

Обработка многострочных событий на примере AuditD в KUMA

Официальный способ получения multiline auditd - через переключатель "auditd", который доступен в KUMA 3.2: <https://support.kaspersky.com/help/KUMA/3.2/ru-RU/220739.htm>

Описанный ниже способ является wa и примером обработки многострочных событий на примере AuditD и может быть перенесен на другие источники с похожей структурой событий.

Введение

В данной статье будет рассмотрен процесс (**workaround**) обработки многострочных событий на примере событий AuditD.

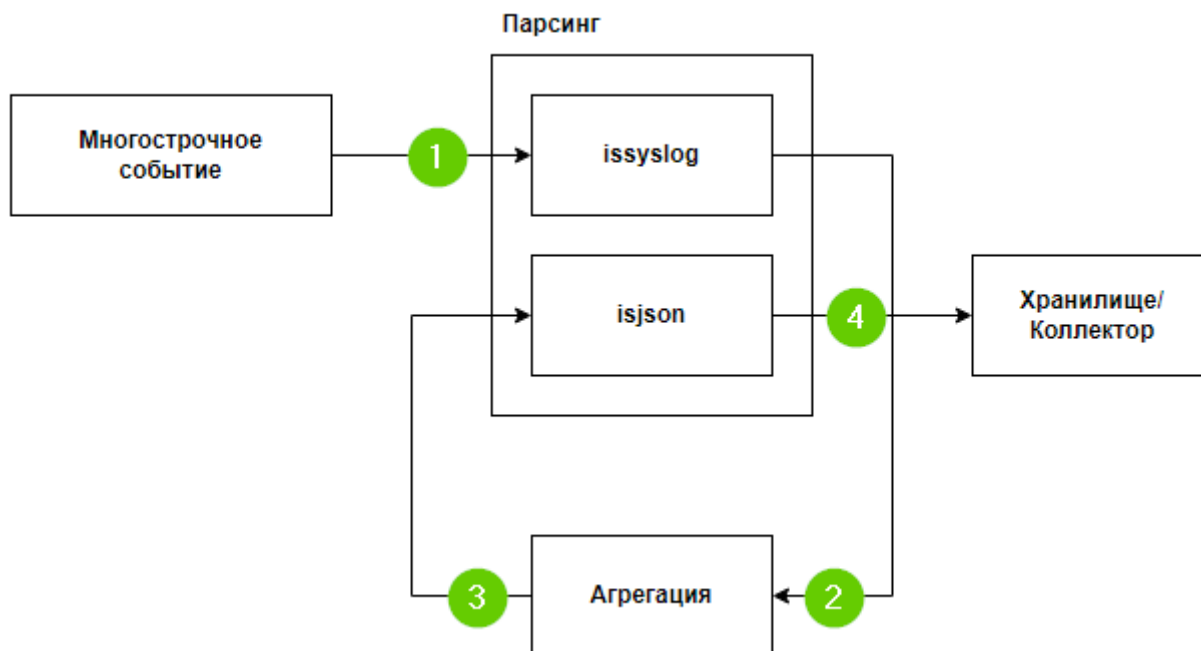
- 1) Каждая пачка событий, попавшая под агрегацию будет посчитана как 2 события (не 2 пачки). Так что в общем случае, когда в многострочном событии 2 и более событий увеличение числа EPS не произойдет.
- 2) Метод не подразумевает сохранение исходных событий (**Raw**), при необходимости собирать сырые события нужно будет создать дополнительный коллектор, который будет за это отвечать
- 3) После склейки, многострочное событие записывается в одно поле (в примере далее это **Extra и Message**). Соответственно потребуется использование измененных нормализаторов.

Коробочный парсер в данном случае не подходит, но его можно использовать за основу.

Так как алгоритм работы циклический, крайне рекомендуется прочитать данную статью два раза.

Принцип работы

Ниже представлена схема работы склейки многострочного события.



Основная идея заключается в том, чтобы на коллекторе сагрегировать все события одного многострочного события в одно. Вся полезную информацию при агрегации записать в одно поле и затем отправить такое агрегированное событие обратно на вход коллектору для нормализации.

Алгоритм работы на примере Auditd

1) На первом шаге в коллектор поступает событие, где происходит проверка, был ли отправлен syslog или json, в условиях экстранормализации, соответственно, выполняется проверка формата сообщения.

В общем виде, проверка может быть любой. Основная идея - разделить на главном нормализаторе исходные события от источника от агрегированных событий.

Пример нормализации.

```
^(?P<all>(?P<issyslog>\<).+)
```

Нормализация*

```
^(?P<all>(?P<isjson>\{.\}+)
```

📄 Перенести названия полей в таблицу

+ Добавить регулярное выражение

Пример условий для передачи событий в экстранормализатор, если событие исходное.

Дополнительный парсинг событий

Условия дополнительной нормализации

Схема нормализации

Обогащение

Использовать сырое событие*

Да

Параметры фильтра

И

+ Добавить условие

+ Добавить группу

issyslog



=



<

Пример условий для передачи событий в экстранормализатор, если событие агрегированное.

Дополнительный парсинг событий

Условия дополнительной нормализации

Схема нормализации ●

Обогащение

Использовать сырое событие*

Да



Параметры фильтра

И

+ Добавить условие

+ Добавить группу

isjson



=



{

2) В случае отправки syslog'a на этапе нормализации не будет происходить обогащения полями **DeviceProduct** и **DeviceVendor**, поэтому их можно будет добавить в условие для агрегации. Ниже приведен пример для агрегации событий AuditD:

Здесь также проверка может быть любой. Основная идея - под правило агрегации должно попадать только исходное событие. Попадание под правила агрегации должно быть исключено для событий, попавших в коллектор второй раз. Также в поле суммы должно быть указано то поле, в которое на этапе парсинга пишутся полезные данные.

Правило агрегации

AuditD Multiline

Предел событий

20

Время ожидания событий*

2

Группирующие поля*

DeviceProcessName

DeviceFacility

DeviceHostName

DeviceAddress

DeviceEventClassID

DeviceProcessID

Severity

DeviceReceiptTime

DeviceExternalID

Уникальные поля

Поля суммы

Message

Параметры фильтра

Фильтр

Создать

☐ Сохранить фильтр

Конструктор

</> Код

И

+ Добавить условие

+ Добавить группу

Если

е: DeviceProduct

=

<Пустая строка>

Если

е: DeviceVendor

=

<Пустая строка>

3) На этапе маршрутизации есть две точки назначения: 1. **Storage for AuditD** - хранилище и **loop** - перенаправка событий обратно в коллектор (сам в себя). Во время этого этапа происходит проверка типа событий **loop** - для **Type=2** (агрегированные события) и **Storage for AuditD** для **Type=1** (базовые события). Обратите внимание, что при создании точки **loop**, нужно будет указать выходной формат **json**:

Основная идея - разграничение точек назначения по типу событий. Обратно на вход коллектора должны подаваться агрегированные события, а в хранилище должны записываться только повторно обработанные события.

Пример точки назначения для отправки агрегированных событий в тот же коллектор

Создание точки назначения

Основные параметры

Дополнительные параметры

Точка назначения

Создать



Название*

loop


Состояние



Тип*

tcp



URL* 

127.0.0.1:6689

+ Добавить

Пример условия для точки назначения для отправки агрегированных событий в тот же коллектор

Создание точки назначения

×

Основные параметры

Дополнительные параметры

Время ожидания ⓘ	0
Размер дискового буфера ⓘ	0
Интервал очистки буфера ⓘ	0
Обработчики	0
Выходной формат	json
Режим TLS	Выключено
Сжатие	Выключено
Политика выбора URL ⓘ	Любой
Разделитель	
Дисковый буфер	<input checked="" type="checkbox"/>
Отладка	<input type="checkbox"/>

Параметры фильтра

Фильтр	Создать
	<input type="checkbox"/> Сохранить фильтр

Конструктор

</> Код

И

+

Добавить условие

+

Добавить группу

Если e: Type = 2

×

4) После отправки агрегированного события на вход тому же коллектору, коллектор получает json, где уже происходит обогащение, а также вынос информации из Message в Extra. Это событие уже не будет считаться агрегированным из-за чего будет отправка в другую точку назначения.

В общем случае, поле, в которое помещается информация из нескольких событий может быть любым. Также стоит учитывать максимально возможную длину полей.

Название*

Audit Message

Метод парсинга* ⓘ

kv

Разделитель пар*

Разделитель значений*

=

Сохранить дополнительные поля*

Да

+ Загрузить из файла

Заключение

Метод приведенный выше позволяет склеивать и нормализовывать многострочные события, у которых есть общий идентификатор (id, timestamp или комбинация таких полей), по которому можно однозначно определить принадлежность каждого конкретного события к многострочному.

Также, для простоты администрирования и нормализации можно использовать 2 коллектора вместо одного:

- Первый коллектор агрегирует события, записывая всю полезную информацию в одно или несколько полей KUMA и отправляет на вход второму коллектору без отправки в Хранилище/Коррелятор.
- Второй коллектор парсит полученные агрегированные события от первого и направляет нормализованные события в Хранилище/Коррелятор.

Полезные ссылки

1. Пример нормализатора и правила агрегации из примера: https://github.com/KUMA-Community/kuma_auditd_multiline_wa/
2. Принцип работы правил агрегации (схематично): <https://kb.kuma-community.ru/books/sozdanie-parserov-v-kuma-cookbook/page/princip-raboty-pravila-agregacii-schematicno>

