中国研究生网络安全创新大赛

作品报告

作品名称：SOCLLM：基于大模型的 SOC 设备网络侧告警降噪研判系统

提交日期：2024 年 9 月 23 日

填写说明

1. 所有参赛项目必须为一个基本完整的设计。作品报告书旨在能够清晰准确地阐述（或图示）该参赛队的参赛项目（或方案）。

2. 作品报告采用A4纸撰写。除标题外，所有内容必需为宋体、小四号字、1.5倍行距。

3. 作品报告中各项目说明文字部分仅供参考，作品报告书撰写完毕后，请删除所有说明文字。(本页不删除)

4. 作品报告模板里已经列的内容仅供参考，作者可以在此基础上增加内容或对文档结构进行微调。

5. 为保证网评的公平、公正，作品报告中应避免出现作者所在学校、院系和指导教师等泄露身份的信息。一经发现，取消作品参赛资格。

目　　录

摘要

在网络技术飞速发展的今天，网络安全问题日益凸显，传统的人工安全告警分析方法由于其效率低下和准确度不足正面临挑战。尤其在大数据环境下，高频率的告警数据需要更快速、更精确的处理方式。此外，以ChatGPT为代表的大语言模型虽在数据处理上表现出色，但其在网络安全应用中还存在成本高、隐私泄露风险大等问题。因此，迫切需要为特定的网络安全场景开发优化的人工智能模型，以提高告警的处理效率和准确性。

为解决上述问题，我们设计并实现了基于大模型的SOC设备网络侧告警降噪研判系统，该系统具备三大核心功能：一是高效的告警过滤，支持告警降噪和聚合；二是智能的告警研判，能够通过多维度方法对告警进行深度分析，识别出高威胁告警；三是大模型的辅助研判，利用大模型在理解告警语言的优势和人工交互的能力辅助运维人员进行告警研判，显著提升告警研判效率。

在实际部署运营后，本告警降噪研判系统可以将每日百万级别的网络侧告警数据高效地聚合，大幅减少重复告警，降噪率最高可达99.56%。对聚合后的告警，大模型可研判的部分达到82.01%，有效减轻了人工研判的压力。辅以用户友好的前端交互界面，本项目可以大幅减轻实际生产中每日百万条告警带来的人力负担，为行业提供可靠的安全防护支持，促进网络安全运维新技术的发展。

关键词：告警过滤；告警研判；大语言模型

第一章 作品概述

1.1 背景分析

告警日志是信息技术和网络安全领域中一种关键的数据记录方式，专门用于捕捉和记录安全系统检测到的各种异常和潜在威胁。这些日志详细记录了从可疑网络活动到确认的安全事件的信息，包括但不限于未授权访问、恶意软件活动、数据泄露尝试等。它们为企业的安全运营中心（Security Operations Center, SOC）提供了实时的数据支持，使得安全团队能够快速识别、调查和响应各种安全威胁，以维护企业内部的安全[1]。通过定期和系统地分析告警日志，企业不仅能够防御当前的攻击，还能预防未来的安全威胁，从而保护关键的业务资产和数据。

但是，随着企业网络环境的日益复杂化和安全威胁的持续增加，手动分析和处理大量的安全告警日志变得越来越困难。对于一个中等规模的企业而言，其日志数据量通常可达到百万级别，每天可能产生超过 1TB 的日志数据，但其中只有低于 1%的告警是需要真正关注的潜在安全威胁，这远远超出了手工分析的能力范围。这种海量告警数据不仅大幅降低了分析效率，还延长了告警的响应时间。此外，告警日志本身的多样性与复杂性增加了分析的挑战，即使是经验丰富的安全分析师也可能错过一些关键告警。这些遗漏的告警可能导致潜在的安全风险未能被及时发现和处理，从而对企业的安全防护能力构成严重威胁。

在过去几年中，人工智能技术已逐步应用于网络安全领域[2]，显著提高了威胁检测与响应的效率。特别是像 ChatGPT 这样的大模型技术的发展，为网络安全领域带来了新的活力，推动了安全技术的革新与进步。然而，尽管这些 AI 大模型在处理大规模数据方面表现出卓越的能力，它们在具体的网络安全应用中仍面临成本高、数据隐私风险大以及缺乏针对特定场景的定制化等挑战。因此，开发专为企业私有化部署并优化的网络安全场景的 AI 大模型已成为行业的重要发展方向。

1.2 相关工作

本节将首先从学术层面介绍现阶段大模型技术在网络安全中的应用。这包括详细分析当前的研究文献，探讨学者们如何利用这些先进的模型来识别、分类和响应网络威胁。接着，转向工业界层面，列举几个标志性的业界案例，这些案例展示了如何将大模型技术集成到企业的安全运营中心，以及这些技术如何帮助企业提升了处理安全告警的效率和准确性。

1.2.1 大模型在网络安全领域的应用

大语言模型在网络安全领域的应用日益增加[3]，特别是在日志分析、威胁检测方面。

Ferrag 等人（2023 年）介绍了 SecurityLLM 模型[4]，这是一种专为网络安全威胁检测设计的预训练语言模型，展示了在日志数据分析和威胁检测上的突破性进展。该模型包括两个关键生成元素：SecurityBERT 用于网络威胁检测，而 FalconLLM 用作事故响应和恢复系统。他们的实验显示，该模型能够以 98% 的准确率识别十四种不同类型的攻击，超越了传统机器学习和深度学习方法的性能。

Sultana 等人（2023 年）提出了一个评估和理解大型语言模型在网络操作自动化中应用的框架[5]。这一框架首先识别了一系列具有逐渐增加复杂性的防御性网络操作任务，并提出创建新的数据集来完成这些任务。通过审视近期利用大型语言模型进行下游网络操作任务的工作，他们识别了研究空白和未解决的问题。此外，他们还提出了一个理解和评估网络操作任务的框架，并报告了可能的解决方案和研究方向，为大型语言模型在网络领域的可靠应用提供了理论和实践的支持。

Bayer 等人（2022 年）开发了一个为网络安全领域量身定制的语言模型，称为 CySecBERT[6]。这个模型通过为网络安全系统提供一个基本构建块，解决了通用语言模型在处理技术术语和领域知识时可能遗漏重要信息的问题。CySecBERT 在 15 种不同的领域依赖任务中进行了比较，包括序列标记和分类任务，显示出在特定应用场景中的优势。该模型还通过反对灾难性遗忘的措施，能够保留之前训练的领域独立知识，从而改善了内部表示空间的单词，比其他模型表现更好。该模型和使用的数据集已公开可用，为网络安全研究和实践提供支持。

此外，Karlsen 等人（2023 年）通过研究不同架构的大语言模型（包括 BERT[7]、RoBERTa[8]、DistilRoBERTa[9]、GPT-2[10]和 GPT-Neo[11]）在系统和应用程序日志文件分析中的应用，发现这些经过微调的模型能有效进行日志分析，尤其是在特定日志类型的领域适应方面显示出色[12]。

1.2.2 工业界安全大模型

近年来，随着人工智能技术的快速进步，大模型在工业界的应用已显著增加。跟据国际数据公司 IDC 发布的《破土萌芽——大模型在网络安全领域的应用市场洞察报告》显示[13]，奇安信、深信服等多家网络安全厂商已基于通用大模型进行再训练和微调，开发出适用于垂直领域的安全大模型。这些模型具有强大的计算能力和逻辑推理能力，能对多源实时数据进行分析，提升了对异常行为、潜在威胁，尤其是未知威胁的检测效率和准确性。

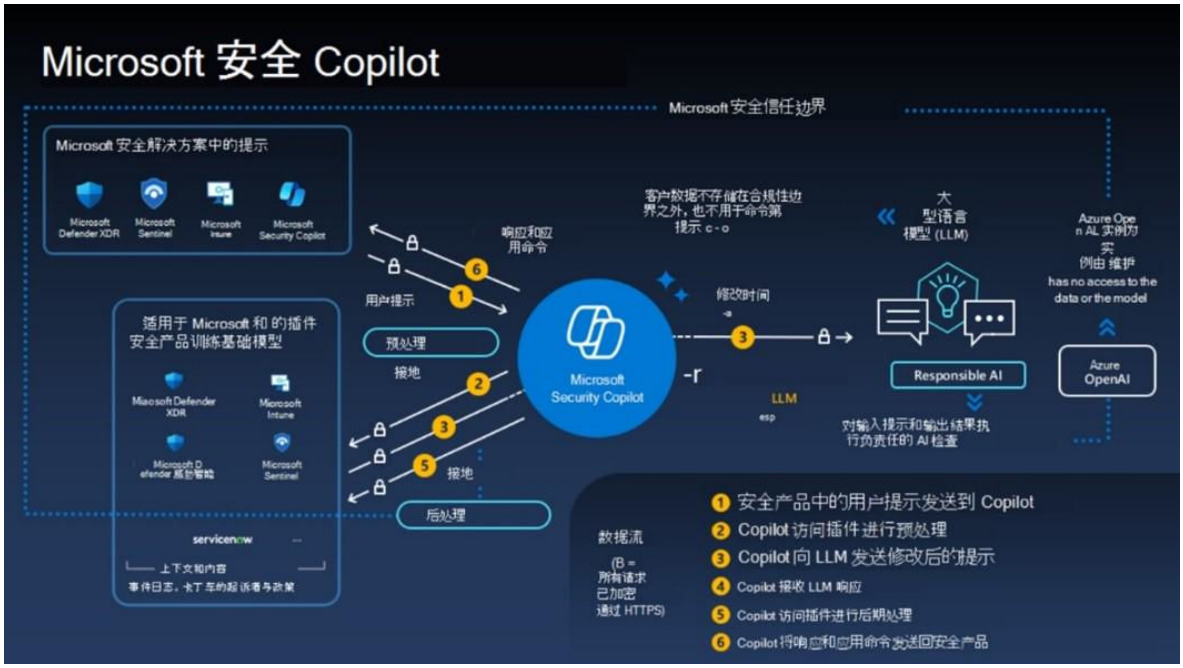

图 1-1 Microsoft Security Copilot

微软推出的 Microsoft Security Copilot 代表了安全大模型技术的前沿应用[14]，展现了人工智能在提高网络安全防御能力和事件响应效率方面的巨大潜力。Security Copilot 是一个集成了先进机器学习和大型语言模型技术的安全分析和操作平台，旨在通过 AI 技术支持安全分析师更有效地识别、分析和应对安全威胁。

该工具的核心优势在于其能够处理和分析大量数据，快速识别潜在的安全威胁和

漏洞，并提供深入的威胁分析报告。通过支持自然语言查询，Security Copilot 允许安全专家以直观的方式提出问题，从而大幅提高工作效率。此外，它还能提供决策支持与自动化响应建议，辅助安全团队做出快速准确的决策，并在可能的情况下，自动化执行响应措施。

图 1-1 展示了 Microsoft Security Copilot 在网络安全解决方案中的应用架构。Security Copilot 集成了 Microsoft 的多种安全产品（如 Microsoft Defender 和 Microsoft Sentinel），通过大型语言模型（如 Azure OpenAI）进行优化。用户的安全查询或请求经过预处理和筛选后，发送给 Security Copilot，利用大型语言模型生成响应，并通过 Responsible AI 模型审核，确保建议的合规性和安全性。最终，经过审核的建议返回给用户，同时 Security Copilot 还可以与其他 Microsoft 安全产品协调，执行自动化的安全响应和处理，从而提升安全告警的处理效率和准确性。

Microsoft Security Copilot 的设计目标是成为安全运营中心（SOC）和安全分析师的强大助手，通过不断学习新的安全事件和威胁情报来优化其检测和响应策略。它的推出不仅体现了大模型技术在安全领域应用的创新，也为提升组织的安全防御体系提供了重要工具，标志着网络安全技术进入了一个以人工智能为核心的新时代。

图 1-2 深信服安全 GPT

深信服安全 GPT 是中国首个自主研发的企业级安全 GPT 云端大模型，专门针对网络安全领域设计。该模型综合了高质量的安全语料和数据分析，通过预训练和微调的方式，增强了网络安全的检测和响应能力。深信服安全 GPT 能够进行流量检测、事件解析、安全建议生成等，显著提升了安全事件的处置效率和准确性。

深信服安全 GPT 具有多个卓越的功能特点，包括高级威胁检测、自然语言交互能力和全天候智能值守。这使得该模型不仅可以提高威胁检出率，还可以通过自然语言与安全系统进行交互，极大地简化了安全运营的流程。此外，深信服安全 GPT 还强调了数据的安全可控性，确保所有操作和数据处理均在用户的可控范围内进行。

图 1-2 展示了深信服安全大模型的技术路线，整个过程从数据收集到模型应用分为三个主要阶段。首先，在数据阶段，通过收集日志、网络流量、事件与告警以及其他相关安全数据来构建训练集。接下来，在模型阶段，利用这些数据对预训练的 Transformer 模型进行微调，保持部分结构固定，同时更新某些层以适应特定的安全分析任务。最后，在应用阶段，微调后的大模型被部署到实际的安全场景中，执行多种任务如自然语言交互、流量研判、事件解析等，为安全管理提供全面的智能化支持。

深信服安全 GPT 的开发和实施体现了中国在网络安全技术领域的自主创新能力，不仅填补了国内在该领域的技术空白，也为全球网络安全技术的发展提供了中国方案。

然而，尽管这些 AI 大模型在处理大规模数据方面展示了卓越的能力，它们在具体的网络安全应用中仍面临成本高、数据隐私风险大以及缺乏针对特定场景的定制化等挑战[15]。

1.3 特色描述

1.3.1 针对网络侧告警

与现有的大多数专注于终端侧设备告警的研判大模型不同，本项目聚焦于更为复杂且多变的网络侧设备告警。与终端侧告警相比，网络侧告警往往数量庞大，来源分散，且涉及多个网络设备，告警信息内容的可读性和理解性较差。这些特性使得网络侧的问题更为隐蔽，运维人员分析和研判的难度显著增加。本项目针对网络侧告警的专门设计，能够更有效地从巨量告警中过滤大量错报漏报，识别潜在的威胁和异常行为。

1.3.2 分析告警来源广

目前，国内一些安全厂商也推出了安全运维大模型产品，但这些产品主要针对厂

商自家产品的告警进行分析，局限于厂商的产品生态内，从而缺乏对其他厂商的设备和系统的综合支持。但是现在大多数企业都采用了多种网络安全设备来维护其网络安全，不同设备很可能来自于不同的厂商。本项目则突破了这一限制，能够整合来自不同厂商和平台的告警信息，进行全面分析，具有更高的兼容性。这种兼容性使得本项目可以接收更广泛的告警来源，更与当前多样化的网络安全运维形势相契合，增强了对复杂网络环境中潜在威胁的识别能力。

1.3.3 可扩展性高

考虑到未来技术的发展，本项目在设计上充分考虑了可扩展性。无论是现有还是新兴的日志类型，都可以通过更改配置文件的方式来集成到系统中。这意味着随着新的安全威胁出现，或者是使用者引入了新的网络安全设备，本项目都能够灵活地适应变化，持续的提供有效的网络告警研判支持。

1.3.4 易用性强

传统的告警研判方式通常需要具备高水平安全知识的运维人员来处理。在全身心的投入到告警研判的处理中时，每位专家平均每天也仅能处理上百条告警，效率较低。本项目通过告警过滤除去大部分业务误报和错报，并且通过多维度的告警研判，选出更据威胁的告警，极大的减少待处理告警数量。并且有了大模型的辅助，简化了告警分析的流程，通过大模型提供的研判建议，运维人员可以节省大量时间，大幅提升整体的告警处理速度和准确度。通过这种智能化辅助，运维人员能够更专注于解决高优先级问题，从而显著提高网络安全的整体运营效率。

1.4 应用前景

1.4.1 大模型提升运维效率与准确性

本项目引入大模型的方法，显著提升运维团队的工作效率。与传统的人工研判相比，大模型能够快速识别告警信息中的关键特征，并给出研判建议，这不仅减少了错误研判的可能性，还大幅降低了运维人员的工作量。

1.4.2 跨厂商设备集成

由于当前市场上大多数运维大模型仅支持特定厂商的产品，而企业的网络环境通常由来自多个厂商的安全设备构成，设备类型和告警数据各异。本项目的设计考虑到了这一点，通过强大的兼容性和扩展性，能够有效整合和分析来自不同厂商设备的告警信息，在多设备、多平台共存的复杂网络环境中具有广泛的应用潜力。不同来源的日志信息，还能实现资源共享，提升运维的整体能力。

1.4.3 多行业应用场景中的广阔前景

本项目不仅适用于传统的互联网行业，在金融、政府、能源等关键行业中也有着广阔的应用前景。随着企业数字化的推进，这些领域对网络安全的需求尤为迫切，对安全的要求也更高。通过本项目的高效告警分析与研判能力，可以为这些行业提供可靠的安全防护支持，助力其数字化转型与安全保障。

1.4.4 促进网络安全运维技术发展

由于大语言模型擅长处理自然语言文本，可以对复杂、模糊的描述进行理解和解析，越来越多的人注意到了大模型在告警研判方面的作用。本项目通过对网络告警研判框架的构建和告警研判大模型的实现，让更多人认识到大模型在这个方向上的潜力，吸引更多科研人员进行该方向的研究，促进网络安全运维新技术的发展。

第二章 作品设计与实现

2.1 整体方案设计与实现

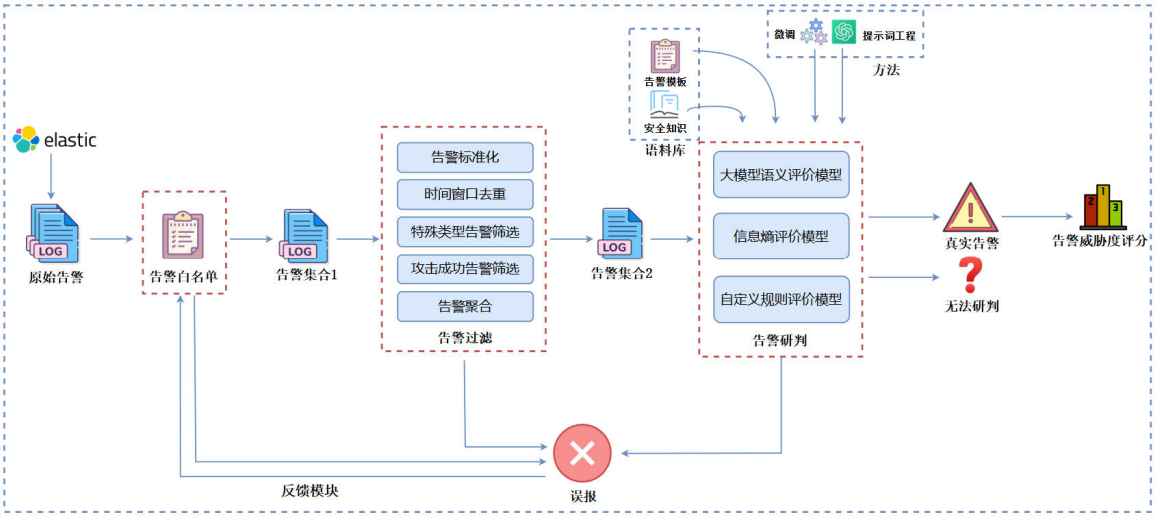

图 2-1 系统总体框架图

本项目主要由研判侧和运营侧两部分组成。整体框架图如图 2-1 所示。

研判侧包括三个主要组成部分：告警白名单模块、告警过滤模块以及告警研判模块。告警白名单模块通过人工更新的白名单来识别和筛除已知无害或常见的误报告警，有效减轻后续处理的压力。此外，该模块还能接收来自后续模块的反馈，根据实际运行情况动态更新白名单，以提高过滤的精确性和实效性。告警过滤模块首步为告警标准化，即将来自不同设备和不同类别的告警统一转换成一种格式，以便进行后续的详细分析。这一环节还包括归并时间接近的告警、剔除特定类型的无效告警、筛选标记为攻击成功的告警，并通过一对多、多对一、一对一的方法进行告警聚合，以优化告警的处理流程和提高准确性。最后，经过筛选与聚合的告警数据进入告警研判模块。该模块融合了大模型语义评价模型、信息熵评价模型及自定义规则评价模型三种评估机制，全面评估每个告警的潜在风险和紧急性。超过预设阈值的告警将被标记为需要进一步关注，确保安全团队能优先处理更高风险的事件。

运营侧主要聚焦于与用户的交互功能，包括通过 UI 界面展示分析结果，提供告警字段解析，以及实现安全知识的问答等功能。

2.2 告警白名单模块

2.2.1 模块概述

告警白名单模块是网络安全告警管理系统的重要组成部分，旨在减少误报告警和正常业务告警对运维人员的干扰和影响。通过将符合条件的告警特征加入白名单，本模块可以过滤掉不具备安全威胁的告警，从而减少无效告警的数量，优化输入大模型的告警流质量。

在与企业的合作实践和交流中，根据不同的需求和应用场景，本模块提供了两种主要的白名单模式：自定义规则白名单和结果反馈白名单。自定义规则白名单允许用户根据自身安全需求手动配置白名单规则；结果反馈白名单则基于实际告警结果进行动态调整，自动将经过研判为无威胁的告警加入白名单。

白名单模块大致工作流程如图 2-2 所示。

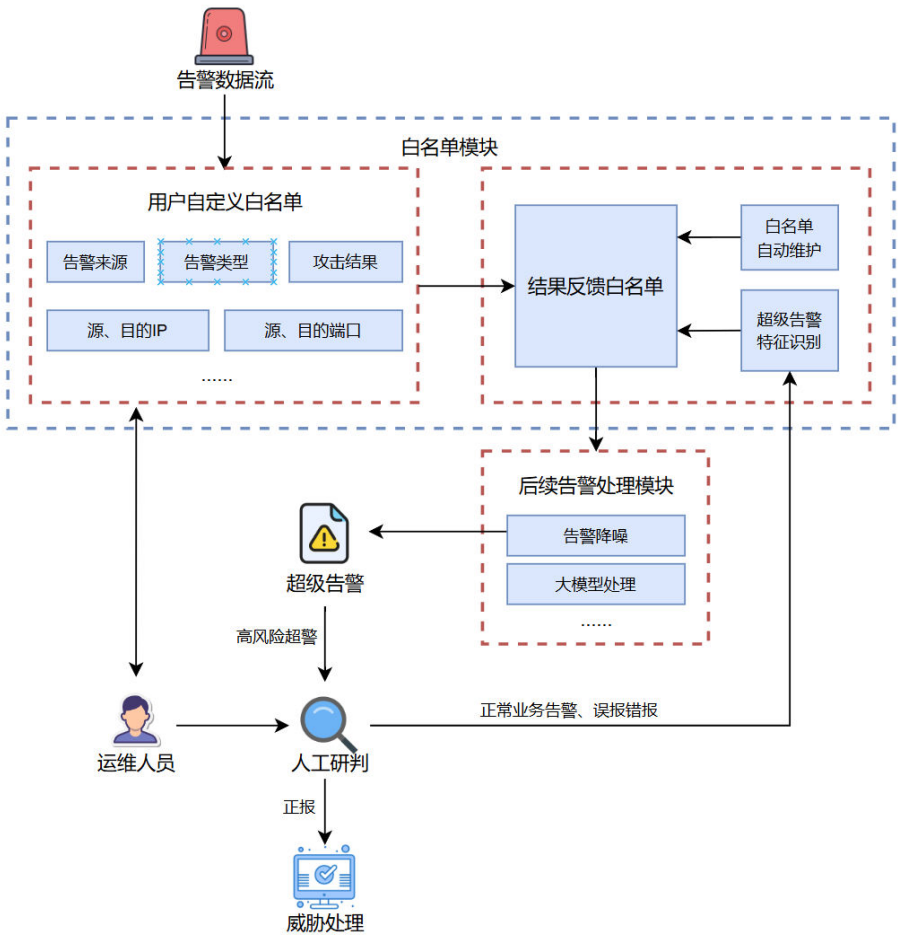

图 2-2 白名单模块工作流程

2.2.2 自定义规则白名单

自定义规则白名单为运维人员提供灵活的配置选项，允许根据不同的告警要素构建符合具体业务需求的白名单，这在应对部分特殊场景时非常有效，比如特定的业务需求可能频繁的触发某些设备的规则导致产生大量的业务误报等。

用户可以在系统中设置多种加白条件，包括但不限于以下几种要素：

- 告警来源：区分告警源头，例如某些已知可信的监控系统或防火墙触发的告警，可以加入白名单。
- IP 地址：用户可以针对特定的内部或可信 IP 地址设置白名单，防止误报。
- 端口号：某些已知的端口在特定场景中可能不具备威胁，用户可以基于此进行配置。
- 告警类型：基于特定的告警类型（如不涉及安全的行为告警或频繁误报的类型），运维人员可以加白过滤。
- 攻击结果：对于某些特定的攻击场景，用户可以根据攻击结果来设置白名单，例如针对未成功的攻击。

用户不仅可以单独使用上述条件来创建白名单项，还可以根据业务需求，将多个条件进行组合，形成更加细粒度的白名单规则。例如，可以为某个特定的 IP 地址和端口组合加白，或为特定的告警类型和告警来源共同作用于某个白名单项。通过自定义规则白名单，用户可以自主地提升告警管理的准确性和效率。

自定义白名单采用 python 加 redis 进行开发。通过维护一个 WhitelistManager 类来对自定义白名单进行管理。WhitelistManager 类中定义了增加、删除、查询、匹配白名单项的方法，并且由于限定了告警特征的顺序，每个相同的白名单生成唯一的键，存储为 redis 哈希。用户可以通过使用类的各种方法来维护自定义白名单；当告警通过白名单时，系统将提取告警的特征并调用白名单匹配方法来判断告警是否允许通过。

2.2.3 结果反馈白名单

结果反馈白名单是告警白名单模块的另一重要功能，旨在根据后续研判结果进行自动化的白名单维护。当某些告警通过后续模块生成了超级告警，并经过人工加大模

型的分析确认该超级告警为误报，或判断该告警属于正常业务产生的非威胁告警时，系统可以自动将这些超级告警加入到结果反馈白名单中，以便未来相似告警不再触发安全告警。下面是结果反馈白名单的两个子模块的具体实现。

2.2.3.1 白名单自动化添加

系统将用户的反馈过程自动化。超级告警是多个告警综合处理后得出的一系列相似告警的集合，当用户将某条超级告警加白后，系统会自动这些告警的特征信息，例如告警来源、IP、端口、告警类型等，基于这些信息生成相应的白名单项。通过自动化识别与处理，系统减少了用户手动配置白名单的负担，同时保证白名单的实时更新和准确性。

当人工研判某超级告警后，系统将自动的提取出超级告警中所有告警的特征，并按照元组的方式形成唯一的 key，并为每个白名单项生成一个 json 作为 value，json 中储存了白名单的命中次数和最近几次的命中时间，最终将这些白名单项自动存入到 redis 中。

2.2.3.2 白名单维护

为了确保白名单的有效性和及时更新，结果反馈白名单采用了分权重和 TTL（Time-to-Live，生存时间）结合的自动化维护机制。该机制确保了白名单中的条目能够根据实际情况进行调整，避免冗余数据积累，保证系统的整体性能和准确性。具体而言，系统通过以下方式对白名单项进行管理：

- 权重分组：系统根据每条白名单项的命中次数，将其分配至不同的权重组。命中次数越多，表明该条白名单项更具持久性价值，其权重也越高。
- TTL 更新机制：权重越高的白名单项，其生存时间（TTL）也越长。在每次白名单项被命中后，系统会自动刷新其 TTL，以延长其有效期；如果某条白名单项在 TTL 过期前未再次命中，系统会自动将其删除，从而避免白名单长期失效项的累积，造成白名单冗余。

利用 redis 的 TTL 功能，每个白名单在创建时会获得一个超时的 TTL，当命中时会自动刷新 TTL 并增加白名单项的命中次数。当命中到达一定次数后，白名单将会获得更高的权重，其 TTL 也会相应的延长。每个权重对应的 TTL 和命中次数都可以在配置文件中设置。

2.3 告警过滤模块

2.3.1 模块概述

告警过滤模块的核心目的是减少告警的总量并对相似告警进行聚合，以优化大模型的处理效率。告警标准化子模块首先统一不同来源的告警日志格式，确保数据的一致性，为高效处理打下基础。随后，时间去重子模块通过识别并过滤短时间内的重复告警，显著减少了数据的量级。特殊类型日志处理子模块进一步筛选并剔除非攻击性或误报的告警，有效降低了无关告警的干扰。告警聚合子模块则通过基于 IP、攻击类型和告警载荷的聚合策略，将相似告警合并，形成更集中的告警信息，便于大模型快速且准确地处理。

告警过滤模块大致工作流程如图 2-3 所示。

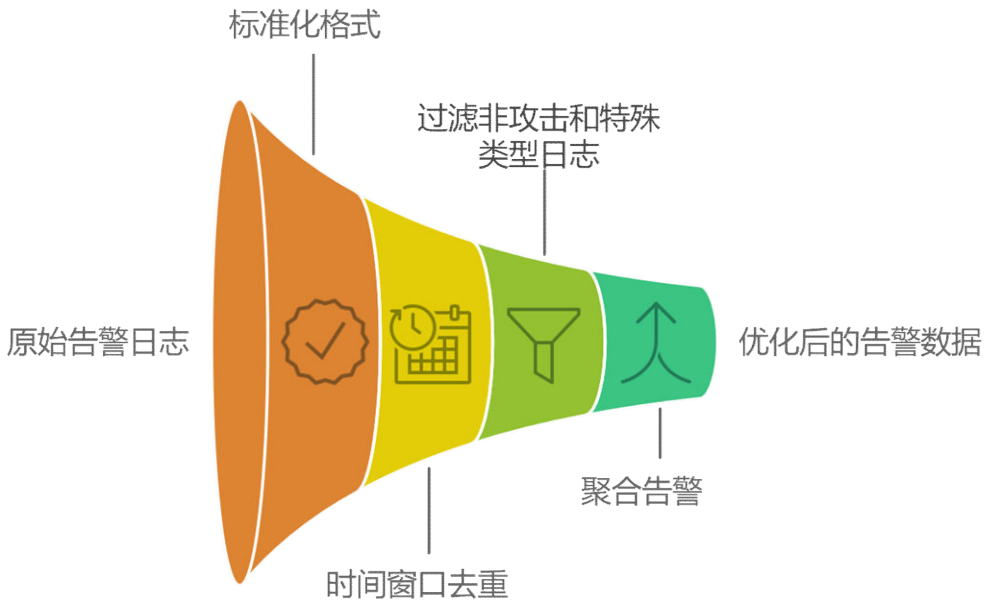

图 2-3 告警过滤模块工作流程

2.3.2 告警标准化

由于来自如 IDS、IPS、WAF 等不同安全设备的告警格式各不相同，告警标准化模块的主要是将这些多样化的告警日志统一转换成一种通用格式，以确保各类告警数据在后续的处理、分析和响应过程中能够被系统有效识别和利用，从而提升整个研判系统的效率和准确性。

在告警标准化模块的处理中，每个接收到的原始告警首先被转成一个标准化的告警格式。这一转换过程包括为每个告警分配一个标准化的名称，并将告警的各个属性按照预定义在标准化数据库中的属性映射关系，复制到新的标准化告警的相应字段中。标准化后的告警将包含一系列统一的属性，这些属性的详细列表和结构如表 2-1 所示。

表 2-1 标准化告警字段

字段名称	解释
log_type	日志类型，指示告警来源或种类
id	日志条目的唯一标识符
sip	源 IP 地址
dip	目标 IP 地址
sport	源端口号
dport	目标端口号
timestamp	告警产生的时间戳
attack_type	攻击类型，描述告警的攻击种类
attack_result	攻击结果，提供攻击是否成功的信息
severity	告警的严重程度
req_header	请求头数据
req_body	请求体数据
rsp_header	响应头数据
rsp_body	响应体数据
packet_data	网络数据包内容
threat_status	威胁状态，标记是否为实际威胁
filtered_stage	过滤阶段，告警在处理流程中的位置
related_alerts_ids	关联的告警 ID 列表，用于追踪相关告警
uri	统一资源标识符，标识请求的具体地址资源
xff	X-Forwarded-For 头信息，用于识别客户端原始 IP 地址

如果系统中只部署了一种类型的安全设备，其输出格式可以直接被定义为标准格式，从而消除了对标准化模块的需求。在这样的系统中，因为所有的告警已经是统一的格

式，额外的格式转换步骤就显得多余。然而，在一个多种安全设备并存的环境中，每种设备可能产生不同格式的数据，这时标准化模块就显得尤为重要。它确保了所有传入的告警都被转换为一个统一的格式，这对于系统中其他依赖标准化数据的模块是必需的。同时，标准化模块仅负责格式转换，并不对接收到的告警进行筛选，保证了数据的完整性和连贯性。

2.3.3 时间窗口去重

对于同一攻击，可能同时有多个检测设备检测到，触发多个告警。时间窗口去重模块的主要功能是减少告警日志中短时间内重复发生的同类告警所产生的冗余。这种去重操作有助于过滤掉短时间内相同来源和目标以及相同类型的重复告警，从而减轻后续处理模块的负担。

系统维护一个固定大小的时间窗口，持续监控并整理新到达的告警，并将输入的告警日志按时间戳进行排序，确保告警按时间顺序处理。在处理每条告警时，利用一个字典来存储每个唯一的告警标识（由源 IP、目标 IP、攻击类型、源端口、目标端口组成的五元组）及其最后出现的时间。遍历排序后的日志数据，对于每条记录，系统检查其是否在短时间内（例如 3 秒内）重复出现。如果是，该记录被标记为重复并从处理队列中去除；如果不是，它将被视为新的或持续的告警并保留。这个过程中使用了异常处理机制来确保程序在面对意外数据结构时不会崩溃。

时间窗口去重过程的伪代码如算法 2-1 所示：

算法 2-1：时间窗口去重算法

Input: Raw logs

Output: Logs after time-based deduplication

```
1: function TIME_DEDUPLICATE(data):
2:     SORT data by timestamp
3:     INIT empty list filtered_data and dictionary last_seen
4:     for each item in data do
5:         key ← (item.sip, item.dip, item.attack_type, item.sport, item.dport)
6:         current_time ← PARSE item.timestamp
```

```
        if key in last_seen and (current_time - last_seen[key]) ≤ 3 seconds then
            item.filtered_stage ← "Time-Deduplication"
            CONTINUE
        end if
        last_seen[key] ← current_time
        APPEND item to filtered_data
    end for
    RETURN filtered_data
end function
```

在我们的系统中，我们选取了 3 秒作为时间窗口的大小，并通过对去重效果的观察，采用启发式方法来确定这一值。整体来看，时间窗口去重所需满足的约束条件虽然显得较为严格，但其目的在于合并短时间内的重复告警，而非聚合载荷相似的告警日志（这一功能将在其他模块中实现）。

2.3.4 特殊类型和攻击成功日志筛选

特殊类型日志筛选致力于筛除那些不涉及实际安全威胁的信息性告警，以提高告警处理系统的效率和精确度。一些特定的行为虽然被系统标记为告警，但实际上是正常的业务活动。例如，“发现远程访问打印机服务行为”和“发现远程连接工具 ToDesk 活动”这类告警，虽然从安全系统的角度可能看似异常，但在企业日常操作中是合理且预期的行为。基于这些信息，在告警处理流程中实施了一个过滤机制，专门排除这些已确认为非安全威胁的业务相关告警。

攻击成功日志筛选的主要目的是识别并过滤出那些确实代表安全威胁的告警，以便专注于真正需要响应的事件。这一过程通过分析告警的特征和上下文信息来实现，确保只有在攻击被确认成功时才记录相应的日志。例如，某些告警可能指示攻击尝试的发生，但并不意味着攻击已成功。通过结合历史数据、攻击路径和防御机制的反馈，筛选机制能够有效区分潜在的误报与真实的安全事件，从而提高告警处理的精准性。我们使用正则表达式和告警字段的联合来判断攻击是否成功，表 2-2 是一些常用的正则表达式及其解释：

表 2-2 筛选攻击成功日志正则表达式示例

正则表达式	解释
(?i)success.*login.*	匹配包含 success 和 login 的日志，表示成功登录事件。
(?i)system compromised.*	匹配包含“system compromised”的日志，表示系统已被攻陷。
(?i)sql.*injection.*success.*	匹配包含“sql”、“injection”和“success”的日志，表明 SQL 注入成功。
(?i)remote.*access.*granted.*	匹配包含“remote access granted”的日志，表示远程访问被允许。
(?i)malicious activity.*detected.*	匹配包含“malicious activity”和“detected”的日志，表明检测到恶意活动。

2.3.5 告警聚合

告警聚合模块的核心目标是通过基于 IP、攻击类型和告警载荷的聚合策略，提升告警处理的效率与准确性。该模块专注于三种常见的攻击模式：多对一、一对多以及一对一。对于多对一场景，例如爆破攻击，当多个源 IP 同时攻击同一目标资产时，系统会将这些告警聚合为一条，避免重复处理。对于一对多场景，例如横向扫描攻击，攻击者使用一个源 IP 扫描多个目标资产，系统同样将此类告警聚合为一条，以反映攻击者的整体行为。最后，对于一对一的点对点攻击，系统将单独记录每一次源 IP 与目标 IP 之间的攻击，确保精确地跟踪每个攻击行为。

具体来说，其核心是两个映射结构：SiptoNDipMapping 和 DiptoNSipMapping，分别用来存储从源 IP 到目的 IP 以及从目的 IP 到源 IP 的映射关系。update_mappings 函数遍历所有告警条目，为每个唯一的源 IP 和目的 IP 对创建映射并记录相应的告警信息和攻击类型。通过这种映射关系，我们能够有效地聚合和分类告警，进而通过 filter_one_to_many、filter_many_to_one 和 filter_one_to_one 函数分别处理不同类型的攻击模式。每个聚合函数根据特定的聚合逻辑（如一个源 IP 关联多个目的 IP）收集并返回相应的映射结构。

依据告警三元组聚合的伪代码如算法 2-2 所示：

算法 2-2：告警三元组聚合

```
Input:
    Raw logs - List of log entries (SIP, DIP, attack type, etc.)
Output:
    one_to_many_mappings - SIP to multiple DIPs
    many_to_one_mappings - DIP to multiple SIPs
    one_to_one_mappings - SIP to one DIP and vice versa

function UPDATE_MAPPINGS(data, sip_to_dip_connections, dip_to_sip_connections):
    for each log in data do
        Update sip_to_dip_connections with SIP to DIP mapping
        Update dip_to_sip_connections with DIP to SIP mapping
    end for
    RETURN updated connections
end function

function FILTER_ONE_TO_MANY(sip_to_dip_connections):
    Filter mappings with SIP to multiple DIPs
    RETURN one_to_many_mappings
end function

function FILTER_MANY_TO_ONE(dip_to_sip_connections):
    Filter mappings with DIP to multiple SIPs
    RETURN many_to_one_mappings
end function

function FILTER_ONE_TO_ONE(sip_to_dip_connections, dip_to_sip_connections):
    Filter mappings with SIP to one DIP and DIP to one SIP
    RETURN one_to_one_mappings
end function
```

同时，由于一对一告警中涉及的告警种类繁多，每种类型的告警所包含的告警载荷类型也各不相同，因此我们决定进一步对这些告警的载荷进行聚合分析。在这一过程中，本项目采用了 OPTICS（Ordering Points To Identify the Clustering Structure）聚类算法来对告警数据进行深入分析。OPTICS 算法是一种鲁棒的聚类方法，特别适合处理不同密度的数据集，能够识别出基于密度的聚类结构，这对于复杂且异质性高的告警数据尤为适用。

具体实施上，我们首先使用 TfidfVectorizer 从每条告警的载荷中提取文本信息，并将其转换为 TF-IDF 矩阵。TF-IDF（Term Frequency-Inverse Document Frequency）

方法通过计算词语的出现频率与逆文档频率，为每个词语在告警集合中的重要性赋予权重。基于生成的 TF-IDF 向量，我们应用 OPTICS 聚类算法对告警数据进行聚类分析。OPTICS 算法通过对数据点的排序，能够在不明确指定簇数量的情况下，发现数据中的簇结构。我们选择使用余弦相似度作为聚类的相似度度量工具，余弦相似度通过计算两个向量之间的夹角，来判断两个告警内容的相似性。通过这种方式，系统能够有效地将内容相似的告警载荷聚合在一起，进一步提高告警处理的精度和效率。

OPTICS 告警聚类算法如 2-3 所示：

算法 2-3：OPTICS 聚类算法

```
Input: Alerts = {a1, a2, ..., an} (a list of alerts, each alert contains 'req_header' and 'req_body')
Output: Clusters of alerts based on content similarity

Initialize an empty list for combined_text

for each alert in Alerts do
    text = concatenate(alert.req_header, alert.req_body)
    Append text to combined_text
end for

Initialize TfidfVectorizer
tfidf_matrix = TfidfVectorizer.fit_transform(combined_text)

Initialize OPTICS Clustering
optics_model = OPTICS(min_samples=MinPts, max_eps=Eps, metric='cosine')
labels = optics_model.fit_predict(tfidf_matrix)

Create clusters
clusters = {}
for i in range(len(labels)) do
    cluster_label = labels[i]
    if cluster_label not in clusters then
        clusters[cluster_label] = []
    end if
    clusters[cluster_label].append(Alerts[i])
end for

Return clusters
```

2.3.6 小结

告警过滤模块通过一系列精细化的处理步骤，有效减少了告警数据的冗余性和无效性，为后续的分析和处理提供了高质量的输入。经过统一告警格式、去除重复告警、筛选非攻击性日志以及聚合相似告警，这一模块不仅显低了告警的总量，还提升了数据的集中度和相关性，使得大模型能够在更短的时间内做出更准确的判断和响应。

2.4 告警研判模块

2.4.1 模块概述

告警研判模块通过应用信息熵分析、自然语言处理及大模型技术，对收集到的告警数据进行深入分析，以识别攻击意图和潜在威胁。该模块包括信息熵评价模型、自定义规则评价模型和大模型语义评价模型三部分，通过对告警信息进行多维度的评估和加权合成，形成一个综合的评分体系。最终，通过设定动态调整的评分阈值，系统能够从海量告警中有效筛选出高风险的告警事件，提示安全团队进行进一步研判和响应。

告警研判模块大致工作流程如图 2-4 所示。

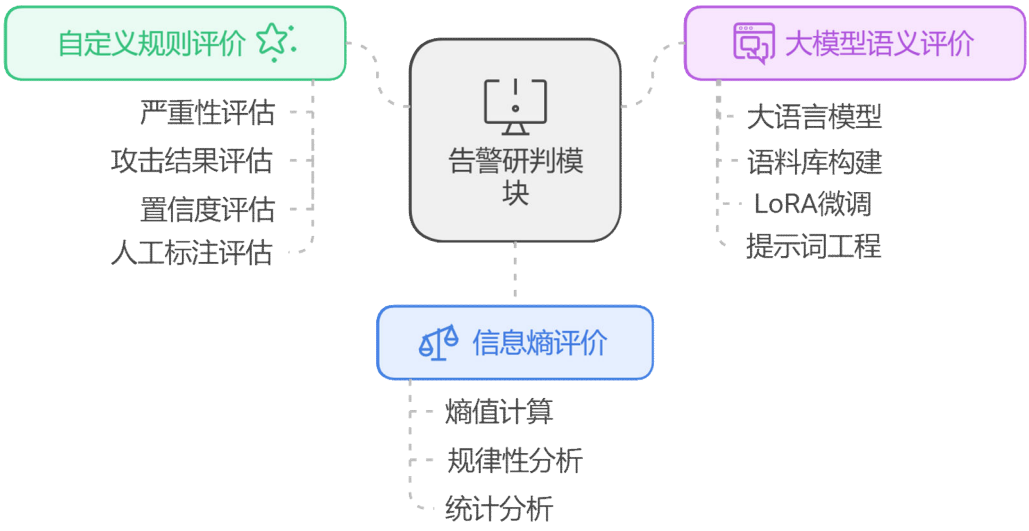

图 2-4 告警研判模块工作流程

2.4.2 信息熵评价模型

信息熵，最初由克劳德·香农在信息论中提出，用于描述信息的不确定性或随机性。信息熵的数学表达可以量化一个信息源产生的信息量的平均不确定度。信息熵越高，信息的不确定性越大；信息熵越低，信息的确定性越高。

在安全领域，信息熵的概念可以用来评估告警的规律性和随机性[16]。如果一个系统或网络中的告警活动非常规律，其信息熵值会较低，表示该系统的行为比较可预测。相反，如果告警活动不规律或者随机性高，其信息熵值会较高，这可能指示未知的安全威胁或异常行为。

在实施信息熵评价模型时，我们首先关注的是告警活动的规律性。例如，考虑一个内部主机，如果它每天产生的告警数量相对稳定，则其信息熵值低，可能意味着这些告警属于常规的系统操作。然而，如果该主机的告警数量在短时间内大幅波动，其信息熵值会显著提高，这可能暗示该主机正遭受攻击或存在其他安全风险。

为了具体实现这一评价，我们会对不同的告警特征进行统计分析，如攻击 IP 的稀有度和告警类型的分布。通过收集一定时间内（例如一小时内）与当前告警在源 IP、目标 IP 和告警类型等方面相似的告警序列，我们可以使用上述信息熵公式计算这些数据的熵值。高熵值表示告警数据高度分散，可能提示潜在的安全风险，而低熵值则表明数据较为集中，可能属于常规操作。

2.4.3 自定义规则评价模型

自定义规则评价模型是为了补充信息熵评分方法而设计的，目的是进一步提升安全告警的识别和评估精度。通过利用告警日志中的关键字段如“severity”（严重程度）、"attack_result"（攻击结果）、和“confidence”（置信度）等，该模型允许进行详尽的评分。告警的严重程度直接影响其分数，较高的严重度得到较高的评分。如果告警结果显示攻击成功，相应的分数会更高，反映出攻击成功对系统可能带来的实际影响。此外，告警的置信度也是一个重要的评分维度，高置信度的告警表明其背后的数据和分析更为可靠，因此获得更高的评分。

该模型还包括了人工标注的环节，安全团队可以根据实际经验对不同类型的告警日志进行得分标注，以此来细化和优化评分逻辑，从而形成一个更为精确的综合加权

评分。模型支持高度个性化的配置，企业可以根据自身业务的具体需要和安全关注的重点调整告警的权重。例如，企业如果在某段时间特别关注某一类攻击，可以增加这类告警的权重，或者对外网到内网的攻击设置更高的告警权重，以确保安全团队能够优先处理这些高风险事件。

2.4.4 大模型语义评价模型

大模型语义评价模型利用自然语言处理（NLP）技术，对告警的文本数据进行语义分析，识别潜在的攻击意图和策略。这一部分可以通过提示词工程、微调等技术提高大模型的识别和响应能力。

例如，某个 SQL 注入告警日志的 payload 可能是：GET /index.php?id=1' OR '1'='1，利用大模型技术，分析 payload 中的文本，基于模型的分析，确定这是否为一个恶意企图。在这个例子中，模型可能已经被训练来识别各种数据库攻击策略，根据攻击的潜在严重性，为这次告警分配一个分数。由于 SQL 注入可能导致数据库被完全控制，因此可能会分配一个高分。随着模型在实际环境中的不断应用，收集反馈并调整模型，以适应新的攻击模式和减少误报。

2.4.4.1 告警语料库的构建

如果仅使用单一类型的数据进行微调容易导致严重的过拟合现象，导致模型丧失其原有的泛化能力。为了解决这一问题，并增强模型的理解能力及适应性，决定引入丰富多样的数据资源。

首先，引入大规模的通用领域文本数据集 BELLE 1.5M[17]，该数据集涵盖了广泛的指令类型和多个领域的文本。通过这些通用领域数据，模型能够在更广泛的自然语言环境中进行训练，增强其理解和处理各种上下文信息的能力。

考虑到专业领域的需求，需要特别强化网络安全方面的数据收集和利用。安全领域数据集整合了来自安全书籍、知识库、研究论文和安全社区的文章，以及漏洞数据库中的内容，构建了一个综合的安全领域数据集，并使用大模型根据现有的文本数据自动生成 SFT 数据集，提高了模型在安全领域的专业能力和应用效果。

同时，为了克服实际应用中对显存和资源的限制，使得大模型能够在消费级显卡上运行，我们采用了知识蒸馏的方法来构建特定任务（如告警解读）的数据集。具体

来说，通过提示词工程[18]技术，我们引导大参数模型生成具有意义的回答，这些回答随后被用作训练小参数模型的数据集。知识蒸馏过程中，我们特别注重保留大模型处理复杂告警情况时的精确性和细节处理能力。通过这种方式，即使是参数相对较少的模型也能在安全告警研判任务中表现出色。

但是，由于上述方法无法完全保证 100%的准确性和深度，因此人工审核成为确保数据质量的必要步骤。在本项目中，引入了一个人工审核阶段，验证和校正大参数模型生成的标注数据，确保其准确性和一致性。

以 SQL 注入攻击为例[19]，审核的关键在于深入分析请求头和请求体中的内容。常见情况下，如果请求体包含看似正常的业务查询语句，例如因为使用了较旧版本的应用而将查询语句直接嵌入请求体内，虽然这可能触发安全设备的告警，通常认定这类告警为误报。这是因为该行为虽触发告警，但缺乏恶意的意图。相反，如果请求体中包含具有明显攻击特征的语句，如使用了 sleep 命令或 union 查询以尝试提升查询权限或进行数据泄露，这种情况下，结合其载荷内容的具体分析，会将此类告警判定为实际的攻击行为。

2.4.4.2 大模型选型与微调

本项目选定了百川智能推出的新一代开源大语言模型：Baichuan2-7b[20]用于微调和研判。该模型通过利用 2.6 万亿 Tokens 的高质量语料进行训练，在多种通用和专业领域 benchmark 上展现了的卓越性能。Baichuan 2 不仅在同等规模的模型中取得了最佳效果，其对中文的支持尤为出色。此外，该模型的显存资源占用较小，使其在资源受限的环境下仍能高效运行。考虑到这些优势，Baichuan2-7b 成为了安全告警数据微调的理想选择。

模型在小规模数据集上的训练和推理测试表明，使用 FP16 半精度进行运算时，Baichuan 2-7B 的 LoRA 微调[21]过程大约需要 23GB 的显存，而仅推理则需约 16GB 显存。这一显存需求使得该模型可以在消费级显卡如 NVIDIA RTX 3090 上进行微调和部署，无需依赖更高端的企业级硬件资源。

基准大模型、微调数据集都准备好后，进行大模型微调阶段。LoRA（Low Rank Adaptation）是一种微调大语言模型的技术，它允许在几乎不增加模型参数的情况下，有效地适应新的任务或数据集。这种方法特别适合于处理那些拥有巨大参数量的模

型，如 GPT-3 或 BERT 等，通过微调使得这些模型能够在特定任务上表现更好，而不需要对整个模型进行重训练。

模型层面，首先使用 transformers.AutoConfig 从预定义的路径加载 Baichuan 2-7B 模型的配置，并关闭缓存功能以优化性能。随后，通过 AutoModelForCausalLM 加载预训练的模型，并自动分配至可用的计算设备。对于 LoRA 的集成，首先检查是否提供了预训练的 LoRA 模型路径。如果未提供，则使用 find_all_linear_names 函数来确定模型中所有线性层的名称，这些层将被配置为 LoRA 层。随后，设置 LoRA 参数（包括秩和 α 值），并通过 get_peft_model 将 LoRA 集成到模型中。

数据层面，使用 AutoTokenizer 来处理数据：它将文本数据转换成令牌（tokens）[22]，随后这些令牌被转换为数字 ID 这是模型训练过程中使用的编码形式，同时为了满足模型输入的具体要求，将长度不一的输入序列截断或填充到固定长度。采用 DataLoader 进行数据的自动化批处理和优化。DataLoader 支持多线程操作以减少 I/O 等待时间，提高整体数据处理效率。此外，DataLoader 允许在每个训练周期开始时随机打乱数据顺序，并自动将数据组织成批次，每个批次包括一组输入和对应的标签。

训练过程设定了全局变量以跟踪损失值，并初始化了进度条来监控训练状态。通过批量处理数据，模型计算得到损失并进行反向传播[23]。为了提高内存使用效率，采用了梯度累积技术，这意味着模型参数在经过多个前向和反向传播步骤后才更新。训练过程中的损失值定期记录并通过图形化工具绘制损失曲线，这有助于追踪模型训练进度并及时发现潜在问题。此外，模型的检查点也按计划保存，允许在必要时恢复训练或回退到较优状态[24]。LoRA 微调的算法伪代码如 2-4 所示：

算法 2-4：LoRA 微调算法

```
1: Define training parameters such as max_position_embeddings, batch_size, lr, etc.
2: Define paths for pre-trained model and dataset
3: Initialize global structures for tracking training progress and loss

4: function prepare_data():
5:         Load data using SFT settings
6:     return data_engine

7: function prepare_model():
8:     Load pre-trained causal language model from specified path
9:     if using LoRA:
10:        Apply LoRA configurations to the model
```

```
    Enable model features like gradient checkpointing
    return model

function save_model(model, path):
    Save the model to the path

function train(model, data_engine):
    Set model to training mode
    for each epoch:
        for each data batch from data_engine:
            Perform model forward pass to compute loss
            Perform backward pass and update model parameters
            Log loss and periodically save the model
    Save final model and loss plot

Main Program:
    Setup tokenizer from pre-trained path
    data_engine ← prepare_data()
    model ← prepare_model()
    Display model trainable parameters
    Train model using data_engine
    Optionally, repeat training for multiple epochs
```

2.4.4.3 提示词工程

尽管通过微调，大模型在告警研判的效果已经有所提升，但企业的业务和安全环境是不断变化的。新的攻击方式和告警载荷形式的出现要求模型能够迅速适应这些变化。频繁地对模型进行微调不仅耗费资源，也可能影响到模型的部署和运行效率。为了应对这种快速变化的需求，引入了提示词工程技术，这可以在不频繁微调模型的前提下，快速适应新的业务和安全需求。

提示词工程是一种通过精心设计的提示，来激发和引导预训练的大模型生成特定输出的技术。在不直接修改模型内部参数的情况下，通过调整输入的方式来改变模型的行为。这涉及到对输入数据前缀或结构进行调整，使其包含特定的关键词或短语，这些关键词或短语是事先定义好的，能引导模型根据新的业务逻辑产生预期的答案。

在本项目中，为了提高告警研判的精确性和响应速度，我们通过人工研判精细化地总结了各种告警类型的显著业务特征，并将这些特征存储在一个配置文件中。在大模型进行告警研判前，会从此配置文件中提取相应的提示词，这些提示词随后被整合

到模型的输入数据中。这样，如果待研判的告警载荷与配置文件中的业务特征相匹配，模型则将其识别为正常的业务侧告警；若无任何匹配，则默认采取拒绝方式，将其视为疑似的安全攻击。配置文件的设计具有高度的灵活性和扩展性，允许用户根据实际业务的变化和需求，随时向文件中添加新的业务侧载荷内容。这种动态更新能力使得大模型能持续适应业务进展和新的安全挑战，无需进行频繁的模型重训练或大规模的微调。

2.4.5 综合评价与阈值设定

为了形成一个综合的评估结果，我们将这些独立的评分通过加权合成的方法整合到一起。在分配权值时，我们特别利用斯皮尔曼系数来确定每种评分方式的权重。斯皮尔曼系数，作为一种衡量两个变量等级相关性的非参数指标，能有效反映各评分方式与最终安全事件结果之间的相关强度[25]。我们可以根据斯皮尔曼系数的结果来调整权值，确保每种评分方式在整体安全分析中的重要性得到合理的体现。

我们还设定了一个阈值，这个阈值根据历史数据和实际安全需求进行动态调整。当综合评分超过这一阈值时，系统将标记此事件为潜在的安全事件，需要安全团队进行进一步的研判和响应。示意图如图 2-5 所示。

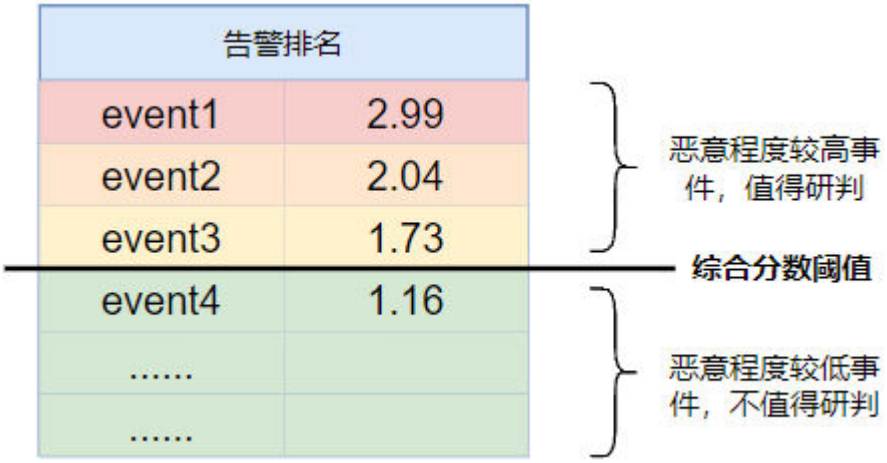

图 2-5 综合评价与阈值工作流程

2.5 前端展示模块

2.4.1 模块概述

在本项目中，前端展示模块提供了一个用户友好的交互界面，使得用户能够更方便的管理整个系统。该前端交互界面为企业的运营、运维和安全团队提供了高效的工具。通过整合多种功能（包括仪表盘、调查工具、聊天和上下文记忆管理等），用户可以更好地理解、分析和处理告警信息。该模块的框架如图 2-6 所示。

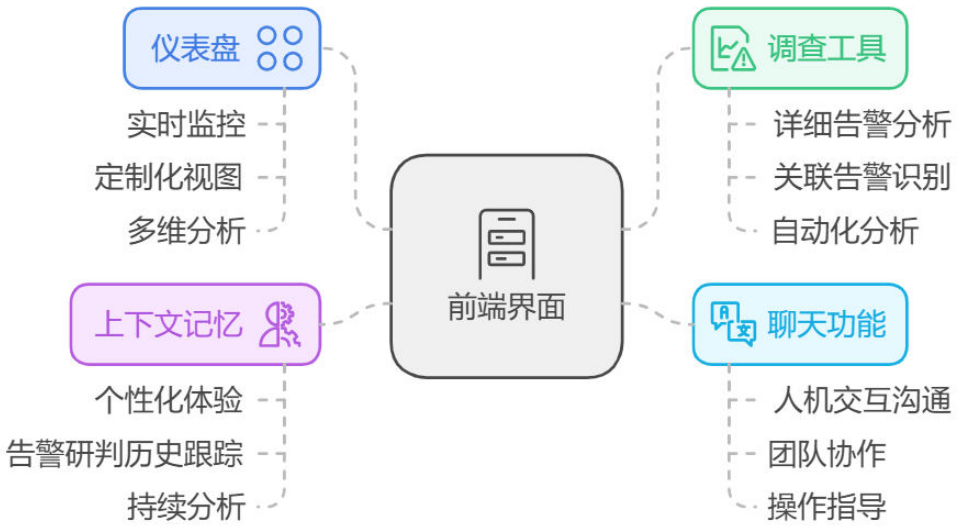

图 2-6 前端展示模块框架

仪表盘能够实时显示系统告警的整体情况，帮助用户快速发现潜在问题。通过图表、图形和关键性能指标（KPIs），用户可以直观地了解系统健康状况。

通过调查工具，用户可以进一步深入到具体告警的详细信息，查看告警源、历史记录和潜在影响范围。这有助于了解问题的严重性和关联性。

通过聊天界面，用户可以与大模型进行自然语言对话，快速获取告警的详细信息或询问建议。这样的交互方式降低了学习成本，尤其适合非技术用户。通过聊天界面，系统可以实时指导用户如何处理某个告警，提供具体的步骤，减少操作失误。

上下文记忆功能可以根据用户的历史操作、询问和告警研判记录，自动调整显示信息和提供更具针对性的建议。这大大提高了工作效率。

该交互界面将仪表盘、调查工具、聊天与上下文记忆结合，可以显著提升告警处理的效率与准确性。这不仅帮助用户从多个角度快速做出告警研判决策，还通过智能化、个性化功能，提升了整体用户体验。

2.4.2 技术架构

本项目采用 Nuxt.js 搭配 Tailwind CSS 和 FastAPI 来构建应用，使用了现代前后端分离的全栈架构方案。

Nuxt.js 是基于 Vue.js 的服务端渲染框架，它支持静态站点生成和服务器端渲染（SSR），提供了良好的 SEO 支持和页面加载性能。其强大的路由、页面生成、模块和插件系统简化了复杂前端应用的开发。

Tailwind CSS 是一个实用的 CSS 框架，它通过一系列的预定义类，让你以极简的方式构建响应式和现代的 UI。通过 Tailwind，可以快速实现高度可定制的设计，同时保持代码的整洁。

FastAPI 是一个基于 Python 的现代异步 Web 框架，因其高性能、简洁的代码结构和自动生成 API 文档（如 Swagger 和 Redoc）而广受欢迎。非常适合构建需要处理实时数据或高并发请求的 RESTful API。

本项目中使用 Nuxt.js + Tailwind CSS 前端和 FastAPI 后端架构，软件效率高、性能优越。Nuxt.js 提供自动路由和服务端渲染，提升 SEO 和页面加载速度，Tailwind CSS 快速构建响应式 UI，FastAPI 具备高效异步处理和自动 API 文档生成，确保后端可靠、可扩展。

第三章 作品测试与分析

3.1 测试方案

为了全面评估我们开发的系统在实际工作环境中的表现，我们设计了一套综合测试方案。该方案包括将程序部署到具有代表性的企业运营环境中，并在该环境中连续运行程序七天。这一时间长度能够覆盖企业的正常业务周期，包括高峰和非高峰时段，从而确保测试结果的全面性和实用性。

测试期间，我们将重点关注以下几个方面的系统表现：降噪模块的效果，包括其在告警削减数量和告警提升质量方面的能力；研判模块的准确性，特别是其识别复杂攻击模式的能力；以及前端展示的用户友好性，包括信息展示的实时性、准确性和用户交互的便捷性。此外，我们将监测系统的资源消耗（如 CPU 使用率、内存占用和网络带宽使用）来评估其对企业资源的影响，以及记录系统的稳定性表现，如崩溃和重启的次数及原因。系统将自动记录所有关键组件的性能数据和运行日志，供后续分析之用。

3.2 测试环境

硬件环境配置如下：

CPU：AMD EPYC 7713 64 核 128 线程 * 2
内存：1TB 128GB * 8
GPU：NVIDIA RTX 3090 24G * 2
系统：Linux ubuntu 5.15.0-79-generic

软件环境配置如下：

Python: 3.10
Torch: 2.2.0
Transformers: 4.40.1
Datasets: 2.19.1
Numpy: 1.24.4

Accelerate: 0.30.0

Peft: 0.10.0

Gradio: 3.37.0

3.3 测试数据

本次测试基于于某企业真实内部环境，对来自三家厂商的五种网络安全设备进行运营测试，包括奇安信天眼威胁监测与分析系统，中资网安网络流量监测系统和瑞树网络防火墙。

我们对一周内产生的告警进行了统计与测试，对五种日志分别命名为"tianyan"、"alarm-tianyan"、"zhongzi"、"v2zhongzi"、"waf"。具体统计结果如表 3-1 所示。

表 3-1 告警日志数量统计

日期	tianyan	alarm-tianyan	zhongzi	v2zhongzi	waf	总告警量
2024/8/25	674530	684808	1020	3288	5698	1369344
2024/8/24	700339	711035	1252	1951	8581	1423158
2024/8/23	798469	797666	7177	9334	24985	1637631
2024/8/22	933196	926687	14703	8037	17614	1900237
2024/8/21	871534	873506	28962	9498	21698	1805198
2024/8/20	904668	911445	30344	8493	21221	1876171
2024/8/19	887609	893140	23535	2436	21002	1827722

平均每日告警数量超过一百三十万，工作日告警数量明显高于周末数量。

3.4 测试结果分析

3.4.1 告警降噪效果

从 2024 年 8 月 19 日至 2024 年 8 月 25 日，我们对告警降噪模块进行了综合测试。结果如表 3-2 所示：

表 3-2 告警降噪结果统计

日期	告警日志数量	时间去除后告警数量	保留的元告警数量	聚合后告警数量	聚合率
2024.08.19	1763576	845914	440431	13437	0.9695
2024.08.20	1813354	872406	456582	12404	0.9728
2024.08.21	1742998	838954	442532	12385	0.9720
2024.08.22	1826783	865894	464225	12848	0.9723
2024.08.23	1568596	748152	369742	13532	0.9634
2024.08.24	1388118	656430	328415	5099	0.9845
2024.08.25	1338985	637626	310746	4154	0.9866

在 2024 年 8 月 19 日至 8 月 25 日的测试期间，告警日志数量在这一周内表现出波动。8 月 19 日的 ES 告警日志数量达到 176 万条，为测试期间的最高点，而在 8 月 25 日下降至 133 万条。尽管告警日志总量有较大变化，保留的原告警数量基本保持稳定，波动范围较小，从 440431 条（8 月 19 日）到 328154 条（8 月 24 日），这表明降噪模块能够有效筛选出需要进一步处理的告警。

经过降噪处理后的聚合告警数量也表现出显著的变化。19 日的聚合告警数量为 13437 条，到了 25 日则下降至 4154 条，展示了在不同日子里告警事件的变化趋势。同时，聚合率在这一周内略有波动，最低为 0.9596（8 月 26 日），最高为 0.9723（8 月 22 日），整体保持在一个较高的水平，表明告警降噪模块在这一时间段内能够高效地聚合相似告警，并大幅减少重复告警，保障系统的稳定性和准确性。

表 3-3 告警降噪结果统计

攻击类型	源告警数量	聚合告警数量	聚合率 (%)
发现使用远程连接工具向日葵	129370	1695	98.69
发现 Web 空口令登录行为	70869	383	99.46
Java 代码执行攻击(机器学习)	48995	1053	97.85
SMB 账号暴力猜解	8309	376	95.48
发现远程连接工具 ToDesk 通信流量	7970	410	94.86
发现域用户口令爆破行为	6903	387	94.39
IMAP 明文密码	6497	97	98.51
发现使用 SMB Create Request File 对 lsarpc 进行操作	6279	902	85.64
疑似 SSH 账号暴力猜解	4465	34	99.24

SQL 注入攻击	2877	898	68.78

为了评估告警降噪模块在处理大量告警中的效果，我们从 2024 年 8 月 19 日的日志中选取了几种出现频率较高的告警类型，并计算了它们的聚合率，结果如表 3-3 所示。总体来看，降噪系统在不同类型的告警上均展现了良好的聚合效果。例如，针对 Web 空口令登录行为的告警，聚合率达到了 99.46%，而像 SQL 注入攻击等复杂告警类型，聚合率也接近 70%，展现出系统在处理复杂攻击行为时的稳健性。

3.4.2 告警研判效果

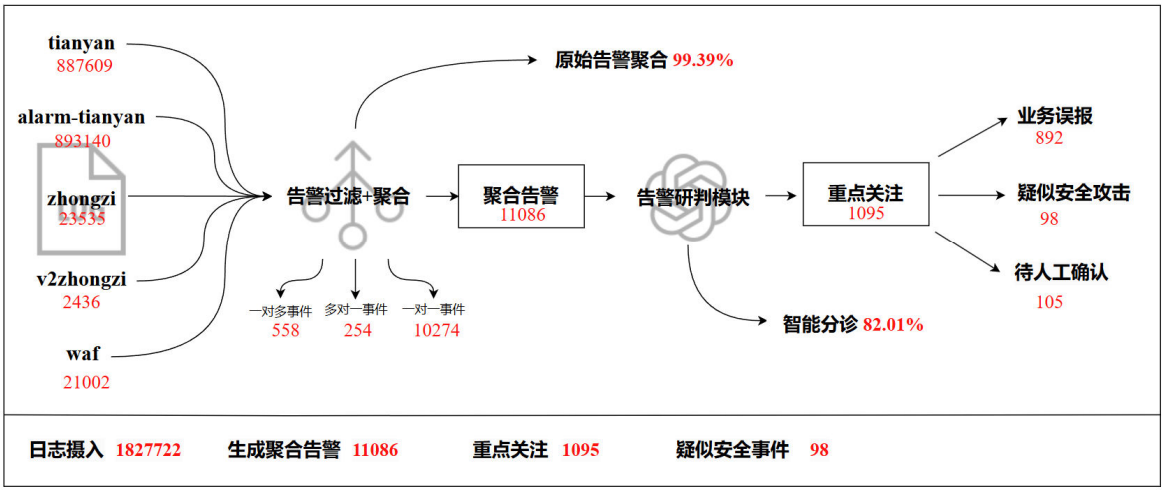

图 3-1 8 月 19 日告警研判结果

根据告警研判模块的流程图 3-1 所示，系统的整个告警处理流程从原始告警的筛选、聚合到最终的研判和分类，各个环节的效果和效率清晰地呈现出来。

首先，系统共处理了 1,827,722 条告警日志，经过初步的过滤与聚合后，告警数量大幅减少。通过聚合策略，系统将这些告警压缩至 11,086 条，聚合率高达 99.39%。这种高效的聚合处理表明，告警降噪模块能够有效识别相同或相似的告警并合并，大幅降低了需要人工处理的告警量。

聚合后的告警进一步进入告警研判模块。在这一阶段，系统通过智能分诊机制对告警进行进一步分类和处理，结果为：1,095 条告警被列为“重点关注”对象。这些告警中，892 条被判定为业务误报，表明这些事件在实际业务流程中属于正常行为，无需进一步关注。而剩余的 203 条告警中，98 条被识别为可能的安全攻击事件，另外

105 条告警则需要人工进一步确认。这一结果显示，智能分诊达到了 82.01%，有效减轻了人工研判的压力。

3.4.3 人工交互效果

为了提升告警处理系统的可用性和效率，我们开发了一个专用的 Web 界面，以简化安全人员的日常操作流程和研判工作。该界面分为四个主要部分，分别为仪表盘、调查界面、聊天界面和上下文记忆模块。

首先，仪表盘部分展示了告警的整体情况。通过图表和统计信息，安全团队可以直观地查看告警的总体数量、告警分类以及当前系统的运行状态。调查界面则主要用于展示告警列表，并提供了研判接口。在这里，安全人员可以查看每一条告警的详细信息，并通过内置的研判工具进行深入分析。该界面还支持与系统中的其他模块互动，便于安全人员直接在平台上进行告警研判，极大地提高了研判的效率和准确性。聊天界面为安全人员提供了与大模型进行交互的功能。在处理复杂告警时，安全人员可以通过该界面向大模型提问或提供上下文，借助大模型的能力辅助决策。这种实时交互不仅增强了系统的智能化水平，还减轻了安全人员的工作负担，特别是在需要大量数据分析和判断的场景下。最后，上下文记忆模块用于存储一些业务侧的模板信息。其设计目的是帮助安全人员更好地处理与业务相关的告警。在上下文记忆模块中，系统会保存一些与业务逻辑相关的告警模板，并在研判过程中将这些模板作为提示词输入大模型进行分析。如果告警匹配了预先存储的模板，则系统会自动判定其为误报，无需人工干预。这一功能有效减少了误报数量，提升了系统整体的处理效率。

3.4.3.1 仪表盘界面

图 3-2 仪表盘界面

仪表盘界面是系统的核心展示部分，主要用于对告警数据进行汇总和可视化分析，方便安全人员快速掌握系统的整体运行状况，如图 3-2 所示。整个界面分为四个主要部分：

左上角展示系统已过滤和研判的告警总数。这一部分能够让用户一目了然地看到系统的处理进度和当前告警数量，帮助快速了解告警的总体规模。右上角的曲线图显示了不同类型告警在一天中的分布情况。不同颜色的曲线分别表示 tianyan、alarm-tianyan、zhongzi、v2zhongzi 和 waf 类型的告警，帮助用户分析某个时间段内不同告警的集中度，从而识别告警高峰期或异常时间点。左下角展示了告警频率排名前十的事件，以柱状图的形式显示每种告警的频率。右下角的饼图和分类列表显示了告警的类型和比例，包括如 SQL 注入、目录遍历、配置错误等各种告警的具体分布情况。这一部分通过详细的数据分类帮助用户了解系统面临的主要威胁类型，便于制定针对性的安全策略。

3.4.3.2 调查界面

图 3-3 调查主界面

调查界面主要呈现一个详细的告警列表，帮助安全人员快速查看和处理每一条告警。该列表包含了告警的所有重要信息，例如告警名称、源 IP 地址（sip）、目标 IP 地址（dip）、聚合的告警数量、告警发生时间、告警的日志来源等，如图 3-3 所示。

在界面的右侧，每条告警均配有“处理”按钮，安全人员可以根据告警的研判结果进行处理或进一步操作。此外，列表中的告警状态（如“待处理”）和研判结论（如“业务误报”）也会实时更新，便于安全团队优先处理那些确认为实际威胁的告警。

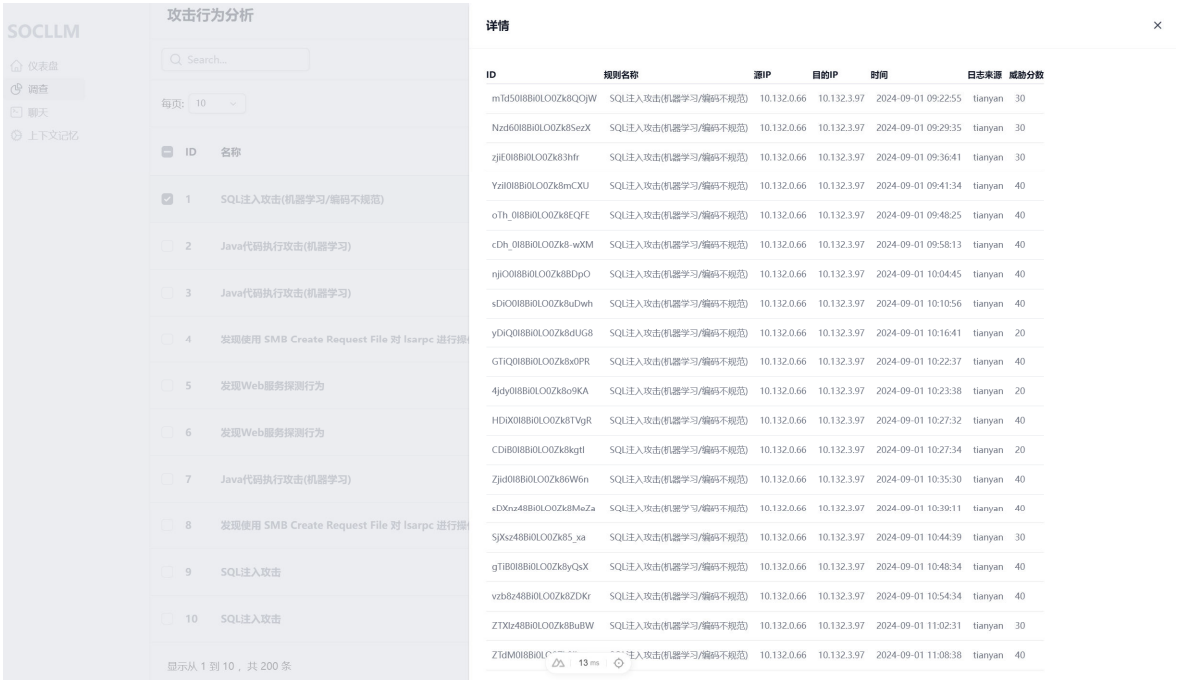

图 3-4 聚合告警详情界面

在调查界面中，安全人员可以点击某个聚合告警的名称，从而展开一个详细信息菜单。该菜单展示了聚合告警背后的具体组成告警，列出了所有原始告警的详细信息，包括原始告警的 ID、源 IP、目标 IP、告警发生时间、日志来源等，如图 3-4 所示。

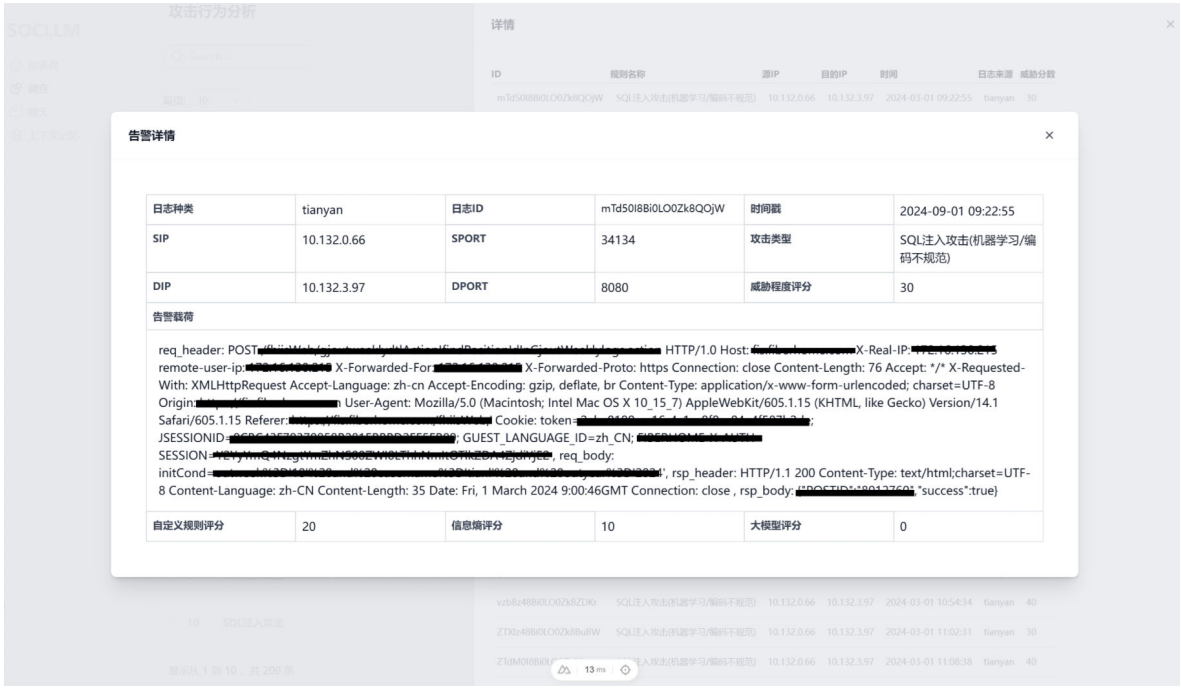

图3-5 告警详情界面

在调查界面中，点击某个原始告警的 ID 后，系统会弹出一个详细信息窗口，展示该告警的完整数据，图 3-6 所示。此窗口包含了该条原始告警的所有关键信息，如日志类型、源 IP（SIP）、目标 IP（DIP）、端口号、时间戳、攻击类型和威胁程度评分等，如图 3-5 所示。此外，告警的请求头（req_header）、请求体（req_body）、响应头（rsp_header）等详细载荷信息也会在该界面中完整展示，帮助安全人员进行更深入的分析。

在窗口的底部，系统还提供了三个模型对该条告警的评分，包括自定义规则评分、信息指示评分和大模型评分。每个评分模块根据告警的特征、内容和匹配的规则，对告警进行独立分析并打分。这一评分机制为安全人员提供了多维度的研判依据。

3.4.3.3 聊天界面

聊天界面为安全人员提供了与大模型直接交互的便捷渠道，帮助他们在处理复杂告警时作出更精准的决策。通过这一界面，安全人员可以向大模型提问，提供上下文

信息，或者请求进一步的告警分析。在告警处理过程中，聊天界面支持多种提问方式，包括告警解读、告警查询和安全知识问答等。

安全知识问答：对于不熟悉的攻击手段或安全术语，安全人员可以在聊天界面中提问，大模型将提供相关的安全知识解答，帮助安全团队快速提升对网络威胁的理解。

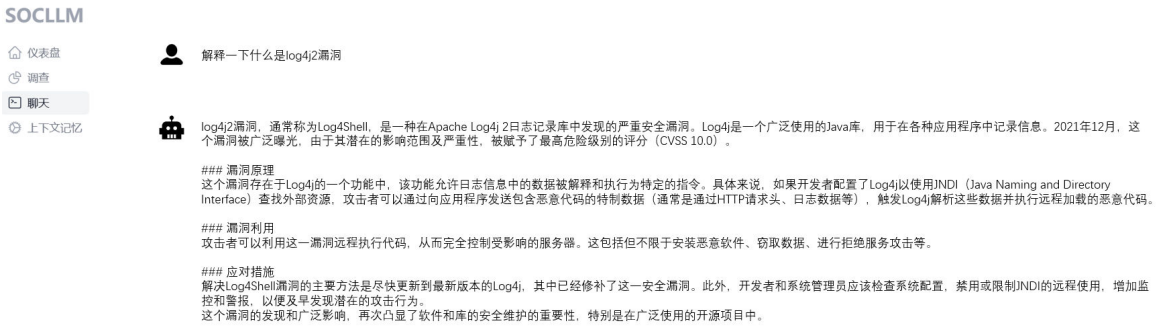

图3-6 安全知识问答

告警查询：可以快速查询某个特定告警的详细信息，或回溯历史告警，以便分析类似事件的处理方法。

图3-7 告警查询

告警解读：安全人员可以通过该界面请大模型解释某条告警的具体含义，帮助理解告警的背景和潜在风险。

图3-8 告警解读

3.4.3.4 上下文记忆模块

图3-9 上下文记忆界面

上下文记忆模块的设计目的是帮助大模型更好地识别业务侧告警，并避免将其误判为真实攻击。在实际的研判过程中，如果出现大模型未能识别的新型业务侧告警类型，而被误判为攻击事件，安全人员可以通过上下文记忆界面对这些业务告警进行描述和补充。

具体而言，安全人员可以在上下文记忆模块中为这些新型业务侧告警输入自然语言描述，详细说明该告警的特征和业务场景。这些描述信息将被保存并在后续的告警处理流程中作为提示词发送给大模型。当相似的告警再次出现时，这些提示词会帮助大模型更好地理解该告警属于业务活动，而非攻击行为，从而做出更准确的判断。

第四章 创新性说明

4.1 研判对象的创新

本项目在研判对象上具备创新。现有的告警大模型主要集中在终端侧告警的分析上，这类告警通常具有相对明确的语义特征，比如命令提示符（cmd）等，借助大模型对自然语言的理解能力和交互处理能力，能够有效地辅助运维人员进行告警分析和研判。

而本项目的创新点在于专注于网络侧告警的研判。网络侧告警的语义通常更加抽象且复杂，具有高度的动态性和多样性。相较于终端侧，网络侧的告警数量庞大，远超现有大模型的负荷处理能力。这些告警分布于多个设备，且告警信息往往缺乏明确的语义提示，增加了分析的难度。

为应对这些挑战，本项目采用了一系列创新措施来优化网络侧告警的处理流程。首先，通过告警过滤和威胁评分机制，有效剔除大量误报漏报和无关的低危告警，仅保留高威胁度的告警交由大模型进一步处理。其次，项目通过微调大模型，增强其在处理网络侧告警中的理解能力。与此同时，项目为不同告警类型构建了独立的模板库，帮助大模型在处理多来源、多类型告警时具有更强的适应性和分析深度。这种创新设计，极大提高了大模型在处理大量复杂网络侧告警时的效率与准确性，突破了现有告警模型的局限。

4.2 跨厂商兼容性与多源告警分析创新

本项目在跨厂商兼容性方面具备创新。当前大多数安全运维系统仅限于厂商自有生态的设备告警处理，而现代企业网络环境中通常使用来自多个厂商的安全设备，这种多样化增加了运维复杂程度。

本项目突破这一限制，通过引入统一的告警格式和处理流程，支持整合不同厂商、不同平台的网络安全设备和系统的告警数据。项目的告警研判模块能够无缝接入多源告警信息，兼顾各种设备日志和网络流量，进行全面分析。通过这种跨平台、跨厂商的高兼容性设计，项目不仅适用于单一厂商的网络安全环境，还能适应复杂、多样的

企业网络架构，提供全面的安全分析和预警。

4.3 智能化辅助与易用性创新

本项目在智能化辅助与易用性方面具备创新。传统的告警分析常常依赖专家经验，处理速度慢且容易错过关键威胁。

本项目采用了创新性的大模型研判机制，能够在极大简化运维流程的同时，提升告警处理效率。通过智能化告警过滤和优先级排序技术，自动筛选出最具威胁的告警，减少了无关或低优先级告警的干扰。结合自学习和白名单结果反馈机制，系统能够动态调整告警研判策略，适应不断变化的业务场景和威胁环境。此外，项目特别注重用户体验的优化，提供了简单直观的界面，运维人员只需要具备一定的安全能力也能轻松使用。这种智能化和易用性创新，极大提高了网络侧告警处理的准确性和响应速度，为企业节省了运维成本，同时保障了高效的网络安全运维。

第五章 总结

随着网络安全威胁的日益复杂，现有网络安全运维方案在处理海量告警和实现高效研判时面临诸多挑战。为此，本团队开发了一种基于大语言模型的网络安全运维系统，旨在通过智能化的告警过滤和研判机制，提高运维效率和准确性。

首先分析了当前安全领域大模型应用的现状，指出了现有大模型应用在网络侧告警处理、可扩展性和泛用性方面的不足。针对这些问题，我们设计并实现了一个新型的网络安全运维系统，该系统具备三大核心功能：一是高效的告警过滤，包括告警降噪和聚合；二是智能的告警研判，能够通过多维度方法对告警进行深度分析，识别出高威胁告警；三是大模型的辅助研判，利用大模型在理解告警语言的优势和人工交互的能力辅助运维人员进行告警研判，显著提升告警研判效率。

本系统在真实企业内网环境下试运行，成功验证了这种基于大模型的网络安全运维系统在处理真实网络环境中的有效性和高效性。不仅填补了大模型在网络安全运维领域应用的空白，还为行业提供了一种创新的解决方案，助力提升整体安全防护能力。

综上所述，本研究为网络安全运维提供了新的视角和方法，不仅在理论上推动了大模型的应用探索，更在实践中为安全管理带来了切实可行的工具。未来的工作将继续探索大模型在网络侧告警运维上的更多潜力。

参考文献

[1]赵彬, 王亚弟, 徐宁, 等. 网络安全运营中心关键技术研究[J]. 计算机工程与设计, 2009 (9): 2117-2120.

[2]彭祯方,邢国强,陈兴跃.人工智能在网络安全领域的应用及技术综述[J].信息安全研究,2022,8(02):110-116.

[3]白雪,王鸿元.大语言模型在网络安全领域的应用探索[J].电信工程技术与标准化,2023,36(12):23-30.DOI:10.13992/j.cnki.tetas.2023.12.008.

[4]Ferrag M A, Ndhlovu M, Tihanyi N, et al. Revolutionizing cyber threat detection with large language models. arXiv preprint arXiv:2306.14263

[5]Sultana M, Taylor A, Li L, et al. Towards Evaluation and Understanding of Large Language Models for Cyber Operation Automation//2023 IEEE Conference on Communications and Network Security (CNS). IEEE, 2023: 1-6.23: 1-6.

[6]BBayer M, Kuehn P, Shanehsaz R, et al. Cysecbert: A domain-adapted language model for the cybersecurity domain. ACM Transactions on Privacy and Security, 2024, 27(2): 1-20.

[7]Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[8]Liu Y, Ott M, Goyal N, et al. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[9]Sanh V, Debut L, Chaumond J, et al. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019.

[10]Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners. OpenAI blog, 2019, 1(8): 9.

[11]Black S, Biderman S, Hallahan E, et al. Gpt-neox-20b: An open-source autoregressive language model. arXiv preprint arXiv:2204.06745, 2022.

[12]Karlsen E, Luo X, Zincir-Heywood N, et al. Benchmarking Large Language Models for Log Analysis, Security, and Interpretation. arXiv preprint arXiv:2311.14519, 2023.

[13]Austin Zhao. 大模型在网络安全领域的应用市场洞察. IDC Perspective, 2023(12). 文档编号：#CHC51403423.

[14]Freitas S, Kalajdjieski J, Gharib A, et al. AI-Driven Guided Response for Security Operation Centers with Microsoft Copilot for Security[J]. arXiv preprint arXiv:2407.09017, 2024.

[15]李建华.网络空间威胁情报感知、共享与分析技术综述[J] .网络与信息安全学报，2016, Vol. 2(2): 16-29. （样例，参考国标GB/T7714-2015）

[16]张婷婷.基于信息熵的网络入侵检测系统设计[J].数字通信世界,2024,(08):92-94.

[17]Ji Y, Deng Y, Gong Y, et al. Belle: Be everyone's large language model engine[J]. 2023.

[18]Sahoo P, Singh A K, Saha S, et al. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. arXiv preprint arXiv:2402.07927, 2024.

[19]Halfond W G, Viegas J, Orso A. A classification of SQL-injection attacks and countermeasures//Proceedings of the IEEE international symposium on secure software engineering. Piscataway, NJ: IEEE, 2006, 1: 13-15.

[20]Yang A, Xiao B, Wang B, et al. Baichuan 2: Open large-scale language models. arXiv preprint arXiv:2309.10305, 2023.

[21]Hu E J, Shen Y, Wallis P, et al. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.

[22]Ali M, Fromm M, Thellmann K, et al. Tokenizer Choice For LLM Training: Negligible or Crucial?[J]. arXiv preprint arXiv:2310.08754, 2023.

[23]Zhou X, Zhang W, Chen Z, et al. Efficient neural network training via forward and backward propagation sparsification[J]. Advances in neural information processing systems, 2021, 34: 15216-15229.

[24]Di S, Robert Y, Vivien F, et al. Toward an optimal online checkpoint solution under a two-level HPC checkpoint model[J]. IEEE Transactions on parallel and distributed systems, 2016, 28(1): 244-259.

[25]Sedgwick P. Spearman's rank correlation coefficient[J]. Bmj, 2014, 349.

[26]Ji Y, Deng Y, Gong Y, et al. Exploring the impact of instruction data scaling on large

language models: An empirical study on real-world use cases. arXiv preprint arXiv:2303.14742, 2023.
[27] Addington S. ChatGPT: cyber security threats and countermeasures. Available at SSRN 4425678, 2023.