Module 3 – NFC Intents

Scan this tag to compromise the application

MALICIOUS ERECTION LLC

- This module requires a device with NFC capabilities
- Additionally, a NFC tag with re-writable capabilities will also be required
  - Tags which are ISO 14443A certified will work
- If you do not have access to these requirements, this module cannot be completed
- Skip this module if you do not meet the requirements

MALICIOUS ERECTION LLC

NFC Intents

- At a high level, Android has 3 NFC functions
  - Reader/writer mode – allows device to read and/or write passive NFC tags
    - When the NFC antenna is turned on, then the Android device is always listening for new NFC tags to scan
  - P2P mode – allows devices to exchange data
  - Card emulation mode – allows devices to emulate an NFC tag
    - This is how Google Pay / Samsung Pay works
    - Your device emulates a tap-to-pay credit card
- If an Android device detects and reads a NFC tag, Android will attempt to create an Intent based around the NFC data
- Then, just like other Intents, Android will execute the Intent
- NOTE: the type of data that can be stored on a NFC tag is limited
  - More information on how Android interprets NFC tags:
    - https://developer.android.com/guide/topics/connectivity/nfc/nfc

NFC Intents

- An Activity can be opened with an NFC Intent if the Activity has declared one of the below Intent Actions in its Intent Filter:
  - `android.nfc.action.NDEF_DISCOVERED`
  - `android.nfc.action.TECH_DISCOVERED`
  - `android.nfc.action.TAG_DISCOVERED`
- For example, below is a screenshot that shows an example Intent Filter with the `android.nfc.action.NDEF_DISCOVERED` Action declared

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED" />
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain" />
</intent-filter>
```

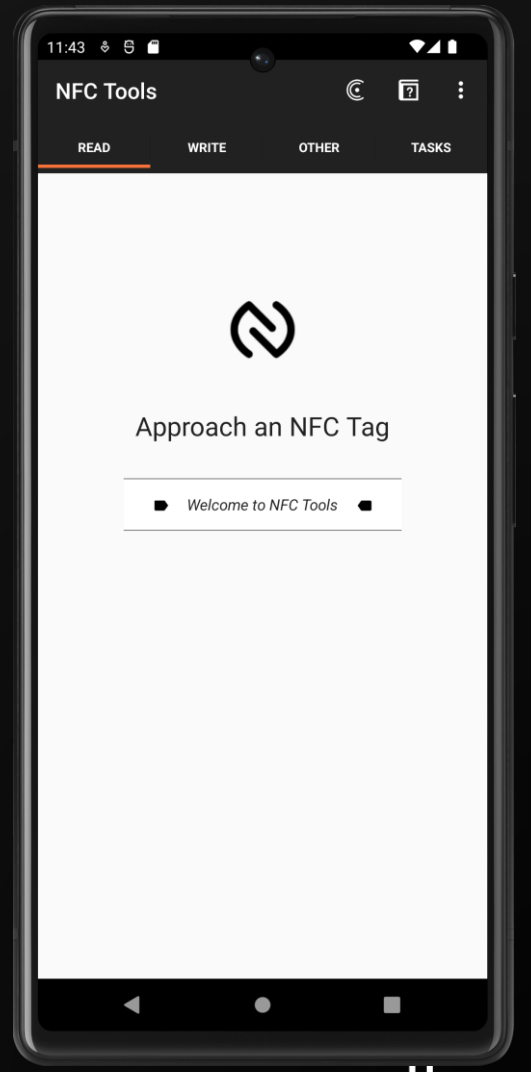
Android Developer documentation showing an example Intent Filter with the `NDEF_DISCOVERED` Intent Action

MALICIOUS ERECTION LLC

NFC Intents

- There is a free Android application that can help read and write NFC tags
  - Name: NFC Tools
  - Downloadable from the Google Play store
    - https://play.google.com/store/apps/details?id=com.wakdev.wdnfc
    - An `.apk` file for this application has also been provided with this course
- We will be using this application to write custom NFC tags in order to interface with Axolotl
- Take a moment to download and install this app before proceeding

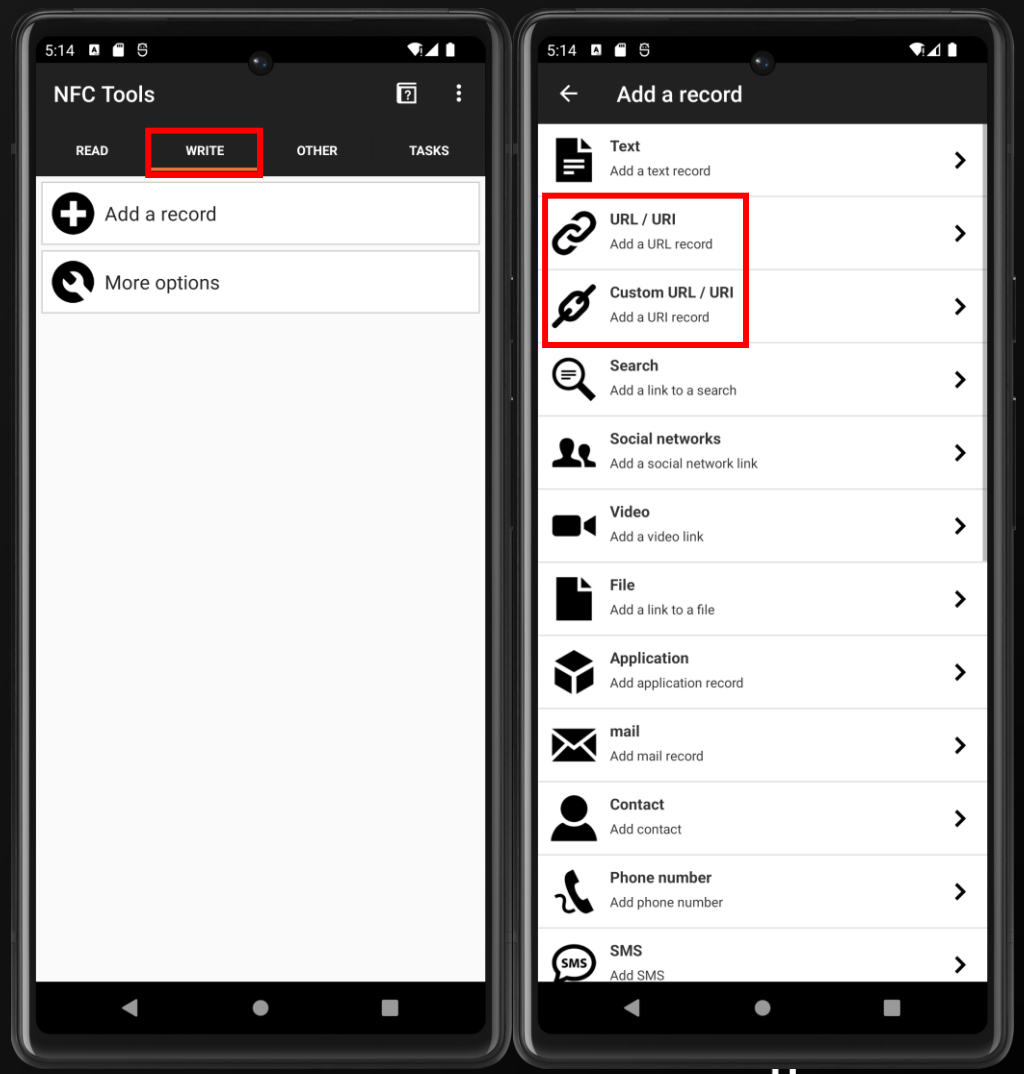
NFC Tools application



NFC Intents

- The NFC Tools application can write certain types of data to NFC tags
- For example, the application can be used to write Uri Data to a NFC tag
- In NFC Tools, tap:
  - “Write”
  - “Add a record”
  - “URL / URI”
- The “URL / URI” has a pre-set list of Uri schemes
  - If you want a custom scheme, you can use “Custom URL / URI” instead

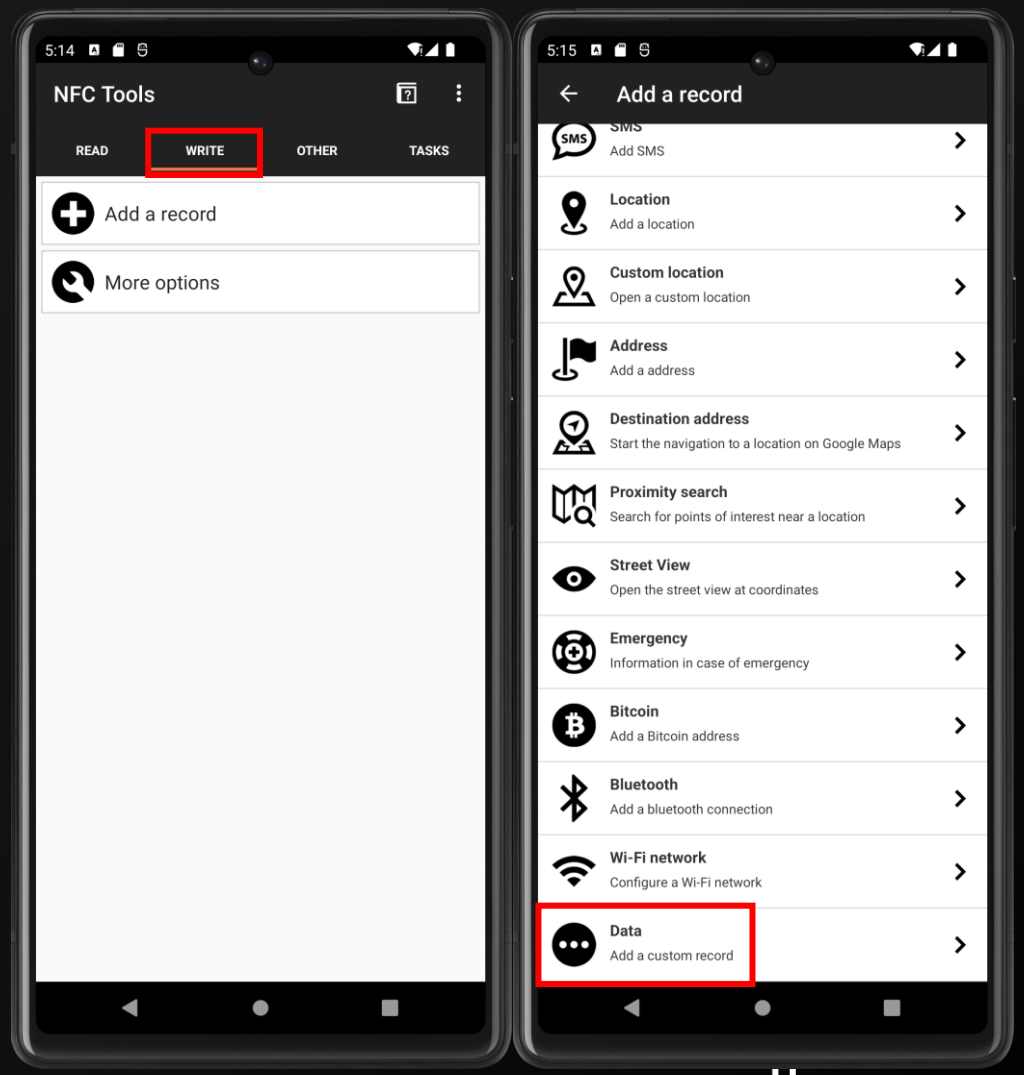
NFC Tools application



NFC Intents

- When it comes to raw data, you will need to specify a MIME type and the data itself
  - For example, the MIME type of a .txt file is `` `text/plain` ``
- You can essentially write an entire file to NFC tags and have the Android OS interpret and process the file
- In NFC Tools, tap:
  - “Write”
  - “Add a record”
  - “Data”

NFC Tools application



MALICIOUS ERECTION LLC

NFC Intents - Example

- We will now utilize Axolotl to better demonstrate how to use NFC Tags to work with NFC Intents
- On Axolotl's main menu, tap:
  - "Exercise Modules"
  - "NFC Intents"
- A blank activity will appear with some text
  - The launched activity is programmed via the Java class `com.maliciouserection.axolotl.example.activity.nfc.nfcIntent`

MALICIOUS ERECTION LLC

NFC Intents - Example

- According to Axolotl's Manifest, the Activity `com.maliciouserection.axolotl.example.activity.nfc.nfcIntent` can be opened via NFC tags
- First, we will open it with a NFC tag embedded with a custom Uri
- Per Axolotl's Manifest file, a valid Uri should contain the scheme `` `axolotlnfcscheme` `` and the host `` `axolotlnfchost` ``
- So, the Uri should start with `` `axolotlnfcscheme://axolotlnfchost` ``

```
<activity android:name="com.maliciouserection.axolotl.example.activity.nfc.nfcIntent" android:exported="true">
    <intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="axolotlnfcscheme" android:host="axolotlnfchost"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="application/axolotlmimetype"/>
    </intent-filter>
</activity>
```

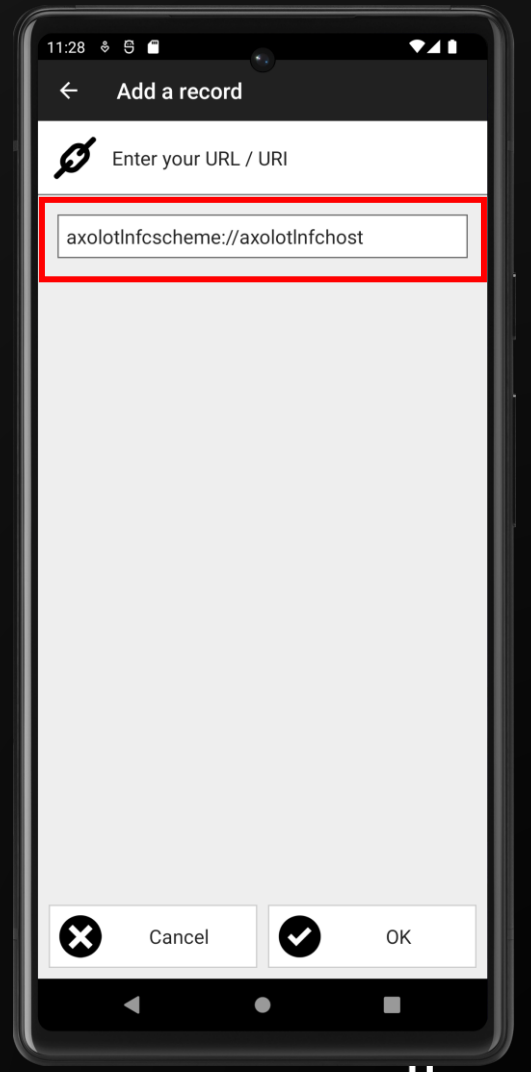
Manifest of Axolotl highlighting the custom Uri scheme and host

MALICIOUS ERECTION LLC

NFC Intents - Example

- Let's make a NFC tag that can use the custom Uri
- Open "NFC Tools" and perform the following actions:
  - Tap "Add Record"
  - Tap "Custom URL/URI"
  - Type the custom Uri `axolotlnfcscheme://axolotlnfchost`
  - Tap "OK"
  - Tap "Write"
  - Put the writable NFC tag on your device to write the tag
- Minimize or close the "NFC Tools" application and tap the NFC tag against the device to scan the tag
- Hopefully, the Activity `nfcIntent` launches

NFC Tools application



NFC Intents - Example

- When `nfcIntent` launches, you should see the custom Uri presented on the screen
- If you decompile `com.maliciouserection.axolotl.example.activity.nfc.nfcIntent`, you should see that the Activity presents additional data based on what the Uri value was on the NFC tag

```
private void theMainMethod() {
    Intent yayintentyay = getIntent();
    if (yayintentyay.getData() != null) {
        Uri yaydatayay = yayintentyay.getData();
        this.text.setText("theMainMethod - getIntent().getData().toString(): " + yaydatayay.toString());
        List<String> yaylistyay = getIntent().getData().getPathSegments();
        if (yaylistyay.size() > 0) {
            String yaysegmentyay = yaylistyay.get(1);
```

Decompiled `nfcIntent` with the appropriate code highlighted

Axolotl Application

MALICIOUS ERECTION LLC

NFC Intents - Example

- Looking at Axolotl's Manifest again, the Activity `com.maliciouserection.axolotl.example.activity.nfc.nfcIntent` has another Intent Filter
- The Intent Filter specifies that the NFC tag's data must have a MIME type of `` `application/axolotlmimetype` ``
  - This technically means that the NFC tag must contain Raw Data
- So let's craft a NFC tag with data that uses that MIME type

```
<activity android:name="com.maliciouserection.axolotl.example.activity.nfc.nfcIntent" android:exported="true">
    <intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="axolotlnfcscheme" android:host="axolotlnfchost"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="application/axolotlmimetype"/>
    </intent-filter>
</activity>
```

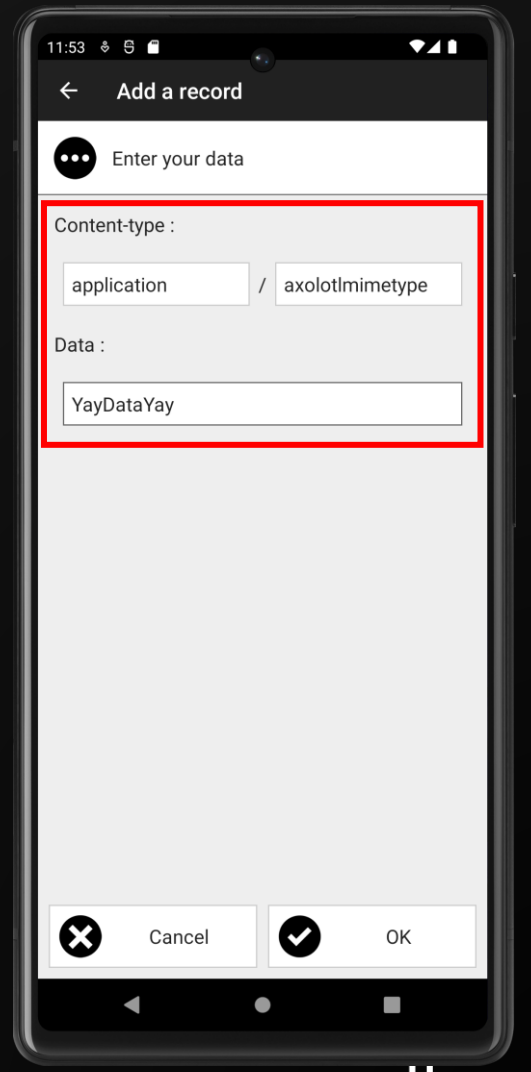
Manifest of Axolotl highlighting the custom Uri scheme and host

MALICIOUS ERECTION LLC

NFC Intents - Example

- Open “NFC Tools” and perform the following actions:
  - Tap “Add Record”
  - Tap “Data”
  - Type the following `Content-type`
    - `application/axolotlmimetype`
  - Type any data you want under `Data`
  - Tap “OK”
  - Tap “Write”
  - Put the writable NFC tag on your device to write the tag
- Minimize or close the “NFC Tools” application and tap the NFC tag against the device to scan it
- Hopefully, the Activity `nfcIntent` launches

NFC Tools application



MALICIOUS ERECTION LLC

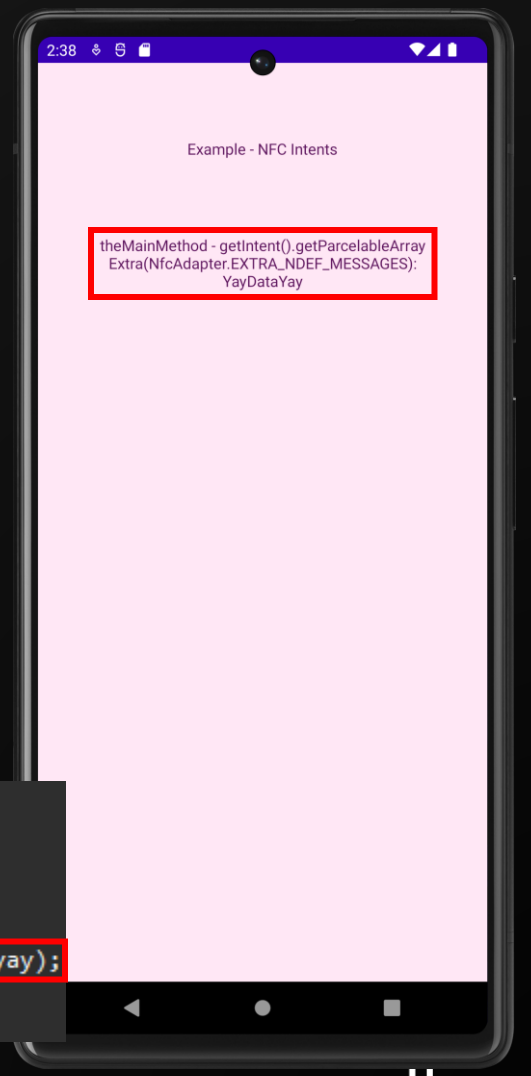
NFC Intents - Example

- When `nfcIntent` launches, you should see the custom data presented on the screen
- If you decompile `com.maliciouserection.axolotl.example.activity.nfc.nfcIntent`, you should see that the Activity presents additional information based on what Raw Data was on the NFC tag

```
if (rawMessages != null) {
    NdefMessage[] messages = new NdefMessage[rawMessages.length];
    for (int i = 0; i < rawMessages.length; i++) {
        messages[i] = (NdefMessage) rawMessages[i];
        String yaystringyay = random functions.getAsciiFromHex(messages[i].getRecords()[0].getPayload());
        this.text.setText("theMainMethod - getIntent().getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES): " + yaystringyay);
        Uri yayuriyay = Uri.parse(yaystringyay);
        if (yayuriyay.getScheme() != null) {
```

Decompiled `nfcIntent` with the appropriate code highlighted

Axolotl Application



MALICIOUS ERECTION LLC

Module 3 Exercise

- The Activity `com.maliciouserection.axolotl.example.activity.nfc.nfcIntent` had the methods `` `aSecondMethod()` `` and `` `aThirdMethod()` ``.
  - Craft two NFC tag payloads that would execute both methods.
    - You will need one payload for `` `aSecondMethod()` `` and a separate payload for `` `aThirdMethod()` ``.
- Capture The Flag - The activity `com.maliciouserection.axolotl.activity.nfc_activity` contains two different flags. Craft two different NFC tag payloads that will retrieve both flags.
  - “Flag 3.1” is retrieved via a Uri NFC Tag
  - “Flag 3.2” is retrieved via a raw data NFC Tag

MALICIOUS ERECTION LLC