软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2024,35(8):3610−3625 [doi: 10.13328/j.cnki.jos.007118]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于多元数据融合的网络侧告警排序方法 *

王维靖[1,2], 陈俊洁[1], 杨 林[1], 侯德俊[3], 王星凯[4], 吴复迪[4], 张润滋[4], 王 赞[1,2]

[1](天津大学 智能与计算学部, 天津 300350)

[2](天津大学 新媒体与传播学院, 天津 300072)

[3](天津大学 信息与网络中心, 天津 300072)

[4](绿盟科技集团股份有限公司, 北京 100089)

通信作者: 侯德俊, E-mail: hdj@tju.edu.cn

摘 要: 部署在网络节点上的网络安全监控系统每天会生成海量网络侧告警, 导致安全人员面临巨大压力, 并使其对高风险告警不再敏感, 无法及时发现网络攻击行为. 由于网络攻击行为的复杂多变以及网络侧告警信息的局限性, 已有面向IT运维的告警排序/分类方法并不适用于网络侧告警. 因此, 提出了基于多元数据融合的首个网络侧告警排序方法 NAP (network-side alert prioritization). NAP 首先设计了一个基于源 IP 地址与目的 IP 地址的多策略上下文编码器, 用于捕获告警的上下文信息; 其次, NAP 设计了一个基于注意力机制双向 GRU (gated recurrent unit)模型与 ChineseBERT 模型的文本编码器, 从告警报文等文本数据中学习网络侧告警的语义信息; 最后, NAP 构建了排序模型得到告警排序值, 并按其降序将攻击性强的高风险告警排在前面, 从而优化网络侧告警管理流程. 在 3 组绿盟科技网络攻防数据上的实验表明: NAP 能够有效且稳定地排序网络侧告警, 并且显著优于对比方法. 例如: 平均排序指标 $NDCG@k$ ($k\in[1,10]$) (即前1–10个排序结果的归一化折损累计增益)均在0.893 1–0.958 3之间, 比最先进的方法提升64.73%以上. 另外, 通过将NAP应用于天津大学真实的网络侧告警数据, 进一步证实了其实用性.

关键词: 网络安全; 网络侧告警; 排序; 数据融合

中图法分类号: TP311

中文引用格式: 王维靖, 陈俊洁, 杨林, 侯德俊, 王星凯, 吴复迪, 张润滋, 王赞. 基于多元数据融合的网络侧告警排序方法. 软件学报, 2024, 35(8): 3610–3625. http://www.jos.org.cn/1000-9825/7118.htm

英文引用格式: Wang WJ, Chen JJ, Yang L, Hou DJ, Wang XK, Wu FD, Zhang RZ, Wang Z. Network-side Alert Prioritization Method Based on Multivariate Data Fusion. Ruan Jian Xue Bao/Journal of Software, 2024, 35(8): 3610–3625 (in Chinese). http://www.jos.org.cn/ 1000-9825/7118.htm

Network-side Alert Prioritization Method Based on Multivariate Data Fusion

WANG Wei-Jing[1,2], CHEN Jun-Jie[1], YANG Lin[1], HOU De-Jun[3], WANG Xing-Kai[4], WU Fu-Di[4], ZHANG Run-Zi[4], WANG Zan[1,2]

[1](College of Intelligence and Computing, Tianjin University, Tianjin 300350, China)

[2](School of New Media and Communication, Tianjin University, Tianjin 300072, China)

[3](Information and Network Center, Tianjin University, Tianjin 300072, China)

[4](NSFOCUS Technologies Group Co. Ltd., Beijing 100089, China)

Abstract: The network security monitoring systems deployed on network nodes generate a large number of network-side alerts every day,

* 基金项目: 北京市科技新星计划(Z211100002121150)

本文由“系统与网络软件安全”专题特约编辑向剑文教授、陈厅教授、王浩宇教授、罗夏朴教授、杨珉教授推荐.

收稿时间: 2023-09-10; 修改时间: 2023-10-30; 采用时间: 2023-12-15; jos 在线出版时间: 2024-01-05

CNKI 网络首发时间: 2024-04-03

causing the security engineers to face significant pressure to lose sensitivity to high-risk alerts and fail to detect network attacks in time. Due to the complexity and variability of cyber attacks and the limitation of network-side alert information, existing alert prioritization/classification methods for IT operations are unsuitable for network-side alerts. Thus, network-side alert prioritization (NAP), the first network-side alert prioritization method, is proposed based on multivariate data fusion. NAP first designs a multi-strategy context encoder based on source IP address and destination IP address to capture the context information of network-side alerts. And then, NAP designs a text encoder based on the attention-based bidirectional GRU model and the ChineseBERT model to learn the semantic information of network-side alerts from the text data such as alert messages. Finally, NAP builds a ranking model to obtain the alert ranking values and then ranks the high-risk alerts with cyber attack intention in the front according to their descending order to optimize the network-side alert management process. The experiments on three groups of network attack and defense data from NSFOCUS show that NAP can achieve effective and stable prioritization results, and significantly outperforms the compared methods. For example, the average *NDCG@k* ($k\in[1,10]$) (i.e., normalized discounted cumulative gain of the first 1 to 10 ranking results) ranges from 0.893 1 to 0.958 3, and outperforms the state-of-the-art method more than 64.73%. Besides, NAP has been applied to a real-world network-side alert dataset from Tianjin University, further confirming its practicability.

Key words: cyber security; network-side alert; prioritization; data fusion

近年来, 随着全球数字经济的飞速发展, 网络安全事件层出不穷. 2021 年 5 月, 勒索软件 DarkSide 通过网络攻击迫使美国主要燃料管道暂停运营[1]. 因此, 为了保障网络信息安全, 及时发现网络攻击行为, 大量网络安全监控系统被广泛部署在企业的网络节点上, 例如网络入侵检测系统(network intrusion detection system, IDS)[2]、Web 应用防火墙(Web application firewall, WAF)[3]. 这些网络安全监控系统能够实时监测网络流量, 收集海量网络侧告警, 并发送给安全运营中心(security operations center, SOC)进行处理.

由于网络环境复杂, 告警数据量巨大且安全人员人数十分有限, 安全人员们面临着“告警疲劳(alert fatigue)”的问题[4], 即: 安全人员遭受处理海量告警的巨大压力, 对高风险告警不再敏感, 无法及时发现网络攻击行为. Orca Security 公司一份基于 2022 年公有云安全的报告显示[5]: 59%的受访者每天会收到 500 个以上的新告警, 79%的受访者每天有 500 个以上未处理完的告警. 然而, 由于这些告警中存在大量误报与噪声告警[6,7], 实际具有攻击性的高风险告警很少, 安全人员需要花费大量时间对收到的告警进行分析. 正如该报告指出: 有 64%的受访者认为, 真正重要的告警占比低于 10%; 56%的受访者每天花费 20%以上的时间查看告警并确定需要优先处理的告警.

为了尽快找出高风险告警, 研究者提出了基于规则和基于机器学习/深度学习的方法, 这些已有方法主要存在如下两方面不足. (1) 尽管基于规则的方法能在一定时间内适用, 但这类方法并不可靠[7]: 首先, 随着系统的变化、攻击方法的改变[8], 特定的规则会很快失效[9,10]; 另外, 由于不同的安全人员在先验知识与业务理解上有所差异[11,12], 告警的质量并不稳定. (2) 研究者提出了许多基于分类[13,14]和排序[15]的自动化检测方法, 但它们并不适用于网络侧告警. 网络侧告警仅记录 HTTP 行为, 包括 IP 地址、端口号、告警名称(name)、告警载荷(payload)、Web 访问请求体和响应体(q-body 和 r-body), 缺少直接反应系统状态的数据, 例如文献[9,13]所需的告警类型(如应用程序、数据库、内存、网络、硬件等)、文献[14]所需的 CPU 使用率、内存使用率、I/O 吞吐量、设备读取速度、网络延迟等指标数据以及文献[15]所需的响应时间、处理时间、进程数量等指标数据.

正是由于 IT 运维告警提供了丰富的系统状态, 这些 IT 运维工作能够有效判断告警或故障(incident)的严重程度[15], 甚至通过“修复故障时的解决方案”学习更精细的排序方法[9]. 因此, 网络攻击行为的复杂多变以及网络侧告警信息的局限性为网络侧告警的研判工作带来挑战. 此外, 现有的网络安全领域的告警研判工作[16]通常将告警进行二分类, 即判断告警是否有恶意, 并没有从告警的风险级别上进一步细化, 这为安全人员增加工作量.

为了从有限的网络侧告警信息中理解复杂多变的网络行为, 并及时发现具有网络攻击意图的高风险告警, 本文提出了基于多元数据融合的网络侧告警排序方法 NAP (network-side alert prioritization). NAP 使用 3 种输入数据: (1) 上下文数据, 即基于源 IP 地址(s-ip)与目的 IP 地址(d-ip)的上下文告警; (2) 文本数据, 包括告警名称、告警载荷、Web 访问请求体、Web 访问响应体; (3) 离散数据, 即目的端口(d-port). 首先, 本方法基

于 s-ip 和 d-ip 设计了多策略上下文编码器, 用于生成告警频数矩阵, 从而获取丰富的告警上下文信息; 其次, 为了从有限的告警数据中挖掘出更多的信息, NAP 设计了基于双向 GRU(gated recurrent unit)模型和 ChineseBERT 模型的文本编码器, 从告警载荷、Web 访问请求体、Web 访问响应体、告警名称文本数据中学习网络侧告警的语义信息, 从而进一步理解复杂多变的网络活动; 最后, NAP 构建了排序模型, 通过概率分布计算排序值, 并按照排序值的降序将攻击性强的告警排在前面, 从而优化网络侧告警管理流程.

为了评估本方法的有效性, 本文在 3 组绿盟科技的网络攻防数据上进行实验. 实验结果表明: NAP 能够有效地对网络侧告警进行排序, 并且显著优于其他对比方法. 例如: NAP 在 3 组实验对象上的平均排序指标 *NDCG@k* ($k\in[1,10]$) (即前 1–10 个排序结果的归一化折损累计增益)均在 0.893 1–0.958 3 之间, 而最佳的对比方法仅达到 0.546 6–0.571 7. 实验还证明了 NAP 的主要组件(即上下文编码器的多策略编码机制、文本编码器中的双向 GRU 模型和注意力机制以及多层感知机)以及每类告警数据(即文本数据、离散数据和上下文数据)对排序结果的贡献. 另外, 本文对 NAP 主要参数(即排序间隔时间、上下文编码器活动窗口的大小、GRU 模型隐藏状态的大小)的影响以及 NAP 的排序效率进行了讨论. 最后, 通过真实的网络侧告警数据进行实践评估.

主要贡献可归纳如下:

(1) 本文提出了基于多元数据融合的首个网络侧告警的排序方法 NAP, 该方法能及时发现带有攻击意图的网络侧告警, 有效地优化告警管理流程, 提高安全人员的工作效率;

(2) 本方法充分利用告警的源 IP 地址、目的 IP 地址、目的端口、告警载荷、Web 访问请求体、Web 访问响应体、告警名称等信息, 设计了多策略上下文告警编码器和基于注意力机制双向 GRU 模型和 ChineseBERT 模型的文本编码器, 它们分别学习网络侧告警的上下文信息和语义信息, 进而理解当前复杂多变的网络状态与网络行为;

(3) 本方法在绿盟科技的 3 组网络攻防数据集上进行实验, 实验结果表明: 本方法在 3 组实验对象上的平均排序指标 *NDCG@k* ($k\in[1,10]$)均在 0.893 1–0.958 3 之间, 比最先进的方法提升 64.73%以上. 另外, 通过将 NAP 应用于天津大学真实的网络侧告警数据, 本文进一步证实该方法的实用性.

本文第 1 节介绍网络侧告警的背景以及告警管理的相关工作. 第 2 节介绍基于多元数据融合的网络侧告警排序方法模型 NAP. 第 3 节通过 5 个研究问题分别验证 NAP 的有效性、NAP 的 4 个主要组件对排序的贡献、NAP 的 3 类输入数据对排序的贡献、NAP 的主要参数对排序的影响以及 NAP 的排序效率. 第 4 节通过将本方法应用于真实数据, 评估其实用性. 第 5 节为有效性威胁. 第 6 节为全文工作总结.

1 网络侧告警的背景与告警管理的相关工作

1.1 网络侧告警

网络侧告警来源于部署在企业网络节点上的网络安全监控系统, 如网络入侵检测系统(NIDS)和 Web 应用防火墙. 其中: 网络入侵检测系统部署在网络中, 用于监控网络流量, 能够识别恶意行为(如有针对性的端口扫描、路径穿越等)并产生告警[2]; Web 应用防火墙部署在应用层, 能够监控应用程序与互联网之间的 HTTP 行为, 阻止攻击者利用应用程序的已知漏洞(如 SQL 注入、文件包含漏洞、xss 跨站脚本等)进行网络攻击, 并发送告警[3].

一条网络侧告警包含时间戳(timestamp)、源 IP 地址、目的 IP 地址、目的端口、告警名称、告警载荷、Web 访问请求体、Web 访问响应体等信息. 表 1 展示一条网络侧告警示例. 具体而言, 告警名称是一段极其简短的文本描述, 因此被广泛应用于网络安全的工作, 如文献[16]利用告警名称序列识别出需要进一步人工研判的告警, 文献[17]利用告警名称序列探寻网络攻击的演化. 告警名称是安全人员为新的漏洞或攻击制定规则时设置, 在本文数据集中, 告警名称由中文编写. 告警载荷、Web 访问请求体以及 Web 访问响应体包含网络传输过程中的诸多细节, 因此安全人员可通过这些告警报文数据调查网络活动的意图, 研判该告警是否具有攻击性. 例如: 由于告警载荷的重要性, 文献[18–21]等工作将告警载荷的信息应用于应用层异常的检测.

1.2 告警管理

在网络安全[4,22]、IT 运维[23−25]等领域中, 告警是重要的分析依据. 告警管理通常包括告警生成、告警处理等过程. 首先, 当网络遭受攻击或软件系统出现异常时, 告警应被及时触发[26,27]. 例如: Zhao 等人[27]提出了 SCWarn 方法, 旨在识别大型在线服务系统软件的不良变更, 并生成可解释的告警. 另外, 为应对海量告警, 学界提出大量告警处理方案, 包括告警聚合[24,28]、告警摘要[23]、告警分级/排序[9,16], 旨在降低安全人员的调查成本, 提高严重告警的处理效率. 例如: Landauer 等人[28]提出一种面向入侵检测系统告警的聚合方法, 该方法首先按照告警发生时间进行分组, 再根据告警序列的相似性对告警组进行聚类; Chen 等人[23]提出一种面向在线服务系统的告警摘要方法, 该方法分别利用告警日志和告警序列学习告警的语义信息和行为信息, 并通过深度学习模型确定告警之间的相关性, 生成告警摘要.

与上述方法不同, 本文提出一种基于多元数据融合的网络侧告警排序方法. 本方法通过上下文编码器和文本编码器从有限的网络侧告警信息中理解复杂多变的网络行为, 旨在帮助安全人员及时发现具有网络攻击意图的高风险告警, 从而优化网络侧告警管理流程. 相比于其他工作, 本方法能够通过多种策略捕获告警的上下文信息, 并从多元数据学习告警的语义信息, 从而理解复杂多变的网络状态与网络行为, 并在一定程度上应对新告警.

表 1 网络侧告警示例

项目	示例
时间戳(timestamp)	2021-03-24 20:50:03
源 IP 地址(s-ip)	10.5.xxx.xxx
目的 IP 地址(d-ip)	10.67.xxx.xxx
目的端口(d-port)	8020
告警名称(name)	Apache Shiro 1.5.1 身份验证绕过漏洞(CVE-2020-1957)
告警载荷 (payload)	POST /tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp HTTP/1.1 Host: 10.67.***.***:*** User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/***.***.***.*** Safari/537.36 Content-Length: 28 Content-Type: application/x-www-form-urlencoded Accept-Encoding: gzip fileName=/etc/f5-release
Web 访问请求体 (q-body)	POST /tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp HTTP/1.1 Host: 10.67.***.***:*** User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/***.***.***.*** Safari/537.36 Content-Length: 28 Content-Type: application/x-www-form-urlencoded Accept-Encoding: gzip fileName=/etc/f5-release
Web 访问响应体 (r-body)	HTTP/1.1 405 Method Not Allowed Date: Wed, 24 Mar 2021 20:51:26 GMT Content-Type: text/plain; charset=utf-8 Content-Security-Policy: script-src ‘self’ ‘unsafe-inline’ ‘unsafe-eval’ http://***.***.***.***:*** http://localhost:9881 https://hub.jetbrains.com www.google-analytics.com www.google.com www.googletagmanager.com mc.yandex.ru http://***.***.***.***:***/hub; frame-ancestors ‘self’ http://***.***.***.***:***/hub Content-Encoding: gzip Content-Length: 60 Failure: 405 Method Not Allowed

2 方法结构

为了从海量告警中找出高风险告警, 本文提出一种基于多元数据融合的网络侧告警排序方法 NAP. 本方法考虑 3 种输入数据——上下文数据、文本数据和离散数据.

(1) 上下文数据: 为了获取告警的上下文信息, 我们需要从本告警发生前后相邻时间段内的告警中提取上下文特征;

(2) 文本数据: 由于网络侧告警缺少直接反映系统状态的数据(如 CPU 使用率、内存使用率、I/O 吞吐量等), 因此我们需要充分利用告警名称、告警载荷、Web 访问请求体和 Web 访问响应体中的文本信息, 进而理解复杂多变的网络活动;

(3) 离散数据: 由于不同的端口承担不同的任务, 某些端口更容易被攻击者利用, 因此本方法将目的端口列为输入数据.

本方法的整体框架如图 1 所示: 首先, 为了有效地从多元数据中提取特征, 本方法设计了一个基于源 IP 地址与目的 IP 地址的多策略上下文告警编码器和一个基于注意力机制的双向 GRU 模型和 ChineseBERT 模型的文本编码器. 上下文编码器能够从告警发生前后相邻时间段内告警的频数中学习上下文特征, 进而理解网络环境. 文本编码器共包含两个语言模型, 其中, 基于注意力机制的双向 GRU 模型和已训练好的 ChineseBERT 模型能分别从丰富的英文报文(payload, q-body 和 r~body)和简短的中文告警名称(name)中捕获语义信息. 最后, 本方法利用分类器产生的概率分布构建一个排序模型, 使安全人员能够根据告警排序结果及时处理高风险告警, 提高工作效率.

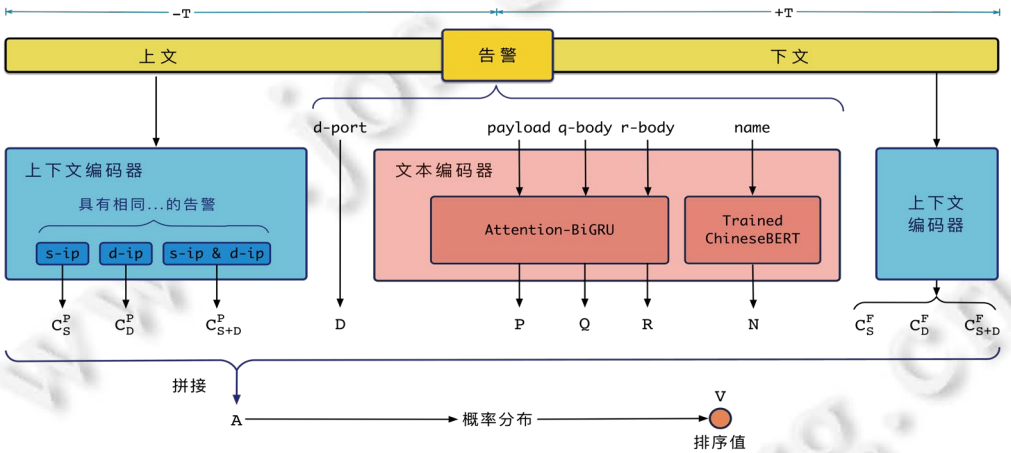


图 1 NAP 的整体框架

2.1 上下文特征表示

2.1.1 上下文告警的收集

为了获取告警的上下文信息, 上下文编码器从告警发生的相邻时间段内收集告警. 具体而言, 上下文编码器采用 2 个大小为ΔT 的活动窗口, 分别获取当前告警发生前 T min 和发生后 T min 的告警序列, 即上文(preceding)告警序列与下文(following)告警序列.

然而, 相邻时间段内的告警并不一定与当前告警有关. 直觉来看, 同一 IP 地址通过发送与接收所产生的告警能从一定程度上反映出该 IP 地址的网络行为. 为了提取出更精确的上下文特征, 本方法设计了 3 种策略: 基于源 IP 地址的上下文告警收集策略、基于目的 IP 地址的上下文告警收集策略以及基于 s-ip 与 d-ip 的上下文告警收集策略. 具体而言, 上下文编码器从上文告警序列与下文告警序列中分别收集: (1) 与当前告警具有相同 s-ip 的告警(即基于 s-ip 的上下文告警收集策略); (2) 与当前告警具有相同 d-ip 的告警(即基于 d-ip 的上下文告警收集策略); (3) 与当前告警同时具有相同 s-ip 与 d-ip 的告警(即基于 s-ip 与 d-ip 的上下文告警收集策略). 为便于理解, 我们将借助图 2 的示例, 展示每一种上下文告警收集策略.

2.1.2 上下文数据的特征表示

直觉上, 正常业务产生的上下文告警序列与网络遭受攻击时有所不同. 例如: 当攻击发生时, 多种罕见告警的数量可能在短时间内急剧增加. 因此, 告警名称的类别以及每种告警名称的数量能够反映当前告警所面临的网络环境信息. 所以, NAP 基于多种收集策略设计出上下文告警的多策略编码机制. 具体而言, 上下文编码器将每种策略收集到的告警序列编码为 n 维向量, 表示为 $C=[c_1,c_2,\ldots,c_i,\ldots,c_n]$, 其中, n 表示已知告警名称

的总数, c_i 表示第 i 种告警在上下文序列中出现的次数. 为了区分不同收集策略与发生前后的上下文特征表示, 我们把通过上述 3 种上下文告警收集策略获得的上文特征分别定义为 C_S^P, C_D^P 以及 $C_{S\&D}^P$. 同理, 将相应的下文特征分别定义为 C_S^F, C_D^F 以及 $C_{S\&D}^F$. 在图 2 的示例中(假设 n=5), 通过统计每种告警名称($N1$ 到 $N5$)出现的数量, 使用“基于 s-ip 的上下文告警收集策略”收集到的上文特征可以表示为 5 维向量 $C_S^P=[1,0,4,0,0]$.

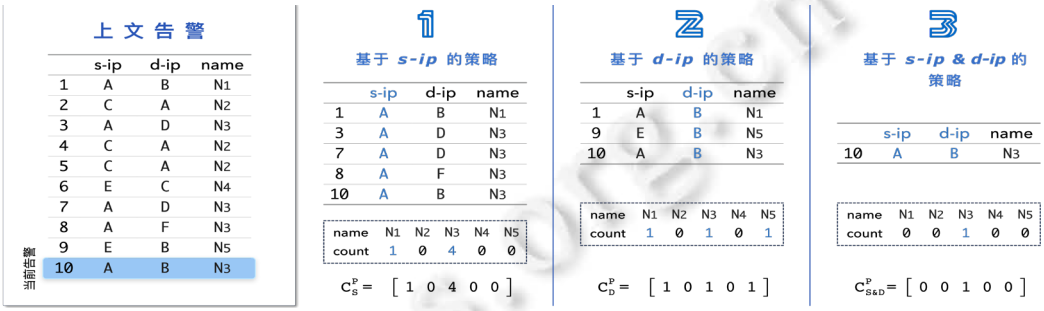


图 2 NAP 上下文编码器的 3 种策略

2.2 文本数据的语义表示

为了深入理解复杂多变的网络行为, 本方法将告警名称、告警载荷、Web 访问请求体和 Web 访问响应体均视为一个句子, 通过文本编码器充分学习网络侧告警的语义信息, 并弥补上下文编码器无法捕获新告警的缺陷.

2.2.1 告警报文的文本语义表示

对于报文信息, 本方法通过 GRU[29]捕获语义信息. 此时, 文本编码器的输入是一段英文报文, 即告警载荷、Web 访问请求体和 Web 访问响应体. 首先, 通信过程中, 报文为二进制的字节流, 需要解码. 其次, NAP 将报文中标点符号、分隔符、操作符等非字母部分去掉, 把组合词(如 AppsLocalLogin, getServletPath)拆分成单独的词语, 并去掉停用词. 接下来, NAP 通过 GRU[29]神经网络对报文信息进行编码. 诸多工作表明, GRU 能有效地提取出文本中的信息[30−32]. 具体而言, NAP 使用的词向量为 GloVe 模型[33]提前训练好的词向量, 其语料库通过爬网(common crawl)收集, 因此可以涵盖大部分常见的英文单词, 应对训练集中未出现的新词. 随后, NAP 将一个句子中的词向量序列定义为 $X=[x_1,x_2,\ldots,x_i,\ldots,x_n]$, 其中: x_i 表示门控循环单元的第 i 个时间步, 即句中第 i 个单词. 根据前一步(第 i−1 步)的隐藏状态 h_{i-1}, 可计算出更新门控 z_i 与重置门控 r_i, 如公式(1)所示:

$$\begin{cases} z_t=\sigma(w_z\cdot[h_{i-1},x_i]) \\ r_t=\sigma(w_r\cdot[h_{i-1},x_i]) \end{cases} \tag{1}$$

其中, σ表示 Sigmoid 函数, w_z 和 w_r 分别表示更新门控与重置门控中的网络参数. 接下来, 公式(2)结合前一个时间步的隐藏状态 h_{i-1}, 计算出当前时间步的隐藏状态 $\tilde{h}_i$:

$$\begin{cases} \tilde{h}_i=\tanh(w_h\cdot[r_i\odot h_{i-1},x_i]) \\ h_i=z_i\odot h_{i-1}+(1-z_i)\odot\tilde{h}_i \end{cases} \tag{2}$$

其中, w_h 表示此时的网络参数, $\odot$表示逐元素乘法运算符. 为了捕获整句的语义信息, NAP 通过双向 GRU 模型学习输入序列. 其中, 前向 GRU 根据前一个隐藏状态 $\vec{h}_{i-1}$ 计算当前的隐藏状态 $\vec{h}_i$, 而后向 GRU 通过前一个隐藏状态 $\overleftarrow{h}_{i-1}$ 来计算当前的隐藏状态 $\overleftarrow{h}_i$. 最后, NAP 将两个隐藏状态连接起来, 得到句子的隐藏状态 H_i, 即 $H_i=[\vec{h}_i;\overleftarrow{h}_i]$. NAP 合并从上下文中学到的信息, 从而提高对整句话的编码效果.

在报文中, 并非所有单词都与告警的风险级别相关, 而且报文中可能存在由解码失败导致的乱码, 因此, NAP 采用注意力机制识别告警报文中重要的单词. 具体而言, NAP 为句中每个单词隐藏状态$[H_1,\ldots,H_n]$设置相应的权重, 最终得到基于注意力机制的句向量 S, 如公式(3)所示:

$$S=\sum_{i}^{n}\alpha_i\times H_i \tag{3}$$

其中, α_i 是 H_i 学习到的权重, 由 Softmax 函数计算. 为区分不同数据的语义表示, 本文分别用 P, Q 和 R 作为告警载荷、Web 访问请求体和 Web 访问响应体的语义表示.

2.2.2 中文告警名称的文本语义表示

作为描述告警的简介, 告警名称的内容简明扼要, 噪声信息较少, 且符合自然语言规范. 因此, 告警名称可以通过训练好的语言模型(pre-trained language models, PLM)进行编码. 其中, 最具代表性的 PLM 是基于变换器的双向编码器表示(bidirectional encoder representations from transformers, BERT)模型[34]. BERT 模型通过多头注意力机制有效捕获句子全局特征, 因此被广泛应用于众多领域[35−37]. 由于本文数据集的告警名称由中文编写, NAP 通过 ChineseBERT[38]捕获中文告警名称的语义表示 N.

2.3 离散数据的特征表示

对于离散数据目的端口, NAP 将其编码为固定维度的向量, 在训练过程中, 将向量作为模型参数不断优化. 本文将 d-port 表示为特征向量 $D=[d_1,d_2,\ldots,d_k]$, 其中, k 表示 d-port 向量预定义的维度.

2.4 模型构建

经过上述编码过程, NAP 得到以下 11 种向量表示.

(1) 上下文数据: 上文告警的 3 种策略的特征表示(即 C_S^P, C_D^P 与 $C_{S\&D}^P$)以及下文告警的 3 种策略的特征表示(即 C_S^F, C_D^F 与 $C_{S\&D}^F$);

(2) 文本数据: 告警载荷的向量表示 P、Web 访问请求体的向量表示 Q、Web 访问响应体的向量表示 R 以及告警名称的向量表示 N;

(3) 离散数据: 目的端口的向量表示 D. 为了得到告警的表示 A, NAP 将上述特征表示连接起来,如公式(4)所示, 其中, ||表示连接运算符:

$$A=C_S^P\|C_D^P\|C_{S\&D}^P\|D\|P\|Q\|R\|N\|C_S^F\|C_D^F\|C_{S\&D}^F \tag{4}$$

为了得到告警的排序依据, NAP 利用多层感知机(multi layer perception, MLP)构建预测模型, 并在最后接入 Softmax 层, 将 MLP 的输出向量转化为每一类的概率分布(probability distribution). 例如: 当告警的风险程度分为 3 类时, 输出的概率分布依照风险级别从低到高分别为$[p_1,p_2,p_3]$, 其中, p_1 是预测为无风险告警的概率, p_2 是预测为低风险告警的概率, p_3 是预测为高风险告警的概率. 在模型训练阶段, NAP 通过交叉熵损失函数(cross entropy loss)计算损失值, 并利用梯度下降使损失值最小化. 在反向传播的过程中, NAP 的预测结果能够不断优化.

另外, 在实际网络环境中, 无风险告警的占比很大, 因此每个类别的数据并不均衡(imbalance). 在训练时, 模型会偏向于预测成样本数量较大的类别(即无风险告警). 为了应对该问题, NAP 对训练集中的无风险告警进行采样(本文的采样比例为 30%), 并在模型训练时, 依据各个类的样本数量设置类别权重 w_k^{class}, 如公式(5)所示:

$$w_k^{class}=\frac{n_{all}}{n_c\times n_k} \tag{5}$$

其中, n_{all}表示训练集中所有样本的数量, n_c表示告警风险级别的数量, n_k表示训练集中第 k 类($k\geqslant 1$)风险级别告警的数量. 此时, 数量少的高风险类别会被赋予较大权重, 从而缓解数据集不均衡的问题.

最后, 为了计算告警的排序值 V, NAP 为每一个类按照告警的风险程度分别赋予不同的权重, 如公式(6)所示:

$$V=\sum_{k=1}^{n_c}p_k\times w_k^{rank} \tag{6}$$

其中, p_k 表示第 k 类($k\geqslant 1$)的概率分布, k 值越小, 则风险级别越低; k 值越大, 则攻击意图越显著. w_k^{rank} 表示对

第 k 类赋予的权重值, 可取值为 $w_k^{rank}=k-1(k\geqslant 1)$. 因此, 告警被预测的风险级别越高, 其排序值 V 越大. NAP 将所有告警按照排序值 V 的降序进行排序, 通过优先处理排在前面的高风险告警, 安全人员能够及时发现有攻击意图的网络行为, 进而优化对网络侧告警的管理.

在实际应用中, 安全运营中心持续收到大量告警, NAP 会每隔固定时间对新生成的网络侧告警进行一次排序, 并计算相应的评价指标.

3 实 验

本节主要研究以下 5 个研究问题.

- RQ1: NAP 在网络侧告警上的排序效果如何?
- RQ2: NAP 的每个主要组件是否对排序结果做出贡献?
- RQ3: NAP 的每种输入数据是否对排序结果做出贡献?
- RQ4: NAP 的主要参数对排序结果产生怎样的影响?
- RQ5: NAP 在排序效率上表现如何?

为讨论以上 5 个研究问题, 本文在第 3.1 节–第 3.3 节分别介绍数据集、评价指标和实验设置, 并在第 3.4 节–第 3.6 节分别设计对比实验、消融实验以及参数实验, 最后在第 3.7 节展示排序时长.

3.1 数据集

本文使用绿盟科技构建的网络攻防数据集, 该数据的网络侧告警来源于绿盟智能安全运营管理平台 (NSFOCUS intelligent security operation platform), 告警包括命令注入、路径穿越、网络爬虫抓取网页信息、未授权访问、远程代码执行漏洞、XSS 跨站脚本等. 本节从该数据集的有效性与多样性做具体介绍.

- 有效性. 由于安全类工具存在大量误报[7], 因此为保证实验的有效性, 这 3 组实验数据均由绿盟科技的专家人工判定, 并结合攻击队伍的复盘, 得到告警风险级别的准确标注;
- 多样性. 这 3 组实验对象中, 每组包含 7 h 的网络侧告警数据. 表 2 展示了数据集的详细信息, 包括 3 组实验对象所涉及的源 IP 地址种类的数量、目的 IP 地址种类的数量、目的端口种类的数量、告警名称种类的数量、非空告警载荷的占比、非空 Web 访问请求体的占比、非空 Web 访问响应体的占比以及各种风险级别的占比, 其中: 无风险告警表示正常业务触发的误报; 低风险告警包括不会造成实际危害的常规攻击; 而高风险告警需要立即人工排查, 以确认攻击者意图.

表 2 3 组实验对象基本信息

项目	S1	S2	S3	Avg.
# s-ip 的种类	4 379	4 145	12 605	7 043.00
# d-ip 的种类	4 700	4 227	4 520	4 482.33
# d-port 的种类	542	976	4 102	1 873.33
# name 的种类	348	427	381	385.33
非空 payload (%)	16.51	49.63	32.68	32.94
非空 q-body (%)	7.28	23.51	15.04	15.28
非空 r-body (%)	6.92	21.09	13.87	13.96
无风险(%)	94.00	79.61	82.03	85.21
低风险(%)	3.72	20.23	17.93	13.96
高风险(%)	2.28	0.17	0.04	0.83

从表 2 可知: 该数据类型丰富, 3 组实验对象平均包含 385.33 种告警名称、7 043.00 种源 IP 地址、4 482.33 种目的 IP 地址以及 1 873.33 种目的端口, 并且在 3 组实验对象上差别很大. 例如, S3 涉及到的 s-ip 的种类(12 605)约为 S2(4 145)的 3 倍. 另外, 在报文的非空占比与告警风险级别的占比方面, 3 组实验对象具有较大差异. 例如: S2 中低风险告警的占比(20.23%)约为 S1(3.72%)的 5 倍, S2 中非空告警载荷的占比(49.63%)约为 S1(16.51%)的 3 倍. 此外, 平均而言, 32.94%的网络侧告警具有丰富的 payload 文本信息, 但具有 q-body 和 r-body 的告警占比较少, 平均占比分别为 15.28%和 13.96%. 对于每组实验对象,本文将前 6 h 的告警在对无风

险告警采样(如第2.4节所述)的基础上, 按照3:1的比例划分为训练集和验证集, 将最后1 h的告警作为测试集.

3.2 评价指标

本文任务是对多种风险等级的网络侧告警进行排序, 因此本文继现有工作[39−42], 使用归一化折损累计增益(normalized discounted cumulative gain, NDCG)作为指标. NDCG[43,44]是衡量排序系统有效性的指标, 该指标将告警在排序序列中的位置纳入衡量标准, 由公式(7)得出:

$$NDCG@k = \frac{DCG@k}{IDCG@k} \tag{7}$$

其中, $DCG@k$ 表示前 k 个排序结果的折损累积增益, 由公式(8)可得:

$$DCG@k = \sum_{i=1}^{k} \frac{gain_i}{\log_2(i+1)} \tag{8}$$

该式中, $gain_i$ 表示第 i 个告警实际的风险级别: 无风险告警的 $gain$ 值为 0; 随着风险级别的上升, 告警的 $gain$ 值越高. 另外, $IDCG@k$ 是理想排序结果下的折损累积增益. $NDCG@k \in [1,10]$, $NDCG@k$ 越大, 表明排序结果越好.

在实际生产实践中, 安全人员会按照固定的时间间隔调查收到的告警. 为满足生产需要, 本方法会对每个排序时间间隔收到的告警流进行排序, 因此继现有工作[9], 对于一个固定的时间间隔, 本文将所有指标的平均值作为最终指标. 本文实验的排序时间间隔设为 15 min. 例如, 对于时间戳为 8:00–9:00 的测试告警数据, 评价指标 $NDCG@1_{8:00\text{-}9:00}$ 可由公式(9)得出:

$$NDCG@1_{8:00\text{-}9:00} = \frac{NDCG@1_{8:00\text{-}8:15} + NDCG@1_{8:15\text{-}8:30} + NDCG@1_{8:30\text{-}8:45} + NDCG@1_{8:45\text{-}9:00}}{4} \tag{9}$$

3.3 实现与配置

本文基于 Python 3.9 和 Pytorch 1.13 实现 NAP. 通过网格搜索, 本文设置排序间隔时间为 15 min, 上下文编码器活动窗口的大小ΔT为 5 min, GRU 模型隐藏状态的大小为 100, 训练迭代次数为 10, GloVe 模型提前训练好的词向量维度为 300. 在实现对比方法时, 本文尽可能使用相应的开源代码, 并按照其论文描述设置参数. 本文在 Ubuntu 20.04.6 LTS 上进行所有实验, 该机器配备 Intel(R) Xeon(R) Gold 5218R CPU@2.10 GHz、64 位操作系统以及 NVIDIA Tesla A100 40 G GPU 显卡.

3.4 对比实验

3.4.1 对比方法

本节实验的目的是研究 RQ1, 即 NAP 的有效性. 由于本文为首个排序网络侧告警的工作没有可以直接比较的方法, 因此本节实验将网络侧告警的分类工作[16]以及 IT 运维告警的排序工作[9]尽可能地应用于网络侧告警的排序任务.

- AlertRank: Zhao 等人[9]提出一种面向 IT 运维告警的排序方法, 该方法首先从告警的文本描述中提取主题(topic)特征和 IDF 特征, 并收集其他特征(如告警出现频率、上文告警的数量、告警发生时段、与前一个告警之间的时间间隔等), 最后利用 XGBoost 进行排序.
- DeepCASE: Van 等人[16]提出一种基于注意力机制的二分类方法, 该方法仅考虑告警名称所组成的序列. 该方法先从上文告警序列中捕获当前告警与上文告警之间的相关性, 并将向量表示进行聚类. 当出现待排序的告警时, 将其向量表示与已知集群进行比较, 从而实现按照风险级别的自动分类. 为适配本文的告警排序场景, 本文按照风险级别的预测结果得到排序结果.

3.4.2 对比结果

表 3 为 NAP 与其他对比方法(AlertRank 与 DeepCASE)在 3 组实验对象(S1−S3)上的排序指标 $NDCG@k$ ($k \in [1,10]$), 每个指标的粗体值表示所有方法中最好的结果. 由表 3 可知: NAP 达到有效且稳定的排序结果, 在 3 组实验对象上, NAP 均达到最优的排序指标 $NDCG@k$, 范围在 0.840 8−1.000 0 之间, 且 $NDCG@1$ 与

NDCG@10 的最大差异仅为 0.0815(S3). 与 AlertRank 和 DeepCASE 相比, NAP 在 10 个 *NDCG@k* (k∈[1,10]) 指标上的平均提升率分别为 64.73%和 1 260.87%. 这是因为 NAP 由多元数据驱动,通过基于源 IP 地址与目的 IP 地址的多策略上下文告警编码器捕捉告警上下文信息, 并利用基于注意力机制双向 GRU 模型和 ChineseBERT 模型的文本编码器学习告警报文等文本数据, 因此 NAP 能够充分理解当前复杂多变的网络环境和网络行为. 对于 AlertRank, 因为网络侧告警源于攻击者的攻击行为, 更具随机性, 而 AlertRank 提取的特征(如告警出现频率、告警发生时段等)并不能充分反映网络状态, 而且基于历史数据的统计性文本特征无法应对新告警, 因此该方法难以充分捕获到网络侧告警的信息. 另外, DeepCASE 并未学习告警中的文本信息, 且仅通过少量上文告警序列学习告警的特征表示, 因此难以在较短的排序间隔时间(15 min)内有效识别出高风险告警.

表 3 NAP 及对比方法在 3 组实验对象上的排序结果

实验对象	方法	*NDCG@k*									
		1	2	3	4	5	6	7	8	9	10
S1	NAP	**0.875 0**	**0.875 0**	**0.874 9**	**0.874 9**	**0.874 9**	**0.874 9**	**0.863 5**	**0.854 4**	**0.847 0**	**0.840 8**
	AlertRank	0.791 7	0.791 7	0.762 3	0.718 2	0.709 3	0.702 9	0.675 1	0.693 1	0.707 7	0.710 4
	DeepCASE	0.020 5	0.020 5	0.020 5	0.020 5	0.020 5	0.020 5	0.020 5	0.020 5	0.020 5	0.020 5
S2	NAP	**1.000 0**	**1.000 0**	**0.970 7**	**0.954 6**	**0.944 1**	**0.936 7**	**0.931 0**	**0.926 6**	**0.922 9**	**0.919 9**
	AlertRank	0.500 0	0.500 0	0.500 0	0.500 0	0.500 0	0.500 0	0.499 0	0.498 3	0.497 6	0.497 1
	DeepCASE	0.158 6	0.158 6	0.158 6	0.158 6	0.158 6	0.158 6	0.158 6	0.158 6	0.158 6	0.158 6
S3	NAP	**1.000 0**	**1.000 0**	**1.000 0**	**0.958 0**	**0.930 7**	**0.911 2**	**0.907 9**	**0.922 1**	**0.915 5**	**0.918 5**
	AlertRank	0.375 0	0.423 4	0.441 3	0.451 2	0.457 6	0.462 2	0.465 6	0.473 6	0.480 4	0.486 3
	DeepCASE	0.023 9	0.023 9	0.023 9	0.023 9	0.023 9	0.023 9	0.023 9	0.023 9	0.023 9	0.023 9
Avg.	NAP	**0.958 3**	**0.958 3**	**0.948 5**	**0.929 2**	**0.916 6**	**0.907 6**	**0.900 8**	**0.901 0**	**0.895 1**	**0.893 1**
	AlertRank	0.555 6	0.571 7	0.567 9	0.556 5	0.555 6	0.555 0	0.546 6	0.555 0	0.561 9	0.564 6
	DeepCASE	0.067 7	0.067 7	0.067 7	0.067 7	0.067 7	0.067 7	0.067 7	0.067 7	0.067 7	0.067 7

3.5 消融实验

为回答 RQ2 和 RQ3, 本节通过以下 2 组消融实验分别研究 NAP 的 4 种主要组件和 3 类输入数据的贡献.

3.5.1 关于 NAP 主要组件的消融实验

NAP 的主要组件包括上下文编码器的多策略编码机制、文本编码器中的双向 GRU 模型和注意力机制以及多层感知机(MLP). 本文构建以下 5 种变体.

- $\text{NAP}^{\text{ALL}}_{\text{Context}}$ 对上下文信息编码时, 不再依据源 IP 地址与目的 IP 地址进行多策略编码, 而是将所有告警按照告警名称直接生成一个频数矩阵;
- $\text{NAP}_{\text{TFIDF}}$ 和 NAP_{CNN} 将 NAP 文本编码器中的双向 GRU 模型分别替换为 TF×IDF[45]和 TextCNN[46]. 具体而言, $\text{NAP}_{\text{TFIDF}}$ 通过 TF×IDF 计算每个单词的权重 w, 并通过第 i 个单词的词向量 x_i 得到句向量 S, 即 $S=\sum_{i=1}^{N} w_i \cdot x_i$. NAP_{CNN} 在 TextCNN 模型中应用长度分别为 2–4 的 3 组卷积核, 且每组均有 100 个核;
- $\text{NAP}^{\text{NO}}_{\text{ATT}}$ 移除文本编码器中双向 GRU 模型的注意力机制;
- $\text{NAP}^{\text{NO}}_{\text{MLP}}$ 将多层感知机替换为单层感知机.

上述方法在 3 组实验对象上的平均 *NDCG@k* (k∈[1,10])见表 4. 与 $\text{NAP}^{\text{ALL}}_{\text{Context}}$ 相比, NAP 在 10 个 *NDCG@k* (k∈[1,10])指标上平均提升 14.40%. 尤其是在排序需求严格时(即 k=1 时), NAP 在 *NDCG*@1 指标上比 $\text{NAP}^{\text{ALL}}_{\text{Context}}$ 提高 29.36%. 该结果表明: 与单一的上下文告警编码机制相比, NAP 中基于源 IP 地址与目的 IP 地址的多策略编码机制更能捕捉到上下文告警中的信息. 与 $\text{NAP}_{\text{TFIDF}}$ 和 NAP_{CNN} 相比, NAP 在 10 个 *NDCG@k*(k∈[1,10])指标上的平均提升率分别为 49.68%和 5.07%. 该结果表明: 与 TF×IDF 和 TextCNN 相比, NAP 中的双向 GRU 模型更能学习到告警报文中的语义信息. 通过对比 NAP 与 $\text{NAP}^{\text{NO}}_{\text{ATT}}$ 可知, NAP 在 *NDCG@k*(k∈[1,10])指标上平均提高 16.07%. 这说明注意力机制能够有效地处理告警报文中的噪声信息, 帮助模型捕获不同风险级别的告警的特点. 另外, 与 $\text{NAP}^{\text{NO}}_{\text{MLP}}$ 相比, NAP 在 10 个 *NDCG@k*(k∈[1,10])指标上平均提高 6.00%, 该结果证明了多层

感知机对排序的贡献.

表 4 NAP 及其关于主要组件的变体在 3 组实验对象上的平均排序结果

方法	*NDCG@k*									
	1	2	3	4	5	6	7	8	9	10
NAP	**0.958 3**	**0.958 3**	**0.948 5**	**0.929 2**	**0.916 6**	**0.907 6**	**0.900 8**	**0.901 0**	**0.895 1**	**0.893 1**
$NAP^{ALL}_{Context}$	0.740 8	0.773 0	0.789 8	0.806 6	0.817 5	0.820 8	0.827 1	0.828 7	0.829 7	0.833 4
NAP_{TFIDF}	0.666 6	0.602 2	0.539 1	0.602 5	0.600 0	0.625 2	0.636 6	0.633 9	0.632 0	0.635 5
NAP_{CNN}	0.833 3	0.849 5	0.865 0	0.880 6	0.885 2	0.888 5	0.891 0	0.895 6	0.892 6	0.892 5
NAP^{NO}_{ATT}	0.812 5	0.812 5	0.805 2	0.801 1	0.794 4	0.789 6	0.777 4	0.777 9	0.780 7	0.781 1
NAP^{NO}_{MLP}	0.814 8	0.857 8	0.873 7	0.875 5	0.876 6	0.877 4	0.878 0	0.881 2	0.881 1	0.879 3

3.5.2 关于 NAP 输入数据的消融实验

本文构建 8 种变体方法研究 NAP 的 3 种输入数据(即上下文数据、文本数据和离散数据)对排序的贡献. 其中, $NAP^{NO}_{Context}$, $NAP^{NO}_{Payload}$, $NAP^{NO}_{Q\text{-}Body}$, $NAP^{NO}_{R\text{-}Body}$, NAP^{NO}_{Name}, $NAP^{NO}_{D\text{-}Port}$ 分别将输入数据中的上下文数据、告警载荷、Web 访问请求体、Web 访问响应体、告警名称、目的端口移除. 另外, $NAP^{Preceding}_{Context}$ 和 $NAP^{Following}_{Context}$ 分别仅保留上文(preceding)信息与下文(following)信息. 上述方法在 3 组实验对象上的平均 *NDCG@k*(k∈[1,10])见表 5.

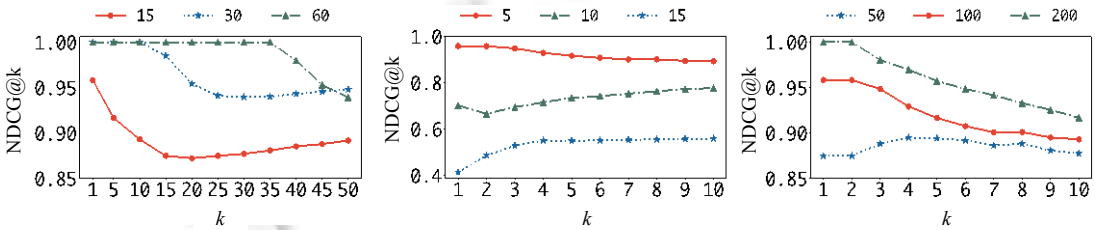
表 5 NAP 及其关于主要组件的变体在 3 组实验对象上的平均排序结果

方法	*NDCG@k*									
	1	2	3	4	5	6	7	8	9	10
NAP	**0.958 3**	**0.958 3**	**0.948 5**	**0.929 2**	**0.916 6**	**0.907 6**	**0.900 8**	**0.901 0**	**0.895 1**	**0.893 1**
$NAP^{NO}_{Context}$	0.613 9	0.692 2	0.728 1	0.753 7	0.775 8	0.791 5	0.803 5	0.814 8	0.824 4	0.832 6
$NAP^{Preceding}_{Context}$	0.666 7	0.729 4	0.752 7	0.790 5	0.809 5	0.814 1	0.817 6	0.829 9	0.833 8	0.836 9
$NAP^{Following}_{Context}$	0.708 3	0.724 4	0.720 6	0.725 6	0.745 2	0.751 4	0.756 1	0.768 0	0.777 7	0.785 8
$NAP^{NO}_{Payload}$	0.603 2	0.554 8	0.527 1	0.525 9	0.530 5	0.524 9	0.520 6	0.514 5	0.524 0	0.526 6
$NAP^{NO}_{Q\text{-}Body}$	0.833 3	0.865 6	0.877 5	0.877 1	0.876 8	0.876 6	0.876 5	0.876 4	0.876 3	0.876 2
$NAP^{NO}_{R\text{-}Body}$	0.833 3	0.849 5	0.855 3	0.858 5	0.860 6	0.857 6	0.855 3	0.850 0	0.842 7	0.836 7
NAP^{NO}_{Name}	0.658 2	0.642 0	0.660 8	0.657 1	0.652 8	0.645 4	0.635 8	0.641 2	0.645 5	0.648 2
$NAP^{NO}_{D\text{-}Port}$	0.812 5	0.794 1	0.826 3	0.814 6	0.822 1	0.823 0	0.823 7	0.826 7	0.826 6	0.824 0

与 $NAP^{NO}_{Context}$ 相比, NAP 在 *NDCG@k*(k∈[1,10])指标上的平均提升率为 21.94%, 这是因为上下文数据能够反映告警发生时的环境与状态. 另外, 通过将 NAP 与变体方法 $NAP^{Preceding}_{Context}$ 和 $NAP^{Following}_{Context}$ 相对比可知: NAP 在 *NDCG@k* (k∈[1,10])指标上比 $NAP^{Preceding}_{Context}$ 平均提高 17.62%, 比 $NAP^{Following}_{Context}$ 平均提高 23.63%, 这分别证明下文信息与上文信息对排序的贡献. 通过对比 NAP 与 $NAP^{NO}_{Payload}$ 可知, NAP 在 *NDCG@k* (k∈[1,10])指标上平均提高 72.25%. 尽管 3 组实验对象中平均只有 32.94%的告警有非空告警载荷(payload)(见表 2), 但由于 payload 对告警信息进行了详细描述, 因此, 其文本信息对排序有显著作用. 此外, 通过将 NAP 与变体方法 $NAP^{NO}_{Q\text{-}Body}$ 和 $NAP^{NO}_{R\text{-}Body}$ 相对比可知: NAP 在 *NDCG@k* (k∈[1,10])指标上比 $NAP^{NO}_{Q\text{-}Body}$ 平均提高 5.75%, 比 $NAP^{NO}_{R\text{-}Body}$ 平均提高 8.36%, 可证明 Web 访问请求体和 Web 访问响应体对排序的贡献. 该结果还表明, NAP 相对于 $NAP^{NO}_{Q\text{-}Body}$ 和 $NAP^{NO}_{R\text{-}Body}$ 的提升率明显低于 NAP 相对于 $NAP^{NO}_{Payload}$ 提升率. 这是因为 q-body 和 r-body 非空的平均比例分别为 15.28%和 13.96%(见表 2), 明显低于 payload 非空的平均比例(32.94%), 因此, q-body 和 r-body 对告警排序的帮助比 payload 有限. 另外, 与 NAP^{NO}_{Name} 相比, NAP 在 *NDCG@k* (k∈[1,10])指标上的平均提升率为 41.95%. 这表明告警名称尽管为简短中文文本, 但在一定程度上反映出告警的风险性. 最后,通过对比 NAP 与 $NAP^{NO}_{D\text{-}Port}$ 可知, NAP 在 *NDCG@k* (k∈[1,10])指标上平均提高 12.43%, 这证明离散数据目的端口对排序的贡献.

3.6 主要参数的影响

为回答 RQ4(主要参数的影响), 本节实验分别从排序间隔时间、上下文编码器活动窗口的大小、GRU 模型隐藏状态的大小这 3 个方面进行研究, 实验结果如图 3 所示. 其中: x 轴表示 k 值, y 轴表示 3 组实验对象上排序指标 *NDCG@k* 的平均值. 首先, 为研究排序间隔时间对实验结果的影响, 实验将该参数分别设置为 15 min、30 min 和 60 min. 由于当参数为 30 min 与 60 min 时, NAP 的排序指标 *NDCG@k* ($k\in[1,10]$)均达到最大值(即 1), 因此本实验将 k 值的范围扩大到[1,50], 如图 3(a)所示. 该结果表明: 随着排序间隔时间的减小、排序频率的上升, 排序难度也进一步加大. 因此, 本文设置 15 min 的排序间隔时间具有一定的挑战性. 其次,对于参数上下文编码器活动窗口的大小ΔT, 实验将其分别设置为 5 min、10 min 和 15 min. 如图 3(b)所示:随着活动窗口的增大, 排序指标 *NDCG@k* 逐渐变差. 这是因为过大的滑动窗口引入了大量无关的上下文告警, 对排序结果造成负面影响, 这也说明在排序时有必要确定合适的滑动窗口大小. 最后, 为研究 GRU 模型隐藏状态的大小对实验结果的影响, 实验将该参数分别设置为 50, 100 和 200. 由图 3(c)可知: 在一定范围内, NAP 对 GRU 模型隐藏状态的大小并不敏感.



(a) 排序间隔时间 (b) 上下文编码器活动窗口的大小 (c) GRU 模型隐藏状态的大小

图 3 不同参数配置下 NAP 在 3 组实验对象上的平均排序结果

3.7 NAP的排序效率

为研究 NAP 及对比方法面临大量待排序的网络侧告警时的表现(RQ5), 表 6 展示了各方法在排序每组实验对象中所有测试数据时所消耗的时间. 总体而言, 所有方法均能在 6 s 以内排序 1 000 个网络侧告警, 这对安全人员而言是容易接受的. 但是本方法的排序时间高于 AlertRank 和 DeepCASE, 这是由于为了挖掘出具有攻击意图的告警的特点、理解复杂多变的网络活动, 本方法融合了多元数据进行告警排序, 特别是从告警载荷、Web 访问请求体、Web 访问响应体、告警名称文本数据中学习语义信息.但总体而言, 本方法具有可接受的排序时间.

表 6 NAP 及对比方法在 3 组实验对象上的排序时间

方法	S1 (min)	S2 (min)	S3 (min)	Avg. (s/1000 个)
NAP	27.1	41.3	18.3	5.4
AlertRank	8.3	10.9	6.6	1.2
DeepCASE	24.4	32.6	20.9	3.5

4 实践评估

本方法成功应用于我校信息与网络中心的真实网络侧告警数据. 该数据来源于大规模的分布式复杂在线系统, 安全人员的数量十分有限, 因此, 安全人员难以在有限时间内对收到的网络侧告警进行人工分析, 从而及时发现出具有攻击意图的高风险告警. 本方法 NAP 能够在较短的排序间隔时间对网络侧告警按照风险级别进行排序, 满足该系统的网络安全需求, 因此, 本节将该数据用于实践评估. 该数据来源于 3 h 的真实网络侧告警, 包括暴力破解、远程代码执行、未授权访问、命令注入等常见告警, 其中: 告警名称共 141 种, 源 IP 地址共 1 761 种, 目标 IP 地址共 947 种, 风险级别共 8 种. 在实践评估中, 本文将前 15 min 的网络侧告警作为训练数据, 将其余 165 min 的告警作为测试数据.

表 7 为 NAP 与其他对比方法(AlertRank 与 DeepCASE)在真实数据集上的排序指标 *NDCG@k* ($k\in[1,10]$), 每个指标的粗体值表示所有方法中最好的结果. 由表 7 可知: 在真实数据集上, NAP 达到有效且稳定的排序结果, 排序指标 *NDCG@k* ($k\in[1,10]$)在 0.933 4–1.000 0 之间, 始终高于对比方法 AlertRank (介于 0.638 1–0.752 3)与 DeepCASE (0.857 1), 该结果进一步证明了本方法的通用性与实用性.

表 7 NAP 及对比方法在真实数据集上的排序结果

方法	*NDCG@k*									
	1	2	3	4	5	6	7	8	9	10
NAP	**1.000 0**	**0.969 9**	**0.958 6**	**0.956 9**	**0.955 7**	**0.960 5**	**0.961 7**	**0.952 3**	**0.938 4**	**0.933 4**
AlertRank	0.647 2	0.638 1	0.660 7	0.685 4	0.701 5	0.712 9	0.726 1	0.736 6	0.745 1	0.752 3
DeepCASE	0.857 1	0.857 1	0.857 1	0.857 1	0.857 1	0.857 1	0.857 1	0.857 1	0.857 1	0.857 1

5 有效性威胁

5.1 方法实现

为减少此类威胁, 本文在方法实现过程中均使用被科研社区广泛使用的工具, 从而保证方法的可靠性. 另外, 为避免在复现对比方法时出错, 本文尽可能使用开源代码, 在缺少开源代码时, 则会严格按照其论文描述复现方法并设置参数(如第 3.3 节所述).

5.2 实验数据

本文使用绿盟科技的 3 组网络攻防数据集进行实验(第 3.4 节), 并在实践评估部分(第 4 节)使用我校信息与网络中心提供的真实网络侧告警数据. 尽管第 3.1 节与第 4 节已证明了本文所使用数据的有效性和多样性, 但是这些实验数据并不能代表其他公司的网络侧告警. 另外, 对比方法 DeepCase 和 AlertRank 在评估告警风险等级时使用的数据集并未公开, 因此, 未来我们将收集更多公司的网络侧告警, 进行更大规模的评估.

6 未来工作

尽管 NAP 与对比方法 AlertRank 与 DeepCASE 相比达到最优的排序指标(如第 3.4.2 节所示), 但作为一个基于多元数据融合的方法, NAP 在追求高排序质量的同时, 在模型体量、排序效率等方面具有一定的局限性. 为弥补该缺陷, 本方法首先通过特征选择技术减少数据中的冗余信息并提高所选特征的质量, 本文在第 3.5.2 节已通过消融实验证明了每种输入数据对排序的贡献; 其次, 本方法对训练集中的无风险告警数据进行下采样(如第 2.4 节所述), 旨在保留重要特征的同时, 减少计算成本并加速模型训练.

为了进一步缓解本方法的局限性, 未来工作将从以下 3 方面展开.

(1) 模型压缩. 在后续工作中, 本方法将通过模型剪枝、量化和蒸馏等技术[47]减少模型的参数量和计算复杂度, 从而在一定程度上降低资源消耗[48];

(2) 分布式计算和并行处理. 未来我们会将网络侧告警排序任务分解为多个子任务, 并在多个计算节点上并行处理, 进而提高计算效率和资源利用率;

(3) 硬件加速和优化. 在后续工作中, 我们会在优化算法的同时, 利用专用硬件(如 GPU, TPU 等)进行加速, 以加快网络侧告警的排序速度.

7 总 结

随着网络应用的飞速发展, 各种网络安全监控系统被部署在网络节点中, 并产生大量告警. 安全人员面临处理海量告警的巨大压力, 变得对高风险告警不再敏感, 难以及时采取措施. 为尽快找出高风险的网络侧告警进行优先处理, 本文提出了基于多元数据融合的首个网络侧告警排序方法 NAP (network-side alert prioritization). 首先, 本方法设计了基于源 IP 地址与目的 IP 地址的多策略上下文告警编码器, 能够有效捕获告警的上下文信息; 其次, NAP 设计了基于注意力机制双向 GRU 模型和 ChineseBERT 模型的文本编码器, 从

告警载荷、Web 访问请求体、Web 访问响应体、告警名称文本数据中学习网络侧告警的语义信息, 进而理解当前复杂多变的网络活动; 最后, NAP 构建了排序模型得到告警排序值, 并按其降序将具有攻击意图的高风险告警排在前面, 从而优化网络侧告警的管理流程. 本文在绿盟科技的3组网络攻防数据上进行实验, 平均排序指标 *NDCG@k* ($k\in[1,10]$)均在 0.893 1–0.958 3 之间, 比最先进的方法提升 64.73%以上. 实验结果表明: NAP 能有效且稳定地排序网络侧告警, 且显著优于对比方法. 另外, 通过将 NAP 应用于我校真实的网络侧告警数据, 本文进一步证实了其实用性.

References:

[1] CNN. What we know about the pipeline ransomware attack: How it happened, who is responsible and more. 2021. https://edition.cnn.com/2021/05/10/politics/colonial-ransomware-attack-explainer/index.html

[2] Khraisat A, Gondal I, Vamplew P, *et al*. Survey of intrusion detection systems: Techniques, datasets and challenges. Cybersecurity, 2019, 2(1): 1–22. [doi: 10.1186/s42400-019-0038-7]

[3] Prandl S, Lazarescu M, Pham DS. A study of web application firewall solutions. In: Proc. of the 11th Int'l Conf. on Information Systems Security (ICISS 2015). 2015. 501–510. [doi: 10.1007/978-3-319-26961-0_29]

[4] Hassan WU, Guo S, Li D, *et al*. Nodoze: Combatting threat alert fatigue with automated provenance triage. In: Proc. of the 26th Annual Network and Distributed System Security Symp. (NDSS 2019). 2019.

[5] Orca. The orca security 2022 cloud security alert fatigue report. 2022. https://orca.security/resources/blog/2022-cloud-cyber-security-alert-fatigue-report/

[6] Lin Y, Chen Z, Cao C, *et al*. Collaborative alert ranking for anomaly detection. In: Proc. of the 27th ACM Int'l Conf. on Information and Knowledge Management (CIKM 2018). 2018. 1987–1995. [doi: 10.1145/3269206.3272013]

[7] Alahmadi BA, Axon L, Martinovic I. 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. In: Proc. of the 31st USENIX Security Symp. (USENIX Security 2022). 2022. 2783–2800.

[8] Liu J, Zhang R, Liu W, *et al*. Context2Vector: Accelerating security event triage via context representation learning. Information and Software Technology, 2022, 146: 106856. [doi: 10.1016/j.infsof.2022.106856]

[9] Zhao N, Jin P, Wang L, *et al*. Automatically and adaptively identifying severe alerts for online service systems. In: Proc. of the IEEE Conf. on Computer Communications (INFOCOM 2020). 2020. 2420–2429. [doi: 10.1109/ INFOCOM41043.2020.9155219]

[10] Liu J, Su PR, Yang M, *et al*. Software and cyber security—A survey. Ruan Jian Xue Bao/Journal of Software, 2018, 29(1): 42–68 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5320.htm [doi: 10.13328/j.cnki. jos.005320]

[11] Jiang G, Chen H, Yoshihira K, *et al*. Ranking the importance of alerts for problem determination in large computer systems. In: Proc. of the 6th Int'l Conf. on Autonomic Computing. 2009. 3–12. [doi: 10.1145/1555228.1555232]

[12] Ben-Asher N, Gonzalez C. Effects of cyber security knowledge on attack detection. Computers in Human Behavior, 2015, 48: 51–61. [doi: 10.1016/j.chb.2015.01.039]

[13] Zhao N, Chen J, Wang Z, *et al*. Real-time incident prediction for online service systems. In: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. 2020. 315–326. [doi: 10.1145/3368089.3409672]

[14] Li B, Yang T, Chen Z, *et al*. Heterogeneous anomaly detection for software systems via attentive multi-modal learning. arXiv:2207.02918, 2022.

[15] Chen J, Zhang S, He X, *et al*. How incidental are the incidents? Characterizing and prioritizing incidents for large-scale online service systems. In: Proc. of the 35th IEEE/ACM Int'l Conf. on Automated Software Engineering. 2020. 373–384. [doi: 10.1145/3324884.3416624]

[16] Van Ede T, Aghakhani H, Spahn N, *et al*. Deepcase: Semi-supervised contextual analysis of security events. In: Proc. of the 2022 IEEE Symp. on Security and Privacy (SP 2022). IEEE, 2022. 522–539. [doi: 10.1109/SP46214.2022.00036]

[17] Shen Y, Stringhini G. Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In: Proc. of the 28th USENIX Security Symp. (USENIX Security 2019). 2019. 905–921.

[18] Qin ZQ, Ma XK, Wang YJ. Attentional payload anomaly detector for Web applications. In: Proc. of the 25th Int'l Conf. on Neural Information Processing (ICONIP 2018). Springer, 2018. 588–599. [doi: 10.1007/978-3-030-04212-7_52]

[19] Jin X, Cui B, Yang J, *et al*. Payload-based Web attack detection using deep neural network. In: Advances on Broad-band Wireless Computing, Communication and Applications: Proc. of the 12th Int'l Conf. on Broad-band Wireless Computing, Communication and Applications (BWCCA 2017). Springer, 2018. 482–488. [doi: 10.1007/978-3-319-69811- 3_44]

[20] Jin X, Cui B, Li D, *et al*. An improved payload-based anomaly detector for Web applications. Journal of Network and Computer Applications, 2018, 106: 111–116. [doi: 10.1016/j.jnca.2018.01.002]

[21] Torrano-Gimenez C, Nguyen HT, Alvarez G, *et al*. Applying feature selection to payload-based web application firewalls. In: Proc. of the 3rd Int'l Workshop on Security and Communication Networks (IWSCN 2011). IEEE, 2011. 75–81. [doi: 10.1109/IWSCN.2011.6827720]

[22] Wang LM, Bu L, Ma LZ, *et al*. Attack detection method based on indicator-dependent model construction and monitoring. Ruan Jian Xue Bao/Journal of Software, 2023, 34(6): 2641–2668 (in Chinese with English abstract). http://www.jos. org.cn/1000-9825/6847.htm [doi: 10.13328/j.cnki.jos.006847]

[23] Chen J, Wang P, Wang W. Online summarizing alerts through semantic and behavior information. In: Proc. of the 44th Int'l Conf. on Software Engineering (ICSE 2022). 2022. 1646–1657. [doi: 10.1145/3510003.3510055]

[24] Zhao N, Chen J, Peng X, *et al*. Understanding and handling alert storm for online service systems. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering: Software Engineering in Practice (ICSE-SEIP 2020). 2020. 162–171. [doi: 10.1145/3377813.3381363]

[25] Lin D, Raghu R, Ramamurthy V, *et al*. Unveiling clusters of events for alert and incident management in large-scale enterprise it. In: Proc. of the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD 2014). 2014. 1630–1639. [doi: 10.1145/2623330.2623360]

[26] Ma M, Zhang S, Chen J, *et al*. Jump-Starting multivariate time series anomaly detection for online service systems. In: Proc. of the 2021 USENIX Annual Technical Conf. (USENIX ATC 2021). 2021. 413–426.

[27] Zhao N, Chen J, Yu Z, *et al*. Identifying bad software changes via multimodal anomaly detection for online service systems. In: Proc. of the 29th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering (ESEC/FSE 2021). 2021. 527–539. [doi: 10.1145/3468264.3468543]

[28] Landauer M, Skopik F, Wurzenberger M, *et al*. Dealing with security alert flooding: using machine learning for domain-independent alert aggregation. ACM Trans. on Privacy and Security, 2022, 25(3): 1–36. [doi: 10.1145/3510581]

[29] Chung J, Gulcehre C, Cho K, *et al*. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: Proc. of the Neural Information Processing Systems Workshop on Deep Learning (NeurIPS 2014). 2014.

[30] Wang W, Chen J, Yang L, *et al*. How long will it take to mitigate this incident for online service systems? In: Proc. of the 32nd Int'l Symp. on Software Reliability Engineering (ISSRE 2021). IEEE, 2021. 36–46. [doi: 10.1109/ISSRE52982.2021.00017]

[31] Wang W, Chen J, Yang L, *et al*. Understanding and predicting incident mitigation time. Information and Software Technology, 2023, 155: 107119. [doi: 10.1016/j.infsof.2022.107119]

[32] Yang L, Chen J, Wang Z, *et al*. Semi-supervised log-based anomaly detection via probabilistic label estimation. In: Proc. of the 43rd IEEE/ACM Int'l Conf. on Software Engineering (ICSE 2021). IEEE, 2021. 1448–1460. [doi: 10.1109/ICSE43902.2021.00130]

[33] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. In: Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP 2014). 2014. 1532–1543.

[34] Devlin J, Chang MW, Lee K, *et al*. Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019). 2019. 4171–4186.

[35] Kang Y, Wang Z, Zhang H, *et al*. Apirecx: Cross-library api recommendation via pre-trained language model. In: Proc. of the 2021 Conf. on Empirical Methods in Natural Language Processing (EMNLP 2021). 2021. 3425–3436. [doi: 10.18653/v1/ 2021.emnlp-main.275]

[36] Shen Q, Chen J, Zhang JM, *et al*. Natural test generation for precise testing of question answering software. In: Proc. of the 37th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2022). 2022. 1–12. [doi: 10.1145/3551349. 3556953]

[37] Gao T, Chen J, Zhao Y, *et al*. Vectorizing program ingredients for better JVM testing. In: Proc. of the 32nd ACM SIGSOFT Int'l Symp. on Software Testing and Analysis (ISSTA 2023). 2023. 526–537. [doi: 10.1145/3597926.3598075]

[38] Sun Z, Li X, Sun X, *et al*. Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. In: Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int'l Joint Conf. on Natural Language Processing (ACL/IJCNLP 2021) (Vol.1: Long Papers). 2021. 2065–2075. [doi: 10.18653/v1/2021.acl-long.161]

[39] Zhang Y, Wu X, Fang Q, *et al*. Knowledge-enhanced attributed multi-task learning for medicine recommendation. ACM Trans. on Information Systems (TOIS), 2023, 41(1): 1–24. [doi: 10.1145/3527662]

[40] Mostafa S, Wang X, Xie T. Perfranker: Prioritization of performance regression tests for collection-intensive software. In: Proc. of the 26th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis (ISSTA 2017). 2017. 23–34. [doi: 10.1145/3092703.3092725]

[41] Su Y, Xing Z, Peng X, *et al*. Reducing bug triaging confusion by learning from mistakes with a bug tossing knowledge graph. In: Proc. of the 36th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2021). IEEE, 2021. 191–202. [doi: 10.1109/ASE51524.2021.9678574]

[42] Chen N, Lin J, Hoi SCH, *et al*. AR-Miner: Mining informative reviews for developers from mobile app marketplace. In: Proc. of the 36th Int'l Conf. on Software Engineering (ICSE 2014). 2014. 767–778. [doi: 10.1145/2568225.2568263]

[43] Al-Maskari A, Sanderson M, Clough P. The relationship between IR effectiveness measures and user satisfaction. In: Proc. of the 30th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. 2007. 773–774. [doi: 10.1145/1277741.1277902]

[44] Järvelin K, Kekäläinen J. Cumulated gain-based evaluation of IR techniques. ACM Trans. on Information Systems (TOIS), 2002, 20(4): 422–446. [doi: 10.1145/582415.582418]

[45] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing & Management, 1988, 24(5): 513–523.

[46] Dos Santos C, Gatti M. Deep convolutional neural networks for sentiment analysis of short texts. In: Proc. of the 25th Int'l Conf. on Computational Linguistics: Technical Papers (COLING 2014). 2014. 69–78.

[47] Lei J, Gao X, Song J, *et al*. Survey of deep neural network model compression. Ruan Jian Xue Bao/Journal of Software, 2018, 29(2): 251–266 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5428.htm [doi: 10.13328/ j.cnki.jos.005428]

[48] Gao H, Tian YL, Xu FY, *et al*. Survey of deep learning model compression and acceleration. Ruan Jian Xue Bao/Journal of Software, 2021, 32(1): 68–92 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6096.htm [doi: 10.13328/j.cnki.jos.006096]

附中文参考文献:

[10] 刘剑, 苏璞睿, 杨珉, 等. 软件与网络安全研究综述. 软件学报, 2018, 29(1): 42–68. http://www.jos.org.cn/1000-9825/5320.htm [doi: 10.13328/j.cnki.jos.005320]

[22] 王立敏, 卜磊, 马乐之, 等. 基于指标依赖模型构建与监控的攻击检测方法. 软件学报, 2023, 34(6): 2641–2668. http://www.jos.org.cn/1000-9825/6847.htm [doi: 10.13328/j.cnki.jos.006847]

[47] 雷杰, 高鑫, 宋杰, 等. 深度网络模型压缩综述. 软件学报, 2018, 29(2): 251–266. http://www.jos.org.cn/1000-9825/5428.htm [doi: 10.13328/j.cnki.jos.005428]

[48] 高晗, 田育龙, 许封元, 等. 深度学习模型压缩与加速综述. 软件学报, 2021, 32(1): 68–92. http://www.jos.org.cn/1000-9825/6096.htm [doi: 10.13328/j.cnki.jos.006096]

王维靖(1997－), 女, 博士生, CCF 学生会员, 主要研究领域为智能运维, 软件工程.

陈俊洁(1992－), 男, 博士, 副教授, 博士生导师, CCF 高级会员, 主要研究领域为软件分析与测试.

杨林(1995－), 男, 博士生, CCF 学生会员, 主要研究领域为软件工程, 软件测试生成.

侯德俊(1979－), 男, 高级工程师, 主要研究领域为网络与空间安全, 软件开发及安全.

王星凯(1992－), 男, 博士, CCF 专业会员, 主要研究领域为安全智能分析, 人工智能安全, 安全知识图谱.

吴复迪(1993－), 男, 主要研究领域为安全攻防及其自动化.

张润滋(1989－), 男, 博士, CCF 高级会员, 主要研究领域为智能安全运营, 威胁狩猎.

王赞(1979－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为软件测试, 机器学习.