**Практическая работа № 6.**

**Хеширование пароля на PHP**

Хранить пароль в открытом виде - неправильно. Хакер-злоумышленник может получить доступ к вашей базе данных и украсть пароли.

Поэтому обычно логин хранится в открытом виде, а пароль хешируется специальной функцией md5, которая параметром принимает пароль, а возвращает его *хеш*, по которому нельзя восстановить этот самый пароль.

Давайте, например, найдем хеш какой-нибудь строки:

<?php

echo md5('12345'); // выведет '827ccb0eea8a706c4c34a16891f84e7b'

?>

Сейчас нам необходимо переделать нашу регистрацию и нашу авторизацию. Для начала я бы советовал очистить таблицу с юзерами, так как там сейчас хранятся пароли в открытом виде, а должны хранится их хеши. Затем при тестировании регистрации таблица заполнится данными в новом формате.

Давайте теперь поправим нашу регистрацию так, чтобы при сохранении нового пользователя в базу добавлялся не пароль, а его хеш.

Описанная правка будет представлять собой что-то такое:

<?php

$login = **$\_POST**['login'];

$password = md5(**$\_POST**['password']); // преобразуем пароль в его хеш

$query = "INSERT INTO users SET login='**$login**', password='**$password**'";

?>

Внесем аналогичные правки в авторизацию:

<?php

$login = **$\_POST**['login'];

$password = md5(**$\_POST**['password']); // преобразуем пароль в его хеш

$query = "SELECT \* FROM users WHERE login='**$login**' AND password='**$password**'";

?>

№1

Внесите изменения в регистрацию с учетом хеширования, зарегистрируйте пару новых пользователей, убедитесь, что в базу данных они добавились с хешированными паролями.

№2

Внесите изменения в авторизацию с учетом хеширования, попробуйте авторизоваться под зарегистрированными ранее пользователями.

**Добавляем соль в регистрацию**

Итак, вы уже знаете, что хеширование через md5 - необратимый процесс и хакер, получивший доступ к хешу, не сможет получить по этому хешу пароль.

На самом деле это утверждение не совсем верное - в настоящее время злые хакеры составили библиотеки хешей популярных и не очень паролей и любой дурак может разгадать пароль, просто загуглив его хеш.

Речь идет о достаточно простых, популярных паролях.

Загуглите, например, хеш *827ccb0eea8a706c4c34a16891f84e7b* и сразу в поиске гугла вы увидите, что это пароль '12345'.

Хеши достаточно сложных паролей таким образом не разгадать (попробуйте).

Вы можете спросить, в чем тогда проблема - давайте все мы будем регистрироваться со сложными паролями. Есть, однако, проблема - большинство пользователей не задумываются о безопасности своих данных и могут вводить достаточно простые пароли.

Мы можем при регистрации заставлять придумывать более длинные пароли, ограничивая, к примеру, минимальное количество символов 6-ю или 8-ю, однако, все равно будут появляться пароли вида '123456' или '12345678'.

Можно, конечно, придумать более умный алгоритм проверки пароля на сложность, но есть другое решение.

Суть этого решения такая: пароли надо *посолить*. Соль - это специальная случайная строка, которая будет добавляться к паролю при регистрации и хеш уже будет вычисляться не от простого пароля типа, а от строки соль+пароль, то есть от соленого пароля.

То есть при регистрации вы будете делать что-то типа такого:

<?php

$salt = '1sJg3hfdf'; // соль - сложная случайная строка

$password = md5($salt . **$\_POST**['password']); // преобразуем пароль в соленый хеш

?>

При этом соль будет разная для каждого пользователя, ее нужно будет генерировать случайным образом в момент регистрации.

Вот готовая функция, которая сделает это:

<?php

function generateSalt()

{

$salt = '';

$saltLength = 8; // длина соли

for($i = 0; $i < $saltLength; $i++) {

$salt .= chr(mt\_rand(33, 126)); // символ из ASCII-table

}

return $salt;

}

?>

С помощью этой функции можно переписать наш код вот так:

<?php

$salt = generateSalt(); // соль

$password = md5($salt . **$\_POST**['password']); // соленый пароль

?>

Еще раз повторю, что это были изменения при регистрации - в БД сохраняем не просто хеш пароля, а хеш соленого пароля.

Это еще не все: в таблице с юзерами кроме поля login и password нужно сделать еще и поле salt, в котором мы будем хранить соль каждого пользователя.

№3

Реализуйте описанную выше регистрацию с соленым паролем.

**Добавляем соль в авторизацию**

Теперь нам необходимо поменять авторизацию. Здесь уже изменения будут более существенными.

Уже не получится проверить правильность пары логин-пароль сразу же, одним запросом. Почему: потому что, чтобы проверить пароль, надо получить его соленый хеш, а соль хранится в базе данных и уникальная для каждого логина.

Придется вначале получить запись только по логину, прочитать соль, посолить введенный пароль и сравнить с соленым паролем из базы и только, если они совпадают, - авторизовывать пользователя.

Учтите, что может такое быть, что логин вбит неправильно, в этом случае проверку пароля можно не осуществлять, а сразу вывести, что авторизация не возможна - данные не верны:

<?php

$login = **$\_POST**['login'];

$query = "SELECT \* FROM users WHERE login='**$login**'";

$result = mysqli\_query($link, $query);

$user = mysqli\_fetch\_assoc($result);

if (!empty($user)) {

// пользователь с таким логином есть, теперь надо проверять пароль...

} else {

// пользователя с таким логином нет, выведем сообщение

}

?>

Давайте добавим проверку пароля:

<?php

$login = **$\_POST**['login'];

$query = "SELECT \* FROM users WHERE login='**$login**'";

$result = mysqli\_query($link, $query);

$user = mysqli\_fetch\_assoc($result);

if (!empty($user)) {

$salt = $user['salt']; // соль из БД

$hash = $user['password']; // соленый пароль из БД

$password = md5($salt . **$\_POST**['password']); // соленый пароль от юзера

// Сравниваем соленые хеши

if ($password == $hash) {

// все ок, авторизуем...

} else {

// пароль не подошел, выведем сообщение

}

} else {

// пользователя с таким логином нет, выведем сообщение

}

?>

В целях безопасности пользователю обычно не сообщают, что именно не подошло - логин или пароль, чтобы усложнить подбор пар логин-пароль хакерами. Просто выводят сообщение о том, что пара логин-пароль неверна или что-то в таком роде.

№4

Реализуйте описанную выше авторизацию с соленым паролем. Попробуйте зарегистрируйтесь, авторизуйтесь, убедитесь, что все работает.

**Используем функцию password\_hash**

На самом деле функция md5 и соление пароля с ее помощью считается устаревшим. Мы изучали ее, чтобы вы поняли дальнейший материал, а также потому, что вы можете столкнуться с этим, работая с чужими проектами.

Существует более совершенный способ получить соленый пароль. Для этого используется функция password\_hash. Первым параметром она принимает строку, а вторым - алгоритм шифрования (о нем позднее), и возвращает хеш этой строки вместе с солью.

Попробуйте несколько раз запустите этот код:

<?php

echo password\_hash('12345', PASSWORD\_DEFAULT);

?>

Вы каждый раз будете получать разный результат и в этом результате первая часть строки будет являться солью, а вторая часть - соленым паролем.

Пусть у нас есть хеш, полученный из функции password\_hash и какой-то пароль. Чтобы проверить, это хеш этого пароля или нет, следует использовать функцию password\_verify - первым параметром она принимает пароль, а вторым - хеш, и возвращает true или false.

Давайте посмотрим на примере:

<?php

$password = '12345'; // пароль

$hash = '$2y$10$xoYFX1mFPxBSyxaRe3iIRutxkIWhxGShzEhjYUVd3qpCUKfJE1k7a'; // хеш

if (password\_verify($password, $hash)) {

// хеш от этого пароля

} else {

// хеш не от этого пароля

}

?>

Что это дает нам на практике: мы можем не создавать в базе данных отдельное поле для хранения соли, не заморачиваться с генерированием этой соли - PHP все сделает за нас!

То есть получится, что в базе данных в поле password мы будем хранить соленый пароль вместе с его солью. При этом хешированный пароль будет иметь большую длину. Поэтому в базе данных нам нужно исправить размер поля с паролем и установить ее в 60 символов.

Теперь давайте поправим код регистрации. Вот то, что есть сейчас:

<?php

function generateSalt()

{

$salt = '';

$saltLength = 8; // длина соли

for($i = 0; $i < $saltLength; $i++) {

$salt .= chr(mt\_rand(33, 126)); // символ из ASCII-table

}

return $salt;

}

$salt = generateSalt(); // соль

$password = md5($salt . **$\_POST**['password']); // преобразуем пароль в соленый хеш

?>

С помощью password\_hash мы сократим это до:

<?php

$password = password\_hash(**$\_POST**['password'], PASSWORD\_DEFAULT);

?>

Аналогичным образом подправится код авторизации:

<?php

$login = **$\_POST**['login'];

$query = "SELECT \* FROM users WHERE login='**$login**'"; // получаем юзера по логину

$result = mysqli\_query($link, $query);

$user = mysqli\_fetch\_assoc($result);

if (!empty($user)) {

$hash = $user['password']; // соленый пароль из БД

// Проверяем соответствие хеша из базы введенному паролю

if (password\_verify(**$\_POST**['password'], $hash)) {

// все ок, авторизуем...

} else {

// пароль не подошел, выведем сообщение

}

} else {

// пользователя с таким логином нет, выведем сообщение

}

?>

№5

Переделайте вашу авторизацию и регистрацию на новые изученные функции.