

3.2 Emaildienste

322.2 Gewichtung: 2

Beschreibung: Die Kandidaten sollten Erfahrung und Wissen in den Sicherheitsmerkmalen für die Anwendung und Konfiguration von Postfix Emaildiensten besitzen. Ein Bewusstsein für die sicherheitsrelevanten Eigenschaften von Sendmail ist ebenfalls gefordert, jedoch nicht die Konfiguration.

Wichtige Wissensgebiete:

- Postfix Sicherheit: zentrale Konfiguration
- Absichern von Sendmail
- chroot Umgebungen

Liste benutzter Dateien, Begriffe und Werkzeuge:

- /etc/postfix/
 - TLS
-

3.2.1 Postfix

3.2.1.1 main.cf

Die Datei main.cf ist die Hauptkonfigurationsdatei von Postfix, in der die Einstellungen für den laufenden Betrieb des Emaildienstes vorgenommen werden.

postconf -d zeigt nur die voreingestellten (kompilierten) Parameter an.

postconf -n zeigt die manuellen Einträge in der main.cf an! Also die gültigen aktuellen Werte.

In dieser Datei werden viele sicherheitsrelevante Einstellungen gesetzt, zum Beispiel für die Verwendung von Cyrus SASL als Authentifizierung.

Das folgende Beispiel entstammt der Dokumentation von Postfix:

```

## TLS Settings
smtp_tls_loglevel = 1
#
# smtp_enforce_tls = yes
# Above is commented because doing it site by site below
smtp_tls_per_site = hash:/etc/postfix/tls_per_site
smtp_use_tls = yes
smtp_tls_CAfile = /etc/postfix/cacert.pem
smtp_tls_cert_file = /etc/postfix/FOO-cert.pem
smtp_tls_key_file = /etc/postfix/FOO-key.pem
smtp_tls_session_cache_database = btree:/var/run/smtp_tls_session_cache

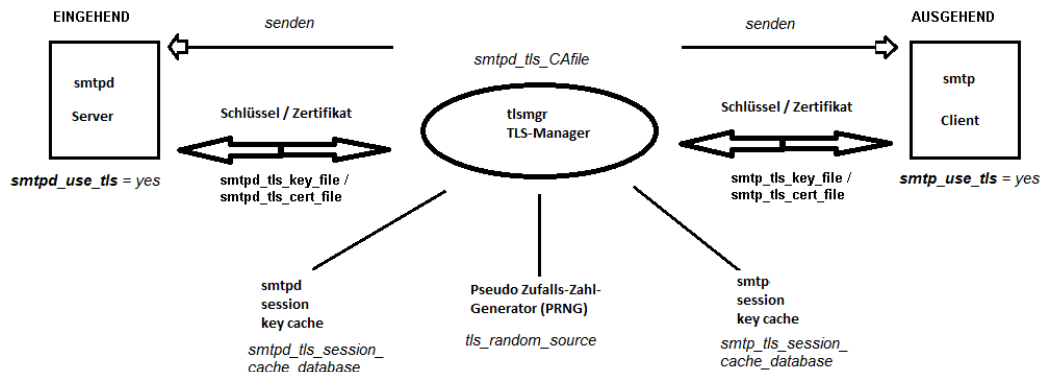
smtpd_tls_CAfile = /etc/postfix/cacert.pem
smtpd_tls_cert_file = /etc/postfix/FOO-cert.pem
smtpd_tls_key_file = /etc/postfix/FOO-key.pem
smtpd_tls_received_header = yes
.....smtpd_tls_session_cache_database =
    btree:/var/run/smtpd_tls_session_cache
smtpd_use_tls = yes
tls_random_source = dev:/dev/urandom

## SASL Settings
# This is going in to THIS server
smtpd_sasl_auth_enable = no
# We need this
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtpd_sasl_local_domain = $myhostname
smtp_sasl_security_options = noanonymous
#smtp_sasl_security_options =
smtp_sasl_tls_security_options = noanonymous
smtpd_sasl_application_name = smtpd
# Disable DNS Lookups
disable_dns_lookups = yes
#
# Great New feature Address Mapping
smtp_generic_maps = hash:/etc/postfix/generic
transport_maps = hash:/etc/postfix/transport

```

Die Datei main.cf wird von oben nach unten abgearbeitet, so dass die letzten gefundenen Werte genommen werden.

Sie sehen hier, wie die TLS-Zertifikate im Ordner /etc/postfix abgelegt werden. Man beachte, dass "*.pem" files jeweils doppelt vorkommen, und dass ein Unterschied zwischen "smtp" und "smtpd" besteht. Einmal für Client-Konnektivität und zum anderen für den Server.



Graphik 11: Zertifikate mit Postfix

Die obige Grafik soll die TLS-Unterstützung von Postfix mit den jeweiligen TLS-Parametern in der main.cf verdeutlichen:

- Der smtpd-Server implementiert SMTP über TLS mit dem Server-Zertifikat und seinem öffentlichen Schlüssel serverseitig.
- Der smtp-Client implementiert SMTP über TLS mit dem Client-Zertifikat und seinem öffentlichen Schlüssel clientseitig.
- Der TLS-Manager auf dem Server verwaltet mit seinem privatem Schlüssel den Pseudo Zufallszahl-Generator (PRNG), sendet die TLS-Verarbeitung in die Server- und Clientprozesse und verwaltet die "TLS session key cache files".

3.2.1.2 SASL

Machen Sie sich mit drei wichtigen Begriffen zum Verständnis von SASL vertraut:

- a) **SASL (Simple Authentication and Security Layer)** ist eine Methode, die Authentifizierung für verbindungsorientierte Protokolle bereitstellt, und ist in RFC 4422 (früher RFC 2222) definiert.

SASL bietet einem Anwendungsprotokoll (LDAP, SSH, IMAP, POP3, SMTP und andere) eine standardisierte Möglichkeit der Aushandlung von Kommunikationsparametern.

In SASL sind folgende Mechanismen enthalten:

ANONYMOUS, CRAM-MD5PLAIN, **GSSAPI** (MIT Kerberos 5 oder Heimdal Kerberos 5) und DIGEST-MD5. Unterstützt werden: LOGIN, SRP ,NTLM, OTP und KERBEROS_V4

- b) **Die GSS-API** (Generic Security Service API) bietet zwei Sicherheitsdienste jenseits der traditionellen Authentifizierungsdienste (AUTH_DH, AUTH_SYS, und AUTH_KERB): Integrität und Geheimhaltung. Für Integrität sorgen **kryptografische Prüfsummen** zur Authentifizierung. Für Geheimhaltung sorgen - nach Verschlüsselung der Daten - **RPCSEC_GSS benutzende Anwendungen**, die verfahrensunabhängig spezifizieren, welchen Dienst Sie nutzen wollen.

Oft ist GSSAPI-Authentifizierung mit Kerberos-Authentifizierung ohne erneute Passwordeingabe gleichzusetzen. Um nur einige Beispiele in der Praxis zu nennen:

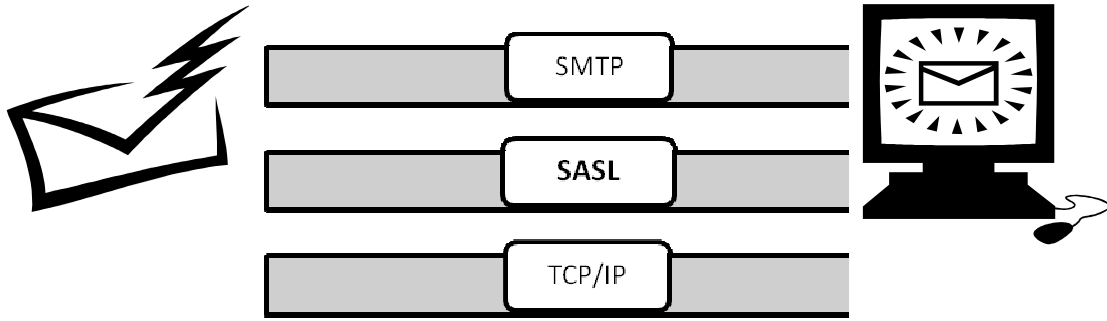
- IMAP- und POP3-Zugriff auf den Mailserver
- ssh-Zugang zu einem ssh-Server
- LDAP-Authentifizierung
- NFSv4 mit Kerberos

NFSv4 mit Kerberos betrachten Sie noch ausführlich in einem späteren Kapitel!

Sie benötigen für SASL mit Kerberos ein gültiges Kerberos-Ticket auf dem Rechner, an dem Sie gerade arbeiten, sowie eine Netzwerkverbindung zum Kerberos-Server, eventuell auch über ein VPN.

- c) **RPCSEC_GSS** ist eine Sicherheitsoption, die **oberhalb der GSS-API** für Netzwerkübertragungen liegt. RPCSEC_GSS gebrauchende Anwendungen können die GSS-API-Sicherheitsmerkmale nutzen und auch einen Sicherheitsmechanismus (wie RSA-PKI oder Kerberos), der mit der GSS-API zusammenarbeitet.

Weil SASL auf der Protokollebene arbeitet, muss SASL-Unterstützung meistens mit einer Option in die Dienstpakete (Postfix, LDAP, und viele mehr) kompiliert werden. Vor Postfix Version 2.3 wurde nur Cyrus SASL unterstützt. Heute hat Postfix eine Plug-In Architektur, die verschiedene SASL-Implementierungen unterstützt. Momentan unterstützt Postfix SMTP-Server Cyrus SASL und Dovecot SASL.



Graphik 12: SASL Layer

Ab Postfix Version 2.3 können Sie folgende Befehle eingeben, um herauszufinden, welche SASL-Implementierungen in Postfix kompiliert sind:

postconf -a SASL-Unterstützung für SMTP Server: z.B. cyrus und dovecot
postconf -A SASL-Unterstützung für SMTP+LMTP Client: z.B. cyrus

Sie wollen ein Benutzerpasswort für SASL-Dienste erzeugen:

```
saslpasswd2 -c -u `postconf -h mail.immer-linux.ok` benutzername
```

Dieser Befehl erzeugt das Passwort, das von Postfix benutzt wird, und in der Datenbankdatei `/etc/postfix/sasl_passwd.db` abgelegt wird.

Der Dienst für SASL ist der Daemon **saslauthd**, der "Cyrus SASL Passwort-Überprüfungsdienst".

Die Kommunikation zwischen dem Postfix SMTP-Server und dem saslauthd findet mit Cyrus SASL über ein UNIX-Domain Socket statt, welches in `/var/run/saslauthd/` liegt und auf Authentifizierungsanfragen wartet. Der Postfix SMTP-Server muss Lese- und Ausführrechte auf dieses Verzeichnis haben, ansonsten wird die Authentifizierung fehlschlagen. Einige Distributionen erfordern zusätzlich, dass der Benutzer postfix Mitglied einer speziellen Gruppe (z.B. sasl) ist.²⁰

Das folgende Beispiel konfiguriert die Cyrus-SASL-Bibliothek für saslauthd als Passwort-Verifizierungsdienst:

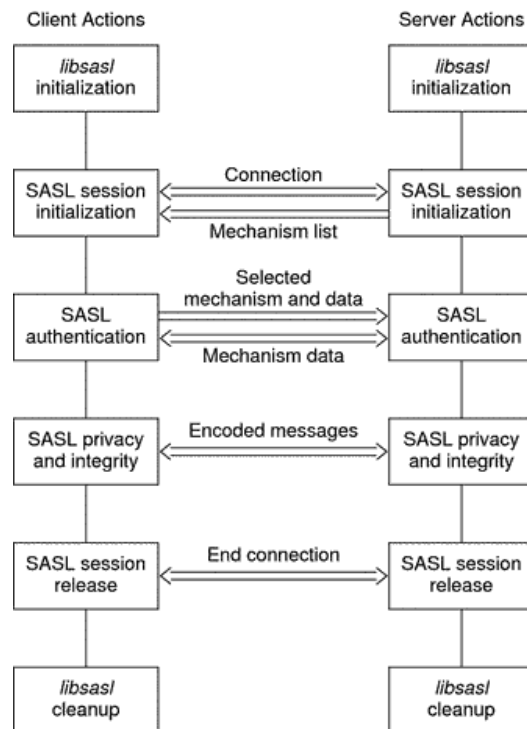
`/etc/sasl2/smtpd.conf:`

```
pwcheck_method: saslauthd  
mech_list: PLAIN LOGIN
```

Der Mechanismus "Plain" ist nicht so unsicher wie man denken könnte, denn er ist ja nur ein zusätzlicher Mechanismus, der die Verbindungsparameter abgleicht. Das benötigte Kerberos Ticket

²⁰ http://www.postfix.org/SASL_README.html

- oder das Zertifikat wie im Fall mit TLS/SSL - bekommen Sie ja auf eine andere Art und Weise. Beachten Sie den Parameter `smtp_sasl_tls_security_options = noanonymous` in der obigen Konfigurationsdatei `main.cf`



Quelle: <http://docs.sun.com/source/816-4863/images/sasl-flowchart-overview.gif>

Graphik 13: SASL Authentifizierung

Der `saslauthd` Server überprüft beispielsweise Passwörter gegen `/etc/shadow`, wenn er so gestartet wird:

```
saslauthd -a shadow und mit dem PAM-Framework saslauthd -a pam
```

Viele Fehlermeldungen lassen sich so mit SASL vermeiden!

Betrachten Sie nach diesem Exkurs die zweite wichtige Konfigurationsdatei von Postfix.

3.2.1.3 master.cf

Mit Postfix wird der Masterprozess master(8) gestartet, der verantwortlich ist für den Start von bestimmten Unterprozessen, die für verschiedene Teilaufgaben benötigt werden. Die Einstellungen hierfür werden in der Datei master.cf vorgenommen. Hier können Sie einstellen, welche Programme für welche Aufgaben gestartet werden müssen.

Diese Datei ist in der Regel schon passend vorkonfiguriert. Hier sind nur Eingriffe notwendig, wenn der Administrator Einstellungen für erweiterte Setups konfigurieren will. Als wichtigste Sicherheitseinstellung sei hier die Möglichkeit genannt, Postfix in einer **chroot-Umgebung** laufen zu lassen. Die Datei muss auch geändert werden, wenn Virens Scanner wie ClamAV oder Spam Filter wie Spamassassin eingebunden werden sollen.

Aufbau:

Jede Konfigurationszeile ist in maximal 8 Spalten unterteilt und beginnt immer in der ersten Spalte. Nach einem Zeilenumbruch wird sie nach einem Leerzeichen fortgesetzt (kein \ für einen Zeilenumbruch). Die Spalten werden ebenfalls nur durch Leerzeichen voneinander getrennt. "#" kommentiert wie immer aus. Einige der Spalten sind mit einem Standardwert voreingestellt. Ist in dieser Spalte ein "-" eingetragen, so gilt der Standardwert, der darunter in Klammern steht.

Service type private unpriv chroot wakeup maxproc command+args

Standardmäßig laufen also die Postfix-Dienste in einer chroot-Umgebung. Das könnte man zumindest annehmen, aber das unten stehende Bild einer master.cf zeigt, dass erstens die Tabelle nicht leer ist (wie man vermuten könnte, wenn nur die Vorgabewerte verwendet werden) und zweitens, dass viele Dienste explizit nicht in einer chroot-Umgebung laufen. In einer frischen Postfix-Installation läuft proxymap nicht in einer chroot-Umgebung! Wichtig für Sie ist noch zu wissen, dass nach Änderungen in der master.cf folgender Befehl abgesetzt werden sollte, um die Konfiguration sofort neu einzulesen: postfix reload

```

# = = = = =
# service type private unpriv chroot wakeup maxproc command + args
# name (yes) (yes) (yes) (never) (100)
# = = = = =
smtp inet n - y - - smtpd
pickup fifo n - n 60 1 pickup
cleanup unix n - n - 0 cleanup
qmgr fifo n - n 300 1 qmgr
rewrite unix - - n - - trivial-rewrite
bounce unix - - n - 0 bounce
defer unix - - n - 0 bounce
flush unix n - n 1000? 0 flush
proxymap unix - - n - - proxymap
smtp unix - - y - - smtp
relay unix - - y - - smtp
-o smtp_helo_timeout=5 -o smtp_connect_timeout=5
showq unix n - n - - showq
error unix - - n - - error

```

Graphik 14: master.cf

Sie sehen hier auch den Eintrag proxymap, der bekanntermaßen das Herz von postfix ist, und den wir im Folgenden genauer betrachten wollen:

3.2.2 Proxymap

Lassen Sie uns folgenden leicht abgeänderten Beitrag aus einer Postfix Mailingliste betrachten:

ist es korrekt, dass es jeweils genau einen proxymap Prozess für jede in der main.cf referenzierten "proxy:....." gibt? d.h. Änderungen in der master.cf unter MaxProcs für proxy werden ignoriert..?

Und jetzt zur eigentlichen Frage:

Hier laufen im Moment insgesamt 10 proxymap-Prozesse und jeder steht im top mit CPU-Werten zwischen 10% und 20%). Irgendwie ist mein System also zu 90% mit proxymap-Prozessen beschäftigt. ... was muss der abfragende Client bei einer SQL-Abfrage denn so groß "rechnen".... das macht doch der Datenbankserver Proxymap sollte nahezu keinerlei CPU-Zeit verbrauchen da er nur als einfacher Multiplexer für die anderen Prozesse arbeitet.

The proxymap(8) server provides service to multiple clients, and must therefore not be used for tables that have high-latency lookups.

.....Wie kann ich das System beschleunigen? Die Anzahl der Proxymap Server hochsetzen bzw. die Anzahl der Clientprozesse senken? Bei sehr vielen Abfragen pro Zeiteinheit den "max_use" Parameter zu erhöhen um die Anzahl der Forks zu senken?

In der master.cf steht in der proxy-Zeile bei max_use: "-". Das heißt doch, dass die maximale Anzahl von der eingestellten max_use bei smtpd beschränkt wird, oder?"

Als erstes erinnern Sie sich, was proxymap überhaupt ist und macht:

Proxymap liest/schreibt und verwaltet eine Tabelle mit Postfix-Prozessen und läuft unter der Kontrolle des Postfix-Masterservers. Die allgemeine Konfiguration erfolgt dagegen über Parameter in der main.cf.

In der Man Page von Proxymap finden Sie auch den Sinn der Proxymap-Dienste:

- **Chroot-Restriktionen umgehen.** Es wäre beispielsweise unsicher eine Kopie des Passwortes für einen SMTP-Server, der in einer chroot-Umgebung läuft, in der chroot-Umgebung zusätzlich abzulegen.
- **Die Konsolidierung offener Lookup-Tabellen** (zur Umadressierung und Filtern von Mails) durch das Teilen einer offenen Tabelle mit mehreren Prozessen.
- **Single-Update-Funktionalität** für Lookup-Tabellen, d.h. Schreibzugriff für Tabellen, die keine Unterstützung für mehrfachen Schreibzugriff bieten (hauptsächlich Dateien).

In diesem Beispiel wurde in der Mailingliste Folgendes empfohlen:

- Anzahl der Lookups pro Mailtransaktion klein halten
- diese Abfragen so effizient wie möglich gestalten
- Das ganze über eine passende Anzahl an "proxymap-Servern" verteilen.

Sicherheit der Dienste

Wahrscheinlich hatte der Kollege von der Mailingliste hauptsächlich ein mysql-Problem: max_use: "-" ist standardmäßig "100" und bietet also noch genügend Spielraum.

Uns interessieren aber hauptsächlich die **Sicherheitsaspekte** von proxymap:

- Proxymap in einer chroot-Umgebung schränkt die **Nutzbarkeit** stark ein, da nur Tabellen in einer chroot-Umgebung geöffnet werden können.
- Proxymap ist **kein vertrauter Prozess**, der nicht für sensitive Daten (wie Benutzer- und Gruppen-IDs, Mailbox Dateinamen oder Verzeichnisnamen usw.) verwendet werden sollte.
- Ab Postfix Version 2.2 erkennt proxymap automatisch **sicherheitskritische Anfragen** und öffnet die Tabelle direkt. Dadurch können dieselben Sicherheitseinstellungen in der main.cf für sensitive und nicht-sensitive Prozesse verwendet werden.
- Dateien, die von Postfix beschrieben werden können, sollten in einem **eigenen Verzeichnis**, das nur vom Systembenutzer postfix beschrieben werden kann, abgelegt werden. Auch sollten die Dateien und Verzeichnisse **nicht im Besitz von root-Rechten** sein.

Postfix wurde speziell für maximale Sicherheit entworfen. Postfix ist im Gegensatz zu Sendmail kein einheitliches Programm, sondern besteht aus vielen einzelnen Programmteilen, die nur mit den unbedingt notwendigen Rechten laufen. Kein Prozess von Postfix, der mit dem Netzwerk kommuniziert, hat root-Rechte!

3.2.3 Sendmail

In den Topics für die LPI-303-Prüfung steht ausdrücklich, dass die genaue Konfiguration von Sendmail nicht zu den Prüfungsthemen gehört. Trotzdem sollten Sie wissen, dass die Einstellungen für sendmail in der /etc/mail/sendmail.mc gehandhabt werden. Wir beschränken uns aber auf allgemeine Sicherheitsüberlegungen zu Sendmail.

Wenn Sie nach "sendmail security" googeln, treffen Sie auf eine bemerkenswerte Überschrift "Sendmail absichern, indem man es abschaltet"!

Sendmail ist im Allgemeinen ein sehr kritischer Prozess in Linux-Systemen. Die Hauptaufgabe von Sendmail ist es auf Port 25/tcp auf eingehende Mails zu horchen. Dies macht das System offen für alle möglichen Remote-Angriffe.

Grundsätzlich gilt, dass Sendmail mit root-Rechten laufen muss! Zwar gibt es Überlegungen, sendmail unter einer eigenen Benutzerkennung und Gruppe laufen zu lassen. Dies bereitet aber weitere Probleme. Außerdem löst dies nicht die eigentlichen Probleme, zumal Sie sendmail selbstverständlich in einer **chroot-Umgebung** laufen lassen sollten.

Weitere Sicherheitsaspekte von Sendmail:

- Sendmail unter **xinetd** laufen lassen
- **GSSAPI** verwenden (SASL)
- Ausschalten von VRFY und EXPN

NOVRFY verhindert Telnet-Zugriff auf den Server und überprüft, ob eine Emailadresse gültig ist. Der Zweck von VRFY ist es, einem Remote-Server zu ermöglichen, eine Emailüberprüfung durchzuführen, bevor eine E-Mail gesendet wird.

NOEXPN verhindert Telnet-Zugriff auf den Server und dass jemand eine Liste mit Aliasnamen der Emailempfänger erhält.

Dafür muss folgende Einstellung in der `sendmail.mc` gemacht werden:

```
# privacy flags
PrivacyOptions=authwarnings,novrfy,noexpn
```

- **TLS** benutzen (sendmail muss mit STARTTLS kompiliert sein) und folgende Einstellungen müssen in der Datei `sendmail.cf` für `sendmail.cf` gemacht werden:

```
define('confCACERT_PATH', '/etc/mail/certs')dnl
define('confCACERT', '/etc/mail/certs/CAcert.pem')dnl
define('confSERVER_CERT', '/etc/mail/certs/MYcert.pem')dnl
define('confSERVER_KEY', '/etc/mail/certs/MYkey.pem')dnl
define('confCLIENT_CERT', '/etc/mail/certs/MYcert.pem')dnl
define('confCLIENT_KEY', '/etc/mail/certs/MYkey.pem')dnl
define(`confTLS_SRV_OPTIONS', '')
```

und auch für die Access Map:

```
dnl FEATURE('access_db')
```

In die Datei `/etc/mail/access` tragen Sie beispielsweise Folgendes ein:

```
CERTIssuer:/C=DE/ST=Berlin/L=Berlin/O=immer-linux.ok/CN=immer-
linux.ok+20CA/Email=postmaster@example.org RELAY
```

Die gesicherte Übertragung des Emailverkehrs ist natürlich nur eine Sache. Wie die Emails auf einem Server gespeichert werden, das ist eine ganz andere. Werden sie auch verschlüsselt abgespeichert? Meistens nicht. GnuPG stellt - wie schon ausführlich behandelt - eine erweiterte, sicherere Variante der PGP-Verschlüsselung dar.

Zwar nicht Prüfungsthema, aber unbedingt sollte auch ein Filter wie Spamassassin und ein Antivirusprogramm wie ClamAV verwendet werden.