doi:10.3772/j.issn.1002 - 0470.2026.08.001

基于混合图专家的入侵检测方法①

刘丁玮②*** 严 瑾③*** 李振宇***

(*中国科学院计算技术研究所 北京 100190)

(**中国科学院大学 北京 100049)

(***中国互联网络信息中心 北京 100070)

摘 要 网络攻击对企业和政府造成日益严重的威胁,入侵主机的行为尤为突出。溯源图因其能反映主机日志中的攻击上下文,被广泛应用于入侵检测。早期基于图粒度的方法仅能识别攻击的发生,而无法定位具体节点。近年来,越来越多的工作基于图学习实现了节点级的检测。然而,这些方法由于训练集的节点类别和特征分布不均衡,损害了模型的泛化能力,难以兼顾精确度和召回率。针对这一问题,本文提出了一种基于混合图专家的检测方法。其核心多门混合图专家模型(multi-gate mixture-of-graph-experts,MGE)融合了图神经网络(graph neural network, GNN)和多门混合专家(multi-gate mixture-of-experts, MMOE)。MGE 设计了 GNN 图专家,用以捕捉反映攻击上下文的长流信息。特别地,共享的图专家由门控网络根据数据和下游任务自动调整组合,考虑了特征空间和任务间的差异性。为有效训练 MGE,本文设计了 3 种监督信号:上游的输入重构用于提升特征表示的质量,下游的节点特征重构学习长流行为,和边的多标签分类学习短流行为。最后,利用下游的特征重构误差定位可疑进程并溯源报警。实验结果表明,MGE 同时提升了节点级的精确度和召回率,在多个数据集上取得了优于基线方法的结果。

关键词 溯源图; 入侵检测; 图神经网络; 多任务训练; 混合专家模型

当前,针对企业和政府的网络攻击日益增多,攻击者入侵其主机等设备,窃取机密信息,实施破坏活动,造成经济损失并威胁国家安全。尽管网络攻击具有极强的隐蔽性,但在被入侵主机的系统日志中仍会留下痕迹。例如,钓鱼邮件引发的恶意负载安装会被记录为系统日志中的数据接收、写文件、文件执行等事件。为了更好地挖掘系统日志中蕴含的信息来检测入侵,研究者们[1-5]开始使用溯源图来建模系统日志中的复杂关联关系。溯源图将系统实体视为节点,事件视为边,事件中的数据流(如写文件)和控制流(如进程克隆)的方向视为边的方向(源节点至目标节点)。因此,图结构能够清晰地表示一跳至多跳实体间复杂的上下文关联关系。

基于规则的溯源图检测系统[2,5-7]依赖规则集的完备性确保检测的精确度。因此,它们难以应对日益演进的攻击(如零日漏洞等未知技术)。为了避免对专家知识和规则的依赖,越来越多的工作[1,8-13]采用基于异常的假设,通过学习主机或实体的行为特征,将偏离历史的行为识别为异常。早期的学习主机行为的检测系统[1,8]通过编码时间片内所有事件获取主机当前时间片的行为特征,但只能检测攻击发生的时间片,无法区分时间片内出现的正常实体和攻击实体。为了实现节点级检测,学习实体行为的检测系统[9-13]利用神经网络模型编

① 国家重点研发计划(2022YFB3103000)资助项目。

② 女,1997 年生,博士生;研究方向:网络安全,入侵检测;E-mail: liudingwei19g@ict.ac.cn。

③ 通信作者,E-mail: yanjin@cnnic.cn。

(收稿日期:2025 - 07 - 18)

码实体(节点)的有限跳结构特征来判定异常,从而大幅缩减了报警中的节点规模。然而,由于训练数据中实体类别和特征分布不均衡,这些方法的模型可能对频繁数据过拟合,从而影响泛化能力。以进程为例,实际训练集中不同进程名(暗示特征及行为不同)的进程数量差异显著(例如从 1 个到 18 454 个不等)。这最终导致了现有的方法难以兼顾精确度和召回率。

多门混合专家(multi-gate mixture-of-experts, MMOE)[14]架构通过多任务协同学习缓解数据不均衡对训练的影响,并利用门控网络面向数据和下游任务差异化地分配专家组合,从而在不同数据分布下保持稳定的性能。因此,MMOE 架构有望帮助解决学习实体行为的方法的训练难题。然而,在溯源图的检测模型中应用 MMOE 架构面临 2 个挑战。首先,传统的专家网络为处理图片、序列等规则数据而设计,采用前馈神经网络,无法处理图结构信息以编码行为特征;其次,MMOE 架构的训练效果依赖于任务的相关性,这要求面向溯源图检测场景精心设计多任务建模,以充分利用无标签数据学习实体的行为模式。

本文提出了一种基于混合图专家的入侵检测方法,其中检测模型设计为多门混合图专家模型(multi-gate mixture-of-graph-experts, MGE)。MGE 设计了使用图卷积层聚合有限跳内特征的图专家。每个图专家生成的特征捕捉了同一终点的所有局部长流的信息。MGE 使用门控网络根据终点的特征分配专家权重,自动调整专家组合,生成捕捉长流信息的结构嵌入(挑战 1)。为了有效协同训练 MGE 模型(挑战 2),本文考虑溯源图的特点设计了 3 种训练任务。在 MGE 的上游,特征提取模块从节点的向量化输入中获取节点的特征。任务 1 是使用该特征重构节点的输入,以提升特征表示的质量。在 MGE 的下游,关注溯源图上长流和短流对不同粒度行为信息的贡献。任务 2 侧重长流,直接用混合图专家编码的结构嵌入重构终点特征,以学习实体的多跳行为模式。任务 3 侧重短流(边),组合目标节点的结构嵌入和源节点的节点特征预测边的多标签分类,以学习多跳视野下的一跳关联模式。因为任务 2 的重构误差能够评估进程行为与历史的偏移,本文方法在检测时利用该重构误差判定进程的异常程度。在此基础上,后续模块使用高异常度的进程及其关联邻居构造报警。总的来说,本文的主要贡献如下。

(1)融合图神经网络和多门混合专家,提出了图学习模型 MGE。MGE 通过双门控网络针对长流和短流任务,面向数据(终点)调整图专家组合以提取高质量的结构嵌入。

(2)设计了三任务协同训练方案,包括上游的节点任务和下游的长流、短流任务,有效提升了训练效果,并最终增强了检测任务的性能。

(3)在公开数据集上进行性能评估,并与已有最先进的代表性基线方法进行对比。实验结果表明,本文方法在所有数据集上取得了最优或接近最优的 $F1$ 分数。这表明本文方法能够同时取得更好的精确度和召回率,为安全人员提供更精确、全面的攻击信息。

1 背景和相关工作

1.1 威胁模型

本文重点在于检测对主机的潜在攻击。系统级审计日志,如由 Windows ETW[15]和 Camflow[16]等工具记录的数据,保留了这些攻击的痕迹。本文的威胁模型与已有工作[9-13]相似。首先,假设收集到的系统级日志完整且可信,它们包含与攻击相关的关键事件。例如在钓鱼邮件安装恶意负载时,系统中的接收数据、写入文件等事件。其次,与攻击相关的结构区别于良性结构,例如邻居中出现恶意文件。本文通常认为在良性数据中出现过的事件是良性的。此外,本文不考虑在系统日志中不可见的数据流。

1.2 基于溯源图的异常检测

1.2.1 溯源图

系统级审计日志记录了系统实体间的操作。溯源图将这些系统实体(如进程、文件和网络对象)建模为节点。将它们之间的操作(如读、写)建模为边。图 1 展示了一个溯源图的实例。其中边的标签表示事件类型,边的方向表示数据流或控制流的方

向。例如,进程 nignx 接收网络协议(Internet protocol,IP)地址为 81.49.200.166 的网络对象的数据,对应的数据流方向为从网络对象到进程。通过局部结构(连通子图),溯源图能够反映出攻击的上下文。例如,图 1 反映出攻击者利用 nginx 漏洞下载攻击负载并执行的过程。

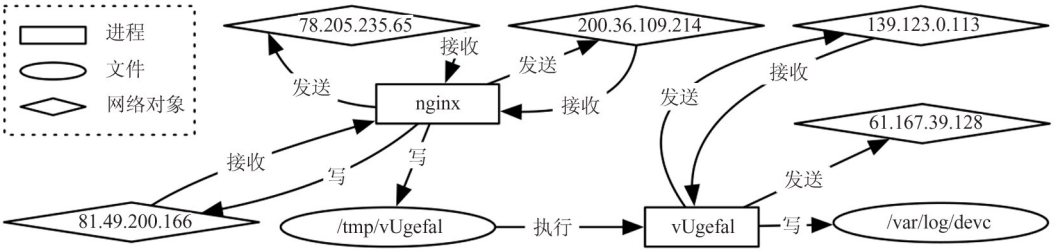

图 1　溯源图的实例

1.2.2　基于异常的溯源图上入侵检测

早期的基于规则的溯源图上入侵检测系统[1-5]依赖外部知识和专家经验构造规则集,这限制了它们应对日益复杂的攻击技术的性能。因此,越来越多的工作[1,8-10,12-13]采用基于异常的方法,学习主机或实体的正常行为,并将偏离正常的行为判定为异常。本文方法延续了基于异常的设计思路。

其中,学习主机行为的检测方法[1,8]以时间片为粒度编码行为特征并报告异常。以 UNICORN[8]为例,它按时间窗口划分系统日志,并使用草图化(graph sketching)技术编码每个时间窗口的主机行为特征,构造特征序列。UNICORN 通过聚类建模主机的可能状态及状态转移。检测时,序列特征中不符合历史状态或非法状态转移的时间窗口被报告为异常。这类方法无法区分检测到的时间片内不同实体的区别,而攻击相关的实体可能仅占 0.1%。学习实体行为的检测方法[9-10,12-13]编码实体有限跳内的上下文信息以提取行为特征。这类方法将检测的粒度细化到实体级别,大幅降低了报警规模。以 ThreaTrace[9]为例,它通过图神经网络(graph neural network,GNN)编码局部结构学习实体的在溯源图中的角色,并通过多模型框架提升检测性能。然而,由于训练集的节点类别和特征分布不均衡损害了训练效果,进而影响了现有的学习实体行为的方法的检测性能。它们呈现出精确度和召回率的权衡,在部分数据集上精确度高而召回率低,在其他数据集则相反。也就是说,现有方法的局限性在于难以兼顾实体级的精确度和召回率。

1.3　多任务学习和混合专家模型

多任务学习[17-20]是一种通过共享模型参数或特征表示来协同训练多个任务的方法。通过共享隐藏层,多任务学习能够降低过拟合的风险并学习到更具泛化能力的特征。考虑到共享底层可能因任务差异而影响训练效果,多门混合专家[14]引入混合专家网络(mixture-of-expert,MOE)。MOE 被认为能够自动学习多个专家,分别处理输入空间的不同部分,并在前向传播时通过门控网络自动针对不同输入调整专家[21-22]。在此基础上,MMOE 进一步为不同任务使用不同的门控网络,从而实现共享底层模型针对不同任务的差异化激活。MMOE 的思想被应用于推荐系统[14]、计算机视觉[23-24]等领域,并取得了优秀的效果。

2　动机和挑战

为了解决实体级检测性能的局限性,本文深入探究了现有的学习实体行为的方法,从模型设计和训练 2 个角度进行分析。越来越多的工作[9,12-13]使用 GNN 捕捉实体在图中的局部结构信息。具体来说,它们的模型通过多个堆叠的图卷积层(文献[9,12]中的 SAGE[25],文献[13]中的 GAT[26])实现沿边的消息传播,从而为目标实体编码有限跳内的邻居信息。然而,不同的目标实体使用相同模型参数进行编码,忽略了输入特征空间不同部分的差异,这潜在限制了模型的表达能力。就训练而言,鉴于没有先验知识和攻击训练数据,基于异常的方法[1,8-9,11-13]设

计了不同的监督信号。其中,ThreaTrace[9] 和 FLASH[12] 采用半监督(semi-supervised)训练,根据模型提取的节点行为特征预测其分类(如文件、进程等)。然而,训练数据集中的节点类别不均衡可能导致模型对特定类型的节点过拟合,造成检测性能欠佳。MAGIC[13] 采用自监督(self-supervised)训练,设计了掩蔽特征重构和边重构 2 个训练任务。然而,不同任务共享相同的网络参数可能引发任务间的冲突,妨害模型产生嵌入的质量。

本文注意到 MMOE 架构在本场景中的潜力。首先,它能自动地面向数据调整模型中的专家组合;其次,MMOE 的多任务学习能够充分利用数据提高泛化能力。此外,通过多个门控网络,它能够捕捉任务间关系,协同训练模型。然而,在溯源图的检测模型中应用 MMOE 面临 2 个挑战:一是传统的专家网络无法处理图结构信息;二是如何面向溯源图检测的场景设计协同训练任务。因此,本文设计了一种基于混合图专家的检测方法,检测模型自动地根据目标实体的特征面向任务调整图专家组合。在图专家网络中,本文设计了由 2 个图卷积层(SAGE)构成的图专家,以捕捉反映长流行为信息的实体结构特征(挑战 1)。在训练时,本文设计了包含上游的节点任务和下游的长流和短流任务的三任务训练方案,高效利用数据促使专家学习跨任务的通用特征(挑战 2)。

3　方　法

本文提出的溯源图上入侵检测系统如图 2 所示,其包含 3 个主要模块:构造溯源图、图学习和威胁检测。构造溯源图模块将输入的系统日志转换为图,生成待检测的图结构及节点输入特征。图学习模块学习溯源图中的实体特征和结构特征,用于异常检测。训练时,图学习模块利用 3 个任务产生监督信号;检测时,图学习模块仅执行任务 2 以评估进程节点的异常值。威胁检测模块持续更新进程的异常值,并缓存其可疑邻居。最后,本文方法利用高异常值的进程及其缓存的邻居构造报警。

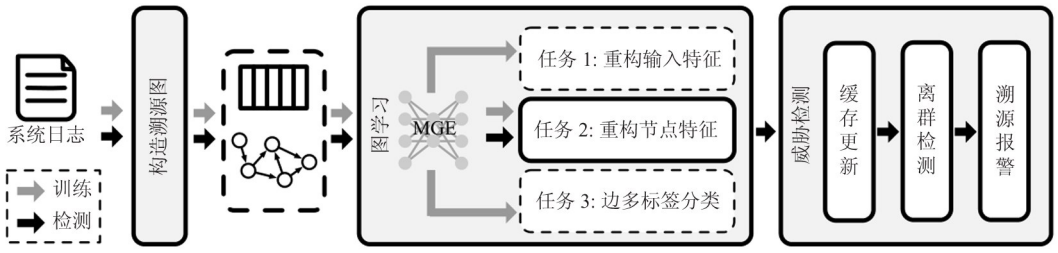

图 2　系统概览

3.1　构造溯源图

该模块解析系统日志以构造溯源图,用于图学习和威胁检测。该过程包含 3 个步骤。

步骤 1　日志解析。日志中的每条记录代表一个事件,由主体、客体和事件类型组成。其中主体(进程)和客体(进程、文件、网络对象)是系统实体,被视为节点。根据事件类型可以确定主客体间数据流和控制流的方向。例如,进程读文件事件的数据流方向是文件(客体)→进程(主体)。本文将左侧的节点称为源节点,右侧的节点称为目标节点。

步骤 2　节点合并和初始特征。该模块合并了具有相同属性的节点。进程属性是进程号和进程名,文件属性是文件名,网络对象属性是 IP 地址。节点的初始特征通过特征哈希[27](feature hashing)将属性字符串向量化得到。特征哈希是一种低开销的快速字符串向量化方法,也被 Kairos[10] 采用。

步骤 3　边合并和边标签。图 3 展示了一个本文方法处理溯源图中原始边的实例。原始日志数据中存在大量仅时间戳不同的重复事件,造成大量的重复边(图 3(a))。因此,本研究去除了具有相同源节点、目标节点和事件类型的重复边(图 3(b))。考虑到同一组源节点和目标节点的不同事件类型的边在邻居结构和特征上并无差异,本文进一步合并源节点和目标节点都相同的边,并重新生成其标

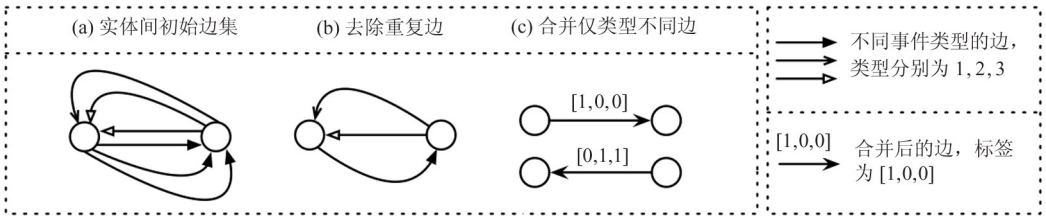

图 3　边合并和边标签

签(图 3(c))。合并后的边的标签采用多热编码(multi-hot embedding)表示事件类型。

3.2　图学习

该模块在训练时使用本文提出的 GME 学习实体的行为模式,在测试时评估进程的异常情况。下文将将详细介绍 GME 的模型结构和训练与测试方法。

3.2.1　模　型

如图 4 所示,本文将 GME 的模型结构大致划分为上游、中间层和下游 3 个部分。它们分别负责节点特征提取、结构嵌入提取和 2 个下游任务。

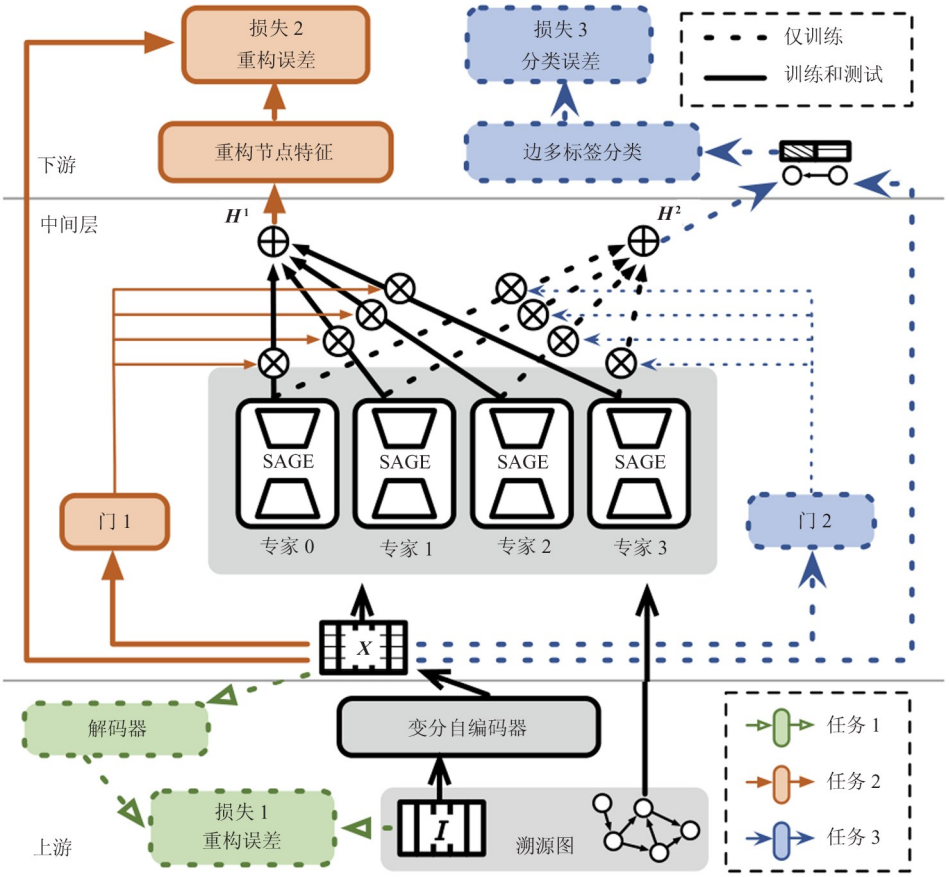

图 4　MGE 的模型结构

(1)上游

GME 的上游采用编码-解码结构的变分自编码器[28](variational autoencoder, VAE)。VAE 通过概率模型描述数据空间,能够有效地从无标注数据中学习特征表示。这里,GME 将向量化的初始特征 $\boldsymbol{I}$ 输入自编码器以提取节点特征 $\boldsymbol{X}$,然后通过一个多层感知机(multilayer perceptron, MLP)构成的解码器重构输入特征 $\boldsymbol{I}$。该重构任务是 GME 的上游任务,用于增强提取的节点的特征表示的质量。重构特征 $\boldsymbol{I}'$ 的计算公式为

$$\boldsymbol{I}' = \mathrm{MLP}(\boldsymbol{X}) \tag{1}$$

(2)中间层

GME 的中间层是一个多任务共享的图专家网络。其中,每个专家由一个包含 2 个 SAGE 图卷积层

的网络构成,用于获取节点的局部结构嵌入以捕捉多跳行为信息。每层图卷积通过聚合目标节点 d 的邻居前一层的输出特征得到 d 在该层的输出特征。节点 d 在第 l 层的输出特征 $\boldsymbol{X}_d^l$ 的计算公式为

$$\boldsymbol{X}_d^l = \sigma(\boldsymbol{W}_1^l \boldsymbol{X}_d^{l-1} + \boldsymbol{W}_2^l \sum_{s \in N(d)} \boldsymbol{X}_s^{l-1}) \tag{2}$$

式中:$\boldsymbol{W}_1^l$ 和 $\boldsymbol{W}_2^l$ 为第 l 层映射特征空间的可训练权重;$N(d)$ 为 d 的一跳邻居集合;s 代表该集合中的一个节点;$\sigma(\cdot)$ 为非线性激活函数,本文采用 LeakyReLu;本文记专家 k 对目标节点 d 的输出 $\boldsymbol{E}_d^k = \boldsymbol{X}_d^2$。

记所有 $|\boldsymbol{E}|$ 个专家的输出为 $\boldsymbol{E}_d = [\boldsymbol{E}_d^0, \boldsymbol{E}_d^1, \cdots, \boldsymbol{E}_d^{|\boldsymbol{E}|-1}]$。针对特定下游任务 i,记中间层混合专家对目标节点 d 的输出(结构嵌入)为 $\boldsymbol{H}_d^i$。那么,节点 d 的混合结构嵌入的计算公式为

$$\boldsymbol{H}_d^i = \boldsymbol{G}_d^i \times \boldsymbol{E}_d \tag{3}$$

式中:$\boldsymbol{G}_d^i$ 是由门控网络根据节点特征 $\boldsymbol{X}_d$ 计算的图专家权重分配矩阵。具体来说,门控网络采用噪声路由机制。$\boldsymbol{X}_d$ 经过线性变换分别得到专家得分向量和噪声向量,并由柔性最大值函数(softmax)得到图专家的分配矩阵 $\boldsymbol{G}_d^i$,具体的计算公式为

$$\boldsymbol{G}_d^i = \mathrm{softmax}(\boldsymbol{W}_g^i \boldsymbol{X}_d + \mathrm{noise}(\boldsymbol{W}_n^i \boldsymbol{X}_d)) \tag{4}$$

式中:$\boldsymbol{W}_g^i \in \mathbb{R}^{|\boldsymbol{E}| \times |\boldsymbol{X}_d|}$ 和 $\boldsymbol{W}_n^i \in \mathbb{R}^{|\boldsymbol{E}| \times |\boldsymbol{X}_d|}$ 分别为得到专家得分向量和噪声向量的线性变换的可训练权重;$|\boldsymbol{X}_d|$ 为节点特征的维度;noise(·)为噪声生成函数。

考虑到同一专家在处理不同下游任务时的能力可能存在差异,GME 的中间层为不同任务使用不同的门控网络分配专家权重,以在训练时自动学习不同任务的专家分配方案。

(3)下游

GME 的下游包含 2 个任务:节点的特征重构和边的多标签分类。对于目标节点 d,中间层输出的结构嵌入 $\boldsymbol{H}_d^1$ 和 $\boldsymbol{H}_d^2$ 通过图专家网络编码了其多跳结构的行为信息。考虑到溯源图上的长流能反映攻击的上下文,GME 使用图专家为 d 编码的结构嵌入 $\boldsymbol{H}_d^1$ 通过 MLP 重构其节点特征 $\boldsymbol{X}_d$,以学习目标节点的多跳行为模式。重构特征 $\boldsymbol{O}_d^1$ 的计算公式为

$$\boldsymbol{O}_d^1 = \mathrm{MLP}(\boldsymbol{H}_d^1) \tag{5}$$

考虑到短流(边)是攻击的最小粒度,本文构造边特征通过 MLP 预测其多热标签以学习短流关联模式。边 $s \rightarrow d$ 的边特征 $\boldsymbol{H}_{(s,d)}^2$ 由目标节点 d 的结构嵌入 $\boldsymbol{H}_d^2$ 和源节点 s 的节点特征 $\boldsymbol{X}_s$ 组合得到。多热标签的预测结果 $\boldsymbol{O}_d^2$ 的计算公式为

$$\boldsymbol{O}_{(s,d)}^2 = \mathrm{MLP}(\boldsymbol{H}_{(s,d)}^2) \tag{6}$$

$$\boldsymbol{H}_{(s,d)}^2 = \mathrm{CONCAT}(\boldsymbol{H}_d^2, \boldsymbol{X}_s) \tag{7}$$

式中:CONCAT(·)表示拼接。

由于 $\boldsymbol{X}_s$ 提供一跳短流信息,$\boldsymbol{H}_{(s,d)}^2$ 能够在具备多跳视野的同时专注特定边。算法 1 中的 MGE 函数展示了模型中具体的数据流向以及模型中游产生的特征与下游任务的衔接机制。

算法 1:MGE 模型的训练过程

输入:节点输入特征 $\boldsymbol{I}$,溯源图邻接表 Adj,边表多热标签 $\boldsymbol{Y}$,总训练轮次 $total_epoch$,损失平衡参数 α, β, γ

输出:模型 $\boldsymbol{\theta}$

```
function MMOE(θ, X, Adj):
    for k = 0 to |E| − 1 do                                  # 多个共享图专家
        E^k = Expert^k_[SAGE](θ, X, Adj)
    E = [E^0, ⋯, E^(|E|−1)]
    G^1, G^2 = Gate(θ, X), Gate(θ, X)                        # 双任务独立门控
    H^1, H^2 = G^1 × E, G^2 × E
    return H^1, H^2
function MGE(θ, I, Adj):
    X = VAE(θ, I)
    H^1, H^2 = MMOE(θ, X, Adj)
    I′ = MLP(θ, X)                                           # 任务 1:重构输入特征
```

```
12   O^1 = MLP(θ, H^1)                                  # 任务 2:重构节点特征
13   for (s, d) in Adj do                               # 任务 3:预测边多热标签
14     H^2_(s,d) = CONCAT(H^2_d, X_s)
15     O^2_(s,d) = MLP(θ, H^2_(s,d))
16   return I', X, O^1, O^2
17                                                      #训练阶段
18  θ ← Random( )                                       # 初始化模型权重
19  for e = 0 to total _ epoch do
20    I', X, O^1, O^2 = MGE(θ, I, Adj)
21    Loss1 = MSELoss (I', I) + KLLoss
22    Loss2 = MSELoss (O^1, X)
23    Loss3 = BCELoss (O^2, Y)
24    Loss = αLoss1 + βLoss2 + γLoss3
25    θ ← backward(θ, Loss)                             # 反向传播
```

3.2.2 训 练

为了有效地训练 GME,本文设计了 3 种监督信号对应 3 种训练任务。针对上游的特征重构任务,和原始 VAE 相同,本文使用均方差(mean square error,MSE)损失[29]和 KL 散度(Kullback-Leibler divergence,KLD)损失[30]来衡量重构误差和学习隐变量分布。记重构特征为 $\boldsymbol{I}'$,输入的向量化特征为 $\boldsymbol{I}$,损失 1 的计算公式为

$$\text{Loss1} = \text{MSELoss}(\boldsymbol{I}', \boldsymbol{I}) + \text{KLLoss} \tag{8}$$

针对下游的节点特征重构任务,记输出特征为 $\boldsymbol{O}^1$。我们同样使用 MSE 损失来衡量重构误差:

$$\text{Loss2} = \text{MSELoss}(\boldsymbol{O}^1, \boldsymbol{X}) \tag{9}$$

针对下游的边多标签分类任务,记输出为 $\boldsymbol{O}^2$,边的多热标签为 $\boldsymbol{Y}$,我们使用二元交叉熵损失(binary cross entropy loss,BCELoss)来计算多标签分类任务的误差:

$$\text{Loss3} = \text{BCELoss}(\boldsymbol{O}^2, \boldsymbol{Y}) \tag{10}$$

鉴于联合多任务训练时不同任务的损失值差异巨大,本文使用 3 种误差的加权和来训练整个模型:

$$\text{Loss} = \alpha\text{Loss1} + \beta\text{Loss2} + \gamma\text{Loss3} \tag{11}$$

式中:α、β 和 γ 用于平衡不同任务的损失,以缓解损失差异造成的任务偏向。算法 1 的 18 ~ 25 行描述了 GME 的训练过程。

3.2.3 测 试

在测试时,本文方法通过 GME 中的任务 2 来评估进程节点(主体)的异常情况。一方面,任务 2 本质上建模了进程与行为模式的关系,因此重构误差 Loss2 能够直接反映进程的行为模式与历史的偏离。另一方面,忽略边而独立检测节点可能会造成大量孤立点(包括攻击节点和正常节点),从而丢失体现在边上的关键事件信息。鉴于客体节点(文件和网络对象)一定会出现在攻击进程的邻居中,本文仅评估进程节点,后续通过扩展邻居来追溯事件(边)和其他类型的节点。

之前的研究发现直接用重构误差评估可能会导致大量行为不稳定的正常进程被误报[11,31]。例如,不同浏览器进程访问的 IP 地址可能存在显著差异。因此,在评估进程节点的异常情况时,我们考虑其稀有程度。具体而言,我们利用历史数据中同名进程的数量为每个进程名计算了一个放缩系数 S。对于进程名 p,记其历史数据中不同进程号的数量为 C_p,那么:

$$S_p = \begin{cases} \text{TemperatureSoftmax}\left(\ln\left(\dfrac{C_{\max}+1}{C_p}\right)\right), & p \in P_{\text{seen}} \\ \text{MAX}(S_p) \times 1.1, & p \notin P_{\text{seen}} \end{cases} \tag{12}$$

式中:TemperatureSoftmax 是对放缩系数的平滑处理;$C_{\max}$ 是 C_p 的最大值;P_{seen} 表示历史出现的进程名集合。对于历史中未出现的进程名,本文将其放缩系数 S_p 设为最大值的 1.1 倍,以表示其稀有度高于历史数据中出现的任意进程。基于此,将进程的异常值定义为

$$AS(p) = Loss2_p \cdot S_p \tag{13}$$

式中:AS 表示异常分数。

3.3 异常检测

该模块根据异常值识别并追溯攻击,主要包含 3 个阶段:缓存更新、离群检测和溯源报警。每个时间窗口执行完图学习模块的检测后,进入缓存更新阶段。与之前的工作类似[4,10],本研究为历史数据中常见的客体(文件和 IP 地址)生成白名单,用于在缓存时过滤冗余的节点和边。同时,缓存记录了进程在各窗口中异常值的最大值。在离群检测阶段,本文使用格拉布斯试验[32](Grubbs' s test)识别进程异常值中的离群值,相应的进程被认定为候选节点。最后溯源报警模块从缓存中取出源节点或目标节点为候选节点的边,并根据数据流和控制流构造多个连通图。排除孤立点边后,这些连通图作为报警被送至安全人员处审核。

4 实 验

本实验用 Python 实现了本文方法的全部模块。其中,模型使用 PyG[33]实现。模型的默认参数设置如下:专家数目 $|\boldsymbol{E}| = 4$,输入特征维度 $|\boldsymbol{I}| = 256$,结构嵌入维度 $|\boldsymbol{H}^1| = |\boldsymbol{H}^2| = 16$。所有实验均在同一台配备 Intel(R) Xeon(R) Silver 4214 CPU(128 GB)和 NVIDIA GeForce RTX 3080Ti GPU(12 GB)的服务器上进行。本节详细介绍了实验设置(表 1),并全面评估本文方法的性能。

表 1 数据集详情

数据集	攻击进程数量	攻击文件数量	攻击 IP 数量
CADETS	18	8	11
THEIA	62	4	5
TRACE	43	3	10
OPTC-1	18	5	8
OPTC-2	30	10	3
OPTC-3	81	4	4

4.1 实验设置

4.1.1 数据集

本文实验使用广泛用于入侵检测性能评估的公开数据集 DARPA TC[34]和 OPTC[35]进行评测。这 2 个数据集都是由美国国防高级研究计划局(defense advanced research projects agency,DARPA)在透明计算(transparent computing,TC)项目中采集。具体来说,在攻防演练中,蓝队多次攻击红队主机,并记录了红队主机的系统级执行日志。同时,项目根据攻击实施的时间、技术等形成基准报告。特别地,OPTC 数据集中的后 3 天对应 3 种不同类型的攻击。第 1 天 OPTC-1 为 powershell 渗透,第 2 天 OPTC-2 为数据泄漏,第 3 天 OPTC-3 为恶意软件升级。DARPA TC 数据集,本实验使用了 3 个子集 CADETS、THEIA 和 TRACE。OPTC 数据集,本实验使用了 201、501、51 三台被入侵机器的日志,分别对应 OPTC-1、OPTC-2 和 OPTC-3。本实验删除了数据集中无属性的实体,仅保留有属性的进程、文件和 IP 地址 3 类节点。本文根据基准报告标注了与攻击直接相关的节点,并和开源标注[36]对齐。具体的统计信息如表 1 所示。

4.1.2 基 线

本文对比的基线包括目前最先进的节点级入侵检测系统 ThreaTrace[9]、FLASH[12]和 MAGIC[13]。为了保证评价指标的一致性,本实验使用的测试数据片段与基线一致,并使用基线方法开源的代码和模型重新测试并处理其结果。其中,ThreaTrace 和 MAGIC 仅报告了 DARPA TC 的结果,FLASH 报告了 DARPA TC 和 OPTC 的结果。

4.1.3 指 标

本文使用各类型节点的节点级指标评估方法的性能。节点级指标包括真阳数目(TP)、假阳数目(FP)、假阴数目(FN)、精确度(precision,Prec),召回率(recall,Rec)和 $F1$ 分数。其中 TP、FP、FN 直接由统计获得。精确度表示攻击占识别为异常的比例,召回率表示识别到的攻击占所有攻击的比例。$F1$ 分数结合精确度和召回率,全面揭示模型的预测性能,其计算公式如式(14)所示。

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{14}$$

4.2 对比实验

表 2 展示了本文方法和基线在测试数据集上不同类型节点的性能对比。对于进程节点,本文方法

表 2　节点级性能对比结果*

数据集	方法	进程						文件						IP					
		TP	FP	FN	Rec	Prec	*F*1	TP	FP	FN	Rec	Prec	*F*1	TP	FP	FN	Rec	Prec	*F*1
CADETS	ThreaTrace	10	54	8	0.56	0.16	0.24	7	83	1	0.88	0.080	0.140	7	5	4	0.64	0.58	0.61
	FLASH	13	17	5	0.72	0.43	0.54	7	55	1	0.88	0.110	0.200	7	3	4	0.64	0.70	0.67
	MAGIC	4	141	14	0.22	0.03	0.05	1	90	7	0.12	0.010	0.020	7	5	4	0.64	0.58	0.61
	本文方法	13	20	5	0.72	0.39	0.51	8	39	0	1.00	0.170	0.290	10	7	1	0.91	0.59	0.71
THEIA	ThreaTrace	5	8	57	0.08	0.38	0.13	1	812	3	0.25	0.001	0.002	2	4	3	0.40	0.33	0.36
	FLASH	22	1	40	0.35	0.96	0.52	4	952	0	1.00	0.004	0.008	5	14	0	1.00	0.26	0.42
	MAGIC	60	146	2	0.97	0.29	0.45	4	69	0	1.00	0.050	0.100	5	4	0	1.00	0.56	0.71
	本文方法	52	49	10	0.84	0.51	0.64	4	66	0	1.00	0.060	0.110	4	0	1	0.80	1.00	0.89
TRACE	ThreaTrace	14	1975	29	0.33	0.01	0.01	3	2 076	0	1.00	0.001	0.003	10	7	0	1.00	0.59	0.74
	FLASH	10	923	33	0.23	0.01	0.02	2	239	1	0.67	0.010	0.020	10	56	0	1.00	0.15	0.26
	本文方法	27	35	16	0.63	0.44	0.51	3	60	0	1.00	0.050	0.090	10	2	0	1.00	0.83	0.91
OPTC-1	FLASH	9	51	9	0.50	0.15	0.23	2	9	1	0.67	0.180	0.290	0	0	8	0.00	0.00	0.00
	本文方法	7	9	11	0.39	0.44	0.41	4	58	1	0.80	0.060	0.120	5	19	3	0.62	0.21	0.31
OPTC-2	FLASH	14	26	16	0.47	0.35	0.40	2	59	8	0.20	0.030	0.060	3	0	0	1.00	1.00	1.00
	本文方法	13	20	17	0.43	0.39	0.41	6	141	4	0.60	0.040	0.080	3	35	0	1.00	0.08	0.15
OPTC-3	FLASH	2	13	79	0.02	0.13	0.04	0	42	4	0.00	0.000	0.000	2	3	2	0.50	0.40	0.44
	本文方法	55	19	26	0.68	0.74	0.71	4	88	0	1.00	0.040	0.080	2	9	2	0.50	0.18	0.27

* 由于历史未见的噪声(如大量临时文件)的存在,部分数据集文件节点的假阳较其他节点更多,导致本文方法和基线方法文件节点的 *F*1 分数较低。然而,本文方法显著降低了假阳数量,控制其在 150 以内(基线为百甚至千级)。此外,与基线以孤立节点形式报警不同,本文方法以连通图形式报警。大量假阳文件出现在不含攻击的图中,实际包含攻击的报警图中假阳文件的数量可能很少,这降低了安全人员核验报警的难度。例如 THEIA 的攻击报警图中仅有一个假阳文件。另外,由于没能复现 MAGIC 在 TRACE 数据集的结果,本文没有将其列在表中。

在大多数据集上取得了最优的 *F*1 分数,仅在 CADETS 上略低于 FLASH。这表明了本文方法在不同数据集上均能同时取得优秀的精确度和召回率。这得益于本文提出的混合图专家模型和多任务建模,有效地捕捉了实体的行为特征。相比之下,基线方法在不同数据集则呈现出显著的性能差异。例如 FLASH 在 CADETS 上取得了最优的进程 *F*1 分数,但在 TRACE 上的进程 *F*1 分数仅为 0.02,远低于本文方法。对于文件和 IP 节点,本文方法在 DARPA TC 的 3 个子集上均取得了最优的 *F*1 分数。训练数据集存在节点类别和特征分布的不均衡,这可能导致基线的模型对特定类别或特征的节点过拟合。因此,直接用模型判定文件和 IP 节点时产生大量假阳,导致精确度较低。以 ThreaTrace 为例,它报告了 773 个/run/shm/org. chromium/目录下的文件。这些文件是临时文件,且在训练集中未曾出现,使得 ThreaTrace 模型难以判断。相比之下,本文方法仅考虑与检测到的进程直接关联的罕见文件和 IP 节点,从而大量避免了对这类难以判断且不在攻击进程上下文中的节点的误报。另一方面,得益于本文方法产生了较少的假阳进程,检测到的进程直接关联的节点规模有限。在 OPTC 数据集上,尽管 FLASH 在部分文件和 IP 节点的指标更优秀,但遗漏了部分重要的攻击信息。以第一天为例,FLASH 取得了优秀文件节点 *F*1 分数,但遗漏了全部的 IP 节点,无法描述攻击者扫描网络(142.20.56.0/24)这一攻击阶段。与之对比,本文方法报告了相关的进程和 IP 节点(142.20.56.202、142.20.61.134、42.20.56.204、42.20.57.147),对应该网络扫描行为。

4.3　消融实验

为了验证本文提出的三任务协同训练策略的有效性,尝试使用不同的任务子集训练模型。本实验分别评估了仅使用任务 2(T2),使用任务 1 和任务 2(T1 + T2),使用任务 2 和任务 3(T2 + T3)3 种任务子集的训练效果。表 3 展示了这 3 种任务子集训练的模型和使用全部任务(T1 + T2 + T3)训练的模型用于检测的性能对比。使用全部任务训练的模型在不同数据集的不同节点类型上均取得了最优或接近最优的 *F*1 分数。因为本文模型直接评估进程的异常值,本文格外关注进程节点的性能。使用 3 个任务训练的模型在所有测试集上均取得最优的 *F*1 分数。相比之下,使用任务子集训练的模型在不同数

据集上表现出明显的性能差异。以进程节点为例，使用任务 2 和任务 3 训练的模型在 THEIA 上取得次优的 $F1$ 分数，但在另外 2 个数据集上取得最差的 $F1$ 分数。以上实验结果表明，本文提出的多任务训练策略有效地提高了模型的训练效果。而且，为保证在不同数据集上的性能的稳定性，3 个任务缺一不可。此外，由于检测时仅执行任务 2，检测开销与使用单任务的模型无异。

表 3　不同任务子集训练的模型性能对比

数据集	任务子集	进程						文件						IP					
		TP	FP	FN	Rec	Prec	$F1$	TP	FP	FN	Rec	Prec	$F1$	TP	FP	FN	Rec	Prec	$F1$
CADETS	T2	6	21	12	0.33	0.22	0.27	2	20	6	0.25	0.09	0.13	6	2	5	0.55	0.75	0.63
	T2 + T1	15	38	3	0.83	0.28	0.42	8	69	0	1.00	0.10	0.19	10	7	1	0.91	0.59	0.71
	T2 + T3	2	46	16	0.11	0.04	0.06	4	66	4	0.5	0.06	0.1	0	0	11	0.00	0.00	0.00
	T2 + T1 + T3	13	20	5	0.72	0.39	0.51	8	39	0	1.00	0.17	0.29	10	7	1	0.91	0.59	0.71
THEIA	T2	19	146	43	0.31	0.12	0.17	3	39	1	0.75	0.07	0.13	3	0	2	0.60	1.00	0.75
	T2 + T1	20	54	42	0.32	0.27	0.29	3	57	1	0.75	0.05	0.09	3	0	2	0.60	1.00	0.75
	T2 + T3	54	111	8	0.87	0.33	0.48	4	115	0	1.00	0.03	0.07	4	1	1	0.80	0.80	0.80
	T2 + T1 + T3	52	49	10	0.84	0.51	0.64	4	66	0	1.00	0.06	0.11	4	0	1	0.80	1.00	0.89
TRACE	T2	27	48	16	0.63	0.36	0.46	3	102	0	1.00	0.03	0.06	10	17	0	1.00	0.37	0.54
	T2 + T1	21	21	22	0.49	0.50	0.49	3	44	0	1.00	0.06	0.12	10	0	0	1.00	1.00	1.00
	T2 + T3	10	31	33	0.23	0.24	0.24	2	44	1	0.67	0.04	0.08	10	17	0	1.00	0.37	0.54
	T2 + T1 + T3	27	35	16	0.63	0.44	0.51	3	60	0	1.00	0.05	0.09	10	2	0	1.00	0.83	0.91

为了验证多任务独立门控机制的有效性，本实验对比了使用共享门控的模型，即 2 个下游任务共享一个门控网络。表 4 展示了保持本文方法其他模块不变的检测结果。可以看到，使用独立门控的模型节点级在指标上全面优于共享门控，这得益于独立门控能够在训练时自动地学习任务间的差异。与之对比，共享门控无法根据任务调整专家分配。

另外，本实验对比了训练时是否平衡不同任务的损失对性能的影响。平衡损失的设置下，我们根据初始训练损失的数量级差异，默认将多任务损失

表 4　独立门控与共享门控的性能对比

数据集	门控	进程						文件						IP					
		TP	FP	FN	Rec	Prec	$F1$	TP	FP	FN	Rec	Prec	$F1$	TP	FP	FN	Rec	Prec	$F1$
CADETS	独立门控	13	20	5	0.72	0.39	0.51	8	39	0	1.00	0.17	0.29	10	7	1	0.91	0.59	0.71
	共享门控	11	32	7	0.61	0.26	0.36	8	44	0	1.00	0.15	0.27	6	7	5	0.55	0.46	0.50
THEIA	独立门控	52	49	10	0.84	0.51	0.64	4	66	0	1.00	0.06	0.11	4	0	1	0.80	1.00	0.89
	共享门控	47	45	15	0.76	0.51	0.61	4	66	0	1.00	0.06	0.11	2	0	3	0.40	1.00	0.57
TRACE	独立门控	27	35	16	0.63	0.44	0.51	3	60	0	1.00	0.05	0.09	10	2	0	1.00	0.83	0.91
	共享门控	4	9	39	0.09	0.31	0.14	1	37	2	0.33	0.03	0.05	10	0	0	1.00	1.00	1.00

权重 α、β 和 γ 设置为 1×10^{-6}、1×10^{-6} 和 1。不平衡损失的设置下，α、β 和 γ 都被设置为 1。表 5 列出了节点级指标的性能对比。可以看到，平衡任务损失的设置下训练的模型检测的节点级指标更优，充分利用了多任务训练学习更具泛化能力的特征。

为了研究多任务损失权重对收敛的影响，本文将 CADETS 的正常数据按 8∶2 划分为训练集和验证集，以观察训练过程中损失的变化情况。图 5 展示了平衡与不平衡设置下训练集损失、验证集损失以及验证集任务 2 损失随轮次的变化。可以看到，平衡任务损失的设置下，训练集和验证集损失收敛得更快。鉴于检测时仅关注任务 2，本文单独报告了任务 2 在验证集的损失变化。可以看到，平衡损失的设置下收敛更快。尽管不平衡的设置下损失的值更低，但我们检测时关注异常节点的重构误差是否离群而非数值的大小。正如表 5 所示，平衡设置下达到的检测指标更优，进程、文件和 IP 的 $F1$ 分数分别高 16%、8% 和 2%。

表 5　训练时平衡任务损失与否的性能对比

数据集	平衡与否	进程						文件						IP					
		TP	FP	FN	Rec	Prec	*F*1	TP	FP	FN	Rec	Prec	*F*1	TP	FP	FN	Rec	Prec	*F*1
CADETS	平衡	13	20	5	0.72	0.39	0.51	8	39	0	1.00	0.17	0.29	10	7	1	0.91	0.59	0.71
	不平衡	8	20	10	0.44	0.29	0.35	8	60	0	1.00	0.12	0.21	10	8	1	0.91	0.56	0.69
THEIA	平衡	52	49	10	0.84	0.51	0.64	4	66	0	1.00	0.06	0.11	4	0	1	0.80	1.00	0.89
	不平衡	48	46	14	0.77	0.51	0.62	4	67	0	1.00	0.06	0.11	4	0	1	0.80	1.00	0.89
TRACE	平衡	27	35	16	0.63	0.44	0.51	3	60	0	1.00	0.05	0.09	10	2	0	1.00	0.83	0.91
	不平衡	11	9	32	0.26	0.55	0.35	2	22	1	0.67	0.08	0.15	10	2	0	1.00	0.83	0.91

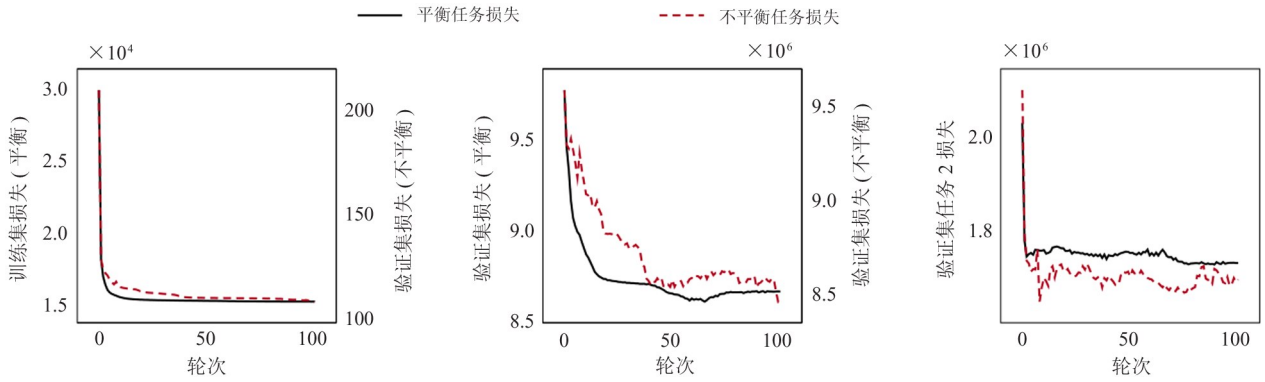

图 5　CADETS 训练时的损失变化情况

4.4　参数实验

本实验使用 DARPA TC 数据集分析了模型参数对本文方法性能的影响。具体来说,本实验测试了在不同专家数量$|\boldsymbol{E}|$、输入特征维度$|\boldsymbol{I}|$和结构嵌入维度$|\boldsymbol{H}^1|$下本文方法的精确度和召回率。图 6 ~ 图 8 分别展示了不同$|\boldsymbol{E}|$、$|\boldsymbol{I}|$和$|\boldsymbol{H}^1|$下测试的结果。图中的虚线标识了本文的默认参数设置。可以看到,默认参数下的模型在所有数据集上均取得了最优或接近最优的性能。特别地,图 6 显示出在大多数情况下,多专家取得了比单专家更优秀的性能。这验证了多专家混合模型的有效性。在开销方面,单专家模型的参数量为 24 807,默认的 4 专家模型的参数量为 28 179,仅增加 12%。就单个窗口模型的平均推理时间而言,在 3 个数据集上单专家模型为 1.7 ~ 3.1 ms,4 专家模型为 4.2 ~ 6.1 ms。鉴于检测系统的其他部分开销为 10.0 ms 至秒级别,该推理时间的差异影响甚微。

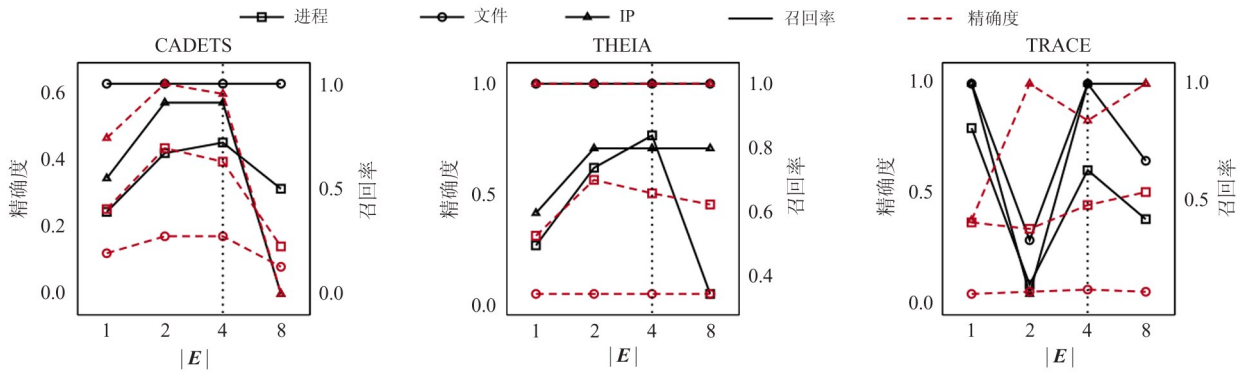

图 6　不同专家数目下的精确度和召回率

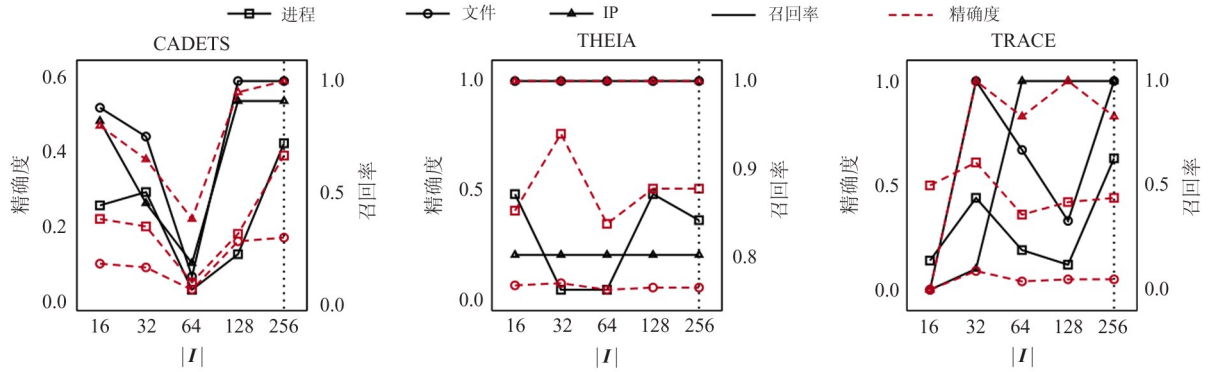

图 7　不同输入特征维度下的精确度和召回率

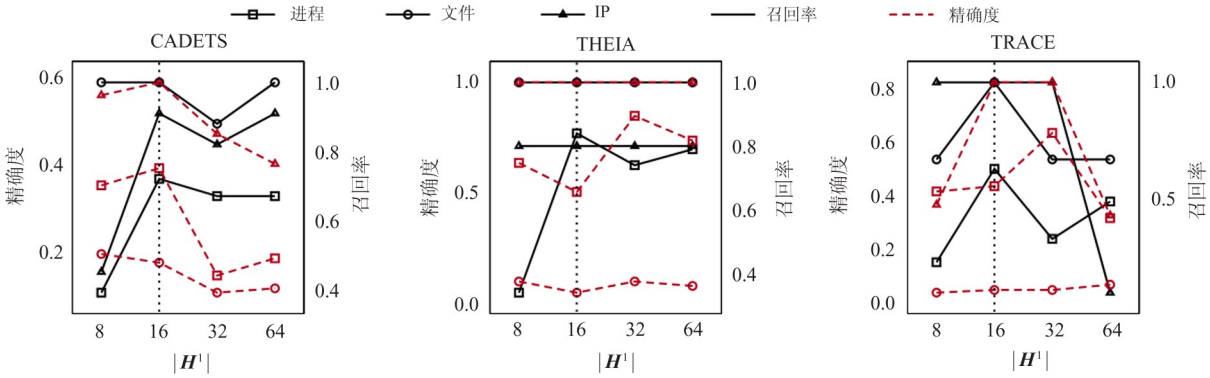

图 8　不同结构嵌入维度下的精确度和召回率

5　结　论

本文提出了一种新颖的基于混合图专家的入侵检测方法。特别地，本文方法在图学习模型中融合 GNN 和 MMOE 架构，增加了模型的表达能力。另外，本文方法针对攻击在溯源图中体现的上下文特点，设计了考虑长流和短流的 2 种下游任务。配合上游的特征重构任务，三任务的协同训练有效提升了长流结构嵌入的质量，并最终同时提升了节点级的精确度和召回率。实验结果表明，本文方法在多个数据集上取得了优于基线的结果。

参考文献

[1] Manzoor E, Milajerdi S M, Akoglu L. Fast memory-efficient anomaly detection in streaming heterogeneous graphs[C]// Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM, 2016: 1035 – 104.

[2] Hossain M N, Milajerdi S M, Wang J, et al. Sleuth: real-time attack scenario reconstruction from cots audit data[C]// The 26th USENIX Security Symposium. Vancouver, Canada: USENIX Association, 2017: 487 – 504.

[3] Gao P, Xiao X, Li Z, et al. Aiql: enabling efficient attack investigation from system monitoring data[C]// The 2018 USENIX Annual Technical Conference. Boston, USA: USENIX Association, 2018: 113 – 126.

[4] Hassan W U, Guo S, Li D, et al. Nodoze: combatting threat alert fatigue with automated provenance triage[C]// The Proceedings of Network and Distributed System Security Symposium. San Diego, USA: The Internet Society, 2019: 23349.

[5] Milajerdi S M, Eshete B, Gjomemo R, et al. Poirot: aligning attack behavior with kernel audit records for cyber threat hunting[C]// Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. London, UK: ACM, 2019: 1795 – 1812.

[6] Hossain M N, Sheikhi S, Sekar R. Combating dependence explosion in forensic analysis using alternative tag propagation semantics[C]// 2020 IEEE Symposium on

Security and Privacy. San Francisco, USA:IEEE, 2020: 1139 – 1155.

[7] Milajerdi S M, Gjomemo R, Eshete B, et al. Holmes: real-time APT detection through correlation of suspicious information flows[C]//2019 IEEE Symposium on Security and Privacy. San Francisco, USA: IEEE, 2019: 1137 – 1152.

[8] Han X, Pasquier T, Bates A, et al. Unicorn: runtime provenance-based detector for advanced persistent threats[C]// The 27th Annual Network and Distributed System Security Symposium. San Diego, USA: The Internet Society, 2020: 209862334.

[9] Wang S, Wang Z, Zhou T, et al. ThreaTrace: detecting and tracing host-based threats in node level through provenance graph learning[J]. IEEE Transactions on Information Forensics and Security, 2022, 17: 3972 – 3987.

[10] Cheng Z, Lyu Q, Liang J, et al. Kairos: practical intrusion detection and investigation using whole-system provenance[C]//2024 IEEE Symposium on Security and Privacy. San Francisco, USA: IEEE, 2024: 3533 – 3552.

[11] Li S, Dong F, Xiao X, et al. Nodlink: an online system for fine-grained APT attack detection and investigation[C]// 2024 Symposium on Network and Distributed System Security. San Diego, USA: The Internet Society, 2024: 23204.

[12] Ur Rehman M, Ahmadi H, Ul Hassan W. Flash: a comprehensive approach to intrusion detection via provenance graph representation learning[C]//2024 IEEE Symposium on Security and Privacy. Los Alamitos, USA: IEEE, 2024: 3552 – 3570

[13] Jia Z, Xiong Y, Nan Y, et al. MAGIC: detecting advanced persistent threats via masked graph representation learning[C]//The 33rd USENIX Security Symposium. Philadelphia, USA: USENIX Association, 2024: 5197 – 5214.

[14] Ma J, Zhao Z, Yi X, et al. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts[C]// Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. London, UK: ACM, 2018: 1930 – 1939.

[15] Marshall D. Winodws event tracing[EB/OL]. [2025 – 06 – 19]. https://learn.microsoft.com/en-us/windows-hardware/drivers/devtest/event-tracing-for-windows--etw-.

[16] Pasquier T, Han X, Goldstein M, et al. Practical whole-system provenance capture[C]//Proceedings of the 2017 Symposium on Cloud Computing. Santa Clara, USA: ACM, 2017: 405 – 418.

[17] Ruder S. An overview of multi-task learning in deep neural networks[EB/OL]. (2017 – 06 – 15)[2025 – 06 – 19]. https://arxiv.org/pdf/1706.05098.

[18] Caruana R. Multitask learning[J]. Machine learning, 1997, 28: 41 – 75.

[19] Baxter J. A model of inductive bias learning[J]. Journal of Artificial Intelligence Research, 2000, 12: 149 – 198.

[20] Johnson M, Schuster M, Le Q V, et al. Google's multilingual neural machine translation system: enabling zero-shot translation[J]. Transactions of the Association for Computational Linguistics, 2017, 5: 339 – 351.

[21] Shazeer N, Mirhoseini A, Maziarz K, et al. Outrageously large neural networks: the sparsely-gated mixture-of-experts layer[EB/OL]. (2017 – 01 – 23)[2025 – 06 – 19]. https://arxiv.org/pdf/1701.06538.

[22] Eigen D, Ranzato M, Sutskever I. Learning factored representations in a deep mixture of experts[EB/OL]. (2013 – 12 – 16)[2025 – 06 – 19]. https://arxiv.org/pdf/1312.4314.

[23] Chen T, Chen X, Du X, et al. Adamv-MOE: adaptive multi-task vision mixture-of-experts[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Paris, France: IEEE, 2023: 17346 – 17357.

[24] Yang E, Wang Z, Shen L, et al. Adamerging: adaptive model merging for multi-task learning[EB/OL]. (2023 – 10 – 04)[2025 – 06 – 19]. https://arxiv.org/pdf/2310.02575.

[25] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, USA: ACM, 2017: 1025 – 1035.

[26] Veli čkovi č P, Cucurull G, Casanova A, et al. Graph attention networks[C]//International Conference on Learning Representations. Vancouver, Canada: ICLR, 2018: 3292002.

[27] Weinberger K, Dasgupta A, Langford J, et al. Feature hashing for large scale multitask learning[EB/OL]. (2009 – 02 – 12)[2025 – 06 – 19]. https://arxiv.org/pdf/0902.2206.

[28] Kingma D P, Welling M. Auto-encoding variational Bayes[EB/OL]. (2013 – 12 – 20)[2025 – 06 – 19]. https://arxiv.org/pdf/1312.6114.

[29] Papachristodoulou A, Kyrkou C, Theocharides T. Driveguard: robustification of automated driving systems with

deep spatio-temporal convolutional autoencoder[C]// Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. Waikola, USA: IEEE, 2021:107 - 116.

[30] Kullback S, Leibler R A. On information and sufficiency[J]. The Annals of Mathematical Statistics, 1951, 22(1):79 - 86.

[31] Liu Y, Zhang M, Li D, et al. Towards a timely causality analysis for enterprise securit. [C]// The Proceedings of Network and Distributed System Security Symposium. San Diego, USA: The Internet Society, 2018:51951640.

[32] Grubbs F E. Procedures for detecting outlying observations in samples[J]. Technometrics, 1969, 11(1):1 - 21.

[33] Fey M, Lenssen J E. Fast graph representation learning with PyTorch geometric[EB/OL]. (2019 - 05 - 06)[2025 - 06 - 19]. https://arxiv.org/pdf/1903.02428.

[34] I2o D. Darpa TC[EB/OL]. [2025 - 06 - 19]. https://github.com/darpa-i2o/Transparent-Computing/.

[35] FiveDirections. Darpa OPTC[EB/OL]. [2025 - 06 - 19]. https://github.com/FiveDirections/OpTC-data.

[36] Liu J, Inam A, Goyal A, et al. Reapr: recovery every attack process[EB/OL]. [2025 - 06 - 19]. https://bitbucket.org/sts-lab/reapr-ground-truth.

Detecting intrusions with mixture-of-graph-experts model

Liu Dingwei[*,**], Yan Jin[***], Li Zhenyu[*,**]

([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100049)

([***] China Internet Network Information Center, Beijing 100070)

Abstract

Attacks pose an increasingly serious threat to enterprises and governments, and intrusions into hosts are particularly prominent. Provenance graphs are widely used in intrusion detection for their ability to reflect the attack context. Early graph-grained methods can only identify the occurrence of attacks but cannot locate specific nodes. Recently, more and more work has realized node-level detection based on graph learning. However, the uneven distribution of node categories and features in the training dataset undermines the generalization ability of the models, making it difficult to balance precision and recall. To solve this problem, this paper proposes a model called MGE that combines graph neural network (GNN) and multi-gate mixture-of-experts (MMOE). MGE assigns GNN expert combinations for different tasks via a gating network to capture long data flow information. In particular, the expert combination is automatically selected by the gating network according to node features, considering the differences in different parts of the feature space. To effectively train MGE, we design three types of supervise signals: upstream input reconstruction is used to improve the quality of feature representation, downstream node feature reconstruction learns long flow behavior, and edge multi-label classification learns short one-hop flow behavior. During testing, we use downstream feature reconstruction errors to locate suspicious processes and construct alarms. Experimental results show that the proposed method simultaneously improves accuracy and recall rate and achieves better results than baselines in multiple datasets.

Key words: provenance graph, intrusion detection, graph neural network, multi-task learning, mixture-of-expert