KAPITEL 8

AMaViS

AMaViS steht für »A Mail Virus Scanner«. Es ist allerdings kein Virenscanner, sondern eine Software, die dazu dient, Virenscanner in Mailserver einzubinden. Neuere AMaViS-Varianten können auch SpamAssassin und andere Anti-Spam-Software einbinden. AMaViS kann daher als komplettes E-Mail-Filter-Framework verstanden werden. Dieses Kapitel beschreibt die Funktionsweise von AMaViS, die Einbindung in den MTA und die Konfigurationsmöglichkeiten. Weitere E-Mail-Filter-Frameworks, die ähnliche Funktionalität wie AMaViS bieten, werden in den folgenden Kapiteln behandelt.

Wie AMaViS arbeitet

Computerviren gab es schon, bevor E-Mail eine nennenswerte Verbreitung erreicht hatte. Ebenso gab es schon Antivirenprogramme, bevor es E-Mail-Viren gab. Als wirksamen Schutz vor E-Mail-Viren waren diese Programme allerdings ungeeignet, denn E-Mail-Viren nutzen in der Regel Schwachstellen der E-Mail-Clients oder die Unerfahrenheit der Anwender aus und sollten daher abgefangen werden, bevor sie überhaupt das System des Empfängers erreichen. Am effektivsten sind Virenscanner also, wenn sie schon auf dem Mailserver ausgeführt werden.

Bei der Integration von Virenscannern in MTA gibt es einige Probleme. Virenscanner arbeiten primär auf Dateiebene: Sie scannen eine Datei und stellen fest, ob ein Virus enthalten ist. E-Mails liegen auf dem Mailserver aber nicht als Datei vor, sondern nur im Speicher und in internen Spool-Dateien. Außerdem können E-Mails durch verschiedene Mechanismen komplex kodiert werden: Anhänge, ZIP-Archive, MIME, Base64 und weitere. Zwar können moderne Virenscanner oft auch Archive und bestimmte kodierte Formen scannen, aber nicht unbedingt alle. Und es wäre insbesondere beim Einsatz von mehreren Virenscannern effizienter, wenn die Dekodierung nur einmal geschehen müsste. Außerdem gibt es dutzende Virenscanner-Produkte und auch mehrere MTA-Produkte, aber keine normierte Schnittstelle zwischen beiden Welten. Die primäre Aufgabe von AMaViS ist es, diese Brücke zu schlagen.

AMaViS erhält eine E-Mail vom MTA über eine standardisierte Schnittstelle, entweder über eine Pipe oder über SMTP. Dann zerlegt AMaViS die E-Mail in ihre MIME-Bestandteile, dekodiert den Inhalt und entpackt eventuelle Archive. Diese Einzelteile, die nun in Dateiform auf der Festplatte vorliegen, werden dann von den eingestellten Virenscannern überprüft. Falls ein Virus gefunden wird, kann AMaViS nun je nach Konfiguration die E-Mail verwerfen, einen Fehler an den MTA melden oder eine Benachrichtigung versenden. Ist kein Virus gefunden worden, wird die E-Mail weitergesendet. Dazu wird sie über SMTP an einen zweiten MTA weitergereicht, der die E-Mail dann normal an die Empfänger ausliefern kann.

Wenn die E-Mail schon einmal dekodiert und ausgepackt ist, dann bietet es sich auch an, gleich diverse andere Prüfungen mit der nun im Klartext vorliegenden E-Mail durchzuführen. Neuere AMaViS-Versionen erlauben es zum Beispiel, E-Mail-Anhänge mit bestimmten Dateitypen abzulehnen. Sie können auch die komplette Spam-Erkennung unter Zuhilfenahme von SpamAssassin abwickeln. Unter einigen Anhängern von AMaViS ist diese Erweiterung der Kompetenzen zwar umstritten, in der Praxis hat sich der Einsatz von AMaViS (oder auch der in den folgenden Kapiteln besprochenen Alternativen) als Bindeglied zwischen allen an der E-Mail-Filterung beteiligten Software-Komponenten jedoch als nützlich erwiesen.

AMaViS-Zweige

Die ursprüngliche AMaViS-Software wurde mehrfach in verschiedenen Varianten reimplementiert und unabhängig weiterentwickelt, wobei die Namen der Pakete nur leicht oder gar nicht verändert worden sind. Wer also nur nach »AMaViS« sucht, kann leicht die Übersicht verlieren. In diesem Abschnitt werden diese verschiedenen Entwicklungszweige vorgestellt. Die aktuellste und mächtigste AMaViS-Variante, die sich mittlerweile als Standard etabliert hat, wird dann im Rest dieses Kapitels im Detail besprochen.

amavis

Die ursprüngliche Version von AMaViS entstand um 1997 und wurde zuletzt im Jahr 2000 aktualisiert (Version 0.2.1). Das ursprüngliche AMaViS diente ausschließlich der Einbindung von Virenscannern in den Mailserver. Es war als Shell-Skript mit einigen in C geschriebenen Hilfsprogrammen implementiert und wurde ähnlich wie zum Beispiel Procmail als Delivery- oder Transport-Programm in die Mailserver-Konfiguration eingebunden. Dieser ursprüngliche AMaViS-Entwicklungszweig ist eingestellt und sollte nicht mehr verwendet werden.

amavis-perl

amavis-perl ist eine Reimplementierung des ursprünglichen AMaViS-Pakets in Perl und wurde erstmals im Jahr 2000 veröffentlicht. Es kann als legitimer Nachfolger des ursprünglichen AMaViS gelten und führt auch dessen Changelog und Versionsnummern fort. (Die Pakete heißen in der Tat auch amavis. Der Name amavis-perl wird nur zur Unterscheidung verwendet.) Die aktuell stabile Version ist 0.3.12 aus dem Jahr 2003. Die Einbindung in den Mailserver funktioniert hier ebenfalls als Delivery- oder Transport-Programm. Es gibt aber auch komplexere Möglichkeiten, zum Beispiel die Einbindung als Content-Filter in Postfix. Der Entwicklungszweig amavis-perl ist theoretisch noch aktiv, die Software hat aber im Vergleich zu späteren AMaViS-Varianten erhebliche Funktionsdefizite und wird hier nicht zum Einsatz empfohlen.

amavisd

amavisd ist eine Server-Variante von amavis-perl. Die Implementierung begann im Jahr 2001. Sie wird ebenfalls vom erweiterten ursprünglichen Entwicklerteam gepflegt. Um einen Teil des erheblichen Start-Overheads des `amavis`-Perl-Skripts zu sparen, wird der eigentliche Scan-Vorgang von einem ständig im Hintergrund laufenden Server, `amavisd`, übernommen. Das dazugehörige Client-Programm wird in den Mailserver eingebunden und leitet lediglich die zu prüfende E-Mail an den amavisd-Server weiter. Das Client-Programm hat den gleichen Namen und die gleiche externe Schnittstelle wie bei amavis-perl; die Integration in den Mailserver funktioniert daher identisch.

Das erste und bisher letzte offizielle Release von amavisd (0.1) gab es im Jahr 2003. Die Zukunft des Projekts ist wie bei amavis-perl unklar, und auch hier gilt, dass andere AMaViS-Varianten besseren Funktionsumfang bieten.

amavis-ng

amavis-ng ist die selbst ernannte »Next Generation« von amavis und entstand etwa 2002. Dieses Paket ist eine unabhängige Reimplementierung von amavis-perl und amavisd und ist ebenfalls in Perl geschrieben. Die Ziele dieses Projekts waren, die interne Architektur modularer zu gestalten und bei der Konfiguration und der Einbindung in den Mailserver mehr Flexibilität zu ermöglichen. So kann amavis-ng als Delivery-Programm, als Server oder als Content-Filter eingesetzt werden oder gar in Exim direkt im eingebetteten Perl-Interpreter laufen. Die Konfiguration erfolgt bei amavis-ng erstmals komplett über eine Konfigurationsdatei zur Laufzeit, im Gegensatz zu den bisherigen Implementierungen, bei denen die Konfiguration größtenteils schon bei der Installation festgelegt werden musste.

amavis-ng hat eine gewisse Verbreitung erreicht, wird von den Projektbetreuern aber als »tot« eingestuft. Es gab zwar bis zuletzt regelmäßige Minor Releases, die Zukunft des Projekts ist angesichts solcher Aussagen allerdings zweifelhaft.

amavisd-new

amavisd-new entstand 2002 als unabhängige Weiterentwicklung von amavisd (Aussage des Autors: »finally a version which I can recommend to friends«). Es läuft ausschließlich als Server und wird als Content-Filter in den Mailserver eingebunden. Neben zahlreichen Verbesserungen bei der Performance, Stabilität und Konfigurierbarkeit bricht amavisd-new als erstes Mitglied der AMaViS-Familie aus der strikten Ausrichtung auf das Virenscannen aus und bietet zusätzlich die Integration von anderen Filterprogrammen wie SpamAssassin. Durch die Integration von Virenscannern und SpamAssassin kann amavisd-new fast das komplette heute übliche Spektrum an E-Mail-Filterung unter einem gemeinsamen Dach bedienen. In der Tat können die MTA beim Einsatz von amavisd-new von E-Mail-Filterungsaufgaben weitestgehend entbunden werden.

amavisd-new ist zurzeit die leistungsfähigste und am besten gepflegte AMaViS-Variante und wird aus diesem Grund im Folgenden im Detail besprochen. Der Einfachheit halber steht »AMaViS« im Folgenden für die Variante amavisd-new. Das aktuelle Release zum Zeitpunkt des Schreibens hat die Version 2.2.1.

Einbindung in das E-Mail-System

In diesem Abschnitt wird beschrieben, wie AMaViS in die verschiedenen MTA eingebunden werden kann. Da die Konfigurationssprachen der MTA bei speziellen Anforderungen eine sehr große Flexibilität erlauben, kann hier nur jeweils eine einfache Konfiguration beschrieben werden, die für die meisten Fälle jedoch geeignet ist.

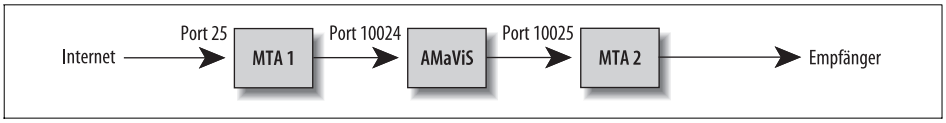

Abbildung 8-1: Fluss einer E-Mail durch ein MTA/AMaViS-System

Abbildung 8-1 zeigt den Fluss einer E-Mail durch ein MTA/AMaViS-System. Die E-Mail wird von einer ersten MTA-Instanz aus dem Internet über den normalen SMTP-Port 25 angenommen. Diese Instanz nimmt jetzt die Prüfungen vor, die ein MTA normalerweise durchführt, zum Beispiel ob die Empfängeradresse zu diesem Mailserver gehört. Dann wird die E-Mail ebenfalls über SMTP an den AMaViS-Server auf Port 10024 gesendet. AMaViS prüft nun die E-Mail nach den eingestellten Kriterien, ruft also meist Virenscanner und SpamAssassin auf. Wenn die E-Mail verworfen werden soll, ist die Verarbeitung hier beendet. Wenn die E-Mail aber ausge-

liefert werden soll, wird sie über SMTP an eine zweite MTA-Instanz auf Port 10025 gesendet. Diese MTA-Instanz wird die E-Mail nun wie ein normaler MTA ausliefern, entweder in lokale Postfächer oder an den nächsten zuständigen Mailserver. Die Portnummern 10024 und 10025 sind hier willkürlich gewählt, haben sich aber in diesen Systemen als Standard etabliert. Alle drei Subsysteme laufen in den meisten Fällen auf demselben Rechner.

Die Konfiguration eines MTA/AMaViS-Systems besteht also aus drei Schritten:

1. AMaViS konfigurieren
2. die Verbindung vom MTA zu AMaViS konfigurieren
3. die Verbindung von AMaViS zum MTA konfigurieren

AMaViS ist in der Standardkonfiguration bereits fertig voreingestellt. Um dies zu testen, startet man den AMaViS-Server und baut eine Verbindung auf Port 10024 auf:

```
$ telnet localhost 10024
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
220 [127.0.0.1] ESMTP amavisd-new service ready
QUIT
221 2.0.0 [127.0.0.1] (amavisd) closing transmission channel
Connection closed by foreign host.
```

Wenn es hierbei Probleme gibt, sollte man in der Konfigurationsdatei prüfen, ob folgende drei Einstellungen vorhanden sind:

```
$inet_socket_port = 10024;
$inet_socket_bind = '127.0.0.1';
@inet_acl = qw( 127.0.0.1 );
```

Diese Einstellungen bestimmen den Port, die IP-Adresse, an die gebunden wird, und die IP-Adressen, von denen Verbindungen angenommen werden sollen. Einzelheiten folgen später in diesem Kapitel.

Die Konfiguration der Verbindung vom MTA zu AMaViS und zurück hängt vom MTA ab und wird im Folgenden beschrieben.

Postfix

Obwohl AMaViS mit verschiedenen MTA zusammenarbeiten kann, ist die Kombination mit Postfix die vom AMaViS-Entwicklerteam am besten getestete. In der Postfix-Terminologie läuft AMaViS als »Content Filter« oder in Postfix-Versionen ab Version 2.1 genauer gesagt als »After-Queue Content Filter«. Diese Content-Filter sind beliebige Programme, die SMTP sprechen und E-Mails nach eigenem Gutdünken filtern, umschreiben oder löschen können.

Obwohl eine E-Mail – wie in Abbildung 8-1 gezeigt – zwei logische MTA-Instanzen durchläuft, werden diese Instanzen vom selben Postfix-Server betrieben. Der tat-

sächliche Ablauf sieht daher eher wie in Abbildung 8-2 aus. Insbesondere teilen sich beide Instanzen einen Queue-Bereich und können über dieselben Dateien und Kommandos konfiguriert und administriert werden.

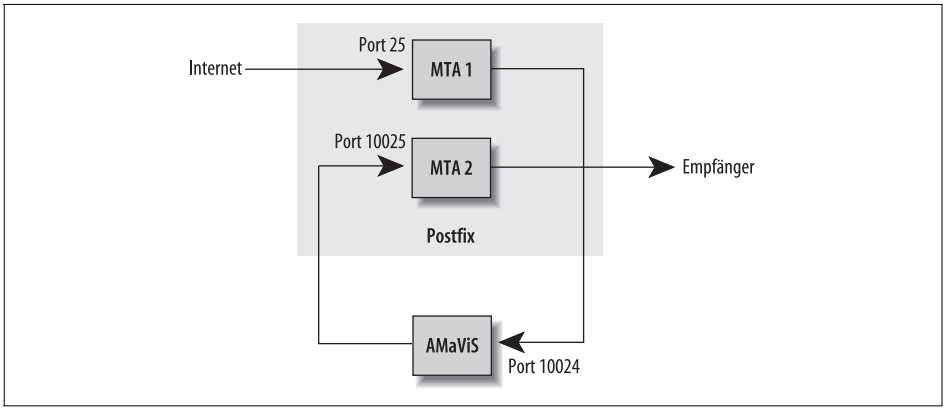

Abbildung 8-2: Fluss einer E-Mail durch ein Postfix/AMaViS-System

Zuerst konfiguriert man die Verbindung von Postfix zu AMaViS. Dazu bearbeitet man die Konfigurationsdatei */etc/postfix/master.cf* und fügt folgende Zeilen ans Ende an:

```
smtp-amavis unix -      -       y/n     -       2  smtp
    -o smtp_data_done_timeout=1200
    -o smtp_send_xforward_command=yes
    -o disable_dns_lookups=yes
```

Statt `y/n` sollte entweder `y` oder `n` geschrieben werden, abhängig davon, ob der Postfix-Server in einer Chroot-Umgebung läuft oder nicht. Wenn Postfix in einer Chroot-Umgebung läuft, steht in dieser Spalte in allen Zeilen von *master.cf* ein `y`, ansonsten ein `n`. Den gleichen Buchstaben, den die anderen Einträge verwenden, verwendet man auch hier.

Die gezeigten Konfigurationszeilen definieren einen ausgehenden SMTP-Dienst (in Postfix vom Programm *smtp* implementiert) unter dem Namen `smtp-amavis`. Dieser Name wird später verwendet, um E-Mails zum Zweck der Filterung an AMaViS umzuleiten. Die Optionen `-o` legen bestimmte Konfigurationseinstellungen fest, die nur für diesen Dienst gelten sollen. Wichtig sind diese Einstellungen nicht, sie dienen nur der Optimierung, weil man zum Beispiel für die lokale Auslieferung keine DNS-Anfragen benötigt.

Als Nächstes wird die Verbindung von AMaViS zurück zu Postfix konfiguriert. Dazu werden diese Zeilen ebenfalls an die Datei */etc/postfix/master.cf* angefügt (die relative Reihenfolge ist egal):

```
127.0.0.1:10025 inet n  -       y/n     -       -  smtpd
    -o content_filter=
    -o local_recipient_maps=
```

```
    -o relay_recipient_maps=
    -o smtpd_restriction_classes=
    -o smtpd_client_restrictions=
    -o smtpd_helo_restrictions=
    -o smtpd_sender_restrictions=
    -o smtpd_recipient_restrictions=permit_mynetworks,reject
    -o mynetworks=127.0.0.0/8
    -o strict_rfc821_envelopes=yes
    -o smtpd_error_sleep_time=0
    -o smtpd_soft_error_limit=1001
    -o smtpd_hard_error_limit=1000
    -o smtpd_client_connection_count_limit=0
    -o smtpd_client_connection_rate_limit=0
    -o receive_override_options=no_header_body_checks
```

Auch hier sollte man bei `y/n` die passende Wahl treffen.

Diese Zeilen definieren einen eingehenden SMTP-Dienst (in Postfix vom Programm *smtpd* implementiert), der auf der IP-Adresse 127.0.0.1 auf Port 10025 läuft. Dies ist die oben beschriebene zweite Postfix-Instanz. Die Einstellung `-o content_filter=` sorgt hierbei dafür, dass dieser Server *keinen* Content-Filter verwendet, da an dieser Stelle ja bereits ein Content-Filter, nämlich AMaViS, durchlaufen wurde und sonst eine unendliche Schleife entstünde. Die anderen Einstellungen sind wieder optional und schalten diverse Prüfungen ab, die schon von der ersten Postfix-Instanz durchgeführt wurden.

Anschließend kann man den Postfix-Server mit dem Befehl `postfix reload` die Konfiguration neu laden lassen und die zweite Instanz testen:

```
$ telnet localhost 10025
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
220 mail.example.net ESMTP Postfix (Debian/GNU)
QUIT
221 Bye
Connection closed by foreign host.
```

Falls es hierbei Probleme geben sollte, prüft man am besten die Postfix-Log-Dateien, ob es eventuell Probleme bei der Konfiguration gegeben hat.

Nun fehlt noch, dass die erste Postfix-Instanz ihre E-Mails zur Filterung an AMaViS weiterleitet. Bevor das System sozusagen scharf geschaltet wird, sollte man noch einen Blick auf die AMaViS-Konfiguration werfen, um zu überprüfen, ob das eingestellte Verhalten den eigenen Vorstellungen entspricht. Als letzten Test kann man auch eine E-Mail direkt an den AMaViS-Server senden und prüfen, ob sie korrekt ausgeliefert wird. Hier ein Beispiel für eine solche SMTP-Sitzung:

```
$ telnet 127.0.0.1 10024
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 [127.0.0.1] ESMTP amavisd-new service ready
```

```
MAIL FROM:<test@example.com>
250 2.1.0 Sender test@example.com OK
RCPT TO:<postmaster>
250 2.1.5 Recipient postmaster OK
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: test1

test1
.
***  250 2.6.0 Ok, id=31859-01, from MTA: 250 Ok: queued as 90B7F16F
QUIT
```

Die E-Mail sollte anschließend in der Mailbox des Empfängers zu finden sein.

Denkbar wären an dieser Stelle auch Tests mit Viren- oder Spam-Exemplaren. Beispiele dazu finden sich in Kapitel 4, *SpamAssassin* und Kapitel 7, *Virenscanner*.

Um schließlich alle E-Mails an den AMaViS-Server umzuleiten, wird folgende Zeile in die Konfigurationsdatei */etc/postfix/main.cf* eingefügt:

```
content_filter = smtp-amavis:[127.0.0.1]:10024
```

Anschließend muss die Postfix-Konfiguration mit `postfix reload` neu geladen werden.

Wenn es jemals Probleme mit AMaViS geben sollte, kommentiert man einfach diese letzte Zeile aus und umgeht AMaViS damit vorübergehend.

Sind Postfix und AMaViS so konfiguriert, dass sie Log-Meldungen an Syslog senden (was in der Voreinstellung der Fall ist), kann man nun in */var/log/mail* oder */var/log/mail.info* (je nach Betriebssystem) verfolgen, wie die E-Mails Postfix, AMaViS und noch mal Postfix durchlaufen. Beim Einsatz von Log-Analyse-Werkzeugen ist zu beachten, dass jede E-Mail von Postfix zweimal gesehen wird und daher die in der Statistik gezeigte E-Mail-Zahl halbiert werden muss.

Exim

Der Fluss einer E-Mail durch ein Exim/AMaViS-System ist vom Konzept her identisch mit dem soeben für Postfix beschriebenen Setup. Insbesondere werden auch bei Exim die beiden logischen MTA-Instanzen vom selben Server betrieben, wie in Abbildung 8-2 illustriert.

Um ein Exim/AMaViS-System zu konfigurieren, wird zunächst Port 10025 in Exim freigeschaltet. Dazu wird folgende Zeile in den allgemeinen Konfigurationsabschnitt (am Anfang der Konfigurationsdatei) eingefügt beziehungsweise eine bestehende Zeile entsprechend geändert:

```
local_interfaces = 0.0.0.0.25 : 127.0.0.1.10025
```

Wenn man sichergestellt hat, dass die AMaViS-Konfiguration den eigenen Vorstellungen entspricht, kann man nun die Exim-Konfiguration neu laden lassen, wie oben beschrieben eine Test-E-Mail direkt an den AMaViS-Server auf Port 10024

senden und schauen, ob sie zugestellt oder gefiltert wird. Eine E-Mail, die normal an Exim gesendet wird, ist zu diesem Zeitpunkt noch nicht betroffen.

Nun konfiguriert man die Verbindung von Exim zu AMaViS. Dazu bearbeitet man die Exim-Konfigurationsdatei und fügt im Abschnitt `routers` folgenden Block als ersten Router ein:

```
amavis:
        driver = manualroute
        condition = "${if eq {$interface_port}{10025} {0}{1}}"
        domains = +local_domains
        transport = amavis
        route_list = "* localhost byname"
        self = send
```

Dieser Abschnitt besagt, dass alle E-Mails außer denen, die über Port 10025 hereinkamen und damit schon von AMaViS verarbeitet worden sind, an den Transport namens `amavis` übergeben werden sollen. Die Reihenfolge der Router ist wichtig, und durch Variieren der Reihenfolge lassen sich verschiedene spezielle Konfigurationen realisieren. Wenn zum Beispiel nur eingehende E-Mails gefiltert werden sollen, kann der Router `amavis` hinter den Router `remote_smtp` gesetzt werden.

Sollte es jemals Probleme mit AMaViS geben, kann man den obigen Block auskommentieren und damit AMaViS umgehen.

Schließlich wird der Transport `amavis` eingerichtet. Dazu wird in den Abschnitt `transports` folgender Block eingefügt:

```
amavis:
       driver = smtp
       port = 10024
       allow_localhost
```

Der oben definierte Router ordnet E-Mails, die von AMaViS gefiltert werden sollen, diesem Transport zu. Die Definition des Transports besagt, dass die E-Mails per SMTP an Port 10024 gesendet werden.

Anschließend lädt man die Exim-Konfiguration neu, und AMaViS ist somit aktiv.

Sendmail

Dieser Abschnitt beschreibt das so genannte Dual-Sendmail-Setup, das das in Abbildung 8-1 gezeigte Prinzip realisiert. Alternativ gibt es auch das so genannte Sendmail/Milter-Setup, das aber nur eingeschränkte Funktionalität bietet und in diesem Buch nicht behandelt wird. Die Einrichtung des Dual-Sendmail-Setup ist relativ aufwendig und wird nur denjenigen Anwendern empfohlen, die schon einige Erfahrung mit Sendmail haben. Einsteiger sollten zuerst die Einrichtung mit Postfix oder Exim versuchen.

Die AMaViS-Dokumentation bezeichnet die erste MTA-Instanz als Sendmail-RX (»receiving«) und die zweite Instanz als Sendmail-TX (»transmitting«). Die vorhandene Sendmail-Installation wird zur TX-Instanz umfunktioniert, und die RX-In-

stanz muss neu aufgesetzt werden. Dazu werden getrennte Konfigurationsdateien und ein neuer Spool-Bereich eingerichtet.

Für die TX-Instanz werden die folgenden Pfade verwendet. Sie entsprechen den Pfaden einer einfachen Sendmail-Installation.

- Queue-Verzeichnis: */var/spool/mqueue*
- Konfigurationsdatei: */etc/mail/sendmail.cf*
- Quelle der Konfigurationsdateien: *sendmail-tx.mc*

Für die RX-Instanz werden die folgenden Pfade vorgeschlagen.

- Queue-Verzeichnis: */var/spool/mqueue-rx*
- Konfigurationsdatei: */etc/mail/sendmail-rx.cf*, */etc/mail/submit.cf*
- Quelle der Konfigurationsdateien: *sendmail-rx.mc*

Wenn bereits eine Sendmail-Installation vorhanden ist, kann die Konfiguration größtenteils übernommen werden. Die Konfigurationseinstellungen, die sich auf das Annehmen von E-Mail beziehen, insbesondere Ressourceneinstellungen, verschiebt man nach *sendmail-rx.mc*. Die Einstellungen, die sich auf das Ausliefern von E-Mail beziehen, verschiebt man nach *sendmail-tx.mc*. Allgemeine Einstellungen kopiert man in beide Dateien.

Nun muss ein neuer Queue-Bereich für die RX-Instanz angelegt werden. Dazu verwendet man folgende Befehle als root:

```
mkdir /var/spool/mqueue-rx
chown root:wheel /var/spool/mqueue-rx
chmod 700 /var/spool/mqueue-rx
```

Der Eigentümer und die Zugriffsrechte sollten mit dem TX-Queue-Bereich in */var/spool/mqueue* übereinstimmen.

Jetzt werden die Konfigurationsdateien angepasst. Die Datei *sendmail-rx.mc* kontrolliert die Annahme von E-Mail aus dem Internet. Die relevanten Einstellungen können aus der bisherigen Sendmail-Konfiguration übernommen werden. Dazu gehört beispielsweise, welche Domains akzeptiert werden, sowie Client-Authentifizierung, Ressourcenlimits und Milter-Einstellungen. Schließlich wird folgender Block angehängt. Die Bedeutung der einzelnen Einstellungen kann man der Sendmail- oder der AMaViS-Dokumentation entnehmen.

```
define(`confRUN_AS_USER',`smmsp:smmsp')dnl  je nach Betriebssystem

define(`confPID_FILE', `/var/run/sendmail-rx.pid')dnl
define(`STATUS_FILE', `/etc/mail/stat-rx')dnl
define(`QUEUE_DIR', `/var/spool/mqueue-rx')dnl
define(`confQUEUE_SORT_ORDER',`Modification')dnl  oder `Random'

QUEUE_GROUP(`mqueue', `P=/var/spool/mqueue-rx, R=2, F=f')dnl

FEATURE(stickyhost)dnl
```

```
define(`MAIL_HUB',  `esmtp:[127.0.0.1]')dnl
define(`SMART_HOST',`esmtp:[127.0.0.1]')dnl

define(`confDELIVERY_MODE',`q')dnl
define(`ESMTP_MAILER_ARGS',`TCP $h 10024')dnl
MODIFY_MAILER_FLAGS(`ESMTP',`+z')dnl
define(`SMTP_MAILER_MAXMSGS',`10')dnl
define(`confTO_DATAFINAL',`20m')dnl
DAEMON_OPTIONS(`Name=MTA-RX')dnl

undefine(`ALIAS_FILE')dnl
define(`confFORWARD_PATH')dnl
undefine(`UUCP_RELAY')dnl
undefine(`BITNET_RELAY')dnl
undefine(`DECNET_RELAY')dnl

MAILER(smtp)
```

In der Zeile `QUEUE_GROUP` sollte die Anzahl der Queue-Runner (`R=`) mit dem Wert der Variablen `$max_servers` in der AMaViS-Konfiguration übereinstimmen.

Die Datei *sendmail-tx.mc* kontrolliert die Auslieferung von E-Mail. Auch dazu können die relevanten Einstellungen aus der bisherigen Sendmail-Konfiguration übernommen werden. Dazu gehören unter anderem die benötigten Mailer, Aliase und Queue-Einstellungen, nicht jedoch Ressourcenlimits. Die zusätzliche Konfiguration ist nun noch:

```
FEATURE(`no_default_msa')dnl
DAEMON_OPTIONS(`Addr=127.0.0.1, Port=10025, Name=MTA-TX')dnl
define(`confSMTP_LOGIN_MSG', `$w.tx.$m Sendmail $v/$Z; $b')dnl
define(`confTO_IDENT', `0')dnl

MAILER(smtp)
MAILER(local)
```

Jetzt kann man mit `m4` die eigentlichen Konfigurationsdateien erzeugen:

```
# m4 /usr/share/sendmail/cf/m4/cf.m4 sendmail-rx.mc >/etc/mail/sendmail-rx.cf
# m4 /usr/share/sendmail/cf/m4/cf.m4 sendmail-tx.mc >/etc/mail/sendmail.cf
```

Anschließend können beide Sendmail-Instanzen gestartet werden:

```
# /usr/sbin/sendmail -C /etc/mail/sendmail-rx.cf -L sm-mta-rx -bd -qp
# /usr/sbin/sendmail                             -L sm-mta-tx -bd -q15m
```

Mit `telnet` kann man zum Beispiel nun testen, ob SMTP-Verbindungen auf Port 25 und 10025 möglich sind.

Wenn Sendmail beim Booten automatisch gestartet wird, sollte man das entsprechende Skript anpassen, damit automatisch beide Sendmail-Instanzen gestartet und gestoppt werden.

Bei allen Sendmail-Befehlen, die mit der RX-Instanz arbeiten sollen, muss die Konfigurationsdatei explizit angegeben werden, zum Beispiel:

```
mailq -C /etc/mail/sendmail-rx.cf
sendmail -C /etc/mail/sendmail-rx.cf
```

Diese Sendmail-Konfiguration ist zweifellos sehr aufwendig und fehleranfällig. Sehr viel einfacher geht es zum Beispiel mit Postfix und Exim, wie in den vorangegangenen Abschnitten beschrieben wurde.

Konfiguration

Die Konfiguration von AMaViS erfolgt primär über die Konfigurationsdatei *amavisd.conf*, die normalerweise in */etc/amavis/* liegt. Zusätzliche externe Dateien können verwendet werden, um den Text von diversen Benachrichtigungs-E-Mails einzustellen. Diese liegen abhängig vom Betriebssystem entweder ebenfalls unter */etc/amavis/*, unter */usr/share/amavis/* oder unter */usr/local/share/amavis/*. Weitere Konfigurationsdaten können in SQL- oder LDAP-Datenbanken abgelegt werden.

Im Folgenden werden die wichtigsten und interessantesten Konfigurationsoptionen besprochen, insbesondere jene, die ein Administrator vor dem ersten Start anpassen oder zumindest überdenken sollte. Eine vollständige Liste aller Konfigurationsvariablen und -möglichkeiten kann man in der kommentierten Beispielkonfigurationsdatei *amavisd.conf-sample* in der AMaViS-Distribution finden. Die Reihenfolge, in der die Konfigurationsoptionen unten besprochen werden, entspricht in etwa der Reihenfolge in der Beispielkonfigurationsdatei.

Perl-Schnellstart

Die Konfigurationsdatei *amavisd.conf* wird als normales Perl-Skript gelesen. Als Konfigurationssyntax steht somit die gesamte Perl-Sprache zur Verfügung. Es ist zwar normalerweise nicht nötig, in Perl programmieren zu können, da die meisten Konfigurationseinstellungen einfache Variablenzuweisungen sind, es kann aber in komplexeren Setups durchaus von Nutzen sein. Anwender, die von dieser Möglichkeit Gebrauch machen wollen, müssen beachten, dass AMaViS sowohl als »tainted« als auch als »strict« läuft. Das heißt unter anderem, dass alle neuen Variablen deklariert und Daten aus externen Quellen ent-»taint«-et werden müssen. Wenn in diesen Bereichen Fehler gemacht werden, erscheint beim Start des AMaViS-Servers oft ein Perl-Compiler-Fehler.

Für Leser, die sich mit Perl nicht auskennen, seien hier die wichtigsten Syntaxregeln und Idiome, die bei der Konfiguration von AMaViS zur Anwendung kommen, vorgestellt. Um die Funktionalität von AMaViS voll ausnutzen zu können, ist ein detaillierteres Studium der Perl-Syntax und der Funktionsweise von verschachtelten Datenstrukturen jedoch sehr zu empfehlen. Aushilfsweise kann man sich auch an die vielen Beispiele halten und setzt dort nur die eigenen Werte ein.

Alle Befehle in Perl können über mehrere Zeilen gehen und enden mit einem Semikolon. Kommentare fangen mit # an und gehen bis zum Ende der Zeile. Die meisten Konfigurationseinstellungen in AMaViS sind Variablenzuweisungen irgendeiner Art. Die Namen einfacher Variablen, so genannter Scalars, fangen mit $ an. Die Namen von Arrays (mit numerischem Index) beginnen mit @, die Namen von assoziativen Arrays (mit Strings als Index), so genannten Hashes, mit %. Der Zuweisungsoperator ist =. Werte für Scalars sind entweder Zahlen, Strings (gequotet mit " oder ') oder Ausdrücke. Einige Beispiele:

```
# Scalar-Variable mit Zahl als Wert
$log_level = 0;

# Scalar-Variable mit String als Wert
$SYSLOG_LEVEL = 'mail.debug';

# "undef" ist ein Sonderwert, der für "nichts" steht.
$LOGFILE = undef;

# Arithmetischer Ausdruck
$child_timeout = 8*60;

# In mit " gequoteten Strings werden Variablen eingesetzt, in mit ' gequoteten
# Strings dagegen nicht.
$MYHOME   = '/var/amavis';
$pid_file = "$MYHOME/amavisd.pid";

# Sonderzeichen müssen in mit " gequoteten Strings escapt werden.
$virus_admin = "virusalert\@$mydomain";

# Funktionsaufruf
$notify_sender_templ = read_text("$MYHOME/notify_sender.txt");
```

In vielen Einstellungen wird ein Feature durch Angabe eines logischen Werts ein- oder ausgeschaltet. In Perl sind alle Werte logisch wahr mit Ausnahme von 0, "0", leeren Strings und undef. Der Einfachheit halber wird für »wahr« oft die Zahl 1 angegeben.

Werte von Arrays und Hashes werden aus einfachen Ausdrücken aufgebaut. Auch hier einige erklärende Beispiele:

```
# Array mit drei String-Werten
@inet_acl = ('127/8', '10/8', '172.16/12', '192.168/16');

# Alternative Schreibweise mit Leerzeichen als Trennung
@inet_acl = qw( 127/8 10/8 172.16/12 192.168/16 );

# Eckige Klammern ergeben eine Array-Referenz. Die Scalar-Variable fungiert
# als eine Art Pointer auf das Array.
$uncompress = ['uncompress', 'gzip -d', 'zcat'];
```

Funktionsprinzipien

AMaViS erkennt vier verschiedene Arten unerwünschter E-Mails:

Ungültige Header
: Ungültige Header (bad headers) sind E-Mail-Header, die nicht den Standards entsprechend kodiert sind. AMaViS erkennt diese durch eingebaute Prüfungen.

  Die Prüfung auf ungültige Header ist in der Praxis relativ unnötig, weil sie gleichermaßen Spam und legitime E-Mail von mangelhaft konfigurierten E-Mail-Programmen identifiziert. Sie sollte daher nicht eingeschaltet werden.

Verbotene Dateien
: Verbotene Dateien (banned files) sind E-Mail-Anhänge mit unerwünschten Dateitypen. Welche Dateitypen verboten sind, wird in der AMaViS-Konfiguration eingestellt.

  Man sollte diese Funktionalität als eine Art handgesteuerten Virenscanner sehen, wenn man weiß, dass bestimmte Dateitypen oft Ärger machen und normalerweise nicht über E-Mail verschickt werden müssen (zum Beispiel vorgebliche Bildschirmschonerprogramme). Das Durchsetzen von Arbeitsplatzbestimmungen oder Ähnlichem (Beispiel: Anwender dürfen keine MP3-Dateien versenden) ist zwar auch möglich, ist aber nicht das Ziel dieser Implementierung und ließe sich von findigen Anwendern auch leicht umgehen.

Viren
: Viren werden durch externe Virenscanner-Programme erkannt. Virenscanner müssen getrennt installiert und in AMaViS konfiguriert werden. Eine Vielzahl von Virenscannern ist bereits vorkonfiguriert; diese werden automatisch verwendet, wenn sie beim Start des AMaViS-Servers gefunden werden.

Spam
: Spam wird durch das Programm SpamAssassin analysiert. Dazu ermittelt SpamAssassin eine Punktzahl, die anzeigt, wie wahrscheinlich eine E-Mail Spam ist. SpamAssassin wird in Kapitel 4, *SpamAssassin* im Detail besprochen.

Diese vier Prüfungen werden in dieser Reihenfolge durchlaufen. Alle vier Prüfungen können auch mehr oder weniger getrennt konfiguriert werden. In einigen Fällen ist diese Trennung aber nur unscharf. Insbesondere werden aus historischen Gründen das Virenscannen und das Erkennen verbotener Dateien teilweise durch die gleichen Einstellungen konfiguriert.

Allgemeine Einstellungen

Die ersten Einstellungen bestimmen einige Pfadangaben und den Ressourcenverbrauch. Man sollte diese Einstellungen vor dem ersten Start an das eigene System anpassen.

`$MYHOME`
: Diese Variable setzt das Wurzelverzeichnis für verschiedene Pfadangaben. Speziell ist dies das Arbeitsverzeichnis, in dem zu prüfende E-Mails ausgepackt

werden. Außerdem werden in diesem Verzeichnis die Bayes-Datenbank von SpamAssassin und AMaViS-interne Datenbanken angelegt. Dieses Verzeichnis sollte daher auf einer Partition liegen, die ausreichend Platz hat und auf einer schnellen Festplatte liegt. Für maximale Performance kann man dieses Verzeichnis und das Spool-Verzeichnis des Mailservers (zum Beispiel */var/spool/postfix*) auf verschiedene Festplatten legen.

Die Voreinstellung ist */var/amavis*, für Linux-Systeme passt aber */var/lib/amavis* besser in die Dateisystemhierarchie. Dieses Verzeichnis muss angelegt werden, bevor der AMaViS-Server gestartet werden kann. Außerdem muss dem mit `$daemon_user` ausgewählten Benutzer Schreibzugriff auf dieses Verzeichnis gewährt werden. Bei Binärpaketen passiert beides automatisch.

`$mydomain`
: In dieser Variablen wird die eigene E-Mail-Domain eingestellt. In Wahrheit setzt diese Variable lediglich die Voreinstellung für verschiedene andere Einstellungen, insbesondere `@local_domains_maps` und Administrator-E-Mail-Adressen (siehe unten). Wenn die Mailserver-Installation nur eine Domain betreut, stellt man diese hier ein, ansonsten kann man hier seine »Hauptdomain« angeben.

`$daemon_user`
`$daemon_group`
: Diese Variablen bestimmen, unter welchem Benutzer und unter welcher Gruppe der AMaViS-Server laufen soll. Wie alle Server sollte auch der AMaViS-Server nicht als root laufen, sondern unter einem eigenen Benutzer. Typischerweise verwendet man hier `amavis` als Benutzer- und als Gruppenname. Der gewählte Benutzer muss bei einer manuellen Installation von Hand angelegt werden; bei Binärpaketen geschieht dies automatisch.

`$max_servers`
: Diese Variable ist der wichtigste Performance-Tuning-Parameter in AMaViS. Sie bestimmt, wie viele Serverprozesse der AMaViS-Server vorrätig hält. Jeder Serverprozess kann eine E-Mail gleichzeitig bearbeiten. Der Wert dieser Variablen sollte daher mit derjenigen Einstellung im MTA übereinstimmen, die festlegt, wie viele E-Mails parallel angenommen oder an AMaViS ausgeliefert werden. Ein sinnvoller Ausgangswert sind zwei oder drei Serverprozesse pro CPU. Danach sollte man im laufenden Betrieb die CPU-Last und den Speicherverbrauch überwachen. Wenn der Speicher auf der Maschine ausgeht oder die Load-Average pro CPU dauerhaft größer als 2 oder 3 ist, sollte man die Einstellung reduzieren. Ist die Maschine jedoch noch nicht ausgelastet, kann man die Einstellung erhöhen, aber ab 10 sind keine merkbaren Verbesserungen mehr zu erwarten.

`@local_domains_maps`
: Diese Einstellung bestimmt, welche Domains lokal sind. Dadurch wird festgestellt, welche E-Mails eingehend und welche ausgehend sind. Einige Konfigurationseinstellungen und Features unterscheiden zwischen eingehenden und

ausgehenden E-Mails, zum Beispiel Virenbenachrichtigungen. In der Voreinstellung ist `$mydomain` als lokal gelistet. Wenn man mehrere Domains betreut, trägt man diese hier ein. Zur Syntax siehe Abschnitt »Lookup-Maps« unten.

`$insert_received_line`

Wenn diese Variable logisch wahr ist, fügt AMaViS wie ein MTA einen Received-Header in jede E-Mail ein, zum Beispiel einen wie diesen:

```
Received: from mail.example.net ([127.0.0.1])
 by localhost (mail.example.net [127.0.0.1]) (amavisd-new, port 10024)
 with ESMTP id 10684-10 for <peter@example.net>;
 Fri, 18 Feb 2005 23:48:23 +0100 (CET)
```

Dies ist durchaus sinnvoll, damit der Lauf der E-Mail genau protokolliert wird. Wer seine AMaViS-Instanz allerdings verstecken will, kann dies ausschalten.

`$X_HEADER_TAG`

`$X_HEADER_LINE`

Mit diesen Einstellungen kann ein zusätzlicher Header in jede E-Mail eingefügt werden, um anzuzeigen, dass die E-Mail auf Viren gescannt wurde. Die Voreinstellung ist:

```
$X_HEADER_TAG = 'X-Virus-Scanned';
$X_HEADER_LINE = "$myproduct_name at $mydomain";
```

Daraus ergibt sich zum Beispiel folgender Header:

```
X-Virus-Scanned: by amavisd-new at example.net
```

Auch diese Einstellungen dienen der Protokollierung und Überwachung, können aber ohne Probleme abgeschaltet werden, indem die Variablen auf `undef` gesetzt werden.

`$inet_socket_port`

Diese Variable bestimmt, auf welchem TCP-Port der AMaViS-Server SMTP-Verbindungen für zu filternde E-Mail annimmt. In den meisten Situationen sollte diese Einstellung

```
$inet_socket_port = 10024;
```

sein, in Abstimmung mit der Konfiguration des MTA (siehe oben). In einigen Situationen kann es sinnvoll sein, AMaViS für mehrere Ports zu konfigurieren, zum Beispiel um verschiedene Konfigurationen für verschiedene Clients zu verwenden (siehe Abschnitt »Policy-Banks«). In dem Fall enthält diese Variable eine Referenz auf ein Array mit den Portnummern, zum Beispiel:

```
$inet_socket_port = [10024, 10026, 10028];
```

`$inet_socket_bind`

Diese Variable bestimmt, auf welche IP-Adresse sich der AMaViS-Server bindet. Die Voreinstellung ist `'127.0.0.1'`, wodurch nur Verbindungen vom lokalen Rechner ermöglicht werden. In den meisten Fällen laufen MTA und AMaViS auf demselben Rechner, und diese Einstellung ist ausreichend und sinnvoll. Wenn AMaViS nicht auf demselben Rechner wie der MTA läuft, wird hier entweder die IP-Adresse des entsprechenden externen Netzwerk-Interface eingetragen oder `undef`, um auf alle Interfaces zu binden.

AMaViS ist nicht dafür vorgesehen, auf einem für jedermann zugänglichen Port zu laufen. Es hat keine besonderen Schutzmechanismen für Angriffe aus dem Internet, wie MTA sie normalerweise haben. Daher sollte man, wenn man AMaViS überhaupt für externe Verbindungen freischalten muss, mit Hilfe dieser und der nächsten Einstellung sowie Firewall-Regeln größtmöglichen Schutz einrichten.

`@inet_acl`
: Diese Variable bestimmt, von welchen IP-Adressen AMaViS Verbindungen akzeptiert. Dies unterscheidet sich von `$inet_socket_bind` dadurch, dass Letzteres die »Sichtbarkeit« des AMaViS-Servers einschränkt, wohingegen diese Variable nur relevant wird, wenn vom Kernel bereits eine TCP/IP-Verbindung aufgebaut worden ist. Die Voreinstellung ist:

```
@inet_acl = qw( 127.0.0.1 ::1 );
```

Soll AMaViS Verbindungen von einem MTA auf einem anderen Rechner annehmen, muss diese Variable sowie `$inet_socket_bind` entsprechend angepasst werden.

`$DO_SYSLOG`
: Wenn diese Variable logisch wahr ist, wird über den Syslog-Dienst geloggt, ansonsten in eine Datei. Die Verwendung von Syslog ist zu empfehlen.

`$log_level`
: Diese Variable bestimmt, wie viele Details in den Log geschrieben werden. Sinnvolle Werte gehen von 0 (sehr wenig, nur Fehler) bis 5 (sehr viel, einschließlich Debug-Meldungen). Die Voreinstellung ist 0, die beste Balance bietet jedoch die Einstellung 2. Damit wird unter anderem für jede verarbeitete E-Mail eine Zeile mit dem Erkennungsergebnis (Spam/Virus/sauber) in den Log geschrieben, was die Überwachung des E-Mail-Verkehrs erleichtert. Log-Analysewerkzeuge benötigen ebenfalls diese Informationen, um Statistiken über den E-Mail-Durchsatz zu erstellen (siehe zum Beispiel Abschnitt »Statistiken mit Amavis-stats« unten).

Verhalten

Die interessantesten Einstellungen in AMaViS sind die, die bestimmen, wie mit gescannten E-Mails im Detail verfahren wird. Hier kann man einstellen, ob unerwünschte E-Mails verworfen werden, nur markiert werden, ob Benachrichtigungen versendet werden und mehr.

`$final_bad_header_destiny`
`$final_banned_destiny`
`$final_virus_destiny`
`$final_spam_destiny`
: Diese Einstellungen bestimmen, was mit einer E-Mail passiert, die einen ungültigen Header, eine verbotene Datei oder ein Virus enthält beziehungsweise als

Spam eingestuft wird. Jede dieser Variablen kann auf einen der folgenden vordefinierten Werte gesetzt werden:

`D_PASS`
: Die E-Mail wird an die beabsichtigten Empfänger ausgeliefert, egal ob ein ungültiger Header, eine verbotene Datei, ein Virus oder Spam erkannt worden ist. Andere Aktionen, wie das Einfügen von zusätzlichen E-Mail-Headerzeilen oder das Umschreiben der Betreffzeile werden, falls konfiguriert, trotzdem durchgeführt.

`D_DISCARD`
: Die E-Mail wird angenommen und ohne Benachrichtigung verworfen. Das heißt, die E-Mail geht im Prinzip verloren. Diese Einstellung wird auch für das Quarantänisieren genutzt (siehe weiter unten).

`D_BOUNCE`
: Die E-Mail wird nicht an die Empfänger ausgeliefert. Stattdessen wird eine Benachrichtigung an den Absender gesendet, die besagt, dass die E-Mail nicht zugestellt wurde (so genannte Delivery Status Notification – DSN). Eine DSN wird nicht gesendet, wenn die E-Mail ein Virus enthält und AMaViS der Meinung ist, dass der Absender gefälscht ist, oder wenn die E-Mail von einer Mailingliste kam oder wenn die Spam-Punktzahl `$sa_dsn_cutoff_level` übersteigt.

`D_REJECT`
: Die E-Mail wird nicht an die Empfänger ausgeliefert. Stattdessen versucht AMaViS, die Ablehnung an den Mailserver zu kommunizieren, der dann eine DSN an den Absender senden könnte. Diese Einstellung hat den gleichen Endeffekt wie `D_BOUNCE`. Der Unterschied ist nur, dass bei `D_REJECT` der Mailserver die DSN sendet, während bei `D_BOUNCE` AMaViS dies selbst tut. Die Einstellung `D_REJECT` funktioniert nur mit Sendmail/Milter, nicht mit den oben beschriebenen Setups für Postfix, Exim und Dual-Sendmail.

Von der Verwendung von `D_BOUNCE` und `D_REJECT` ist grundsätzlich abzuraten. Die Absenderadressen von Viren und Spam sind heutzutage fast ausschließlich gefälscht, und es ist daher nahezu unmöglich, sinnvolle DSN für solche E-Mails zu verschicken. Die in AMaViS eingebaute Heuristik, die bei gefälschten Absenderadressen DSN unterbinden soll, funktioniert nur in den seltensten Fällen. Dazu kommen zwei weitere Problematiken: Erstens sind viele Anwender wegen der Vielzahl der fehlgeleiteten Virenbenachrichtigungen dazu übergegangen, solche Meldungen prinzipiell zu ignorieren oder gar selbst als eine Art Spam zu blocken. Und zweitens spielen solche Benachrichtigungen vielen Viren sogar in die Hände, weil sie die Denial-of-Service-Attacken noch verschlimmern, teilweise exponentiell. Wer E-Mail nicht ausliefern, aber auch nicht kommentarlos verwerfen will, sollte daher die Einrichtung einer Quarantäne in Erwägung ziehen (siehe unten).

Die Voreinstellungen sind: D_PASS für ungültige Header, D_BOUNCE für verbotene Dateien und Spam sowie D_DISCARD für Viren.

Die besten Einstellungen sind wohl: D_PASS für ungültige Header, D_DISCARD für verbotene Dateien und Viren, bei Bedarf in Verbindung mit einer Quarantäne, sowie D_PASS für Spam mit nachträglicher Aussortierung im Client.

`$warnbadhsender`
`$warnbannedsender`
`$warnvirussender`
`$warnspamsender`

Wenn diese Variablen logisch wahr sind, wird eine Warnmeldung an den *Absender* einer E-Mail gesendet, wenn sie einen ungültigen Header, eine verbotene Datei oder ein Virus enthält beziehungsweise als Spam eingestuft wurde. Diese Warnmeldung wird auch gesendet, wenn `$final_*_destiny` auf D_PASS gesetzt ist. Wenn `$final_*_destiny` auf D_DISCARD gesetzt ist und das entsprechende `$warn*sender` auf wahr, wird aus dem D_DISCARD automatisch ein D_BOUNCE.

Alle vier Werte sind in der Voreinstellung falsch. Wie oben beschrieben, ist vom Versenden von Warnmeldungen aller Art an Absender prinzipiell abzuraten.

`$warnbadhrecip`
`$warnbannedrecip`
`$warnvirusrecip`

Wenn diese Variablen logisch wahr sind, dann wird eine Warnmeldung an den *Empfänger* einer E-Mail gesendet, wenn sie einen ungültigen Header, einen verbotenen Anhang beziehungsweise ein Virus enthält. Sinnvoll sind diese Einstellungen in Verbindung mit D_DISCARD und einer Quarantäne. Dann erhält der Empfänger die E-Mail zunächst nicht, hat aber die Möglichkeit, bei Bedarf auf die Benachrichtigung hin die eigentlich verworfene E-Mail aus der Quarantäne zu holen. Die Voreinstellung für alle drei Werte ist falsch. (Man beachte, dass es keine analoge Einstellung für Spam gibt.)

`$warn_offsite`

Wenn diese Variable logisch wahr ist, werden Warnmeldungen an den *Empfänger* einer E-Mail (laut `$warn*recip`) auch dann gesendet, wenn der Empfänger nicht zu einer lokalen Domain gehört. Dies wird über die Variable `@local_domains_maps` bestimmt. Die Voreinstellung ist »aus«, und diese Einstellung ist in der Regel auch angebracht, denn man möchte in den seltensten Fällen externe Personen darüber informieren, dass die eigenen Anwender Viren versenden. Sinnvoller ist die Benachrichtigung des Administrators in diesen Fällen (siehe nächste Einstellung).

`$virus_admin`
`$spam_admin`

Diese Variablen bestimmen E-Mail-Adressen, an die zusätzlich Benachrichtigungen gesendet werden sollen, wenn eine E-Mail ein Virus (oder eine verbotene Datei) enthält beziehungsweise als Spam eingestuft wurde. Wenn `D_DISCARD` mit einer Quarantäne verwendet wird, ist es eventuell sinnvoll, hiermit den E-Mail-Verkehr von einem Administrator überwachen zu lassen. Dies erfordert natürlich zusätzlichen personellen Aufwand.

Die Voreinstellung für beide Werte ist `undef`, das heißt, es werden keine Nachrichten an Administratoren versendet. Denkbare Einstellungen sind zum Beispiel:

```
$virus_admin = "virusalert\@$mydomain";
$spam_admin = "spamalert\@$mydomain";
```

Natürlich müssen diese E-Mail-Adressen irgendwo einrichtet werden.

`$mailfrom_notify_recip`
`$mailfrom_notify_admin`
`$mailfrom_notify_spamadmin`

Diese Einstellungen bestimmen den Envelope-Absender, der bei Benachrichtigungen an den Empfänger einer E-Mail, an den Administrator bei einem erkannten Virus beziehungsweise an den Administrator bei erkanntem Spam verwendet wird. Die Voreinstellungen sind in allen drei Fällen `undef`, womit ein so genannter Null-Return-Path verwendet wird (`MAIL FROM: <>`), der keine Bounces zulässt. Das ist in den meisten Fällen die richtige Einstellung. Alternativ kann man eine E-Mail-Adresse eintragen, die dann auch Bounces erhalten würde.

`$hdrfrom_notify_sender`
`$hdrfrom_notify_admin`
`$hdrfrom_notify_spamadmin`

Diese Einstellungen bestimmen den Absender in der `From:`-Zeile bei Benachrichtigungen an den Absender einer E-Mail, an den Administrator beziehungsweise an den Administrator bei erkanntem Spam. Dies sollte eine gültige E-Mail-Adresse sein, an die Anwender bei eventuellen Fragen eine Antwort-E-Mail senden können. Die Voreinstellung ist:

```
"\"Content-filter at $myhostname\" <postmaster\@$myhostname>"
```

Wer zum Beispiel Postmaster und Content-Filter-Administration trennen möchte, kann hier eine andere E-Mail-Adresse angeben.

Die Texte der verschiedenen Benachrichtigungs-E-Mails, die hier konfiguriert werden, sind von AMaViS vorgegeben. Sie informieren den Empfänger, was an der E-Mail zu beanstanden war, wie mit ihr verfahren wurde (zum Beispiel verworfen) und welche Möglichkeiten es gibt, sie zurückzubekommen (zum Beispiel aus der Quarantäne). Die Texte können über die Konfigurationsdatei angepasst werden; dieses Verfahren kann hier aus Platzgründen nicht beschrieben werden.

Quarantäne

Wenn man E-Mails mit Viren, Spam oder anderem abgelehnten Inhalt weder an den Empfänger weiterleiten möchte noch die E-Mail ohne weiteres verwerfen möchte, bietet AMaViS die Möglichkeit, derartige E-Mails in einem Quarantäne-Bereich abzulegen. Die Idee dahinter ist, dass der beabsichtigte Empfänger die E-Mail in Einzelfällen mit Hilfe eines Administrators oder einer anderen berechtigten Person aus dem Quarantäne-Bereich herausholen kann, wenn sich herausstellt, dass die Viren- oder Spam-Erkennung einen Fehler gemacht hat. Der Quarantäne-Bereich kann ein spezielles Verzeichnis oder ein Postfach auf irgendeinem Mailserver sein. Wichtig ist, dass die Quarantäne, wie Quarantänen in anderen Bereichen, nur berechtigten Personen zugänglich ist und dass verdächtige E-Mails die Quarantäne erst nach sorgfältiger Prüfung verlassen können.

Es gibt getrennte Quarantäne-Einstellungen für ungültige Header, verbotene Dateien, Viren und Spam. Der eigentliche Quarantäne-Bereich kann aber für alle Arten derselbe sein. Am sinnvollsten ist eine Quarantäne für die Kategorien »verbotene Dateien« und »Viren«, da diese E-Mails gefährlich für Endanwender sind. Wenn eine Quarantäne verwendet wird, sollte die entsprechende Einstellung `$final_*_destiny` nicht `D_PASS` sein, da sonst das Quarantäne-Konzept ausgehebelt würde. Die normale Wahl wäre `D_DISCARD`. Außerdem sollte die entsprechende Einstellung `$warn*recip` an sein, damit der Empfänger informiert wird, wenn eine E-Mail in die Quarantäne überführt wurde. (Daraus ergibt sich, dass eine Quarantäne für Spam sinnlos ist, denn das würde den E-Mail-Fluss insgesamt noch erhöhen statt ihn zu reduzieren.)

Bevor man sich zur Verwendung einer Quarantäne entschließt, sollte man bedenken, dass ein Quarantäne-Bereich nicht nur eingerichtet, sondern auch betrieben werden muss. Dazu gehört, dass die Endanwender darüber unterrichtet werden, wie sie E-Mails aus der Quarantäne erhalten können, dass Administratoren auf diese Anfragen reagieren und dass E-Mails nach einer Zeit aus der Quarantäne gelöscht werden.

Es gibt zwei prinzipielle Varianten, wie der Quarantäne-Bereich organisiert werden kann. Der Quarantäne-Bereich kann ein Verzeichnis auf dem Rechner sein, auf dem AMaViS läuft. Das hat den Vorteil, dass dieses sehr einfach einzurichten ist und die in Quarantäne geschickten E-Mails im Klartext vorliegen. Der Nachteil ist, dass es eventuell nicht besonders komfortabel ist, an die aussortierten E-Mails heranzukommen oder sie weiterzuleiten. Alternativ kann man die E-Mail-Nachrichten per SMTP zur Quarantäne an ein beliebiges E-Mail-Postfach senden lassen, entweder auf demselben Rechner wie AMaViS oder auf einem anderen. Das hat den Vorteil, dass man die Konfigurationsmöglichkeiten des MTA nutzen kann, um die E-Mails auszuliefern. Zum Beispiel kann man Alias-Tabellen verwenden, Procmail einsetzen und ist auch beim File-Locking auf der sicheren Seite. Außerdem kann man einfach einen E-Mail-Client verwenden, um die Quarantäne zu verwalten. Der Nachteil ist,

dass man vorsichtig sein muss, um bei der Inspektion der Quarantäne nicht eben jene Viren zu aktivieren, die zuvor von AMaViS aussortiert worden sind, weil zum Beispiel der E-Mail-Client eine Sicherheitslücke hat oder ungefragt Anhänge öffnet. Außerdem ist diese Variante etwas komplizierter in der Installation.

Denkbar wäre zum Beispiel folgendes Arrangement: Die Quarantäne wird in einem Postfach auf demselben Rechner abgelegt, auf dem AMaViS läuft. Dazu ist in der Regel keine zusätzliche Konfiguration im MTA erforderlich. Als E-Mail-Client wird ein einfaches textbasiertes Programm oder eventuell ein Webmail-Paket mit wenigen »dynamischen« Features verwendet. Zugriff auf das Postfach erfolgt entweder direkt über das Dateisystem oder über lokales IMAP. Dadurch kann kein Virus das Filtersystem ohne manuelles Eingreifen verlassen.

Um E-Mails aus der Quarantäne weiterzuleiten, kann einfach die entsprechende Funktion des E-Mail-Programms verwendet werden. Bei einer Quarantäne auf demselben Rechner wie AMaViS muss man jedoch darauf achten, dass diese weitergeleiteten E-Mails nicht noch einmal gescannt werden. Dazu setzt man im E-Mail-Client als Versandmethode SMTP auf Port 10025, die zweite MTA-Instanz. Wenn die Quarantäne auf einem anderen Rechner liegt, gibt es dieses Problem nicht, und man kann ganz normal SMTP auf Port 25 als Versandmethode verwenden. Auch hier muss man allerdings darauf achten, dass nicht zusätzliche Virenscanner oder andere Filtersysteme die weitergeleitete E-Mail aufhalten.

Die im Folgenden besprochenen Variablen konfigurieren den AMaViS-Quarantäne-Bereich.

`$bad_header_quarantine_to`
`$banned_quarantine_to`
`$virus_quarantine_to`
`$spam_quarantine_to`

Diese Variablen bestimmen das Quarantäneverfahren von E-Mails, wobei diese für ungültige Header, verbotene Dateien, Viren und Spam unterschiedlich sein können, was aber meistens nicht sinnvoll ist.

Wenn der Wert einer Variablen `undef` oder ein leerer String ist, wird für die E-Mail der entsprechenden Kategorie keine Quarantäne verwendet. Die E-Mail wird in diesem Fall ausschließlich entsprechend der Einstellung `$final_*_destiny` verarbeitet.

Wenn ein Wert kein @ enthält, wird der Wert über eine spezielle Alias-Tabelle aufgelöst und die zu quarantänisierende E-Mail entsprechend dem Ergebnis in einem lokalen Verzeichnis abgelegt. Vordefinierte Aliase sind `'bad-header-quarantine'`, `'banned-quarantine'`, `'virus-quarantine'` und `'spam-quarantine'`, die auch die Voreinstellungen für die entsprechenden Variablen sind. Diese Aliase führen alle dazu, dass die E-Mails unter dem durch die Variable `$QUARANTINEDIR` bestimmten Verzeichnis abgelegt werden.

Wenn ein Wert ein @ enthält, ist dies eine E-Mail-Adresse, an die zu quarantänisierende E-Mail-Nachrichten gesendet werden. Mögliche Werte sind sowohl vollständige E-Mail-Adressen wie `"infected\@$mydomain"` als auch implizit lokale Adressen wie `"virus-quarantine\@"`.

`$QUARANTINEDIR`

Diese Einstellung bestimmt das Wurzelverzeichnis für den Quarantäne-Bereich, wenn eine lokale Quarantäne verwendet wird (siehe vorige Einstellung). Die Voreinstellung ist `undef`, das heißt, es gibt keine Quarantäne. Ein sinnvolles Verzeichnis wäre zum Beispiel */var/lib/amavis/virusmails* (kein Schrägstrich am Ende).

Wenn `$QUARANTINEDIR` ein bestehendes Verzeichnis ist, werden quarantänisierte E-Mails in einzelnen Dateien in diesem Verzeichnis abgelegt, wie beim Maildir-Postfach-Format. Ansonsten wird `$QUARANTINEDIR` als Datei verstanden, und quarantänisierte E-Mails werden an die Datei angehängt, wie beim Mbox-Postfach-Format (einschließlich der `From`-Zeile am Anfang). Auf beide Formate kann man auch mit einem E-Mail-Client oder IMAP-Server zugreifen, der das entsprechende Format versteht. Das Mbox-Format verwendet jedoch einen Lock-Mechanismus, der sehr ineffizient und nicht NFS-sicher ist. Wer solchen Problemen aus dem Weg gehen möchte, der sollte die zu quarantänisierenden E-Mails per SMTP über einen MTA ausliefern lassen.

Verbotene Dateien

In diesem Abschnitt wird beschrieben, wie man AMaViS konfiguriert, um Anhänge mit bestimmten Dateitypen zu filtern. Zu beachten ist, dass in diesem Fall die ganze E-Mail gefiltert wird. AMaViS kann nicht einzelne Anhänge aus einer E-Mail entfernen.

Der Zweck dieses Features ist, dass man Dateitypen, die Anwendern oft Ärger machen und normalerweise nicht über E-Mail verschickt werden müssen, herausfiltern kann, selbst wenn sie keine Viren im eigentlichen Sinn sind. Dazu gehören zum Beispiel vorgebliche Bildschirmschonerprogramme oder bestimmte MIME-Typen, die selten sinnvoll sind und oft E-Mail-Programme verwirren. Außerdem kann durch Blockieren von bestimmten Dateitypen schnell auf neu entdeckte Viren oder andere Schwachstellen reagiert werden, indem man den entsprechenden Dateityp komplett blockiert, bis eine bessere Lösung gefunden ist. Dieses Feature kann allerdings *nicht* verhindern, dass Anwender bestimmte Dateiarten in das interne Rechnernetz einschleusen. Es verhindert nur, dass unvorsichtige Anwender bestimmte Dateiarten per E-Mail erhalten. Wie Ersteres verhindert werden kann, ist nicht Thema dieses Buchs.

Der Typ einer als E-Mail-Anhang verschickten Datei wird anhand folgender Kriterien bestimmt:

- der deklarierte MIME-Typ der Datei (E-Mail-Header `Content-Type`)
- der vorgeschlagene Dateiname der Datei (Subfeld `name` im E-Mail-Header `Content-Type` sowie Subfeld `filename` im E-Mail-Header `Content-Disposition`)
- der mit dem Unix-Programm `file` ermittelte Dateityp

Damit werden unerwünschte Dateien sowohl erkannt, wenn sie einen bestimmten Namen haben, als auch, wenn der Inhalt einem bestimmten Dateityp entspricht, egal wie der Dateiname lautet.

Dies sei an einem Beispiel illustriert. Ein JPEG-Bild wird als Anhang kodiert im Base64-Format verschickt. Der Anfang des entsprechenden MIME-Parts sieht so aus:

```
--Boundary_(ID_fl/pYw2qfD9IVBPNZXA7PQ)
Content-Type: image/jpeg; name=beispiel.JPG
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=beispiel.JPG

/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0a
HBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIy
...
```

Diese Datei würde blockiert werden, wenn einer der folgenden Einträge in der Blockliste stehen würde:

- ein regulärer Ausdruck, der auf `image/jpeg` passt
- ein regulärer Ausdruck, der auf `beispiel.JPG` passt

Wenn diese Datei abgespeichert und `file` ausgeführt wird, ergibt sich Folgendes:

```
$ file beispiel.JPG
beispiel.JPG: JPEG image data, JFIF standard 1.01
```

AMaViS parst diese Ausgabe und »übersetzt« sie zurück in eine Dateiendung, die diese Dateiart typischerweise haben würde, in diesem Fall `.jpg`. (Diese Werte beginnen immer mit einem Punkt.) Die Datei würde also auch blockiert werden, wenn der folgende Eintrag in der Blockliste stehen würde:

- ein regulärer Ausdruck, der auf `.jpg` passt

Die Übersetzung der `file`-Ausgabe in Dateiendungen wird intern von AMaViS erledigt und hat nichts mit dem eigentlichen Dateinamen zu tun, sondern dient nur der einfacheren Konfiguration. Ein Eintrag von `.jpg` in die Blockliste würde somit nämlich Dateien erkennen, die vom Namen her vorgeben, JPEG-Dateien zu sein, also auch Dateien, die intern wirklich JPEG-Dateien sind, auch wenn sie (zur Tarnung?) anders heißen. Außerdem muss man sich dadurch nicht um die Details des `file`-Ausgabeformats kümmern, das nicht einheitlich oder standardisiert ist.

Die Internet Assigned Numbers Authority (IANA) verwaltet die Liste gültiger MIME-Typen, die unter *http://www.iana.org/assignments/media-types/* eingesehen werden kann. Informationen über Dateiendungen gibt es auf verschiedenen Websites, unter anderem *http://filext.com/*. Da beide Angaben aber in der Praxis variieren können, nicht zuletzt um E-Mail-Filterprogramme zu täuschen, sollten die Angaben in tatsächlich erhaltenen E-Mail-Exemplaren bei der Erstellung von Blocklisten mit einbezogen werden.

Die Beispielkonfiguration von AMaViS enthält bereits eine umfangreiche Liste von Dateitypen, deren Blockierung im Mailserver sinnvoll sein könnte.

Die zu blockierenden Dateitypen werden durch die Variable `$banned_filename_re` festgelegt. Der Wert dieser Variablen ist eine Liste aus regulären Ausdrücken, die durch einen speziellen Konstruktur erzeugt wird. Hier ist ein Beispiel:

```
$banned_filename_re = new_RE(
    qr'^image/jpeg$'i,
    qr'\.jpg$'i,
);
```

Die Funktion `new_RE()` ist vorgegeben; die eigentlichen regulären Ausdrücke werden durch den Operator `qr` definiert. Informationen zu regulären Ausdrücken findet man in Anhang A, *Das SMTP-Protokoll* sowie in der Perl-Dokumentation, insbesondere der Manpage `perlre`. Das `i` am Ende sorgt jeweils dafür, dass zwischen Groß- und Kleinschreibung nicht unterschieden wird.

Um mehrere Dateiformate in einem Ausdruck anzugeben, kann man folgende Syntax verwenden:

```
qr'^\.(Z|gz|bz2)$'
```

Es ist zu beachten, dass reguläre Ausdrücke, die einen Punkt am Anfang erzwingen (wie der gerade gezeigte), nur auf mit dem Programm `file` ermittelte Formate passen, da diese immer mit einem Punkt starten, MIME-Typen und Dateinamen in der Regel jedoch nicht. Diese Unterscheidung kann man ausnutzen, wenn man lieber nach Dateiname oder nach tatsächlichem Dateityp filtern möchte. Sinnvoll ist das normalerweise aber nicht.

`$banned_filename_re` ist ein Fall einer Lookup-Map (siehe Abschnitt »Lookup-Maps«), das heißt, es kann sowohl positive als auch negative Einträge enthalten. Die bisher gezeigten Einträge waren Abkürzungen für positive Einträge (also Einträge, die zu einer Filterung der E-Mail führen). In Langschreibweise würden sie so aussehen:

```
$banned_filename_re = new_RE(
    [ qr'^\.(Z|gz|bz2)$' => 1 ],
    ...
);
```

Ein negativer Eintrag gibt an, ein bestimmtes Dateiformat *nicht* zu filtern. Dazu verwendet man folgende Konstruktion:

```
$banned_filename_re = new_RE(
    [ qr'^\.(Z|gz|bz2)$' => 0 ],
    ...
);
```

Die lange und die kurze Schreibweise können in einer Liste gemischt verwendet werden.

Die gelisteten Ausdrücke werden für jede in der E-Mail enthaltenen Datei in der angegebenen Reihenfolge angewendet. Wenn die Datei in einem Archiv oder einem Multipart-MIME-Teil enthalten ist, werden die Ausdrücke zuerst für das Archiv beziehungsweise den Multipart-Teil, dann für die Datei selbst angewendet, bei tieferen Schachtelungen analog beginnend beim äußersten Archiv. Sobald dabei eine Übereinstimmung gefunden wurde, wird die gesamte Suche für diese Datei beendet. Regeln für Archivtypen haben also Vorrang vor den Regeln für die enthaltenen Dateien. Somit kann man durch Kombination von positiven und negativen Regeln verschiedene Effekte erzielen. Die Einstellung

```
$banned_filename_re = new_RE(
    [ qr'^\.zip$' => 0 ],
    [ qr'^\.exe$' => 1 ],
);
```

würde zum Beispiel *.exe*-Dateien verbieten, außer wenn sie sich in einem ZIP-Archiv befinden.

Die Voreinstellung von `$banned_filename_re` ist `undef`. Die Beispielkonfigurationsdatei von AMaViS enthält eine ganze Reihe von Einstellungen zu verbotenen Dateitypen, die man je nach den eigenen Anforderungen übernehmen kann.

SpamAssassin

AMaViS hat eine eingebaute SpamAssassin-Instanz, kann also die komplette Spam-Erkennung abwickeln. Dies bietet sich an, wenn der Einsatz von SpamAssassin ebenfalls geplant ist, da die Konfiguration und Administration so zentralisiert werden kann und das Scannen der E-Mails beschleunigt wird.

Wenn SpamAssassin schon an anderer Stelle eingebunden ist, sollte man entweder diese Instanz entfernen oder die Einbindung in AMaViS abschalten, um doppelte Arbeit zu vermeiden. Die SpamAssassin-Instanz in AMaViS wird mit folgender Einstellung *aus*geschaltet:

```
@bypass_spam_checks_maps  = (1);
```

Wenn das in AMaViS integrierte SpamAssassin verwendet wird, übernimmt AMaViS auch die Konfiguration von SpamAssassin. Einige der wichtigsten SpamAssassin-Einstellungen sind auf AMaViS-Konfigurationsvariablen abgebildet. Es besteht

jedoch nach wie vor die Möglichkeit, die normalen SpamAssassin-Konfigurationsdateien wie */etc/mail/spamassassin/local.cf* zu verwenden. Man sollte aber beachten, dass das Home-Verzeichnis der AMaViS-Instanz durch die Variable `$MYHOME` bestimmt wird, also zum Beispiel */var/lib/amavis* ist. Die »benutzerspezifische« SpamAssassin-Konfigurationsdatei wäre dann in diesem Fall */var/lib/amavis/.spamassassin/user_prefs*. Wirkliche benutzerspezifische Konfigurationseinstellungen sind bei der Einbindung von SpamAssassin in AMaViS nicht möglich. Das kann eventuell ein Grund sein, diesen Ansatz nicht zu wählen.

Folgende SpamAssassin-Konfigurationseinstellungen können in der AMaViS-Konfigurationsdatei vorgenommen werden. Weitere Informationen zur Konfiguration von SpamAssassin finden sich in Kapitel 4, *SpamAssassin*.

`$sa_local_tests_only`
: Wenn diese Einstellung wahr ist, werden nur SpamAssassin-Tests ausgeführt, die keinen Netzwerkzugriff benötigen (zum Beispiel keine DNSBL-Abfragen). Die Voreinstellung ist falsch, das heißt, Tests, die Netzwerkzugriff benötigen, werden durchgeführt.

`$sa_tag_level_deflt`
: Diese Einstellung bestimmt, ab welcher Spam-Punktzahl (inklusive) von AMaViS Header zur Information über die Spam-Punktzahl eingefügt werden, genauer gesagt die Header `X-Spam-Status` und `X-Spam-Level`. Dies ist nicht die Grenze zur Klassifizierung als Spam, sondern nur die Grenze, ab der die Spam-Punktzahl überhaupt berichtet wird. Die Vorgabe in der Beispielkonfiguration ist 2,0. Wenn man die Variable auf `undef` setzt, werden die Header immer eingefügt. Um das Verhalten von SpamAssassin in allen Situationen beobachten zu können, erscheint letztere Einstellung empfehlenswert.

`$sa_tag2_level_deflt`
: Diese Einstellung bestimmt, ab welcher Spam-Punktzahl (inklusive) eine E-Mail als Spam eingestuft wird. Eine E-Mail, die diese Punktzahl erreicht, erhält einen Header `X-Spam-Flag: YES` (zum einfachen Aussortieren im Client) und abhängig von der Einstellung `$sa_spam_modifies_subj` eine umgeschriebene Betreffzeile. Die Vorgabe in der Beispielkonfiguration ist 6,31.

`$sa_kill_level_deflt`
: Diese Einstellung bestimmt, ab welcher Spam-Punktzahl (inklusive) die in `$final_spam_destiny` angegebene Aktion ausgeführt werden soll (`D_BOUNCE`, `D_DISCARD` oder `D_REJECT`) und, falls konfiguriert, die E-Mail in die Quarantäne gelegt werden soll. Die Vorgabe in der Beispielkonfiguration ist gleich `$sa_tag2_level_deflt`. Es ist normalerweise sinnvoll, diese Werte gleich zu behalten.

`$sa_spam_modifies_subj`
`$sa_spam_subject_tag`

Diese beiden Einstellungen konfigurieren, ob die Betreffzeile einer als Spam eingestuften E-Mail umgeschrieben werden soll. Wenn dies gewünscht ist, muss `$sa_spam_modifies_subj` auf wahr gesetzt und das gewünschte Präfix in `$sa_spam_subject_tag` eingetragen werden. Denkbar wäre zum Beispiel folgende Konfiguration:

```
$sa_spam_modifies_subj = 1;
$sa_spam_subject_tag = '***SPAM*** ';
```

Eine E-Mail mit der Betreffzeile

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

würde, wenn als Spam eingestuft, mit der Betreffzeile

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

beim Empfänger ankommen. In der Voreinstellung ist `$sa_spam_modifies_subj` wahr, aber `$sa_spam_subject_tag` ist `undef`, womit dieses Feature ausgeschaltet ist.

`$sa_spam_level_char`

In eine E-Mail über dem ersten Tag-Level (siehe oben) kann ein Header `X-Spam-Level` eingefügt werden, der die Spam-Punktzahl symbolisch durch die Wiederholung eines Symbols darstellt. Eine E-Mail mit Punktzahl 5 (gerundet) würde zum Beispiel einen solchen Header erhalten, wenn der Stern das gewählte Symbol ist:

```
X-Spam-Level: *****
```

Dieser Header kann nützlich zum automatischen Sortieren von Spam mit Procmail oder ähnlicher Software sein. (Er ist einfacher zu parsen als eine numerische Angabe.)

Um diesen Header abzuschalten, setzt man die Variable auf `undef`. Ansonsten bestimmt diese Variable, welches Zeichen zur Darstellung der Punktzahl verwendet werden soll. Die Voreinstellung ist

```
$sa_spam_level_char = '*';
```

Dies entspricht der SpamAssassin-Einstellung

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

`$sa_spam_report_header`

Diese Variable bestimmt, ob der Header X-Spam-Report in die E-Mail eingefügt wird. In diesem Header wird aufgeschlüsselt, aus welchen Tests sich die Spam-Punktzahl zusammensetzt. Dies ist zum Debuggen nützlich, ansonsten eher nicht. In der Voreinstellung ist dieser Header abgeschaltet.

Virenscanner

Zur Erkennung von Viren benötigt AMaViS externe Virenscanner-Programme. Diese muss der Anwender selbst erwerben und installieren. Informationen zur Auswahl von Virenscannern finden sich in Kapitel 7, *Virenscanner*.

AMaViS verwaltet zwei Listen von Virenscannern: eine primäre und eine sekundäre. Bei jeder zu scannenden E-Mail wird zuerst versucht, alle Virenscanner der primären Liste auszuführen. Wenn einer der primären Virenscanner die E-Mail als infiziert oder sauber eingestuft hat, ist die Virenerkennung abgeschlossen. Ist keiner der primären Virenscanner verfügbar (entweder nicht installiert oder fehlerhaft), werden alle sekundären Virenscanner auf die gleiche Art ausgeführt. Der Sinn dieser Aufteilung ist, dass langsamere Virenscanner in die sekundäre Liste aufgenommen werden, damit sie nur im Ausnahmefall verwendet werden. Insbesondere werden einige Virenscanner-Produkte in zwei Varianten ausgeliefert: einer langsameren Kommandozeilen-Variante und einer schnelleren Server-Variante. In diesem Fall würde man die Servervariante in die primäre Liste eintragen und die Kommandozeilen-Variante in die sekundäre Liste. Nur wenn der Server ausfällt (und alle anderen primären Scanner ebenfalls), würde die Kommandozeilen-Variante verwendet werden.

Die Einbindung von Virenscannern in AMaViS ist nicht ganz einfach, da alle Virenscanner-Programme unterschiedliche Kommandozeilen-Optionen verwenden und die Scan-Ergebnisse durch Parsen der Ausgabe des Scan-Programms ermittelt werden müssen. Aus diesem Grund ist AMaViS schon für über 30 Virenscanner-Produkte vorkonfiguriert.

Beim Start des AMaViS-Servers wird überprüft, welche der konfigurierten Produkte tatsächlich installiert sind, und nur diese werden verwendet. Um zu überprüfen, ob der beabsichtigte Virenscanner tatsächlich gefunden wurde und verwendet wird, sollte man also zur Sicherheit den AMaViS-Log lesen.

Zum Konfigurieren seiner gewünschten Virenscanner sucht man daher am besten in der Konfigurationsdatei nach den Variablen `@av_scanners` (primäre Scanner) und `@av_scanners_backup` (sekundäre Scanner) und kommentiert die nicht gewünschten Einträge aus beziehungsweise entfernt die Kommentare vor den gewünschten Einträgen.

Um zu verhindern, dass bei einem erkannten Virus alle restlichen Virenscanner auch noch ausgeführt werden, kann man die Variable `$first_infected_stops_scan` auf wahr setzen. Dies beschleunigt zwar den Scan-Vorgang, verhindert aber eventuell einen vollständigen Bericht, wenn andere Virenscanner umfangreichere Virendatenbanken haben.

Lookup-Maps

Die bisher gezeigten Konfigurationseinstellungen von AMaViS sind alle global, das heißt, sie werden bei jeder von AMaViS verarbeiteten E-Mail angewendet. AMaViS bietet darüber hinaus die Möglichkeit, viele Konfigurationseinstellungen von der Empfängeradresse oder anderen Suchschlüsseln abhängig zu machen. Zur Steuerung dieser Konfiguration können verschiedene Datenquellen wie Hashes und Tabellen mit regulären Ausdrücken oder auch SQL- und LDAP-Datenbanken verwendet werden.

Eine komplette Abhandlung der Lookup-Funktionalität würde den Rahmen dieses Kapitels sprengen. Die Datei *README.lookups* in der AMaViS-Distribution enthält die vollständige Beschreibung und erläutert ebenfalls die Einbindung von SQL- und LDAP-Datenbanken.

Die vielfältigen Möglichkeiten dieses Features sollen hier mit einem Beispiel angedeutet werden. Den Schwellenwert, ab dem eine E-Mail als Spam eingestuft und eventuell verworfen wird, bestimmt im einfachen Fall wie zuvor beschrieben die Variable `$sa_kill_level_deflt`. Diese Variable hat einen numerischen Wert, zum Beispiel `$sa_kill_level_deflt = 6.0`. Die eigentliche Konfigurationsvariable für dieses Feature ist jedoch `@spam_kill_level_maps`. Dieses Array enthält eine Liste von Tabellen und anderen Quellen, so genannten Lookup-Maps, die anhand der Empfängeradresse durchsucht werden und eine numerische Antwort zurückgeben sollen. Hier ist eine etwas komplexere Konfiguration, die diese Möglichkeit ausnutzt:

```
our %some_hash = (
    'beispiel.de' => 6.5,
    'info@' => 3.0,
    'postmaster@mydomain.com' => 10
);

$sa_kill_level_deflt = 5.0;

@spam_kill_level_maps = (\%some_hash, \$sa_kill_level_deftl);
```

In diesem Fall sind zwei Lookup-Maps aufgeführt. Zur Bestimmung des tatsächlich zu verwendenden Werts werden die Lookup-Maps von links nach rechts durchprobiert, bis es eine Antwort gibt. Wenn ein Hash angegeben ist, wird in dem Hash mit verschiedenen Schreibweisen der E-Mail-Adresse nachgeschlagen. Wurde keine Antwort gefunden, wird die nächste Lookup-Map probiert. Im obigen Fall ist dies eine Referenz auf eine Konstante beziehungsweise einen skalaren Wert. Dieser Wert wird dann in jedem Fall als Antwort genommen. Konstanten sind also nur als letzter Eintrag in einer Lookup-Map-Liste sinnvoll. Es ist zu beachten, dass die Variable `$sa_kill_level_deflt` keine Sonderbehandlung erfährt: Wenn sie nicht in `@spam_kill_level_maps` gelistet ist, wird sie nicht verwendet. Die Voreinstellung ist allerdings:

```
@spam_kill_level_maps = (\$sa_kill_level_deflt);
```

wodurch `$sa_kill_level_deflt` die einzige relevante Einstellung ist.

Ein anderes Beispiel ist die Einstellung `@local_domains_maps`, die, wie oben beschrieben, bestimmt, welche Domains als lokal betrachtet werden sollen. Die Einstellung in der Beispielkonfigurationsdatei von AMaViS ist:

```
@local_domains_maps = ( [".$mydomain"] );
```

Der führende Punkt steht für die Domain und ihre Subdomains. Dies könnte man auch so schreiben:

```
our @some_array = (
    ".$mydomain"
);

@local_domains_maps = (\@some_array);
```

In diesem Fall ist die einzige angegebene Lookup-Map ein Array. Ein Array kann lediglich wahr oder falsch als Antwort zurückgeben, keine anderen Werte, wie dies beim Hash möglich ist. Ob logische, numerische oder andere Werte nötig sind, hängt selbstverständlich von der jeweiligen Konfigurationsoption ab.

Eine vollständige Auflistung aller Einstellungen, die über Lookup-Maps konfigurierbar sind, gibt es leider nicht. Die Beispielkonfigurationsdatei von AMaViS erwähnt allerdings die meisten. Die Namen der Variablen enden alle auf `_maps`.

Obwohl das Lookup-Map-Feature sehr große Flexibilität bei der Konfiguration von AMaViS ermöglicht, sollte man sich auch überlegen, welchen Administrationsaufwand man dadurch erzeugt. Es ist eine Sache, für einige besondere E-Mail-Adressen, zum Beispiel von Administratoren, Ausnahmen einzurichten. Es ist aber eine andere Sache, für jeden Benutzer die persönliche Wunschkonfiguration zu pflegen. Zurückhaltung ist also angesagt.

Policy-Banks

Eine Policy-Bank ist eine separate Menge von Konfigurationseinstellungen, die nur verwendet wird, wenn die bei AMaViS ankommende SMTP-Verbindung über einen bestimmten TCP-Port hereinkommt. Dadurch ist es möglich, für verschiedene Benutzergruppen unterschiedliche Konfigurationen (»Policys«) zu verwenden.

Es liegt in der Hand des Administrators, wie es arrangiert wird, dass die Verbindungen über verschiedene TCP-Ports eingehen. Entweder kommen die Verbindungen sowieso von separaten MTA auf verschiedenen Rechnern, wobei jeder Rechner einen Port zugewiesen bekommt, oder man weist durch Filter im MTA jeder Empfänger-Domain einen anderen Port zu (in Postfix zum Beispiel über Header-Checks).

Um verschiedene Policy-Banks zu verwenden, schaltet man zunächst die gewünschten Portnummern in AMaViS frei, zum Beispiel:

```
$inet_socket_port = [10024, 10026, 10028];
```

Man beachte, dass der MTA in der Regel schon den Port 10025 verwendet.

Dann kann man die eigentlichen Policy-Banks definieren, zum Beispiel:

```
$policy_bank{'SPECIAL'} = {
    log_level => 1,
    final_bad_header_destiny => D_BOUNCE,
    spam_kill_level_maps => 10.0,
};

$policy_bank{'TEST1'} = {
    log_level => 5,
    final_virus_destiny => D_PASS,
};
```

`SPECIAL` und `TEST1` sind hier willkürliche Namen. Die Namen der Hash-Schlüssel entsprechen den in der globalen Konfiguration verwendeten Variablennamen. Falls für eine Variable sowohl eine Lookup-Map als auch Scalars existieren, muss, wie hier bei `spam_kill_level_maps`, der Name der Lookup-Map verwendet werden. Das Policy-Bank-Feature ist noch relativ neu, und eine einheitliche Namensgebung wird sich erst noch herausarbeiten müssen.

Anschließend werden die definierten Policy-Banks den Portnummern zugewiesen:

```
$interface_policy{'10026'} = 'SPECIAL';
$interface_policy{'10028'} = 'TEST1';
```

Die tatsächlich zur Laufzeit verwendete Konfiguration ergibt sich dann folgendermaßen: Zuerst werden die nicht an eine Policy-Bank gebundenen, globalen Einstellungen ausgewertet. Dies ist die implizite Policy-Bank `$policy_bank{''}`. Danach wird, falls vorhanden, die Policy-Bank für die verwendete Portnummer ausgewertet, wodurch einige der globalen Einstellungen überschrieben werden. Eine Policy-Bank muss also nicht alle möglichen Einstellungen angeben, sondern nur die, die sich von der globalen Konfiguration unterscheiden. Wenn AMaViS die IP-Adresse des ursprünglichen SMTP-Clients kennt und diese in `@mynetworks` enthalten ist (also der Client vermutlich aus dem eigenen Netzwerk kommt), wird schließlich die Policy-Bank `$policy_bank{'MYNETS'}` ausgewertet, die wiederum einige Einstellungen überschreiben könnte. Dies funktioniert nur, wenn die XFORWARD-Erweiterung von Postfix verwendet wird, was in der oben beschriebenen Konfiguration für Postfix der Fall ist. In jedem Fall sollte man Tests durchführen, um sicherzustellen, dass die Policy-Bank `MYNETS` richtig verwendet wird. Eine denkbare Einstellung für `MYNETS` wäre zum Beispiel:

```
policy_bank{'MYNETS'} = {
    bypass_spam_checks_maps   => [1],
};
```

Damit würde die Spam-Prüfung für E-Mails von internen Absendern ausgeschaltet.

Man beachte den Unterschied zwischen diesem Feature und Lookup-Maps. Lookup-Maps unterscheiden die Konfiguration nach Empfängeradresse oder anderen Eigenschaften der zu verarbeitenden E-Mail; Policy-Banks unterscheiden die Konfiguration nach eingehendem TCP-Port. Eine Policy-Bank kann Konfigurationen von Lookup-Maps enthalten, aber nicht umgekehrt.

Statistiken mit Amavis-stats

Um das E-Mail-Aufkommen des eigenen Mailservers sowie die Leistung von AMaViS zu überwachen, kann man mit Hilfe des Programmpakets Amavis-stats ansehnliche Grafiken erstellen lassen.

Amavis-stats analysiert die Log-Dateien von AMaViS und erstellt Statistiken über die Anzahl der versendeten, geblockten und als Spam eingestuften E-Mails. Zusätzlich wird aufgeschlüsselt, welcher Virentyp wie oft gesehen wurde. Diese Statistiken können in Diagrammen mit verschiedener Zeitstaffelung grafisch dargestellt werden. Die Grafiken werden dynamisch durch ein PHP-Skript erzeugt und können mit einem Webbrowser angesehen werden. Abbildung 8-3 zeigt, wie eine monatliche Durchsatzgrafik aussieht, Abbildung 8-4 zeigt eine monatliche Zusammenfassung aller erfassten Virentypen.

Die letzte »stabile« Release von Amavis-stats, Version 0.1.12, erkennt keine Spam-E-Mails im AMaViS-Log. Wer Statistiken über Spam-E-Mails in den Diagrammen sehen möchte, wie in Abbildung Abbildung 8-3, sollte die »instabile« Version 0.1.13-rc6 verwenden, die in der Einschätzung der Buchautoren durchaus stabil läuft.

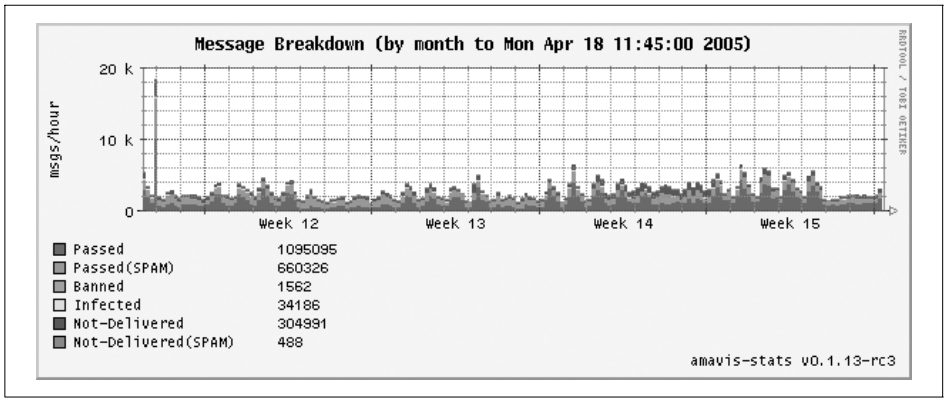

Abbildung 8-3: Grafische Darstellung des monatlichen Durchsatzes von AMaViS

Die internen Statistiken von Amavis-stats werden über einen regelmäßigen Cronjob aktualisiert. Somit kann man die Aktualität der Statistiken selbst beeinflussen. Voreingestellt sind Updates alle fünf Minuten.

Damit Amavis-stats die Log-Dateien von AMaViS analysieren kann, muss die Konfigurationsvariable `$log_level` mindestens auf 2 stehen, so dass der Log genügend Informationen darüber enthält, wie mit jeder E-Mail verfahren wurde.

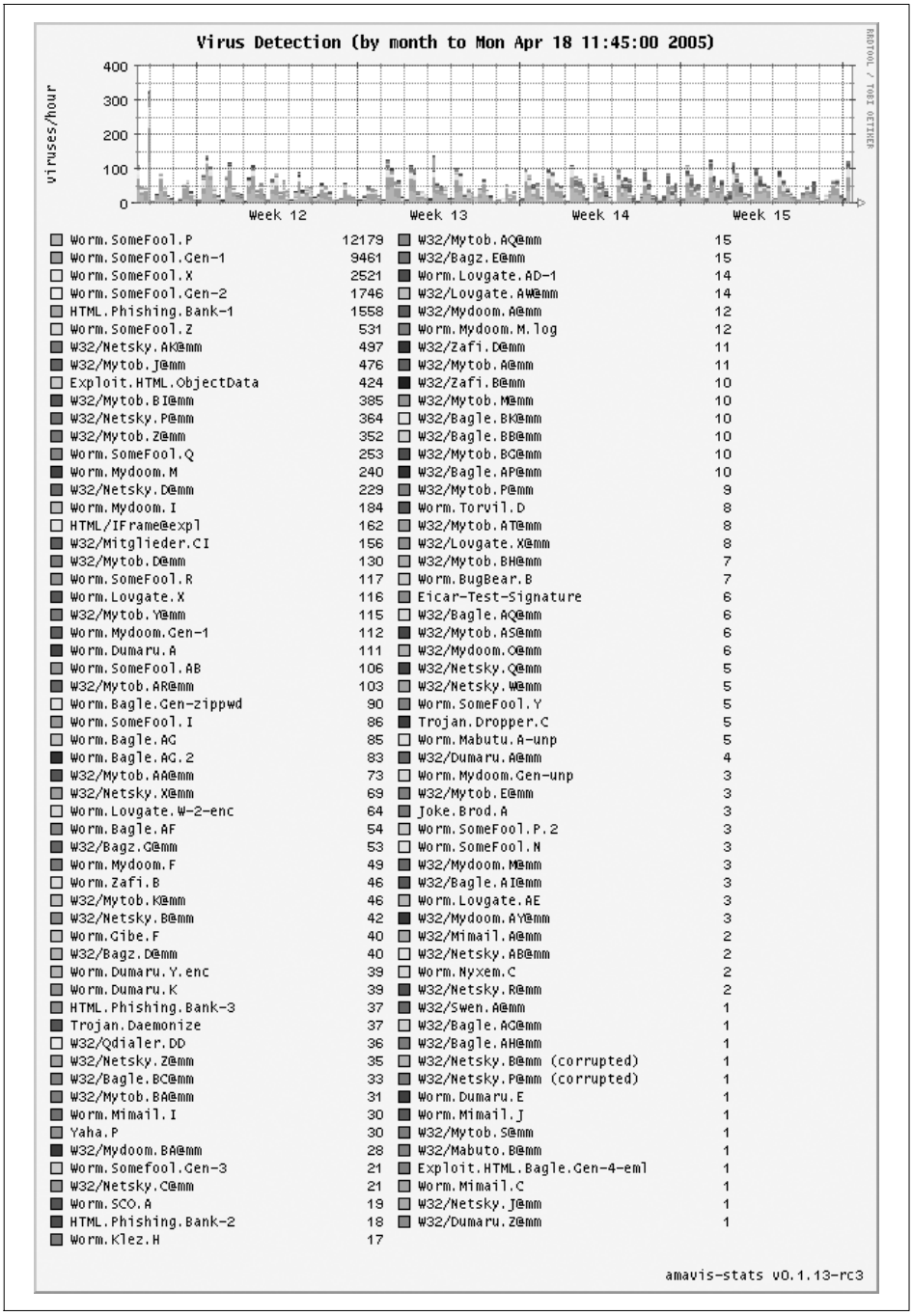

Abbildung 8-4: Grafische Darstellung der von AMaViS erfassten Viren in einem Monat