

# 基于攻击上下文分析的多阶段攻击趋势预测

朱光明<sup>1</sup>, 卢梓杰<sup>1</sup>, 冯家伟<sup>2</sup>, 张向东<sup>2</sup>, 张锋军<sup>3</sup>, 牛作元<sup>3</sup>, 张亮<sup>1</sup>

(1. 西安 **基于攻击上下文分析的多阶段攻击趋势预测** 710071;

2. 西安电子科技大学 通信工程学院, 陕西 西安 710071;

3. 中国电子科技集团公司第三十研究所, 四川 成都 610041)

**摘要:**高级可持续威胁(Advanced Persistent Threat, APT)等多阶段攻击具有复杂多样性和隐蔽持续性的特点,给网络安全带来了极大的威胁。研究攻击方的攻击策略并对其后续攻击步骤进行预测,是防御方的一个重要研究课题。针对多阶段攻击趋势预测难的问题,该文提出了基于攻击上下文分析的多阶段攻击趋势预测算法,从系统日志中梳理攻击上下文并对后续的攻击趋势进行预测。该算法先通过因果图构建、异常日志序列提取、抽象文本表示等步骤实现对已有攻击上下文的分析,然后基于已经检测到的攻击序列,利用Transformer模型对后续攻击趋势进行预测。在开源的ATLAS数据集和HDFS数据集上对算法进行了验证。在ATLAS数据集的超过7 000个序列中,该算法的单步预测准确率可达90%以上,五步预测准确率也能达到74%。实验表明基于攻击上下文分析的攻击趋势预测是一种可行的方法,为网络攻击预测研究提供了一种新思路。

**关键词:**网络安全;因果图;攻击预测;自然语言处理;Transformer

中图分类号:TP391

文献标识码:A

文章编号:1673-629X(2023)07-0104-07

doi:10.3969/j.issn.1673-629X.2023.07.016

## Multi-stage Attack Prediction Based on Attack Context Analysis

ZHU Guang-ming<sup>1</sup>, LU Zi-jie<sup>1</sup>, FENG Jia-wei<sup>2</sup>, ZHANG Xiang-dong<sup>2</sup>, ZHANG Feng-jun<sup>3</sup>,  
NIU Zuo-yuan<sup>3</sup>, ZHANG Liang<sup>1</sup>

(1. School of Computer Science and Technology, Xidian University, Xi'an 710071, China;

2. School of Communication Engineering, Xidian University, Xi'an 710071, China;

3. The 30th Research Institute of China Electronics Technology Group Corporation, Chengdu 610041, China)

**Abstract:** Multi-stage attacks, such as Advanced Persistent Threat (APT), have the characteristics of complex diversity and concealment persistence, and pose a great threat to the network security. Therefore, to study the attack strategies of attackers and predict the subsequent attack steps is still an important research topic for defenders. In order to overcome the difficulty to predict the trend of multi-stage attacks, we propose a multi-stage attack trend prediction algorithm based on the attack context analysis, which analyzes the attack context from the system logs and predicts the subsequent attack steps. The proposed algorithm firstly fulfills the attack context analysis through the construction of causal graphs, the extraction of abnormal log sequences and the abstract text representation. Then, the subsequent attack steps are predicted using the Transformer-based model based on the detected attack sequences. The proposed algorithm has been evaluated on the released ATLAS dataset and HDFS dataset, and it has achieved the accuracy of more than 90% on one-step prediction and the accuracy of 74% on five-step prediction, among the more than 7 000 sequences of ATLAS. The experiments demonstrate that it is practicable and reasonable to predict the trend of multi-stage attacks based on the attack context analysis. This also supplies a new idea for researches on network attack prediction.

**Key words:** network security; causal graph; attack prediction; natural language processing; Transformer

## 0 引言

与以影响或瘫痪目标系统为目标的网络攻击不同,高级可持续威胁(Advanced Persistent Threat, APT)

攻击具有非常强的隐蔽性和持续性<sup>[1]</sup>,一般是由专业组织发起,长期综合运用多种攻击手段对特定目标进行渗透,主要目的是获取目标的关键信息<sup>[2]</sup>。根据网

络杀伤链模型(Cyber Kill Chain,CKC),APT多阶段网络攻击可分为7个步骤:侦察、武器化、投递、漏洞利用、安装、指挥和控制、目标行动<sup>[3]</sup>。对标CKC模型,MITRE ATT&CK<sup>[4]</sup>模型根据真实的观察数据来描述和分类对抗行为,总结出了常用的14种战术和200多种技术。传统的攻击防御系统,如入侵检测系统(Intrusion Detection System,IDS)、入侵防御系统(Intrusion Prevention System,IPS)、高级安全设备(Advanced Security Appliances,ASA)等<sup>[5-6]</sup>,无法有效发现隐蔽的APT攻击,难以对后续的攻击做出预测及防御。网络流和系统日志虽然可以记录APT攻击过程,但是APT攻击的隐蔽性和持续性决定了无法依靠单步攻击检测来发现整个攻击过程。随着网络规模的日益增长,网络流和系统日志数量也与日俱增,迫切需要把机器学习方法运用到网络安全领域,实现对网络攻击过程的自动检测。

基于机器学习的网络攻击检测算法研究需要相应的数据集来支撑。在网络流数据集方面,相继有UNSW-NB15<sup>[7]</sup>、NSL-KDD<sup>[8]</sup>、CICIDS2017<sup>[9]</sup>、CICIDS2018<sup>[10]</sup>、DAPT2020<sup>[11]</sup>等数据集被提出并开源。目前基于网络流量的异常检测主要聚焦于单步攻击检测,无法捕获多阶段攻击的长期行为。刘景美等人<sup>[12]</sup>提出了基于自适应分箱特征选择的快速网络入侵检测算法,主要解决传统入侵检测系统查全率较低以及基于深度学习的入侵检测的训练用时过长的问題,在NSL-KDD数据集上进行了验证。Myneni等人<sup>[11]</sup>在DAPT2020数据集上使用编码-解码模型训练重建良性网络流数据,重建误差过大的数据被判定为异常数据。这种方法单独对每个网络流包进行检测,忽略了多阶段攻击的上下文关系,导致异常的准确率和查全率低。Allard<sup>[13]</sup>在DAPT2020论文模型方法的基础上引入了有效载荷,但还是无法有效解决这个弊端,在横向移动阶段检测性能依旧较差。针对当前研究对APT攻击多阶段流量特征的多样性感知不足的问题,谢丽霞等人<sup>[14]</sup>提出一种基于样本特征强化的APT攻击多阶段检测方法,引入多阶段感知注意力机制,提高了APT攻击多阶段检测的精度。

与网络流相比,系统日志可以更加详尽地记录APT攻击在主机上的执行过程。在基于系统日志分析的攻击检测方面,通过系统日志数据构建抽象表达能力强的溯源图并分析因果关系,可以有效表达威胁事件的起因、攻击路径和攻击影响,为威胁发现和取证分析提供更高的检测效率和性能<sup>[15]</sup>。自然语言处理技术在系统日志文本分析方面也发挥重要作用<sup>[16-17]</sup>。ATLAS<sup>[18]</sup>是一种用于重建攻击故事的框架,利用自然语言处理技术和基于序列的模型学习技术从审计日志

中恢复攻击步骤。LogAnomaly<sup>[19]</sup>是一个数据驱动的深度学习框架,用于非结构化日志流的异常检测,日志解析中用到了Word2Vec<sup>[20]</sup>方法来构成日志序列,利用LSTM(Long Short-Term Memory)模型来预测日志是否异常。LogBERT<sup>[21]</sup>是一种基于BERT(Bidirectional Encoder Representation from Transformers)<sup>[22]</sup>的日志异常检测方法,通过两个自监督训练任务学习正常日志序列的模式,能够检测出底层模式偏离正常日志序列的异常。DeepLog<sup>[23]</sup>是一种基于LSTM的深度神经网络模型,把日志信息建模成自然语言序列来处理,自动提炼正常的日志序列进行训练;当日志序列偏离训练的模型时,可以检测出异常。Li等人<sup>[24]</sup>提出了DeepAG,能够同时检测APT序列和利用日志语义向量和索引来定位序列中的攻击阶段,并根据上述日志索引构建攻击图。

上述方法都利用了日志的语义向量序列,并基于深度学习框架进行攻击检测;可以定位日志序列中的异常点,但并没有提出如何预测攻击者下一步的攻击行为。尽管这些方法能做到实时监测日志,但是它们不能从现有攻击序列对后续一步或多步的可能的攻击进行预测。APT攻击的检测与预测不是独立的两个方向,它们是高度联系、相辅相成的。该文结合上述日志分析中因果溯源图和自然语言处理的方法,在ATLAS日志数据集上构建因果图,根据恶意标签的节点提取完整的攻击序列,提出基于攻击上下文分析的多阶段攻击预测算法。该文的贡献可以概括为以下几点:

(1)提出了对攻击者行为进行多阶段预测的概念,挖掘攻击序列之间的非线性依赖关系;

(2)通过构建因果图来提取具有上下文关系的异常日志序列,避免了正常日志对预测结果的影响;

(3)利用Transformer模型,将日志分析并进行攻击预测的任务转化成文本分类的处理方式,并在开源数据集上进行了验证。

## 1 算法流程

### 1.1 算法基本框架

基于攻击上下文分析的多阶段攻击预测算法的基本框架如图1所示,包含五个主要阶段:

阶段1:利用日志数据构建因果图;

阶段2:以已知恶意节点为线索,提取异常日志序列;

阶段3:将提取的异常日志序列进行抽象化文本表示,进一步解析成日志索引;

阶段4:用Transformer模型训练日志向量数据;

阶段5:用训练好的模型进行多阶段攻击预测。

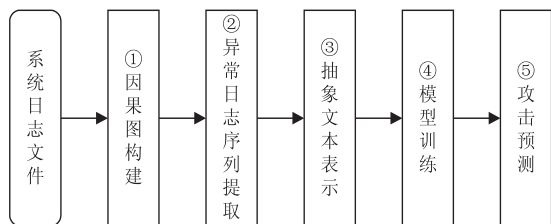


图 1 多阶段攻击预测算法流程框架

### 1.2 因果图构建

因果图<sup>[25-26]</sup> (Causal Graph) 常用于溯源追踪, 表征数据之间的因果关系或者依赖关系。该文从系统日志中提取数据来构建因果图, 表征了主体 (例如进程) 和对象 (例如文件或连接) 之间的因果关系。在因果图中, 以主体和对象作为图中的节点, 以主体对对象的动作 (例如读、写) 来生成有向边。

因果图中的节点代表从系统日志中提取的具有唯一 ID 的系统主体或对象, 比如进程名、文件名、IP 地址、域名和会话等。边从主体指向对象, 连接两个节点, 表示主体对对象执行的动作。因果中每两个节点和边的组合对应了一个日志项。将日志中的数据按照上述原则生成节点和边, 可以构建一个复杂的因果图。图 2 展示了一个因果图示例, 其中灰色节点表示一个已知的攻击节点。

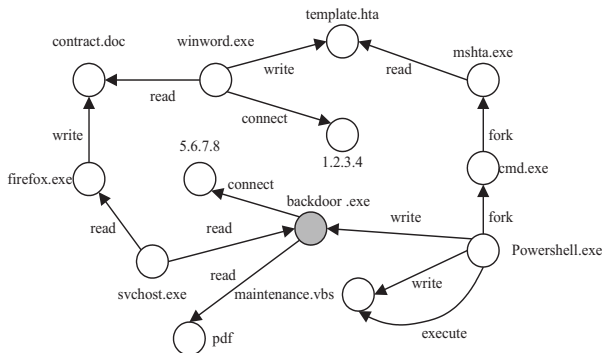


图 2 因果图示例

### 1.3 异常日志序列提取

在 ATLAS 数据集中, 异常的节点都被打上了恶意标签。该文围绕这些恶意节点提取出异常日志序列, 具体分为三个步骤:

第一步: 提取邻域图。在因果图中, 通过边相连的两个节点称为邻居。提取一个或多个节点的邻域图, 只需把它们所有的邻居和相连的边提取出来。

第二步: 从邻域图中分离出事件。一个事件被组织成一个四元组 <源节点, 目的节点, 动作, 时间戳>。比如 cmd.exe 在时间 t 打开了文件 flag.txt, 事件表示为四元组 <cmd.exe, open, flag.txt, t>。把邻域图中的每一个四元组分离出来, 每个事件实际上对应了一条日志项。

第三步, 按时间戳对事件进行排序。以已知的单

个或多个恶意节点的组合作为中心, 提取邻域图, 再从邻域图中提取出事件, 最后把事件按照时间戳进行排序, 得到的序列称之为异常日志序列。如图 3 所示, 已知节点 C 为恶意节点, 那么所有与它有关的日志项都被判定为异常, 并且按照时间排序后, 形成的日志序列也是异常的日志序列。通过这种方式, 可以把所有与恶意节点有关的日志项提取出来, 形成按时间排序的、具有明确上下文关系的异常序列, 排除了在预测任务中正常行为日志项对预测的影响。文中的实验数据就是在这些异常日志序列的基础上构建的。

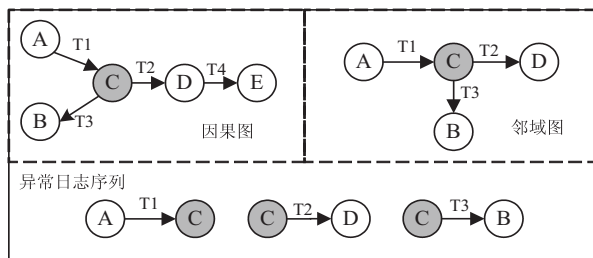


图 3 提取异常日志序列流程

### 1.4 抽象文本表示

为了能将提取出来的异常日志序列用于模型训练, 需要将词汇抽象化, 把日志序列转化成能用于语义解释的通用模板。在 ATLAS 词汇抽象的基础上做了进一步的词汇组合抽象, 构成句子抽象。下面将详细介绍具体流程。

表 1 抽象词汇集合

类型	抽象词汇
进程	system_process, lib_process, programs_process, user_process
文件	system_file, lib_file, programs_file, user_file, combined_files
网络连接	system_file, lib_file, programs_file, user_file, combined_files
动作	read, write, delete, execute, invoke, fork, request, refer, bind, receive, send, connect, ip_connect, session_connect, resolve

如表 1 所示, 在 ATLAS 中根据日志中词的细粒度语义将词分为四种不同的类型: 进程、文件、网络连接和动作。四种类型中总共包含 30 个抽象词汇, 将原本日志项中带有具体 ID 的实体映射到对应类型的相关抽象词汇, 就可以将日志项抽象成统一的三元组模板。例如, “c:/windows/system32/taskhost.exe\_1416 read c:/windows/inf/tapisrv/0409” 转化为 <system\_process read system\_file>, 抽象之后的句子仍然保留完整的关键语义。这样, 日志序列就可以被抽象成多个三元组构成的句子。将具体的日志项抽象成三元组句子, 可以在不牺牲攻击调查的关键语义的情况下降低日志复杂性, 使得日志种类大大减少。这个过程保留了完整



序列的原始语义,有利于基于序列的模型学习,保证模型的有效性和精度。

对于基于异常日志的 APT 攻击预测任务来说,一个三元组代表一条日志项的抽象程度还不够,需要将日志项抽象到日志类型。经过分析抽象为三元组的异常日志后,三元组的组合方式的数量是有限且理想的,一般有  $\langle \text{user\_process}, \text{process}, \text{programs\_process} \rangle \text{read}$   $\langle \text{user\_file}, \text{combined\_files} \rangle$  这样相对固定的组合。该文整理了所有出现的三元组的组合方式,并赋予它们类型索引。

表2展示了部分三元组日志和它们对应的日志类型索引,完整三元组数据一共有54种组合方式,于是设置了对应数目的日志类型索引。可以将异常日志序列映射为日志类型索引的序列,经过词嵌入 Embedding<sup>[22]</sup>后,每个日志索引都转化成词向量,按照索引构成序列的顺序,词向量构成了传入 Transformer 模型的词向量矩阵,如图4所示。

表2 三元组日志对应的日志类型索引

三元组日志	日志类型索引
IP_Address resolve domain_name	1
process connect connection	2
user_process connect connection	3
...	...
process delete user_file	54

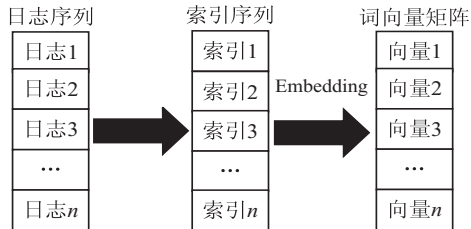


图4 日志序列处理流程

### 1.5 模型训练

该文提出基于 Transformer 模型<sup>[4]</sup>的多阶段攻击预测算法 LogTransformer。为了训练用于预测多阶段攻击的 Transformer 模型,以日志索引序列对应的词向量序列作为输入,以该索引序列的下一个或者多个索引作为输出。即由若干长度的异常日志序列,推测出下一步或多步可能产生的异常日志,以此来预测攻击者接下来的攻击意图。这种做法比较类似于文本分类等多分类模型的算法,接下来将详细介绍 LogTransformer 模型如何进行预测工作。

Transformer 模型是自然语言处理方面极为先进的模型,它放弃了传统的卷积神经网络和循环神经网络,整个网络结构完全是由注意力(Attention)机制组成,主要包括编码器和解码器两个部件。不同于循环

神经网络只能从左向右或者从右向左依次计算,Transformer 可以并行处理向量,从而充分利用 GPU 资源,减少了处理语义向量等高维数据的运行时间。考虑到模型最终的输出只需要概率的向量表示,LogTransformer 模型只采用了 Transformer 的编码器(Encoder)对日志序列进行编码,然后预测。

如图5所示,LogTransformer 主体部分由4个 Transformer 编码器组成,每个编码器包含一个5头注意力层和一个前馈全连接层。在日志向量输入到注意力层之前,需要与位置编码(Position Encoding)做相加,这是因为 Transformer 没有循环神经网络中的顺序结构,因此需要加入词的位置信息来显式地表明词的上下文关系。该文利用公式(1)和公式(2)来进行位置嵌入:

$$\text{PE}(\text{pos}, 2i) = \sin(\text{pos} / 10\,000^{2i/d_{\text{model}}}) \quad (1)$$

$$\text{PE}(\text{pos}, 2i + 1) = \cos(\text{pos} / 10\,000^{2i/d_{\text{model}}}) \quad (2)$$

其中,PE 是一个“序列长度×词向量维度”形状的二维矩阵,pos 是词在序列中的位置,  $d_{\text{model}}$  表示词嵌入的维度,  $i$  表示词向量的位置。

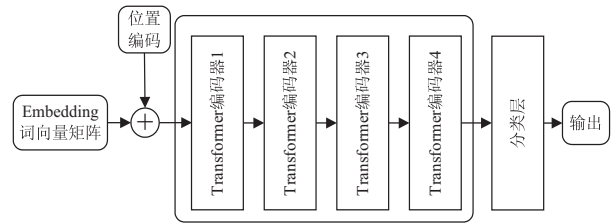


图5 LogTransformer 模型框架

### 1.6 攻击预测

经过多个 LogTransformer 的多个编码块的数据处理后,再连接一个分类层将高维数据映射为低维数据,最后将二维张量变形成三维,输出的数据是  $(B, S, C)$  格式的矩阵。 $B$  为每次批处理的序列数量; $S$  表示最终预测的步长,即如输入异常序列之后,预测接下来的  $S$  动作; $C$  的大小是54,对应54个日志类型索引。图6说明了数据变换的过程。

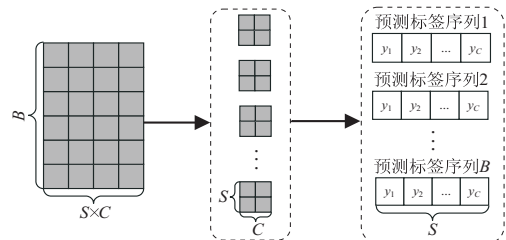


图6 输出数据到标签预测的转换

## 2 实验设计

### 2.1 数据集构建

#### 2.1.1 ATLAS 数据集

ATLAS 数据集包含三种日志,分别是 DNS 日志、

浏览器日志、审计日志。DNS 日志记录了域名解析活动,浏览器日志记录了 Web 请求,审计日志记录了进程、IP 及文件等系统活动。ATLAS 日志数据集分为两个部分,一个是记录单主机受 APT 攻击的日志,表示为 ATLAS-s;另一个是记录多主机环境下受 APT 攻击的日志,表示为 ATLAS-m。由于多主机情况下会出现横向移动,为了避免这种情况给实验带来的影响,该文将实验数据分为单主机和多主机两个部分。同时,还设置了另一个公开数据集 HDFS<sup>[27-28]</sup>作为对比。与 ATLAS 数据集不同的是,HDFS 日志序列中既包含了异常日志也包含了正常日志,而且没有明确的上下文关系。设置这个实验的目的是体现通过构建因果图提取异常日志序列的优势。

### 2.1.2 实验数据集构建

经过 1.4 节对异常日志序列进行文本表示以及索引化之后,得到了许多序列。在这些长序列上,设置长度为 10 的滑动窗口,移动步长为 1。依次在所有原始不定长序列的基础上,构建出等长度方便用于模型训练的序列数据。例如,假设长序列数据集中有一段数据为:{39, 37, 1, 11, 10, 6, 4, 34, 34, 29, 35, 34, 28, 28, 34}。滑动窗口大小为 10,输出标签长度为 1,则提取出来的训练数据为:{39, 37, 1, 11, 10, 6, 4, 34, 34, 29 → 35},{37, 1, 11, 10, 6, 4, 34, 34, 29, 35 → 34}。同样,如果设置输出预测步长为 2,则训练数据为:{39, 37, 1, 11, 10, 6, 4, 34, 34, 29 → 35, 34},{37, 1, 11, 10, 6, 4, 34, 34, 29, 35 → 34, 28}。

表 3 是 ATLAS 和 HDFS 数据集统计数据对比。ATLAS-s graph 和 ATLAS-m graph 是经过因果图提取攻击上下文后的 ATLAS 数据集。ATLAS-s seq 和 ATLAS-m seq 是仅对日志进行时间排序处理的 ATLAS 数据集,其中有大量的正常数据,也有更多的日志类型。为了排除日志种类过多对预测结果产生影响,把异常日志序列中没有的日志类型都归于一种正常日志类型。HDFS seq 是按时间排序的日志 HDFS 数据集。

表 3 ATLAS 和 HDFS 数据集对比

数据集	序列数量		日志索引 类型数量
	训练集	测试集	
ATLAS-s graph	1 995	686	46
ATLAS-m graph	3 741	1 410	54
ATLAS-s seq	19 580	6 771	47
ATLAS-m seq	59 114	19 212	55
HDFS seq	121 550	40 585	28

## 2.2 实验设置

### 2.2.1 实验参数

在实验中,输入序列的长度设置为 10,Embedding 词向量和 Transformer 编码器输入的维度  $d_{\text{model}} = 200$ ,Transformer 前馈网络层中神经元个数是 1 024。

### 2.2.2 对比算法

为了体现算法 LogTransformer 的有效性,引入了另外两个算法 DeepLog<sup>[23]</sup>和 DeepAG<sup>[24]</sup>进行对比。DeepLog 算法模型的主体部分采用传统的单向 LSTM 模型;DeepAG 算法模型的主体部分采用了双向 LSTM 模型。三种算法都基于章节 1.2、1.3、1.4 所描述流程处理后的数据,只是在预测模型上有差异。另外,也设置使用和不使用章节 1.2、1.3、1.4 所描述流程情况下的对比实验,以此验证通过因果图构建攻击序列进行多阶段攻击预测的有效性。

### 2.2.3 评估指标

在单步预测中,计算了每个数据集以及不同算法的预测结果的精确率(Precision)、召回率(Recall)和 F1 分数(F1-score)。在多步长的预测中,精确率、召回率和 F1 分数等评价指标变得难以计算。因为多步预测中,模型输出的预测值序列需要与实际的标签序列完全一致,才会判定它预测正确。比如设置预测标签长度为 5,那么预测值的 5 个索引值要与实际标签的 5 个索引值都分别对应相等。在这种情况下,上述三个评估指标极难计算,而且不再适用。因此,在多步预测中,只采用准确率(Accuracy)作为唯一的评估指标。

## 2.3 实验结果

把实验结果分为单步预测和多步预测两种情况,记录了不同算法在 ATLAS 数据集上的表现,并对实验结果进行了分析。

### 2.3.1 单步预测算法性能对比

表 4 分别显示了基于不同数据集的不同算法的精确率、召回率和 F1 分数。分析其中的数据,可以发现无论是在 ATLAS 的单主机日志数据集(ATLAS-s graph)上,还是在多主机日志数据集(ATLAS-m graph)上,三个算法模型中,LogTransformer 的性能最高,其中在 ATLAS 的单主机数据集和多主机数据集上的精确率分别为 92.42% 和 90.92%,召回率分别为 93.67% 和 90.43%,F1 分数分别为 93.04% 和 90.67%。

表 4 单步预测性能对比 %

数据集	ATLAS-s graph			ATLAS-m graph		
	精确率	召回率	F1 分数	精确率	召回率	F1 分数
DeepLog <sup>[23]</sup>	89.80	90.53	90.16	88.95	88.16	88.55
DeepAG <sup>[24]</sup>	91.25	91.12	91.18	90.43	90.43	90.43
LogTransformer	92.42	93.67	93.04	90.92	90.43	90.67

都是三个算法模型中最高性能表现。

### 2.3.2 多步预测算法性能对比

分析表5和表6可以发现,在多步预测的情况下,DeepAG和LogTransformer在ATLAS单主机数据集上的准确率对比互有胜负;而在多主机数据集上,LogTransformer则具有优势,在准确率上比DeepAG平均领先0.7%。DeepLog在ATLAS的两个数据集上的表现都是最差的,步长从2到5都是准确率最低的,说明传统单向LSTM模型优势不足。

表5 在ATLAS-s graph数据集上多步预测准确率对比 %

预测步长	DeepLog <sup>[23]</sup>	DeepAG <sup>[24]</sup>	LogTransformer
2	81.76	85.50	85.05
3	74.85	78.53	80.06
4	68.82	75.75	76.22
5	66.83	74.43	73.62

表6 在ATLAS-m graph数据集上多步预测准确率对比 %

预测步长	DeepLog <sup>[23]</sup>	DeepAG <sup>[24]</sup>	LogTransformer
2	80.74	84.38	84.96
3	76.75	80.70	81.00
4	68.45	76.55	77.46
5	63.98	74.47	74.63

### 2.3.3 LogTransformer 多步预测分析

为了比较LogTransformer在经过因果图提取算法的日志数据和未经过处理的日志数据上性能的差别,设置了因果图提取后的ATLAS数据集(ATLAS-s graph和ATLAS-m graph)、原始ATLAS数据集(ATLAS-s seq和ATLAS-m seq)和HDFS数据集(HDFS seq)的对照实验。图7统计了LogTransformer在五中数据集上预测步长从1到5的准确率的变化趋势。

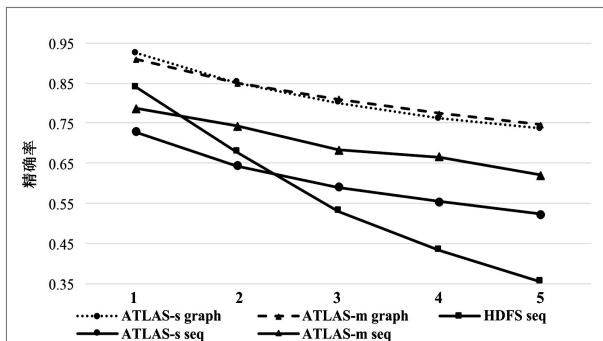


图7 LogTransformer在五中数据集上的性能对比

从整体上看,随着预测步长的增加,LogTransformer算法的准确率也在下降。在因果图提取后的ATLAS数据集(ATLAS-s graph和ATLAS-m

graph)上,即使预测步长达到了5,LogTransformer的准确率依然能保持在74%左右。在原始ATLAS单主机数据集(ATLAS-s seq)上,不同预测步长的准确率整体上都比因果图提取后的ATLAS-s graph数据集低20%左右。在原始ATLAS多主机数据集(ATLAS-m seq)上,不同的预测步长的准确率平均比因果图提取后的ATLAS-m graph数据集低12%。在HDFS seq数据集上,下降幅度比较大,到三步预测准确率已经下降到53%了,五步预测的准确率已经下降到了35%。这说明未经提取上下文关系的日志数据在用于预测的时候稳定性较差。

这是因为原始ATLAS数据集和HDFS数据集提取的日志序列并不是按照因果图来提取的,异常的日志中夹杂了很多正常操作的日志数据,所以序列里的日志项之间并没有明确的攻击上下文关系。对比分析HDFS数据集和ATLAS数据集的规模,ATLAS数据集的数据量比HDFS少很多,而ATLAS的日志类型数目是HDFS的将近2倍。LogTransformer在经因果图处理后的ATLAS数据集上获得了更佳的性能,这也证明了通过构建因果图来提取异常日志序列进行多阶段攻击趋势预测的有效性和先进性。

## 3 结束语

通过构建因果图来提取具有攻击上下文关系的异常日志序列,提出了一个基于攻击上下文分析的多阶段攻击预测算法。先通过因果图构建、异常日志序列提取、抽象文本表示等步骤实现对已有攻击上下文的分析;然后基于已经检测到的攻击序列,利用Transformer模型对后续攻击趋势进行预测。经过实验验证,所提算法在多阶段攻击预测上取得了良好的性能。但该算法依然存在一个缺陷,就是攻击趋势预测只能预测已经存在的54种日志类型,如果出现未知的攻击日志类型,算法并不能做出预测的更新。所以,接下来,会在这方面做出努力,完善未知攻击的实时更新处理,提高算法的拓展能力。

### 参考文献:

- [1] 田继鹏. APT攻击行为的阶段演变和多态检测方法探讨[J]. 中原工学院学报, 2019, 30(1): 89-94.
- [2] WILKENS F, ORTMANN F, HAAS S, et al. Multi-stage attack detection via kill chain state machines[C]//Proceedings of the 3rd workshop on cyber-security arms race. New York: ACM, 2021: 13-24.
- [3] MARTIN L. Cyber kill chain[EB/OL]. 2022. <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>.
- [4] MITRE. Mitre att&ck[EB/OL]. 2020. <https://attack.mitre>.

- org/.
- [5] ZOHREVAND Z, GLASSER U. Should I raise the red flag? A comprehensive survey of anomaly scoring methods toward mitigating false alarms[J]. arXiv:1904.06646, 2019.
- [6] ANWAR S, MOHAMAD Z J, ZOLKIPLI M F, et al. From intrusion detection to an intrusion response system: fundamentals, requirements, and future directions[J]. Algorithms, 2017, 10(2):39.
- [7] MOUSTAFA N, SLAY J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)[C]//2015 military communications and information systems conference. Canberra; IEEE, 2015: 1-6.
- [8] DHANABAL L, SHANTHARAJAH S P. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms[J]. International Journal of Advanced Research in Computer and Communication Engineering, 2015, 4(6):446-452.
- [9] SHARAFALDIN I, LASHKARI A H, GHORBANI A A. A Detailed analysis of the Cicans2017 data set[C]//International conference on information systems security and privacy. Berlin; Springer, 2018:172-188.
- [10] LEEVY J L, KHOSHGOFTAAR T M. A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data[J]. Journal of Big Data, 2020, 7(1):1-19.
- [11] MYNENI S, CHOWDHARY A, SABUR A, et al. DAPT 2020-constructing a benchmark dataset for advanced persistent threats[C]//International workshop on deployable machine learning for security defense. [s. l.]: Springer, 2020: 138-163.
- [12] 刘景美, 高源伯. 自适应分箱特征选择的快速网络入侵检测系统[J]. 西安电子科技大学学报, 2021, 48(1):176-182.
- [13] DIJK A. Detection of advanced persistent threats using artificial intelligence for deep packet inspection[C]//2021 IEEE international conference on big data (big data). [s. l.]: IEEE, 2021:2092-2097.
- [14] 谢丽霞, 李雪鸥, 杨宏宇, 等. 基于样本特征强化的 APT 攻击多阶段检测方法[J]. 通信学报, 2022, 43(12):66-76.
- [15] HAN X, PASQUIER T, SELTZER M. Provenance-based intrusion detection: opportunities and challenges[C]//10th USENIX workshop on the theory and practice of provenance. Berkeley; USENIX, 2018.
- [16] QIU J, ZHANG J, LUO W, et al. A survey of android malware detection with deep neural models[J]. ACM Computing Surveys, 2020, 53(6):1-36.
- [17] LIN G, WEN S, HAN Q L, et al. Software vulnerability detection using deep neural networks: a survey[J]. Proceedings of the IEEE, 2020, 108(10):1825-1848.
- [18] ALSAHEEL A, NAN Y, MA S, et al. {ATLAS}: a sequence-based learning approach for attack investigation[C]//30th USENIX security symposium (USENIX security 21). Berkeley; USENIX, 2021:3005-3022.
- [19] MENG W, LIU Y, ZHU Y, et al. LogAnomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs[C]//Proceedings of the twenty-eighth international joint conference on artificial intelligence. California; IJCAI, 2019:4739-4745.
- [20] RONG X. word2vec parameter learning explained[J]. arXiv: 1411.2738, 2014.
- [21] GUO H, YUAN S, WU X. Logbert: log anomaly detection via bert[C]//2021 international joint conference on neural networks. Shenzhen; IEEE, 2021:1-8.
- [22] DEVLIN J, CHANG M W, LEE K, et al. Bert: pre-training of deep bidirectional transformers for language understanding[J]. arXiv:1810.04805, 2018.
- [23] DU M, LI F, ZHENG G, et al. Deeplog: anomaly detection and diagnosis from system logs through deep learning[C]//Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. New York; ACM, 2017: 1285-1298.
- [24] LI T, JIANG Y, LIN C, et al. DeepAG: attack graph construction and threats prediction with bi-directional deep learning[J]. IEEE Transactions on Dependable and Secure Computing, 2023, 20(1):740-757.
- [25] LEE K H, ZHANG X, XU D. High accuracy attack provenance via binary-based execution partition[C]//Network and distributed systems security symposium. San Diego; IS, 2013.
- [26] MA S, ZHANG X, XU D. Protracer: towards practical provenance tracing by alternating between logging and tainting[C]//Network and distributed systems security symposium. San Diego; IS, 2016.
- [27] LOU J G, FU Q, YANG S, et al. Mining invariants from console logs for system problem detection[C]//2010 USENIX annual technical conference (USENIX ATC 10). Berkeley; USENIX, 2010.
- [28] XU W, HUANG L, FOX A, et al. Detecting large-scale system problems by mining console logs[C]//Proceedings of the ACM SIGOPS 22nd symposium on operating systems principles. New York; ACM, 2009:117-132.