Percona

Unbiased Open Source Database Experts

PERCONA

PostgreSQL Security

Missteps and Opportunities

robert.bernier@percona.com

PERCONA

Introduction

- A presentation geared towards people familiar with PostgreSQL
- It's going to be moving pretty fast
- Methodology: show a problem followed by its mitigation
- This is just a starting point ...

INITIALIZING THE DATACLUSTER

(INITDB: never assume the initial settings)

PERCONA

EXAMPLE: environment variables changes everything

```
#
# Datacluster environment variables are already defined
#   UNIX process owner=rbernier,
#   export PGPORT=10014
#
initdb -D $PGDATA

#
# SERVER STARTUP
#
2022-04-19 08:00:15.706 PDT [1872397] FATAL:  database "rbernier" does not exist
psql: error: connection to server on socket "/tmp/.s.PGSQL.10014" failed: FATAL:  database "rbernier" does not exist


rbernier@wolven-xps:~/bin$ psql postgres
psql (14.0)
Type "help" for help.

postgres=# select user;
   user
----------
 rbernier
(1 row)

postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 rbernier  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}

postgres=# \q
rbernier@wolven-xps:~/bin$ ps aux | grep postgres
rbernier 1872272  0.0  0.2 205948 23232 ?        Ss   08:00   0:00 /home/rbernier/pg14/bin/postgres
rbernier 1872284  0.0  0.0 205948  3256 ?        Ss   08:00   0:00 postgres: checkpointer
rbernier 1872285  0.0  0.0 205948  3256 ?        Ss   08:00   0:00 postgres: background writer
rbernier 1872286  0.0  0.1 205948  8056 ?        Ss   08:00   0:00 postgres: walwriter
rbernier 1872287  0.0  0.0 206488  6164 ?        Ss   08:00   0:00 postgres: autovacuum launcher
rbernier 1872288  0.0  0.0  60548  3260 ?        Ss   08:00   0:00 postgres: stats collector
rbernier 1872289  0.0  0.0 206380  4228 ?        Ss   08:00   0:00 postgres: logical replication launcher
rbernier 1876781  0.0  0.0   9032   720 pts/1    S+   08:02   0:00 grep postgres
```

EXAMPLE: how you init the cluster determines the authentication

```
initdb -U postgres -D $PGDATA

# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
```

EXAMPLE: Security, a first step

```
initdb -A md5 -U postgres -D $PGDATA -W

# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5


rbernier@wolven-xps:~/bin$ psql 'user=postgres password=password'
postgres=#
```

EXAMPLE: Be explicit

```
initdb -A peer -U postgres -D $PGDATA

rbernier@wolven-xps:~$ psql -U  postgres
2022-04-19 08:32:10.934 PDT [1895245] LOG:  provided user name (postgres) and authenticated user name (rbernier) do not match
2022-04-19 08:32:10.934 PDT [1895245] FATAL:  Peer authentication failed for user "postgres"
2022-04-19 08:32:10.934 PDT [1895245] DETAIL:  Connection matched pg_hba.conf line 85: "local   all             all
          peer"
psql: error: connection to server on socket "/tmp/.s.PGSQL.10014" failed: FATAL:  Peer authentication failed for user "postgres"

rbernier@wolven-xps:~$ psql -U rbernier
2022-04-19 08:32:26.019 PDT [1895299] FATAL:  role "rbernier" does not exist
psql: error: connection to server on socket "/tmp/.s.PGSQL.10014" failed: FATAL:  role "rbernier" does not exist

-------------------------------------------
initdb -A peer -U rbernier -D $PGDATA

rbernier@wolven-xps:~/bin$ psql -U rbernier postgres
psql (14.0)
Type "help" for help.

postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 rbernier  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}


# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            ident
# IPv6 local connections:
host    all             all             ::1/128                 ident
```

EXAMPLE: init a superuser with a password

`initdb --auth-local=peer --auth-host=md5 -U rbernier -D $PGDATA -W`

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
```

```
rbernier@wolven-xps:~$ psql -h /tmp postgres
psql (14.0)
Type "help" for help.

rbernier@wolven-xps:~$ psql -h localhost postgres
Password for user rbernier:
psql (14.0)
Type "help" for help.
```

INITDB: A final word

REDHAT/CENTOS vs DEBIAN/UBUNTU

ISSUES

- Data cluster creation
- Data cluster location
- Host based authentication
- Server state
- Encrypted sessions
- Mistakes made while compensating for a lack of awareness of choices

WORKING WITH HOST BASED AUTHENTICATION RULES

(PG_HBA.CONF: confusion with hba rules)

PERCONA

OVERVIEW

```
The pg_hba.conf documentation is your friend:

- About the default host based authentication policy
- Rules based
- TYPE
    - local (UNIX domain socket)
    - host (TCP/IP)
    - hostssl (TCP/IP SSL)
    - nohostssl
    - hostgssenc (GSSAPI encrypted)         --> pg12+
    - hostnogssenc (not GSSAPI encrypted)   --> pg12+
- DATABASE
- USER
- ADDRESS
    - IP v4 vs IP v6
        - About IPv4
            - class (A,B,C)
                - inet 192.168.9.16
                - netmask 255.255.255.0
                - broadcast 192.168.9.255
            - CIDR
        - About IP v6
    - working with CIDR (https://www.vultr.com/resources/subnet-calculator/)
- METHOD
```

EXAMPLE AUTHENTICATION RULES (PG_HBA.CONF)

EXAMPLE: superuser is not always postgres

```
initdb --auth-local=peer --auth-host=md5 -U rbernier -D $PGDATA -W


# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5

rbernier@wolven-xps:~$ psql 'host=/tmp dbname=postgres user=rbernier password=password'
postgres=#

rbernier@wolven-xps:~$ psql 'host=127.0.0.1 dbname=postgres user=rbernier password=password'
postgres=#

postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 rbernier  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

PERCONA

EXAMPLE: connection reject fails

```
#
# netstat -tlnp
#
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:10014           0.0.0.0:*               LISTEN      1898000/postgres
tcp6       0      0 :::10014                :::*                    LISTEN      1898000/postgres


# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     reject
# IPv4 local connections:
host    all             all             127.0.0.1/32            reject
# IPv6 local connections:
host    all             all             ::1/128                 md5


rbernier@wolven-xps:~$ psql 'host=127.0.0.1 dbname=postgres user=rbernier password=password'
2022-04-19 09:01:58.628 PDT [1900107] FATAL:  pg_hba.conf rejects connection for host "127.0.0.1", user "rbernier", database "postgres", no encryption
psql: error: connection to server at "127.0.0.1", port 10014 failed: FATAL:  pg_hba.conf rejects connection for host "127.0.0.1", user "rbernier",
 database "postgres", no encryption

rbernier@wolven-xps:~$ psql 'host=/tmp dbname=postgres user=rbernier password=password'
2022-04-19 09:02:14.426 PDT [1900118] FATAL:  pg_hba.conf rejects connection for host "[local]", user "rbernier", database "postgres", no encryption
psql: error: connection to server on socket "/tmp/.s.PGSQL.10014" failed: FATAL:  pg_hba.conf rejects connection for host "[local]", user "rbernier",
 database "postgres", no encryption


psql 'host=::1 dbname=postgres user=rbernier password=password port=10014'
postgres=#
```

PERCONA

EXAMPLE: reject fails, forgetting about the host IP address

```
root@pg:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.231.38.1  netmask 255.255.255.0  broadcast 0.0.0.0
        inet6 fe80::216:3eff:fefa:d62f  prefixlen 64  scopeid 0x20<link>
        inet6 fd42:cb6a:5384:9a60::1  prefixlen 64  scopeid 0x0<global>
        ether 00:16:3e:fa:d6:2f  txqueuelen 1000  (Ethernet)
        RX packets 4324  bytes 266814 (266.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6200  bytes 47414445 (47.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     reject
# IPv4 local connections:
host    all             all             127.0.0.1/32            reject
host    all             all             0.0.0.0/0               md5
# IPv6 local connections:
host    all             all             ::1/128                 reject
```

```
rbernier@wolven-xps:~$ pg_ctl reload
2022-04-19 09:16:56.471 PDT [1898000] LOG:  received SIGHUP, reloading configuration files

rbernier@wolven-xps:~$ psql 'host=::1 dbname=postgres user=rbernier password=password'
2022-04-19 09:22:17.773 PDT [1903700] FATAL:  pg_hba.conf rejects connection for host "::1", user "rbernier", database "postgres", no encryption
psql: error: connection to server at "::1", port 10014 failed: FATAL:  pg_hba.conf rejects connection for host "::1", user "rbernier", database "postgres", no encryption

rbernier@wolven-xps:~$ psql 'host=/tmp dbname=postgres user=rbernier password=password'
2022-04-19 09:22:26.532 PDT [1903762] FATAL:  pg_hba.conf rejects connection for host "[local]", user "rbernier", database "postgres", no encryption
psql: error: connection to server on socket "/tmp/.s.PGSQL.10014" failed: FATAL:  pg_hba.conf rejects connection for host "[local]", user "rbernier", database "postgres", no
 encryption

rbernier@wolven-xps:~$ psql 'host=127.0.0.1 dbname=postgres user=rbernier password=password'
2022-04-19 09:22:56.574 PDT [1903809] FATAL:  pg_hba.conf rejects connection for host "127.0.0.1", user "rbernier", database "postgres", no encryption
psql: error: connection to server at "127.0.0.1", port 10014 failed: FATAL:  pg_hba.conf rejects connection for host "127.0.0.1", user "rbernier", database "postgres", no
 encryption

rbernier@wolven-xps:~$ psql 'host=10.231.38.1 dbname=postgres user=rbernier password=password'
psql (14.0)
Type "help" for help.
postgres=#
```

PERCONA

WORKING WITH CIDR

(PG_HBA.CONF)

PERCONA

Setup

```
HOSTS
pg: 10.231.38.73
    fd42:cb6a:5384:9a60:216:3eff:fe3f:d69c

h1: 10.231.38.108
    fd42:cb6a:5384:9a60:216:3eff:fe27:bdf0

h2: 10.231.38.42
    fd42:cb6a:5384:9a60:216:3eff:fe67:518a

Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN      254/postgres
tcp6       0      0 :::5432                 :::*                    LISTEN      254/postgres

LEGEND
- "::1/128" means localhost only
- "::0/0: open to all addresses

CAVEAT
- "ifconfig" is your friend
```

EXAMPLE

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   all             postgres                                peer
local   all             all                                     md5
host    all             all             0.0.0.0/0               md5
host    all             all             ::0/0                   md5

# WORKS: h1, h2

psql 'host=10.231.38.73 user=postgres password=postgres dbname=postgres sslmode=disable'

psql 'host=fd42:cb6a:5384:9a60:216:3eff:fe3f:d69c user=postgres password=postgres
 dbname=postgres sslmode=disable'
```

PERCONA

EXAMPLE: did you forget a database?

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   all             postgres                                peer
local   all             all                                     md5

# host h1:
host    all             all             10.231.38.96/28         md5
host    all             all             fd42:cb6a:5384:9a60:0216:3eff:fe27:bdf0/128 reject
#
# host h2:
host    db01            all             10.231.38.42/32         md5
host    all             all             fd42:cb6a:5384:9a60:0216:3eff:fe67:518a/128   md5

host    all             all             0.0.0.0/0               reject
host    all             all             ::0/0                   reject


# WORKS: h1 (RANGE: 10.231.38.96 - 10.231.38.111)
psql 'host=10.231.38.73 user=postgres password=postgres dbname=postgres sslmode=disable'

# FAILS: h1
psql 'host=fd42:cb6a:5384:9a60:216:3eff:fe3f:d69c user=postgres password=postgres dbname=postgres sslmode=disable'

# FAILS: h2
psql 'host=10.231.38.73 user=postgres password=postgres dbname=postgres sslmode=disable'

# WORKS: h2
psql 'host=10.231.38.73 user=postgres password=postgres dbname=db01 sslmode=disable'
psql 'host=fd42:cb6a:5384:9a60:216:3eff:fe3f:d69c user=postgres password=postgres dbname=postgres sslmode=disable'
```

PERCONA

WORKING WITH SECURE SOCKET LAYERS (SSL)

Enabling Ssl Encryption

1) Create either a Self-Sign Certificate or acquire one.
2) Enable SSL in PostgreSQL.
3) Restart PostgreSQL Service.

Create Self-Sign Certificate: a nice little script

Copy the generated "server.crt" and "server.key" into the datacluster, or an otherwise appropriate location.

```
    #!/bin/bash

    set -e

    SUBJ="/C=US/ST=Washington/L=Seattle/O=Percona/OU=Professional Services/CN=$(hostname -f)
/emailAddress=robert.bernier@zonarsystems.com"

    KEY="server.key"
    CRT="server.crt"

    /usr/bin/openssl  req \
            -nodes \
            -x509 \
            -newkey rsa:2048 \
            -keyout $KEY \
            -out $CRT \
            -days 3560 \
            -subj "$SUBJ"

    chmod 600 $KEY
    chmod 664 $CRT

    echo "DONE"
```

Enable SSL

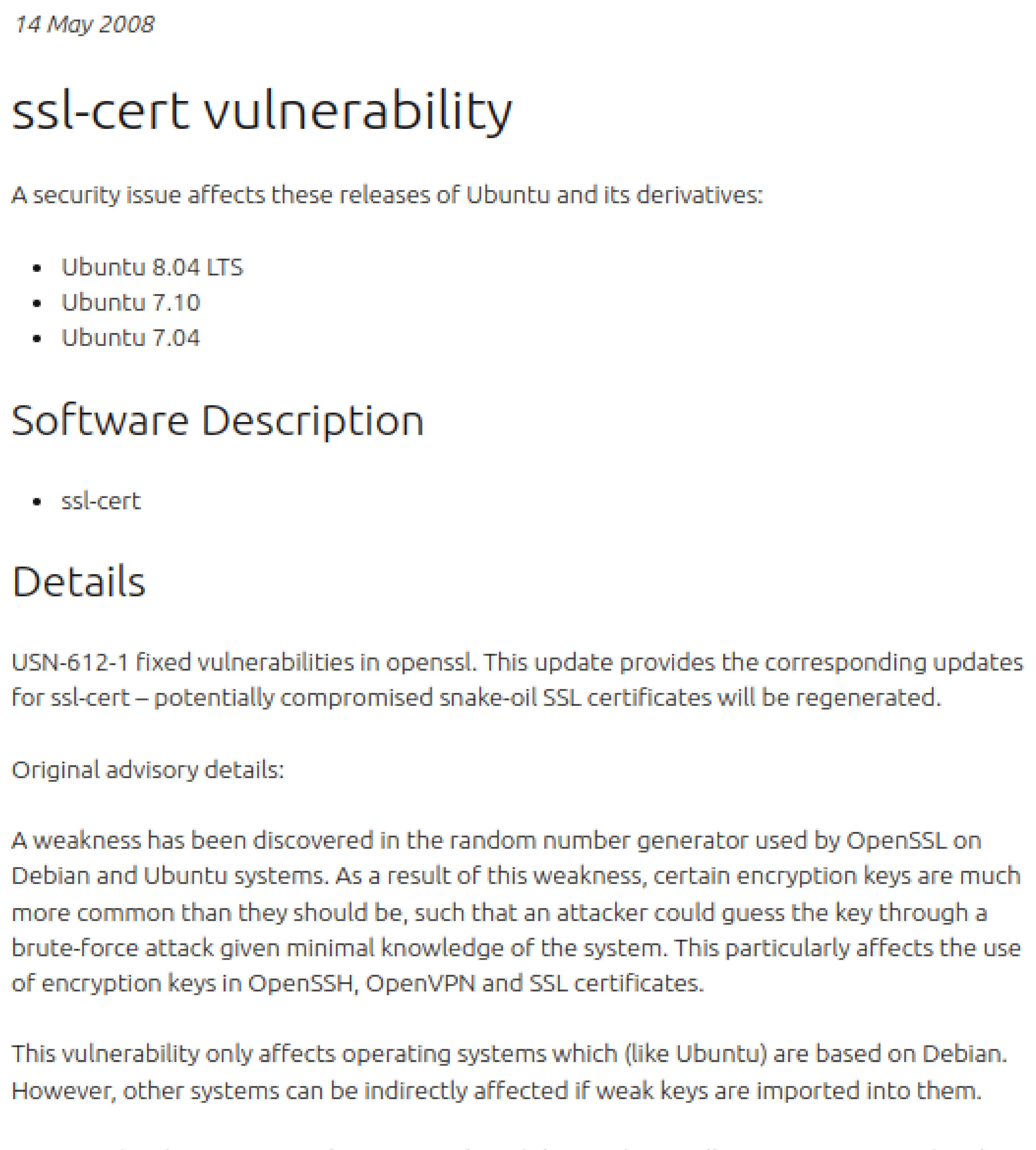
```
alter system set ssl = on;
```

Restart Server

```
# assumes PostgreSQL version 13
    systemctl restart postgresql-13
```

Caveat: SSL is enabled on Linux Debian derivative distributions using the snakeoil certificate

ATTENTION: always consider where the cert originates



14 May 2008

ssl-cert vulnerability

A security issue affects these releases of Ubuntu and its derivatives:

- Ubuntu 8.04 LTS
- Ubuntu 7.10
- Ubuntu 7.04

Software Description

- ssl-cert

Details

USN-612-1 fixed vulnerabilities in openssl. This update provides the corresponding updates for ssl-cert – potentially compromised snake-oil SSL certificates will be regenerated.

Original advisory details:

A weakness has been discovered in the random number generator used by OpenSSL on Debian and Ubuntu systems. As a result of this weakness, certain encryption keys are much more common than they should be, such that an attacker could guess the key through a brute-force attack given minimal knowledge of the system. This particularly affects the use of encryption keys in OpenSSH, OpenVPN and SSL certificates.

This vulnerability only affects operating systems which (like Ubuntu) are based on Debian. However, other systems can be indirectly affected if weak keys are imported into them.

SSL CONNECTIVITY OPTIONS

PERCONA

SSL CONNECTIVITY OPTIONS

SERVER SIDE (pg_hba.conf)
- **host**: client decides
- **hostssl**: server requires SSL
- **hostnossl**: server refuses SSL

CLIENT SIDE (conninfo, sslmode)
- **prefer** (*default*): first try an SSL connection; if that fails, try a non-SSL connection
- **disable**: only try a non-SSL connection
- **allow**: first try a non-SSL connection; if that fails, try an SSL connection
- **require**: only try an SSL connection. If a root CA file is present, verify the certificate in the same way as if verify-ca was specified
- **verify-ca**: only try an SSL connection, and verify that the server certificate is issued by a trusted certificate authority (CA)
- **verify-full**: only try an SSL connection, verify that the server certificate is issued by a trusted CA and that the requested server host name matches that in the certificate

```
# DEFAULT behaviour
postgres@h1:~$ psql 'host=10.231.38.73 user=postgres password=postgres dbname=postgres sslmode=prefer'
psql (13.6 (Ubuntu 13.6-1.pgdg18.04+1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.
postgres=#

# client chooses not to encrypt sessions
postgres@h1:~$ psql 'host=10.231.38.73 user=postgres password=postgres dbname=postgres sslmode=disable'
psql (13.6 (Ubuntu 13.6-1.pgdg18.04+1))
Type "help" for help.
postgres=#
```

CAVEAT
- Always Use SSL sessions for administrative activities
- Recommend using SSL sessions for monitoring
- When SSL required enforce SSL sessions using "**hostssl**"
- Consider security vs performance on client-server connections

EXAMPLE: do you really want the client to choose?

```
psql 'host=10.231.38.73 user=postgres password=postgres dbname=postgres sslmode=prefer'
psql (13.6 (Ubuntu 13.6-1.pgdg18.04+1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# show ssl;
 ssl
-----
 on

# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   all             postgres                                peer
local   all             all                                     md5

hostssl all             all             0.0.0.0/0               md5

host    all             all             0.0.0.0/0               reject
host    all             all             ::0/0                   reject
```

ABOUT SSL CIPHERS

(POSTGRESQL.CONF)

Encryption Cipher Usage

PostgreSQL obtains a list of encryption ciphers and chooses the one to use based upon upon its availability by both client and server. The list is sorted by key length, strength, and excludes ciphers offering neither encryption nor authentication:

```
postgres=# show ssl_ciphers;
       ssl_ciphers
--------------------------
 HIGH:MEDIUM:+3DES:!aNULL
```

Legend: Each cipher string can be optionally preceded by the characters !, - or +:

- **!** : ciphers are permanently deleted from the list
- **-** : ciphers are deleted from the list but can be added again by later options.
- **+** : ciphers are moved to the end of the list.
- @: sort order

EXAMPLE: working with ciphers

```
# list ALL available ciphers, 144 ciphers available on Ubuntu 18.04
# irrespective of strength, encryption or authentication
# sorted by key length

openssl ciphers -v 'ALL:@STRENGTH' | less -N

 1 TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256) Mac=AEAD
 2 TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any      Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
 3 TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
 4 ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(256) Mac=AEAD
 5 ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(256) Mac=AEAD
 ...
140 SEED-SHA                SSLv3 Kx=RSA      Au=RSA  Enc=SEED(128) Mac=SHA1
141 CAMELLIA128-SHA         SSLv3 Kx=RSA      Au=RSA  Enc=Camellia(128) Mac=SHA1
142 PSK-AES128-CBC-SHA256   TLSv1 Kx=PSK      Au=PSK  Enc=AES(128)  Mac=SHA256
143 PSK-AES128-CBC-SHA      SSLv3 Kx=PSK      Au=PSK  Enc=AES(128)  Mac=SHA1
144 PSK-CAMELLIA128-SHA256  TLSv1 Kx=PSK      Au=PSK  Enc=Camellia(128) Mac=SHA256
```

PERCONA

```
# list of 140 of the strongest ciphers (ubuntu 18.04)

openssl ciphers -v 'HIGH' | less -N

 1 TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any        Au=any  Enc=AESGCM(256) Mac=AEAD
 2 TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any   Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
 3 TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any        Au=any  Enc=AESGCM(128) Mac=AEAD
 4 ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(256) Mac=AEAD
 5 ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH   Au=RSA  Enc=AESGCM(256) Mac=AEAD
 …
135 DHE-PSK-CAMELLIA128-SHA256 TLSv1 Kx=DHEPSK   Au=PSK  Enc=Camellia(128) Mac=SHA256
136 AES128-SHA             SSLv3 Kx=RSA      Au=RSA  Enc=AES(128)  Mac=SHA1
137 CAMELLIA128-SHA        SSLv3 Kx=RSA      Au=RSA  Enc=Camellia(128) Mac=SHA1
138 PSK-AES128-CBC-SHA256  TLSv1 Kx=PSK      Au=PSK  Enc=AES(128)  Mac=SHA256
139 PSK-AES128-CBC-SHA     SSLv3 Kx=PSK      Au=PSK  Enc=AES(128)  Mac=SHA1
140 PSK-CAMELLIA128-SHA256  TLSv1 Kx=PSK      Au=PSK  Enc=Camellia(128) Mac=SHA25
```

```
# list medium strength, 7 ciphers, sorted by key length (ubuntu 18.04)

openssl ciphers -v 'MEDIUM:@STRENGTH' | less -N

1 TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256) Mac=AEAD
2 TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any      Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
3 TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
4 DHE-RSA-SEED-SHA        SSLv3 Kx=DH       Au=RSA  Enc=SEED(128) Mac=SHA1
5 DHE-DSS-SEED-SHA        SSLv3 Kx=DH       Au=DSS  Enc=SEED(128) Mac=SHA1
6 ADH-SEED-SHA            SSLv3 Kx=DH       Au=None Enc=SEED(128) Mac=SHA1
7 SEED-SHA                SSLv3 Kx=RSA      Au=RSA  Enc=SEED(128) Mac=SHA1
```

```
# list of 21 ciphers without encryption (ubuntu 18.04)

openssl ciphers -v 'eNULL' | less -N

1 TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256) Mac=AEAD
2 TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any      Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
3 TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
4 ECDHE-ECDSA-NULL-SHA    TLSv1 Kx=ECDH     Au=ECDSA Enc=None      Mac=SHA1
5 ECDHE-RSA-NULL-SHA      TLSv1 Kx=ECDH     Au=RSA  Enc=None      Mac=SHA1
...
17 NULL-SHA                SSLv3 Kx=RSA      Au=RSA  Enc=None      Mac=SHA1
18 NULL-MD5                SSLv3 Kx=RSA      Au=RSA  Enc=None      Mac=MD5
19 PSK-NULL-SHA384         TLSv1 Kx=PSK      Au=PSK  Enc=None      Mac=SHA384
20 PSK-NULL-SHA256         TLSv1 Kx=PSK      Au=PSK  Enc=None      Mac=SHA256
21 PSK-NULL-SHA            SSLv3 Kx=PSK      Au=PSK  Enc=None      Mac=SHA1
```

PERCONA

```
# list of 17 ciphers without authentication (Man-in-the-middle-attacks)
openssl ciphers -v 'aNULL'

# list of 40 ciphers based upon SHA1
openssl ciphers -v 'SHA1'

# list of 36 ciphers based upon SHA256
openssl ciphers -v 'SHA256'

# Identifying the cipher used in the session
postgres=# \d *ssl*
              View "pg_catalog.pg_stat_ssl"
    Column     |  Type   | Collation | Nullable | Default
---------------+---------+-----------+----------+---------
 pid           | integer |           |          |
 ssl           | boolean |           |          |
 version       | text    |           |          |
 cipher        | text    |           |          |
 bits          | integer |           |          |
 compression   | boolean |           |          |
 client_dn     | text    |           |          |
 client_serial | numeric |           |          |
 issuer_dn     | text    |           |          |
```

WORKING WITH ROLES

ISSUES

1. user accounts with too much privilege:

   - a user account that can login to unauthorized databases

   - a user account possessing unnecessary, redundant, escalation privileges such as that of the owner of the database.

2. user accounts used for the wrong task:

   - a super user account used by a monitoring process

   - an account with superuser privileges managing routine application processes. Just try logging into a system when you’re out of connections and see how that works out.

3. default behaviour

   - no restriction to create anything

   - no checks on password strength

   - no imposed life span

Create Role

```
CREATE ROLE name [ [ WITH ] option [ ... ] ]

where option can be:

      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | REPLICATION | NOREPLICATION
    | BYPASSRLS | NOBYPASSRLS
    | CONNECTION LIMIT connlimit
    | [ ENCRYPTED ] PASSWORD 'password' | PASSWORD NULL
    | VALID UNTIL 'timestamp'
    | IN ROLE role_name [, ...]
    | IN GROUP role_name [, ...]
    | ROLE role_name [, ...]
    | ADMIN role_name [, ...]
    | USER role_name [, ...]
    | SYSID uid

URL: https://www.postgresql.org/docs/14/sql-createrole.html
```

Alter Role

```
ALTER ROLE role_specification [ WITH ] option [ ... ]

where option can be:

      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | REPLICATION | NOREPLICATION
    | BYPASSRLS | NOBYPASSRLS
    | CONNECTION LIMIT connlimit
    | [ ENCRYPTED ] PASSWORD 'password' | PASSWORD NULL
    | VALID UNTIL 'timestamp'

ALTER ROLE name RENAME TO new_name

ALTER ROLE { role_specification | ALL } [ IN DATABASE database_name ] SET configuration_parameter { TO | = } { value | DEFAULT }
ALTER ROLE { role_specification | ALL } [ IN DATABASE database_name ] SET configuration_parameter FROM CURRENT
ALTER ROLE { role_specification | ALL } [ IN DATABASE database_name ] RESET configuration_parameter
ALTER ROLE { role_specification | ALL } [ IN DATABASE database_name ] RESET ALL

where role_specification can be:

    role_name
  | CURRENT_ROLE
  | CURRENT_USER
  | SESSION_USER
```

URL: https://www.postgresql.org/docs/14/sql-alterrole.html

Grant Role

```
GRANT role_name [, ...] TO role_specification [, ...]
    [ WITH ADMIN OPTION ]
    [ GRANTED BY role_specification ]

where role_specification can be:

    [ GROUP ] role_name
  | PUBLIC
  | CURRENT_ROLE
  | CURRENT_USER
  | SESSION_USER
```

Revoke Role

```
REVOKE [ ADMIN OPTION FOR ]
    role_name [, ...] FROM role_specification [, ...]
    [ GRANTED BY role_specification ]
    [ CASCADE | RESTRICT ]

where role_specification can be:

    [ GROUP ] role_name
  | PUBLIC
  | CURRENT_ROLE
  | CURRENT_USER
  | SESSION_USER
```

PERCONA

WORKING WITH ROLES

About Passwords

PERCONA

Enforcing Strong Passwords

Available Password Complexity and Length Enforcement Mechanisms

3rd Party Mechanisms:

- ldap: yes (requires some effort) gss
- gss: yes
- gspi: yes
- pam: yes

PostgreSQL:

- default: no control of any kind
- PostgreSQL extension: **passwordcheck**
  - default: somewhat/sorta
  - patch password module src enabling use of cracklib: yes

Working With Extension “passwordcheck”

The passwordcheck module enforces a few simple rules for password strength length, mixing numbers and letters.

EXAMPLE: installing passwordcheck

```
-- add passwordcheck library
alter system set shared_preload_libraries='passwordcheck';

# restart service
systemctl restart postgresql-13

postgres=# create role usr1 with login password '123';
ERROR:  password is too short

postgres=# create role usr1 with login password 'password';
ERROR:  password must contain both letters and nonletters

postgres=# create role usr1 with login password '1234abc789';
CREATE ROLE
```

Enhancing passwordcheck With cracklib

You can adapt module “passwordcheck” using CrackLib by recompiling the module’s source code:

- Install the development libraries for Cracklib
  i.e. `apt install libcrack2-dev`
- Uncomment two lines in the Makefile

```
vi $SRC/contrib/passwordcheck/Makefile

# uncomment the following two lines to enable cracklib support
PG_CPPFLAGS = -DUSE_CRACKLIB '-DCRACKLIB_DICTPATH="/usr/lib/cracklib_dict"'
SHLIB_LINK = -lcrack
```

- Rebuild and up the module in the PostgreSQL binary path

```
make
cp $SRC/contrib/passwordcheck/passwordcheck.so $BIN/lib/postgresql/passwordcheck.so
```

NB: It was necessary performing these additional operations on Ubuntu 20.04

```
cp /var/cache/cracklib/cracklib_dict.pwi /usr/lib/cracklib_dict.pwi
gzip -c /var/cache/cracklib/cracklib_dict.pwd > /usr/lib/cracklib_dict.pwd.gz
```

Using passwordcheck With cracklib

```
-- what worked before now fails
postgres=# alter role usr1 with login password '1234abc789';
ERROR:  password is easily cracked

-- this works
postgres=# alter role usr1 with login password 'FjCEo13KjY32u';
ALTER ROLE
```

PERCONA

Thank You!

PERCONA

Questions?

PERCONA

Open Source Database Experts

PERCONA