Spamschutz für ein Kontaktformular mit CSS, HTML, JS und PHP

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

Tabellenverzeichnis

1 Problemstellung

In diesem Beitrag wird beschrieben, wie ein Kontaktformular mit einfachen Mitteln gegen Schadsoftware (engl. *spambots, spider, crawler*, etc.) geschützt werden kann. Zunächst wird das betreffende Kontaktformular präsentiert. Danach werden einfache Schutzmaßnahmen beschrieben, wie es vor Eingabefehlern und/oder Missbrauch gesichert werden kann. Dabei kommen das sog. "Honigtopf" (engl. *honeypot*)-Verfahren und das Messen der Eingabegeschwindigkeit zum Einsatz.

Zum Verständnis werden Grundkenntnisse der in der Überschrift genannten Programmiersprachen vorausgesetzt.[1]

2 Kontaktformular

2.1 Darstellung

Als Beispiel dient ein Kontaktformular, das am Ende meiner Homepage *http://www.dr-thormaehlen.de* eingebaut ist, um die Kommunikation mit dem jeweiligen Besucher zu ermöglichen.

Abbildung 1: Erscheinungsbild des Kontaktformulars

2.2 Formularfelder

Das Kontaktformular umfasst 2 unsichtbare und 7 sichtbare Formularfelder. Tabelle 1 enthält die Einzelheiten dieser 9 Formularfelder (s. Tabelle 1):

[1] **CSS** steht ***C**ascading **S**tyle **S**heets* und ist eine Gestaltungs- und Formatierungssprache. **HTML** steht für ***H**yper-**T**ext **M**arkup **L**anguage* und ist eine Auszeichnungssprache für Webseiten. **JS** steht für *JavaScript* und ist eine clientseitige Skriptsprache für Webanwendungen. **PHP** steht für ***P**ersonal **H**ome **P**age* ist eine serverseitige Skriptsprache für Webanwendungen.

Bezeichnung	HTML Name-Attribut	Formularfeld	HTML-Feldtyp	Sichtbarkeit
keine	url	einzeiliges Textfeld	text	nein
keine	frm_load_tm	einzeiliges Textfeld	hidden	nein
Anrede	salutation	Knöpfe	radio	ja
Name	fullname	einzeiliges Textfeld	text	ja
E-Mail	email	einzeiliges Textfeld	email	ja
Betreff	subject	einzeilige Textfeld	text	ja
Nachricht	message	mehrzeiliges Textfeld	textarea	ja
absenden	submit	Schaltfläche	button	ja
zurücksetzen	reset	Schaltfläche	button	ja

Tabelle 1: Eigenschaft der 9 Felder des Kontaktformular

Die beiden unsichtbaren Formularfelder (s. Tabelle 1) werden zum Schutz des Kontaktformulars vor Spam benutzt.

2.3 Struktur

Ein Kontaktformular wird im Body einer HTML-Datei zwischen den HTML-Tags **<form>** … **</form>** definiert. Das **<form>**-Element selbst besitzt u. a. die Attribute **onsubmit** und **action** (s. Listing 1, zweite Zeile):

- Das Ereignis **onsubmit**" wird zuerst ausgeführt und löst die Plausibilitätsprüfung der *sichtbaren* Formularfelder mit *JavaScript* aus. Ist das Ergebnis negativ, werden entsprechende Fehlermeldungen in einem separaten Popup-Fenster ausgegeben.
- Ist das Ergebnis der Überprüfung positiv, wird das Formular angewiesen, das Skript *kontakt.php* auszuführen, das im Attribut **action** spezifiziert ist. Damit werden *alle* Formulareingaben rigoros ausgewertet. Ist das Ergebnis negativ, wird der Inhalt des Formulars nicht abgesandt.

In Listing 1 wird lediglich die tabellarische Struktur der beiden unsichtbaren Formularfelder gezeigt:

- Das Formularfeld mit dem Namen **url** dient zum Anlegen einer sog. Honigtopf-Falle. Die Unsichtbarkeit wird durch die Zuweisung der CSS-Klasse „**antispam**“ (d.h. *display:none*) erreicht.
- Das Formularfeld mit dem Namen **frm-load_tm** besitzt ein **hidden**-Attribut**.** Es verhindert die Anzeige des Formularfelds. Der zugehörige Feldwert wird durch das **value**-Attribut festgelegt. Es enthält hier den Ladezeitpunkt ($_SERVER['REQUEST_TIME']) des Kontaktformulars.

```
<form name="myForm" method="post"
      onsubmit="return checkForm(this);" action="kontakt.php" >
   <fieldset>
      <legend><b>Kontaktformular</b></legend>
      <table>
         <colgroup>
            <col width="60px">
            <col width="255px">
         </colgroup>
         <tr class="antispam">
            <td><label for="url">Nichts eingeben!:</label></td>
            <td><input type="text" name="url" size="30"</td>
         </tr><tr>
            <td><input type="hidden" name="frm_load_tm" autocomplete="off"
               value="<?php echo $_SERVER['REQUEST_TIME'];?>" />
            </td>
         </tr>
         … u.s.w.
      </table>
   </fieldset>
</form>
```

Listing 1: Die beiden unsichtbaren Formularfelder des HTML-Kontaktformulars

2.4 Formatierung

Die Gestaltung des Kontaktformulars erfolgt mit der Gestaltungs- und Formatierungssprache CSS[2]. Auf die Wiedergabe des Inhalts der betreffenden CSS-Datei wird verzichtet.

3 Spamschutz

3.1 Honigtopf

Ein Honigtopf bezeichnet eine einfache Falle, um *Spambots* zu erkennen. Es ist ein einzeiliges Formularfeld, das für einen menschlichen Benutzer nicht sichtbar ist, aber von Skripten, Bots, Robots oder anderer Schadsoftware gelesen und automatisch ausgefüllt werden kann. Da solche Programme dazu neigen, jedes verfügbare Formularfeld auszufüllen, kann nach dem Absenden des Kontaktformulars geprüft werden, ob das entsprechende Formularfeld weiterhin leer ist. Ist es aber gefüllt, ist zu befürchten, dass das betreffende Kontaktformular angegriffen wurde.

In der Datei **kontakt.php** (siehe Listing 1) zur Validierung des Inhalts des Kontaktformulars kann also durch die Abfrage

```
if(isset($_POST['url']) && $_POST['url'] == '') {…}
```

entschieden werden, ob das Formular sorglos an den Empfänger verschickt werden kann.

[2] Die Abkürzung *CSS* steht für **C**ascading **S**tyle **S**heets. Das heißt auf Deutsch: „gestufte Stilvorlagen“.

3.2 Messung der Eingabegeschwindigkeit

Die Grundidee ist, die Zeitdifferenz zwischen dem Laden eines Kontaktformulars *($loadTm*) und dessen Absendung *($currTm*) zu ermitteln. Ist diese Dauer relativ klein, kann vermutet werden, dass nicht ein menschlicher Benutzer, sondern ein Roboter (sog. spambot, spider, crawler o. search-engine) das Kontaktformular automatisch mit Inhalt gefüllt und abgesandt hat.

Die entsprechende Abfrage in der schon erwähnten Datei **kontakt.php** zur Validierung der Formulardaten lautet folglich (s. Listing 2):

```
if (isset($_POST['frm_load_tm']) && !empty($_POST['frm_load_tm'])) {
      $loadTm = intVal(sanitize($_POST["frm_load_tm"]));
      $currTm = intVal($_SERVER['REQUEST_TIME']);
   if (SpamBotFilling($loadTm,$currTm,2)) {
      die('Das Kontaktformur wurde vermutlich von einem sog. Spambot belegt.');
   }
}
```

Listing 2: Aufruf der Funktion SpamBotFilling mit den drei Argumenten $loadTm, $CurrTm und 2

```
function SpamBotFilling($loadTm,$requestTm,$fillTm) {
   return (($requestTm - $loadTm) < $fillTm) ? true : false;
}
```

Listing 3: PHP-Funktion zur Messung Eingabegeschwindigkeit eines Kontaktformulars

Wenn die PHP-Funktion *SpamBotFilling* (s. Listing 3) den Wert *true* zurückgibt, wurde das Kontaktformular in weniger als 2 Sekunden ausgefüllt, so dass eine unbemerkte und unerwünschte Belegung durch Schadsoftware anzunehmen ist.

3.3 Filterung

Vor dem Absenden des Kontaktformulars werden alle zugehörigen Eingabefelder in einer großen *foreach*-Schleife durchlaufen und mit 2 geschachtelten PHP-Funktionen überprüft (s. Listing 4):

- **remove_email_injection**
- **sanitize**

```
foreach($_POST as $key => $value) {
   $_POST[$key] = remove_email_injection(sanitize(trim($value)));
}
```

Listing 4: Große foreach-Schleife zur rigorosen Validierung des Inhalts aller Formularfelder

Die PHP-Funktion **remove_email_injection** (s. Listing 5) entfernt die Zeichen für Wagenrücklauf bzw. Zeilenumbruch aus einem Formularfeld. Diese Zeichen werden von Spammern gerne zum unbefugten Einfügen einer **CC**- oder **BCC**-Liste[3] verwendet:

```
function remove_email_injection($field){
// Nach Wagenrücklauf bzw. Zeilenumbruch suchen, der von Spammern gerne zum
   unbefugten Einfügen einer CC-/BCC-Liste verwendet wird.
   return preg_replace(array("/\r/", "/\n/", "/%0a/", "/%0d/"),"",$field);
}
```

Listing 5: Wagenrücklauf bzw. Zeilenumbruch durch Leerzeichen ersetzen

Die PHP-Funktion **sanitize** (s. Listing 6) entfernt u. a. missbräuchliche E-Mail Einfügungen (d. h. Angriffsmuster) aus einem Formularfeld:

```
function sanitize($data){
   // Entfernt HTML- und PHP-Tags und wandelt Sonderzeichen in HTML-Code um.
   $data = trim(strip_tags(htmlspecialchars($data)));
   // Entfernt Maskierungszeichen z. B. Backslashes vor Anführungszeichen.
   $data = stripslashes($data);
   // Entfernt unbefugte E-Mail Einfügungen (d. h. Angriffsmuster)
   $data = preg_replace("/(href|Content-Transfer-Encoding:|MIME-Version:|content-
   type:|Subject:|to:|cc:|bcc:|from:|reply-to:)/ims", "", $data);
   return $data;
}
```

Listing 6: Angriffsmuster in Formularfeldern erkennen und entfernen

3.4 Captchas

Captchas[4] werden sehr häufig verwendet um zu prüfen, ob ein Kontaktformular von einem Menschen oder missbräuchlich durch ein Computerprogramm ausgefüllt wurde[5]. Die oben beschriebene Lösung kommt ohne für Nutzer möglicherweise störende *Captchas* aus, vgl. dazu [1].

[3] **CC** steht für den engl. Begriff ***c**arbon **c**opy*. **BCC** bedeutet ***b**lind **c**arbon **c**opy*. Eine E-Mail kann zeitgleich an einen oder weitere Empfänger in Kopie übermittelt werden.
[4] Die engl. Abkürzung **captscha** steht für einen Test zur Unterscheidung zwischen Mensch und Maschine.
[5] Captchas sind eine der beliebtesten Anti-Spam-Maßnahmen für Kontaktformulare.

4 Plausibilitätsprüfung der sichtbaren Formularfelder

Mit dem Attribut **onsubmit="return checkForm(this);“** des HTML-Elements **<form>** wird die formale Überprüfung der sichtbaren Felder des Kontaktformulars mit *JavaScript* vorgegeben (siehe Listing 1 in Verbindung mit Listing 7).

```
// Rechtsklick verhindern
function click() {
   if (event.button==2) {alert('Diese Funktion ist stillgelegt!');}
   document.onmousedown=click;
};
// Cursor in das erste editierbare Feld des Kontaktformulars setzen
function setCursor(){
   if (document.forms.length > 0) {
      var field = document.forms[0];
      for (i = 0; i < field.length; i++) {
         if (field.elements[i].type == "text" || field.elements[i].type == "textarea") {
            document.forms[0].elements[i].focus();
            break;
         }
      }
   }
};
// Fokus auf die erste Option des Elements 'Anrede' setzen
function setFocus() {
   document.myForm.elements["salutation"][0].select();
   document.myForm.elements["salutation"][0].focus();
};
// Alle Felder auf Gültigkeit prüfen
function checkForm(frm) {
   var errMsg = strEmail = strName = "";
   // Der Name 'Prof. Dr. Hans-Peter O'Donnell' soll gültig sein!
   // alphaExp: Klein- und Großbuchstaben, Bindestrich, Hochkomma und Leerzeichen
   var alphaExp = /^[a-zA-Z-'. ]+$/;
   var txtMessage = "Ihre Nachricht...";
   if (!frm.elements["salutation"][0].checked &&
       !frm.elements["salutation"][1].checked) {
      errMsg += "Geben Sie bitte Ihre Anrede ein.\n";
      setFocus();
      markObj("salutation");
   }
   strName = frm.elements["fullname"]
   if (strName.value.length < 2) {
      errMsg += "Der Name ist zu kurz: min. 2 Zeichen!\n";
      frm.elements.fullname.focus();
      markObj("fullname");
   }
   // Namen wie 'Prof. Dr. Hans-Peter O'Donnell' sollen gültig sein!
   if (! strName.value.match(alphaExp)) {
      errMsg += "Der Name enthält falsche Zeichen.\n";
      frm.elements.fullname.focus();
      markObj("fullname");
   }
```

```
  if (frm.elements["fullname"].value == frm.elements["fullname"].defaultValue ||
      frm.elements["fullname"].value == "" ) {
    errMsg += "Geben Sie bitte Ihren Namen ein.\n";
    frm.elements.fullname.focus();
    markObj("fullname");
  }
  strEmail = frm.elements["email"].value
  if (strEmail == frm.elements["email"].defaultValue || strEmail == "") {
    errMsg += "Geben Sie bitte Ihre E-Mail-Adresse ein.\n";
    frm.elements.email.focus();
    markObj("email");
  } else {
    if(strEmail.indexOf('@')==-1 ||strEmail.indexOf('.')==-1) {
      errMsg += "Geben Sie bitte Ihre E-Mail-Adrese ein.\n\nEine gültige E-Mail-
      Adresse muss ein \n '@' (At-Zeichen) und einen '.' (Punkt)\n enthalten.\n";
      frm.elements.email.focus();
      markObj("email");
    }
  }
  if (frm.elements["subject"].value == frm.elements["subject"].defaultValue ||
      frm.elements["subject"].value == "") {
    errMsg += "Geben Sie bitte Ihren Betreff ein.\n";
    frm.elements.subject.focus();
    markObj("subject");
  } else {
    if (frm.elements["subject"].value.length < 5) {
      errMsg += "Der Betreff ist zu kurz: min. 5 Zeichen!\n";
      frm.elements.message.focus();
      markObj("subject");
    }
  }

  if (frm.elements["message"].value == "" || frm.elements["message"].value ==
      txtMessage) {
    errMsg += "Geben Sie bitte Ihre Nachricht ein.\n";
    frm.elements.message.focus();
    markObj("message");
  } else {
    if (frm.elements["message"].value.length < 10) {
      errMsg += "Die Nachricht ist zu kurz: min. 10 Zeichen!\n";
      frm.elements.message.focus();
      markObj("message");
    }
  }

  if (errMsg == "") {
    return true;
  } else {
    alert("Es ist ein Fehler aufgetreten!\n\n" + errMsg);
    return false;
  }
};
```

```
// Diese Funktion färbt das Feld, indem es das DOM-Objekt ändert.
function markObj(errObj) {
   var formFld = errObj;
   if (errObj == formFld) {
      var formFld = document.getElementsByName(formFld)[0];
      formFld.className = "errColor";
   }
   if (errObj == "salutation") {
      var formFld = document.getElementsByName("salutation")[0];
      formFld.className = "errColor";
      var formFld = document.getElementsByName("salutation")[1];
      formFld.className = "errColor";
   }
};
// Diese Funktion setzt die Hintergrundfarbe aller Felder auf 'strColor'
function resetBackColor(frm, strColor) {
   if(!frm || !frm.elements) { return; }
   var elms = frm.elements;
   for(var i = 0, maxI = elms.length; i < maxI; ++i) {
      var elm = elms[i];
      if (elm.type == "text" || elm.type == "textarea") {
         elm.style.background = strColor;
      }
   }
   document.getElementsByName("salutation")[0].style.background = "#336666";
   document.getElementsByName("salutation")[1].style.background = "#336666";
};

// Platzhalter für Pflichtfelder löschen
function focused(element) {
   if (element.value == element.defaultValue){element.value='';}
};

// Platzhalter für Pflichtfelder neu setzen
function blurred(element) {
   if (element.value == ''){element.value = element.defaultValue;}
};

// Mehrzeiliges Textfeld zeigt/löscht Vorbelegung beim Klicken
function SetMsg (txt, active) {
   var txtMessage = "Ihre Nachricht";
   if (txt == null) return;
   if (active) {
      if (txt.value == txtMessage) txt.value = '';
   } else {
      if (txt.value == '') txt.value = txtMessage;
   }
};
```

Listing 7: Plausibilitätsprüfung der sichtbaren Formularfelder mit *JavaScript*

5 Datenschutzerklärung

Auch bei einem privaten Kontaktformular ist eine Datenschutzerklärung notwendig. Sie lautet im vorliegenden Fall (s. auch Abbildung 2):

> Personenbezogene Angaben im Kontaktformular werden nach den geltenden Datenschutzbestimmungen streng vertraulich behandelt. Sie werden gegebenenfalls zu Beantwortung genutzt, aber nicht dauerhaft gespeichert. Bitte beachten Sie, dass der Inhalt des Kontaktformulars unverschlüsselt ist. Die Dienste des Webhosters **Strato AG**, Berlin, werden genutzt.

6 Verbesserungsmöglichkeiten

Falls gewünscht, kann das gezeigte Kontaktformular u. a. wie folgt verbessert werden:

6.1 Verzicht auf JavaScript

Die Plausibilitätsprüfungen der sichtbaren Felder des Kontaktformulars können genauso gut mit einem PHP-Skript erfolgen statt mit *JavaScript*. Das ist vorteilhaft, wenn *JavaScript* im jeweiligen Web-Browser deaktiviert oder nicht verfügbar ist. In diesem Fall muss gewöhnlich mit dem HTML-Block *<noscript> …</noscript>* eine entsprechende Meldung ausgegeben werden[6].

6.2 Tabellenloses Kontaktformular

Mit CSS kann ein tabellenloses statt eines tabellenbasierten Kontaktformulars erstellt werden. In [2] werden 9 gute Gründe aufgezählt, warum die tabellenlose Formulargestaltung vorteilhaft ist.

6.3 Vermeidung eines Popup-Fensters für Fehlermeldungen

Fehlermeldungen können direkt im Kontaktformular ausgegeben werden statt mit *JavaScript* in einem Popup-Fenster. Am Beispiel des Formularfelds mit dem Namen **email** wird diese Möglichkeit in Listing 8 demonstriert:

```
<p>
   <label for="email">E-Mail:</label>
   <input type="email" class="hvr" name="email" size="40"
        placeholder="Ihre E-Mail-Adresse"
        value="<?php printf("%s", $email);?>">
   <span class="error"><?php echo $email_Err;?></span><br>
</p>
```

Listing 8: Anzeige einer Fehlermeldung rechts neben dem betreffenden Formularfeld

Die Syntax in Listing 8 veranschaulicht,

- dass das **value**-Attribut dazu verwendet werden kann, um einen zuvor in der PHP-Variablen **$email** gespeicherten Wert erneut im Kontaktformular anzuzeigen
- dass die in der PHP-Variablen **$emailErr** gespeicherte Fehlermeldung unmittelbar hinter dem zugehörigen Formularfeld angezeigt wird, so dass ein entsprechendes Popup-Fenster gar nicht mehr benötigt wird.

[6] JavaScript im Internet-Explorer an-/abschalten: *Extras - Internetoptionen - Sicherheit - Stufe anpassen - Skripting - Active Scripting aktivieren bzw. deaktivieren.*

Formularwerte erneut anzuzeigen, wobei diese exakt so erhalten bleiben wie der Benutzer sie erfasst hat als das Formular das letzte Mal mit der Schaltfläche *submit* gesendet wurde, ist als „*sticky form*“ bekannt[7], vgl. dazu [3].

Wird der in Listing 8 gezeigte Ansatz analog auf alle übrigen Formularfelder übertragen, ergibt sich ein „klebriges“ Kontaktformular (s. Abbildung 2) mit folgenden Eigenschaften:

- Die Formularwerte bleiben nach dem Absenden des Formulars unverändert erhalten.
- Fehlermeldungen werden rechts neben den entsprechenden Formularfeldern ausgegeben.

Abbildung 2: Kontaktformular mit Erhalt der Eingabewerte und dazugehörigen Fehlermeldungen

In der Rubrik „*Verschiedenes*“ meiner Hompage *http://www.dr-thormaehlen.de* ist ein herunter ladbarer Beitrag mit dem Titel „*Formulardaten erhalten nach dem Absenden“* gespeichert. Anhand eines komplexen Beispielformulars wird dort detailliert erläutert, wie die Eingabedaten eines HTML-Formulars aufrecht erhalten werden können, nachdem es abgesendet wurde.

[7] Wörtlich aus dem Englischen übersetzt: „*klebriges Formular*“. Auf Deutsch: „*Affenformular*“.

Literaturverzeichnis

[1] T. Aschermann, „Was ist ein Captcha?,“ 16 01 2019. [Online]. Available: https://praxistipps.chip.de/was-ist-ein-captcha_11883. [Zugriff am 25 03 2019].

[2] o. V., „CSS vs. Tables- 9 Reasons Why CSS Tableless Designs are Better,“ o. J.. [Online]. Available: http://www.itegritygroup.com/css-vs-tables/. [Zugriff am 25 03 2019].

[3] o. V., „Sticky Forms,“ 07 03 2007. [Online]. Available: http://www.tutorialcode.com/php/sticky-forms/. [Zugriff am 25 03 2019].

[4] V. Garcia, „How to Use Honeypots to Fight Spam,“ 27 01 2015. [Online]. Available: https://www.ostraining.com/blog/coding/honeypot/. [Zugriff am 25 03 2019].

[5] o. V., „Simple CSS/PHP AntiSpam solution for a contact form,“ 26 09 2012. [Online]. Available: https://www.zen-cart.com/showthread.php?166212-Simple-CSS-PHP-AntiSpam-solution-for-a-contact-form/page11. [Zugriff am 25 03 2019].

[6] o. V., „Wie Sie JavaScript in Ihrem Browser aktivieren,“ [Online]. Available: https://www.enable-javascript.com/de/. [Zugriff am 25 03 2019].

[7] o. V., „Affenformular,“ 16 05 2015. [Online]. Available: https://de.wikipedia.org/wiki/Affenformular. [Zugriff am 25 05 2019].