Editing Functionality of Host Card Emulation in Android

I'm currently in the process of developing a project for my University course wherein I will be hopefully editing the functionality of the HCE Feature of Android to allow me to set my own UID when emulating a card.

Now, i've downloaded the AOSP source, and built a custom image with no edited code and installed that to my Nexus 7 (This includes downloading and including the Vendor specific hardware drivers), and i'm stuck on the next part.

I physically cannot find the device code that governs the NFC features of Android, and i'm unsure how to go about a) Looking for it, and b) How I should be editing this code.

Is the code for NFC in Android in the base Kernel? and if so how would I edit that before I run "make" again and hope it builds? or is it elsewhere? I've noticed that the files in the Vendor folder i've downloaded and extracted are in a .ncd format, which I don't think is editable.

Any help I can get on this would be greatly appreciated.

android kernel nfc hce

asked Feb 9 at 12:33

Jay Allen
26 ● 4

2 Answers

Ok ! So i've found a solution to the problem I was having!

On the Nexus 7, when the NFC is turned on, it gets its information from a config file in "/etc/" called "libnfc-brcm-20791b05.conf"

Inside of this file there is a parameter called "NFA_DM_START_UP_CFG"

By default, it looks like this:

```
NFA_DM_START_UP_CFG=
{42:CB:01:01:A5:01:01:CA:14:00:00:00:00:0E:C0:D4:01:00:0F:00:00:00:00:C0:C6:2D:00:14:0A:B5:0
```

To edit the UID that is generated at Emulation, you need to add some bytes to the end of this parameter.

The first byte you add is 0x33 (This means that you are going to manually set the UID)

The second byte that is added is the length of the UID you wish to set (This can be either 4,7 or 10 bytes, so this second byte can be 0x04, 0x07 or 0x0A)

The next bytes are then the ID you wish to set! (NOTE: Depending on how many Bytes you add, you should change the first byte of the array to reflect the new size of the array - it starts at 42, so if you were to add 6 bytes it should change to 48)

For example, if you wished to set a 7 byte ID of 01 02 03 04 05 06 07, the new config line would look like this:

```
NFA_DM_START_UP_CFG=
{4B:CB:01:01:A5:01:01:CA:14:00:00:00:00:0E:C0:D4:01:00:0F:00:00:00:00:C0:C6:2D:00:14:0A:B5:0
```

You can then push this config file to your nexus device using adb:

```
-> adb root
-> adb remount
-> adb push libnfc-brcm-20791b05.conf /etc/
-> adb reboot
```

This will reset the Nexus with the new config file in, and upon emulation the UID will now be set to 01 02 03 04 05 06 07

Hope this helps anyone reading my question!

answered Feb 25 at 13:32

Jay Allen
26 • 4

Hi, I tried to fixe the UID of my OnePlus One on the libnfc-brcm.conf file but doesn't seems to works... :(Do you have any idea of the right configuration for the OnePlus ? – Spawnrider Jun 22 at 14:43

Android's NFC stack is basically split into five parts:

- The NFC interface device driver. This is part of the kernel. In a nutshell, this driver simply tunnels data frames (e.g. NCI protocol frames) between a character device file and the NFC controller hardware. You won't have to touch that part for your project.
- The low-level interface library written in C (libnfc-nci, or libnfc-nxp for devices with NXP's PN544 NFC controller). This library provides a set of high-level functions to interact with the NFC controller. So it basically translates high-level functionality (e.g. "start polling for technologies X, Y and Z") into NCI commands that are sent to the NFC controller through the kernel driver. This is certainly a place where you will need to add modifications. As it's part of AOSP you can compile it using the normal AOSP build system.
- The JNI interface library written in C++ (libnfc_nci_jni). This layer connects the libnfc-nci C library with high-level Java code. If you want to modify the emulated UID from Android apps, this is certainly a place where you will need to add modifications. As it's part of AOSP you can compile it using the normal AOSP build system.
- The Android NFC system service written in Java. This service takes control over the whole NFC stack and implements the high-level functionality based on the resources provided by libnfc-nci. If you want to modify the emulated UID from Android apps, this is certainly a place where you will need to add modifications. As it's part of AOSP you can compile it using the normal AOSP build system.
- The Android core framework provides an API to the functionality of the NFC system service that can be accessed by Android apps.

With regard to setting/modifying the emulated UID you will certainly want to have a look at these projects that I recently published on GitHub (though they are still work in progress):

- https://github.com/mobilesec/swr50-android-external_libnfc-nci
- https://github.com/mobilesec/swr50-android-packages_apps_nfcwatch1

edited Mar 30 at 17:29

answered Feb 9 at 16:29

Michael Roland
13.2k • 4 • 23 • 49