Snort软件代码分析、改进及再开发

杨　仝，　乔向东，　郑连清
(空军工程大学 电讯工程学院，陕西 西安　710077)

摘　要:出于研发网络入侵检测系统的需要,首先详细分析了入侵检测系统的规则链表结构;其次,针对 snort规则链表的冗余提出一种改进方法,并证明了该方法的正确性和有效性;最后,以改进后的 snort为内核,开发了网络入侵检测系统。并相继完成了包括 Snort内核模块、用户认证模块、网络流量监视模块等的开发工作,完成了网络入侵检测系统的架构。经过验证,系统稳定、出色的性能再次证明了改进方法是有效的。

关键词: snort;入侵检测;规则链表
中图分类号: TN936.24　　**文献标识码**: A　　**文章编号**: 1009 - 3516(2008)02 - 0084 - 04

snort是一款非常优秀的、开放源代码的入侵检测软件。它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,对攻击进行实时告警。从功能模块上看,各个模块功能明晰,相对独立,设计合理;从规则语法上看, snort的规则语法几乎已经成为规则语法标准[1-2]。但对 snort规则链表的冗余改进目前尚无见诸报导,本文对此进行研究。

1　snort规则及规则链表

检测规则是入侵检测系统的重要组成部分,它直接决定了系统是否能够高效、准确地检测出入侵行为。因此, snort为规则数据建立多种规则链表结构体,并在此基础上实现了快速规则检测引擎。下面首先分析规则语法。

1.1　规则语法

snort的规则[3]在逻辑上分为两部分:规则头和规则选项。规则头定义了规则的行为、所匹配报文的协议、源地址、目的地址及其网络掩码、源端口和目的端口等信息;规则选项[4]部分则包含了所要显示给用户查看的警告信息以及用来判定此报文是否为攻击报文的其他信息。

1.2　适合快速检测引擎的规则链表框架

snort2.0以后的新版本采用了新的快速规则检测引擎来力求体改检测效能,该快速检测引擎主要包含 3个方面的内容:规则最优化、多模式搜索引擎和事件选择器。

1.2.1　规则最优化

snort快速检测引擎的规则最优化原则包括[5-6]:

1)划分规则组要尽可能小、高效;

2)规则组不能互相重叠,也就是说,一条规则不能同时出现在两个或两个以上的规则组;

3)要使接收到的每一个数据包只需要搜索一个规则组。

snort快速检测引擎的基本过程是根据规则协议,先将已经解析好的按照规则动作划分为 Activation、Dynamic、Alert、Pass和 Log 5大类的 snort规则重新分为 4个 PORT_RULE_MAP结构体的规则组,分别为 prmTcpRTNX、prmUdpRTNX、prmIpRTNX和 prmIcmpRTNX。每个规则组又分为 3个端口组:源端口 PORT_

* 收稿日期: 2006 - 11 - 30
作者简介:杨　仝(1982 -),男,陕西西安人,助教,硕士生,主要从事入侵检测系统研究.
E - mail: yangtong_99@163.com.

GROUP。

数组(简称 SRC组)、目的端口 PORT_GROUP数组(简称 DST组)和通用端口 PORT_GROUP(简称 GEN组),前两者数组的维数为 65 535,我们这里称其中的每一维分量为槽。如图 1所示,源、目的组中以 NULL填充表示该槽为空,以阴影填充为表示该槽非空,以省略号填充表示省略了若干槽。

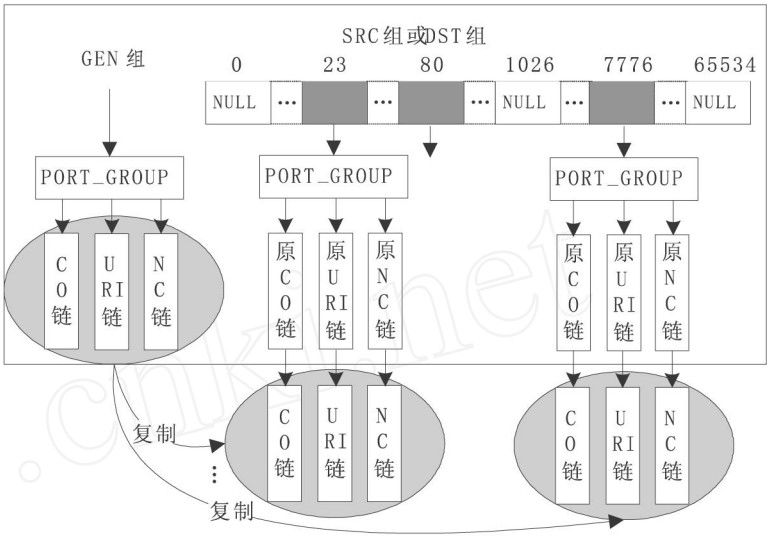

图 1　改进前的链表

Fig. 1　Rule Chain not improved

snort通过函数 fpCreateFastPacketDetection()经过一个庞大的 for循环遍历了全局变量 RuleLists中包含的 Activation、Dynamic、Alert、Pass和 Log5大类 ListHead的 TCP规则链表、UDP规则链表、ICMP规则链表和 IP规则链表,将所有的规则节点以 OTNX的形式加入规则组中,具体方法如下[7-8]:

第 1步　如果源端口值为特定值,目的端口为任意值(ANY),则依据端口值将该规则加入到源端口 PORT_GROUP数组的相应槽中。

第 2步　如果目的端口值为特定值,源端口为任意值(ANY),则依据端口值将该规则加入到目的端口 PORT_GROUP数组的相应槽中。

第 3步　如果源端口和目的端口都为特定值,则依据源端口值和目的端口值将该规则同时加入到源端口 PORT_GROUP数组和目的端口 PORT_GROUP数组的相应槽中。

第 4步　如果源端口和目的端口都为任意值(ANY),则该规则加入到通用 PORT_GROUP中。

第 5步　对于规则中端口值求反操作或者指定值范围的情况,等于端口值为 ANY情况。

每个 PORT_GROUP结构体包含 content、uri-content以及 no-content3个链表,如图 1虚框中所示。添加某一规则到某个 PORT_GROUP结构体时,具体依据该规则的规则选项 OTN是否有 content选项、uricontent选项将该规则以 OTNX结构体的形式加入到 PORT_GROUP的 content、uri-content、no-content3个链表中的一个。

对于通用端口链表而言,在构建快速规则匹配引擎中,它仅是作为一个过渡性的函数结构。为了确保一个数据包只需经历一个规则组的搜索匹配,函数 fpCreateFastPacketDetection()在完成上述规则添加遍历操作后还将对通用 PORT_GROUP进行了进一步处理——调用函数 mCompileGroups()。该函数将通用 PORT_GROUP中 content、uri-content、no-content链表存储的规则再添加到每个非空的源、目地端口 PORT_GROUP数组元素所包含的 content、uri-content、no-content链表中,如图 1椭圆框中所示。

这样做的原因是有一些规则没有特定端口的,那么它就被加入了通用端口链表中,一旦捕获到一个数据包,不管它有没有源、目的端口,除了要与源端口链表或者目的端口链表匹配外,还应该与通用端口链表中的规则进行匹配。但是,这样做的同时,带来了内存上的浪费,在后面我们会提出改进方法。

1.2.2　多模式搜索引擎

这里的模式指的是 content选项和 uri-content选项。多模式[9-10]指的是规则选项中含有多个 content选项或 uri-content选项,这样有利于更加准确地检测入侵,降低虚警概率。

为适应多模式引擎算法,snort通过函数 fpCreateFastPacketDetection调用 BuildMultiPatemGroups函数在每个 Port_Group结构中构建多模式搜索引擎所需的数据结构。

1.2.3　事件选择器

由于一个数据包可能匹配一个 PORT_GROUP规则组中的多条规则,因而可能触发多个事件,事件选择器就是跟踪一个数据包触发的所有事件。每个事件都有一个优先级,事件选择器将优先级最高的事件,发送给 snort输出系统[11]。

2 对 snort规则链表的改进

从图 1可以清楚地看出,通用链表的内容被多次重复放入了源端口链表和目的端口链表中。假设源端口数组非空槽的个数有 n_1 个,目的端口数组非空槽个数为 n_2 个,通用链表需要分配的内存为 m 字节,那么总共浪费的空间为 $m*(n_1+n_2)$字节。

为此,我们提出一种改进方法:

第 1步 将通用链表重组为 3个链表,分别为含有 content的链表(称作 GCO链)、含有 uri-content的链表(称作 GURI链)和不含 content的链表(称作 GNC链)。重组的方法就是将通用链表每个节点的 content list、uri-content list、no-content list分别首尾连接起来,形成 CO链、URI链、NC链,如图 2所示。

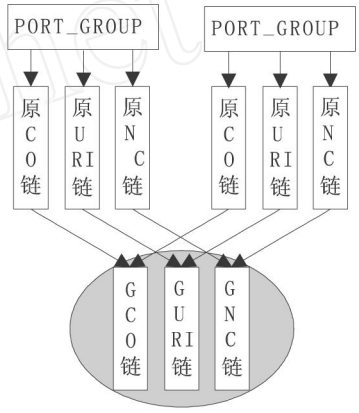

图 2 改进后的链表

Fig. 2 Improved rule chain

第 2步 遍历源端口组和目的端口组,如果槽非空,将该槽的 content list、uri-content list、和 no-content list的尾节点的 next指针分别指向 CO链、URI链、NC链头部。

```
int yt_prmCompileGroups( PORT_RULE_MAP * p, BOOL OnlyDst) //函数原形;
{PORT_GROUP * pgSrc0 = (PORT_GROUP *) calloc (1, sizeof (PORT_GROUP)); //定义端口组头指针;
PORT_GROUP * pgDst0 = (PORT_GROUP *) calloc(1, sizeof(PORT_GROUP)); //分配空间;
if(! pgSrc0||! pgDst0) return 0; //分配空间失败,返回 0;
pgGen = p->prmGeneric; //取得通用链表头指针;
if(! OnlyDst) //IF协议为 TCP或者 UDP;
{…} //构造原 CO链、原 URI链和原 NC链;
for( i=1; i<MAX_PORTS; i++) //MAX_PORTS为非空槽数;
{…} //将所有的原 CO链、原 URI链、原 NC链的尾指针指向 CO链、URI链、NC链的头部
```

在函数 fpCreateFastPacketDetection中调用 BuildMultiPatternGroups之前,调用自定义的 yt_prmCompileGroups函数,如下:

```
//依次将 TCP、UDP、ICMP、IP规则组的通用端口链表附加到源、目的端口组中。
yt_prmCompileGroups(prmTcpRTNX, FALSE); yt_prmCompileGroups(prmUdpRTNX, FALSE);
yt_prmCompileGroups(prmIcmpRTNX, TRUE); yt_prmCompileGroups(prmIpRTNX, TRUE);
yt_Output(); //自定义的测试结果的输出函数。
```

代码在 Visual C++ 2005下编译通过。利用这种方法,节约的内存空间为 $m*(n_1+n_2)-1$字节。

3 定量分析

为了进一步确认改进的正确性和有效性,这里对改进的效果进行定量分析。实验环境为:snort 2.2.01 727条规则,VC++ 2005。测试结果如表 1所示,从表中可以清楚地看出,改进后节省内存 7万字节。

表 1 实验结果

Tab. 1 Experiment result byte

	改进前	改进后	节省
初始 snort	779 6	779 6	0
解析配置文件前	60584	60 584	0
解析配置文件后	7 770 221	7 770 221	0
建立快速引擎前	7 791 595	7 791 595	0
prmCompileGroups前	9 957 819	9 957 819	0
prmCompileGroups后	10 169 211	9 962 903	206 308

4　snort源码的再开发

课题组于 2006年 6月完成了 snort内核的改进、提取工作,在此基础上进行了网络入侵检测系统的开发,相继完成了包括 snort内核模块、用户认证管理模块、网络流量监视模块、语音报警模块、动态规则管理模块、日志分析模块和数据源监听模块的开发工作。

5　结束语

本文重点分析了 snort规则链表,提出了一种改进方法,并证明了改进的正确性和有效性,文章最后阐述了以 snort为内核的网络入侵检测系统的架构。

参考文献:

[1] 薛静锋,宁宇鹏,阎　慧. 入侵检测技术[M]. 北京:机械工业出版社, 2004.
XUE Jingfeng, NING Yupeng, YAN Hui. Intrusion Detection Technology[M]. Beijing: Machine Engineering Press, 2004. (in Chinese)

[2] 韩仲祥. 基于 MBII的 IDS实现研究[J]. 空军工程大学学报:自然科学版, 2005, 6(5): 55 - 59.
HAN Zhongxiang. Implementation of IDS Based on MBII[J]. Journal of Air Force Engineering University: Natural Science Edition, 2005, 6(5): 55 - 59. (in Chinese)

[3] 刘文涛. 网络安全开发包详解[M]. 北京:电子工业出版社, 2005.
LIU Wentao. Detail Explanation of Network Security Development Packet[M]. Beijing: Electricity Engineering Press, 2005. (in Chinese)

[4] Kinage. Snort user guaid [EB/OL]. (2005 - 11 - 10)[2006 - 03 - 12] http://blog.tianya.cn/blogger/post_show.asp?BlogID = 154815&PostID = 3192150&idWriter = 0&Key = 0. 2005. 11.

[5] snort 2.0 RULE OPTMIZER [CP/OL]. 2003[2006 - 03 - 15] February 2003 http://www.sourcefire.com.

[6] snort 2.0 Hi - performance Muti - rule Inspection[CP/OL]. 2004[2006 - 03 - 15] http://www.sourcefire.com.

[7] snort 2.0 Detection Revisited[CP/OL]. 2004[2006 - 03 - 15] http://www.sourcefire.com.

[8] snort 2.0 Protocol Flow Analyzer[CP/OL]. 2004[2006 - 03 - 15] http://www.sourcefire.com.

[9] Roberto P. Alarm Clustering for Intrusion Detection Systems in Computer Networks[J]. Engineering Application of Artificial Intelligence, 2006, 19(3): 429 - 438.

[10] 陈铁柱. Snort规则集优化[J]. 海军工程学院学报, 2005, 20(6): 664 - 666.
CHEN Tiezhu. Optimization on the Rules Set of Snort[J]. Journal of Naval Aeronautical Engineering Institute, 2005, 20(6): 664 - 666. (in Chinese)

[11] Denning D E. An Intrusion - detection Model[J]. IEEE Trans - action on Software Engineering, 1987, 13(SE): 222 - 232.

(编辑:田新华,徐楠楠)

Analysis, Improvement and Redevelopment of the snort

YANG Tong, QIAO Xiang - dong, ZHENG Lian - qing
(Telecommunication Engineering Institute, Air Force Engineering University, Xi'an 710077, China)

Abstract: Snort, one of the best Open Source Network Intrusion Detection Systems, is analysed in detail, in this paper, for the sake of searching network intrusion detection system. Then a solution is proposed to eliminate the redundancy of snort's rule chain. Experiments are done, which show that the solution proposed is correct and effective. Finally, on the basis of ARP technology approach, NIDS is developed with the improved snort as kernel module. Its excellent performance proves the solution to be valid once again.

Key words: snort; intrusion detection; rule chain