Fehler beheben, Sicherheit erhöhen

Dauerhafte Konfiguration der Anzeige von Fehlern

CODERS.BAY

Eine Möglichkeit, die Anzeige von Fehlern zu steuern, bieten die Parameter in der Konfigurationsdatei *php.ini.* Die Datei beinhaltet unter anderem die folgenden Parameter (mit Beispielwerten):

```
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT

display_errors = Off

log_errors = On

error_log = php_errors.log
```

Dauerhafte Konfiguration der Anzeige von Fehlern

CODERS.BAY

- Der Parameter `error_reporting` filtert die anzuzeigenden Fehler. Bei den Werten handelt es sich um eine Kombination von Konstanten, die mit Bit-Operatoren verbunden werden

Konstante	Zahlenwert	Bedeutung
E_ERROR	1	Fehler
E_WARNING	2	Warnung
E_NOTICE	8	Hinweis
E_DEPRECATED	8192	Warnung zu einem Codeteil, der in einer zukünftigen PHP-Version nicht mehr funktionieren wird
E_ALL	32767	Alle Fehler, Warnungen und Hinweise
E_ALL & ~E_NOTICE	32759 (= 32767 – 8)	Alle Fehler außer Hinweisen
E_STRICT	2048	Hinweis zur besseren Vorwärtskompatibilität des Codes
E_ERROR | E_WARNING	3 (= 1 + 2)	Fehler und Warnungen

Operator	Bedeutung
|	Bitweises logisches Oder
&	Bitweises logisches Und
~	Bitweises logisches Nicht

Dauerhafte Konfiguration der Anzeige von Fehlern

CODERS.BAY

Mit dem Parameter display_errors wird entschieden, ob Fehler auf dem Bildschirm ausgegeben werden. Mögliche Werte sind `On` und Off

Der Parameter log_errors entscheidet darüber, ob Fehler in einer Log-Datei (mit Datum und Uhrzeit) gespeichert werden. Die möglichen Werte sind `On` und `Off`

Mithilfe des Parameters error_log wird festgelegt, ob und in welcher Datei auf dem Webserver Fehler gespeichert werden, falls `log_errors` auf `On` steht

Parameter	Entwicklung	Produktion
Error_reporting	E_ALL	E_ALL & ~E_DEPRECATED & ~E_STRICT
display_errors	On	Off
log_errors	On	On
error_log	php_errors.log	php_errors.log

Dauerhafte Konfiguration der Anzeige von Fehlern

CODERS.BAY

Um herauszufinden welche Einstellungen gelten, hilft `ini_get()`

```
<!DOCTYPE html><html><head><meta charset="utf-8"></head><body>
 <?php
  echo "error_reporting: " . ini_get("error_reporting") . "<br>";
  echo "display_errors: " . ini_get("display_errors") . "<br>";
  echo "log_errors: " . ini_get("log_errors") . "<br>";
  echo "error_log: " . ini_get("error_log") . "<br>";
 ?>
</body></html>
```

- Der Parameter `error_reporting` steht auf `E_ALL & ~E_DEPRECATED & ~E_STRICT;` daher wird der Wert `22527` ausgegeben.

- Der Parameter `display_errors` steht auf On, daher erfolgen Ausgaben.
  Der Parameter `log_errors` steht ebenfalls auf On. Bei `error_log` steht der Name der Log-Datei.

Temporare Konfiguration der Anzeige von Fehlern

CODERS.BAY

Falls es keine Möglichkeit gib, die Datei `php.ini` zu modifizieren, kann der Parameter auch über die Funktion `ini_set()` verändert werden. Gilt allerdings nur für das aktuelle Programm.

```
<!DOCTYPE html><html><head><meta charset="utf-8"></head><body>
 <?php
  ini_set("error_reporting", 1);
  ini_set("display_errors", 1);
 ?>
</body></html>
```

- Es werden alle Arten von Fehlern angezeigt
- Die entsprechenden Anweisungen können natürlich auch in eine `include`-Datei ausgelagert werden.

Angriffe & Sicherheit

- Webserver im Internet sind häufig Angriffen ausgesetzt. Es gibt keine völlige Sicherheit. Man sollte aber versuchen, den Grad der Sicherheit so weit wie möglich zu erhöhen, ohne dass die Benutzbarkeit des Systems eingeschränkt wird. Viele Angriffe sind automatisiert und unterscheiden nicht nach der Systemgröße.

- Untersuchungen zeigen, dass ein beachtlicher Anteil der Sicherheitslücken durch unzureichende Programmierung entsteht.

PROGRAMMPAKETE

Es existieren fertige Programmpakete, die dem PHP-Programmierer einen Teil seiner Arbeit abnehmen können. Bei diesen Paketen sollte man sich darüber im Klaren sein, dass ihr hoher Verbreitungsgrad auch dazu führen kann, dass ihre Sicherheitslücken ebenfalls verbreitet sind. Man sollte darauf achten, dass die Pakete weiterhin vom Hersteller gepflegt werden und dass auch an der Beseitigung von Sicherheitslücken gearbeitet wird.

Sichtbare Daten

Daten, die nur der Programmierer benötigt, die aber für den Benutzer unwichtig sind, sollte man nicht sichtbar machen. Alle sichtbaren Daten ergeben Informationen, die für Angriffe genutzt werden können. Im Einzelnen bedeutet dies:

- Daten, die von einem PHP-Programm gesendet werden, sollten möglichst nicht (mit den Zeichen `&` und `?`) an die URL angehängt werden (`$_GET[…]`), da sie anschließend lesbar in der Adresszeile des Browsers erscheinen
- Daten sollten nicht mithilfe von Formularelementen des Typs hidden von einem PHP-Programm zum nächsten übertragen werden. Sie sind in der Quellcodeansicht des Browsers lesbar.
- Zur Datenübermittlung sollte man die Sessions verwenden
- Hinweise, Warnungen und Fehlermeldungen sollte man auf einem Produktionssystem nicht ausgeben lassen
- Das Zeichen `@` (der sogenannte Silence-Operator) direkt vor einem Funktionsnamen dient dazu, eine eventuelle Fehlermeldung der Funktion zu unterdrücken (zum Beispiel: `@fopen(test.txt)`).

SESSIONS

CODERS.BAY

Bei der Übermittlung von Daten zwischen PHP-Programmen bieten Sessions eine recht hohe Sicherheit. Aber auch hier gibt es Verbesserungsmöglichkeiten

- Informationen über die Session werden in einem Cookie auf dem Client-PC festgehalten. Die Lebensdauer der Cookies kann man mithilfe der Funktion `session_set_cookie_params()` beeinflussen, sodass diese nur eine begrenzte Zeit auf dem PC verbleiben. Allerdings gelten die Einstellungen nur für das aktuelle PHP-Programm. Diese Funktion muss vor der Funktion session_start() aufgerufen werden
- Situation: Der Benutzer entfernt sich während einer Session vom Arbeitsplatz und hinterlässt seinen PC unbeaufsichtigt. Abhilfe: Den Startzeitpunkt einer Session in einer Datenbank auf dem Server festhalten und die Gültigkeit einer Session nach Ablauf einer maximal erlaubten Zeit automatisch beenden
- Die Funktion `session_regenerate_id()` erzeugt eine neue Session mit einer neuen ID; dabei werden alle Daten der alten Session übernommen. So ist es einem Angreifer nicht mehr möglich, mit einer ihm bekannt gewordenen Session-ID oder einer von außen vorgegebenen Session-ID an weitere Daten zu gelangen
- Mithilfe der Funktion `session_cache_limiter()` kann man das Speichern von Seiten, die auch Session-Daten beinhalten, im Cache des Client-PCs oder auf einem Proxy vermeiden

VARIABLEN

Der Zugriff, das Hinzufügen oder das Auslesen von Variablen der PHP-Programme sollte nicht möglich sein. Man sollte daher Folgendes beachten:

- Manche Websites laufen noch unter der PHP-Version 5.3 oder einer noch älteren Version. Bei diesen früheren Versionen konnte man den Schalter register_globals in der Datei php.ini sicherheitsgefährdend auf On stellen. Seit PHP 5.4 gibt es diesn Schalter nicht mehr. Die Inhalte von gesendeten Formularen können seither nur noch mithilfe des Arrays `$_POST[…]` verarbeitet werden, wie man es gewohnt ist
- Variablen können vor ihrer Nutzung initialisiert werden und nach ihrer Nutzung mithilfe der Funktion `unset()` gelöscht werden
- Den Gültigkeitsbereich von Variablen kann man kleinhalten, wenn ein Programm stark modularisiert, das heißt in Funktionen zerlegt wird.

Eingaben prüfen

Formulare bieten viele Angriffsmöglichkeiten, da es hier notwendig ist, eingegebene Daten vom Client an den Server zu übertragen. Abhilfe kann Folgendes schaffen

- Man sollte die Eingaben vor deren Weiterverarbeitung auf Typ und Plausibilität hin überprüfen. Entsprechen sie dem erwarteten Datentyp? Liegen sie im erlaubten Wertebereich? Handelt es sich um eine der vorgegebenen Möglichkeiten?
- Falls man die Eingabe von Zahlenwerten erwartet, sollte man die Werte vor der weiteren Verarbeitung mithilfe der beiden Funktionen `intval()` und `doubleval()` in ganze Zahlen beziehungsweise Zahlen mit Nachkommastellen umwandeln. Auf diese Weise können auch Fehleingaben abgefangen werden.
- Eingaben sollte man mithilfe der Funktion `htmlspecialchars()` oder der Funktion `htmlentitites()` vor der weiteren Verarbeitung umwandeln, sodass kein schädlicher HTML-Code oder JavaScript-Code eingebettet werden kann. Die Funktion wandelt zum Beispiel die Zeichen `<` (Beginn eines HTML-Tags) in die entsprechenden THML-Entitites um (`<`).
- Man sollte prüfen, woher die Daten kommen handelt es sich um die erwartete Quelle?
- Bei den übermittelten Werten sollte es sich nicht um Funktionsnamen oder Dateinamen handeln, die anschließen direkt aufgerufen oder anderweitig bearbeitet werden. Dies spart zwar Code, setzt aber die Sicherheit herab.
- Falls man Eingaben benötigt, um anschließen eine Datenbankabfrage zu generieren, sollte man mit der Funktion `mysqli_real_escape_strig()` arbeiten. Diese setzt unter anderem jedes Hochkomma in der Eingabe in die Kombination Backslash und Hochkomma um. Dadurch wird eine sogenannte SQL Injection erschwert. Bei einer SQL Injection handelt es sich um das Einbetten von zusätzlichem, eventuell schädlichem SQL-Code.

PASSWÖRTER

Häufig kann der Zugang zu bestimmten Seiten nur mit einem Passwort erlangt werden. Die Sicherheit lässt sich hierbei durch die folgenden Maßnahmen erhöhen.

- Passwörter sollten eine bestimmte MIndeslänge haben. Sie sollten sowohl kleine Buchstaben als auch große Buchstaben sowie Ziffern und gegebenenfalls auch bestimmte Sonderzeichen beinhalten
- Diese Regel sollten in beiden möglichen Fällen beachtet werden:
  - Falls der Benutzer sein Passwort selbst wählen kann
  - Falls der Betreiber des Webservers ein Passwort vorgibt, das von einem PHP-Programm generiert wird
- Die Speicherung des Passworts in einer Datenbank sollte in verschlüsselter Form erfolgen, zum Beispiel mithilfe der Funktion `password_hash()`

QUELLE

Quelle: Einstieg in PHP 7 und MySQL

Rheinwerk Computing
ISBN 978-3-8362-6312-2