OpenSSL in Postfix

Viktor Dukhovni
Credits: Lutz Jänicke, Wietse Venema, …

temporarily:
https://dnssec-stats.ant.isi.edu/~viktor/prague.pdf

Structure of this talk

- Highlight use cases in which Postfix takes advantage of OpenSSL
  - Describe briefly the problem solved
  - The Postfix code is written with care, and generally well commented
- Postfix is a rich source of example real-world code using OpenSSL
  - https://github.com/vdukhovni/postfix/tree/master/postfix/src/tls/
  - Focus is TLS and X.509 authentication, not data at rest cryptography
  - The meat of the content is behind links to the underlying code
  - No time for code walk-through during the talk
    - Your homework is to open the links and study the code

BACKGROUND

Some SMTP and Postfix basics

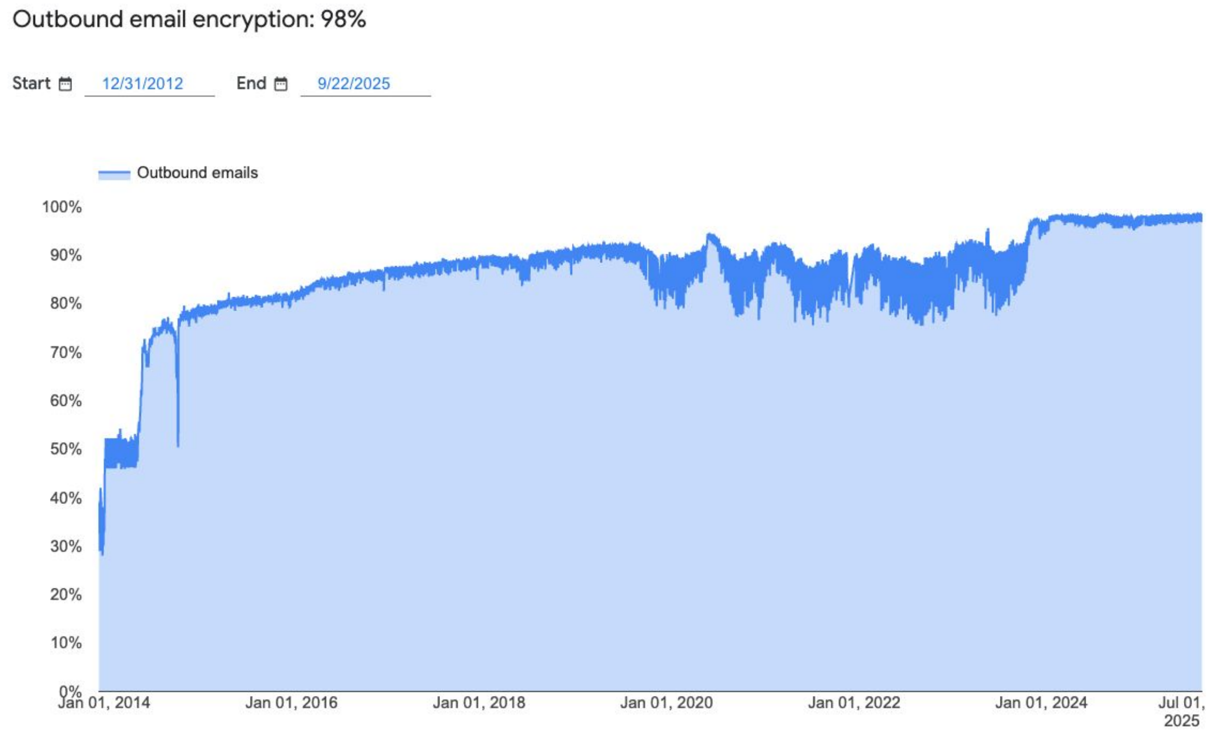
Complex SMTP TLS policy landscape

- Transport security policy is hop-by-hop and largely up to the sending client
  - Some mail sent in the clear when STARTTLS is neither required nor offered (or fails)
- SMTP TLS is mostly opportunistic
  - Typically unauthenticated (client ignores server's certificate) TLS
    - Protects only against *passive monitoring* (wiretaps)
  - Some reasons why in RFC 7672 Section 1.3
  - End-to-end (E2E) message encryption is mostly impractical:
    - Hampers blocking email abuse
    - Complicates search, archiving, key rotation, usability

Active (MiTM) attack resistance

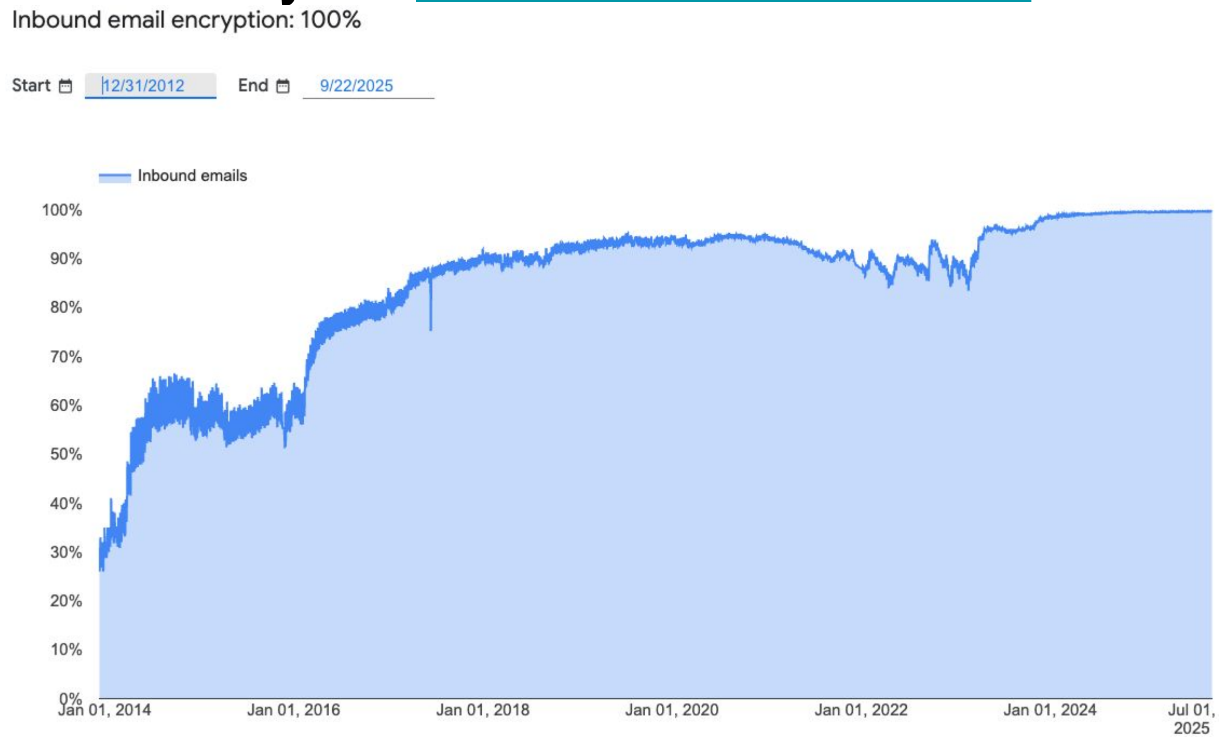
- Requires that client:
  - Knows that messages to a particular destination MUST use TLS
  - Knows how server MUST be authenticated
  - This needs to be downgrade resistant
- Possible with:
  - DANE (4.2+ million domains, downgrade-resistant via DNSSEC: 24 million domains)
  - MTA-STS (mostly between the largest email providers, weaker downgrade resistance)
  - Manual policy for business partners or other important peer domains
  - Possibly per-message metadata (REQUIRETLS support due soon in Postfix 3.11)

Almost all Gmail outbound traffic is TLS-protected


Outbound email encryption: 98%

Start 12/31/2012 End 9/22/2025

Practically all Gmail inbound traffic is TLS-protected


Inbound email encryption: 100%

Start 12/31/2012 End 9/22/2025

Brief history of Postfix

- After rich history of Sendmail security issues, …
  - Alpha: 1998/01/05 (for select group of testers)
  - Public beta: 1999/01/22
  - 1.0: 2001/02/28
  - Wietse merged TLS support: 2005/07
    - Based on patch series by Lutz Jänicke, starting 1999/03/29 with OpenSSL 0.9.2!
- 25+ years of solid examples of OpenSSL in action
- Postfix 3.11 dev: ~144k LOC (cf. OpenSSL ~450k)
  - ~13k TLS-related LOC
  - Total of ~6 CVEs in project history

Wietse's philosophy

“I learned to program carefully for selfish reasons. I did not want to sleep on the floor next to my physics experiments”. Wietse

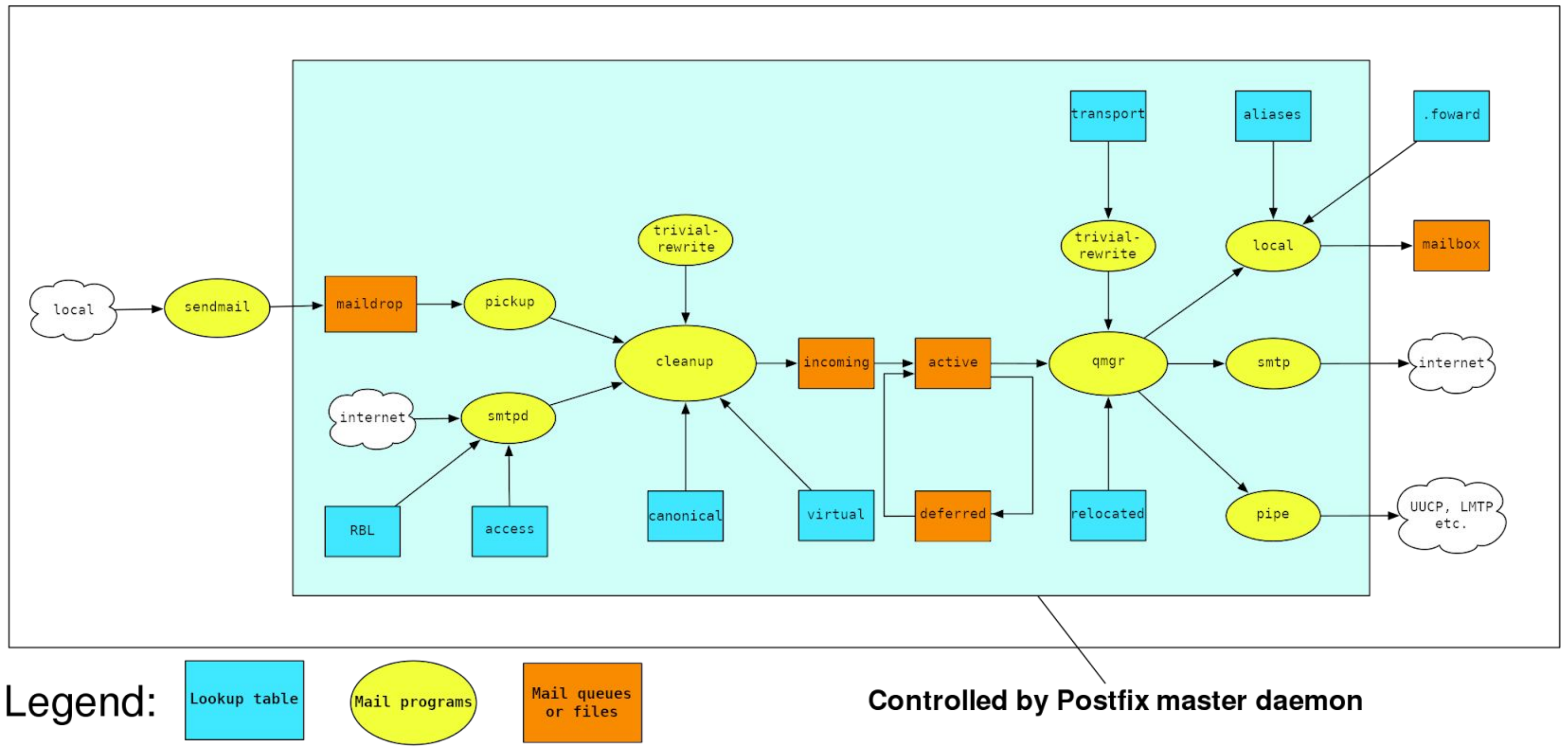
"people expect that my programs solve more problems than they cause. [It's] something close to perfection. ... I am preparing an incomplete system for release [to experimentally determine people's needs]. That's why I call it a beta. It has nothing to do with software quality." Wietse

- Strong commitment to backwards compatibility, decade or more old configurations typically work unchanged today.

My work on Postfix (and OpenSSL)

- Somewhat late to the party, I'm a Postfix user since 2001/05
  - First contributed patch merged 2001/07
- Google IPO ran on Postfix servers
- Refactored Postfix TLS stack in 2006 and its primary maintainer since
- Implemented DANE support in 2013–2015 and authored DANE for SMTP RFCs
- Contributed DANE support to OpenSSL 1.1.0 in 2016 and joined project
  - Refactored OpenSSL X.509 validation, still focused on that part of the code base

Postfix multi-process architecture



Postfix multi-process architecture

- The **master**(8) server manages per-service worker processes
  - Workers each handle up to ~100 requests (connections?) and exit
  - New workers are spawned when a request comes in while all workers are *busy*
  - A few core workers (**qmgr, tlsmgr**) run indefinitely
  - Worker processes drop root privileges, trust only their own configuration
- **smtpd**(8) SMTP servers receive incoming mail
- **smtp**(8) SMTP clients deliver outgoing mail
- **tlsmgr**(8) stores TLS session tickets for SMTP clients
- **tlsmgr**(8) stores/rotates session ticket keys for SMTP servers
- **tlsproxy**(8) supports TLS connection reuse or (cleartext end) handoff

Postfix SMTP server TLS

Server operating modes

- Inbound **Message Transfer Agent** (**MTA**):
  - Port 25: mail from remote MTAs, optional `STARTTLS`
- Outbound **Message Submission Agent** (**MSA**):
  - Port 587: outbound mail from authenticated users, mandatory `STARTTLS`
  - Port 465: outbound mail, implicit TLS
- TLS settings can vary between MTA and MSA
- Optionally requests client certs to authenticate trusted clients (SMTP VPN?)
  - Mostly MSA SMTP relay access control via key fingerprint
    - Rarely by valid signature from a trusted CA
    - Allowed to originate outbound email?
    - Exempted from anti-spam filters?

Explicit OpenSSL library initialisation

- `tls_library_init`() avoids system-wide `openssl.cnf` by default
    - Called once in each process, **prior** to any use of OpenSSL
    - Optional custom or default config file, and application name
    - Largely unaffected by RedHat crypto policy (not well suited to opportunistic TLS)
- Relevant APIs:
    - OPENSSL_INIT_new()
    - OPENSSL_INIT_set_config_file_flags()
    - OPENSSL_INIT_set_config_filename()
    - OPENSSL_INIT_set_config_appname()
    - OPENSSL_init_ssl()
    - OPENSSL_INIT_free()

SSL_CTX construction

- `tls_server_init`()
    - Calls library version check (warning if run-time version too different)
    - Creates primary SSL_CTX object and twin for SNI
    - Applies operator-specified min/max protocol
        - Was once `SSL_OP_NO_SSLv3`, … but experience with Postfix suggested a better way that made it into OpenSSL
    - Arranges to tag SSL handles with application data

SSL_CTX construction

- `tls_server_init`()...
  - Defaults security level to 0 (opportunistic TLS)
  - Turns off truncation detection (`SSL_OP_IGNORE_UNEXPECTED_EOF`)
  - Sets up optional session caching
- APIs:
  - `SSL_get_ex_new_index(),`
  - `SSL_CTX_set_options(),`
  - `SSL_CTX_set_min_proto_version(),`
  - `SSL_CTX_set_security_level()`

Server SSL_CTX

- `tls_server_init`() …
  - Sets up stateless resumption key rollover
  - Enables server to client RFC 7250 raw public key (**RPK**) support
  - Loads server certificate chains
  - Optionally, configures key exchange supported "groups"
  - Configures optional trust anchors (CAfile, CApath)
  - Configures optional client certificate solicitation
    - CA hints, verify callback

Optional loading of trust anchors (CAfile, CApath)

- Only when client certificates are used and rely on CA trust!
  - Postfix discourages relying on CAs for validation of client certificates
    - Server operator issues own certificates to "known" clients
  - Instead, ACL files with public key (or else certificate) fingerprints
  - No support for or need for CRLs, just prune stale ACL entries
  - *Usual* system-wide WebPKI CAs not loaded by default
- APIs:
  - SSL_CTX_load_verify_locations()
  - SSL_CTX_set_default_verify_paths()

Loading of server's own certificate chain(s)

- Prefers key + chain in a single file, and loads these atomically
  - File opened just once to read both key and cert chain when same name is used for both
  - Postfix-specific PEM multi-chain format (underlying parser)
    - Sequence of (key1, cert1, issuer certs …), (key2, cert2, issuer certs …), …
    - Also used with SNI key/value tables
    - Or ordered list of files one or more per algorithm
  - Legacy support for up to three separate key and cert+chain files,
    - Nominally for DSA, RSA and ECDSA, but really any three distinct algorithms
- APIs:
  - SSL_CTX_use_PrivateKey_file(), SSL_CTX_use_certificate_chain_file(), SSL_CTX_check_private_key()
  - PEM_read_bio(), d2i_PrivateKey(), d2i_PKCS8_PRIV_KEY_INFO(), d2i_X509()

Multiple server names: SNI-specific chain(s)

- Accessed via indexed key value tables (LMDB, …)
- Value is a PEM blob with one or more (key, cert, issuers), … sequences
- Source format is a text table with filenames:

```
name1.com file1.pem, file2.pem, …
name2.net file3.pem, …
…
```

- `postmap -F` converts source form to key/value tables
  - Files concatenated and copied to table value
  - Source files and tables root-readable only
  - Tables opened before dropping privileges

SNI processing

- SMTP server opens tables before dropping privs
    - Registers SNI callback
    - Lookups happen after privs dropped in the SNI callback,
        - Default key/chain used if no SNI match
            - HTTPS servers may want to be more strict (DNS rebinding)
        - Result loaded into the SNI SSL_CTX
        - Server workers are single-threaded, so no concurrency concerns
- APIs:
    - SSL_CTX_set_tlsext_servername_callback()
    - SSL_CTX_set_tlsext_servername_arg()
    - SSL_get_servername()
    - SSL_set_SSL_CTX()

Requesting client certificates

- Typically optional (`smtpd_tls_ask_ccert`)
  - Clients that don't present certs don't get special access
- Can be required (`smtpd_tls_req_ccert`)
  - Sadly, also requires that the certificate be issued by a trusted CA
  - Rarely used legacy feature
- Set up verification policy and callback
  - Callback never aborts handshake, graceful SMTP disconnect, your needs may vary!
- APIs:
  - SSL_CTX_set_verify()
  - SSL_load_client_CA_file()
  - SSL_CTX_set_client_CA_list()
  - SSL_dup_CA_list()

Session tickets (resumption PSKs)

- RFC 5077 session ticket (typical format):

```
struct {
    opaque key_name[16]; // Supports key rollover
    opaque iv[16];       // Fresh for each ticket
    opaque encrypted_state<0..2^16-1>; // Payload
    opaque mac[32];      // HMAC-SHA256 & similar
} ticket;
```

- Internal to server, secret (name, block cipher key, HMAC key) triples
- OpenSSL default: random key, fixed for server process lifetime
- Postfix uses multiple ephemeral processes, need persistent shared keys
- Unchanging shared key risks loss of forward secrecy

Key rollover

- Server needs only a two-slot cache with an ***active*** and ***previous*** key
- The ***active*** key encrypts sent tickets and decrypts received tickets
- The ***previous*** key is used to decrypt only, enabling non-disruptive rollover
- The "name" in the client's ticket determines which key to apply
- The MAC key handles tamper-proofing

Key rollover

- Server callback registered via `SSL_CTX_set_tlsext_ticket_key_evp_cb()`
  - Creates new tickets
  - Decrypts received tickets (indicating whether to issue a replacement or not)
  - Postfix always allows reuse of unexpired tickets
  - When current active key expires, server requests the *active* key (null name) from **tlsmgr**
  - When receiving a ticket with an unknown name, request that name from **tlsmgr**
    - This might be the newest *active* key just minted by a peer server
    - Or an existing *previous* key just learned by a fresh server receiving an older ticket
    - Key expiration time determines which of the two key slots is chosen

TODO: Someday, key rollover in OpenSSL?

- Non-trivial:
    - Performant thread safety?
        - Lockless if suitable key known in current thread?
        - Are keys refreshed in the background?
    - Distributed (multi-process and/or multi node) variant of **tlsmgr** service
        - Is there an existing protocol for this?
        - What 3rd-party key management systems participate?
        - …

Raw Public Keys (RFC7250)

- Server certificate message is just a DER SubjectPublicKeyInfo (X.509 SPKI)
- Used when enabled by the server and client indicates support
- Other clients continue to receive X.509 certificates
- The server is configured with a private key + certificate as usual
  - The RPK is extracted from the certificate (can be minimal self-signed if for RPK-only)
- Servers can also solicit RPKs from clients
  - When server access control is based on just the client's public key and not its cert
  - Optionally enabled in Postfix, X.509 always also accepted
- APIs:
  - SSL_CTX_set1_server_cert_type(), SSL_set1_server_cert_type()
  - SSL_CTX_set1_client_cert_type(), SSL_set1_client_cert_type()

Key exchange (supported groups)

- Postfix has legacy code for explicit server DH groups
  - auto-negotiation strongly recommended
- Explicit EC curves no longer supported
- With OpenSSL 3.5 changes for PQC, recommend to use default groups
  - Some risk of problems with larger TLS client hello
    - Fixed at originally reported `boeing.com`
  - Else customise via an *openssl.cnf* file
    - Avoid application code to set supported groups

PQC supported groups

- Client sends both a hybrid X25519 + MLKEM768 and an X25519 keyshare
- Server prefers the former, requesting a fresh client hello if supported but not sent

```
Groups = ?*X25519MLKEM768 / ?*X25519:?secp256r1 / ?X448:?secp384r1:?secp521r1 / ?ffdhe2048:?ffdhe3072
```

- Possible client-side "boeing.com" work-around:
  - Support, but don't automatically send keyshare for X25519MLKEM768

```
Groups = ?X25519MLKEM768 / ?*X25519:?secp256r1 / ?X448:?secp384r1:?secp521r1 / ?ffdhe2048:?ffdhe3072
```

- APIs (to NOT use directly):
  - SSL_CTX_set1_curves_list()
  - Deprecated since 3.0: SSL_CTX_set_tmp_dh(), SSL_CTX_set_tmp_ecdh()

More TLS settings?

- Never wrote Postfix code to customise supported signature algorithms
      - Or TLS 1.3 symmetric ciphers, …
  - But easily set in a dedicated (Postfix-only) config file
  - Set preference order of available server certs

```
SignatureAlgorithms = mldsa65:ecdsa_secp256r1_sha256
```

  - When requesting client certs, may need to also set `ClientSignatureAlgorithms`
    - Above server setting rules out RSA certs from clients,
    - A longer list in `SignatureAlgorithms` may serve both needs

External session cache

- Still supports optional *stateful* session cache
  - Mostly obsoleted by session tickets
  - External, shared via **tlsmgr**
  - Internal cache has just one slot
  - No "remove" callbacks, **tlsmgr** schedules its own removal of stale sessions
- APIs:
  - SSL_CTX_set_session_id_context()
  - SSL_CTX_sess_set_cache_size()
  - SSL_CTX_set_session_cache_mode()
  - SSL_CTX_sess_set_new_cb()
  - SSL_CTX_sess_set_get_cb()
  - SSL_CTX_set_timeout()

Postfix SMTP server TLS

TLS connection setup

tls_server_start()

- Unexpected buffered data is flushed *before* initiating TLS handshake
  - Multi-protocol STARTTLS vulnerability discovered by Wietse
- Creates SSL handle, configures TLS 1.2 ciphers, adds application context
- Security level raised to 1 when client certificates are required
- Optionally enables client to server RPK (in lieu of client "certs")
- SSL_set_fd() called with non-blocking socket

tls_server_start()...

- Initiates SSL_accept() via tls_bio() I/O handler
  - Also used by SSL_connect(), and data reads and writes
  - IMPORTANT: clears error stack
  - Deals with WANT_READ, WANT_WRITE, timeouts and TLS errors
  - Per I/O timeouts (default) and (under stress) deadline timeouts to receive a command or data
    - Postfix has its own "**vstream**" buffering I/O akin to OpenSSL **BIO**s

Post-handshake tasks

- Collect and log handshake properties
  - Client certificate subject, issuer and fingerprints (cert and key)
    - or perhaps RPK fingerprint
  - Protocol, cipher, certificate type, key exchange type, MAC

```
Anonymous TLS connection established
  from mail.dnswl.org[130.255.78.51]:
  TLSv1.3 with cipher TLS_AES_256_GCM_SHA384 (256/256 bits)
  key-exchange X25519MLKEM768
  server-signature ML-DSA-65 (raw public key)
```

- Subsequently available for access control decisions
  - Issuer and subject exposed only if verified

SMTP server highlights

- Explicit initialisation
- Avoids WebPKI trust anchors
- Advanced certificate configuration
- Resumption key rollover
- Raw public key support
- Dedicated config file for non-default Groups, SignatureAlgorithms, …
- Non-blocking I/O
- Reflection (logging, …)

Postfix SMTP client TLS

SMTP Client TLS security levels (unauthenticatd)

- **none**: no TLS
- **may**: Unauthenticated opportunistic TLS
- **encrypt**: Mandatory unauthenticated TLS

SMTP Client TLS security levels (authenticated)

- **fingerprint**: Pinned peer certificate or public key
  - via locally synthesised DANE TLSA records
- **dane**: Opportunistic DANE TLS
  - when usable DNSSEC TLSA records are found, else "may"
- **dane-only**: Mandatory DANE TLS, avoids non-conformant MX hosts
- **secure**: WebPKI authentication with configurable hostname checks
  - Supports per-destination trust-anchor certs or public keys (as PEM files)
    - via locally synthesised DANE TLSA records

Security policy

- Global default for most domains is **"may"** or **"dane"**
- Per destination policy table
- Per-message TLS policy:
  - Supports TLS-Required: no header, for error notices, …
  - Supports (3.11 dev snapshots) ESMTP **REQUIRETLS** option from sender

Security policy…

- MTA operator (Postfix user) is presumed to be technically savvy, has many more knobs to tweak than typical TLS user:
    - 37 SMTP client settings
    - 31 SMTP server settings
    - 28 underlying TLS-library settings

Postfix SMTP client TLS

Initialisation

Key differences from server

- tls_client_init() similar to server, but
- Enables DANE support
  - Optionally used in subsequent connections
- Just one SSL_CTX, no need for SNI-twin
- SNI name sent with DANE and MTA-STS
  - Otherwise defaults off, but can be set to "hostname" (i.e. use name of MX host)
- Enables client-to-server RPK when client cert configured
- Client-side cache is always stateful
  - External only, shared via **tlsmgr**(8)

DANE in OpenSSL

- OpenSSL does not do the DNS lookups
    - Providing the relevant TLSA records is application responsibility
    - Feature, because TLSA records don't have to come from DNS
    - TLSA records also useful to express various local policies
    - Not used that way in Postfix, but can augment rather than replace WebPKI
        - Usage PKIX-TA(0) requires a WebPKI chain with a matching CA certificate or key
        - Usage PKIX-EE(1) requires a WebPKI chain with a matching EE certificate or key
    - DANE is only OpenSSL mechanism to authenticate peer raw public keys
        - Bidirectional RPKs can be a good choice for fixed server-to-server mutual TLS
- APIs:
    - `SSL_dane_enable(),`
    - `SSL_dane_set_flags(),`
    - `SSL_dane_tlsa_add()`

Postfix SMTP client TLS

Connection setup

Client chooses security policy

- tls_client_start() security policies
  - Unauthenticated: **`encrypt`**
    - Server certificate is ignored, TLS <= 1.2 prefers anonDH ciphers
  - Direct pin: **`fingerprint, dane`**
    - When matching only server's key, enables server to client RPK
  - Server managed pin: **`dane`**
  - WebPKI: **`secure`**
    - Multiple names or subdomain patterns can match the server cert
    - Per-destination trust-anchors via synthetic DANE-TA(2) TLSA records

Verification of server certificate

- Verification setup in tls_auth_enable()
- Verification callback always continues handshake
  - Authentication failure checked after, with graceful disconnect (QUIT) at application layer
- Resumed session verification status in `SSL_get_verify_result`()
  - But not certificate chain details
  - Care to distinguish between trust chain verification and hostname mismatch
    - Prioritise storing other errors over hostname mismatch
    - Then report "Untrusted" only if not hostname mismatch
  - This is because untrusted sessions may also be cached

Client audit trail

- Handshake outcome logging:

```
Untrusted TLS connection established to aspmx.l.google.com[64.233.170.26]:25:
  TLSv1.3 with cipher TLS_AES_256_GCM_SHA384 (256/256 bits)
  key-exchange X25519MLKEM768
  server-signature ECDSA (prime256v1)
  server-digest SHA256

92616901F9D: to=<openssl-users@openssl.org>,
  relay=aspmx.l.google.com[64.233.170.26]:25,
  delay=4.6, delays=0.03/0.01/2.8/1.7,
  tls=may, dsn=2.0.0, status=sent
  (250 2.0.0 OK  1758854597 d2e1a72fcca58-78102be944esi1727982b3a.776 - gsmtp)
```

Session caching

- Avoids insecure reuse
  - Two domains might share the same MX host, but have different security policies
  - Caching by either or both of hostname and IP address is not safe
  - Even behind the same load balanced IP address session caches may be disjoint
  - So cache key includes ehlo response hostname, found to correlate with shared state
  - Plus security level, cipher selection, protocol range, name matching, per-message policy, …
- Shared via the **tlsmgr**(8)
  - Store/Lookup by above key, then (de)serialised via {d2i,i2d}_SESSION().
  - Just one slot per lookup key, no support for multiple concurrent tickets in client or server
  - Assume that servers don't enforce single-use sessions.
    - Tracking of client is not a concern for infrastructure such as MTAs.

Questions?