中国科学：信息科学
SCIENTIA SINICA Informationis
论文
《中国科学》杂志社
SCIENCE CHINA PRESS

基于序列特征提取的溯源图上 APT 攻击检测方法

梁若舟，高跃，赵曦滨*

清华大学软件学院，北京信息科学与技术国家研究中心，信息系统安全教育部重点实验室，北京 100084
* 通信作者. E-mail: zxb@tsinghua.edu.cn

收稿日期: 2021–07–26; 修回日期: 2021–09–19; 接受日期: 2021–12–01

国家重点研发计划 (批准号: 2018YFB1703404)、国家自然科学基金 (批准号: 62076146, U1801263, U20A6003, U19A2062) 和广东省重点领域研发计划 (批准号: 2020B010164001) 资助项目

摘要 在真实场景，特别是工业场景下的高级持续性威胁 (advanced persistent threat, APT) 具有复杂性和长期性，但当前方法无法有效提取攻击中的长期关联关系. 针对这一问题，本文提出一种基于溯源图的 APT 攻击检测方法 SeqNet. SeqNet 采用序列特征提取方式，实现 APT 攻击检测. 在 SeqNet 中，首先将描述系统运行状态的溯源图序列转化为特征向量序列，然后使用 GRU (gate recurrent unit) 模型提取系统状态变化特征，并使用结合局部注意力机制的编解码器模型训练 GRU 模型，最后利用 K-means 聚类方法对系统正常行为进行建模. 本文在 5 个公开数据集 StreamSpot, wget, shellshock, ClearScope 和 CADETS 上进行了实验，并与当前具有代表性的方法进行了对比. 本文方法在 5 个数据集上都取得了相似或更好的效果. 实验结果证明本文方法能够实现真实场景下的 APT 攻击检测.

关键词 APT 攻击检测，溯源图，异常检测，序列特征提取，局部注意力机制

1 前言

随着信息技术的飞速发展，对企业信息系统的攻击日益频繁，给企业和社会带来很大的经济损失. APT (advanced persistent threat) 攻击通常具有长期性和隐蔽性. APT 攻击的实施者为了达到攻击目的，可能会针对特定的系统和软件，组合使用多种攻击方式，逐步在系统中渗透，同时在攻击过程中也可能会利用零日漏洞. APT 攻击的这些特点使得传统的安全设备以及防御方法难以有效地检测和防御 APT 攻击.

近年来，越来越多的研究工作聚焦在使用溯源图来记录系统的行为，利用溯源图进行 APT 攻击检测和攻击场景的还原等任务. 溯源图是根据系统中系统调用日志生成的具有有向图结构的数据，可以用于描述系统行为. 所有系统级别的实体被当作溯源图中的节点，而实体之间的操作被当作溯源图

引用格式: 梁若舟, 高跃, 赵曦滨. 基于序列特征提取的溯源图上 APT 攻击检测方法. 中国科学: 信息科学, 2022, doi: 10.1360/SSI-2021-0252
Liang R Z, Gao Y, Zhao X B. Sequence feature extraction-based APT attack detection method with provenance graphs (in Chinese). Sci Sin Inform, 2022, doi: 10.1360/SSI-2021-0252

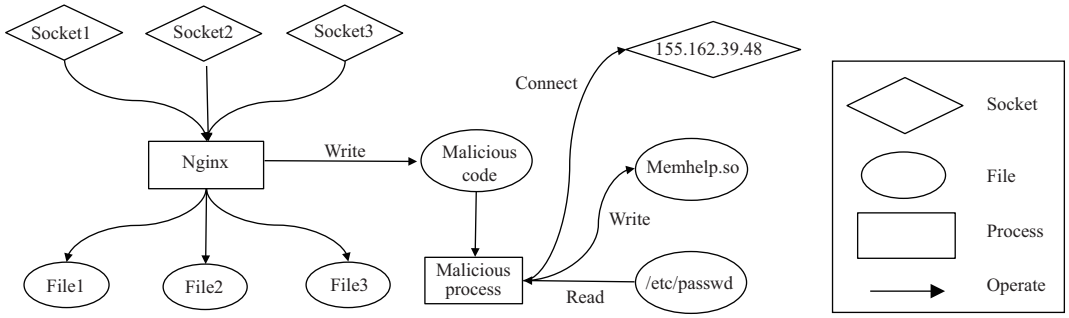

图 1　一个溯源图的例子

Figure 1　An example of the provenance graph

的边. 具体来说, 图中的节点分为两类, 一类被称为主体, 包括系统中存在的进程等. 另一类节点被称为客体, 指系统中的文件、套接字等. 图中的一条有向边表示一个事件, 比如进程 A 读入了文件 B, 进程 C 创建了进程 D. 一条边可以用一个四元组 ⟨ 主体, 客体, 时间戳, 操作 ⟩ 来描述. 边的方向表明了系统中数据的流动方向. 图 1 是溯源图的一个例子. 它来自于 DARPA TC 数据集, 是系统溯源图的一个子图. 在图中, 长方形表示进程, 椭圆形表示文件, 菱形表示套接字, 有向箭头表示一种操作. 在该溯源图中, 完整地记录了一个恶意程序读取敏感文件/etc/passwd, 修改文件 memhelp.so, 以及连接 IP 为 155.162.39.48 的主机. 从溯源图的定义可以看出其具有两点优势: 一是操作系统的系统调用是一种底层的操作, 大多数的攻击方式无法避免使用系统调用, 所以溯源图能够捕捉到攻击者留下的痕迹; 第二, 溯源图记录了系统中主客体之间的调用关系和数据流动关系, 所以在溯源图不被篡改的情况下, 攻击者的所有攻击行为会在溯源图上形成一个完整的攻击链路, 有利于检测以及还原复杂的攻击场景.

由于溯源图保留了丰富的系统行为信息, 近年来, 越来越多的研究者开始关注基于溯源图的攻击检测和还原的方法. 对于一些已有工作的分析如表 1 [1~5] 所示. Hossain 等 [1] 使用标签传播的方法在溯源图上寻找攻击路径, 该方法需要设置规则确定可疑节点和标签传播策略. Milajerdi 等 [3] 将溯源图映射到杀伤链模型中, 形成高级场景图, 并判断这些高级场景图是否为 APT 攻击, 映射是通过规则完成的. Milajerdi 等 [2] 提出使用图匹配的方法在溯源图上检测攻击子图, 该方法需要预先定义攻击子图, 且无法对未知攻击告警. 这些方法都依赖于规则设计和专家经验, 无法自动从数据中学习到正常行为或者攻击行为模式. Manzoor 等 [4] 使用溯源图局部子图的相似性定义溯源图之间的相似性, 通过聚类算法将相似的溯源图聚集在一起, 但判断整张溯源图的相似性无法体现攻击时序性. Han 等 [5] 提出基于机器学习的溯源图上攻击检测算法 UNICORN, 能够自动构建系统行为模型. UNICORN 首先将用于描述系统变化的溯源图序列转化为特征向量序列, 对特征向量聚类产生系统状态, 利用自动机描述系统状态变化, 并且 UNICORN 为每个正常序列建立一个自动机模型. 但是 UNICORN 依然存在不足之处:

• 在真实场景, 特别是工业场景中, APT 攻击具有持续时间久且隐蔽性强的特点. 在长序列中, 单个特征向量与正常行为对应的特征向量差异较小. UNICORN 使用自动机建模系统状态变化, 这种方式无法将这种差异累积, 因此无法有效地检测真实场景, 特别是工业场景下的长周期的 APT 攻击.

• UNICORN 对于每个训练数据都建立一个自动机模型, 在检测过程中需要尝试匹配所有自动机模型. 当收集到的训练数据增加时, 检测时间也会线性增加. 但是在真实场景中, 系统的正常行为是复杂的, 需要大量模型描述. 所以多模型的方法会影响检测效率.

APT 攻击具有隐蔽性和长期性, 这使 APT 攻击检测成为难点. APT 的隐蔽性和长期性意味着,

表 1　前人工作分析

Table 1　Analysis of previous work

	Rule-based	Need attack knowledge	Long-term attack
SLEUTH [1]	√	×	Find false attack path
HOLMES [3]	√	√	Find false attack scene
POIROT [2]	√	√	Imprecise attack graph matching
StreamSpot [4]	×	×	Hard to extract long-term features
UNICORN [5]	×	×	Hard to extract long-term features

在攻击过程中, 攻击行为所产生的数据占整个系统产生的数据的比例很低, 所以从短期采集到的数据来看, 不容易区分正常行为和异常行为. 但是从长期看, 由于异常行为产生数据的累计, 会体现出正常行为与异常行为的差异. 例如, 系统中一个进程与未访问过的外网 IP 进行通信, 从短期看, 很难判断这一行为是否异常, 因为系统会和很多 IP 产生通信. 但如果产生长时间、持续性的通信, 则有可能是系统被植入恶意软件, 恶意软件接收指令或向外界发送获取的敏感数据. 针对这一特点, 需要检测方法能够捕捉短期的微小异常, 形成长期的明显异常. 如图 2(a) 中, 从含有异常的短期溯源图中提取的特征和从正常溯源图中提取的特征距离很近难以区分, 在图 2(b) 中, 通过累计短期的差异, 拉大正常行为与异常行为在特征空间的距离.

针对 APT 攻击的特点以及现有方法的不足, 本文提出了一个新的方法, 取名为 SeqNet (sequence networks). SeqNet 首先使用与 UNICORN 同样的方法将溯源图序列转化为特征向量序列. 之后对输入特征序列中所有特征向量进行降维, 去除冗余信息. 然后使用能够有效提取长序列特征的 GRU (gate recurrent unit) 模型作为特征提取网络, 从特征序列中抽取一个包含序列长期信息的特征向量. 在安全领域中, 由于 APT 攻击方式和策略多种多样, 可能会利用系统中存在的不同漏洞, 甚至于使用零日漏洞来达到窃取机密信息, 破坏系统或者其他目的, 难以对所有攻击行为建模, 所以本文只对系统正常行为建模. 为了仅使用正常数据训练特征提取网络, 本文采用编码器 – 解码器架构, 重构输入的特征序列, 通过最小化重构误差的方式训练网络. 同时为了解决训练过程中模型存在的准确性不高以及迭代次数过多的问题, 本文使用了局部注意力机制, 利用 GRU 模型构建了一个局部注意力网络, 使解码器网络能够更有效地利用输入序列的信息. 最后本文使用聚类算法, 将序列特征聚为 K 类, 作为系统的正常行为模型.

本文的主要贡献如下:

• 针对 APT 攻击具有的长期性、隐蔽性等特点, 本文使用 GRU 模型建模系统的状态变化, 提取 APT 攻击对系统产生的长期影响, 使得攻击行为的特征与正常行为的特征更易区分, 解决当前方法存在的无法有效提取攻击中的长期特征的问题.

• 针对真实环境中通常只能收集到正常行为数据的问题, 使用结合局部注意力机制的编码器 – 解码器的神经网络结构提取特征, 并通过聚类建立正常行为模型, 有效解决了攻击数据难于收集的问题.

• 本文在 5 个公开的 APT 攻击数据集上进行了实验, 并与当前具有代表性的方法进行了比较. 本文的方法在 5 个数据集上都取得了相似或更好的效果, 实验结果证明了本文方法能够有效地检测出 APT 攻击.

本文的组织结构如下: 第 1 节简述了 APT 攻击检测的背景和意义. 第 2 节回顾了当前对溯源图的研究和基于溯源图的攻击检测方法. 第 3 节详细介绍了本文提出的基于溯源图的攻击检测算法. 第 4 节详细介绍了在 5 个公开数据集 StreamSpot, wget, shellshock, ClearScope 和 CADETS (causal,

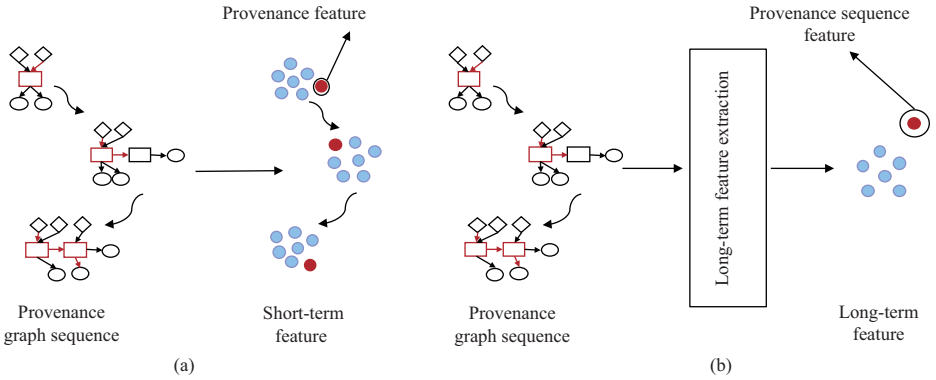

图 2　(网络版彩图) 文章动机

Figure 2　(Color online) Motivation. (a) Short-term feature; (b) long-term feature

adaptive, distributed, and efficient tracing system) 上的实验结果和分析. 第 5 节讨论了本文方法的不足. 第 6 节是本文的结论.

2　相关工作

由于溯源图具有丰富的操作系统级调用关系的信息, 近年来许多科研工作者聚焦在与溯源图相关的研究上, 本节将对这些工作进行综述和分析. 本节主要分为两个部分, 2.1 小节介绍对于溯源图的研究, 包括溯源图的采集和压缩, 2.2 小节介绍基于溯源图的攻击检测的工作.

2.1　溯源图采集和压缩

系统日志采集是所有溯源图相关研究的第 1 步. 在 Linux 平台下, 工具 Auditd 可以用于采集系统日志. 在 Windows 平台下, 可以利用 ETW (event tracing for windows) 获取系统日志信息. 除此之外, 研究者还构建了很多溯源收集系统, 如 PASS [6], CamFlow [7], SPADE [8], Story Book [9], TREC [10].

机器每天会产生大量系统级日志, 需要研究溯源图压缩算法降低存储压力, 同时提升检测效率. 溯源图压缩一般分为节点压缩和边压缩. 节点压缩的方式包括节点删除和节点合并. Lee 等 [11] 提出删除孤立的临时节点, 删除这些节点不会影响溯源图的因果分析. Tang 等 [12] 提出 NodeMerge, 可以自动学习一些模板来压缩溯源图. 边压缩主要研究如何在保留溯源图基本性质的情况下, 删除冗余的边. Xu 等 [13] 研究如何在保持溯源图中因果关系不变的条件下去除冗余边. Hossain 等 [14] 提出在保留溯源图依赖关系的条件下进一步提高压缩率的方法.

2.2　基于溯源图的攻击检测

由于溯源图具有丰富的结构信息, 越来越多的研究者利用溯源图实现各种任务, 如攻击检测、攻击场景还原, 以及消除虚假告警等. 对溯源图最早的研究工作可以追溯到文献 [15], 研究者通过从溯源图上的一个攻击点出发, 在溯源图上进行反向搜索, 寻找那些能够到达攻击点的所有节点, 这些节点在溯源图上构成的路径就是可能的攻击路径. 近些年来, 基于溯源图的研究工作逐渐丰富起来. Xie 等在文献 [16,17] 中建立了正常规则库, 在文献 [18] 中建立异常规则库, 通过在溯源图上进行搜索, 生成待检测路径, 计算每条路径的异常分数, 再合成整个溯源图的异常分数, 以此作为判断是否存在攻击的依据. NoDoze [19] 通过计算溯源图上路径异常分数的方式消除告警疲劳. Sun 等 [20,21] 提出使用

贝叶斯 (Bayes) 网络建模溯源图, 先将可疑信息对应到贝叶斯网络中的节点, 然后根据条件概率表计算其他节点被感染的概率, 从而找出零日攻击路径. Omegalog [22] 以及 ALchemist [23] 通过将系统级的溯源图与应用级的日志相结合, 丰富了溯源图信息. Hossain 等 [1] 提出了 SLEUTH, 利用溯源图进行实时攻击场景还原. SLEUTH 通过规则首先对溯源图中所有节点赋予初始标签, 通过标签传播策略更新节点标签, 根据边上节点的标签确定边权重, 从异常节点出发, 进行后向分析, 利用单源最短路径算法找到入度为 0 的入口节点, 以入口点为起点进行前向搜索, 并根据路径权重进行剪枝, 搜索得到的就是重建后的攻击场景. Hossain 等 [24] 针对 SLEUTH 在处理长期攻击时生成的图中含有大量良性节点的问题, 在标签传播过程中引入了两种衰减策略, 减少了良性节点被识别为可疑节点的情况, 解决了依赖爆炸的问题. 溯源图虽然包含丰富的系统级别的信息, 但是缺少高阶的攻击语义信息. 为了解决这个问题, Milajerdi 等 [3] 建立了一套映射规则, 将含有低阶信息的溯源图, 映射到含有高阶语义的杀伤链模型中, 形成高级场景图, 并计算高级场景图的异常分数, 超过阈值的识别为 APT 攻击. 虽然这些方法都能够识别攻击, 但是需要领域知识和专家经验, 效果取决于规则设计. Milajerdi 等 [2] 提出使用图匹配的方法识别溯源图中的攻击. 先利用领域知识构建攻击模式图, 将溯源图中和攻击图相似的子图识别为攻击. 虽然基于图匹配的方法能比较准确地找到攻击, 但是需要根据先验知识构建攻击图, 并且无法检测出未知的攻击手段. UNICORN [5] 使用机器学习的方法, 首先将溯源图转化为特征序列, 然后对特征序列进行聚类, 生成 K 个状态, 利用自动机建模 K 个状态之间的转移. UNICORN 能够不借助专家经验检测未知攻击. 但是依然存在着自动机对特征序列描述能力不强, 以及大量自动机会影响检测效率等问题.

3 方法

本文提出的基于溯源图的攻击检测方法 SeqNet 的框架图如图 3 所示. 该方法主要分为 3 个阶段: 第 1 阶段是溯源图特征提取, 该阶段将反映系统变化的溯源图序列抽取为特征序列; 第 2 阶段是序列特征提取, 该阶段使用结合局部注意力机制的编解码器模型训练一个序列特征提取器; 第 3 个阶段是行为建模, 该阶段使用聚类算法对系统正常行为建立 K 个聚类中心, 并在测试阶段将远离聚类中心的数据识别为攻击行为. 本节将分为 3 个小节详细描述 3 个阶段.

3.1 溯源图特征提取

在基于溯源图的攻击检测框架中, 第 1 个模块的功能是将系统调用日志流提取成为能够反映系统状态变化的特征序列, 如图 3 中特征序列提取模块所示. 该模块采用 UNICORN 中生成溯源图特征序列的方式, 具体分为以下 3 个步骤: 步骤一获取系统调用日志并且根据这些日志构建出溯源图; 步骤二利用溯源图节点以及其邻域节点信息, 生成节点标签, 并以直方图的形式记录溯源图节点标签的统计信息; 步骤三将直方图转化为特征向量.

步骤一的目的是生成系统溯源图. 首先收集系统调用日志, 然后从日志中提取主体和客体作为溯源图节点, 根据节点之间的调用关系生成溯源图的边. 溯源图可以使用 CamFlow [7] 或者其他工具生成.

步骤二的目的是将一个溯源图转化为一个直方图. 给定一个溯源图 G, 利用图同构问题中的 WL 测试以及与之相关的 WL 子树图核函数, 在每轮迭代中, 可以对 G 中的节点赋予新的标签, 迭代 K 轮之后, 该标签包含了该节点 K 跳邻域内邻居节点的信息. 具体来说, 第 1 步融合距离为一跳的节点信息, 针对溯源图中的每个节点 v, 计算该节点的入边集合, 对于集合中的每一条边, 将边标签与源节点

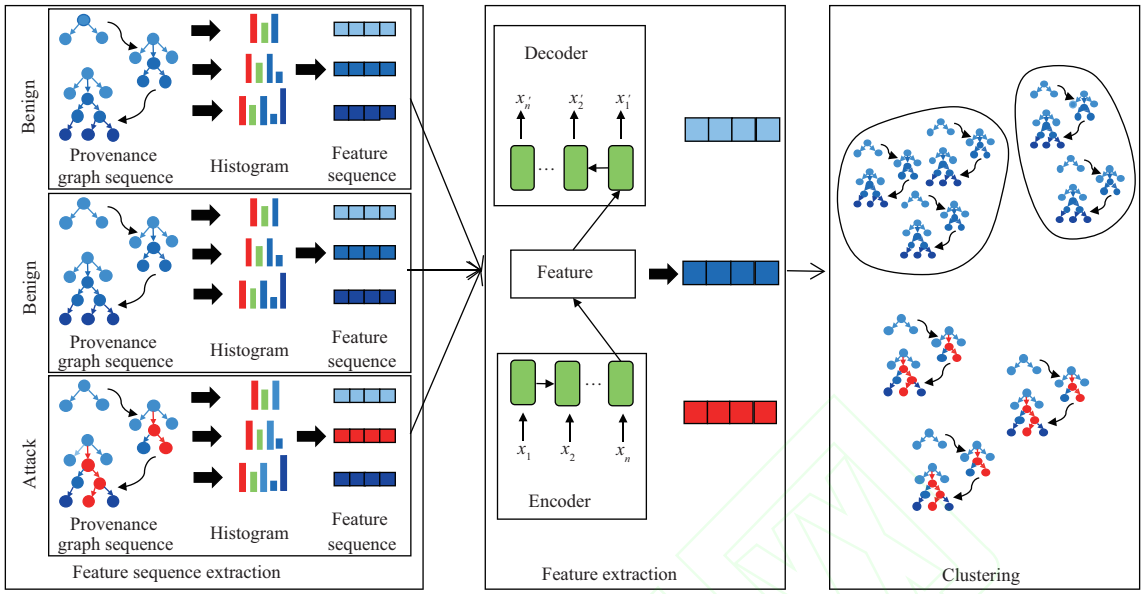

图 3　(网络版彩图) SeqNet 的框架. SeqNet 主要分为 3 个部分, 第 1 部分将溯源图序列抽取为特征序列. 第 2 部分使用 GRU 模型将特征序列转化为一个特征向量. 第 3 部分使用聚类算法将系统正常行为聚集为 K 类

Figure 3　(Color online) The framework of SeqNet. SeqNet is mainly divided into three parts. In the first part, the provenance sequence is extracted into feature sequence. In the second part, the feature sequence is encoded into a feature vector using GRU model. The third part uses clustering algorithm to cluster the normal behavior into K classes

标签与 v 的标签结合, 作为 v 的新标签, 这个标签是包含了 v 的邻接边和节点信息的; 第 2 步迭代融合距离为 K 跳的节点信息, 在每一轮迭代中, 针对每个节点 v, 计算其邻接节点集合 $N(v)$, 将 v 的标签和 $N(v)$ 中所有节点的标签放入集合 S_v 中, 利用哈希 (Hash) 函数对 S_v 生成一个编号, 这个编号就作为 v 的新的标签. 容易看出, 节点 v 的标签包含了其 K 阶邻域信息; 第 3 步绘制统计直方图, 更新溯源图中节点的标签之后, 需要统计每个标签出现的次数, 以标签为横坐标, 出现次数为纵坐标形成直方图. 在流式更新直方图过程中, 作者引入了遗忘因子, 即对越早出现的标签赋予的权重越小, 对某个标签 l_i 的纵坐标的计算公式如下:

$$l_i = \sum_{j=1}^{n} \mathrm{e}^{-\lambda t_j}, \tag{1}$$

其中 n 表示标签 l_i 一共出现了 n 次, t_j 表示第 j 次出现的时间.

步骤三将溯源图对应的直方图转化为一个特征向量. 由于步骤二中可能会产生未出现过的节点标签, 所以直方图的横坐标数量是不确定的, 而一般的算法需要一个固定长度的特征向量作为输入. HistoSketch [25] 算法能够有效地解决这一问题, 将步骤二生成的直方图转化为一个固定长度的特征向量.

3.2　序列特征提取

在第 1 个模块中, 将由系统调用日志流生成的溯源图序列转化为了一个特征向量的序列, 本文将这个序列记为 $S = \{f_1, f_2, \ldots, f_n\}$, 其中 $f_i \in \mathbb{R}^d$ 表示第 i 个特征向量, 其维度是 d 维, n 表示序列长度. 生成描述系统变化的特征序列之后, 需要对这个序列进行建模. 在 UNICRON 中, 作者利用聚类将 n 个特征聚成 K 类, 表示 K 个状态, 用自动机的方式描述这 K 个状态之间的转移关系, 以此对序

列进行建模. 但是这种方式本身对于序列的长期特征捕捉能力不足, 所以本文使用能够有效建模时序信息的 GRU 模型对系统的特征序列建模. 但是由于在安全领域中, 存在着攻击数据获取困难以及对攻击行为建模难以涵盖所有攻击方式等问题, 所以不适合同时使用系统正常行为数据和攻击数据对模型进行有监督地训练. 本文选择仅使用正常数据训练 GRU 模型, 并且使用局部注意力机制提升模型的表示能力. 该模块将分为数据降维部分和序列特征提取部分, 本小节分别介绍两个部分.

在输入的特征的序列 S 中, 序列中的特征 f_i 描述了系统在 i 时刻的状态, 所以当两个系统状态相似的时候, 比如相邻的系统状态, 对应的两个特征向量也是相似的. 这就使得直接从溯源图提取出的特征维度较高, 含有大量冗余信息. 所以在该模块的第 1 部分, 首先使用多层感知机, 对序列中所有特征降低维度, 生成低维空间中更紧致的特征向量. 通过这一方式既可以去除冗余信息, 减少计算量, 提升攻击检测效率, 同时也使得输入到后续模块中的特征中有效信息的比重提升. 将降维之后的特征序列记为 $X = \{x_1, x_2, \ldots, x_n\}$.

对特征进行降维之后, 序列特征提取部分将特征序列转化为一个特征向量. 本小节将分为基于编码器 – 解码器架构的序列特征提取以及局部注意力机制两个方面介绍特征提取部分.

由于 GRU 模型是之后编码器模块、解码器模块, 以及注意力机制模块中所使用的模型, 所以先介绍 GRU 模型. GRU 模型是一种有效的时间序列建模方法, 它能够捕捉长期序列特征. GRU 模型的单次迭代公式如下:

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}), \tag{2}$$

$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}), \tag{3}$$

$$n_t = \tanh(W_{in}x_t + b_{in} + r_t * (W_{hn}h_{t-1} + b_{hn})), \tag{4}$$

$$h_t = (1 - z_t) * n_t + z_t * h_{t-1}, \tag{5}$$

其中 x_t 表示第 t 次迭代的输入, h_{t-1} 表示第 $t-1$ 次迭代输出的隐藏特征, W_{ir}, W_{hr}, W_{iz}, W_{hz}, W_{in}, W_{hn} 表示映射矩阵, b_{ir}, b_{hr}, b_{iz}, b_{hz}, b_{in}, b_{hn} 表示偏置向量. 其中最重要的两个概念是式 (2) 中的重置门 r_t 和式 (3) 中的更新门 z_t. 式 (4) 中重置门赋予过去信息不同的权重, 并且与当前输入的信息相结合, 得到当前候选特征 n_t. 在式 (5) 中, 由当前候选特征和过去隐藏特征生成当前隐藏特征, 两者的权重由更新门控制.

在 APT 攻击过程中, 攻击方式多样甚至经常使用零日漏洞, 难以对所有攻击行为建立模型, 所以本文采用正常行为建模的方式建立模型. 本文采用基于编码器 – 解码器的模型训练 GRU 模型提取序列特征. 而在测试阶段, 仅保留编码器部分的模型和参数.

结合了局部注意力机制的编码器 – 解码器模型如图 4 所示. 其中编码器的作用是将特征序列编码成为一个特征向量, 该特征向量包含了整个序列的信息, 所以编码器模块采用了能够有效地提取长期序列特征的 GRU 模型. 编码器模块的形式化描述如下所示:

$$\text{Output}, \text{feature} = \text{Encoder}(X). \tag{6}$$

将经过降维之后的特征序列 X 输入到由 GRU 模型构成的编码器模块中, 输出包括特征序列 Output 和特征向量 feature, 其中 Output$\in \mathbb{R}^{n\times d}$ 是由特征向量组成的序列, 其特征维度等于输入的序列中特征向量的维度, 序列长度等于输入序列的长度. 在每一轮迭代过程中, GRU 模型都会输出一个特征向量, 这个特征向量包含了输入特征序列的局部特征信息. 这些输出的特征向量按照顺序放在一起, 构成了输出特征序列 Output, 这个特征序列是对输入特征序列的优化. 同时将最后一轮迭代之后的输

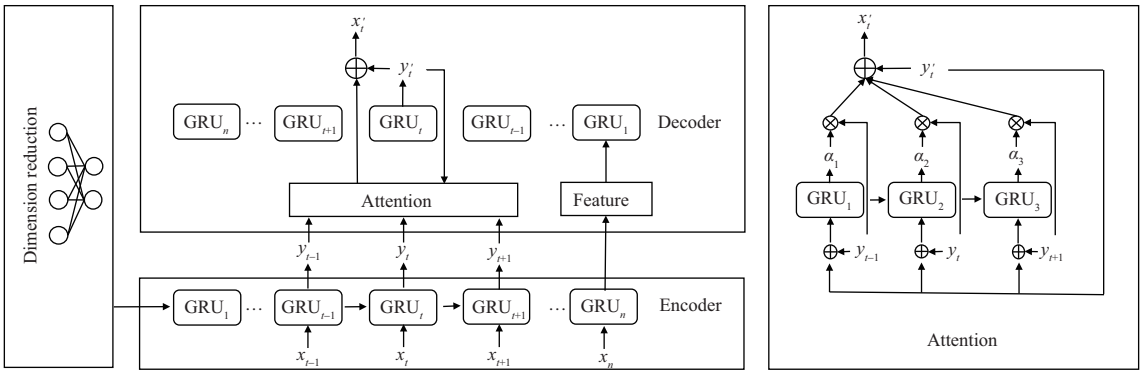

图 4 (a) 序列特征提取模块的整体结构, 数据降维模块对特征序列中每个特征降维, 编码器模块将特征序列编码为特征向量, 解码器模块使用局部注意力机制, 将特征向量恢复为特征序列, 通过最小化原始序列和恢复后序列误差的方式训练网络. (b) 注意力模块的详细结构, 通过 GRU 模型分配注意力系数

Figure 4 (a) The structure of the sequence feature extraction module. The data dimensionality reduction module reduces the dimension of each feature in the feature sequence. The encoder module encodes the feature sequence into a feature vector. The decoder module uses the local attention mechanism to restore the feature vector into a feature sequence, and trains the network by minimizing the error of the original sequence and the recovered sequence. (b) The detailed structure of the attention module, which allocates the attention coefficient through the GRU model

出作为特征向量 feature, feature 融合了整个序列的特征, 包含了序列长期的历史信息, 能够反映出整个序列的特征. 在攻击检测阶段, 直接输出 feature 作为后续聚类模块的输入.

解码器模块的功能是根据编码器生成的序列特征 feature, 重构出输入的特征序列 X. 解码器模块的伪代码如算法 1 所示. 本文同样以 GRU 模型作为基础搭建解码器. 因为 feature 中包含了原始特征序列 X 中的信息, 所以将 feature 作为输入向量, 输入到解码器中, 将 feature 记为 y'_0. 不同于编码器部分, 由于输入只是一个特征向量而不是一个特征序列, 所以在每轮迭代过程中, 需要将上一轮 GRU 模型的输出作为下一次的输入, 解码器迭代公式如伪代码第 2 行所示. 其中 $y'_{t-1} \in \mathbb{R}^d$, y'_{t-1} 是上一轮中 GRU 模型的输出. 在 n 次迭代之后, 将每次输出的特征向量 y'_i 按照顺序排列好, 就得到解码器的输出特征序列 $Y' = \{y'_1, y'_2, \ldots, y'_n\}$.

Algorithm 1 Decoder

Input: Feature y'_0, hidden state h_0, feature sequence Y, local attention range R.
Output: Feature sequence X'.

1: **while** $t = 1, 2, \ldots, n$ **do**
2: $y'_t, h'_t = \mathrm{GRU}(y'_{t-1}, h'_{t-1})$;
3: $\mathrm{left} = \max(0, t - \frac{R-1}{2})$;
4: $\mathrm{right} = \min(n, t + \frac{R-1}{2})$;
5: $Y_{\mathrm{local}} = Y[\mathrm{left} : \mathrm{right}]$;
6: $\mathrm{Weight} = \mathrm{Attion}(y'_t, Y_{\mathrm{local}})$;
7: $x'_t = y'_t + Y_{\mathrm{local}} \times \mathrm{Weight}$;
8: $X'[t] = x'_t$;
9: **end while**

如果直接使用 Y' 作为重构出来的特征序列, 会面临两个问题: 一是模型准确性不高的问题, 二是模型训练过程中的收敛速度慢的问题. 造成这两个问题的原因是在解码过程中, 仅使用了编码器模块输出的序列特征 feature 作为输入. 虽然编码过程中使用的 GRU 模型能够捕捉序列长期依赖, 最终生

成的特征向量能够反映整个序列的特征, 但在编码过程中, 经过当前特征与历史特征的加权, 会损失部分信息, 所以仅从这一特征向量恢复出来的序列与输入的原始序列相比, 有较大的信息损失, 这就会导致模型准确性下降. 同时也会加大训练过程中寻找到合适参数的难度, 从而使得训练需要迭代的次数大大增加. 注意力机制 [26] 的引入, 能够有效地解决这一问题.

本文设计了一个时序局部注意力机制, 能够有效减少模型在训练过程中的迭代次数, 并且提高模型检测效果. 对于输入的一个特征序列来说, 该序列包含了系统行为随时间变化的信息. 系统行为在长期会有较大变化, 但是在短期中, 系统行为会有较强地关联. 在特征序列中, 局部的子序列包含了短期的时序信息. 在解码阶段, 编码器生成的特征向量包含了原始特征序列的全局特征, 如果在恢复原始序列中某一个特征向量时, 能够结合其局部子序列的特征, 解码器的性能会有较大提升. 本文使用了一个 GRU 模型作为局部注意力机制的实现. 本文选用 GRU 网络生成权重, 是因为在子特征序列中, 也蕴含着时序信息, 在权重计算过程中, 需要考虑特征在时序上的关联性. 算法 1 中的第 3~8 行是相关伪代码. 在第 t 次循环中, 重构第 t 个特征向量, 需要用到编码器输出的特征序列 $Y' = \{y'_1, y'_2, \ldots, y'_n\}$ 中在区间 $[t - \frac{R-1}{2}, t + \frac{R-1}{2}]$ 中的特征向量子序列, 记作 Y_{local}. 对于左右边界情况, 舍弃超出边界范围的部分. 对于这个区间中的所有特征, 需要经过加权之后, 与解码器的输出 y_t 融合. 每个向量的权重除了与 Y_{local} 相关, 还与解码器当前输出 y_t 有关, 所以先将 y_t 复制 R 份, 然后与 Y_{local} 拼接形成新的序列 F, 如下所示:

$$F = \text{concat}(y'_t, Y_{\text{local}}), \tag{7}$$

其中 $F \in \mathbb{R}^{(2k+1)\times 2d}$. 之后将 F 输入到局部注意力网络中, 如下所示:

$$w = \text{GRU}(F), \tag{8}$$

其中 $w \in \mathbb{R}^{2k+1}$. 利用公式

$$\text{Weight}_i = \frac{e^{w_i}}{\sum_{i=0}^{2k} e^{w_i}}, \tag{9}$$

将 w 归一化到 $[0, 1]$ 区间内得到 Weight 就是 Y_{local} 对应的权重. 求出 Y_{local} 的加权和, 再与解码器输出 y_t 求和, 得到重构出的第 t 个特征向量 x'_t. n 次循环之后, 将所有重构出的特征向量放在一起构成一个序列, 记为 X', $X' \in \mathbb{R}^{(n\times d)}$.

本文通过最小化输入特征序列和重构出地特征序列之间的误差的方式训练网络. 损失函数定义如下所示:

$$\text{loss} = \|X - X'\|_{\text{F}}, \tag{10}$$

两个矩阵之间的差异用矩阵的 F 范数来度量.

以上过程是对网络的训练过程, 在攻击检测阶段, 将去掉解码器部分, 只保留编码器部分, 编码器输出结果也只保留序列特征 feature.

3.3 行为建模

由于在训练过程中, 只有系统的正常行为数据, 所以本文采用聚类的方式, 将从训练集提取出的全部特征向量聚集为 K 类. 其中 K 值是一个超参数, 需要提前设置. 通过参数实验可知, K 在 7 附近的范围内都有较好的表现, 本文将聚类数目设置为 7. 这些特征向量包含了系统行为特征, 具有相似行为性的系统行为, 会在特征空间比较接近. 在攻击检测阶段, 计算抽取的特征向量与聚类中心的距离, 将超过阈值的行为判定为攻击. 由于聚类数目远小于训练数据的数目, 与 UNICORN 相比, 本文方法所

表 2 StreamSpot 数据集的属性

Table 2 Characteristics of the StreamSpot dataset

Dataset	Label	Number of graphs	Average $\|V\|$	Average $\|E\|$	Preprocessed data size (GB)
StreamSpot	YouTube	100	8292	113229	0.3
	Gmail	100	6827	37382	0.1
	Download	100	8831	310814	1
	VGame	100	8637	112958	0.4
	CNN	100	8990	294903	0.9
	Attack	100	8891	28423	0.1

构建的模型数量不会随着训练数据数目增加而线性增加. 这样能够保证在收集充足的训练数据同时, 不影响检测效率.

4 实验

在本节中, 将详细介绍实验部分. 为了评价本文提出的方法, 本文在 5 个公开数据集上进行了实验. 本节将从以下几个方面依次展开叙述: 第 1 部分将介绍实验所采用的 3 组共 5 个公开数据集. 第 2 部分将介绍方法的实现、参数的设置, 以及实验设置. 第 3 部分简要介绍两种当前具有代表性的基于溯源图的攻击检测方法 UNICORN 算法和 StreamSpot 算法. 第 4 部分将介绍与 UNICORN 算法和 StreamSpot 算法的对比实验结果. 第 5 部分将讨论聚类数量 K 以及局部注意力机制范围 R 这两个超参数对方法的影响. 第 6 部分将讨论局部注意力机制对模型的影响.

4.1 数据集

在这一小节, 将介绍实验所使用的数据集. 数据集共分为 3 组, 分别是 StreamSpot, DARPA TC 和 SC, 其中 DARPA TC 数据集和 SC 数据集中各包含 2 个数据集.

4.1.1 StreamSpot 数据集

StreamSpot[1)]数据集中一共包含 6 个场景, 其中 5 个正常行为, 包括看 YouTube 视频、检测邮件等, 以及一个攻击行为. 攻击场景是从一个恶意 URL 下载程序并利用一个闪存漏洞获取系统管理员权限. 该数据集收集过程中, 每个场景运行了 100 次, 并使用 SystemTap 记录信息. StreamSpot 数据集的详细信息如表 2 所示.

4.1.2 DARPA TC 数据集

DARPA TC 数据集来自美国国防高级研究计划局 (Defense Advanced Research Projects Agency, DARPA) 的透明计算 (transparent computing, TC) 项目. 该项目组织红队蓝队进行攻防演练, 在此过程中, 收集系统细粒度行为数据, 进行攻击检测和取证溯源, 并形成报告. 本文所使用数据集来自为期两周的第 3 次攻防对抗, 从不同平台收集得到. CADETS 数据集是在 FreeBSD 系统上使用 CADETS 收集得到. ClearScope 是在 Android 系统上收集得到. 数据集具体的统计信息如表 3 所示.

1) StreamSpot datasets. https://github.com/sbustreamspot/sbustreamspot-data.

表 3 DARPA TC 数据集的属性

Table 3 Characteristics of the DARPA TC dataset

Dataset	Label	Number of graphs	Average $\|V\|$	Average $\|E\|$	Preprocessed data size (GB)
ClearScope	Benign	43	2309	4199309	441
	Attack	51	11769	4273003	432
CADETS	Benign	110	59983	4811836	271
	Attack	3	386548	5160963	38

表 4 SC 数据集的属性

Table 4 Characteristics of the SC dataset

Dataset	Label	Number of graphs	Average $\|V\|$	Average $\|E\|$	Preprocessed data size (GB)
wget	Benign	125	265424	975226	64
	Attack	24	257156	957968	12
shellshock	Benign	125	238338	911153	59
	Attack	25	243658	949887	12

4.1.3 SC 数据集

SC 数据集由 UNICORN [5] 作者收集. 在 SC 数据集中共模拟了两个 APT 攻击场景 wget 和 shellshock. 对于每个场景, 实验持续了 3 天, 在此过程中, 使用 CamFlow (v0.5.0) 作为工具收集溯源图. 数据集 wget 和 shellshock 的统计信息如表 4 所示.

4.2 实现

本文使用 C++ 和 python 实现算法, 其中从溯源图中提取特征序列使用了 UNICORN 的开源代码. 本文使用 PyTorch 搭建了结合局部注意力机制的序列特征提取网络模型. 在参数设置方面, 从溯源图生成特征序列模块使用了和 UNICORN 相同的参数设置. 序列特征提取模块中, 编码器和解码器都使用了都使用了单层 GRU 模型. 除了对聚类数目 K 这一超参数进行讨论的实验之外, 其余实验中聚类数目都选择 7 类. 在实验阶段需要将正常数据划分为两部分, 一部分作为训练集, 一部分和所有攻击数据构成测试集. 本文实验中正常数据的划分采用和对比方法 UNICORN 相同的比例, 在 wget 和 shellshock 数据集上使用 20% 的正常数据做测试, 在 ClearScope 和 CADETS 数据集上使用 10% 的数据做测试, 在 StreamSpot 数据集上对每个正常类别划分 25% 的数据作为测试集. 为了减小训练集和测试集划分对实验结果的影响, 实验采取 5 折交叉验证的方式.

精确度 (precision) 和召回率 (recall) 是分类问题两个重要的评价指标. 这两个指标由 4 个数据 TP (真阳性数目)、FP (假阳性数目)、TN (真阴性数目) 和 FN (假阴性数目) 计算得到. 在本文中, 精度计算公式如下:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{11}$$

精度表示在所有被识别为攻击的数据中, 真正的攻击所占比例. 召回率计算公式如下:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{12}$$

召回率表示被检测出的攻击占所有真实攻击的比例.

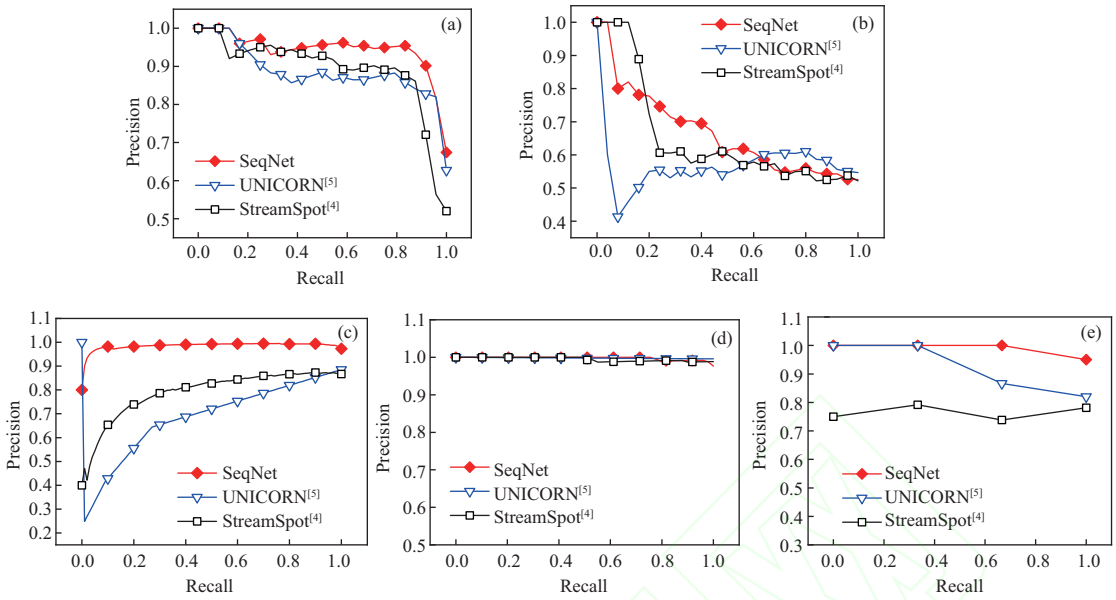

图 5 (网络版彩图) 对比实验结果

Figure 5 (Color online) Experimental results of two methods. (a) wget; (b) shellshock; (c) StreamSpot; (d) ClearScope; (e) CADETS

由于本文方法与 UNICORN 相似, 都是对正常行为进行建模, 所以在测试过程中都需要指定阈值. 但是, 在真实环境中, 无法知道最佳阈值是多少, 因此本文根据不同阈值下精确度和召回率的计算结果, 绘制 PR (precision-recall) 曲线, 并使用 PR 曲线与坐标轴之间的面积作为评价指标, 面积越大, 表示分类效果在不同阈值下整体效果越好.

4.3 对比方法

• UNICORN [5]. UNICORN 首先将溯源图转化为直方图, 使用 histogram2vec 算法将直方图转化为特征向量, 通过对特征向量聚类生成状态节点, 利用状态节点之间的转移描述系统行为.

• StreamSpot [25]. StreamSpot 算法使用溯源图局部子图的相似性定义溯源图之间的相似性, 通过聚类算法将相似的溯源图聚集在一起.

4.4 对比实验结果

为了验证本文提出的方法的效果, 与当前具有代表性的方法 UNICORN [5] 和 StreamSpot [25] 在 5 个公开数据集上进行对比实验. 绘制出的 PR 曲线图如图 5 所示. 在 PR 曲线中, 横坐标是召回率, 纵坐标是相应的精确度. 在召回率相同情况下, 精确度越高, 效果越好. 若不同方法绘制的曲线有交叉, 需要计算 PR 曲线下面积, 面积较大的方法效果更好. 在 wget 数据集上, 本文方法对应面积为 0.9452, 相比于 UNICORN 的 0.8065 提高 17.09%, 相比于 StreamSpot 的 0.8923 提高 5.93%. 在 StreamSpot 数据集上, 本文结果为 0.9856, 相较于 UNICORN 的 0.7730 提高 27.50%, 相较于 StreamSpot 算法的 0.7909 提升 24.61%. 在 CADETS 数据集上, 本文结果为 0.9917, 相较于 UNICORN 的 0.9256 提高 7.14%, 相较于 StreamSpot 算法的 0.7651 提高 20.98%. 在 ClearScope 数据集上, 3 条曲线几乎是重合的, 通过计算面积, UNICORN 对应面积为 0.9979, 本文方法对应面积为 0.9978, StreamSpot 算法对应面积为 0.9949, 本文方法略高于 StreamSpot 算法, 略低于 UNICORN. 在 shellshock 数据集上, 3 种方

法的 PR 曲线相交. 本文方法对应的面积为 0.6634, StreamSpot 算法对应的面积为 0.6526, UNICORN 对应面积为 0.5655, 从整体上看, 本文方法更有效. 3 种算法在 shellshock 数据集上效果都不够好, 是因为该数据集正常行为的随机性过强, 导致提取的正常行为特征与异常行为特征难以区别.

本文提出的方法相较于 UNICORN 算法和 StreamSpot 算法在整体上取得更好的效果, 主要原因是 GRU 模型能够捕捉长序列中的高阶特征. 在真实网络攻击场景中, APT 攻击周期长, 其对应的特征序列也较长, 本文提出的方法能够更加有效地提取长序列的模式, 因此能够更好地适应真实场景中 APT 攻击特点, 实现更好的检测效果. 与 StreamSpot 算法相比, 本文的优势在于将溯源图序列提取成为特征向量序列, 特征向量能够有效地反映出系统的状态, 而特征向量序列包含了系统状态变化的时序特征, 本文方法能够有效地利用这种序列的时序特征. UNICORN 算法虽然也利用了特征序列, 但是其匹配状态机模板的做法, 当异常行为与正常行为的差异较小时, 这种差异会被忽略, 只有当这种差异较大时, 才会标识为异常. 本文使用的 GRU 模型在捕捉序列特征过程中, 融合序列的历史特征和当前特征, 最后计算得到的序列特征融合了整个序列的特征, 能够保留和累计序列中的微小差异, 最终加大正常行为和异常行为在特征空间的距离. 同时, 在模型训练过程中使用局部时序注意力机制, 也使得模型能够更准确地捕捉序列特征.

4.5 参数实验

在本文提出的方法中, 有两个重要参数, 一个是局部注意力网络中局部的范围 R, 另一个是聚类数目 K. 本小节将通过实验讨论这两个参数的取值对结果的影响.

4.5.1 局部注意力范围对实验结果的影响

在解码器中, 为了能够更加快速以及准确地重构出原始特征序列, 本文使用了局部注意力机制, 利用编码器输出的特征序列中长度为 R 的子序列提供额外信息帮助重构输入序列. 为了讨论局部范围 R 对实验结果的影响, 本文在 5 个公开数据集上, 设置局部子序列长度为 $\{1, 3, 5, 7, 9, 11\}$, 分别计算了每个取值对应的 PR 曲线下的面积. 实验结果如图 6 所示. 在 wget 数据集上, R 取 1~3, 效果小幅上升, R 取 3~11, 面积在 0.01 内波动. 在 shellshock 数据集上, R 取 1~5 时, 效果有较大提升, R 取 5~9, 效果较好, R 到 11 时, 效果出现较大下降. 在 StreamSpot 数据集上, R 取 1~7 时, 面积在大约 0.01 范围内波动, 当 R 取 9~11 时, 面积出现小幅下降. 在 ClearScope 数据集上, R 取 1~11, 效果都较为好. 在 CADETS 数据集上, 效果先小幅上升, 在取值 7~11 范围内, 取得较好效果.

从整体上可以看出, 随着 R 值的增大, 效果先呈上升趋势. 这一现象的原因是系统行为连续变化, 序列中的一个特征向量会和其前后的特征向量具有时序上的关联, 随着 R 的增大, 可以使用的有效信息增加, 会使得模型效果变好. 但是在序列中, 两个相隔较远的特征向量, 其关联性就很弱, 所以相隔较远的特征对于当前特征的重构, 无法提供有效信息, 甚至会使模型提取出不正确的特征. 所以随着 R 的继续增大, 模型效果会保持平稳或者出现下降趋势. 从实验结果中可以看出, R 取 5, 7, 9 时都有不错的效果.

4.5.2 聚类数目对实验结果的影响

本小节将讨论聚类数目 K 这一超参数对于实验结果的影响. 本小节在 5 个数据集上, 分别设置 K 的值为 $\{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$, 计算每一个 K 值对应的 PR 曲线面积. 实验结果如图 7 所示. 从图中可以看出, 随着 K 值的增大, 模型效果呈先上升, 后较为平稳的趋势. 由于系统行为本身的复杂性, 描述系统行为的特征向量在特征空间中一般不会紧密地聚集在一起. 当 K 比较小时, 聚类算法会将

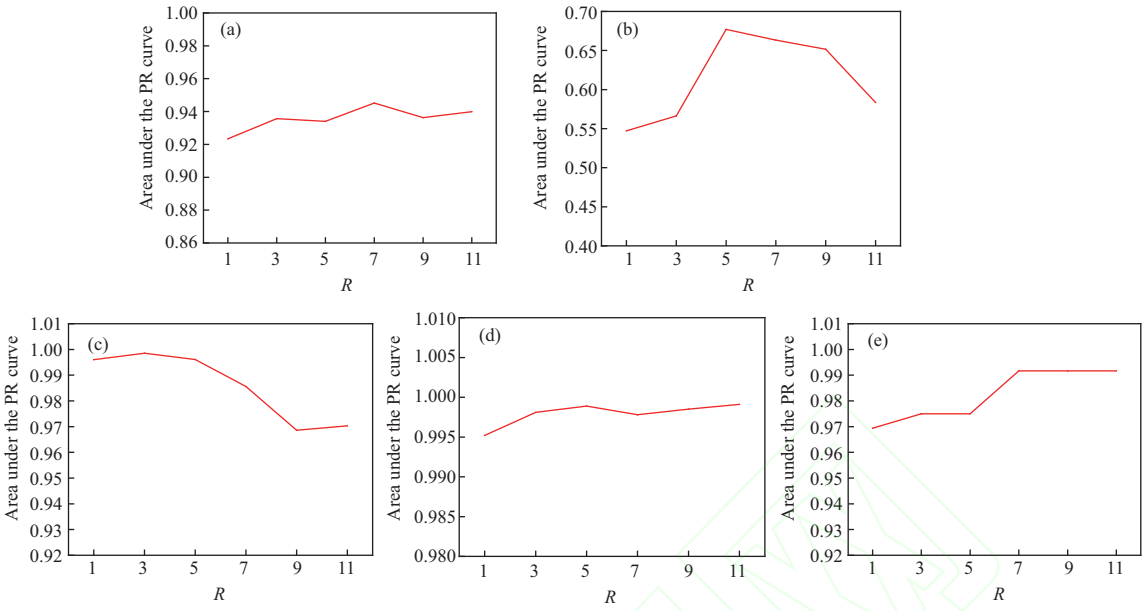

图 6 (网络版彩图) 参数 R 对实验结果的影响

Figure 6 (Color online) Influence of parameter R on experimental results. (a) wget; (b) shellshock; (c) StreamSpot; (d) ClearScope; (e) CADETS

原本距离较远的特征聚到一起, 会导致聚类中心偏移, 增大正常行为的特征到聚类中心的距离, 模型效果变差. 例如在 StreamSpot 数据集上, 正常行为有 5 类, 当 $K = 2$ 时, 面积只有 0.7211, 当 $K = 5$ 时, 增加到 0.9791, 当 $K > 5$ 时, 稳定在 0.98 之上. 当 K 值增大, 在 wget, StreamSpot, ClearScope 和 CADETS 数据集上都表现出比较平稳的趋势, 而在 shellshock 数据集上出现下降趋势. 通过实验表明, 在大多数情况下, 实验结果对 K 值不太敏感, 在较宽的范围内都有较好的效果.

4.6 消融实验

对于序列特征提取模块来说, 解码器部分只存在于模型训练阶段, 测试阶段只保留编码器部分. 本小节将通过消融实验讨论解码器中的局部注意力机制对 SeqNet 的影响. 本小节将从训练过程中的迭代次数以及 PR 曲线面积两个评价指标来对比加入局部注意力机制的解码器和去掉注意力机制的解码器, 实验在 5 个数据集上进行.

表 5 记录了两种模型在训练阶段迭代次数. 从实验结果可以看出, SeqNet 在所有数据集上所经历的迭代次数均小于去掉注意力机制的模型, 其中效果最为明显的是 wget 数据集, 迭代次数减少了 92.86%. 在 shellshock 数据集上, 减少了 88.86%. 在 StreamSpot 数据集上, 减少了 69.44%. 在 ClearScope 数据集上, 减少了 67.16%. 在 CADETS 数据集上, 迭代次数减少了 59.16%. 迭代次数的减少是因为局部注意力机制能够在特征序列重构过程中提供额外的有效信息, 使得模型能够更快的收敛.

表 6 记录了 5 个数据集上两种方法对应的 PR 曲线的面积. 从表中可以看出, 除了在 CADETS 这个数据集上, 二者表现相同外, 在其余 4 个数据集上, 本文方法都要比去掉注意力机制的方法效果更好. 在 wget 数据集上, 效果提升了 17.20%. 在 shellshock 数据集上, 效果提升了 10.79%. 在 StreamSpot 数据集上提升效果最明显, 提升了 27.50%. 在 ClearScope 数据集上提升了 2.42%. 实验结果说明局部

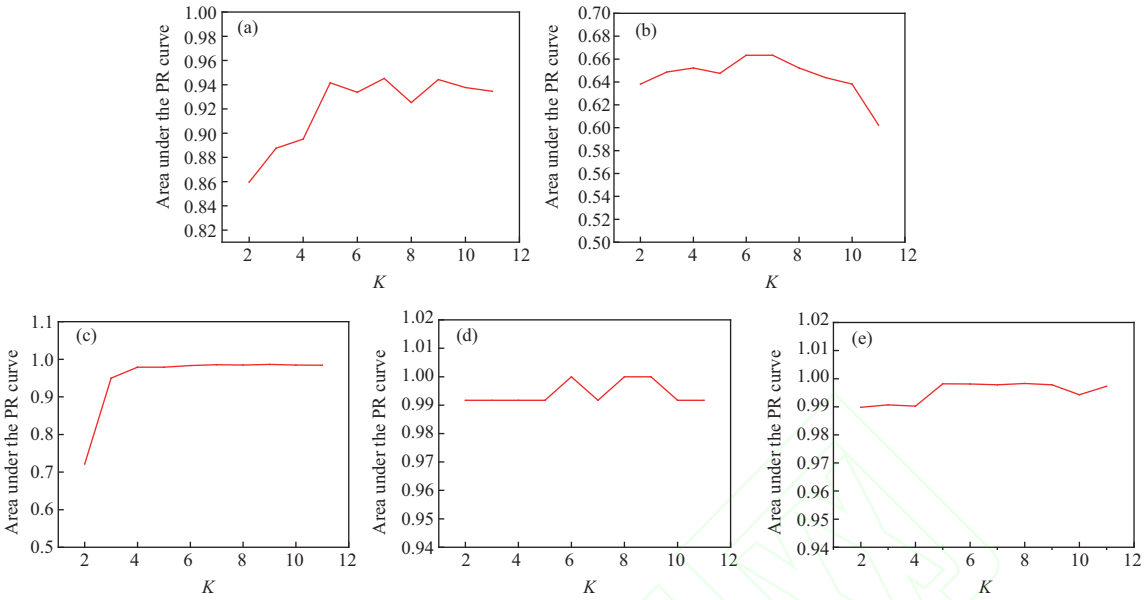

图 7 (网络版彩图) 参数 K 对实验结果的影响

Figure 7 (Color online) Influence of parameter K on experimental results. (a) wget; (b) shellshock; (c) StreamSpot; (d) CADETS; (e) ClearsCope

表 5 迭代次数

Table 5 Iterations

	wget	shellshock	StreamSpot	ClearScope	CADETS
SeqNet	**245**	**253**	**154**	**507**	**205**
SeqNet without attention	3430	2271	504	1544	502

表 6 PR 曲线面积

Table 6 The area of PR curve

	wget	shellshock	StreamSpot	ClearScope	CADETS
SeqNet	**0.9452**	**0.6634**	**0.9856**	**0.9978**	**0.9917**
SeqNet without attention	0.8065	0.5988	0.7730	0.9742	**0.9917**

注意力机制的引入, 能够帮助编码器模型找到更好的参数, 提升模型效果.

4.7 运行时间、显存消耗和 GPU 使用率

本实验主要计算工作在 GPU 上进行, 所以本小节将评估本文所提方法的显存占用和 GPU 使用率. 本实验的硬件环境是单张 GTX 1080Ti, 分别记录程序在 5 个数据集上运行的显存占用和 GPU 平均使用率. 结果如表 7 所示. 从表中可以看出, 在各个数据集上 GPU 平均使用率都在 16% 以下, 显存占用也在 600 MB 附近, 属于合理范围. 本文也评估了检测时间. 由于本文采用流式方法实现 APT 攻击检测, 其中流式更新溯源图部分的性能在 UNICORN 中已有实验证实可以满足实时检测性能, 本实验对检测器部分进行了检测时间对比, 结果如表 8 所示. 检测时间相较于攻击持续时间是可以忽略的, 满足检测的实时性要求.

表 7　显存占用与 GPU 平均使用率

Table 7　Video memory utilization and GPU utilization

	wget	shellshock	StreamSpot	ClearScope	CADETS
Video memory utilization (MB)	599	597	569	601	575
GPU utilization (%)	15.3	15.7	12.7	14.6	10.3

表 8　运行时间评估 (s)

Table 8　Evaluation of running time (s)

	wget	shellshock	StreamSpot	ClearScope	CADETS
SeqNet	10.24	8.45	5.90	6.85	0.73
UNICORN	12.02	10.66	7.55	9.88	1.13
StreamSpot	194.78	449.87	55.19	709.82	12.63

5　不足之处

本节主要讨论本文所提方法存在的不足之处.

本文所采用的正常行为建模的方式, 虽然能够解决真实场景中攻击行为数据难以收集, 以及 APT 攻击中会采用一些未知攻击的问题, 但如果系统产生一些新的行为, 本文方法会产生误报. 例如当系统安装新的软件之后, 新的软件的行为模式与训练数据集有较大差别, 将会导致生成的溯源图序列有较大差别, 最终生成的新的特征向量与聚类中心存在较大距离, 导致被错误识别为攻击行为. 这种情况下需要重新收集数据更新模型.

虽然溯源图包含丰富的系统调用信息, 但是溯源图本身会记录系统所有行为产生的系统调用, 所以攻击行为容易被正常行为掩盖. 虽然本文所提方法能够累计攻击行为与正常行为之前的差异, 增大攻击行为与正常行为之间的距离. 但如果正常行为本身过于复杂, 攻击行为比例过低, 也会导致正常行为与异常行为之间差异不显著. 所以需要研究对溯源图进行剪枝, 删除部分正常行为, 突出攻击行为.

6　结论

本文提出了一种溯源图上的 APT 攻击检测方法. 该方法使用溯源图序列描述系统行为的变化, 并且将溯源图序列转化为特征序列, 之后使用 GRU 模型捕捉序列的长期变化特征. 在模型训练阶段, 使用了基于编码器 – 解码器架构的模型, 并结合局部注意力机制, 通过最小化重构误差的方式训练 GRU 网络模型; 在攻击检测阶段, 直接使用训练好的 GRU 模型提取序列特征. 最后, 在训练数据上利用聚类算法, 建立系统的正常行为模型, 在攻击检测阶段, 将远离聚类中心的序列识别为攻击行为. 为了评价方法的效果, 本文在 5 个公开数据集 wget, shellshock, StreamSpot, ClearScope 和 CADETS 上进行了实验, 并与当前具有代表性的方法 UNICORN 和 StreamSpot 进行了对比, 实验结果验证了本文方法能够更加有效地对长特征序列进行建模, 对真实场景下 APT 攻击检测更加有效.

参考文献

1　Hossain M N, Milajerdi S M, Wang J, et al. SLEUTH: real-time attack scenario reconstruction from COTS audit data. In: Proceedings of the 26th USENIX Security Symposium, Vancouver, 2017. 487–504

2 Milajerdi S M, Eshete B, Gjomemo R, et al. POIROT: aligning attack behavior with kernel audit records for cyber threat hunting. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, London, 2019. 1813–1830

3 Milajerdi S M, Gjomemo R, Eshete B, et al. HOLMES: real-time APT detection through correlation of suspicious information flows. In: Proceedings of IEEE Symposium on Security and Privacy, San Francisco, 2019. 1137–1152

4 Manzoor E, Milajerdi S M, Akoglu L. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In: Proceedings of International Conference on Knowledge Discovery and Data Mining, 2016. 1035–1044

5 Han X Y, Pasquier T F J M, Bates A, et al. UNICORN: runtime provenance-based detector for advanced persistent threats. In: Proceedings of the 27th Annual Network and Distributed System Security Symposium, San Diego, 2020

6 Muniswamy-Reddy K K, Holland D A, Braun U, et al. Provenance-aware storage systems. In: Proceedings of USENIX Annual Technical Conference, Boston, 2006. 43–56

7 Pasquier T F J M, Han X Y, Goldstein M, et al. Practical whole-system provenance capture. In: Proceedings of Symposium on Cloud Computing, Santa Clara, 2017. 405–418

8 Gehani A, Tariq D. SPADE: support for provenance auditing in distributed environments. In: Proceedings of the 13th International Middleware Conference, Montreal, 2012. 101–120

9 Spillane R P, Sears R, Yalamanchili C, et al. Story book: an efficient extensible provenance framework. In: Proceedings of the 1st Workshop on the Theory and Practice of Provenance, San Francisco, 2009

10 Vahdat A, Anderson T E. Transparent result caching. In: Proceedings of USENIX Annual Technical Conference, New Orleans, 1998

11 Lee K H, Zhang X Y, Xu D Y. LogGC: garbage collecting audit log. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, Berlin, 2013. 1005–1016

12 Tang Y T, Li D, Li Z C, et al. Nodemerge: template based efficient data reduction for big-data causality analysis. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, Toronto, 2018. 1324–1337

13 Xu Z, Wu Z Y, Li Z C, et al. High fidelity data reduction for big data security dependency analyses. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, Vienna, 2016. 504–516

14 Hossain M N, Wang J, Sekar R, et al. Dependence-preserving data compaction for scalable forensic analysis. In: Proceedings of the 27th USENIX Security Symposium, Baltimore, 2018. 1723–1740

15 King S T, Chen P M. Backtracking intrusions. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, 2003. 223–236

16 Xie Y L, Feng D, Tan Z P, et al. Unifying intrusion detection and forensic analysis via provenance awareness. Future Gener Comput Syst, 2016, 61: 26–36

17 Xie Y L, Feng D, Hu Y C, et al. Pagoda: a hybrid approach to enable efficient real-time provenance based intrusion detection in big data environments. IEEE Trans Dependable Secure Comput, 2020, 17: 1283–1296

18 Xie Y L, Wu Y F, Feng D, et al. P-Gaussian: provenance-based Gaussian distribution for detecting intrusion behavior variants using high efficient and real time memory databases. IEEE Trans Dependable Secure Comput, 2021, 18: 2658–2674

19 Hassan W U, Guo S J, Li D, et al. Nodoze: combatting threat alert fatigue with automated provenance triage. In: Proceedings of the 26th Annual Network and Distributed System Security Symposium, San Diego, 2019

20 Sun X Y, Dai J, Liu P, et al. Towards probabilistic identification of zero-day attack paths. In: Proceedings of IEEE Conference on Communications and Network Security, Philadelphia, 2016. 64–72

21 Sun X Y, Dai J, Liu P, et al. Using Bayesian networks for probabilistic identification of zero-day attack paths. IEEE Trans Inform Forensic Secur, 2018, 13: 2506–2521

22 Hassan W U, Noureddine M A, Datta P, et al. Omegalog: high-fidelity attack investigation via transparent multi-layer log analysis. In: Proceedings of the 27th Annual Network and Distributed System Security Symposium, San Diego, 2020

23 Yu L, Ma S Q, Zhang Z, et al. ALchemist: fusing application and audit logs for precise attack provenance without instrumentation. In: Proceedings of Symposium on Network and Distributed System Security, 2021

24 Hossain M N, Sheikhi S, Sekar R. Combating dependenceexplosion in forensic analysis using alternative tag propagation semantics. In: Proceedings of IEEE Symposium on Security and Privacy, San Francisco, 2020. 1139–1155

25 Yang D Q, Li B, Rettig L, et al. Histosketch: fast similarity-preserving sketching of streaming histograms with concept

drift. In: Proceedings of IEEE International Conference on Data Mining, New Orleans, 2017. 545–554
26 Luong T, Pham H, Manning C D. Effective approaches to attention-based neural machine translation. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, Lisbon, 2015. 1412–1421

Sequence feature extraction-based APT attack detection method with provenance graphs

Ruozhou LIANG, Yue GAO & Xibin ZHAO*

Beijing National Research Center for Information Science and Technology (*BNRist*)*, Key Laboratory for Information System Security, Ministry of Education* (*KLISS*)*, School of Software, Tsinghua University, Beijing* 100084*, China*

* Corresponding author. E-mail: zxb@tsinghua.edu.cn

Abstract Advanced persistent threat (APT) in real scenes, especially in industrial scenes, is complex and long term, but the current methods cannot effectively extract the long term relationship in the attack. An attack detection method with provenance graphs, called SeqNet, is proposed. SeqNet uses sequence feature extraction to detect APT attacks. In SeqNet, the provenance graph sequence describing the running state of the system is transformed into the feature sequence first, and then the gate recurrent unit (GRU) model is used to extract the feature of the system. The encoder-decoder model with the local attention mechanism is used to train the GRU model. Finally, the K-means clustering method is used to model the normal behavior of the system. In this study, experiments are carried out on five public datasets, such as StreamSpot, wget, shellshock, ClearScope, and CADETS, compared with the state-of-the-art methods. The method used here achieves similar or improved results on all five datasets. Experimental results show that the proposed method can detect real-life APT scenarios.

Keywords APT attack detection, provenance graphs, anomaly-based detection, sequence feature extraction, local attention mechanism