(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 114969734 A
(43)申请公布日 2022.08.30

(21)申请号 202210526872.6

(22)申请日 2022.05.16

(71)申请人 北京航空航天大学
地址 100191 北京市海淀区学院路37号

(72)发明人 李博 刘振龙 刘陈 刘旭东

(74)专利代理机构 北京中创阳光知识产权代理有限责任公司 11003
专利代理师 尹振启

(51)Int.Cl.
G06F 21/56(2013.01)
G06F 21/53(2013.01)
G06F 9/448(2018.01)
G06N 3/04(2006.01)
G06N 3/08(2006.01)

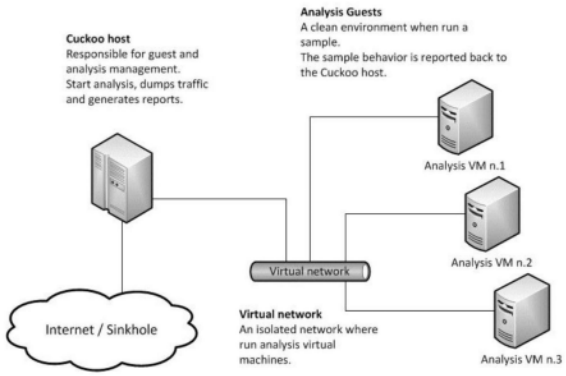
权利要求书2页 说明书11页 附图2页

(54)发明名称

一种基于API调用序列的勒索病毒变种检测方法

(57)摘要

本发明通过信息安全领域的方法,实现了一种基于API调用序列的勒索病毒变种检测方法。首首先设置基于API调用序列的勒索病毒家族分类技术单元,通过对于输入的勒索病毒特征,部署Cuckoo沙箱,构建勒索病毒数据集;收集大量API调用序列构成语料库,使用word2vec进行预训练;选取API调用序列作为学习特征,进行预处理;训练检测模型并评价后获得可用的模型,进而获取分类结果;在对所有动态行为分类的基础上,对于分类结果采用基于Graphviz和Neo4j的勒索病毒攻击流程可视化技术单元,以基于Graphviz的攻击流程可视化流程为主,基于Neo4j的攻击流程可视化流程为辅的方式,输出对于病毒的分类结果。在勒索病毒家族分类任务上展现出了较好的效果。



CN 114969734 A

1.一种基于API调用序列的勒索病毒变种检测方法,其特征在于:首先设置基于API调用序列的勒索病毒家族分类技术单元,通过输入勒索病毒样本,部署Cuckoo沙箱,构建勒索病毒数据集;收集大量API调用序列构成语料库,使用word2vec进行预训练;选取API调用序列作为学习特征,进行预处理;训练检测模型并评价后获得可用的模型,进而获取分类结果;

之后,在对所有动态行为分为进程、系统、Shell代码检测、内存、文件、注册表、网络七类的基础上,对于分类结果采用基于Graphviz和Neo4j的勒索病毒攻击流程可视化技术单元,以基于Graphviz的攻击流程可视化流程为主,基于Neo4j的攻击流程可视化流程为辅的方式,使用Graphviz的可视化结果查看攻击流程的总体,使用Neo4j的可视化结果查看攻击流程的特定行为或细节,并最终,输出对于勒索病毒家族分类结果的可视化视图。

2.如权利要求1所述的一种基于API调用序列的勒索病毒变种检测方法,其特征在于:所述构建勒索病毒数据集的具体方法为:采用两种方式为数据来源;

具体地,一种方法为:调用现成的行为分析报告或API调用序列数据集,收集行为分析报告和API调用序列数据集,然后利用Python脚本提取API调用序列和勒索病毒家族名称作为标签;

另一种方法为:搭建一个Cukoo host和一个Analysis Guest模式的Cuckoo沙箱作为勒索病毒分析环境,以开源网站中获取原始的恶意样本批量放入勒索病毒分析环境运行,批量处理样本,汇总现成的行为分析报告后,然后提取API调用序列。

3.如权利要求2所述的一种基于API调用序列的勒索病毒变种检测方法,其特征在于:所述预训练方法为:首先需要构建语料库,其中API调用序列不需要有具体的标签,所以来源是恶意代码数据集、无标签的勒索病毒数据集、有标签的勒索病毒数据集;

预训练模型选择包括跳元模型和连续词袋模型的word2vec,然后选用跳元模型,使用中心词预测文本序列中生成其周围的单词;

通过最大化似然函数来学习模型参数,

$$-\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log P\left(w^{(t+j)}|w^{(t)}\right)$$

其中API调用文本序列长度为T,上下文窗口大小为m,在时间t处的词元为$w^{(t)}$,并采用下采样技术,降低高频词的重要性,提高低频词的重要性,然后进行小批量处理。

4.如权利要求3所述的一种基于API调用序列的勒索病毒变种检测方法,其特征在于:所述训练检测模型方法具体为:构建基于API调用序列的检测模型,模型输入是API调用序列,输出是类别,TextCNN+Attention模型输入API调用序列,经过嵌入层转化为二维词向量矩阵;卷积层设置不同尺寸的卷积核,其中第二维的长度为词向量长度,激活函数使用ReLU函数,加入非线性因素,经过卷积层会生成3个不同长度的一维向量;池化层,采用最大池化策略,生成三个数值,其中每个数值是原向量中的最大元素;池化层生成的三个数值经过拼接生成1*3的向量,然后经过Attention来聚焦设定的信息;最后经过全连接层输出对应类别。

5.如权利要求4所述的一种基于API调用序列的勒索病毒变种检测方法,其特征在于:模型的所述评价方法为,采用加权平均的方法计算多分类整体精确率:$P_w=\sum_i^N P_i * \frac{Cnt_i}{Total}$、

召回率：$R_w = \sum_i^N R_i * \frac{Cnt_i}{Total}$、基于精确率和召回率的调和平均：$F_{1w} = \sum_i^N F_{1i} * \frac{Cnt_i}{Total}$，其中，N代表类别总数，i代表样本对应类别，Total代表样本总数，Cnt_i代表类别i的数量，P_i、R_i、F_{1i}分别代表类别i的精确率、召回率、精确率和召回率的调和平均。

6.如权利要求5所述的一种基于API调用序列的勒索病毒变种检测方法，其特征在于：所述基于Graphviz的攻击流程可视化流程的实现方式为：

步骤1：通过pip安装Python库graphviz；

步骤2：安装Graphviz，并确保包含dot可执行文件的目录在系统路径上，安装过程中需要添加PATH；

步骤3：为了显示直观，突出重点行为，设置端点形状、端点颜色、边的颜色、字体颜色；

步骤4：使用minidom解析器打开XML文档，获取所有字段action_list，进而解析该字段的必要属性，最后存入Action对象列表；

步骤5：初始化端点，如动态分析开始、动态分析异常结束、动态分析结束、样本文件、开始、vasstarter.exe等端点；

步骤6：遍历Action对象列表，获取端点，利用辅助数组进行端点去重；

步骤7：初始化边，如动态分析过程、关闭主机、进程开始执行等边；

步骤8：遍历Action对象列表，获取边，然后连接上面得到的端点；

步骤9：然后利用Python调用API进行图的导出，如png、jpg、svg、pdf等格式；

步骤10：如果边、端点的数量过多，会影响绘图效率，采用多线程编程来缓解该问题。

7.如权利要求6所述的一种基于API调用序列的勒索病毒变种检测方法，其特征在于：所述基于Neo4j的攻击流程可视化流程具体方法为：

步骤1：从Oracle官方网站下载Java SE JDK并进行安装，Neo4j是基于Java的图形数据库，运行Neo4j需要启动JVM进程，因此必须安装JAVA SE的JDK；

步骤2：在图数据库Neo4j官网下载Neo4j并安装，安装过程需要配置环境变量；

步骤3：通过控制台启动Neo4j程序，此时Neo4j服务已经在本地部署；

步骤4：通过pip安装Python库py2neo，然后连接Neo4j图数据库；

步骤5：使用Graphviz中相同的方法获取Action对象列表，并构建端点和边；

步骤6：在已有节点上批量创建关系，以提高作图效率；

步骤7：打开浏览器，访问http://localhost:7474，得到可视化结果。

一种基于API调用序列的勒索病毒变种检测方法

技术领域

[0001] 本发明涉及信息安全技术领域，尤其涉及一种基于API调用序列的勒索病毒变种检测方法。

背景技术

[0002] 近几年，由于勒索病毒攻击性强，攻击者已经开始使用它们作为网络攻击的有效方法。根据咨询机构埃森哲调查显示，2021年上半年，全球网络威胁活动较去年增长125％。聚焦在勒索攻击领域，针对的行业主要是保险、消费品和服务、电信，占比分别为23％、17％、16％，三者总计占比达到56％。2020年下半年相比，2021年上半年，由勒索软件攻击引起的数据泄露事件增长了24％。勒索攻击不仅带来了代价高昂的服务中断，还直接威胁政治安全、经济安全、科技安全等各个方面。

[0003] 勒索病毒是一种特殊的恶意软件，它会感染计算机并限制用户访问，直到支付赎金才能解锁。攻击者可以通过多态等技术迅速产生大量勒索病毒变种，而且此种方法成本低。尽管已经开发了防火墙、反病毒程序和自动分析程序等安全机制来对抗这种威胁，但这些传统机制收效甚微，无法保护存储在本地或云存储资源中的宝贵资产。

[0004] 勒索病毒分析技术是分析勒索病毒目的和功能的过程，主要分为静态分析和动态分析两种方法。静态分析是不执行勒索病毒的自动分析，提取MD5、操作码等静态特征，进而对其行为进行推断。但是攻击者通常使用加壳、代码混淆等技术，使得静态分析难以应对。动态分析是通过在虚拟环境中执行收集的勒索病毒样本，对恶意行为或勒索病毒的风险进行运行时监控和分析。一般来说，动态分析根据使用的特征和应用的技术有两种方式。首先，通过使用的特征进行动态分析，利用诸如API调用的频率或顺序，编译的十六进制代码，程序执行路径和其他的信息作为特征。其次，通过应用技术分析，利用序列对齐和数据挖掘或机器学习，对收集的特征数据进行分析。但是许多勒索病毒变种会识别沙箱而延迟执行，或者执行期间大量调用无关良性API，使得动态分析变得困难。因此，现有检测方法缺乏有效的勒索病毒变种检测方法。

[0005] 随着深度学习的迅速发展，研究者开始将其应用到安全领域，并取得了不错的效果。如果学习特征选择API调用序列，并将其看作是一个文本，其中的单个API函数看作是一个词元，那么勒索病毒分类问题就可以转换为NLP领域的文本分类问题。因此，文本分类中的一些经典模型就可以应用到勒索病毒家族分类任务中。

[0006] 近年来，勒索病毒攻击行为隐蔽性强且危害显著，异较快且易传播，攻击路径和目标多元化发展，受勒索攻击领域更加宽泛。此外，由于攻击者对勒索病毒变种转化技术的滥用，变种数量和质量激增，传统反病毒技术很难检测到此类攻击。因此，亟待一种良好的勒索病毒检测技术，迅速确定样本所属的家族或相近家族，进而采取针对性地分析，编写解密程序或安全补丁，从而尽可能地减少损失。

[0007] 静态分析技术难以应对多态、变种、加壳、压缩等技术，动态技术难以应对延迟运行、添加大量无关的良性API调用等。而且两种方法主要对已知勒索病毒样本或样本库具有

较好的检测效果,不适用于勒索病毒变种。

发明内容

[0008] 为此,本发明首先提出一种基于API调用序列的勒索病毒变种检测方法,首先设置基于API调用序列的勒索病毒家族分类技术单元,通过对于输入的勒索病毒样本,部署Cuckoo沙箱,构建勒索病毒数据集;收集大量API调用序列构成语料库,使用word2vec进行预训练;选取API调用序列作为学习特征,进行预处理;训练检测模型并评价后获得可用的模型,进而获取分类结果;

[0009] 之后,在对所有动态行为分为进程、系统、Shell代码检测、内存、文件、注册表、网络七类的基础上,对于分类结果采用基于Graphviz和Neo4j的勒索病毒攻击流程可视化技术单元,以基于Graphviz的攻击流程可视化流程为主,基于Neo4j的攻击流程可视化流程为辅的方式,使用Graphviz的可视化结果查看攻击流程的总体,使用Neo4j的可视化结果查看攻击流程的特定行为或细节,并最终,输出对于勒索病毒家族分类结果的可视化视图。

[0010] 所述构建勒索病毒数据集的具体方法为:采用两种方式为数据来源;

[0011] 具体地,一种方法为:调用现成的行为分析报告或API调用序列数据集,收集行为分析报告和API调用序列数据集,然后利用Python脚本提取API调用序列和勒索病毒家族名称作为标签;

[0012] 另一种方法为:搭建一个Cukoo host和一个Analysis Guest模式的Cuckoo沙箱作为勒索病毒分析环境,以开源网站中获取原始的恶意样本批量放入勒索病毒分析环境运行,批量处理样本,汇总现成的行为分析报告后,然后提取API调用序列。

[0013] 所述预训练方法为:首先需要构建语料库,其中API调用序列不需要有具体的标签,所以来源是恶意代码数据集、无标签的勒索病毒数据集、有标签的勒索病毒数据集;

[0014] 预训练模型选择包括跳元模型和连续词袋模型的word2vec,然后选用跳元模型,使用中心词预测文本序列中生成其周围的单词;

[0015] 通过最大化似然函数来学习模型参数,

[0016] $$-\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log P\left(w^{(t+j)}|w^{(t)}\right)$$

[0017] 其中API调用文本序列长度为T,上下文窗口大小为m,在时间t处的词元为$w^{(t)}$,并采用下采样技术,降低高频词的重要性,提高低频词的重要性,然后进行小批量处理。

[0018] 所述训练检测模型方法具体为:构建基于API调用序列的检测模型,模型输入是API调用序列,输出是类别,TextCNN+Attention模型输入API调用序列,经过嵌入层转化为二维词向量矩阵;卷积层设置不同尺寸的卷积核,其中第二维的长度为词向量长度,激活函数使用ReLU函数,加入非线性因素,经过卷积层会生成3个不同长度的一维向量;池化层,采用最大池化策略,生成三个数值,其中每个数值是原向量中的最大元素;池化层生成的三个数值经过拼接生成1*3的向量,然后经过Attention来聚焦设定的信息;最后经过全连接层输出对应类别。

[0019] 模型的所述评价方法为,采用加权平均的方法计算多分类整体精确率:

[0020] $P_w=\sum_i^N P_i * \frac{Cnt_i}{Total}$、召回率:$R_w=\sum_i^N R_i * \frac{Cnt_i}{Total}$、基于精确率和召回率的调和平

均：$F_{1w} = \sum_i^N F_{1i} * \frac{Cnt_i}{Total}$，其中，N代表类别总数，i代表样本对应类别，Total代表样本总数，Cnt_i代表类别i的数量，P_i、R_i、F_{1i}分别代表类别i的精确率、召回率、精确率和召回率的调和平均。

[0021] 所述基于Graphviz的攻击流程可视化流程的实现方式为：

[0022] 步骤1：通过pip安装Python库graphviz；

[0023] 步骤2：安装Graphviz，并确保包含dot可执行文件的目录在系统路径上，安装过程中需要添加PATH；

[0024] 步骤3：为了显示直观，突出重点行为，设置端点形状、端点颜色、边的颜色、字体颜色；

[0025] 步骤4：使用minidom解析器打开XML文档，获取所有字段action_list，进而解析该字段的必要属性，最后存入Action对象列表；

[0026] 步骤5：初始化端点，如动态分析开始、动态分析异常结束、动态分析结束、样本文件、开始、vasstarter.exe等端点；

[0027] 步骤6：遍历Action对象列表，获取端点，利用辅助数组进行端点去重；

[0028] 步骤7：初始化边，如动态分析过程、关闭主机、进程开始执行等边；

[0029] 步骤8：遍历Action对象列表，获取边，然后连接上面得到的端点；

[0030] 步骤9：然后利用Python调用API进行图的导出，如png、jpg、svg、pdf等格式；

[0031] 步骤10：如果边、端点的数量过多，会影响绘图效率，采用多线程编程来缓解该问题。

[0032] 所述基于Neo4j的攻击流程可视化流程具体方法为：

[0033] 步骤1：从Oracle官方网站下载Java SE JDK并进行安装，Neo4j是基于Java的图形数据库，运行Neo4j需要启动JVM进程，因此必须安装JAVA SE的JDK；

[0034] 步骤2：在图数据库Neo4j官网下载Neo4j并安装，安装过程需要配置环境变量；

[0035] 步骤3：通过控制台启动Neo4j程序，此时Neo4j服务已经在本地部署；

[0036] 步骤4：通过pip安装Python库py2neo，然后连接Neo4j图数据库；

[0037] 步骤5：使用Graphviz中相同的方法获取Action对象列表，并构建端点和边；

[0038] 步骤6：在已有节点上批量创建关系，以提高作图效率；

[0039] 步骤7：打开浏览器，访问http://localhost:7474，就可以看到可视化结果。

[0040] 本发明所要实现的技术效果在于：

[0041] 实现一种搜索方法，并使方法具备如下特性：

[0042] 充分利用API调用序列以实现对勒索病毒变种的检测，引入了word2vec+TextCNN+Attention的检测方式，在API语料库上进行预训练，然后利用TextCNN和注意力机制进行分类学习，该方法在勒索病毒家族分类任务上展现出了较好的效果。

附图说明

[0043] 图1Cuckoo主要架构；

[0044] 图2跳元模型示例；

[0045] 图3API调用序列长度分布图；

[0046] 图4word2vec+TextCNN+Attention模型结构；

[0047] 图5基于Graphviz的行为可视化示例；

[0048] 图6基于Neo4j的行为可视化示例；

具体实施方式

[0049] 以下是本发明的优选实施例并结合附图，对本发明的技术方案作进一步的描述，但本发明并不限于此实施例。

[0050] 本发明提出了一种基于API调用序列的勒索病毒变种检测方法。

[0051] 静态分析技术难以应对多态、变种、加壳、压缩等技术，动态技术难以应对延迟运行、添加大量无关的良性API调用等。而且两种方法主要对已知勒索病毒样本或样本库具有较好的检测效果，不适用于勒索病毒变种。本专利从动态分析出发，选择API调用序列，去除重复子序列后作为学习特征，然后利用深度学习分类模型进行训练，最终获取分类结果。接下来，为了提高检测结果的可解释性，对勒索病毒变种的攻击流程进行可视化。基于此，本专利由“基于API调用序列的勒索病毒家族分类技术”和“基于Graphviz和Neo4j的勒索病毒攻击流程可视化技术”两大部分组成。

[0052] 基于API调用序列的勒索病毒家族分类技术

[0053] 本节旨在解决勒索病毒家族分类问题，部署Cuckoo沙箱，构建勒索病毒数据集；收集大量API调用序列构成语料库，使用word2vec进行预训练；选取A PI调用序列作为学习特征，进行预处理；训练检测模型，获取分类结果。

[0054] 勒索病毒数据集构建

[0055] 查阅相关文献得，目前可用的勒索病毒数据集匮乏，开源社区提供的数据集老旧，不会定时更新；一些竞赛、开源网站大多提供的是恶意代码数据集，没有细化为勒索病毒家族。因此，需要自主构建勒索病毒数据集。

[0056] 数据来源主要有两种：其一，现成的行为分析报告或API调用序列数据集，如ACT-KingKong数据集、MMCC微软恶意软件分类挑战数据集、Ember数据集等；其二，原始的恶意样本，从MalShare、VirusShare、Exploit Database、Viru sTotal等开源网站中获取，批量放入勒索病毒分析环境运行，进而获取API调用序列。

[0057] 数据来源方式一：收集行为分析报告和API调用序列数据集，然后利用Py thon脚本提取API调用序列和标签(勒索病毒家族名称)。

[0058] 数据来源方式二：选取Cuckoo沙箱作为勒索病毒分析环境，Cuckoo是一个开源的恶意软件自动分析系统，其主要架构如图1所示。

[0059] 搭建一个Cukoo host和一个Analysis Guest模式的Cuckoo沙箱，批量处理样本，汇总现成的行为分析报告后，然后提取API调用序列。

[0060] 通过上述两种数据来源进行收集与处理勒索病毒样本，构建了一个包含13887条数据的勒索病毒数据集。数据集包含8个类别，良性和7个勒索病毒家族，具体分布情况见表1。

[0061] 表1勒索病毒数据集分布情况表

[0062]

类别	Benign	Phobos	Sodinokibi	WannaCry	Ryuk	Avaddon	Stop	Globelmposter
数量	4978	1487	515	4289	100	820	1196	502

[0063] 预训练

[0064] 由于独热编码或TF-IDF等向量化的方法无法表达API函数之间的相似程度，故采取预训练的方式获取词向量。

[0065] 首先需要构建语料库，其中API调用序列不需要有具体的标签，所以来源可以是恶意代码数据集、无标签的勒索病毒数据集、有标签的勒索病毒数据集等。这样一来，收集到的无效数据也得到了有效利用。

[0066] 预训练模型选择word2vec(包括跳元模型和连续词袋模型)，然后选用跳元模型，使用中心词预测文本序列中生成其周围的单词。

[0067] 跳元模型参数是词元的中心词向量和上下文词向量。在训练中，我们通过最大化似然函数(即极大似然估计)来学习模型参数。这相当于最小化公式1的损失函数，其中文本(API调用)序列长度为T，上下文窗口大小为m，在时间t处的词元为$w^{(t)}$。

[0068] 公式1跳元模型损失函数:

[0069] $$-\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log P\left(w^{(t+j)}|w^{(t)}\right)$$

[0070] 为了降低计算复杂度，通常进行近似训练，如负采样和分层softmax。文本数据中通常有高频词，这些词经常出现在上下文窗口，但是提供的有用信息却很少，所以采用下采样技术，降低高频词的重要性，提高低频词的重要性。然后进行小批量处理，以便在训练过程中迭代加载。

[0071] 经过预训练，会将词元映射为向量，也就是词向量，这些词向量包含不同词元间的相似信息。

[0072] 预处理

[0073] API调用序列中的API函数具有前后逻辑关系，而且API函数具有特殊含义，表2是勒索病毒常用的API函数调用及其功能描述。

[0074] 表2勒索病毒API函数功能表

[0075]

API 函数	功能描述
NtQueryKey	检索存储的缓冲
ZwOpenKey	例程打开现有的注册表项
AuxKlibInitialize	例程初始化辅助 Kernel-Mode 库
WRITE_PORT_UCHAR	例程将字节写入指定的端口地址
WdmlibIoGetAffinityInterrup	函数获取中断对象的组关联
InterlockedAnd	宏以原子方式计算按位 "与" 运算
ExUuidCreate	例程将 UUID (GUID) 结构初始化为新生成的值
NtClose	例程关闭对象句柄

[0076] API调用序列长短不一，多则成千上万，少则几十条，为了API序列数据可用，能够作为检测模型的输入，需要进行预处理。

[0077] 一方面，软件本身有大量的重复API调用；另一方面，攻击者为了增加安全人员的

分析难度，刻意增加大量无用的API调用。有文献提出并实验证明：API调用序列去除重复子序列，不影响API序列间的相似性计算。另外，对API调用序列进行压缩，会减少计算时间。综上，对API调用序列去重是必要的。采用去除连续API函数调用的方法，比如原序列为“QWEERRRT”，去重后是“QWERT”。

[0078] 去重后统计API调用序列长度可得，该数据集的序列长度主要分布在0到200，具体见图3。因此，选取200作为序列长度上限，序列长度大于200的API调用序列进行截断操作，小于200的API调用序列进行补零操作。

[0079] 检测模型

[0080] 卷积神经网络(CNN)是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现。典型的卷积神经网络由卷积层、池化层和全连接层三部分构成。卷积层负责提取图像中的局部特征；池化层用来大幅降低参数量级(降维)；全连接层类似传统神经网络的部分，用来输出想要的结果。CNN已经得到了广泛的应用，比如：人脸识别、自动驾驶、安防等很多领域。TextCNN是在文本分类上的应用，核心思想是捕捉局部特征，对于文本来说，局部特征就是由若干单词组成的滑动窗口，类似于N-gram。卷积神经网络的优势在于能够自动地对N-gram特征进行组合和筛选，获得不同抽象层次的语义信息。

[0081] 注意力机制(Attention)是在计算能力有限的情况下，将计算资源分配给更重要的任务，同时解决信息超载问题的一种资源分配方案。在神经网络学习中，一般而言模型的参数越多则模型的表达能力越强，模型所存储的信息量也越大，但这会带来信息过载的问题。那么通过引入注意力机制，在众多的输入信息中聚焦于对当前任务更为关键的信息，降低对其他信息的关注度，甚至过滤掉无关信息，就可以解决信息过载问题，并提高任务处理的效率和准确性。

[0082] 借鉴文本分类思路，构建了基于API调用序列的检测模型，模型输入是API调用序列，输出是类别。

[0083] TextCNN+Attention模型结构见图4，输入API调用序列，经过嵌入层转化为二维词向量矩阵；卷积层，设置不同尺寸的卷积核，比如3、4、5，其中第二维的长度为词向量长度，激活函数使用ReLU函数，加入非线性因素，经过卷积层会生成3个不同长度的一维向量；池化层，采用最大池化策略，生成三个数值，其中每个数值是原向量中的最大元素；池化层生成的三个数值经过拼接生成1*3的向量，然后经过Attention来聚焦设定的信息；最后经过全连接层输出对应类别。

[0084] 评价模型

[0085] 评价一个模型的优劣，通常有准确率(Accuracy)、精确率(Precision)、召回率(Recall)、F1等指标。

[0086] 准确率(Accuracy)：所有预测正确的样本在样本总量中的占比，计算公式如下：

[0087] 公式2准确率计算公式

[0088] $$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

[0089] 精确率(Precision)：也称为查准率，准确率是模型只找到相关目标的能力，计算公式如下：

[0090] 公式3精确率计算公式

[0091] $$\text{Precision} = \frac{TP}{TP + FP}$$

[0092] 召回率(Recall):也称为查全率,是模型找到所有相关目标的能力,即模型给出的预测结果最多能覆盖多少真实目标,计算公式如下:

[0093] 公式4召回率计算公式

[0094] $$\text{Recall} = \frac{TP}{TP + FN}$$

[0095] F1:基于精确率和召回率的调和平均,计算公式如下:

[0096] 公式5F1计算公式

[0097] $$F_1 = \frac{2 * Recall * Precison}{Recall + Precison}$$

[0098] 其中,TP(True Positive):真实值为真,预测值为真;FP(False Posi tive):真实值为假,预测值为真;FN(False Negative):真实值为真,预测值为假;TN(True Negative):真实值为假,预测值为假。

[0099] 上述计算公式适用于二分类问题,多分类问题需要进一步处理,可以看作N(数据集类别数)个二分类任务计算上述指标。本专利采用了weighted-aver age(加权平均)的方法计算多分类整体Precision、Recall、F1,计算方式如下:

[0100] 公式6多分类精确率计算公式

[0101] $$P_w = \sum_{i}^{N} P_i * \frac{Cnt_i}{Total}$$

[0102] 公式7多分类召回率计算公式

[0103] $$R_w = \sum_{i}^{N} R_i * \frac{Cnt_i}{Total}$$

[0104] 公式8多分类F1计算公式

[0105] $$F_{1w} = \sum_{i}^{N} F_{1i} * \frac{Cnt_i}{Total}$$

[0106] 其中,N代表类别总数,i代表样本对应类别,Total代表样本总数,Cnt_i代表类别i的数量,P_i、R_i、F_{1i}分别代表类别i的Precison、Recall、F1-sco re。

[0107] 基于Graphviz和Neo4j的勒索病毒攻击流程可视化技术

[0108] 勒索病毒分析报告中的行为部分,仅仅是简单的罗列,不直观,短时间内很难发现重点行为。所以需要对勒索病毒的攻击流程进行可视化,可视化工作还增加了检测模型的可解释性。

[0109] 下面针对ACT-KingKong数据集继续实现,该数据集有xml文件组成。每个xml文件对应一个勒索病毒(或恶意代码)样本,其中包含了一系列动态行为,如创建进程、删改注册表键值、加载模块等。

[0110] 经过综合分析,将所有动态行为分为进程、系统、Shell代码检测、内存、文件、注册

表、网络七类，具体分类情况如表3所示。

[0111] 表3勒索病毒动态行为分类表

[0112]

类别	代表行为
进程	设置崩溃进程环境
	启动 CUI 程序
系统	监控系统消息
	控制远程注册表服务
Shell 代码检测	内嵌可执行模块
	内嵌可疑代码
内存	创建勒索软件服务
	加载模块
文件	覆盖任务管理器
	创建或修改系统文件
注册表	修改注册表系统键值
	修改文件夹隐藏选项
网络	创建到本机的网络连接
	接收本机网络数据包

[0113] 如果将每个进程看作一个端点，把动态行为看作连接两个端点之间的边，因此勒索病毒攻击行为可视化实质上是有向图构建。

[0114] 基于Graphviz的行为可视化

[0115] Graphviz(图形可视化软件的简称)是一个由AT&T Labs Research发起的开源工具包，用于绘制文件扩展名为“gv”的DOT语言脚本中指定的图形，还为软件应用程序提供了使用这些工具的库。Python库graphviz为图形绘制软件Graphviz提供了一个简单的纯python接口，利用Python编程可以高效地进行图形绘制；该模块提供了两个类:Graph和Digraph，它们分别以DOT语言为无向图和有向图创建图描述，具有相同的API。

[0116] 基于Graphviz的攻击流程可视化，具体实现步骤如下：

[0117] Step 1:通过pip安装Python库graphviz；

[0118] Step 2:安装Graphviz，并确保包含dot可执行文件的目录在系统路径上，安装过程中需要添加PATH；

[0119] Step 3:为了显示直观，突出重点行为，设置端点形状、端点颜色、边的颜色、字体颜色；

[0120] Step 4:使用minidom解析器打开XML文档，获取所有字段action_list，进而解析该字段的必要属性，最后存入Action对象列表；

[0121] Step 5:初始化端点，如动态分析开始、动态分析异常结束、动态分析结束、样本文件、开始、vasstarter.exe等端点；

[0122] Step 6:遍历Action对象列表，获取端点，利用辅助数组进行端点去重；

[0123] Step 7:初始化边，如动态分析过程、关闭主机、进程开始执行等边；

[0124] Step 8:遍历Action对象列表,获取边,然后连接上面得到的端点;

[0125] Step 9:然后利用Python调用API进行图的导出,如png、jpg、svg、pd f等格式;

[0126] Step 10:如果边、端点的数量过多,会影响绘图效率,采用多线程编程来缓解该问题。

[0127] 最终实现效果如图5所示,通过示例图可以清晰看出行为执行的前后逻辑关系,不同的端点具有不同颜色,有利于快速地定位关键行为,而且可以使得检测模型具有可解释性。然而,当勒索病毒样本包含大量端点、边时,生成的图庞大且复杂,许多次要的行为会影响关键行为的发现。

[0128] 基于图数据库Neo4j的行为可视化

[0129] Neo4j是Neo4j,Inc.开发的图形数据库管理系统,开发人员将其描述为兼容acid的事务数据库,具有原生图形存储和处理功能,基于Java实现。Pytho n库py2neo提供了Python操控Neo4J的相应API,简单、安全且高效。

[0130] 使用Neo4j对勒索病毒攻击流程可视化的步骤如下:

[0131] Step 1:从Oracle官方网站下载Java SE JDK并进行安装,Neo4j是基于Java的图形数据库,运行Neo4j需要启动JVM进程,因此必须安装JAVA SE的JDK。

[0132] Step 2:在图数据库Neo4j官网下载Neo4j并安装,安装过程需要配置环境变量;

[0133] Step 3:通过控制台启动Neo4j程序,此时Neo4j服务已经在本地部署;

[0134] Step 4:通过pip安装Python库py2neo,然后连接Neo4j图数据库;

[0135] Step 5:使用Graphviz中相同的方法获取Action对象列表,并构建端点和边;

[0136] Step 6:在已有节点上批量创建关系,以提高作图效率;

[0137] Step 7:打开浏览器,访问http://localhost:7474,就可以看到可视化结果。

[0138] 最终实现效果如图6所示,由于端点和边是浮动的,行为之间的前后逻辑关系不易发现。但是,由于图数据库Neo4j可以进行增删查改操作,可操作性强且灵活。可以通过删除次要端点,快速定位关键端点和关键边;也可以查询与指定端点相关的其他端点。

[0139] 综上所述,基于Graphviz的行为可视化和基于图数据库Neo4j的行为可视化各有优缺点,因此以Graphviz为主,Neo4j为辅的方案是一个不错的可视化方式。

[0140] 本发明使用实验室的集群进行训练,其操作系统类型为CentOS Linux rel ease 7.6.1810(Core),处理器为Intel(R)Xeon(R)Silver 4214R,内存大小为256GB,具体的硬件配置如表4所示。此外,本发明的开发语言为Python,深度学习框架为pytorch,具体的软件配置如表5所示。本发明的检测模型使用TextCNN,结合注意力机制,具体模型参数如表6所示。

[0141] 表4实验硬件配置

[0142]

名称	参数
操作系统	CentOS Linux release 7.6.1810 (Core)
CPU	Intel(R) Xeon(R) Silver 4214R
内存	256G
硬盘	3TB

[0143] 表5实验软件配置

[0144]

名称	信息
开发语言	python（3.9.12）
深度学习框架	pytorch（1.10.0+cu113）
Python 相关库	numpy（科学计算）
	scikit-learn（机器学习库）
	matplotlib（绘制图表）
	graphviz（Graphviz 的 Python 实现）
	sqlalchemy（提供 SQL 工具包和 ORM）
	time（处理时间和日期）

[0145] 表6检测模型参数表

[0146]

内容	参数
Embedding layer	嵌入层数：269 嵌入层维数：100
Convolutional layer	卷积层数：3 个（核尺寸为 3、4、5） 输入管道数量：200 输出管道数量：100 激活函数：ReLU
Pooling layer	一维的自适应平均池化 AdaptiveAvgPool1d
Attention layer	DropOut：0.5
Softmax layer	输入特征数：300 输出特征数：8 bias：True
其他参数	学习率：0.01 学习率衰减的乘积因子：0.5 Batch 大小：64 Epoch 数量：100

[0147] 本发明使用的勒索病毒数据集是自主构建的，数据来源包括ACT-KingKong数据集、开源社区、竞赛数据集、恶意代码网站等，由13887个样本组成，由七个勒索病毒家族样本和良性样本组成，按照8:2的比例划分为训练集和测试集。

[0148] 为了验证本发明检测模型的优越性(word2vec+TextCNN+Attention，简称WTA)，在构建的勒索病毒数据集上进行对比实验。同样选取随机森林(RF)、多层感知机(MLP)、TextCNN、长短期记忆网络(LSTM)、word2vec结合TextCNN(简称WT)等方法进行对比，其中RF算法选取API编号进行向量化；多层感知机采用API编号作为学习特征，网络结构包括两个

全连接层,其中激活函数使用ReLU;TextCNN模型采用独热编码进行向量化;LSTM模型使用word2vec进行预训练,采用双向长短期记忆网络;WT模型使用word2vec进行预训练,分类模型使用TextCNN,与本发明模型的不同是未使用注意力机制。

[0149] 表7对比实验结果

[0150]

分类算法	Accuracy	Precision	Recall	F1	耗时(分钟)
RF	0.806	0.805	0.806	0.797	0.042
MLP	0.679	0.701	0.679	0.678	0.473
TextCNN	0.829	0.829	0.829	0.827	18.491
LSTM	0.852	0.853	0.852	0.851	56.378
WT	0.848	0.846	0.848	0.845	17.128
WTA(本发明)	0.855	0.862	0.855	0.853	18.778

[0151] 实验结果如表7所示,可以得出以下结论:RF机器学习算法在各项指标上具有不错的表现,特别是耗时最短;对比RF和MLP两种算法,与随机森林算法相比,多层感知机并不占任何优势;对比TextCNN和WT模型两种模型,发现使用预训练模型后准确率提升约2%;对比WT与WTA两种模型,发现使用注意力机制后准确率提升约1%;对比LSTM和WTA两种模型,发现两种模型在各个评价指标上相差不大,但是LSTM模型训练时间更长;综上,本发明模型WTA在各项指标上均有突出的表现,而且训练耗时可以接受。

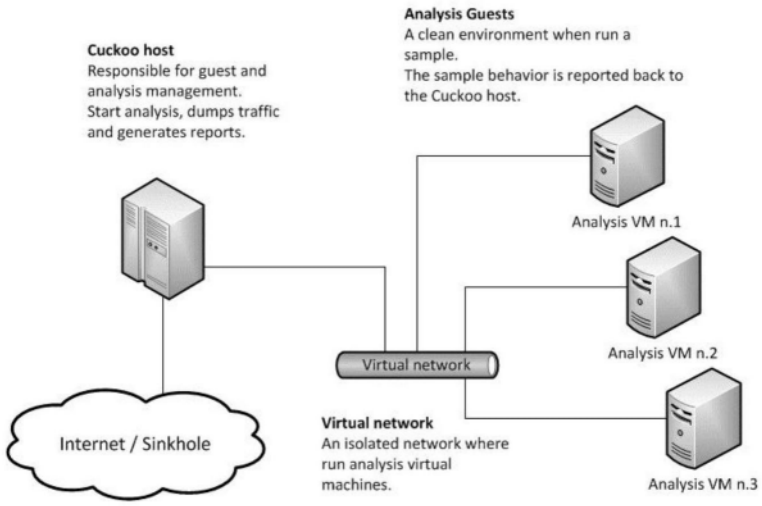

Cuckoo host
Responsible for guest and analysis management.
Start analysis, dumps traffic and generates reports.
Analysis Guests
A clean environment when run a sample.
The sample behavior is reported back to the Cuckoo host.
Analysis VM n.1
Analysis VM n.2
Virtual network
Internet / Sinkhole
Virtual network
An isolated network where run analysis virtual machines.
Analysis VM n.3

图1

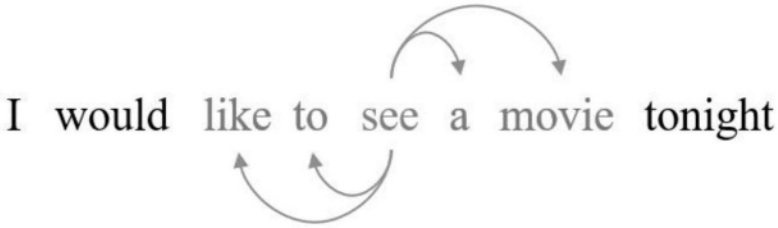

I would like to see a movie tonight

图2

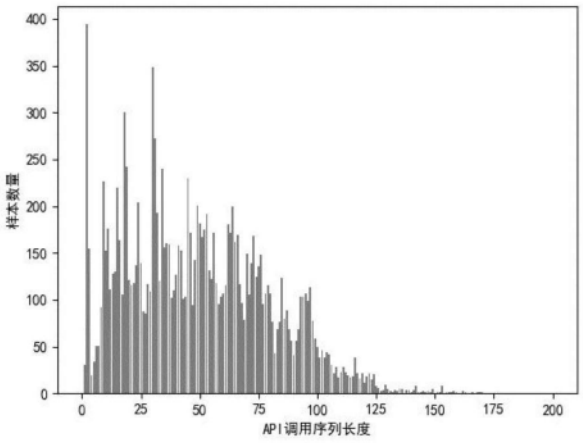

样本数量
API调用序列长度
0
50
100
150
200
250
300
350
400
0
25
50
75
100
125
150
175
200

图3

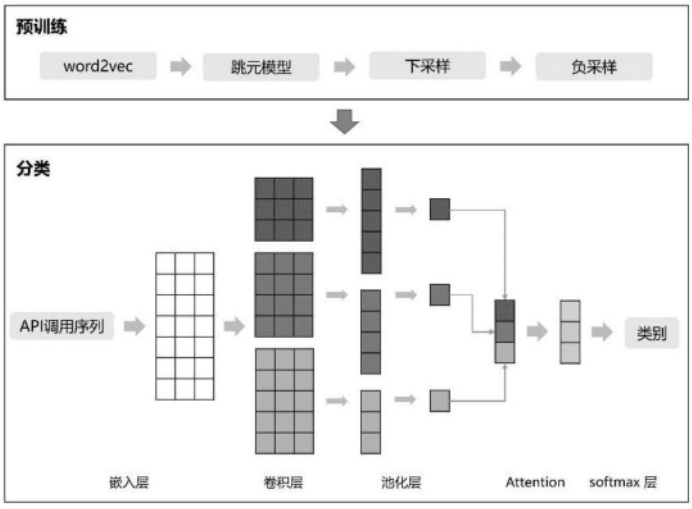

预训练
word2vec
跳元模型
下采样
负采样
分类
API调用序列
类别
嵌入层
卷积层
池化层
Attention
softmax 层

图4

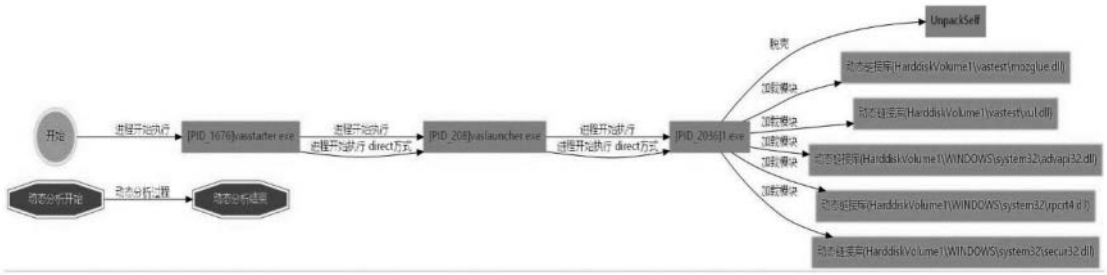

开始
进程开始执行
[PID_1676]vasstarter.exe
进程开始执行
进程开始执行 direct方式
[PID_208]vaslauncher.exe
进程开始执行
进程开始执行 direct方式
[PID_2036]1.exe
脱壳
加载模块
Unpack5e8
动态分析开始
动态分析进程

图5

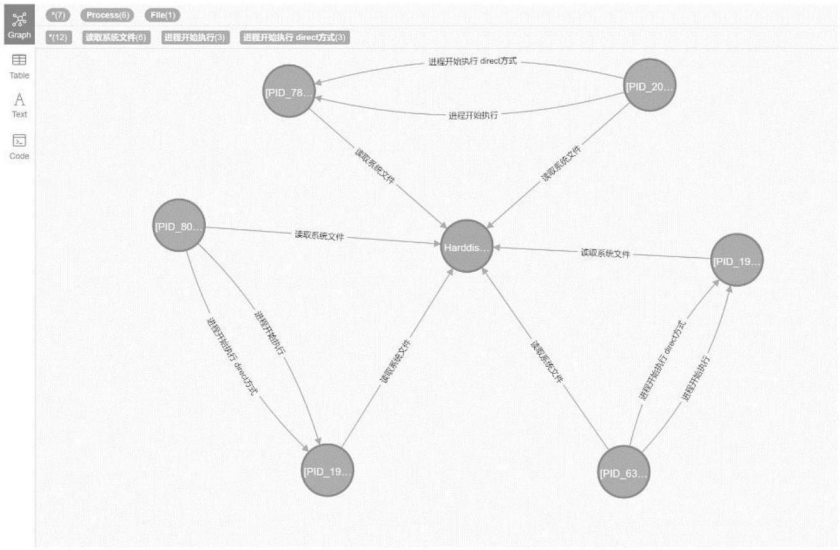

Graph
Table
Text
Code
进程开始执行 direct方式
进程开始执行
读取系统文件
[PID_78...
[PID_20...
[PID_80...
Harddis...
[PID_19...
[PID_19...
[PID_63...

图6