Security

CS174

Chris Pollett

Nov. 5, 2008.

Outline

- Captchas
- Two topics related to captchas
- XSS
- CSRF

Security

- Today we will be looking at various things vaguely connected with security.
- Often data from clients comes to your server via some form. For example, the file upload processing we did last day.
- One annoyance you will have to deal with is robots that find your web forms and upload garbage to your site, spamming you.
- One solution to this problem is to use Captcha's.
- We are first going to talk about Captchas then give two uses for the code we create.

Captcha

- CAPTCHA stands for **C**ompletely **A**utomated **P**ublic **T**uring **T**est to tell **C**omputers and **H**umans **A**part.
- These were developed at Carnegie Mellon around 2000 by Luis von Ahn, Manuel Blum, Nicholas Hopper, and John Langford.
- The basic idea is that you put on your form an image of a distorted string.
- You hope the robot cannot decipher the string from the image so won't be able to fill out that portion of the form correctly:

  Please type the following text:

Making a Simple Captcha in PHP

- The previous CAPTCHA might be created with the code:

```
<p><b>Please type in the following text:</b>
<?php
   $md5 = md5(microtime() * mktime());
   $captcha_string = substr($md5,0,5);
    $captcha_img = imagecreatetruecolor(70, 40);
     $color = imagecolorallocate($captcha_img, 255, 0, 255);
     $line = imagecolorallocate($captcha_img,233,239,239);
     imagestring($captcha_img, 5, 10, 10, $captcha_string, $color);
     imageline($captcha_img,0,0,39,29,$line);
     imageline($captcha_img,40,0,64,29,$line);
     imageline($captcha_img,0,40,64,0,$line);  imagejpeg($captcha_img,
     "images/captcha.jpg",100); imagedestroy($captcha_img);
    // 100 is the jpeg quality
$_SESSION['key'] = $captcha_string;  ?>
<img src="images/captcha.jpg" alt="captcha" style="position:relative; top:15px;"/>
<input type="text" name="key" size="5" /></p>
```

- When the form this is on is submitted the form value for key can be compared with the value for key in $_SESSION.
- The above writes to a file; which is a little bit flaky. You can also use PHP to write to a stream with lines like:

```
header("Content-type: image/jpeg");
imagejpeg($captcha);
```

Thumbnails

- The code for creating a captcha uses the image libraries which can also be useful for creating such things as thumbnails:

```
function createThumb($tname, $base, $filename)    {
    $image = imagecreatefromjpeg($base.$filename);
   //or could have imagecreatefromgif, etc
   $size = getimagesize($base.$filename);
   $thumb = imagecreatetruecolor(50, 50);
   imagecopyresampled($thumb, $image, 0,0, 0,0, 50, 50,
   $size[0],$size[1]);
    imagejpeg( $thumb, $base.$tname,".jpg", 100 );
   imagedestroy($image);
   imagedestroy($thumb);
    }
```

Sending a Mail Message

- It is often useful to collect a person's e-mail address with a form.
- By mailing, a person a special code that allows them to complete a registration process, one can verify that one has a real e-mail address of a real person.
- The simplest way to do this is to use the mail() command:

```
$message = "Here is a mail message";
mail("Someone@somewhere.com",
        "Here is the title",
      $message,
      "From: cpollett@somewhereelse.com");
```

- This could be combined with a captcha to try to reduce the risk of your site spamming other sites.

Attacking Web-sites

- We are next going to look at different ways a web-site might be attacked and we are going to try to find ways to mitigate these attacks.
- These attacks are: XSS, CSRF.

XSS

- XSS stands for Cross-Site Scripting. Notice the initials are actually CSS but that already means cascading stylesheets.
- There are several variants of XSS, some are as simple as embedding bad links into emails.
- Usually, though this vulnerability is caused when:
  - a website has a form on it
  - a malicious user or robot enters data into the form that contains code (say HTML, Javascript)
  - The form data is later displayed to other users, say because it was a comment to a blog entry or a guestbook entry, or a post to a newsgroup, etc.
  - Merely viewing the post causes, cookies, etc of the viewer to be sent to the malicious user who then uses them for evil.

Mitigations

- Validate input from users! Treat all data posted from forms as tainted and use things like htmlentities to encode it before ever displaying it.
- Try to filter certain kinds of input (like scripts)
- If your site uses cookies make sure to tie the cookie to the IP address so it can't be used by a malicious third party.

CSRF

- CSRF stands for Cross-site Request Forgery. Here is the idea of this attack
  - Suppose a user logs into his online bank.
  - The bank stores a cookie on his machine and only uses that to check if he is logged in.
  - The user browses to some other site (not the bank) with malware in it and clicks on an image link that actually contains a URL for the bank together with an action like do a money transfer to a bad person.
  - Since the user still has the banks cookie, the transaction goes ahead without the user even knowing.

Mitigations

- Include user specific tokens in forms, so form data won't be used to take actions on databases unless that token is present.
- Check the referrer of any data sent to your server.
- As a user make sure to log-off sites before browsing elsewhere.