文章编号: 1006-2475(2009)02-0102-03

Snort检测引擎的分析与改进

周 巍,江泽涛,李克伟

(南昌航空大学计算机学院,江西 南昌 330063)

摘要:介绍了 Snort入侵检测系统的规则树和检测引擎的工作原理,分析了几种常用的模式匹配算法,并对其中的 AC-BM 算法进行了改进,通过实验证明该算法是快速高效的。

关键词: Snort; 入侵规则树; 模式匹配

中图分类号: TP393.08 **文献标识码**: A

Analysis and Improvement of Snort Detection Engine

ZHOU Wei, JIANG Ze-tao, LI Ke-wei

(School of Computer, Nanchang Hangkong University, Nanchang 330063, China)

Abstract: The paper introduces the rule tree and the detection engine mechanism of Snort, analyzes the main pattern matching algorithms, presents an improved AC-BM algorithm. Experimental results verify its efficiency and fastness

Key words: Snort; intrusion rule tree; pattern matching

0 引 言

Snort作为一种开源的网络入侵检测系统,当它一出现,就得到众多网络安全人士的追捧,并得到不断的完善。它通过网卡截获网络中原始的传输数据,分析捕获的数据包,匹配入侵行为的特征检测异常行为,进而发现诸如缓冲区溢出、秘密端口扫描、CGI攻击、SMB嗅探、拇纹采集尝试等大量的攻击和探测等的攻击。

检测引擎的设计是提高入侵检测系统速度的关键技术。通过匹配算法完成数据包与规则库比对发现入侵行为。因此,模式匹配算法的性能直接影响检测引擎的检测效率,以下将分析 Snort规则树和几种匹配算法,并给出对 AC-BM算法的改进。

1 Snort规则树结构

Snort规则树是用来将入侵规则整理后优化的数据结构,以下为一条入侵规则:

alert tcp $EXTERNAL_NET any-> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI/cgi-bin/ access"; flow: to_server, established; uricontent:"/cgi-bin/"; content:"/cgi-bin/ HTTP"; nocase; classtype: web-application-attack; sid: 1668; rev: 5;)

Snort通过在初始化函数 ParseRulesFile()和 ParseRule()解析配置文件和入侵规则库,最后将入侵规则分析并构建了一张规则树结构,结构如图 1所示。

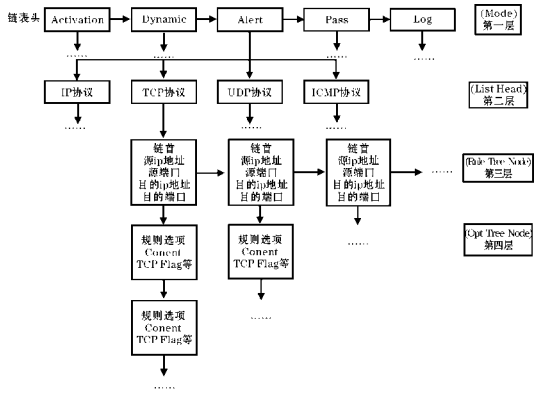

图 1 Snort规则树

当 Snort通过 Lipbcap从网络中捕获的包首先根据响应动作,网络协议匹配 TCP、UDP或者 ICMP中

收稿日期: 2008-03-04

基金项目:江西省重大攻关招标项目(2005A016);江西省重点工业攻关项目(20051B01005)

作者简介:周巍(1981-),男,湖北咸宁人,南昌航空大学计算机学院硕士研究生,研究方向:入侵检测,网络安全;江泽涛(1961-),男,江西都昌人,教授,硕士生导师,博士,研究方向:信息安全。

的一个,进入指定相同的源 IP、源端口、目的 IP和目的端口以及通信方向项,根据数据包内容与规则链子树的特征选项,进行模式匹配检测是否为入侵行为。

2　Snort匹配算法介绍

2.1 BM算法

BM匹配算法是 Boyer和 Moore两人在 KMP算法的启发下提出的一种字符串快速匹配算法。它在采用自右向左的方式扫描模式(P表示)与正文(T表示),一旦发现正文中出现模式中没有的字符时就可以将模式、正文进行一个大幅度的偏移。模式与正文在匹配比较的过程中,将 P与 T左对齐,即 P[1]与 T[1]对齐。匹配从 P的最右端字符开始,判断 P[strlen(P)](strlen(P)为模式长度)与 T[strlen(P)]是否匹配,如果匹配成功,则继续判断 P[strlen(P)-1]与 T[strlen(P)-1]是否匹配,如此反复比较,直到 P中的所有字符全部匹配成功或是有不匹配的字符出现。其移动遵循以下规律:

(1)坏字符移动:当发生不匹配时,移动模式串使得不匹配的正文字符与该字符在模式串中的最右出现对齐。如果字符没有在模式中出现,移动距离为字符到模式首位的长度;如果字符在模式中出现,移动模式中字符出现位置到正文字符处;

(2)好后缀移动:当匹配过程中发生不匹配时,如果已匹配子串长度不为 0,且匹配子串在模式串中有相同的其他子串,移动模式串使得已匹配部分与该部分在模式串中的最右端出现对齐。

如果同时满足两个条件,选取两者中的较大者作为模式串右移的距离。

例如:设正文为"otootowtmwotwom",要在其中搜索"ttowtow"。

正文 Text:　otootowtmwotwom
模式 Pattern:　ttowtow　　初始状态
模式 Pattern:　ttowtow　　根据坏字符移动 1位
模式 Pattern:　ttowtow　　根据好后缀移动 3位

根据坏字符移动规则,失配字符 o在模式中出现在 P[3],模式可以右移的距离为 1个字符,根据后缀移动规则,已成功匹配的子串 tow在模式前端又再次出现,可移动 3个字符。比较两种偏移值,实际 BM法的偏移量为两者中较大的,这里选择移动 3个字符。

2.2 AC算法

AC算法是基于 FSA(有限状态自动机)的,在进行匹配之前先对模式串集合进行预处理形成树形结构,然后只需对正文串 T扫描一次就可以找出与其匹配的所有模式串。

有限自动机的构造中,每个模式串的字符是从前到后依次加到树形的有限自动机中的,在匹配时,目标串的输入,即匹配过程,也是按照从前到后的次序。预处理生成 3个函数: goto(转移)函数, fail(失效)函数和 output(输出)函数。

设 U={0,1,2,…}为状态集合,转移函数 g: (U,P)→U为一映射,其建立过程为:逐个取出 P中模式串的每个字符,从状态 0出发,由当前状态和新取出的字符决定下一状态。如果有从当前状态出发并标注该字符的矢线,则将矢线所指的状态赋为当前状态,否则,添加一个标号比已有状态标号大 1的新状态,并用一条矢线从当前状态指向新加入的状态,并将新加入的状态赋为当前状态。P中的所有模式串处理完后,再画一条从 0状态到 0状态的自返线,标注的字符为不能从 0开始的字符集。例如, P={this, tow, there}的 goto函数如图 2所示。

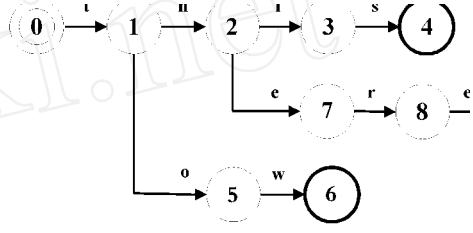

图 2　模式 P的 goto函数图

失效函数 f用来指明当某个模式与正文匹配不成功时,应处理的下一状态。f的构造方法为:所有第一层状态的失效函数为 0,如 f(1)=0;对于非第一层状态 s,若其父状态为 r(即存在字符 a,使 g(r,a)=s),则 f(s)=g(f(s*),a),其中状态 s*为追溯 s的祖先状态得到的第一个使 g(f(s*),a)存在的状态。如 f(4)=g(f(3),w)=g(f(1),o)=g(f(0),t)=0。由此计算出本例中 f(1)=f(2)=…=f(9)=0。

输出函数 output的作用是在匹配过程中输出已经匹配的模式串。goto函数图中黑粗圈的状态表示为输出, output的值如下: output(4)={this}; output(6)={tow}; output(9)={there};其他状态的输出为空。

AC算法的匹配过程如下:从状态 0出发,每取出正文串中的一个字符,利用 g和 f函数进入下一状态。当某个状态的 output函数不为空时输出其值,表示在正文串中找到该模式串。

2.3 AC-BM算法

AC-BM算法是在以上两种方法的基础上提出来的,对于多模式串的匹配而言,虽然 AC自动机匹配算法比基于 BM算法思想的多模式匹配算法高效得多。但 AC算法必须逐一地查看正文串的每个字符,而 BM算法能够利用跳转表跃过正文串中的大段字符,从而提高搜索速度。将 BM算法的这种启发式搜索技术应用到 AC算法中,则可大大提高多模式匹配算法的效率。

例如,设有字符串模式集合 {whataboutthese, whataboutthename, whataboutbanana, whataboutapples},要匹配的正文内容为 stringsearchingxampleinvolving。首先,构造字符串有限状态自动机,匹配从字符 a开始,显然 a与 w不匹配,由于字符串树中下一个 a出现在深度 3的位置,字符串树的最小字符串长度为 14,根据坏字符跳转,字符串树可以安全向左移动 2个字符。坏字符跳转的前后如图 3所示。

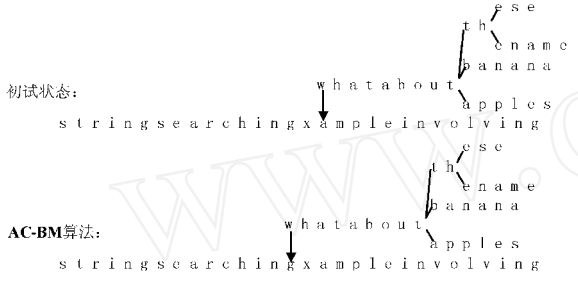

图 3 AC-BM算法

当既有好后缀跳转,又有坏字符跳转时,则判断如果有后缀跳转,就使用好前缀跳转 (子串跳转与后缀跳转的最小值);否则,使用好前缀跳转与坏字符跳转的最大值。

2.4 改进算法

原有的 AC-BM算法考虑到不匹配正文位置的字符在模式中下一次出现的深度,却未考虑其前一个字符在模式中的位置,在这个基础上,这里提出了改进措施,进一步提高其每次跳转步数。

分别计算正文不匹配字符与其前缀字符在模式中对应模式字符开始下一个出现相同字符的深度,取其中最大的进行偏移,其中如果前缀字符在模式中未找到对应字符,则取下一个字符继续与模式比较,直到匹配结束或者发现一个正文字符在模式中出现,此时该正文字符与模式字符串树中最长的字符串右对齐。

以上例进行分析,见图 4,初试状态正文字符 a在模式树中的移动位置为 2,而 a的前缀字符 x, g并未在模式中出现,直到字符 n,因此移动位置应该为字符串树的最大字符串长度位 +两个字符长度 (x, g),即: 16 + 2;使正文前缀 n与模式数最长子串右对齐,此时模式数已超过文本长度,无须移动进行比较,如果没有超过长度则继续下一轮比较,直到结束搜索。

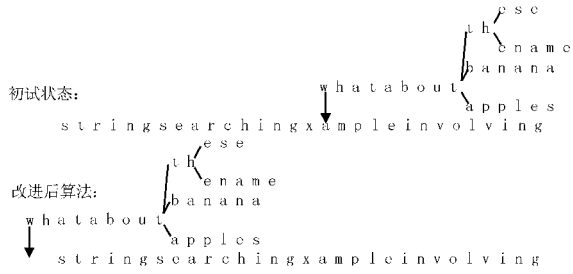

图 4 改进后的算法

3 实验测试和结果分析

这里对两种匹配算法进行匹配次数和比较的字符个数实验,得到如表 1的数据 (表中的数据是匹配次数比较 /比较的字符个数)。

表 1 算法比较

主串 T长度	模式串 Pi长度	AC-BM	改进算法
1024	P1: 12 P2: 10 P3: 7	1.114	0.995

其中,主串字符是随机的,给定三个模式串并且 P1和 P3在主串中。从表 1可以看出,改进后的算法在速度上有了进一步的提升。

4 结束语

本文介绍了 Snort检测引擎中构造的规则树结构和检测过程中使用到的模式匹配算法,分别分析了 BM算法、AC算法和 AC-BM算法并提出了改进。通过实验测试证明该算法能提高 Snort的性能。

参考文献:

[1] 唐正军. 入侵检测技术导论 [M]. 北京:机械工业出版社, 2004.

[2] 宋劲松. 网络入侵检测:分析、发现和报告攻击 [M]. 北京:国防工业出版社, 2004.

[3] 唐歉,张大方. 基于 Snort的入侵检测引擎比较分析 [J]. 计算机工程与设计, 2005, 26(11): 2884-2886.

[4] 赵念强,鞠时光. 入侵检测系统中模式匹配算法的研究 [J]. 微计算机信息, 2005(24): 22-24.

[5] 李晓芳,姚远. 入侵检测工具 Snort的研究与使用 [J]. 计算机应用与软件, 2006, 23(3): 123-124.