Referat

Rooting al Dispozitivelor Android

Student: **Taradaciuc Nicolae**

Grupa: 2A

Cuprins

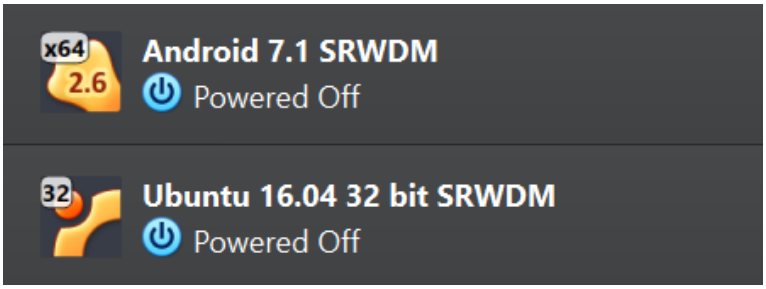
1. Introducere

Scopul acestui laborator este de a analiza procesul de rooting al dispozitivelor Android și de a înțelege în detaliu pașii necesari pentru a efectua acest proces. În cadrul laboratorului, voi construi un pachet OTA simplu pentru a demonstra obținerea privilegiilor de root pe dispozitivele Android, voi explora o metodă de injectare a codului în sistem prin intermediul app_process și voi implementa un tool numit SimpleSu, care va permite obținerea unui acces root shell pe dispozitivul Android.

2. Configurarea mediului de testare

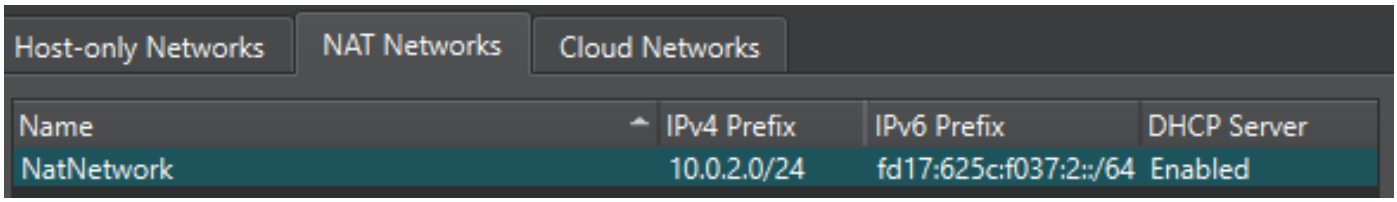
- Crearea mașinilor virtuale

  Instalarea a două mașini virtuale:
  - ***Ubuntu*** *16.04*
  - ***Android*** *7.1*

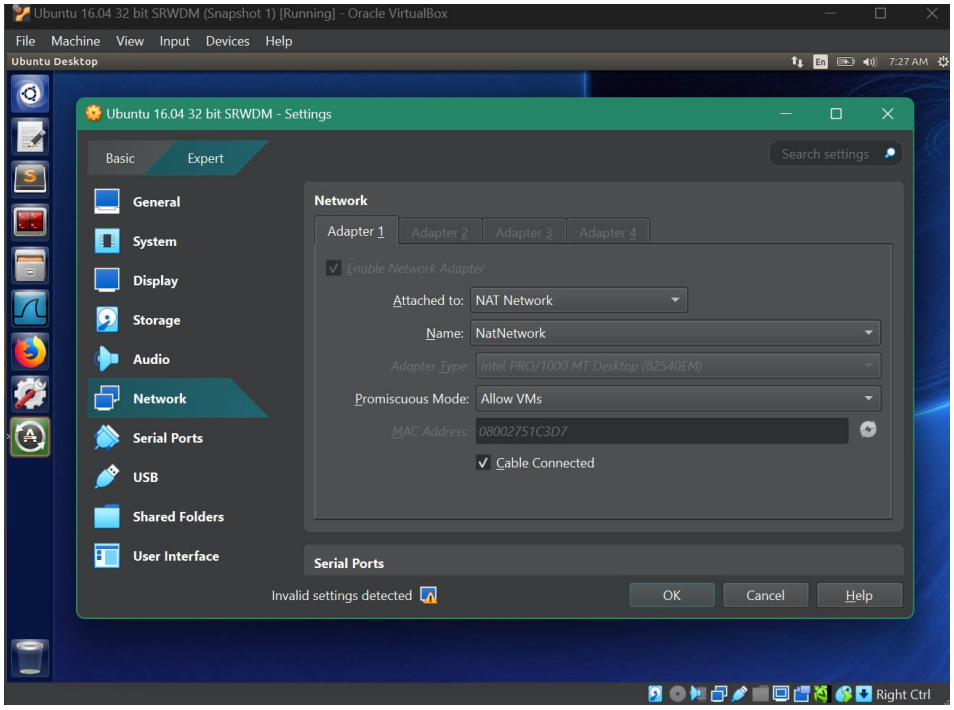


- Configurarea rețelei

Primul pas este conectarea celor două mașini virtuale la aceeași rețea locală, utilizând un adaptor de rețea numit “NAT Network”. Acest adaptor, Network Address Translation funcționează similar cu o rețea locală (LAN), permițând comunicația între mașinile virtuale din aceeași rețea locală și accesul la Internet pentru fiecare mașină virtuală.

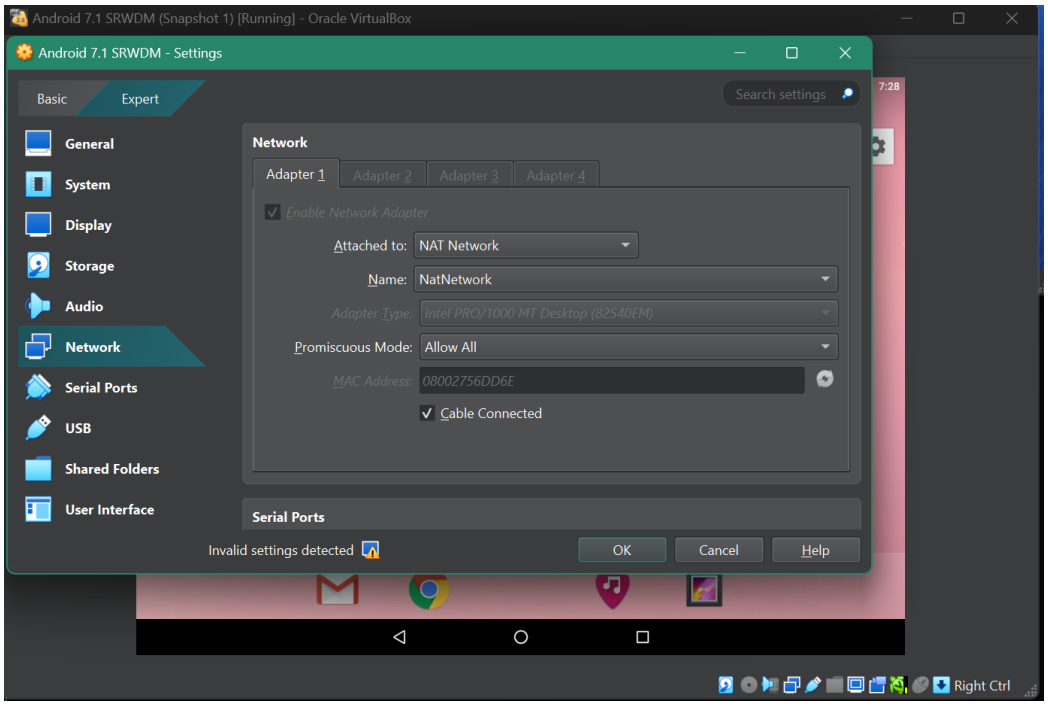


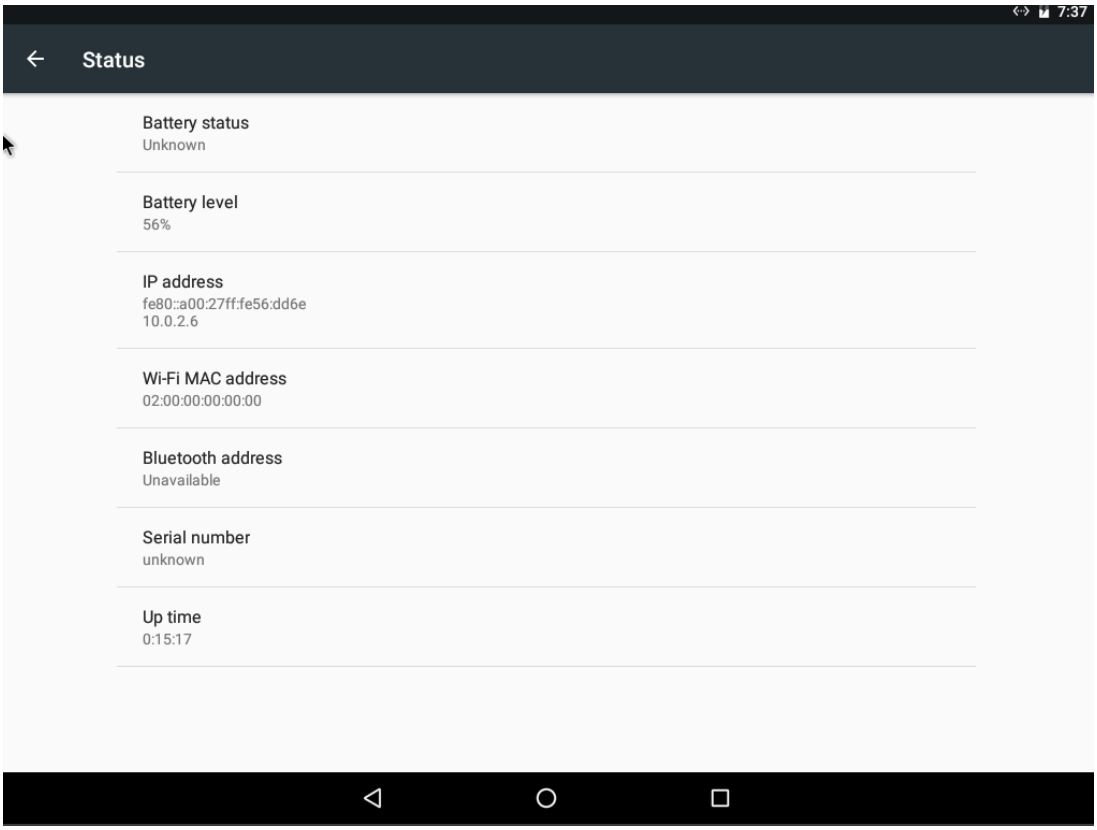
Configurare rețea Ubuntu 16.04:




```
root@VM:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:51:c3:d7 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.5/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 554sec preferred_lft 554sec
    inet6 fe80::cdb1:1432:a9ea:673c/64 scope link
       valid_lft forever preferred_lft forever
```

Configurare rețea Android 7.1:





În urma analizei ambelor dispositive, am extras IP-urile acestora:

- Android: **10.0.2.6**
- Ubuntu: **10.0.2.5**

3. Task 1: Crearea unui packet OTA

În aceast task vom construi un pachet OTA pentru rootarea Android-ului OS.

Structura fișierelor va fi următoarea:

```
META-INF/
  com/
    google/
      android/
        update-binary
        updater-script

system/
  xbin/
    dummy.sh
```

- Crearea fișierului dummy.sh (Crearea scriptului shell ce va fi injectat în Android)

Fișierul dummy.sh este un script shell simplu care va fi injectat în sistemul Android pentru a executa o comanda ce introduce textul “hello” în /system/dummy. Acest script, în practică poate include comenzi ce vor permite obținerea permisiunilor de root și alte setări necesare pentru rootarea dispozitivului Android.

```
root@VM:~/task1/META-INF/com/google/android# ls
dummy.sh
root@VM:~/task1/META-INF/com/google/android# cat dummy.sh
echo hello > /system/dummy
```

- Stocarea scriptului în pachetul OTA (Crearea fișierului update binary)

În această etapă, vom stoca scriptul creat (dummy.sh) în pachetul OTA astfel încât să îl putem aplica ulterior pe sistemul Android și vom adăuga o comandă nouă în fișierul *init.sh* (pentru a se executa la pornirea OS-ului).

```
root@VM:~/task1/META-INF/com/google/android# ls
dummy.sh  update-binary
root@VM:~/task1/META-INF/com/google/android# cat 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
```

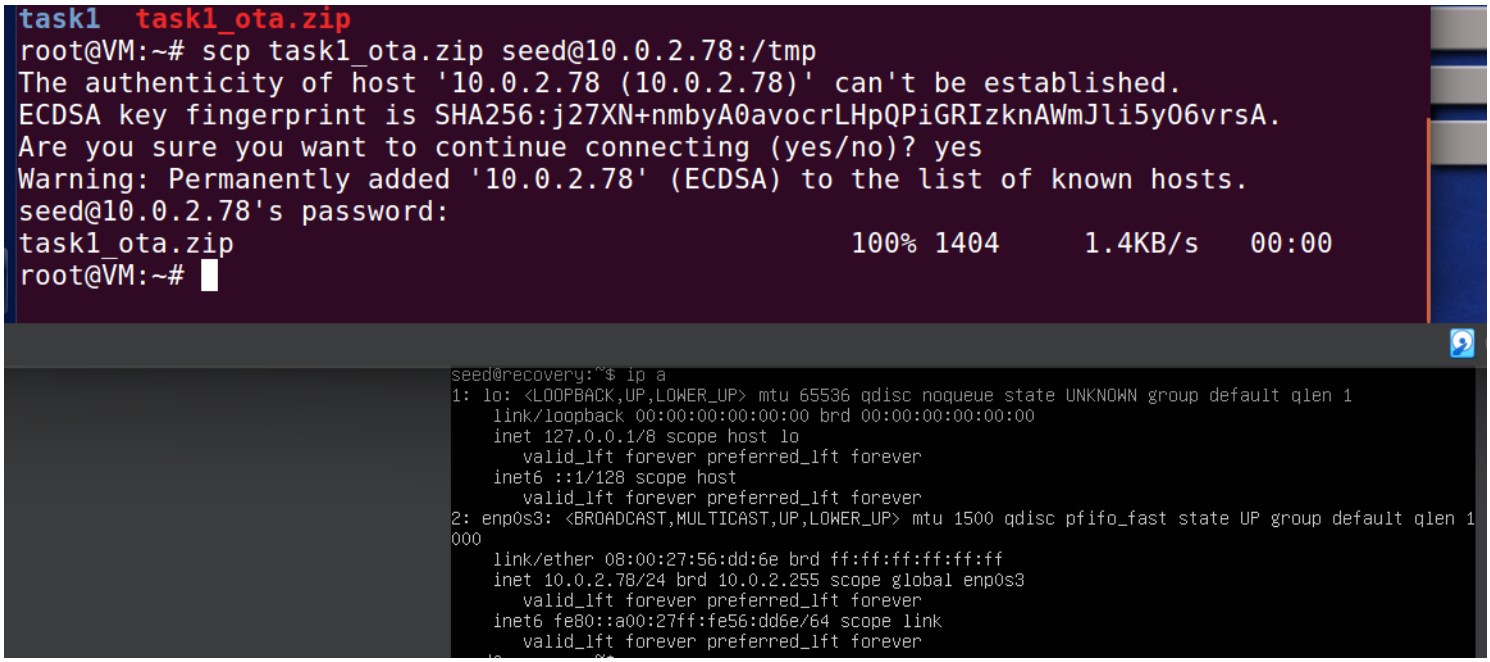
- Crearea arhivei ZIP

După ce am configurat fișierele, vom folosi comanda zip pentru a împacheta totul într-un fișier ZIP.

```
root@VM:~# ls
task1
root@VM:~# zip -r task1_ota.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%)
root@VM:~# ls
task1  task1_ota.zip
```

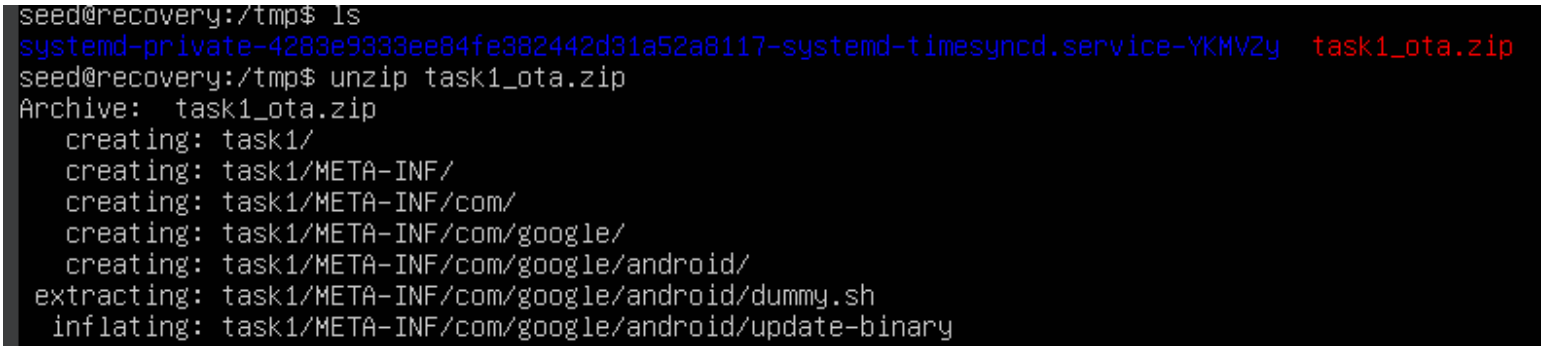
- Transferul fisierului OTA pe OS-ul Android (recovery)

După ce am creat pachetul OTA și am verificat structura acestuia, următorul pas a fost transferul fișierului ZIP pe sistemul de recovery Android pentru a-l putea aplica. Am folosit comanda scp pentru a transfera fișierul ZIP de pe sistemul local pe recovery OS-ul Android.



- Dezarhivarea pachetului OTA

După ce transferal fișierului a fost realizat cu succes, am folosit comanda *unzip* pentru a extrage conținutul ZIP-ului.



- Rularea pachetului OTA

După extragerea fișierelor, am mers în directorul *META-INF/com/google/android*, unde am găsit fișierul *update-binary* și l-am rulat, pentru a actualiza sistemul Android și a active rularea sciptului închis.

```
seed@recovery:/tmp$ cd task1
seed@recovery:/tmp/task1$ cd META-INF/com/google/android/
seed@recovery:/tmp/task1/META-INF/com/google/android$ ls -la
total 16
drwxr-xr-x 2 seed seed 4096 Jan  1 08:39 .
drwxr-xr-x 3 seed seed 4096 Jan  1 08:27 ..
-rw-r--r-- 1 seed seed   27 Jan  1 08:30 dummy.sh
-rwxr-xr-x 1 seed seed  144 Jan  1 08:39 update-binary
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$ _
```

- Verificare

După reboot, am verificat dacă fișierul *dummy* a fost creat în */system* și dacă textul "hello" a fost introdus în acesta.

```
x86_64:/ # cd system
x86_64:/system # ls
app         dummy      fake-libs64  lib         media     vendor
bin         etc        fonts        lib64       priv-app  xbin
build.prop  fake-libs  framework    lost+found  usr
x86_64:/system # cat dummy
hello
```

4. Task 2: Injectarea de cod prin app_process

În acest task, am demonstrat cum să injectăm un program care rulează automat în timpul procesului de boot al Android-ului, folosind procesul app_process

- Stocarea scriptului în pachetul OTA (Crearea fișierului update binary)

În această etapă, vom crea un script ce conține comenzi pentru a copia fișierele necesare în locațiile corespunzătoare pe Android și pentru a schimba permisiunile fișierelor.

```
root@VM:~/task2/META-INF/com/google/android# ls
update-binary
root@VM:~/task2/META-INF/com/google/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
root@VM:~/task2/META-INF/com/google/android#
```

- Crearea fișierelor de configurare pentru NDK

În această etapă, vom crea un script ce conține comenzi pentru a copia fișierele necesare în locațiile corespunzătoare pe Android și pentru a schimba permisiunile fișierelor.

Application.mk:

```
root@VM:~/task2codes# cat Application.mk
APP_ABI := x86
APP_PLATFORM := android-21
APP_STL := stlport_static
APP_BUILD_SCRIPT := Android.mk
```

Android.mk:

```
root@VM:~/task2codes# cat Android.mk
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := my_app_process
LOCAL_SRC_FILES := my_app_process.c
include $(BUILD_EXECUTABLE)
```

- Crearea scriptului pentru compilarea codului

În această etapă am folosit ndk-build pentru a compila codul.

```
root@VM:~/task2codes# cat compile.sh
export NDK_PROJECT_PATH=.
ndk-build NDK_APPLICATION_MK=./Application.mk
```

Continutul codului wrapper (my_app_process.c):

```
root@VM:~/task2codes# cat my_app_process.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);
//Launch the original binary
char* cmd = "/system/bin/app_process_original";
execve(cmd, argv, environ);
//execve() returns only if it fails
return EXIT_FAILURE;
}
```

- Compilarea codului

După compliare fișierele **libs** și **obj** au apărut în directorul principal.

```
root@VM:~/task2codes# ls -la
total 24
drwxr-xr-x  2 root root 4096 Jan  1 09:47 .
drwx------ 11 root root 4096 Jan  1 09:35 ..
-rw-r--r--  1 root root  146 Jan  1 09:47 Android.mk
-rw-r--r--  1 root root   99 Jan  1 09:36 Application.mk
-rwxr-xr-x  1 root root   72 Jan  1 09:40 compile.sh
-rw-r--r--  1 root root  425 Jan  1 09:43 my_app_process.c
root@VM:~/task2codes# ./compile.sh
Compile x86    : my_app_process <= my_app_process.c
Executable     : my_app_process
Install        : my_app_process => libs/x86/my_app_process
root@VM:~/task2codes# ls -la
total 32
drwxr-xr-x  4 root root 4096 Jan  1 09:48 .
drwx------ 11 root root 4096 Jan  1 09:35 ..
-rw-r--r--  1 root root  146 Jan  1 09:47 Android.mk
-rw-r--r--  1 root root   99 Jan  1 09:36 Application.mk
-rwxr-xr-x  1 root root   72 Jan  1 09:40 compile.sh
drwxr-xr-x  3 root root 4096 Jan  1 09:48 libs
-rw-r--r--  1 root root  425 Jan  1 09:43 my_app_process.c
drwxr-xr-x  3 root root 4096 Jan  1 09:48 obj
```

- Mutarea fișierului compilat în locația corespunzătoare din Android

După compilarea codului, vom schimba locația fișierul *my_app_process.c*, pentru a-l utiliza ulterior.

```
root@VM:~/task2codes/libs/x86# mv my_app_process ~/task2/META-INF/com/google/android/
root@VM:~/task2codes/libs/x86#
```

```
root@VM:~/task2/META-INF/com/google/android# ls -la
total 20
drwxr-xr-x 2 root root 4096 Jan  1 09:54 .
drwxr-xr-x 3 root root 4096 Jan  1 09:27 ..
-rwxr-xr-x 1 root root 5116 Jan  1 09:48 my_app_process
-rwxr-xr-x 1 root root  174 Jan  1 09:32 update-binary
root@VM:~/task2/META-INF/com/google/android#
```

- Crearea arhivei ZIP

După ce am configurat fișierele, vom folosi comanda zip pentru a împacheta totul într-un fișier ZIP.

```
root@VM:~# ls
task1  task1_ota.zip  task2  task2codes
root@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%)
root@VM:~# ls
task1  task1_ota.zip  task2  task2codes  task2.zip
root@VM:~#
```

- Transferul fisierului OTA pe OS-ul Android (recovery)

După ce am creat pachetul OTA și am verificat structura acestuia, următorul pas a fost transferul fișierului ZIP pe sistemul de recovery Android pentru a-l putea aplica. Am folosit comanda scp pentru a transfera fișierul ZIP de pe sistemul local pe recovery OS-ul Android.

```
root@VM:~# ls
task1  task1_ota.zip  task2  task2codes  task2.zip
root@VM:~# scp task2.zip seed@10.0.2.78:/tmp
seed@10.0.2.78's password:
task2.zip                                     100% 2830     2.8KB/s   00:00
```

- Dezarhivarea pachetului OTA

După ce transferal fișierului a fost realizat cu succes, am folosit comanda *unzip* pentru a extrage conținutul ZIP-ului.

```
seed@recovery:/tmp$ ls
systemd-private-986fe324aa2e4bde89c623d41a6ed159-systemd-timesyncd.service-07rjqS  task2.zip
seed@recovery:/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$ ls
systemd-private-986fe324aa2e4bde89c623d41a6ed159-systemd-timesyncd.service-07rjqS  task2  task2.zip
```

- Rularea pachetului OTA

După extragerea fișierelor, am mers în directorul *META-INF/com/google/android*, unde am găsit fișierul *update-binary* și l-am rulat, pentru a actualiza sistemul Android și a active rularea sciptului înclus.

```
seed@recovery:/tmp$ cd task2/META-INF/com/google/android/
seed@recovery:/tmp/task2/META-INF/com/google/android$ ls
my_app_process  update-binary
seed@recovery:/tmp/task2/META-INF/com/google/android$ ls -la
total 20
drwxr-xr-x 2 seed seed 4096 Jan  1 09:54 .
drwxr-xr-x 3 seed seed 4096 Jan  1 09:27 ..
-rwxr-xr-x 1 seed seed 5116 Jan  1 09:48 my_app_process
-rwxr-xr-x 1 seed seed  174 Jan  1 09:32 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$ _
```

- Verificare

După reboot, am verificat dacă fișierul *dummy2* a fost creat în */system*.

```
x86_64:/ # cd system
x86_64:/system # ls
app         dummy   fake-libs    framework  lost+found  usr
bin         dummy2  fake-libs64  lib        media       vendor
build.prop  etc     fonts        lib64      priv-app    xbin
```

5. Task 3: Impementarea *SimpleSu* pentru a obtine un shell cu priviegii de root

Scopul acestui task este de a implementa un sistem de obținere a unui shell cu privilegii de root pe un dispozitiv Android folosind tool-ul SimpleSU

- Downloadarea si dezahivarea SimpleSU

Primul pas este descarcarea tool-ului SimpleSu și dezarhivarea acestuia pentru a accesa codul sursă necesar creării pachetului OTA.

```
root@VM:/home/seed/Downloads# ls
SimpleSU.zip
root@VM:/home/seed/Downloads# pwd
/home/seed/Downloads
root@VM:/home/seed/Downloads# cd
root@VM:~# ls
task1  task1_ota.zip  task2  task2codes  task2.zip  task3  task3codes
root@VM:~# cd task3codes
root@VM:~/task3codes# unzip /home/seed/Downloads/SimpleSU.zip
Archive:  /home/seed/Downloads/SimpleSU.zip
   creating: SimpleSU/
   creating: SimpleSU/socket_util/
  inflating: SimpleSU/socket_util/socket_util.c
  inflating: SimpleSU/socket_util/socket_util.h
   creating: SimpleSU/mydaemon/
  inflating: SimpleSU/mydaemon/Android.mk
  inflating: SimpleSU/mydaemon/compile.sh
  inflating: SimpleSU/mydaemon/mydaemonsu.c
  inflating: SimpleSU/mydaemon/Application.mk
  inflating: SimpleSU/compile_all.sh
  inflating: SimpleSU/server_loc.h
   creating: SimpleSU/mysu/
  inflating: SimpleSU/mysu/Android.mk
  inflating: SimpleSU/mysu/compile.sh
  inflating: SimpleSU/mysu/mysu.c
  inflating: SimpleSU/mysu/Application.mk
```

- Compilarea codului

```
root@VM:~/task3codes# ls
SimpleSU
root@VM:~/task3codes# cd SimpleSU/
root@VM:~/task3codes/SimpleSU# ls
compile_all.sh  mydaemon  mysu  server_loc.h  socket_util
root@VM:~/task3codes/SimpleSU# la -la
total 28
drwxr-xr-x 5 root root 4096 May 22  2018 .
drwxr-xr-x 3 root root 4096 Jan  1 11:57 ..
-rw-r--r-- 1 root root  138 Mar 31  2016 compile_all.sh
drwxr-xr-x 2 root root 4096 Jun  9  2018 mydaemon
drwxr-xr-x 2 root root 4096 Jun  9  2018 mysu
-rw-r--r-- 1 root root  371 Mar 11  2016 server_loc.h
drwxr-xr-x 2 root root 4096 Mar 31  2016 socket_util
root@VM:~/task3codes/SimpleSU# chmod a+x compile_all.sh
root@VM:~/task3codes/SimpleSU# la -la
total 28
drwxr-xr-x 5 root root 4096 May 22  2018 .
drwxr-xr-x 3 root root 4096 Jan  1 11:57 ..
-rwxr-xr-x 1 root root  138 Mar 31  2016 compile_all.sh
drwxr-xr-x 2 root root 4096 Jun  9  2018 mydaemon
drwxr-xr-x 2 root root 4096 Jun  9  2018 mysu
-rw-r--r-- 1 root root  371 Mar 11  2016 server_loc.h
drwxr-xr-x 2 root root 4096 Mar 31  2016 socket_util
root@VM:~/task3codes/SimpleSU# ./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/////////
```

- Mutarea fișierelor **mydaemon** și **mysu**

Vom copia fișierul mydaemon(server) și mysu(client), în locația corespunzătoare acestora.

```
root@VM:~/task3codes/SimpleSU/mydaemon/libs/x86# ls
mydaemon
root@VM:~/task3codes/SimpleSU/mydaemon/libs/x86#
root@VM:~/task3codes/SimpleSU/mydaemon/libs/x86#
root@VM:~/task3codes/SimpleSU/mydaemon/libs/x86#
root@VM:~/task3codes/SimpleSU/mydaemon/libs/x86# mv mydaemon ~/task3/META-INF/com/googl
e/android/
```

```
root@VM:~/task3codes/SimpleSU/mysu/libs/x86# mv mysu ~/task3/META-INF/com/google/android/
```

Verificare:

```
root@VM:~/task3codes/SimpleSU/mysu/libs/x86# cd ~/task3/META-INF/com/google/android/
root@VM:~/task3/META-INF/com/google/android# ls
mydaemon  mysu  update-binary
```

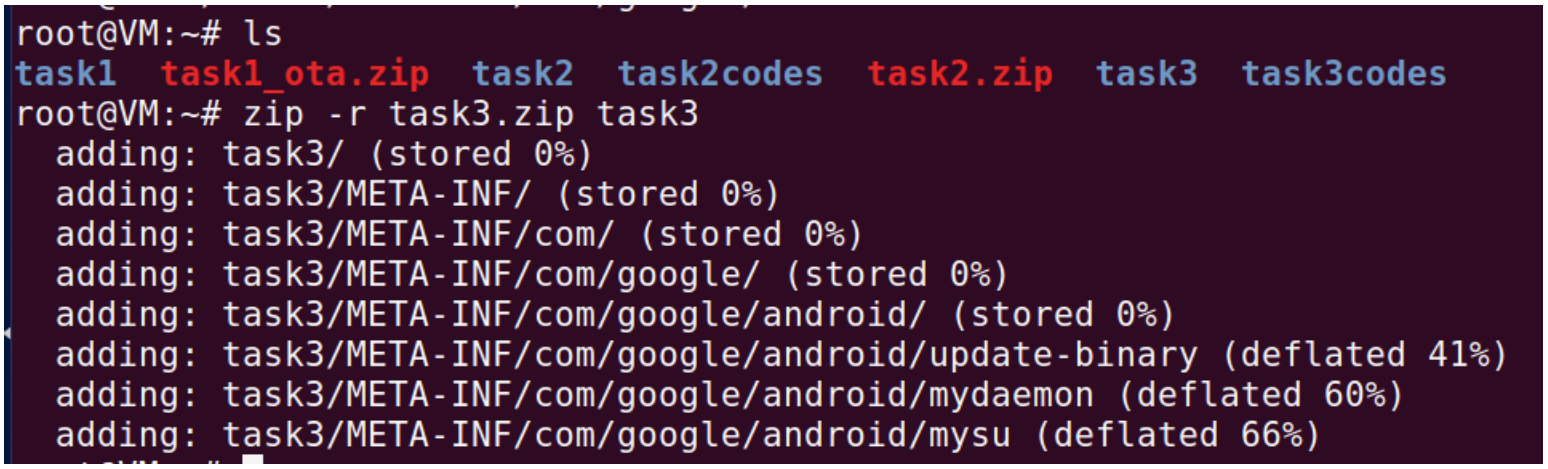
- Stocarea scriptului în pachetul OTA (Crearea fișierului update binary)

În această etapă, vom crea un script ce conține comenzi pentru a copia fișierele necesare în locațiile corespunzătoare pe Android și adăugarea unei comenzi noi în fișierul *init.sh* (pentru a se executa la pornirea OS-ului).

```
root@VM:~/task3/META-INF/com/google/android# ls
update-binary
root@VM:~/task3/META-INF/com/google/android# cat update-binary
cp mysu /android/system/xbin
cp mydaemon /android/system/xbin
sed -i "/return 0/i /system/xbin/mydaemon" /android/system/etc/init.sh
root@VM:~/task3/META-INF/com/google/android#
```

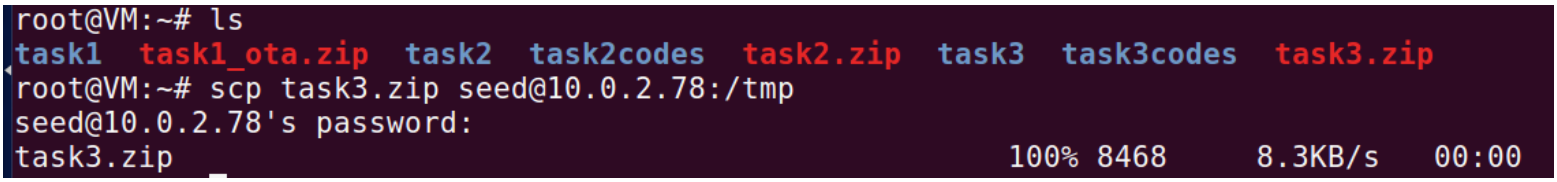
- Crearea arhivei ZIP

După ce am configurat fișierele, vom folosi comanda zip pentru a împacheta totul într-un fișier ZIP.



- Transferul fisierului OTA pe OS-ul Android (recovery)

După ce am creat pachetul OTA și am verificat structura acestuia, următorul pas a fost transferul fișierului ZIP pe sistemul de recovery Android pentru a-l putea aplica. Am folosit comanda scp pentru a transfera fișierul ZIP de pe sistemul local pe recovery OS-ul Android.



- Dezarhivarea pachetului OTA

După ce transferal fișierului a fost realizat cu succes, am folosit comanda *unzip* pentru a extrage conținutul ZIP-ului.

```
seed@recovery:/tmp$ ls
systemd-private-34a93bf622884ad8932d6432bc2874ab-systemd-timesyncd.service-yYMTJl  task3.zip
seed@recovery:/tmp$ unzip task3.zip
Archive:  task3.zip
   creating: task3/
   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
  inflating: task3/META-INF/com/google/android/mydaemon
  inflating: task3/META-INF/com/google/android/mysu
```

- Rularea pachetului OTA

După extragerea fișierelor, am mers în directorul *META-INF/com/google/android*, unde am găsit fișierul *update-binary* și l-am rulat, pentru a actualiza sistemul Android și a active rularea sciptului înclus.

```
seed@recovery:/tmp$ cd task3/META-INF/com/google/android/
seed@recovery:/tmp/task3/META-INF/com/google/android$ ls
mydaemon  mysu  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$
```

- Verificare

După reboot, am folosit comanda *whoami.* Inițial, la rularea comenzii, am primit răspunsul “u0_a36” (user), dar după rularea scriptului *(./mysu)*, la executarea comenzii *whoami,* răspunsul a fost ***“root”***, cee ce a indicat faptul că scriptul a avut succes și am obținut privilegii de adminstrator, având acces complet asupra sistemului Android.

```
x86_64:/ $ whoami
u0_a36
x86_64:/ $ cd system/xbin
x86_64:/system/xbin $ ls -la my*
-rwxr-xr-x 1 root root 9232 2025-01-01 12:20 mydaemon
-rwxr-xr-x 1 root root 9232 2025-01-01 12:20 mysu
x86_64:/system/xbin $ ./mysu
WARNING: linker: /system/xbin/mysu has text relocations. This is wasting memory and
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:/ # whoami
root
```

Pentru a identifica procesele active atât la client (mysu), cât și la server (mydaemon), observăm că mysu este lansat inițial ca proces al utilizatorului, iar mydaemon este activat de mysu și rulează cu privilegii de root.

```
x86_64:/proc # ps | grep mysu
u0_a36    3278  3172  5064   1964          0 0000000000 S ./mysu
x86_64:/proc # ps | grep mydaemon
root      1037  1     5064   364           0 0000000000 S /system/xbin/mydaemon
root      1051  1037  0      0             0 0000000000 Z mydaemon
root      1474  1037  0      0             0 0000000000 Z mydaemon
root      1971  1037  0      0             0 0000000000 Z mydaemon
```

6. Bibliografie

https://seedsecuritylabs.org/Labs_20.04/Mobile/SEEDAndroid_VirtualBox.pdf

https://seedsecuritylabs.org/Labs_16.04/Documents/SEEDVM_VirtualBoxManual.pdf

https://seedsecuritylabs.org/Labs_20.04/Files/Android_Rooting/Android_Rooting.pdf

https://seedsecuritylabs.org/Labs_20.04/Mobile/SEEDAndroid_Recovery_UserManual.pdf

https://seedsecuritylabs.org/Labs_20.04/Mobile/SEEDAndroid_UserManual.pdf