

文章编号:1001-9081(2009)01-0181-04

## 基于 RS\_Adaboost 的入侵检测方法

李恒杰

(甘肃联合大学 理工学院, 兰州 730000)

(li-hengjie@163.com)

**摘 要:**针对入侵检测系统存在的对入侵事件高漏报率和误报率,提出了一种将粗糙集(RS)方法与自适应增强(Adaboost)算法相结合的入侵检测方法。利用粗糙集理论在处理大数据量、消除冗余信息等方面的优势,减少 Adaboost 训练数据,提高处理速度。Adaboost 是一种构建准确分类器的学习算法,它将一族弱学习算法通过一定规则结合成为一个强学习算法,从而通过样本训练得到一个识别准确率理想的分类器。实验表明,该方法具有较高的检测率和检测效率。

**关键词:**入侵检测;粗糙集;约简;Adaboost 算法;分类

**中图分类号:** TP393.08 **文献标志码:** A

## Intrusion detection method based on rough set and adaptive boost

LI Heng-jie

(College of Science, Gansu Lianhe University, Lanzhou Gansu 73000, China)

**Abstract:** To solve the problem of high rate of false negatives and false positives of IDS, an intrusion detection method was proposed in this paper, which combined Rough Set and Adaboost algorithm. Rough set was used to reduce amount of Adaboost' training data and improve running speed. Adaboost was a learning algorithm for constructing accurate classifiers. It can obtain a strong learning algorithm by combining a series of weak learning algorithms through some rules. The experimental results show that the model has high detection rate and detection efficiency.

**Key words:** intrusion detection; Rough Set (RS); reduction; Adaboost algorithm; classification

### 0 引言

随着计算机应用越来越广泛,各种安全问题层出不穷。目前,已经采用了许多措施来保护计算机系统的安全,但这些都属于静态防护措施,难以满足复杂多变的应用环境,入侵检测系统因其能提供有效的动态保护功能而成为当今的研究热点。

由于分类算法和训练样本等多方面的问题,国内外的多种入侵检测方法存在对入侵事件高漏报率和误报率的问题。因此,结合粗糙集理论和自适应提高算法,本文提出了一种新的入侵检测模型:基于 RS\_Adaboost 的入侵检测方法。该方法利用抓包工具获取网络数据包,并进行数据预处理,使其适合粗糙集的数据格式,利于数据的约简。然后运用粗糙集约简方法对已获取的网络数据进行约简,可减少大量的训练数据、测试数据;并对自适应增强 Adaboost 算法进行了改进,将权重更新公式改进,更有利于分类检测。利用自适应增强 Adaboost 改进算法构建准确分类器,将一族弱学习算法通过一定规则结合成为一个强学习算法,从而通过样本训练得到一个识别准确率理想的分类器。最后以 1999 DARPA 为入侵检测数据,对 Adaboost 算法改进前和改进后的检测性能以及其他算法进行了对比,实验表明,该方法具有很高的检测率和检测效率。

### 1 基于 RS\_Adaboost 的入侵检测模型

本文提出的基于 RS\_Adaboost 入侵检测模型如图 1 所示,由网络数据获取、数据预处理/特征提取、RS 粗糙集约简、Adaboost 自适应增强分类和检测模块五部分组成。

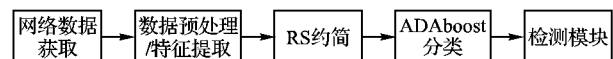


图 1 基于 RS\_Adaboost 的入侵检测模型

### 2 网络数据获取模块

网络数据包是入侵系统的最主要的信息来源。网络数据包获取的基本原理是<sup>[1]</sup>:当网络数据流在网段中传播时,由于局域网普遍采用的是基于广播机制的以太网协议,网上所传输的数据包能被共享信道的所有主机所接收,这样就可以利用特殊的数据捕获技术,收集网络中正在传输的数据包,作为入侵检测系统的数据源。

以太网数据传输是通过广播方式实现的。但是一般在系统正常工作的时候,应用程序只能接收到以本主机为目标主机的数据包,其他数据包将被丢弃不作处理。因此要截获到流经网卡的不属于自己主机的数据,必须绕过系统正常工作的处理机制,直接访问网络底层。首先将网卡工作模式置于混杂模式,使之可以接受目标 MAC 地址而不是自己 MAC 地址的数据包,然后直接访问数据链路层,截获相关数据,并进行过滤处理,这样就可以截获流经网卡的所有数据。

Tepdump<sup>[1]</sup>是 UNIX 系统提供的一个截获和分析网络数据包的工具,它通过调用 Libpcap 数据捕获函数直接与内核驱动程序交互操作,来实现网络数据的截获。在 Windows 上有相应的 Windump 工具及 WinPcap 数据捕获函数。WinPcap 是由伯克利分组捕获库派生而来的分组捕获库,它能够在 Windows 操作平台上实现对底层包的截取过滤。

数据包解析的功能是辨别数据包的协议类型,还原对应的数据包头信息和部分协议的数据内容,对数据包进行必要

的重组和还原,以便智能检测程序做进一步的检测分析。数据包解析的主要工作是将接收到的原始数据包进行分解,分析出数据包的种类、各种参数以及特定协议的数据内容。在分解过程中,拆分的依据是在网络协议中定义的各种包的类型和包的格式,主要是 TCP/IP 协议。

获取了数据包后,首先从用户缓冲区中取出原始的数据包,按照以太网数据帧头的数据结构分别取出目标以太网网卡地址、源以太网网卡地址和协议类型,然后根据不同的协议类型解码,区分出 IP、地址解析协议(Address Resolution Protocol, ARP)包以及反向地址解析协议(Reverse Address Resolution Protocol, RARP)包等,对于 IP 包的解析是其中的重点,将进一步还原出 TCP、用户数据报协议(User Datagram Protocol, UDP)等协议包头的信息以及必要的内容;在预处理模块中计算一次连接会话的延续时间,分析会话结束时 TCP 连接所处的状态,并根据特定协议的端口号对 HTTP, TELNET, FTP 等关键数据包的内容进行重组和整理。

数据包的解析完成了对原始数据包的分类还原,解析后的数据和主机日志审计数据一起提供给智能检测程序相应的检测模块进行分析。

### 3 数据预处理/特征提取模块

由于从网络上截获的数据是二进制的,首先要对二进制网络数据进行预处理,处理成适合粗糙集的格式<sup>[2]</sup>。由于在同一段时间内,网络上可能会同时建立很多连接,这些连接数据包是按时间顺序排列的,这样就会导致不同连接的数据包相互穿插。因此,为了收集有关连接的信息,需要把所有关于一条连接的所有数据包整理成一条连接记录。首先把截获的二进制数据包文件转换成 ASCII 格式的分组数据包,一个数据包一行,将这些数据包按照时间戳排序。然后再把这些数据包处理成一组由属性特征组成的连接记录。用一个脚本程序(如 nmap 工具)扫描 ASCII 格式文件中的每行数据,把属于同一连接的所有数据包总结成一条连接记录,对每条 TCP 连接,脚本程序完成的工作:

在连接建立阶段,检查 TCP 三次握手是否正常,如果没有正常建立连接,检查连接的结束状态以及连接建立不成功的原因。在数据传输阶段,监控所有的数据包和控制包,记录与连接有关的某些统计值。如:连接持续时间、重传率、两方向上传送的字节数等。在连接结束阶段,检查连接结束的状态。如 SF: TCP 会话正常结束; REJ: 一方发出 SYN(同步),另一方发送 RST(复位连接)作为应答结束连接; SO: 一方发出 SYN,另一方没有应答。

因为 UDP 是面向无连接的数据传输协议。在数据传输过程中不存在握手连接或拆除连接的过程,所以把每个 UDP 数据包看成是一条连接。此外,把每个 Internet 控制消息协议(Internet Control Message Protocol, ICMP)包也看作是一条连接。Tcpdump 原始数据经过预处理被转换成每个连接用一条记录表示的数据集集合。这样,就可以进一步建立基于网络行为分类的入侵检测系统了。

经过预处理后,原始审计数据转化成为一条条的连接记录,每条连接记录的基本属性有:连接持续时间、发送接收的字节数、连接状态标志、协议类型(TCP 或 UDP)、服务类型(目的端口)等。

但是在入侵检测实验中如果直接将上述连接记录用于粗糙集算法,效果并不理想,因为网络事件通常在时间上有很强的相关性,尤其对于拒绝服务攻击(SYN-Flood, teardrop 等)以

及 PROBE-探测攻击(portscan, ping-sweep 等)更是如此。因此,考虑在检测数据中加入基于时间的统计特性,采用的方法是使用时间窗概念,针对每一条连接记录,统计在指定时间窗(本文采用时间窗为 2 s)内与当前连接记录在属性上存在某种联系的数据记录,包括以下两种统计方式:

1) 检查在 2 s 时间窗口内目标地址是同一主机的记录,包括:出现 SYN、REJ 错误的百分比;目标端口相同或不同的连接各占的百分比;目标端口相同的连接次数等。

2) 检查在 2 s 时间窗口内目标端口是同一服务端口的记录,包括出现 SYN、REJ 错误的百分比,目标主机不同的连接所占的百分比等。

## 4 RS 约简

### 4.1 数据标准化

在对检测数据进行约简之前要先对数据进行标准化<sup>[3]</sup>。因为数据各属性选用的度量单位将直接影响约简分析的结果。标准化的方法是将原来的度量值转换为无单位的值。

对于经过分析处理后的每条网络连接记录,都包含有两种类型的特征向量,分别是数字形式的特征值和离散形式的特征值,如连接传送的字节数、同一端口的连接数都属于数字形式的特征值,而连接使用的协议类型等则属于离散形式的特征值。还有一些用数字表示的特征值实际上也属于离散形式的特征值,如连接的目的端口地址等。在标准化时对于这两种特征值采用了不同的处理方法。

#### 4.1.1 数字形式特征值的标准化处理

可以根据各个属性值的平均值  $AVG_j$  和平均绝对偏差  $S_j$  ( $1 \leq j \leq d$ ) 来进行标准化处理:

$$AVG_j = \frac{1}{N} (X_{1j} + X_{2j} + \dots + X_{Nj}) \quad (1)$$

$$S_j = \frac{1}{N} (|X_{1j} - AVG_j| + |X_{2j} - AVG_j| + \dots + |X_{Nj} - AVG_j|) \quad (2)$$

其中  $X_{1j}, X_{2j}, \dots, X_{Nj}$  分别对应于第  $j$  个属性在  $N$  个数据项中的取值,则第  $i$  个  $j$  数据项的第  $j$  个属性值  $X_{ij}$  标准化为:

$$new\_X_{ij} = \frac{X_{ij} - AVG_j}{S_j} \quad (3)$$

#### 4.1.2 离散形式特征值的标准化处理

对于离散形式的特征  $i$ ,若  $\Sigma i$  是它的所有可能取值的集合,那么这个离散特征就对应着特征空间中的  $|\Sigma i|$  个坐标 ( $|\Sigma i|$  表示  $\Sigma i$  集合的元素个数)。对于特征  $i$  的某个给定的值,与这个值相对应的坐标值就是  $1/|\Sigma i|$ ,其他的  $|\Sigma i| - 1$  个坐标的值为 0。如果两条记录的离散特征  $i$  有不同的取值,那么在两条记录的平方距离和中加上  $2/(|\Sigma i|^2)$ ;如果取值相同则加上 0。可以看到离散形式的特征值间的距离是由其对应特征的取值范围决定的。例如在训练数据集集中的离散特征值 *service*,若其取值有 ftp、http、smtp 三种,记录 *record1* 的 *service* 取值为 ftp,则这个 *service* 属性上的值可以被标准化为  $(1/3, 0, 0)$  的形式,而记录 *record2* 的 *service* 取值为 http,那么标准化后的值可记为  $(0, 1/3, 0)$ ,计算 *record1* 和 *record2* 的距离时其距离的平方和中此项的值为  $2 \times [(1/3)^2]$ 。

### 4.2 实值属性离散化

对数据集集中的实值属性进行离散化,不但有利于属性约简,而且能够减少后续步骤的处理时间。有许多离散化方法,包括 booreasoning 算法、semi-naive 算法、equal interval width 算法等,本文中我们选用了局部离散化方法(local strategy)<sup>[4]</sup>。

局部离散化方法是首先找到最好的分割,然后把所有对

象分成两部分,然后重复这一过程,直到所有对象都分割完毕,具体过程如下所示:

1) 初始化二叉树  $T$  为空树,标记树根为对象集合  $U$ ,树根的状态为 unready。

2) 有叶子节点的状态标记为 unready 时,执行第3步;否则算法结束。

3) 对任何状态标记为 unready 的叶子节点  $N$ ,如果节点  $N$  中的对象有相同的决策值,则转到第4步;否则转到第5步。

4) 改变  $N$  的状态为 ready,替代  $N$  的对象集合为它的公共决策值。

5) 对于  $N$  中的对象,搜索最好的  $cut(a, c)$ ,把  $N$  的标记替换为  $N_2 = \{u \in N: L_{HN}(u) \geq 0\}$ ,建立两个新的节点  $N_1, N_2$ :  $N_1 = \{u \in N: L_{HN}(u) < 0\}, N_2 = \{u \in N: L_{HN}(u) \geq 0\}$ 。

## 5 Adaboost 分类检测

### 5.1 Adaboost 算法基本思想

Adaboost<sup>[5]</sup>的前身是 Boosting 算法。Boosting 作为一种通用的学习算法,可以提高任一给定算法的性能。Boosting 根源于研究机器学习的理论框架——“PAC”学习模型(由 Valiant 提出的关于可学习性的理论: Probably Approximately Correct)。Kearns and Valiant 最先指出<sup>[6]</sup>,在“PAC”学习模型中,若存在一个多项式级的学习算法来识别一组概念,并且识别率很高,那么这组概念是强可学习的;而如果学习算法识别一组概念的正确率仅比随机猜测的略好,那么这组概念是弱可学习的。如果能将一个弱学习算法提升为强学习算法,那么在学习概念时,只要找到一个比随机猜测略好的弱学习算法,就可以将其提升为强学习算法,而不必直接去找通常情况下很难获得的强学习算法。

Adaboost 算法的输入是一个训练集  $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$  其中  $x_i$  属于某个域或实例空间  $X$ ,  $Y$  为有限的表示空间,  $Y \in \{0, 1\}$ , Adaboost 在一系列的迭代中重复调用给定的弱或基本学习算法。

算法的主要思想是维持训练集上的一个分布或权重集(表示为  $D_t$ )。初始状态下,分布  $D$  的权值是相同的。在每次迭代中,运用表中的调整公式调整每个样本的权值,使每次输入弱学习器的样本集具有不同的权重,让弱学习器集中学习那些使用前一规则最难以预测的样本上。

弱规则的作用是用来产生一个弱假设<sup>[7]</sup>:  $h_t: X \in \{0, 1\}$ 。弱假设的好坏由错误率  $\varepsilon_t$  来衡量,而且误差的测量是与学习器被训练的分布  $D_t$  相关的。在实际使用中,弱学习器的算法可以是作用于训练实例的权重分布  $D_t$ ,也可以是取样于分布  $D_t$  的某一训练子集(未赋权重)。

算法的描述如下:

1) 输入:  $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ , 其中  $x_i \in X$ ,  $y_i \in Y$  且  $Y \in \{0, 1\}$

2) 初始化:  $D_1(i) = 1/m$

3) 训练过程:

For  $t = 1, 2, \dots, T$ :

① 把  $D_t$  传给并调用弱规则学习器

② 获得  $\varepsilon_t = P_{t \sim D_t}[h_t(x_i) \neq y_i]$  的弱规则

$h_t: X \rightarrow \{0, 1\}$

③ 选择  $\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_t}{\varepsilon_t} \right)$

④ 利用调整公式调整矩阵  $D$  的权值

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i [l] h_t(x_i))}{Z_t} \quad (4)$$

其中  $Z_t$  是正规化因子。

4) 输出: 最终假设

$$H(x) = \text{sign} \left[ \sum_{t=1}^T \alpha_t h_t(x) \right] \quad (5)$$

算法的数据流程图参见图2。

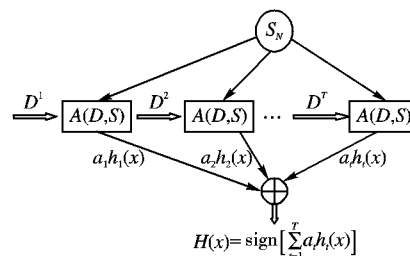


图2 Adaboost 算法的数据流程

在得到弱假设  $h_t$  后, Adaboost 需要选择一个参数  $\alpha_t \in R$ ,  $\alpha_t$  测量了  $h_t$  的重要性, 如果  $\varepsilon_t \leq 1/2$ , 则  $\alpha_t \geq 0$  (这种情况下, 可以认为没有总误差), 而且  $\varepsilon_t$  越小,  $\alpha_t$  越大。然后使用规则修正分布  $D_t$ 。规则的作用是增加被  $h_t$  错分的实例权重, 减少正确分类的实例的权重。因此, 权重趋向于难以分类的实例上。最终规则  $H$  是  $T$  个弱假设 ( $\alpha_t$  是赋给  $h_t$  的权重) 的占绝大多数的投票。

### 5.2 Adaboost 算法的改进

先对分类器的几个重要性能指标作以定义, 这些定义将在后面用到:

FP (False Positive): 发生错检的正目标;

FN (False Negative): 发生错检的负目标;

FPR (False Positive Rate): 错检的正目标数目与全部正目标数目之比;

FNR (False Negative Rate): 错检的负目标数目与全部负目标数目之比;

上述的 Adaboost 算法中样本权重调整过程是:

$$\omega_{t+1,i} = \omega_{t,i} \beta_i^{1-\varepsilon_i} \quad (6)$$

若样本  $x_i$  被 Adaboost 在第  $t$  轮迭代后产生弱分类器  $h_t$  正确分类则  $\varepsilon_i = 0$ , 否则  $\varepsilon_i = 1$ ,  $\beta_i = \varepsilon_i / (1 - \varepsilon_i)$ ,  $\varepsilon_i$  表示  $h_t$  对样本集错误分类的样本权重之和。从中我们可以看出, 如果前次循环生成的弱分类器  $h_t$  在某一样本点上发生误判, 那么下一轮该样本的权重将会加大, 也就表明, Adaboost 在下次迭代中, 弱学习算法将更加关注于这些权重较大的困难样本上。但是每一轮的弱学习规则以对样本集的总体误差最小的原则来生成。因而每一轮生成的弱分类器更加关注的是样本集的总体误差, 所以传统的 Adaboost 算法只能同时降低对样本集的 FNR 和 FPR。如果我们对某一类目标的正确率有着更高的要求, 可以通过改变权重调整过程来实现。

改进的 Adaboost 算法如下:

设输入为  $n$  个训练样本:  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ 。其中:  $y_i = \{0, 1\}$  对应假样本和真样本; 已知训练样本中有  $m$  个假样本,  $l$  个真样本。第  $j$  个特征生成的简单分类器形式为:

$$h_j(x) = \begin{cases} 1, & p_j f_j(x) < p_j \theta_j \\ 0, & \text{其他} \end{cases} \quad (7)$$

则生成强分类器的学习过程如下:

1) 初始化误差权重, 对于  $y_i = 0$  的样本,  $\omega_{1,1} = (1/2)m$ , 对于  $y_i = 1$  的样本,  $\omega_{1,1} = (1/2)l$ 。

2) 每个  $t = 1, 2, \dots, T$  (其中  $T$  为训练的次数):

① 权重归一化:

$$\omega_{t,i} = \frac{\omega_{t,i}}{\sum_{j=1}^n \omega_{t,j}} \quad (8)$$

② 对于每个特征  $j$ , 按照上面方法生成相应的简单分类器  $h_j$ , 计算相对于当前权重的误差:

$$\varepsilon_j = \sum_i \omega_{i,j} |h_j(x_i) - y_i| \quad (9)$$

③ 选择具有最小误差  $\varepsilon_i$  的简单分类器  $h_i$  加入到强分类器中去。

④ 更新每个样本所对应的权重:

$$\omega_{t+1,i} = \omega_{t,i} (\beta_i^{1-\varepsilon_i})^{1-FPR} \quad (10)$$

其中: 如果第  $i$  个样本  $x_i$  被正确分类, 则  $\varepsilon_i = 0$ , 反之  $\varepsilon_i = 1$ ,  $\beta_i = \varepsilon_i / (1 - \varepsilon_i)$ 。

3) 最后形成的强分类器为:

$$H(x) = \begin{cases} 1, & \sum_{i=1}^T \alpha_i h_i(x) \geq \frac{1}{2} \sum_{i=1}^T \alpha_i \\ 0, & \text{其他} \end{cases} \quad (11)$$

其中  $\alpha_i = \lg \frac{1}{\beta_i}$ 。

传统 Adaboost 算法中的单个样本权重更新过程会使得被前一轮的弱分类器  $h_i$  分类错误的那些样本的权重不断增大, 如果这些样本被错分多次, 那么他们的权重将会不断扩张, 这直接导致每轮产生的弱分类器  $h_i$  的错误率  $\varepsilon_i$  将相对增大, 这样会导致弱分类器在最后的加权投票中所占的权重变得很小, 在更严重的情况下 ( $\varepsilon_i > 0.5$ ) 会导致 Adaboost 训练的停止。将式 (10) 的权重更新过程修改为:

$$\omega_{t+1,i} = \omega_{t,i} (\beta_i^{1-\varepsilon_i})^{1-FPR} \quad (12)$$

则可以使样本权重过程更加关注于  $FPR$  的变化。 $FPR$  表示前  $t$  个弱分类器组成的强分类器对负样本集的错分率。

这种权重更新的改进方法使得那些被前一轮弱分类器错误分类的样本的权重扩张幅度减小, 其道理很简单:

$$\left. \begin{aligned} \varepsilon_i < 0.5 \Rightarrow 0 < \beta < 1 \\ 1 - FPR \leq 1 \end{aligned} \right\} \Rightarrow (\beta_i^{1-\varepsilon_i})^{1-FPR} \geq \beta_i^{1-\varepsilon_i} \quad (13)$$

上式说明那些被错分的样本权重的扩张幅度将会对  $FPR$  敏感, 其限制程度将由  $FPR$  来控制,  $FPR$  越大, 则被错分的样本权重的受限程度越大, 那么这个弱分类器的权重  $\alpha_i = \ln \frac{1}{\beta_i}$ , 也会相应增大, 从一定程度上均衡了各个弱分类器的权重, 从而加速了  $FPR$  的降低。

## 6 仿真实验

选用 Intel Celeron 2.0 GHz CPU, 256 MB 内存, Windows XP 操作系统、Matlab 7.0 语言编程环境。评价入侵检测算法时采用两个性能指标:

检测率 (Detection Rate, DR): 是指被系统检测到的入侵实例数目与检测数据集中总的入侵实例数目之比。

误检率 (False Alarm Rate, FR): 是指被误判为入侵的正常实例数目与总的正常实例数目之比。

这两项性能指标充分反映了算法的入侵检测能力, 能够对系统检测出多少入侵和对多少不正确的分类做出评价。显然, 一个好的入侵检测系统应该使 DR 尽可能的大, 而 FR 则尽可能的小。

为方便进行仿真实验, 实验时从 KDD Cup 1999 数据集<sup>[8]</sup>中随机选取 5 组具有 2 100 个数据的样本集。由于该数据集的仿真数据比较分散, 子集中攻击数量和类型分布不平衡, 导致某些子集仅包含某一种攻击类型, 这对于反映整个网络真实环境是不利的, 因此再以随机方式重新建立 5 个数据集, 使得每个数据集中有 2 000 个正常连接, 100 个异常连接, 而且使异常连接的种类尽可能平均。

我们分别选 8、10、14、16、18、20、30 作为它的训练轮数 ( $T$  值), 用前面得到的 80% 数据集进行训练, 生成相应个数的分类器, 用 20% 的数据集进行测试, 测试结果如表 1 所示; 同时也进行了基于粗糙集 (Rough Set, RS) 算法、支持向量机 (Support Vector Machine, SVM) 算法、自适应增强 (Adaptive Boost, Adaboost) 算法同基于 RS\_Adaboost 算法的入侵检测比较实验, 结果如表 2 所示。

通过不同  $T$  值实验比较可知:  $T$  值取 15 左右比较合适, 小于这个值误检率较高; 太大误检率虽低, 但循环次数增多, 耗时, 检测效率降低, 因此选  $T = 15$  较合适。

通过算法实验比较可知: RS 算法、SVM 算法的检测效率比 RS\_Adaboost 算法的检测效率稍高, 但检测率比 RS\_Adaboost 低很多; RS\_Adaboost 算法在检测率、检测效率上均比 Adaboost 算法要好。故基于 RS\_Adaboost 的入侵检测方法不失为一种好的检测方法。

表 1 不同  $T$  值的比较

| $T$ 值 | 误检率/% | $T$ 值 | 误检率/% |
|-------|-------|-------|-------|
| 8     | 3.00  | 18    | 0.95  |
| 10    | 2.10  | 20    | 0.92  |
| 14    | 1.20  | 30    | 0.86  |
| 16    | 1.08  |       |       |

表 2 入侵检测算法比较实验

| 算法             | 检测率/% | 平均规则数/s |
|----------------|-------|---------|
| RS 算法          | 88.7  | 6.2     |
| SVM 算法         | 91.3  | 5.4     |
| RS_Adaboost 算法 | 94.9  | 4.8     |
| Adaboost 算法    | 93.6  | 3.2     |

## 7 结语

通过对粗糙集理论的研究和对现有入侵检测技术和方法的研究, 本文对网络数据进行了约简, 并对 Adaboost 算法进行了改进, 提出了一种新的入侵检测方法: 基于 RS\_Adaboost 的入侵检测模型。同时, 本文提出的方法也有很多不完善的地方: 1) 找到最小的约简还须进一步研究; 2) 将这种方法应用于实际的网络中进行检测还有很多工作要做。以上这些都是今后须努力工作的方向。

### 参考文献:

- [1] KUMAR S. Classification and detection of computer intrusions[D]. West Lafayette, USA: Purdue University, 1995.
- [2] 陈伟统, 钱云涛. 基于粗糙集理论的网络入侵检测方法[J]. 计算机工程, 2006, 32(16): 133-135.
- [3] 蔡忠闻, 管晓宏, 邵萍, 等. 基于粗糙集理论的入侵检测新方法[J]. 计算机学报, 2003, 26(3): 361-366.
- [4] HOCHBERG J, JACKSON K, STALLINGS C, et al. An automated system for detecting network intrusions and misuse[J]. Computers and Security, 1993, 12(3): 253-248.
- [5] FREUND Y, SCHAPIRE R E. A decision-theoretic generalization of online learning and an application to boosting[C]. Proceedings of the 2nd European Conference on Computational Learning Theory, LNCS 904. London, UK: Springer-Verlag, 1995: 23-27.
- [6] 赵万鹏. 基于 Adaboost 算法的数字识别技术的研究与应用[D]. 北京: 中国科学院研究生院, 2006.
- [7] 朱谊强. 基于 Adaboost 算法的实时行人检测系统[D]. 西安: 西北工业大学, 2006.
- [8] University of California Irvine. KDD cup 1999 data[EB/OL]. [2008-03-01]. <http://kdd.ics.uci.edu/databases/kddcup99/kdd-cup99.html>.