КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 12
(Продължение)

ТЕМА: Защита и сигурност на информационните системи

ЦЕЛ:

Целта на упражнението е студентите да получат практически знания и умения за разработване на по сигурни и защитени информационни системи чрез HTML и PHP скрипт.

!ВАЖНО: За да можете да тествате упражнението ви е необходим следния софтуер. Web server, Php интерпретатор, MYSQL база данни. Всеки един от тези софтуери ги има в пакетите WAMP или XAMP. Настоящото упражнение е тествано върху WAMP Version 3.2.0.

I. ТЕОРЕТИЧНА ЧАСТ

В предното упражнение се запознахме с някои добри практики при разработване на информационните системи. В сегашното упражнение ще се запознаем с някои мерки за защита. Ето и някои от случаите на които трябва да наблегнат програмистите при разработването на ИС:

1. Противодействие на злонамерено вкарване на данни

Ще разгледаме следните методи за злонамерено вкарване на данни:

- чрез HTML елементи в текстови полета, пароли или многоредови текстови полета на формуляр;
- чрез SQL елементи в текстови полета, пароли или многоредови текстови полета на формуляр.

Тези проблеми са тясно свързани с валидацията на въведените данни и дали данните, които са получени, са точно тези които очаквате. Вторият може да бъде дори и малко по опасен, понеже той чрез него може да се осъществи неоторизиран достъп до базата данни. За да преодолеем тези проблеми може да използваме следните функции:

- *htmlentities(string низ)* – функция която заменя знака „<“ със съответния му код „&lt“ и знака „>“ с „&gt“. Тези кодове са изобразяват в браузъра като „<“ и“>“, но без да ги разпознава като HTML елемент.

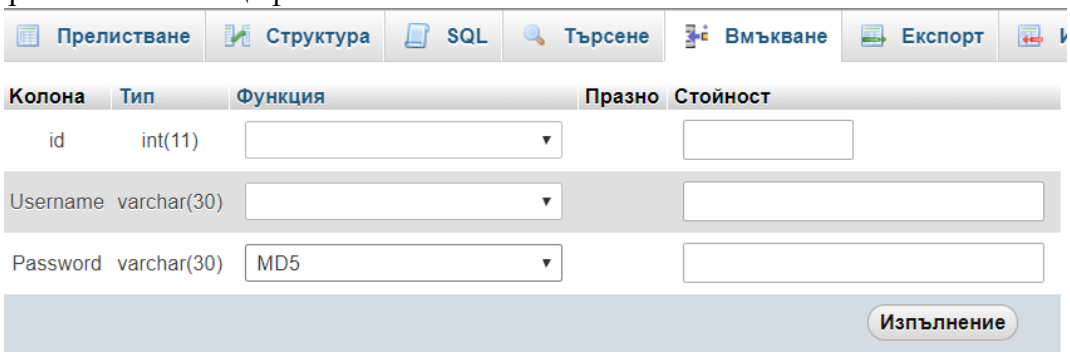
- *mysqli_escape_string()* – функция която замества кавичките в низа на SQL. По този начин се предотвратява възможността SQL интерпретатора да възприеме кавичките неправилно.

За да видите как работят тези функции разгледайте задачи 1 и 2 от практическата част.

2. Идентификация на потребители и пароли.

Тази част засяга това дали наистина този потребител е този за когото се представя. Тя може да се раздели на следните под области:

- проверка за референцията - Ние не бихме искали потребителят или някое злонамерено лице да прескочи логин формата отиде директно на втората страница. Това може да се разреши като се направи една проверка от къде се изпращат данните. Тази проверка може да се направи като проверим стойността на променливата на обкръжението: $_SERVER[“HTTP_REFERER”] която даваше информация за URL адреса, от къде идва потребителя. Ако той не идва от началната страница за вход то може автоматично да се препрати към нея. Също така, за решаването на този проблем би могло да се включат и сесии.
- съхраняване на паролите в криптиран вид – Данните на паролите да се съхраняват във криптиран вид. Предимството е че ако някои злонамерен потребител пробие сигурността на базата от данни то няма да може да разчете паролите на потребителя ни директно. За да създадем криптирана версия на паролите, която да запишем в базата данни може да използваме функцията md5(). Когато поискате да съхраните данни в криптиран формат просто въвеждаме данните в некриптиран формат (по познатия вече начин в чрез phpMyAdmin) и за полето function избирате MD5, както се вижда на фиг.1.
- изискване на сигурни пароли – Това засяга качеството на паролите, които потребителите ползват. Системата ви трябва да дава избор за надеждна парола и да изисква някакъв минимален стандарт. Примерно изискванията може да бъдат следните: минимална дължина; съдържание на главни и малки букви, да съдържа поне една цифра, да съдържа някакъв специфичен символ.



Фиг.1. Криптиране на данни

II. ПРАКТИЧЕСКА ЧАСТ

ЗАДАЧА1: *Създайте формуляр който съдържа два елемента: многоредово текстово поле и бутон, така че потребителя да може да напише произволно съобщение, което след натискането на бутона да се показва на страницата. Въведете следното съобщение в многоредовото поле “<i> Hello, this is in italics”. Разгледайте резултата. Вземете мерки за преодоляване на проблема.*

Код за решение на задача:

```
<html>
<body>
	<h2> Please enter your message:</h2>
	<form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='POST'>
	<p>
		<label for="strMessage"> Message: </label>
		<textarea name="strMessage" id="strMessage"></textarea>
```

```
        </p>
        <p><input type='submit' name='submit'/></p>
        </form>
</body>
</html>
<?php
if (isset($_POST["submit"])){
        $strMessage=$_POST['strMessage'];
        echo "<p>Your Message is $strMessage.</p>";
}
echo "<p>This is some text further down the web page</p>";
?>
```

Пояснение за задача 1

Горния код предоставя възможност на потребителя да въведе съобщение в текстовото поле и при натискане на бутона това съобщение се показва на екрана. В кода не са предприети мерки за HTML вкарвания. Ако потребителя въвежда коректен текст той ще се разпечата на екрана със обикновен шрифт. Нека обаче да тестваме кода като въведем следния текст“. В този текс се съдържда html код „<i>“ който прави текста да излиза под курсив. Понеже тага не е затворен това ще повлияе на цялата страница под това съобщение. Проблема е показан на фиг.2.

Фиг.2. HTML вкарвания на данни

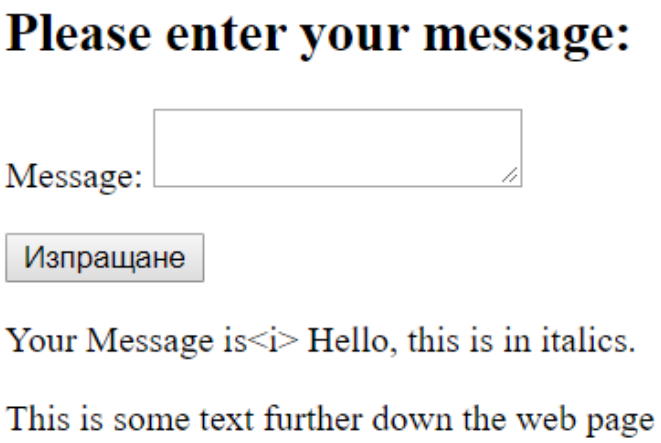
За да преудолеем тази уязвимост е необходимо във следния ред от кода:

```
echo "<p>Your Message is $strMessage.</p>";
```

да използваме функцията „*htmlentities(string низ)*“ . Така реда ще стане следния:

```
echo "<p>".htmlentities("Your Message is".$strMessage).".</p>";
```

Функцията заменя знака „<“ със съответния му код „&lt“ и знака „>“ с „&gt“. Тези кодове се разпознават от браузъра като „<“ и „>“, но без да ги разпознава като начало и край на HTML елемент. На фиг.3. е показано демонстрацията от изпълнението на кода при въвеждането на следния текст *“<i> Hello, this is in italics”*.



Фиг.3. Предотвратяване на HTML вкарване на данни

Задача 2: *Чрез phpmyadmin създайте базата данни "users" със таблица „users" със следните полета:*

ID – int - auto_increment,primary key
Username -Varchar(30)
Password – Varchar(30)

Попълнете таблицата със следните записи:

Запис1- Потребителско име Alan и парола Smith
Запис2- Потребителско име David и парола Regan

Създайте php скрипт, за удостоверяване на достъп с потребителско име и парола. Тествайте кода като предположим че потребителя въвежда Некоректен потребител и не посочи парола или въведе следния текст за потребителско име: „' OR ' 1=1"; и празна парола. Вземете мерки за преодоляване на проблема.

Код за решение на задача:

```
<?php
$strUsername='David';
$strPassword='Regan';
$dblocalhost=mysqli_connect("localhost","root","","users")
or die ("Could not connect:".mysqli_connect_error());
$dbRecords=mysqli_query($dblocalhost,"Select * FROM users WHERE
Username='$strUsername'");
$arrRecords=mysqli_fetch_array($dbRecords);
if($strPassword != $arrRecords["Password"])
      echo "<p>Invalid Password/Username</p>";
else echo "<p>Password and Username match!</p>";
?>
```

Пояснение за задача 2

Скриптът представлява базов вариант за удостоверяване на достъп с потребителско име и парола. В началото записваме потребителско име и парола, които се предполага че потребителя е въвел(за по – просто ние сме ги записали твърдо в кода). Можем да проверим дали името съществува в базата данни, да докладваме проблем, ако липсва, и

накрая да проверим дали паролата, съхранена в базата, съответства на подадената заедно с потребителското име. Скриптът работи.

Какво ще се случи обаче ако потребителя въведе празна парола и посочи следния текст за потребителско име: "' OR '1=1". Тогава нашата заявка става следната:

*SELECT * FROM users WHERE Username='' OR '1=1';*

Този ред дефакто няма върне записи, но тъй като променливата на паролата е празна операторът if, който сравнява подадената парола с тази от базата ще върне стойност true. Затова скриптът ще си мисли че потребителското име и парола съответстват на тези от базата.

Този проблем може да бъде решен чрез функцията mysqli_escape_string(низ). Тя замества кавичките в низа на SQL така че в нашия пример заявката вече би излгеждала така:

*SELECT * FROM users WHERE Username='\' OR \'1=1'*

Така в кода точно преди да изпратим заявката към сървъра трябва да поставим следния ред:

```
$strUsername=mysqli_real_escape_string($strUsername);
```

Заместващите знаципредотвратяват опасността SQL интерпретаторът да възприеме кавичките неправилно. Като допълнение е важно също да се уверим, че в кода е добавена проверка дали заявката към базата данни е върнала някакви записи. Това става като добвим следния оператор If преди оператора за проверка в паролата, като същия ще отиде в графата else на новия оператор:

```
if (mysqli_num_rows($dbRecords)==0)
      echo "<p>Username not found!</p>";
else{
//кода с оператора If проверяващ паролата
}
```

III. Задача за самостоятелна работа.

1. Редактирайте задача 2 от практическата част, така че да се работи с криптирани пароли. Криптирането да се извършва с функция md5().
   Упътване: *За да проверите въведената не криптирана парола от потребителя е необходимо първо да я криптирате и после да сравнявате криптираните стойности.*
2. Напишете скрипт съдържащ функция, която определя силата на генерирана парола. Силата да се изчислява като се дава по 1 точка за изпълнение на всяко от изискванията по-долу (максималният брой точки е 5)
   - Минимална дължина (не по малко от 6 символа)
   - Съдържа главни букви
   - Съдържа малки букви
   - Съдържа поне едно число
   - Съдържа поне един от следните символи !$%^&*@#+