

# **Methoden der Darstellung von Emailadressen in Webseiten zur Verhinderung des automatisierten Auslesens durch Spambots**

Eine Studienarbeit von

**Christian Matthias Reuter**

vorgelegt an der

**Universität Paderborn**

im Fachbereich Informatik bei

**Prof. Gerd Szwillus**

Paderborn, den 13. September 2006

# Inhalt

<b>Einleitung.....</b>	<b>5</b>
Spam ist ein Problem.....	5
Woher kommt das Problem?.....	6
Lösungsansätze des Spamproblems.....	7
<b>Mailverweise auf Internetseiten.....</b>	<b>9</b>
Einbinden einer Emailadresse.....	9
Sicherheit dieser Methode.....	9
Mailverweise: Standards und Gestaltungsregeln.....	9
Mailverweise nach HTML 2.0.....	9
Uniform Resource Identifier.....	10
Format einer Emailadresse.....	10
Aktivierung eines Verweises.....	11
Gestaltungsregeln.....	11
<b>Kriterien für eine gute Darstellungsmethode.....</b>	<b>12</b>
Funktionalität.....	12
Gebrauchstauglichkeit.....	12
Sicherheit.....	15
<b>Untersuchung: Darstellungsmethoden und Nutzererkennung.....</b>	<b>18</b>
<b>Methoden der Darstellung von Emailadressen.....</b>	<b>20</b>
Standardmailverweis.....	20
Funktionalität.....	21
Gebrauchstauglichkeit.....	21
Sicherheit.....	21
Verschleierung der Emailadresse.....	22
Maskierung des @-Zeichens.....	22
Funktionalität.....	23
Gebrauchstauglichkeit.....	23
Sicherheit.....	24
Hinzufügen von adressfremden Bestandteilen.....	24
Funktionalität.....	25
Gebrauchstauglichkeit.....	25
Sicherheit.....	26
Maskierung mit Unicode.....	27
Funktionalität.....	28
Gebrauchstauglichkeit.....	28
Sicherheit.....	28
Adresse als Grafik.....	29
Funktionalität.....	30
Gebrauchstauglichkeit.....	30
Sicherheit.....	31
Entzerren der Adresse.....	31
Funktionalität.....	32

Gebrauchstauglichkeit.....	32
Sicherheit.....	33
Methoden, die Javascript verwenden.....	33
Javascript: Text dynamisch erzeugen.....	34
Funktionalität.....	35
Gebrauchstauglichkeit.....	35
Sicherheit.....	36
Javascript: linkto_uncryptmailto().....	36
Funktionalität.....	38
Gebrauchstauglichkeit.....	38
Sicherheit.....	38
Eine Möglichkeit, die Sicherheit zu erhöhen.....	39
Javascript mit Nutzerinteraktion.....	40
Funktionalität.....	41
Gebrauchstauglichkeit.....	41
Sicherheit.....	42
Methoden mit CSS.....	43
CSS-Eigenschaft direction:rtl.....	43
Funktionalität.....	44
Gebrauchstauglichkeit.....	44
Sicherheit.....	45
CSS-Pseudoklasse :after.....	46
Funktionalität.....	47
Gebrauchstauglichkeit.....	47
Sicherheit.....	48
<b>Gweax – ein gutartiger Spambot.....</b>	<b>50</b>
Hauptklasse Gweax.....	50
Addressverwaltung AddressManager.....	51
Filter GweaxFilter.....	51
MailtoFilter.....	51
MaskedAtFilter.....	51
AddedCapitalsFilter.....	52
FindUnlinkedAddressesFilter.....	52
BackwardMaskingFilter.....	52
LinktoUncryptFilter.....	52
<b>Zusammenfassung.....</b>	<b>53</b>
Unbrauchbare Methoden.....	53
Fehlende Sicherheit.....	53
Fehlende Gebrauchstauglichkeit.....	53
Brauchbare Methoden.....	54
Standardmethode.....	54
Adresse als Grafik.....	54
Entzerren der Adresse.....	54
Javascript: Text dynamisch erzeugen.....	55
Javascript: linkto_uncryptmailto().....	56

CSS-Pseudoklasse :after.....	56
Auf einen Blick.....	57
Auswertung der Untersuchung.....	58
<b>Quellen.....</b>	<b>59</b>
Standards.....	59
Untersuchungen und Berichte.....	59
Methoden.....	60
<b>Anhang: Inhalt der CD.....</b>	<b>62</b>

## Abbildungsverzeichnis

Abbildung 1: Standardmailverweis.....	20
Abbildung 2: Maskierung des @-Zeichens.....	23
Abbildung 3: Hinzufügen von adressfremden Bestandteilen.....	25
Abbildung 4: Maskierung mit Unicode.....	27
Abbildung 5: Adresse als Grafik, zum Vergleich darunter Adresse als Text.....	30
Abbildung 6: Entzerren der Adresse.....	32
Abbildung 7: Javascript: Text dynamisch erzeugen.....	34
Abbildung 8: Javascript: linkto_uncryptmailto().....	37
Abbildung 9: Eingabefenster für die Nutzerinteraktion.....	40
Abbildung 10: CSS-Eigenschaft direction:rtl.....	44
Abbildung 11: CSS-Pseudoklasse :after.....	47
Abbildung 12: Bildschirmfoto Gweax.....	50

## Einleitung

---

Ich habe heute bisher zwölf Emails bekommen – davon elf Spam. In den letzten sieben Tagen waren es 388 Mails, davon 349 Spam (89,9 %). Seit Jahresbeginn bekam ich 6320 Mails, davon 5672 Spam (89,8 %). Etwa 90 Prozent aller meiner Emails sind Spam. Die Firma Messagelabs, die Sicherheitslösungen in der Informationstechnologie für Unternehmen anbietet und dazu insbesondere Emails auf Spam- und Virenbefall analysiert, gibt in ihrem Bericht für Juli 2006<sup>1</sup> die Rate an Spammails mit 62,5 % an.

Diese beiden Werte entsprechen in etwa dem oberen und unteren Ende dessen, was im Allgemeinen an seriösen Schätzungen veröffentlicht wird; und unabhängig davon, ob es sich nun um 60 oder 90 Prozent handelt, sagen diese Zahlen eindeutig, dass Spam ein großes Problem im Umgang mit Emails ist.

### *Spam ist ein Problem*

Fast jeder, der eine Emailadresse hat, kennt die Probleme, die mit Spam auftreten: Vielfache Belästigung, Inhalte, die eventuell die persönlichen Gefühle verletzen, Mails, die irrtümlich für Spam gehalten werden, Mails, die irrtümlich für seriös gehalten werden und so weiter.

Spam ist mehr als eine störende, aber ansonsten harmlose Belästigung. Eine Studie von Group Technologies anlässlich des zweiten deutschen Antispamkongresses im Jahr 2004 beziffert den finanziellen Schaden für ein Unternehmen, der durch das Auftreten von Spam entsteht, auf etwa 300 Euro pro Mitarbeiter und Jahr<sup>2</sup>. Das norwegische Unternehmen Norman geht sogar von 1000 Euro pro Mitarbeiter und Jahr aus<sup>3</sup>. Spam ist also nicht nur lästig, sondern auch teuer. Der volkswirtschaftliche Schaden in der EU wird auf 2,4 Milliarden Euro jährlich geschätzt.<sup>4</sup>

---

1 [http://www.messagelabs.com/publishedcontent/publish/threat\\_watch\\_dotcom\\_en/intelligence\\_reports/july\\_2006/DA\\_155200.chp.html](http://www.messagelabs.com/publishedcontent/publish/threat_watch_dotcom_en/intelligence_reports/july_2006/DA_155200.chp.html)

2 *E-Mail Lifecycle Management – mehr als nur Anti-Spam*, Folie 4, Download nicht mehr verfügbar. <http://www.group-technologies.com>

3 aus: *Why Spammers spam*, [http://download.norman.no/whitepapers/Why\\_spammers\\_spam.pdf](http://download.norman.no/whitepapers/Why_spammers_spam.pdf)

4 [http://www.tagesschau.de/aktuell/meldungen/0,1185,OID5624320\\_REF2,00.html](http://www.tagesschau.de/aktuell/meldungen/0,1185,OID5624320_REF2,00.html)

## *Woher kommt das Problem?*

Der Versand von Spam ist im Gegensatz zu herkömmlicher Werbung sehr kostengünstig; verwendet ein Spammer trojanerbefallene Rechner zum Versand, entstehen ihm fast keine Kosten. Zudem werden meist Produkte mit hohen Gewinnmargen verkauft, so dass sogar eine sehr geringe Antwortrate ausreicht, um großen Gewinn zu erzielen. Das schon erwähnte Unternehmen Norman hat in einer Untersuchung der Motive der Spammer Kontakt zu aktiven Spammern gesucht. Diese gaben ihren Gewinn mit 150 bis 6.500 Dollar an, im Schnitt 1.200\$ - pro Woche<sup>5</sup>.

*„I calculated that I earned around \$1,200.00 per week last year. The most I earned in a week was \$6,500.00, and the least was \$1.00, so I have done the gambit.“*

Topspammer dürften ein Vielfaches davon verdienen. Dazu versenden sie täglich Millionen von Emails: Ein kürzlich in den Vereinigten Staaten verurteilter Spammer gab an, bis zu 25 Millionen Spams täglich versendet zu haben. Damit lag er auf der ROKSO-Liste<sup>6</sup> der bekannten Spammer auf Platz vier<sup>7</sup>.

Wie kommen Spammer an Emailadressen? Das Center for Democracy & Technology<sup>8</sup> begann im Sommer 2002 eine Studie, in deren Rahmen es 250 neue Emailadressen auf unterschiedliche Art veröffentlichte; auf Webseiten, im Usenet, bei Bestellungen von Newslettern etc. Über sechs Monate wurden die eingehenden Mails untersucht, das Ergebnis war erstaunlich: 97% der Spammails gingen an Adressen, die auf Webseiten veröffentlicht wurden.

Eine ähnliche Studie der Federal Trade Commission (FTC<sup>9</sup>) aus dem Jahr 2002 kam zu dem Ergebnis, dass

*„86 percent of the addresses posted to web pages received spam. It didn't matter where the addresses were posted on the page: if the address had the "@" sign in it, it drew spam.“*

---

5 *Why Spammers spam*, Seite 4

6 Register Of Known Spam Operations  
<http://www.spamhaus.org/rokso/index.lasso>

7 <http://www.heise.de/newsticker/meldung/73914>

8 <http://www.cdt.org/speech/spam/030319spamreport.shtml>

9 <http://www.ftc.gov/bcp/online/pubs/alerts/spamalrt.htm>

Spammer können so leicht arbeiten, weil es sehr einfach ist, an Emailadressen zu kommen. Mit Hilfe sogenannter Spambots durchsuchen sie Webseiten nach Emailadressen. Dabei werden die Emailadressen einer Seite extrahiert und dann die Seiten durchsucht, auf die von dieser Seite verwiesen wird. Wer im Internet danach sucht, findet schnell viele Programme, die laut Herstellerangaben bis zu 100.000 Adressen pro Stunde finden<sup>10</sup>.

Alternativ können Spammer Adresslisten von Adresshändlern kaufen (mir wurde vor etwa einem Jahr eine Liste mit 250 Millionen Adressen für 70 Dollar angeboten<sup>11</sup>). Auch in dem schon zitierten Norman-Bericht geben die Befragten an, anfangs Listen gekauft zu haben:

*„I bought my first list from the friend who introduced me to spamming for \$100.00; this was a pretty good deal because it had over 15 million names on it.“*

### *Lösungsansätze des Spamproblems*

Nun gibt es verschiedene Methoden zur Spambekämpfung, am häufigsten eingesetzt werden Spamfilter, die nach verschiedenen Kriterien reguläre und Spammails unterscheiden wollen. Ich selbst habe in einer Untersuchung im Rahmen eines Praktikums eine dreistufige Strategie vorgeschlagen: aktives Vermeiden (Sicherstellen, dass eigene Mails nicht für Spam gehalten werden), Erkennung (Filtern von Spam) sowie Prävention (Verhindern, dass Spammer die eigene Emailadresse bekommen).

In dieser Arbeit möchte ich einen Punkt der Prävention untersuchen: verschiedene Methoden der Darstellung von Emailadressen auf Webseiten zur Verhinderung des automatisierten Auslesens durch Spambots<sup>12</sup>.

---

<sup>10</sup> z.B. <http://tecsoftware.biz/extractor.htm>

<sup>11</sup> Dieses Angebot habe ich per Email erhalten, es war also selbst Spam. Über die Aktualität der Liste kann ich keine Aussagen machen.

<sup>12</sup> Der Begriff „Spambot“ wird gelegentlich verwendet für Computer, die durch Trojaner zum Spamversand ferngesteuert werden. Programme, die nach Emailadressen suchen, werden dann als „Crawler“, „Spider“ oder „Harvester“ bezeichnet. Meines Erachtens ist der Begriff „bot“ für Suchprogramme weiter verbreitet, so bezeichnet die Suchmaschine Google das eigene Suchprogramm als „googlebot“. Da es sich hier um die Suche nach Emailadressen handelt, verwende ich den Begriff „Spambot“.

In den letzten Jahren wurden nun mehrere Verfahren entwickelt, um ein solches automatisiertes Auslesen zu verhindern; die ersten Methoden sind mittlerweile nicht mehr effektiv, einige der neueren sind so ausgefeilt, dass die Adressen nur mit großem Aufwand ausgelesen werden können.

Leider gibt es keine Methode, die Gebrauchstauglichkeit, Funktionalität und Sicherheit gleichermaßen im vollen Umfang bietet. Daher müssen die vorhandenen Methoden auf möglichst gute Umsetzung hin verglichen werden.

## Mailverweise auf Internetseiten

---

### *Einbinden einer Emailadresse*

Wie eigentlich werden Emailadressen auf Webseiten verwendet? Im Prinzip wird ein Verweis erstellt, der als Zieladresse eine Emailadresse enthält. Mit

```
<a href="mailto:name@example.com">
  name@example.com
</a>
```

wird so ein Emailverweis definiert. Im Browser wird das in etwa so dargestellt: [name@example.com](mailto:name@example.com) (siehe auch , Seite ). Beim Anklicken dieses Verweises öffnet sich das Mailprogramm mit einer neuen, an diese Adresse gerichteten Mail. Diese Methode bezeichne ich als Standardmethode oder Standardemailverweis.

### *Sicherheit dieser Methode*

Dass Adressen so nicht geschützt sind, liegt auf der Hand. Man muss nicht viel tun, um sie auszulesen: Man durchsucht den Quelltext der Webseite nach dem Vorkommen von `href="mailto:` und kopiert alle Zeichen nach dem Doppelpunkt bis zum nächsten Anführungszeichen. Offensichtlich ist das sehr einfach.

### *Mailverweise: Standards und Gestaltungsregeln*

Soweit die Praxis. Zum tieferen Verständnis dieser und der anderen Methoden möchte ich hier kurz die Theorie, also Standards und Gestaltungsregeln, darlegen.

### Mailverweise nach HTML 2.0

Die erste offizielle Version von HTML war HTML 2.0 vom November 1995<sup>13</sup>. Im Request for Comments (RFC) 1866 werden die verschiedenen Eigenschaften von HTML beschrieben, unter anderem Hyperlinks. Hyperlinks

---

<sup>13</sup> Berners-Lee, Tim und Conolly, Daniel.: *Hypertext Markup Language*, RFC1866  
<http://www.ietf.org/rfc/rfc1866.txt>

sind Elemente zur Navigation zwischen verschiedenen Ressourcen im Internet und bestehen aus einem Start- und einem Zielanker, die jeweils durch einen Uniform Resource Identifier (URI) eindeutig bestimmt sind:

*„HTML documents can express hyperlinks. An HTML user agent allows the user to navigate these hyperlinks. A hyperlink is a relationship between two anchors, called the head and the tail of the hyperlink. Anchors are identified by an anchor address: an absolute Uniform Resource Identifier (URI).“*

## Uniform Resource Identifier

Ein Mailverweis wird nicht explizit definiert, kann aber durch einen passenden URI erstellt werden. Was als URI zulässig ist, beschreibt das Request for Comments 1630 vom Juni 1994<sup>14</sup>. Unter anderem wird dort das so genannte Schema `mailto:` beschrieben, um eine Emailadresse nach RFC822 zu referenzieren:

*„Mailto: This allows a URL to specify an RFC822 addr-spec mail address.“*

## Format einer Emailadresse

Der Standard RFC1630 verweist bei der Definition eines Mailverweises auf RFC822. Eine Emailadresse hat demnach das Format `localpart@domain` (zum Beispiel `name@example.com`).

Im Allgemeinen werden als `localpart` nur alphanumerische Zeichen sowie Unterstrich, Bindestrich und Punkt verwendet, wobei die Sonderzeichen immer von alphanumerischen Zeichen umgeben sind. RFC822 erlaubt daneben noch weitere Zeichen: `!#$%&'*/=?^`{|}` In der Praxis sind diese Zeichen aber sehr selten anzutreffen.

Als Zeichen im Domainpart sind alphanumerische Zeichen sowie Punkt und Bindestrich erlaubt, wobei der Punkt von alphanumerischen Zeichen umgeben sein muss.

---

<sup>14</sup> Berners-Lee, Tim: *Universal Resource Identifiers in WWW*, RFC 1630  
<http://www.ietf.org/rfc/rfc1630.txt>

## Aktivierung eines Verweises

RFC1866 schreibt nicht vor, wie ein Browser bei Aktivierung eines Hyperlinks (zum Beispiel durch Anklicken) zu verfahren hat, es wird lediglich festgelegt, dass der Browser sich eine Repräsentation der Ressource verschafft. Wie das auszusehen hat und wie der Umgang damit erfolgt, wird nur für HTML-Dokumente empfohlen.

In der Praxis ist es bei den meisten Browsern möglich, dass bei Aktivierung eines Hyperlinks, der auf eine Emailadresse verweist, das Emailprogramm aufgerufen wird, das eine neue Email öffnet in der der Empfänger durch die angegebene Adresse bereits bestimmt ist.

## Gestaltungsregeln

Von dieser Möglichkeit wird häufig kein Gebrauch gemacht. Viele Nutzer sind Kunden webbasierter Mailanbieter und verwenden kein Emailprogramm. Um einen Emailverweis nutzen zu können, müssen diese Personen in der Lage sein, die Adresse selbständig zu extrahieren. Im Sinne der Gebrauchstauglichkeit empfiehlt es sich also, Gestaltungsregeln zu beachten. So schreibt Stefan Münz, Autor eines populären HTML-Tutoriums:

*„Es ist sinnvoll, im Verweistext die E-Mail-Adresse noch einmal explizit zu nennen (...), damit Anwender, bei denen der E-Mail-Verweis nicht ausführbar ist, auf Wunsch separat eine E-Mail senden können.“<sup>15</sup>*

Wie wichtig das ist, hat sich in einer Untersuchung gezeigt, die ich im Rahmen dieser Arbeit durchgeführt habe (siehe übernächstes Kapitel). Gut ein Viertel der Teilnehmer waren nicht in der Lage, eine Emailadresse zu erkennen, wenn sie nicht im Klartext zu lesen war.

---

<sup>15</sup> <http://de.selfhtml.org/html/verweise/email.htm>

## Kriterien für eine gute Darstellungsmethode

---

Ich habe bereits angesprochen, dass sich die verschiedenen Darstellungsmethoden von Emailadressen in Funktionalität, Gebrauchstauglichkeit und Sicherheit unterscheiden und keine der Methoden alle drei Kriterien voll erfüllt. Bevor ich die Methoden analysiere, möchte ich die Untersuchungskriterien definieren.

### *Funktionalität*

Da in keinem Standard definiert ist, wie sich Browser bei Aktivierung eines Mailverweises verhalten sollen, erscheint eine Definition von Funktionalität verschiedener Darstellungsformen schwierig. Greift man aber zurück auf das Verhalten des Browsers bei einem Standardmailverweis, ergibt sich eine Grundlage. Dazu gehört sowohl das Verhalten bei Aktivierung des Verweises als auch das Verhalten beim Überfahren des Verweises mit der Maus (Anzeige von Informationen z.B. in der Statuszeile).

**Eine Darstellungsmethode einer Emailadresse ist dann funktional, wenn der Browser sich so verhält wie bei einem normalen Emailverweis.**

Zur Beurteilung werde ich mich an folgenden Fragen orientieren (und sie in der Zusammenfassung explizit beantworten). Je mehr Fragen mit „Ja“ beantwortet werden können, desto besser ist die Methode.

1. Öffnet sich bei Aktivieren des Verweises das Emailprogramm (sofern im Browser die Behandlung des mailto-Schemas eingestellt ist)?
2. Wird beim Überfahren des Verweises die Emailadresse angezeigt?

### *Gebrauchstauglichkeit*

Das Internet und seine Nutzer sind in ihrer Heterogenität kaum zu überbieten: Mein dreijähriger Sohn spielt einfache Browserspiele, ich selbst habe beruflich mit dem Internet zu tun, meine Eltern erledigen Recherche und Unterrichtsvorbereitung im Internet und sogar meine Großeltern mit über 80

Jahren besuchen gelegentlich die Webseiten ihrer Enkel. Dazu verwenden wir unterschiedliche Browser in verschiedenen Versionen (unter anderem Opera, Firefox, Internet Explorer) unter verschiedenen Betriebssystemen (Windows, Linux).

Diese Browser (sowie sonstige wie z.B. der Netscape Navigator, Lynx, Safari, Konquerer etc.) unterscheiden sich unter anderem in der Unterstützung der verschiedenen Standards. Die meisten Browser bieten mittlerweile eine vollständige Umsetzung von Javascript 1.3; die Version 1.5 und mit ihr das Document Object Model 2.0 werden jedoch nur von den neuesten Versionen unterstützt, und das nicht immer vollständig. Ähnlich verhält es sich bei CSS. Die Version 1.0 ist fast überall komplett umgesetzt, CSS 2.0 dagegen nur teilweise.

Damit eine Methode als gebrauchstauglich bezeichnet werden kann, muss sie sich Techniken bedienen, die eine Darstellung in allen heute gebräuchlichen Browsern verschiedener Generationen erlaubt.

Das allein reicht jedoch noch nicht aus. Die Darstellung muss auch so gewählt sein, dass alle Nutzer des Internets damit zurecht kommen. Das bedeutet insbesondere, dass Personen mit Sehschwächen damit umgehen können müssen. Die Darstellung einer Emailadresse als Grafik kann – je nach Größe – somit untauglich sein. Darüber hinaus müssen auch Laien mit der Methode zurecht kommen. Methoden, die die Emailadresse so verändern, dass die übliche Struktur gebrochen wird, können zwar Spambots verwirren, aber eben auch legitime Nutzer.

Zur Beurteilung der Gebrauchstauglichkeit habe ich eine Untersuchung durchgeführt, in der ich Emailadressen mit Hilfe der verschiedenen Methoden dargestellt habe und die Teilnehmer gebeten habe, die dahinterstehenden Adressen zu erkennen. Neben den theoretischen Überlegungen liefert diese Untersuchung wichtige Erkenntnisse zur Gebrauchstauglichkeit, die in die Beurteilung der Methoden einfließen. Nähere Informationen zu der Untersuchung finden sich im nächsten Kapitel.

Gebrauchstauglichkeit definiere ich also wie folgt:

**Eine Darstellungsmethode einer Emailadresse ist dann gebrauchstauglich, wenn sie von allen Nutzern verstanden wird und sich Techniken bedient, die eine Darstellung in allen heute gebräuchlichen Browsern ermöglicht.**

Zur Beurteilung werde ich mich an folgenden Fragen orientieren (und sie in der Zusammenfassung explizit beantworten). Je mehr Fragen mit „Ja“ beantwortet werden können, desto besser ist die Methode:

1. Wird die Adresse wie gewünscht dargestellt im Internet Explorer 6.0?

Der Internet Explorer ist der am weitesten verbreitete Browser. Eine Methode sollte zumindest den Großteil der Browser abdecken können.

2. ... in Firefox 1.5?

Firefox ist der zweithäufigst genutzte Browser und sollte daher ebenfalls beachtet werden.

3. ... in Opera 9.0?

Der Marktanteil von Opera ist eher gering. Opera ist aber als Browser vorbildlich in der Umsetzung von Standards. Wenn eine Methode nicht von Opera unterstützt wird, scheint sie noch nicht ausgereift zu sein.

4. ... in Safari?

Safari ist der Standardbrowser des Applebetriebssystems MacOS. Eine gute Methode sollte betriebssystem- und browserunabhängig sein, Safari wurde hier stellvertretend gewählt für eine ganze Reihe von Nischenbrowsern.

5. ... in Lynx (einem Textbrowser)?

Der Marktanteil von Lynx ist wirklich vernachlässigbar. Die Darstellbarkeit in Lynx ist aber ein gutes Maß für die Barrierefreiheit, was er darstellt ist

vergleichbar mit dem, was z.B. Sehbehinderte auf einer Webseite erkennen können.

#### 6. Kommt die Methode ohne Javascript aus?

Es gibt eine Reihe von Internetnutzern, die Javascript aus Sicherheitsgründen deaktiviert haben; eine gute Methode sollte auch dann funktionieren.

#### 7. Kommt die Methode ohne CSS aus?

Ältere Generationen von Browsern sowie Browser für Mobiltelefone unterstützen CSS nicht oder nur teilweise. Auch auf solchen Browsern sollte eine Methode nutzbar sein.

#### 8. Wird die Emailadresse angezeigt?

In der Untersuchung, die ich im Rahmen dieser Arbeit durchgeführt habe, hat sich gezeigt, wie sehr die Nutzer darauf angewiesen sind, die Emailadresse sichtbar vor sich zu haben. Insbesondere ist wichtig, dass die Anzeige der Emailadresse nicht an die Verarbeitung des mailto-Schemas gebunden ist.

#### 9. Ist die Emailadresse selektierbar?

Eine Methode sollte so benutzerfreundlich wie möglich sein. Dazu gehört, dass der Nutzer die Emailadresse nicht abtippen müssen soll, um sie zu nutzen.

Zusätzlich fließen die Ergebnisse der Untersuchung in die Bewertung ein. Dazu ermittle ich die Erkennung und die Akzeptanz. Unter Akzeptanz verstehe ich den Anteil der gewerteten Teilnehmer, die eine Adresse erkannt haben, unabhängig davon, ob sie richtig ist oder nicht. Der Anteil davon, der die Adresse richtig erkannt hat, bestimmt die Erkennung<sup>16</sup>.

## *Sicherheit*

Im Idealfall verhindert eine Methode das automatisierte Auslesen. Dies ist aber schwer verifizierbar. Zwar kann ich ein Programm schreiben, das Email-

---

<sup>16</sup> Beispiel: Bei einer Methode wurden zwei Teilnehmer nicht gewertet; 20 haben keine, 23 eine falsche und 22 die richtige Adresse angegeben. Die Akzeptanz beträgt nun 45 von 65, also 0,692, die Erkennung 22 von 45, also 0,489

adressen aus Webseiten extrahiert; ein Erfolg zeigt aber weder die Unsicherheit einer Methode, noch ist ein Misserfolg ein Beweis für die Sicherheit. Eine Analyse der von Spammern in der Praxis verwendeten Programme ist wegen der fehlenden Quelloffenheit nicht möglich.

Eine Methode wäre die stochastische Analyse. Dazu könnte ich eine große Anzahl von Emailadressen neu erstellen und mit den unterschiedlichen Darstellungsmethoden auf verschiedenen Webseiten publizieren. Nach einem bestimmten Zeitraum müsste dann überprüft werden, welche Adressen Spam erhalten haben – und damit welche Methoden in der Praxis nicht sicher sind. Eine solche Untersuchung übersteigt aber den Rahmen dieser Arbeit.

Ich werde versuchen, die Sicherheit (insbesondere die fehlende) argumentativ zu belegen. Eine strikte Definition von Sicherheit, nach der die Methoden untersucht werden, gibt es somit nicht.

Wichtig für alle Methoden ist, dass die Sicherheit nicht auf ihrer geringen Verbreitung oder Obskürität beruhen sollte. Eine Methode zu entwickeln, die zwar sicher ist, aber nicht verbreitet werden darf, weil sie dadurch unsicher würde, ist sinnlos. Grundsätzlich gilt: Wenn eine Methode Verbreitung findet, lohnt es sich für Spammer zu versuchen, sie anzugreifen.

Zur Beurteilung werde ich mich an folgenden Fragen orientieren (und sie in der Zusammenfassung explizit beantworten). Je mehr Fragen mit „Ja“ beantwortet werden können, desto besser ist die Methode.

1. Werden Indikatoren wie „mailto“ oder das @-Zeichen vermieden?

Durch „mailto“ oder das @-Zeichen wird deutlich auf eine Emailadresse hingewiesen. Gute Spambots können aus der Umgebung des @-Zeichens oder von „mailto“ die Emailadresse extrahieren.

2. Braucht ein Spambot zum Auslesen einen Javascriptinterpreter?

Javascript bietet die Möglichkeit, eine Emailadresse so zu verschleiern oder zu verschlüsseln, dass ein Spambot ohne einen vollständigen Javascriptinterpreter die Adresse nicht auslesen kann. Der Aufwand, jedes Skript auszuführen, um

eine Emailadresse auszulesen, ist viel zu hoch, als dass es sich für einen Spambot lohnt.

### 3. Braucht ein Spambot zum Auslesen Bilderkennung?

Die automatische Bilderkennung ist nicht in der Lage, in Grafiken vorhandene Emailadressen sicher auszulesen. Eine Methode, die darauf baut, ist also ziemlich sicher.

### 4. Bietet die Methode sonstige Schutzmechanismen?

Bei manchen Methoden müssen die drei vorigen Fragen mit nein beantwortet werden. Sicherheit kann dennoch gewährleistet sein, wenn sie ein anderes Sicherheitskonzept verfolgen.

### 5. Widersteht die Methode meinem Adresssuchprogramm?

Um zu zeigen, dass viele der hier vorgestellten Methoden nicht sicher sind, habe ich Gweax geschrieben, ein Programm, das Emailadressen aus einer Datei extrahiert. Nähere Informationen finden sich im Kapitel „Gweax – ein gutartiger Spambot“, das Programm selbst auf der beiliegenden CD.

## Untersuchung: Darstellungsmethoden und Nutzererkennung

---

Neben den theoretischen Überlegungen, wie die einzelnen Methoden sich an Standards halten, welche Technologien sie einsetzen, die eventuell nicht von allen Browsern umgesetzt werden, und welche Schwierigkeiten bei den Nutzern auftreten können hat mich besonders interessiert, wie die Nutzer in der Praxis in der Lage sind, die hinter den verschiedenen Darstellungsmethoden verborgenen Emailadressen zu erkennen.

Dazu habe ich eine Untersuchung durchgeführt, bei der die zehn hier vorgestellten Verfahren und zusätzlich die Standardmethode verwendet wurden, um Emailadressen darzustellen. In einer Tabelle befanden sich in der ersten Spalte die verschiedenen Methoden angewandt auf fiktive Emailadressen und in einer zweiten Spalte Eingabefelder. Aufgabe der Teilnehmer der Untersuchung war nun, die Emailadressen zu erkennen und in die entsprechenden Eingabefelder einzutragen.

Für die verwendeten Emailadressen benutzte ich die für solche Zwecke vorgesehenen Domains *example.com*, *example.net* und *example.org*, teilweise mit zusätzlichen Subdomains. Das diente dazu, die Teilnehmer der Untersuchung daran zu hindern, versehentlich Emails an existierende Adressen unbeteiligter Domains zu senden.

Bei gut der Hälfte der Adressen habe ich entgegen der Gestaltungsregeln die Emailadresse im Verweistext nicht wiederholt. Das geschah einerseits, um herauszufinden, wie sehr Nutzer darauf angewiesen sind, die Adresse im Klartext vorliegen zu haben, andererseits hätte es bei zwei Methoden die Sicherheit reduziert. Sofern das Fehlen dieser Adresswiederholung nicht Bestandteil der Methode. War, habe ich zur Beurteilung der Gebrauchstauglichkeit nur die Ergebnisse herangezogen, bei denen die Teilnehmer eine Adresse erkannt haben.

Die Untersuchung habe ich im Freundeskreis, unter Verwandten sowie in diversen Foren im Internet bekannt gemacht. Um auch Nutzer außerhalb des deutschen Sprachraums zu erreichen, gab es auch ein Formular auf Englisch, das ich in zwei englischsprachigen Foren veröffentlicht habe. Bei der Auswahl der Foren habe ich computerorientierte Foren vermieden, da bei Nutzern

solcher Foren eine überdurchschnittliche Kompetenz in diesem Bereich zu erwarten ist, was die Ergebnisse verfälscht hätte.

Das Formular findet sich im Internet unter <http://homepages.upb.de/cmreuter/arbeiten/bt/nutzerumfrage.html>. Zum besseren Verständnis der Methoden ist es sinnvoll, die Umfrage gesehen oder sogar selbst daran teilgenommen zu haben.

Insgesamt gab es 67 Teilnehmer, davon fünf englischsprachige. Von den 67 Teilnehmern haben offensichtlich zwei die Aufgabe nicht verstanden und wurden in der Auswertung nicht berücksichtigt. Leider ist die Anzahl der Teilnehmer für eine repräsentative Umfrage zu gering, dennoch lassen sich für die Gebrauchstauglichkeit verwertbare Tendenzen erkennen. Keine Aussage kann ich jedoch darüber treffen, inwiefern das Alter der Teilnehmer für das Ergebnis eine Rolle spielt, da über sechzig Prozent der Teilnehmer der Altersgruppe der Zwanzig- bis Neunundzwanzigjährigen angehören, und die anderen Altersgruppen zu gering besetzt sind.

Eine genauere Auswertung der Untersuchung findet sich auf oben genannter Webseite, aufgrund der fehlenden Aussagekraft fließen keine detaillierteren Ergebnisse in die Bewertung der Methoden ein.

## Methoden der Darstellung von Emailadressen

---

Nun endlich komme ich zur Analyse der Methoden. Die verschiedenen Methoden lassen sich in drei Gruppen einteilen: Die meistgenutzten versuchen, die Emailadresse so zu maskieren, dass sie nicht mehr von Suchmaschinen als solche erkannt wird, wohl aber von menschlichen Nutzern. Dass dabei die Funktionalität teilweise oder ganz verloren geht, wird in Kauf genommen. Andere Methoden verwenden Javascript, um die Emailadresse dynamisch zu erzeugen. Die neusten Methoden verwenden CSS-Eigenschaften zu Darstellung.

Zunächst möchte ich aber – als Referenz – die Standardmethode untersuchen.

### Standardmailverweis

Mit

```
<a href="mailto:name@example.com">
  name@example.com
</a>
```

wird ein Emailverweis definiert; als Anker dient ein URI mit Angabe einer Emailadresse. Als Verweistext wird diese Adresse wiederholt, was aus Gründen der Nutzbarkeit geschieht: der Nutzer soll einerseits vor Aktivierung eines Verweises wissen, dass es sich um einen Emailverweis handelt, andererseits soll er die Adresse auch dann erkennen können, wenn der Emailverweis nicht ausführbar ist.

Im Browser sieht der Verweis dann so aus (Abbildung 1):



Abbildung 1: Standardmailverweis

Wie verhält sich die Standardmethode nun in Hinblick auf die zu untersuchenden Kriterien Funktionalität, Gebrauchstauglichkeit und Sicherheit?

### ***Funktionalität***

Die Standardmethode ist per definitionem funktional. Schließlich dient sie als Grundlage der Definition von Funktionalität.

### ***Gebrauchstauglichkeit***

Die von der Standardmethode verwendeten Elemente entstammen dem HTML-Standard 2.0 aus dem Jahr 1994 und werden z.B. vom Netscape Navigator seit der Version 2.0, vom Internet Explorer seit der Version 4.0 und von Opera seit der Version 3.0 unterstützt<sup>17</sup>. Eine Darstellung in allen Browsern ist somit gegeben. Zudem wird die Emailadresse im Klartext angegeben, so dass sie ohne weiteres von den Nutzern als solche erkannt wird.

In der Untersuchung wurde von allen 63 Teilnehmern, die eine Adresse erkannt haben, diese korrekt extrahiert.

### ***Sicherheit***

Leider ist die Standardmethode nicht sicher; mit nur wenigen Zeilen kann ein Filter programmiert werden, der die Adressen automatisch ausliest<sup>18</sup>:

```
function getEmailAddress(line)
{
  // überprüfen, ob die Zeile einen Mailverweis
  //enthält, gleichzeitig wird der Anfangspunkt
  //der Adresse bestimmt
  var start = line.indexOf("href=\"mailto:");

  if (start != -1) // Zeile enthält Mailverweis
  {
```

<sup>17</sup> Sämtliche Angaben zur Unterstützung durch verschiedene Browser sind Selfhtml (de.selfhtml.org) entnommen. Es kann durchaus sein, dass einige Browser die Elemente auch schon in älteren Versionen als angegeben unterstützen. So werden frühere Versionen von Opera, das erst seit der Version 5 in einer kostenlosen Variante vorliegt, in Selfhtml ignoriert. Soweit ich Informationen zu unterstützten Technologien älterer Versionen bei Opera (www.opera.com) gefunden habe, sind diese angegeben.

<sup>18</sup> Dieses und die weiteren Programmbeispiele sind in Pseudocode geschrieben, der sich an Javascript anlehnt.

```

// Entferne alles vor der Adresse
var newline = line.substring(start + 13);

// Entferne alles nach der Adresse
newline = newline.substring(0,
                             newline.indexOf('"'));

return newline;
}

// kein Mailverweis gefunden
else return null;
}

```

Mit nur effektiv sieben Zeilen Programmcode<sup>19</sup> wird aus einer übergebenen Zeile eine Emailadresse herausgezogen. Damit ist diese Methode nicht sicher.

Mein Emailadresssuchprogramm Gweax umfasst einen Filter, der genau das tut: Er erkennt Emailadressen, die über einen Standardmailverweis in einer Internetseite eingebettet sind.

### *Verschleierung der Emailadresse*

Die ältesten Methoden zum Schutz einer Emailadresse versuchen, die Adresse so zu verschleiern, dass sie nicht als solche erkannt wird. Dabei besteht jedoch die Gefahr, dass nicht nur Spambots, sondern auch legitime Nutzer die Adresse nicht mehr erkennen.

Als problematisch wird sich dabei herausstellen, dass, sofern in dem URI `mailto` auftaucht, die Emailadresse mit relativ wenig Aufwand wieder ermittelt werden kann. Darum verzichten einige Methoden auf die Funktionalität und stellen die Adresse ohne Verweis dar.

### Maskierung des @-Zeichens

Die ersten Methoden zur Verschleierung einer Emailadresse haben versucht, durch Maskierung des @-Zeichens das Vorhandensein einer Emailadresse zu verbergen. Ein Emailverweis sah dann so aus:

<sup>19</sup> Natürlich kann diese Funktion bezüglich der Zeilenanzahl noch optimiert werden; zur besseren Lesbarkeit habe ich hier darauf verzichtet. Auch ist diese Funktion nicht ausgereift: bei mehreren Mailverweisen innerhalb einer Zeile würde nur die erste erkannt.

```
<a href="mailto:name(at)example.com">
  name(at)example.com
</a>
```

Statt (at) werden auch (AT) oder (at-sign) verwendet, auch in eckigen oder geschweiften Klammern.

Im Browser sieht dieses Beispiel dann aus wie folgt (Abbildung 2):

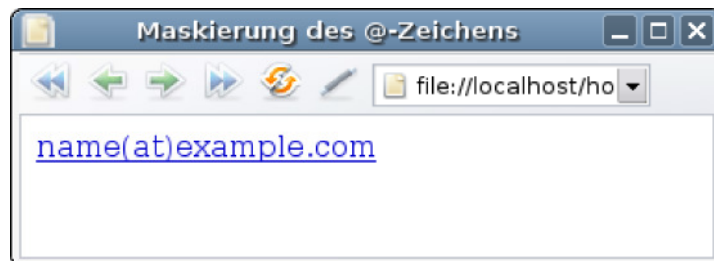


Abbildung 2: Maskierung des @-Zeichens

### ***Funktionalität***

Bei Aktivierung dieses Mailverweises öffnet sich tatsächlich das Emailprogramm mit einer neuen Mail und der im Verweis angegebenen Emailadresse. Auch beim Überfahren mit der Maus wird das Verweisziel in der Statuszeile angezeigt. In beiden Fällen ist die Adresse allerdings nicht gültig. Somit scheitert der Versand der Email, sofern der Nutzer die Adresse nicht manuell korrigiert.

### ***Gebrauchstauglichkeit***

Hier werden keine anderen HTML-Elemente verwendet als bei der Standardmethode auch, daher kann diese Methode ebenfalls in allen heute gebräuchlichen Browsern dargestellt werden. Allerdings ist nicht sichergestellt, dass alle Nutzer erkennen, dass und auf welche Weise die Emailadresse korrigiert werden muss.

In der Untersuchung haben von 51 Teilnehmern, die hier eine Adresse erkannt haben, zwei die nötige Korrektur nicht oder falsch vorgenommen.

Formal gesehen liegt hier aber ein Verstoß gegen diverse Standards vor: Die Emailadresse ist nicht nach RFC822 gebildet, damit liegt kein gültiger URI nach RFC 1630 vor, und damit kein gültiger Verweis nach RFC1866. Es ist also

theoretisch möglich, dass ein Browser das Dokument fehlerhaft anzeigt. In der Praxis jedoch sind alle Browser fehlertolerant genug, um nicht darüber zu stolpern.

### *Sicherheit*

Angenommen, ich übergebe dem Filter, der die Standardmethode erkennt, eine Zeile, in der eine auf diese Art maskierte Adresse enthalten ist, wie lautet dann das Ergebnis? Die Zeile enthält einen Emailverweis, also versucht der Filter, die Adresse zu extrahieren. Das Ergebnis ist dann `name(at)example.com`. Dabei handelt es sich noch nicht um die korrekte Adresse, dies kann aber mit einer weiteren Zeile, die über reguläre Ausdrücke die Maskierung rückgängig macht, gelöst werden:

```
newline = newline.replace  
          (/(\(|\[[|\{)(at|at-sign)(\)|\]|\})/i, '@');
```

Durch diese Zeile wird ein Vorkommen von `at` oder `at-sign` in runden, eckigen oder geschweiften Klammern unabhängig von Groß- oder Kleinschreibung durch das `@`-Zeichen ersetzt. Wenn dazu eine einzige zusätzliche Zeile ausreicht, ist die Sicherheit nicht gewährleistet.

In Gweax werden solche Adressen durch den `MaskedAtFilter` erkannt, der wie oben beschrieben die Maskierung des `@`-Zeichens rückgängig macht.

### Hinzufügen von adressfremden Bestandteilen

Diese Methode ist mir zum ersten Mal im Usenet begegnet. Das Usenet zeichnet sich dadurch aus, dass Nachrichten in Form einer Email verfasst und dann auf einem Newsserver aufbewahrt werden. Das impliziert natürlich, dass eine Absenderemailadresse angegeben werden muss, und es entspricht den Gepflogenheiten des Usenet, dabei eine gültige zu verwenden. Um sich nun vor Spam zu schützen, ist es üblich, die Emailadresse zu verfremden, indem man an beliebiger Stelle einen Zusatz einfügt, etwa `NOSPAM` oder `REMOVE`. Vor dem Senden einer Email an eine derart maskierte Adresse müssen diese Bestandteile dann entfernt werden. Ein Emailverweis mit dieser Methode sieht dann so aus:

```
<a href="mailto:name@REMOVE.example.com">
  name@REMOVE.example.com
</a>
```

oder auch so:

```
<a href="mailto:nameNOSPAM@example.com">
  nameNOSPAM@example.com
</a>
```

Im Browser wird das dargestellt wie in Abbildung 3:

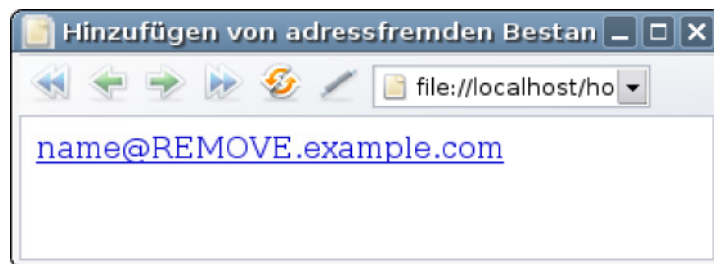


Abbildung 3: Hinzufügen von adressfremden Bestandteilen

### ***Funktionalität***

Es handelt sich hierbei um einen korrekten Mailverweis mit einer syntaktisch korrekten Emailadresse. Diese Methode verhält sich also wie die Standardmethode und ist somit voll funktional.

### ***Gebrauchstauglichkeit***

Diese Methode benutzt keine besonderen Elemente zur Darstellung; ein Browser, der einen normalen Mailverweis darstellen kann, kann auch Emailadressen dieser Art darstellen.

Schwierig wird es bei der Erkennung der Adresse: Erkennen die Nutzer, dass es sich noch nicht um die korrekte Adresse handelt, dass sie etwas entfernen müssen und was sie entfernen müssen? Im Usenet ja, außerhalb nicht: Von 51 Teilnehmern haben 24 nicht erkannt, dass sie den Zusatz hätten entfernen müssen.

## *Sicherheit*

Die in dieser Methode angegebene Mailadresse ist syntaktisch korrekt, ein Spambot würde sie als Emailadresse erkennen. Das ist jedoch kein Problem, denn solange er die Verfremdung nicht erkennt, ist die Adresse geschützt. Für Spammer, die ihre Mails durchaus an mehrere Millionen Empfänger schicken, ist der Aufwand viel zu groß, einzelne Adressen zu überprüfen.

Ist eine automatische Erkennung der zu entfernenden Bestandteile möglich? Sofern man nur nach REMOVE und NOSPAM sucht, gewiss; dazu müsste man dem schon oben verwendeten Script eine Zeile hinzufügen:

```
newline = newline.replace(/(NOSPAM|REMOVE)/, '');
```

Dadurch würde aus einer Emailadresse NOSPAM oder REMOVE entfernt.<sup>20</sup>

Diese Erkennungsmethode wird ineffizient, wenn andere Variationen eingebracht werden: REMOVEME, REMOVETHIS, DELETE etc., durchaus auch in anderen Sprachen, je nachdem, woher der Adressinhaber kommt. Ein aggressiver Spammer könnte zwar eine lange Liste von möglichen zu entfernenden Bestandteilen erstellen und alle Adressen danach filtern, ob sich dieser Aufwand aber lohnt, ist fraglich.

Einfacher ist es auszunutzen, dass ja auch der menschliche Nutzer diese Bestandteile erkennen muss. Um das zu gewährleisten, werden sie meist ausschließlich in Großbuchstaben gesetzt, und sind deshalb einfach zu erkennen, da die meisten Emailadressen nur aus Kleinbuchstaben bestehen. Dieses Muster dient als Anknüpfungspunkt für die automatische Erkennung.

```
newline = newline.replace(/[A-Z]{2,}/, '');
```

Diese Zeile entfernt jedes Vorkommen von mindestens zwei Großbuchstaben hintereinander, einzelne Großbuchstaben werden als Bestandteil der Adresse angenommen.

Somit kann auch diese Methode nicht als sicher angesehen werden.

---

<sup>20</sup> Wenn man diese Funktion auf das erste Beispiel anwendet (name@REMOVE.example.com), so wird nicht die korrekte Adresse extrahiert, sondern name@example.com. Dieser Syntaxfehler (und andere, die bei der Erkennung anderer Methoden entstehen können) kann jedoch leicht durch ein zusätzliches Skript entfernt werden.

Der Filter in Gweax, der solche Adressen korrigiert, die einen Zusatz in Großbuchstaben enthalten, heißt AddedCapitalsFilter. Mögliche Syntaxfehler (siehe Fußnote 20) werden dort ebenfalls behoben.

## Maskierung mit Unicode

Diese Methode geht davon aus, dass Spambots Webseiten nach bestimmten Schlagworten durchsuchen, beispielsweise `mailto` oder `@` und diese als Hinweis auf eine Emailadresse sehen. Kann man nun verhindern, dass diese Schlagworte auf einer Webseite auftauchen, gehen die Spammer leer aus.

Eine Möglichkeit, dies zu tun, besteht darin, einzelne Zeichen zu maskieren. In HTML können Buchstaben durch z.B. `&#97;` kodiert werden, die Zahl steht dabei für den Unicodewert eines Zeichens, in diesem Fall für ein kleines `a`. URIs können ähnlich kodiert werden, die Folge `%61` steht ebenfalls für ein kleines `a` (`61hex = 97dec`). Ein Mailverweis sieht dann so aus:

```
<a href="%6D%61%69%6C%74%6F%3A%6E%61%6D%65%40%65%78%61%6D%70%6C%65%2E%63%6F%6D">
  &#110; &#97; &#109; &#101; &#64; &#101; &#120; &#97; &#109;
  &#112; &#108; &#101; &#46; &#99; &#111; &#109;
</a>
```

Ein Browser wandelt diese Zeichenfolge bei der Darstellung um in [name@example.com](mailto:name@example.com) (Abbildung 4)



Abbildung 4: Maskierung mit Unicode

Leider schlägt der Versuch teilweise fehl. Wird auch das `mailto` in dem URI kodiert, so erkennt der Browser den Verweis nicht mehr als Emailverweis, sondern behandelt ihn wie einen lokalen Verweis – die Funktionalität geht verloren. Bleibt es stehen, ist das für Spambots wiederum ein Indiz für eine Emailadresse.

### ***Funktionalität***

Der Browser dekodiert beim Aufbau der Seite den Unicodewert in das entsprechende Zeichen<sup>21</sup>, für den Nutzer stellt sich diese Methode also transparent dar, es besteht kein Unterschied zu einem Standardmailverweis.

### ***Gebrauchstauglichkeit***

Die Verwendung von Unicode in HTML ist seit der Version 4.0 (Dezember 1997) definiert, wurde teilweise aber schon vorher von einigen Browsern umgesetzt. Alle aktuellen Browser sind in der Lage, diese Zeichen darzustellen. Durch die Transparenz dieser Methode bemerkt der Nutzer nichts von der Verschleierung. Bei der Untersuchung haben von 63 Teilnehmern (zwei haben keine Adresse ermittelt) alle die korrekte Adresse erkannt (bis auf einige Tippfehler, die nicht auf die Methode zurückzuführen sind).

Somit ist diese Methode voll gebrauchstauglich.

### ***Sicherheit***

Wenn Browser die Umwandlung von Unicode beherrschen, erscheint es logisch, dass auch Spambots das können. In der Tat ist das sehr einfach, eine Zeile reicht dazu bereits aus.

```
var newline = unescape(line);
```

Der Befehl `unescape` wandelt alle druckbaren Unicodezeichen der Form `%61` um in das entsprechende Ascii-Zeichen. Eine Funktion, die gleiches für Zeichen der Form `&#97;` tut, ist einfach zu programmieren:

```
function unescapeHTML(line)
{
    var newline = '';
    for (i = 0; i < line.length; i++)
    {
        // Suche nach Zeichenfolge &#
        if (line.charAt(i) == '&'
            && line.charAt(i+1) == '#')
```

---

<sup>21</sup> Opera dekodiert das Zeichen `%40` (Unicode für `@`) in der Adresse nicht direkt. Wird die Adresse z.B. in der Statuszeile angezeigt, so lautet sie `name%40example.net`. Wird der Verweis aktiviert, so öffnet sich das Emailprogramm hingegen mit der richtig dekodierten Adresse.

```

    {
        // Extrahieren des Zahlwertes
        var unicodechar = line.substring(i+2,
                                         line.indexOf(';', i));
        newline += String.fromCharCode(
                                         parseInt(unicodechar));
        i+= unicodechar.length+2;
    }
    else newline += line.charAt(i);
}
return newline;
}

```

Diese Methode ist somit nicht sicher.

Selbstverständlich erkennt auch Gweax mit Unicode kodierte Emailadressen. Bevor eine Zeile aus dem Quelltext den Filtern übergeben wird, wird sie dekodiert. Dazu habe ich zwei Methoden geschrieben, `unescapeURL()` und `unescapeHTML()`.

## Adresse als Grafik

Wenn es nicht möglich ist, auf das `mailto` zu verzichten oder das `@`-Zeichen wirkungsvoll zu maskieren, dann müssen radikalere Methoden her. Wenn ich bereit bin, auf die Funktionalität zu verzichten, dann kann ich ausnutzen, dass Spambots Maschinen sind, die nur eingeschränkt fähig sind, Bilder zu erkennen.

Ersetzt man die Emailadresse durch eine Grafik, die selbige darstellt, haben Spambots keinen Anhaltspunkt, dass es sich um eine Emailadresse handelt. Der Versuch der automatischen Bilderkennung muss schon daran scheitern, dass der überwiegende Teil der Grafiken auf Webseiten *keine* Emailadressen enthält. Um aus zehntausend Grafiken eine Adresse zu ziehen, lohnt sich der Aufwand der Bilderkennung nicht.

Eine Grafik in HTML einzufügen geschieht über das `img`-Tag:

```



```

Im Browser sieht das so aus (Abbildung 5):



Abbildung 5: Adresse als Grafik, zum Vergleich darunter Adresse als Text

### ***Funktionalität***

Um keinen Anhaltspunkt für Spambots zu geben, wurde hier bewusst darauf verzichtet, einen Emailverweis zu verwenden. Die Funktionalität ist somit ganz verloren gegangen.

### ***Gebrauchstauglichkeit***

Die Einbettung von Grafiken in Webseiten ist seit der HTML-Version 2.0 möglich, ein Standard, der von allen relevanten Browsern umgesetzt wird. Aus der Verwendung von Grafiken resultieren allerdings auch einige Probleme: Textbrowser können in der Regel keine Grafiken darstellen; Nutzer solcher Programme können somit keine Emailadresse erkennen. Weiterhin sind Grafiken nicht wie Text skalierbar, Nutzer mit Sehschwächen können somit Schwierigkeiten haben, die Adresse zu lesen.

Problematisch ist auch, dass Text in Grafiken nicht selektierbar ist. Gerade bei Methoden, die auf die Funktionalität verzichten, sollte dem Nutzer die Möglichkeit gegeben werden, durch Kopieren und Einfügen die Adresse leicht in das Mailprogramm zu übertragen. Das ist hier leider nicht möglich.

Diese Methode ist somit zwar meist darstellbar, widerspricht aber dem Gedanken der Barrierefreiheit.

Bei der Untersuchung haben zwei von 65 Teilnehmern keine Adresse erkannt, ein weiterer hat die Adresse der Grafik angegeben. Von den übrigen 62 Teilnehmern haben 60 die Adresse richtig erkannt und fehlerfrei abgetippt, bei

einem Teilnehmer kam es zu einem Tippfehler. Trotz der Bedenken wegen der Barrierefreiheit scheint die Methode in der Praxis gut zu funktionieren.

### *Sicherheit*

Diese Methode baut darauf, dass es keine Indizien für eine Emailadresse gibt. Ein einfacher Scan nach den übliche Schlagworten wie `mailto` oder dem `@`-Zeichen führt zu keinem Ergebnis. Alle Grafiken einer Webseite mit Hilfe von Bilderkennungssoftware nach Emailadressen zu durchsuchen, ist schlicht zu aufwendig. Sofern man in den Attributen des `Imagetags` auf Wörter verzichtet, die auf eine Emailadresse hinweisen, ist die Grafik ausreichend geschützt.

In dem obigen Beispiel ist der Bildname sowie der Alternativtext unter Umständen ungünstig gewählt. Wenn ein Spambot nach Grafiken sucht, die bestimmten Mustern entsprechen (z.B. geringe Höhe, wesentlich breiter als hoch, Alternativtext oder Dateiname, der „mail“, „adresse“ oder ähnliches enthält), würde er hier fündig.

Dies ist die erste Methode, die mein Programm Gweax nicht überwindet. Die Schwierigkeit der Bilderkennung ist eine Grenze, die zu überwinden ich nicht versucht habe, da der Aufwand den Rahmen dieser Arbeit übersteigen würde.

### Entzerren der Adresse

Noch eine weitere Methode verzichtet auf die Funktionalität, um Spambots keine Anhaltspunkte für das Vorhandensein einer Emailadresse zu geben. Statt einer Grafik und der damit verbundenen Schwierigkeiten wird die Adresse in Textform angegeben, wobei die einzelnen Bestandteile ausgeschrieben werden. Statt des `@`-Zeichens schreibt man `at`, statt eines Punktes `dot`. Eine Emailadresse wird dann wie folgt kodiert:

```
name at example dot com
```

Kompliziertere Adressen werden ebenso entzerrt:

`john.doe@sales.example.com` würde zu

```
john dot doe at sales dot example dot com.
```

Im Browser sieht das so aus (Abbildung 6):



Abbildung 6: Entzerren der Adresse

### ***Funktionalität***

Wie bei der vorigen Methode wird zugunsten der Sicherheit auf die Funktionalität verzichtet.

### ***Gebrauchstauglichkeit***

Hier werden keine besonderen Techniken verwendet, eine Darstellung ist auch in Textbrowsern und auf Mobilgeräten problemlos möglich. Die Frage ist hierbei vielmehr, ob die Nutzer die Adresse erkennen. Wenn sie das tun, können sie den entsprechenden Textbereich selektieren, kopieren und im Emailprogramm einfügen, wo der Text noch in die korrekte Emailadresse umgewandelt werden muss.

Problematisch bei dieser Methode erscheint, dass in unterschiedlichen Sprachen das @-Zeichen und vor allem der Punkt unterschiedlich benannt werden. Ein Nutzer aus dem englischsprachigen Raum wird eine Adresse der Form `name at example dot com` leichter erkennen als ein deutschsprachiger. Umgekehrt kann ersterer mit `name klammeraffe example punkt com` wohl kaum etwas anfangen.

Schwierig ist auch, Anfang und Ende der Adresse zu bestimmen, da durch die eingefügten Leerzeichen die Adresse nicht als ein zusammengehöriger Block wahrgenommen wird. Unter Umständen sind Nutzer nicht in der Lage, die vollständige Adresse zu erkennen.

Wie sind die Teilnehmer der Untersuchung also damit zurecht gekommen? Drei von 65 Teilnehmern haben kein Ergebnis ermittelt. Von den übrigen 62 haben sieben die korrekte Adresse nicht erkannt: Sechs haben nicht erkannt,

wo die Adresse beginnt, einer hat nicht erkannt, dass `dot` hätte umgewandelt werden müssen.

Um diese Methode gebrauchstauglich zu machen, ist eine deutliche Hervorhebung der Adresse vom übrigen Inhalt der Seite nötig, beispielsweise durch eine andere Text- oder Hintergrundfarbe oder eine andere Schriftart. Das würde es den Spambots aber wieder erleichtern, die Adresse zu erkennen.

### *Sicherheit*

Wie bei der vorigen Methode finden sich hier keine direkten Indizien für eine Emailadresse, weder ein `mailto` noch ein `@`-Zeichen. Eine Suche nach dem Substitut `at` ist zwecklos, da `at` als englisches Wort sehr verbreitet ist (verschiedene Suchmaschinen finden mehr als eine Milliarde Seiten zu dem Suchbegriff „at“). Eine kombinierte Suche nach den Begriffen `at` und `dot` innerhalb einer Zeile erscheint erfolgversprechender, allerdings bleibt das schon oben diskutierte Problem der Begrenzung der Emailadresse bestehen. Dies zu lösen ist für menschliche Nutzer deutlich einfacher als für ein Programm.

Diese Methode ist somit ausreichend sicher.

Auch Gweax scheitert an dieser Methode. Ich habe es nicht geschafft, ein Methode zu programmieren, die die Grenzen einer derart dargestellten Adresse zuverlässig erkennt.

### *Methoden, die Javascript verwenden*

Wir haben gesehen, dass Methoden, die einen Mailverweis mit `mailto` verwenden, im Allgemeinen unsicher sind. Verzichtet man auf den Verweis und stellt die Emailadresse davon losgelöst dar, muss man Einbußen in der Benutzerfreundlichkeit hinnehmen.

Mit Hilfe von Javascript versuchen einige Methoden, dieses Problem zu lösen. Javascript ist eine Ergänzungstechnologie zu HTML, mit der einer Seite zum Beispiel Dynamik hinzugefügt werden kann.

## Javascript: Text dynamisch erzeugen

Eine Möglichkeit der Dynamik auf Webseiten besteht darin, dass Text generiert werden kann, beispielsweise beim Laden der Seite.

Das wird nun in folgender Methode ausgenutzt, um einen Emailverweis zu erzeugen:

```
var user = "name";
var host = "example";
var tld = "com";
document.write("<a href=\"mailto:" + user + "@" + host
               + "." + tld + "\">" + user + "@" + host
               + "." + tld + "</" + "a>");
```

Das Ergebnis im Browser sieht aus wie ein gewöhnlicher Mailverweis (Abbildung 7). Die Emailadresse wird aufgespalten in ihre einzelnen Bestandteile, die in drei Variablen gespeichert werden. In dem Javascriptbefehl `document.write` werden die einzelnen Bestandteile wieder zusammengesetzt, wobei erst hier das `@`-Zeichen eingesetzt wird. Ein Scannen nach dem `@`-Zeichen führt zu keiner Adresse.

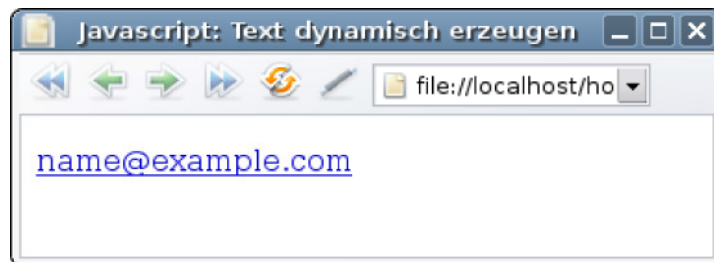


Abbildung 7: Javascript: Text dynamisch erzeugen

Die Mächtigkeit dieser Methode zeigt sich darin, dass mit nur geringen Änderungen das Skript bei gleicher Funktion ganz anders aussieht – und so die automatische Erkennung erschwert wird:

```
var v1 = "name", v2 = "example", v3 = "com";
var v4 = "macvbnilasdfto";
document.write("<a href=\"\" + v4.substring(0,2)
               + v4.substring(6,8) + v4.substring(12)
               + v1 + String.fromCharCode(64) + v2
               + "." + v3 + "\">" + v1
               + String.fromCharCode(0x40) + v2 + "."
               + v3 + "</" + "a>");
```

In diesem Beispiel beinhaltet der String in der Variablen `v4` unter anderem `ma`, `il` und `to`. Beim Erzeugen des Verweises werden die einzelnen Teilstrings zusammengehängt. Das `@`-Zeichen wird über die Methode `String.fromCharCode` erzeugt, einmal angegeben als Dezimal-, einmal als Hexadezimalwert.

Die Anzahl der Möglichkeiten zur Verschleierung sind so groß, dass jede einzelne Adresse individuell verschleiert werden kann.

### ***Funktionalität***

Durch das Skript wird ein Emailverweis erzeugt, der sich nicht von der Standardmethode unterscheidet. Die Methode ist somit voll funktional.

### ***Gebrauchstauglichkeit***

Javascript existiert in verschiedenen Versionen, die von den unterschiedlichen Browsern nicht immer vollständig unterstützt werden. Der Befehl `document.write` stammt jedoch aus der Version 1.0 (aus dem Jahr 1995), die von Netscape seit der Version 2.0 und vom Internet Explorer seit der Version 3.0 unterstützt wird. Von daher wird hier eine Technik eingesetzt, die von allen aktuellen Browsern verstanden wird.

Dennoch ist diese Methode nicht ohne Einschränkung gebrauchstauglich. Abgesehen von den Browsern, die Javascript nicht unterstützen, gibt es Nutzer, die aus Sicherheitsgründen Javascript deaktiviert haben. Immer wieder finden sich in Browsern Sicherheitslücken im Zusammenhang mit Javascript, so dass gelegentlich empfohlen wird, auf Javascript zu verzichten. Von den 67 Teilnehmer der Untersuchung hatten jedoch alle 67 Javascript aktiviert, in der Praxis scheint die Verwendung von Javascript möglich zu sein, ohne Einbußen hinnehmen zu müssen<sup>22</sup>.

Alle fünfzig gewerteten Teilnehmer der Untersuchung haben die Adresse fehlerfrei erkannt. Das ist ein sehr gutes Ergebnis.

---

<sup>22</sup> Die Untersuchung ist wegen der geringen Teilnehmerzahl jedoch keineswegs repräsentativ. Diese Aussage ist also mit Vorsicht zu genießen und sollte vor der Verwendung dieser Methode bei der entsprechenden Zielgruppe überprüft werden.

Nutzern ohne Javascript wird bei dieser Methode nichts angezeigt, in der Regel nicht mal ein Hinweis, dass ein Skript nicht ausgeführt wurde. Dieses Problem kann man halbwegs umgehen, indem man einen `noscript`-Bereich hinzufügt. Der Inhalt eines solchen Bereiches wird nur dann angezeigt, wenn Javascript nicht aktiviert ist. Somit könnte man in diesem Bereich die Emailadresse angeben, auf eine Art, die die Sicherheit dieser Methode nicht senkt.

### ***Sicherheit***

Diese Methode baut primär auf der Annahme auf, dass Spambots über keinen Javascriptinterpreter verfügen, also nicht in der Lage sind, den Befehl `document.write()` auszuführen oder zu verstehen. Im Quelltext finden sich keine Schlagwörter, kein `mailto`, kein `@`-Zeichen oder ähnliches. Ohne Anhaltspunkt für eine Emailadresse ist das automatische Auslesen unmöglich.

Angenommen, ein Spambot möchte auch mit Javascript kodierte Emailadressen erkennen. Er müsste nun jedes Skript ausführen und das Ergebnis analysieren. Hier gilt wie für Grafiken auch: der Anteil an Skripten, die eine Emailadresse verbergen ist sehr gering; ein Großteil der Rechenzeit würde sinnlos verbraucht werden. Aus diesem Grund erkennt auch Gweax diese Emailadressen nicht.

### **Javascript: *linkto\_uncryptmailto()***

Javascript bietet also die Möglichkeit, die Emailadresse wirkungsvoll zu schützen. Die folgende Methode, die unter anderem auf den Seiten des Fachbereichs Mathematik der Universität Paderborn<sup>23</sup> genutzt wird, verwendet auch Javascript, verfolgt aber einen anderen Ansatz. Leider ist sie dabei weniger sicher, wie ich bei der Sicherheitsanalyse aufzeigen werde. Der Ursprung dieser Methode ist unklar, möglicherweise stammt sie von `jumk.de`<sup>24</sup>.

Bei dieser Methode wird ein Anker definiert, der auf ein Javascript verweist. Bei Aktivierung des Ankers wird nun ein Skript ausgeführt, das einen URI erzeugt, der die Emailadresse enthält:

---

<sup>23</sup> <http://www2.math.uni-paderborn.de/personen.html>

<sup>24</sup> <http://jumk.de/nospam/>

```
<a href="javascript:linkto_unencryptmailto('nbjmup;obnfAfybnqmf/dpn') ">Sende Mail</a>.
```

Im Browser sieht der Verweis dann aus wie in Abbildung 8:

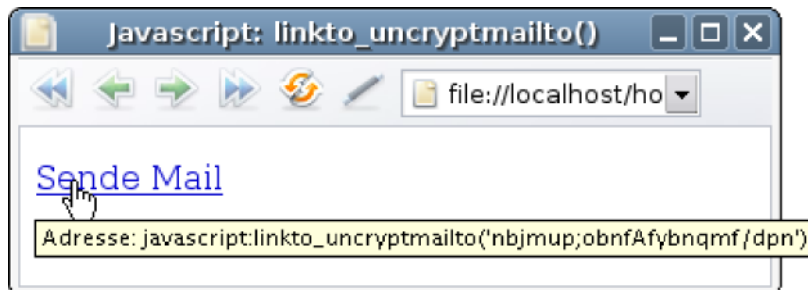


Abbildung 8: Javascript: linkto\_unencryptmailto()

Anders als bei der vorigen Methode kann hier die Emailadresse nicht als Verweistext mit angegeben werden (sie wäre sonst nicht geschützt). Der Funktion `linkto_unencryptmailto` wird ein Parameter übergeben, der die Adresse in kodierter Form enthält. Die Kodierung ist denkbar einfach, jedes Zeichen wird durch das im Ascii-Code nachfolgende ersetzt: ein „a“ wird zum „b“, ein „m“ zum „n“, ein „@“ zum „A“, ein Punkt zum Schrägstrich und ein Doppelpunkt zum Semikolon. Alles, was das Skript nun bewirkt, ist das Zurückwandeln des Codetextes in die Emailadresse sowie das Aufrufen des so gebildeten URI.

```
function linkto_unencryptmailto(s)
{
    location.href=unencryptmailto(s);
}

function unencryptmailto(s)
{
    var n=0;
    var r="";
    for(var i=0; i < s.length; i++)
    {
        n=s.charCodeAt(i);
        if (n>=8364) n = 128;
        r += String.fromCharCode(n-1);
    }
    return r;
}
```

### ***Funktionalität***

Wenn Javascript aktiviert ist, verhält sich dieser Verweis bei Aktivierung wie ein Standardemailverweis: er ruft einen URI auf, der einem Mailverweis entspricht. Der Browser öffnet das Emailprogramm mit der entsprechenden Adresse. Beim Überfahren des Verweises mit der Maus dagegen wird nicht die Emailadresse angezeigt, sondern der Name der auszuführenden Javascriptfunktion. Diese Methode ist somit nur teilweise funktional.

### ***Gebrauchstauglichkeit***

Diese Methode verwendet Javascriptsprachelemente, die seit der Version 1.0 bzw. 1.2 definiert sind und von den aktuellen Browsern unterstützt werden. Wie bei der vorigen Methode gilt auch hier, Nutzer, deren Browser kein Javascript unterstützen, werden ausgeschlossen.

Im Gegensatz zu der vorigen Methode wird aber angezeigt, dass hier theoretisch die Möglichkeit besteht, eine Email zu versenden. Wer diese Methode nicht kennt, hat kaum eine Möglichkeit, die Adresse selbst zu erkennen.

Nur 37 von 65 Teilnehmern der Untersuchung haben die Adresse richtig erkannt, was für die Praxis viel zu wenig ist. Selbst wenn man nur die Teilnehmer berücksichtigt, die auch bei anderen Methoden damit zurecht gekommen sind, dass die Adresse nicht im Klartext angegeben wurde, bleibt diese Methode unbrauchbar, denn dann konnten 37 von 45 Teilnehmern die Adresse erkennen. Die Fehlerrate liegt mit knapp 18% deutlich zu hoch.

### ***Sicherheit***

Grundsätzlich gilt hier wieder, dass es zu aufwendig ist, alle Skripte einer Seite auszuführen, um einen eventuell vorhandenen Emailverweis zu entdecken. Allerdings gibt es hier einige Hinweise auf eine Adresse. Wenn diese Methode weiter verbreitet wird, lohnt es sich für Spambots, nach `linkto_uncryptmailto` zu suchen und den übergebenen Parameter auszuwerten. Durch die simple Verschlüsselung ist das denkbar einfach:

```

function extractLinkto(line)
{
  var start = line.indexOf("linkto_uncryptmailto");
  if (start == -1) return null;

  // extrahieren des Parameters
  var newline = line.substring(
    line.indexOf('(',start)+9,
    line.indexOf(')',start)-1);

  // Umwandeln in Emailadresse
  var r = "";
  for(var i=0; i < s.length; i++)
  {
    var n= s.charCodeAt(i);
    if (n >= 8364) n = 128;
    r += String.fromCharCode(n-1);
  }
  return r;
}

```

Diese Funktion überprüft, ob eine Zeile `linkto_uncryptmailto` enthält und extrahiert im positiven Fall den übergebenen Parameter, aus dem dann die Emailadresse dekodiert wird. Das ganze funktioniert, ohne dass der Spambot ein Skript ausführen muss. In dieser Version ist diese Methode nicht sicher.

In Gweax wird diese Methode durch den `LinktoUncryptFilter` überwunden, der wie oben beschrieben funktioniert.

### ***Eine Möglichkeit, die Sicherheit zu erhöhen***

Die Unsicherheit dieser Methode liegt an der schlechten Verschlüsselung. Mit nur geringen Veränderungen kann sie jedoch verbessert werden.

Eine Möglichkeit besteht darin, ein Zeichen nicht durch seinen Nachfolger zu kodieren, sondern durch das Zeichen, das im Asciiicode zwei, drei, allgemein k Stellen weiter vorkommt. Das Skript verändert sich dann wie folgt:

```

function uncryptmailto_v2(s)
{
  var key = 17; // das ist der Schlüssel
  var r = "";

```

```

for(var i=0; i < s.length; i++)
{
    var n = s.charCodeAt(i);
    if (n >= 8364) n = 128;
    r += String.fromCharCode(n - key);
}
return r;
}

```

In der Praxis sollte man den Schlüssel nicht so deutlich benennen, damit ein ausgefeilter Spambot nicht in der Lage ist, ihn zu erkennen und anzuwenden. Um das Skript noch weiter zu verbessern, könnte man statt *eines* festen Schlüssels *mehrere* verwenden (sogenannte Vigenèrverschlüsselung). Die folgende Methode macht unter anderem davon Gebrauch.

## Javascript mit Nutzerinteraktion

Die Firma Syronex<sup>25</sup> bietet eine sehr ausgefeilte Methode an, um Emailadressen wirkungsvoll vor Spambots zu schützen. Dabei wird eine Nutzereingabe benötigt, um die Emailadresse zu entschlüsseln.

Auch hier wird ein Anker definiert, der auf ein Javascript verweist. Bei Aktivierung des Ankers öffnet sich ein Fenster, in dem sich eine Grafik und ein Eingabefeld befinden (siehe Abbildung 9). Der Nutzer muss nun den in der Grafik enthaltenen Code eingeben. Darauf erstellt das Skript einen URI und ruft ihn auf. Das Emailprogramm öffnet eine neue Email mit der entsprechenden Adresse.

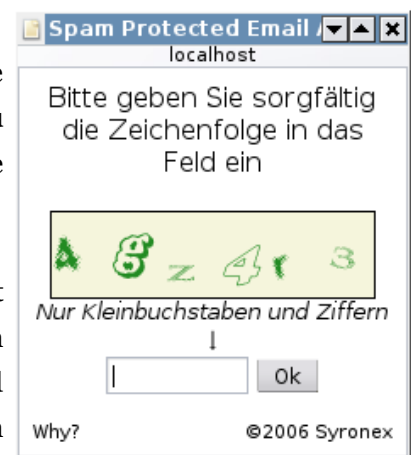


Abbildung 9: Eingabefenster für die Nutzerinteraktion

Wie funktioniert das Skript? Die Emailadresse wird verschlüsselt, indem ein Schlüssel bitweise addiert wird. Die ersten sechs Zeichen des Schlüssels entsprechen dem in der Grafik enthaltenen Code, die zusätzlichen Zeichen stehen im Skript im Klartext. Da die bitweise Addition selbstinvers ist, muss zur Entschlüsselung der Schlüssel wieder addiert werden.

<sup>25</sup> <http://w2.syronex.com/jmr/safemailto/de/>

Ich möchte hier anmerken, dass das von Syronex zur Verfügung gestellte Skript unter Opera nicht funktionsfähig war. Erst nach einer Korrektur war die Nutzung mit Opera möglich.

Syronex bietet auf seinen Seiten alles an, was zum Einsatz dieser Methode benötigt wird, der verwendete Schlüssel entsteht dabei zufällig und ist quasi einmalig.

### ***Funktionalität***

Wieder gilt: Ist Javascript aktiviert, verhält sich das Programm bei Aktivierung des Verweises letztendlich wie ein normaler Mailverweis, es wird lediglich ein Zwischenschritt eingeführt. Beim Überfahren mit der Maus hingegen wird der Name der Javascriptfunktion angezeigt. Diese Methode ist somit nur teilweise funktional.

### ***Gebrauchstauglichkeit***

Die in dieser Methode verwendeten Javascriptelemente sind Bestandteil der Version 1.0 bzw. 1.2, die in den aktuellen Browsern vollständig unterstützt werden. Wie auch bei den anderen Methoden, die Javascript verwenden, werden unter Umständen Nutzer ausgeschlossen.

Zwei zusätzliche Probleme können hierbei auftreten. Zum einen wird ein Fenster geöffnet, was eventuell von installierten Popublockern verhindert wird. Ein guter Popublocker wird registrieren, dass aufgrund einer Nutzerhandlung ein Fenster geöffnet werden soll, und dies zulassen. In der Praxis wurde das Fenster bei unterschiedlichen Systemen nicht unterdrückt (Popublocker von Opera, Webwasher und der Googletoolbar), das kann aber nicht grundsätzlich ausgeschlossen werden. Man kann aber leicht die Funktionalität des neuen Fensters in das übergeordnete verlagern und das Problem so umgehen.

Das zweite Problem ist das schwerwiegendere. Um eine automatische Bilderkennung zu verhindern, werden die Zeichen in der Grafik teilweise verzerrt, gedreht, invers und in verschiedenen Schriftarten angegeben. Dadurch kann es leicht dazu kommen, dass Zeichen nicht mehr eindeutig erkannt werden können. Eine gedrehte, verzerrte 4 kann einem A sehr ähnlich sehen, eine 9 einem g und so weiter. Die Vielzahl an Verwechslungsmöglichkeiten ist insbesondere für ältere Internetnutzer eine große Hürde, mehr noch für Sehbehinderte.

In der Untersuchung hat diese Methode sehr schlecht abgeschnitten, nur 22 von 65 Teilnehmern haben die Adresse richtig erkannt. Dreizehn Teilnehmer haben das erste Zeichen des Codes statt für eine 4 für ein A gehalten, trotz des Hinweises, dass nur Kleinbuchstaben und Ziffern eingegeben werden sollten. Dadurch wurde das erste Zeichen der Emailadresse zum Ascii-Wert 16 codiert, was als nichtdruckbares Steuerzeichen von manchen Emailprogrammen entweder als %16 dargestellt, ganz weggelassen oder durch ein leeres Kästchen □ dargestellt wurde. Zwanzig Teilnehmer haben keine Adresse erkannt, acht haben versucht, aus dem Verweistext („Mail an Clara“) die Adresse zu raten. Zwei Teilnehmer haben den Namen der Javascriptfunktion benutzt, um daraus eine Adresse zu konstruieren, natürlich falsch.

Von allen untersuchten Methoden schneidet diese in der Gebrauchstauglichkeit am schlechtesten ab. Eine Erkennungsrate von gerade einem Drittel macht diese Methode absolut unbrauchbar.

### *Sicherheit*

Schon in der Verbesserung der vorigen Methode habe ich gezeigt, dass mit der Vergrößerung des Schlüsselraumes die Sicherheit einer Javascriptmethode stark gesteigert werden kann. Das Verfahren von Syronex verwendete Verfahren entspricht einem One-Time-Pad, dem einzigen Verschlüsselungssystem, dessen Sicherheit beweisbar ist. Selbst wenn ein Spambot alle Schlüssel ausprobiert (und das sind  $36 \text{ hoch } 6 = 2.176.782.336$ ), wird er viele mögliche Adressen bekommen, aus denen er die korrekte nicht erkennen kann. Die Stärke dieses Verfahrens liegt in der erzwungenen Nutzerinteraktion, der Schlüssel liegt nirgends im Klartext vor. Die automatische Bilderkennung scheitert an den schon bekannten Gründen.

Der Anbieter dieser Methode hat ihre Sicherheit in einer empirischen Studie überprüft: Über drei Jahre hinweg wurden für diesen Zweck erstellte Emailadressen auf verschiedenen Webseiten veröffentlicht, die eine Hälfte ungeschützt, die andere Hälfte mit dem hier vorgestellten Verfahren geschützt. Jede ungeschützte Adresse erhielt Spam, jedoch keine einzige der geschützten.<sup>26</sup>

---

<sup>26</sup> <http://www.syronex.com/antispam/efficiency-study>

Die Kombination von Javascript und der Einbettung des Schlüssels in eine Grafik verhindert auch die Erkennung dieser Adressen durch Gweax.

## *Methoden mit CSS*

Cascading Style Sheets (CSS) sind eine Ergänzungstechnologie zu HTML. Der ursprüngliche Gedanke von Auszeichnungssprachen wie HTML ist die logische Auszeichnung, das heißt einem Element wird die Bedeutung innerhalb des Dokuments zugeordnet (zum Beispiel Überschriften oder Textabsätze). Die Darstellung der einzelnen Elemente wurde dem Browser überlassen, weshalb Designer eine Möglichkeit gefordert haben, die Darstellung nach ihren Wünschen beeinflussen zu können. Diese Möglichkeit ist mit CSS gegeben. Durch die strikte Trennung zwischen HTML (Struktur) und CSS (Darstellung) können auch Browser ohne CSS-Unterstützung die Seite darstellen..

Die Version 1.0 wurde im Jahr 1996 veröffentlicht und ist mittlerweile von allen aktuellen Browsern annähernd vollständig umgesetzt. Im Jahr 1998 erschien die Version 2.0, deren Eigenschaften in aktuellen Versionen von Opera und Firefox mit wenigen Ausnahmen unterstützt werden, in der aktuellen Version des Internet Explorers jedoch in deutlich geringerem Umfang. Textbrowser wie Lynx unterstützen CSS in der Regel gar nicht. Darum sind die hier vorgestellten Methoden eher als Spielerei oder Proof-of-concept zu sehen.

## **CSS-Eigenschaft `direction:rtl`**

Eine Eigenschaft von CSS besteht darin, die Leserichtung eines HTML-Bereiches anzugeben. Das ist dazu gedacht, Webseiten in Sprachen wie Arabisch oder Hebräisch darzustellen, in denen von rechts nach links geschrieben wird.

Diese Eigenschaft kann man nun ausnutzen, um die Emailadresse zu verschleiern. Eingeführt wurde diese Möglichkeit durch Stu Nicholls auf seiner Seite über CSS-Experimente<sup>27</sup>. Er selbst schreibt jedoch dazu:

---

<sup>27</sup> <http://www.cssplay.co.uk/menu/email.html>

„This was never intended to be a *SERIOUS* contender for email hiding. It didn't work in Opera and, so people keep telling me, it doesn't work in Safari. It was only intended as a demonstration into backwards writing using css.“

Durch

```
<span style="direction:rtl; unicode-bidi:bidirectional-override">moc.elpmaxe@eman</span>
```

wird ein Bereich definiert, in dem die Schreibrichtung von rechts nach links geht. Die Emailadresse wird nun rückwärts im Quelltext notiert, so dass Browser sie vorwärts, also korrekt darstellen (Abbildung 10).

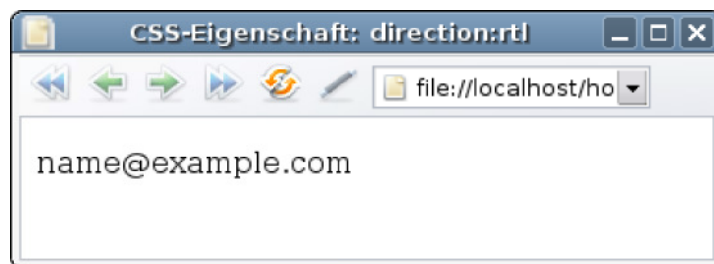


Abbildung 10: CSS-Eigenschaft *direction:rtl*

### ***Funktionalität***

Hier wird nur die Emailadresse dargestellt, ohne einen Mailverweis. Auf die Funktionalität wurde zugunsten der Sicherheit verzichtet.

### ***Gebrauchstauglichkeit***

Die verwendete CSS-Eigenschaft entstammt dem Standard CSS 2.0 (Mai 1998) und wird von vielen aktuellen Browsern umgesetzt, nicht jedoch von Safari.

Eine interessante Nebenwirkung zeigt sich, wenn man die Adresse selektiert, kopiert und dann versucht, sie z.B. im Mailprogramm einzufügen: Die Adresse wird dann wie im Quelltext rückwärts dargestellt. Ein Nutzer müsste dann per Hand die Adresse korrigieren – einfacher erscheint da, von vonherein die Adresse abzutippen.

In Browsern, die kein CSS unterstützen, wird der Bereich angegeben wie im Quelltext: rückwärts. Das kann dazu führen, dass der Nutzer die Emailadresse nicht oder nur schwer erkennt.

Wie konnten die Teilnehmer der Untersuchung damit umgehen? Immerhin 53 haben die Adresse richtig erkannt, sechs Teilnehmer haben die Adresse rückwärts angegeben, was bei genauerer Betrachtung vermutlich aufgefallen wäre<sup>28</sup>. Somit ist die Methode mit Einschränkungen gebrauchstauglich.

### *Sicherheit*

Ein Spambot, der eine Webseite nach dem Vorkommen des @-Zeichens durchsucht, wird hier fündig. Bei einer Überprüfung, ob die Adresse korrekt gebildet wurde, wird ein Fehler festgestellt, dem Domainpart fehlt ein Teil<sup>29</sup>. Wenn diese Methode große Verbreitung findet, werden Spambots in dem Fall sicher den Rückwert untersuchen und so eine korrekte Adresse entdecken:

```
function checkBackwards(pattern)
{
  if (! isValidEmailAddress(pattern) &&
      isValidEmailAddress(pattern.reverse()))
    return pattern.reverse();
  else return null;
}
```

Da die Sicherheit aber nicht auf der geringen Verbreitung basieren darf, ist das Rückwärtsschreiben keine geeignete Methode.

In Gweax habe ich dazu den BackwardMaskingFilter programmiert, der den Rückwert einer Zeile nach Emailadressen durchsucht.

---

28 Der Fehler ist wohl durch das Problem entstanden, dass bei Kopieren und Einfügen die Adresse rückwärts dargestellt wird. Die bei diesen Teilnehmern verwendeten Browser sind in der Lage, die Adresse wie gewünscht darzustellen.

29 Durch eine spezielle Emailadresse, deren Localpart dem Rückwert einer möglichen Domain entspricht, kann ein Spambot getäuscht werden (z.B. die Adresse `moc.name@example.com` wird dann durch `moc.elpmaxe@eman.com` dargestellt). Im Allgemeinen ist das jedoch nicht der Fall.

## CSS-Pseudoklasse :after

Mit Hilfe der Pseudoklassen :after und :before können einem HTML-Bereich Inhalte voran- oder nachgestellt werden. Wikipedia<sup>30</sup> verwendet das beispielsweise, um aus Wikipedia ausgehende Verweise zu kennzeichnen, indem eine symbolhafte Grafik nachgestellt wird.

Diese Eigenschaft lässt sich dazu nutzen, eine Emailadresse zu verschleiern. Vorgestellt wurde diese Methode in Grundzügen von Lim Chee Aun im Jahr 2003<sup>31</sup>, eine deutlich bessere Variante wurde im Januar 2006 von Ned Daze vorgestellt<sup>32</sup>, die ich hier in Details verbessert habe. Diese Methode nutzt CSS in Verbindung mit Javascript, um einen funktionalen Mailverweis zu erstellen.

Zunächst wird ein Verweis erstellt, der die Attribute title und id enthält<sup>33</sup>. Dem title-Attribut wird nun als Wert der Domainpart, dem id-Attribut der Localpart zugewiesen. Um den Bereich mit CSS formatieren zu können, wird ihm noch eine Klasse zugewiesen:

```
<a title="example.com" id="name" class="email">
  Email senden
</a>
```

Dieser Bereich wird nun mit CSS formatiert:

```
a.email:after
{ content: " <"attr(id) "@"attr(title) ">; }
```

Dadurch wird dem Verweis der Klasse `email` die in content bestimmten Elemente angehängt: ein Leerzeichen gefolgt (in spitzen Klammern) von dem Inhalt des id-Attributs, dem @-Zeichen und dem Inhalt des title-Attributs.

---

30 <http://de.wikipedia.org>

31 [http://phoenity.com/newtedge/hide\\_email\\_spambots/](http://phoenity.com/newtedge/hide_email_spambots/)

32 <http://www.loudandlonely.com/depot/articles/spider.html>

33 Daze verwendet `rel` statt `id`, missbraucht so aber das `rel`-Attribut (das eine Relation zwischen Start- und Zielanker beschreiben soll). Nutzt man stattdessen `id`, bleibt die Methode HTML-konform.

Im Browser wird der Link dann so dargestellt (Abbildung 11):



Abbildung 11: CSS-Pseudoklasse :after

Bisher wird nur eine Adresse angezeigt, dem Verweis fehlt noch die Funktionalität. Einen Mailverweis daraus zu machen, wäre wegen der fehlenden Sicherheit unsinnig. Stattdessen kommt auch hier Javascript zum Einsatz:

```
<a class="email" id="info" title="example.net"
  href="keinskript.html"
  onclick="this.href='mailto:'
          + this.getAttribute('id') + '@'
          + this.getAttribute('title') ">
  Mail senden
</a>
```

Durch Aktivieren des Verweises wird das Event `onclick` ausgelöst und ein Skript ausgeführt, das den Anker mit einem neuen Wert überschreibt. Der Inhalt des neuen URI setzt sich zusammen aus `mailto:`, dem Wert des `id`-Attributs, dem `@`-Zeichen sowie dem Inhalt des `title`-Attributs. Danach wird der Zielanker des Verweises aufgerufen. Diese Lösung hat den Vorteil, dass Nutzer ohne Javascript zu einer Seite weitergeleitet werden können, auf der sie über den Schutz der Adresse informiert werden.

### ***Funktionalität***

Ein auf diese Art gebildeter Verweis verhält sich bei Aktivierung wie ein Standardemailverweis, zeigt beim Überfahren mit der Maus jedoch die Adresse der Erklärungsseite an.

### ***Gebrauchstauglichkeit***

Die Pseudoklassen `:after` und `:before` sind Bestandteile von CSS 2.0 und werden von modernen Browsern wie Firefox oder Opera unterstützt; der

Internet Explorer 6.0 dagegen kennt diese Pseudoklassen nicht. Damit wird der Verweis in einem Großteil genutzten Browser nicht dargestellt wie gewünscht. Das ist aber weniger problematisch, als es jetzt anmutet. Im Internet Explorer sieht man nun einen Verweis, der „Mail senden“ als Verweistext enthält. Aktiviert man den Verweis, so verhält sich der Internet Explorer wie die anderen Browser auch.<sup>34</sup>

Die verwendete Javascriptmethode `getAttribute()` entstammt der Version 1.5, die von älteren Browsern nicht unterstützt wird. Aktuelle Versionen der verbreitetsten Browser sind in der Lage, diese Funktion auszuführen.<sup>35</sup>

Was passiert nun, wenn ein Browser CSS, aber nicht Javascript unterstützt? Dann funktioniert der Verweis nicht, aber im Verweistext wird ja die Adresse angegeben. Ein Nachteil der Pseudoklassen `:after` und `:before` ist allerdings, dass so generierter Inhalt nicht in allen Browsern selektierbar ist. In Firefox zum Beispiel muss der Nutzer die Adresse abtippen.

Was passiert, wenn ein Browser weder CSS noch Javascript unterstützt? Dann wird die Emailadresse nicht angezeigt, auch funktioniert der Emailverweis nicht, aber der Nutzer wird auf die entsprechende Hinweisseite weitergeleitet, so dass er zumindest auf das Problem hingewiesen wird.

Bei der Untersuchung haben 54 Teilnehmer die Adresse richtig erkannt. Sieben von zehn Teilnehmern, die keine Adresse erkannt haben, benutzten den Internet Explorer, in dem die Adresse nicht dargestellt wird.

Die Kombination von CSS und Javascript ergibt eine durchaus gebrauchstaugliche Methode, die ihrer Zeit aber noch ein wenig voraus ist.

### ***Sicherheit***

Diese Methode ist so neu und ungewöhnlich, dass sie derzeit wohl wirksam vor Spambots schützt. Was ist aber, wenn die Methode weite Verbreitung findet, so dass Spammer daran interessiert sind, sie anzugreifen?

---

<sup>34</sup> Die für den Herbst 2006 angekündigte Version 7.0 des Internet Explorers unterstützt CSS 2.0

<sup>35</sup> Internet Explorer seit Version 5.5, Netscape seit Version 6.0, Firefox seit Version 1.0, Opera seit Version 5.12, Safari seit Version 1.0

Um die Emailadresse zu extrahieren, muss ein Spambot in der Lage sein, die Werte der Attribute zu lesen und passend zusammensetzen. Einfach ist das nicht, aber mit genügend Aufwand möglich. Sehr viel schwieriger wird es, wenn das angegebene Schema nicht strikt verwendet wird, sondern in Variationen, z.B.

```
<a class="email" id="inf" title="ample"
  href="keinskript.html"
  onclick="this.href='ma'+ilt+'o:'
          + this.getAttribute('id')
          + '%6F@e%78'
          + this.getAttribute('title')
          + '.net'">
  Mail senden
</a>
```

Hier sind keine Grenzen gesetzt, wie schon bei der dynamischen Erzeugung des Mailverweises ist die Anzahl der Javascriptmethoden grenzenlos, ein Auslesen durch Spambots ist nur möglich, wenn diese über einen Javascriptinterpreter verfügen, wobei dann wieder das Problem auftritt, dass der Aufwand, alle Skripte auszuführen, viel zu groß ist.

Diese Methode ist demnach ausreichend sicher.

## Gweax – ein gutartiger Spambot

---

Um zu verdeutlichen, dass viele der hier vorgestellten Methoden nicht sicher sind, habe ich ein Programm geschrieben, das versucht, möglichst viele der Verschleierungen zu überwinden.

Ich habe keinen Spambot geschrieben. Das Programm durchsucht zwar eine Datei nach Emailadressen, arbeitet aber nicht rekursiv, d.h. verlinkte Seiten werden nicht durchsucht. Um den möglichen Missbrauch zu erschweren, habe ich zudem das Programm künstlich verlangsamt. Daher kommt der Name Gweax: ein **GoodWill Email Address eXtractor**.

Im Grunde besteht das Programm aus drei Komponenten: der Hauptklasse, einem Adressverwalter und einigen Filtern.

### *Hauptklasse Gweax*

Diese Klasse steuert das Programm. Eine übergebene URL (lokal oder aus dem Internet) wird geladen und Zeile für Zeile den Filtern übergeben. Nach dem Auslesen werden die gefundenen Adressen an der Standardausgabe ausgegeben (Abbildung 12).



```
matthias@Rosalyn: ~/java/Gweax
Datei Bearbeiten Ansicht Terminal Reiter Hilfe
matthias@Rosalyn:~/java/Gweax$ java gweax.Gweax http://wwwcs.uni-paderborn.de/cs
/ag-szwillus/personen/szwillus/adresse_main.html
Durchsucht http://wwwcs.uni-paderborn.de/cs/ag-szwillus/personen/szwillus/adress
e_main.html
szwillus@upb.de
matthias@Rosalyn:~/java/Gweax$
```

Abbildung 12: Bildschirmfoto Gweax

## *Addressverwaltung* AddressManager

Der AddressManager ist für die Verwaltung der gefundenen Adressen verantwortlich. Er überprüft, ob eine Adresse syntaktisch korrekt ist<sup>36</sup>, speichert sie ohne Duplikate und hält Methoden bereit, um Adressen auszugeben oder wieder zu löschen.

Der AddressManager hält auch eine Methode `repairAddress()` bereit, um fehlerhafte Adressen zu korrigieren. Insbesondere wird die Maskierung des `@`-Zeichens rückgängig gemacht sowie Adresszusätze wie `REMOVE` entfernt und dadurch entstandene Syntaxfehler (z.B. zwei aufeinanderfolgende Punkte) behoben.

## *Filter* GweaxFilter

GweaxFilter ist als Oberklasse für alle Filter eine abstrakte Klasse. Die einzige relevante Methode ist `extractEmailAddresses(String Line)`, mit der die Filter aus einer übergebenen Zeile Emailadressen extrahieren sollen. Die einzelnen Unterklassen sollen jeweils eine Darstellungsmethode überwinden.

### MailtoFilter

Der MailtoFilter ist der erste Filter und dafür vorgesehen, Emailadressen auszulesen, die mit der Standardmethode dargestellt werden. Er sucht in der Zeile nach dem Vorkommen von `mailto`, und extrahiert die darauf folgende Zeichenkette bis zum nächsten Anführungszeichen.

### MaskedAtFilter

Dieser Filter ergänzt den MailtoFilter um eine Methode, maskierte `@`-Zeichen zurückzuwandeln. Er wurde durch die `repairAddress()`-Methode in AddressManager überflüssig.

---

<sup>36</sup> Dabei orientiert er sich an der im Allgemeinen verwendeten Syntax, nicht an der in RFC 2822 erlaubten, Der Unterschied besteht darin, dass in RFC 2822 auch Sonderzeichen wie z.B. Klammern erlaubt sind, die in der Praxis aber sehr selten genutzt werden.

## AddedCapitalsFilter

Dieser Filter ergänzt den MailtoFilter um eine Methode, adressfremde Bestandteile in Großbuchstaben zu entfernen. Problematisch dabei ist, dass solche Adressen auch von dem MailtoFilter erkannt, nicht aber korrigiert werden. Darum habe ich die Korrektur von Adressen dem AddressManager überlassen. Dieser Filter ist somit überflüssig.

## FindUnlinkedAddressesFilter

Um Emailadressen zu finden, die nicht in einem Emailverweis dargestellt sind, sucht dieser Filter nach dem Vorkommen des @-Zeichens innerhalb einer Zeile. Davon ausgehend wird die größtmögliche Umgebung gesucht, die eine korrekte Emailadresse darstellt. Dieser Filter findet also alle Emailadressen, die das @-Zeichen enthalten, unabhängig von der Art der Darstellung.

## BackwardMaskingFilter

Der BackwardMaskingFilter ist eine Unterklasse des FindUnlinkedAddressesFilters. Er überprüft den Rückwert einer Zeile nach dem Vorkommen einer Emailadresse, so wird eine Maskierung mittels der CSS-Eigenschaft `direction: rtl` erkannt.

## LinktoUncryptFilter

Dieser Filter wurde geschrieben, um Emailadressen zu erkennen, die mit der Javascriptmethode `linkto_uncryptmailto()` verschlüsselt wurden. Der Filter extrahiert den Parameter der Javascriptmethode und wandelt ihn entsprechend um.

## Zusammenfassung

---

Welche Methode soll man nun verwenden? In welchem Verhältnis gewichtet man die drei Kriterien Sicherheit, Gebrauchstauglichkeit und Funktionalität? Das hängt wohl vom Einsatzgebiet ab.

Ein Unternehmen wird die Gebrauchstauglich höher gewichten als die Sicherheit. Ein Kunde, der nicht in der Lage ist, die Kontaktmailadresse zu erkennen, und daraufhin zur Konkurrenz wechselt, wiegt deutlich mehr als eine Spammail.

Eine private Homepage wird die Sicherheit stärker bewerten, die Barrierefreiheit ist weniger wichtig, zumal wenn der Nutzerkreis bekannt ist.

Eine Seite, die sich an Computerfreaks wendet, kann Methoden verwenden, die ein Laie nicht versteht.

### *Unbrauchbare Methoden*

Trotz der unterschiedlichen Anforderungsprofile gibt es einige Methoden, von deren Verwendung ich abraten möchte, weil sie ein oder mehrere Kriterien nicht erfüllen.

### Fehlende Sicherheit

Die Methode der Verschleierung des @-Zeichens ist nicht sicher genug. Die Zeichenmaskierung mit Unicode ist ebenfalls zu unsicher. Die Sicherheit des Rückwärtsschreibens beruht lediglich auf ihrer geringen Verbreitung. Auch das Hinzufügen von Elementen wie REMOVE ist nicht sicher. Alle vier Methoden werden übrigens von meinem Programm überwunden.

### Fehlende Gebrauchstauglichkeit

Das Hinzufügen von adressfremden Bestandteilen wie REMOVE ist nicht nur unsicher, sondern auch unbrauchbar, weil die Nutzer nicht damit zurechtkommen. Javascript mit Nutzerinteraktion ist für den Nutzer zu kompliziert; das Untersuchungsergebnis ist hier deutlich genug.

## *Brauchbare Methoden*

Von den elf Methoden bleiben somit noch sechs übrig, deren Vor- und Nachteile abgewogen werden müssen.

### **Standardmethode**

Eigentlich sollte diese Methode nicht verwendet werden, weil sie keine Sicherheit bietet. Wer auf diese Art eine Emailadresse darstellt, wird Spam erhalten, ohne Zweifel. Andererseits ist sie der Standard, wird also von jedem Browser dargestellt und von jedem Nutzer verstanden. Außerdem kommt keine andere Methode an ihre Funktionalität heran. Wem die Sicherheit nicht so wichtig ist, der sollte erwägen, zugunsten der Nutzer ganz auf den Schutz anderer Methoden zu verzichten. Ein guter Spamfilter reicht dann aus.

Ich persönlich nutze die Standardmethode. Seit Jahren findet sich meine Hauptemailadresse im Internet, und es gibt keine Möglichkeit, sie wieder herauszuziehen. Ich bekomme täglich etwa 60 Spammails, von denen mein Filter in den letzten Monaten 99,7 % erkannt hat. Der Einsatz einer anderen Methode würde keine Verbesserung bringen.

### **Adresse als Grafik**

Auf diese Methode müsste eigentlich wegen der fehlenden Funktionalität und der fehlenden Barrierefreiheit verzichtet werden. Unternehmen, Behörden, Schulen und andere Einrichtungen werden sicher auch darauf verzichten. Allerdings ist sie ziemlich sicher und wurde – zu meiner Überraschung – im Test von fast allen Teilnehmern erkannt. Deswegen kann man über einen Einsatz auf privaten Homepages nachdenken.

### **Entzerren der Adresse**

Der große Nachteil dieser Methode ist die hohe Fehlerrate von über elf Prozent in der Erkennung, die sich in der Untersuchung gezeigt hat, weshalb sie im Grunde unbrauchbar ist. Dennoch gibt es ein Einsatzgebiet für das Entzerren der Adresse. In Internetforen und Gästebüchern kann man häufig keine HTML-Elemente verwenden, doch wird von der Forensoftware versucht, Internet- und Emailadressen automatisch zu erkennen und als Verweise

darzustellen. Gibt man nun seine Emailadresse an, wird sie erkannt und steht dann ungeschützt im Internet. Häufig ist die einzige Möglichkeit, die dann noch zum Schutz bleibt, das Entzerren.

## Javascript: Text dynamisch erzeugen

Der einzige Nachteil dieser Methode ist die Unbrauchbarkeit, wenn Javascript deaktiviert wurde. Ansonsten ist sie sicher, funktional und gebrauchstauglich. Um die Sicherheit auch bei großer Verbreitung zu gewährleisten, sollten die Variablen und Funktionen auf jeder Seite unterschiedlich benannt werden. Die zusätzliche Verschleierung der Werte durch geeignete Methoden erhöht die Sicherheit noch einmal.

Das Problem, dass bei fehlender Javascriptunterstützung diese und die anderen Methoden, die auf Javascript bauen, unbrauchbar sind, kann umgangen werden, indem man einen `noscript`-Bereich einfügt, in dem dem Nutzer ein Kontaktformular angezeigt wird:

```
<noscript>
<form action="kontakt.php">
  Ihre Emailadresse:
  <input type="text" name="absender">
  Betreff:
  <input type="text" name="betreff">
  Ihre Nachricht:
  <textarea cols="50" rows="5" name="nachricht">
  </textarea>
  <input type="submit">
</form>
</noscript>
```

In diesem Bereich, der nur angezeigt wird, wenn der Browser kein Javascript unterstützt (oder es deaktiviert wurde), findet sich ein Formular mit je einem Eingabefeld für die Emailadresse des Absenders und den Betreff sowie ein Eingabebereich für die Nachricht<sup>37</sup>. Dieses Formular bietet größtenteils die Funktionalität einer Email<sup>38</sup>, ohne dass die Emailadresse des Empfängers sichtbar ist. Im Beispiel wird das Formular von einem Skript `kontakt.php` serverseitig verarbeitet und z.B. als Email verschickt.

---

<sup>37</sup> Es geht mir hier um die prinzipielle Möglichkeit der Verwendung eines Formulars, darum ist das Beispiel sehr knapp gehalten.

<sup>38</sup> Was fehlt, ist die lokale Kopie der gesendeten Email im Emailprogramm. Der Nutzer sendet also eine Email, ohne sie später noch einmal lesen zu können.

Wird nun eine Javascriptmethode mit einem Kontaktformular kombiniert, ergibt sich eine gebrauchstaugliche Methode.

### **Javascript: linkto\_uncryptmailto()**

Diese Methode bietet im Vergleich zur letztgenannten keine Verbesserung; im Gegenteil: sie ist weniger sicher, bietet geringere Funktionalität und ist auch weniger gebrauchstauglich. Auch wenn die Untersuchungsergebnisse diese Methode als praktikabel belegen, spricht nichts dafür, diese anstelle der vorherigen zu verwenden.

### **CSS-Pseudoklasse :after**

Diese Methode ist mein persönlicher Favorit, wegen der eleganten Art, wie die Emailadresse dem Verweistext angefügt wird. Wenn auf einer Webseite mehrere Adressen stehen, muss nicht jede einzeln verschlüsselt werden, die Formatierung einer CSS-Klasse wirkt sich auf jede Adresse aus. Das hebt diese Methode z.B. von der Verschlüsselung mit Javascript ab.

Der Nachteil dieser Methode ist allerdings, dass sie vom Internet Explorer bis einschließlich Version 6 nicht unterstützt wird. Die Version 7, die die in dieser Methode verwendeten Elemente unterstützt, ist allerdings für den Herbst 2006 angekündigt und wird durch automatische Updates unter Windows schnell verbreitet werden. Dadurch hat diese Methode eine Zukunft.

## Auf einen Blick

	Standardmailverweis	@-Zeichen ersetzt	REMOVE zugefügt	Unicode	Adresse als Grafik	Entzerren	JS Text dynamisch erzeugen	JS linkto_ unencryptmailto	JS mit Nutzerinteraktion	CSS direction:rtl	CSS-Pseudoklasse :after
Anklicken öffnet Mailprogramm	●	●	●	●	-	-	●	●	O <sup>2</sup>	-	●
Überfahren zeigt Mailadresse	●	O <sup>1</sup>	O <sup>1</sup>	●	-	-	●	-	-	-	-
Wird dargestellt in											
Internet Explorer 6.0	●	●	●	●	●	●	●	●	●	●	O <sup>6</sup>
Firefox 1.5	●	●	●	●	●	●	●	●	●	●	●
Opera 9.0	●	●	●	●	●	●	●	●	●	●	●
Safari	●	●	●	●	●	●	●	●	●	_4	●
Lynx	●	●	●	●	-	●	-	-	-	-	O <sup>6</sup>
Kommt ohne JS aus	●	●	●	●	●	●	-	-	-	●	-
Kommt ohne CSS aus	●	●	●	●	●	●	●	●	●	-	-
Adresse wird angezeigt	●	O <sup>1</sup>	O <sup>1</sup>	●	●	O <sup>3</sup>	●	-	-	●	●
Adresse ist selektierbar	●	●	●	●	-	O <sup>3</sup>	●	-	-	O <sup>5</sup>	O <sup>7</sup>
Nutzeruntersuchung											
Erkennung <sup>8</sup>	1,0	0,961	0,529	1,0	0,984	0,887	1,0	0,822	0,489	0,883	1,0
Akzeptanz <sup>9</sup>	0,955	0,895	0,895	0,955	0,955	0,954	0,877	0,682	0,692	0,923	0,844
Vermeidet Indikatoren (@, mailto)											
Auslesen benötigt JS-Interpreter	-	-	-	-	-	-	●	0	●	-	●
Auslesen benötigt Bilderkennung	-	-	-	-	●	-	-	-	●	-	-
sonstiger Schutz	-	-	●	-	-	-	-	-	-	●	-
Widersteht Gweax	-	-	-	-	●	●	●	-	●	-	●

- trifft ohne Einschränkungen zu
- 0 trifft mit Einschränkungen / nur teilweise zu
- trifft nicht zu
- 1 Verschleierte Emailadresse wird gezeigt
- 2 mit Zwischenschritt
- 3 Anfang und Ende der Adresse nicht trivial erkennbar
- 4 Rückwert wird angezeigt

- 5 Rückwert wird selektiert
- 6 Darstellung der Adresse fehlt, sonst vollständig
- 7 Browserabhängig
- 8 Anteil der gewerteten Teilnehmer der Untersuchung, der die Adresse richtig oder mit Tippfehlern erkannt hat
- 9 Anteil der gewerteten Teilnehmer der Untersuchung, der eine Adresse angegeben hat (richtig oder falsch)

## Auswertung der Untersuchung

	<i>Standardemailverweis</i>	<i>@-Zeichen ersetzt</i>	<i>REMOVE zugefügt</i>	<i>Unicode</i>	<i>Adresse als Grafik</i>	<i>Entzerren</i>	<i>JS Text dynamisch erzeugen</i>	<i>JS linkto_unicryptmailto</i>	<i>JS mit Nutzerinteraktion</i>	<i>CSS direction:rtl</i>	<i>CSS-Pseudoklasse :after</i>
korrekt	<b>63</b>	47	27	59	61	53	49	37	20	50	54
Tippfehler	<b>0</b>	2	<b>0</b>	4	1	2	1	<b>0</b>	2	3	<b>0</b>
falsch	<b>0</b>	2	24	<b>0</b>	1	7	<b>0</b>	8	23	7	<b>0</b>
nicht ausgefüllt	<b>2</b>	6	6	<b>2</b>	<b>2</b>	3	7	20	20	4	10
nicht gewertet	<b>2</b>	10	10	<b>2</b>	<b>2</b>	<b>2</b>	10	<b>2</b>	<b>2</b>	3	3

korrekt:	Anzahl der Teilnehmer, die die Adresse fehlerfrei erkannt haben
Tippfehler:	Anzahl der Teilnehmer, die die Adresse mit unbedeutenden Tippfehlern erkannt haben (z.B. zwei Buchstaben vertauscht)
falsch:	Anzahl der Teilnehmer, die eine falsche Adresse angegeben haben
nicht ausgefüllt:	Anzahl der Teilnehmer, die das Eingabefeld nicht ausgefüllt haben
nicht gewertet:	Anzahl der Teilnehmer, die bei einer Methode nicht in die Wertung aufgenommen wurden. Das sind einerseits die zwei Teilnehmer, die die Aufgabe offensichtlich nicht verstanden haben, andererseits die, die Schwierigkeiten hatten, die Adresse zu erkennen, wenn sie entgegen der Gestaltungsregeln nicht lesbar wiederholt wurde.

## Quellen

---

Spam ist ein relativ junges Problem, daher existiert zum Thema der Spamprävention im Allgemeinen und der Darstellungsmethoden von Emailadressen im Speziellen wenig bzw. keine Literatur. Die hier erläuterten Methoden habe ich durch intensive Nutzung des Internets kennengelernt, manche durch aktive Suche. Bei einigen Methoden sind die Urheber nicht mehr ermittelbar, bei anderen dagegen schon.

### *Standards*

- Berners-Lee, Tim und Conolly, Daniel: *Hypertext Markup Language*, RFC 1866  
<http://www.ietf.org/rfc/rfc1866.txt>
- Berners-Lee, Tim: *Universal Resource Identifiers in WWW*, RFC 1630  
<http://www.ietf.org/rfc/rfc1630.txt>
- Crocker, David H. (Hrsg.): *Standard for the format of ARPA internet text messages*, RFC 822  
<http://www.ietf.org/rfc/rfc0822.txt>
- Münz, Stefan: *SelfHTML*, insbesondere das Kapitel *Emailverweise definieren*  
<http://de.selfhtml.org>  
<http://de.selfhtml.org/html/verweise/email.htm>

### *Untersuchungen und Berichte*

- Center for Democracy and Technology: *Why Am I Getting All This Spam? Unsolicited Commercial E-mail Research Six Month Report*, März 2003 [Untersuchung darüber, wo Spammer Emailadressen sammeln]  
<http://www.cdt.org/speech/spam/030319spamreport.shtml>
- Federal Trade Commission: *Email Address Harvesting: How Spammers Reap What You Sow*, November 2002 [Untersuchung darüber, wo Spammer Emailadressen sammeln]  
<http://www.ftc.gov/bcp/online/pubs/alerts/spamalrt.htm>

- Group Technologies: *E-Mail Lifecycle Management – mehr als nur Anti-Spam*, Folie 4, September 2004 [Kosten von Spam in Unternehmen],  
Download nicht mehr verfügbar. <http://www.group-technologies.com>
- heise.de: *24-jähriger Massen-Spammer zahlt eine Million US-Dollar Strafe*, 06.06.2006  
<http://www.heise.de/newsticker/meldung/73914>
- Messagelabs: *Monthly Report: July 2006*, Juli 2006 [Analyse des Spamanteils an Email weltweit]  
[http://www.messagelabs.com/publishedcontent/publish/treat\\_watch\\_dotcom\\_en/intelligence\\_reports/july\\_2006/DA\\_155200.chp.html](http://www.messagelabs.com/publishedcontent/publish/treat_watch_dotcom_en/intelligence_reports/july_2006/DA_155200.chp.html)
- Norman: *Why spammers spam*, undatiert [Untersuchung über Motive und Methoden der Spammer]  
[http://downloads.norman.no/whitepaper/Why\\_spammers\\_spam.pdf](http://downloads.norman.no/whitepaper/Why_spammers_spam.pdf)
- tagesschau.de: *Neues Gesetz soll Spam bekämpfen*, 15.06.2006  
[http://www.tagesschau.de/aktuell/meldungen/0,1185,OID5624320\\_REF2,00.html](http://www.tagesschau.de/aktuell/meldungen/0,1185,OID5624320_REF2,00.html)
- TEC Software, *Power Email Extractor 4.1* [Werbung für einen Spambot]  
<http://tecsoftware.biz/extractor.htm>

## *Methoden*

- Daze, Ned: *Spider stomping 2k6* [CSS-Pseudoklasse :after]  
<http://www.loudandlonely.com/depot/articles/spider.html>
- Kummer, Jürgen: *JavaScript eMail Encrypter* [Javascript-Verschlüsselung]  
<http://jumk.de/nospam/>
- Nicholls, Stu: *CSSplay – Hiding email addresses* [CSS-Eigenschaft direction:rtl]  
<http://www.cssplay.co.uk/menu/email.html>

- Syronex:

*Anti-Spam Email Address Encoder* [Javascript mit Nutzerinteraktion]

<http://w2.syronex.com/jmr/safemailto/>

*Efficiency Study* [Sicherheitsanalyse dieser Methode]

<http://www.syronex.com/antispam/efficiency-study>

## Anhang: Inhalt der CD

---

Dieser Studienarbeit liegt eine CD bei. Auf ihr finden sich folgende Inhalte:

<i>index.html</i>	Das Inhaltsverzeichnis in HTML-Format. Alle Inhalte sind von hier aus verlinkt.
<i>Quellen</i>	In diesem Ordner finden sich Kopien der für diese Arbeit verwendeten Quellen.
<i>gweax</i>	Dieser Ordner beinhaltet Quelltext und kompilierte Klassen des Programms Gweax.
<i>Untersuchung</i>	Hier findet sich die im Rahmen dieser Arbeit durchgeführte Untersuchung.