Finding Clues For Your Secrets: Semantics-Driven, Learning-Based Privacy Discovery in Mobile Apps

Yuhong Nan, Zhemin Yang, Yuan Zhang, Donglai Zhu and Min Yang

Fudan University

Xiaofeng Wang

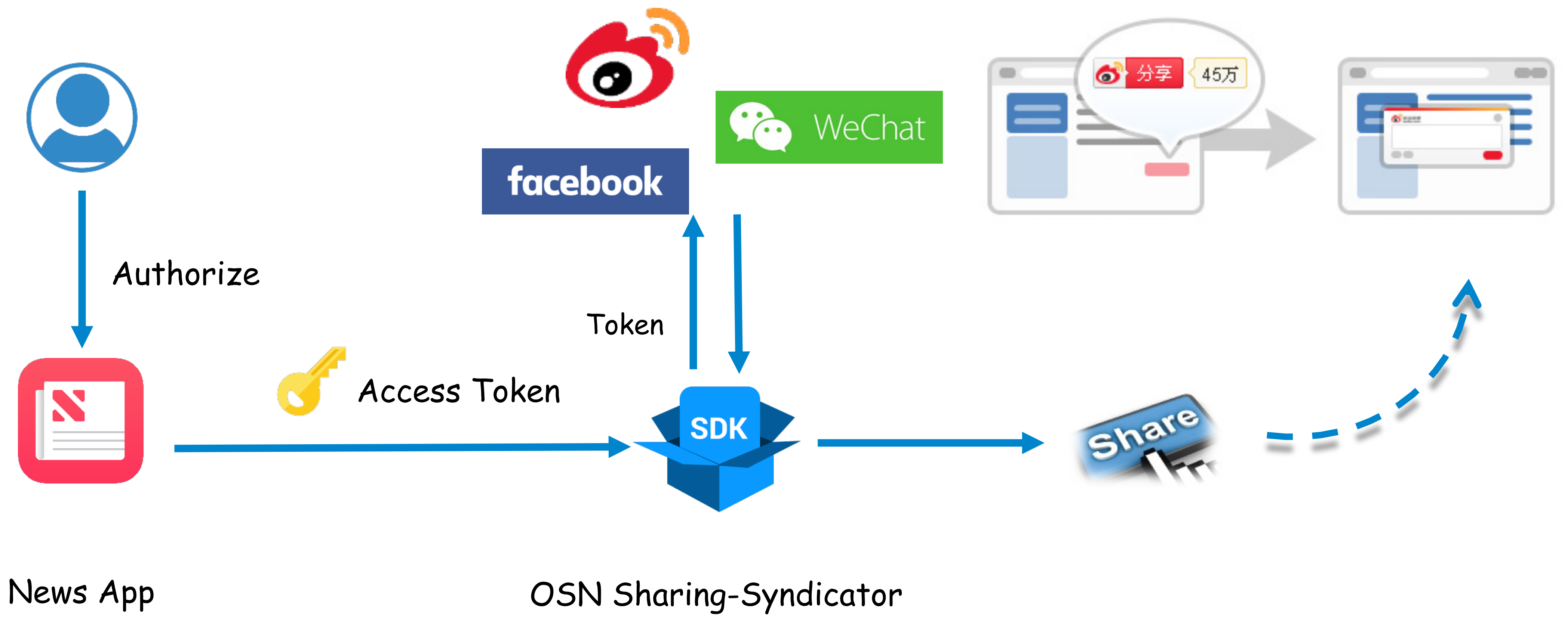
Indiana University Bloomington

Today's Mobile Apps

- Multiple web services integration
  - Ad services, social SDKs, development tools, etc.

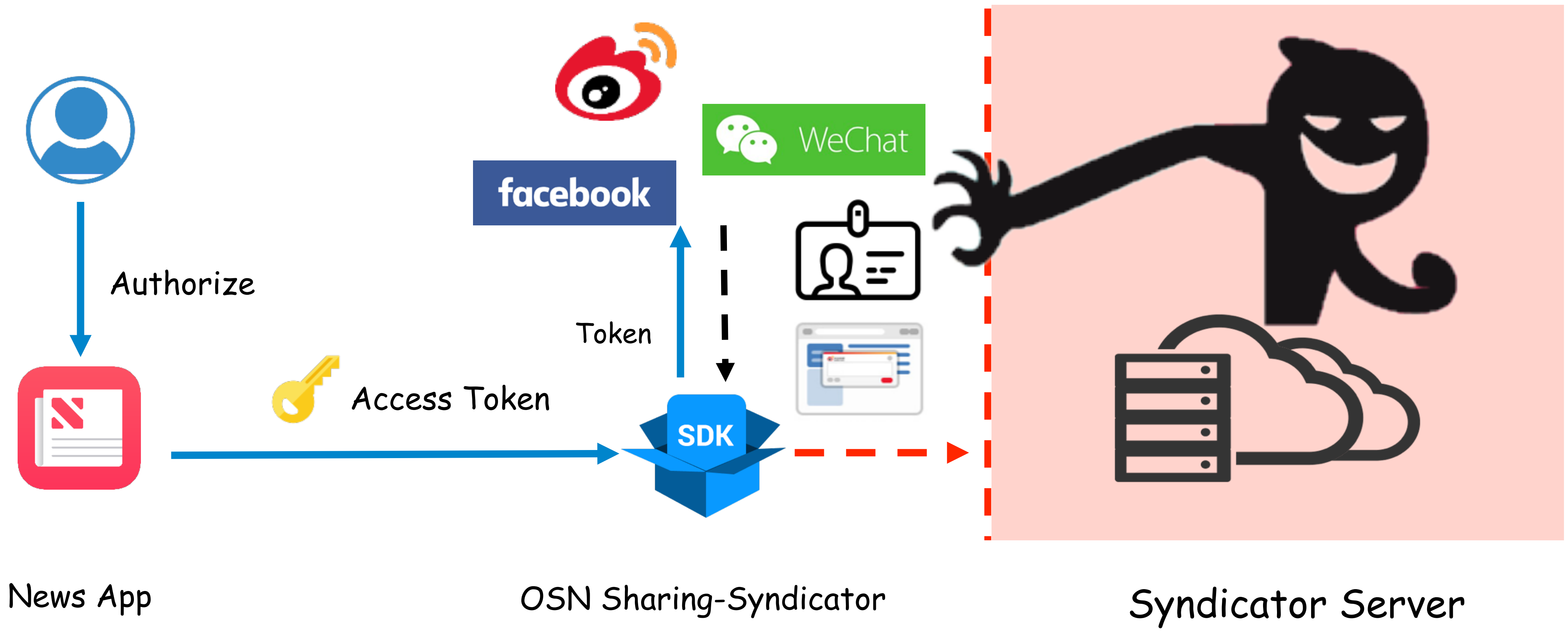
- Privacy implications
  - Ability to collect user data. E.g., Pluto [NDSS'16]
  - Ability to associate user activities, infer user secrets. E.g., Linkdroid [Security'15]

Motivation



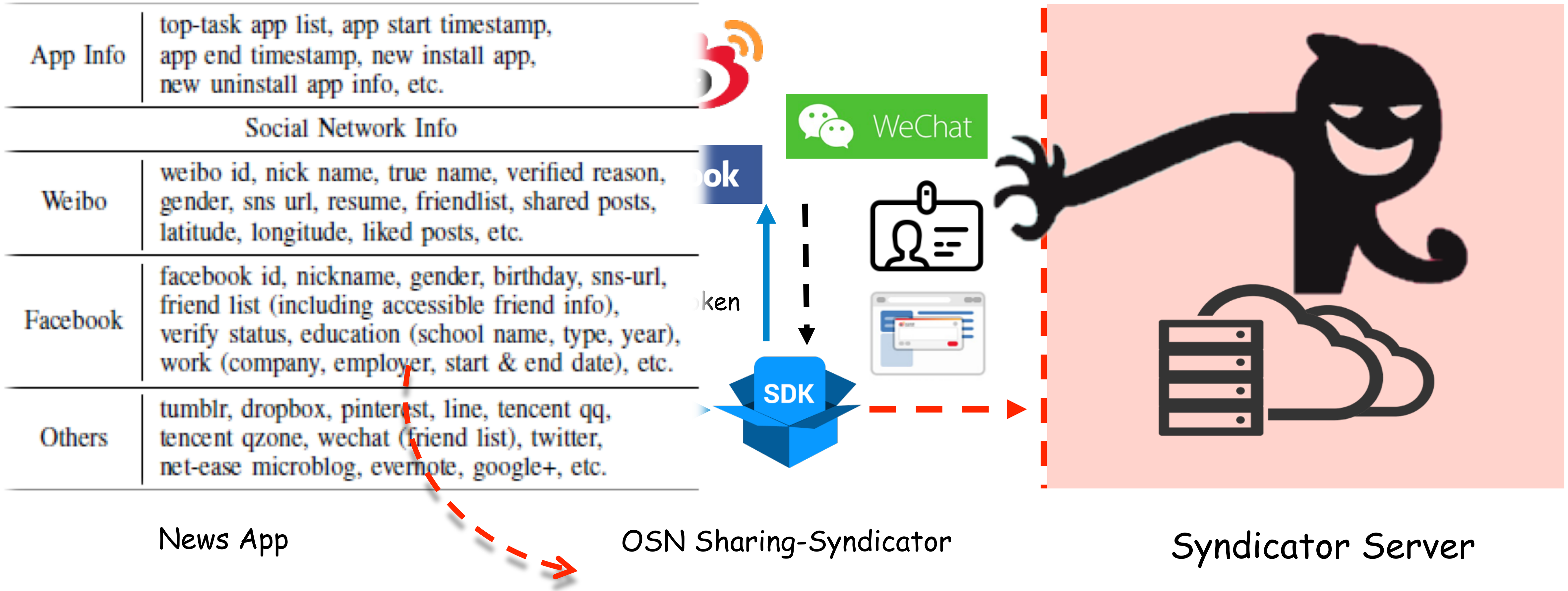
An OSN sharing syndicator (SDK) for post-sharing

Motivation



Collecting user's detailed profile and shared content…

Motivation



App Info	top-task app list, app start timestamp, app end timestamp, new install app, new uninstall app info, etc.
Social Network Info	
Weibo	weibo id, nick name, true name, verified reason, gender, sns url, resume, friendlist, shared posts, latitude, longitude, liked posts, etc.
Facebook	facebook id, nickname, gender, birthday, sns-url, friend list (including accessible friend info), verify status, education (school name, type, year), work (company, employer, start & end date), etc.
Others	tumblr, dropbox, pinterest, line, tencent qq, tencent qzone, wechat (friend list), twitter, net-ease microblog, evernote, google+, etc.

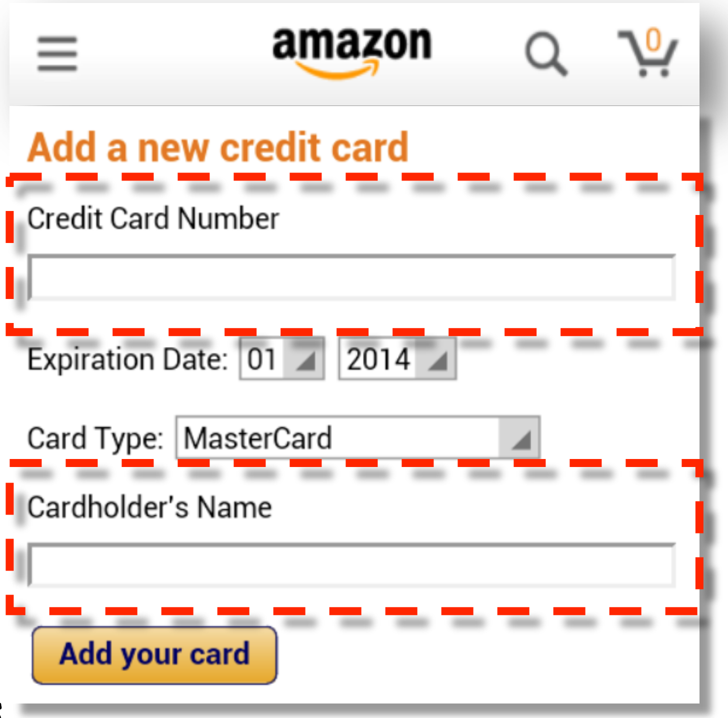
Preliminary question: **sensitive data identification**

Automated leakage analysis

- System-controlled privacy
  - Fixed list of APIs

➢ Location	✓ LocationManager. getLastKnownLocation()
➢ Contact	✓ ContentResolver.query(CONTACT_URI)
➢ SMS	✓ SmsMessage. getMessageBody ()
➢ Phone Number	✓ TelephonyManager. getLine1Number()
➢ …	✓ …

- User-input privacy (UIP data)
  - UI-based identification.
    - SUPOR, UIPicker [USENIX Security'15]



Add a new credit card

Credit Card Number

Expiration Date: 01 2014

Card Type: MasterCard

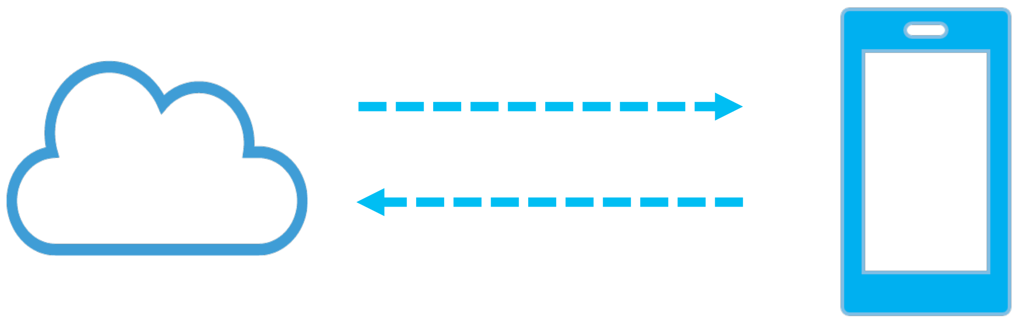
Cardholder's Name

Add your card

- UIPicker: User-Input Privacy Identification in Mobile Applications
- SUPOR: Precise and Scalable Sensitive User Input Detection for Android Apps

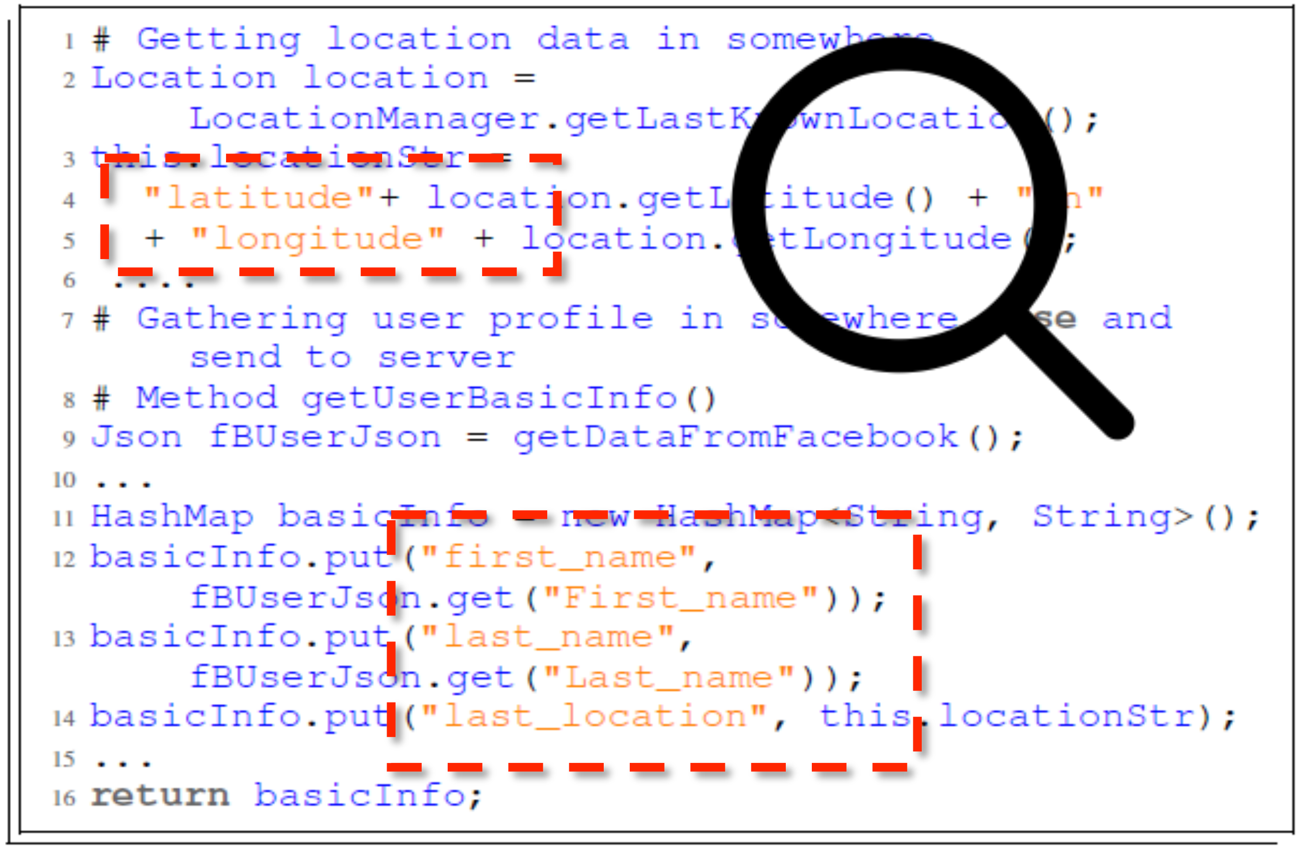
Automated leakage analysis

- Server-side sensitive data
  - UI or System API-based Labelling?
    - Go through system API without specific characters
  - Network Communication?
    - Difficult to capture network traffic at a large scale with runtime analysis
      - E.g., a valid login for each app



Observation

- Finding clues from app code
  - Preserved semantics



```
# Getting location data in somewhere
Location location =
     LocationManager.getLastKnownLocation();
this.locationStr =
  "latitude"+ location.getLatitude() + "\n"
  + "longitude" + location.getLongitude();
 ....
# Gathering user profile in somewhere else and
     send to server
# Method getUserBasicInfo()
Json fBUserJson = getDataFromFacebook();
...
HashMap basicInfo = new HashMap<String, String>();
basicInfo.put("first_name",
     fBUserJson.get("First_name"));
basicInfo.put("last_name",
     fBUserJson.get("Last_name"));
basicInfo.put("last_location", this.locationStr);
...
return basicInfo;
```

Our Work

- ClueFinder
  - New technique for **sensitive data source discovery** from app code
    - ✓ System APIs
    - ✓ User interfaces
    - ✓ Server-side sensitive data

- Large-scale exposure risk analysis for third-party libraries in Android apps
  - 445,688 apps from multiple app stores
  - New findings

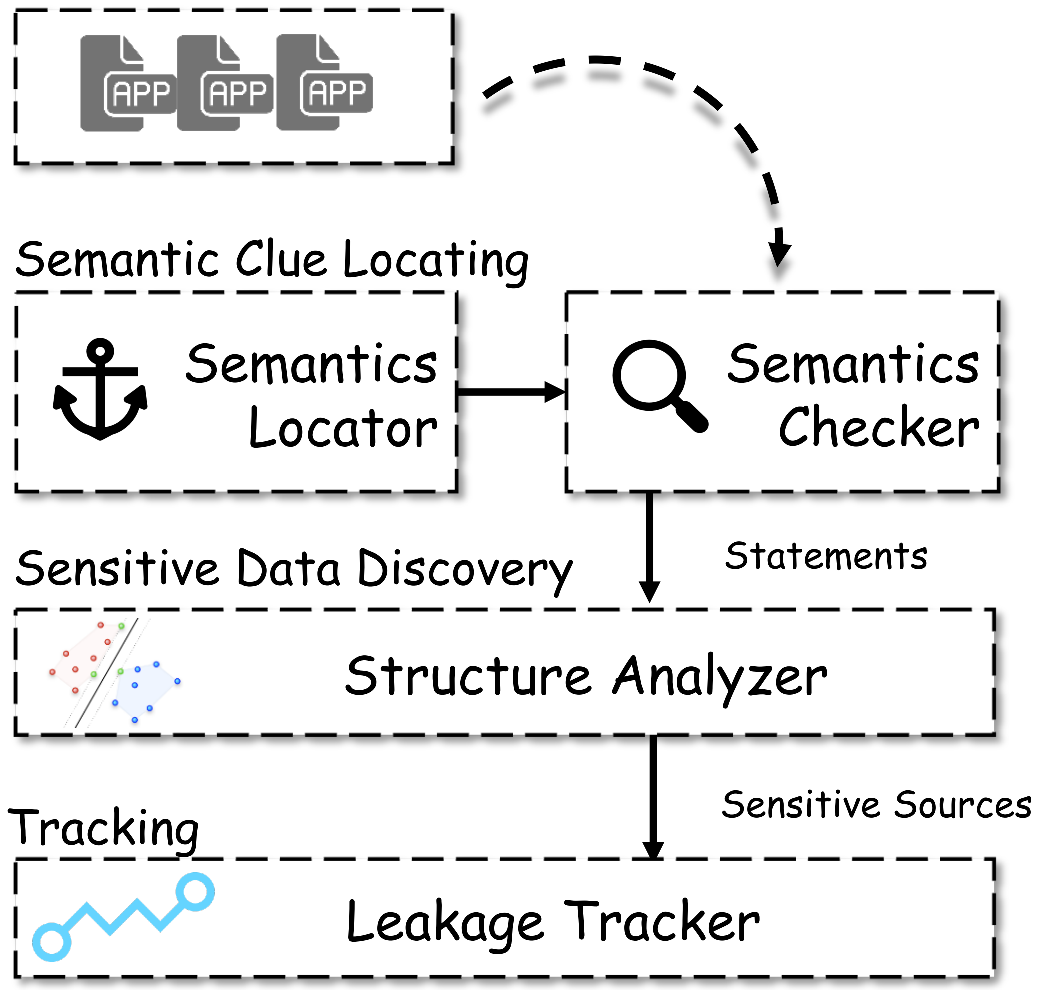
Technical Challenges

- Ambiguity of text strings in app code
  - com/tencent/padqq/activity/Add**FriendList**Activity
  - **UserProfile**Uri
  - is_**mobile_phone**_valid

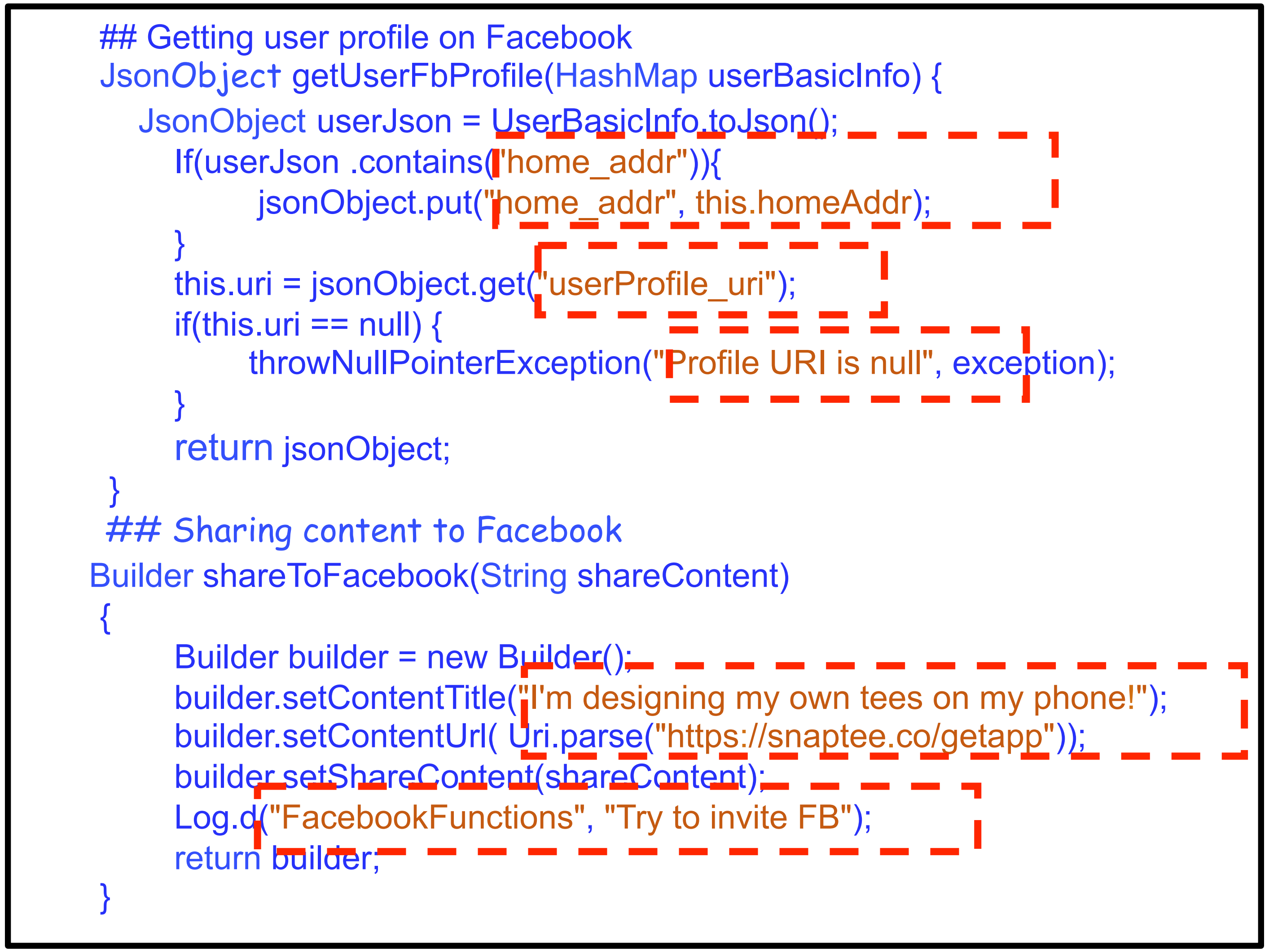
- Privacy-related strings != Sensitive Data
  - Log.e ("**Username** is null, check valid user input..")
  - XXX.setContentTitle(" Your current **Location:** ");

ClueFinder Design

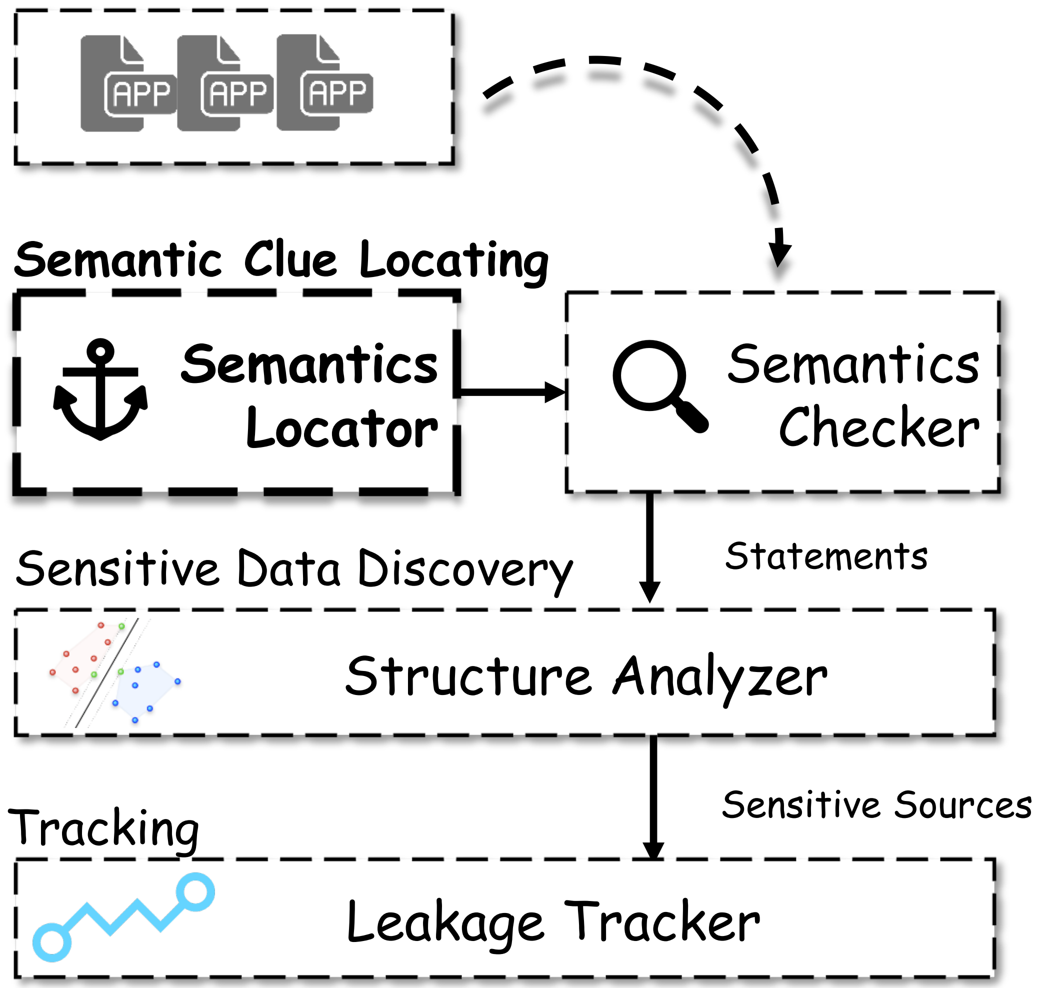
```
## Getting user profile on Facebook
JsonObject getUserFbProfile(HashMap userBasicInfo) {
    JsonObject userJson = UserBasicInfo.toJson();
        If(userJson .contains("home_addr")){
                jsonObject.put("home_addr", this.homeAddr);
        }
        this.uri = jsonObject.get("userProfile_uri");
        if(this.uri == null) {
                throwNullPointerException("Profile URI is null", exception);
        }
        return jsonObject;
 }
 ## Sharing content to Facebook
Builder shareToFacebook(String shareContent)
{
        Builder builder = new Builder();
        builder.setContentTitle("I'm designing my own tees on my phone!");
        builder.setContentUrl( Uri.parse("https://snaptee.co/getapp"));
        builder.setShareContent(shareContent);
        Log.d("FacebookFunctions", "Try to invite FB");
        return builder;
}
```



ClueFinder Design



```
## Getting user profile on Facebook
JsonObject getUserFbProfile(HashMap userBasicInfo) {
   JsonObject userJson = UserBasicInfo.toJson();
      If(userJson .contains("home_addr")){
            jsonObject.put("home_addr", this.homeAddr);
      }
      this.uri = jsonObject.get("userProfile_uri");
      if(this.uri == null) {
            throwNullPointerException("Profile URI is null", exception);
      }
      return jsonObject;
 }
 ## Sharing content to Facebook
Builder shareToFacebook(String shareContent)
{
      Builder builder = new Builder();
      builder.setContentTitle("I'm designing my own tees on my phone!");
      builder.setContentUrl( Uri.parse("https://snaptee.co/getapp"));
      builder.setShareContent(shareContent);
      Log.d("FacebookFunctions", "Try to invite FB");
      return builder;
}
```

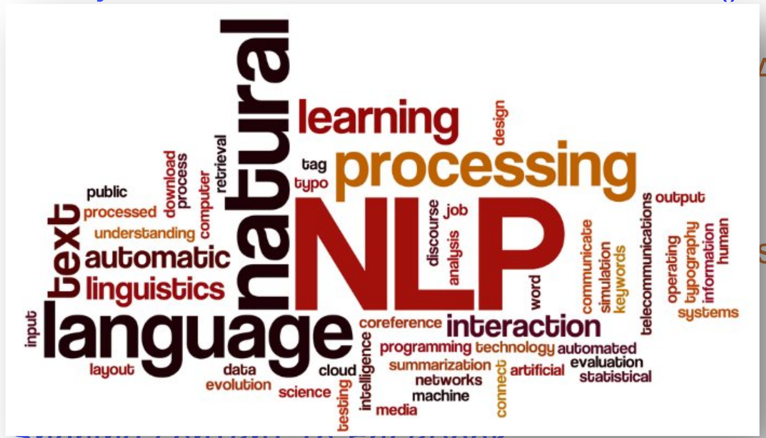


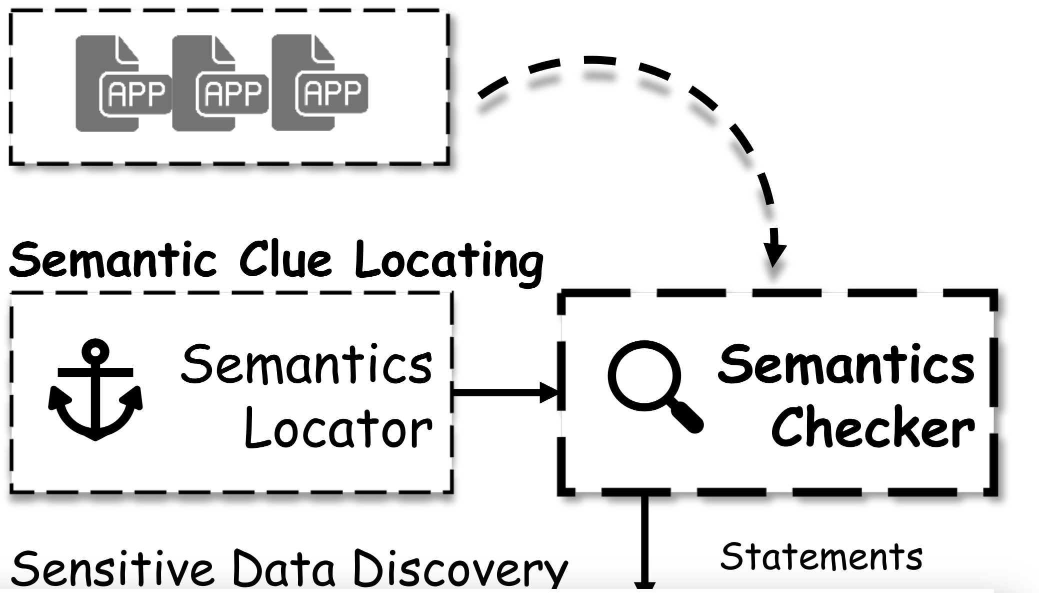
ClueFinder Design

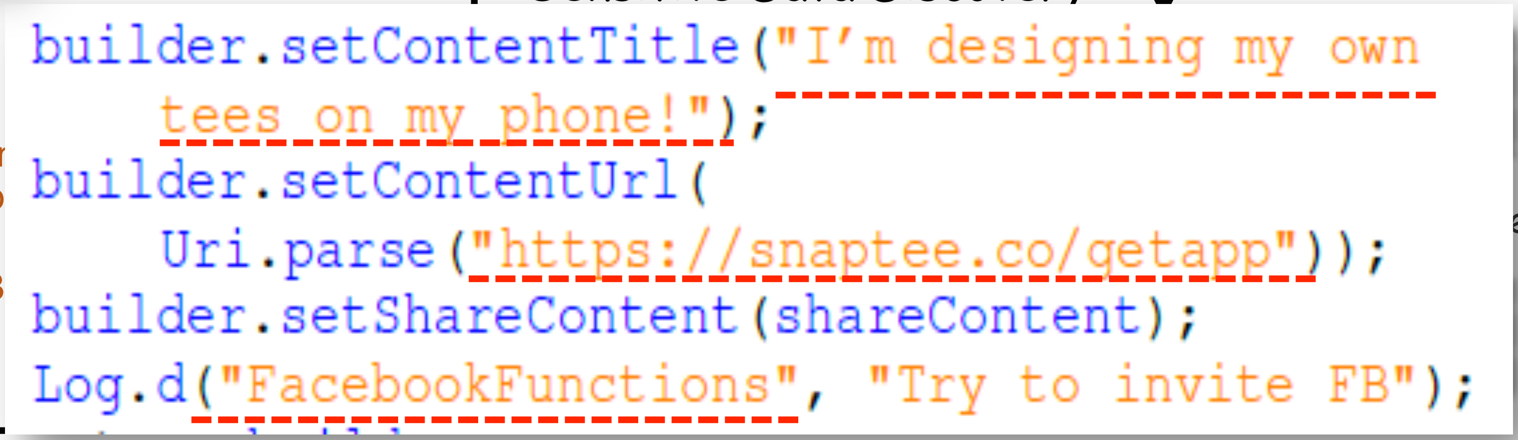
```
## Getting user profile on Facebook
JsonObject getUserFbProfile(HashMap userBasicInfo) {
   JsonObject userJson = UserBasicInfo.toJson();
                                               Addr);

                                               s null", exception);

 }
 ## Sharing content To Facebook
Builder shareToFacebook(String shareContent)
{
       Builder builder = new Builder();
       builder.setContentTitle("I'm designing my owr
       builder.setContentUrl( Uri.parse("https://snap
       builder.setShareContent(shareContent);
       Log.d("FacebookFunctions", "Try to invite FB
       return builder;
}
```

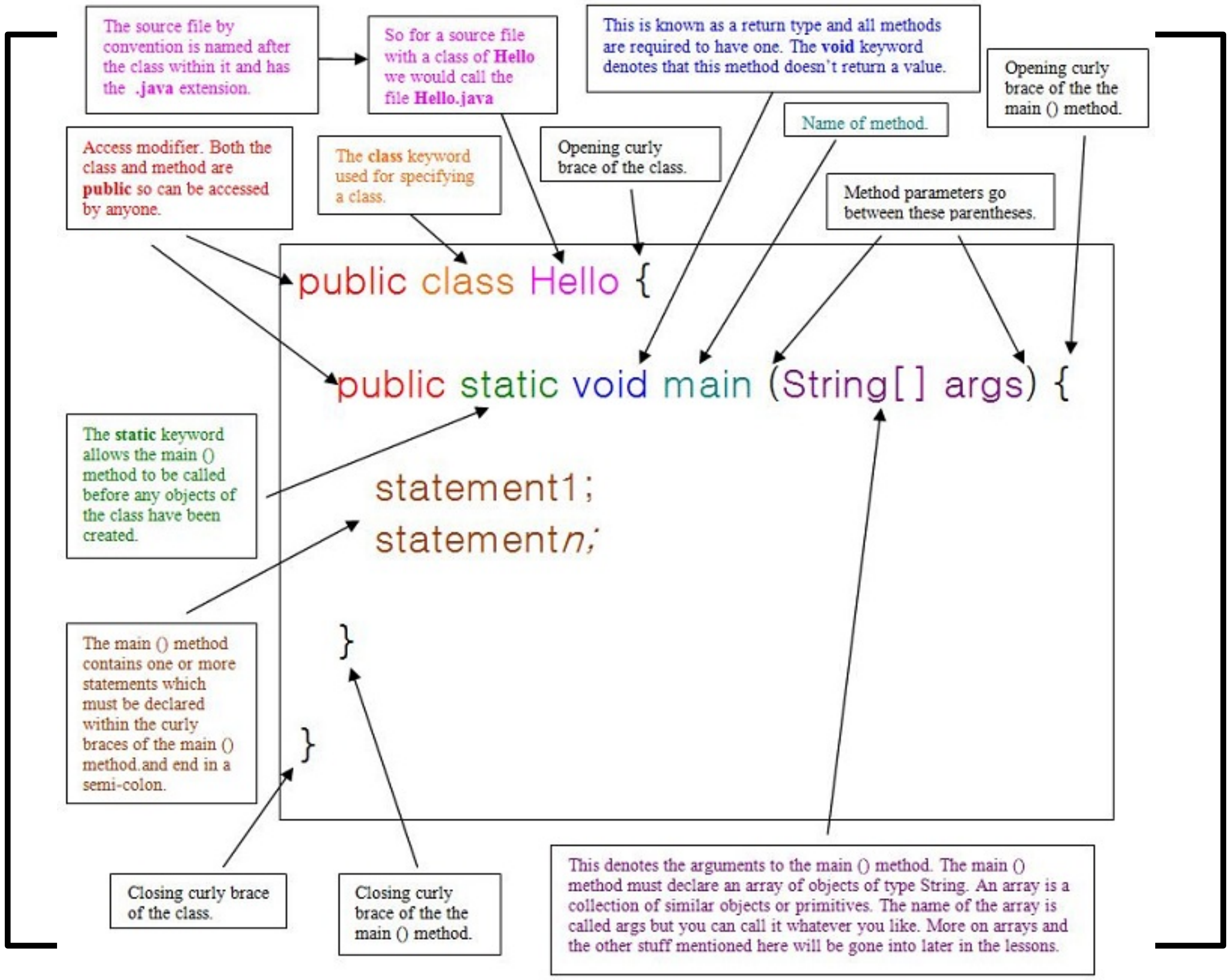






```
builder.setContentTitle("I'm designing my own
    tees on my phone!");
builder.setContentUrl(
    Uri.parse("https://snaptee.co/getapp"));
builder.setShareContent(shareContent);
Log.d("FacebookFunctions", "Try to invite FB");
```

ClueFinder Design



The source file by convention is named after the class within it and has the .java extension.

So for a source file with a class of **Hello** we would call the file **Hello.java**

This is known as a return type and all methods are required to have one. The **void** keyword denotes that this method doesn't return a value.

Opening curly brace of the the main () method.

Access modifier. Both the class and method are **public** so can be accessed by anyone.

The **class** keyword used for specifying a class.

Opening curly brace of the class.

Name of method.

Method parameters go between these parentheses.

```
public class Hello {

    public static void main (String[ ] args) {

        statement1;
        statementn;

    }
}
```

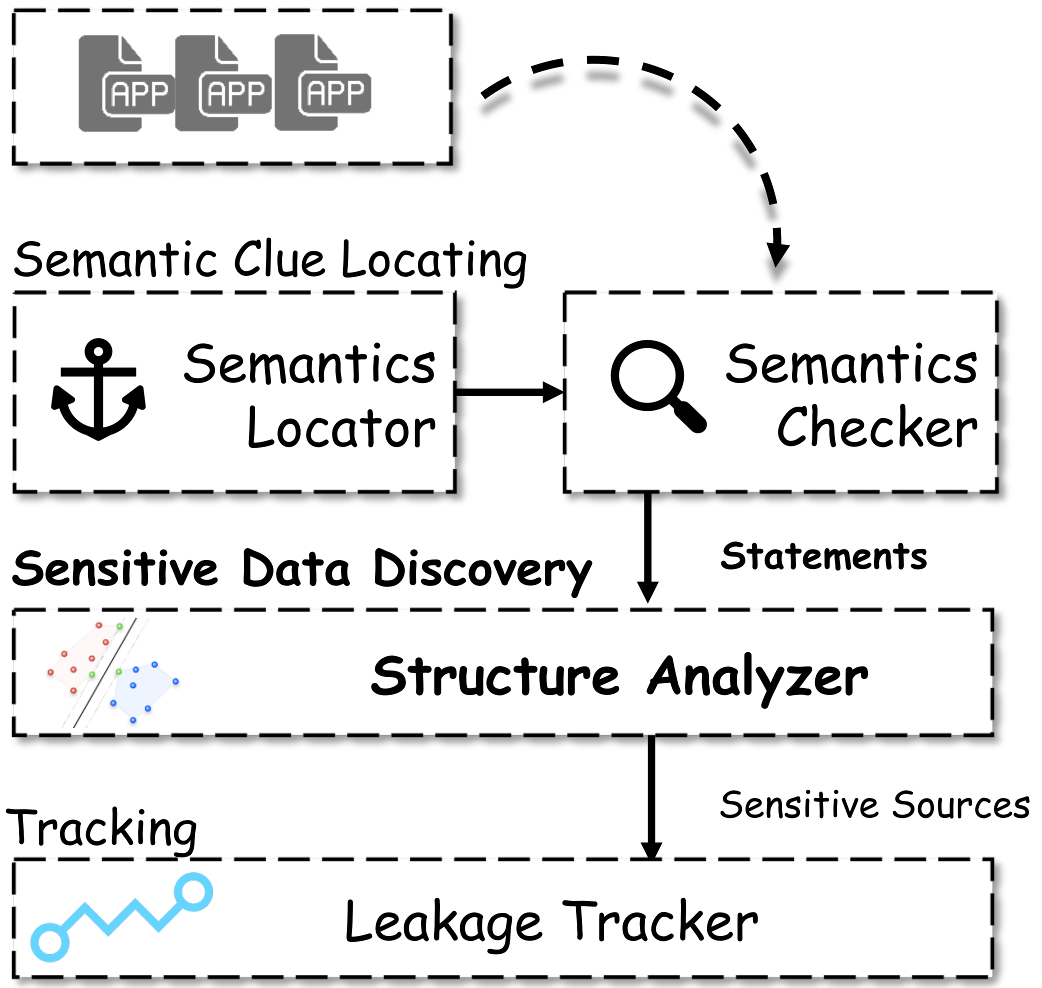
The **static** keyword allows the main () method to be called before any objects of the class have been created.

The main () method contains one or more statements which must be declared within the curly braces of the main () method.and end in a semi-colon.

Closing curly brace of the class.

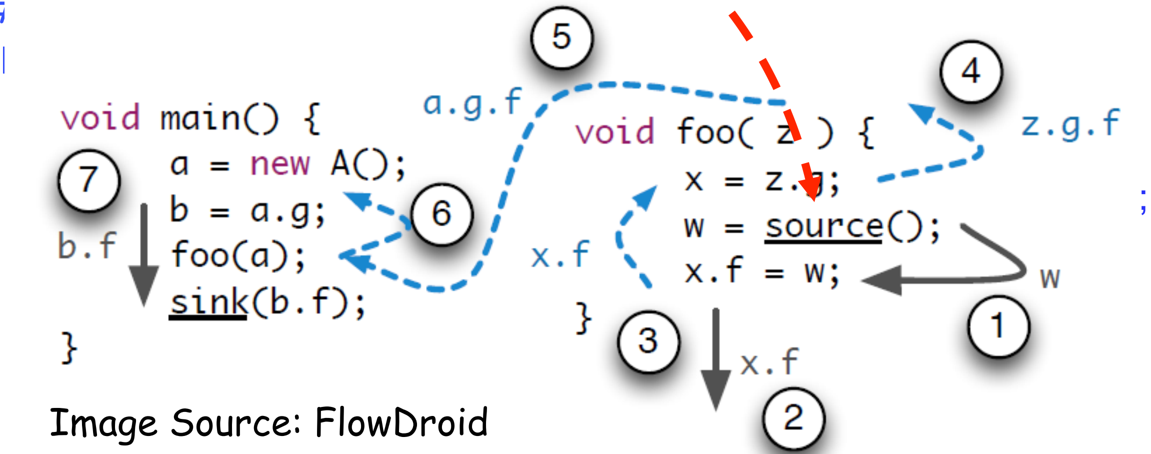
Closing curly brace of the the main () method.

This denotes the arguments to the main () method. The main () method must declare an array of objects of type String. An array is a collection of similar objects or primitives. The name of the array is called args but you can call it whatever you like. More on arrays and the other stuff mentioned here will be gone into later in the lessons.



ClueFinder Design

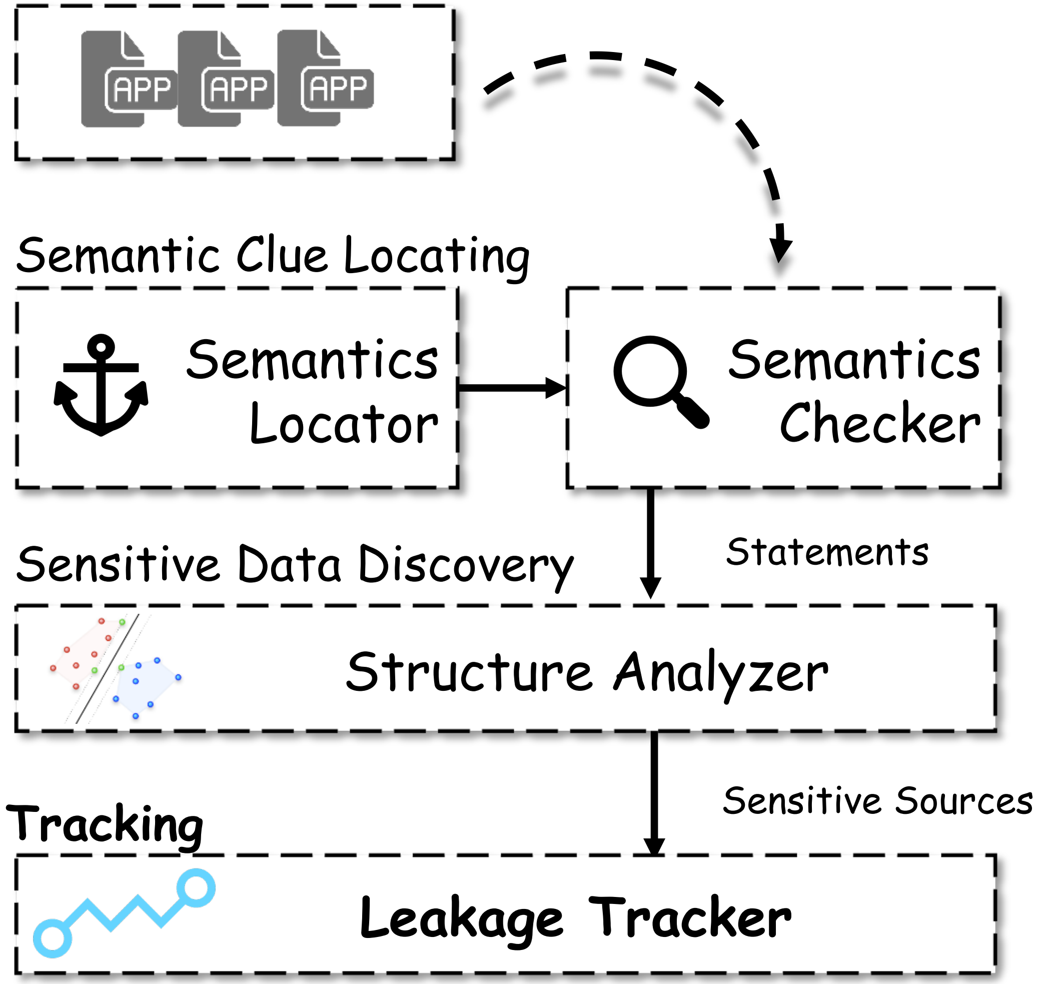
```
## Getting user profile on Facebook
JsonObject getUserFbProfile(HashMap userBasicInfo) {
    JsonObject userJson = UserBasicInfo.toJson();
        If(userJson .contains("home_addr")){
                jsonObject.put("home_addr", this.homeAddr);
        }
        this.uri = jsonObject.get("userProfile_uri");
        if(this.uri == null) {
              throwNullPointerException("Profile URI is null", exception);
        }
        return jsonObject;
}
#
Buil
{
                                                              ;
}
```



```
void main() {
    a = new A();
    b = a.g;
    foo(a);
    sink(b.f);
}
```

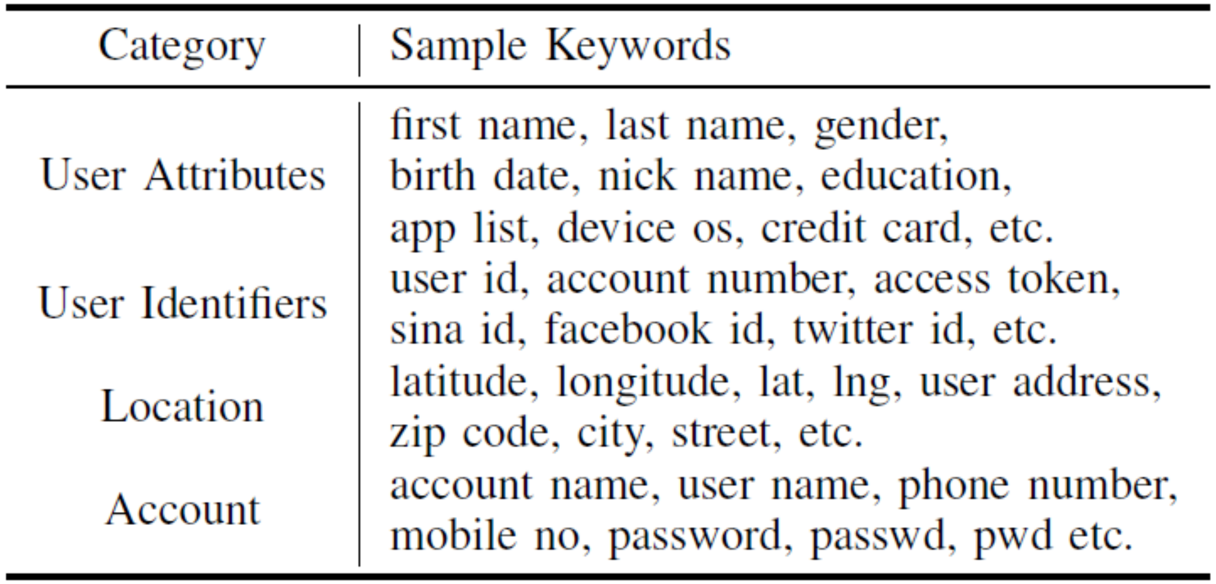
```
void foo( z ) {
    x = z.g;
    w = source();
    x.f = w;
}
```

Image Source: FlowDroid



Semantic Clue Locating

- Semantics Locator
  - Knowledge base : 35 privacy items
    - Google Privacy Policy, Financial Times report, prior research, etc.
  - Resources in focus
    - Method names
    - Variable names
    - Constant strings

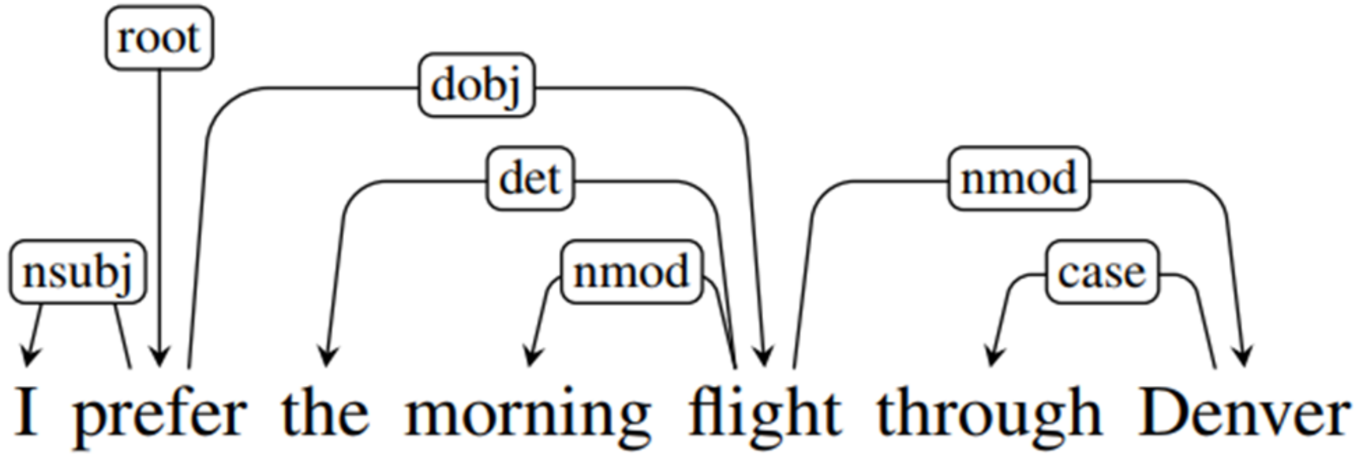


Category	Sample Keywords
User Attributes	first name, last name, gender, birth date, nick name, education, app list, device os, credit card, etc.
User Identifiers	user id, account number, access token, sina id, facebook id, twitter id, etc.
Location	latitude, longitude, lat, lng, user address, zip code, city, street, etc.
Account	account name, user name, phone number, mobile no, password, passwd, pwd etc.

Semantic Clue Locating

- Semantics Checker
  - Goal: In-depth semantic analysis for **privacy-related tokens**
  - Typed-dependency parsing

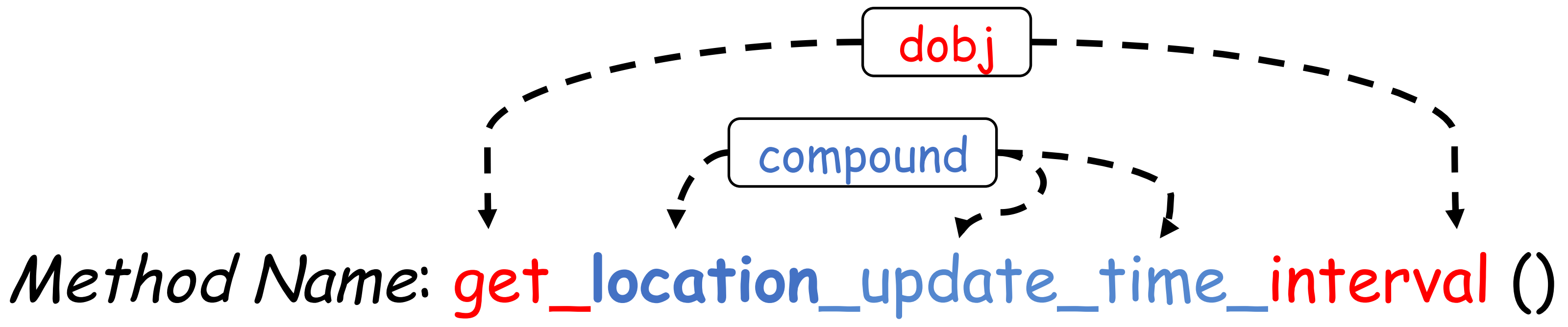
- Direct-object relation (dobj)
- Nominal subject (nsubj)
- Negation modifier (neg)
- …



Source: *Stanford typed dependencies manual.* Stanford University, 2008.

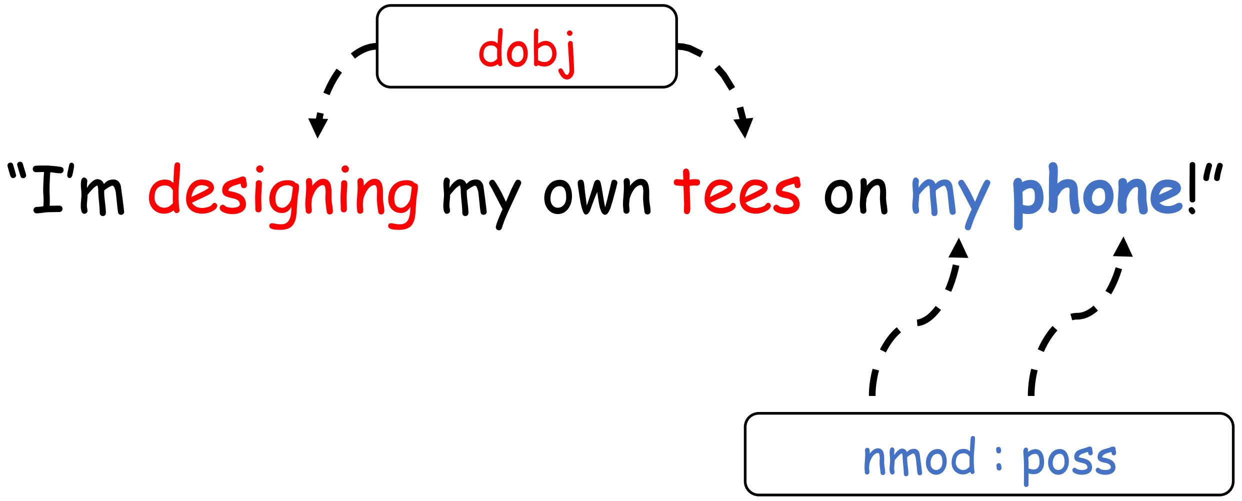
Semantic Clue Locating

- Semantics Checker
  - Goal: In-depth semantic analysis for **privacy-related tokens**
  - Typed-dependency parsing



Semantic Clue Locating

- Semantics Checker
  - Goal: In-depth semantic analysis for **privacy-related tokens**
  - Typed-dependency parsing



Sensitive Data Discovery

Privacy-related semantics **!=** **Sensitive data**

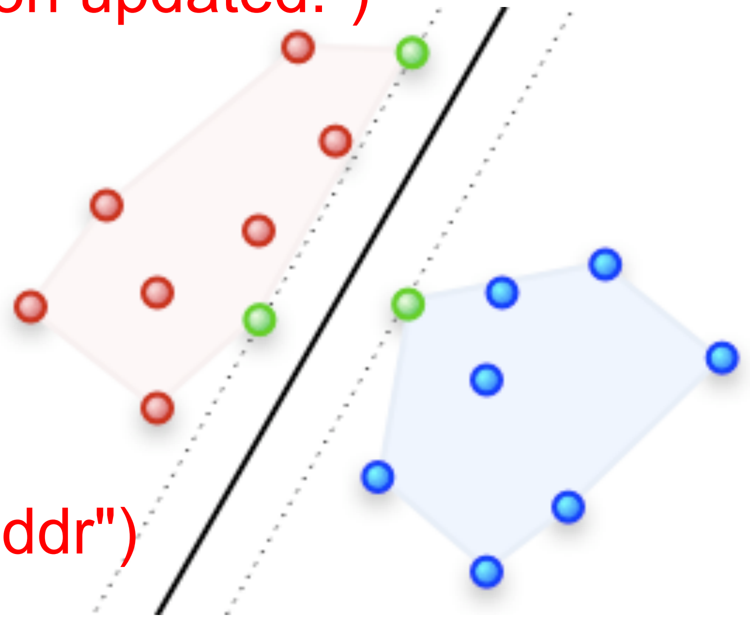
```
If ( userJson.contains ("home_addr") ) ✗
{
        .....
     ✓ userJson.get ( "home_addr" )
}

Else {
     ✓ userJson.put ("home_addr", this. homeAddr)
        Log.d ("location_info", "location updated.") ✗
}
```

Sensitive Data Discovery

- Structure Analyzer
  - SVM classifier for identifying **data objects**

Log.d ("Location_info", "location updated.")



userJson.get("home_addr")

Non-data objects

Data objects

userJson .contains("home_addr")

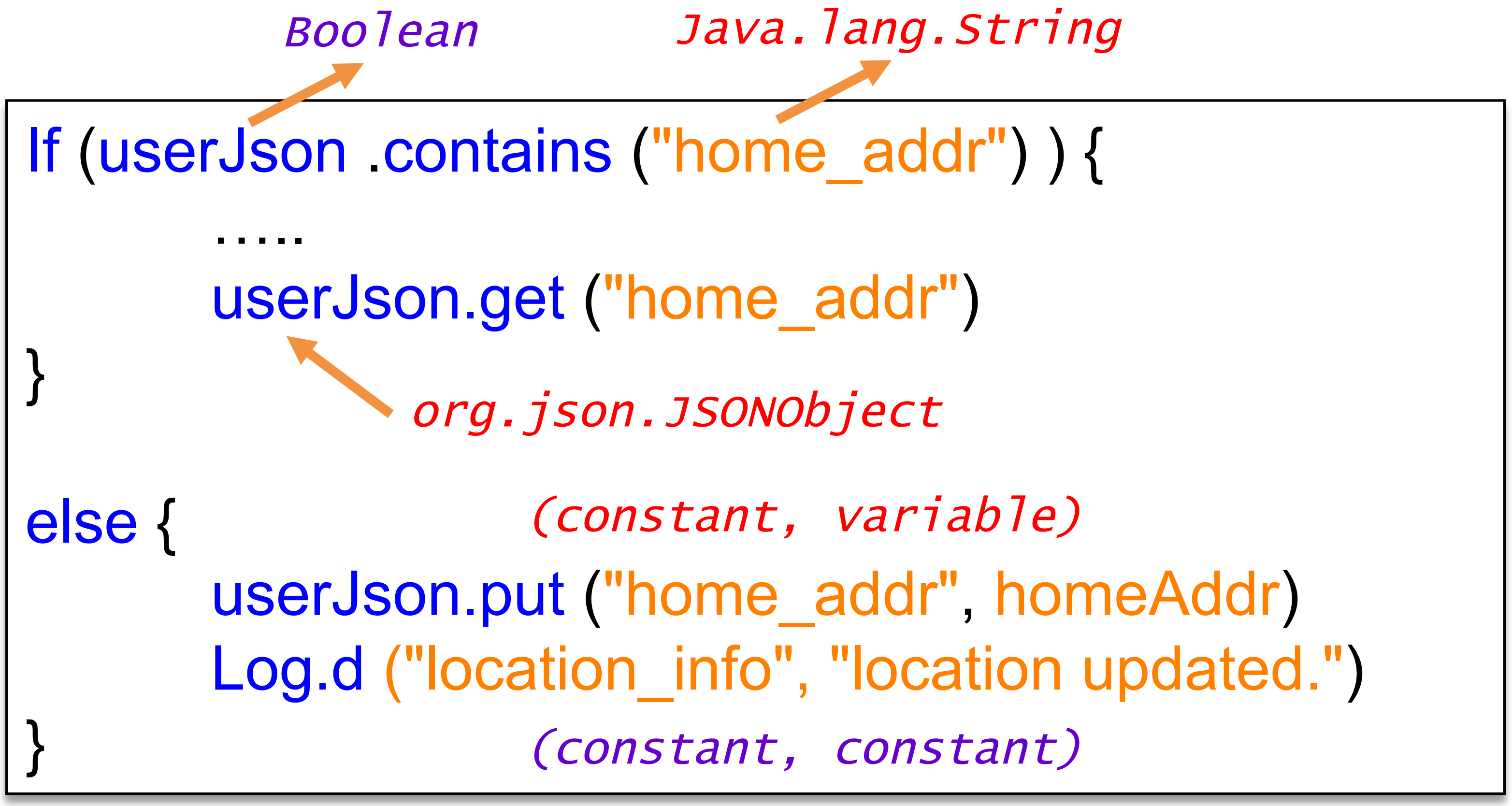
userJson.put("home_addr", this. homeAddr

Sensitive Data Discovery

- Structure Analyzer
  - SVM classifier for identifying **data objects**
  - Selected features

Data read or write operations:

- **Method name**
- **Parameter/Return type**
- **Base value (Class) type**
- **Constant-variable pattern**



```
If (userJson .contains ("home_addr") ) {
        .....
        userJson.get ("home_addr")
}

else {
        userJson.put ("home_addr", homeAddr)
        Log.d ("location_info", "location updated.")
}
```

Leakage Tracker

- Integrate with existing framework
  - Sources: **parameters** or **return values** in identified statements
  - Data-flow based taint analysis
    - E.g., FlowDroid [PLDI'14], Epicc [Security'13]

Evaluation

- Overall effectiveness
  - Manual validation
    - 100 randomly selected popular apps from Google Play
  - Final precision: 91.5%

- FP/FN analysis
  - Insufficient semantic analysis

```
void saveEvent("init", "put access token to extras", $r1);
```

  - Cases not covered by our labeled training set

```
Integer gender = getUserGender(user);
```

Limitation

- Obfuscation
  - Limited to deeply obfuscated code with all its semantic information removed
  - Preserved semantic information under moderate obfuscation
    - System-level methods (APIs)
    - Reflections
    - Interfaces of third-party SDKs
  - 11.3% (426/3,775) of the statements were obfuscated in our testing dataset.

Measurement Highlight

Dataset	Collect Time	Total Apps
Play-2015	Nov.15 - Dec.15	13,500
Play-2016	Jul.16 - Aug.16	71,686
Tencent-2015	Feb.15 - Apr.15	169,051
Tencent-2016	Jun.16 - Jul.16	191,431
Total	Nov.15 - Aug.16	445,668

- Seek for data exposure risk to third-party libraries
- 118,296 apps (26.5%) leak private user data
  - Exclude system controlled sources (e.g., IMEI, ICCID)

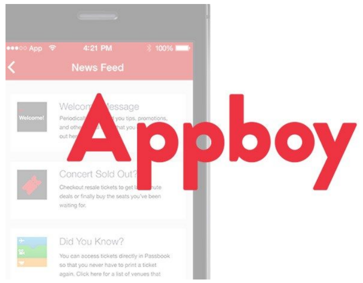
Leakage Patterns

- Third-party libraries
  - Deliberately harvest information from apps
    - E.g., Social network Sharing SDK (over 30% + share market in China)

- Apps developers
  - Give private information in apps to third-party libraries through API interfaces
    - Intended information disclosure and over-sharing

Intended Disclosure

- Popular Dating App



- Each time user seeks for a nearby potential dating target, the app sends user's **precise location**, **bio information**, **dating targets**, **name on Instagram**, etc. to *Appboy*.

"user":{"Can Create Group":true,"Seeking Distance":50,"Account Creation Date":"2016-11-17T16:56:32.163Z","Profile Enabled Groups":false,**"gender":"f","Seeking Gender":1,"Group Status":0,"Has Work Info":true,"Has Education Info":true,"Instagram"**:"***Susan_*****","Has e0c000e0a2b9" , "start_time" : 1.479401816693E9, "events" : [{"n":"lr","d"**:{"ll_accuracy":19.80900001525879,“altitude":0,"longitude":-86.47*,"latitude": 39.16*}**

Conclusion

- ClueFinder
  - A novel technique for identifying sensitive sources
  - Extend scope for labelling more sensitive data from app code
- Large scale measurement
  - Privacy exposure risk to third-party libraries
  - Highlight the importance of data protection in today's software composition

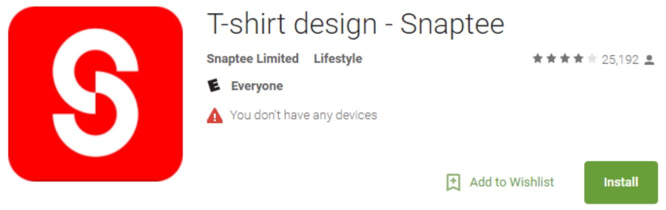
Thanks！

Q&A

- nanyuhong@fudan.edu.cn

Over-sharing

- SnapTee
  - Customize Tee design and shopping.
  - Installs 1,000,000 - 5,000,000
- MixPanel
  - “understand who your users are, see what they do before or after they sign up”



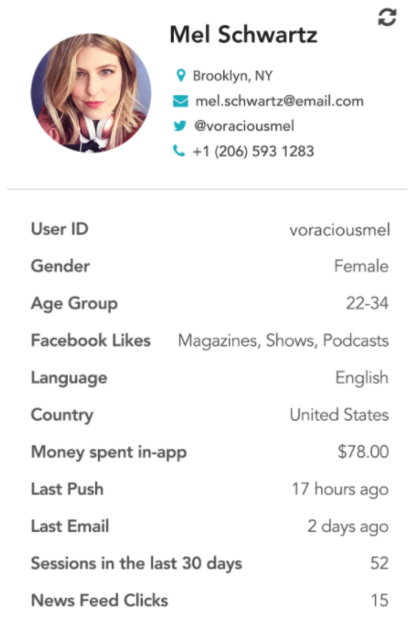
T-shirt design - Snaptee

Snaptee Limited Lifestyle ★★★★ 25,192

Everyone

You don't have any devices

```
{"$set":
   {"$username":"p***t",
        ..,"$email":"li**v@gmail.com",
   ..,"$first_name":"John",
         "$last_name":"Smith",
   "Twitter":"795**16"},
         "$token":"f81d***cdf96",
   "$time":"1479324910201",...}
}
```



Mel Schwartz

Brooklyn, NY

mel.schwartz@email.com

@voraciousmel

+1 (206) 593 1283

User ID	voraciousmel
Gender	Female
Age Group	22-34
Facebook Likes	Magazines, Shows, Podcasts
Language	English
Country	United States
Money spent in-app	$78.00
Last Push	17 hours ago
Last Email	2 days ago
Sessions in the last 30 days	52
News Feed Clicks	15

Evaluation

- Implementation
  - Java (1,604 LoCs) and Python (609 LoCs)
  - Extends FlowDroid framework
  - Stanford Parser for NLP analysis (in Java)
  - SVM from scikit-learn (in Python)
- Experimental Settings
  - 32-core server
  - Linux 2.6.32 kernel
  - 64GB memory

Evaluation

- Classifier for Structure Analyzer
  - Training
    - Randomly selected Statements from 100 popular apps
      - Processed by **Semantic Clue Locating** first
    - 4,326 manually labelled statements
      - Half positive and Half Negative
  - Effectiveness
    - 92.7% precision and 97.2% recall
      - Based on ten-fold cross validation (over labelled dataset)

Landscape

- 118,296 apps (26.5%) leaking private user data to 3,502 third-party libraries.
  - Exclude system controlled sources (e.g., IMEI, ICCID)
- Play-15 (most popular apps on GP) dataset, was found to have 39.9% of its apps leaking out user data
  - Half of the flagged method invocations (53.1%) are related to HTTP connections

DataSet	Affected Apps			Affected Libs	
	% Apps	Avg.Items/App	Avg.Libs/ App	# Libs	Avg. Items/ Lib
Play-2015	39.9%	7.6	2.83	709	2.45
Play-2016	22.8%	5.26	1.32	1,011	2.36
Tencent-2015	26.3%	7.55	1.64	2,315	2.43
Tencent-2016	27.3%	9.53	2.1	3,097	2.33
Total	26.5%	8.07	1.97	3,502	2.39

Landscape

- Averagely, each app exposes 8.07 data items (e.g., an identifier name, location, etc.) to 1.97 libraries.
- Individual apps on the un-official market (Tencent) tend to integrate more third-party libraries (1.32 vs. 2.1).

DataSet	Affected Apps			Affected Libs	
	% Apps	Avg.Items/App	Avg.Libs/ App	# Libs	Avg. Items/ Lib
Play-2015	39.9%	7.6	2.83	709	2.45
Play-2016	22.8%	5.26	1.32	1,011	2.36
Tencent-2015	26.3%	7.55	1.64	2,315	2.43
Tencent-2016	27.3%	9.53	2.1	3,097	2.33
Total	26.5%	8.07	1.97	3,502	2.39