Empowered by Innovation NEC

The Security-Enhanced PostgreSQL

- "system wide" consistency in access controls -

NEC OSS Promotion Center

KaiGai Kohei

<kaigai@ak.jp.nec.com>

Self Introduction

- Working for NEC, come from Tokyo, Japan
- Primary developer of SE-PostgreSQL
  - I've forcused on the work for more than 2 years.
- 6 year's experience in Linux kernel development
  - Especially, SELinux and security related region
    - SMP Scalability improvement (2.6.11)
    - XATTR Support in JFFS (2.6.18)
    - SELinux support in busybox
    - Type boundary and Multithreading (2.6.28)

The Background

Price of Notebook : $10.00
Price of Individual Info: priceless

- What should it be protected from harms by security?
  - Personal Info, Corporate secrets, Authentication data, ...
  - They are called as **Information-Assets**.
- Information asset is not tangible.
  - It always has to be stored in something.
  - Filesystem, Database, IC Card, Paper, Brain, Lithograph, ...

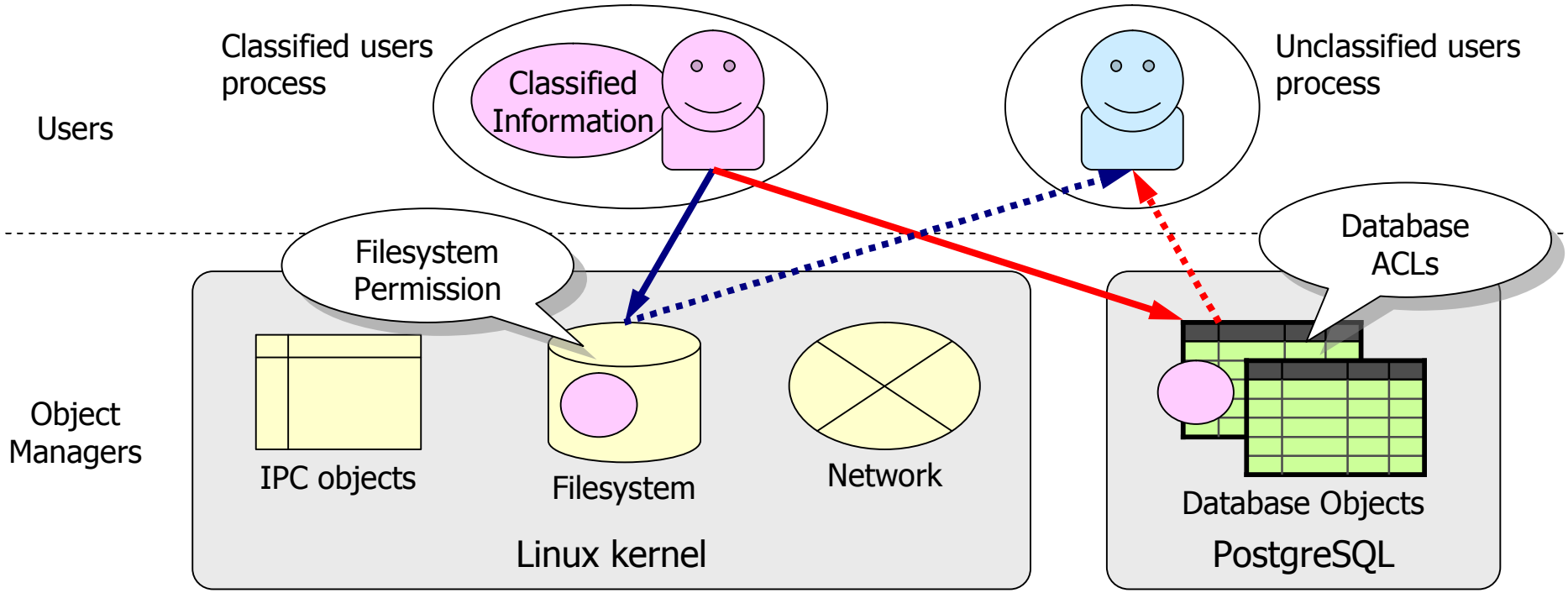
I dislike a term of "Database Security"

- What determines the value of information asset?
  - Contents, not the way to store them
- How access control mechanism works?
  - Filesystem　　Filesystem permission (combination of r,w,x)
  - Database　　Database ACLs (GRANT and REVOKE)
  - It completely depends on the way to store them!

We should apply consistent access control rules for same information assets, independent from the way to store them!

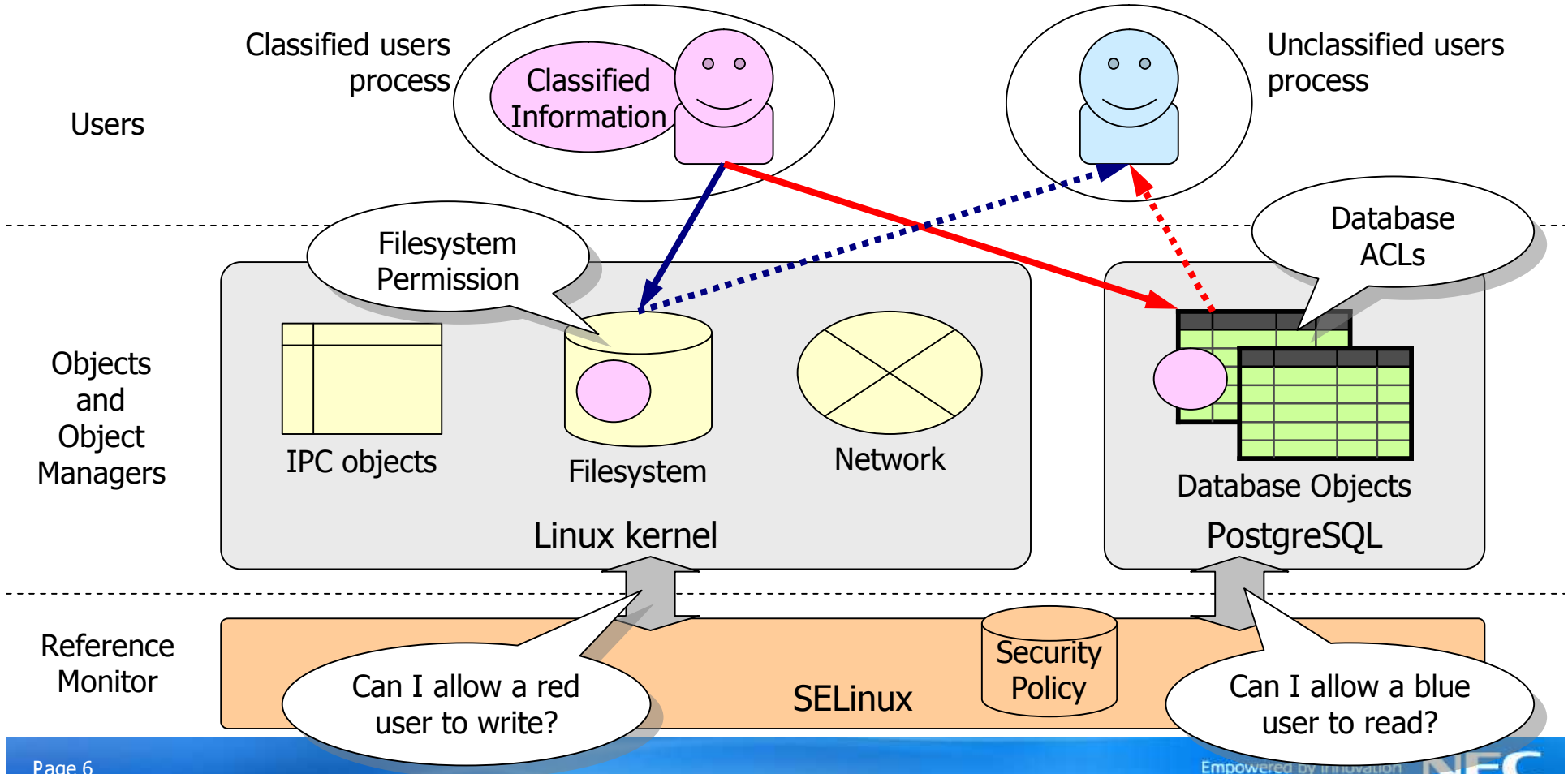
Consistency in Access controls (1/2)

- Access control policy depends on the way to store information asset.
- They have a possibility to apply inconsistent access control policy.



Consistency in Access controls (2/2)

- Object manager queries SELinux about required actions.
- SELinux makes its consistent decision based on a unique policy.



The Feature of SE-PostgreSQL

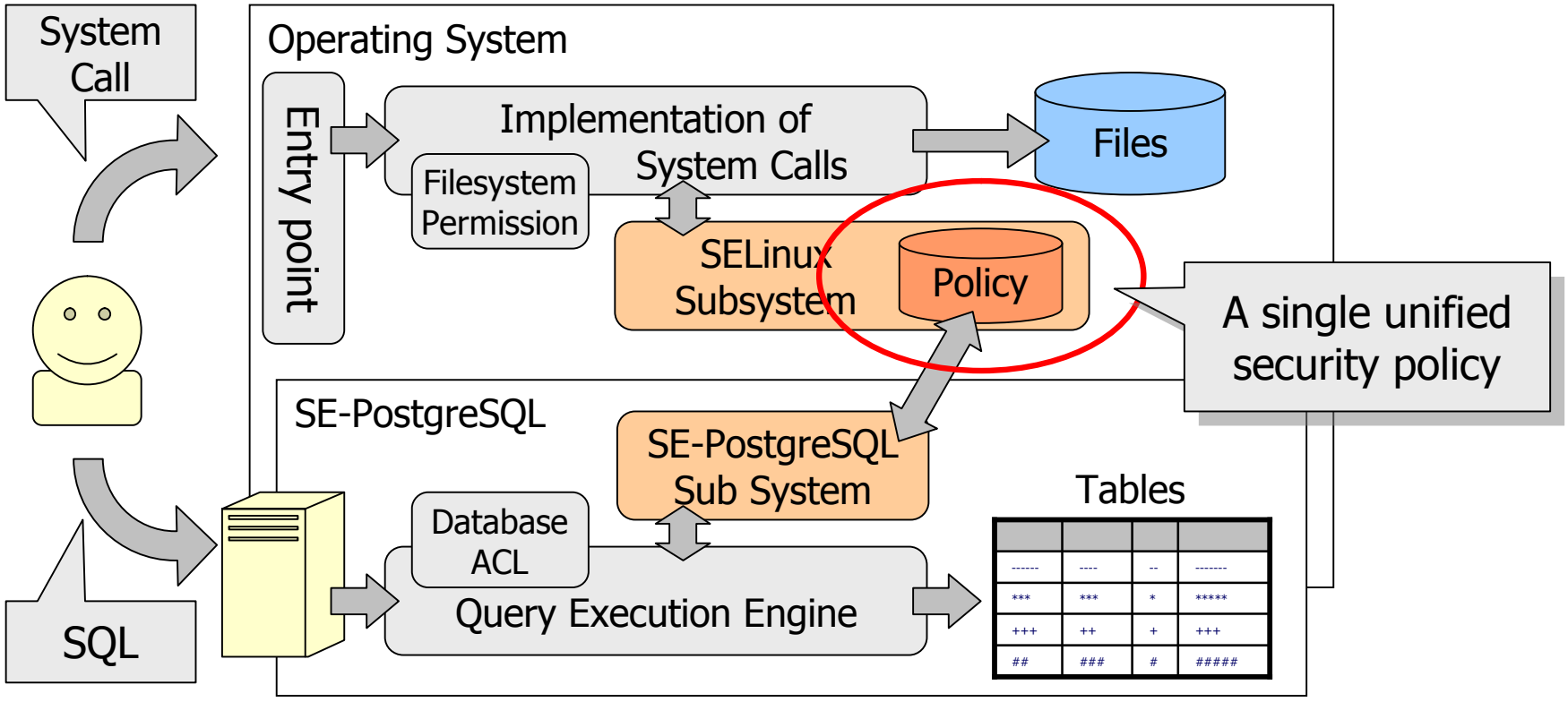
- "System-wide" consistency in access controls
  - A single unified security policy between OS and DBMS
  - Common representation in security attributes
- Fine-grained Mandatory access controls
  - Including column-/row-level access controls
  - Non-bypassable, even if database super users
- The Goal of SE-PostgreSQL
  - DBMS as a part of Data Flow Control schemes
  - Prevention for leaking/manipulation by malicious ones
  - Minimization of damages via SQL injections

Empowered by Innovation NEC

"System-wide" consistency in Access Controls

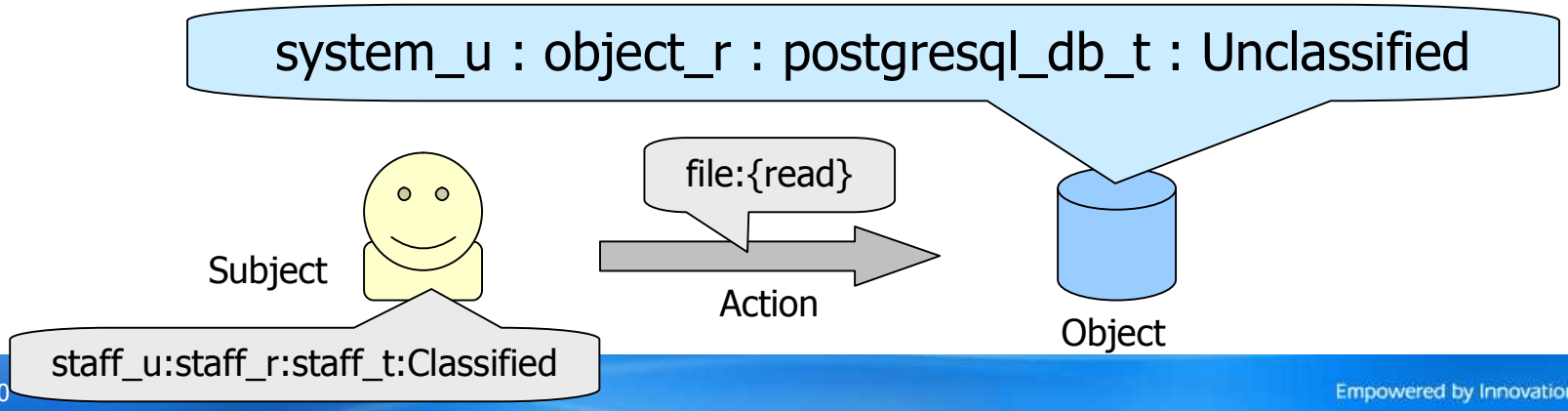
SE-PostgreSQL system design

- A single unified security policy is applied,
  - when user tries to read a file via system-calls
  - when user tries to select a table via SQL-queries



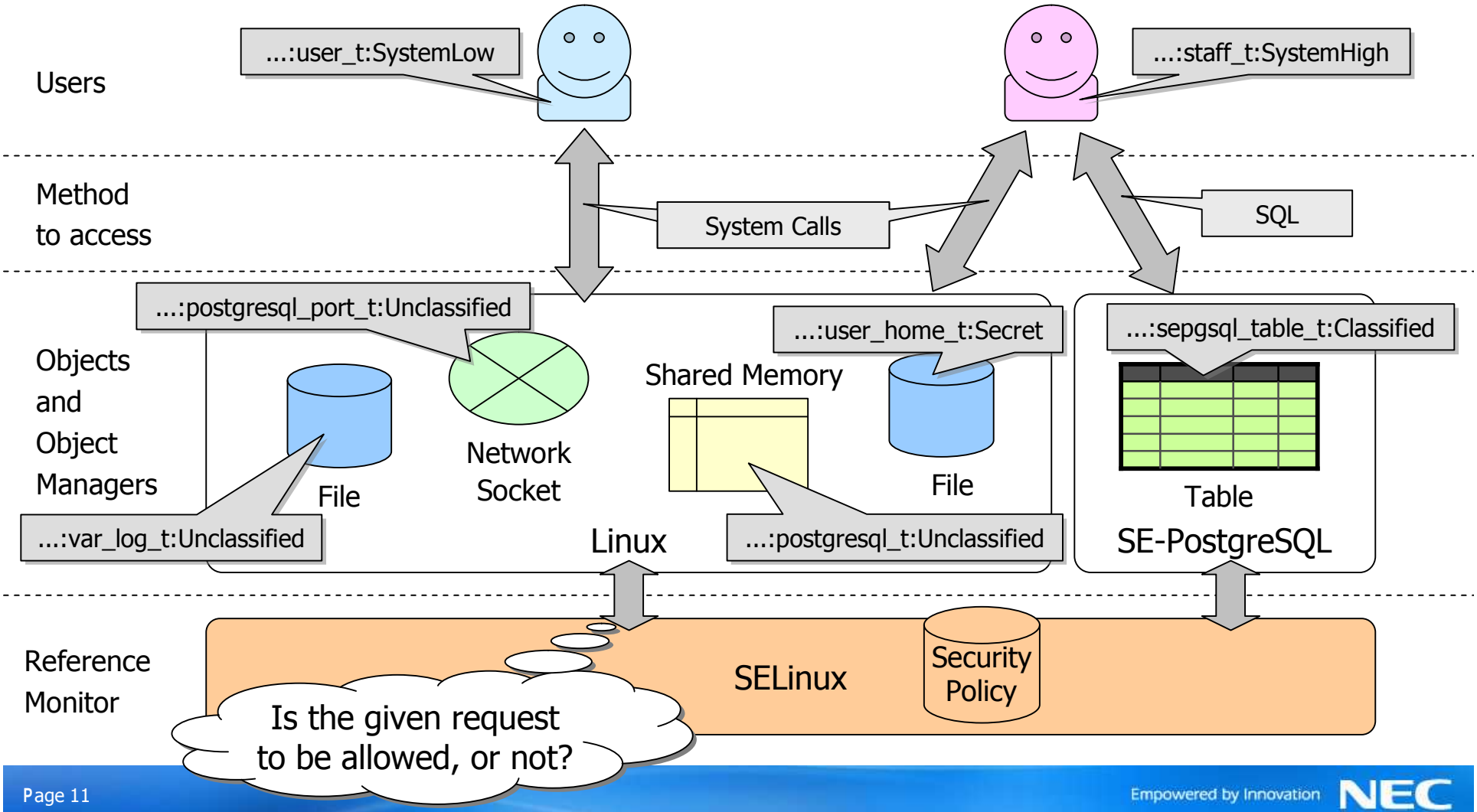
How security policy works? (1/2)

- What is the security policy?
  - A set of rules, managed by SELinux
  - Individual rule describes who is allowed to do what operations for what objects.
  - Any entities are identified by **security context**.
- What is the security context?
  - A formatted text for security attribute
  - Common representation for various kind of objects



How security policy works? (2/2)

- All the objects have its security context managed by Object managers.
- SELinux makes its decision, and Object managers follows it.



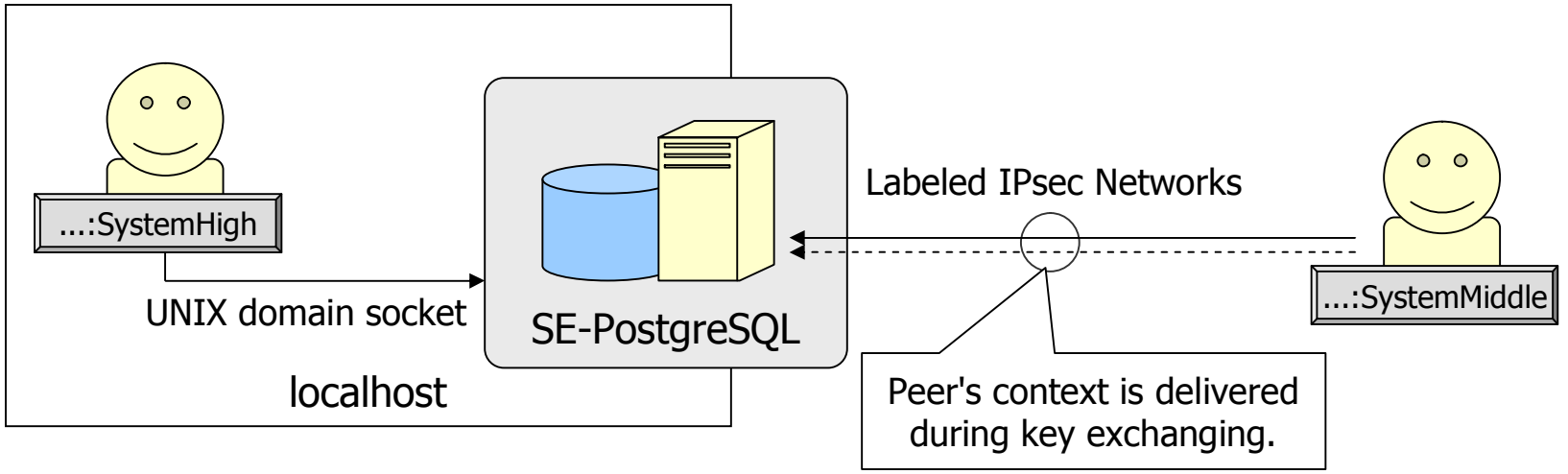
"security_label" system column

```
postgres=# SELECT security_label, * FROM drink;
                 security_label                  | id |  name   | price
-------------------------------------------------+----+---------+-------
 system_u:object_r:sepgsql_ro_table_t            |  1 | water   |   100
 system_u:object_r:sepgsql_ro_table_t            |  2 | coke    |   120
 system_u:object_r:sepgsql_table_t               |  3 | juice   |   130
 system_u:object_r:sepgsql_table_t               |  4 | coffee  |   180
 system_u:object_r:sepgsql_table_t:Classified    |  5 | beer    |   240
 system_u:object_r:sepgsql_table_t:Classified    |  6 | sake    |   320
(6 rows)
```

- It enables to export/import security context of tuples.
  - Note: PostgreSQL has special relations called as system catalog.
    Security context of tuples within them shows ones of tables, columns, ...
- The "security_label" system column is writable.
- A default security context is assigned for newly inserted tuple.

Privileges of Clients

- Access controls, as if user reads files via system-calls
  - But, queries come through networks
- Labeled Networking Technology
  - **getpeercon()** API in SELinux
  - It enables to obtain security context of the peer process.
  - SE-PostgreSQL applies it as privileges of client.



Empowered by Innovation NEC

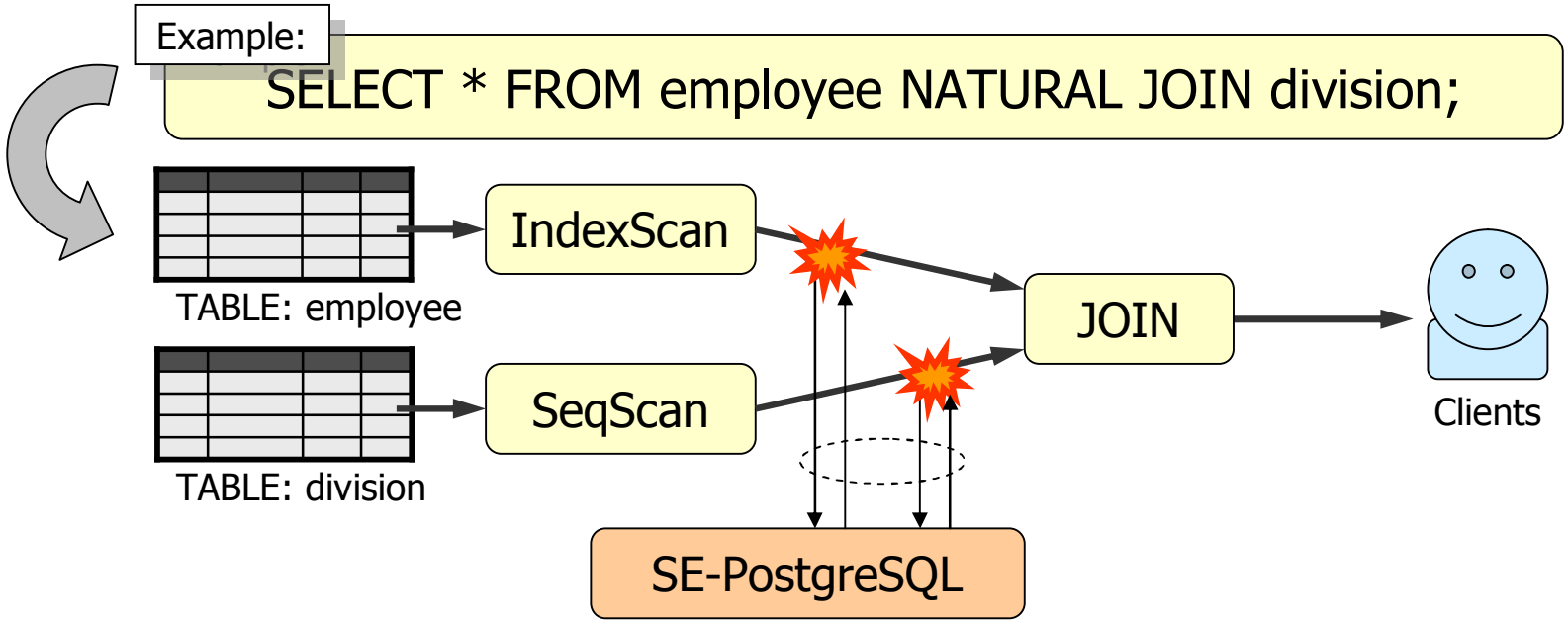
Fine-Grained Mandatory Access Controls

Mandatory Access Controls

- PostgreSQL has **superuser**
  - It is allowed to bypass all the access controls
  - Like a **root** in operating system, nightmare for security
- Resource owner can change its access rights
  - Possibly, he can leak classified information assets.
- How does SE-PostgreSQL handle them?
  - Is applies its security policy on all the queries, including ones come from superusers.
  - It does not allow to bypass its access controls.
  - Any DB objects are labeled based on security policy.

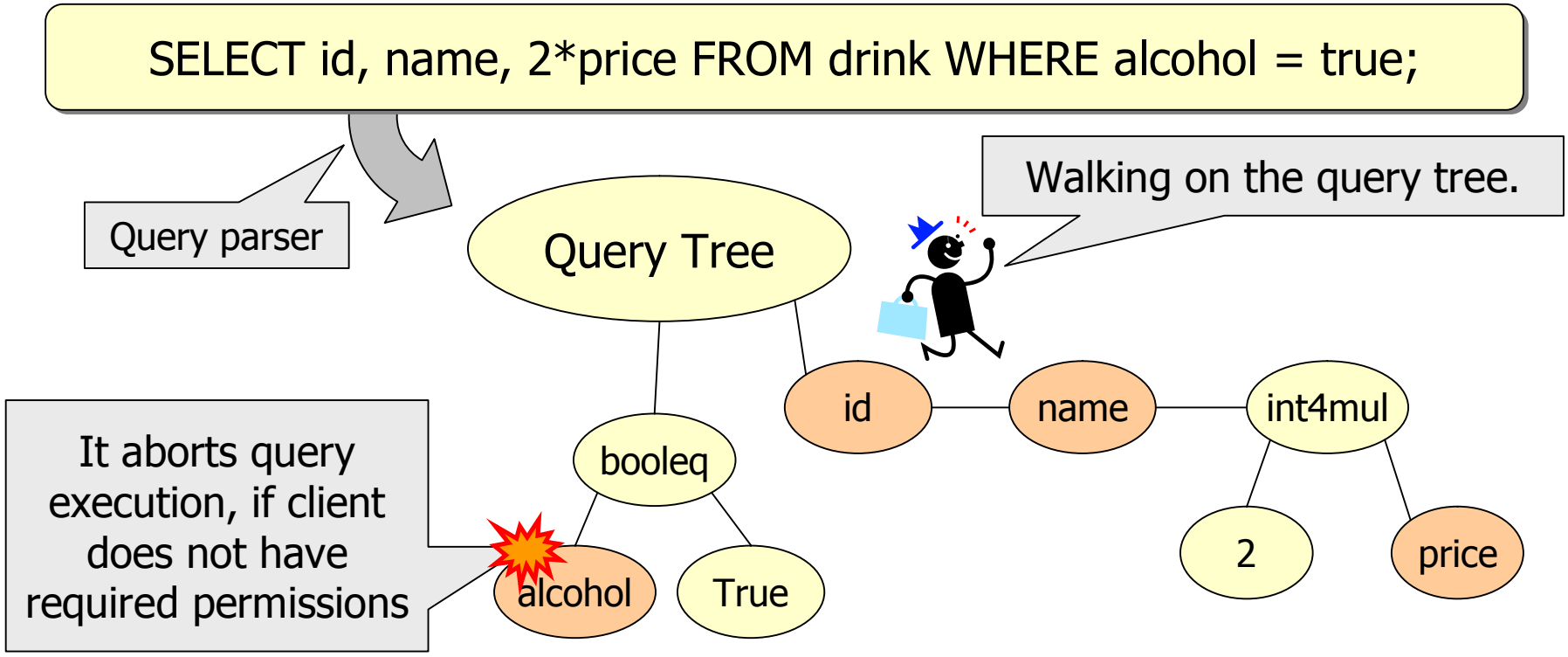
Row-level Access Controls

- SE-PostgreSQL filters any violated tuples from result set, as if they are not on scaned relations.
- It skips to modity violated ones on UPDATE/DELETE
- It checks a new tuple on INSERT.



Column-level Access Controls

- SE-PostgreSQL checks any columns appeared in the queries.
  - It aborts query execution, if violated one is found.
  - All checks are applied before query execution.



Case Study (1/2)

```
SELECT name, price * 2 FROM drink WHERE id < 40;
```

- db_column:{select} for **name** and **price** column
- db_column:{use} for **id** column
  - {use} permission means "referred but consumed internally"
- db_procedure:{execute} for **int4mul** and **int4lt** function
- db_table:{select use} for **drink** table
  - It raises an error, if privileges are not enough.

Implementation of operators.

And

- db_tuple:{select use} for each tuples
  - Any violated tuples are filtered from result set.

Case Study (2/2)

```
UPDATE drink SET size = 500, price = price * 2
                 WHERE alcohol = true;
```

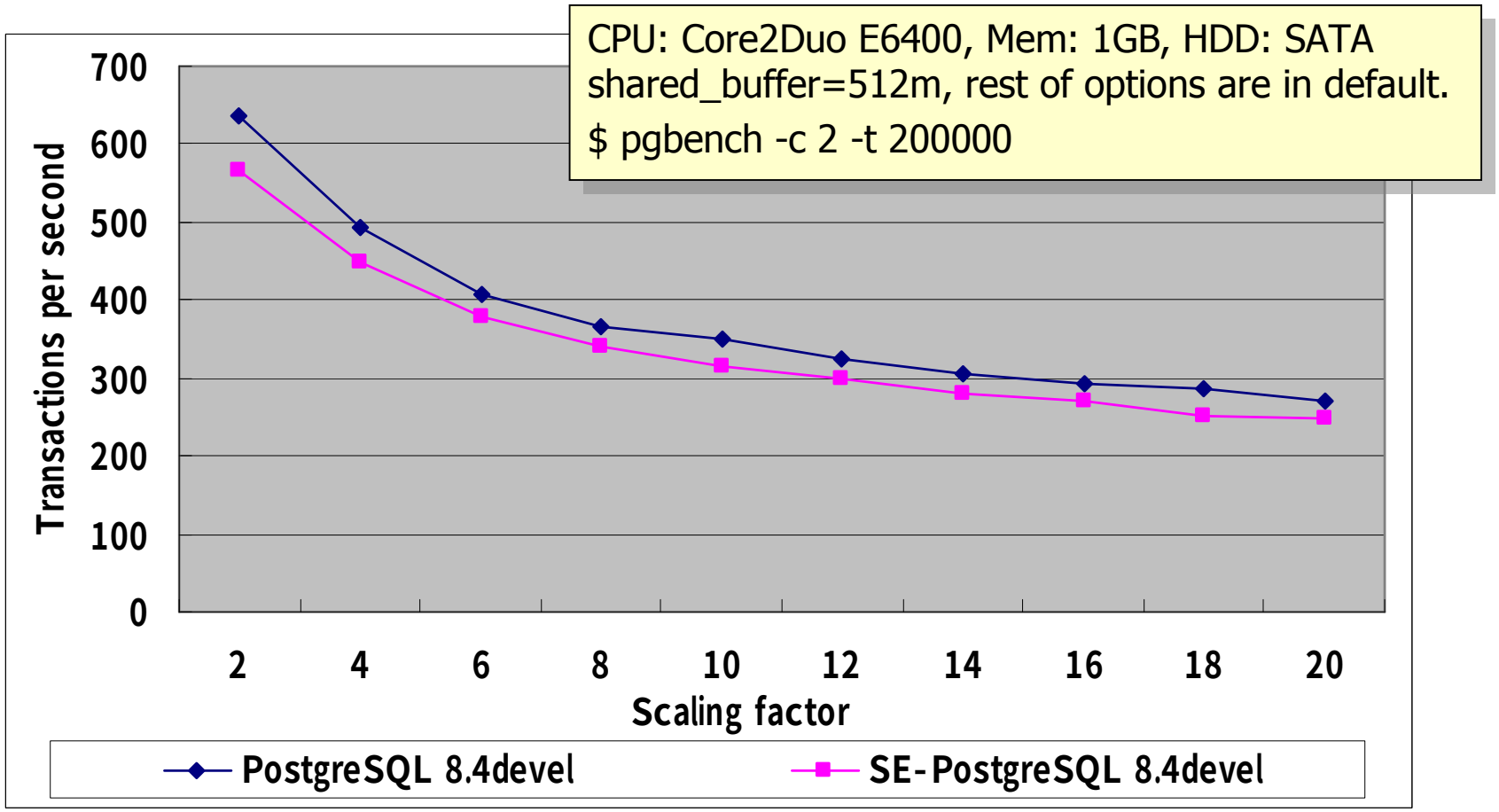
- db_column:{update} for **size** column
- db_column:{select update} for **price** column
  - price column is also read, not only updated.
- db_column:{use} for **alcohol** column
- db_procedure:{execute} for **booleq** and **int4mul** function
- db_table:{select use update} for **drink** table
  - It raises an error, if privilges are not enough.

And

- db_tuple:{select use update} for each tuples
  - Any violated tuples are excepted from the target of updating.

Demonstration

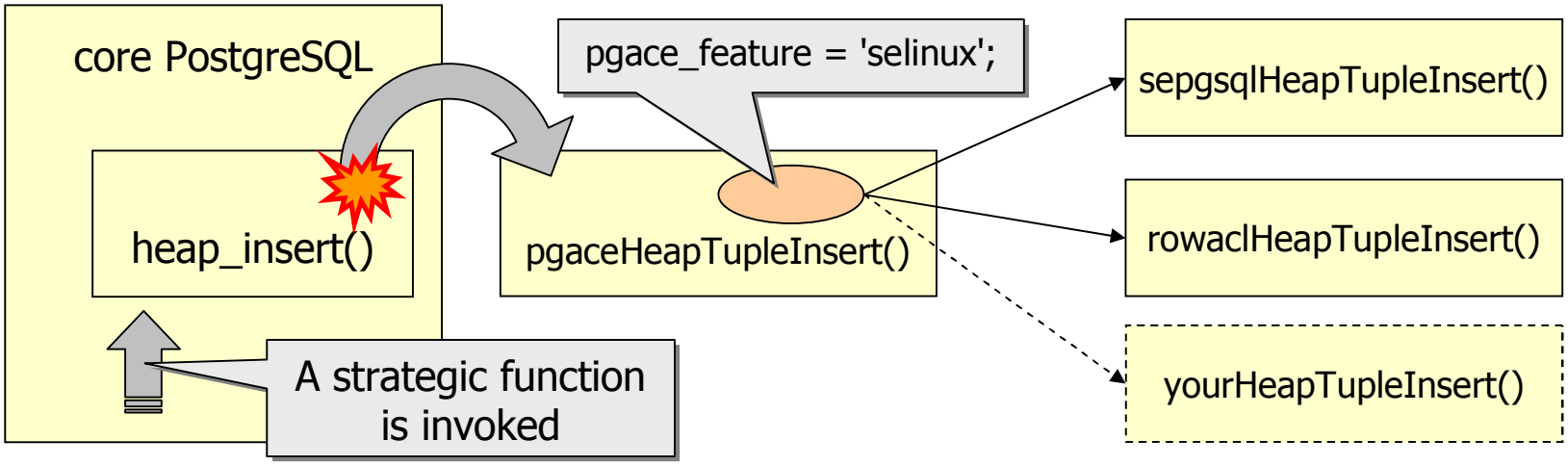
Performance



- about 10% security-tradeoff in maximum
- access vector cache (AVC) minimizes # of system-call invocation

PGACE Security Framework

- PGACE : PostgreSQL Access Control Extension
  - A common framework for various security designs
    - various security hooks in strategic points
    - facilities to manage security attribute of DB objects
    - Add enhanced security features with minimum impact
  - Available features
    - SE-PostgreSQL, Row-level ACLs, Trusted-Solaris (upcoming)



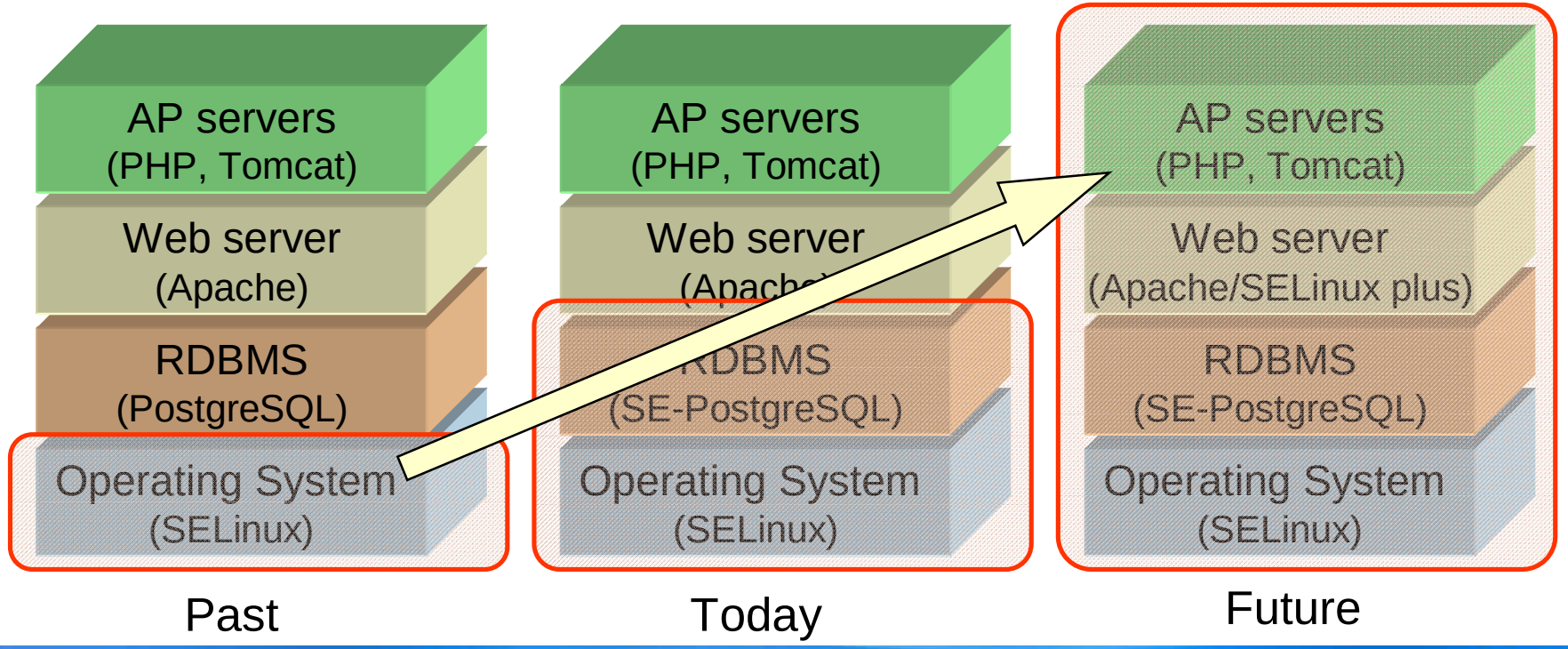
The current status of SE-PostgreSQL

- Upstreaming status
  - Currently, we are working under PostgreSQL v8.4 development cycle.

    http://wiki.postgresql.org/wiki/CommitFest:2008-11
  - It has been unclear whether we can enjoy SE-PostgreSQL on the next version of vanilla PostgreSQL, or not :(
- Distribution Support
  - **sepostgresql** package is available on Fedora 8 or later.
  - The default security policy also support SE-PostgreSQL.
- Resources
  - http://code.google.com/p/sepgsql/
  - http://wiki.postgresql.org/wiki/SEPostgreSQL

Future Visions

- SE-PostgreSQL as a foundation of secure web application stack.
- SELinux enables to controls whole of the LAPP stack.
- Security is a concept of whole of the system, not only individual conponents, so I dislike a term of "Database Security".



Any Question?

Empowered by Innovation NEC

Thank you!