计算机科学与探索
Journal of Frontiers of Computer Science and Technology
1673-9418/2024/18(04)-1083-11
doi: 10.3778/j.issn.1673-9418.2307023

基于AECD词嵌入的挖矿恶意软件早期检测方法

曹传博[1],郭　春[1+],李显超[2],申国伟[1]
1. 贵州大学 计算机科学与技术学院 公共大数据国家重点实验室,贵阳 550025
2. 贵州翔明科技有限责任公司,贵阳 550025
+ 通信作者 E-mail: gc_gzedu@163.com

摘　要:挖矿恶意软件会损害系统安全,缩减硬件寿命,以及造成大量电力消耗,实施对挖矿恶意软件的早期检测以及时阻止其损害对于维护系统安全至关重要。现有的基于动态分析的挖矿恶意软件早期检测方法未能兼顾检测的及时性和准确率。为及时且准确地检测挖矿恶意软件,将挖矿恶意软件运行初期所调用的一定长度的API(application programming interface)名称、API操作类别和调用API的DLL(dynamic link library)进行融合以更充分地描述其在运行初期的行为信息,提出AECD(API embedding based on category and DLL)词嵌入方法并进一步提出基于AECD词嵌入的挖矿恶意软件早期检测方法(CEDMA)。CEDMA以软件在运行初期所调用的一定长度的API序列为检测对象,使用AECD词嵌入和TextCNN(text convolutional neural network)建立检测模型来实施对挖矿恶意软件的早期检测。实验结果显示,CEDMA以软件运行后首次调用的长度为3 000的API序列作为输入时,可分别以98.21%、96.76%的Accuracy值检测实验中已知和未知的挖矿恶意软件样本。

关键词:挖矿恶意软件;动态分析;早期检测;深度学习
文献标志码:A　**中图分类号:**TP309

Cryptomining Malware Early Detection Method Based on AECD Embedding

CAO Chuanbo[1], GUO Chun[1+], LI Xianchao[2], SHEN Guowei[1]
1. State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China
2. Guizhou Xiangming Technology Co., Ltd., Guiyang 550025, China

Abstract: Cryptomining malware can compromise system security, reduce hardware lifetime, and cause significant power consumption. Therefore, implementing cryptomining malware early detection to stop its damage in time is critical to system security. The existing dynamic analysis-based cryptomining malware early detection methods are hard to balance the timeliness and accuracy of detection. To detect cryptomining malware accurately and timely, this paper integrates a certain length of API (application programming interface) names, API operation categories and DLLs (dynamic link libraries) called by cryptomining malware in the early stage of operation to more fully describe its behavioral information in this stage, and proposes the AECD (API embedding based on category and DLL) embedding and further proposes a cryptomining malware early detection method based on AECD embedding (CEDMA). CEDMA uses the API sequence called by software in the early stage of operation as the object of detection and uses

基金项目:国家自然科学基金(62162009);贵州省科技支撑计划(黔科合支撑[2022]一般071);贵州省高等学校大数据与网络安全创新团队(黔教技[2023]052);贵州省科技计划项目(黔科合平台人才GHB[2023]001)。
This work was supported by the National Natural Science Foundation of China (62162009), the Science and Technology Support Program of Guizhou Province ([2022]071), the Big Data and Network Security Innovation Team of Universities in Guizhou Province ([2023]052), and the Science and Technology Program of Guizhou Province (GHB[2023]001).
收稿日期:2023-07-10　**修回日期:**2023-09-18

AECD embedding and TextCNN (text convolutional neural network) to build a detection model to implement cryptomining malware early detection. Experimental results show that when CEDMA takes the 3000 API sequence called for the first time after the software runs as input, it can detect the known and unknown cryptomining malware samples in the experiment with 98.21% and 96.76% accuracy values, respectively.

Key words: cryptomining malware; dynamic analysis; early detection; deep learning

由于加密货币匿名性的特点及其逐渐攀升的交易价格[1],近年来网络攻击者越来越青睐能够实现非法挖矿的挖矿恶意软件以获取经济利益。挖矿恶意软件被植入目标主机后会极大地消耗电力资源并影响受感染主机上其他进程的正常运行。网络安全厂商sonicwall最近的报告显示,即使在加密货币市场趋于崩溃的情况下,2022年的挖矿恶意软件攻击仍增长了30%[2]。因此,挖矿恶意软件检测及防御是目前网络安全领域的一个重要课题。

为了逃避检测以获得持续的收益,挖矿恶意软件开发者注重挖矿恶意软件的隐蔽性和运行的持久性,由此集成了诸多逃避检测的措施,如监控性能监视工具、将自身伪装为系统文件、限制进程CPU(central processing unit)利用率阈值等[3-5]提升其隐蔽性的手段,以及通过修改计划任务和启动项、创建守护进程等增强其运行持久性的方法。知名挖矿恶意软件H2Miner就采用了创建计划任务、将自身伪装为系统文件等措施来逃避检测。为准确检测挖矿恶意软件,基于机器学习的动态检测方法和流量检测方法成为近年来研究的主流。但目前这两类方法或需要较长的数据收集时间,或需要依赖人工的特征工程,导致检测及时性不佳。为实施挖矿恶意软件的早期检测以及时阻止其对系统的损害,目前的检测方法通常通过缩短软件行为数据收集时长来达到提升检测及时性的目的,但这会导致可用于方法分析的行为减少而不易达到高准确率地检测挖矿恶意软件的目的[6]。因此,如何在提升检测及时性的同时维持高的检测精度是目前挖矿恶意软件检测领域的一个研究难点。

为此,本文从提升挖矿恶意软件早期检测准确率的意图出发,探索利用挖矿恶意软件运行初期调用的API(application programming interface)的操作类别和调用API的DLL(dynamic link library)来提升检测精度的可行性。本文提出了一种词嵌入方法AECD(API embedding based on category and DLL)来丰富软件的API词向量所表达的信息。此外,为了避免人工的特征工程,本文选用了能保留API上下文关系且结构简单、训练速度快的TextCNN(text convolutional neural network)算法[7]来建立检测模型。结合AECD词嵌入和TextCNN算法,本文提出了基于AECD词嵌入的挖矿恶意软件早期检测方法CEDMA(cryptomining malware early detection method based on AECD embedding)。与现有挖矿恶意软件检测方法相比,CEDMA能够兼顾检测的及时性和准确率。本文的主要贡献如下:

(1)从挖矿恶意软件调用API的DLL和API操作类别两方面分析挖矿恶意软件和良性软件的行为,以此为基础提出一种可丰富软件的API词向量所蕴含信息的API词嵌入方法AECD,为构建基于AECD词嵌入的挖矿恶意软件早期检测方法奠定了基础。

(2)提出一种挖矿恶意软件早期检测方法CEDMA。CEDMA以软件在运行初期所调用的一定长度的API序列为检测对象来获取高的检测及时性,使用AECD词嵌入缓解挖矿恶意软件早期行为信息量不足的问题,并使用具有结构简单特性的TextCNN建立高效的检测模型。

(3)利用涉及多种加密货币的多款挖矿恶意软件以及多种类型的良性软件对CEDMA进行实验测试。实验结果表明CEDMA能够使用软件在建立网络连接前的API序列、DLL序列和API操作类别序列实现对挖矿恶意软件样本的高准确率早期检测。

1 相关工作

近年来国内外研究人员提出了多种挖矿恶意软件检测方法。本文研究属于动态检测方法范畴,现有挖矿恶意软件动态检测方法可划分为基于主机的检测方法、基于流量的检测方法以及早期检测方法三类。

1.1 基于主机的检测方法

基于主机的检测方法关注挖矿恶意软件在运行时对系统硬件性能造成的影响、所调用的API序列等在主机上的动态行为。基于主机的动态检测方法中基于性能影响的检测方法易受同一硬件环境中其他进程影响且移植性差。基于API序列的检测方法中,Berecz等人[8]选择与挖矿行为相关的10个API和5个动态链接库以及另外5个PE(portable executable)

文件属性作为特征，使用机器学习建立检测模型。Karn等人[9]使用 n-gram 模型从8种挖矿恶意软件和8个良性应用的持续1 min的API序列中提取特征，并使用决策树等算法构建检测模型。动态检测方法从样本的动态行为中提取能反映行为特性的特征集合，该类方法在对抗混淆、加壳等逃逸技术上具有一定优势，但缺点是需要较大的时间和性能开销，检测及时性差。

1.2 基于流量的检测方法

基于流量的检测方法可以监控网络入口的流量，易于部署且能覆盖监控网络内的所有设备。i Muñoz等人[10]通过分析正常流量与来自5种挖矿恶意软件的流量在通信行为上的不同，总结出5个通信行为特征，并使用机器学习算法构建检测模型。Caprolu等人[11]将3种开源挖矿客户端的流量与3种良性软件的流量在数据包的大小、间隔时间等方面进行比较以总结出特征，然后结合机器学习建立检测模型。基于流量的检测方法通常需要等待挖矿恶意软件建立网络连接后才能实施检测，与基于主机的检测方法相比有更大的检测延迟，且难以对抗应用了延迟数据包发送、增加无用数据包、添加代理、数据包填充等逃逸技术的挖矿恶意软件。

1.3 早期检测方法

挖矿恶意软件在运行时具有执行横向移动、组建僵尸网络、窃取受害者的数据等的可能性，其挖矿进程亦会极大地消耗系统的硬件资源并影响其他进程的正常运行，因此实施对挖矿恶意软件的早期检测对于及时阻止其对系统安全的损害具有重要意义。曹传博等人[12]提出挖矿软件行为多样期的概念，所提检测方法以样本建立网络连接前的行为作为检测对象来实施挖矿恶意软件早期检测，在收集4 s的API序列时可在其实验集上获得96.55%的F1-score值。Sun等人[13]提出了一种挖矿软件早期流量的卷积特征提取方法，该方法以8种加密货币的挖矿程序为样本，借助卷积函数从挖矿流量的前数个数据包构成的包负载大小序列中提取特征，再使用机器学习建立检测模型。以上方法通过缩短数据收集时间或减少收集的数据量达到早期检测的目的，但由于所能用于分析的数据量少，在检测准确率上相对传统基于主机的检测方法和基于流量的检测方法不具有优势。

综上，目前基于主机的动态检测方法所需的数据收集时间长而导致其检测及时性差。基于主机的早期检测方法在检测准确率方面还存在较大的提升空间。基于流量的早期检测方法需要等待样本建立网络连接后才能实施检测。因此，为了在提升挖矿恶意软件检测及时性的同时也获取高的检测精度，本文以挖矿恶意软件的API名称、API操作类别、调用API的DLL作为分析对象，设计了一种AECD词嵌入方法以更充分地描述挖矿恶意软件运行初期的行为信息，并以此为基础构建挖矿恶意软件早期检测方法。

2 挖矿恶意软件行为分析

2.1 挖矿恶意软件网络连接行为分析

为实施挖矿恶意软件早期检测，曹传博等人[12]提出了挖矿软件行为多样期BDP（behavioral diversity period of cryptominer）的概念，以挖矿恶意软件建立网络连接之前的行为作为早期检测的对象，并使用socket/WSASocket在API序列中定位建立网络连接行为，同时设置了一个数据收集时间上限4 s。本文同样使用BDP内的API序列作为检测对象，但是试图解决该文献方法存在的两个方面的问题：一是部分挖矿恶意软件首次调用socket/WSASocket的时间相对较晚而使得对这部分软件的检测及时性较差。二是部分挖矿恶意软件在开始运行后的短时间内即调用了大量的API，所以该文献方法难以稳定地限制挖矿恶意软件在其被检出前的操作数量。

软件在Windows系统中运行时会通过DLL调用API来实现其功能，所以DLL调用也能够在一定程度上反映软件运行中的行为信息。本文将某个DLL调用首个API的时刻作为首次调用该DLL的时刻。由于挖矿恶意软件执行挖矿前需要通过socket连接矿池，所以本文推测socket相关DLL（ws2_32.dll、wsock32.dll、mswsock.dll）会被挖矿恶意软件调用，且存在socket相关DLL的首次调用时刻早于socket相关API（socket/WSASocket）的首次调用时刻的情况。为了验证这一推测，收集了涵盖多种加密货币类型的100款挖矿恶意软件，在VMware创建的Windows10虚拟机中收集各样本在其默认的进程CPU利用率阈值下运行1 min（以样本开始执行为起始时间）所产生的API序列，并记录调用这些API的DLL序列。所收集样本的socket相关DLL和socket/WSASocket的首次调用时刻情况对比如图1所示。

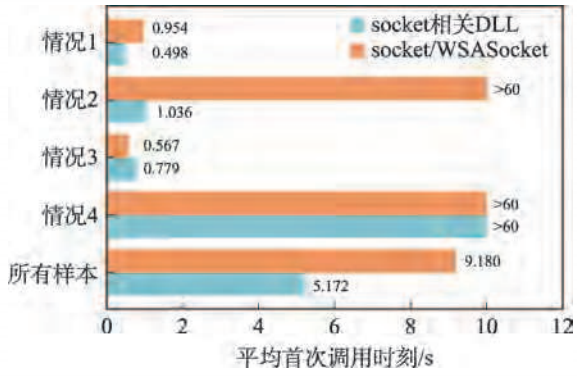

图1 挖矿恶意软件socket相关DLL和API首次调用时刻对比

Fig.1 Comparison of first call time of socket related DLLs and APIs

根据socket相关DLL和socket/WSASocket的首次调用时刻的前后关系可将所分析样本分为图1中4种不同的情况，然后分别计算每种情况所属样本的平均首次调用时刻（若样本在60 s内未调用相应API或DLL，则其首次调用时刻记为60 s）。情况1、情况2中socket相关DLL的首次调用时刻早于socket/WSASocket。其中情况2表示样本在1 min内未调用socket/WSASocket，但是调用了socket相关DLL。存在少部分样本因缺少运行环境或矿池域名失效而未调用socket相关DLL以及socket/WSASocket，被归类到情况4。部分挖矿恶意软件首次调用的socket/WSASocket来自PE文件自身而非windows的DLL，这导致其socket相关DLL的首次调用时刻较其调用socket/WSASocket的时间略晚，如图1情况3所示。综合来看，所有所分析样本的socket相关DLL的平均首次调用时刻相比socket/WSASocket的平均首次调用时刻提前了约4 s。由上述结果可知，若以软件首次调用socket/WSASocket和socket相关DLL较早的那个时刻来定位建立网络连接行为，在序列采集时长上少于单独使用socket/WSASocket来定位网络连接行为的方法。

2.2 挖矿恶意软件DLL及API类别分析

调用API的DLL能够访问该API，可以体现该API所服务的任务类型，所以软件运行时的DLL调用情况能一定程度反映其行为倾向。由2.1节可知挖矿恶意软件大多会调用socket相关DLL，这与其连接矿池的行为相对应。而除了网络连接之外，挖矿恶意软件还存在诸如进程注入、系统监控、创建副本等其他行为，与良性软件具有明显不同的行为倾向。因此，本文推测挖矿恶意软件和良性软件在DLL调用上具有明显区别。

为分析挖矿恶意软件运行初期的DLL调用情况，在Windows10虚拟机中收集2.1节所分析样本在各自默认的进程CPU利用率阈值下所产生的首个长度为3 000的API序列。为进行比较，也收集了来自多种类型的100款良性软件开始运行后所产生的首个长度为3 000的API序列。本文以调用这些API的DLL序列来计算各DLL被各软件调用的频率。表1给出了挖矿恶意软件运行初期频繁调用的部分DLL，频繁调用cudart64_80.dll、cudart64_75.dll是因为挖掘加密货币需要借助CUDA（computer unified device architecture）来实现高速的加密计算。

表1 挖矿恶意软件运行初期频繁调用的部分DLL

Table 1 Part of DLLs frequently called by cryptomining malware in early stage of operation

挖矿恶意软件常用DLL	DLL功能描述
ulib.dll	文件工具支持DLL
cudart64_80.dll	CUDA相关
cudart64_75.dll	CUDA相关
cuda_djezo.dll	CUDA相关
libcurl-4.dll	Windows系统相关

良性软件运行初期频繁调用的部分DLL如表2所示，其运行初期更倾向于调用与图形界面、系统框架相关的DLL。为量化各DLL对于区分挖矿恶意软件和良性软件的价值，本文基于各DLL在挖矿恶意软件和良性软件中的调用频率，计算每个DLL的信息增益[14]并使用IG-DLL（information gain of dynamic link library）表示其值，计算方法见3.2节。IG-DLL值越高表示该DLL对于区分挖矿恶意软件和良性软件越重要，因此本文将IG-DLL称为挖矿相关度。挖矿恶意软件运行初期所调用的DLL中IG-DLL值排名前10的DLL名称如表3所示，其中self表示样本自身。

表2 良性软件运行初期频繁调用的部分DLL

Table 2 Part of DLLs frequently called by benign software in early stage of operation

DLL名称	DLL功能描述
apphelp.dll	应用程序兼容性客户端库
kernelbase.dll	控制系统的内存管理、数据的输入输出操作和中断处理等
comctl32.dll	用户体验控件库
user32.dll	Windows用户界面相关程序接口
msvcr90.dll	Visual C++运行库

表3 挖矿恶意软件运行初期调用的具有高IG-DLL值的DLL

Table 3 DLLs with high IG-DLL values called by cryptomining malware in early stage of operation

DLL名称	IG-DLL值
kernelbase.dll	0.627 5
self	0.451 9
user32.dll	0.438 1
kernel32.dll	0.423 9
apphelp.dll	0.358 9
ole32.dll	0.236 0
comctl32.dll	0.232 6
ntdll.dll	0.211 7
powrprof.dll	0.198 1
version.dll	0.182 8

除DLL之外，有研究指出API所属的操作类别可用于提升恶意软件的分类精度[15]。不同的API根据其功能可以归属为不同的操作类别，比如文件系统操作、注册表操作、字符串操作、进程操作、内存操作等。本文以2.2节中所述的API序列为分析对象，得到了挖矿恶意软件和良性软件运行初期平均调用频率之差排名前5的API操作类别，如表4所示。表4显示相比于良性软件，挖矿恶意软件在运行初期会进行更多的字符串和注册表操作。由于API名称未包含这些操作类别所蕴含的软件行为类型信息，所以综合API名称和其操作类别信息有助于提升检测精度。

综上，调用API的DLL的信息增益值（即IG-DLL值）可体现该DLL对于区分挖矿恶意软件的重要程度，API的操作类别可以体现软件的软件行为类型信息。因此，本文将综合调用API的DLL和API操作类别来丰富软件API序列所表达的信息，以缓解可用于检测的挖矿恶意软件早期行为信息量不足的问题。

表4 不同类别软件运行初期调用API的主要操作类别

Table 4 Main operational categories of API called by different types of software in early stage of operation 单位：%

API操作类别	挖矿恶意软件	良性软件
Strings	16.61	9.83
String Manipulation	11.82	7.53
Registry	5.30	1.29
Errors	7.63	3.73
Loader	1.97	0.87

3 CEDMA方法框架

基于第2章的分析，本文提出了CEDMA。CEDMA包含如图2所示的三个阶段：数据收集、AECD词嵌入、训练和检测。在数据收集阶段，CEDMA收集样本的API名称、API操作类别以及调用API的DLL；在AECD词嵌入阶段，CEDMA通过AECD词嵌入方法得到AECD词向量；在训练和检测阶段，CEDMA基于样本的词向量来构建检测模型以及给出待测样本的检测结果。下面分别进行详细介绍。

3.1 数据收集

本阶段CEDMA会在受控环境中对训练样本运

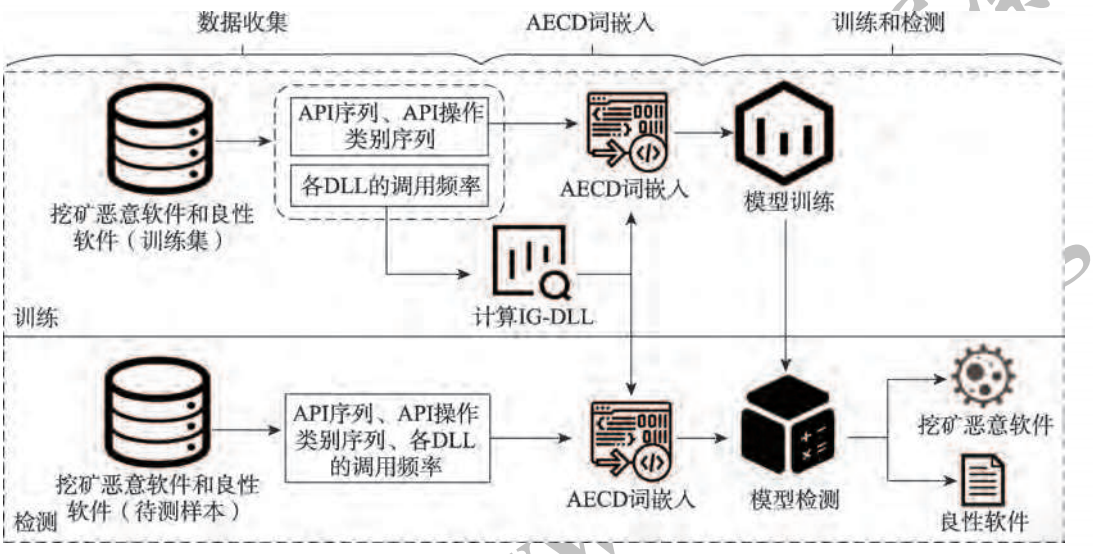

图2 CEDMA框架

Fig.2 CEDMA framework

行时产生的API序列进行收集，并提取API的操作类别和调用API的DLL，由此得到与API序列长度相同的DLL序列和API操作类别序列各一。对于训练样本，本文参考文献[12]的方法，分别收集各挖矿恶意软件样本分别在25%、50%、75%、100%的进程CPU利用率阈值下调用的API序列。

2.1节提到以时间上限（记为T）作为数据收集阈值的方法难以稳定地限制检出挖矿恶意软件前期的操作数量。表5统计了2.1节所分析的挖矿恶意软件的API序列，可以发现样本开始运行后前T秒的API序列长度大且随时间增长。表6显示若以固定的API序列长度来结束数据收集，挖矿恶意软件和良性软件调用长度在4 000以内的API序列所需时长通常很少。因此，CEDMA以API序列长度为限（记为L）来收集数据，即当软件开始运行后其调用的API序列长度首次到达某个阈值时停止收集。

表5 不同类别软件在不同T值下的平均API序列长度

Table 5 Average length of API sequences called by different types of software for different T values

T/s	挖矿恶意软件	良性软件
1	9 055	14 040
2	15 841	28 409
3	20 635	41 293
4	24 212	53 487

综合2.1节和本节的分析，CEDMA的数据收集方法如下：若样本在开始运行后首次调用socket相关DLL（ws2_32.dll、wsock32.dll、mswsock.dll）或socket/WSASocket时其API序列长度不足L，则在该首次调用时刻结束API序列收集，不足L的部分用0填充；若样本在开始运行后产生的API序列长度达到L时未调用socket相关DLL或socket/WSASocket，则在长度达到L时结束收集。

表6 不同类别软件不同L值的平均调用时间

Table 6 Average call time of different types of software for different L values

API序列长度	挖矿恶意软件/s	良性软件/s
1 000	0.204	0.200
2 000	0.311	0.393
3 000	0.412	0.567
4 000	0.507	0.732

3.2 AECD词嵌入

在进行模型训练和检测之前，CEDMA需要将API序列处理为可输入深度学习模型的数值形式，即对API进行词嵌入。一方面，不同的API对检测具有不同的价值，并非所有API都能使深度学习模型学习到对分类有帮助的信息[15]。另一方面，根据2.2节的分析，API名称未提供其操作类别信息以及挖矿相关度代表的类别区分信息。因此，为了缓解仅使用API名称进行早期检测而存在的行为信息不足问题，本文提出词嵌入方法AECD。CEDMA中AECD的具体步骤如图3所示。

步骤1 基于训练样本在数据收集阶段得到固定长度的API序列使用word2vec算法[16]来生成每个API的词向量。一个长度为n的API序列中的每个API均可映射为一个维度为d的向量$\boldsymbol{U}_j=[u_{j,1},u_{j,2},\cdots,u_{j,d}],j\in1,2,\cdots,n$，如图3所示。本文将这一步得到的词向量$\boldsymbol{U}_j$称为原生API词向量。这里使用

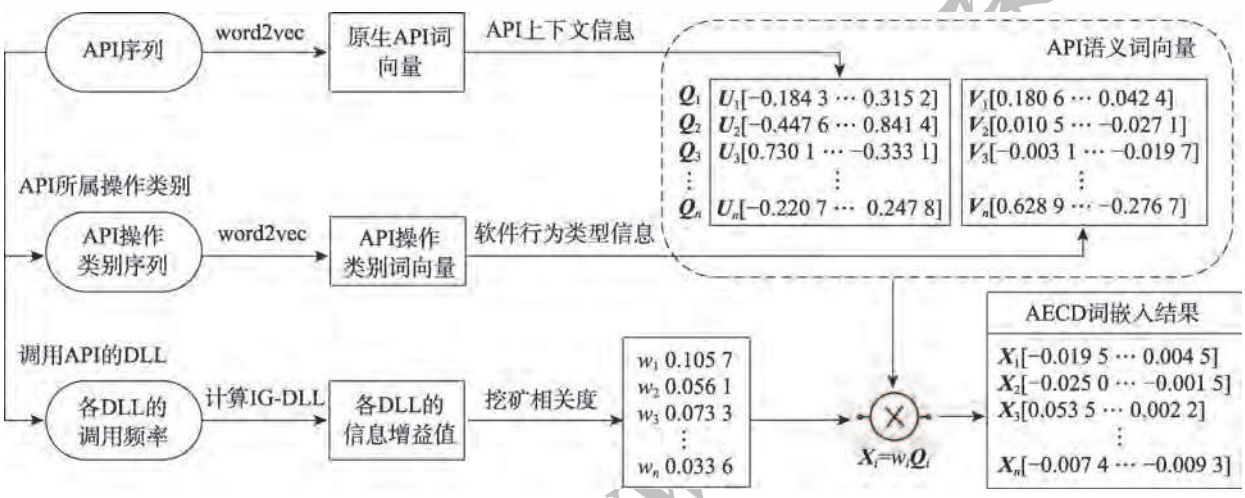

图3 AECD词嵌入方法

Fig.3 AECD embedding method

word2vec算法的原因是其生成词向量过程中会考虑API的上下文调用关系，并且具有计算速度快、词向量维度低的优点。

步骤2 基于训练样本的API操作类别序列（其长度与API序列相同）使用word2vec算法生成与原生API词向量维度相同的API操作类别词向量，再将两者拼接为一个词向量。一个长度为 n 的API操作类别序列中的每个API操作类别均可映射为一个维度为 d 的向量 $\boldsymbol{V}_j=[v_{j,1},v_{j,2},\cdots,v_{j,d}],j\in 1,2,\cdots,n$，与原生API词向量水平拼接后的结果为 $\boldsymbol{Q}_j=[u_{j,1},u_{j,2},\cdots,u_{j,d},v_{j,1},v_{j,2},\cdots,v_{j,d}]$，$j\in 1,2,\cdots,n$。本文将这一步得到的维度为 $2\times d$ 的词向量 $\boldsymbol{Q}_j$ 称为API语义词向量。如2.2节的内容所述，API的操作类别蕴含了API的软件行为类型信息，该步骤可以增大属于相同操作类别的API之间的相似度。比如同属注册表操作的NtQueryValueKey和NtOpenKeyEx，两者的原生API词向量的余弦相似度为0.754 1，而拼接操作类别词向量之后的API语义词向量的相似度升高到0.842 1。

步骤3 基于训练样本的DLL调用频率计算IG-DLL，然后用其对API语义词向量加权。API语义词向量虽然通过API操作类别丰富了软件行为类型信息，但是仍然缺少对与挖矿恶意软件相关API的重点关注。如2.2节所述，IG-DLL能够体现API对区分挖矿恶意软件和良性软件的贡献度。因此，AECD词嵌入的最后一步为使用IG-DLL对API语义词向量进行加权。如图3所示，根据调用API的DLL可以计算各DLL的调用频率，进而基于各训练样本的DLL序列（其长度与API序列相同）统计出各DLL的调用频率，进而计算出各个DLL的IG-DLL值。在训练阶段，一个长度为 n 的API序列所对应的DLL序列的IG-DLL值可以组成向量 $[w_1,w_2,\cdots,w_i,\cdots,w_n],i\in 1,2,\cdots,n$。第 i 个API的加权词向量可由 $\boldsymbol{X}_i=w_i\boldsymbol{Q}_i$ 计算得到，$\boldsymbol{X}_i$ 即为API序列中第 i 个API的AECD词向量。

上述过程中各DLL的IG-DLL值由式（1）计算得到。其中，$GR(X,DLL_i)$ 表示 DLL_i 的IG-DLL值，X 表示训练集且 X 中各样本由该样本各DLL的调用频率表征，$H(X)$ 表示 X 的信息熵，$H(X|DLL_i)$ 表示使用 DLL_i 的调用频率来对 X 进行样本划分的情况下 X 的信息熵。若 X 包含 k 个不同的样本类别 $class_i,i=1,2,\cdots,k$（本文中 k 为2，即 X 含有挖矿恶意软件与良性软件两个类别），且 DLL_i 的调用频率在 X 中共有 s 种不同取值 $num_j(DLL_i),j=1,2,\cdots,s$，则 $H(X)$ 和 $H(X|DLL_i)$ 分别可以根据式（2）、（3）计算得到。其中，$p(f_c)$ 表示第 c 类样本在 X 中的比例，$p(num_j(DLL_i))$ 表示 DLL_i 的调用频率为 $num_j(DLL_i)$ 的样本在 X 中的比例，$X|num_j(DLL_i)$ 表示 X 中 DLL_i 的调用频率为 $num_j(DLL_i)$ 的样本组成的集合。

$$GR(X,DLL_i)=H(X)-H(X|DLL_i) \tag{1}$$

$$H(X)=-\sum_{c=1}^{k}p(f_c)\mathrm{lb}\,p(f_c) \tag{2}$$

$$H(X|DLL_i)=\sum_{j=1}^{s}p(num_j(DLL_i))H(X|num_j(DLL_i)) \tag{3}$$

由于调用某个API的DLL不固定，在CEDMA的检测阶段，AECD词嵌入需要分为两步完成：首先针对待测API序列中的API及其操作类别，基于训练阶段的词嵌入结果将每个API映射为维度为 $2\times d$ 的API语义词向量，训练样本中未出现的API则映射为全0向量。然后根据调用待测API序列中各API的DLL以及训练阶段计算的各DLL的IG-DLL，将API语义词向量加权为AECD词向量。综上，待测API序列的每个API都映射为一个维度为 $2\times d$ 的AECD词向量。

3.3 模型训练与检测

为训练一个可实施挖矿恶意软件与良性软件的二分类的检测模型，需要先给每个训练样本赋上其所属的类别标签——良性软件和挖矿恶意软件分别标记为0和1。据此可以得到训练集 $D_{\mathrm{tr}}(D_{\mathrm{tr}}=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_i,y_i),\cdots,(\boldsymbol{x}_m,y_m)\})$，其中 $y_i\in\{0,1\}$，$\boldsymbol{x}_i$ 由第 i 个样本的AECD词向量表征，m 为样本数量。

CEDMA使用TextCNN构建检测模型。TextCNN是CNN（convolutional neural networks）用于文本分类任务的一种变体[7]，其可以利用一个或多个一维滑动窗口从序列数据中提取特征。一方面，CEDMA使用TextCNN可以从AECD词向量中自动提取特征并实施分类。另一方面，TextCNN模型结构简单、参数少、训练速度快。

CEDMA使用的TextCNN模型由输入层、卷积层、池化层、全连接层和输出层组成。输入层将样本运行记录（包含API序列、API操作类别序列和DLL序列）转化为AECD词向量输出。假设输入的样本运行记录中API序列长度为 n，词向量维度为 d，则经输入层后为 $n\times d$ 的矩阵。若第 i 个API的词向量表示为 $\boldsymbol{X}_i$，则一个样本运行记录经过词嵌入后可以表示为式（4）：

$$\boldsymbol{X}_{(1:n)}=\boldsymbol{X}_1\oplus\boldsymbol{X}_2\oplus\cdots\oplus\boldsymbol{X}_n \tag{4}$$

卷积层可以设置不同大小的卷积核。设卷积核

的大小为 $h\times d$，其中 h 代表卷积核高度，即用于提取特征的相邻API调用的数量，d 代表词向量的维度。卷积核会在 $\boldsymbol{X}_{(1:n)}$ 上纵向滑动，通过式(5)计算特征值 c_i。式中 $\boldsymbol{w}$ 表示权重，b 表示偏差值，f 表示激活函数，$\boldsymbol{X}_{i:(i+h-1)}$ 表示第 i 到第 $i+h-1$ 个API词向量构成的矩阵。

$$c_i=f(\boldsymbol{w}\cdot\boldsymbol{X}_{i:(i+h-1)}+b) \tag{5}$$

如式(6)所示，一个卷积核通过纵向移动最终可以获得 $n-h+1$ 个特征值。

$$\boldsymbol{c}=[c_1,c_2,\cdots,c_{n-h+1}] \tag{6}$$

池化层采取最大池化的方法，卷积结果 $\boldsymbol{c}$ 的池化过程可由式(7)表示：

$$c'=\max(\boldsymbol{c}) \tag{7}$$

所有池化后的特征值经过全连接层的拼接，可以得到一个完整的向量。若有 p 种卷积核，每种卷积核的个数为 $[q]$，则全连接层输出的向量如式(8)所示：

$$\boldsymbol{z}=[c_1',c_2',\cdots,c_{(p\times q)}'] \tag{8}$$

最后，输出层会输出一个长度为 s 的向量，s 为类别数量。该向量的值表示预测样本属于各类别的概率。

在检测步骤，将采集的待检测软件的API序列、API操作类别序列和DLL序列作为样本运行记录，然后将其输入检测模型以判别该软件是良性软件还是挖矿恶意软件。

4 实验及结果

4.1 实验环境与样本

样本的运行环境为Windows 10的VMware虚拟机，Intel i7 10700F CPU和4 GB内存。使用API Monitor收集样本运行记录。检测模型所在主机的配置为AMD EPYC 7742 64-Core Processor，1 TB内存，1块A100-SXM4-40 GB，模型采用Python 3.8并使用keras 2.4.3和tensorflow-gpu 2.4.0库实现。CEDMA在TextCNN模型中设置了高度分别为2、3、4、5的4种卷积核，分别用于4个卷积层，然后加入了一个Dropout层，最后采用sigmoid函数输出标签概率。TextCNN的超参数设置如下：学习率为0.000 1，Dropout为0.2，Epoch为20，Batch size为32。

实验样本包括175个挖矿恶意软件样本和700个良性软件样本。挖矿恶意软件样本来源于github项目、virusshare(https://virusshare.com)、微步在线云沙箱(https://s.threatbook.cn)、malware traffic analysis(https://malware-traffic-analysis.net)，以及any.run(https://app.any.run)。良性软件样本包括办公软件、安全杀毒、加密软件等类别。良性软件样本来源于各软件的官网或360软件安装管家。

根据3.1节所述的数据收集方法，每个挖矿恶意软件样本收集4条API序列(含API名称、API所属的操作类别和调用API的DLL)，而每个良性软件收集1条API序列。为了验证CEDMA对已知和未知挖矿恶意软件的检测能力，本文将收集的API序列划分为表7所示的训练集和两个测试集。首先，随机选取140个挖矿恶意软件的各2条API序列，再随机取280条良性软件的API序列，组成包含560条API序列的训练集。接着选取上述140个挖矿恶意软件未被选入训练集的另外2条API序列，与随机选择的280条良性软件API序列共同组成包含560条API序列的测试集1。之后将在训练集中未出现过的35个挖矿恶意软件的各4条API序列和另外140条良性软件API序列组成包含280条API序列的测试集2。测试集1和测试集2分别可以评估CEDMA对设置了不同进程CPU利用率阈值的已知挖矿恶意软件和未知挖矿恶意软件的检测能力。

表7 实验中API序列的种类及数量

Table 7 Types and quantities of API sequences in experiment

类别	训练集	测试集1	测试集2	总数
办公软件	42	42	21	105
聊天通讯	31	31	15	77
安全杀毒	16	16	8	40
影音软件	48	48	24	120
游戏	32	32	16	80
压缩软件	16	16	8	40
压力测试	5	5	4	14
加密软件	20	20	10	50
其他应用	70	70	34	174
良性软件API序列数	280	280	140	700
挖矿恶意软件API序列数	280	280	140	700
API序列总数	560	560	280	1 400

4.2 评估指标

为了对CEDMA的检测模型进行评估，本文使用了多个恶意软件检测中常用的评估标准，包括准确率(Accuracy)，精确率(Precision)，召回率(Recall)和F1-score(F1)[9]。

4.3 实验结果

本节给出了CEDMA在测试集1和测试集2上的

检测结果。另外，也进行了CEDMA在不同API序列长度L下的结果对比。

4.3.1 测试集1的检测结果

CEDMA在不同L值下的检测结果如表8所示。表8显示当L在{1 000, 2 000, 3 000}内取值时，CEDMA的检测结果随着L的增大而上升，但是当L取值4 000时检测结果下降。经分析，这是因为L在{1 000, 2 000, 3 000}内取值时，随着L值的增大，大多数挖矿恶意软件还未进入挖矿阶段，并在此期间执行了越来越多的行为，其API序列中与字符串、注册表等操作相关的API逐渐增多，与良性软件API序列区别明显。而L=4 000时，一部分挖矿恶意软件建立网络连接前的行为已经结束，使得API序列中与这些行为相关的API比例下降，与良性软件API序列的相似度升高。

表8 CEDMA使用不同L值在测试集1上的检测结果

Table 8 Detection results of CEDMA on test set 1 with different values of L

L	Accuracy	Recall	Precision	F1
1 000	0.975 0	**0.985 7**	0.965 0	0.975 3
2 000	0.980 4	**0.985 7**	0.975 3	0.980 5
3 000	**0.982 1**	**0.985 7**	**0.978 7**	**0.982 2**
4 000	0.935 7	0.978 5	0.901 3	0.938 3

为评估不同词向量对CEDMA检测结果的影响，将L=3 000的API序列生成的原生API词向量、API语义词向量和AECD词向量分别作为TextCNN的输入，在测试集1上得到的检测结果如表9所示。表9显示TextCNN使用API语义词向量得到的检测精度相比使用原生API词向量有所提升；TextCNN使用AECD词向量获得的Accuracy和F1值则优于使用另外两种词向量获得的值。

表9 CEDMA使用不同词向量在测试集1上的检测结果

Table 9 Detection results of CEDMA on test set 1 using different word vectors

词向量	Accuracy	Recall	Precision	F1
原生API词向量	0.958 9	0.960 7	0.957 3	0.959 0
API语义词向量	0.980 4	**0.985 7**	0.975 3	0.980 4
AECD词向量	**0.982 1**	**0.985 7**	**0.978 7**	**0.982 2**

CEDMA与现有数种挖矿恶意软件检测方法在测试集1上的检测结果如表10所示，其中平均检测时间为一个方法检测单个测试样本所需的平均数据收集时间、平均特征计算时间（生成特征向量所需时间）及平均分类时间的总和，“—”表示无此项。

表10 不同检测方法在测试集1上的检测结果

Table 10 Detection results of different detection methods on test set 1

方法	Accuracy	F1	平均检测时间/s
Berecz[8]	0.950 0	0.951 0	3.650 6
Karn[9]	0.941 1	0.941 8	60.582 9
CEDMB[12]	0.966 1	0.966 3	2.118 1
CEDMA	**0.982 1**	**0.982 2**	**0.493 7**

文献[8]未说明其方法的数据收集时间，对比实验中其数据收集方法设置为与CEDMA相同，文献[9]方法在实验中使用API。在检测精度方面，表10显示CEDMA在测试集1上获得的Accuracy和F1值优于其他几种方法。这是因为CEDMA所用的AECD词向量融合的软件行为信息（API名称、API操作类别和调用API的DLL）多于其他几种方法所用于检测的软件行为信息。在检测及时性方面，CEDMA的平均检测时间也优于其他几种方法。这主要因为CEDMA以挖矿恶意软件建立网络连接前调用的较短长度的API序列信息为检测对象，所需的软件行为采集时间少于另外几种方法，且生成AECD词向量所需时间少以及所用的TextCNN模型结构简单而具有很高的分类效率。

4.3.2 测试集2的检测结果

CEDMA在不同L值下在测试集2上得到的检测结果如表11所示。当L在{1 000, 2 000, 3 000, 4 000}内取值时，表11显示CEDMA在L=3 000时获得的检测结果最佳，其Accuracy和F1分别为96.76%和96.77%。

表11 CEDMA使用不同L值在测试集2上的检测结果

Table 11 Detection results of CEDMA on test set 2 with different values of L

L	Accuracy	Recall	Precision	F1
1 000	0.921 4	0.907 1	0.933 8	0.920 3
2 000	0.928 6	0.928 6	0.928 6	0.928 6
3 000	**0.967 6**	**0.964 3**	**0.971 2**	**0.967 7**
4 000	0.907 1	0.864 3	0.945 3	0.903 0

将L=3 000的API序列生成的三种词向量作为TextCNN的输入，在测试集2上得到的检测结果如表12所示。表12显示TextCNN使用AECD词向量获得的Accuracy和F1值在对比的三种词向量中最佳。

表12 CEDMA使用不同词向量在测试集2上的检测结果

Table 12 Detection results of CEDMA on test set 2 using different word vectors

词向量	Accuracy	Recall	Precision	F1
原生API词向量	0.946 4	0.907 1	**0.984 5**	0.944 2
API语义词向量	0.950 0	0.950 0	0.950 0	0.950 0
AECD词向量	**0.967 6**	**0.964 3**	0.971 2	**0.967 7**

表13给出了CEDMA与现有数种挖矿恶意软件检测方法在测试集2上的检测结果。表13显示CEDMA在测试集2上得到的Accuracy和F1值优于其他几种方法，所需的平均检测时间也最少。这主要因为与其他几种方法相比，CEDMA所用的AECD词向量融合的软件行为信息最多，所需的软件行为采集时间和特征向量生成时间之和最少。

表13 不同检测方法在测试集2上的检测结果

Table 13 Detection results of different detection methods on test set 2

方法	Accuracy	F1	平均检测时间/s
Berecz[8]	0.921 4	0.918 5	2.669 8
Karn[9]	0.907 1	0.905 1	60.415 2
CEDMB[12]	0.946 4	0.945 1	2.110 9
CEDMA	**0.967 6**	**0.967 7**	**0.749 6**

4.4 实验总结

经上述实验验证，CEDMA能够以挖矿恶意软件运行初期所执行的少量行为将其高准确率地检出。表9和表12的实验结果表明，相比于使用API词向量，使用AECD词向量有助于获取更高的检测精度。表10和表13的实验结果表明：与对比方法相比，CEDMA在检测精度和及时性方面均具有优势，更适用于挖矿恶意软件早期检测。

5 结束语

本文针对当前挖矿恶意软件早期阶段的API名称所提供信息有限的问题，在挖矿恶意软件早期阶段的API序列中融合了API名称、API操作类别和调用API的DLL以更充分地描述其初期的行为信息，进而提出了一种词嵌入方法ACED，并由此提出了挖矿恶意软件早期检测方法CEDMA。实验结果表明，CEDMA只需收集BDP内长度为3 000的API序列即可分别以98.22%、96.77%的F1值检测数据集中已知和未知的挖矿恶意软件样本。相比目前其他挖矿恶意软件检测方法，CEDMA在检测及时性和准确度上均具有优势，在挖矿恶意软件早期检测领域具有应用价值。在后续工作中，将继续探索如何准确检测应用了逃逸技术的挖矿恶意软件。

参考文献：

[1] BADAWI E, JOURDAN G V. Cryptocurrencies emerging threats and defensive mechanisms: a systematic literature review[J]. IEEE Access, 2020, 8: 200021-200037.

[2] Sonicwall. 2023 Sonicwall cyber threat report[EB/OL]. (2023-03-23) [2023-04-24]. https://www.sonicwall.com/2023-cyber-threat-report/.

[3] PASTRANA S, SUAREZ-TANGIL G. A first look at the crypto-mining malware ecosystem: a decade of unrestricted wealth [C]//Proceedings of the Internet Measurement Conference, Amsterdam, Oct 21-23, 2019. New York: Association for Computing Machinery, 2019: 73-86.

[4] BIJMANS H L J, BOOIJ T M, DOERR C. Inadvertently making cyber criminals rich: a comprehensive study of crypto-jacking campaigns at internet scale[C]//Proceedings of the 28th USENIX Security Symposium, Santa Clara, Aug 16-19, 2019. Berkeley: USENIX Association, 2019: 1627-1644.

[5] MANI G, PASUMARTI V, BHARGAVA B, et al. Decrypto pro: deep learning based cryptomining malware detection using performance counters[C]//Proceedings of the 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems, Washington, Aug 17-21, 2020. Piscataway: IEEE, 2020: 109-118.

[6] AL-RIMY B A S, MAAROF M A, ALAZAB M, et al. Redundancy coefficient gradual up-weighting-based mutual information feature selection technique for crypto-ransomware early detection[J]. Future Generation Computer Systems, 2021, 115: 641-658.

[7] KIM Y. Convolutional neural networks for sentence classification[J]. arXiv:1408.5882, 2014.

[8] BERECZ G J, CZIBULA I G. Hunting traits for cryptojackers[C]//Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, Prague, Jul 26-28, 2019. Francisco: SciTePress, 2019: 386-393.

[9] KARN R R, KUDVA P, HUANG H, et al. Cryptomining detection in container clouds using system calls and explainable machine learning[J]. IEEE Transactions on Parallel and Distrib-

uted Systems, 2020, 32(3): 674-691.
[10] I MUÑOZ J Z, SUÁREZ-VARELA J, BARLET-ROS P. Detecting cryptocurrency miners with NetFlow/IPFIX network measurements[C]//Proceedings of the 2019 IEEE International Symposium on Measurements & Networking, Catania, Jul 8-10, 2019. Piscataway: IEEE, 2019: 1-6.
[11] CAPROLU M, RAPONI S, OLIGERI G, et al. Cryptomining makes noise: detecting cryptojacking via machine learning[J]. Computer Communications, 2021, 171: 126-139.
[12] 曹传博, 郭春, 申国伟, 等. 面向行为多样期的挖矿恶意软件早期检测方法[J]. 电子学报, 2023, 51(7): 1850-1858.
CAO C B, GUO C, SHEN G W, et al. Cryptomining malware early detection method in behavioral diversity period[J]. Acta Electronica Sinica, 2023, 51(7): 1850-1858.
[13] SUN P F, LYU M D, LI H, et al. An early stage convolutional feature extracting method using for mining traffic detection [J]. Computer Communications, 2022, 193: 346-354.
[14] YING Q, YU Y, TIAN D, et al. CJSpector: a novel cryptojacking detection method using hardware trace and deep learning[J]. Journal of Grid Computing, 2022, 20(3): 1-15.
[15] TANG M, QIAN Q. Dynamic API call sequence visualisation for malware classification[J]. IET Information Security, 2019, 13(4): 367-377.
[16] MIKOLOV T, SUTSKEVER I, CHEN K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in Neural Information Processing Systems, 2013.

曹传博（1998—），男，湖北孝感人，硕士，CCF学生会员，主要研究方向为计算机网络与信息安全。

CAO Chuanbo, born in 1998, M.S., CCF student member. His research interests include computer network and information security.

郭春（1986—），男，贵州贵阳人，博士，教授，CCF高级会员，主要研究方向为恶意代码检测、入侵检测。

GUO Chun, born in 1986, Ph.D., professor, CCF senior member. His research interests include malicious code detection and intrusion detection.

李显超（1979—），男，河南邓州人，硕士，主要研究方向为数据中心、物联网、云计算。

LI Xianchao, born in 1979, M.S. His research interests include data center, Internet of things and cloud computing.

申国伟（1986—），男，湖南邵东人，博士，教授，CCF高级会员，主要研究方向为网络与信息安全、大数据。

SHEN Guowei, born in 1986, Ph.D., professor, CCF senior member. His research interests include computer network and information security, big data.