# Безопасность и PHP

## Чек-лист

 Смотрим список активных модулей: # php -m Отключаем ненужные: # mv /etc/php.d/sqlite3.ini /etc/php.d/sqlite3.disable

 Отключаем отображение информации о версии PHP: expose_php=Off Это параметр для php.ini.

 Отключаем отображение ошибок PHP для посетителей: display_errors=Off Вместо этого логируем их отдельно: log_errors=On error_log=/var/log/httpd/php_scripts_error.log

 Если не нужна загрузка файлов на веб сервер, отключите эту возможность: file_uploads=Off Если загрузка нужна, то хотя бы ограничьте максимальный размер файла до необходимого предела: file_uploads=On upload_max_filesize=10M

 Отключаем функцию allow_url_fopen, если не используются сайтом. Она открывает широкие возможности для взлома, если разработчики забудут о фильтрации входящих данных. allow_url_fopen=Off

 Настройка размера POST запросов. Этот параметр должен быть не меньше upload_max_filesize, если разрешена загрузка файлов. Если же она запрещена, то большой разрешённый размер post запросов не нужен. Вряд ли вам через формы потребуется заливать большой объём данных. post_max_size=10M или post_max_size=10K

 Подобрать необходимые лимиты выполнения скриптов. Тут всё сильно зависит от самого сайта. В идеале, много ресурсов не выделять, но, к примеру, тот же Bitrix, требует очень много оперативной памяти и времени выполнения для своих скриптов. max_execution_time = 30 max_input_time = 30 memory_limit = 64M

 Отключаем потенциально опасные функции PHP. Оставляем только то, что реально нужно. disable_functions = exec,passthru,shell_exec,system, proc_open,popen,curl_exec,curl_multi_exec, parse_ini_file,show_source

 Убедиться, что параметр cgi.force_redirect не отключен принудительно. По дефолту, если его явно не указать, то он будет включен. cgi.force_redirect=On

 Убедиться, что php работает от отдельного непривилегированного пользователя. Настройка будет зависеть от используемого менеджера процессов php. Проверяем примерно так: # ps aux | grep php

 Ограничиваем доступ php к файловой системе: open_basedir = «/var/www/html/»

 Настраиваем место хранения для сессий: session.save_path = «/var/lib/php/session» Важно убедиться, что туда нет доступа посторонним. Кроме веб сервера эта директория никому не нужна. Также туда не должно быть доступа с сайта.

# Suhosin

Suhosin позволяет бороться с SQL-инъекциями (SQL injections), атаками на переполнение буфера, с отправкой спама через некачественно написанные скрипты, с воровством cookie и т.д. В некоторых случаях может вызывать проблемы, поэтому использовать осторожно. Возможно стоит сразу прописать в /etc/php5/apache2/conf.d/suhosin.ini значение suhosin.get.max_value_length = 4096

Если требуется передавать параметры suhosin через .htaccess то надо сделать следующее:

- Изменить в файле /etc/php5/conf.d/suhosin.ini параметр suhosin.perdir на

```
suhosin.perdir = "p"
```

- Прописать в конфиге Апача для сайта или папки параметр

```
AllowOverride All
```

- Добавить в .htaccess необходимые параметры, например

```
suhosin.post.max_vars = 2048
suhosin.request.max_vars = 2048
```

# Тюним php.ini

- Отключаем выполнение некоторых функций:
  - disable_function=«exec, system, passthru, shell_exec, proc_open»
- Те самые параметры о которых везде понаписано много буковок:
  - magic_quotes_gpc = 1
  - safe_mode = 0
  - register_globals = 0
- Нет, ничего подключать не будем:
  - allow_url_fopen = Off
  - allow_url_include = Off
- Никому не надо знать что за ошибки возникают, а если нужно - посмотрим логи
  - display_errors = Off
  - display_startup_errors = Off
  - log_errors = On
  - error_log = php_error.log
- Запираем php в определенных папках:
  - open_basedir=/var/www/you_mega_site.com:/tmp
- Можно спрятать информацию о версии PHP:
  - expose_php = Off

# Чистота - залог покоя

Проверяем все места со скриптами - нигде не должно валяться беспризорных php-файлов. В

особенности всяких шеллов, приблуд типа phpmyadmin и т.д. Если что-то подобное используется - в отдельную папку с хитрым именем, на папку пароль и закрыть ей листинг из апача.

## Разрешаем некоторые функции PHP для определенных виртуальных хостов

В некоторых случаях требуется запретить использование некоторых функций через disable_functions и сделать индивидуальные настройки для определенных виртуальных хостов. Ан нет, disable_functions работает только из php.ini. Пр попытке включить конструкцию php_admin_value в .htaccess получаем сообщение « php_admin_value not allowed here», а попытки сделать это через конфиги Apache так же не дают эффекта: получаем ошибку «PHP Warning: exec() has been disabled for security reasons in...»

Как оказалось, победить можно, но при условии использования на сервере suhosin. Сборка php5-suhosin 0.9.29-1ubuntu1 старая, я пересобрал руками 0.9.33.

Итак, делаем:

- В php.ini устанавливаем

```
disable_functions =
```

- В suhosin.ini (а если такого нет, то в php.ini) запрещаем требуемые функции, например:

```
suhosin.executor.func.blacklist =
"exec,system,passthru,shell_exec,proc_open"
```

- В настройках виртуального хоста Apache добавляем конструкцию (например, разрешаем exec)

```
php_admin_value suhosin.executor.func.blacklist "system, passthru,
shell_exec, proc_open"
```

## Ищем взломанные файлы

Отсюда: http://habrahabr.ru/post/188878/

### Измененные файлы за 7 дней

```
# find . -type f -name '*.php' -mtime -7
```

### Подсчет хэш-сумм файлов и поиск измененных

```
#!/bin/sh
```

```
$files_path = "/root/bin/data"
$search_path = "/var/www/site/"
$mail_to = "your@mail.ru"

rm  $files_path/before.txt
rm  $files_path/diff.txt

mv $files_path/current.txt $files_path/before.txt

find $search_path -name "*.php" -type f -print0 | xargs -0 sha1sum >
$files_path/current.txt
find $search_path -name "*.js" -type f -print0 | xargs -0 sha1sum >>
$files_path/current.txt
find $search_path -name "*.html" -type f -print0 | xargs -0 sha1sum >>
$files_path/current.txt
find $search_path -name "*.css" -type f -print0 | xargs -0 sha1sum >>
$files_path/current.txt

diff -urN $files_path/before.txt $files_path/current.txt >
$files_path/diff.txt

if [ `ls -la $files_path/diff.txt | awk '{print $5}'` -ne 0 ]; then

 echo "Files changed!"

 mail -s "Files changed!" $mail_to < $files_path/diff.txt

fi
```

### Поиск вставок

```
# find . -type f -name '*.php' | xargs grep -l "eval *(" --color
# find . -type f -name '*.php' | xargs grep -l "base64_decode *(" --color
# find . -type f -name '*.php' | xargs grep -l "gzinflate *(" --color
# find . -type f -name '*.php' | xargs egrep -i "preg_replace
*\((['|\"])(.).*\2[a-z]*e[^\1]*\1 *," --color
```

### Сравнить две копии сайта

```
# diff -r copy-clean/ copy-compromised/ -x список_исключений
```

### Поиск папок, открытых для записи

Ищет директории доступные для записи и php файлы в них. Результат будет сохранен в файл results.txt. Скрипт работает рекурсивно.

```
#!/bin/bash
```

```
search_dir=$(pwd)
writable_dirs=$(find $search_dir -type d -perm 0777)

for dir in $writable_dirs
do
    #echo $dir
    find $dir -type f -name '*.php'
done
```

php, suhosin, security, безопасность, .htaccess, expose php