并行入侵检测系统的动态自适应负载均衡算法

唐拥政,刘解放,周 宁
(盐城工学院 现代教育技术中心,江苏 盐城 224051)

摘要:网络入侵检测系统的处理速度难以跟上网络的速度,使用多个分析引擎并行处理网络报文可以大幅度提高网络入侵检测系统的性能。考虑到负载均衡的要求,提出了一种并行入侵检测系统的动态自适应负载均衡算法,该算法给每个分析引擎设置了一个数据包接受区间,通过对网络报文的报头信息做哈希运算,把数据包映射到分析引擎的接收区间内;根据分析引擎的处理能力和负载情况调节各个分析引擎接受区间的宽度,从而合理分配每个分析引擎上的网络流量,充分利用所有分析引擎的计算能力。理论分析和实验结果表明,该算法在高带宽环境中有较高的效率。

关键词:并行入侵检测;负载均衡;高可用性;哈希函数
中图分类号:TP393.08 **文献标识码**:A **文章编号**:1671-5322(2011)04-0039-05

作为一种有效的网络安全保障技术,网络入侵检测系统(Network Intrusion Detection System,简称NIDS)在当前的企业网络环境整体保护中的作用变得越来越重要。一个典型的NIDS主要由分配器、IDS分析引擎和控制器等组成。分配器通过轮询策略监听获取所有进入监控网段的数据包,并分配到分析引擎;分析引擎对数据包进行检测之后,将产生的信息上报控制器,控制器主要负责控制和管理分析引擎,并接收和显示来自分析引擎的信息。

现有的NIDS很多基于特征模式匹配的算法,在当前高速网络环境下,这类系统的分析引擎面临严重的性能瓶颈问题。主要包括以下方面的因素:一是随着网络带宽飞速提高,分析引擎需要分析处理的网络流量大大增加;另一个是网络攻击方式的复杂多变,分析引擎需要匹配的特征模式随之大大增加,单个数据包的处理效率因而下降。这些原因使得原有的入侵检测系统在当前的高速网络环境下会出现严重的丢包现象,因而产生较高的漏报率。文献[1]指出当前面临的主要挑战就是NIDS跟不上网络带宽的不断增长。

为了解决高速网络环境下NIDS的性能瓶颈问题,人们从不同方向做出努力。一种办法是通过提高单个分析引擎的处理能力和提高入侵检测算法的效率来增强NIDS的性能[2];另一种办法是利用并行化处理技术,将流量分成几部分,由不同的分析引擎并行检测。TopLayer Networks推出一种NIDS负载均衡器IDS Balancer[3],这种负载均衡器,可以使多个分析引擎并行处理,共同承担高负载下的检测任务。文献[4]探讨了一种通过并行处理以实现高速网络检测的NIDS方法。但是,文献[4]和TopLayer Networks的负载均衡器一样,都没有对IDS的负载均衡算法进行深入探讨,而只是采用了简单的轮转(Round Robin)算法[5,6]。

本文通过对网络模型以及影响负载均衡的因素进行分析,提出并实现了并行入侵检测系统的动态自适应负载均衡算法(Dynamic Adaptive Load Balancing Algorithm,简称DALBA)。

1 负载均衡理论

负载均衡机制包括两种:一种是静态负载均衡,另一种是自适应动态负载均衡。静态负载均衡根据已有的知识和经验分配任务,运行时不再

收稿日期:2011-08-27
基金项目:江苏省盐城市科技发展计划项目(YK2009092);江苏省现代教育技术研究项目(2010R15239)
作者简介:唐拥政(1973-),男,江苏盐城人,副教授,硕士,主要研究方向为移动通信、无线传感网。

改变;而自适应动态负载均衡能够根据当前负载情况自动调整分配,能够更好地调度系统资源,提高检测系统的整体性能。

自适应负载均衡主要包括状态监控和负载分配两个过程。状态监控对系统内的各个节点的负载信息进行周期性的监控和收集;负载分配就是根据收集到的信息,对系统中的负载资源进行重新分配。

1.1 静态负载均衡算法

目前常见的静态负载均衡算法有以下几种:

(1)轮转算法:轮转算法的活动可以预知,假设有N个节点,每个节点被选择的机会就是$1/N$,很容易就可以算出节点的负载分布。该算法最大的特点是简单,不需要记录当前的连接状态及各节点的负载情况。轮转法适用于所有节点处理能力和性能都相同的情况。

(2)加权轮转算法:对轮转算法做了改进,不同节点按处理能力分配不同的权值,权值高的比权值低的分配更多的数据流量。

(3)散列算法:又称哈希算法(HASH),通过单射不可逆的HASH函数,按照某种规则将网络请求发往节点,否则返回空。

1.2 动态负载均衡算法

常见的动态负载均衡算法有以下几种:

(1)最少连接法:记录当前所有的活跃连接,并把下一个新的请求分发给当前连接数最少的节点。

(2)加权最少连接法:克服最少连接法的不足,为每个节点分配不同的权值,将新的请求分发给连接数和加权值比值最小的节点。

(3)最快响应法:记录到达每个节点的网络响应时间,并把下一个到达的连接请求分配给响应时间最短的节点。

通过实验证明,在通常情况下,动态负载均衡算法比静态负载均衡算法的性能有30% ~40%左右的提高。

1.3 对负载均衡设计的要求

负载均衡算法决定了入侵检测系统的性能和效率,如果设计不当会增大IDS的额外开销,增大入侵检测时间,造成系统负载失衡。负载均衡算法的设计应当满足以下要求:

(1)能够保证各个分析引擎的负载均衡,防止出现单一节点负载过大的情况。通常使用分析引擎输入队列的长度来衡量分析引擎的负载情况,如果某个分析引擎队列长度远大于其它分析引擎,就有可能因为队列溢出而造成数据包丢失。对于不同的分析引擎,应当能够根据其不同性能分配相应大小的负载,防止出现多个节点争夺处理同一个数据包的情况。

(2)负载的分配应当由分配器协商并行处理完成,不需要独立的调度部件来处理,以消除单点故障的瓶颈。同时负载的分配不能消耗分配器过多的性能,应当能够减少负载分配时间,这样可以增加单位时间内系统的吞吐量。

(3)每个分析引擎上的数据流量应当包含检测攻击所需的全部证据,这就保证不需要其它分析引擎的协助,分析引擎就能够单独检测攻击。

(4)检测系统应该能够通过控制器自动检测整个系统的资源状态,提高系统响应速度。当某个分析引擎超负荷运行或发生故障以后,系统应当自动检测并做出迅速的反应,自动将其负载转移到其它节点上,同时新的分析引擎接入或移出时,系统应当能够自动处理,不应影响其它分析引擎的正常运行。

(5)一个面向连接的数据流应当尽可能分配到同一个分析引擎进行处理。分配器采用的是状态检测方式,一个面向连接的数据流分配到不同分析引擎处理会导致状态检测异常。

1.4 负载均衡的影响要素

实现数据流量的动态自适当负载均衡,应当能够准确及时地把握节点的负载情况,根据各分析引擎节点当前资源使用情况进行动态调整。

在自适应负载均衡衡算法的设计过程中,应当周期性地收集以下参数和状态:(1)CPU利用率;(2)内存利用率;(3)当前连接数C_i;(4)当前网络流量T_i;(5)响应时间RT_i。这里的i表示分析引擎的节点号。

当入侵检测系统中的分析引擎首次投入使用时,根据分析引擎的硬件配置以及处理性能对每个分析引擎节点预设一个初始权值$OW(N_i)$,性能越高的分析引擎节点初始权值越高。随着分析引擎节点负载的变化,系统根据分析引擎运行参数不断调整并计算出动态权值,再根据动态权值的大小均衡数据流量[7-8]。

分析引擎节点收集参数的周期应当配置确当,短的周期虽然能更精确地反映分析引擎节点负载状况,但同时会增大分析引擎节点数据流量的开销,会造成分析引擎负载信息出现异常的抖

动。因此,交换周期应该适当调整,通常设为5~10 s。

2　动态自适应负载均衡算法

一种基本的负载均衡算法,可以根据报文的报头信息(如源/目的IP地址、源/目的端口号等),按类别分发到不同的分析引擎,但是这样不能避免分析引擎不超载,比如当某一时段大量信息发送到同一地址,这种算法就会将数据包都发往同一分析引擎。这样,检测效率反而会降低,数据包分发的原则是优先分发到剩余处理能力高,也就是负载小的分析引擎。

2.1　算法设计思想

IDS分析引擎组通过集线器和分配器相接,每个分析引擎都能得到全部来自外部网络的信息。首先将网络报文的报头信息(源/目的地址、源/目的端口、协议号等)作为散列函数的输入参数,计算出相应的散列值,然后将这个散列值按可用分析引擎数进行等分,每个分析引擎占用其中一个数值区间。

分析引擎节点接收到网络连接的数据报文时,首先提取其报头信息,利用预先设计的HASH函数对此报头信息进行处理并得到散列值,然后判断这个结果是否属于本分析引擎的散列区间。如果是,则该分析引擎节点设备对此网络报文进行处理,同时在分析引擎内存中维护的状态表中进行登记,保存与此连接相关的信息以及分析引擎号;否则丢弃这个网络报文。

如果接收到非网络连接的数据报文,分析引擎根据连接信息检索内存中的状态表。如果检索到相关记录并且标记的节点号与本分析引擎相同,则对此数据报文进行处理;否则丢弃。

2.2　算法的基本定义

哈希算法的设计是自适应负载均衡算法的重心,直接决定系统的性能和效率。算法必须简单、易于计算与实现;负载分配要均匀,不能发生振荡现象;同时要求满足低开销和高效率。

按照以上哈希算法设计的要求,定义了HASH包选择函数:

HASH包选择函数指通过散列运算,把数据报文的报头信息映射到分析引擎节点号上。利用函数 $\mathrm{HASH}() = (S_1 \oplus S_2 \oplus S_3 \oplus S_4 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4) \bmod N$ 把各个数据包映射到数值1,2,…,n中的某个值,其中 S_i 和 D_i 分别表示源/目的IP的第 i 个8位,这样,通过散列选择函数映射关系,能够保证同一连接的所有数据包交由同一个分析引擎来处理。

2.3　系统的监控管理——心跳协议

负载均衡的实现以及分析引擎的监控管理通过分析引擎节点间的心跳协议进行,心跳协议周期地对系统中的各个分析引擎节点的工作状态进行监控,节点工作状态发生变化时及时通知其它节点。通过心跳协议,各个节点周期性地交换负载信息,计算动态权值,进行负载均衡调整。如果在规定时间内没有收到某个分析引擎节点的心跳信息,那么就认为这个分析引擎发生故障。

当检测到某个分析引擎节点发生故障或者有新分析引擎节点加入并行入侵检测系统时,系统立即根据可用的分析引擎节点重新计算动态权值,重新进行负载均衡分配。同时,当某个分析引擎节点处理队列达到一定阀值,将不再向该节点分配请求。这样既保证了分析引擎节点流量均匀分布,同时保证每个连接能得到快速的响应。

2.4　系统结构模型

对文献[3]中的典型网络入侵检测系统进行改进,参考模型如图1所示。在此系统中,负载均衡系统由均衡器和多个分析引擎节点组成,负载均衡器从高速网络中监听报文,按照负载均衡算

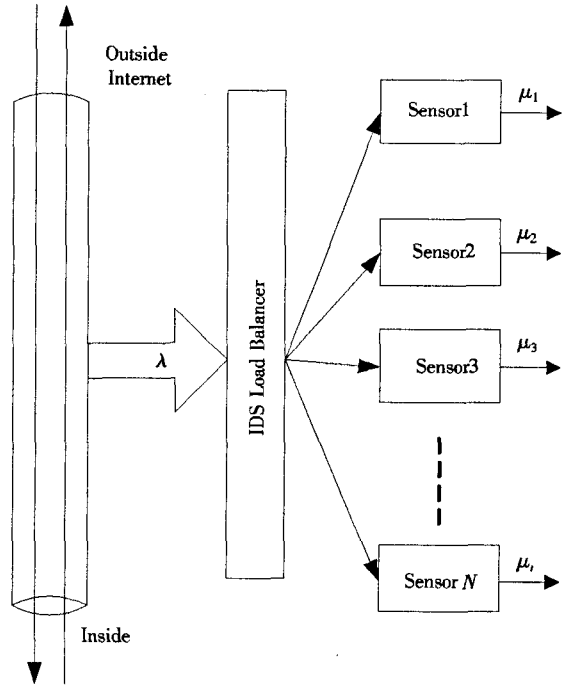

图1　参考模型

Fig. 1　Reference model

法将报文分配到相应的分析引擎,分析引擎独立处理各自的网络报文,并进行入侵检测。并通过心跳协议进行系统的监控和管理。

3　实验结果及分析

根据上述负载均衡算法的设计,对系统进行相关的测试,测试环境是运行 Snort 的 PC 机,操作系统为 RedHat 9.0,处理器和内存分别为 Core 2 Duo2.67GH/2GBM 所构成的并行入侵检测系统,并分别对此并行入侵检测系统进行了速度、平均延迟和吞吐量的测试与评估。

(1)通过一台 SmartBits600 来产生高速的网络流量。分发器从一个网卡接收 SmartBits600 产生的数据包,经过动态自适应负载均衡算法处理后从另外一个网卡发出。SmartBits 600 产生不同大小的数据包时,分发器所能够达到的最高速度被记录下来,如图 2 所示。

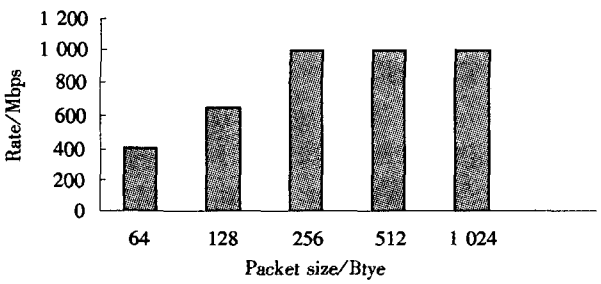

图 2　不同数据包大小时的最高速度

Fig. 2　Top speed of different packet sizes

结果表明,当数据包大于等于 256 Bytes 时,系统达到 1 Gbps 的处理能力,比多数的商用网络入侵检测系统要好。由此可见,文中的动态自适应负载均衡算法是高效的。

(2)对比传统的静态算法、Pick - KX 算法和本文的动态自适应负载均衡算法(DALBA),测试结果如图 3 所示。

如图 3 所示,当数据包较小时,由于静态分配算法不需要负载均衡调控系统,因此速度快于 Pick - KX 和 DALBA 算法;但随着数据包的增大,一般数据包大于 20 MB 后,只要系统还没有出现

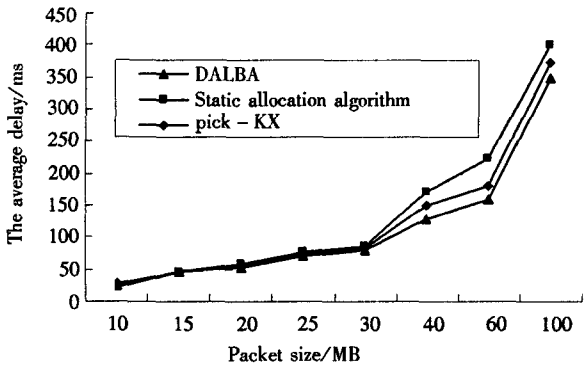

图 3　平均延迟对比

Fig. 3　Contrast of average delay

负载均衡器成为系统颈瓶时,DALBA 算法则略优于静态算法和 Pick - KX 算法。由此说明新算法能够有效地减少平均应答时延。

(3)从吞吐量的测试数据看(如图 4),和单个分析引擎的入侵检测系统比较,系统总体性能稳定,单向数据流吞吐量性能有了 60% ~70% 的提高。

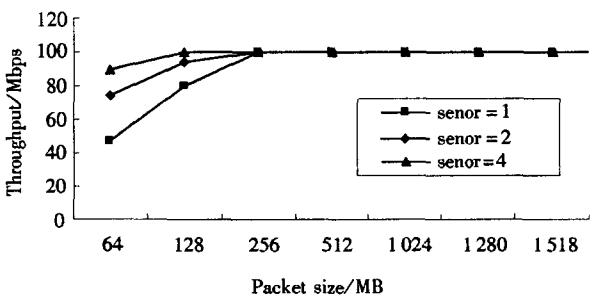

图 4　并行入侵检测系统吞吐量对比

Fig. 4　Throughput contrast of the IDS cluster

4　结 论

本文提出适用于高速网络环境的并行入侵检测系统模型,以及能够根据分析引擎负载情况进行动态自适应负载均衡的算法。性能分析和试验结果表明,该模型及算法具有良好的故障冗余性及高度的可扩展性,模型中的负载均衡算法简单、高效,具有局部最优性。在高带宽环境中,该模型有较高的效率。

参考文献:

[1] Schaelicke L, Slabach T, Moore B, et al. Characterizing the performance of network intrusion detection sensors[C]. Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection (RAID 2003). Lecture Notes in Computer Science, Springer - Verlag, 2003.

[2] Edwards S. Vulnerabilities of Network Intrusion Detection Systems: Realizing and Overcoming the Risks[Z]. Toplayer Networks, 2002.

[3] Coit J, Staniford S, McAlerney J. Towards faster string matching for intrusion detection or exceeding the speed of snort [C]. Proc DARPA Information Survivability Conference and Exposition (DISCEX II). Los Alamitos, Calif: IEEE CS Press, 2001:367 - 373.

[4] Kruegel C, Valeur F, Vigna G, et al. Stateful intrusion detection for high - speed networks[C]. Proceedings of the IEEE Symposium on Security and Privacy. Berkeley, CA: IEEE, 2002:285 - 294.

[5] Bestavros A, Crovella M E, Liu J, etal. Distributed Packet Rewriting and Its Application to Scalable Server Architectures [C]. Proceedings of 6th IEEE International Conference on Network Protocols, 2007:290 - 296.

[6] Dias D M, Kish W, Mukherjee R, etal. A Scalable and Highly Available Web Server[C]. Proc. of 41st IEEE Computer Society Intl. Conf. (COMPCON 2005), 2008,2:85 - 92.

[7] Leland W E, Taqqu M S, Willinger W, etal On the Self - similar Nature of Ethernet Traffic[J]. IEEE/ACM Transactions on Networking, 2011,2(1):1 - 15.

[8] Abry P, Veitch D. Wavelet Analysis of Long range dependent Traffic[J]. IEEE Trans. on Information Theory, 2008,44 (1):2 - 15.

A Dynamic Adaptive Load Balancing Algorithm in Parallel IDS

TANG Yong-zheng, LIU Jie-fang, ZHOU Ning

(School of Information Engineering, Yancheng Institute of Technology, Yancheng Jiangsu 224051, China)

Abstract: As the band of computer networks increases, the processing speed of network intrusion detection system hardly keeps up with the speed of networks. By arranging several analysis engines to deal with the traffic in parallel, the intrusion detection system's throughput can be significantly increased. There is an emerging need for parallel intrusion detection techniques that can keep up with the increased network throughput. A novel algorithm of load balancing among parallel IDS (Intrusion Detection System) was proposed to improve detection ability of parallel IDS. The algorithm was adopted to hash the network packet header information in packet, to map the corresponding packet to scope of the analysis engines' number, and to adjust the scope according to the performance and load of each sensor. Theoretic analysis and experimental results demonstrate that the algorithm can dispatch packets reasonably and utilize all the analysis engines' sources effectively.

Keywords: parallel IDS; load balancing; high availability; hash function

(责任编辑:沈建新)