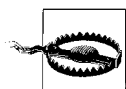


## KAPITEL 4

# SpamAssassin

SpamAssassin ist eine Software zur Erkennung von Spam. SpamAssassin analysiert eingehende E-Mails und errechnet eine Punktzahl, die anzeigt, mit welcher Wahrscheinlichkeit die E-Mail als Spam anzusehen ist. Anhand dieser Punktzahl kann der Anwender entscheiden, ob die E-Mail aussortiert werden soll. Um diese Punktzahl zu berechnen, verwendet SpamAssassin eine umfangreiche und erweiterbare Testsammlung, die Spam auf viele Arten identifizieren kann. Aufgrund dieses umfassenden Ansatzes ist SpamAssassin heute im Open Source-Bereich unangefochten die führende Software zur Spam-Bekämpfung.

Dieses Kapitel beschreibt die Funktionsweise von SpamAssassin, zeigt, wie man SpamAssassin anpassen und erweitern kann, und erläutert die Einbindung von SpamAssassin in vorhandene E-Mail-Systeme.

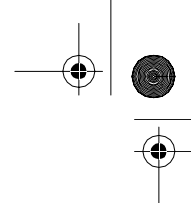


Dieses Kapitel bezieht sich ausschließlich auf SpamAssassin Version 3.0 oder später. Hinweise, die die Unterschiede zwischen dieser und den älteren Versionen der Reihe 2.x beschreiben, findet man auf der Webseite <http://spamassassin.apache.org/full/3.0.x/dist/UPGRADE>.

## Wie SpamAssassin arbeitet

Um eine E-Mail als Spam oder Nicht-Spam einzustufen, gibt es kein exaktes technisches Kriterium. Weder die technischen noch die juristischen Definitionen von Spam sind einheitlich. Aus diesem Grund kann SpamAssassin prinzipiell nicht bestimmen, ob eine bestimmte E-Mail tatsächlich Spam ist, sondern nur, wie wahrscheinlich dies ist. Um diese Wahrscheinlichkeit zu bestimmen, prüft SpamAssassin eine E-Mail auf verschiedene Anzeichen, die auf Spam hindeuten könnten. Dazu gehört unter anderem:

- das Vorkommen von verdächtigen Phrasen in der E-Mail
- das Vorkommen von bestimmten Header-Zeilen in der E-Mail
- die Ähnlichkeit der E-Mail mit zuvor als Spam eingestuft E-Mails
- die Analyse durch externe Spam-Erkennungstechniken



Die Summe der auf diese Art entdeckten verdächtigen Anzeichen bestimmt die Wahrscheinlichkeit, dass eine E-Mail Spam sein könnte, wobei die verschiedenen Kriterien unterschiedlich gewichtet sind.

Der Rest dieses Abschnitts beschreibt, welche Anzeichen zur Spam-Erkennung herangezogen werden und wie die Spam-Wahrscheinlichkeit berechnet wird.

## Tests und Punkte

SpamAssassin kodifiziert die verschiedenen Spam-Anzeichen in so genannten Tests. Ein Test ist eine Prüfungsvorschrift, die auf eine E-Mail angewendet wird und entweder erfolgreich ist oder nicht. Die meisten Tests prüfen, ob ein regulärer Ausdruck in der E-Mail vorkommt, aber es gibt auch andere Testarten. Zu jedem Test gehört eine Punktzahl (englisch: score). Die Summe der Punkte aller erfolgreichen Tests ist die Spam-Punktzahl der E-Mail.

Für die Bedeutung der Spam-Punktzahlen gilt Folgendes: E-Mails mit negativer Punktzahl sind eher nicht Spam; E-Mails mit positiver Punktzahl enthalten tendenziell Spam. Je höher die Punktzahl in der jeweiligen Richtung, desto wahrscheinlicher ist die Einstufung. Von den SpamAssassin-Entwicklern ist vorgegeben, dass eine E-Mail ab der Punktzahl 5 als Spam angesehen werden kann. Dies ist aber letzten Endes Geschmackssache. Nach der Lektüre dieses Kapitels kann der Leser selbst entscheiden, wie aggressiv oder konservativ die eigene SpamAssassin-Installation konfiguriert werden soll.

Die meisten Tests in SpamAssassin sind positive Tests, das heißt, sie testen auf Anzeichen von Spam. Es gibt jedoch auch einige negative Tests, die auf Anzeichen prüfen, dass eine E-Mail eher nicht Spam ist. Beide Arten von Tests funktionieren vollkommen identisch.

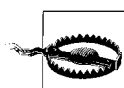
## Statische Tests

Der Großteil der Tests in SpamAssassin sind so genannte statische Tests. Sie prüfen einen bestimmten Aspekt der vorliegenden E-Mail und sind genau dann erfolgreich, wenn dieser Aspekt aufzufinden war. Diese Tests sind also unabhängig von früheren Läufen oder externen Datenbanken.

Die Zusammenstellung der statischen Tests hat sich im Laufe der Jahre aus den Beobachtungen und Erfahrungen der SpamAssassin-Entwickler und -Anwender ergeben. Viele Tests könnte der Gelegenheitsanwender selbst erstellen, zum Beispiel die zahlreichen Tests, die nach Werbephrasen in den E-Mails suchen, wie »Make money fast«, »Extra Cash« oder auch »Click to be removed«. Dazu kommen zahlreiche Phrasen, in denen Medizin, Drogen oder pornografisches Material angeboten wird. Einige Tests »bestrafen« E-Mails, die HTML enthalten. Dies mag kontrovers sein, Tatsache ist aber, dass E-Mails, die HTML enthalten, öfter Spam sind

als solche, die kein HTML enthalten. Weitere Tests prüfen die Kopfzeilen der E-Mails auf Verdachtsmomente. Dazu gehören zum Beispiel E-Mails ohne Betreff, Betreffzeilen mit vielen Ausrufezeichen, Absenderadressen ohne Namensangabe oder viele ähnliche Empfängeradressen, woraus man schließen könnte, dass diese aus einer Adressdatenbank stammen. Schließlich prüfen einige Tests auch, ob die interne Kodierung der E-Mail gültig ist oder Merkmale aufweist, die oft in Spam vorkommen.

Keiner dieser Tests ist absolut zuverlässig. Es kann sicherlich vorkommen, dass jemand einen Newsletter per E-Mail bestellt, der den Hinweis »Click to be removed« enthält. Ebenso ist das Senden von HTML-E-Mails in vielen Kreisen legitim. Fehlerhafte Header können gleichermaßen auf fehlerhaft konfigurierte E-Mail-Programme hindeuten. Entscheidend ist, dass diese Tests von den SpamAssassin-Entwicklern in einem aufwendigen Verfahren gegen eine große Anzahl tatsächlich versendeter E-Mails geprüft werden. Dabei werden die Punktzahlen der einzelnen Tests so gewichtet, dass eine Minimalzahl falscher Positive und falscher Negative auftreten. Einzelne Tests könnten also in einzelnen Fällen bei einer E-Mail falsch liegen. Die Gesamtheit der Tests und ihrer Gewichtungen kann aber als ausgesprochen treffsicher betrachtet werden.



Wenn die Endanwender des E-Mail-Systems in Bereichen tätig sind, die oft das Thema von Spam-E-Mails sind, zum Beispiel im Gesundheitswesen oder im Finanzwesen, insbesondere, wenn sie auch mit englischsprachiger Korrespondenz zu tun haben, sollte der Einsatz von SpamAssassin unbedingt sorgfältig evaluiert werden, und eventuell müssen die Tests angepasst werden, um den Verlust von legitimen E-Mails zu verhindern.

Der Nachteil von statischen Tests ist, insbesondere bei einer weit verbreiteten Software wie SpamAssassin, dass die Spammer die Tests ebenfalls kennen. In vielen Fällen werden Spam-E-Mails anscheinend auf SpamAssassin »optimiert«, so dass möglichst wenige Tests erfolgreich sind. Dies schränkt den Nutzen von statischen Tests erheblich ein. Ungeachtet dessen sind sie nach wie vor effektiv, und auch dynamische Tests sind nicht vor »Optimierung« gefeit. Wichtig ist auf jeden Fall, dass die Testsammlung regelmäßig angepasst und aktualisiert wird, was mit jedem SpamAssassin-Release geschieht.

## Dynamische Tests

Dynamische Tests im hier verwendeten Sinn sind SpamAssassin-Tests, die lernfähig sind. Durch Analyse des laufenden E-Mail-Verkehrs und durch gezieltes Training kann SpamAssassin die E-Mail-Bewertung an die spezifische Umgebung anpassen. Dazu bietet SpamAssassin zwei verschiedene Systeme: Bayessche Klassifizierung und Autowhitelisting.

## Bayessche Klassifizierung

Das Bayessche Klassifizierungssystem vergleicht den Inhalt einer neuen E-Mail mit den Inhalten der bisher als Spam oder Nicht-Spam klassifizierten E-Mails und kann daraus berechnen, wie wahrscheinlich es ist, dass die neue E-Mail Spam oder Nicht-Spam ist. Genauer gesagt, hält SpamAssassin eine Datenbank vor, die zählt, wie oft ein Wort bisher in Spam- und in Nicht-Spam-E-Mails vorgekommen ist, und kann anhand dieser Datenbank für eine neue E-Mail eine Wahrscheinlichkeit berechnen. Von SpamAssassin werden den Bayes-Wahrscheinlichkeiten Punkte zugewiesen, die in die Gesamtpunktzahl mit einfließen.

Der Name Bayes geht auf den britischen Mathematiker Thomas Bayes (1702-1761) zurück, dem ein Spezialfall des Satzes von Bayes, eines Satzes in der Wahrscheinlichkeitsrechnung, zugeschrieben wird. Die in der Praxis angewendeten Formeln haben aber genau genommen nichts mit dem Satz von Bayes zu tun. Die Namensgebung ist daher unzutreffend, aber sicherlich war der Satz von Bayes bei der Entwicklung dieser Technik inspirierend.

Als das Bayessche Klassifizierungssystem im Jahr 2002 vorgeschlagen wurde, revolutionierte es die Spam-Erkennung. Es war Spammern nun nicht mehr möglich, E-Mails einfach an den wohl bekannten statischen Erkennungsverfahren vorbeizuschleusen, da dieses neue System Spam quasi intelligent erkennen konnte. Mittlerweile haben auch die Spammer hinzugehört und stattdessen ihre E-Mails regelmäßig mit unsinnigem Fülltext aus, der die Bayes-Datenbank mit Müll füllen oder verfälschen soll, oder sie variieren ihre E-Mails einfach ausreichend, so dass sie keine Ähnlichkeit mit früherem Spam haben. Das Bayessche Klassifizierungssystem ist daher auch keine Wunderwaffe, aber ein nützliches und auf jeden Fall zu empfehlendes Werkzeug bei der Spam-Erkennung.

## Autowhitelisting

Das Autowhitelisting-System merkt sich, inwiefern bestimmte Absenderadressen und SMTP-Relays normalerweise Spam oder Nicht-Spam schicken, und verändert die Einstufung späterer E-Mails vom gleichen Absender beziehungsweise Relay in die entsprechende Richtung. Wenn ein legitimer Kommunikationspartner mit konstanter E-Mail-Adresse zum Beispiel normalerweise kein Spam sendet, aber eine E-Mail aus Versehen als Spam eingestuft wird, verändert das Autowhitelisting-System entsprechend diversen Konfigurationseinstellungen die Spam-Punktzahl der E-Mail in Richtung Nicht-Spam. Damit werden falsche Positive verringert. Der Kommunikationspartner wird also quasi automatisch »gewhitelistet«.

Das Autowhitelisting-System ist genauso ein Autoblacklisting-System. Wenn von einer E-Mail-Adresse normalerweise Spam gesendet wird, aber eine bestimmte E-Mail von diesem Absender nicht als Spam eingestuft wird, verändert das Autowhitelisting-System die Spam-Punktzahl der E-Mail in Richtung Spam. Da Spam aber wohl-

weislich selten von konstanten E-Mail-Adressen gesendet wird, ist diese Funktionalität in der Praxis unbedeutend.

Der Einsatz des Autowhitelisting-Systems ist unbedenklich. Wie gesagt, dient es aber hauptsächlich der Verhinderung von falschen Positiven, trägt also nicht zur aktiven Spam-Erkennung bei.

## Externe Software

Neben den eingebauten statischen und lernenden Tests kann SpamAssassin auch auf externe Spam-Erkennungstechniken zurückgreifen. Dazu gehören:

### *Pyzor, Razor*

Bei diesen Verfahren wird die vorliegende E-Mail über im Internet verfügbare Datenbanken mit bekannten Instanzen von Spam-E-Mails verglichen. Diese Techniken werden in Kapitel 6, *Zusätzliche Ansätze gegen Spam* im Detail beschrieben.

### *DNSBL*

Bei diesem Verfahren werden die Hosts, die die E-Mail auf ihrem Weg durch das Internet durchlaufen hat, in Datenbanken im Internet gesucht, die bekannte Spammer-Hosts enthalten. Diese Technik wird in Kapitel 5, *DNS-basierte Blackhole-Lists* im Detail beschrieben.

### *Hashcash*

Bei diesem Verfahren muss der Absender vor dem Versenden der E-Mail Rechenzeit investieren, um einen bestimmten Hash-Wert über die E-Mail zu berechnen, unter der Annahme, dass Spammer diese Zeit nicht aufbringen können. Diese Technik wird in Kapitel 6, *Zusätzliche Ansätze gegen Spam* beschrieben.

### *SPF*

Bei diesem Verfahren wird über zusätzliche DNS-Einträge bestimmt, ob der versendende Host berechtigt war, E-Mails aus der Absender-Domain zu senden. Diese Technik wird ebenfalls in Kapitel 6, *Zusätzliche Ansätze gegen Spam* beschrieben.

Diese Verfahren könnten auch unabhängig von SpamAssassin verwendet werden, um E-Mails auf Grund der entsprechenden Kriterien abzulehnen oder auszusortieren. In SpamAssassin werden diese Verfahren auf Tests abgebildet, die genauso wie andere Tests mit einer gewichteten Punktzahl zur endgültigen Spam-Punktzahl beitragen. Damit wird einerseits der Tatsache Rechnung getragen, dass keines dieser Verfahren hundertprozentig treffsicher ist, weshalb sie in SpamAssassin auch nur als eines von vielen Erkennungsverfahren verwendet werden. Andererseits erleichtert die Einbindung in SpamAssassin auch den administrativen Aufwand, da keine zusätzliche externe Konfiguration erforderlich ist.

## Eigene Tests

Der Anwender von SpamAssassin kann auch eigene Tests zu den vordefinierten Tests hinzufügen. Beim Entwerfen von eigenen Tests steht man natürlich selbst in der Verantwortung, die Tests so aufzubauen, dass sie keine falschen Positive erzeugen. Der Aufbau von Tests, die in das Gefüge der anderen SpamAssassin-Tests eingebaut und gewichtet werden sollen, ist nicht ganz einfach und für die meisten Anwender unnötig.

Genau genommen sind keine Tests in SpamAssassin eingebaut. Alle vordefinierten Tests sind über externe Dateien eingebunden, wie die, die der Anwender selbst schreiben könnte. Das Definieren eigener Tests wird später in diesem Kapitel beschrieben.

Einige Projekte und Einzelpersonen haben beträchtliche Testsammlungen zusammengetragen und stellen diese frei im Internet zur Verfügung. Weitere Informationen dazu findet man unter <http://wiki.apache.org/spamassassin/CustomRulesets>. Darunter finden sich auch einige Tests für deutschsprachigen E-Mail-Verkehr, wovon es unter den von SpamAssassin mitgelieferten Tests leider keine gibt. Externe Testsammlungen sind natürlich alle mit Vorsicht zu genießen und müssen sorgfältig evaluiert werden.

## Funktionsweise von SpamAssassin

Die Funktion von SpamAssassin ist es, eine E-Mail anhand von Tests zu analysieren und daraufhin die E-Mail möglicherweise zu verändern, indem zum Beispiel zusätzliche Header-Zeilen eingefügt werden. SpamAssassin kann selbst keine E-Mail löschen, ablehnen oder aussortieren. Dies muss in anderen Teilen des E-Mail-Systems auf Grundlage der von SpamAssassin eingefügten Header-Zeilen geschehen.

SpamAssassin kann durch drei Schnittstellen aufgerufen werden:

- ein allein stehendes Programm namens `spamassassin`
- ein Programm namens `spamc`, das mit einem separaten Serverprozess zusammenarbeitet
- ein Perl-Modul namens `Mail::SpamAssassin`

Die Einbindung von SpamAssassin in ein E-Mail-System geschieht über eine dieser drei Schnittstellen.

Das Programm `spamassassin` liest eine E-Mail auf der Standardeingabe, analysiert sie und gibt die möglicherweise modifizierte E-Mail auf der Standardausgabe wieder aus. Auf Grund dieser einfachen Schnittstelle kann dieses Programm auf viele Arten in ein E-Mail-System eingebunden oder auch zum manuellen Testen verwendet werden.

Beispiel 4-1 zeigt einen Aufruf von `spamassassin` mit einer sehr einfachen (an sich ungültigen) E-Mail aus einer Textdatei. Wie man sehen kann, hat SpamAssassin einige Header eingefügt, die darüber informieren, dass die E-Mail von SpamAssassin geprüft wurde, und über das Ergebnis der Prüfung. Das Format dieser Header kann man selbst konfigurieren.

*Beispiel 4-1: Aufruf von spamassassin von der Kommandozeile*

```
$ cat test.txt
From: test1@localhost
To: test2@localhost
Subject: Test

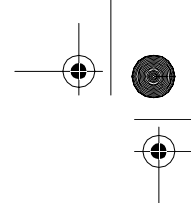
Nur ein Test
$ spamassassin < test.txt
From: test1@localhost
To: test2@localhost
Subject: Test
X-Spam-Checker-Version: SpamAssassin 3.0.2 (2004-11-16) on host.example.net
X-Spam-Level:
X-Spam-Status: No, score=-3.7 required=5.0 tests=ALL_TRUSTED,BAYES_00,
NO_REAL_NAME,SARE_MSGID_EMPTY,TO_MALFORMED autolearn=ham version=3.0.2
```

Nur ein Test

Erwähnenswert ist beim Programm `spamassassin` die Option `--local`. Damit werden alle Tests abgeschaltet, die Netzwerkzugriff benötigen (unter anderem DNSBL, Pyzor, DCC). Diese Option ist sinnvoll, wenn der Rechner, auf dem SpamAssassin ausgeführt wird, keine permanente oder nur eine langsame Internetverbindung hat.

Um SpamAssassin zu testen, ist es auch sinnvoll, es mit E-Mails zu füttern, die garantiert Spam sind. Um dies zu vereinfachen und zu vereinheitlichen, gibt es eine standardisierte Test-E-Mail, die von allen Spam-Filtern als Spam erkannt werden sollte. Diese E-Mail wird von SpamAssassin als Datei `sample-spam.txt` mitgeliefert. Außerdem gibt es eine Beispiel-E-Mail, die garantiert nicht Spam ist, unter dem Namen `sample-nospam.txt`.

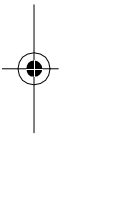
Das Programm `spamassassin` muss sich bei jedem Aufruf neu initialisieren, die Konfiguration und die umfangreiche Testsammlung einlesen. Dies beansprucht Rechenzeit und kann in großen E-Mail-Systemen zu Ressourcenproblemen führen. Daher ist es alternativ möglich, SpamAssassin als Server zu starten, der nur einmal initialisiert werden muss. Um eine E-Mail zu überprüfen, wird diese an ein spezielles Client-Programm übergeben, das sie an den Server zur eigentlichen Überprüfung sendet. Das Server-Programm heißt `spamd` und sollte, wenn der Einsatz gewünscht ist, wie andere Serverprozesse beim Booten des Systems gestartet werden. Das Client-Programm heißt `spamc` und hat, bis auf einige Kommandozeilen-Argumente, die gleiche Interface wie das Programm `spamassassin`. Es ist also sehr einfach, die allein stehende Variante durch die servergestützte Variante zu ersetzen. Beispiel 4-1 trifft analog zu.



Die servergestützte Variante sollte der allein stehenden Variante im normalen Betrieb vorgezogen werden, wenn keine außergewöhnlichen Umstände dem entgegenstehen. Der zusätzliche Administrationsaufwand ist minimal (ein zusätzlicher Serverprozess), und die Ressourcenersparnis kann erheblich sein. Da `spamc` und `spamd` über TCP/IP kommunizieren, ist es auch möglich, Client und Server auf verschiedenen Rechnern laufen zu lassen. Die notwendigen Kommunikationsparameter werden über Kommandozeilen-Optionen übergeben.

Schließlich kann SpamAssassin direkt über das Perl-Modul `Mail::SpamAssassin` angesprochen werden. Dieses Modul wird auch von den Programmen `spamassassin` und `spamd` intern verwendet. Der direkte Gebrauch des Perl-Moduls ist nicht für Endanwender gedacht, aber es gibt einige Programmpakete, die über diese Schnittstelle die SpamAssassin-Funktionalität integrieren. In diesen Fällen verwendet der Endanwender die Schnittstellen jenes Programmpakets. Beispiele dafür sind `Amavisd-new` und `MailScanner`, die später besprochen werden.

## Konfiguration



SpamAssassin-Einstellungen werden in verschiedenen Konfigurationsdateien vorgenommen. SpamAssassin sucht Konfigurationsdateien an mehreren Stellen im System. Die für ein bestimmtes System gültigen Pfade können der Manpage von `spamassassin` entnommen werden; hier werden stellvertretend die üblichen Pfade verwendet. Konfigurationsdaten werden an den folgenden Stellen gesucht:

### */usr/share/spamassassin/*

In diesem Verzeichnis liegen Konfigurationsdateien, die von SpamAssassin mitgeliefert werden und die Voreinstellungen und Standardtests bestimmen. Sie verwenden dieselbe Syntax und Semantik wie die anderen Konfigurationsdateien. Es ist nicht vorgesehen, dass diese Dateien von Anwendern verändert werden, aber sie können als umfangreiche Quelle von Beispielen dienen.

SpamAssassin liest alle Dateien mit der Endung `.cf` in diesem Verzeichnis, und zwar in alphanumerischer Reihenfolge. Üblicherweise beginnen die Namen der Dateien mit Zahlen, um, wenn nötig, die Reihenfolge der Verarbeitung zu bestimmen.

### */etc/mail/spamassassin/*

In diesem Verzeichnis liegen die globalen Konfigurationsdateien. Diese werden von allen SpamAssassin-Prozessen auf dem System (`spamassassin` oder `spamd`) verwendet und sind vom Administrator zu bearbeiten.

Hier gilt ebenso, dass alle Dateien mit der Endung `.cf` in alphanumerischer Reihenfolge gelesen werden. Wenn man nur wenige eigene Konfigurationseinstellungen hat, wird üblicherweise der Dateiname *`etc/mail/spamassassin/local.cf`* verwendet. In den meisten Installationen ist diese Datei bereits vorhanden und



enthält einige angepasste Einstellungen. Sollen umfangreichere Einstellungen vorgenommen werden, kann man auch zusätzliche Dateien anlegen.

#### *~/spamassassin/user\_prefs*

Diese Datei enthält benutzerspezifische Einstellungen. Dies ist sinnvoll, wenn jeder Anwender spamassassin nach eigenem Gutdünken aufruft. Wenn spamc/spamd verwendet werden, ist der Inhalt dieser Datei aus Sicherheitsgründen eingeschränkt; Erläuterungen dazu siehe unten. Wenn ein E-Mail-Filter-Gateway ohne lokale Benutzer eingerichtet wird, ist die Verwendung dieser benutzerspezifischen Konfigurationsdatei nicht sinnvoll.

SpamAssassin verwendet eine eigene Syntax für seine Konfigurationsdateien, die sich aber in vielerlei Hinsicht an diesbezüglichen Unix-Traditionen orientiert. Beispiel 4-2 zeigt eine einfache Variante einer Konfigurationsdatei, die als */etc/mail/spamassassin/local.cf* verwendet werden könnte.

#### *Beispiel 4-2: Eine einfache local.cf-Datei*

```
# Welcher Punktestand soll als Spam angesehen werden?
required_score 6.3

# Betreff von Spam-E-Mails nicht umschreiben.
rewrite_subject 0

# DNSBL-(RBL-)Prüfungen ausschalten.
skip_rbl_checks 1
```

Jede Einstellung, Direktive genannt, steht in einer Zeile. Kommentare beginnen mit # und enden am Zeilenende. Eine Einstellung besteht aus einem Wort am Zeilenanfang (zum Beispiel *required\_score*), gefolgt von einem oder mehreren Argumenten (zum Beispiel *6.3*).

Sämtliche Konfigurationsdateien werden bei jedem Aufruf des Programms spamassassin neu eingelesen. Bei der Verwendung des Server-Programms spamd trifft dies nicht zu; hier müssen Änderungen in der Konfiguration explizit aktiviert werden, indem dem Serverprozess das Signal HUP geschickt wird. Dies kann man auf den meisten Systemen auch mit dem Befehl */etc/init.d/spamd reload* oder */etc/init.d/spamassassin reload* erreichen.

In den folgenden Unterabschnitten werden die wichtigsten Konfigurationsdirektiven von SpamAssassin beschrieben. Eine vollständige Liste aller Konfigurationsdirektiven findet man auf der Manpage *Mail::SpamAssassin::Conf*.

## E-Mail-Markierung

Die Hauptaufgabe von SpamAssassin ist es, als Spam erkannte E-Mails zu markieren, damit sie später von anderen Teilen des E-Mail-Systems gefiltert oder aussortiert werden können. Wann diese Markierungen angewendet werden und wie sie

genau aussehen, kann man mit einigen Direktiven einstellen. In der Ausgangskonfiguration sind allerdings schon einige nützliche Einstellungen gemacht, so dass es nicht unbedingt erforderlich ist, sich mit dieser Materie zu beschäftigen.

Später in diesem Kapitel wird beschrieben, wie SpamAssassin in das Gesamt-E-Mail-System eingebunden wird. Dabei wird sich teilweise auf die hier eingefügten Markierungen bezogen.

#### `required_score`

Diese Direktive legt fest, ab welcher Punktzahl eine E-Mail als Spam eingestuft wird. Die Voreinstellung ist 5, was normalerweise ausreicht. Für vorsichtigerer Benutzer können höhere Werte sinnvoll sein. Niedrigere Werte sind relativ gefährlich, da sie die Gefahr von falschen Positiven erhöhen.

Prinzipiell sagt der absolute Wert der Punktzahl nichts aus; er ist nur relativ zu den den Tests zugewiesenen Punktzahlen zu sehen. Die von den SpamAssassin-Entwicklern ausgearbeiteten Tests und Punktzahlen sind auf den Wert 5 als Grenze zugeschnitten, und daher sollte man seine eigene Grenze auch in diesem Bereich festlegen.

#### `rewrite_header`

Mit dieser Direktive kann eingestellt werden, wie die Header-Zeilen Subject, From und To abgeändert werden, wenn eine E-Mail als Spam eingestuft worden ist. In der Voreinstellung werden keine dieser drei Header-Zeilen verändert.

Am häufigsten wird man den Betreff, also die Subject-Zeile, verändern wollen. Dabei wird dem ursprünglichen Betreff eine beliebige Zeichenkette vorangestellt. Denkbar wäre zum Beispiel folgende Konfiguration:

```
rewrite_header Subject ***SPAM***
```

Eine E-Mail mit der Betreffzeile

```
Subject: Check this out!
```

käme also, wenn sie als Spam eingestuft würde, mit der Betreffzeile

```
Subject: ***SPAM*** Check this out!
```

beim Empfänger an.

Soll die From- oder To-Zeile verändert werden, wird die Zeichenkette in Klammern vor die Adresse gestellt. Aus From: Sender <sender@example.net> würde zum Beispiel From: (spam) Sender <sender@example.net>.

Das Verändern der Header ist sinnvoll, wenn die Endanwender die E-Mails mit Systemen filtern, die nur anhand dieser Header filtern können, was zum Beispiel bei Outlook Express der Fall ist. Mächtigere Systeme wie Procmail oder moderne E-Mail-Clients können nach beliebigen Headern filtern; in solchen Fällen sollten lieber die unten beschriebenen SpamAssassin-spezifischen Header verwendet werden und Subject, From und To unverändert bleiben.

### add\_header

Mit dieser Direktive können zusätzliche Header in eine von SpamAssassin verarbeitete E-Mail eingefügt werden. Die Syntax der Direktive ist:

```
add_header TYP NAME STRING
```

Dabei ist *TYP* entweder *all* oder *ham* oder *spam*, wodurch bestimmt wird, ob der Header in alle E-Mails oder nur in Nicht-Spam oder nur in Spam eingefügt werden soll. *NAME* ist der Name des Headers, wobei vor den Namen immer *X-Spam-* gestellt wird. Aus `add_header Foo` wird also der Header *X-Spam-Foo*. *STRING* ist schließlich beliebiger Text, der den Header-Inhalt darstellt. In diesem Text können verschiedene Platzhalter verwendet werden. Eine Liste findet sich am Ende dieses Abschnitts.

In der Voreinstellung werden bereits einige Header in E-Mails eingefügt, um anzuzeigen, dass SpamAssassin aktiv war, und um das Filtern der E-Mail zu ermöglichen. Diese Voreinstellungen entsprechen folgenden Direktiven:

```
add_header spam Flag _YESNOCAPS_  
add_header all Status _YESNO_ , score=_SCORE_ required=_REQD_ tests=_TESTS_  
autolearn=_AUTOLEARN_ version=_VERSION_  
add_header all Level _STARS(*)_  
add_header all Checker-Version SpamAssassin _VERSION_ (_SUBVERSION_) on _  
HOSTNAME_
```

Beispiel 4-3 zeigt, wie diese Header in einer E-Mail aussehen. Die voreingestellten Header können mit `remove_header` oder `clean_headers` (siehe unten) entfernt oder durch zusätzliche `add_header`-Direktiven überschrieben werden. Der Header *X-Spam-Checker-Version* kann nicht verändert oder entfernt werden.

### remove\_header

Mit dieser Direktive kann ein mit `add_header` hinzugefügter Header entfernt werden – auch ein vordefinierter –, außer *X-Spam-Checker-Version*. Die Syntax ist:

```
remove_header TYP NAME
```

*TYP* und *NAME* haben die gleiche Bedeutung wie oben.

### clear\_headers

Mit dieser Direktive werden alle mit `add_header` hinzugefügten Header entfernt – auch alle vordefinierten –, außer *X-Spam-Checker-Version*. Die Direktive hat keine Argumente.

### report\_safe

Diese Direktive wählt eine von drei Varianten, wie SpamAssassin eine als Spam eingestufte E-Mail markieren soll. Wenn `report_safe 0` angegeben ist, fügt SpamAssassin lediglich einige *X-Spam-Header* ein und lässt die E-Mail ansonsten unverändert. Das Ergebnis dieser Einstellung ist in Beispiel 4-3 zu sehen.

*Beispiel 4-3: Spam-E-Mail mit report\_safe 0*

Subject: Test spam mail (GTUBE)  
Message-ID: <GTUBE1.1010101@example.net>  
Date: Wed, 23 Jul 2003 23:30:00 +0200  
From: Sender <sender@example.net>  
To: Recipient <recipient@example.net>  
Precedence: junk  
MIME-Version: 1.0  
Content-Type: text/plain; charset=us-ascii  
Content-Transfer-Encoding: 7bit  
X-Spam-Flag: YES  
X-Spam-Checker-Version: SpamAssassin 3.0.2 (2004-11-16) on ramrod.pezone.net  
X-Spam-Level: \*\*\*\*\*  
X-Spam-Status: Yes, score=994.4 required=5.0 tests=ALL\_TRUSTED,BAYES\_00,  
DNS\_FROM\_AHBL\_RHSBL,GTUBE autolearn=unavailable version=3.0.2  
X-Spam-Report:  
\* -3.3 ALL\_TRUSTED Did not pass through any untrusted hosts  
\* 1000 GTUBE BODY: Generic Test for Unsolicited Bulk Email  
\* -2.6 BAYES\_00 BODY: Bayesian spam probability is 0 to 1%  
\* [score: 0.0000]  
\* 0.3 DNS\_FROM\_AHBL\_RHSBL RBL: From: sender listed in dnsbl.ahbl.org

This is the GTUBE, the  
Generic  
Test for  
Unsolicited  
Bulk  
Email

If your spam filter supports it, the GTUBE provides a test by which you can verify that the filter is installed correctly and is detecting incoming spam. You can send yourself a test mail containing the following string of characters (in upper case and with no white spaces and line breaks):

XJS\*C4JDBQADN1.NSBN3\*2IDNEN\*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL\*C.34X

You should send this test mail from an account outside of your network.

Wenn report\_safe 1 angegeben ist (die Voreinstellung), wird statt der ursprünglichen E-Mail eine neue Nachricht aufgebaut, die einen informativen Text und die ursprüngliche E-Mail als Anhang mit dem MIME-Typ message/rfc822 enthält. Das Ergebnis dieser Einstellung ist in Beispiel 4-4 zu sehen.

*Beispiel 4-4: Spam-E-Mail mit report\_safe 1*

Received: from localhost by ramrod.pezone.net  
with SpamAssassin (version 3.0.2);  
Thu, 07 Apr 2005 17:54:41 +0200  
From: Sender <sender@example.net>  
To: Recipient <recipient@example.net>

*Beispiel 4-4: Spam-E-Mail mit report\_safe 1 (Fortsetzung)*

Subject: Test spam mail (GTUBE)  
Date: Wed, 23 Jul 2003 23:30:00 +0200  
Message-Id: <GTUBE1.1010101@example.net>  
X-Spam-Flag: YES  
X-Spam-Checker-Version: SpamAssassin 3.0.2 (2004-11-16) on ramrod.pezone.net  
X-Spam-Level: \*\*\*\*\*  
X-Spam-Status: Yes, score=994.4 required=5.0 tests=ALL\_TRUSTED,BAYES\_00,  
DNS\_FROM\_AHBL\_RHSBL,GTUBE autolearn=unavailable version=3.0.2  
MIME-Version: 1.0  
Content-Type: multipart/mixed; boundary="-----=\_425557C1.B535E32C"

This is a multi-part message in MIME format.

-----=\_425557C1.B535E32C  
Content-Type: text/plain  
Content-Disposition: inline  
Content-Transfer-Encoding: 8bit

Spam detection software, running on the system "ramrod.pezone.net", has identified this incoming email as possible spam. The original message has been attached to this so you can view it (if it isn't spam) or label similar future email. If you have any questions, see the administrator of that system for details.

Content preview: This is the GTUBE, the Generic Test for Unsolicited Bulk Email. If your spam filter supports it, the GTUBE provides a test by which you can verify that the filter is installed correctly and is detecting incoming spam. You can send yourself a test mail containing the following string of characters (in upper case and with no white spaces and line breaks): [...]

Content analysis details: (994.4 points, 5.0 required)

pts	rule name	description
-3.3	ALL_TRUSTED	Did not pass through any untrusted hosts
1000	GTUBE	BODY: Generic Test for Unsolicited Bulk Email
-2.6	BAYES_00	BODY: Bayesian spam probability is 0 to 1% [score: 0.0000]
0.3	DNS_FROM_AHBL_RHSBL	RBL: From: sender listed in dnsbl.ahbl.org

-----=\_425557C1.B535E32C  
Content-Type: message/rfc822; x-spam-type=original  
Content-Description: original message before SpamAssassin  
Content-Disposition: inline  
Content-Transfer-Encoding: 8bit

Subject: Test spam mail (GTUBE)  
Message-ID: <GTUBE1.1010101@example.net>

*Beispiel 4-4: Spam-E-Mail mit report\_safe 1 (Fortsetzung)*

Date: Wed, 23 Jul 2003 23:30:00 +0200  
From: Sender <sender@example.net>  
To: Recipient <recipient@example.net>  
Precedence: junk  
MIME-Version: 1.0  
Content-Type: text/plain; charset=us-ascii  
Content-Transfer-Encoding: 7bit

This is the GTUBE, the  
Generic  
Test for  
Unsolicited  
Bulk  
Email

If your spam filter supports it, the GTUBE provides a test by which you can verify that the filter is installed correctly and is detecting incoming spam. You can send yourself a test mail containing the following string of characters (in upper case and with no white spaces and line breaks):

XJS\*C4JDBQADN1.NSBN3\*2IDNEN\*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL\*C.34X

You should send this test mail from an account outside of your network.

-----=\_425557C1.B535E32C--

Wenn report\_safe 2 angegeben ist, wird wiederum die ursprüngliche E-Mail als Anhang an eine neue E-Mail-Nachricht gehängt, hier aber mit MIME-Typ text/plain. Das Ergebnis dieser Einstellung ist in Beispiel 4-5 zu sehen.

*Beispiel 4-5: Spam-E-Mail mit report\_safe 2*

Received: from localhost by ramrod.pezone.net  
with SpamAssassin (version 3.0.2);  
Thu, 07 Apr 2005 17:54:41 +0200  
From: Sender <sender@example.net>  
To: Recipient <recipient@example.net>  
Subject: Test spam mail (GTUBE)  
Date: Wed, 23 Jul 2003 23:30:00 +0200  
Message-Id: <GTUBE1.1010101@example.net>  
X-Spam-Flag: YES  
X-Spam-Checker-Version: SpamAssassin 3.0.2 (2004-11-16) on ramrod.pezone.net  
X-Spam-Level: \*\*\*\*\*  
X-Spam-Status: Yes, score=994.4 required=5.0 tests=ALL\_TRUSTED,BAYES\_00,  
DNS\_FROM\_AHBL\_RHSBL,GTUBE autolearn=unavailable version=3.0.2  
MIME-Version: 1.0  
Content-Type: multipart/mixed; boundary="-----=\_42555A8C.2C6011E0"

*Beispiel 4-5: Spam-E-Mail mit report\_safe 2 (Fortsetzung)*

This is a multi-part message in MIME format.

-----=\_42555A8C.2C6011E0  
Content-Type: text/plain  
Content-Disposition: inline  
Content-Transfer-Encoding: 8bit

Spam detection software, running on the system "ramrod.pezone.net", has identified this incoming email as possible spam. The original message has been attached to this so you can view it (if it isn't spam) or label similar future email. If you have any questions, see the administrator of that system for details.

Content preview: This is the GTUBE, the Generic Test for Unsolicited Bulk Email. If your spam filter supports it, the GTUBE provides a test by which you can verify that the filter is installed correctly and is detecting incoming spam. You can send yourself a test mail containing the following string of characters (in upper case and with no white spaces and line breaks): [...]

Content analysis details: (994.4 points, 5.0 required)

pts	rule name	description
-3.3	ALL_TRUSTED	Did not pass through any untrusted hosts
1000	GTUBE	BODY: Generic Test for Unsolicited Bulk Email
-2.6	BAYES_00	BODY: Bayesian spam probability is 0 to 1% [score: 0.0000]
0.3	DNS_FROM_AHBL_RHSBL	RBL: From: sender listed in dnsbl.ahbl.org

-----=\_42555A8C.2C6011E0  
Content-Type: text/plain; x-spam-type=original  
Content-Description: original message before SpamAssassin  
Content-Disposition: inline  
Content-Transfer-Encoding: 8bit

Subject: Test spam mail (GTUBE)  
Message-ID: <GTUBE1.1010101@example.net>  
Date: Wed, 23 Jul 2003 23:30:00 +0200  
From: Sender <sender@example.net>  
To: Recipient <recipient@example.net>  
Precedence: junk  
MIME-Version: 1.0  
Content-Type: text/plain; charset=us-ascii  
Content-Transfer-Encoding: 7bit

This is the GTUBE, the  
Generic  
Test for

*Beispiel 4-5: Spam-E-Mail mit report\_safe 2 (Fortsetzung)*

```
Unsolicited
Bulk
Email
```

If your spam filter supports it, the GTUBE provides a test by which you can verify that the filter is installed correctly and is detecting incoming spam. You can send yourself a test mail containing the following string of characters (in upper case and with no white spaces and line breaks):

```
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X
```

You should send this test mail from an account outside of your network.

```
-----=_42555A8C.2C6011E0--
```

Wie der Name der Direktive andeutet, ist das Ziel der Einstellungen 1 und 2, den Empfänger der E-Mail vor unsicheren E-Mails zu schützen. Der Schutz bezieht sich dabei zum Beispiel auf das unvorsichtige Klicken auf mitgesendete Links, das Öffnen von Anhängen oder gar das Antworten auf die E-Mail. Wenn die Endanwender im Umgang mit E-Mails unerfahren sind, ist Einstellung 1 die richtige, weil sie dann zuerst die Warnmeldung über den Spam sehen und die ursprüngliche E-Mail von Hand öffnen müssen. Sind die Endanwender erfahren und lassen Spam automatisch aussortieren, ist Einstellung 1 wahrscheinlich eher lästig.

Einstellung 2 ist als Alternative zu Einstellung 1 vorgesehen, falls E-Mail-Clients mit dem MIME-Typ `message/rfc822` nicht vernünftig umgehen können und solche Anhänge automatisch öffnen. Einstellung 2 verhindert dies, macht es aber schwieriger, die ursprüngliche, unveränderte E-Mail aus dem Anhang her auszugewinnen.

`report`

Mit dieser Direktive kann der Hinweistext, der in Beispiel 4-4 und Beispiel 4-5 zu sehen ist, geändert werden. Im Text können verschiedene Platzhalter verwendet werden. Eine Liste befindet sich am Ende dieses Abschnitts. Jede `report`-Direktive hängt eine Zeile an den Text an. Um mit einem neuen Text anzufangen, muss zuerst `clear_report_template` aufgerufen werden. Die originale Definition des Hinweistexts befindet sich in der Datei `10_misc.cf` im Verzeichnis `/usr/share/spamassassin/`. Sie kann als Ausgangspunkt für Anpassungen oder Übersetzungen dienen.

`clear_report_template`

Diese Direktive löscht den Hinweistext; siehe oben.



### report\_charset

Diese Direktive bestimmt den Zeichensatz für den Hinweistext. In der Ausgangskonfiguration ist kein Zeichensatz eingestellt. Wenn der Hinweistext so geändert wird, dass er nicht nur ASCII-Zeichen enthält, zum Beispiel bei einer Übersetzung, muss hier der Zeichensatz angegeben werden, beispielsweise ISO-8859-1 oder UTF-8.

Bei einigen der eben beschriebenen Direktiven können im Text Platzhalter verwendet werden, die in der eigentlichen E-Mail durch anderen Text ersetzt werden. Die vollständige Liste der Platzhalter findet man in der SpamAssassin-Dokumentation. Folgende Platzhalter seien exemplarisch genannt:

#### `_YESNO_`

»Yes« oder »No«, abhängig davon, ob die E-Mail als Spam eingestuft wurde. Dies wird typischerweise im Header `X-Spam-Flag` oder `X-Spam-Status` verwendet.

#### `_YESNOCAPS_`

»YES« oder »NO«, abhängig davon, ob die E-Mail als Spam eingestuft wurden (wie zuvor, nur in Großbuchstaben).

#### `_SCORE_`

Die Spam-Punktzahl der E-Mail, zum Beispiel »4.2«.

#### `_REQD_`

Die Punktzahl, ab der eine E-Mail als Spam eingestuft wird (eingestellt mit `required_score`).

#### `_HOSTNAME_`

Der Name des Rechners, auf dem SpamAssassin läuft.

#### `_STARS(*)_`

Stellt die Spam-Punktzahl symbolisch durch die Wiederholung eines Symbols dar, das in den Klammern angegeben wird (der Stern ist nur ein gebräuchliches Beispiel). Voreingestellt ist folgender Header:

```
add_header all Level _STARS(*)_
```

Dieser Header kann nützlich zum automatischen Sortieren von Spam mit Procmail oder ähnlicher Software sein. (Er ist einfacher zu parsen als eine numerische Angabe.) Da der Stern ein Sonderzeichen in der Syntax der regulären Ausdrücke ist, bietet es sich an, ein anderes Zeichen zu verwenden, damit das Schreiben der Procmail-Regeln einfacher ist.

## Spracheinstellungen

Ein relativ eindeutiges Anzeichen von Spam ist es, wenn er in einer Sprache daherkommt, die man sowieso nicht versteht. SpamAssassin kann so eingestellt werden, dass es E-Mail in bestimmten Sprachen oder in bestimmten Zeichensätzen mit

Spam-Punkten bestraft. Die Bestimmung der Sprache ist selbstverständlich ungenau, die Bestimmung des Zeichensatzes ist dagegen verlässlich.

Selbstverständlich muss man alle diese Einstellungen an die eigenen Umstände anpassen.

Die dazugehörigen Direktiven sind:

#### ok\_languages

Diese Direktive bestimmt, welche Sprachen in E-Mail erlaubt sind, also *nicht* als Spam betrachtet werden. Als Argumente werden ein oder mehrere Sprachkürzel aufgelistet, zum Beispiel

```
ok_languages de en fr nl
```

für Deutsch, Englisch, Französisch und Niederländisch. Die Sprachkürzel entsprechen größtenteils den im Internetbereich bekannten Codes nach ISO 639. Die vollständige Liste der von SpamAssassin erkannten Sprachen findet sich jedoch in der SpamAssassin-Dokumentation.

Um alle Sprachen zu erlauben, verwendet man `ok_languages all`. Dies ist auch die Voreinstellung.

Wenn eine E-Mail eine unerwünschte Sprache enthält, springt der Test `UNWANTED_LANGUAGE_BODY` an, der in der Voreinstellung mit 2,8 Punkten bewertet ist.

#### ok\_locales

Diese Direktive bestimmt, welche Zeichensätze in E-Mails erlaubt sind. Dies ist eine gröbere Einstellung als die obige Sprachauswahl, ist aber auch genauer, weil der Zeichensatz einer E-Mail sicher bestimmt werden kann. Als Argumente werden ein oder mehrere Sprachkürzel aufgelistet, die stellvertretend für die von diesen Sprachen verwendeten Zeichensätze stehen. Folgende Kürzel stehen zur Verfügung:

en

Zeichensätze für westeuropäische Sprachen (zum Beispiel für Englisch)

ja

Zeichensätze für Japanisch

ko

Zeichensätze für Koreanisch

ru

kyrillische Zeichensätze (zum Beispiel für Russisch)

th

Zeichensätze für Thai

zh

Zeichensätze für Chinesisch (vereinfachte und traditionelle Varianten)

Für Anwender im deutschsprachigen Raum sollte zumindest en ausgewählt werden sowie andere Zeichensätze nach persönlicher Wahl.

Um alle Sprachen zu erlauben, verwendet man `ok_locales all`. Dies ist auch die Voreinstellung.

Wenn eine E-Mail einen unerwünschten Zeichensatz enthält, springt entweder der Test `CHARSET_FARAWAY` (Header-Test) oder der Test `CHARSET_FARAWAY_BODY` (Body-Test) an, die beide in der Voreinstellung mit 3,2 Punkten bewertet sind.

## Whitelists und Blacklists

Mit Hilfe von Whitelists und Blacklists können E-Mails mit bestimmten Absender- und Empfängeradressen direkt als Spam oder Nicht-Spam eingestuft werden. Dies ist nützlich, wenn SpamAssassin mit E-Mails von bestimmten Kommunikationspartnern wiederholt Fehler macht. Man könnte zum Beispiel die E-Mail-Adressen von wichtigen Geschäftspartnern »whitelisten«, um sicherzugehen, dass keine wichtige E-Mail verloren geht. Oder man könnte die E-Mail-Adressen von Anwendern »whitelisten«, die keine Filterung durch SpamAssassin wünschen. Die Blacklists eignen sich zum Ablehnen von E-Mails mit bestimmten Absenderadressen.

Die Whitelist- und Blacklist-Direktiven erwarten als Argumente reine E-Mail-Adressen, bestehend aus Benutzername und Domain und ohne Namen und Kommentare. Als Wildcards können »?« für ein beliebiges Zeichen und »\*« für eine beliebige Zeichenkette verwendet werden; volle reguläre Ausdrücke können nicht verwendet werden.

### `whitelist_from`

Diese Direktive nimmt eine Absenderadresse in die Whitelist auf. Absenderadressen sind Adressen in einem der Header `Resent-From`, `Envelope-Sender`, `X-Envelope-From` und `From`. Um mehrere Adressen in die Liste aufzunehmen, können mehrere Adressen angegeben werden, oder die Direktive kann wiederholt werden.

Beispiel:

```
whitelist_from *@example.com *@example.net
```

Wird eine Absenderadresse in der Whitelist gefunden, wird der Test `USER_IN_WHITELIST` als erfolgreich bewertet. Dieser Test hat in der Voreinstellung -100 Punkte.

### `unwhitelist_from`

Mit dieser Direktive wird eine Absenderadresse aus der Whitelist entfernt. Dies ist zum Beispiel nützlich, wenn ein Anwender in seiner benutzerspezifischen Konfigurationsdatei Adressen aus der globalen Whitelist entfernen möchte. Es können auch mehrere Adressen angegeben werden.

#### whitelist\_from\_rcvd

Diese Direktive nimmt eine Absenderadresse in die Whitelist auf und führt zusätzlich eine Reverse-DNS-Prüfung über den letzten Host durch, den die E-Mail durchlaufen hat, bevor sie auf dem lokalen Mailserver ankam. Das ist unter anderem nützlich, wenn man seine eigene Domain »whitelisten« möchte, da Spammer gern die Empfänger-Domain in der (gefälschten) Absenderadresse verwenden. Als erstes Argument wird die E-Mail-Adresse angegeben und als zweites Argument der Hostname oder der Domainname des letzten Relays.

Beispiel:

```
whitelist_from_rcvd *@mydomain.de mydomain.de
whitelist_from_rcvd *@mydomain.com mydomain.de
```

#### whitelist\_allows\_relays

Wenn eine E-Mail mit einer Absenderadresse, die in `whitelist_from_rcvd` eingetragen ist, über ein anderes Relay eingeht, wird dies normalerweise als Fälschung betrachtet. Wenn es aber legitim ist, dass diese Absenderadresse auch andere Relays verwendet, dann muss die E-Mail-Adresse in `whitelist_allows_relays` eingetragen werden.

Beispiel:

```
whitelist_allows_relays *@mydomain.de *@mydomain.com
```

#### unwhitelist\_from\_rcvd

Mit dieser Direktive wird eine Absenderadresse aus der mit `whitelist_from_rcvd` angelegten Whitelist entfernt.

#### blacklist\_from

Diese Direktive nimmt eine Absenderadresse in die Blacklist auf. Es können auch mehrere Adressen angegeben werden.

Wird eine Absenderadresse in der Blacklist gefunden, wird der Test `USER_IN_BLACKLIST` als erfolgreich bewertet. Dieser Test hat in der Voreinstellung 100 Punkte.

#### unblacklist\_from

Mit dieser Direktive wird eine Absenderadresse aus der Blacklist entfernt. Es können auch mehrere Adressen angegeben werden.

#### whitelist\_to

#### more\_spam\_to

#### all\_spam\_to

Diese Direktiven nehmen eine Empfängeradresse in die Whitelist auf. Empfängeradressen sind Adressen in einem der Header `To`, `Cc`, `Resent-To` und in verschiedenen weiteren. Um mehrere Adressen in die Liste aufzunehmen, können mehrere Adressen angegeben oder die Direktive kann wiederholt werden.

Beispiel:

```
whitelist_to *@example.com *@example.net
```

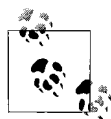
Die drei Direktiven unterscheiden sich lediglich durch die Anzahl der Punkte, die sie der erkannten E-Mail geben. `whitelist_to` entspricht der Regel `USER_IN_WHITELIST_TO` mit  $-6$  Punkten, `more_spam_to` entspricht der Regel `USER_IN_MORE_SPAM_TO` mit  $-20$  Punkten- und `all_spam_to` entspricht der Regel `USER_IN_ALL_SPAM_TO` mit  $-100$  Punkten.

Es gibt keine Direktive `unwhitelist_to`, da eine Empfänger-Whitelist nur global Sinn macht. Wenn einzelne Benutzer die Empfänger-Whitelist übergehen wollen, müssen sie nur den Wert für `required_score` ändern.

#### `blacklist_to`

Diese Direktive nimmt eine Empfängeradresse in die Blacklist auf. Es können auch mehrere Adressen angegeben werden. Empfänger-Blacklists sind nützlich, um Spam herauszufiltern, der bestimmte gefälschte Adressen im To-Feld verwendet.

Wird eine Empfängeradresse in der Blacklist gefunden, wird der Test `USER_IN_BLACKLIST_TO` als erfolgreich bewertet. Dieser Test hat in der Voreinstellung 10 Punkte.



Es ist in der Regel effektiver, eine Blacklist im MTA einzurichten und die E-Mail sofort abzulehnen, anstatt sie erst anzunehmen, Spam-Assassin laufen zu lassen und sie dann zum Beispiel von Procmail aussortieren zu lassen.

## Externe Testsysteme

Wie oben angedeutet, kann SpamAssassin verschiedene externe Testsysteme in die Spam-Bewertung einbeziehen. Mit folgenden Direktiven können diese selektiv ein- und ausgeschaltet werden.

#### `use_dcc`

Schaltet die Verwendung von DCC ein (`use_dcc 1`) oder aus (`use_dcc 0`). Die Voreinstellung ist an.

#### `use_pyzor`

Schaltet die Verwendung von Pyzor ein (`use_pyzor 1`) oder aus (`use_pyzor 0`). Die Voreinstellung ist an.

#### `use_razor2`

Schaltet die Verwendung von Razor Version 2 ein (`use_razor2 1`) oder aus (`use_razor2 0`). Die Voreinstellung ist an.

#### `skip_rbl_checks`

Schaltet die Verwendung von DNSBL-(RBL-)Prüfungen aus (`skip_rbl_checks 1`) oder ein (`skip_rbl_checks 0`). Die Voreinstellung ist 0, also DNSBL-Prüfungen sind an.

In den Fällen DCC, Pyzor und Razor gilt, dass das Feature nur eingeschaltet wird, falls die entsprechende Software installiert ist. Dies wird beim Starten von SpamAssassin geprüft. Es ist daher unbedenklich, alle Direktiven zunächst eingeschaltet zu lassen.

## Einstellungen des Bayes-Systems

Das Bayessche Klassifizierungssystem in SpamAssassin kann durch eine Vielzahl von Konfigurationsdirektiven sehr detailliert eingestellt werden. Hier werden nur die wichtigsten beschrieben.

### use\_bayes

Diese Direktive schaltet die Verwendung des Bayesschen Klassifizierungssystems komplett ein (`use_bayes 1`) oder aus (`use_bayes 0`). Die Voreinstellung ist an.

### use\_bayes\_rules

Wenn diese Option ausgeschaltet ist (0), werden die Ergebnisse der Bayesschen Klassifizierung nicht in die Spam-Bewertung einbezogen, der Rest des Systems läuft jedoch weiter. Die Voreinstellung ist 1. Mit dieser Option kann man das Bayes-System vorübergehend ausschalten, aber das Lernen weiterlaufen lassen.

### bayes\_auto\_learn

Wenn diese Option eingeschaltet ist (1), werden E-Mails, die eindeutig als Spam oder eindeutig als Nicht-Spam eingestuft worden sind, automatisch zum Lernen an das Bayes-System übergeben. Die Voreinstellung ist 1. Die Grenzen der »Eindeutigkeit« werden über die Direktiven `bayes_auto_learn_threshold_*` eingestellt. Bei der Ermittlung der Punktzahl zum automatischen Lernen werden die Punkte, die von den Bayes-Tests selbst ermittelt wurden, nicht mit einbezogen.

Die Verwendung des automatischen Lernsystems widerspricht streng genommen dem ursprünglichen Geist des Bayesschen Verfahrens, nämlich dass der Anwender sein eigenes E-Mail-Aufkommen nach seinem Geschmack klassifiziert, während dies hier letzten Endes SpamAssassin übernimmt. Trotzdem hat das automatische Lernsystem große praktische Effektivität bewiesen, kann das manuelle Lernen aber zumindest nicht vollständig ersetzen.

### bayes\_auto\_learn\_threshold\_nospam

### bayes\_auto\_learn\_threshold\_spam

Diese Direktiven bestimmen die Punktzahl, unter der E-Mails automatisch als Nicht-Spam erlernt werden, sowie die Punktzahl, über der E-Mails automatisch als Spam erlernt werden. Die Voreinstellungen sind 0,1 und 12,0. Wenn durch eigene Tests oder besondere Anforderungen die Spam-Erkennungsgrenze von 5,0 erheblich abweicht, sollten eventuell auch diese Punktzahlen geändert werden.

#### bayes\_ignore\_header

Mit dieser Direktive können bestimmte Header vor dem Bayes-System versteckt werden. Wenn zum Beispiel ein weiteres Spam-Erkennungssystem vorgeschaltet ist, etwa beim ISP, das bestimmte Header in E-Mails einfügt, würden diese Header mit ziemlicher Sicherheit sehr schnell in der Bayes-Datenbank unter Spam landen. Obwohl dies prinzipiell erst mal nicht falsch ist, beeinflusst es doch die Unabhängigkeit des Bayes-Systems. Daher sollte man solche Header – und nur solche – vom Bayes-System ignorieren lassen. Ein Beispiel:

```
bayes_ignore_header X-Hokuspokus-Spamfilter
```

#### bayes\_ignore\_from

#### bayes\_ignore\_to

Mit diesen Direktiven werden E-Mails mit den angegebenen Absender- beziehungsweise Empfängeradressen vom Bayes-System ausgenommen. Sinnvoll ist dies insbesondere für die Postmaster-Adresse, zum Beispiel:

```
bayes_ignore_to postmaster@mysite.com
```

Um mehrere Adressen anzugeben, verwendet man entweder Wildcards (\* und ?), wie bei White- und Blacklists), gibt mehrere Adressen an oder schreibt die Direktive mehrmals.

#### bayes\_path

Diese Direktive bestimmt, wo die Bayes-Datenbank abgespeichert wird. Die Voreinstellung ist

```
bayes_path ~/.spamassassin/bayes
```

Damit erhält jeder Benutzer eine private Bayes-Datenbank im Verzeichnis *~/.spamassassin/* in mehreren Dateien, deren Name mit *bayes* anfängt, nämlich *bayes\_journal*, *bayes\_seen* und *bayes\_toks*, sowie einigen kurzfristigen Sperrdateien. Die Endungen werden automatisch angehängt.

Mit dieser Direktive könnte man zum Beispiel eine globale Bayes-Datenbank für alle Benutzer erzwingen, etwa weil die Benutzer keine Home-Verzeichnisse haben oder weil Festplattenplatz gespart werden soll.

Das Ergebnis des Bayesschen Klassifizierungssystems ist eine Wahrscheinlichkeit, dass die E-Mail Spam ist. Der Wert der Wahrscheinlichkeit ist zwischen 0 und 1 oder zwischen 0% und 100%. Werte unter 50% sind eher kein Spam, Werte über 50% dagegen eher Spam. Dieser Wahrscheinlichkeitswert wird auf SpamAssassin-Tests mit Namen wie *BAYES\_60* abgebildet. Tabelle 4-1 zeigt die Zuordnung von der Wahrscheinlichkeit zu den Tests sowie die zugehörigen Spam-Punktzahlen. Es ist richtig, dass die Punktzahlen teilweise nicht stetig ansteigen. Dies liegt am automatischen Gewichtungsprozess, den alle Tests durchlaufen. Es ist zum Beispiel anzunehmen, dass E-Mails im Bereich 99% bis 100% mit eingeschalteten Netzwerktests schon von vielen anderen Tests erkannt werden, sodass eine hohe Gewichtung des Bayes-Tests nicht erforderlich erscheint.

Tabelle 4-1: Bayes-Tests

Testname	Untere Grenze	Obere Grenze	Punktzahl ohne Netzwerktests	Punktzahl mit Netzwerktests
BAYES_00	0%	1%	-1,665	-2,599
BAYES_05	1%	5%	-0,925	-0,413
BAYES_20	5%	20%	-0,730	-1,951
BAYES_40	20%	40%	-0,276	-1,096
BAYES_50	40%	60%	1,567	0,001
BAYES_60	60%	80%	3,515	0,372
BAYES_80	80%	95%	3,608	2,087
BAYES_95	95%	99%	3,514	2,063
BAYES_99	99%	100%	4,070	1,886

## Einstellungen des Autowhitelisting-Systems

Auch das Autowhitelisting-System kann durch eine Vielzahl von Einstellungen konfiguriert werden. Hier werden aber nur die für die Praxis wichtigsten vorgestellt:

### `use_auto_whitelist`

Mit dieser Direktive wird das Autowhitelisting-System eingeschaltet (1) oder ausgeschaltet (0). Die Voreinstellung ist 1.

### `auto_whitelist_path`

Diese Direktive bestimmt, wo die Autowhitelist-Datenbank abgespeichert wird. Die Voreinstellung ist:

```
auto_whitelist_path ~/.spamassassin/auto-whitelist
```

Damit erhält jeder Benutzer eine private Autowhitelist-Datenbank in der Datei `~/.spamassassin/auto-whitelist` mit einigen zugehörigen kurzfristigen Sperrdateien im selben Verzeichnis.

Mit dieser Direktive könnte man zum Beispiel eine globale Autowhitelist-Datenbank für alle Benutzer erzwingen, etwa weil die Benutzer keine Home-Verzeichnisse haben oder weil Festplattenplatz gespart werden soll.

## Direktiven für eigene Tests

Die in diesem Abschnitt aufgeführten Direktiven werden beim Schreiben von eigenen Tests verwendet. Konfigurationseinstellungen und Testdefinitionen werden von SpamAssassin von der Syntax her nicht unterschieden.

Weitere Informationen zum Schreiben eigener Tests findet man unter »Definition eigener Tests«. Die dazugehörigen Direktiven werden hier nur kurz angerissen, um den Überblick über die Konfigurationseinstellungen zu komplettieren.



#### score

Diese Direktive bestimmt die Punktzahl, die einem Test zugewiesen wird. Einem Test können bis zu vier Punktzahlen zugewiesen werden, die unter verschiedenen Umständen verwendet werden, nämlich in dieser Reihenfolge: ohne Bayes- und Netzwerktests, ohne Bayes- und mit Netzwerktests, mit Bayes- und ohne Netzwerktests, mit Bayes- und mit Netzwerktests. Wenn nur eine Punktzahl angegeben ist, wird sie für alle vier Fälle verwendet.

Beispiele:

```
score FROM_ENDS_IN_NUMS 0.177 0.516 0.517 0.000
score USER_IN_WHITELIST -100.00
```

Mit dieser Direktive können auch die Punktzahlen der eingebauten Tests beliebig geändert werden. Insbesondere kann man eine Punktzahl auf 0 setzen, um einen Test auszuschalten.

#### describe

Diese Direktive weist einem Test eine Beschreibung zu.

#### header

#### body

#### rawbody

#### uri

#### full

#### meta

Diese Direktiven definieren die eigentlichen Testregeln für verschiedene Testarten.

#### tflags

Mit dieser Direktive können Tests verschiedene Attribute (»Flags«) zugeordnet werden, die bestimmen, unter welchen Umständen der Test verwendet werden soll.

#### priority

Diese Direktive kontrolliert, in welcher Reihenfolge die Tests ausgeführt werden sollen. Für benutzerdefinierte Tests ist dies in der Regel nicht relevant.

## Diverse Einstellungen

Dieser Abschnitt erläutert schließlich noch einige weitere für die Praxis interessante Einstellungen.

#### trusted\_networks

Diese Direktive bestimmt, welche Hosts oder Netzwerke »vertrauenswürdig« sind. Es wird angenommen, dass Hosts in vertrauenswürdigen Netzwerken keinen Spam senden, keine offenen Relays betreiben und keine Header fälschen. Daher werden solche Hosts aus Effizienzgründen von DNSBL-Prüfungen ausgenommen.

Um mehrere Hosts oder Netzwerke als vertrauenswürdig einzustufen, können mehrere Adressen angegeben oder die Direktive wiederholt werden.

Beispiele:

```
trusted_networks 192.168/16 127/8 # 192.168.*.* und 127.*.*.*
trusted_networks 212.17.35.15 # nur dieser Host
trusted_networks 127. # 127.*.*.*
```

Alle in `internal_networks` aufgelisteten Netzwerke sollten auch als `trusted_networks` gelistet werden. Wenn `trusted_networks` nicht definiert ist, wird der Wert von `internal_networks` verwendet.

Ist DNS verfügbar, gibt es eine zusätzliche Heuristik, um die vertrauenswürdigen Netzwerke automatisch zu erkennen.

`clear_trusted_networks`

Diese Direktive löscht die mit `trusted_networks` angelegte Liste.

`internal_networks`

Diese Direktive bestimmt, welche Hosts oder Netzwerke »intern« sind. Das Format ist das gleiche wie für `trusted_networks`. Alle MX-Server für die eigenen Domains sowie interne Relays sollten hier eingetragen werden. Relays, die die E-Mail direkt von Dial up-Adressen annehmen, sollten nicht hier, aber möglicherweise in `trusted_networks` eingetragen werden. Diese Einstellung ist nötig für das korrekte Funktionieren von `whitelist_from_rcvd` und DNSBL mit Dial up-Einträgen.

Wenn `trusted_networks` gesetzt ist, jedoch `internal_networks` nicht, wird der Wert von `trusted_networks` für `internal_networks` übernommen. Wenn keiner der beiden Werte gesetzt ist, gilt kein Host mit Ausnahme des eigenen als intern.

`clear_internal_networks`

Diese Direktive löscht die mit `internal_networks` angelegte Liste.

`lock_method`

Diese Direktive bestimmt, welche Sperrmethode verwendet wird, um die Bayes- und Autowhitelisting-Datenbanken vor gleichzeitigem Zugriff zu schützen. Mögliche Sperrmethoden sind `nfssafe` (nur Unix), `flock` (nur Unix) und `win32` (nur Windows). Die Voreinstellung ist je nach Betriebssystem entweder `nfssafe` oder `win32`. Wenn sichergestellt ist, dass auf die Datenbanken nie über NFS zugegriffen wird, kann auf Unix-Plattformen erheblich an Performance gewonnen werden, indem die Sperrmethode `flock` verwendet wird.

`allow_user_rules`

Wenn diese Option auf 1 gesetzt ist, werden die Tests, die Anwender in ihrer `user_prefs`-Datei definiert haben, von `spamd` verwendet. Das ist in der Voreinstellung nicht der Fall, da dies eine Sicherheitslücke darstellen kann und auch die Performance beeinträchtigt. Eigene Tests sollten am besten vom Administrator in der globalen Konfiguration angelegt werden.

## Einbindung in das E-Mail-System

In diesem Abschnitt wird beschrieben, wie die SpamAssassin-Software in ein vorhandenes E-Mail-System eingebunden werden kann. Hier gibt es einige Möglichkeiten, die sich in Sachen Administrationsaufwand, Flexibilität und Geschwindigkeit zum Teil erheblich unterscheiden.

### Einbindung über Procmail

Procmail ist ein Programm, das vom Mailserver zur lokalen Auslieferung von E-Mails verwendet werden kann. Es liest eine E-Mail über die Standardeingabe und verfährt damit entsprechend einer Konfigurationsdatei. Diese Konfigurationsdatei enthält Regeln, die dazu dienen, die E-Mail anhand bestimmter Kriterien in unterschiedliche Postfächer zu sortieren. Vorher können jedoch auch externe Programme zur Analyse oder Filterung der E-Mail aufgerufen werden.

Bei E-Mail-Systemen, die bereits Procmail verwenden, kann SpamAssassin sehr einfach von Procmail aus aufgerufen werden. Bei Unix-basierten E-Mail-Server-Programmen ist Procmail entweder vorkonfiguriert oder kann mit wenigen Handgriffen aktiviert werden. (Siehe auch Kapitel 12, *Procmail*.) Andere Software, die analog zu Procmail arbeitet, kann selbstverständlich auch verwendet werden, wird hier aber nicht beschrieben.

Die Einbindung von SpamAssassin kann entweder über die globale Procmail-Konfigurationsdatei */etc/procmailrc* oder über die benutzerspezifischen Konfigurationsdateien *~/procmailrc* geschehen. Auf die letztere Art kann auch jeder Benutzer selbst bestimmen, ob er SpamAssassin einsetzen möchte. Dies erfordert natürlich, dass jeder Benutzer Shell-Zugang zum Mailserver hat, was jedoch häufig nicht der Fall ist.

Beispiel 4-6 zeigt den Codeabschnitt, der in die gewählte Procmail-Konfigurationsdatei eingefügt werden muss, um SpamAssassin aufzurufen. Das »f« in der ersten Zeile bestimmt, dass die folgende Regel eine Filterregel ist, also die E-Mail nicht zustellt, sondern nur manipuliert und wieder zurückgibt. Das »w« bestimmt, dass Procmail warten soll, bis das Filterprogramm beendet ist, und den Rückgabecode prüfen soll. Dies schützt vor Fehlfunktionen des *spamassassin*-Prozesses.

Der Abschnitt kann an beliebiger Stelle eingefügt werden, sinnvollerweise aber direkt am Anfang nach eventuellen Variablenzuweisungen. Im Beispiel wird das Programm *spamassassin* verwendet, aber *spamc* kann genauso genommen werden, wenn dafür gesorgt ist, dass der Serverprozess *spamd* läuft.

*Beispiel 4-6: Einbindung von SpamAssassin in procmailrc*

```
:0 fw  
| spamassassin
```



benutzerspezifische Bayes-Datenbanken verwendet werden können. Der Einsatz von Procmail ist allerdings nur möglich, wenn die E-Mail an das lokale System ausgeliefert wird. Bei E-Mail-Gateway-Systemen, die E-Mails nur prüfen und weiterleiten, kann Procmail nicht verwendet werden. Wenn Procmail eingesetzt wird, muss außerdem die E-Mail zunächst komplett angenommen werden und kann nicht eventuell schon während des SMTP-Dialogs abgelehnt werden.

Viele Mailserver arbeiten außerdem isoliert vom Empfänger der E-Mails, und jene Empfänger haben keine Möglichkeit, auf den Mailserver zuzugreifen und eigene Procmail-Regeln zu schreiben. Das Programm Fetchmail kann E-Mail vom Mailserver holen und in einen lokal laufenden MTA einspeisen, der dann Procmail ausführen könnte. Dann könnte man diese Procmail-Regeln auf dem lokalen Rechner einrichten. In allen anderen Fällen ist eine Einbindung von SpamAssassin in den MTA zu empfehlen, die im Folgenden beschrieben wird. Nichtsdestoweniger ist es in solchen Fällen möglich, Procmail oder ähnliche Software zum Sortieren der geprüften E-Mails zu verwenden oder gar SpamAssassin mit anderen – gegebenenfalls schärferen – Tests erneut aufzurufen.

## Einbindung in Postfix

Dieser Abschnitt beschreibt die Einbindung von SpamAssassin in ein E-Mail-System, das Postfix als MTA verwendet.

Content-Filter in Postfix sind beliebige Programme, die SMTP sprechen und E-Mails nach eigenem Gutdünken filtern, umschreiben oder löschen können. SpamAssassin selbst kann nicht als ein solcher Content-Filter fungieren, da es die notwendigen Protokolle nicht unterstützt. Stattdessen wird ein Programm verwendet, das diese Protokolle unterstützt und SpamAssassin intern aufruft.

Ein solches Programmpaket ist Amavisd-new. Dieses Paket enthält einen Serverdienst, der als Content-Filter in Postfix eingebunden wird und wahlweise E-Mails auf Viren, Spam oder verbotene Anhänge prüfen kann. Dem Thema Amavisd-new ist das komplette Kapitel 8, *AMaViS* gewidmet. Die Vorteile gegenüber dem Procmail-Ansatz sind, dass Amavisd-new sehr viel mehr als nur Spam-Erkennung unterstützt und dass es auch auf Gateway-Systemen ohne lokale Benutzer laufen kann. Gegenüber Procmail ist Amavisd-new aber viel komplexer und erfordert einigen Administrationsaufwand. Amavisd-new verwendet im Übrigen das oben erwähnte Perl-Modul `Mail::SpamAssassin`, um SpamAssassin zu integrieren. Da Amavisd-new selbst ein Serverdienst ist, ist es nicht nötig, einen separaten Serverdienst für SpamAssassin zu betreiben.

Alternativ zu Amavisd-new bietet sich auch MailScanner an, das in Kapitel 9, *MailScanner* beschrieben wird.

## Einbindung in Exim

Dieser Abschnitt beschreibt die Einbindung von SpamAssassin in ein E-Mail-System, das Exim als MTA verwendet.

### Einbindung über Router und Transports

Die einfachste Möglichkeit, SpamAssassin in Exim einzubinden, ist, einen Transport zu schreiben, der die E-Mails über eine Pipe an SpamAssassin übergibt und danach wieder in Exim einspeist. Der benötigte Transport sieht so aus:

```
spamassassin:
  driver = pipe
  use_bsmtp = true
  command = /usr/sbin/exim -bS -oMr sa-checked
  transport_filter = /usr/bin/spamc -f
  home_directory = "/tmp"
  current_directory = "/tmp"
  return_fail_output = true
  return_path_add = false
  user = exim
  group = exim
```

Die Reihenfolge der Transport-Definitionen ist egal. Je nach Betriebssystem muss der Pfad zum Programm *exim* sowie der Name des Exim-Benutzers und der Exim-Gruppe angepasst werden.

Nun muss ein Router definiert werden, der E-Mails an den eben definierten Transport leitet. Diese Router-Definition sieht so aus:

```
spamassassin_router:
  driver = accept
  transport = spamassassin
  condition = "${if {!eq} {$received_protocol}{sa-checked} {1} {0}}"
  no_expn
  no_verify
```

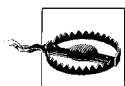
Dieser Router wird nur verwendet, wenn die Variable *\$received\_protocol* nicht auf *sa-checked* gesetzt ist. Dies wird oben im Transport mit der Option *-oMr sa-checked* getan. Zusammen verhindert dieses System, dass eine schon von SpamAssassin geprüfte E-Mail ein zweites Mal an denselben Transport übergeben wird.

Die Reihenfolge der Router ist wichtig, und durch Variieren der Reihenfolge lassen sich verschiedene spezielle Konfigurationen realisieren. Wenn der Router der erste der Liste ist, werden alle E-Mails, die vom Exim-Mailserver verarbeitet werden, von SpamAssassin überprüft. Dies ist insbesondere auf E-Mail-Gateways ohne lokale Benutzer sinnvoll. Wenn nur die E-Mail von lokalen Benutzern geprüft werden soll, sollte der Router hinter dem Router *dnslookup* stehen. Indem man den Router *spamassassin\_router* weiter nach unten schiebt, kann man verschiedene Klassen von E-Mails von der Spam-Erkennung ausnehmen. Auf jeden Fall muss der Router vor dem Router *local\_user* stehen, damit er überhaupt in Betracht gezogen wird.

Dieses System markiert die E-Mails lediglich, sorgt aber selbst nicht dafür, dass Spam aussortiert wird. Dies könnte wiederum mit Procmail oder im E-Mail-Client des Empfängers geschehen.

### Einbindung über Content-Scanning

Das Content-Scanning-Feature von Exim erlaubt es, SpamAssassin aus einer ACL (Access Control List) heraus aufzurufen. Dadurch kann Spam schon während des SMTP-Dialogs erkannt und bei Bedarf direkt abgelehnt werden, ohne dass die E-Mail erst angenommen werden muss. Außerdem können Virens Scanner aufgerufen und unerwünschte MIME-Typen blockiert werden. Content-Scanning gibt es seit der Version 4.50 in Exim; für frühere Versionen existieren Patches unter dem Namen Exiscan.



Wenn das Content-Scanning-Feature mit SpamAssassin verwendet wird, muss die SpamAssassin-Option `report_safe` auf 0 gesetzt werden, sonst kommt Exim durcheinander. Außerdem darf `spamd` nicht mit der Option `--create-prefs/-c` gestartet werden, da es in diesem Fall kein sinnvolles Home-Verzeichnis für die zu erzeugenden Konfigurationsdateien gibt.

Mit dem Content-Scanning-Feature hat die ACL-Sprache von Exim eine zusätzliche Bedingung namens `spam`, die SpamAssassin per `spamc/spamd` abfragt. Hat SpamAssassin die E-Mail als Spam eingestuft, ist die Bedingung erfolgreich, und die ACL kann Folgemaßnahmen ergreifen. Dieses System verändert nichts an der E-Mail; wenn die bekannten Markierungen wie zusätzliche Header gewünscht sind, muss das in der ACL angegeben werden. Das folgende Beispiel zeigt den Aufruf der ACL-Regel `spam` und zeigt, wie der typische Header `X-Spam-Flag: YES` in als Spam erkannte E-Mails eingefügt wird:

```
acl_smtp_data:
  warn message = X-Spam-Flag: YES
  spam = nobody
```

Die Angabe `nobody` bestimmt den Benutzer, unter dem der Aufruf von `spamc` erfolgt.

Die ACL-Bedingung `spam` setzt zusätzlich noch einige Variablen, die verwendet werden können, um unterschiedlichen Umgang mit den SpamAssassin-Ergebnissen zu implementieren.

#### `$spam_bar`

Bei einer positiven Spam-Punktzahl besteht diese Variable aus so vielen Pluszeichen wie Punkten, bei einer negativen Punktzahl aus entsprechend vielen Minuszeichen, bei null Punkten aus einem Schrägstrich. Diese Variable erfüllt also in etwa den gleichen Zweck wie der Platzhalter `_STARS(*)_` in SpamAssassin.

#### `$spam_report`

Diese Variable enthält den vollständigen SpamAssassin-Bericht zu einer E-Mail, so wie der, der sonst typischerweise im Header X-Spam-Report steht.

#### `$spam_score`

Diese Variable enthält die Spam-Punktzahl der E-Mail.

#### `$spam_score_int`

Diese Variable enthält die Spam-Punktzahl multipliziert mit 10 als Integer-Wert. Damit können ACL-Bedingungen definiert werden, die die Punktzahl auswerten; siehe nachfolgende Beispiele.

Das folgende Beispiel fügt in jede E-Mail einen Header X-Spam-Report ein, so wie SpamAssassin das normalerweise selbst machen würde. Indem `true` an den Benutzernamen (`nobody`) angehängt wird, wird die Regel jedes Mal angewendet, unabhängig vom SpamAssassin-Ergebnis.

```
acl_smtp_data:
warn message = X-Spam-Report: $spam_report
spam = nobody:true
```

Das folgende Beispiel lehnt E-Mails mit einer Spam-Punktzahl über 15 schon während des SMTP-Dialogs ab:

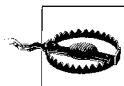
```
acl_smtp_data:
deny message = Spam with score $spam_score rejected
spam = nobody
condition = ${if >{$spam_score_int}{150}{1}{0}}
```

Diese ACL-Anweisungen könnten auch in eine einzige ACL zusammengefasst werden, um beide Verhaltensweisen gleichzeitig zu aktivieren.

Weitere Informationen zur Verwendung von Exim finden sich in Kapitel 3, *Spam- und Virenabwehr mit Postfix, Exim und Sendmail*.

### Einbindung über SA-Exim

SA-Exim ist ein Plugin für Exim, das SpamAssassin über `spamc/spamd` aufruft und viele einfach zu konfigurierende Möglichkeiten für den Umgang mit erkanntem Spam bietet. Es ist sogar üblich, das eingebaute Content-Scanning-Feature zum Virenschannen einzusetzen und gleichzeitig SpamAssassin über SA-Exim aufzurufen. SA-Exim hat eine umfangreiche Konfigurationsdatei, in der Regel in `/etc/exim/sa-exim.conf`, die schon sinnvolle Voreinstellungen enthält und viele andere Konfigurationsmöglichkeiten zum einfachen Auskommentieren bietet.



Wenn SA-Exim verwendet wird, darf `spamd` nicht mit der Option `--create-prefs/-c` gestartet werden, da es in diesem Fall kein sinnvolles Home-Verzeichnis für die zu erzeugenden Konfigurationsdateien gibt.

Zuerst muss

```
SAEximRunCond: 0
```



auskommentiert werden, damit SA-Exim überhaupt aktiviert ist.

Die Option `SAEximRejCond` bestimmt, ob Exim mit als Spam eingestuften E-Mails bestimmte Aktionen durchführen soll oder sie einfach mit den üblichen Markierungen im Header durchlassen soll, falls dies in SpamAssassin konfiguriert ist. Die Standardkonfigurationsdatei enthält diese Einstellung:

```
SAEximRejCond: ${if !eq {$h_X-SA-Do-Not-Rej:}{Yes} {1}{0}}
```

Dies bedeutet, dass mit allen Spam-E-Mails die konfigurierten Aktionen durchgeführt werden, außer mit denen mit dem Header `X-SA-Do-Not-Rej: Yes`. Dieser Header könnte folgendermaßen in der ACL `acl_smtp_rcpt` eingefügt werden, wenn es sich zum Beispiel um E-Mails an den Postmaster handelt.

```
warn message = X-SA-Do-Not-Rej: Yes  
local_parts = postmaster
```

Mit der Option `SAteergrube` kann man eine in der Tat so genannte Teergrube einrichten: Wenn eine E-Mail als Spam eingestuft wird, dann wird der SMTP-Dialog extrem verlangsamt, um Spammer in ihren Aktivitäten zu bremsen oder zumindest den Mailserver unattraktiv für Spammer zu machen. Die Option nimmt als Wert die Spam-Punktzahl, oberhalb der das Teergrubing aktiv werden soll. Damit man nicht den eigenen Host ausbremst, muss man einen etwas komplexeren Ausdruck anwenden, beispielsweise den, der in der Standardkonfigurationsdatei vorgeschlagen wird:

```
SAteergrube: ${if and { {!eq {$sender_host_address}{127.0.0.1}} {!eq {$sender_host_address}{127.0.0.2}} } {25}{1048576}}
```

Der Wert 1048576 dient dazu, das Teergrubing bei lokalen Verbindungen durch eine sehr hohe Punktzahlgrenze effektiv auszuschalten.

Die Option `SAdevnull` bestimmt, ab welcher Spam-Punktzahl eine E-Mail von SA-Exim verworfen werden soll. Wenn die Option nicht gesetzt ist, werden keine E-Mails verworfen.

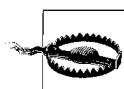
Beispiel:

```
SAdevnull: 20.0
```

Die Option `SApermreject` bestimmt, ab welcher Spam-Punktzahl SA-Exim eine E-Mail während des SMTP-Dialogs ablehnen soll. Wenn die Option nicht gesetzt ist, werden keine E-Mails abgelehnt. Die folgende Einstellung ist in der Standardkonfiguration aktiviert:

```
SApermreject: 12.0
```

Weitere SA-Exim-Einstellungen, die aber selten verändert werden müssen, findet man in der sehr gut kommentierten Konfigurationsdatei von SA-Exim.



Wenn die SpamAssassin-Option `report_safe` auf 1 oder 2 gesetzt ist, muss die SA-Exim-Option `SARewriteBody` auf 1 gesetzt werden, sonst kommt SA-Exim durcheinander. Diese Einstellung kostet allerdings Performance.

## Einbindung über AMaViS

Schließlich gilt auch für Exim, dass SpamAssassin über den Umweg Amavisd-new eingebunden werden kann. Bei der Fülle der Konfigurationsmöglichkeiten, die Exim bietet, erscheint es aber nicht nötig, noch ein so komplexes System wie Amavisd-new ins Spiel zu bringen, aber Amavisd-new bietet in jedem Fall einen ähnlichen Funktionsumfang wie die anderen hier vorgestellten Lösungen. Kapitel 8, *AMaViS* enthält ausführliche Informationen über Amavisd-new.

## Einbindung in Sendmail

Dieser Abschnitt beschreibt die Einbindung von SpamAssassin in ein E-Mail-System, das Sendmail als MTA verwendet.

Milter sind Programme, die von Sendmail über eine spezielle API aufgerufen werden und die eingehende E-Mails an verschiedenen Stellen im SMTP-Dialog annehmen, abweisen oder abändern können. SpamAssassin selbst kann nicht als ein solcher Milter fungieren, aber es gibt Programmpakete, die als Milter eingesetzt werden und die SpamAssassin intern aufrufen.

Am einfachsten geht es mit dem SpamAssassin-Milter-Plugin, kurz *spamass-milter* genannt. Der Milter läuft als Serverprozess, muss also vor Inbetriebnahme gestartet werden. Wenn ein fertiges Paket verwendet wird, geschieht dies automatisch beim Booten oder über ein Startskript (zum Beispiel */etc/init.d/spamass-milter* auf Debian GNU/Linux). Von Hand kann man ihn folgendermaßen starten:

```
/usr/sbin/spamass-milter -p /var/run/sendmail/spamass.sock -f
```

Wenn nichts anderes eingestellt ist, lässt der Milter die E-Mails lediglich durch SpamAssassin markieren und leitet sie an Sendmail zurück unter der Annahme, dass sie später von Procmail, dem E-Mail-Client oder an einer anderen Stelle sortiert werden.

Das Verhalten des Milters wird über Kommandozeilen-Optionen gesteuert. Die wichtigsten sind:

**-p** *dateiname*

Diese Option bestimmt den Pfad zur Socket-Datei, über die Sendmail mit dem SpamAssassin-Milter kommuniziert. Der Pfad kann frei gewählt werden, aber er muss mit dem in der Sendmail-Konfigurationsdatei angegebenen übereinstimmen (siehe unten). Diese Option muss angegeben werden.

**-f**

Diese Option sorgt dafür, dass der Daemon im Hintergrund gestartet wird, was der Normalfall sein sollte.

**-i** *netzwerke*

Diese Option bestimmt, dass E-Mails aus den angegebenen Netzwerken ignoriert, also nicht geprüft werden. Damit können zum Beispiel E-Mails aus dem eigenen Netzwerk von der Spam-Prüfung ausgeschlossen werden. Das Argu-

ment kann eine einzelne IP-Adresse sein (zum Beispiel 127.0.0.1), in CIDR-Schreibweise (zum Beispiel 192.168.1.0/24) erfolgen oder als Netzwerk/Netzmaske-Paar (zum Beispiel 192.168.1.0/255.255.255.0) geschrieben werden. Mehrere Adressen können durch Kommata getrennt werden, oder die Option wird mehrmals angegeben.

**-P *dateiname***

Diese Option bestimmt den Namen der PID-Datei, wenn eine gewünscht ist.

**-r *nn***

Mit der Option wird eine E-Mail abgelehnt, wenn die Spam-Punktzahl größer als *nn* ist. Normalerweise werden alle E-Mails durchgelassen.

**-u *user***

Die Option bestimmt, dass SpamAssassin auf benutzerspezifische Einstellungen zugreift. Wenn die E-Mail nur einen Empfänger hat, wird dessen Einstellungsdatei verwendet. Bei mehreren Empfängern wird der hiermit angegebene Benutzer verwendet, üblicherweise »nobody«.

Nachdem der Milter konfiguriert ist, wird die folgende Zeile in die Sendmail-Konfigurationsdatei eingefügt:

```
INPUT_MAIL_FILTER(`spamassassin', `S=local:/var/run/sendmail/spamass.sock, F=T,  
T=C:15m;S:4m;R:4m;E:10m')dnl  
define(`confMILTER_MACROS_CONNECT', `b, j, _, {daemon_name}, {if_name},  
{if_addr}')dnl
```

Die Timeouts (T=) wurden hier gegenüber den Vorgaben etwas erhöht, da SpamAssassin ein relativ langsamer Prozess sein kann. Nach dieser Änderung muss die Sendmail-Konfiguration wie üblich neu gebaut werden.

Weitere Informationen zur Sendmail-Konfiguration und zum Thema Milter finden sich in Kapitel 3, *Spam- und Virenschutz mit Postfix, Exim und Sendmail*.

Ein alternatives Programmpaket zum Aufruf von SpamAssassin aus Sendmail ist MIMEDefang. MIMEDefang unterstützt nicht nur Spam-Erkennung, sondern kann auch Virens Scanner aufrufen und unerwünschte Anhänge erkennen. Gegenüber Procmail und dem SpamAssassin-Milter ist MIMEDefang aber viel komplexer und erfordert einigen Administrationsaufwand. MIMEDefang wird in Kapitel 10, *MIMEDefang* behandelt.

Alternativ zu MIMEDefang bieten sich auch Amavisd-new und MailScanner an (siehe Kapitel 8, *AMaViS* und Kapitel 9, *MailScanner*), die allerdings nicht speziell für Sendmail ausgelegt sind.

## Definition eigener Tests

Dieser Abschnitt beschreibt, wie eigene SpamAssassin-Tests geschrieben werden können.

## Grundlagen

Jeder Test hat zunächst einen Namen. Gebräuchlich sind Testnamen bestehend aus Großbuchstaben, Zahlen und Unterstrichen mit maximal 22 Zeichen, zum Beispiel SUBJ\_2\_NUM\_PARENS. Die Namen eigener Tests sollten mit den eigenen Initialen oder einem ähnlichen Unterscheidungsmerkmal beginnen, damit es keine Namenskonflikte mit den offiziellen oder anderen Testsammlungen gibt. Testnamen, die mit zwei Unterstrichen beginnen, haben eine besondere Bedeutung, siehe unten im Abschnitt »Metatests«.

Nun muss der eigentliche Test definiert werden. Tests können nach Mustern im Header oder Body der E-Mail suchen oder spezielle Funktionen ausführen. Dies wird im Anschluss im Detail beschrieben.

Optional kann jedem Test eine Beschreibung zugeordnet werden. Diese Beschreibung ist eine einfache Zeichenkette, die je nach Konfiguration in die E-Mail eingefügt wird, um die erfolgreichen Tests dem Benutzer zu erklären. Siehe dazu die Beispiele 4-3 und 4-4.

Ebenfalls optional können jedem Test bestimmte Flags zugeordnet werden, die bestimmen, unter welchen Umständen und Konfigurationen der Test verwendet wird. Dies ist insbesondere dann nützlich, wenn größere Testsammlungen auch für die externe Verwendung freigegeben werden. In Tests, die für den eigenen Bedarf geschrieben sind, ist dies in der Regel nicht notwendig, da die lokale Konfiguration bekannt ist.

Schließlich muss jedem Test eine Punktzahl zugeordnet werden.

Das folgende Beispiel aus der vordefinierten Testsammlung in SpamAssassin ist eine vollständige Testdefinition.

```
header FROM_ENDS_IN_NUMS      From =~ /\d\d\@/
describe FROM_ENDS_IN_NUMS    From: ends in numbers
lang de describe FROM_ENDS_IN_NUMS Absenderadresse endet mit Ziffern im
Benutzernamen
score FROM_ENDS_IN_NUMS      0.177 0.516 0.517 0.000
```

Diese Zeilen definieren einen Test mit dem Namen FROM\_ENDS\_IN\_NUMS. Die Direktive header bestimmt, dass dieser Test die Header der E-Mail prüft, und gibt die eigentliche Testregel an: Die From-Zeile wird mit einem regulären Ausdruck verglichen. Der reguläre Ausdruck passt, wenn der Teil der E-Mail-Adresse vor dem @ mit zwei (oder mehr) Zahlen endet, was darauf hindeuten kann, dass der Absender automatisch erzeugte »Wegwerf«-E-Mail-Adressen verwendet. Der Operator =~ steht für einen Vergleich mit einem regulären Ausdruck. Genauer gesagt, der Test ist dann erfolgreich, wenn der Header den regulären Ausdruck *enthält*. Alternativ kann der Operator !~ verwendet werden, der prüft, ob ein Header den regulären Ausdruck nicht enthält.

Die regulären Ausdrücke entsprechen den in Perl verwendeten. Insbesondere können auch Modifikatoren wie /i (Groß-/Kleinschreibung nicht unterscheiden) ange-

hängt werden. Im Gegensatz zu Perl muss ein # »escapt« werden (\#), weil es sonst als Start eines Kommentars interpretiert würde.

Die Direktive `describe` definiert die Beschreibung des Tests. Die dritte Zeile definiert die deutsche Version der Beschreibung, die an Stelle der englischen verwendet wird, wenn der Anwender im Betriebssystem die entsprechende Sprache (Locale) eingestellt hat.

Die Direktive `score` bestimmt schließlich die Punktzahl für diesen Test. Einem Test können bis zu vier Punktzahlen zugewiesen werden, die unter verschiedenen Umständen verwendet werden, nämlich in dieser Reihenfolge:

- ohne Bayes- und Netzwerktests
- ohne Bayes- und mit Netzwerktests
- mit Bayes- und ohne Netzwerktests
- mit Bayes- und mit Netzwerktests

Die vordefinierten Tests werden automatisch gewichtet und haben daher sehr präzise und teilweise merkwürdige Punktzahlen. Bei selbst definierten Tests ist es ausreichend, eine Punktzahl anzugeben, die dann in allen Fällen verwendet wird.

## Beispiele

Die mit SpamAssassin mitgelieferten Tests sind in Dateien im Verzeichnis `/usr/share/spamassassin/` definiert. Es ist eine umfangreiche Bibliothek von Beispielen und soll daher auch hier zur Erklärung der Features dienen. Aus verschiedenen Gründen sind die Tests über verschiedene Dateien verteilt, und die Punktzahl-Definitionen sind in eine getrennte Datei ausgelagert. Dies ist aber letzten Endes unerheblich.

Der folgende Test prüft einen regulären Ausdruck in *allen* Headern der E-Mail.

```
header CHINA_HEADER          ALL =~ /\@china\.com/i
describe CHINA_HEADER        Involves 'china.com'
score CHINA_HEADER           1.840 1.911 2.312 2.386
```

Der Header `ALL` ist einer von mehreren Pseudo-Headern, die das Schreiben von Tests erleichtern. Außerdem sind `ToCc` (alle Empfängeradressen), `EnvelopeFrom` (alle Envelope-Absenderadressen) und `MESSAGEID` (alle Header mit Message-IDs) definiert.

Der folgende Test prüft den `From`-Header, aber weil `addr` angehängt wurde, wird nur die erste E-Mail-Adresse geprüft. Die folgenden E-Mail-Adressen sowie alle Namenszusätze werden vorher entfernt.

```
header FROM_NONSENDING_DOMAIN From:addr =~ /\@(?:altavista\.com|eudora\.com)$/i
describe FROM_NONSENDING_DOMAIN Message is from domain that never sends email
score FROM_NONSENDING_DOMAIN 1.486 0.308 1.678 0.000
```

Statt `addr` kann auch `name` angehängt werden, um nur den ersten Namen zu testen. Diese Features erleichtern das »Auseinanderparsen« von E-Mail-Adressangaben.

Der folgende Test prüft, ob der Header X-Library vorhanden ist.

```
header X_LIBRARY          exists:X-Library
describe X_LIBRARY        Message has X-Library header
score X_LIBRARY           2.105 1.369 1.863 2.755
```

Der folgende Test ruft eine eingebaute Funktion `check_for_long_header()` auf, die prüft, ob die Header-Zeilen der E-Mail außergewöhnlich lang sind. Es gibt eine Reihe solcher eingebauter Funktionen, die Prüfungen vornehmen, die ansonsten nicht gut zu implementieren wären.

```
header HEAD_LONG          eval:check_for_long_header()
describe HEAD_LONG        Message headers are very long
score HEAD_LONG           2.5
```

Der folgende Test ruft ebenfalls eine eingebaute Funktion auf, die in diesem Fall eine DNSBL-Abfrage durchführt. Zusätzlich wird der Test mit »net« als Netzwerktest markiert und somit nicht ausgeführt, wenn Netzwerktests ausgeschaltet sind (`spamassassin --local` oder `spamd --local`). Die Punktzahlen, die angewendet werden, wenn Netzwerktests ausgeschaltet sind, sind hier der Einfachheit halber auf 0 gesetzt worden. Kapitel 5, *DNS-basierte Blackhole-Lists* beschreibt die DNSBL-Tests in SpamAssassin im Detail.

```
header RCVD_IN_NJABL_RELAY eval:check_rbl_sub('njabl', '127.0.0.2')
describe RCVD_IN_NJABL_RELAY NJABL: sender is confirmed open relay
tflags RCVD_IN_NJABL_RELAY net
score RCVD_IN_NJABL_RELAY 0 0.934 0 1.397
```

Der folgende Test ist ein Body-Test, das heißt, er prüft den Inhalt der E-Mail. Bevor ein Body-Test angewendet wird, wird die E-Mail in eine »lesbare« Form umgewandelt, was bedeutet, dass Nicht-Text-Teile wie Anhänge entfernt, Quoted-Printable- und Base64-Kodierungen aufgelöst und Zeilenumbrüche sowie HTML-Tags entfernt werden.

```
body BILLION_DOLLARS      /[BM]ILLION DOLLAR/
describe BILLION_DOLLARS  Talks about lots of money
score BILLION_DOLLARS    0.193 1.185 0.407 0.134
```

Auch für Body-Tests gibt es `eval`-Tests. Der folgende Test überprüft mit Hilfe einer eingebauten Funktion, ob die E-Mail per HTML einen besonders großen Font verwendet.

```
body HTML_FONT_BIG        eval:html_test('big_font')
describe HTML_FONT_BIG    HTML tag for a big font size
score HTML_FONT_BIG       0 0.232 0 0.142
```

Um einen Body-Test stattdessen auf der nicht dekodierten E-Mail auszuführen, verwendet man die Direktive `rawbody`. Folgendes Beispiel zeigt dies:

```
rawbody HIDE_WIN_STATUS   /<[^>]+onMouseOver=[^>]+window\.status=/i
describe HIDE_WIN_STATUS  Javascript to hide URLs in browser
score HIDE_WIN_STATUS     0.032 0 0 0.063
```

Relativ selten verwendet wird die Direktive `full`, die den Test über die vollständige, nicht dekodierte E-Mail mit Headern und Inhalt ausführt. So wird zum Beispiel der Pyzor-Test eingebunden:

```
full PYZOR_CHECK          eval:check_pyzor()
describe PYZOR_CHECK      Listed in Pyzor (http://pyzor.sf.net/)
tflags PYZOR_CHECK        net
score PYZOR_CHECK         0 2.041 0 3.451
```

Der folgende Test ist ein URI-Test. URI-Tests suchen alle URIs im Body der E-Mail und testen den regulären Ausdruck dagegen. Derartige Tests könnten theoretisch auch als Body-Tests geschrieben werden, aber die Verwendung der Direktive `uri` ist einfacher und weniger fehleranfällig, weil man nicht selbst den Anfang und das Ende der URI erkennen muss. Außerdem sind Tests so schneller.

```
uri REMOVE_PAGE           /^https?:\:\/\/[^\/]+\//.*?remove/
describe REMOVE_PAGE      URL of page called "remove"
score REMOVE_PAGE         0.081 0.604 0 0.191
```

## Metatests

Gelegentlich ist es sinnvoll, mehrere Tests zu einem zusammenzufassen. Dies ist insbesondere dann nötig, wenn Spam-Indizien an mehreren Stellen in einer E-Mail abgeglichen werden müssen. Mit einfachen regulären Ausdrücken ist das sehr schwierig zu implementieren.

Das folgende Beispiel soll dies anhand eines der vordefinierten Tests illustrieren. Der Test erkennt E-Mails, die vorgeben, mit dem AOL-E-Mail-Client geschrieben worden zu sein, die aber nicht von einer *aol.com*-Adresse kommen. Der E-Mail-Client wird über die Header-Zeile `X-Mailer` identifiziert, die Absenderadresse steht in der Header-Zeile `From`. Über die Direktive `meta` wird ein Metatest definiert, der erfolgreich ist, wenn der dahinter stehende logische Ausdruck aus anderen Tests zutrifft. Im logischen Ausdruck können `&&` für »und«, `||` für »oder«, `!` für »nicht« und Klammern zum Gruppieren verwendet werden.

```
header __AOL_MUA          X-Mailer =~ /\bAOL\b/
header __AOL_FROM         From:addr =~ /\@aol\.com$/i
meta FORGED_MUA_AOL_FROM  (__AOL_MUA && !__AOL_FROM)
describe FORGED_MUA_AOL_FROM Forged mail pretending to be from AOL (by From)
score FORGED_MUA_AOL_FROM 1.076 1.278 0.863 1.516
```

Ein weiteres Beispiel, das die Möglichkeit von Metatests ausnutzt, ist die Definition des Tests, der feststellt, ob der Subject-Header fehlt:

```
header __HAS_SUBJECT      exists:Subject
meta MISSING_SUBJECT      !__HAS_SUBJECT
describe MISSING_SUBJECT   Missing Subject: header
score                      MISSING_SUBJECT 1.109 1.570 1.282 1.226
```

Wenn Metatests bei einer E-Mail erfolgreich sind, zählen sowohl die Metatests als auch die Tests, die Bestandteile der Metaregeln sind, für die Gesamtpunktzahl der E-Mail. In der Regel möchte man aber die Untertests nicht noch mal getrennt mitzählen. SpamAssassin ermöglicht dies, indem man den Untertests Namen gibt, die mit zwei Unterstrichen anfangen, wie in den obigen Beispielen zu sehen. Derartige Tests zählen grundsätzlich nicht separat in der Endabrechnung.

## Eigene Tests überprüfen

Bevor selbst geschriebene Tests aktiviert werden, sollten sie auf gültige Syntax und auf korrektes Funktionieren getestet werden. Das Programm `spamassassin` bietet dazu einige Hilfen.

Zuerst bietet sich die Option `--lint` an. Der einfache Aufruf

```
spamassassin --lint
```

überprüft die Syntax aller Konfigurationsdateien und informiert über eventuelle Fehler.

Man sollte die eigenen Regeln testen, bevor man sie in die offiziellen Konfigurationsverzeichnisse kopiert. Dazu kann man mit der Option `--siteconfigpath` ein Verzeichnis als Ersatz für `/etc/mail/spamassassin/` bestimmen. Beispiel 4-9 zeigt beide Optionen im Einsatz. Der erste Fehler wurde in diesem Beispiel absichtlich eingebaut, die anderen drei Warnungen stammen interessanterweise aus den vordefinierten Regeln.

*Beispiel 4-9: Automatische Überprüfung eigener SpamAssassin-Regeln*

```
$ spamassassin --lint --siteconfigpath=test/  
config: SpamAssassin failed to parse line, skipping: boddy FOO_TEST /foo/  
warning: score set for non-existent rule RCVD_IN_NJABL  
warning: score set for non-existent rule RCVD_IN_SORBS  
warning: score set for non-existent rule HTML_FONT_COLOR_RED  
lint: 4 issues detected. please rerun with debug enabled for more information.
```

## Training und Anwender-Feedback

Da das Bayessche Klassifizierungssystem E-Mails anhand von Statistiken einstuft, müssen diese Statistiken angelegt und aktuell gehalten werden. Dies wird als das Training des Bayes-Systems bezeichnet. Das Training besteht daraus, dem Bayes-System mitzuteilen, welche E-Mails der eigenen Einschätzung nach Spam und welche Nicht-Spam sind. Es ist wichtig, dem Bayes-System beide Arten von E-Mails zu übergeben.

### Automatisches Training

SpamAssassin kann das Bayes-System automatisch trainieren, indem es E-Mails, die auf Basis der statischen Tests klassifiziert worden sind, dem Bayes-System zum Lernen übergibt. Dazu muss die Option `bayes_auto_learn` auf 1 gesetzt werden, was in der Voreinstellung der Fall ist. Das automatische Training erkennt aber nur E-Mails, die SpamAssassin sowieso tendenziell als Spam einstufen würde, kann also nicht auf die spezifischen Umstände des tatsächlichen E-Mail-Aufkommens des Anwenders reagieren. Deswegen muss das Bayes-System zusätzlich noch von Hand trainiert werden.



Ein Nachteil des automatischen Lernens kann es sein, dass es Performance kostet, da jede eingehende E-Mail durch das Lernsystem laufen muss. Experimente haben gezeigt, dass die Vorteile des automatischen Lernens ab 10.000 gelernten E-Mails nachlassen. Wenn die Performance also ein Problem wird, könnte man das automatische Lernen nach einer Weile abschalten.

## Anlernen

Bevor das Bayes-System anfangen kann, neue E-Mails selbstständig zu bewerten, muss es eine große Zahl alter E-Mails gesehen und erlernt haben. In der Voreinstellung benötigt das Bayes-System mindestens 200 erlernte Spam-E-Mails und mindestens 200 erlernte Nicht-Spam-E-Mails. In der Praxis benötigt man aber erheblich mehr, bevor die Schlussfolgerungen des Bayes-Systems signifikant besser als Spam-Assassin ohne Bayes-System sind.

Wenn das automatische Lernen eingeschaltet ist, muss man im einfachsten Fall nur warten, bis eine ausreichend große Anzahl E-Mails aufgelaufen ist. Die Bayes-Tests werden danach automatisch anspringen.

Wer bereits eine ausreichende Anzahl Spam *und* Nicht-Spam gesammelt und selbst sortiert hat, der kann diese zu Beginn von Hand zum Lernen an das Bayes-System übergeben.

## Befehle

Um E-Mails an das Bayes-System zum Lernen zu übergeben, wird das Programm `sa-learn`, das in der SpamAssassin-Distribution enthalten ist, verwendet. `sa-learn` kann einzelne E-Mails, aber auch E-Mail-Ordner in verschiedenen Formaten am Stück erlernen. Um eine einzelne E-Mail als Nicht-Spam (»Ham«) zu erlernen, wird folgender Befehl verwendet:

```
sa-learn --ham dateiname
```

Um eine E-Mail als Spam zu erlernen, wird folgender Befehl verwendet:

```
sa-learn --spam dateiname
```

Diese Befehle könnte man zum Beispiel an Tastenkürzel im E-Mail-Client binden.

Bequemer ist es in der Regel, wenn man `sa-learn` direkt auf die E-Mail-Ordnerdateien ausführt. E-Mail-Ordner liegen auf Unix-Systemen in der Regel in einem von zwei Formaten vor: Mbox oder Maildir. Die Entscheidung zwischen diesen Formaten wird durch die Wahl des SMTP-Servers und des IMAP- oder POP3-Servers bestimmt, aber beide Formate werden von SpamAssassin gleichermaßen unterstützt.

E-Mail-Ordner im Mbox-Format bestehen aus einer Datei pro Ordner. Der Ordner *Posteingang* oder *Inbox* liegt in der Regel unter `/var/spool/mail/$USER` oder `/var/mail/$USER`, weitere Ordner in der Regel unter `~/mail/ordnername`. Die Lage der

Ordner ist aber für SpamAssassin unerheblich. Um das Bayes-System den Inhalt eines Mbox-Ordners lernen zu lassen, wird `sa-learn` mit der Option `--mbox` verwendet, also zum Beispiel:

```
sa-learn --mbox --ham /var/spool/mail/$USER
sa-learn --mbox --ham mail/Gelesen
sa-learn --mbox --spam mail/Spam
```

Hier wird automatisch der Posteingang und der Ordner mit den gelesenen Nachrichten als Nicht-Spam eingestuft und der Ordner *Spam* als Spam. Der Inhalt der Ordner *Gelesen* und *Spam* könnte hiernach gelöscht werden.

Diese Befehlsfolge, natürlich an die lokalen Gegebenheiten angepasst, sollte regelmäßig ausgeführt werden, zum Beispiel per `cron`. SpamAssassin merkt sich, welche E-Mails bereits erlernt worden sind, und ignoriert diese, wenn ein Ordner erneut zum Lernen übergeben wird. Es besteht also keine Gefahr, die Statistiken durch zu häufiges Lernen zu verfälschen.

Wenn später eine E-Mail im Posteingang vom Anwender als Spam eingestuft wird und in den Ordner *Spam* verschoben wird, kann diese Befehlsfolge ebenfalls einfach erneut ausgeführt werden. SpamAssassin wird erkennen, dass eine nun als Spam eingestufte E-Mail zuvor als Nicht-Spam erlernt worden ist, und wird die vorherige Einstufung vergessen.

E-Mail-Ordner im Maildir-Format bestehen aus einem Verzeichnis mit einer Datei pro E-Mail. Die Verzeichnisse liegen meist unter `~/Maildir/`. Jedes Verzeichnis, das einen logischen Ordner enthält, enthält drei Unterverzeichnisse: *cur*, *new* und *tmp*. *cur* enthält alle aktuell im E-Mail-Ordner enthaltenen, schon gelesenen E-Mails, *new* enthält alle ungelesenen E-Mails, und *tmp* wird als temporäres Verzeichnis während der Zustellung verwendet. Zum Lernen sollte nur das Verzeichnis *cur* verwendet werden. Die interne Organisation von anderen Ordnern und Unterordnern hängt von der verwendeten Software ab, kann aber einfach durch Nachsehen herausgefunden werden. Die nachfolgenden Beispiele beziehen sich auf den Courier IMAP-Server, der Unterordner als Verzeichnisse, deren Namen mit einem Punkt beginnen, anlegt. `sa-learn` unterstützt das Erlernen kompletter Verzeichnisse automatisch. Die Befehle, die analog zum obigen Beispiel wären, sind:

```
sa-learn --ham Maildir/cur/
sa-learn --ham Maildir/.Gelesen/cur/
sa-learn --spam Maildir/.Spam/cur/
```

Im Fall von Maildir-Ordnern gibt es noch eine Optimierungsmöglichkeit: Die Bayes-Datenbank besteht aus den eigentlichen Daten und einem Journal. Wenn eine E-Mail erlernt wird, werden die neuen Daten zunächst in das Journal geschrieben, und erst am Ende von `sa-learn` wird das Journal in die eigentliche Datenbank zurückgeschrieben. Dies ist für Ordner im Mbox-Format das ideale Verhalten. Bei Ordnern im Maildir-Format ist es sinnvoll, die einzelnen E-Mails nur in das Journal

schreiben zu lassen und erst am Ende die Datenbank mit dem Journal abzugleichen. Dazu verwendet man beim Lernen die Option `--no-sync` und ruft am Ende `sa-learn` noch einmal mit der Option `--sync` auf, zum Beispiel:

```
sa-learn --ham --no-sync Maildir/cur/  
sa-learn --ham --no-sync Maildir/.Gelesen/cur/  
sa-learn --spam --no-sync Maildir/.Spam/cur/  
sa-learn --sync
```

## Lernstrategien

Nachdem das System angelernt ist, muss eine Strategie entworfen werden, wie die Bayes-Datenbank auf dem aktuellen Stand gehalten werden kann. Es reicht nicht aus, das System einmal zu trainieren und dann in Ruhe zu lassen.

Wenn das automatische Lernen eingeschaltet ist, wird der Großteil der E-Mails bereits von allein erlernt. Wichtig ist es in diesem Fall, die Fehlklassifizierungen, also falsche Positive und falsche Negative, zu berichtigen. Ein empfehlenswertes Vorgehen ist dabei, die falschen Positive und falschen Negative in zwei getrennte E-Mail-Ordner, zum Beispiel *falsepos* und *falseneg*, zu verschieben und sie per Cron-Job oder beim Verlassen des E-Mail-Clients regelmäßig automatisch erlernen zu lassen. Danach können sie, falls gewünscht, gelöscht werden, zum Beispiel gleich im selben Cron-Job. Diese Vorgehensweise erfordert nach einiger Zeit nur noch wenig Eingreifen, je nachdem, wie gut SpamAssassin arbeitet, und benötigt im Einzelfall nur wenige Tastendrucke oder Mausbewegungen, um einige E-Mail-Nachrichten in andere Ordner zu verschieben.

Wenn das automatische Lernen nicht eingeschaltet ist, müssen theoretisch alle E-Mails von Hand erlernt werden. Dazu sollte man alle E-Mails aufheben, zumindest bis sie dem Bayes-System zum Lernen übergeben worden sind. Nachrichten, die SpamAssassin als Spam erkannt hat, sowie solche, die man manuell als Spam identifiziert hat, sollte man in einen getrennten E-Mail-Ordner verschieben. Dort können sie per Cron-Job oder beim Verlassen des E-Mail-Clients regelmäßig als Spam erlernt und danach falls gewünscht gelöscht werden. Für Nachrichten, die kein Spam sind, gibt es verschiedene Verfahrensweisen. Einige Anwender heben alle E-Mails auf, dann ist das regelmäßige Erlernen dieser E-Mails kein Problem. Andere Anwender löschen alle E-Mails, nachdem sie sie gelesen haben. In dem Fall sollte man stattdessen gelesene E-Mails in einen anderen Ordner, zum Beispiel *Gelesen*, verschieben und sie erst löschen, wenn das Bayes-System sie erlernt hat.

Wie oben gesagt, ist es erfahrungsgemäß ab etwa 10.000 erlernter E-Mails fast genauso effektiv, nur die Fehler zu trainieren, anstatt alle E-Mails. Wenn diese Zahl erreicht ist und man die Performance verbessern möchte, kann man dann selbst bei abgeschaltetem automatischem Lernen dazu übergehen, wie oben nur die falschen Positive und die falschen Negative erlernen zu lassen.

## Einsatz auf Mailservern

Die oben vorgeschlagenen Lernstrategien können nur umgesetzt werden, wenn Spam-Assassin auf dem lokalen Rechner läuft oder die Endanwender Shell-Zugang zum Mailserver-System und eigene Home-Verzeichnisse darauf für die jeweils eigene Bayes-Datenbank haben. Ist dies nicht der Fall, wird es schwierig, eine sinnvolle Lernstrategie umzusetzen.

Wenn lediglich der Shell-Zugang unmöglich ist, könnten die Administratoren eventuell für jeden Benutzer die nötigen Postfächer und Cron-Jobs einrichten. Dies funktioniert jedoch nicht auf Gateway-Systemen, die die E-Mail lediglich filtern und zur Zustellung an andere E-Mail-Server weiterleiten.

Es wäre möglich, für alle Benutzer ein zentrales Postfach zum Mitteilen von Spam einzurichten. Dies kann kein öffentlich zugängliches Postfach sein, damit die Privatsphäre der einzelnen Nutzer gewahrt bleibt. Stattdessen könnte man eine E-Mail-Adresse einrichten, an die zum Lernen vorgeschlagene E-Mails gesendet werden. Leider werden aber von vielen E-Mail-Clients weitergeleitete E-Mails erheblich entstellt, sodass ein sinnvolles Lernen unter diesem System nicht möglich erscheint. Außerdem ist für das effektive Training sowohl Spam als auch Nicht-Spam nötig. Endanwender werden aber nur ungern gewillt sein, ihre legitimen Nicht-Spam-E-Mails kontinuierlich zum Training weiterzuleiten.

Ein effektives und präzises Training des Bayes-Systems ist also unter diesen erschwerten Umständen offensichtlich nicht möglich. In der Praxis wird sich in solchen Fällen auf das automatische Lernen verlassen. Wer dennoch ein Feedback-System für Anwender aufbauen will, sollte die Einrichtung eines eigenen Pyzor-Servers in Erwägung ziehen.