Whitepaper: Postfix/Cyrus und Spamassassin

Im Folgenden wird die Grund-Installation/Konfiguration eines SMTP/IMAP-Systems inklusive SPAM-Filter unter der Fedora Linux-Distribution beschrieben. Als Mail-Server verwenden wir ***Postfix***, als IMAP-Server den ***Cyrus-Impad*** und als SPAM-Filter ***Spamassassin***. Alle Produkte unterliegen einer GPL-angelehnten Lizenz und können damit kostenlos aus dem Internet heruntergeladen und verwendet werden.

Installation

Da die Software im RPM-Format vorliegt, können alle hier aufgeführten Pakete über yum- bzw. rpm-Zeilenkommandos installiert werden. Das Kompilieren von Source-Distributionen ist nicht Bestandteil dieser Dokumentation.

Postfix

Für die Postfix Installation wird folgendes Paket benötigt:

```
postfix-2.x
```

Mit der Eingabe von

```
yum search postfix
```

kann überprüft werden ob das Paket auf einem der konfigurierten yum-Mirrors vorhanden, oder bereits installiert ist. Normalerweise sollte Postfix Bestandteil einer jeden modernen Linux-Distribution sein. Sollte Postfix noch nicht installiert sein, so kann es mit folgenden Kommandos vom yum-Server bzw. lokal via rpm installiert werden:

```
yum install postfix-2.x
 oder
rpm -Uvh postfix-2.x
```

wobei das '.x' im Postfix-Paketnamen für die aktuelle Versionsnummer des Postfix-Servers steht.

Cyrus Impad

Für den Cyrus-Imapd Server werden folgende Pakte benötigt:

```
cyrus-sasl-md5-2.1.x
cyrus-sasl-2.1.x
cyrus-sasl-plain-2.1.x
cyrus-imapd-2.2.x
cyrus-imapd-utils-2.2.x
```

Die Pakete des Cyrus-Impad Servers können analog zu Postfix entweder mit *yum* oder *rpm* installiert werden.

Konfiguration

Die Konfiguration von Postfix und Cyrus-Imapd geschieht ausschließlich über das Editieren von

Config-Dateien. Bei der Installation über *rpm* bzw. *yum* werden diese Dateien bereits automatisch in den richtigen Verzeichnissen angelegt.

Postfix

Die zentralen Konfigurationsdateien für Postfix sind main.cf und master.cf. Beide Dateien befinden sich standardmäßig im Verzeichnis /etc/postfix.

Einstellungen in `main.cf`:

Grundkonfiguration

Die erste Zeile setzt die Variable *$myhostname* auf den FQDN des Mailservers. Mit diesem Namen meldet sich Postfix bei benachbarten SMTP-Servern um Mails auszutauschen. Hierbei ist wichtig, daß der Name im DNS aufgelöst wird und tatsächlich auf die IP-Adresse des Hosts verweist auf dem Postfix läuft. Das gleiche gilt für den Eintrag *$mydomain.*

/etc/postfix/main.cf

```
myhostname = smtp.yourdomain.com
mydomain = yourdomain.com
```

Mail Hub Konfiguration

Wenn Mail an lokale Benutzer zugestellt werden soll, muß Postfix mitgeteilt werden welche Domains als eigene zu betrachten sind (von Bedeutung ist hierbei der Teil der Email-Adresse nach dem @-Zeichen). Der *$mydestination* Parameter listet alle Domains auf, die der SMTP-Server als eigene beansprucht. In der Regel ist das der Wert von `$mydomain` und `$myhostname`. Es können jedoch weitere Domänen aufgelistet werden.

/etc/postfix/main.cf

```
mydestination = $mydomain, $myhostname, localhost.$mydomain
```

Wenn Postfix ein Mail erhält dessen Domänenbestandteil hier nicht aufgelistet ist, wird es den für diese Domäne zuständigen SMTP-Server kontaktieren und das Mail weiterleiten (vorausgesetzt die Konfiguration von Postfix erlaubt dies, siehe *Relay-Kontrolle*).

Satelite Client Konfiguration

Standardmäßig sollten Computer im Unternehmen keine Mails für die lokale Zustellung akzeptieren, sondern diese an einen zentralen Mail Hub weiterleiten.

/etc/postfix/main.cf

```
mydestination =
relayhost = smtp.example.com
```

Durch Leerlassen der `$mydestination`–Variable wird erreicht, daß keine Mail an lokale Benutzer zugestellt wird sondern an den Host, der als `$relayhost` (Mail Hub) konfiguriert ist. Wie man leicht erraten kann schließen sich *Satelite*- und *Hub*-Konfiguration gegenseitig aus. In unserem Fall wollen wir Mails lokal zustellen, deshalb wählen wir die *Mail Hub-Konfiguration*.

Mailstore Delivery

Standardmäßig wird Mail für lokale User nach `/var/spool/mail/username` geschrieben. Hierbei handelt es sich um die traditionelle Art und Weise wie Unix-Systeme ihre Mails lokal zustellen. In der Regel verfügen die meisten Anwender jedoch nicht über Clients, mit denen sie

auf diese Art von mailstore zugreifen können. Die meisten Anwender verwenden heutzutage GUI-basierende Mail-Clients und greifen auf ihre Mails über POP oder IMAP zu.

Aus diesem Grund verwenden wir für die Speicherung von Emails den Cyrus Imap-Daemon. Er unterstützt sowohl POP- als auch IMAP-Zugriffe und bringt viele andere nützliche Features mit, zum Beispiel SSL-Support und Sieve-Filterung.

Wie viele andere, moderne Storage Systeme benutzt Cyrus „LMTP“ (Local Mail Transport Protocol) für eingehende Email. Glücklicherweise spricht auch Postfix LMTP. Wenn Cyrus auf dem gleichen Computer wie Postfix installiert ist, kann folgender Filesystem Socket zur lokalen Zustellung benutzt werden:

/etc/postfix/main.cf
```
mailbox_transport = lmtp:unix:/var/lib/imap/socket/lmtp
```

Wenn Cyrus auf einem anderen Computer installiert ist, kann anstelle dessen ein Netzwerk Socket benutzt werden:

/etc/postfix/main.cf
```
mailbox_transport = lmtp:inet:imap.example.com
```

Da in unserem Beispiel beide Server (SMTP u. IMAP) auf der gleichen Maschine laufen, verwenden wir den Filesystem Socket zur Mailzustellung (lmtp:unix).

Alias Kontrolle

Aliases sind alternative Name für lokale User. Wenn Postfix ein Mail an die Adresse *an-alias@example.com* erhält, prüft es seine Alias-Tabelle ob es sich hierbei einen Alias-Eintrag handelt und stellt dann die Mail an den entsprechenden User zu.

Folgende Aliases sind traditionell und nach RFC auf einem System vorhanden:

Alias	Rolle
sysadmin	System Administrator
postmaster	Mail Administrator
webmaster	Website Administrator
abuse	Adresse um Spammer und Hacker zu reporten

Postfix speichert diese Zuordnung in der Datei `/etc/aliases`. Alternativ können diese Einträge auch in einem LDAP-Verzeichnis gespeichert werden.

/etc/postfix/main.cf
```
#alias_maps = hash:/etc/aliases, ldap:ldapdata
alias_maps = hash:/etc/aliases
```

Da Postfix nicht direkt auf `/etc/aliases` zugreift sondern zur Beschleunigung einen Hash dieser Datei erstellt, muß nach Änderungen in `/etc/aliases` der Befehl:

```
# postalias /etc/aliases
```

an der Kommandozeile ausgeführt werden.

Relay Kontrolle

Wenn Postfix ein Email erhält, vergleicht es den Domänenteil der Adresse in RCPT mit *$mydestination*, um zu entscheiden ob das Email lokal zugestellt, oder an einen Remote-host weitergeleitet werden soll.

In den frühen, unbeschwerten Tagen des Internets war es eine Selbstverständlichkeit, daß man jede Email weiterleitet, die nicht für die eigene Domäne bestimmt war. Solche SMTP-Server nennt man Open-Relay. Heutzutage wimmelt das Internet nur so von Bandbreitendieben und Spammern, die offene relays dazu benutzen ihre fragwürdigen Botschaften zu verbreiten.

Ein offenes relay zu betreiben ist heutzutage nicht nur unverantwortlich, sondern befördert das eigene Unternehmen sehr schnell auf eine Spam-Host Blacklist.
Postfix kann so konfiguriert werden, daß nur autorisierte Mail weitergeleitet wird. Die Weiterleitung wird von der *smtp_recipient_restrictions*-Direktive gesteuert, die als Eingabe einen String von kombinierten Restriktionen auswertet bis es auf das Schlüsselwort REJECT od PERMIT stößt.

IP-based Relay Control
Die einfachste Form der Relay-Kontrolle stellt das IP-basierte Verfahren dar. Es wird nur bestimmten IP-Adressen bzw. Adress-Ranges das Relaying erlaubt. Ausgewertet wird dabei die Source Adressen des senden SMTP-Relays.

/etc/postfix/main.cf
```
mynetworks = 127.0.0.0/8 , 192.168.0.0/24
relay_domains =
smtpd_recipient_restrictions = permit_mynetworks,
                               reject_unauth_destination,
                               permit
```

Die *permit_mynetworks*-Restriktion liefert „PERMIT“ zurück falls sich die IP-Adresse der kontaktierenden Maschine mit *$mynetworks* deckt. Die Regel *reject_unauth_destination* liefert „REJECT“ zurück, falls das Ziel nicht die lokale Maschine oder ein Eintrag aus *$relay_domains* ist.

Auth-based Relay Control

Eine Alternative zur IP-basierten Relay-Kontrolle stellt die authentifizierende Relay-Kontrolle dar. Bevor Postfix ein Email weiterleitet erwartet es einen Usernamen und ein Passwort. Das Email wird nur dann versendet, wenn der User die Berechtigung dazu hat und wenn das zugehörige Passwort korrekt war.

Durch Verwendung von **SASL** (**Simple Authentication and Security Layer)** wird vor der Weiterleitung Username und Passwort abgefragt. Die hierzu benötigten Softwarepakete haben wir bereits bei der Cyrus-Installation eingespielt (*cyrus-sasl-md5-2.1.x*, *cyrus-sasl-2.1.x*, und *cyrus-sasl-plain-2.1.x,* siehe weiter oben).

Der erste Schritt besteht darin, eine *permit_sasl_authenticated*-Regel in die *smtp_recipient_rstrictions* Liste einzufügen.

/etc/postfix/main.cf
```
mynetworks = 127.0.0.0/8 , 192.168.0.0/24
relay_domains =
smtpd_recipient_restrictions = permit_mynetworks,
                               permit_sasl_authenticated,
                               reject_unauth_destination,
                               permit
```

Dadurch wird für authentifizierte Benutzer ein „PERMIT“ zurückgeliefert. Als nächstes muß Postfix

so konfiguriert werden, daß es Authentifizierung unterstützt.

Zunächst sollte man aber dafür sorgen, daß Postfix verschlüsselte Verbindungen zwischen den Partnern aufbauen kann. Wer überträgt schon gerne sein Passwort in Klarschrift über das Internet? Damit Postfix auch SSL unterstützt müssen wir zunächst ein Zertifikat und einen Serverschlüssel für unseren Mail-Server erstellen.
OpenSSL stellt zu diesem Zweck Commandline-Utilities zur Verfügung.

```
openssl genrsa -des3 -out pass.key 1024
openssl rsa -in pass.key -out yourdomain.com.key
openssl req -new -key pass.key -x509 -days 1500 > yourdomain.com.crt
openssl x509 -text < yourdomain.com.crt
```

In der ersten Zeile wird ein 1024-bit RSA-Key mit dem Namen *pass.key* erstellt. Bei der Erstellung des Schlüssels werden wir nach einer Passphrase gefragt.

Da wir diese mit Postfix nicht übergeben, löschen wir sie mit dem Kommando in Zeile zwei wieder und speichern den Schlüssel (ohne Passprase) unter dem Namen *yourdomain.com.key* ab.

Mit dem Kommando in Zeile 3 wird ein neues x509 Zertifikat erstellt. Dieses Zertifikat unterzeichnen wir mit unserem eben erstellten Privatekey (pass.key) und speichern es unter *yourdomain.com.crt* ab.

Das Kommando in der vierten Zeile gibt den Inhalt des Zertifikates aus. Damit können wir unsere Eingaben überprüfen.

Als nächstes müssen wir Postfix mitteilen wo diese beiden Dateien im Filesystem zu finden sind:

/etc/postfix/main.cf
```
smtpd_use_tls = yes
smtpd_tls_cert_file = /etc/postfix/yourdomain.com.crt
smtpd_tls_key_file = /etc/postfix/yourdomain.com.key
```

Mit der Direktive *smtp_use_tls* kann dann der TLS-Encryption-Layer (Transport Layer Security) aktiviert werden, bevor die Authentifizierung erfolgt.

Zu diesem Zeitpunkt weiß Postfix, daß es das SMTP AUTH Kommando unterstützen soll. Was es noch nicht weiß, ist wie es den Usernamen und das Passwort, welche es von den Clients bekommt überprüfen soll. Für eine funktionierende Authentifizierung muß die Methode angegeben werden, wie die SASL-Library mit Unsername und Passwort umgehen soll. Die bekannteste ist hierbei PAM.

Zu diesem Zweck müssen wir Postfix mitteilen, daß es den *saslauthd* (der SASL Authentication Daemon) zur Prüfung der Passwörter verwenden soll

/etc/postfix/main.cf
```
smtpd_sasl_auth_enable = yes
smtpd_tls_auth_only = yes
```

... und dem *saslauthd* daß er dazu das PAM verwenden soll. Hierzu werden in den Dateien */usr/lib/sasl/smtpd.conf, /etc/sysconfig/saslauthd* und */etc/pam/smtp* folgende Einträge gesetzt.

/etc/usr/lib/sasl/smtp.conf
```
pwcheck_method: saslauthd
mech_list: plain login
```

/etc/sysconfig/saslauthd

```
MECH=pam
```

/etc/pam/smtp

```
auth       required     pam_stack.so service=system-auth
account    required     pam_stack.so service=system-auth
```

Der *saslauthd* kann mit folgendem Kommando gestartet werden. Es wird empfohlen diesen Dienst, wie auch alle anderen beteiligten Dienste über die runlevel-Steuerung automatisch beim Systemboot zu starten.

```
service saslauthd start
```

Starten des Postfix Servers:

```
service postfix start
```

Die Grundkonfiguration unseres Postfix-Servers ist damit abgeschlossen.

Cyrus-Imapd

Über die Datei */etc/imapd.conf* wird die Grundkonfiguration des IMAP-Servers vorgenommen. *sievedir* und *sendmail* sind auskommentiert da wir statt Sendmail ja Postfix einsetzen wollen. Auf sieve wollen wir hier nicht weiter eingehen.

/etc/imapd.conf

```
configdirectory: /var/lib/imap
partition-default: /var/spool/imap
admins: cyrus root
# sievedir: /var/lib/imap/sieve
# sendmail: /usr/sbin/sendmail
hashimapspool: true
sasl_pwcheck_method: saslauthd
sasl_mech_list: PLAIN
```

Damit die User-Credentials und die Mail-Inhalte nicht im Klartext über das Netzwerk gelangen sollen müssen wir dem IMAP-Server mitteilen, wo im Filesystem das Server-Keyfile und das Serverzertifikat zu finden sind. Der Einfachheit halber verwenden wir hier den gleichen Schlüssel und das gleiche Zertifikat wie bei der Einrichtung des Postfix-Servers. Normalerweise sollten man für jeden Server einen eigenen Schlüssel generieren.

/etc/imapd.conf

```
tls_cert_file: /etc/postfix/yourdomain.com.crt
tls_key_file: /etc/postfix/yourdomain.com.key
tls_ca_file: /etc/postfix/yourdomain.com.crt
```

Starten des Cyrus Servers:

```
service cyrus-imapd start
```

Die Grundkonfiguration unseres Cyrus-Servers ist damit abgeschlossen.

Verwendung von cyradm zur Mailbox-Administration

Bei *cyradm* handelt es sich um ein einfaches Kommandozeilen-Tool das zur Administration der User-Mailboxen verwendet werden kann. Das Tool sollte ausschließlich vom *cyrus* Unix-Account aus gestartet werden. Nach dem Einloggen kann man sich mit dem Befehl *help* eine Liste des zur Verfügung stehenden Befehlumfangs und eine kurze Erklärung des jeweiligen Befehls anzeigen lassen.

Das folgende Beispiel soll veranschaulichen wie sich mit dem *cyradm*-Tool die grundlegende Userverwaltung durchführen läßt. Im Beispiel wird gezeigt wie man eine neue Mailbox anlegt und sich die ACL anzeigen läßt. Anschließend wird zur ACL ein User mit Löschrechten hinzugefügt um die Mailbox hinterher wieder zu löschen.

```
# su cyrus
$ cyradm -u cyrus localhost
localhost.localdomain> createmailbox user.test
localhost.localdomain> listmailbox
user.test (\HasNoChildren)
localhost.localdomain> listaclmailbox user.test
test lrswipcda
localhost.localdomain> setaclmailbox user.test cyrus c
test lrswipcda
cyrus c
localhost.localdomain> deletemailbox user.test
localhost.localdomain> listmailbox
localhost.localdomain>
localhost.localdomain> quit
$
```

Test der Grundfunktionalität

Zum Testen der Funktionalität, legt man am besten einen neuen Unix-User an und versucht sich über telnet von einem anderen Computer aus auf dem SMTP- und dem IMAP-Server einzuloggen.

Postfix testen

```
~/> telnet smtp.example.de smtp
Trying 211.130.123.101...
Connected to smtp.example.de.
Escape character is '^]'.
220 yourdomain.com ESMTP Postfix
helo smtp.somewhere.de
250 smtp.example.de
mail from: galder@discworld.com
250 Ok
```

```
rcpt to: someone@example.de
250 Ok
data
354 End data with <CR><LF>.<CR><LF>
blablabla
blablabla
.
250 Ok: queued as E4BA080009B
quit
221 Bye
Connection closed by foreign host.
```

Cyrus testen

```
~/> telnet imap.example.de imap
Trying 211.130.123.101...
Connected to imap.example.de.
Escape character is '^]'.
* OK imap.example.de Cyrus IMAP4 v2.2.10-Invoca-RPM-2.2.10-3.fc2 server
ready
. capability
* CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ MAILBOX-REFERRALS
NAMESPACE UIDPLUS ID NO_ATOMIC_RENAME UNSELECT CHILDREN MULTIAPPEND BINARY
SORT THREAD=ORDEREDSUBJECT THREAD=REFERENCES ANNOTATEMORE IDLE STARTTLS
LISTEXT LIST-SUBSCRIBED X-NETSCAPE
. OK Completed
. login username password
. OK User logged in
. logout
* BYE LOGOUT received
. OK Completed
Connection closed by foreign host.
```

Spamassassin

Bei Spamassassin handelt es sich um einen eMailfilter der in der Lage ist sogenannte Junk-Mails zu identifizieren. Dabei wird der komplette Inhalt einer Mailnachricht nach bestimmten Phrasen und Indikatoren untersucht, die in der Regel bei den meisten SPAMs zu finden sind.

Der Ansatz von Spamassassin ist jedoch wesentlich komplexer als jener der gängigen Anti-Viren-Scanner, die im Grunde nur nach bestimmten Schlüsselwörtern innerhalb des Mailheaders/body scannen. Spamassassin verwendet ein sogenanntes Scoring-System: Mail wird nur dann als SPAM klassifiziert, wenn es in Summe ausreichend viel SPAM-Charakteristiken aufweist. In Kombination mit anderen Features, die zum Teil als Plugins nachrüstbar sind, erzielt SA damit nur sehr wenige False Positives (weniger als 1%) und erkennt zwischen 90% und 95% korrekt als SPAM.

SA blockiert keine SPAM. Stattdessen versieht es Subject und Mailheader von vermeintlichen SPAM-Messages mit einem Hinweis. Das ist auch gut so, da kein automatisiertes System in der Lage ist SPAM 100%ig zu identifizieren. Alle automatischen Spamfilter produzieren sowohl False Positives (eMail, die fälschlicherweise als SPAM klassifiziert wurde) und False Negatives (SPAM, das nicht als solches erkannt wurde). SA identifiziert mögliche SPAM-eMails, läßt aber die Entscheidung darüber, wie mit dem Mail zu verfahren ist dem Anwender. Über entsprechende

Filterregeln im eMail-Client kann dann SPAM zuerst aussortiert und später gelöscht werden.

Installation

Vor der eigentlichen Installation von Spamassassin müssen folgende Perl-Module auf dem System vorhanden sein:

- MIME::Base64
- MIME::QuotedPrint
- Net::DNS
- DB_File

Das Perlscript (getModules.pl) kann dazu verwendet werden um herauszufinden welche Module bereits auf dem Rechner installiert sind. Beim Aufruf sollte die Ausgabe in eine Datei mit der Endung .html umgeleitet werden. Diese Datei kann dann über einen Webbrowser betrachtet werden (z.B.: `./getModules.pl > perlmodules.html`).

getModules.pl

```
#!/usr/bin/perl

use strict;
use File::Find;

my $count = 0;

my (@mod, %done, $dir);
find(\&get, grep { -r and -d } @INC);
@mod = grep(!$done{$_}++, @mod);
foreach $dir (sort { length $b <=> length $a } @INC) {
   foreach (@mod) { next if s,^\Q$dir,,; }
}
print "Content-type: text/html\n\n";

print "<HTML><HEAD><TITLE>Installed Perl Modules</TITLE></HEAD><BODY
bgcolor='#c0c0c0'>";
print "<h3 align='center'>Perl Modules Installed</h3>";

# list table heading
 print "<table border=1 bgcolor='#999999' align='center'>";
     print "<tr><th>Perl Module Number</th><th>Perl Module
Name</th></tr>\n";

foreach (@mod) { s,^/(.*)\.pm$,$1,; s,/,::,g;

print "<tr align='center'>";
print "<td>$count</td>";
print "<td>$_</td>";
print "</tr>\n";

$count++;
}

print "</table>";
print "Total : ($#mod modules!)\n\n";
sub get { /^.*\.pm$/ && /$ARGV[0]/i && push @mod, $File::Find::name; }
```

Alle Module können von CPAN (http://search.cpan.org/) kostenfrei heruntergeladen werden. Nach dem Download müssen wir die Perl-Module entpacken, kompilieren und installieren. Wie das genau geht kann man im README des jeweiligen Moduls nachlesen.

Einfacher geht es jedoch, wenn man CPAN für Download, Kompilierung und Installation verwendet. Voraussetzung hierfür ist, daß man das CPAN-Modul bei der Perl-Installation mitinstalliert hat.

```
perl -MCPAN -e shell
       o conf prerequisites_policy ask
       install MIME::Base64
       install MIME::QuotedPrint
       install Net::DNS
       install DB_File
       quit
```

Wenn alle Voraussetzungen erfüllt sind, können wir mit der eigentlichen Installation von Spamassassin beginnen. Unter Fedora liegt Spamassassin als RPM-Paket vor. Deshalb können wir für die Installation entweder yum oder die rpm-Zeilenkommandos verwenden:

```
yum install spamassassin
  oder
rpm -Uvh spamassassin-3.x.fc4
```

Da es sich bei Spamassassin um ein Perl-Modul handelt können wir alternativ auch CPAN für den Installationprozess verwenden:

```
perl -MCPAN -e 'install Mail::SpamAssassin'
```

bzw. SA von CPAN herunterladen und anschließend kompilieren und installieren.

Eine weitere Möglichkeit Perl-Module zu installieren ist der Download des Perl-Paketes im .gz Format von der Seite "http://search.cpan.org" und die anschließende Kompilierung der Sourcen mit dem gcc-Compiler:

```
tar -xzf Net-DNS-0.54.tar.gz
cd Net-DNS-0.54
perl Makefile.PL
make
make test
make install
```

Zur Verarbeitung der eingehenden Mails mit Spamassassin benötigen wir ein kleines Skript, das die Mails über eine Pipe von Postfix zu Spamassassin und anschließend an den lokalen Mailerdaemon (z.B. sendmail) weiterleitet. Als root-User erstellen wir dazu im Verzeichnis /usr/local/bin das folgende kleine Shell-Skript:

spamfilter

```
#!/bin/bash
/usr/local/bin/spamc | /usr/sbin/sendmail -i "$@"
exit $?
```

Anschließend ändern wir die Zugriffrechte der Datei folgendermaßen:

```
chmod 755 /usr/local/bin/spamfilter
```

Für die Ausführung des Skripts erstellen wir einen dedizierten User:

```
groupadd spamfilter
useradd -g spamfilter -c "Spamfilter User" -d /home/spam -m spamfilter
```

Als nächstes setzen wir als Besitzer von "/usr/local/bin/spamfilter" den eben erstellten User:

```
chown spamfilter /usr/local/bin/spamfilter
```

In der Datei "/etc/postfix/master.cf" ändern wir in der "Services"-Section die Zeile beginnend mit "smtp" folgendermaßen:

master.cf

```
#-----------------------------------------------------------------
smtp inet n - n - - smtpd
  -o content_filter=spamfilter:
#-----------------------------------------------------------------
```

Achtung: In der Datei existieren zwei Zeilen, die mit "smtp" beginnen. Wir ändern jene, die das Schlüsselwort "inet" enthält.

Als nächstes fügen wir noch folgende Zeile in der Datei "/etc/postfix/master.cf" unter der "Postfix to non-Postfix software"-Section hinzu:

master.cf

```
#-----------------------------------------------------------------
spamfilter unix - n n - - pipe
  flags=Rq user=spamfilter argv=/usr/local/bin/spamfilter -f \
${sender} -- ${recipient}
#-----------------------------------------------------------------
```

Die Grundkonfiguration vom Spamassassin ist damit abgeschlossen. Zum Testen sollte sowohl Postfix als auch Spamassassin neu gestartet werden:

```
service postfix restart
service spamassassin restart
```

Praktische Tips und Tricks zu Spamassassin

Upgrade von Spamassassin von Version 2.x auf 3.x
Nach Upgrade von 2.x auf 3.x funktionieren die Bayes-Checks nicht mehr. Um das korrekte Verhalten wieder herzustellen, muss an der Kommandozeile folgender Befehl eingegeben werden:

```
sa-learn --sync
```

Test der Grundfunktionalität von Spamassassin:
Um die korrekte Funktionsweise von Spamassassin zu testen, wechseln wir zunächst in den

Account des Spamassassin-Users:

```
su - spamfilter
```

Als nächstes legen wir im Homedirectory des SA-Users eine Textdatei (testmail.txt) mit folgendem Inhalt an:

testmail.txt

```
From: <me@anydomain.com>
To:   <you@mydomain.com>

Hallo!
Dies ist eine Testnachricht.
```

Mit dem Befehl

```
spamassassin -t -D < testmail.txt
```

wird der Inhalt der Textdatei zur Prüfung an Spamassassin übergeben. Durch den Parameter "-t" startet Spamassassin im Testmode, der Parameter -D veranlasst Spamassassin Debug-Informationen auszugeben. Wenn SA korrekt installiert wurde, sollte Spamassassin mit der Ausgabe der durchgeführten Tests und einer zusammenfassenden Bewertung antworten.

Mit diesem Befehl kann die SA-Konfig-Hilfe angezeigt werden.

```
perldoc Mail::SpamAssassin::Conf
```

Mit diesem Befehl kann die Konfiguration von SA neu eingelesen werden:

```
 service spamassassin restart
```

Die Konfigurationsdateien von Spamassassin liegen in den Verzeichnissen "/usr/share/spamassassin" (Systemkonfiguration!!! sollte vom Anwender nicht geändert werden, bietet aber wichtige Hinweise auf Syntax der Konfiguration von Spamassassin) und "/etc/mail/spamassassin" (Userkonfiguration, hier können User individuelle Anpassungen machen).