

Server-side PHP Security

Ben Ramsey
Atlanta PHP
2 June 2005

Server-side Security

- ▶ **Definitions & Examples:**
 - ▶ **Cross-site Scripting (XSS)**
 - ▶ **Cross-site Request Forgeries (CSRF)**
 - ▶ **SQL Injection**
- ▶ **Filesystem Attack**
- ▶ **“Hardening” Apache**
- ▶ **“Hardening” PHP**
- ▶ **Running in PHP’s safe_mode**
- ▶ **Tips for include files**
- ▶ **Security by obscurity**

XSS (Cross-site Scripting)

- ▶ Exploits user/browser trust in a Web site
- ▶ Generally involve sites that display external data (forums, Web mail clients, RSS feed readers)
- ▶ Inject content of attacker's choosing
- ▶ Can be used to steal cookies
- ▶ Let's see an example . . .

SQL Injection

- ▶ Users enter SQL commands into unprotected form fields
- ▶ Extremely simple to defend against but many applications remain vulnerable (`add_slashes()`, `mysql_real_escape_string()`, etc.)
- ▶ Let's see an example . . .

CSRF (Cross-site Request Forgeries)

- ▶ Exploits a Web site's trust in the user
- ▶ Sometimes called “session riding,” though this is misleading since the key is that it's a *forged request*
- ▶ Generally involve Web sites that rely on the identity of the users
- ▶ Perform HTTP requests of the attacker's choosing
- ▶ Does not have to be “cross-site”
- ▶ Let's see an example . . .

Filesystem Attack

```
<?php
$d = dir('/home');
while (($entry = $d->read()) !== FALSE) {
    echo $entry . "\n";
}
$d->close();
?>
```

- ▶ **Not yet an attack, but...**
- ▶ **Can see all files 'nobody' user can see**
- ▶ **Can get information about these files**

Filesystem Attack

```
<?php
$d = dir('/home/ramsey');
while (($entry = $d->read()) !== FALSE) {
    echo $entry . "\n";
    $fp = fopen("$d->path/$entry", 'r');
    $fstat = fstat($fp);
    fclose($fp);
    print_r(array_slice($fstat, 13));
}
$d->close();
?>
```

Filesystem Attack

```
<?php
$d = dir('/home/ramsey');
while (($entry = $d->read()) !== FALSE) {
    echo file_get_contents("$d->path/$entry");
}
$d->close();
?>
```

Filesystem Attack

```
<?php  
echo file_get_contents('/etc/passwd');  
?>
```

“Hardening” Apache

- ▶ Apache doesn't block users from seeing/ accessing/modifying files to which it has permission to see/access/modify
- ▶ Apache doesn't log data from POST requests
- ▶ Apache doesn't buffer requests through a validation engine
- ▶ `mod_security` does

What is mod_security?

- ▶ An Apache module
- ▶ Offers the following features:
 - ▶ Can place Apache in a chroot jail
 - ▶ Request filtering
 - ▶ POST payload analysis
 - ▶ Paths and parameters normalized before analysis takes place
 - ▶ HTTPS filtering
 - ▶ Compressed content filtering

chroot with mod_security

- ▶ `mod_security` can set Apache to run in a root jail:

```
SecChrootDir /var/www
```

POST Filtering

- ▶ Can force POST requests to contain certain headers

```
SecFilterSelective REQUEST_METHOD "^POST$" chain  
SecFilterSelective HTTP_Content-Length "^$"
```

POST Filtering

- ▶ Can force POST variables to contain (or not contain) certain values

```
# Only for the FormMail script
<Location /cgi-bin/FormMail.pl>
  SecFilterSelective ARG_recipient "!@benramsey.com$"
</Location>
```

POST Filtering

- ▶ Can force POST requests to accept only certain IP addresses for certain values detected in POST content

```
SecFilterSelective ARG_username admin chain  
SecFilterSelective REMOTE_ADDR "!^127.0.0.1$"
```

Prevent XSS Attacks

- ▶ `mod_security` can be used to prevent Cross-Site Scripting (XSS) attacks by restricting the use of specific tags

Prevent XSS Attacks

```
# Prevents JavaScript  
SecFilter "<script"
```

```
# Prevents all HTML  
SecFilter "<.+>"
```

```
# Allows HTML for a specific field in a script  
<Location /path/to/form.php>  
    SecFilterInheritance Off  
    SecFilterSelective "ARGS!ARG_body" "<.+>"  
</Location>
```

Prevent SQL Injection

- ▶ `mod_security` can be used to prevent SQL injection in requests

```
SecFilter "delete[[:space:]]+from"  
SecFilter "insert[[:space:]]+into"  
SecFilter "select.+from"
```

Prevent Shell Execution

- ▶ **mod_security** can be used to prevent execution from the shell or of operating system commands

```
# Detect shell command execution  
SecFilter /bin/sh
```

```
# Prevent execution of commands from a directory  
SecFilterSelective ARGS "bin/"
```

mod_security Caveats

- ▶ Apache will run slower & use more memory
- ▶ About a 10% speed difference
- ▶ Stores request data to memory in order to analyze it

“Hardening” PHP

- ▶ **Hardened PHP is a patch to the PHP source code; apply before configuring and making PHP**
- ▶ **Here’s what it does:**
 - ▶ **Protects Zend Memory Manager with canaries**
 - ▶ **Protects Zend Linked Lists with canaries**
 - ▶ **Protects against internal format string exploits**
 - ▶ **Protects against arbitrary code inclusion**
 - ▶ **Syslog logging of attacker’s IP**

Hardened PHP in php.ini

▶ Hardened PHP's php.ini directives:

```
; These are the default values
varfilter.max_request_variables    200
varfilter.max_varname_length      64
varfilter.max_value_length        10000
varfilter.max_array_depth         100
```

Hardened PHP & Includes

- ▶ Hardened PHP disallows any include filename that looks like a URL (and logs the attempt to syslog)

```
<?php
include $_GET['action'];

// Hardened PHP will not allow if 'action' is a URL
// (e.g. /script.php?action=http://example.org/
// bad-code.php)
?>
```

Hardened PHP & Uploads

- ▶ When `file_uploads` and `register_globals` are turned on, a POST file upload may be performed on a vulnerable script and the code included
- ▶ Hardened PHP does not allow uploaded files to be included

```
<?php  
include $action;  
?>
```

Null-byte Attacks

- ▶ Hardened PHP protects against null bytes planted within variables
- ▶ Consider the following code:

```
<?php
include "templates/" . $_REQUEST['template'] . ".tmpl"?>

// A null byte code bypasses the .tmpl extension:
// script.php?template=../../../../../../../../etc/passwd%00
?>
```

Overlong Filenames

- ▶ **Hardened PHP will not allow filenames that are too long to be included because this could signal a buffer overflow attack**
- ▶ **Checks that the supplied filename given to the include statement does not exceed the max path length; if it does, it refuses to include it and logs the attack**

Hardened PHP Caveats

- ▶ **Speed impact due to increased cycles performed on sanity checks**
- ▶ **Memory impact due to addition of canaries**
- ▶ **Does not currently allow inclusion of any remote files**
- ▶ **Mainly developed on Linux, so may not work elsewhere**

Running in PHP's `safe_mode`

- ▶ PHP's `safe_mode` tries to solve the shared-server security problem
- ▶ This “problem” should be handled from the Web server or OS level instead; but this doesn't mean `safe_mode` shouldn't be used
- ▶ Only applies to PHP scripts; all other scripts (e.g. Perl, etc.) are unaffected

Running in PHP's `safe_mode`

- ▶ Restricts user access to files they own (regardless of Web server user)
- ▶ Can set an executables directory
- ▶ Can set allowed/protected environment variables
- ▶ Can disable functions and classes
- ▶ Disables/restricts certain functions by default (i.e. `chdir()`, `dL()`, `shell_exec()`)

Running in PHP's `safe_mode`

- ▶ `open_basedir` is often thought of a `safe_mode` directive, but it may be used with `safe_mode` turned off
- ▶ `open_basedir` limits the files that PHP can open to a specific directory, essentially jailing PHP

Tips for Include Files

- ▶ Don't store files with names such as `foo.inc` in the Web root, as they can be read as plain text files
- ▶ In general, store all files not directly accessed by the browser outside the Web root (even `.php` files)
- ▶ No files should be accessed out of context, so don't give users a chance

Security by Obscurity

- ▶ Not a particularly effective means to security by itself, but okay as another line of defense

```
# Make Apache process other files through PHP engine  
AddType application/x-httpd-php .html .py .pl .asp
```

For more information...

- ▶ **mod_security:** <http://modsecurity.org>
- ▶ **Hardened PHP:** <http://hardened-php.net>
- ▶ **safe_mode:** http://php.net/safe_mode
- ▶ **My Web site:** <http://benramsey.com>
- ▶ **PHP Security Consortium:** <http://phpsec.org>

Questions?