计算机科学
COMPUTER SCIENCE

基于深度学习的勒索软件早期检测方法

刘文静, 郭春, 申国伟, 谢博, 吕晓丹

引用本文

刘文静, 郭春, 申国伟, 谢博, 吕晓丹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 Wenjing, GUO Chun, SHEN Guowei, XIE Bo, LYU Xiaodan. Ransomware Early Detection Method Based on Deep Learning [J]. Computer Science, 2023, 50(3): 391-398.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220200182

基于深度学习的勒索软件早期检测方法

刘文静　郭　春　申国伟　谢　博　吕晓丹
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　贵阳 550025
(lwj_gzedu@163.com)

摘　要　近年来,勒索软件的活跃度高居不下,给社会造成了严重的经济损失。文件一旦被勒索软件加密后将难以恢复,因此如何及时且准确地检测出勒索软件成为了当前的研究热点。为了提升勒索软件检测的及时性和准确性,在分析多种勒索软件家族与良性软件运行初期行为的基础上,提出了一种基于深度学习的勒索软件早期检测方法(Ransomware Early Detection Method Based on Deep Learning,REDMDL)。REDMDL以软件运行初期所调用的一定长度的应用程序编程接口(Application Programming Interface,API)序列为输入,结合词向量和位置向量对API序列进行向量化表征,再构建深度卷积网络与长短时记忆网络(Convolutional Neural Network-Long Short Term Memory,CNN-LSTM)相结合的神经网络模型,来实现对勒索软件的早期检测。实验结果显示,REDMDL能够在一个软件运行后数秒内高准确率地判定其是勒索软件还是良性软件。

关键词:勒索软件;早期检测;CNN;LSTM;API
中图法分类号　TP309

Ransomware Early Detection Method Based on Deep Learning

LIU Wenjing,GUO Chun,SHEN Guowei,XIE Bo and LYU Xiaodan
State Key Laboratory of Public Big Data,College of Computer Science and Technology,Guizhou University,Guiyang 550025,China

Abstract　In recent years,ransomware is becoming increasingly prevalent,causing serious economic losses. Since files encrypted by ransomware are difficult to recover,how to timely and accurately detect ransomware is a hot point nowadays. To improve the timeliness and accuracy of ransomware detection,this paper analyzes the behavior of ransomware family and benign software in the early stage of operation and proposes a ransomware early detection method based on deep learning(REDMDL). REDMDL takes a certain length of application programming interface(API) sequence that is obtained by software running at the initial stage as input,combines word vector and position vector to vectorize the collected API sequence,and then constructs a convolutional neural network-long short term memory(CNN-LSTM) neural network model for early detection of ransomware. Experimental results show that REDMDL can accurately determine whether the software is ransomware or benign within seconds after it starting to run.

Keywords　Ransomware,Early detection,CNN,LSTM,API

1　引言

勒索软件是一种破坏数据可用性的恶意软件[1]。它通过加密用户的文件来阻止用户使用其重要文件,并向用户勒索赎金以换取文件的解密密钥[2]。据DeCapua发布的数据显示,在2013年1月10日至2019年7月11日期间,大约有价值1.4435亿美元的比特币被支付给了勒索软件攻击者[3]。不断出现的勒索软件及其变种给企业和个人造成了重大损失,已成为当前最具破坏性的安全威胁之一[4]。

目前勒索软件主要分为两个类型:第一种类型是锁屏型勒索软件,其感染主机后会锁定其屏幕[5],此时用户只能执行支付赎金等少量操作[6];第二种类型是加密型勒索软件,其会在受害主机上检索特定的文件并进行加密后删除原文件[7]。自2015年以来,新出现的勒索软件中加密型勒索软件的占比逐年增加[8]。因此,本文将加密型勒索软件作为研究对象,下文无特指时所提及的"勒索软件"均指加密型勒索软件。

在勒索软件完成对受害主机特定文件的加密后,受害主机上的这些文件将丧失可用性且难以在未支付赎金的情况下进行解密[9]。即便可以通过前期的备份来实现文件恢复,但仍然会出现在恢复期内无法正常使用信息系统或文件的情况[10]。因此,及时且准确地检测出勒索软件,对于保障企业信息系统的安全稳定运行具有非常重大的意义[11]。现有勒索软件检测方法可以划分为静态检测方法和动态检测方法。静态检测方法在获取集成了加壳[12]等静态分析逃逸技术的

到稿日期:2022-02-28　返修日期:2022-06-20
基金项目:国家自然科学基金(62162009);贵州省自然科学基金(黔科合基础[2020]1Y268);贵州省科技计划项目(黔科合重大专项字[2018]3001)
This work was supported by the National Natural Science Foundation of China(62162009),Science and Technology Foundation of Guizhou Province,China([2020]1Y268) and Guizhou Provincial Science and Technology Project([2018]3001).
通信作者:郭春(gc_gzedu@163.com)

勒索软件的特征上存在难度；主流的勒索软件动态检测方法往往需要采集长时间的软件行为才能准确判别勒索软件，难以满足对勒索软件进行及时检测的需求。而重点关注检测及时性的勒索软件早期检测方法，在兼顾检测的及时性和准确性方面还有较大的提升空间。

近年来，深度学习在图像识别、语言识别等多个领域取得了巨大突破，在恶意软件检测领域也有一些成功的应用[13]。为了在获取高准确精度的同时进一步提升勒索软件检测的及时性，本文提出了一种基于深度学习的勒索软件早期检测方法。REDMDL 以勒索软件和良性软件运行初期所调用的一定长度的 API 序列为分析对象，采用深度卷积网络与长短时记忆网络相结合的网络模型，能够在软件运行后数秒内判别其是勒索软件还是良性软件。本文的主要贡献如下：

(1)从 API 层面分析多个不同家族的勒索软件和多种类型的良性软件在运行初期的行为，探索以软件运行初期所调用的一定长度的 API 序列来区分勒索软件和良性软件的合理性，为构建以 API 序列为输入的基于深度学习的勒索软件检测方法奠定基础。

(2) 基于贡献(1)，提出了一种基于深度学习的勒索软件检测方法 REDMDL。REDMDL 运用词向量和位置向量将软件运行初期所调用的一定长度的 API 序列向量化，并结合 CNN-LSTM 神经网络模型实现对勒索软件的早期检测。

(3)使用来自 30 个不同家族的 300 款勒索软件以及 200 款多种类型的良性软件对 REDMDL 进行实验测试。实验结果表明，REDMDL 能够通过软件运行初期调用的一定长度的 API 序列，来实现对勒索软件样本的高准确率检测。

2 相关工作

国内外研究人员近年来提出了多种勒索软件检测方法，本文将这些勒索软件的检测方法分为 3 类：勒索软件静态检测方法、勒索软件动态检测方法和勒索软件早期检测方法。

2.1 勒索软件静态检测方法

静态检测方法指，在不运行程序的情况下，对程序文件中的字节序列和操作码等静态属性进行分析，从中提取特征并训练检测模型。Khammas[14]使用频繁模式挖掘算法，从勒索软件源文件的原始字节中提取特征，再通过随机森林(Random Forest，RF)算法建立检测模型。Guo 等[15]提出了一种基于可视化的勒索软件分类方法，该方法将待分类软件转化为灰度图后，再利用 VGG16 提取图像特征，使用机器学习算法训练分类模型，以进行勒索软件分类。Zhang 等[16]从勒索软件的可执行文件中提取操作码 *N*-grams，测试了多种机器算法所训练的分类模型的分类效果。Xiao 等[17]提出了一种基于操作码 N-gram 和深度学习的静态分析框架，该框架在处理操作码序列的过程中加入了自注意力机制，能够适用于长操作码序列分析。Hsu 等[18]通过分析 WannaCry，Phobos，GandCrab 和 Globelmposter 4 种勒索软件感染系统后加密文件所具有的特性，提取了 17 种特征作为输入，利用支持向量机构建了检测模型。总体来说，静态检测方法侧重于分析和利用勒索软件源文件中的特性，当勒索软件应用了代码混淆或加壳等静态分析逃逸技术时，提取其静态特征的难度会大大增加。

2.2 勒索软件动态检测方法

动态检测方法是使用从程序执行所表现出的行为中提取的特征进行检测的方法。Hwang 等[5]以软件的 API 序列为特征，建立了一个马尔可夫链结合 RF 算法的勒索软件两阶段混合检测模型。为了动态分析勒索软件的行为，Kharraz 等[19]提出了一种名为 UNVEIL 的动态分析系统，该系统通过监测系统 I/O 数据缓冲区熵、访问模式和文件系统活动来实现勒索软件检测。Ramesh 等[20]设计了一个有限状态机模型，通过监视底层系统的状态变化来检测勒索软件。Peña 等[21]利用勒索软件运行过程中扫描网络、注册表活动和文件系统操作等 14 种行为特征来建立分类模型，以实现勒索软件检测。Daku 等[22]运用勒索软件的行为特征来识别勒索软件变体。总体来说，传统的动态检测方法需要采集较长时间的软件行为才能得到高准确率的检测结果，难以满足对勒索软件进行及时检测的需求。

2.3 勒索软件的早期检测方法

勒索软件感染受害主机后被加密的文件难以恢复，因此在勒索软件完成对受害主机全部文件加密前将其检出的早期检测方法得到了越来越多的重视。Scaife 等[23]提出了一种勒索软件预警系统，在监测到文件类型等属性变化时向用户发出警报，但是他们未给出该系统所需的具体判别时长。Morato 等[24]提出了一种通过分析网络共享卷的交互流量来检测勒索软件的早期检测方法，该方法达到其最佳检测精度所需的流量监控时间为 20 s。Al-Rimy 等[25]将基于互信息的特征选择技术运用于勒索软件早期检测，该方法能够利用从勒索软件运行初期的行为数据中提取并选出的特征获得良好的检测精度，但是文章中并未给出其检测出勒索软件所需的具体时长。Chen 等[26]提出了“勒索软件检测关键时间段”的概念，并从该概念出发提出了一种基于 API 短序列的勒索软件早期检测方法。该方法在 API 采集时段为前 7 秒且使用 RF 算法时获取其最佳的检测精度。

由上述相关研究可以看到，现有的勒索软件动态检测方法虽能获得较高的检测精度，但是通常需要采集较长时间的软件行为。而勒索软件在短时间内便能完成对被害系统的加密，这导致这些方法难以满足对其进行及时检测的需求；现有的勒索软件早期检测方法在同时兼顾检测精度和及时性方面还有较大的提升空间。CNN-LSTM 神经网络模型是一种复合型网络，将 CNN 在特征提取方面的优异性能与 LSTM 能解决长依赖问题的特性相结合，在入侵检测等领域取得了优异的成绩[27]。本文从及时且准确检测勒索软件的需求出发，重点关注勒索软件运行初期调用的一定长度的 API 序列，并结合 CNN-LSTM 模型构建勒索软件早期检测方法。

3 勒索软件运行初期的 API 序列分析

勒索软件攻击通常包含系统感染、联系 C&C 服务器、加密密钥管理、文件加密、勒索赎金 5 个阶段。感染系统后，勒索软件会与 C&C 服务器联系以获取加密密钥。在获取到

加密密钥后，勒索软件会利用从 C&C 服务器获得的密钥对受害主机中的文件进行加密[28]。在现实中，在勒索软件完成对受害主机上全部文件的加密后才将其检出将没有实际意义。因此，如何在勒索软件的文件加密阶段完成前（也即弹出勒索告知文本之前）将其准确检出，是勒索软件防御的关键问题。

现有可感染 Windows 系统的勒索软件均是通过调用 Windows API 来实施操作，其调用的 API 构成的 API 序列能够反映该软件的动态行为。本文依据文献[29]中的勒索软件阶段划分方法，将勒索软件运行后首次调用加密类 API 视为其开始进入文件加密阶段的标志。在 Windows 7 操作系统中分别运行 3 款勒索软件，并用 API Monitor 监控这些软件的 API 序列，记录各勒索软件首次调用加密类 API 之前的 API 调用数量，结果如表 1 所列。表 1 中，3 款勒索软件在首次调用加密 API 之前所调用的 API 数量分别为 3601，3615 和 9804，这表明勒索软件从开始运行到进入数据加密阶段之前需要先调用一定数量的 API。勒索软件的攻击意图决定了其感染受害者主机后会力求尽早地进入文件加密阶段。因此，本文推测，相比良性软件，勒索软件运行后将更快地调用一定数量的 API，以完成其遍历文件、联系 C&C 服务器及管理密钥等行为，即勒索软件在运行后调用一定数量 API 所需的时长会短于大多数良性软件调用相同数量 API 所需的时长。

表 1 勒索软件首次调用的加密类 API 及其序号

Table 1 Encryption API called by different ransomware for the first time and its corresponding sequence number

名称	首次调用的加密类 API	对应序号
LockerGoga	CryptStringToBinaryA	3601
SAGE	CryptAcquireContextW	3615
GandCrab	CryptAcquireContextW	9804

为了验证上述推测，本文在 Any. Run 沙箱部署了以 Windows 7 为操作系统的运行环境，并利用 API Monitor 监控来自 30 个不同家族的 300 款勒索软件以及 200 款多种类型的良性软件运行中所调用的 API 序列。由于训练深度学习模型时需要统一输入的大小，因此本文重点关注软件运行后所调用的一定数量的 API。表 2 和表 3 分别列出了各良性软件和勒索软件完成长度为 1 000 的 API 序列调用所需要的时长。由表 2 可知，本文所分析的各良性软件完成第一个长度为 1 000 的 API 序列调用时平均需要 2 219.74 ms。由表 3 可知，本文所分析的各勒索软件运行后首次完成长度为 1 000 的 API 序列调用平均仅需要 1 016.05 ms。

表 2 良性软件首次完成长度为 1 000 的 API 序列调用所需时长

Table 2 Runtime required for benign software to complete the first API sequence of length 1 000

良性软件类型（软件数）	平均所需时长/ms
办公软件(36)	1 343
聊天通讯(17)	2 031
安全杀毒(15)	1 500
视频软件(24)	2 438
工具软件(45)	2 031
游戏娱乐(20)	1 102
其他(43)	3 875
总平均所需运行时间/ms	2 219.74

表 3 勒索软件首次完成长度为 1 000 的 API 序列调用所需时长

Table 3 Runtime required for ransomware to complete the first API sequence of length 1 000

勒索软件（变种数量）	平均所需时长/ms
Alphacrypt(9)	1 180
Cerber(14)	1 076
CryptoShield(8)	1 138
CryptoWire(11)	1 185
Crysis(13)	266
CTB-Locker(8)	1 357
Gandcrab(12)	1 138
GenericKD(8)	637
Lockbit(10)	1 660
Maze(11)	573
Mzrevenge(7)	1 515
PolyRansom(12)	1 422
Prolock(9)	1 219
Ransomware. Dharma(10)	1 154
Ransomware. Eleta(11)	1 123
Ransomware. FRS(9)	1 092
Ransom. HelloKitty(6)	1 297
Ransom. LockerGoga(12)	907
Ransomware. Kraken(11)	1 750
Ransomware. SAGE(9)	1 170
Ransomware. Santa(7)	1 422
Ransomware. Wlu(9)	1 123
Ryuk(11)	532
Spora(9)	203
SATURN(10)	1 092
Sodinokibi(9)	828
SilentSpring(12)	109
Teslacrypt(8)	1 692
Troldesh(9)	172
Wannacry(16)	1 107
总平均所需运行时间/ms	1 016.05

基于上述结果，本文选用软件运行后所调用的一定长度的 API 序列来区分勒索和良性软件。鉴于勒索和良性软件调用相同长度的 API 序列所需要的时间相差较大，为实现勒索软件早期检测的目标，本文在采集 API 序列时还考虑了时间因素：以勒索软件从开始运行到调用某一长度（记为 n）的 API 序列的平均时长（记为 t）为限来采集各软件的 API 序列。若在时长 t 内所采集的序列长度不足 n，则对该序列填充 0 至其长度达到 n。后续本文将通过这些长度为 n 的 API 序列进行勒索软件早期检测。

4 基于深度学习的勒索软件早期检测方法

CNN 和 LSTM 在恶意软件领域均取得了不错的检测效果。与传统的机器学习算法相比，深度学习的优势在于其能够利用深层次神经网络自动提取高层抽象特征，从而减少繁琐的特征工程。以第 3 节勒索软件和良性软件运行初期调用的 API 序列分析结果为基础，本文构建了如图 1 所示的检测方法 REDMDL。REDMDL 首先采集待测样本的 API 序列，再对 API 序列进行词嵌入表示，最后使用训练好的 CNN-LSTM 模型区分其是勒索软件还是良性软件。

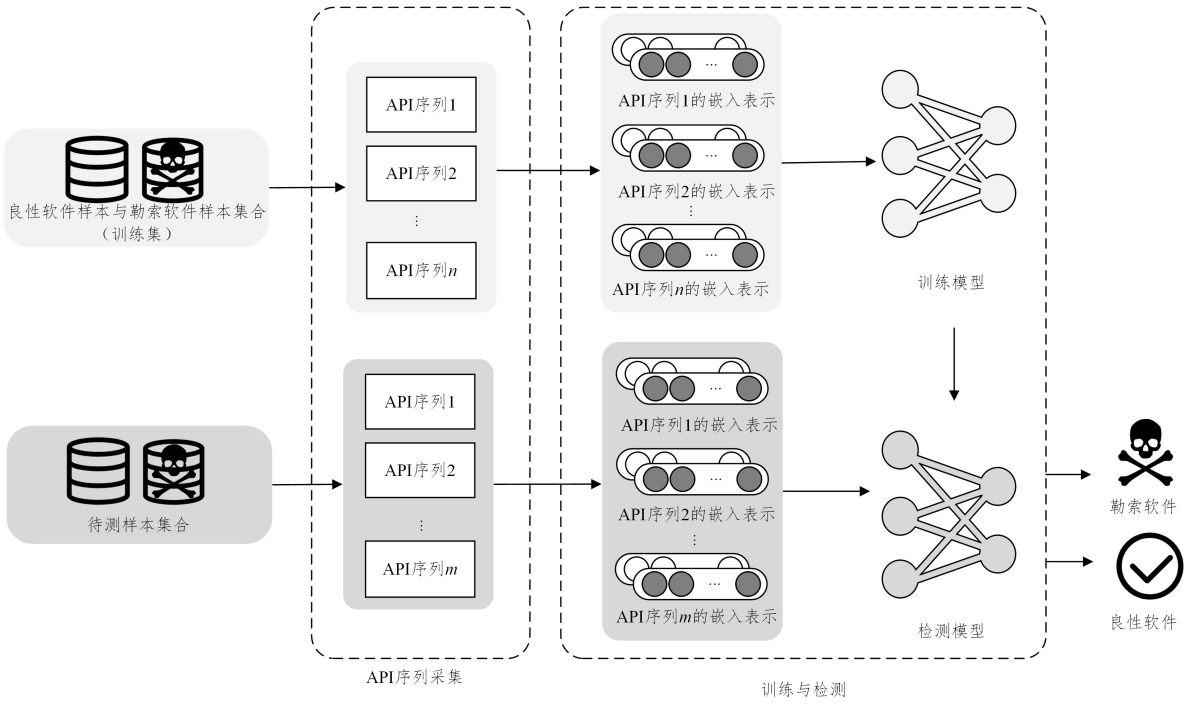

图 1 基于深度学习的勒索软件早期检测方法的整体框架图

Fig. 1 Overall framework of REDMDL

4.1 API 序列采集

REDMDL 在本阶段将采集软件运行后在时限 t 内所调用的长度为 n 的 API 序列，若在 t 内采集到的序列长度不足 n，则填充 0 至此长度。n 决定了 REDMDL 采集 API 序列的长度，因此 n 的取值对 REDMDL 的检测效果会产生影响。n 的取值既不能太小也不能太大：若取值太小，则对应的 API 序列通常只包含了软件的少量行为，以致于难以通过其进行准确判别；若取值太大，则所需要的采集时间将很长，以致于难以实现在勒索软件完成文件加密阶段之前将其检出的目标。基于对勒索软件行为的分析结果，REDMDL 重点关注文件系统 API 和进程间通信 API。REDMDL 在 API 序列采集阶段监控的 API 如表 4 所列，其将表 4 中各 API 通过一个唯一的标识号（API No. ∈ N+）进行对应，从而将软件运行中调用的 API 序列映射为实数向量。

表 4 REDMDL 监控的 API

Table 4 API monitored by REDMDL

API No.	API 名称	API No.	API 名称	API No.	API 名称	API No.	API 名称
1	CopyFile	8	GetSystemPowerStatus	15	LCreate	22	ReplaceFile
2	CreateFile	9	GetCurrentDirectory	16	LRead	23	SetEndOfFile
3	CreateMailslot	10	GetSystemDirectory	17	MoveFile	24	SetVolumeMountPiont
4	CreatePipe	11	GetVolumeInformation	18	OpenFile	25	SetVolumeLabel
5	DeleteFile	12	GetdriverType	19	QueryDosDevice	26	SetFileAttributes
6	DeviceIoControl	13	HWrite	20	ReadFile	27	VerQueryValue
7	FlushFileBuffers	14	LClose	21	ReOpenFile	28	WriteFile

4.2 训练与检测

为了更好地提取所输入 API 序列之间的关系，REDMDL 运用了如图 2 所示的 CNN＋LSTM 相结合的网络模型。REDMDL 首先结合词向量和位置向量对上一阶段得到的 API 序列进行向量化表征，然后使用 CNN 从中自动获取特征，再运用 LSTM 建立时间序列预测模型学习新特征，最后通过全连接层和输出层得到待测样本序列的预测结果。REDMDL 中的 CNN-LSTM 模型包含以下层。

(1)词嵌入层。REDMDL 通过结合词向量和位置向量来保留 API 序列之间的调用关系。REDMDL 使用连续词袋算法(Continuous Bag Of Words，CBOW)[30]模型生成词向量，再利用正、余弦函数表示向量的绝对位置，并将位置向量和词向量加起来作为模型的输入。对于长度为 n 的 API 序列 $\boldsymbol{S}$，将 $\boldsymbol{S}$ 中的每个 API 视为一个词。词嵌入表示为 $\boldsymbol{e}_n$，位置向量表示为 $\boldsymbol{pos}$，相加后的向量 $\boldsymbol{Z}$ 如式(1)所示：

$$\boldsymbol{Z}=\boldsymbol{e}_n+\boldsymbol{pos} \tag{1}$$

(2)卷积层。CNN 是一种参数共享的前馈神经网络，常用于提取输入向量中的特征[31]。REDMDL 包含卷积核大小分别为 2，3，4 的 3 个卷积层，每个卷积核的权重矩阵不同，对应提取 API 序列中某一部分特征。第 l 层卷积层输出的特征矩阵 $\boldsymbol{E}_l$ 如式(2)所示：

$$\boldsymbol{E}_l=f(\boldsymbol{W}_l\boldsymbol{Z}+\boldsymbol{b}_l) \tag{2}$$

其中，权重矩阵 $\boldsymbol{W}_l$ 和偏置向量 $\boldsymbol{b}_l$ 为 l 层网络学习的参数。

(3)池化层。REDMDL 将各卷积层输出的特征矩阵传递给一个池化层。REDMDL 在该层使用 Max Pooling 操作，它是 CNN 模型中一种常用的下采样操作，可以减少模型参数数量进而减少模型过拟合的问题[32]。Max Pooling 选择每个过滤器中的最大值来代表该过滤器所表示的语义信息，并

保留这些特征值原始的先后顺序。因此，经过池化层后每个卷积核所提取特征矩阵的维度明显降低，但仍保留了 API 序列的顺序信息。接下来，REDMDL 将池化后所得的 3 个通道的特征矩阵拼接在一起，作为 LSTM 层的输入。

(4)LSTM 层。API 序列中相邻 API 之间具有调用关系。LSTM 是一种特殊的循环神经网络，具有学习长期依赖关系的能力[33]。REDMDL 利用 LSTM 从前一层输出的特征矩阵中提取新特征矩阵 $\boldsymbol{X}$。LSTM 有两个传输状态，分别是细胞状态 $\boldsymbol{c}_t$、隐藏层状态 $\boldsymbol{h}_t$。令 $\boldsymbol{x}_t$ 为前时间步的输入，$\boldsymbol{h}_{t-1}$ 为上一时刻隐藏层的输出，则 t 时刻 LSTM 单元的输入由当前时刻的输入和 $t-1$ 时刻 LSTM 单元的输出 $\boldsymbol{h}_{t-1}$ 组成。每一个 LSTM 单元包括 3 个门，分别是遗忘门 $\boldsymbol{f}_t$、输入门 $\boldsymbol{i}_t$、输出门 $\boldsymbol{o}_t$。这 3 个门决定了如何更新细胞状态 $\boldsymbol{c}_t$ 和隐藏层状态 $\boldsymbol{h}_{t-1}$。LSTM 每个时间步的计算式如式(3)－式(8)所示[34]：

$$\boldsymbol{f}_t=\sigma(\boldsymbol{W}_f\cdot[\boldsymbol{h}_{t-1},\boldsymbol{x}_t]+\boldsymbol{b}_f) \tag{3}$$

$$\boldsymbol{i}_t=\sigma(\boldsymbol{W}_i\cdot[\boldsymbol{h}_{t-1},\boldsymbol{x}_t]+\boldsymbol{b}_i) \tag{4}$$

$$\tilde{\boldsymbol{c}}_t=\tanh(\boldsymbol{W}_c\cdot[\boldsymbol{h}_{t-1},\boldsymbol{x}_t]+\boldsymbol{b}_c) \tag{5}$$

$$\boldsymbol{c}_t=\boldsymbol{f}_t*\boldsymbol{c}_{t-1}+\boldsymbol{i}_t*\tilde{\boldsymbol{c}}_t \tag{6}$$

$$\boldsymbol{o}_t=\sigma(\boldsymbol{W}_o\cdot[\boldsymbol{h}_{t-1},\boldsymbol{x}_t]+\boldsymbol{b}_o) \tag{7}$$

$$\boldsymbol{h}_t=\boldsymbol{o}_t\cdot\tanh(\boldsymbol{c}_t) \tag{8}$$

(5)全连接层。REDMDL 将前一层输出的特征矩阵输入到全连接层以进行处理，得到一维向量 $\boldsymbol{Y}$。

(6)输出层。REDMDL 通过 Softmax 函数计算 $\boldsymbol{Y}$ 得到最终的预测结果，即待测样本是勒索软件还是良性软件。完整执行过程如算法 1 所示。

算法 1 REDMDL

输入：S //待测软件的 API 序列

输出：Result //检测结果

1. BEGIN//读取 API 序列
2. $\mathbf{e}_n$＝word embedding(**S**)//提取 **S** 的词向量编码
3. **pos**＝position embedding**S**) // 提取 **S** 的位置向量编码
4. **Z**＝$\mathbf{e}_n$＋ **pos** //将位置向量和词向量相加
5. $\mathbf{F}_1$＝relu($\mathbf{W}_1$ * **Z**＋$\mathbf{b}_1$) //通过对 **Z** 进行卷积操作得到特征矩阵 $\mathbf{F}_1$
6. $\mathbf{C}_1$＝MaxPooling1D($\mathbf{F}_1$) //通过对 $\mathbf{F}_1$ 池化得到特征矩阵 $\mathbf{C}_1$
7. $\mathbf{F}_2$＝relu($\mathbf{W}_2$ * $\mathbf{C}_1$＋$\mathbf{b}_2$)
8. $\mathbf{C}_2$＝MaxPooling1D($\mathbf{F}_2$) //通过对 $\mathbf{F}_2$ 池化得到特征矩阵 $\mathbf{C}_2$
9. $\mathbf{F}_3$＝relu($\mathbf{W}_3$ * $\mathbf{C}_2$＋$\mathbf{b}_3$)
10. $\mathbf{C}_3$＝MaxPooling1D($\mathbf{F}_3$) //通过对 $\mathbf{F}_3$ 池化得到特征矩阵 $\mathbf{C}_3$
11. **H**＝concat($\mathbf{C}_1$，$\mathbf{C}_2$，$\mathbf{C}_3$) //将 $\mathbf{C}_1$，$\mathbf{C}_2$ 和 $\mathbf{C}_3$ 拼接得到特征矩阵 **H**
12. **X**＝LSTM(**H**) //对 **H** 提取特征得到特征矩阵 **X**
13. **Y**＝FullyConnected(**X**) //将 **X** 转化为一维向量 **Y**
14. Result＝Softmax(**Y**) //预测 **Y** 是勒索软件还是良性软件
15. END
16. RETURN Result

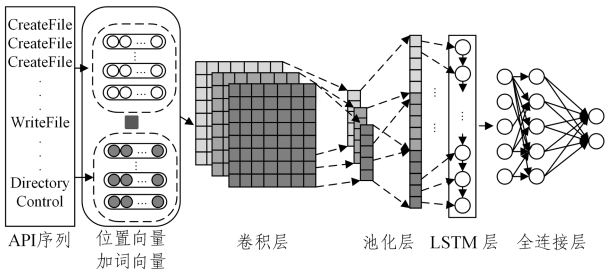

图 2 REDMDL 中所使用的 CNN-LSTM 模型

Fig. 2 CNN-LSTM model used in REDMDL

5 实验及结果

5.1 实验设置

本文在 Any. Run 沙箱部署了 Window7(x64)的运行环境，并在其中放置了数 10 个不同类型(doc，ppt 等)的文档。检测勒索软件的主机硬件配置为 Intel i7 9700K CPU、64.0 GB 内存、TITAN RTX 24GB GPU。REDMDL 通过 Python3.6 和 TensorFlow2.4.1 进行编程实现和模型训练。

5.2 实验数据集

学术界目前尚无公开的勒索软件标准数据集。本文从 VirusShare，GitHub，Any. Run 这 3 个公开网站采集可运行的勒索软件，从 360 软件管家采集良性软件。实验样本包括 30 个家族的 300 个不同勒索软件样本、7 个类别的 200 个不同良性软件样本。为使得实验中的检测任务具有复杂性和现实性，本文将上述样本划分为如表 5 所列的训练集、勒索软件变种测试集和未知勒索软件测试集。

表 5 实验样本及划分

Table 5 Experimental samples and division

类型	家族	训练集	勒索软件变种测试集	未知勒索软件测试集	总数
勒索软件	Alphacrypt	7	2	—	9
	Cerber	11	3	—	14
	CryptoWire	8	3	—	11
	Crysis	10	3	—	13
	Gandcrab	9	3	—	12
	Lockbit	7	3	—	10
	Maze	8	3	—	11
	PolyRansom	9	3	—	12
	Prolock	7	2	—	9
	Ransomware. Dharma	7	3	—	10
	Ransomware. Eleta	8	3	—	11
	Ransomware. FRS	7	2	—	9
	Ransom. LockerGoga	9	3	—	12
	Ransomware. Kraken	8	3	—	11
	Ransomware. SAGE	7	2	—	9
	Ransomware. Wlu	7	2	—	9
	Ryuk	8	3	—	11
	SATURN	7	3	—	10
	SilentSpring	9	3	—	12
	Wannacry	13	3	—	16
	CryptoShield	—	—	8	8
	CTB-Locker	—	—	8	8
	GenericKD	—	—	8	8
	Mzrevenge	—	—	7	7
	Ransom. HelloKitty	—	—	6	6
	Ransomware. Santa	—	—	7	7
	Spora	—	—	9	9
	Sodinokibi	—	—	9	9
	Teslacrypt	—	—	8	8
	Troldesh	—	—	9	9
良性软件	办公软件	20	8	8	36
	聊天通讯	10	4	3	17
	安全杀毒	8	4	3	15
	视频软件	12	6	6	24
	工具软件	22	11	12	45
	游戏娱乐	10	5	5	20
	其他应用	18	12	13	43
总数		266	105	129	500

训练集用于训练 REDMDL 的检测模型，勒索软件变种测试集和未知勒索软件测试集分别用于测试 REDMDL 对

勒索软件变种(待测样本为训练集中勒索软件样本的同家族变种)和未知勒索软件(待测样本所属勒索软件家族未在训练集中出现过)的检测能力。

5.3 评价指标

本文使用准确率(Accuracy)、精确率(Precision,P)、召回率(Recall,R)以及 F1-score(F1)这 4 个常用的指标作为方法的检测精度评价指标,计算方法如式(9)—式(12)所示:

$$Accuracy=\frac{TP+TN}{TP+TN+FP+FN} \tag{9}$$

$$P=\frac{TP}{TP+FP} \tag{10}$$

$$R=\frac{TP}{TP+FN} \tag{11}$$

$$F1=2\times\frac{P\times R}{P+R} \tag{12}$$

5.4 实验结果

本节分别给出了 REDMDL 对勒索软件变种及未知勒索软件的检测结果。此外,为了评估 REDMDL 在不同 API 序列长度 n 下的检测效果,本文基于兼顾检测的准确性和及时性的原则,将 n 分别设置为 1000,2000,3000,4000 来测试其检测精度和检测效率。

5.4.1 勒索软件变种检测结果

图 3 给出了 REDMDL 在不同 n 下检测勒索软件变种测试集所得到的 *Accuracy* 值。随着 n 值在 1000 至 4000 范围内增加,REDMDL 所得到的 *Accuracy* 值也越来越高。这是因为 n 在一定范围内增加时,REDMDL 能够获取到更多的软件行为轨迹来进行分析及关系捕获,这有助于其获得更好的检测精度。但是 n 的取值不宜过大,因为当 n 值过大时,其对应的 API 序列将包含勒索软件完成文件加密阶段后所调用的 API 序列(勒索软件完成加密并生成勒索告知文本之后,其调用 API 数量会大大减少,且与良性软件行为的区别不显著[26]),这对 REDMDL 的检测精度会有不利影响。此外,从早期检测的角度看,n 的取值也不宜过大,因为过大的 n 值将使得 REDMDL 需要很长的序列采集时长,进而大大增加其判别勒索软件所需的时间,以致于无法满足在勒索软件完成文件加密阶段之前将其检出的需求。

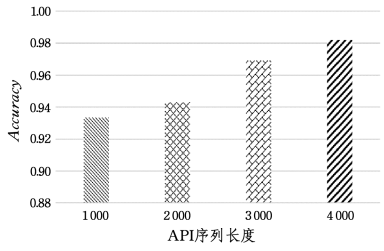

图 3 REDMDL 在不同 n 下检测勒索软件变种样本集得到的 *Accuracy*

Fig. 3 *Accuracy* of REDMDL for different n values on ransomware variant test set

表 6 列出了 REDMDL 在不同 n 下检测勒索软件变种样本集所得到的 P,R 以及 $F1$。由表 6 可知,在 n 为 4000 时,REDMDL 获得了最高的 P 和 $F1$ 值,分别为 100%,98.7%。

表 6 REDMDL 在不同 n 下检测勒索软件变种测试集得到的 P,R 以及 $F1$

Table 6 P, R, and $F1$ of REDMDL for different n values on ransomware variant test set

API 序列长度	P	R	$F1$
1000	0.900	0.982	0.939
2000	0.930	0.964	0.947
3000	0.963	**0.987**	0.975
4000	**1.000**	0.975	**0.987**

为评估 REDMDL 的检测及时性,本文记录了 REDMDL 在实验中所需的检出时间(序列采集时间与检测时间之和),结果如表 7 所列。表 7 中序列采集时间为 REDMDL 中设定的用于采集 API 序列的时长(为本文训练集中各勒索软件样本调用表 4 中 API 达到长度为 n 的序列所需的平均时长),检测时间为 REDMDL 检测勒索软件变种样本集中单个样本所花费的最大时长。如表 7 所列,随着 n 值的增大,REDMDL 对相应长度 API 序列的采集时间及最终的检出时间也在增加。为了使得 REDMDL 在检出时间内满足“勒索软件关键检测时间段”[26]的要求且能够获得较高的检测精度,本文建议 REDMDL 中 n 的取值范围为[3000,4000]且其默认值可设定为 4000。表 7 显示,当 n 为 4000 时,REDMDL 检测变种样本集中单个待测样本所需要的最长检出时间仅为 5.513 s,这表明 REDMDL 能够在一款软件运行后数秒内高效判别其是勒索软件还是良性软件。

表 7 REDMDL 在不同 n 下检测勒索软件变种样本所需的时间

Table 7 Time needed for REDMDL to detect ransomware variants for different n values

(单位:s)

API 序列长度	列采集时间	检测时间	检出时间
1000	2.386	0.093	2.479
2000	2.894	0.125	3.019
3000	4.604	0.159	4.763
4000	5.322	0.191	5.513

为了进一步分析 REDMDL 对勒索软件的检测效果,本文从 Accuracy 和检出时间两方面将其与 Qin 等[35]提出的基于 TextCNN 的检测方法(输入的 API 序列长度设定为 4000 以与 REDMDL 相同)、Chen 等[26]提出的早期检测方法 REDMS(所用分类算法、N-gram、API 采集时间分别设定为其文中获取最佳检测精度的 RF 算法、4-gram、7s)在本文勒索软件变种测试集上进行对比实验,结果如表 8 所列。

表 8 不同方法在勒索软件变种测试集上的 *Accuracy* 和检出时间

Table 8 *Accuracy* and detection time of different methods on ransomware variant test set

方法	*Accuracy*/%	检出时间/s
TextCNN[35]	97.1	**5.338**
REDMS[26]	98.1	7.004
REDMDL	**98.2**	5.513

由表 8 可知,与 TextCNN 相比,REDMDL 中使用的 CNN-LSTM 模型所获得的 *Accuracy* 值高出 1.1%,而所需的检出时间仅略长 0.175 s;与 REDMS 相比,REDMDL 的

Accuracy 值和检出时间均更佳。

5.4.2 未知勒索软件检测结果

REDMDL 在不同 n 下检测未知勒索软件测试集所得到的 *Accuracy* 值如图 4 所示。图 4 显示，当 n 为 4 000 时，REDMDL 的 *Accuracy* 值达到了 96.3%。

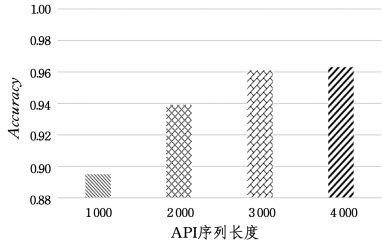

图 4 REDMDL 在不同 n 下检测未知勒索软件测试集所得到的 *Accuracy*

Fig. 4 *Accuracy* of REDMDL for detecting unseen ransomware test set for different n values

REDMDL 在不同 n 下检测未知勒索软件样本集所得到的 P，R 以及 $F1$ 如表 9 所列。由表 9 可知，当 n 为 4 000 时，REDMDL 所得到的 P，R，$F1$ 均为 97.5%。该实验结果表明，REDMDL 利用软件运行初期调用的一定长度 API 序列能够高精度地区分实验中的未知勒索软件和良性软件。

表 9 REDMDL 在不同 n 下检测未知勒索软件测试集所得到的 P，R 以及 $F1$

Table 9 P，R，and $F1$ of REDMDL for detecting unseen ransomware test set for different n values

API 序列长度	P	R	$F1$
1 000	0.867	0.945	0.904
2 000	0.944	0.971	0.957
3 000	0.963	**0.975**	0.969
4 000	**0.975**	**0.975**	**0.975**

在检测及时性方面，表 10 列出了 REDMDL 检测未知勒索软件测试集中单个样本所需的序列采集时间、检测时间和检出时间。由表 10 可以看到，当 REDMDL 使用 n 为 4 000 的 API 序列时，其检测未知勒索软件测试集中单个待测样本所需要的检出时间为 5.513 s，即至多需要 5.513 s 即可判别该测试集中单个待测样本是勒索软件还是良性软件。

表 10 REDMDL 在不同 n 下检测未知勒索软件测试样本所需时间

Table 10 Times needed for REDMDL to detect unseen ransomware sample for different n

（单位：s）

API 序列长度	序列采集时间	检测时间	检出时间
1 000	2.386	0.108	2.494
2 000	2.894	0.142	3.036
3 000	4.604	0.158	4.762
4 000	5.322	0.191	5.513

为了进一步评估 REDMDL 对未知勒索软件的检测效果，本文继续从 *Accuracy* 和检出时间两方面将 REDMDL，REDMS[26] 和 TextCNN[35] 在本文的未知勒索软件测试集上进行对比实验，结果如表 11 所列。

表 11 不同方法在未知勒索软件测试集上的 *Accuracy* 和检出时间

Table 11 *Accuracy* and detection time of different methods on unseen ransomware test set

方法	*Accuracy*/%	检出时间/s
TextCNN[35]	94.6	**5.340**
REDMS[26]	95.2	7.004
REDMDL	**96.3**	5.513

由表 11 可知，REDMDL 在未知勒索软件测试集上能够以 5.513s 的检出时间获得 96.3% 的 *Accuracy* 值，在 *Accuracy* 值上优于 TextCNN 和 REDMS，在检出时间上略逊于 TextCNN，但优于 REDMS。

结束语 本文通过分析勒索软件与良性软件运行初期所调用一定长度 API 序列的差异性，提出了一种基于深度学习的勒索软件早期检测方法 REDMDL。实验结果显示，REDMDL 使用长度为 4 000 的 API 序列时能够分别以 98.2%，96.3% 的 *Accuracy* 值检测出实验中的勒索软件变种样本和未知勒索软件样本。本文实验结果表明，软件运行初期所调用的一定长度 API 序列可以用于高效区分勒索软件和良性软件，在勒索软件早期检测领域具有巨大的应用价值。本文方法尚未能实现对所检出勒索软件样本所属家族进行识别。在后续工作中，我们将致力于此项工作以及进一步缩短勒索软件检出时间。

参 考 文 献

[1] GREENGARDS. The worsening state of ransomware[J]. Communications of the ACM, 2021, 64(4): 15-17.

[2] MOUSSAILEB R, CUPPENS N, LANET J L, et al. A Survey on Windows-based Ransomware Taxonomy and Detection Mechanisms[J]. ACM Computing Surveys, 2022, 54(6): 1-36.

[3] FreeBuf. FBI 透露，近六年里支付给勒索攻击者的赎金超过 1.4 亿美金[EB/OL]. (2020-02-28) [2021-10-27]. https://www.freebuf.com/news/228665.html.

[4] WANG H Z, CHEN J, CHEN X Y, et al. An Android Ransomware Detection Scheme Based on Evidence Chain Generation [J]. Chinese Journal of Computers, 2018, 41(10): 2344-2358.

[5] HWANG J, KIM J, LEE S, et al. Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques [J]. Wireless Personal Communications, 2020, 2020(2): 1-13.

[6] YILMAZ Y, CETIN O, ARIEFB, et al. Investigating the impact of ransomware splash screens[J]. Journal of Information Security and Applications, 2021, 61: 102934.

[7] 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: 144-166.

[8] ZIMBA A, WANG Z S, CHISHIMBA M. Addressing Crypto-Ransomware Attacks: Before You Decide whether To-Pay or Not-To[J]. Journal of Computer Information Systems, 2021, 61(1): 53-63.

[9] XIA T, SUN Y, ZHU S, et al. Toward A Network-Assisted Approach for Effective Ransomware Detection[J/OL]. EAI Endorsed Transactions on Security and Safety, 2021, 7(24): e3. https://eudl.eu/doi/10.4108/eai.28-1-2021.168506.

[10] HAYES K. Ransomware: a growing geopolitical threat [J]. Network Security, 2021, 2021(8): 11-13.

[11] BAJPAI P, ENBODY R. Dissecting. NET ransomware: key generation, encryption and operation[J]. Network Security, 2020, 2020(2): 8-14.

[12] LIU H, GUO C, CUI Y, et al. 2-SPIFF: a 2-stage packer identification method based on function call graph and file attributes [J]. Applied Intelligence, 2021, 51(12): 9038-9053.

[13] SUN G S, QIAN Q. Deep Learning and Visualization for Identifying Malware Families[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 18(18): 283-295.

[14] KHAMMAS B. Ransomware Detection using Random Forest Technique[J]. ICT Express, 2020, 6(4): 325-331.

[15] GUO C, CHEN C Q, SHEN G Y, et al. A Ransomware Classification Method Based on Visualization [J]. Netinfo Security, 2020, 20(4): 31-39.

[16] 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.

[17] XIAO W, ZHANG B, 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.

[18] HSU C M, YANG C C, CHENG H H, et al. Enhancing File Entropy Analysis to Improve Machine Learning Detection Rate of Ransomware[J]. IEEE Access, 2021, 9: 138345-138351.

[19] KHARRAZ A, ARSHAD S, MULLINERC, et al. UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware [C]// USENIX Security Symposium. Austin: Association, 2016: 757-772.

[20] RAMESH G, MENEN A. Automated dynamic approach for detecting ransomware using finite-state machine[J/OL]. Decision Support Systems, 2020, 138: 113400. https://www.sciencedirect.com/science/article/abs/pii/S016792362030155X?via%3Dihub.

[21] PEÑA A J, ULLAH F, JAVAID Q, et al. Modified Decision Tree Technique for Ransomware Detection at Runtime through API Calls[J]. Scientific Programming, 2020, 2020: 8845833.

[22] DAKU H, ZAVARSKY P, MALIK Y. Behavioral-Based Classification and Identification of Ransomware Variants Using Machine Learning[C]// 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications. Washington: IEEE Computer Society, 2018: 1560-1564.

[23] SCAIFE N, CARTER H, TRAYNOR P, et al. Cryptolock (and drop it): stopping ransomware attacks on user data[C]// 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS). Washington: IEEE Computer Society, 2016: 303-312.

[24] 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.

[25] 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.

[26] CHEN C Q, GUO C, CUI Y H, et al. Ransomw-are Early Detection Method Based on Short API Sequence[J]. Acta Electronica Sinica, 2021, 49(3): 586-595.

[27] VIVEKANANDAN K, PRAVEENA N. Hybrid convolutional neural network (CNN) and long-short term memory (LSTM) based deep learning model for detecting shilling attack in the social-aware network[J]. Journal of Ambient Intelligence and Humanized Computing, 2021, 12(1): 1197-1210.

[28] BORAH P, BHATTACHARYYA D K, KALITA J K. Cost Effective Method for Ransomware Detection: An Ensemble Approach[C]// International Conference on Distributed Computing and Internet Technology 2021. Washington: IEEE Computer Society, 2021, 12582: 203-219.

[29] Al-RIMY B, MAAROF M A, SHAID S. Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection[J]. Future Generation Computer Systems, 2019, 101: 476-491.

[30] LI Z Q, LI T. Query-by-Example with Acoustic Word Embeddings Using wav2vec Pretraining[J]. Computer Science, 2022, 49(1): 59-64.

[31] SUDHAKAR, KUMAR S. MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things[J]. Future Generation Computer Systems, 2021, 125: 334-351.

[32] YOUSFI S, RHANOUI M, MIKRAM M. Comparative Study of CNN and LSTM for Opinion Mining in Long Text[J]. Journal of Automation, 2019, 14(3): 50-55.

[33] LIU X X, JI Y, LIU C P. Voiceprint Recognition Based on LSTM Neural Network[J]. Computer Science, 2021, 48(S2): 270-274.

[34] HOCHREITER S, SCHMIDHUBER J. Long Short-Term Memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[35] QIN B, WANG Y, MA C. API Call Based Ransomware Dynamic Detection Approach UsingTextCNN[C]// 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE). Washington: IEEE Computer Society, 2020: 162-166.

LIU Wenjing, born in 1996, postgraduate, is a member of China Computer Federation. Her main research interests include network and information security.

GUO Chun, born in 1986, Ph. D, associate professor, is a member of China Computer Federation. His main research interests include malware analysis, data mining and intrusion detection.

(责任编辑:喻藜)