文章编号: 1003-5850(2009)01-0023-03

基于Snort的入侵检测系统的研究及BM 算法的改进

Intrusion Detection System Research and Improvement of BM-algorithm based on Snort

赵绍东 闫宏印

(太原理工大学计算机与软件学院 太原 030024)

【摘 要】入侵检测系统是传统被动方式的网络安全检测手段的重要补充, 它是一种主动、实时、自动检测入侵行为的工具和手段。Snort 是一个成熟的开放源代码的网络入侵检测系统, 介绍了入侵检测系统的基本原理, 研究了入侵检测系统Snort 的体系结构、规则的解析流程和检测流程, 对Snort 的BM 字符匹配算法进行深入研究, 提出了BM 字符匹配算法的改进方法。

【关键词】Snort, 入侵检测, BM 算法

中图分类号: TP393.08 文献标识码: A

ABSTRACT Intrusion Detection System is an important supplement of the traditional passive way of network security detection means, it is a tool and means which is pro-active, real-time and can automatically detect the invasion. Snort is a mature open source code network invasion detection system , Introducing to the basic principles of Intrusion Detection System, The essay introduces its structure, process and detection flow of rule, and then Boyer-Moore algorithmic of snort is researched deeply. Improved method of Boyer-Moore algorithmic is proposed.

KEYWORDS Snort, intrusion detection, Boyer-Moore algorithmic

当今对计算机网络的攻击已经成为很严重的问题。防火墙作为主要的安全防范手段, 在很多方面存在弱点, 而入侵检测系统能够自动的监控网络的数据流, 迅速发现攻击, 对可疑的事件给予检测和响应。下面对Snort 的体系结构、规则的解析与检测流程进行研究, 阐述了Snort 在网络入侵检测方面的应用。

1 Snort 的研究

1.1 Snort 的系统概述

Snort 系统是一个以开放源代码(Open Source)形式开发的网络入侵检测系统, 由Martin Roesch 编写, 并由遍布世界各地的众多程序员共同维护和升级。Snort 运行在Libpcap 库函数基础之上, 并支持多种系统软硬件平台, 如RedHat Linux、Debian Linux、HP-UX、Solaris(X86 和Sparc)、x86 Free/Net/OpenBSD、NetBSD 以及MacOS X 等。系统代码遵循GNU/GPL 协议。

Snort 系统具有系统尺寸小、易于安装、便于配置、功能强大、使用灵活等诸多优点。实际上, Snort 不仅是一个网络入侵检测系统, 它还可作为网络数据包分析器(Sniffer)和记录器(Logger)来使用。它采用基于规则的工作方式, 对数据包内容进行规则匹配来检测多种不同的入侵行为和探测活动, 例如缓存溢出、隐蔽端口扫描、CGI 攻击、SMB 探测等等。Snort 具备实时报警功能, 可以发送警报消息到系统日志文件、SMB 消息或者是指定的警报文件中。系统采用命令行开关选项和可选BPF 命令的形式进行配置。系统检测引擎采用了一种简单规则语言进行编程, 用于描述对每一个数据包所应进行的测试和对应的可能响应动作。

1.2 系统程序架构

整个Snort 系统架构的着眼点在于强调性能、简洁和灵活性三个方面。大体上, Snort 系统由三个子系统构成: 数据包解析器、检测引擎和日志/报警子系统。所有这些子系统都是建立在数据包截获库函数接口Libpcap 的基础之上, Libpcap 为它们提供了一个可移植的数据包截获和过滤机制。整个程序的配置、规则的解析以及数据结构的初始化都在系统进行数据包分析和检测之前完成, 以保证对每个数据包的处理时间压缩到最小, 以获得最好的运行性能。

1.3 Snort 的工作流程

snort 工作流程如下(如图1 所示):

①初始化重要的变量和结构, 调用命令行参数解析函数ParseCmdLine(), 该函数就是解析制定的各种

* 2008-08-29 收到, 2008-11-29 改回

** 基金项目: 山西省教育改革项目(基于网络环境的计算机基础教学改革)。

*** 赵绍东, 男, 1981 年生, 在读硕士, 研究方向: 计算机网络。

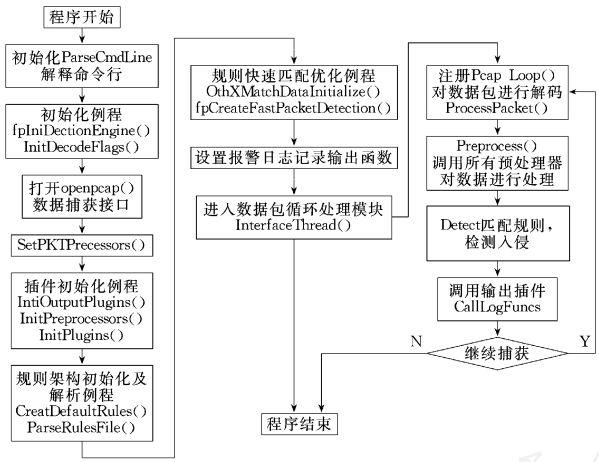

图1 Snort 工作流程图

命令行参数,并将解析结构存放在全局变量PV 类型的变量*pv* 中。数据类型PV 是snort 最重要的结构之一,其中包含各种标识字段,用来指示各种系统参数设置情况。同时进行检测引擎初始化例程fpIniDectionEngine(),通过对该函数的调用制定快速规则匹配模块的缺省配置参数,包括缺省模式搜索算法(BM 算法等)。协议初始化例程InitDecodeFlags(),负责指定在协议解析中产生警报信息的错误类型。

②调用OpenPcap()打开数据包截获端口。该函数是根据ParseCmdLine()的解析结果,分别调用Libpcap 库函数Pcap_open_live()和Pcap_open_offline(),并获得对应的数据包截获端口数据结构。SetPKTProcessor()函数的功能是根据当前网络链路层类型,设置不同的底层链路层协议解析例程。

③对插件进行初始化。插件初始化例程主要包括输出插件初始化例程InitOutputPlugins()、检测插件初始化例程InitPlugins()和预处理插件初始化例程InitPreprocessors()等。各种插件的初始化都是类似的,主要任务就是将各种插件的关键字名称与对应的初始化处理程序对应起来,并注册到对应的关键字链表结构中,以便随后的规则解析例程使用。

④进行规则架构初始化及解析例程。调用CreateDefault-Rules()进行初始化架构的建设工作,该函数并没有执行规则解析的任务,只是负责搭建默认的上层规则架构。关键的检测规则解析任务是由函数ParseRulesFile()负责完成的。ParseRulesFile()是snort 系统中一个核心的关键模块,该函数不仅是检测规则集,而且对于所有的系统配置都进行解析,包括预处理器、输出插件、配置命令等。

⑤规则快速匹配优化例程是在snort2.0 以后才引入的最重要的设计。在SnortMain()函数中可以找到这些例程,该部分主要涉及调用了构造和初始化用于规则优化和快速匹配数据结构的例程。

⑥进入数据包处理模块InterfaceThread()。该函数主要是调用Libpcap 库函数的接口回调函数Pcap_loop(),并由Pcap_loop 函数调用snort 两个主要的接口函数:ProcessPacket()对数据包进行解码分析和Preprocess()调用所有的预处理器进行入侵检测的任务。接口函数ProcessPacket()中调用数据包协议解码模块,并将解码结果存储在特定的Packet 类型数据结构中、根据snort 的不同运行模式,调用不同的处理模块和对入侵检测的运行模式,调用Preprocess()模块。当snort 运行在入侵检测模式下,接口函数Preprocess()被调用。该函数将遍历所有的预处理模块,如果满足进行检测的条件,则调用规则检测模块Detect()。判断完成之后调用输出插件函数,如:CallLogFuncs()等以判断应该采用什么形式进行报警。

2 Snort 规则匹配算法分析

2.1 BM 算法

大多数的NIDS 都是采用模式匹配的方法进行判定,而在进行模式匹配时又大多采用精确的字符串匹配的方式作为其判断入侵行为的依据。在系统中进行字符串匹配是一种非常耗时的工作,尤其是现在网络数据流量巨大的情况下。如:在snort2.4 系统中所自带的规则项目(rules 文件夹中)就有3 036 条之多,在这3 000 多条的规则中还不包含用户自定义的规则项目,同时其所涉及的范围只是针对于IP、TCP、UDP 和ICMP 协议。根据实验统计: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(Boyer—Moore)算法是一种非常有效的字符串匹配算法。该算法,在对于文本匹配的时候,匹配字符串越长,字符串匹配所需要的时间就越短。

BM 算法的主要步骤如下:

该算法所需变量:将目标字符串模板,设为Prtc,长度为*m*;将所需要检测的文本字符串,设为Txtc,长度为*n*;两种情况下启发式搜索过程的字符偏移数组分别设为skip 和shift。步骤1:将Prtc 与Txtc 从左向右对齐,然后选择Prtc 最右字符所对应的Txtc 中的字符作为开始比较的字符,用于检查Prtc 中是否包含此字符;在检查的过程中,如果发现Prtc 中不包含该字符,就将Prtc 整个字符串向右移第*m* 个字符的距离;如果发现在Prtc 中包含该字符,就启动skip 搜索过程,计算所需要的字符长度,然后将Prtc 向右移动相应的距离。步骤2:如果发现不匹配字符,同时不匹配字符右边已经有部分的字符匹配的情况下,就触发

shift 搜索过程,计算所需移动的字符长度,然后将Prtc向右移动相应的字符距离。步骤3:当在Txtc中查找到匹配的Prtc字符串后,就结束检查,退出树结构并返回一个值。

2.2 BM 改进算法

在snort的BM 算法中,如果考虑从下一个字节开始进行匹配,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。对于被检测的字符文本长度一定的情况下,可以加快字符串的偏移量,从而提高BM 算法的运行速度。因此考虑在基于BM 算法的基础上进行优化处理。改进算法就是考虑怎么使用T 文本中下一个字节作为匹配的基础。如果匹配开始的时候就采用该算法,将会出现下面的情况:如果T 文本中第一次比较的字符串就是P字符串,则不能精确匹配。所以在开始进行匹配的时候要保证检测的准确性,就需要选择使用T 文本中与之对应的字符进行比较,而选用下一个字符时只能在右边有部分字符已经发生匹配的情况下进行。只要检测到不匹配字符的右边有匹配字符,就采用下一个字符与P 中最后一个字符。但是源程序中的shift 算法要继续,以便于比较字符偏移的大小。因为在匹配过程中,可能shift 算法将是更大的偏移量,所以需要比较最大偏移值。

3 结束语

Snort 简洁、高效、自由并且易于扩展,能实时分析流量,检测多种攻击和嗅探。它和防火墙联动,在发现入侵行为时能够很快地进行阻断,因此可以很好地保护内部网络免受攻击。本文主要论述了网络入侵检测系统Snort 的具体实现和应用。并从规则优化和匹配规则优化两个角度讨论了对Snort 安全性能提高的方法。

参 考 文 献

[1] 邓 吉,张奎亭,罗诗尧. 网络安全攻防实战[M]. 北京:电子工业出版社,2008.

[2] 唐正军. 网络入侵检测系统的设计与实现[M]. 北京:电子工业出版社,2002.

[3] 兰景英,王永恒. Snort 研究及BM 算法改进[J]. 计算机工程与设计,2008,29(9):5-6.

[4] Fisk M, Varghese G. An Analysis of Fast String Matching Applied to Content Basedforwarding and Intrusion Detection[R]. TechnicalReport CS2001-0670,2002.

[5] 谷晓钢,荣 安. Snort 的高效规则匹配算法[J]. 计算机工程,2006,32(18):155-156.

(上接第18页)

* 将数据进行分类,并将类别进行编码;
* 按实际情况,给用户分配相应的数据类别;
* 按用户可操作的数据类别,只给用户提供可查看或操作的数据。

由于某个用户可操作的数据类别可能会随时改变,所以应用程序中应该提供按用户定制其可操作的数据类别的功能。

1.6 控件层安全策略

面向对象的GUI应用程序的界面往往包含多个控件。它们被用于显示、选择、输入数据、运行脚本。用户可以不分先后地对其进行操作。这就存在着各种操作次序的组合。但按照工作流程、制度规范、职务权限等来看,这些操作组合中的大部分是不合理的、不存在的。一个好的应用程序应该避免这些操作组合(即,堵塞所有可能的漏洞),以免影响安全性。一般需要采取如下策略:

* 修改控件的visible、enable 属性,限制控件的可视性、可用性,即使其不可以被操作;
* 在窗口的open 事件脚本中,限制用户首先不可以操作的控件;
* 在控件的某个事件脚本中,限制用户下一步不可以操作的控件;
* 在运行控件的脚本过程中,首先判断该项操作是否违反了安全性;
* 给用户提供联机的操作提示、向导。

2 结束语

在OA、MIS 系统中要考虑和解决的安全性问题是比较多的,其中一些是明显的、直接的,而另一些是隐含的、间接的。能否提供一组严密的安全策略,关系到应用程序是否具备实用性、是否能够产品化的问题。

参 考 文 献

[1] 朱爱民. PowerBuilder 8.0 编程实用技术与案例[M]. 北京:清华大学出版社,2002.

[2] 北京计算机教育培训中心. Oracle 9i for Windows NT/2000 数据库系统培训教程(基础篇)[M]. 北京:清华大学出版社,2002.

[3] 胡欣杰. Oracle 9i 数据库管理员指南[M]. 北京:希望电子出版社,2002.

[4] 路 川. Oracle 10g 宝典[M]. 北京:电子工业出版社,2006.