基于API潜在语义的勒索软件早期检测方法

罗 斌[1],郭 春[1*],申国伟[1],崔允贺[1],陈 意[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. 许昌学院信息工程学院,河南许昌461000)

摘 要: 加密型勒索软件通过加密用户文件来勒索赎金.现有的基于第一条加密应用编程接口(Application Programming Interface,API)的早期检测方法无法在勒索软件执行加密行为前将其检出.由于不同家族的勒索软件开始执行其加密行为的时刻各不相同,现有的基于固定时间阈值的早期检测方法仅能将少量勒索软件在其执行加密行为前准确检出.为进一步提升勒索软件检测的及时性,本文在分析多款勒索软件运行初期调用动态链接库(Dynamic Link Library,DLL)和API行为的基础上,提出了一个表征软件从开始运行到首次调用加密相关DLL之间的时间段的概念——运行初始阶段(Initial Phase of Operation,IPO),并提出了一个以软件在IPO内产生的API序列为检测对象的勒索软件早期检测方法,即基于API潜在语义的勒索软件早期检测方法(Ransomware Early Detection Method based on API Latent Semantics,REDMALS).REDMALS采集IPO内的API序列后,采用TF-IDF(Term Frequency-Inverse Document Frequency)算法以及潜在语义分析(Latent Semantic Analysis,LSA)算法对采集的API序列生成特征向量及提取潜在的语义结构,再运用机器学习算法构建检测模型用于勒索软件检测.实验结果显示运用随机森林算法的REDMALS在构建的变种测试集和未知测试集上可分别获得97.7%、96.0%的准确率,且两个测试集中83%和76%的勒索软件样本可在其执行加密行为前被检出.

关键词: 勒索软件;早期检测;API;TF-IDF;潜在语义分析;随机森林

基金项目: 国家自然科学基金(No.62162009);贵州省科技支撑计划(No.[2022]071);贵州省高等学校大数据与网络安全创新团队(No.[2023]052);河南省科技攻关计划项目(No.222102210048)

中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112(2024)04-1288-08

电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20231039

Ransomware Early Detection Method Based on API Latent Semantics

LUO Bin[1], GUO Chun[1*], SHEN Guo-wei[1], CUI Yun-he[1], CHEN Yi[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 Engineering, Xuchang University, Xuchang, Henan* 461000, *China*)

Abstract: Cryptographic ransomware extorts a ransom by encrypting user files. Existing early detection methods based on the first encryption-related application programming interface (API) cannot detect ransomware before it executes encryption behavior. Because the point at which different ransomware families begin executing their encryption behavior varies, existing early detection methods based on fixed time thresholds can only accurately detect a small fraction of ransomware before it executes encryption behavior. To further improve the timeliness of ransomware detection, this article proposes a concept that characterizes the time period from the start of software operation to the first call of encryption-related dynamic-link libraries (DLLs), namely the initial phase of operation (IPO). Based on the analysis of DLL and API call behavior in the early operational phase of several ransomwares, this article presents a method based on the API sequences generated by the software within the IPO as the detection object, namely the ransomware early detection method based on API latent semantics (REDMALS). REDMALS captures the API sequences within the IPO, uses the term frequency-inverse document frequency algorithm and the latent semantic analysis algorithm to generate feature vectors on the captured API sequences and to extract potential semantic structures, respectively, and then uses a machine learning algorithm to construct a detection model for ransomware detection. The experimental results show that REDMALS using the random forest algorithm achieves 97.7% and 96.0% accuracy on the constructed variant test set and unknown test set, respectively, and 83% and 76% of the ransomware samples in both test sets, respectively, can be detected before they perform any encryption behavior.

收稿日期:2023-11-06;修回日期:2024-03-13;责任编辑:覃怀银
*通讯作者:郭春

Key words: ransomware; early detection; API; TF-IDF; latent semantic analysis; random forest
Foundation Item(s): National Natural Science Foundation of China (No.62162009); Science and Technology Support Program of Guizhou Province (No.[2022]071); Big Data Security and Network Security Innovation Team of Guizhou Provincial High Education Institution (No.[2023]052); Key Technologies R&D Program of Henan Province (No.222102210048)

1　引言

勒索软件是一种破坏性很强的电脑病毒. 它们加密受害主机中文件使得用户无法正常使用文件,再以此勒索用户钱财. 据美国网络安全投资咨询公司Cybersecurity Ventures预测,2031年因勒索软件攻击损失总额将超过2 650亿美元[1]. 勒索攻击已经成为网络空间中最主要的安全威胁之一.

勒索软件主要分为ScreenLocker(锁屏型勒索软件)和Crypto-Ransomware(加密型勒索软件). 加密型勒索软件是当前勒索软件最主要的类型,占2019年勒索软件攻击的90%[2]. 本文以加密型勒索软件为研究对象,下文中"勒索软件"均指加密型勒索软件.

当前针对勒索软件的检测方法可大致分为基于静态分析的检测方法、动态检测方法和早期检测方法. 静态分析方法是在不运行恶意软件的情况下,利用从恶意软件源文件的原始字节[3]、操作码[4]、PE文件标头[5]等静态属性中提取的特征来分析恶意软件的方法. 基于静态分析的勒索软件检测方法具有无需运行勒索软件即可进行高效检测的优点,但其在检测采用了代码混淆、加壳等技术的勒索软件时效果不佳. 动态检测方法指提取软件在运行过程中的行为特征再加以分析的检测方法. 传统的勒索软件动态检测方法[6]重点关注如何获取高的检测精度. 随着深度学习的广泛应用,研究者提出了基于深度学习的勒索软件动态检测方法[7~9],这些方法将采集的软件行为数据通过改进的TextCNN[7]、卷积神经网络[8]或长短时记忆网络[9]等深度学习算法来训练检测模型以实施勒索软件检测. 但是上述传统的勒索软件动态检测方法通常需要收集较长时间的软件行为数据才能获取较高的检测精度. 为提升勒索软件检测的及时性,近年来一些研究者提出了基于第一条加密API的勒索软件早期检测方法[10,11]和基于时间阈值的勒索软件早期检测方法[12~14]. 基于第一条加密API的勒索软件早期检测方法以软件开始运行到其首次调用加密API之间的API序列来进行勒索软件检测. 但由于对采集到的API序列提取特征及检测均需要一定时间,该类方法无法在勒索软件开始执行加密行为(对应勒索软件运行后首次调用加密API)前将其检出. 基于时间阈值的早期检测方法通过设定行为采集时间阈值来减少软件行为采集时间. 由于不同勒索软件开始执行加密行为的时间各不相同,现有基于时间阈值的检测方法难以兼顾勒索软件检测的及时性和精度,表现为它们选用一个小的时间阈值时难以获取高的检测精度.

为进一步提升勒索软件检测的及时性,本文定义了一个"运行初始阶段"(Initial Phase of Operation, IPO),即从软件开始运行到其首次调用加密相关动态链接库(Dynamic-Link Library, DLL)之间的时间段. 针对勒索软件在所定义的IPO内调用的API序列相对较短的现实情况,我们使用了TF-IDF(Term Frequency-Inverse Document Frequency)和潜在语义分析(Latent Semantic Analysis, LSA)算法[15]对软件IPO内采集的API序列进行处理,构建了基于API潜在语义的勒索软件早期检测方法(Ransomware Early Detection Method based on API Latent Semantics, REDMALS),旨在兼顾勒索软件检测的及时性和准确率.

2　勒索软件早期行为分析

勒索软件的攻击过程通常包含如图1所示的7个步骤[10]:感染、安装、加密密钥生成、搜索文件、加密文件、勒索、支付勒索赎金. 由于被勒索软件所加密的文件不易恢复,因此在勒索软件开始加密文件前将其检出是勒索软件早期检测的最终目标.

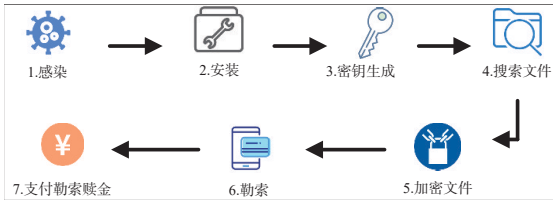


图1　勒索软件攻击流程

由于以勒索软件首次调用加密API作为行为采集的截止条件不能在勒索软件执行加密行为前将其检出,本文探索早于勒索软件首次调用加密API的软件行为来作为行为采集的截止条件. 程序调用Windows API的一般流程为:程序所需的DLL首先由DLL加载函数(如LoadLibrary)加载到内存中,接着程序通过API地址获取函数(如GetProcAddress)获取API的内存地址,然后执行API,最后当程序执行完毕后卸载DLL[16]. 作为程序的一种,勒索软件也需要通过DLL来调用API,所以我们推测勒索软件首次调用加密相关DLL(即用于调用Windows系统中加密API的DLL)的时刻早于其首次调用加密API的时刻.

为验证上述推测,我们实验分析了400款勒索软件

的加密过程,观察到这些勒索软件使用的加密相关DLL如表1所示,即这些勒索软件使用表1中一个或多个加密相关DLL调用加密API以加密文件.

表1 所分析的勒索软件使用的加密相关DLL

DLL名称	DLL描述
bcryptprimitives.dll	Windows加密原语库
cryptdll.dll	加密管理器
crypt32.dll	Windows加密API应用程序接口模块
advapi32.dll	提供高级服务的函数和功能,包括加密、安全、权限管理等
rsaenh.dll	Microsoft增强加密服务相关文件,用于128位加密
ncrypt.dll	Windows加密链接库
cryptsp.dll	含有Windows中很多加密API函数
cryptsvc.dll	提供加密服务功能
cryptext.dll	与文件加密外壳扩展相关的模块

我们还观察到勒索软件在首次调用加密API之前会进行一系列操作,例如部分勒索软件调用遍历资源API(如WNetOpenEnumW)来遍历受害者网络资源以为密钥交互作准备.部分勒索软件样本运行后首次调用加密相关DLL的API位次及其首次调用加密API的位次如表2所示.可看到这些勒索软件运行后首次调用加密相关DLL的时刻早于其首次调用加密API的时刻.

表2 部分勒索软件首次调用加密相关DLL和首次调用加密API的位次

勒索软件	首次调用加密相关DLL的位次	首次调用加密API的位次
alphacrypt	4 901	189 653
cerber	1 772	63 727
gandcrab	7 797	9 593
teslacrypt	1 074	15 982
lockbit	124	6 419
saturn	739	264 738
spora	10	5 225
sodinokibi	1 659	14 760
wannacry	1 196	36 056
mzrevenge	3 009	23 874

基于上述结果,我们认为勒索软件在运行后首次调用加密相关DLL的行为可视为其不久后将实施加密行为的提示.相应的,我们以勒索软件首次调用加密相关DLL的时刻来定位勒索软件的"预加密节点",将软件从开始运行到其预加密节点之间的时间段称为IPO.

定义1 运行初始阶段(IPO)勒索软件的IPO为勒索软件开始运行到首次调用加密相关DLL之间的时间段.

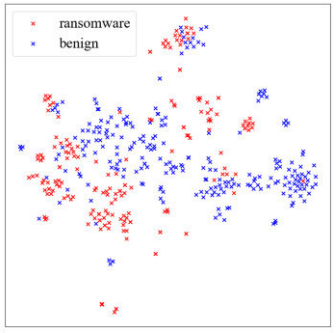
本文以软件IPO内调用的API序列作为判别勒索软件的分析对象.为避免良性软件和未知勒索软件不调用或运行很长时间后才调用表1中加密相关DLL而导致API采集时长过长的情况,本文在采集API序列时还加入一个时间阈值来共同作为采集截止条件.表3给出了部分勒索软件的IPO的时长,其显示所分析的大多数勒索软件的IPO时长在2 s以内.于是本文设定API的采集时间阈值为2 s,即某个软件首次调用表1中加密相关DLL的时刻在2 s以内则以该时刻作为API采集截止时刻,若该软件在2 s内未调用表1中加密相关DLL则API采集截止时刻为2 s.此时,本文的API序列采集时长(记为T)≤IPO时长.

表3 部分勒索软件IPO时长 单位:ms

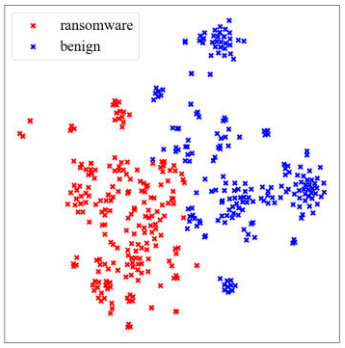
勒索软件家族(数量)	IPO时长(平均值)
chaos(12)	2 491
lorenz(12)	46
mzrevenge(8)	373
phobos(7)	262
dharma(10)	1 031
eleta(2)	54
kraken(2)	264
saturn(9)	108
sfile(9)	330
spora(10)	393
crysis(10)	1 059
hive(9)	2 192

由于勒索软件和良性软件在T内调用的API数量较少,以这些API来高准确率地检测勒索软件是一个挑战.本文分析了400款勒索软件和多款良性软件在T内调用的API序列的语义结构,发现勒索软件与良性软件的API序列存在多词一义问题.例如:在API序列中,NtCreateFile和CreateFileW均表示创建文件,RtlFreeHeap和HeapFree均表示释放空间.传统的向量空间模型(如TF-IDF模型)中会将例如NtCreateFile和CreateFileW等同义词表示为不同的单词向量,因此在同义词较多时计算出的TF-IDF向量的维度高.LSA可以将传统向量空间模型中相同语义的单词向量(例如NtCreateFile和CreateFileW)融合成一个单词向量,可以在不降低单词向量类别区分度的情况下减少特征向量的维数.

为了体现LSA对TF-IDF向量的降维效果,我们将单独使用TF-IDF生成的API特征向量以及使用TF-IDF和LSA生成的API特征向量用t分布随机近邻嵌入进行可视化,样本分布如图2所示.由图2可知,使用TF-IDF和LSA处理API序列所得到特征向量的类别区分度较单独使用TF-IDF处理API序列所得到的特征向量类别区分度高.



(a) 使用TF-IDF处理



(b) 使用TF-IDF+LSA处理

图2　不同方法处理API序列后的样本可视化

3　基于API潜在语义的勒索软件早期检测方法

本文提出的REDMALS以软件运行初期调用的API序列进行勒索软件检测. 如图3所示,REDMALS包括数据收集、特征计算、训练与检测三个阶段.

3.1　数据收集

如第2节所述,REDMALS采集API序列时,综合运用了软件首次调用加密相关DLL的时刻和时间阈值(2 s)作为采集截止条件. 具体来说,若样本首次调用加密相关DLL的时刻在开始运行后2 s内,则以该时刻作为采集截止时刻;若样本在运行后2 s内未调用加密相关DLL,则以2 s作为采集截止时刻. 之后将采集的API序列仅保留API名称,去除其调用地址、返回值、参数等信息.

3.2　特征计算

REDMALS在本阶段运用TF-IDF计算上阶段采集的API序列的TF-IDF值,再使用LSA对由TF-IDF值填充的矩阵进行降维. REDMALS对上阶段采集的API序列使用TF-IDF计算特征值. REDMALS使用式(1)计算API序列的TF值.

$$\mathrm{TF}\left(a_i,\boldsymbol{s}_j\right)=\frac{n\left(a_i,\boldsymbol{s}_j\right)}{\sum_k n\left(k,\boldsymbol{s}_j\right)} \tag{1}$$

其中,$\mathrm{TF}\left(a_i,\boldsymbol{s}_j\right)$表示API序列$\boldsymbol{s}_j$中API词条$a_i$出现的频率;$\sum_k n\left(k,\boldsymbol{s}_j\right)$表示API序列$\boldsymbol{s}_j$中所有API词条出现的总和;$n\left(a_i,\boldsymbol{s}_j\right)$表示API词条$a_i$在API序列$\boldsymbol{s}_j$中出现的次数. 计算出一个API词条的频率之后,通过式(2)计算其IDF值.

$$\mathrm{IDF}_{a_i}=\log_2\left(\frac{D}{d}\right) \tag{2}$$

其中,IDF_{a_i}描述的是某一特定API词条a_i在全部API序列中的重要程度. a_i越少,IDF_{a_i}越大,说明API词条a_i具有好的类别区分能力. D表示API序列总数. d表示包含API词条a_i的API序列的数量. 之后采用式(3)计算其TF-IDF值:

$$\text{TF-IDF}_{a_i\boldsymbol{s}_j}=\mathrm{TF}_{a_i\boldsymbol{s}_j}\times\mathrm{IDF}_{a_i} \tag{3}$$

$\text{TF-IDF}_{a_i\boldsymbol{s}_j}$表示API词条$a_i$在API序列$\boldsymbol{s}_j$上的TF-IDF值. 在得到每个API的TF-IDF值之后,用这些值生成如式(4)所示的TF-IDF矩阵$\boldsymbol{A}$.

$$\boldsymbol{A}=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{4}$$

$\boldsymbol{A}$中,x_{ij}表示词条a_i在序列$\boldsymbol{s}_j$的TF-IDF值. 接下来,REDMALS运用LSA对$\boldsymbol{A}$进行降维,具体步骤如下:以式(5)将$\boldsymbol{A}^{\mathrm{T}}$和$\boldsymbol{A}$做矩阵乘法来得到一个$n$阶方阵$\boldsymbol{W}$. 接下来,求解式(6)的特征方程,$x$为特征方程的解. 可得到$n$个特征值和$n$个特征向量,把这些特征向量组合起来可得到$n$阶正交矩阵$\boldsymbol{V}=\left[v_1\,v_2\,v_3\,\ldots\,v_n\right]$.

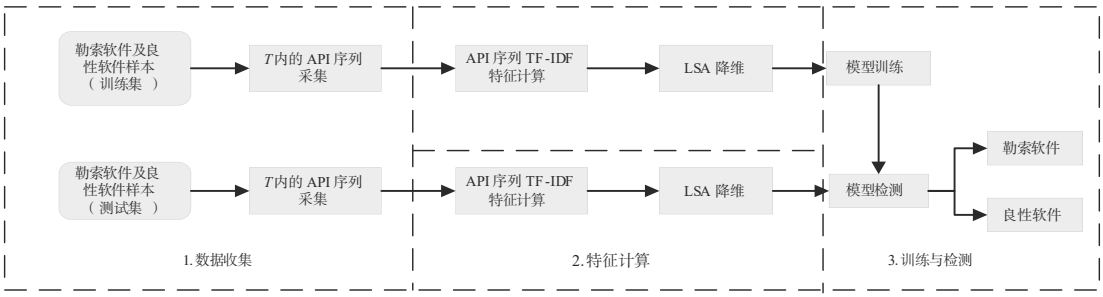


图3　REDMALS框架图

$$\boldsymbol{W}=\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} \tag{5}$$

$$(\boldsymbol{W}-\lambda\boldsymbol{I})x=0 \tag{6}$$

由于$\boldsymbol{W}$是n阶方阵,因此式(6)中的$\boldsymbol{I}$是n阶单位矩阵. 矩阵$\boldsymbol{V}$中的各特征向量均是$\boldsymbol{A}$的右奇异向量. 同理可通过式(7)求$\boldsymbol{A}$的左奇异矩阵$\boldsymbol{U}$.

$$\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}\Rightarrow\boldsymbol{U} \tag{7}$$

将$\boldsymbol{A}$和$\boldsymbol{A}^{\mathrm{T}}$做矩阵乘法可得到一个$m$阶的方阵$\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$,对$\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$进行特征分解可得到该方阵的$m$个特征值和$m$个特征向量. 把$m$个特征向量组合起来得到矩阵$\boldsymbol{U}=[u_1\,u_2\,u_3\ldots\,u_m]$,矩阵中各特征向量均是$\boldsymbol{A}$的左奇异向量. 对各特征值开根号得到相应的奇异值,再将奇异值按值大小降序排列并选择前k个奇异值排列在对角线上得到矩阵$\boldsymbol{\Sigma}_k$. 如式(8),$\boldsymbol{A}$近似等于降维后的矩阵$\boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^{\mathrm{T}}$.

$$\begin{aligned}\boldsymbol{A}&\approx\boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^{\mathrm{T}}\\&=[u_1\,u_2\,u_3\ldots\,u_m]\begin{bmatrix}\sigma_1&0&0&0\\0&\sigma_2&0&0\\0&0&\ddots&0\\0&0&0&\sigma_k\end{bmatrix}\begin{bmatrix}\boldsymbol{v}_1^{\mathrm{T}}\\\boldsymbol{v}_2^{\mathrm{T}}\\\vdots\\\boldsymbol{v}_k^{\mathrm{T}}\end{bmatrix}\end{aligned} \tag{8}$$

式(8)中,$\boldsymbol{\Sigma}_k$除了主对角线上的元素以外其他元素全为0. 最后,我们将$\boldsymbol{\Sigma}_k\boldsymbol{V}_k^{\mathrm{T}}$和对矩阵$\boldsymbol{A}$预测所得的类别概率矩阵$\boldsymbol{B}$进行拼接得到矩阵$\boldsymbol{Q}$.

3.3 训练与检测

REDMALS在本阶段构建勒索软件检测模型,用于区分勒索软件和良性软件. 首先将上一阶段提取到的特征向量添加类别标签(0对应良性软件,1对应勒索软件),得到训练集S_{tr}($S_{tr}=\{(Q_k,Y_k)|k\in\mathbf{N}^*\}$). 其中,$Q_k$为样本特征向量,$Y_k\in\{0,1\}$,$k$为样本总数. 之后运用分类算法在训练集$S_{tr}$上训练勒索软件检测模型. 在检测时,使用训练好的检测模型判别待测软件为勒索软件还是良性软件.

4 实验及结果

4.1 实验环境

本文在VMware16.2.2中进行实验环境(Windows 7和Windows 10)搭建,在环境中放置了十余种不同类型的文件(doc、txt、zip等),使用API monitor捕获软件运行时所调用的API信息. 检测勒索软件所用的主机硬件配置为Intel i7-10700F CPU,16 GB内存. 编程语言采用python 3.6.5,使用scikit-learn1.0.2库进行检测模型训练.

4.2 实验数据集

鉴于目前学术界尚无公开的勒索软件数据集,本文从Any.run、VirusShare、Vx-underground等平台收集勒索软件样本,从360软件管家收集良性软件样本. 所收集样本包括32个家族的400款勒索软件以及8个类别的458款良性软件. 为让检测任务具备现实性和复杂性,我们把实验样本划分成表4和表5所示的训练集、变种测试集以及未知测试集. 此外,为测试REDMALS对运行在不同操作系统版本中勒索软件的检测能力,未知测试集中dharma、eleta、silentspring、phobos、mzrevenge、lorenz、yanluowang七个家族的勒索软件的API序列在Windows 10中采集,其余样本的API序列在Windows 7中采集.

表4 勒索软件数据集划分

家族	训练集	变种测试集	未知测试集
alphacrypt	5	2	0
cerber	9	4	0
crysis	16	10	0
gandcrab	20	12	0
lockbit	30	15	0
maze	15	4	0
teslacrypt	14	3	0
troldesh	14	7	0
sage	10	7	0
ryuk	15	8	0
sodinokibi	10	3	0
wannacry	15	11	0
conti	10	4	0
hive	17	9	0
polyransom	0	0	1
frs	0	0	1
kraken	0	0	2
lockergoga	0	0	3
prolock	0	0	3
spora	0	0	10
chaos	0	0	12
cryptoshield	0	0	2
saturn	0	0	9
cryptowire	0	0	6
sfile	0	0	9
dharma	0	0	10
eleta	0	0	2
silentspring	0	0	2
phobos	0	0	7
mzrevenge	0	0	8
lorenz	0	0	12
yanluowang	0	0	2

4.3 评估指标

本文使用恶意软件检测领域常用的准确率(ACCuracy,ACC)、精确率(Precision,P)、召回率(Recall,R)、F_1-score(F_1)作为方法的评价指标[11]. 此外,为评价方法能否在勒索软件实施文件加密行为前将其检出,本

表5　良性软件数据集划分

类别	训练集	变种测试集	未知测试集
加密软件	35	22	14
办公软件	30	20	12
聊天通讯	20	9	9
杀毒软件	20	4	8
视频软件	30	11	8
其他软件	35	18	23
系统工具	45	18	17
游戏	25	15	10

文除了使用检出时间(指判别测试集中单个样本是勒索软件还是良性软件所需要的最长时间,为序列采集时长与检测时长之和)之外,还定义了一个评价指标——成功预防率(Successful Prevention Rate,SPR). SPR指被正确预测且检出时刻在勒索软件首次调用加密API之前的勒索软件的数量(记为M)占测试集中勒索软件数量的百分比,由式(9)计算. TP代表勒索软件被正确预测的数量,FN代表勒索软件被错误判别的数量. SPR值越高代表及时检测勒索软件的能力越强.

$$\mathrm{SPR}=\frac{M}{\mathrm{TP}+\mathrm{FN}} \tag{9}$$

4.4　实验结果

本文给出了REDMALS使用K-Nearest Neighbor(KNN)、Multilayer Perceptron(MLP)、Gradient Boosting Decision Tree(GBDT)、Adaptive Boosting(AdaBoost)、Random Forest(RF)、Light Gradient Boosting Machine(LightGBM)、Decision Tree(DT)七种分类算法所得的检测结果. 各算法的参数如表6所示,表中未提及的算法参数采用默认参数. 为评估REDMALS运用LSA是否带来检测效果提升,我们还给出了未使用LSA(即仅使用TF-IDF处理API序列)的方法的检测结果.

表6　算法参数

算法	参数
KNN	n_neighbors=5
MLP	solver='adam',alpha=1×10^{-5},activation='tanh'
RF	n_estimators=40,criterion='gini'
DT	criterion='gini',max_depth=50

4.4.1　勒索软件变种检测结果

表7给出了REDMALS使用不同分类算法在变种测试集上的检测结果. 表7显示REDMALS使用RF算法得到的ACC、R、F_1值优于其使用其他分类算法. 在检出时间方面,REDMALS使用表中不同分类算法所需的检出时间相差不大,其检测变种测试集中单个样本最多仅需2.156 s. 在实验中,REDMALS使用RF算法时能将83%的勒索软件变种样本在其首次调用加密API前检出,这些样本的检出时刻与其首次调用加密API时刻的差值区间为[-80.319 s,-125 ms]. 由于在Windows系统中仅需约10 ms即可终止一个进程,通过进程终止可在上述样本执行加密行为前将其终止.

表7　REDMALS使用不同分类算法在变种测试集上的检测结果

算法	ACC	R	P	F_1	检出时间/s
KNN	0.917	0.859	0.955	0.904	2.104
DT	0.958	0.960	0.950	0.955	2.078
MLP	0.944	0.919	0.958	0.938	2.063
GBDT	0.958	0.970	0.941	0.955	2.156
AdaBoost	0.977	0.980	0.970	0.975	2.097
RF	0.977	0.990	0.960	0.975	2.094
LightGBM	0.954	0.990	0.916	0.951	2.104

为进一步评估REDMALS检测勒索软件变种的准确率和及时性,本文将其与文献[11]方法(使用RF)、文献[14]方法(使用RF)、文献[7]方法(使用改进的TextCNN)在变种测试集上进行了对比实验,结果如表8所示. 文献[7]方法未明确给出其API序列采集时长,在对比实验中该时长设置为1 s及30 s.

表8　不同勒索软件检测方法在变种测试集上的检测结果

方法	ACC	检出时间/s	SPR
RF[11]	0.949	71.31	0
RF[14]	0.940	7.112	0.25
TextCNN(1s)[7]	0.838	1.088	0.56
TextCNN(30s)[7]	0.949	30.062	0.08
REDMALS(仅TF-IDF)	0.949	2.133	0.61
REDMALS	0.977	2.094	0.83

由表8可知,REDMALS能够以2.094 s的检出时间在变种测试集上获得97.7%的ACC值,在ACC值上优于其他对比方法,在检出时间上仅略逊于以1 s为采集时长的文献[7]方法. 值得一提的是REDMALS在变种测试集上获得了83%的SPR值,优于其他对比方法. 文献[11]方法使用勒索软件从开始运行到首次调用加密API之间的API序列作为分析对象,其方法的检出时间均在勒索软件首次调用加密API之后(即SPR值为0);文献[7]方法和文献[14]方法均采用固定时间阈值方法,由于勒索软件从开始运行到开始执行加密行为之间的时间很短,基于固定时间阈值的方法若设定的行为采集时长过大(如将文献[7]方法采集时长设置为30 s)虽有助于获取高的ACC值,但会导致SPR值很低,设定的采集时长过短(如将文献[7]方法采集时长设置为1 s)会导致方法的ACC低;仅使用TF-IDF的REDMALS未利用API之间潜在的语义结构,其ACC低于使用TF-IDF和LSA的REDMALS.

4.4.2　未知勒索软件检测结果

表9给出了REDMALS使用不同分类算法在未知

测试集上的检测结果. 表9显示REDMALS使用RF算法在未知测试集上获得的ACC、R、F_1值优于其使用其他分类算法所获得的相应值. 在实验中,REDMALS使用RF算法时能将76%的未知勒索软件样本在其首次调用加密API前检出,这些样本的检出时刻与其首次调用加密API时刻的差值区间为[−60.145 s, −202 ms],足够采用进程终止方式来终止这些样本的进程.

表9 REDMALS使用不同分类算法在未知测试集上的检测结果

算法	ACC	R	P	F_1	检出时间/s
KNN	0.807	0.624	0.984	0.764	2.130
DT	0.896	0.930	0.870	0.900	2.125
MLP	0.886	0.802	0.964	0.876	2.103
GBDT	0.951	0.980	0.925	0.952	2.107
AdaBoost	0.960	0.960	0.960	0.960	2.222
RF	0.960	0.990	0.934	0.962	2.115
LightGBM	0.946	0.980	0.917	0.947	2.135

为了进一步评估REDMALS检测未知勒索软件的准确率和及时性,本文将其与文献[11]方法、文献[14]方法、文献[7]方法在未知测试集上进行了对比实验,结果如表10所示.

表10 不同勒索软件检测方法对未知测试集的检测结果

方法	ACC	检出时间/s	SPR
RF[11]	0.896	62.020	0
RF[14]	0.851	7.111	0.48
TextCNN(1 s)[7]	0.807	1.112	0.55
TextCNN(30 s)[7]	0.832	30.105	0.01
REDMALS(仅TF-IDF)	0.866	2.135	0.56
REDMALS	0.960	2.107	0.76

表10显示以1 s为采集时长的文献[7]方法在未知测试集上的检出时间优于其他方法,但行为采集时长过短使得其ACC值仅有80.7%. REDMALS在检出时间上略逊于以1 s为采集时长的文献[7]方法,但REDMALS在ACC和SPR两个指标上均优于其他对比方法.

5 结论

本文通过分析多款勒索软件运行初期调用DLL和API的行为,提出了一个对勒索软件早期检测领域具有重要意义的概念——IPO. 以IPO内的软件行为为分析对象有利于在勒索软件执行加密行为前即开始对其进行检测. 在此基础上,本文提出了一种利用IPO内的API序列进行勒索软件检测的方法REDMALS. 实验结果表明,REDMALS使用RF算法时在变种测试集和未知测试集上可获得较高的ACC以及SPR值. 在后续工作中,我们将致力于研究如何进一步提升检测方法的SPR值以及集成合适的响应措施.

参考文献

[1] BRAUE D. Global ransomware damage costs predicted to exceed $265 billion by 2031[EB/OL]. (2021-06-03)[2024-01-03]. https://cybersecurityventures. com/ransomware-report-2021/.

[2] MCINTOSH T, KAYES A S M, CHEN Y P P, et al. Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions[J]. ACM Computing Surveys, 2021, 54(9): 197.

[3] KHAMMAS B M. Ransomware detection using random forest technique[J]. ICT Express, 2020, 6(4): 325-331.

[4] ZHANG B, XIAO W T, XIAO X, et al. Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes[J]. Future Generation Computer Systems, 2020, 110: 708-720.

[5] DENG X Z, CEN M C, JIANG M, et al. Ransomware early detection using deep reinforcement learning on portable executable header[J/OL]. Cluster Computing, 2023. https://doi.org/10.1007/s10586-023-04043-5.

[6] JETHVA B, TRAORÉ I, GHALEB A, et al. Multilayer ransomware detection using grouped registry key operations, file entropy and file signature monitoring[J]. Journal of Computer Security, 2020, 28(3): 337-373.

[7] QIN B, WANG Y L, MA C C. API call based ransomware dynamic detection approach using TextCNN[C]//2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE). Piscataway: IEEE, 2020: 162-166.

[8] GULMEZ S, GORGULU KAKISIM A, SOGUKPINAR I. XRan: Explainable deep learning-based ransomware detection using dynamic analysis[J]. Computers & Security, 2024, 139: 103703.

[9] 刘文静, 郭春, 申国伟, 等. 基于深度学习的勒索软件早期检测方法[J]. 计算机科学, 2023, 50(3): 391-398.
LIU W J, GUO C, SHEN G W, et al. Ransomware early detection method based on deep learning[J]. Computer Science, 2023, 50(3): 391-398. (in Chinese)

[10] ALI SALEH AL-RIMY B, 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.

[11] KOK S H, ABDULLAH A, JHANJHI N Z. Early detection of crypto-ransomware using pre-encryption detection algorithm[J]. Journal of King Saud University - Computer and Information Sciences, 2022, 34(5): 1984-1999.

[12] HOMAYOUN S, DEHGHANTANHA A, AHMADZADEH M, et al. Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence[J]. IEEE Transactions on Emerging Topics in Computing, 2020, 8(2): 341-351.

[13] RHODE M, BURNAP P, JONES K. Early-stage malware prediction using recurrent neural networks[J]. Computers and Security, 2018, 77(C): 578-594.

[14] 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. Ransomware early detection method based on short API sequence[J]. Acta Electronica Sinica, 2021, 49(3): 586-595. (in Chinese)

[15] BELLEGARDA J R. Exploiting latent semantic information in statistical language modeling[J]. Proceedings of the IEEE, 2000, 88(8): 1279-1296.

[16] AMER E, ZELINKA I. A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence[J]. Computers & Security, 2020, 92: 101760.

作者简介

罗　斌　男,1999年生,贵州毕节人. 贵州大学计算机科学与技术学院硕士研究生. 主要研究方向为计算机网络与信息安全.
E-mail: gzu_bin@163.com

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教授. 主要研究领域为恶意代码检测、入侵检测.
E-mail: gc_gzedu@163.com