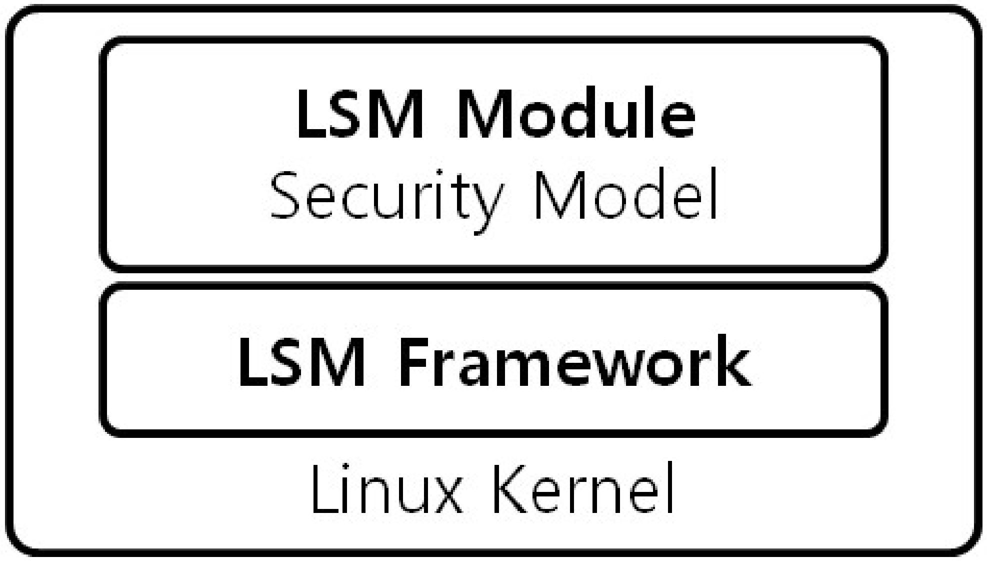
LG ELECTRONICS DEVELOPMENT VIETNAM
DA NANG BRANCH

SMACK

AGENDA

LGEDV | 2024. 07. 30

SMACK Background Knowledge

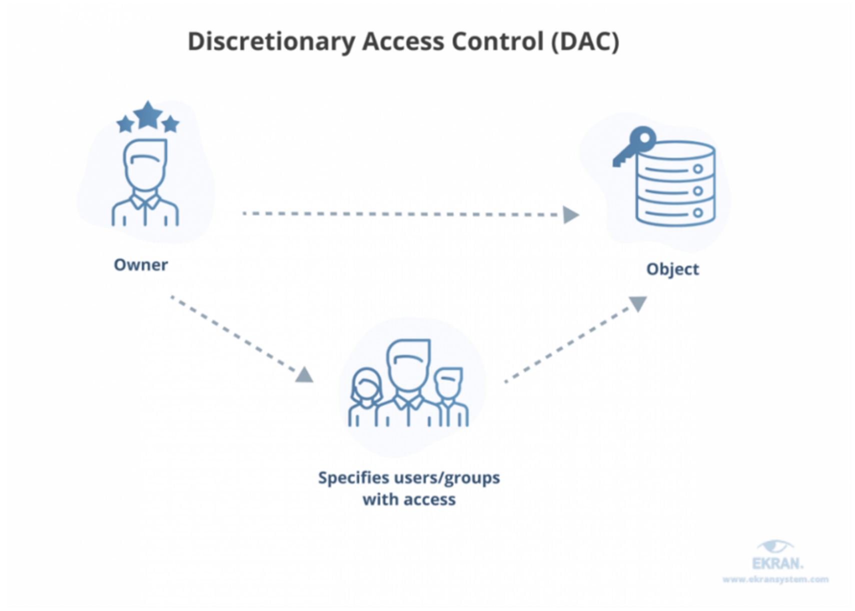


LSM (Linux Security Module)

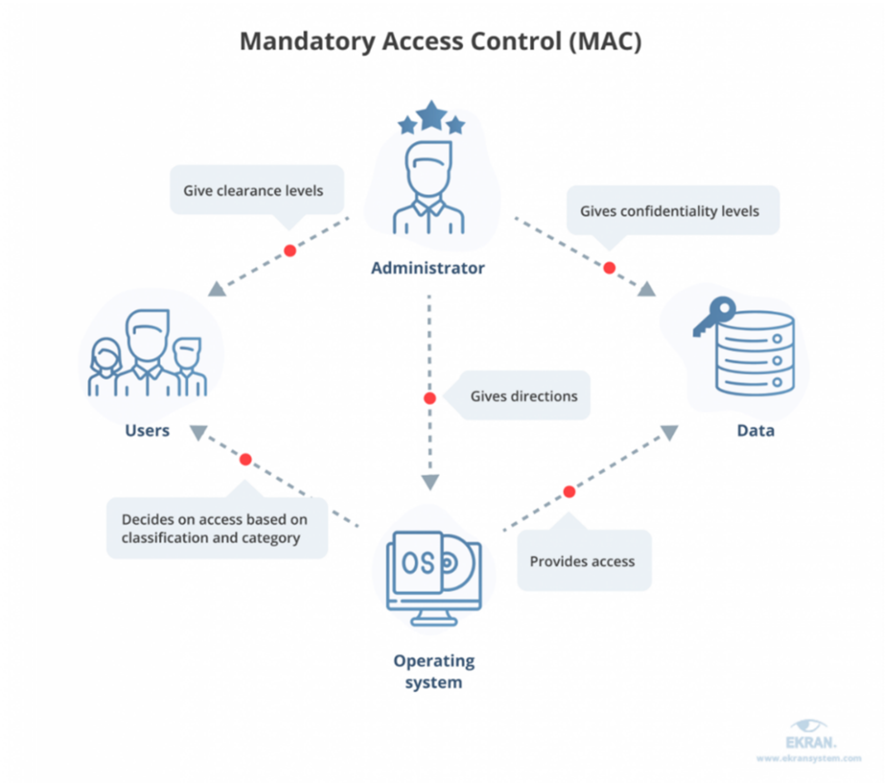
: As a framework, the Linux kernel supports a variety of computer security models while avoiding a single security implementation.

SMACK Background Knowledge

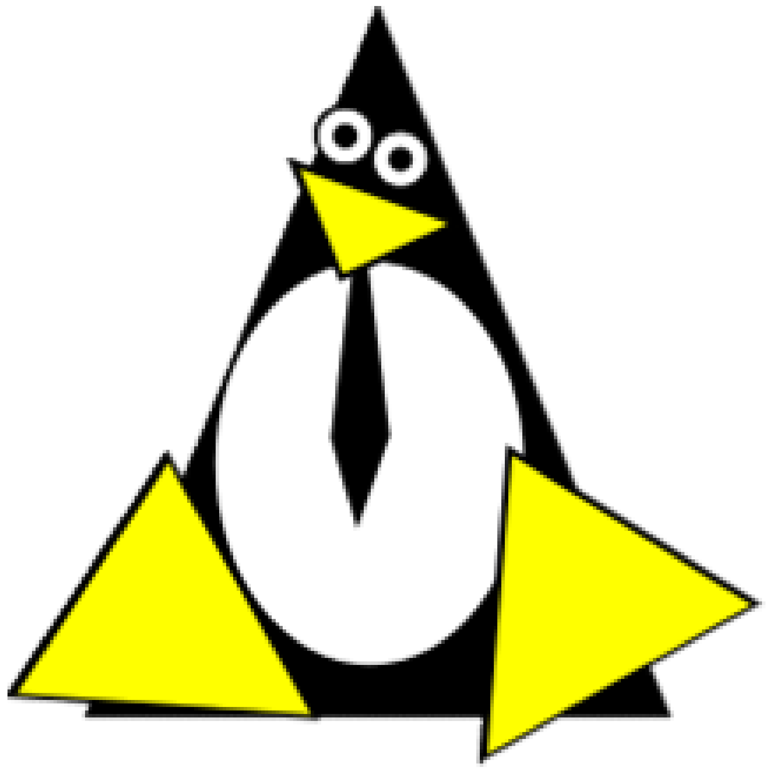
SMACK Background Knowledge



Discretionary Access Control (DAC)
Owner
Object
Specifies users/groups with access
EKRAN.
www.ekransystem.com



Overview



SMACK (Simplified Mandatory Access Control Kernel)

- It is one of the LSMs implementing the same label-based MAC technology as SELinux.
- Compared to SELinux, the execution time is shorter, the rule setting is simple, and the system load is less.
- Grant access between subject and object.
- Mainly used in embedded systems.

Terminology Description

1. Subject : Means process for accessing objects.

2. Object : Means resources accessed by the subject.

3. Label : Smack identifies access rights based on labels. Multiple files and processes

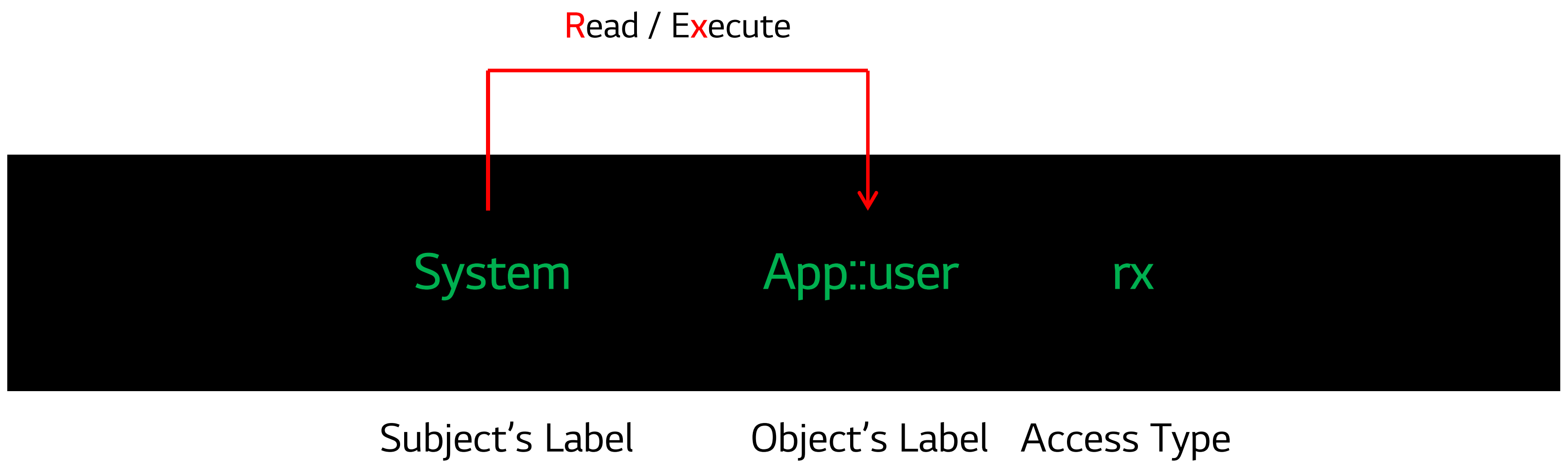
can have the same label.

- Access Label : Applies when the resource becomes an access object.
- Execute Label : Applies when the resource becomes the access principal.

4. Policy : Access rules defined on a label basis.

5. Access : Permissions that the Subject has on the Object. (r, w, x, a, l, t, b)

Labels and Rules – SMACK Rules' Format



```
System    App::user    rx
```

How to apply Rule : smackctl apply

How to remove Rule : smackctl clear

Labels and Rules – How to Apply

1. Apply Label

   - chsmack –a ${ACCESS_LABEL} ${TARGET_RESOURCE}
   - chsmack –e ${EXECUTE_LABEL} ${TARGET_PROCESS}

```
# chsmack –a App::test1 test
# chsmack –e App::test2 test
# chsmack test
./test        access="App::test1"    execute="App::test2"
```

2. Remove Label

   - chsmack –A ${ACCESS_LABEL} ${TARGET_RESOURCE}
   - chsmack –E ${EXECUTE_LABEL} ${TARGET_PROCESS}

Labels and Rules – How to Apply

3. Recursive Label

- chsmack –ra ${ACCESS_LABEL} ${TARGET_RESOURCE}
- chsmack –re ${EXECUTE_LABEL} ${TARGET_PROCESS}

```
# chsmack –ra App::test1 test
# chsmack –re App::test2 test
# chsmack test
./test          access="App::test1"     execute="App::test2"

# cd test
# chsmack
./1             access="App::test1"     execute="App::test2"
./2             access="App::test1"     execute="App::test2"

# touch 3
# chsmack
(1 and 2 is omitted)
./3             access="_"
```

Labels and Rules – Transmute Option

Transmute Option : Files in the directory of the Transmute option inherit the label of that directory. t must be set in the Access rule.

- chsmack –t ${TARGET_DIRECTORY}

```
# chsmack –a App::test1 test
# chsmack –t test
# chsmack test
./test          access="App::test1"     transmute="TRUE"

# cd test
# chsmack
(1 and 2 is omitted)
./3             access="_"

# echo "_ App::test1 rwxt" | smackload
# mkdir 4
(1, 2 and 3 is omitted)
./4             access="App::test1"     transmute="TRUE"
```

Labels and Rules – Predefined Rules

Step-by-step permission check

1. If the process is labeled "*", no files are accessible.
2. If the process is labeled "^", you have permissions to read and execute.

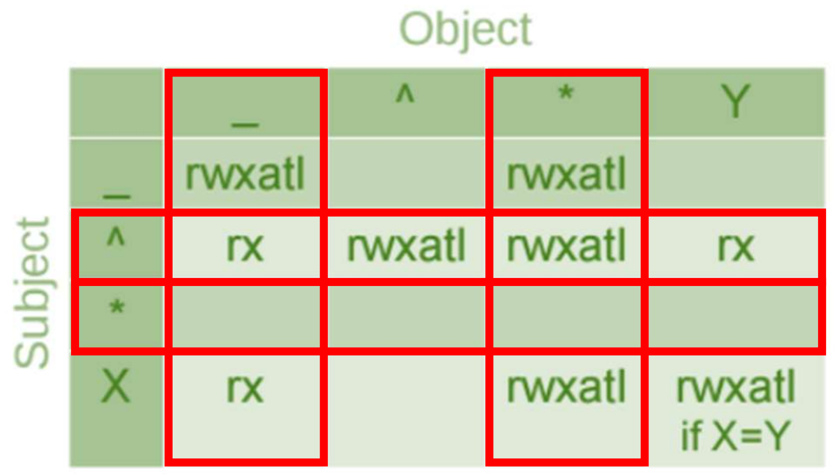
3. If the resource is labeled "_", you have permissions to read and execute.

4. If the resource is labeled "*", it has all permissions regardless of process.

5. If the process and resource have the same label except when the process has a "*" label, it has all permissions.

6. In the case of a clearly defined process in the loaded rule, it has permissions set by the user in the rule.

7. Any other approach is denied.



Object

Subject

	_	^	*	Y
_	rwxatl		rwxatl	
^	rx	rwxatl	rwxatl	rx
*				
X	rx		rwxatl	rwxatl if X=Y

Some implicit rules

Enforce Mode VS Permissive Mode VS Bring-up Mode

Enforce Mode : Default mode of SMACK, output 'Permission Denied' Error to prevent unauthorized labels from being accessed, logged in Denial Log.

```
# chsmack
./test          access="App::test1"

# cat /sys/fs/smackfs/load2 | grep App::test1
_               App::test1                  rb

# cat test
Hello,

# echo "test" >> World!
-sh: test: Permission denied

# cat /data/audit/audit.log
Type=AVC msg=audit(1663036102.115:38): lsm=SMACK
fn=smack_inode_permission action=denied subject="_" object="App::test1"
requested=w(US)
(The rest has been omitted.)
```

Enforce Mode VS Permissive Mode VS Bring-up Mode

Permissive Mode : Regardless of whether or not the policy is applied, all actions in which the label is included as a process and resource are allowed.

```
# chsmack
./test          access="App::test1"

# cat /sys/fs/smackfs/load2 | grep App::test1
_               App::test1              rb

# cat test
Hello,

# echo "test" >> World!

# cat /data/audit/audit.log
Type=AVC msg=audit(1663036942.596:43): lsm=SMACK
fn=smack_inode_getattr action=granted subject="_" object="App::test1"
requested=r
(The rest has been omitted.)
Type=AVC msg=audit(1663036943.709:46): lsm=SMACK
fn=smack_inode_setattr action=granted subject="_" object="App::test1"
requested=w(US)
(The rest has been omitted.)
```

CONFIG_SECURITY_SMACK_BRINGUP=y

How to change to Permissive Mode

1. Kernel Build
2. echo ${LABEL_NAME} > /sys/fs/smackfs/unconfined

Enforce Mode VS Permissive Mode VS Bring-up Mode

Bring-up Mode : If b permissions are added to the Access rule, both granted and denied log are recorded.

```
# chsmack
./test             access="App::test1"

# cat /sys/fs/smackfs/load2 | grep App::test1
_                  App::test1                      rb

# cat test
Hello,

# echo "test" >> World!
-sh: test: Permission denied

# cat /data/audit/audit.log
Type=AVC msg=audit(1663036942.596:43): lsm=SMACK fn=smack_inode_getattr
action=granted subject="_" object="App::test1" requested=r
(The rest has been omitted.)
Type=AVC msg=audit(1663036943.709:46): lsm=SMACK fn=smack_inode_setattr
action=denied subject="_" object="App::test1" requested=w(US)
(The rest has been omitted.)
```

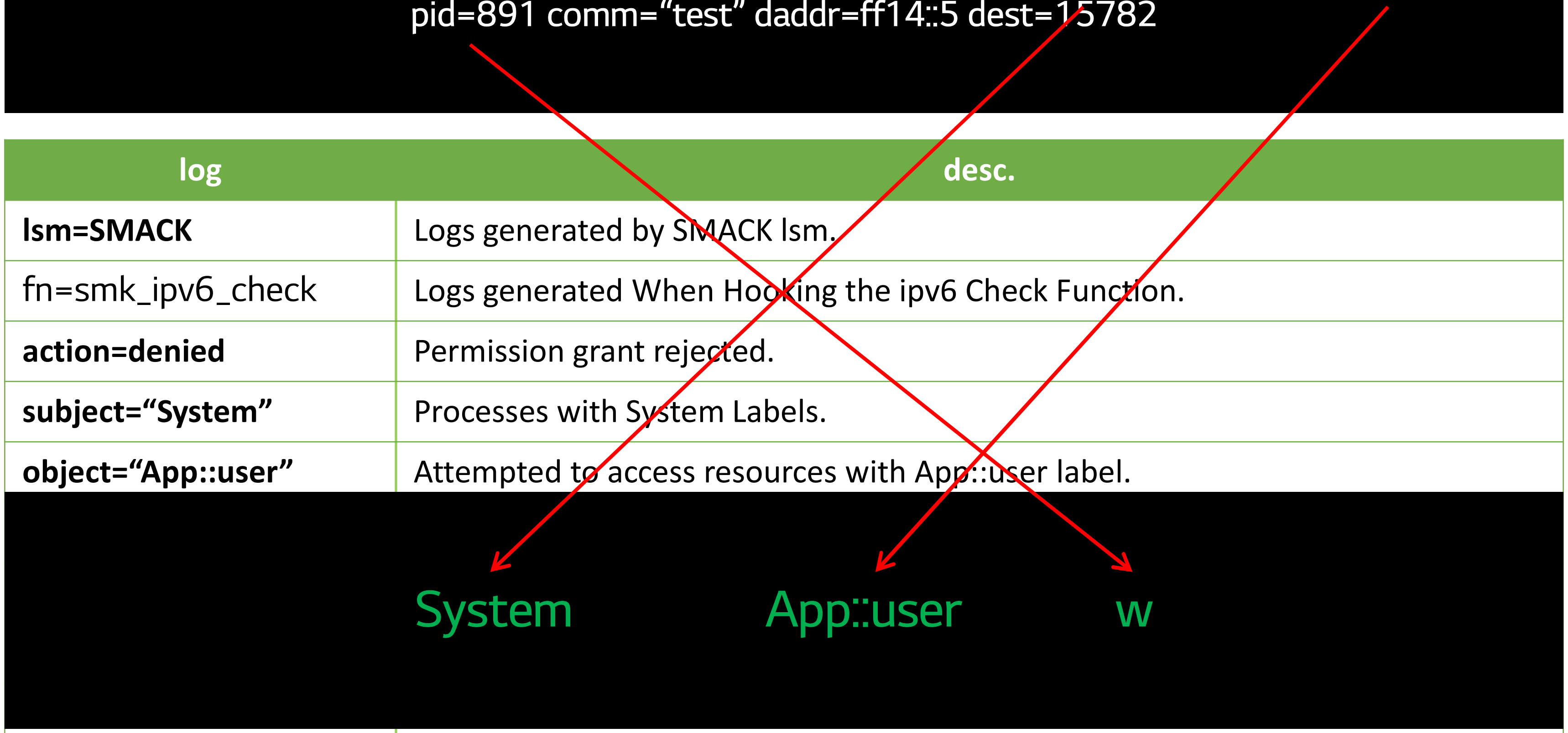
CONFIG_SECURITY_SMACK_BRINGUP=y

Denial Log

How to interpret Denial log

```
lsm=SMACK fn=smk_ipv6_check action=denied subject="System" object="App::user" requested=w
pid=891 comm="test" daddr=ff14::5 dest=15782
```

log	desc.
lsm=SMACK	Logs generated by SMACK lsm.
fn=smk_ipv6_check	Logs generated When Hooking the ipv6 Check Function.
action=denied	Permission grant rejected.
subject="System"	Processes with System Labels.
object="App::user"	Attempted to access resources with App::user label.
dest=15782	Indicates dest port.



Onlycap Mode

Onlycap Mode : Mode to delete **root** permissions for all labels except privileged labels.

The **root** has both of the following functions. (Privileged Process)

- CAP_MAC_ADMIN : Process can modify labels and rules.
- CAP_MAC_OVERRIDE : Process can ignore rules.

Therefore, if onlycap mode is set, labels and rules can't be modified, and ignored.

How to turn off the onlycap mode

1. For the full labels
   - /etc/smack/conf/configure.sh : # echo "Privileged" >> $ONLYCAP_FILE
   - /etc/smack/conf/smack_setup.sh : # echo "Privileged" >> $ONLYCAP_FILE
   - /etc/smack_setup.sh : # echo "Privileged" >> $ONLYCAP_FILE
2. For specific labels : echo "App::user" >> /etc/smack/onlycap

Daemons related to SMACK

1. smack.service : To use smackctl and mount smackfs.
   - load_smack_labels.service : To execute the load_label.sh file that load the labels.
2. auditd.service : To create an audit.log file that logs the entire path.
3. smack-profiler.service : To provide a variety of capabilities for the user. Admin shell, Label snapshot, Rule Generator, Rule syntax checker, Runtime mode change, etc.
4. smack-setup.service : To invoke smack_setup.sh. It is to set the rule and the onlycap mode.
5. smack-test.service : To execute a script that tests the SMACK basic functionality.

THANK YOU!

LGEDV | 2023. 11. 24