Computer Science
NC STATE UNIVERSITY

Detecting Passive Content Leaks and Pollution in Android Applications

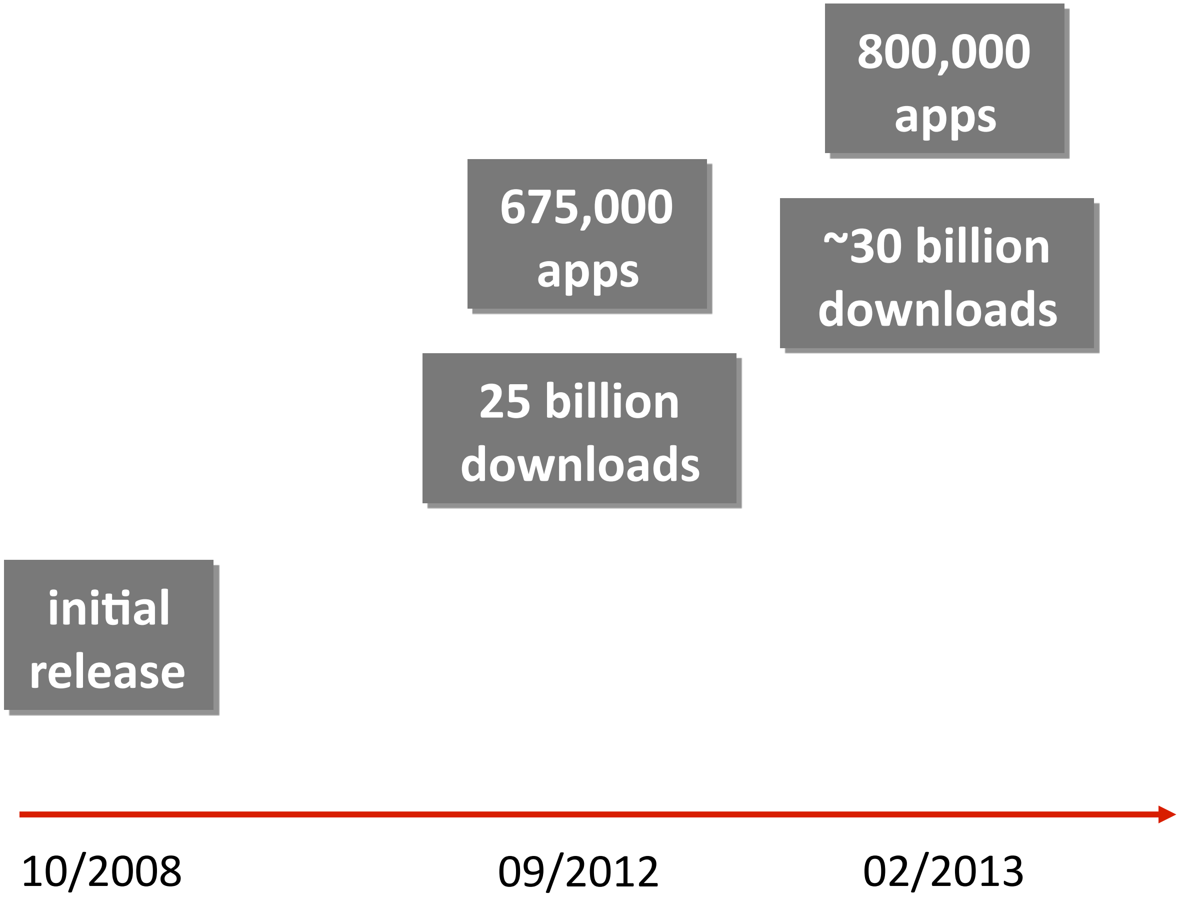
Yajin Zhou and Xuxian Jiang

North Carolina State University

NDSS
SYMPOSIUM

Apps Are Becoming Popular

Google play



initial release

675,000 apps

25 billion downloads

800,000 apps

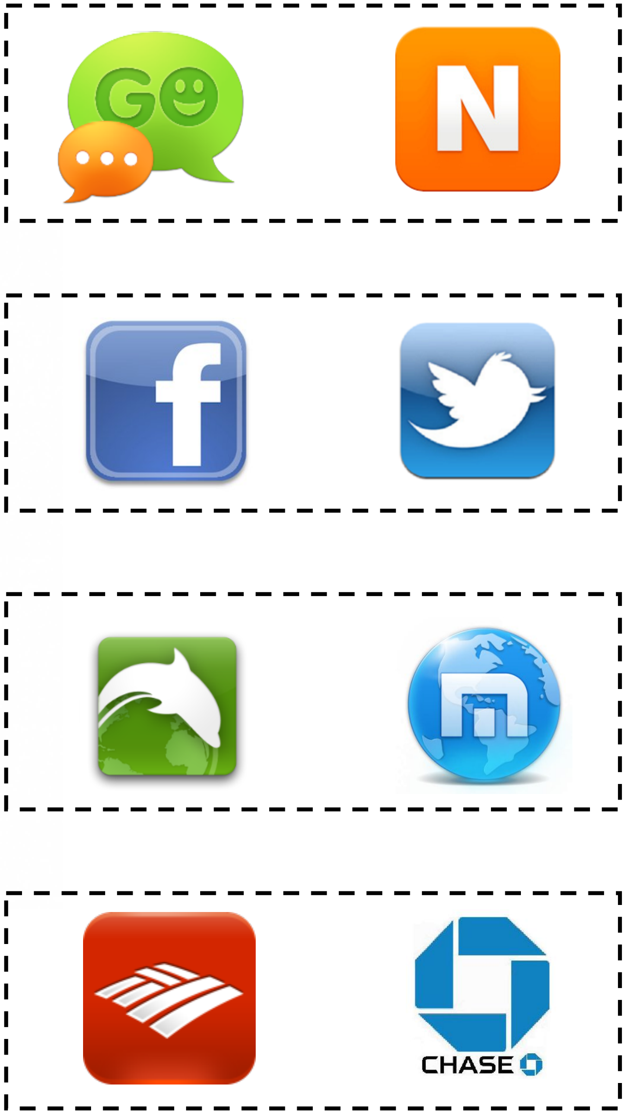
~30 billion downloads

10/2008

09/2012

02/2013

Apps Are Managing User Data



Messages

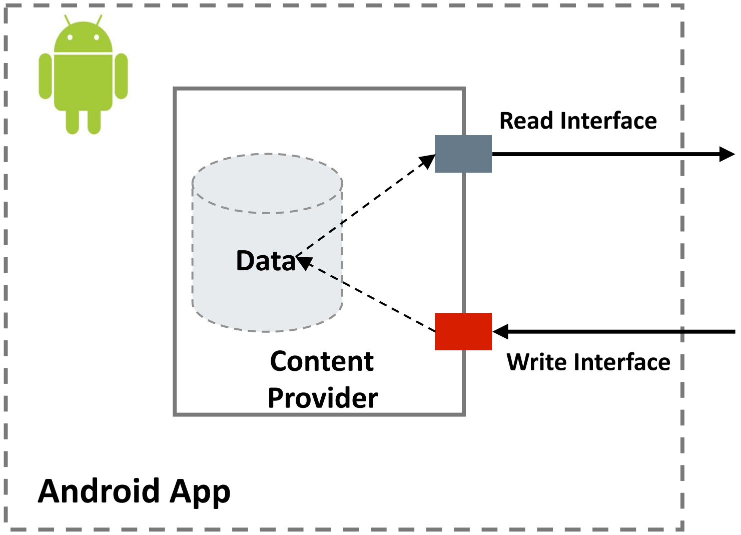
Friends

Browser Histories

Bank Accounts

Content Providers

- ❑ Manage access to a structured set of data

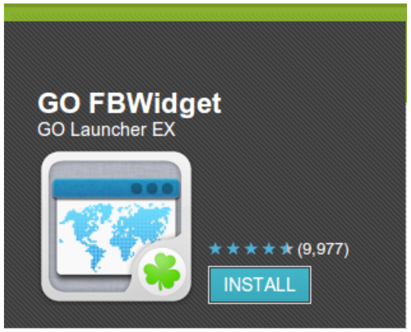


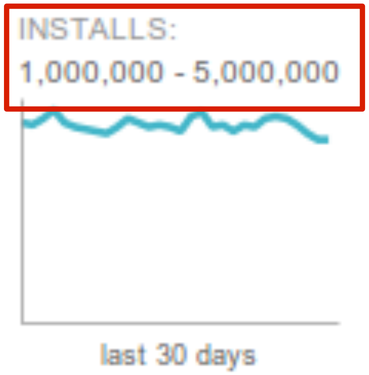
- ❑ ***By default*** *are* ***open*** to ***all*** apps on the phone
  (before Android 4.2)

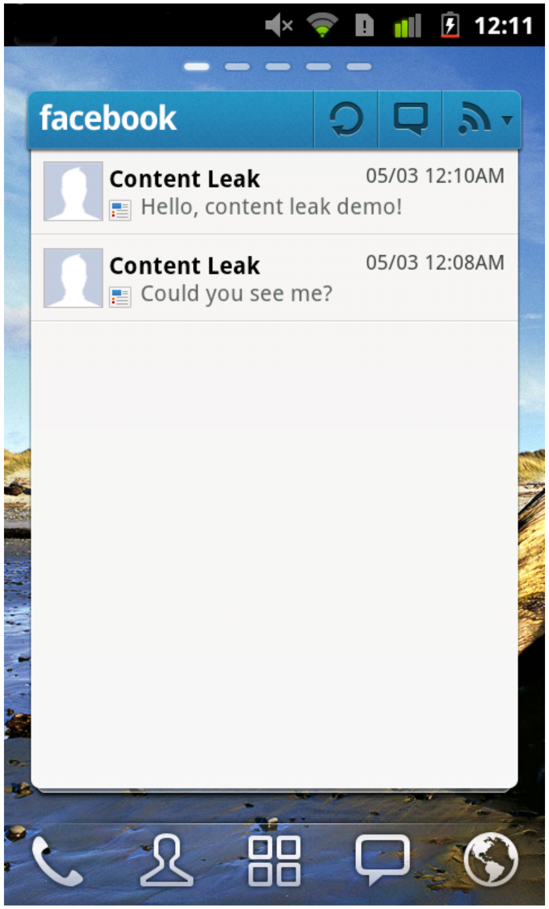
Any potential security risks?

A Motivating Example

- GO FBWidget: popular Android app with more than 1 million installs



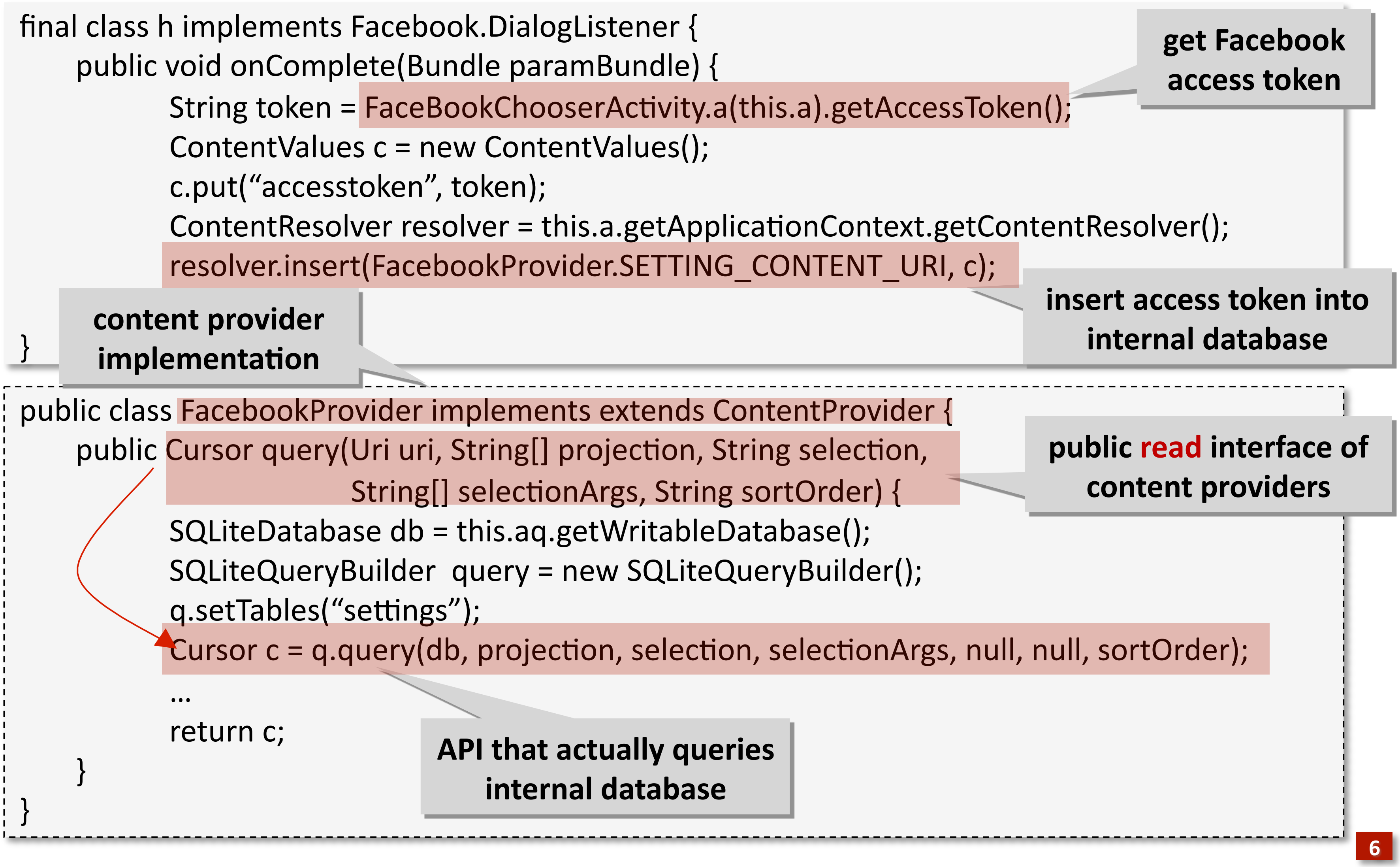




Content Leak 05/03 12:10AM
Hello, content leak demo!

Content Leak 05/03 12:08AM
Could you see me?

A Motivating Example



```
final class h implements Facebook.DialogListener {
    public void onComplete(Bundle paramBundle) {
            String token = FaceBookChooserActivity.a(this.a).getAccessToken();
            ContentValues c = new ContentValues();
            c.put("accesstoken", token);
            ContentResolver resolver = this.a.getApplicationContext.getContentResolver();
            resolver.insert(FacebookProvider.SETTING_CONTENT_URI, c);
}
```

get Facebook access token

insert access token into internal database

content provider implementation

```
public class FacebookProvider implements extends ContentProvider {
    public Cursor query(Uri uri, String[] projection, String selection,
                        String[] selectionArgs, String sortOrder) {
            SQLiteDatabase db = this.aq.getWritableDatabase();
            SQLiteQueryBuilder  query = new SQLiteQueryBuilder();
            q.setTables("settings");
            Cursor c = q.query(db, projection, selection, selectionArgs, null, null, sortOrder);
            …
            return c;
    }
}
```

public **read** interface of content providers

API that actually queries internal database

A Motivating Example

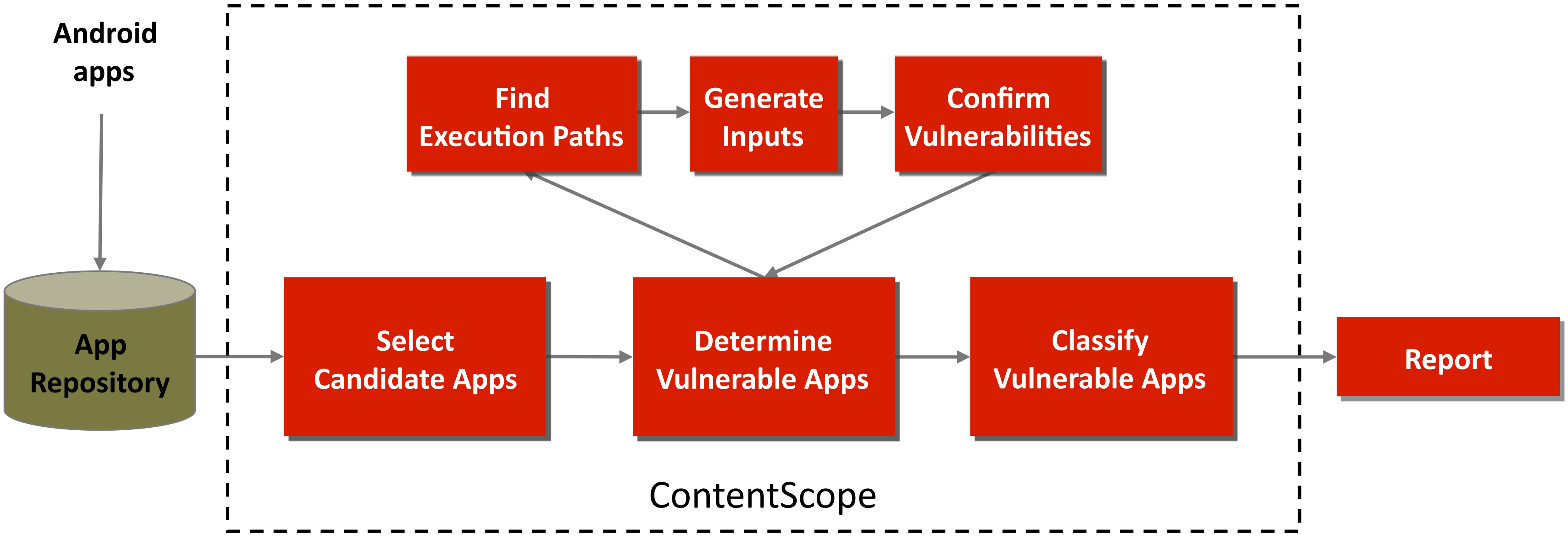
- ❑ Can be exploited to leak private data
    - ❑ *Access token,* Facebook posts

Automatically log into user's Facebook account and make posts

Our Work

- ❑ Systematically study two vulnerabilities: content leaks and content pollution
  - ❑ 2.0% and 1.4% of apps are susceptible, respectively
  - ❑ Types of information leaked
    - ❑ SMS messages, contacts, user credentials, …
  - ❑ Possible side-effects
    - ❑ Block SMS messages and phone calls
    - ❑ Download apps and prompt for installation

System Design

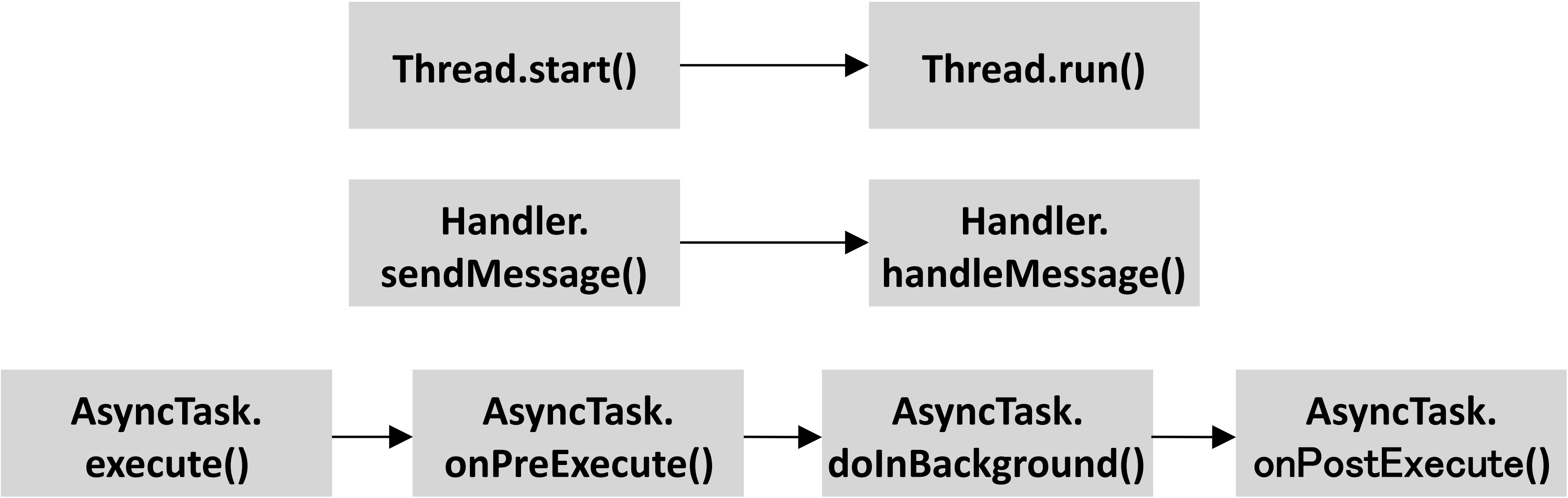


Find Execution Paths

- From public interfaces of content providers to functions that actually operate on internal database

Find Execution Paths

- Function call graph
  - Object reference resolution
  - Call graph discontinuity



Thread.start() → Thread.run()

Handler. sendMessage() → Handler. handleMessage()

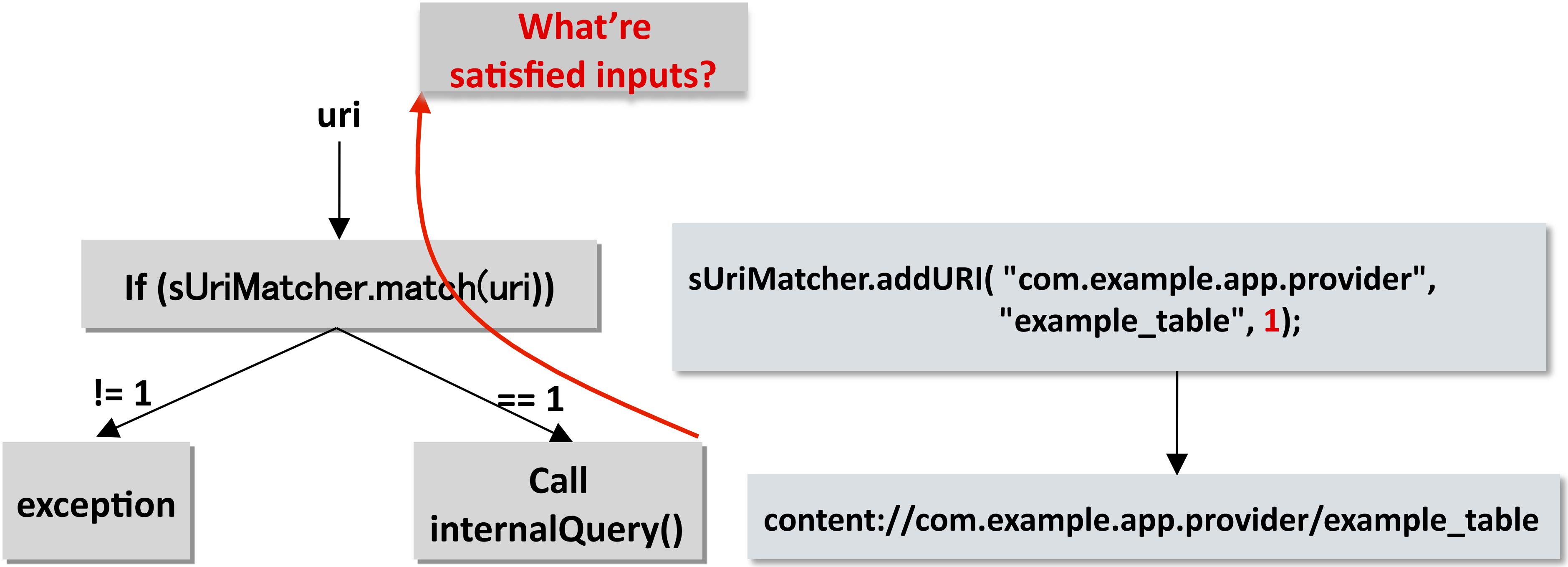
AsyncTask. execute() → AsyncTask. onPreExecute() → AsyncTask. doInBackground() → AsyncTask. onPostExecute()

Generate Inputs

- ❑ Generate control flow graph
- ❑ Obtain constraints
- ❑ Resolve constraints

Generate Inputs

- Android specific APIs
  - UriMatcher



What're satisfied inputs?

uri

If (sUriMatcher.match(uri))

!= 1

== 1

exception

Call internalQuery()

sUriMatcher.addURI("com.example.app.provider",
"example_table", 1);

content://com.example.app.provider/example_table

Confirm Vulnerabilities

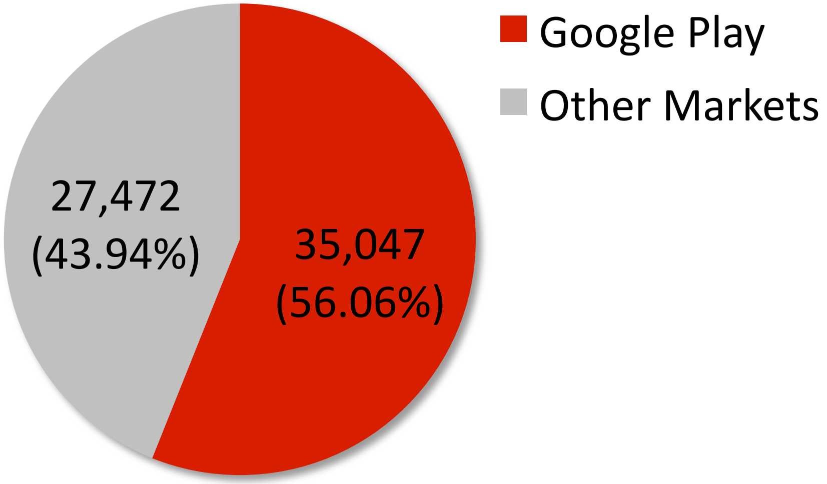
- Feed generated inputs into a test app
- Invoke public interfaces of content providers
  - query(), insert(), …
- Determine the existence of vulnerabilities based on return value
  - query(): Cursor object
  - insert(): URI object

System Implementation

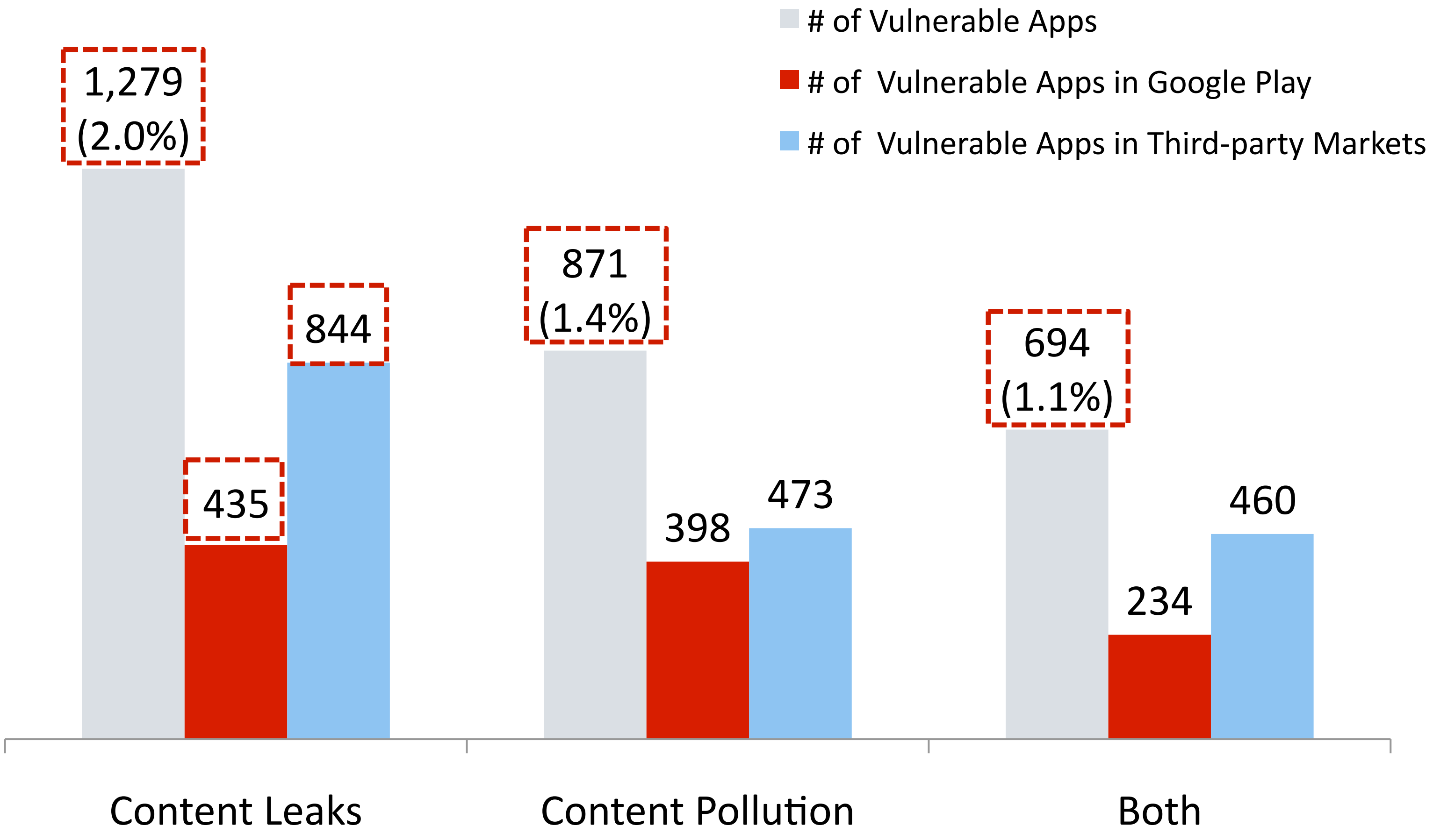
- Around 6,500 SLOCs
- Public interfaces of content providers
  - query(), openFile()
  - insert(), update()
- APIs that actually read or write internal database
  - SQLiteDatabase.query(), SQLiteDatabase.insert(), SQLiteQueryBuilder.query(), …

Evaluation

- Dataset: 62,519 free apps
  - Sources: Google Play and ten other Android markets
  - Time: February 2012



Overall Results



Main Types of Leaked Data

Category	# of apps	Representative App	# of Installs
SMS messages	268	Pansi SMS	500,000 – 1,000,000
Contacts	128	mOffice – Outlook sync	100,000 – 500,000
Private information in IM Apps	121	Messenger With You	10,000,000 – 50,000,000
User credentials	80	GO FB Widget	1,000,000 – 5,000,000
Browser History	70	Dolphin Browser HD	10,000,000 – 50,000,000
Call logs	61	Droid Call Filter	100,000 – 500,000
Private information In social network apps	27	Sina Weibo	100,000 – 500,000

Side-effects of Content Pollution

- Block SMS messages and phone calls: by manipulating security settings
  - DW Contacts
- Download apps and prompt for installation
  - Baidu Appsearch, Qihoo Browser

Vulnerable Security Apps

- Mobile Security Personal Ed.
  - Leak browser histories
- QQPimSecure, Anguanjia
  - Leak SMS, phone call logs
  - Block SMS and phone calls

Possible Mitigations

- App Developers
  - Patch their vulnerable apps
- Platform provider (Google)
  - Change the default setting of content provider interface

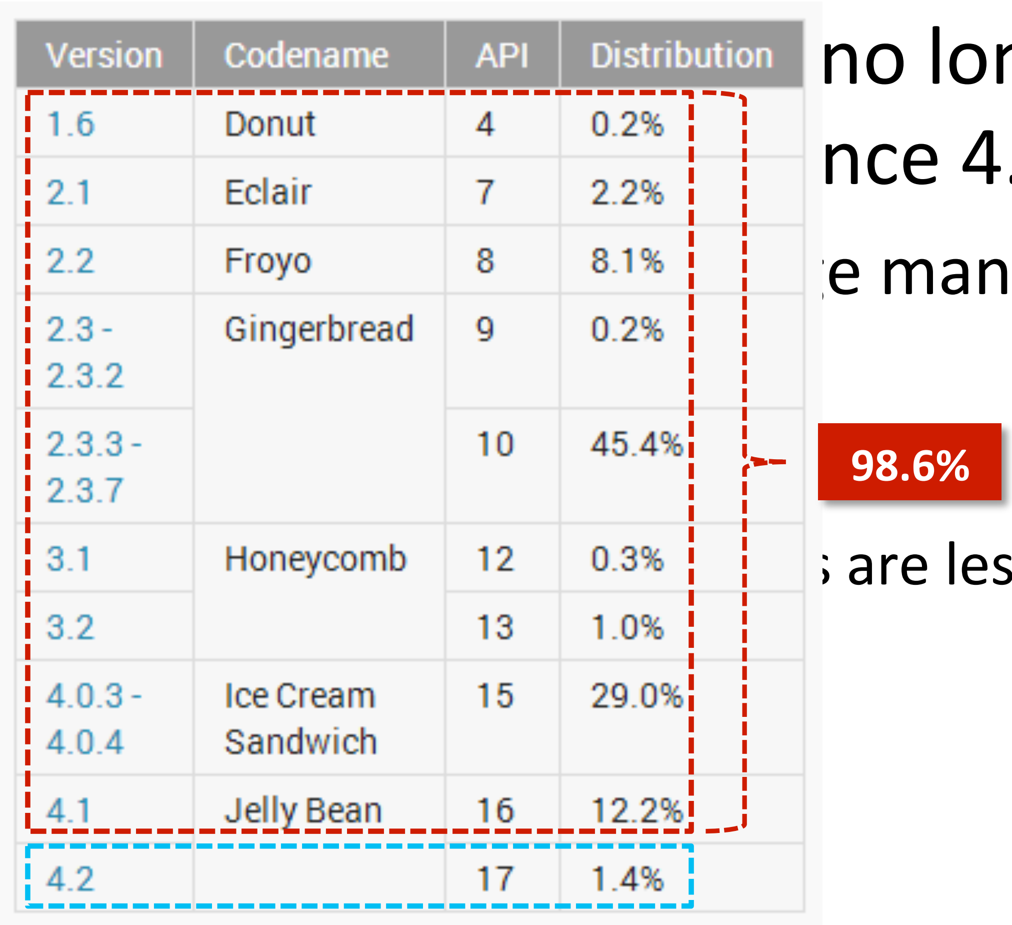
Possible Mitigations

- By Google: content providers are no longer exported by default on Android since 4.2
  - Developers need to **explicitly** change manifest file
    - Set targetSdkVersion to 17 (or higher)
  - Problems remain on old Android versions
    - The API level of ***98.6%*** Android devices are less than 17 on February 04, 2013 [1]

[1] http://developer.android.com/about/dashboards/index.html

Possible Mitigations

- By Google no longer exported nce 4.2
  - Develop e manifest file
    - Set tar
  - Problem
    - The AF ; are less than 17 on Feb



Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	2.2%
2.2	Froyo	8	8.1%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	45.4%
3.1	Honeycomb	12	0.3%
3.2		13	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.0%
4.1	Jelly Bean	16	12.2%
4.2		17	1.4%

98.6%

[1] http://developer.android.com/about/dashboards/index.html

Related Work

- Smartphone privacy
  - TaintDroid [Enck *et al.*, OSDI 10], AdRisk [Grace *et al.*, ACM WiSec 12] ...
- Confused deputy
  - Woodpecker [Grace *et al.*, NDSS 12], Permission Re-Delegation [Felt *et al.*, USENIX Security 11] ...
- Vulnerability detection
  - BitBlaze [Song *et al.*, ICISS 08], KLEE [Cadar *et al.*, USENIX Security 08] ...

Conclusion

- Systematically study two vulnerabilities: content leaks and content pollution
  - 2.0% and 1.4% of apps are susceptible, respectively
  - Types of information leaked
    - SMS messages, contacts, user credentials, …
  - Possible side-effects:
    - Block SMS messages and phone calls, …

Q&A

Yajin Zhou
http://yajin.org
(yajin_zhou@ncsu.edu)