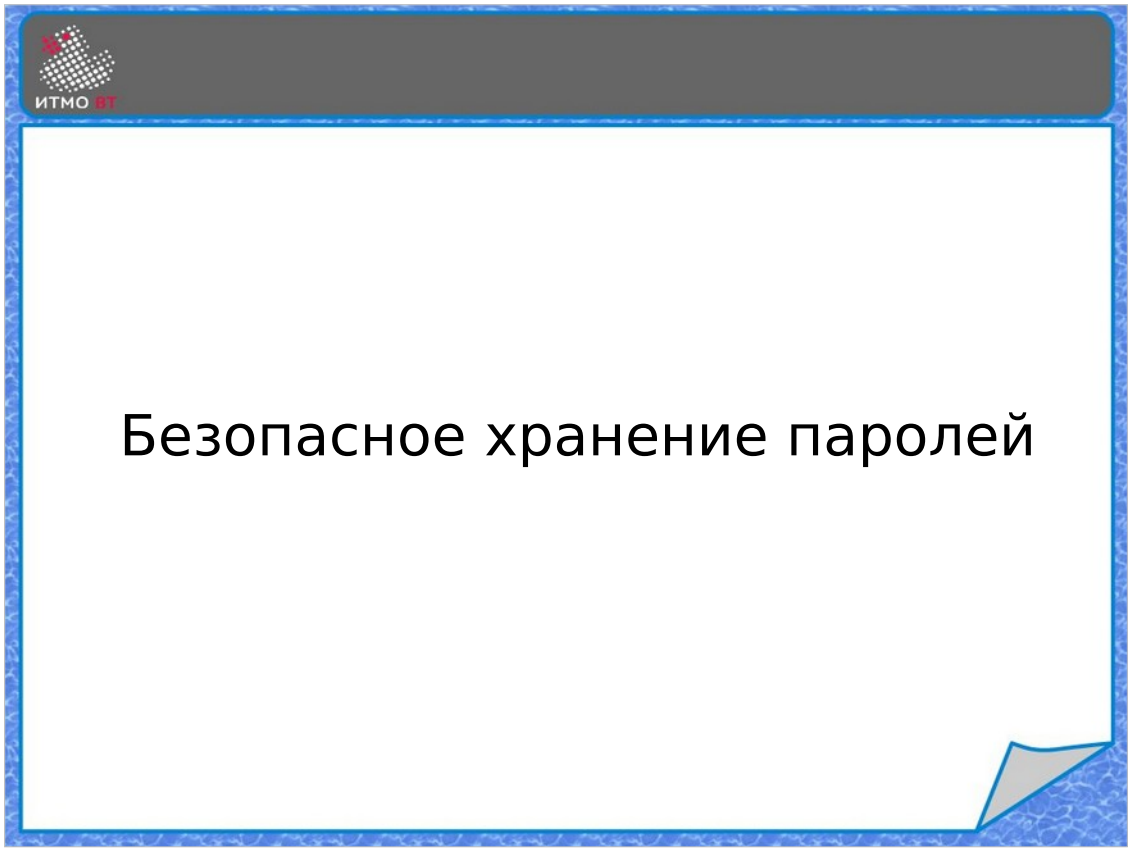
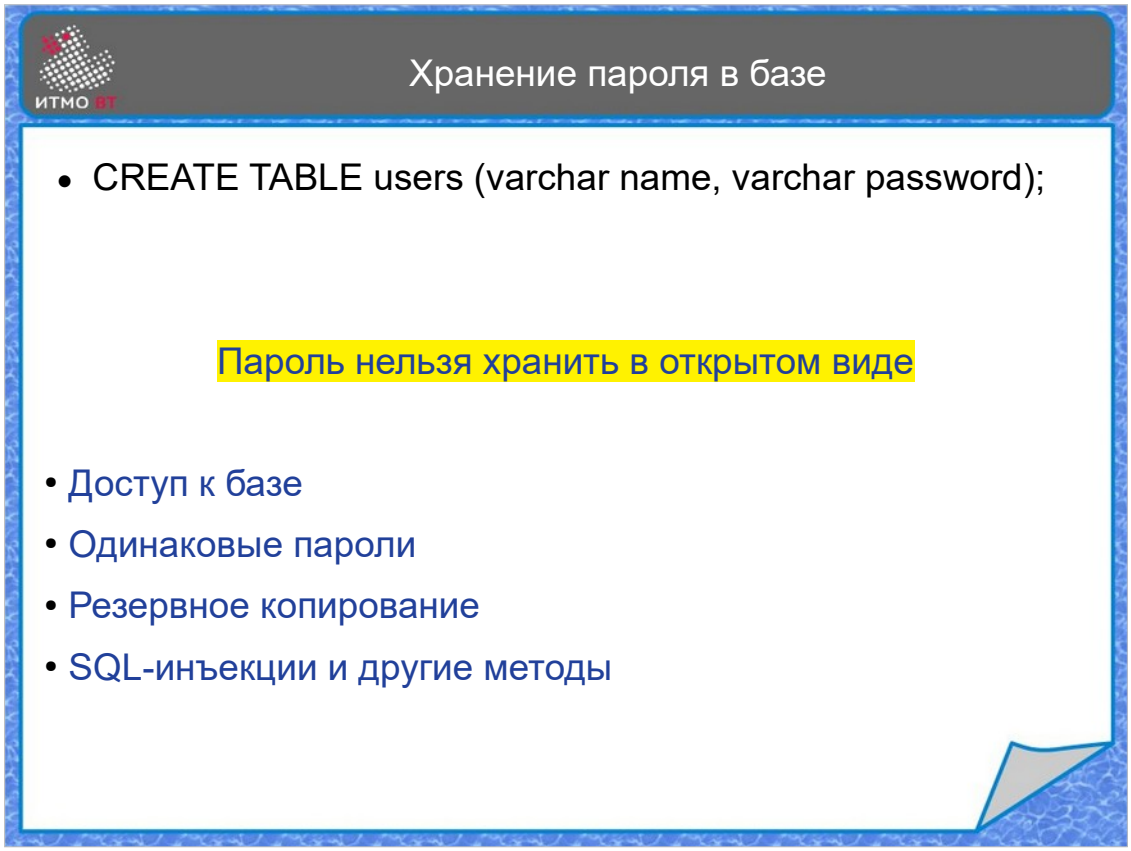
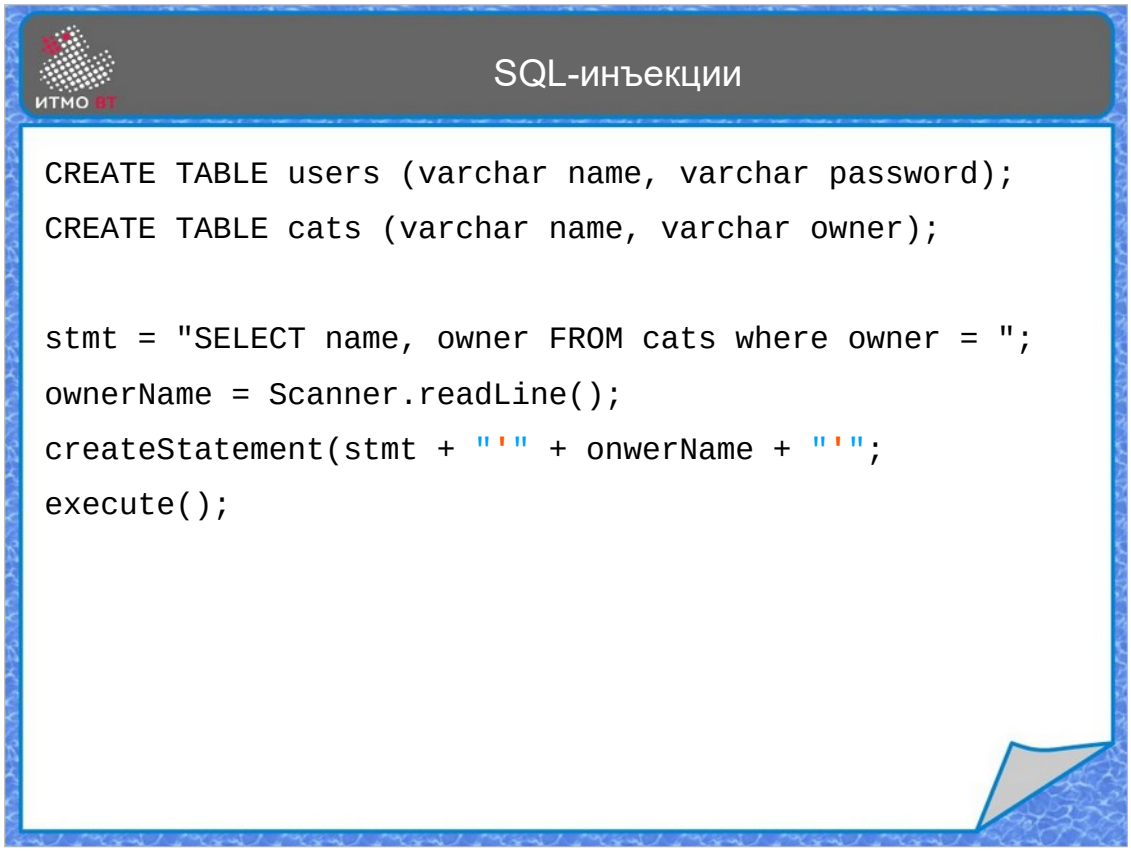

ИТМО ВТ
Безопасное хранение паролей

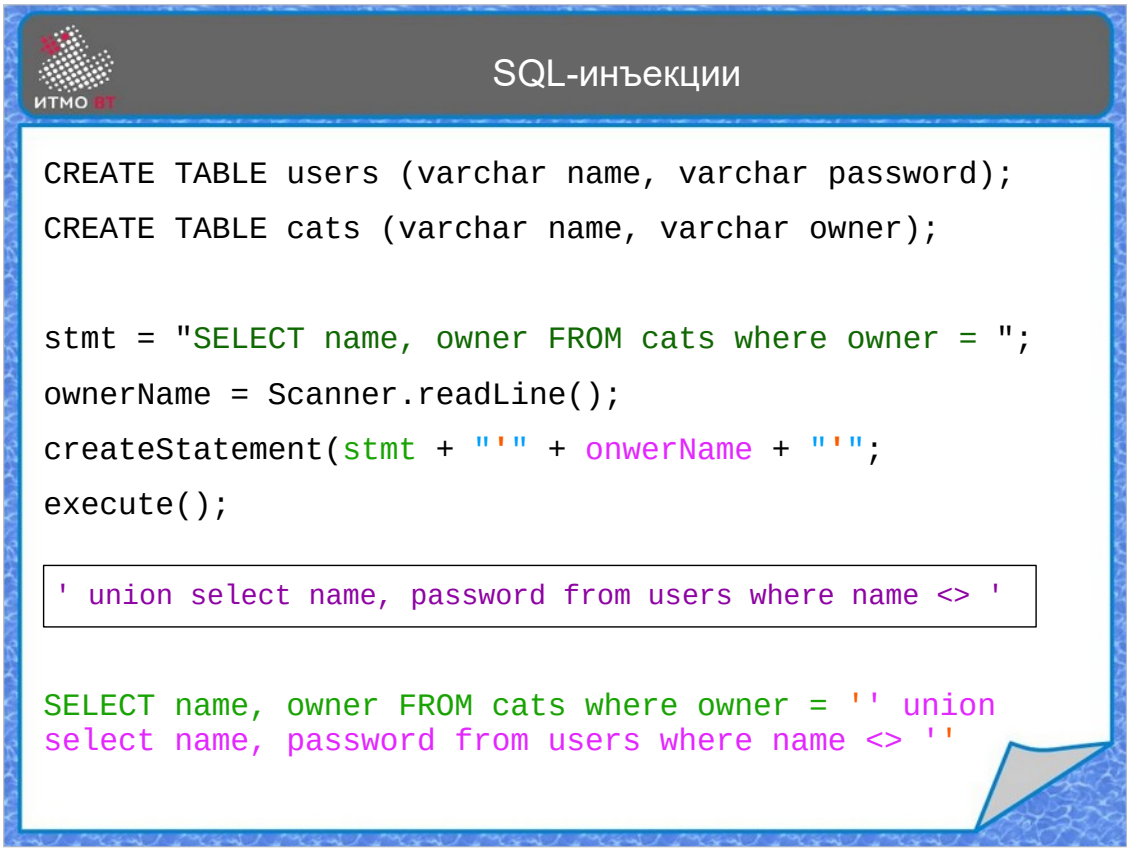


Пароли для авторизации пользователей часто хранятся в базе данных. Для обеспечения безопасности пароли хранить в открытом виде не рекомендуется. Во-первых потому, что к базе данных могут иметь доступ системные администраторы. Во-вторых, пользователи часто применяют одинаковые пароли для разных сайтов и приложений. В третьих, периодически выполняется резервное копирование базы данных, это копии потом где-то хранятся. В четвертых, есть много других способов, включая SQL-инъекции, которые позволяют получить пароли из базы, даже не обладая правами администратора.



Выглядит это примерно так. Представим, что есть таблица с пользователями, в которой хранится имя пользователя и пароль, причем в виде обычных строк. Пароль не зашифрован. И есть таблица с котиками, в которой хранятся имена котиков и имена владельцев. Вы пишете приложение, которое позволяет для каждого владельца узнать имя его котика.

Формируем запрос для получения имени котика, запрашиваем у пользователя его имя, создаем Statement, подставляем в него имя хозяина, и выполняем запрос. Вроде все хорошо.

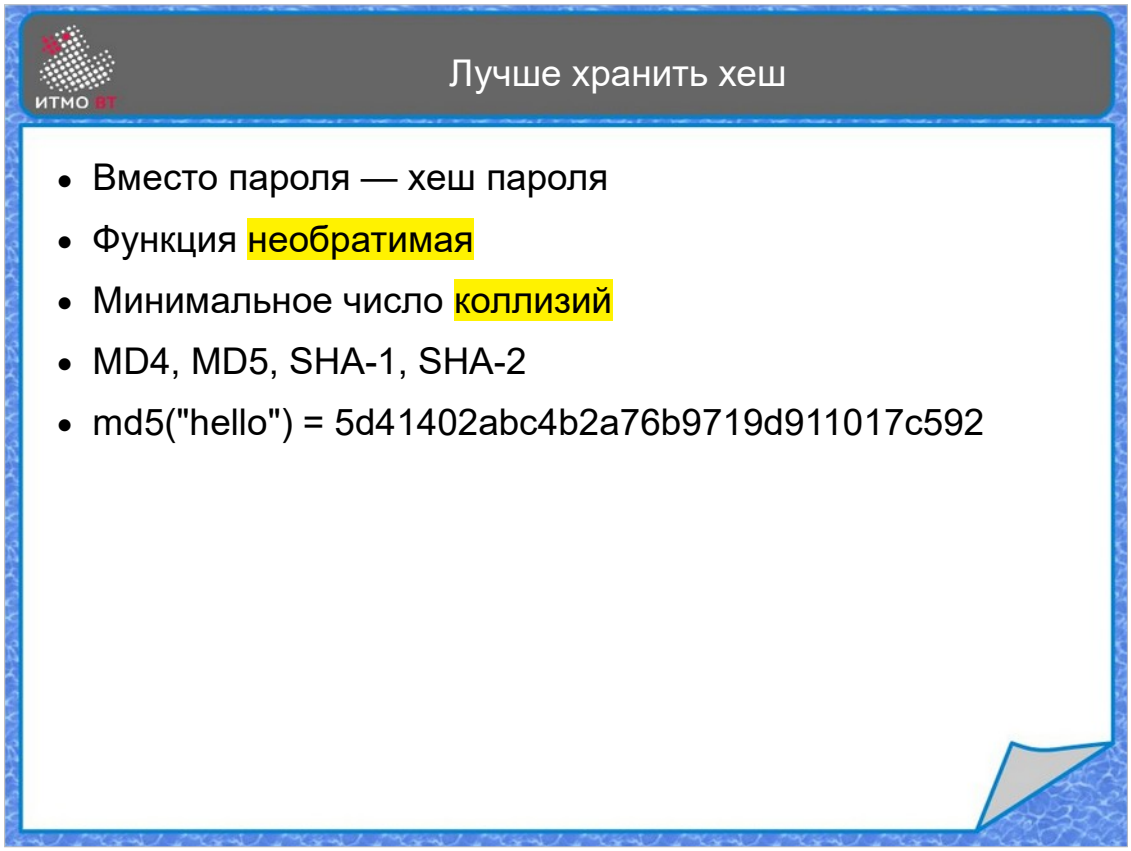


Но вот появляется злоумышленник, который в качестве имени владельца котика вводит в форму вот такую строку. И безобидный запрос становится опасным: сначала выбирается имя котика и владельца из таблицы котиков, где имя владельца равно пустой строке, и этот запрос объединяется с выборкой имени и пароля из таблицы пользователей, где имя пользователя не равно пустой строке.

В итоге вместо имени котика на экран выводится список пользователей с их открытыми паролями. Это пример SQL-инъекции.

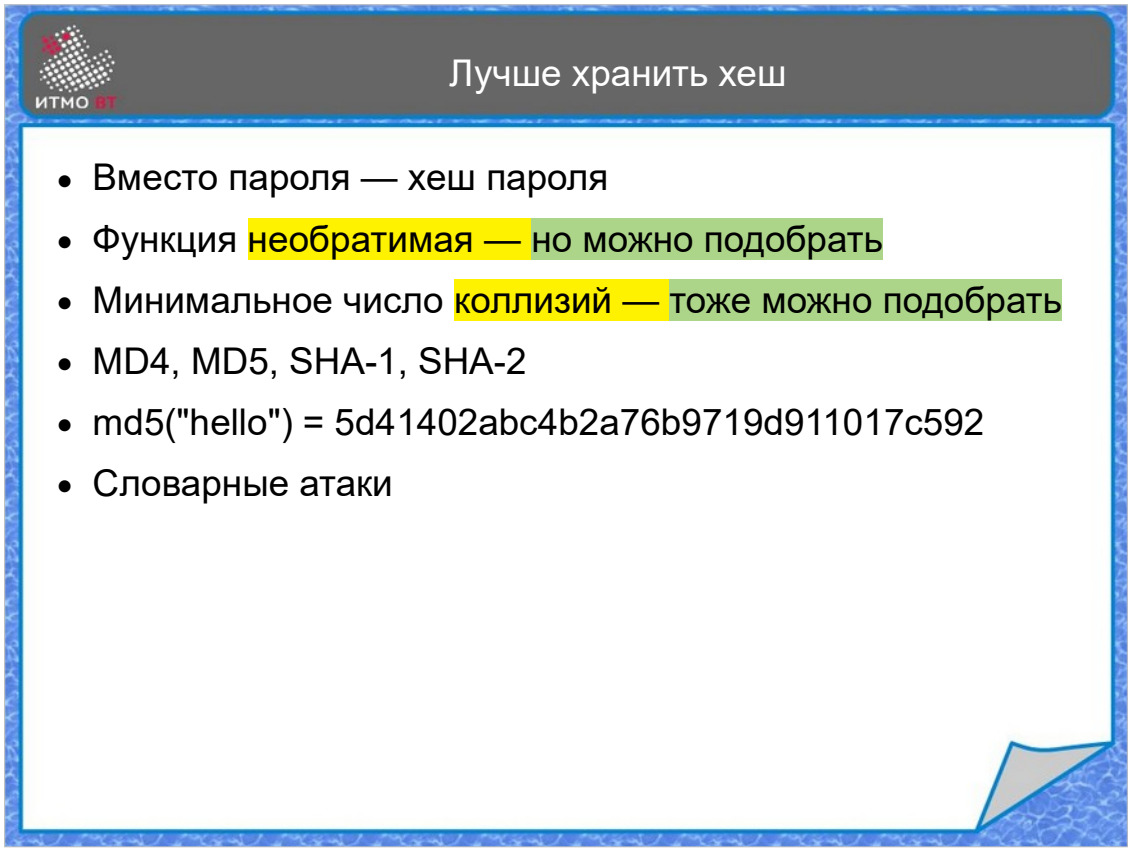
Для того, чтобы подобное не случалось, нужно валидировать и обрабатывать данные, полученные от пользователей, так как доверять им нельзя (данным и пользователям). Все специальные символы (кавычки, апострофы, слэши) должны экранироваться, тогда они станут частью строки. И еще плохо хранить пароли в открытом виде.

Как сделать так, чтобы пароль не хранился открыто, но при этом была возможность сравнить его с тем, который вводит пользователь?



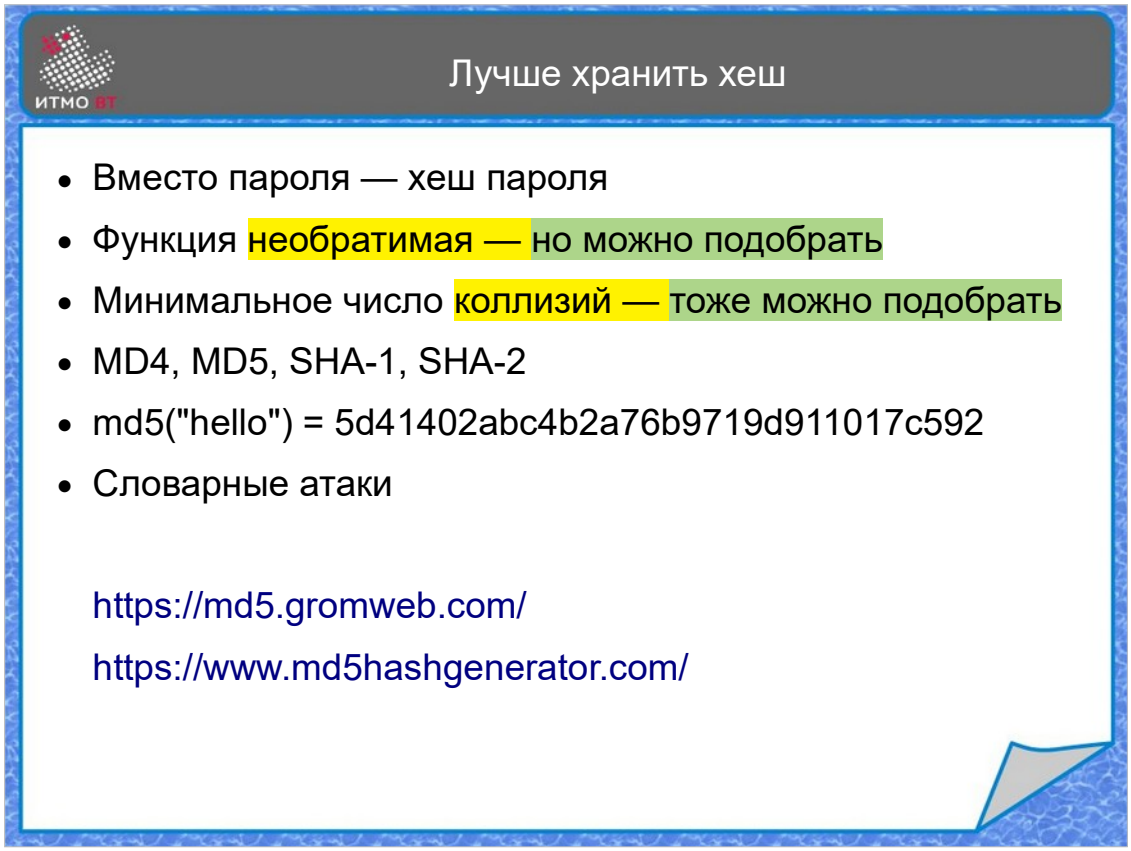
Вместо пароля можно хранить его хеш. При вводе пароля вычисляется его хеш и сравнивается с хранящимся. Если они совпали, то считаем, что был введен верный пароль. Хеш-функция должна быть необратимой, то есть не должно быть возможности расшифровать зашифрованный пароль. Также хеш-функция должна обеспечивать минимальное число коллизий.

Есть разные алгоритмы хеширования: MD4, MD5, SHA-1, SHA-2 и т. д. На слайде приведен пример получения MD5-хеша от строки «hello».



Хотя хеш-функции необратимы, но пароль можно подобрать. При наличии коллизий можно подобрать другой пароль, имеющий тот же самый хеш, тогда даже с неверным паролем можно войти в аккаунт.

Для подбора обычно используются словарные атаки, при которых составляется список наиболее распространенных паролей и слов из разных языков, и соответствующих им значений хеш-функций. Тогда по известной хеш-функции можно из словаря получить исходное значение пароля.



Посмотреть, как это работает, можно в интернете. Например, на сайте md5hashgenerator.com можно сгенерировать md5-хеш любой строки. А затем проверить этот хеш на сайте md5.gromweb.com. Если исходная строка представляет собой слово или простую последовательность символов, то с большой вероятностью его хеш будет найден в базе. Алгоритма восстановления пароля по хешу нет, но простой поиск помогает.

Более того, найти пароль по известному md5-хешу можно даже в Google. Попробуйте ввести в строку поиска «5d41402abc4b2a76b9719d911017c592».

Лучше хранить хеш с солью

- Вместо пароля — хеш пароля + соль
- Соль — некая случайная последовательность

md5("hello" + "$Ins50D") = fcac10c41f9f67090e450161db4bca5a

md5("hello" + "HOxc3@") = a893f9063e4314635e05f8c46b547f7a

Как можно усилить защиту? Можно пытаться заставить пользователей выбирать сложные пароли. Но их сложно запоминать. Можно добавить к простому паролю, типа (12345 или hello) «соль» - некую случайную последовательность, которая тоже будет храниться в базе. Хеш тогда будет генерироваться из склеенных пароля и соли. Тогда в базе данных хранится соль и хеш комбинации пароля с солью. Когда пользователь вводит свой пароль, из базы берется соль, вычисляется хеш пароля с солью, и сравнивается с тем хешем, что хранится в базе. Такой хеш скорее всего ни в Google, ни в словарях не найдется.

Лучше хранить хеш с солью

- Вместо пароля — хеш пароля + соль
- Соль — некая случайная последовательность

md5("hello" + "$Ins50D") = fcac10c41f9f67090e450161db4bca5a
md5("hello" + "HOxc3@") = a893f9063e4314635e05f8c46b547f7a

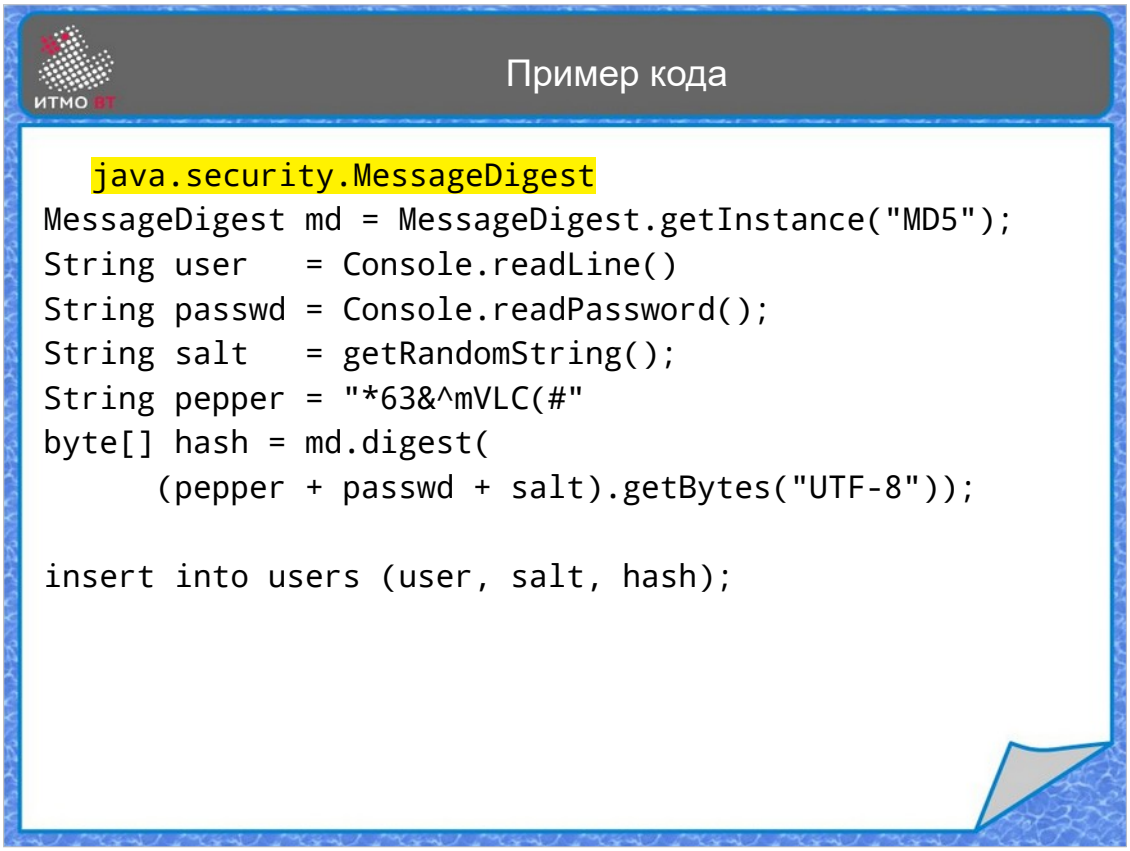
- Храним пароль + соль + алгоритм + сложность
- Все равно — "hello", "12345" — словарные атаки

Для каждого пользователя можно использовать свою соль, чтобы для одинаковых паролей получались разные хеши. Но и такой способ подвержен атакам. Так как соль в базе хранится практически в открытом виде, то хакер берет просто словарь возможных паролей, добавляет соль, и сравнивает с хешем. Это уже намного дольше, чем простой поиск в словаре. Можно использовать более сложные алгоритмы, и применять их не один раз, а несколько раз подряд, это количество можно также хранить в базе, например, сложность 1000 — это повторить хеширование 1000 раз. Подобрать все равно можно, но сделать это становится еще сложнее.

Добавим перец

- В базе храним хеш пароля + соль
- Перец храним в приложении (отдельно)
- md5("kFz<Q%ps" + "hello" + "$Ins50D")

Можно еще усилить защиту дополнительно добавив «перец». Перец - еще одна последовательность символов, которая хранится уже в коде приложения. В базе перец не хранится, чтобы узнать и соль и перец нужно иметь доступ и к базе, и к коду приложения. Соответственно, алгоритм может быть таким: приложение, получив пароль, получает из базы данных соль, добавляет соль и перец к паролю, получает хеш, и сравнивает его с хешем, который хранится в базе.



Пример кода, как это реализовать. Вызываем у класса MessageDigest метод getInstance с алгоритмом кодирования, запрашиваем имя пользователя и пароль, далее нужна соль. При регистрации пользователя соль генерируем случайно, при аутентификации пользователя запрашиваем соль из базы данных. Перец хранится в коде. Далее формируем строку из пароля, соли и перца, получаем из нее байты, и передаем массив байтов методу digest, который вернет хеш-код. Его либо сохраняем в базу при регистрации, либо проверяем на совпадение с тем, что хранится в базе.