工程科学学报
Chinese Journal of Engineering

基于强化学习的工控系统恶意软件行为检测方法

高洋 王礼伟 任望 谢丰 莫晓锋 罗熊 王卫苹 杨玺

Reinforcement learning-based detection method for malware behavior in industrial control systems

GAO Yang, WANG Li-wei, REN Wang, XIE Feng, MO Xiao-feng, LUO Xiong, WANG Wei-ping, YANG Xi

引用本文:
高洋, 王礼伟, 任望, 谢丰, 莫晓锋, 罗熊, 王卫苹, 杨玺. 基于强化学习的工控系统恶意软件行为检测方法[J]. 工程科学学报, 2020, 42(4): 455–462. doi: 10.13374/j.issn2095-9389.2019.09.16.005
GAO Yang, WANG Li-wei, REN Wang, XIE Feng, MO Xiao-feng, LUO Xiong, WANG Wei-ping, YANG Xi. Reinforcement learning-based detection method for malware behavior in industrial control systems[J]. *Chinese Journal of Engineering*, 2020, 42(4): 455–462. doi: 10.13374/j.issn2095-9389.2019.09.16.005

在线阅读 View online: https://doi.org/10.13374/j.issn2095-9389.2019.09.16.005

工程科学学报, 第 42 卷, 第 4 期: 455−462, 2020 年 4 月
Chinese Journal of Engineering, Vol. 42, No. 4: 455−462, April 2020
https://doi.org/10.13374/j.issn2095-9389.2019.09.16.005; http://cje.ustb.edu.cn

基于强化学习的工控系统恶意软件行为检测方法

高　洋[1)], 王礼伟[2,3,4)], 任　望[1)], 谢　丰[1)], 莫晓锋[2,3,4)], 罗　熊[2,3,4)]✉, 王卫苹[2,3,4)], 杨　玺[5)]

1) 中国信息安全测评中心, 北京 100085　2) 北京科技大学计算机与通信工程学院, 北京 100083　3) 北京科技大学人工智能研究院, 北京 100083　4) 材料领域知识工程北京市重点实验室, 北京 100083　5) 北京市智能物流系统协同创新中心, 北京 101149

✉通信作者, E-mail: xluo@ustb.edu.cn

摘　要　网络环境下的恶意软件严重威胁着工控系统的安全, 随着目前恶意软件变种的逐渐增多, 给工控系统恶意软件的检测和安全防护带来了巨大的挑战. 现有的检测方法存在着自适应检测识别的智能化程度不高等局限性. 针对此问题, 围绕威胁工控系统网络安全的恶意软件对象, 本文通过结合利用强化学习这一高级的机器学习算法, 设计了一个检测应用方法框架. 在实现过程中, 根据恶意软件行为检测的实际需求, 充分结合强化学习的序列决策和动态反馈学习等智能特征, 详细讨论并设计了其中的特征提取网络、策略网络和分类网络等关键应用模块. 基于恶意软件实际测试数据集进行的应用实验验证了本文方法的有效性, 可为一般恶意软件行为检测提供一种智能化的决策辅助手段.

关键词　恶意软件; 检测方法; 强化学习; 特征提取; 策略网络

分类号　TP273

Reinforcement learning-based detection method for malware behavior in industrial control systems

GAO Yang[1)], *WANG Li-wei*[2,3,4)], *REN Wang*[1)], *XIE Feng*[1)], *MO Xiao-feng*[2,3,4)], *LUO Xiong*[2,3,4)]✉, *WANG Wei-ping*[2,3,4)], *YANG Xi*[5)]

1) China Information Technology Security Evaluation Center, Beijing 100085, China

2) School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China

3) Institute of Artificial Intelligence, University of Science and Technology Beijing, Beijing 100083, China

4) Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing 100083, China

5) Beijing Intelligent Logistics System Collaborative Innovation Center, Beijing 101149, China

✉ Corresponding author, E-mail: xluo@ustb.edu.cn

ABSTRACT　Due to the popularity of intelligent mobile devices, malwares in the internet have seriously threatened the security of industrial control systems. Increasing number of malware attacks has become a major concern in the information security community. Currently, with the increase of malware variants in a wide range of application fields, some technical challenges must be addressed to detect malwares and achieve security protection in industrial control systems. Although many traditional solutions have been developed to provide effective ways of detecting malwares, some current approaches have their limitations in intelligently detecting and recognizing malwares, as more complex malwares exist. Given the success of machine learning methods and techniques in data analysis applications, some advanced algorithms can also be applied in the detection and analysis of complex malwares. To detect malwares and

收稿日期: 2019−09−15

基金项目: 国家自然科学基金资助项目 (U1736117, U1836106); 北京市自然科学基金资助项目 (19L2029, 9204028); 北京市智能物流系统协同创新中心开放课题资助项目 (BILSCIC-2019KF-08); 北京科技大学顺德研究生院科技创新专项资金资助项目 (BK19BF006); 材料领域知识工程北京市重点实验室基本业务费资助项目 (FRF-BD-19-012A)

consider the advantages of machine learning algorithms, we developed a detection framework for malwares that threatens the network security of industrial control systems through the combination of an advanced machine learning algorithm, i.e., reinforcement learning. During the implementation process, according to the actual needs of malware behavior detection, key modules including feature extraction, policy, and classification networks were designed on the basis of the intelligent features of reinforcement learning algorithms in relation to sequence decision and dynamic feedback learning. Moreover, the training algorithms for the above key modules were presented while providing the detailed functional analysis and implementation framework. In the application experiments, after preprocessing the actual dataset of malwares, the developed method was tested and the satisfactory classification performance for malware was achieved that verified the efficiency and effectiveness of the reinforcement learning-based method. This method can provide an intelligent decision aid for general malware behavior detection.

KEY WORDS malware; detection method; reinforcement learning; feature extraction; policy network

在国家信息安全的范畴中, 与国家基础设施相关联的工业控制系统的安全占据着极其重要的地位[1]. 目前, 互联网环境下的工控系统安全是一个重要的关注点, 对其中非正常行为的监测主要围绕网络层面和使用端分别进行防护. 网络端防护通常是指网络流量进行分析, 主要是指对工业控制系统的特定协议、各种终端(包括移动终端)的流量异常等进行分析; 使用端的防护主要是识别样本的异常行为来分析. 本文主要研究终端方的工控系统安全防护问题, 对其中的恶意软件进行检测和分析, 以提高监测预警的准确性和可用性. 当前应对的方法主要基于人工线下检查, 对于存在的安全隐含, 还是以讨论解决方案为主, 并不具备智能化特点. 因此, 研发恶意软件的智能检测算法, 具有重要的现实意义[2].

一般来说, 可以通过提取恶意软件的特征码来判断该程序是否属于某一类已知的恶意软件, 虽然这样对已知特征的恶意软件能有很高的识别率, 但对于恶意软件的某些新型变种而言, 检测效果可能不太理想[3]. 从长远的角度来看, 随着新型恶意软件的变种不断出现, 病毒库将会变得日渐臃肿. 因此通过更新病毒库来检测新型变种的方法, 虽然可行, 但总是落后于恶意软件发展和更新的速度. 因此, 我们需要寻找新的能够识别恶意程序的有效方法. 随着深度学习等高级算法在众多领域上的成功应用, 于是大量的研究者开始尝试使用这些新兴技术来解决这一问题.

深度学习模型能够分析较长的系统调用序列, 并通过捕捉高层次的特征为语义学习做出更好的决策. Xiao 等[4] 利用前向神经网络和循环神经网络对权限申请和系统调用进行特征分析, 从而检测恶意软件. Su 等[5] 将多层次的特征输入到深度置信网络这一深度学习模型, 进而将学习到的恶意软件的典型特征输入到基于支持向量机的恶意软件检测器. 但是, 当新的恶意软件变种加入到训练数据集时, 模型需要逐步重新训练, 所以需要更多的计算时间来估计检测的概率. 基于深度学习的新解决方法虽然在计算准确率方面有了较大提升, 但是存在着计算效率不够理想和智能自适应学习能力不强等局限性. 基于上述考虑, 本文主要基于强化学习这一高级机器学习算法来实现恶意软件行为的检测与识别.

强化学习是机器学习的范式和方法论之一, 可针对智能体(Agent), 描述和解决其在与环境的交互过程中, 通过学习策略以达成回报最大化或实现特定目标的问题, 强化学习算法在信息论、博弈论、自动控制等领域有一定应用[6]. 但是, 将强化学习方法直接用于恶意软件检测方面的工作还很少. 目前虽然已有一些工作[7], 但为了进一步提升检测效率和智能化程度, 考虑采用本文的解决方法. 虽然恶意软件自身的内部程序结构有差异, 但其恶意行为最终必须落实到实际的动态行为中. 因此本文决定使用行为序列分析方法对恶意软件的行为序列进行分析, 提取出合适的行为特征, 然后通过这些特征, 对恶意软件进行判别和分类. 由于这种基于序列特征的分析操作, 与强化学习适用于序列决策特征相符合[8]. 这种需要对序列数据进行逐步操作的问题, 可以使用强化学习中的策略梯度来解决.

基于上述分析, 本文针对恶意软件检测这一特殊应用背景, 设计了一种基于强化学习模型的应用实现方法框架, 并完成了应用测试.

1 理论基础

1.1 恶意软件检测的分析方法

目前用于分析恶意软件特征的主流方法有两大类: 静态分析与动态分析.

在静态分析方法方面, 可以直接基于恶意程

序的可执行文件进行分析，也可以将恶意程序的可执行文件进行反编译以此获得程序更为底层的特征，然后对其进行分析．例如，Schultz 等[9] 首次将数据挖掘的相关算法应用于恶意软件分析，其通过朴素贝叶斯分类算法得到的分类结果，比基于特征匹配的传统方法具有更高的准确率．Santos 等[10] 则是基于操作码序列来进行分析，他们根据操作码的出现频率以及操作码之间的关联性来对恶意软件进行识别和分类．近年来，Zhang 等[11] 将可执行文件反编译得到操作码序列，然后将这些序列转换成图像的形式，最后通过卷积神经网络 CNN 来进行进一步的特征提取和识别．

在动态分析方法方面，该方法主要根据恶意软件运行时的行为特征来进行分析．这些动态特征包括：系统调用、文件读写操作、网络通信行为以及进程行为等．例如，Tandon 和 Chan[12] 使用规则学习算法的变体来学习系统调用中的规则信息，以此来检测新型的恶意软件行为．另外，考虑到动态分析方法中，频繁进行数据的采集，因此，需要有合适的行为采集工具，Willems 等[13] 开发了 CWSandbox 沙箱，它可以实现自动快速分析 Win32 平台上的恶意软件，极大提升了分析效率．基于此沙箱，Rieck 等[14] 构建了一个恶意软件分析框架，将恶意软件放入沙箱中运行，然后采集恶意软件运行时的行为特征，然后，基于这些动态特征，使用机器学习算法进行识别和分类．此外，Ki 等[15] 通过构建 API（Application program interface）特征数据库，利用对比 API 序列特征的方法来判断是否属于恶意软件．

1.2　强化学习

强化学习近年来引发了广泛的关注．一个基本的强化学习模型包含智能 Agent 和环境两部分．智能 Agent 通过观察环境的当前状态，然后根据相应的策略选择方案选择一个动作，将其作用到环境上．环境将根据这个动作改变自己的状态，并反馈给智能 Agent 一个奖励值．智能 Agent 根据这个奖励值来判断它所选取动作的优劣程度，并以此更新自己的策略．在不断交互过程中，智能 Agent 将逐渐倾向于选择奖励值高的动作．最终，找到一个能够完成当前目标，并能获得高奖励的方案[16]．

强化学习根据策略学习的方式可以分为两大类：基于价值的强化学习以及基于策略的强化学习．在基于价值的强化学习方法中，与环境进行交互时，每次都将尝试选择价值最大的动作．基于策略的强化学习，则主要基于概率分布来选择动作．在本文中，主要采用了基于策略的强化学习算法．相对于基于价值的强化学习而言，基于策略的方法可以更加方便地产生一连串的连续动作．

强化学习在自然语言处理中展示了很好应用效果．例如，Zhang 等[17] 提出了强化学习框架下的信息提取长短期记忆网络 ID-LSTM（Information distilled long short-term memory）模型，可提取到对分类问题更有效的序列特征．目前恶意软件动态分析主要基于系统调用序列来进行，由于强化学习在产生动作序列方面具有的优势，本文将基于上述工作，将 ID-LSTM 模型迁移到恶意软件行为分析领域，实现优化分析．

2　基于强化学习的检测方法

本文将基于强化学习的自然语言处理方法迁移到恶意软件行为特征分析领域．借助于强化学习具有序列决策和可根据反馈调整策略的特点，来筛选恶意软件行为序列，从而获得更好的行为序列特征．利用得到的特征，可以实现恶意软件检测分类应用．

2.1　方法框架

整个强化学习模型的总体结构，如图 1 所示．其中，$X=[x_1,x_2,\cdots,x_L]$是序列数据，表示程序实际执行时的一段 API 调用序列，其中的分量 x_i 是一个经过编码后的 API 调用函数；s_t 代表特征提取网络当前时刻的状态，是策略网络用来做出决策的判断依据；a_t 代表策略网络当前所选择的动作，该动作将影响特征提取网络的提取行为；h_L 是整个序列最终提取出来的特征，用于提供给分类网络，分类网络根据这个特征来进行分类选择；R_L 是延迟奖励，它在训练过程中，基于分类网络的预测结果来调整策略网络的参数．

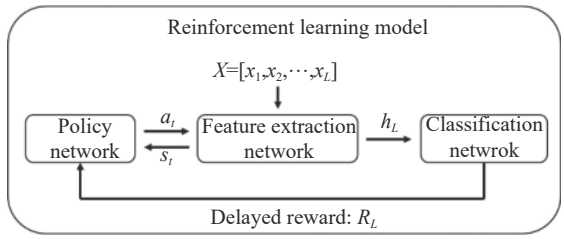

图 1　总体结构

Fig.1　Framework

在这个方法框架中，序列数据 X 中的每一个 API 调用函数将会按顺序依次输入到特征提取网络中，每当特征提取网络接收到一个 API 调用

函数 x_i 之后, 便会产生一个当前状态 s_t, 并将其送给策略网络. 策略网络根据当前的状态 s_t, 进行决策从而选择一个动作 a_t, 反馈给特征提取网络. 特征提取网络根据动作 a_t, 进行相应的更新操作. 不断重复上述过程, 当整个序列 X 被处理完之后, 特征提取网络将最终得到的特征 h_L 提供给分类网络, 分类网络根据这个提取出来的序列特征进行分类预测. 最后根据预测的结果计算一个延迟奖励 R_L 反馈给策略网络, 以便策略网络更新网络参数.

2.2 数据预处理

在针对恶意软件行为序列处理分析过程中, 需要对收集的原始数据进行必要的预处理, 以方便后续模型的训练和测试. 整个数据预处理的流程分为三步.

(1)提取 API 调用序列: 从 Cuckoo 沙箱生成的原始报告中提取 API 序列数据, 并且将每一份报告(每一个样本)单独形成一个 API 调用序列文件.

(2)过滤 API 函数: 计算各个 API 函数的信息增益值, 然后设定一个阈值, 最后将所有信息增益值小于阈值的 API 函数剔除. 这样做的目的是, 一方面可以剔除对分类帮助不大的 API 函数, 另一方面也希望尽可能保留较多的原始信息.

(3)编码 API 序列: 将每一个 API 函数编码成一个独热向量, 再将整个 API 序列中的 API 函数替换成对应的独热向量即可.

2.3 模型设计

下面分别介绍图 1 中三个关键网络模块的设计过程.

2.3.1 特征提取网络

特征提取网络主要基于 LSTM 网络的功能特点来提取时序数据的长距离时序特征. 这里的状态 s_t 定义为: $s_t = c_{t-1} \oplus h_{t-1} \oplus x_t$, 其中, $\oplus$ 表示拼接操作, $c_{t\text{-}1}$ 是 t-1 时刻记忆单元中的信息, $h_{t\text{-}1}$ 是用于存放时序数据累积特征的张量, x_t 是当前输入的 API 调用函数. s_t 表示当前的状态, 一方面考虑了之前序列中所蕴涵的信息, 另一方面还考虑了当前的输入, 可全面地体现当前的具体状态. 动作 a_t 定义为: $a_t \in \{\text{Retain}, \text{Delete}\}$. 这是一个二元选择, 每一个时刻所选择的动作只有两种可能: 删除(Delete)和保留(Retain). Retain 操作是保留当前对应的 API 函数, 而 Delete 操作则是去除对应的 API 函数, 被去除掉的 API 函数将不会对最终的分类结果产生影响.

特征提取网络的工作过程如算法 1 所示.

算法 1: 特征网络提取过程

1: 初始化 c_0 和 h_0: c_0=0, h_0=0;
2: i=1;
3: while i<=L
4: 输入 API 调用函数 x_i;
5: 计算当前状态: s_i=c_{i-1}⊕h_{i-1}⊕x_i;
6: 将 s_i 传递给策略网络;
7: 获取策略网络所选择的动作 a_i;
8: if a_i=Delete then
9: h_i=h_{i-1}; c_i=c_{i-1};
10: else if a_i=Retain then
11: h_i 和 c_i 根据 LSTM 网络功能进行正常更新;
12: i=i+1;
13: 输出 h_L, 将其提供给分类网络.

2.3.2 分类网络

分类网络利用从特征提取网络那里得到的序列特征 h_L, 来预测当前样本的所属类别. 这里的分类网络设计为一个三层的神经网络(输入层、隐含层和输出层), 输入是序列特征 $h_L = [h_1, h_2, \cdots, h_m]$; 输出的 $P(y|X)$ 是一个概率分布, 用来描述当前序列 X 所属的可能类别. 因为本文目的是区分恶意软件和良性软件, 这是一个二分类问题, 所以 $y \in \{\text{benign}, \text{malicious}\}$.

在实际训练过程中, 分类网络所计算出来的概率分布, 一方面用来估计当前样本所属的类别, 另一方面需要利用这个概率分布来计算延迟奖励 R_L, 计算方法如下:

$$R_L = \log_2(P(c|X)) + \gamma \frac{L'}{L} + b \tag{1}$$

其中: $P(c|X)$ 的值是从分类网络预测结果 $P(y|X)$ 中获得; c 表示样本 X 的实际所属类别. L' 是被删除掉的 API 函数的个数(也就是执行 Delete 操作的次数); L 是原始的序列长度; γ 则是起到调节删除力度功能的超参数. 例如, 在相同的情况下, 如果 γ 的值越大, 延迟奖励 R_L 第二项的值就会越大, 那么策略网络所得到的奖励也随之变大, 因此策略网络就会更倾向于删除更多信息. 此外, 在延迟奖励中增加一个偏置项 b. 这个偏置项的作用是用来调节奖励值的正负情况.

2.3.3 策略网络

策略网络接收特征提取网络给出的状态 s_t, 然

后计算得到一个动作选择的概率分布, 最后基于这个概率分布, 使用随机策略来选择一个动作.

这里的策略网络也是一个三层的神经网络. 输入是当前状态 $s_t = [s_1, s_2, \cdots, s_n]$, 输出 $\pi(a_t|s_t; \Theta)$ 表示当前选择动作的策略, 这是一个概率值, 其中的 Θ 是当前策略网络的网络参数. 这里的 $\pi(a_t|s_t; \Theta)$ 计算得到一个介于 0 至 1 之间的实数, 用来表示进行 Delete 操作的概率. 在实际使用过程中, 策略网络的基本结构与分类网络相同, 只是用途不同而已, 一个用于策略选择, 另一个则是用于预测分类.

2.4 训练算法

当设计好网络模型结构之后, 需要为网络选择合适的损失函数. 损失函数的值用来评价当前网络参数与训练集数据的拟合程度, 损失函数的值越小, 说明拟合的效果越好. 在选择一个合适的损失函数的同时, 还需要设计一个恰当的优化算法.

2.4.1 特征提取网络和分类网络的损失函数

因为分类网络的分类依据是基于特征提取网络所提取出来的最终特征, 因此, 这两个模块在这里共享同一个损失函数. 由于分类网络的输出结果是一个概率分布, 因此选择使用交叉熵作为损失函数. 当参与计算的两个概率分布越接近, 那么交叉熵计算结果越小, 因此, 交叉熵可以用来衡量两个概率分布之间的相似程度, 基于此, 交叉熵可以做预测任务的损失函数. 相关计算公式如下:

$$\text{Loss} = \sum_{X\in D}\sum_{c\in C} Q(c,X)\log_2(P(c|X)) \tag{2}$$

其中: D 表示训练集; C 是类别的集合, 在本文中, 有 $C \in \{\text{benign}, \text{malicious}\}$; $Q(c,X)$ 是 One-Hot 形式的概率分布, 这有 $Q(\text{benign}, X) = 1$, $Q(\text{malicious}, X) = 0$; $P(c|X)$ 是分类网络计算输出的概率分布.

2.4.2 策略网络的目标函数

策略网络的目标是做出能够获得最大收益的决策. 考虑到做出的决策行为是基于概率的, 这是一个随机变量. 因此, 应该通过期望值来衡量所获得的奖励, 对此可以定义目标函数为:

$$\begin{aligned} J(\Theta) &= E[R_L(s_1a_1\cdots s_La_L)] = \sum_{s_1a_1\cdots s_La_L} \\ &P_\Theta(s_1a_1\cdots s_La_L)R_L = \sum_{s_1a_1\cdots s_La_L}\prod_t \pi(a_t|s_t;\Theta)R_L \end{aligned} \tag{3}$$

其中: $s_1a_1\cdots s_La_L$ 表示状态–动作序列; $P_\Theta(s_1a_1\cdots s_La_L)$ 表示状态–动作序列出现的概率; $R_L(s_1a_1\cdots s_La_L)$ 表示状态–动作序列所对应的延迟奖励值; $E[R_L(s_1a_1\cdots s_La_L)]$ 表示延迟奖励的期望值. 这里需要注意的是, t+1 时刻的状态完全由 t 时刻的状态和 t 时刻的动作所决定, 并且初始的状态是确定的, 因此 $p(s_1)$=1 且 $p(s_{t+1}|s_t;a_t) = 1$.

2.4.3 强化学习模型的训练算法

在训练需要的损失函数定义好之后, 就可以根据所需的优化目标来训练整个模型. 整个强化学习模型的训练可以被分为三个阶段: 两个预训练阶段以及正式训练阶段.

预训练过程如算法 2 所述.

算法 2: 预训练过程

```
初始化参数: 分块尺寸 batchsize, 阈值 threshold, 迭代次数上限 iterations;
将数据集均分成 n 块, 每块大小为 batchsize, 每一块记为 b_i (i=1,2,···,n);
预训练特征提取网络和分类网络:
    保持策略网络参数不变, 并且策略网络仅输出 Retain;
    整个训练集上的准确率记为 accuracy;
    for i=1 : n
        计算 b_i 上的准确率, 记为 acc;
        计算 b_i 上的损失, 更新网络参数;
        更新 accuracy (未经过训练的 b_i 上的准确率认为是 0);
    i=1;
    while accuracy<threshold
        计算 b_i 上的准确率, 记为 acc;
        计算 b_i 上的损失, 更新网络参数;
        更新 accuracy;
        i=i % n+1;
预训练策略网络:
    保持特征提取网络和分类网络参数不变;
    for i=1 : n
        计算 b_i 上的损失, 更新网络参数.
```

在预训练特征提取网络和分类网络时, 施加了两个附加条件: 1) 训练集中所有样本至少经过一次预训练; 2) 分类网络的预测准确率要高于事先设定的阈值. 设置这两个条件是出于两方面的考虑: 一方面要充分利用到整个训练集的信息, 另一方面要使得策略网络的预训练更加有针对性.

预训练完成之后, 正式训练过程如算法 3 所述.

算法 3: 正式训练过程

```
i=1;
```

2:　while i<iterations
3:　　计算 b_i 上的损失, 更新网络参数;
4:　　if i % 100=0 then
5:　　　保存当前网络参数到文件中;
6:　　i=i % n+1;

在正式训练过程中, 由于保存了训练过程中不同迭代次数的强化学习模型. 当正式训练结束之后, 分别用这些保存下来的网络参数初始化强化学习模型, 然后分别对测试集上的数据进行测试, 查看不同迭代次数下的强化学习模型的表现, 从中选择一个表现最好的模型.

3　实验结果

3.1　数据来源

实验选用的数据集包含了 2 万多条 Windows 平台下的 API 调用序列数据[15]. 在实际使用中, 从中选取了 3000 条良性软件的 API 调用序列和 2411 条恶意软件的 API 调用序列(共有 3 种不同类型的恶意软件).

3.2　数据预处理的结果

数据集中的每一条数据都是 API 调用序列, 但是不同数据之间序列长度不统一, 需要进行初步的处理. 这里包括两个步骤: 统计所有 API 的个数, 计算 API 函数的信息增益值, 删去低增益的 API 函数; 利用 One-Hot 编码将每一个原始序列数据编码成长度为 400 的序列数据. 在选取的数据集中, 根据信息增益由大到小排序. 经检查, 发现信息增益值低的 API 函数, 在良性与恶意样本中出现比率都较低. 因此, 这些 API 函数从信息增益的角度来说, 对于分类的贡献较小, 只有少部分的样本含有这些 API 函数. 这里所选取的信息增益阈值为 0.0001, 信息增益值低于该阈值的都将被去除. 经过这样的操作后, 数据集中的 API 函数, 由过滤前的 946 个, 调整为信息增益过滤之后的 828 个.

3.3　参数设置

在经过实验测试后, 强化学习模型训练时所需的参数设置为: 特征提取网络、策略网络、分类网络输入数据的维度分别为 828、1084、128; 特征提取网络、策略网络、分类网络中隐含层神经元数目分别为 128、128、128; 分类网络输出张量的维度为 2; 超参数 γ 为 0.4; 特征提取网络、策略网络、分类网络的学习率为 0.005、0.01、0.005; 延迟奖励的偏置项为 0.8; 每次训练的样本数目为 16; 预训练时的准确率阈值为 0.6; 正式训练时的迭代学习次数上限为 4000. 另外, 在本实验的数据集中, 测试集占 70%(其中, 恶意样本和良性样本各有 700 个), 测试集占 30%(其中, 恶意样本和良性样本各有 300 个).

3.4　算法学习结果

如图 2 所示, 当强化学习模型在训练集上迭代 500 次之后, 测试集上的准确率接近于 90% 左右. 图 3 展示了测试集上查准率和召回率随迭代次数的变化情况, 可以看到准确率曲线(实线)的变化趋势, 和召回率曲线(虚线)的变化趋势往往相反. 在迭代 2000 次之前, 查全率能维持在 80% 左右, 查准率能接近 100%. 这里可以通过查看混淆矩阵, 来评价模型的计算性能. 如表 1 所示, 这里选择了迭代次数 2000 次后的模型, 其中, TP 表示正确预测恶意样本的数量, FP 表示错误预测恶意样本的数量, FN 表示错误预测良性样本的数量, TN 表示正确预测良性样本的数量. 基于此, 可以计算查准率为: P=TP/(TP+FP)=257/(257+4)=98.1%; 查全率为: R=TP/(TP+FN)=257/(257+43)=85.7%.

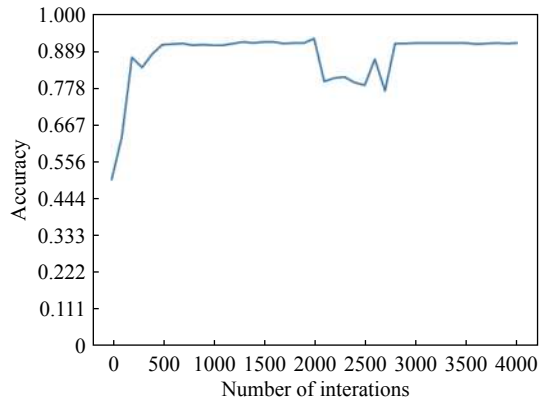

图 2　测试集上的准确率

Fig.2　Accuracy in the test dataset

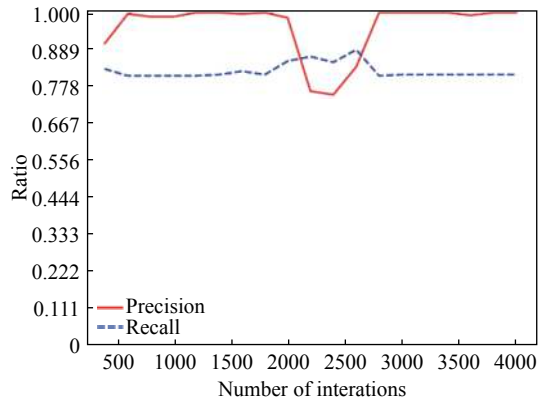

图 3　测试集上查准率和查全率随迭代次数的变化

Fig.3　Precision and recall in the test dataset

表 1　分类结果的混淆矩阵

Table 1　Confusion matrix

Confusion matrix	Prediction : malicious	Prediction : benign
Truth : malicious	257 (TP)	43 (FN)
Truth : benign	4 (FP)	296 (TN)

因为本文主要关注恶意样本的行为特征，在利用强化学习模型处理恶意样本时，更倾向于确定删除和保留哪些 API 函数. 因此，在测试集恶意样本中，统计出现次数超过 100 的 API 函数，并且按照删除比例（删除次数/总出现次数）来排序. 如表 2 中的前半部分和后半部分所示，这里分别选择了删除比例最高和最低的的 5 个 API 函数. 由表中的结果可知，VirtualAllocEx 等函数在实际分类过程中起到的作用比较次要，尤其是 VirtualAllocEx 函数对于强化学习模型而言，更容易被删除. 这说明，虽然 VirtualAllocEx 在大多数的恶意样本中都出现，但是对于分类来说，反而没有起到太大的贡献. 而 GetProcAddress 和 CloseHandle 等函数出现次数多，并且都未被删除，说明这些函数对于所训练的强化学习模型提出取来的特征的贡献是较为重要的. 从恶意软件行为的角度来看，这些 API 函数相当于是恶意样本中关键的行为.

表 2　删除比例最高和最低的各 5 个 API 函数

Table 2　Five API functions with the highest and lowest deletion rates

API Functions	Number of deleting operation	Number of retaining operation	Rate of deleting operation
VirtualAllocEx	174	209	0.454308
IsDBCSLeadByte	89	135	0.397321
GetSystemDirectoryA	101	206	0.328990
CreateThread	38	106	0.263889
GetDC	82	229	0.263666
GetProcAddress	0	2883	0
CloseHandle	0	2853	0
LocalFree	0	1939	0
GetModuleFileNameW	0	1485	0
lstrlenW	0	1460	0

4　结论

为了有效检测工控系统中的恶意软件行为特征，本文通过结合使用强化学习这一高级机器学习算法模型，设计了一个智能检测方法框架. 借助于强化学习具有序列决策和可根据反馈调整学习策略的特殊优势，对恶意软件行为序列进行了筛选，以获得有效的行为序列特征，并利用得到的特征，实现了恶意软件的检测分类应用. 围绕设计的方法框架，详细讨论和分析了其中的特征提取网络、策略网络和分类网络三个关键模块. 通过结合实际数据集进行的实验验证结果表明，文中设计的基于强化学习的检测方法，可在一定程度上，智能实现应用检测任务.

参　考　文　献

[1] Shi Y J. *Research on the Key Security Issues of Mobile and Open Industrial Control System*[Dissertation]. Beijing: Beijing University of Posts and Telecommunications, 2016
（时忆杰. 移动互联环境下工业控制系统安全问题研究[学位论文]. 北京: 北京邮电大学, 2016）

[2] Demontis A, Melis M, Biggio B, et al. Yes, machine learning can be more secure! A case study on android malware detection. *IEEE Trans Dependable Secure Comput*, 2019, 16(4): 711

[3] Sharif M, Lanzi A, Giffin J, et al. Impeding malware analysis using conditional code obfuscation // *Proceedings of the Network and Distributed System Security Symposium*. San Diego, 2008: 1939

[4] Xiao X, Wang Z, Li Q, et al. Back-propagation neural network on Markov chains from system call sequences: a new approach for detecting Android malware with system call sequences. *IET Inf Secur*, 2016, 11(1): 8

[5] Su X, Zhang D F, Li W J, et al. A deep learning approach to android malware feature learning and detection // 2016 *IEEE Trustcom/BigDataSE/ISPA*. Tianjin, 2016: 244

[6] Li G L, Gomez R, Nakamura K, et al. Human-centered reinforcement learning: a survey. *IEEE Trans Human Mach Syst*, 2019, 49(4): 337

[7] Wu C S, Shi J Y, Yang Y X, et al. Enhancing machine learning based malware detection model by reinforcement learning // *Proceedings of the 8th International Conference on*

Communication and Network Security. Qingdao, 2018: 74

[8] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature*, 2015, 518(7540): 529

[9] Schultz M, Eskin E, Zadok F, et al. Data mining methods for detection of new malicious executables // *Proceedings of the IEEE Symposium on Security and Privacy*. Oakland, 2001: 38

[10] Santos I, Brezo F, Ugarte-Pedrero X, et al. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Inf Sci*, 2013, 231: 64

[11] Zhang J X, Qin Z, Yin H, et al. IRMD: Malware variant detection using opcode image recognition // *Proceedings of the IEEE* 22*nd International Conference on Parallel and Distributed Systems*. Wuhan, 2016: 1175

[12] Tandon G, Chan P. Learning rules from system call arguments and sequences for anomaly detection // *Proceedings of the International Workshop on Data Mining for Computer Security*. Melbourne, 2003: 20

[13] Willems C, Holz T, Freiling F. Toward automated dynamic malware analysis using CWSandbox. *IEEE Secur Privacy*, 2007, 5(2): 32

[14] Rieck K, Trinius P, Willems C, et al. Automatic analysis of malware behavior using machine learning. *J Comput Secur*, 2011, 19(4): 639

[15] Ki Y, Kim E, Kim H K. A novel approach to detect malware based on API call sequence analysis. *Int J Distrib Sens Netw*, 2015, 11(6): 659101

[16] Busoniu L, Babuška R, De Schutter B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans Syst Man Cybern Part C Appl Rev*, 2008, 38(2): 156

[17] Zhang T Y, Huang M L, Zhao L, et al. Learning structured representation for text classification via reinforcement learning // *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. New Orleans, 2018: 6053