SUPOR: Precise and Scalable Sensitive User Input Detection for Android Apps

Jianjun Huang, Zhichun Li, Xusheng Xiao, Zhenyu Wu, Kangjie Lu, Xiangyu Zhang, Guofei Jiang
Sensitive Data Disclosures

Disclosed to public

Hijacked/maliciously retrieved
Sensitive Data Disclosures

Local Storage Disclosed to public
Hijacked/maliciously retrieved

8/14/15
USENIX Security 2015
Sensitive Data Disclosures

Local Storage

Disclosed to public

Hijacked/maliciously retrieved
Sensitive Data Disclosures

- Local Storage disclosed to public
- Local Storage hijacked/maliciously retrieved

8/14/15 USENIX Security 2015
Sensitive Data Disclosures

- Local Storage disclosed to public
- Hijacked/maliciously retrieved

Date: 8/14/15

USENIX Security 2015
Sensitive Data

• **Existing work** focused on sensitive data defined by certain API methods.

TaintDroid[^OSDI'10^], AndroidLeaks[^TRUST'12^], FlowDroid[^PLDI'14^]

PiOS[^NDSS'11^]
Sensitive Data

• *Existing work* focused on sensitive data defined by certain API methods.
  • Most of them are permission protected
  • E.g., in Android, `TelephonyManager.getDeviceId()`
Sensitive User Inputs

• We are among the first to detect user inputs as sensitive sources in mobile apps.
  • None of them are permission protected
  • E.g., user id/password, credit card number...
Sensitive User Inputs

• We are among the *first* to detect user inputs as sensitive sources in mobile apps.
  • None of them are permission protected
  • E.g., user id/password, credit card number...

---

Credit card type
Select Card Type

Card number
15 or 16 digit

Expiration date
MM - YYYY

Sensitive
Sensitive User Inputs

• We are among the *first* to detect user inputs as sensitive sources in mobile apps.
  • None of them are permission protected
  • E.g., user id/password, credit card number...
Example User Inputs Disclosures

```java
1  EditText txtCN = findViewById(R.id.cardnum);
2  String cnum = txtCN.getText().toString();
3  ...
```

HTTP

Web Server

11/8/14 USENIX Security 2015
Example User Inputs Disclosures

1. EditText txtCN = findViewById(R.id.cardnum);
2. String cnum = txtCN.getText().toString();
3. ...

HTTP

1. EditText txtCM = findViewById(R.id.comment);
2. String comment = txtCM.getText().toString();
3. ...

HTTP

Web Server
Research Problems

• How to systematically discover the input fields from an app’s UI?

• How to identify which input fields are sensitive?

• How to associate the sensitive input fields to the corresponding variables in the apps that store their values?
Research Problems

• How to systematically discover the input fields from an app’s UI?

• How to identify which input fields are sensitive?

• How to associate the sensitive input fields to the corresponding variables in the apps that store their values?
Intuition

• From the user’s perspective, if we can mimic how a user looks at the UIs, we can determine which input fields can contain sensitive data within the UI context.
Feasibility

• Render the statically defined UI layouts
Feasibility

• Render the statically defined UI layouts

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>iOS</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout format</td>
<td>XML</td>
<td>NIB/XIB/Storyboard</td>
<td>XAML/HTML</td>
</tr>
<tr>
<td>Static UI Render</td>
<td>ADT</td>
<td>Xcode</td>
<td>Visual Studio</td>
</tr>
<tr>
<td>APIs map widgets to code</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Feasibility

• Render the statically defined UI layouts

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>iOS</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout format</td>
<td>XML</td>
<td>NIB/XIB/Storyboard</td>
<td>XAML/HTML</td>
</tr>
<tr>
<td>Static UI Render</td>
<td>ADT</td>
<td>Xcode</td>
<td>Visual Studio</td>
</tr>
<tr>
<td>APIs map widgets to code</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

• Associate labels to input fields based on physical locations
SUPOR: Sensitive User Input detector
Background - UI

Credit card type
Select Card Type

Card number
15 or 16 digit

Expiration date
MM - YYYY
Background - UI

Credit card type
Select Card Type

Card number
15 or 16 digit

Expiration date
MM - YYYY

Text Label
Background - UI

Credit card type
Select Card Type

Card number
15 or 16 digit

Expiration date
MM - YYYY

Text Label
Input Field
Background - UI

Credit card type
Select Card Type

Card number
15 or 16 digit

Expiration date
MM - YYYY

Text Label
Input Field
Input Hint
Background - UI

Credit card type
Select Card Type

Card number
15 or 16 digit

Expiration date
MM - YYYY

Text Label
Input Field
Input Hint
Widget
Background – Layout File

• A piece in an Android layout example.

```xml
<EditText

    android:id="@+id/pwd"

    android:inputType="textPassword" />
```
Background – Layout File

• A piece in an Android layout example.

```xml
<EditText
    android:id="@+id/pwd"
    android:inputType="textPassword"/>
```
Background – Layout File

• A piece in an Android layout example.

<EditText
    android:id="@+id/pwd"
    android:inputType="textPassword"/>

Interesting Attribute
Overview of SUPOR
Parsing Layout

• We need to know which layout files contain input fields.

Is Sensitive User Input Detection Needed?
Parsing Layout

• We need to know which layout files contain input fields.

Is Sensitive User Input Detection Needed?

Layout file

- layout contains input fields
- layout doesn’t contain input fields
Rendering UI

• Statically render layout files to UIs as users look at on smartphones via tools like ADT in Android.
Rendering UI

- Statically render layout files to UIs as users look at on smartphones via tools like ADT in Android.
Rendering UI

• Statically render layout files to UIs as users look at on smartphones via tools like ADT in Android.
Extracting Information

Credit card type
Select Card Type

Card number
15 or 16 digit

Expiration date
MM - YYYY
Extracting Information

Credit card type
Select Card Type

Card number
15 or 16 digit

Expiration date
MM - YYYY
Extracting Information

Collect information

- **Text Label**
  - Text: Card Number
  - Coordinates: [16, 231, 109, 249]
- **Input Field**
  - Hint: 15 or 16 digit
  - Coordinates: [16, 249, 464, 297]
UI Sensitiveness Analysis
UI Sensitiveness Analysis

Sensitive Attributes in Layout Files

```xml
<EditText android:id="@+id/pwd"
    android:inputType="textPassword" />
```
UI Sensitiveness Analysis

Sensitive Attributes in Layout Files

Yes

The Input Field is Sensitive
UI Sensitiveness Analysis

Sensitive Attributes in Layout Files

Sensitive Input Hint

Enter Password

15 or 16 digit MM - YYYY

The Input Field is Sensitive

Yes

No
UI Sensitiveness Analysis

- **Sensitive Attributes in Layout Files**
- **The Input Field is Sensitive**
- **Sensitive Input Hint**

Flow:
- If **Yes**: 
  - **Enter Password**
  - If **Yes**: 
    - **The Input Field is Sensitive**
  - If **No**: 
    - **Sensitive Input Hint**
UI Sensitiveness Analysis

Sensitive Attributes in Layout Files

The Input Field is Sensitive

Sensitive Input Hint

Sensitive Text Label

Card number
Expiration date

Comment

8/14/15
USENIX Security 2015
UI Sensitiveness Analysis

Sensitive Attributes in Layout Files

Sensitive Input Hint

Sensitive Text Label

The Input Field is Sensitive

The Input Field is Insensitive
UI Sensitiveness Analysis

Sensitive Attributes in Layout Files

Sensitive Input Hint

Sensitive Text Label

The Input Field is Sensitive

The Input Field is Insensitive

Yes

No

Yes

No
UI Sensitiveness Analysis

Challenge: How to precisely associate the correlated text label to a given input field?
Associating Labels (1)

• Intuition: labels at different positions relative to the input field have different probabilities to be correlated.
Associating Labels (1)

• Intuition: labels at different positions relative to the input field have different probabilities to be correlated.
Associating Labels (2)

• Assign position-based weights based on empirical observations
  • The smaller the weight, the closer the correlation
Associating Labels (2)

- Assign position-based weights based on empirical observations
  - The **smaller** the weight, the **closer** the correlation

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>0.8</td>
<td><strong>Input Field</strong></td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>
## Associating Labels (2)

- Assign position-based weights based on empirical observations
  - The **smaller** the weight, the **closer** the correlation

<table>
<thead>
<tr>
<th>4</th>
<th>2</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td><strong>Input Field</strong></td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>
Associating Labels (2)

- Assign position-based weights based on empirical observations
  - The **smaller** the weight, the **closer** the correlation

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th></th>
<th>2</th>
<th></th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>Input Field</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td>9</td>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>
Associating Labels (3)

• Geometry-based correlation score computation
Associating Labels (3)

- Geometry-based correlation score computation

\[(x_1, y_1)\]
\[(x_2, y_2)\]

Label

Input Field \((I)\)
Associating Labels (3)

• Geometry-based correlation score computation

For each pixel \((x,y)\) in a text label
- \(\text{distance}(I, x, y) \times \text{posWeight}(I, x, y)\)

\((x_1, y_1)\)

(\(x_2, y_2\))

Input Field \((I)\)
Associating Labels (3)

• Geometry-based correlation score computation

For each pixel \((x,y)\) in a text label

- \(\text{distance}(I, x, y) \times \text{posWeight}(I, x, y)\)

Average the correlation score for the text label
Associating Labels (4)

• Find out the label with the smallest correlation score among all potential labels for a given input field
Associating Labels (4)

• Find out the label with the smallest correlation score among all potential labels for a given input field

<table>
<thead>
<tr>
<th>Credit card type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Card Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Card number</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 or 16 digit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expiration date</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM - YYYY</td>
</tr>
</tbody>
</table>
Associating Labels (4)

- Find out the label with the smallest correlation score among all potential labels for a given input field.

<table>
<thead>
<tr>
<th>Label</th>
<th>Number Field</th>
<th>Date Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit card type</td>
<td>265.57</td>
<td>456.42</td>
</tr>
<tr>
<td>Card number</td>
<td>76.47</td>
<td>271.23</td>
</tr>
<tr>
<td>Expiration date</td>
<td>205.29</td>
<td>75.40</td>
</tr>
</tbody>
</table>
Associating Labels (4)

- Find out the label with the smallest correlation score among all potential labels for a given input field

<table>
<thead>
<tr>
<th>Label</th>
<th>Number Field</th>
<th>Date Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit card type</td>
<td>265.57</td>
<td>456.42</td>
</tr>
<tr>
<td>Card number</td>
<td>76.47</td>
<td>271.23</td>
</tr>
<tr>
<td>Expiration date</td>
<td>205.29</td>
<td>75.40</td>
</tr>
</tbody>
</table>
Determining Sensitiveness (1)

<table>
<thead>
<tr>
<th>Card number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expiration date</td>
</tr>
<tr>
<td>Comment</td>
</tr>
</tbody>
</table>
Determining Sensitiveness (1)

• Keyword matching approach

Card number
Expiration date
Comment

敏感关键词数据集

是否匹配？
Determining Sensitiveness (1)

- Keyword matching approach

![Diagram showing keyword matching approach]
Determining Sensitiveness (1)

- Keyword matching approach

![Diagram showing the process of determining sensitiveness.]

- Sensitive Keywords Dataset
  - Card number
  - Expiration date
  - Comment

- Matches?
  - Yes: Sensitive
  - No: Insensitive
Determining Sensitiveness (2)

• Why is keyword matching approach effective?
Determining Sensitiveness (2)

• Why is keyword matching approach effective?

• Small screen and short phrases or sentences
Determining Sensitiveness (2)

• Why is keyword matching approach effective?

  • Small screen and short phrases or sentences

  • We only analyze the most relevant text label
Binding Variables (1)

<table>
<thead>
<tr>
<th>Credit card type</th>
<th>Select Card Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card number</td>
<td>15 or 16 digit</td>
</tr>
<tr>
<td>Expiration date</td>
<td>MM - YYYY</td>
</tr>
</tbody>
</table>
Binding Variables (1)

Credit card type
Select Card Type

Card number
15 or 16 digit

Expiration date
MM - YYYY
Binding Variables (1)

1. Widget txtCN = findViewById(X);
2. Data cnum = txtCN.getText();
3. // use of “cnum”
Binding Variables (1)

```
1 Widget txtCN = findViewById(X);
2 Data cnum = txtCN.getText();
3 // use of "cnum"
```
Binding Variables (2)

- Challenge: different widgets within one app have the same identifier

```
< TextView android:text="Card Number" />
< EditText android:id="@+id/input1" ... />
...
```

```
< TextView android:text="Search" />
< EditText android:id="@+id/input1" ... />
...
```
Binding Variables (2)

• Challenge: different widgets within one app have the same identifier

```xml
<TextView android:text="Card Number" />
<EditText android:id="@+id/input1" ... />
...

<TextView android:text="Search" />
<EditText android:id="@+id/input1" ... />
... 
```

```java
    txtInput1 = this.findViewById(input1);
    txtInput2 = this.findViewById(input1);
```
Binding Variables (2)

• Challenge: different widgets within one app have the same identifier

```xml
<TextView android:text="Card Number" />
<EditText android:id="@+id/input1" … />
...

<TextView android:text="Search" />
<EditText android:id="@+id/input1" … />
...
```

txtInput1 = this.findViewById(input1);

txtInput2 = this.findViewById(input1);

Binding Variables (3)

```
<TextView android:text="Card Number" />
<EditText android:id="@+id/input1" ...
...
```

```
[layout: billing_information.xml]
```

```
<EditText android:id="@+id/input1" ...
...
```

```
[layout: search.xml]
```

Sensitive

```
id/input1
```

Insensitive
Binding Variables (3)

```
<TextView android:text="Card Number" />
<EditText android:id="@+id/input1" />
...
```

[layout: billing_information.xml]

```
<TextView android:text="Search" />
<EditText android:id="@+id/input1" />
...
```

[layout: search.xml]

```
txtInput1 = this.findViewById(input1);
```

```
txtInput2 = this.findViewById(input1);
```

Sensitive

Insensitive
Binding Variables (3)

```java
binding Variables (3)

Binding Variables (3)

<textview android:text= "Card Number" />
<editText android:id="@+id/input1" ... />
...
[layout: billing_information.xml]

<textview android:text= "Search" />
<editText android:id="@+id/input1" ... />
...
[layout: search.xml]

Sensitive

Insensitive

id/input1

#1

#2

#3

#4

txtInput1 = this.findViewById(input1);

this.setContentViewById(billing_information);

8/14/15 USENIX Security 2015
Binding Variables (3)

```java
binding.variables

<TextView android:text="Card Number" />
<EditText android:id="@+id/input1" ... />
...
[layout: billing_information.xml]

Sensitive

id/input1

Sensive

<TextView android:text="Search" />
<EditText android:id="@+id/input1" ... />
...
[layout: search.xml]

```

txtInput1 = this.findViewById(input1);

dd.setView(billing_information);
```
Binding Variables (3)

```java
this.setInput1 = this.findViewById(input1);

this.setContentView(billing_information);

[layout: billing_information.xml]

< TextView android:text="Card Number" />
< EditText android:id="@+id/input1" ... />
...

[layout: search.xml]

< TextView android:text="Search" /> 
< EditText android:id="@+id/input1" ... />
...

Sensitive

Sensitive

id/input1

Sensible

Insensitive
```
Binding Variables (3)

```xml
< TextView android:text="Card Number" />
< EditText android:id="@+id/input1" ... />

[layout: billing_information.xml]
```

```java
id/input1
```

```java
txtInput2 = this.findViewById(input1);
```

```xml
< TextView android:text="Search" />
< EditText android:id="@+id/input1" ... />

[layout: search.xml]
```

Sensitive

Insensitive
Binding Variables (3)

```xml
<TextView android:text="Card Number" />
<EditText android:id="@+id/input1" ...

[layout: billing_information.xml]

<EditText android:id="@+id/input1" ...

[layout: search.xml]

Sensive

id/input1

Insensitive

txtInput2 = this.findViewById(input1);
this.setContentViewById(search);
```
Binding Variables (3)

<TextView android:text="Card Number" />
<EditText android:id="@+id/input1" />

[layout: billing_information.xml]

<EditText android:id="@+id/input1" />

[layout: search.xml]

 SUSCEPTIBLE

    id/input1

        txtInput2 = this.findViewById(input1);
        this.setContentView(search);

INSUSCEPTIBLE
Binding Variables (3)

```
<ContentView android:id="@+id/input1" ... />
...[layout: billing_information.xml]

Sensitive

id/input1

Insensitive

txtInput2 = this.findViewById(input1);
this.setContentView(search);
...[layout: search.xml]

Insensitive
```

<ContentView android:id="@+id/input1" ... />
...
Implementation & Evaluation

• Implemented for Android apps and built on Dalysis\textsuperscript{[CHEX CCS’12]}, IBM WALA and ADT.

• Only input fields of type EditText are analyzed, i.e. other user inputs like checkbox are ignored.

• Implemented a sensitive user inputs disclosure detection system by combining SUPOR and static taint analysis

• 16,000 apps were evaluated
Evaluating UI Sensitiveness Analysis (1)

• 9,653 apps (60.33%) contains input fields
  • Performance:
    • Average analysis time is **5.7 seconds for one app**
Evaluating UI Sensitiveness Analysis (2)

• 9,653 apps (60.33%) contains input fields
  • Accuracy
    • Manually examined 40 apps. 115 layouts are rendered and 485 input fields are analyzed.
    • TP: sensitive user inputs are identified as sensitive
    • FP: insensitive user inputs are identified as sensitive
    • FN: sensitive user inputs are identified as insensitive

\[
\text{Recall} = \frac{TP}{TP + FN} = 97.3\% \quad \text{Precision} = \frac{TP}{TP + FP} = 97.3\%
\]
• Causes for FN and FP

  • Insufficient context to identify sensitive keywords.
    • False negative: “Answer” vs “Security Answer”
    • False Positive: “Height” of an image file and for a human being
• Causes for FN and FP

  • Insufficient context to identify sensitive keywords.
    • False negative: “Answer” vs “Security Answer”
    • False Positive: “Height” of an image file and for a human being

  • Inaccurate text label association
    • False positive: e.g. the long sentence (with keyword “email”) is associated with the “Delivery Instructions” field
• Causes for FN and FP
  • Insufficient context to identify sensitive keywords.
    • False negative: “Answer” vs “Security Answer”
    • False Positive: “Height” of an image file and for a human being
  • Inaccurate text label association
    • False positive: e.g. the long sentence (with keyword “email”) is associated with the “Delivery Instructions” field
• Causes for FN and FP

  • Insufficient context to identify sensitive keywords.
    • False negative: “Answer” vs “Security Answer”
    • False Positive: “Height” of an image file and for a human being

  • Inaccurate text label association
    • False positive: e.g. the long sentence (with keyword “email”) is associated with the “Delivery Instructions” field
• Causes for FN and FP
  • Insufficient context to identify sensitive keywords.
    • False negative: “Answer” vs “Security Answer”
    • False Positive: “Height” of an image file and for a human being
  • Inaccurate text label association
    • False positive: e.g. the long sentence (with keyword “email”) is associated with the “Delivery Instructions” field
• Causes for FN and FP
  • Insufficient context to identify sensitive keywords.
    • False negative: “Answer” vs “Security Answer”
    • False Positive: “Height” of an image file and for a human being
  • Inaccurate text label association
    • False positive: e.g. the long sentence (with keyword “email”) is associated with the “Delivery Instructions” field
• Causes for FN and FP
  • Insufficient context to identify sensitive keywords.
    • False negative: “Answer” vs “Security Answer”
    • False Positive: “Height” of an image file and for a human being
  • Inaccurate text label association
    • False positive: e.g. the long sentence (with keyword “email”) is associated with the “Delivery Instructions” field

![Diagram of text label association error]
Evaluating Disclosure Analysis

• For all 16,000 apps
  • Throughput: 11.1 apps/minute
    • A cluster of 8 servers
    • 3 apps are analyzed on each server in parallel
Evaluating Disclosure Analysis

• For all 16,000 apps
  • Throughput: 11.1 apps/minute
    • A cluster of 8 servers
    • 3 apps are analyzed on each server in parallel
  • Manually examined 104 apps
  • False positive rate is 8.7%
    • Limitations of underlying taint analysis framework
      • E.g. lack of accurate modeling of arrays
Case Studies (1)

3 input fields associated with labels “Weight”, “Height” and “Age” are identified sensitive.
Case Studies (1)

**com.canofsleep.wwdiary**

3 input fields associated with labels “Weight”, “Height” and “Age” are identified sensitive.

**com.nitrogen.android**

The 3 marked inputs fields are identified sensitive and their data are disclosed.
Case Studies (2)

txtWeight = this.findViewById(R.id.edt_weight);
valWeight = txtWeight.getText().toString();
Log.i("weight", valWeight);
Case Studies (2)

- Disclosure analysis based on existing approach which directly define certain APIs as sensitive sources.

```java
int txtWeight = this.findViewById(R.id.edt_weight);

String valWeight = txtWeight.getText().toString();

Log.i("weight", valWeight);
```
Case Studies (2)

- Disclosure analysis based on **existing approach** which directly define certain APIs as sensitive sources.

```java
    txtWeight = this.findViewById(R.id.edt_weight);
    valWeight = txtWeight.getText().toString();
    Log.i("weight", valWeight);
```

Sink: Undetected
Case Studies (2)

- Disclosure analysis based on SUPOR

```java
    txtWeight = this.findViewById(R.id.edt_weight);

    valWeight = txtWeight.getText().toString();

    Log.i("weight", valWeight);
```

Source

```
    valWeight = txtWeight.getText().toString();
```

Sink

```
    Log.i("weight", valWeight);
```
Conclusion

• We study the possibility of detecting sensitive user inputs, an important yet mostly neglected sensitive source in mobile apps.
Conclusion

• We study the possibility of detecting sensitive user inputs, an important yet mostly neglected sensitive source in mobile apps.

• We propose SUPOR, among the first known approaches to detect sensitive user inputs with high recall and precision.
  • Mimics from the user’s perspective by statically and scalably rendering the layout files.
  • Leverages a geometry-based approach to precisely associated text labels to input fields.
  • Utilizes textual analysis to determine the sensitiveness of the texts in labels.
Conclusion

• We study the possibility of detecting sensitive user inputs, an important yet mostly neglected sensitive source in mobile apps.

• We propose SUPOR, among the first known approaches to detect sensitive user inputs with high recall and precision.
  • Mimics from the user’s perspective by statically and scalably rendering the layout files.
  • Leverages a geometry-based approach to precisely associated text labels to input fields.
  • Utilizes textual analysis to determine the sensitiveness of the texts in labels.

• We perform a sensitive user inputs disclosure analysis, with FP rate of 8.7%, to demonstrate the usefulness of SUPOR.
Related work

• A lot of work focus on privacy disclosure problems on predefined sensitive data sources in the phone. [FlowDroid PLDI’14, PiOS NDSS’11, AAPL NDSS’15]

• FlowDroid employs a limited form of sensitive input fields—password fields. [PLDI’14]

• AsDroid checks UI text to detect the contradiction between the expected behaviors and program behaviors. [ICSE’14]

• UIPicker uses supervised learning to collect sensitive keywords and corresponding layouts. It also uses the sibling elements in layout files as the description text for a widget. [USENIX Security’15]
Keyword dataset construction

- Crawl texts from apps’ resource files
- Adapt NLP techniques to extract nouns and noun phrases from the top 5,000 frequent text lines.
- Manually inspect top frequent nouns and noun phrases to identify sensitive keywords.
Why not use XML structure to compute correlation scores?

• Many developers define relative positions of the widgets, which are not what users perceive
  • XML structure in this case does not guarantee that sibling widgets are physically close.
Why not use XML structure to compute correlation scores?

• Some cases in real Android apps.
Why not use XML structure to compute correlation scores?

• Some cases in real Android apps.

```xml
<LinearLayout android:orientation="horizontal">
  <LinearLayout android:orientation="vertical">
    <TextView android:text="Label 1" />
    <TextView android:text="Label 2" />
  </LinearLayout>
</LinearLayout>
<LinearLayout android:orientation="vertical">
  <EditText android:id="@+id/input1" ... />
  <EditText android:id="@+id/input2" ... />
</LinearLayout>
</LinearLayout>
```
Why not use XML structure to compute correlation scores?

• Some cases in real Android apps.

```xml
<LinearLayout android:orientation="horizontal">
  <LinearLayout android:orientation="vertical">
    <TextView android:text="Label 1" />
    <TextView android:text="Label 2" />
  </LinearLayout>
</LinearLayout>

<LinearLayout android:orientation="vertical">
  <EditText android:id="@+id/input1" ...
  <EditText android:id="@+id/input2" ...
</LinearLayout>
</LinearLayout>
```