Modernes SSL ideal einsetzen

Peter Eisentraut

08.05.2025

peter@eisentraut.org
https://peter.eisentraut.org/
@petereisentraut@mastodon.social

peter.eisentraut@enterprisedb.com
https://www.enterprisedb.com/
@edbpostgres@mastodon.social

Für diese Präsentation …

SSL = TLS

Quiz

Seit wann gibt es SSL in PostgreSQL?

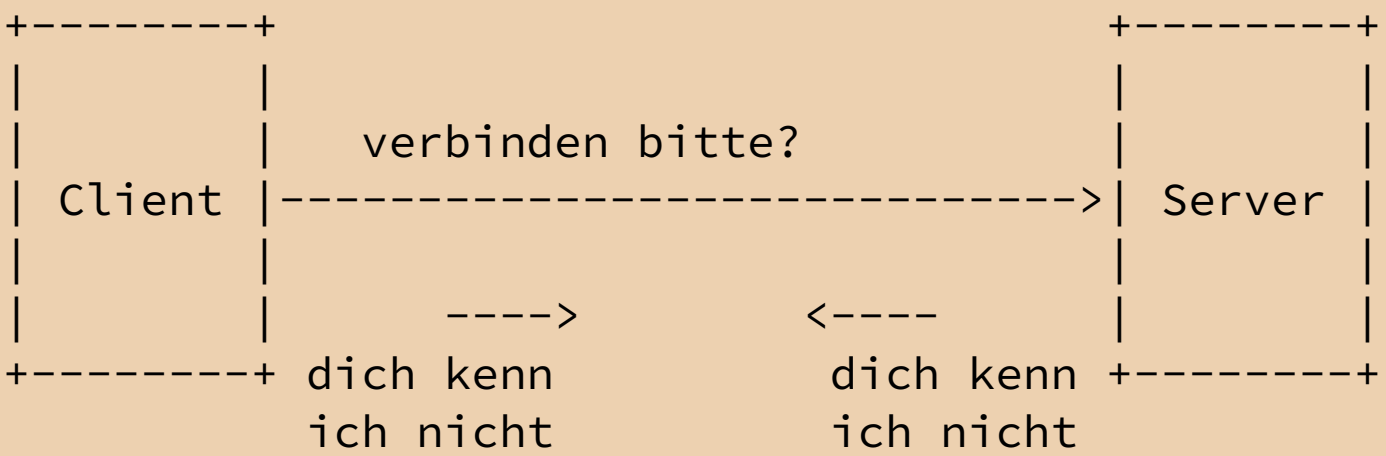
Welche Version?

Quiz

Seit wann gibt es SSL in PostgreSQL? — 2001

Welche Version? — 7.1

Client und Server



Server-Konfiguration

postgresql.conf

```
#ssl = off
#ssl_ca_file = ''
#ssl_cert_file = 'server.crt'
#ssl_crl_file = ''
#ssl_crl_dir = ''
#ssl_key_file = 'server.key'
#ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'   # allowed SSL ciphers
#ssl_prefer_server_ciphers = on
#ssl_ecdh_curve = 'prime256v1'
#ssl_min_protocol_version = 'TLSv1.2'
#ssl_max_protocol_version = ''
#ssl_dh_params_file = ''
#ssl_passphrase_command = ''
#ssl_passphrase_command_supports_reload = off
```

Client-Konfiguration

libpq Verbindungsparameter

- `sslmode`
- `sslnegotiation`
- `sslcompression`
- `sslcert`
- `sslkey`
- `sslcertmode`
- `sslpassword`
- `sslrootcert`
- `sslcrl`
- `sslcrldir`
- `sslsni`
- `ssl_min_protocol_version`
- `ssl_max_protocol_version`

Client-Konfiguration

JDBC Verbindungsparameter (properties)

- `ssl`
- `sslfactory`
- `sslfactoryarg`
- `sslmode`
- `sslNegotiation`
- `sslcert`
- `sslkey`
- `sslrootcert`
- `sslhostnameverifier`
- `sslpasswordcallback`
- `sslpassword`
- `sslResponseTimeout`

OS-Konfiguration

```
$ openssl version -d
OPENSSLDIR: "/usr/lib/ssl"
```

${OPENSSLDIR}/openssl.cnf:

```
openssl_conf = openssl_init

[openssl_init]
ssl_conf = ssl_configuration

[ssl_configuration]
server = server_tls_config
client = client_tls_config
system_default = tls_system_default

[server_tls_config]
... configuration for SSL/TLS servers ...

[client_tls_config]
... configuration for SSL/TLS clients ...

[tls_system_default]
MinProtocol = TLSv1.2
CipherString = ...
```

siehe man `config(5ssl)`, `SSL_CONF_CMD(3ssl)`

wahrscheinlich einfacher alles innerhalb PostgreSQL zu machen

Wann sollte SSL benutzt werden

- für alle TCP/IP-Verbindungen
- wahrscheinlich sogar für localhost
- (vielleicht sogar für Unix-Domain-Sockets, funktioniert aber (noch) nicht)

SSL im Server erzwingen

pg_hba.conf

```
# TYPE     DATABASE        USER   ADDRESS     METHOD
hostssl    all             all    xxx         yyy
hostssl    replication     all    xxx         yyy
```

SSL im Client erzwingen

libpq / JDBC

```
sslmode=require  (*)
```

SSL-Versionen

Version	Spezifikation	veröffentlicht	»deprecated«	OpenSSL	PostgreSQL
SSL 2.0	—	1995	2011 (RFC 6176)		—
SSL 3.0	(RFC 6101)	1996	2015 (RFC 7568)		—
TLS 1.0	RFC 2246	1999	2021 (RFC 8996)		7.1
TLS 1.1	RFC 4346	2006	2021 (RFC 8996)	1.0.1 (2012)	
TLS 1.2	RFC 5246	2008	—	1.0.1 (2012)	
TLS 1.3	RFC 8446	2018	—	1.1.1 (2018)	10/9.5

SSL-Versionen

- nur TLS 1.2 und 1.3 sind akzeptabel
- TLS 1.2 ist voreingestellte Minimalversion ab PostgreSQL 13
- für neue Installationen TLS 1.3 als Minimum erwägen
    - bietet auch schnellere Verbindungen

SSL-Version im Server einstellen

postgresql.conf

```
ssl_min_protocol_version = 'TLSv1.2'
```

(benötigt PostgreSQL 12)

SSL-Version im Client einstellen

libpq

```
ssl_min_protocol_version=TLSv1.2
```

(benötigt PostgreSQL 13)

SSL-Version im OS einstellen

openssl.cnf

```
MinProtocol=TLSv1.2
```

(benötigt OpenSSL 1.1.0)

Cipher Suites

```
#ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'
```

🤔

- betrifft nur TLS ≤ 1.2!

```
#ssl_tls13_ciphers = ''
```

- neu in PostgreSQL 18

Cipher Suites im OS einstellen

openssl.cnf

```
# TLS ≤1.2
CipherString = ...

# TLS 1.3
Ciphersuites = ...
```

Zertifikate und Schlüssel

- CA verwenden oder eigene erzeugen
  - keine selbst-signierten Zertifikate
- Server: Zertifikat mit Hostname
- (Client: Zertifikat mit Benutzername)
- Automatisieren! / Key-Management / Certificate-Management

Zertifikatsprüfung — Client

- ~~`sslmode=disable`~~
- ~~`sslmode=allow`~~
- ~~`sslmode=prefer`~~
- ~~`sslmode=require`~~
- ~~`sslmode=verify-ca`~~
- `sslmode=verify-full`

(Das bedeutet auch keine IP-Adressen.)

`sslrootcert=system` in Betracht ziehen

Zertifikatsprüfung — Server

für Client-Authentifizierung; gut für vom DBA kontrollierte Clients, z. B. Replikation

pg_hba.conf

```
# TYPE    DATABASE     USER  ADDRESS   METHOD
hostssl   replication  all   xxx       cert
```

oder

```
# TYPE    DATABASE     USER  ADDRESS   METHOD
hostssl   all          all   xxx       yyy clientcert=verify-full
```

benötigt CRL

Zertifikate Gültigkeitsdauer

≤398 Tage

- wenn vom Internet erreichbar: stimmt mit öffentlichen CAs überein
- ansonsten: erzwingt, dass man regelmäßig nachschaut

Zertifikate Gültigkeitsdauer Zukunft

Maximale Gültigkeitsdauer im Web ändert sich:

- ab März 2026: 200 Tage
- ab März 2027: 100 Tage
- ab März 2029: 47 Tage

unklar ob nicht-Web-Dienste folgen sollten

Passwortgeschützte Schlüssel

eher unnütz, aber manchmal von internen Regularien verlangt

- Server: meistens unpraktisch
- Client: besser ungeschützten Schlüssel mit richtiger Passwort-Authentifizierung verbinden

https://ssl-config.mozilla.org/

Paranoide Bonuspunkte

libpq

```
gssencmode=disable

require_auth=scram-sha-256
channel_binding=require

sslcertmode=require
```

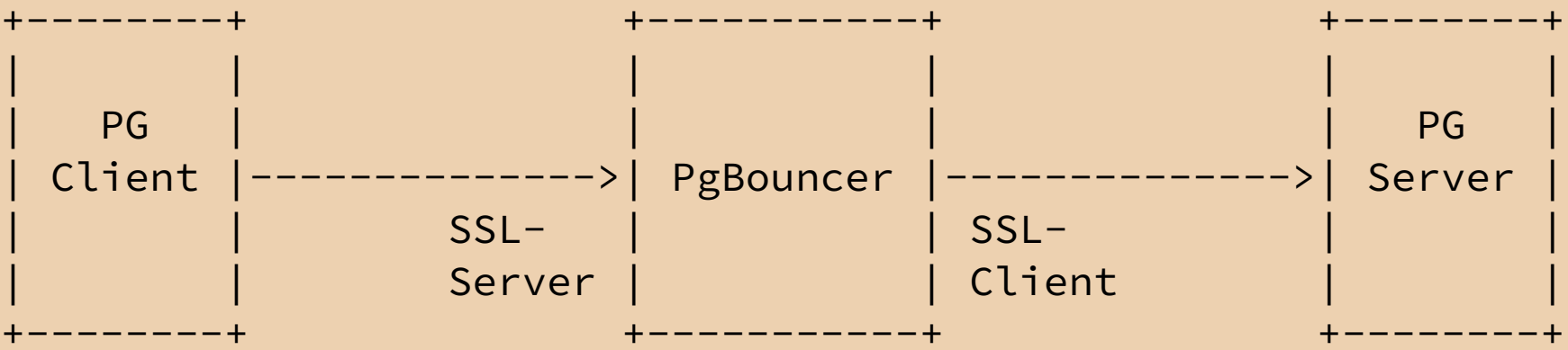
Paranoid — keine Bonuspunkte

Server

```
ssl_dh_params_file = ...
ssl_ecdh_curve = ...  # ≤PG17
ssl_groups = ...      # PG18
```

PgBouncer* und SSL

* oder ähnliches



PgBouncer SSL-Konfiguration

pgbouncer.ini

```
;;; TLS settings for accepting clients
;client_tls_sslmode = disable
;client_tls_ca_file = system default
;client_tls_key_file =
;client_tls_cert_file =
;client_tls_ciphers = default
;client_tls_protocols = secure
;client_tls_dheparams = auto
;client_tls_ecdhcurve = auto

;;; TLS settings for connecting to backend databases
;server_tls_sslmode = prefer
;server_tls_ca_file = system default
;server_tls_key_file =
;server_tls_cert_file =
;server_tls_protocols = secure
;server_tls_ciphers = default
```

PgBouncer SSL-Konfig. Empfehlungen

pgbouncer.ini

```
;;; TLS settings for accepting clients
;client_tls_sslmode = require
;client_tls_ca_file = eigene Datei
;client_tls_key_file = eigene Datei
;client_tls_cert_file = eigene Datei
;client_tls_ciphers = default
;client_tls_protocols = secure  ; = tlsv1.2,tlsv1.3
;client_tls_dheparams = auto
;client_tls_ecdhcurve = auto

;;; TLS settings for connecting to backend databases
;server_tls_sslmode = verify-full
;server_tls_ca_file = eigene Datei
;server_tls_key_file =
;server_tls_cert_file =
;server_tls_protocols = secure  ; = tlsv1.2,tlsv1.3
;server_tls_ciphers = default
```

Neu in PostgreSQL 17

`sslnegotation=postgres` (alt)

```
C --> S: Wie wärs mit SSL?
C <-- S: ok
C --> S: SSL starten, StartupMessage
```

`sslnegotation=direct` (neu)

```
C --> S: SSL starten, StartupMessage
```

Bald neu in PostgreSQL 18

- Cipher-Konfiguration für TLS 1.3
- bessere Curve/Group-Konfiguration
- Unterstützung für SSL-Debugging (libpq sslkeylogfile)
- Unterstützung für OpenSSL 1.0.2 entfernt (neues Min. 1.1.1)

In Arbeit für PostgreSQL 19

- SNI auf Server-Seite
- SSL per Voreinstellung erzwingen???

Zusammenfassung

- SSL immer (mehr) verwenden
- nur TLS 1.2 und 1.3
- immer verify-full verwenden
- Mozilla Konfigurationsgenerator anschauen
- interessante Neuigkeiten in PG17+

Tschüss / Fragen / Kontakt

peter@eisentraut.org
https://peter.eisentraut.org/
@petereisentraut@mastodon.social

peter.eisentraut@enterprisedb.com
https://www.enterprisedb.com/
@edbpostgres@mastodon.social