



组会文献分享： API2Vec:用于恶意软件检测的 API序列的学习表示

关键字：恶意软件检测 · API · 深度学习

汇报人：顾羽桥

指导老师：李振源

目录

C O N T E N T

01. 文章主旨

02. 威胁模型与假设

03. 模型设计

04. 模型评估

05. 相关工作

06. 总结与讨论

01

PART ONE

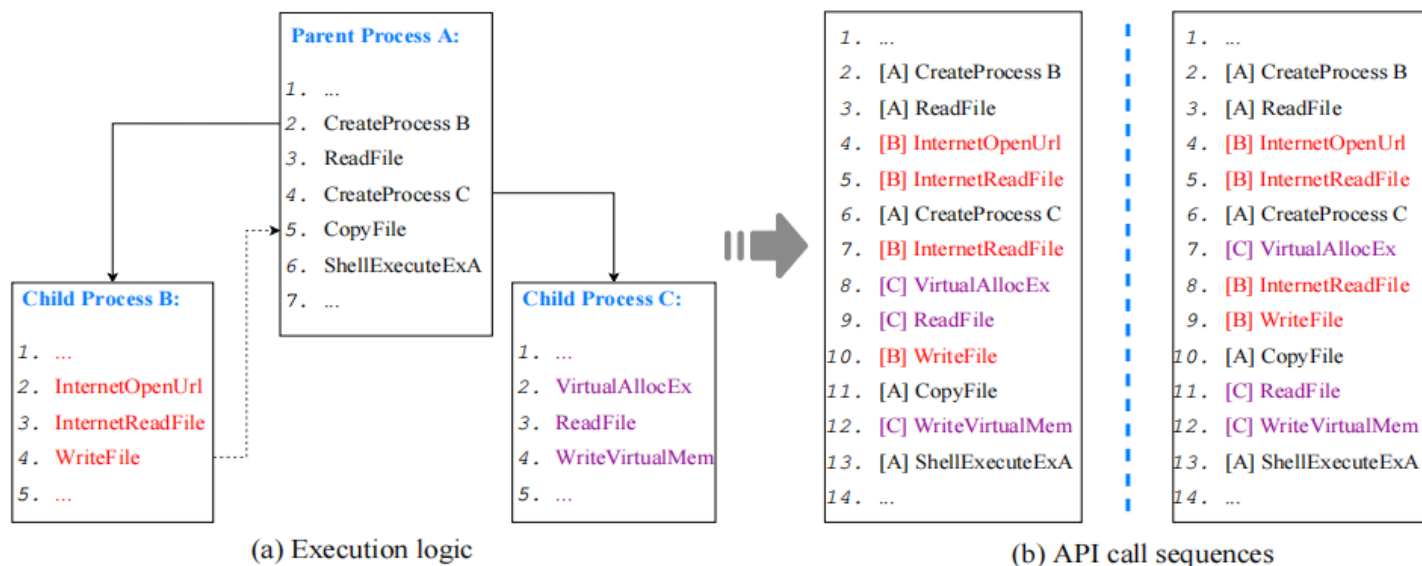
文章主旨

本文研究的主要问题

基于图的API序列嵌入方法进行多线程恶意软件的检测

目前将深度学习用于API序列检测的方法多集中在原始序列的操作，但是这种方法不能有效应对多线程恶意软件。这篇论文提出了API2Vec方法。该方法首先构建图模型表示原始序列，设计时态进程图（TPG）表示进程间的行为，设计时态API图（TAG）表示进程内的行为。然后通过启发式随机游走算法生成可以捕获细粒度恶意软件行为的路径。在用Doc2Vec模型预训练这些路径后，就能够将其用于恶意软件检测。

研究动机



- 恶意软件可以将API序列调用分发到不同的进程，因此来自不同进程的API调用顺序可以与其他进程交错，导致API调用顺序具有任意性，这就是API交错问题。直接从原始序列学习特征的检测方法不准确就是因为多进程机制容易混淆。

本文主要贡献

基于图的API序列表示

通过时间进程图和时间API图对进程间和进程内行为进行建模

面向行为和覆盖范围的启发式随机游走算法

该算法可以提取细粒度的行为并且能够用更少的路径挖掘到更多的内在关联

广泛评估模型效果

本文实验基于真实的恶意软件数据集进行评估，并且通过与其他模型对比确定检测效果

开源代码

本文实验数据和代码上传到github:[Coming98/API2Vec: API2Vec: Learning Representations of API Sequences for Malware Detection \(github.com\)](https://github.com/Coming98/API2Vec)

02

PART TWO

威胁模型与假设

威胁模型与假设

本文主要关注Windows平台PE可执行恶意软件的检测，具体包括蠕虫、病毒、木马以及流氓软件等。这篇论文采用Cuckoo沙箱记录程序运行时的活动，并且假设沙箱可以成功跟踪设计恶意行为的API调用，同时恶意软件也不会被沙箱和操作系统中的保护程序阻断运行。这个假设是动态恶意软件检测的标准模型。

03



PART THREE



模型设计

时间图相关概念

逻辑时间

如果一个API a比API b在序列中领先，表示a会比b先被调用。逻辑时间从0开始，单调递增+1，直至最后一个API调用

时间API图

TAG可以形式化的用 $\langle V, E, A \rangle$ 表示，V集合中每一个元素表示API，E集合中每一个元素表示API间的。如果API a在API b之后立即执行，则存在一条从a指向b的有向边，每条边使用逻辑时间作为属性进行区分

时间进程图

TPG可以用 $\langle PV, PE, PA \rangle$ 形式化表示。PV表示TPG的顶点每个顶点都是一个TAG，PE表示TAG之间的有向边，PA用逻辑时间对表示每条边的属性。注意，有2种类型的边，即父进程fork子进程和非父子进程间的调用

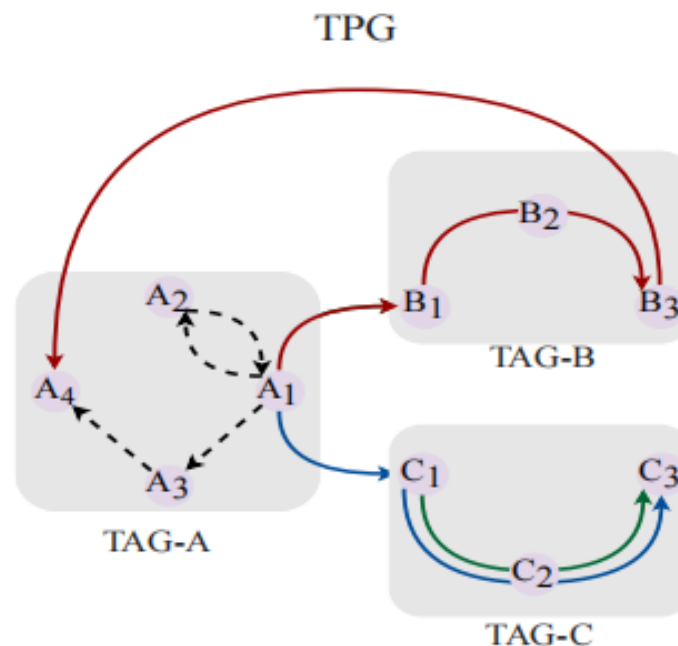
建图方法

TAG图的构建方式

- 标记序列中每个API的逻辑时间
- 对同一进程的API进行分组，每组的第一个API直接添加到TAG中
- 下一次调用将检查关联的API是否已在TAG图中，如果没有，则把API作为新节点添加。并在原来的API和新节点之间连边，并分配关联的逻辑时间，该步骤反复执行直到整个序列遍历完成

TPG图的构建方式

- 首先构建TAG并将每个TAG表示为TPG的顶点
- 如果2个TAG共享相邻的API调用，则表示为有父子关系或子子关系的边
- 最后，对于序列中任意两个相邻但是跨TAG的任意两个API调用



TAG内部的随机游走算法规则

TAG内部的随机游走目的在于挖掘进程内的行为。

节点选择

- 边数越多，优先级越高
- 时间跨度越小，优先级越高
- 访问次数越多，优先级越低
- 备选节点的逻辑时间应该小于tc

边选择

- 时间跨度较小的边应该被选择

结束条件

- 没有可用的候选节点
- 行走路径长度达到设定阈值

返回TPG机制

R2P有助于获得进程间的行为

- 时间跨度越大，优先级越高。较大的值表示两个关联的API调用被其他进程的API打断
- 访问次数越多，优先级越高。一些关键API可能被重复调用

TPG内部的随机游走算法规则

当需要R2P和TAG内部已经遍历完后，启动TPG随机游走。

- 优先选择子TAG。进程与子进程间的交互通常比兄弟进程间的交互多
- 时间跨度越小，优先级越高。
- 边数越多，优先级越高。

起始顶点选择

假设终点为 pv_d ，就需要在 pv_d 中选择起点，假设 pv_d 与 pv_s 中 v_s 相邻的顶点 v_i 是候选点，若这两个点之间不存在边，则需要在这二者间增加一条虚拟边，其属性为 $\{t_s, t_i\}$ ，其中 t_s 表示当前的逻辑时间， t_i 表示作为 v_i 的最小逻辑时间

结束条件

如果存在TPG的逻辑时间小于 t_c ，算法应该停止

学习任务

API嵌入

使用Doc2Vec模型从大量路径语料中学习路径和API的表示，从而通过向量化的方式嵌入来表示路径和API

恶意软件检测

API2Vec使用预训练的路径嵌入对API调用序列进行编码，并且因为一个序列产生多个编码，所以最终得到的编码由提取的所有路径的64维嵌入的平均值表示，然后在编码序列上设置3个检测模型，分别为kNN，SVM，RF

04

PART FOUR

模型评估

模型评估相关概念

概念漂移:

当新样本随着时间的推移出现时, 就会产生概念漂移, 导致训练模型错误预测新样本

对抗性攻击:

生成API序列片段, 并将其插入到原始恶意序列 m 中生成对抗序列 m^* , 其目的是最小化RNN预测模型中 m^* 的预测概率。对抗攻击通过插入API调用来改变其语法和语义

组件的增益:

API2Vec的关键组件是遵循三类启发式规则的图模型和随机游走算法

- (1) Sequence。直接对原始序列执行Doc2Vec算法
- (2) Graph。配置了朴素随机游走算法
- (3) G^* (Graph+Affinity) 。扩展了关于边数的规则
- (4) G^* +Coverage。扩展了关于整体覆盖率的规则。
- (5) G^* +behaviour。扩展了面向行为的规则。
- (6) API2Vec。使用了所有规则。

恶意软件检测能力图表

Table 2: F1-score on multi-process programs.

#Proc	2	3	4	5	≥6
Word2Vec	97.04%	98.44%	94.97%	98.39%	96.92%
DeepWalk	96.57%	98.44%	95.13%	97.56%	96.97%
Node2VecB	96.78%	98.57%	94.82%	96.72%	96.97%
Node2VecD	96.78%	98.27%	94.67%	97.56%	96.18%
Frequency	96.83%	98.48%	95.19%	98.39%	96.06%
Markov	68.08%	79.61%	70.73%	80.37%	88.52%
BiLSTM	97.91%	99.39%	95.89%	99.20%	99.22%
vBiLSTM	96.75%	98.53%	89.02%	97.56%	96.92%
API-Bert	91.82%	97.34%	90.52%	99.20%	96.24%
API-SIF	96.68%	98.66%	90.80%	100.00%	97.60%
DMalNet	96.05%	98.15%	91.83%	99.20%	96.88%
CruParamer	95.01%	98.36%	91.59%	96.72%	98.46%
API2Vec	99.57%	99.87%	99.85%	100.00%	99.22%

Table 3: F1-score on samples from varying years.

Model	2019	2020	Model	2019	2020
Word2Vec	86.46%	80.04%	vBiLSTM	89.08%	83.35%
DeepWalk	81.82%	78.66%	API-Bert	84.25%	83.95%
Node2VecD	82.64%	77.62%	API-SIF	87.65%	85.94%
Node2VecB	82.50%	77.58%	DMalNet	89.64%	86.78%
Frequency	80.51%	88.95%	CruParamer	86.02%	91.43%
Markov	77.12%	51.82	API2Vec	98.59%	99.13%
BiLSTM	86.02%	91.43%			

Table 4: F1-score on samples from varying types.

Type	virus	backdoor	worm	grayware	downloader
Word2Vec	88.45%	95.87%	93.39%	91.28%	97.03%
DeppWalk	77.72%	95.17%	93.39%	87.50%	96.50%
Node2VecB	77.90%	95.51%	93.54%	90.65%	96.40%
Node2VecD	77.95%	94.99%	93.63%	87.16%	96.54%
Frequency	91.79%	93.07%	81.83%	85.65%	96.11%
Markov	74.66%	71.83%	75.31%	67.65%	69.25%
BiLSTM	86.23%	75.55%	96.46%	92.81%	96.95%
vBiLSTM	90.98%	96.31%	93.24%	90.13%	95.46%
API-Bert	84.71%	93.29%	91.87%	81.49%	92.73%
API-SIF	88.24%	96.91%	88.89%	87.83%	95.59%
DMalNet	82.46%	96.12%	87.33%	85.09%	95.03%
CruParamer	90.98%	96.31%	93.24%	90.13%	95.46%
API2Vec	99.24%	99.49%	99.25%	99.15%	99.41%

恶意软件检测能力图表

Table 5: Performance on detecting adversarial samples.

Model	rate	Model	rate	Model	rate
Word2Vec	62.49%	Frequency	70.79%	API-Bert	49.00%
DeepWalk	67.13%	Markov	70.06%	API-SIF	67.63%
Node2VecB	66.47%	BiLSTM	55.16%	DMalNet	71.77%
Node2VecD	66.54%	vBiLSTM	59.90%	CruParamer	75.86%
API2Vec	97.38%				

Table 6: Performance of various learning models.

Model	Word2Vec			Doc2Vec		
	precision	recall	f1-score	precision	recall	f1-score
SVM	87.33%	89.08%	88.20%	99.16%	99.25%	99.20%
RF	96.75%	92.08%	94.36%	99.47%	99.04%	99.26%
<i>k</i> -NN	94.70%	94.29%	94.49%	99.23%	99.84%	99.54%

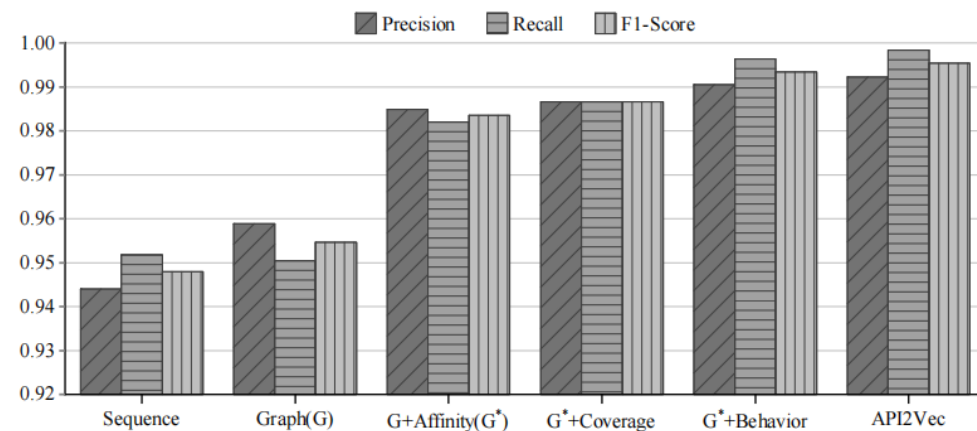


Figure 5: Performance of proposed components.

05

PART FIVE

相关工作

相关工作

- NtMalDetect:使用n-gram和TF-IDF生成系统调用的特征, 并使用SVM进行恶意软件检测
- API-Bert:使用Fasttext和Bert学习API嵌入
- Generic Black-Box End-to-End Attack Against State of the Art API Call Based Malware Classifiers:将原始API序列分为几个部分, 每个部分分别采用RNN模型学习
- An approach for detection and family classification of malware based on behavioral analysis: 将恶意软件序列转换为特征向量, 主要包括API频率, 然后利用特征向量上的信息熵对恶意软件进行分类
- Improving the Detection of Malware Behaviour Using Simplified Data Dependent API Call Graph : 从原始数据依赖中提取数据依赖关系并构建API图, 采用最长公共子序列算法来进行图匹配
- Improving the Detection of Malware Behaviour Using Simplified Data Dependent API Call Graph : 通过解析Android API文档构建API关系图, 并设计知识图谱嵌入算法学习API的表示, 将其用于恶意软件

06


PART SIX

总结与讨论

总结

API2Vec整体由图构建（包括TAG图和TPG图）、路径生成、API嵌入和恶意软件检测四个部分组成。该方法因为采用图模型，因此能够更简洁的捕捉进程间和进程内的行为，系统的鲁棒性也更好。

模型缺陷



相邻API之间存在大量重复良性的路径，因此能够逃避检测

提取到的API序列较短，
比如恶意行为提前终止

许多进程有大量重复的操作，这会加强其中的良性语义

讨论

1. Cuckoo沙箱？还有哪些可以使用的工具？如果有些恶意软件专门针对沙箱进行逃逸？甚至通过沙箱危害物理机环境？
2. 是否也可以参考溯源图中查询图的模式，确定一些常用恶意软件API调用的序列或类似的模式，使得时间进程图（TPG）和时间API图（TAG）也可以通过定义成图论的问题？
3. LLM在恶意软件检测中可以发挥的作用？
4. 针对本文的方法，可以改进的点？



THANKS

谢谢观看，恳请批评指正

汇报人：顾羽桥

指导老师：李振源

