Empowered by Innovation NEC

Getting Memcached Secure

NEC OSS Promotion Center

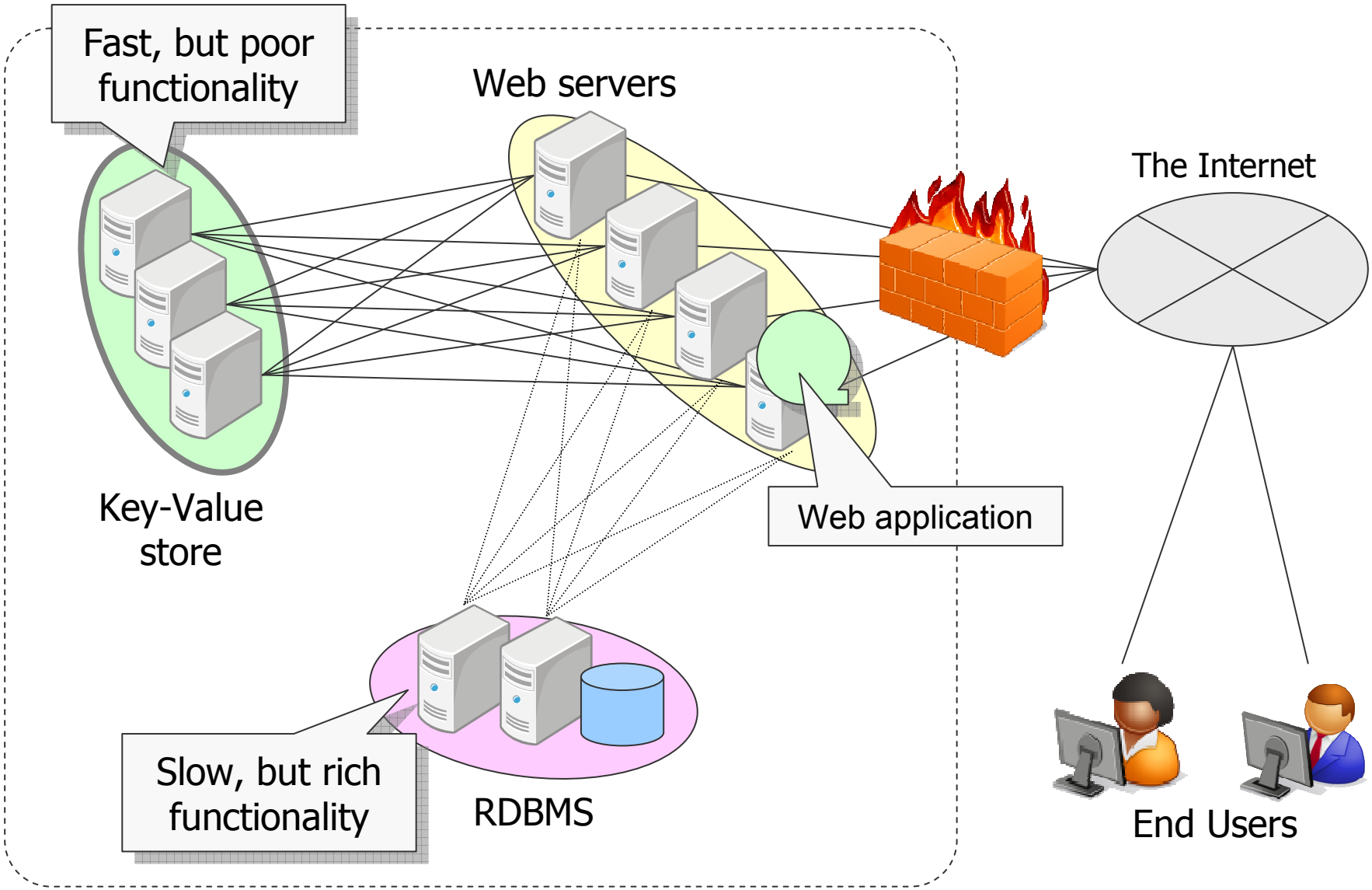
KaiGai Kohei <kaigai@ak.jp.nec.com>

Self Introduction

- Name KaiGai Kohei
- Company NEC, OSS Promotion Center
- Works 7 years experiences of OSS development
  - » SELinux
  - » PostgreSQL
  - » **Memcached**
  - » Apache (mod_selinux)
- Memcached - selinux engine
  - A memcached plugin to apply mandatory access control according to the SELinux policy.

1. Memcached and security
   - Background
   - Centralized security and SELinux
2. Getting Memcached secure
   - Adjustment of security model
   - Engine framework performing with libselinux
   - The selinux_engine.so plugin

Recent web-system's architecture



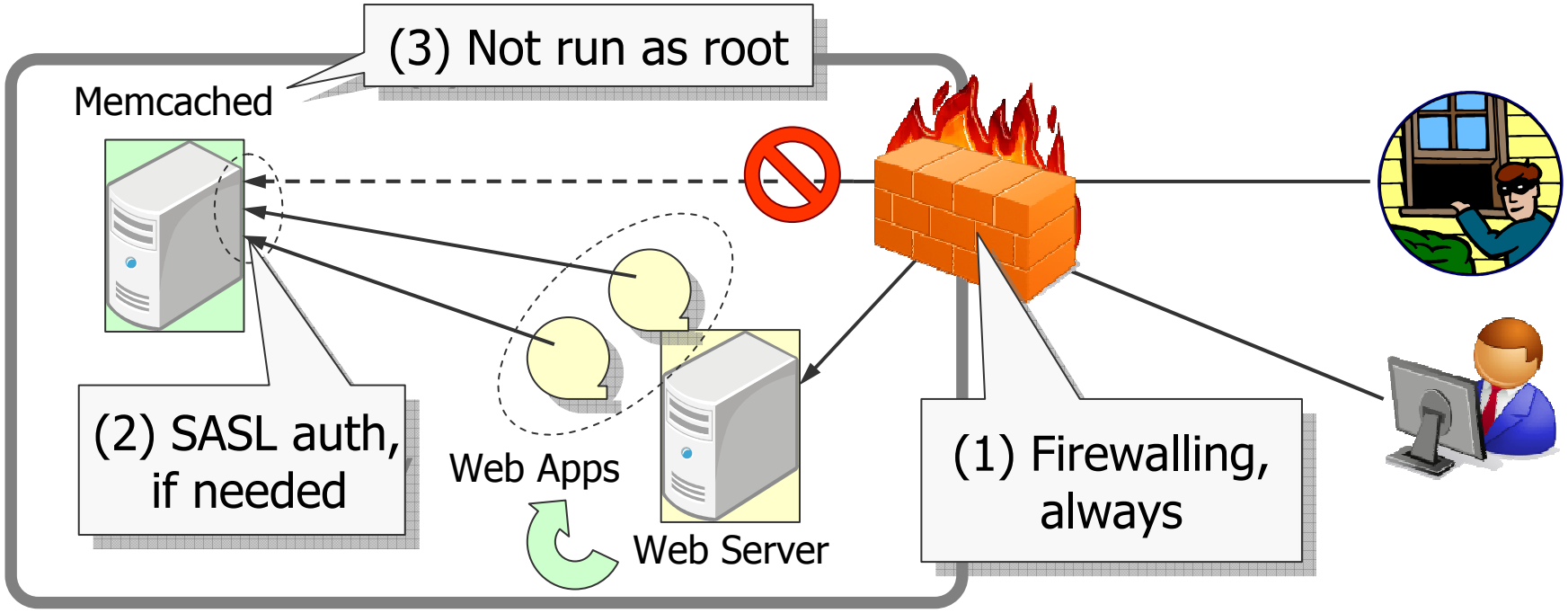
What is Memcached

Memcached

- general purpose, high-performance, distributed memory caches
- Typically, used to backends of high-traffic web systems
- Much faster than RDBMS, but less functionalities

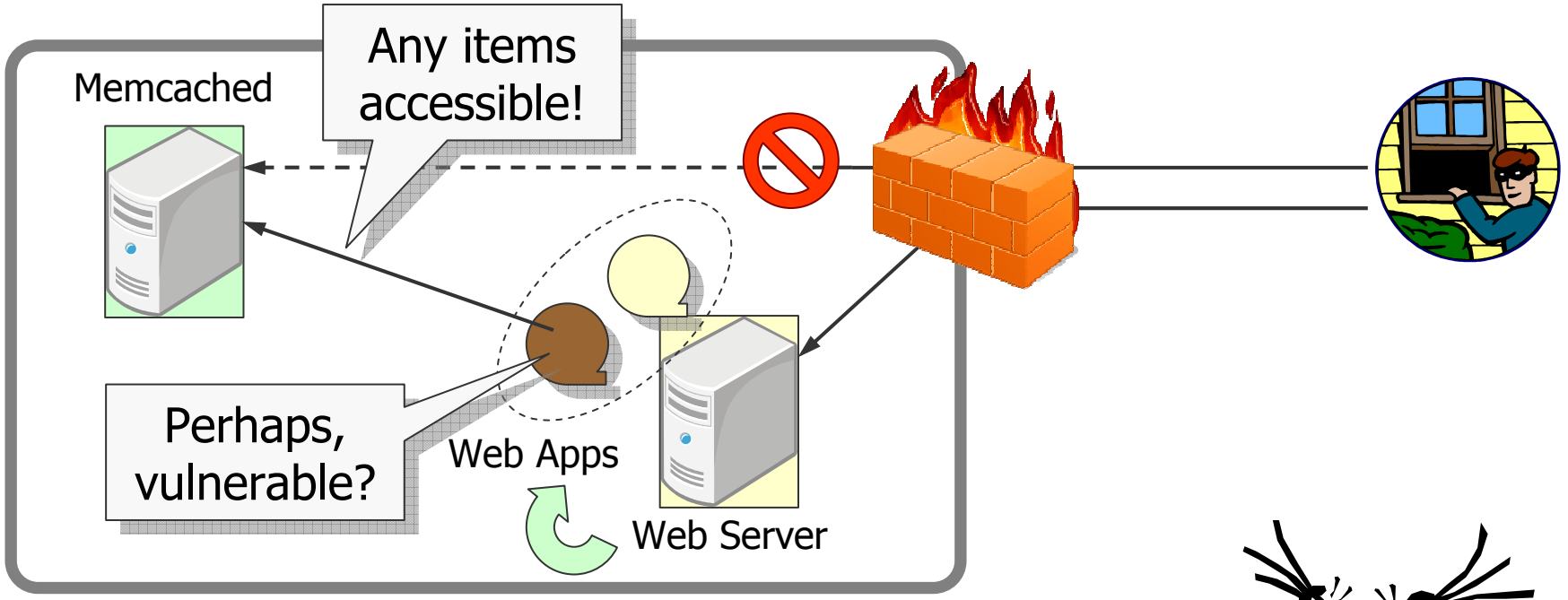
	PostgreSQL	Memcached
Client Interface	SQL	memcached protocol
Script support	OK	OK
Schemed Data	good	bad
Data Integrity	good	bad
Performance	relatively worse	good
Scaling-out	not easy	much easier
Security	authentication & access controls	less features

Memcached from security perspective (1/2)



- We have few options to keep Memcached secure :-(
  - Should never allow to connect from external network
  - SASL authentication
  - Should never run as root
- Memcached Security; by Dustin Sallings
  http://dustin.github.com/2010/08/08/memcached-security.html

Memcached from security perspective (2/2)

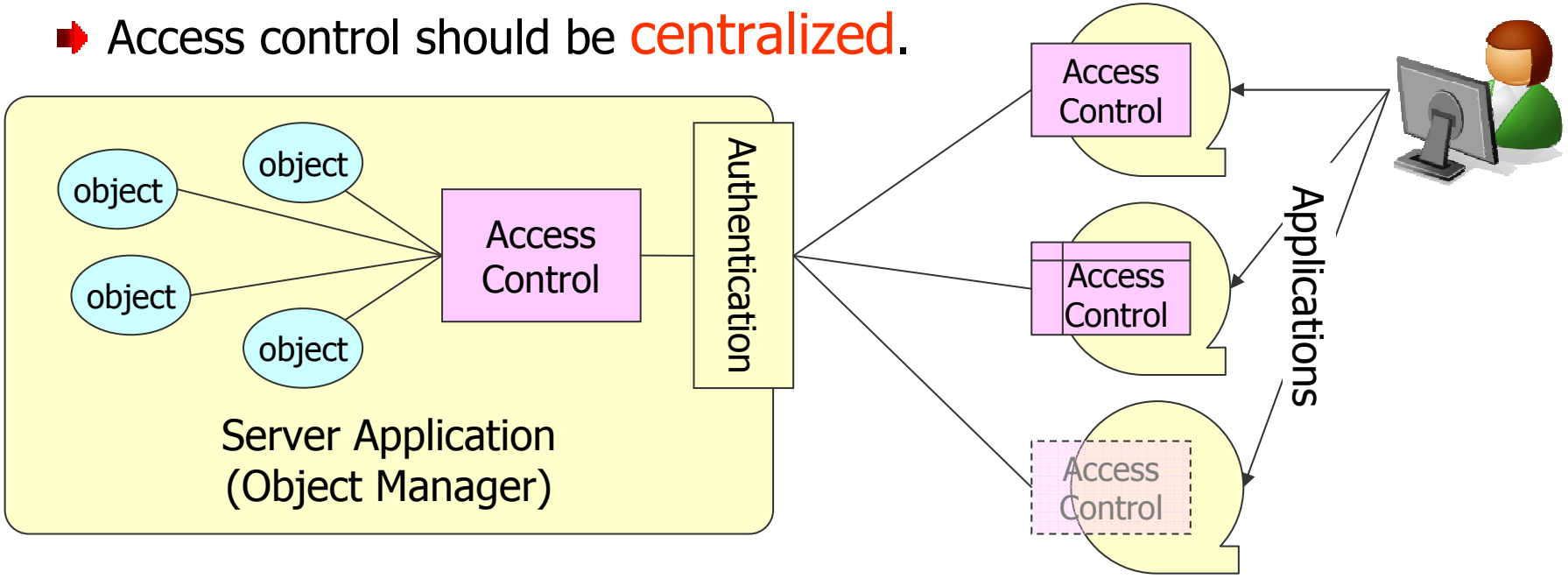


Our concern

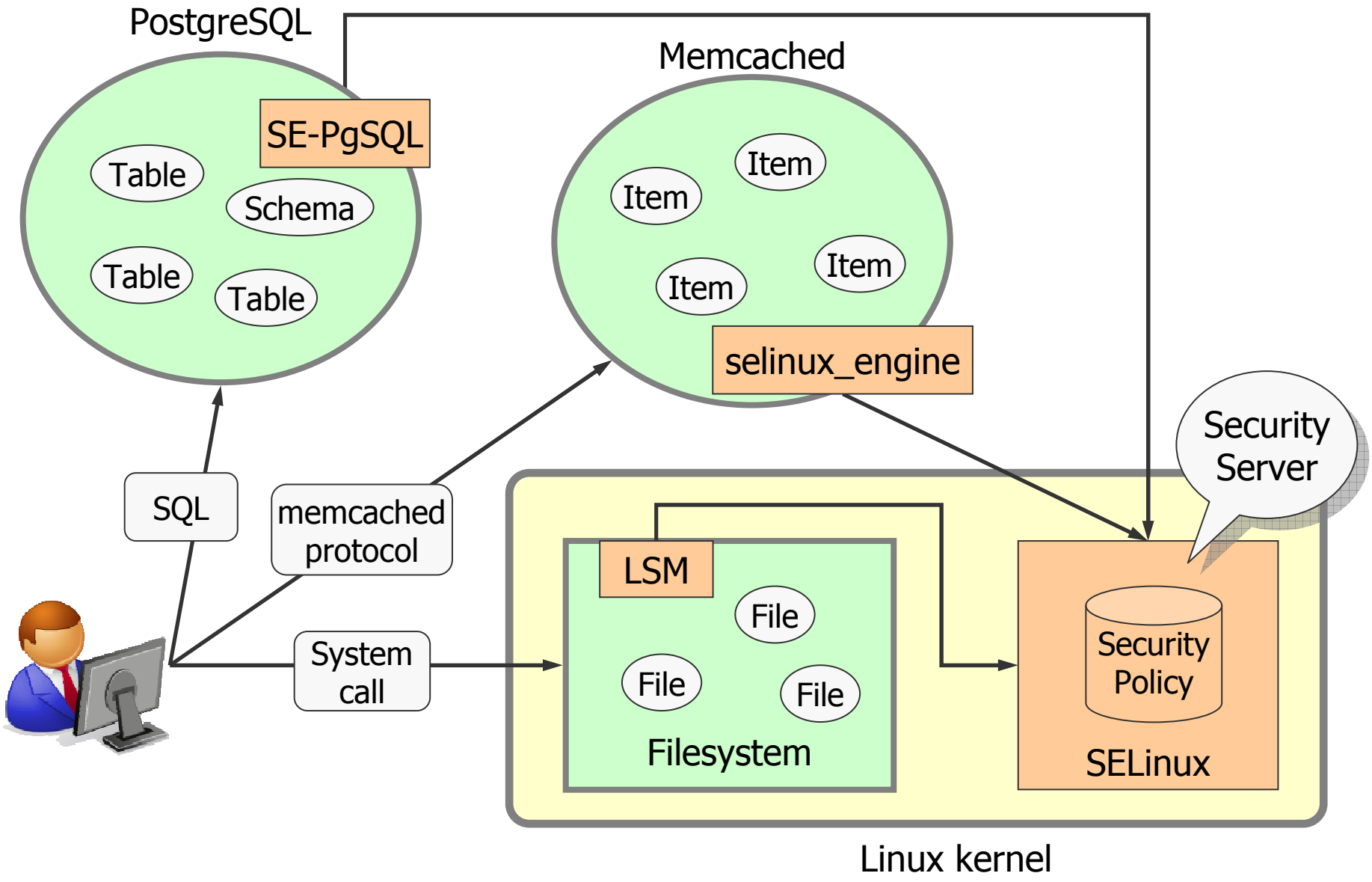
- No protection from internal threats
- Buggy application turns an external threats into an internal threat.
- It means all the application must be FREE from BUGS and VULNERABILITIES!

Why server software applies access controls

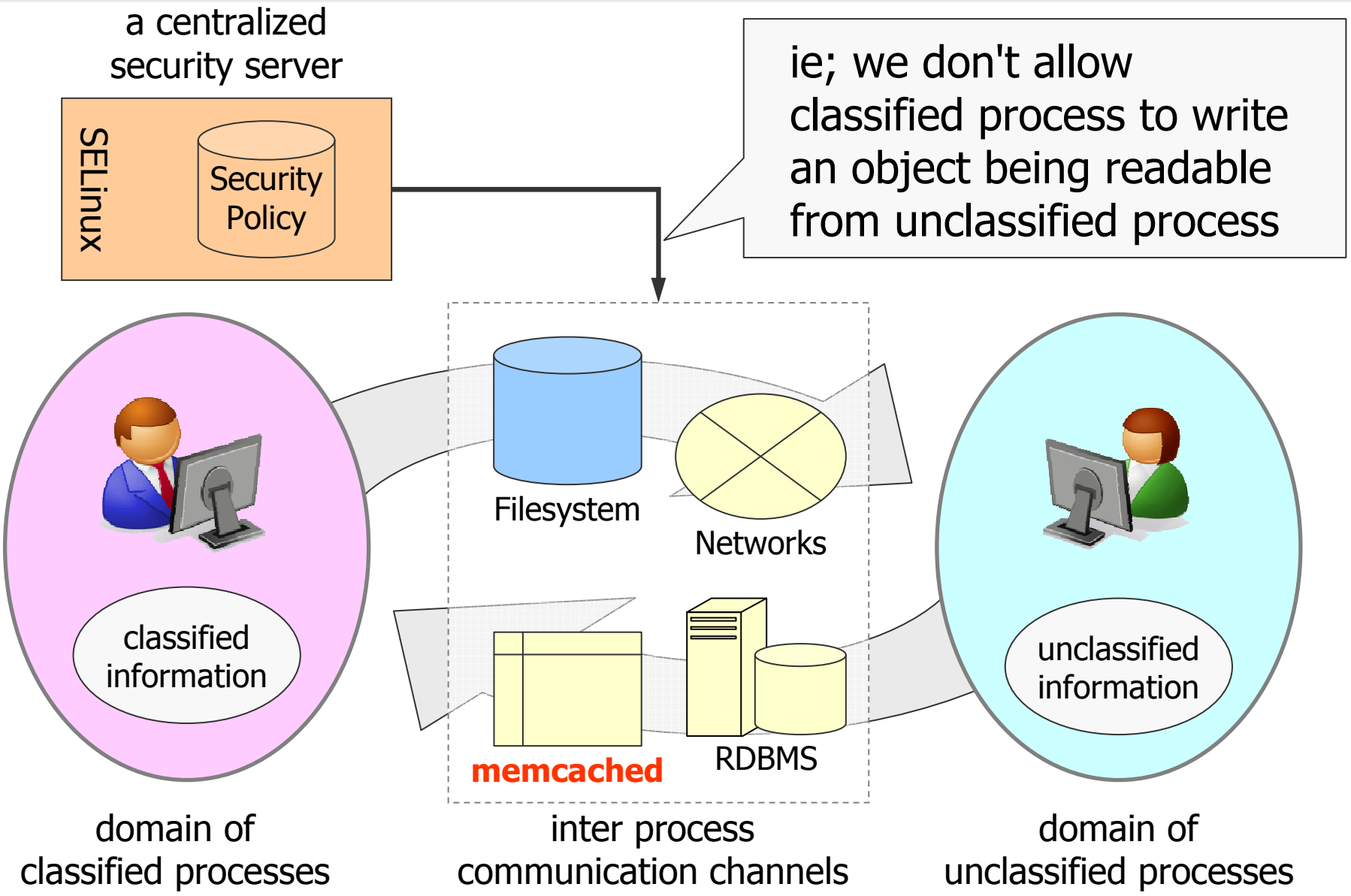
- How reliable is the security feature?
  - Consistency and Comprehensiveness
- Which is more preferable to apply access control?
  - If each applications apply access control?
    - ✓ Some of them may not be right
    - ✓ Some of them may check nothing...
  - Access control should be centralized.



More centralized access control (1/2)



More centralized access control (2/2)



SELinux as a Security Server (1/3)

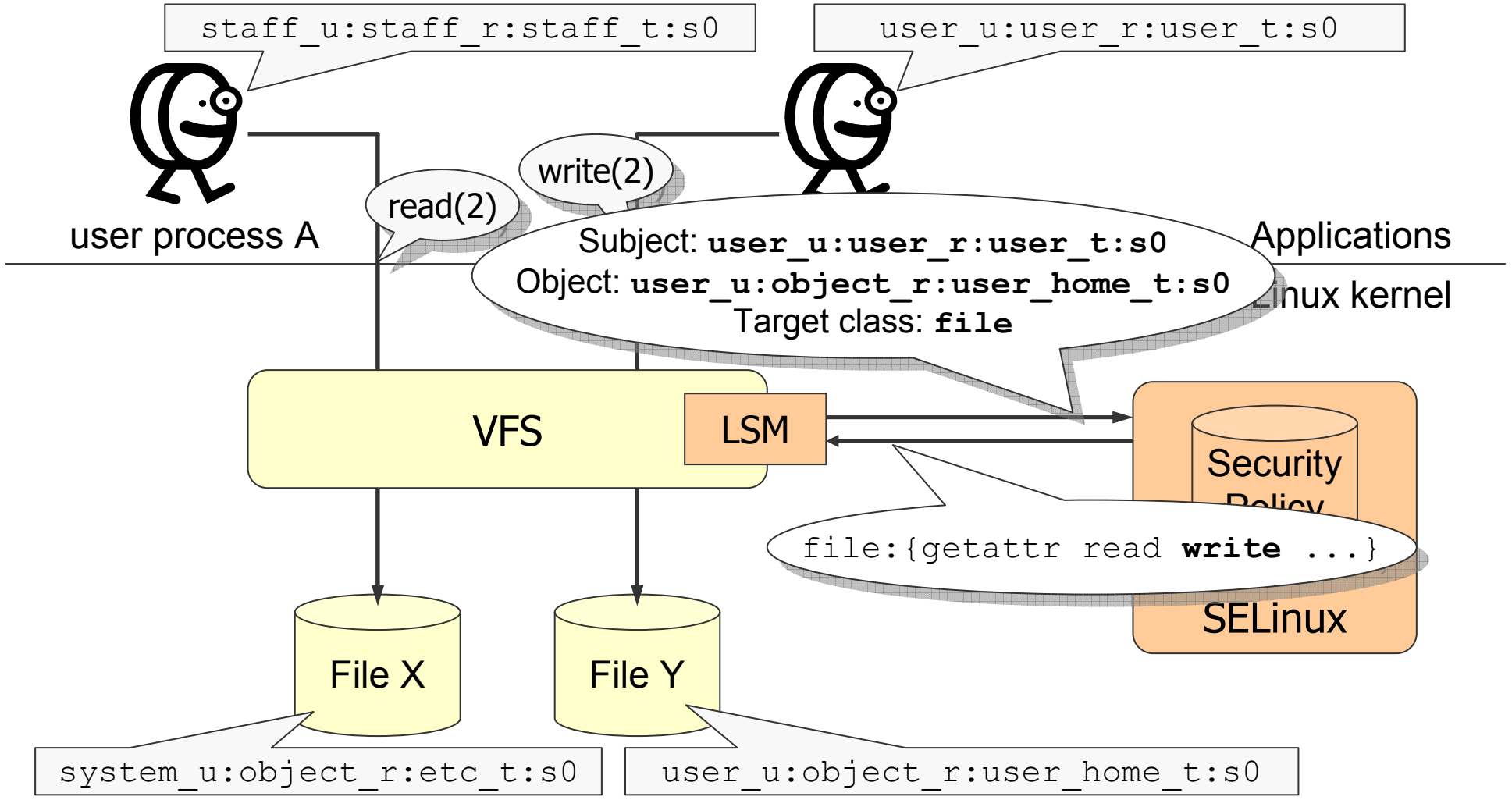
- Interactions with object managers
  - Kernel subsystems do queries via LSM.
  - Userspace applications do queries via libselinux.
  - ➡ Both of them control user's requests according to the decision.
- Security context as a common identifier

```
system_u:system_r:memcached_t:s0
system_u:object_r:var_log_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.

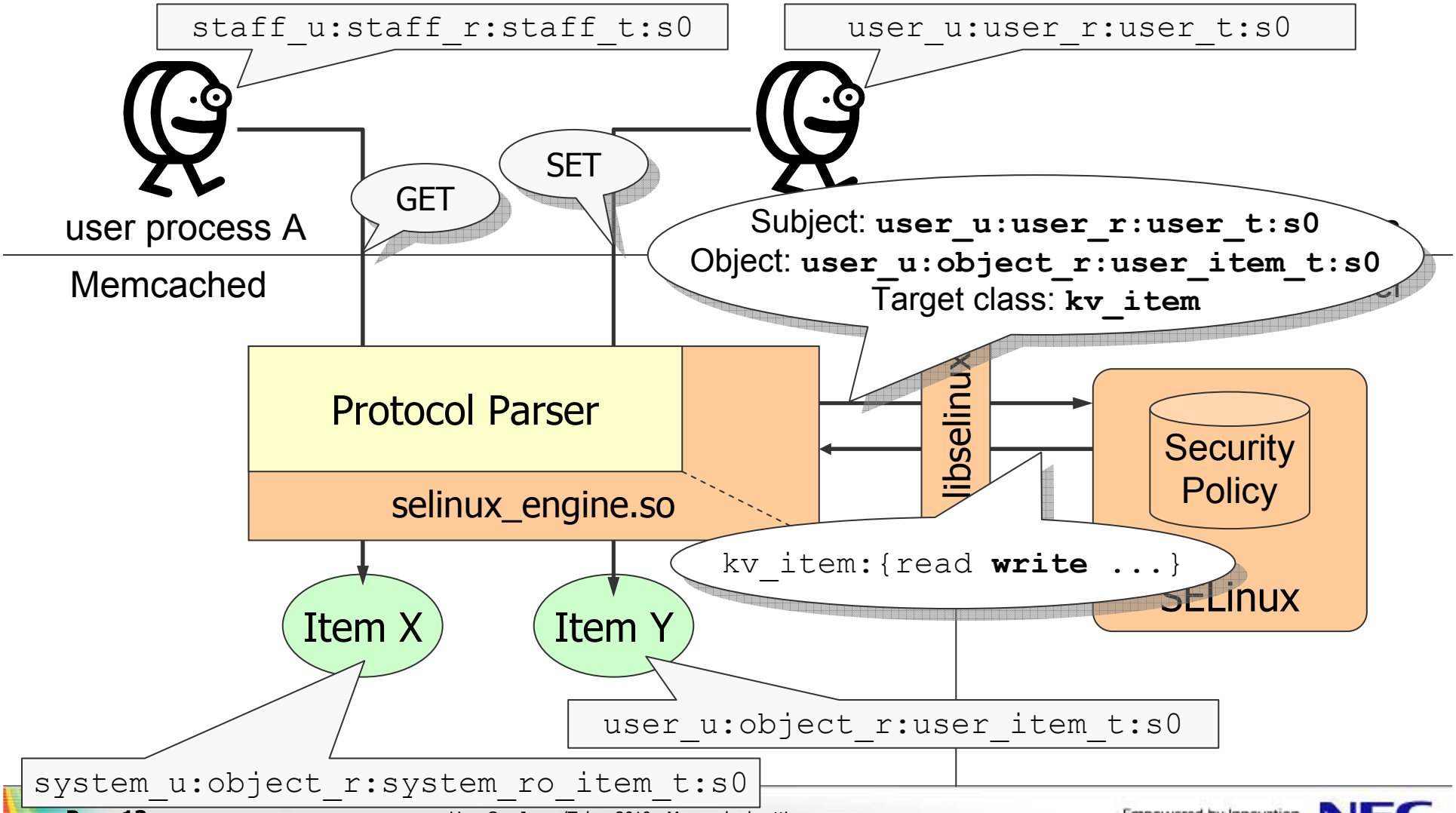
SELinux as a Security Server (2/3)

Case of Linux Kernel



SELinux as a Security Server (3/3)

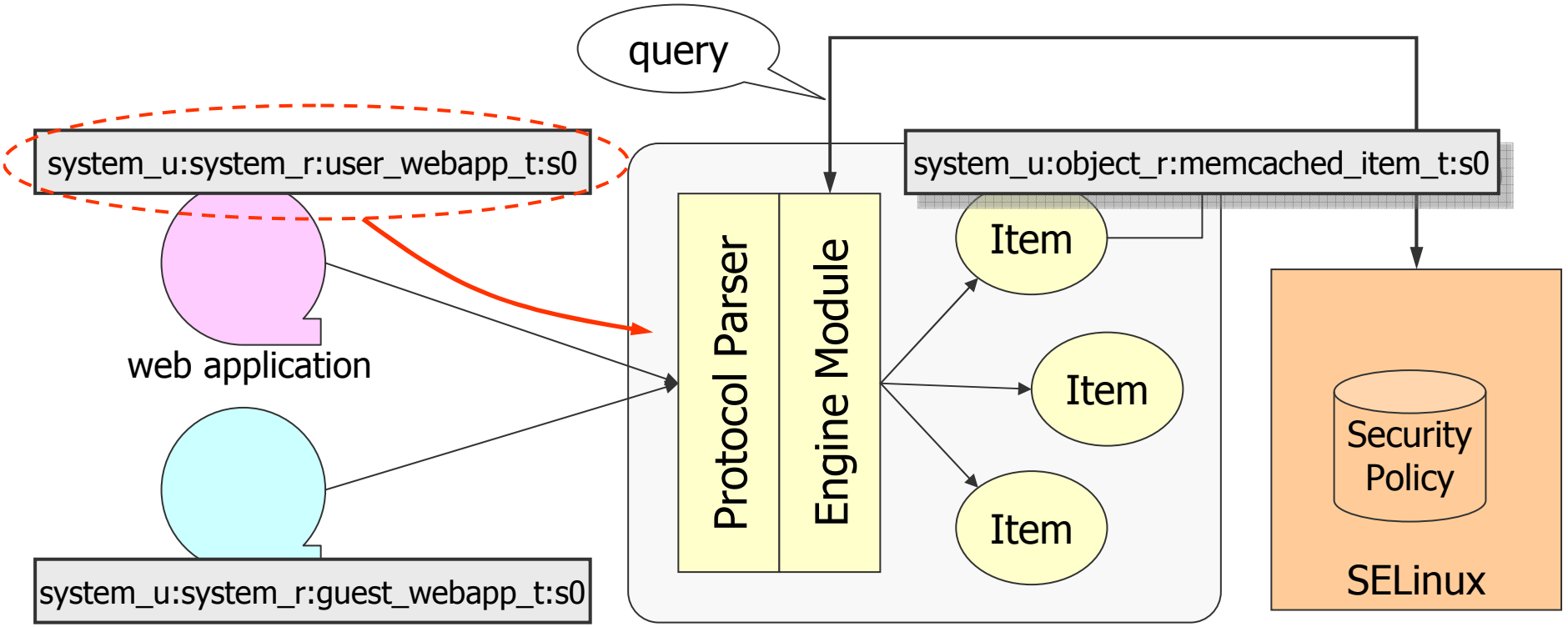
Case of Memcached



1. Memcached and security
   - Background
   - Centralized security and SELinux
2. Getting Memcached secure
   - Adjustment of security model
   - Engine framework performing with libselinux
   - The selinux_engine.so plugin

Needed features to be enhanced

- Memcached needs to get enhanced
  1. Facility to retrieve security context of client process
  2. Facility to assign security context on key-value item
  3. Facility to ask SELinux its access control decision



Security context of the clients

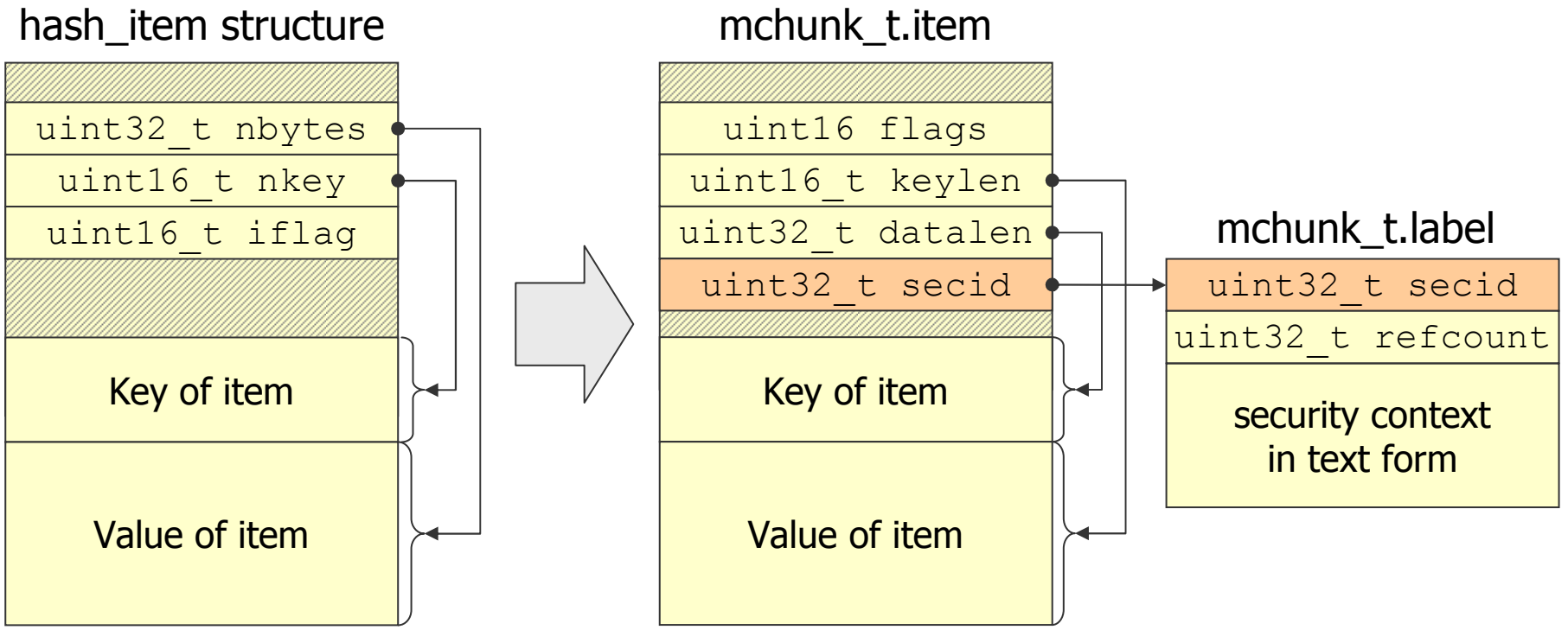
`getpeercon(int sockfd, security_context_t *con)`

- It allows to retrieve security context of the client process that connected to the server using **`sockfd`**.
- If UNIX domain socket, no configurations are necessary
- If TCP/IP socket, also need to set up labeled IPsec.

Labeled IPsec

- It uses an enhanced version of key-exchange daemon that transfers peer security context during IKE exchanges.
- getpeercon(3) enables to retrieve the delivered one.
- For more details:
  Introduction to Labeled Networking on Linux (Paul Moore, HP)
  http://www.linuxfoundation.jp/jp_uploads/seminar20080709/paul_moore-r1.pdf

Security context of key/value item



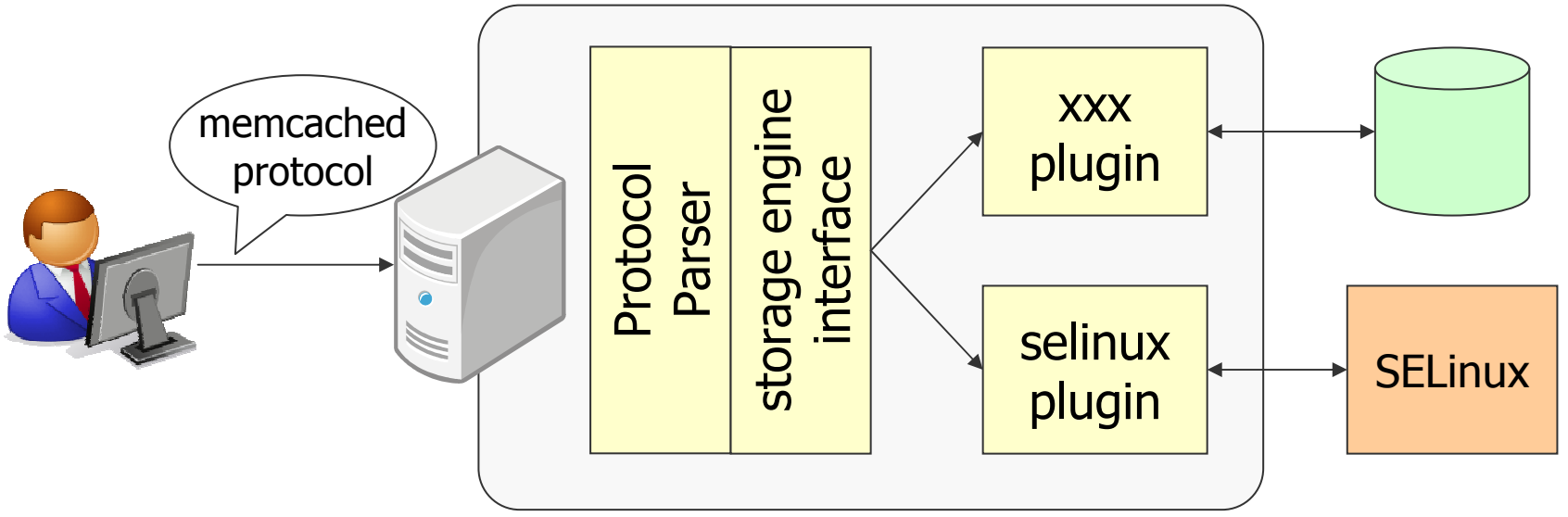
- SELinux needs key-value item to be labeled
  - But original hash_item is not designed to store a security context.
- Revised data format that allows to point a certain security context
  - ✓ Large number of objects tend to share small number of security contexts

memcached - storage engine interface (1/2)

What is the storage engine interface?

- An upcoming feature in memcached v1.6.x
- It allows a plugin to provide its mechanism to manage key/value pair.
- Well designed protocol between the core and engine plugin.
  - Some plugins may provide persistent storage support.
  - Some plugins may provide access control.

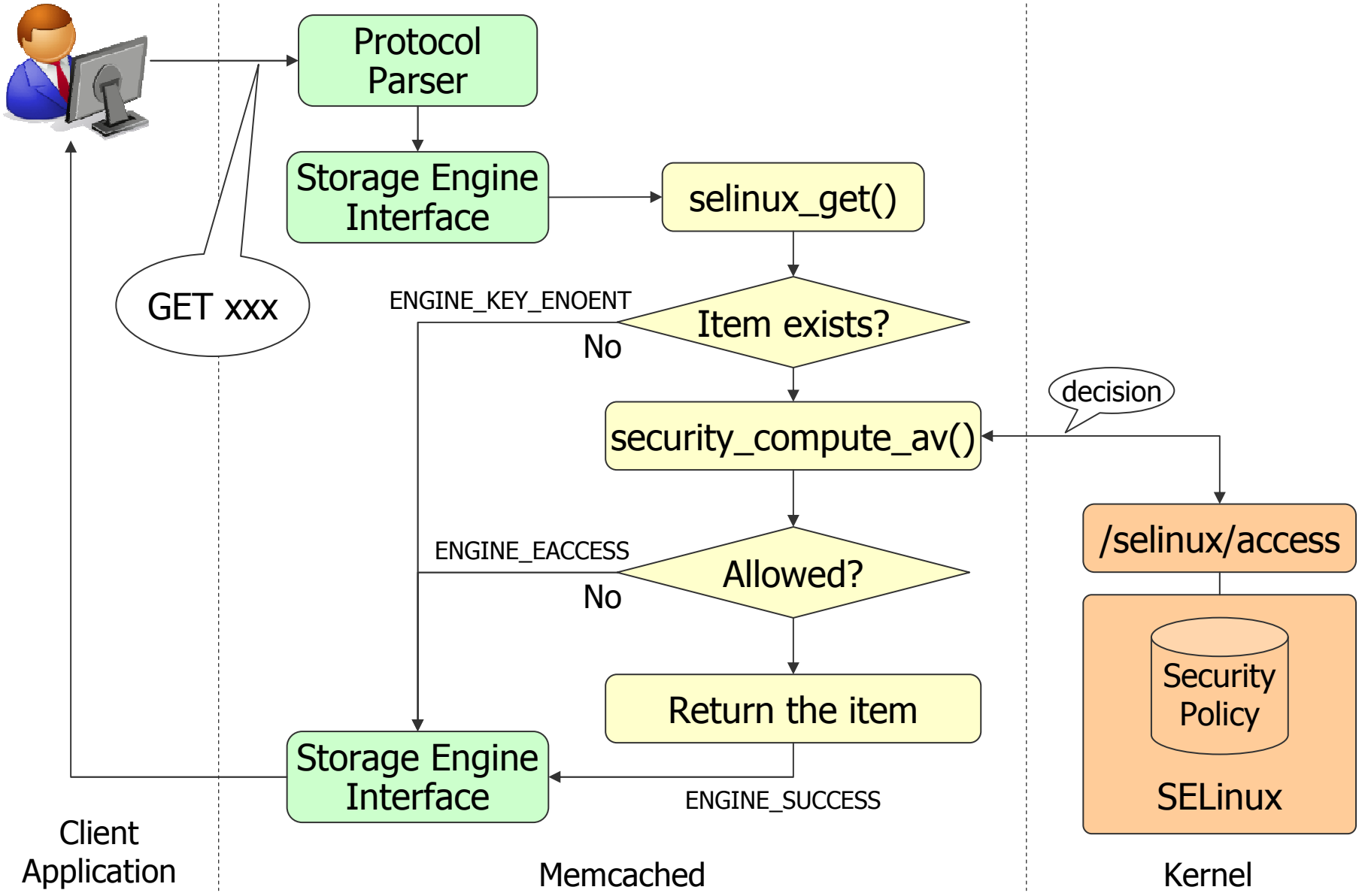
:



memcached - storage engine interface (2/2)

```
typedef struct engine_interface_v1 {
        :
/**
 * Retrieve an item.
 *
 * @param handle the engine handle
 * @param cookie The cookie provided by the frontend
 * @param item output variable that will receive the located item
 * @param key the key to look up
 * @param nkey the length of the key
 * @param vbucket the virtual bucket id
 *
 * @return ENGINE_SUCCESS if all goes well
 */
ENGINE_ERROR_CODE (*get)(ENGINE_HANDLE* handle,
                         const void* cookie,
                         item** item,
                         const void* key,
                         const int nkey,
                         uint16_t vbucket);
        :
}
```

Flow-chart in GET command



selinuxfs and libselinux (1/2)

```
[kaigai@saba ~]$ ls /selinux
access                  context           load                  reject_unknown
avc/                    create            member                relabel
booleans/               deny_unknown      mls                   status
checkreqprot            disable           null                  user
class/                  enforce           policy_capabilities/
commit_pending_bools    initial_contexts/ policyvers
```

selinuxfs

- A pseudo filesystem as an interface to applications
- Eg; write and read on **`/selinux/access`**
  ➔ it asks selinux its access control decision

libselinux

- A set of wrapper functions for selinuxfs and configuration files.
- Eg; **`security_getenforce()`** ➔ read **`/selinux/enforce`**
- Userspace access vector cache

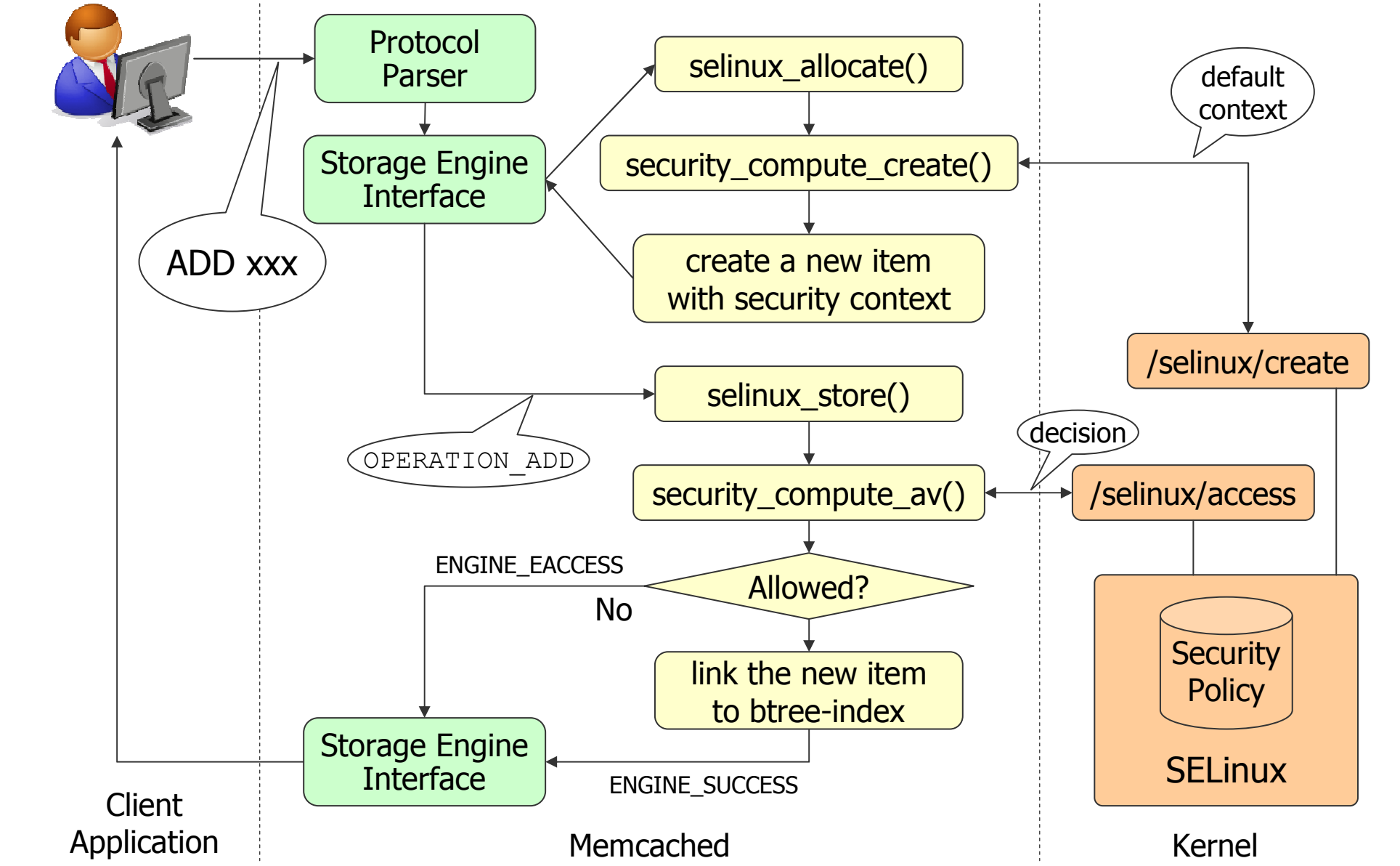
selinuxfs and libselinux (2/2)

```
extern int security_compute_av(const security_context_t scon,
                               const security_context_t tcon,
                               security_class_t tclass,
                               access_vector_t required,
                               struct av_decision *avd);
```

It contains bitmask of **allowed** permissions.

- security_compute_av
  - **scon** ... security context of the user process
  - **tcon** ... security context of the item to be referenced
  - **tclass** ... code of object class
  - **required**... an obsolete argument
  - **avd** ... result shall be set in this structure
  - It writes scon, tcon and tclass to **/selinux/access**, then SELinux returns allowed actions on a pair of them.

Flow-chart in ADD command



Memcached - selinux engine

To obtain the source code

git clone git://github.com/trondn/memcached.git -b engine

svn co http://sepgsql.googlecode.com/svn/trunk/memcached

Features

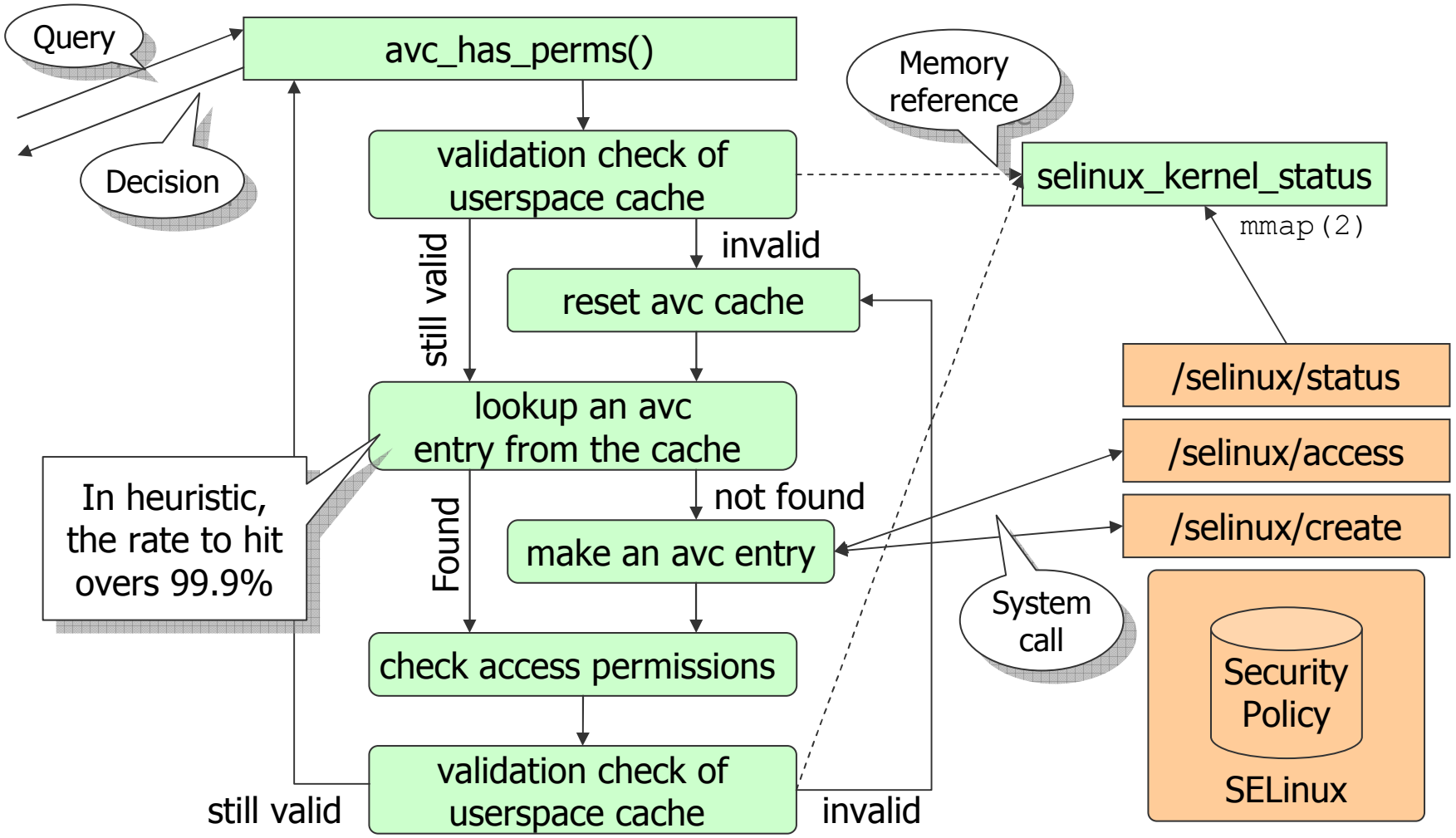
- Mandatory access control with SELinux policy
- Using B+tree index
- Persistent storage support

Future works

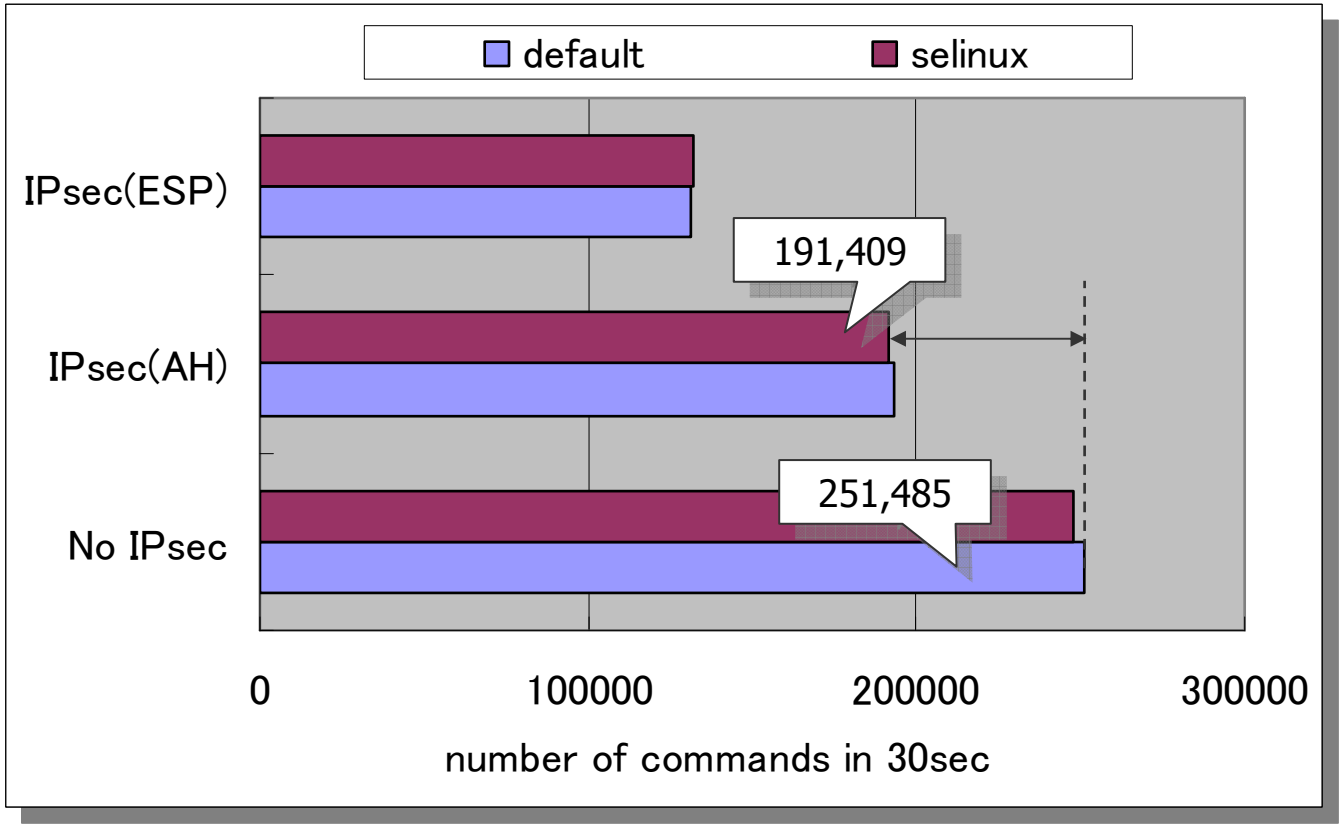
- Waiting for Memcached v1.6.x release :-)
- Pushing the package to Fedora project
- Scalability improvement
- Comprehensive statistical information
- Documentations

Userspace access vector cache (avc)

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



Benchmark



- Iteration of GET/SET mixture, 8threads-client, 4core server x 2, Gb-ether
- Less significant differences in same network environment
  - default = no access control, selinux = mandatory access control
- Penalties in IPsec(AH) communication (~20%?)

Summary

- Why object managers apply access controls
  - Access control should be centralized
    - Consistency
    - Coverage
  - Server is better than applications, Kernel is better than servers.
- SELinux as a Security Server
  - SELinux returns its access control decision,
    then object manager control accesses according to the decision.
  - User and data object need to be identified with security context.
- Using libselinux
  - Libselinux encapsulates raw accesses to selinuxfs.
  - Userspace access vector cache reduces number of kernel invocations

Any Questions?

Empowered by Innovation NEC

Thank you!