计算机科学
COMPUTER SCIENCE

基于样本嵌入的挖矿恶意软件检测方法

傅建明, 姜宇谦, 何佳, 郑锐, 苏日古嘎, 彭国军

引用本文

傅建明, 姜宇谦, 何佳, 郑锐, 苏日古嘎, 彭国军. 基于样本嵌入的挖矿恶意软件检测方法[J]. 计算机科学, 2024, 51(1): 327-334.

FU Jianming, JIANG Yuqian, HE Jia, ZHENG Rui, SURI Guga, PENG Guojun. Cryptocurrency Mining Malware Detection Method Based on Sample Embedding [J]. Computer Science, 2024, 51(1): 327-334.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230100116

基于样本嵌入的挖矿恶意软件检测方法

傅建明[1]　姜宇谦[1]　何　佳[2]　郑　锐[3]　苏日古嘎[1]　彭国军[1]
1 武汉大学国家网络安全学院空天信息安全与可信计算教育部重点实验室　武汉 430072
2 嵩山实验室技术中心　郑州 450046
3 河南大学计算机与信息工程学院　河南 开封 475000

摘　要　加密货币挖矿恶意软件的高盈利性和匿名性,对计算机用户造成了巨大威胁和损失。为了对抗挖矿恶意软件带来的威胁,基于软件静态特征的机器学习检测器通常选取单一类型的静态特征,或者通过集成学习来融合不同种类静态特征的检测结果,忽略了不同种类静态特征之间的内在联系,其检测率有待提升。文章从挖矿恶意软件的内在层级联系出发,自下而上提取样本的基本块、控制流程图和函数调用图作为静态特征,训练三层模型以将这些特征分别嵌入向量化,并逐渐汇集从底层到高层的特征,最终输入分类器实现对挖矿恶意软件的检测。为了模拟真实环境中的检测情形,先在一个小的实验数据集上训练模型,再在另一个更大的数据集上测试模型的性能。实验结果表明,三层嵌入模型在挖矿恶意软件检测上的性能领先于近年提出的机器学习模型,在召回率和准确率上相比其他模型分别提高了7%和3%以上。

关键词:挖矿恶意软件;静态分析;机器学习;图嵌入
中图分类号　TP311

Cryptocurrency Mining Malware Detection Method Based on Sample Embedding

FU Jianming[1],JIANG Yuqian[1],HE Jia[2],ZHENG Rui[3],SURI Guga[1] and PENG Guojun[1]
1 Key Laboratory of Aerospace Information Security and Trusted Computing,Ministry of Education,School of Cyber Science and Engineering,Wuhan University,Wuhan 430072,China
2 Technology Center of Songshan Laboratory,Zhengzhou 450046,China
3 College of Computer and Information Engineering,Henan University,Kaifeng,Henan 475000,China

Abstract　Due to its high profitability and anonymity,cryptocurrency mining malware poses a great threat and loss to computer users. In order to confront the threat posed by mining malware,machine learning detectors based on software static features usually select a single type of static features,or integrate the detection results of different kinds of static features through integrated learning,ignoring the internal relationship between different kinds of static features,and its detection rate remains to be discussed. This paper starts from the internal hierarchical relationship of mining malware. It extracts basic blocks,control flow graphs and function call graphs of samples as static features,trains the three-layer model to embed these features into the vector respectively,and gradually gathers the features from the bottom to the top,and finally sends top features to the classifier to detect mining malware. To simulate the detection situation in real world,it first trains the model on a relatively smaller experimental data set,and then tests the performance of the model on another much larger data set. Experiment results show that the performance of th proposed method is much better than that of some machine learning models proposed in recent years. The recall rate and accuracy rate of three-layer-embedding model is more than 7% and 3% higher than that of other models,respectively.

Keywords　Cryptocurrency mining malware,Static analysis,Machine learning,Graph embedding

1　引言

挖矿恶意软件是近年来出现的一种新型恶意软件,随着加密货币的增值而不断传播。挖矿恶意软件通过窃取受害者计算机的计算资源来进行加密货币的挖矿,并通过各种方式将获得的加密货币变现。挖矿恶意软件与其他恶意软件有明显的区别。首先,挖矿恶意软件感染计算机的目的是窃取受害者的计算资源[1],而常见的恶意软件主要是窃取或破坏受害者计算机中的数据。其次,由于需要进行加密计算,挖矿恶意软件会使用大量加解密函数,并消耗计算资源[2],从而导致

到稿日期:2023-01-30　返修日期:2023-07-11
基金项目:国家自然科学基金(61972297,62172308,62272351);国家重点研发计划(2021YFB3101201)
This work was supported by the National Natural Science Foundation of China(61972297,62172308,62272351) and National Key R&D Program of China(2021YFB3101201).
通信作者:傅建明(jmfu@whu.edu.cn)

受害者计算机的执行速度减慢，一些逻辑错误的挖矿恶意软件甚至会导致计算机瘫痪[3]。另一方面，挖矿恶意软件窃取的计算资源会消耗大量的电量，并导致硬件生命周期[4]缩短，受害者最终为攻击者的挖矿成本买单。由于挖矿恶意软件具有高匿名性、高收益等特点，其数量正在逐步增长。据报告[5]显示，2017 年至 2019 年挖矿恶意软件的数量增长了 15 倍，McAfee 发现，2020 年挖矿恶意软件第四季度环比增长了 53%[6]。显然，挖矿恶意软件已成为一个巨大的安全威胁。因此我们需要一个高性能、高效率的挖矿恶意软件检测方法来对抗日渐增长的挖矿恶意软件。

许多研究[7-12]表明，机器学习提供了一种有前景的大规模恶意软件检测方法。随着深度学习[13-14]的发展，这些检测方法拥有了更强大的特征处理能力，越来越多的静态特征方案被提出并应用。例如直接使用二进制文件的原始字节[15]作为特征向量，实验表明该特征处理方法取得了较好的效果。由于挖矿行为以密码学运算为主，特征更多地体现在指令序列上，一些基于二进制文件反汇编得到的特征适用于挖矿行为的检测，例如有工作以操作码(Opcode)作为特征，利用长序列双向长短期记忆(Bidirectional Long Short-Term Memory，BiLSTM)网络[16]的计算能力，实现了性能良好的挖矿恶意软件检测方法。此外，以恶意软件的控制流程图[17]为特征的方案也取得了优异的性能。

但这类方法对训练数据集的依赖程度较高，仅在较小的数据集[18]上进行测试，因此，这些结果是否能在现实世界的挖矿恶意软件检测中保持稳定值得怀疑。以上工作都只采用了单一种类的静态特征进行恶意软件检测，现有的利用多种不同静态特征进行恶意软件检测的工作主要有两种：一种是采用集成学习[19]的方式来融合不同种类静态特征的检测结果；另一种是将不同种类静态特征向量直接进行拼接合并[20]或者加权拼接合并[21]，从而提高特征向量的维度。但这两种方式都没有寻找不同特征之间的内在联系，并且没有根据这种内在联系设计模型和规则。

针对现有工作在恶意软件静态特征使用中存在的问题，本文基于恶意软件样本中基本块、控制流程图和函数调用图的内在联系，提出了一种基于三层嵌入的挖矿恶意软件检测器。通过反汇编工具输出恶意软件样本的指令序列、基本块、控制流程图和函数调用图，并计算对应的基本块嵌入、函数嵌入以及软件样本嵌入，最后使用软件样本嵌入进行挖矿恶意软件的检测。本工作除了在模拟实验环境下的小数据集上进行了训练，还在一个模拟真实世界环境的更大的数据集上进行了测试。实验结果表明，在挖矿恶意软件的检测上，三层嵌入模型的性能领先于其他机器学习模型，其在挖矿恶意软件检测的召回率上比其他模型提高了 7%以上，在准确率上比其他模型提高了 3%以上。

2 相关工作

深度学习近年来推动了高维静态特征提取的发展，越来越多的高维静态特征和相应的深度网络被提出并取得了良好的效果。

2.1 使用操作码特征的 LSTM 模型

通过对软件进行反汇编，可以获取其操作码序列，在一定程度上表征该软件的行为。Yazdinejad 等[16]使用软件的操作码作为静态特征，提出了一种深度长短期记忆网络(LSTM)模型用于挖矿恶意软件的识别，该网络在小数据集上达到了最高 98%的检测准确率。

然而，以操作码作为特征有明显的缺点，实际的恶意软件执行过程中存在大量的跳转操作，这导致输入模型的线性顺序的操作码特征与操作码的实际执行顺序存在一定的差距。

2.2 使用原始字节特征的 Malconv 模型

原始字节指二进制文件的字节值，它们可以被编制成十六进制数，并可以作为特征向量来训练恶意软件分类模型。原始字节特征具有较高的特征维数，这决定了浅层机器学习方法难以拟合分类边界。相比之下，深度学习模型适合于这类任务。Malconv 模型[15]是一种典型的基于原始字节的恶意软件检测模型，该模型以深度卷积网络为基础搭建神经网络模型，只需要输入 Windows 可执行文件的原始字节序列，就能区分该文件是良性的还是恶意的。

2.3 使用 CFG 特征的 MAGIC 模型

Yan 等[17]设计了一个名为 MAGIC 的恶意软件家族分类模型，其以恶意软件 CFG(控制流图)为输入，使用 GCN(图神经网络)进行分类。其中，CFG 中的每个基本块都由低维特征向量表示[22]，GCN 通过节点更新规则使 CFG 中的节点汇聚来自临界点的特征信息，从而对 CFG 图节点特征做进一步嵌入表达。最终，MAGIC 根据 CFG 图的嵌入向量和 Softmax 分类器获取恶意软件的分类结果。

该模型取得了和基于手工提取特征的恶意软件检测方法[20,23]相当的性能，但该模型也有不足之处。MAGIC 使用恶意软件的 CFG 作为模型的输入，而一个恶意软件样本通常由多个函数组成，每个函数可以由一个 CFG 来表示。而当恶意软件样本本身需要用 CFG 来表示时，就意味着在恶意软件样本中所有表示函数的 CFG 会共同拼接成一个表示该恶意软件的大 CFG，拼接的方式是将样本中所有表示函数的 CFG 作为大 CFG 的子图，而子图与子图之间没有任何连接。特征数据结构规模的增大，不仅会导致 GCN 计算需求的膨胀，而且在聚合出整个图的特征向量时会丢失大量的图结构信息。除此之外，MAGIC 并没有使用函数调用图 FCG 作为特征，即函数间的调用信息。

3 三层嵌入模型结构

三层嵌入模型的设计思路如下，首先从函数出发，一个软件样本由许多函数构成，每一个函数可以用一个 CFG 来表示。每个 CFG 的节点都由基本块构成，我们通过构造基本块嵌入模型，将基本块转化成一个向量表达，使得 CFG 的节点都拥有固定长度的特征向量，此时的 CFG 被称为属性化的 CFG。我们构造一个函数嵌入模型，将属性化的 CFG 输入函数嵌入模型，从而将属性化的 CFG 转化为一个向量表达，该向量为函数的嵌入向量，可以表达函数的特征。

将得到的函数嵌入向量赋值给 FCG 中该函数对应的节点，即可将 FCG 也转化为属性化的 FCG。这时就可以构造

一个软件样本嵌入模型，将属性化的 FCG 输入软件样本嵌入模型，从而将属性化的 FCG 转化为一个向量表达，该向量为软件样本的嵌入向量，可以表达软件样本的特征，并可用于后续的软件分类。

三层嵌入模型由 3 个基本模型组成，分别为底层的基本块嵌入模型、中间层的函数嵌入模型和顶层的软件嵌入模型，它们按照从底层到顶层的顺序依次提取软件样本中基本块的特征、函数(CFG)的特征和软件样本(FCG)的特征，最后基于软件嵌入向量输出判定为挖矿软件的概率。三层嵌入模型的总体结构如图 1 所示。

本章将分别介绍三层模型的结构和组合方式，以及每层模型的训练。

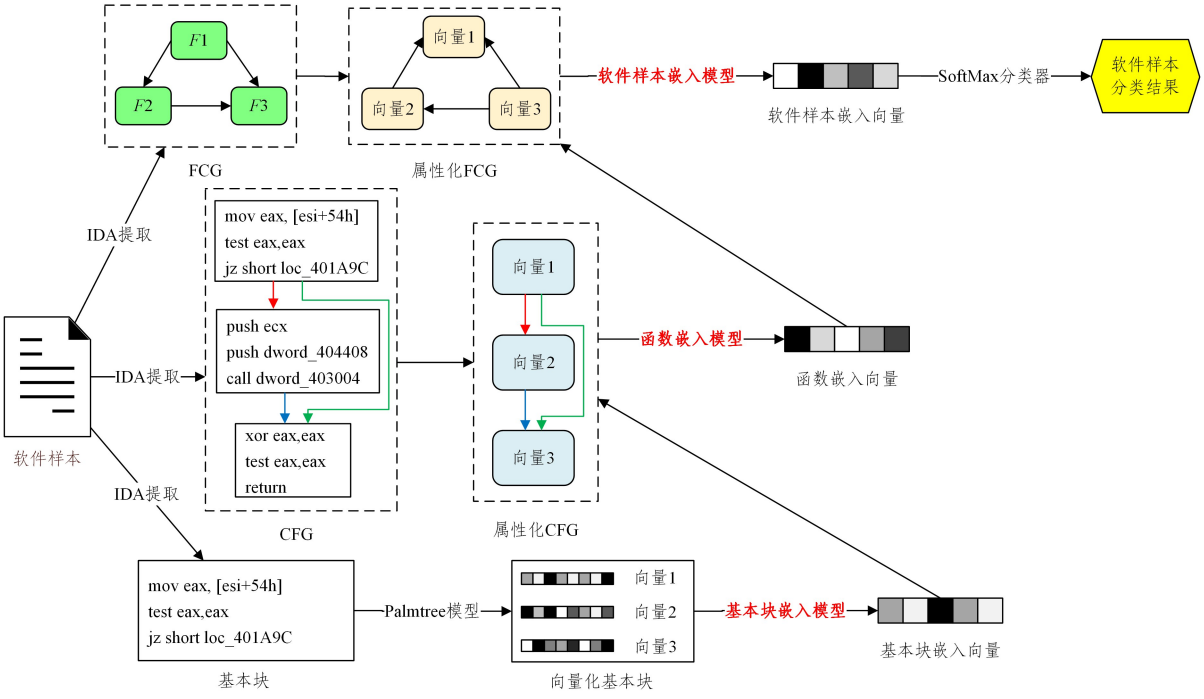

图 1　三层嵌入模型整体结构

Fig. 1　Structure of three-layer-embedding model

3.1　基本块嵌入模型

定义一个基本块为指令的集合 $B=[ins_1, ins_2, \cdots, ins_n]$，其中 ins_i 表示基本块 B 内的第 i 条指令。然后根据已开源的 Palmtree 预训练好的模型[24]，获取基本块中的所有指令嵌入。将 ins_i 输入 Palmtree 模型后，可以得到其指令嵌入 $h_i \in R^{1\times d}$，d 为嵌入的维度且 $d=128$。对于整个基本块而言，其已转化为向量化基本块。Palmtree 模型输出其特征矩阵 $\boldsymbol{H}=[\boldsymbol{h}_1^{\mathrm{T}}, \boldsymbol{h}_2^{\mathrm{T}}, \cdots, \boldsymbol{h}_n^{\mathrm{T}}]^{\mathrm{T}} \in R^{n\times d}$。基本块特征矩阵 $\boldsymbol{H}$ 的获取过程可以用式(1)表示：

$$\boldsymbol{H}=Palmtree(B) \tag{1}$$

接下来将 $\boldsymbol{H}$ 输入 Transformer 的编码组件中，我们将这个编码组件称为基本块编码组件，即基本块嵌入模型，它和 Transformer 的编码组件结构相同，由多个编码器层堆叠组成。本文按照 BERT 模型的输入处理方式来处理基本块编码组件，每一个指令嵌入 h_i 被视为一个输入 token，而基本块的整个特征矩阵 $\boldsymbol{H}$ 则可以看作由 token 组成的句子。由于我们需要获得整个基本块(即句子)的特征，因此需要在输入 token 序列的最前端放入一个 CLS token，用来预测基本块的类别。按照 Transformer 模型中添加位置嵌入的方法，为输入的 token 添加位置嵌入。基本块编码组件经过训练后，最后一层编码器层的 CLS token 向量记做 $\boldsymbol{H}_B$，可以表示整个基本块的特征。

为了训练基本块编码组件，我们提取实验数据集中所有样本中所有指令序列长度范围为 20～100 的基本块，将基本块通过 Palmtree 模型转换为基本块的特征矩阵 $\boldsymbol{H}$，并标记基本块的类别，如果该基本块来源于挖矿样本，则将其标签设置为 1，如果该基本块来源于非挖矿样本，则将其标签设置为 0。我们将这个数据集称为基本块数据集，该数据集的单个样本记作 $(\boldsymbol{H}_i, label_i)$，其中 $\boldsymbol{H}_i$ 表示第 i 个基本块样本的特征矩阵，$label_i$ 表示其标签。

在训练基本块编码器时，最后一层编码器的 CLS token 向量 $\boldsymbol{H}_B$ 被送入 Softmax 分类器，在误差反向传播和注意力机制的驱使下，$\boldsymbol{H}_B$ 不仅能表示整个基本块的语义信息，还能潜在性地表达基本块是否含有挖矿特征。为了减少时间和空间占用，对于指令序列长度超过 100 的基本块，本文参照 BERT 模型[25]的方法对其进行截断操作，只取前 100 个指令序列送入基本块嵌入模型中。

3.2　函数嵌入模型

本文工作借鉴了 Graphormer[26] 来实现函数嵌入模型，名称 Graphormer 源于 Graph 与 Transformer 的组合，它是 Ying 等提出的一种将 Transformer 神经网络架构应用到图数据结构的一种模型。相较于图神经网络，Graphormer 有着更加优越的性能。与 Transformer 的编码组件结构相似，Graphormer 模型由多个编码器层堆叠组成，它的单个编码器层也由一个自注意力模块和一个前馈神经网络组成，唯一不同的是，Graphormer 模型的自注意力模块在自注意力模块(或

编码器层)的输入之上,原先的位置嵌入编码(Position Embedding)被替换成了中心性编码(Centrality Encoding)。在计算注意力矩阵 $\boldsymbol{A}$ 时,需要将原先的矩阵 $\boldsymbol{A}$ 和两个偏置矩阵相加,分别叫做空间编码(Spatial Encoding)和边编码(Edge Encoding)。除了上述不同之处以外,Graphormer 的其余结构均与 Transformer 的编码组件结构相同。

接下来介绍从 CFG 到标准图数据结构的转换方法。对于 CFG 中的任意一个节点 v_i,本文按照 3.1 节所述基本块嵌入方法,将其转化为一个嵌入向量 $\boldsymbol{x}_i \in R^{1\times d}$,其中 $d=128$ 为基本块嵌入向量的维度。这样即可得到 CFG 的特征矩阵 $\boldsymbol{X}=[\boldsymbol{x}_1^{\mathrm{T}},\boldsymbol{x}_2^{\mathrm{T}},\cdots,\boldsymbol{x}_n^{\mathrm{T}}]^{\mathrm{T}} \in R^{n\times d}$。函数嵌入模型的结构与 Graphormer 模型结构相似,但为了表示 CFG 基本块执行顺序、自连边的特征,我们给 CFG 中的每个基本块添加了一个深度特征,即定义 CFG 起始基本块的深度为 0,其他基本块的深度为起点基本块到该基本块的最短距离。深度特征使得模型能掌握一定 CFG 的节点顺序信息,这是其他图数据结构所不具备的,后续的实验也将证明使用深度特征对模型的性能有所提升。本文对 Graphormer 的中心性编码和边编码进行了改进,具体改动如下:

1)中心性编码与深度编码

式(2)是 Graphormer 原来的中心性编码,其中 $z^-, z^+ \in R^d$ 是将节点 v_i 的入度 $deg^-(v_i)$ 和出度 $deg^+(v_i)$ 进行编码的可学习的嵌入向量。本文在其基础上,将一个深度编码增加到节点特征上作为输入,其中 $\boldsymbol{z} \in R^d$ 是将节点 v_i 的深度 $deep(v_i)$ 进行编码的可学习的嵌入向量,如式(3)所示:

$$\boldsymbol{h}_i^{(0)}=\boldsymbol{x}_i+z^-_{deg^-(v_i)}+z^+_{deg^+(v_i)} \tag{2}$$

$$\boldsymbol{h}_i^{(0)}=\boldsymbol{x}_i+z^-_{deg^-(v_i)}+z^+_{deg^+(v_i)}+z_{deep(v_i)} \tag{3}$$

2)空间编码

空间编码部分与原始 Graphormer 模型方法完全相同,两个节点的距离可以反映出节点的相关程度,因而在注意力机制下,距离越近的两个节点的注意力的值更高。为了满足这一点,Ying 等设计了空间编码,他们用 $\phi(v_i,v_j)$ 表示两个节点 v_i 和 v_j 之间的最短距离。如果两个节点之间没有连接,ϕ 会被设置为特殊值 -1。再设置一个函数 $b_{\phi(v_i,v_j)}$,其为自变量为 $\phi(v_i,v_j)$ 的可学习的标量函数。记 A_{ij} 为自注意力矩阵 $\boldsymbol{A}$ 中的第 i 行、第 j 列的元素,则空间编码会为 A_{ij} 添加一个偏置项 $b_{\phi(v_i,v_j)}$,如式(4)所示:

$$A_{ij}=\frac{(\boldsymbol{h}_i\boldsymbol{W}_Q)(\boldsymbol{h}_j\boldsymbol{W}_K)^{\mathrm{T}}}{\sqrt{d}}+b_{\phi(v_i,v_j)} \tag{4}$$

当 $b_{\phi(v_i,v_j)}$ 被学习成一个递减函数时,模型会使得每个节点对离其近的节点有更多的注意力,对其离的远的节点有更少的注意力,这就是空间编码所取得的效果。

3)边编码

为了将边特征编码到注意力层,需要再给 A_{ij} 添加一个偏置项 c_{ij},即:

$$A_{ij}=\frac{(\boldsymbol{h}_i\boldsymbol{W}_Q)(\boldsymbol{h}_j\boldsymbol{W}_K)^{\mathrm{T}}}{\sqrt{d}}+b_{\phi(v_i,v_j)}+c_{ij} \tag{5}$$

由于 GPU 资源的限制,我们不再对所有节点对 (v_i,v_j) 计算 c_{ij},当且仅当节点对满足以下两个条件时,才计算 c_{ij}。

(1)从 v_i 到 v_j 恰好只有一条有向边时;

(2)$i=j$ 且对于节点 v_i 有一条自连边,这个条件的设置是因为 CFG 存在自连边的可能性。

这样所有的边的特征依然可以被输入模型,但只有和这条边相邻的两个节点(自连边时为一个节点)才会使用这个边特征,这样即可平衡捕获的图中边信息量与模型训练时间和空间。

对于满足条件的节点对 (v_i,v_j),c_{ij} 的表达式如式(6)所示:

$$c_{ij}=\boldsymbol{x}_{en}(\boldsymbol{w}^E)^{\mathrm{T}} \tag{6}$$

其中,$x_{en} \in R^{1\times d_E}$,为节点 v_i 到节点 v_j 的边的特征,d_E 是边特征的维度且 $d_E=3$。因为本文采用三维独热编码来表示一条边为满足分支条件时的跳转、不满足时的跳转或无条件跳转,所以 $w^E \in R^{1\times d_E}$ 是可学习的权值嵌入。

对于不满足条件的节点对 (v_i,v_j),$c_{ij}=0$。通过 3 种编码,CFG 节点特征通过注意力机制在每层编码器层中不断更新。按照 Graphormer 所述,为了汇聚表示全图的特征,需要向图中引入一个特殊节点 $[VNode]$,并使 $[VNode]$ 和所有节点互相用一条虚拟边连接。$\varphi([VNode],v_j)$ 和 $\phi(v_j,[VNode])$ 的值虽然都为 1,但边的连接是虚拟的。为了区别虚拟边和真实边,需要将 $b_{\phi([VNode],v_j)}$ 和 $b_{\phi(v_j,[VNode])}$ 设置为和真实连接边不一样的可学习函数。除此之外,$[VNode]$ 的初始特征是一个可学习的向量 $\boldsymbol{h}_{[VNode]}$,在通过 Graphormer 模型后,$[VNode]$ 在最后一层的特征即可表示整个图的特征,可以用于后续的 CFG 分类任务。

为了训练函数嵌入模型,我们提取实验数据集中所有样本中的 CFG,并将 CFG 中的基本块通过基本块嵌入模型转化为基本块嵌入向量,从而得到 CFG 的特征矩阵 $\boldsymbol{X}_{\mathrm{CFG}}=[\boldsymbol{x}_1^{\mathrm{T}},\boldsymbol{x}_2^{\mathrm{T}},\cdots,\boldsymbol{x}_n^{\mathrm{T}}]^{\mathrm{T}} \in R^{n\times d}$,同时我们求得 CFG 节点深度序列 $\boldsymbol{D}=[d_1,d_2,\cdots,d_n] \in R^{1\times n}$,两点最短距离矩阵 $\boldsymbol{S} \in R^{n\times n}$,两点间是否只存在一条边的边矩阵 $\boldsymbol{P} \in R^{n\times n\times d_E}$。之后按照样本是否属于挖矿恶意软件分别为其打上 1 和 0 的标签,得到的数据集被称为 CFG 数据集,用于训练函数嵌入模型。

在训练函数嵌入模型时,最后一层编码器的 $[VNode]$ 向量被送入 Softmax 分类器,在误差反向传播和注意力机制的驱使下,$[VNode]$ 不仅能表示整个 CFG 的语义和结构信息,还能潜在性地表达基本块是否含有挖矿特征。当函数嵌入模型训练完成后,我们可以将任何 CFG 的特征矩阵 $\boldsymbol{X}$ 输入模型,提取最后一层编码器的 $[VNode]$ 向量,$[VNode]$ 向量即为该 CFG 的嵌入,也就是用该 CFG 表示的函数嵌入向量。

3.3 软件样本嵌入模型

函数调用图 FCG 包含软件样本中所有的函数调用信息,可以表示软件样本。函数调用图的节点为软件样本中的各个函数,有向边表示函数的调用关系。对于 FCG 中的任意一个节点 v_i,按 3.2 节所述函数嵌入模型,将其转化为一个函数嵌入向量 $\boldsymbol{x}_i \in R^{1\times d}$,其中 $d=128$ 为函数嵌入向量的维度,即可得到 FCG 的特征矩阵 $\boldsymbol{X}=[\boldsymbol{x}_1^{\mathrm{T}},\boldsymbol{x}_2^{\mathrm{T}},\cdots,\boldsymbol{x}_n^{\mathrm{T}}]^{\mathrm{T}} \in R^{n\times d}$。

在 FCG 中,函数 v_i 调用函数 v_j 可以用一条从节点 v_i 指向节点 v_j 的有向边表示。实际上,v_i 调用 v_j 即 v_i 执行 v_j 中的代码,故 v_i 中会包含部分 v_j 的特征,所以函数特征的流向应该是从 v_j 流向 v_i,与实际调用方向相反。

我们同样使用 Graphormer 训练软件样本嵌入模型,需要

注意的是，在更新节点特征时，应当遵循节点特征流动的方向。因此，原 FCG 中所有的边的方向都需要变成和原先方向相反的边。由于 FCG 的边没有特征，软件嵌入样本的 Graphormer 模型不再需要边编码，仅需中心性编码和空间编码即可，即直接使用原始的 Graphormer 模型作为软件嵌入模型，软件嵌入最终将用来进行软件样本分类。

4 实验分析

本节通过实验对三层嵌入模型进行性能评估，包括分层性能评估以及整个模型性能与基线机器学习模型的对比。

4.1 实验数据

本文以 2020 大数据安全分析大赛（DataCon2020）[27] 发布的 Windows PE 格式挖矿恶意软件数据集作为基准数据，样本集包括小数据集和大数据集。在比赛中，参赛者可以拿到小数据集及样本对应标签，但只能拿到大数据集的样本而没有标签。因此，参赛者只能使用小数据集来设计挖矿恶意软件检测器，再在大数据集上测试检测器性能，这样的流程类似于现实世界中挖矿恶意软件检测器的开发流程。本文将小数据集称为实验数据集，将大数据集称为真实世界数据集。

我们的实验数据集和真实世界数据集分别包含挖矿恶意软件子集和非挖矿恶意软件子集。其中，挖矿恶意软件子集包含由 DataCon2020 组织者标记为挖矿恶意软件的样本；非挖矿恶意软件子集包含其他样本，包括未标记为挖矿恶意软件的良性样本和恶意样本。需要注意的是，赛事组织者为了禁止参赛者运行样本，将样本 PE 结构中的 MZ 头、PE 头和导入表区域抹去。因此，我们首先固定 PE 格式，然后使用反汇编工具提取特征。

在训练和测试中，挖矿二进制文件被视为正样本，而非挖矿进制文件被视为负样本。实验数据集与真实数据集的规模之比约为 1∶3，两个数据集的正、负样本数量之比为 1∶2。在评估中，本文使用实验数据集训练和验证机器学习分类的性能，并以真实世界数据集作为评估基准，比较所有机器学习模型的性能。这种安排比使用随机抽样将训练数据集从测试数据集中分离出来的工作[20] 更严谨，因为对数据集的随机抽样通常会得到一个相对较小的测试集，从而导致结果与现实世界中的不一致。本文使用常用的标准来评估不同挖矿恶意软件检测器的性能，包括准确率、精确率、召回率、F1 值、AUC 面积。

4.2 实验数据集分析

为了确定合理的基本块、CFG、FCG 最大建模序列长度（或节点数），首先对实验数据集中基本块、CFG 和 FCG 的规模分布进行分析。

使用反汇编工具提取软件样本的 FCG，并绘制 FCG 节点数累计频率分布图，如图 2 所示。可以看到，约有 86% 的 FCG 规模小于 700，覆盖了绝大多数样本，因此我们设定 FCG 的最大建模节点数为 700。通过类似方法确定 CFG 的最大建模节点数，得到其规模的累计频率分布如图 3 所示。可以看出，约有 97% 的 CFG 规模小于 100。为了最大程度地利用样本进行训练，我们将 CFG 的最大建模节点数设置为 700（GPU 资源的最大限制值）。最后统计各样本中所有的基本块的指令序列长度，得到基本块指令序列长度的累计频率分布如图 4 所示。其中约有 98% 的基本块序列长度小于 40。对于一个基本块而言，如果指令序列太短，该基本块难以包含明显的挖矿或非挖矿特征，在训练时容易充当噪声样本或坏样本的角色，从而导致模型的训练被干扰。如果指令序列太长，则会增加训练的时间和空间成本。因此我们选取指令序列长度为 20～100 的基本块作为基本块数据集的数据。

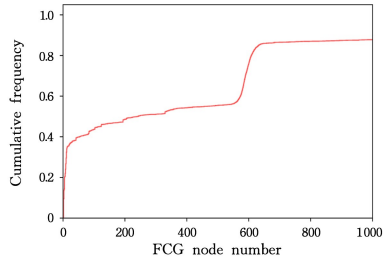

图 2 FCG 节点规模累计频率分布图

Fig. 2 Cumulative frequency distribution of FCG nodes' count

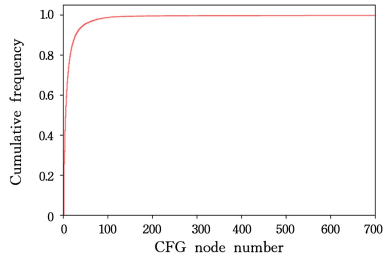

图 3 CFG 节点规模累计频率分布图

Fig. 3 Cumulative frequency distribution of CFG nodes' count

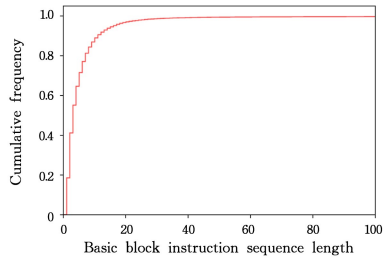

图 4 基本块指令序列长度的累计频率分布图

Fig. 4 Cumulative frequency distribution of the length of instruction sequences in basic blocks

4.3 三层嵌入模型内部评估

三层嵌入模型的每一层都是独立进行训练的，其中软件样本嵌入模型用于最后的挖矿恶意软件检测，而基本块嵌入和函数嵌入模型也需要训练到一定的效果才能投入使用。为了评估基本块嵌入、函数嵌入模型的效果，本节将介绍它们的训练方法并对它们的性能进行测试。

4.3.1 基本块嵌入实验

按照 3.1 节所述方法，我们提取实验数据集中所有样本中所有指令序列长度为 20～100 的基本块，通过 Palmtree 模型将其转换为基本块的特征矩阵 $\boldsymbol{H}$，并标记基本块的类别。如果该基本块来源于挖矿样本，则将其标签设置为 1，否则设置为 0。我们将这个数据集称为基本块数据集，基本块数据集中共有 796941 个样本，其中有 274293 个标签为 1 的样本，522648 个标签为 0 的样本。

使用此数据集训练基本块编码器，随机选取 80%的数据作为训练集，剩下的 20%作为验证集。为防止模型过拟合，我们使用了可变学习率和早停法相结合的方法。如果验证集的损失连续 3 轮不降低，我们将学习率调低到原先的 1/10（初始学习率为 0.001），如果验证集的损失连续 6 轮不降低，则停止训练，保留前 6 轮的最优模型，最优模型在训练集和验证集上的表现如表 1 所列。

表 1 基本块嵌入模型的性能

Table 1 Performance of basic block embedding model

Basic block data set	*Accuracy*/%	*Loss*
Training set(80%)	94.27	0.1620
Validation set(20%)	92.70	0.2132

可以看到，基本块编码器模型在验证集上的准确率为 92.7%。事实上，此实验的目的只是捕获基本块中可能存在的挖矿特征，而并非直接用基本块嵌入进行软件样本分类。一个挖矿软件样本中也可能存在不含挖矿特征的基本块。有的基本块可能本身不含挖矿特征，但其本身可能会经常配合挖矿行为一起出现。因此，我们并不需要将基本块嵌入模型的准确度训练得很高，只要能初步判断一个基本块含有多少挖矿特征，为后面的函数嵌入做准备即可。

4.3.2 函数嵌入实验

我们保存 4.3.1 节训练出的最优的基本块编码器，这样就可以将任何基本块的特征矩阵 $\boldsymbol{H}$ 输入基本块编码器，获取该基本块的嵌入 $\boldsymbol{H}_B$。为了降低时间和空间占用，对于指令序列长度超过 100 的基本块，我们对其进行截断操作，只取前 100 个指令序列。

按照 3.2 节的方法，我们提取实验数据集中所有样本中的 CFG，由于 GPU 资源限制，对于节点数超过 700 的 CFG，我们会丢弃该 CFG 样本，并将 CFG 中的基本块通过基本块嵌入模型转化为基本块嵌入向量，从而得到 CFG 的特征矩阵 $\boldsymbol{X}_{\mathrm{CFG}}=[x_1^{\mathrm{T}},x_2^{\mathrm{T}},\cdots x_n^{\mathrm{T}}]^{\mathrm{T}}\in R^{n\times d}$。之后我们标记该 CFG 的类别，如果该 CFG 来源于挖矿样本，则将其标签设置为 1；如果该 CFG 来源于非挖矿样本，则将其标签设置为 0。我们将这个数据集称为 CFG 数据集，并用 CFG 数据集训练函数嵌入模型。数据经过提取后，CFG 数据集中共计有 1 227 513 个样本，其中有 235 690 个标签为 1 的样本，991 823 个标签为 0 的样本。

使用此数据集训练函数嵌入模型，随机选取 80%的数据作为训练集，剩下的 20%作为验证集，防止过拟合的措施与基本块嵌入实验的一致。最优模型在训练集和验证集上的表现如表 3 所列。可以看到，最优函数嵌入模型在验证集上的准确率为 97.73%。虽然准确率已经很高，但此实验的目的只是捕获 CFG 中可能存在的挖矿特征，而并非直接用函数嵌入进行软件样本分类。一个挖矿软件样本中也可能存在不含有挖矿特征的 CFG。同理，我们并不需要将函数嵌入模型的准确度训练得很高，它只需要能判断一个 CFG 含有多少挖矿特征，为最终的软件样本嵌入做准备即可。

表 3 函数嵌入模型的性能

Table 3 Performance of function embedding model

CFG data set	*Accuracy*/%	*Loss*
Training set(80%)	97.94	0.0662
Validation set(20%)	97.73	0.0750

4.4 三层嵌入模型与基线模型的对比

本节将通过实验对比三层嵌入模型和第 2 节列出的基线机器学习模型的挖矿恶意软件检测性能。三层嵌入模型的软件样本嵌入模型的性能即为三层嵌入模型最终的挖矿恶意软件检测性能，对于其余的基线机器学习模型，我们按照文献[15-17]的方案对它们进行复现，从而对比所有模型的性能。

为了对比三层嵌入模型和其他基线机器学习模型的性能，所有的模型都在实验数据集和真实世界数据集上进行了评估。其中操作码特征＋LSTM 模型[16]和 CFG 特征＋MAGIC 模型[17]按照原论文中的方式进行复现，而对于 Malconv 模型[15]，其使用的特征为原始字节，考虑到实验中模型计算速度和所需 GPU 显存大小，我们提取了前 20 万字节的二进制文件作为特征向量，用于分类器训练。长度不足的二进制文件被填充为零字节，或者将长度超出阈值的部分截断。在文献[15]中，样本长度被设置为 100 万字节，利用深度学习模型对大样本的拟合能力构造长序列字节码的分类模型。实验结果表明，20 万的参数量在计算硬件性能与分类效果之间取得了平衡，在可接受的硬件性能开销范围内，实现了相对其他方法更好的分类效果。

为了减少由实验样本较小造成的偶然性，我们使用随机交叉验证来测试不同模型的性能。实验数据集进行了 3 次重排序，每次进行 5 次交叉验证。最后，将所有的实验结果取平均值作为相应分类模型的性能值，如表 3 所列。从表中可以看出，三层模型在实验数据集上的性能大幅领先于其他基线机器学习模型，三层嵌入模型的各项训练指标都领先于其他基线模型，除了 MAGIC 模型[16]的精确率和三层嵌入模型持平，其余都达到了 100%，尤其在召回率方面更是领先次优的 MAGIC 模型 2.49%，说明三层嵌入模型在实验数据集上的性能最优。但由于实验的训练集和验证集都来自于样本量相对较小的实验数据集，模型的泛化性如何并不能确定，因此我们还需要将模型在真实数据集上进行测量来模拟模型在真实世界中的性能。

表 3 实验数据集上各机器学习模型的性能对比

Table 3 Performance comparison of machine learning models on experimental datasets

Ranking	Feature	Model	*Accuracy*	*Precision*	*Recall*	*F1-score*	*AUC*
1	Basic block＋CFG＋FCG	Three-layer-embedding Model	**0.9978**	**1.0000**	**0.9822**	**0.9899**	**0.9904**
2	CFG	MAGIC[16]	0.9859	**1.0000**	0.9573	0.9782	0.9787
3	Raw bytes	Malconv[14]	0.9761	0.9949	0.9330	0.9628	0.9653
4	Opcode	LSTM[15]	0.9569	0.9573	0.9195	0.9364	0.9479

不同于实验数据集上的使用随机交叉验证来测试性能，真实数据集上的实验采取另一个独立的流程来训练和测试模型。首先在完整的实验数据集上训练基于不同特征的挖矿恶意软件检测器，所有检测器的结构与前面的实验相同，然后将训练好的模型在真实世界的数据集上进行测试，实验结果如表4所列。

表4 真实世界数据集上各机器学习模型的性能对比

Table 4 Performance comparison of machine learning models on real-world datasets

Ranking	Feature	Model	*Accuracy*	*Precision*	*Recall*	*F1-score*	*AUC*
1	Basic block+CFG+FCG	Three-layer-embedding Model	**0.9778**	**0.9970**	**0.9428**	**0.9692**	**0.9706**
2	CFG	MAGIC[16]	0.9574	0.9931	0.8717	0.9285	0.9344
3	Raw bytes	Malconv[14]	0.9504	0.9719	0.8768	0.9219	0.9320
4	Opcode	LSTM[15]	0.9144	0.9453	0.7967	0.8647	0.8861

表4的各项指标结果相较于表3有明显的降低，这符合我们的预想，因为当模型从实验小数据集（训练集和验证集）泛化到真实世界数据集（测试集）时，性能会有所降低。但三层嵌入模型依旧保持了最优的总体性能，在准确率方面领先第二名的MAGIC模型3.04%；在召回率方面，甚至领先MAGIC模型7.11%之多，这证明了三层嵌入模型在挖矿恶意软件的识别能力出众，领先于其他基线模型；在精确率方面，三层嵌入模型也达到了最高的99.70%，意味着误报率仅有0.30%。MAGIC模型的总体性能居第二位，各项指标都明显低于三层嵌入模型，但我们也可以看出，MAGIC模型的召回率较低，低于第三名的Malconv模型，说明MAGIC在真实世界数据集上有着召回率偏低的缺点。而使用操作码作为特征的LSTM模型，相较于表3中的其他模型各项指标性能降低最为明显，说明该模型的泛化性不佳。总的来说，三层嵌入模型的泛化能力很强，在真实世界数据集上的性能大幅领先于其他基线机器学习模型。

4.5 CFG深度特征和边编码对三层嵌入模型性能的影响

在3.2节中，我们提到CFG深度特征使得模型能掌握一定CFG的节点顺序信息，这是其他图数据结构所不具备的，此外我们还将CFG的3种不同类型的边进行了编码。本节将证明使用深度特征和边编码对模型的某些性能指标有所提升。分别去掉函数嵌入模型中的深度特征和边编码，并重新训练三层嵌入模型，将实验结果在真实世界数据集上的性能和原始的三层嵌入模型进行对比，实验结果如表5所列。可以明显看到，相较于去掉CFG深度特征的三层嵌入模型，原始三层嵌入模型的准确率高出0.99%，精确率高出3.74%，召回率高出0.53%，充分证明了CFG深度特征对模型性能的提升作用。

表5 真实世界数据集上去掉CFG特征的三层嵌入模型的性能对比

Table 5 Performance comparison of three-layer-embedding models without CFG features on real-world datasets

Ranking	Model	*Accuracy*	*Precision*	*Recall*	*F1-score*	*AUC*
1	Original Three-layer-embedding Model	**0.9778**	**0.9970**	0.9428	**0.9692**	**0.9706**
2	Three-layer-embedding Model (without CFG edge encoding feature)	0.9772	0.9751	**0.9630**	0.9690	0.9743
3	Three-layer-embedding Model (without CFG depth feature)	0.9679	0.9596	0.9375	0.9484	0.9584

相较于去掉CFG边编码的三层嵌入模型，原始三层嵌入模型的准确率基本保持不变，精确率高出2.19%，召回率却降低了2.02%，说明去掉CFG边编码特征的三层嵌入模型在挖矿恶意软件的召回率上表现非常优秀，达到了96.30%，但其缺点是精确率低于原始三层嵌入模型。在实际应用中应当根据实际性能指标的需求选择是否去掉CFG边编码特征。

结束语 本文提出了一种基于静态特征的三层嵌入挖矿恶意软件检测模型，该模型由基本块嵌入模型、函数嵌入模型和软件嵌入模型组成，可以自下而上地逐层提取出样本基本块、CFG和FCG的特征，最终得到一个软件嵌入向量用于挖矿恶意软件的检测。三层嵌入模型分层读取了软件样本的语义信息和跳转调用信息，这也是该模型性能大幅领先其他模型的主要原因。通过实验发现，三层嵌入模型在模拟真实世界数据集的实验环境下大幅优于其他机器学习模型的性能，证明其性能良好，泛化性强，能在真实世界应用的场景下实现精准的挖矿恶意软件检测能力。此外，三层嵌入模型中底层的基本块嵌入模型和中间层的函数嵌入模型也可以单独使用。例如基本块嵌入向量可以用作异常基本块的侦测，函数嵌入向量可以用来完成函数相似性度量[28]、二进制代码相似性度量[29-30]，以及代码克隆[31]检测等任务。

参 考 文 献

[1] TEKINER E,ACAR A,ULUAGAC A S,et al. SoK:Cryptojacking Malware[C]// 2021 IEEE European Symposium on Security and Privacy(EuroS&P). IEEE,2021:120-139.

[2] 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. 2019:73-86.

[3] 360TS. Cryptominer,winstarnssmminer,has made a fortune by brutally hijacking computers[EB/OL]. [2021-12-31]. https://blog.360totalsecurity.com/en/cryptominer-winstarnssmminer-made-fortune-brutally-hijacking-computer.

[4] TAHIR R,HUZAIFA M,DAS A,et al. Mining on someone else's dime:Mitigating covert mining operations in clouds and enterprises[C]// International Symposium on Research in Attacks,Intrusions,and Defenses. Cham:Springer,2017:287-310.

[5] ESENTIRE I. Cryptocurrency craze drives 1,500% increase in coin-mining malware[EB/OL]. [2021-12-31]. https://www.esentire.com/news-releases/2018s-cryptocurrency-craze-helps-drive-1500-percent-increase-in-coinmining-malware.

[6] GRIFFTHS J. Coinminers target vulnerable users as bitcoin hits all-time high, [EB/OL]. [2021-12-31]. https://www.avira.com/en/blog/coinminers-target-vulnerable-users-as-bitcoin-hits-all-time-high/.

[7] YAN G. Be sensitive to your errors: Chaining neyman-pearson criteria for automated malware classification[C]// Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. 2015: 121-132.

[8] YOUSEFI-AZAR M, VARADHARAJAN V, HAMEY L, et al. Autoencoder-based feature learning for cyber security applications[C]// 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017: 3854-3861.

[9] KEBEDE T M, DJANEYE-BOUNDJOU O, NARAYANAN B N, et al. Classification of malware programs using autoencoders based deep learning architecture and its application to the microsoft malware classification challenge (big 2015) dataset[C]// 2017 IEEE National Aerospace and Electronics Conference (NAECON). IEEE, 2017: 70-75.

[10] HASSEN M, CARVALHO M M, CHAN P K. Malware classification using static analysis based features[C]// 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2017: 1-7.

[11] DREW J, MOORE T, HAHSLER M. Polymorphic malware detection using sequence classification methods[C]// 2016 IEEE Security and Privacy Workshops (SPW). IEEE, 2016: 81-87.

[12] WANG Z W, LIU G Q, HAN X H, et al. Survey on Machine-learning-based Malware Identification Research[J]. Journal of Chinese Computer Systems, 2022, 43(12): 2628-2637.

[13] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet Classification with Deep Convolutional Neural Networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[14] DING Y X, ZHU S Y. Malware detection based on deep learning algorithm[J]. Neural Computing and Applications, 2019, 31(2): 461-472.

[15] RAFF E, BARKER J, SYLVESTER J, et al. Malware detection by eating a whole exe[C]// Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[16] YAZDINEJAD A, HADDADPAJOUH H, DEHGHANTANHA A, et al. Cryptocurrency malware hunting: A deep recurrent neural network approach[J]. Applied Soft Computing, 2020, 96: 106630.

[17] YAN J, YAN G, JIN D. Classifying malware represented as control flow graphs using deep graph convolutional neural network [C]// 2019 49th annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2019: 52-63.

[18] LE Q, BOYDELL O, MAC NAMEE B, et al. Deep learning at the shallow end: Malware classification for non-domain experts [J]. Digital Investigation, 2018, 26: S118-S126.

[19] AZEEZ N A, ODUFUWA O E, MISRA S, et al. Windows PE malware detection using ensemble learning[J]. Informatics, 2021, 8(1): 1-22.

[20] YU Z, CAO R, TANG Q, et al. Order matters: semantic-aware neural networks for binary code similarity detection[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2020: 1145-1152.

[21] AHMADI M, ULYANOV D, SEMENOV S, et al. Novel feature extraction, selection and fusion for effective malware family classification[C]// Proceedings of the sixth ACM Conference on Data and Application Security and Privacy. 2016: 183-194.

[22] XU X, LIU C, FENG Q, et al. Neural network-based graph embedding for cross-platform binary code similarity detection[C]// Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 363-376.

[23] HASSEN M, CHAN P K. Scalable function call graph-based malware classification[C]// Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. 2017: 239-248.

[24] "pre-trained PalmTree model" [EB/OL]. [2022-03-31]. https://drive.google.com/file/d/1yC3M-kVTFWql6hCgM_QCbKtc1PbdVdvp/view/.

[25] KIPF T N, WELLING M. Semi-supervised classification with graph convolutional networks[J]. arXiv: 1609.02907, 2016.

[26] YING C, CAI T, LUO S, et al. Do Transformers Really Perform Badly for Graph Representation? [J]. arXiv: 2106.05234, 2021.

[27] "DataCon" [EB/OL]. [2021-12-31]. https://datacon.qianxin.com/opendata/maliciouscode.

[28] MASSARELLI L, LUNA G A D, PETRONI F, et al. Safe: Self-attentive function embeddings for binary similarity[C]// International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Cham: Springer, 2019: 309-329.

[29] XU X, LIU C, FENG Q, et al. Neural network-based graph embedding for cross-platform binary code similarity detection[C]// Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 363-376.

[30] ZUO F, LI X, YOUNG P, et al. Neural machine translation inspired binary code similarity comparison beyond function pairs [J]. arXiv: 1808.04706, 2018.

[31] DING S H H, FUNG B C M, CHARLAND P. Asm2vec: Boosting static representation robustness for binary clone search against code obfuscation and compiler optimization[C]// 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019: 472-489.

FU Jianming, born in 1969, Ph. D, professor, Ph. D supervisor, is a member of CCF (No. 07112S). His main research interests include system security and mobile security.

(责任编辑:何杨)