基于 API 短序列的勒索软件早期检测方法

陈长青,郭 春,崔允贺,申国伟,蒋朝惠
(贵州省公共大数据重点实验室,贵州大学计算机科学与技术学院,贵州贵阳 550025)

摘 要: 传统的勒索软件动态检测方法需要收集较长时间的软件行为,难以满足勒索软件及时检测的需求. 本文从勒索软件及时检测的角度出发,提出了"勒索软件检测关键时间段(Critical Time Periods for Ransomware Detection, CTP)"的概念,并基于 CTP 的要求提出了一种基于应用程序编程接口(Application Programming Interface, API)短序列的勒索软件早期检测方法(Ransomware Early Detection Method based on short API Sequence, REDMS). REDMS 以软件在 CTP 内执行时所调用的 API 短序列为分析对象,通过 *n*-gram 模型和词频-逆文档频率算法对采集到的 API 短序列进行计算以生成特征向量,然后运用机器学习算法建立检测模型对勒索软件进行早期检测. 实验结果显示,REDMS 在 API 采集时段为前 7s 且使用随机森林算法时,分别能以 98.2% 、96.7% 的准确率检测出已知和未知的勒索软件样本.

关键词: 勒索软件; 早期检测; 机器学习; 应用程序编程接口
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112 (2021)03-0586-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20200623

Ransomware Early Detection Method Based on Short API Sequence

CHEN Chang-qing, CUO Chun, CUI Yun-he, SHEN Guo-wei, JIANG Chao-hui
(*Guizhou Provincial Key Laboratory of Public Big Data, School of Computer Science and Technology, Guizhou University, Guiyang, Guizhou* 550025, *China*)

Abstract: Traditional ransomware dynamic detection methods need to collect software behaviors for a long time, which is difficult to meet the need for timely detection of ransomware. From the perspective of the timely detection of ransomware, this article proposes a concept named "Critical Time Periods for Ransomware Detection (CTP)", and proposes an early ransomware detection method based on short application programming interface (API) sequence (REDMS) to fit the requirement of CTP. REDMS takes the short API sequences that are obtained by software running during the CTP as the analysis object, and calculates these short API sequences through the *n*-gram model and the term frequency-inverse document frequency algorithm to generate the feature vectors, and then uses a machine-learning algorithm to build a detection model for detecting ransomware. The experimental results show that when the first 7 seconds of API collection period and random forest algorithm are used, REDMS achieves 98.2% and 96.7% accuracy respectively for detecting the known and unknown ransomware samples.

Key words: ransomware; early detection; machine learning; API (Application Programming Interface)

1 引言

近年来,随着数字经济的快速发展,针对高价值资产的勒索软件逐渐增加,成为当前网络空间安全面临的重大威胁之一. 勒索软件起源于 1989 年的"AIDS"病毒,是一类通过封锁目标系统资源或加密目标系统文件以获取高额赎金的恶意软件[1]. 2019 年的赛门铁克网络威胁情报显示,2018 年全球企业的勒索软件感染率相比 2017 年增长了 12%[2].

当前常见的勒索软件可分为锁屏型和加密型两种类型. 锁屏型勒索软件感染受害者主机后会锁定屏幕或操作系统;加密型勒索软件会加密被感染主机的文件及文件夹[3]. 目前公开的勒索软件中,加密型勒索软件是最常见的类型[3~6]. 因此,本文以加密型勒索软件为研究对象,下文无特指时所提及的"勒索软件"均为

收稿日期:2020-06-28;修回日期:2020-12-01;责任编辑:马兰英
基金项目:国家自然科学基金(No. 61802081);贵州省科学技术基金(No. 黔科合基础[2017]1051);贵州省科技计划项目(No. 黔科合重大专项字[2018]3001);贵州省公共大数据重点实验室开放课题(No. 2017BDKFJJ025)

加密型勒索软件.

勒索软件与传统的窃密类恶意软件(如木马、间谍软件等)的行为表现具有较大的差别. 勒索软件感染主机后会尽快地加密被感染主机中文件,故在短时间内勒索软件会执行大量的操作[4],而窃密类恶意软件具有长期潜伏的需求,其在感染主机后初期的一段时间内执行的操作相对较少[7]. 勒索软件一般在较短时间内就能完成对被感染主机全部特定文件的加密操作,而传统的恶意软件动态检测方法往往需要收集较长时间的软件行为,难以满足勒索软件及时检测的现实需求.

针对上述问题,本文通过分析多个不同家族的勒索软件运行过程,从勒索软件及时检测的角度提出了"勒索软件检测关键时间段"(CTP)的概念,并进一步提出了一种基于应用程序编程接口(API)短序列的勒索软件早期检测方法(REDMS). 为了与软件长时间运行后所产生的 API 序列进行区分,本文将在 CTP 内采集到的 API 序列称为"API 短序列". REDMS 以软件的 API 短序列为分析对象,通过 n-gram[8] 和词频-逆文档频率(Term Frequency-Inverse Document Frequency, TF-IDF)[9] 生成特征向量,再运用机器学习算法建立检测模型来实现对勒索软件的早期检测.

本文主要工作如下:

(1)分析多个不同家族的勒索软件和多种类型的正常软件的运行过程,从勒索软件及时检测的角度提出 CTP 的概念,明确了勒索软件早期检测需要满足的时间要求;

(2)基于检测时间少于 CTP 的要求提出一种勒索软件早期检测方法 REDMS. 该方法采集各软件在 CTP 内调用的 API 序列,运用 n-gram、TF-IDF 和分类算法对勒索软件进行早期检测;

(3)利用来自 24 个不同家族的 78 个勒索软件以及 40 个多种类型的正常软件对 REDMS 进行实验测试. 实验结果表明,REDMS 能够在软件开始运行后数秒内高准确率地检测出已知和未知的勒索软件样本.

2　相关工作

由于勒索软件具有很大的危害性,近年来国内外研究者提出了一系列勒索软件检测方法. 本文将这些检测方法分为三类,即静态检测方法、动态检测方法以及早期检测方法.

2.1　静态检测方法

在不运行程序的情况下,静态检测方法运用静态分析技术分析程序的源代码或反汇编代码,从中提取特征来检测恶意软件[10]. Zhang 等人[11] 运用反汇编技术提取勒索软件的操作码序列作为特征,通过随机森林算法(Random Forests, RF)[12] 建立检测模型,其多分类准确率为 91.4%;Guo 等人[13] 提出一种基于可视化的勒索软件检测及分类方法,该方法对勒索软件的分类准确率可达到 96.7%;Sharma 等人[14] 提出了一种基于纹理的勒索软件分类方法,通过本地二进制模式算法提取勒索软件纹理实现分类. 总体来说,静态检测方法实施检测时无需运行软件,但是此类方法难以应对使用了混淆技术的勒索软件[15].

2.2　动态检测方法

动态检测方法通常在可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[16]. Hampton 等人[17] 通过 Windows 平台分析了 14 个不同勒索软件家族样本运行后的 API 调用情况,给出了 API 调用能够用于识别勒索软件的观点. Kharraz 等人[18] 提出了一种勒索软件动态分析系统,该系统通过监测 I/O 数据缓冲区的熵值变化、文件访问模式以及文件系统活动来实现对勒索软件的检测. Vinayakumar 等人[19] 在沙箱中运行了 7 个勒索软件家族的多个样本,采集了 131 个 API 作为多层感知机(Multilayer Perceptron, MLP)[20] 的输入以实现勒索软件分类,实验结果表明该方法的多分类准确率达到 98%,但其需要从日志文件中获取软件长时间运行后的 API 调用,难以满足勒索软件及时检测的需求. Feng 等人[21] 提出利用诱饵文件是否被加密来检测勒索软件,其方法中诱饵文件设定为难以被正常用户操作的文件. 由于不同勒索软件在文件的加密顺序上没有固定规律,该方法难以适用于所有勒索软件的快速检测. Kok 等人[5] 提出一种关注潜在的勒索软件预加密行为的检测方法,具有较高的实用性,但该方法的误报率较高.

2.3　早期检测方法

由于勒索软件感染主机后即会快速加密其文件,早期检测对于被感染主机的文件保护和及时防御具有重大意义. Scaife 等人[22] 提出一种早期检测系统 CryptoDrop,该系统通过检测文件类型以及文件熵的变化来快速检测勒索软件,但是文章未给出其检测出勒索软件所需的具体时长. Morato 等人[23] 通过网络共享卷中的 SMB 流量来对勒索软件进行早期检测,其方法分析 SMB 流量中勒索软件交互指令在一定时间内的变化速度以及共享卷中文件变化来判定是否有勒索软件,该方法达到最佳检测效果所需的流量监控时间为 20s.

由上述分析可知,大部分现有的动态检测方法在给出检测结果前需要收集较长时间的软件行为,难以满足勒索软件及时检测需求;而现有的勒索软件早期检测研究并未详细分析其所提方法的检测时间是否为真正意义上的"早期". 针对现有检测方法存在的不足,本文重点关注勒索软件运行初期的 API 调用,从勒索软

件及时检测的需求出发构建勒索软件早期检测方法.

3 勒索软件行为分析

勒索软件的攻击过程可分为系统感染、文件遍历和加密、索取赎金与文件解密四个阶段[3]. 系统感染阶段,勒索软件可通过网络钓鱼、漏洞利用等方式来感染主机;在文件遍历和加密阶段,勒索软件通常会执行一系列诸如文件查找及识别、文件加密等操作[6],这些操作一旦执行完成,被感染主机中全部特定文件都会被加密;索取赎金阶段通过勒索文本告知受害用户如何支付赎金;当受害用户支付赎金后,即进入文件解密阶段,攻击者释放解密器以解密被加密的文件. 本文给出了如图 1 所示的勒索软件的文件遍历和加密阶段的一般执行过程.

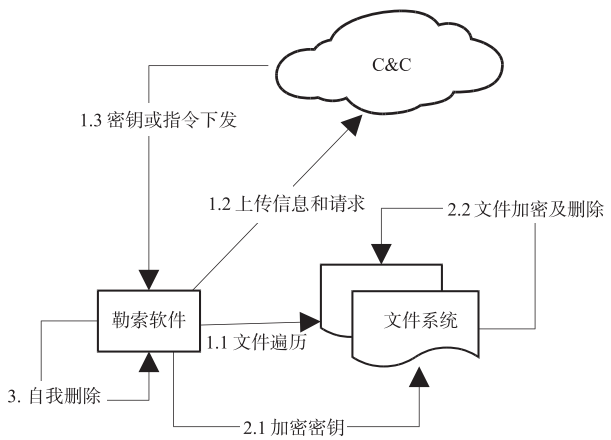

图1 勒索软件文件遍历和加密阶段一般执行过程

勒索软件在文件遍历和加密阶段的执行过程中,首先会进行信息收集(图 1 中 1.1 ~ 1.3),通过收集目标系统信息以及遍历其中的特定文件,并将获取的信息上传到 C&C 服务器以得到执行指令或加密秘钥;然后进入到文件加密步骤(图 1 中 2.1 ~ 2.2),勒索软件在该步骤会对系统中的特定文件进行加密并删除或覆盖原始文件;完成上述步骤,勒索软件会在被感染主机中生成勒索文本并删除自身. 由上述过程可以推测,处于该阶段的勒索软件会频繁操作被感染主机,并将大量调用与文件遍历和加密、删除等操作相关的 API.

为验证上述推测,本文在沙箱 ANY. RUN 上部署了以 Windows 7 为操作系统的运行环境,运行并分析了来自 24 个不同家族的 78 个勒索软件以及 40 个多种类型的正常软件开始运行后 60s 内的 API 调用情况,记录了各勒索软件在执行过程中弹出其勒索文本的时间点以及在该时间点前后的 API 调用数量,结果如表 1、表 2 和图 2 所示.

从表 1 可知,本文所分析的绝大部分勒索软件在开始运行后 60s 内弹出了勒索文本,意味着这些勒索软件在此期间已经完成了文件遍历和加密. 表 1 还显示绝大部分勒索软件在弹出其勒索文本前的 API 调用数量明显多于该文本弹出后到其开始运行到第 60s 这段时间内的 API 调用数量,印证了需要调用大量 API 的推测. 表 2 显示正常软件在开始运行后的一段时间内(前 10s 和前 60s)未大量调用 API. 因此,本文认为软件在其开始运行后一段时间内执行的 API 数量可以用于区分正常软件和勒索软件.

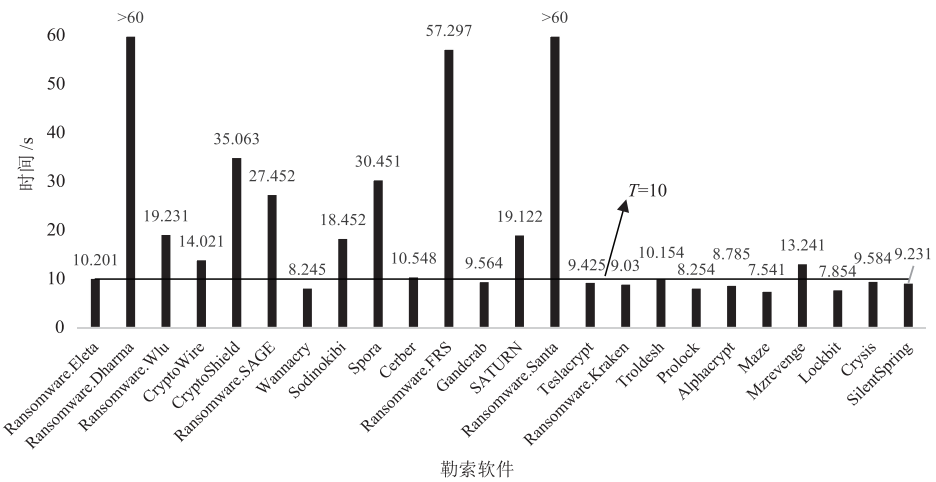

图2 不同勒索软件在本文实验中弹出勒索文本时间

定义 1 (勒索软件检测关键时间段 CTP) CTP 为勒索软件从开始运行到其在被感染主机弹出勒索文本之间的时间.

为满足勒索软件及时检测需求,检测方法需要在 CTP 内检测出勒索软件,才有较大可能保证此时被感染主机中文件未被全部加密. 图 2 显示本文所分析的半数以上勒索软件家族在其开始运行后 10s 左右弹出勒索文本. 因此,CTP 在本文中被设定为 10s.

表 1　勒索软件开始运行后 60s 内的 API 调用情况

勒索软件(数量)	主机弹出勒索文本时间(平均时间/s)	主机弹出勒索文本前的 API 数量(平均值)	主机弹出勒索文本后到软件运行第 60s 的 API 数量(平均值)
Ransomware. Eleta(2)	10. 201	101278	55421
Ransomware. Dharma(5)	未提示	88174	0
Ransomware. Wlu(2)	19. 231	42327	13625
CryptoWire(2)	14. 021	3911	125
CryptoShield(2)	35. 063	4220	3102
Ransomware. SAGE(2)	27. 452	3024	186
Wannacry(4)	8. 245	18522	6520
Sodinokibi(6)	18. 452	32879	22542
Spora(4)	30. 451	4302	216
Cerber(4)	10. 548	5299	2105
Ransomware. FRS(3)	57. 297	72061	52461
Gandcrab(6)	9. 564	4967	2532
SATURN(2)	19. 122	5785	4620
Ransomware. Santa(2)	未提示	4521	0
Teslacrypt(5)	9. 425	140077	52414
Ransomware. Kraken(2)	9. 030	4103	25
Troldesh	10. 154	95569	58712
Prolock(2)	8. 254	909995	25461
Alphacrypt(4)	8. 785	4369	152
Maze(3)	7. 541	7200	1452
Mzrevenge(4)	13. 241	3093	1532
Lockbit(2)	7. 854	126210	108214
Crysis(8)	9. 584	24307	15241
SilentSpring	9. 231	3777	3600

注:“未提示”表示 60s 内主机未弹出勒索文本

表 2　部分正常软件运行后 60s 内的 API 调用情况

正常软件	前 10s 的 API 数量	前 60s 的 API 数量
360 安全卫士	1476	2868
Notepad ++	1482	1602
CSDN	785	1237
迅雷	653	1005
PPTV	2844	3386

4　基于 API 短序列的勒索软件早期检测方法

针对检测时间少于 CTP 的要求,本文提出了一种基于 API 短序列的勒索软件检测方法 REDMS,框架如图 3 所示. 该方法重点关注软件开始运行后初期的 API 调用,通过 CTP 内各软件调用的 API 短序列判别该软件是否为勒索软件. REDMS 包含预处理、特征计算、训练和检测三个阶段. 下面分别对三个阶段进行详细阐述.

4. 1　预处理

REDMS 在预处理阶段采集勒索软件与正常软件 CTP 内的 API 短序列,并将其作为下一个阶段的输入. 表 3 给出了本文分析勒索软件运行时所统计的 24 个 API,包含文件系统 API、进程间通信 API 等.

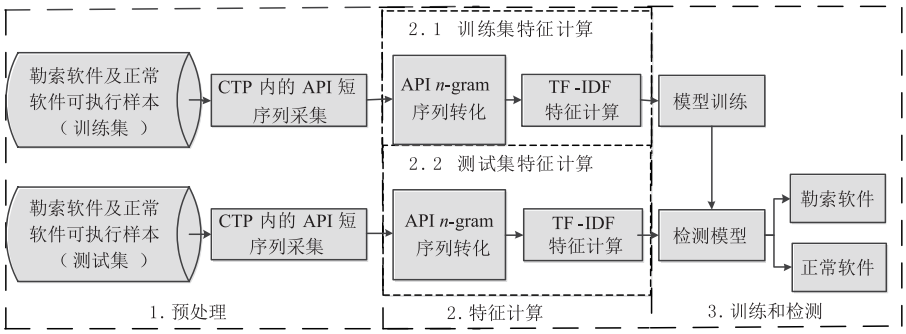

图3 REDMS执行框架

表 3 本文监控的 API

序号	API 名称	序号	API 名称	序号	API 名称	序号	API 名称
1	CreatFile	7	SetFileAttributes	13	CreatPipe	19	MoveFile
2	WriteFile	8	SetEndOfFile	14	SetVolumeLabel	20	GetdriverType
3	ReadFile	9	QueryDosDevice	15	GetCurrentDirectory	21	LCreat
4	LClose	10	FlushFileBuffers	16	GetSystemDirectory	22	LRead
5	DeleteFile	11	SetVolumeMountPiont	17	DeviceIoControl	23	HWrite
6	VerQueryValue	12	GetSystemPowerStatus	18	GetVolumeInformation	24	CreatMailslot

4.2 特征计算

REDMS 在本阶段运用 *n*-gram 算法分割 API 序列并通过 TF-IDF 算法计算出对应的特征值. *n*-gram 算法的运用使得在所生成的特征向量中保留了 API 序列中 API 的相邻关系.

通过 *n*-gram 算法处理 API 序列之后,REDMS 运用自然语言处理[24]中的 TF-IDF 算法来计算特征值以生成特征向量. REDMS 使用(1)计算勒索软件以及正常软件的 *n*-gram 序列的 TF 值.

$$\mathrm{TF}(i,j)=\frac{n(i,j)}{\sum_k n(k,j)} \tag{1}$$

其中 $\mathrm{TF}(i,j)$ 表示在 *n*-gram 序列 j 中 *n*-gram i 出现的频率,其中 $n(i,j)$ 表示 *n*-gram i 在 *n*-gram 序列 j 中出现的次数, $\sum_k n(k,j)$ 表示 *n*-gram 序列 j 中 *n*-gram 的总数. 在计算出一个 *n*-gram 的频率之后,通过式(2)计算其 IDF.

$$\mathrm{IDF}(i)=\log_2\frac{|D|}{|\{j:i\in j|j\in D\}|} \tag{2}$$

其中 $\mathrm{IDF}(i)$ 描述 *n*-gram i 在整个 *n*-gram 序列中的稀有程度,D 表示所有 *n*-gram 序列的集合. $|\{j:i\in j|j\in D\}|$ 表示包含 *n*-gram i 的 *n*-gram 序列 j 的数量. 当计算出 $\mathrm{IDF}(i)$ 之后,采用式(3)完成对 TF-IDF 的计算.

$$\mathrm{TF-IDF}(i,j)=\mathrm{TF}(i,j)\times\mathrm{IDF}(i) \tag{3}$$

$\mathrm{TF\text{-}IDF}(i,j)$ 表示 *n*-gram i 在 *n*-gram 序列 j 的 TF-IDF 值.

综上,REDMS 将软件开始运行后一个时间段内所采集的 API 序列视为一个整体,通过算法 1 得到相应的特征向量.

算法 1 基于 *n*-gram 的特征提取算法

输入:API 序列集 $A=\{A_1,A_2,\cdots,A_k\}$ //k 为样本数
过程:
1. for $j=1,2,\cdots,k$ do
2. 对 A_j 进行 *n*-gram 切片
3. 将切片序列保存到 An_j
4. end for
5. 计算所有出现的 *n*-gram 切片数 D
6. for $j=1,2,\cdots,k$ do
7. for $i=1,2,\cdots,\mathrm{len}(An_j)$ do
8. 根据式(1)计算 An_j 的 *n*-gram i 的频率 $\mathrm{TF}(i,j)$
9. 根据式(2)计算 An_j 的 *n*-gram i 的逆文本频率指数 $\mathrm{IDF}(i)$
10. 根据式(3)计算 An_j 的 $\mathrm{TF\text{-}IDF}(i,j)$
11. 将 $\mathrm{TF\text{-}IDF}(i,j)$ 添加到 $\mathrm{TF\text{-}IDT}_i$ 中
12. end for
13. 将 $\mathrm{TF\text{-}IDF}_i$ 添加到 X_j 中
14. end for
15. 输出特征向量矩阵 $\boldsymbol{X}=\{X_1,X_2,\cdots,X_k\}$

4.3 训练与检测

REDMS 在本阶段将基于特征计算阶段所得的特征向量构建一个勒索软件检测模型,用于区分勒索软件和正常软件. 通过将上一阶段提取的特征向量添加类别标签(正常软件和勒索软件分别标记为 -1 和 1),构建训练集 $S_{\mathrm{tr}}(S_{\mathrm{tr}}=\{(X_k,Y_k)|k\in N^*\}$,其中 $Y_k\in\{-1,1\}$,$X_k=(\mathrm{TF-IDF}(i,k)|i,k\in N^*))$. 之后,运用分类

算法结合训练集 S_{tr}建立一个勒索软件检测模型.

在检测过程中,训练好的检测模型将对测试集中各软件的特征向量进行检测,以判别该软件属于正常软件还是勒索软件.

5　实验及结果

5.1　实验环境和样本

本文在沙箱 ANY. RUN 搭建了 Windows 7 操作系统作为软件的运行环境,在桌面和磁盘分区中存放了数十个不同类型的文档. 检测勒索软件的主机硬件配置为 8G 内存,Intel i5 9600K 处理器. 编程语言采用 python3. 7 并使用 scikit-learn0. 20. 1 库进行检测模型训练. 实验所用的勒索软件样本采集于 virusshare(https://virusshare. com)以及 any. run (https://app. any. run),正常样本收集于 360 软件管家,实验样本种类和数量参见表 4,其中同一勒索样本家族中不同样本的版本不同,正常软件覆盖了杀毒软件、办公软件、聊天工具、下载工具、视频软件、浏览器、输入法等常用类型.

表 4　实验样本种类及数量

正常样本	数量	勒索样本家族	数量	勒索样本家族	数量	勒索样本家族	数量
安全杀毒	2	Ransomware. Eleta	2	Spora	4	Troldesh	1
办公软件	16	Ransomware. Dharma	5	Cerber	4	Prolock	2
聊天工具	2	Ransomware. Wlu	2	Ransomware. FRS	3	Alphacrypt	4
下载工具	4	CryptoWire	2	Gandcrab	6	Maze	3
视频软件	5	CryptoShield	2	SATURN	2	Mzrevenge	4
浏览器	2	Ransomware. SAGE	2	Ransomware. Santa	2	Lockbit	2
输入法	3	Wannacry	4	Teslacrypt	5	Crysis	8
其他应用	6	Ransomware. Kraken	2	Sodinokibi	6	SilentSpring	1
正常样本总数	40	勒索样本总数	78				

5.2　评估指标

本文使用恶意软件检测领域常用的准确率(Accuracy)、精确率(Precision,P)、召回率(Recall,R)以及 $F1$-score($F1$)作为 REDMS 的性能评估指标[14].

5.3　实验结果

在已知和未知的勒索软件测试集上,对比了 REDMS 分别运用支持向量机(Support Vector Machine,SVM)[12]、RF[12]、朴素贝叶斯(Naive Bayes,NB)[12]、K 临近(K-Nearest Neighbor,KNN)[25]、MLP[20]五种分类算法的检测效果,并进一步比较了 REDMS 在不同 n-gram 序列以及 CTP 内不同采集时段 T 的检测效果.

5.3.1　已知勒索软件样本检测

为测试 REDMS 对已知勒索软件的早期检测效果,本文将 78 个勒索软件及 40 个正常软件运行时调用的 API 短序列所计算出的特征向量作为训练集,将新采集的来自其中 24 个不同家族的 24 个勒索软件以及 16 个来自 8 种类型的正常软件在 CTP 内所调用的 API 短序列所计算出的特征向量作为测试集.

实验首先对比了 REDMS 运用不同的 n-gram 在 CTP 内四个不同采集时段 T 的检测效果. 图 4 为 REDMS 在不同 T 下使用 2-gram 及不同分类算法的准确率.

如图 4 所示,在使用 2-gram 划分序列的情况下,当 T 为前 2s 时 REDMS 使用 NB 和 SVM 的准确率仅有 77. 5%,而使用 RF 获得的准确率最高,可达 90%;当 T 为前 5s 时,REDMS 使用本文所提及五种算法的准确率均有上升. 图 4 还给出了 T 为前 7s 时的准确率,此时 REDMS 使用 RF 达到了 2-gram 划分下的最高准确率(95. 5%),而 T 为前 10s 时 REDMS 使用五种算法获得的准确率相比前 7s 时都有所下降,原因将结合 3-gram 和 4-gram 的结果在下文给出.

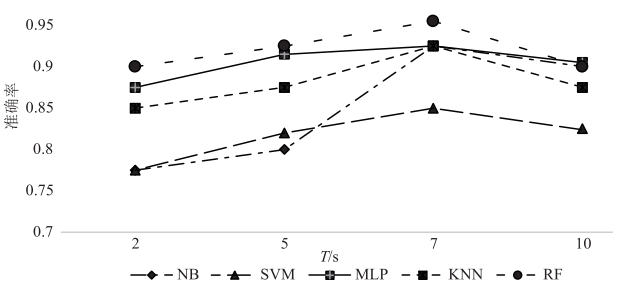

图4　REDMS在不同T下使用2-gram及不同算法的准确率

表 5 为 REDMS 在不同 T 下使用 2-gram 时五种算法获得的 P、R 以及 $F1$ 值. 由表 5 可知,在使用 2-gram 划分序列的情况下,当 T 为前 2s 时,REDMS 使用 RF 算法获得的 $F1$ 值最高但其值仅有 90. 1%. 这是由于勒索软件和正常软件的 2-gram 序列中有较多的相同部分,使得两者的特征向量区别不够明显.

表 5 REDMS 在不同 T 下使用 2-gram 及五种算法的 P、R 以及 $F1$

算法 \ T(s)	2			5			7			10		
	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$
NB	0.801	0.775	0.758	0.800	0.800	0.800	0.925	0.925	0.925	0.900	0.900	0.900
SVM	0.804	0.775	0.777	0.838	0.825	0.827	0.857	0.850	0.851	0.831	0.825	0.824
MLP	0.905	0.875	0.876	0.925	0.920	0.919	0.937	0.925	0.926	0.909	0.905	0.904
KNN	0.891	0.850	0.851	0.877	0.875	0.876	0.937	0.925	0.926	0.882	0.878	0.877
RF	**0.920**	**0.900**	**0.901**	**0.937**	**0.925**	**0.926**	**0.962**	**0.956**	**0.956**	**0.915**	**0.900**	**0.901**

在图 5 中给出了 REDMS 在不同 T 下使用 3-gram 及五种算法获得的准确率，当 T 为前 2s 时 REDMS 使用 MLP 能够达到 92.5% 的准确率，在 T 为前 5s 时 REDMS 使用 MLP 以及 RF 能够达到 95% 的准确率，而在 T 为前 7s 时使用 MLP 和 RF 分别能达到 96.2% 和 97.5% 的准确率.

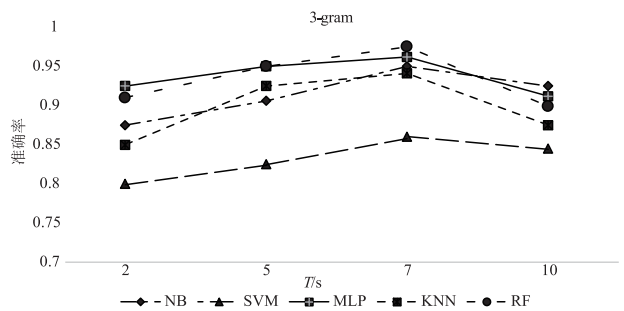

图5 REDMS在不同T下使用3-gram及五种算法的准确率

表 6 给出了 REDMS 在不同 T 下使用 3-gram 及五种算法的 P、R 以及 $F1$ 值. 由表 6 可知，在使用 3-gram 划分序列的情况下，当 T 为前 2s 时，REDMS 使用 MLP 所获得的 P、R 和 $F1$ 值高于使用其余四种算法时所得到的结果；当 T 为前 7s 时，REDMS 使用 RF 获得最高的 P、R 和 $F1$ 值.

表 6 REDMS 在不同 T 下使用 3-gram 及五种算法的 P、R 以及 $F1$

算法 \ T(s)	2			5			7			10		
	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$
NB	0.905	0.875	0.876	0.915	0.905	0.906	0.956	0.950	0.950	**0.932**	**0.925**	**0.924**
SVM	0.820	0.800	0.802	0.838	0.825	0.827	0.867	0.860	0.861	0.855	0.845	0.845
MLP	**0.937**	**0.925**	**0.926**	**0.956**	**0.950**	0.950	0.967	0.962	0.961	0.920	0.912	0.913
KNN	0.891	0.850	0.851	0.937	0.925	0.926	0.945	0.941	0.940	0.900	0.877	0.877
RF	0.929	0.917	0.919	**0.956**	**0.950**	**0.950**	**0.976**	**0.975**	**0.975**	0.910	0.899	0.900

图 6 给出了 REDMS 在不同 T 下使用 4-gram 及五种算法获得的准确率. 在使用 4-gram 划分序列的情况下，当 T 为前 2s 时，REDMS 使用 MLP 的准确率为 94.2%，当 T 为前 5s 时，REDMS 使用 RF 的准确率上升到 95%，当 T 为前 7s 时，REDMS 使用 RF 的准确率进一步上升到 98.2%，而其使用 MLP 也达到了 97.5% 的准确率.

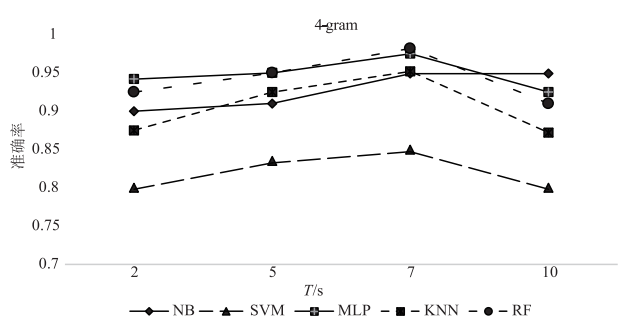

图6 REDMS在不同T下使用4-gram及五种算法获得的准确率

表 7 给出了 REDMS 在不同 T 下使用 4-gram 及五种算法的 P、R 以及 $F1$ 值. 由表 5 ~ 表 7 的结果可知，在本文所提及的五种分类算法中，REDMS 使用 RF 和 MLP 所获得的检测结果优于其他三种算法.

REDMS 在不同 n-gram 下的准确率如图 7 中所示. 由图 7 可知，当使用 MLP 和 RF 时，REDMS 在 T 为前 2s、前 7s 和前 10s 下使用不同 n-gram 所得到的准确率均满足 4-gram > 3-gram > 2-gram. 由表 5 ~ 表 7 的结果可知，REDMS 使用 4-gram 时所获得的 P、R 以及 $F1$ 值总体优于使用 2-gram 及 3-gram 所获得的结果. 经分析发现，出现该结果的原因是勒索软件的一次文件操作经常需要调用四个 API，如快速创建文件需要调用(CreatFile，WriteFile，HWrite，LClose)等四个 API.

从文件保护的角度看，及早检测到勒索软件对保护被感染主机中的文件具有重要意义. 由表 5 ~ 表 7 的结果可知，在 CTP 内分别取 T 为前 2s、前 5s、前 7s 和前 10s 时，REDMS 使用五种分类算法均在 T 为前 7s 时获

得最佳的检测结果,并且在各采集时段上所获得的检测结果基本满足 7s > 5s > 2s 的关系. 当 T 为前 10s 时,REDMS 使用五种分类算法获得的准确率均不是很高. 经分析发现,这是因为勒索软件在完成文件加密并弹出勒索文本之后,其所执行的操作及对应的 API 调用数量将大大减少,而本文实验中有部分勒索软件样本在 10s 前已经完成了文件加密及弹出勒索文本.

表 7　REDMS 在不同 T 下使用 4-gram 及五种算法获得的 P、R 以及 $F1$

T(s) / 算法	2			5			7			10		
	P	R	$F1$	P	R	$F1$	P	R	$F1$	P	R	$F1$
NB	0.920	0.900	0.901	0.926	0.908	0.909	0.952	0.946	0.946	**0.952**	**0.946**	**0.946**
SVM	0.820	0.800	0.802	0.845	0.844	0.845	0.857	0.850	0.851	0.810	0.800	0.800
MLP	**0.944**	**0.942**	**0.942**	**0.956**	**0.950**	0.950	0.971	0.979	0.975	0.935	0.925	0.926
KNN	0.905	0.876	0.875	0.921	0.938	0.924	0.956	0.952	0.951	0.895	0.872	0.873
RF	0.937	0.925	0.926	**0.956**	**0.950**	**0.950**	**0.986**	**0.982**	**0.983**	0.920	0.910	0.910

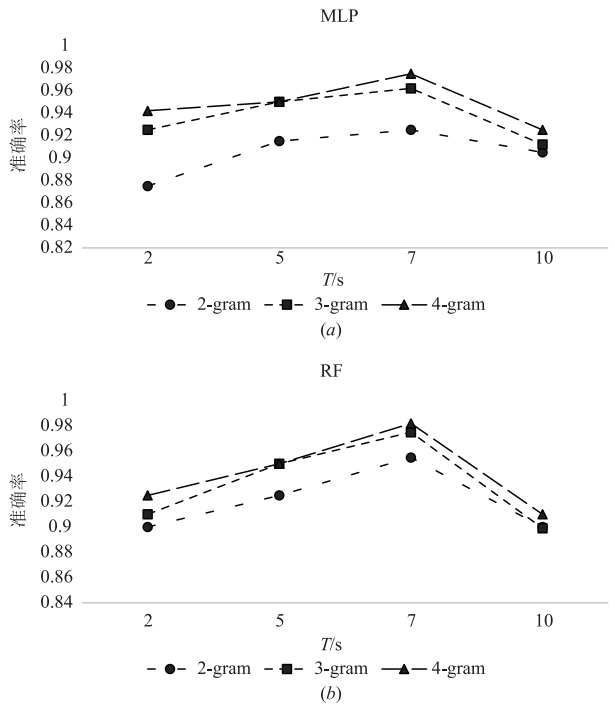

图7　REDMS在不同T下使用不同n-gram及MLP和RF获得的准确率

此外,表 8 给出了 REDMS 在 T 为前 7s 及使用 4-gram 序列时所需要的特征计算时间以及采用不同分类算法所需要的训练时间和测试时间. 特征计算时间是将从全部测试样本所采集的 API 短序列生成特征向量的时间,在实验中仅需要 0.248s. 另外,在本文实验中获得最佳准确率的 RF 算法仅需要 0.008s 和 0.001s 的训练及检测时间. 因此,综合考虑采集时段、特征计算时间和测试时间(三者相加即为方法对勒索软件的检测时间),REDMS 能够在数秒内判定一个软件是否为勒索软件.

为进一步评估 REDMS 对勒索软件检测的准确率,本文在表 9 中对比了文献[11]、文献[13]、文献[23]的检测结果.

表 8　REDMS 在 T 为前 7s 且使用 4-gram 时的运行时间

时间(s)	NB	SVM	MLP	KNN	RF
特征计算时间	0.248	0.248	0.248	0.248	0.248
训练时间	0.003	0.012	1.325	**0.001**	0.008
测试时间	**0.001**	0.003	0.085	**0.001**	**0.001**
检测时间	**7.249**	7.251	7.333	**7.249**	**7.249**

表 9　不同勒索软件检测方法的准确率

方法	准确率(%)	检测时间(s)
ZHANG[11]	97.5	–
Guo[13]	95	–
Morato[23]	**99**	20
REDMS(RF + 4-gram)	98.2	**7.2**

由表 9 可知,REDMS 使用 RF 算法及 4-gram 时能够以 7.2s 的检测时间获得 98.2% 的准确率. 文献[23]的方法在监控时段为 20s 时准确率为 99%,准确率略优于本文方法,但从检测时间上看,REDMS 所需的检测时间明显优于文献[23]的方法. 文献[11]与文献[13]的方法在本文测试集上所得的准确率分别为 97.5% 和 95%,均低于 REDMS.

5.3.2　未知勒索软件样本检测

本节主要给出 REDMS 对未知勒索软件样本的检测效果. 本文将 78 个勒索软件样本中来自 16 个家族的 64 个勒索软件样本和 40 个正常软件样本在 CTP 内调用的 API 短序列所计算出的特征向量作为训练集,将 16 个正常软件样本和未包含在训练集中的 8 个家族共 14 个勒索软件样本在 CTP 内调用的 API 短序列所计算出的特征向量作为测试集.

表 10 给出了 REDMS 在不同 T 下使用 4-gram 及五种算法在未知勒索软件测试集上得到的准确率.

表 10 REDMS 在不同 T 下使用 4-gram 及五种算法对未知勒索软件测试集的准确率

算法 \ T(s)	2	5	7	10
NB	0.858	0.909	0.909	0.867
SVM	0.733	0.800	0.833	0.800
MLP	0.867	0.925	**0.967**	0.887
KNN	0.919	0.900	0.900	0.867
RF	**0.933**	**0.937**	**0.967**	**0.900**

由表 10 可知,在使用 4-gram 划分序列的情况下,当 T 为前 2s 时,REDMS 使用 RF 获得的准确率为 93.3%;当 T 为前 5s 时,REDMS 的准确率可达 93.7%;当 T 为前 7s 时 REDMS 获得最佳准确率(96.7%).为进一步评估 REDMS 的准确率,本文将其与文献[11]和文献[13]的方法进行对比,对比结果如表 11 所示.

表 11 不同方法在未知勒索软件集上的准确率

方法	准确率(%)	检测时间(s)
ZHANG[11]	83.3	-
Guo[13]	93.3	-
REDMS(RF+4-gram)	**96.7**	7.2

由表 11 可知,REDMS 能够以 7.2s 的检测时间在未知软件测试集上能够获得 96.7% 的准确率,优于文献[11]和文献[13]方法在相同数据集上得到的准确率,这表明 REDMS 优于文献[11]和文献[13]所提方法,且能够以较高的准确率检测出未知勒索软件.

6 总结

本文通过分析多个勒索软件的运行机制,提出了基于 API 短序列的勒索软件早期检测方法,其引入一个在勒索软件及时检测方面具有重要意义的概念——CTP. CTP 的提出明确了勒索软件早期检测需要满足的时间要求,CTP 内采集的 API 短序列的使用有利于及早地检测出勒索软件. 实验结果表明,本文所提出的 REDMS 使用 4-gram 及 RF 算法时,能够在软件开始运行后 7.2s 时分别以 98.2% 和 96.7% 的准确率检测出实验中的已知和未知勒索软件样本. 在后续工作中,我们将致力于分析和检测锁屏型勒索软件. 此外,也将探索深度学习中长短期记忆网络和注意力机制等方法在勒索软件检测中的应用.

参考文献

[1] Bridges L. The changing face of malware[J]. Network Security,2008,2008(1):17-20.

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

[3] Al-Rimy B A S, Maarof M A, Shaid S Z M. Ransomware threat success factors, taxonomy, and countermeasures: a survey and research directions[J]. Computers & Security, 2018,74(5):144-166.

[4] Kok S H, Abdullah A, Jhanjhi N Z, et al. Ransomware, threat and detection techniques: A review[J]. International Journal Computer Science and Network Security, 2019, 19(2):136-146.

[5] Kok S H, Abdullah A, Jhanjhi N Z, et al. Prevention of crypto-ransomware using a pre-encryption detection algorithm[J]. Computers, 2019, 8(4):79.

[6] Sgandurra D, Muñoz-González L, Mohsen R, et al. Automated dynamic analysis of ransomware: benefits, limitations and use for detection[J]. Cryptography and Security, 2016,9:03-20.

[7] San C C, Thwin M M S, Htun N L. Malicious software family classification using machine learning multi-class classifiers[J]. Computational Science and Technology, 2019,41:423-433.

[8] 任卓君,陈光,卢文科. 基于 N-gram 特征的恶意代码可视化方法[J]. 电子学报,2019,47(10):2108-2115.
REN Zhuo-jun, CHEN Guang, LU Wen-ke. Malware visualization methods based on N-gram features[J]. Acta Electronica Sinica, 2019, 47(10):2108-2115. (in Chinese)

[9] Zhang W, Yoshida T, Tang X. A comparative study of TF-IDF, LSI and multi-words for text classification[J]. Expert Systems with Applications, 2011, 38(3):2758-2765.

[10] 李鹏伟,姜宇谦,薛飞扬,黄佳佳,徐超. 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):1502-1508.
LI Peng-wei, JIANG Yu-qian, XUE Fei-yang, HUANG Jia-jia, XU Chao. Arobust approach for android malware detection based on deep learning[J]. Acta Electronica Sinica, 2020, 48(8):1502-1508. (in Chinese)

[11] Zhang H, Xiao X, Mercaldo F, et al. Classification of ransomware families with machine learning based on N-gram of opcodes[J]. Future Generation Computer Systems, 2019,90:211-221.

[12] Harrington P. Machine Learning in Action[M]. USA: Manning Publications Co, 2012. 122-136.

[13] 郭春,陈长青,申国伟,蒋朝惠. 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 Chun, CHEN Chang-qing, SHEN Guo-wei, et al. A ransomware classification method based on visualization[J]. Netinfo Security, 2020, 20(4):31-39. (in Chinese)

[14] Sharma S,Singh S. Texture-Basedautomated Classification of Ransomware[EB/OL]. https://link. springer. com/article/10. 1007/s40031-020-00499-w. 2020-10-31.
[15] 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.
[16] Damodaran A,Troia FD,Visaggio CA,et al. A comparison of static,dynamic,and hybrid analysis for malware detection[J]. Computer Virology and Hacking Techniques,2017,13(1):1 – 12.
[17] Hampton N,Baig Z,Zeadally S. Ransomware behavioural analysis on windows platforms[J]. Journal of Information Security and Applications,2018,40:44 – 51.
[18] Kharaz A,Arshad S,Mulliner C,et al. UNVEIL:A large-scale,automated approach to detecting ransomware[A]. 25th USENIX Security Symposium[C]. Austin,TX:Association,2016. 757 – 772.
[19] Vinayakumar R,Soman K P,Velan K K,et al. Evaluating shallow and deep networks for ransomware detection and classification[A]. International Conference on Advances in Computing,Communications and Informatics (ICACCI)[C]. Croatia:IEEE,2017. 259 – 265.
[20] LeCun Y,Bengio Y,Hinton G. Deep learning[J]. Nature,2015,521(7553):436 – 444.
[21] Feng Y,Liu C,Liu B. Poster:A New Approach to Detecting Ransomware with Deception[EB/OL]. http://173. 236. 186. 201/TC/SP2017/poster-abstracts/IEEE-SP17 _ Posters_paper_26. pdf. 2020-09-24.
[22] Scaife N,Carter H,Traynor P,et al. Cryptolock (and drop it):Stopping ransomware attacks on user data[A]. IEEE 36th International Conference on Distributed Computing Systems (ICDCS)[C]. Nara Japan:IEEE,2016. 303 – 312.
[23] Morato D,Berrueta E,Magaña E,et al. Ransomware early detection by the analysis of file sharing traffic[J]. Journal of Network and Computer Applications,2018,124:14 – 32.
[24] 吴玉佳,李晶,宋成芳,常军. 基于高效用神经网络的文本分类方法[J]. 电子学报,2020,48(2):279 – 284.
WU Yu-jia,LI Jing,SONG Cheng-fang,CHANG Jun. High utility neural networks for text classification[J]. Acta Electronica Sinica,2020,48(2):279 – 284. (in Chinese)
[25] Singh K,Agrawal S. Comparative analysis of five machine learning algorithms for IP traffic classification[A]. International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)[C]. Udaipur,India:IEEE,2011. 33 – 38.

作者简介

陈长青　男,1997 年出生,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:ccq_study@163. com

郭　春(通信作者)　男,1986 年出生,博士,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