ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.5,No.3,January 2009, pp.556-559

E-mail: info@cccc.net.cn
http://www.dnzs.net.cn
Tel:+86-551-5690963 5690964

基于 Snort 的模式匹配算法研究

刘影[1,2],张维勇[1]

(1.合肥工业大学 计算机与信息学院,安徽 合肥 233009;2.安徽电子信息职业技术学院 软件学院,安徽 蚌埠 233000)

摘要:随着网络的迅速发展,网络安全问题日益突出,入侵检测技术的应用越来越广泛,对 Snort 入侵检测系统来说,模式匹配算法仍是其使用最多的基本算法,模式匹配算法的效率直接影响到入侵检测系统的性能。该文介绍了 KMP 和 BM 算法,并对其进行了比较,并对 BM 算法的进行了改进,提高了模式匹配的速度。

关键词:入侵检测技术;KMP 算法;BM 算法

中图分类号:TP393 **文献标识码**:A **文章编号**:1009-3044(2009)03-0556-04

Research of the Pattern-matching Algorithm Based on Snort

LIU Ying[1,2], ZHANG Wei-yong[1]

(1.School of Computer & Information, Heifei University of Technology,Heifei 230009, China; 2.Software College, Anhui Vocational College of Electronics & Information, Bengbu 233000, China)

Abstract: With the rapid development of the network,Network Security is becoming increasingly prominent,and Intrusion Detection technology more widely.The Snort Intrusion Detection System,the pattern-matching algorithms is still using its most basic algorithm,the efficiency of the pattern-matching algorithms have a direct impact to the intrusion detection system performance.The KMP and BM algorithms are introduced in this paper.Research is carried out to improve the BM algorithm and a better BM algorithm is proposed which can improve the pattern-matching speed.

Key words: intrusion detection system(IDS); KBM algorithm; BM algorithm

1 入侵检测技术

随着计算机及网络技术的发展,计算机网络应用已渗透到社会生活的各个领域,各种新兴的网络应用层出不穷,如网上银行、网上购物、电子政务、企业信息化等。且近年来,我国网民规模继续呈现持续快速发展的趋势,网民数量已跃居世界第一位[1]。据统计结果显示:全球范围内平均每 20 秒就发生一次网络人侵,且 80%以上的入侵行为来自网络内部。许多著名的网络如 Yahoo 等都遭受过网络攻击,直接造成了很大的经济损失。因此研究计算机网络的安全技术特别是对内部入侵行为的检测技术,便具有更重要的现实意义。入侵检测技术是一种积极主动的安全防护技术,提供了对内部攻击、外部攻击和误操作的实时保护,在网络系统受到危害之前拦截和响应入侵。从某种意义上说是防火墙的补充。

入侵检测根据检测原理可以分为异常检测和误用检测两类。

异常检测需要建立目标系统及其用户的正常活动模型,然后基于这个模型对系统和用户的实际活动进行审计,以判定用户的行为是否对系统构成威胁。异常检测的方法有神经网络、机器学习和人工免疫等。异常检测的优点是它不需要有系统缺陷的知识,且具有较强的适应性和通用性。

误用检测是对利用已知的系统缺陷和已知的入侵方法进行入侵活动的检测。运用已知攻击方法,根据已定义好的入侵模式,通过判断这些入侵模式是否出现来检测。由于依据具体特征库进行判断,所以检测准确度很高,并且因为检测结果有明确的参照,也为系统管理员做出相应措施提供了方便,可以有针对性的建立高效的入侵检测系统。

2 Snort 入侵检测系统

2.1 概述

Snort 是一个用 C 语言编写的开放源代码的轻量级网络入侵检测系统(NIDS),所谓轻量级是指在检测时尽可能少地影响网络的正常操作。它采用误用检测的检测方法,使用已知攻击的规则的定义来检查网络入侵问题。具有实时数据流量分析和日志 IP 网络数据包的能力,能够进行协议分析,对内容进行搜索/匹配。它能够检测各种不同的攻击方式,对攻击进行实时报警。由于其源代码的开放性,任何一个程序员都可以合法的都其进行功能的添加、修改错误并进行传播。所以 Snort 代码极为简洁、短小,但功能强大,具备跨系统平台操作等特征,并且允许管理员在短时间内通过修改配置进行实时的安全响应。相较于商业软件许多都是昂贵而庞大的系统,使得基于 Snort 的免费的入侵检测系统使用得非常广泛。

Snort 有三种主要的工作模式:嗅探器(Sniffer)模式、包记录器(Packet Logger)模式和网络人侵检测系统(NetWork Intrusion Detection System, NIDS)模式。

·嗅探器模式仅仅是从网络上读取数据包并作为连续不断的流显示在终端上。

·数据包记录器模式把数据包记录到硬盘上。

·网络人侵检测模式有最复杂的结构,同时它的配置也是最多样化的。通过 Snort 分析网络数据流以匹配用户定义的一些规则,并根据检测结果采取一定的措施。

2.2 Snort 系统结构

Snort 包含很多可配置的内部组件,它们对误报、漏报以及抓包和记录日志等性能都有很大影响。Snort 分成五个主要的组件:捕

收稿日期:2008-12-16
作者简介:刘影(1981-),安徽砀山人,安徽电子信息职业技术学院教师,合肥工业大学研究生在读。

包装置、包解码器、预处理程序、检测引擎和输出插件[2,6]。

1) 包捕获

Snort 通过两种机制来满足网络流量的需要:将网卡设置为混杂模式和利用外部的捕包程序库 libpcap(Windows 环境下可以使用 Winpcap)。利用 libpcap 独立的从物理链路上捕获原始包,原始包是由客户端到服务器传输时未被修改的保持原始状态的数据包,它所有的协议头信息都保持完整,未被操作系统更改。当包被以原始的状态收集到以后,送给包解码器。

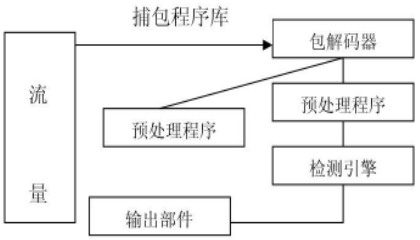

图 1 Snort 数据流程示意图

2) 包解码器

包解码器需要对获得的数据链路层的原始数据包进行解码。Snort 能够识别 IP、TCP 和 UDP 等高层协议。包解码器实际上是一系列对具体协议元素逐一进行解码的解码器。将特殊的协议元素翻译成内部的数据结构。它建立网络堆栈,从较低层次的数据连接协议开始,逐层上移,对每一个协议元素进行解码。在包通过各种协议的解码器时,解码后的包数据一被存入数据据构中,就会迅速被送到预处理程序和检测引擎进行分析。

3) 预处理器插件

Snort 捕获的数据包,经过包解码器处理以后,由预处理器插件提供规网络流量的处理。Snort 预处理程序可以用来针对可疑行为检查包或者修改包,以便检测引擎能对其进行正确解释,从而提高检测的准确性和速度。Snort 使包通过每一个预处理程序来检查发现攻击,每个预处理检测数据包并确定需要进行什么样的操作(注意、报警或修改)。

4) 检测引擎

检测引擎是 Snort 的一个主要部件。它有两个主要功能:规则分析和特征检测。检测引擎通过分析 Snort 规则来建立攻击特征。对数据包进行检测时,通过对各种规则文件中的不同选项与每个包的特征和信息进行逐一、简单的检测,一旦发现数据包中的内容和某条规则相匹配,就通知报警模块。检测引擎根据需要规定和组织的先后次序,将流量与规则按其载入内存的顺序一次进行匹配。规则按功能分两个部分:规则头(规则树节点)和规则选项(选项树节点)。规则头包含特征应用的条件信息,在规则头规定协议,源 IP、目的 IP 地址范围,端口和日志类型。检测引擎对规则头和规则选项进行不同处理。

5) 输出插件

Snort 的输出插件接收通过检测引擎、预处理和解码引擎传来的入侵数据的报警信息。它的目的是将报警数据转储到另一种资源或文件中。各种输出插件用于各种不同的功能。Snort 的输出功能也是模块化和插件式的,它能将任何原始二进制 tcpdump 输出转换成多种关系数据库输出,从 MySQL 到 Oracle 甚至微软的 SQL Server。Snort 可以以各种格式记录日志以便入侵数据能方便地为其他应用程序或者工具使用,如:Syslog (系统日志文件)、Text logfile (关系数据库)和 Snort Unified (Snort 统一格式)。

3 入侵检测的模式匹配算法

误用检测技术是基于特征匹配检测,而特征匹配检测算法(字符串搜索算法)是入侵检测系统检测引擎的核心,它的效率直接决定了这类入侵检测系统的性能。因而如何改进字符串匹配搜索算法提高检测速度,是目前 IDS 研究的重点之一。字符串搜索算法有很多,其中较为常用的是 KMP(Knuth Morris Pratt)算法和 BM(Boyer Moore)算法等。

3.1 KMP 算法

KMP(Knuth Morris Pratt)算法是由 D.E.Knuth 与 V.R.Pratt 和 J.H.Morris 3 人于 1977 年提出,是一种改进的字符串匹配算法。它的核心思想是:在主字符串中寻找模式子串的过程中,若发生字符串不匹配的情况下,减少不必要的会溯,利用已经得到的"部分匹配"结果将模式串右移尽可能远的距离,然后继续与主串字符进行比较,直到找到该字串或未找到匹配的模式子串结束查找。需要注意的是,该算法主要是考虑了模式子串本身的特性,将模式串一次右移多个字符的位置,右移后可以从模式串起点后的某处开始重新匹配字符。

如:在模式主串(记做 s)搜索模式子串(记做 t),将 s 的第一个字符记做 s1,t 的第一个字符记做 t1。 在主串 s 中匹配子串 t 的过程如下

主字符串:ababcdabcaabcaabab 模式子串: abcaabab

1) 计算模式子串的 next 函数值。 假设子串为 ´ $p_1\ p_2\ p_3\ p_4 \cdots\cdots p_m$ ´

$$next[j]=\begin{cases} -1 & j=0 \\ k=Max\{\ k\ |\ 1<k<j\ \&\&\ 'p_1 \ldots p_{k-1}'='p_{j-k+1} \ldots p_{j-1}'\ \} \\ 0 & \text{其他情况} \end{cases}$$

表 1 为模式子串各字符 next 值。

2) 从左至右匹配字符

表 1 模式子串匹配过程

j	1	2	3	4	5	6	7	8
模式子串	a	b	c	a	a	b	a	b
next[j]	-1	0	0	0	1	1	2	1

① 首先以指针 i 和 j 分别指向模式主串 s 和模式子串 t 的第一个字符 s1 和 t1,将两者个字符进行匹配。

② 若在匹配时,主串指针 i 指向的字符和模式子串的指针指向的字符相同,则 i 和 j 分别增 1,接着比较后面的字符,若不相同时,则 i 不变,则 j 退到 next[j]位置。

③ 当 i 不变,则 j 退到 next[j]位置,仍将两者指向的字符继续比较,若相等,接着比较后面的字符,不等则 j 退到下一个 next[j]值对应的位置,依此类推。假如在 j 退到 next[j]处时第一个字符不匹配,则 i 增 1,从主串的下一个字符起和模式子串重新开始匹配。

```
#define MaxSize 100
typedef struct
{ char c[MaxSize];
```

```
int len;
}Sclx;
void SNext(Sclx t,int next[])/* 由模式串 t 求出 next 值 */
{ int i=0;k=-1;next[0]=-1;
while(i<t.len-1) {
if(k==-1||t.c[i]==t.c[k])
{   i++; k++;
next[i]=k;  }
else k=next[k];}
}
int KMPsf(Sclx s, Sclx t)/*KMP 算法 */
{   int next[MaxSize];
int i=0,j=0;   int wz;
SNext(t,next);
while(i<s.len && j<t.len)
{  if(j==-1||s.c[i]==t.c[j])
{   i++;    j++;   }
else j=next[j];
}   if(j>=t.len)
wz=i-t.len;
else   wz=-1;
return wz;
}
```

表 2　模式子串匹配过程

KMP 算法	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
主字符串	a	b	a	b	c	d	a	b	c	a	a	b	c	a	a	b	a	b
1	a	b	**c**	a	a	b	a	b										
2			a	b	c	**a**	a	b	a	b								
3						**a**	b	c	a	a	b	a	b					
4							a	b	c	a	a	b	**a**	b				
5											a	b	c	a	a	b	a	b

说明：当第一次比较时，i 指向主字符串 s 的首字母‘a’，j 指向模式子串 t 的首字母‘a’，两者相同，继续向后比较，直到 i=3,j=3 时，‘a’与‘c’比较不同，将 i 值不变，将模式子串中 j 的指向改为指向 next[j]对应的字符，根据表 1，得 next[3]=0,则 j 指向模式子串第一个字符，重新开始下一轮的比较。表中显示的字符串在比较时，当出现不相同的字符（粗体标示），根据 next 表决定移动的距离。在本次匹配过程中比较的次数为 21 次。

3.2 BM 算法

BM（Boyer-Moore）**算法是一种常用的精确匹配算法，其中著名的开放源程序代码的入侵检测系统** Snort **就是使用** BM **算法来进行特征匹配检测。据统计：**snort **在进行检测的时候调用字符串匹配函数所需用的时间占检测总时间的** 25.2%，**所以字符串匹配算法的效率对** snort **来说很重要。**BM **算法利用跳过不必要的比较来减少模式字符与主字符串中字符的比较次数来提高匹配效率。**

如：在主字符串(记做 P)**搜索的文本记做** T，**将** P **的第一个字符记做** P1，P **的最后一个字符记做** Pm；T **的第一个字符记做** T1，T **的最后一个字符记做** Tn；**当** Pm=Tn **时，在** T **中搜索** P **成功。主字符串** P：ababcdabcaabcaabab　**模式子串** T：abcaabab

为了在字符匹配不成功时，进行移动，根据主字符串 p，**定义一个函数** BMWZ：x→{1，2，…，m}。**函数给出了在模式字符串中可能出现的字符在模式子串中的位置。**

$$BMWZ\ [x]=\begin{cases} m & \text{若任意字符x不出现在P中或者x=}P_i\ (i=m) \\ m-i & \text{若 x 在 P，这里的 } i=\max\{i;\ P_i=x,\ 1\leqslant i\leqslant m-1\} \end{cases}$$

1) **匹配自右向左进行**

首先在搜索时，先把 P **的左部和** T **的左部对齐，而匹配搜索从** P **的最右边一个字符开始。**

2) **忽略不匹配字符**

当 Pm **与** Tm **匹配时，若匹配成功，则向前移动匹配前一个字符** Pm-1 **与** Tm-1 **是否匹配。假如当匹配** Pi **与** Ti **时不成功，且** Ti **不出现在要搜索的模式字符串** p **中，则** p **整个字符串向右移地** m **个字符的距离，；如果发现在** T **中包含该字符，则根据该字符所处位置，计算所需要的移动的字符长度，然后将** p **向右移动相应的距离。即移动的距离由函数** BMWZ **中对应的函数值决定。**

3) **发现匹配字符**

当在搜索文本 T **中查找到匹配的** p **字符串后，就结束检查。**

```
void BMsz(SqString t, int BMGJ[])
{ int k=0;
for(k=minchars;k<=maxchars;k++)
BMGJ[k]=t.len;
for(k=0;k<t.len-1;k++)
BMGJ[t.c[k]]=t.len-k+1;}
int BMsf (SqString p,SqString t,int BMGJ[])
{  int x=t.len;  int y=p.len;
int k=x-1; int i,j;
if(x>y)
return -1;
while(k<y)
{  j=x-1; i=k;
while(j>=0 && p.c[i]==t.c[j])
{j- -;i--; }
if(j= =-1)
return i+1;
```

```
if (k>i+BMGJ[p.c[i]])
k+=BMGJ[p.c[i]];
else k=i+BMGJ[p.c[i]];}
return -1;
}
```

表 3 BM 算法匹配过程

BM 算法	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
正文串	a	b	a	b	c	d	a	b	c	a	a	b	c	a	a	b	a	b
1	a	b	c	a	a	**b**	a	b										
2							a	b	c	a	a	b	a	**b**				
3								a	b	c	a	a	b	a	**b**			
4									a	b	c	a	a	**b**	a	b		
5										a	b	c	a	a	b	a	**b**	
6											a	b	c	a	a	b	a	b

说明:使用 BM 算法匹配字符串时,从右边的字符开始比较,第一趟时,当比较至‘d’和‘b’元素时,不相同(不同时将模式子串中字符粗体显示)且‘d’不在模式子串中,移动的距离为模式子串的长度即 8。继续按规则匹配,在第二趟时,不同时主串的中的‘a’字符包含在模式子串中,因此找到该字符在模式串中最大的位置决定移动的位置即移动长度为 8-7。按此规则依次比较各字符,直到找到模式子串。在此次匹配过程中,比较的次数为 17 次。与 KMP 算法比较,若以匹配次数来说,BM 算法的匹配速度较快。

3.3 BM 算法的改进

改进思路:利用已比较字符串收集信息,使模式字符串尽量移动最大的距离。

BM 算法在匹配过程中,如果考虑从下一个字节开始进行匹配,使得仅在首次调用字符匹配的过程中,所产生的偏移量每次将比原算法的偏移量最少可以多移动 2 个字符的位置,而单个字符匹配的执行仅仅增加一次[4]。且 BM 算法在匹配时主要利用了主字符串的信息,对模式子串的特点使用得很少,可以充分考虑利用模式子串的内容信息扩大偏移量。因此考虑从以上方面对 BM 算法进行优化处理。

字符串的匹配仍从右侧开始,当比较到的字符不相同时,根据规则信息模式字符串要发生相应的移动,进行下次的匹配。在确定移动的距离时,充分考虑对模式主串中下一个字符的利用和对模式串本身的结构的特点的利用。如果某个字符发生不匹配的时候,查找模式主串中,与模式子串位置对应的后一个字符,查找其在模式子串中的位置,如果不在,则模式子串移动的距离为偏移模式子串的长度加 1。此种查找方式需要确定在就需要在当前匹配字符串的后面有字符时方可使用,若无字符,则采用原有规则。若在模式子串中,则找到模式子串中相应的字符位置对齐。如果在匹配过程中,发生不匹配的字符同时右边的字符已经有部分匹配的情况发生,则考虑已匹配的字符中的 next 属性,查看在已匹配的字符中在模式字符串前面的字符是否相同,如果不同,采用匹配后一个字符的移动结果,若有相同,则由此确定移动距离,并与上一种匹配方法的移动距离想比较,采用移动距离较大的作为最终模式子串移动的距离。从而使匹配不相符时,模式字符串发生最大的偏移量。

表 4 模式子串匹配过程

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
正文串	a	b	a	b	c	d	a	b	c	a	a	b	c	a	a	b	a	b
1	a	b	c	a	a	**b**	a	b										
2									a	b	c	a	a	**b**	a	b		
3											a	b	c	a	a	b	a	b

4 总结

入侵检测是一种积极主动的安全防护技术,是网络安全的最后一道防线,防火墙的有力补充,提供了对内部攻击、外部攻击和误操作的实时保护功能。随着 Snort 系统的广泛应用及计算机网络速度的不断提高,入侵检测中最核心的模式检测算法的效率也亟待提高,对常用的 KMP、BM 算法进行了应用和比较,BM 的检测速度要高于 KMP 算法,并通过对 BM 算法的改进进一步提高了检测的效率。由于基于模式匹配的入侵检测不能满足日益多样化与新的入侵行为的检测,因此研究更加快速、高效的匹配算法是很有必要的。

参考文献:

[1] 中国互联网络信息中心.第二十二次中国互联网络发展状况统计报告[R/OL].(2008-07).http://www.cnnic.net.cn/uploadfiles/pdf/2008/7/23/170516.pdf.
[2] Koziol J.Snort 入侵检测实用解决方案[M].吴溥峰,译.北京:机械工业出版社,2005.
[3] 余冬梅.基于改进的 BM 算法在 IDS 中的实现[J].甘肃工业大学学报,2003.
[4] 兰景英,王永恒.Snort 研究及 BM 算法改进[J].计算机工程与设计,2008(5).
[5] 吴国伟,毕玲,汪世义.一种快速 Snort 入侵检测系统研究[J].大连理工大学学报,2005(S1).
[6] 刘东远,李昊松.基于 Snort 的校园网入侵检测系统的规划和设计[J].佛山科学技术学院学报:自然科学版,2008(5).