

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: @
sOARRecord: 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: dlzSOARRecord
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 sinnvoll, 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}
    {password} 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 Leerzeichen 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 dann funktioniert, wenn das zugrunde liegende System Multithreading-fähig ist. Bei modernen Linux-Systemen ist dies in der Regel der Fall.

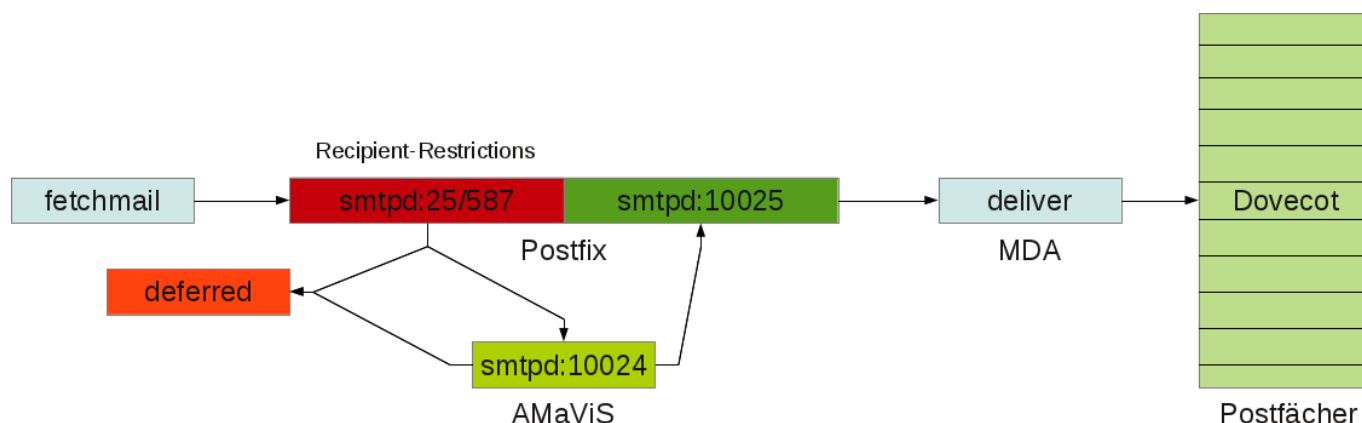
Postfix, Amavis & Dovecot

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

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.

From:

<https://wiki.invis-server.org/> - **invis-server.org**

Permanent link:

<https://wiki.invis-server.org/doku.php?id=kb&rev=1325860922>

Last update: **2012/01/06 14:42**

