串匹配型入侵检测系统的改进与实现

Improvement and Implementation of Content – based Intrusion Detection System

李红霞　（辽宁工程技术大学 软件学院 辽宁葫芦岛 125000）

摘　要： 针对当前串匹配型入侵检测系统普遍面临的误报漏报率高、自身性能难以适应快速增长的网络流量需求等问题，本文以提高检测速度为目的对串匹配型入侵检测系统进行分析，从串匹配算法、入侵特征库方面对其进行改进，提出并实现了两种改进方案。实验表明，本文提出的改进方案是可行的。

关键词： 串匹配 入侵检测系统 入侵特征库

1　概述

随着网络技术的迅速发展，计算机已经从独立的主机发展到复杂的、互联的开放式系统，这给人们在信息利用和资源共享上带来了很大的便利。与此同时人们也面临着由于入侵而引发的一系列安全问题的困扰，因此如何保证网络安全已是一个必须面对的重要问题。

当前有一类基于入侵特征的网络入侵检测系统，其检测原理如下：将每种已知入侵描述成一条规则，并将所有规则组织成一个规则库。检测时，首先对捕获到的网络数据包进行解析，然后与规则库中的所有规则逐条尝试匹配。如果匹配成功则认为入侵发生。由于数据包包头中携带的信息量十分有限，大部分规则都需要通过包负载中的特征字符串来识别入侵。因此，这类入侵检测系统又称为串匹配型入侵检测系统。

串匹配型入侵检测系统能否准确地识别出入侵行为，一方面依赖于规则库的完备性，另一方面也取决于能否对网络上的全部数据包进行监听和分析。近几年来，随着千兆以太网、ATM（Asynchronous Transfer Mode，异步传输模式）等技术的出现，网络带宽迅速增加，网络入侵检测系统面临着巨大的挑战，它的分析、处理速度越来越难跟得上网络流量，从而造成数据包丢失，因此迫切需要提高入侵检测系统的处理速度，进而解决数据包丢失问题。

针对这一问题，本文首先对串匹配算法进行改进，提出了一种基于排除的串匹配算法，该算法能快速而准确地排除负载中不包含匹配模式串的数据包，从而提高了匹配的效率。其次对入侵特征库进行了改进，一方面通过优化规则的分类标准，使得每条规则唯一地从属于一个规则链表，从而减少了存储空间和检测时间；另一方面考虑到应用服务使用频率的差异性，将常用的服务端口放在规则链表的前面，从而缩短了检测时间，提高了检测效率。

2　串匹配算法的改进

对于串匹配型入侵检测系统来说，匹配包负载中的特征字符串是检测过程中最费时的部分。以轻量级开源网络入侵检测系统 Snort 为例，通过对 Snort 实际运行过程中各函数的调用时间比例进行分析[1]，发现对数据包负载中特征字符串的匹配耗时最多，约占 31%。此外在 Snort 2.6.1 的规则集中有内容匹配选项（content 或 uricontent）的规则数目达到总规则数的 92% 以上。由上述数字可以看出对于串匹配型入侵检测，串匹配算法至关重要，它的性能将直接影响到入侵检测系统的快速性和准确性。针对这一问题，本文提出了一种基于排除的串匹配算法。

2.1　算法的基本思想

假设要检测数据包负载 T 中是否包含模式串 P。如果 P 中至少有一个字符不在 T 中，那么 P 就不可能在 T 中；相反如果 P 中的每个字符都包含在 T 中，则使用标准的字符串匹配算法来确定 P 是否是 T 的子串。在第二种情况下使用标准字符串匹配算法的原因是可

能存在下面错误匹配的情况：P 中的每个字符都在 T 中，但是它们在 T 中的出现次序与在 P 中的不同，因此 P 也不是 T 的子串。所以不能仅仅根据 P 中的每个字符都在 T 中而判定 P 是 T 的子串，而是需要标准的字符串匹配算法来做进一步的验证。

2.2 算法的实现

算法的实现分为两个阶段：预处理阶段和搜索阶段。前者依赖于一个能快速判断字符 c 是否在 T 中的方法，后者依赖于一个快速的标准串匹配算法。

（1）预处理阶段。这里借助存在图来判断字符 c 是否在 T 中。首先预处理 T，对于在 T 中出现的字符，用二进制值 1 来标记其在存在图中的相应位置；对于不在 T 中出现的字符，其在存在图中的相应位置标记为 0。实验表明对于每个新的数据包，清空存在图的费用很高。为了减少这个费用，用数据包的序列号来标记存在图。这样如果存在图的第 c_{th}（c_{th}为字符 c 的 ASCII 值）位置包含当前数据包的序列号，则表明字符 c 在此数据包中；否则 c 就不在数据包中。更加一般化，本文算法采用的是固定长度的位串（称其为元素 element），即判断位串是否在数据包中。因此就存在一个平衡的问题：如果元素太小，出现错误匹配的几率大，调用标准串匹配算法的次数增加，检测的时间也会随之增加；相反，如果元素太大，错误匹配的几率会降低，但同时也使存在图过于庞大（存在位图需要记录所有长度为 element size 的位串），占用大量的内存空间，从而导致系统运行时内存缺页率升高，影响性能。

（2）搜索阶段。由上所述，若模式 P 中的每个字符都在数据包负载 T 中，基于排除的串匹配算法就调用标准的串匹配算法来做进一步地判断。因此标准串匹配算法的性能好坏也对系统的整体性能产生影响。本文采用的是单模式匹配算法 BMH[2]（Boyer－Moore－Horspool）算法。

2.3 算法的性能分析

本文提出的基于排除的串匹配算法的预处理和搜索阶段的伪代码如下：

```
pre_process(char *T, int len){
    for(int idx=0; idx<len; idx++){
      c=T[idx]<<(elementsize-8)^T[idx+1];
      occurrence_map[c]=pktid;}}
    search(char *P, char *T, int len_P, int len){
    for(int idx=0; idx<len_P; idx++){
      c=P[idx]<<(elementsize-8)^P[idx+1];
      if(occurrence_map[c]!=pktid)
        return DOES_NOT_EXIST;}
    return BMH(P, len_P, T, len);}
```

从上面的伪代码可以很清楚地看出，预处理阶段的时间复杂度与数据包负载的大小成线性关系，搜索阶段的时间复杂度与所有模式的长度和成线性关系。

本文提出的基于排除的串匹配算法的一大特点就是它能够快速地排除负载中不包含匹配模式串的数据包。由于在大多数数据集中，入侵数据包毕竟是少数。据统计，仅有 1.6% 的数据包需要用标准的字符串匹配算法进一步确定该数据包中是否包含入侵特征串。从而可以得出在检测入侵时调用标准串匹配算法的几率很小，缩短了检测的时间，所以该基于排除的串匹配算法适用于入侵检测领域。

3 入侵特征库的改进

串匹配型入侵检测系统分析各种类型的攻击，找出可能的攻击特征集合，利用这些特征集合或者对应的规则集合，对当前数据包进行各种处理，再进行特征匹配。由此看出入侵特征库对检测引擎的性能将会产生一定的影响。本文将从规则划分标准、规则链表结构两方面进行改进。

3.1 规则划分标准的改进

通过对 Snort 规则划分标准的分析，可以看出如果规则的源、目的端口都唯一，则这条规则将会加入到两条规则链表中，即对应于特定目的端口值的 PORT_GROUP 结构的规则链表和对应于特定源端口值的 PORT_GROUP 结构的规则链表，就出现了规则的冗余。这样既浪费了内存空间，又增加了检测的时间。因此，在此对 Snort 现有规则划分标准进行改进，使得每条规则唯一地属于某一规则链表，进而消除冗余。

根据对 Snort 规则的分析，可以将所有的规则划分为三个互不相交的子集合：目的端口唯一（源端口或者唯一或者是任意值）的规则子集合；目的端口不唯一，源端口唯一的规则子集合；目的端口和源端口均不唯一的规则子集合。据统计，Snort 2.6.1 总共 3600 多条规则，其中具有单一目的端口值的规则占 86.3%。因此当进行规则划分时，首先判断规则的目的端口值是

否唯一，如果唯一就将其加入到第一个规则子集合中，否则继续判断其源端口值是否唯一；如果源端口值唯一将其加入到第二个规则子集合中，否则将其加入到第三个即通用规则子集合中；对于规则中端口值求反操作或者指定值范围的情况，等同于端口值为任意值的情况加以处理。此外，系统中对于不含端口值的规则类型(ICMP 或者 IP 协议类型)，采用的处理方法与原来的相同。

3.2 规则链表结构的改进

通过对 Snort 快速规则匹配引擎构建部分代码的分析，可以知道对于每个 PORT_RULE_MAP 数据结构中的两个 PORT_GROUP 类型的数组，是按顺序规则存储的；即规则的端口号与其在数组中的索引号是一致的。例如对于目的端口为 80 的规则链表，其在数组 prmDstPort 中的索引就是 80。假设一个应用服务使用很频繁，即数据集中这种服务的数据包占很大比例，但是它所使用的端口号较大，这样在每次检测时都要从数组的第一个元素开始查找，找到相应的端口号对应的规则链表之后，再进行检索，这样势必影响检测的效率。因此考虑到应用程序使用频率的差异性，将常用的应用程序端口号放在数组的前面，这样就缩短了检测的时间，提高了系统的效率。

规则链表结构的改进涉及到两处：一是构建快速规则匹配引擎部分；二是利用快速规则匹配链表进行入侵检测部分。在构建快速规则匹配引擎时，将常用端口号与前面的端口号进行互换；在检测入侵时，为保证数据包的端口号与规则链表中的端口号一致，需要再次进行端口号的转换。涉及到的相应函数为 prmAddRule( )、prmAddRuleUri( )、prmAddRuleNC( )、fpEvalHeaderTcp( )、fpEvalHeaderUdp( )。

4 实验与结果分析

4.1 实验环境

本实验使用一台 PC 机，运行 RedHat Linux 9.0 操作系统，内核版本 2.4.20，处理机 P4，主频 2.4GHz，内存 512M，一级数据缓存 8KB，二级数据缓存 512KB，Snort 版本是 2.6.1，gcc 版本 3.2.2。

4.2 算法性能测试

(1) 最佳元素大小的测试。首先确定系统性能最佳时元素的大小。图 1 显示了不同元素大小对应的错误匹配比例，图 2 显示了相应 Snort 的运行时间，这些值是利用 Linux 操作系统本身的 time 命令获得的。

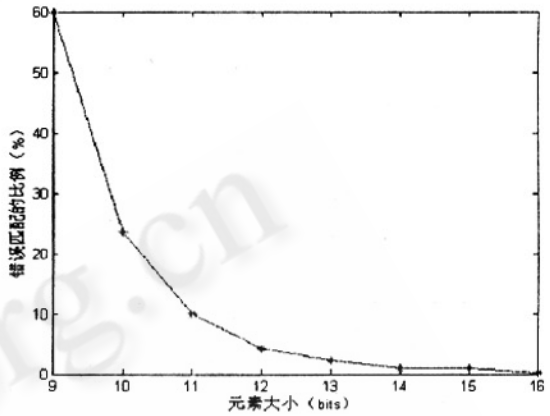

图 1　元素大小对错误匹配比例的影响

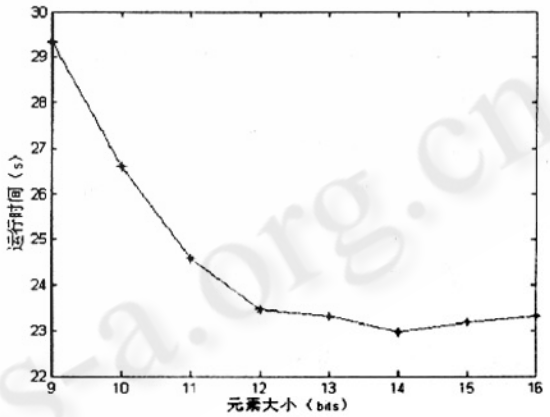

图 2　元素大小对 Snort 总运行时间的影响

从图 1 可观察到当元素大小超过 14 位时，错误匹配的比例低于 1%。随着元素大小的增加，错误匹配的比例不断减少，因此调用标准串匹配算法的次数明显减少，所以总运行时间也随着减少。然而从图 2 可以看出运行时间不是随着元素大小的增加而严格地减少，当元素大小超过 14，运行时间就开始有所上升，这与机器本身高速缓存的数据结构有关。对于我们具体的配置，元素大小为 14 时系统性能达到了最佳。

(2) 不同数据集的测试。为了说明本文提出的算法的通用性，在本实验中采用了三种不同的数据集：麻省理工学院林肯实验室提供的"1999 DARPA 入侵检测

测试数据集",通过本机网卡的所有数据包,本校校园网的数据集。实验结果如表1所示:

表 1　不同串匹配算法的时间比较

数据集性质			运行时间(s)		
数据集名称	数据包总数	包平均长度(字节)	原来的算法集	本文提出的算法	提高(%)
inside1. tcpdump	1,647,573	165	13.29	12.25	7.83
inside2. tcpdump	1,766,014	216	17.62	15.45	12.32
inside3. tcpdump	2,356,477	210	22.84	20.94	8.32
outside1. tcpdump	1,279,543	161	10.24	8.81	13.96
outside2. tcpdump	1,309,242	220	12.20	10.02	17.87
pc. tcpdump	267,942	245	1.49	1.48	0.67
lan. tcpdump	255,007	337	1.59	1.55	2.52

通过观察我们可以得出,本文所提出的算法比Snort原来算法集的性能都有所提高,虽然对于不同的测试数据集提高的幅度不同。另外,对于相同类型的数据集,数据包越大,性能提高得越明显。这点是可以理解的,因为对于小数据包,处理包头的相对费用比大数据包要高,因此对于小数据包,一个改进的串匹配算法不可能达到非常明显的效果。

(3) 不同大小规则集的测试。为了更好地理解本文算法的性能,我们测试了规则数与运行时间的关系。实验结果如表2所示。

表 2　不同规则数对 Snort 运行时间的影响

规则数目	运行时间(s)		
	原来算法集	本文的算法	提高(%)
1414	19.65	17.87	9.06
1597	20.50	18.41	10.20
1737	22.11	19.50	11.80
1864	23.90	21.06	11.88
2246	24.50	21.21	13.43

从表2可以看出,随着规则数目的增加,本文算法的性能明显优于Snort原来的算法集。这是因为虽然本文算法要为每个数据包的预处理付出代价,但是当数据包与多个规则进行匹配时,开始的预处理开销被大量规则平摊,因此它对总运行时间的影响就降低了。

4.3 入侵特征库的改进测试

为了更明显地体现本文提出的入侵特征库改进方案对系统性能的影响,在这部分测试之前要清楚地知道对于所测试数据集数据包的分布情况,即各个协议数据包的所占比例,选择出所占比例最大的协议,然后改进对应的规则链表结构。对于本实验所采用的数据集,通过实验我们得出目的端口号为80、139、445、21、25的规则数目最多,因此在程序中将这些目的端口放在对应数组的前面,即原来的80、139、445、21、25端口与最前面的0、1、2、3、4端口的位置互换,而其他的端口号链表不变。改进后的实验结果显示特定源端口链数由原来的34变为33;改进后的系统在准确率(包括漏报率和误报率)不变的情况下,时间缩短了5%左右。

5 结束语

本文以提高检测速度和准确率为目标,从串匹配算法、入侵特征库方面对串匹配型入侵检测系统进行了改进。实验结果显示,本文提出的基于排除的串匹配算法和入侵特征库的改进方案,使得Snort在准确率不变的前提下,检测效率有了一定的提高,表现出了令人较为满意的结果。

参考文献

1 M. FISKY, G. VARGHESE. An analysis of fast string matching applied to content - based forwarding and intrusion detection[R]. UCSD Technical Report CS2001 - 0670(updated version), 2002.

2 MIKE FISK, GEORGE VARGHESE. Applying fast string matching to intrusion detection, 2002.