EC521 Cyber Security

ANDROID APP VULNERABILITY ANALYSIS

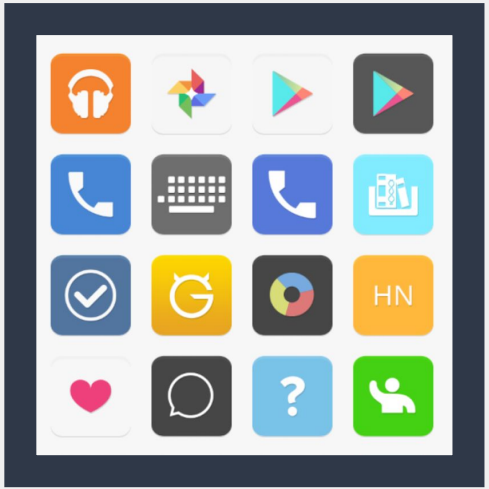
Carlton Duffett
Luke Sorenson
Igor de Paula
Petar Ojdrovic
Konstantino Sparakis
Zachary Lister

Introduction

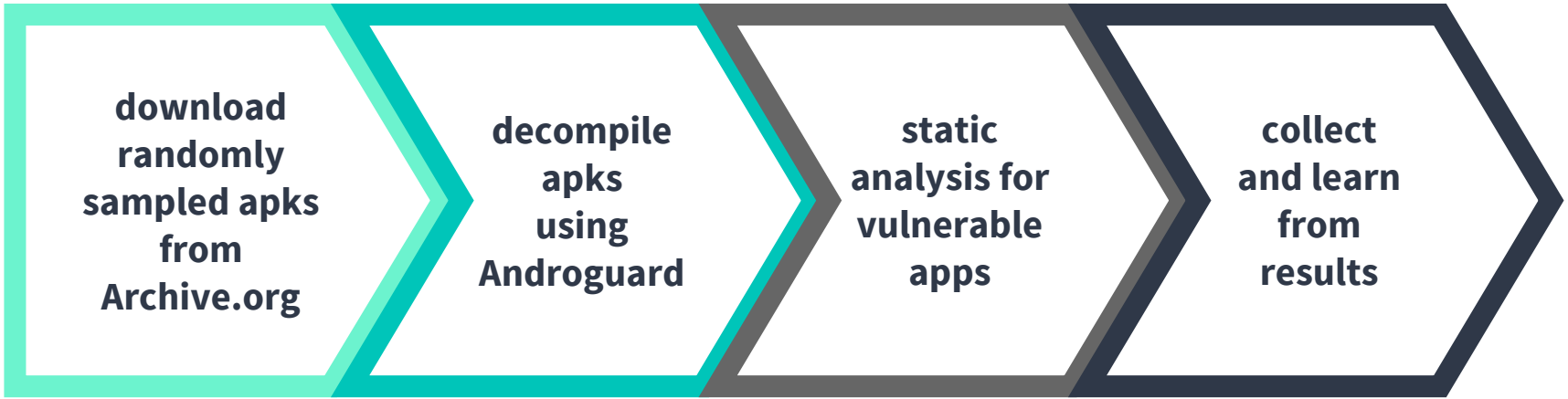
what we’re all about

Our Goal

Test a random sampling of apps
in the Android marketplace
to gain a practical understanding
of the extent of common
vulnerabilities



Our Analysis Process



Our Toolkit

Key Components:

1. **crawler**
   downloads n randomly sampled APKs from Archive.org
2. **disassembler**
   uses Androguard to disassemble the APKs into source code
3. **testing modules**
   tests for a small subset of common vulnerabilities
4. **sqlite backend**
   stores metadata and results

Crawling Archive.org

sampling apks in the marketplace

The PlayDrone Project

Created at Columbia University

June 2014

Deployed Independently by Archive.org

October 2014

13 Snapshots of the Android Marketplace

Latest snapshot on October 31, 2014

PlayDrone APK's

This collection contains Android APK's crawled from the Google Play store using PlayDrone.

Share
Favorite

About | Collection

SORT BY VIEWS · TITLE · DATE ARCHIVED · CREATOR

255 RESULTS

Search this Collection

software 255

PART OF
Android Apps


playdrone-apk-e8
9,315 | 0 | 0


playdrone-apk-ee
7,741 | 1 | 0


playdrone-apk-b6
3,561 | 0 | 0


playdrone-apk-09
3,124 | 0 | 0


playdrone-apk-fa
2,864 | 0 | 0


playdrone-apk-7e
2,414 | 1 | 0


playdrone-apk-70
1,809 | 0 | 0


playdrone-apk-04
1,747 | 0 | 0


playdrone-apk-67
1,534 | 0 | 0


playdrone-apk-11
1,435 | 0 | 0

1,402,894

APKs available on Archive.org

1,000
Randomly Sampled APKs downloaded

6.1GB
Of space needed

3:18:37
Time needed to download

Aggregating Permissions

- Requested metadata for each APK from Google
- Counted total permissions for our sample set


```
"permission":[
        "android.permission.INTERNET",
        "android.permission.ACCESS_NETWORK_STATE",
        "android.permission.CHANGE_NETWORK_STATE",
        "android.permission.ACCESS_WIFI_STATE",
        "android.permission.WRITE_EXTERNAL_STORAGE",
        "android.permission.RECEIVE_BOOT_COMPLETED",
        "android.permission.MANAGE_DOCUMENTS",
        "android.permission.GET_ACCOUNTS",
        "android.permission.MANAGE_ACCOUNTS",
        "android.permission.USE_CREDENTIALS",
        "com.google.android.providers.gsf.permission.READ_GSERVICES",
        "com.google.android.googleapps.permission.GOOGLE_AUTH",
        "com.google.android.googleapps.permission.GOOGLE_AUTH.youtube",
        "com.google.android.googleapps.permission.GOOGLE_AUTH.YouTubeUser",
        "com.google.android.c2dm.permission.RECEIVE",
        "android.permission.WAKE_LOCK",
        "android.permission.NFC",
        "android.permission.CAMERA",
        "com.google.android.youtube.permission.C2D_MESSAGE",
        "android.permission.READ_EXTERNAL_STORAGE"
      ],
```

Decompilation Using Androguard

going from apks to source code

Androguard

- Python toolkit
- disassembles and decompiles Android apps
- Converts .apk files back into .java source code

Decompilation Pipeline

Input: .apk files
Output: potential vulnerabilities

loops through every java file and looks for common vulnerabilities or security blunders

995

APKs fully decompiled

12

Vulnerabilities we looked for

6:02:38

To decompile and test for vulnerabilities

Vulnerability Testing

how we tested our apks

Devices Running Vulnerable Versions of Android

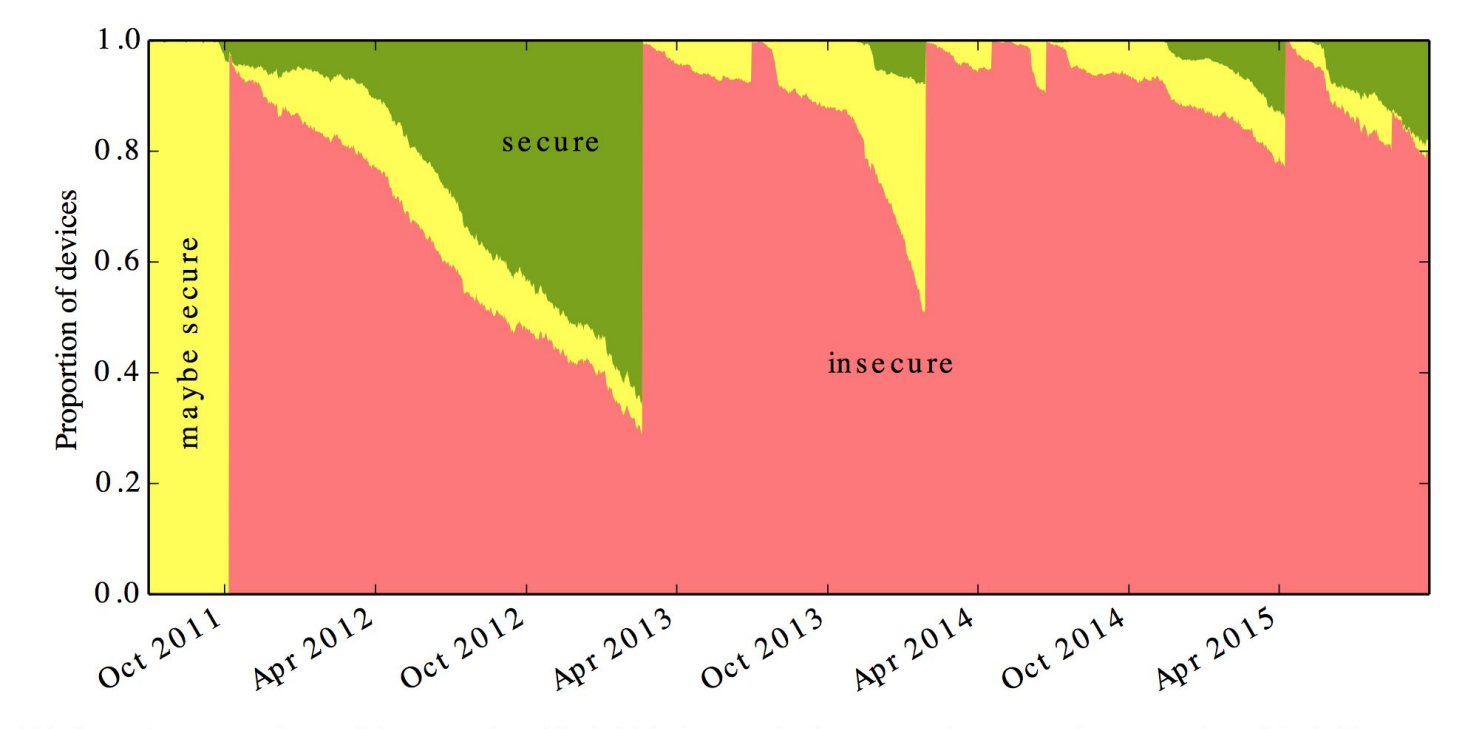


Image from: http://androidvulnerabilities.org

Known Vulnerabilities

CVE-2014-8610
CVE-2014-8507
CVE-2014-7911
CVE-2012-4222
CVE-2011-4276
CVE-2011-3874
CVE-2013-2597
CVE-2014-3100
Google bug 13678484

- Function calls to libraries that have shown vulnerabilities in the past
- Some are permission dependent
- Some of the vulnerabilities depend on the Android version running

API / Security Keys

```
//Fake AWS keys
String accesskey = "AKIAAD9046B4XXJXLX6Q";
String secretkey = "qe9/mgOzc-5uN1oNFfnk++2xcOleLllFGkVj56Nk";
//Fake Twilio keys
String twiliokey = "SK2a0747eba6abf96b7e3c3ff0b4530f6e";
//Fake Plivio keys
String authID    = "WOMDA30G44GNZPZLFKMD";
String authToken = "PBM4ATZlZ88jZ2lj3jLjNdQypGEfZjAyAzM4Pk34";
//Fake Twitter key
//
//Fake FB key
String appSecret2= "e4cf63df1f2e1c2u27e3j8dda70j05cl";
String appSecret = "f4fc54590da4c6ct1a6gc6d1f42134nk";
String appID     = "629903034072938";
String appID2    = "1340434267886851";

//Paypal
String apiUsername = "schandrasekaras-us_api1.paypal.com";
String apiPassword = "4AK8KBHSKUQE6JK8";
String apiPassword2= "R2CV5XCLEYEMQ7RH";
String signature   = "ACxwNDtvkO2nsXs2fa4LdAjRq98TA204gsd2ODrvR4mTZY.e.nYU15SH";
String certificate = "6CC5F19AB23DB0C45026DFA45D21057D";
String certificate2= "DA9610E00EA1B8F1AD7711FCEDCE1971";
```

API / Security Keys Continued...

- Tricky problem of detecting randomness in a string

- Not all keys are vulnerabilities
- 98.7% Accuracy on the 1000 apps

HTTP instead of HTTPS

- Many developers use HTTP instead of HTTPS
- Easy to eavesdrop on HTTP connections, some of which are transmitting potentially sensitive information
- Even experienced developers forget or neglect to use HTTPS

Just a few of the urls we found:

“public static final String ACCESS_URL = "http://api.twitter.com/oauth/access_token";"
“public static final String AUTHORIZE_URL = "http://api.twitter.com/oauth/authorize";"
“this(1, "http://filmgrail.com:8001/api/account/login", p5, p6);"

Comments Not Removed

- Comments can reveal too much about code
- Allow an attacker to reverse engineer an app more easily

Some real comments we found:

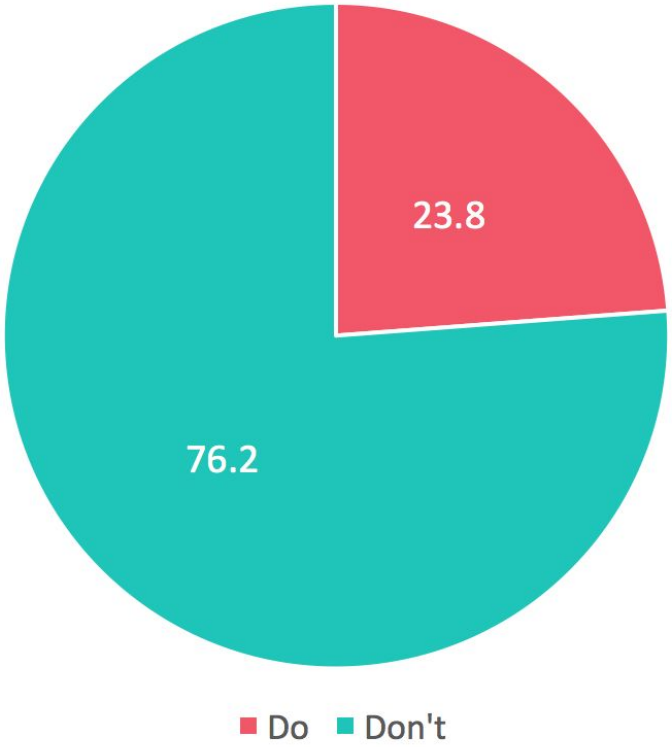
```
// if (!this.isServiceRunning) {
```

```
// if (!v0.containsKey(p9)) {
```

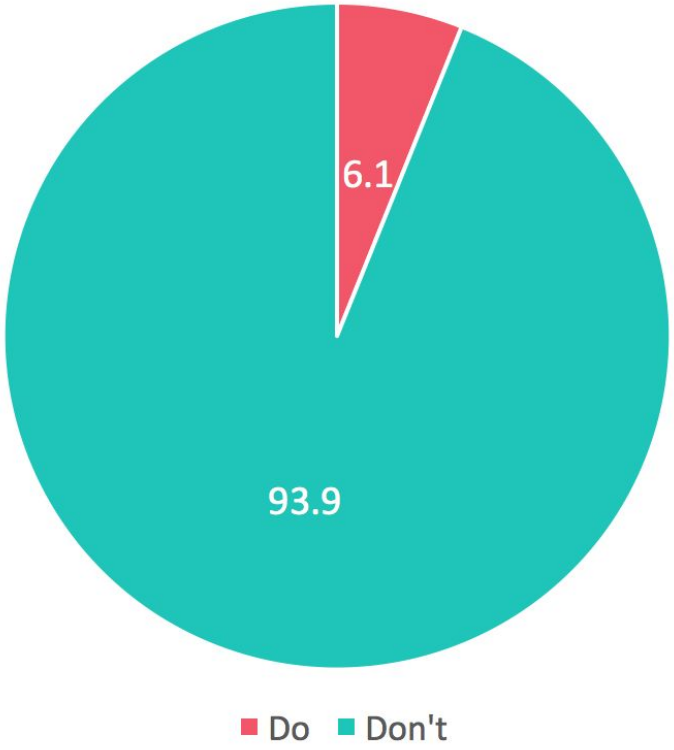
Results

some surprising numbers

ACCESS_FINE_LOCATION

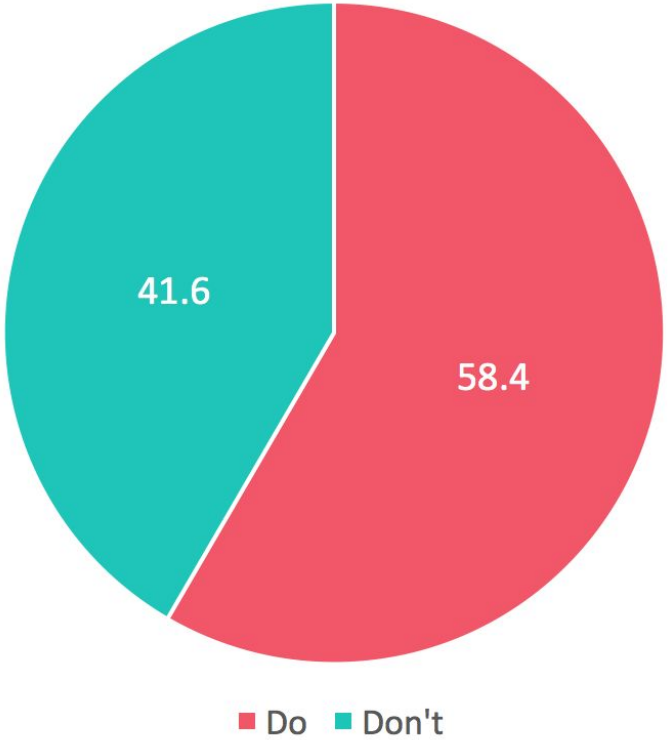


READ_CONTACTS



READ_EXTERNAL_STORAGE

Note: not required for application specific directories



11,799

Potentially Vulnerable Lines of Code

514

Potentially Vulnerable Apps

HTTP instead of HTTPS

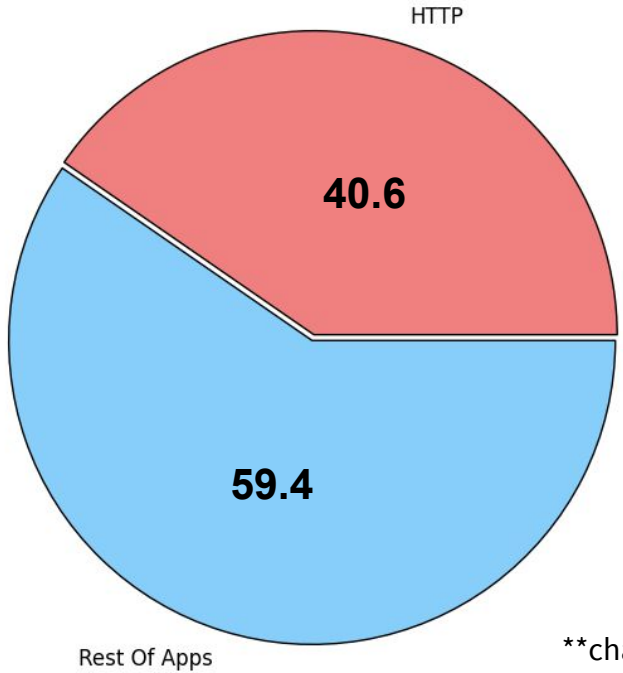


**chart generated by our pipeline

CVE-2014-7911

Use of ObjectInputStream java library that before Android 5.0.0 would allow an attacker to inject and execute arbitrary code

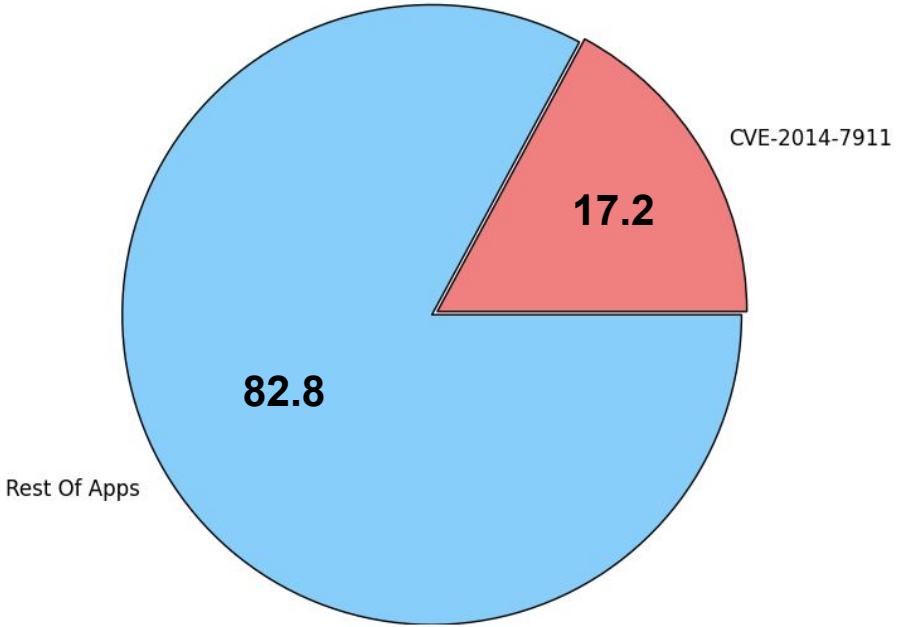


**chart generated by our pipeline

API / Security Keys

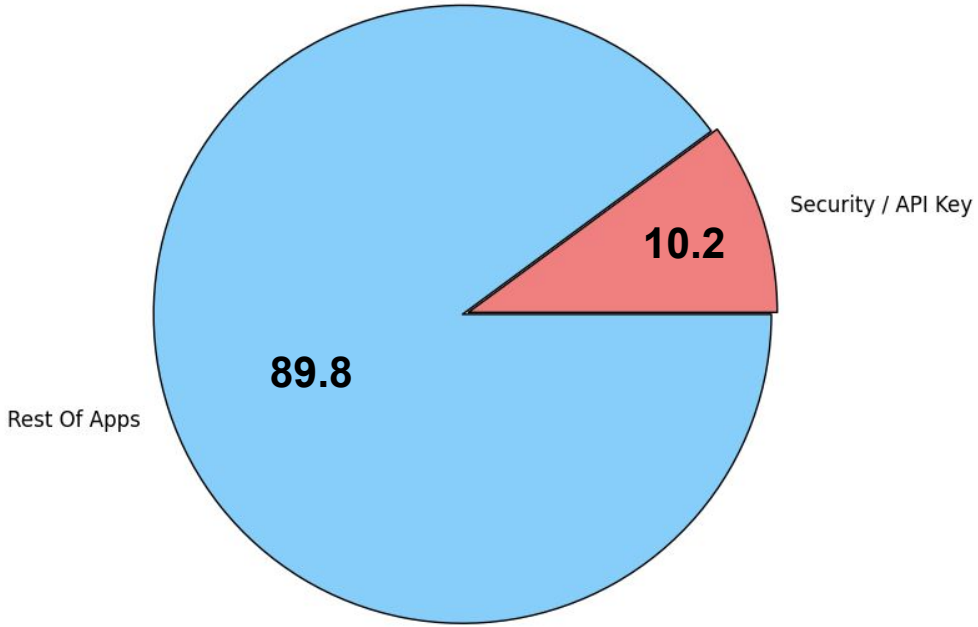


**chart generated by our pipeline

API / Security Keys

We found **valid** AWS keys...

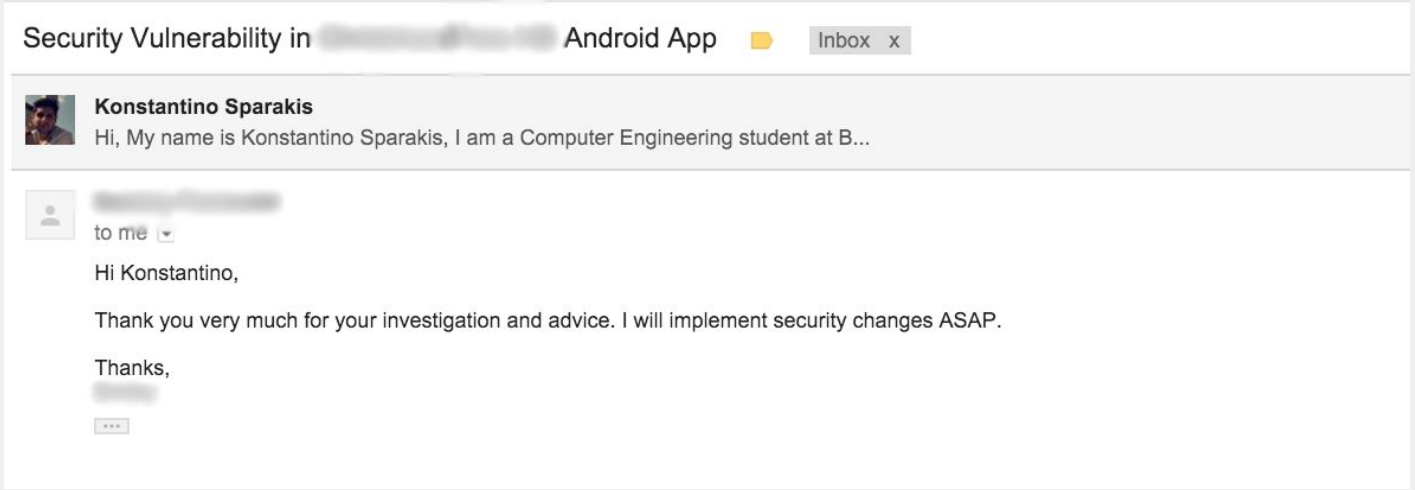
```
this.s3Client = new com.amazonaws.services.s3.AmazonS3Client(
    new com.amazonaws.auth.BasicAWSCredentials("AKIAJY2T7FN7R4P447VA",
                                               "NGTkk0ymhaE+vLL6lHkg9+Im5oNgTq3280unYGFM"));
```

```
wireless1x-                    $ AWSSECRET=NGTkp0
wireless1x-                    $ AWSKEY=AKIA
wireless1x-                    $ aws s3 ls
2013-12-21 07:56:04            hd
2014-11-23 01:46:58            photos
```

along with Flurry analytics keys, Facebook Keys,
Google Keys, passwords… and more

Ethical Hacking

… so we emailed the developer.


Security Vulnerability in [redacted] Android App Inbox x

Konstantino Sparakis
Hi, My name is Konstantino Sparakis, I am a Computer Engineering student at B...

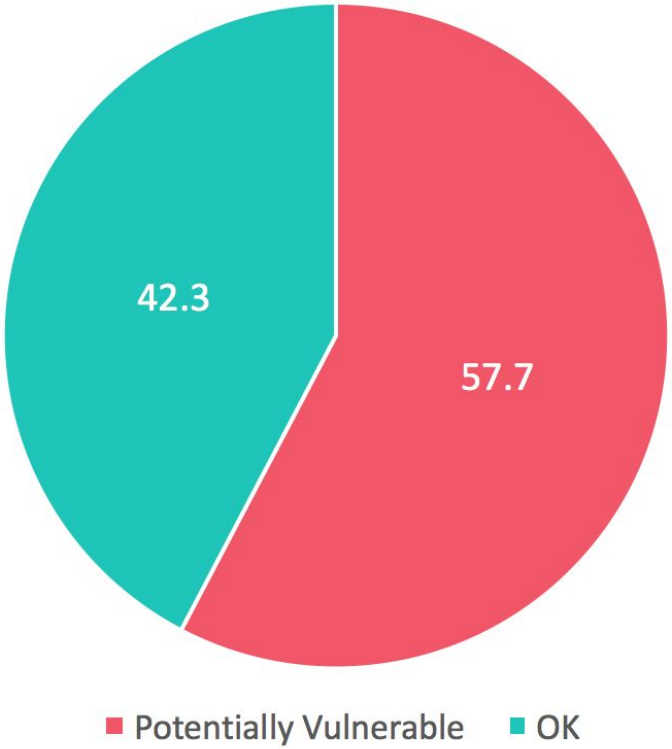
[redacted]
to me

Hi Konstantino,

Thank you very much for your investigation and advice. I will implement security changes ASAP.

Thanks,
[redacted]

Popular Apps (greater than 50,000+ downloads)



Future Work

Run through more APKs
(currently testing 10,000)

Support multithreading
to perform downloads and app analysis simultaneously

Get more forks on GitHub!
we've been forked 3 times
we've been cloned 17 times

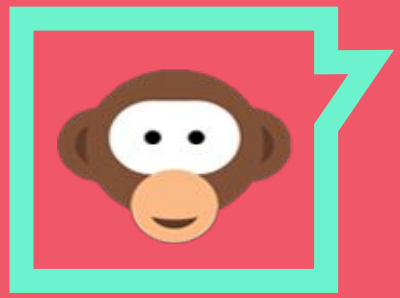
Test for more vulnerabilities

Improve static analysis scripts

Improve charting utility

Test 100 most popular apps

Automated email to developers



Thanks!

Any questions?

Don’t make us go apekit.