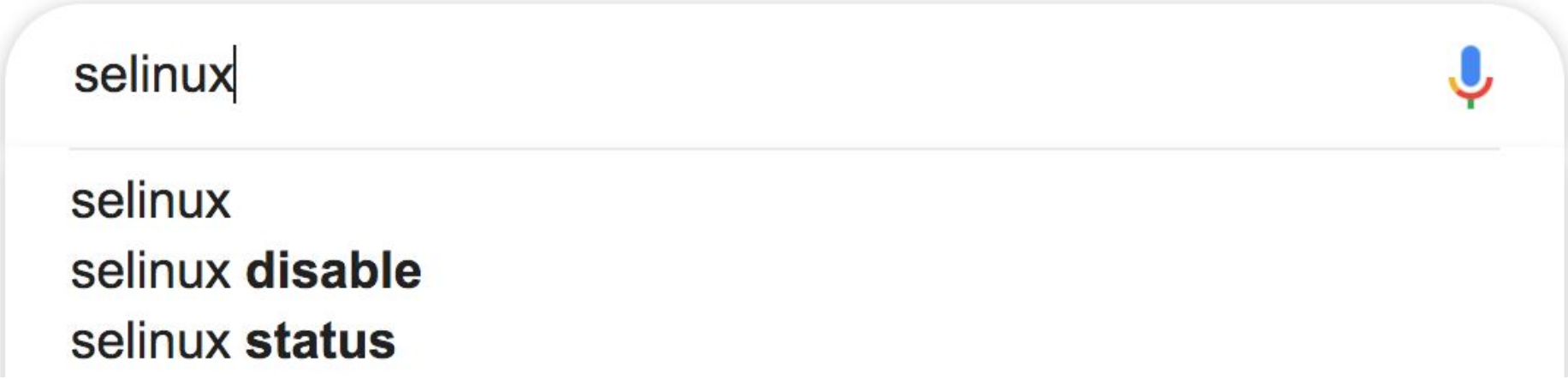
HITCON 101 Sharing
SELinux
從不認識到在一起

About Me

王禹軒 (Bighead)

- 中央大學 Advanced Defense Lab
  - 打胖
- 工研院 Intern
  - Whitelist 1.0 PoC
  - Hypervisor-based Whitelist (page verification)
  - SELinux

SELinux Top Search



The ways to disable SELinux

- Setenforce 0
- Edit /etc/selinux/config : SELINUX = permissive or disable
- Delete policy
- Get rid of the boot argument : security=selinux selinux=1

The ways to disable SELinux

- Setenforce 0
- Edit /etc/selinux/config : SELINUX = permissive or disable
- Delete policy
- Get rid of the boot argument : security=selinux selinux=1
- Do ***NOT*** use default SELinux-enabled distro (CentOS)

The ways to disable SELinux

- Setenforce 0
- Edit /etc/selinux/config : SELINUX = permissive or disable
- Delete policy
- Get rid of the boot argument : security=selinux selinux=1
- Do ***NOT*** use default SELinux-enabled distro (CentOS)

SELinux gives you the power to close it

Don’t be Afraid of SELinux

- 60 page survey paper
- 400 page SELinux Notebook
- Makefile survey
- Policy Set survey
- Powerful mentor

Don’t be Afraid of SELinux

- 60 page survey paper
- 400 page SELinux Notebook
- Makefile survey
- Policy Set survey
- Powerful mentor

Don’t be afraid! It is not scary

Trust Lovely Santa Claus

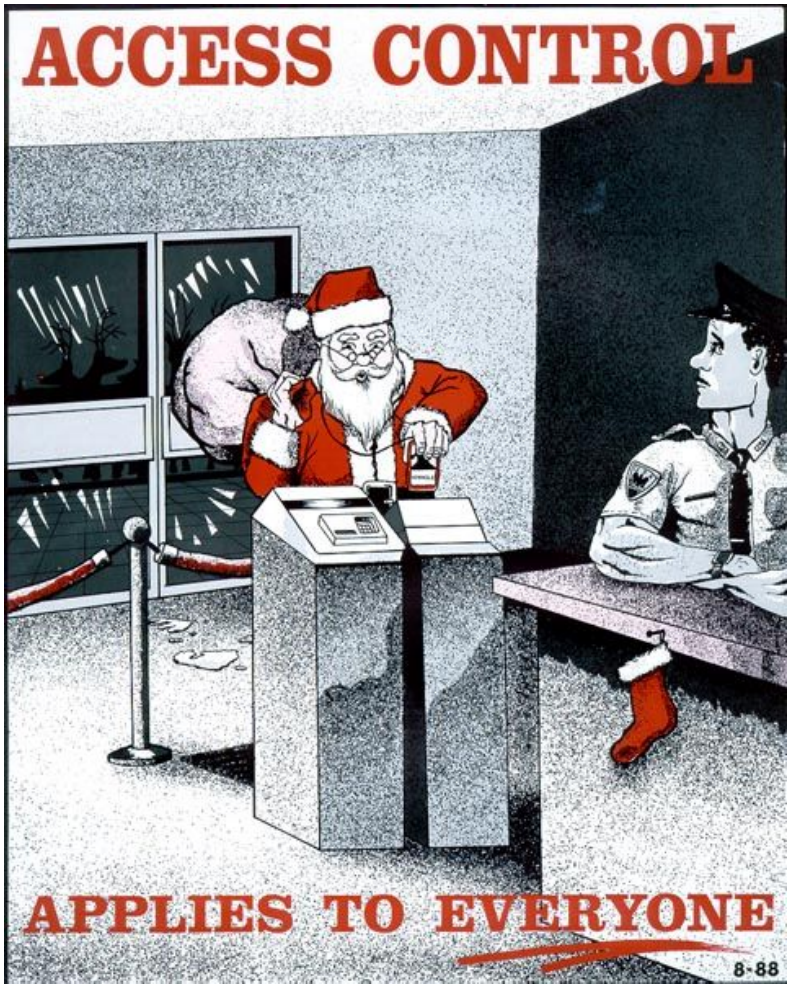
Reference : Santa Claus PNG Transparent Image - PngPix

Trust Evil Santa Claus !?

Futurama : Robot Santa Claus

Why Access Control ?

- **Goal**: Protect data and resources from unauthorized use
  - **Confidentiality** (or secrecy) : Related to disclosure of information
  - **Integrity** : Related to modification of information
  - **Availability** : Related to denial of access to information



Reference: Security Awareness Posters

Access Control Basic Terminology

- Subject: Active entity – user or process
- Object: Passive entity – file or resource
- Access operations: read, write, ...

Access Control is Hard Because

- Access control requirements are **domain-specific**
  - Generic approaches **over-generalize**
- Access control requirements can **change**
  - Anyone could be an administrator

Reference : https://profile.cheezburger.com/imaguid/

Basic Concepts of Different Access Control Policies

- **Discretionary (DAC)**: (**authorization-based**) policies control access based on the **identity** of the requestor and on access rules stating what requestors are (or are not) allowed to do.
- **Mandatory (MAC)**: policies control access based on mandated regulations determined by a **central authority**.

DAC : Access Matrix Model

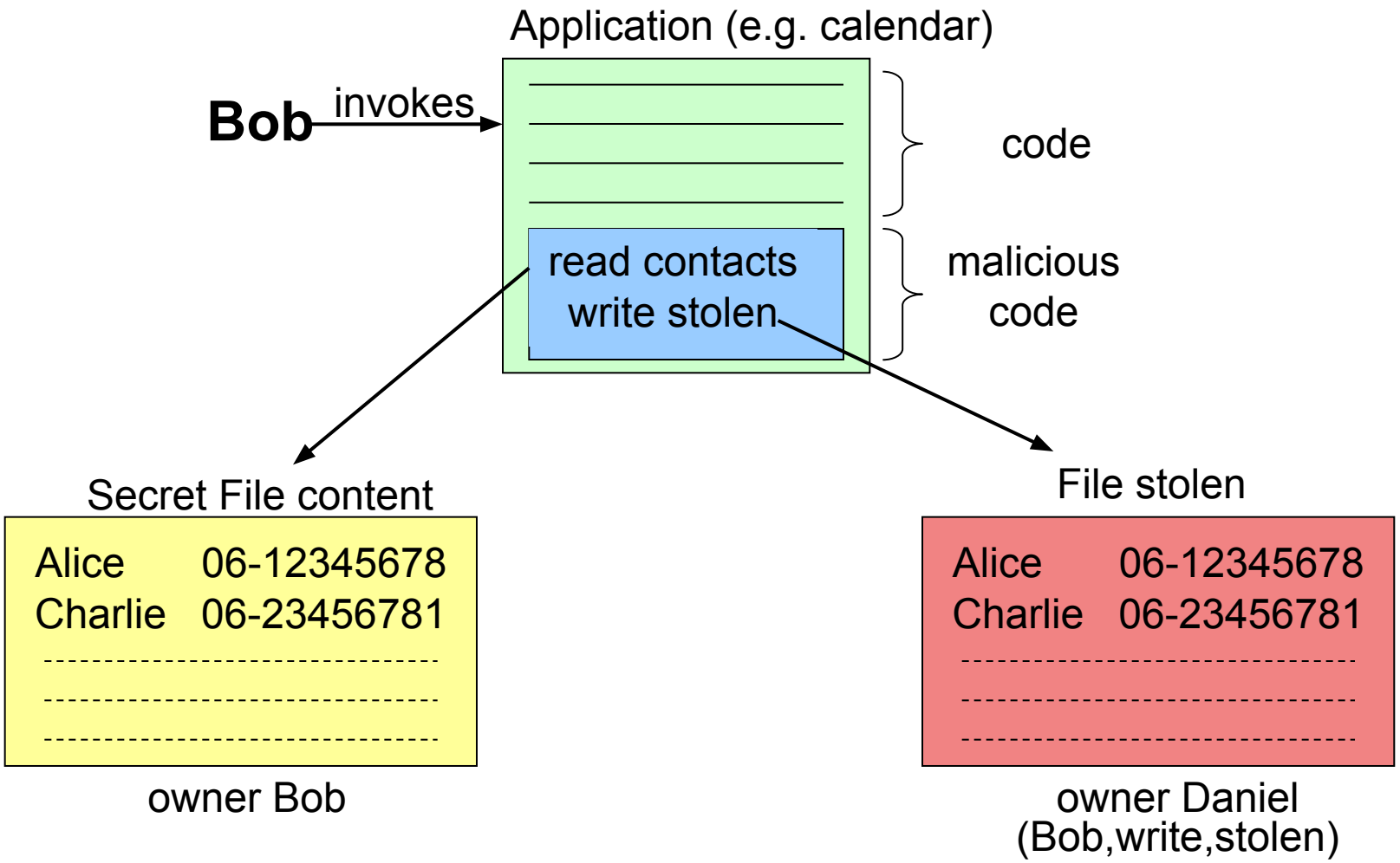
	File 1	File 2	File 3	Program 1
Alice	own read write		read write	
Bob	read	read write		execute
Charlie			read	execute read

DAC - Identity !!

DAC weaknesses (1/2)

- Scenario
  - Bob owns a secret file, Bob can read it, but not Daniel
  - In DAC, Bob can be cheated to leak the information to Daniel.
  - How?
    - Trojan horse: software containing hidden code that performs (illegitimate) functions not known to the caller

Trojan horse - Simple Example



DAC weaknesses (2/2)

- DAC constraints only identity, no control on what happens to information during execution.
- No separation of User identity and execution instance.
- Trojan Horses exploit access privileges of calling subjects identity.

MAC - Behavior !!

How MAC fix the DAC weakness (1/2)

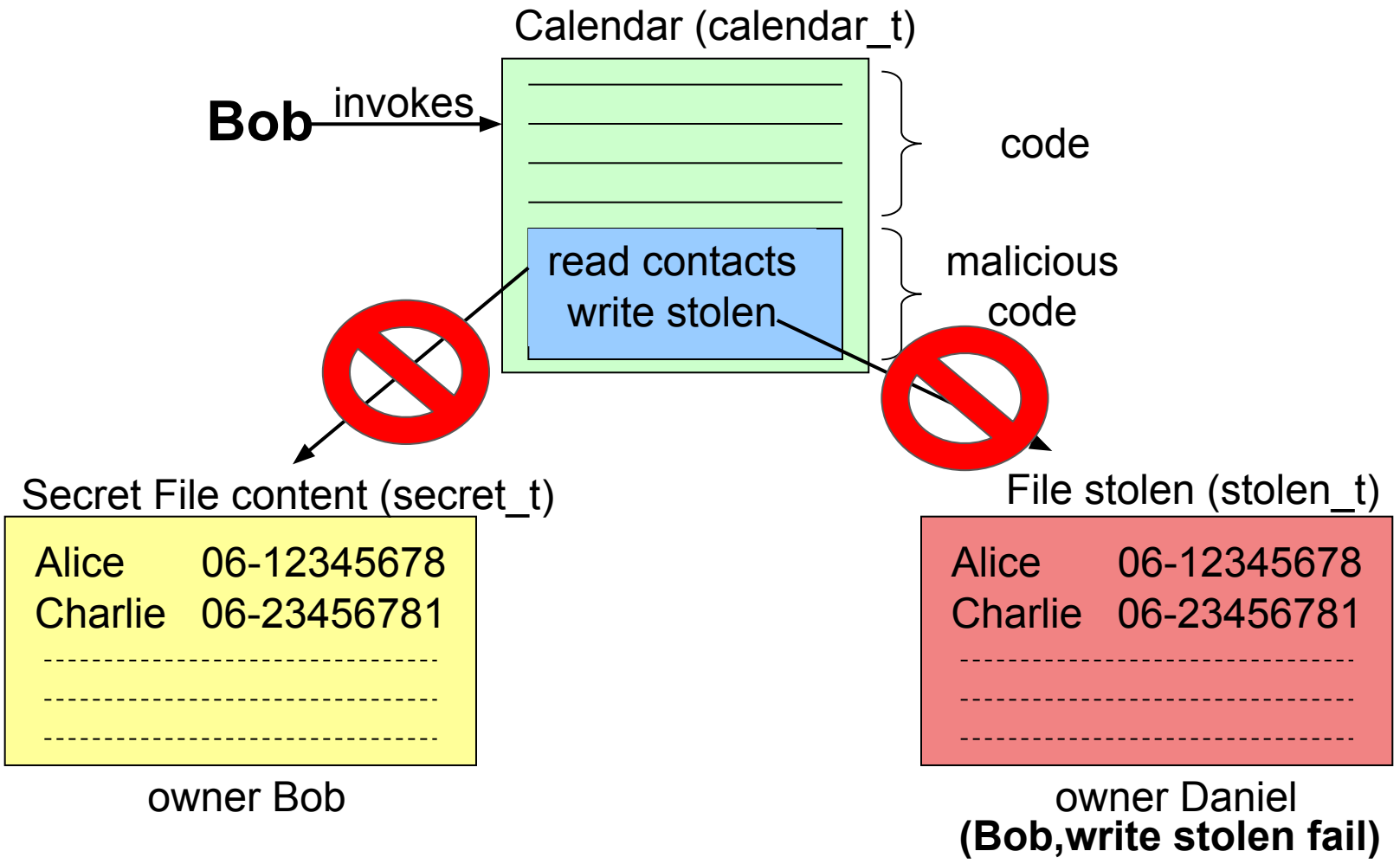
- Policies control access based on mandated regulations determined by a central authority.

User	Application Process Label
Bob	calendar_t

File name	Object Label
Secret file	secret_t
File stolen	stolen_t

Central Authority Rule		
Subject Label	Object Label	Permission
calendar_t	secret_t	No read
calendar_t	stolen_t	Read, No write

How MAC fix the DAC weakness (2/2)



Different MAC Mechanisms

Apparmor

- Path-based system : filesystem no need to support extended attribute
- Per-program profile : describe what program can do.
- **Concept of Different Subject Domain :** If you want a different Subject Domain, you should create a hard link & rename the program & create a new profile for it.

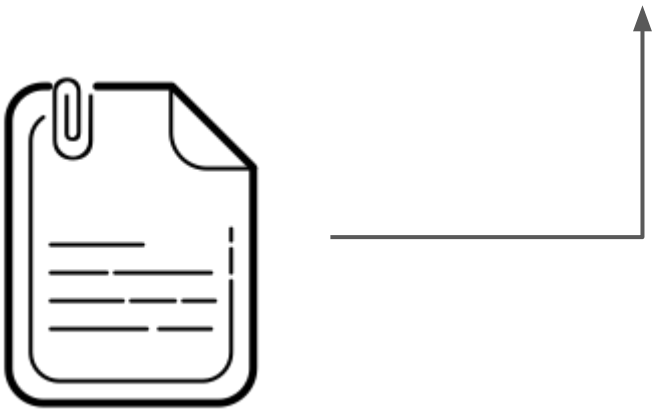
Apparmor Profile

/etc/apparmor.d/usr.bin.test

```
#include <tunables/global>

profile test /usr/lib/test/test_binary {
    #include <abstractions/base>

    # Main libraries and plugins
    /usr/share/TEST/** r,
    /usr/lib/TEST/** rm,

    # Configuration files and logs
    @{HOME}/.config/ r,
    @{HOME}/.config/TEST/** rw,
}
```

Extended Attribute

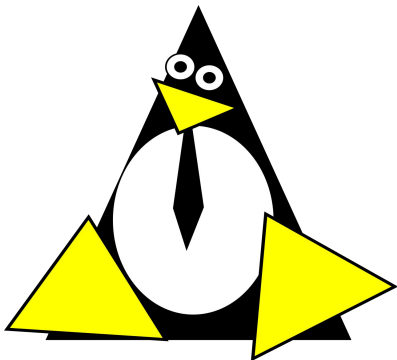
Security.selinux = “Label”



File inode

Smack
(Simplified Mandatory Access Control Kernel)

- Label base : file system should support extended attribute
- Default rules are fixed in kernel
  - Any access requested by a task labelled "*" is denied.
  - A read or execute access requested by a task labelled "^" is permitted.
  - A read or execute access requested on an object labelled "_" is permitted.
  - Any access requested on an object labelled "*" is permitted.
  - Any access requested by a task on an object with the same label is permitted.
  - Any access requested that is explicitly defined in the loaded rule set is permitted.
  - Any other access is denied.



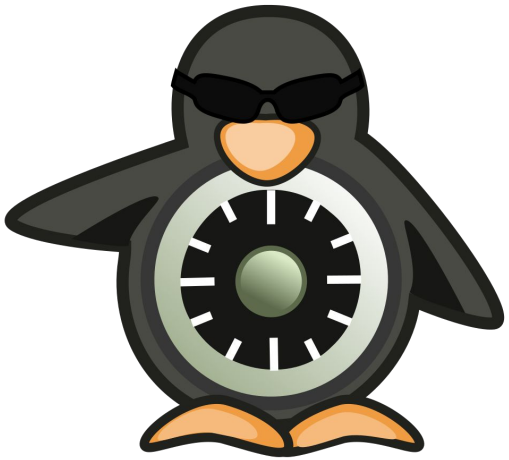
SELinux

- Label base : file system should support extended attribute
- Finer granularity :

Action

Subject ⟶ **Object:Class**

- Different MAC model support :
  Type Enforcement, MCS, MLS, RBAC
- Hard to learn

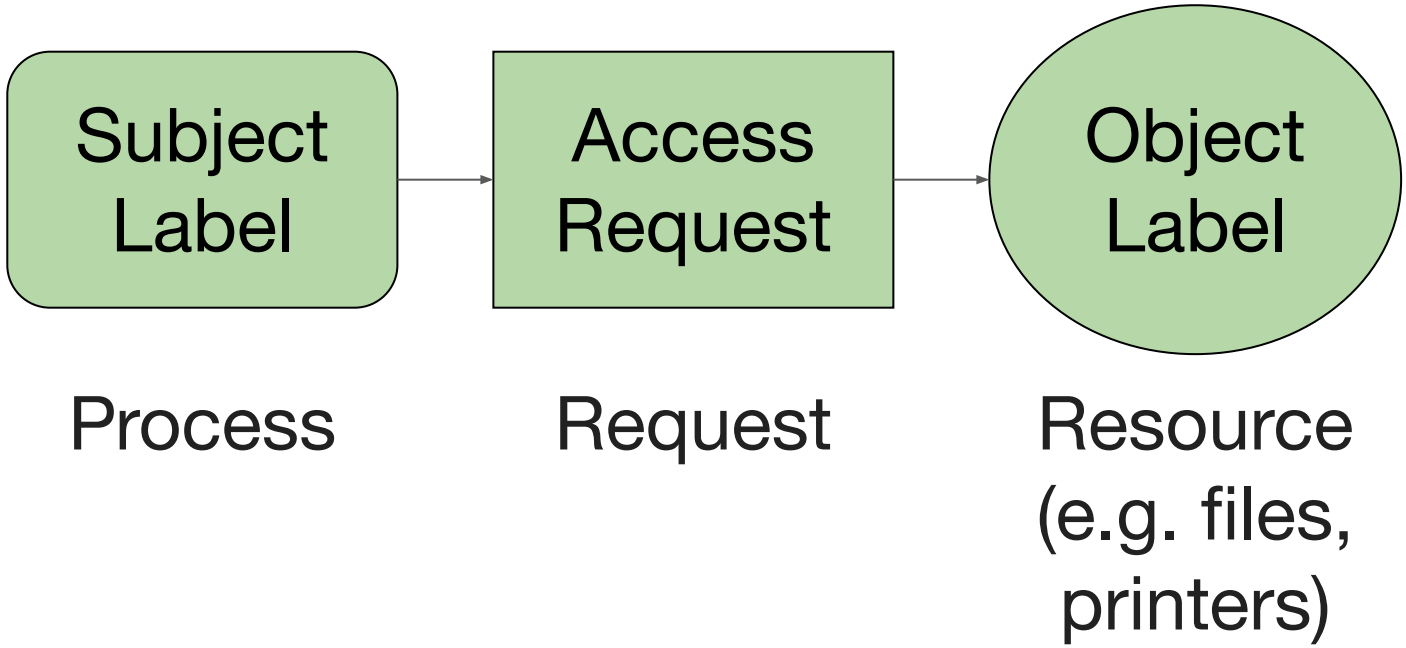


Why Choose SELinux : Comparison

NAME	**SELinux**	Smack	Apparmor
Type	**MAC**	MAC	MAC
Granularity (Hook Point)	**176**	114	62
Extended Attribute	**Yes**	Yes	No
Separation of Policy and Mechanism	**Yes**	Partial	Yes

SELinux Concept (1/2)

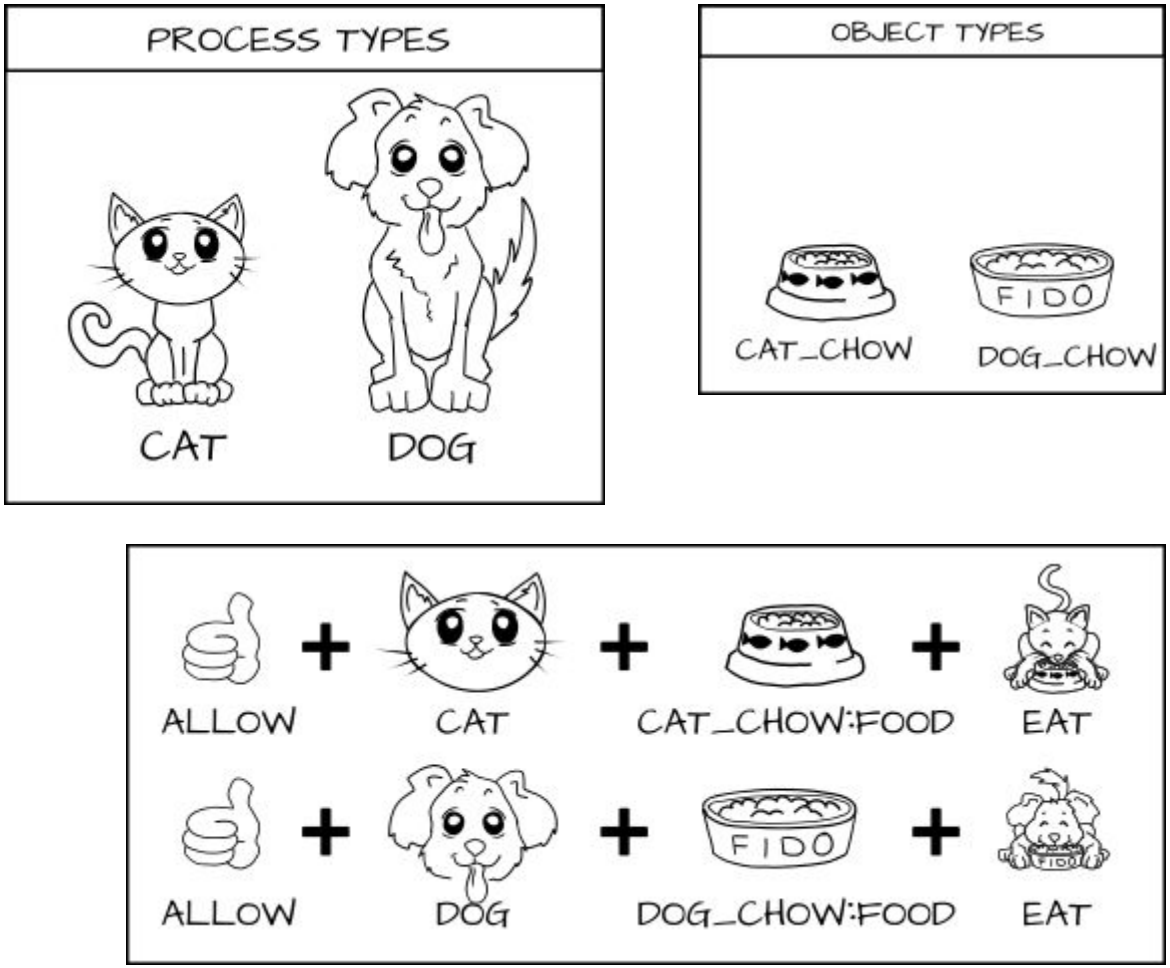
- Mode :
  - Enforce & Permissive & Disable
- Label Format :
  - User:Role:Type:Range



SELinux Concept Outline (2/2)

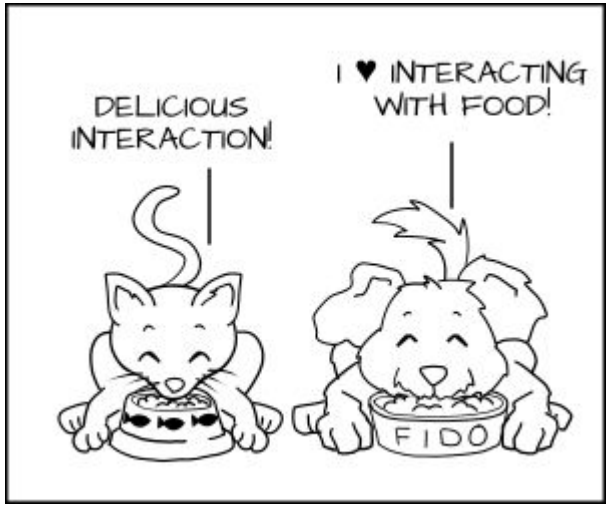
- **Type Enforcement (TE)**: Type Enforcement is the primary mechanism of access control used in the targeted policy
- **Multi-Category Security(MCS)**: An extension of Multi-Level Security.
- **Multi-Level Security (MLS)**: Not commonly used and often hidden in the default targeted policy.

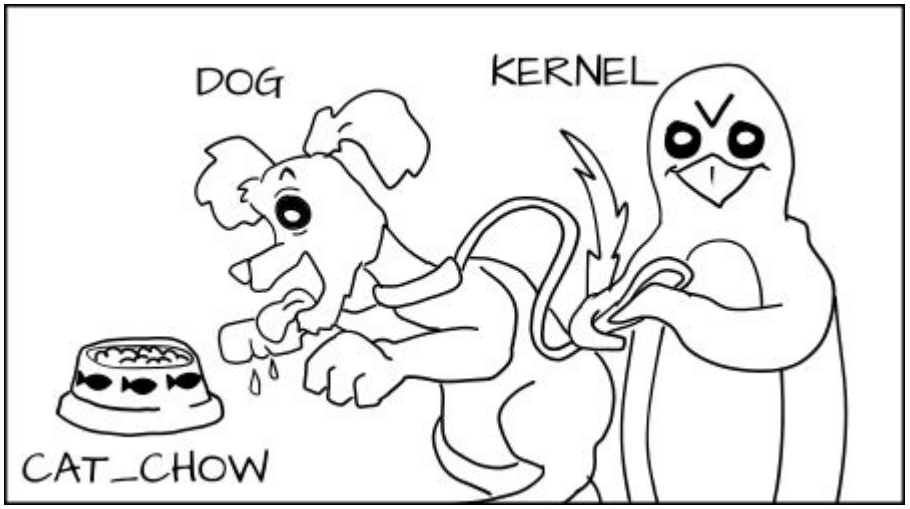
Type enforcement (1/2)

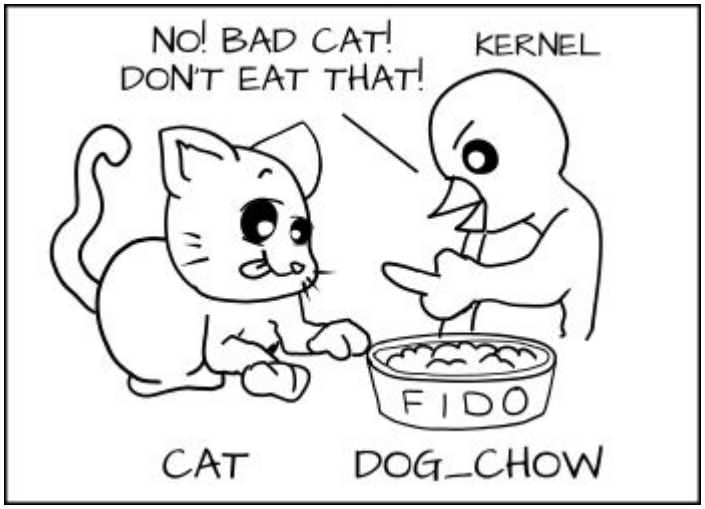


Reference : https://opensource.com/business/13/11/selinux-policy-guide

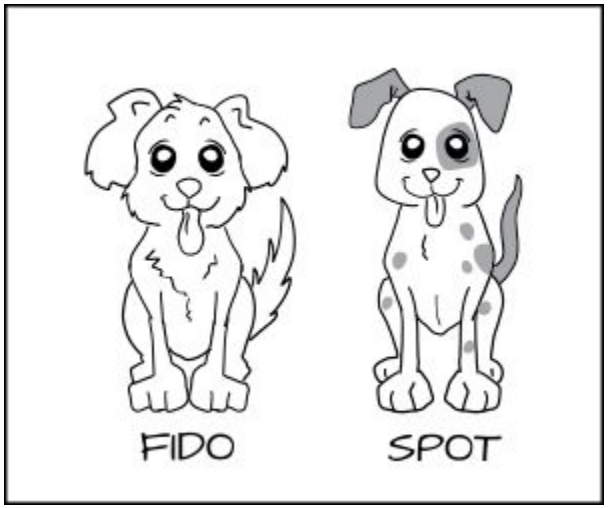
Type enforcement (2/2)

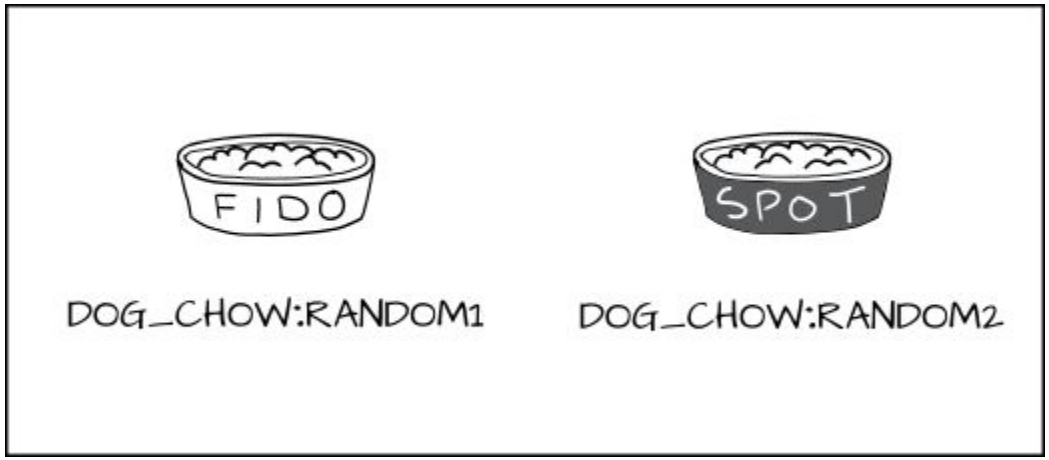




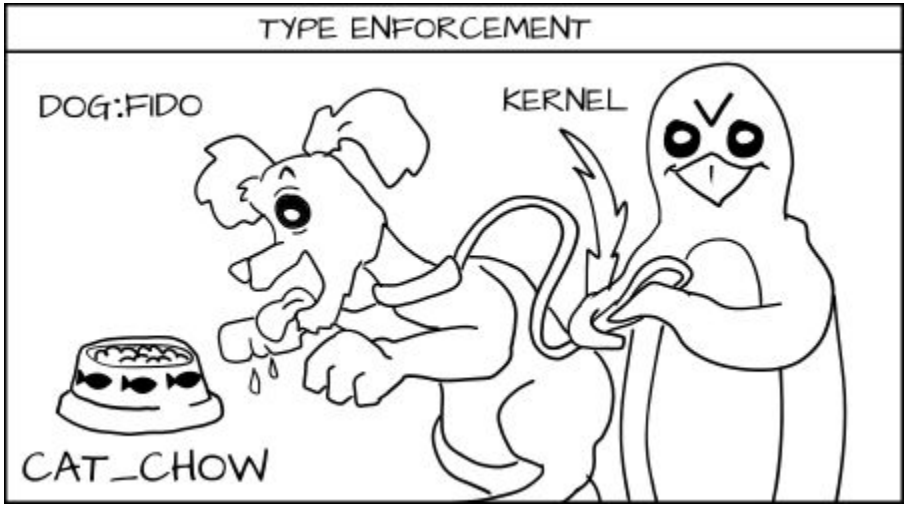


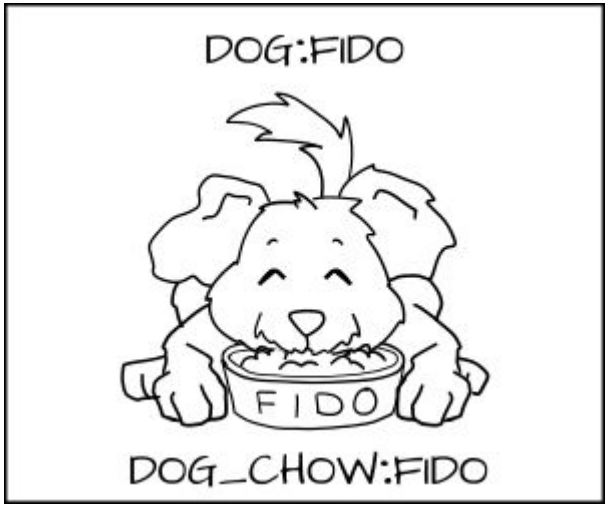
MCS (1/2)

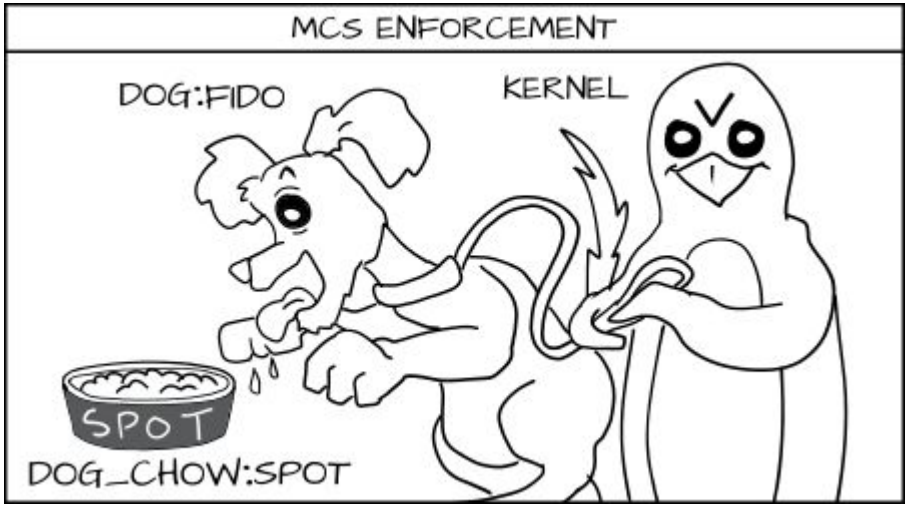




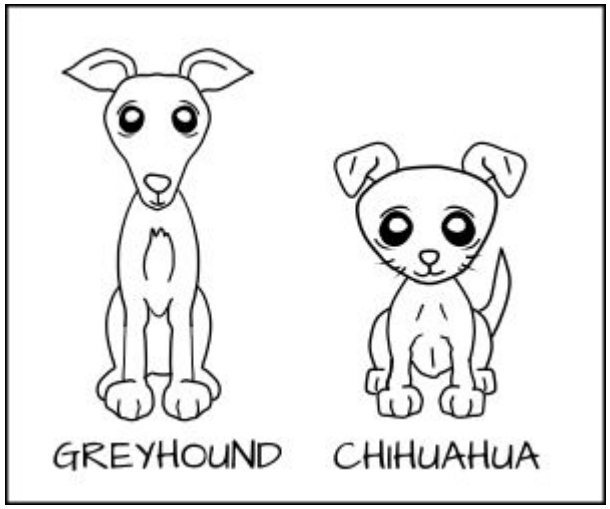
MCS (2/2)


TYPE ENFORCEMENT
DOG:FIDO
KERNEL
CAT_CHOW

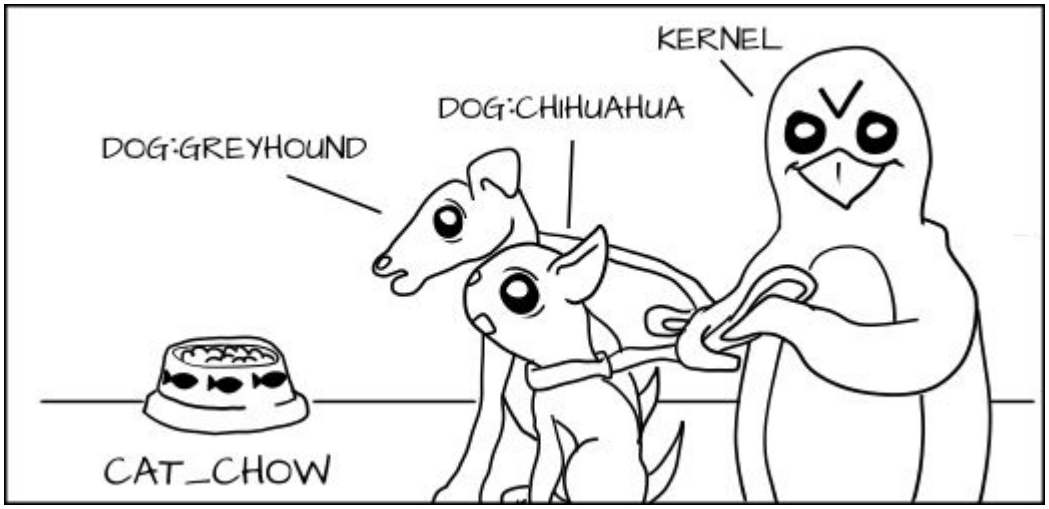

DOG:FIDO
FIDO
DOG_CHOW:FIDO


MCS ENFORCEMENT
DOG:FIDO
KERNEL
SPOT
DOG_CHOW:SPOT

MLS (1/2)


GREYHOUND
CHIHUAHUA


DOG_CHOW:
GREYHOUND
DOG_CHOW:
CHIHUAHUA


KERNEL
DOG:CHIHUAHUA
DOG:GREYHOUND
CAT_CHOW

MLS (2/2)


DOG:GREYHOUND
DOG:CHIHUAHUA
DOG_CHOW:
FIDO
DOG_CHOW:
GREYHOUND
DOG_CHOW:CHIHUAHUA
DOG_CHOW:CHIHUAHUA


THIS IS A BIT
TOO BEEFY
FOR YOU...
KERNEL
DOG:CHIHUAHUA
DOG_CHOW:GREYHOUND

How to Use SELinux Management Tool

Enable SELinux First !

SELinux Management : Get Selinux Context (Label)

- ls -Z (get file selinux context)
- ps Z (get process selinux context)
- seinfo -t : lists all contexts currently in use on your system

```
bighead@ubuntu:/bin$ ls -Z | head
   system_u:object_r:shell_exec_t:s0 bash
          system_u:object_r:bin_t:s0 bunzip2
          system_u:object_r:bin_t:s0 busybox
          system_u:object_r:bin_t:s0 bzcat
          system_u:object_r:bin_t:s0 bzcmp
```

```
bighead@ubuntu:~$ seinfo -t

Types: 1041
   bluetooth_conf_t
   etc_runtime_t
   audisp_var_run_t
   auditd_var_run_t
   ipsecnat_port_t
   semanage_tmp_t
   apmd_var_run_t
   apt_var_lib_t
```

```
bighead@ubuntu:/bin$ ps Z
LABEL                                PID TTY      STAT   TIME COMMAND
system_u:system_r:kernel_t:s0       1177 tty1     S      0:00 -bash
system_u:system_r:kernel_t:s0       1298 tty1     R+     0:00 ps Z
```

SELinux Management :
2 Step Used to Relabel File Type Using Setfiles

- **File_contexts** : used by the file labeling utilities.
- semanage fcontext --add --type httpd_sys_content_t "/var/www(/.*)?"
  - First write the new context to the /etc/selinux/targeted/contexts/files/file_contexts.local file.

- setfiles **file_contexts /var/www**
  - Next, we will run the setfiles command. This will relabel the file or directory with what's been recorded in the previous step

SELinux Management :
Command to Change File Label & Check Policy

- chcon --type bin_t test.c
  - change the context of the file.

```
bighead@ubuntu:~/Downloads$ ls -Z
system_u:object_r:user_home_dir_t:s0 test.c
bighead@ubuntu:~/Downloads$ chcon -t bin_t test.c
bighead@ubuntu:~/Downloads$ ls -Z
system_u:object_r:bin_t:s0 test.c
```

- runcon -t kernel_t /bin/bash
- sesearch --allow --source kernel_t --target proc_t
  - check the type of access allowed for ourselves

```
allow kernel_t proc_t : dir { ioctl read getattr lock search open }
```

SELinux Management : Boolean

- List Boolean :
  - getsebool -a
- Set Boolean :
  - setsebool BooleanName (1 or 0)

```
bighead@ubuntu:~/Downloads$ getsebool -a
allow_execheap --> on
allow_execmem --> on
allow_execmod --> on
allow_execstack --> on
allow_mount_anyfile --> on
allow_polyinstantiation --> off
allow_ptrace --> off
allow_ssh_keysign --> off
allow_user_mysql_connect --> off
allow_user_postgresql_connect --> off
allow_write_xshm --> off
allow_ypbind --> off
cron_can_relabel --> off
fcron_crond --> off
global_ssp --> off
init_upstart --> on
mail_read_content --> off
nfs_export_all_ro --> off
nfs_export_all_rw --> off
secure_mode --> off
secure_mode_insmod --> off
secure_mode_policyload --> off
ssh_sysadm_login --> off
use_lpd_server --> off
use_nfs_home_dirs --> off
use_samba_home_dirs --> off
user_direct_mouse --> off
user_dmesg --> off
user_ping --> off
user_rw_noexattrfile --> off
user_tcp_server --> off
user_ttyfile_stat --> off
xdm_sysadm_login --> off
xserver_object_manager --> off
```

Troubleshoot : Audit Message (1/2)

- avc : denied { relabelto } for pid=1382 comm=”chcon” name=”test.c” dev=”sda1” ino=418253 scontext=system_u:system_r:kernel_t:s0 tcontext=system_u:object_r:unconfined_t:s0 tclass=file

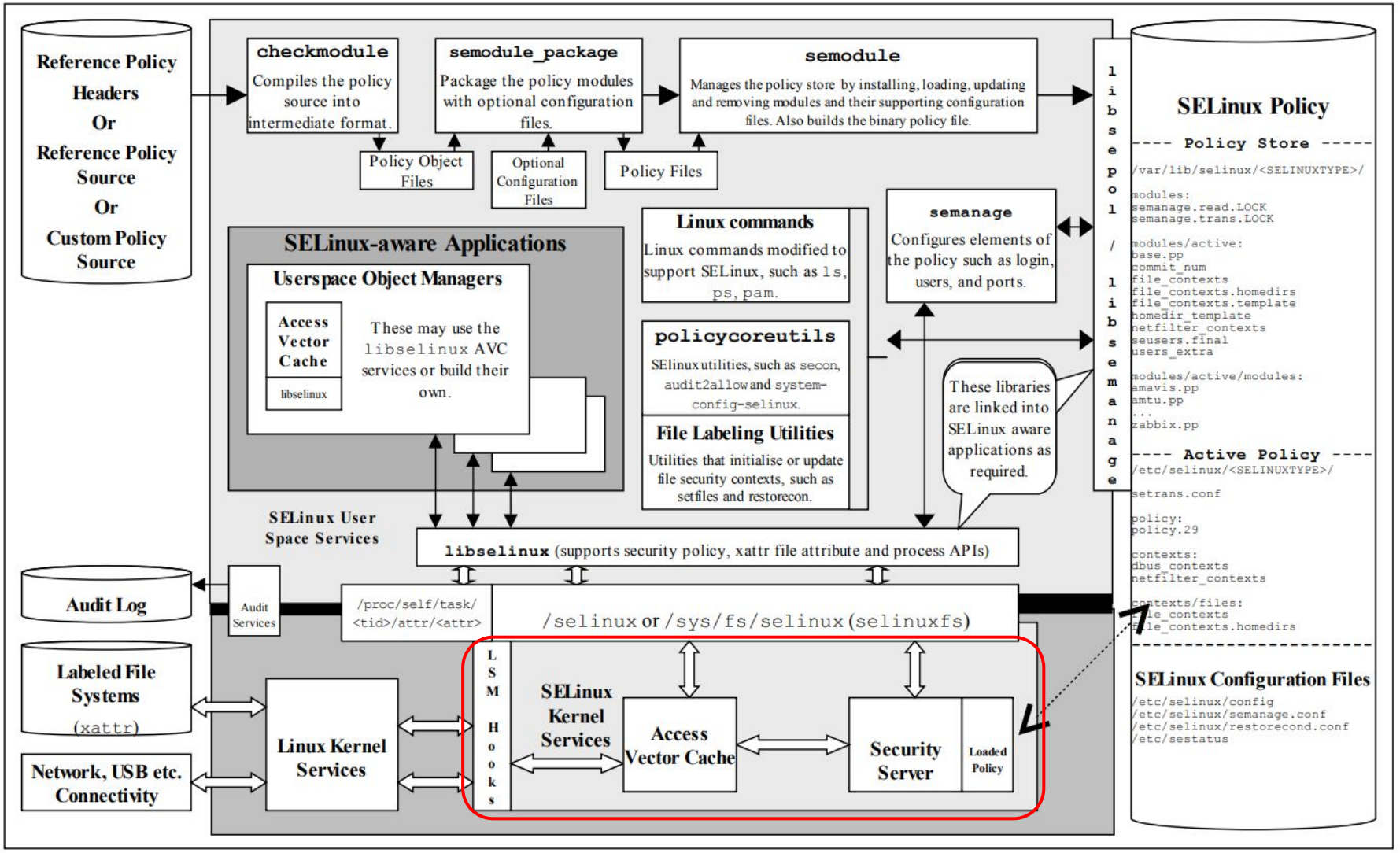
- Dmesg | grep avc | audit2allow -M test
  - Generate test.pp, use semodule -i test.pp to install policy module.

Troubleshoot : Audit Message (2/2)

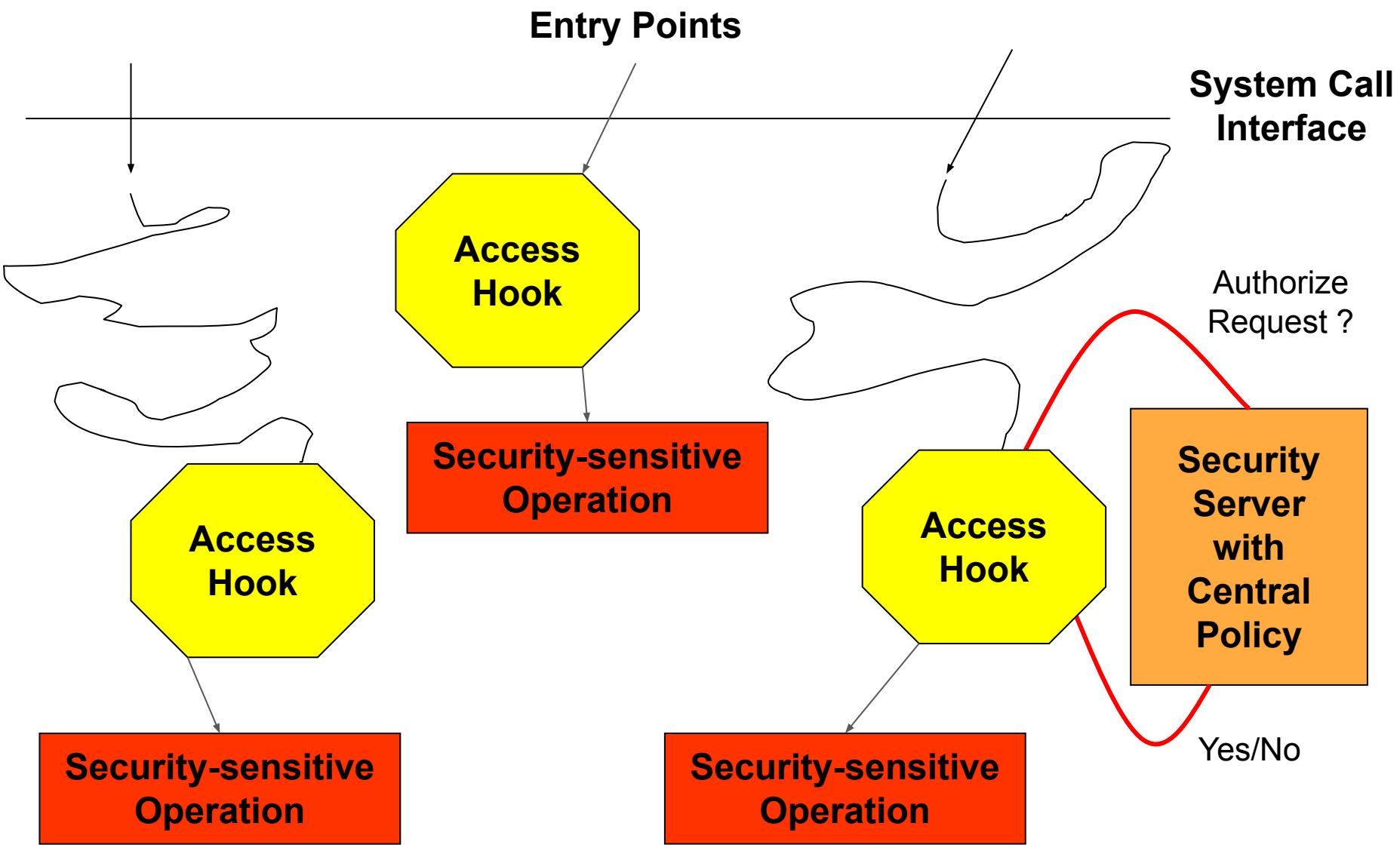
```
module test 1.0;

require {
        type unlabeled_t;
        type unconfined_t;
        type kernel_t;
        class file { relabelfrom relabelto };
        class filesystem associate;
}

#============= kernel_t ==============
allow kernel_t unconfined_t:file { relabelfrom relabelto };

#============= unlabeled_t ==============
allow unlabeled_t self:filesystem associate;
```

User to Developer : What Change ?

SELinux Architecture - LSM Hook

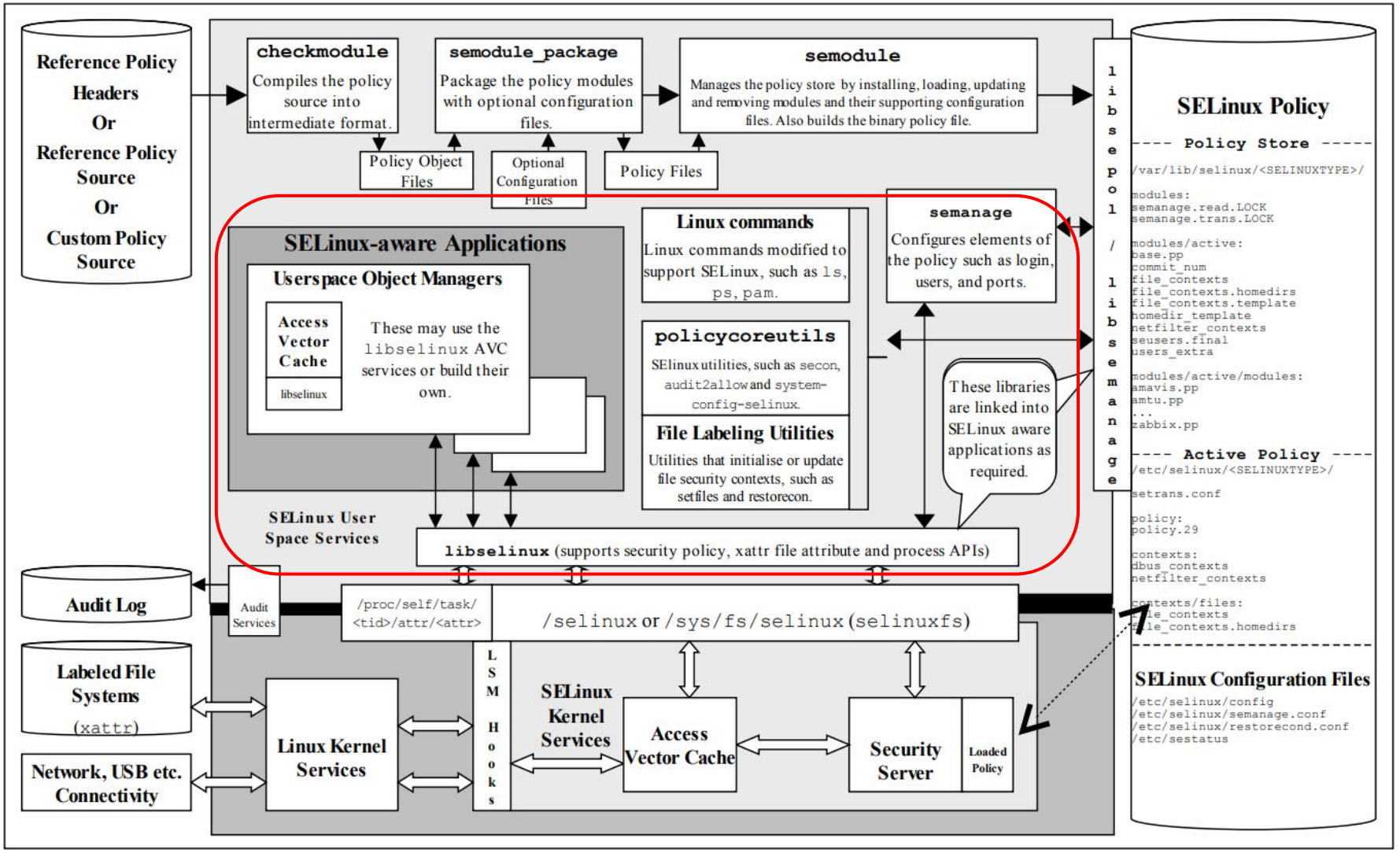


LSM Hook and SELinux Security Server

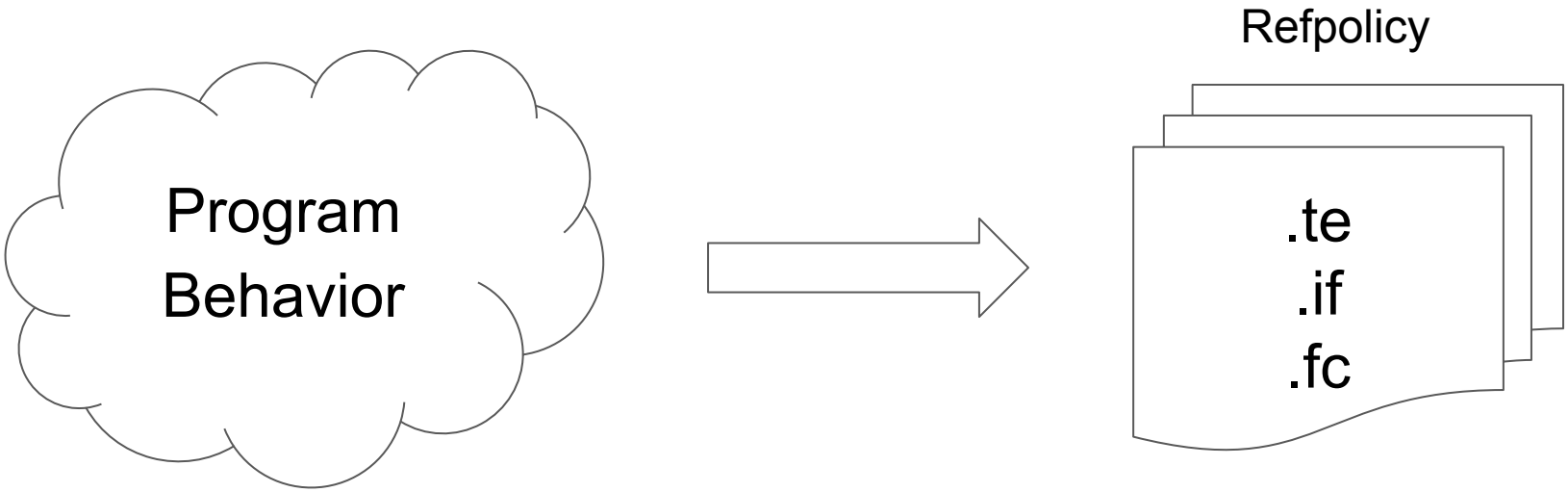


Reference : http://web.eecs.umich.edu/~aprakash/security/handouts/AccessModel_040112_v2.ppt

SELinux Architecture - SELinux-aware Application



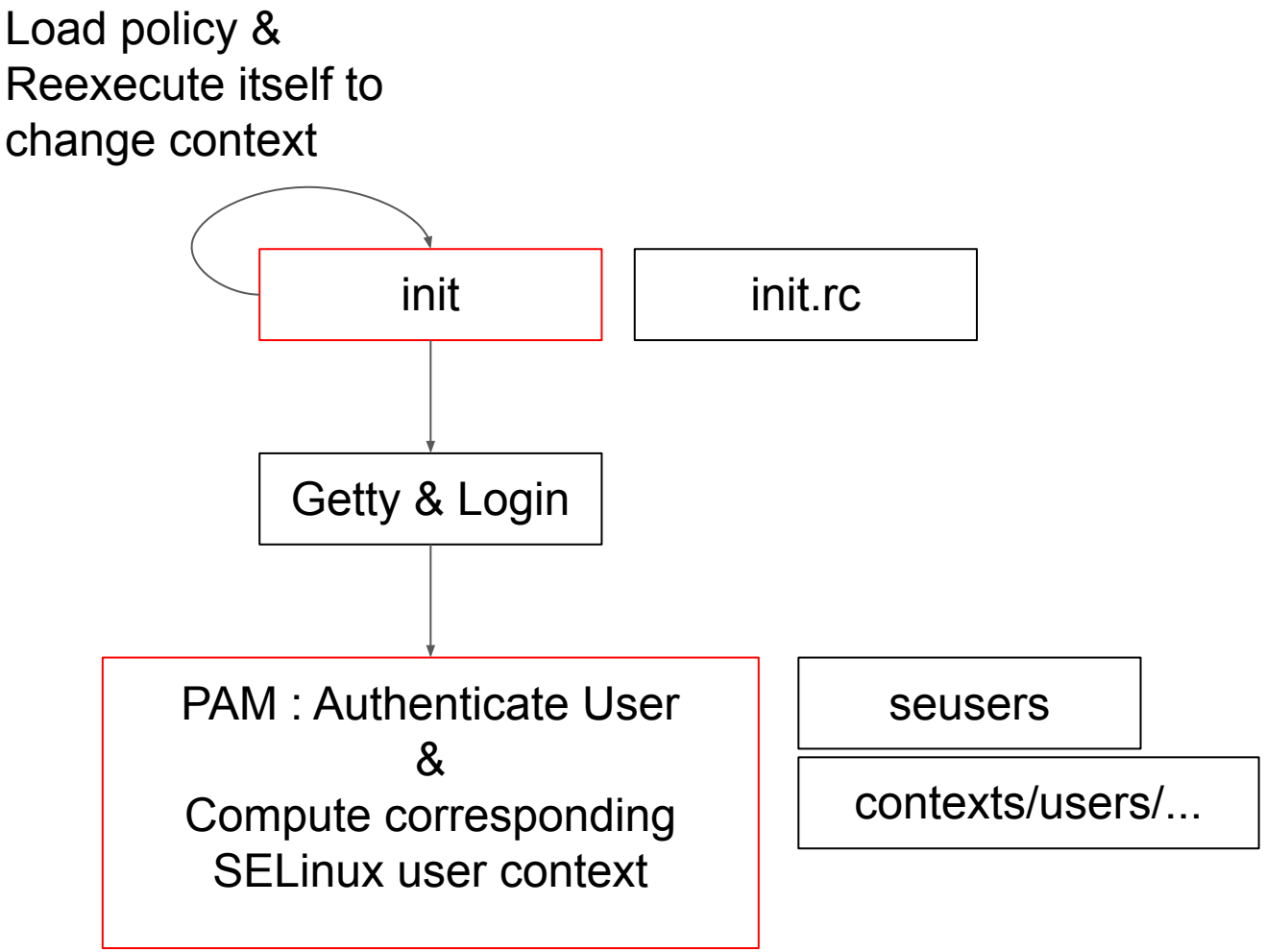
What is the SELinux-aware Package



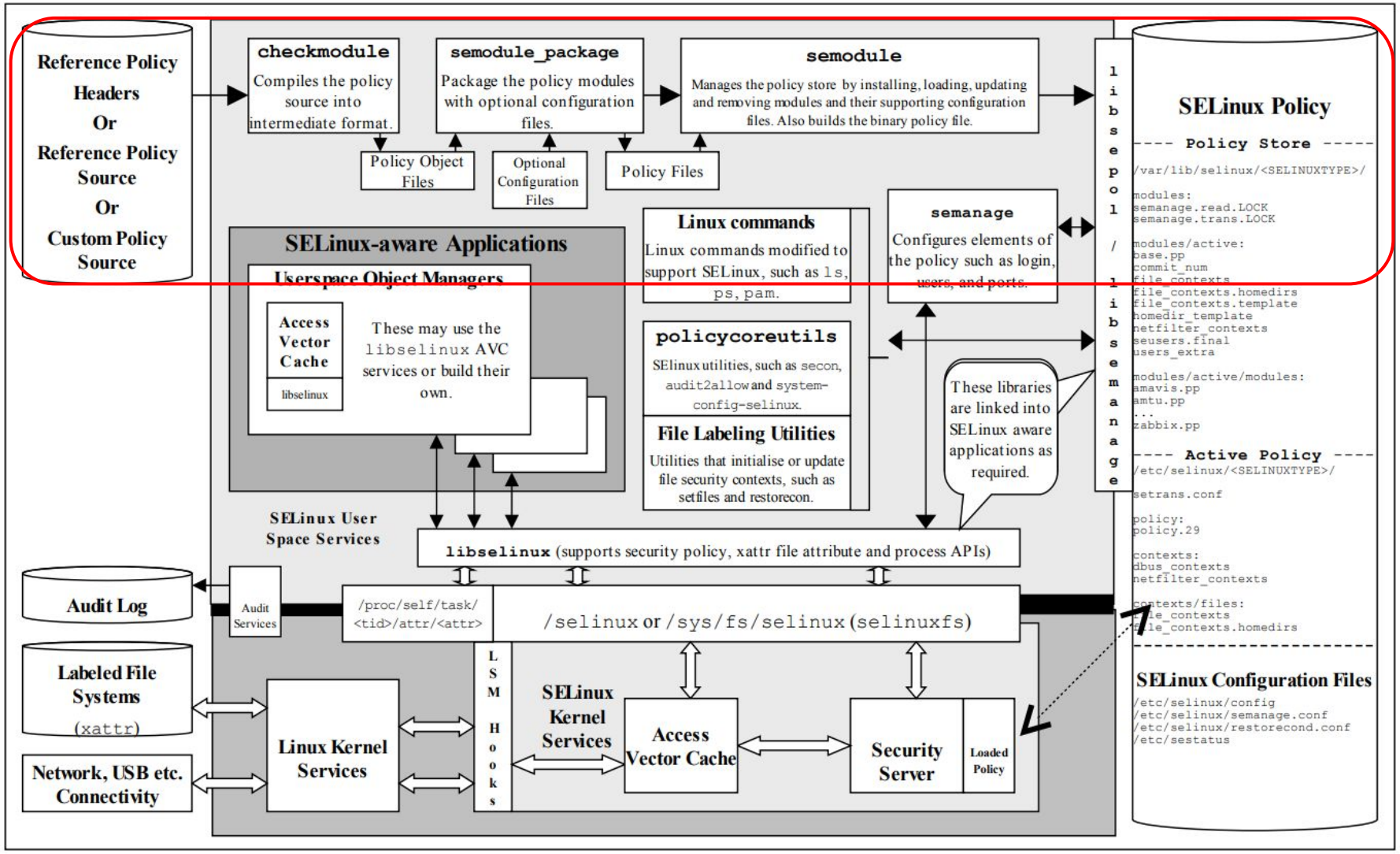
SELinux-aware Level

1. Unaware (e.q. rm)
2. Aware, but not necessary (e.q. ls, ps)
3. Access Securityfs without checking special class (e.q. getenforce)
4. In addition to access Securityfs, check the permission in special class below (e.q. systemd, init, setenforce)
    a. File, Socket, Database, Filesystem class
        i. Relabelto
        ii. Relabelfrom
    b. Process class
        i. Dyntransition
        ii. Setexec
        iii. Setfscreate
        iv. Setkeycreate
        v. Setsockcreate
    c. Security class
    d. Kernel service class

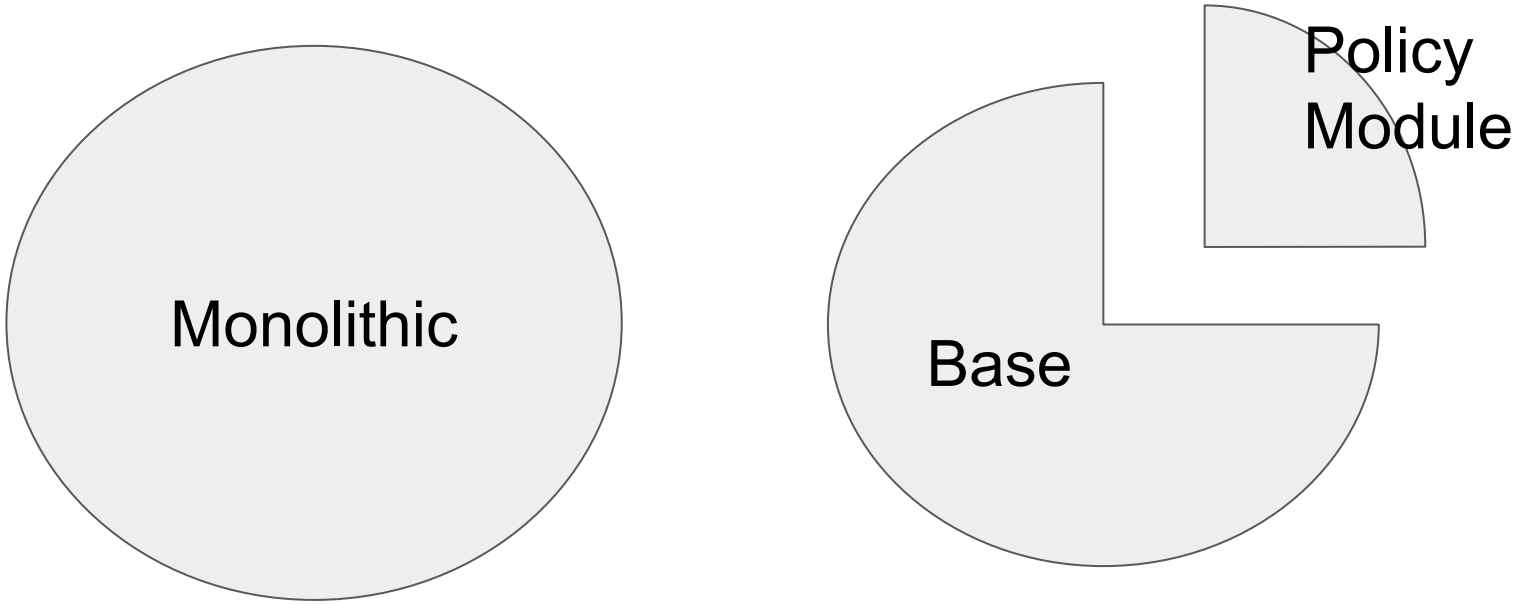
Example : Linux Initialization



SELinux Architecture - Build Policy

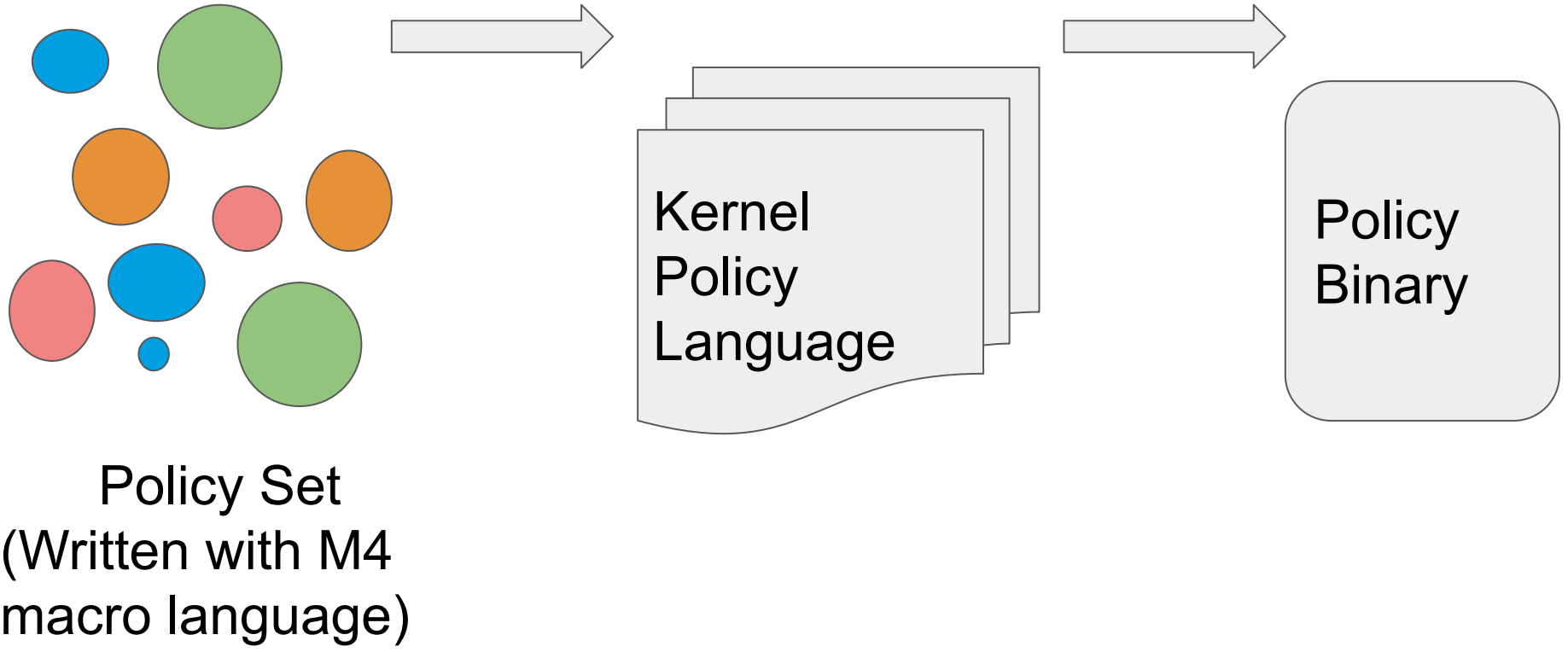


How to Write Policy by Yourself

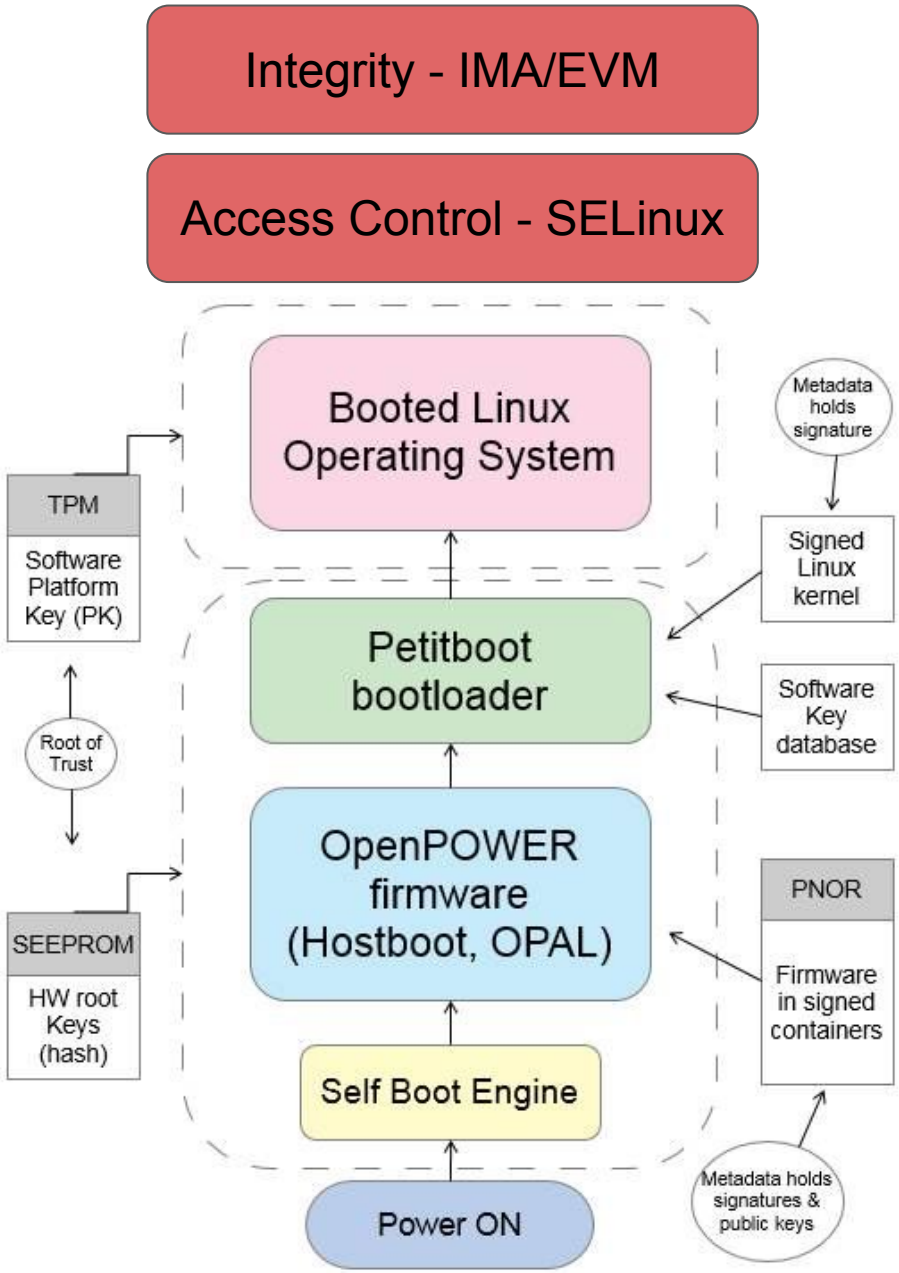


- All build by 3 file :
  - .te : like .c file
  - .if : like .h file
  - .fc (describe file context)

Policy Build Sequence



Secure Boot



Reference : https://developer.ibm.com/articles/protect-system-firmware-openpower/

Call Our Team

```
[    4.842107] random: crng init done

RaspberryPi3B+ login: pchang9
pchang9@RaspberryPi3B+ ~ $ uname -a
Linux RaspberryPi3B+ 5.1.0 #11 SMP Wed Jun 26 10:10:32 CST 2019 armv7l GNU/Linux
pchang9@RaspberryPi3B+ ~ $ getenforce
Enforcing
pchang9@RaspberryPi3B+ ~ $ id
uid=1000(pchang9) gid=10(wheel) groups=10(wheel) context=user_u:user_r:user_t
pchang9@RaspberryPi3B+ ~ $ ls -Z
user_u:object_r:user_home_t      hello-world-app
user_u:object_r:mcstst_home_t    lab_for_mcstst
pchang9@RaspberryPi3B+ ~ $ cd lab_for_mcstst/
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ ls -Z
user_u:object_r:mcstst_home_t    vi
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ getsebool -a | grep mcstst
allow_mcstst_create_del --> on
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ ls
vi
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ touch test
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ ls
test  vi
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ rm test
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ ls
vi
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ ./vi test
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ ls
test  vi
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $ cat test
123
pchang9@RaspberryPi3B+ ~/lab_for_mcstst $
```

```
1. Check SELinux status
        getenforce
2. Check directory & file content
        ls -Z
3. Check boolean status for permission (ON)
        getsebool -a | grep mcstst
4. Try create & delete file
        touch test
        rm test
        ./vi test
5. Turn boolean off
        sudo setsebool allow_mcstst_create_del 0
6. User & Root can't create and delete file, but special labeled vi can.

7. User & Root can't send sigkill signal to special labeled process.
```

Q&A **X** SELinux Demo

SELinux enforce mode

SELinux permissive mode

SELinux enforce mode
on Raspberry Pi 3 Model B+

Busybox (Embedded System)

限定指定資料夾
僅能指定程序存取

保護特定程序
不被任何人kill

Ubuntu