КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 11

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.

I. ТЕОРЕТИЧНА ЧАСТ

При разработването на една информационна система трябва да вземат под внимание и някои други особености на дизайна, а именно сигурността на техния софтуер и защитата на данните му. За да е максимално сигурно и защитено едно уеб приложението, не е достатъчно то да бъде инсталирано на сигурен уеб сървър. Необходимо е да се следва цялостната стратегия за повишаване на качеството и сигурността. Тази стратегия се състои от:

- Развитие на добри навици за програмиране, които да ви гарантират, че приложенията ви са реализирани правилно;
- Познаване на различните форми на атака, които могат да бъдат извършени спрямо приложенията ви, и вземане на мерки за защита от тях;
- Подсигуряване идентификацията на потребителите при влизане в системата и реализация на сигурен механизъм за защита на паролите.

Ето и някои добри практики в разработката на система чрез PHP скрипт:

1. Докладване на всички грешки.

Първият и най-важен съвет при разработването на приложения на PHP е да бъдем сигурни, че получаваме всички сигнали за грешки и предупреждения, които интерпретаторът на PHP ни генерира. Някои по-стари версии на PHP по подразбиране не показват всички съобщения. Затова се налага да погледнем в конфигурационния файл „php.ini“.

Ако сте инсталирали същата версия на WAMP пакета, то може да стигнете до него като кликнете с левия бутон на мишката върху иконата на WAMP, изберете PHP след което изберете файла “php.ini”. Ако сте инсталирали по друг начин php на компютъра, то ще трябва да потърсите този файл директно в Windows.

След като стартирате този файл той се зарежда чрез Notpade и ви позволява да го редактирате. В него трябва да намерим раздела „Error handing and loging“.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Error handling and logging ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; This directive informs PHP of which errors, warnings and notices you would like
; it to take action for. The recommended way of setting values for this
; directive is through the use of the error level constants and bitwise
```

```
; operators. The error level constants are below here for convenience as well as
; some common settings and their meanings.
; By default, PHP is set to take action on all errors, notices and warnings EXCEPT
; those related to E_NOTICE and E_STRICT, which together cover best practices and
; recommended coding standards in PHP. For performance reasons, this is the
; recommend error reporting setting. Your production server shouldn't be wasting
; resources complaining about best practices and coding standards. That's what
; development servers and development settings are for.
; Note: The php.ini-development file has this setting as E_ALL. This
; means it pretty much reports everything which is exactly what you want during
; development and early testing.
;
; Error Level Constants:
; E_ALL             - All errors and warnings (includes E_STRICT as of PHP 5.4.0)
; E_ERROR           - fatal run-time errors
; E_RECOVERABLE_ERROR  - almost fatal run-time errors
……
; Common Values:
;   E_ALL (Show all errors, warnings and notices including coding standards.)
;   E_ALL & ~E_NOTICE  (Show all errors, except for notices)
;   E_ALL & ~E_NOTICE & ~E_STRICT  (Show all errors, except for notices and coding standards warnings.)
;       E_COMPILE_ERROR|E_RECOVERABLE_ERROR|E_ERROR|E_CORE_ERROR (Show only errors)
; Default Value: E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED
; Development Value: E_ALL
; Production Value: E_ALL & ~E_DEPRECATED & ~E_STRICT
; http://php.net/error-reporting
error_reporting = E_ALL
```

В този раздел на файла се настройват начините на докладване на грешки. За да сме сигурни, че получаваме всички грешки е необходимо в последния ред на този код да бъде равен на E_ALL. За да разберете разликата за получаването на грешки разгледайте задача 1 от практическата част.

2. Валидиране на данните от формуляр

Едно от основните средства с които злонамерените потребители могат да повлияят на изпълнението на един скрипт е основният интерфейс, който скриптовете използват за комуникация с потребителите си – формулярите. Те се състоят от най различни полета в което потребителят трябва да въведе данни, но не бихме могли да сме сигурни че въведените данни наистина са във формата в която очакваме. Някои елементи на формуляр се валидират по лесно от други, например списъците, полетата за отметки и радиобутоните. Те са по лесни за валидиране за разлика от текстовите полета, в които потребителя използва по-голяма свобода. (Задача 2 от практическата част).

II. ПРАКТИЧЕСКА ЧАСТ

!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.

ЗАДАЧА1: *Дефинирайте променлива без да и задавате начална стойност. Проверете с оператор if дали дадената променлива е 0.*

Код за решение на задача:

```
<?php

$intNoValue;
If ($intNoValue==0)
      echo "<p>Equals zero</p>"
?>
```

Пояснение за задача 1

По принцип горния код би се изпълнил без да бъдат докладвани грешки а резултатът на екрана ще бъде „Equals zero“. Когато обаче нивото за съобщаване на грешки е направено на E_ALL ще открием че скриптът генерира предупреждение, което ще гласи че променливата не е дефинирана. :Лоша практика на програмиране е да оставяме променливите без стойност, затова е по правилно дефинирането на променливата е по следния начин:

$intNotValue=0;

Когао пишем скриптове е най добре винаги да е включено най високо ниво за съобщаване на грешки

Задача 2: *Да се разработи формуляр който се състои от три елемента: поле за потребителско име, за парола и бутон за изпращане. Нека при натискане на бутона да се показва на екрана потребителското име и паролата.*

Разгледайте примера и установете, какви ограничения са добавени в решението на задачата.

Код за решение на задача:

```
<html>
<h1>Please login:</h1>
<form action="<?php echo $_SERVER["PHP_SELF"]?>" method="post">
<label for="users">USERNAME:</label>
<input name="strUserName" type="text" id="users" maxlength='8'>
<label for="pass">PASSWORD:</label>
<input name="strPassword" type="password" id="pass" maxlength='12'>
<input name="Submit" type="Submit">
</form>

<?php
if(isset($_POST["Submit"])){
      $Username=$_POST["strUserName"];
      $Password=$_POST["strPassword"];
      $intErrors=0;
```

```
        if(strlen($Username)<8){
              echo "Username less then 8 characters";
              $intErrors++;
        }
        if(strlen($Password)<6){
              echo "Password less then 6 characters";
              $intErrors++;
        }
        If($Username{0}!="5"){
              echo "Username doesn't start with a 5";
              $intErrors++;
        }
        If($intErrors==0)
              echo "Username: $Username Password: $Password and all is
              well";
}
```

Пояснение за задача 2

Понеже програмиста неможе да бъде сигурен че потребителя ще въвежда само очакваната от него информация то е необходимо да се направи валидация. Това е постигнато с въвеждането на определени ограничения и проверка за коректност на данните. Ограниченията за този пример са следните:

- Потребителското име трябва да бъде 8 цифрено число започващо с 5 и съдържащо само цифри
- Паролата може да представлява произволна комбинация от символи но трябва да бъде минимум 6 и максимум 12 символа

Ограничението за максимален брои символи в текстовите полета е постигнато чрез свойството maxlength на тага input. Проверката за въведения брой символи в полетата се постига с функцията strlen(). Чрез следната проверка if($Username{0}!=5) се проверява дали потребителското име започва с 5. За да разберем дали потребителското име се състои само от цифри е използвана функцията is_numeric(). В скрипта е включен брояч на осъществените грешки и потребителското име и паролата се показват само, ако не е допусната нито една грешка.

III. Задача за самостоятелна работа.

1. Разгледайте следния код, който трябва да покаже фамилията на „Алан“ три пъти:

   ```
   <?php
   $intCount;
   $arrFirstSurname = array(Simon=>”Jones”, James=>”Smith”, Alan=>“Barclay”,
   Gema =>”West”);
   For($intA=0;$intA<=$intCount;$intA++)
   Echo $arrFirstSurname[Alan]. “ ”;
   ?>
   ```

   Открийте грешките и ги поправете. Ако сте открили само една грешка значи не се стараете достатъчно.
2. Създайте прост формуляр с три елемента: списък, текстово поле и бутон за изпращане. В списъка да се изброяват следните титли Mr, Miss, Mrs, Dr and Other. При натискане на бутон да се изпращат изписа на екрана избраната титла.

Направете така скрипта, че когато от списъка е избран елементът Other то да се чете титлата в текстовото поле. Добавете допълнително ограничение за данните в полето да са с минимална дължина минимум 2 символа.