文章编号:1007-1423(2018)08-0114-03 DOI:10.3969/j.issn.1007-1423.2018.08.027

# 一种基于控制流图特征的 Linux 平台恶意代码检测方法

刘留,尹颜朋,王玉伟,丁祝祥

(四川大学计算机学院,成都 610065)

**摘要:**

随着 Linux 操作系统的普及和恶意代码的增长,Linux 系统的安全性也在下降。为了有效地检测 Linux 平台上的恶意软件,提出一种基于控制流图特征的恶意软件检测模型。以对 ELF 样本进行静态分析得到的汇编代码为基础构建控制流图,并提取一些属性作为特征向量,使用机器学习算法进行训练分类器。以恶意软件样本 202 个、良性软件 305 个为数据集,采用 WEKA 集成的 4 种算法 J48、RandomForest、IBK 和 NaiveBays 训练分类器,采用 10 折交叉验证评估分类器的性能。实验证明 4 个分类器能够有效地检测出恶意代码,并且准确率高,误报率低,其中 RandomForest 效果最好。

**关键词:**

恶意代码检测; 机器学习; Linux

## 0 引言

Linux 系统的开源特点使得其能快速发展,许多的不同的发行版出现,越来越多的人也开始使用 Linux 系统满足工作需要。随着恶意代码的快速增长,Linux 平台的安全性也在逐渐降低。

目前,有许多的恶意代码检测方法,这些方法可以分为静态分析和动态分析两类。静态分析通过分析指令序列的二进制代码来判断可执行文件是否是恶意的[1]。最基本的静态分析方法是基于签名的检测方法,也是应用最广泛的方法。签名是一段能够标识特定恶意软件的字节序列。通过分析大量的恶意软件,我们能够创建一个签名库。依靠模式匹配,如果一个未知文件包含了签名库的一个签名,则可将该文件标识为恶意软件,这种方法对大多数恶意软件有很好的识别效果。一般来说,基于签名的方法可以检测已知的恶意软件,但不能检测未知的恶意软件或变种,因为没有可用的签名[2]。同时也存在很多的代码混淆手段可以隐藏可执行的恶意代码,包括垃圾代码插入、寄存器重命名、子程序重排、等价指令替换等[3]。基于混淆技术,黑客可以方便地开发一个新的恶意软件变种来绕过检测[4]。基于行为的恶意代码检测方法是普遍应用的动态分析方法。行为分析关注软件执行时的信息获取。这些信息必须精确可靠,这意味着必须在一个仿真环境中运行程序[5]。然而恶意软件可能仅在和用户交互时才表现出恶意行为,或者在触发特定事件时表现出恶意行为,如到达特定时间。通过分析、收集恶意软件的行为模式,可以建立一个数据库。在检测阶段,将未知软件的行为模式与数据库中的模式作比较即可以检测恶意软件。

近年来,机器学习和数据挖掘算法逐渐应用于恶意代码检测领域,取得了不错的效果。为了解决以上问题,本文提出了一种基于软件控制流图特征的恶意代码检测方法。通过对软件进行静态分析得到汇编代码,进而分析汇编代码构建控制流图。提取控制流图的属性作为特征向量,将这些特征向量送入机器学习算法进行训练,得出分类器,使用分类器来检测恶意软件。

## 1 相关定义

基本块:一段有序指令序列,从第一条指令顺序执行到序列的最后一条指令。除了最后一条指令外,序列中的其他指令不会是分支或者跳转指令[6]。

控制流图：定义 V = {VI, V2, ... , Vn} 为代表基本块的顶点集，E = {brijl brij 表示 Vi 到 Vj 的控制流}为指示基本块之间的控制流的边集，P = {V,E}即可表示一个程序的控制流图[7]。

ELF(Executable and Linkable Format)：是一种用于二进制文件、可执行文件、目标代码、共享库和核心转储格式文件，是 Linux 的主要可执行文件格式。

## 2 控制流图构建

控制流图构建基于汇编代码，因此要先对软件进行反汇编。本文采用了 Linux 系统提供的反汇编工具“objdump”对软件进行反汇编，“objdump -D”命令会反汇编 ELF 文件中所有的段。在得到的汇编代码中，“text”段包含了程序运行所需的主要指令，“plt”段包含了程序所调用的库函数信息，之后通过分析这两个段的内容来构建控制流图。遍历“text”段中的汇编指令，将其划分为不同的基本块，即控制流图的顶点；分析控制流指令来建立基本块之间的联系，即控制流图的边。边无权重，且为有向边，边的方向表明了控制流转移方向。

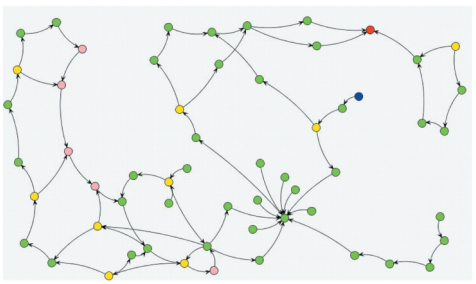

图 1 构建的控制流图

## 3 特征选择

对控制流图中的各类信息进行指标化，是特征选择的一种常用方法。我们通过统计方法建立了 22 个指标，每项指标都可以作为软件的一个特征，由此可以产生基于控制流图的软件特征集。表 1 描述了图特征的定义和说明。

表 1 基于图指标的软件特征描述

| 特征名 | 说明 |
|---|---|
| vertexNum | 顶点总数 |
| edgeNum | 边的总数 |
| pltVexNum | 导入表函数总数 |
| maxOutD | 图的最大出度 |
| namedVexNum | 常规函数总数 |
| zeroInDNum | 入度为零的顶点数 |
| zeroOutDNum | 出度为零的顶点数 |
| maxDegree | 图的最大度数 |
| maxInD | 最大入度 |
| MCSVexNum | 最大连通子图所含顶点数 |
| pltEdgNum | 指向导入表顶点的边的数目 |
| zeroIODNum | 出入度都为零的顶点数 |
| namedEdgNum | 指向常规函数顶点的边的数目 |
| pltVexRatio | 导入表顶点占总顶点的比例 |
| csNum | 连通子图的数目 |
| namedVexRatio | 常规函数顶点占比 |
| zeroInDRatio | 入度为零的顶点占比 |
| zeroOutDRatio | 出度为零的顶点占比 |
| MCSVexRatio | 最大连通子图所含顶点占比 |
| zeroIODRatio | 出入度均为零的顶点占比 |
| pltEdgRatio | 指向导入表顶点的边占总边数的比例 |
| namedEdgRatio | 指向常规函数顶点的边的占比 |

## 4 实验和分析

实验数据集包含了 202 个恶意软件样本和 305 个良性样本。良性软件样本取自 Linux 系统“/bin”、“/sbin”、“/usr/bin”等目录下的 ELF 文件，恶意软件样本通过在 Linux 系统上编译源代码得来，源代码可从网站 https://www.exploit-db.com 下载。使用 WEKA 平台来训练分类器，主要使用了四种机器学习算法：J48、Randomforest、NaiveBays、IBK，采用 10-fold 交叉验证检验分类器的效果。

表 2 四种分类器的分类效果

| Classifier | TPR(%) | FPR(%) | Accuracy(%) |
|---|---|---|---|
| J48 | 99.0 | 1.3 | 98.8 |
| RandomForest | 100 | 0.7 | 99.6 |
| IBK | 99.0 | 0.7 | 99.2 |
| NaiveBays | 97.0 | 0.3 | 98.6 |

四种分类器都取得了高于 98%的准确率，且 FPR 均低于 2%，说明这四种分类器均可有效检测 Linux 平台的恶意软件。其中，RandomForest 算法的 TPR 达到了 100%，FPR 为 0.7%，且准确率达到 99.6%，说明该分类器的检测效果最好。

参考文献：

[1] Natani P, Vidyarthi D. An Overview of Detection Techniques for Metamorphic Malware[M]. Intelligent Computing, Networking, and

Informatics. Springer India, 2014: 637-643.
[2]Yin H, Song D, Egele M, et al. Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis[C]. Proceedings of the 14th ACM conference on Computer and Communications Security. ACM, 2007: 116-127.
[3]Pluskal O. Behavioural Malware Detection Using Efficient SVM Implementation[C]. Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems. ACM, 2015: 296-301.
[4]Yan L K, Yin H. Droidscope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis[C]. Presented as part of the 21st USENIX Security Symposium (USENIX Security 12), 2012: 569-584.
[5]Shabtai A, Kanonov U, Elovici Y, et al. "Andromaly": a Behavioral Malware Detection framework for Android Devices[J]. Journal of Intelligent Information Systems, 2012, 38(1): 161-190.
[6]Oh N, Shirvani P P, McCluskey E J. Control-Flow Checking by Software Signatures[J]. IEEE transactions on Reliability, 2002, 51(1): 111-122.
[7]Li J, Tan Q, Xu J. Reconstructing Control Flow Graph for Control Flow Checking[C]. Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on. IEEE, 2010, 1: 527-531.

**作者简介：**
刘留(1995-),男,安徽阜阳人,硕士,研究方向为恶意代码检测
收稿日期:2018-01-23　　修稿日期:2018-02-28

# A Malware Detection Method Based on Control Flow Graph Features for Linux

LIU Liu, YIN Yan-peng, WANG Yu-wei, DING Zhu-xiang

(College of Computer Science, Sichuan University, Chengdu 610065)

**Abstract:**
As Linux operation system becomes popular, it is now considered as a platform with decreasing security since the malware increases rapidly. In order to detect the malware on Linux platform effectively, proposes a malware detection model based on Control Flow Graph features. By taking attributes of Control Flow Graph, which is constructed from assembly code of ELF samples using static analysis, as feature vectors that are subject to training and classification upon machine learning and data mining algorithm, a classifier is established. With 202 malware and 305 benign programs as a dataset, uses four algorithms involved in WEKA, namely J48, RandomForest, IBK, and NaiveBays to train classifiers and adopt 10-fold cross validation to evaluate performance of the classifiers so that we can pick the most effective one. The experiment demonstrates that the four classifiers can effectively detect program with high accuracy and low false positive rate, and RandomForest works best on the dataset.
**Keywords:**
Malware Detection; Machine Learning; Linux