

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ
БАЗАМИ ДАННЫХ «ЈАТОВА»

Руководство по настройке. Часть 17.
Выявление и предотвращение исполнения нетипичных
SQL-запросов.
Компонент «SQL_Firewall»

643.72410666.00067-07 98 01-17

Листов 26

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

В документе приведены сведения, необходимые для установки и эксплуатации компонента «SQL_Firewall» (далее – компонент или SQL_Firewall). Настоящее руководство предназначено для администратора защищенной системы управления базами данных (СУБД) «Jatoba».

Администратор СУБД «Jatoba» должен иметь навыки по работе с СУБД PostgreSQL или защищенной СУБД Jatoba (ООО «Газинформсервис»).



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра 4.x, для других версий все шаги выполняются аналогично, разница состоит в именах директорий.

Например, СУБД «Jatoba» версии 6.x по умолчанию устанавливается в директорию ОС Linux – «/usr/jatoba-6/bin».

Для СУБД «Jatoba» версии ядра 4, 5 и 6 используется версия компонента — 0.8

Для СУБД «Jatoba» версии ядра 18 используется версия компонента — 1.0



Важная информация

Для сертифицированной версии СУБД «Jatoba» поддерживается работа только на ОС, указанных в формуляре на поставку!

Степени важности примечаний, применяемые в документе:



Важная информация – указания, требующие особого внимания



Дополнительная информация – указания, позволяющие упростить работу с изделием

СОДЕРЖАНИЕ

1. Назначение компонента.....	4
1.1. Условия применения.....	4
2. Установка и настройка.....	7
2.1. Установка компонента «SQL_Firewall» в ОС GNU/Linux.....	7
2.2. Настройка конфигурационного файла postgresql.conf.....	8
2.3. Установка расширения компонента «SQL_Firewall».....	8
3. Функциональные возможности	10
3.1. Режимы функционирования компонента	10
3.1.1. Режим обучения «learning»	10
3.1.2. Режим применения «enforcing».....	11
3.1.3. Режим разрешающий «permissive»	12
3.1.4. Режим отключения компонента «disabled»	12
3.2. Режим обнаружения SQL-инъекций.....	13
3.2.1. Принцип работы ML-модели в компоненте SQL Firewall	13
3.2.2. Параметры режима обнаружения SQL-инъекций	15
3.2.3. Активация режима обнаружения SQL-инъекций.....	15
3.2.4. Сочетание режимов работы SQL Firewall при обнаружении SQL-инъекций.....	16
3.2.5. Проверка режима обнаружения SQL-инъекций	18
4. Описание операций.....	19
4.1. Функции просмотра	19
4.1.1. Просмотр правил брандмауэра (sql_firewall.sql_firewall_statements)	19
4.1.2. Просмотр статистики (sql_firewall.sql_firewall_stat)	20
4.2. Управление функциями мониторинга	20
4.2.1. Экспорт правил компонента (sql_firewall_export_rule)	20
4.2.2. Импорт правил компонента (sql_firewall_import_rule).....	21
4.2.3. Очистка правил (sql_firewall_reset).....	22
4.2.4. Очистка предупреждений и ошибок (sql_firewall_stat_reset).....	22
5. Удаление	24
Перечень сокращений.....	25

1. НАЗНАЧЕНИЕ КОМПОНЕНТА

Компонент SQL_Firewall просматривает запросы к СУБД, которые могут быть выполнены, и предотвращает либо предупреждает о выполнении запросов, которые не найдены в установленных правилах («белых списках», WhiteList) по аналогии с брандмауэром, т.е. фильтрует трафик SQL-запросов.



Компонент SQL_Firewall не является межсетевым экраном (firewall) в классическом понимании, как программный или программно-аппаратный элемент компьютерной сети, применяемого для фильтрации сетевого трафика.

Компонент SQL_Firewall предназначен для защиты базы данных от SQL-инъекций или неожиданных запросов. Входящий запрос перед исполнением подается на обработку в обученную модель машинного обучения, которая возвращает в результате одно значение – вероятность содержания во входящем запросе SQL-инъекции. После этого вычисленная вероятность сравнивается со значением порога классификации, устанавливаемым с помощью параметра СУБД. Запросы с превышенной вероятностью наличия SQL-инъекций прерываются с выводом ошибки.

1.1. Условия применения

Компонент «SQL_Firewall» может использоваться совместно с СУБД «Jatoba» версии 1.x и выше.

В текущей реализации не поддерживается управление с помощью пользовательского веб-интерфейса для администраторов «Jatoba data safe».

Режим обнаружения SQL-инъекций (см. п. 3.2) доступен только для версии компонента «SQL_Firewall» версии 1.0 для СУБД «Jatoba» версии 18.



Список ОС, на которых доступен режим обнаружения SQL-инъекций:

- Astra Linux 1.8;
- Debian 11;
- Debian 12;
- АЛЬТ 10 СП;
- АЛЬТ 10 Server;
- Ubuntu 22.04;

- Ubuntu 24.04;
- РЕДОС 7.3;
- РЕДОС 8;
- РОСА 12.4.

Применение режима обнаружения SQL-инъекций может приводить к снижению производительности СУБД.

Суммарное влияние SQL Firewall (обнаружение SQL-инъекций + базовые функции фильтрации) на снижение производительности может составлять:

- в части влияния на TPS - 39,8%, в случае если в белом списке нет запросов из теста и 6,1%, если белый список содержит все запросы теста производительности;
- в части метрики Latency - 3,14 мс и 0,21 мс, соответственно.

Влияние отдельных компонентов SQL Firewall на снижение производительности в проведенных может составить:

- в части обнаружения SQL-инъекций - до 6,6% TPS и до 0,31 мс Latency (только для ОС, поддерживающих ML-модель);
- в части использования базовых функций фильтрации SQL Firewall (белые списки) - до 38,6% TPS и 3,1 мс Latency, если в белом списке нет запросов из теста и до 1,4% TPS и 0,03 мс Latency, если белый список содержит все запросы теста производительности.

Настройка мониторинга компонента SQL Firewall приведена в документе «Поддержка мониторинга СУБД» 643.72410666.00067-07 98 01-28.

При использовании в компоненте SQL Firewall режима обнаружения SQL-инъекций для библиотеки ONNX Runtime (поставляется в составе установочного пакета компонента) в ОС должна быть обеспечена поддержка локали en_US.UTF8. Вывод список доступных локалей в ОС осуществляется при помощи команды:

```
# locale -a
```

В случае, если поддержка локали en_US.UTF8 отсутствует в ОС, то необходимо сгенерировать локаль при помощи следующих команд:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

```
# locale-gen en_US  
# locale-gen en_US.UTF-8  
# update-locale
```

2. УСТАНОВКА И НАСТРОЙКА

Установка компонента должна производиться от имени пользователя, обладающего административными привилегиями в системе. Компонент штатным образом может быть установлен только с СУБД «Jatoba» (см. документ «Защищенная система управления базами данных «Jatoba». Руководство по установке).

2.1. Установка компонента «SQL_Firewall» в ОС GNU/Linux

Установка компонента осуществляется в процессе установки СУБД «Jatoba», также компонент можно установить опционально после основной инсталляции СУБД.

Установку компонента возможно провести двумя способами:

- 1) установка из локального репозитория (CDROM) – производится из файлов, записанных на компакт-диск или скопированных с него;
- 2) установка непосредственно из deb/rpm-файлов – производится опционально, по усмотрению пользователя.

Компонент выполнен в виде отдельного deb или rpm-пакета. Установка компонента осуществляется средствами пакетного менеджера ОС. Для разных типов пакетных менеджеров команда установки немного отличается. Ниже приведены основные типы:

- для систем на основе пакетного менеджера APT (к таким системам относятся все ОС семейства Debian, использующие deb-пакеты) команда установки следующая:

```
apt-get install jatoba<ver>-sql-firewall
```

Здесь и далее <ver> - номер используемой версии СУБД «Jatoba».

- для систем на основе пакетных менеджеров YUM/DNF (к таким системам относятся все ОС семейства RedHat и вышедшие из нее, использующие rpm-пакеты) команда установки следующая:

```
yum install jatoba<ver>-sql-firewall
```

Отдельного уточнения требуют операционные системы ALT Linux.

- ALT Linux использует пакетный менеджер APT, но распространяется в виде rpm-пакетов и для нее команда установки выглядит аналогично Debian:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

```
apt-get install jatoba<ver>-sql-firewall
```

Установка компонента в составе других версий СУБД «Jatoba» осуществляется аналогично. Отличие будет только в номере версии СУБД, в составе которой он распространяется. Например, jatoba5-sql-firewall и т.п.

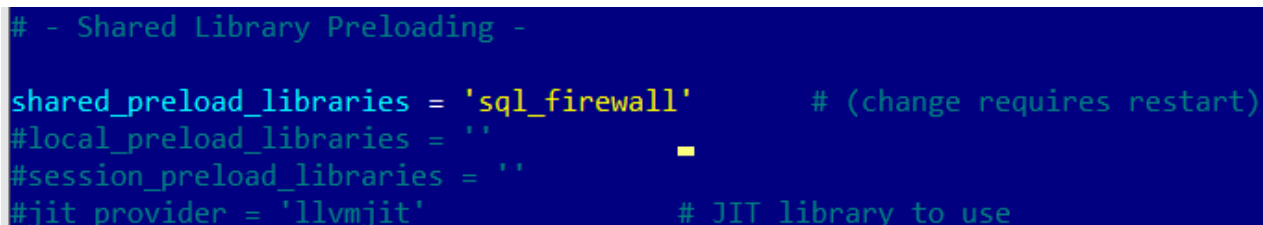
Удаление модуля также осуществляется средствами пакетного менеджера ОС. Вместо команды install нужно использовать соответствующую данному пакетному менеджеру команду удаления (remove, purge, erase и т.п.).

Для получения детальной информации по пакетному менеджеру рекомендуется обратиться к документации по ОС.

2.2. Настройка конфигурационного файла postgresql.conf

В разделе «Shared Library Preloading», для последующей загрузки расширения, установить параметр:

```
shared_preload_libraries = 'sql_firewall'
```



```
# - Shared Library Preloading -
shared_preload_libraries = 'sql_firewall'          # (change requires restart)
#local_preload_libraries = ''
#session_preload_libraries = ''
#jit_provider = 'llvmjit'                          # JIT library to use
```

Рисунок 2.1 – Параметр загрузки расширения



В случае, если в параметре shared_preload_libraries также используется значение pg_stat_statements, то значение sql_firewall должно быть в обязательном порядке через запятую установлено после него:

```
shared_preload_libraries = 'pg_stat_statements,
sql_firewall'
```

Для применения параметров потребуется перезапустить СУБД.

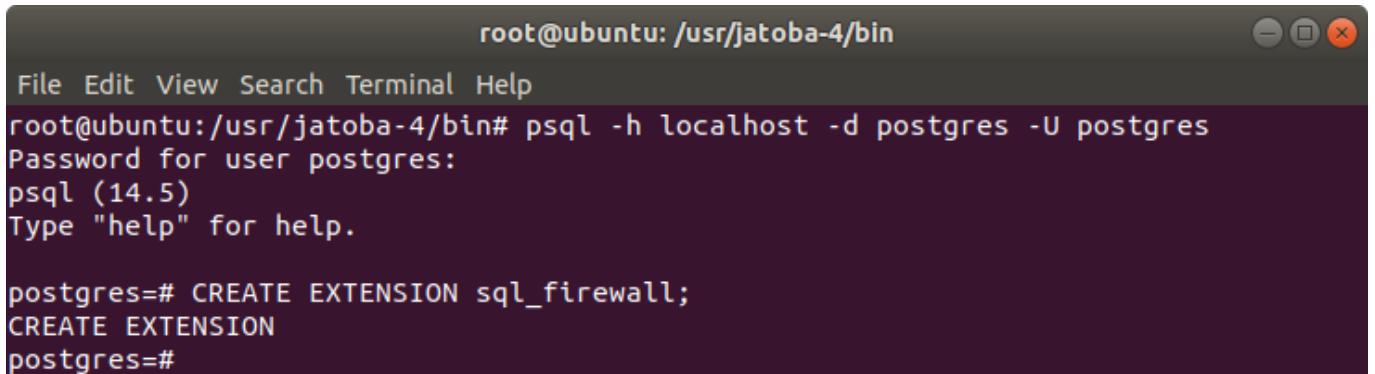
2.3. Установка расширения компонента «SQL_Firewall»

После перезагрузки СУБД и загрузки расширения станет доступной установка расширения «SQL_Firewall».

Расширение устанавливается на всех узлах кластера при помощи SQL-команды:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------


```
CREATE EXTENSION sql_firewall;
```



```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
root@ubuntu:/usr/jatoba-4/bin# psql -h localhost -d postgres -U postgres
Password for user postgres:
psql (14.5)
Type "help" for help.

postgres=# CREATE EXTENSION sql_firewall;
CREATE EXTENSION
postgres=#
```

Рисунок 2.2 – Команда установки расширения в ОС GNU/Linux

В результате выполнения SQL-команды будет создано расширение (extension) «sql_firewall» в схеме данных.

3. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ

Компонент «SQL_Firewall» выполнен в форме расширения СУБД и предназначен для фильтрации SQL-запросов (далее – «запрос»), поступающих в СУБД.

Фильтрация выполняется в два этапа:

- 1) определение наличия SQL-инъекции в запросе. В случае наличия выполняется блокировка поступившего запроса в соответствии с указанными параметрами. Определение наличия SQL-инъекции выполняется с помощью обученной ML-модели (см. п. 3.2).
- 2) определяется наличие запроса в перечне разрешенных для выполнения запросов.

3.1. Режимы функционирования компонента

Компонент «SQL_Firewall» функционирует в следующих режимах, указанных в параметре `sql_firewall.firewall`:

- "learning" – режим обучения;
- "enforcing" – режим применения;
- "permissive" – режим разрешающий любые SQL запросы;
- "disabled" – режим отключенного модуля.



Перед установкой параметра режима компонента `sql_firewall.firewall` в `enforcing` или `permissive` необходимо активировать мониторинг всех метрик СУБД в режиме `learning` для того, чтобы используемые запросы вошли в состав разрешённых.

3.1.1. Режим обучения «learning»

В режиме обучения модуль собирает информацию о пользователе и его запросах. Собирает, записывает и связывает информацию об идентификаторе пользователя «`userid`» и идентификаторе SQL-запроса «`query id`».

«`query id`» устанавливается по дереву синтаксического анализа, аналогично `pg_stat_statements`.

Режим обучения инициализируется установкой параметра:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

```
sql_firewall.firewall = 'learning'
```

в конфигурационном файле СУБД postgresql.conf и последующей перезагрузкой СУБД.

```
# - Shared Library Preloading -  
  
shared_preload_libraries = 'sql_firewall'      # (change requires restart)  
sql_firewall.firewall = 'learning'  
#local_preload_libraries = ''  
#session_preload_libraries = ''  
#jit_provider = 'llvmjit'                      # JIT library to use
```

Рисунок 3.1 – Инициализация режима обучения

По умолчанию в режиме обучения может обработаться 5000 запросов. Запросы превышающие данное значение не будут изучаться.

Параметр изучаемых запросов может быть установлен от 100 до 5000, который устанавливается в конфигурационном файле СУБД postgresql.conf

```
# - Shared Library Preloading -  
  
#local_preload_libraries = ''  
#session_preload_libraries = ''  
shared_preload_libraries = 'sql_firewall'      # (change requires restart)  
sql_firewall.firewall = 'disabled'  
sql_firewall.firewall = 5000
```

Рисунок 3.2 – Установка параметра количества обрабатываемых SQL - запросов

3.1.2. Режим применения «enforcing»

В режиме применения «enforcing», модуль проверяет находятся ли идентификатор пользователя «userid» и его SQL-запрос «query id» в паре, которая была ранее записана в режиме обучения «learning».

Если обнаруживается несоответствие, то пользователю выдается предупреждение о предотвращении выполнения.

Режим применения инициализируется установкой параметра:

```
sql_firewall.firewall = 'enforcing'
```

в конфигурационном файле СУБД postgresql.conf и последующей перезагрузкой СУБД.

```
# - Shared Library Preloading -  
  
shared_preload_libraries = 'sql_firewall'          # (change requires restart)  
sql_firewall.firewall = 'enforcing'  
#local_preload_libraries = ''  
#session_preload_libraries = ''  
#jit_provider = 'llvmjit'                          # JIT library to use
```

Рисунок 3.3 – Инициализация режима применения

3.1.3. Режим разрешающий «permissive»

В разрешающем режиме «permissive» модуль проверяет все запросы, но позволяет их выполнение, даже в случае, если таких запросов нет в правилах модуля. При несовпадениях выполняет запрос и сигнализирует пользователю о выявленном несоответствии.

Режим разрешения инициализируется установкой параметра:

```
sql_firewall.firewall = 'permissive'
```

в конфигурационном файле СУБД postgresql.conf и последующей перезагрузкой СУБД.

```
# - Shared Library Preloading -  
  
shared_preload_libraries = 'sql_firewall'          # (change requires restart)  
sql_firewall.firewall = 'permissive'  
#local_preload_libraries = ''
```

Рисунок 3.4 – Инициализация разрешающего режима

3.1.4. Режим отключения компонента «disabled»

В данном режиме модуль отключен и не проводит ни каких действий и включен по умолчанию.

Режим отключения компонента инициализируется установкой параметра:

```
sql_firewall.firewall = 'disabled'
```

в конфигурационном файле СУБД postgresql.conf и последующей перезагрузкой СУБД.

```
# - Shared Library Preloading -  
  
shared_preload_libraries = 'sql_firewall'      # (change requires restart)  
sql_firewall.firewall = 'disabled'  
#local_preload_libraries = ''  
#session_preload_libraries = ''  
#jit_provider = 'llvmjit'                     # JIT library to use
```

Рисунок 3.5 – Инициализация отключения модуля

3.2. Режим обнаружения SQL-инъекций

В данном режиме компонент SQL Firewall реализует функционал обнаружения SQL-инъекций в входящих запросах к СУБД.

SQL-инъекция – это уязвимость, которая позволяет атакующей стороне использовать фрагмент вредоносного кода на языке SQL для получения доступа к потенциально ценной (коммерческой, конфиденциальной, персональной) информации и манипулирования СУБД.

Обнаружение SQL-инъекций производится с помощью обученной ML-модели (Machine Learning, машинное обучение), принцип которой приводится в п. 3.2.1.

3.2.1. Принцип работы ML-модели в компоненте SQL Firewall

Модель машинного обучения обучена распознавать широкий спектр техник SQL-инъекций, включая, но не ограничиваясь: IN-BAND (включая ERROR-BASED и UNION-BASED), BLIND (включая TIME-BASED) и OUT OF BAND атаки.

ML-модель компонента SQL Firewall детектирует следующие основные типы SQL-инъекций:

1) IN-BAND SQLi – для атаки и получения данных используется один канал. К этому типу относятся:

- ERROR-BASED (на основе ошибок) – вид атаки, при проведении которой нарушители используют сообщения об ошибках сервера базы данных для разведки и определения её структуры. В результате атаки будет передана ошибка, в тексте которой будет присутствовать полная версия используемой СУБД;

```
AND (SELECT version())::int=1 -- -
```

- UNION-BASED – предоставляет возможность получения информации с помощью оператора UNION для объединения результатов двух или более запросов SELECT. В ходе проведения атаки позволяет определить количество столбцов в исходном запросе. При совпадении количества NULL с количеством столбцов запрос выполнится, иначе возникнет ошибка.

```
UNION ALL SELECT NULL, NULL, NULL; -- -
```

2) BLIND SQLi («слепая» SQL-инъекция) – вид атаки, при которой ответ от базы данных можно получить лишь в формате «да»/«нет» (true/false) или через анализ времени ответа. Применение такой инъекции позволяет атакующему определить тип используемой базы данных: если это СУБД «Jatoba», то значение выражения будет True, иначе False.

```
' AND substr(version(),1,10) = 'PostgreSQL' and '1
```

3) TIME-BASED (на основе времени ответа) – еще один тип «слепой» SQL-инъекций, при которой атакующий специально заставляют СУБД задержать ответ на запрос на несколько секунд. Таким образом злоумышленник может понять корректность составленного запроса.

```
SELECT CASE WHEN substring(datname,1,1)='1' THEN pg_sleep(5)  
ELSE pg_sleep(0) END FROM pg_database LIMIT 1;
```

4) OUT OF BAND – атакующий отправляет SQL-запросы к СУБД способом, отличным от обычного взаимодействия. В качестве примера – использование DNS- или HTTP-запросов.

Входящий запрос перед исполнением подается на обработку в ML-модель, которая возвращает в результате одно значение – вероятность (P) содержания во входящем запросе SQL-инъекции.

После этого значение P сравнивается с пороговым значением T , которое устанавливает администратором (см. далее п. 3.2.2).

При $P \leq T$ запросы считаются безопасными и допускаются к выполнению.

При $P > T$ запрос классифицируется как содержащий SQL-инъекцию, и его выполнение прерывается с ошибкой.

3.2.2. Параметры режима обнаружения SQL-инъекций

Параметры режима обнаружения SQL-инъекций определяются в конфигурационном файле postgresql.conf.

sql_firewall.ml_enabled – параметр отвечает за работу алгоритма определения SQL-инъекций.

Тип: boolean

При значении «true» режим обнаружения SQL-инъекций включён, при «false» - выключен.

Значение по умолчанию: false

sql_firewall.ml_score_threshold – параметр определяет предельный порог T , при превышении которого обработанный запрос с значением P будет классифицироваться как содержащий SQL-инъекцию.

Тип: float

Значение по умолчанию: 0.5.

Пример.

Установлено стандартное значение порога $ml_score_threshold = 0.5$

В СУБД поступают два запроса, их вычисленное значение P — 0.25 и 0.75 соответственно.

В этом случае, первый запрос будет считаться безопасным, а второй – SQL-инъекцией, и он будет заблокирован компонентом.

3.2.3. Активация режима обнаружения SQL-инъекций

Режим обнаружения SQL-инъекций инициализируется установкой параметров:

```
shared_preload_libraries = 'sql_firewall'  
sql_firewall.ml_enabled = 'true'
```

в конфигурационном файле СУБД postgresql.conf и последующей перезагрузкой СУБД.

Также для работы режима обнаружения SQL-инъекций необходимо установленное расширение в СУБД (см. п. 2.3).

3.2.4. Сочетание режимов работы SQL Firewall при обнаружении SQL-инъекций

Таблица 3.1 – Режимы работы SQL Firewall при обнаружении SQL-инъекций

№	sqli_catcher (ml_enabled)	SQL Firewall (firewall)	Комментарий
1	False	disabled	
2	False	learning	
3	False	permissive	
4	False	enforcing	
5	True	disabled	Только обнаружение SQL-инъекций
6	True	learning	Обучение выполняется не для обнаружения SQL-инъекций
7	True	permissive	Только для запросов, не являющихся SQL-инъекции. В случае, если запрос не входит в белый список SQL Firewall, выдается соответствующее предупреждение
8	True	enforcing	Только для запросов: 1) не являющихся SQL-инъекциями 2) входящих в белый список SQL Firewall

Схема взаимодействия и обработки SQL-запросов в различных режимах работы компонента SQL Firewall представлена на рисунке 3.6.

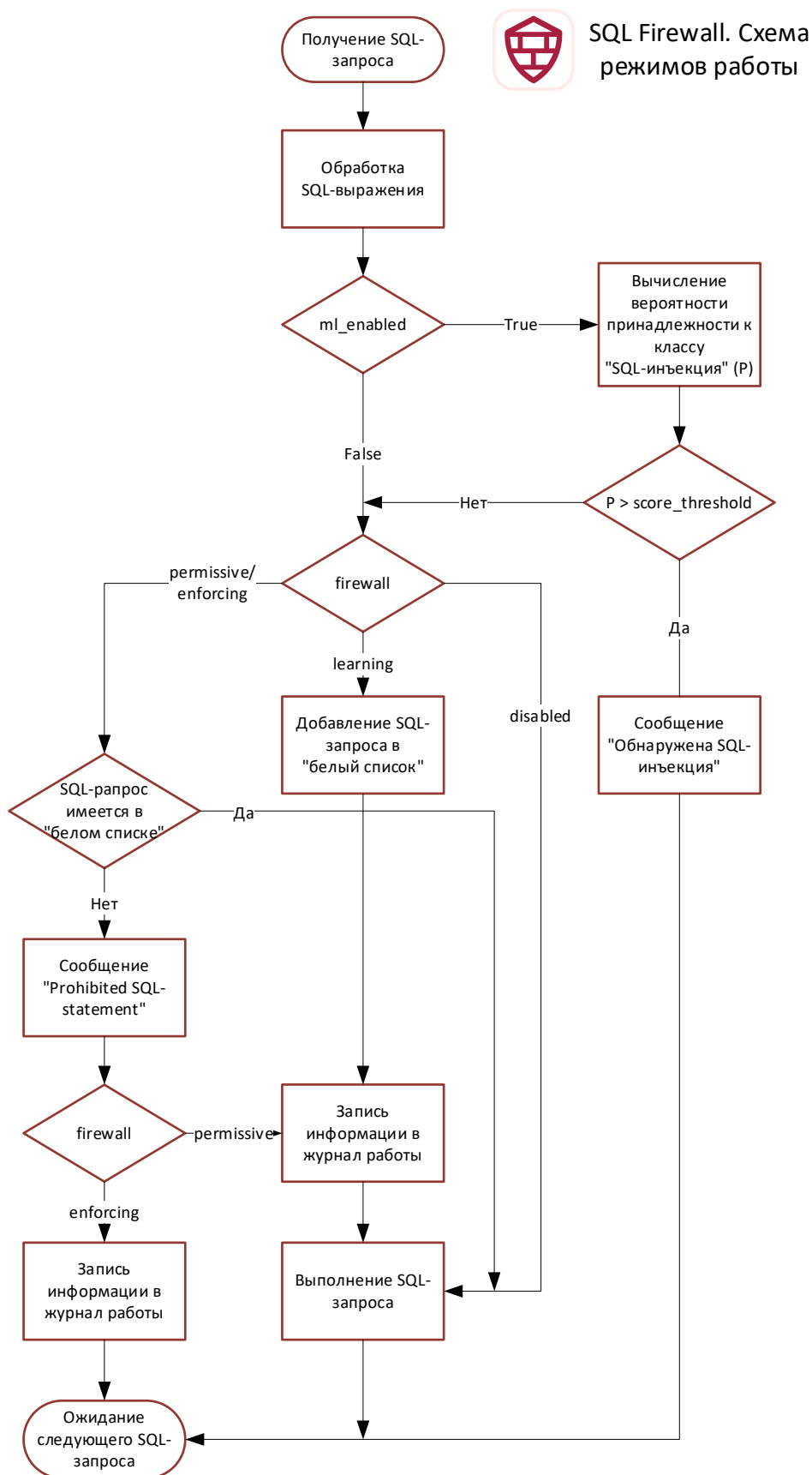


Рисунок 3.6 – Схема взаимодействия и обработки SQL-запросов в различных режимах работы компонента SQL Firewall

3.2.5. Проверка режима обнаружения SQL-инъекций

После выполнения настройки режима обнаружения SQL-инъекций необходимо выполнить несколько тестовых запросов.

Тестовые запросы составлены специальным образом и содержат наглядные примеры инъекций:

```
SELECT * FROM tbl WHERE name='admin' OR 1=1 UNION ALL SELECT  
NULL, NULL, NULL;  
  
SELECT * FROM tbl WHERE name='user' AND 1=CAST((SELECT  
concat('DATABASE: ',current_database())) AS int) AND '1'='1';
```

Данные запросы должны быть заблокированы компонента SQL Firewall с выводом следующего сообщения:

ОШИБКА: Обнаружена SQL-инъекция! Выполнение запроса прервано

4. ОПИСАНИЕ ОПЕРАЦИЙ

4.1. Функции просмотра

Компонент обладает функциональными возможностями просмотра:

- правил брандмауэра;
- статистики брандмауэра.

4.1.1. Просмотр правил брандмауэра (sql_firewall.sql_firewall_statements)

Представление sql_firewall_statements показывает правила брандмауэра и счетчик выполнения для каждого запроса.

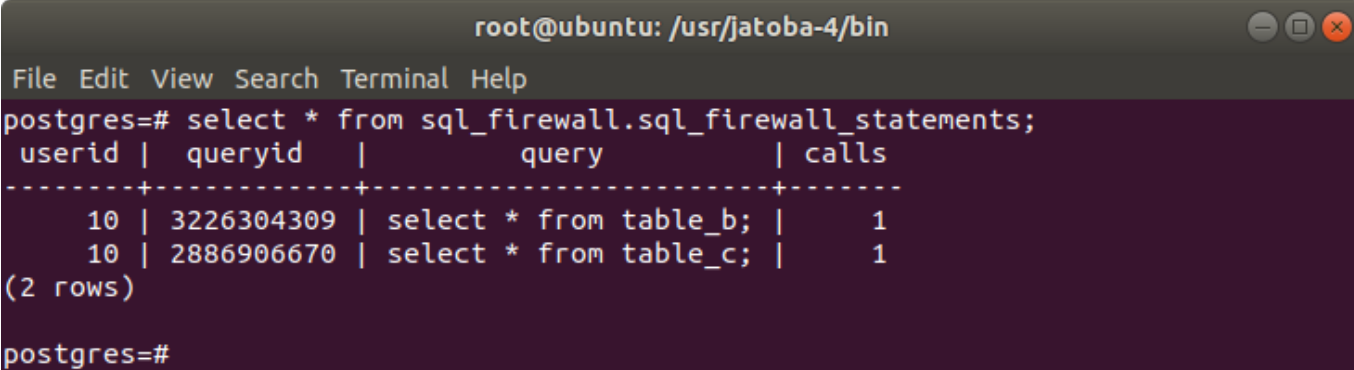
Просмотр правил брандмауэра выполняется SQL-командой:

```
SELECT * from sql_firewall.sql_firewall_statements;
```

Данную SQL-команду целесообразно использовать для специального программного обеспечения по управлению СУБД.

При работе в CLI целесообразно изменить ее и использовать ограничение количества символов в поле «query» оператором «substring»:

```
SELECT userid, queryid, calls, substring (query from 0 for 90)  
from sql_firewall.sql_firewall_statements;
```



The screenshot shows a terminal window titled 'root@ubuntu: /usr/jatoba-4/bin'. The user has entered the command 'postgres=# select * from sql_firewall.sql_firewall_statements;'. The output is a table with four columns: 'userid', 'queryid', 'query', and 'calls'. There are two rows of data. The first row shows 'userid' 10, 'queryid' 3226304309, 'query' 'select * from table_b;', and 'calls' 1. The second row shows 'userid' 10, 'queryid' 2886906670, 'query' 'select * from table_c;', and 'calls' 1. The output is followed by '(2 rows)' and the prompt 'postgres=#'.

userid	queryid	query	calls
10	3226304309	select * from table_b;	1
10	2886906670	select * from table_c;	1

Рисунок 4.1 – Просмотр правил брандмауэра в ОС GNU/Linux

В полученном списке отражены поля:

- userid – идентификационный номер пользователя (идентификационный номер 10 присваивается роли postgres);
- Queryid – идентификационный номер запроса;

- Query – тело запроса;
- Calls – вызовы.

4.1.2. Просмотр статистики (sql_firewall.sql_firewall_stat)

В представлении sql_firewall_stat есть два счетчика: "sql_warning" и "sql_error".

«sql_warning» показывает количество выполненных запросов с предупреждениями в «разрешающем» режиме ([permissive](#)).

«sql_error» показывает количество предотвращенных запросов в «применительном» режиме ([enforcing](#)).

Просмотр статистики выполняется SQL-командой:

```
SELECT * from sql_firewall.sql_firewall_stat;
```

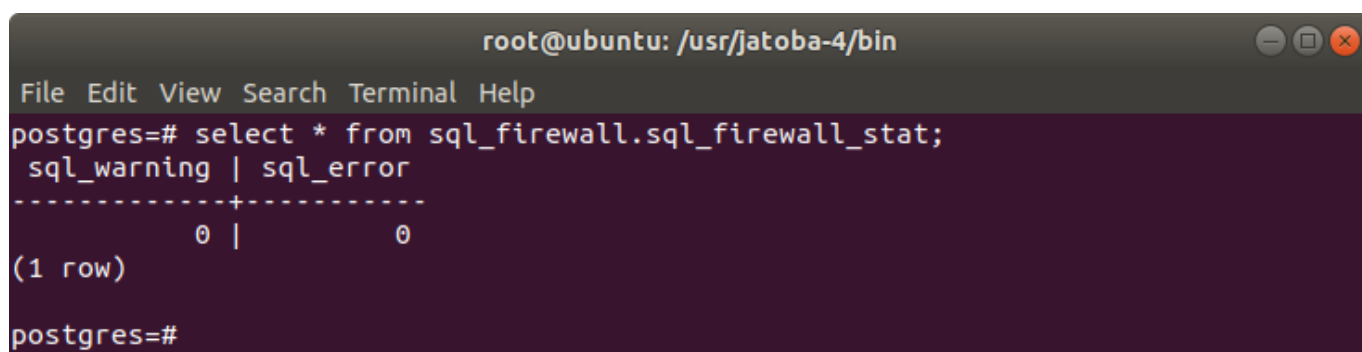


Рисунок 4.2 – Просмотр статистики в ОС GNU/Linux

4.2. Управление функциями мониторинга

Компонент обладает функциональными возможностями управления правилами, такими как:

- экспорт правил (п. 4.2.1);
- импорт правил (п. 4.2.2);
- очистка правил (п. 4.2.3);
- очистка предупреждений и ошибок (п. 4.2.4).

4.2.1. Экспорт правил компонента (sql_firewall_export_rule)

Функциональная возможность экспорта правил компонента «sql_firewall» доступна в режиме «[disabled](#)» от имени и с правами привилегированного пользователя с атрибутом «Superuser».

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

SQL-команда экспорта правил SQL-брандмауэра имеет синтаксис:

```
sql_firewall_export_rule('/path/to/rule.txt')
```

Для предотвращения ошибки доступа к файлу рекомендуется размещать его в директории DATA, где у системной учетной записи postgres есть полные права на чтение и запись.

В рассматриваемом примере для ОС GNU/Linux файл rule.txt расположен по пути:

```
/var/lib/jatoba/4/data/rule.txt
```

SQL-команда будет иметь вид:

```
SELECT  
sql_firewall_export_rule('/var/lib/jatoba/<ver>/data/rule.txt')  
;
```

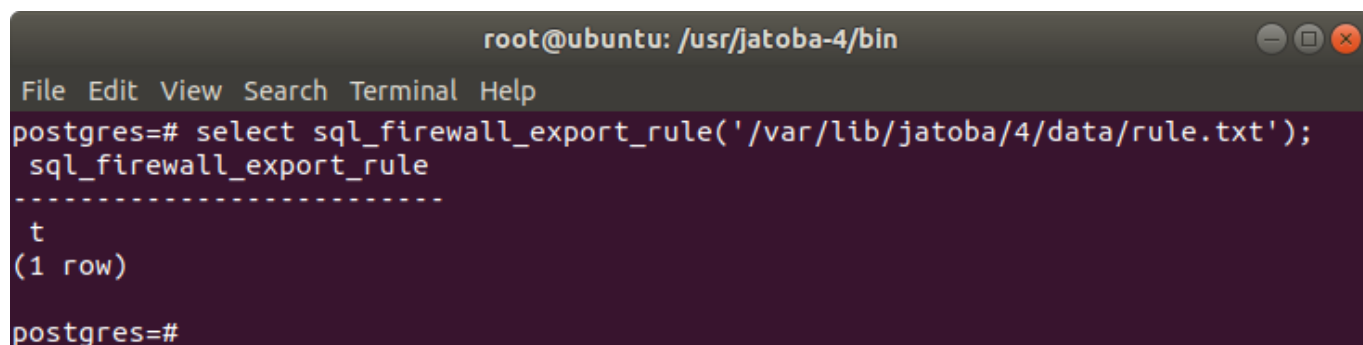


Рисунок 4.3 – Выполнение команды экспорта правил в ОС GNU/Linux

4.2.2. Импорт правил компонента (sql_firewall_import_rule)

Функциональная возможность импорта правил компонента «sql_firewall» доступна в режиме «[disabled](#)» от имени и с правами привилегированного пользователя с атрибутом «Superuser».

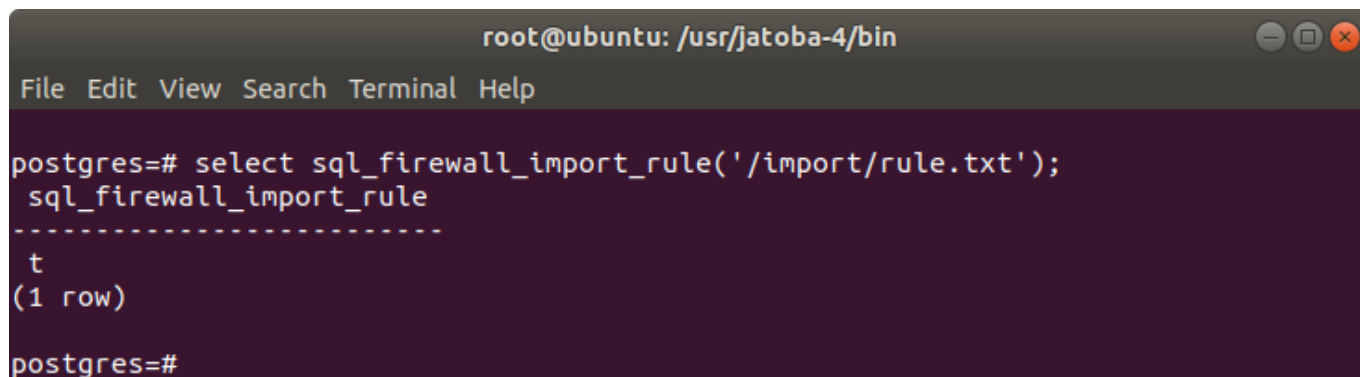
SQL-команда экспорта правил SQL-брандмауэра имеет синтаксис:

```
sql_firewall_import_rule('/path/to/rule.txt')
```

В рассматриваемом примере файл rule.txt расположен в директории «import».

В ОС GNU/Linux SQL-команда будет иметь вид:

```
SELECT sql_firewall_import_rule('/import/rule.txt');
```



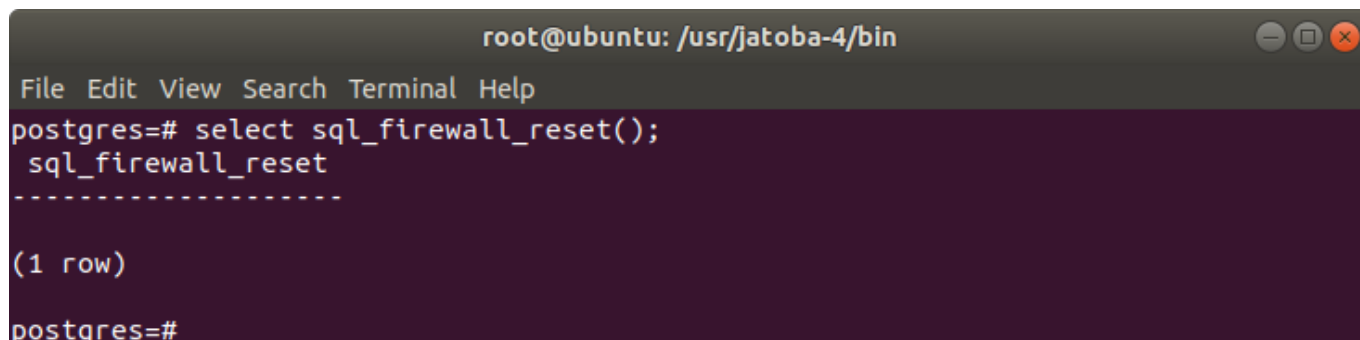
The screenshot shows a terminal window titled 'root@ubuntu: /usr/jatoba-4/bin'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command 'select sql_firewall_import_rule('/import/rule.txt');' is entered at the 'postgres=#' prompt. The output shows the command name 'sql_firewall_import_rule' followed by a separator line '-----', then the letter 't', and finally '(1 row)'. The prompt returns to 'postgres=#'.

Рисунок 4.4 – Выполнение команды импорта правил в ОС GNU/Linux

4.2.3. Очистка правил (sql_firewall_reset)

Функциональная возможность очистки правил компонента «sql_firewall» доступна в режиме «[disabled](#)» от имени и с правами привилегированного пользователя с атрибутом «Superuser» и выполняется SQL-командой:

```
SELECT sql_firewall_reset();
```



The screenshot shows a terminal window titled 'root@ubuntu: /usr/jatoba-4/bin'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command 'select sql_firewall_reset();' is entered at the 'postgres=#' prompt. The output shows the command name 'sql_firewall_reset' followed by a separator line '-----', and finally '(1 row)'. The prompt returns to 'postgres=#'.

Рисунок 4.5 – SQL-команда очистки правил в ОС GNU/Linux

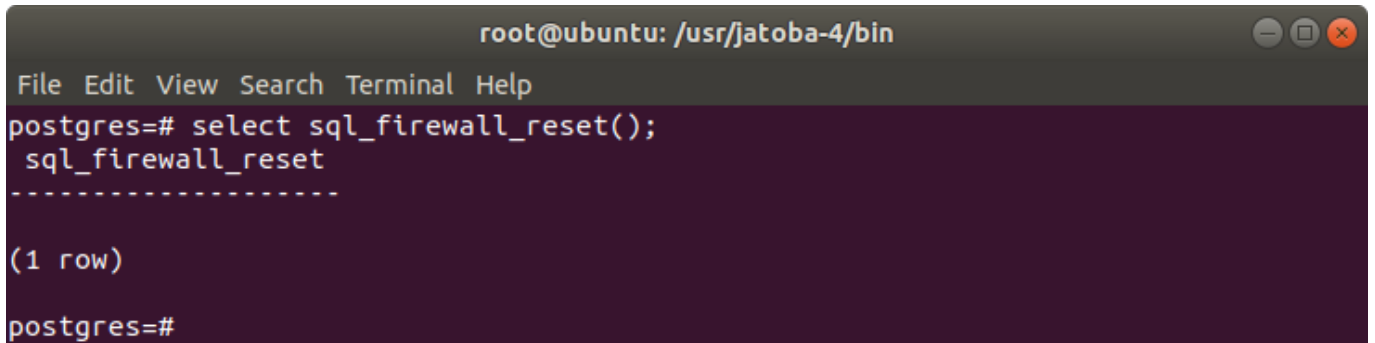
В итоге очистятся сформированные правила и перезапишется файл «rule.txt»

Проверить выполнение очистки правил компонента возможно вызовом функции «[sql_firewall.sql_firewall_statements](#)».

4.2.4. Очистка предупреждений и ошибок (sql_firewall_stat_reset)

Функциональная возможность очистки правил компонента «sql_firewall» доступна в режиме «[disabled](#)» от имени и с правами привилегированного пользователя с атрибутом «Superuser» и выполняется SQL-командой:

```
SELECT sql_firewall_reset();
```



The screenshot shows a terminal window titled 'root@ubuntu: /usr/jatoba-4/bin'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command prompt is 'postgres=#'. The user enters the command 'select sql_firewall_reset();'. The terminal displays the output 'sql_firewall_reset' followed by a dashed line and '(1 row)'. The prompt returns to 'postgres=#'.

Рисунок 4.6 – SQL-команда очистки предупреждений и ошибок в ОС GNU/Linux

По результатам выполнения команды выполнится очистка счетчиков предупреждений и ошибок.

Проверить выполнение очистки правил компонента возможно вызовом функции «[sql_firewall.sql_firewall_stat](#)», через SQL-команду:

```
SELECT * from sql_firewall.sql_firewall_stat;
```

5. УДАЛЕНИЕ

Для полного удаления компонента необходимо выполнить следующие действия:

- 1) удалить расширение SQL-командой:

```
DROP EXTENSION sql_firewall
```

- 2) удалить или закомментировать в конфигурационном файле postgresql.conf загрузку компонента:

```
#shared_preload_libraries = 'sql_firewall'  
#sql_firewall.firewall = 'disabled'  
#sql_firewall.max = 5000
```

- 3) перезапустить службу СУБД.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

SQL	–	Structured Query Language – язык структурированных запросов
БД	–	База данных
ОС	–	Операционная система
СУБД	–	Система управления базами данных

[illegible]

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм.: _____
--------------------	--------------------------	---------------------------