Android Device Rooting Lab

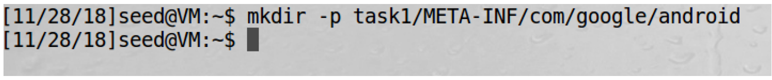
Kai Li

11/29 2018

1 Lab Tasks

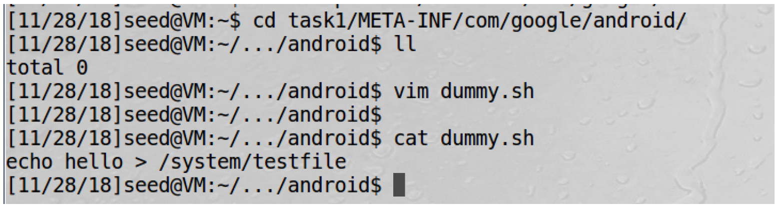
1.1 Lab Task 1: Build a simple OTA package

Step 1: Create OTA structure.



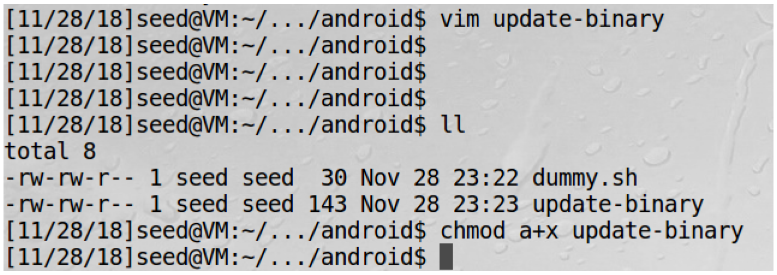
```
[11/28/18]seed@VM:~$ mkdir -p task1/META-INF/com/google/android
[11/28/18]seed@VM:~$ 
```

Step 2: Create dummy.sh.



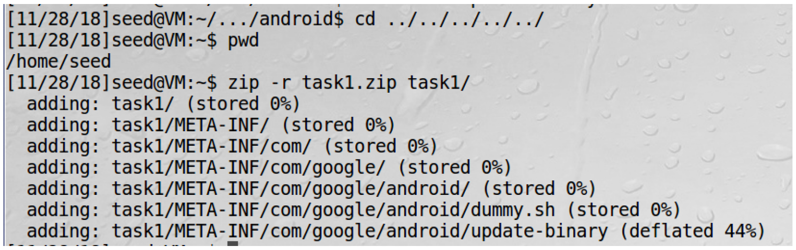
```
[11/28/18]seed@VM:~$ cd task1/META-INF/com/google/android/
[11/28/18]seed@VM:~/.../android$ ll
total 0
[11/28/18]seed@VM:~/.../android$ vim dummy.sh
[11/28/18]seed@VM:~/.../android$
[11/28/18]seed@VM:~/.../android$ cat dummy.sh
echo hello > /system/testfile
[11/28/18]seed@VM:~/.../android$ 
```

Step 3: Create update-binary.



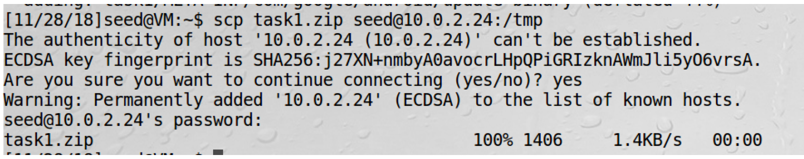
```
[11/28/18]seed@VM:~/.../android$ vim update-binary
[11/28/18]seed@VM:~/.../android$
[11/28/18]seed@VM:~/.../android$
[11/28/18]seed@VM:~/.../android$
[11/28/18]seed@VM:~/.../android$ ll
total 8
-rw-rw-r-- 1 seed seed  30 Nov 28 23:22 dummy.sh
-rw-rw-r-- 1 seed seed 143 Nov 28 23:23 update-binary
[11/28/18]seed@VM:~/.../android$ chmod a+x update-binary
[11/28/18]seed@VM:~/.../android$ 
```

Step 4: Zip OTA.

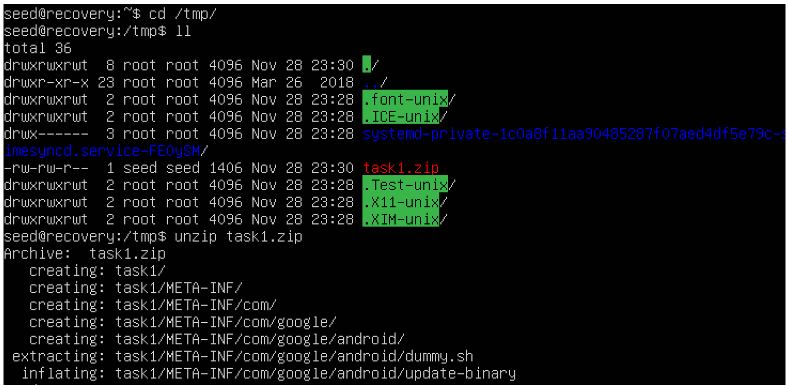


```
[11/28/18]seed@VM:~/.../android$ cd ../../../../../
[11/28/18]seed@VM:~$ pwd
/home/seed
[11/28/18]seed@VM:~$ zip -r task1.zip task1/
  adding: task1/ (stored 0%)
  adding: task1/META-INF/ (stored 0%)
  adding: task1/META-INF/com/ (stored 0%)
  adding: task1/META-INF/com/google/ (stored 0%)
  adding: 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%)
```

Step 5: Copy OTA to Recovery OS.

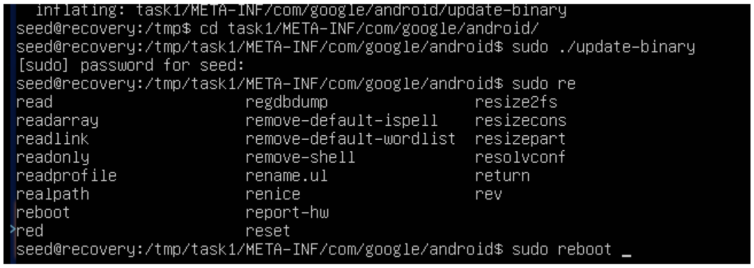


```
[11/28/18]seed@VM:~$ scp task1.zip seed@10.0.2.24:/tmp
The authenticity of host '10.0.2.24 (10.0.2.24)' can't be established.
ECDSA key fingerprint is SHA256:j27XN+nmbyA0avocrLHpQPiGRIzknAWmJli5yO6vrsA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.24' (ECDSA) to the list of known hosts.
seed@10.0.2.24's password:
task1.zip                                     100% 1406     1.4KB/s   00:00
```

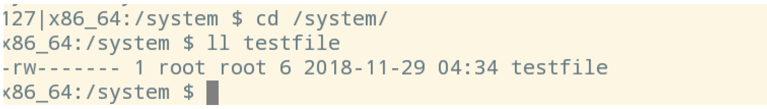


```
seed@recovery:~$ cd /tmp/
seed@recovery:/tmp$ ll
total 36
drwxrwxrwt  8 root root 4096 Nov 28 23:30 ./
drwxr-xr-x 23 root root 4096 Mar 26  2018 ../
drwxrwxrwt  2 root root 4096 Nov 28 23:28 .font-unix/
drwxrwxrwt  2 root root 4096 Nov 28 23:28 .ICE-unix/
drwx------  3 root root 4096 Nov 28 23:28 systemd-private-1c0a8f11aa90485287f07aed4df5e79c-s
imesyncd.service-FEOySM/
-rw-rw-r--  1 seed seed 1406 Nov 28 23:30 task1.zip
drwxrwxrwt  2 root root 4096 Nov 28 23:28 .Test-unix/
drwxrwxrwt  2 root root 4096 Nov 28 23:28 .X11-unix/
drwxrwxrwt  2 root root 4096 Nov 28 23:28 .XIM-unix/
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
```

Step 6:Run OTA and verify result.



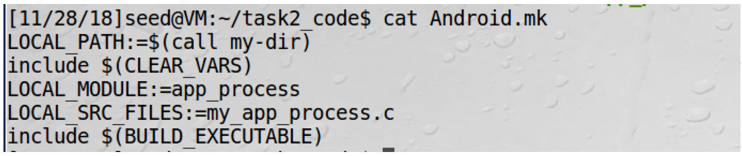
```
  inflating: task1/META-INF/com/google/android/update-binary
seed@recovery:/tmp$ cd task1/META-INF/com/google/android/
seed@recovery:/tmp/task1/META-INF/com/google/android$ sudo ./update-binary
[sudo] password for seed:
seed@recovery:/tmp/task1/META-INF/com/google/android$ sudo re
read                    regdbdump               resize2fs
readarray               remove-default-ispell   resizecons
readlink                remove-default-wordlist resizepart
readonly                remove-shell            resolvconf
readprofile             rename.ul               return
realpath                renice                  rev
reboot                  report-hw
red                     reset
seed@recovery:/tmp/task1/META-INF/com/google/android$ sudo reboot _
```



```
127|x86_64:/system $ cd /system/
x86_64:/system $ ll testfile
-rw------- 1 root root 6 2018-11-29 04:34 testfile
x86_64:/system $
```

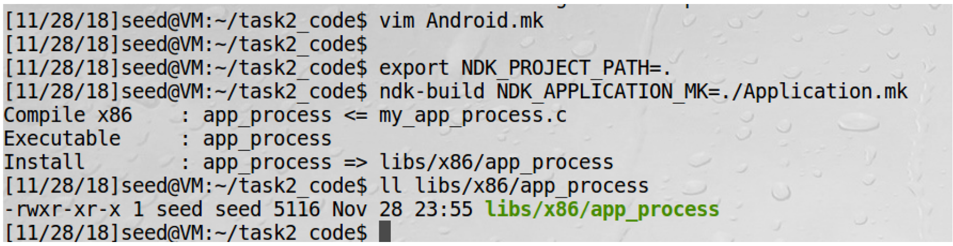
1.2 Task 2: Inject code via app process

Step 1. Compile the code. Firstly let's create a folder 'task2_code' in the Ubuntu16.04, and place the source code inside the folder. Then we need to modify the Android.mk to be the following:



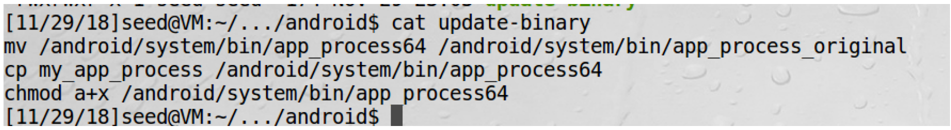
```
[11/28/18]seed@VM:~/task2_code$ cat Android.mk
LOCAL_PATH:=$(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE:=app_process
LOCAL_SRC_FILES:=my_app_process.c
include $(BUILD_EXECUTABLE)
```

then compile the source code as the lab description.



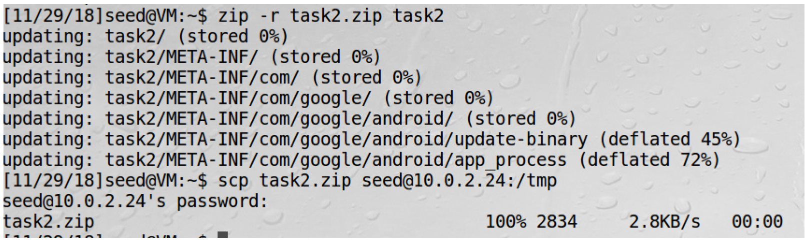
```
[11/28/18]seed@VM:~/task2_code$ vim Android.mk
[11/28/18]seed@VM:~/task2_code$
[11/28/18]seed@VM:~/task2_code$ export NDK_PROJECT_PATH=.
[11/28/18]seed@VM:~/task2_code$ ndk-build NDK_APPLICATION_MK=./Application.mk
Compile x86    : app_process <= my_app_process.c
Executable     : app_process
Install        : app_process => libs/x86/app_process
[11/28/18]seed@VM:~/task2_code$ ll libs/x86/app_process
-rwxr-xr-x 1 seed seed 5116 Nov 28 23:55 libs/x86/app_process
[11/28/18]seed@VM:~/task2 code$
```

Step 2. Write the update script and build OTA package. Firstly, we need to modfify the update_binay as the following:

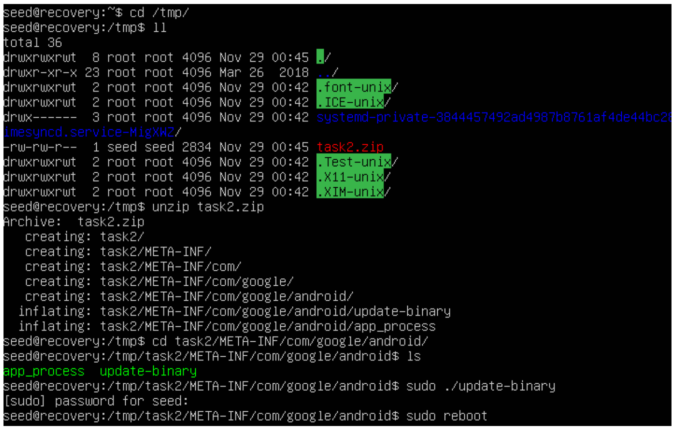


```
[11/29/18]seed@VM:~/.../android$ cat update-binary
mv /android/system/bin/app_process64 /android/system/bin/app_process_original
cp my_app_process /android/system/bin/app_process64
chmod a+x /android/system/bin/app_process64
[11/29/18]seed@VM:~/.../android$
```

Then, we repeat the step 2 and step 3 as the previous task.

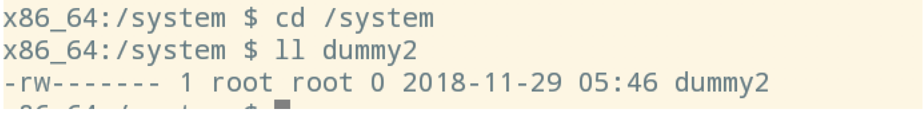


```
[11/29/18]seed@VM:~$ zip -r task2.zip task2
updating: task2/ (stored 0%)
updating: task2/META-INF/ (stored 0%)
updating: task2/META-INF/com/ (stored 0%)
updating: task2/META-INF/com/google/ (stored 0%)
updating: task2/META-INF/com/google/android/ (stored 0%)
updating: task2/META-INF/com/google/android/update-binary (deflated 45%)
updating: task2/META-INF/com/google/android/app_process (deflated 72%)
[11/29/18]seed@VM:~$ scp task2.zip seed@10.0.2.24:/tmp
seed@10.0.2.24's password:
task2.zip                                     100% 2834     2.8KB/s   00:00
```



```
seed@recovery:~$ cd /tmp/
seed@recovery:/tmp$ ll
total 36
drwxrwxrwt  8 root root 4096 Nov 29 00:45 ./
drwxr-xr-x 23 root root 4096 Mar 26  2018 ../
drwxrwxrwt  2 root root 4096 Nov 29 00:42 .font-unix/
drwxrwxrwt  2 root root 4096 Nov 29 00:42 .ICE-unix/
drwx------  3 root root 4096 Nov 29 00:42 systemd-private-3844457492ad4987b8761af4de44bc2
imesyncd.service-MigXWZ/
-rw-rw-r--  1 seed seed 2834 Nov 29 00:45 task2.zip
drwxrwxrwt  2 root root 4096 Nov 29 00:42 .Test-unix/
drwxrwxrwt  2 root root 4096 Nov 29 00:42 .X11-unix/
drwxrwxrwt  2 root root 4096 Nov 29 00:42 .XIM-unix/
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/app_process
seed@recovery:/tmp$ cd task2/META-INF/com/google/android/
seed@recovery:/tmp/task2/META-INF/com/google/android$ ls
app_process  update-binary
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
```

Enter the Android OS and check the result:



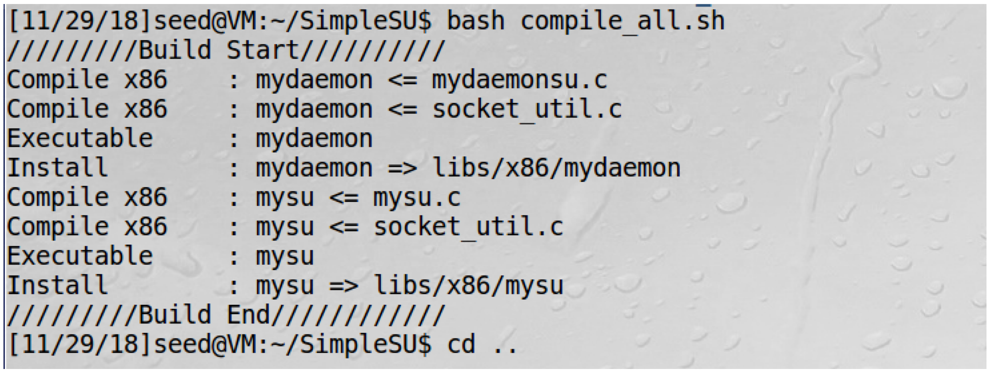
```
x86_64:/system $ cd /system
x86_64:/system $ ll dummy2
-rw------- 1 root root 0 2018-11-29 05:46 dummy2
x86_64:/system $
```

Observation: From the above screenshot, we can see that the dummy2 file was generated in the */system* folder.

Explanation: By following the guidelines prompted in the lab description, once the textitZygote deamon get started, it will invoke the **app_process** which now is linked to our code, our code wraps up the original app_process, it will create the dummy2 file in /system folder, and invoke the original **app_process**.

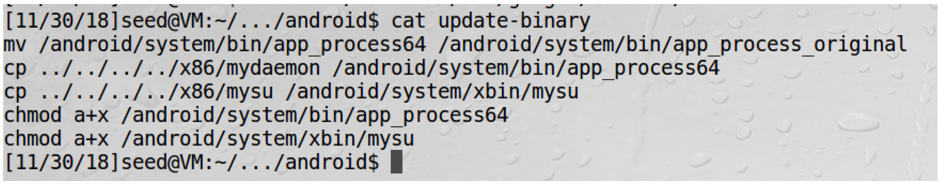
1.3 Task 3: Implement SimpleSU for Getting Root Shell

Step 1. Compile the code.



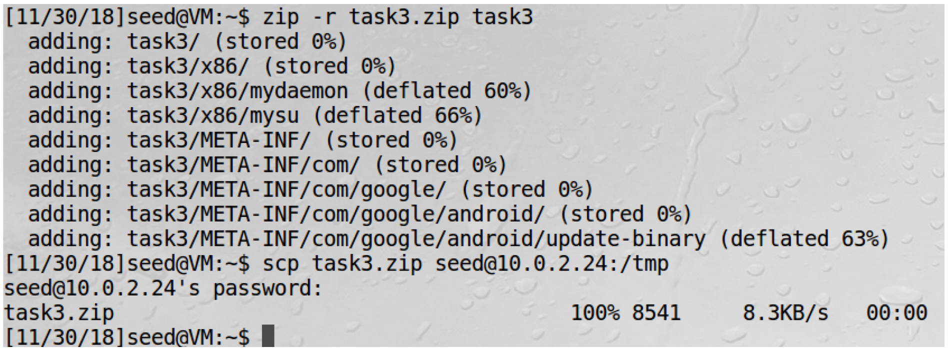
```
[11/29/18]seed@VM:~/SimpleSU$ bash compile_all.sh
/////////Build Start/////////
Compile x86     : mydaemon <= mydaemonsu.c
Compile x86     : mydaemon <= socket_util.c
Executable      : mydaemon
Install         : mydaemon => libs/x86/mydaemon
Compile x86     : mysu <= mysu.c
Compile x86     : mysu <= socket_util.c
Executable      : mysu
Install         : mysu => libs/x86/mysu
/////////Build End///////////
[11/29/18]seed@VM:~/SimpleSU$ cd ..
```

Step 2. Write the update script and build OTA package. Below is the specific update-binary for this task.



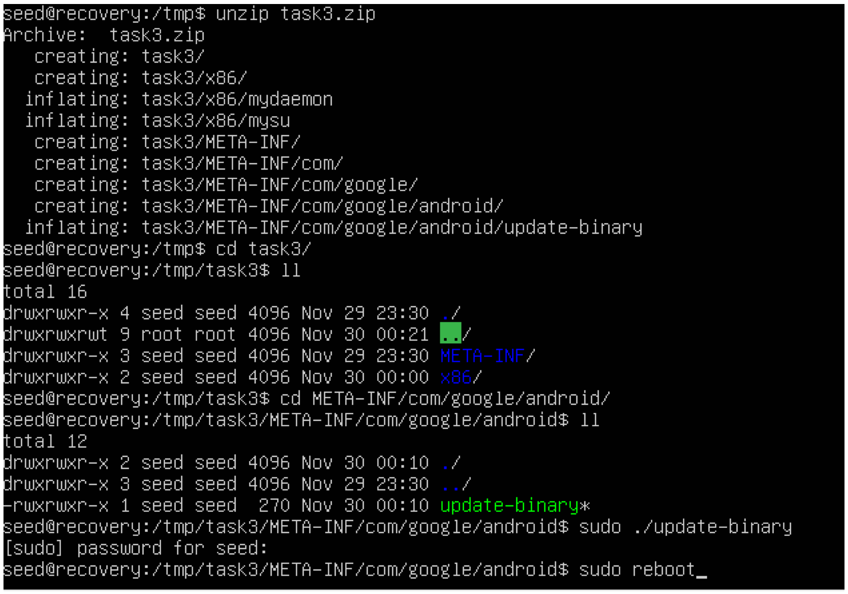
```
[11/30/18]seed@VM:~/.../android$ cat update-binary
mv /android/system/bin/app_process64 /android/system/bin/app_process_original
cp ../../../../x86/mydaemon /android/system/bin/app_process64
cp ../../../../x86/mysu /android/system/xbin/mysu
chmod a+x /android/system/bin/app_process64
chmod a+x /android/system/xbin/mysu
[11/30/18]seed@VM:~/.../android$ █
```

We use the following command to construct the OTA package.



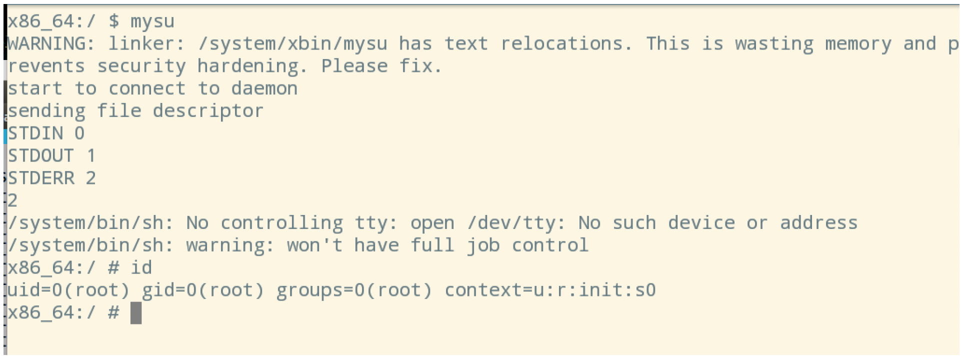
```
[11/30/18]seed@VM:~$ zip -r task3.zip task3
  adding: task3/ (stored 0%)
  adding: task3/x86/ (stored 0%)
  adding: task3/x86/mydaemon (deflated 60%)
  adding: task3/x86/mysu (deflated 66%)
  adding: task3/META-INF/ (stored 0%)
  adding: task3/META-INF/com/ (stored 0%)
  adding: task3/META-INF/com/google/ (stored 0%)
  adding: task3/META-INF/com/google/android/ (stored 0%)
  adding: task3/META-INF/com/google/android/update-binary (deflated 63%)
[11/30/18]seed@VM:~$ scp task3.zip seed@10.0.2.24:/tmp
seed@10.0.2.24's password:
task3.zip                                     100% 8541     8.3KB/s   00:00
[11/30/18]seed@VM:~$ █
```

Step 3. Switch to the Recovery OS and execute the update-binary



```
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/
seed@recovery:/tmp/task3$ ll
total 16
drwxrwxr-x 4 seed seed 4096 Nov 29 23:30 ./
drwxrwxrwt 9 root root 4096 Nov 30 00:21 ../
drwxrwxr-x 3 seed seed 4096 Nov 29 23:30 META-INF/
drwxrwxr-x 2 seed seed 4096 Nov 30 00:00 x86/
seed@recovery:/tmp/task3$ cd META-INF/com/google/android/
seed@recovery:/tmp/task3/META-INF/com/google/android$ ll
total 12
drwxrwxr-x 2 seed seed 4096 Nov 30 00:10 ./
drwxrwxr-x 3 seed seed 4096 Nov 29 23:30 ../
-rwxrwxr-x 1 seed seed  270 Nov 30 00:10 update-binary*
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 reboot_
```

Observation: Once we rebooted the Android VM, open the terminal and run mysu.



```
x86_64:/ $ mysu
WARNING: linker: /system/xbin/mysu has text relocations. This is wasting memory and p
revents security hardening. Please fix.
start to connect to daemon
sending file descriptor
STDIN 0
STDOUT 1
STDERR 2
2
/system/bin/sh: No controlling tty: open /dev/tty: No such device or address
/system/bin/sh: warning: won't have full job control
x86_64:/ # id
uid=0(root) gid=0(root) groups=0(root) context=u:r:init:s0
x86_64:/ #
```

From the above screenshot, we can see that we successfully got the root shell.

Explanation:

- Server launches the original app process binary: mydaemonsu.c line # 255, in **main** function.
- Client sends its FDs: mysu.c, line # 112,113,114, in **connect_daemon** function.
- Server forks to a child process: mydaemonsu.c, line # 189, in **run_daemon** function.
- Child process receives client's FDs: mydaemonsu.c, line # 147,148,149, in **child_process** function.
- Child process redirects its standard I/O FDs: mydaemonsu.c, line # 152,153,154, in **child_process** function.
- Child process launches a root shell: mysu.c, line # 154, in **main** function.