CS9223 – Mobile Security

Donald Raikes and Brandi Crawford

Team Project 04/25/2014

Near Field Communication

Near field communication is a relatively new technology. It is an outgrowth from RFID, and employs very short distance electromagnetic communication similar to Bluetooth. Over the last few years a number of smartphones have been equipped with NFC chips which allow them to do a variety of functions. In this paper, we will discuss not only how nFC works, but also how its capabilities can be explored without having an actual NFC-enabled device.

NFC can perform any of four major kinds of functions:

* Reading/Writing NFC tags
* Pairing of devices
* Sharing of information
* Making payments

NFC tags are passive circuits which can be embedded into almost anything from a tag on merchandise in a store, to posters at a theaters. These circuits have no power source of their own, but when an NFC-enabled device such as a smartphone comes within roughly 1 inch of the circuit, the electromagnetic field created by the NFC chip in the phone induces an electrical current in the NFC tag which can then “transmit” a small amount of data to the smartphone. Actually the phone reads the information off of the tag.

There are a variety of different types of NFC tags which can contain varying amounts of data. The simplest can only hold 512 bytes of data while some tags can hold as much as 32k bytes of data. The types of data stored on the tags could be as simple as a website’s URL for more information about a product, or as detailed as technical information about an exhibit in a museum.

Smartphones already have the capability to be paired with other devices like speakers or headphones but the process involves enabling Bluetooth on the phone and doing some configuration steps. With an NFC enabled phone and external device, all that has to happen is that the phone is brought to within 1 inch of the device and voila, they are paired and the Bluetooth connection is established. At present this is probably the most common use of NFC technology.

NFC also makes it very simple for two NFC-enabled devices to exchange information. You have probably seen the commercials for android phones where two android users just “bump” their phones and share contacts or music playlists. This peer-to-peer sharing is done via NFC. Both devices need to have NFC-enabled applications running and in the foreground of the phones like the contact list or music player, and then when the phones are “bumped” or brought within the 1-inch distance, the data is sent from one phone to the other.

The final type of function that an NFC-enabled phone can perform is making payments. This is becoming more wide-spread in Europe and Asia although it is not widely adopted yet in the United States.

In this type of functionalty, a customer can take his/her purchases to an NFC-enabled cash register and “wave” his phone over the NFC sensor and it will transfer information about the customer’s credit card to the cash register. The customer then confirms the transaction and the payment is completed. This capability requires that the customer enter hhis credit card information into an application like Google Wallet or one of its competitor apps that are NFC enabled. The organization which promotes NFC (<http://nfc.org>) invisions that NFC-enabled phones will eventually replace the traditional wallet full of credit cards since all credit card information will be in the phone and all vendors will accept this kind of payment.

The payment capability is also being used for situations like paying tolls in place of using something like an easypass card.

Exploring NFC:

Since NFC-enabled phones are still very expensive and not all mobile providers carry them, the Open-NFC project (<http://www.open-nfc.org>) has developed a simulator which allows a user to explore the capabilities of NFC within a simulated environment. The simulator works with a specially compiled android rom that includes the open-nfc implementation of NFC. The android rom is run in the android SDK’s emulator and connects to the NFCC Simulator via a third application the open-nfc connection-center.

Unfortunately, the most recent version of open-nfc (4.5.2) is missing a special app which allows the user to enable open-nfc on the emulated phone so there is no way to test with this version of open-nfc. After a lot of research and experimentation I discovered that the oldest version of the open-nfc android-sdk available on the open-nfc download page (<http://open-nfc.org/wp/home/downloads/>) which has a rom compiled with android 4.0.3 does contain all the necessary pieces. This version of open-nfc is 4.4.1.

Setting up the test environment is a little tricky the first time, but once al lthe pieces are in place, it is just a matter of launching several application and the user is ready to test.

Step 1: Download and install the latest version of the Android SDK from <http://developer.android.com/sdk/index.html>. I used the ADT bundle for windows. I extracted it to c:\adt for the sake of simplicity.

2. Download the open-nfc android-sdk 4.4.1 from (<http://open-nfc.org/wp/home/downloads/> and unzip it to somewhere easy to find. I extracted this to c:\adt\nfc441.

3. Update the android SDK with some android images and their associated development tools. Use the SDK Manager found in the top-level folder of the SDK and select the packages you wish to include. For this project I included android version 4.0.3 and its tools since I was going to be creating an android virtual device based on version 4.0.3.

Once the updates have downloaded and installed, you need to create an android virtual device (AVD). The simplest and cleanest way to do this is to open a command prompt and switch to the sdk\tools folder (for me c:\adt\sdk\tools). Run the command:

c:\ android list targets

This will list all targets that you can use to create AVDs. Depending on how many versions of android you downloaded in the SDK Manager, you could have just one or multiple entries. Take note of the id number for the target associated with android 4.0.3.

Now run:

C:\ android create avd –n <avd name> -t <target\_id>

You can specify any name you want for the avd, but do not include any spaces.