Руководство пользователя: Управление парольными политиками

Центр разработки PostgreSQL

Exported on 05/29/2020

Table of Contents

Парольные политики - это набор правил, регулирующих создание и использование паролей.

PostgreSQL Sber Edition управляет парольными политики:

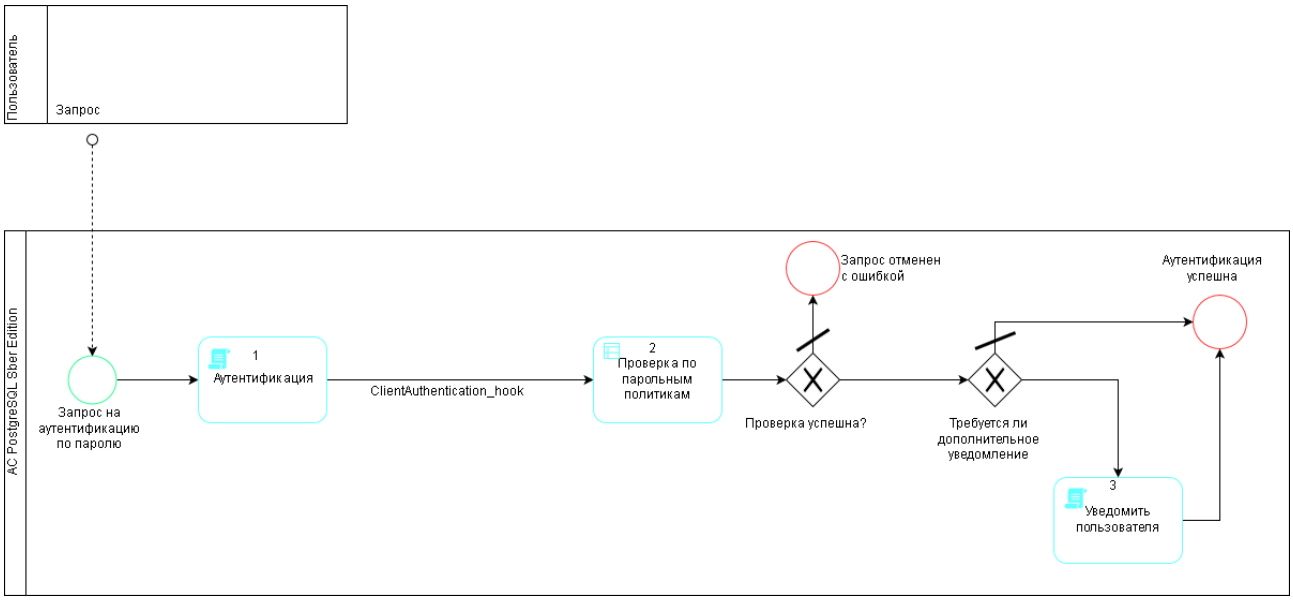
- группирует пользователей по ролям.
- назначает пользователям и ролям соответствующие политики.

Позволяет настраивать правила:

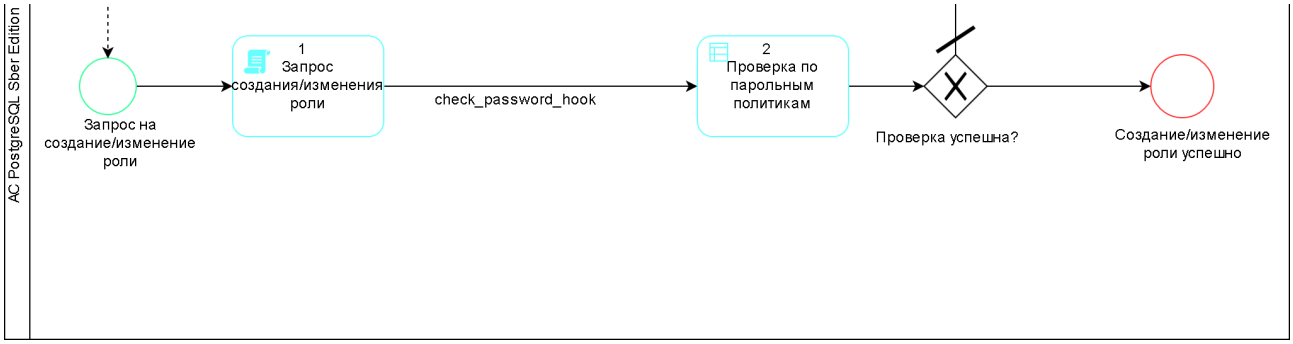
- время жизни пароля (после истечения которого пароль устаревает);
- минимальная длина пароля;
- минимальное количество букв/цифр в пароле;
- максимальное количество повторяющихся символов;
- использование строчных/прописных букв в пароле;
- период (или кол-во использований) времени после устаревания пароля, в течении которого пароль все еще может быть использован для аутентификации;
- запрет смены пароля на ранее использовавшийся (количество ранее используемых паролей должно быть настраиваемое);
- проверка пароля на списках часто используемых паролей.
- pl/sql функция для проверки пароля

Схема процесса

Аутентификация по паролю



Создание/изменение роли



Создание/изменение парольных политик

Создание/изменение парольных политик, привязанных к определенной роли. Не предполагается изменений никаких процессов: парольные политики привязываются обычными SQL запросами.

Разблокировка роли

Разблокировка входа по паролю для роли, если он был заблокирован в результате превышения максимального количества неудачных попыток аутентификации по паролю или долгой неактивности.Не предполагается изменений никаких процессов: разблокировка производится вызовом SQL процедуры.

1 API

1.1 Поиск парольной политики для пользователя (роли)

Вывод политики, приписанной этой роли

recognize_password_policy(name role_name)

recognize_password_policy_by_role_id(oid roleid)

Ответ: кортеж с описание политики.

1.2 Поиск парольной политики для пользователя (роли)

Вывод эффективной политику, получившуюся в сценарии (выбор политики для пользователя) с указанием какая политика из какой роли (или из postgresql.conf) действует.

recognize_password_policy_detailed(name role_name)

recognize_password_policy_detailed_by_role_id(oid roleid)

Ответ: кортеж с политикой с полями указывающими роль (или строку config) для каждой политики.

1.3 Приписать парольную политику роли

Приписать роли значения парольных политик.

set_role_policies(name role_name, policy p1, policy p2, ...);

set_role_policies_by_id(oid roleid, policy p1, policy p2, ...);

Пользовательский тип policy может быть получен из вспомогательных функций:

Имя политики	Функция и ее синонимы
reuse_time	reuse_time()
in_history	in_history()
max_age	max_age()
min_age	min_age()
grace_login_limit	grace_login_limit(), grace_li()
grace_login_time_limit	grace_login_time_limit(), grace_lti()
expire_warning	expire_warning()

Имя политики	Функция и ее синонимы
lockout	lockout()
lockout_duration	lockout_duration(), lduration()
max_failure	max_failure()
failure_count_interval	*failure_count_interval*(), fc_interval()
check_syntax	*check_syntax*()
min_length	*min_length*()
illegal_values	*illegal_values*()
alpha_numeric	*alpha_numeric*()
min_alpha_chars	*min_alpha_chars*()
min_special_chars	*min_special_chars*()
min_uppercase	*min_uppercase*()
min_lowercase	*min_lowercase*()
max_rpt_chars	*max_rpt_chars*()
policy_enable	*policy_enable*()
track_login	*track_login*()
max_inactivity	*max_inactivity*()
use_password_strength_estimator	*use_password_strength_estimator*(), use_zxcvbn()
password_strength_estimator_score	*password_strength_estimator_score*(), zxcvbn_score()
custom_function	*custom_function*()

Пример вызова: SELECT * FROM set_role_policies('test_user', max_age("2 days"), in_history(4));

Функция вернет такой же результат, как функция *recognize_password_policy* отобразив изменения.

Изменить параметр policy_enable также можно функцией активации/дезактивации политики:

enable_policy(name role_name)

enable_policy_by_id(oid roleid)

disable_policy(name role_name)

disable_policy_by_id(oid roleid)

1.4 Вывод всех политик

Поиск всех ролей, к которым прикреплены политика паролей.

select_all_password_policies()

1.5 Разблокировка пользователя

Разблокировать пользователя заблокированного в результате действия парольных политик.

unblock_role(name role_name)

unblock_role_by_id(oid roleid)

Результат выполнения: устанавливает: unblock_expiry_time = current_time

1.6 Пользовательская функция проверки пароля

Пользовательская функция проверки пароля должна вызываться из PSQL (SELECT func(params)).

Принимаемые параметры:

Параметр	Тип	Описание
username	name	имя пользователя
password	text	пароль
password_type	integer	тип пароля: 0 - незашифрованный 1- зашифрованный (md5) 2 - зашифрованный (scram sha 256) Зашифрованный пароль приходит, если АС PostgreSQL Sber Edition подключен к сторонней системе аутентификации

Возвращаемый параметр типа bool: true - проверка пройдена, false - не пройдена.

1.7 Ранее использованные пароли

Таблица *pp_history*

Название колонки	Тип	Примечание
roloid	regrole	идентификатор роли
password	text	хэш пароля
last_success_time	timestamptz	время последнего успешного входа в систему с использованием этого пароля
create_time	timestamptz	время создания пароля
archive_time	timestamptz	время перемещения пароля в таблицу

1.8 Информация о текущем пароле

Таблица *pp_password*

Название колонки	Тип	Описание
roloid	*regrole*	идентификатор роли
password	*text*	хэш пароля (нам нужна копия, на случай, если пользователю будет установлен пустой пароль NULL - в этом нету возможности передать управление модулю)
fail_counter	*integer*	количество попыток входа (подряд) с использованием неверного пароля
last_fail_time	*timestamptz*	время последней неудачной попытки входа с неверным паролем
grace_success_counter	*integer*	счетчик удачных аутентификаций после истечения max_age
last_success_time	*timestamptz*	время последнего удачной авторизации с паролем
create_time	*timestamptz*	время создания текущего пароля

unblock_expiry_time	*timestamptz*	время разблокировки, если роль была заблокирована в связи с неактивностью

1.9 Парольная политика

Таблица *pp_policy*

Название колонки	**Тип**	**Описание**	Недоступные действия, если политика неопределена (NULL), если deny_default = true (см. параметры postgresql.conf)
roloid	*regrole*	идентификатор роли	-
reuse_time	*interval*	время, в течении которого старый пароль сохраняется и попытка сменить пароль на совпадающий со старым заканчивается ошибкой	изменение пароля (если при этом неопределена и in_history)
in_history	*integer*	максимальное количество сохраненных старых паролей. При достижении максимума добавление еще одного старого пароля приводит к удалению наиболее старого (по create_time) из них	изменение пароля (если при этом неопределена и reuse_time)
max_age	*interval*	время жизни пароля, после которого пароль считается истекшим	аутентификация

min_age	*interval*	время, которое должно пройти между двумя изменениями пароля	изменение пароля
grace_login_limit	*integer*	максимальное количество аутентификации, доступных роли после истечения времени жизни пароля	аутентификация (если пароль просрочен)
grace_login_time_limit	*interval*	время после окончания действия пароля, в течении которого он продолжает работать	аутентификация (если пароль просрочен)
expire_warning	*interval*	время до окончания действия пароля, в течении которого пользователю будет отображаться предупреждение	-
lockout	*boolean*	признак включенного правила блока аккаунта в результате достижения максимума попыток входа с неверным паролем	аутентификация
lockout_duration	*interval*	время, на которое блокируется аккаунт в результате достижения максимума попыток входа с неверным паролем	аутентификация (если lockout = true)
max_failure	*integer*	максимальное количество подряд введных неверных паролей, при достижении которого вызывается блокировка пароля	аутентификация (если lockout = true)

failure_count_interval	*interval*	время, после которого обнуляется кол-во неверных вводов пароля	аутентификация (если lockout = true)
check_syntax	*boolean*	признак включенных правил синтаксической проверки пароля (если она возможно)	изменение пароля
min_length	*integer*	минимальная длинна пароля	изменение пароля (если check_syntax = true)
illegal_values	*boolean*	признак включенного правила проверке пароля по списку часто используемых (crack.h)	изменение пароля
alpha_numeric	*integer*	минимальное количество цифр в пароле	изменение пароля (если check_syntax = true)
min_alpha_chars	*integer*	минимальное количество букв в пароле	изменение пароля (если check_syntax = true)
min_special_chars	*integer*	минимальное количество символов, не являющихся буквой или цифрой, в пароле	изменение пароля (если check_syntax = true)

min_uppercase	*integer*	минимальное количество прописных букв	изменение пароля (если check_syntax = true)
min_lowercase	*integer*	минимальное количество строчных букв	изменение пароля (если check_syntax = true)
max_rpt_chars	*integer*	максимальное количество повторяющихся символов	изменение пароля (если check_syntax = true)
policy_enable	*boolean*	признак включенной политики	-
track_login	*boolean*	запоминать ли время последней аутентификации	аутентификация
max_inactivity	*interval*	время после последней аутентификации после которого роль будет заблокирована	аутентификация (если track_login = true и превышен интервал неактивности)
use_password_strength_estimator	*boolean*	использовать библиотеку, которая будет проверять сложность пароля, вместо встроенной синтаксической проверки?	изменение пароля

password_strength_estimator_score	*integer*	минимальная оценка сложности пароля, допустимая в системе (0-4)	изменение пароля (если use_password_strength_estimator = true)
custom_function	*string*	название пользовательской функции проверки пароля	изменение пароля

Предполагается, что сами политики не имеют своего имени - для них будет отображаться имя соответствующей роли, т.е. того самого профиля.

2 Параметры в postgresql.conf:

Название колонки	Тип	Описание	Ограничения на значение
password_policy.deny_default	*boolean*	Если true, то запрещает использовать значение политик из postgresql.conf или по умолчанию, только из ролевой модели (pp_policy)	
password_policy.reuse_time	*string*	время, в течении которого старый пароль сохраняется и попытка сменить пароль на совпадающий со старым заканчивается ошибкой	>= 0
password_policy.in_history	*integer*	максимальное количество сохраненных старых паролей. При достижении максимума добавление еще одного старого пароля приводит к удалению наиболее старого (по create_time) из них	0 - 1000
password_policy.max_age	*string*	время жизни пароля, после которого пароль считается истекшим	>= 0
password_policy.min_age	*string*	время, которое должно пройти между двумя изменениями пароля	>= 0
password_policy.grace_login_limit	*integer*	максимальное количество аутентификации, доступных роли после истечения времени жизни пароля	0 - 1000
password_policy.grace_login_time_limit	*string*	время после окончания действия пароля, в течении которого он продолжает работать	>= 0
password_policy.expire_warning	*string*	время до окончания действия пароля, в течении которого пользователю будет отображаться предупреждение	>= 0

password_policy.lockout	*boolean*	признак включенного правила блока аккаунта в результате достижения максимума попыток входа с неверным паролем	
password_policy.lockout_duration	*string*	время, на которое блокируется аккаунт в результате достижения максимума попыток входа с неверным паролем	>= 0
password_policy.max_failure	*integer*	максимальное количество подряд введных неверных паролей, при достижении которого вызывается блокировка пароля	1 - 1000
password_policy.failure_count_interval	*string*	время, после которого обнуляется кол-во неверных вводов пароля	>= 0
password_policy.check_syntax	*boolean*	признак включенных правил синтаксической проверки пароля (если она возможно)	
password_policy.min_length	*integer*	минимальная длинна пароля	0 - 1000
password_policy.illegal_values	*boolean*	признак включенного правила проверке пароля по списку часто используемых (crack.h)	
password_policy.alpha_numeric	*integer*	минимальное количество цифр в пароле	0 - 1000
password_policy.min_alpha_chars	*integer*	минимальное количество букв в пароле	0 - 1000

password_policy.min_special_chars	*integer*	минимальное количество символов, не являющихся буквой или цифрой, в пароле	0 - 1000
password_policy.min_uppercase	*integer*	минимальное количество прописных букв	0 - 1000
password_policy.min_lowercase	*integer*	минимальное количество строчных букв	0 - 1000
password_policy.max_rpt_chars	*integer*	максимальное количество повторяющихся символов	0 - 1000
password_policy.policy_enable	*boolean*	признак включенной политики	
password_policy.track_login	*boolean*	запоминать ли время последней аутентификации	
password_policy.max_inactivity	*string*	время после последней аутентификации после которого роль будет заблокирована	>= 0
password_policy.use_password_strength_estimator	*boolean*	использовать библиотеку, которая будет проверять сложность пароля, вместо встроенной синтаксической проверки?	
password_policy.password_strength_estimator_score	*integer*	минимальная оценка сложности пароля, допустимая в системе	0 - 4

password_policy.custom_function	*string*	название пользовательской функции проверки пароля	

3 Значения по умолчанию (если в postgresql.conf не указаны)

Политика	Значение
deny_default	off
reuse_time	0
in_history	0
max_age	120 days
min_age	0
grace_login_limit	5
grace_login_time_limit	0 (если grace_login_limit - не ноль, то это значение не будет учитываться)
expire_warning	7 days
lockout	on
lockout_duration	24 hours
max_failure	10
failure_count_interval	0 (не обнуляется)
check_syntax	on
min_length	5
illegal_values	off
alpha_numeric	1
min_alpha_chars	0
min_special_chars	0
min_uppercase	0
min_lowercase	0
max_rpt_chars	0 (не учитывается)

policy_enable	on
track_login	off
max_inactivity	0 (вечно)
use_password_strength_estimator	off
password_strength_estimator_score	3

4 Специальные значения параметров

Политика	Значения	Специальное действие
in_history	0	Проверка на совпадение пароля с ранее использованным не проводится (при условии reuse_time = 0)
max_age	0	Проверка максимальной времени жизни пароля не производится
min_age	0	Проверка минимального времени жизни пароля не производится
grace_login_time_limit	0	Дополнительное время, в течении которого аутентификация доступна после истечения времени жизни пароля, не предусмотрено (при условии)
expire_warning	0	Вывод предупреждений о скором истечении времени жизни пароля не производится
lockout_duration	0	Блокировка пользователя по кол-ву неудачных аутентификаций бессрочна
failure_count_interval	0	Счетчик неудачных аутентификаций не устаревает
min_length	0	Минимальная длина пароля не проверяется при синтаксической проверке
alpha_numeric	0	Минимальное кол-во цифр не проверяется при синтаксической проверке
min_alpha_chars	0	Минимальное кол-во букв не проверяется при синтаксической проверке
min_special_chars	0	Минимальное кол-во специальных символов не проверяется при синтаксической проверке
min_uppercase	0	Минимальное кол-во заглавных букв не проверяется при синтаксической проверке
min_lowercase	0	Минимальное кол-во строчных букв не проверяется при синтаксической проверке

max_rpt_chars	0	Максимальное кол-во повторяющихся символов не проверяется при синтаксической проверке
max_inactivity	0	Время неактивности не учитывается для блокировки

4.1 Сценарии внедряемые модулем в процессы создания/ изменения пользователя и аутентификации

Все запросы к базе выполняются через SPI интерфейс (т.е. не через внутренний API AC PostgreSQL Sber Edition), т.к. вероятность изменения SQL интерфейса меньше

- Аутентификация пользователя

  N/A

- Изменение пароля

  N/A

- Проверка сложности пароля

  Для проверки сложности пароля требуется включить (use_password_strength_estimator = true). Результат - оценка (0-4) сложности подборки пароля (в скобках указывается оценочное количество итераций подбора пароля):

Сложность	Описание
0	слишком простой пароль (< 10^3)
1	скорее подбираемый (< 10^6)
2	возможно подбираемый (< 10^8)
3	надежный пароль (< 10^10)
4	очень надежный пароль (>= 10^10)

  Возможно одновременное использование use_password_strength_estimator = true и use_cracklib = true. Если пароль зашифрован в виде хэша - часть проверок (синтаксические, списочные) невозможно провести.

- Проверка сложности пароля при использовании \password
- Выбор политики для пользователя (помним, что в AC PostgreSQL Sber Edition пользователь == роль)

  Алгоритм: выставленное значение политики (если не nil) всегда перебивает значение из родительских ролей. Если из двух разных родительских ролей приходят разные значения политики - применяется более строгая. Если после этого какое-то из значений - nil, смотрим postgresql.conf. Если и там нету этих политик - берем значения по умолчанию.

1. Ищем роль и переходим с ней к шагу 2;

2. Если роли присвоена своя эксклюзивная политика – берем ее и переходим к шагу 4, если нет для каждой роли в которой есть роль является участником переходим к шагу 1. Если нет - к шагу 3;
3. Смотрим роли, в которых роль является участником. Для каждой роли переходим к шагу 2;
4. Тут мы собрали n>0 ролей с потенциально разными политиками. Теперь задача их смержить: для каждого поля политики ищем самое строгое ограничение и его используем. Если для определенного поля нету политик (nil) – используем значение по умолчанию из postgresql.conf.

При этом в родительских ролях будут искаться и мержиться только те политики, которые не были определены (NULL) на текущем уровне. При этом действуют правила взаимозависимости ролей:

Политика_1	**Условие, при котором Политика_2 не работает**	**Политика_2**
policy_enabled	policy_enabled == false	все остальные политики
reuse_time	reuse_time > 0	in_history
max_age	max_age == NULL || max_age == 0	grace_login_limit, grace_login_time_limit, expire_warning
lockout	lockout == NULL || lockout == false	lockout_duration, max_failure, failure_count_interval
check_syntax	check_syntax == NULL || check_syntax == false	min_length, alpha_numeric, min_alpha_chars, min_special_chars, min_uppercase, min_lowercase, max_rpt_chars
track_login	track_login == NULL || track_login == false	max_inactivity
use_password_strength_estimator	use_password_strength_estimator == NULL || use_password_strength_estimator == false	password_strength_estimator_score

Это означает, что вы можете проставить любые значения параметров политики, но если из-за взаимозависимости они не работают - при мерже (как и в результате вычисления эффективной политики пользователя) они посчитаются как NULL.

- Разблокировка пользователя

Создаются процедуры *pp_unblock_role(text role_name)* и *pp_unblock_role_by_id(oid roleid)* для разблокировки входа по паролю для роли, если он был заблокирован в результате превышения максимального количества неудачных попыток аутентификации по паролю или долгой неактивности.

1. Ищем роль и переходим с ней к шагу 2;
2. fail_counter = 0, unblock_expiry_time = current_time.

- Подключение модуля

Подключение модуля возможно исключительно через параметр shared_preload_libraries конфига postgresql.conf. Связано это с использованием кастомных шаренных кэшей - их возможно создать

только изменяя процесс инициализации relation cache фазы 2 и инициализации кэшей catcache фазы 2.

Изменение параметров конфигурации возможно стандартным путем - pg_ctl reload.

4.2 Сообщения в лог

Сообщение	Расшифровка	Решение
User blocked: too many login fails	Пользователь заблокирован из-за превышения счетчика неудачных аутентификаций	Пользователь будет разблокирован когда пройдет lockout_duration с момента последней неудачной аутентификации. Пользователь может быть разблокирован с помощью команд unblock_role и unblock_role_by_id
Password was expired.	Пользователь заблокирован из-за просроченного пароля.	Сменить пароль пользователю
Role blocked cause long inactivity	Пользователь заблокирован из-за долгой неактивности	Пользователь может быть разблокирован с помощью команд unblock_role и unblock_role_by_id
Password will expire in <интервал>	Предупреждение об оставшемся времени до обязательной смены пароля	
Password was expired. <Число> grace logins left	Время жизни пароля превышено. Осталось <число> входов, после которых пользователь будет заблокирован	
Password was expired. Grace period ends in <интервал>	Время жизни пароля превышено. Осталось <интервал>, после истечения которого пользователь будет заблокирован	

5 Руководство для сотрудника сопровождения

Реализованная функциональность не содержит специальных инструкций для сотрудника сопровождения.

6 Руководство для администратора безопасности

Реализованная функциональность не содержит специальных инструкций для администратора безопасности.

7 Руководство для разработчика прикладных сервисов

Реализованная функциональность не содержит специальных инструкций для разработчика прикладных сервисов.