改进的模式匹配算法及在入侵检测中的应用

甘学士,孙力娟
(南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:基于字符串匹配的检测方法是入侵检测系统中一类重要的分析方法。文中在分析现有模式匹配算法的基础上,针对入侵检测的特点,对一种允许插入模式匹配算法——计数过滤法进行了深入研究,提出了改进的计数过滤法。算法利用末位字符跳过尽可能多的字符,同时根据文本窗中字符的有序性,最大限度地减少验证次数,加快匹配速度,从而提高入侵检测的效率。

关键词:入侵检测;模式匹配;计数过滤法;末位字符
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2006)07-0150-03

An Improved Algorithm for Pattern Matching and Its Application in Intrusion Detection System

GAN Xue-shi, SUN Li-juan
(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: String matching algorithms are very important analyzing methods in intrusion detection. This paper analyzes the existing pattern-matching algorithms, studies deeply on an algorithm named counting fiter allowing errors, and shows an improved one for intrusion detection. This algorithm makes the best of the ultimate character to overlap characters as much as possible, and reduces verifying. All of these speed up the matching to advance the efficiency of intrusion detection.

Key words: intrusion detection; pattern matching; counting filter approach; ultimate character

0 引 言

随着网络规模的日益扩大,网络安全越来越受到人们的重视。由于网络攻击手段的多样性,传统的防火墙技术已经难以单独保障网络的安全。在目前主动防御技术还不够成熟的情况下,入侵检测(Intrusion Detection, ID)[1~4]作为一种更积极的安全防护技术,已成为网络安全领域的研究热点。

入侵检测(ID)可分为异常检测和误用检测。相对于异常检测,误用检测尽管存在规则库大等缺陷,但它尽可能利用了已有入侵手段的记录,通过一定的变异(如允许插入)同样保持了相当的容错能力,并且不存在异常检测中的学习过程以及在此过程中可能造成的学习错误。误用检测主要通过模式匹配[5~9]来实现,有字符串匹配、协议匹配、大小匹配(长度匹配)、累积匹配、逻辑匹配等。一般模式匹配指的是字符串匹配。

IDS的模式匹配即通过对数据包的分析,并匹配自身的规则库,若能够匹配则产生事件报警或者保留更多相关信息的分析,否则就丢弃(即便属于攻击)。在IDS中,通常采用规则链的形式进行网络数据包扫描,如Snort[10]中的"规则头-规则选项"。在规则头及规则选项其它关键字缩小范围的前提下,将规则选项中的内容(content)关键字值作为模式串,对数据包负载进行检测。

字符串匹配按其精确性可以分为精确匹配和模糊匹配。常见的精确匹配算法有BF算法、KMP算法、KR算法、BM算法以及由此衍生出来的各种多模式匹配算法。为了保持一定的容错能力,扩大误用检测范围,文中着重探讨模糊匹配。其中,一类相对简单的匹配问题就是 k 不同问题,通常,特指 k 插入问题,即允许某段文本相对于模式多了 k 个插入字符。

文中针对入侵检测中允许插入问题的特点,提出了一种改进的计数过滤算法,它利用末位字符有效地跳过了大量字符,减少了验证次数,加快了匹配速度。

1 计数过滤算法

计数过滤算法,就是一种允许插入的模糊匹配算法。它是基于这样的原理:模式在文本位置 j 被找到,即在位置 j 发生匹配,则其所有的字母都必定在文本窗中 $T_{j-m-k+1\cdots j}$ 出现。用计数器(counter)记录在任何一个文本位置 j,有多少模式字母出现在文本窗中,在文本中每

收稿日期:2005-11-02
基金项目:江苏省高校自然科学研究基金项目(04KJB520095)
作者简介:甘学士(1981-),男,四川资中人,硕士研究生,研究方向为入侵检测;孙力娟,教授,研究方向为智能优化技术、入侵检测。

移动一个位置,就更新相关信息。需要注意的是,并不能确定模式字母出现的正确顺序,所以这是一个必要条件,如果文本窗中对应的字母数目不够,就认为它不能满足一个匹配而过滤掉。

通常通过检查一个 $m+k$ 字符长的窗口来过滤。过滤器记录了 P 中有多少字符存在于当前文本窗中(多个也一样)。假设在给定文本位置 j,若 P 有 m 个字符在窗口 $T_{j-m-k+1\cdots j}$ 中,窗口域另行核查。

具体做法是:建立一个表 $A[]$,初始化为每个 $c\in\sum$ 在 P 中出现的次数。在算法中,$A[c]$ 反映了 c 在 P 中出现的次数与在当前窗口中出现的次数的不同。只有当 $A[c]$ 为正时,用计数器记录窗口中满足匹配的字符数目。通过添加新字符 T_{j+1},抛弃最后一个字符 $T_{j-m-k+1}$ 来实现窗口的向后移动。为了添加新字符,$A[T_{j+1}]$ 减小,若操作前 $A[T_{j+1}]$ 超过 0,表明该字符存在于 P 中,则计数器增大。为了抛弃旧字符,$A[T_{j-m-k+1}]$ 增大,若操作后 $A[T_{j-m-k+1}]$ 超过 0,说明 P 中存在该字符,则计数器减小。当计数器达到 m,核查当前窗口域。

当 $A[c]$ 为负时,就是说字符 c 在认为它再次归属于模式之前离窗口还有 $-A[c]$ 的时间。例如,在文本"aaaaaaaa"上运行模式"abc",$k=1$,则 $A['a']=-3$,$A['b']=A['c']=1$,计数器值为 2。图 1 演示了一个例子。

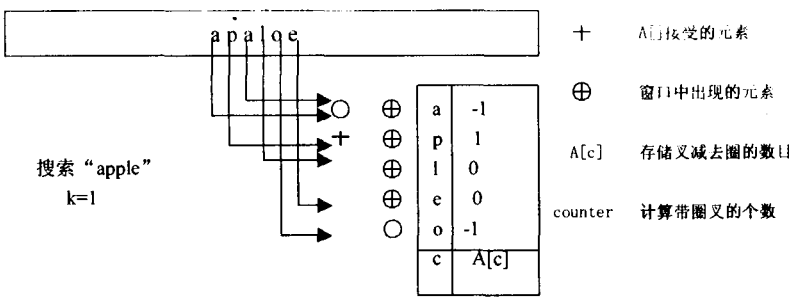

图 1　一个计数过滤的例子

2　改进的计数过滤算法

在计数过滤算法中,影响算法效率的因素主要有两点:首先是窗口移动,必定进行改进算法伪码中 9 到 12 行所有操作;其次是窗口验证,在文献[5]中,采用的是经典的位并行算法,尽管该算法在独立运行时有相当高的效率,但是其预处理时间较长,而且对于一个长为 $m+k$ 的窗口,验证一个长为 m 的模式是否存在,还有更简单的方式。文中针对以上两点提出了算法改进。

改进的计数过滤法伪代码如下:

```
ImpovedCounterFilter(T,n,P,m,k)
//预处理
1) for c∈∑{A[c]←0;A1[c]←0;}
2) for i∈1…m{A[Pi]←A[Pi]+1;}
3) for i∈m…1//寻找模式中末位字符 c 及其位置
4) {if A1[Pi]==0 then{c=Pi;posp=i;}A1[Pi]++;}
5) count←0;
//搜索
6)while j<n //搜索文本中的末位字符
7) {if Tj=c{qindex=j;lindex=j;break;}//qindex 记录第一个符合位置条件的末位字符
8) j++;}
9) for j∈qindex-posp…qindex-posp+m+k //初始化窗口
10) {if A[Tj]>0 then count←count+1;
11) A[Tj]←A[Tj]-1;
12) if Tj=c and j>qindex //Qpc[]记录末位字符下一次出现的位置
13) {Qpc[oindex]=j;lindex=j;}
14) }
15) if count=m{//验证窗口
16) {if(verify Tj-m-k…j-1==1){return 1;}
17) if Qpc[qindex]>qindex{qindex=Qpc[qindex];goto 9);}}//若窗口内存在另外的末位字符直接利用
18) if j>=n return0;goto 5);
```

首先是窗口移动。文中提出了一个新概念——末位字符(ultimate character)。在模式 P 中,若存在字符 c,称 c 出现的最后位置为字符末位,对应的该字符 c 称为末位字符。为了减少处理步骤,仅记录了一个末位字符,并且为了尽可能多地跳过字符,记录的是最早出现的末位字符,称为最前末位字符(first ultimate character)。例如模式"abcadb",末位字符(位置)有 a(4),b(6),c(3),d(5),而最前末位字符为 c(3)。

在算法运行时,其实质是多了个判断条件,即文本窗中是否满足末位字符存在条件。记最前末位字符为 c,在模式中的位置为 posp,模式长度为 m,允许插入数为 k,当前搜索位置为 j,文本中字符 c 出现的位置 pc 需要满足的条件形式化为:

$$m-\text{posp}\leqslant j-\text{pc}\leqslant m-\text{posp}+k \tag{1}$$

式(1)确定了 pc 的一个验证窗,当且仅当在这个验证窗中存在至少一个字符 c 时,模式才有可能发生匹配。

注意到在文本窗中,pc 之后仍然有可能存在字符 c,为了提高速度,算法放弃了链表结构,而采用 HASH 表来记录后面进入文本窗的字符 c 的位置。

当式(1)得到满足时,算法伪码中第 16 行的验证才进行,否则 pc 赋值为 HASH 表中记录的文本窗中字符 c 出现的下一个位置,并直到再次满足式(1)时才有可能进行验证。

若 pc 后文本窗中不存在字符 c,则文本窗继续往后移动,直到出现字符 c 并且进入 pc 验证窗口。很明显,这个移动减少了原算法的处理次数并减少了验证次数。这点在实验中得到验证。

对于文本窗的验证。注意到这是在一个长为 $m+k$ 的串中寻找一个长为 m 的模式串,中间允许插入。一般来

说，k 值并不大，所以，利用串中字符的有序性，可以很好地提高验证效率。记当前文本为 buffer，触发验证的位置为 mindex，伪码如下：

```
int jj = mindex - m - k;
int ii = 0;
while((mindex - jj >= m - ii - 1) & &(m > ii))
{
  if(buffer[jj] ! = p[ii])
    jj++
  else
    {jj++;ii++;}
  if(ii > m - 1)return1;//匹配成功
}
```

3 实验及结果分析

首先考察文本长度与搜索时间的关系，很明显，两者是线性相关的，以模式长为 3 为例，结果如图 2 所示。

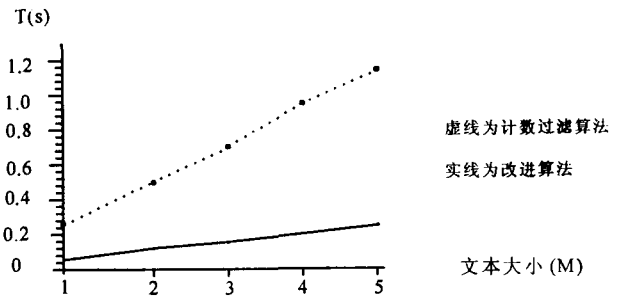

图 2 文本长度与搜索时间关系

再考察模式长度与搜索时间的关系。用随机产生的模式去扫描 1M 的随机文本，并记录文本发生匹配的次数及总的扫描时间。所有的实验都进行了 10 次，取平均值，实验结果如图 3 所示。

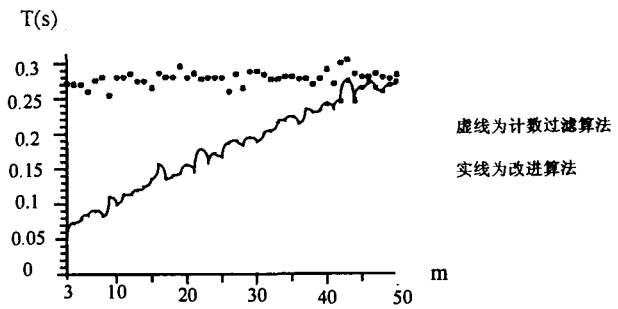

图 3 模式长度与搜索时间关系

从图中可以看出，模式长度 m 对计数过滤算法影响不大，而对改进算法却影响显著。随着 m 的增大，改进的算法的搜索时间不断增加，并无限趋近于计数过滤算法所耗时间。似乎改进算法存在很大缺陷，但是应该注意到一个事实，在入侵检测中，模式并不会太长，以 Snort 为例，有 95% 以上的 content 关键字值（即模式匹配中的模式）长度小于 30，其中又有超过 70% 的长度小于 20。也就是说，在入侵检测中，改进算法的效率基本上都是原算法的 2～4 倍。对于短模式，尤其高效。

另一方面，文本窗验证次数也大大减少，对于短模式尤其明显，例如长为 3 的模式，验证次数由平均 1387 降至 377。总体上，改进算法验证次数是原算法的 1/5～1/3。

同多模式算法相比较，在模式长度小于 20 的情况下，改进算法效率与之是非常接近的。在实际的入侵检测系统中，在规则头及除 content 关键字外的其它规则选项关键字的约束下，相同的规则并不是太多，也就是说，多模式匹配目前还存在一些难题急待解决，高效的单模式匹配算法是其必要的补充。文中提出的计数过滤改进算法就是一种很好的选择。

4 结束语

文中介绍了针对入侵检测中允许插入问题的模式匹配算法——计数过滤法，提出了一种改进的计数过滤算法，算法中定义了一个新概念——末位字符，并利用它有效地跳过了大量字符，减少了验证次数，同时根据文本窗中字符的有序性，最大限度地减少了验证步骤，使得改进算法大大加快了模式匹配速度，提高了入侵检测的效率。

参考文献：

[1] James P. Anderson Co. Computer Security Threat Monitoring and Surveillance[Z]. Fort Washington, PA, 1980.

[2] SANS Institute. Intrusion Detection FAQ, The Internet's most trusted site for vendor neutral intrusion detection information. Version 1.80[EB/OL]. http://www.sans.org/resources/idfaq/, 2003-06-12.

[3] Harmer P K, Williams P D, Gunsch G H, et al. An Artificial Immune System Architecture for Computer Security Applications[J]. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, 2002, 6(3):239-251.

[4] Kim J, Bentley P J. Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection[A]. The Congress on Evolutionary Computation (CEC-2002)[C]. Honolulu: [s.n.], 2002. 1015-1020.

[5] Kuri J, Navarro G, Me L. Fast Multipattern Search Algorithms for Intrusion Detection[A]. Fundamenta Informaticae XXI[C]. [s.l.]:[s.n.], 2001. 1009-1027.

[6] Fredriksson K, Navarro G. Average-optimal Multiple Approximate String Matching[EB/OL]. In Proc. 14th Ann. Symp. on Combinatorial Pattern Matching (CPM'03), LNCS v. 2676, 109-128. http://citeseer.ist.psu.edu/fredriksson03averageoptimal.html, 2003.

[7] Kim J W, Integrating Artificial Immune Algorithms for Intrusion Detection[D]. London: University of London, 2002.

[8] Navarro G. A guided tour to approximate string matching[J]. ACM Computing Surveys, 2001, 33(1):31-88.

[9] Hyyro H, Navarro G. Faster Bit-parallel Approximate String Matching[R]. Santiago: University of Chile, 2002.

[10] Roesch M. Snort Users Manual, Snort Release 1.8[EB/OL]. http://www.snort.org, 2001-07-09.