??????? ????? ? ??????? ?????? (Data Mining)

В отличие от потоковой корреляции, работающей в режиме реального времени, Data Mining правила позволяют с помощью языка SQL и функций ClickHouse (примеры запросов, почти все возможно использовать) распознавать и анализировать события, сохраненных в хранилище KUMA (можно указать и конкретный спейс хранилища).

??????? ??????

Выполнение **SQL-запросов к ClickHouse** и приведение результатов к формату нормализованных событий KUMA происходит на уровне **Core** с помощью новой сущности **DataMiningRule** и встроенного механизма **Scheduler**. Полученные результаты преобразуются в события и распространяются по корреляторам через стандартный API, который также используется коллекторами.

Важно, что:

- **запрос к ClickHouse выполняется строго один раз** за указанный период расписания;
- сформированный результат может быть направлен **одному или нескольким корреляторам**, а также в другие подсистемы в будущем;
- корреляторы, получающие данные, **могут принадлежать различным тенантам**, что обеспечивает гибкость и масштабируемость архитектуры.

Таким образом, Data Mining правила открывают возможность выявлять долгие и сложные цепочки активности - те, которые невозможно или крайне трудно обнаружить только средствами потоковой корреляции.

???????????? ? ??????????

Плюсы	Минусы
Снижение нагрузки на корреляторы — отсутствует необходимость хранить большие объёмы временных данных в оперативной памяти	**Увеличение нагрузки на хранилище данных**, для которого постоянные сложные запросы не являются целевой нагрузкой.
Кросстенантное обнаружение — правило может передавать результаты нескольким корреляторам разных тенантов.	**Риск тяжелых запросов** — неэффективный SQL может существенно нагрузить кластер.

Гибкость создания правил — возможность строить корреляцию напрямую на основе SQL-запросов.	**Отложенное обнаружение** — аналитика работает постфактум, поэтому алерт приходит позже, чем при потоковой корреляции.
Распределённое выполнение запросов — нагрузка обрабатывается кластером хранилища, а не одним сервером корреляции.	
Поддержка поиска аномалий и долгих сценариев атак — отклонения от нормы, тренды, девиации, редкие последовательности.	
Устойчивость к задержкам и несинхронности событий — если события приходят с опозданием или в неправильном порядке (например, правила по Golden Ticket), анализ всё равно будет корректным.	
Сохранность состояния при рестарте — бакеты и промежуточные данные не сбрасываются при перезагрузке коррелятора.	

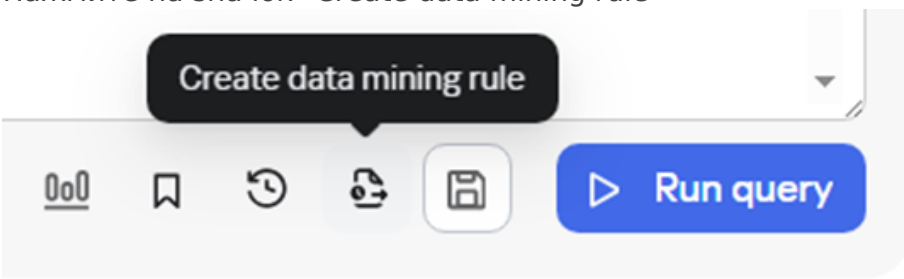
???????? ? ????????? ???????

Процесс создания правила можно разделить на три этапа:

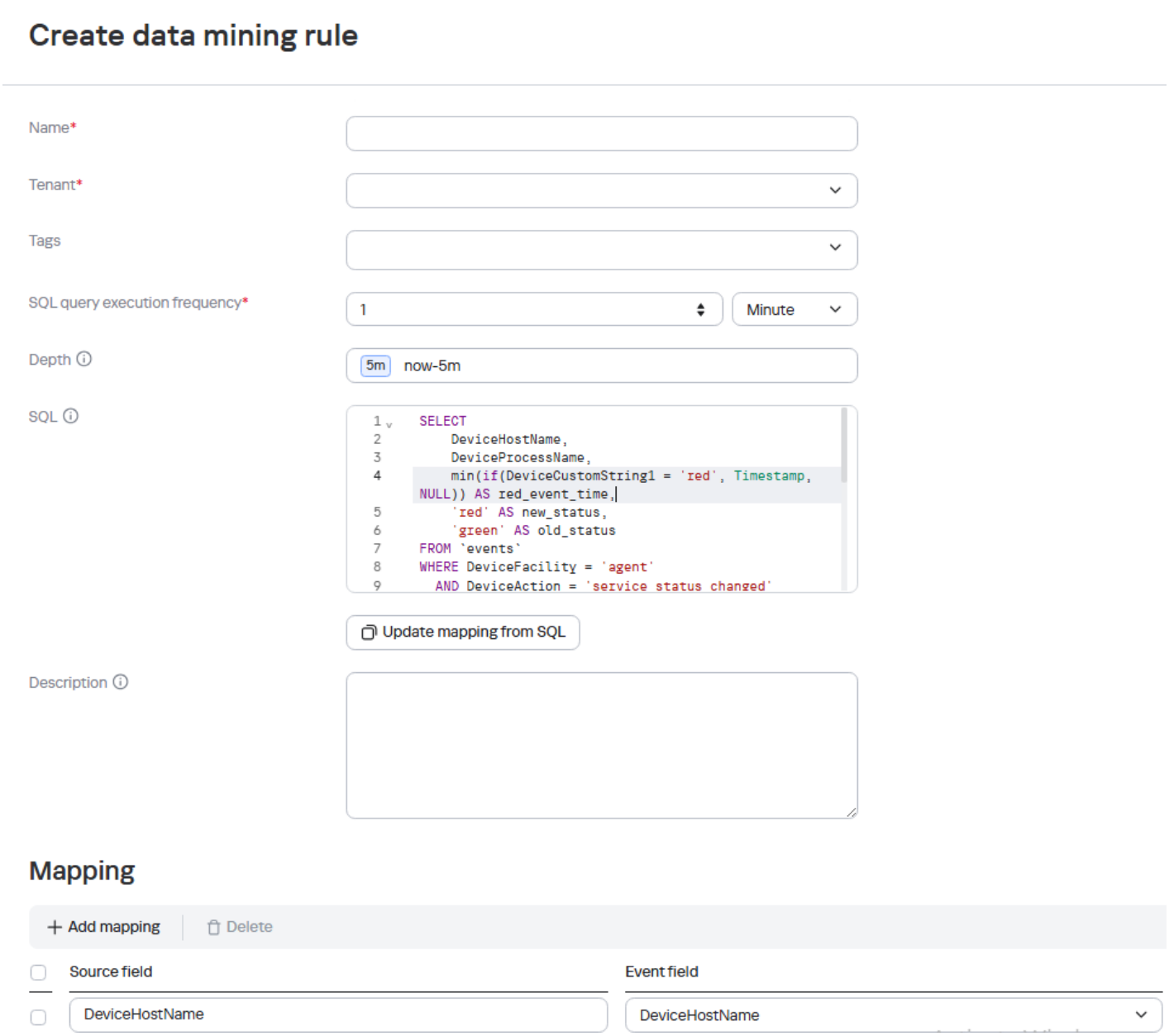
1 ????. ???????? ?????????????? ?????? Data Mining ???????

Создать правило можно двумя способами (из SQL-запроса в разделе Поиск по событиям):

- Формируем SQL-запрос в разделе Поиска по событиям (тестируем гипотезы, проводим атаку на полигоне, наполняем БД синтетическими событиями)
- Необходимо проверить что запрос выполняется, не вешает базу, возвращает осмысленный результат, который можно далее анализировать с помощью коррелятора
- Нажмите на значок "Create data mining rule"



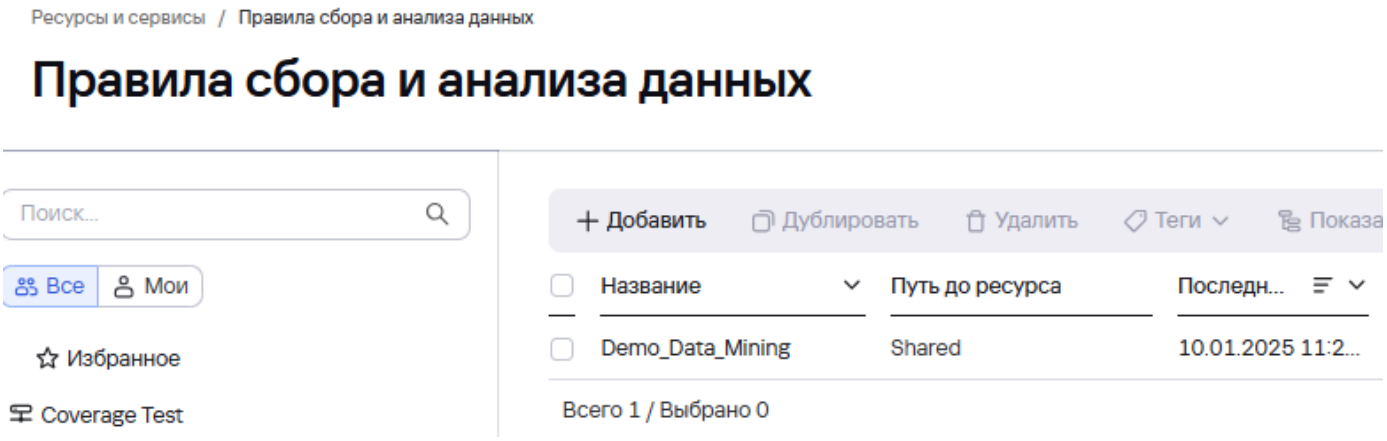
- Далее открывается окно Создания правила, автоматически заполнится сам запрос, глубина и частота запуска. Заполнится маппинг полей из запроса в поля KUMA



- Здесь необходимо вписать название, выбрать тенант, дозаполнить поля и создать правило.

Второй способ (создать правило как ресурс):

- В **Ресурсах - Правила сбора** и анализа данных Создать правило



- В правиле указать:
  - **Интервал (частота) выполнения SQL-запроса** можно указать в минутах, часах и днях (минимум 1 минута)
  - **SQL-запрос** должен содержать функцию агрегации (примеры) и/или группировку (GROUP BY) данных с обязательным указанием ограничения LIMIT (от 1 до 10 000)

> Каждое выполнение такого правила происходит в виде запроса в Хранилище, а это значит неосторожным движением в виде частого или тяжелого правила можно нагрузить базу больше чем хотелось бы

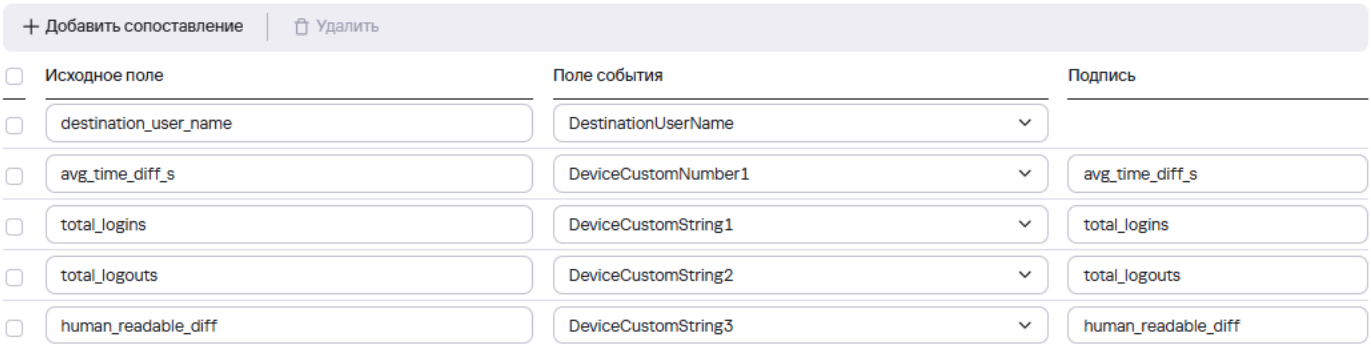
В примере рассматривается запрос на основе событий Windows по пользователям (DestinationUserName) событиям входа (EventID 4624) и выхода (EventID 4634) с расчетом среднего времени сесии пользователя за последние 24 часа.

Посмотреть SQL запрос (пример)

```
SELECT
    login_events.DestinationUserName AS destination_user_name,
    round(AVG(logout_events.logout_time - login_events.login_time)/1000) AS
avg_time_diff_s,
    COUNT(DISTINCT login_events.login_time) AS total_logins,
    COUNT(DISTINCT logout_events.logout_time) AS total_logouts,
    concat(
        toString(floor(avg_time_diff_s / 86400)), ' days, ',
        toString(floor((avg_time_diff_s % 86400) / 3600)), ' hours, ',
        toString(floor((avg_time_diff_s % 3600) / 60)), ' minutes, ',
        toString(avg_time_diff_s % 60), ' seconds'
    ) AS human_readable_diff
```

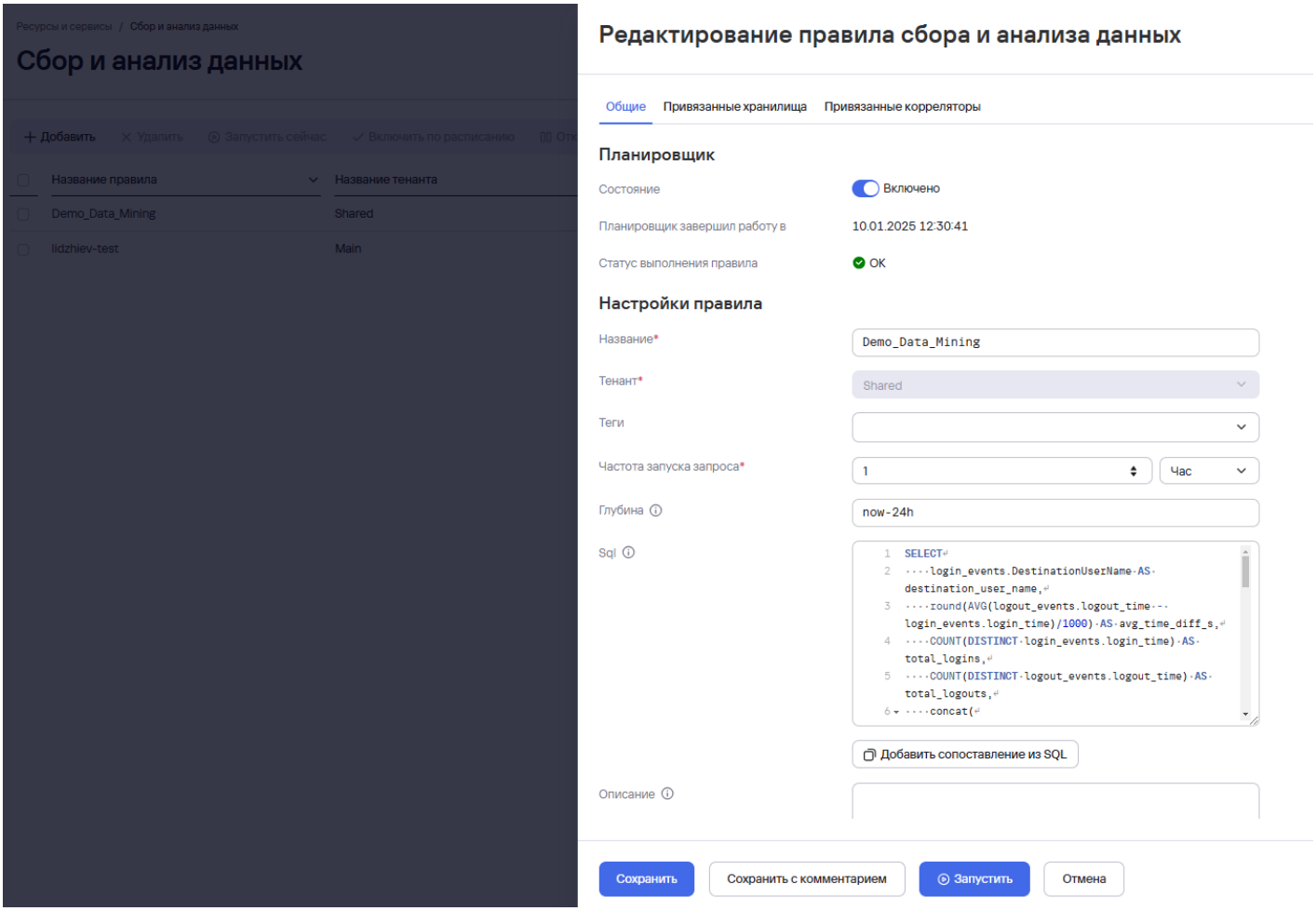
```
FROM
    (SELECT
        DestinationUserName,
        toUnixTimestamp(EndTime) AS login_time,
        FlexString1 AS logon_id
    FROM `events`
    WHERE DeviceEventClassID = '4624'
    AND EndTime >= now() - INTERVAL 24 HOUR
    AND DestinationUserName NOT LIKE '%$%') AS login_events
INNER JOIN
    (SELECT
        DestinationUserName,
        toUnixTimestamp(EndTime) AS logout_time,
        FlexString1 AS logon_id
    FROM `events`
    WHERE DeviceEventClassID = '4634'
    AND EndTime >= now() - INTERVAL 24 HOUR
    AND DestinationUserName NOT LIKE '%$%') AS logout_events

    ON login_events.DestinationUserName = logout_events.DestinationUserName
    AND logout_events.logon_id = login_events.logon_id

WHERE logout_events.logout_time >= login_events.login_time
GROUP BY login_events.DestinationUserName
ORDER BY avg_time_diff_s DESC
LIMIT 100
```

- **Добавить маппинг** (сопоставление) по полям запроса и модели KUMA



2 ????. ???????? ???????????

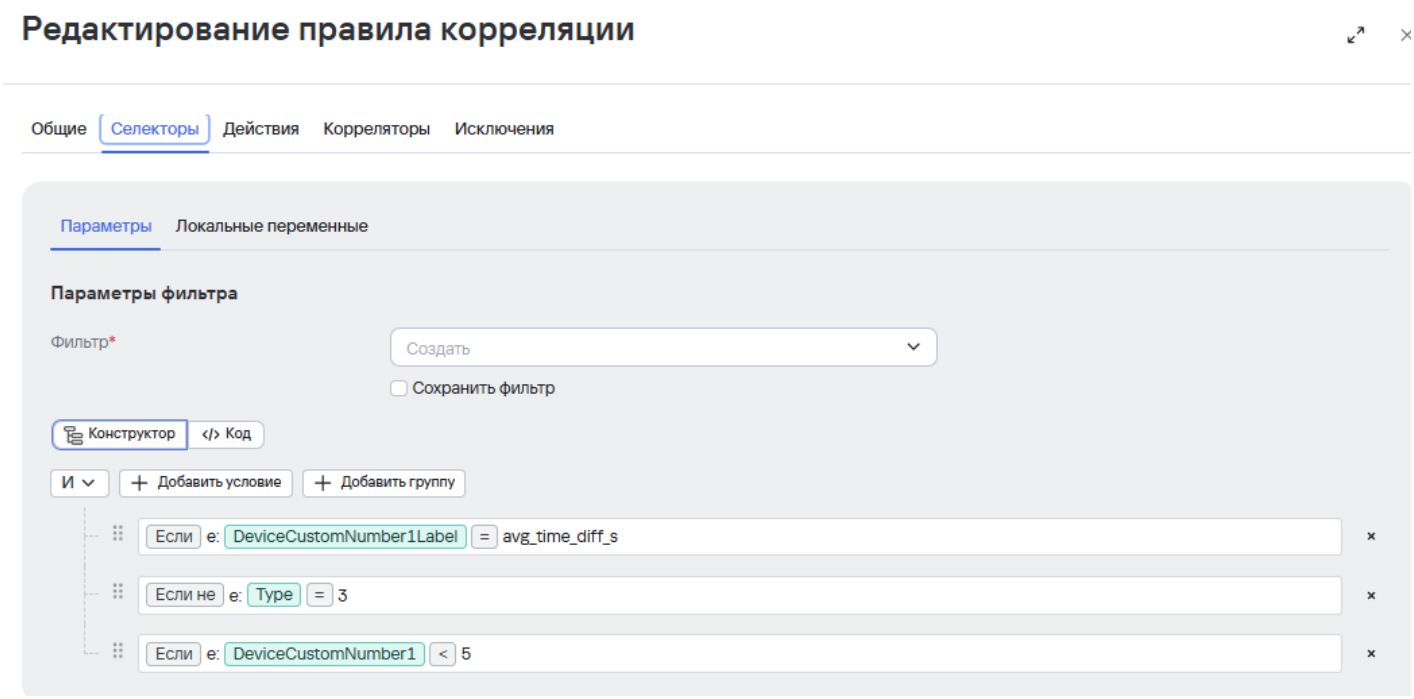
- Перейти в раздел **Ресурсы - Сбор и анализ данных** добавить планировщик по ранее созданному правилу
- Открыть правило и установить связи:
  - **Привязать хранилище** по которому будет осуществляться поиск на вкладке **Привязанные хранилища**
  - **Привязать коррелятор** с соответвующим правилом корреляции для сработки на вкладке **Привязанные корреляторы**
- Для ручного запуска нажмите кнопку **Запустить**



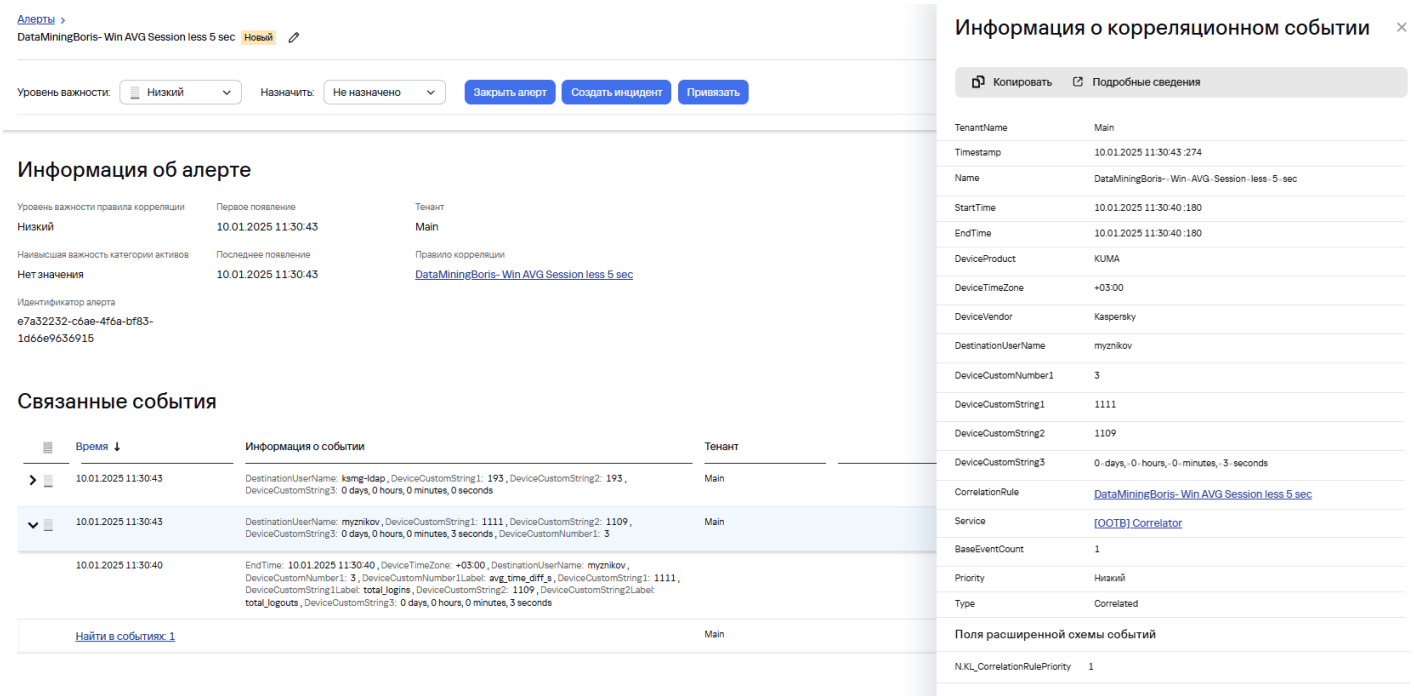
- По результатам запроса на выходе будут сформированы базовые события, которые не будут сохранены. Далее необходимо создать простое правило корреляции, чтобы создать корреляционное событие и алерт на данное событие и привязать правило к нужным корреляторам

3 ???? ???????? simple ??????? ?? ?????????

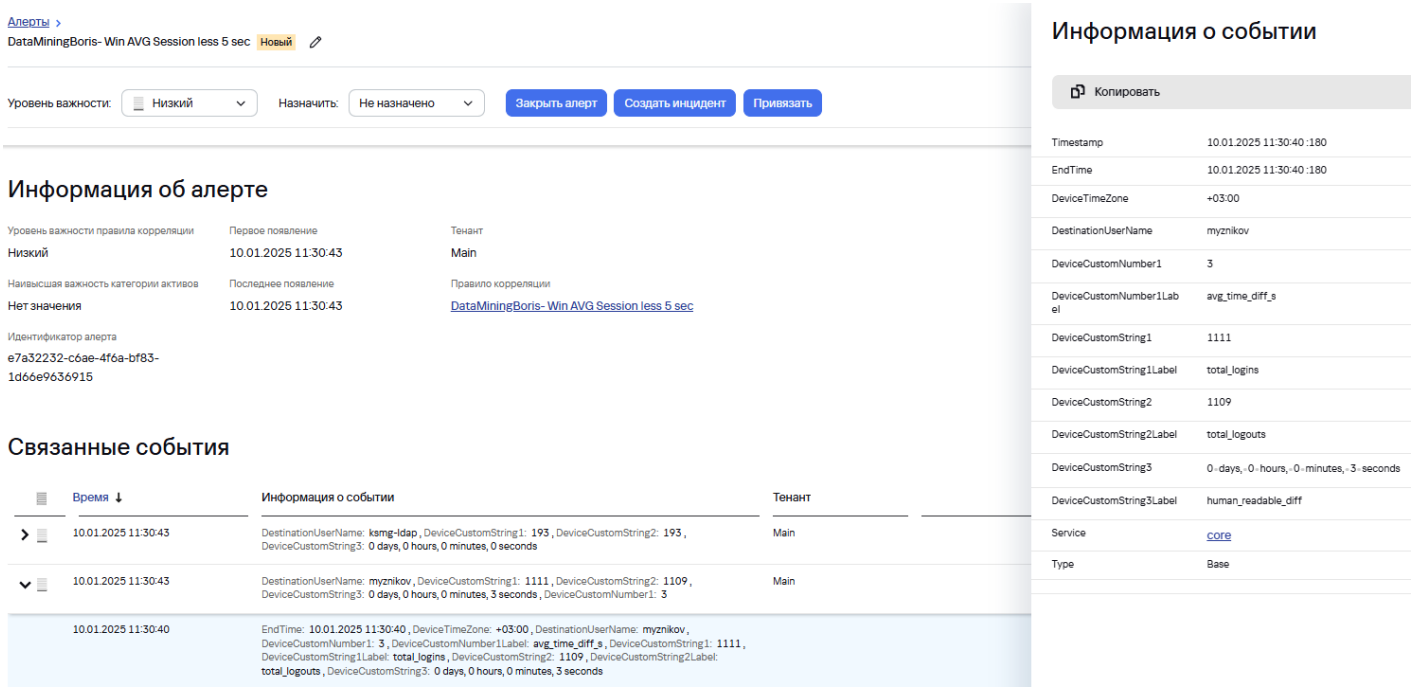
В нашем случае правило ловит события, где время сессии меньше 5 секунд:



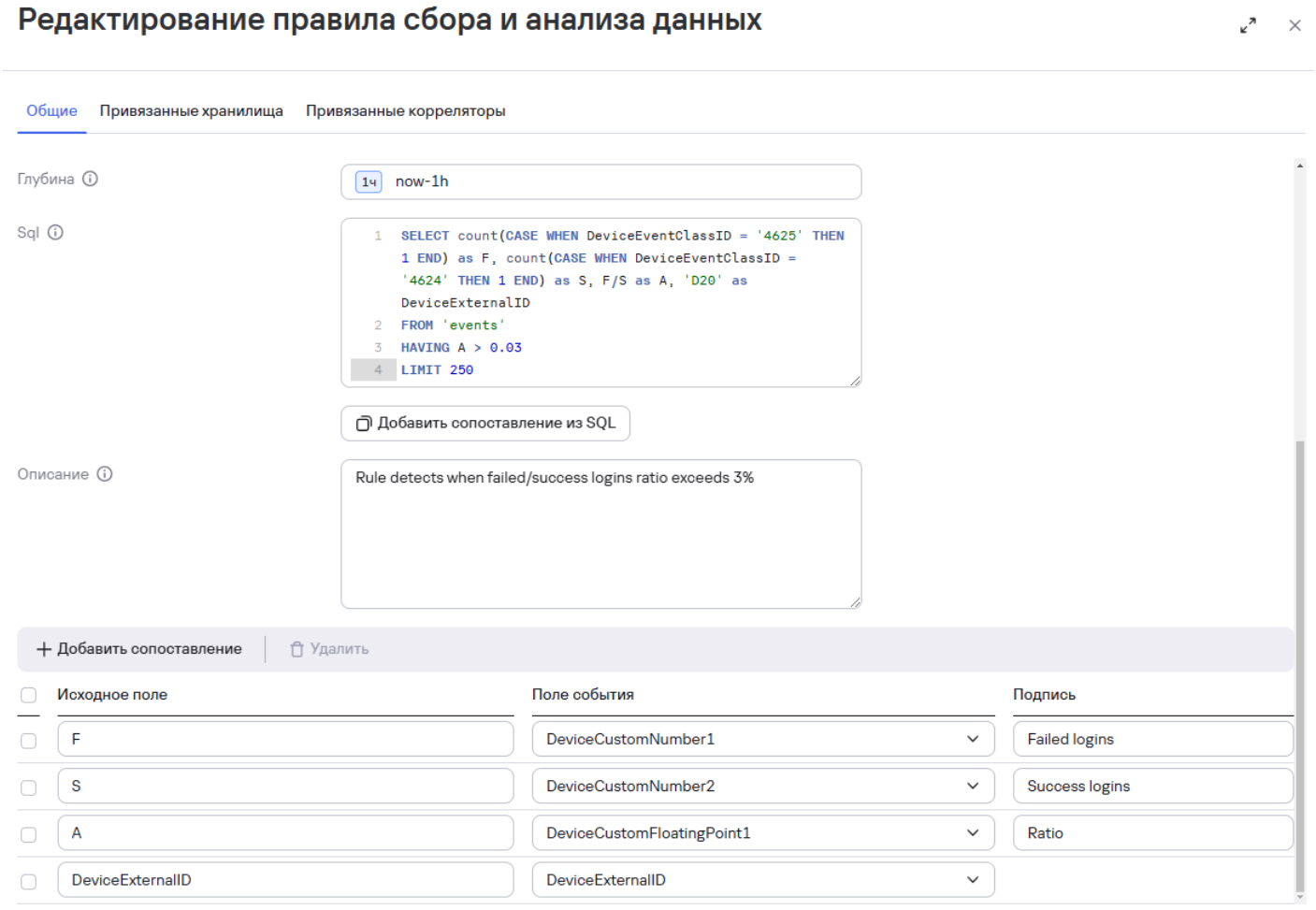
Корреляционное событие выглядит следующим образом:

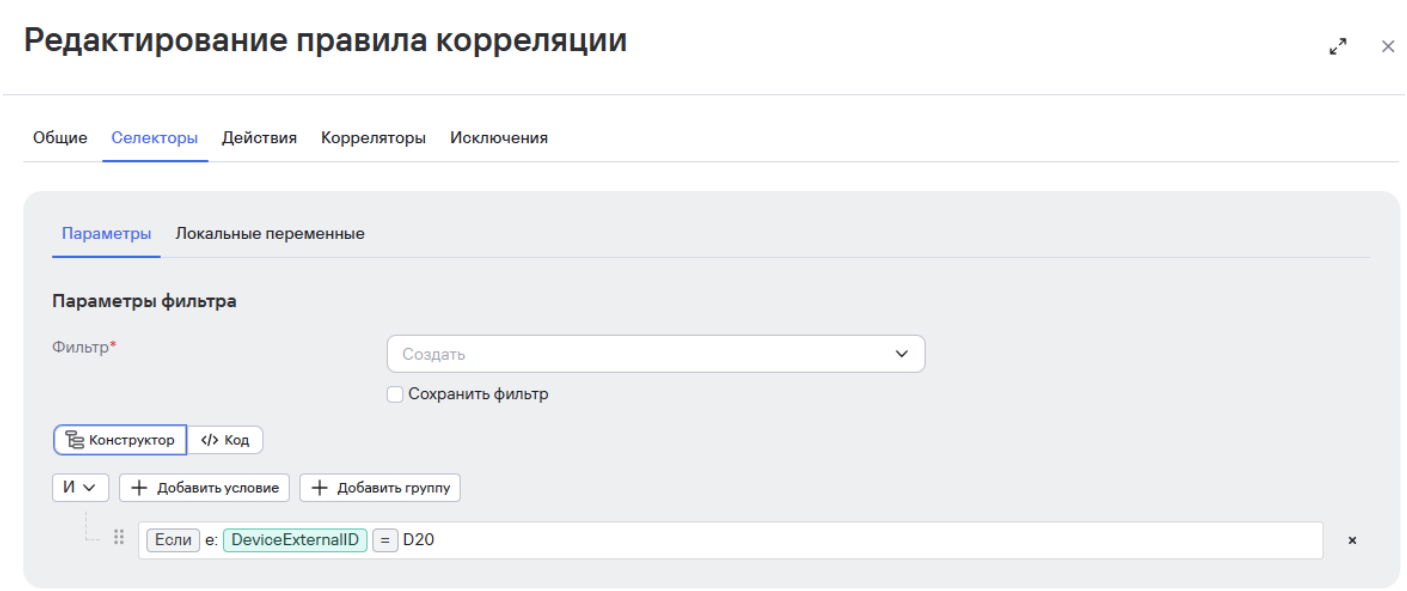


А событие на основе которого произошла сработка:

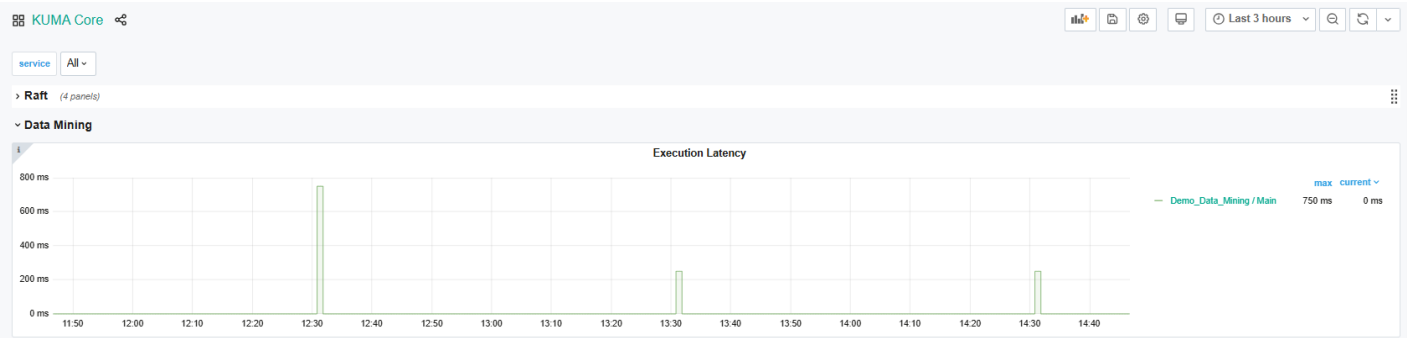

Алерты
DataMiningBoris- Win AVG Session less 5 sec
Новый
Уровень важности:
Низкий
Назначить:
Не назначено
Закрыть алерт
Создать инцидент
Привязать
Информация об алерте
Уровень важности правила корреляции
Низкий
Первое появление
10.01.2025 11:30:43
Тенант
Main
Наивысшая важность категории активов
Нет значения
Последнее появление
10.01.2025 11:30:43
Правило корреляции
DataMiningBoris- Win AVG Session less 5 sec
Идентификатор алерта
e7a32232-c6ae-4f6a-bf83-1d66e9636915
Связанные события
Время
Информация о событии
Тенант
10.01.2025 11:30:43
DestinationUserName: ksmg-ldap , DeviceCustomString1: 193 , DeviceCustomString2: 193 , DeviceCustomString3: 0 days, 0 hours, 0 minutes, 0 seconds
Main
10.01.2025 11:30:43
DestinationUserName: myznikov , DeviceCustomString1: 1111 , DeviceCustomString2: 1109 , DeviceCustomString3: 0 days, 0 hours, 0 minutes, 3 seconds , DeviceCustomNumber1: 3
Main
10.01.2025 11:30:40
EndTime: 10.01.2025 11:30:40 , DeviceTimeZone: +03:00 , DestinationUserName: myznikov , DeviceCustomNumber1: 3 , DeviceCustomNumber1Label: avg_time_diff_s , DeviceCustomString1: 1111 , DeviceCustomString1Label: total_logins , DeviceCustomString2: 1109 , DeviceCustomString2Label: total_logouts , DeviceCustomString3: 0 days, 0 hours, 0 minutes, 3 seconds
Информация о событии
Копировать
Timestamp 10.01.2025 11:30:40 :180
EndTime 10.01.2025 11:30:40 :180
DeviceTimeZone +03:00
DestinationUserName myznikov
DeviceCustomNumber1 3
DeviceCustomNumber1Label avg_time_diff_s
DeviceCustomString1 1111
DeviceCustomString1Label total_logins
DeviceCustomString2 1109
DeviceCustomString2Label total_logouts
DeviceCustomString3 0 days, 0 hours, 0 minutes, 3 seconds
DeviceCustomString3Label human_readable_diff
Service core
Type Base

Еще пример:


Редактирование правила сбора и анализа данных
Общие
Привязанные хранилища
Привязанные корреляторы
Глубина
1ч now-1h
Sql
SELECT count(CASE WHEN DeviceEventClassID = '4625' THEN 1 END) as F, count(CASE WHEN DeviceEventClassID = '4624' THEN 1 END) as S, F/S as A, 'D20' as DeviceExternalID
FROM 'events'
HAVING A > 0.03
LIMIT 250
Добавить сопоставление из SQL
Описание
Rule detects when failed/success logins ratio exceeds 3%
Добавить сопоставление
Удалить
Исходное поле
Поле события
Подпись
F
DeviceCustomNumber1
Failed logins
S
DeviceCustomNumber2
Success logins
A
DeviceCustomFloatingPoint1
Ratio
DeviceExternalID
DeviceExternalID



Работу правил можно отслеживать с помощью метрик в разделе KUMA Core:



????? ????????????? ??????

Использование Data Mining правил особенно актуально в ситуациях, когда классическая потоковая корреляция либо неэффективна, либо слишком ресурсоёмка. Рассмотрим основные практические сценарии:

1. **Когда нужно обработать много событий за период**
   Например, большое число неуспешных логинов за 5 минут или массовое сканирование портов.
   Data Mining позволяет считать такие вещи в ClickHouse, не загружая корреляторы.
2. **Когда нужно суммировать или усреднять значения**
   Можно использовать агрегирующие функции SQL: `SUM()`, `AVG()` и т.д.
   Пример: средний объём исходящего трафика или количество DNS-запросов.
   Алерт срабатывает при превышении порога.
3. **Когда нужен сравнительный анализ**
   Например, сравнить количество событий за последний час с таким же периодом сутки назад.
4. **Когда нужно работать со “скользящим” окном времени**
   Анализировать события за период, независимо от того, с какой задержкой они

пришли.

5. **Операции которые невозможны на уровне цепочки событий** (Количественный прирост, анализ на схожесть, а не на одинаковость)
6. **Подсчет энтропии**
   Определение при помощи энтропии, какие хосты генерируют одни и те же сработки для формирования исключений (рандомность, неожиданность цепочки)

???????? ??????

Рассмотрим более подробно на парочке примеров:

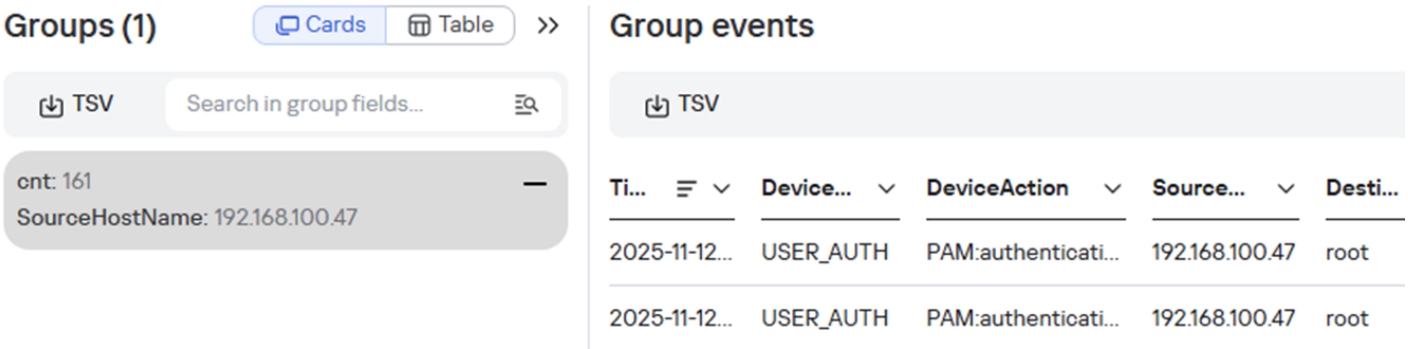
1. ?????? ?????????? ??????? ?????

Корреляционная логика, при которой требуется длительное накопление событий. Например:

- более **10 неуспешных попыток аутентификации под пользователем** `root`
- Берём **окно поиска 15 минут** и запускаем правило каждые **14 минут**.
- Так мы анализируем накопившиеся события и получаем результат без необходимости хранить все данные в памяти коррелятора.

```
SELECT
count(ID) AS cnt,
SourceHostName
FROM `events`
WHERE DeviceCustomString2 = 'ssh' AND DeviceEventClassID = 'USER_AUTH' AND DeviceVendor = 'Unix'
AND EventOutcome = 'failed' AND DestinationUserName = 'root'
GROUP BY SourceHostName
HAVING cnt > 10
```

Press Ctrl + Enter to run query



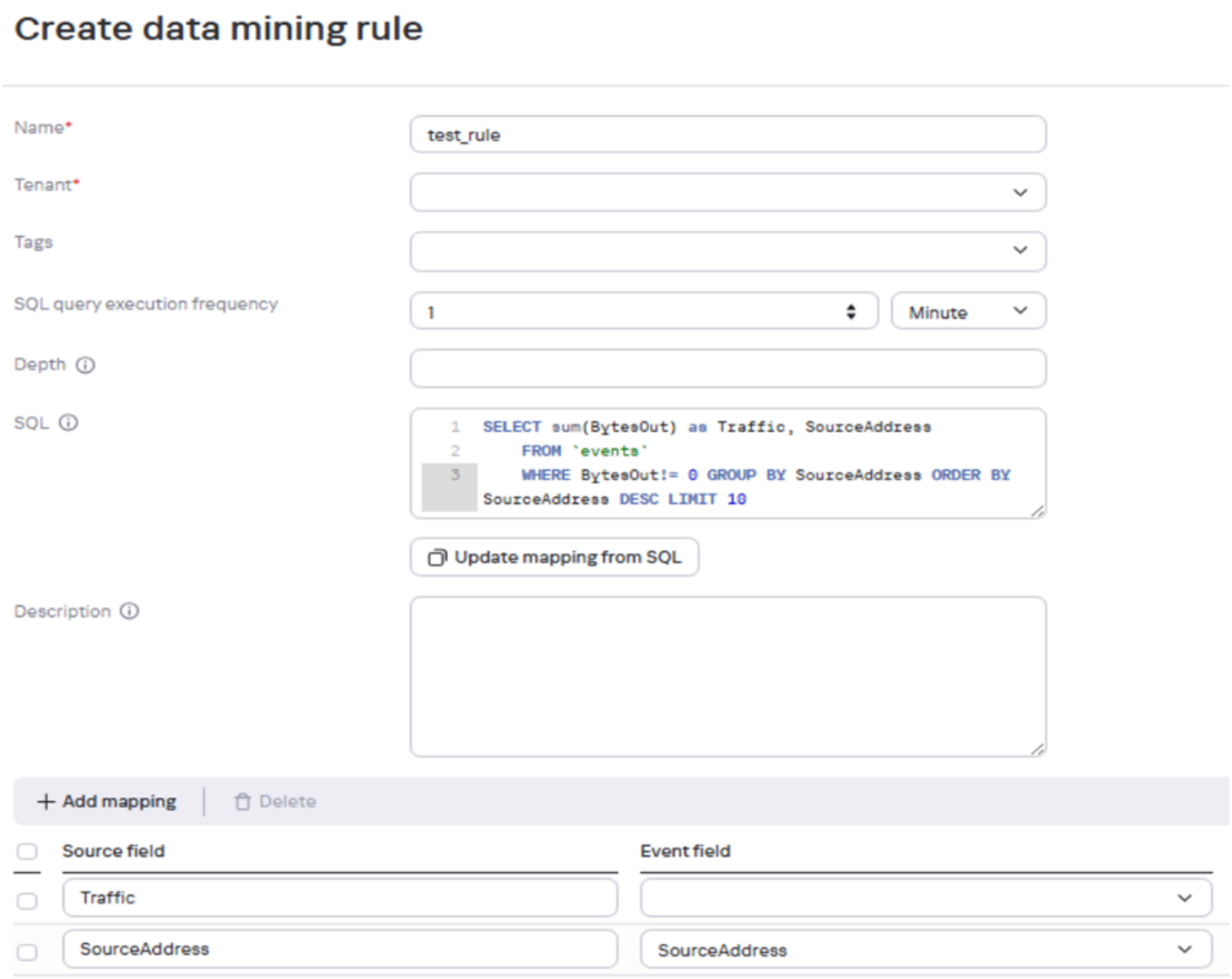
Частые неуспешные попытки входа под УЗ root

```
SELECT
SourceAddress,
SourceHostName,
DestinationUserName,
min(Timestamp) as StartTime,
max(Timestamp) as EndTime,
count(Distinct(DestinationServiceName)) as cnt_spn_names,
arrayCompact(groupUniqArray(DestinationServiceName)) as spn_names,
arrayStringConcat(
  arrayMap(x -> '\'' || x || '\'', groupUniqArray(Distinct(ID))),
  ', '
) AS BaseEventIDs,
 Count(*) as EventIDsCount,
 'SOCSh_Kerberoasting' as exID
FROM 'events'
WHERE
DeviceEventClassID = '4769' AND SourceUserID  != '' AND NOT ( SourceAddress in ('','::1')
or SourceAddress like '127.0.0.%') AND NOT endsWith(DestinationUserName,'$')
GROUP BY SourceAddress, SourceHostName, DestinationUserName
HAVING cnt_spn_names > 10
LIMIT 100
```

2. ??????? ?????????? ???????? ???????

Простой пример, где нужно суммировать данные и обнаруживать превышение порога.

- Суммируем исходящий трафик (`SUM(bytes_out)`) и группируем по адресу источника.
- В поле **«Глубина»** оставляем пусто — тогда нижняя граница интервала определяется автоматически как конец предыдущего запроса + 1.
- В поле **«Частота запуска»** ставим минимальное значение — **1 минута**.



В результате Scheduler каждую минуту запускает SQL-запрос с небольшим окном данных. Получается **скользящее окно**, которое постоянно обновляется и позволяет корректно работать даже при задержках в доставке событий и нарушении их порядка. Это как раз тот случай, когда **Data Mining правила способны сделать то, с чем потоковая корреляция справиться не может.**

??????? ??????? ??????

1. ???????? ??????? TGS ??????? (Kerberoasting)

Kerberoasting

```
SELECT
SourceAddress,
SourceHostName,
```

```
DestinationUserName,
min(Timestamp) as StartTime,
max(Timestamp) as EndTime,
count(Distinct(DestinationServiceName)) as cnt_spn_names,
arrayCompact(groupUniqArray(DestinationServiceName)) as spn_names,
arrayStringConcat(
  arrayMap(x -> '\'' || x || '\'', groupUniqArray(Distinct(ID))),
  ', '
) AS BaseEventIDs,
 Count(*) as EventIDsCount,
 'SOCSh_Kerberoasting' as exID
FROM 'events'
WHERE
DeviceEventClassID = '4769' AND SourceUserID  != '' AND NOT ( SourceAddress in ('','::1')
or SourceAddress like '127.0.0.%') AND NOT endsWith(DestinationUserName,'$')
GROUP BY SourceAddress, SourceHostName, DestinationUserName
HAVING cnt_spn_names > 10
LIMIT 100
```

2. ???????????? ?????? ? ???????????? ?????? (??????? ???????)

Сканирование портов

```
SELECT
arrayStringConcat(arraySort(groupUniqArray(Distinct(DeviceProduct))), ', ' ) AS
DeviceProduct,
arrayStringConcat(arraySort(groupUniqArray(Distinct(DeviceAddress))), ', ' ) AS
DeviceAddresses,
SourceAddress,
SourceHostName,
SourceNtDomain,
DestinationAddress,
min(Timestamp) as StartTime,
max(Timestamp) as EndTime,
arrayStringConcat(arraySort(groupUniqArray(Distinct(DestinationPort))), ', ' ) AS
DeviceCustomString1,
arrayStringConcat(
   arrayMap(x -> '\'' || x || '\'', groupUniqArray(Distinct(ID))),
```

```
   ', '
) AS DeviceCustomString2,
  Count(*) as DeviceCustomNumber2,
  Count(Distinct(DestinationPort)) as DeviceCustomNumber1,
  'SOCSh_ScanPort' as exID
FROM `events`
WHERE
Type=1 and
(DestinationPort < 1024 or DestinationPort in
(1434,1521,3306,3389,5432,8080,9200,1352,1540,1541)) AND
SourcePort>1024 and DestinationPort!=0 and SourceAddress!='' and DestinationAddress!=''
AND
(isIPAddressInRange(SourceAddress, '192.168.0.0/16') or isIPAddressInRange(SourceAddress,
'10.0.0.0/8') or isIPAddressInRange(SourceAddress, '172.16.0.0/12')) AND
(isIPAddressInRange(DestinationAddress, '192.168.0.0/16') or
isIPAddressInRange(DestinationAddress, '10.0.0.0/8') or
isIPAddressInRange(DestinationAddress, '172.16.0.0/12'))
GROUP BY SourceAddress,SourceHostName,SourceNtDomain,DestinationAddress
HAVING DeviceCustomNumber1>=10
LIMIT 100
```

3. ??????? ????????????? ??????? (??????????? ??????????)

Прирост корреляционных событий более 20% за сутки

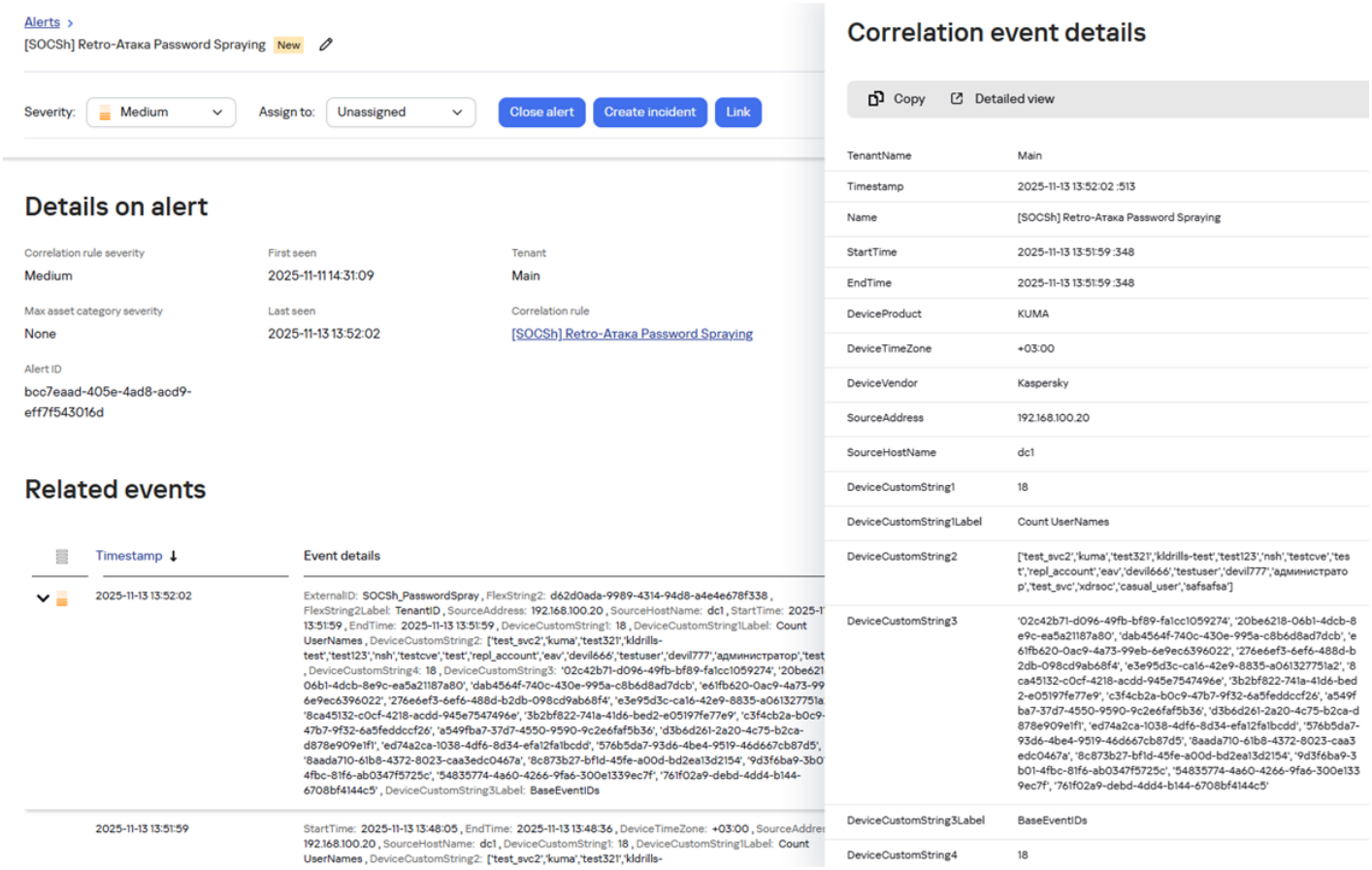
```
SELECT
   'CorrelationSplash' as ExternalId,
   TenantID,
   CorrelationRuleID,
   CorrelationRuleName,
   countIf(Timestamp between toUnixTimestamp64Milli(now64()) - 1*3600000 and
toUnixTimestamp64Milli(now64())) as today,
   countIf(Timestamp between toUnixTimestamp64Milli(now64()) - 25*3600000 and
toUnixTimestamp64Milli(now64())-24*3600000) as yesterday,
   round(today/yesterday,2) as k
FROM `events`
WHERE Type=3 and toDayOfWeek(now64())!=1
GROUP  BY TenantID,CorrelationRuleID,CorrelationRuleName
```

```
HAVING yesterday > 20 and k>1.2
LIMIT 250
```

4. ??????????/?????? ???????

SQL запрос правила Password Spraying

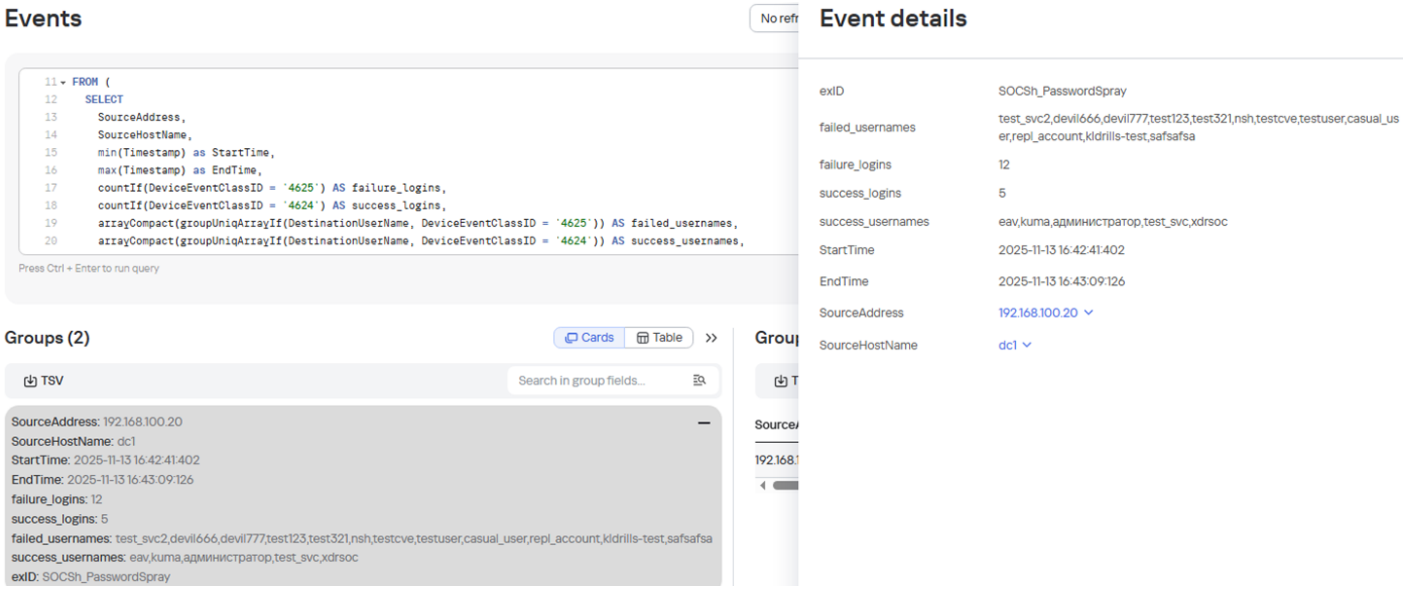
```
SELECT
SourceAddress, SourceHostName,
min(Timestamp) as StartTime, max(Timestamp) as EndTime,
    count(Distinct(DestinationUserName)) as cnt_usernames, /*кол-во уникальных УЗ*/
    arrayCompact(groupUniqArray(DestinationUserName)) as spray_usernames, /*уникальные
сортированные имена УЗ, склеенные в строку*/
    arrayStringConcat(arrayMap(x -> '\'' || x || '\'', groupUniqArray(Distinct(ID))),', ')
as BaseEventIDs, /*уникальные сортированные ID базовых событий, склеенные в строку*/
Count(*) as EventIDsCount,
 'SOCSh_PasswordSpray' as exID
FROM
  'events'
WHERE
  DeviceEventClassID = '4625'
  AND DestinationNtDomain  != ''
  AND NOT endsWith(DestinationUserName,'$')
GROUP BY SourceAddress, SourceHostName
HAVING cnt_usernames > 10
LIMIT 100
```



SQL запрос правила Password Spraying с сохранением имен пользователей успешного и неудачного логина

```
SELECT
  SourceAddress, SourceHostName, StartTime,
EndTime,failure_logins,success_logins,failed_usernames,success_usernames,exID
FROM (
  SELECT
      SourceAddress, SourceHostName, min(Timestamp) as StartTime, max(Timestamp) as
EndTime,
      countIf(DeviceEventClassID = '4625') AS failure_logins,
      countIf(DeviceEventClassID = '4624') AS success_logins,
      arrayCompact(groupUniqArrayIf(DestinationUserName, DeviceEventClassID = '4625')) AS
failed_usernames,
      arrayCompact(groupUniqArrayIf(DestinationUserName, DeviceEventClassID = '4624')) AS
success_usernames,
      'SOCSh_PasswordSpray' as exID
  FROM events
  WHERE
        DeviceEventClassID IN ('4625', '4624')
```

```
        AND DestinationNtDomain != ''
        AND NOT endsWith(DestinationUserName,'$')
  GROUP BY SourceAddress, SourceHostName)
WHERE failure_logins > 10
LIMIT 100
```



6. ??????? ???????? ??? ??????????? ??????????.

Запрос на Подсчет энтропии, который может помочь для определения и внесения исключений в правила корреляции.

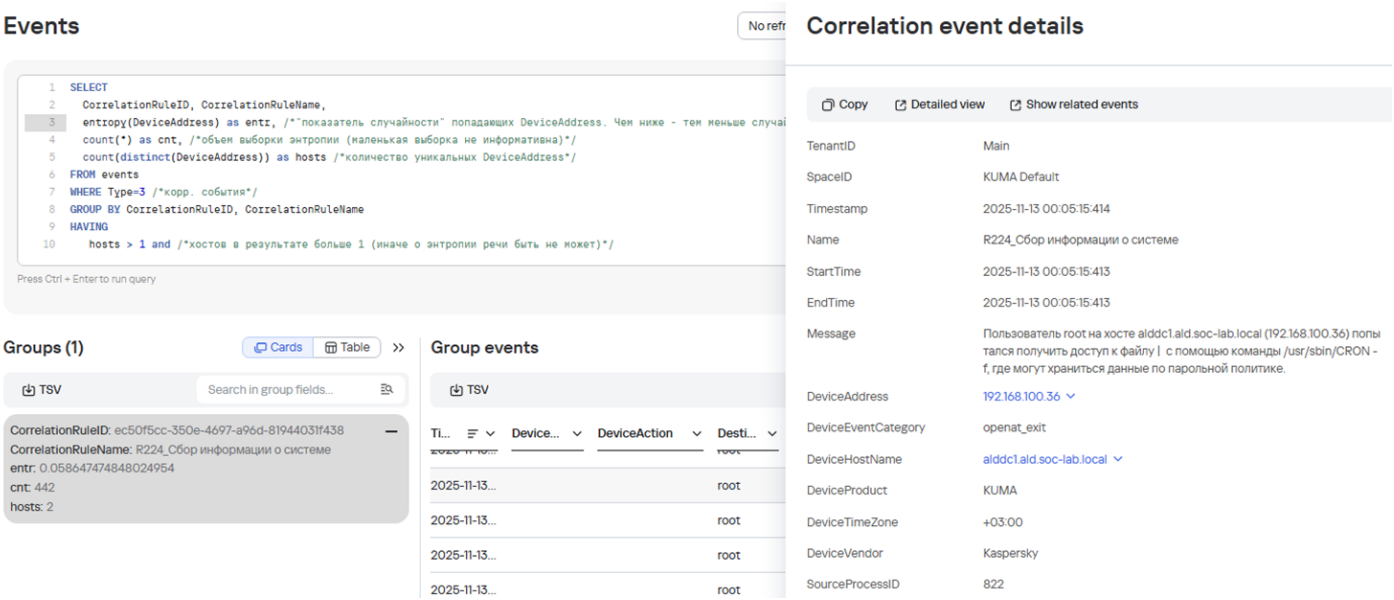
Если очень грубо, энтропия это показатель случайности и чем она ниже - тем ниже случайности попадания источника DeviceAddress в корреляционное событие

Иными словами, можно определить, какие одни и те же хосты попадают в одни и те же алерты на постоянной основе и после проверки внести их в исключения

Подсчет энтропии для определения исключений

```
SELECT
  CorrelationRuleID,
  CorrelationRuleName,
  entropy(DeviceAddress) as entr, /*"показатель случайности" попадающих DeviceAddress. Чем ниже - тем меньше случайности*/
  count(*) as cnt, /*объем выборки энтропии (маленькая выборка не информативна)*/
  count(distinct(DeviceAddress)) as hosts, /*количество уникальных DeviceAddress*/
'SOCSh_Entropy'as ExternalID
```

```
FROM events
WHERE Type=3 /*корр. события*/
GROUP BY CorrelationRuleID, CorrelationRuleName
HAVING
   hosts > 1 and /*хостов в результате больше 1 (иначе о энтропии речи быть не может)*/
   cnt > 10 /*количество событий достаточно для оптимальной оценки*/
ORDER BY entr ASC /*сортируем по принципу "наименее случайные последовательности"*/
LIMIT 250
```



А также другие примеры:

1. Скачок событий/алертов со средств защиты
2. Большое количество DNS запросов с хоста

Пакет ресурсов Data Mining правил: Shared_20251113_231513_DMRules
Пароль к ресурсу: `!QAZ2wsx#EDC_!QAZ2wsx#EDC_`

????? ???????????? ??????? SQL ? ????????

Здесь перечислены самые часто встречающиеся функции, которые используются в запросах:

1. `arrayStringConcat` - объединяет элементы массива в строку
2. `arrayCompact` - удаляет последовательные дублирующиеся элементы из массива
3. `distinct` - уникальные значения
4. `groupUniqArray` - собирает значения в массив
5. `arraySort` - сортирует массив

6. `arrayMap` - применяет выражение к каждому элементу массива и возвращает новый массив с результатами

Возможно использовать все функции, описанные в документации ClickHouse:
https://clickhouse.com/docs/ru/sql-reference/functions

А также набор специальных функций enrich и lookup в KUMA:
https://support.kaspersky.com/help/KUMA/4.0/ru-RU/294927.htm

Например:

1. Уникальные отсортированные имена пользователей, склеенные в строку.

```
arrayCompact(arraySort(groupUniqArray(DestinationUserName)))
```

spray_usernames: casual_user,devil666,devil777,eav,kldrills-test,kuma,nsh,repl_account,safsafsa,test,test123,test321,test_svc,test_svc2,testcve,testuser,xdrsoc,администратор

2. Уникальные ID базовых событий, склеенные в строку

```
arrayStringConcat(  arrayMap(x -> '\'' || x || '\'', groupUniqArray(Distinct(ID))),  ', ') AS BaseEventIDs
```

BaseEventIDs: '02c42b71-d096-49fb-bf89-fa1cc1059274', '20be6218-06b1-4dcb-8e9c-ea5a21187a80', 'dab4564f-740c-430e-995a-c8b6d8ad7dcb', 'e61fb620-0ac9-4a73-99eb-6e9ec6396022', '276e6ef3-6ef6-488d-b2db-098cd9ab68f4', 'e3e95d3...

Revision #11
Created 2025-01-10 08:44:23 UTC by Boris Rzr
Updated 2026-06-16 12:05:04 UTC by Boris Rzr