Empfehlungen für den Einsatz von Transportverschlüsselung zwischen Mailservern

11.08.2017

DFN CERT®

Dokument-Informationen	
Autoren	Antje Bendrich, Jürgen Brauckmann, Lars Weber, Marc Thomas, Stefan Kelm
Dateiname	smtp-transportverschluesslung.pdf
letzte Bearbeitung	15. August 2017
Seitenzahl	19

Version	**Datum**	**Autor(en)**	**Änderungen**
0.1	1.1.1970	Marc Thomas	Formatvorlage
0.2	28.11.2016	Marc Thomas	Struktur, Inhalt
0.3	07.12.2016	Lars Weber	Kapitel 2.4, 3
0.4	24.02.2017	Marc Thomas	Kapitel 4
0.5	27.03.2017	Stefan Kelm	QA
0.6	28.03.2017	Antje Bendrich	QA
0.7	01.06.2017	Jürgen Brauckmann	QA
0.8	13.06.2017	Lars Weber	Kapitel 2.4.3
0.9	11.08.2017	Lars Weber	Kapitel 2.4 überarbeitet

Inhaltsverzeichnis

1. Einleitung

Moderne Bürokommunikation läuft zum großen Teil über E-Mail. Die Inhalte der E-Mails sowie die dabei entstehenden Metadaten enthalten sensible Daten. Hierbei handelt es sich nicht nur um z. B. Firmengeheimnisse sondern vor allem auch um personenbezogene Daten nach dem Bundesdatenschutzgesetz (BDSG). Die sichere und ordnungsgemäße Konfiguration eines Mailservers bedarf dabei viel Erfahrung und Sorgfalt. Dieses Dokument soll Empfehlungen geben sowie auf einfache Art und Weise erklären, wie ein Mailserver konfiguriert werden kann, um die Kommunikation zwischen Mailservern abzusichern. Die Kommunikation zwischen Mailservern und Endnutzer-Software wie z.B. Thunderbird wird nicht berücksichtigt. Es wird dabei speziell auf die Konfiguration der beiden geläufigsten Mailserver-Software Postfix und Exim eingegangen. Zusätzlich wird allgemein beschrieben, worauf bei der Konfiguration zu achten ist. So lassen sich die Beschreibungen ggf. auch auf andere Mailserver übertragen.

In Kapitel 2 wird im Detail auf die sichere Konfiguration von Mailservern eingegangen, so dass diese ihre Daten verschlüsselt austauschen können. In Kapitel 3 wird auf das ergänzende Protokoll DANE eingegangen, welches die Sicherheit erhöhen kann und insbesondere bei der Authentifizierung des Mailservers hilft. Wenn Sie sich bereits mit TLS-Konfiguration in Mailservern sowie mit DANE auskennen, können Sie direkt zu Kapitel 4 springen. Dort finden Sie fertige Konfigurationsbeispiele für die einzelnen Mailserver. Der Anhang enthält komplette Konfigurationsbeispiele der entsprechenden Konfigurationsteile.

Die Konfigurationen sind teilweise dem Bettercrypto Projekt entnommen. Dieses Paper berücksichtigt ebenfalls Erkenntnisse aus BSI TR-03108-1 sowie BSI TR-03116-4.

2. TLS-Konfiguration

Die Kommunikation zwischen zwei Mailservern funktioniert bekanntermaßen mittels des Simple Mail Transport Protokolls (SMTP). Dieses Protokoll, definiert 1982, kennt ursprünglich keine Verschlüsselung. SMTP wurde erst 1999 im RFC2487 mit der STARTTLS Erweiterung um diese Möglichkeit erweitert. Wie bei allen Protokollen, die eine STARTTLS-Erweiterung nutzen, wird der Beginn der Kommunikation immer unverschlüsselt durchgeführt und dann mittels des STARTTLS-Befehls auf Verschlüsselung umgeschaltet.

Mit Aufrufen des STARTTLS-Befehls beginnt das Aushandeln der TLS-Verschlüsselung. Damit diese überhaupt erfolgen kann, benötigen beide Mailserver ein X.509-Zertifikat. Da jeder Mailserver sowohl als Server als auch als Client agieren kann, benötigen diese Zertifikate die Extended Key Usages „TLS Web Server Authentication“ und „TLS Client Authentication“. Bei der DFN-PKI existiert extra für diese Zweck ein Zertifikatsprofil „Mailserver“. Zertifikate, die mit diesem Profil beantragt wurden, erfüllen alle notwendigen Voraussetzungen.

Es gibt eine Reihe von Konfigurationsoptionen, um die TLS Verschlüsselung sicher zu betreiben. Ein Teil der Optionen ändert sich immer wieder, z. B. durch neue Erkenntnisse der Sicherheitsforscher oder durch neue Standards. Außerdem sind die Werte oft ein Kompromiss aus Sicherheit und Kompatibilität mit verschiedenen Clients.

Es müssen zunächst die benutzbaren TLS-Protokolle konfiguriert werden, dies wird in 2.1 beschrieben. Zusätzlich zu den benutzbaren TLS-Protokollen müssen die benutzbaren Algorithmen konfiguriert werden, dies wird in 2.2 behandelt. Zum Schluss müssen noch einige allgemeine Optionen konfiguriert werden, darauf wird in 2.3 eingegangen. Alle weiteren Konfigurationsoptionen, welche für eine sichere Konfiguration des Mailserver nötig sind, werden in 2.4 behandelt.

2.1. SSL/TLS-Protokolle

SMTP mit STARTTLS nutzt zur Kommunikation das TLS-Verschlüsselungsprotokoll bzw. den Vorgänger SSL. Die Protokolle selbst wiederum nutzen verschiedene Algorithmen, Blockverschlüsselungmodi und Schlüsselaustauschverfahren. Es gibt seit vielen Jahren immer wieder Forschungarbeiten, die Probleme mit dem SSL- und TLS-Protokoll aufzeigen. Insbesondere sind dabei die Protokolle SSLv2 und SSLv3 betroffen, die nicht mehr verwendet werden sollten, da sie keinerlei Sicherheit bieten. Nach aktuellem Stand sollen nur die TLS Protokolle verwendet werden. Es gibt momentan drei nutzbare Varianten von TLS: Die Versionen TLS 1.0, TLS 1.1 und TLS 1.2. TLS 1.3 befindet sich zur Zeit noch in der Entwicklung. Es gibt bereits Angriffspunkte sowohl auf TLS 1.0 als auch auf TLS 1.1, welche aber bei sorgfältiger Konfiguration nicht praktisch anwendbar sind. Die sicherste real nutzbare Version ist TLS 1.2, wobei auch dabei die Auswahl der Algorithmen, Modi etc. noch eingeschränkt werden muss. Darauf wird in 2.2 näher eingegangen.

Von einer alleinigen Verwendung von TLS 1.2 ist momentan allerdings auch abzuraten, da es noch zahlreiche Mailserver gibt, die TLS 1.2 nicht unterstützen. Daher sollten alle TLS-Protokolle angeboten werden, aber keine SSL Protokolle mehr.

Entsprechend muss die TLS-Protokoll Konfiguration des Mailserver angepasst werden. Diese Konfigurationen sind normalerweise so aufgebaut, dass immer alle Protokolle erlaubt sind und explizit einzelne Protokolle deaktiviert werden, hier also SSLv2 und SSLv3.

Bei Postfix werden die Protokolle über mehrere Konfigurationsoptionen eingestellt, sowohl für den SMTP-Client (smtp) und den SMTP-Server (smtpd). Die Konfigurationsoptionen mit „mandatory“ sind nur wichtig, wenn die Verschlüsselung mit einem anderen Mailserver erzwungen wird, darauf wird genauer in 2.4 eingegangen.

```
smtp_tls_mandatory_protocols = !SSLv2, !SSLv3
smtp_tls_protocols = !SSLv2, !SSLv3

smtpd_tls_mandatory_protocols = !SSLv2, !SSLv3
smtpd_tls_protocols = !SSLv2, !SSLv3
```

Bei Exim ist die Konfiguration davon abhängig, ob Exim die SSL/TLS-Implementierung OpenSSL oder GnuTLS nutzt. Dies kann entweder durch das Werkzeug ldd ermittelt oder einfach durch Probieren festgestellt werden. Bei Exim (das OpenSSL nutzt) müssen die folgenden Optionen gesetzt werden:

```
openssl_options = +no_sslv2 +no_sslv3
```


Bei Exim, das gegen GnuTLS gelinkt ist, erfolgt die Einstellung mit der Ciphersuite-Konfiguration, auf die in 2.2 eingegangen wird. Der entscheidende Teil ist der hinter `NORMAL`. Da GnuTLS SSLv2 nie implementiert hatte, muss dies hier nicht gesondert deaktiviert werden.

```
tls_require_ciphers = NORMAL:!VERS-SSL3.0
```

2.2. SSL/TLS-Algorithmen

Um Daten mit TLS sicher übertragen zu können, muss man eine sichere Auswahl an Ciphersuites für einen Mailserver festlegen. Eine Ciphersuite ist eine Kombination aus Schlüsselaustauschverfahren, Chiffre inkl. Modi und Message Authentication Code, z. B. TLS_ECDHE_RSA_WITH_AES_256_GCM

Nach aktuellem Stand der Forschung im Bereich der Cryptoalgorithmen und -verfahren gelten insbesondere RC4 und 3DES sowie alle Export Cipher und auch die Hash Verfahren MD5 und SHA1 als unsicher. Daten, die mit diesen Chiffren und Hashes verschlüsselt sind, lassen sich teilweise innerhalb von wenigen Minuten bzw. Stunden ohne Kenntnis des geheimen Schlüssels entschlüsseln. Ähnliches gilt für einige Blockverschlüsselungsmodi. Hier gilt Cipher Block Chaining (CBC) als unsicher und soll nicht mehr genutzt werden. Moderne Modi, die erst mit TLS 1.2 engeführt wurden, gelten dagegen derzeit als sicher. Zu nennen sind hier u. a. Galois Counter Mode (GCM).

Moderne Ciphersuites haben die Sicherheitseigenschaft `Perfect Forward Secrecy`. Derartige Ciphersuites erzeugen regelmäßig neue Sitzungsschlüssel zwischen den Kommunikationspartnern. Dadurch ist selbst Datenverkehr, welcher aufgezeichnet wurde und für den der geheime Schlüssel des Zertifikates bekannt ist, nicht entschlüsselbar.

Es wird immer eine Menge von Ciphersuites für eine TLS-Kommunikation konfiguriert. Die Kommunikationspartner wählen dabei eine Ciphersuite aus, welche beide Partner unterstützen. Die Menge wird durch einen Textstring repäsentiert, der die Reihenfolge festlegt, in der die Ciphersuites ausgewählt werden sollen. Dieser sogenannte Priority String (z. B. `ALL:!SSLv3:!LOW:!RC4:!MD5`) muss regelmäßig überprüft werden und entsprechend der aktuellen Forschungsergebnisse sowie Regelungen von Behörden und Herstellervorgaben angepasst werden.

Die DFN-PKI veröffentlicht entsprechende Priority Strings für Webserver z. B. auf der Webseite `blog.pki.dfn.de`. Auch das bereits in der Einleitung erwähnte Bettercrypto-Projekt pflegt entsprechende Priority Strings, dort auch speziell für Mailserver.

Die Priority Strings für Mailserver sollten nicht so restriktiv gehalten werden wie für Webserver, da Mailserver teilweise sehr lange im Einsatz oder auch schlecht wartbar (z. B. Appliances) sind, so dass neue Algorithmen bzw. Entwicklungen in diesem Bereich nicht so schnell anwendbar sind. Daher ist eine konservativere Konfiguration zu empfehlen, um mit möglichst vielen Mailservern verschlüsselt kommunizieren zu können.

Es gibt verschiedene Aliase für Priority Strings, die eine definierte Menge von Ciphersuites repräsentieren, z. B. `high`, `medium`, `all`, `AES128`, `AES128`. Die genaue Bedeutung dieser Aliase muss der Dokumentation der eingesetzten Crypto-Bibliothek entnommen werden, für OpenSSL `https://www.openssl.org/docs/man1.0.1/apps/ciphers.html` sowie für GnuTLS `https://`

`www.gnutls.org/manual/html_node/Supported-ciphersuites.html` und `https://gnutls.org/manual/html_node/Priority-Strings.html`.

Postfix nutzt nur OpenSSL, daher sollten die folgenden Konfigurationsoptionen mit den Priority Strings genutzt werden:

```
smtpd_tls_mandatory_ciphers=high
tls_high_cipherlist=EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:EECDH+aRSA+SHA256
    :EECDH:+CAMELLIA128:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!
    PSK:!DSS:!RC4:!SEED:!IDEA:!ECDSA:kEDH:CAMELLIA128-SHA:AES128-SHA
```

Die Konfiguration für Exim unterscheidet sich je nach verwendeter Crypto-Bibliothek. Für OpenSSL:

```
tls_require_cipher = EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:EECDH+aRSA+
    SHA256:EECDH:+CAMELLIA128:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!
    EXP:!PSK:!DSS:!RC4:!SEED:!IDEA:!ECDSA:kEDH:CAMELLIA128-SHA:AES128-SHA
openssl_options = +all +no_sslv2 +no_sslv3 +no_compression +
    cipher_server_preference
```

und für GnuTLS

```
tls_require_cipher = NORMAL:!VERS-SSL3.0:!ECDHE-ECDSA:!3DES-CBC:!AES-128-CBC
    :!AES-256-CBC:!ARCFOUR-128:!CAMELLIA-128-GCM:!CAMELLIA-256-GCM:!COMP-
    DEFLATE:!DHE-DSS:!SIGN-DSA-SHA1:!SIGN-RSA-SHA1:!VERS-DTLS1.2:!VERS-DTLS1
    .0:!SIGN-ECDSA-SHA384:!SIGN-ECDSA-SHA512:!SIGN-ECDSA-SHA224:!SIGN-ECDSA-
    SHA1:+COMP-NULL
```

2.3. SSL/TLS-Parameter

Es gibt neben den TLS-Protokollen und den Ciphersuites noch weitere ergänzende Parameter zu konfigurieren. Dazu zählt z. B. die Verwendung eines eigenen Diffie-Hellmann-Parameters. Die falsche Konfiguration bzw. die Nutzung der von Betriebssystem oder Softwarehersteller gewählten Standardwerte kann die Sicherheit der verschlüsselten Kommunikation schwächen.

Zunächst sollte ein eigener DH-Parameter-Satz (DH) mit möglichst großer Länge erzeugt werden:

```
openssl dhparam -out dh2048.pem 2048
```

Diese Datei muss in der Konfiguration des Mailserver berücksichtigt werden, für Postfix mittels:

```
smtpd_tls_dh1024_param_file = /etc/postfix/dh2048.pem
```

Für Exim

```
tls_dhparam = /etc/exim/dh2048.pem
```


Eine weiterer Parameter betrifft die Kompression der Daten vor der TLS-Verschlüsselung. Es hat sich gezeigt, dass die TLS-Kompression Angriffe auf die Verschlüsselung ermöglicht. Daher sollte sie abgeschaltet werden. Allerdings ist dies in älteren Versionen der Mailserver Software oft nicht möglich. Bei Systemen mit OpenSSL wird eine Version OpenSSL $\geq$ 1.0 sowie Postfix ab Version 2.11 oder Exim 4.80 (OpenSSL) benötigt. Bei Systemen, die GnuTLS verwenden, muss mindestens Version 2.2.0 installiert sein.

Die entsprechende Konfigurationsoption für Postfix sieht folgendermaßen aus:

```
tls_ssl_options = NO_COMPRESSION
```

Für Exim mit OpenSSL Bibliothek:

```
openssl_options = no_compression
```

Für Exim mit GnuTLS muss nichts extra konfiguriert werden, da hier standardmäßig die TLS-Kompression deaktiviert ist. Damit diese nicht wieder explizit eingeschaltet wird, muss darauf geachtetet werden, dass `tls_require_ciphers` nicht die Werte `COMP-DEFLATE` oder `COMP-ALL` enthält.

2.4. MTA-Konfiguration

Grundsätzlich lässt sich die Unterstützung von Transportverschlüsselung im MTA einfach konfigurieren, sowohl für den SMTP-Client als auch für den SMTP-Server. Für den SMTP-Client ist die Unterstützung i. d. R. auch per Voreinstellung aktiv.

In diesem Abschnitt wird auf die wichtigsten Konfigurations-Optionen zur Aktivierung der TLS-Unterstützung in Postfix bzw. Exim eingegangen. Zusammen mit den in den vorigen Abschnitten vorgestellten Optionen ergeben sich dann die im Anhang dokumentierten Beispielkonfigurationen.

Erforderliche Zertifikate für Mailserver sind übrigens für DFN-Anwender kostenlos erhältlich sind. Nähere Informationen dazu finden sich unter https://www.pki.dfn.de/

2.4.1. Grenzen der Konfiguration

Grundsätzlich ist die Realisierung einer verschlüsselten Verbindung und deren Güte immer abhängig von der Unterstützung durch die Gegenstelle. Nur wenn beide Seiten, sowohl SMTP-Server als auch SMTP-Client, verschlüsselte und authentifizierte Verbindungen erzwingen, kann davon ausgegangen werden dass eine Mail vertraulich übermittelt wird. Ohne das Erzwingen efolgt bei Problemen mit der Verschlüsselung oder Authentifizierung ggf. eine unverschlüsselte Übertragung zu einer ggf. nicht authentifizierten Gegenstelle. So soll sichergestellt werden, dass die Mail auf jeden Fall zugestellt wird. Für dieses Verhalten sich der Begriff „Opportunistic TLS“ etabliert.

Das Aushandeln einer verschlüsselten Verbindung per STARTTLS erfolgt im Klartext. Daher kann ein Angreifer die Aushandlung so manipulieren, dass keine Unterstützung von TLS signalisiert wird, obwohl beide Seite TLS unterstützen („Downgrade Attack“). Wegen des Opportunistic TLS erfolgt dann die Übertragung der Mail unverschlüsselt.

Ein weiteres Problem besteht darin, dass ein SMTP-Client den zuständigen Mail-Server für Mail-Adressen einer Domain üblicherweise über die im DNS hinterlegten MX-Records der Domain ermittelt. Die zuständigen Mail-Server können beliebige Hostnamen tragen und somit ohne weiteres auch ein gültiges Zertifikat vorweisen. Da sich die Zuständigkeit der Mail-Server für eine Domain auf die MX-Einträge stützt, könnte ein Angreifer über DNS Cache-Poisioning die Mails für eine Domain „einfach“ über andere Mail-Server umleiten.
Die Verbindung wäre dann zwar immer noch verschlüsselt, die Mails erreichen dann aber nicht (ausschließlich) das richtige Ziel.

Die Problematik der Authentifizierung von zuständigen Mail-Servern und die Möglichkeit von Downgrade-Angriffen kann mit DANE TLSA-Records (siehe Kapitel 3) umgangen werden.

2.4.2. Postfix

Für den Postfix SMTP-Server beginnen die Konfigurations-Optionen mit der Zeichenfolge `smtpd_`, für den SMTP-Client mit `smtp_`. Beide werden wie gewohnt in `main.cf` konfiguriert.

Mit der folgenden Konfigurations-Option wird die TLS-Unterstützung für den Postfix-Client aktiviert. Mails werden dann über eine verschlüsselte Verbindung verschickt, wenn die Gegenstelle Verschlüsselung unterstützt. Es ist hierzu keine Konfiguration von Zertifikaten erforderlich.

```
smtp_tls_security_level  = may
```

Werden ein Zertifikat und ein Schlüssel für den SMTP-Client konfiguriert, kann der Client auf Anfrage durch die Gegenstelle ein Zertifikat vorweisen. Wenn eine Gegenstelle die Gültigkeit der Zertifikats-Kette des Clients überprüfen soll, sollten die Zertifikate der Intermediate-CAs zusammen mit dem Client-Zertikat in eine Datei geschrieben werden. Die Root-CA kann im Prizip auch mit eingetragen werden, das ist aber wenig sinnvoll. Die Gegenstelle sollte eine Liste der von ihr vertrauten Root-CAs für die Prüfung eines Client-Zertifikats konfiguriert haben. Der Client sendet dann ggf. sein Zertifikat zusammen mit den Zertifikaten der Intermediate-CAs an die Gegenstelle. Mit folgendem Kommando werden die Zertifikate in eine Datei geschrieben. Die Reihenfolge ist dabei zu beachten.

```
$ cat server_cert.pem subsub_CA.pem sub_CA.pem [root_CA.pem] > mta-cert.pem
```

Die Konfiguration von Schlüssel und Zertifikat(-kette) für den SMTP-Client:

```
smtp_tls_key_file        = /etc/postfix/ssl/private/mta-key.pem
smtp_tls_cert_file       = /etc/postfix/ssl/certs/mta-cert.pem
```

Die Prüfung von Server-Zertikaten durch den SMTP-Client ist ebenfalls möglich. Dazu muss der SMTP-Client die CA-Zertifikate kennen, denen vertraut wird. Diese CA-Zertifikate werden im PEM-Format hintereinander in eine Datei geschrieben. Wenn die Datei nur die Root-CAs enthält, muss der Server der Gegenstelle natürlich neben seinem Server-Zertikat auch die Zertifikate der Intermediate-CAs mitliefern. Andernfalls kann die Prüfung der Gültigkeit der Zertifikatskette nicht erfolgen.

Werden die Intermediate-CAs nicht von der Gegenstelle geliefert, können diese ggf. mit in `CAfile.pem` aufgenommen werden. Das sollte aber eher die Ausnahme sein, da sonst neben den Root-CA Zertifikaten auch noch die Liste der Imtermediate-CA Zertifikate gepflegt werden

müsste. Der unten stehende Parameter `smtp_tls_scert_verifydepth` gibt an, bis zu welcher Tiefe ein Zertifikat als gültig anerkannt wird. Eine Tiefe von 1 bedeutet, dass das ausstellende Zertifikat direkt im CAFile konfiguriert ist.

```
smtp_tls_CAfile            = /etc/postfix/ssl/certs/CAfile.pem
smtp_tls_scert_verifydepth = 5
```

Da die oben die Option `smtp_tls_security_level = may` gesetzt wurde, wird die Verbindung auch bei fehlgeschlagener Prüfung aufgebaut.

Alternativ zu `smtp_tls_CAfile` können die CA-Zertifikate auch in einem Verzeichnis liegen und über die Option `smtp_tls_CApath` eingebunden werden. Es gilt analog das weiter unten für die Option `smtpd_tls_CApath` geschriebene.

Die folgenden Optionen schließlich setzen das Logging von TLS-Verbindungsinfomationen und konfigurieren die TLS-Parameter in geeigneter Weise (siehe Kapitel 2.3):

```
smtp_tls_loglevel        = 1
smtp_tls_protocols       = !SSLv2, !SSLv3
```

Für den Postfix-Server wird die TLS-Unterstützung mit der Konfigurations-Option `smtpd_tls_security_level` aktiviert. Postfix signalisiert einem Client die Unterstützung des STARTTLS-Verfahrens, so dass eine Verschlüsselung der Verbindung ausgehandelt werden kann. Die Option unterstützt auch restriktivere Parameter als `may`, allerdings sollte der Parameter `may` für einen öffentlichen MTA gewählt werden. Es ist i. d. R. nicht sichergestellt, dass alle Mail-Clients Verschlüsselung unterstützen.

```
smtpd_tls_security_level  = may
```

Für die TLS-Unterstützung im Server müssen Zertifikat und Schlüssel konfiguriert werden. Das können die selben Dateien wie für den SMTP-Client sein, ggf. mit den Intermediate-CA-Zertifikaten. Der Server sendet dann auf Anfrage durch den Client neben dem Server-Zertifikat die Liste der Intermediate-CAs an den Client. Dieser kann dann die Gültigkeit der Zertifikatskette prüfen. Der Client muss dazu dann natürlich die Root-CA als vertrauenswürdige CA konfiguriert haben.

```
smtpd_tls_key_file        = /etc/postfix/ssl/private/mta-key.pem
smtpd_tls_cert_file       = /etc/postfix/ssl/certs/mta-cert.pem
```

Für die Konfiguration der CA-Zertifikate stehen zwei Optionen zur Verfügung, `smtpd_tls_CApath` und `smtpd_tls_CAfile`. Die Konfiguration von CA-Zertifikaten ist für die Verschlüsselung einer Verbindung keine Voraussetzung. Die CA-Zertifikate werden für die Überprüfung von Client-Zertifikaten gebraucht. Die referenzierte Datei enthält die PEM-kodierten CA-Zertifikate.

```
smtpd_tls_CAfile          = /etc/postfix/ssl/certs/CAfile.pem
smtpd_tls_ask_ccert       = yes
```

Da diese Liste beim Start von Postfix in den Speicher geladen wird, sollte eine große Anzahl von CA-Zertifikaten eher über die Option `smtpd_tls_CApath` eingebunden werden. Die Option verweist dabei auf ein Verzeichnis, das PEM-kodierte CA-Zertifikate enthält, die als vertrauenswürdig erachtet werden. Zu jeden Zertifikat muss ein Hash-Link des Zertifikats existieren, wie er mit dem Tool `c_rehash` erzeugt wird. Da Postfix meistens in einer chroot-Umgebung läuft,

`/var/spool/postfix`, muss das Verzeichnis in der Umgebung vorhanden und korrekt über die Option referenziert sein.

Die folgenden Optionen aktivieren das Logging von TLS-Verbindungsinfomationen und konfigurieren die TLS-Parameter in geigneter Weise:

```
smtpd_tls_loglevel          = 1
smtpd_tls_protocols         = !SSLv2, !SSLv3
```

Verschlüsselung mit ausgewählten Gegenstellen

Mit dem Parameter `smtp_tls_policy_maps` kann eine Key/Value-Datei angegeben werden, über die festgelegt wird, wie sich Postfix zu ausgewählten Domains verbindet. Die Festlegung in dieser Datei überschreibt den weniger Domain-spezifischen Wert von `smtp_tls_security_level`.

```
smtp_tls_policy_maps        = hash:/etc/postfix/smtp_tls_policy_map
```

Sollen z. B. Mails für die Domain dfn-cert.de nur über verschlüsselte Verbindungen zugestellt werden, könnte die Datei `smtp_tls_policy_map` dazu wie folgt aussehen:

```
dfn-cert.de     secure
```

Postfix wird dann versuchen, den Next-Hop für die Domain anhand des präsentierten Zertifikats zu überprüfen. Für dfn-cert.de wäre der Next-Hop mail1.dfn-cert.de oder mail2.dfn-cert.de. Voraussetzung ist natürlich, dass der SMTP-Client die beiden Server direkt erreichen kann und nicht noch lokal ein weiterer Hop (wie z. B. ein zentrales Firmen-Relay) auf dem Weg liegt. Dann müsste die Konfiguration auf diesen Firmen-Relay erfolgen.

2.4.3. Exim

Exim unterstützt per Default TLS-Verbindungen für den SMTP-Client, wenn Exim mit OpenSSL- oder GNUTLS-Unterstützung kompiliert wurde.

Die TLS-Optionen werden im Hauptabschnitt der Exim-Konfigurations-Datei eingetragen. Sie gelten für den Exim SMTP-Server. Für den Exim SMTP-Client werden die Optionen im entsprechenden Abschnitt konfiguriert, siehe weiter unten.

Damit Exim als SMTP-Server Verschlüsselung anbieten kann, müssen Zertifikat und Schlüssel konfiguriert werden. Außerdem muss Exim dem SMTP-Client signalisieren, dass STARTTLS unterstützt wird. Dazu dient die Option `tls_advertise_host`. Über die Option `log_selector` wird das Protokollieren von TLS-Informationen zu den Verbindungen aktiviert.

Wenn das Zertifikat des Exim-Servers durch eine Gegenstelle geprüft werden soll, müssen die CA-Zertifikate der Zertifizierungskette (außer der Root-CA) konfiguriert werden. Wie bei Postfix werden die CA-Zertifikate zusammen mit dem Server-Zertifikat in eine Datei geschrieben:

```
$ cat server_cert.pem subsub_CA.pem sub_CA.pem [root_CA.pem] > mta-cert.pem
```

Die Konfiguration für den SMTP-Server zusammengefasst:


```
log_selector = +tls_cipher +tls_peerdn
tls_advertise_hosts = *
tls_certificate = /etc/exim/mta.crt
tls_privatekey = /etc/exim/mta.key
```

Soll der Exim-Server von ausgewählten Gegenstellen ein gültiges Zertifikat erzwingen, kann das über folgende Optionen erreicht werden:

```
tls_verify_hosts = mail1.dfn-cert.de : mail2.dfn-cert.de
tls_verify_certificates = /etc/exim/CAfile.pem
```

`tls_verify_hosts` ist die Liste der Hosts, die ein gültiges Zertifikat vorweisen müssen. Diese Liste kann natürlich auch im Hauptabschnitt des Exim-Konfiguration definiert und dann im obigen Abschnitt referenziert werden.

`tls_verify_certificate` enthält eine Liste von PEM-kodierten Root-Zertifikaten. Das impliziert natürlich, dass die zu prüfende Gegenstelle die erforderlichen Intermediate CA-Zertifikate zusammen mit ihrem Client-Zertifikat liefert. Verläuft die Prüfung der Zertifikatskette nicht erfolgreich, wird die Verbindung abgelehnt.

Alternativ zu `tls_verify_hosts` kann die Option `tls_try_verify_hosts` konfiguriert werden. Wenn die Prüfung fehlschlägt, wird trotzdem zumindest eine verschlüsselte Verbindung aufgebaut.

Die Konfiguration des Exim SMTP-Clients erfolgt im Abschnitt `remote_smtp`. Damit eine Gegenstelle das Zertifikat des Clients prüfen kann, müssen (analog der Konfiguration des SMTP-Servers) Zertifikat und Schlüssel konfiguriert werden. Auch gilt wieder, dass Intermediate-CAs zusammen mit dem Client-Zertifikat in eine Datei geschrieben werden können. Und natürlich können Zertifikat und Schlüssel für den SMTP-Server und -Client identisch sein.

Wenn Exim in seiner Funktion als SMTP-Client die Zertifikatskette der Gegenstelle prüfen soll, kann die Option `tls_try_verify_hosts` verwendet werden. Der Parameter erwartet eine Liste von zu prüfenden Hosts, `*` meint alle Hosts. Die Prüfung muss nicht erfolgreich sein, es wird ggf. trotzdem eine verschlüsselte Verbindung aufgebaut.

Die Option `tls_tempfail_tryclear` regelt, was im Fall eines fehlgeschlagenen TLS-Verbindungsaufbaus passieren soll, obwohl die Gegenstelle TLS-Unterstützung per STARTTLS signalisiert hat. Mit dem Parameter `true` wird dann eine Verbindung ggf. unverschlüsselt hergestellt.

```
tls_verify_certificates = /etc/exim/CAfile.pem
tls_certificate = /etc/exim/mta.crt
tls_privatekey = /etc/exim/mta.key
tls_try_verify_hosts = *
tls_tempfail_tryclear = true
```

Verschlüsselung mit ausgewählten Gegenstellen

Auch in Exim kann eine Verschlüsselung zu einem Mail-Server erzwungen werden. Im Main-Abschnitt der Exim-Konfiguration wird zunächst eine Liste der Domains definiert, für die Mails nur über eine sichere Verbindung geschickt werden soll:

```
domainlist tls_force_domains = dfn-cert.de
```


Diese Liste kann dann in der weiteren Konfiguration referenziert werden

Als nächstes wird im router-Abschnitt der Exim-Konfiguration ein Router definiert, der für diese Domain-Liste zuständig ist und die MX-Lookups durchführt. Für die Domains wird dann ein dedizierter Transport festgelegt.

```
tls_router:
  driver = dnslookup
  domains = +tls_force_domains
  transport = tls_smtp
```

Abschließend wird im transport-Abschnitt der smtp-transport definiert, der dann für die Domains Verschlüsselung erzwingt.

```
tls_smtp:
  driver = smtp
  hosts_require_tls = *
  tls_try_verify_hosts = *
```


3. DNS-based Authentication of Named Entities (DANE)

DANE wurde entwickelt, um die Abhängigkeit von Certification Authorities im TLS-Protokoll zu vermeiden. Das TLS-Protokoll basiert auf der Realisierung einer vertrauenswürdigen Zertifizierungshierarchie, die ein initiales Vertrauen in Root-CAs erfordert und, davon abgeleitet, das Vertrauen in deren Sub-CAs. Verschiedene CAs sind in den letzten Jahren aufgefallen, indem sie Zertifikate für Domains ausgestellt haben, für die sie keine hätten ausstellen dürfen. Das hat das Vertrauen in die Zertifizierungshierarchien nachhaltig negativ beeinflusst.

Es handelt sich bei DANE um ein Netzwerkprotokoll, das das DNS um neue Resource-Records erweitert. Diese Resource-Records können verschiedene Informationen enthalten, die per DNSSEC kryptografisch gesichert sind. Die für die Absicherung der SMTP-Verschlüsselung relevante Erweiterung sind TLSA-Records.

Um die Abhängigkeit von CAs zu vermeiden, ist die Idee, die für die Validierung eines Hostnamens erforderlichen Zertifikate im DNS abzulegen. Dazu wurde ein neuer DNS Resource Record „TLSA“ definiert, der das Zertifikat im PKIX-Format oder einen Hash des Zertifikats enthält oder den Digest des öffentlichen Schlüssels. So kann der Verwalter der DNS-Zone selber festlegen, welche Zertifikate für welche Hostnamen in der Domain gültig sein soll.

Damit die Gültigkeit der Antworten eines DNS-Servers überprüft werden können, muss ein anfragender DNSSEC-fähiger DNS-Client die Antworten überprüfen und als gültig an eine Applikation weitergeben. Idealerweise läuft zu diesem Zweck ein lokaler Caching Name-Server auf den Mail-Host, der DNSSEC unterstützt und entsprechende Antworten validieren kann. Andernfalls besteht die Möglichkeit, dass Man-in-the-Middle-Angriffe die Antworten des Name-Servers manipulieren.

DANE/TLSA sollte nur für die Gegenstellen erzwungen werden, für die auch gültige TLSA-Records hinterlegt sind. Am einfachsten lässt sich das mit dem Werkzeug `dig` erledigen. Nachdem die gültigen MX-Hosts für eine Domain ermittelt wurden, werden die einzelnen Hosts geprüft. Hier ein gekürztes Beispiel für einen web.de-MX:

```
$ dig _25._tcp.mx-ha02.web.de IN TLSA

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4464
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;_25._tcp.mx-ha02.web.de. IN TLSA

;; ANSWER SECTION:
_25._tcp.mx-ha02.web.de. 698 IN TLSA 3 1 1
            409C9E91A2A9F4D7881DBF0094B3839D4343A4A57D9BF559FDEB0C1F 4C5B8B3E
```

Die Antwort-Zeile in der Answer-Section ist vor dem Hash manuell umgebrochen. Wichtig ist, dass `flags: ad` (Authenticated answer) gesetzt ist. Das zeigt zum einen, dass der verwendete Resolver DNSSEC-fähig ist, und zum anderen, dass die Antwort signiert ist.

3.1. Postfix

Postfix ab Version 2.11 unterstützt DANE/TLSA. Zunächst muss DNSSEC für den Postfix-SMTP aktiviert werden, dazu wird der Parameter `smtp_dns_support_level` auf `dnssec` gesetzt. Postfix prüft jetzt, ob DNS-Antworten vom DNSSEC-fähigen Resolver als gültig gekennzeichnet sind.

```
smtp_dns_support_level = dnssec
smtp_host_lookup = dns, native
```

DANE/TLSA sollte nur für Domains aktiviert und erzwungen werden, die entsprechende TLSA-Records im ihrem DNS konfiguriert haben. Wenn das der Fall ist, kann das Beispiel aus Kapitel 2.4.3 in der `smtp_tls_policy_map` erweitert werden:

```
smtp_tls_policy_maps      = hash:/etc/postfix/smtp_tls_policy_map
```

Und die `smtp_tls_policy_map`:

```
dfn-cert.de  secure
web.de       dane-only
```

Mails an die Domain `web.de` werden jetzt nur noch per DANE TLS übertragen. Sollte der TLSA-Record für die MXe der Domain nicht verfügbar oder ungültig sein, wird die Mail zurückgestellt und nicht mehr zugestellt.

Es besteht auch die Möglichkeit, Mandatory DANE-Authentication zu aktivieren:

```
smtp_dns_support_level = dnssec
smtp_host_lookup = dns, native
smtp_tls_security_level = dane
```

Mit dieser Konfiguration werden TLSA-Records verwendet. Sollte allerdings die Validierung per DNSSEC fehlschlagen, werden Mails nicht zugestellt. Sind wiederum keine TLSA-Records für die Domain konfiguriert, wird Opportunistic-TLS verwendet.

3.2. Exim

Exim unterstützt ab der Version 4.85 DANE/TLSA. Es ist aber anzumerken, dass die Implementierung noch „recht frisch“ ist. Die hier gemachten Angaben konnten nicht getestet werden und es gibt auch wenig Dokumentation zu diesem Thema. Die in diesem Abschnitt gemachten Angaben sind somit möglicherweise fehlerhaft.

Zunächst wird im Main-Abschnitt der Exim-Konfiguration die Option

```
dns_dnssec_ok = 1
```

gesetzt. Dadurch werden DNSSEC-Anfragen aktiviert. Analog der im Kapitel 2.4.3 beschriebenen Verschlüsselung mit ausgewählten Gegenstellen sollte auch hier wieder mit Domain-Listen gearbeitet werden. Im Main-Abschnitt wird dazu eine Liste der Domains definiert, für die Mails ausschließlich per DANE/TLSA zugestellt werden sollen:

```
domainlist dane_force_domains = web.de
```


Dann wird für diese Liste ein Router im Abschnitt `router` definiert. Die Option `dnssec_require_domains` sorgt dafür, dass nur DNS-Antworten mit gesetzten ad-Flag akzeptiert werden. Ohne das ad-Flag wird ein Host-Lookup Fehler erzeugt und die Mail nicht zugestellt.

```
dane_router:
  driver = dnslookup
  transport = dane_smtp
  dnssec_require_domains = +dane_force_domains
```

Abschließend wird im transport-Abschnitt der smtp-transport definiert, der Verschlüsselung für die Domains erzwingt.

```
dane_smtp:
  driver = smtp
  hosts_require_dane = *
```

4. Best-Practices-Konfiguration

In diesem Kapitel werden die Best-Practice Konfigurationen für die einzelnen Mailserver zusammenfassend beschrieben. Die Beispiel-Konfiguration enthält eine vollständige Konfiguration für SMTP-Transportverschlüsselung. Details zu den einzelnen Konfigurationswerten können den Kapiteln 2 und 3 entnommen werden.

Die Konfigurationen bieten einen guten Kompromiss zwischen Sicherheit und Verfügbarkeit des Dienstes. Bei allen Konfigurationen wird davon ausgegangen, dass das Paket ca-certificates installiert ist und die CA-Zertifikate unter `/etc/ssl/certs` mit den entsprechenden Hash-Links liegen.

4.1. Postfix

Die folgenden Konfigurationszeilen müssen in die `main.cf` des Postfix Mailservers eingetragen werden.

```
smtp_tls_security_level         = may
smtp_tls_key_file               = /etc/postfix/ssl/private/mta-key.pem
smtp_tls_cert_file              = /etc/postfix/ssl/certs/mta-cert.pem
smtp_tls_CApath                 = /etc/ssl/certs
smtp_tls_mandatory_protocols    = !SSLv2, !SSLv3
smtp_tls_protocols              = !SSLv2, !SSLv3
smtp_tls_scert_verifydepth      = 5
smtp_tls_loglevel               = 1

smtpd_tls_security_level        = may
smtpd_tls_key_file              = /etc/postfix/ssl/private/mta-key.pem
smtpd_tls_cert_file             = /etc/postfix/ssl/certs/mta-cert.pem
smtpd_tls_CApath                = /etc/ssl/certs
smtpd_tls_mandatory_protocols   = !SSLv2, !SSLv3
smtpd_tls_protocols             = !SSLv2, !SSLv3
smtpd_tls_loglevel              = 1
```



```
smtpd_tls_mandatory_ciphers     = high
smtpd_tls_ask_ccert             = yes
smtpd_tls_dh1024_param_file     = /etc/postfix/dh2048.pem

tls_high_cipherlist             = EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:⤸
    ⤹EECDH+aRSA+SHA256:EECDH:+CAMELLIA128:+AES128:+SSLv3:!aNULL:!eNULL:!LOW⤸
    ⤹:!3DES:!MD5:!EXP:!PSK:!DSS:!RC4:!SEED:!IDEA:!ECDSA:kEDH:CAMELLIA128-SHA:⤸
    ⤹AES128-SHA
tls_ssl_options                 = NO_COMPRESSION
```

4.2. Exim

Die folgenden Konfigurationen müssen in der Exim-Konfiguration in den jeweiligen Abschnitten eingetragen werden. Zunächst die Konfigurationsvariablen, welche sowohl für OpenSSL als auch GnuTLS gelten:

```
# Main-Abschnitt
log_selector = +tls_cipher +tls_peerdn
tls_advertise_hosts = *
tls_certificate = /etc/exim/mta.crt
tls_privatekey = /etc/exim/mta.key
tls_dhparam = /etc/exim/dh2048.pem
tls_verify_certificates = /etc/exim/CAfile.pem



# Abschnitt transport
remote_smtp:
  # ...
  tls_verify_certificates = /etc/exim/CAfile.pem
  tls_certificate = /etc/exim/mta.crt
  tls_privatekey = /etc/exim/mta.key
  tls_try_verify_hosts = *
  tls_tempfail_tryclear = true
```

Zusätzlich müssen noch folgende Zeilen verwendet werden, je nachdem, gegen welche Krypto-Library Exim gebaut ist. Die Parameter müssen im Main-Abschnitt für den Exim-Server und auch im Abschnitt transport unter `remote_smtp` für den Exim-Client gesetzt werden.

Für OpenSSL:

```
tls_require_ciphers = EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:EECDH+aRSA+⤸
    ⤹SHA256:EECDH:+CAMELLIA128:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!⤸
    ⤹EXP:!PSK:!DSS:!RC4:!SEED:!IDEA:!ECDSA:kEDH:CAMELLIA128-SHA:AES128-SHA
openssl_options = +all +no_sslv2 +no_sslv3 +no_compression ⤸
    ⤹cipher_server_preference
```

Für GnuTLS:


```
tls_require_ciphers = NORMAL:!VERS-SSL3.0:!ECDHE-ECDSA:!3DES-CBC:!AES-128-
    CBC:!AES-256-CBC:!ARCFOUR-128:!CAMELLIA-128-GCM:!CAMELLIA-256-GCM:!
    COMP-DEFLATE:!DHE-DSS:!SIGN-DSA-SHA1:!SIGN-RSA-SHA1:!VERS-DTLS1.2:!
    VERS-DTLS1.0:!SIGN-ECDSA-SHA384:!SIGN-ECDSA-SHA512:!SIGN-ECDSA-SHA224
    :!SIGN-ECDSA-SHA1:+COMP-NULL
```

A. Quellen

- BSI TR-03108-1: `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03108/TR03108-1.pdf?__blob=publicationFile&v=4`
- BSI TR-03116-4: `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03116/BSI-TR-03116-4.pdf?__blob=publicationFile`
- RFC7672, SMTP Security via Opportunistic DANE TLS: https://tools.ietf.org/html/rfc7672
- RFC2487, SMTP Service Extension: https://tools.ietf.org/html/rfc2487
- Postfix TLS: http://www.postfix.org/TLS_README.html
- Postfix Forward Secrecy: http://www.postfix.org/FORWARD_SECRECY_README.html
- BetterCrypto.org: https://bettercrypto.org/static/applied-crypto-hardening.pdf
- Exim-Dokumentation: http://www.exim.org/docs.html

