CS353: Linux Kernel Project 4 Report

Weichen Li
5120309662
eizo.lee@sjtu.edu.cn
Department of Computer
Science and Engineering
School of Electronic Information
and Electrical Engineering
Shanghai Jiao Tong University

CONTENTS

I. OBJECT

- Source:
  - Super.c/Makefile (kernel source of romfs).
  - test.img (a romfs image, you can mount it to a directory *xxx* with *mount -o loop test.img xxx*).
  - Say test.img is mounted in directory *t*, *find t* output:
    - aa
    - bb
    - ft
    - fo
    - fo/aa
- Practice 1:
  - Change romfs code to hide a file/directory with special name.
  - Test & result:
    - *insmod romfs hided_file_name="aa"*.
    - *mount -o loop test.img t*.
    - Then *ls t*, *ls t/fo*, no "aa" and "fo/aa" found.
    - *ls t/aa*, or *ls fo/aa*, no found.
    - Without the code change, above two oerations can find the file "aa".
- Practice 2:
  - Change the code of romfs to correctly read info of an 'encrypted' romfs.
  - Test & result:
    - *insmod romfs encrypted_file_name="bb"*.
    - *mount -o loop test.img t*.
    - Say bb's original content is 'bbbbbbb'.
    - With the change, cat t/bb output 'ccccccc'.
- Practice 3:
  - Change the code of romfs to add 'x'(execution) bit for a specific file.
  - Test & result:

* *insmod romfs hided_file_name="aa".*
* *mount -o loop test.img t.*
* Without code changes *ls -l t*, output is *-rw-r–r–.*
* With the change, output is *-rwxr-xr-x.*

II. Compile steps

A. Preprocess working

At the very beginning, to save some trivial operations on authority verification, we need get the root authority first. (Need to input the key)

```
su
```

B. Create & modify relevant file

To start our work, we need to create a new directory to store our work files.

```
mkdir /usr/src/Project4
cd /usr/src/Project4
```

1) romfs directory: Firstly, copy the whole *romfs* directory into directory *Project4*.

```
cp -r /usr/src/linux-xxxx/fs/romfs /usr/src/Project4
cd romfs
```

Next we need to modify Makefile file:

```
obj-m := romfs.o

romfs-y := super.o storage.o

KDIR := /lib/modules/$(shell uname -r)/build

EXTRA_FLAGS := -I(PWD)

PWD := $(shell pwd)

all:

        make -C $(KDIR) M=$(PWD) modules

clean:

        rm *.o *.ko *.mod.c Module.symvers modules.order -f
```

Finally we need to modify *super.c* (shown from Fig 1 to Fig 4):

```
77  #include <linux/moduleparam.h>
78  #include "internal.h"
79
80  //Add module parameters
81  static char *hided_file_name="null";
82  static char *encrypted_file_name="null";
83  static char *exec_file_name="null";
84  module_param(hided_file_name,charp,S_IRUGO);
85  module_param(encrypted_file_name,charp,S_IRUGO);
86  module_param(exec_file_name,charp,S_IRUGO);
87
88  static struct kmem_cache *romfs_inode_cachep;
```

Fig. 1. First modification

```
        if (ret < 0)
            goto out;
        fsname[j] = '\0';

        ino = offset;
        nextfh = be32_to_cpu(ri.next);

        //*If the file name matches hided_file_name, skip it
        if(strcmp(fsname, hided_file_name)==0){
            offset = nextfh & ROMFH_MASK;
            continue;
        }
        ////


        if ((nextfh & ROMFH_TYPE) == ROMFH_HRD)
            ino = be32_to_cpu(ri.spec);
```

Fig. 2. Second modification

```
        ret = romfs_dev_read(inode->i_sb, pos, buf, fillsize);
        if (ret < 0) {
            SetPageError(page);
            fillsize = 0;
            ret = -EIO;
        }
    }


    //Encryption. Replace all the character by c
    if(strcmp(fname, encrypted_file_name)==0){
        for(i = 0; i < fillsize; i++)
            *((char *)buf + i) = 'c';
    }

    if (fillsize < PAGE_SIZE)
        memset(buf + fillsize, 0, PAGE_SIZE - fillsize);
    if (ret == 0)
        SetPageUptodate(page);
```

Fig. 3. Third modification

2) test.img: Create a directory *test*

```
mkdir test
```

and create some directories and files in it.
Then create a romfs image file *test.img*

```
genromfs -V "vromfs" -f test.img -d test
```

Add a directory *t* to which we mount the image file

```
    /* Hard link handling */
    if ((be32_to_cpu(ri.next) & ROMFH_TYPE) == ROMFH_HRD)
        offset = be32_to_cpu(ri.spec) & ROMFH_MASK;

    inode = romfs_iget(dir->i_sb, offset);


    //Make the file executable
    if(strcmp(name, exec_file_name)==0){
        inode->i_mode |= S_IXUGO;
        //for test
        //printk(KERN_INFO"file name: %s \n", name);
    }


    if (IS_ERR(inode)) {
        ret = PTR_ERR(inode);
        goto error;
    }
    goto outi;
```

Fig. 4. Fourth modification

```
mkdir t
```

C. Compile the module

So far we have completed all the preparation. Next, input

```
make
```

to compile the *romfs* module.

D. Test the romfs module

1) Original outputs: Here is the original outputs corresponding to Practice 1, 2 and 3 (shown from Fig 5 to 7):

```
root@ubuntu:/home/blade/Desktop/romfs# find t
t
t/ft
t/bb
t/cc
t/aa
t/fo
t/fo/aa
```

Fig. 5. Practice 1 (without modification)

```
root@ubuntu:/home/blade/Desktop/romfs# cat t/bb
bbbbbbbbbbbbbbbroot@ubuntu:/home/blade/Desktop/romfs#
```

Fig. 6. Practice 2 (without modification)

```
root@ubuntu:/home/blade/Desktop/romfs# ls -l t
total 0
drwxr-xr-x 1 root root 32 Jan  1  1970 aa
-rw-r--r-- 1 root root 15 Jan  1  1970 bb
-rw-r--r-- 1 root root 16 Jan  1  1970 cc
drwxr-xr-x 1 root root 32 Jan  1  1970 fo
drwxr-xr-x 1 root root 32 Jan  1  1970 ft
```

Fig. 7. Practice 3 (without modification)

2) Outputs with modifications on source: Next, here is the outputs with modification on source code (shown from Fig 8 to 10):

```
root@ubuntu:/home/blade/Desktop/romfs# insmod romfs.ko hided_file_name=aa
root@ubuntu:/home/blade/Desktop/romfs# mount -o loop test.img t
mount: warning: t seems to be mounted read-only.
root@ubuntu:/home/blade/Desktop/romfs# find t
t
t/ft
t/bb
t/cc
t/fo
```

Fig. 8. Practice 1 (with modification)

Note that every time we want to test a practice, we need to unmount the .img from the system, so that we can remove the *romfs.ko* and insert it again.

```
root@ubuntu:/home/blade/Desktop/romfs# insmod romfs.ko encrypted_file_name=bb
root@ubuntu:/home/blade/Desktop/romfs# mount -o loop test.img t
mount: warning: t seems to be mounted read-only.
root@ubuntu:/home/blade/Desktop/romfs# cat t/bb
```

Fig. 9. Practice 2 (with modification)

```
root@ubuntu:/home/blade/Desktop/romfs# insmod romfs.ko exec_file_name=cc
root@ubuntu:/home/blade/Desktop/romfs# mount -o loop test.img t
mount: warning: t seems to be mounted read-only.
root@ubuntu:/home/blade/Desktop/romfs# ls -l t
total 0
drwxr-xr-x 1 root root 32 Jan  1  1970 aa
-rw-r--r-- 1 root root 15 Jan  1  1970 bb
-rwxr-xr-x 1 root root 16 Jan  1  1970 cc
drwxr-xr-x 1 root root 32 Jan  1  1970 fo
drwxr-xr-x 1 root root 32 Jan  1  1970 ft
```

Fig. 10. Practice 3 (with modification)