

```
manager.requestLocationUpdates(LocationManager.GPS_PROVIDER, minTime: 0, minDistance: 0,
    new LocationListener() {
        @Override
        public void onLocationChanged(
            @LocationAnnotation(
                purpose = {LocationPurpose.provide_location_based_content},
                purposeDescription = {"Provide location-based weather information"},
                visibility = {Visibility.WHILE_IN_USE},
                frequency = {"The location will be updated as fast as possible"},
                dataType = {LocationDataType.FINE_GRAINED_LATITUDE_LONGITUDE})) Location locat
    }
}
```

AnnotationExampleActivity > onCreate

Checker: Personal Data Access Overview

LOCATION
request location update Data Type: LocationDataType.FINE_GRAINED_LATITUDE_LONGITUDE. Purpose: (Loca)

UNIQUE_IDENTIFIER

CONTACTS

Id TODO 6: Logcat 9: Version Control Ten PrivacyChecker

Coconut: An IDE Plugin for Developing Privacy-Friendly Apps

Tianshi Li

Yuvraj Agarwal

Jason I. Hong

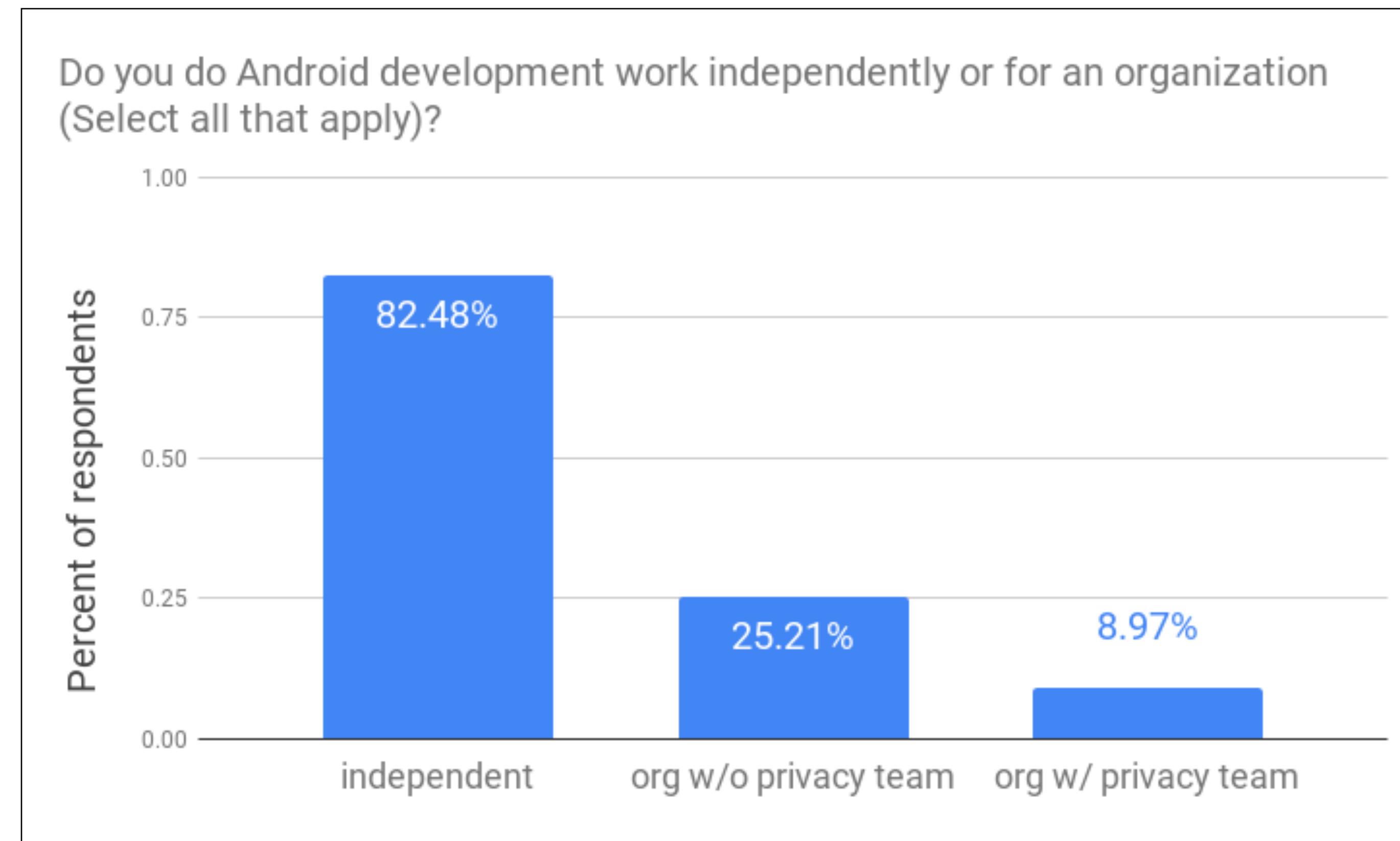
Carnegie Mellon University

Carnegie Mellon University

Carnegie Mellon University



Only 8.97% developer survey respondents have worked with a privacy team



According to a survey study we recently conducted with developers randomly sampled on Google Play (N=234)

**How can we support (independent)
Android developers to adopt good
privacy practices?**

Limitation in prior work

- Privacy research for developers is still at an early stage.
- **Heavily focused on** developers' attitudes/knowledge/habits that cause **security** issues, while **very little in-depth examination about** different aspects of **privacy** issues.
- A lot of study on the high-level privacy/security attitudes, while they offer little help for designing developer tools to address these issues.
- Very little work explored building developer tools for privacy.

Semi-structured interview (N=9)

- Part 1 (Study **high-level challenges in understanding**): Asked general questions about their app development experience, privacy training background, and perceptions about privacy.
- Part 2 (Study **concrete challenges during programming**): Asked about recent apps (up to 3 apps) they had developed.
 - Specifically, we asked whether certain categories of personal data were acquired from these apps, and, if so, how and why.
 - We tested the apps before or during the interview to check their statements.

Challenge 1: Partial understanding of privacy

- Obtain user consent before collection (P1)
- Avoid using PII (P1)
- minimizing data usage (P2, P6, P7, P8)
- encrypting or obfuscating data before egress (P4, P8).

Challenge 2: Inaccurate understanding of app behaviors

- Insufficient understanding of how some API works, especially ad libs
- Varying data practices in different versions
- Developers come and go, but data practices are not well-documented

Challenge 3: Lacking privacy knowledge

- None of them heard about the “Best practices for unique identifiers” which is listed in the official Android documentation

Challenge 4: Collecting unneeded data when lacking constraints

- Requested multiple types of data controlled by the same permission (P3, P6)
- Requested sensitive data for multiple purposes, while only explained partial reasons to users (P5)

Privacy challenges faced by developers

Partial understanding of privacy

Inaccurate understanding of app behaviors

Lacking privacy knowledge

Collecting unneeded data when lacking constraints

Designing privacy annotations to help address the four challenges

- Privacy annotations are custom Java annotations that are designed to:
 - guide developers to think through privacy risks
 - document privacy practices (especially hard-to-analyze factors such as purposes)
 - suggest better privacy practices
 - make privacy practices more transparent

```
@LocationAnnotation(  
    purpose = {LocationPurpose.provide_location_based_content},  
    purposeDescription = {"Get local weather info"},  
    visibility = {Visibility.WHILE_IN_USE},  
    frequency = {"One-time access"},  
    dataType = {LocationDataType.COARSE_GRAINED_LATITUDE_LONGITUDE}  
)  
Location location = locationManager.getLastKnownLocation(  
    LocationManager.NETWORK_PROVIDER);
```

Privacy annotations corresponding to info flows

```
@LocationAnnotation(  
    purpose = {LocationPurpose.provide_location_based_content},  
    purposeDescription = {"Get local weather info"},  
    visibility = {Visibility.WHILE_IN_USE},  
    frequency = {"One-time access"},  
    dataType = {LocationDataType.COARSE_GRAINED_LATITUDE_LONGITUDE}  
)  
Location location = locationManager.getLastKnownLocation(  
    LocationManager.NETWORK_PROVIDER);
```

Source Annotations

-  @UserFile
-  @Camera
-  @Location
-  @Microphone
- ...

```
@LocationAnnotation(  
    purpose = {LocationPurpose.provide_location_based_content},  
    purposeDescription = {"Get local weather info"},  
    visibility = {Visibility.WHILE_IN_USE},  
    frequency = {"One-time access"},  
    dataType = {LocationDataType.COARSE_GRAINED_LATITUDE_LONGITUDE}  
)  
@NetworkAnnotation(  
    retentionTime = "Not stored",  
    destination = {"OpenWeather API server"},  
    purposeDescription = {"Update current local weather"},  
    encryptedInTransmission = true)  
JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET,  
    url: currentWeatherURL + locationString + "test",
```

Sink Annotations

-  @Network
-  @Storage
- ...

Coconut enforces and facilitates developers to provide privacy annotations

```
questQueue = new RequestQueue(cache, network);
questQueue.start();

locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
(locationManager != null) {
    if (ActivityCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED)
        return;
}
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, minTime: 0, minDistance: 0,
    new LocationListener() {
        LocationAnnotation annotation is required. more... (⌘F1)
        public void onLocationChanged(Location location) {

        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {

        }

        @Override
        public void onProviderEnabled(String provider) {

        }

        @Override
        public void onProviderDisabled(String provider) {
```

Privacy lint: nudge towards best privacy practices

The screenshot shows an Android Studio code editor with the following code snippet:

```
94
95
96
97
98     @UniqueIdentifierAnnotation(
99         purpose = {UIDPurpose.tracking_user_data_collected_within_this_app},
100        purposeDescription = {"Tracking user data for analysis"},
101        uidType = {UIDType.ANDROID_ID},
102        scope = {UIDScope.PER_DEVICE},
103        resetability = UIDResetability.RESET_WHEN_FACTORY_RESET)
104
105
106
107     private void startNetworkTraffic(String runRouteRecord) {
108         JsonObjectRequest networkRequest = new JsonObjectRequest(Request.Method.GET,
109             url: targetURL + runRouteRecord, jsonRequest: null, new Response.Listener<JSONObject>()
110             @Override
111             public void onResponse(JSONObject response) {
112             }
113         }, new Response.ErrorListener() {
114             @Override
115             public void onErrorResponse(VolleyError error) {
116             }
117         );
118         mRequestQueue.add(networkRequest);
119     }
120
121
122 }
```

A callout box highlights the annotation at line 103, which includes a lightbulb icon indicating a suggestion. A tooltip message below the code states: "The more appropriate scopes of the unique identifier for the purpose "UIDPurpose.tracking_user_data_collected_within_this_app" are PER_APP" followed by a link to "more...".

The code editor interface includes:

- Left sidebar icons: Captures (camera), Build Variants (Android), Favorites (star).
- Bottom navigation bar: TODO, Logcat (6), Version Control (9), Terminal, PrivacyChecker, Build (0/10).

Privacy overview panel: auto-generated, interactive privacy documentation

The screenshot shows the Android Studio interface with the Privacy Checker plugin open. The main area displays Java code from `AnnotationExampleActivity.java` with annotations like `@Override`. Below the code, the `onCreate()` method is expanded. The `PrivacyChecker` tab is selected in the navigation bar. The sidebar lists categories of permissions: LOCATION, UNIQUE_IDENTIFIER, CONTACTS, CALENDAR, CAMERA, MICROPHONE, CALL_LOG, SENSORS, SMS, USER_DATA, and OTHER_PERSONAL_DATA. The `LOCATION` category is expanded, showing a single entry: "request location update Data Type: LocationDataType.COARSE_GRAINED_LATITUDE_LONGITUDE. Purpose: (LocationPurpose.provide_location_based_content)". This entry is highlighted with a blue background and has a cursor arrow pointing to it. The bottom navigation bar includes icons for TODO, Logcat, Version Control, Terminal, PrivacyChecker (selected), and Build.

```
74     purpose = {LocationPurpose.provide_location_based_content},  
75     purposeDescription = {"Provide location-based weather informat  
76     dataType = {LocationDataType.COARSE_GRAINED_LATITUDE_LONGITUDE  
77     visibility = {Visibility.WHILE_IN_USE},  
78     frequency = {"The location will be updated as fast as possible  
79     Location location) {  
80         }  
81     }  
82     }  
83     @Override
```

AnnotationExampleActivity > onCreate()

PrivacyChecker: Personal Data Access Overview Personal Data Leak Overview

LOCATION
request location update Data Type: LocationDataType.COARSE_GRAINED_LATITUDE_LONGITUDE. Purpose: (LocationPurpose.provide_location_based_content)

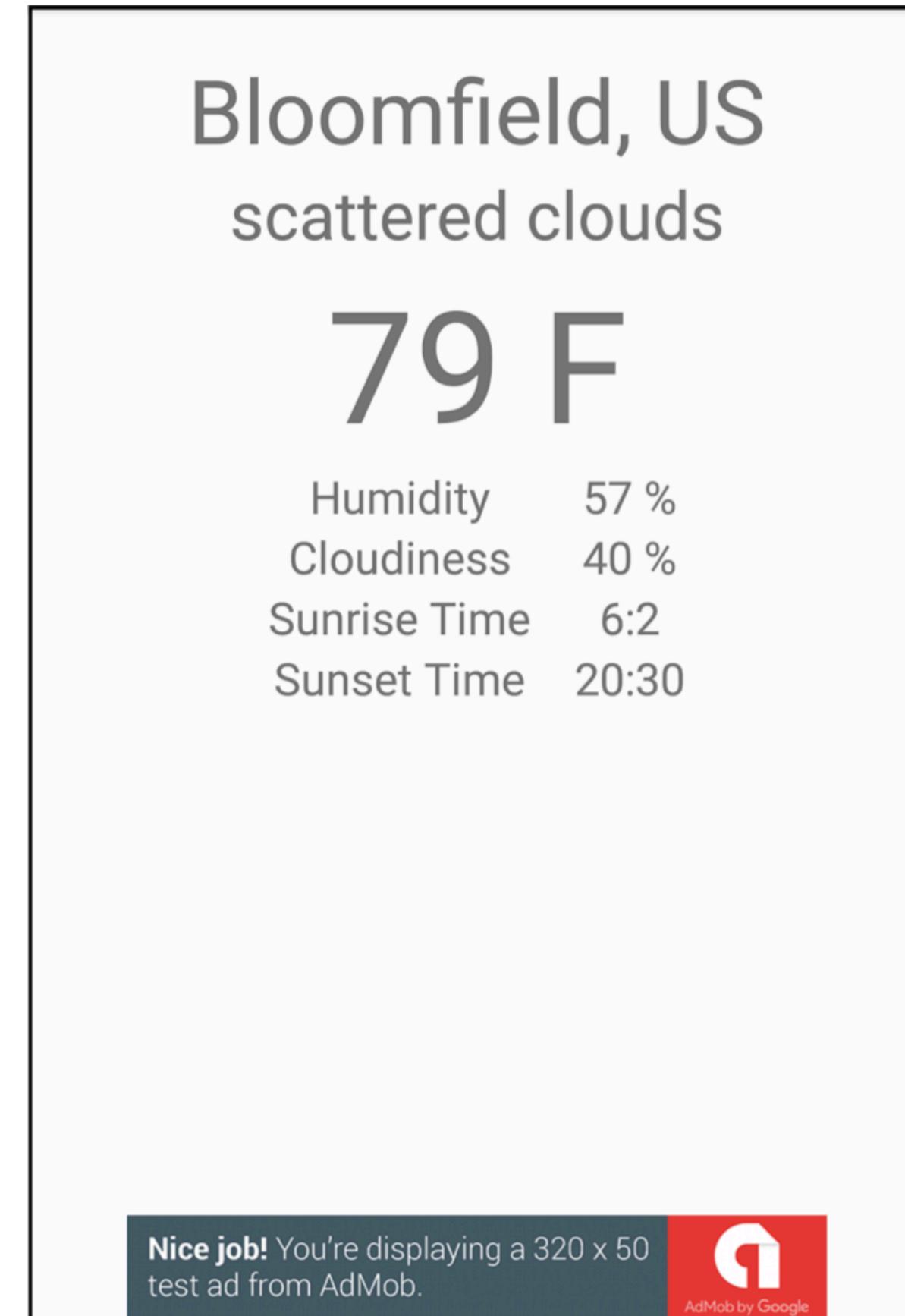
UNIQUE_IDENTIFIER
Google Instance ID Data Type: UIDType.INSTANCE_ID. Purpose: (UIDPurpose.tracking_user_data_collected_within_this_app) "Tracking user data"

CONTACTS
CALENDAR
CAMERA
MICROPHONE
CALL_LOG
SENSORS
SMS
USER_DATA
OTHER_PERSONAL_DATA

TODO 6: Logcat 9: Version Control Terminal PrivacyChecker Build

Evaluating Coconut via a lab study

- 18 developers, between-subjects design
- Warm-up task: Obtain the current lat-long location data and display it on the screen
- Main task (the UI and weather API were pre-implemented):
 - Obtain the current lat-long location data to update the current weather.
 - Store the location data locally for future analysis with a unique identifier.
 - Implement a banner ad using Google Admob.



A screenshot of the weather app in the main task

Lab evaluation result highlights

- **Result 1: Privacy annotations were perceived useful and usable**
 - Developers perceived high usefulness and low disruptiveness and time spent on the annotating work.
- **Result 2: Coconut helped developers write more privacy-preserving code**
 - In the control group, only 36.7% of implemented features followed best privacy practices, while 77.8% followed best practices in the Coconut group.
- **Result 3: Coconut helped developers better understand the app's behavior**
 - The correct rate of the factual questions in the post-study survey were 66.7% and 88.1% respectively for the control group and the Coconut group
- **Result 4: Coconut helped developers write better privacy policies**

Summary of our contributions

- Four challenges for handling privacy: incomplete understanding of privacy, inaccurate understanding of app behavior, lacking privacy knowledge, lacking constraints.
- We present Coconut, an IDE plugin that helps Android developers handle privacy.
- Annotations provide a low-cost and natural method to help document privacy practices and convey immediate privacy feedback while programming
- **Check out our website:** <https://coconut-ide.github.io>
 - We release a pre-compiled version and the source code of the plugin and annotation library, and an example Android app so you can try it out yourself!



Thanks!

Tianshi Li (tianshil@cs.cmu.edu)

<https://tianshili.me> <https://coconut-ide.github.io>



Human-
Computer
Interaction
Institute



institute for
SOFTWARE
RESEARCH