以固定速率发送固定大小的 UDP 包，不能调整发送速率。发送 1024kB 大小的包时，发包速率维持在 450Mbps 到 600Mbps 之间。

5.3 入侵检测系统性能测试

5.3.1 攻击工具

选用了 TFN(Tribe Flood Network)[11] 作为系统测试的攻击工具，它使用了分布式客户服务器、加密技术及其它功能，可以产生随机匿名的 DDoS 攻击和远程访问，可以用于控制任意数量的远程机器。TFN 由客户端和守护程序组成，通过提供绑定到 TCP 端口的 root shell 控制，实施 ICMP flood、SYN flood、UDP flood 和 Smurf 等多种 DDOS 攻击。它的守护程序的二进制代码包最初是在一些 Solaris 2. x 主机中发布的，这些主机是被攻击者利用 RPC 服务安全漏洞“statd”、“cmsd”和“ttdbserverd”入侵的。最初的 TFN 守护程序来源于一些远程嗅探器(sniffer)和有访问控制的远程命令 shell，并可能结合了能自动记录的嗅探器。攻击者常常控制一个或多个 TFN 客户端，而每一个 TFN 客户端能控制多个 TFN 守护程序。所有接收到来自 TFN 客户端攻击指令的 TFN 守护程序都使用攻击数据包同时攻击一个或多个目标系统。TFN 网络的远程控制通过 TFN 客户端程序命令行的执行来实现。TFN 客户端程序运行无需口令，但需要有 TFN 守护程序端的 IP 列表文件“iplist”。从 TFN 客户端到 TFN 守护程序端的通讯通过 ICMP_ECHOREPLY 数据包完成，这样在 TFN 客户端和 TFN 守护程序端就根本不会有任何基于 TCP 或 UDP 的通讯。

由于设备限制，在测试中只使用了两台机器，分别作为攻击方和检测方。

5.3.2 测试结果

测试由两个部分组成。

1)新平台下的系统性能

第一部分测试的内容是入侵检测系统在本文所述高速报文捕获平台下对于系统性能的影响。使用传统的 snort 工具，在进行数据包捕获时使用 Libpcap 函数库，所以报文捕获方式也是 Libpcap 所使用的“每包驱动”方式。在改进的数据捕获平台中，snort 的报文捕获由 open_pcap()函数引向新的网卡驱动程序，由新的报文捕获平台完成数据捕获和记录。将在新报文捕获平台下的 snort 工具的丢包率与系统调用情况与在 Libpcap 下的做对比，如图 2 和图 3 所示。

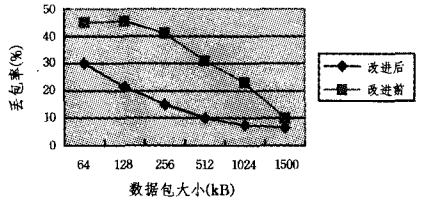

图 2 Libpcap 与高速报文捕获平台下 snort 丢包率对比

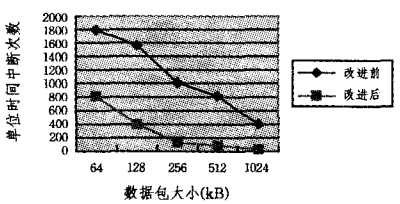

图 3 Libpcap 与高速报文捕获平台下 snort 对系统调用对比

从结果中可以看出，在使用了新的报文捕获平台之后，由于在进行数据包捕获时使用零拷贝技术屏蔽了内核，并且由于使用了半轮询技术，降低了中断次数，故 snort 的效率较在 Libpcap 下有所提高，特别是在系统调用方面，大大降低了系统负载，使得系统在接收数据包之余可以有较多空闲来进行数据处理和入侵检测。但在捕获的数据包较小时，与报文捕获平台的情况相似，丢包问题仍差强人意，尚待进一步改善。

2)入侵检测算法效率分析

新的防 DDoS 入侵检测算法主要强调系统发现攻击的效率和系统发现攻击后在检测阶段和入侵处理阶段之间的切换效率，故第二项测试内容是对 snort 报警效率的检测。测试时，先发送普通的数据包，设置 TFN 从若干分钟(本实验设置为 10 分钟)后开始发送 DDoS 攻击包。实验证明，检测系统的反应速度在 2～3 秒左右，能及时地从检测阶段切换到处理阶段；攻击持续 10 分钟，在攻击结束后 3 秒内判断攻击结束，切换到检测阶段。从判断的准确性来说，使用 TFN 攻击工具进行的攻击能 100%被检测到。对于未加入本文所设计的算法之前的 snort 工具，测试出从攻击发生到切换到处理阶段的响应时间大约为 4～5 秒。可见使用 IP 地址分布和系统资源作为参数来判断入侵检测的发生，对提高 snort 的响应速度有一定的帮助。

结束语 NACP 虽然就 Libpcap 来说在高速网络环境下其性能有了较大提高，但是对于小数据包而言，捕获性能仍然有待加强。下一步重点将放在如下两个方面：改进小数据包的捕获效率；在真实网络环境下实现不同大小数据包捕获速率测试。

参 考 文 献

[1] 温曙光，谢高岗. libpcap-MT：一种多线程的通用数据包捕获库[J]. 计算机研究与发展，2011，48(5)：756-764

[2] 王佰玲，方滨兴，云小春. 零拷贝报文捕获平台的研究与实现[J]. 计算机学报，2005，28(1)：46-51

[3] 乔思远. 基于 DMA_ring 的高速网络报文捕获机制的实现及应用[D]. 济南：山东大学，2007

[4] 王磊. 基于 Linux 的高速网络数据包捕捉技术的研究与实现[D]. 杭州：浙江工业大学，2007

[5] 倪继利. Linux 内核分析及编程[M]. 北京：电子工业出版社，2005

[6] 文旭，陈兵. μC/OS-II 高速网络通讯中 NAPI 的设计与实现[J]. 小型微型计算机系统，2008，29(2)：265-268

[7] 孙知信，姜举良，焦琳. DDOS 攻击检测和防御模型[J]. 软件学报，2007，18(9)：2245-2258

[8] 张永铮，肖军，云晓春，等. DDoS 攻击检测和控制方法[J]. 软件学报，2012，23(8)：2058-2071

[9] 郑康锋，王秀娟. 利用边际谱 Hurst 参数检测 DDoS 攻击[J]. 北京邮电大学学报，2011，34(5)：128-132

[10] 许爱军，谢娟，张华，基于 WinPcap 的网络数据解析及其实现[J]. 科学技术与工程，2009，9(10)：2799-2800

[11] 沈辉，张龙. 基于 WinPcap 的网络数据监测及分析[J]. 计算机科学，2012，39(10)：15-18