·安全技术· 文章编号:1000-3428(2018)08-0161-06 文献标志码:A 中图分类号:TP391

基于 AdaBoost 算法的 Linux 病毒检测研究

吴 恋[1a,1b],马敏耀[1a,1b],黄一峰[2],赵 勇[3]

(1. 贵州师范学院 a. 数学与计算机科学学院; b. 大数据科学与智能工程研究院,贵阳 550018;
2. 重庆邮电大学 新一代宽带移动通信终端研究所,重庆 400065; 3. 北京大学深圳研究生院 信息工程学院,深圳 518055)

摘 要: 根据机器学习中分类和集成学习理论,设计一种高性能的 Linux 病毒检测方法。通过训练得到若干基分类器,然后将基分类器的分类结果进行整合,以获得最终检测结果。该方法以 ELF 文件特征为样本特征、BP 神经网络为基分类器,用 AdaBoost 算法进行基分类器的训练和整合。对 AdaBoost 算法的基分类器权重计算及基分类器整合部分进行改进,使之更适用于病毒检测问题。实验结果表明,该方法的病毒检测效果优于 Avria Linux 和 F-PROT 方法。

关键词: 病毒检测;AdaBoost 算法;BP 神经网络;ELF 文件特征;D-S 证据理论

中文引用格式: 吴 恋,马敏耀,黄一峰,等. 基于 AdaBoost 算法的 Linux 病毒检测研究[J]. 计算机工程,2018,44(8):161-166,173.

英文引用格式: WU Lian,MA Minyao,HUANG Yifeng,et al. Linux virus detection study based on AdaBoost algorithm[J]. Computer Engineering,2018,44(8):161-166,173.

Linux Virus Detection Study Based on AdaBoost Algorithm

WU Lian[1a,1b],MA Minyao[1a,1b],HUANG Yifeng[2],ZHAO Yong[3]

(1a. Department of Mathematics and Computer Science; 1b. Big Data Science and Intelligent Engineering Research Institute, Guizhou Education University,Guiyang 550018,China;2. Next Generation Mobile Communication Terminal Laboratory, Chongqing University of Posts and Telecommunications,Chongqing 400065,China;
3. School of Electronic and Computer Engineering,Peking University Shenzhen Graduate School,Shenzhen 518055,China)

【Abstract】 A high-performance Linux virus detection method is designed using the theory of classification and integration learning in machine learning. This method is, first to train and obtain a number of base classifiers, then to integrate the base classifiers' classification results to get the final detection result. The method extracts the ELF file feature as the sample feature, and selects BP neural networks as the base classifiers, and uses the improved AdaBoost algorithm to train and integrate the base classifiers. In the Adaboost algorithm, the weight calculation method and integration method of the base classifiers are improved, which makes this algorithm more suitable for the virus detection problem. Experimental results show that the effectiveness of the proposed detection method is verified by relative experiments, which is better than Avria Linux and F-PROT methods.

【Key words】 virus detection; AdaBoost algorithm; BP neural network; ELF file feature; D-S evidence theory

DOI: 10.19678/j.issn.1000-3428.0048395

0 概述

随着大数据及智能化时代的到来,稳定性好、性能高的 Linux 操作系统已逐渐成为当下应用主流,针对 Linux 的各种病毒攻击也越来越频繁。但是由于各种原因,Linux 上的病毒检测工具远不如 Windows 平台上的丰富和有效,而且相关的理论研究也不多[1-2]。因此,对于 Linux 病毒检测技术的研究具有较大的现实意义。

本文以机器学习中分类和集成学习问题的处理框架为基础,结合 Linux 操作系统的 ELF 文件特点,基于 AdaBoost 算法设计并实现了一种高性能的 Linux 病毒检测方案。

基金项目: 贵州省教育厅自然科学研究项目(青年项目)(黔教合 KY 字【2015】425);贵州省省级重点支持学科"计算机应用技术"(黔学位合字 ZDXK【2016】20 号);贵州省科技平台及人才团队专项资金(黔科合平台人才【2016】5609);贵州省科学技术基金(黔科合基础【2016】1115)。

作者简介: 吴 恋(1988—),女,讲师、硕士,主研方向为网络安全、机器学习;马敏耀,副教授、博士;黄一峰,工程师、硕士;赵 勇,教授、博士。

收稿日期: 2017-08-17 **修回日期:** 2017-11-03 **E-mail:** 373201377@qq.com

1 基于 AdaBoost 的 Linux 病毒检测方案

简单的病毒检测方案设计如图 1 所示。采用特征提取方法从待测文件中得到样本特征,然后通过某种分类算法进行判断,最后得到检测结果。但这种方案存在不足,单个分类器分类能力是有局限的,其检测准确率不是太高,且容易对特定样本产生过拟合性。

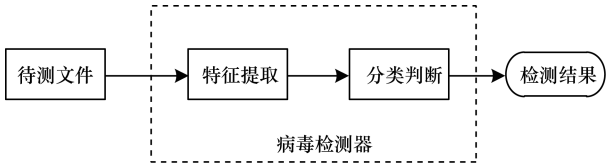

图 1 简单的病毒检测方案

为了解决上述方案的不足,本文在分类学习问题处理框架的基础上,引入集成学习的思想,综合多个分类器的检测结果,通过整合的方式来提高检测精度[3]。本文设计的 Linux 病毒检测方案如图 2 所示,总的检测流程如下:

1)从待测文件中提取出样本特征。

2)从基分类器 1 至基分类器 n,每个基分类器都独立地对待测样本进行检测,得到各自的检测结果。

3)对所有基分类器的检测结果进行整合,得到系统的最终检测结果。

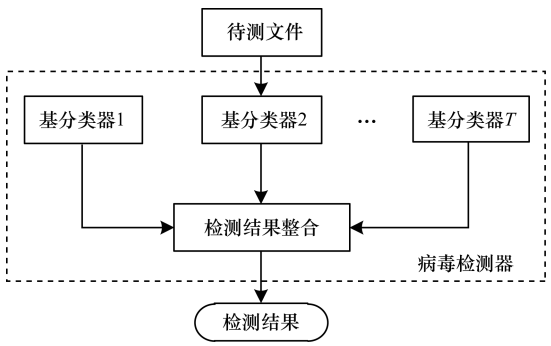

图 2 本文 Linux 病毒检测方案

本文的病毒检测方案是构建在简单的病毒检测方案之上的,从单个分类器变成了多个有差异的分类器集成组合,所以,涉及到的技术问题除了文件样本特征提取、基分类器的设计外,还有一个关键的技术点就是关于基分类器的训练和整合。

1.1 样本特征的提取

样本特征选择的标准是要能最大程度地反映出病毒和正常文件的区别。本文的研究目标是 Linux 操作系统病毒,所以,在特征选取方面尽量考虑 Linux 操作系统独有的特性。本文选择了 ELF 文件的头表信息作为样本特征来源。

执行与连接格式(Executable and Linking Format, ELF)文件格式是 Linux 操作系统下一种主要的目标文件格式,Linux 系统下所有的可执行文件和库文件都必须符合 ELF 格式的标准。ELF 头表中描述了文件的整体结构,包括当前文件类型、版本号、节头表和段头表的偏移量等。节头表(Section Header Table)记录的是编译器和链接器需要使用的每一个节的描述信息,包括指令节、数据节、符号表节等。段头表(Program Header Table)记录的是程序段的内容,包含着程序文件到内存的映射信息。以上的各部分头表信息决定了目标文件如何与其他目标文件连接,载入内存以及如何执行等[4-5]。对于可执行文件型的 Linux 病毒而言,在将自身病毒体代码插入寄主体之前,必须修改寄生体的头、节头表、段头表里面的信息,否则是无法完成寄生宿主、驻留内存等工作的。而对于蠕虫等能独立运行的 Linux 病毒来说,为了完成其破坏行为,在其 ELF 文件信息中也必定包含一些与正常文件不同的地方,例如在代码段中访问非法的内存地址、反复读取系统参数等,这些行为最终也会反映在头表信息中。因此,可以使用 ELF 文件的头表信息作为样本特征来源。在 Linux 操作系统下,可以使用 readelf 工具读取 ELF 文件的头表信息,本文从中提取了可能和病毒检测相关的 78 个特征属性,如图 3 所示。

属性描述		数据类型	个数
ELF header		Integer	13
Section header table	Text header	Integer	8
	Data header	Integer	8
	BSS header	Integer	8
	Symbol header	Integer	9
Program header table	PT_LOAD1 header	Integer	8
	PT_LOAD2 header	Integer	8
	PT_DYNAMIC header	Integer	8
	PT_INTERP header	Integer	8

图 3 提取的 ELF 文件头表特征

1.2 BP 神经网络基分类器的设计

在有了待选的样本特征来源之后,还需要确定基分类器。分类算法的选择对分类结果有重大的影响,病毒的检测实质上就是机器学习理论中的一个分类问题。分类问题的最终目的是要训练得到一个分类器,以便使用这个分类器来对待分类的样本进行分类。本文选用 BP 神经网络作为基分类器,BP 神经网络在病毒的检测应用上已有相关文献证实其可行性和优越性[6-7]。神经网络算法的结构特点使其比其他分类算法具有更好的容噪性和健壮性。本文的 BP 神经网络网络结构设计为 $36 \times 20 \times 12$,学习算法为误差反向传播算法,学习率设为 $l_r = 0.1$,最大学习次数为 20 000 次,学习目标误差平方和为 $Err\_goal = 10^{-5}$。

1.3 基分类器的训练和整合方案

系统的检测结果是通过综合多个基分类器的分类结果而得来的,所以,基分类器之间应该是各不相

同的,否则对同一个样本,基分类器的分类结果都差不多,便失去了整合的意义。根据集成学习理论,基分类器的产生方式有异态学习和同态学习 2 种。一般来说,异态集成学习需要实现多个不同的基分类器分类算法,系统开发的工作量是比较大的,因此,本文采用同态集成学习的方法来得到基分类器。在同态学习中,每个基分类器的分类算法是不变的,问题便转化到如何调整基分类器分类算法的训练过程上来。有一种思路是通过对样本集进行某种处理,使得每次在对分类算法进行训练时的样本集都是不同的,从而得到不同的分类器[8]。本文采用了 AdaBoost 算法来实现基分类器的生成及整合,并对 AdaBoost 算法的基分类器权重计算及基分类器整合部分进行了算法改进,使之更契合于病毒检测问题。

2　AdaBoost 算法及其改进

2.1　原 AdaBoost 算法

Adaboost 是一种迭代算法,其核心思想是针对同一个训练集训练不同的分类器(基分类器),然后把这些基分类器集合起来,构成一个更强的最终分类器(强分类器)[9]。在训练过程中,每个训练样本被赋予一个初始权值,当一个基分类器训练完成后,根据其在训练集上的分类结果对所有的样本权值进行调整,使得下一次训练的基分类器更关注那些被识别错误的样本。最后的强分类器的判决结果是所有分类器的判决结果的加权和。

AdaBoost 算法的流程如下[7,9]:

步骤 1　给定训练样本集:

$S=\{(x_1,y_1),(x_2,y_2),\cdots,(x_i,y_i),\cdots,(x_m,y_m)\}$

其中,x_i 是实例样本,有 $x_i\in X$;y_i 是类别标志,$y_i\in Y=\{-1,+1\}$,m 表示当前训练样本有 m 个。

步骤 2　对样本权重进行初始化:

$D_t(i)=1/m$

步骤 3　循环 $t=1,2,\cdots,T$(T 表示设置了 T 个基分类器):

1)在当前的样本权重分布 D_t 下,训练得到基分类器:$h_t=H(x,y,D_t)$。

2)计算该基分类器的错误率:

$$\varepsilon_t=P(h_t(x_i)\neq y_i)=\sum_{i=1}^{m}D_t(i)[h_t(x_i)\neq y_i]$$

3)计算得到基分类器的权重:

$$a_t=\ln(1-\varepsilon_t)/\varepsilon_t$$

4)更新样本权重:

$$D_{t+1}(i)=(D_t(i)/Z_t)\times\begin{cases}e^{-a_t},h_t(x_i)=y_i\\ e^{a_t},h_t(x_i)\neq y_i\end{cases}$$

其中,Z_t-为归一化因子。

步骤 4　得到强分类器:

$$H(x)=sign(\sum_{t=1}^{T}a_th_t(x))$$

从算法流程可以看出,AdaBoost 算法在每一轮中不仅更新了样本的权重,还计算出了每轮生成的基分类器的权重,权重更高的基分类器将获得更大的话语权。

2.2　AdaBoost 算法改进

在 AdaBoost 算法中,基分类器的权重由分类错误率给出,如下:

$$a_t=\ln(1-\varepsilon_t)/\varepsilon_t \tag{1}$$

其中,a_t 为基分类器的权重,ε_t 为该基分类器的分类错误率。

基分类器的分类错误率越大,其获得的权重就越小。在病毒检测中,分类错误也即是把病毒识别为了正常文件或是把正常文件识别成了病毒。对于病毒检测来说,系统对正样本(病毒)的识别率是比负样本更为重要的,即所谓宁可错杀,不可轻易放过。因此,如果能够在基分类器权重计算时加入基分类器对正样本识别能力强弱的考虑,则算法能更好地适用于病毒检测应用场景[10-11]。为此,本文定义了对正样本识别能力的变量 ϕ_t 如下:

$$\phi_t=\sum_{y_i=1,h_t(x_i)=1}D_t(i) \tag{2}$$

而对基分类器权重重新定义为式(3),其中,ξ 为正常数。

$$a_t=\ln(1-\varepsilon_t)/\varepsilon_t+\xi\cdot e^{\phi_t} \tag{3}$$

其中,ϕ_t 是第 t 轮训练时基分类器中所有被识别为正样本的权重之和,它能够反映基分类器对正样本的识别能力。a_t 是第 t 轮训练时基分类器的权重,即第 t 个基分类器的权重($t\in[1,T]$)。

此外,原始的 AdaBoost 算法采用加权投票的方式来整合基分类器结果,但是由于训练样本通常是有限的,部分基分类器在被训练时可能会存在过拟合现象,对此本文引入 D-S 理论的方法改进 AdaBoost 算法中基分类器的整合方式,降低那些存在过拟合现象的基分类器对整合结果的影响,进一步提高整合后的效果。

改进后的算法描述如下:

步骤 1　给定训练样本集

$S=\{(x_1,y_1),(x_2,y_2),\cdots,(x_i,y_i),\cdots,(x_m,y_m)\}$

其中,x_i 是实例样本,有 $x_i\in X$;y_i 是类别标志,$y_i\in Y=\{-1,+1\}$,m 表示当前训练样本有 m 个。

步骤 2　对样本权重进行初始化:$D_t(i)=1/m$。

步骤 3　循环 $t=1,2,\cdots,T$(T 表示设置了 T 个基分类器):

1)在当前的样本权重分布 D_t 下,训练得到基分类器:$h_t=H(x,y,D_t)$。

2)计算该基分类器的错误率:

$$\varepsilon_t=P(h_t(x_i)\neq y_i)=\sum_{i=1}^{m}D_t(i)[h_t(x_i)\neq y_i]$$

3)计算对正样本的识别正确率:

$$\phi_t=\sum_{y_i=1,h_t(x_i)=1}D_t(i)$$

4)计算得到基分类器的权重:

$a_t = \ln(1-\varepsilon_t)/\varepsilon_t + \xi \cdot e^{\phi_t}$

5)更新样本权重:

$$D_{t+1}(i) = (D_t(i)/Z_t) \times \begin{cases} e^{-a_t}, h_t(x_i) = y_i \\ e^{a_t}, h_t(x_i) \neq y_i \end{cases}$$

其中,Z_t 为归一化因子。

步骤 4 得到强分类器:

$H(x) = D - S_{\text{theory}}(a_t, h_t(x))$

2.3 改进算法分析

通过上小节的讨论可以看出,本文对 AdaBoost 算法的改进有 2 点:

1)对基分类器的权重计算进行了改进,将基分类器对正样本(病毒)的分类能力考虑到基分类器权重分配中。

基分类器权重被重新定义为:

$a_t = \ln(1-\varepsilon_t)/\varepsilon_t + \xi \cdot e^{\phi_t}$

其中,$\phi_t = \sum_{y_i=1, h_t(x_i)=1} D_t(i)$ 是第 t 轮训练时基分类器中所有被正确识别为正样本的权重之和,它能够反映基分类器对正样本的识别能力。而式(3)中的 $\xi \cdot e^{\phi_t}$ 是 ϕ_t 增函数,通过这种方式求得新的基分类器权重,在分类错误率相同的情形下,那些具有更高正样本识别能力的基分类器将被赋予更大的权重[10]。整个检测系统对正样本的识别能力也因此提高。

2)对基分类器的整合进行了改进,将带权重投票的整合的方式改为 D-S 证据理论整合的方式。

基于 D-S 证据理论整合如图 4 所示,主要包括证据空间和证据合成规则[12-14]。在 D-S 理论中,证据合成规则是固定的,所以,关键就在于怎么结合本文的病毒检测系统来生成证据空间。

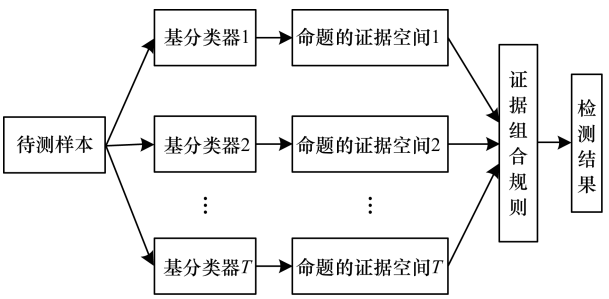

图 4 基于 D-S 证据理论的整合

在 D-S 证据理论应用到 Linux 病毒检测的过程中,本文的定义如下[13-14]:

1)命题、传感器和命题空间

病毒检测系统共包括 2 个基本命题:(1)样本为正常文件,记为 N;(2)样本为病毒,记为 ΔN,且 $N \cap \Delta N = 0$。所以,命题空间为 $A = \{N, \Delta N\}$,其中,N 表示对样本判定为正常文件的信任,ΔN 对样本判定为病毒的信任。传感器为基分类器,可以对基本命题做出判断。对于一个给定的测试样本,每个基分类器都会做出对其类型的判断,即信任其正常文件(N)或者信任其为病毒(ΔN)。

2)证据空间

根据病毒检测系统命题空间的定义,得出其基本概率函数有:

$$M: 2^{\{N, \neg N\}} \rightarrow [0,1] \tag{4}$$

其中:

$$M(\{N, \neg N\}) = 1 - M(N) - M(\neg N) \tag{5}$$

每个命题中不再包含子集,所以,命题的信任函数就是它的基本概率函数,反映了对它的基本信任度分配,即:

$$Bel(A) = M(A), A = \{N, \neg N\} \tag{6}$$

同时,似然函数为:

$$P_i(N) = 1 - Bel(\neg N) = 1 - M(\neg N) \tag{7}$$

从以上分析可以看出,对命题整合的关键在于对每个基分类器的基本概率函数的定义。在 AdaBoost 算法中,对每个基分类器都赋予了投票权重,作为对其分类表现的评价。因此,可以把每个将基分类器的权重作为其基本概率函数。例如 2 个基分类器 m_1、m_2 的权重分别为 a_1、a_2,在一次检测中,它们对待测样本的判断分别为 N 与 $\neg N$,则它们的基本概率函数定义如表 1 所示。同理,可以对多个基分类器的基本概率函数做出定义。

表 1 2 个基分类器的基本概率分配

命题	基分类器 m_1	基分类器 m_2
N	a_1	$1-a_2$
$\neg N$	$1-a_1$	a_2

3)合成规则

对由 T 个基分类器组成的检测系统,采用 Dempster 正交合成规则来对命题的基本概率函数进行合成,得到检测系统对该命题的信任度分配,如下:

$$M(A) = m_1(A) \oplus m_2(A) \oplus \cdots \oplus m_T(A) \tag{8}$$

最后,观察检测系统对命题 $\neg N$ 的信任度大小,当信任度超过设定的阈值(例如 0.5)时,判定待测文件为病毒,否则判定其为正常文件。

在 D-S 证据理论引入后,检测系统对病毒的检测从简单的“是“与“否”的判断转化为可量化的数值输出。例如当对检测敏感度要求比较高时,可以把病毒判定的阈值调低,从而将更多可疑文件判定为病毒,反之亦然。通过这种方式,使系统更加适用于不同的检测场景。

综上所述,改进后的 AdaBoost 算法在基分类器权重计算中加入正样本(病毒)识别能力的影响,让 AdaBoost 算法变得更适用于病毒检测的应用。其使

得在分类错误率相同的情形下,那些具有更高正样本识别能力的基分类器将被赋予更大的权重,整个检测系统对正样本的识别能力也因此提高。在 AdaBoost 算法的基分类器整合问题中改进引入 D-S 理论方法,进一步提高多个基分类器结果的整合效果,使最终的判决结果正确率更高。

3　实验结果与分析

3.1　实验样本集

实验的样本集共包含 1 200 个样本,其中,正常文件 700 个,病毒文件 500 个。病毒样本是从专门提供病毒样本的 www. vxheaven. org 网站上下载的[15],包括蠕虫、后门、特洛伊木马等病毒类型共 500 个,其各类型病毒占比情况如表 2 所示。正常文件样本主要采集于 Ubuntu 和 Android 操作系统的系统文件,以及流行 Linux 应用程序的可执行文件和库文件,共 700 个,其各类型分布情况如表 3 所示。正常样本和病毒样本的大小信息如表 4 所示。

表 2　病毒样本集中各类 Linux 病毒占比情况　%

病毒类型	占比
Backdoor	24.60
Dos	18.00
Rootkit	22.40
Worm	14.80
Trojan	10.00
Other	10.20

表 3　正常文件样本集中各类正常文件占比情况　%

正常文件类型	占比
Ubuntu 系统文件	30.00
Android 系统文件	30.00
流行 Linux 程序	40.00

表 4　正常样本和病毒样本的大小信息　KB

类型	最大	最小	平均
病毒	1 280	126	467
正常文件	9 897	234	2 083

3.2　实验内容

实验内容具体如下:

1)本文病毒检测方案的可行性验证

将上述所述实验样本 80% 的样本(病毒文件、正常文件均匀取 80%)作为训练集训练基分类器 BP 神经网络,剩下的 20% 样本集作为测试集验证本文病毒检测方案的检测效果。本文在 Ubuntu 操作系统上,用 C 语言编程实现了本文所述的病毒检测方案。代码主要分为 2 个部分,一个是用于训练基分类器的训练代码,另一个是用于训练完成后进行检测的检测代码。训练过程首先完成对训练样本集的特征表示,即提取每个样本的 ELF 文件头特征来代表每个样本文件,将得到的训练样本特征集作为 AdaBoost 算法的样本集,利用算法描述中的前 3 个步骤训练得到 T 个 BP 基分类器,并得到各自权重。检验过程首先同样先完成检测样本集的特征表示,然后用每个检测样本的 ELF 文件特征作为基分类器的输入,分别由 T 个基分类器对同一样本进行检测,得到 T 个结果,应用算法中第 4 步 D-S 证据整合得到该样本的最终判决结果。其实验流程如图 5 所示。

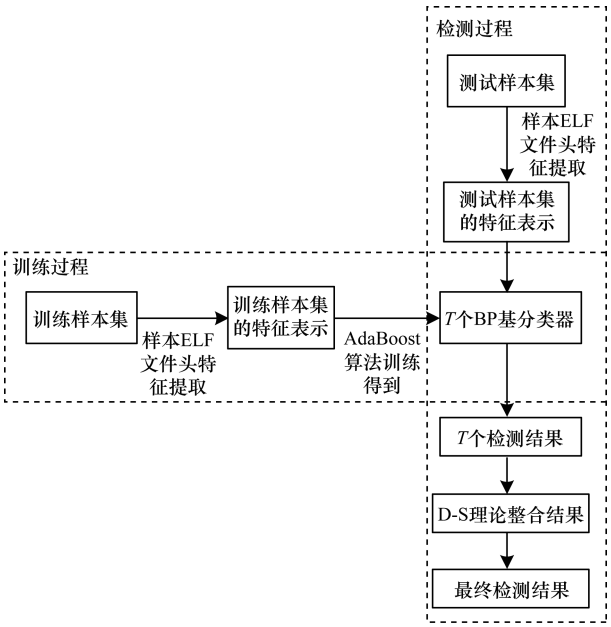

图 5　实验运行流程

实验测试了在设置不同数量基分类器情况下的检测性能,表 5 为设置不同基分类器个数下的检测正确率,本文中的正确率是指所有被正确分类的样本在待测样本中的比例,其计算公式为:

$$TA = \frac{TP + TN}{TP + TN + FP + FN}$$

其中,TP 表示病毒样本被正确检测出来的数目,TN 表示正常文件被正确检测出来的数目,FP 表示正常文件被错误判定为病毒的数目,FN 表示病毒样本被误判为正常文件的数目。

表 5　设置不同数目基分类器的检测准确性

基分类器个数	正确率/%
1	86.2
2	89.3
4	93.6
8	96.9
16	97.1
32	97.2

2)改进后的 AdaBoost 算法检测效果验证

本文针对 AdaBoost 算法是否改进,编程实现了 2 套代码,一套是采用原始 AdaBoost 算法思路实现

的训练代码 1 和检测代码 1,另一套是采用改进后的 AdaBoost 算法思路实现的训练代码 2 和检测代码 2。训练代码 1 与训练代码 2 实现的区别在于每次循环中基分类器的权重计算时是否加入了对正样本(病毒)的分类能力考虑,即计算中加上了 $\xi \cdot e^{\phi_t}$ 项;检测代码 1 和检测代码 2 实现的区别在于整合所有基分类器检测结果的方式不同,检测代码 1 是简单的将所有基分类器的检测结果加权和作为整合结果(即最终的检测结果),而检测代码 2 是采用 D-S 证据理论思想来整合所有基分类器结果的,代码实现主要包括计算归一化参数、计算证据空间(基分类器的基本概率函数)、证据合成 3 个步骤,代码实现思路如下:

步骤 1 计算归一化函数,定义为变量 C:

$$C = 1 - \sum_{\cap A_i = \varphi} \prod_{1 \leqslant i \leqslant n} M_i(A_i) = \sum_{\cap A_i \neq \varphi} \prod_{1 \leqslant i \leqslant n} M_i(A_i)$$

步骤 2 计算证据空间,其值为:

$$M(A) = C^{-1} \sum_{\cap A_i = A} \prod_{1 \leqslant i \leqslant n} M_i(A_i), A \neq \varphi$$

步骤 3 正交合成规则来对命题的基本概率函数进行合成,公式如下:

$$M = m_{(1)} \oplus m_{(2)} \oplus \cdots \oplus m_{(T)}$$

实验将未改进的 AdaBoost 检测方法和改进后的 AdaBoost 检测方法其基分类器个数都设置为 16 个,实验的训练样本和测试样本均采用了如上所述的相同样本集,其各自检测效果如表 6 前两行所示。

表 6 本文检测方法与其他杀毒软件对比 %

检测系统	正确率
本文检测方法(改进的 AdaBoost)	97.1
本文检测方法(未改进的 AdaBoost)	92.6
Avria Linux	96.2
F-PROT	94.8

同时,本文对市场上常见的 2 款 Linux 病毒检测软件(Avria Linux 和 F-PROT)[16] 也进行了比对测试,表 6 中后两行为 Avria Linux 和 F-PROT 的检测结果。Linux 版本的 Avria 杀毒软件是 Linux 平台上最受欢迎的杀毒软件之一,是一款免费的杀毒软件,它的使用模式基本上和平时在 Windows 平台所使用的金山毒霸、KILL、瑞星等杀毒软件的 DOS 版本的使用方法是一样的,用户选择不同的扫描参数可以让 Avria 执行不同的任务,在使用 Avria 扫描完系统后,Avria 会显示一个扫描清单,用户可以清楚知道 Avria 的扫描结果。本文在 Ubuntu 系统下安装了 Linux 版本的 Avria 杀毒软件,并让其扫描本文实验样本集中的测试样本集文件夹,得到了其扫描结果,其正确率为 96.2%。本文对比测试的另一款 Linux 杀毒软件是 F-PORT,对家庭用户免费,它有使用克龙(cron)工具的任务调度的特性,能在指定时间执行扫描任务,同时它还可以扫描 USB HDD、Pendrive、CD-ROM、网络驱动、指定文件或目录、引导区病毒扫描、镜像。本文同样在 Ubuntu 系统下安装了该杀毒软件,并指定让其扫描本文实验样本集中的测试样本集文件夹,得到其病毒识别正确率为 94.8%。

3.3 结果分析

从表 5 可以看出:1)当基分类器数量为 1 个时,相当于没有进行集成学习,此时的检测系统正确率能达到 85% 以上,说明本文所选用的 ELF 文件特征、BP 神经网络分类算法是比较适用于 Linux 病毒检测问题的;2)当基分类器数量从 2 个到 16 个依次递增时,检测系统的正确率有较大程度的提升,说明集成学习对系统检测性能的提升起到了明显的作用,本文的病毒检测方案比传统简单的单分类器方案好;3)当基分类器数量从 16 增加到 32 个时,准确率不再有明显的提升,说明基分类器数量的不断增加并不能无限地提升系统的检测性能,在性能不会再有明显提升的情况下,无需牺牲大量计算量和时间来设置更大的基分类器数。

从表 6 可以看出,本文对 AdaBoost 算法进行改进后其在病毒检测效果上有明显提升,改进后的 AdaBoost 算法更适用于病毒检测问题。本文提出的病毒检测方案其检测效果优于当前市场上已有的两款 Linux 病毒检测软件。

4 结束语

本文设计一种 Linux 病毒检测方案,选取了 ELF 文件的头表信息作为样本特征,分类器采用了 BP 神经网络,通过 AdaBoost 算法利用有限的样本集训练出多个基分类器。在 AdaBoost 算法的基分类器整合问题中引入 D-S 理论,进一步提高整合效果。实验结果表明,该检测方法取得了较好的效果。下一步将尝试使用除了 ELF 文件头表信息外的其他特征属性,如系统性能指标、网络流量指标等,以图像的方式来处理病毒,并引入深度学习研究病毒检测问题。

参考文献

[1] KEHI W,GE Yueguang,CHEN Wei. The research and implementation of the linux process real-time monitoring technology[C]//Proceedings of International Conference on Computational and Information Sciences. Washington D. C., USA:IEEE Press,2012:1046-1049.

[2] 冯文林. 虚拟环境下病毒查杀问题研究[J]. 计算机工程,2016,42(11):182-188.

[3] TOMOVSKI I, KOCAREV L. Simple algorithm for virus spreading control on complex networks[J]. IEEE Transactions on Circuits and Systems I:Regular Papers, 2012,59(4):763-771.

[4] 谢 瑶,潘剑峰,朱 明. Linux/Unix 环境中 ELF 格式病毒的分析[J]. 计算机工程,2005,31(4):166-168.

(下转第 173 页)

APPScan,说明在攻击向量与合法向量的选取上有较强的能力。

4　结束语

本文在分析跨站脚本攻击原理的基础上,提出一种动态检测反射型与存储型跨站脚本方法,设计并实现了一款自动化检测模型。该模型与同类检测工具相比,能够降低误报率和漏报率,从而更加有效地发现 Web 应用中的跨站脚本漏洞。下一步将针对 Dom-based XSS 进行分析与研究,使所提模型能够检测出该类型的跨站脚本漏洞。

参考文献

[1] 邱永华. XSS 跨站脚本攻击剖析与防御[M]. 北京:人民邮电出版社,2013.

[2] PIETRASZEK T, BERGHE C V. Defending against injection attacks through context-sensitive string evaluation[C]//Proceedings of International Conference on Recent Advances in Intrusion Detection. Berlin, Germany:Springer-Verlag,2005:124-145.

[3] ISMAIL O, ETOH M, KADOBAYASHI Y, et al. A proposal and implementation of automatic detection/collection system for cross-site scripting vulnerability[C]//Proceedings of International Conference on Advanced Information Networking and Applications. Washington D. C. ,USA:IEEE Press,2004:145-151.

[4] 王　丹,赵文兵,丁治明. Web 应用常见注入式安全漏洞检测关键技术综述[J]. 北京工业大学学报,2016,42(12):1822-1832.

[5] JOVANOVIC N,KRUEGEL C,KIRDA E. Pixy:a static analysis tool for detecting web application vulnerabilities[C]//Proceedings of IEEE Symposium on Security and Privacy. Washington D. C. , USA: IEEE Press,2006:136-263.

[6] CEPONIS J, CEPONIENE L, VENCKAUSKAS A, et al. Evaluation of open source server-side XSS protection solutions[J]. Communications in Computer & Information Science,2013,403:345-356.

[7] DONG Guowei, ZHANG Yan, WANG Xin, et al. Detecting cross site scripting vulnerabilities introduced by HTML5 [C]//Proceedings of International Joint Conference on Computer Science and Software Engineering. Washington D. C. , USA: IEEE Press, 2014:319-323.

[8] GUO Xiaobing, JIN Shuyuan, ZHANG Yaxing. XSS vulnerability detection using optimized attack vector repertory[C]//Proceedings of International Conference on Cyber-enabled Distributed Computing and Knowledge Discovery. Washington D. C. , USA: IEEE Press, 2015: 29-36.

[9] 钟晨鸣,徐少培. Web 前端黑客技术揭秘[J]. 计算机安全,2013(1):47.

[10] 吴子敬,张宪忠,管　磊,等. 基于反过滤规则集和自动爬虫的 XSS 漏洞深度挖掘技术[J]. 北京理工大学学报,2012,32(4):395-401.

[11] ROBERT H. XSS filter evasion chea-tsheet[EB/OL]. [2016-12-21]. https://www. owasp. org/index. php/XSS_Filter_Evasion_Cheat_Sheet.

[12] 潘古兵,周彦晖. 基于静态分析和动态检测的 XSS 漏洞发现[J]. 计算机科学,2012,39(6):51-53.

[13] SHAHRIAR H, ZULKERNINE M. MUTEC: mutation-based testing of cross site scripting[C]//Proceedings of ICSE Workshop on Software Engineering for Secure Systems. Washington D. C. , USA: IEEE Computer Society,2009:47-53.

[14] 沈寿忠,张玉清. 基于爬虫的 XSS 漏洞检测工具设计与实现[J]. 计算机工程,2009,35(21):151-154.

[15] Bloom filter [EB/OL]. [2016-11-21]. https://en. wikipedia. org/wiki/Bloom_filter.

编辑　刘　冰

(上接第 166 页)

[5] NARAYANAN A,CHEN Yongsheng, PANG Shuping. The effects of different representations on static structure analysis of computer malware signatures[J]. Scientific World Journal,2013,9(10):1155-1160.

[6] 郭　晨,梁家荣,梁美莲. 基于 BP 神经网络的病毒检测方法[J]. 计算机工程,2005,31(2):152-156.

[7] 吴俊利,张步涵,王　魁. 基于 Adaboost 的 BP 神经网络改进算法在短期风速预测中的应用[J]. 电网技术,2012,36(9):221-225.

[8] XU Lei, LI Yanda. Machine learning and intelligence science [J]. Frontiers of Electrical and Electronic Engineering in China,2012,7(11):1-4.

[9] 杨新武,马　壮,袁　顺. 基于弱分类器调整的多分类 Adaboost 算法[J]. 电子与信息学报,2016,38(2):373-380.

[10] 李　闯,丁晓青,吴佑寿. 一种改进的 AdaBoost 算法——AD AdaBoost[J]. 计算机学报,2007,30(1):103-109.

[11] 邱仁博,娄　震. 一种改进的带参数 AdaBoost 算法[J]. 计算机工程,2016,42(7):199-202,208.

[12] 刘明亮,甄建聚,孙来军,等. 基于 DS 证据理论的 SVM 分类模糊域数据修正[J]. 电力自动化设备,2012,32(3):71-75.

[13] 费　翔,周　健. 一种处理冲突证据的 D-S 证据权重计算方法[J]. 计算机工程,2016,42(2):142-145.

[14] 廖　平,郑友娟,覃才珑. 基于改进证据理论的全信息故障诊断[J]. 计算机工程,2016,42(3):308-311,316.

[15] BAYOGLU B, SOGUKPINAR I. Polymorphic worm detection using strong token-pair signatures[J]. Turkish Journal of Electrical Engineering and Computer Sciences,2012,17(2):163-182.

[16] YERIMA S Y, SEZER S, MCWILLIAMS G. Analysis of Bayesian classification based approaches for android malware detection[J]. IET Information Security,2013,8(1):25-36.

编辑　刘　冰