SBRACK

Stony Brook   Role-based Access Control Kernel

CSE509 - Project 1

Udit Gupta

CONTENTS

INTRODUCTION

1. This project aims to design, implement and test a role based file access control module in the kernel.
2. The Project implements a basic RBAC model as a kernel level reference monitor that would satisfy regular user demand for file system access control.
3. The monitor uses the Linux Security Module (LSM) APIs/hooks and mediate only few basic inode access in the current version.

ENVIRONMENT

- Ubuntu 14.04
- Linux-kernel-3.14.17 (Vanilla) - Separate Entry in grub along with existing default (ubuntu) kernel
- SBRACK Security Module - Added as the default security module replacing existing security configurations (*via make menuconfig)*

TECHNICAL OVERVIEW

1. The security module has been named **rbslinux (Role Based Secure Linux)** and has been added in the kernel **3.14.17 (vanilla)** as s default security module. All other security options(like selinux) has been disabled in the kernel configuration.

2. The security module uses the **security_operations** structure and register **rbslinux_*** functions for inode access hooks. The current version implement only five of them.

3. All **rbslinux_*** functions read the **configuration file (/etc/rbslinux.conf)** for all the users (user ids greater than 1000) and their roles. Role can be Admin (0) or Non-admin (1). Depending on the role, inode access are permitted.

LSM FRAMEWORK

LSM HOOKS FOR INODE ACCESS

```
static struct security_operations rbslinux_ops = {
    .name           =    "rbslinux",
    .inode_create   =    rbslinux_inode_create,
    .inode_unlink   =    rbslinux_inode_unlink,
    .inode_mkdir    =    rbslinux_inode_mkdir,
    .inode_rmdir    =    rbslinux_inode_rmdir,
    .inode_rename   =    rbslinux_inode_rename
};
```

INSTALLATION STEPS

1. Copy the **rbslinux** directory to the linux kernel security folder (**parallel to selinux directory**).

2. Change the **KConfig** file in the security folder of kernel as per the changes in **KConfig** file of archive. (This is different from **KConfig** inside **rbslinux** directory)

3. Change the **Makefile** in the security folder of the kernel as per the chnages in the **Makefile** of archive. (This is different from **Makefile** inside **rbslinux** directory)

ARCHIVE CONTENTS

- rbslinux
  - rbslinux.c
  - rbslinux.o
  - KConfig
  - Makefile
- test
  - rbslinux.conf
  - README
- user-space-utility
  - rbs_script.sh
  - README
- Documentation
- Makefile
- KConfig

TESTING

1. To test, **copy the configuration file into /etc** folder OR **run rbs_script.sh with root permission.** Make sure that Configuration file has been created in /etc folder
2. Considering this default configuration file provided here, there are total five users who have **user id greater than 1000**. We have taken only user having ids greater than 1000 because below that all the users are actually **start-up/login processes** which should definitely be allowed access to all the functionality otherwise system may halt in an inconsistent state.
3. Now, on my ubuntu system I tested with **two users - udit** and **syssec_udit**. Make sure that both users have already been added in the system. [**Check /etc/passwd**].
4. **udit** user has the **admin** role while **syssec_udit** user has the **non-admin** role in the provided file.

TESTING (ADMIN)

For user udit:

```
udit@mylaptop:~/CSE509/test$ touch file
udit@mylaptop:~/CSE509/test$ ls
file  rbslinux.conf  README
udit@mylaptop:~/CSE509/test$ rm file
udit@mylaptop:~/CSE509/test$ ls
rbslinux.conf  README
udit@mylaptop:~/CSE509/test$ mkdir dir
udit@mylaptop:~/CSE509/test$ ls
dir  rbslinux.conf  README
```

TESTING (ADMIN)

```
udit@mylaptop:~/CSE509/test$ rmdir dir
udit@mylaptop:~/CSE509/test$ ls
rbslinux.conf  README
udit@mylaptop:~/CSE509/test$ touch file
udit@mylaptop:~/CSE509/test$ ls
file  rbslinux.conf  README
udit@mylaptop:~/CSE509/test$ mv file file1
udit@mylaptop:~/CSE509/test$ ls
file1  rbslinux.conf  README
udit@mylaptop:~/CSE509/test$ rm file1
udit@mylaptop:~/CSE509/test$ ls
rbslinux.conf  README
udit@mylaptop:~/CSE509/test$
```

TESTING (Non-Admin)

For user syssec_udit:

```
syssec_udit@mylaptop:~/CSE509/test$ ls
dirs  rbslinux.conf  README
syssec_udit@mylaptop:~/CSE509/test$ touch file
touch: setting times of ‘file’: No such file or directory
syssec_udit@mylaptop:~/CSE509/test$ rm rbslinux.conf
rm: cannot remove ‘rbslinux.conf’: Operation not permitted
syssec_udit@mylaptop:~/CSE509/test$ mv rbslinux.conf somethingn-else
mv: cannot move ‘rbslinux.conf’ to ‘somethingn-else’: Operation not permitted
syssec_udit@mylaptop:~/CSE509/test$ mkdir dir
mkdir: cannot create directory ‘dir’: Operation not permitted
syssec_udit@mylaptop:~/CSE509/test$ rmdir dirs
rmdir: failed to remove ‘dirs’: Operation not permitted
```

POLICY STORE (CONFIGURATION)

/etc/rbslinux.conf - [NAME USERID ROLE]

```
nobody 65534 0
udit 1000 0
syssec_udit 1001 1
sec_udit1 1002 0
sec_udit2 1003 0
```

USER SPACE UTILITY

1. User-Space Utility to add and delete roles for role based file system access control.
2. Two roles are present in current version: Admin(0) and Non-admin(1).
3. Both Roles have some implicit inode permissions.

A) Admin can create files and directories, remove them and rename them.

B) Non-Admin can neither create files/directories nor they can remove/rename them.

USER SPACE UTILITY

```
#Configuration Options
CONFIG_FILE="/etc/rbslinux.conf"
CONFIG_FILE_DLIM=" "

USER_DB="/etc/passwd"
USER_DB_DLIM=":"

MIN_VALID_UID=1000

ADMIN_ID=0
NON_ADMIN_ID=1
```

USER SPACE UTILITY

USAGE & DESCRIPTION

Usage: ./rbs_script.sh [-p] [-i <admin uid>] [-a <non-admin uid>] [-u <admin username>] [-v <non-admin username>]

- ❖ Update Role based Security Database (RBAC Policy Configuration) by adding and deleting roles for the specified users/uids (Default role is admin for all users).

USER SPACE UTILITY

USAGE & DESCRIPTION

-p	Repopulate the whole RBS database with default values.
-i <admin uid>	Add Admin role for user with specified uid.
-a <non-admin uid>	Add Non-Admin role for user with specified uid.
-u <admin username>	Add Admin role for specified user name.
-v <non-admin username>	Add Non-Admin role for specified user name.

USER SPACE UTILITY

INSTALLATION STEPS

1. Run the script **./rbs_script.sh -p** to **create and populate the RBAC policy database** with **default initial configuration** in the **/etc/rbslinux.conf**.

2. The script should be run with **root permission**. Only authenticated administrators should be allowed to add and create roles in the system.

REFERENCES

1. *http://blog.ptsecurity.com/2012/09/writing-linux-security-module.html*

Questions ??

THANK YOU !!!