СРАВНИТЕЛЬНЫЙ АНАЛИЗ ФОРМАТОВ SQL И NoSQL ДЛЯ ОПИСАНИЯ СОБЫТИЙ БЕЗОПАСНОСТИ

Э.В. Бирих, *Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, be1982@mail.ru;*
Н.С. Ершова, *Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича, Ershova.for.work@gmail.com.*

УДК 004.056

Аннотация. Проведено исследование форматов *SQL* и *NoSQL* в контексте использования их для описания событий безопасности. В результате представлен анализ особенностей, преимуществ и недостатков каждого из форматов, а также их применимость для описания событий безопасности. Рассмотрены различные сценарии использования и представлены примеры применения форматов для конкретных потребностей.

***Ключевые слова*:** события безопасности; *NoSQL*; *SQL*; базы данных; базы данных событий безопасности; описание событий безопасности.

COMPARATIVE ANALYSIS OF SQL AND NoSQL FORMATS FOR DESCRIBING SECURITY EVENTS

Ernest Birikh, *The Bonch-Bruevich Saint-Petersburg State University of Telecommunications;*
Natalya Ershova, *The Bonch-Bruevich Saint-Petersburg State University of Telecommunications.*

Annotation. A study of the *SQL* and *NoSQL* formats has been conducted in order to use them to describe security events. As a result, an analysis of the features, advantages and disadvantages of each of the formats is presented, as well as their applicability to describe security events. Various usage scenarios are considered and examples of using formats for specific needs are presented.

***Keywords*:** security events; *NoSQL*; *SQL*; databases; databases of security events; description of security events.

Введение

Обработка событий безопасности играет критически важную роль в современной информационной безопасности. События безопасности могут включать в себя различные инциденты, такие как попытки взлома, вирусные атаки, утечки данных и другие потенциальные угрозы. Быстрое и эффективное реагирование на эти события может помочь предотвратить ущерб и минимизировать риски, а также выявить уязвимости в системе и принять меры по их устранению и предотвращению возможных атак в будущем. Цель данной статьи – провести сравнительный анализ форматов *SQL* и *NoSQL* в контексте описания событий безопасности. Данная работа позволит обосновать выбор между этими форматами в зависимости от конкретных потребностей пользователя.

SQL (*Structured Query Language*) и *NoSQL* (*Not Only SQL*) – это два основных формата баз данных, которые могут быть использованы для описания событий безопасности. Они предоставляют средства для хранения, обработки и анализа данных о событиях безопасности, что позволяет специалистам по безопасности быстро выявлять угрозы и принимать меры. Важность эффективного описания и

анализа событий безопасности не может быть недооценена. Выбор подходящего формата для хранения и обработки этих данных является критическим фактором.

Прежде чем приступить к анализу данных форматов, следует рассмотреть определения основных понятий, используемых в данной статье:

Событие безопасности – это некоторое происшествие в сети или системе, которое может иметь влияние на конфиденциальность, целостность или доступность данных [1]. Это может включать в себя широкий спектр инцидентов, от попыток несанкционированного доступа и вирусных атак до утечек данных и других нарушений безопасности [2]. События безопасности могут быть идентифицированы с помощью различных средств мониторинга и анализа, таких как системы обнаружения вторжений (*IDS*), системы предотвращения вторжений (*IPS*) и системы управления информацией и событиями безопасности (*SIEM*) [3].

Системы, поддерживающие эффективное программное обеспечение для автоматизированного анализа событий безопасности на основе технологий СУБД, позволяют регулярно вносить информацию о событиях в базу данных, а также выполнять сложные операции по хранению, фильтрации, отображению и поиску инцидентов безопасности [4].

Базы данных, хранящие информацию о событиях безопасности, имеют название базы данных событий безопасности (БДСБ). События в них добавляются специальным модулем, отвечающим за регистрацию событий безопасности. Информация о событиях направляется в модуль регистрации событий безопасности от модуля фильтрации событий аудита операционной системы и агентов дополнительной регистрации событий. Эти агенты отфильтровывают события аудита приложений и, при необходимости, регистрируют те события безопасности, которые не были зарегистрированы с помощью инструментов аудита операционной системы и приложений [1].

Периодическое пополнение базы данных может происходить, например, с помощью программы конвертера данных, который преобразует данные из журналов аудита операционной системы и приложений [5]. Непрерывное пополнение базы данных происходит с помощью модуля регистрации событий, который в режиме реального времени отслеживает состояние журналов аудита и пополняет БД при обнаружении новых записей.

Форматы *SQL* и *NoSQL* в контексте описания событий безопасности

SQL – это стандартный язык для управления данными в реляционных базах данных. *SQL* базы данных используют структурированный подход, где информация организована в таблицы, и каждая из них имеет определенную схему, определяющую структуру данных [6]. *SQL* поддерживает сложные запросы и обеспечивает мощные средства для анализа данных. В контексте описания событий безопасности *SQL* может быть использован для хранения, обработки и анализа данных о различных инцидентах безопасности, таких как попытки взлома, вирусные атаки, утечки данных и т.д.

К преимуществам использования *SQL* для описания событий безопасности можно отнести следующее:

- Структурированность. *SQL* базы данных применяют структурированный подход, что позволяет эффективно организовать данные о событиях безопасности. Это облегчает поиск и анализ данных.
- Сложные запросы. *SQL* предоставляет мощные средства для выполнения сложных запросов. Это может быть особенно полезно при анализе событий безопасности, когда необходимо выявить сложные взаимосвязи и корреляции.

- Транзакционность. *SQL* базы данных поддерживают транзакции, что обеспечивает целостность данных даже в случае сбоев или ошибок.
- Безопасность. Большинство *SQL* баз данных предлагают продвинутые функции безопасности, такие как управление доступом, шифрование данных и аудит.

Недостатки и ограничения *SQL* в этом контексте описания событий безопасности:

- Масштабируемость. *SQL* базы данных могут столкнуться с проблемами масштабируемости при обработке очень больших объемов данных о событиях безопасности.
- Гибкость схемы. *SQL* требует строгой схемы данных, что может быть неудобно при работе с неструктурированными данными, такими как логи событий безопасности.

Если говорить о примерах использования *SQL* для описания событий безопасности, то он может быть использован для создания таблицы *security_events.*

На рис. 1 показан пример создания таблицы, которая содержит данные о различных событиях безопасности.

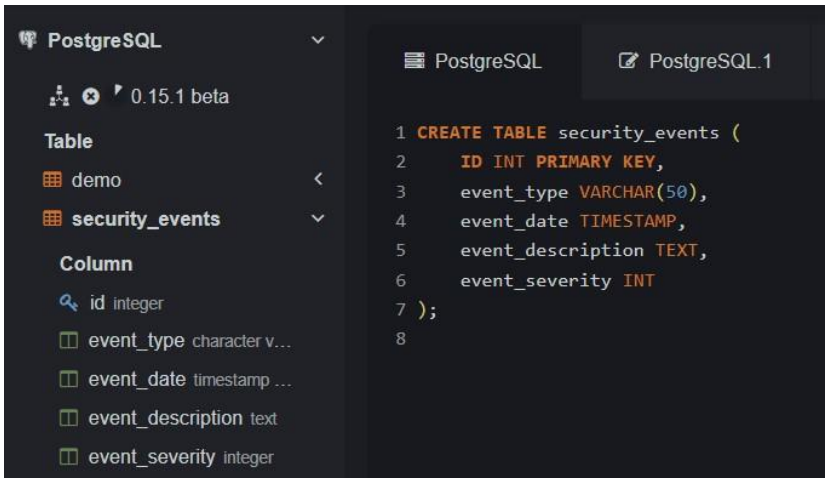

Рисунок 1

Затем можно использовать *SQL* – запросы для добавления, извлечения и анализа данных о событиях безопасности. Например, запрос на рис. 2 добавляет новое событие безопасности в таблицу.

```
1 INSERT INTO security_events (ID, event_type, event_date,
2                              event_description, event_severity)
3 VALUES (1, 'Unauthorized Access Attempt', '2024-04-20 00:00:00',
4         'An unauthorized access attempt was detected.', 5);
5
```

Рисунок 2

А запрос на рис. 3 извлекает все события безопасности с высокой степенью серьезности.

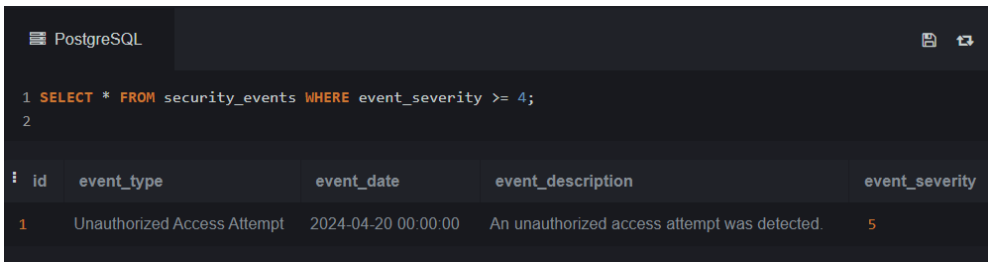

Рисунок 3

Это лишь некоторые из примеров того, как *SQL* может быть использован для описания и анализа событий безопасности. Однако возможности *SQL* в этом контексте значительно шире и могут включать в себя более сложные запросы и аналитические функции.

NoSQL – это альтернативный подход к управлению данными, который был разработан для обработки больших объемов данных, которые могут быть структурированы, полуструктурированы или неструктурированы. *NoSQL* базы данных не требуют фиксированной схемы, они могут масштабироваться горизонтально и гарантируют высокую эффективность при работе с большими объемами данных [7]. *NoSQL* включает в себя различные типы баз данных, такие как документо-ориентированные, ключ-значение, колоночные и графовые базы данных. В контексте описания событий безопасности, *NoSQ*L может быть использован для хранения и обработки больших объемов данных о событиях безопасности, включая неструктурированные данные, такие как логи, сетевой трафик и т.д.

Так как информация о событиях безопасности приходит из разных мест, то зачастую рационален выбор баз данных семейства *NoSQL* для получения, хранения и обновления в режиме реального времени большого потока информации. С целью оптимизации запросов к данным о событиях безопасности в *NoSQL* базах данных можно использовать индексы, кэширования данных и партиционирования. Индексы позволяют быстро находить данные по определенным атрибутам, кэширование данных позволяет ускорить доступ к часто запрашиваемым данным, а партиционирование позволяет распределить данные по нескольким узлам для увеличения производительности.

Преимущества использования *NoSQL* для описания событий безопасности:

- Гибкость схемы. *NoSQL* базы данных не требуют фиксированной схемы, что позволяет легко и быстро адаптироваться к изменяющимся требованиям и структурам данных. Это может быть особенно полезно при работе с неструктурированными данными, такими как логи событий безопасности.
- Масштабируемость. *NoSQL* базы данных обеспечивают высокую горизонтальную масштабируемость, что позволяет эффективно обрабатывать большие объемы данных.
- Быстродействие. *NoSQL* базы данных имеют высокую производительность при работе с большими объемами данных, что может быть критически важно при обработке данных о событиях безопасности в реальном времени [8].

Недостатки и ограничения *NoSQL* в этом контексте:

- Сложность запросов. *NoSQL* базы данных могут не поддерживать такой широкий спектр сложных запросов, как *SQL*. Это может ограничить возможности анализа данных о событиях безопасности.

- Транзакционность. В отличие от *SQL*, многие *NoSQL* базы данных не поддерживают полную *ACID* транзакционность, что может быть проблемой в некоторых сценариях.

NoSQL может быть использован для хранения и обработки больших объемов данных о событиях безопасности. Например, документо-ориентированная *NoSQL* база данных, такая как *MongoDB*, может быть использована для хранения логов событий безопасности в формате *JSON*.

Также можно использовать *NoSQL* запросы для извлечения и анализа данных о событиях безопасности. Например, описанный на рис. 4 код создает коллекцию «*security_events*», вставляет новый документ в коллекцию «*security_events*». Документ, который добавляется, содержит информацию о событии безопасности, включая идентификатор события, тип события, дату события, описание события и уровень серьезности события. Далее выполняется поиск в коллекции «*security_events*» и извлекаются все документы, где значение поля «*event_severity*» больше или равно 4 (высокая степень серьезности).

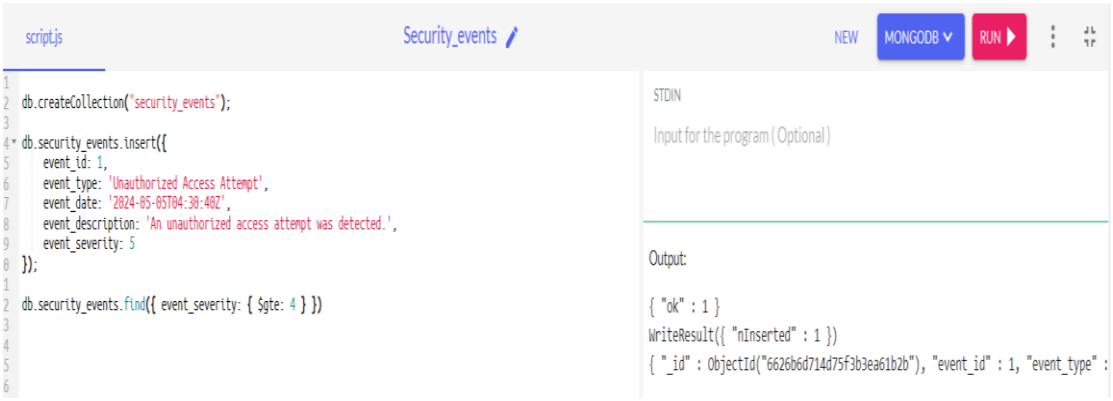

Рисунок 4

Это лишь некоторые из примеров того, как *NoSQL* может быть использован для описания и анализа событий безопасности. *NoSQL* следует выбирать для описания событий безопасности, когда необходимо обрабатывать большие объемы данных с высокой скоростью и гибкостью.

Сравнительный анализ *SQL* и *NoSQL*

Нереляционные *NoSQL* базы данных отличаются от реляционных *SQL* тем, что они не используют традиционную схему таблиц и связей между ними. Вместо этого они используют различные модели данных, такие как документы, ключ-значение, столбцы и графы [9]. Эти модели данных позволяют гибко хранить и обрабатывать данные о событиях безопасности.

SQL и *NoSQL* базы данных могут предложить различные показатели, которые полезны в зависимости от конкретных требований системы и условий использования. Рассмотрим их более подробно на примере *MySQL* и *MongoDB*. Результаты представлены в табл. 1.

Таблица 1.

	SQL	***NoSQL***
Тип	Реляционный	Нереляционный

	SQL	***NoSQL***
Данные	Структурированные данные, хранящиеся в таблицах	Неструктурированные данные, хранятся в файлах *JSON*
Схема	Статический	Динамический
Масштабируемость	Вертикальный	Горизонтальный
Язык	Язык структурированных запросов	Язык неструктурированных запросов
Объединение элементов (*Joins*)	Присутствует для написания сложных запросов	Отсутствует
OLTP	Рекомендуется в *OLTP*-системах	С наименьшей вероятностью будет рассмотрен в системе *OLTP*
Поддержка	Активная поддержка	Расширяется
Интегрированное кэширование	Поддерживает встроенную память	Поддерживает интегрированное кэширование
Гибкость	Жесткая схема, привязанная к отношениям	Нежесткая схема и гибкость
Операции	*ACID*	Теоремы *CAP*
Эластичные запросы	В большинстве случаев требуется время простоя	Автоматические, не требующие отключения

Описанные выше характеристики *SQL* и *NoSQL* баз данных могут напрямую влиять на эффективность описания событий безопасности следующим образом:

Данные и схема. Тип данных, которые необходимо хранить, может определить, какой формат базы данных будет наиболее эффективным. *SQL* использует фиксированную схему, что обеспечивает структурированность и предсказуемость данных, в то время как *NoSQL* предлагает гибкую схему, что позволяет легко адаптироваться к изменяющимся требованиям и структурам данных, также он полезен для работы с данными, которые не имеют жесткой структуры, например, данные о поведении пользователей.

Масштабируемость. *SQL* базы данных обычно поддерживают вертикальное масштабирование, означающее, что можно увеличить производительность, добавив больше ресурсов (например, *CPU, RAM*) к серверу. Однако, они могут столкнуться с проблемами при горизонтальном масштабировании (т.е., добавлении большего количества серверов). *NoSQL* базы данных предлагают высокую горизонтальную масштабируемость, подразумевающую, что можно увеличить производительность, добавив больше серверов в систему [10]. Это может быть особенно полезно при обработке больших объемов данных о событиях безопасности. Если данные сравнительно статичны и не ожидается значительного роста, то *SQL* может быть более подходящим.

Язык. SQL предлагает мощный язык запросов для сложного анализа данных, что может быть полезно при исследовании событий безопасности. *NoSQL* может быть менее мощным в этом отношении, но некоторые *NoSQL* базы данных

предлагают гибкие языки запросов, в зависимости от типа базы данных (например, *MongoDB* использует *JavaScript*-подобный язык запросов), которые могут быть достаточно мощными для многих задач.

Объединение (*Joins*). Если присутствует необходимость связывать данные из разных таблиц или коллекций, *SQL* может быть более эффективным благодаря своей поддержке операций объединения. Большинство *NoSQL* баз данных не поддерживают операции объединения.

OLTP. Если нужно обрабатывать транзакции, *SQL* может быть более эффективным, благодаря своей поддержке *ACID* транзакций. *NoSQL* может поддерживать транзакции, но это зависит от конкретной базы данных.

Поддержка. *SQL* базы данных имеют долгую историю и большое сообщество поддержки, что может облегчить решение проблем и улучшение производительности. *NoSQL* базы данных также имеют активные сообщества, но они могут быть менее зрелыми.

Интегрированное кэширование. Некоторые *NoSQL* базы данных предлагают интегрированные возможности кэширования, что может улучшить производительность при обработке больших объемов данных. *SQL* базы данных, как правило, нуждаются в специализированном решении для кэширования, таком как *Memcached.*

Гибкость. *NoSQL* базы данных обычно более гибкие в отношении структуры данных и схемы, что может быть полезно при работе с неструктурированными данными, такими как логи событий безопасности [6]. *SQL* базы данных требуют более строгой структуры и схемы [11].

Операции. *SQL* поддерживает широкий спектр операций и функций, включая сложные запросы и аналитические функции. *NoSQL* базы данных могут поддерживать различные операции, в зависимости от типа базы данных, но они могут быть менее мощными для сложных запросов [12].

Эластичные запросы. Некоторые *NoSQL* базы данных, такие как *Elasticsearch,* поддерживают эластичные запросы, что позволяет выполнять гибкие и динамические запросы. *SQL* базы данных обычно требуют более строгих и статических запросов.

Пример использования

Рассмотрим пример события безопасности, которое включает в себя обнаружение несанкционированного доступа к системе.

Если используется *SQL* база данных для описания этого события, можно создать таблицу с полями для *IP*-адреса, времени доступа, местоположения и других связанных данных. *SQL* позволяет вам легко связывать эти данные с другими таблицами, например, с таблицей пользователей или таблицей серверов. Это может быть полезно для анализа паттернов поведения и выявления подозрительной активности. Однако, если объем данных очень большой или данные приходят в большом объеме и быстро меняются, *SQL* база данных может столкнуться с проблемами производительности и масштабируемости.

Если используется *NoSQL* база данных, например, документо-ориентированная база данных типа *MongoDB*, можно хранить данные о событии безопасности в формате *JSON*, что может включать в себя различные типы данных, и можно легко добавлять новые типы данных по мере необходимости. *NoSQL* база данных может легко масштабироваться для обработки больших объемов данных, что может быть полезно для обработки больших потоков данных о событиях безопасности в реальном времени. Однако, *NoSQL* может быть менее эффективным

для сложного анализа данных, так как он не поддерживает сложные запросы и операции объединения, как *SQL*.

В обоих случаях выбор между *SQL* и *NoSQL* будет зависеть от конкретных требований к описанию событий безопасности, включая объем данных, скорость изменения данных, необходимость в сложном анализе данных и другие факторы.

Примеры использования *SQL* баз данных для хранения и анализа данных о событиях безопасности

В области обработки событий безопасности *SQL* базы данных могут быть использованы для хранения и анализа данных о событиях безопасности. Приведем несколько примеров:

Журналы аудита. *SQL* базы данных могут быть использованы для хранения журналов аудита, которые содержат информацию о действиях, произведенных пользователями или системами. Это может включать в себя действия, такие как вход в систему, изменение настроек безопасности или доступ к конфиденциальной информации.

События IDS/IPS. Системы обнаружения и предотвращения вторжений (*IDS/IPS*) часто используют *SQL* базы данных для хранения информации об обнаруженных угрозах. Это может включать в себя детали о типе угрозы, целевом устройстве и времени обнаружения.

Данные об угрозах. *SQL* базы данных могут быть использованы для хранения данных об угрозах, таких как *IP*-адреса, связанные с известными злонамеренными активностями, или хэши файлов, связанных с вредоносным ПО.

События журнала безопасности. *SQL* базы данных могут быть использованы для хранения событий из журналов безопасности, таких как журналы безопасности *Windows* или журналы событий *Linux*.

Важно отметить, что при использовании *SQL* баз данных для обработки событий безопасности необходимо учесть некоторые вопросы безопасности. Например, база данных должна быть защищена от несанкционированного доступа, а данные должны быть зашифрованы при хранении и передаче. Кроме того, может потребоваться регулярное резервное копирование данных для обеспечения их восстановления в случае сбоя или атаки.

Рекомендации по использованию *NoSQL* баз данных и их возможности для хранения и анализа данных о событиях безопасности

Различные типы *NoSQL* предоставляют множество возможностей, поэтому выбор конкретного типа зависит от требований к системе обработки событий безопасности. Для решения задач мониторинга кибербезопасности можно использовать различные базы данных *NoSQL* в зависимости от конкретных требований и объема данных, например:

1. Документно-ориентированные базы данных хранят данные в формате документов, что особенно полезно при хранении неструктурированных данных о событиях безопасности, таких как логи и сообщения. Эти базы данных обеспечивают гибкость и масштабируемость, а также возможность анализировать данные с помощью запросов на языке запросов к документам (например, *MongoDB*). Например, системы мониторинга событий могут использовать их для хранения логов событий в виде документов, содержащих информацию о времени события, типе события и других атрибутах. Индексы могут быть использованы для быстрого поиска логов по определенным атрибутам.

2. Столбцовые базы данных предоставляют возможность хранить данные в виде столбцов, что особенно полезно при обработке больших объемов

структурированных данных, таких как журналы событий безопасности. Эти базы данных обеспечивают быстрый доступ к данным и масштабируемость, а также возможность анализировать данные с помощью запросов на языке запросов к столбцам (например, *Apache Cassandra*) [13]. Системы управления доступом могут использовать эти базы данных для хранения информации о пользователях, ролях и правах доступа к определенным ресурсам [14].

3. Ключ-значение базы данных позволяют хранить данные в виде пар ключ-значение, что особенно полезно при хранении метаданных о событиях безопасности, таких как время и место события [15]. Также они могут быть использованы для отслеживания сессий пользователей и ассоциированных с ними событий безопасности. Ключ может представлять собой идентификатор сессии, а значение – детали сессии или связанные события безопасности. Эти базы данных обеспечивают высокую производительность и масштабируемость, а также возможность анализировать данные с помощью запросов на языке ключ-значение (например, *Redis*).

4. Графовые базы данных представляют данные и связи в виде графа, что особенно полезно при анализе связей между различными событиями безопасности. Эти базы данных обеспечивают гибкость и масштабируемость, а также возможность анализировать данные с помощью запросов на языке графовых запросов (например, *Neo4j*). Такие базы данных могут быть использованы системами обнаружения вторжений для хранения информации о событиях безопасности и их связях с узлами, представляющими пользователей, ресурсы и другие объекты в системе безопасности. Это позволяет быстро находить связанные события и анализировать их в контексте системы безопасности.

Заключение

В соответствии с проведенным исследованием, итоговым результатом данной статьи является таблица, в которой отображены характеристики исследуемых форматов. В результате исследования описаны преимущества и недостатки форматов *SQL* и *NoSQL* в контексте описания событий безопасности, а также даны рекомендации по использованию форматов. Этот сравнительный анализ позволяет выбрать оптимальный формат для системы в соответствии с её требованиями безопасности.

Помимо вышеперечисленного, дальнейшее развитие исследования в этой области может быть направлено не только на углубление понимания процессов, но и на моделирование надежной защиты информации, хранящейся в базах данных. Это подчеркивает важность выбора правильного формата базы данных для описания событий безопасности.

В целом, использование баз данных и средств автоматизированного анализа данных является важным шагом в обеспечении безопасности информационных систем и защите от кибератак. Описание событий безопасности с помощью верно подобранной БД позволяет выделять общие черты инцидентов, по ним составлять шаблоны, а также выводить необходимую информацию для их анализа и ликвидирования, что во многом повышает эффективность анализа событий и быстрого реагирования на инциденты.

Литература

1. Макаревич О. Б., Шелудько И. А. Регистрация и анализ событий безопасности в информационных системах // Известия ТРТУ, 2003. – № 4 (33). – С. 211-216.

2. Бирих Э. В., Виткова Л. А., Гореленко В. В., Казаков Д. Б. Защита информации в базах данных // В сборнике: Актуальные проблемы инфотелекоммуникаций в науке и образовании. Материалы конференции АПИНО 2017, 2017. – С. 89-92.
3. Ушаков И. А. Обнаружение инсайдеров в компьютерных сетях на основе комбинирования экспертных правил, методов машинного обучения и обработки больших данных: Дис. канд. техн. наук: 05.13.19; [Место защиты: Санкт-Петербургский институт информатики и автоматизации Российской академии наук]. – Санкт-Петербург, 2020. – 32 с.
4. Шелудько И.А. Разработка и исследование системы оперативного сетевого мониторинга событий безопасности: Дис. канд. техн. наук: 05.13.19; [Место защиты: Таганрогский государственный радиотехнический университет]. – Таганрог, 2004. – 179 с.
5. Цифровые технологии и проблемы информационной безопасности / под ред. Е.В. Стельмашонок, И.Н. Васильевой. – СПб.: Изд-во СПбГЭУ, 2021. – 47 с.
6. URLhttps://cc.bingj.com/cache.aspx?q=1+таблица+1+sql+nosql+тип+реляционный+нереляционный+данные+структурированные&d=456867736448676&mkt=en-US&setlang=en-US&w=ut03PNqWZEwdmGPcoiZ_RsXI9-_-UIKr (дата обращения – апрель 2024 г.).
7. URL https://habr.com/ru/companies/otus/articles/760226/ (дата обращения – апрель 2024 г.).
8. URL https://cc.bingj.com/cache.aspx?q=данных+требуют+фиксированной+схемы+могут+масштабироваться+горизонтально+обеспечивают+высокую+производительность&d=4835038350961123&mkt=en-US&setlang=en-US&w=Zb11VPLBF02WuOhch2q4_y9CxOfCDF1F (дата обращения – апрель 2024 г.).
9. URL https://www.guru99.com/ru/nosql-tutorial.html (дата обращения – апрель 2024 г.).
10. URL https://veesp.com/ru/blog/sql-or-nosql/ (дата обращения – апрель 2024 г.).
11. URL https://kartaslov.ru/книги/Ponin_Fedor_Методика_эффективного_управления_данными_в_IT-проектах/3 (дата обращения – апрель 2024 г.).
12. URL https://bpium.ru/blog/chem-baza-dannyh-luchshe-elektronnoy-tablicy (дата обращения – апрель 2024 г.).
13. URL https://www.arsis.ru/blog/nosql (дата обращения – апрель 2024 г.).
14. URL https://www.astera.com/ru/knowledge-center/sql-vs-nosql/ (дата обращения – апрель 2024 г.).
15. URL https://habr.com/ru/companies/otus/articles/562852/ (дата обращения – апрель 2024 г.).