Hintergründe und Basiswissen

Diese Seite schließt eine Lücke im Wiki. Sie soll Hintergrund und Basiswissen vermitteln, welches dabei hilft sich an der Entwicklung des invis-Servers zu beteiligen bzw. genutzte Techniken und Konfigurationen zu verstehen.

DNS-Server bind und LDAP

Schon von Anfang an setzen wir ein LDAP-Verzeichnis als Daten-Backend für den DNS-Server *bind* ein. Der zugrunde liegende Patch, welchen wir schon seit einiger Zeit selbst in die Software-Pakete einbauen müssen, wird inzwischen kaum noch weiter entwickelt. Statt dessen gehört der sogenannte DLZ (dynamic loadable zone) Treiber seit etwa openSUSE 11.2 (oder so) fest zum Funktionsumfang der openSUSE-Pakete. Die Nutzung dieses Treibers würde uns eine Menge Arbeit ersparen, da wir keine eigenen *bind* Pakete mehr im openSUSE-Buildservice pflegen müssten.

Schema-Definitionen

Jede Form der Datenspeicherung in einem LDAP-Verzeichnis bedarf der Definition von Attributen und Objektklassen (Tabellen- und Felddefinitionen einer SQL-Datenbank nicht ganz unähnlich). Diese Definitionen werden in Form sogenannter Schema-Dateien getroffen und haben Server-weit Gültigkeit. Zu finden sind Sie im Verzeichnis

```
/etc/openldap/schema
```

Der Aufbau der LDAP-Schemas (kein Witz, es schreibt sich so) „dlz.schema“ für den DLZ-Treiber und „dnszone.schema“ für den alten SDB-LDAP-Patch unterscheiden sich grundsätzlich voneinander.

Einige Unterschiede im Überblick:

- Allen Attributen und Objektklassen des DLZ-Schemas ist zur Vermeidung von Namensdopplungen die Silbe „dlz“ vorangestellt.
- Während das *DNSZone-Schema* lediglich die Objektklasse „dNSZone“ kennt über die alle DNS-Objekte dargestellt werden, verfügt das *DLZ-Schema* über Objektklassen für die verschiedenen DNS-Record-Typen. Dies dürfte etwas verwirrend wirken, da es für die verschiedenen DNS-Records auch entsprechende Attributdefinitionen gibt.
- Nach dem alten Schema beginnt jedes DNS-Objekt im LDAP mit dem Attribut „relativeDomainName“, welches als Schlüssel-Attribut angesehen werden kann. Im DLZ-Schema nimmt das Attribut „dlzRecordID“ diese Rolle ein. Eine Kennzeichnung des Objekts wird dabei erst über das Folge-Attribut erreicht: „dlzRecordID=05,dlzHostName=pc-buchhaltung,…“
- Während alle kennzeichnenden Informationen eines SOA-Records beim alten Schema im Attribut „sOARecord“ zusammengefasst wurden, kennt das DLZ-Schema für jeden Wert ein eigenes Attribut: z.B. „dlzSerial“, „dlzRefresh“, usw.
- Die Daten der MX-Records sind beim DLZ-Schema nicht Teil des SOA-Eintrages, sondern werden als eigene LDAP-Objekte angelegt.
- Das DLZ-Schema kennt weder Attribute noch Objektklassen für SRV-, mINFO, hINFO, SIG-, KEY-, sowie einige weitere DNS-Record-Typen. Diese können entweder über die Objektklasse

„dlzGenericRecord“ oder eigene Erweiterungen des Schemas realisiert werden. Für Letzteres ist mit „1.3.6.1.4.1.18420.1.3.X“ im Schema ein eigener Object-Identifier (OID) vorgesehen.
- Die meisten Attribute des DLZ-Schemas sind als Single-Values gekennzeichnet. D.H. gehören etwa zu einem Pointer-Record mehrere Hostnamen, können diese nicht in einem Objekt zusammengefasst werden, sondern es muss für jeden Hostnamen ein eigenes PTR-Objekt erzeugt werden.

Aufbau der LDAP-Objekte

Um die Unterschiede im Aufbau der jeweiligen LDAP-Struktur zu verdeutlichen werden im folgenden einige Standard-Einträge gegenübergestellt.

SOA-Record

- dNSZone-Schema

```
dn: relativeDomainName=@,ou=invis-net.loc,ou=forward,ou=zone.master,ou=DNS-
Server,dc=invis-net,dc=loc
dNSClass: IN
dNSTTL: 3600
mXRecord: 10 mail.invis-net.loc.
nSRecord: ns.invis-net.loc.
objectClass: dNSZone
objectClass: top
relativeDomainName: @
sOARecord: ns.invis-net.loc. root.invis-net.loc. 2006260342 3600 1800 604800
86400
zoneName: invis-net.loc
```

- DLZ-Schema

```
dn: dlzRecordID=11,dlzHostName=@,dlzZoneName=invis-
net.loc,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzSOARecord
dlzRecordID: 11
dlzHostName: @
dlzType: soa
dlzSerial: 2006260342
dlzRefresh: 3600
dlzRetry: 1800
dlzExpire: 604800
dlzMinimum: 86400
dlzAdminEmail: root.invis-net.loc.
dlzPrimaryns: ns.invis-net.loc.
dlzTTL: 10
```

```
dn: dlzRecordID=5,dlzHostName=@,dlzZoneName=invis-
server.loc,ou=zone.master,ou=DNS-Server,dc=invis-server,dc=loc
objectclass: dlzMXRecord
dlzRecordID: 5
dlzHostName: @
```

```
dlzType: mx
dlzData: mail
dlzPreference: 10
dlzTTL: 10
```

A-Record

- dNSZone-Schema

```
# A-Record des Servers
dn: relativeDomainName=invis5,ou=invis-
net.loc,ou=forward,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
aRecord: 192.168.220.10
dNSClass: IN
dNSTTL: 86400
objectClass: dNSZone
objectClass: top
relativeDomainName: invis5
zoneName: invis-net.loc
```

- DLZ-Schema

```
dn: dlzRecordID=14,dlzHostName=invis5,dlzZoneName=invis-
net.loc,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzARecord
dlzRecordID: 14
dlzHostName: invis5
dlzType: a
dlzIPAddr: 192.168.220.10
dlzTTL: 86400
```

PTR-Record

- dNSZone-Schema

```
dn: relativeDomainName=10,ou=220.168.192.in-
addr.arpa,ou=reverse,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
pTRRecord: invis5.invis-net.loc.
pTRRecord: ns.invis-net.loc.
pTRRecord: mail.invis-net.loc.
dNSClass: IN
dNSTTL: 86400
objectClass: dNSZone
objectClass: top
# Test ohne Leerzeichen
relativeDomainName:10
zoneName: 220.168.192.in-addr.arpa
```

- DLZ-Schema

```
dn: dlzHostName=10,dlzZoneName=220.168.192.in-
addr.arpa,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
```

```
objectclass: dlzHost
dlzHostName: 10
```

Die Begründung für diesen Eintrag finden Sie im nächsten Abschnitt.

```
dn: dlzRecordID=15,dlzHostName=10,dlzZoneName=220.168.192.in-
addr.arpa,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzPTRRecord
dlzRecordID: 15
dlzHostName: 10
dlzType: ptr
dlzData: invis5.invis-net.loc.
dlzTTL: 86400
```

```
dn: dlzRecordID=16,dlzHostName=10,dlzZoneName=220.168.192.in-
addr.arpa,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzPTRRecord
dlzRecordID: 16
dlzHostName: 10
dlzType: ptr
dlzData: mail.invis-net.loc.
dlzTTL: 86400
```

```
dn: dlzRecordID=17,dlzHostName=10,dlzZoneName=220.168.192.in-
addr.arpa,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzPTRRecord
dlzRecordID: 15
dlzHostName: 10
dlzType: ptr
dlzData: ns.invis-net.loc.
dlzTTL: 86400
```

LDAP-Struktur

Auch die Struktur des LDAP-Zweiges für den DNS-Server sollte sich mit Umstieg auf das DLZ-Schema ändern.

Bisher wurden beim invis-Server die DNS-Einträge nach Zone, Zonen-Typ und „Richtung“ (Vorwärts/Rückwärts) sortiert. Dies war in sich eher etwas unglücklich. Mit einem Umstieg werden wir uns an die Empfehlung des „Bind DLZ“ Projekts halten und lediglich nach Zone und Host sortieren. Eine Unterscheidung nach Vorwärts- oder Rückwärtsauflösung ist nicht generell nicht notwendig, da dies bereits über die Zone selbst klar wird.

Der Basisknoten bleibt hingegen unverändert:

```
ou=DNS-Server,ou=domain,ou=tld
```

Dem wurde beim alten Schema noch der Zonentyp (Master oder Slave), die Richtung der Auflösung sowie der Zonenname vorangestellt.

```
ou=domain.tld,ou=forward,ou=zone.master,...
```

Zukünftig wird sich dies Ändern. Zunächst ist der Zonenname hier mit einem eigenen Attribut gekennzeichnet, dafür eine eigene „Organizational Unit“ (ou) in den DN (Distinguished Name) eines DNS-Objektes einzufügen ist also obsolet. Weiterhin entfällt wie bereits erwähnt die Unterscheidung zwischen Vorwärts- und Rückwärtsauflösung, etwas worauf wir auch schon früher hätten verzichten können. Zu guter Letzt fügen wir der DLZ-Empfehlung folgend allerdings bei Bedarf den Hostnamen als zusätzlichen Knoten ein. Dies ist dann sinvoll, wenn etwa zu einem Host mehrere Einträge, seien es A- oder PTR-Records, folgen:

Also, entweder:

```
dlzZoneName=domain.tld,ou=zone.master,....
```

oder:

```
dlzHostname=host,dlzZoneName=domain.tld,ou=zone.master,....
```

Letzteres eben im Falle mehrerer Records zu einem Host.

Nameserver Konfiguration

Grundsätzlich hat sich auch die Konfiguration des Nameservers bind in Sachen Zugriff auf das LDAP-Backend geändert. Statt getrennte Zonenkonfigurationen für Forward und Reverse Zonen gibt es nur noch einen Eintrag in der Datei named.conf.

```
## LDAP Backend mit DLZ-Schema
dlz "ldap zone" {
        database "ldap 2
           v3 simple {uid=Admin,ou=DNS-Server,dc=invis-server,dc=loc}
{passwort} 127.0.0.1
           ldap:///dlzZoneName=$zone$,ou=zone.master,ou=DNS-Server,dc=invis-
server,dc=loc???objectclass=dlzZone
ldap:///dlzHostName=$record$,dlzZoneName=$zone$,ou=zone.master,ou=DNS-
Server,dc=invis-
server,dc=loc?dlzTTL,dlzType,dlzPreference,dlzData,dlzIPAddr,dlzPrimaryNS,dl
zAdminEmail,dlzSerial,dlzRefresh,dlzRetry,dlzExpire,dlzMinimum?sub?objectcla
ss=dlzAbstractRecord";
};
```

Wichtig bei der Bearbeitung der Einträge ist, dass die einzelnen LDAP-URLs nicht umbrochen werden und keine Leerzeichnen enthalten dürfen. Die notwendigen Anpassungen dieses Eintrags auf die jeweilige Umgebung beschränken sich auf den Bind-DN, mit dem sich der Dienst am LDAP-Verzeichnis anmeldet und den DN, der Zonen in der ersten LDAP-URL.

Daneben ist ggf. noch die Möglichkeit von Interesse bind mit mehreren Threads gleichzeitig auf das LDAP-Verzeichnis zugreifen zu lassen. Dies ist vor allem bei hochfrequentierten Nameservern ein gutes Mittel die Performance des Nameservers zu steigern. In kleineren Umgebungen spielt die keine große Rolle.

Im Beispiel wurden zwei gleichzeitige Threads ermöglicht. Hierfür steht die Ziffer (2) am Beginn der Definition des Daten-Backends. Vorgegeben ist der Wert 1, da die Erhöhung nur dan funktioniert, wenn das zugrunde liegende System Multithreading-fähig ist. Bei modernen Linux-Systemen ist dies in der Regel der Fall.

DDOS / Amplifier Attacke auf bind

Noch ein schönes Thema für die Knowledgebase, diesmal im Hinblick auf das Thema Rootserver.

Beginnend etwa am 25. Oktober 2012 geriet einer meiner produktiv genutzten Rootserver unter Beschuss. Aufgefallen ist mir das Ganze nur, weil invis-Server meiner Kunden immer schlechter via Internet erreichbar waren. Zunächst ließ mich mein Monitoring-System vermuten, dass es Probleme mit den DSL-Anbindungen bei den Kunden gibt. Es stellte sich aber heraus, dass der DNS-Server der für meine Kunden dynamisches DNS betreibt attackiert wurde und dadurch seinen Dienst nicht mehr zuverlässig erledigen konnte.

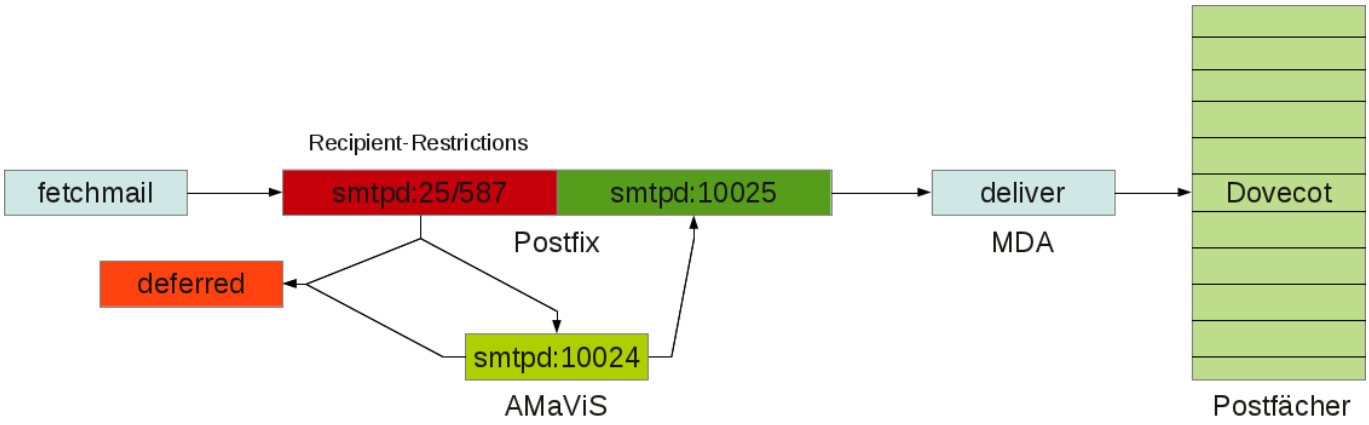
Postfix, Amavis & Dovecot

Da wir uns über kurz oder lang möglichst wieder von Cyrus-IMAP verabschieden wollen kümmern wir uns derzeit intensiv um das reibungslose Zusammenspiel von Postfix, Amavis und Dovecot. Ziel des Setups ist es:

- Dovecot als SASL-Authentifizierungsbackend für Postfix zu nutzen.
- Dovecots Deliver als MDA einzusetzen.
- IMAP-ACLs zu ermöglichen.
- Amavis als content_filter zu nutzen.

Die genannten Anforderungen ergeben ein recht komplexes Setup, daher widme ich mich hier den Hintergründen. Zunächst allerdings nur dem Teil der Einlieferung neuer Mails via ***fetchmail***.

Größte Schwierigkeit hierbei ist ein sauberes Mailrouting durch alle Komponenten. Dies beginnt bereits beim Fetchmail-Daemon. Zunächst ein schematischer Überblick über die Wege einer eingehenden Email:



fetchmail

In der Grundkonfiguration hat der fetchmail-Daemon auf openSUSE Systemen die unangenehme Eigenschaft eingehende Emails an Adressen des Schemas „user@***localhost***.domain.tld“, statt an „user@domain.tld“ weiterzuleiten. Dies kann das gesamte Verhalten der nachfolgenden Komponenten beeinflussen. Leider nicht zum Guten.

Dies kann auf zweierlei Wegen behoben werden. Entweder wird in der genutzten „fetchmailrc“ Datei die korrekte Adresse des lokalen Empfängers eingetragen oder ***fetchmail*** wird mit der Option „-D domain.tld“ aufgerufen.

Letzteres kann auf openSUSE Systemen in

```
/etc/sysconfig/fetchmail
```

bzw. auf invis-Servern in

```
/var/cornaz/sysconfig/fetchmail
```

konfiguriert werden:

```
## Type:        string
## Default:     ""
#
# Any additional fetchmail options. See fetchmail(1) manual page for
# more information. If you want to use --mda option, it may be required
# to change FETCHMAIL_USER to root. Consult your MDA documentation for
# more.
#
FETCHMAIL_EXPERT_OPTIONS="-D invis-net.loc"
```

Eingetragen werden muss an dieser Stelle die lokale Domain des Servers.

Postfix - Mail-Annahme

Postfix spielt die zentrale Rolle im gesamten Mail-Setup.

Da wir es bezogen auf ein invis-Server-Setup „lediglich“ mit einem Mailserver in einem lokalen Netzwerk zu tun haben, werden die in der Grafik genannten Restrictions nicht konfiguriert. Unser Server bekommt emails entweder nur via ***fetchmail*** oder von lokalen Benutzern eingeliefert. In beiden Fällen ist das Einliefern von Mails ohne Beschränkung erlaubt.

Zunächst muss Postfix der eigene Zuständigkeitsbereich bekannt gemacht werden. Die zugehörigen Einstellungen werden in

```
/etc/postfix/main.cf
```

vorgenommen.

Die notwendigen Konfigurationen:

```
# INTERNET HOST AND DOMAIN NAMES
myhostname = server.invis-net.loc

mydomain = invis-net.loc
masquerade_domains = $mydomain

# SENDING MAIL
myorigin = $mydomain

# RECEIVING MAIL
inet_interfaces = $myhostname, localhost
....
mydestination = $myhostname, $mydomain, localhost.$mydomain, localhost
```

Dabei legt *myhostname* den lokal gültigen FQDN und *mydomain* die lokale Domain des Servers fest. Die Option *myorigin* definiert beim Versand von Mails die Absende-Domain, so nicht angegeben. Die Option *masquerade_domains* maskiert Subdomains oder Hostnamen in Email-Adressen mit der dahinter angegebenen Domain. D.h. Wird eine Mail von „host1.invis-net.loc“ versandt so würde Postfix im gezeigten Beispiel daraus schlicht „invis-net.loc“ machen. Emails versendende Hosts im lokalen Netz würden also hinter der lokalen Domain „versteckt“.

Dies kann als zusätzliche Sicherheitsmaßnahme verstanden werden, die dabei hilft, das lokale Mailrouting vor Fehlern zu schützen.

Über die Option *inet_interfaces* wird festgelegt, auf welchen Netzwerkschnittstellen Postfix überhaupt Mails annimmt. Die hier getroffene Einstellung erlaubt die Annahme nur auf der Loopback-Schnittstelle sowie allen Schnittstellen denen der eigene FQDN zugewiesen wird. In der Praxis wird hier häufig einfach der Wert „all“ angegeben, was bedeutet, das Postfix die Einlieferung von Mails auf allen Schnittstellen des Servers erlaubt. Dies schließt auch das externe, mit dem Internet verbundene Netzwerk-Device mit ein. In Kombination mit einer fehlerhaft konfigurierten Firewall, kann das durchaus dazu führen, dass der Server von unerlaubter Seite als Mail-Relay missbraucht werden könnte.

Die Option *mydestination* legt die Ziele fest, für die Postfix Endpunkt der Email-Zustellung, also nicht weiterleitendes „Relay“ ist.

Nach der Annahme einer Mail, muss diese an AMaViS weitergegeben werden. Für die Einbindung von AMaViS als Viren- und Spam-Filter stehen mit „content_filter“ und „smtpd_filter_proxy“ zwei deutlich unterschiedliche Wege zur Verfügung. Während sich AMaViS mit der „smtpd_filter_proxy“ Methode direkt in die Einliefernde SMTP-Sitzung einklinkt und diese bis zum Abschluss der Prüfung offen hält, schließt sich die „content_filter“ Methode an die erfolgte Einlieferung an. Ist auf einem Internet-Mailserver Methode 1 die deutlich bessere, da hier „verseuchte“ gar nicht erst angenommen werden, so macht dieses Vorgehen auf einem invis-Server keinen Sinn, da die Mails hier vom lokalen Fetchmail-Dienst eingeliefert werden und somit eine Blockade nicht Zielführend ist.

Die Einbindung von AMaVis als Content-Filter findet an zwei Stellen im Postfix-Setup statt. In der Datei „main.cf“ erfolgt die eigentliche Integration des Filters durch Benennung eines „Transportweges“, welcher in

```
/etc/postfix/master.cf
```

konfiguriert wird. Zunächst die Datei „main.cf“:

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

Hier wird festgelegt, dass alle auf „normalem“ Wege eingehenden Mails über die Adresse 127.0.0.1 auf Port „10024“ dem Content-Filter zugeführt werden. Der eigentliche Transportweg wird hier mit „smtp-amavis“ benannt. Damit Postfix weiß, was es mit diesem Transport weg auf sich hat, muss dieser wie gesagt in der Datei „master.cf“ konfiguriert werden:

```
# Weitergabe an Amavis (betrifft nur content_filter, nicht
smtp_proxy_filter)
smtp-amavis     unix    -       -       y       -       2       smtp
    -o smtp_data_done_timeout=1200
    -o smtp_send_xforward_command=yes
    -o disable_dns_lookups=yes
    -o max_use=20
```

Wichtig hierbei sind vor allem die beiden Angaben in Zeile eins. Postfix wird mitgeteilt, dass die Weiterleitung per „smtp“ erfolgt und maximal zwei parallele Sitzungen gleichzeitig geöffnet werden dürfen. Die Anzahl der maximalen SMTP-Verbindungen kann in Abhängigkeit der lokalen Rechenleistung und dem Mail-Aufkommen variiert werden. In der Konfiguration des AMaViS-Dienstes befindet sich ein korrespondierender Parameter, der mindestens auf den gleichen Wert eingestellt sein muss.

Die nachfolgenden Optionen im Einzelnen:

- **smtp_data_done_timeout**: Timeout, nachdem die Weitergabe abgebrochen wird. Die Einstellung hier ist mit 1200 Sekunden doppelt so hoch wie der Vorgabewert, kann ja sein, dass AMaViS mal länger braucht. 😉
- **smtp_send_xforward_command**: Die Daten der einliefernden Sitzung wie Name, Adresse, Protokoll und HELO Name des original SMTP-Clients werden einfach an den Content-Filter weitergereicht.
- **disable_dns_lookups**: Es werden zur Ermittlung des Endpunkts keine DNS-Anfragen gestartet. Da ja ohnehin die IP-Adresse angegeben wurde ist das auch nicht notwendig.
- **max_use**: Postfix erlaubt maximal X eingehende Verbindungen für diesen Transportweg.

Von Bedeutung ist hier vor allem das „xforward“ Kommando, da die darüber an AMaViS übermittelten Informationen in die SPAM-Bewertung einfließen können.

AMaViS

In der AMaViS-Konfiguration müssen keine besonderen Einstellungen für Annahme und Rückgabe der Mails vorgenommen werden, da die bisher besprochenen Transportwege den Standard-Vorgaben von AMaViS entsprechen.

Wichtig ist allerdings die Konfiguration des lokalen Host-Namens sowie der lokalen Domain in der Datei

```
/etc/amavisd.conf
```

Hinweis: *Bei der Datei handelt es sich um eine Perl-Datei, es ist also besonders auf korrekte Syntax zu achten.*

```
##invis-server.org -- Default-Settings
$mydomain = 'julgs-net.loc';      # (no useful default)

$myhostname = 'zentrale.julgs-net.loc';  # fqdn of this host, default by
uname(3)
```

Sollte der Weg der Rücklieferung vom Standard abweichen, müssen die folgenden Zeilen vom Kommentarzeichen befreit und entsprechend den Gegebenheiten angepasst werden:

```
#$forward_method = 'smtp:[127.0.0.1]:10025';  # where to forward checked
mail
#$notify_method = $forward_method;            # where to submit
notifications
```

Weiterhin ist darauf zu achten, dass die Anzahl der maximal gestarteten AMaViS-Prozessen gleich oder größer der für Postfix vorgenommenen Einstellung ist. Der im Beispiel genannte Wert war „2“:

```
$max_servers = 2;     # num of pre-forked children (2..30 is common), -m
```

Postfix - Rücknahme und Weitergabe an Dovecot

Postfix nimmt von AMaViS geprüfte und nicht beanstandete Mails auf Port „10025“ wieder entgegen. Auch hierfür muss in

```
/etc/postfix/master.cf
```

ein Transportweg eingerichtet werden:

```
## Ruecktransport von amavis an Postfix
localhost:10025 inet    n       -       n       -       -       smtpd
  -o smtpd_tls_security_level=none
  -o content_filter=
  -o smtpd_proxy_filter=
  -o mynetworks=127.0.0.0/8
  -o smtpd_delay_reject=no
  -o smtpd_client_restrictions=permit_mynetworks,reject
  -o smtpd_helo_restrictions=
  -o smtpd_sender_restrictions=
  -o smtpd_recipient_restrictions=permit_mynetworks,reject
  -o smtpd_data_restrictions=reject_unauth_pipelining
  -o smtpd_end_of_data_restrictions=
  -o smtpd_restriction_classes=
  -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,no_unknown_recipient_checks
```

In Zeile 1 wird zunächst festgelegt, dass der Endpunkt dieses Transportweges ein SMTP-Dienst ist. Eine Beschränkung der maximalen Anzahl der Prozesse ist hier nicht notwendig, da hier ohnehin nur das ankommen kann was zuvor den bereits auf zwei max. Sitzungen beschränkten Weg durch AMaViS genommen hat. (Zugegeben, so ganz richtig ist das nicht, da durchaus auch andere Programme auf dem Server Mails auf 127.0.0.1:10025 einliefern könnten.)

Die weiteren Optionen im Einzelnen:

- **smtp_tls_security_level**: Da die gesamte Kommunikation über die Loopback-Schnittstelle des Servers läuft ist eine Verschlüsselung obsolet.
- **content_filter** & **smtpd_proxy_filter**: Beiden Optionen ist kein Wert zugewiesen, was dazu führt, dass bei der Rücklieferung der Mails diese nicht in Form einer Endlosschleife wieder an AMaViS übergeben werden.
- **mynetworks**: Legt das Netzwerk fest auf dem dieser Transportweg zur Verfügung steht. Hier werden die im normalen Postfix-Setup genannten Netzwerke auf dass Loopback-Netz beschränkt. Benötigt wird diese Einstellung vor allem für die nachfolgende Zeile.
- **smtpd_recipient_restrictions**: Hiermit wird die Einlieferung von Mails auf den zuvor benannten Netzwerken ohne weitere Restriktionen erlaubt. Alle Einlieferungsversuche aus anderen Netzen jedoch unterbunden.

Achtung: *In einigen der jüngeren invis-Releases (7.0/7.1) habe ich darüber hinaus die Option „-o local_header_rewrite_clients=“ gesetzt. Dies wird auch an anderer Stelle so angegeben. Diese Option sorgt allerdings dafür, dass das zuvor vorgenommene Address-Rewriting (aus lokaler Mail-Adresse mache externe Adresse) beim Rückliefern der Mail wieder rückgängig gemacht wird. Bedeutet die Empfänger einer solchen Mail sehen als Absende-Adresse die lokale Email-Adresse des Absender, die aber im Internet-Mailverkehr keine Gültigkeit hat.*

Postfix ist zwar durchaus in der Lage Mails direkt in sogenannte „Maildirs“ einzuliefern, allerdings ist dieser direkte Weg mit allerlei Nachteilen verbunden, da dadurch einige Stärken von Dovecot ungenutzt bleiben. Besser ist es die Verteilung der Mails in die jeweiligen Postfächer Dovecots eigenem MDA ***deliver*** zu übertragen.

Üblicherweise sind Maildirs in den Home-Verzeichnissen der Benutzer zu finden. Dies birgt einige Nachteile und Gefahren in sich. Die größte Gefahr dabei, geht wohl von den Nutzern selbst aus, die in der Lage sind diese Verzeichnisse einfach zu löschen. Dass ein Benutzer im „Aufräumrausch“ einfach mal Verzeichnisse löscht, mit denen er nichts anfangen kann, dürfte wohl keine Seltenheit sein.

Weiterhin verhindern Maildirs in den Home-Verzeichnissen der Benutzer die Anwendung von IMAP-ACLs also die Freigabe von Mail-Ordnern für andere Nutzer. Ursache hierfür sind die meist restriktiven Dateisystem-Zugriffsrechte bei Home-Verzeichnissen, die gesetzte IMAP-ACLs überlagern. D.h. die Maildirs müssen an einen anderen Ort verfrachtet werden und einem neutralen Benutzer gehören.

Üblicherweise werden für diesen Zweck sowohl ein Benutzer als auch eine Gruppe „vmail“ angelegt:

```
linux:~ # groupadd -g 399 vmail
linux:~ # useradd -c "Benutzerkonto fuer Dovecot Mailhandling" -s /bin/false
```

```
-g vmail vmail
```

Anschließend müssen die Rechte am Arbeits und Mail-Spool-Verzeichnis so angepasst werden, dass *vmail* darin schreiben darf:

```
linux:~ # chown vmail.mail /var/lib/dovecot
linux:~ #  chmod 0775 /var/lib/dovecot
linux:~ #  chown -R .vmail /var/spool/mail
```

Damit Postfix weiß, auf welchem Weg es Mails lokal zuzustellen hat, muss dies in Abhängigkeit zur Empfänger-Domain bekannt gemacht werden. Genutzt werden für diesen Zweck entsprechende Lookup-Tables. Hintergrund ist hier, dass Postfix über die Empfänger keine Kenntnis hat oder braucht. Es bekommt einfach nur gesagt, welche Domains lokal „relayed“ werden und wer sich um die Zustellung kümmert. Für beide Zwecke gibt es mit *relay_domains* und *transport_maps* eigene Lookup-Tables. Da in unserem Fall beide Tables den gleichen Inhalt haben, werden Sie an die selbe Datenquelle gebunden. Vorgenommen wird diese Konfiguration wieder in der Datei:

```
/etc/postfix/main.cf
```

```
# Relay Domains und Transportwege-Regelungs Tabelle bekannt machen
relay_domains = hash:/etc/postfix/relay
transport_maps = hash:/etc/postfix/relay
```

Die in der Konfiguration benannte Datei muss manuell angelegt und mit folgendem Inhalt versehen werden:

```
# for relaying domain
# domain.de OK
invis-net.loc   dovecot:
```

Darin enthält Spalte 1 die Domain und Spalte 2 den zugehörigen Transportweg. Letzterer muss natürlich in der Datei

```
/etc/postfix/master.cf
```

noch angelegt werden:

```
## Dovecot Transport -- invis-server.org -- Stefan Schaefer
dovecot   unix  -       n       n       -       -       pipe
    flags=DRhu user=vmail:vmail argv=/usr/lib/dovecot/deliver
    -f $sender -d $recipient
```

Benannt werden hier vor allem der Benutzer unter dessen Kennung der MDA (deliver) ausgeführt wird, der zu verwendende Delivery-Agent selbst, Absender- und Empfängeradresse sowie ein paar Flags, die das Verhalten des Delivery-Agents beeinflussen.

Letzte Etappe - Dovecot

Ab invis-Server Version 7.1-R3 kommt Dovecot in Version 2.1 zum Einsatz. Erwähnenswert ist das, da

sich an dessen Konfiguration mit Einführung von V. 2.x einiges geändert hat. Einerseits betrifft dies Namen von Konfigurationsoptionen und andererseits den Aufbau der Konfiguration an sich. Unter openSUSE wurde die Konfiguration der Übersicht halber in mehrere Dateien gesplittet. Im Verzeichnis „/etc/dovecot“ ist nach wie vor eine (wenn auch stark geschrumpfte) Datei „dovecot.conf“ zu finden. Sie dient quasi als Hauptkonfigurationsdatei in die alle weiteren Dateien inkludiert werden. Ähnlich wie das bereits beim Apache-Webserver der Fall ist.

Alle weiteren Dateien sind logisch unterteilt im Verzeichnis „/etc/dovecot/conf.d“ zu finden.