**Практическая работа № 19.**

**Динамическое формирование запроса в SQL в PHP**

Составлять SQL запросы вручную не очень удобно. Как правило, набравшись некоторого опыта, программист пишет функцию, которая будет формировать SQL запрос динамически. Обычно такая функция параметром принимает массив имен полей и их значений и формирует SQL запрос.

Зачастую такая функция используется, чтобы подсунуть ей массив данных из формы. Тут-то нас и поджидают уязвимости, если мы никак не обработаем эти данные.

Давайте посмотрим на примере. Пусть у нас есть вот такой запрос, который мы сформировали вручную:

<?php

$query = "UPDATE users SET login='user', password='123' WHERE id=1"

?>

Пусть данные этого запроса были получены из следующей формы:

<form action="" method="POST">

<input name="login">

<input name="password" type="password">

<input type="submit">

</form>

После отправки формы данные попадут в массив $\_POST:

<?php

var\_dump(**$\_POST**);

?>

Содержимое этого массива будет примерно следующим:

[

'login' => 'user',

'password' => '123',

];

Давайте сформируем приведенный в начале урока запрос динамически из значений формы. Для начала сформируем часть с именами полей и их значениями:

<?php

$set = '';

$arr = '';

foreach (**$\_POST** as $key=>$elem) {

$arr[] = "**$key**='**$elem**'";

}

$set = implode(', ');

var\_dump($set); // login='user', password='123'

?>

Подставим нашу переменную $set в запрос:

<?php

$query = "UPDATE users SET **$set** WHERE id=1"

?>

Проблема тут в том, что мы берем все полученные поля от юзера и тупо вставляем их в запрос. Ничто не мешает хакеру отправить форму с еще одним полем, например, role в значении 1 и легко стать админом.

№1

Исправьте проблему с помощью белого списка разрешенных полей.

№2

Приведенный код подвержен SQL инъекции. Проверьте, что инъекция проходит. Исправьте проблему.