Empowered by Innovation

SE-PostgreSQL

System-wide consistency of access control

NEC OSS Promotion Center

KaiGai Kohei <kaigai@ak.jp.nec.com>

Self Introduction

Name	KaiGai Kohei
Company	NEC, OSS Promotion Center
Works	7 years experiences of OSS development

- » SELinux
- » **PostgreSQL**
- » Memcached
- » Apache (mod_selinux)

SE-PostgreSQL Project

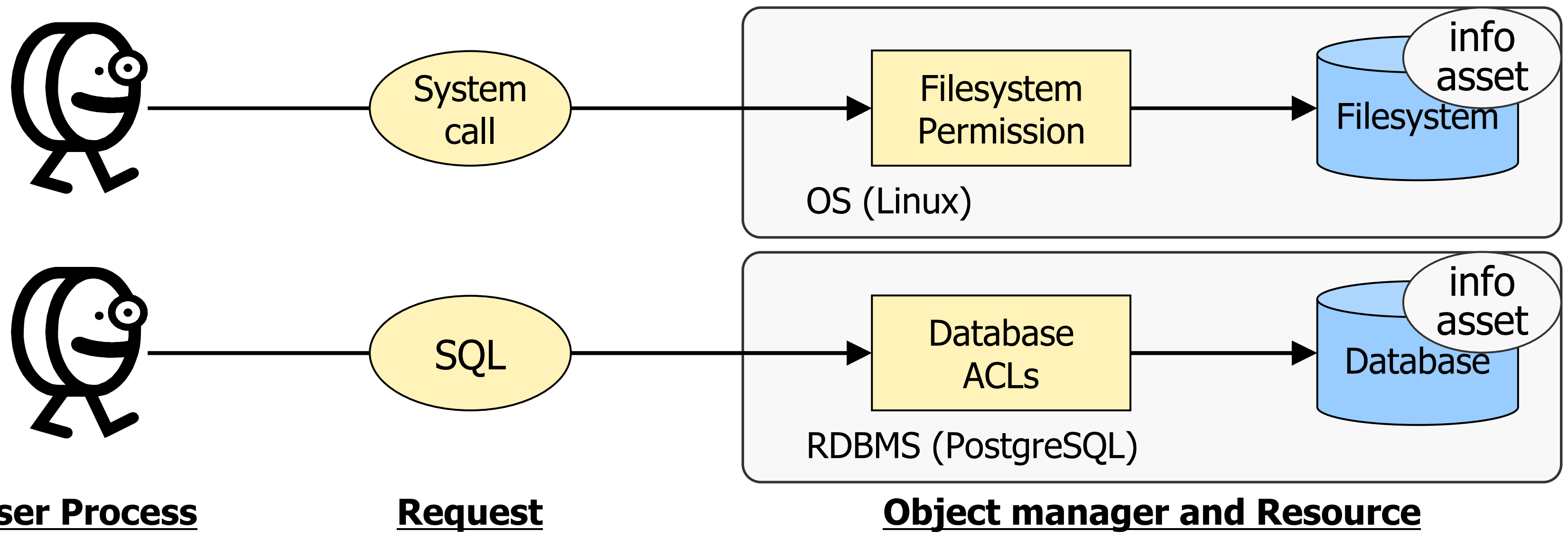
- It enables to control accesses to database objects using a centralized security policy of SELinux.
- Launched at 2006, then I've worked together both of SELinux and PostgreSQL community.
- Now, under development as a plugin for PostgreSQL v9.1.

Agenda

1. The Goal of this project
2. Architecture of SE-PostgreSQL
3. Playing with SE-PostgreSQL (demonstration)
4. Today, and the Future

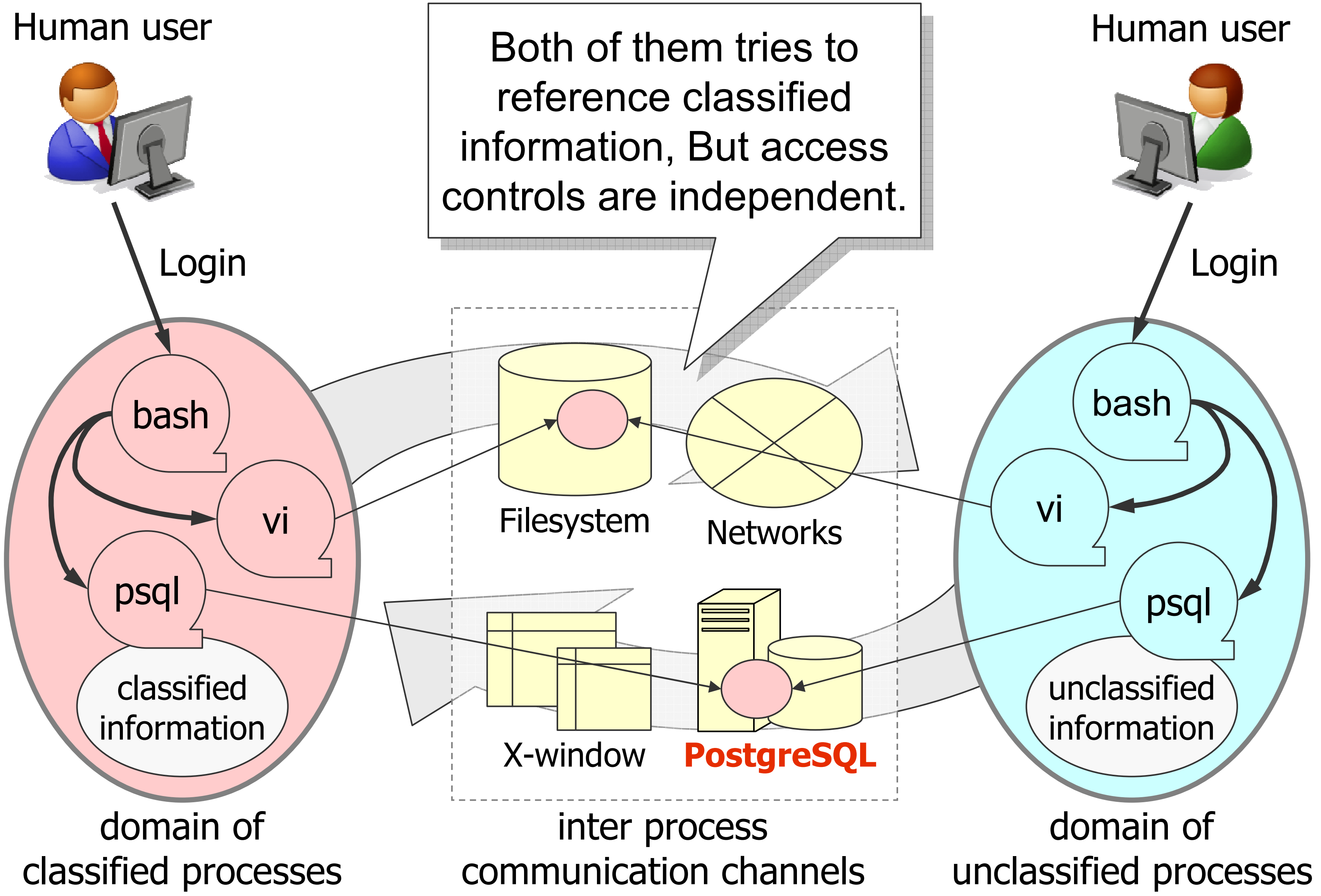
1. The Goal of this Project

An analogy on Filesystem and Database



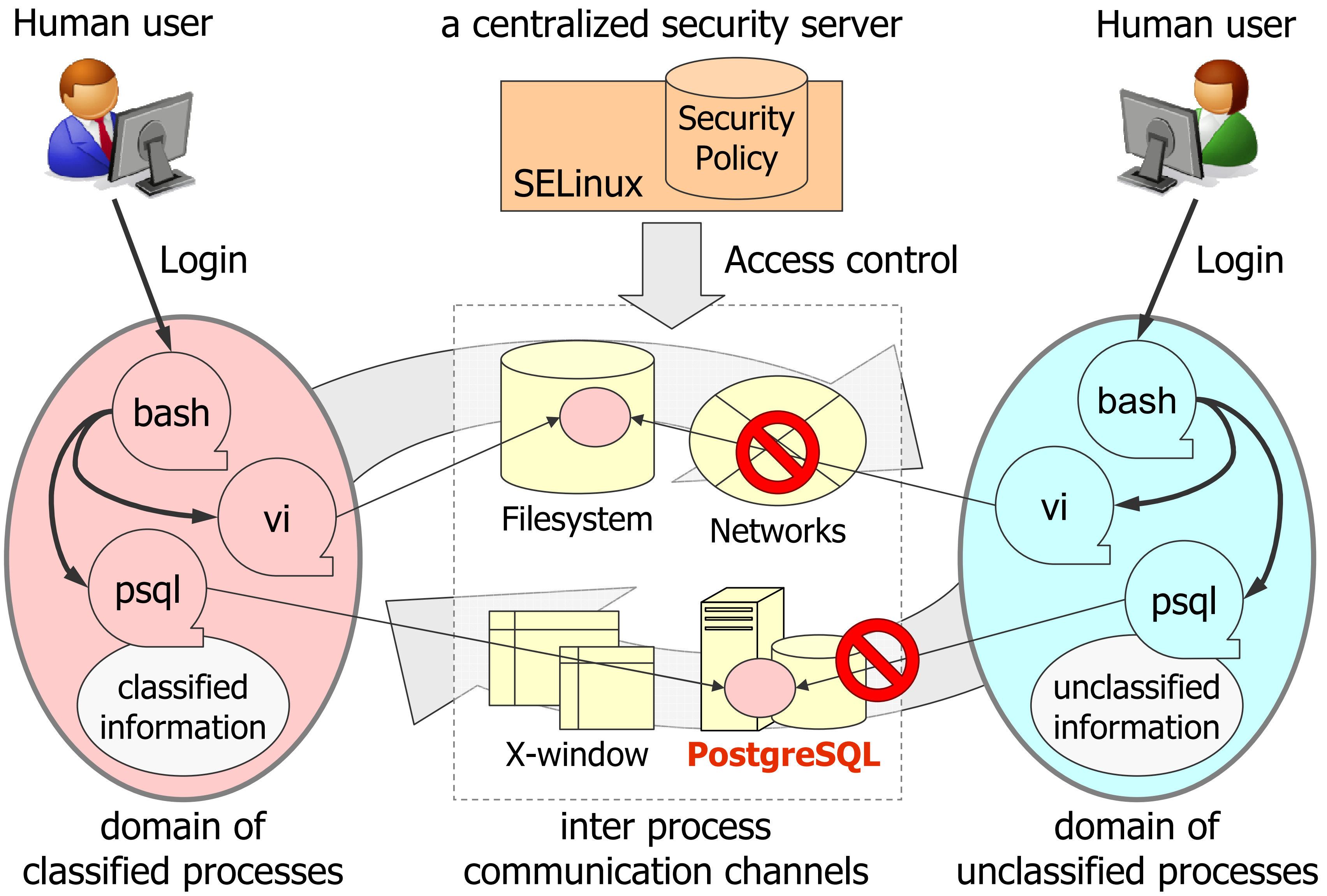
- Same relationship on user processes, requests, object manager and information assets.
- Differences in the way to store and access them
  - System call for Filesystem
  - SQL for Databases
- Also differences in access control model.
  - ➡ what does it make differences in the result?

The Goal of this project (1/2)

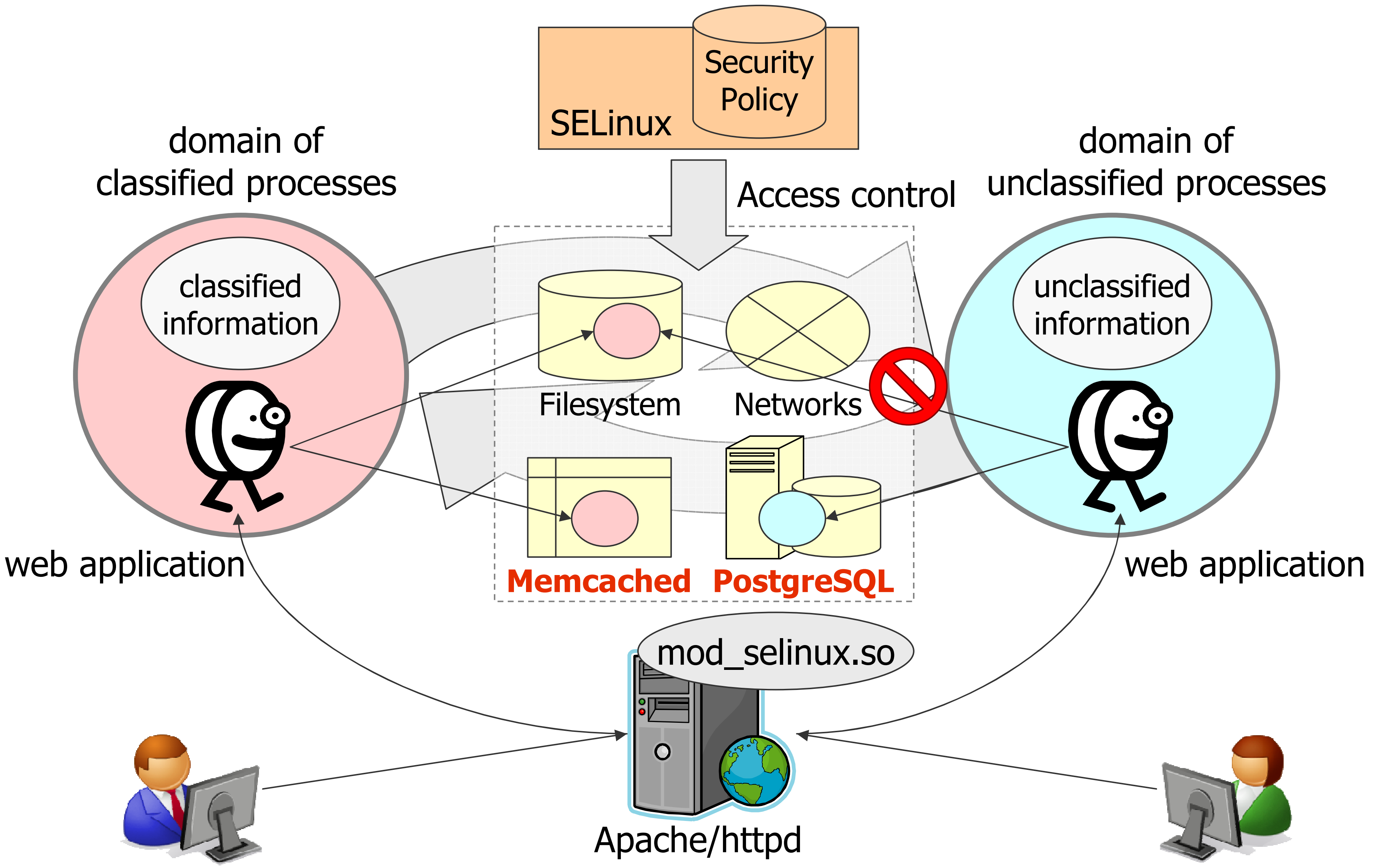

Human user
Login
Both of them tries to reference classified information, But access controls are independent.
Human user
Login
bash
vi
psql
classified information
Filesystem
Networks
X-window
PostgreSQL
bash
vi
psql
unclassified information
domain of classified processes
inter process communication channels
domain of unclassified processes

Lack of conductor

The Goal of this project (2/2)



OT: LAMP/SELinux



SELinux as a Security Server (1/3)

Interactions with object managers

- Kernel subsystems do queries via LSM.
- Userspace applications do queries via libselinux.
- ➡ Both of them control user's requests according to the decision.

Security context as a common identifier

```
system_u:system_r:postgresql_t:s0
system_u:object_r:sepgsql_table_t:s0
```

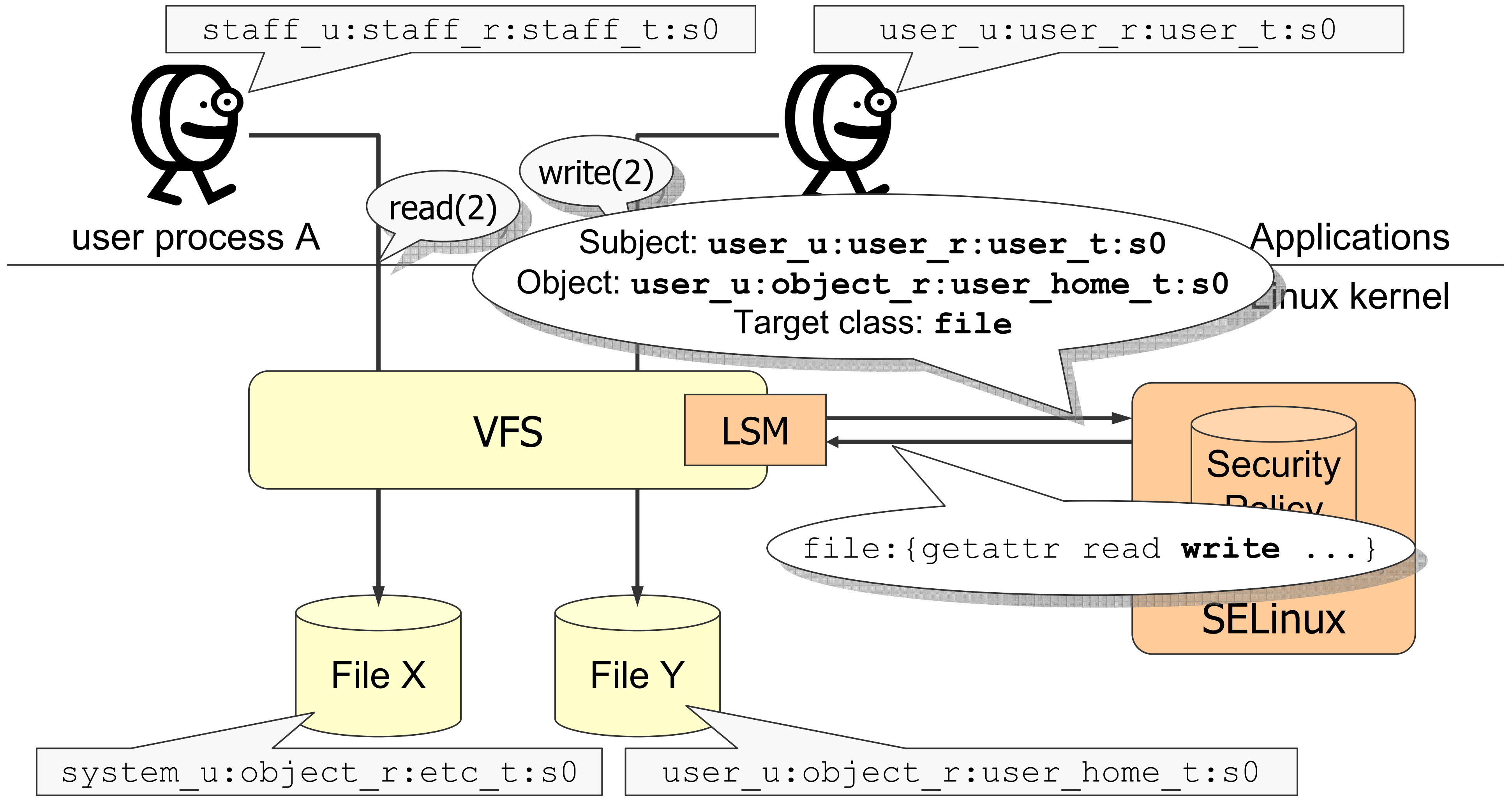
- ➡ A short formatted text, independent from object classes.

Security policy

- A massive set of access control rules.
- A rule describes a set of actions to be allowed on a pair of a security context of the subject (process being accessing) and a security context of the object being accessed.

SELinux as a Security Server (2/3)

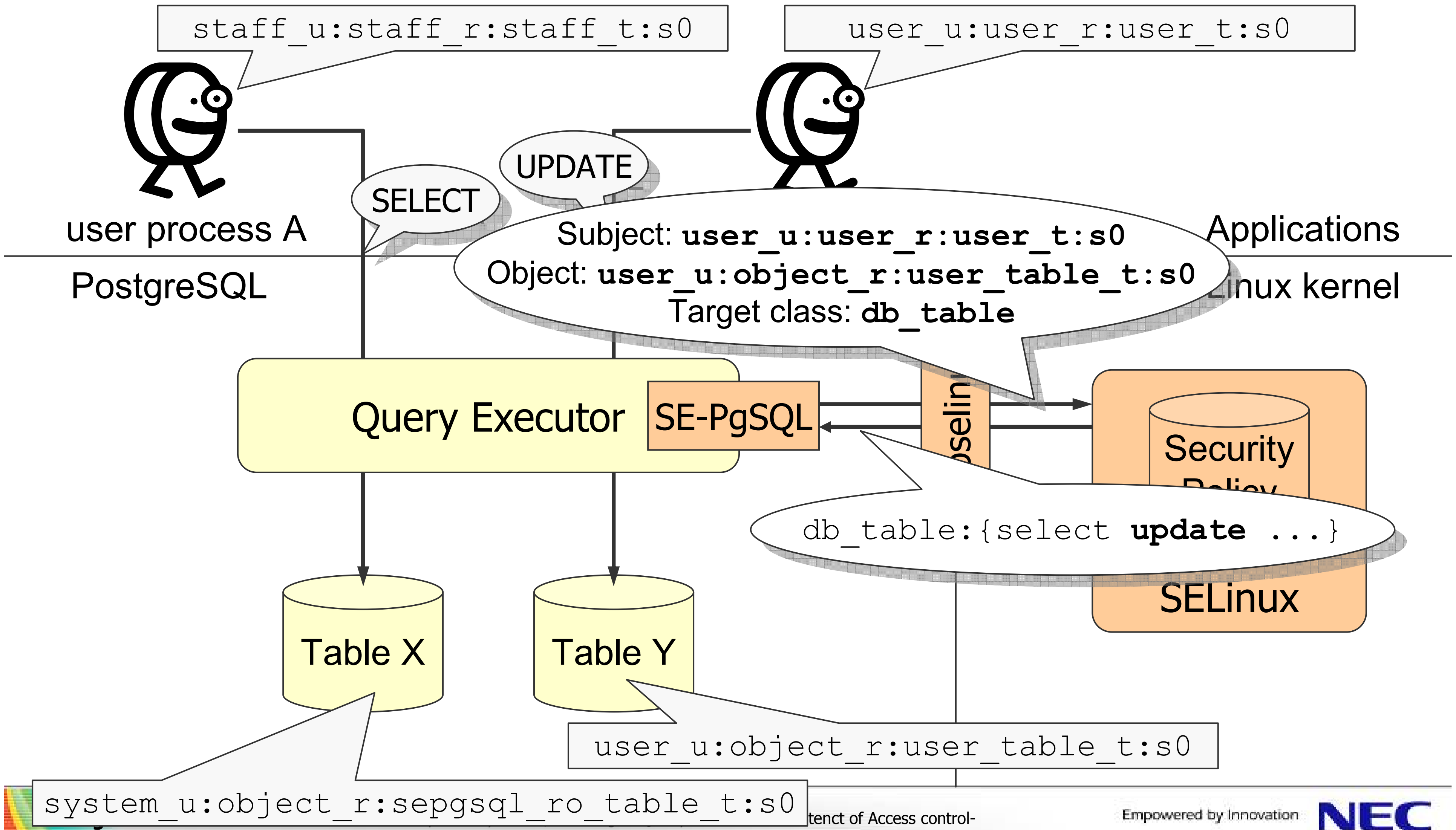
Case of Linux Kernel



staff_u:staff_r:staff_t:s0
user_u:user_r:user_t:s0
user process A
read(2)
write(2)
Subject: user_u:user_r:user_t:s0
Object: user_u:object_r:user_home_t:s0
Target class: file
Applications
Linux kernel
VFS
LSM
Security Policy
file:{getattr read write ...}
SELinux
File X
File Y
system_u:object_r:etc_t:s0
user_u:object_r:user_home_t:s0

SELinux as a Security Server (3/3)

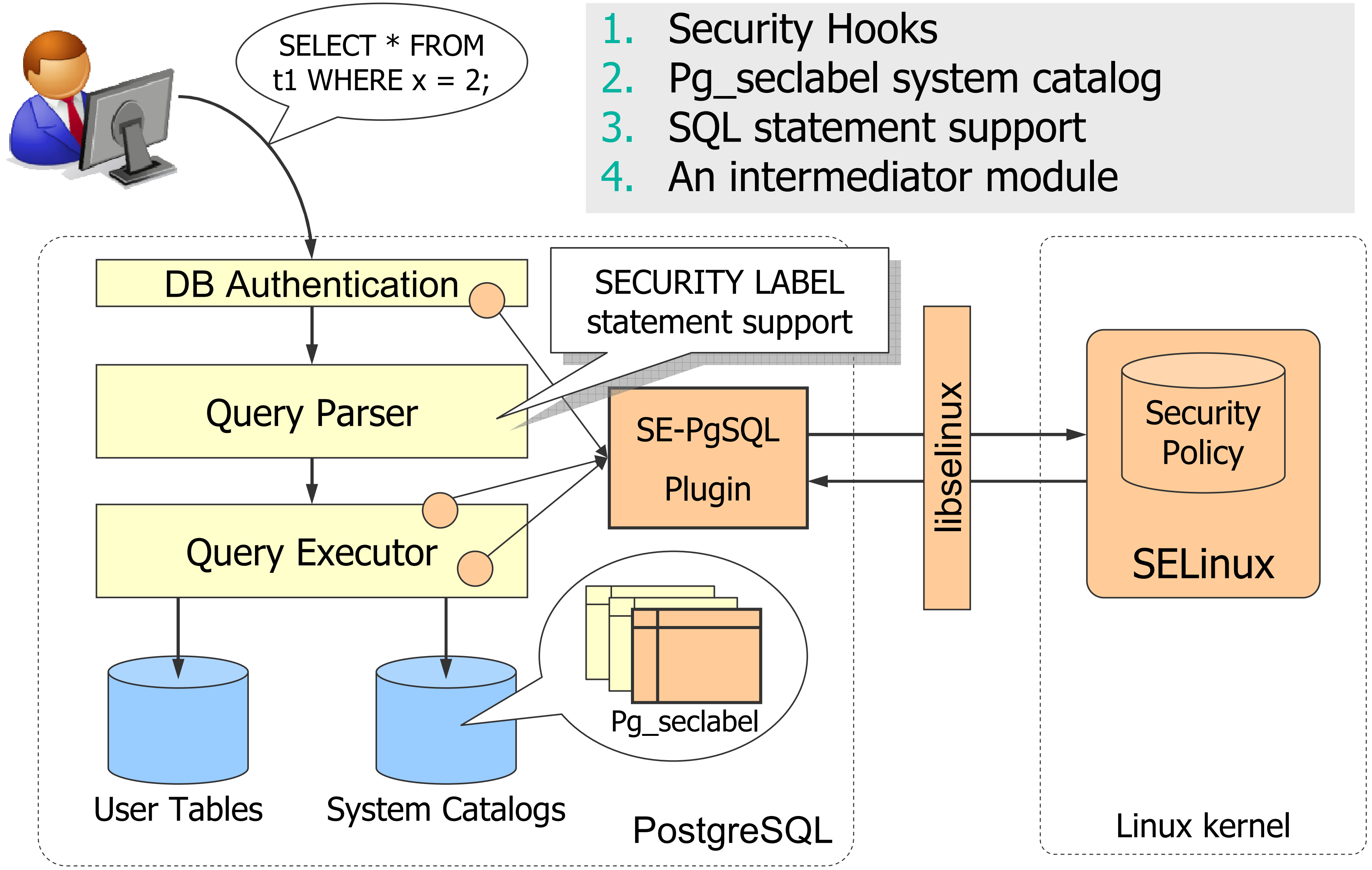
Case of PostgreSQL


staff_u:staff_r:staff_t:s0
user_u:user_r:user_t:s0
UPDATE
SELECT
user process A
Applications
PostgreSQL
Linux kernel
Subject: user_u:user_r:user_t:s0
Object: user_u:object_r:user_table_t:s0
Target class: db_table
Query Executor
SE-PgSQL
selinux
Security Policy
db_table:{select update ...}
SELinux
Table X
Table Y
user_u:object_r:user_table_t:s0
system_u:object_r:sepgsql_ro_table_t:s0

2. Architecture of SE-PostgreSQL

What was necessary to be enhanced

1. Security Hooks
2. Pg_seclabel system catalog
3. SQL statement support
4. An intermediator module



Idea of External Security Provider

Background

- Earlier version of SE-PostgreSQL was launched at 2006
- Not an easy path to get merged, because of ...
  - A large scale patch, even if minimum functionalities
  - Few people are familiar with SELinux in PgSQL community
  - Being not neutral to other security mechanism

Idea of External Security Provider (ESP)

- Similar idea to LSM, XACE
- PG provides a set of security hooks which allow third party plugins to make its access control decision.
  - The patch can be broken up to smaller pieces.
  - SELinux specific code can be moved into the plugin modules.
  - Being open to the upcoming other security models
- The first version of ESP shall be bundled in v9.1.

Security Hooks (1/2)

```
bool
ExecCheckRTPerms(List *rangeTable, bool ereport_on_violation)
{
          :
    if (ExecutorCheckPerms_hook)
        result = (*ExecutorCheckPerms_hook)(rangeTable,
                                            ereport_on_violation);
    return result;
}
```

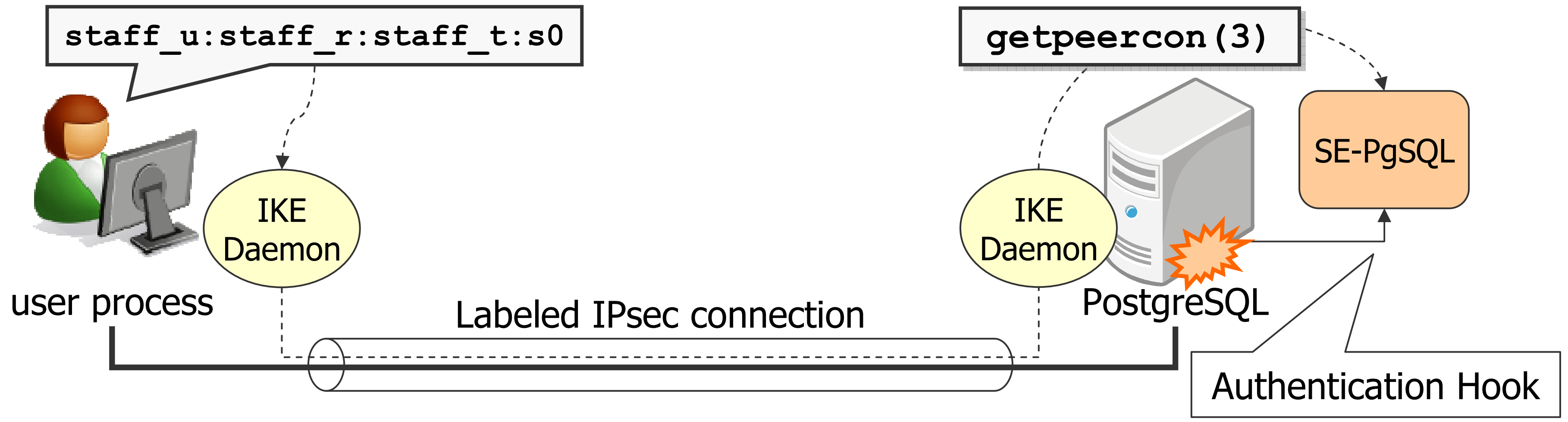
bool sepgsql_relation_privileges(...)

ExecCheckRTPerms()

- It is a routine to check permissions on DMSs
- List of RangeTblEntry contains all the necessary information.
  - OID of the relation to be referenced
  - A flag of required privileges (e.g, ACL_SELECT, ACL_UPDATE, ...)

➡ The ESP hook allows plugins to make its access control decision. If violated, it raises and returns an error according to the spec.

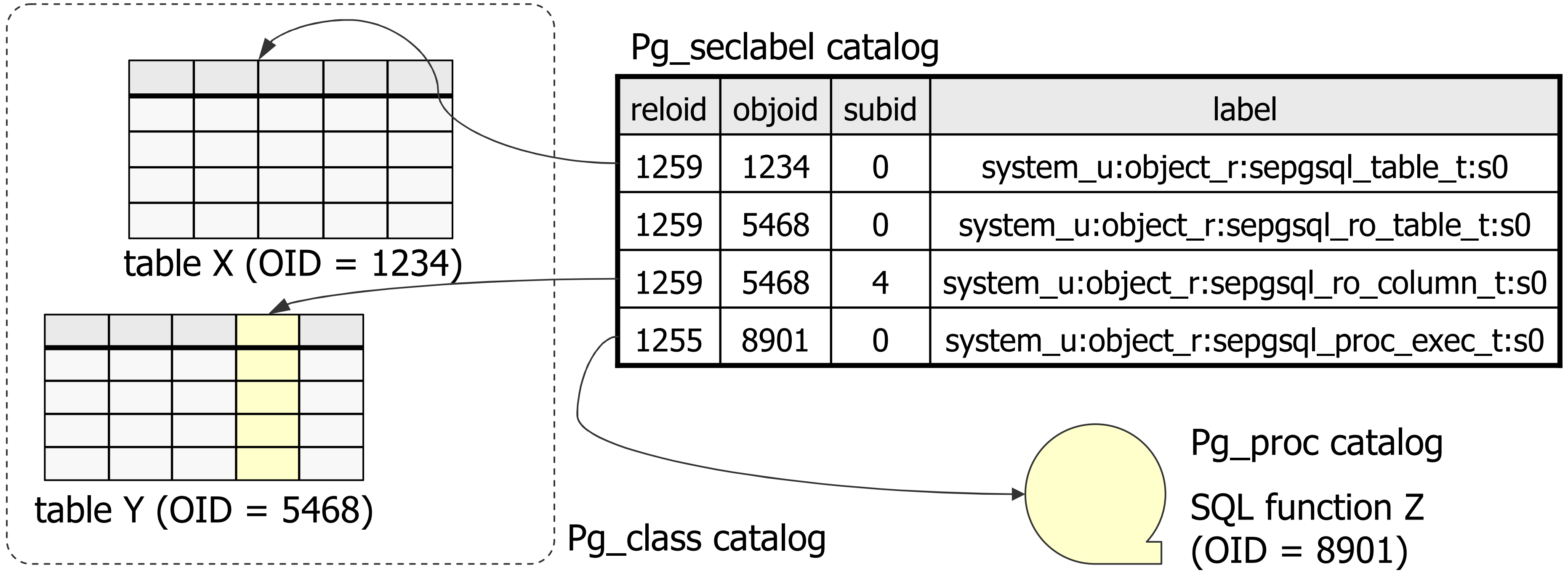
Security Hooks (2/2)



- SELinux provides labeled IPsec and **`getpeercon(3)`**
  - IKE daemon delivers security context of the user process
  - **`getpeercon(3)`** allows to retrieve the delivered security context
  - ✓ requires: kernel >= 2.6.18, ipsec-tools >= 0.7.2
- (Post) Authentication hook
  - It allows ESP plugins to get control post database authentication.
  - SE-PgSQL retrieve security context of the peer process, as privileges of the client.

Pg_seclabel system catalog

```
postgres=# SELECT * FROM pg_catalog.pg_seclabel;
 reloid | objoid | subid |   tag   |                 label
--------+--------+-------+---------+--------------------------------------
   1259 |   2619 |     0 | selinux | system_u:object_r:sepgsql_sysobj_t:s0
   1259 |   2619 |    -7 | selinux | system_u:object_r:sepgsql_sysobj_t:s0
   1259 |   2619 |    -6 | selinux | system_u:object_r:sepgsql_sysobj_t:s0
   1259 |   2619 |    -5 | selinux | system_u:object_r:sepgsql_sysobj_t:s0
     :        :        :       :                     :
```



Pg_seclabel catalog

reloid	objoid	subid	label
1259	1234	0	system_u:object_r:sepgsql_table_t:s0
1259	5468	0	system_u:object_r:sepgsql_ro_table_t:s0
1259	5468	4	system_u:object_r:sepgsql_ro_column_t:s0
1255	8901	0	system_u:object_r:sepgsql_proc_exec_t:s0

SECURITY LABEL statement

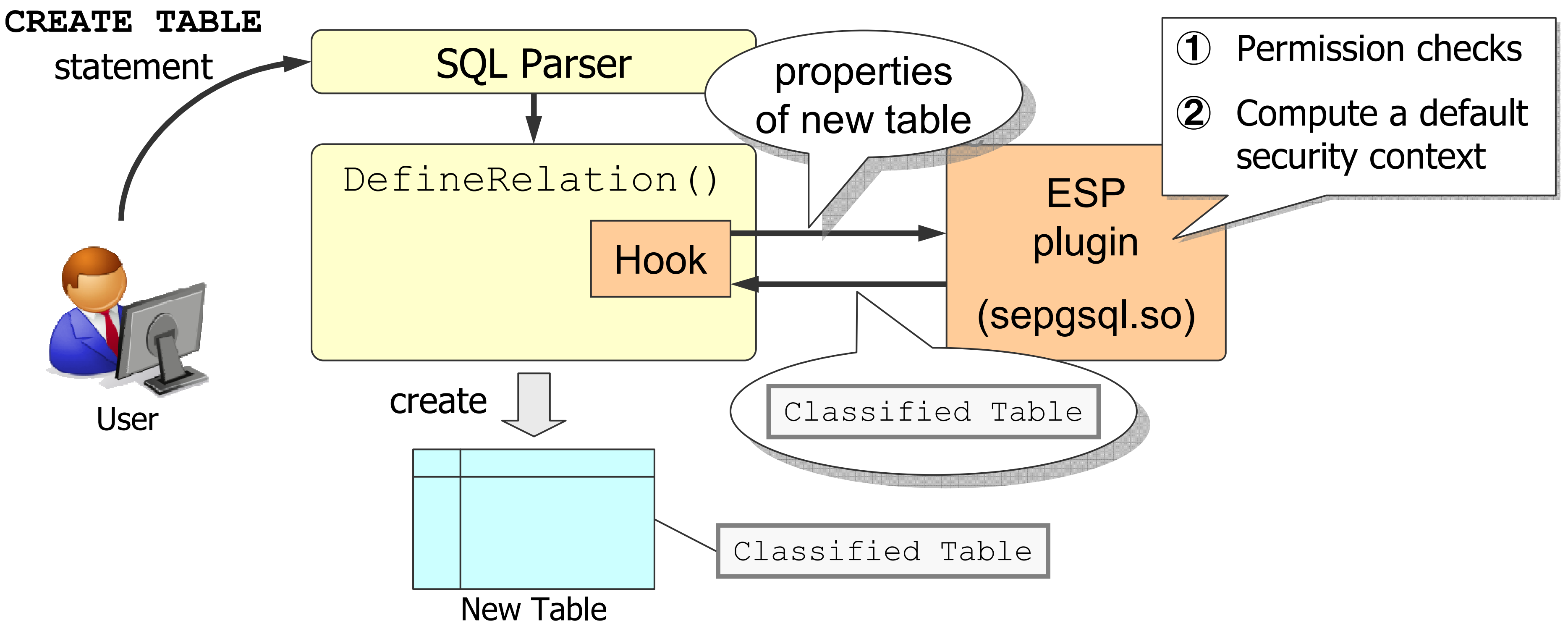
```
SECURITY LABEL [ FOR <provider> ]
      ON <objtype> <objname> IS <security label>
```

- This new SQL syntax provides an interface to change security label of database objects.
- ESP can validate the supplied label and check user's privileges.

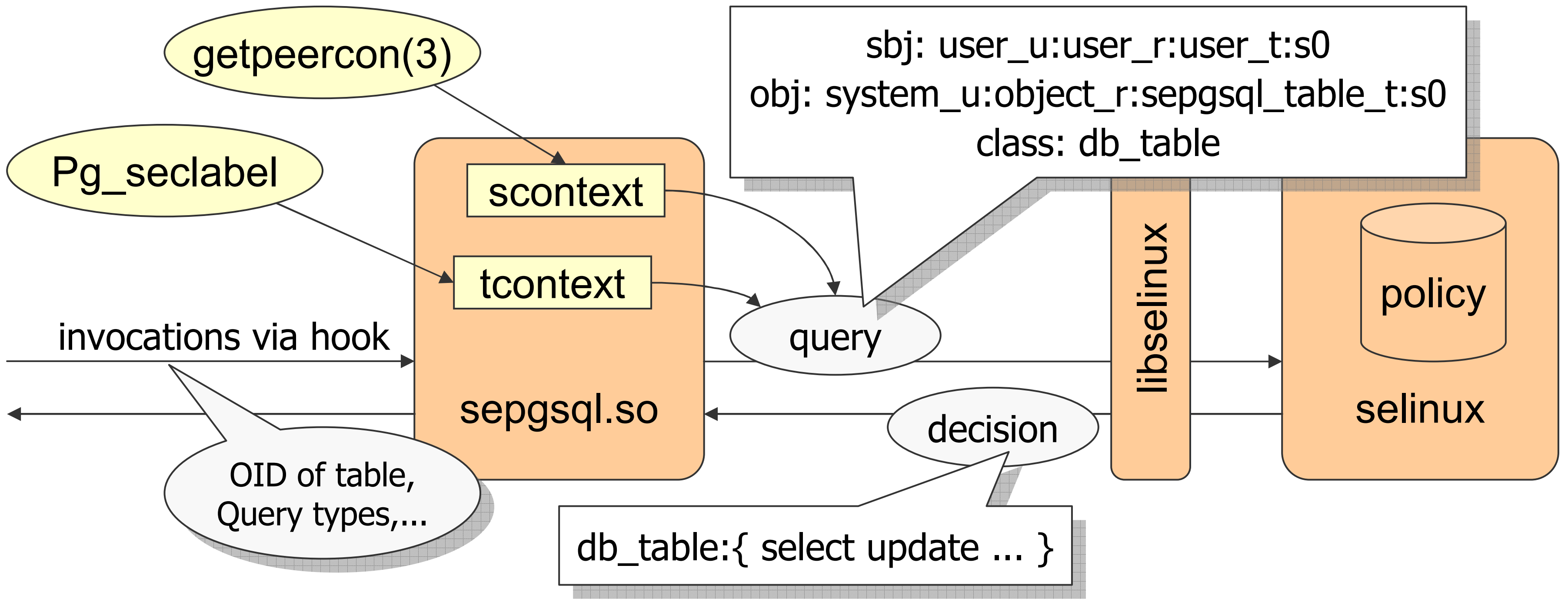
```
postgres=# SECURITY LABEL ON TABLE t1 IS
                'system_u:object_r:sepgsql_ro_table_t:s0';
LOG:  SELinux: allowed { setattr relabelfrom }
    scontext=unconfined_u:unconfined_r:unconfined_t:s0
    tcontext=system_u:object_r:sepgsql_table_t:s0
    tclass=db_table name=t1
LOG:  SELinux: allowed { relabelto }
    scontext=unconfined_u:unconfined_r:unconfined_t:s0
    tcontext=system_u:object_r:sepgsql_ro_table_t:s0
    tclass=db_table name=t1
SECURITY LABEL
```

OT: Default security context on table creation

- **`DefineRelation()`** also calls ESP plugin ...
  1. to check permission of table creation
  2. to get security context to be assigned on the new table
- A table has its security context on its creation time, then user can relabel it using **`SECURITY LABEL`** statement.



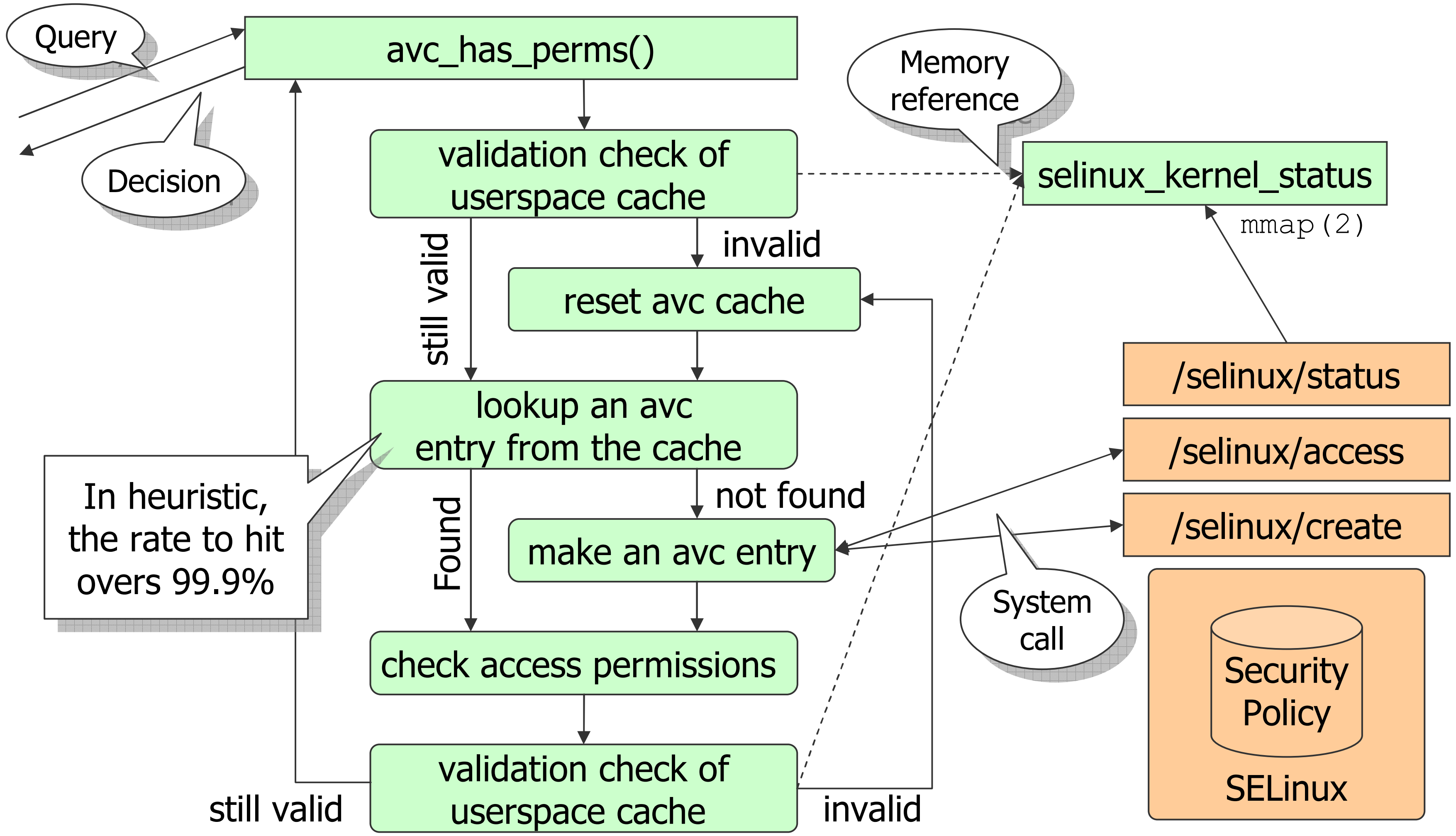
As an intermediator between PgSQL and SELinux



- sepgsql.so is the ESP plugin of SE-PostgreSQL
- It interprets a term of PgSQL into a term of SELinux
  - OID of the table ➔ security context of the table
  - ACL_SELECT ➔ db_table:{select} permission
- Then, it interprets SELinux's decision into status of PgSQL.
  - access denied ➔ ereport(ERROR, ...)

OT: Userspace access vector cache (avc)

- **`security_compute_xxx()`** always invokes a system-call
  - AVC enables to cache access control decisions recently used.



Empowered by Innovation NEC

3. Playing with SE-PostgreSQL
(demonstration)

Empowered by Innovation NEC

4. Today and the Future

Current status of SE-PostgreSQL

- Under development based on the v9.1
- Works in completion
  - Security hook on DML permission checks
- Works in progress
  - Pg_seclabel and security label support
  - Security hook on authentication
  - Security hook on table creation

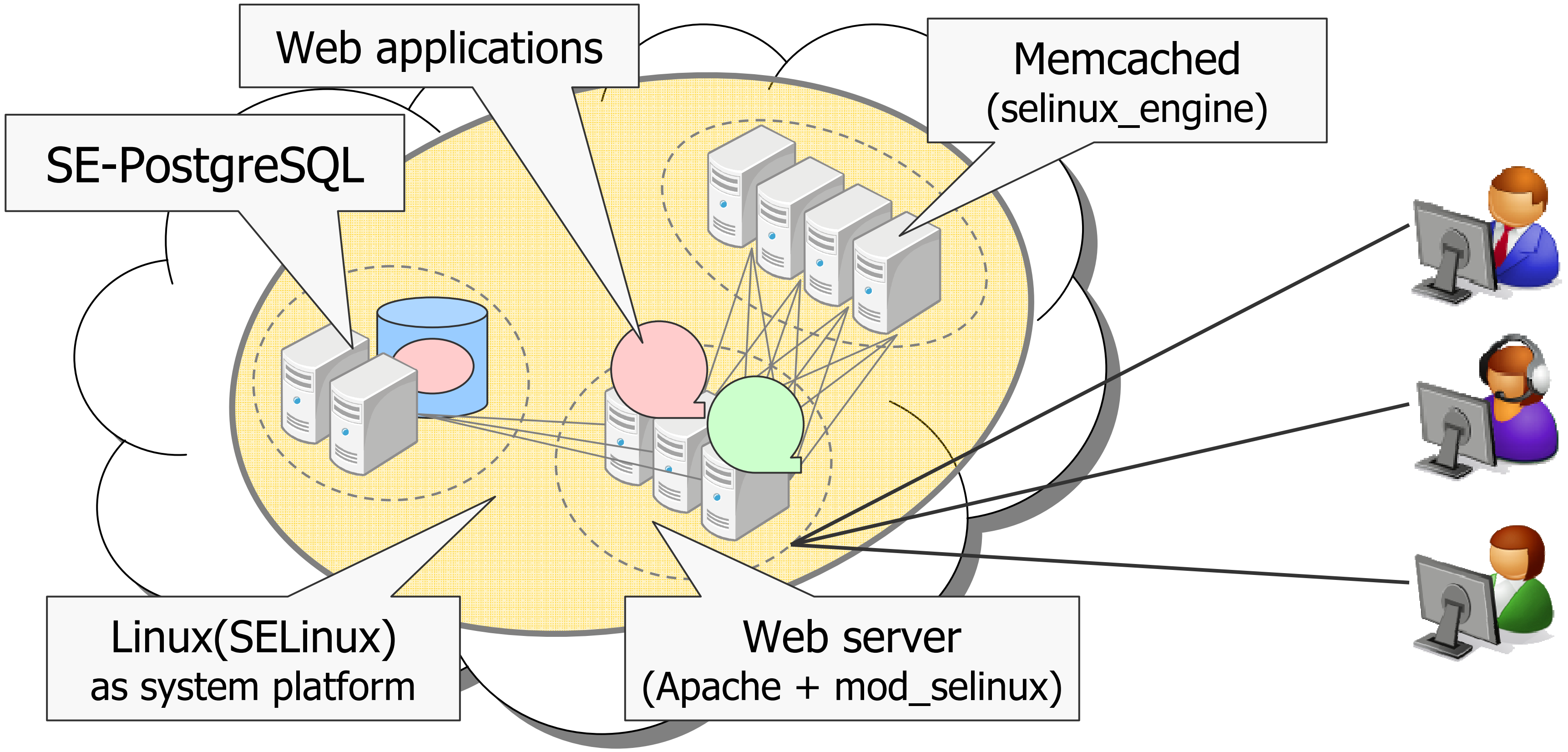
  → we have discussion on the CommitFest-2nd
- Source of the SE-PgSQL plugin

  http://code.google.com/p/sepgsql/

Future works of SE-PostgreSQL

- Comprehensive security hooks
- Backup/Restore support
- Trusted Procedure
- Security label of user tuples
- Row-level access control
- Integration with system audit

Our Information Assets over the Cloud



- Information assets getting consolidated at somewhere in the cloud
  - We can reference them anywhere, anytime, and anybody?
- Need to ensure both of data sharing and separation at the same time.

➡ System-wide consistency of access control on such a complex system

Empowered by Innovation NEC

Any Questions?

Empowered by Innovation NEC

Thank you!