Chuan Ji

November 21, 2012 · ANDROID TECHNICAL NOTES

Android Rooting: A Developer's Guide

First things first — this is not about how to apply a rooting method, e.g., a one-click-root, to an Android device. Rather, it is about how one could go about *developing* a rooting method for a device that no one has rooted before, and is told through my experiences with rooting a particular device — the Barnes & Noble Nook Tablet 8GB. For context, you can read my original thread "Root for Nook Tablet 8GB (w/ Android Market) on XDA-Developers where I published my rooting method, which has reached a download count of — wait for it — OVER NINE THOUSAND!!!

For an overview of how rooting works behind the scenes, you may want to read my previous article How Rooting Works - A Technical Explanation of the Android Rooting Process as background.

Sometime in late February (2012), on a visit to to a Barnes & Noble store in Boston, I bought the then freshly released Nook Tablet 8GB entirely on impulse for $199. Being the hax0r that I am, the first thing I did when I got home was to try to root the device. It came as a nasty surprise, therefore, when I discovered that no one had yet succeeded in

rooting the device. All I could find was a YouTube video showing that the existing rooting method for its cousin, the Nook Tablet 16GB, did not work. After waiting for a few days, the absolutely pathetic app store and handicaps instituted by B&N finally motivated me to develop a rooting method for the device myself.

The Plan

So, how do you go about rooting an Android device? As I explain in my previous article, rooting is basically a two-step process:

1. Find exploit that allows execution of arbitrary code as root
2. Use exploit to install `su` (with SUID bit set) and `Superuser.apk` as root

After `su` and `Superuser.apk` are installed correctly, apps that require root (such as Titanium Backup or AdAway) will invoke `su` to run code as the privileged user.

The Process

There are many generic or device-specific exploits that a hax0r may leverage to achieve privileged execution of arbitrary code. I would again refer you to this excellent presentation on various Android root exploits that have been or may still be used for this purpose.

However, none of these methods that I knew of could work on the Nook Tablet, which is

probably one of the most locked-down Android ROMs out there:

- Bootloader is locked.
- ADB is disabled, and cannot be enabled from the UI.
- Installing non-market apps (raw APKs) is disabled, and cannot be enabled from the UI.
- No access to Google Play / Android Market (or any Google Apps).

This rules out 1) root APKs and 2) the majority of exploits out there that require executing commands over ADB. It means that one cannot run any code on the device that does not come from B&N period.

But of course there was another way in. Somebody on XDA-Developers had discovered that the bootloader of the Nook Tablet supported booting off an Android system located in partition images stored on an external microSD card. This mechanism is probably used to repair corrupted system partitions by B&N customer support.

The solution, then, is clear: we create dummy system partition images that, instead of booting an Android system, installs `su` and `Superuser.apk` into the "normal" Android system in internal flash memory. More concretely, I modified the system initialization file inside the initrd inside the boot partition image to invoke a custom script that copied the relevant files into the

`system` partition in the internal flash memory.

I based my work on bauwks's 2nduboot images and used abootimg to unpack files in the boot partition image `boot.img`:

```
# Extracts files in the boot partition
image into the current directory.
abootimg -x ./boot.img
# Extract files in the initrd cpio arc
hive into the folder ./ramdisk/
aboot-unpack-initrd ./initrd.img
```

I changed the system initialization file `init.omap4430.rc` in the initrd to mount the system partition of the internal flash memory at `/foo` rather than `/system`, because later system initialization steps attempt to remount `/system` as read-only and so on, and for some reason I could not disable that behavior:

```
on fs
    mkdir /foo
    mount ext4 /dev/block/platform/mmc
i-omap-hs.1/by-name/system /foo wait
```

I added the following to the system initialization file `init.rc` in the initrd to start my rooting script:

```
service root_script /sbin/busybox ash
/assets/run.sh
    oneshot
```

My rooting script, which I place in the directory `assets` in the initrd, installs not only `su`, but also Google Play and other

Google apps which are missing from the Nook Tablet ROM:

```
# Install su and Superuser.apk
/sbin/busybox cp /assets/su /foo/bin/
/sbin/busybox cp /assets/su /foo/xbin/
/sbin/busybox chmod 06755 /foo/xbin/su
/sbin/busybox chmod 06755 /foo/bin/su
/sbin/busybox cp /assets/Superuser.apk
/foo/app/

# Install Busybox.
/sbin/busybox cp /sbin/busybox /foo/xb
in/
/sbin/busybox chmod 06755 /foo/xbin/bu
sybox

# Install Google Play and other Google
apps.
/sbin/busybox cp /assets/*.apk /foo/ap
p/
/sbin/busybox cp /assets/com.google.an
droid.maps..xml /foo/etc/permissions/
/sbin/busybox cp /assets/com.google.an
droid.maps.jar /foo/framework/
/sbin/busybox cp /assets/libvoicesearc
h.so /foo/lib/

# Done.
/sbin/busybox mount -o ro,remount /foo
```

In accordance with the above script, I had placed all assets in the `assets` directory inside the initrd, and the Busybox binary in `sbin` inside the initrd. With all changes done, I pack everything back into a new boot.img:

```
# Build initrd cpio archive
abootimg-pack-initrd initrd.img.new
# Build new boot.img using previously
extracted components
abootimg --create boot.img.new -f ./bo
otimg.cfg -k ./zImage -r ./initrd.img.
new
```

Then I replace the `boot.img` on the SD card image with my `boot.img.new`, and the rooting method is done.

Final words

The actual process, of course, was much, much more painful. The Nook Tablet's bootloader is very picky; some microSD cards just won't work, there is a file size limit on `boot.img`, etc.. It was also after much frustration that I discovered the mount-to-`/foo` trick. And I could not even keep track of how many factory restores I had to perform on the device to undo bad modifications. But it was still a lot of fun, and the euphoria at the end and the feeling of accomplishment when reply posts started rolling in and random people were donating $5 as a token of their appreciation could not be overstated.

Good luck rooting!

Chuan Ji

/

15 Comments

1 Login

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

5 Share

Best Newest Oldest