Postfix & Dovecot Mailserver unter Debian GNU/Linux 5.0 "Lenny"

Einleitung

Benötigt man einen eigenen Mailserver, sollte man zunächst ein genaues und durchdachtes Konzept für diesen erstellen. Da dies jedoch einfach gesagt, aber relativ komplex umzusetzen ist, möchte ich mit diesem Howto einen guten Einstiegspunkt bieten, um seine eigene E-Mail Server-Infrastruktur aufzubauen.

Dabei verwende ich Postfix als MTA und Dovecot als IMAP/POP3 Server. Der von Wietse Venema 1998 entwickelte MTA Postfix bietet eine breite Fülle an Funktionen und ist zudem auf Sicherheit getrimmt. So unterstützt Postfix von sich aus die sichere Authentifizierung über die Cyrus SASL bzw. Dovecot SASL Module und Bibliotheken (Simple Authentication and Security Layer) sowie verschlüsselte Verbindungen via TLS.

Als Basissystem kommt, wie so oft, Debian GNU/Linux 5.0 “Lenny” zum Einsatz. In meinem Fall betreibe ich den Mailserver auf einem in einem Rechenzentrum untergestellten Server (Housing). Dies beduetet insbesondere, dass der Mailserver direkt an das Internet angebunden ist und damit solide gegen den Missbrauch durch Dritte geschützt werden muss. Daher widme ich mich in diesem Howto verstärkt auch den Sicherheitsaspekten zu.

Verwendetes Netzwerkszenario

Um das Howto nachvollziehbar zu machen, verwende ich hier durchgehend das folgende Netzwerk-Setup:

- FQHN: mail.example.com
- Mailserver ist zuständig für die Domains: mydomain.com und mydomain.de

Installation benötigter Softwarekomponenten

In diesem Howto gehe ich von einer Minimalinstallation von Debian GNU/Linux 5.0 “Lenny” aus. Daher installieren wir zunächst einige Softwarepakete, die wir später benötigen werden:

```
# aptitude install less unzip
```

MTA Postfix

Wie bereits erwähnt, verwenden wir Postfix als MTA, da bei dessen Entwicklung ein besonderer Wert auf die Sicherheit des Dienstes gelegt wird. Zudem ist Postfix modular aufgebaut und ermöglicht damit die Realisierung einer Vielzahl von Einsatzszenarien. Eine wichtige Voraussetzung für den sicheren Betrieb des MTA’s ist jedoch dessen korrekte und lückenlose Konfiguration. Daher möchte ich das folgende Kapitel allein der Installation und Konfiguration des MTA’s widmen.

Installation von Postfix

Zunächst einmal installieren wir den MTA Postfix mittels:

```
# aptitude install postfix
```

Der Debian Installer fragt uns dabei nach dem Einsatzzweck unseres Mailservers. Da wir einen vollwertigen Mail-Server aufbauen und betreiben wollen, der direkt an das Internet angeschlossen ist und Nachrichten direkt via SMTP mit anderen MTAs austauscht (sowohl ein- als auch ausliefernd), wählen wir an dieser Stelle den Menüpunkt Internet-site aus. Anschließend fragt uns der Installer

nach einem Domainnamen, der bei unvollständigen E-Mail-Adressen diesen angehangen wird. In unserem Fall geben wir hier den FQHN mail.example.com an (s. Netzwerkszenario).

Konfiguration von Postfix

Anschließend konfigurieren wir den MTA Postfix, indem wir dessen Hauptkonfigurationsdatei /etc/postfix/main.cf anpassen. Da es sehr wichtig ist, die einzelnen Optionen der Konfigurationsdatei zu verstehen, möchte ich diese Direktiven sukzessive erläutern.

Allgemeine Einstellungen

Bei jeder Verbindung zum SMTP-Server erhält der Client einige Informationen über diesen in Form eines sogenannten Banners. Standardmäßg gibt der MTA einige Informationen preis, die auf die verwendete MTA-Software und Versionsnummer hindeuten. Da ich stets aus Sicherheitsgründen empfehle, so wenig Informationen, wie nur möglich, über das verwendete System offenzulegen, lassen wir lediglich den Hostnamen und das verwendete Protokoll ESMTP (Extended SMTP) ausgeben:

```
smtpd_banner = $myhostname ESMTP
```

Auf jedem UNIX-System existiert ein Tool namens biff, das dazu verwendet werden kann, um die lokalen Benutzer auf dem Mailserver über neu eingegangene E-Mails zu informieren. Da wir keine lokalen Benutzer verwenden werden (d.h. die E-Mail Nutzer besitzen keinen Shell-Account auf dem Mailserver), deaktivieren wir dieses:

```
biff = no
```

Nun weisen wir Postfix an, E-Mail Adressen, die nicht vollständig sind, nicht zu vervollständigen. Viele Spammer verwenden unvollständige E-Mail-Adressen. Würde man die Postfix-Korrektur aktivieren, würden diese Spam-Mails nach Außen auf einmal so erscheinen, als ob sie von unserem Server verschickt worden sind. Wir verschicken auch keine Systemmails, bei den bspw. eine E-Mail Absenderadresse wie rootlocalhost@ korrigiert werden müsste. Stattdessen verwenden wir diesen Server als reines E-Mail Gateway und verwenden von Anfang an E-Mail Adressen, die RFC2822-konform sind.

```
append_dot_mydomain = no
```

Anschließend fügen wir noch einige allgemeine Konfigurationsoptionen der Konfigurationsdatei hinzu:

```
recipient_delimiter = +
readme_directory = no
```

Server-/Domaineinstellungen

Jeder Mailserver sollte über einen eindeutigen Hostnamen (FQHN) verfügen. In diesem Howto verwende ich dafür den FQHN mail.example.com (s. Netzwerkszenario). Um den Server später per DNS in unsere Domains mydomain.com und mydomain.de integrieren zu können (z.B. Setzen der MX-Records), verwenden wir den kostenlosen DNS-Dienst EveryDNS:

```
myhostname = mail.example.com
mydomain = mail.example.com
```

Sollten die Benutzer des E-Mail Systems eine E-Mail senden oder empfangen, die keinen Domainnamen in den Envelope- bzw. Header-Adressen enthält, wird der hier angegebene String als Domainname angehängt (in unserem Fall also der uns zugewiesene FQHN mail.example.com):

```
myorigin = $mydomain
```

Unser Mailserver nimmt SMTP-Anfragen auf allen Interfaces an:

```
inet_interfaces = all
```

Die folgenden Domains bzw. Hostnamen werden als Primärziele definiert, für die unser Mailserver zuständig ist. Enthält eine eingehende E-Mail einen dieser Strings hinter dem at-Zeichen in der E-Mail-Adresse des Empfängers, wird diese Mail nicht nach Außen weitergereicht, sondern direkt auf dem Mailserver dem jeweiligen Benutzerpostfach hinzugefügt (lokale Auslieferung der E-Mail durch das Programm local). An dieser Stelle konfigurieren wir Postfix so, dass dieser nur Mails für localhost und mail.example.com annimmt. Später werden wir die Zuständigkeit unseres Mailservers um weitere Domains erweitern und dabei virtuelle Domains verwenden:

```
mydestination = localhost, $mydomain
```

Die folgende Direktive definiert die Netzwerke, die als lokale Netzwerke behandelt werden. Clients, die sich innerhalb dieser Netzwerke befinden, werden als vertrauenswürdig eingestuft und können direkt via SMTP E-Mails versenden, ohne, dass sich diese gegenüber dem MTA authentifizieren müssten:

```
mynetworks = 127.0.0.0/8
```

In unserem Fall vertrauen wir allen Anwendungen, die von localhost aus solche Mails versenden wollen. So können wir bspw. Statusmeldungen und Systemmails in unser E-Mail System integrieren. Alle anderen Rechner, die über unseren MTA Mails versenden wollen, müssen sich zunächst bei diesem authentifizieren und gelten somit zunächst als nicht vertrauenswürdig.

Virtuelle Mailboxen

Da wir eine Vielzahl von E-Mail Accounts verwalten werden und einen auf Sicherheit optimierten Mailserver anstreben, schließen wir das Szenario aus, dass jeder E-Mail Benutzer einen entsprechenden Shell-Account besitzt und dessen E-Mails in seinem Home-Directory gespeichert werden. Stattdessen benutzen wir in unserem Fall virtuelle Benutzer mit virtuellen Mailboxen. Alle E-Mails werden dabei von einem einzigen Systembenutzer verwaltet. Um virtuelle Mailboxen verwenden zu können, fügen wir zunächst die folgenden Optionen der obigen Konfigurationsdatei hinzu:

```
virtual_mailbox_domains = /etc/postfix/virtual_domains
virtual_mailbox_base = /var/mail/vhosts
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
virtual_alias_maps = hash:/etc/postfix/virtual_alias
virtual_minimum_uid = 100
virtual_uid_maps = static:5000
virtual_gid_maps = static:5000
virtual_transport = dovecot
dovecot_destination_recipient_limit = 1
mailbox_size_limit = 0
```

Erläuterungen:

- virtual_mailbox_domains: legt die Datei fest, in der alle zu verwaltenden Domains zeilenweise definiert werden (s.u.).
- virtual_mailbox_base: legt das Verzeichnis fest, in dem sich die virtuellen Mailboxen physikalisch auf der Festplatte befinden.
- virtual_mailbox_maps: legt die Mapping-Datei fest, in der alle Zuordnungen von E-Mail-Adresse zur physikalischen Mailboxdatei zeilenweise definiert sind (s.u.). Die in der Hash-Datei aufgeführten Dateinamen werden an das in virtual_mailbox_base definierte Verzeichnis angehangen und somit die absoluten Dateinamen der Mailboxen gebildet.
- virtual_alias_maps: legt die Mapping-Datei fest, in der alle Aliase für alle virtuellen Domains zeilenweise definiert werden (s.u.).
- virtual_uid_maps und virtual_gid_maps legen die UID bzw. GID fest mit der der Delivery Agent virtual für die Zustellung der E-Mail in ein Postfach ausgeführt wird. Die Dateien für die Mailboxen werden unter der hier angegebenen UID/GID als Eigentümer- bzw. Gruppenzugehörigkeit geführt. In unserem Fall werden alle Mailboxen von einem einzelnen Benutzer mit der UID/GID 5000 verwaltet (s. weiter unten).

Anmerkung: wird die E-Mail-Infrastruktur für mehrere Hundert Domains oder mehr verwendet, sollte man an dieser Stelle nicht mit Mapping-Dateien, wie hier beschrieben arbeiten, sondern eine entsprechende Datenbank (bspw. PostgreSQL) oder einen Verzeichnisdienst, wie OpenLDAP für das Backend verwenden.

Wir belassen die obige Konfiguration zunächst wie hier erläutert und werden später die einzelnen Mapping-Dateien erstellen.

Wie bereits oben beschrieben, benötigen wir für die Verwaltung der Mailboxen einen eigenen Systembenutzer. Dieser sollte jedoch über keinen direkten Login verfügen, sondern lediglich für das Management der Mail-Storage verwendet werden. In unserem Fall erhält der Benutzer bzw. die Gruppe den Namen vmail und als UID bzw. GID 5000 zugewiesen.

groupadd -g 5000 vmail

useradd -s /usr/sbin/nologin -u 5000 -g 5000 vmail

Wir überprüfen, ob die obigen Kommandos korrekt ausgeführt wurden, indem wir uns die Benutzeridentität von vmail anschauen:

id vmail

Als Ausgabe erhalten wir die entsprechenden Benutzerinformationen:

```
uid=5000(vmail) gid=5000(vmail) groups=5000(vmail)
```

Anschließend erstellen wir das Mailbox-Verzeichnis und ordnen dieses dem Benutzer und der Gruppe vmail zu:

mkdir -p /var/mail/vhosts/example.com

chown -R vmail:vmail /var/mail/vhosts

Authentifizierung der Clients via Dovecot SASL

SMTP bietet von sich aus keinerlei Authentifizierungsmechanismen, was insbesondere von Anfang an die Spam-Problematik verstärkt hat. Auch heute findet man unzählige SMTP-Server, die keinerlei Authentifizierung erfordern und E-Mails von beliebigen Absendern annehmen (sogenannte offene Relays). Um u.a. diesen Misstand zu beseitigen, wurden die SASL Bibliotheken geschaffen, die SMTP um Authentifizierung erweitern (SMTP AUTH). Zum einen existiert eine Cyrus SASL Implementierung, zum anderen Dovecot SASL. Da wir später Dovecot als IMAP/POP3 Server installieren und konfigurieren werden, verwenden wir an dieser Stelle Dovecot SASL. Wir aktivieren dieses mittels der folgenden Direktiven in der Hauptkonfigurationsdatei von Postfix /etc/postfix/main.cf:

```
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain = $mydomain
```

Erläuterungen:

- smtpd_sasl_auth_enable: aktiviert die Authentifizierung via SASL beim SMTP Daemon
- smtpd_sasl_type: legt fest, welche SASL Implementierung verwendet wird. Aktuell können Cyrus SASL bzw. Dovecot SASL verwendet werden.
- smtpd_sasl_path: definiert den Pfad, wo der Authentifizierungsdaemon zu finden ist. In unserem Fall verwenden wir einen relativen Pfad zur Spool-Queue von Postfix /var/spool/postfix (keine absolute Pfadangabe an dieser Stelle, das Postfix u.U. in einer chroot-Umgebung läuft).

Sicherung der Kommunikation via TLS/SSL

Da wir u.a. die Zugangsdaten der Clients, die sich an unserem MTA anmelden im Klartext (durch den verwendeten Authentifizierungsmechanismus PLAIN lediglich Base64 codiert) an diesen übergeben, müssen wir auf jeden Fall für eine sichere Verschlüsselung der gesamten Client-Server-Kommunikation sorgen. Dies erreichen wir durch die Verschlüsselung des Kommunikationskanals mittels TLS (früher SSL).

Zunächst aktivieren und konfigurieren wir die Verwendung von TLS für die Kommunikation zwischen den E-Mail Clients und unserem MTA. Dazu legen wir in der Hauptkonfigurationsdatei /etc/postfix/main.cf die folgenden Direktiven fest:

```
smtpd_use_tls=yes
smtpd_tls_security_level = may
smtpd_tls_auth_only = no
smtpd_tls_cert_file=/etc/ssl/certs/example.pem
smtpd_tls_key_file=/etc/ssl/private/example.key
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtpd_tls_received_header = yes
tls_random_source = dev:/dev/urandom
```

Nun benötigen wir das entsprechende SSL-Zertifikat. Da ich ein treuer Anhänger von CAcert bin, empfehle ich an dieser Stelle stets ein solches freies CAcert-Zertifikat zu verwenden. In unserem Fall arbeiten wir mit einem Wildcard-Zertifikat, das für alle Hosts der Domain example.com gültig ist. Ich habe bereits in einem **anderen Howto** beschrieben, wie ein solches CAcert-Zertifikat erstellt wird. Nachdem wir dieses generiert haben, kopieren wir das Zertifikat selbst in das Verzeichnis /etc/ssl/certs, sowie den zugehörigen privaten Schlüssel in das Verzeichnis /etc/ssl/private:

cp example.pem /etc/ssl/certs/

cp example.key /etc/ssl/private/

Anschließend schützen wir den privaten Schlüssel vor Lesezugriffen Dritter:

chmod 600 /etc/ssl/private/example.key

Sonstige Policies

```
smtpd_require_helo = yes
smtpd_client_restrictions = permit_mynetworks, permit_sasl_authenticated, \
  reject_invalid_hostname,  reject_unknown_client, reject_rbl_client sbl-xbl.spamhaus.org \
  smtpd_sender_restrictions = permit_mynetworks, reject_unknown_address, \
  reject_unknown_sender_domain, reject_non_fqdn_sender
smtpd_recipient_restrictions = permit_mynetworks, permit_sasl_authenticated, \
  reject_unauth_destination
smtpd_recipient_limit = 250
```

In einigen früheren E-Mail Clients wurde die Authentifizierung via SMTP fehlerhaft implementiert (RFC 2554). Während die SMTP-Spezifikation vom MTA verlangt, dass die von ihm unterstützten Authentifizierungsmechanismen hinter dem Schlüsselwort AUTH durch ein Leerzeichen getrennt angegeben werden, erwarten die besagten E-Mail Clients (u.a. Outlook Express 4.0 und Exchange 5.0) hinter AUTH ein Gleichheitszeichen. Daher existiert eine entsprechende Direktive, die dieses fehlerhafte Verhalten bewirkt und damit die alten E-Mail Clients mit dem Mailserver kommunizieren können. Alle aktuellen E-Mail Clients, die diesen Bug nicht aufweisen, sind so implementiert, dass sie sich an dem Gleichheitszeichen nicht stören:

```
broken_sasl_auth_clients = yes
```

Mapping von virtuellen Domains und virtuellen Mailboxen

Wie bereits in einem der vorherigen Abschnitte beschrieben, benötigen wir einige Konfigurationsdateien, um die virtuellen Domains, die einzelnen Mailboxen und das Mapping zwischen E-Mail User und Mailbox zu definieren.

Zunächst einmal erstellen wir die Konfigurationsdatei /etc/postfix/virtual_domains und legen darin zeilenweise alle Domains (die nicht bereits in mydestination definiert wurden) fest, für die der E-Mail Server zuständig sein soll. Alle E-Mails, die einen dieser Domainnamen als Empfänger beinhalten, werden vom Delivery Agent virtual dem entsprechenden E-Mail Konto auf unserem Mailserver zugestellt. In unserem Fall haben wir Postfix bisher lediglich für localhost in /etc/postfix/main.cf als Hauptdomain definiert und fügen an dieser Stelle die beiden Domains mydomain.com und mydomain.de als virtuelle Domains hinzu:

```
example.com
example.de
```

Zudem erstellen wir eine neue Konfigurationsdatei /etc/postfix/vmailbox, in die wir zeilenweise Mappings eintragen zwischen den virtuellen E-Mail-Adressen und den physischen Mailboxen der jeweiligen Benutzer:

```
info@mydomain.com               mydomain.com/info/
admin@mydomain.com              mydomain.com/info/
abuse@mydomain.com              mydomain.com/info/
webmaster@mydomain.com          mydomain.com/info/
m.mustermann@mydomain.com       mydomain.com/m.mustermann/
@mydomain.com                   mydomain.com/misc/
@mydomain.de                    mydomain.com/misc/
```

Der Ort, an dem der LDA die Mailbox im Dateisystem speichert, ist in der zweiten Spalte (relativ zu dem Verzeichnis virtual_mailbox_base) angegeben. Mittels des Slashes am Ende der Mailbox-Definition weisen wir Postfix an, Maildir als Mailbox-Format zu verwenden. Würden wir das angehangene Slash weglassen, würde Postfix hierfür mbox verwenden. Es ist in jedem Fall empfehlenswert sich aus Performancegründen für Maildir zu entscheiden. So benötigt Maildir u.a. keine Locking-Mechanismen, da jede E-Mail in einer eigenen Datei abgelegt wird, während mbox alle E-Mails in einer einzigen physikalischen Datei aneinander hängt.

Anmerkung: neben der Zuordnung von E-Mail Adressen zu Mailboxen enthält diese Datei sogenannte Catchall-Adressen (die letzten beiden Zeilen). Diese Definition bewirkt, dass alle E-Mails, die an einen nicht existierenden Benutzer an diese Domain geschickt werden, automatisch der in dieser Zeile festgelegten Mailbox zugestellt werden. An dieser Stelle ist jedoch Vorsicht geboten, da viele Spammer ihre Spam-Mails an Adressen verschicken, die nicht existieren. Von daher sollte man an dieser Stelle diese Mails nicht an eine bereits andersweitig benötigte Mailbox zustellen, sondern eine eigene Mailbox dafür schaffen (in unserem Fall also die Mailbox /var/mail/vhost/mydomain.com/misc).

Nun generieren wir eine entsprechende Lookup-Tabelle aus der obigen Konfigurationsdatei mittels:

```
# postmap /etc/postfix/vmailbox
```

Anmerkung: das obige Kommando erstellt aus der Konfigurationsdatei /etc/postfix/vmailbox die binäre Lookup-Tabelle /etc/postfix/vmailbox.db. Diesen Vorgang müssen wir nach jeder Änderung der Konfigurationsdatei wiederholen.

Eine weitere Konfigurationsdatei /etc/postfix/virtual_alias bildet die Basis für das Aliasing von E-Mail Empfängern. In dieser Mapping-Datei können wir Weiterleitungen für bestimmte E-Mail Adressen aktivieren. In unserem Fall sieht die Mapping-Datei folgendermaßen aus:

```
maxmustermann@mydomain.com          m.mustermann@mydomain.com
max.mustermann@mydomain.com         m.mustermann@mydomain.com
mmu@mydomain.com        m.mustermann@mydomain.com
```

Auch diese Konfigurationsdatei müssen wir in eine Lookup-Table überführen mittels:

```
# postmap /etc/postfix/virtual_alias
```

Dovecot SASL

Wie bereits oben beschrieben, besitzt SMTP von sich aus keine Authentifizierungsmechanismen. Daher verwenden wir bei unserem Setup die Dovecot SASL Bibliotheken und Module. Durch deren Verwendung gewähren wir nur den Clients den Versand von E-Mails über den Mailserver, die sich vorher am SMTP-Server erfolgreich authentifizieren konnten. Bei der Konfiguration des Postfix MTA haben wir Dovecot SASL bereits aktiviert. Durch die Wahl von Dovecot SASL gegenüber Cyrus SASL gestaltet sich die Konfiguration der SASL-Authentifizierung relativ einfach: die gesamte Authentifizierung wird von Dovecot und der ihm zugrundeliegenden Benutzerdatenbank übernommen. Daher müssen wir an dieser Stelle keine weiteren Anpassungen unserer MTA-Konfiguration vornehmen.

Logging

Die Art, wie Postfix Ereignisse loggt, kann in vielerlei Hinsicht beeinflusst werden. Zum einen schreibt Postfix von sich aus alle Statusmeldungen in die Logdatei /var/log/mail.log. Reicht die Granularität der Meldungen nicht aus, können wir den Loglevel erhöhen, indem wir den SMTP Daemon im Verbose Modus starten. Dazu ändern wir in der Konfigurationsdatei /etc/postfix/master.cf die Zeile 11:

```
smtp    inet  n       -       -       -       -       smtpd
```

in

```
smtp    inet  n       -       -       -       -       smtpd -v
```

Anmerkung: sollten wir eine noch feinere Granularität der Log-Meldungen benötigen, erreichen wir dies durch den Parameter -vv.

Anschließend starten wir den MTA neu mittels:

/etc/init.d/postfix restart

Sonstiges

Neben der Hauptkonfigurationsdatei /etc/postfix/main.cf liest der MTA bei dessen Start eine weitere Konfigurationsdatei /etc/postfix/master.cf ein. Genauer gesagt wird ein Prozess master gestartet, dessen Aufgabe es wiederum ist alle für den Betrieb des MTA benötigten Dienste und Prozesse zu starten. Dessen Verhalten wird durch die obige Konfigurationsdatei bestimmt. Im Allgemeinen brauchen wir uns bei einer Standardinstallation eines Mailsystems um die Konfigurationsdatei nicht kümmern. Sollen dagegen Mails vor der Aus- bzw. Einlieferung bestimmte Prozesse durchlaufen, wie bspw. einen Virusscan oder Spamfilter, muss diese Konfigurationsdatei angepasst werden.

Anmerkung: wollen wir alle durch uns in der Konfigurationsdatei /etc/postfix/main.cf gesetzten Konfigurationsoptionen aufbereitet anzeigen, genügt das folgende Kommando:

postconf -n

IMAP/POP3 Server Dovecot

Wie bereits am Anfang dieses Howtos beschrieben, verwenden wir in unserem Fall Dovecot als IMAP und POP3 Server. Obwohl es sich bei Dovecot bspw. im Vergleich zu Courier um ein recht junges Projekt handelt, ist dennoch der IMAP/POP3 Server extrem ausgereift, performant und reich an Features.

Installation von Dovecot

Zunächst einmal installieren wir Dovecot mittels:

aptitude install dovecot-imapd dovecot-pop3d

Konfiguration von Dovecot

Die Konfiguration des IMAP/POP3 Servers Dovecot erweist sich als sehr einfach, da die Anzahl der Konfigurationsdateien und -optionen sehr übersichtlich ist. Zunächst einmal modifizieren wir die Konfigurationsdatei /etc/dovecot/dovecot.conf. Darin legen wir bspw. fest, welche Protokolle Dovecot verwenden soll. Da wir sowohl IMAP als auch POP3 benötigen, muss die entsprechende Definition (ca. Zeile 24) folgendermaßen aussehen:

```
protocols = imap imaps pop3 pop3s
```

Auch, wenn es in jedem Fall ratsam ist, nur über verschlüsselte Verbindungen mit dem IMAP/POP3 Server zu kommunizieren, kann es dennoch Situationen bzw. Software geben, bei der die Aktivierung von TLS nicht möglich ist. Daher erlauben wir den Clients sich auch auf unverschlüsseltem Wege bei unserem Dovecot-Server zu authentifizieren (ca. Zeile 53):

```
disable_plaintext_auth = no
```

Ebenfalls legen wir den Pfad zur Logdatei fest (ca. Zeile 69):

```
log_path = /var/log/dovecot
```

Zudem wollen wir die sicheren Varianten IMAPS und POP3S zur Abholung von Mails verwenden. Daher benötigen wir an dieser Stelle ein SSL-Zertifikat und müssen den Ort, an dem dieses zu finden ist, Dovecot mitteilen. In unserem Fall verwenden wir ein bereits bei der Installation von Postfix generiertes Wildcard-Zertifikat (für *.example.com). Damit Dovecot mit diesem arbeiten kann, teilen wir Dovecot mit, an welcher Stelle der IMAP/POP3 Server das SSL-Zertifikat und den

Schlüssel findet (ca. Zeile 99):

```
ssl_cert_file = /etc/ssl/certs/example.pem
ssl_key_file = /etc/ssl/private/example.key
```

Anschließend legen wir den Pfad zu unserem Postfix-Verzeichnis, das die Mailboxen beinhaltet (ca. Zeile 218):

```
mail_location = maildir:/var/mail/vhosts/%d/%n
```

Anmerkung: Bei der Konfiguration von Postfix haben wir uns entschieden, die E-Mails eines jeden Benutzers in dem Verzeichnis /var/mail/vhosts/Domainname/Benutzername im Maildir-Format zu speichern. Daher ist die obige Konfigurationsoption dafür zuständig, dass Dovecot dieses Verzeichnis und die darin enthaltenen E-Mails lesen kann.

Für die Zustellung der E-Mails verwenden wir den Dovecot-eigenen LDA. Auch dieser benötigt einige Konfigurationsoptionen (ca. Zeile 704)

```
protocol lda {
  hostname = mail.example.com
  postmaster_address = postmaster@example.com
  auth_socket_path = /var/run/dovecot/auth-master
  mail_plugin_dir = /usr/lib/dovecot/modules/lda
  mail_plugins = cmusieve
}
```

Nun müssen wir die Authentifizierungsmechanismen, die den Clients zur Verfügung gestellt werden, festlegen. Zudem übernimmt Dovecot die gesamte Authentifizierung aller SMTP-Clients, da wir Postfix so konfiguriert haben, dass dieser die Authentifizierung an die Dovecot SASL Bibliotheken und Module delegiert. Dazu passen wir den entsprechenden Konfigurationsblock (ca. Zeile 796) folgendermaßen an:

```
auth default {
  mechanisms = plain login
  userdb passwd-file {
    args = /var/mail/vhosts/%d/passwd
  }
  passdb passwd-file {
    args = /var/mail/vhosts/%d/shadow
  }
  socket listen {
    client {
      path = /var/spool/postfix/private/auth
      mode = 0660
      user = postfix
      group = vmail
    }
    master {
      path = /var/run/dovecot/auth-master
      mode = 0600
      user = vmail
      group = vmail
    }
  }
}
```

Als Authentifizierungsmechanismus verwenden wir PLAIN, d.h. die Credentials (Benutzername und Passwort) werden lediglich Base64 codiert und so ohne eine weitere Verschlüsselung auf der Ebene der Authentifizierung zwischen Client und Server ausgetauscht. Daher werden wir den gesamten Traffic zwischen E-Mail Client und IMAP-Server via TLS verschlüsseln. Für ältere Clients, die sich nicht standardkonform bzgl. der Authentifizierung verhalten, verwenden wir zusätzlich den Authentifizierungsmechanismus LOGIN, bei dem die Credentials zwischen Client und Server ebenfalls im Klartext ausgetauscht werden.

Zudem verwalten wir die Benutzerdaten, wie Benutzername und Passwort in Dateien, die dem Dovecot passwd-file Schema entsprechen (einfache Textdateien; Dovecot kann die Zugangsdaten jedoch auch via Datenbank, LDAP etc. verwalten). Dabei richten wir unseren Dovecot IMAP/POP3 Server so ein, dass dieser die entsprechenden Credentials in den beiden Dateien passwd und shadow unterhalb des Domain-Verzeichnisses verwaltet (angelehnt an die beiden typischen

Dateien /etc/passwd und /etc/shadow, wie sie bei der Benutzerverwaltung bei UNIX-Systemen üblich sind). So legen wir für jede Domain die entsprechenden Benutzer an.

Nun fügen wir den Benutzer postfix der Gruppe vmail hinzu:

usermod -aG vmail postfix

Wir überprüfen das Ergebnis der obigen Operation mittels:

id postfix

... und erhalten die folgende Ausgabe:

```
uid=106(postfix) gid=108(postfix) Gruppen=108(postfix),5000(vmail)
```

Zudem erzeugen wir eine leere Logdatei /var/log/dovecot und setzen deren Zugriffsrechte so, dass die entsprechende Dienste diese lesen und schreiben können:

touch /var/log/dovecot

chgrp vmail /var/log/dovecot

chmod 660 /var/log/dovecot

Um einen neuen E-Mail-Benutzer anzulegen, verwenden wir das folgende Skript, das wir unter /usr/local/bin/addmailuser anlegen:

```
#!/bin/bash
#
# Script to add users for
# Dovecot/PostFix using Virtual Users

USAGE="Usage: $0 EMAIL PASSWORD [BASEDIR]";

if [ ! -n "$2" ]
then
  echo $USAGE;
  exit 1;
fi

USERNAME=$(echo "$1" | cut -f1 -d@);
DOMAIN=$(echo "$1" | cut -f2 -d@);
ADDRESS=$1;
PASSWD=$2;

if [ -n "$3" ]
then
  if [ ! -d "$3" ]
  then
    echo $USAGE;
    echo "BASEDIR must be a valid directory!";
    echo "I would have tried, $(postconf | grep ^virtual_mailbox_base | cut -f3 -d' ')";
    exit 2;
  else
    BASEDIR="$3";
  fi
else
  BASEDIR="$(postconf | grep ^virtual_mailbox_base | cut -f3 -d' ')";
fi

if [ -f /etc/postfix/vmailbox ]
then

  echo "Adding Postfix user configuration..."
  echo $ADDRESS $DOMAIN/$USERNAME/ >> /etc/postfix/vmailbox
  postmap /etc/postfix/vmailbox

  if [ $? -eq 0 ]
  then
    echo "Adding Dovecot user configuration..."
    echo $USERNAME::5000:5000::$BASEDIR/$DOMAIN/$USERNAME >> $BASEDIR/$DOMAIN/
    echo $USERNAME":"$(dovecotpw -p $PASSWD) >> $BASEDIR/$DOMAIN/shadow
    chown root:root $BASEDIR/$DOMAIN/passwd && chmod 600 $BASEDIR/$DOMAIN/passwd
    chown root:root $BASEDIR/$DOMAIN/shadow && chmod 600 $BASEDIR/$DOMAIN/shadow
    /etc/init.d/postfix reload
  fi
```

```
fi
```

Anschließend machen wir das Skript ausführbar:

```
# chmod a+x /usr/local/bin/addmailuser
```

Beim Anlegen eines neuen Benutzers werden die beiden Dateien /var/mail/vhosts/Domainname/passwd und /var/mail/vhosts/Domainname/shadow um die entsprechenden Nutzerdaten erweitert (s. u.).

Postfix selbst bringt seinen eigenen MDA (Mail Delivery Agent) virtual mit, der E-Mails den virtuellen Mailboxen zustellt. Da wir in unserem Fall Postfix in Verbindung mit dem IMAP/POP3 Server Dovecot verwenden, benutzen wir statt virtual den Dovecot-eigenen LDA (Local Delivery Agent) dovecot. Dazu registrieren wir einen neuen Service dovecot am Ende der Konfigurationsdatei /etc/postfix/master.cf:

```
dovecot   unix  -       n       n       -       -       pipe
  flags=DRhu user=vmail:vmail argv=/usr/lib/dovecot/deliver -f ${sender} -d ${recipient}
```

Anmerkung: einen entsprechenden Benutzer und die Gruppe vmail haben wir bereits bei der Konfiguration von Postfix erstellt (UID/GID 5000).

Zum Schluss starten wir Postfix und Dovecot neu:

```
# /etc/init.d/sendmail stop && /etc/init.d/postfix restart && /etc/init.d/dovecot restart
```

Logging

Nicht immer geht die Inbetriebnahme eines Mail-Servers problemlos einher, daher müssen wir u.U. uns auf Fehlersuche begeben. Wir haben Dovecot bereits so konfiguriert, dass die entsprechenden Fehler- und Infomeldungen in der Logdatei /var/log/dovecot festgehalten werden. Es werden dabei jedoch nur die groben Fehler protokolliert. Möchte man bestimmte Funktionalitäten des IMAP/POP3 Servers debuggen, empfiehlt es sich, die entsprechenden Logging-Optionen zu aktivieren bzw. den Log-Level zu erhöhen. Dazu bietet die Konfigurationsdatei /etc/dovecot/dovecot.conf die folgenden Optionen:

- auth_verbose = yes: loggt alle fehlerhaften Authentifizierungsversuche.
- auth_debug = yes: loggt alle Authentifizierungsversuche im Debug-Modus (aktiviert automatisch auth_verbose). Passwörter werden in den Logs mit <hidden> markiert.
- auth_debug_passwords = yes: verhält sich wie auth_debug, Passwörter werden jedoch nicht maskiert.
- mail_debug = yes: loggt alles, was für die E-Mail Kommunikation relevant ist (u.a. wo Dovecot die E-Mail-Folder vermutet).
- verbose_ssl = yes: loggt alle Fehler und Warnungen, die mit SSL in Verbindung stehen.

Hinzufügen eines neuen E-Mail Accounts

Da wir Dovecot SASL die gesamte Authentifizierung der Benutzer überlassen, müssen wir neue Benutzer dessen Datenbasis hinzufügen (s.o.). Mittels des obigen Skriptes gestaltet sich das Hinzufügen eines neuen Benutzers sehr einfach:

```
# addmailuser m.mustermann@example.com MyMailPasswort
```

Mittels dieses Kommandos werden u.a. die beiden Dateien /var/mail/vhosts/example.com/passwd und /var/mail/vhosts/example.com/shadow erstellt, bzw. falls diese bereits existieren sollten, diesen die entsprechenden Benutzerdaten hinzugefügt:

- /var/mail/vhosts/example.com/passwd:

```
m.mustermann::5000:5000::/var/mail/vhosts/example.com/m.mustermann
```

- /var/mail/vhosts/example.com/shadow:

```
m.mustermann:{HMAC-MD5}xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Testen der Mailserver-Umgebung

Nachdem wir sowohl den MTA Postfix als auch den IMAP/POP3-Server Dovecot soweit installiert und konfiguriert haben, können wir an dieser Stelle die Installation ausgiebig testen. Dazu öffnen wir zunächst zwei weitere Shells und lassen uns darin die Log-Ausgaben anzeigen:

```
# tail -f /var/log/mail.log
```

```
# tail -f /var/log/dovecot
```

Anschließend starten wir die einzelnen Dienste neu mittels:

```
# /etc/init.d/postfix restart
```

```
# /etc/init.d/dovecot restart
```

An dieser Stelle sollten in den Logdateien keine Fehlermeldungen auftauchen. Konnten alle Konfigurationsdateien sauber eingelesen und die Dienste gestartet werden, sollten die Logs in etwa die folgenden Meldungen liefern:

```
Sep  9 11:28:01 mail postfix/master[13529]: terminating on signal 15
Sep  9 11:28:02 mail postfix/master[13842]: daemon started -- version 2.5.5, \
                                  configuration /etc/postfix
```

bzw.

```
dovecot: 2010-09-09 11:28:43 Warning: Killed with signal 15
dovecot: 2010-09-09 11:28:43 Info: Dovecot v1.0.15 starting up
```

Testen des MTA's Postfix

Bevor wir den Mailserver in Produktivbetrieb nehmen, werden wir zunächst einmal unseren MTA Postfix testen. Zunächst einmal benötigen wir die Credentials eines der in der SASL Datenbank vorhandenen Benutzers in dessen Base64 Codierung (Test des Authentifizierungsmechanismus PLAIN). Dazu rufen wir das folgende Kommando auf:

```
# perl -MMIME::Base64 -e 'print
encode_base64("\000m.mustermann\@example.com\000MeinSMTPPasswort")'
```

Das Kommando liefert uns den benötigten Base64-String:

```
AG0ubXVzdGVybWFubkBleGFtcGxlLmNvbQBNZWluU01UUFBhc3N3b3J0
```

Anmerkung: obwohl der obige String sehr kryptisch aussieht, handelt es sich lediglich um eine einfache Base64-Codierung und keine Verschlüsselung! Das Decodieren des Base64-Strings ist genauso einfach möglich, um den ursprünglichen Text zu erhalten:

```
# perl -MMIME::Base64 -e 'print
decode_base64("AG0ubXVzdGVybWFubkBleGFtcGxlLmNvbQBNZWluU01UUFBhc3N3b3J0")'
```

```
m.mustermann@example.comMeinSMTPPasswort
```

Als Erstes führen wir einen einfachen Test des Mailservers durch, indem wir eine Mail von diesem aus an eine der vorhandenen E-Mail Adressen schicken:

```
# echo "%Dies ist ein Test ..." | mail -s "Testmail" m.mustermann@example.com%
```

Im Anschluss daran sollte diese Mail im Verzeichnis /var/mail/vhosts/example.com/m.mustermann/new zu finden sein. In unserem Fall trägt die Datei mit dem Inhalt der Mail den Namen 1252520668.V36I7648d2M833891.mail.example.com:

```
Return-Path: <root@example.com>
X-Original-To: m.mustermann@example.com
Delivered-To: m.mustermann@example.com
Received: by mail.example.com (Postfix, from userid 0)
      id C3D7F6705FC; Wed,  9 Sep 2010 20:24:28 +0200 (CEST)
To: m.mustermann@example.com
```

```
Subject: Testmail
Message-Id: <20090909182428.C3D7F6705FC@mail.example.com>
Date: Wed,  9 Sep 2009 20:24:28 +0200 (CEST)
From: root@example.com (root)

Dies ist ein Test ...
```

Damit haben wir bereits gezeigt, dass die lokale Zustellung der E-Mails ordnungsgemäß funktioniert. Auch die Zustellung der E-Mails, die an unterschiedliche Aliase (s. Datei /etc/postfix/virtual_alias) adressiert sind, sollte kein Problem darstellen und an dieser Stelle getestet werden:

echo “Dies ist ein Test …” | mail -s “Testmail” mmu@example.com

echo “Dies ist ein Test …” | mail -s “Testmail” max.mustermann@example.de

Zunächst sollten die obigen Tests reichen. Im Folgenden testen wir den Mailserver ausgiebig und systematisch. Zunächst schauen wir von einem beliebigen Rechner aus, ob der entsprechende SMTP Dienst ansprechbar ist, indem wir uns mit diesem via Telnet verbinden:

$ telnet mail.example.com 25

Bei einem erfolgreichen Verbindungsaufbau präsentiert uns Postfix einen entsprechenden Eingabeprompt:

```
Trying xx.xx.xx.xx...
Connected to mail.example.com
Escape character is '^]'.
220 mail.example.com ESMTP
```

Inbesondere die letzte Zeile der obigen Ausgabe (Banner des Mailservers) quittiert uns, dass der SMTP-Server ordnungsgemäß initialisiert ist und Anfragen entgegennehmen kann. Dies nutzen wir und bauen manuell eine Verbindung zu diesem auf:

EHLO localhost

Daraufhin erhalten wir die folgende Statusmeldung:

```
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH PLAIN LOGIN
250-AUTH=PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```

Authentifizierung via SASL

Nun überprüfen wir, ob die Authentifizierung via Dovecot SASL ordnungsgemäß funktioniert. An der obigen Statusausgabe erkennen wir, dass der Server den Authentifizierungmechanismus PLAIN unterstützt. Ebenso wird ein Login via TLS unterstützt. Das folgende Kommando simuliert einen Login-Versuch eines zu authentifizierenden Mail-Clients (Credentials s.o.), so dass dieser unseren SMTP-Server als ausgehenden Mailserver nutzen kann:

AUTH PLAIN AG0ubXVzdGVybWFubkBleGFtcGxlLmNvbQBNZWIuU01UUFBhc3N3b3J0

Sind wir bis hierhin Schritt-für-Schritt diesem Howto gefolgt, quittiert uns der MTA die erfolgreiche Authentifizierung:

```
235 2.7.0 Authentication successful
```

Neben dem Authentifizierungsmechanismus PLAIN verwenden wir in unserem Fall auch LOGIN, der bspw. zur Anbindung älterer Microsoft-Clients benötigt wird. Auch diesen Mechanismus können wir überprüfen. Dazu benötigen wir zunächst unseren Benutzernamen und unser Passwort als Base-64 String (einzeln, nicht zusammenhängend wie bei PLAIN):

perl -MMIME::Base64 -e 'print encode_base64("\000m.mustermann\@example.com")'

```
AG0ubXVzdGVybWFubkBleGFtcGxlLmNvbQ==
```

perl -MMIME::Base64 -e 'print encode_base64("\000MeinSMTPPasswort")'

```
AE1laW5TTVRQUGFzc3dvcnQ=
```

Anschließend verbinden wir uns, wie oben beschrieben per Telnet mit unserem Mailserver und setzen eine EHLO Nachricht ab. Zunächst teilen wir Dovecot unseren Benutzernamen mit:

AUTH LOGIN AG0ubXVzdGVybWFubkBleGFtcGxlLmNvbQ==

Daraufhin sollte Dovecot den String annehmen und uns mittels des folgenden Strings (Base64-Codierung von 'Passwort:') nach dem Passwort fragen:

```
334 UGFzc3dvcmQ6
```

Daraufhin geben wir den Base64-String für unser Passwort ein:

AE1laW5TTVRQUGFzc3dvcnQ=

Auch dieser Versuch, sich gegenüber dem MTA zu authentifizieren sollte erfolgreich verlaufen.

Versenden von E-Mails

Haben wir uns, wie oben beschrieben, beim Mailserver authentifiziert, können wir manuell eine E-Mail verschicken. Zunächst versenden wir eine E-Mail direkt an einen lokalen, auf unserem Mailserver existierenden E-Mail Account (in unserem Fall von m.mustermannexample.com@ an den Alias max.mustermannexample.de@). Dazu setzen wir im Telnet-Prompt die folgenden Kommandos ab:

MAIL FROM: m.mustermann@example.com

```
250 2.1.0 Ok
```

RCPT TO: max.mustermann@example.de

```
250 2.1.5 Ok
```

data

```
354 End data with <CR><LF>.<CR><LF>
```

.

```
250 2.0.0 Ok: queued as 66808670597
```

Diese E-Mail sollte nun ebenfalls in unserem Mail-Spool zu finden sein. Ebenfalls sollten wir an dieser Stelle manuell eine E-Mail an eine externe E-Mail Adresse versenden. Gleichzeitig sollten wir dabei die Logs, die in /var/log/mail.log auflaufen, beobachten. Insbesondere wird hier der Mailversand an externe E-Mail Adressen protokolliert.

Um sicherzugehen, dass unser E-Mail System den Restriktionen folgt und kein Open Relay darstellt, sollten wir folgende Szenarien simulieren:

- Versand von E-Mails ohne Authentifizierung sollte fehlschlagen
- Nach Authentifizierung sollten E-Mail sowohl lokal als auch extern zugestellt werden

Backup des Mailsystems

Da es sich bei einem Mailserver um ein komplexes Gefüge einzelner Dienste handelt, listen wir in diesem Abschnitt alle Konfigurationsdateien und Verzeichnisse auf, die in einer Sicherung des Mailservers auf keinen Fall fehlen dürfen. Dies beinhaltet insbesondere die von uns angepassten Konfigurationsdateien, sowie das Mail-Storage:

Postfix MTA

- /etc/postfix/main.cf: Hauptkonfiguration des MTA Postfix
- /etc/postfix/master.cf: Konfiguration des Master Prozesses
- /etc/postfix/virtual_alias: Definition von E-Mail Aliasen
- /etc/postfix/virtual_domains: Definition aller durch den MTA zu verwaltenden Domains
- /etc/postfix/vmailbox: Mapping zwischen E-Mail Adressen und Mailboxen
- /var/mail/vhosts/: Filestorage für E-Mails

Dovecot IMAP/POP3 Server

- /etc/dovecot/dovecot.conf: Hauptkonfigurationsdatei von Dovecot

Sonstiges (optional)

Sollte unser Server Ziel von Spam-Attacken werden, können wir das unten stehende, aber auch sehr restriktive Regelwerk für die Annahme und die Weiterleitung von E-Mails verwenden (Konfigurationsdatei /etc/postfix/main.cf):

```
smtpd_helo_required = yes
smtpd_helo_restrictions =
      reject_invalid_hostname,
      reject_non_fqdn_hostname

smtpd_recipient_restrictions =
      permit_mynetworks,
      reject_unknown_recipient_domain,
      permit_sasl_authenticated,
      reject_unauth_destination

smtpd_sender_restrictions =
      reject_unknown_address,
      reject_unknown_sender_domain,
      reject_non_fqdn_sender,
      reject_rhsbl_sender rhsbl.ahbl.org,
      reject_rhsbl_sender cart00ney.surriel.com,
      reject_rhsbl_sender in.dnsbl.org,
      reject_rhsbl_sender rhsbl.sorbs.net

smtpd_client_restrictions =
      permit_sasl_authenticated,
      reject_invalid_hostname,
      reject_rbl_client relays.ordb.org,
      reject_unknown_client,
      reject_rbl_client relays.bl.kundenserver.de,
      reject_rbl_client relays.visi.com,
      reject_rbl_client sbl-xbl.spamhaus.org,
      reject_rbl_client list.dsbl.org,
      reject_rbl_client multihop.dsbl.org,
      reject_rbl_client spamsources.fabel.dk,
      reject_rbl_client combined.njabl.org,
      reject_rbl_client dnsbl.sorbs.net
```

Verwendete Quellen

Dieses Howto wäre nie ohne die teilweise schon sehr gut aufbereiteten und teilweise doch sehr im Internet verstreuten Informationen und andere Howtos und Tutorials entstanden. Daher an dieser Stelle vielen Dank für den Input, die mir die folgenden Websites bieten und zur Erstellung dieses Howtos beitragen konnten:

- http://www.thinkdebian.org/archives/652
- http://holl.co.at/howto-email
- http://neranjara.org/article/title/How_to_configure_PostFix_and_Dovecot_for_Virtual_Users_with_out_a_Database_

vor 1 Monat 1 Gefällt mir Gefällt mir Antworten

Rufushischi

Super Tut aber mir sind fatale Probleme aufgefallen und zwar bei Ubuntu LTS 10.04 aktueller Dovecot/Postfix (2011):

1. Das webmail Script erstellt in der shadow nur info oder benutzername d.h. Authentifizierung geht ned man muss von Hand **domainl** .tld hinzufügen

2. Das LDA Mailplugin das ist anders und muss jetzt lauten:

```
mail_plugin_dir = /usr/lib/dovecot/modules/lda
mail_plugins = sieve
sieve_extensions = +imapflags
```

ansonsten funktioniert die Mailzustellung nicht. Das Tut ist ansonsten perfekt, innerhalb von nur 20 minuten (wenn man o.a. Fehler kennt) hat man einen voll funktionierenden Mailserver!

Gruss Rufus