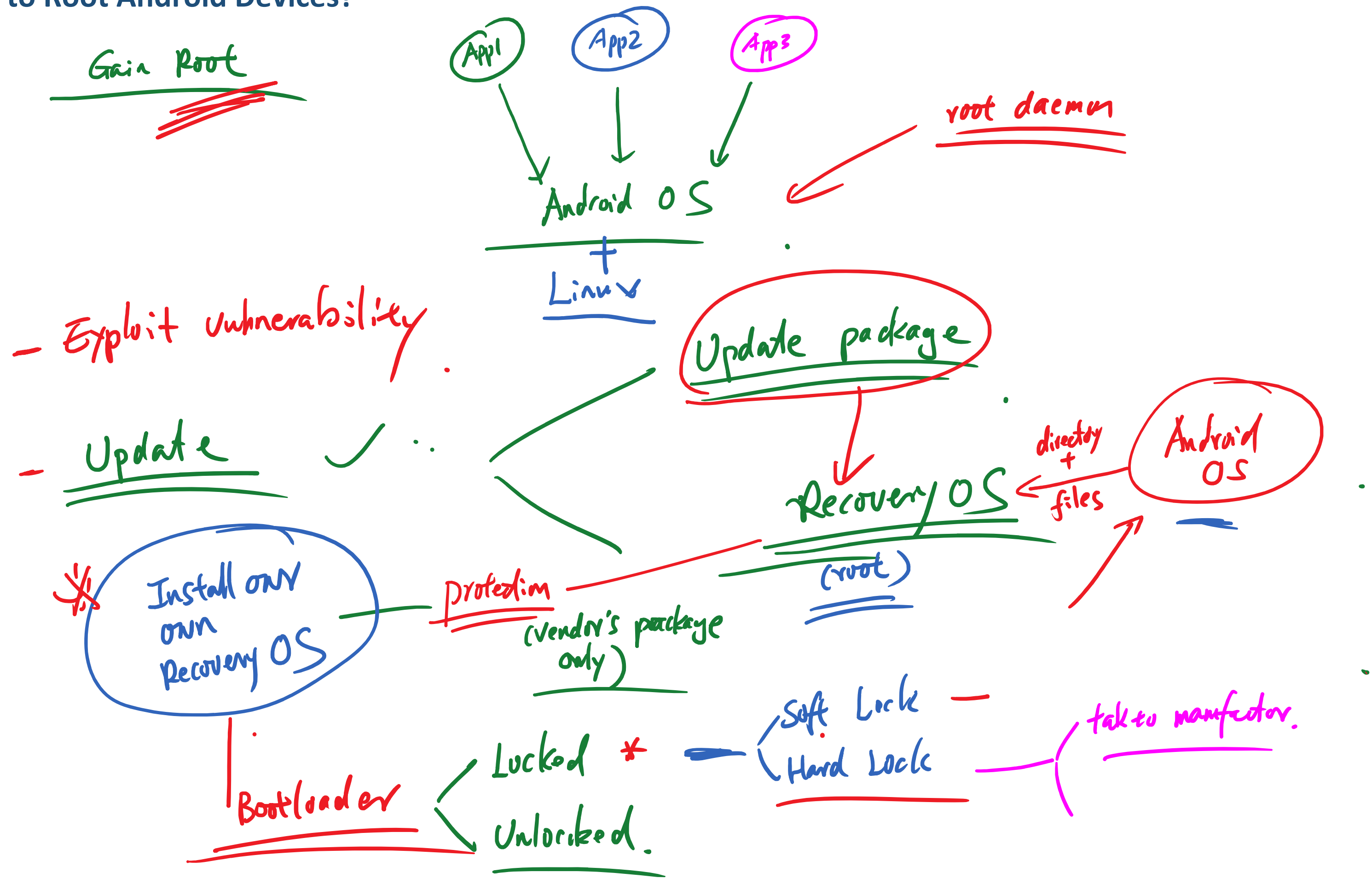
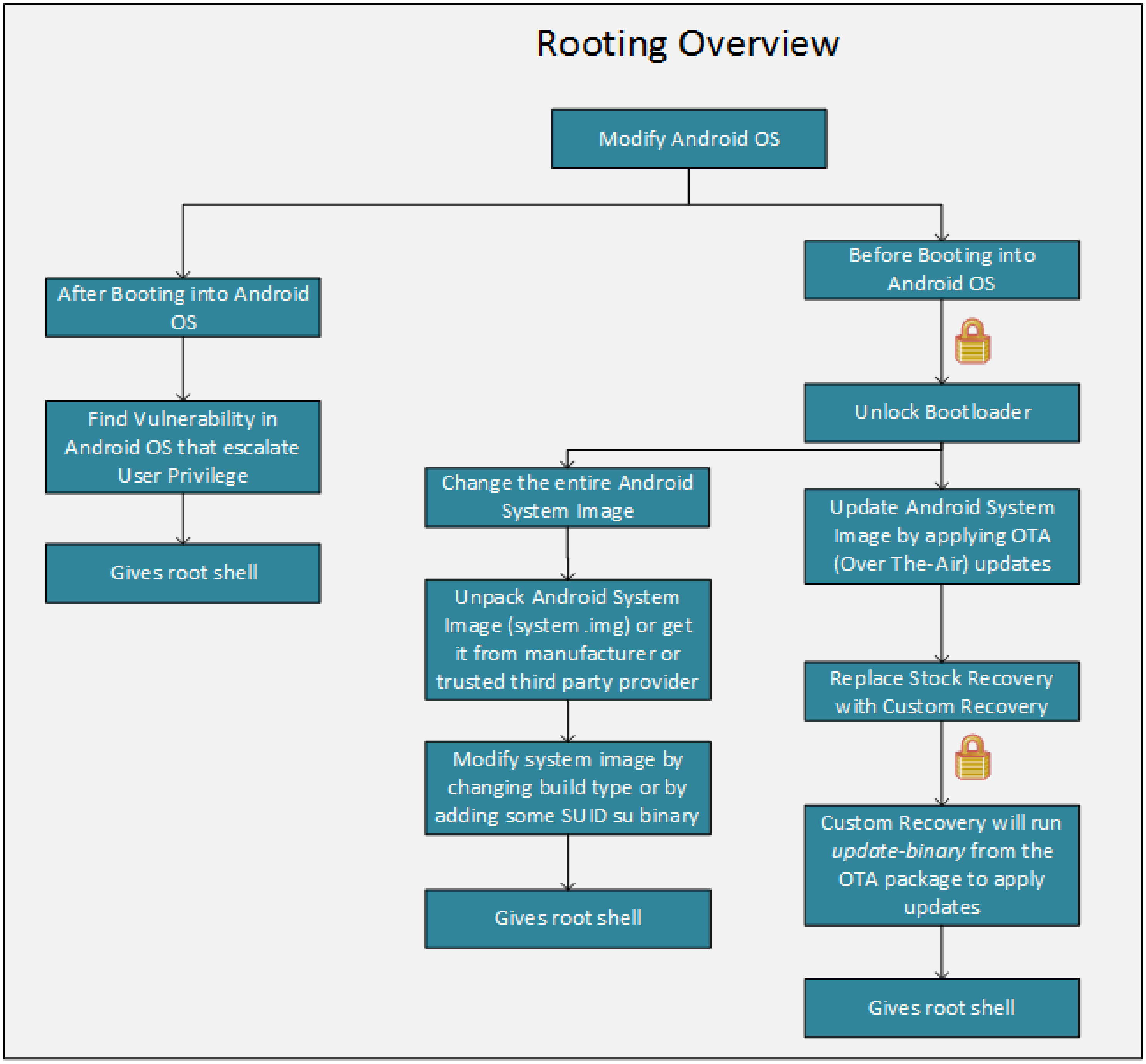
Android Rooting Attack

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

How to Root Android Devices?



Rooting Overview


Rooting Overview
Modify Android OS
After Booting into Android OS
Before Booting into Android OS
Find Vulnerability in Android OS that escalate User Privilege
Gives root shell
Unlock Bootloader
Change the entire Android System Image
Update Android System Image by applying OTA (Over The-Air) updates
Unpack Android System Image (system.img) or get it from manufacturer or trusted third party provider
Replace Stock Recovery with Custom Recovery
Modify system image by changing build type or by adding some SUID su binary
Custom Recovery will run *update-binary* from the OTA package to apply updates
Gives root shell
Gives root shell

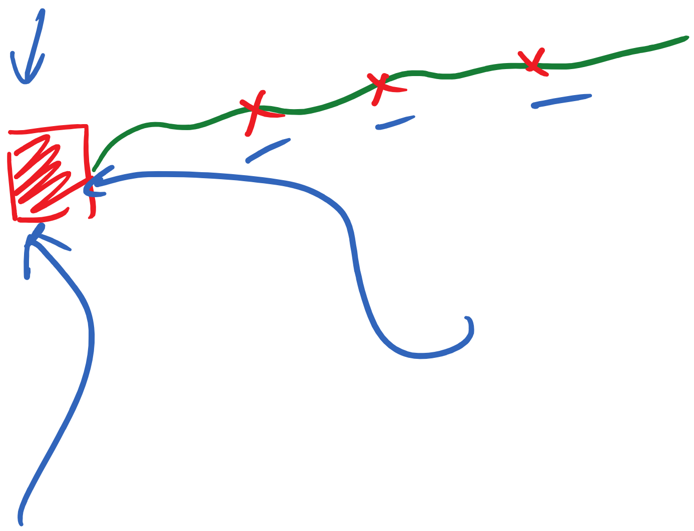
Rooting Approaches

Rooting From Inside

Case Study: Using the "Dirty COW" Exploit

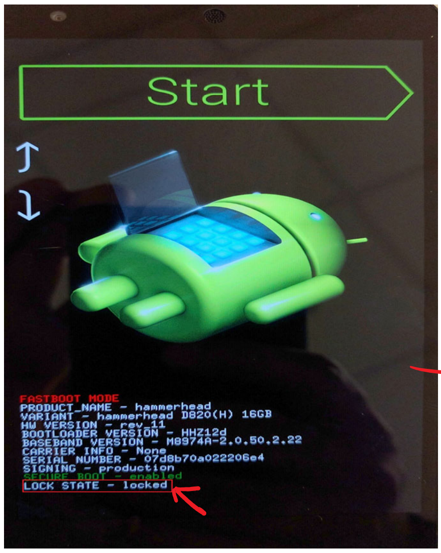
Rooting From Outside

Rooting Real Devices

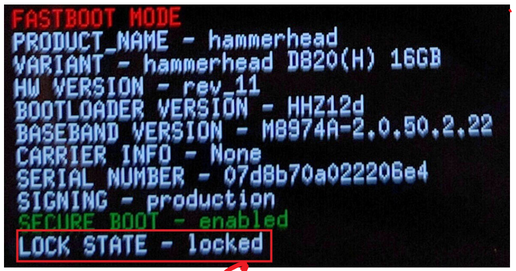


SYRACUSE UNIVERSITY ENGINEERING & COMPUTER SCIENCE

Real Device: Unlocking Bootloader



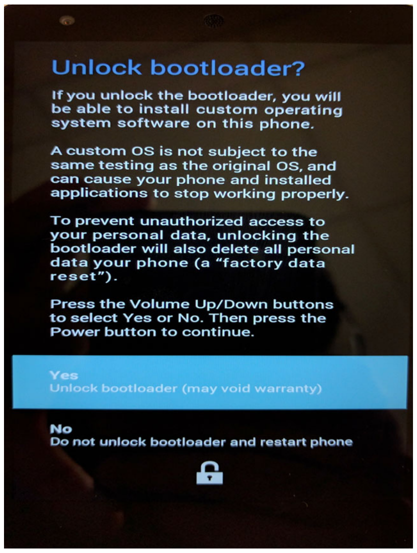
```
FASTBOOT MODE
PRODUCT_NAME - hammerhead
VARIANT - hammerhead D820(H) 16GB
HW VERSION - rev_11
BOOTLOADER VERSION - HHZ12d
BASEBAND VERSION - M8974A-2.0.50.2.22
CARRIER INFO - None
SERIAL NUMBER - 07d8b70a022206e4
SIGNING - production
SECURE BOOT - enabled
LOCK STATE - locked
```



```
FASTBOOT MODE
PRODUCT_NAME - hammerhead
VARIANT - hammerhead D820(H) 16GB
HW VERSION - rev_11
BOOTLOADER VERSION - HHZ12d
BASEBAND VERSION - M8974A-2.0.50.2.22
CARRIER INFO - None
SERIAL NUMBER - 07d8b70a022206e4
SIGNING - production
SECURE BOOT - enabled
LOCK STATE - locked
```

recovery OS

Locked Bootloader



Unlock bootloader?

If you unlock the bootloader, you will be able to install custom operating system software on this phone.

A custom OS is not subject to the same testing as the original OS, and can cause your phone and installed applications to stop working properly.

To prevent unauthorized access to your personal data, unlocking the bootloader will also delete all personal data your phone (a "factory data reset").

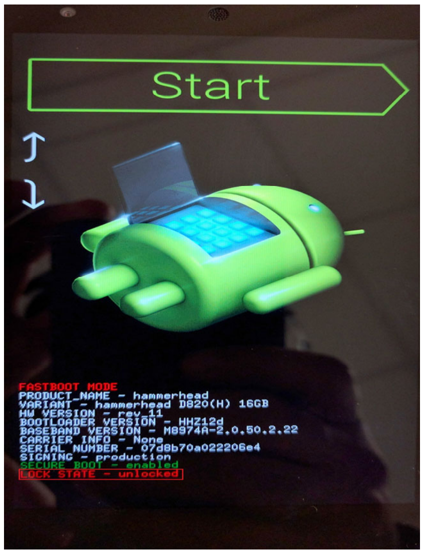
Press the Volume Up/Down buttons to select Yes or No. Then press the Power button to continue.

Yes
Unlock bootloader (may void warranty)

No
Do not unlock bootloader and restart phone

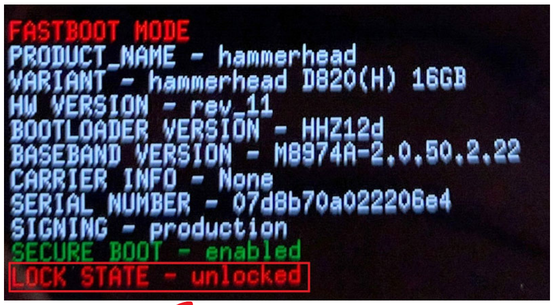
Fastboot: a useful tool

- Send commands to the bootloader
- Modify phone's firmware
- Run "fastboot oem unlock" to unlock



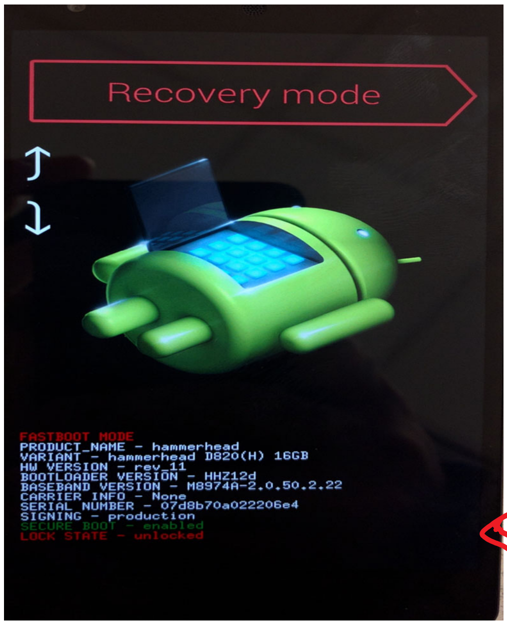
```
FASTBOOT MODE
PRODUCT_NAME - hammerhead
VARIANT - hammerhead D820(H) 16GB
HW VERSION - rev_11
BOOTLOADER VERSION - HHZ12d
BASEBAND VERSION - M8974A-2.0.50.2.22
CARRIER INFO - None
SERIAL NUMBER - 07d8b70a022206e4
SIGNING - production
SECURE BOOT - enabled
LOCK STATE - unlocked
```

Unlocked Bootloader



```
FASTBOOT MODE
PRODUCT_NAME - hammerhead
VARIANT - hammerhead D820(H) 16GB
HW VERSION - rev_11
BOOTLOADER VERSION - HHZ12d
BASEBAND VERSION - M8974A-2.0.50.2.22
CARRIER INFO - None
SERIAL NUMBER - 07d8b70a022206e4
SIGNING - production
SECURE BOOT - enabled
LOCK STATE - unlocked
```

Boot the Custom Recovery OS



```
FASTBOOT MODE
PRODUCT_NAME - hammerhead
VARIANT - hammerhead D820(H) 16GB
HW VERSION - rev_11
BOOTLOADER VERSION - HHZ12d
BASEBAND VERSION - M8974A-2.0.50.2.22
CARRIER INFO - None
SERIAL NUMBER - 07d8b70a022206e4
SIGNING - production
SECURE BOOT - enabled
LOCK STATE - unlocked
```

Boot phone using the custom recovery OS

- Run "fastboot boot CustomRecoveryOS.img"

Replace the recovery OS

```
# fastboot flash recovery CustomRecoveryOS.img
sending 'recovery' (11600 KB) ...
OKAY [ 0.483s]
writing 'recovery' ...
OKAY [ 0.948s]
finished. total time: 1.435s
```

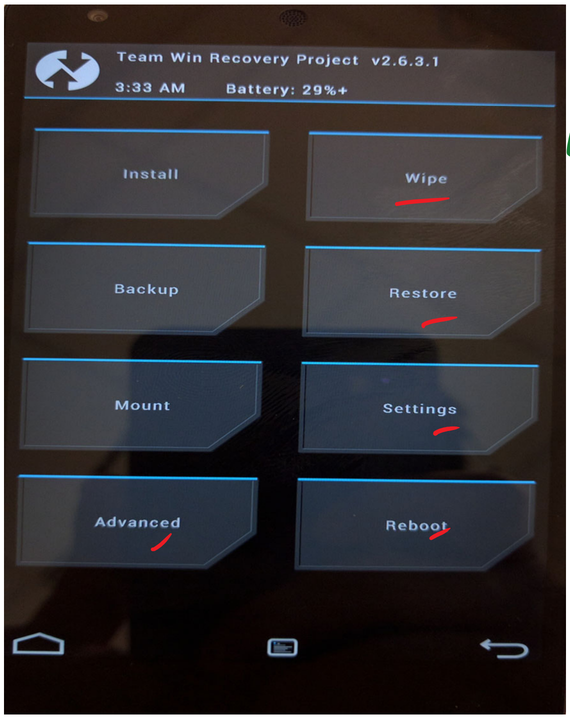
PC

Lab

Recovery OS

Android

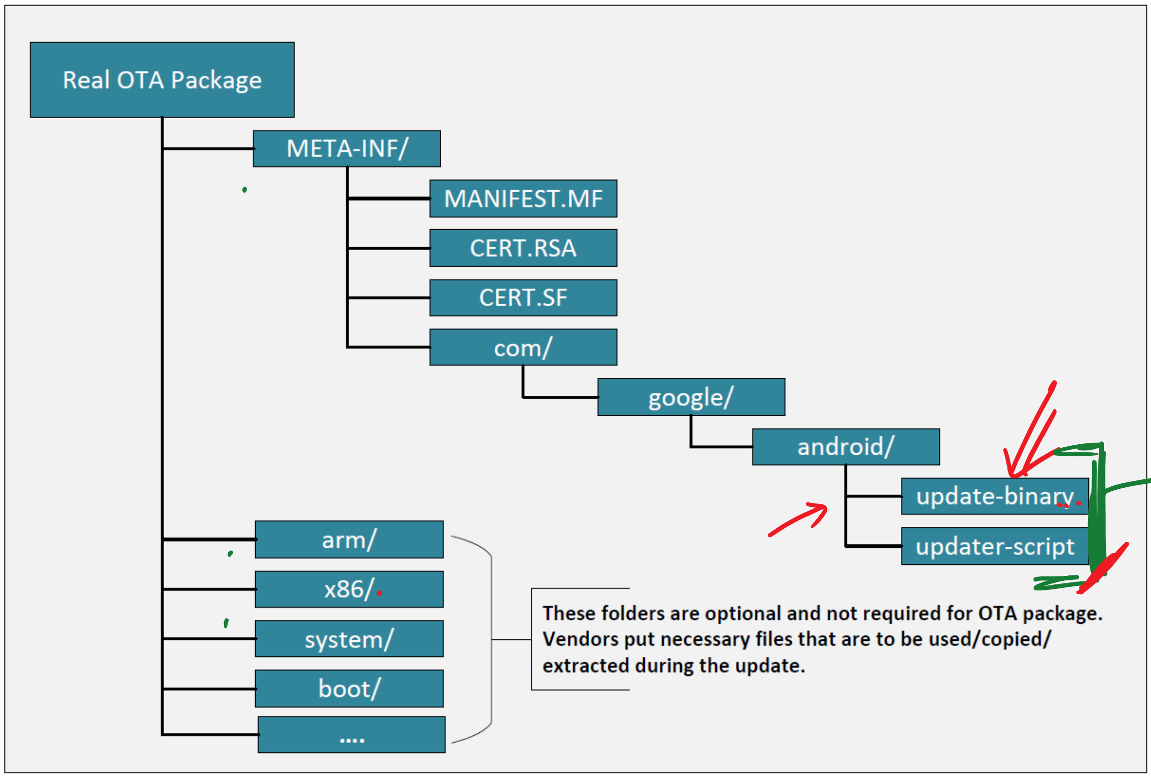
Linux



Custom Recovery OS

How OTA Works

The OTA Structure



Real OTA Package
META-INF/
MANIFEST.MF
CERT.RSA
CERT.SF
com/
google/
android/
update-binary
updater-script
arm/
x86/
system/
boot/
....
These folders are optional and not required for OTA package. Vendors put necessary files that are to be used/copied/ extracted during the update.

OTA

Over The Air

zip file

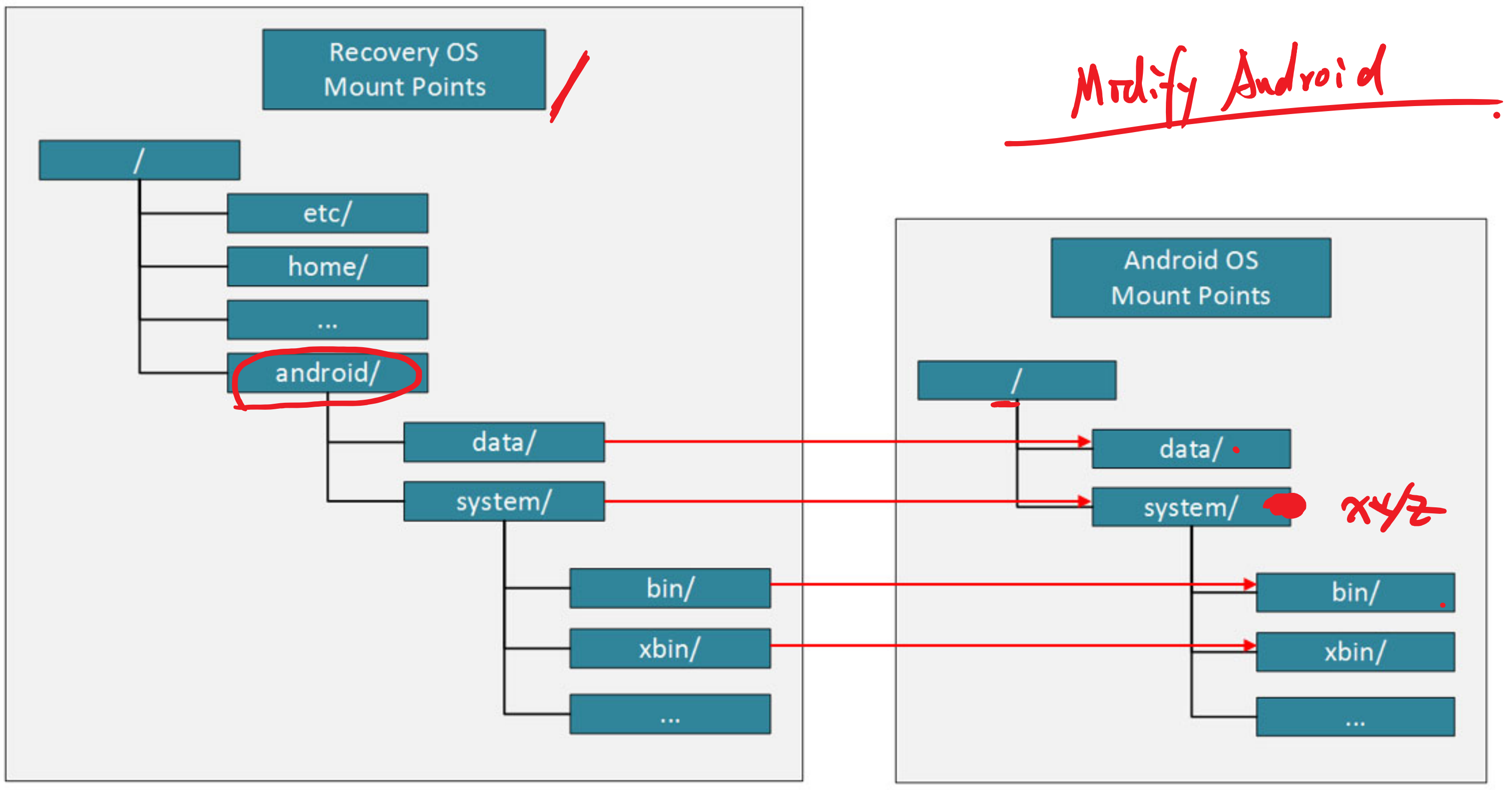
program (update)

Auto Run

Constructing OTA Package

Mount Points and OTA Script

Modify Android

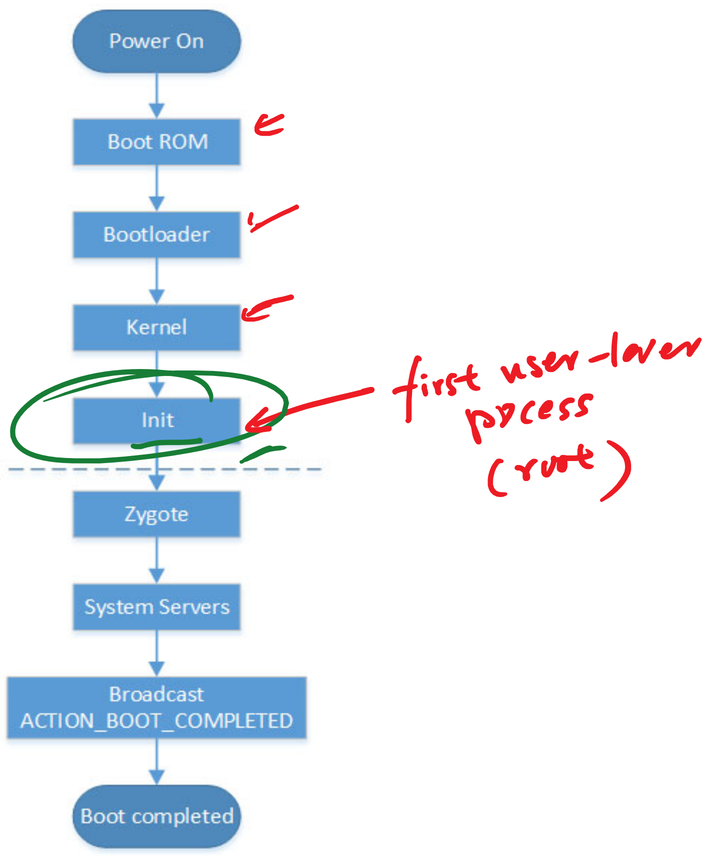


/android/system/xyz

/system/xyz

Inject Code via `Init.sh`

Objective: Update Android OS—create a dummy file during the Android Runtime bootup



creat

/system/xyz .

root

Build OTA package

invoke by init.sh

❖ **Create** `dummy.sh`

```
echo hello > /system/testfile
```

❖ **Modify** `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
```

❖ **Build the OTA package**

```
$ zip -r task1.zip task1/
```

/system/xbin/dummy.sh
→ return 0

Install and Execute OTA

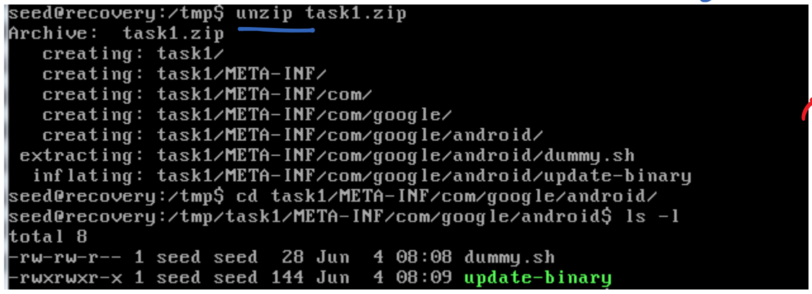
- **Power off the Android VM, and boot into the Recovery OS (holding the left-shift key during the booting) Get its IP Address using the "`ifconfig`" command.**
- **Copy the OTA package to the recovery OS (you need to change `10.0.2.10` to the IP of your recovery OS)**

```
$ scp task1.zip seed@10.0.2.10:/tmp
```

- **Go to the recovery OS, and Unzip the OTA package**

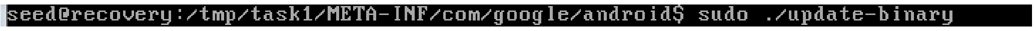
scp.

16.04 VM



```
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 task1/META-INF/com/google/android/
seed@recovery:/tmp/task1/META-INF/com/google/android$ ls -l
total 8
-rw-rw-r-- 1 seed seed  28 Jun  4 08:08 dummy.sh
-rwxrwxr-x 1 seed seed 144 Jun  4 08:09 update-binary
```

- **Emulate what recovery OS does: execute `update-binary`**



```
seed@recovery:/tmp/task1/META-INF/com/google/android$ sudo ./update-binary
```

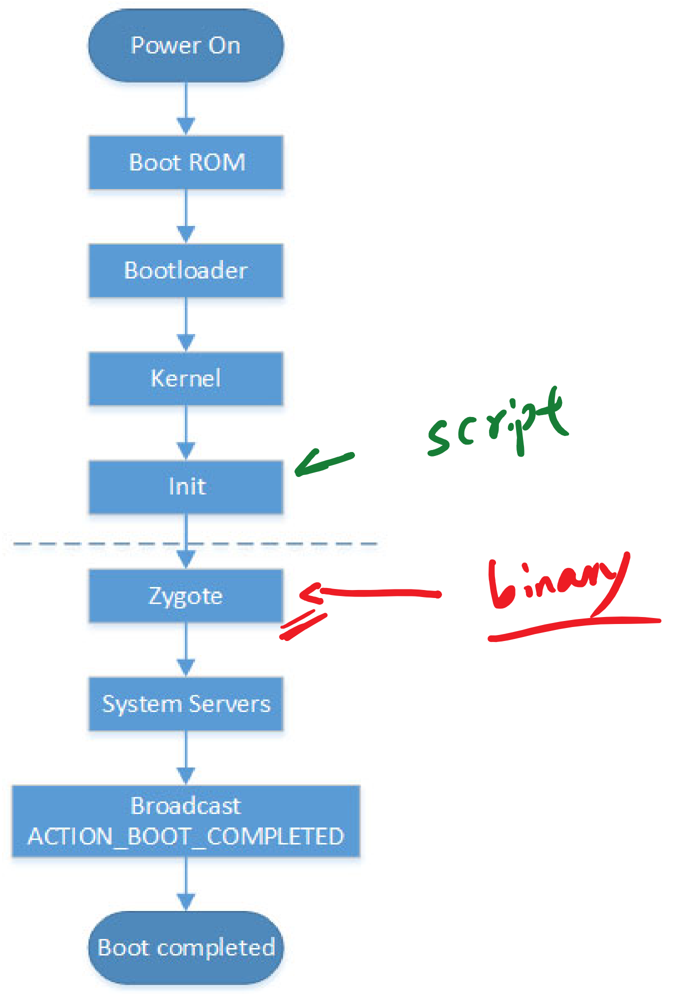
- **Reboot the Android VM and see results**

$ sudo reboot

Check whether a file called "`testfile`" is created inside the `/system` folder or not.

Inject Code via app_process

Objective: Update Android OS—create a dummy file during the Android Runtime bootup



Modified 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;
}
```

app_process

} creat dumy file

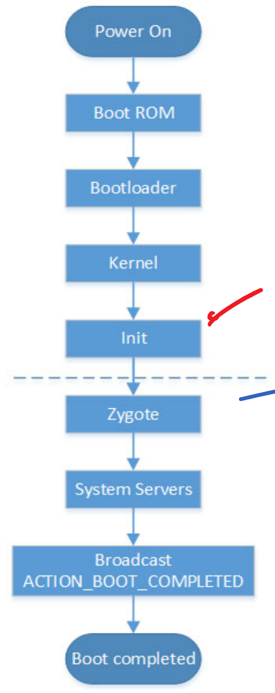
} run zygote

app_process

Get a Root Shell

Get a Root Shell

Objective: Update Android OS—enable users to get a root shell.



run /bin/bash (child process)

IN : root
OUT : root
ERR : root

X

① chmod 777 filename
/bin/bash < filename . > anotherfile

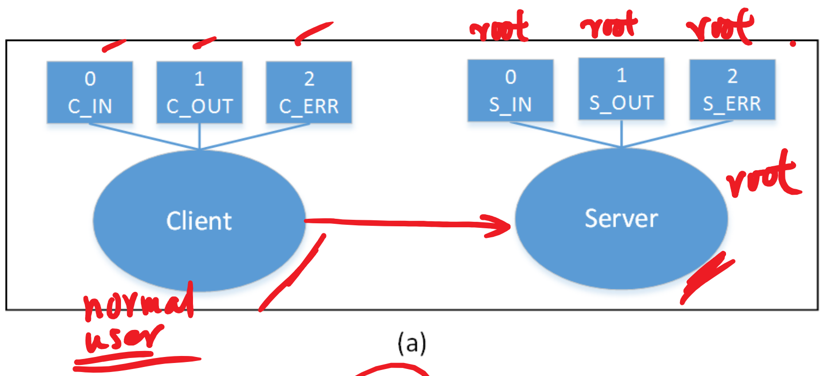
② reverse shell

③

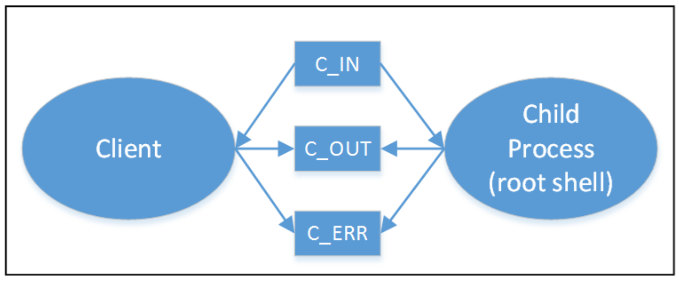
Get a Root Shell

Objective: Update Android OS: enable users to get a root shell.

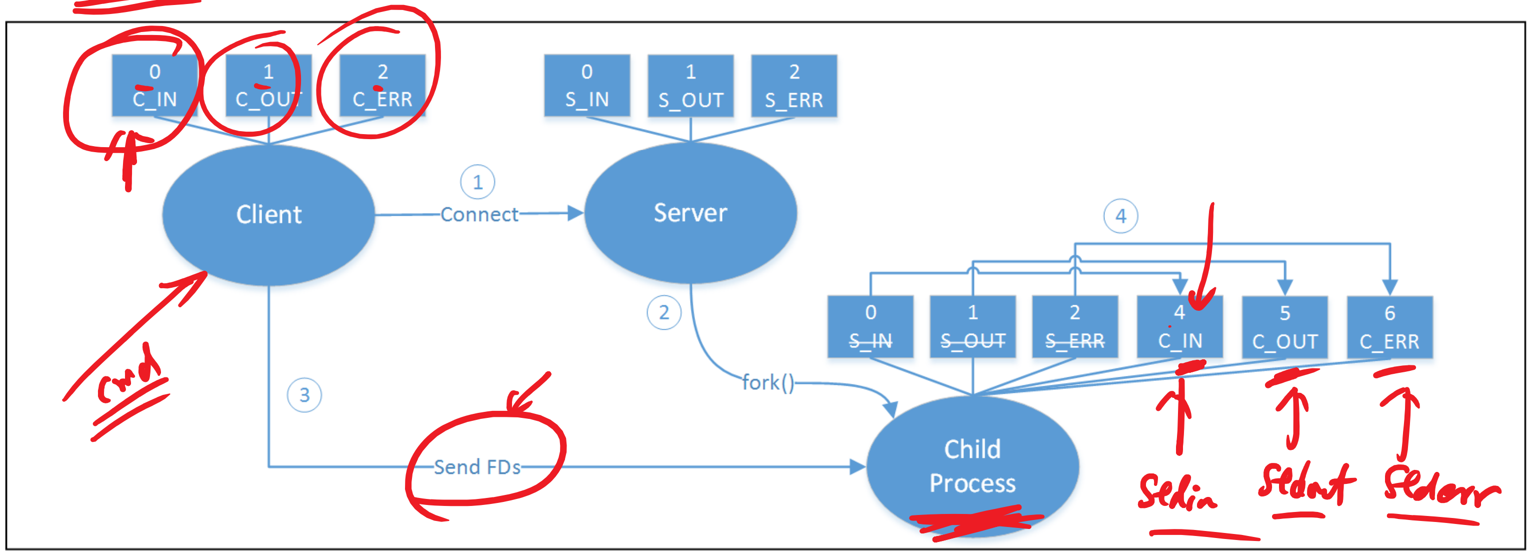
Get a Root Shell From a Root Daemon



(a)



(c)



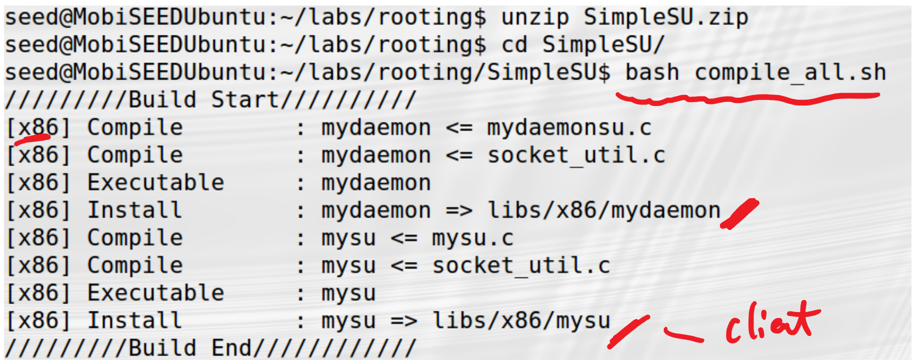
(b)

Unix Domain Socket

Task 3 Steps (Part I)

Cross compilation

- Download SimpleSU.zip from the lab web site
- Unzip it and compile the SimpleSU program

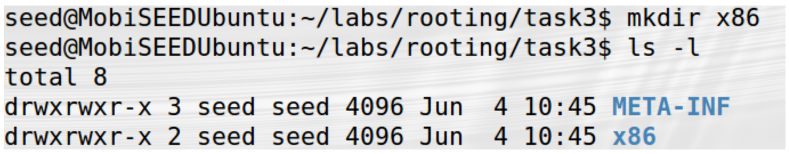


```
seed@MobiSEEDUbuntu:~/labs/rooting$ unzip SimpleSU.zip
seed@MobiSEEDUbuntu:~/labs/rooting$ cd SimpleSU/
seed@MobiSEEDUbuntu:~/labs/rooting/SimpleSU$ bash compile_all.sh
/////////Build Start//////////
[x86] Compile        : mydaemon <= mydaemonsu.c
[x86] Compile        : mydaemon <= socket_util.c
[x86] Executable     : mydaemon
[x86] Install        : mydaemon => libs/x86/mydaemon
[x86] Compile        : mysu <= mysu.c
[x86] Compile        : mysu <= socket_util.c
[x86] Executable     : mysu
[x86] Install        : mysu => libs/x86/mysu
/////////Build End////////////
```

← client

NDK

- Go to the task3 folder (the OTA folder), create a folder called x86



```
seed@MobiSEEDUbuntu:~/labs/rooting/task3$ mkdir x86
seed@MobiSEEDUbuntu:~/labs/rooting/task3$ ls -l
total 8
drwxrwxr-x 3 seed seed 4096 Jun  4 10:45 META-INF
drwxrwxr-x 2 seed seed 4096 Jun  4 10:45 x86
```

- Copy SimpleSU/mydaemon/libs/x86/mydaemon and SimpleSU/mysu/libs/x86/mysu to the task3/x86 folder

Task 3 Steps (Part II)

❖ **Go to the `task3/META-INF/com/google/android/` folder**

❖ **Construct update-binary**

- Run "`gedit update-binary`"
- Add the following lines to the file (copy and paste from Android_Rooting.txt from Piazza)

```
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
```

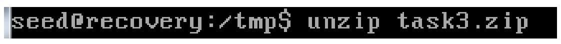
- Make `update-binary` executable

```
$ chmod a+x update-binary
```

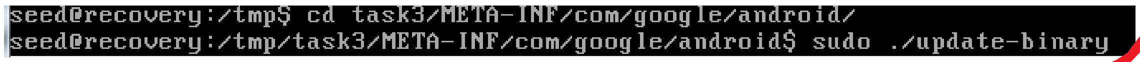
❖ **Build the OTA package (zip -r) and copy to the recovery OS (power it on first)**

```
$ zip -r task3.zip task3/
$ scp task3.zip seed@10.0.2.10:/tmp
```

❖ **Go to the recovery OS, unzip the OTA package, and do the update**



```
seed@recovery:/tmp$ unzip task3.zip
```



```
seed@recovery:/tmp$ cd task3/META-INF/com/google/android/
seed@recovery:/tmp/task3/META-INF/com/google/android$ sudo ./update-binary
```

❖ **Reboot the Android VM and see results**

Inside Android, click the terminal app, type "mysu" and see whether you get a root shell or not.

Code Details

❖ Start the root daemon

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

❖ Give the client access to the root shell process

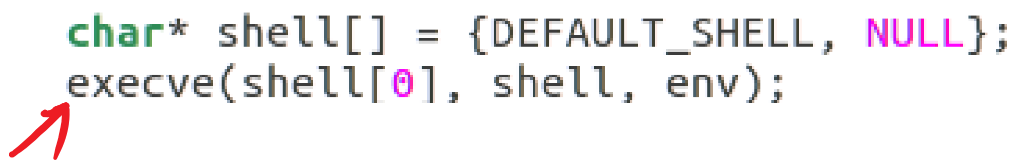
```
int client_in = recv_fd(socket);
int client_out = recv_fd(socket);
int client_err = recv_fd(socket);

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

//change current directory
chdir("/");

//construct essential environment variables
char* env[] = {SHELL_ENV, PATH_ENV};

char* shell[] = {DEFAULT_SHELL, NULL};
execve(shell[0], shell, env);
```



```
char* shell[] = {DEFAULT_SHELL, NULL};
execve(shell[0], shell, env);
```

Summary

- How rooting works
- How to use OTA to root Android devices