# Tutorial 9: Formulare mit PHP vor Spam schützen

Wenn auch Sie das Problem haben, dass Sie Werbemails über Ihr Kontaktformular erhalten oder immer wieder Spam-Einträge in Ihren Foren, Gästebüchern oder Blogs finden, haben Sie vielleicht auch schon mal darüber nachgedacht, wie sich das verhindern lässt. Eine gängige Praxis sind so genannte grafische Captchas, also die verzerrten Buchstaben auf unruhigem Hintergrund, die man mühsam entziffern und in ein Feld eintippen muss. Wie Sie diese lästige Methode den Besuchern Ihrer Homepage ersparen und trotzdem ein spam-sicheres Formular ganz einfach selbst realisieren können, können Sie in dieser Anleitung anhand zweier Beispiele, die sich auch sehr gut erweitern und individuell anpassen lassen, nachlesen. Für die hier aufgeführten Beispiele benötigen Sie die Skriptsprache PHP, die in Ihrem Homepage-Produkt ab der Homepage Basic inklusive ist.

**Wozu werden Captchas eingesetzt?**

Captcha ist ein Akronym für "Completely Automated Public Turing test to tell Computers and Humans Apart". Ein Turing-Test, benannt nach Alan Turing, versucht die Frage zu beantworten, ob Maschinen denken können. Bei einem Turing-Test soll ein Mensch entscheiden, ob sein Gegenüber ein Rechner oder ein anderer Mensch ist. Auch Captchas werden verwendet, um zu entscheiden, ob das Gegenüber ein Mensch oder eine Maschine ist, wobei bei Captchas diese Unterscheidung von einem Computer gemacht werden soll.

Das Prinzip ist relativ einfach: Dem Besucher einer Webseite wird eine Aufgabe gestellt, die – theoretisch – von menschlichen Besuchern leicht gelöst werden kann, von Computern, z. B. Spam-Bots, hingegen praktisch nicht oder nur unter unverhältnismäßig großem Rechenaufwand. In der Praxis handelt es sich meist um eine verzerrt dargestellte Zeichenfolge, die gängige Muster-erkennungsalgorithmen überfordert. Damit soll verhindert werden, dass Web-Formulare (z. B. Registrierungen, Gästebücher oder Foren) automatisch ausgefüllt und abgesendet werden können. Leider schließen bildbasierte Captchas aber sehbehinderte Menschen von vornherein aus und auch Menschen ohne Sehschwäche können die oft stark verzerrten Zeichen nicht mehr eindeutig identifizieren. Andererseits wurden bereits Lösungen entwickelt, mit denen bildbasierte Captchas von Spam-Bots umgangen werden können.

Es ist aber nach wie vor wichtig, Web-Formulare vor Spam zu schützen, nicht zuletzt aus rechtlichen Gründen, da Sie als Homepage-Besitzer z. B. für die Inhalte in Ihren Foren, Blogs oder Gästebüchern haften. Deshalb kann es interessant sein, Alternativen zu bildbasierten Captchas einzusetzen.

**Beispiel 1: Unsichtbare Eingabefelder, die nur von Maschinen ausgefüllt werden**

Spam-Bots sind so programmiert, dass sie erkennen können, in welchen Feldern welche Eingaben erwartet werden und diese passend befüllen können. Zum Beispiel wird ein Feld mit dem Namen "email" mit ziemlicher Sicherheit mit einer formal gültigen E-Mail-Adresse befüllt.

Wenn ein solches Eingabefeld, das menschliche Besucher nicht sehen können und das auch von Braille-Readern nicht wiedergegeben wird, mit Daten befüllt wird, handelt es sich mit ziemlicher Sicherheit um Spam. Bei der Programmierung eines solchen Captcha-Feldes gibt es ein paar Dinge zu beachten. Hier ist ein Code-Beispiel:

Zunächst muss im Stylesheet eine Klasse definiert werden, die die Anzeige unterdrückt.

```
.highlight { display:none; }
```

Im HTML-Code des Formulars wird dann ein Eingabefeld ergänzt, das Spam-Robots mit Sicherheit ausfüllen würden. Deshalb sollte der Name des Eingabefeldes E-Mail, Homepage, URL oder ähnliches suggerieren. Da Spam-Robots i. d. R. versuchen, URLs in Webformularen zu publizieren und E-Mail-Adressen in den meisten Formularen Pflichtfelder sind, können Sie davon ausgehen, dass Felder mit solchen Namen vom Bot mit Daten belegt werden.

```
<div class="highlight">
  <label for="email">Ihre E-Mail-Adresse wird hier nicht abgefragt, tragen Sie hier bitte NICHTS
ein:</label>
  <input id="email" name="email" size="60" value="" />
</div>
```

1. Es genügt nicht, das Eingabefeld direkt auf 'hidden' zu setzen, da das von Spam-Bots leicht erkannt wird. Statt dessen sollten Sie einen ganzen Abschnitt über CSS unsichtbar setzen. Hier erfolgt das über die Zuweisung der Klasse "highlight" in einem umgebenden DIV-Tag. Es empfiehlt sich, die Klasse möglichst unverfänglich zu benennen.
2. Das unsichtbare Eingabefeld sollte im Quelltext nicht von den anderen Eingabefeldern abgegrenzt werden. Am besten setzen Sie es mitten rein.
3. Das eigentliche Eingabefeld für eine E-Mail-Adresse des Besuchers – so vorhanden – muss natürlich einen anderen Namen erhalten. Dabei sollte aus der Bezeichnung möglichst nicht hervorgehen, dass hier eine E-Mail-Adresse erwartet wird.
4. Im Label-Tag können Sie Text eingeben, der menschlichen Besuchern, die kein CSS einsetzen, darüber informiert, dass sie das Feld leer lassen sollen.

Nachdem ein Spam-Robot die Daten Ihres Web-Formulars inkl. des versteckten Feldes ausgefüllt hat, werden alle Formulardaten zunächst an den Webserver gesendet. Dort werden Formulardaten i. d. R. per PHP ausgewertet. Sie können dieses PHP-Skript nun dahingehend ergänzen, dass Datensätze, bei denen die Variable "email" einen Wert enthält, gesondert behandelt werden. In einer if-Abfrage prüfen Sie zunächst, ob "email" einen Wert enthält, und zwar mit der Bedingung isset();, der true oder false zurückgibt.

```
<?php
  if (isset($_POST['email'])) {
   echo "Ihre Beitrag wurde als Spam identifiziert. Bitte überprüfen Sie Ihre Eingaben.";
   die ();
  }
else {
  // Hier beginnt die Verarbeitung der Daten aus dem von einem Menschen versendeten Formular
}
?>
```

Im Anweisungsteil nach der if-Abfrage, der in geschweiften Klammern steht, können Sie dann festlegen, wie mit Datensätzen umgegangen werden soll, bei denen es sich vermutlich um Spam handelt, da dieser nur ausgeführt wird, wenn das versteckte Formularfeld abgeschickt wurde. Hier haben wir uns dazu entschieden, zunächst einen Text ausgeben zu lassen, für den Fall, dass der Beitrag doch von einem menschlichen User abgesendet wurde. Dieser erhält dann die Chance, seine Daten nochmals abzusenden und dabei den im Label-Tag gegebenen Tipp zu beachten. Mit dem Befehl die(); wird die Verarbeitung des Datensatzes dann abgebrochen.

**Beispiel 2: Dynamisch generierte Variable mit übergeben**

Das zweite Beispiel macht sich die Arbeitsweise der meisten Spam-Bots zunutze. Diese interpretieren ein Formular einmal und rufen dann die im Form-Tag im Attribut "action" angegebene URL – dabei handelt es sich um das Skript, das die Formulareingaben entgegennimmt und verarbeitet – direkt mit dem Befehl POST auf. Der Unterschied zum menschlichen Benutzer ist der, dass die eigentlich Formularseite nicht jedes Mal neu generiert und interpretiert wird.

Wenn Sie dem Formular ein verstecktes Eingabefeld hinzufügen, das einen dynamisch generierten Namen enthält, können Sie bei der Auswertung der Eingaben prüfen, ob der Variablenname kurz vorher generiert wurde. Hierzu ist es lediglich erforderlich, dass das aktuelle Datum sowie ggf. die Uhrzeit Bestandteil des Variablennamens ist.

```
<?php
function captcha(){
$zahl = 1455;
$no_spam = (date("dmy", time())) x $zahl;
return $no_spam;
}
echo '<input type="hidden" name="sp-'.captcha().'" value="1" />';
?>
```

Die Berechnung des Feldes kann natürlich auch anderes erfolgen. Im obigen Beispiel wird der Variablen "no_spam" ein Wert zugewiesen, der das aktuelle Datum enthält. Dieses wird zunächst mit der Funktion time(); ermittelt, die den Unix-Timestamp ausliest, und dann mit der Funktion date(); in einen String umgewandelt. Da dabei nur Tag, Monat und Jahr berücksichtigt werden, ist die Variable flexibel genug, um die Zeitdifferenz, die zwischen dem Aufrufen einer Formularseite, deren Ausfüllen und Absenden durch einen menschlichen Besucher, zu berücksichtigen.

Bei der Auswertung der ankommenden Daten wird Spam dann folgendermaßen ermittelt:

```
<?php

if (isset ($_POST['sp-'.captcha()])) {
//Kein Spam, es folgen die Befehle zur normalen Formularverarbeitung
}

else {
// Spam, es folgen die Befehle zum Umgang mit Spam; das kann statt eines simplen die (); z. B.
// auch die gesonderte Speicherung in einer eigenen Datenbanktabelle sein.
}

?>
```

Wichtig hierbei ist, dass das auswertende Skript die Funktion captcha(); nochmals durchführt. Um den Code nicht unnötigerweise in 2 verschiednen PHP-Skripten zu wiederholen, bietet es sich an, die Funktion in eine eigene Datei auszulagern und per include-Befehl einzubinden:

```
<?php

include 'name_der_datei.php';

?>
```

Andererseits kann es sinnvoll sein, die Funktion im auswertenden Skript nicht einfach zu wiederholen, sondern statt dessen eine Variante anzuwenden. Bei der oben dargestellten Methode kann es nämlich passieren, dass ein menschlicher Benutzer, der das Formular kurz vor Mitternacht (GMT, da der Unix-Timestamp verwendet wird) aufgerufen hat, aber erst nach Mitternacht (GMT) absendet, fälschlicherweise als Spammer identifiziert wird. Um das zu vermeiden, können Sie in die Prüfung der Bedingung noch einen Puffer von z. B. einer Stunde einbauen.

```
<?php

$zahl = 1455;

function captcha(){
$no_spam = (date("dmy", time())) x $zahl;
return $no_spam;
}

function captcha2(){
if (date('H', time())=='00') {
$no_spam2 = (date("dmy", time()-86400)) x $zahl;
return $no_spam2;
}

if (isset ($_POST['sp-'.captcha()]) || isset ($_POST['sp-'.captcha2()])) {
//Kein Spam, es folgen die Befehle zur normalen Formularverarbeitung
}
else {
// Spam, es folgen die Befehle zum Umgang mit Spam; das kann statt eines simplen die (); z. B.
// auch die gesonderte Speicherung in einer eigenen Datenbanktabelle sein.
}

?>
```

Natürlich empfiehlt es sich, die o. a. Beispiele nicht einfach eins zu eins zu übernehmen, sondern kreativ abzuwandeln und ggf. zu erweitern. Falls bestimmte Skripte zu häufig eingesetzt werden, würden Spam-Bots darauf reagieren und sie auszuhebeln versuchen.

Eine Variante von Beispiel 2 wäre, die Uhrzeit (date-Funktion mit Parameter "ymdHis" statt "dmy" aufrufen) mit in die Variable zu kodieren. Dann genügt es allerdings nicht mehr, einfach die Variablennamen direkt miteinander zu vergleichen. Sie müssten sie erst wieder in den Wert des ursprünglichen Unix-Timestamp zurückführen.

```
<?php

$rest=substr($_POST['sp-'.captcha()], 3);

?>
```

Mit der Funktion substr(); werden erst einmal die Buchstaben, mit denen eine Variable beginnen muss, entfernt. Obwohl die Variable $rest theoretisch einen String, also eine Zeichenkette enthält, kann man in PHP einfach mit ihr weiterrechnen. Die Umwandlung in eine Ganzzahlvariable (Integer) erfolgt dabei automatisch.

Führen Sie anschließend einfach alle Rechenschritte in umgekehrter Reihenfolge durch. Dadurch erhalten Sie den Unix-Timestamp des Zeitpunktes, an dem der Besucher die Formularseite aufgerufen hat. Danach ermitteln Sie den aktuellen Unix-Timestamp in einer neuen Variable. Mit der Differenz dieser beiden Werte können Sie eine if-Abfrage formulieren, in der ein realistischer Zeitraum ermittelt wird, den ein menschlicher Besucher zum Ausfüllen des Formulars benötigt haben könnte. Z. B. zwischen 5 Sekunden und einem Tag + 1 Stunde. Mit dem Maximalwert hätten Sie auch gleich wieder einen Puffer für Datensätze eingebaut, die kurz nach Mitternacht abgesendet wurden, während Sie mit dem Minimalwert sehr effektiv Spam-Bots ausschalten, denn alles unter 5 Sekunden ist bei einem Formular, das Name, Mailadresse und in der Regel noch eine Nachricht beinhaltet, sehr verdächtig und kann als SPAM abgefangen werden.

Mit dieser Abwandlung des zweiten Beispiels können Sie auch Bots ausschließen, die die Daten über die Formularseite abschicken und diese jedes Mal neu "lesen". Denn auch das geschieht in der Regel so schnell, dass das in derselben Zeit unmöglich ein Mensch geschafft haben kann.

**Zum Nachlesen und Nachschlagen: Ressourcen im Internet**

- Auf der offiziellen PHP-Homepage http://www.php.net finden Sie ausführliche Informationen rund um das Thema PHP, unter anderem auch eine komplette deutsche PHP-Referenz.
- Ein ausführliches Tutorial zu PHP finden Sie auch unter http://www.php-quake.net/
- Die deutschsprachige Newsgroup de.comp.lang.php liefert erfahrenen PHP-Anwendern die Möglichkeit, Ideen auszutauschen. Die offizielle FAQ zu dieser Newsgroup findet sich unter http://www.dclp-faq.de/, wo man die Antworten zu häufig gestellten Fragen, aber auch eine ganze Menge nützlicher Tipps und Tricks findet. Eine ebenfalls deutschsprachige Einführung zu PHP ist dort unter anderem auch vorhanden.