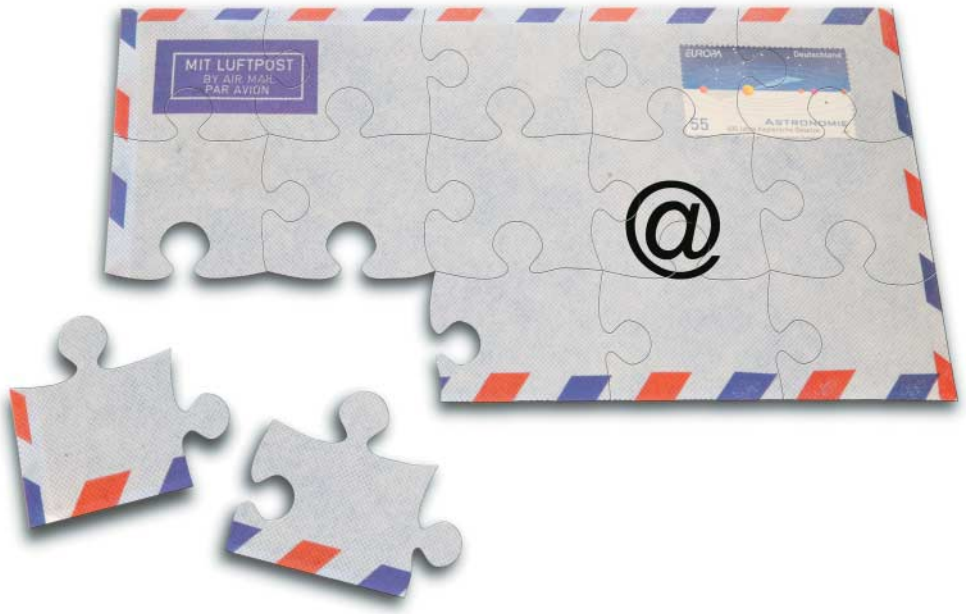
Nachrichten über Skripte und andere Programme versenden



Zusammenbau

Peer Heinlein

Nachdem die letzten beiden Ausgaben dieser Tutorial-Reihe das Empfangen und Versenden von E-Mails behandelt haben, geht es diesmal um den eigentlichen Inhalt der transportierten E-Mails. Es gilt nicht nur diverse Spamfilter zu passieren, sondern auch die eigene Außendarstellung nicht zu beschädigen.

Programmierer und Administratoren, die mehr Zeit für Trial-and-Error-Tests aufwenden als für das Lesen einschlägiger RFCs, gefährden ihren Arbeitsplatz. Das gilt besonders für E-Mail-Inhalte, die von Webapplikationen oder anderen Programmen generiert werden. Hier genügt es eben nicht, sich den Output mit dem eigenen Mailclient anzuschauen und für gut zu befinden. Die korrekte Darstellung einer E-Mail kann reiner Zufall sein.

Häufig vergessen Entwickler das Festlegen des Zeichensatzes und geben damit die Kontrolle aus der Hand, wie der Client des Empfängers das Ergebnis aussehen lässt. Er kann den Body einer E-Mail nahezu beliebig interpretieren – sei es nach UTF8, ISO-8859-12 oder gar gemäß einem (zumindest für Europäer) exotisch anmutenden chinesischen Zeichensatz wie BIG5. Ohne die genaue Angabe der genutzten Codierung stellt der Client die E-Mail mit ihren spezifischen Sonderzeichen höchstwahrscheinlich falsch dar. Fehlt im Mail-Header die Angabe des Fonts, muss der Client raten – und nimmt im Zweifel den Font, der als Standard eingerichtet ist.

Leider gilt es unter Programmierern als ein Leichtes, „schnell“ eine E-Mail zu erzeugen. Welche Details zu beachten sind, wissen längst nicht alle – woher auch, wenn schon viele Postmaster darüber nicht Bescheid wissen. Und wer nach dem Verfahren „Trial and Error“ programmiert, bemerkt derartige Fehler nicht unbedingt: Der Programmierer arbeitet unter UTF-8, er erzeugt seine Testmails unter UTF-8, am Ende liest er sie unter UTF-8 – und alles scheint korrekt.

Fehlender Zeichensatz alarmiert Spamfilter

Viele von Webshops, Ticket-Systemen, Web-Communities oder anderen Programmen erzeugte Status-E-Mails sehen daher anderswo fehlerhaft aus. Statt Sonderzeichen und Umlauten gibt's dann Kästchen oder Fragezeichen als Platzhalter – oder einfach falsche Sonderzeichen. Gegenüber Kunden mit anderen Default-Zeichensätzen, die solchen Datenschrott zu Gesicht bekommen, entsteht eine peinliche Außendarstellung. Dem Kunden bleibt die Hoffnung, dass das Unternehmen nicht immer so schludrig arbeitet – oder er bestellt gleich woanders.

Auch Spamfilter achten auf den Zeichensatz. Fehlt diese Vorgabe, haben die fraglichen E-Mails offensichtlich nichts mit der Kommunikation zwischen zwei Anwendern und deren ganz normalen Mailclients zu tun. Dass es sich vielmehr um von einem anderen Programm generierte und noch dazu falsch deklarierte Mails handelt, macht sie spamverdächtig.

Wer Header-Zeilen unter Missachtung der RFCs erzeugt und Grundlegendes wie Zeichensatzdefinitionen vergisst, trägt dazu bei, dass die eigenen E-Mails häufiger in Filtern hängenbleiben und den Empfänger nicht sicher erreichen. Postmaster sollten sicherstellen, dass alle E-Mails aus dem eigenen Haus den RFCs genügen. Ein Blick in den Mail-Header zeigt schnell, ob der Zeichensatz Berücksichtigung findet:

```
Content-Type: text/plain; charset=iso-8859-1
```

Datenmüll im Betreff vergrault Kunden

Zeichensätze spielen nicht nur im eigentlichen Inhalt (Body) einer E-Mail einer Rolle. Auch Fehler im Betreff (Subject) selbst generierter E-Mails sind häufig anzutreffen – und fallen dort besonders unangenehm auf.

Die Subject-Zeile gehört nicht zum Body, sondern zum Header einer E-Mail. Eine für den Body vorgenommene

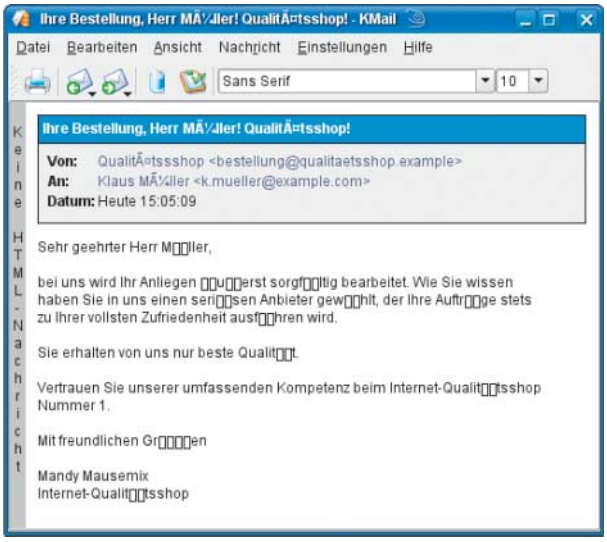
Definition des „charset“ gilt für ihn also gar nicht. Im Header einer E-Mail sind prinzipiell nur 7-Bit-Zeichen erlaubt, um einen reibungslosen Transport und eine problemlose Verarbeitung der E-Mail sicherzustellen. Andere Zeichen – fast ausschließlich im Betreff und beim Realnamen – sind einzeln zu codieren. Die Betreffzeile

```
Subject: Schöne Grüße
```

sieht richtig codiert wie folgt aus:

```
Subject: =?iso-8859-1?b?U2No9m5l?=
         =?iso-8859-1?b?IEdy/N9l?=
```

E-Mails, in denen Umlaute und andere Sonderzeichen falsch oder gar nicht deklariert sind, können nicht nur scheußlich aussehen, sondern sogar geschäftsschädigend sein (Abb. 1).



Wenn Teile des Header nicht von vornherein feststehen, beispielsweise weil der Realname des Kunden dynamisch in die Adresszeile einzufügen ist, sollten Programmierer nach fertigen Bibliotheken ihrer Programmiersprache suchen, etwa *phpmailer*. Statische Header-Bestandteile lassen sich einfach einer mittels Mailclient an sich selbst versendeten Beispiel-E-Mail entnehmen.

Absendernamen mit Umlauten

Gerade in der To:-Zeile stehen oft Angaben des Anwenders, die nicht ungefiltert in versendete E-Mails gelangen dürfen. Daher sind hier passende Umcodierungsroutinen unverzichtbar. Realnamen bergen weitere Tücken, die immer wieder Verwirrung stiften – und am Ende sogar dazu führen können, dass E-Mails in falsche Hände gelangen.

In E-Mails an mehrere Empfänger sind deren Adressen im Header-Feld „To“ durch ein Komma oder Semikolon voneinander getrennt. Knifflig wird es, wenn im Realnamen bereits ein Komma enthalten ist. Der Header-Eintrag

```
To: Mueller, Klaus <klaus.mueller@example.com>
```

definiert aus Sicht der Mailserver gleich zwei Empfänger. Zum einen „klaus.mueller@example.com“ mit dem Realnamen „Klaus“, zum anderen den „Mueller“, dessen Mailadresse erst einmal ohne Hostnamen daherkommt. Den ergänzt der erstbeste Mailserver des Providers zu „mueller@$myhostname“. Im schlimmsten Fall gibt es diese Mailadresse tatsächlich – und unbeteiligte Dritte empfangen die E-Mail. Realnamen sind daher immer in Anführungszeichen einzukapseln:

```
To: "Mueller, Klaus" <klaus.mueller@example.com>
```

Da sogar der Header-Eintrag „From“ mehrere durch Kommas voneinander getrennte Absender enthalten kann, hat auch hier eine fehlende Kapselung durch Anführungszeichen oder eine fehlende Codierung der Umlaute ihre besondere Wirkung. Etwaige Antworten der Nutzer gehen dann an beide Absender. Das führt entweder dazu, dass die E-Mail in falsche Hände gelangt oder dass der Antwortende eine mysteriöse Bounce-Meldung erhält, obwohl seine E-Mail beim Empfänger angekommen ist.

Komplikationen drohen, wenn der Realname wegen seiner Sonderzeichen MIME-codiert ist und zugleich spezielle Zeichen wie ein Komma enthält. Je nach Kombination der beteiligten Softwarekomponenten und je nachdem, welche Teile sie wie codieren, kann es unvorhersagbare Effekte geben. So erzeugt eine bestimmte Kombination von Outlook und Exchange in diesen Fällen einen Realnamen, der fälschlicherweise auch die einkapselnden Anführungszeichen mit codiert.

Die Folge: Das Komma ist nun wieder ungeschützt und trennt aus Sicht der Mailserver ganz normal zwei Empfänger der E-Mail. Wer mit derartigen Software-Bugs konfrontiert wird, sollte aus eigenem Interesse Satzzeichen, vor allem das Komma und das Semikolon, in Realnamen vermeiden. Besser ist es, ein Empfänger heißt „Klaus Müller“ statt „Müller, Klaus“. Dasselbe gilt für den Absender, denn beim Antworten fügen Mailer in der Regel dessen Realnamen in die Empfängeradresse ein.

Da es in der Vergangenheit immer wieder Verwirrung stiftete, dass Mailserver kaputte Mailadressen durch das Anfügen des eigenen Hostnamens zu reparieren versuchen, bietet Postfix in den neueren Versionen die Möglichkeit, derartige Mailadressen durch „@domain.invalid“ zu ergänzen, damit kein falscher Eindruck entsteht. Dafür muss Postfix zunächst erfahren, welche E-Mails er als „lokal“ erzeugt ansehen soll – denn für diese ist die Ergänzung um „@$myorigin“ weiterhin notwendig und sinnvoll:

```
local_header_rewrite_clients=permit_inet_interfaces,
permit_sasl_authenticated, permit_tls_clientcerts
```

Sodann konfiguriert der Postmaster im Postfix, mit welcher Dummy-Domain der MTA Adressen externer Mails „reparieren“ soll:

```
remote_header_rewrite_domain = domain.invalid
```

Auf diese Weise gelangen keine eigenen E-Mails mit missverständlichen Adressen nach draußen.

Ob aus Laxheit oder Unwissenheit: Wer Zusammenhänge ignoriert und die Maildienste unsauber konfiguriert, wird bestraft. Besonders häufig gehören

iX-TRACT

- Ein Großteil aller E-Mails entsteht nicht „von Hand“ in Mailclients wie Outlook oder Thunderbird, sondern wird von Webskripten und anderen einfachen Programmen automatisch generiert.
- Zu den häufigsten Fehlerquellen beim Generieren von E-Mails gehören falsche oder fehlende Vorgaben für den Zeichensatz, den der Mailclient des Empfängers anzeigen soll.
- Webformulare, die dem Versenden von E-Mails dienen, sind beliebte Angriffsziele von Spammern und erfordern besondere Aufmerksamkeit.

dazu Absendernamen oder -Domains, die in hausinternen, automatisch generierten E-Mails zum Einsatz kommen.

Alarme, die niemand wahrnimmt

Zu oft gehen Postmaster davon aus, dass keine Reaktionen auf derlei Aussendungen erfolgen – beispielsweise, wenn es um Statusmitteilungen von Geräten oder Monitoring-Systemen geht. Ungemach droht, wenn diese E-Mails irgendwann nicht mehr zustellbar sind, beispielsweise weil der ursprüngliche Administrator das Unternehmen verlassen hat. Ist weder die Alarm-Mail selbst noch die von ihr ausgelöste Bounce-Meldung zustellbar, bleibt den beteiligten Servern nichts anderes übrig, als die möglicherweise kritischen Benachrichtigungen zu verwerfen. Das Admin-Team erfährt dann vielleicht zu spät, dass etwa ein RAID-Controller den Ausfall einer Festplatte beklagt. Und wenn es doch auffällt, dann im schlimmsten Fall dadurch, dass das System den Dienst endgültig quittiert.

Gerade bei Alarmsystemen, die normalerweise jahrelang schweigen, sollten Administratoren daher sowohl auf richtige Absenderangaben als auch auf die Nutzung immer vorhandener Role-Accounts als Empfänger achten. Role-Accounts führen zwar zu wechselnden Anwendern, verschwinden aber anders als personengebundene Mailkonten nicht einfach, wenn jemand den Arbeitgeber wechselt oder in den Ruhestand geht.

Einbahnstraße Kundenkommunikation?

Auch und gerade bei Newslettern und anderen automatisch erzeugten und versendeten Mailings sollten die Absenderangaben stimmen. Zum einen lassen sich nur auf diese Weise Bounces auswerten und veraltete Empfängeradressen aus dem Bestand werfen. So viel Rücksichtnahme auf andere Systeme, deren Ressourcen sonst verschwendet wären, sollte selbstverständlich sein. Zum anderen handeln Postmaster im eigenen Interesse, wenn sie ihren Datenbestand aktuell halten: Fast alle großen Provider sperren andere Server, die zu viele unzustellbare Mailadressen anmailen wollen.

Wer Absenderadressen wie noreply@… nutzt, signalisiert seinen Kunden zunächst, dass er auf Feedback und bidirektionale Kommunikation mit ihnen keinen Wert legt. Der Kunde soll die beworbenen Produkte nur kaufen, nicht kommentieren. Kundenwünsche können schließlich Marktforschungsinstitute ermitteln. Doch es bleibt festzuhalten, dass manche Mailserver eine Absendervalidierung betreiben und Mails nicht existenter Absender nicht annehmen. Mit einer derartigen Überprüfung tut sich das empfangende System zwar keinen Gefallen, wie in den ersten beiden Teilen dieses Tutorials bereits erläutert. Doch wer seine E-Mails zuverlässig versenden möchte, muss auch mit solchen Systemen umgehen können und sollte darum auch für noreply@… empfangsbereit sein.

Envelope- ungleich Header-From

Wer Bounces von Antworten der Empfänger unterscheiden will, sollte bei der Erzeugung der E-Mails auf die korrekte Definition von Header- und Envelope-Absender achten. Wie schon in Teil 1 dieses Tutorials ausgiebig beleuchtet, sollten Bounces und automatisierte Antworten der Empfänger an die Envelope-Adresse gehen, in der Regel im Header als „Sender“ oder „Return-Path“ angegeben.

Auch wenn sich viele Autoresponder leider nicht an die korrekte Verwendung von „Sender“ und „From“ halten, sollten Absender von Newslettern und Betreiber von Autorespondern ihrerseits auf die korrekten Angaben im Mail-Header achten. So lässt sich wenigstens ein Großteil der Rückläufer von „handgeschriebenen“ Antworten trennen. Darüber hinaus lassen sich auf diese Weise Mail-Endlosschleifen vermeiden.

Einige SMTP-Erweiterungen der letzten Jahre widmen sich den sogenannten „Delivery Status Notifications“ (DSN). War es bislang nur üblich, Bounces zu bekommen, wenn eine E-Mail nicht zustellbar war, lassen sich Mailserver über DSN heute auch anweisen, den Absender über eine erfolgreiche Zustellung zu benachrichtigen. Überdies kann der Absender einer E-Mail anordnen, dass selbst im Falle einer Unzustellbarkeit keine Bounce-Nachricht an den Absender gehen soll. Wer für sich sowieso beschlossen hat, seine Newsletter unter einem nicht existenten noreply-Absender zu versenden, könnte zumindest auf eine entsprechende DSN-Angabe zurückgreifen, um die Systeme allgemein zu entlasten.

DSN-fähige Mailserver übertragen die zusätzlichen Informationen durch erweiterte Angaben im SMTP. Sie können für jeden mittels „RCPT TO“ angegebenen Empfänger definieren, ob der Absender eine Benachrichtigung im Falle einer Unzustellbarkeit (FAILURE), einer verspäteten (DELAY) oder einer erfolgreichen Zustellung (SUCCESS) verlangt – oder ob er in keinem Fall an einer Rückmeldung interessiert ist (NONE). Ohne DSN-Angabe gehen Mailserver wie früher üblich von FAILURE und DELAY aus.

```
RCPT TO: <user@example.com> NOTIFY= ↵
                                FAILURE,SUCCESS
```

Beim Mailserver Postfix lässt sich diese Angabe auch beim Versenden per Kommandozeile mit dem Aufrufparameter „-N“ angeben, sodass sich DSN für alle in Webapplikationen erzeugten E-Mails nutzen lässt:

```
sendmail -N FAILURE,SUCCESS [...]
```

Während diese Definition pro Empfänger nötig ist, kann ein Absender für die gesamte SMTP-Sitzung vorgeben, ob Bounces die ursprüngliche Nachricht enthalten (FULL) oder ob nur Header-Zeilen zurückgehen sollen (HDRS):

```
MAIL FROM: <user@example.net> RET=HDRS
```

Das *DSN_README* von Postfix erläutert, wie sich eine E-Mail um eine individuelle Envelope-ID ergänzen lässt und erleichtert es herauszufinden, welche E-Mail einen Empfänger nicht erreichte. Das Readme erläutert zudem weitere Sicherheitsaspekte rund um den Einsatz von DSN.

Verdächtiges Verhalten vermeiden

Wer Kunden, Newsletterempfänger oder Bestellungen verwaltet, nutzt häufig individualisierte Zahlencodes. Doch Vorsicht: Eine mittels Zahlenkolonne abgeschlossene Betreffzeile – beispielsweise mit der Bestellnummer – kann Spamverdacht auslösen, da es ein

Tutorialinhalt

Teil I: E-Mails sicher versenden
Teil II: Alle E-Mails außer Spam empfangen
Teil III: E-Mails korrekt erzeugen

vor einigen Jahren weitverbreitetes Spammerkmal war. Spammer versuchten mit stets wechselnden Zahlen, die Prüfsummenverfahren mancher Filtersysteme auszutricksen, die identische E-Mails wiedererkennen sollen.

Auch wenn dieser Trick heute seltener zum Einsatz kommt, enthalten viele Anti-Spam-Systeme nach wie vor Regelsätze, die auf solche spezifischen Symptome in Betreff oder Abspann einer E-Mail anschlagen. Ähnliches gilt für Textklauseln, die typischerweise in Spam enthalten sind. Wer seine E-Mails mit Textpassagen wie „This email is not spam“ abschließt, muss sich nicht wundern, wenn sie der Filter des Empfängers am Ende genau dafür hält.

Webformulare als Spamverteiler

Webanwendungen, die E-Mails erzeugen, verarbeiten nicht selten Eingaben der Anwender. Ob Bestellbestätigungen oder per E-Mail versandte Kopien von Gästebucheinträgen: Bekanntlich sollte niemand frei eingegebene Daten ungeprüft weiterverarbeiten – weder zum Speichern in der Datenbank noch zum Generieren von E-Mails.

Mailadressen sind auf korrekte Syntax und gültige Zeichen zu überprüfen. Leider findet de facto entweder gar keine oder aber eine übertrieben strenge Filterung statt, die sogar einige zulässige Zeichen verbietet. Viele Programmierer übersehen beispielsweise, dass entgegen vielen im Netz verbreiteten Anleitungen und Beispielen ein „+“-Zeichen in Mail-Accounts zulässig und auch tatsächlich sinnvoll und verbreitet ist.

Zeilenumbrüche und andere Steuerzeichen haben dagegen weder im Betreff noch in Mailadressen etwas zu suchen. Spammer versuchen systematisch, in Webformularen präparierte Eingaben an den Server zu senden, die Zeilenumbrüche enthalten. Per Betreff-Feld in einem Kontaktformular können Spammer auf diese Weise komplette eigene Header-Zeilen und ganze Empfängerlisten generieren:

```
Kontaktanfrage<cr><lf>To:
andereruser@example.org<cr><lf>To:
nocheinuser@example.net
```

Übernimmt der Mailer diese Eingaben mitsamt der Zeilenumbrüche in den generierten Mail-Header, können Spammer auf diesem Weg ihre eigenen Header-Zeilen einfügen – und damit auch beliebige Empfängeradressen in die erzeugten E-Mails einschleusen, selbst wenn der eigentliche Empfänger der E-Mail hartcodiert vorgegeben war. Mit dem gleichen Trick können Spammer sogar einen eigenen Mail-Body vorgeben – und damit Spam an beliebige Empfänger über fremde Webformulare versenden.

In der fertigen E-Mail sieht das am Ende wie folgt aus:

```
[...]
Subject: Kontaktanfrage
To: andereruser@example.org
To: nocheinuser@example.net
Buy Viagra!
[...]
```

Postmaster sollten also einen kritischen Blick auf alle Webformulare werfen, die E-Mails erzeugen können. Daten, die später im Mailheader landen, dürfen keine ungeschützten Zeilenumbrüche enthalten. Hier sind klärende Gespräche mit den Webagenturen oder der hauseigenen Entwicklungsabteilung notwendig.

Webformulare gehören nicht zu den ergiebigsten Spamquellen. Doch diese Methode ist weit genug verbreitet, Unheil zu stiften – besonders für den Hoster des Webservers. Schließlich können seine Mailserver in solchen Fällen über kurz oder lang auf Blacklists landen, was den Versand von E-Mails aller Kunden stört. Schon aus eigenem Interesse – und auch aus Rücksichtnahme gegenüber anderen Nutzern – gilt es darum auch ausgehende E-Mails stets einer Spamfilterung zu unterziehen.

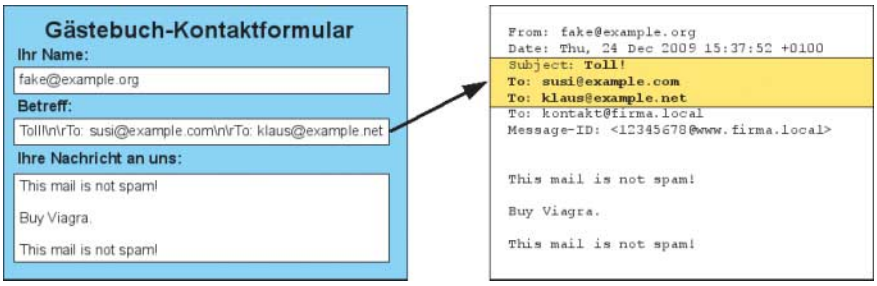
Wenn PHP-Skripte auf Webservern E-Mails über */usr/sbin/sendmail* versenden, tragen die Mails häufig die User-ID des Webservers als Absenderadresse ein, beispielsweise wwwrun@www.example.com. Bounces an diesen Envelope-Absender sind jedoch meist unzustellbar, und Schwierigkeiten beim Versenden bleiben unbemerkt – leider. Denn gerade wenn ein Spammer ein Webformular kapert, wären die dadurch erzeugten zahlreichen Bounces ein zuverlässiges und einigermaßen frühes Alarmzeichen. Auch Betreiber von Webshops und -foren achten besser darauf, ob und wie viele ihrer E-Mails unzustellbar sind – sonst bleiben Fehler zu lange unbemerkt und das Geschäft leidet.

Programmierer sollten beim Versenden der E-Mails über */usr/sbin/sendmail* den korrekten Envelope-Absender über den Aufrufparameter „-f“ definieren. Anbieter von Shared-Webservern, die viele Kunden auf einem System beherbergen, sind praktisch gezwungen, einzugreifen. Sie können mittels

```
php_admin_value sendmail_path "sendmail ≠t ≠i ≠f
kunde@example.com"
```

im VirtualHost-Container einer Apache-Konfiguration für jede Kundendomain eine individuelle Absenderadresse vorgeben. So landen Bounces stets beim Kunden – und nicht beim Webmaster des Hosters. Darüber hinaus lassen sich Spamangriffe auf Webformulare aufgrund des nun eindeutigen Envelope-Absenders dem zuständigen Kunden zuordnen. Das langwierige und mühsame Analysieren der Logfiles entfällt, der Schaden bleibt durch ein schnelles Sperren des defekten Formulars begrenzt.

Ein weiterer angenehmer Nebeneffekt für Webhoster: Oft bemerkt der Kunde aufgrund der Bounces von sich aus etwas von dem Angriff und beseitigt die Ursache umgehend selbst. Wer erst einmal 10 000 und mehr Bounces in seiner Inbox vorfindet, die ein eigenes Formular hervorgerufen hat, benötigt erfahrungsgemäß nicht mehr viel Überzeugungsarbeit, bis er den Fehler im Webformular behebt. (un)



Viele Webformulare lassen sich zum Verbreiten von Spam missbrauchen – im Falle falsch konfigurierter Absenderadressen sogar für lange Zeit unbemerkt (Abb. 2).

PEER HEINLEIN

ist seit 1992 auf E-Mail spezialisiert, Autor des „Postfix-Buchs“ und für die Mailserver, Spam- und Virenabwehr diverser ISPs, Rechenzentren und Unternehmen verantwortlich.