一种提高网络入侵检测系统检测率的方法

曾茂林，文志诚，杨 润，向华政

（湖南工业大学 计算机与通信学院，湖南 株洲 412008）

摘 要：在繁重的网络流量或流量突爆的情况下，网络入侵检测系统（NIDS）无法对所有监控的流进行检测。针对NIDS的这种局限性，提出了一种选择弃包方法，该方法受系统负载的影响较小，能使NIDS预测到系统负载条件。通过对系统的性能分析实验，证明了在流量过载情况下，选择弃包法有效地提高了网络入侵检测系统的检测率。

关键词：入侵检测；选择弃包；过载控制；Snort

中图分类号：TP393.08 **文献标志码**：A **文章编号**：1673-9833(2011)02-0062-05

A Method to Improve the Accuracy of Network Intrusion Detection System

Zeng Maolin，Wen Zhicheng，Yang Run，Xiang Huazheng

（College of Computer and Communication，Hunan University of Technology，Zhuzhou Hunan 412008，China）

Abstract：Under conditions of heavy traffic load or sudden traffic busts, network intrusion detection system (NIDS) may not inspect all monitored traffic. In view of the NIDS limitations, presents a method of selective packet discarding, which is less affected by system load and enables NIDS to predict the system load conditions. By analyzing the system's performance, proves that the method improves the detection accuracy of NIDS under the traffic overload.

Keywords：intrusion detection；selective packet discarding；overload control；Snort

0 引言

网络入侵检测系统（network intrusion detection system，NIDS）是网络安全和网络鲁棒性的重要保障。当前，由于链路传输速度的提高以及网络威胁数量的不断增多，NIDS面临严重的容载问题。NIDS急需扩大处理流的容量，提高单位包的处理能力。当网络流负载超过NIDS本身所能承受的最大处理能力时，CPU处于饱和状态，将会在网络包被传送至NIDS前弃包。由于一部分攻击或恶意的网络包没有传送至NIDS中，这些攻击或恶意的网络包逃避了检测，引起NIDS处理能力下降。

文献[1-4]提出了改善NIDS包吞吐量的方法来提高入侵检测系统的性能。文献[5-6]采用调整NIDS配置的方法，谋求检测效率和资源需求之间的平衡来提高性能。然而，在高速网络环境下，基于非专门的硬件配置，NIDS同样面临无法处理所有监测到的网络包的问题[6]。尽管根据网络环境对NIDS进行了详细的配置，但它依然面临流爆发或进程急增的问题[7]。

通过观察网络包的传输内容发现，链路上最初传输的包对网路的建立至关重要[7]。例如：网络服务

收稿日期：2010-12-01

基金项目：湖南省教育厅科研基金资助项目（09C327）

作者简介：曾茂林（1984-），男，湖南永州人，湖南工业大学硕士生，主要研究方向为网络入侵检测，

E-mail：zmsc888@126.com

探测和侦查攻击，暴力登陆，反协议行为以及注入代码攻击的签名等，通常包含在网络流开头的前几百个包中。一个传输控制协议（transmission control protocol，TCP）链接开头的控制包对相关流跟踪以及 TCP 流重组起重要的作用，这也是当前主流 NIDS 的一个重要特征[8]。如果任何通过 TCP 三次握手过程中的包丢失，其相应的流将无法建立，潜在的攻击向量可能会逃过检测。网络上产生的流量大部分是与文件交换、点对点（peer to peer，P2P）流或者流媒体应用相关，而它们不属于安全威胁。虽然它们占用大量的网络流量，检测这些“严重攻击者”的所有包对 NIDS 的检测没有多大的意义。

本文提出选择弃包法，用此法 NIDS 动态分析产生过载的原因，使被舍弃的包对检测效率影响最小。通过选择性地舍弃对系统检测影响最小的包，使系统集中处理那些对检测进程来说最“有用的”包。将选择弃包方法应用于入侵检测系统 Snort [9]的预处理器，使预处理器运行在检测引擎之前。在过载情况下，Snort 丢弃了大量的包，检测出了大多数本可能丢失的包。因此，选择弃包方法提高了在高流量情况下 Snort 的检测率。

1　选择弃包的方法

理论上，NIDS 应该对监测点上的所有网络流进行监测，在高速的网络流下，由于监测传感器的有限计算能力，这种状况是不可能发生的。当网络流速达到 100 Mbit/s 时，NIDS 无法处理所有监测到的流，因此必须弃包[9]。解决 NIDS 无规则弃包问题的方法一般有 2 种：一种是，在初始过程中通过包过滤技术使一部分包不通过 NIDS，这样监测传感器不会过载，但同样存在过载和流爆发的问题。另一种是在网络流爆发的初期主动地丢弃那些与检测相关性很小的包，而不是由操作系统（operation system，OS）随机地丢包。依据第二种方法分析包在流中的位置关系确定哪些包应该被舍弃；然后通过评估 Snort 系统运行时性能，决定什么时候进行选择性弃包；最后根据系统性能评估，提出了用于动态估算舍弃包数量的算法。

1.1　基于流的包选择

选择一个典型的网络入侵检测系统，观察一些典型攻击，例如：端口检测、服务探索及 OS fingerprinting、代码注入攻击、暴力登陆，发现对网络威胁起重要作用的包存在于网络流初始建立时，其攻击目标包含在通信流的前若干 MB 中。一些占用网络流量巨大的应用，如：文件传输，基于因特网语音（voice over internet protocol，VOIP）电话，流媒体应用产生的超大流等，对网络安全不构成威胁，但它们占用大量网络流。NIDS 在网络通信的过程中，一直对这些与应用相关的包进行检测没有多大意义。

现代 NIDS 具有 TCP 流重组及跟踪功能，是网络通信中位置靠前的包之所以重要的另一个原因。通过 TCP 三次握手的包是处于 TCP 流前端的包，也是流的控制包，这些包对建立新流时更新自我的状态十分重要，因为这些包可以识别流的传输方向，能够重组 TCP 流。如果一个控制包在链接建立初期丢失，NIDS 认为没有建立相应的流，随后的攻击向量可能会逃脱检测。

高负载网络通信环境下，通过 TCP 的三次握手的包没有到达 NIDS 的流跟踪预处理器，基于包跟踪的规则将不会匹配流，潜在的威胁将无法检测到[10-11]。占用流开头的大部分包的攻击向量，如代码注入攻击的壳代码或者一个恶意的跨站脚本（cross site scripting，XSS）请求，通常是流重建后才能被发现。如果控制包被 OS 随机地舍弃，流重建预处理器将无法获取组成攻击的部分包，无法对攻击过程进行完整地重建。文献 [11]分析了网络攻击的轨迹，获取了攻击向量在流中的位置，利用 Snort 实验捕获互联网上 120 条不同的轨迹产生的包，用 Snort 自带的规则集进行分析发现，其中大部分的报警是由流通信过程中前期的包触发。例如，90% 的报警是由流开头的 30 个包组成，只有 3% 左右的报警由流 100 个包之后的包触发。流的产生通常发生在 Internet 广泛的包扩散传输之后，大部分网络包占的流量较少，只有小部分包占的流量较多[12]。

通过以上的分析，可得出结论：在高网络负载条件下，NIDS 应集中处理每个流开始的包，舍弃其余的包。

1.2　系统负载监测

实现选择性弃包需要在内核开始弃包前对过载条件进行分析。分析系统过载基于 3 个计算量：弃包发生的时间，CPU 使用度，NIDS 中包到达和处理的时间差。

理想状况下，要对系统负载进行监测，最优的选择是记录检测每个包时的 CPU 使用度，但是这种做法不现实，它要耗费大量的系统内存和硬盘资源。一种替代的方法是，测量 NIDS 的 CPU 使用度、处理时间及对每 N 个包进行处理时的弃包数（根据套接字缓冲大小自动选择 N）。依据准确测量 CPU 使用时间的时序解析度（timing resolution），可以获得弃包发

生的时间。

对每组 N 个包，系统检测某一周期中内核是否发生了弃包现象，测量 NIDS 的使用时间，操作系统的当前时间和操作系统运行时间。CPU 使用度可以用公式（1）表示，

$$\text{CPU使用度}=\frac{\text{NIDS使用时间+操作系统当前时间}}{\text{操作系统运行时间}}, \quad (1)$$

要得到 NIDS 中包到达和处理的时间差，需将处理每组 N 个包所需要的时间 t 和这些包在网络上被捕捉的时间间隔 s 进行比较，如果 $t>s$，表示内核可能发生弃包行为；如果 $t<s$，表示内核在这个间隔中没有发生任何弃包行为。

由以上的理论可知，在 CPU 使用度达到饱和之前预测包丢弃事件，需设定 CPU 使用度的阈值（下限阈值和上限阈值）以及获取 NIDS 处理每组 N 个包的运行时间，根据这些数值进行选择弃包。当 $t<s$ 和 CPU 使用度没有达到饱和的情况下不会发生弃包，这时系统可以根据使用度下限阈值决定是否处理流中更多的包。设定的最低阈值和最高阈值差值不能过大。这样，资源使用及 CPU 处理能力才不至于浪费，例如：设定上下阈值为 0.95 和 0.85，这时 CPU 使用度上限和下限分别为 95% 和 85%，以这 2 个值为依据进行考虑。在处理前 N 个包时 NIDS 进行选择弃包，由 2 种情况引起：1）CPU 舍弃一些包；2）CPU 使用度大于 0.95 且 $t>s$。当 CPU 使用度小于 0.8 且 $t<s$ 时，表明 CPU 没有发挥最大的效能，无需考虑负载情况。除此之外，表明系统需进行负载监测。

1.3 流大小限制调度算法

根据过载条件的查询，NIDS 应该减少检测包的数量。首先，指定应该舍弃的包的数量，然后，以这个数量作为限制网络流大小的依据，NIDS 获取网络流的大小。理想状态下，被丢弃包的数量是 CPU 处于理想状态下舍弃包的数量。

将 NIDS 处理每 N 个包的运行时间 t 设置为这 N 个包的捕获时间 s 的 0.95 倍，意味下一个 NIDS 处理组内被丢弃的包的数量将和进程时间 $t-0.95s$ 相关，下个组内的弃包数为 $(t-0.95s)N/t$。

因此，被丢弃的包的数量是以下两者的最大值：1）在时间间隔 t 中丢弃的包；2）下个组内的弃包数。如果 CPU 使用度在下限阈值之下，NIDS 应该增加并处理更多的包。NIDS 花费 $0.8s-t$ 时间处理下一组 N 个包，相当于处理（$0.8s-t$）N/t 个包。根据以上方法，获得估算 NIDS 在过载时弃包数量的机制，而选择策略是基于流大小限制，这需要设定合适大小的流。获取新流大小限制算法，是基于在将已到达 Snort 的包进行分流时 NIDS 收集的各个流的统计信息。流分类引擎利用分类表形式为预定义大小不同的流保存包的数量。

一旦发生弃包，算法开始缩小流统计表中当前的流限制范围，重新计算每个有更低限制范围的流对应的弃包数量，直到统计表中流的大小为理想的数量；增加流大小限制的过程和上述减少流大小限制的过程类似，即提高流统计表中当前流的限制范围直到弃包的数量满足要求为止。流大小限制调度算法过程如下。

Flow size limit adjustment algorithm:

input: size of socket（N=6 MB）, processing time（t=10 second）

steps:

1）record the captured time of pockets in every socket N, that is s; calculate the limit superior t_{max} and limit inferior t_{min} of CPU process-time.

$$t_{max}=0.95s,\ t_{min}=0.80s;$$

2）if $t<t_{min}$ turn step 8）, else turn step 3）;

3）calculate the counts of discarded packets in every flow in t, that is NUM1

$$\text{NUM1}=(t-0.95s)N/t;$$

4）Descends the size of socket N of flow statistics table in flow classification engine with NUM1;

5) reset t

$$t=t-0.01;$$

6）if $t<t_{max}$ turn step 10）, else turn step 1）;

7）Calculate the counts of dropped pockets from the next group, NUM2

$$\text{NUM2}=(0.8s-t)N/t;$$

8）reset t

$$t=t+0.01;$$

9）if $t>t_{min}$ turn step 10）, else turn step 1）;

10）keep the NUM1 and NUM2.

1.4 基于选择弃包方法的 Snort 系统改进模型

将选择弃包方法设计为弃包预处理器配置在 Snort 中，将此预处理器运行在 Snort 的检测器和其它预处理器之前。弃包预处理器从 Snort 的包解码器中得到网络包的信息，通过 hash 表查找包对应的流。基于流的大小和当前截流限制值，弃包预处理器决定包应该被舍弃还是转发到另外的 Snort 预处理器以及核心处理器。在 NIDS 中，当一段时间内流不再活动或正常的 TCP 连接终止时流将关闭。精确地记录 TCP 通信中止时间十分重要，通过记录 TCP 通信中止时

间可以阻止入侵者通过马上开启或关闭一个 TCP 通信以逃避检测。因此将每个新的连接定义为新流，并利用 Snort 检测流的开始包。检测的时间间隔大小及包处理量 N 根据套接字缓冲区大小和系统 CPU 资源使用情况自动设定。当处理了 N 个包后，处理器推理潜在的过载条件，并利用流大小限制调度算法判定流大小。改进后系统总体结构图如图 1 所示。

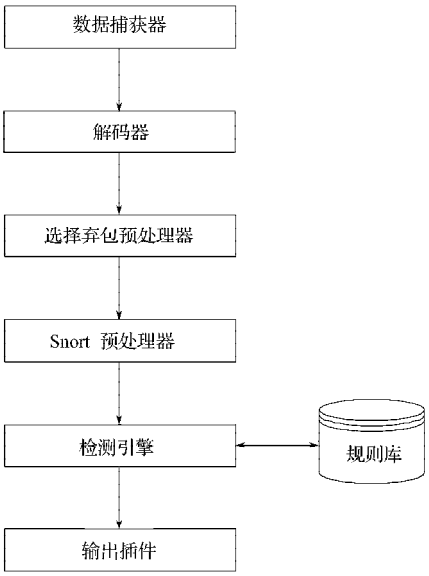

图 1　改进后的 Snort 系统结构图

Fig. 1　The diagram of improved Snort system

2　实验评估

2.1　实验环境

采用 10 Gbit 交换机相连的 2 台 PC 机进行实验，第一台 PC 机用于产生网络流量，第二台为 NIDS PC 机，套接字缓冲，大小设置为 6 MB。模型选择 Snort V2.8.3.2，Snort 设置为默认方式。2 台 PC 的配置参数为：CPU Intel(R) Pentuim(R) Dual T2370 @1.73 GHz；硬盘 60 GB；内存，2 GB；网卡 10 Gbit。

用网关跟踪网络流量，产生 58 714 906 个包，得到 1 493 032 个流，总大小约 40 GB。在跟踪过程中，Snort 用 78 个不同的规则一共产生了 1 976 个报警。对每个报警及其相匹配的包进行手动检测，由于没有对所有警报进行检测，跟踪到的报警可能是误报（false negative）。为了提高实验的正确性，在实验数据中混入 276 个报警，这些报警都是从 Internet 监测到的已定义为真阳性（true negative）的网络攻击。

2.2　高负载下的检测及分析

对原系统 Snort 进行实验，系统的性能分析如图 2。由图 2 可知，当处理速度超过 500 Mbit/s，内核开始弃包，这些丢弃的包超过整个实验包的 15%。当弃包时，CPU 的使用度一直高居 99%，由此可知，Snort 无法在高流量下有效地检测包。当流速为 500 Mbit/s 时，正好有 15% 的包被 OS 随机舍弃，这时 Snort 漏检了 18% 的报警。当流速达到 900 Mbit/s 时，46% 的包被舍弃。此外，在相同流速的多次运行下，每次 Snort 产生不同的报警集。这说明随机弃包引起了不确定性的报警报告。

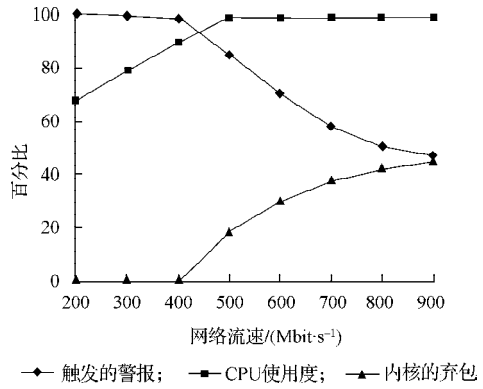

图 2　Snort 的性能分析

Fig. 2　Performance analysis of Snort

对改进后的系统进行实验，实验结果如图 3。由图可知，被 OS 舍弃的包数量的改变是微不足道的，当 NIDS 的处理速度达到 600 Mbit/s 时，没有发生弃包，当处理速度达到 900 Mbit/s 时，弃包率只有 0.098%。CPU 的使用度一直处于 0.8 的下限和 0.95 上限阈值之间。图中还显示被弃的包占所有包的百分比，被丢弃的包随着流速的增加而增加。因此可根据流量丢弃预期数量的包，Snort 控制 CPU 的使用度，使 CPU 处于理想状态。

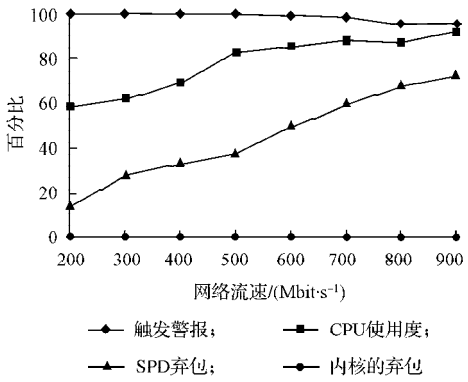

图 3　改进后的 Snort 性能分析

Fig. 3　Performance analysis of improved Snort

在相同的流速下，选择弃包的数量比没有改进时内核弃包要多，其原因有 2 个：一是在内核弃包前选择弃包算法主动地弃包，这使得预处理器在内核随机弃包前舍弃了流尾的大部分包；二是选择弃包算法将所有包先传到用户区，然后再由 Snort 的预处

理器弃包，尽管Snort检测的包比较少，选择弃包方法使系统依然具有较好检测率。

选择弃包方法能使Snort检测率显著提高。图3中,当流速达到900 Mbit/s时，Snort几乎能够检测到监测流中所有的报警。当流速为500 Mbit/s时，改进后的Snort检测到了2 234（总计2 252个）个报警，占总数的99.2%。相对没有改进的Snort而言，检测效率提高了20%。随着流量的增加，触发的报警率下降较少，当流量上升到900 Mbit/s，仅漏报84个报警。

3 结论

网络流过载是影响NDIS性能的一个较普遍的原因[13]。在超载的网络流或流的突爆情况下，系统无法对所有需要检测的流中的包进行处理，OS将不可避免地随机性丢包。本文提出的选择性弃包，能较好地减少流的数量，使NIDS的检测引擎在流过载情况下的影响最少，从而提高了系统的检测率。把选择性弃包方法融合到Snort系统中，使系统增强了包的处理能力。值得注意的是，一些攻击可能会利用在流大小允许范围内进行合法的请求，然后再发送攻击向量，这些攻击对于HTTP协议来说是可行的，因为HTTP协议允许在同一个连接中发送多次相同的请求。因此，改进选择性弃包方法，使之能更有效地控制流的大小还有待进一步研究。

参考文献：

[1] Attig M，Lockwood J. A Framework for Rule Processing in Reconfigurable Network Systems[C]//13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. Washington DC：IEEE Press，2005：225-234.

[2] Bruijn W D，Slowinska A，Reeuwijk K V，et al. Safe Card: a Gigabit IPS on the Network Card[C]//Proceedings of 9th International Symposium on Recent Advances in Intrusion Detection(RAID). New York：Springer press，2006：311-330.

[3] 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. Washington DC：IEEE Press，2002：285-294.

[4] Yusuf S，Luk W. Bitwise Optimised CAM for Network Intrusion Detection Systems[C]//Proceedings of International Conference on Field Programmable Logic and Applications. Washington DC：IEEE Press，2005：444-449.

[5] Dreger H，Feldmann A，Paxson V，et al. Predicting the Resource Consumption of Network Intrusion Detection Systems[C]//Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID). New York：Springer Press，2008：135-154.

[6] Schaelicke L，Slabach T，Moore B，et al. Characterizing the Performance of Network Intrusion Detection Sensors [C]//Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection(RAID). New York：Springer Press，2003：155-172.

[7] Smith R，Estan C，Jha S. Backtracking Algorithmic Complexity Attacks Against a Nids[C]// Proceedings of the Annual Computer Security Applications Conference. Washington DC：IEEE Press，2006：263-265.

[8] Ptacek T H，Newsham T N. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection[EB/OL]. [2010-11-02]. http://www.creangel.com/papers/Eluding%20Network%20Intrusion%20Detection.pdf.

[9]] Polychronakis M，Anagnostakis K G，Markatos E P. An Empirical Study of Real-World Polymorphic Code Injection Attacks[EB/OL]. [2010-10-12]. http://www.ics.forth.gr/dcs/Activities/papers/nemu-study.leet09.pdf.

[10] Lee W，Cabrera J B D，Thomas A，et al. Performance Adaptation in Real-Time Intrusion Detection Systems[C]// Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection(RAID). New York：Springer Press，2002：252-273.

[11] Roesch M. Snort-Lightweight Intrusion Detection for Networks[C]// Proceedings of the 1999 USENIX LISA Systems Administration Conference. Washington DC：IEEE Press，1999：536-546.

[12] Fang W，Petrson L. Inter-as Traffic Patterns and Their Implications[C]//Global Telecommunications Conference. Washington DC：IEEE Press，1999：205-207.

[13] McCanne S，Jacobson V. The BSD Packet Filter：A New Architecture for User-Level Packet Capture[C]//Proceedings of the Winter 1993 USENIX Conference. Washington DC：IEEE Press，1993：259-270.

（责任编辑：邓光辉）