Introduction

What do you mean — install Android? I already *have* Android on my phone!

Yeah, well, ... actually we are going to install `CyanogenMod` (`CM`) in place of whatever was there when you bought the phone. `CM` is a version of Android created, maintained and continuously modified with tweaks and security fixes by the `CM community`[1].

An indeterminate amount of the information on the web is confused, incomplete, contradictory or out-of-date. Everything has to be verified, not because the procedure did not work when the blogger tried it, but because when they came to document it they left out some step or other. In this talk we shall try out procedures from the web and document one that works.

Three Cases Examined

You can get a good understanding of Android from the Android Hacker's Handbook[2]. I have direct experience of working with three devices, so for interest and completeness this is what we shall do:

Root an older Android smartphone: a Samsung Galaxy S Duos GT-S7562.

Install CM-11 on an older Android tablet: an ASUS Transformer Pad Infinity XF700T.

Install CM-12 and Upgrade it on a recent smartphone: a Motorola Moto G 2014 XT1068.

Why would you do this?

Lots of reasons:

- to get timely updates
- to get rid of bloatware
- to make readable backups

How things work on a mobile phone

Your mobile phone is a small computer.

If you purchase a laptop computer with a pre-installed operating system (OS) and want to change it you need to know: the laptop boots to a filesystem on a device and executes an `init` program; this can be an installer to the HDD. Typically the OS can be on an SD card, easily swapped for one of your choice.

If you purchase a mobile phone or tablet with a pre-installed OS and want to change it you need to know: the phone or tablet boots to a filesystem in ROM and executes an `init` program; this ROM needs replacing by one of your choice.

Because of this difference in the underlying engineering, replacing the OS is a little more complicated for the phone and tablet.

Host Computer Preliminaries

You'll need some extra tools. On my Debian *Jessie* linux computer I did this:

```
linux:~# apt-get install android-tools-adb android-tools-fastboot
linux:~$ which fastboot adb
/usr/bin/fastboot
/usr/bin/adb
```

[1]http://www.cyanogenmod.org/

[2]by Drake, Fora, Lanier *et al*, ISBN 978-1-118-60864-7, published by Wiley in 2014

Let's summarize the manuals for `fastboot` and `adb`.

fastboot

```
NAME fastboot - manipulate the non-volatile flash partitions
DESCRIPTION
fastboot is a program used to manipulate (list, install, erase) the non-volatile memory
such as flash filesystem partitions on devices that adhere to the fastboot protocol, via
a USB connection from a host computer. It requires that the device be started in a
boot loader mode with the fastboot protocol enabled. Once connected to the device, the
program accepts a specific set of commands sent to it via the USB using the fastboot
program on the host.

fastboot is primarily used for installing the operating system binary ‘images’ into the
non-volatile flash memory of the devices. The partition adheres to a specific layout.
fastboot is designed for use with phones & tablets running the Android operating system.
```

adb

```
NAME adb - android debug bridge
SYNOPSIS adb [ options ] command
DESCRIPTION adb is a command line tool that lets you communicate with a connected Android
device. It is a client-server program that includes three components: (1) a client, which
runs on your host machine. You can invoke a client from a shell by issuing an adb command;
(2) a server, which runs as a background process on your host machine. The server manages
communication between the client and the adb daemon running on a device;  (3) a daemon,
which runs as a background process on each device.

       For the full and up to date documentation visit:
       http://developer.android.com/guide/developing/tools/adb.html
```

Case 1: Root an Older Android Smartphone

The `Samsung Galaxy S Duos GT-S7562` was to be retired because I was sick of not getting prompt security updates. When the `StageFright` vulnerabilities[3] were exposed, I downloaded the `StageFright detector` program[4], ran it, and saw . . . `VULNERABLE` in red. I no longer wanted that. I also wanted to save its contacts and messages in a readable format. As we shall see, we only need to `root the phone`[5] to do this.

- boot the phone into "download" mode
- transfer the rooting program into ROM
- confirm the installation

Boot the phone into "download" mode

The less said about the frustrating procedure[6] using the `samsung ODIN` program the better. I advise you to make friends with someone who has an old `XP` system, or it will be you who are rooted, not your phone.

[3]http://www.androidcentral.com/stagefright
[4]https://play.google.com/store/apps/details?id=com.zimperium.stagefrightdetector
[5]install a suitable version of the linux 'switch user' or 'su' program
[6]http://theunlockr.com/2013/06/11/how-to-install-custom-recovery-and-root-the-samsung-galaxy-s-duos-s7562/

They can perform most of this procedure on their Windows computer much better than a Linux hacker can. In any case, you'll find that it is physically tricky to get the timing right, but after about ten attempts it eventually worked for me.

Transfer the rooting program to the phone

Continue to follow the procedure mentioned above, from the XP computer.

Confirm the installation was successful

- First, make sure you can use the USB cord for access:

  ```
  get developer options menu [tap on build number 4-7 times]
  open developer options
  tap ok to warning
  tick usb debugging
  ```

- Check that the phone is recognised when connected with a USB cord:

  ```
  linux:~$ adb devices
  * daemon not running. starting it now on port 5037 *
  * daemon started successfully *
  List of devices attached
      35da7a4b   device
  ```

- Then find out where the `su` program has been installed:

  ```
  linux:~$ adb shell
  shell@android:/ $ echo $PATH
      /sbin:/vendor/bin:/system/sbin:/system/bin:/system/xbin
  shell@android:/ $ ls -l /system/xbin
      -rwsr-sr-x root     root        91980 2016-02-02 23:27 su
  ```

- And finally try it out.

  ```
  shell@android:/ $ su
  shell@android:/ $ whoami
  shell@android:/ $ shell
  ```

- Oops! Not quite `root` yet. Go to `settings`, `Developer options`, `Root access`. Select `Apps and ADB`. Return to terminal.

  ```
  shell@android:/ $ su
  root@android:/ #
  ```

  Seems to be installed OK.

- Now go back to settings and disable `Root access` for security ...

Copying Contacts and Messages

Now let's use `su` to get at our data. For security, contacts and messages are not readable by an ordinary user, so to get at them we need root permission. But, for the same reason, we cannot run a root shell from outside. However, we always have universal read/write access to the /sdcard directory (with no need for an actual card). So here is the trick[7].

[7] http://stackoverflow.com/questions/12266374/backup-full-sms-mms-contents-via-adb

- connect phone to linux, start a user shell and then become root:

```
linux:~$ adb shell
shell@android:/ $ su
shell@android:/ #
```

- copy data from location on phone to sdcard directory using the `cat` command, as there appears to be no `cp` command in this phone:

```
shell@android:/ #
  cat /data/data/com.android.providers.telephony/databases/mmssms.db \
      >/sdcard/mmssms.db
  cat /data/data/com.android.providers.contacts/databases/contacts2.db \
      >/sdcard/contacts2.db
shell@android:/ # ls -l /sdcard
  -rw-rw-r-- root sdcard_rw 585728 2015-11-06 12:58 contacts2.db
  -rw-rw-r-- root sdcard_rw 167936 2015-11-06 13:00 telephony.db
```

- quit both shells and copy data from sdcard back to the linux computer:

```
shell@android:/sdcard # exit
shell@android:/ $ exit
linux:~$ adb pull /sdcard/mmssms.db .
    6940 KB/s (585728 bytes in 0.082s)
linux:~$ adb pull /sdcard/telephony.db .
    3828 KB/s (167936 bytes in 0.042s)
linux:~$ file mmssms.db contacts2.db
    mmssms.db:    SQLite 3.x database, user version 58
    contacts2.db: SQLite 3.x database, user version 629
```

These two files containing your contacts and messages may now be explored using `sqlite`[8],[9].

Case 2: Install CM-11 on an older Android Tablet

I have used an `ASUS Transformer Pad Infinity XF700T` for some years. We essentially follow this procedure[10]:

1. unlock the tablet
2. install a recovery program
3. install version CM-11
4. confirm the installation

[8]http://stackoverflow.com/questions/11455164/how-to-install-sqlite-on-debian

[9]http://www.sitepoint.com/getting-started-sqlite3-basic-commands/

[10]https://wiki.cyanogenmod.org/w/Install_CM_for_tf700t

unlock the tablet

For reasons best known to themselves, ASUS provides a procedure for unlocking this particular tablet.

```
unlocked ok from asus [follow proc.]
no need to root the phone
linux:~$ file UnLock_app_V8.apk
UnLock_app_V8.apk: Java archive data (JAR)
linux:~$ jar tvf UnLock_app_V8.apk
  2771 Thu Dec 22 14:22:18 ACDT 2011 META-INF/MANIFEST.MF
  2813 Thu Dec 22 14:22:18 ACDT 2011 META-INF/CERT.SF
  1690 Thu Dec 22 14:22:18 ACDT 2011 META-INF/CERT.RSA
  3212 Thu Dec 22 14:22:18 ACDT 2011 AndroidManifest.xml
 32772 Thu Dec 22 14:22:18 ACDT 2011 classes.dex
       ...
       ...
329276 Thu Dec 22 14:22:18 ACDT 2011 resources.arsc
```

On booting, you'll see very small print in the top left corner which says `The Device is UnLocked`.

install a recovery program

I installed this recovery image[11].

```
linux:~$ fastboot -i 0xb05 flash recovery twrp-2.8.7.0-tf700t.img
sending 'recovery' (7372 KB)...
OKAY [  2.824s]
writing 'recovery'...
OKAY [  2.204s]
finished. total time: 5.028s
```

install version CM-11

```
linux:~$ adb push cm-11-20151108-NIGHTLY-tf700t.zip /sdcard/
1078 KB/s (220266276 bytes in 199.375s)

linux:~$ fastboot -i 0x0B05 flash staging ww_bootloaderblob
sending 'staging' (1576 KB)...
OKAY [  0.338s]
writing 'staging'...
OKAY [  9.234s]
finished. total time: 9.571s

linux:~$ fastboot reboot
rebooting...
```

install suitable version of gapps

I chose the `mini` version of `gapps` so it would fit easily on the tablet.

[11] https://dl.twrp.me/tf700t/twrp-2.8.7.0-tf700t.img

```
linux:~$ adb push open_gapps-arm-4.4-mini-20151115.zip /sdcard/
1302 KB/s (231439445 bytes in 173.512s)
linux:~$ adb reboot recovery
```

Now install *via* the `teamwin` recovery program you installed before ...and success!! You get playstore access, phone, sms, etc.

confirm the installation

Run `StageFright` detector ... NOT VULNERABLE in green. Check `settings` – `about tablet` – `cyanogenmod version` 11-2016-0111-NIGHTLY-tf700t – `Android security patch level` january 1st, 2016.

Check root access: tap `settings` – tap `about phone` – tap tap `build number` about 7 times until you see `you have enabled developer options` – tap `settings` – tap `developer options` – tap `root access` – tap `Apps and ADB`. Now connect tablet to linux computer by the original USB cable.

```
linux:~$ adb devices
List of devices attached
019d376e4456660d device
linux:~$ adb shell
shell@tf700t:/ $ su
Permission denied
1|shell@tf700t:/ $ su
root@tf700t:/ # whoami
root
root@tf700t:/ #
```

Seems OK.

Case 3: Install CM-12 and Upgrade it on a Recent Smartphone

The `Motorola Moto G (2nd generation)` was selected because: it is very cheap ($239); it is unlockable (*q.v.*); it was on sale at Dick Smith; and it is well supported for this sort of endeaver, both by `Motorola` and by the user community.

Installing Procedure

In what follows I go through the procedure[12] to make sure it works and highlight important details. When the `StageFright` vulnerabilities[13] were exposed, I downloaded the `StageFright detector` program[14], ran it, and ...saw `VULNERABLE` in red. I no longer wanted that.

The procedure is:

- connect via USB
- prepare for unlocking
- unlock the phone
- reboot to recovery

[12]For installation, see https://wiki.cyanogenmod.org/w/Install_CM_for_titan
[13]http://www.androidcentral.com/stagefright
[14]https://play.google.com/store/apps/details?id=com.zimperium.stagefrightdetector

- root the phone
- install recovery program and then install cyanogenmod via recovery program
- install gapps via recovery
- reboot to system via recovery

connect via USB Connect your phone by USB to linux and check that all is recognised by running:

```
linux:~# /usr/bin/fastboot devices
ZC098A548E fastboot
```

prepare for unlocking To deal with your phone you need to boot it into a special state instead of the normal boot state. You also need to tell it to run a daemon that can attend to your requests.

All phones are different. On the `Moto G-2` you first arrange for a daemon to deal with your requests. Go to `settings` and scroll to `About phone`; go to the bottom item `Build number`; tap this item a few times until it says `you are now a developer`. When this happens, a new menu item appears under `Settings: Developer options`. Go into this and tick `USB debugging` and answer `OK` to the warning notice.

Power the phone off. Now you can boot it into the `fastboot` mode by holding the volume-down whilst powering on. You will see some text saying `AP Fastboot Flash Mode (S)` and so on.

In yellow will show `Device is LOCKED. Status Code: 0.`

Once there you can connect a USB cable *via* the micro-USB slot in the phone to your laptop. You should see `USB Connected` on the phone.

Check that all is recognised by running:

```
linux:~$ /usr/bin/fastboot devices
ZC098A548E fastboot
```

unlock the phone Now go to Motorola's unlocking portal[15], and follow the instructions (they work fine).

```
linux:~# /usr/bin/fastboot oem get_unlock_data
(bootloader) 0A40040192024205#4C4D3556313230
(bootloader) 30373731363031303332323239#BD00
(bootloader) 8A672BA4746C2CE02328A2AC0C39F95
(bootloader) 1A3E5#1F53280002000000000000000
(bootloader) 0000000.
OKAY [ 0.136s]
finished. total time: 0.136s
```

Paste your unlock data (sample shown below) but put it all in one long line, into the unlock portal, and receive your unlock code by email from Motorola.

```
UNLOCK DATA:
0A40040192024205#4C4D3556313230
30373731363031303332323239#BD00
8A672BA4746C2CE02328A2AC0C39F95
1A3E5#1F53280002000000000000000
0000000
```

[15] motorola-global-portal.custhelp.com/app/standalone/bootloader/unlock-your-device-a

reboot to recovery `linux:~# adb reboot bootloader`

```
UNLOCK CODE: C292BCCD14F018F6943D
linux:~# fastboot oem unlock C292BCCD14F018F6943D
OK
```

The phone is now unlocked. When it reboots you should see a warning screen `WARNING: BOOTLOADER UNLOCKED`.

root the phone Go to a site with a good version of the `su` (switch user) command[16] and download the version[17] for the `Moto G`. This works for Android 5.0, which was on my phone when I bought it.

```
# /usr/bin/fastboot boot image/CF-Auto-Root-titanumtsds-titanretaildsds-xt1068.img
downloading 'boot.img'...
OKAY [ 0.399s]
booting...
OKAY [ 0.559s]
finished. total time: 0.958s
```

Power the phone off. Boot it into the `fastboot` mode by holding the volume-down whilst powering on. This time you will see some text saying `AP Fastboot Flash Mode (S)` and so on. In yellow will show `Device is UNLOCKED. Status Code: 3`. The next time it boots normally you'll get a white screen `WARNING: BOOTLOADER UNLOCKED, WARRANTY VOID` which stays there for at least half a minute.

install recovery program then install cyanogenmod via recovery program We shall replace the `Motorola` OS with `CyanogenMod`. There is a very comprehensive list of phones, models and suitable versions at this `CyanogenMod` download site[18].

Let's get a recent stable version for our phone. If we look down the left hand side, we come to `Motorola` and find the model `Moto G 2014 (titan)`. Selecting that we go to the download page[19]. Here we find a stable snapshot of the version[20] and recovery image[21], both of which we need, along with SHA-1 checksums. The link to the `wiki` gives more information about what we are doing.

Download these two files (253MB and 9MB), created on 2015-10-07, from the site[22] onto our linux computer.

Transfer both files to phone:

```
linux:~$ adb push cm-12.1-20151007-SNAPSHOT-YOG4PAO338-titan.zip  /sdcard/
linux:~$ adb push cm-12.1-20151007-SNAPSHOT-YOG4PAO338-titan-recovery.img /sdcard/
```

(this takes about one minute) Now install both, recovery first:

```
linux:~# adb reboot bootloader
linux:~# fastboot devices
ZC098A548E fastboot
linux:~# fastboot flash recovery cm-12.1-20151007-SNAPSHOT-YOG4PAO338-titan-recovery.img
```

[16] http://download.chainfire.eu/603/CF-Root/CF-Auto-Root/
[17] CF-Auto-Root-titanumtsds-titanretaildsds-xt1068.zip
[18] https://download.cyanogenmod.org/
[19] https://download.cyanogenmod.org/?device=titan
[20] cm-12.1-20151007-SNAPSHOT-YOG4PAO338-titan.zip
[21] cm-12.1-20151007-SNAPSHOT-YOG4PAO338-titan-recovery.img
[22] https://download.cyanogenmod.org/get/jenkins/129282/

Now install CM-12: Hold Volume Down and Power simultaneously. On the next screen use Volume Down to scroll to recovery and then use Volume Up to select. In Team Win Recovery Project, select menu choices by tapping on the appropriately labelled button. Select Wipe and then Factory Reset. Select Install. Navigate to /sdcard and select the CyanogenMod .zip package. Follow the on-screen notices to install the package.

install gapps Transfer gapps to phone:

```
linux:~$ adb push open-gapps-arm-6.0-nano-20151212.zip /sdcard/
```

(this takes about three minutes)

install gapps via recovery similar to above for CM-12.

reboot to system via recovery

This version `CyanogenMod 12.1` needs extra work because the GPS requires a signal lock to function correctly. According to this blog[23], to get the GPS to work, simply add:

```
linux:~# fastboot erase modemst1
linux:~# fastboot erase modemst2
```

Now Let's Explore the Phone a bit

Connect the `moto g` to linux via USB to see the commands available on the phone, by using the `abd` program:

```
linux:~$ adb devices
List of devices attached
ZX1D23JPLK device
linux:~$ adb shell
shell@titan_umtsds:/ $ su root
[this will ask you to grant root; you have to physically tap on the phone itself]
root@titan_umtsds:/ #
```

First, list the shell built-in commands:

```
root@titan_umtsds:/ # help help -a|grep usage [and extract the second field]
acpi base64 basename blkid blockdev bzcat cal cat chattr chcon chgrp chmod
chown chroot cksum cmp comm cp cpio cut date dd df diff dirname dmesg dos2unix
du disk echo grep env expand expr fallocate fdisk grep find flock free
freeramdisk fsfreeze fstype ftpget ftpput ftpget ftpput getenforce getprop grep
groups head help usage host hostname hwclock id ifconfig inotifyd insmod
install ionice iorenice kill killall ln load_policy logname losetup ls lsattr
lsmod lsof lsusb makedevs md5sum mkdir mkfifo mknod mkswap mktemp modinfo more
mount mountpoint mv nbd-client nbd-client netcat netcat netstat nice nl nohup
nproc od partprobe paste patch pgrep pidof pivot_root pkill pmap printenv
printf ps pwd pwdx readahead readlink realpath renice reset resize restorecon
rev rm rmdir rmmod route runcon sed seq setenforce setprop setsid sha1sum sleep
sort split stat strings swapoff swapon switch_root sync sysctl tac tail tar
taskset tee telnet test time timeout top toybox show touch tr traceroute
traceroute truncate tty umount uname uniq unix2dos uptime usleep vconfig vmstat
watch wc which logname xargs xxd xzcat yes
```

[23] http://forum.xda-developers.com/showpost.php?p=58376017&postcount=731

Second, list all the other commands:

```
root@titan_umtsds:/ # echo $PATH
PATH=/sbin:/vendor/bin:/system/sbin:/system/bin:/system/xbin
root@titan_umtsds:/ # for d in /sbin /vendor/bin /system/sbin /system/bin /system/xbin
> do
> [ -d $d ] && ls $d
> done
adbd healthd mkfs.f2fs ueventd watchdogd ATFWD-daemon PktRspTest StoreKeybox
[ adsprpcd am ap_gain.bin ap_gain_mmul.bin apanic_annotate.sh aplogcat aplogd
app_process app_process32 app_process32_original app_process_init applypatch
appops appwidget atrace batch batt_health bcc blkid bmgr bootanimation brctl
btnvtool bu bug2go-bugreport bug2go-bugreport-oem bugreport cat charge_only_mode
charger_monitor chcon checknmount chmod chown clatd clear cmp cnd content cp
cplay curl dalvikvm dalvikvm32 date dbvc_atvc_property_set dd debuggerd dex2oat
df dhcpcd diag_callback_client diag_dci_sample diag_klog diag_mdlog diag_mdlog-getlogs
diag_mdlog-wrap diag_socket_log diag_uart_log dmesg dnsmasq dpm drmserver dropboxd
dropboxhelper du dumpstate dumpsys e2fsck ebtables esdpll fm_qsoc_patches fmconfig
fmfactorytest fmfactorytestserver fsck.f2fs fsck_msdos ftmdaemon ftmipcd getconfig
getenforce getevent getprop getsebool gpsone_daemon grep gzip hardware_revisions.sh
hci_qcomm_init hd hostapd hvdcp id idmap ifconfig iftop ime ims_rtp_daemon imsdatadaemon
imsqmidaemon input insmod install-recovery.sh install-recovery_original.sh installd
ioctl ionice ip ip6tables ipInfo iptables irsc_util isdbtmmtest keymaster_test keystore
kill kpgather linker lmkd ln load_policy location-mq log logcat logd logwrapper ls
lsmod lsof make_ext4fs masterclear mbm_spy mcStarter md5 md5sum mdnsd media mediaserver
mkdir mknod mkswap mm-qcamera-app mm-qcamera-daemon mm-qjpeg-dec-test mm-qjpeg-enc-test
mm-qomx-idec-test mm-qomx-ienc-test mm-vdec-omx-test mm-venc-omx-test720p
mm-video-driver-test mm-video-encdrv-test monkey moto_com.sh motobox mount mount_ext4.sh
mpdecision mtpd mv n_smux nandread ndc netcfg netd netmgrd netstat newfs_msdos nohup
notify oatdump patchoat ping ping6 pm pppd printenv ps ptf ptt_socket_app qmi_motext_hook
qmuxd qrngd qrngp qrngtest qseecom_sample_client qseecom_security_test qseecomd racoon
radish readlink reboot renice requestsync resize2fs restorecon rfs_access rild rm rmdir
rmmod rmt_storage route run-as runcon schedtest schedtop screencap screenrecord sdcard
sendevent sendevent2 sensord sensorservice service servicemanager setconfig setenforce
setfattr setprop setsebool settings setup_fs sh simg2img sleep smd stacker start stop
subsystem_ramdump surfaceflinger svc swapoff swapon sync tc tcmd tcmdhelp test test_diag
thermal-engine time_daemon timedexec tinycap tinymix tinypcminfo tinyplay toolbox top
touch uiautomator umount uncrypt updater uptime vdc vmstat vold watchprops wcnss_filter
wcnss_service wdsdaemon wipe wm wpa_supplicant xtwifi-client xtwifi-inet-agent bttest
daemonsu dexdump su sugote sugote-mksh supolicy
127|root@titan_umtsds:/ # reboot
```

To find out how to use a command, you may use `help commandname` instead of the unavailable `man commandname`. For example, to see options for the `tr` command:

```
shell@titan:/ $ help tr
usage: tr [-cds] SET1 [SET2]
Translate, squeeze, or delete characters from stdin, writing to stdout
-c/-C  Take complement of SET1
-d     Delete input characters coded SET1
-s     Squeeze multiple output characters of SET2 into one character
```

Upgrading Procedure

Upgrading to a new release of CM is a breeze. To update to the latest nightly build, we:

- check for latest update;
- install latest update;
- confirm updated details.

check for latest update

Connect your phone to the internet *via* your home modem. Tap `settings`; tap `about phone`; tap `CyanogenMod updates`. Check for update by tapping the `reload circle icon`. Available update should then show [eg] `cm-13.0-20160206-NIGHTLY (new)`. Download it to phone by tapping the download icon to its right. The `CM Updater` downloads the latest version (about 250MB, takes about 25 mins).

install latest update

When it reports `Downloaded` to phone, tap the oblong download icon. It shall warn you are about to `Apply update`; tap `UPDATE`. It reboots to recovery mode; you'll see `Warning: bootloader unlocked` warning. You'll see the blue `teamwin` splash screen running `openrecovery` script; this all happens automatically (takes a few minutes). Then it reboots; you'll see the CM face icon for a few minutes; see `Android is upgrading` as it optimizes all your apps. Finally the home screen appears ... All Done!

confirm updated details

To check status: tap `settings` – tap `about phone` – see [eg] `Android version 6.0.1` – see [eg] `CyanogenMod version 13.0-20160206-NIGHTLY-titan` – see [eg] `Android security patch level February 1, 2016`. Run the `StageFright detector` program and see `NOT VULNERABLE` in green.

Can make `phone calls`, `SMS`, browse `web`, access `playstore`, use `GPS` . . . configure the system.

Quite wonderful, indeed!

Enjoy a better Android!