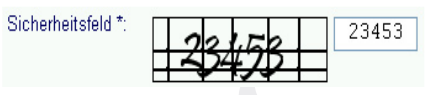
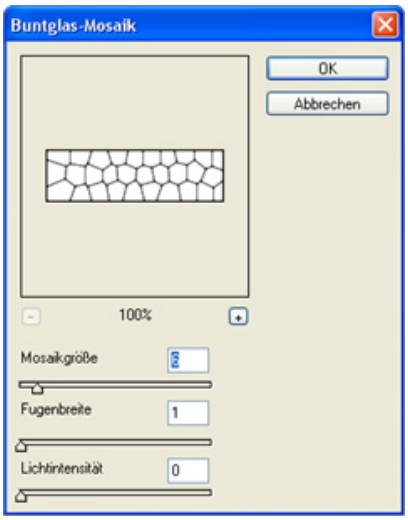
Sicherheitsfeld mit PHP, GDlib und FreeType

Inhalte und Informationen sind entscheidend für den Erfolg einer Webseite. Umso ärgerlicher ist es, wenn diese Inhalte durch direktes Verlinken auf Fremdseiten angeboten werden. Auch bei Formularen kann eine Personenvalidierung interessant sein. Wie man eine reale Person erkennt, zeigen wir mit diesem vielseitigen, kleinen Script.

Jeder Webseitenbetreiber ärgert sich, wenn die mühsam erstellten, oder bei Agenturen gekauften, Inhalte durch direkte Verlinkung unerlaubt auf Fremdseiten erscheinen und Besucher vom eigenen Angebot abziehen. Da darunter nicht nur die Besucherzahlen leiden, sondern zudem auch das eigene Trafficaufkommen steigt, sollte man diesem Verhalten möglichst einen Riegel vorschieben. Auch bei Onlineformularen ist eine Validierung wünschenswert, um ausschließen zu können, dass es sich beim Ausfüllenden nicht um ein automatisiertes Script, sondern um eine reale Person handelt. Dies ist insbesondere bei Anmeldungs-, Registrierungs- und Mailformularen interessant. Die Validierung

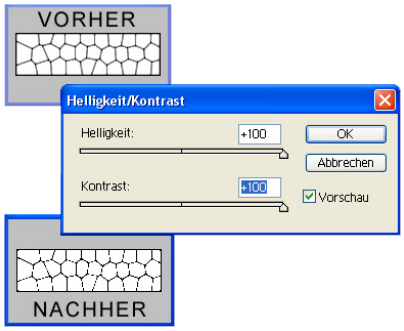


sollte dabei möglichst ohne allzu große Probleme zu verstehen sein, um den Besucher nicht unnötig zu verunsichern. Im ersten Teil dieser Artikelreihe zeigen wir Ihnen, wie Sie eine solche Validierung mithilfe von dynamisch mit PHP erzeugten Grafiken umsetzen können, deren Inhalt dann vom Besucher in ein Textfeld eingegeben werden muss. Um zu verhindern, dass texterkennende Scripte (OCR-Engines) dennoch in der Lage sind dieses Grafikfeld zu „lesen“, benötigen wir als erstes eine Hintergrundgrafik. In unserer Variante wird dieses Hintergrundbild mit einem Bildbearbeitungsprogramm erstellt. Wir nutzen dazu Adobe Photoshop. Der Vorgang lässt sich Problemlos auf andere Grafiksoftware übertragen.



Erstellen der Feldgrafik

Um genug Platz für die dynamisch generierten Buchstaben und Ziffern zu haben, erstellen wir eine neue Bilddatei mit einer Breite von 140px und einer Höhe von 40px. Der Hintergrund der Datei ist weiß. Nun gilt es ein Muster zu erstellen, das die Effizienz von OCR-Engines beeinträchtigt, die Lesbarkeit für das menschliche Auge aber nur geringfügig verschlechtert. Dazu wählen wir, mit einer schwarzen Vordergrundfarbe, den Strukturierungsfilter „Buntglas-Mosaik“. Wir haben für dieses Beispiel die Werte: Mosiakgrösse:6, Fugenbreite:1 und Lichtintensität:0 gewählt. Natürlich können Sie auch einen der anderen Strukturierungsfilter einsetzen. Das resultierende Bild wird nun noch per Kontrast- und Helligkeitsänderung um je +100 modifiziert, und als zweifarbige, indizierte PNG-Datei abgespeichert. In unserem Fall geben wir der Datei den Namen „security_background.png“. Nun haben wir die Grundlage, um im nächsten Schritt die dynamisch generier-



ten Inhalte einfügen zu können. Was noch fehlt ist eine True-Type Schriftart, mit der der Text gezeichnet und über das Hintergrundbild gelegt wird. Wir nutzen hier die Schriftart ‚pristina.ttf'. Sie liefert auch bei den wildesten Hintergrundmustern noch einigermaßen lesbare Ergebnisse. Sie können aber auch jede andere Schriftart nutzen. Achten Sie beim Einsatz von kommerziellen Schriftarten aber auf eventuell einschränkende Lizenz- und Nutzungsbedingungen.

Listing: script_v1.php

```
<?php
//headerinfo für PNG-Bilder
header('Content-type: image/png');

//Bereitstellen des Image-Objektes
$img = ImageCreateFromPNG('security_background.png');

//Textfeld-Wert
$text= '12345';

//Festlegen der Schriftfarbe
$color = ImageColorAllocate($img, 0, 0, 0);

//Schrifttyp-Einstellungen
$ttf = 'pristina.ttf';
$ttfsize = 30;
$angle = 0;
$text_x = 30;
$text_y = 30;

//Einfügen des Textes ins Image-Objekt
imagettftext($img, $ttfsize, $angle, $text_x, $text_y, $color,
$ttf, $text);

//Ausgabe und löschen des Image-Objektes
imagepng($img);
imagedestroy($img);

?>
```

Dynamische Bildinhalte

Das folgende Script benötigt als Grundvorraussetzungen neben einem PHP-fähigen Server auch die GDlib sowie eine funktionsfähige FreeType Umgebung. Um zu dem bereits erstellten Hintergrundbild dynamische Texte hinzuzufügen erledigt das Script im groben folgende Aufgaben:

- Hintergrundbild wird zu einem Bildobjekt
- Schrifttyp-Eigenschaften werden festgelegt
- Hinzufügen von Text zum Bildobjekt
- Rückgabe des Bildobjektes als PNG-Bild

Eine mögliche Umsetzung der oben genannten Schritte finden Sie im Listing: script_v1.php. Das Script ist kommentiert und sollte somit keine Probleme bereiten.

Der erste Aufruf

Wenn Sie alles richtig gemacht haben, sollte der Aufruf dieses Scriptes folgende Ausgabe generieren:

Wenn Sie keine Ausgabe erhalten, prüfen Sie bitte, ob sich die Schriftarten-Datei ‚pristina.ttf' sowie das Hintergrundbild ‚security_background.png' im selben Verzeichniss befindet wie das Script. Auch muss die Schreibweise der Dateinamen im Script denen auf dem Server entsprechen.

Und nun etwas Abwechslung

Bevor wir uns um den Einsatz des Scriptes in Ihre Formulare kümmern, müssen wir noch diverse Kleinigkeiten abändern. Im Moment generiert das Script immer das gleiche Bild. Für einen potentiellen Übeltäter wäre es nun ein leichtes die Ziffernfolge von Hand in ein Script zu übertragen und so die von uns angestrebte Prüfung zu umgehen. Um dem vorzubeugen erstellen wir am Anfang des Scriptes ein Array, das verschiedene Werte enthält. Aus diesen Werten wird dann einer ausgewählt um das Textfeld zu erstellen.

```
//Vergleichs- bzw. Inhalts-
werte       $werte = Array(
'44323' , '89932' , '90323'
);
```

Damit diese Werte auch in das Textfeld gelangen können, müssen wir eine Zuweisung der Array-Inhalte zur Variablen $text vornehmen. Die neue Zuweisung hat die folgende Form:

```
//Textfeld-Wert
$text= $werte[rand(0,count($w
erte)-1)];
```

Da wir gerade dabei sind, können wir es einer OCR-Engine noch ein wenig schwerer machen indem wir den Winkel ($angle) und die horizontale Position des Textfeldes ($text_x) zufällig ermitteln:

```
//Schrifttyp-Einstellungen
$ttf = 'pristina.ttf';
$ttfsize = 30;
$angle = rand(0,5);
$text_x = rand(5,50);
$text_y = 30;
```

Dabei ist zu beachten, dass diese Werte auf der Schriftart „pristina.ttf“ mit der Schriftgrösse 30px basieren und ein Hintergrundbild mit den Maßen 140px mal 40px benötigen. Bei anderen Schriften und/oder Bildmaßen sind diese Werte evtl. nicht optimal. Das Script finden Sie in der geänderten Form unter dem Namen „script_v2.php“ bei den zum Artikel gehörenden Ressourcen. Es generiert nicht nur verschiedene Textfeld-Inhalte, sondern dreht und verschiebt das Textfeld anhand (mehr oder weniger) zufälliger Werte.

Ergänzung einer Prüffunktion

Damit das Script eine sinnvolle Aufgaben erfüllen kann, erweitern wir den Funktionsumfang um eine Prüffunktion. Wir legen anhand der per $_GET übergebenen Variable „$doit“ fest, wie sich das Script beim Aufruf verhalten soll. „$doit“ kann dabei die Werte ‚gen‘ oder ‚check‘ annehmen.

Wir haben zwei Optionen mit denen man die Ausgabe des Scriptes beeinflussen kann. Bei einem Aufruf von „script_v3.php?doit=check“ erfolgt z.Zt. die Ausgabe: „Eingabe wird geprüft ...“. Bei einem Aufruf von „script_v3.php?doit=gen“ generiert das Script unser dynamisches Bild. Dieses wird dann vom benutzten Browser angezeigt wird. Natürlich ist die Check-Funktion im Moment noch vollkommen nutzlos. Es steht bisher weder fest, was geprüft werden soll, noch wie eine Reaktion aussehen könnte. Da wir uns ja bereits am Anfang entschlossen haben, die zu einem Vergleich benötigte Eingabe über ein normales Formularfeld zu erfragen, erweitern wir die Prüffunktion aus „script_v3.php“ um eine weitere Bedingung:

Auszug aus: script_v4.php

```
//Wenn das Script die Einga-
be Checken soll
if($_GET['doit'] == 'check'){
  if(in_array($_POST['sec_
    eingabe'],$werte)){
    print('Eingabe war richtig...
    !!');
  }
  else{
    print('Eingabe war falsch...
    !!');
  }
}
```

Im Klartext bedeutet der Code-Teil:

Hole aus dem per Formular mit method=“post“ übergebenen Array: $_POST, den Inhalt vom Index: ‚sec_eingabe‘ und prüfe ob dieser

Listing: script_v3.php

```
<?php

//Vergleichs- bzw. Inhaltwerte
$werte = Array('44323','89932','90323');

//Wenn das Script die Eingabe Checken soll
if($_GET['doit'] == 'check'){
      print('Eingabe wird geprüft...');
}


//Wenn das Script ein Bild generieren soll
elseif ($_GET['doit'] == 'gen'){
      header('Content-type: image/png');
      $img = ImageCreateFromPNG('security_background.png');
      $text= $werte[rand(0,count($werte)-1)];
      $color = ImageColorAllocate($img,0,0,0);
      $ttf = 'pristina.ttf';
      $ttfsize = 30;
      $angle = rand(0,5);
      $text_x = rand(5,50);
      $text_y = 30;
      imagettftext($img,$ttfsize,$angle,$text_x,$text_y,
            $color, $ttf, $text);
      imagepng($img);
      imagedestroy($img);
}

?>
```

Inhalt im $werte-Array vorhanden ist. Wenn dem so ist, hat der Nutzer einen der möglichen Werte eingegeben. Ansonsten war die Eingabe falsch.

Nun das Formular

Wir verfügen nun über die Möglichkeit ein Formular durch eine Kombination aus dynamisch generiertem Bild und Textfeld vor automatisierter Nutzung durch Fremdscripte zu schützen. Was noch fehlt ist ein entsprechendes Formular, das unser Script einbindet. Zur Veranschaulichung haben wir uns für ein kleines Kontaktformular entschieden. Im Formular wird die Scriptdatei „script_v5.php“ genutzt. Diese Variante enthält eine Funktion zum Versand der Formulardaten als Mail. Alternativ können Sie hier Ihre eigene Prüfroutine einfügen.

Wie Sie sehen, ist der Einsatz des Scriptes recht einfach. Im <form>-Tag wird die prüfende Funktion unseres Scriptes eingesetzt. Das dazugehörige Bild generieren wir, indem wir im <img>-Tag das Script als Bildquelle einsetzen. Das dazugehörige Textfeld ‚sec_eingabe‘ enthält den vom Nutzer eingegebenen Prüfwert und wird bei der Übermittlung der Formulardaten ebenfalls weitergereicht.

Weitere Überlegungen

Jetzt haben Sie die Möglichkeit, das automatisierte Ausfüllen Ihrer Formulare zu unterbinden. Sie sollten bedenken, dass der alleinige Einsatz dieser Technik nicht immer sinnvoll ist. So schließen Sie z.B. sehbehinderte Menschen von der Nutzung Ihres Online-Angebotes aus, wenn Sie nicht alternative Validierungsmöglichkeiten anbieten.

Listing: mailform.html

```
<html>
<head><title>Mailformular</title></head>
<body>
<form action='script_v5.php?doit=check' method='post'>
<br />Name:
<input type='text' name='name' size='15' />
<br />Betreff:
<input type='text' name='betreff' size='15' />
<br />Memo:
<input type='text' name='memo' size='15' />
<br />
<img src='script_v5.php?doit=gen' />
<input type='text' name='sec_eingabe' size='5' />
<br />
<input type='submit' />
</form>
</body>
</html>
```

Ein weiterer Artikel wird eine Verbesserung der Sicherheit durch sich in unregelmäßigen Abständen ändernde Vergleichswerte beschreiben. Dort werden wir zudem erklären, wie sich im Script mehrere Hintergrundbilder einsetzen lassen, um der einfachen Bildung von Differenzbildern entgegenzutreten. Der Artikel wie sich das Script, nach kleineren Anpassungen, auch als Schutz vor scriptgenerierten Downloads einsetzen lässt, wird dann das Thema Personenvalidierung abschliessen. Auf unserer Internetseite www.contentcharge.de finden Sie neben den im Artikel erstellten Scripten weitere Infos rund um Webdesign und Media.

Christian Lehmann

Listing: script_v5.php

```
<?php

//Vergleichs- bzw. Inhaltwerte
$werte = Array('44323','89932','90323');

//Wenn das Script die Eingabe Checken soll
if($_GET['doit'] == 'check'){

    if(in_array($_POST['sec_eingabe'],$werte)){
        //datum der Sendung
        $datum = date('d.m.Y - H:i');

        //Hier bitte die Empfänger-Mailadresse einfuegen
        $empfaenger = 'DeineMail@irgendwo.de';

        //Zeilenende ersetzen
        $suchen = '\r';
        $ersetzen = '\n';
        $kommentarp = str_replace($suchen,$ersetzen,$_POST['memo']);
        $betreff = $_POST['betreff'];

        //Ausführen der Mailfunktion
        if(mail(„$empfaenger“, „$betreff“,
        „
        Name: „.$_POST['name'].“
        Memo: „.$kommentarp.“
        Datum und Zeit: $datum „)){
        print('Versand war erfolgreich');
        }
     }
    else {
        print('BITTE PRÜFEN SIE IHRE EINGABE IM SICHERHEITSFELD');
    }
}

//Wenn das Script ein Bild generieren soll
elseif ($_GET['doit'] == 'gen'){
    header('Content-type: image/png');
    $img = ImageCreateFromPNG('security_background.png');
    $text= $werte[rand(0,count($werte)-1)];
    $color = ImageColorAllocate($img, 0, 0, 0);
    $ttf = 'pristina.ttf';
    $ttfsize = 30;
    $angle = rand(0,5);
    $text_x = rand(5,50);
    $text_y = 30;
    imagettftext($img, $ttfsize, $angle, $text_x, $text_y, $color, $ttf, $text);
    imagepng($img);
    imagedestroy($img);
}

?>
```

Listing: mailform.html

```
<html>
<head>
<title>Mailformular</title>
</head>
<body>
<form action='script_v5.php?doit=check' method='post'>
<br />Name:
<input type='text' name='name' size='15' />
<br />Betreff:
<input type='text' name='betreff' size='15' />
<br />Memo:
<input type='text' name='memo' size='15' />
<br />
<img src='script_v5.php?doit=gen' />
<input type='text' name='sec_eingabe' size='5' />
<br />
<input type='submit' />
</form>
</body>
</html>
```