САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Безопасность систем баз данных

Лекция 9
Аудит

Агафонов Антон Александрович
к.т.н., доцент кафедры ГИиИБ

Самара

План лекции

- Аудит
  - Задачи аудита
  - Журнал аудита
  - Методы аудита

- Возможности аудита в СУБД
  - MySQL
  - PostgreSQL
  - MongoDB

Задачи аудита

Аудит событий безопасности БД — процесс получения и анализа данных о происходящих в системе событиях и степени их соответствия требованиям к защите данных. Аудит может отслеживать события, происходящие на различных уровнях. События записываются в журналы событий или файлы аудита.

Когда аудит активен, каждая контролируемая операция с базой данных порождает аудиторский след с информацией о том, какой объект базы данных был задействован, а также кто и когда выполнял эту операцию.

Аудит должен давать ответы на такие вопросы, как:

- Кто получал доступ к данным на чтение или запись?
- Когда данные были изменены?
- Каким было значение данных до изменения?

Комплексный аудиторский след, накопленный в течение определенного отрезка времени, позволяет проводить углубленный анализ данных аудита и строить модели типичного поведения, которые могут быть использованы для автоматического выявления нетипичного поведения.

Журнал аудита

Журнал аудита обеспечивает индивидуальную ответственность пользователя за все его действия, для этого все события аудита однозначно связываются с учетной записью пользователя в СУБД. В журнале содержится:

- описание стандартного набора событий (авторизации пользователя, доступа к тем или иным данным и операций с ними; создания, модификации и уничтожения объектов БД; выполнение нештатных SQL-команд и т. д.);
- настраиваемый перечень атрибутов в отдельной записи журнала аудита (даты и времени события, идентификатор пользователя, имя и сетевой адрес компьютера, описание события, связанные с событием объекты, признак успешного или неудачного завершения события).

Проблемы аудита

Обычно средства аудита позволяют проводить аудит на разных уровнях: на уровне сервера, на уровне БД, на уровне объектов БД, на уровне пользователей и т.д.

Недостатки использования аудита:

- **снижение производительности**: с одной стороны, аудиторский след должен быть достаточно подробным, с другой стороны, сбор такого большого количества информации может отрицательно влиять на производительность системы;
- **объем данных**: чем более подробный аудиторский след требуется, тем больший объем памяти необходим для его хранения.

Желательно иметь возможности гибкой настройки параметров аудита, чтобы минимизировать проблемы с производительностью и хранением.

Методы аудита

- Трассировка
- Анализ журналов транзакций
- Использование темпоральных данных
- Аудиторские следы в данных
- Мониторинг сетевого трафика сервера БД
- Мониторинг сервера БД

Трассировка

Суть трассировки заключается в том, что при активной функции аудита СУБД протоколирует все события, связанные с контролируемыми объектами. Хотя каждая СУБД предоставляет различные возможности настройки аудита, к числу протоколируемых событий, как правило, относятся:

- аутентификация пользователей (как успешная, так и неудачная);
- предоставление прав доступа к БД;
- создание, изменение и удаление объектов БД;
- вставка, редактирование и удаление данных;
- нарушения ссылочной целостности при модификации данных;
- исполнение хранимых процедур;
- изменение настроек сервера и прикладного ПО;
- любые отказы в обслуживании пользователя;
- попытки осуществить потенциально опасные операции без наличия соответствующих прав;
- различного рода исключительные ситуации и ошибки в работе ПО.

Недостатком этого подхода является высокий риск деградации производительности в нагруженных системах.

Анализ журналов транзакций

Этот подход основан на использовании программных средств интерпретации и анализа журналов транзакций, чтобы определить, какие данные были изменены, когда, и какими пользователями.

Основным недостатком этого метода является то, что в журналах транзакции фиксируются только операции модификации данных, а операции чтения или, например, попытки аутентификации на сервере – не фиксируются. Также существуют способы отключения ведения журналов транзакций, либо варианты ведения с неполным протоколированием. В этом случае информация об изменениях данных будет потеряна полностью или частично. Попутно возникает необходимость резервирования журналов для возможности анализа, до того момента как они будут усечены средствами СУБД.

Достоинством данного подхода можно считать отсутствие влияния на производительность системы, т.к. сканирование журналов хотя и достаточно затратный процесс, но выполняется отдельным программным обеспечением и, возможно, на отдельном сервере.

Использование темпоральных данных

Многие современные СУБД поддерживают так называемые *темпоральные* таблицы. В темпоральных таблицах все записи имеют две временных метки, определяющих начало и конец периода актуальности данных.

Операции модификации данных для этих таблиц происходят иначе, чем для обычных: например, при выполнении операции UPDATE происходит не замена значения атрибутов в существующей записи, а корректировка конца периода текущей записи и добавление новой записи с новыми значениями атрибутов и новым периодом актуальности.

Фактически таблица представляет собой исторический набор данных и на любую дату может быть получен актуальный срез данных.

Недостатком метода является то, что отслеживаются только изменения данных и их время, но не то, кем они были сделаны. Также этот метод не может быть использован для отслеживания операций чтения и других потенциально важных событий.

Достоинством данного подхода можно считать незначительное влияние на производительность системы, т.к. он не требует дополнительного протоколирования, а анализ изменений в данном случае хотя и производится средствами СУБД, но достаточно легковесен.

Аудиторские следы в данных

Подход заключается в добавлении необходимых «столбцов аудита» в таблицы, например, `LAST_MODIFIED_DATE` и `LAST_MODIFIED_USER`, которые заполняются на уровне клиентских приложений или на уровне триггеров БД.

Это достаточно проблемное решение, т.к. у администратора, например, есть возможность отключить триггеры, или изменить данные напрямую, а не через клиентское приложение, либо изменить данные в «столбцах аудита».

Как и остальные методы отслеживания изменений, этот метод не может быть использован для отслеживания операций чтения и других потенциально важных событий. Кроме того, хорошей практикой является, когда аудиторский след хранится отдельно от данных, а в данном случае при удалении строки данных будет потерян и аудиторский след.

Влияние данного метода на производительность системы незначительно, но приводит к увеличению объема хранимых данных.

Мониторинг сетевого трафика сервера БД

Подход основан на прослушивании сетевого трафика СУБД. Протоколируются все SQL-инструкции, которые передаются по сети от клиентов к серверу и относятся к контролируемым объектам.

Недостатком данного подхода является то, что отслеживаться могут только те инструкции, которые переданы по сети, т.е. все события, выполненные непосредственно на сервере отследить невозможно. Кроме того, существуют такие реализации клиент-серверного взаимодействия, при которых большая часть операций централизована, и важные транзакции не осуществляются по сети.

Мониторинг сети не потребляет ресурсов СУБД и, соответственно, не оказывает влияния на производительность системы. Также возможна достаточно гибкая настройка аудита на уровне определения интересующих инструкций.

Мониторинг сервера БД

Подход, при котором перехват и анализ SQL-инструкций выполняется не на уровне сети, а непосредственно на уровне сервера. Этот подход позволяет отслеживать все SQL-инструкции, исполняемые ядром СУБД.

Потенциальная опасность такого подхода заключается в том, что в этом случае процесс аудита взаимодействует непосредственно с ядром СУБД, и какие-либо ошибки могут приводить к критическим сбоям в работе СУБД.

Такой подход также не оказывает влияния на производительность системы и допускает гибкую настройку аудита на уровне определения интересующих инструкций.

Общие рекомендации

Стратегия организации аудита сервера баз данных зависит от хранимых данных, происходящих с ними процессов, а также от организационных нормативов и требований в данной предметной области. В любом случае, аудиторский след должен охватывать все конфиденциальные данные, а также ряд событий, которые необходимо отслеживать:

- **Попытки неудачной аутентификации**. Данные события сигнализируют о возможных попытках подбора параметров аутентификации с целью получения доступа к серверу БД.
- **Изменения в системе безопасности**. Создание и удаление пользователей и ролей, изменение членства в ролях, а также отзыв и предоставление привилегий — всё это события, которым стоит уделять внимание.
- **Выполнения резервного копирования**. Аудит событий создания резервных копий позволит отследить несанкционированное выполнение резервных копий и предотвратить или расследовать утечку данных.
- **Действия привилегированных пользователей**. Аудит действий привилегированных пользователей позволит быть уверенным, в том, что они не получают доступ к производственным данным и не вносят изменения в них без производственной необходимости.
- **События подсистемы аудита**. Отслеживание событий, связанных с администрированием подсистемы аудита, позволят быть уверенным в том, что эта подсистема функционирует должным образом.

Возможности аудита в MySQL. Плагины аудита

Плагин аудита *MySQL Enterprise Audit Plugin* в настоящий момент предлагается только в составе коммерческой версии MySQL Enterprise Edition. Плагин предоставляет обширные функции аудита: механизмы фильтрации событий, ведение журнала событий в формате JSON, компрессия и шифрование данных, динамическое изменение конфигурации без перезагрузки БД, вывод информации в формате аудита в соответствии с различными стандартами.

Плагин аудите *Percona Audit Log Plugin* (для СУБД Percona Server for MySQL) позволяет вести журнал событий в различных форматах, а также отправлять события на syslog-сервер. В плагине можно настроить ведение аудита для определенных групп команд и для определенных пользователей, также поддерживается ротация журналов событий.

Плагин аудита *MariaDB Audit Plugin* (для СУБД MariaDB) позволяет вести журнал событий в текстовом формате, поддерживает ротацию журналов событий и отправки событий на syslog-сервер, фильтры по типам команд и пользователям. Плагин поставляется в составе MariaDB, но допускает использование и с Percona Server for MySQL и Oracle MySQL (версии 5.7).

Возможности аудита в MySQL. Анализ серверных журналов

- **Журнал ошибок** (Error Log). Содержит информацию о том, когда был запущен или остановлен экземпляр сервера MySQL, а также о любых критических ошибках в процессе его работы. В задачах аудита этот журнал можно использовать только для обнаружения неудачных попыток аутентификации.
- **Журнал медленных запросов** (Slow Query Log). Содержит данные о SQL-запросах, на выполнение которых ушло времени больше, чем определено переменной `long_query_time`. В этот журнал записываются все медленные запросы, включая SELECT, а также информация о пользователе и времени исполнения.
- **Двоичный журнал** (Binary Log). Содержит только те запросы, которые изменяли данные, поэтому не может быть использован для аудита запросов выборки данных. Тем не менее, этот журнал может быть использован для обнаружения времени модификации данных. Ведение этого журнала не создаст дополнительной нагрузки на сервер, поскольку он, как правило, всегда ведется для задач резервного копирования или репликации.
- **Журнал запросов** (General Query Log). Содержит информацию о подключении и отключении клиентов, а также все SQL-запросы, полученные от клиентов (включая синтаксически правильные, но не выполненные вследствие ошибок). Этот журнал предоставляет наиболее полные данные с точки зрения аудита, но его ведение может оказывать значительное влияние на производительность сервера.

Возможности аудита в MySQL. Анализ производительности

Для отслеживания работы сервера MySQL и настройки его производительности одним из самых полезных инструментов является **журнал медленных запросов** – запросов, время выполнения которых превышает заданное количество секунд.

Настройки логирования:

- `slow-query-log` — определяет состояние журнала медленных запросов (0 – выключен, 1– включен);
- `long_query_time` — определяет минимальное время выполнения запроса в секундах, при котором данный запрос фиксируется в журнале.

Содержимое журнала:

- `Query_time` — время выполнения запроса в секундах;
- `Lock_time` — время получения блокировки в секундах;
- `Rows_sent` — количество строк, отправленных клиенту;
- `Rows_examined` — количество строк, просмотренных на сервере для проверки выполнения условия запроса.

Для анализа данных, представленных в файле журнала медленных запросов, в состав MySQL входит утилита `mysqldumpslow`.

Возможности аудита в PostgreSQL. Протоколирование

В PostgreSQL представлено стандартное средство протоколирования работы сервера. Настройка протоколирования событий управляется множеством параметров, которые позволяют определить метод и формат протоколирования, расположение файлов журналов событий, параметры ротации журналов, типы событий сервера и SQL-выражений, которые необходимо протоколировать и т.д.

Куда протоколировать:

- `log_destination` — указывает метод протоколирования сообщений сервера (`stderr`, `csvlog`, `jsonlog`, `syslog`, `eventlog`). По умолчанию используется `stderr`.
- `logging_collector` — включает сборщик сообщений — фоновый процесс, который собирает отправленные в `stderr` сообщения и перенаправляет их в журнальные файлы.
- `log_directory` — определяет каталог, в котором создаются журнальные файлы.
- `log_filename` — задаёт имена журнальных файлов.

Когда протоколировать:

- `log_min_messages` — управляет минимальным уровнем сообщений, записываемых в журнал сервера. Допустимые значения `DEBUG5`, `DEBUG4`, `DEBUG3`, `DEBUG2`, `DEBUG1`, `INFO`, `NOTICE`, `WARNING`, `ERROR`, `LOG`, `FATAL` и `PANIC`.
- `log_min_error_statement` — управляет тем, какие SQL-операторы, завершившиеся ошибкой, записываются в журнал сервера.
- `log_min_duration_statement` — записывает в журнал продолжительность выполнения всех команд, время работы которых не меньше указанного.

Что протоколировать:

- `log_statement` — задает типы протоколируемых SQL-выражений:
  - `none` — SQL-команды не протоколируются;
  - `ddl` — протоколируются все команды определения данных;
  - `mod` — протоколируются все команды ddl, а также команды модификации данных;
  - `all` — протоколируются все SQL-команды, включая оператор SELECT.

Также в виде строки форматирования можно задать так называемый префикс записи журнала, который выводится в начале каждой строки. Используя различные спецификаторы вывода, в префикс можно включить название текущей БД, имя пользователя или приложения, время события и т.д.

Возможности аудита в PostgreSQL. Расширение PGAudit

Расширение PGAudit обеспечивает детальный аудит событий сессий и/или объектов с помощью стандартной подсистемы протоколирования PostgreSQL, а также позволяет создавать журналы аудита, соответствующие правительственным и финансовым требованиям, а также стандартам ISO.

- **Аудит событий сессии** подразумевает протоколирование всех SQL-выражений, выполненных ядром СУБД в рамках клиентской сессии. Настройка протоколируемых выражений выполняется с помощью перечисления классов команд (`READ`, `WRITE`, `ROLE`, `DDL`, …) в конфигурационных файлах. Причем классы команд могут включаться/отключаться на уровне сервера, базы данных и пользователя.
- **Аудит событий объектов** протоколирует выражения, которые затрагивают конкретную таблицу. Поддерживаются только команды `SELECT`, `INSERT`, `UPDATE` и `DELETE`, команда `TRUNCATE` не протоколируется. Аудит событий объектов основан на системе ролей. Можно определить несколько ролей аудита, что позволит нескольким группам отвечать за различные аспекты ведения журналов аудита.

Возможности аудита в MongoDB

MongoDB Enterprise включает возможность аудита экземпляров `mongod` и `mongos`. Средства аудита позволяют администраторам и пользователям отслеживать активность системы со многими пользователями и приложениями.

Средство аудита может записывать события аудита в консоль, системный журнал, файл JSON или файл BSON.

Чтобы включить аудит в MongoDB Enterprise, необходимо задать назначение вывода событий аудита с помощью параметра `--auditDestination`.

После включения система аудита может записывать следующие операции:

- операции со схемой (DDL-операции);
- операции с набором реплик и сегментированным кластером;
- события аутентификации и авторизации;
- операции модификации данных (требуется установка параметра `auditAuthorizationSuccess`).

С помощью системы аудита можно настроить фильтры для ограничения записываемых событий.

Возможности аудита в MongoDB. Гарантия аудита

Система аудита записывает каждое событие аудита (если не была настроена фильтрация) в буфер событий аудита в памяти. MongoDB периодически записывает этот буфер на диск. Для событий, полученных в рамках любого отдельного соединения, события имеют общий порядок: если одно событие записывается на диск, система гарантирует, что на диск записаны все предыдущие события для этого соединения.

Если запись о событии аудита соответствует операции, влияющей на долговременное состояние базы данных, например, изменение данных, событие аудита всегда записывается на диск перед записью в журнал транзакций для этой операции.

То есть перед добавлением операции в журнал транзакций записываются все события аудита в рамках соединения, которое инициировало операцию, вплоть до записи об операции включительно.

Для обеспечения гарантии записей о событиях аудита требуется, чтобы СУБД работала с включенным ведением журнала транзакций.

Кроме того, если сервер не может выполнить запись в журнал аудита, сервер прекратит работу.

MongoDB. Настройка аудита

Включение аудита с записью событий журнала в консоль:

```
storage:
     dbPath: data/db
auditLog:
     destination: console  # syslog
```

Включение аудита с записью событий журнала в JSON-файл:

```
storage:
     dbPath: data/db
auditLog:
     destination: file
     format: JSON  # BSON
     path: data/db/auditLog.json
```

Фильтры аудита можно указать в командной строке или в файле конфигурации, используемом для запуска экземпляра mongod или mongos.

- **Фильтр для нескольких типов операций**

```
{ atype: { $in: [ "createCollection", "dropCollection" ] } }
```

```
storage:
     dbPath: data/db
auditLog:
     destination: file
     format: BSON
     path: data/db/auditLog.json
     filter: '{ atype: { $in: [ "createCollection", "dropCollection" ] } }'
```

- **Фильтрация операций аутентификации в одной базе данных**

Поле <field> может включать любое поле в сообщении аудита.

Для операций аутентификации сообщения аудита включают поле db в документе param.

```
{ atype: "authenticate", "param.db": "test" }
```

MongoDB. Настройка фильтрации

- **Фильтрация операций создания и удаления коллекций для одной базы данных**

```
{ atype: { $in: [ "createCollection", "dropCollection" ] }, "param.ns": /^test\\./ } }
```

- **Фильтр по роли авторизации**

```
{ roles: { role: "readWrite", db: "test" } }
```

- **Фильтрация операций чтения и записи**

Чтобы фиксировать операции чтения и записи в аудите, необходимо также разрешить системе аудита регистрировать события авторизации с помощью параметра `auditAuthorizationSuccess`.

```
{
   atype: "authCheck",
   "param.command": { $in: [ "find", "insert", "delete", "update", "findandmodify" ] }
}
```

MongoDB. Сообщения аудита

Функция аудита событий может записывать события в формате JSON.

Сообщения аудита имеют следующий синтаксис:

```
{
  atype: <string>,
  ts : { $date: <timestamp> },
  uuid : { $binary: <string>, $type: <string> },
  local: { ip: <string>, port: <int> || isSystemUser: <boolean> || unix: <string> },
  remote: { ip: <string>, port: <int> || isSystemUser: <boolean> || unix: <string> },
  users : [ { user: <string>, db: <string> }, ... ],
  roles: [ { role: <string>, db: <string> }, ... ],
  param: <document>,
  result: <int>
}
```

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
к.т.н., доцент кафедры ГИиИБ