Peer Heinlein

Das Postfix-Buch

Sichere Mailserver mit Linux

3., aktualisierte und erweiterte Auflage

Bibliografische Information Der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Gesamtlektorat: Dr. Markus Wirtz
Satz: Open Source Press (LaTeX)
Umschlaggestaltung: www.fritzdesign.de unter Verwendung des Postfix-Logos von Karol Kreński
Gesamtherstellung: Kösel, Krugzell

ISBN 978-3-937514-50-5 http://www.opensourcepress.de

Kapitel 9

Wenn's komplizierter wird: Postfix Policy Delegation

Mit geschickter Zusammenstellung der Restrictions lässt sich erstaunlich viel erreichen, aber es gibt Entscheidungsszenarien, die sich mit Postfix-Bordmitteln nicht umsetzen lassen. Einige Beispiele:

Greylisting
: Abhängig davon, ob ein Client schon mehrfach versucht hat, eine Mail einzuliefern, wird die Mail mal erlaubt, mal nicht.

Rate-Limiting
: Die Prüfung, dass ein Nutzer nicht mehr als *n* Mails innerhalb eines Tages oder eines bestimmten Zeitraums versendet.

Zeitliche Entscheidungen
: Zum Beispiel die Regel, dass die Schüler einer Schule erst nach 15 Uhr nach draußen senden dürfen, vorher nur schul- oder klassenintern.

Wenn-Dann-Entscheidungen
: Postfix kommt schnell an seine Grenzen, wenn zwei Faktoren gemeinsam eine Entscheidung beeinflussen sollen: *Wenn* der HELO-Name `abc` *und* die Empfängeradresse `xyz` lautet, *dann* soll die Mail angenommen werden.

Weiteres ...
: Man kann ja die Entscheidung von beliebigen Faktoren abhängig machen wollen: Dem Wetter, dem Wochentag, der Laune des Administrators, dem Zufallsgenerator oder der Mondphase. Ich gebe zu, dass mir hier gerade ein praktisch relevantes Beispiel fehlt, aber dem geschulten B.P.f.H. (*Bastard Postmaster from Hell*) mangelt es sicher nicht an Erfindungsreichtum.

9.1 Das Policy Delegation Protocol

Für all diese Zwecke bietet Postfix die sogenannte *Policy Delegation*, bei der Postfix über einen Unix-Socket oder eine TCP-Verbindung im Klartext bislang bekannten technische Merkmale der E-Mail an einen Policy-Dämon übermittelt, damit dieser die Auswertung vornimmt, eine Entscheidung fällt und diese in der gleichen Verbindung als Ergebnis an Postfix zurückmeldet.

Im Prinzip handelt es sich um eine „programmierbare `access`-Map“, und so bietet eine Policy Delegation auch dieselben Reaktionsmöglichkeiten wie die statischen `access`-Maps: `OK`, `REJECT`, `FILTER`, `DISCARD`, `APPEND`, `HOLD` und vieles andere mehr (siehe Kapitel 5.2.7 ab Seite 123).

Mit den verschiedenen Postfix-Versionen ist die Zahl der übergebenen Parameter deutlich angestiegen. Postfix übergibt sämtliche zum aktuellen Zeitpunkt bekannten Parameter der laufenden Verbindung – nur nicht die Mail selbst. Da sich die laufende Verbindung also noch vor dem `DATA`-Kommando befindet, wurde die Mail selbst auch noch nicht übertragen.[1]

```
Ab Postfix-Version 2.1:
request=smtpd_access_policy
protocol_state=RCPT
protocol_name=SMTP
helo_name=some.domain.tld
queue_id=8045F2AB23
sender=foo@bar.tld
recipient=bar@foo.tld
recipient_count=0
client_address=1.2.3.4
```

[1] Kommt es auf den Inhalt der Mail an, so muss auf `smtpd_proxy_filter`, `content_filter` oder `milter` zurückgegriffen werden.

```
client_name=another.domain.tld
reverse_client_name=another.domain.tld
instance=123.456.7
Ab Postfix-Version 2.2:
sasl_method=plain
sasl_username=you
sasl_sender=
size=12345
ccert_subject=solaris9.porcupine.org
ccert_issuer=Wietse+20Venema
ccert_fingerprint=C2:9D:F4:87:71:73:73:D9:18:E7:C2:F3:C1:DA:6E:04
Ab Postfix-Version 2.3:
encryption_protocol=TLSv1/SSLv3
encryption_cipher=DHE-RSA-AES256-SHA
encryption_keysize=256
etrn_domain=
Ab Postfix-Version 2.5:
stress=
```

Die Policy-Dämonen werden wie die anderen Prüfungen in die `smtpd_*_restrictions` eingetragen. Da in aller Regel die Empfängeradresse eine Rolle spielt, bieten sich meist nur die `smtpd_recipient_restrictions` an. Sie müssen lediglich angeben, wo Postfix den Policy-Dämon findet, also den Pfad zum Socket im Dateisystem oder eben IP-Nummer und Port. Läuft der Dämon lokal auf dem Postfix-Server, ist der Unix-Socket im Dateisystem geringfügig performanter als eine TCP/IP-Verbindung an `localhost`.

```
smtpd_recipient_restrictions=
    [... andere Checks ...]
    check_policy_service unix:private/socketname,
    [... andere Checks ...]
```

Allerdings müssen Sie dann darauf achten, dass der Socket für Postfix auch beschreibbar ist; gleichzeitig darf der Socket aber auch nicht für beliebige User-IDs lesbar sein, sonst könnte ein Dritter die Daten des Policy-Delegation-Protokolls mitlesen – und das sind immerhin personenebezogene, sensible Informationen. Außerdem muss der Socket ggf. auch aus einer chroot-Umgebung heraus erreichbar sein – darum sollte er wie in diesem Beispiel nach `/var/spool/postfix/private/` gelegt werden.

Deutlich unkomplizierter ist es, den Dämon über TCP/IP anzusprechen, denn die Probleme mit Dateirechten und chroot-Umgebung stellen sich hier nicht. Solange Sie nicht ernsthaft auf höchste Performance-Optimierung achten müssen, ist es durchaus legitim, auch lokal auf TCP/IP zu setzen.

```
smtpd_recipient_restrictions=
     [...]
     check_policy_service inet:127.0.0.1:10022,
     [...]
```

Natürlich kann der Policy-Dämon auch auf einem entfernten Rechner installiert sein. Ein zentraler Dämon für mehrere Mail-Relays ist auf diese Weise realisierbar:

```
smtpd_recipient_restrictions=
      [...]
      check_policy_service inet:192.168.10.20:10023,
      [...]
```

Entwickelt wurde die Postfix Policy Delegation ursprünglich für die Einbindung von Greylisting in Postfix. Wietse Venema entschloss sich damals, gleich eine eigene Schnittstelle für die Einbindung solcher Zusatzprogramme zu schaffen. Einige solcher Dämonen wurden mittlerweile entwickelt, nennenswerte praktische Bedeutung haben in meinen Augen bislang vier von ihnen erlangt:

postgrey
: von David Schweikert[2] bzw. alternativ dazu

SQLgrey
: von Lionel Bouton[3] und als weiterer Spam-Schutz

policyd-weight
: von Robert Felber[4] und

policyd
: von Nigel Kukard[5] als eierlegende Wollmilchsau

Wer einen eigenen Dämon entwickeln möchte, findet auf den Postfix-Webseiten[6] zahlreiche Anregungen und mit dem dort gelisteten Paket `smtpd-policy-template` von Michael Tokarev eine gute Ausgangsbasis in Perl, die man nur um eigene Entscheidungsstrukturen erweitern muss. Zudem hat David Schweikert für die 2. Mailserver-Konferenz 2005 einen lehrreichen Vortrag über die Programmierung stabiler und performanter Policy-Dämonen zusammengestellt.[7]

9.2 Greylisting

Im Jahr 2004 entwickelte sich eine neue Idee zur effektiven Vermeidung von Spam: *Greylisting*. Dieses Konzept lässt den Inhalt einer Mail vollkommen

[2] `http://postgrey.schweikert.ch/`
[3] `http://sqlgrey.sourceforge.net/`
[4] `http://www.policyd-weight.org/`
[5] `http://www.policyd.org/`
[6] `http://www.postfix.org/addon.html`
[7] Folien auf `http://www.heinlein-support.de/web/akademie/mailserver-konferenz-2005/`

außer Acht und vermeidet Spam, indem es einige Besonderheiten ausnutzt, *wie* Spammer ihre E-Mails versenden. Ausgangspunkt dieses Verfahrens ist die Erkenntnis, dass die Ziele von Mailservern und Spammern grundsätzlich verschieden sind:

- Mailserver verwalten E-Mails, die sie alle sorgfältig und gezielt zustellen sollen – jede einzelne E-Mail und jeder einzelne Empfänger sind wichtig. Entscheidend ist, dass eine bestimmte Mail bei einem bestimmten Empfänger ankommt. Dafür betreiben Mailserver einen gewissen Aufwand und unternehmen ggf. zahlreiche Zustellversuche.
- Ein Spammer hingegen hat nur eines im Sinn: *viele* Mails zu versenden. An *wen* diese Mails gehen ist ihm vollkommen egal. Bekommt die Mail nicht der, dann bekommt sie eben ein anderer – was soll's!? Es zählt allein die Quote: Wie viele Millionen Mails konnten in einer bestimmten Zeit verschickt werden und wie hoch war die Response Rate? Denn es gilt: Je mehr versendete Mails, desto mehr Cash.

9.2.1 So funktioniert Greylisting

Ein greylistendes System geht nun wie folgt vor:

1. Hat der Client `RCPT TO:` gesendet, schaut sich der Mailserver noch vor der Übertragung des Mailinhalts ein sogenanntes *Triple* aus Absender-IP, Absender-Mailadresse und Empfänger-Mailadresse an. Das sind sozusagen die technischen Eckdaten dieses Mailtransports: Welcher Server liefert hier von wem an wen eine Mail ein? Die Empfänger-IP hingegen ist unwichtig – das ist ja das Empfangssystem selbst.
2. Dieses Triple speichert der Server zusammen mit einem Zeitstempel in einer kleinen Datenbank. Ist das Triple neu, wird die Mail *temporär* abgewiesen, also mit einem `4xx`-er Fehler.
3. Ist der Client ein echter Mailserver, wird er die Mail kurz in seine Deferred-Queue verschieben und nach wenigen Minuten erneut zustellen.
4. Beim nächsten Zustellversuch erkennt das empfangende System anhand des dann schon in der Datenbank bekannten Triples, dass dieser Server bereits einen Zustellversuch für diese Mailverbindung unternommen hat. Er akzeptiert die E-Mail nun, sofern keine weiteren Filterregeln dagegen sprechen, denn dieser kleine Test hat das gewünschte Ergebnis geliefert: Der einliefernde Server treibt „Brutpflege“, er queued die einzelne E-Mail, er versucht es erneut. Sprich: Es handelt sich nicht um einen Fire-and-Forget-Spammer, sondern einen echten Mailserver.

5. Die Mail wird – mit einer kurzen Zeitverzögerung – anstandslos angenommen.

Jeder „normale“ Mailserver kann heute mit temporären Ablehnungen umgehen. Gerade große Provider geben seit vielen Jahren immer wieder stundenlang (unfreiwillig) temporäre Fehler aus, wenn deren Systeme überlastet oder die Datenbank zusammengebrochen sind. Wer bei temporären Fehlern passt, könnte an diese großen Provider daher kaum noch Mails senden ... Temporäre Fehler sind keineswegs selten: Platte voll, DNS kaputt oder einfach nur zu viele Verbindungen. Selbstverständlich muss ein Mailserver damit umgehen und die entsprechenden Mails queuen. Wer Angst hat, dass Greylisting zu Mailverlusten führen wird, der muss ehrlicherweise ständig davon ausgehen, dass Mails ohnehin so gut wie nie ankommen:

Hier zwei jüngere Beispiele aus einer Mailqueue – Greylisting der unfreiwilligen Art:

```
E276BF6751     3661 Sun Feb 10 00:10:07  user@example.com
(host mx-ha02.web.de[217.72.192.188] said: 451 <xxx@web.de> Datenbank
nicht zugreifbar/Database not available (in reply to RCPT TO command))
                                        xxxx@web.de


EDA0BF639C     4621 Sat Feb  9 00:30:03  MAILER-DAEMON
(host ASPMX.L.GOOGLE.com[209.85.135.114] said: 450-4.2.1 The Gmail user
you are trying to contact is receiving 450-4.2.1 mail at a rate that
prevents additional messages from 450-4.2.1 being delivered. Please
resend your message at a later 450-4.2.1 time; if the user is able to
receive mail at that time, 450 4.2.1 your message will be delivered.
(in reply to RCPT TO command))
                                        xxxx@gmail.com
```

Häufig höre ich die Befürchtung, es könne Mailserver geben, die (RFC-widrig) nicht mit temporären Fehlern umgehen können. Doch wer sich darum sorgt, der muss sich konsequenterweise auch Gedanken darüber machen, ob diese Mailserver vielleicht auch nicht mit „`@`“-Zeichen in Mailadressen umgehen können oder gar nicht wissen, was eine IP-Adresse ist. Mailserver *müssen* mit temporären Fehlern umgehen, denn temporäre Fehler sind kein optionales Feature, sondern Grundbestandteil des SMTP-Protokolls. Eine Software, die das nicht kann, ist schlichtweg kein Mailserver.

Natürlich wäre es auch für Spammer mit ihren Botnetzen kein Problem, Greylisting zu überleben, denn technisch ist das keine Schwierigkeit: Sie müssten nur eine kleine Empfängerliste führen und nach einer bestimmten Zeit erneute Zustellversuche unternehmen. Aus diesem Grunde wird mir seit Jahren prognostiziert, dass Greylisting doch bestimmt bald wirkungslos werde; es sei doch nur eine Frage der Zeit, bis Spammer Greylisting überleben würden.

Merkwürdigerweise funktioniert Greylisting aber seit Jahren ganz hervorragend – was mich allerdings keineswegs wundert. Warum also ist Greylisting allen Unkenrufen zum Trotz nach wie vor so effektiv?

9.2.2 Wollen Spammer überhaupt Greylisting überleben?

Wenn Greylisting für Spammer keine technische Hürde bedeutet stellt sich also die Frage, ob Spammer Greylisting überhaupt überleben *wollen*.

Ich behaupte: Nein.

Für den Spammer zählt, wie geschildert, primär der tatsächliche Durchsatz pro Stunde. Jeder erneute Zustellversuch ist für ihn mit dem Risiko behaftet, dass wir ihn erneut greylisten und wieder wegschicken. In dieser Zeit ist er besser beraten, es anderswo zu probieren.

Oder anders formuliert: Jeder nutzlose Zustellversuch beim Greylisting ist eine nicht-verschickte E-Mail an einen anderen Empfänger.

Natürlich könnte es auch sein, dass der Spammer auch an anderer Stelle auf Greylisting stößt – doch die Wahrscheinlichkeit ist ausreichend gering. Viel zu viele Systeme sind auch heute noch völlig ungeschützt. In vielen Systemen läuft heute ja nicht einmal ein SpamAssassin mit Default-Einstellungen.

Oder wieder anders formuliert: Die Wahrscheinlichkeit, beim Wiederholungsversuch wieder gegreylistet zu werden und/oder an anderer Stelle bei diesem offensichtlich gut geschützten System zu scheitern, ist hoch. Die Chance, bei einem blinden Versand an eine x-beliebige andere Adresse im ersten Anlauf durchzukommen, ebenfalls. Es ergibt also keinen Sinn, erneute Zustellversuche zu unternehmen, wenn der Spammer dadurch nur seinen effektiven Durchsatz ruiniert.

Natürlich lohnt es sich für einen Spammer, Greylisting zu überleben, je mehr Systeme Greylisting einsetzen. Nach wie vor sind das sehr wenige, aber die Zahl steigt. Doch das verschiebt das Rechenexperiment lediglich: Je mehr greylistende Systeme es gibt, umso häufiger und länger muss ich einen Spammer zwingen, erneute Zustellversuche zu unternehmen. Und schon verschiebt sich für den Spammer die Kosten-Nutzen-Rechnung wieder dahingehend, dass er an anderer Stelle seine E-Mails schneller und damit besser losbekommt.

Betrachten wir eine Analogie aus dem realen Leben: In der Fußgängerzone trifft man oft die einschlägig bekannten „Sind-Sie-nicht-auch-dagegen-dass-Tiere-gefoltert-werden"-Spendensammler der einschlägig unbekannten Tierschutzorganisationen. Diese Spendensammler erhalten (genau wie Spammer) Abschlussprovisionen.

Versetzen Sie sich kurz in die Lage eines solchen Spendensammlers: Lohnt es sich, einen einzelnen Passanten eine halbe Stunde lang mühsam zu

überzeugen, wenn er von vornherein zu verstehen gegeben hat, dass er dieser speziellen Tierschutzorganisationen kein Stück weit traut? Was wäre der Lohn am Ende der Arbeitsstunde? Selbst wenn die Skeptiker mühsam überzeugt werden konnten – am Ende winken aufgrund des hohen Zeiteinsatzes höchstens zwei abgeschlossene Mitgliedsanträge – wenn überhaupt.

Ist es demgegenüber nicht viel aussichtsreicher, die Finger von den Zweiflern zu lassen? Masse sorgt für Cash: Nur wer im Sekundentakt x-beliebige Passanten anquatscht, kurz deren Bereitschaft prüft und nur dann Zeit in ein Gespräch investiert, wenn einer anbeißen könnte, wird am Ende der Arbeitsstunde mit einer relevanten Zahl abgeschlossener Verträge rechnen können.

Denn auch hier gilt: Entscheidend ist, was pro Stunde geschafft wurde. Und viele schnelle Geschäfte bei mäßiger Erfolgsaussicht liefern oft bessere Ergebnisse als wenige intensive Gespräche mit guten Chancen.

9.2.3 Spammer haben es bereits getestet

Und siehe da: Im Herbst 2006 begannen einige Botnetze Greylisting zu überleben. Wie man sieht, war die dafür notwendige Technik also bereits programmiert. Doch schon nach wenigen Wochen änderten die Spammer wieder ihr Verhalten und zogen bei drohendem Greylisting umgehend von dannen, um ihr Glück andernorts zu versuchen. Offensichtlich ergaben diese Testläufe, dass die Erfolgsquote sank und der Einsatz unrentabel war.

Doch keine Regel ohne Ausnahme: Einige Aktien-Spammer bzw. die dafür maßgeblich verantwortliche Spammer-Gruppe überleben Greylisting. Diese Gruppe verfolgte aber ohnehin eine andere Taktik: „Qualitäts-Spam". Technisch versiert und kreativ, bewarb sie zunächst ihre Produkte durch Bild-Spam, bis Texterkennungssysteme in SpamAassassin & Co. eingebaut wurden. Daraufhin reagierten diese Spammer umgehend und versendeten Interlaced-GIF-Bilder, bei denen die Werbebotschaft erst im letzten Frame versteckt war. Als die Anti-Spam-Filter auch darauf reagierten, folgten sehr schwer zu scannende Bilder (hellblauer Wackeltext auf blauem Grund); später stellten sie auf PDF-Attachments um. Den Ausflug in Word- und Excel-Dateien haben diese Spammer nach sehr kurzer Zeit wieder beendet – offensichtlich gingen diese Dateiformate zu stark zu Lasten der Erfolgsquote. Letzter Stand der Technik (Frühjahr 2008) dieser Gruppe ist MP3-Spam, also der tägliche Viagra-Podcast ...

Diese Leute kennen sich wirklich aus und betreiben viel Aufwand, um durch Spamfilter zu gelangen, wobei auch schlechtere Versandquoten in Kauf genommen werden. Es geht um „Kunden", die offenbar nicht sonderlich Spam-erfahren sind und diesen Spam-Mails entsprechend mehr Aufmerksamkeit und damit Reaktionsquote verschaffen. Bei einer solchen Taktik will man dann auch Greylisting überleben.

Doch wir können sie mit ihren eigenen Waffen schlagen. Zumindest derzeit unternimmt diese Spammer-Gruppe nur dann erneute Zustellversuche, wenn sie erkennt, dass sie gegreylistet wird. Melden die greylistenden Server jedoch irgendwelche unverfänglichen Texte (`Please try again later` oder `Server error. Try again later`), ist zu beobachten, dass keine weiteren Zustellversuche unternommen werden.

Eine kleine Änderung des Rückgabetextes erledigt (zumindest derzeit) das Problem von GIF-Bild-Spam, PDF-Spam, Word/Excel-Spam und auch MP3-Spam vollständig. Und da soll noch einmal jemand behaupten, Greylisting sei kein wirkungsvoller Spamschutz! Man muss es nur richtig einsetzen.

Doch selbst wenn Botnetze Greylisting überleben und später wiederkommen, verbessern wir den Spamschutz, denn wir gewinnen wertvolle Zeit,

- in der der verseuchte PC durch sein Verhalten auffällt und vom Besitzer abgeschaltet wird
- in der ein Update des Virenkillers auf dem infizierten PC dem Schädling den Garaus machen könnte
- in der der verseuchte PC durch die DSL-Zwangstrennung eventuell eine neue IP-Adresse bekommen hat und im Greylisting von vorne beginnen muss
- in der andere Systeme Spam und Viren dieses Clients treudoof empfangen haben und in der der fortwährende Spam-Versand alsbald dafür sorgt, dass der Client auf den einschlägigen RBL-Sperrlisten landet, so dass wir diesen Client bei späteren Zustellversuchen entspannt blocken können.

9.2.4 E-Mails dürfen nicht verzögert werden!

Auch ein weiteres Argument gegen Greylisting basiert vorrangig auf unbegründeten Befürchtungen oder schlechter Erfahrung mit unzureichender Greylisting-Software: Natürlich wird nun *nicht* plötzlich *jede* E-Mail stundenlang verzögert. Das wäre ebenso katastrophal wie dumm, denn was sollte das bringen?!

Gute Greylisting-Systeme (wie der hier vorgestellte `postgrey` von David Schweikert) gehen natürlich intelligent und vor allem selbstlernend vor. Denn wenn ein anderes Mailsystem bereits mehrfach erfolgreich Mails von verschiedenenen Absendern an verschiedene Empfänger eingeliefert hat, so steht doch sicher fest: Hinter dieser IP-Adresse verbirgt sich kein Botnetz-Spammer, sondern ein echter Mailserver, der Mails queued und zuverlässig zustellt.

Sobald das feststeht, wird ein gutes Greylisting-System diese IP-Nummer vollautomatisch vom Greylisting befreien. „Whitelisten“ wäre dennoch der

unpassende Begriff, denn es werden weiterhin alle anderen Postfix-Restrictions noch durchgeprüft, und diese können noch zu einer Ablehnung der E-Mail führen. Aber diese Systeme werden zumindest vom Greylisting gewhitelistet, so dass sämtliche E-Mails dieses Servers nicht mehr verzögert werden.

Nach nur sehr wenigen Tagen Training hat ein Mailrelay alle relevanten Mailserver nicht nur der großen Provider, sondern ggf. auch die kleinen Mailrelays von Geschäftspartnern, Zulieferern oder anderen Niederlassungen gelernt. Fortan werden all diese Mails nicht mehr verzögert, ganz gleich, wer Absender oder Empfänger ist.

`postgrey` geht sogar noch weiter: Sind aus einem /24-Netz mehrere Mailserver bekannt, wird das gesamte /24-Netz vom Greylisting ausgenommen, da es unwahrscheinlich ist, dass sich in diesem IP-Bereich auch Botnetze oder Dialin-IP-Adressen befinden.

Die Praxis zeigt: Von allen echten E-Mails werden nur rund 2% verzögert. 98% aller echten E-Mails kommen beim Einsatz eines vernünftigen Greylisting-Systems verzögerungsfrei an.

Und welche Art Mails verbirgt sich hinter diesen 2%? Es sind ausschließlich auf unserem System völlig unbekannte Triple, d. h. Absender, die noch *nie* von dieser IP-Adresse eine E-Mail an unseren Empfänger versendet haben und die noch dazu von einem uns bis dahin unbekannten Mailrelay stammen. Bis auf wenige Ausnahmen handelt es sich also stets um Erstkontakte – und dort spielen wenige Minuten Verzögerung nun wahrlich keine große Rolle.

Keine Regel ohne Ausnahme: Natürlich kann es *vereinzelt* Mailadressen geben, bei denen der zeitnahe Empfang extrem wichtig ist: Bestelladressen („Bis 16 Uhr bestellt, morgen geliefert!“), Helpdesk-Adressen („Ich maile Ihnen mal schnell das Logfile rüber“) oder andere zeitkritische Mailadressen z. B. zur Übermittlung von Beiträgen an Zeitungen.[8]

Aber auch das ist kein Argument gegen Greylisting, denn in solchen Fällen muss man eben durch vernünftige Konfiguration *nur diese* wenigen Mailadressen vom Greylisting ausnehmen und den dadurch verminderten Spamschutz akzeptieren. Alle anderen Mailaccounts eines Unternehmens oder Providers kann man mit dieser schönen Technik weiterhin schützen.

Die Angst vor eventuell verzögerten geschäftlichen E-Mails halte ich darum in den meisten Fällen für unbegründet.

Denn was wäre die Alternative angesichts von stets überlasteten Mailrelays, die heute bei jeder Spamwelle zusammenbrechen und nach jeder Spam-

[8] Sieht man einmal davon ab, dass E-Mail ohnehin kein Medium der Echtzeitkommunikation ist… Tatsächlich habe ich aber schon erlebt, dass noch während einer schon laufenden Sendung ganze TV- oder Radio-Beiträge von den Redakteuren *per E-Mail* ins Studio geschickt wurden. Hier wäre Greylisting für die Studio-Adresse natürlich fatal.

Aktion eines Script-Kids einen Mailstau von teilweise mehreren Stunden vor sich herschieben? Es ist doch vielmehr so, dass *hier* Mailverzögerungen die Regel sind – und zwar unkontrolliert bei *allen* Mails! Demgegenüber kann man die gelegentlich verzögerte Annahme unbekannter Absender sicher vernachlässigen.

Oder anders: Nur mit Greylisting und dem damit ggf. verbundenen Ausbremsen weniger unbekannter E-Mails kann ich die verzögerungsfreie Annahme aller wichtigen E-Mails auch während akuter Spammwellen sicherstellen. Wenn wichtige E-Mails nicht verzögert werden dürfen, dann *muss* ich geradezu greylisten!

9.2.5 `postgrey` installieren

Nachdem die immer wieder kursierenden Bedenken gegen Greylisting hoffentlich ausgeräumt wurden, soll es nun um Installation von `postgrey` gehen.

Die aktuellen Distributionen enthalten `postgrey` bereits; bei Debian und SUSE können fertige `postgrey`-Pakete installiert werden, so dass natürlich auch das nachfolgend beschriebene Anlegen eines entsprechenden Nutzers und eines Arbeitsverzeichnisses entfällt. Sie können dann den Dienst in üblicher Weise starten und loslegen.

Installieren Sie `postgrey` jedoch „von Hand“ aus dem Quellcode, so ist auch das kein Problem. Es handelt sich um ein Perl-Script, das nach `/usr/local/sbin` kopiert wird, zudem ein kleines Reporting-Tool `postgrey-report` und zwei Konfigurationsdateien für das Postfix-Verzeichnis `/etc/postfix`:

```
linux:~ # wget http://postgrey.schweikert.ch/pub/postgrey-1.31.tar.gz
linux:~ # tar -xvzf postgrey-1.31.tar.gz
linux:~ # cd postgrey-1.31
linux:~/postgrey-1.31 # cp postgrey /usr/local/sbin/
linux:~/postgrey-1.31 # cp contrib/postgreyreport /usr/local/sbin/
linux:~/postgrey-1.31 # cp postgrey_whitelist_* /etc/postfix/
```

Postgrey sollte unter einem eigenen Nutzer laufen; per Default ist das `postgrey`:

```
linux:~/postgrey-1.31 # useradd -r -s /bin/false postgrey
```

Zu guter Letzt benötigt Postgrey ein kleines Arbeitsverzeichnis, in dem es seine Datenbanken ablegen kann und für das es natürlich auch Schreibrechte benötigt:

```
linux:~/postgrey-1.31 # mkdir /var/spool/postfix/postgrey
linux:~/postgrey-1.31 # chown postgrey /var/spool/postfix/postgrey/
```

Anschließend muss es nur noch gestartet werden. Dabei können Sie sich aussuchen, ob es als Policy-Dämon über einen TCP/IP-Socket erreichbar sein soll:

```
linux:~/postgrey-1.31 # postgrey -d --inet localhost:10023
```

oder über einen Unix-Dateisystemsocket:

```
linux:~/postgrey-1.31 # postgrey -d \
--unix /var/spool/postfix/public/postgrey
```

Wenn Sie `postgrey` anfangs noch zu Kontrollzwecken zusehen wollen, so lassen Sie den Parameter `-d` weg, damit der Dämon im Vordergrund läuft und sichtbar mitprotokolliert.

Sollten Sie `postgrey` nicht starten können, so liegt das wahrscheinlich an einem fehlenden Perl-Modul. Benötigt werden `Net::Server`, `IO::Multiplex` und `BerkeleyDB`.

Über `postgrey --help` erhalten Sie eine vollständige Liste aller zulässigen Parameter. Für den Betrieb sind diese jedoch selten notwendig – es genügt der oben genannte Aufruf.

Wenn der Dienst nun läuft

```
linux:~/postgrey-1.31 # ps ax | grep postgrey
 8756 ?        Ss     0:00 /usr/local/sbin/postgrey -d --inet localhost:
10023
 8758 pts/4    R+     0:00 grep postgrey
linux:~/postgrey-1.31 # lsof -i :10023
COMMAND   PID     USER   FD   TYPE DEVICE SIZE NODE NAME
postgrey 8756 postgrey    5u  IPv4  76174       TCP localhost:10023 (LIS
TEN)
```

können Sie die entsprechende Abfrage in die `smtpd_recipient_restrictions` aufnehmen. Beachten Sie dazu die Ausführungen in Kapitel 8.7 ab Seite 210, um sicherzustellen, dass Sie `postgrey` an der richtigen Stelle einbinden!

```
linux:~/postgrey-1.31 # cd /etc/postfix
linux:/etc/postfix # joe main.cf

smtpd_recipient_restrictions=
[...]
# Unsere Kinderchens erlauben!
        permit_sasl_authenticated,
        permit_mynetworks,
# RBL checken (könnte auch nur über policyd-weight erfolgen)
        reject_rbl_client zen.spamhaus.org,
```

```
        reject_rbl_client ix.dnsbl.manitu.net,
        reject_rbl_client bl.spamcop.net,
        reject_rbl_client dnsbl.njabl.org,
        reject_rbl_client list.dsbl.org,
# policyd-weight
        check_policy_service inet:127.0.0.1:12525,
# Greylisting checken über TCP-Socket:
        check_policy_service inet:127.0.0.1:10023,
# Alternativ bei Zugriff über Unix-Socket:
#        check_policy_service unix:public/postgrey
# Wir prüfen dynamisch auf existente Relay-Empfänger
        reject_unverified_recipient,
# Backup MX erlauben!
        permit_mx_backup,
[...]
```

Vergessen Sie nicht, sich ggf. noch ein eigenes Start-Script für `postgrey` zu schreiben und dafür zu sorgen, dass `postgrey` auch nach einem Neustart des Systems automatisch gestartet wird. Ist ein Policy-Dämon nicht erreichbar, kann Postfix keine E-Mails mehr annehmen.

In der Datei `/etc/postfix/postgrey_whitelist_recipients` können Sie Empfänger angeben, die vom Greylisting ausgenommen sind:

```
linux:/etc/postfix # joe postgrey_whiteliste_recipients
# postgrey whitelist for mail recipients
# -------------------------------------
# put this file in /etc/postfix or specify its path
# with --whitelist-recipients=xxx

postmaster@
abuse@
support@heinlein-support.de
```

Die Datei `/etc/postfix/postgrey_whitelist_clients` listet Server, die vom Greylisting ausgenommen sind. Das sind per Default größtenteils alte Einträge von 2004, als es noch sehr viele Server mit sehr langen Retry-Zeiten gab. Nur äußerst selten fallen heute noch Server auf, die mit Greylisting schlecht umgehen, weil sie mehrere Stunden für erneute Zustellversuche benötigen.[9]

Ich rate davon ab, jetzt mühsam alle möglichen Mailserver in dieser Datei zu whitelisten: Ihr Postgrey wird selbstständig alle für Sie relevanten Server binnen kürzester Zeit automatisch lernen. Probieren Sie es aus.

[9] Und selbst dann sehe ich das nur bedingt als mein Problem an: Wenn der dortige Admin der Meinung ist, er müsse nur alle vier Stunden zustellen, so scheinen diese Mails auch nicht sonderlich wichtig und eilig zu sein – es entspricht dann dem Willen des dortigen Admins, dass diese Mails eben vier Stunden oder länger unterwegs sind. Wenn er Absender schon keine Eile hat – warum soll ich mir dann dieses Problem zu Eigen machen?

9.2.6 Greylisting ist ein hervorragender Virenschutz!

Und ein weiteren enormen Vorteil bietet Greylisting: Es ist ein hervorragender Virenschutz!

Mailviren werden typischerweise über Botnetze versendet. Genauer gesagt sind es ja die Viren und Würmer, die heute nicht mehr nur sich selbst versenden, sondern auch Spam. Werden diese Botnetze durch Greylisting verjagt, können so sehr erfolgreich Virenmails ferngehalten werden. Es ist erstaunlich, in welch gepflegter Langeweile sich die üblichen Anti-Viren-Mailgateways rekeln können, sobald Greylisting vorgeschaltet wurde.

Schon aus einem allgemeinen Sicherheitsgedanken heraus sollte Greylisting darum eingesetzt werden: Selbst wenn das Viren-Botnetz Greylisting überlebt, verschafft uns das System unmittelbar nach dem Ausbruch neuer Viren und Würmer wichtige Zeit bis zum dann hoffentlich schnell eintreffenden Sigantur-Update der Anti-Viren-Hersteller. Firmen geben Unsummen für den Anti-Viren-Schutz eines Netzwerkes aus – dieser kostenlose 99%-Schutz wird jedoch sträflich vernachlässigt.[10]

9.2.7 Die Grenzen von Greylisting

Greylisting schützt auf die beschriebene Weise ganz gezielt vor *Botnetzen*, die heute rund 90% des Spams und vermutlich rund 99% der Viren aussenden. Sobald Mails jedoch über „normale" Mailserver relayt werden, ist Greylisting selbstverständlich wirkungslos – der andere Mailserver wird die Mails queuen und trotz Greylisting einliefern können. Spam, der über gehackte Mailaccounts oder Webformulare versendet wird, kann damit also ebenso wenig geblockt werden wie Spam, der zuerst an Mailadressen bei anderen Providern adressiert war und der dann an uns weitergeleitet wird. Beachten Sie dazu unbedingt die Diskussion in Kapitel 8.7 ab Seite 210!

9.3 policyd-weight

Robert Felber[11] begann vor einigen Jahren mit der Entwicklung des *policyd-weight*,[12] der sich zu einem stabilen und ausgereiften Policy-Dämon gemausert hat und einen wichtigen Beitrag zum Spam- und Virenschutz leistet.

[10] Jede Firma ist herzlich eingeladen, einen Obulus als freiwillige Gebühr an die ehrenamtlichen Autoren solcher Software zu entrichten. Rechnen Sie nur einmal durch, vor welchen finanziellen Schäden Sie Programme wie `postgrey` schützen. Und solange kommerzieller Spam- und Virenschutz schnell mittlere fünfstellige Summen kostet, sollte man mit einer Projektspende hier nicht knausern!

[11] `http://www.selling-it.de`

[12] `http://www.policyd-weight.org`

policyd-weight bekommt alle Client-Angaben während des SMTP-Dialogs und prüft deren Plausibilität. Auf diese Weise werden vornehmlich Botnetz-Spammer entlarvt, die im SMTP-Dialog häufig den Hostnamen im `HELO` und die Absender-Mailadresse im `MAIL FROM:` fälschen. Zudem prüft er, ob die Client-IP-Adresse auf einer oder mehreren RBLs gelistet ist und ob alle beteiligten DNS-Einträge korrekt gesetzt sind.

Einige dieser Prüfungen lassen sich zwar schon mit Postfix-Bordmitteln erreichen, doch führen Fehler dann zum sofortigen Reject der E-Mail, so dass vergleichsweise schnell echte E-Mails herausgefiltert werden, obwohl deren Mailserver nur einen leichten Konfigurationsfehler aufweist.

policyd-weight geht einen anderen Weg: Hier werden alle Erkenntnisse zu einem Gesamt-Score zusammengerechnet. Ein einzelner Fehler reicht darum zur Ablehnung der E-Mail nicht aus. Kommen jedoch mehrere Fehler zusammen, ist für policyd-weight das Maß voll.

Diese Scoring-Idee findet sich so auch bei SpamAssassin. Der Unterschied: SpamAssassin prüft Header und Body der übertragenen E-Mail, während policyd-weight *vor* dem DATA-Kommando ansetzt und damit ausschließlich die technischen Details der SMTP-Übertragung bewertet. SpamAssassin und policyd-weight decken also zwei klar getrennte Anwendungsbereiche ab und ergänzen sich hervorragend.

9.3.1 Prüfungen

Stark vereinfacht dargestellt, prüft policyd-weight vor allem folgende Punkte:

- Welchen Reverse-Lookup hat der Client? Zeigt der Name des Reverse-Lookup seinerseits wieder auf die IP-Adresse des Clients? Hat der Client diesen Hostnamen auch korrekterweise im `HELO` angegeben?
- Sind die Angaben in Reverse-Lookup, HELO und Absender-Domain plausibel?
- Steht die IP-Adresse des Clients auf *mehreren* Sperrlisten? Dabei wird zwischen wichtigen und weniger wichtigen Sperrlisten unterschieden, so dass es einen Unterschied macht, auf *welcher* RBL-Liste der Client steht. Auch die Tatsache, dass er *nicht* auf einer wichtigen RBL-Liste steht, wird – dann positiv – bewertet.
- Steht die Domain des Absenders auf `rfc-ignorant.org` (Kapitel 13.13 ab Seite 408), ist sie also öffentlich für RFC-widrig fehlende `postmaster@`- und `abuse@`-Accounts gebrandmarkt?

Hinter policyd-weight verbirgt sich ein komplexes Regelwerk mit vielen Prüfungen, bei denen Positiv- und Negativ-Punkte vergeben werden. Nur

die Kombination aus verschiedenen Fehlern führt zum `REJECT`. Einige Beispiele:

- Reverse-Lookup und HELO stimmen nicht überein, und zudem ist der Client auf mindestens einer RBL geblacklistet. Neben dem Konfigurationsfehler scheint demnach auch Spamversand von dieser IP-Adresse aus stattzufinden.

- Reverse-Lookup und HELO stimmen nicht überein, und außerdem steht die Absender-Domain in keinem erkennbaren Zusammenhang mit diesen Hostnamen und/oder der Client-IP. Ein schlecht konfigurierter Mailserver kann damit Mails seiner eigenen Domain noch versenden – nicht aber für andere Domains (klassischer Spam mit gefälschtem Absender, ggf. aber auch Weiterleitungen!).

- Die Absender-Domain ist gemäß `rfc-ignorant.org` als RFC-widrig und damit spamfreundliche Domain gelistet, *und* die Client-IP steht auf zwei kleinen oder einer wichtigen RBL-Liste.

- Der Client weist keine Konfigurationsfehler auf, steht aber auf zwei großen/wichtigen RBL-Listen. Konkreter Spamversand von dieser IP-Adresse scheint stattgefunden zu haben, die Chance für ein unglückliches Listing aufgrund eines verzeihlichen Einzelfalls ist eher gering.

Nie jedoch führt ein einfacher Fehler zum sofortigen Ausschluss. Jeder Client darf gewisse Fehler machen – solange die übrigen Angaben plausibel sind und alles nur auf einen kleinen Konfigurationsfehler hinweist.

Einige Beispiele für unproblematische Fehler:

- Stimmen Reverse-Lookup und HELO nicht überein, so kann das toleriert werden, wenn die Absender-Domain trotzdem zum Reverse-Lookup passt, der Client also nur einen falschen HELO nutzt.

- Stimmen alle drei Angaben nicht überein, kann auch das noch toleriert werden, wenn die Inbound-MX-Server der Absenderdomain „dicht" bei der IP-Adresse des Clients liegen, es also durchaus plausibel ist, dass dieser Client für diesen Absender Mails verschickt.

policyd-weight kann mit diesen Prüfungen Spammer und Botnetze zielsicher herausfiltern, denn es macht sich deren typische Fehler zu nutze: Der Reverse-Lookup des Clients zeigt bei Botnetzen aufgrund der Dialup-IP üblicherweise auf irgendeine Dummy-Domain des Einwahl-Providers. Zudem sind `HELO` und `MAIL FROM:` gefälscht und nichts passt schlüssig zueinander.

9.3.2 Installation

Auch wenn Ihre Distribution kein fertiges policyd-weight-Paket mitbringen sollte, müssen Sie auf dessen Dienste nicht verzichten. Die Installation ist trivial, da es sich um ein einzelnes Perl-Script handelt, das noch nicht einmal eine Config-Datei benötigt, wenn es mit den Defaulteinstellungen betrieben wird.

Laden Sie von `http://www.policyd-weight.org` die aktuellste Fassung und speichern Sie diese als `/usr/local/sbin/policyd-weight`:

```
linux:~ # cd /usr/local/sbin
linux:/usr/local/sbin # wget http://www.policyd-weight.org/policyd-weight
```

Geben Sie `root` Ausführungsrechte an dem Perl-Script:

```
linux:/usr/local/sbin # chmod u+rx policyd-weight
```

Per Default läuft policyd-weight unter den User- und Gruppenrechten `polw`. Richten Sie diese Accounts als abgesicherte Systemaccounts ein:

```
linux:/usr/local/sbin # useradd -r -s /bin/false polw
linux:/usr/local/sbin # groupadd -r polw
```

Sie können den Dienst nun starten:

```
linux:/usr/local/sbin # ./policyd-weight start
linux:/usr/local/sbin # ps ax | grep policyd-weight
 8163 ?        Ss     0:00 policyd-weight (master)
 8164 ?        Ss     0:00 policyd-weight (cache)
 8169 pts/0    R+     0:00 grep policyd-weight
linux:/usr/local/sbin # lsof -i :12525
COMMAND    PID USER   FD   TYPE DEVICE SIZE NODE NAME
policyd-w 8163 polw    4u  IPv4  62366       TCP localhost:12525 (LISTEN)
```

Beklagt sich policyd-weight über fehlende Perl-Pakete (`perl-Net-DNS` alias `Net::DNS` aus CPAN), so installieren Sie diese über Ihre Distribution nach.

Nach dem Start lässt sich policyd-weight einfach als Policy-Dämon in Postfix einbinden.

Da er per Default als Policy-Dämon auf dem TCP/IP-Port `localhost:12525` horcht, müssen Sie die `smtpd_recipient_restrictions` nur um einen entsprechenden Aufruf ergänzen. Dieser steht meist nach den RBL-Prüfungen, aber vor Greylisting und `permit_mx_backup`. Details zur exakten Platzierung finden sich in Kapitel 8.7 ab Seite 210.

```
linux:/etc/postfix # joe main.cf
[...]
# RBL checken (könnte auch nur über policyd-weight erfolgen)
        reject_rbl_client zen.spamhaus.org,
        reject_rbl_client ix.dnsbl.manitu.net,
        reject_rbl_client bl.spamcop.net,
        reject_rbl_client dnsbl.njabl.org,
        reject_rbl_client list.dsbl.org,
# Policyd-Weight
        check_policy_service inet:127.0.0.1:12525,
# Greylisting checken!
        check_policy_service inet:127.0.0.1:10023,
# Wir prüfen dynamisch auf existente Relay-Empfänger
        reject_unverified_recipient,
# Backup MX erlauben!
        permit_mx_backup,
# Alles andere Relaying verbieten!
        reject_unauth_destination,
```

Nach einem `postfix reload` sollten Sie schon mit der nächsten Mail entsprechende Logmeldungen in `/var/log/mail` vorfinden.

9.3.3 Logmeldungen

Eine abgelehnte E-Mail hat positive Score-Werte:

```
Jan 20 04:18:42 plasma postfix/policyd-weight[10709]: weighted check:
NOT_IN_SBL_XBL_SPAMHAUS=-1.5 NOT_IN_SPAMCOP=-1.5 NOT_IN_BL_NJABL=-1.5
CL_IP_NE_HELO=1.5 CL_SEEMS_DIALUP=3.75 RESOLVED_IP_IS_NOT_HELO=1.5
HELO_NUMERIC=1.5 (check from: .pluscare. - helo: .casa-7b096dea51. -
helo-domain: .casa-7b096dea51.)  FROM_NOT_FAILED_HELO(DOMAIN)=6.75
<client=201.236.15.17> <helo=casa-7b096dea51>
<from=fleetingtqc5@pluscare.com> <to=user@example.com>, rate: 10.5

Jan 20 04:18:42 plasma postfix/policyd-weight[10709]: decided action=550
Mail appeared to be SPAM or forged. Ask your Mail/DNS-Administrator to
correct HELO and DNS MX settings or to get removed from DNSBLs; MTA
helo: casa-7b096dea51, MTA hostname: unknown[201.236.15.17]
(helo/hostname mismatch); delay: 0s
```

Eine anstandslos angenommene E-Mail hat negative Score-Werte, wird aber über die Postfix-Aktion `PREPEND` zu Informationszwecken mit einer entsprechenden Headerzeile ausgestattet:

```
Jan 21 00:56:31 plasma postfix/policyd-weight[25002]: weighted check:
NOT_IN_SBL_XBL_SPAMHAUS=-1.5 NOT_IN_SPAMCOP=-1.5 NOT_IN_BL_NJABL=-1.5
CL_IP_EQ_HELO_MX=-3.1 (check from: .postfixbuch. - helo: .postfixbuch.
- helo-domain: .postfixbuch.)  FROM/MX_MATCHES_NOT_HELO(DOMAIN)=1
<client=213.203.238.10> <helo=mail.postfixbuch.de>
```

```
<from=user@postfixbuch.de> <to=user@example.com>, rate: -6.6

Jan 21 00:56:31 plasma postfix/policyd-weight[25002]: decided
action=PREPEND X-policyd-weight:  NOT_IN_SBL_XBL_SPAMHAUS=-1.5
NOT_IN_SPAMCOP=- 1.5 NOT_IN_BL_NJABL=-1.5 CL_IP_EQ_HELO_MX=-3.1 (check
from: .postfixbuch. - helo: .postfixbuch. - helo-domain: .postfixbuch.)
FROM/MX_MATCHES_NOT_HELO(DOMAIN)=1 <client=213.203.238.10>
<helo=mail.postfixbuch.de> <from=user@postfixbuch.de>
<to=user@example.com>, rate: -6.6; delay: 2s
```

Außerdem hat policyd-weight einen eigenen Cache-Dämon, damit sich wiederholende Client-Anfragen nicht ständig neu geprüft und aufgelöst werden müssen. Kommt der policyd-weight aufgrund seiner Cache-Daten zu dem Schluss, dass ein Client zu häufig mit denselben Fehlern aufschlägt, beantwortet er die Abfrage mit einem anderen Ablehnungsgrund und vergibt für jeden weiteren zu schnellen Wiederholungsversuch eine „Strafwartezeit“:

```
Jan 20 04:20:12 plasma postfix/policyd-weight[17784]: decided action=550
temporarily blocked because of previous errors - retrying too fast.
penalty: 30 seconds x 0 retries.; <client=201.236.15.17>
<helo=casa-7b096dea51> <from=fleetingtqc5@pluscare.com>
<to=user@example.com>; delay: 0s
```

Hier erkennen Sie aus den Logmeldungen nicht den wahren Grund der Ablehnung. Suchen Sie in diesen Fällen nach dem ersten abgelehnten Zustellversuch dieser Client-IP-Adresse, dort stehen alle Details, die zu der Entscheidung geführt haben.

Übrigens: Ist ein Policy-Dämon nicht gestartet, kann Postfix keine Mails mehr annehmen und gibt dem einliefernden Client die Meldung `451 server configuration error`. Denken Sie also daran, dafür zu sorgen, dass der policyd-weight auch nach einem Reboot des Systems selbstständig startet.

9.3.4 False Positives

Allerdings können mit policyd-weight auch False Positives auftreten, weshalb man sich den Einsatz in der Praxis sehr genau überlegen muss. Ein Beispiel:

Firma A betreibt einen schlecht gepflegten Mailserver. Er hat keinen gültigen bzw. nur einen Dummy-Reverse-Lookup des Upstream-Providers (`host-xxxx.static.provider.de`), und aufgrund unglücklicher Spielereien meldet er sich mit einer ungültigen Domain (`example.local`) im `HELO`, während er jedoch Mails für eine andere Domain (`example.com`) versendet.

Leider ist dies sehr häufig bei kleinen Firmen (DSL-Standleitung oder Betrieb eines Alles-in-einem-Rootservers) der Fall – und in der Praxis sind dies auch kleine, aber wichtige Partner wie Zuliefer-Firmen, die Web-/PR-Agentur, das Architektur-Büro oder der Partner im Ausland.

Doch auch große Firmen überraschen hier durch deprimierend unglückliche Setups. Der bekannte Provider Tele2 beispielsweise meldet sich auch aktuell noch grundsätzlich im `HELO` mit der allgemeinen Domain `swip.net` (*Swedish IP-Network*, also der Muttergesellschaft `tele2.se`). Besser wäre der Reverse Lookup des Mailrelays (in der Art `mailfe01.tele2.de`) oder wenigstens die Domain des Mailrelays (`tele2.de`), nicht aber `swip.net` als eine technisch gesehen x-beliebige andere Domain. Da aber sonst alles in Ordnung ist und die Inbound-Mailserver für `tele2.de` zum IP-Bereich des Clients passen, sieht policyd-weight über diesen Fehler großzügig hinweg.

Anders jedoch die Situation bei der österreichischen Tele2. Auch dort passte der Reverse-Lookup (Beispiel: `mailfe01.tele2.at`) nicht zum `HELO`-Namen `swip.net`. Anders als bei `tele2.de` war jedoch auch die von Kunden genutzte Domain `tele2.at` zudem auf `rfc-ignorant.org` als spamfreundlich gelistet (siehe Kapitel 13.13 ab Seite 408). Das brachte das Fass für policyd-weight zum Überlaufen.

In der Folge wurden Mails der österreichischen Tele2 (in meinen Augen zu Recht) abgelehnt. Doch das „Drama“ hatte ein Happy End: Der österreichische Provider `iplace.at` konnte `tele2.at` von Nachbesserungen überzeugen, so dass vier Jahre (!) nach dem ersten Listing auf `rfc-ignorant.org` die Mailadressen `postmaster@tele2.at` und `abuse@tele2.at` eingerichtet wurden. `tele2.at` wurde folgerichtig wenig später nicht mehr auf `rfc-ignorant.org` gelistet. Damit konnte Tele2 Österreich wieder sauber E-Mails versenden – und wurde durch den Abuse-Account auch wieder seiner Verantwortung für einen möglichen Missbrauch durch seine Nutzer gerecht (Abbildung 9.1).

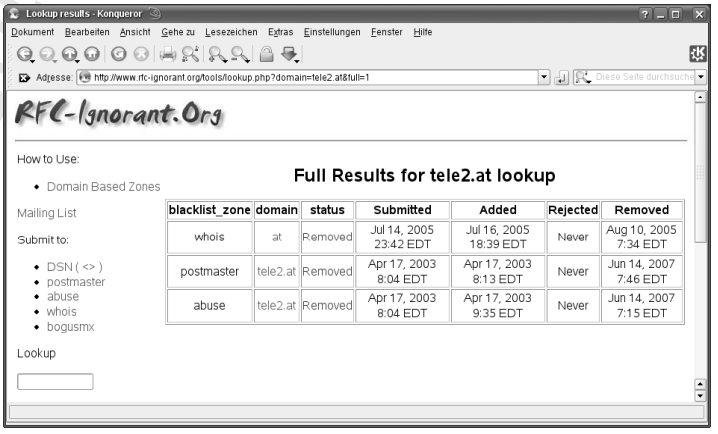

Abbildung 9.1: Auch `tele2.at` befolgt jetzt RFC 2822 und 2421 und konnte darum am 14. Juni 2007 von `rfc-ignorant.org` entfernt werden.

Ob kleine Firma oder großer Provider: In den hier vorgestellten Beispielen waren die Mailserver schlichtweg falsch und RFC-widrig konfiguriert. Die Postmaster hatten ihre Hausaufgaben nicht gemacht und dürfen sich über einen Block ihrer Server nicht wundern. Nicht nur policyd-weight, auch andere Anti-Spam-Systeme und alle großen Provider werden solche Fehler ahnden; wer einen Server in dieser Weise betreibt wird eben häufiger im Spamverdachtsordner landen.

Man kann in solchen Fällen auch kaum von einem „false positive“ sprechen, denn policyd-weight hat keinen Fehler gemacht: Wir wollen ja genau diese falsch konfigurierten Mailserver herausfiltern, die gefälschte oder ungültige Domains benutzen. Man muss hier (auch gegenüber Kunden) die Argumentation umkehren: Nicht der Empfänger trägt die Schuld und die Mail würde zu unrecht geblockt. Vielmehr hat der *Absender* die Mail nicht richtig versendet! Wenn per Post ein Brief ohne Postleitzahl, ohne Briefmarke oder ohne Anschrift verschickt wird, liegt ja auch kein Verschulden des Empfängers vor.

Analog ist die Situation im E-Mailverkehr. Wer einen sauber konfigurierten Mailserver mit richtigem DNS- und Hostnamen-Setup betreibt wird auch anstandslos versenden können. Wer RFC-widrig konfigurierte Server ans Netz lässt, muss sich nicht wundern, wenn er seine Mails nicht losbekommt.

Die zugrunde liegenden RFC-Vorschriften sind wichtig und sinnvoll – nicht zuletzt weil sie helfen, Spam und Missbrauch zu verhindern. Ob große oder kleine Firma – niemand sollte sich aus Bequemlichkeit über diese für alle (!) geltenden Vorschriften hinwegsetzen dürfen, schon gar nicht zu Lasten anderer. Es besteht kein Anlass, gerade bei großen ISPs beide Augen zuzudrücken und Ausnahmen zuzulassen – schon gar nicht, wenn man dadurch noch mehr Spam empfängt. Gerade diese sollten in der Lage sein, ein korrektes Setup auf die Beine zu stellen. Tele2 Österreich zeigt ja auch, dass Druck Änderungen bewirkt.

9.4 policyd

Kommen wir nun zum nächsten Policy-Dämon, denn auch der `policyd` erfreut sich zunehmender Beliebtheit. Er implementiert gleich mehrere Funktionen in einem einzigen Policy-Dämon:

Greylisting
: Nicht anders als `postgrey`: Greylisting eben.

Sender-/Recipient-Throttling
: Kann den Maildurchsatz auf Basis des Mailvolumens oder der Anzahl der E-Mails pro Zeiteinheit beschränken. Die Limits lassen sich

nicht nur an den Envelope-Mailadressen, sondern auch am SASL-Usernamen festmachen.

Spamtrap
: Kann einzelne IP-Adressen oder einen Netzblock für eine bestimmte Zeit blacklisten, wenn von dort bestimmte Test-Mailadressen („Spamtrap") beschickt wurden.

Blacklisting bei HELO-Fakes
: Blockt IP-Adressen oder ganze Netzblöcke, wenn von dort unser eigener Hostname oder unsere eigene IP-Adresse als gefaketer HELO übermittelt wurde.

HELO Randomization Prevention (HRP)
: Protokolliert, welche IP-Adressen sich mit welchem HELO gemeldet haben und sperrt IP-Adressen oder Netzblöcke, wenn sich die HELO-Angaben einer IP fortlaufend ändern (also vermutlich gefälscht werden).

Allgemeines White- und Blacklisting
: Ebenso kann man mit `policyd` ein allgemeines White- und Blacklisting für einzelne Empfänger durchsetzen; da der `policyd` seine Daten aus einer MySQL-Datenbank liest, lassen sich hier einfach entsprechende Nutzer-Interfaces anbinden.

Seine Stärken wie Spamtraps oder HRP kann der `policyd` erst auf großen Mailservern ausspielen, denn je mehr Mails er bekommt, umso wahrscheinlicher ist es, dass ein Spammer in die Falle läuft, und umso effektiver ist es, daraufhin den Spammer zu sperren.

Allerdings benötigt `policyd` eine MySQL-Datenbank – nicht jeder mag das auf seinem Mailserver. In HA-Umgebungen stellt sich zudem die Frage, wie MySQL entsprechend redundant abgesichert werden kann. Zudem ist `policyd` nicht in Perl, sondern in C geschrieben, was auch nicht jedem gefällt.

Doch genau diese beiden Eigenschaften werden sich ändern. Aktuell wird an der Versionsreihe 2.x gearbeitet (Codename „cluebringer"), von der im Mai 2008 die erste stabile Version 2.0.1 verfügbar war. Diese Version wird nicht mehr in C, sondern in Perl realisiert, so dass über entsprechende Perl-Module neben MySQL auch andere Datenbanken angebunden werden können.

In einigen Distributionen ist `policyd` bereits fertig paketiert enthalten; andernfalls müssen Sie sich von `http://www.policyd.org` den Quellcode besorgen. Da sich der `policyd` derzeit grundlegend verändert, sind dazu keine Detailangaben möglich. Es sollte jedoch unproblematisch sein, den Dämon gemäß den Anweisungen in der `README` selbst einzurichten. Importierbare Datenbank-Schemata sind ebenfalls im Paket enthalten.