25. PHP Secure E-mails

There is a weakness in the PHP e-mail script in the previous chapter.

25.1. PHP E-mail Injections

First, look at the PHP code from the previous chapter:

```
<html>
<body>

<?php
if (isset($_REQUEST['email']))
//if "email" is filled out, send email
  {
  //send email
  $email = $_REQUEST['email'] ;
  $subject = $_REQUEST['subject'] ;
  $message = $_REQUEST['message'] ;
  mail("someone@example.com", "Subject: $subject",
  $message, "From: $email" );
  echo "Thank you for using our mail form";
  }
else
//if "email" is not filled out, display the form
  {
  echo "<form method='post' action='mailform.php'>
  Email: <input name='email' type='text'><br>
  Subject: <input name='subject' type='text'><br>
  Message:<br>
  <textarea name='message' rows='15' cols='40'>
  </textarea><br>
  <input type='submit'>
  </form>";
  }
?>

</body>
</html>
```

The problem with the code above is that unauthorized users can insert data into the mail headers via the input form.

What happens if the user adds the following text to the email input field in the form?

```
someone@example.com%0ACc:person2@example.com
%0ABcc:person3@example.com,person3@example.com,
anotherperson4@example.com,person5@example.com
%0ABTo:person6@example.com
```

The mail() function puts the text above into the mail headers as usual, and now the header has an extra Cc:, Bcc:, and To: field. When the user clicks the submit button, the e-mail will be sent to all of the addresses above!

25.2. PHP Stopping E-mail Injections

The best way to stop e-mail injections is to validate the input.

The code below is the same as in the previous chapter, but now we have added an input validator that checks the email field in the form:

```
<html>
<body>
<?php
function spamcheck($field)
  {
  //filter_var() sanitizes the e-mail
  //address using FILTER_SANITIZE_EMAIL
  $field=filter_var($field, FILTER_SANITIZE_EMAIL);

  //filter_var() validates the e-mail
  //address using FILTER_VALIDATE_EMAIL
  if(filter_var($field, FILTER_VALIDATE_EMAIL))
    {
    return TRUE;
    }
  else
    {
    return FALSE;
    }
  }

if (isset($_REQUEST['email']))
  {//if "email" is filled out, proceed

  //check if the email address is invalid
  $mailcheck = spamcheck($_REQUEST['email']);
  if ($mailcheck==FALSE)
    {
    echo "Invalid input";
    }
  else
    {//send email
    $email = $_REQUEST['email'] ;
    $subject = $_REQUEST['subject'] ;
    $message = $_REQUEST['message'] ;
    mail("someone@example.com", "Subject: $subject",
    $message, "From: $email" );
    echo "Thank you for using our mail form";
    }
  }
else
  {//if "email" is not filled out, display the form
  echo "<form method='post' action='mailform.php'>
  Email: <input name='email' type='text'><br>
  Subject: <input name='subject' type='text'><br>
  Message:<br>
  <textarea name='message' rows='15' cols='40'>
  </textarea><br>
  <input type='submit'>
  </form>";
  }
?>

</body>
</html>
```

In the code above we use PHP filters to validate input:

- The FILTER_SANITIZE_EMAIL filter removes all illegal e-mail characters from a string
- The FILTER_VALIDATE_EMAIL filter validates value as an e-mail address

You can read more about filters in our PHP Filter chapter.