一种基于诱导机制的间谍软件检测方法

郭 春[1],罗 迪[1],申国伟[1],崔允贺[1],平 源[2]
(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,贵州贵阳 550025; 2. 许昌学院信息学院,河南许昌 461000)

摘 要: 间谍软件是攻击者广泛采用的一类信息窃取类恶意软件,具有高威胁性、高隐蔽性等特点. 间谍软件在实施窃密行为时通常采用触发执行策略,使得基于软件行为的动态检测方法难以在短时间内将其捕获,故上述方法检测间谍软件效果不佳. 针对该问题,本文采用主动诱导间谍软件执行窃密行为的思路,从应用程序编程接口(Application Programming Interface,API)层面分析不同诱导操作和诱导强度对间谍软件的不同诱发效果,进而提出一种基于诱导机制的间谍软件检测方法(Spyware Detection Method based on Inducement Mechanism,SDMIM). SDMIM包含诱导操作筛选、软件"活跃度"计算、间谍软件判别3个阶段,能够适用于多种类型间谍软件的诱导式检测. 实验结果表明,SDMIM能够在包含5种不同类型间谍软件的样本集上获得95.98%的检测准确率.

关键词: 间谍软件; 诱导操作; 动态检测; 触发执行策略; API调用
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112(2022)04-1014-11
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20211017

A Spyware Detection Method based on Inducement Mechanism

GUO Chun[1], LUO Di[1], SHEN Guo-wei[1], CUI Yun-he[1], PING Yuan[2]
(1. *State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, Guizhou* 550025, *China*;
2. *School of Information, Xuchang University, Xuchang, Henan* 461000, *China*)

Abstract: As a kind of information-stealing software, spyware is featured with high threat and concealment and is widely exploited by attackers nowadays. Since the stealing behavior is executed under a specific trigger strategy, it can hardly be captured by the mainstream malware detection methods based on dynamic behavior analysis in a short time. Frequently, the corresponding performance of spyware detection is below expectation. To tackle this problem, in this paper, the influence of different inducement operations and inducement strengths on the inducement effects of spyware from the (Application Programming Interface, API) level is firstly analyzed by introducing the idea of actively inducing spyware to perform its secret stealing behavior. Then, a Spyware detection method based on inducement mechanism (SDMIM) is proposed. SDMIM consists of three phases: inducible operation filtering, software "activity" calculation, and spyware discrimination. It is fit for the inducible detection of various types of spyware. Experimental results show that SDMIM can achieve an accuracy of 95.98% for detecting a dataset consisting of five kinds of spyware.

Key words: spyware; inducement operation; dynamic detection; trigger implementation strategy; API call

1 引言

间谍软件是一种高威胁性、高隐蔽性的恶意软件,它能够收集受害者信息并将这些信息在用户不知情的情况下发送给攻击者. 间谍软件窃取的信息包括用户访问过的网站列表、电子邮件、键盘击键记录、屏幕截图、信用卡号等[1],在多种数据泄露事件中间谍软件扮演着重要角色.《NTT Security 2019全球威胁情报报告》显示间谍软件在恶意软件中数量排名第一[2]. 卡巴斯基实验室公布的2019年Top 10病毒排名中,排名第一的也是窃取敏感信息的恶意软件. 间谍软件已经成为互联网所面临的主要安全威胁之一[3].

间谍软件作为以信息窃取为主要意图的恶意软件,比其他恶意软件更重视隐蔽性[4]. 间谍软件通常集

收稿日期:2021-08-01;修回日期:2022-03-04;责任编辑:崔兴华
基金项目:国家自然科学基金(No.62162009);贵州省自然科学基金(No.黔科合基础[2020]1Y268);河南省重点研发与推广专项(No.212102210084)

成触发执行、进程隐藏和文件隐藏等技术以提升自身在运行中的隐蔽性,其中触发执行指间谍软件需要在其预设的触发条件得到满足时才执行特定行为[5]. 当前主流的间谍软件检测方法是基于软件行为的动态检测方法,该类方法大多以软件在一定时长内的行为特征作为该软件有无恶意性的判别依据. 由于集成触发执行的间谍软件往往在大部分时间内仅执行少量不易暴露的"正常行为"[6],因此传统的基于软件行为的检测方法在实际环境中对该类软件的漏检率高或检测及时性不佳.

针对上述问题,本文采用诱导的思路,主动诱发间谍软件执行其窃密行为(主要为收集或泄露信息行为). 为提升间谍软件诱导的有效性,本文从应用程序编程接口(Application Programming Interface,API)层面分析不同诱导操作和诱导强度对间谍软件的不同诱发效果,并以此为基础构建了一种基于诱导机制的间谍软件检测方法(Spyware Detection Method based on Inducement Mechanism,SDMIM). SDMIM包含诱导操作筛选、软件"活跃度"计算、间谍软件判别3个阶段,能够适用于多种类型间谍软件的诱导式检测.

本文主要工作如下.

(1)对多款不同类型的间谍软件的行为执行方式进行分析,从API调用数量的角度探索间谍软件在有/无诱导操作情况下的行为差异以及不同诱导操作、诱导强度对间谍软件的不同诱发效果,为设计面向间谍软件的诱导式检测方法奠定基础.

(2)在对间谍软件行为诱发效果分析的基础上,提出了一种基于诱导机制的间谍软件检测方法SDMIM. SDMIM以组建的诱导操作集合为基础,能够适用于多种类型间谍软件的诱导式检测.

(3)使用不同类型的多款间谍软件及正常软件组成的样本集对SDMIM进行实验测试. 实验结果表明,SDMIM能够在该样本集上得到95.98%的检测准确率.

2　相关工作

间谍软件检测是当前恶意软件检测领域的研究热点之一. 近年来国内外研究者提出了多种检测方法,这些方法可分为静态检测方法、动态检测方法和基于诱导的检测方法.

2.1　静态检测方法

静态检测是指在不运行应用程序的情况下,运用静态分析技术分析程序的二进制文件或反汇编代码等固有属性,并从中提取程序的静态特征来进行恶意软件检测的方法[7]. Ding等人[8]提取恶意软件的n-gram操作码作为特征,利用深度置信网络检测恶意软件. Kumar等人[9]利用图像化恶意软件的OpCode并结合进化算法进行恶意软件检测. Guo等人[10]将勒索软件可视化为灰度图,利用VGG16提取特征后运用支持向量机分类勒索软件. Chu等人[11]将恶意代码转化为图像后,使用卷积神经网络分类恶意软件. 总体来说,静态检测方法无需运行应用程序便可进行检测,但是难以应对使用了加壳、代码混淆等技术的恶意软件[12].

2.2　动态检测方法

动态检测指在受控环境中执行恶意软件并观察软件与操作系统的交互行为,并以所观察到的行为特征进行检测的方法[13]. Javaheri等人[14]对系统进程行为进行分析,使用JRIP和J48决策树作为分类器识别间谍软件. Chen等人[15]在沙箱中运行勒索软件并采集API调用序列作为特征,再使用机器学习算法实现勒索软件检测. Allan等人[16]使用Cuckoo沙箱提取恶意软件API调用序列,结合N-gram和TF-IDF算法检测恶意软件. Fasano等人[17]运行并监控间谍软件的行为,使用时态逻辑公式检测间谍软件. Amer等人[18]使用Word2vec表示API函数之间上下文关系,构造马尔可夫链的检测恶意软件. Wang等人[19]使用Word2vec模型得到软件的API词向量,再利用残差网络提取特征并结合注意力机制检测恶意软件. 相比于静态检测方法,动态检测方法在检测集成了加壳等静态混淆技术的恶意软件上具有一定优势,但是该类方法大多对集成了触发执行等动态逃避技术的间谍软件的检测率低或检测及时性差[20].

2.3　基于诱导的检测方法

间谍软件常集成触发执行等动态逃避技术,执行窃密操作时隐蔽性强. 因此,开始使用诱导间谍软件执行窃密操作的思路[21~23]. 傅军等人[22]提出基于人工NK细胞的计算机先天免疫模型,通过释放一些间谍软件感兴趣的"诱导因子"引诱其实施恶意活动. 该方法借鉴了先天免疫系统中诱导的原理,但其诱导因子的种类较少且未对不同类型间谍软件的诱发效果进行分析. Alsaleh等人[23]提出了一种生成欺骗参数的方法,诱使恶意软件表现出与正常软件不同的恶意行为以便于检测. 总体来说,现有基于诱导的间谍软件检测研究尚未形成系统化的体系,还有待继续深入研究.

综上所述,现有的间谍软件检测方法尚存在一些不足:静态检测方法难以应对集成了代码混淆和加壳技术的间谍软件[24];基于行为的动态检测方法需要以软件的行为特征作为判别依据,在面对集成了触发执行策略的间谍软件时容易出现漏检率高或检测及时性不佳的情况[25];当前基于诱导的检测方法存在着诱导操作数量少且诱导操作有效性评估不够等问题,难以有效诱导多种类型的间谍软件. 针对现有方法存在的不足,本文从API调用数量的角度探索不同诱导操作、

诱导强度对间谍软件的不同诱发效果,并以此为基础构建适用于检测多种类型的间谍软件的诱导式检测方法.

3 间谍软件行为诱发效果分析

随着间谍软件检测与反检测的对抗博弈不断持续,间谍软件开发者为逃避检测在间谍软件中集成了触发执行等逃避技术.由于触发执行能够使间谍软件执行窃密行为时具有很强的隐蔽性,当前集成触发执行的间谍软件数量不断增加.针对该趋势,本文对当前主流的5种类型间谍软件主要窃密的行为执行方式进行了实验分析.表1给出了5种类型的303个间谍软件运行样本(由117款间谍软件采用不同设置得到,间谍软、件植入主机后无法更改其设置)的主要窃密行为所采用的行为执行方式.

表1 不同类型间谍软件的主要窃密行为执行方式

样本类型	样本占比	主要窃密行为执行方式	是否运用触发执行
键盘信息记录型(S_1)	33.3%(101/303)	系统中存在击键行为时,捕获敲击的键盘信息	√
剪切信息记录型(S_2)	18.5%(56/303)	系统中存在复制、剪切或粘贴行为时,记录复制、剪切或粘贴的内容	√
网页信息记录型(S_3)	17.2%(52/303)	系统中存在通过浏览器浏览网页行为时,记录浏览的网页信息	√
屏幕信息截取型(S_4)	16.2%(49/303)	系统中鼠标指针移动或打开/关闭软件时,截取屏幕信息	√
综合信息记录型(S_5)	14.8%(45/303)	系统中存在账号、口令、文件、软件等操作时,记录相关信息	√

从表1可知,上述间谍软件样本均采用触发执行策略执行其主要窃密行为,即当系统中存在特定操作时这些间谍软件才会实施其相应窃密行为.为检测集成了触发执行的间谍软件,本文采用诱导的思路,从API调用的角度探索间谍软件在有/无诱导操作情况下的行为差异以及不同诱导操作、诱导强度对间谍软件的不同诱发效果.

首先,为了探索间谍软件在有/无诱导操作情况下的行为差异,本文随机选取了5种类型的15个间谍软件样本和6种类型的8个正常软件样本.这些样本在安装有QQ、谷歌浏览器、360杀毒、爱奇艺视频、Wegame等软件的Win7操作系统中分别运行,并将各自在30 s的诱导操作(针对S_1~S_5类型的间谍软件分别执行键盘敲击、复制粘贴、浏览网页、打开应用程序和创建文件操作各5次)执行时间段(本文将该时间段命名为诱导期)和30 s的无诱导操作执行时间段(本文将该时间段命名为非诱导期)的API调用数量进行记录,得到表2的结果.

如表2所示,本文分析的5种类型的间谍软件样本在其诱导期和非诱导期均调用了一定数量的API,但各类型的间谍软件样本在其诱导期所调用的API数量明显多于非诱导期.这表明这些间谍软件在诱导期中实施了更多的行为.与此同时,表2还显示所分析的正常软件样本在诱导期和非诱导期调用的API数量相差很小,说明是否有诱导操作对这些正常软件的行为数量没有明显影响.

接下来分析不同诱导操作对间谍软件行为的影响.由表1可知键盘信息记录型间谍软件在其中占比最高,因此这里以键盘信息记录型的3个间谍软件以及多个类型的8个正常软件作为分析对象.在每个诱导期仅执行一种诱导操作5次的情况下,这些样本在5个诱导期(每个时长30 s)和5个非诱导期(每个时长30 s)所调用API的平均数量如表3所示.

表2 实验样本非诱导期和诱导期API调用数量

样本类型		样本名称	API调用数量	
			非诱导期/30 s	诱导期/30 s
间谍软件	键盘信息记录型	Keyloggerspymonitor	9 763	96 528
		Enregisterkey	10 507	72 013
		KmsKeylogger	11 307	70 155
	剪切信息记录型	Allinonekeylogger	11 089	70 274
		Isafe	9 813	75 996
		HZRMC	11 274	69 975
	网页信息记录型	Refog	11 913	156 488
		Actualspy	10 376	72 054
		PowrSpy	10 625	58 074
	屏幕信息截取型	Cyborg	10 575	57 877
		SpyMonitor	9 858	75 724
		Keyneniry	12 015	156 179
	综合信息记录型	HawkEye	10 572	82 368
		Predator	10 695	61 496
		Actualkeylogger	11 673	165 826
		间谍软件均值	10 803.6	89 401.8
正常软件	工具软件	WinRAR	10 076	9 675
		Notepad	10 462	10 589
	系统软件	Explorer	9 502	9 795
	办公软件	Office	9 152	9 733
	游戏软件	GameBar	9 592	9 756
	浏览器	Firefox	10 339	9 576
	其他软件	Wireshark	10 268	10 683
		360好压	9 493	10 865
		正常软件均值	9 860.5	10 084.0

表3显示所分析的间谍软件样本在不同的诱导操作下,所增加的API调用数量(诱导期的API数量减去

表3　实验样本在不同诱导操作下的API调用数量

诱导操作		平均API调用数		API变化数量(诱导期与非诱导期之差)
		非诱导期	诱导期	
敲击键盘	间谍软件	10 613.7	76 321.0	65 707.3
	正常软件	9 895.1	10 086.8	1 91.7
复制粘贴	间谍软件	10 976.0	19 614.7	8 638.7
	正常软件	10 362.6	10 429.0	66.4
浏览网页	间谍软件	9 799.8	14 278.5	4 478.7
	正常软件	10 696.5	10 056.8	−639.7
创建文件	间谍软件	10 703.3	16 123.4	5 420.1
	正常软件	10 193.0	10 237.3	44.3
编辑文件	间谍软件	9 641.2	16 348.2	6 707.0
	正常软件	10 679.3	10 158.5	−520.8

非诱导期的API数量)有一定差别,即不同的诱导操作对同一类间谍软件所诱发的行为数量存在着不同. 表3还显示正常软件样本在各诱导期和非诱导期中调用的API数量差别很小,再次验证了是否有诱导操作对正常软件的行为数量几乎没有影响.

最后分析单位时间内执行不同次数的诱导操作(本文命名为诱导强度,单位为“次/min”)对间谍软件行为的影响. 本文在3个30 s的诱导期内分别执行5次、10次和15次键盘敲击操作,记录3个键盘信息记录型间谍软件样本在各诱导期内各自所调用的API数量,结果如图1所示. 由图1可知,随着诱导强度的增加,各间谍软件在其诱导期内调用了更多数量的API. 这表明不同的诱导强度对间谍软件的诱发效果产生不同影响,且诱导强度与所诱发的间谍软件行为数量成正相关关系.

综合上述分析可知,间谍软件在其诱导期调用的API数量明显多于其非诱导期,且不同的诱导操作及诱导强度对间谍软件所诱发产生的API数量存在着影响. 该结果为本文设计基于诱导机制的间谍软件检测方法奠定了基础. 需要注意的是若某款间谍软件能够使用低级的代码指令直接访问操作系统底层资源(即绕过系统函数接口调用),则无法通过API分析该间谍软件,此时可以通过网络流量分析该类间谍软件.

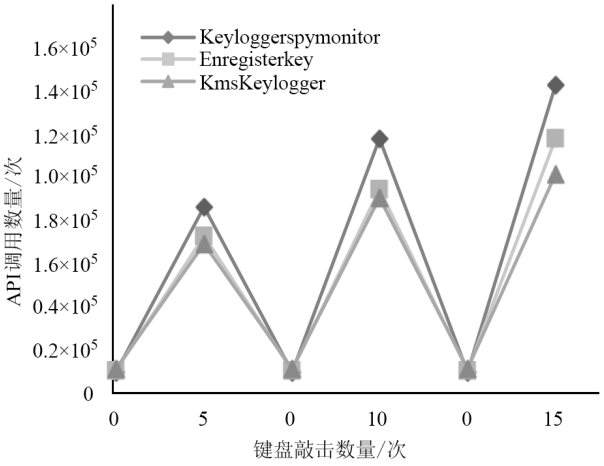

图1　间谍软件在不同诱导强度下的API调用数量

4　基于诱导机制的间谍软件检测方法

基于第3节的分析结果,本文提出了一种间谍软件检测方法SDMIM. SDMIM总体框架如图2所示,包含了诱导操作筛选、软件“活跃度”计算、间谍软件判别3个主要阶段. 下面分别对3个阶段进行详细描述.

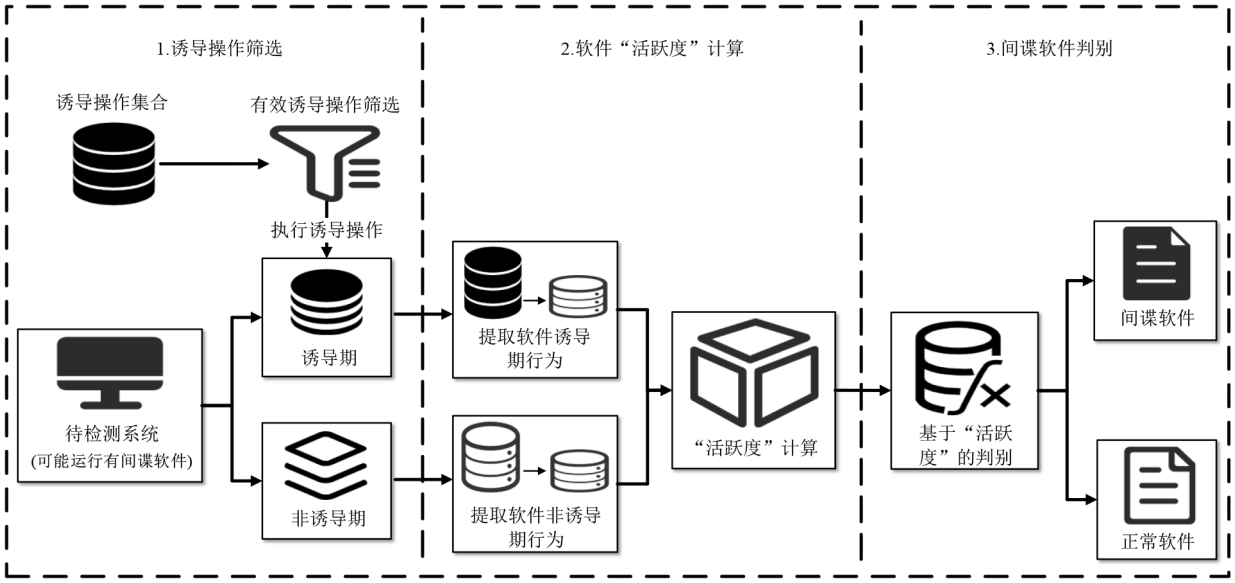

图2　基于诱导机制的间谍软件检测方法总体框图

4.1 诱导操作筛选

第3节的分析结果显示,不同的诱导操作对同一类间谍软件的诱发效果不同. 为了提升对不同类型间谍软件的诱发效果,SDMIM组建了一个包含多种诱导操作的集合,并通过所设计的诱导操作筛选算法从中选取对各间谍软件有效的诱导操作.

(1)诱导操作集合组建

本文对多种类型的间谍软件样本进行逆向分析以及运行过程分析,发现间谍软件对一些用户事件具有很高的关注度. 这些用户事件大多能够通过用户可执行的操作产生. 根据该特点,本文在系统中主动实施不同种类及数量的用户操作,观察间谍软件样本对这些用户操作的反应情况,最终从上百种用户操作中选出了如表4所示的18种受到较多间谍软件样本关注的用户操作,组建为诱导操作集合$O=\{(i_1,10),(i_2,5),\cdots,(i_{18},2)\}$. O中每一个诱导操作(i_i,c_i)代表着诱导行为i_i及其执行的强度c_i.

表4 诱导操作集合

序号	诱导行为	强度	序号	诱导行为	强度
1	敲击键盘	12	10	重命名文件操作	4
2	复制粘贴操作	6	11	安装软件	4
3	打开文本文档	6	12	卸载软件	4
4	关闭文本文档	6	13	打开浏览器	6
5	打开应用程序	6	14	关闭浏览器	6
6	关闭应用程序	6	15	浏览网页	6
7	创建文件操作	4	16	截取屏幕	6
8	编辑文件操作	4	17	网络下载	4
9	删除文件操作	4	18	收藏网站	4

(2)有效诱导操作筛选

由于不同诱导操作对同一类间谍软件的API调用数量产生的影响不同,SDMIM需要从诱导操作集合O中选出能够有效诱发待检测间谍软件实施大量行为的诱导操作. 本文设计了如算法1所示的有效诱导操作筛选算法. 该算法首先从O中依次选取各诱导操作(i_i, c_i)在系统中执行30秒并获取该时间段内系统中API调用数量m_i,再将其与设定好的阈值$M_{\text{Threshold}}$进行比较:若执行(i_i, c_i)所得到的m_i大于等于$M_{\text{Threshold}}$则停止筛选并(i_i, c_i)选为有效诱导操作;若O中各诱导操作所得到的m_i均小于$M_{\text{Threshold}}$,则选取其中产生最大m_i的(i_i, c_i)作为有效诱导操作.

算法1 有效诱导操作筛选

输入:诱导操作集合O={$(i_1,10),(i_2,5),\cdots,(i_{18},2)$},阈值$M_{\text{Threshold}}$

输出:选取的有效诱导操作(i_i, c_i)

```
FOR i = 1,2,…,18 DO
  自动执行诱导操作(i_i, c_i)
  记录系统中API调用数量 m_i
  IF ( m_i ≥ M_Threshold )
    选取该诱导操作(i_i, c_i)
    BREAK
  ELSE
    i = i + 1
END FOR
IF ( i >18 )
  选取 M_i 最大的诱导操作(i_i, c_i)
END IF
```

4.2 软件“活跃度”计算

第3节的分析结果显示,间谍软件在诱导期和非诱导期的API调用数量存在明显差别,而正常软件在这两个时间段的API调用数量几乎没有差别. 因此,为数量化表示软件在诱导期和非诱导期的活跃程度,本文定义了一个“活跃度”(I_{behavior}),软件在一个时间段的“活跃度”数值越高表示其在该时期的行为执行越频繁. 软件“活跃度”计算阶段主要包括软件行为提取和活跃度计算两个步骤. 下面分别进行描述.

(1)软件行为提取

本文认为间谍软件在诱导期相对于其在非诱导期多出的API调用数量主要是由执行诱导操作所诱发. 为提取诱导期和非诱导期的软件行为,SDMIM将上一阶段筛选出的有效诱导操作,按照预先设定好的时间间隔在待检测系统中执行并通过监控工具Process Monitor提取系统中各软件的API(表5所示的间谍软件常用的文件系统API、进程间通信API等). 诱导期中提取的API表示为诱导期行为集合Induce_Key$B$=$\{(\text{API}_1, H_1),(\text{API}_2,H_2),\cdots,(\text{API}_n,H_n)\}$,非诱导期中提取的API表示为非诱导期行为集合No-induce_Key$B$=$\{(\text{API}_1, J_1),(\text{API}_2,J_2),\cdots,(\text{API}_n,J_n)\}$. 其中$\text{API}_i$代表某种API,$H_i$和$J_i$代表该API的调用数.

(2)“活跃度”计算

基于上一步骤得到的Induce_Key$B$与No-induce_Key$B$,SDMIM将通过式(1)和式(2)分别计算API_i在诱导期和非诱导期的活动值$R_{\text{induce_}i}$和$R_{\text{no-induce_}i}$.

$$R_{\text{induce\_}i}=\frac{H_i}{T_{\text{induce}}\times c_i} \tag{1}$$

其中,H_i为Induce_Key$B$中API_i的调用数,T_{induce}为诱导期时长(单位为min),c_i为诱导强度.

$$R_{\text{no-induce\_}i}=\frac{J_i}{T_{\text{no-induce}}} \tag{2}$$

其中,J_i为No-induce_Key$B$中API_i的调用数,$T_{\text{no-induce}}$为非诱导期时长.

软件在诱导期和非诱导期中调用的API均来源于该软件在此期间所实施的行为. SDMIM从行为覆盖全

表5　SDMIM监控的API

API序号	API名称	API序号	API名称
API_1	CloseFile	API_{17}	QueryBasicInformationFile
API_2	CreateFile	API_{18}	QueryDirectory
API_3	LockFile	API_{19}	QueryNOpenInformationFile
API_4	ReadFile	API_{20}	QueryNameInformationFile
API_5	WriteFile	API_{21}	QueryOpen
API_6	RegCloseKey	API_{22}	QueryStandardInformationFile
API_7	RegCreateKey	API_{23}	CreateFileMapping
API_8	RegEnumKey	API_{24}	FRForSectionSynchronization
API_9	RegEnumValue	API_{25}	FileSystemControl
API_{10}	RegOpenKey	API_{26}	FlushBuffersFile
API_{11}	RegQueryKey	API_{27}	IrpMJClose
API_{12}	RegQueryKeySecurity	API_{28}	Process Profiling
API_{13}	RegQueryValue	API_{29}	SetEndOfFileInformationFile
API_{14}	RegSetInfoKey	API_{30}	UDP Receive
API_{15}	RegSetValue	API_{31}	UDP Send
API_{16}	QueryAInformationVolume	N/A	N/A

面性的角度出发,通过式(3)将API_i在诱导期的R_{induce_i}和非诱导期的$R_{no\text{-}induce_i}$进行加权结合以得到该API综合活动值R_i.

$$R_i = w_1 \times R_{induce\_i} + w_2 \times R_{no-induce\_i} \tag{3}$$

其中,w_1和w_2为权重因子. 为了突出诱导操作对间谍软件的诱发效果,本文将w_1和w_2分别设定为0.9和0.1.

接下来SDMIM将API_i的综合活动值R_i与预设阈值$R_{Threshold_i}$如式(4)所示进行比较:当R_i大于$R_{Threshold_i}$则将该API选定为高敏感API(代表其对诱导操作反映敏感),其敏感度量值$Sensitive_i$(本文用于数量化表示API的敏感程度)等于R_i;反之则其$Sensitive_i$等于0. 最后,SDMIM使用式(5)以软件为单位叠加$Sensitive_i$(i=1,2,…,31)得到待测软件的活跃度$I_{behavior}$.

$$Sensitive_i = \begin{cases} 0, R_i \leqslant R_{Threshold\_i} \\ R_i, R_i > R_{Threshold\_i} \end{cases} \tag{4}$$

$$I_{behavior} = \sum_{i=1}^{n=31} Sensitive_i \tag{5}$$

综上所述,通过所提取的软件行为集合计算该软件活跃度的流程如算法2所示.

4.3　间谍软件判别

SDMIM在本阶段将利用式(6)对软件的活跃度$I_{behavior}$与预定的阈值$I_{Threshold}$进行比较,根据比较结果判断待测软件是间谍软件还是正常软件:当Y=0时,将待检测样本判别为正常软件;当Y=1时,则将待检测软件判别为间谍软件.

$$Y = \begin{cases} 0, I_{behavior} < I_{Threshold} \\ 1, I_{behavior} \geqslant I_{Threshold} \end{cases} \tag{6}$$

算法2　软件"活跃度"计算

输入:样本x的诱导期行为集合Induce_KeyB与非诱导期行为集合No-induce_Key$B$

输出:样本x的活跃度$I_{behavior}$

1. FOR i=1,2,…,31 do
2. 　根据式(1)计算Induce_Key$B$中API_i的R_{induce_i}
3. 　根据式(2)计算No-induce_Key$B$中API_i的$R_{no\text{-}induce_i}$
4. 　根据式(3)计算API_i的R_i
5. 　根据式(4)计算API_i的$Sensitive_{_i}$
6. END FOR
7. 根据式(5)将x的所有$Sensitive_i$累加得到$I_{behavior}$

5　实验与结果

5.1　实验环境

为了测试SDMIM的有效性,本文在配置为20 G内存、Intel i7-8700处理器的物理机中搭建FTP服务器接收间谍软件回传的信息;以配置为2 G内存、60 G硬盘的VMware Workstation虚拟机作为实验样本的实验环境,该环境中运行着如表6所示的正常软件. 无特别说明时实验中T_{induce}和$T_{no\text{-}induce}$均设置为1 min(两者之和在实验表示为Time,对应SDMIM提取行为的时长),$I_{Threshold}$设置为7000,使用Process Monitor捕获API数据.

5.2　数据集

本文实验集中间谍软件主要来源于virustotal.com,

表6　实验环境中已运行的正常软件

应用类型	应用名	应用类型	应用名
浏览器类	谷歌浏览器	娱乐类	爱奇艺视频、Wegame
聊天类	腾讯QQ	系统类	csrss.exe、dwn.exe
安全软件类	360杀毒	工具类	PDF阅读器、迅雷下载

virusshare.com，github.com等网站，正常软件来源于360软件管家. 实验集包含5种类型共303个间谍软件运行样本(由117款间谍软件采用不同设置得到，间谍软件植入主机后无法更改其设置)和7种类型220个的正常软件样本，包括办公软件、聊天交友软件、查杀类软件、工具软件等. 实验数据集中样本分布如表7所示.

表7 实验样本类型及数量

正常软件样本类型	数量	间谍软件样本类型	数量
办公软件	52	键盘信息记录型	101
视频软件	26	剪切信息记录型	56
游戏软件	18	网页信息记录型	52
工具软件	46	屏幕信息截取型	49
查杀类软件	18	综合信息记录型	45
聊天交友软件	20	N/A	N/A
其它应用软件	40	N/A	N/A
正常软件样本总计	220	间谍软件样本总计	303

5.3 评估标准

本文采用恶意软件检测领域常用的准确率(Accuracy)、精确率(Precision)、召回率(Recall)以及F1作为SDMIM检测性能的评估指标，这些指标可通过式(7)~(10)进行计算.

$$\text{Accuracy}=\frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} \tag{7}$$

$$\text{Precision}=\frac{\text{TP}}{\text{TP}+\text{FP}} \tag{8}$$

$$\text{Recall}=\frac{\text{TP}}{\text{TP}+\text{FN}} \tag{9}$$

$$\text{F1}=2\times\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}} \tag{10}$$

5.4 实验结果及分析

5.4.1 $M_{\text{Threshold}}$阈值与$R_{\text{Threshold_}i}$阈值选取

首先针对算法1中的阈值$M_{\text{Threshold}}$和式(4)中阈值$R_{\text{Threshold_}i}$选取进行实验. 本文从实验集中随机抽取10个键盘信息记录型间谍软件样本和10个正常软件样本，在搭建的实验环境中执行强度为12次/min的键盘敲击诱导操作. 本文将仅有正常软件运行(实验环境+某个正常软件)的系统称为正常系统，将有间谍软件运行(实验环境+某个间谍软件)的系统称为异常系统. 上述正常系统和异常系统在30s诱导期内的API调用数量如表8所示.

从表8的统计数据可以看到，存在行为诱发执行的异常系统同时段API调用数量普遍高于正常系统，且波动范围也大于正常系统. 为简化分析，本文参考使正常与异常系统的统计间距最大化的思路，为后续实验的$M_{\text{Threshold}}$赋予略低于二者均值之差的值，如取$M_{\text{Threshold}}$=72212，如图3所示该值略小于异常系统均值(116064.6)与正常系统均值(42978.2)的差值(73086.4).

表8 不同正常系统与异常系统在30秒诱导期的API调用数量

正常系统	API调用数量	异常系统	API调用数量
WinRAR	36 083	Actualkeylog	182 946
Notepad	38 405	Actualspy	105 897
Explorer	40 923	Cyborg	89 362
Office	45 286	Allinonekeylog	97 524
Wireshark	49 827	Isafe	109 450
Firefox	47 439	Refog	176 682
GameBar	39 682	Enregisterkey	96 283
360好压	40 028	PowrSpy	87 449
微信	41 916	KmsKeylog	112 067
Xshell	50 193	SpyMonitor	102 986
正常系统均值	42 978.2	异常系统均值	116 064.6

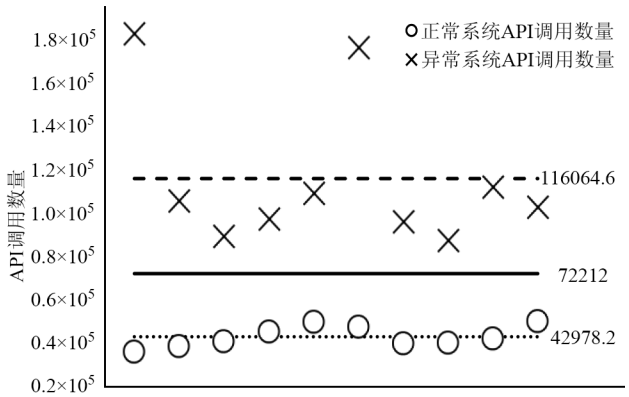

图3 不同正常系统与异常系统在30s诱导期的API调用数量分布

$R_{\text{Threshold_}i}$值关系到高敏感API的判定. 本文以上述10个间谍软件样本和10个正常软件样本30 s内的API调用数量为输入，通过式(1)~(3)计算两类软件的31种API各自的综合活动值R_i，结果如表9所示. 本文将表9中的R_i值以$\frac{(\text{间谍软件}Ri+\text{正常软件}Ri)}{2}$计算得到的值作为各API的$R_{\text{Threshold_}i}$值进行后续实验.

5.4.2 不同$I_{\text{Threshold}}$阈值下SDMIM的检测效果

为测试不同阈值$I_{\text{Threshold}}$下的SDMIM检测精度，本文在5个不同$I_{\text{Threshold}}$下使用实验集(包含303个间谍软件样本和220个正常软件样本)进行检测，结果如表10和图4所示.

如表10所示，当$I_{\text{Threshold}}$为10000时，SDMIM获得最高的Precision(96.62%)；当$I_{\text{Threshold}}$为6000时，SDMIM获得最高的Recall(97.03%)；当$I_{\text{Threshold}}$为8000时，SDMIM获得最高的F1值(96.54%). 分析以上结果，当设定的$I_{\text{Threshold}}$较低时，大部分间谍软件样本能够被正确判定，但是此时的$I_{\text{Threshold}}$会使得部分正常软件样本被错误判定为间谍软件，对应SDMIM获得较高的Recall值和其较低的Precision和F1值. 而当设定的$I_{\text{Threshold}}$较高时，大部分正常软件样本能够被正常判定，但是此时的$I_{\text{Threshold}}$

表9　SDMIM中高敏感API判定阈值$R_{Threshold_i}$取值

API序号	间谍软件R_i	正常软件R_i	阈值$R_{Threshold_i}$	API序号	间谍软件R_i	正常软件R_i	阈值$R_{Threshold_i}$
API_1	1036.3	110.6	573.5	API_{17}	687.1	31.3	359.2
API_2	1113.7	107.9	610.8	API_{18}	2.6	2.5	2.5
API_3	3.6	1.1	2.3	API_{19}	2.8	1.4	2.1
API_4	29.1	12.3	20.7	API_{20}	3.7	2.0	2.8
API_5	506.8	86.4	296.6	API_{21}	462.9	39.2	251.0
API_6	435.2	259.4	347.3	API_{22}	58.2	12.3	35.2
API_7	7.9	5.3	6.6	API_{23}	679.8	28.3	354.1
API_8	361.7	135.3	248.5	API_{24}	662.5	28.4	345.4
API_9	16.6	10.2	13.4	API_{25}	6.1	7.6	6.8
API_{10}	588.7	432.6	510.6	API_{26}	5.1	6.8	5.9
API_{11}	495.6	476.8	486.2	API_{27}	1035.6	55.3	545.4
API_{12}	81.4	63.7	72.5	API_{28}	712.8	688.2	700.5
API_{13}	551.7	494.6	523.1	API_{29}	1.7	1.4	1.5
API_{14}	4.5	1.5	3.0	API_{30}	2.6	0.5	1.6
API_{15}	3.2	1.2	2.2	API_{31}	2.7	0.5	1.6
API_{16}	2.6	2.0	2.3	N/A	N/A	N/A	N/A

表10　SDMIM在不同$I_{Threshold}$下获得的Precision,Recall和F1

$I_{Threshold}$	Time/min	Precision/%	Recall/%	F1/%
6 000	2	90.18	**97.03**	93.48
7 000	2	91.88	97.02	94.38
8 000	2	96.38	96.70	**96.54**
9 000	2	96.32	95.05	95.68
10 000	2	**96.62**	94.39	95.49

会使得部分间谍软件样本被错误判定为正常软件,对应SDMIM获得较高的Precision和较低的Recall和F1值.对应到图4所示的SDMIM在不同$I_{Threshold}$下得到的Accuracy值,也是当$I_{Threshold}$为8000时,SDMIM得到Accuracy值的最高(95.98%).

5.4.3　不同Time下SDMIM的检测效果

为测试不同Time下的SDMIM检测精度本文设置$I_{Threshold}$为8000,在5个不同Time下使用实验集进行检测,结果如表11和图5所示.

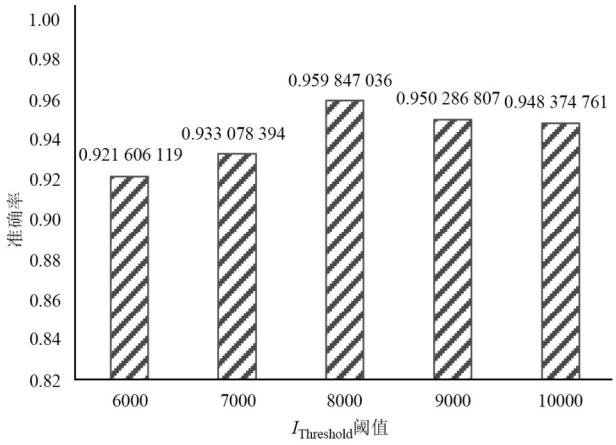

图4　SDMIM在不同$I_{Threshold}$下获得的Accuracy

表11　SDMIM在不同Time下获得的Precision,Recall和F1

Time/min	$I_{Threshold}$	Precision/%	Recall/%	F1/%
0.5	8 000	91.61	93.73	92.66
1	8 000	92.88	94.72	93.79
2	8 000	96.38	96.70	96.54
4	8 000	96.71	97.03	96.86
6	8 000	**96.73**	**97.68**	**97.21**

如表11所示,随着使用的检测时间Time逐渐增加,SDMIM获得的Precision,Recall和F1值呈现上升趋势.具体来说,当Time为6 min时,SDMIM所获得的Precision、Recall以及F1最高,分别达到96.37%,97.68%和97.21%.这是由于间谍软件样本在诱导期中调用的API数量会大幅度增加,而正常软件无此变化,因此随着Time的增加,两类软件的API调用数量差距也随会之扩大.因此,更长的Time能够提升SDMIM对间谍软件和正常软件的区分效果.相应的,图5显示随着Time的增加,SDMIM所获得的Accuracy值也逐渐提升.当Time为6min时,SDMIM获得本实验中最高的Accuracy(96.75%).

5.4.4　方法对比

为进一步评估SDMIM的检测精度,本文将SDMIM与文献[10]、文献[16]、文献[18]中方法在本文实验集上进行了对比实验.文献[10]中方法为基于软件可视化的静态检测方法.文献[16]和文献[18]中方法均为基于行为的动态检测方法.以本文各实验样本在2 min时间段内调用的API序列作为数据集进行实验.上述对比方法将实验集以7:3划分为训练集和测试集,实验

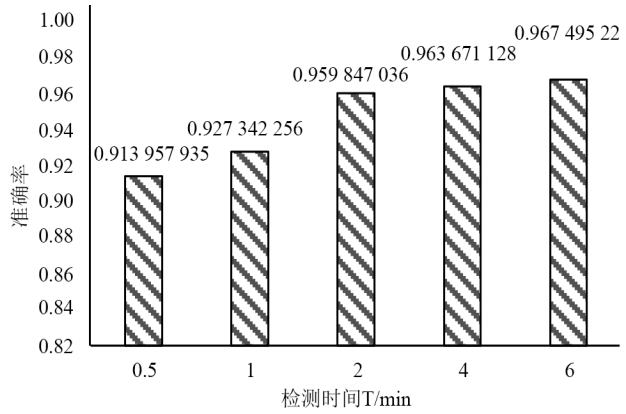

图5 SDMIM在不同Time下获得的Accuracy

结果如表12所示.

表12 不同检测方法的检测结果对比

方法(类别)	Accuracy	Precision	Recall	F1
Guo[10](静态)	93.50%	94.68%	94.06%	94.37%
Ninyesiga[16](动态)	95.03%	96.01%	95.38%	95.70%
Amer[18](动态)	95.41%	96.35%	95.71%	96.03%
SDMIM(动态)	**95.98%**	**96.38%**	**96.70%**	**96.54%**

如表12所示,SDMIM得到的Accuracy、Precision、Recall和F1均优于文献[10]、文献[16]和文献[18]中的方法. 为进一步分析上述结果,本文将各方法在实验中未能正确判别的样本类型及数量进行了统计及分析,结果如表13所示.

表13 各方法判别错误样本统计及分析

方法	间谍软件判别错误样本类型及数量	原因分析	正常软件判别错误类型及数量	原因分析
Guo[10]	网页信息记录:5个 综合信息记录:6个 屏幕信息截取:5个 剪切信息记录:2个	这些间谍软件样本生成的灰度图与用于训练检测模型的训练样本所生成灰度图的相似度低	工具软件:4个 办公软件:4个 视频软件:2个 其它应用软件:6个	这些正常软件样本生成的灰度图与用于训练检测模型的正常软件样本所生成灰度图的相似度低
Ninyesiga[16]	键盘信息记录:4个 剪切信息记录:4个 网页信息记录:3个 综合信息记录:3个	部分间谍软件样本在其触发条件未满足时不执行窃密行为,此时间段内其API序列与正常软件的API序列区别不明显	工具软件:3个 查杀类软件:3个 聊天交友软件:4个 其他应用软件:2个	部分聊天软件类正常软件样本运行时会频繁调用网络API,这与间谍软件信息发送行为相似
Amer[18]	键盘信息记录:4个 剪切信息记录:4个 网页信息记录:3个 综合信息记录:2个	部分间谍软件样本在其触发条件未满足时不执行窃密行为,此时间段内其API调用序列与正常软件的API序列区别不明显	工具软件:3个 查杀类软件:3个 聊天交友软件:4个 其他应用软件:1个	部分聊天软件类正常软件样本运行时会频繁调用网络API,这与间谍软件信息发送行为相似
SDMIM	网页信息记录:5个 屏幕信息截取:3个 剪切信息记录:2个	当前的诱导操作集合对部分网页信息记录类间谍软件样本的诱发效果较差	工具软件:2个 查杀类软件:6个 聊天交友软件:2个 其他应用软件:1个	部分安全杀毒类软件样本对一些诱导操作比较敏感,例如会在系统中出现部分软件行为时记录相关信息并上传至云端进行分析

6 总结

本文通过分析间谍软件在有/无诱导操作情况下的行为差异以及不同诱导操作和诱导强度对间谍软件诱发效果的影响,发现间谍软件在其诱导期调用的API数量明显多于其非诱导期,且不同的诱导操作及诱导强度对间谍软件所诱发产生的API数量存在着不同影响. 基于该分析结果,本文提出了一种基于诱导机制的间谍软件检测方法SDMIM. 实验结果显示,SDMIM对5种类型的间谍软件样本具有良好的检测效果. 间谍软件在诱导过程中调用的API数量增加,表明诱导式检测方案能够使间谍软件实施更多的行为以便对其进行检测,在高隐蔽性间谍软件检测领域具有巨大的应用价值. 后续将探索如何解决本方法对查杀类软件检测效果不佳的问题,以期进一步提高对间谍软件的检测效果.

参考文献

[1] DROZD O, KHARCHENKO V, RUCINSKI A, et al. Development of models in resilient computing[C]//2019 International Conference on Dependable Systems, Services and Technologies. Leeds: IEEE, 2019: 1-6.

[2] Symantec. 2019 Internet Security Threat Report[EB/OL]. [2020-06-28]. https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.html.

[3] AFZULPURKAR A, ALSHEMAILI M, SAMARA K. Outgoing data filtration for detecting spyware on personal computers[C]//Advances in Internet, Data and Web Tech-

nologies. Switzerland: Springer, 2019: 355-362.

[4] WANG Z, LIU Q, CHI Y. Review of android malware detection based on deep learning[J]. IEEE Access, 2020, 8: 181102-181126.

[5] BADIH H, BOND B, RRUSHI J. On second-order detection of webcam spyware[C]//2020 International Conference on Information and Computer Technologies. San Jose: IEEE, 2020: 424-431.

[6] MALLIKARAJUNAN K, PREETHI S R, SELVALAKSHMI S, et al. Detection of spyware in software using virtual environment[C]//2019 International Conference on Trends in Electronics and Informatics. Tirunelveli: IEEE, 2019: 1138-1142.

[7] 李鹏伟, 姜宇谦, 薛飞扬, 等. 一种基于深度学习的强对抗性Android恶意代码检测方法[J]. 电子学报, 2020, 48(8): 48-54.
LI P W, JIANG Y Q, XUE F Y, et al. A robust approach for android malware detection based on deep learning[J]. Acta Electronica Sinica, 2020, 48(8): 1502-1508. (in Chinese)

[8] DING Y X, ZHU S Y. Malware detection based on deep learning algorithm[J]. Neural Comput & Applic, 2017, 31: 461-472.

[9] KUMAR R. Malicious code detection based on image processing using deep learning[C]//2018 Proceedings of the 2018 International Conference on Computing and Artificial Intelligence. New York: ACM, 2018: 81-85.

[10] 郭春, 陈长青, 申国伟, 等. 一种基于可视化的勒索软件分类方法[J]. 信息网络安全, 2020, 20(4): 31-39.
GUO C, CHEN C Q, SHEN G W, et al. A visualization-based ransomware classification method[J]. Information Network Security, 2020, 20(4): 31-39. (in Chinese)

[11] CHU Q, LIU G, ZHU X. Visualization feature and CNN based homology classification of malicious code[J]. Chinese Journal of Electronics, 2020, 29(1): 154-160.

[12] CHOUDHARY S P, VIDYARTHI M D. A simple method for detection of metamorphic malware using dynamic analysis and text mining[J]. Procedia Computer Science, 2015, 54: 265-270.

[13] DAMODARAN A, TROIA F D, VISAGGIO C A, et al. Acomparison of static, dynamic, and hybrid analysis for malware detection[J]. ComputVirol Hack Tech, 2017, 13(1): 1-12.

[14] JAVAHERI D, HOSSEINZADEH M, RAHMANI A M. Detection and elimination of spyware and ransomware by intercepting kernel-level system routines[J]. IEEE Access, 2018, 6: 78321-78332.

[15] 陈长青, 郭春, 崔允贺, 等. 基于API短序列的勒索软件早期检测方法[J]. 电子学报, 2021, 49(3): 586-595.
CHEN C Q, GUO C, CUI Y H, et al. Early detection method of ransomware based on API short sequence[J]. Acta Electronica Sinica, 2021, 49(3): 586-595. (in Chinese)

[16] ALLAN N, NGUBIRI J. Windows PE API calls for malicious and benigin programs[J]. International Journal of Technology and Management, 2019, 3(2): 1-9.

[17] FASANO F, MARTINELLI F, MERCALDO F, et al. Spyware detection using temporal logic[C]//Proceedings of the 5th International Conference on Information Systems Security and Privacy. Portugal: SCITEPRESS, 2019: 690-699.

[18] ESLAM A, IVAN Z. A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence[J]. Computers & Security, 2020, 92: 101760.1-101760.15.

[19] WANG L, WANG B, ZHAO P, et al. Malware detection algorithm based on the attention mechanism and resnet [J]. Chinese Journal of Electronics, 2020, 29(6): 1054-1060.

[20] BELOUS A, SALADUKHA V. Computer viruses, malicious logic, and spyware[M]//Viruses, Hardware and Software Trojans, Attacks and Countermeasures. Switzerland: Springer, 2020: 101-207.

[21] BEJOY B J, SUBBIAH J. An intrusion detection and prevention system using ais-an nk cell-based approach[M]//Lecture Notes in Computational Vision and Biomechanics. Switzerland: Springer, 2018: 883-893.

[22] 傅军, 杨欢, 芮平亮, 等. 基于计算机免疫的间谍软件自适应诱导与检测方法: CN201310466755.6[P]. 2016-08-31.

[23] ALSALEH M N, WEI J, ALSHAER E, et al. Gextractor: automated extraction of malware deception parameters for autonomous cyber deception[M]//Autonomous Cyber Deception. Switzerland: Springer, 2019: 185-207.

[24] HUTCHINSON S, ZHOU B, KARABIYIK U. Are we really protected an investigation into the play protect service [C]//2019 IEEE International Conference on Big Data. San Jose: IEEE, 2019: 4997-5004.

[25] TAHIR R. A study on malware and malware detection techniques[J]. International Journal of Education and

Management Engineering, 2018, 8(2): 20-30.

作者简介

郭 春 男,1986年生,贵州贵阳人.博士,贵州大学计算机科学与技术学院副教授,CCF会员.主要研究领域为数据挖掘、入侵检测、恶意代码检测.
E-mail: gc_gzedu@163.com

罗 迪 男,1996年生,贵州六盘水人.贵州大学计算机科学与技术学院硕士研究生,CCF学生会员.主要研究方向为计算机网络与信息安全.
E-mail: luodi_happy@163.com

申国伟 男,1986年生,湖南邵东人. 贵州大学计算机科学与技术学院教授、硕士生导师,CCF会员.主要研究领域为网络与信息安全、大数据.
E-mail: gwshen@gzu.edu.cn

崔允贺(通讯作者) 男,1987年生,贵州贵阳人.贵州大学计算机科学与技术学院讲师、硕士生导师.主要研究领域为网络安全、云计算、数据中心.
E-mail: yhcui@gzu.edu.cn

平 源 男,1981年生,重庆合川人.博士,许昌学院信息工程学院教授.主要研究领域为机器学习、数据隐私安全、云计算、边缘计算.
E-mail: pyuan. lhn@xcu.edu.cn