

|  |  |
| --- | --- |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | |
| Институт кибербезопасности и цифровых технологий |
| Кафедра КБ-3 «Разработка программных решений и системное программирование»   |  | | --- | | **Отчёт по практике 3** | | **по дисциплине** «Разработка безопасного программного обеспечения» |     **Выполнила:**  Студентка 4 курса  Направление 02.03.02  Группа БФБО-01-20  Митрюхина Екатерина Андреевна  **Преподаватель**:  Поспелов П.М.  Москва, 2023 г. |

1. Необходимо разработать переборщик паролей для формы в задании Bruteforce на сайте dvwa.local (Можно использовать [официальный ресурс](https://github.com/digininja/DVWA) или виртуальную машину Web Security Dojo).

Для перебора паролей я использовала 10 логинов и 10 000 паролей. Не брала большое количество логинов, т. к. перебор длился бы долго, а так минут 10-15 занимает. В случае True, будет выводиться логин + пароль, а иначе Failed.

|  |
| --- |
|  |

Сам код в файле brute.py выглядит следующим образом:

|  |
| --- |
|  |

1. Проанализировать код и сделать кодревью, указав слабые места. Слабость уязвимого кода необходимо указать с использованием метрики CWE (база данных [cwe.mitre.org](http://cwe.mitre.org/))

<?php

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

// Get username

$user = $\_GET[ 'username' ]; (CWE-598)

// Get password

$pass = $\_GET[ 'password' ]; (CWE-598)

$pass = md5( $pass ); (CWE-327, CWE-759)

// Check the database

$query = "SELECT \* FROM `users` WHERE user = '$user' AND password = '$pass';"; (CWE-89)

$result = mysqli\_query($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>' ); (CWE-307)

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);

}

?>

Проанализировав данный код, я обнаружила следующие слабые места:

**CWE-598**: Использование метода запроса GET с чувствительными строками запроса. Описание слабости: *«Строка запроса для URL-адреса может быть сохранена в истории браузера, передана через рефереры на другие веб-сайты, сохранена в веб-журналах или иным образом записана в других источниках. Если строка запроса содержит конфиденциальную информацию, например идентификаторы сеансов, злоумышленники могут использовать эту информацию для запуска дальнейших атак».*

$user = $\_GET[‘username’];

$pass = $\_GET[‘password’]; - В данных строках обнаружена данная слабость.

Данную слабость можно исправить, на метод POST.

**CWE-327**: Использование устаревшего криптографического алгоритма MD5. Описание слабости: *«В продукте используется неработающий или рискованный криптографический алгоритм или протокол».*

$pass = md5($pass); - В данной строке обнаружена данная слабость.

Данную слабость можно исправить, используя более криптостойкий пароль – SHA256. И можно использовать соль, которая будет храниться в базе данных.

**CWE-759**: Использование одностороннего хэша без соли. Описание слабости: «*Продукт использует односторонний криптографический хеш для входных данных, который не должен быть обратимым, например пароль, но продукт также не использует соль как часть входных данных».*

$pass = md5($pass); - В данной строке обнаружена данная слабость.

Данную слабость можно исправить, добавив в генерацию паролей дополнительную соль.

**CWE-89**: Неправильная нейтрализация специальных элементов, используемых в команде SQL («SQL-инъекция»). Описание слабости: *«Без достаточного удаления или цитирования синтаксиса SQL во входных данных, управляемых пользователем, сгенерированный запрос SQL может привести к тому, что эти входные данные будут интерпретироваться как SQL, а не как обычные пользовательские данные. Это можно использовать для изменения логики запроса для обхода проверок безопасности или для вставки дополнительных операторов, которые изменяют внутреннюю базу данных, возможно, включая выполнение системных команд.*

*SQL-инъекция стала распространенной проблемой на веб-сайтах, управляемых базами данных. Уязвимость легко обнаружить и легко использовать, поэтому любой сайт или пакет продуктов даже с минимальной базой пользователей может стать объектом попытки атаки такого рода. Этот недостаток обусловлен тем фактом, что SQL не делает реального различия между плоскостями управления и данных».*

$query = "SELECT \* FROM `users` WHERE user = '$user' AND password = '$pass';"; - В данной строке обнаружена данная слабость.

Данную слабость можно исправить, добавив использование PDO или его аналогов, для устранения SQL-инъекции.

**CWE-307**: Неправильное ограничение чрезмерных попыток аутентификации. Описание слабости: *«В продукте не реализованы достаточные меры для предотвращения нескольких неудачных попыток аутентификации в течение короткого периода времени, что делает его более уязвимым для атак методом перебора».*

$result = mysqli\_query($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>' ); - В данной строке обнаружена данная слабость.

Данную слабость можно исправить:

* Реализация тайм-аута;
* Блокировка целевой учетной записи;
* Требование вычислительной задачи со стороны пользователя;
* Отключение пользователя после небольшого количества попыток.

Исправленный кодревью:

<?php

if( isset($\_POST['Login']) && isset($\_POST['username']) && isset($\_POST['username']) ) {

$user = stripslashes($\_POST['username']);

$total\_failed\_login = 3;

$lockout\_time = 5;

$account\_locked = false;

$data = $db->prepare( 'SELECT failed\_login, last\_login FROM users WHERE user = (:user) LIMIT 1;' );

$data->bindParam( ':user', $user, PDO::PARAM\_STR );

$data->execute();

$row = $data->fetch();

if (($data->rowCount() == 1) && ($row['failed\_login'] >= $total\_failed\_login)) {

$last\_login = strtotime( $row[ 'last\_login' ] );

$timeout = $last\_login + ($lockout\_time \* 60);

$timenow = time();

if( $timenow < $timeout ) {

$account\_locked = true;

}

}

$pass = md5( stripslashes( $\_POST['password'] ) );

// MD5 -> SHA256 (pass + get\_salt($user));

$data = $db->prepare( 'SELECT \* FROM users WHERE user = (:user) AND password = (:password) LIMIT 1;' );

$data->bindParam( ':user', $user, PDO::PARAM\_STR);

$data->bindParam( ':password', $pass, PDO::PARAM\_STR );

$data->execute();

$row = $data->fetch();

$html = "";

if( ( $data->rowCount() == 1 ) && ( $account\_locked == false ) ) {

$avatar = $row[ 'avatar' ];

$html .= "<p>Welcome to the password protected area {$user}</p>";

$html .= "<img src=\"{$avatar}\" />";

$failed\_login = $row[ 'failed\_login' ];

$data = $db->prepare( 'UPDATE users SET failed\_login = "0" WHERE user = (:user) LIMIT 1;' );

$data->bindParam( ':user', $user, PDO::PARAM\_STR );

$data->execute();

} else {

sleep(3);

$html .= "<pre>";

$html .= "Username and/or password incorrect.<br/><br/>";

$html .= "This account has been blocked for too many login attempts.<br/>";

$html .= "Please try again in {$lockout\_time} minutes<br/>";

$data = $db->prepare( 'UPDATE users SET failed\_login = (failed\_login + 1) WHERE user = (:user) LIMIT 1;' );

$data->bindParam( ':user', $user, PDO::PARAM\_STR );

$data->execute();

}

$data = $db->prepare( 'UPDATE users SET last\_login = now() WHERE user = (:user) LIMIT 1;' );

$data->bindParam( ':user', $user, PDO::PARAM\_STR );

$data->execute();

}

?>

1. Разработать свою систему авторизации на любом языке, исключающий возможность подбора паролей разработанным переборщиком паролей в задании 1. Возможно исправление авторизации из dvwa.local  
   Требования к системе авторизации
   * Система авторизации должна использовать запросы GET с параметрами, аналогичными из задания bruteforce dvwa

dvwa.local/vulnerabilities/brute/?username=USER&password=PASS&user\_token=TOKEN&Login=Login

Я разработала мини версию авторизации на веб-сайте с применением капчи. Вот так она выглядит:

|  |
| --- |
|  |

При правильном вводе логина и пароля из базы данных, которую мы интегрировали в код, и капчи происходит следующее (переход на /success с сообщением “Успешная авторизация”).

|  |
| --- |
|  |

При неправильном вводе данных и капчи происходит следующее (выводится сообщение о неправильном вводе данных):

|  |
| --- |
|  |

Вот код, содержащийся в файле “app.py”:

from flask import Flask, request, render\_template, session, redirect, url\_for  
from captcha.image import ImageCaptcha  
import random  
import string  
import base64  
from io import BytesIO  
  
app = Flask(\_\_name\_\_)  
app.secret\_key = 'secretkey'  
  
base = {  
 'user1': 'password1',  
 'user2': 'password2'  
}  
  
def generate\_captcha():  
 captcha\_text = ''.join(random.choices(string.ascii\_uppercase + string.digits, k=6))  
 image = ImageCaptcha()  
 data = BytesIO()  
 image.write(captcha\_text, data)  
 data.seek(0)  
 return captcha\_text, base64.b64encode(data.getvalue()).decode('utf-8')  
  
@app.route('/login', methods=['GET', 'POST'])  
def login():  
 if request.method == 'POST':  
 username = request.form['username']  
 password = request.form['password']  
 user\_captcha = request.form['captcha']  
  
 if username in base and base[username] == password and user\_captcha == session['captcha']:  
 return redirect(url\_for('success', message='Авторизация успешна'))  
 else:  
 return redirect(url\_for('login', message='Неправильный логин, пароль или капча'))  
 else:  
 captcha\_text, captcha\_image = generate\_captcha()  
 session['captcha'] = captcha\_text  
 message = request.args.get('message', '')  
 return render\_template('site.html', captcha\_text=captcha\_text, captcha\_image=captcha\_image, message=message)  
  
  
@app.route('/success')  
def success():  
 message = request.args.get('message', '')   
 return render\_template('success.html', message=message)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

В папке templates находятся 2 файла HTML. В “site.html”:

<!DOCTYPE html>  
<html>  
<head>  
 <title>Форма входа</title>  
</head>  
<body>  
 <h2>Вход</h2>  
 <form action="/login" method="post">  
 <input type="text" name="username" placeholder="Login"><br>  
 <input type="password" name="password" placeholder="Password"><br>  
 <input type="text" name="captcha" placeholder="Введите капчу"><br>  
 <img src="data:image/png;base64,{{ captcha\_image }}" alt="Captcha"><br>  
 <input type="submit" value="Войти">  
 </form>  
 <p>{{ message }}</p>  
</body>  
</html>

В “success.html”:

<!DOCTYPE html>  
<html>  
<head>  
 <title>Успешная авторизация</title>  
</head>  
<body>  
 <h2>Успешная авторизация</h2>  
 <p>{{ message }}</p>  
</body>  
</html>