flink-ml-headers-v3 -重构设计文档

# 背景

**需求：**

现有的 CC Log 自动防护系统是一个融合成一个 JAR 包的单体应用，这在维护、功能迭代和部署灵活性方面带来了挑战。当前迫切的需求是将该项目解耦为多个独立的模块（JAR 包）。具体而言，客户要求：

1. 项目需拆解成 两到三个 独立的模块（JAR 包），这些模块必须能够独立部署在腾讯 Oceanus 平台上。
2. 个别独立部署的模块支持独立的手动参数设置。例如，客户能够针对某个策略分析模块自由设置“超4倍ccQPS阈值和DropAIG量占比>4%”等业务参数，而无需重新打包 JAR 。
3. 日志部分字段未标明含义，不清楚不理解。Jar包需求哪些入参数以及这些入参数是什么含义。
4. 用户增多，系统效率不够？

**目标：**

本次重构的主要目标是：

1. 将 CC Log 智能防护系统重构为清晰的 “分析模块 + 执行模块”两模块模式 。确保每个模块都是一个可独立部署的 Flink 作业（Job），能够在 Oceanus 平台上独立运行并调整资源 。
2. 实现模块间的职责分离和低耦合，**智能分析模块（**Analysis Module**）**专注于复杂的计算和分析（特征提取、QPS 计算、基线学习、CC 攻击检测、智研就监控），**策略执行模块**（Execution Module）负责将分析结果转化为具体防御行动（策略收敛、过滤、格式化、下发防御指令）。
3. 提供灵活的运行时参数配置能力，允许客户在 Oceanus 部署时为不同模块独立设置业务阈值等关键参数，提升系统运维的灵活性和响应速度 。

**验收标准：**

项目成功拆分为 cc-log-feature-processing-job.jar（智能分析模块）和 cc-log-policy-decision-job.jar（策略执行模块）两个独立的 Flink Job JAR 包。两个模块均可在腾讯 Oceanus 平台上独立上传、部署和启动，且能够正常处理数据流。

# 整体架构

## 逻辑架构

图示

AI 生成的内容可能不正确。

## 需求点评估

1. 需求点：将现有单体 JAR 包解耦为独立的 Flink Job 模块

**满足情况：** 本方案将现有的单体 JAR 包拆分为两个独立的 Flink Job 模块，每个模块可在腾讯 Oceanus 平台上独立部署和运行。

* **智能分析模块 (cc-log-analysis.jar)**：负责数据接入、特征提取、基线管理及初步攻击检测。
* **策略执行模块 (cc-log-execution.jar)**：负责消费分析结果、进行策略的收敛、过滤及最终的防护指令下发。
* 每个模块都将构建为独立的 Maven 项目，最终生成各自的 Fat JAR 包，实现了系统层面的职责清晰分离。

本方案设计的两个模块均符合标准的 Flink Job 结构。它们各自拥有独立的 main 方法和 Flink DataStream API 逻辑，可以直接作为独立的 Flink Job JAR 包上传至腾讯 Oceanus 平台进行部署和运行。Oceanus 平台将为每个 Job 分配独立的计算资源，实现相互隔离、独立启停和扩缩容。

1. 需求点：部分模块支持运行时通过外部参数进行灵活配置。

**满足情况：** 本方案充分利用 Flink 的 ParameterTool 机制来支持运行时参数配置。

* 在每个JAR模块的启动类通过ParameterTool.fromArgs(args) 方法解析命令行参数。
* 这些参数（如 Kafka Broker 地址、Topic 名称、各种业务阈值等）可以在 Oceanus 平台上部署 Job 时通过 UI 界面或 API 传入，无需修改代码或重新打包 JAR。
* **示例参数满足情况：**
  + 超 QPS阈值倍数和Drop、ALG 量占比阈值自定义配置：此参数将在**智能分析模块**中进行配置和使用，用于控制是否为恶意攻击的阈值。

1. 需求点：智能分析模块的核心职责。

**满足情况：**

* **消费原始 CC 攻击日志：** 模块通过 Flink Kafka Source 连接到指定的原始日志 Kafka Topic，持续消费数据。
* **实时特征提取、计算 QPS、聚合分析：** 模块内部实现数据解析、清洗，并利用 Flink 的窗口机制，对原始流量进行实时聚合，计算出 QPS、Drop 量、ALG 量等多种维度和时间粒度的特征。
* **动态学习业务流量基线：** 模块负责 Host 基线和 Header 基线的实时更新和维护，通过窗口计算将当前基线叠加到历史基线上，并将基线数据持久化到外部存储（如 ClickHouse/Redis）。
* **进行初步的 CC 攻击检测：** 模块会综合运用计算出的 QPS、Drop 量、AIG 量等特征，并根据配置的 cc\_qps\_threshold\_multiplier 参数以及读取标准基线模型，判断是否存在初步的 CC 攻击行为。
* **从攻击流量中提炼异常 Heade特征：** 一旦检测到攻击，模块会进一步分析异常流量中的 Header 特征，为下游策略生成提供依据。
* **输出分析出的攻击特征（SchemaPolicyHeader）到中间 Kafka Topic：** 模块将初步检测并分析出的攻击特征（封装为 SchemaPolicyHeader 对象）序列化后，发送到 policy-intermediate-topic，作为策略执行模块的输入。
* **全流程监控告警**（智研监控+企业微信告警）

1. 需求点：策略执行模块的核心职责。

**满足情况：**

* **消费上游策略 (policy-intermediate-topic)：** 模块通过 Flink Kafka Source 消费智能分析模块产生的 SchemaPolicyHeader 策略数据流。
* **对策略进行收敛（防抖动）：** 实现时间窗口内的策略去重和收敛逻辑，避免短时间内重复下发相同或类似的防御指令，减少对防护系统的冲击。
* **根据mysql 白名单、cc主机配置库等规则过滤策略：** 模块可集成与外部数据库（Redis）交互的逻辑，查询配置的白名单或其他过滤规则，对收到的策略进行过滤，确保合法的流量或特定业务场景不受影响。
* **格式化策略为下游防护系统可识别的命令：** 将内部的 SchemaPolicyHeader 对象转换为外部防护系统（如宙斯盾）所要求的特定命令格式（例如 JSON 或 Protobuf 消息体）。
* **将最终防御指令下发到防护系统 Kafka Topic：** 模块将格式化后的防御指令发送到指定的防护系统 Kafka Topic，供下游系统消费并执行具体的防护动作（如 IP 阻断、Session 阻断）。
* **中间源辅助策略下发：**将策略发送到中间源系统的Kafka，额外防护层，只在海外EO和国内EO业务中启用，用于在中间源系统也部署相同的防护策略。
* **记录策略流水：**将详细的执行流水记录（策略内容、是否成功下发、不下发的原因）等信息下发到监控topic。
* **智研监控：**通过TCP下发心跳监控给智研。

# 处理流程

图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

# 接口与数据结构

## 原始CC日志（for智能分析模块）

智能分析模块消费来自上游Agent采集并推送到Kafka的原始CC日志。

数据源类型: Kafka Topic: cclog\_topic (或根据不同业务环境配置)

消息格式: JSON字符串

## 外部配置（for智能分析模块）

类型: MySQL数据库

更新机制: Flink Source采用定时轮询机制（HostBaseLineConfigPollSource），周期性拉取最新配置，并通过BroadcastState广播给下游所有处理单元。默认轮询周期为600秒。

## 智能分析模块（for策略执行模块）

智能分析模块完成异常特征的分析后，会将结果封装成SchemaPolicyHeader对象，序列化为JSON格式并推送到Kafka，供策略执行模块消费。

类型: Kafka Topic: policy\_topic (或根据不同业务环境配置)

消息格式: JSON字符串

## 外部配置（for策略执行模块）

策略执行模块在下发最终的防御策略前，会从外部MySQL数据库加载过滤规则，以实现更精细化的策略管控，防止误拦截。

4.4.1 数据源

类型: MySQL数据库

更新机制: 在PolicyFilterFlatMapFunction中，通过定时判断（lastUpdateTimeStamp）来触发重载，更新周期为10秒。

4.4.2 数据结构

1. CC防护主机列表

功能: 定义哪些主机开启了智能CC防护，以及其防护模式。只有在该列表中的主机，其策略才会被下发到宙斯盾系统。

表名: tb\_cc\_host\_config (示例)

表结构:

| 字段名 | 类型 | 描述 |

| :--- | :--- | :--- |

| host | VARCHAR(128) | [主键] 域名 |

| mode | VARCHAR(32) | 防护模式 (如: 'drop', 'observe') |

| is\_enable | TINYINT | 是否启用 (1: 启用, 0: 禁用) |

2. 策略下发过滤规则表

功能: 提供黑名单机制，用于临时或永久性地阻止某些特定策略的下发。

表名: tb\_policy\_host\_filter\_config (示例)

表结构:

| 字段名 | 类型 | 描述 |

| :--- | :--- | :--- |

| host | VARCHAR(128) | [主键] 域名或泛域名 (如: \*.tencent.com) |

| filter | VARCHAR(256) | [主键] 过滤规则 |

| expire\_time | DATETIME | 规则过期时间 |

filter字段规则说明:

\*: 屏蔽该host下的所有策略。

header\_name: 屏蔽该host下针对header\_name这个维度的所有策略。

header\_name=header\_value: 精确屏蔽该host下header\_name维度值为header\_value的策略。

## 策略执行模块（for下游防御系统）

策略执行模块最终会生成文本格式的防御策略，通过不同的Kafka Topic下发给不同的下游防御系统。

4.5.1 下发至宙斯盾系统

适用业务: 所有业务 (国内EO, 海外EO, 大禹高防等)。

Kafka Topic: zsd\_policy\_topic (示例)。

策略特点: 经过了PolicyFilterFlatMapFunction中定义的完整且严格的过滤逻辑。

策略格式:

Apply to RuntimePolic...

=

action: drop

label: 为空

4.5.2 下发至中间源系统

适用业务: 仅海外EO、国内EO。

Kafka Topic: mid\_policy\_topic (示例)。

策略特点: 仅经过基本的策略收敛（CCPolicyProcessFunction）和长度校验，过滤规则相对宽松，旨在快速响应。

策略格式:

Apply to RuntimePolic...]

action: drop (实际在代码中通过label字段区分)

label: mid\_guard\_drop

## 4.6 策略执行模块（for监控/审计系统）

为了保证系统的可观测性和可追溯性，策略执行模块会将策略的生成和下发行为推送到多个监控和审计系统。

4.6.1 策略流水（Kafka -> ClickHouse）

功能: 记录每一条生成（无论是否被过滤）的策略详情，用于事后审计和分析。

Kafka Topic: header\_monitor\_topic (示例)

数据结构 (CCPolicyMonitor):

| 字段名 | 类型 | 描述 |

| :--- | :--- | :--- |

| wc\_timestamp | Long | 事件时间戳 (秒) |

| host | String | 域名 |

| policy\_id | String | 策略ID |

| header\_name | String | Header名称 |

| level | String | 风险等级 |

| header\_value | String | Header值 |

| policy\_text | String | 完整的策略文本 |

| is\_set | Boolean | 策略是否被下发 (true: 下发, false: 被过滤) |

| title | String | 策略的处理结果标题 (如: "set policy", "skip policy by ...") |

# 规格

## 性能

性能规格

|  |  |  |
| --- | --- | --- |
| 性能指标 | 智能分析模块 | 策略执行模块 |
| 端到端处理延迟 (End-to-End Latency) | 从消费一条原始日志，到分析出 SchemaPolicyHeader 并发送至 Kafka 的平均耗时。 | 从接收到 SchemaPolicyHeader，到处理完成并发送最终防御规则到 Kafka 的平均耗t时。 |
| 攻击检测延迟 | 符合攻击模式的流量开始注入，到CCDetectService首次将攻击状态识别为AttackBegin的时间间隔。它衡量了核心检测算法的灵敏度。 |  |
| 基线更新延迟 | 从无攻击状态切换到正常流量模式，到AdaptiveLineService完成对新流量模式的学习并更新相关基线状态的时间。它关系到系统在攻击结束后恢复正常基线的速度，影响后续检测的准确性。 |  |
| Checkpoint性能 | 系统完成一次Checkpoint所需的时间和该次Checkpoint的大小。Checkpoint是Flink容错机制的核心，其性能直接影响系统的故障恢复速度和对正常处理流程的影响。 | 系统完成一次Checkpoint所需的时间和该次Checkpoint的大小。Checkpoint是Flink容错机制的核心，其性能直接影响系统的故障恢复速度和对正常处理流程的影响。 |
| 作业启动/恢复时间 | 从作业失败到自动拉起并恢复处理能力的时间。 | 从作业失败到自动拉起并恢复处理能力的时间。 |

## 容量

容量规格

容量规格定义了在满足性能指标的前提下，单个部署实例能够承载的业务规模。当业务量超过容量规格时，应通过增加 Flink 作业的并行度 (Parallelism) 来进行水平扩展。

|  |  |
| --- | --- |
| 容量指标 | 描述与重要性 |
| 单核处理能力 (QPS/Core) | 平均一个CPU核心（Core）能够处理的原始日志QPS。这是进行CPU资源规划最简单有效的模型。 |
| 可支持的最大并行度 | 系统在横向扩展时，所能有效利用的最大Flink算子并行度（Parallelism）。超过这个并行度后，由于数据倾斜、网络开销或协调成本，系统的整体性能不再提升甚至下降。 |
| Kafka Topic 分区数 | 为保证数据处理的并行性，输入 Topic 的分区数应 大于或等于 Flink 作业的并行度。 |

# 容灾

1. 检查点 (Checkpointing)
2. 状态后端与持久化存储
3. 重启策略
4. 高可用 (High Availability)

# 依赖（外部平台，jdk，flink框架， my sql redis kafka）

# 兼容性设计

新增功能对原有功能的影响

代码优化、功能优化对原有系统的兼容性设计

数据结构、接口设计的兼容性

新增功能是否有开关控制

# 风险

本方案可能存在的风险项，如稳定性、兼容性、性能、容量……

# 监控相关（1.询问天穹 如果一个jar包里某个节点异常怎么检测。 2.代码做努力？比如计算超过P\_500这个功能异常，怎么监控排查）

涉及功能异常的监控点，上报链路

# 运营相关（很重要，上线后效果如何量化数据，我的成果）

涉及功能的打点数据

上线后的效果和量化数据

# 任务分解

|  |  |  |  |
| --- | --- | --- | --- |
| 任务 | 明细 | 模块 | 工作量评估-开发、自测、联调 |
| 1. 智能分析模块重构 | 将核心业务逻辑（攻击检测、基线学习、特征分析）从Flink算子中剥离，重构为独立的、可单独测试的服务。 | core/MainTrainProcessFunction.java  core/\*Service.java  RuntimeTrainStream.java | 10、5、3 |
| 2. 策略执行模块重构 | 策略执行模块负责消费分析结果、进行策略的收敛、过滤及最终的防护指令下发，重构为一个可执行的JAR包。 | RuntimePolicyStream.java  core/PolicyFilterFlatMapFunction.java | 9、4、4 |

# 遗留问题

本次设计未能解决的遗留问题，需明确时间和责任人。

# 概要用例。 （随便写几个测试用例？）

简要描述正常及以异常用例。