Federal Office
for Information Security

User Guide

Script Solution for the Hardening of Content Management Systems

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn
Phone: +49 22899 9582-0
E-Mail: bsi-publikationen@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

1 Introduction

During a security survey conducted by the BSI a number of selected content management systems were subjected to extensive security checks. Based on the outcomes of the survey, which covered security and potential points of attack, practical configuration guides were developed in the form of checklists. These checklists can be accessed via the websites of the ISi-series at the following URL: https://www.bsi.bund.de/dok/6620604. To assist the use of the checklists, a dialogue-based script solution was developed. The script solution implements the automatable configuration changes. These steps are marked in the checklists. In combination with a manual implementation of the non-automatable steps, it is possible for a good level of security to be achieved. A description of the use of the script solution is provided below.

The hardening scripts were developed on the basis of a two-tiered solution. A PHP script takes over the hardening of the PHP-based CMS (Joomla!, TYPO3, WordPress). The script essentially adjusts the configuration files of the CMS without changing any other components. Furthermore, a Python script hardens the non-PHP-based CMS (Liferay and Plone), the operating system and the applied system components, such as Apache Webserver, SSH and Firewall for example. The two script solutions can be utilized independently of each other.

The script solution works for the hardening of new installations as well as for existing ones. For a comprehensive hardening of one of the supported PHP-based CMS, following the hardening of the CMS implemented by the PHP script with a subsequent execution of the Python script for the hardening of the operating system and the system components is recommended.

For the non-PHP-based CMS, only run the Python script. Each script solution is described in further detail below.

1.1 Supported components

The dialogue-based scripts for hardening support the following components in their specified version. The components are based on a Debian 8 environment that was used for the security checks. The functioning of the hardening scripts was not tested for other operating systems or for different component versions, and cannot therefore be evaluated.

Operating system

- Debian Linux (8.2)
- OpenSSH (6.7p1), Open SSL (1.0.1k)
- Firewall Rules (iptables)

Content management systems

- Joomla! (3.4.8)
- Typo3 (7.6.2 LTS)
- WordPress (4.4.1)
- Liferay (6.2 CE GA5)
- Plone (5.0)

Runtime environment

- PHP 5.6
- php5-fpm (5.6.14)
- php5-mysql (5.6.14)

- Java 8 (Oracle JDK 8u66)
- Python (2.7.9)

Web and application server

- Apache 2 Web Server (2.4.10)
- Tomcat Application Server tomcat7 (7.0.56-3)

Databases

- MySQL (5.5.46)

In general, it should be noted that adjustments for hardening an Apache Web Server or a PHP runtime environment should not be implemented if server management tools such as Plesk, Confixx, LiveConfig, sysCP or similar are installed.

2 Access to the script solution

The latest version of the script is available to download on the following website:

https://github.com/CMS-Garden/cmshardening

After downloading, unpack the file with the following command:

```
$ tar xfvz FILE
```

As an alternative to a manual download, it is also possible to clone from the terminal using the following command:

- `$ sudo apt-get install git`
- `$ sudo git clone --recursive https://github.com/CMS-Garden/cmshardening`

The hardening scripts are located in the subdirectories

- `hardening-scripts/php`
- `hardening-scripts/python`

In the following chapters, both hardening scripts are described in greater detail.

3 PHP-based script

A PHP script executes the hardening of PHP-based content management systems. It is therefore also possible to execute the script in an environment provided by shared hosting providers where user permissions are very limited. However, you will need shell access in such an environment to execute the script on the server.

At present, the hardening script supports the following content management systems:

- Joomla! (3.4.8)
- Typo3 (7.6.2 LTS)
- WordPress (4.4.1)

3.1 Requirements

To start the script, you need to have PHP installed. Since you are running a PHP-based CMS this requirement should be already met.

At present, the script can be used from the command-line only. However, it is written in such a way that a web application can build upon it for use through the browser. Such an implementation does not currently exist, however.

While executing the hardening script, several files and directories are read and written. It is therefore important that the user account through which the script is run has the required permissions for these files and directories. A table of relevant files and access permissions can be found in chapter 3.5.4, File Permissions.

3.2 Start and start options

The PHP script can be started entirely without parameters. The configuration from the config.yml file is hereby executed. If any specifications are required that have not been defined in the configuration file, the user is prompted to add them.

Further options can be displayed via the help message `--help`:

```
$ php php/index.php --help

The hardening script aims to make content management systems (CMS) safer through the
respective configuration.

The following parameters are supported:

  -h | --help         This help page

  -c | --checkonly    Execution of the script without making changes

  -i | --interactive  Before each change, the user is prompted to decide if the change
                      should be made or not.

  -s | --silent       The user is not prompted before each change whether the change
                      should be made or not.

  -l | --lang         Allows for the language to be changed to 'en' or 'de' (requires
                      the PHP extension 'classkit' or 'runkit')

This parameterization can also be configured in the config.yml. The use of the above
parameters overwrites the configuration temporarily.
```

The simplest command to launch the script is:

```
$  php ./php/index.php
```

In this respect, the following options are active:

–-checkonly

--lang=de

For example, to run the script in interactive mode (prompts before each change) in English, use the following command:

```
$ php ./php/index.php --lang=en --interactive
```

3.3 Executing the script

The script executes the following steps:

1. Displays some information for the user about the risks and use of the script.
2. The user can enter general information, e.g. if the server supports SSL/TLS.
3. A list of supported modules/CMS is shown for the user to make a selection.
4. The script then prompts the user to enter the required information that is specific to the module/CMS, e.g. the path to the configuration file.
5. For each file that is being changed, the script creates a backup file of the original file (configuration files usually), by default in the `./php/backups/` directory.
6. During this step, the actual hardening scripts are executed (so-called Utils). It is shown which configuration is currently active, how it should be configured safely and which file is affected. If there is a difference between the current and desired configuration the application will prompt the user to confirm whether the changes should be implemented. By changing the configuration `Dialog.Interactive = false` in the `config.yml` these confirmation prompts can be deactivated.
7. The script then shows guidelines in form of a checklist for those hardening instructions that are not implemented automatically by the script but have to be configured manually by the user.
8. Once all Utils have been completed, the user is prompted to check whether the system is working. During this step the user can still revert all the changes and go back to the original state of the system.
9. A short note about the backups that have been created is displayed and the script finishes.

3.4 Script configuration

If the user wants to make changes to the default configuration, two different kinds of settings can be adjusted:

- General configuration in `config.yml`, including settings that impact the execution of the script
- Module configuration files under `modules/<modul>.yml` that contain module-specific settings

The configuration files are written in YAML to allow for more complex specifications such as lists and nested code.

3.4.1 General configuration

The content of the YAML file `config.yml` describes settings that determine the general execution of the script. The general settings are:

- `Dialog.Language` to configure the language of the PHP script dialogue. The available values are currently `'de'` and `'en'`. Alternatively, the value can be configured through the command line parameter `--lang`.
- `HardeningSettings.CheckOnly` to activate check-only mode, a mode that executes the script without actually implementing the hardening. Alternatively, this configuration can be changed through the command line parameter `--checkonly`.

3.4.2 Module configurations

Each content management system (module) has its own YAML file in the `modules/` directory. Which of these specific files is called by the script is defined in the `config.yml` file under Hardening.IncludeModules.

The module configurations include module-specific settings. Path to files or directories of the CMS and the hardening guidelines of the module are defined here.

For example, if the user does not want to enter the path to the CMS installation every time the script is running, the path can be included in the configuration file, following the syntax of YAML.

The hardening guidelines do not usually need to be customized by the user. However, if the user chooses to change the values of a configuration, that can be done here. Additional hardening guidelines of a similar kind can also be added without further programming.

3.5 Additional information

3.5.1 Backups

For all automated changes that the script executes on files, a backup of the original file is created in the `backups/` directory. The file name of the backup contains the original file name and a date stamp.

The script creates no backup when changes are implemented manually.

3.5.2 Log files

During the execution of the script, log entries are created to ensure reproducibility of the script at a later point in time. In this respect, similar information is logged, as is shown to the user. For each run of the script, a new log file is created in the `logs/` directory. A time stamp is included in each file name.

3.5.3 Languages

At present, German (`'de'`) and English (`'en'`) are supported. These languages can be configured in the `config.yml` or from the command line with `--lang=xx` or `-l xx`. Further languages can be added analogous to the existing files in the `lang/` directory.

3.5.4 File permissions

Since the script accesses and executes code changes in several files and directories through different actions, file permissions should be checked before executing the script (see 1).

File/Directory	Access	Description
`php/`	read	The script needs permission to read the script files.
`php/backups/`	write	The script creates backup files. If the directory does not exist, the script will create it with the relevant permissions.
`php/logs/`	write	The script creates log files. If the directory does not exist, the script will create it with the relevant permissions.
`php/lang/langcache/`	read/ write	The script needs to create language cache files. If the directory does not exist, the script will create it with the relevant permissions.
`$WordPress/wp-config.php`	read/ write	Hardens the WordPress configuration
`$TYPO3/typo3conf/LocalConfiguration.php`	read/ write	Hardens the TYPO3 configuration
`$JOOMLA/configuration.php`	read/ write	Hardens the Joomla! configuration

File/Directory	Access	Description
$TYPO3/ENABLE_INSTALL_TOOL	delete	Deletes the markers that activate the installation tools

Table 1: File permissions

The list is not necessarily complete. Further hardening measures can affect additional files and directories. Placeholder such as $JOOMLA write to root directories of CMS installations that have been specified by the user.

4 Python-based script

The Python-based script hardens the operating system, the general system components and the non-PHP-based content management systems. In the current version, the following components are supported:

Operating system

- Debian Linux (8.2)
- OpenSSH (6.7p1), Open SSL (1.0.1k)
- Network, Firewall Rules (iptables)

Content management systems

- Liferay (6.2 CE GA5)
- Plone (5.0)

Runtime environment

- PHP 5.6-FPM
    - Specifics for Wordpress
    - Specifics for Joomla
    - Specifics for Typo3
- Java 8 (Oracle JDK 8u66)
- Python (2.7.9)

Web and application server

- Apache 2 Web Server (2.4.10)
    - Specifics for Wordpress
    - Specifics for Joomla
    - Specifics for Typo3
- Tomcat Application Server tomcat7 (7.0.56-3)

Databases

- MySQL (5.5.46)

If server management tools such as Plesk or Confixx are installed on the system, the changes to PHP and Apache should be made carefully, otherwise the functionality of the server management tools may be affected.

4.1 Requirements

The script can be used for the hardening of new and existing installations. The script is run through the command line and requires “sudo” permissions since several files and directories are read, written and deleted, and changes are made to permissions, ownership of files and directories during the execution and hardening. It is therefore

meant to be used with web server that allow full access. For this reason, the script is not suitable for execution on shared hosting environments.

The execution of the script requires a recent Python installation of major version 2 or 3 and requires the following additional Debian packages:

- python-yaml
- python-netifaces
- python-six
- python-lxml
- netfilter-persistent
- iptables-persistent

These can be installed with the following command:

```
$ sudo apt-get install python-yaml python-netifaces python-six python-lxml
netfilter-persistent iptables-persistent
```

The scripts are to be run locally and do not support remote hardening.

4.2 Start and start options

To run the script, no further configuration is needed. Required configurations are either detected by the script itself, or the user is prompted to enter the relevant information while the script is running.

The script needs to be run with the sudo command and cannot be executed as root. This is necessary because the hardening of OpenSSH limits the user that can register via SSH on the server. During this process, the script detects the current user and adds the user to this list.

A list of all options is displayed via `--help`:

```
$ sudo python hardening.py --help
optional arguments:
  -h, --help            Show this help message and exit
  --mode {interactive,diff,silent,check-only}
                        Select the mode in which this script will run. Default
                        is check-only.
  --lang LANG           Select language for this tool. Default is your system
                        locale or English. Valid options are de_DE (or only
                        'de') and en_US (or only 'en').
  --log logfile         Display a log of applied changes; you can use '-' as
                        filename to log to stdout
  --version             Display the version of this tool.
  --documentation       Display the full documentation in markdown format.
```

The most basic way to start the script is through the command:

```
$ sudo python hardening.py
```

In this respect, the following options are active:

--mode=check-only

--lang=de_DE

--log=/root/TODO

The available options are explained below:

interactive:

In interactive mode, the script will ask which modules should be executed. Once these are selected, the hardening is executed. However, the script will not execute automatically but display which changes are made during each step of the process before they are actually implemented. This gives the user the optionto decide whether to implement the hardening or not, to skip a particular step or to abort the hardening altogether.

If the hardening is aborted, no changes are made, even changes that had been allowed earlier in the hardening process. Instead, a complete rollback is performed.

diff:

In diff mode, the user is also prompted to select which modules to execute via an interactive menu. The actions are sorted by content – for example all changes in a particular file – and are shown to the user as differences to the current configuration. As in interactive mode, the user has the option to decide whether to implement the hardening or not, to skip a particular step, or to abort the hardening altogether. However, the sorting reduces the frequency of these prompts.

Aborting the hardening will prevent any changes – as during interactive mode.

silent:

In silent mode, the hardening which is executed is displayed to the user which, however, there is no prompt that requests information or consent from the user. The hardening modules that are executed are determined via the file “hardening-scripts/python/config.yml“ (for more information see chapter 4.4.).

To abort the script you can use <Strg-C>. The changes made during the hardening will then be reverted if they are revertible.

checkonly:

The checkonly mode is the default mode for the hardening script. The checkonly mode is identical to interactive mode except that no changes are made in this mode and only the text dialogue is displayed.

4.3 Executing the script

The script executes the following steps:

1. Displays explanations for the user about the risks and the use of the script.
2. A list of the supported modules is shown to the user to make a selection. (In silent mode, the modules from the general configuration are initialized.)
3. It is possible to prompt the user to enter general information such as the network interface which should be used.
4. During these steps, the actual hardening scripts are executed (so-called Utils). The user is shown which changes are implemented during each step of the hardening process. Depending on the selected mode, the user can determine whether a particular hardening step should be implemented. During the process, the script will create a backup in the „/root/hardening-backup/“ directory of each file or directory that has been changed.
5. After displaying a note about the required reboot of the system, the script finishes.

4.4 Script configuration

Should the user want to change any of the preconfigured values, there are two different kinds of settings that can be adjusted:

- General configuration in `config.yml` including settings that impact how the script is run.
- The module configurations are under modules/<modul>.yml and contain module-specific settings.

The configuration files are written in YAML to allow for more complex specifications, such as lists and nested code.

4.4.1 General configuration

The code in the YAML file config.yml defines several settings that determine the execution of the script in general. These are described in more detail in the file. The most important settings are:

- `RunModules` and `CMS` to select the executing modules and to activate CMS-specific settings for the modules. Valid values are:
  - `RunModules: [apache, debian, java, mysql, network, php, python, sshd, tomcat]`
  - `CMSModules: [joomla, liferay, plone, typo3, wordpress]`

  The settings are used in silent mode only. For all other modes, the user is prompted to enter these values.
- `HardeningSettings` define very specific information that the user configures. Changes to these settings are generally not necessary, but they provide experienced users with a simple way to configure commonly adjusted values in the general configuration (such as the port for the SSH service).

4.4.2 Module configuration

For each module, there is a YAML file in the `modules/` directory. Which of these files is called by the script is determined by the user during execution. For silent mode, this is set in the `config.yml` under RunModules.

The module configuration files include module specific settings. The path to the files or directories of the CMS and the hardening guidelines of the module are defined here. All parameters are described further in the corresponding files.

The hardening guidelines do not generally require customization by the user. However, if the user chooses to change values of a configuration, it can be done here. Also, additional hardening guidelines of a similar kind can be added without further programming.

4.5 Additional information

4.5.1 Backups

For all the automated changes that the script executes on files, a backup of the original file is created in the directory that is defined in the global configuration file. By default, this is located under `/root/hardening-backups/`. The file name of the backup contains the original file name and a date stamp.

4.5.2 Log files

The `-log` parameter defines a path for the log. The log describes the changes that have been made on the system. Is the `-log` parameter not used, the log is by default located in the home directory of the superuser. To display the log in the terminal use '-' as value.

4.5.3 Languages

At present, German (`'de'`) and English (`'en'`) are supported. These languages can be configured from the command line with `--lang=xx`. Further languages can be added analogous to the existing files in the `locale/` directory. The default language is German.

4.5.4 Color output

The installation of the `coloredlogs` package is recommended to enable color output in the terminal.

The installation can be achieved as follows:

```
sudo apt-get install python-colorlog
```

or when using Python3 with the following command:

```
sudo apt-get install python3-colorlog
```