MLS PostgreSQL

Joe Conway
joe.conway@crunchydata.com
mail@joeconway.com

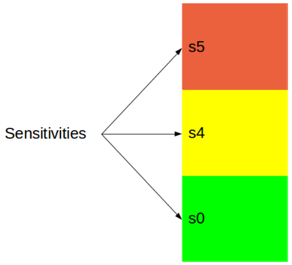
Crunchy Data

May 20, 2016

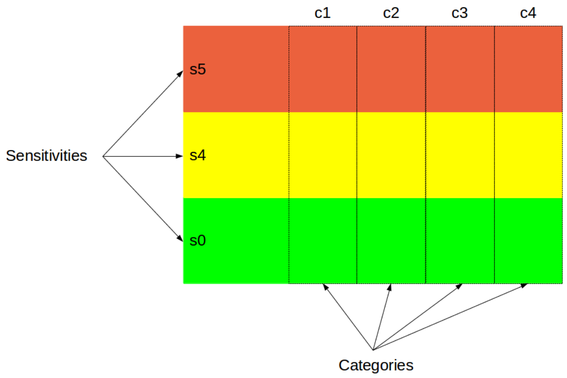
Agenda

- Introduction
  - 50,000 ft Perspective
- Solution Components
  - RLS
  - SELinux
  - sepgsql
- Configuration and Setup
  - Operating System
  - sepgsql
  - Database schema/DDL
- Results

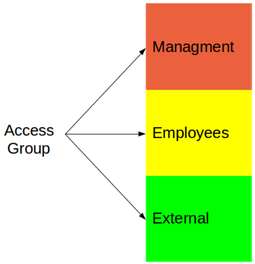
What is MLS?



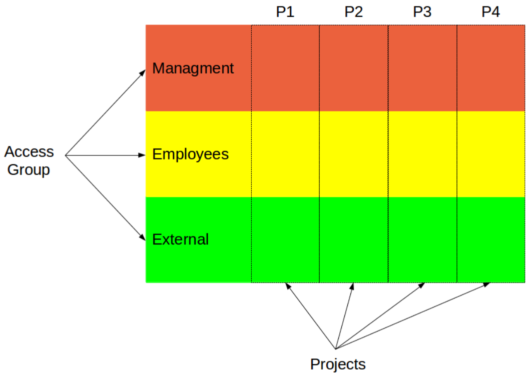
What is MLS?



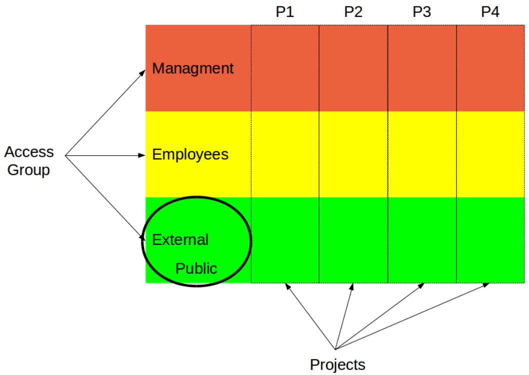
Example Use-case



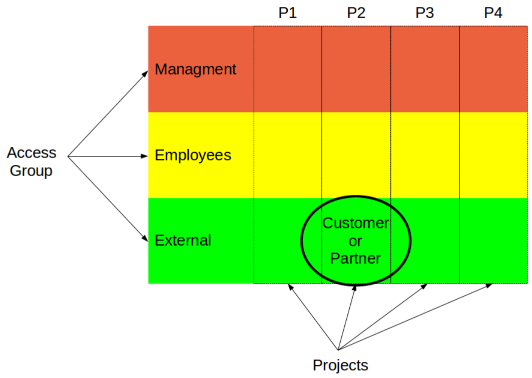
Example Use-case



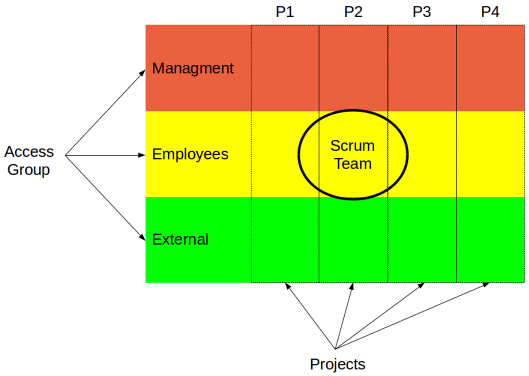
Example Use-case



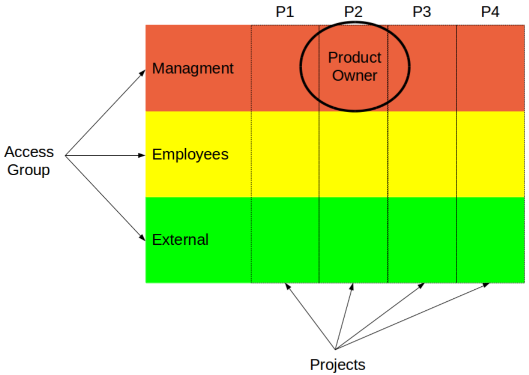
Example Use-case



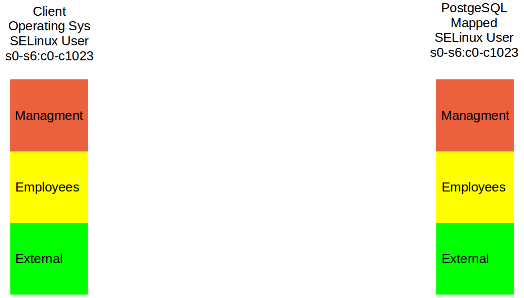
Example Use-case



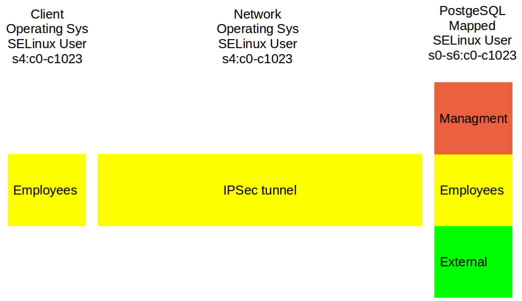
Example Use-case



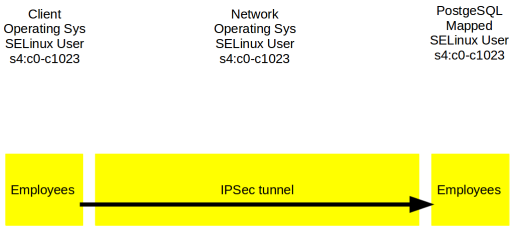
Example Use-case



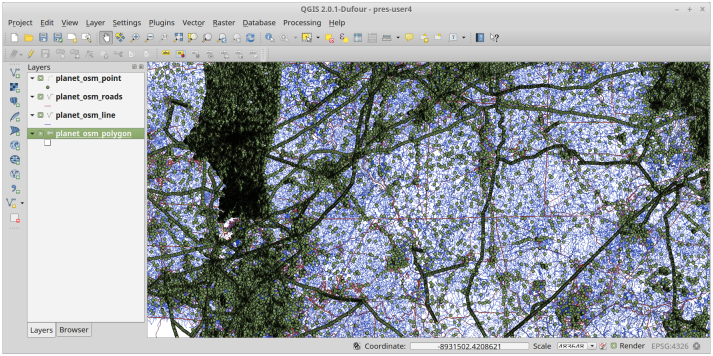
Example Use-case



Example Use-case



Demo


QGIS 2.0.1-Dufour - pres-user4
Project Edit View Layer Settings Plugins Vector Raster Database Processing Help
Layers
planet_osm_point
planet_osm_roads
planet_osm_line
planet_osm_polygon
Layers Browser
Coordinate: -8931502,4208621
Scale 483648
Render
EPSG:4326

Business Case

- Why not just create separate database for each level?
    - Redundant hardware
    - Inhibits reporting and analysis
    - Data duplication
- What about filtering and enforcement by application?
    - Database provides integrity
    - RLS is transparent and performs well

Row Level Security

- New feature in PostgreSQL 9.5
- Enabled on per-table basis
- Enforced with POLICY
    - USING expression (old row)
    - WITH CHECK expression (new row)

Row Level Security - Typical Example

```
CREATE USER bob;
CREATE USER alice;

CREATE TABLE m1 (id int primary key, f1 text, app_user text);
INSERT INTO m1 VALUES(1,'a','bob');
INSERT INTO m1 VALUES(2,'b','alice');
ALTER TABLE m1 ENABLE ROW LEVEL SECURITY;
CREATE POLICY P ON m1 USING (app_user = current_user);
GRANT SELECT ON m1 TO public;
```

Row Level Security - Typical Example

```
SELECT * FROM m1;
 id | f1 | app_user
----+----+----------
  1 | a  | bob
  2 | b  | alice

SET SESSION AUTHORIZATION bob;
SELECT * FROM m1;
 id | f1 | app_user
----+----+----------
  1 | a  | bob

SET SESSION AUTHORIZATION alice;
SELECT * FROM m1;
 id | f1 | app_user
----+----+----------
  2 | b  | alice
```

Security Enhanced Linux

- SELinux: Mandatory Access Control (MAC)
- Versus: Discretionary Access Control (DAC)
- Enforced in kernel space
- Managed via Reference Policy
  - Targeted Policy
  - MLS Policy
- Customized via Policy Modules

https://people.redhat.com/duffy/selinux/selinux-coloring-book_A4-Stapled.pdf

MLS Reference Policy

- Based on Bell-LaPadula model
  - Read-down
  - Write-up
- Modified for Write-equals

Security Context

- `<user>:<role>:<domain>:<sensitivity>:<category>`
    - `<user>` = SElinux user
    - `<role>` = SElinux role
    - `<domain>` = type
    - `<sensitivity>` = low to high, e.g. s0, s1, . . . s15
    - `<category>` = compartmentalization label
- `<level>` = `<sensitivity>:<category>`
- Examples

```
dbs6_u:dbclient_r:dbclient_t:s0
system_u:object_r:sepgsql_table_t:s0-s15:c0.c1023
```

Security Access Decision

- Subject Context (PostgreSQL user)
- Object/Target Context (table, row, etc.)
- Permission (e.g. select, update, etc.)
- Type Enforcement
  - Subject type needs requested permission on object type
- MLS Enforcement
  - Subject Sensitivity (s0-s15) must dominate Object
    $\Rightarrow$ e.g. s5 dominates s3
  - Subject Category (c0.c1023) must include Object category
    $\Rightarrow$ e.g. s5:c1.c5 does not include s3:c42

sepgsql Extension

- PostgreSQL supports `SECURITY LABEL` command
- Label Provider uses the label
- Security label used for SELinux Object context
- Customized with additional functionality
  - Mapping of database user to SELinux user
  - Subject context transition based on postgres user and network peer context
  - `sepgsql_check_row_label()`
  - `sepgsql_create_row_label()`

sepgsql_check_row_label(arg1 [, arg2])

- Object context: arg1 - row security_label
- Subject context: client - SELinux user+network
- Permission Type: default `select`, otherwise arg2:
    - `select`, `insert`, `update`, `delete`
    - `relabelfrom`, `relabelto`
- Access decision: SELinux

sepgsql_check_row_label(arg1 [, arg2])

```
select sepgsql_getcon();
           sepgsql_getcon
------------------------------------
 dbs5_u:dbclient_r:dbclient_t:s5:c1

SELECT
 sepgsql_check_row_label
 ('system_u:object_r:sepgsql_table_t:s0') as s0sel,
 sepgsql_check_row_label
 ('system_u:object_r:sepgsql_table_t:s6') as s6sel;
 s0sel | s6sel
-------+-------
 t     | f
```

sepgsql_check_row_label(arg1 [, arg2])

```
select sepgsql_getcon();
           sepgsql_getcon
------------------------------------
 dbs5_u:dbclient_r:dbclient_t:s5:c1

SELECT
 sepgsql_check_row_label
 ('system_u:object_r:sepgsql_table_t:s0','delete') as s0del,
 sepgsql_check_row_label
 ('system_u:object_r:sepgsql_table_t:s5','delete') as s5del,
 sepgsql_check_row_label
 ('system_u:object_r:sepgsql_table_t:s5:c1','delete') as s5c1del;
 s0del | s5del | s5c1del
-------+-------+---------
 f     | f     | t
```

sepgsql_create_row_label(table_oid)

- Object context: Table security label
- Subject context: client - SELinux user+network
- Derives security_label context, typically used for a row

```
CREATE OR REPLACE FUNCTION get_table_label(tableoid oid)
RETURNS text AS $$
 SELECT label FROM pg_seclabels WHERE objoid = tableoid
 AND objtype = 'table'
$$ LANGUAGE sql;

\x
SELECT get_table_label('m1'::regclass) AS tcontext,
 sepgsql_getcon() AS scontext,
 sepgsql_create_row_label('m1'::regclass) AS security_label;
-[ RECORD 1 ]--+-----------------------------------------
tcontext       | system_u:object_r:sepgsql_table_t:s0-s15:c0.c1023
scontext       | dbs5_u:dbclient_r:dbclient_t:s5:c1
security_label | dbs5_u:object_r:sepgsql_table_t:s5:c1
```

Operating System and Networking

- Red Hat or CentOS 7.2
- With additional SElinux packages
- Network Interfaces
    - Admin subnet and subnet per security level
    - Or use Labeled IPSec
- Routes
- netlabel
- sshd
- firewalld

SELinux - Configuration

- Install custom policy modules
- Create SELinux users
- Build, configure, and install custom sepgsql
- Map database users to SELinux users

SELinux Roles and Mapping

SELinux Role	Role Type	PG Roles	Permissions
dbadm_r	OS	Cluster Owner	Admin DB at OS level
dbsec_r	OS, PgSQL	Sec Admin	Edit security-related files, relabel DB objects
dbsu_r	PgSQL	Superuser	Manage objects within DB
dbstaff_r	PgSQL	Staff User	Perform admin tasks
dbclient_r	PgSQL	Client	CRUD operations
dbguest_r	PgSQL	Client	Read-only operations

PostgreSQL - Custom Module

- Build and Configure custom sepgsql
- Adjust some normal PostgreSQL configuration too

```
cd /opt/src/mls/crunchy-selinux-pgsql
USE_PGXS=1 make
USE_PGXS=1 make install

cat >> /var/lib/pgsql/9.5/data/postgresql.conf << \EOF
listen_addresses = '*'
row_security = on
shared_preload_libraries = 'crunchy-selinux-pgsql'

sepgsql.enable_user_transition = on
sepgsql.default_selinux_user = 'dbguest_u'
sepgsql.force_rls = on
EOF
```

Table Definition

```
CREATE TABLE m1 (
  a int,
  b text,
  security_label text DEFAULT
  sepgsql_create_row_label('m1'::regclass::oid)
);

-- Grant permissions to table
GRANT ALL ON TABLE m1 TO user1, user2, user3, user4;

-- Enable Row Level Security on table.
ALTER TABLE m1 ENABLE ROW LEVEL SECURITY;
```

Table Definition

```
-- Create Row Level MLS policies.
CREATE POLICY mls_select ON m1 FOR SELECT
  USING (sepgsql_check_row_label(security_label));

CREATE POLICY mls_insert ON m1 FOR INSERT WITH CHECK
 (sepgsql_create_row_label('m1'::regclass::oid) = security_label);

CREATE POLICY mls_update ON m1 FOR UPDATE
 USING (sepgsql_check_row_label(security_label))
 WITH CHECK (sepgsql_check_row_label(security_label,'update'));

CREATE POLICY mls_delete ON m1 FOR DELETE
 USING (sepgsql_check_row_label(security_label,'delete'));
```

User Level Versus Subnet Level

```
# s0 user, s4 subnet
psql -h 192.168.6.119 -p 5432 -U user1 mls
Password for user user1:
psql: FATAL:  SELinux: unable to get default context
              for user: user1 (dbs0_u)

# s0 user, s0 subnet
psql -qAt -h 192.168.5.119 -p 5432 -U user1 mls \
 -c "select sepgsql_getcon()"
Password for user user1:
dbs0_u:dbclient_r:dbclient_t:s0

# s6 user, s0 subnet
psql -qAt -h 192.168.5.119 -p 5432 -U user4 mls \
 -c "select sepgsql_getcon()"
Password for user user4:
dbs6_u:dbclient_r:dbclient_t:s0
```

SELECT on s0 Subnet

```
# s0 user, s0 subnet
psql -h 192.168.5.119 -p 5432 -U user1 mls \
 -c "select * from m1"
Password for user user1:
 a | b |             security_label
---+---+--------------------------------------
 1 | a | system_u:object_r:sepgsql_table_t:s0
(1 row)

# s6 user, s0 subnet
psql -h 192.168.5.119 -p 5432 -U user4 mls \
 -c "select * from m1"
Password for user user4:
 a | b |             security_label
---+---+--------------------------------------
 1 | a | system_u:object_r:sepgsql_table_t:s0
(1 row)
```

user4 SELECT on s6 Subnet

```
# s6 user, s6 subnet
psql -h 192.168.8.119 -p 5432 -U user4 mls \
 -c "select * from m1"
Password for user user4:
 a | b |               security_label
---+---+---------------------------------------
 1 | a | system_u:object_r:sepgsql_table_t:s0
 2 | b | system_u:object_r:sepgsql_table_t:s4:c1
 3 | c | system_u:object_r:sepgsql_table_t:s5:c1
 4 | d | system_u:object_r:sepgsql_table_t:s6:c1
(4 rows)
```

INSERT on s0 Subnet

```
# s0 user, s0 subnet
psql -h 192.168.5.119 -p 5432 -U user1 mls \
 -c "insert into m1(a,b) values (11,'a1') returning *"
Password for user user1:
 a  | b  |             security_label
----+----+-----------------------------------
 11 | a1 | dbs0_u:object_r:sepgsql_table_t:s0
(1 row)

# s6 user, s0 subnet
psql -h 192.168.5.119 -p 5432 -U user4 mls \
 -c "insert into m1(a,b) values (41,'a1') returning *"
Password for user user4:
 a  | b  |             security_label
----+----+-----------------------------------
 41 | a1 | dbs6_u:object_r:sepgsql_table_t:s0
(1 row)
```

INSERT on s6 Subnet

```
# s6 user, s6 subnet
psql -h 192.168.8.119 -p 5432 -U user4 mls \
 -c "insert into m1(a,b) values (441,'d1') returning *"
Password for user user4:
  a  | b  |              security_label
-----+----+-----------------------------------------
 441 | d1 | dbs6_u:object_r:sepgsql_table_t:s6:c1
(1 row)
```

UPDATE on s0 Subnet

```
# s0 user, s0 subnet, s0 row
psql -h 192.168.5.119 -p 5432 -U user1 mls \
 -c "update m1 set b = 'a1a' where a = 11 returning *"
Password for user user1:
 a  |  b  |           security_label
----+-----+------------------------------------
 11 | a1a | dbs0_u:object_r:sepgsql_table_t:s0
(1 row)

# s6 user, s0 subnet, s0 row
psql -h 192.168.5.119 -p 5432 -U user4 mls \
 -c "update m1 set b = 'd1d' where a = 41 returning *"
Password for user user4:
 a  |  b  |           security_label
----+-----+------------------------------------
 41 | d1d | dbs6_u:object_r:sepgsql_table_t:s0
(1 row)
```

UPDATE on s6 Subnet

```
# s6 user, s6 subnet, s6 row
psql -h 192.168.8.119 -p 5432 -U user4 mls \
 -c "update m1 set b = 'd1d' where a = 441 returning *"
Password for user user4:
  a  |  b  |           security_label
-----+-----+------------------------------------
 441 | d1d | dbs6_u:object_r:sepgsql_table_t:s6:c1
(1 row)

# however...s6 user, s6 subnet, s0 row
psql -h 192.168.8.119 -p 5432 -U user4 mls \
 -c "update m1 set b = 'd1d1' where a = 41 returning *"
Password for user user4:
ERROR:  new row violates row-level security policy for table "m1"
```

Performance Testing

- Compare t1 (RLS/MLS), r1 (Simple RLS), u1 (no RLS)
- 10 million rows per table
- 4 levels, 25% each
- INSERT test
- SELECT one row
- SELECT 50,000 rows

Performance - INSERT

```
WITH s(c) AS -- RLS/MLS case
(SELECT sepgsql_create_row_label('t1'::regclass::oid))
INSERT INTO t1 SELECT g.i, g.i::text, s.c
FROM generate_series(1, 10000000, 4) as g(i), s;
--Total Time: 22,268.697 ms

WITH s(c) AS -- RLS case
(SELECT sepgsql_create_row_label('r1'::regclass::oid))
INSERT INTO r1 SELECT g.i, g.i::text, s.c
FROM generate_series(1, 10000000, 4) as g(i), s;
--Total Time: 20,309.843 ms

WITH s(c) AS -- no RLS case
(SELECT sepgsql_create_row_label('u1'::regclass::oid))
INSERT INTO u1 SELECT g.i, g.i::text, s.c
FROM generate_series(1, 10000000, 4) as g(i), s;
--Total Time: 27,228.559 ms
```

Performance - SELECT

```
-- SELECT 1 row
SELECT * FROM t1 WHERE a = 40;
-- Avg Time (10 runs): 0.7895 ms
SELECT * FROM r1 WHERE a = 40;
-- Avg Time (10 runs): 0.6829 ms
SELECT * FROM u1 WHERE a = 40;
-- Avg Time (10 runs): 0.5587 ms

-- SELECT 50k rows
SELECT count(1) FROM t1 WHERE a >=0 AND a <= 200000;
-- Avg Time (10 runs): 81.0375 ms
SELECT count(1) FROM r1 WHERE a >=0 AND a <= 200000;
-- Avg Time (10 runs): 46.0586 ms
SELECT count(1) FROM u1 WHERE a >=0 AND a <= 200000 AND a % 4 = 0;
-- Avg Time (10 runs): 55.9427 ms
```

Questions?

Thank You!
mail@joeconway.com