计算机科学
COMPUTER SCIENCE

针对系统调用的基于语义特征的多方面信息融合的主机异常检测框架

樊燚, 胡涛, 伊鹏

引用本文

樊燚, 胡涛, 伊鹏. 针对系统调用的基于语义特征的多方面信息融合的主机异常检测框架[J]. 计算机科学, 2024, 51(7): 380-388.

FAN Yi, HU Tao, YI Peng. Host Anomaly Detection Framework Based on Multifaceted Information Fusion of SemanticFeatures for System Calls [J]. Computer Science, 2024, 51(7): 380-388.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230400023

针对系统调用的基于语义特征的多方面信息融合的主机异常检测框架

樊 燚 胡 涛 伊 鹏
战略支援部队信息工程大学信息技术研究所 郑州 450002
(fy.fzh@foxmail.com)

摘 要 混淆攻击通过修改进程运行时产生的系统调用序列,可以在实现同等攻击效果的前提下,绕过主机安全防护机制的检测。现有的基于系统调用的主机异常检测方法不能对混淆攻击修改后的系统调用序列进行有效检测。针对此问题,提出了一种基于系统调用多方面语义信息融合的主机异常检测方法。从系统调用序列的多方面语义信息入手,通过系统调用语义信息抽象和系统调用语义特征提取充分挖掘系统调用序列的深层语义信息,利用多通道 TextCNN 实现多方面信息的融合以进行异常检测。系统调用语义抽象实现特定系统调用到其类型的映射,通过提取序列的抽象语义信息来屏蔽特定系统调用改变对检测效果的影响;系统调用语义特征提取利用注意力机制获取表征序列行为模式的关键语义特征。在 ADFA-LD 数据集上的实验结果表明,所提方法检测一般主机异常的误报率低于 2.2%,F1 分数达到 0.980;检测混淆攻击的误报率低于 2.8%,F1 分数达到 0.969,检测效果优于对比方法。

关键词:主机异常检测;系统调用语义信息融合;混淆攻击;深度学习;注意力机制
中图分类号 TP309

Host Anomaly Detection Framework Based on Multifaceted Information Fusion of Semantic Features for System Calls

FAN Yi,HU Tao and YI Peng
Information Technology Institute,Information Engineering University,Zhengzhou 450002,China

Abstract Obfuscation attack can bypass the detection of host security protection mechanism on the premise of achieving the same attack effect by modifying the system call sequence generated by the process running. The existing system call-based host anomaly detection methods cannot effectively detect the modified system call sequence after obfuscation attacks. This paper proposes a host anomaly detection method based on the fusion of multiple semantic information of system call. This method starts with the multiple semantic information of the system call sequence,fully mining the deep semantic information of the system call sequence through the system call semantic information abstraction and the system call semantic feature extraction,and uses the multi-channel TextCNN to realize the fusion of multiple information for anomaly detection. Semantic abstraction of system call can realize the mapping of specific system call to its type and shield the influence of specific system call change on detection effect by extracting sequence abstract semantic information. The system call semantic feature extraction uses the attention mechanism to obtain the key semantic features that represent the sequence behavior pattern. Experimental results on ADFA-LD dataset show that the false alarm rate of this method for detecting general host anomaly is lower than 2.2%,and the F1 score reaches 0.980. The false alarm rate of detecting the confusion attack is lower than 2.8%,and the F1 score reaches 0.969. Is detection performance is better than that of other methods.

Keywords Host anomaly detection,System call semantic information fusion,Obfuscation attack,Deep learning,Attention mechanism

1 引言

程序的系统调用序列记录了其运行时的完整行为信息,是监控程序行为模式和主机运行状态的重要指标,因此,基于系统调用的主机异常检测相比其他方法具有难以比拟的优势。但以混淆攻击[1]为代表的进化攻击方式却打破了这种优势,混淆攻击可以通过修改恶意程序运行时产生的系统调用来伪装自身,避免异常系统调用序列的生成,从而躲避入侵检测系统的检测。当前基于系统调用的入侵检测方法以基于 n-gram 子序列[2-4]、基于子序列频率特征[5-6],以及序列特征和频率特征相结合的混合检测方式为主,但是通过这些方法提取的特征都与系统调用子序列的模式密切相关,鲁棒性较差。使用文献[1]中的攻击方式对系统调用序列进行修改或替换后,将会大幅降低上述方法提取特征的有效性,导致

到稿日期:2023-04-05 返修日期:2023-08-01
基金项目:国家自然科学基金面上项目(62176264)
This work was supported by the National Natural Science Foundation of China(62176264).
通信作者:胡涛(hutaondsc@163.com)

检测进化攻击的性能不佳。

借鉴自然语言处理在计算机科学中的应用，可以将应用程序的系统调用序列抽象为一种程序的行为语言，将每个系统调用视作一个“单词”，将若干相关的系统调用子序列视作一条包含独立信息的“短语”，将程序一个生命周期中产生的一条系统调用序列视作一个“句子”。在一个“句子”中使用同义词替换、增删无用词等操作并不会影响句子的整体含义，换句话说，尽管攻击者可以修改系统调用序列，但并不能从本质上掩盖其攻击意图。因此，只要能够充分理解系统调用序列的语义信息表示，并提取其中蕴含的潜在抽象信息，便可以避免一些模糊混淆操作的影响，从而提高检测方法的精确性和鲁棒性。

基于上述判断，本文提出了一种基于系统调用多方面语义信息融合的主机异常检测方法。与以往方法不同的是，该方案在考虑系统调用序列原始信息的同时，融合了系统调用的类型信息和语义特征信息，利用系统调用序列包含的深层融合信息进行异常检测。这种方法在保证检测效果的同时，提升了检测的鲁棒性，适用于混淆技术等进化攻击。本文的主要贡献如下：

(1)分析并总结了基于混淆攻击的主机异常机理，对混淆攻击的特点及实现方式进行了介绍。

(2)提出了一种基于系统调用多方面语义信息融合的主机异常检测框架。该框架面向系统调用序列，并行开展语义信息抽象和语义特征提取以获得序列不同方面的信息，融合得到表征序列特征的深层信息，再利用多通道 TextCNN[7] 进行异常检测。

(3)在公开的数据集上与其他基于系统调用的主机异常检测方法及当前主流的深度学习模型进行了比较。实验结果表明，本文方法不仅能有效检测一般的主机异常，对混淆攻击同样具有较好的检测效果，检测效果明显优于现有方法。

2　相关工作

从 Forrest 等[8]的研究开始，基于系统调用的入侵检测受到广泛关注。Creech 等[9]创建了一种基于上下文无关文法的系统调用语义检测模型，该模型使用多个长度的滑动窗口扫描完整的系统调用序列，将扫描得到的子序列表示为单词，用这些单词组合成长度 1～5 的短语，并将它们输入极限学习机中进行训练和检测。该方法有效提高了检测率并降低了错误警报，但模型的训练却需要花费数周的时间。Murtaza 等[10]通过将系统调用跟踪表示为内核模块交互序列来减少异常检测器的执行时间，将单个系统调用映射到其相应的内核模块，从而将输入值的范围从大约 300 个系统调用减少到 8 个内核模块。这种方法实现了与 Creech 等[9]相当的误报率，但模型的运行速度更快。

Khreich 等[11]提出了一种基于可变 n-gram 特征的特征提取方法。该方法首先将系统调用轨迹分割成可变长度的多个 n-gram 序列，根据术语逆文档频率(Term Frequency-Inverse Document Frequency，TF-IDF)算法对序列中系统调用出现的频率进行加权计算，并将这些 n-gram 序列映射到固定大小的稀疏特征向量，然后使用一类支持向量机对这些向量进行训练和检测。Jiang 等[12]从正常系统调用序列中提取不同长度的 n-gram 子序列，并构建了一个表示正常行为的广义状态的自动机来检测未知序列中的异常行为。Chen 等[13]提出了一种基于集成学习的系统调用实时异常检测框架，该框架可实现对进程系统调用序列的实时分析及对进程行为模式的实时预警，在实时异常检测方面具有较高的准确率。

基于 n-gram 子序列以及一些混合检测的系统调用入侵检测方式的检测效果与精确匹配的 n-gram 子序列密切相关，对系统调用序列的变化十分敏感。对于利用混淆攻击产生的系统调用序列，这些方法的检测能力将显著下降。

Liao 等[14]将系统调用序列视为用每个单词(系统调用)的 TF-IDF 得分向量表示的文档，应用 K 近邻(KNN)算法进行异常检测。如果一个进程被归类为入侵性进程，那么它所属的整个会话也被视为攻击会话。Xie 等[15]结合 KNN 和 K-均值算法，改进了基于频率特征的 KNN 算法在 ADFA-LD 数据集上的检测性能。Liu 等[16]提出了一种基于统计模式的特征提取方法。该方法首先提取系统调用序列中的 n-gram 序列，再将 n-gram 序列转换为其对应的频率序列。然后，提取频率序列上的统计特征，借助一些统计学中的指标(第一四分位值、第二四分位值、偏度等)对其进行分析，使用 4 种不同的异常检测算法进行实验，结果表明，该方法适用于跨平台的异常检测。

基于序列频率的检测方式关注系统调用的频率特征，却忽视了序列的时序性和语义性。一个良性的系统调用序列“open()，read()，write()，close()，exit()”和一个恶意的系统调用序列“close()，open()，write()，read()，exit()”，序列中每个系统调用的频率相等，但序列的性质却有本质的差别，因此基于序列频率的检测方式仍存在一定程度的缺陷。

3　针对系统调用的基于语义特征的多方面信息融合的主机异常检测框架

3.1　威胁模型

本文中混淆攻击指文献[1]中提出的一种恶意攻击方法，它可以在实现原始攻击效果的前提下，修改恶意软件运行所生成的系统调用序列，从而躲避异常检测系统的检测。下面分别介绍两种常用的混淆攻击的实现方法[17]。

(1)添加语义上的“no-ops”。术语“no-ops”指没有效果的系统调用，或者其效果与攻击者的目标无关。例如，打开一个不存在的文件、打开一个文件并立即关闭、从打开的文件描述符中读取 0 字节并丢弃结果，这些操作并没有任何实际作用，是攻击者为了伪装良性序列而添加的。

(2)等效攻击。程序在执行一个操作时，有时可以使用不同的系统调用达到相同的目的。例如，打开一个文件获取文件描述符后，对该文件描述符的 read()系统调用可以替换为 mmap()系统调用；另一些情况下，可以改变恶意程序中某些操作的顺序，以改变对应的系统调用序列的顺序，但并不影响恶意程序本质意图的实现。

图 1 为混淆攻击中添加语义上的“no-ops”示例。如图所示，在原始的系统调用函数中，该程序事先调用 NtCreateFile 创建了一个文件并返回一个文件句柄 hlog，然后将函数返回值 hlog 作为参数传递给 NtClose。当使用混淆技术后，在原始的函数调用中插入了函数 *SetFilePointer*，其作用是移动

文件指针并返回新位置，函数的返回值等于移动距离加上起始点的偏移量，本例为 0(FILE_BEGIN)，该函数将指针移动距离设为与 hlog 相同的值。因此，*SetFilePointer* 的返回值等于 hlog，并将函数返回值 dwFpos 传递给 NtClose 以关闭文件。这前后的两种操作可以实现同样的效果，但是程序的系统调用序列却发生了变化。类似的混淆技术可以改变程序的系统调用序列，使得现有检测手段的效果大大降低甚至失效。

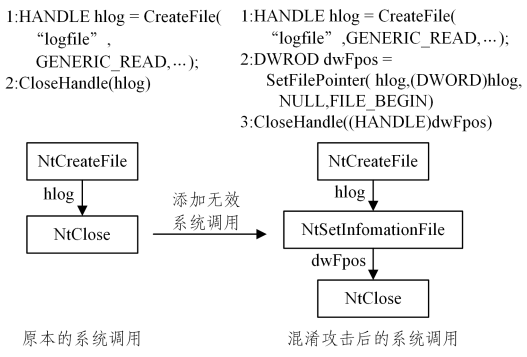


图 1 添加语义上的“no-ops”

Fig. 1 Add semantic “no ops”

图 2 是混淆攻击中等效攻击的示例。当攻击者获得一份恶意软件源代码后，可以利用 API 替换技术[18]修改恶意软件源代码来获取多种不同形式的恶意二进制程序，并通过互联网传播潜伏到主机环境中。由于原始恶意软件的 API 已经被等效替换了，因此那些被修改过的恶意程序在运行时会生成不同于原始恶意软件的系统调用序列。这种情况下，由于恶意程序系统调用序列的改变，一些传统的基于系统调用的检测方法就不能很好地发挥作用。

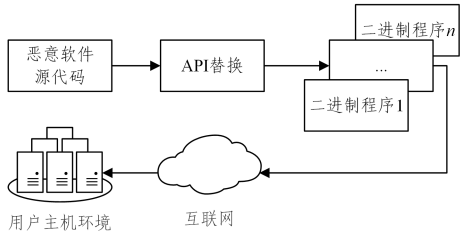


图 2 等效攻击

Fig. 2 Equivalent attack

3.2 框架概述

为了有效应对混淆攻击，本文提出了一种基于系统调用多方面语义信息融合的主机异常检测框架，其原理示意如图 3 所示。该框架主要分为 4 个模块。

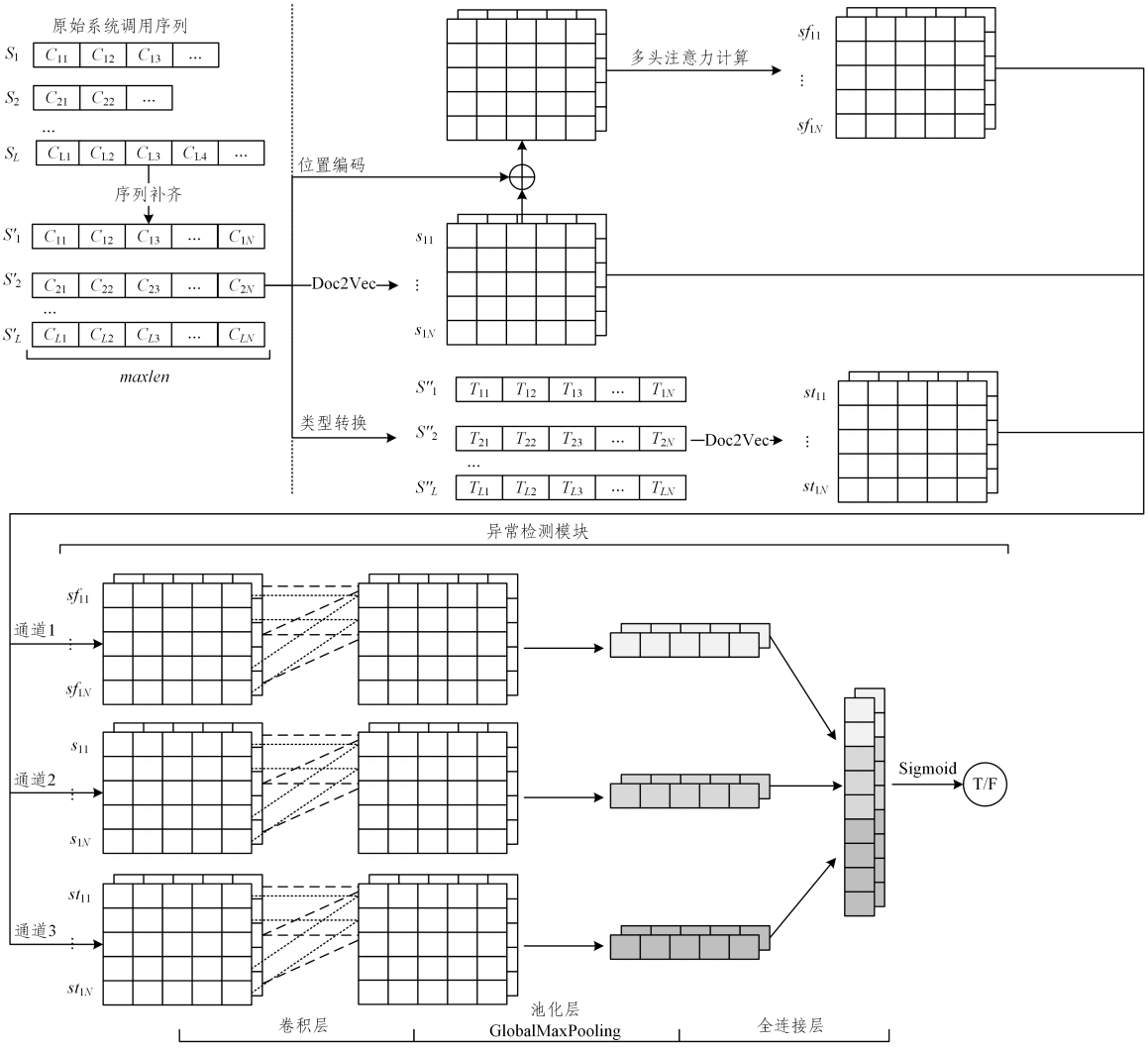


图 3 模型整体框架

Fig. 3 Overall framework of the model

(1)序列补齐：将原始的系统调用序列补齐到统一的长度，然后使用 Doc2Vec[19] 获取序列的嵌入表示，以保留序列原始信息及便于模型处理。

(2)语义信息抽象：将统一长度序列中每个系统调用转化为其对应的类型($C_x \rightarrow T_x$)，从而获得类型抽象序列，并使用 Doc2Vec 获得类型抽象序列的嵌入表示。

(3)语义特征提取：对统一长度序列进行位置编码，获取序列中系统调用的位置信息，并将位置编码信息与原始序列的词向量矩阵相加，再进行多头注意力计算，提取序列的关键语义特征。

(4)异常检测与分类：将上面得到的 3 种类型的嵌入表示矩阵作为输入，使用三通道 TextCNN 对这些信息进行处理，每一个 CNN 通道处理一种类型的信息，使用 Sigmoid 函数得出分类结果，确定输入序列是正常或异常。

图 4 为模型的工作流程。对于一个原始的系统调用序列，首先对其进行序列补齐得到统一长度的序列。然后分别对统一长度序列进行语义信息抽象和语义特征提取，得到序列的类型抽象信息和语义特征信息。最后将 3 种类型的信息输入 TextCNN 模型中，如果模型处在训练阶段，则使用数据对模型进行训练；否则用训练好的模型对序列进行检测，并输出检测结果。

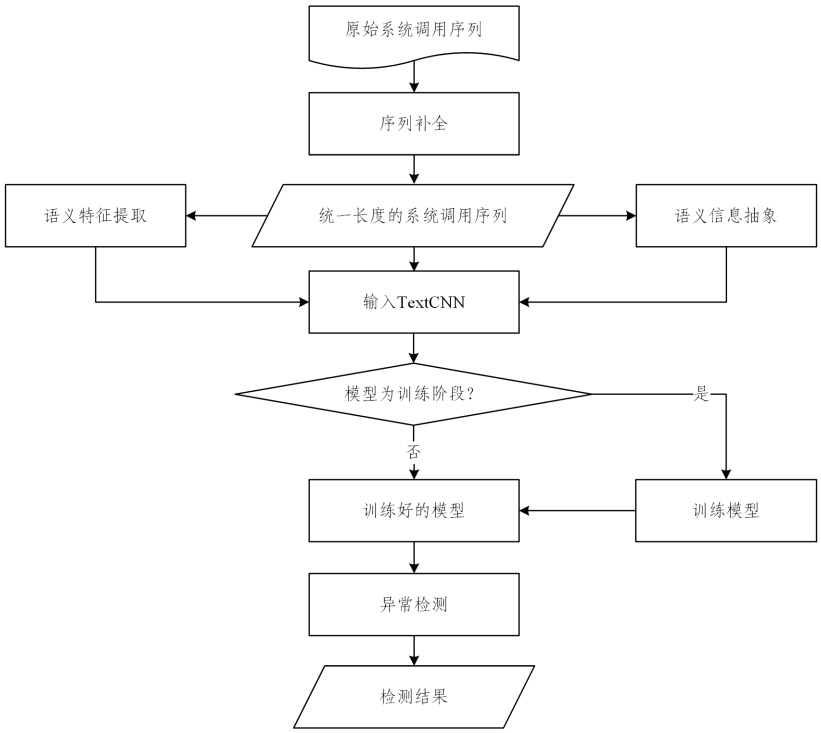


图 4　模型工作流程

Fig. 4　Model workflow

3.3　语义信息抽象

系统调用序列实际上是按时间排序的系统调用的集合，某个或某些系统调用的组合与进程所请求的操作相对应，操作系统中每个系统调用都有一个唯一的编号。操作系统所提供的功能都能在逻辑上抽象为若干功能模块，所有系统调用按照其功能角色的差异被组织成不同的功能模块。典型的 Linux 发行版中共有 8 个不同的功能模块[10]，如表 1 所列。

表 1　Linux 2.6.38 内核中的系统模块

Table 1　System modules in Linux 2.6.38 kernel

模块名	系统调用个数
文件系统	131
内核	127
进程间通信	7
网络	2
内存管理	21
安全	3
系统架构	10
其他	37

每个功能模块中包含若干不同类型的系统调用，同一类型的系统调用具有相近的功能，这些系统调用与其类型的映射关系可以在文件/usr/include/asm-generic/unistd.h 中找到。图 5 展示了 Linux 操作系统下的若干系统调用，它们都与进程间通信模块中共享内存的操作相关。

```
/* ipc/shm.c */
#define __NR__shmget 194
__SYSCALL(__NR_shmget,sys_shmget)
#define __NR__shmctl 195
__SYSCALL(__NR_shmctl,sys_shmctl)
#define __NR__shmat 196
__SYSCALL(__NR_shmat,sys_shmat)
#define __NR__shmdt 197
__SYSCALL(__NR_shmdt,sys_shmdt)
```

图 5　Linux 2.6.38 内核中与共享内存相关的系统调用

Fig. 5　Shared memory related system calls in Linux 2.6.38 kernel

根据以上信息，这里将编号为 194-197 的系统调用类型定为“ipc/shm”，其表示的系统调用类型为进程间通信的共享内存操作。可以将所有的系统调用划分为如表 2 所列的共 63 种类型。若有一个系统调用序列“syscall_1，sy-

$scall_2$,$syscall_3$”,则其对应的类型抽象序列为“$type_1$,$type_2$,$type_3$”。假设经混淆攻击后得到系统调用序列“$syscall_1$,$syscall_x$,$syscall_3$”,即使修改前后系统调用改变,但是其对应的类型仍可能不变,仍为“$type_1$,$type_2$,$type_3$”。因此,使用类型抽象可以有效应对混淆攻击。

按照上述分类规则,可以将待检测系统调用序列中的每个系统调用转换为其对应的抽象类型表示。转换时,需要构建一个包含系统调用与其类型映射关系的词典syscall2type,它以系统调用的编号为键,以系统调用对应的类型为值。那么,对于具有统一长度的系统调用序列 $s_i'=\{sc_1, sc_2, \cdots, sc_{maxlen}\}$,可以利用算法1将一个系统调用序列转换为其对应的类型抽象序列。

表2 系统调用的类型抽象表示

Table 2 Type abstract representation of system calls

模块名	系统调用类型			
文件系统	fs/xattr	fs/dcahe	fs/cookies	fs/splice
	fs/eventfd	fs/eventpoll	fs/fcntl	fs/timerfd
	fs/inotify_user	fs/ioctl	fs/ioprio	fs/stat
	fs/locks	fs/namei	fs/namespace	fs/utimes
	fs/nfsctl	fs/open	fs/pipe	fs/sync
	fs/quota	fs/readdir	fs/read_write	fs/signalfd
	fs/sendfile	fs/select		
内核	kernel/acct	kernel/capability	kernel/exec_domain	kernel/signal
	kernel/fork	kernel/exit	kernel/futex	kernel/sys
	kernel/hrtimer	kernel/itimer	kernel/kexec	kernel/time
	kernel/module	kernel/posix-timers	kernel/printk	kernel/sched
	kernel/ptrace	kernel/timer		
进程间通信	ipc/mqueue	ipc/msg	ipc/shm	ipc/sem
网络	net/socket			
内存管理	mm/filemap	mm/nommu	mm/fadvise	
安全	sc/keys_keyctl			
系统架构	arch/kernel_sys			
其他	无			

算法1 序列抽象算法

输入:统一长度的系统调用序列集合 S′,系统调用类型映射词典 syscall2type

输出:系统调用类型抽象序列 S″

1. $\forall s_i' \in S', \forall sc_j \in s_i'$
2. for s_i' in S′,i=1,2,…,L
3. for s c_j in s_i',j=1,2,…,maxlen
4. S″[i][j]=syscall2type(sc_j)
5. endfor
6. endfor

3.4 语义特征提取

从进程整个生命周期来看,其系统调用序列包含着进程的完整行为信息,即使攻击者可以利用各种攻击手段混淆系统调用序列,但只要其运行时有异常行为出现,其恶意入侵行为的本质意图和影响就会存在。可以将系统调用序列抽象为程序的行为语言,只要充分挖掘系统中所蕴涵的语义信息,就能有效地理解程序的行为意图,对进程的行为模式进行判别。

在得到统一长度序列的嵌入表示后,本文应用 Vaswani 等[20]提出的多头注意力机制来对系统调用序列进行进一步处理,以提取序列的语义特征。多头注意力包含自注意力和缩放点积注意力。自注意力机制是一种学习序列本身特征的方法,它可以捕获序列或数据内部的特征相关性。一个系统调用在序列中扮演的角色不仅与它出现在整个序列中的先后位置有关,还与它自身所在的局部序列紧密相关,且这些关系的紧密程度也有主次之分,只有充分理解这些关系,才能更好地提取序列中的语义特征。这种关系的计算可以通过自注意力来解决。如图6所示,104号系统调用($syscall_{104}$)与它周围的系统调用具有不同程度的紧密关系(以连线的粗细表示)。

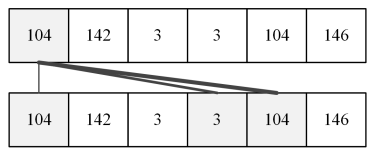


图6 自注意力原理示意

Fig. 6 Schematic diagram of self-attention

缩放点积注意力是自注意力的一种实现,它通过向量间的点积运算计算序列中每个符号(token)与其他符号的相似性关系(相似性越大,联系越紧密),以获取序列间符号的注意力表示。缩放点积注意力的计算方法如式(1)所示,其中 $\boldsymbol{Q}$(Query),$\boldsymbol{K}$(Key),$\boldsymbol{V}$(Value)是符号的嵌入表示经过一个矩阵变换得到,d_k 为符号嵌入表示的维度大小,$1/\sqrt{d_k}$ 是一个缩放因子,它可以避免向量点积过大而使得 softmax 函数的梯度变小,使模型训练时梯度更新更稳定。以图6为例,要计算 $syscall_{104}$ 的注意力分数,首先要将 $\boldsymbol{Q}_{104}$ 与其他3个系统调用的 $\boldsymbol{K}$ 进行点积运算,求得 $syscall_{104}$ 与它们的相似度;再用 softmax 函数进行归一化处理求出概率;最后将概率与每个系统调用的 $\boldsymbol{V}$ 相乘,加权和的结果就是 $syscall_{104}$ 的注意力分数。这个结果包含了一个系统调用与其他系统调用的局部关系信息,能够很好地反映序列的语义特征。

$$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=\mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{1}$$

多头注意力是 h 个缩放点积注意力的线性组合,它通过对一个序列做 h 次缩放点积自注意力计算,让模型在序列的不同位置联合学习来自不同表征子空间的信息,能够更好地提取序列的语义信息。图7展示了多头注意力机制的工作

原理，多头注意力的计算方法如式(2)所示：

$$MultiHead(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=Concat(\boldsymbol{H}_1,\cdots,\boldsymbol{H}_h)\boldsymbol{W}^O$$
$$\boldsymbol{head}_i=Attention(\boldsymbol{QW}_i^Q,\boldsymbol{KW}_i^K,\boldsymbol{KW}_i^V) \tag{2}$$

其中，$\boldsymbol{W}^O,\boldsymbol{W}^Q,\boldsymbol{W}^K,\boldsymbol{W}^V$为线性变换用到的矩阵，它们一开始被随机初始化，在模型的训练过程中逐步更新；$\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}$是词向量的嵌入表示通过线性变换得到，$\boldsymbol{H}_1,\cdots,\boldsymbol{H}_h$为$h$次不同的缩放点积自注意力计算的结果。

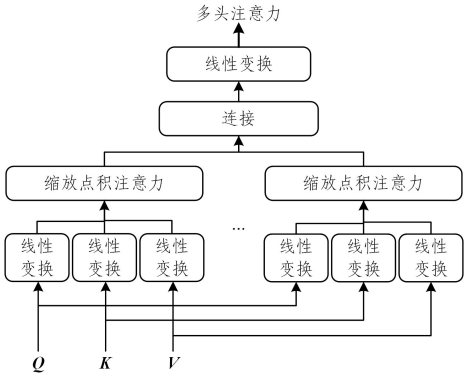


图7　多头注意力

Fig. 7　Multi-head attention

对于一个统一长度的系统调用序列，文中使用上述多头注意力机制对其嵌入表示矩阵进行计算，以得到序列的语义特征矩阵，输出矩阵的元素值为各个系统调用的注意力分数。

3.5　异常检测与分类

图8为文中框架异常检测模块的原理示意。在对统一长度的系统调用序列进行语义信息抽象和语义特征提取后，便得到了一个序列的语义抽象信息st_i和语义特征信息sf_i，然后将这3种信息输入三通道TextCNN进行模型的训练和检测，每种信息输入一个CNN通道中进行处理。

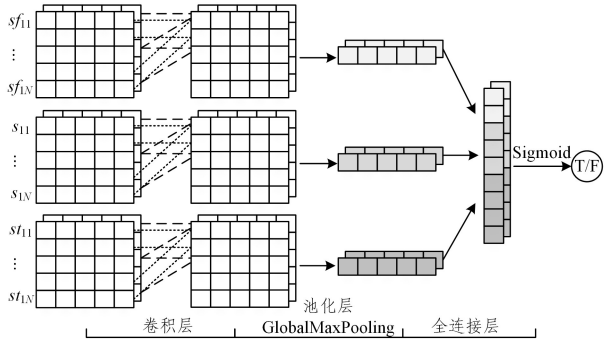


图8　异常检测模块

Fig. 8　Anomaly detection module

文中使用的TextCNN模型一共包括3部分，分别为卷积层、池化层、全连接层。模型中输入矩阵的维度大小为$batch_size\times maxlen\times k$，$batch_size$为模型输入的批量大小，$maxlen$为统一序列的长度，$k$词向量嵌入表示的维度。卷积层卷积核大小为$h\times k\times 3$，其中$h$，$k$，3分别为卷积核的高度、宽度、深度。卷积核的高度指每次卷积操作中卷积核窗口所覆盖的一个矩阵中的向量个数，卷积核的宽度为嵌入表示词向量的维度，卷积核深度是输入通道的数量。令$\boldsymbol{x}_i\in\mathbb{R}^k$为某个序列中第$i$个系统调用对应的$k$维词向量，则一个系统调用序列可以表示为：

$$\boldsymbol{x}_{1:maxlen}=\boldsymbol{x}_1\oplus\boldsymbol{x}_2\oplus\cdots\oplus\boldsymbol{x}_{maxlen} \tag{3}$$

其中，$\oplus$为拼接运算符，$maxlen$为序列长度。一般地，用$\boldsymbol{x}_{i:i+j}$表示序列中连续的j个系统调用，那么给定一个卷积核$\boldsymbol{W}\in\mathbb{R}^{h\times k}$，则一个CNN通道中每次卷积操作的特征输出$c_i$的计算方法如式(4)所示：

$$c_i=f(\boldsymbol{W}\cdot\boldsymbol{x}_{i:i+h}+b) \tag{4}$$

其中，b为卷积层的偏置值，$f(\cdot)$为激活函数ReLU，h为卷积核高度。模型中的卷积运算如式(5)所示：

$$c_t=f(\boldsymbol{W}_0\cdot\boldsymbol{ST}_{j,t:t+h}+\boldsymbol{W}_1\cdot\boldsymbol{S}_{j,t:t+h}+\boldsymbol{W}_2\cdot\boldsymbol{SF}_{j,t:t+h}+\boldsymbol{b}) \tag{5}$$

其中，t表示第t个时间步，$j(0\leqslant j\leqslant L-1)$为第$j$个系统调用序列。当所有卷积操作完成后，就会得到一个特征矩阵$\boldsymbol{C}$：

$$\boldsymbol{C}=[\boldsymbol{c}_1,\boldsymbol{c}_2,\cdots,\boldsymbol{c}_{maxlen-h+1}] \tag{6}$$

池化层将GlobalMaxPooling应用到特征矩阵$\boldsymbol{C}$上，以提取特征输出$\boldsymbol{C}$中对分类结果有着最重要影响的特征值。全连接层将池化层的输出拼接成高维特征向量，此向量包含了前面处理得到的序列原始信息、语义抽象信息及语义特征信息，最后模型使用Sigmoid函数得出分类结果，判定输入的原始系统调用序列是否异常。

4　实验

4.1　实验环境及数据集

本实验使用的操作系统为Ubuntu20.04.5 LTS，软件包版本为Tensorflow v2.5.0，主机内存大小为32GB，CPU型号为英特尔至强E5-2650 v4@2.20GHz，GPU型号为英伟达GeForce RTX 2080 Ti。模型的超参数设置如表3所列。

表3　模型超参数设置

Table 3　Model hyperparameter settings

参数	值
序列统一长度	1750
词向量维度	64
卷积核窗口高度	3,4,5
多头注意力多头数目	8
多头注意力键的维度	8
批量大小	4
训练轮次	15

实验在ADFA-LD[21]数据集上进行，这是当前系统调用异常检测领域通用的当代数据集，该数据集包含Normal Training Data，Normal Validation Data，Attack Data 3种不同类型的数据，每种数据都包含若干原始系统调用序列，其中Attack Data包含Adduser，Hydra_FTP，Hydra_SSH，Java_Meterpreter，Meterpreter，Web_Shell 6种类型的攻击数据。数据集具体描述如表4所列。

表4　ADFA-LD数据集结构

Table 4　ADFA-LD dataset structure

数据类型	系统调用序列数量
Normal Training Data	833
Normal Validation Data	4373
Attack Data	746

4.2　实验方法

为了验证所提框架的检测效果，本文设置了3组实验进行探究分析：(1)不同序列长度下的检测实验；(2)模型性能检测对比实验；(3)混淆攻击检测对比实验。不同长度的序列

包含的系统调用序列的语义完整性不同，而本文是从系统调用的语义信息出发进行检测，因此有必要探究系统调用序列的长度对检测结果的影响。模型性能检测实验是为了探究本文方法与其他异常检测模型检测效果的优劣；混淆攻击检测实验是为了探究本文方法能否有效检测前文所提到的混淆攻击。文中使用召回率(Recall)、精确率(Precision)、F1 分数(F1-Score)、误报率(FPR)、漏报率(FNR)5 个指标作为实验的评判标准。

(1)不同序列长度下的检测实验

Xie 等[22]的研究发现，ADFA-LD 数据集中绝大多数系统调用序列的长度分布在区间[100,500]内。本组实验将进行序列补齐时序列的统一长度 *maxlen* 作为自变量进行对比实验，探究不同序列长度对本文方法检测效果的影响。实验时将数据集按照 7∶3 的比例划分为训练集和测试集。

(2)模型性能检测对比实验

为评估本文方法的有效性，我们使用其他当前流行的深度学习技术搭建了各种异常检测模型，在同样的条件下进行实验对比。

(3)混淆攻击检测对比实验

本组实验选用数据集中 Hydra FTP 和 Hydra SSH 两个子集作为混淆攻击实验的训练和检测数据。这两个子数据集的特征与前文混淆攻击的定义一致，它们是属于同一攻击效果的不同类型的攻击样本，且每个系统调用序列中的攻击子序列也不相同。这里将 Hydra SSH 作为 Hydra FTP 的等效攻击样本数据，Hydra FTP 用作训练数据，Hydra SSH 用作测试数据，同时随机抽取与两个子集等量的正常序列作为训练和测试数据，使用本文方法和文献[3,23-24]中基于组合规则、*n*-gram 子序列等的方法在上述数据集中进行检测，同时使用其他深度学习技术搭建了各种异常检测模型用于对比。

4.3 实验结果与分析

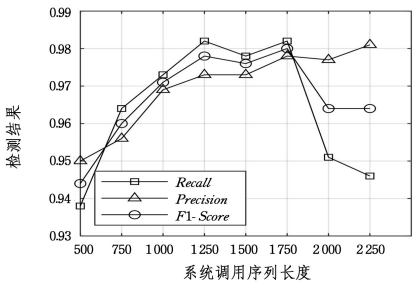
(1)不同序列长度下的检测实验

表 5 列出了不同序列长度下本文方法的检测效果，图 9 为表中各指标的图示。由图可知，随着系统调用序列长度的增加，模型的检测精度不断上升，误报率和漏报率不断下降，因为较长的系统调用序列避免了序列的截断操作，可以较为完整地保留序列的语义信息，从而提升检测结果的精确度。值得注意的是，当序列的长度超过 1 750 时，模型的检测性能出现下降趋势，说明并非序列越长模型检测效果越好。超长的序列中往往含有过多冗余序列，包含过多无用信息，因此这里将序列的统一长度 *maxlen* 设为 1 750，且后文中的实验都使用相同的设置。

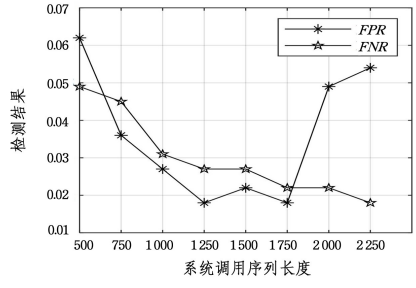
表 5 不同序列长度下模型检测结果

Table 5 Model detection results with different sequence lengths

maxlen	*Recall*	*Precision*	*F1-Score*	*FPR*	*FNR*
500	0.938	0.950	0.944	0.049	0.062
750	0.964	0.956	0.960	0.045	0.036
1 000	0.973	0.969	0.971	0.031	0.027
1 250	0.982	0.973	0.978	0.027	0.018
1 500	0.978	0.973	0.976	0.027	0.022
1 750	0.982	0.978	0.980	0.022	0.018
2 000	0.951	0.977	0.964	0.022	0.049
2 250	0.946	0.981	0.964	0.018	0.054



(a)Recall,Precision,F1-Score



(b)FPR,FNR

图 9 不同序列长度下的检测结果

Fig. 9 Detection results with different sequence lengths

(2)模型性能检测对比实验

表 6 列出了本文方法与其他深度学习异常检测模型的对比结果。从检测结果可知，无论是检测精度还是误报率，本文方法均取得了最好的结果。观察表中召回率和精确率两个指标可知，本文方法的检测结果比对比模型中最好的检测结果分别高出 0.071，0.020，且对比模型的召回率一般低于精确率，说明本文方法鉴别异常样本的效果更好。图 10 展示了各方法的受试者工作特征曲线(Receiver Operating characteristic Curve，ROC)对比结果，本文方法的 ROC 曲线下面积(Area Under Curve，AUC)值明显优于对比方法，达到了 0.982，而对比模型中最高的 AUC 值为 0.914。

表 6 模型检测性能对比实验

Table 6 Comparative experiment of model detection performance

检测方法	*Recall*	*Precision*	*F1-Score*	*FPR*	*FNR*
本文方法	**0.982**	**0.978**	0.980	0.022	0.018
Transformer[20]	0.907	**0.958**	0.932	0.036	0.093
TextAttBiRNN	0.853	0.914	0.882	0.072	0.147
FastText	0.844	0.917	0.879	0.068	0.156
RCNN[25]	0.884	0.850	0.867	0.140	0.116
TextBiRNN	**0.911**	0.813	0.859	0.188	0.089
TextRNN	0.790	0.855	0.821	0.120	0.210

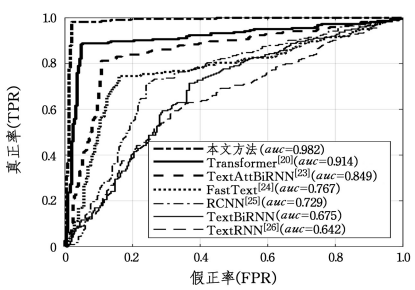


图 10 模型检测性能 ROC 曲线对比

Fig. 10 Comparison of ROC curve of model detection performance

(3)混淆攻击检测对比实验

表7列出了各方法针对混淆攻击的检测效果。从表中对比结果可知，本文方法的F1分数相较其他方法提高了0.058～0.406，各项检测指标相比其他方法均有提升，说明本文方法对系统调用序列语义信息的抽象和特征提取操作在检测混淆攻击上具有显著的效果，在增加模型鲁棒性的同时，提高了模型的检测精度。图11为针对混淆攻击的ROC曲线对比图，可以看到，本文方法的AUC值大于对比方法，说明本文方法的综合性能更好。

表7　混淆攻击检测对比实验

Table 7　Comparative experiment on confusion attack detection

检测方法	*Recall*	*Precision*	*F1-Score*	*FPR*	*FNR*
本文方法	0.966	0.971	**0.969**	0.028	0.034
Transformer[20]	0.875	0.951	**0.911**	0.045	0.125
FastText	0.844	0.922	0.881	0.064	0.156
Khater 等[3]	0.852	0.862	0.857	0.136	0.148
TextAttBiRNN	0.835	0.786	0.810	0.227	0.165
TextBiRNN	0.733	0.896	0.806	0.085	0.267
Borisaniya 等[23]	0.881	0.742	0.805	0.307	0.119
TextRNN	0.818	0.742	0.778	0.284	0.182
RCNN[25]	0.619	0.932	0.744	0.045	0.381
Serpen 等[24]	0.493	0.656	**0.563**	0.269	0.507

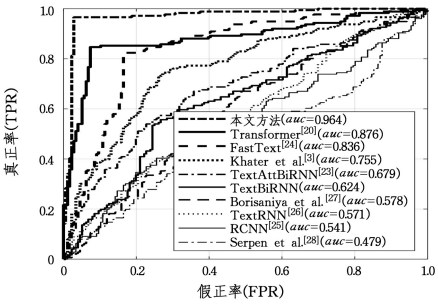


图11　混淆攻击检测效果ROC曲线对比

Fig. 11　ROC curve comparison of obfuscation attack detection effect

结束语　本文提出了一种基于系统调用多方面语义信息融合的主机异常检测框架，该框架在原始系统调用序列的基础上应用语义信息抽象和语义特征提取，获得表征程序行为模式的深层融合信息，结合TextCNN实现异常检测框架的搭建；该框架不仅对一般的主机异常具有较好的检测效果，还能有效检测混淆攻击，具有检测精确度高、误报率低等优点。一般地，基于系统调用n-gram子序列、系统调用序列频率特征等的检测方法，其检测效果对序列模式的变化比较敏感，当对系统调用序列进行篡改混淆后，检测效果就会明显下降。相比之下，本文提出的基于系统调用多方面语义信息融合的主机异常检测框架通过对序列进行语义信息抽象和语义特征提取，获得了序列中更稳定的语义信息，从而提高了模型异常检测的精确性和鲁棒性。此外，考虑到同一个系统调用可能被具有不同行为意图的进程调用，所以系统调用的参数信息也是判别系统调用序列行为模式的关键特征。下一步计划将系统调用的参数信息融合到异常检测模型中，以进一步完善框架的检测机制。

参 考 文 献

[1] ROSENBERG I, GUDES E. Bypassing system calls-based intrusion detection systems[J]. Concurrency and Computation: Practice and Experience, 2017, 29(16): e4023.

[2] TONG F, YAN Z. A hybrid approach of mobile malware detection in Android[J]. Journal of Parallel and Distributed Computing, 2017, 103: 22-31.

[3] KHATER B S, WAHAB A W B A, IDRIS M Y I B, et al. A lightweight perceptron-based intrusion detection system for fog computing[J]. Applied Sciences, 2019, 9(1): 178-199.

[4] AGHAEI E, SERPEN G. Ensemble classifier for misuse detection using N-gram feature vectors through operating system call traces[J]. International Journal of Hybrid Intelligent Systems, 2017, 14(3): 141-154.

[5] XIE M, HU J, SLAY J. Evaluating host-based anomaly detection systems: Application of the one-class SVM algorithm to ADFA-LD[C]// 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). IEEE, 2014: 978-982.

[6] DAS P K, JOSHI A, FININ T. App behavioral analysis using system calls[C]// 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2017: 487-492.

[7] KIM Y. Convolutional Neural Networks for Sentence Classification[C]// Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2014: 1746-1751.

[8] FORREST S, HOFMEYR S A, SOMAYAJI A, et al. A sense of self for unix processes[C]// Proceedings 1996 IEEE symposium on security and privacy. IEEE, 1996: 120-128.

[9] CREECH G, HU J. A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns [J]. IEEE Transactions on Computers, 2013, 63(4): 807-819.

[10] MURTAZA S S, KHREICH W, HAMOU-LHADJ A, et al. A trace abstraction approach for host-based anomaly detection [C]// 2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). IEEE, 2015: 1-8.

[11] KHREICH W, KHOSRAVIFAR B, HAMOU-LHADJ A, et al. An anomaly detection system based on variable N-gram features and one-class SVM[J]. Information and Software Technology, 2017, 91: 186-197.

[12] JIANG G, CHEN H, UNGUREANU C, et al. Multiresolution abnormal trace detection using varied-Length n-grams and automata[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2006, 37(1): 86-97.

[13] CHEN Z L, YI P, CHEN X, et al. Real-time Anomaly Detection Framework via System Calls Based on Integrated Learning[J]. Computer Engineering, 2023, 49(6): 162-169, 179.

[14] LIAO Y, VEMURI V R. Use of k-nearest neighbor classifier for intrusion detection[J]. Computers & Security, 2002, 21(5): 439-448.

[15] XIE M, HU J, YU X, et al. Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to ADFA-LD[C]// Network and System Security: 8th International Conference (NSS 2014). Xi'an, China, Springer International Publishing, 2014: 542-549.

[16] LIU Z, JAPKOWICZ N, WANG R, et al. A statistical pattern based feature extraction method on system call traces for anomaly detection[J]. Information and Software Technology, 2020, 126: 106348.

[17] WAGNER D, SOTO P. Mimicry attacks on host-based intrusion detection systems[C]// Proceedings of the 9th ACM Conference on Computer and Communications Security. 2002: 255-264.

[18] MING J, XIN Z, LAN P, et al. Replacement attacks: automatically impeding behavior-based malware specifications[C]// 13th International Conference on Applied Cryptography and Network Security, ACNS 2015. Springer Verlag, 2015: 497-517.

[19] LE Q, MIKOLOV T. Distributed representations of sentences and documents[C]// International Conference on Machine Learning. PMLR, 2014: 1188-1196.

[20] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]// Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., 2017: 6000-6010.

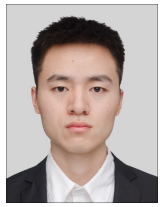
[21] CREECH G, HU J. Generation of a new IDS test dataset: Time to retire the KDD collection[C]// 2013 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2013: 4487-4492.

[22] XIE M, HU J. Evaluating host-based anomaly detection systems: A preliminary analysis of ADFA-LD[C]// 2013 6th International Congress on Image and Signal Processing (CISP). IEEE, 2013: 1711-1716.

[23] BORISANIYA B, PATEL D. Towards virtual machine introspection based security framework for cloud[J]. Sādhanā, 2019, 44: 1-15.

[24] SERPEN G, AGHAEI E. Host-based misuse intrusion detection using PCA feature extraction and kNN classification algorithms [J]. Intelligent Data Analysis, 2018, 22(5): 1101-1114.

[25] LAI S, XU L, LIU K, et al. Recurrent convolutional neural networks for text classification[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2015: 2267-2273.



FAN Yi, born in 1998, postgraduate. His main research interest is host anomaly detection.

HU Tao, born in 1993, Ph. D, assistant researcher. His main research interests include new network architecture and active cyber defense.

(责任编辑:何杨)