TEMA: Security detection and prevention

1. Условие

Възможност за следене на логовете и откриване на неправомерни действия. Хеширане на пароли. Изграждане на защита срещу session fixation, потребителските cookies, brute force.

2. Въведение

В документа е представен начин за следене на потребителските действия и сигнализиране за нарушение при някои от следните основни защити (които също са представени в документа) свързани със:

неоторизирания достъп, brute force атаки както с логването (ограничаване на броя опити за вход), така и с регистрацията (изграждане на captcha) на даден потребител.

В документа са представени още начини за хешироването на пароли, защити свързани със съхранението на потребителски cookiec.

3. Теория

В тази част от документа са описани всички защити които се следят и при нарушени се сигнализират съответните повреди в таблица предназначена за това. Тези защити са свързани с:

- неоторизирания достъп – основна защита свързана със сравняването и валидирането на данните извлечени от базата данни и данните въведени от потребителя (потребителско име и парола);
- brute force атаки – този вид атаки е съсредоточен основно в това да се „налучка“ даден профил (при логване) или да се препълни паметта с множество неизползвани профили (при регистрация). Тези атаки най често се извършват с помощта на ботове.

В документа са описани още няколко допълнителни защити (тези защити са изградени по такъв начин че да не се налага извършване на допълнителни действия – не се налага сигнализиране в таблица) които да повишат нивото на защитата в системата, като:

- хеширане на пароли – използване на алгоритъм за хеширане повишава сигурността на данните в дадената система. Особено голямо значение хешироването има при паролите които се съхраняват в базите данни. В проекта е използван утвърден алгоритъм за хеширане, а именно **Argon2ID** (едновременното хеширане на данните и защита от memory cost);
- съхранението на потребителски cookiec – добри практики сочат че когато искаме да съхраняваме данни свързани в cookiec е добре да се презапишат тези данни по подходящ начин така че да се избегнат евентуални нападения над потребителски профили. В документа е обяснено подробно какво се има в предвид.

4. Използвани технологии

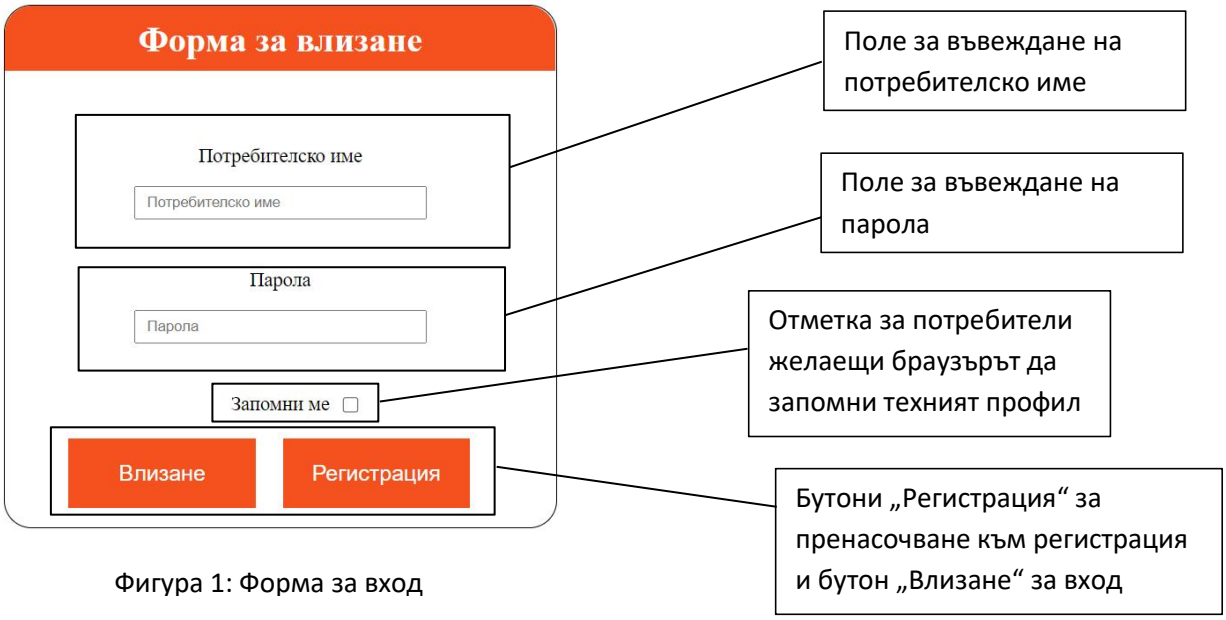
Технологиите използвани в изграждането на проекта са CSS, PHP, MariaDB (mysql), JavaScript

5. Инсталация и настройки

За изпълнението на кода е необходимо единствено да се осигури apache (тествано на xampp apache) и ако се използва xampp да се даде разрешение (да се премахне коментара) на функционалността **extension=gd** намираща се в конфигурационния файл на php. Ако не се използва xampp да се осигури достъп до **gd/gd2** библиотеката в php. За стартирането на проекта трябва да се изпълни файл с име **index.php.**

6. Кратко ръководство на потребителя

Визуализиране на **index.php** файла представящ форма за вход.



Фигура 1: Форма за вход

Визуализиране на **register.php** файла представящ форма за регистрация

Фигура 2: Форма за регистрация

7. Примерни данни

- Регистрация на потребител:
  - Потребителско име: pesho
  - Парола: 123456789
  - Повторна парола: 123456789
  - Секретен код: оказания на картинката (до колкото може да се чете)

- Вход на потребител:
  - Потребителско име: pesho
  - Парола: 123456789

- Качване на снимка, html, css или js: по избор

8. Описание на програмния код

Описание на кода на **index.php** файла:

Това е началният файл на проекта който визуализира форма за вход (Фигура 1). В началото на файла се прави проверка дали има съществуващи потребителски бисквитки за потребителско име и парола (виж Код 1). Ако има се извиква файла **login.php** и се прекъсва изпълнението. Ако не съществуват такива се стартира сесия (чрез командата **session_start()** виж Код 1) и се фиксира ново сесиино ид (чрез командата **session_regenerate_id(true)** виж Код 1)

```
<?php
  if (isset($_COOKIE['user']) && isset($_COOKIE['pass']))
  {
      header("Location: ./login.php");
      exit;
  }
   session_start();
   session_regenerate_id(true);
?>
```

Код 1: Проверка за наличие на потребителски бисквити и стартиране на сесия

След като се изпълнят тези команди се създава посредством HTML код (виж Код 2) формата за вход (виж Фигура 1). Кодът е разделен на две части **head** част в която се извикват стилов файл с име **style.css** и **body** част в която се описват формата за вход която след въвеждане на оказаните полета изпраща данните към **login.php** за валидация.

```
<!DOCTYPE html>
<html>
<head>
   <title>Вход</title>
   <meta charset="UTF-8" lang="bg" />
   <link href="./style/style.css?v=<?php echo time();?>" rel="stylesheet" />
   <link href="./icon/lock_icon.png" rel="icon" type="icon/png" />
</head>
```

```
<body>
   <main>
      <header>
         <h1>Форма за влизане</h1>
      </header>
      <section>
         <form action="login.php" method="POST">
            <label for="user" class="text"> Потребителско име </label>
            <input type="text" name="user" placeholder="Потребителско име" />
            <label for="pass" class="text">Парола</label>
            <input type="password" name="pass" placeholder="Парола" />
            <label class="text" for="remember" >Запомни ме<label>
            <input type="checkbox" name="remember" value="yes" />
            <button  type="submit" class="button" name="login"><span>Влизане</span></button>
            <button  type="submit" class="button"
name="register"><span>Регистрация</span></button>
         </form>
      </section>
   </main>
</body>
</html>
```

Код 2: HTML код за създаването на формата за вход изобразена на Фигура 1

Описание на кода на **login.php** файла:

В този файл се извършва валидацията на данните въведени от потребителя в **index.php** файла. Като първоначално в кода се извиква файл с име **botcheck.php** чрез командата **require** (т.е ако настъпи грешка при изпълнението на файла които е извикан да се прекрати изпълнението на този файл). Повече детайли за този файл може да намерите по – надолу в документа. След което дефинираме три константи **PEPPER** (допълнение за криптирането на паролата и хеширането и в бисквитките), **CIPHER** (метод за шифроване) и **AESKEY** (ключ). След като се декларират константите отново се стартира сесия и се променя сесиното ид. Всички тези декларации може да намерите в Код 3.

Тъй като има два бутона в формата за вход (виж Фигура 1) се прави и проверка дали е избран бутон за регистрация ако да се извиква функцията **add_ip()** (намираща се в файла **botcheck.php**), файла **register.php** и се прекъсва изпълнението на кода (виж Код 3).

```
require("./tools/botcheck.php");

   DEFINE("PEPPER", "g5reoi52ju5r34jgiths");

   DEFINE("CIPHER", "aes-256-ctr");

   DEFINE("AESKEY", 'RE43kl34*&rEweAdYtNnPP[]qwSA7&43');

   session_start();

   session_regenerate_id(true);

   if (isset($_POST['register']))

   {

      add_ip();

      header("Location: ./register.php");

      exit;

   }
```

Код 3: Извикване на файл botcheck.php, деклариране на константите PEPPER, CIPHER, AESKEY, стартиране на сесия и проверка дали потребителя иска да се регистрира

Сега ще дефинираме и основните проверки, като първоначално се прави проверка за наличие на бисквити за потребителското име и паролата. Ако те съществуват се дешифрират с командата **openssl_decrypt** (повече за тази команда може да намерите тук) и стойностите от тези дешифрирания се записват в две променливи **user** (за потребителското име) и **pass** (за паролата, като данните от дешифрирането се конкатенират с константата **PEPPER**). Ако няма налични бисквитки се проверява дали полетата за потребителя и паролата в формата за вход (Фигура 1) са попълнени. Ако да данните от тези полета се записват в променливи **user** (за потребителското име) и **pass** (за паролата, като данните се конкатенират с константата **PEPPER**). Ако няма нито запазени бисквитки, нито въведени данни се извиква файла с име **index.php** и се прекратява изпълнението на програмата. Тези проверки може да намерите в Код 4.

```
if (isset($_COOKIE['user']) && isset($_COOKIE['pass']))

   {

      $user = openssl_decrypt($_COOKIE['user'], CIPHER, AESKEY, 0,
base64_decode($_COOKIE['civ']));

      $pass = openssl_decrypt($_COOKIE['pass'], CIPHER, AESKEY, 0,
base64_decode($_COOKIE['civ'])).PEPPER;
```

```
    }
else if (isset($_POST['user']) && isset($_POST['pass']))
    {
        $user = $_POST['user'];
        $pass = $_POST['pass'].PEPPER;
    }
    else
    {
        header("Location: ./index.php");
        exit;
    }
```

Код 4: Проверка за наличие на бисквитки, въведени данни за вход или нито едно от изброените

Вече може да се премине и към валидацията на данните (проверка дали съществува такъв профил) запазени в променливите **user** и **pass** които съхраняват данни за потребителското име и паролата на потребителя които иска да влезе в системата. За целта предварително е създадена база данни (с име **project**) с таблица (с име **users**) която да съхранява данните на потребителите регистрирани в системата и потребител на базата който има права за селектиране и добавяне на информация от базата (повече за това как е създадена базата и потребителя може да намерите в файл с име **Database data.sql** разположен в поддиректория с име **database information**). Връзката към базата и извличането на данните за паролата посредством името на потребителя което е записано в **user** може да видите в Код 5.

```
    $link = @mysqli_connect("localhost", "login", "login123", "project");
    @mysqli_set_charset($link, "ascii");

    if (!$link)
    {
        echo 'Database maintenance';
        exit;
    }
```

```
$sql = 'SELECT id, pass FROM users WHERE user = ?';


$statement = mysqli_stmt_init($link);
mysqli_stmt_prepare($statement, $sql);
mysqli_stmt_bind_param($statement, "s", $user);
mysqli_stmt_execute($statement);
mysqli_stmt_bind_result($statement, $row['id'], $row['pass']);
mysqli_stmt_fetch($statement);
```

Код 5: Осъществяване на връзка с базата данни и извличане на данни за хеширането пароли за съответния потребител

Данните които са извлечени от базата са id на потребителя (запазено в **row['id']**) и хеширането му парола (запазена в **row['pass']**). След извличането на данните и тяхното запазване се проверява дали има налично такова id, ако няма се извиква функцията **add_ip()** (намираща се в файла **botcheck.php**), премахват се бисквитките (ако има такива) извиква се файла **index.php** и се прекратява изпълнението на програмата. Ако такъв потребител съществува се проверява дали паролата която се съхранява в **pass** съвпада с извлечената хеширане парола която се намира в **row['pass'],** тази проверка се извършва с помощта на вградената функция **password_verify** (повече за тази функция може да намерите тук) и ако те не съвпадат се извиква функцията **add_ip()** (намираща се в файла **botcheck.php**), премахват се бисквитките (ако има такива) извиква се файла **index.php** и се прекратява изпълнението на програмата. Тези проверки може да се видят в Код 6.

```
if (!isset($row['id']))
{
   add_ip();
   setcookie("user", "", mktime(0, 0, 0, 1, 1, 1970));
   setcookie("pass", "", mktime(0, 0, 0, 1, 1, 1970));
   header("Location: ./index.php");
   exit;
}
if (!password_verify($pass, $row['pass']))
{
```

```
    add_ip();
    setcookie("user", "", mktime(0, 0, 0, 1, 1, 1970));
    setcookie("pass", "", mktime(0, 0, 0, 1, 1, 1970));
    header("Location: ./index.php");
    exit;
}
```

Код 6: Проверка дали съществува потребителя и дали въведената парола от него съвпада с тази в базата данни

Накрая се запазват id – то на потребителя, неговото потребителско име и сесиината id в сесиини променливи съответно с имена **uid**, **user** и **user_agent**. След запазването на променливите се прави проверка дали потребителя иска да пази своите данни в бисквитки (т.е проверява се дали потребителя е отбелязал отметката с надпис „Запомни ме“ това може да се види на Фигура 1). Ако той желае данните му да се съхраняват в бисквитки се декларира променлива civ която съдържа генериран псевдо случаен низ от байтове, това става с помощта на **openssl_random_pseudo_bytes** (повече за тази функция може да прочетете тук). След което с помощта на **setcookie** (повече за тази функция може да видите тук) се създават три бисквитки **user** (съдържаща данни за потребителското име на потребителя), **pass** (съдържаща данни за паролата на потребителя) и **civ** (съдържаща низа който се декларира в началото). Всяка от тези бисквитки е валидна един час след което те се премахват автоматично. Накрая се извиква файл с име **securescript.php** и се прекратява изпълнението на програмата. Съхраняването на данните в сесиините променливи и създаването на бисквитките може да видите в Код 7.

```
$_SESSION['uid'] = $row['id'];
$_SESSION['user'] = $user;
$_SESSION['user_agent'] = $_SERVER['HTTP_USER_AGENT'];
    if (isset($_POST['remember']))
    {
        $civ = openssl_random_pseudo_bytes(openssl_cipher_iv_length(CIPHER));
        setcookie("user", openssl_encrypt($_POST['user'], CIPHER, AESKEY, 0, $civ), time()+3600);
        setcookie("pass", openssl_encrypt($_POST['pass'], CIPHER, AESKEY, 0, $civ), time()+3600);
        setcookie("civ", base64_encode($civ), time()+3600);
    }
```

```
    header("Location: ./securescript.php");
    exit;
```

Код 7: Запазване на потребителското id, името на потребител и сесиината id в сесиини променливи и създаване на бисквитките

Описание на кода на **botcheck.php** файла:

В този файл се прави валидация дали системата бива атакувана от робот. Кода се състои в това че първоначално се дефинира функция (с име **insertReport**) която да запазва информация за потребителя, датата и описание на проблемите които са засечени. Отново предварително е създадена таблица (с име **report**) в базата данни (с име **project**) която да съхранява тези сигнали и са дадени права на потребител с име **login** да въвежда данни в таблицата (повече за това как е създадена таблицата и потребителя може да намерите в файл с име **Database data.sql** разположен в поддиректория с име **database information**). Повече за имплементацията на кода може да видите Код 8.

```
function insertReport($user, $description)
{
    $link = @mysqli_connect("localhost", "login", "login123", "project");
    @mysqli_set_charset($link, "ascii");
    if (!$link) {
        echo 'Database maintenance';
        exit;
    }
    $sql = 'INSERT INTO report(users, date, descriptionOfProblem) VALUES
                        ('".$user."',
                         NOW(),
                         '".$description."'
                        )';

    @mysqli_query($link, $sql);
}
```

Код 8: Функция за запазване на потребителско име, дата и час и описаните на настъпил проблем

След което се имплементира функцията **add_ip,** която да следи и запазва (отново в предварително създаден таблица с име **login_logs_hash,** повече за нейното създаване в файла с име **Database data.sql** разположен в поддиректория с име **database information**) броя на опитите за невалиден вход. Повече за имплементацията може да видите Код 9.

```
function add_ip()
  {
    $link = mysqli_connect("localhost", "login", "login123", "project");
    $ip = mysqli_real_escape_string($link, $_SERVER['REMOTE_ADDR']);
    $sql = "INSERT INTO login_logs_hash(ip)
            VALUES (INET6_ATON('".$ip."'))
            ON DUPLICATE KEY UPDATE attempts = attempts+1";

    $result = mysqli_query($link, $sql);
  }
```

Код 9: Функция за отброяване на опитите за невалидни данни за вход

Накрая ако индексът на опитите е стигнал определена бройка заявката която се изпълнява в Код 10 задава на променливата **row['blocked']** стойност 1 и по този начин акаунтът с който е опитано да се влезе е блокиран и се записва в таблица **report** потребителя които се е опитал да влезе, датата и часа и подходящо съобщение.

```
$link = mysqli_connect("localhost", "login", "login123", "project");
  $ip = mysqli_real_escape_string($link, $_SERVER['REMOTE_ADDR']);
  $sql = "SELECT 1 AS blocked
                FROM login_logs_hash
                WHERE ip = INET6_ATON('".$ip."')
                AND attempts > 10";
  $result = @mysqli_query($link, $sql);
```

```
$row = @mysqli_fetch_assoc($result);

if (isset($row['blocked']))

        {

    insertReport($_SESSION['user'], "Засечен е опит за налучкване на парола с това потребителско име евентуално атака от бот, профилът е спрян, необходини са мерки.");

    exit;     }
```

Код 10: Проверка дали брояча отброяващ опитите за вход в системата с невалиден профил е стигнал определен брой (в примера е 10), блокиране на акаунт надвишаващ броя опити и запазване на съобщение за нарушение

В този файл се извършва следене на потребителя които е влял в системата и се представя форма за качване на снимки, html, js и css файлове (виж Фигура 3) за обратна връзка при проблем които не може да бъде фиксира. Като в началото на кода се стартира сесия и се променя сесиното id. След което се декларира функция с име **insertReport** която да запазва потребителското име, дата и час и съобщение на проблема който е засечен (виж Код 8). След което се декларира функция (с име **makeIP**) която връща ip – то на потребителя в зависимост от това къде се намира то (виж Код 11) и функция (с име **makeURL**) която връща адреса на страницата която потребителя е посетил (виж Код 12).

```
function makeIP()

  {

    if (!empty($_SERVER['HTTP_CLIENT_IP']))

    {

       $ip_address = $_SERVER['HTTP_CLIENT_IP'];

    }

    elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR']))

    {

       $ip_address = $_SERVER['HTTP_X_FORWARDED_FOR'];

    }

    else

    {

       $ip_address = $_SERVER['REMOTE_ADDR'];

    }

    return $ip_address;
```

```
}
```

Код 11: Функция връщаща ip –то на потребителя

```
function makeURL()
{
    if (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on')
    {
        $url = "https://";
    }
    else
    {
        $url = "http://";
    }
    $url.= $_SERVER['HTTP_HOST'];
    $url.= $_SERVER['REQUEST_URI'];
    return $url;
}
```

Код 12: Функция връщаща адреса (URL) който е посетен от потребителя

След имплементацията на двете функции се имплементира и трета (с име **insertDataForLogin**) която да следи всеки потребител които е влял в системата с какво id е, от каква операционна система е влял, кои страници е посетил и датата и часа на посещението им. Данните за този потребители се записват в предварително създадена таблица (с име **infoUser**). Повече по кода може да видите Код 13.

```
function insertDataForLogin($user,$ip_address,$os,$url)
{
    $link = @mysqli_connect("localhost", "login", "login123", "project");
```

```
    @mysqli_set_charset($link, "ascii");
    if (!$link) {
        echo 'Database maintenance';
        exit;
    }
    $sql = 'INSERT INTO infoUser(users, ip, os, viewpage, date) VALUES
                            ('".$user."',
                             '".$ip_address."',
                            '".$os."',
                            '".$url."',
                             NOW())';
    @mysqli_query($link, $sql);
}
```

Код 13: Функция въвеждаща данни за потребителите валяли в системата (име на потребител, ip на потребителя, операционна система, адрес на посетената страница и дата и час на посещение) в таблица infoUser

След като са имплементирани всички функции се извиква функцията с име **insertDataForLogin** (със съответните и параметри) и се прави проверка дали id –то на сесията е различно от това което е запазено в сесиината променлива с име **user_agent.** Ако да значи е засечена атака от тип session fixation в резултат на което се записват съобщение в базата, унищожава се сесията и се прекратява изпълнението на системата (виж код 14).

```
insertDataForLogin($_SESSION['user'],makeIP(), php_uname("s"), makeURL());
if ($_SERVER['HTTP_USER_AGENT']!==$_SESSION['user_agent'])
{
    insertReport($_SESSION['user'],"Засечен е Session fixation, в следствие на това правата на потребителя са отнети чак се деиствие.");
    session_destroy();
    exit;    }
```

```
include './uploading file/ uploadForm.php';
```

Код 14: Извикване на функция за следене и записване на действията на влезлият потребител и проверка дали е засечено session fixation

Описание на кода на **register.php** файла:

В този файл се дефинират платформа за регистрация (виж Фигура 2) на потребител в системата. Първоначално се дефинира константа с име **PEPPER** (допълнение за криптирането на паролата), стартира се сесия и се променя сесиното id, след което се дефинират три променливи **time, memory** и **thread** които са необходими за хеширането на паролата. И при стартиране ще се генерира произволен стринг който ще се запазва в сесиина променлива с име **captcha** (виж Код 18).

```
<?php

DEFINE("PEPPER", "g5reoi52ju5r34jgiths");

 session_start();

 session_regenerate_id(true);



$time=74;

$memory=4097;

$thread=1;

$randomString = strtolower(base64_encode(openssl_random_pseudo_bytes(6)));

$_SESSION['captcha'] = $randomString;      ?>
```

Код 18: Дефиниране на константа PEPPER, стартиране на сесия, промяна на сесиино id, задаване на параметри time, memory, thread и генериране на произволен стринг

След което се конструира HTML формата за регистрация (виж Фигура 2) като преди да се пристъпи към въвеждането на данните на потребителя се правят редица проверки:

- дали потребителя е въвел потребителско име, парола и повторна парола
- дали секретният код които е въведен е валиден, ако не е или е пропуснат се генерира нов
- дали двете пароли който потребителя е въвел съвпадат

След което ако всички проверки са валидни се извършва хеширане на паролата използвайки вградената функция **password_hash** (повече за тази функция вижте тук) и алгоритъм за хеширане **Argon2ID** (едновременна сигурност за memory cost и добро хеширане на паролата, повече за този алгоритъм може да прочетете тук). Повече за имплементацията на кода свързана с проверките и хеширането вижте Код 19.

```
<!DOCTYPE html>
<html>
<head>
   <title>Регистрация</title>
   <meta charset="UTF-8" lang="bg" />
   <link href="./style/style.css?v=<?php echo time();?>" rel="stylesheet" />
   <link href="./icon/lock_icon.png" rel="icon" type="icon/png" />
</head>

<body>
<main>
<h1>Регистрация</h1>
<?php
   if (!isset($_POST['user']) || !isset($_POST['pass1']) || !isset($_POST['pass2']))
   {
      goto form;
   }
   if($_SESSION['captcha']!=$_POST['captcha'] || empty($_POST['captcha']))
   {
         $randomString = strtolower(base64_encode(openssl_random_pseudo_bytes(6)));
         $_SESSION['captcha'] = $randomString;

                  echo 'Невалиден код';
                  goto form;
         }
   If (empty($_POST['user']) || empty($_POST['pass1']) || empty($_POST['pass2']))
   {
      goto form;
   }
```

```
    if ($_POST['pass1'] !== $_POST['pass2']) {
        echo 'Паролите не съвпадат<br /><br />';
        goto form;
    }
    $user = $_POST['user'];
    $pass=password_hash($_POST['pass1'].PEPPER, PASSWORD_ARGON2ID,
['memory_cost'=>$memory,'threads'=>$thread,'time_cost'=>$time]);
```

Код 19: Фиксиране на проверки за въведени данни, празни полета, правилен секретен код, съвпадение на паролите и прилагане на алгоритъм за хеширане на паролата

След което записваме потребителското име и хеширането парола в предварително създадена таблица с име **users** и се проверява дали съществува такъв потребител ако всичко е наред потребителя се регистрира в системата, извежда се подходящо съобщение и се предоставя бутон за връщане в формата за вход (виж Фигура 1). Повече за имплементацията на кода може да видите Код 20.

```
$link = @mysqli_connect("localhost", "register", "register123", "project");
    @mysqli_set_charset($link, "ascii");
    $user=@mysqli_real_escape_string($link, $user);
    $pass=@mysqli_real_escape_string($link, $pass);
    if (!$link)
    {
        echo 'Database maintenance';
        exit;
    }
    $sql = 'INSERT INTO users(user, pass) VALUES
                                ("'.$user.'",
                                 "'.$pass.'"
                                )';
    $result = @mysqli_query($link, $sql);
    if (!$result)
```

```
    {
        echo 'Изберете друго потребителско име';
        goto form;
    }
    echo 'Регистрирахте се успешно!<br /><br />';
    goto backButton;
?>
```

Код 20: Запазване на данните на потребителя в базата данни и проверка за наличен профил

Накрая се довършва HTML частта имплементираща и самата форма за регистрация (виж Фигура 2 и Код 21).

```
<header>
        <h1>Форма за регистрация</h1>
    </header>
    <section>
        <?php form:?><form id="myform" action="./register.php" method="POST">
            <label for="user" class="text">Потребителско име<label title="Това поле е задължително"
                class="pointed">*</label> </label>
            <input type="text" name="user" placeholder="Потребителско име" />

            <label for="pass1" class="text">Парола <label title="Това поле е задължително"
                class="pointed">*</label></label>
            <input type="password" name="pass1" placeholder="Парола" />
            <label for="pass2" class="text">Повтори паролата <label title="Това поле е задължително"
                class="pointed">*</label></label>
            <input style="margin-bottom: 50px;" type="password" name="pass2" placeholder="Повторение на парола" />
            <img style=" margin-bottom: 20px; margin-left: 150px; display: block; border: 1px solid black" src="./tools/captcha.php" />
```

```
            <input style="margin-bottom: 50px;" type="text" name="captcha" placeholder="Секретен
код" />

            <button style="margin-left: 45px;" type="submit"
class="button"><span>Регистрираи</span></button>

          <?php backButton:?> <button style="margin-bottom: 50px;" class="button"
onclick="location.href ='./index.php'; return false;"><span>Назад</span></button>

          </form>


      </section>


   </main>


</body>

</html>
```

Код 21: имплементация на гормата за регистрация (допълване)

Описание на кода на **captcha.php** файла:

Този файл конструира изображението на което се представя секретния код (виж Фигура 2). Първоначално се стартира сесия след което се вмъкват няколко функционалности:

- **Content-Type: image/png:** преобразува изображението което се конструира като картина, а не като текст.
- **Cache-Control: no-cache, no-store, must-revalidate:** забранява пазенето на кеш в браузъра;
- **Pragma: no-cache:** позволява изображението да се поддържа от остарели браузъри
- **Expires: 0:** ако браузърът все пак пази кеш, кеширането да става за 0 секунди.

След което достъпваме всички шрифтови файлове (техните дестинации) и ги запазваме стрингове в променлива с име **files** (това става с помощта на вградената функция **glob** повече за нея може да видите тук). Преминаваме се към конструирането на картината с помощта на вградената функция **imagecreatetruecolor** (повече за нея може да видите тук), задаването на цвета на задния фон на картината с помощта на вградената функция **imagecolorallocate** (повече за нея може да видите тук), накрая се обединяват двете неща в едно с помощта на **imagefilledrectangle** (повече за нея може да видите тук).

След като вече е направена картината се обхожда генерирания и запазен в сесиината променлива с име **captcha** низ (виж Код 16). В този цикъл се избира произволен стил и цвят на всяка една буква от низа. Накрая на цикъла се обединяват цвета и стила и се задава позицията на буквата, това става с помощта на вградената функция **imagettftext** (повече за тази функция може да видите тук). Накрая изображението се визуализира ако **png** картина с помощта на вградената функция **imagepng** (повече за тази функция може да видите тук). Цялостната имплементация на този код може да видите в Код 22.

```
<?php
	session_start();
	header("Content-Type: image/png“);
	header("Cache-Control: no-cache, no-store, must-revalidate");
	header("Pragma: no-cache");
	header("Expires: 0");

	$files = glob("../fonts/*.[tT][tT][fF]");

	$image = imagecreatetruecolor(200,60);
	$background = imagecolorallocate($image, rand(150,255), rand(150,255), rand(150,255));
	imagefilledrectangle($image, 0, 0, 200, 60, $background);
	for($i=0; $i<strlen($_SESSION['captcha']); $i++)
	{
		$font = $files[array_rand($files)];
		$fontcolor = imagecolorallocate($image, rand(0,100), rand(0,100), rand(0,100));
		imagettftext($image, rand(16,24), rand(-30, 30), 11+22*$i, rand(26, 58), $fontcolor,
$font, $_SESSION['captcha'][$i]);
	}
	imagepng($image);

?>
```

Код 22: Конструиране на картина изобразяваща секретният код в Фигура 2

9. Приноси на студента, ограничения и възможности за бъдещо разширение

Изграждане на защита срещу едни от най-разпространените атаки в уеб свързани с входа и регистрацията на потребители, следене на профилите на всеки потребител влял в системата (име, ОС, дата и час, посетени адреси и ip) и засичане на настъпилите нарушения в системата в следствие от нарушаването на някоя от имплементираните защити. Възможност за по добро хеширане на пароли, (времево) ограничаване на данните които се записват с цел следене на

потребителите които влизат в системата, възможност за промяна на паролата и подобряване на комуникацията между потребител и администратор при настъпил проблем.

10. Използвани източници

The OWASP, “Session fixation”, [https://owasp.org/www-community/attacks/Session_fixation], статия.

[1] MDN contributors „Using HTTP cookies“ ,[https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies], последно посетен на Apr 13, 2021

[2] The PHP Group, “password_hash”, създадена на 2001-2021 [https://www.php.net/manual/en/function.password-hash.php], документация

[3] The PHP Group, “setcookie”, създадена на 2001-2021 [https://www.php.net/manual/en/function.setcookie.php], документация

[4] CodexWorld, “How to Get IP Address of User in PHP” ,[https://www.codexworld.com/how-to/get-user-ip-address-php/] последно посетена на Feb 7, 2020

[5] JavaTpoint “How to get current page URL in PHP?”, създадена на 2011-2021,[https://www.javatpoint.com/how-to-get-current-page-url-in-php], статия