面向异构化平台的轻量级程序异常检测方法

马海龙 尹梓诺* 胡 涛

(中国人民解放军战略支援部队信息工程大学 郑州 450001)

摘 要：针对现有异常检测方法因为预学习以及噪声敏感所导致的检测时间长和误报率高的问题，该文通过对现有异常检测案例进行分析，从平台异构性角度提出了一种异常检测的新方法：将程序分别运行在多个异构平台，正常程序的所有平台运行结果相同，而异常程序在不同平台显示出差异性。基于此，该文设计了一种面向异构化平台的轻量级程序异常检测方法，收集系统状态数据并利用特征工程构建对异常表征明显的多维矢量，采用标签编码和Max-Min归一化对数据预处理，计算数据间差异度并应用阈值规则比较分析判别异常。相比于无监督特征聚类方法，所提方法的检测准确率提升了13.12%且具有低误报率和较短的检测时间。

关键词：程序异常检测；异构平台；系统状态特征；差异性

中图分类号：TN918; TP393 **文献标识码**：A **文章编号**：1009-5896(2022)02-0602-09

DOI: 10.11999/JEIT210152

A Lightweight Program Anomaly Detection Method for Heterogeneous Platform

MA Hailong YIN Zinuo HU Tao

(*Information Engineering University*, *PLA Strategic Support Force*, *Zhengzhou* 450001, *China*)

Abstract: The existing anomaly detection methods which require pre-learning and are sensitive to noise result in long detection time and high false positive rate. Based on the analysis of the existing anomaly detection cases, a new perspective is proposed from platform heterogeneity: programs are run on multiple heterogeneous platforms, normal programs are run on all platforms with the same result, while anomaly programs show heterogeneity on different platforms. So a lightweight program anomaly detection method for heterogeneous platforms is designed. System state data is collected. Feature engineering is used to construct a multidimensional vector with obvious representation of anomaly. The label code and max-min normalization are used to preprocess the data. The difference degree between the data is calculated and the threshold rule is used to compare, analyze and detect anomaly. Compared with the unsupervised feature clustering method, detection accuracy of the proposed method is improved by 13.12% with low false positive rate and short detection time.

Key words: Program anomaly detection; Heterogeneous platforms; System status features; Diversity

1 引言

随着计算机科学和互联网技术的迅猛发展，各种各样的计算机程序呈现爆炸式增长。但与此同时，主机等信息系统也面临大量恶意程序的威胁，国家互联网应急中心(CNCERT)发布的《2019年中国互联网网络安全报告》中指出2019年全年捕获计算机恶意程序样本数量超过6200万个，仍有大量主机遭受恶意程序控制从而引发信息泄漏等一系列安全危害。一些恶意攻击者通过借助主机中的一些漏洞来编写恶意程序进行攻击，使得用户主机崩溃或被黑客控制，造成较大的安全问题[1]。

近年来，有多种方法被提出，以检测带有恶意企图的程序。这些技术可以分为两大类：静态分析和动态分析[2]。静态分析(基于签名的检测)提取代码本身的特征并搜索恶意标志，如shell代码或与已知恶意程序样本相似的特征进行异常识别[3]。Ma等人[4]提出一种组合静态行为的程序异常分析方法，使用3种静态特征，运用C4.5，长短期记忆网络(Long Short-Term Memory, LSTM)以及深度神经网络(Deep Neural Networks, DNN)3种模型进行训练，结合投票法进行异常样本检测。静态分析虽应用广泛，但存在一定缺陷：部分攻击者对程序模糊

收稿日期：2021-02-18；改回日期：2021-05-22；网络出版：2021-06-04
*通信作者： 尹梓诺 417081026@qq.com
基金项目：国家重点研发计划(2018YFB0804002, 2017YFB0803204)
Foundation Items: The National Key R&D Program of China(2018YFB0804002, 2017YFB0803204)

处理使恶意程序难以被检测出来，导致静态分析技术检测效率大大下降，因此仅依赖静态技术无法识别异常程序。动态分析监控程序执行过程中产生的行为，判断是否存在恶意行为[5]。张若楠等人[6]提出一种融合改进的K-means和K近邻 (K-Nearest Neighbor, KNN)的攻击程序检测方法(I2K)来改善检测的准确率和速度，但K-means和KNN算法都对数据噪声较为敏感；汪洁等人[7]提出基于子图相似性的恶意程序检测方法来减少恶意程序检测的时间开销；陈志峰等人[8]提出一种基于聚类分析的内核恶意程序检测方法来提高恶意软件检测效率。Yoo等人[9]提出一种基于机器学习的混合决策模型，该模型结合随机森林和深度学习模型，可以对恶意程序实现高检测率和低误报率。但是这类结合机器学习的检测方法需要进行大量的预先训练和学习，且具有一定的误报率。基于上述分析可以得出，已有的异常检测方法面临如下问题：需要预先对正常行为模式进行大量学习，时间开销较大；无法有效消除噪声干扰，导致检测准确率低以及误报率高。

针对现有研究中存在的问题，本文从异构平台对应用程序的影响角度出发，研究基于平台异构性的程序异常检测。由于异构化平台在指令集、字节序、内存布局等方面的特异性、差异性，应用程序在某个平台出现的特定错误在其他异构平台中通常不会出现。研究表明，对攻击者而言，跨平台的漏洞利用比针对单个平台的漏洞利用更难实现，并且在多个异构平台间发现相同漏洞的难度很高[10]。Garcia等人[11]分析了10种操作系统发布版在过去18年内发现的漏洞数据，研究结果表明，不同操作系统出现公共漏洞的数量极少。异常程序通常只会作用于特定平台、主机、系统等，当它作为输入同时运行在由多个主机组成的异构平台环境时，只会令特定目标主机出现恶意行为，而其他与之异构的主机正常运行不受影响。通过对比差异化运行结果，可以快速检测出异常程序并准确定位作用域。

基于此，本文提出一种面向异构化平台的轻量级程序异常检测方法，不同于已有静态分析和动态分析检测方案依靠异常样本构造启发式算法，它利用平台异构性检测异常程序，其核心思想是：将安全风险未知的程序同时运行在异构化平台上，如果某个平台的行为特征与其他异构平台存在不一致或者较大差异性，则认为该程序属于异常程序。本文的主要工作总结如下：

(1)以真实环境下异常程序(恶意PDF文档)检测为研究案例，实例化分析说明异构化平台对于异常程序检测的有效性，在两台异构主机上同时执行待检测PDF文档，根据其执行过程中产生的主机间系统行为差异判断其是否异常。

(2)提出了一种面向异构化平台的程序异常检测架构，包含基于3个异构平台的数据收集模块和异常检测模块。数据收集模块实现各种程序在异构平台执行时系统状态信息的获取。异常检测模块实现对收集的数据处理、分析和检测。

(3)建立了基于异构平台差异性的异常检测模型，设计了基于系统状态多维特征差异性的异常检测算法，采用标签编码和Min-Max归一化对系统状态数据预处理，使用熵权法计算特征权重并引入异构平台系统状态差异性计算中，应用阈值规则有效判别异常。

(4)通过实验研究和分析对提出的方法进行评估，综合多种指标评价其性能，并与典型无监督K-means聚类算法进行比较，验证本文所提出方法的有效性。

2　研究动机

目前基于异构性检测异常应用程序进展如下：Österlund等人[12]针对Linux内核漏洞导致信息泄露问题，提出内核多变量执行 (kernel Multi-Variant eXecution, kMVX)技术：在一台机器上同时运行多个不同的内核，这些内核在正常情况下表现出相同行为，但若攻击者试图利用此漏洞，内核间可能会呈现不同行为，因此可检测这一内核信息泄露漏洞。Kirat等人[13]提出了一种基于裸机分析的逃避型恶意软件检测方法，其核心思想是在不考虑资源开销的情况下，通过在不同的系统环境：包括裸机、与基于仿真和不同类型虚拟化恶意软件分析平台中执行恶意软件，获取磁盘级和网络级行为等，根据行为偏差来检测某些对分析环境有一定识别能力的逃避型恶意软件。

本文以Xu等人[14]提出的恶意PDF程序检测为例，证明平台异构性可有效应用于异常程序检测中，其检测架构如图1所示，同时收集PDF程序在两个不同主机：Windows主机和Mac主机上执行时的系统行为，如文件系统操作、网络活动、启动进程、系统调用等，通过比较执行时系统行为差异来判断异常。

系统行为的对比，对于捕获PDF程序执行时触发漏洞利用而产生的行为差异至关重要。如果一个正常PDF在某个主机上执行时触发某些操作(如引发用于表单提交的连接到远程主机)，则在另一主机上执行时会显示相同的操作，两个主机具有相同的系统行为特征。但如果远程连接等操作是恶意PDF执行时触发的，则可能在另一主机上由于系统

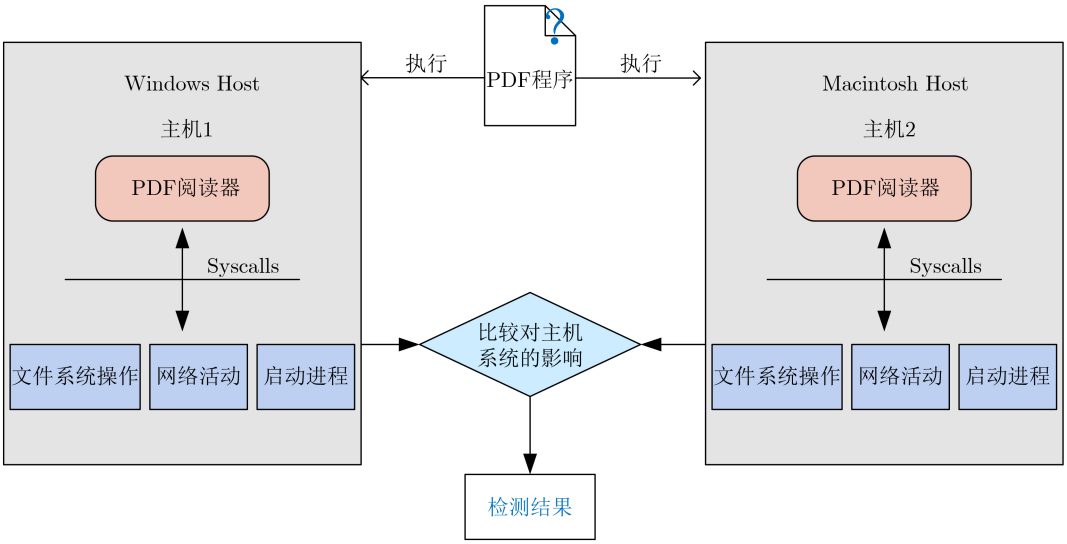

图 1 恶意文档检测架构

调用语义等的差异而触发失败，此时两主机具有不同系统行为特征。

针对这种恶意PDF检测方法的结果分析验证如表1所示，检测结果按照通用漏洞披露 (Common Vulnerabilities & Exposures, CVE)编号分组，两次崩溃表示执行恶意PDF时两个主机均崩溃。差异表示在两台主机上至少观察到一种系统行为差异。

表 1 检测结果

CVE	样本数量	结果	
		两次崩溃	差异
2016-6946	51	8	40
2016-4204	78	7	37
2016-4119	1	0	1
2016-1091	63	6	31
2016-1077	1	0	1
2016-1046	4	0	4
2015-5097	4	0	4
2015-2426	14	6	8
2015-0090	1	0	1
总数	217	27	127

根据表中检测结果不难发现，大多数情况下，恶意PDF程序只能作用于特定平台，可以依据不同主机的系统层行为差异进行判别。

综上分析，基于平台异构性可以实现一种系统层面的程序异常检测。

3 系统架构

基于研究动机案例分析，本文面向异构化平台实施异常程序检测，系统架构如图2所示，主要由以下模块组成：基于3个异构平台的状态数据收集模块和异常检测模块。

异构平台功能等价，但在指令集、内核、操作系统及组件库等层面的实现存在差异以构建异构性，例如指令集可为X86,X86-64,ARM32,ARM64等，操作系统可为Ubuntu,Debian,RedHat,CentOS等。输入代理将程序输入持续分发至各平台执行，状态监测代理分布在各异构平台上，同时收集和统计程序执行过程中产生的系统状态信息，这些状态信息涵盖CPU使用、磁盘读写、网络使用、其他系统信息、当前占用内存资源最高的进程、当前占用i/o资源最高的进程、当前占用CPU资源最高的进程、系统调用读写信息8个数据特征大类，如表2所示。根据攻击规律和异常程序执行时平台行为表现选择这些特征[15]。例如恶意程序等多表现为CPU使用、磁盘读写以及网络使用中的1个或多个显著变化，有些攻击会新产生一些非法进程，这些特征可为程序异常检测提供良好的数据支撑。对这些数据特征大类进行约简和选择，留下其中9维能够较大程度表现内部状态变化的小特征，包括用户空间程序CPU利用率usr、系统空间程序CPU利用率sys、磁盘读带宽read、磁盘写带宽write、网络收包带宽recv、网络发包带宽send、系统中断次数int、占用i/o资源最高的进程i/o process、占用CPU最高的进程cpu process，构成数据集，由各平台状态监测代理发送给异常检测模块进行异常检测。

异常检测模块利用第4节设计的基于系统状态多维特征差异性的异常检测方法，对从3个异构平台收集的数据集进行异常检测和定位：利用特征工程构建多维矢量，分别对其进行量化和标准化以实现对数据的预处理。然后利用熵权法计算特征权

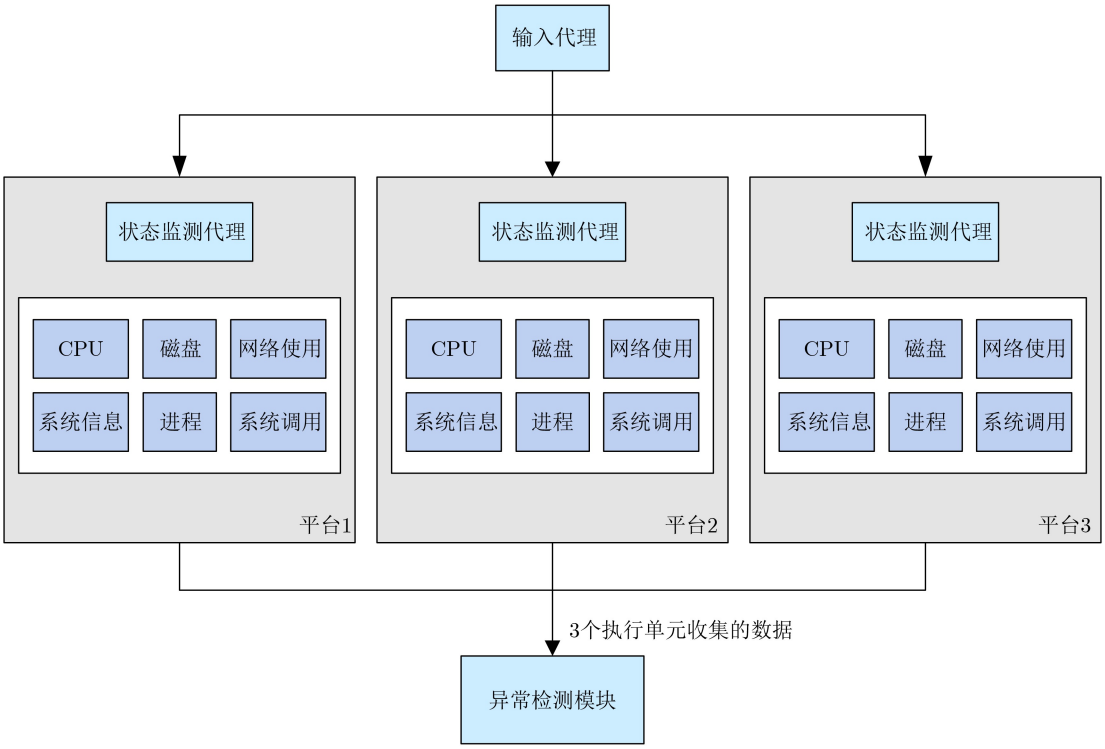

图 2 系统架构

表 2　数据集中的特征

序号	特征	特征描述
1	total cpu usage	用户空间程序CPU利用率usr 系统空间程序CPU利用率sys CPU空闲百分比idl 硬中断hiq 软中断siq
2	dsk/total	磁盘读带宽read 磁盘写带宽write
3	net/total	网络设备发送数据带宽send 网络设备接收数据带宽recv
4	system	统计中断int 上下文切换csw
5	most expensive in memory	当前占用内存资源最高的进程 memory process
6	most expensive in i/o	当前占用i/o资源最高的进程 i/o process
7	most expensive in cpu	当前占用CPU资源最高的进程 cpu process
8	syscall	系统调用读写信息syscall

重，综合数据和权重计算数据集间差异度，应用阈值规则进行异常判别和作用域判定。

相对于从单个平台上收集已知样本来生成区分正常和异常的规则与模式而言，本文所提出的检测架构，仅需比较异构平台间产生的状态差异，综合异构平台的多个特征进行比对，可以更全面地考虑到系统变化的复杂性和难预知性，有效增加检测准确性和平稳性，降低误报率。

4　基于系统状态多维特征差异性的异常检测方法

基于系统状态多维特征差异性的异常检测方法处理流程如图3所示。首先，对数据集进行预处理，将系统状态特征数据从高量级转换至低量级，有效降低高量级的特征对检测的干扰，提升检测的准确性。然后，计算异构平台系统状态特征矢量间的加权差异度。最后，利用阈值规则衡量平台间状态特征的差异程度实现异常判别。

4.1　数据预处理

由于数据集包含诸如使用I/O最大的进程，使用CPU最大的进程名这类特征，其值(如：python3, vim,traceroute等)是非数字值。这些非数字属性需要量化为数字属性，以便于在后续阶段进行处理。本文采用标签编码方法对进程特征多种数据进行数值化操作，对所有进程属性按序编码。采用标签编码的优势在于：对不连续进程特征编码，有效实现数据量化，解决了正常和异常分类过程中无法处理属性数据的问题。

特征量化后，数据集仅包含数值。由于每种属性的数据单位不一定相同，会影响差异度的计算结果，且数据量级之间的差异也会影响计算的准确性。因此，需要将数据集中每个特征值归一化到统

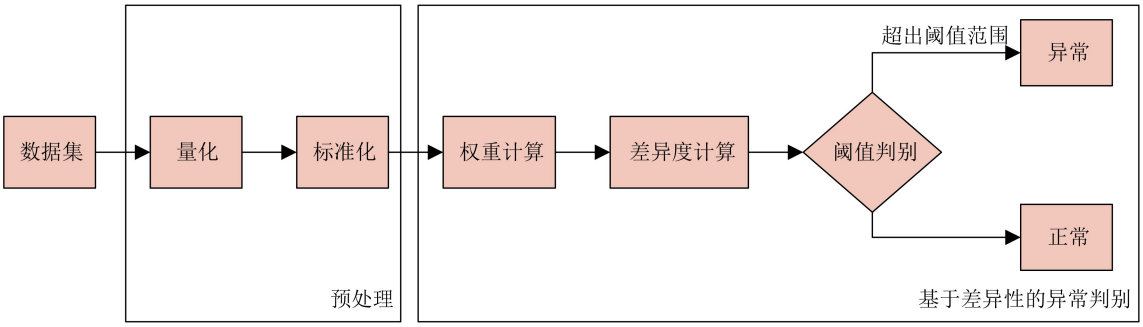

图 3 异常检测处理流程

一范围区间[0, 1]中。使用式(1)实现数据的归一化处理

$$x_n = \frac{x_{ij} - \text{Min}}{\text{Max} - \text{Min}} \tag{1}$$

其中，x_n表示经归一化后映射到[0, 1]区间的特征值；x_{ij}表示第i条系统状态的第j个特征数据值；Min表示该特征的最小值，Max表示最大值。预处理过程见表3(算法1)，行(1)～(4)对数据量化，行(5)～(6)对数据归一化。

表 3 数据预处理算法(算法1)

输入：收集的数据集
输出：预处理后的数据集
(1)取出数据集中的所有数据
(2)对特征进行判断
(3)如果特征是i/o process或cpu process，则取出其所有特征值
(4)创建标签编码器，对所有特征进行编码
(5)对编码后的数据集计算每个特征的最小值$\min(x_j)$ 和最大值$\max(x_j)$
(6)对所有数据，利用式(1)计算其标准化值

4.2 基于差异性的异常检测方法

异常检测模型的目的是发现异常程序。当异常程序执行时，异构平台之间状态特征存在差异。差异度是用来衡量数据样本间差异程度的尺度[16,17]，可通过计算收集到的各平台状态特征样本之间的距离进行度量。

定义1 $N\times M$的空间的样本X和样本Y的欧氏距离定义为

$$D = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij} - y_{ij})^2} \tag{2}$$

差异度

$$\text{dif}(X,Y) = 1 - \frac{1}{D} = 1 - \frac{1}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij} - y_{ij})^2}} \tag{3}$$

其中，X,Y为两个$N\times M$的样本，x_{ij}为样本X的第i个实例的第j个属性，y_{ij}为样本Y的第i个实例的第j个属性，$i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, m$。

利用系统状态多维特征差异度来检测平台是否有异常程序执行，其本质表现在各种系统状态特征属性的变化上。异常程序执行时，不同特征属性的变化程度不同。如果为其分配相同权重，则会导致异常判断不准确。因此在差异度的计算中引入各属性的权重来提高检测灵敏度。

本文使用熵权法根据特征的变异程度来确定特征权重，对在异常检测中表征性强的特征赋予较高权重。计算权重的过程如下：

(1)取出3个平台归一化后的所有数据。

(2)计算第j个特征下第i个样本值占该特征的比重

$$p_{ij} = \frac{x_{ij}}{\sum_{i=1}^{n} x_{ij}}, i = 1, 2, \cdots, n, j = 1, 2, \cdots, m \tag{4}$$

(3)计算第j项特征的熵值

$$e_j = -\frac{1}{\ln(n)}\sum_{i=1}^{n} p_{ij}\ln(p_{ij}) \tag{5}$$

(4)利用熵值计算各项指标的权值

$$w_j = \frac{1 - e_j}{\sum_{j=1}^{m}(1 - e_j)} \tag{6}$$

加权后的两两平台间数据样本X,Y的距离公式为

$$D(X,Y) = \sqrt{\sum_{j=1}^{9}\sum_{i=1}^{n} w_j(x_{ij} - y_{ij})^2} \tag{7}$$

差异度公式为

$$\begin{aligned}\text{dif}(X,Y) &= 1 - \frac{1}{D(X,Y)} \\ &= 1 - \frac{1}{\sqrt{\sum_{j=1}^{9}\sum_{i=1}^{n} w_j(x_{ij} - y_{ij})^2}}\end{aligned} \tag{8}$$

其中，w_j为第j类特征的权值。

应用阈值规则进行异常判别。如果某个平台和其他两个平台的差异度均高于这两个平台间的差异度且差值超出预设的阈值范围，则判断这一平台发生异常；如果差异度差值处于预设的阈值范围内，则认为当前状态正常。例如：平台1和平台2之间的差异度为d1，平台1和平台3之间的差异度为d2，平台2和平台3之间的差异度为d3，如果d1>d3, d2>d3且d1–d3>ε, d2–d3>ε，则判断平台1异常。

基于系统状态多维特征差异性的异常检测方法如表4所示，行(1)～(2)用于特征权重计算，行(3)～(7)用于差异度及其差值计算，行(8)～(11)对异常判别和定位。

表 4　基于系统状态多维特征差异性的异常检测方法(算法2)

输入：平台1,2,3归一化后的数据集E1,E2,E3，差异度间差值的阈值ε
输出：异常判断结果
(1) 将预处理后的数据集合并。
(2) 根据式(4)—式(6)计算各个属性特征的权重。
(3) 根据式(8)计算两两平台间的状态特征差异度。
(4) 进行差异度的比较，并计算差异度的差值。令
(5) diff1=dif(E1,E2)-dif(E2,E3)
(6) diff2=dif(E1,E3)-dif(E2,E3)
(7) diff3=dif(E1,E2)-dif(E1,E3)
(8) 应用阈值规则，判断平台是否发生异常。
(9) 如果diff1>0,diff2>0且|diff1|>ε, |diff2|>ε，则判断平台1异常。
(10) 如果diff2<0,diff3>0且|diff2|>ε, |diff3|>ε，则判断平台2异常。
(11) 如果diff1<0,diff3<0且|diff1|>ε, |diff3|>ε，则判断平台3异常。

5　实验分析

为了对所提程序异常检测方法进行性能测试与验证，搭建了实验环境并进行系统状态数据收集。实验环境中建立了3个具有不同版本操作系统、内核和CPU类型的平台，平台1使用Redhat4.8.5操作系统和ARM64 处理器，平台2使用Ubuntu7.2.0操作系统和X86处理器，平台3使用Ubuntu18.04操作系统和X86-64处理器，同时这些平台中安装有常用软件，具备正常的操作系统功能。实验过程中，在3个平台上同时执行各种待检测程序，同时在3个平台上利用状态收集工具获取当前系统时间和系统状态特征，将收集的信息按照适当的格式存为数据文件以便后续处理。按照此数据采集方法，在3个平台中同时且连续采集28800次系统数据。将dirty-cow提权攻击程序分时段同时在3个平台执行，将攻击时段内平台的系统状态定义为异常状态，其他时段内系统状态定义为正常状态来实现测试数据集的构建。

5.1　性能评估

本节实验利用本文所提方法对收集的数据集进行异常检测来验证其检测效果，并将精确率 (Precision)、召回率(Recall, TPR)、误报率(False Positive Rate, FPR)、漏报率(Missing Alarm Rate, MAR) 4种指标作为验证本文所提面向异构化平台异常检测方法性能的指标，其计算公式如式(9)—式(12)所示。

$$\text{Precision}=\frac{\text{TP}}{\text{TP}+\text{FP}} \tag{9}$$

$$\text{TPR}=\frac{\text{TP}}{\text{TP}+\text{FN}} \tag{10}$$

$$\text{FPR}=\frac{\text{FP}}{\text{TN}+\text{FP}} \tag{11}$$

$$\text{MAR}=\frac{\text{FN}}{\text{FN}+\text{TP}} \tag{12}$$

式(9)—式(12)中各变量的含义解释如下：

真正例(True Positive, TP)：预测异常且实际异常的样本数；真负例(True Negative, TN)：预测正常且实际正常的样本数；假正例(False Positive, FP)：预测异常但实际正常的样本数；假负例(False Negative, FN)：预测正常但实际异常的样本数。其中，正例表示异常数据，负例表示正常数据。

本文所提方法精确性和误报率等方面的性能如图4和图5所示，图4为5次实验情况下的检测精确率和召回率，检测精确率和召回率均大于95%，检测平均精确率为98.79%，平均召回率为98.84%。图5为5次实验的误报率和漏报率，5次实验情况的误报率都不超过10%，其平均值为5.75%，漏报率均不超过5%，其平均值为1.16%。在个别情况下，异常刚发生时，系统状态特征变化不太明显，此时平台间的差异度差值没有超出阈值，异常被判定为正常，因此召回率会有所下降，产生一定的漏报率，由式(10)和式(12)可知，召回率和漏报率成正比，召回率降低导致漏报率上升。由于平台正常状态下，状态间的细微差异的积累可能影响平台间的差异度计算，导致正常状态被判定为异常程序运行状态，此时精确度会有轻微下降，误报率会有轻微上升。图4和图5中实验结果综合表明，本文的方法可以以较高的精确率和召回率及较低误报率和漏报率检测到异常。

5.2　与聚类方法进行异常检测的对比

针对收集的数据集，使用基于距离的K-means

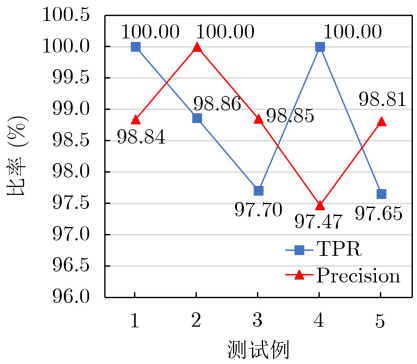

图 4 检测精确率和召回率

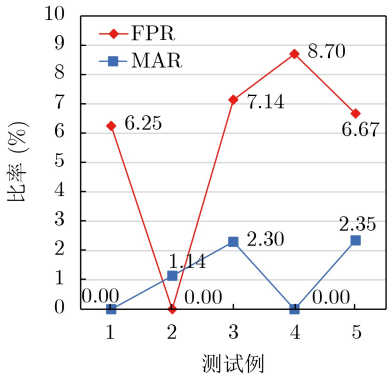

图 5 检测的误报率和漏报率

聚类方法与本文所提出的方法进行检测性能对比实验。将检测准确率和误报率作为本实验的评价标准[18,19]。K-means算法的思想是以k为参数，把n个样本分成k个簇，使每个簇内具有较小的距离，而不同簇间的距离较大。在本实验中使用K-means算法聚类检测，对在3个异构平台收集到的所有测试数据集进行检测，检测的平均准确率为89.8%，而使用本文提出的检测方法对同样的样本进行检测，检测的平均准确率为97.8%。

对数据集进行10次随机抽样检测，结果如图6所示。使用3种方法对数据集进行检测，方法1是使用K-means算法检测单个平台的测试数据集，方法2是使用K-means对从整个异构平台架构中收集的所有数据集进行抽样检测，方法3是使用本文所提出的面向平台异构性的轻量级程序异常检测方法对整个异构平台中收集的所有数据集进行抽样检测。

当使用方法1时，如果异常样本较多，其检测准确率较高，可达99%，但当抽样样本中异常样本较少时，由于K-means算法的效果严重依赖初始聚类中心，因而当初始聚类中心选择不当时，检测的准确性迅速下降到75%左右，并产生22.4%的误报率，误报率较高；当使用方法2时，由于不同平台间也有一定的差异，K-means在检测过程中对噪声敏感产生一定的误报率，检测的平均准确率为

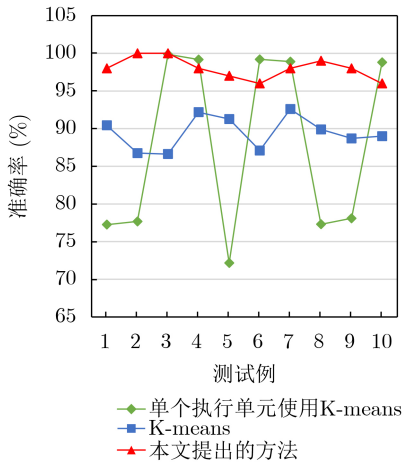

图 6 检测准确率对比

86.46%；使用方法3时，由于利用加权差异度算法横向对比平台间差异，有针对性的计算会对状态差异识别更为明显，因而达到提升准确率的目的，准确率稳定在95%以上，平均检测准确率较高，其值为97.8%，与方法2相比，检测准确率提升了13.12%；与方法1相比，大大减少了检测误报率。

本文还比较了所提算法和K-means算法在不同抽样数下的检测时间，实验结果如表5所示。可以得出，两种算法的检测时间都会随着样本数量增加而增加，本文所提算法与K-means算法相比，由于K-means在每轮迭代时都需要计算所有样本点到质心的距离，而本文所提算法仅需计算平台间样本的距离，因此大大缩短了异常检测时间，具有较好的实时性。

表 5 两种算法在不同抽样数量下的异常检测时间(s)

	抽样数				
	1105	2003	3046	4127	5478
本文算法	0.64	0.72	0.87	1.02	1.35
K-means	1.56	3.32	5.92	8.47	11.70

对上述比对情况进行总结，就检测准确率、误报率和检测时间而言，本文所提算法性能优于K-means算法的性能。

6 结束语

本文针对程序异常检测中准确率低、误报率高和检测时间长的问题，提出一种面向异构化平台的轻量级程序异常检测方法。该方法通过构建异构平台，获取异构平台对相同输入程序执行时产生的系统状态，跨平台横向对比系统状态的差异，进行程序异常检测。实验结果表明，所提方法有效提升了程序异常检测的准确率、降低了误报率且耗时较

短，是一种有效的程序异常检测方法。在后续工作中，将会对算法等进行改进，并提高算法在不同实际攻击场景下的检测准确性。

参 考 文 献

[1] 张祖法. 网络流量中面向缓冲区溢出漏洞的恶意程序检测方法研究[D]. [硕士论文], 江苏大学, 2020.
ZHANG Zufa. Research on malware detection method for buffer overflow vulnerability in network traffic[D]. [Master dissertation], Jiangsu University, 2020.

[2] 张雄冠, 邵培南. 基于textCNN模型的Android恶意程序检测[J]. 计算机系统应用, 2021, 30(1): 114–121. doi: 10.15888/j.cnki.csa.007722.
ZHANG Xiongguan and SHAO Peinan. Android malware detection based on textCNN model[J]. *Computer Systems & Applications*, 2021, 30(1): 114–121. doi: 10.15888/j.cnki.csa.007722.

[3] 吴震雄. Android恶意软件静态检测方案研究[D]. [硕士论文], 南京邮电大学, 2015.
WU Zhenxiong. Research on android malware static detection system[D]. [Master dissertation], Nanjing University of Posts and Telecommunications, 2015.

[4] MA Zhuo, GE Haoran, LIU Yang, *et al*. A combination method for android malware detection based on control flow graphs and machine learning algorithms[J]. *IEEE Access*, 2019, 7: 21235–21245. doi: 10.1109/ACCESS.2019.2896003.

[5] DINABURG A, ROYAL P, SHARIF M, *et al*. Ether: Malware analysis via hardware virtualization extensions[C]. The 15th ACM Conference on Computer and Communications Security, Alexandria, USA, 2008: 51–62. doi: 10.1145/1455770.1455779.

[6] 张若楠, 李红辉, 张骏温. 一种融合改进Kmeans和KNN的网络入侵检测方法[J]. 计算机科学, 2018, 10A(45): 172–176.
ZHANG Ruonan, LI Honghui, and ZHANG Junwen. Hybrid improved Kmeans with improved KNN for network intrusion detection algorithm[J]. *Compouter Science*, 2018, 10A(45): 172–176.

[7] 汪洁, 王长青. 子图相似性的恶意程序检测方法[J]. 软件学报, 2020, 31(11): 3436–3447. doi: 10.13328/j.cnki.jos.005863.
WANG Jie and WANG Changqing. Malware detection method based on subgraph similarity[J]. *Journal of Software*, 2020, 31(11): 3436–3447. doi: 10.13328/j.cnki.jos.005863.

[8] 陈志峰, 李清宝, 张平, 等. 基于聚类分析的内核恶意软件特征选择[J]. 电子与信息学报, 2015, 37(12): 2821–2829. doi: 10.11999/JEIT150387.
CHEN Zhifeng, LI Qingbao, ZHANG Ping, *et al*. Signature selection for kernel malware based on cluster analysis[J]. *Journal of Electronics & Information Technology*, 2015, 37(12): 2821–2829. doi: 10.11999/JEIT150387.

[9] YOO S, KIM S, KIM S, *et al*. AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification[J]. *Information Sciences*, 2021, 546: 420–435. doi: 10.1016/j.ins.2020.08.082.

[10] 邬江兴. 网络空间拟态防御原理[M]. 2版. 北京: 科学出版社, 2018: 148–149.
WU Jiangxing. The Principle of Cyber Mimic Defence[M]. 2nd ed. Beijing: Science Press, 2018: 148–149. .

[11] GARCIA M, BESSANI A, GASHI I, *et al*. Analysis of operating system diversity for intrusion tolerance[J]. *Software*: *Practice and Experience*, 2014, 44(6): 735–770. doi: 10.1002/spe.2180.

[12] ÖSTERLUND S, KONING K, OLIVIER P, *et al*. kMVX: Detecting kernel information leaks with multi-variant execution[C]. The Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, Providence, USA, 2019: 559–572. doi: 10.1145/3297858.3304054.

[13] KIRAT D, VIGNA G, and KRUEGEL C. BareCloud: Bare-metal analysis-based evasive malware detection[C]. The 23rd USENIX conference on Security Symposium, Berkeley, USA, 2014: 287–301.

[14] XU Meng and KIM T. PLATPAL: Detecting malicious documents with platform diversity[C]. The 26th USENIX Conference on Security Symposium, Vancouver, Canada, 2017: 271–287.

[15] 张剑, 童言, 徐明迪, 等. 轻量级主机数据采集与实时异常事件检测方法研究[J]. 西安交通大学学报, 2017, 51(4): 97–102. doi: 10.7652/xjtuxb201704015.
ZHANG Jian, TONG Yan, XU Mingdi, *et al*. A method for data collection and real-time anomaly detection of lightweight hosts[J]. *Journal of Xi'an Jiaotong University*, 2017, 51(4): 97–102. doi: 10.7652/xjtuxb201704015.

[16] 张浚, 张凤荔, 罗琴, 等. 基于多特征相似度的大规模网络异常检测算法[J]. 计算机工程, 2007, 33(24): 181–183. doi: 10.3969/j.issn.1000-3428.2007.24.063.
ZHANG Jun, ZHANG Fengli, LUO Qin, *et al*. Large-scale network anomaly detecting method based on multi-feature similarity[J]. *Computer Engineering*, 2007, 33(24): 181–183. doi: 10.3969/j.issn.1000-3428.2007.24.063.

[17] HU Shuai, XIAO Zhihua, RAO Qiang, *et al*. An anomaly detection model of user behavior based on similarity clustering[C]. Proceedings of 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference, Chongqing, China, 2018. doi: 10.1109/ITOEC.2018.8740748.

[18] 缪祥华, 单小撤. 基于密集连接卷积神经网络的入侵检测技术研究[J]. 电子与信息学报, 2020, 42(11): 2706–2712. doi: 10.

11999/JEIT190655.

MIAO Xianghua and SHAN Xiaoche. Research on intrusion detection technology based on densely connected convolutional neural networks[J]. *Journal of Electronics & Information Technology*, 2020, 42(11): 2706–2712. doi: 10.11999/JEIT190655.

[19] 董书琴, 张斌. 基于深度特征学习的网络流量异常检测方法[J]. 电子与信息学报, 2020, 42(3): 695–703. doi: 10.11999/JEIT190266.

DONG Shuqin and ZHANG Bin. Network traffic anomaly detection method based on deep features learning[J]. *Journal of Electronics & Information Technology*, 2020, 42(3): 695–703. doi: 10.11999/JEIT190266.

马海龙：男，1980年生，副研究员，研究方向为网络安全、路由工程.

尹梓诺：女，1997年生，硕士生，研究方向为网络空间安全.

胡 涛：男，1993年生，博士生，研究方向为新型网络体系结构.

责任编辑：余 蓉