Systems Programming & Scripting

Lecture 20: PHP Security & Forms & Email

PHP Cryptography

- PHP 5.4 provides several library families for cryptography:
  - Crack
  - Hash
  - Mcrypt
  - Mhash
  - OpenSSL

PHP Cryptography

- PHP 5.4 provides several library families for cryptography:
  - Crack
  - Hash
  - Mcrypt
  - Mhash
  - OpenSSL

PHP with OpenSSL

- The OpenSSL function for encrypting data has the following interface:

```
string openssl_encrypt ( string $data,
string $method, string $password [,
bool $raw_output = false [, string $iv
= "" ]] )
```

PHP with OpenSSL

- The OpenSSL function for decrypting data has the following interface:

```
string openssl_decrypt ( string $data,
string $method, string $password [,
bool $raw_output = false [, string $iv
= "" ]] )
```

Example

```
function do_encrypt($text){
   return openssl_encrypt($text, "AES-128-CBC", "mypassword" );
}


function do_decrypt($text){
   return openssl_decrypt($text, "AES-128-CBC", "mypassword" );
}
```

Sealing and Opening

- Other library functions use a combination of public-key and secret-key encoding for encrypting data, as in session keys
- In this context encryption is called “sealing”
- And decryption is called “opening”

Example

```
// Input is in $data

// seal message, only owners of $pk1 and
$pk2 can decrypt $sealed with keys
// $ekeys[0] and $ekeys[1] respectively.
$len = openssl_seal($data, $sealed, $ekeys,
array($pk1, pk2));
$sealed_base64 = base64_encode($sealed);
$ekey_base64 = base64_encode($ekeys[0]);

// now, send this text
```

Example

```
// Receive data in $received_sealed
// and encrypted key in $received_ekey

// this decrypts the data
openssl_open($received_sealed, $open,
$received_ekey, $pkeyid)

// result is in $open
```

Sending Email Using PHP

- PHP allows sending emails from a server.
- This is supported by using the *mail()* function.

  mail($to, $subject, $body, $headers)
- Requirement to use the *mail()* function:
  - Linux: the *sendmail* binary should be in PATH during script compilation.
  - Windows: mail server details must needs to be specified in php.ini file.

Directives Affecting PHP mail

- A number of configuration directives affect *mail()* behaviour.
- Configured in the php.ini file.
  - *SMTP* (Windows): a string used to specify host name or IP address of the mail server PHP uses to send mail.
  - *smtp_port* (Windows): default 25, an int to specify the port number of the mail server.
  - *Sendmail_from:* a string to set the *Return-Path* header.
  - *sendmail_path:* a string specifying the location of *sendmail.* If set in Windows, *smtp, smtp_port* and *sendmail_from* are ignored.

The *mail()* Function

bool ***mail*** *(string $to , string $subject , string $message [, string $additional_headers [, string $additional_parameters]])*

- *$to,* must be a valid email address comply with RFC 2822*. (Request For Comments for Internet Message Format)
- *$subject,* must comply with RFC 2047**
- *$additional_headers* (optional), e.g. from, cc, bcc
- *$additional_parameters (optional),* to send additional parameters to *sendmail.*
- Returns: TRUE if the mail is **accepted** for delivery. Otherwise FALSE is returned.

* http://www.faqs.org/rfcs/rfc2822

** http://www.faqs.org/rfcs/rfc2047

Example

```
<?php
$to = “testexample@systemsprogscrip.edu";
 $subject = “PHP mail support";
 $message = “This is a program to test mail functionality
 of PHP";
 $headers = "From: programmer@hw.ac.uk\n";
 if(mail($to,$subject,$message,$headers)){
        echo “Email is accepted for delivery";
 }
 else{
        echo “Problem with code sending email”;
 }
?>
```

Forms in PHP

- PHP provides support for dealing with HTML forms.
- HTML Form elements will automatically be available to the PHP scripts.
- $_GET and $_POST PHP variables are used to retrieve form information.

HTML Form Code

```
<html>
<body>
   <form action=“personaldetails.php" method="post">
                  Name: <input type="text" name="name" />
                  Age: <input type="text" name="age" />
         <input type="submit" />
   </form>
</body>
</html>
```

- Sends the form data to *personaldetails.php*
- Displays two text fields with *name* and *age* written as default.
- Has a button to submit the data.

Getting the Form data in PHP

- Two methods to get the form data into the PHP script: using $_GET and $_POST
- Both variables are arrays of variable names and values sent by the HTTP GET or POST methods.
- The $_GET variable is used in the PHP script to retrieve data from a form with *method=“get”.*
  - The variables and the data will be visible as they are added to the URL.
- The $_POST variable is used in the PHP script to retrieve data from a form with *method=“post”.*
  - The variables and the data will be invisible.

GET vs POST

- GET
  - Variable names and values can be sent without a form.
  - Page is cacheable.
  - Page can be bookmarked.
  - Not suitable when passing passwords, etc.
  - Value can’t exceed 100 characters. Not suitable for large data.
- POST
  - Form data not displayed as part of the URL.
  - No data length limit.
  - Page not cacheable.

Handling Form Data

- Part of personaldetails.php:

```
<?
$name=$_POST['name'];
$age=$_POST[‘age'];
?>

<h2>You are <?php echo $name; ?></h2>
<h2>Age < <?php echo $age; ?> years old. </h2>
```

Email Form

```
<form action=“email.php" method="post">
    Name: <input type="text" name="name"><br>
    E-mail: <input type="text" name = "email"><br><br>
    Comments<br>
    <textarea name="comments"></textarea><br><br>
    <input type="submit" value="Submit">
    </form>
```

Exercise

Use the OpenSSL library functions from the start of this slide set to encrypt the body of the email before sending it. Extend the form on the previous slide to enable encryption. To test the implementation, also display the encrypted text and add the possibility to decrypt the text.