KNEBLER.DE

27.02.2011 - PHP effektiv gegen Hacker-Angriffe absichern

PHP kann Dateien auf den Server schreiben, ausführbaren Code von Webadressen nachladen oder Linux-Systemkommandos mit den Rechten des Webservers ausführen. Meist sind diese Möglichkeiten nützlich und notwendig. Doch viel zu oft lässt sich, dank Fehlern, eine Funktion von außen missbrauchen, um Schädlingsprogramme auf dem Webserver zu installieren.

Deshalb sollte sich jeder Webserver-Administrator dringend mit den Konfigurationsmöglichkeiten von PHP4 beziehungsweise PHP5 vertraut machen, und nur so viele Funktionen zulassen, wie auf dem Server auch tatsächlich benötigt werden. Die dafür zuständige Datei heißt php.ini und liegt, je nach Version von Webserver und PHP, in einem der folgenden Verzeichnisse:

/etc/php4/apache/

/etc/php4/apache2/

/etc/php5/apache/

/etc/php5/apache2/

Nach jeder Änderung empfiehlt sich ein

apachectl restart

oder

apache2ctl restart

Erst mit dem Neustart werden die Ã"nderungen wirksam. Das graceful-Kommando des Skripts apachectl reicht dazu nicht immer aus.

Gehen Sie die folgenden Anweisungen Schritt für Schritt durch und setzen Sie den vorgeschlagenen Wert in Ihrer php.ini. Setzen Sie den Wert auch, wenn er die Voreinstellung (default) ist. Denn die jeweils gültigen Voreinstellungen ändern sich des öfteren von PHP-Version zu PHP-Version. Angegeben sind die Defaults für PHP5, bei PHP4 sind sie sehr versionsabhängig.

Und das sind die Befehle:

register_globals = off ;(php5-default: off)

verhindert, dass beliebige Variable im PHP-Code durch GET- oder POST-Parameter überschrieben werden können. Beim Aufruf der URL http://server/page.php?login=true ist beispielsweise der Wert in der Variablen $_GET["login"] abgelegt. Mit

register_globals=on

wird zusätzlich alleine über den URL-Parameter die PHP-Variable $login gesetzt. Dieser Parameter sollte darum heutzutage unter keinen Umständen mehr auf on gesetzt werden.

memory_limit = 12M ;(default: 8M)

begrenzt den Speicher (in Megabyte), der bei jedem einzelnen Aufruf eines Skripts verbraucht werden darf. So wird beispielsweise verhindert, dass eine versehentlich programmierte Endlosschleife den kompletten verfügbaren Speicher belegt. Viele PHP-Anwendungen sind recht speicherhungrig, so dass der Wert eventuell höher gewählt werden muss. Das CMS Typo3 benötigt beispielsweise eine Grenze von mindestens 16MB.

max_execution_time = 60 ;(default: 30)

begrenzt die Zeit, die ein Script für die Ausführung bekommt. Auch das verhindert Probleme mit versehentlichen Endlosschleifen. Allerdings darf der Wert nicht zu niedrig sein ? schließlich soll auch bei voller Serverlast etwa eine aufwendige Datenbankabfrage ohne Fehlermeldung bearbeitet werden.

max_input_time = 60 ;(default: 60)

begrenzt die Zeit, die ein Script mit dem Einlesen der übergebenen Parameter verbringen darf.

default_socket_timeout = 60 ;(default: 60)

begrenzt die Zeit, die PHP bei der Ã?bertragung von Streams auf die Gegenseite wartet.

allow_url_fopen = off ;(default: on)

Defaultmäßig erlaubt es PHP, bei allen Befehlen, die Dateien öffnen, statt eines lokalen Dateinamen auch eine URL anzugeben. Das betrifft beispielsweise die Befehle require, include oder fopen. Setzen Sie diesen Wert nur auf on, wenn Sie diese Funktionalität unbedingt benötigen. Denn schon ein include($_GET["filename"]) stellt ein erstklassiges Sicherheitsrisiko dar. Hier sind allerdings auch die PHP-Programmierer gefragt. Denn solche Konstruktionen sind leichtsinnig.

open_basedir = /var/www/:/usr/share/pear/ ;(default:nicht gesetzt)

Dieser Parameter schränkt das Ã?ffnen von Dateien weiter ein. PHP läßt nur noch den Zugriff auf solche Dateien zu, die in oder unterhalb der angegebenen Pfade gespeichert sind. Allerdings sollten Sie hier nicht zu restriktiv sein: Liegt ein PHP-Programmpaket, wie etwa phpMyAdmin, in einem Pfad außerhalb des üblichen www-Verzeichnisses, muss dieser Pfad mit aufgeführt werden, sonst wird das Programmpaket nicht mehr funktionieren.

session.save_path = /var/tmp/www ;(default: nicht gesetzt)

gibt das Verzeichnis an, in dem PHP seine Session-Informationen ablegt.

Aus Sicherheitsgründen sollte dafür nicht das Standard-Tempverzeichnis genutzt werden.

upload_tmp_dir = /var/tmp/www ;(default: nicht gesetzt)

gibt das Verzeichnis an, in dem PHP hochgeladene Dateien ablegt. Auch dafür sollte keinesfalls das Standard-Tempverzeichnis verwendet werden.

upload_max_filesitze = 2M ;(default: 2M)

begrenzt die Größe der Dateien in Megabyte, die via PHP-Funktionen hochgeladen werden können. Der ideale Wert hängt davon ab, für welche Anwendungen PHP eingesetzt wird. Sie können auch einen größeren Wert, etwa 20M wählen, nur unbegrenzt sollte die Dateigröße nicht sein.

enable_dl = off ;(default: on)

Die Standardeinstellung aktiviert den PHP-Befehl dl(), der PHP-Extensions im Programmcode nachlädt. So verlieren Sie aber die Kontrolle, welche Extensions auf Ihrem Webserver verwendet werden. Stellen Sie diesen Parameter also besser auf

off und laden Sie Extensions über die php.ini, etwa

extension=mysql.so

PHP-Parameter individuell einstellen

Aufwendiger wird die Einstellung, wenn nicht für den Webserver einheitliche Werte gelten sollen, etwa, weil der Server unter verschiedenen URLs verschiedene PHP-Anwendungen anbietet. Doch auch dann ist es möglich, sichere Einstellungen zu erreichen. Dazu setzt man zunächst in der php.ini alle global geltenden Werte. Alles Weitere passiert dann in den Config-Dateien des Apache-Werbservers. Dafür gibt es zwei Config-Befehle:

php_admin_flag

php_admin_value

Diese beiden Befehle können auch innerhalb eines

Blocks verwendet werden, so dass sie für einen virtuellen Server gelten. Beispiel:

(...)

php_admin_flag allow_url_fopen off

php_admin_value memory_limit 10M

php_admin_value open_basedir /home/ vserver1/:/usr/share/ pear/

Beachten Sie die Schreibweise ohne =-Zeichen.

Wenn auf Apache mehrere virtuelle Server laufen, sollten Sie open_basedir, session.save_path und upload_tmp_dir für jeden Host getrennt setzen.

Achtung, Fußangel

Nicht alle Beschränkungen, die Sie in der php.ini setzen, können in der Apache-Config wieder aufgehoben werden. Ist in der php.ini etwa allow_url_fopen auf off gesetzt, bleibt es auf off, was auch immer Sie in Apache einstellen.

Verwenden Sie in diesem Fall folgenden Trick:

allow_url_fopen = on

in der php.ini.

php_admin_flag allow_url_fopen off

in der globalen Apache-Konfiguration. Und dann wiederum

php_admin_flag allow_url_fopen = on

innerhalb des Virtualhosts, der diese Funktion benötigt. Um zu überprüfen, welche Einstellungen innerhalb eines Hosts tatsächlich gelten, verwenden Sie ein kurzes Skript mit dem Befehl phpinfo().

Noch mehr Sicherheit für PHP

Darüber hinaus gibt es weitere Parameter, um PHP noch stärker abzusichern, etwa

safe_mode =

disable_functions =

Setzt man allerdings diese Parameter auf sichere Werte, schränkt man die Funktionalität von PHP so stark ein, dass viele Anwendungen nicht mehr oder nur noch eingeschränkt funktionieren. Wer diese Parameter ändert, sollte deshalb danach seine Anwendungen ausführlich testen.