Mandatory Access Control for Carrier-Grade Linux Clusters

(as part of the DSI project)

Miroslaw.Zakrzewski@Ericsson.ca

Ericsson Research Canada

Open System Lab

Montréal – Canada

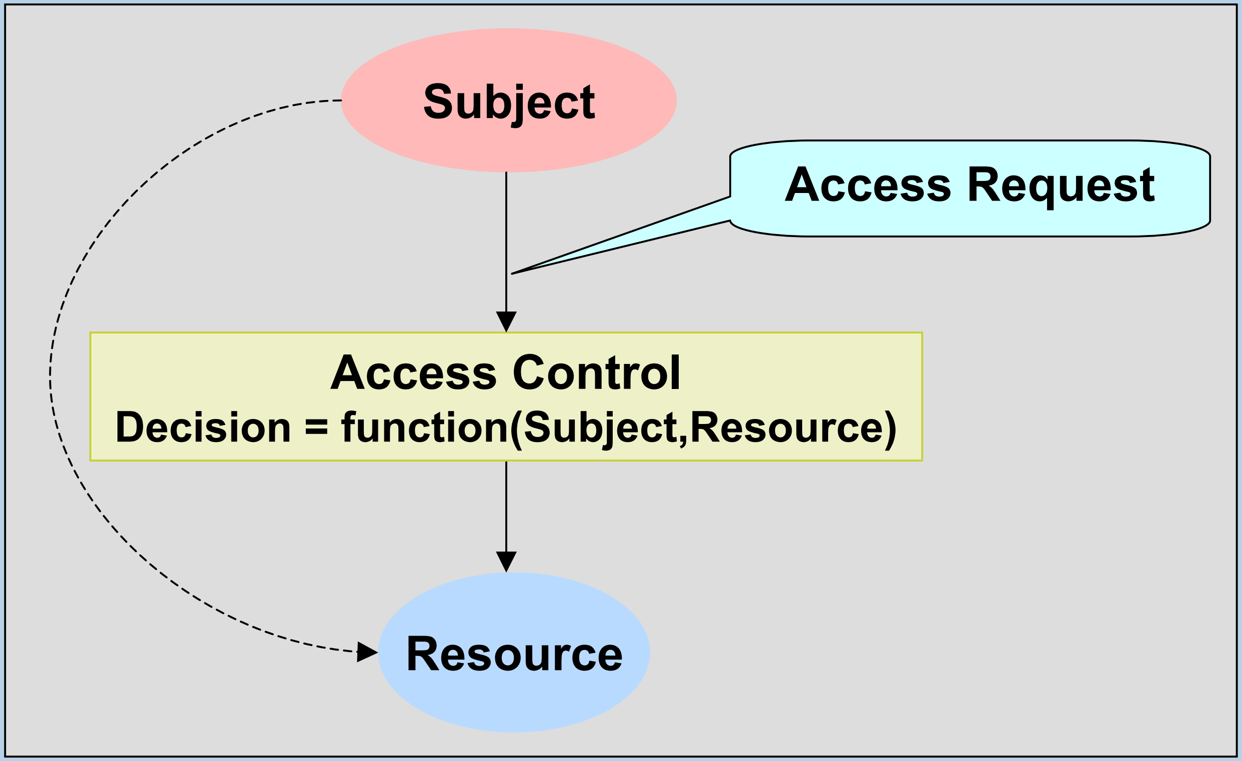
http://www.risq.ericsson.ca

The purpose of the presentation is to explain about ongoing implementation of a Distributed Security Module that provides Mandatory Access Control within a Linux Cluster.

Outline

- Introduction
- DSI Characteristics
- Access Control - General Architecture
- Distributed Security Module
- Security Distribution in DSM
- Demo Architecture (Local and Remote Access)
- Challenges

Introduction (1/5)



Introduction (2/5)

- **Discretionary Access Control**
  - Ordinary users involved in the security policy definition
  - Access decisions based on user identity and ownership
  - Two category of users :
    - completely trusted administrators (root)
    - Completely untrusted ordinary user

Introduction (3/5)

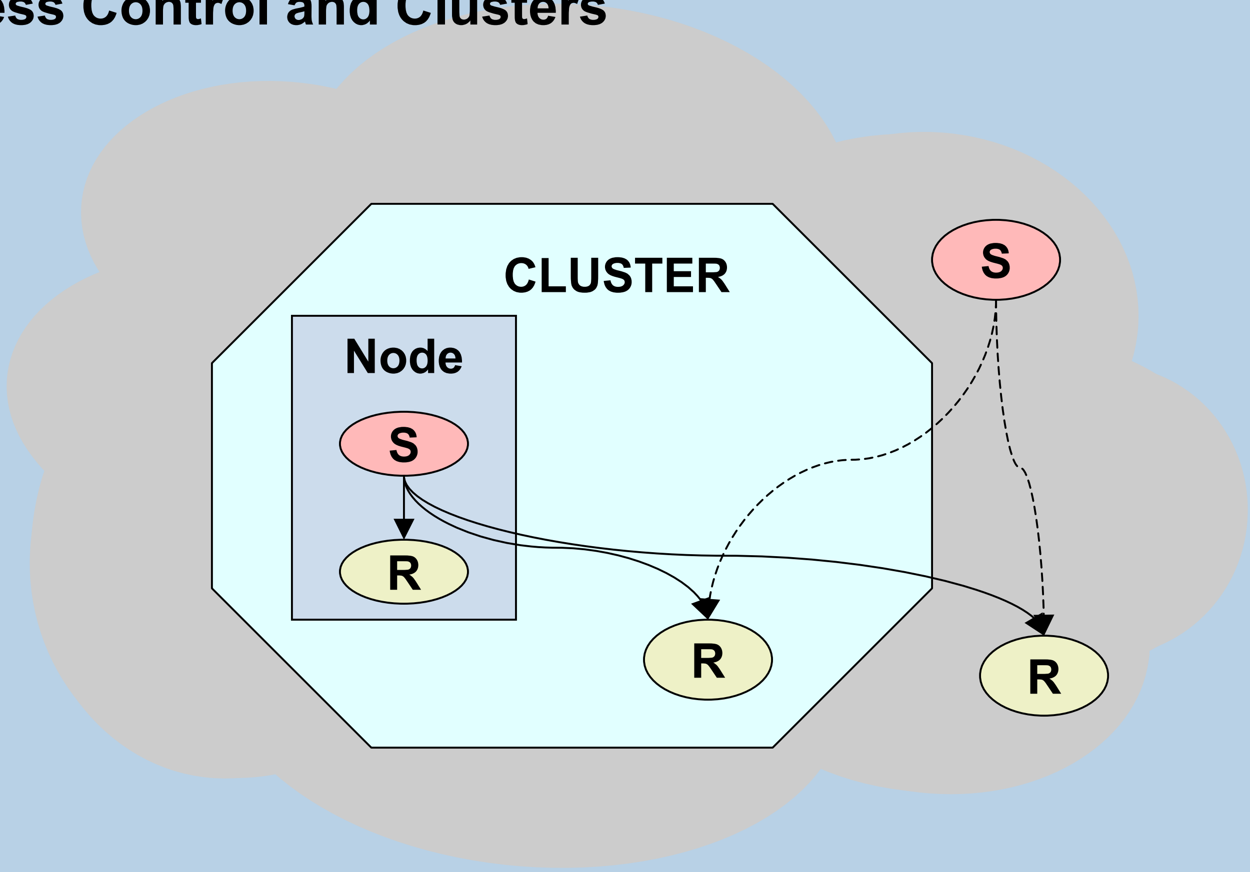
- **Mandatory Access Control**
  - policy definition and assignment of security attributes is controlled by a system security administrator
  - access decisions are based on labels that contain a variety of security-relevant information (every subject and object in the system is labelled)

Introduction (4/5)

- **Cluster:** A collection of interconnected stand-alone computers working together to solve a problem as a single computing entity

Introduction (5/5)

- Access Control and Clusters



Cluster Access Types

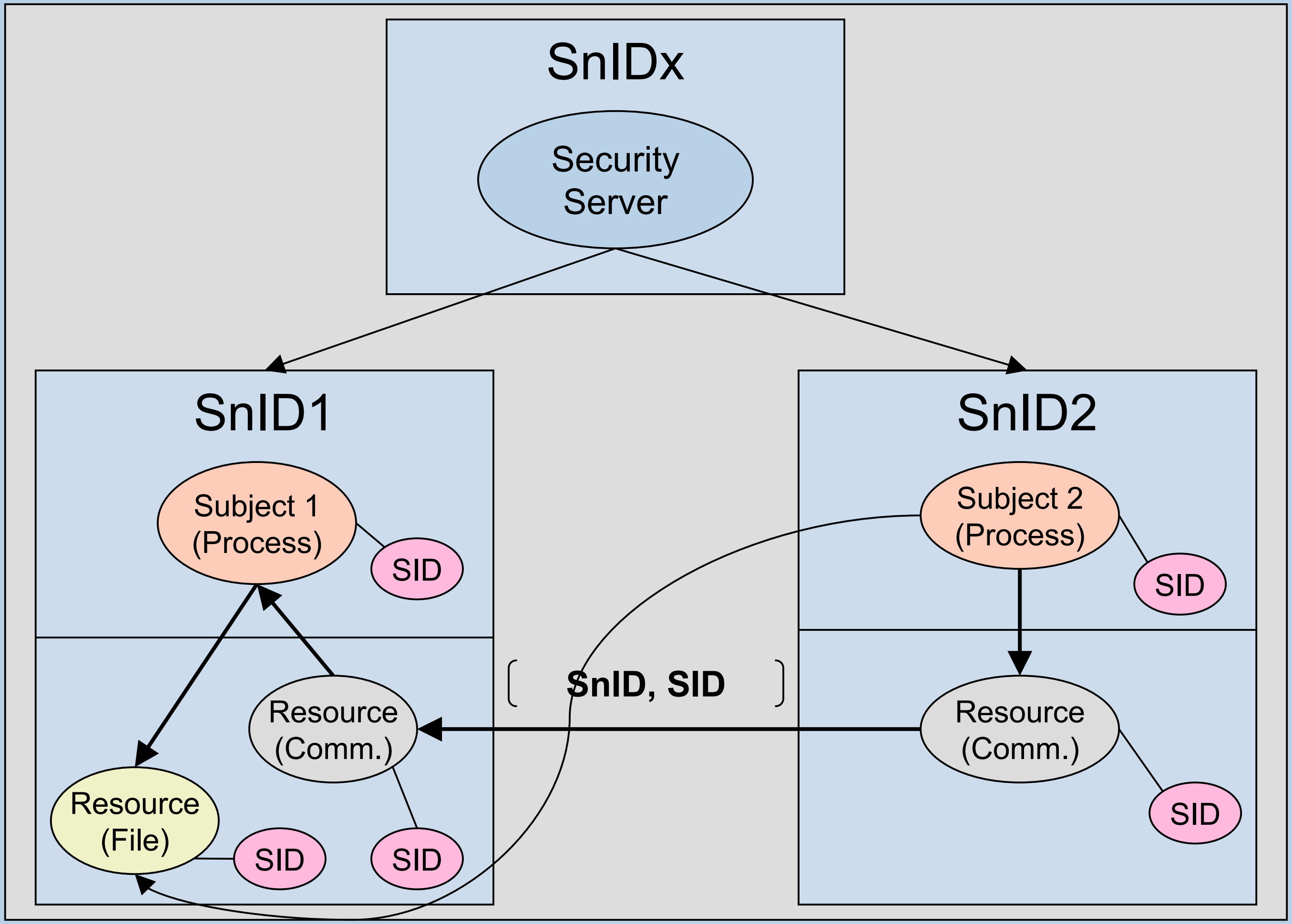
- **Cluster Local Access**
  - subject and resource on the same node inside the cluster
- **Cluster Remote Access**
  - subject and resource on different nodes inside the cluster

- Cluster Outside Access
  - subject inside cluster, resource outside cluster
  - subject outside cluster, resource inside cluster
- No Cluster Access
  - both subject and resource outside cluster

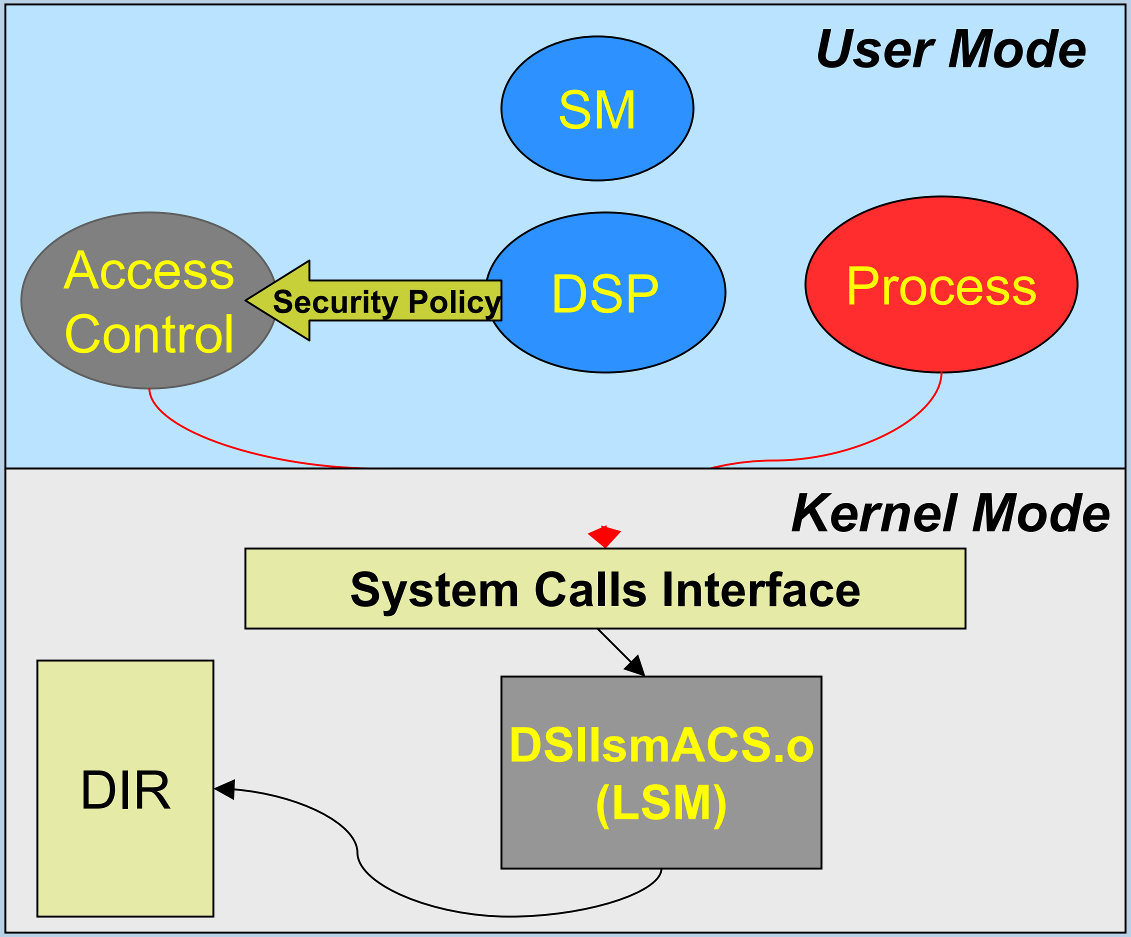
DSI Characteristics
(please see paper on DSI)

- **Process Level Approach**
  - Controlling Single Process
- **Pre-emptive Security**
  - Run-time changes of security attributes
  - Security can be modified without stopping the system
- **Minimal Impact**
  - Performance
  - Transparency
- **Distributed**
  - Clusters

Access Control – General Architecture (1/2)



Access Control – General Architecture (2/2)



Legend:
SM Security Manager
DSP Distributed Security Policy
LSM Linux Security Module
DIR DSP Internal Representation

Distributed Security Module

- **DSM is implemented in Kernel Space.**
  - Performance
  - Transparency

- **DSM uses LSM Framework.**
  (please see paper on LSM)
  - Pre-emptive security
  - Process Level Approach

- **DSM uses IP Options.**
  - Distribution

Linux Security Module Framework (LSM)
(Used by Distributed Security Module)

- Patch to Linux Kernel by WireX (based on NSA prototype)
- Security Hooks - points the kernel to allow the control of nearly every system operation
  - 140 hooks
  - 29 classes
- Flexible:
  - Easy to add user defined security implementations
- Function pointers in terms of programming

LSM Installation for Kernel 2.4.17

http://lsm.immunix.org

```
get lsm-full-2002_01_15 patch for kernel 2.4.17

gunzip lsm-full-2002_01_15-2.4.17.patch.gz

cd /usr/src/linux

patch –p1 < /home/lmcmzak/lsm-full-2002_01_15-2.4.17.patch

rebuild the kernel
```

LSM Framework

- New Code

```
<linux/security>
<include/linux/security.h>
```

- New Global

```
struct security_operations *security_ops;
    /* pointer to all security operation in the kernel */
struct security_operations dummy_security_ops;
    /* set of dummy functions */
```

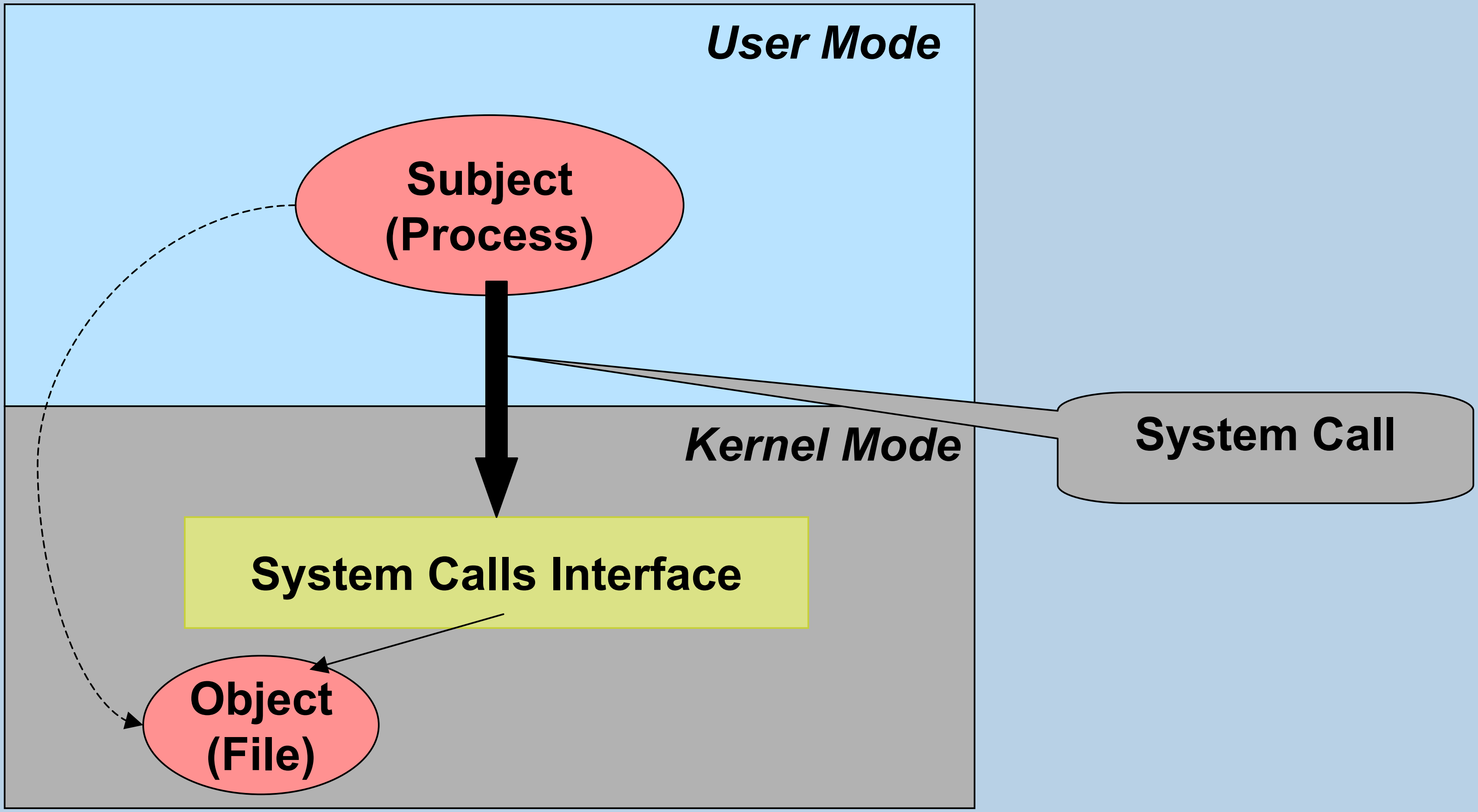
LSM Framework

- Function to Register and UnRegister Security Operation to the Kernel

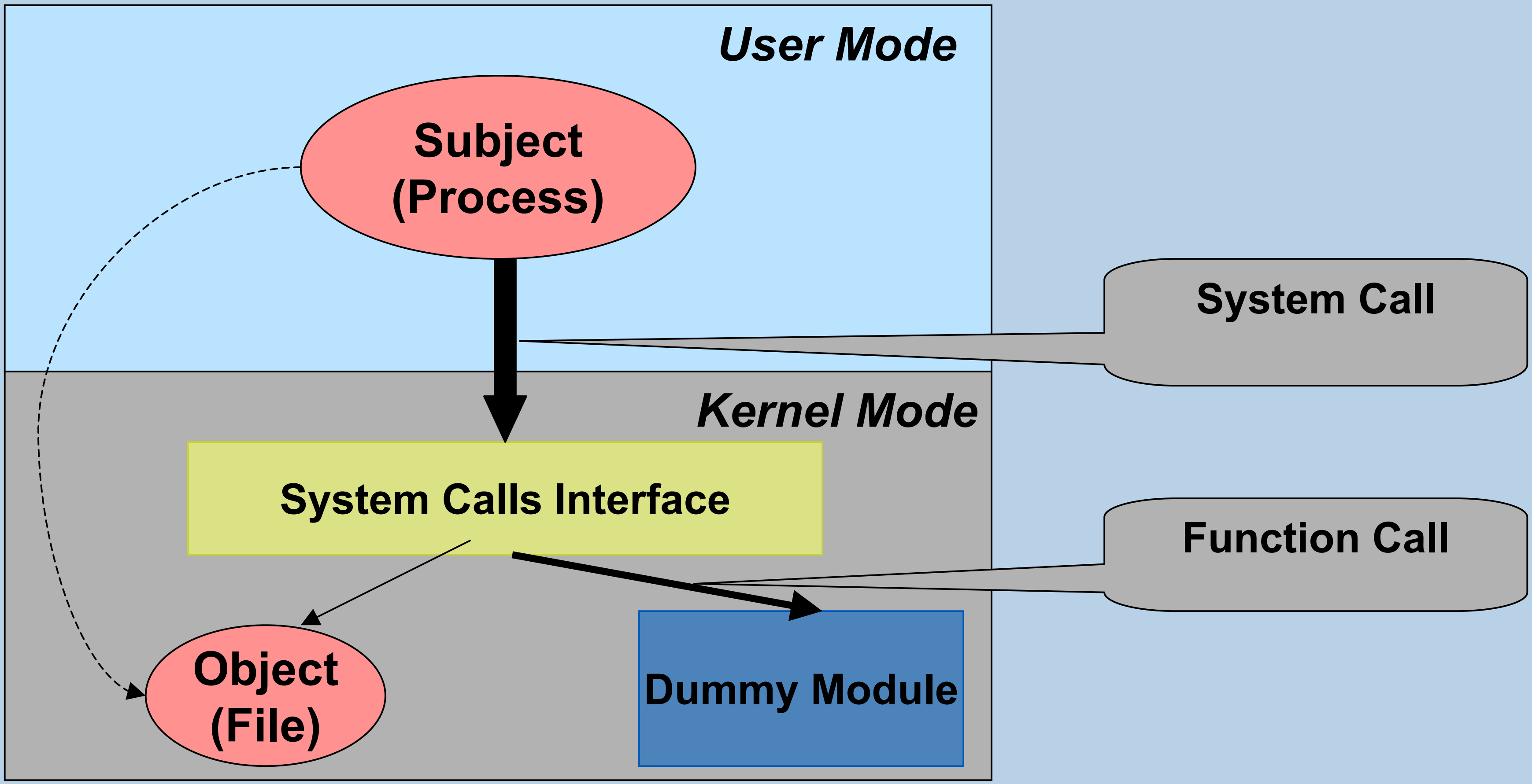
```
int
register_security (struct security_operations *ops);

int
unregister_security (struct security_operations *ops);
```

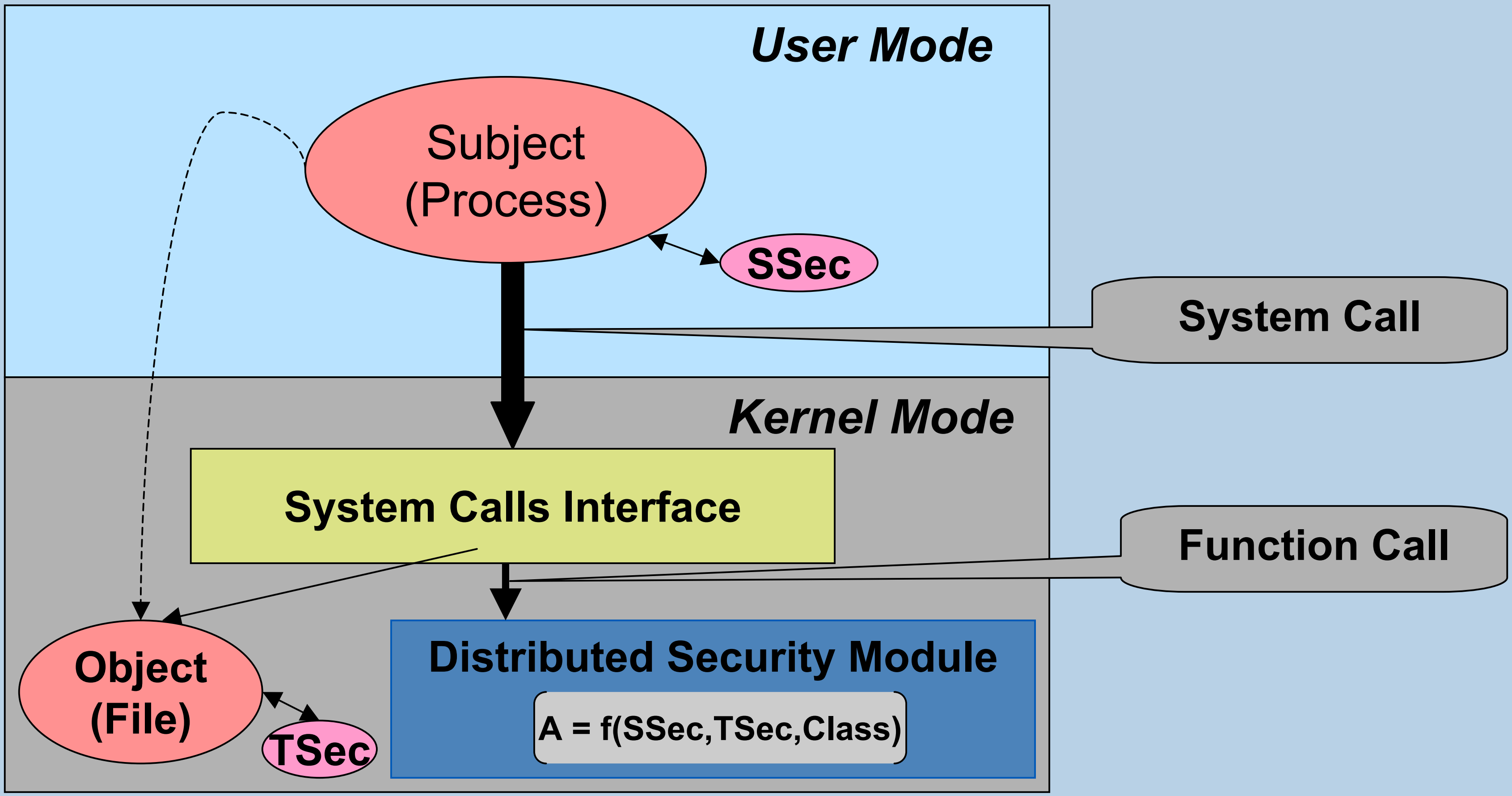
Linux Access Control



Linux Access Control and LSM Framework



Linux Access Control and DSM



Labels in DSM

- Objects attached to Linux structures

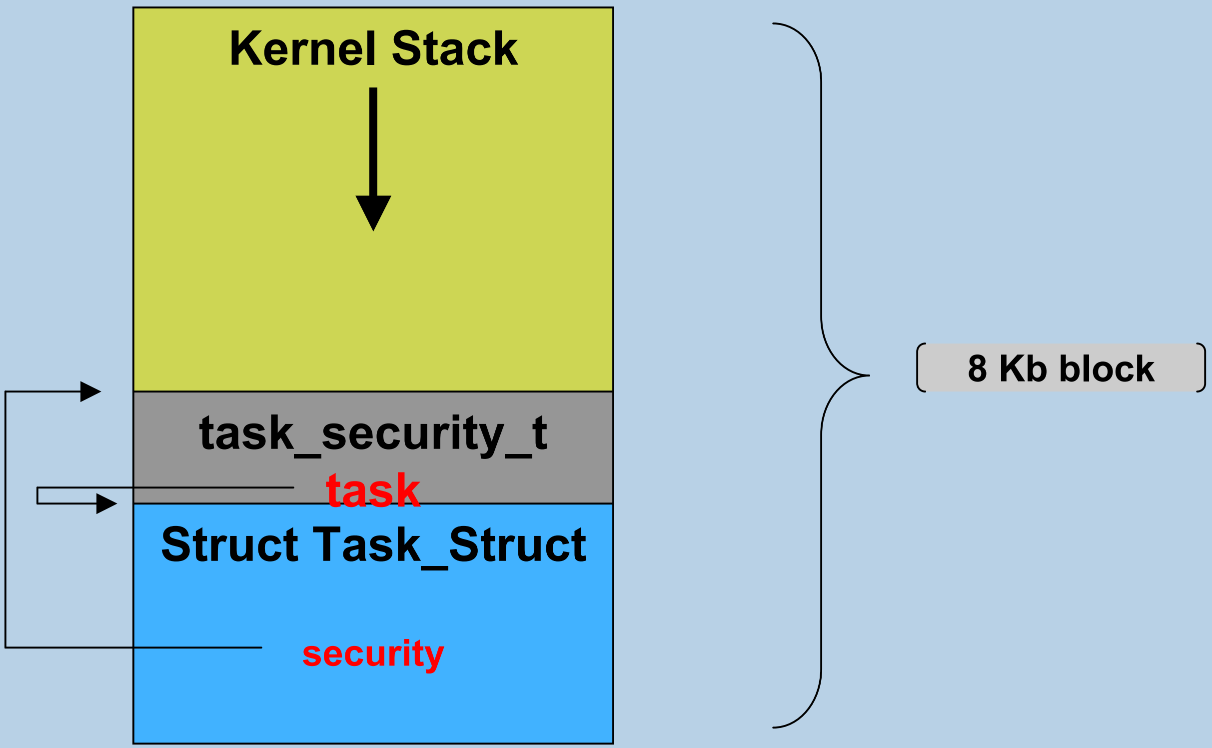
- Example : task label (object attached to task structure
  `struct task_struct` <linux/sched.h>)

```
struct task_struct {
        .
        void *security;
        .
}
```

Task Security Label Format in DSM

```
typedef struct {
        int     sid;
        ...
        ...
        void    *task;
} task_security_t;
```

Task Label in relation to task structure in DSM



Task Label Attachment in DSM

- All running tasks are labelled when the security module is loaded (sid is set to default value)
- After the security module is loaded the tasks are labelled using security hooks (two step process) :
  - **Fork** : sid of parent
  - **Exec** : sid can be modified based on the sid stored in the image (SID is embedded in the ELF format)

Security System Calls in DSM

- Set Node ID
- Change Task SID
- Set Policy
- Check Alarms

Security Distribution

- **Security Information transfer**
  - IP level (first)
  - IP header modification
  - Kernel hooks for IP traffic handling
  - Security information (SID, SnID) transfer as an option in IP header
  - Implementation based on Selopt implementation for SELinux by James Morris
- **IP Options**
  - Commercial Internet Protocol Security Option (CIPSO)
  - Federal Information Processing Standard (FIPS) - 188

Security Distribution

- **Network Labels**
  - Labels used when performing remote access (subject and resource on different nodes)
  - Security Node ID (SnID) and Security ID (SID) of the subject are added to the IP message
  - On the receiving side these two information are extracted and used to build the network security ID (NSID)

    NSID = Function (SnID, SID)
  - NSID is used as a local label for access control decisions

Security Distribution

- Network Buffer Label
  - Socket Buffer (<linux/skbuff.h>) – object to contain network packets in kernel

```
struct sk_buff {
          .
          void *lsm_security;
          .
}
```

Network Labels

- sk_buff Security Label Format

```
typedef struct {
        int                 sid;
        . . .
        struct sk_buff      *sk_buff;
} sk_buff_security_t;
```

Network Labels

- sk_buff Security Label Attachment (sending side)
  - Security ID of sk_buff is taken from Security ID of the sending socket
  - Security Node ID is set up by the security server and is global in LSM module

Network Labels

- Security Information in Network Message
  - Message is modified on IP layer (adding options)
  - Security Node ID is taken from LSM module and attached to the message
  - Security ID is taken from sk_buff Security Label and attached to the message

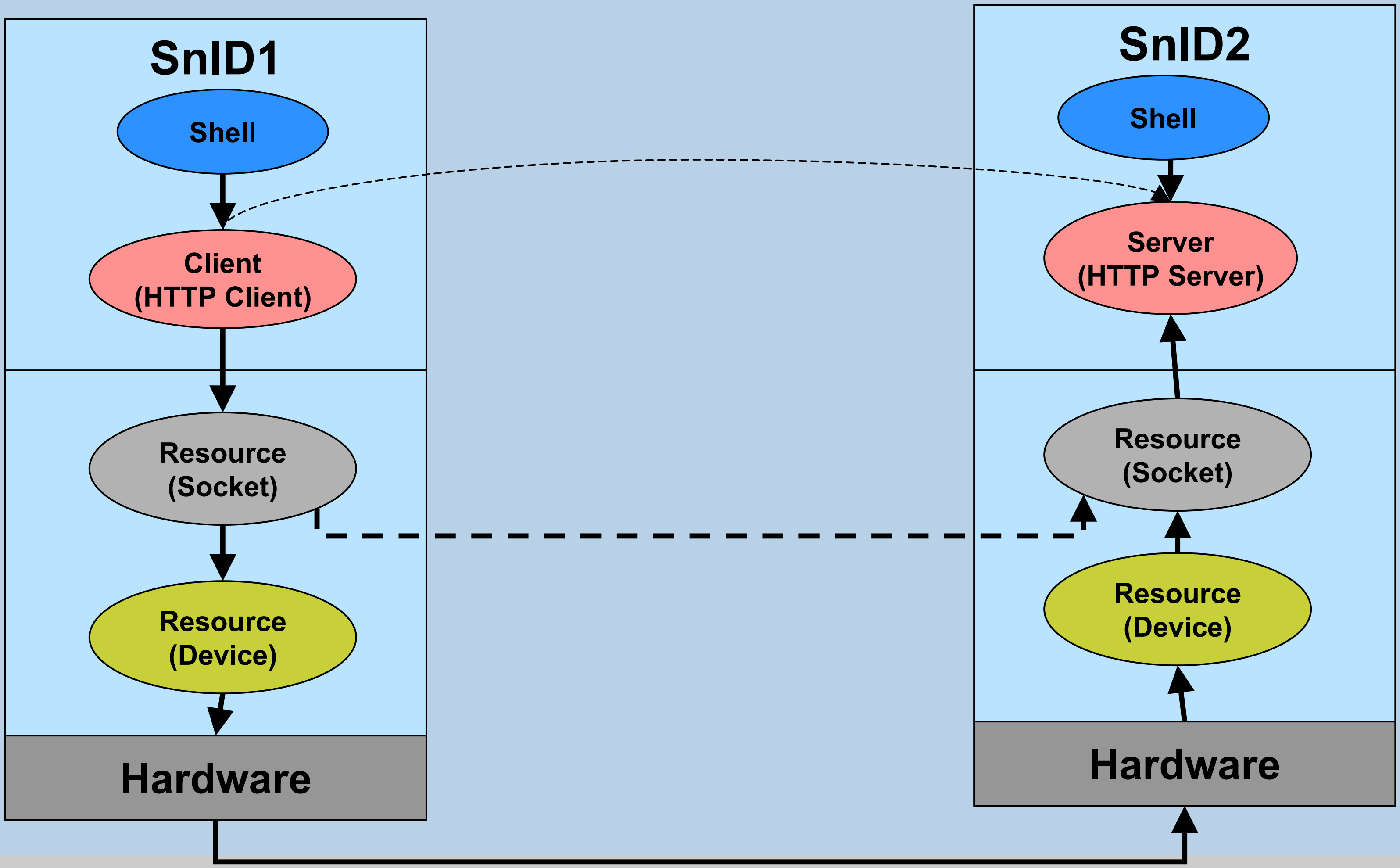
Network Labels

- sk_buff Security Label Attachment (receiving side)
  - Extracting Security Node Id (SnID) and Security ID (SID) from the incoming message
  - Converting SnID and SID pair to Network Security ID (NID) based on the conversion table :

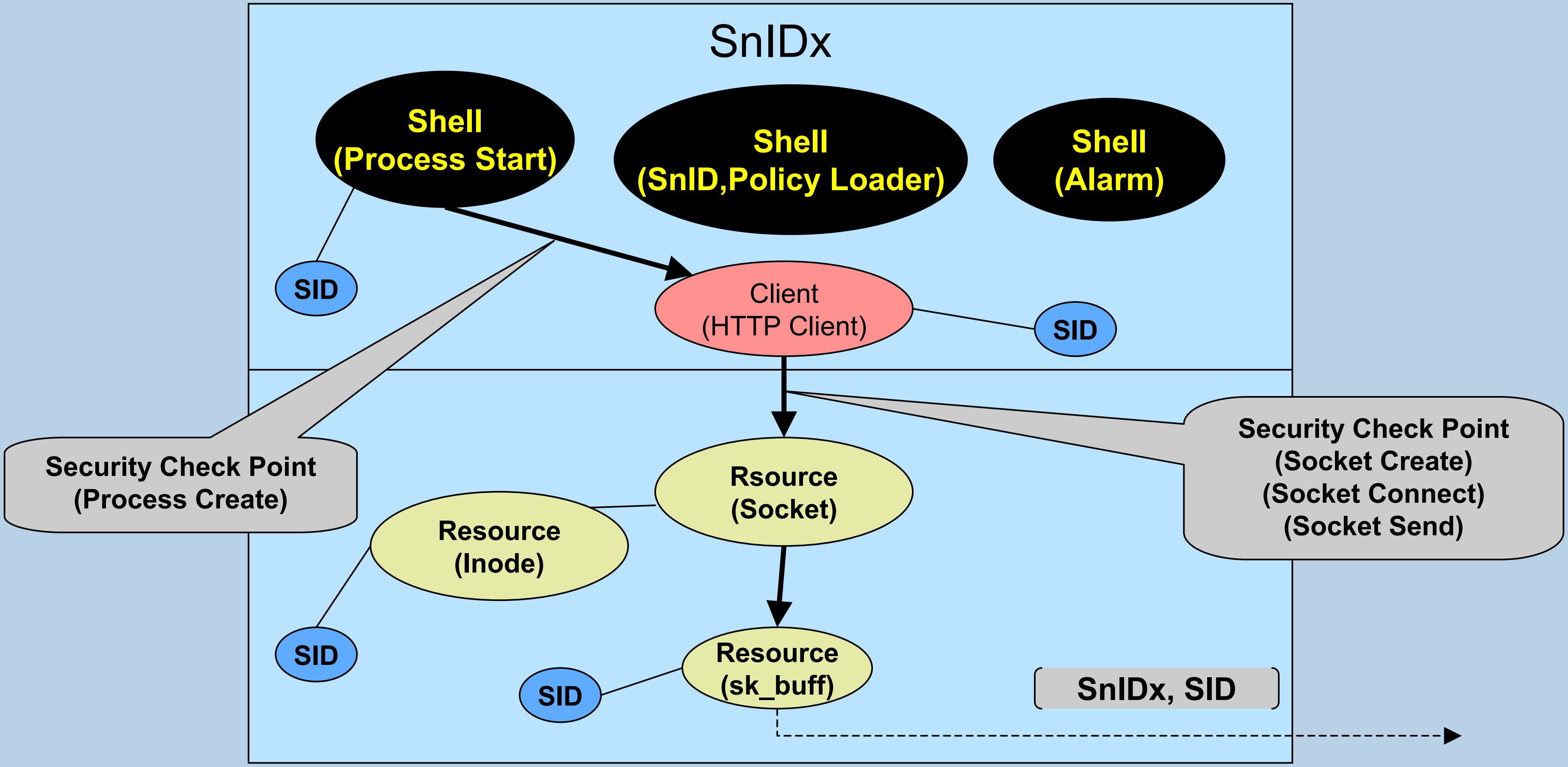
    NID = Fun(SnID,SID)

  - NID will be treated as a local label (local access control)

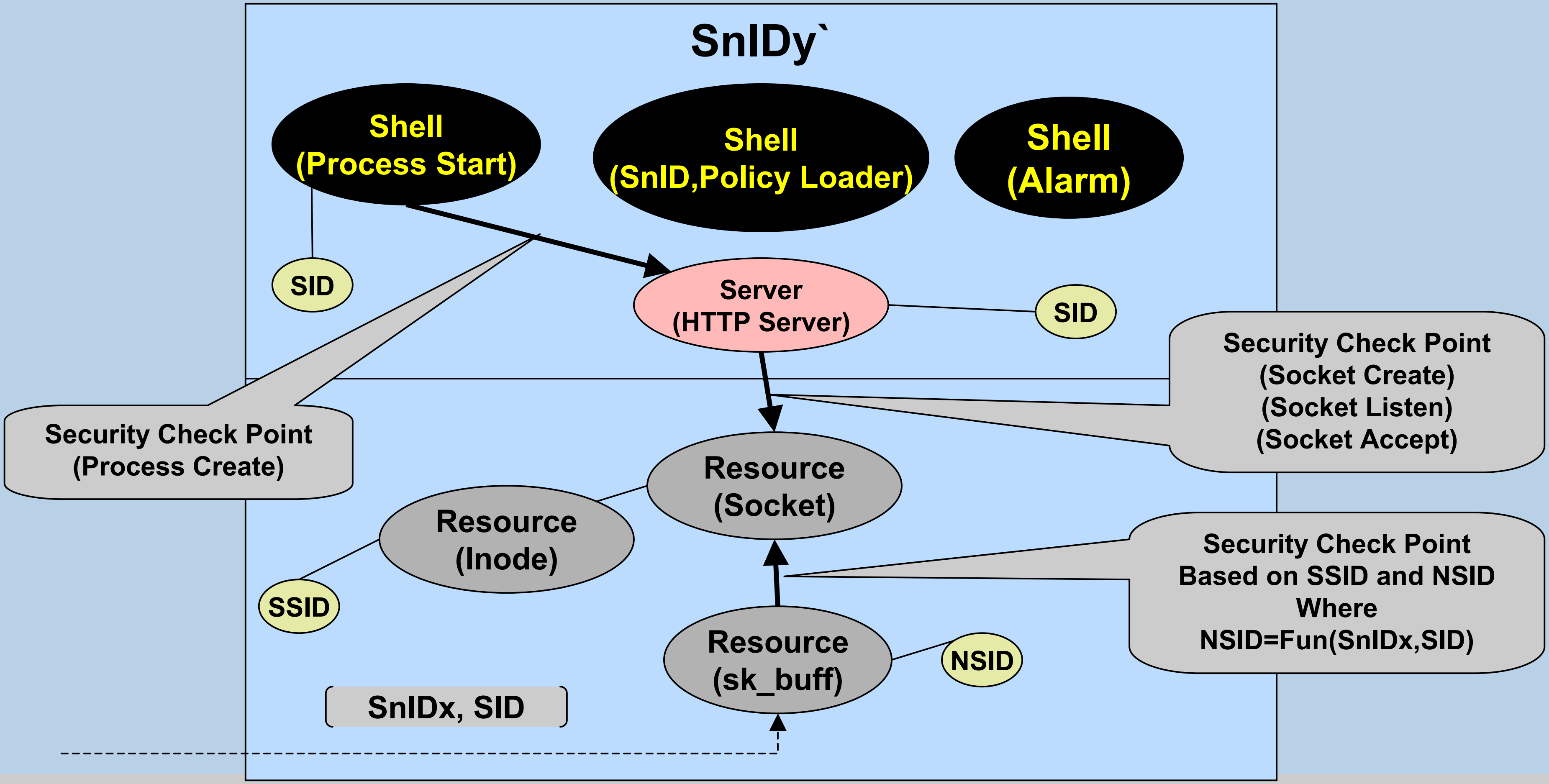
Demo Architecture



Remote Access Control - Demo (sending side)



Remote Access Control - Demo (receiving side)



Challenges: Performance testing

- **Test Types**
  - UDP Local Access (Send Message)
  - UDP Remote Access (Loopback)
- **Results**
  - Performance with IP packet modification
  - Performance without IP packet modification
  - Buffer overflow

Performance Test Results (1/2)

- Performance with IP packet modification

  (all numbers are in microseconds)

	Linux 2.4.17	Linux 2.4.17 with DSM	% Overhead
UDP Local Access (Send Message)	16.388	19.7	+20%
UDP Remote Access (Loopback)	133.44	173.88	+30%

Performance Test Results (2/2)

- Performance without IP packet modification

  (all numbers are in microseconds)

	Linux 2.4.17	Linux 2.4.17 with DSM	% Overhead
UDP Local Access (Send Message)	16.388	17.084	+4.2%
UDP Remote Access (Loopback)	133.44	140.64	+5.4%

Ongoing work

- Performance optimization
- Server resource access on behalf of a client
- Security information protection
- Security information transfer on lower levels of the protocol stack
- Test the new cluster security against different types of attacks
- Investigate the impact of the security information on the resources outside the cluster

References

All references are available from the paper.

DEMO

Questions?

Miroslaw Zakrzewski
Ericsson Research – Corporate Unit

Ericsson Canada Inc.

8400 Decarie Blvd	Phone: 1.514.345.7900 x6458
Town of Mount Royal	Fax: 1.514.345.6105
Quebec H4P 2N2	Email:Miroslaw.Zakrzewski@Ericsson.ca

http://www.risq.ericsson.ca