Root Your T-Mobile LG V20 Using Dirty COW

- BY KEVIN M.
- 12/24/2016 10:35 PM

It's been little over a month since the official release of LG's latest flagship phone, the LG V20. Despite a few minor hiccups, the V20 has been attracting attention from all over for being an amazing phone. But like with most Android phones, there's no better feeling than rooting and taking complete ownership of it.

Unfortunately, unlike most LG phones, the V20 can't be rooted using the traditional method of flashing a stock TOT or KDZ file via LGUP, which means we'll have to use alternative means to achieve Superuser status.

Luckily, though, a new method revolving around the Dirty COW exploit can be used to root the T-Mobile (H918) variant of the LG V20. It'll take a lot of work, and you'll need to make sure to follow every step carefully, but we've got the process covered in detail below.

Don't Miss: Why Linux Kernel Exploits Like Dirty COW Usually Help Android Users More Than They Hurt

Requirements

- T-Mobile LG V20 model H918 (method won't work for other variants)
- Windows, Mac, or Linux computer
- USB data cable

Before We Begin

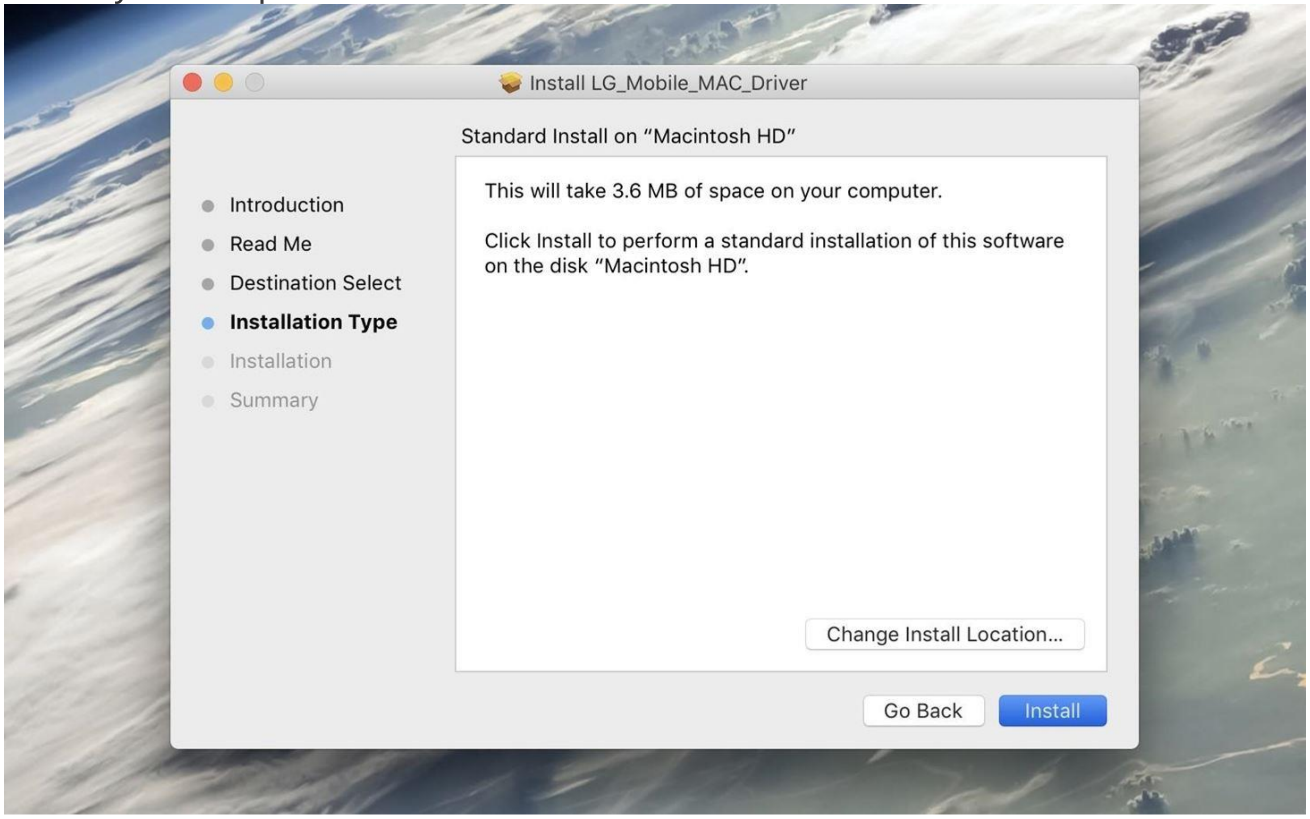
First up, make sure to back up your phone's data before using this method, as **it will wipe all data on your device**. Most of your apps will be restored when you log back in with your Google account after rooting, but personal files like your photos and MP3 library should be transferred to an external storage drive before you begin.

This method utilizes ADB and Fastboot commands, so we'd recommend that you have at least a basic knowledge of these two tools before proceeding. If not, at least make sure to follow the steps perfectly, otherwise you might risk bricking your device.

Finally, we have to give a big thanks to XDA Recognized Developer James Addunono for modifying the Dirty COW exploit script to work with the LG V20, and for compiling TWRP, which makes rooting the V20 possible.

Step 1 Install Needed LG Drivers

To start, install the LG drivers for Windows or Mac, as this will allow your phone and computer to communicate with one another. When you're done with that, restart your computer to make sure the drivers are active.



Standard Install on "Macintosh HD"

This will take 3.6 MB of space on your computer.

Click Install to perform a standard installation of this software on the disk "Macintosh HD".

Installing the LG drivers on a Mac.

Step 2 Install ADB & Fastboot

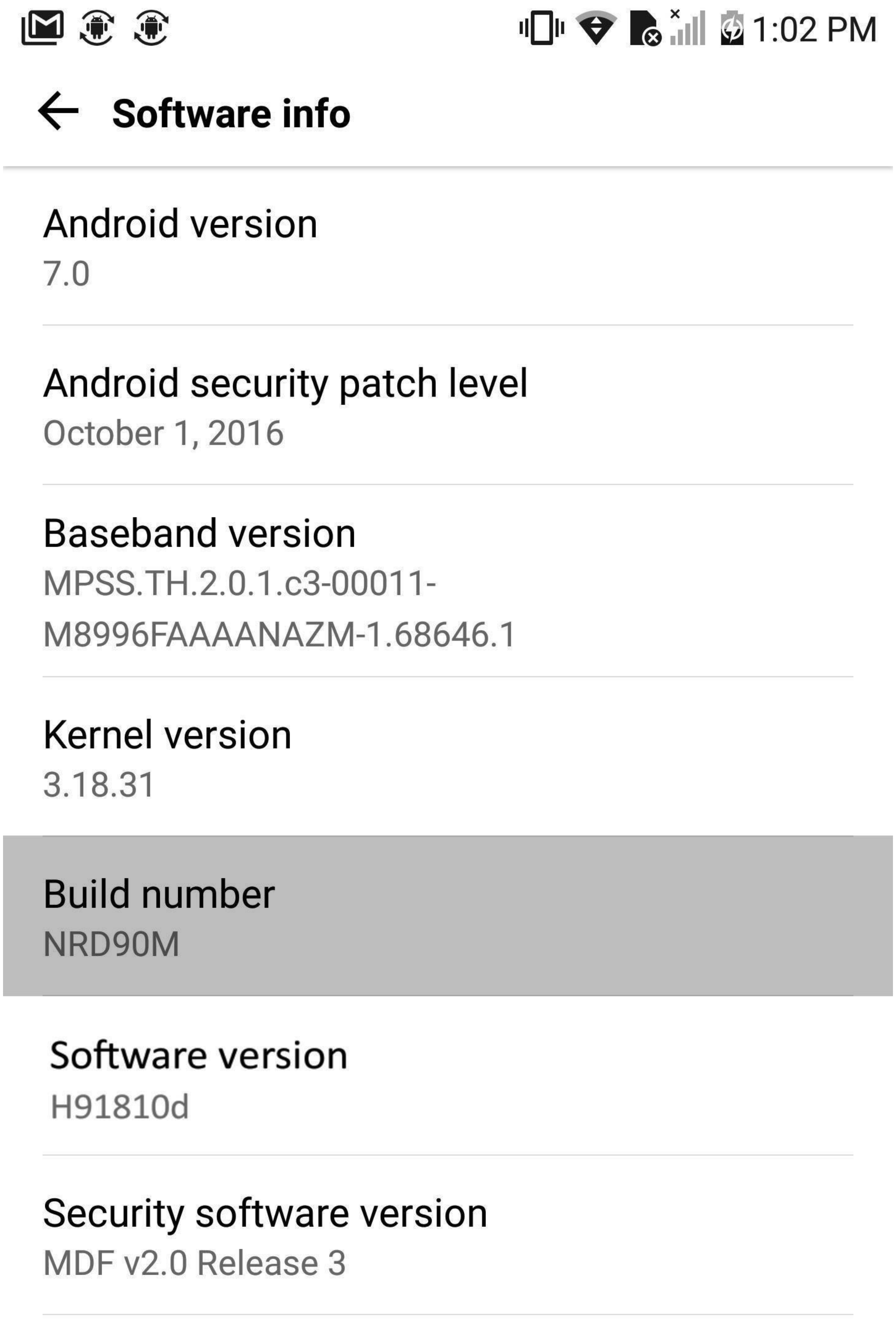
Next up, you'll need to install ADB and Fastboot on your computer. We recommend downloading the official version from Google, and we've outlined that process in Method 1 at the following link:

Don't Miss: How to Install ADB & Fastboot on Windows, Mac, or Linux

Step 3 Enable Developer Settings

After that, you'll have to enable the hidden "Developer options" menu on your phone. To do that, navigate to Settings -> About phone -> Software info, then tap the *Build number* entry seven times in rapid succession.

Don't Miss: How to Enable Developer Options on Your Phone or Tablet



1:02 PM

Software info

Android version
7.0

Android security patch level
October 1, 2016

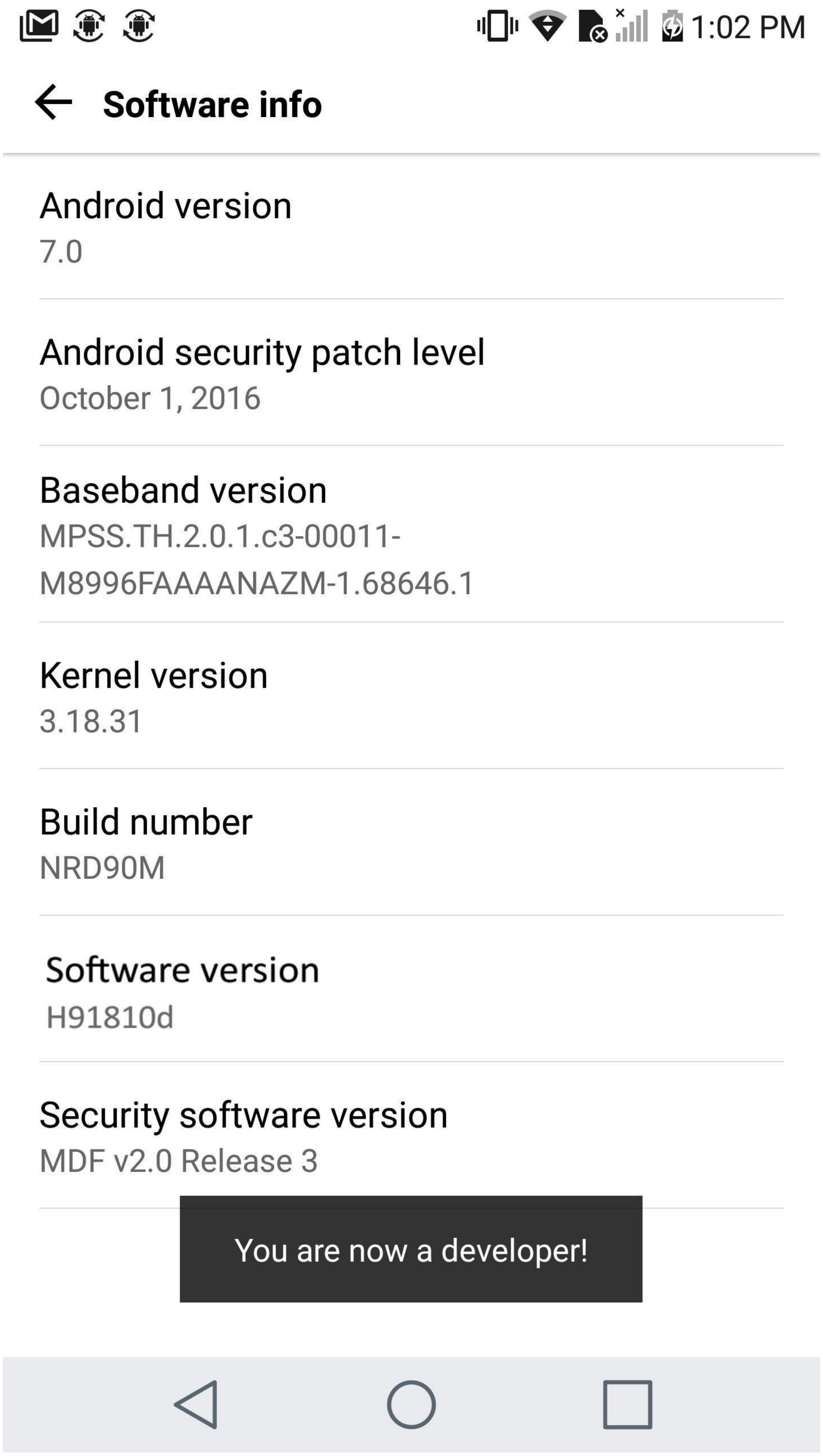
Baseband version
MPSS.TH.2.0.1.c3-00011-M8996FAAAANAZM-1.68646.1

Kernel version
3.18.31

Build number
NRD90M

Software version
H91810d

Security software version
MDF v2.0 Release 3



1:02 PM

Software info

Android version
7.0

Android security patch level
October 1, 2016

Baseband version
MPSS.TH.2.0.1.c3-00011-M8996FAAAANAZM-1.68646.1

Kernel version
3.18.31

Build number
NRD90M

Software version
H91810d

Security software version
MDF v2.0 Release 3

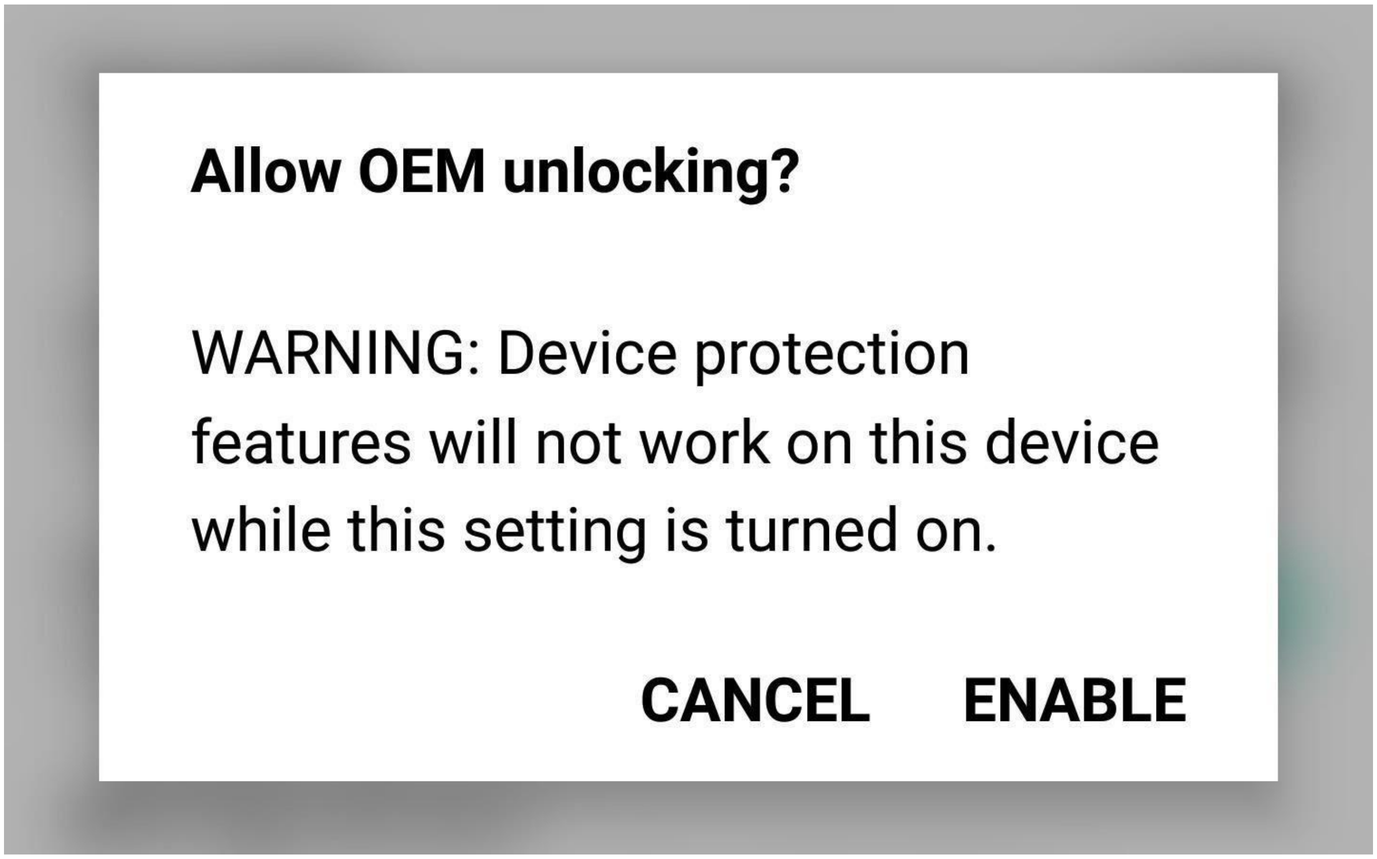
You are now a developer!

Enabling Developer options.

Step 4Enable OEM Unlocking

Now go to Settings -> Developer options, then turn on the "OEM unlocking" option, and press "Enable" when prompted. Do not turn this off while using a custom ROM or a custom recovery, as it will result in a loss of your data.

Don't Miss: PSA: Enable This Hidden Setting Before Modding Anything on Android



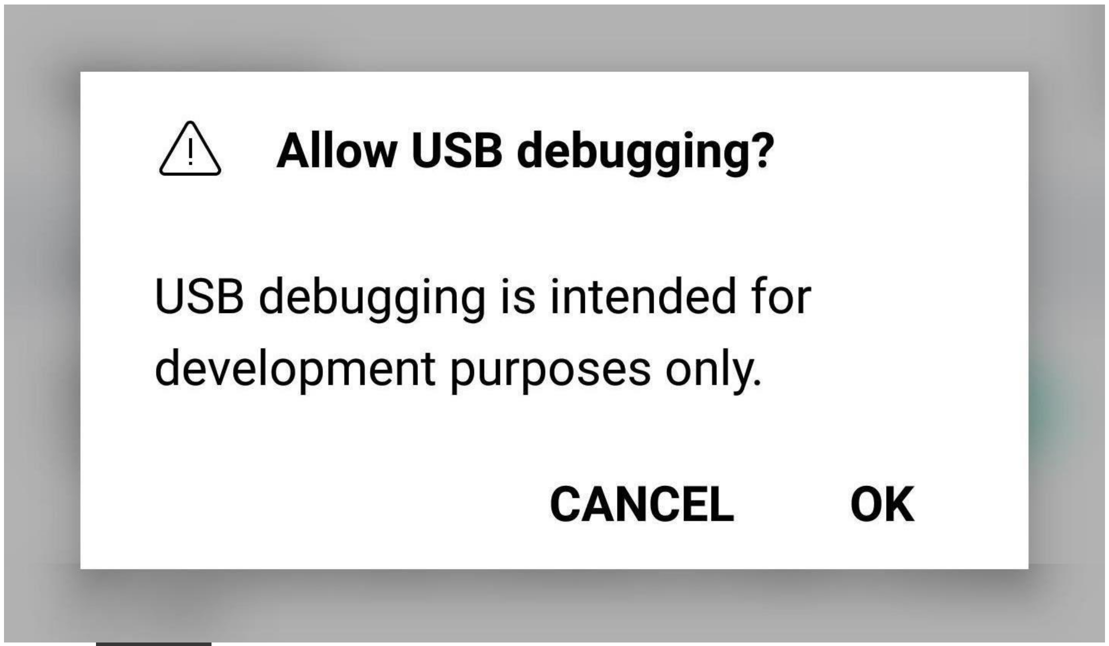
Allow OEM unlocking?

WARNING: Device protection features will not work on this device while this setting is turned on.

Enabling the "OEM Unlocking" setting in the Developer options menu.

Step 5Enable USB Debugging

When you're done there, scroll down just a little bit and enable the "USB debugging" setting in the same menu.

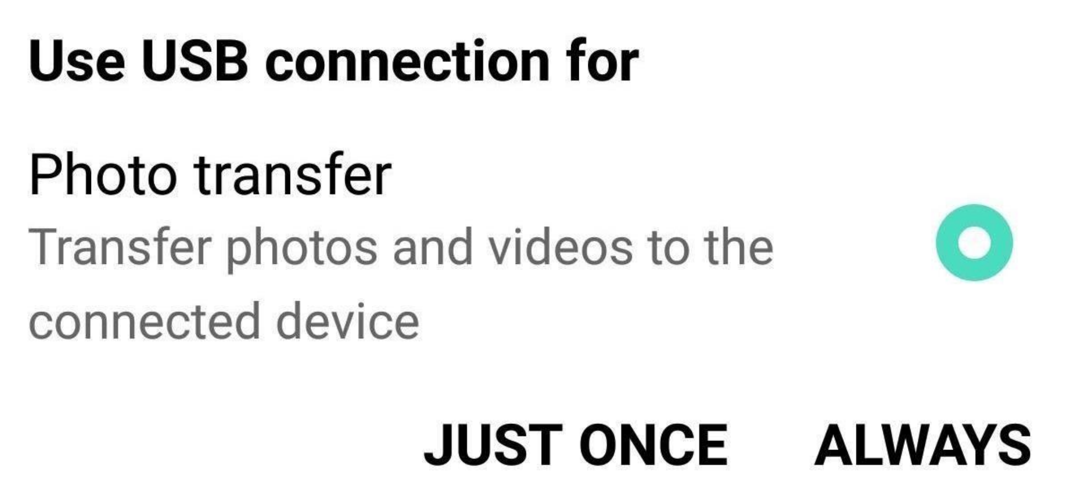


Allow USB debugging?

USB debugging is intended for development purposes only.

Step 6Connect Your Phone to Your Computer

Next up, connect your phone to your computer with a USB data cable. You might need to put the phone into PTP mode (for transferring images), as LG seems to have messed up their driver configurations. To do that, tap the *USB Computer Connection* notification that appears after plugging in your phone, then select "Photo transfer" from the popup.



Use USB connection for

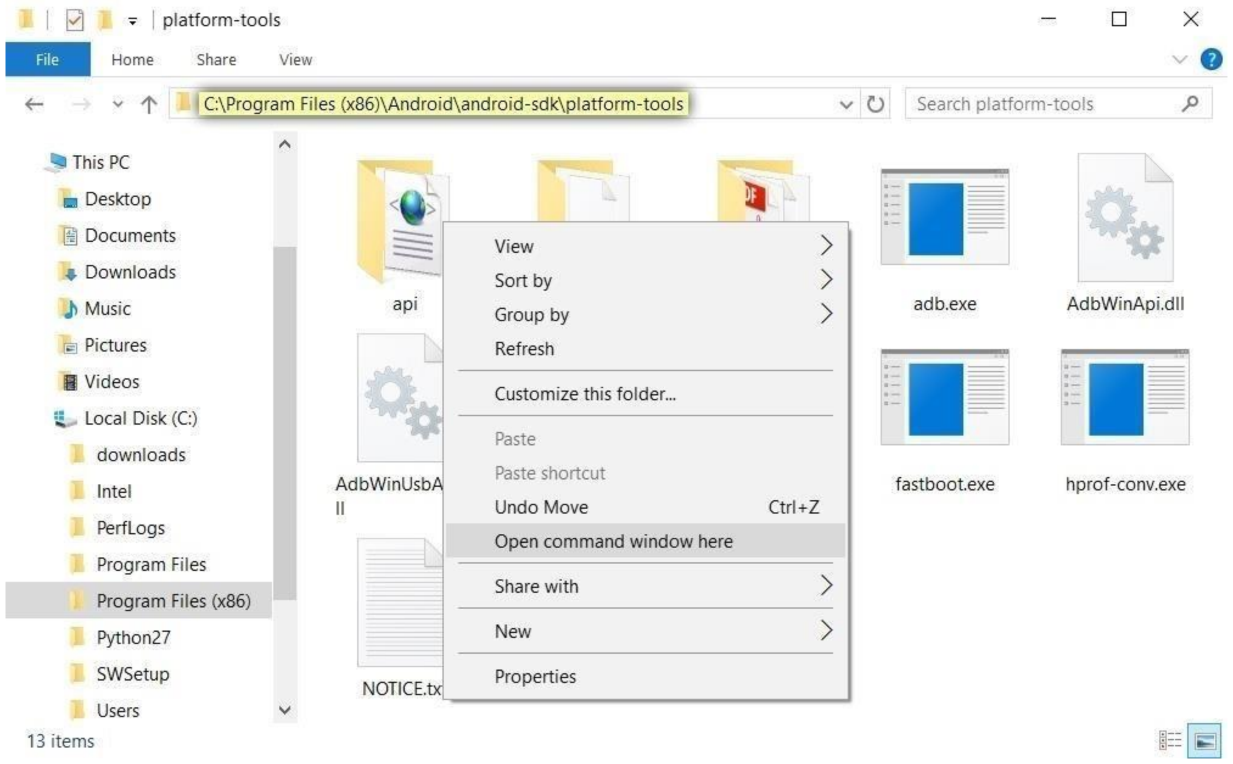
Photo transfer

Transfer photos and videos to the connected device

Step 7Open an ADB Shell on Your Computer

Now open the ADB and Fastboot installation directory on your computer. For most Windows users, this can be found at *C:\Program Files (x86)\Android\android-sdk\platform-tools*. For Mac and Linux users, it depends

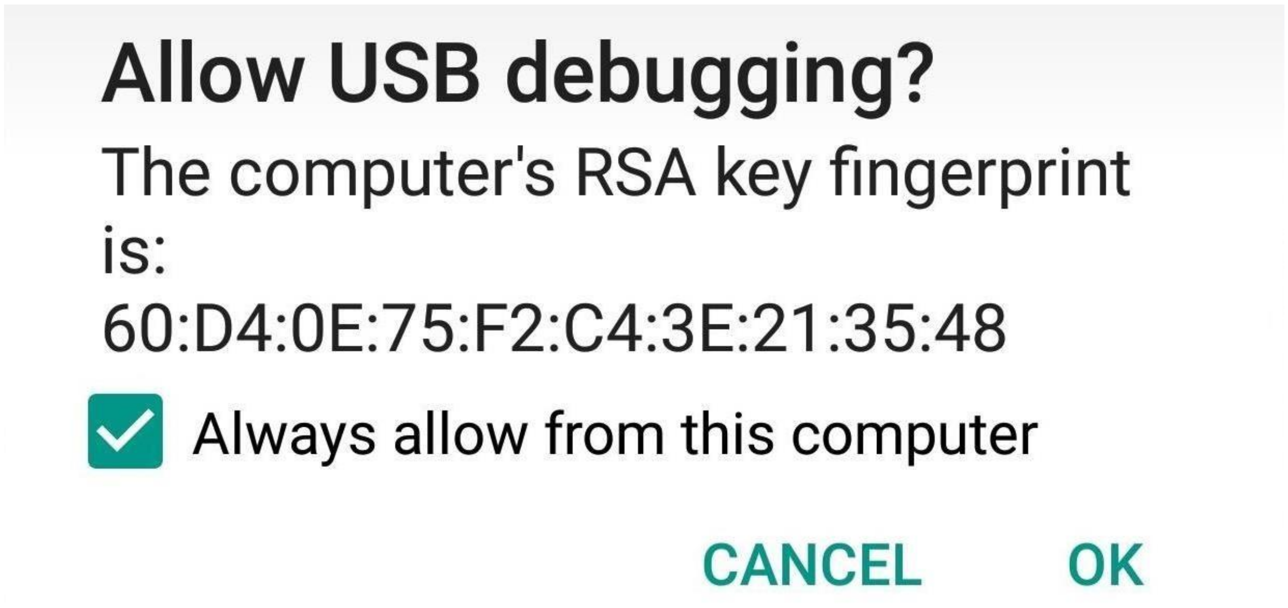
on where you extracted your ADB folder when you installed the utility, so search your hard drive for the *platform-tools* folder if you've forgotten.
From there, if you're using a Windows PC, hold down the shift button on your keyboard, then right-click any empty space and choose "Open command window here." For Mac and Linux users, simply open a Terminal window, then change directories to the *platform-tools* folder inside of your ADB installation directory.



Opening an ADB shell on Windows.

Step 8 Start ADB

Next, type **adb start-server** in the command prompt. Once you've done that, make sure to check the notification on your phone to accept the debugging notification.



Allow USB debugging?

The computer's RSA key fingerprint is:
60:D4:0E:75:F2:C4:3E:21:35:48

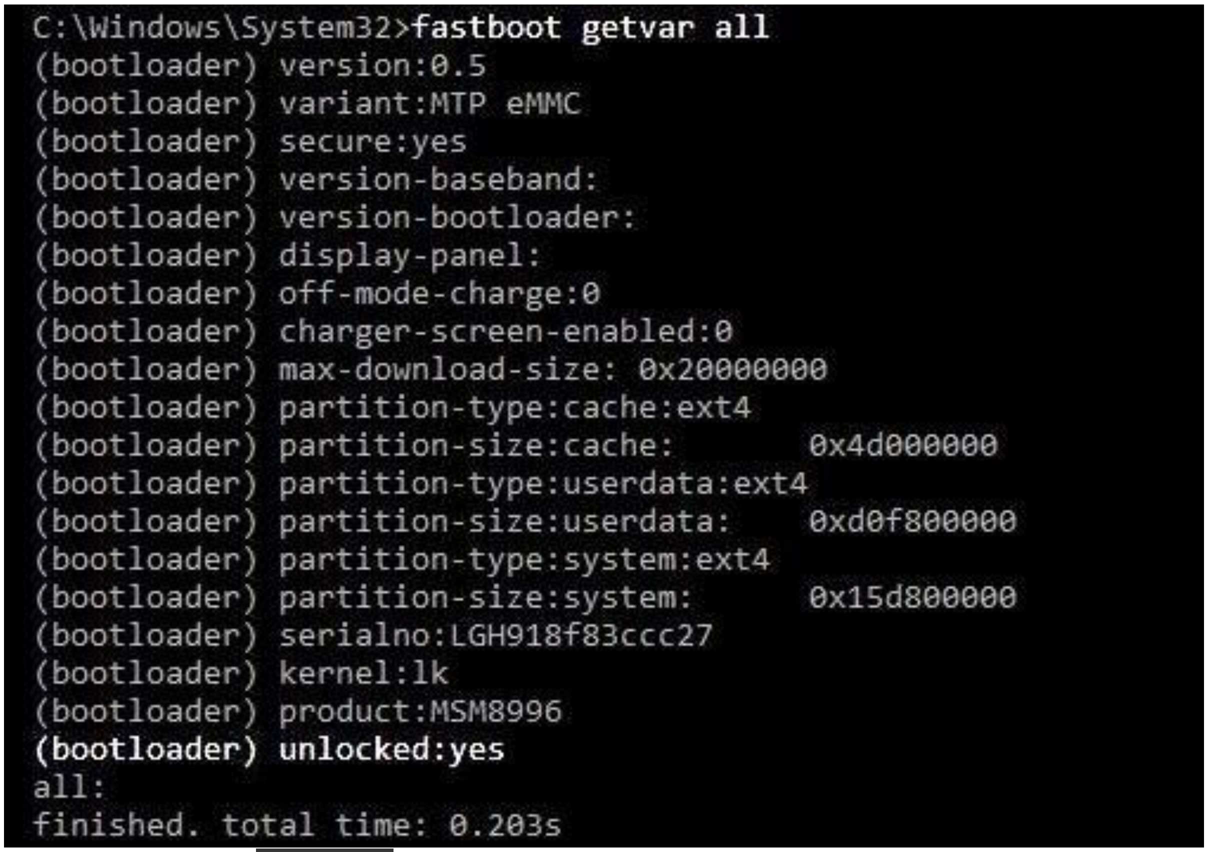
✓ Always allow from this computer

Step 9 Unlock Your Bootloader

NOTE: The following this step will wipe all your data and factory reset your phone.

Reboot into Fastboot mode by typing **adb reboot bootloader** into the ADB shell. Once your phone is in Bootloader mode, also known as Fastboot mode, type in **fastboot oem unlock**. At this point, you should see a message on your phone confirming if you'd like to unlock your bootloader and warning you that, so press your power button to confirm your choice.

Once you're done with that, check the status of your bootloader lock with the command **fastboot getvar all**. Make sure it says your bootloader is unlocked, and then boot back into your phone by typing, **fastboot reboot**.



```
C:\Windows\System32>fastboot getvar all
(bootloader) version:0.5
(bootloader) variant:MTP eMMC
(bootloader) secure:yes
(bootloader) version-baseband:
(bootloader) version-bootloader:
(bootloader) display-panel:
(bootloader) off-mode-charge:0
(bootloader) charger-screen-enabled:0
(bootloader) max-download-size: 0x20000000
(bootloader) partition-type:cache:ext4
(bootloader) partition-size:cache:       0x4d000000
(bootloader) partition-type:userdata:ext4
(bootloader) partition-size:userdata:    0xd0f800000
(bootloader) partition-type:system:ext4
(bootloader) partition-size:system:      0x15d800000
(bootloader) serialno:LGH918f83ccc27
(bootloader) kernel:lk
(bootloader) product:MSM8996
(bootloader) unlocked:yes
all:
finished. total time: 0.203s
```

Step 10Re-Enable USB Debugging

Upon rebooting, you will hit the initial welcome and set-up screen. Skip past as much of it as possible as our goal is to simply gain access to ADB. Then, once you make it to the home screen, repeat Steps 3-8 above to enable USB debugging on your phone again.

Step 11Download Required Exploitation Files

Now download the four required binaries, custom TWRP, and SuperSU onto your computer using the links below.

- the four required binaries
- custom TWRP recovery
- SuperSU

When you're done there, move all of the downloaded files over to the *platform-tools* folder inside of your ADB and Fastboot installation directory. If you're unsure about the location of this folder, check Step 7 above.

NOTE: XDA Senior Member bezeek has put together a one-click utility that will automate steps eleven and twelve. However, I recommend doing it manually as the one-click utility has caused users to bootloop and even brick their devices.

Step 12 Place the Dirty COW Files on Your Phone

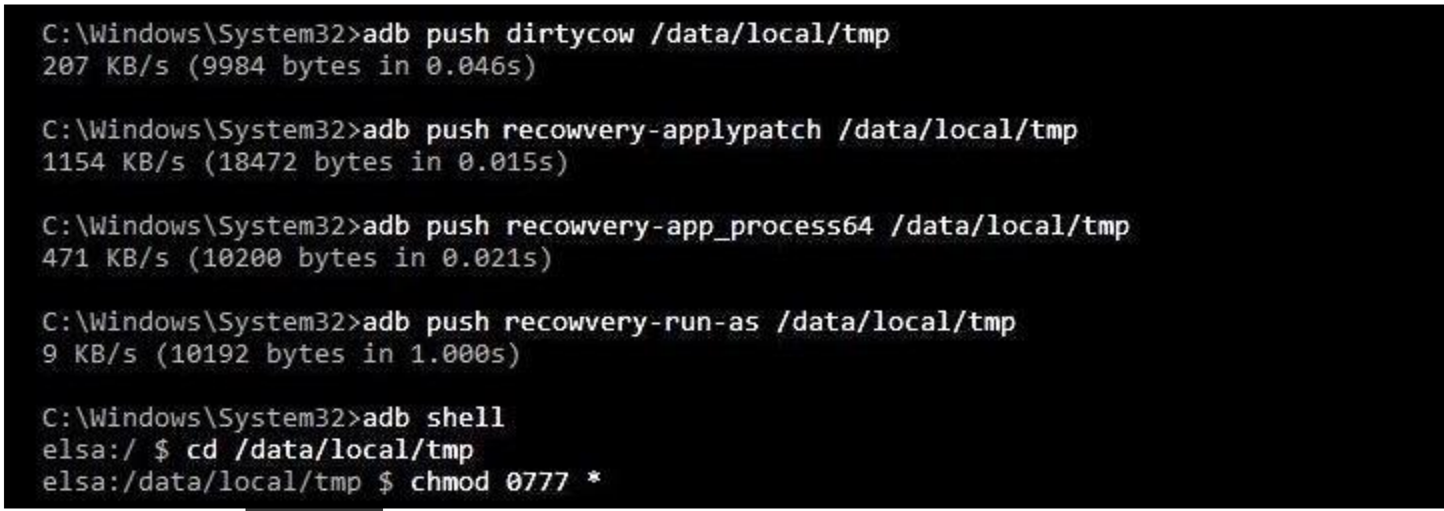
Back in the command prompt or terminal window on your computer, copy and paste the commands below one at a time. After each command or group of commands, I'll quickly explain what we just did or when it's important to be patient.

- adb push dirtycow /data/local/tmp
- adb push recowvery-applypatch /data/local/tmp
- adb push recowvery-app_process64 /data/local/tmp
- adb push recowvery-run-as /data/local/tmp

The four commands above will place the Dirty COW files on your phone so that we can use them later.

- adb shell
- cd /data/local/tmp
- chmod 0777 *

Here, we navigated to the directory with the Dirty COW files and allowed read and write permissions.



```
C:\Windows\System32>adb push dirtycow /data/local/tmp
207 KB/s (9984 bytes in 0.046s)

C:\Windows\System32>adb push recowvery-applypatch /data/local/tmp
1154 KB/s (18472 bytes in 0.015s)

C:\Windows\System32>adb push recowvery-app_process64 /data/local/tmp
471 KB/s (10200 bytes in 0.021s)

C:\Windows\System32>adb push recowvery-run-as /data/local/tmp
9 KB/s (10192 bytes in 1.000s)

C:\Windows\System32>adb shell
elsa:/ $ cd /data/local/tmp
elsa:/data/local/tmp $ chmod 0777 *
```

Step 13 Execute the DIRTY COW Files

Next up, type the following commands, which will execute the Dirty COW exploit on your phone.

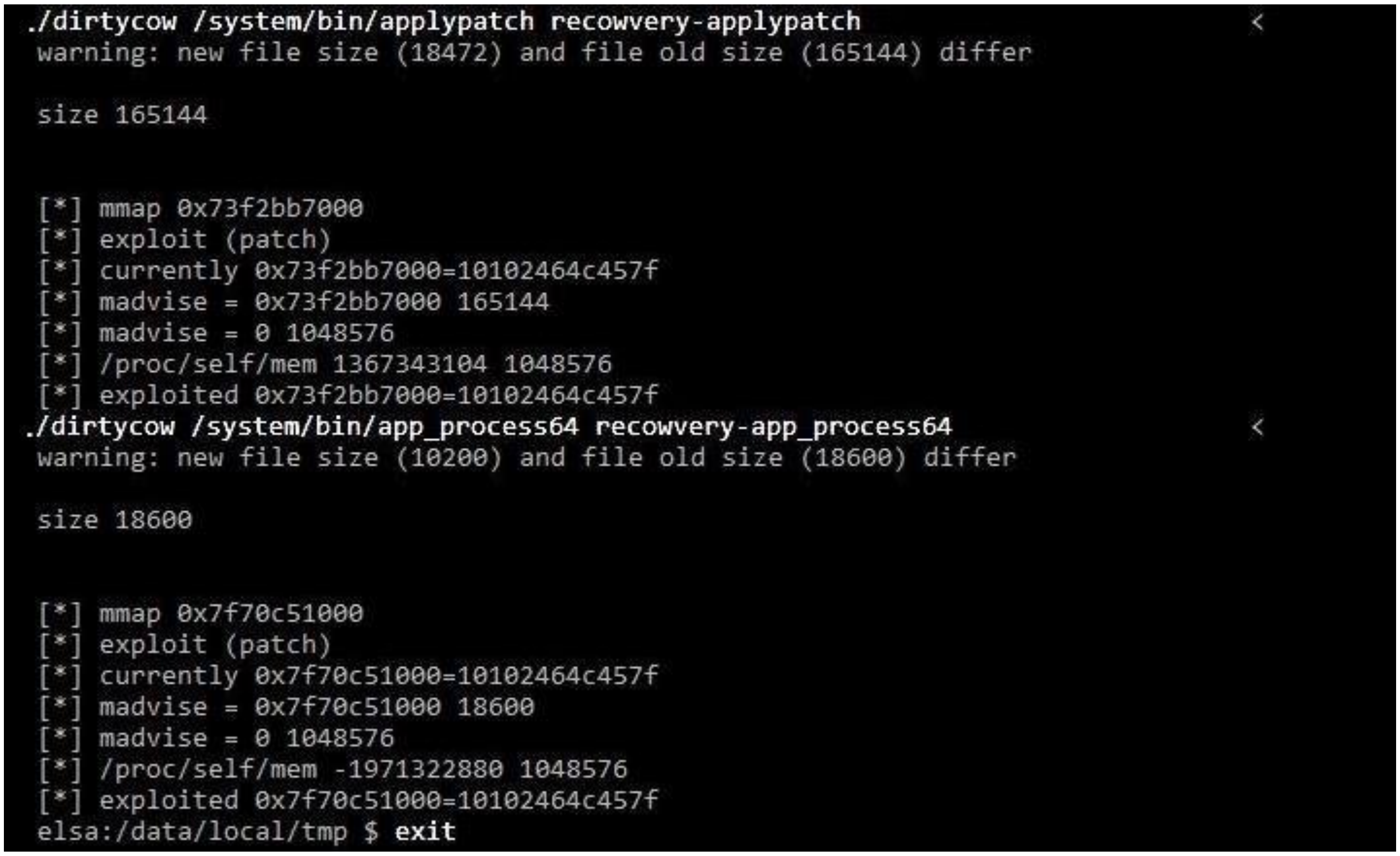
- ./dirtycow /system/bin/applypatch recowvery-applypatch

With the above command, we have begun the exploitation process. Wait for completion before proceeding.

- ./dirtycow /system/bin/app_process64 recowvery-app_process64

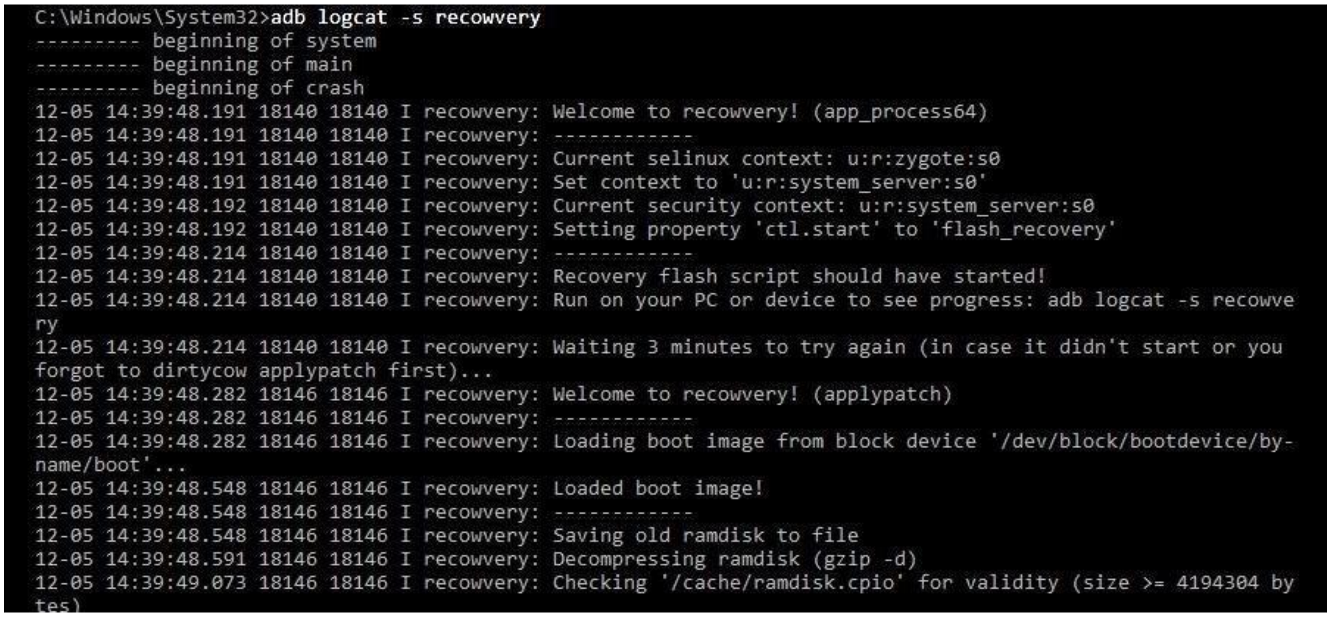
After the above command, your phone will look like it's crashing. Wait for completion before proceeding.

- exit



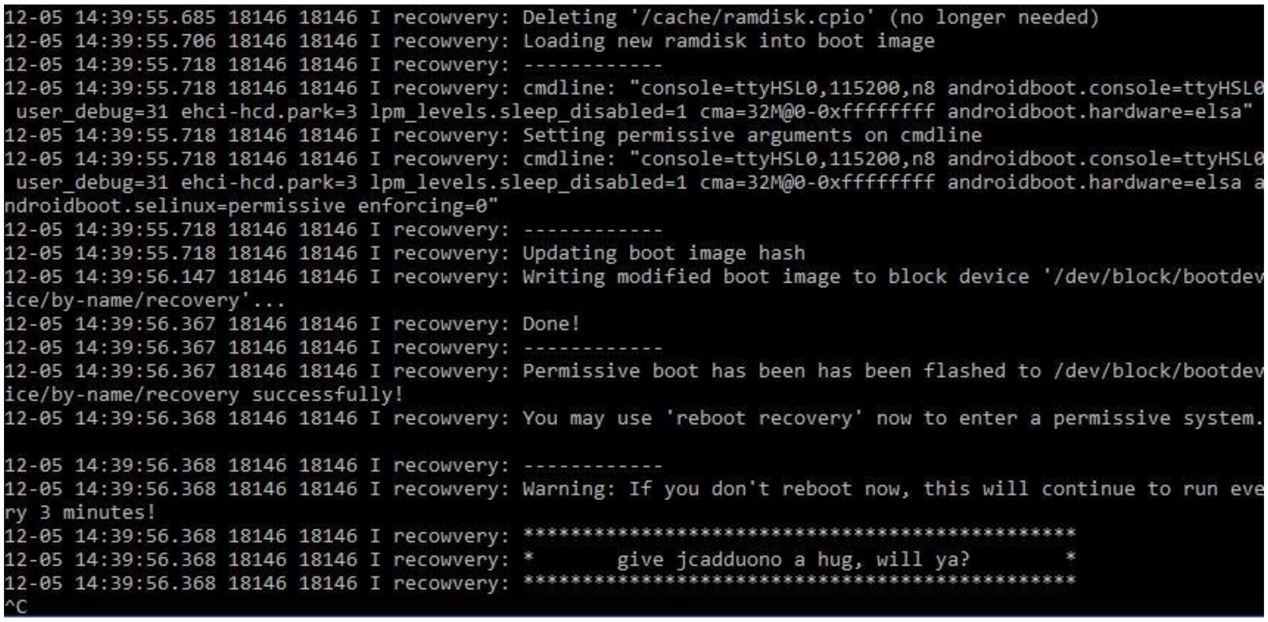
```
./dirtycow /system/bin/applypatch recowvery-applypatch                    <
 warning: new file size (18472) and file old size (165144) differ

 size 165144


 [*] mmap 0x73f2bb7000
 [*] exploit (patch)
 [*] currently 0x73f2bb7000=10102464c457f
 [*] madvise = 0x73f2bb7000 165144
 [*] madvise = 0 1048576
 [*] /proc/self/mem 1367343104 1048576
 [*] exploited 0x73f2bb7000=10102464c457f
./dirtycow /system/bin/app_process64 recowvery-app_process64              <
 warning: new file size (10200) and file old size (18600) differ

 size 18600


 [*] mmap 0x7f70c51000
 [*] exploit (patch)
 [*] currently 0x7f70c51000=10102464c457f
 [*] madvise = 0x7f70c51000 18600
 [*] madvise = 0 1048576
 [*] /proc/self/mem -1971322880 1048576
 [*] exploited 0x7f70c51000=10102464c457f
 elsa:/data/local/tmp $ exit
```

- adb logcat -s recowvery



```
C:\Windows\System32>adb logcat -s recowvery
--------- beginning of system
--------- beginning of main
--------- beginning of crash
12-05 14:39:48.191 18140 18140 I recowvery: Welcome to recowvery! (app_process64)
12-05 14:39:48.191 18140 18140 I recowvery: ------------
12-05 14:39:48.191 18140 18140 I recowvery: Current selinux context: u:r:zygote:s0
12-05 14:39:48.191 18140 18140 I recowvery: Set context to 'u:r:system_server:s0'
12-05 14:39:48.192 18140 18140 I recowvery: Current security context: u:r:system_server:s0
12-05 14:39:48.192 18140 18140 I recowvery: Setting property 'ctl.start' to 'flash_recovery'
12-05 14:39:48.214 18140 18140 I recowvery: ------------
12-05 14:39:48.214 18140 18140 I recowvery: Recovery flash script should have started!
12-05 14:39:48.214 18140 18140 I recowvery: Run on your PC or device to see progress: adb logcat -s recowve
ry
12-05 14:39:48.214 18140 18140 I recowvery: Waiting 3 minutes to try again (in case it didn't start or you
forgot to dirtycow applypatch first)...
12-05 14:39:48.282 18146 18146 I recowvery: Welcome to recowvery! (applypatch)
12-05 14:39:48.282 18146 18146 I recowvery: ------------
12-05 14:39:48.282 18146 18146 I recowvery: Loading boot image from block device '/dev/block/bootdevice/by-
name/boot'...
12-05 14:39:48.548 18146 18146 I recowvery: Loaded boot image!
12-05 14:39:48.548 18146 18146 I recowvery: ------------
12-05 14:39:48.548 18146 18146 I recowvery: Saving old ramdisk to file
12-05 14:39:48.591 18146 18146 I recowvery: Decompressing ramdisk (gzip -d)
12-05 14:39:49.073 18146 18146 I recowvery: Checking '/cache/ramdisk.cpio' for validity (size >= 4194304 by
tes)
```

Now wait for it to tell you the exploiting was successfully executed. We want to make sure the exploit worked before moving to the next step.



```
12-05 14:39:55.685 18146 18146 I recowvery: Deleting '/cache/ramdisk.cpio' (no longer needed)
12-05 14:39:55.706 18146 18146 I recowvery: Loading new ramdisk into boot image
12-05 14:39:55.718 18146 18146 I recowvery: ------------
12-05 14:39:55.718 18146 18146 I recowvery: cmdline: "console=ttyHSL0,115200,n8 androidboot.console=ttyHSL0
 user_debug=31 ehci-hcd.park=3 lpm_levels.sleep_disabled=1 cma=32M@0-0xffffffff androidboot.hardware=elsa"
12-05 14:39:55.718 18146 18146 I recowvery: Setting permissive arguments on cmdline
12-05 14:39:55.718 18146 18146 I recowvery: cmdline: "console=ttyHSL0,115200,n8 androidboot.console=ttyHSL0
 user_debug=31 ehci-hcd.park=3 lpm_levels.sleep_disabled=1 cma=32M@0-0xffffffff androidboot.hardware=elsa a
ndroidboot.selinux=permissive enforcing=0"
12-05 14:39:55.718 18146 18146 I recowvery: ------------
12-05 14:39:55.718 18146 18146 I recowvery: Updating boot image hash
12-05 14:39:56.147 18146 18146 I recowvery: Writing modified boot image to block device '/dev/block/bootdev
ice/by-name/recovery'...
12-05 14:39:56.367 18146 18146 I recowvery: Done!
12-05 14:39:56.367 18146 18146 I recowvery: ------------
12-05 14:39:56.367 18146 18146 I recowvery: Permissive boot has been has been flashed to /dev/block/bootdev
ice/by-name/recovery successfully!
12-05 14:39:56.368 18146 18146 I recowvery: You may use 'reboot recovery' now to enter a permissive system.

12-05 14:39:56.368 18146 18146 I recowvery: ------------
12-05 14:39:56.368 18146 18146 I recowvery: Warning: If you don't reboot now, this will continue to run eve
ry 3 minutes!
12-05 14:39:56.368 18146 18146 I recowvery: ************************************************
12-05 14:39:56.368 18146 18146 I recowvery: *      give jcadduono a hug, will ya?         *
12-05 14:39:56.368 18146 18146 I recowvery: ************************************************
^C
```

Now, before continuing with the following commands, make sure to press ***CTRL+C*** on your keyboard. In other programs it used as a copy function, but here, it will end the current process and start a new one.

- adb shell reboot recovery

Wait for phone to boot up again. Your recovery will be reverted to stock.

- adb shell

- getenforce

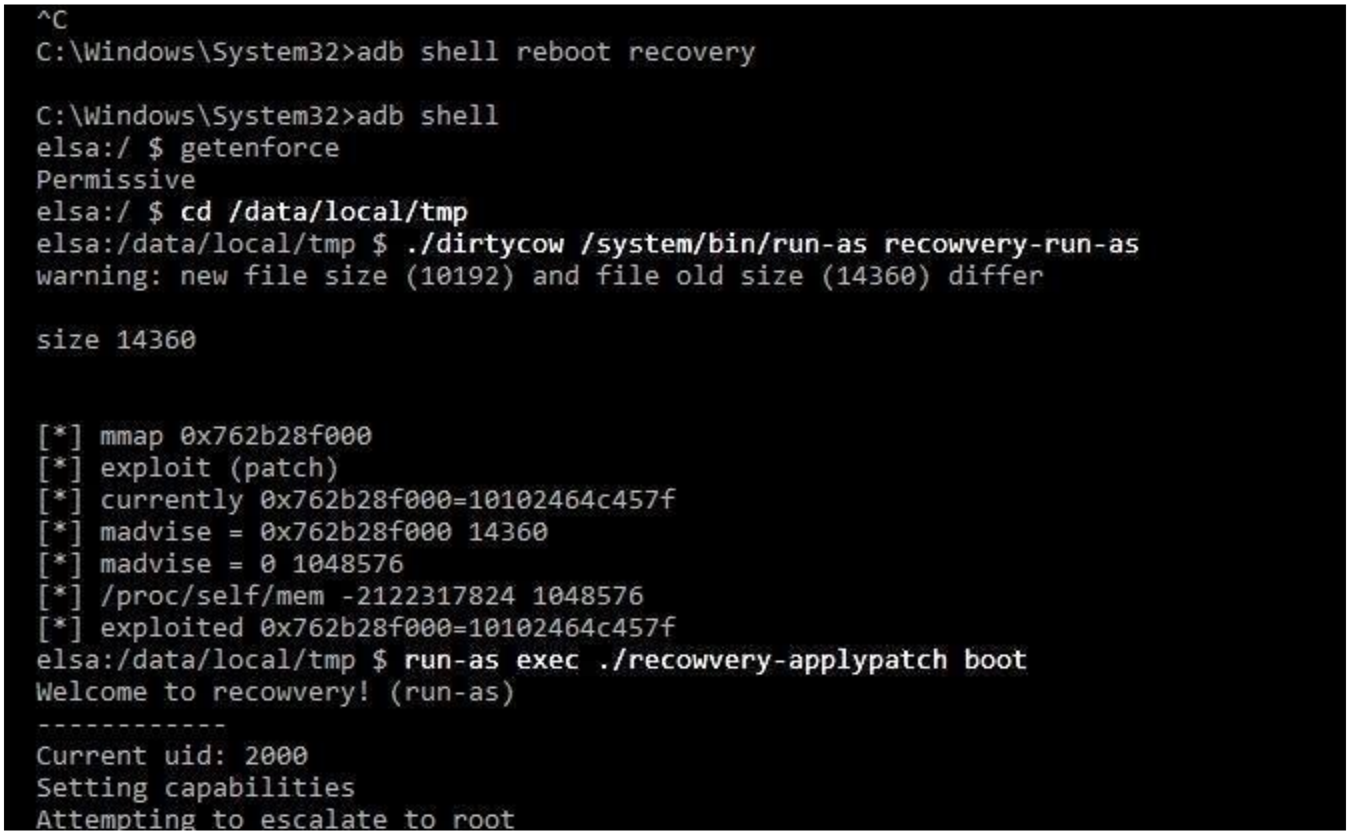
It should say "Permissive" on the screen. If it does, we're done with this step!

Step 14 Install TWRP

After that, we'll install TWRP custom recovery, which will allow us to make root permanent in a future step.

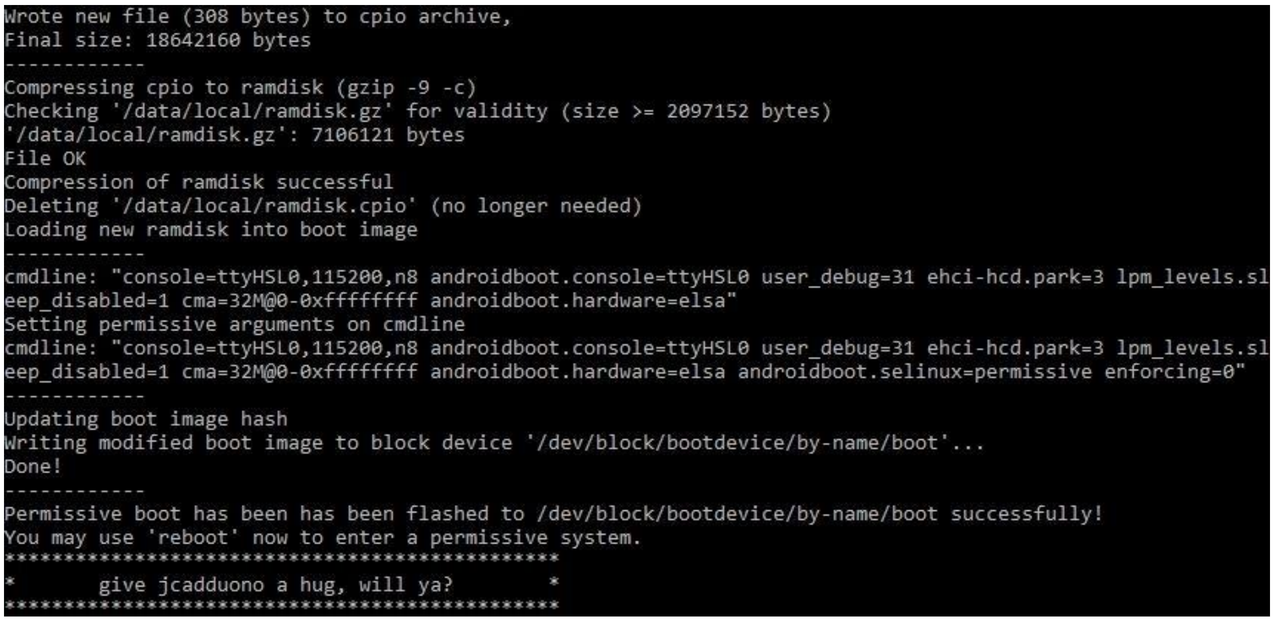
- cd /data/local/tmp
- ./dirtycow /system/bin/run-as recowvery-run-as
- run-as exec ./recowvery-applypatch boot

Wait for it to flash your boot image this time.



```
^C
C:\Windows\System32>adb shell reboot recovery

C:\Windows\System32>adb shell
elsa:/ $ getenforce
Permissive
elsa:/ $ cd /data/local/tmp
elsa:/data/local/tmp $ ./dirtycow /system/bin/run-as recowvery-run-as
warning: new file size (10192) and file old size (14360) differ

size 14360


[*] mmap 0x762b28f000
[*] exploit (patch)
[*] currently 0x762b28f000=10102464c457f
[*] madvise = 0x762b28f000 14360
[*] madvise = 0 1048576
[*] /proc/self/mem -2122317824 1048576
[*] exploited 0x762b28f000=10102464c457f
elsa:/data/local/tmp $ run-as exec ./recowvery-applypatch boot
Welcome to recowvery! (run-as)
------------
Current uid: 2000
Setting capabilities
Attempting to escalate to root
```

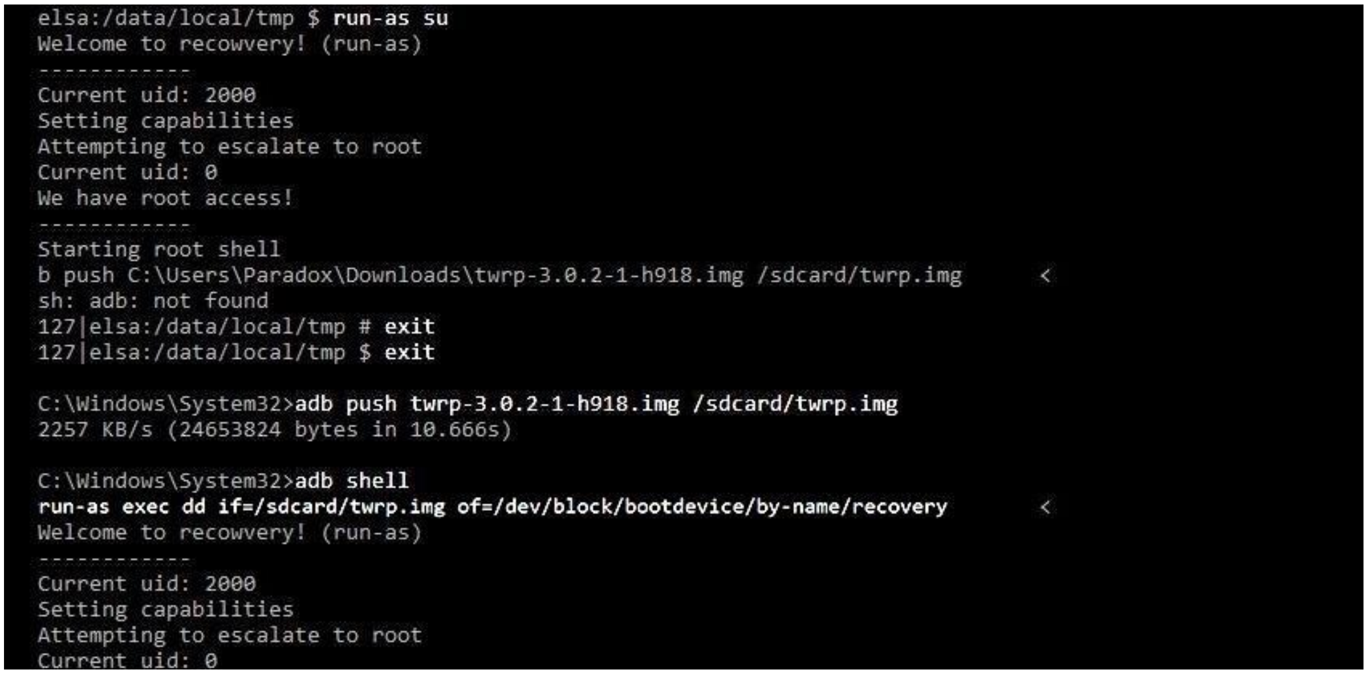
This is flashing a modified boot image with some security features disabled so that we can boot into TWRP after installing it.



```
Wrote new file (308 bytes) to cpio archive,
Final size: 18642160 bytes
------------
Compressing cpio to ramdisk (gzip -9 -c)
Checking '/data/local/ramdisk.gz' for validity (size >= 2097152 bytes)
'/data/local/ramdisk.gz': 7106121 bytes
File OK
Compression of ramdisk successful
Deleting '/data/local/ramdisk.cpio' (no longer needed)
Loading new ramdisk into boot image
------------
cmdline: "console=ttyHSL0,115200,n8 androidboot.console=ttyHSL0 user_debug=31 ehci-hcd.park=3 lpm_levels.sl
eep_disabled=1 cma=32M@0-0xffffffff androidboot.hardware=elsa"
Setting permissive arguments on cmdline
cmdline: "console=ttyHSL0,115200,n8 androidboot.console=ttyHSL0 user_debug=31 ehci-hcd.park=3 lpm_levels.sl
eep_disabled=1 cma=32M@0-0xffffffff androidboot.hardware=elsa androidboot.selinux=permissive enforcing=0"
------------
Updating boot image hash
Writing modified boot image to block device '/dev/block/bootdevice/by-name/boot'...
Done!
------------
Permissive boot has been has been flashed to /dev/block/bootdevice/by-name/boot successfully!
You may use 'reboot' now to enter a permissive system.
*************************************************
*       give jcadduono a hug, will ya?        *
*************************************************
```

- run-as su
- exit
- exit
- adb push twrp-3.0.2-1-h918.img /sdcard/twrp.img
- adb shell
- run-as exec dd if=/sdcard/twrp.img of=/dev/block/bootdevice/by-name/recovery

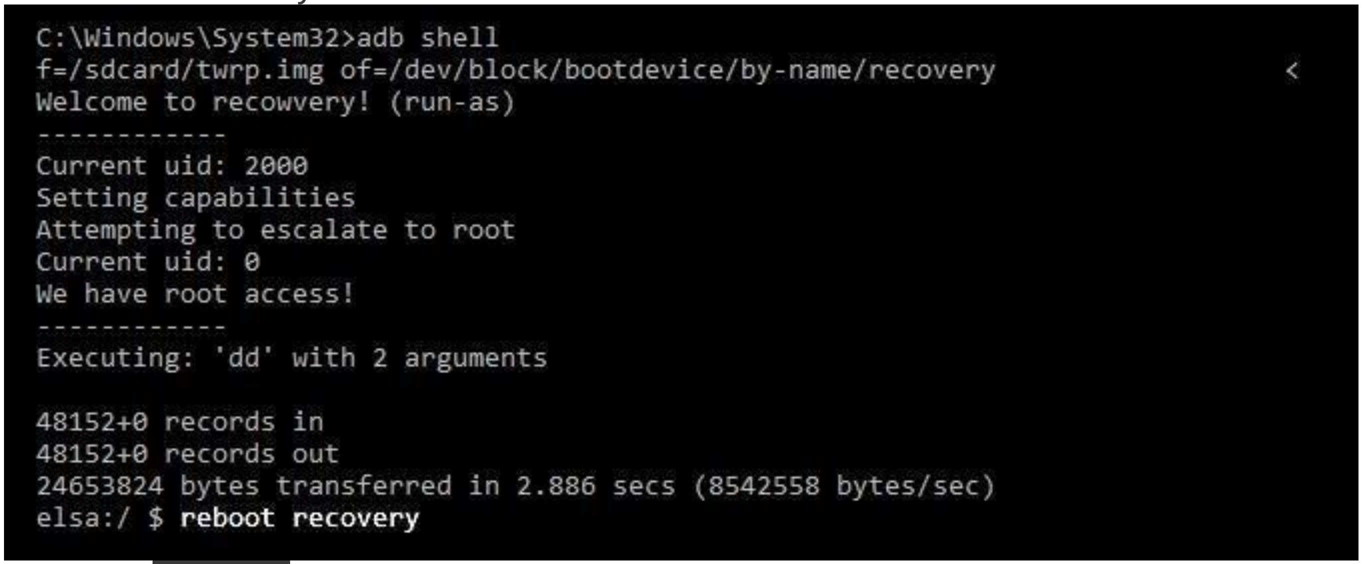
Here, we just replaced the stock Android recovery with TWRP.



```
elsa:/data/local/tmp $ run-as su
Welcome to recowvery! (run-as)
------------
Current uid: 2000
Setting capabilities
Attempting to escalate to root
Current uid: 0
We have root access!
------------
Starting root shell
b push C:\Users\Paradox\Downloads\twrp-3.0.2-1-h918.img /sdcard/twrp.img      <
sh: adb: not found
127|elsa:/data/local/tmp # exit
127|elsa:/data/local/tmp $ exit

C:\Windows\System32>adb push twrp-3.0.2-1-h918.img /sdcard/twrp.img
2257 KB/s (24653824 bytes in 10.666s)

C:\Windows\System32>adb shell
run-as exec dd if=/sdcard/twrp.img of=/dev/block/bootdevice/by-name/recovery      <
Welcome to recowvery! (run-as)
------------
Current uid: 2000
Setting capabilities
Attempting to escalate to root
Current uid: 0
```

Wait for it to complete, then:

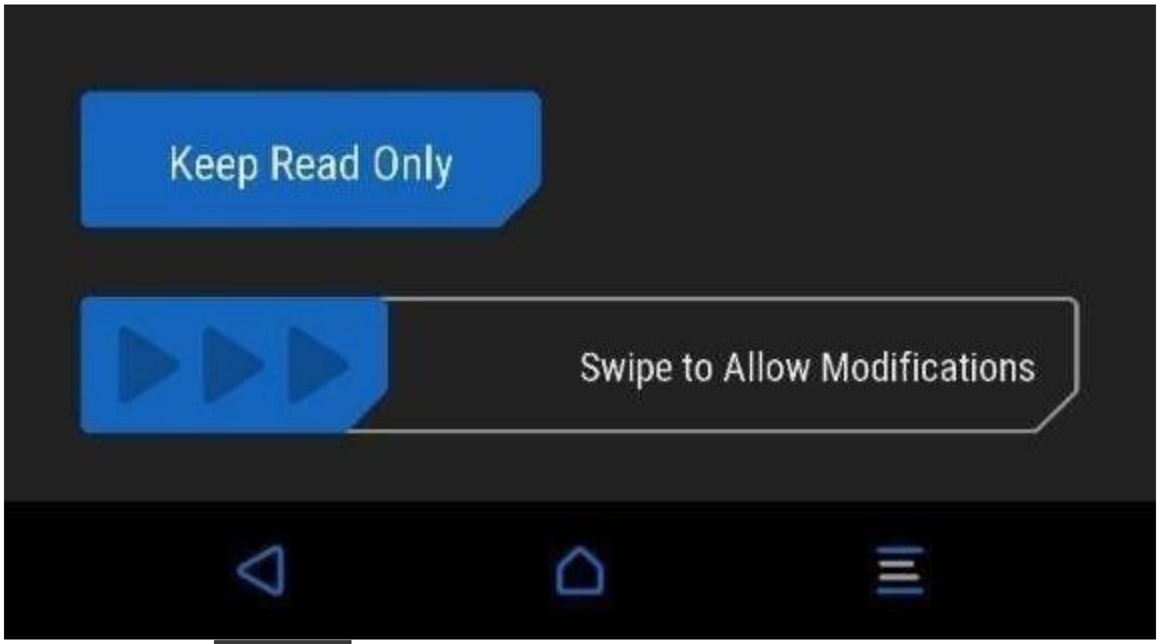
- reboot recovery



```
C:\Windows\System32>adb shell
f=/sdcard/twrp.img of=/dev/block/bootdevice/by-name/recovery                <
Welcome to recowvery! (run-as)
------------
Current uid: 2000
Setting capabilities
Attempting to escalate to root
Current uid: 0
We have root access!
------------
Executing: 'dd' with 2 arguments

48152+0 records in
48152+0 records out
24653824 bytes transferred in 2.886 secs (8542558 bytes/sec)
elsa:/ $ reboot recovery
```

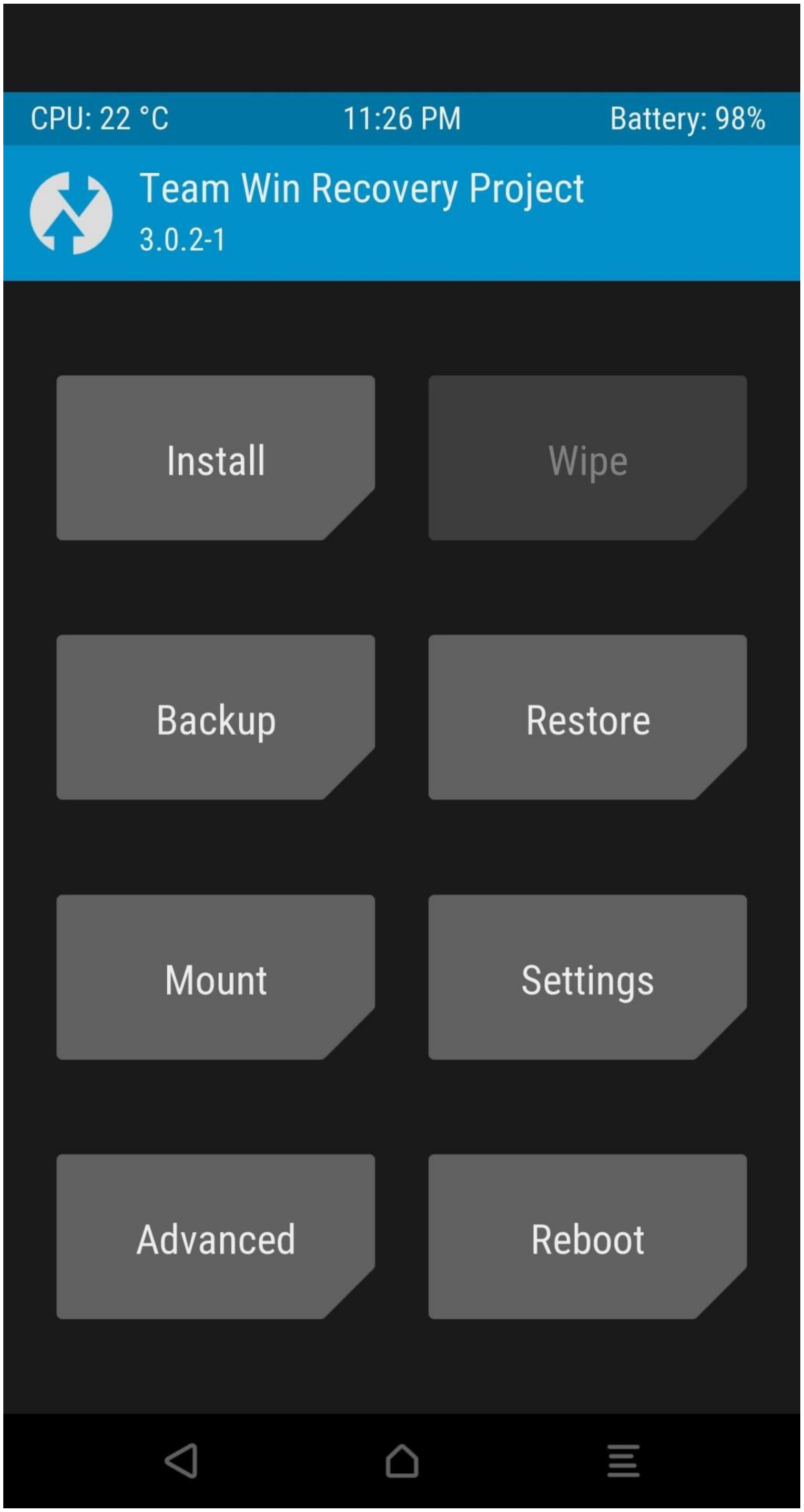
Step 15 Make TWRP Your Permanent Recovery

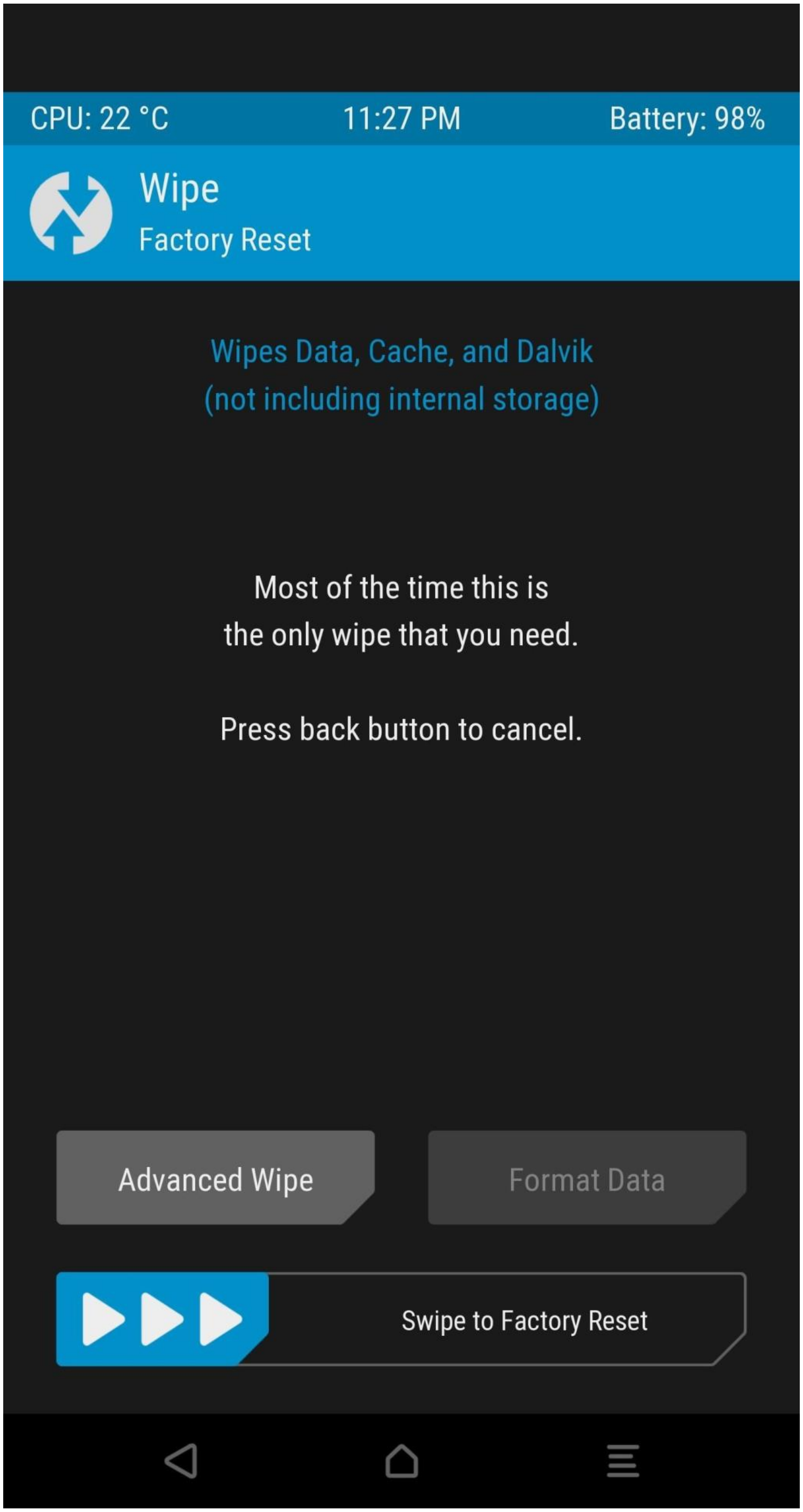
If you've made it this far, that means you have successfully booted into TWRP. At this point, TWRP will ask you if you want to enable system modifications. Make sure to swipe to allow modifications, otherwise your OS will replace TWRP on the next boot.



Step 16 Decrypt Your Device Storage

Next, use the "Format Data" option under the *Wipe* tab in order to disable device encryption. Type in **yes** and tap the check mark.


CPU: 22 °C
11:26 PM
Battery: 98%
Team Win Recovery Project
3.0.2-1
Install
Wipe
Backup
Restore
Mount
Settings
Advanced
Reboot



CPU: 22 °C 11:27 PM Battery: 98%

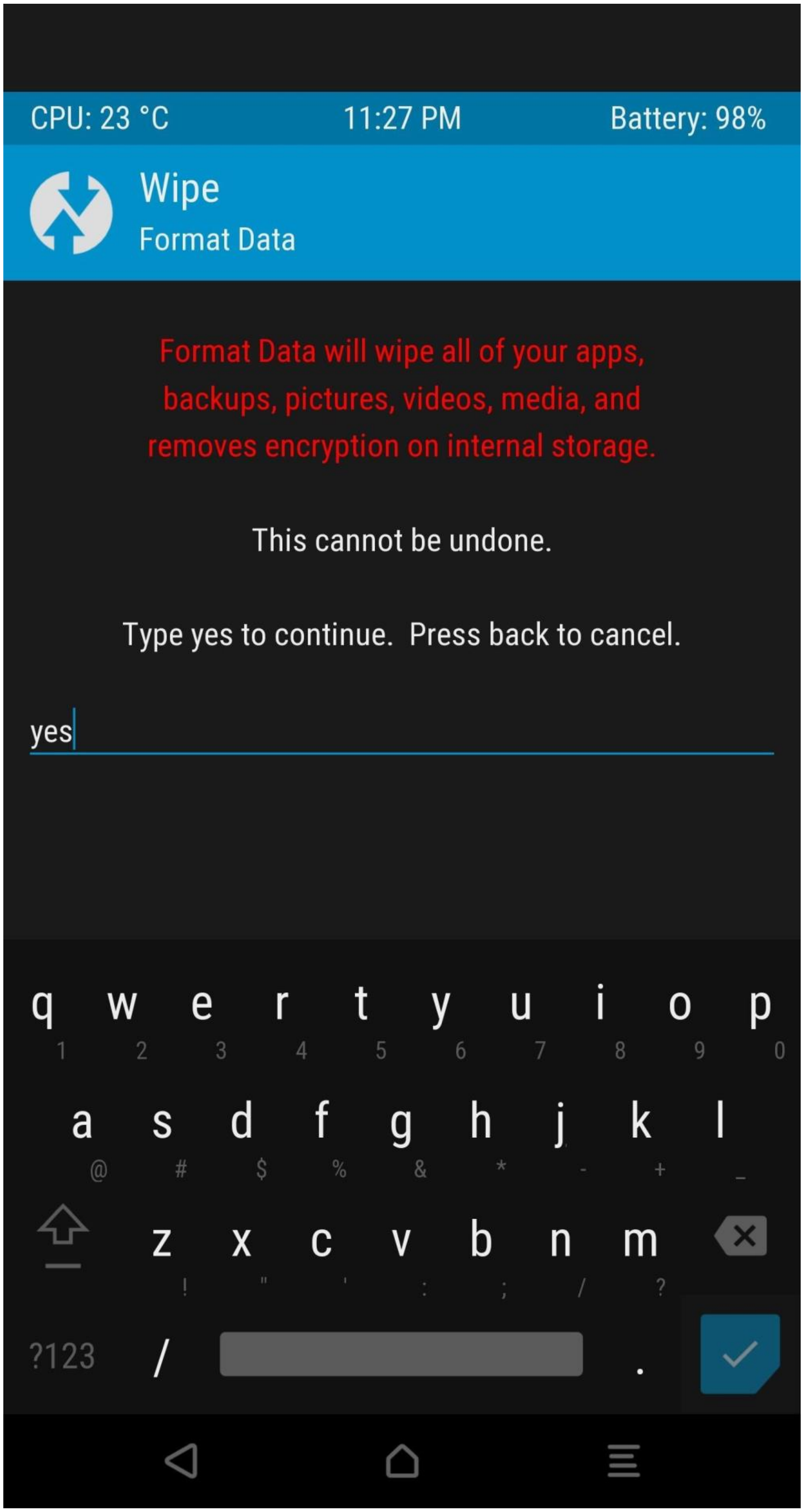
Wipe

Factory Reset

Wipes Data, Cache, and Dalvik
(not including internal storage)

Most of the time this is
the only wipe that you need.

Press back button to cancel.



CPU: 23 °C 11:27 PM Battery: 98%

Wipe

Format Data

Format Data will wipe all of your apps, backups, pictures, videos, media, and removes encryption on internal storage.

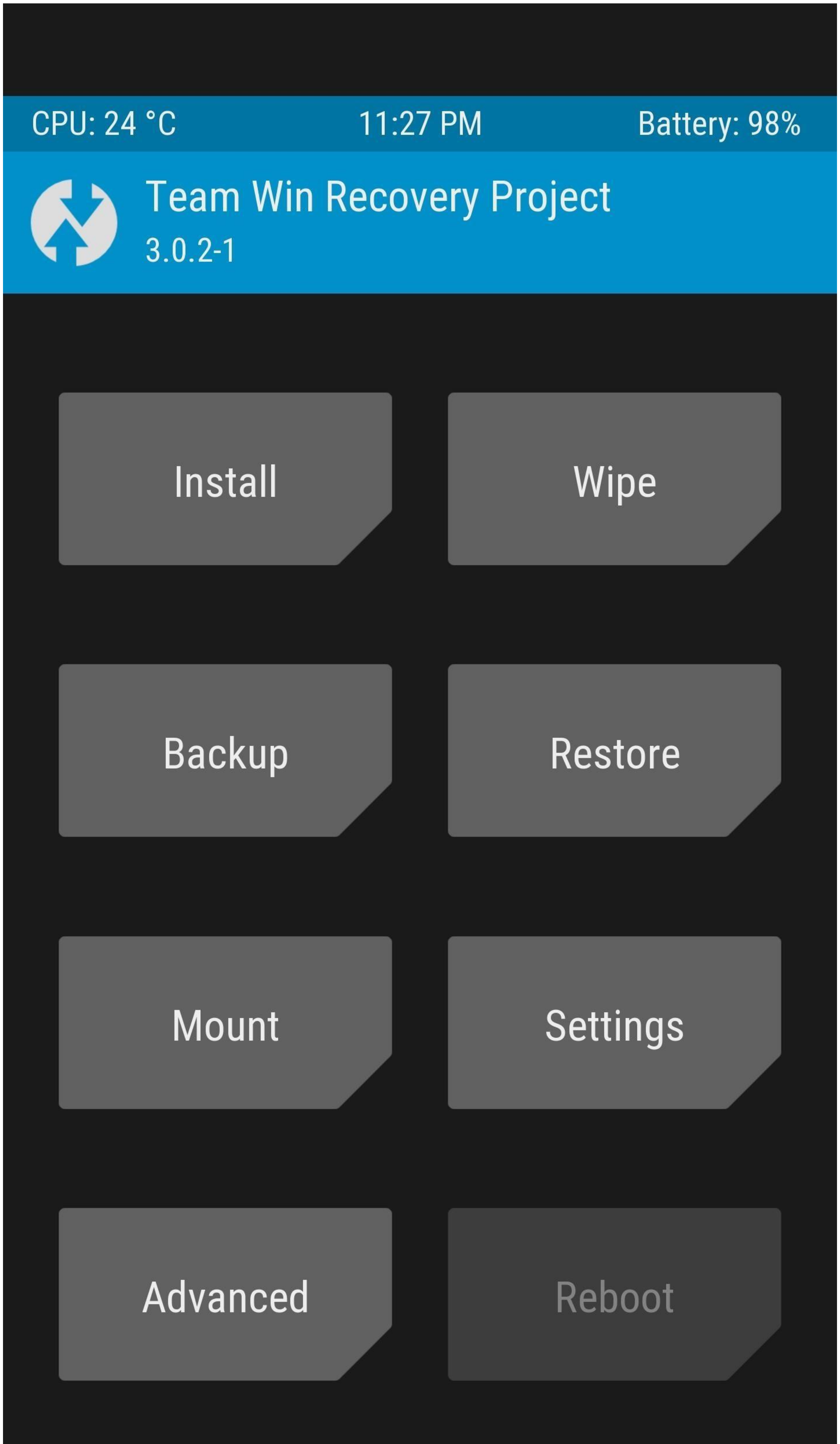
This cannot be undone.

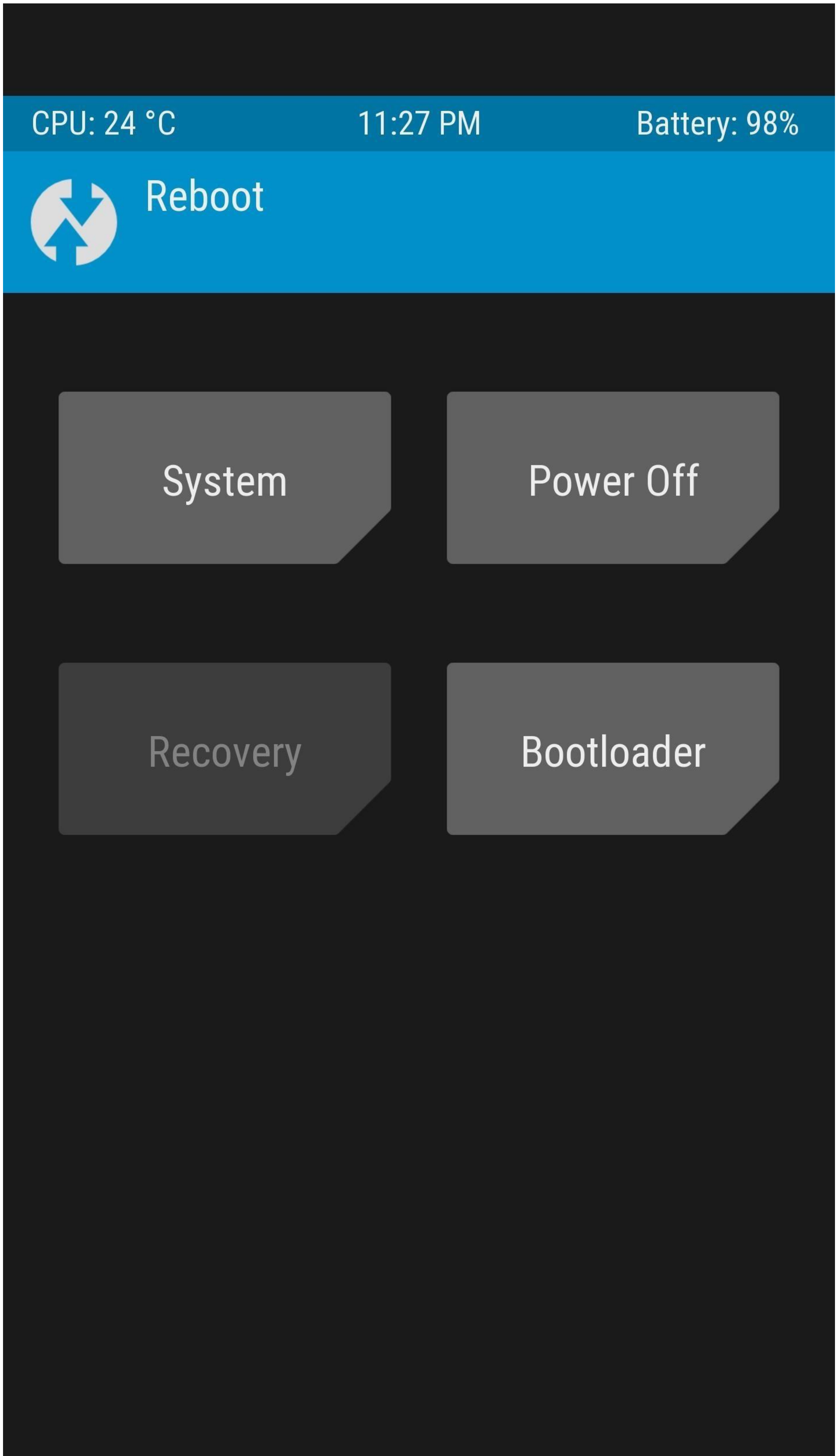
Type yes to continue. Press back to cancel.

yes

Step 17 Reboot TWRP with Decrypted Storage

Go back to the main screen of TWRP and open the *Reboot* tab to select the "Reboot Recovery" option.


CPU: 24 °C
11:27 PM
Battery: 98%
Team Win Recovery Project
3.0.2-1
Install
Wipe
Backup
Restore
Mount
Settings
Advanced
Reboot

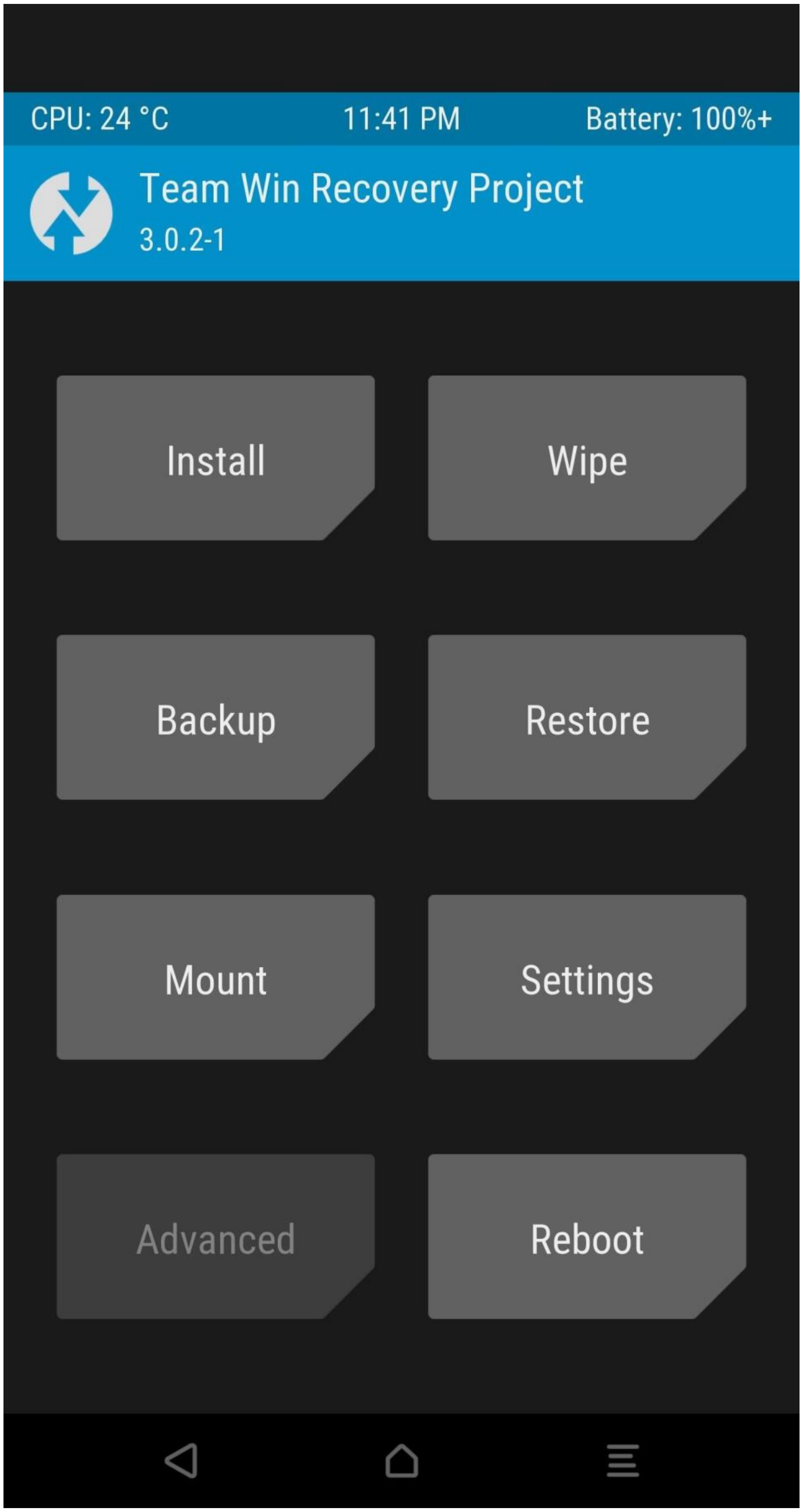

CPU: 24 °C
11:27 PM
Battery: 98%
Reboot
System
Power Off
Recovery
Bootloader

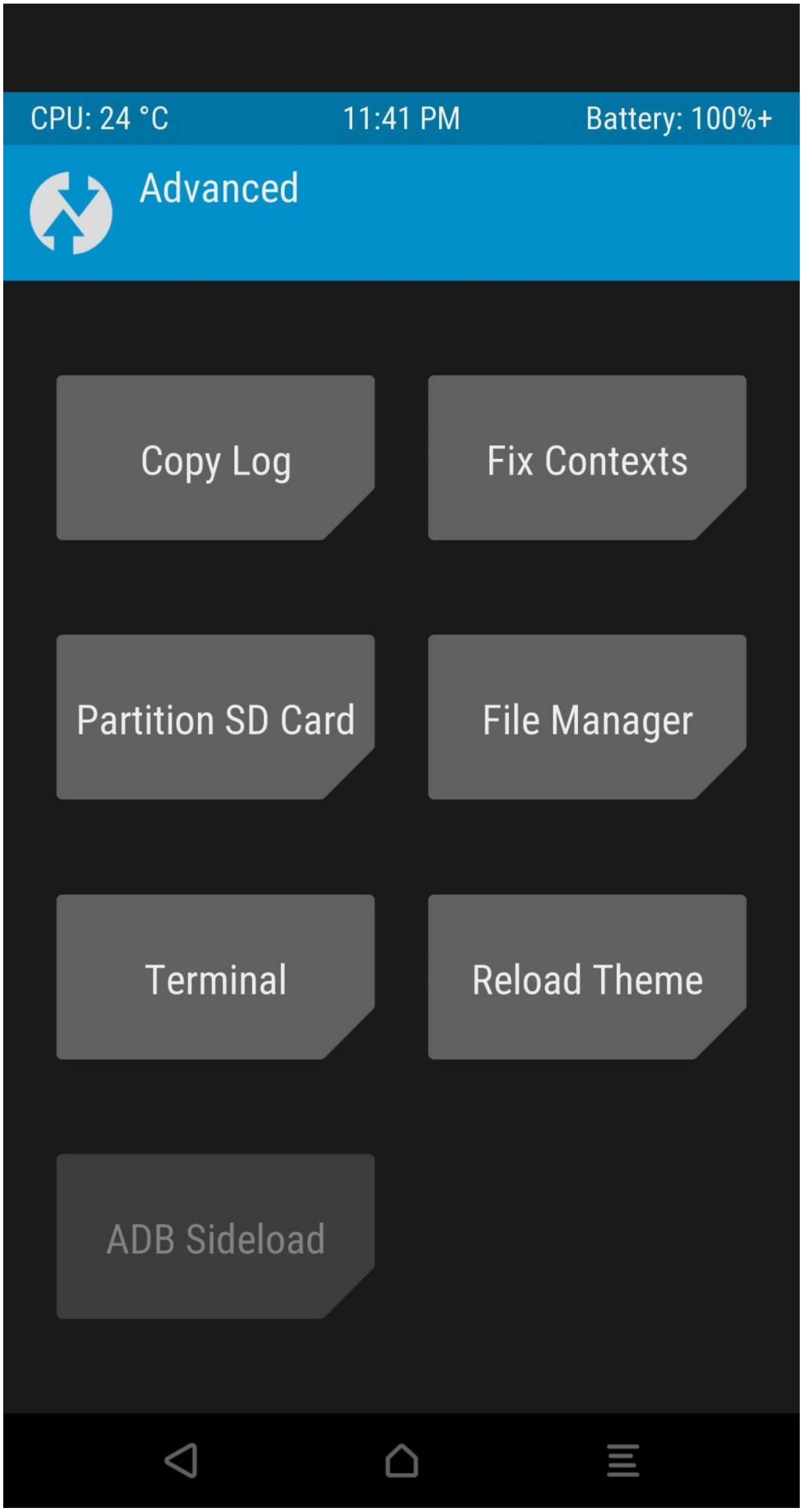
This will reboot you straight back into TWRP with your decrypted phone and allow you to flash the previously-downloaded SuperSU ZIP after you move it to the device or your SD card while in TWRP.

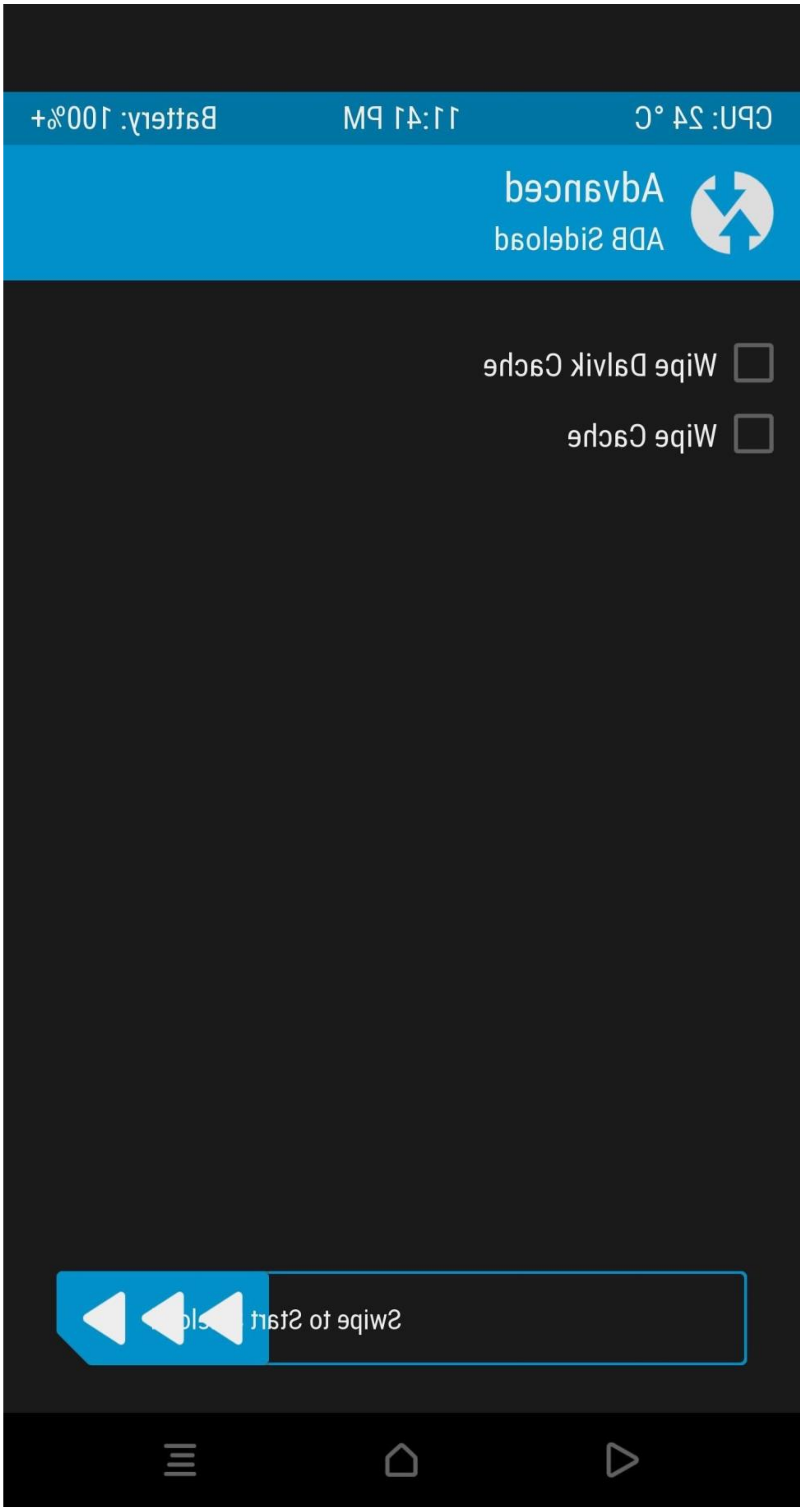
Step 18 Flash SuperSU for Permanent Root

It's time to flash SuperSU in TWRP so that we can keep root active without having to repeat the Dirty COW installation process every time we turn on the phone.

In TWRP, open the *Advanced* tab, select "ADB Sideload", and then Swipe to Start Sideload.


CPU: 24 °C
11:41 PM
Battery: 100%+
Team Win Recovery Project
3.0.2-1
Install
Wipe
Backup
Restore
Mount
Settings
Advanced
Reboot

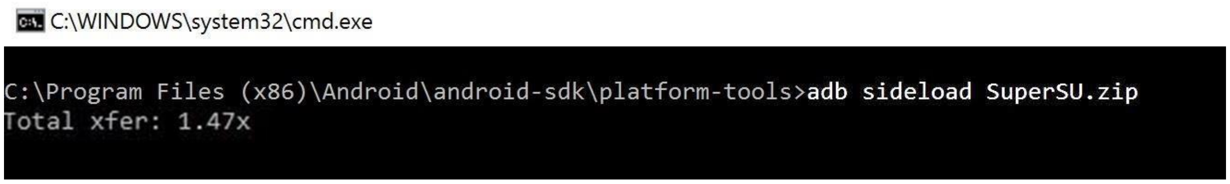

CPU: 24 °C
11:41 PM
Battery: 100%+
Advanced
Copy Log
Fix Contexts
Partition SD Card
File Manager
Terminal
Reload Theme
ADB Sideload


CPU: 24 °C
11:41 PM
Battery: 100%+
Advanced
ADB Sideload
Wipe Dalvik Cache
Wipe Cache
Swipe to Start

Next, find the *SuperSU-vXXX.zip* file that you pasted into the *platform-tools* folder back in Step 11, then rename this file to simply **SuperSU.zip**.
After that, go back to command prompt, command line, or terminal, and type in the following:

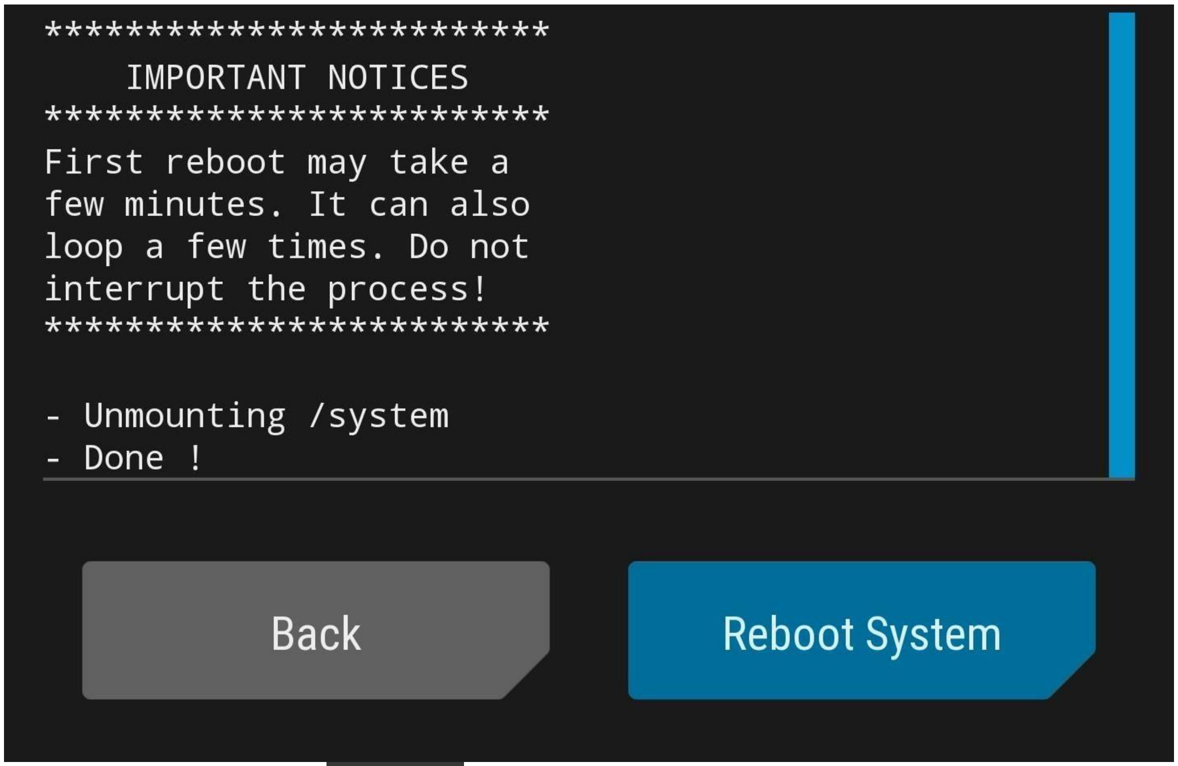
- adb sideload SuperSU.zip

Then hit enter to begin the SuperSU installation.



```
C:\WINDOWS\system32\cmd.exe
C:\Program Files (x86)\Android\android-sdk\platform-tools>adb sideload SuperSU.zip
Total xfer: 1.47x
```

You're all done, hit the "Reboot System" option in TWRP (your phone will reboot a few times due to SuperSU settling down) for a brand new device experience with root!

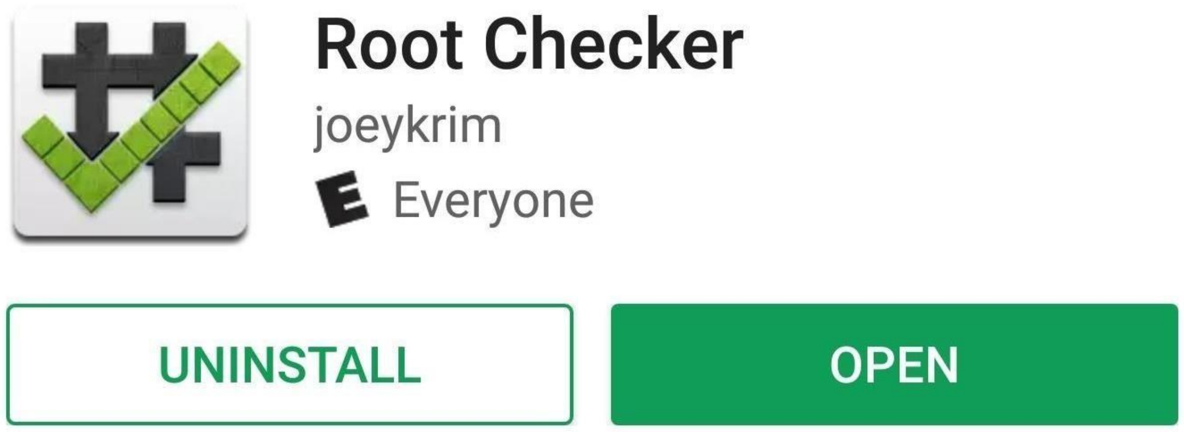


```
*************************
    IMPORTANT NOTICES
*************************
First reboot may take a
few minutes. It can also
loop a few times. Do not
interrupt the process!
*************************

- Unmounting /system
- Done !
```

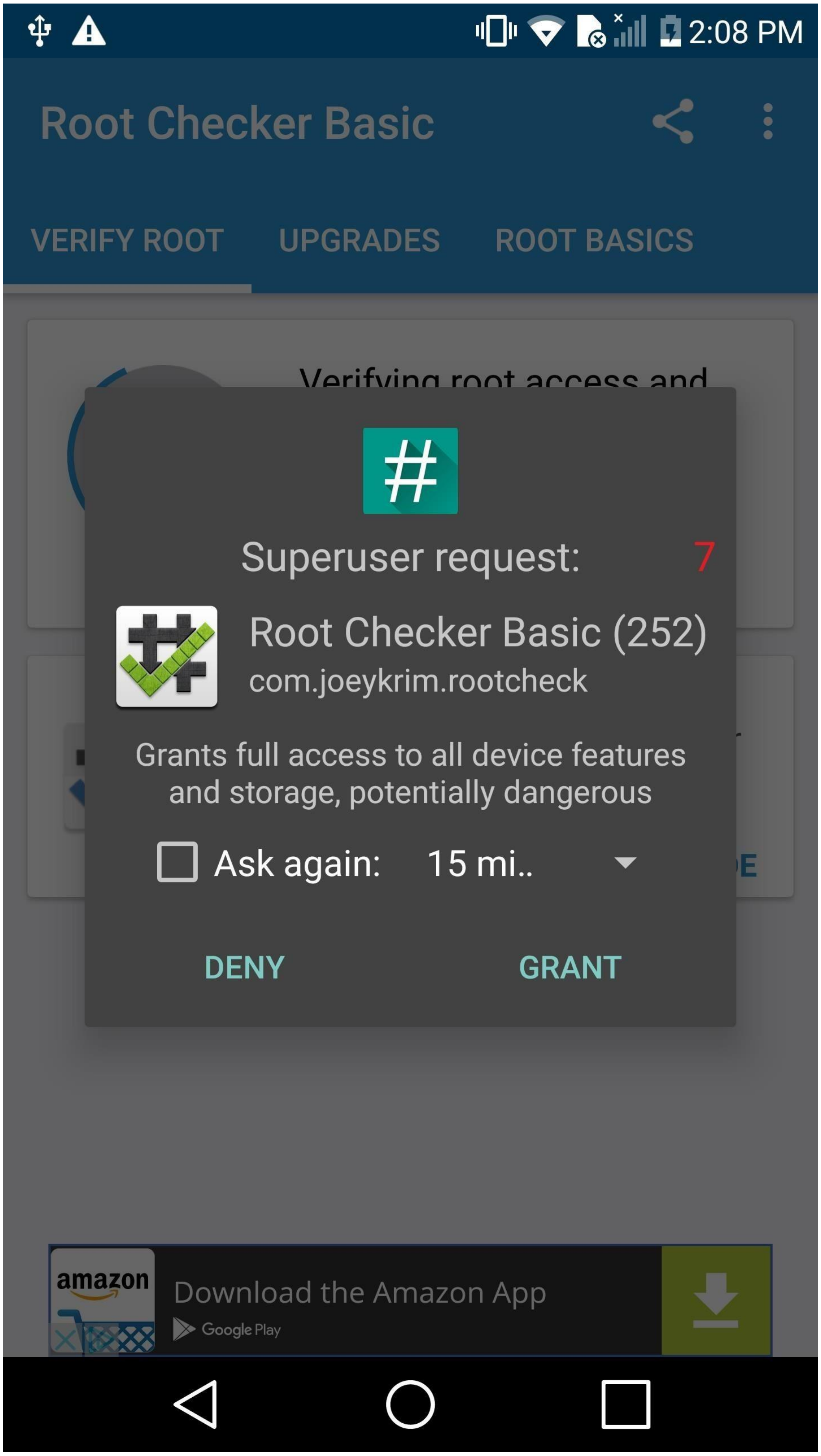
Step 19Verify Root Access

You'll need to set up your phone again, so take care of that, then you'll just need to verify that your phone is properly rooted. To do that, start by installing an app called *Root Checker* by developer joeykrim.

- **Install Root Checker for free from the Google Play Store**



When you first launch the app, a Superuser request will pop up. Tap "Grant" here, and Root Checker will verify that you are now rooted.

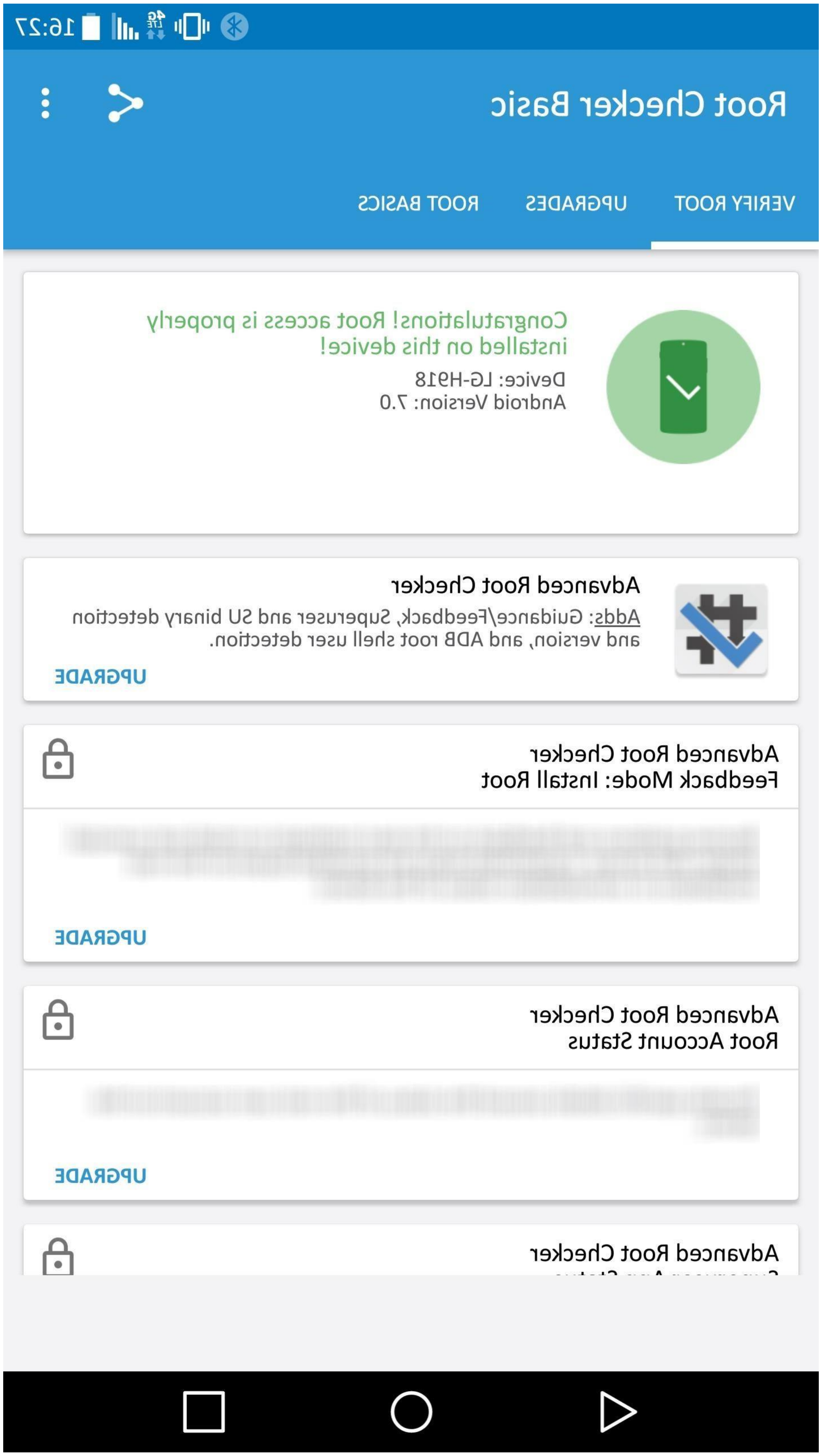


Superuser request: 7

Root Checker Basic (252)
com.joeykrim.rootcheck

Grants full access to all device features and storage, potentially dangerous

Ask again: 15 mi..



16:27

Root Checker Basic

VERIFY ROOT UPGRADES ROOT BASICS

Congratulations! Root access is properly installed on this device!

Device: LG-H918
Android Version: 7.0

Advanced Root Checker

Adds: Guidance/Feedback, Superuser and SU binary detection and version, and ADB root shell user detection.

UPGRADE

Advanced Root Checker
Feedback Mode: Install Root

UPGRADE

Advanced Root Checker
Root Account Status

UPGRADE

Advanced Root Checker

Once you see that green check, that's all there is to it. You're all set to enjoy your rooted LG V20!