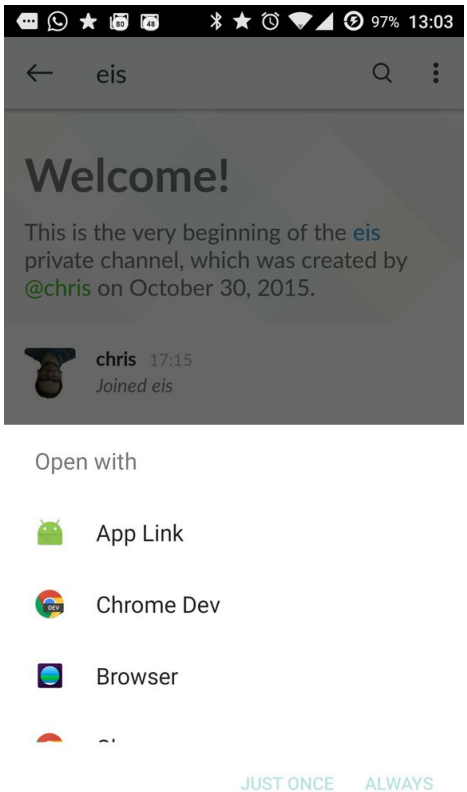
Dokumentation PoC App Links

Die Implementierung von App Links erwies sich als deutlich einfacher als zu Beginn erwartet, da diese von Android voll unterstützt werden und sehr gut dokumentiert sind (siehe http://developer.android.com/training/app-links/index.html).
Es besteht die Möglichkeit, bestimmten Links verschiedenen Activities der App zuzuweisen (in unserem Fall reicht es allerdings, einen Link einer Activity zuzuweisen). Diese Zuweisung wird einfach in der Manifest.xml mit nur wenigen Zeilen Code festgelegt:

```
<intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="http" android:host="app.chrispop.de"/>
</intent-filter>
```

Hier weisen wir testweise allen Links, die mit “http://app.chrispop.de” beginnen die gewünschte Activity unserer App zu (in diesem Testprogramm besteht die App nur aus einer einzigen Activity).
Natürlich ist unsere App nicht die einzige Anwendung, die für das Öffnen dieses Links in frage kommt (andere wären hier z.B. die auf dem Gerät installierten Browser), deswegen werden alle Apps, die sich beim System dafür angemeldet haben, diesen Link weiter zu verarbeiten als Vorschläge in einer Liste angezeigt.

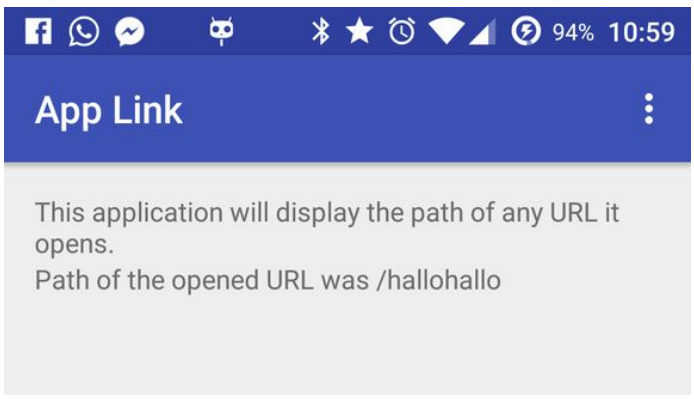


Unter Android M bestünde außerdem die Möglichkeit, eine App ohne Nachfrage für diese Art von Link festzulegen. Hier ist eine Verifizierung mit dem Server über ein dort gehostetes und mit SSL verschlüsseltes JSON-File nötig, die wir im Rahmen dieses Projektes unter Betrachtung des engen zeitlichen Rahmens jedoch außen vor lassen.

Des weiteren wird es für den Prozess der Verifikation nötig sein, zwei Keys zu vergleichen und so zu verifizieren, dass der Nutzer der Anwendung Zugang zur angegebenen Mailadresse hat.
Einer der Keys wird hierbei auf dem System des Benutzers gespeichert, der andere wird über eine URL an die angegebene Mailadresse geschickt.
Das Zugreifen auf den Pfad einer mit der App geöffneten URL stellt sich als kein großes Problem dar:

```
// Handling the incoming intent
Intent intent = getIntent();
String action = intent.getAction();
Uri data = intent.getData();

// Check if data is null to prevent NullPointerException
if(data != null) {
    String path = data.getPath();

    // Debug
    System.out.println(action);
    System.out.println(data);
    System.out.println(path);

    TextView pathText = (TextView) findViewById(R.id.pathText);
    pathText.setText("Path of the opened URL was " + path);

}
```

In unserem Beispiel wird der Pfad der geöffneten URL lediglich in einem TextView ausgegeben, es ist jedoch auch möglich hier anderweitig mit dem extrahierten Pfad zu arbeiten.



Alle verwendeten Funktionen sind in der Android API dokumentiert und haben damit einen hohen Anspruch auf Sicherheit und Funktionalität.