文章编号:1001 - 9081(2008)01 - 0082 - 03

一种面向入侵检测的快速多模式匹配算法

高朝勤,陈元琰,李 梅
(广西师范大学 计算机科学与信息工程学院,广西 桂林 541004)
(chqgao@126.com)

摘 要:随着网络速度和入侵检测规则的持续增长,模式匹配正在成为网络入侵检测系统的性能瓶颈。提出了一种新的 Wu-Manber 类型的模式匹配算法,通过将模式分组,对不同子模式组采用不同匹配方法,显著提高了模式匹配的效率。对比实验表明,当模式组中含有长度小于 3 的模式时,新算法性能比原算法平均提高了 29% ~44%。

关键词:多模式匹配;入侵检测系统;Wu-Manber 算法;网络安全
中图分类号: TP393.08 **文献标志码**:A

Fast multi-pattern matching algorithm for intrusion detection

GAO Chao-qin, CHEN Yuan-yan, LI Mei
(*College of Computer Science and Information Engineering, Guangxi Normal University, Guilin Guangxi* 541004, *China*)

Abstract: With network speed and the number of rules constantly increasing, pattern matching is becoming the bottleneck in Network Intrusion Detection System (NIDS). This paper proposed a fast Wu-Manber-like multi-pattern matching algorithm for intrusion detection, called FWM. By subdividing the pattern group into two subgroups and dealing with the two subgroups in different methods, the FWM algorithm enhanced the efficiency of pattern matching. Experimental results show that, when pattern group contains the pattern that is less than three bytes, the FWM algorithm improves average performance by 29% ~44% compared to the original NIDS pattern matching algorithm.

Key words: multi-pattern matching; intrusion detection system; Wu-Manber algorithm; network security

网络入侵检测系统(NIDS)作为提高网络系统安全性的重要技术之一,已经得到了广泛应用。典型的 NIDS 是用一系列的规则(特征)来描述已知的攻击行为。一条规则包含了一个或多个用以识别某种攻击行为的模式串。许多 NIDS 都是依靠模式匹配技术来进行入侵检测的。测试表明,花费在模式匹配上的时间占到了整个 NIDS 总处理时间的 30%。对于 Web 密集型流量,这一消耗可以达 80%[1]。随着网络速度和入侵检测规则的持续增长,模式匹配正在成为 NIDS 的性能瓶颈。

本文将模式组中影响 Wu-Manber 算法效率的短模式分离出来,在 Wu-Manber 算法的基础上提出了一种性能更好的适合于入侵检测规则集中模式特殊性的多模式匹配算法。

1 入侵检测中的模式匹配算法

Boyer-Moore 算法[2]是已知平均性能较好的单模式匹配算法。它首先对模式进行预处理,通过坏字符启发和好后缀启发构造两个跳跃表。在查找时把模式串和文本左对齐,从模式的最右字符开始自右向左进行比较。当发现不匹配字符时查询两个跳跃表来快速向右移动模式。对于长度为 m 的模式和长度为 n 的文本,在最坏情况下 Boyer-Moore 算法需要做 $3n$ 次的字符比较,平均只需要 n/m 次的比较。Boyer-Moore-Horspool 算法[3]是 Boyer-Moore 算法的一个更加简单高效的实现。它只使用坏字符启发而不使用好后缀启发,在最坏情况下的字符比较次数为 $n \times m$,但是其平均性能并不比 Boyer-Moore 算法差。

Wu-Manber 算法[4]实际上是对 Boyer-Moore 算法在多模式匹配上的扩展,它考虑大小为 B 的字符块。在预处理阶段,Wu-Manber 算法构造 3 个表:SHIFT 表存放大小为 B 的字符块出现在文本中时模式窗口的跳跃距离;Hash 表存放指向 B 个后缀字符相同的模式列表的指针;PREFIX 表存放模式前缀的哈希值,用以过滤那些后缀相同但前缀不同的模式。在查找阶段,Wu-Manber 算法逐个扫描文本中大小为 B 的字符块,通过 SHIFT 表实现模式跳跃,通过 Hash 表和 PREFIX 表快速确定模式是否和文本匹配。Wu-Manber 算法的时间复杂度为 $O[(B \times n)/m]$,这里 B 为字符块大小,n 为文本串长度,m 为最小模式长度。

文献[5]在 snort 2.6.0 中实现了一个 Wu-Manber 算法的变型 Modified Wu-Manber 算法(简称 MWM)。MWM 使用了一个标准的 1 或 2 字节坏字符跳跃表,一个固定的 2 字节前缀 Hash 表。

值得指出的是,基于 Boyer-Moore 跳跃思想的各种算法,最大优点就是字符比较可以跳跃进行,但是其性能对最小模式长度有很高的依赖性。因为模式的最大跳跃距离不能超过该值,否则就会漏掉可能的匹配。最小模式长度越长,坏字符跳跃越有效,查找速度也就越快。

2 FWM 算法的设计与实现

分析 snort 规则集发现,许多规则组的大部分规则是多字节模式(本文中指长度大于等于 3 字节的模式,下同),但是也包含有 1 或 2 字节模式的规则。这些 1 或 2 字节模式尽管数量少,却导致整个模式组的最大跳跃距离为 1,即没有跳跃,从而大大降低了 Wu-Manber 算法的性能。本文提出了一种新的 Wu-Manber 类型的多模式匹配算法,我们称之为 FWM (Fast Wu-Manber)。FWM 把模式分组,对 1 或 2 字节的模式

收稿日期:2007 - 07 - 16;**修回日期**:2007 - 09 - 06。 **基金项目**:广西自然科学基金资助项目(0728099)。

作者简介:高朝勤(1975 -),男,河南南阳人,硕士研究生,主要研究方向:网络安全、网络入侵检测; 陈元琰(1961 -),男,福建仙游人,副教授,博士,主要研究方向:计算机网络; 李梅(1982 -),女,广西桂林人,硕士研究生,主要研究方向:计算机网络。

采用位图方式匹配,使得剩下模式组的最小模式长度大于2,从而显著地提高了算法性能。

假设模式集为 $P = \{P_1, P_2, \cdots, P_k\}$,其中 P_i 表示长度为 i 的模式组;待查找文本(数据包)为 $T = \{t_1, t_2, \cdots, t_n\}$,长度为 n;m 是多字节模式组中的最小模式长度。

FMW 也包括预处理和查找两个阶段。

2.1 预处理阶段

对于含有 1 或 2 字节模式的模式组,首先把模式组划分为两个子模式组:短模式组 $P_{\text{shorter}} = \{P_1, P_2\}$ 和长模式组 $P_{\text{longer}} = \{P_m, P_{m+1}, \cdots, P_k \mid m > 2\}$。在读入规则文件时,把 1 字节模式、2 字节模式和多字节模式分别添加到相应的模式结构 OneBytePat、TwoBytePat 和 MultiBytePat 中。对于不含 1 或 2 字节的模式组,则不划分该模式组,并且只构造 MultiBytePat。本文只讨论含有 1 或 2 字节模式的模式组。

2.1.1 短模式组的预处理

为了快速检查文本中的一个或两个字符是否与 P_1 或 P_2 中的某个模式匹配,首先对短模式组 P_{shorter} 进行预处理。对每个出现在 P_1 中的字符 *char*,在大小为 256 的存在位图 *EXISTONE* 中把相应位置标记为 1。对每个出现在 P_2 中的字符对 *char*1*char*2,在大小为 256 × 256 的存在位图 *EXISTTWO* 中把相应位置标记为 1。这样,在查找阶段就可以直接根据存在位图中相应位置的值是否为 1 来迅速确定文本中的当前字符是否与 P_1 或 P_2 中的某个模式匹配。*EXISTONE* 和 *EXISTTWO* 可以按照如下方法构造:

$$EXISTONE(char) = \begin{cases} 1, & char \in P_1 \\ 0, & char \notin P_1 \end{cases}$$

$$EXISTTWO(char1, char2) = \begin{cases} 1, & char1char2 \in P_2 \\ 0, & char1char2 \notin P_2 \end{cases}$$

2.1.2 长模式组的预处理

对于长模式组 P_{longer},首先对模式排序,然后为其构造一个坏字符跳跃表 BCSHIFT 和一个前缀哈希表 Hash。

BCSHIFT 表中的值决定了在文本串中出现字符 *char* 时的跳跃距离,其初始化和构造过程如下:

```
/* 初始化坏字符跳跃表 BCSHIFT */
for(i = 0; i < 256; i++)
  BCSHIFT[i] = m;
/* 构造坏字符跳跃表 BCSHIFT */
for(i = 0; i < NumPatterns; i++)
                    //NumPatterns 是 MultiBytePat 中的模式数目
  for(k = 0; k < m; k++) {
    shift = m - k - 1;
    cindex = MultiBytePat[ i ].psPat[ k ];
                                        // psPat 为模式指针
    if (shift < BCSHIFT[ cindex ] )
      BCSHIFT[ cindex ] = shift;
  }
```

Hash 表用于判断文本中当前两个字符是否是 P_{longer} 中某个模式的 2 字节前缀,其初始化和构造过程如下:

```
/* 初始化 Hash 表 */
for (i = 0; i < 65536; i++)        //Hash 表大小为 256 × 256
  Hash[i] = 0;
/* 构造 Hash 表 */
Hash( *S) = (( *S) << 8) | *(S + 1);        // 哈希函数
for ( i = 0; i < NumPatterns; i++) {
  hindex = Hash(MultiBytePat [i].psPat[0]);
  sindex = Hash[ hindex ] = i;
}
```

2.2 查找阶段

查找阶段分为两步,首先用 P_{shorter} 中的模式与文本匹配,然后用 P_{longer} 中的模式与文本匹配。两种情况下都从文本 T 的首字符开始。

2.2.1 短模式组中的模式与文本匹配的算法过程

1) 以文本中当前字符 t_c 为索引,检查 $EXISTONE(t_c)$。

2) 以文本中当前 2 个字符 t_c, t_{c+1} 为索引,检查 $EXISTTWO(t_c, t_{c+1})$。

3) 如果 $EXISTONE(t_c)$ 和 $EXISTTWO(t_c, t_{c+1})$ 的值都为 0,文本指针加 1,转第 1 步。

4) 如果当前模式区分大小写,检查在区分大小写情况下是否匹配。如果不匹配,文本指针加 1,转第 1 步。

5) 报告匹配,结束查找。

2.2.2 长模式组中的模式与文本匹配的算法过程

1) 以文本中当前字符 t_c 为索引,查找 $BCSHIFT(t_c)$。

2) 如果 $BCSHIFT(t_c)$ 的值大于 0,文本指针右移 $BCSHIFT(t_c)$ 个字符,转第 1 步。

3) 计算文本中当前 2 个字符 t_c, t_{c+1} 的 Hash 函数值,以 $Hash(t_c)$ 为索引查找 Hash 表,若其值为 0,文本指针加 1,转第 1 步。

4) 检查与当前模式有相同前缀的所有模式。如果没有模式与当前文本匹配,文本指针加 1,转第 1 步。

5) 如果当前模式区分大小写,检查在区分大小写情况下是否匹配。如果不匹配,文本指针加 1,转第 1 步。

6) 报告匹配,结束查找。

与通常的字符串查找不同,NIDS 中的模式匹配算法在检查每一个数据包时,只要确定当前字符串与模式组中的某个模式匹配,则报告该数据包可能含有攻击,不再检查该数据包中剩余字符,而立即开始检查下一个数据包。与长模式相比,短模式被匹配的可能更大一些,所以 FWM 算法首先用短模式组与数据包匹配,再用长模式组与数据包匹配。一旦发现某个数据包中包含与模式匹配的字符串,则直接转入对下一个数据包的检查。由于 FWM 算法把短模式从模式组中分离出来先作处理,长模式组的最小模式长度大于 2,也即最大跳跃距离至少为 3,从而显著地缩短了每个数据包的检查时间,提高了模式匹配算法的性能。

3 实验结果

3.1 实验环境

为了检验新算法的性能,将 FWM 算法和 snort 中实现的 MWM 算法[5]进行了实验对比。实验环境采用 Pentium4 的 PC 计算机作为硬件平台,CPU 为 2.0 GHz,内存为 256 MB;操作系统为 Linux(内核版本 2.6.18);编译器为 gcc4.1.1,在编译时使用了相同的优化选项 -O2,在 snort2.6.0 版本下进行实验。实验用到的规则集是 snort 官方网站 2007 年 5 月 8 日发布的规则集,入侵检测数据为 MIT Lincoln 实验室的 1999 DARPA 入侵检测评价数据集[6]。实验结果中的所有时间数据都是通过 Linux 下的“time”命令测量 10 次取平均值得到的。

3.2 算法性能分析

图 1 中所用的规则集都包含长度为 1 和 2 的模式,数据包大小为 32 MB。从图 1 可以看出,FWM 算法的执行时间明显低于 MWM 算法。

图 2 中所用的规则集是包含长度为 1 和 2 的模式的1 000

条规则,IDEVAL1 ~ IDEVAL5 为分别从 MIT 1999 DARPA 数据集5周的真实流量中提取的大小为32 MB 的数据包。从图2可以看出,在检测不同的真实流量时,FWM 算法的性能均比 MWM 算法有不同程度的提高。

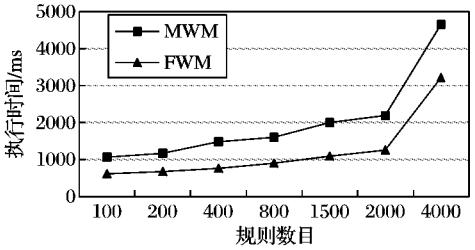

图1 规则集大小不同时的性能比较

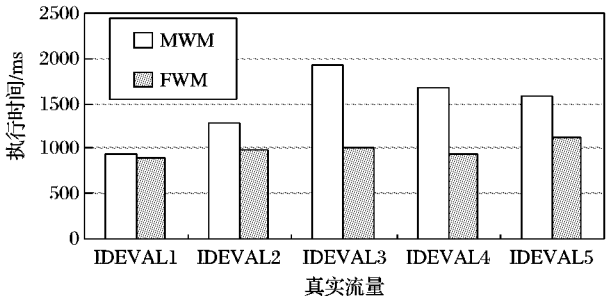

图2 真实流量不同时的性能比较

图3中所用的规则集大小都为1 000 条,数据包大小为32 MB。从图3可以看出,在最小模式长度为1和2时,FWM 算法的执行时间比 MWM 算法有很大下降;当最小模式长度大于2时,两个算法的性能几乎一样。

4 结语

考虑到入侵检测规则集中含有长度为1和2字节的模式,通过将模式分组,将模式组中影响 Wu-Manber 算法效率的短模式分离出来,对不同子模式组采用不同的匹配方法,对短模式采用存在位图的匹配方法,对长模式使用改进的 Wu-Manber算法。对比实验表明,当模式组中含有长度小于3的模式时,新算法性能比原算法平均提高了29% ~44%。

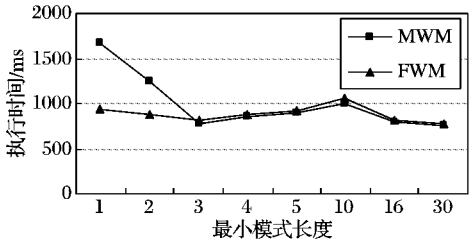

图3 最小模式长度不同时的性能比较

参考文献:

[1] FISK M, VARGHESE G. An analysis of fast string matching applied to content-based forwarding and intrusion detection, CS2001-0670 (updated version)[R]. San Diego: University of California, 2002.

[2] BOYER R S, MOORE J S. A fast string searching algorithm [J]. Communications of the ACM, 1977, 20(10):762 - 772.

[3] HORSPOOL R N. Practical fast searching in strings [J]. Software-Practice and Experience, 1980, 10(6):501 - 506.

[4] WU S, MANBER U. A fast algorithm for multi-pattern searching, TR-94-17[R]. Tucson: University of Arizona, 1994.

[5] Snort2.6.0 [EB/OL]. [2006 - 12 - 05]. http://www.snort.org/dl.

[6] 1999 DARPA intrusion detection evaluation data set [DB/OL]. [2007 - 04 - 09]. http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html.

(上接第81页)

filter(request,OP):过滤无效的访问请求,其中 OP 为资源方所允许的操作权限集。

meet(Cred,P):判断证书(集)Cred 是否满足策略 P。

encrypt(msg,sec):消息加密函数,使用密钥 sec 对消息 msg 进行加密。

decrypt(msg,sec):与 encrypt(msg,sec)相对应的解密函数。

2 RBAM 分析

通过上述对 RBAM 的分析可知,RBAM 具有如下特征:

1)分解访问控制策略,减少敏感策略的暴露机会,切实保护策略中的敏感信息。针对访问控制策略敏感信息暴露与保护的矛盾,RBAM 将访问控制策略分为通用策略与敏感策略两部分,用户只有满足了通用策略,方可与敏感策略进一步协商,从而降低了敏感信息泄露的可能性。

2)引入 Agent 技术完成访问控制策略的实施。在 RBAM 中,使用本地规则 Agent 执行通用策略,过滤掉非法的访问请求,避免资源方遭受拒绝服务等的攻击;使用敏感策略 Agent 与符合通用策略的用户进行进一步协商。

3)简化协商过程,避免了无效的协商,提高了协商效率与成功率。策略的分析可将一些无效的访问请求与攻击拒之门外,真正的协商过程在敏感策略 Agent 处进行,这样减少了协商次数,降低了协商复杂性,从而起到提高协商效率与成功率的作用。

3 结语

自动信任协商为开放环境中用户资源共享提供了一种方法,访问控制策略则规范了合法用户对资源的操作,防止非授权用户对资源的攻击。针对访问控制策略敏感信息保护的问题,本文提出一种基于规则的自动信任协商模型 RBAM。RBAM 将访问控制策略进行分解,并引入 Agent,使用 Agent 技术来实现规则的约束。用户只有通过了本地策略,方可与资源方进行协商,减少了敏感策略暴露次数,避免了无效的协商,提高了协商效率与成功率。

参考文献:

[1] WINSBOROUGH W H, LI N H. Towards practical automated trust negotiation[C]// MICHAEL J B. Proceeding of the 3rd International Workshop on Policies for Distributed Systems and Networks. Washington: IEEE Computer Society Press, 2002:92 - 103.

[2] 廖振松, 金海, 李赤松, 等. 自动信任协商及其发展趋势[J]. 软件学报,2006,17(9):1933 - 1948.

[3] SANDHU R S, COYNE E J, FEINSTEIN H L, *et al*. Role-based access control models[J]. IEEE Computer, 1996,29(2):38 - 47.

[4] 李立新, 陈伟民, 黄尚廉. 强制访问控制在基于角色的安全系统中的实现[J]. 软件学报, 2000, 11(10): 1320 - 1325.

[5] SHIN H J, LEE S G. Architecture environments for e-business Agent based on security [C]// ICCSA 2004, LNCS 3043. Berlin: Springer-Verlag, 2004: 625 - 634.