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

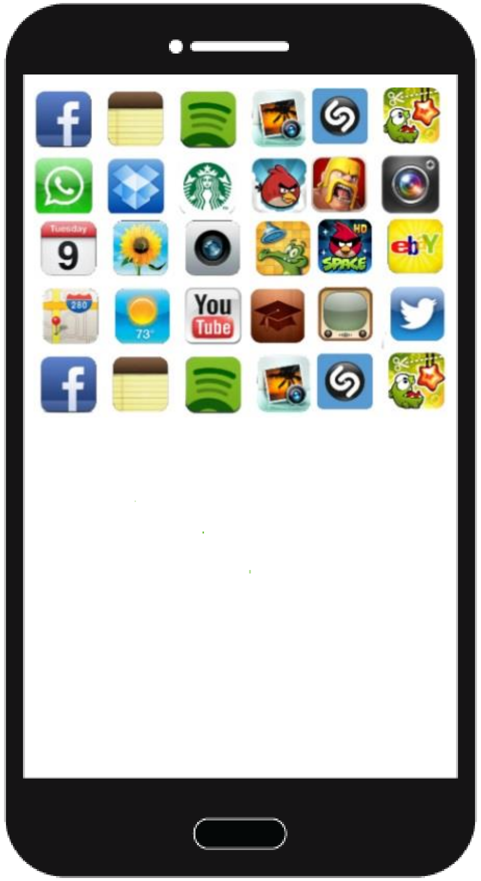
Jianjun Huang, Zhichun Li, Xusheng Xiao, Zhenyu Wu,
Kangjie Lu, Xiangyu Zhang, Guofei Jiang

PURDUE UNIVERSITY

NEC Laboratories America
Relentless passion for innovation

Georgia Tech

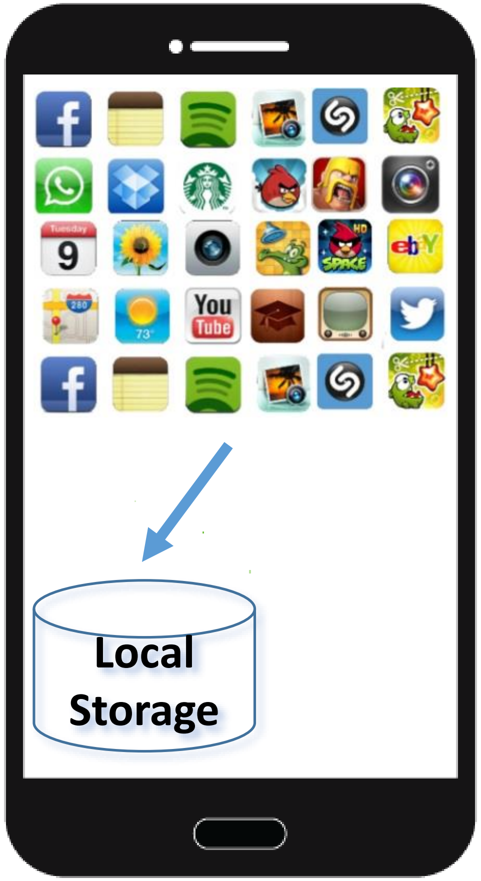
Sensitive Data Disclosures



→ Disclosed to public

→ Hijacked/maliciously retrieved

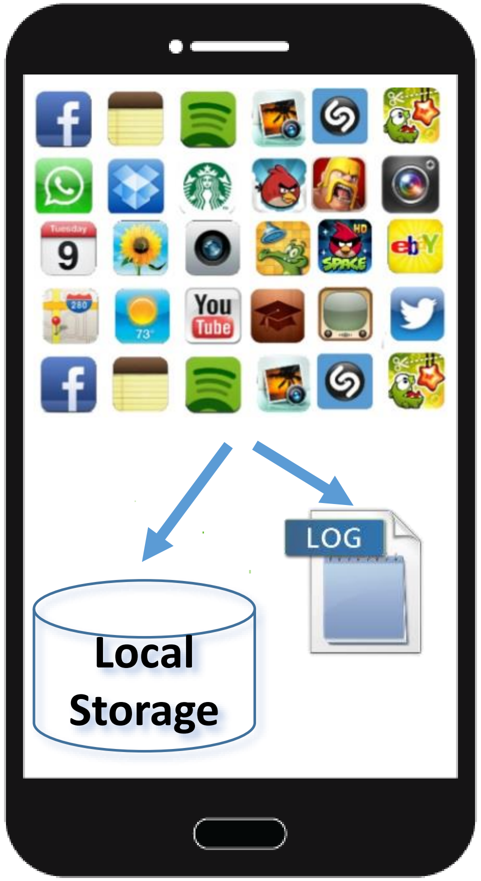
Sensitive Data Disclosures



→ Disclosed to public

→ Hijacked/maliciously retrieved

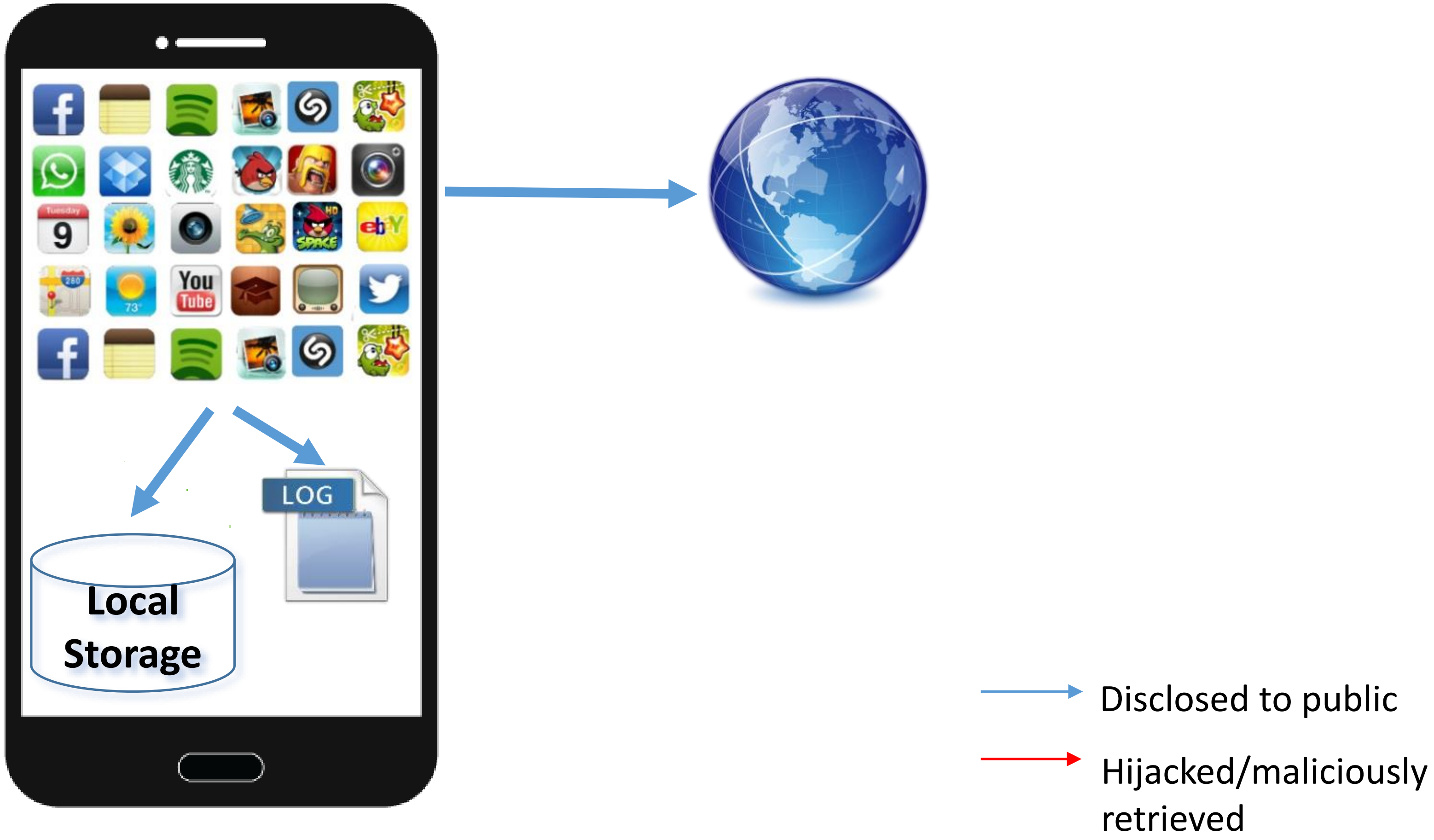
Sensitive Data Disclosures



→ Disclosed to public

→ Hijacked/maliciously retrieved

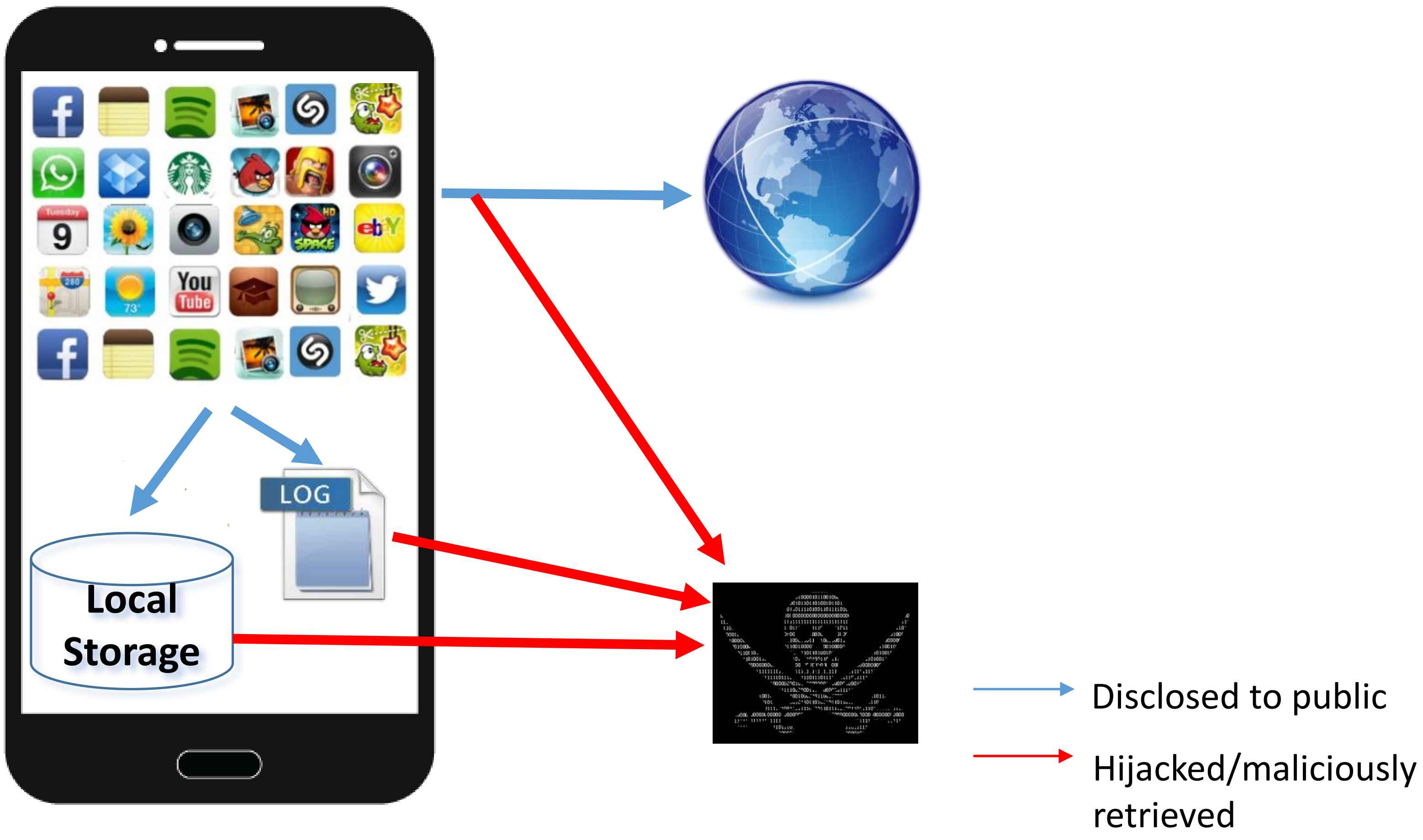
Sensitive Data Disclosures



→ Disclosed to public

→ Hijacked/maliciously retrieved

Sensitive Data Disclosures



Disclosed to public

Hijacked/maliciously retrieved

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

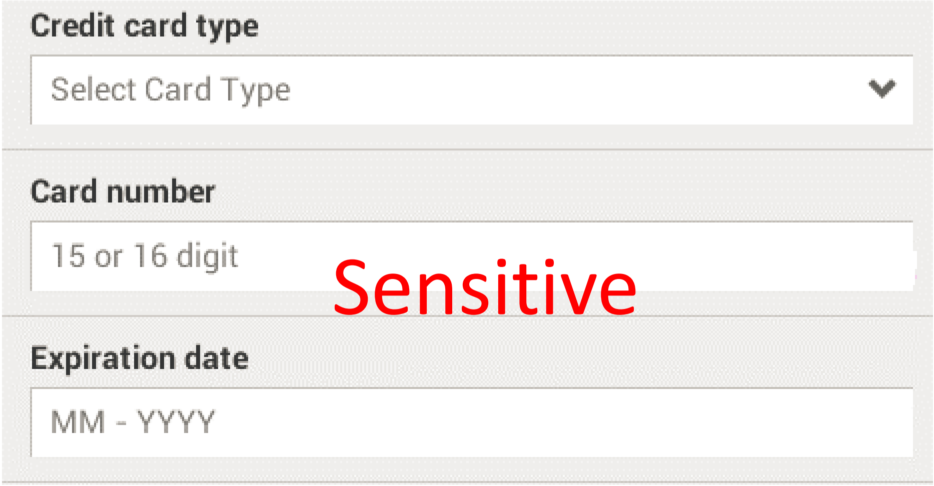
- Existing work focused on sensitive data defined by certain API methods.
  - Most of them are permission protected
  - E.g., in Android, `TelephonyManager.getDeviceId()`

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

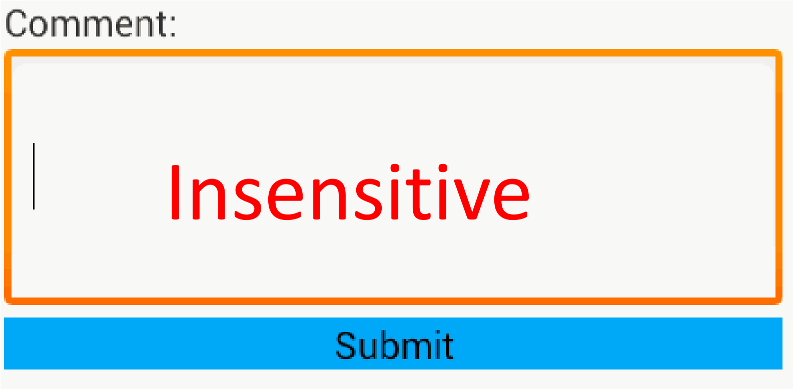
Expiration date

MM - YYYY

Sensitive

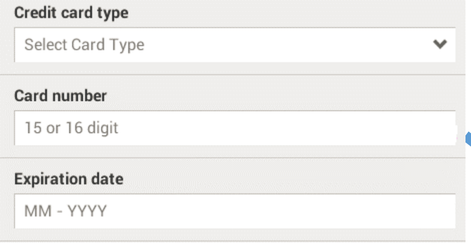
Sensitive User Inputs

- 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…



Insensitive

Example User Inputs Disclosures



Credit card type

Select Card Type

Card number

15 or 16 digit

Expiration date

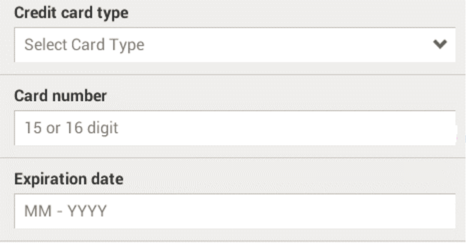
MM - YYYY

```
EditText txtCN = findViewById(R.id.cardnum);
String cnum = txtCN.getText().toString();
…
```

HTTP

Web Server

Example User Inputs Disclosures



Credit card type

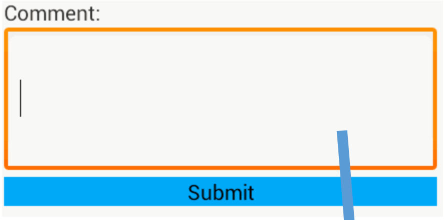
Select Card Type

Card number

15 or 16 digit

Expiration date

MM - YYYY



Comment:

```
EditText txtCN = findViewById(R.id.cardnum);
String cnum = txtCN.getText().toString();
…
```

```
EditText txtCM = findViewById(R.id.comment);
String comment = txtCM.getText().toString();
…
```

HTTP

HTTP

Web Server

Research Problems

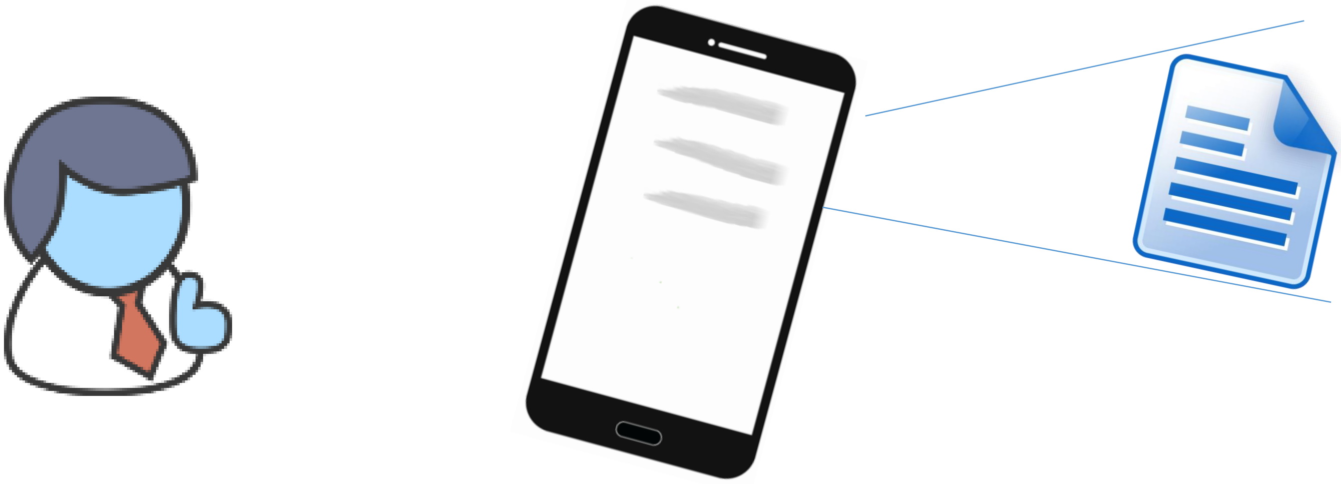
- How to systematically discover the input fields from an app’s UI?
- How to identify which input fields are sensitive?
- How to associate the sensitive input fields to the corresponding variables in the apps that store their values?

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

	Android	iOS	Windows Phone
Layout format	XML	NIB/XIB/Storyboard	XAML/HTML
Static UI Render	ADT	Xcode	Visual Studio
APIs map widgets to code	Yes	Yes	Yes

Feasibility

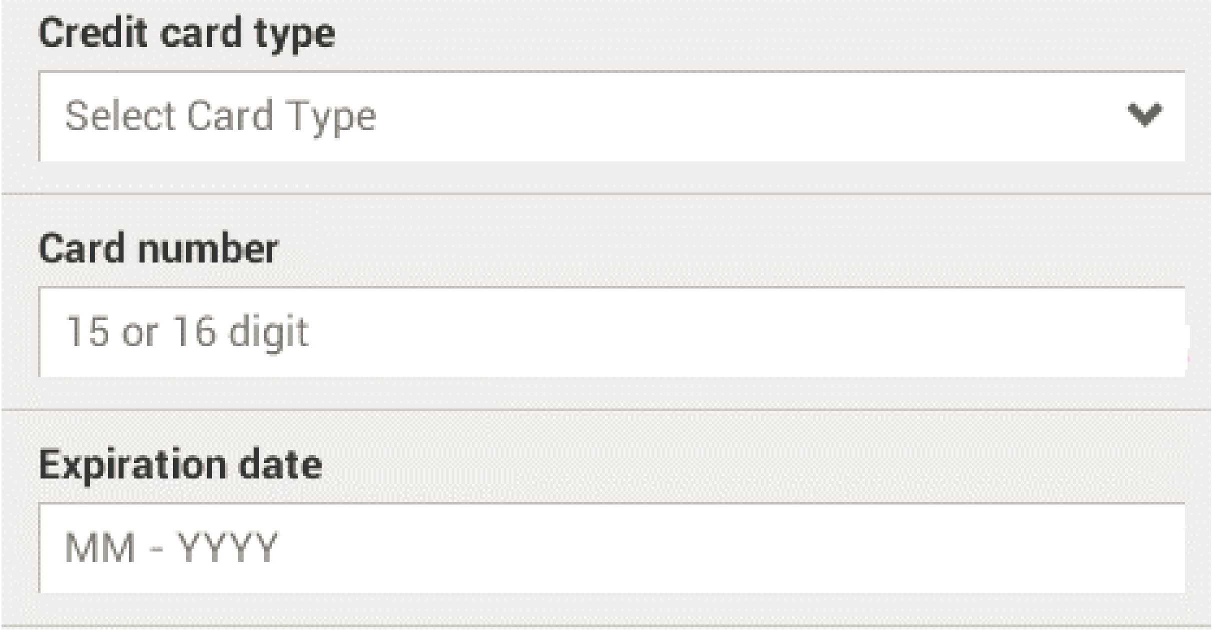
- Render the statically defined UI layouts

	Android	iOS	Windows Phone
Layout format	XML	NIB/XIB/Storyboard	XAML/HTML
Static UI Render	ADT	Xcode	Visual Studio
APIs map widgets to code	Yes	Yes	Yes

- Associate labels to input fields based on physical locations

SUPOR:
Sensitive User inPut detectOR

Background - UI



Credit card type

Select Card Type

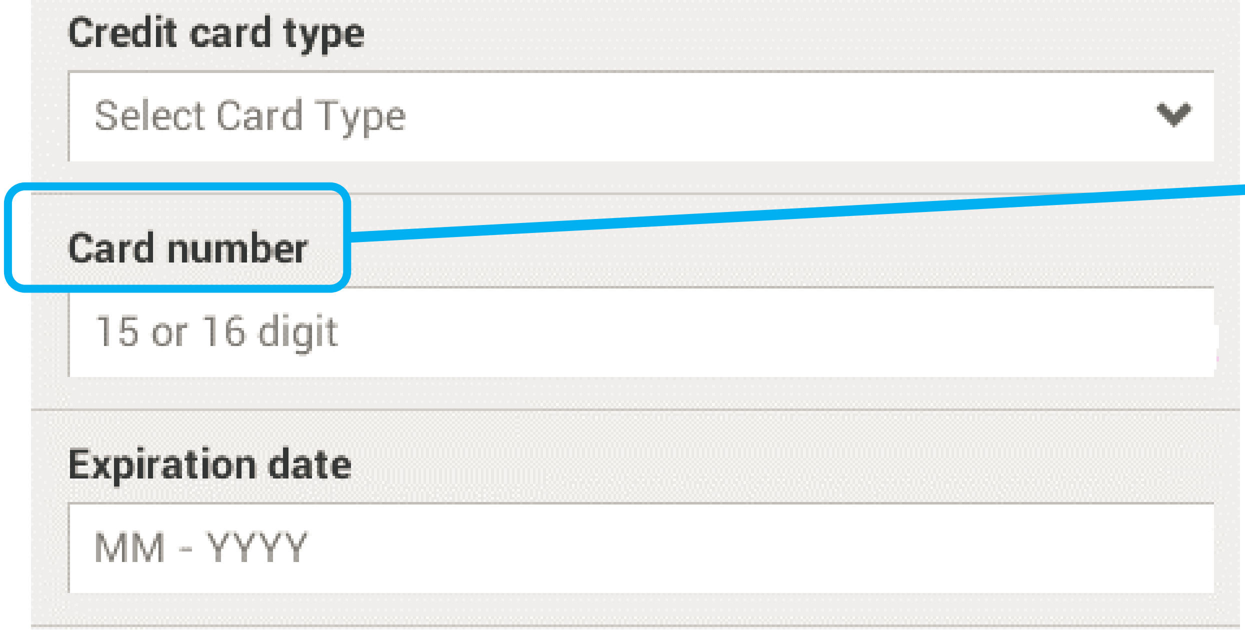
Card number

15 or 16 digit

Expiration date

MM - YYYY

Background - UI



Credit card type

Select Card Type

Card number

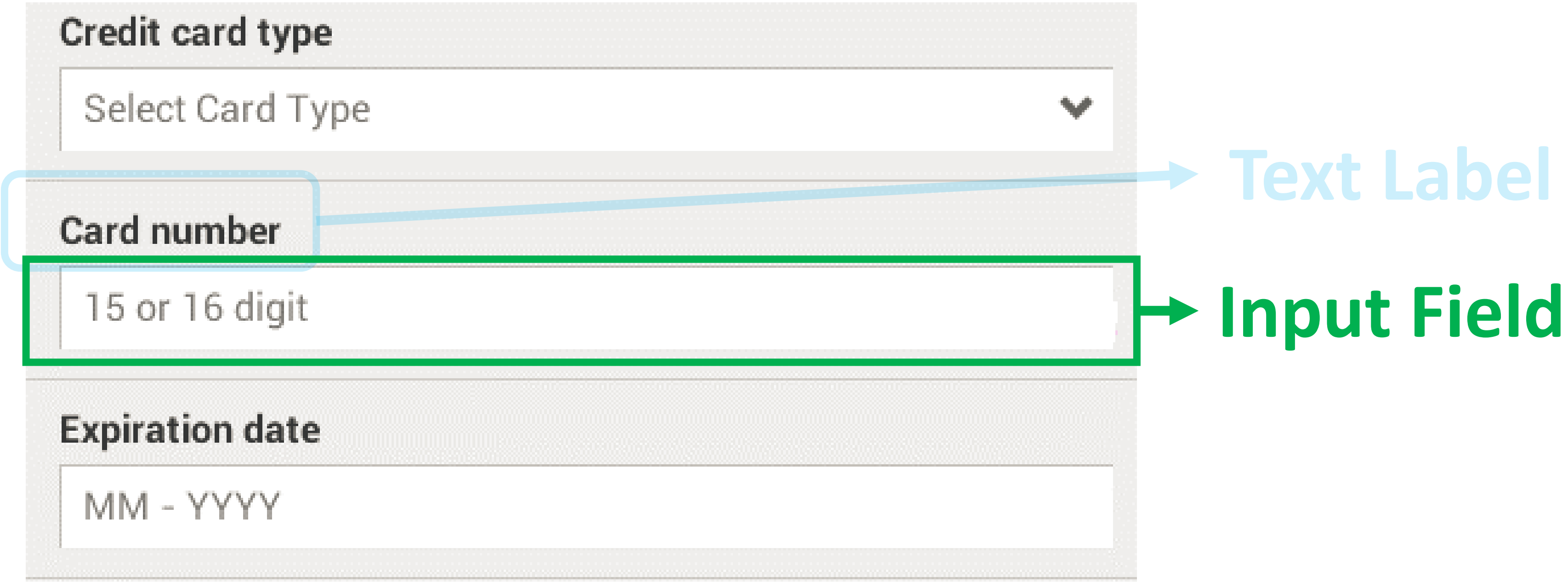
15 or 16 digit

Expiration date

MM - YYYY

Text Label

Background - UI



Credit card type

Select Card Type

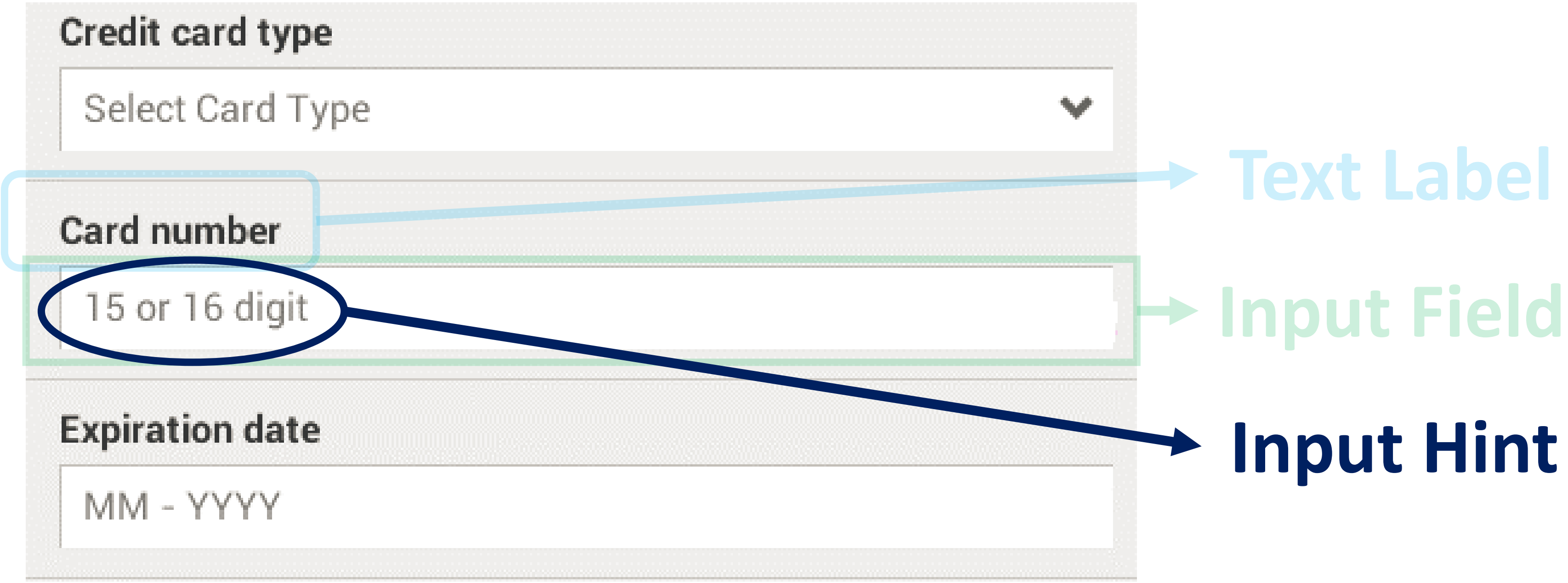
Card number

15 or 16 digit

Expiration date

MM - YYYY

Background - UI



Credit card type

Select Card Type

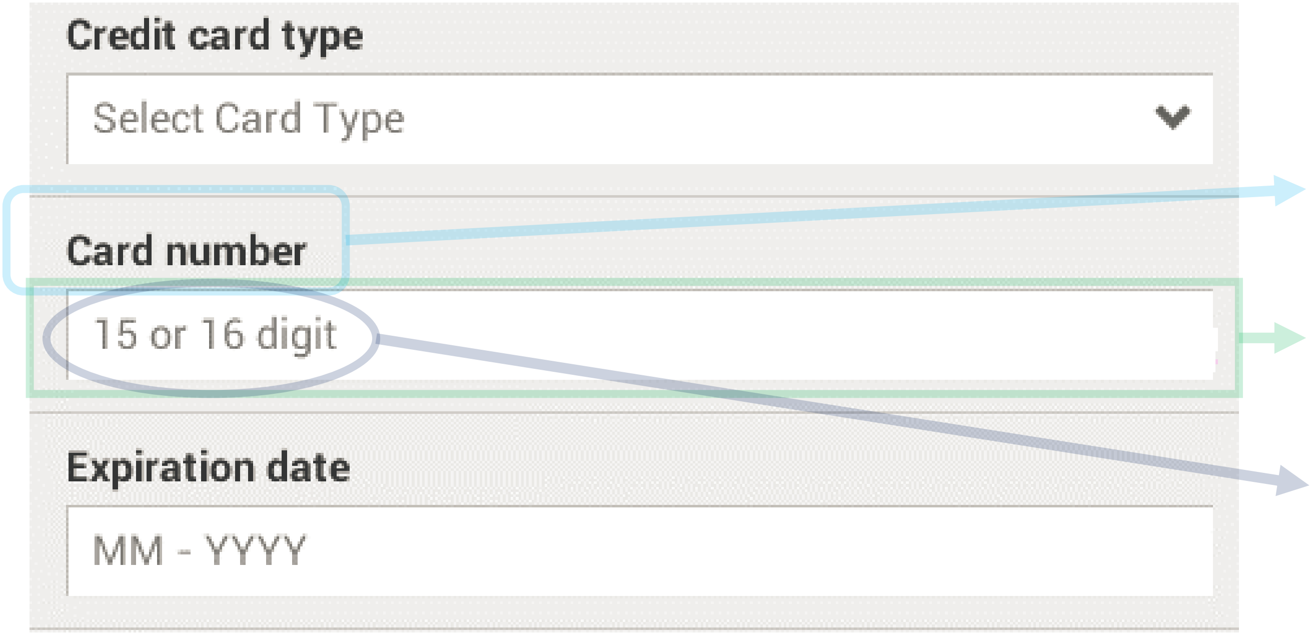
Card number

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

Input Field

Input Hint

Widget

Background – Layout File

- A piece in an Android layout example.

```
<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”/>
```

Identifier

Background – Layout File

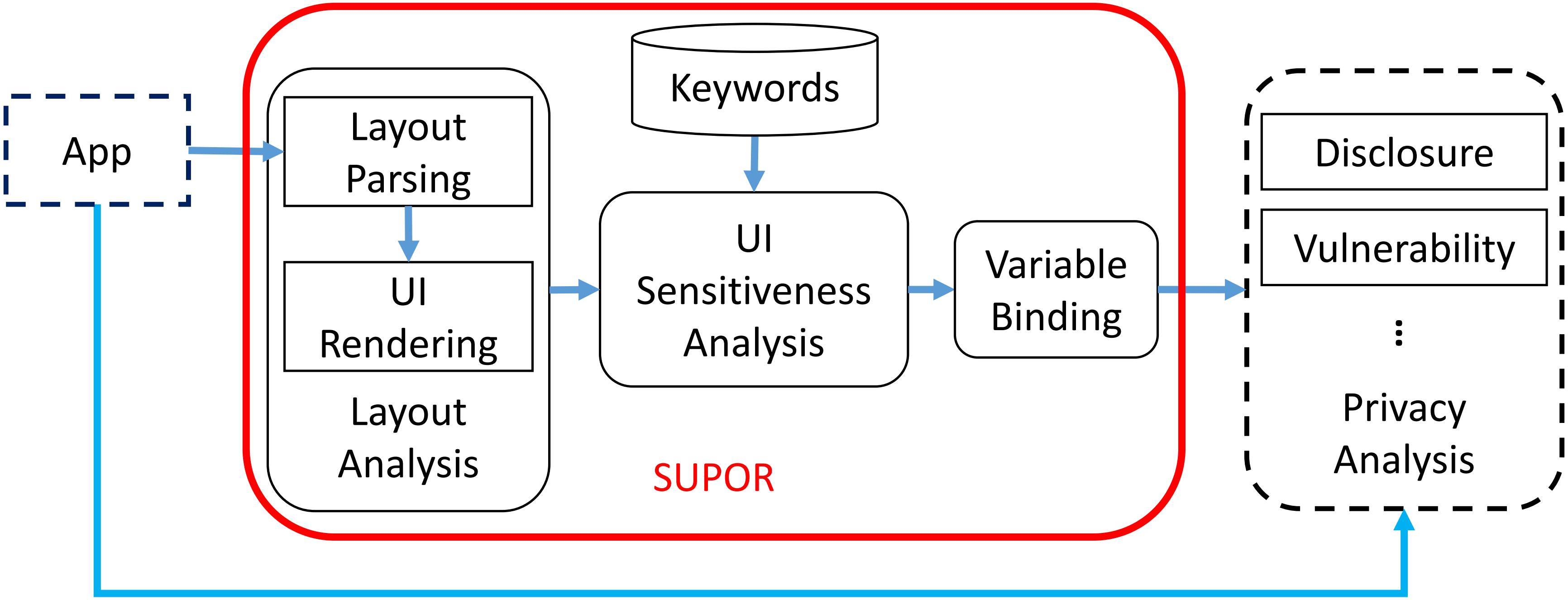
- A piece in an Android layout example.

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

Identifier

Interesting Attribute

Overview of SUPOR



Parsing Layout

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

Is Sensitive User Input Detection Needed?

Layout file

Parsing Layout

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

Layout file

Is Sensitive User Input Detection Needed?

layout contains input fields ✓

✗ layout doesn’t contain input fields

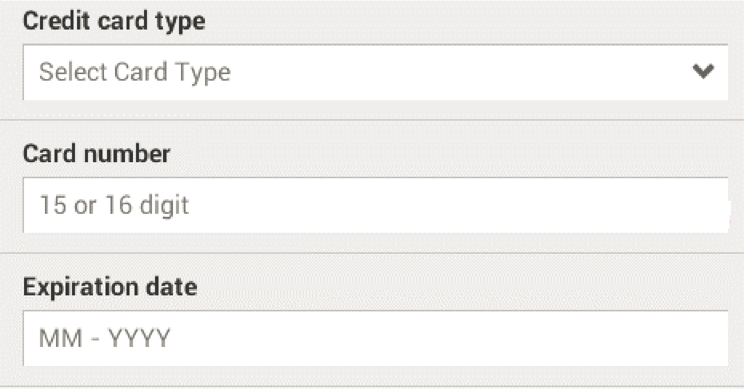
Rendering UI

- 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.

Layout file A


Credit card type

Select Card Type

Card number

15 or 16 digit

Expiration date

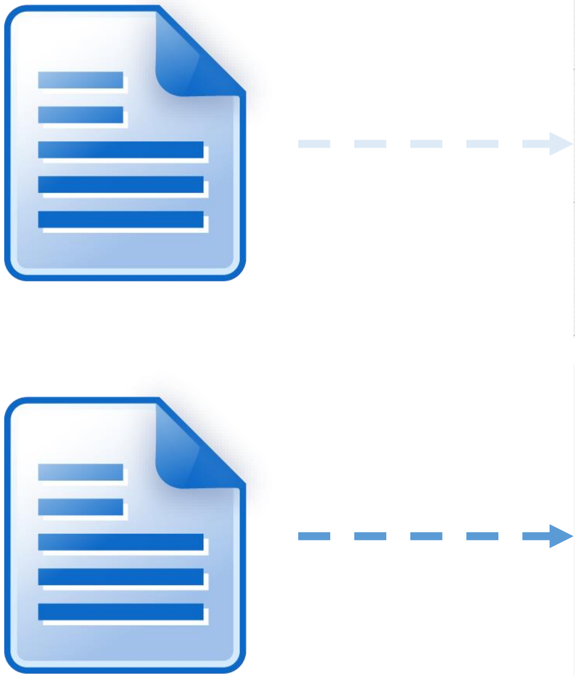
MM - YYYY

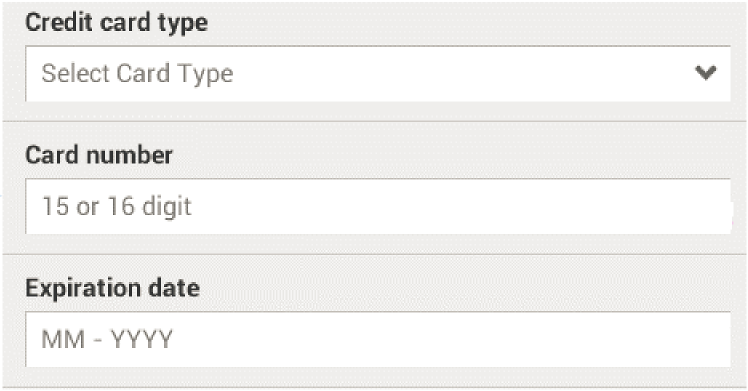
Rendering UI

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

Layout file A

Layout file B





Credit card type

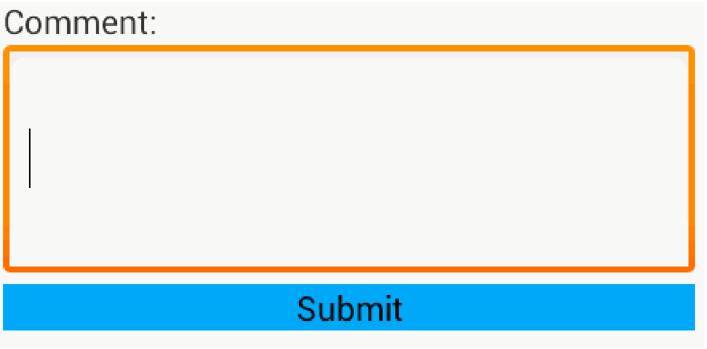
Select Card Type

Card number

15 or 16 digit

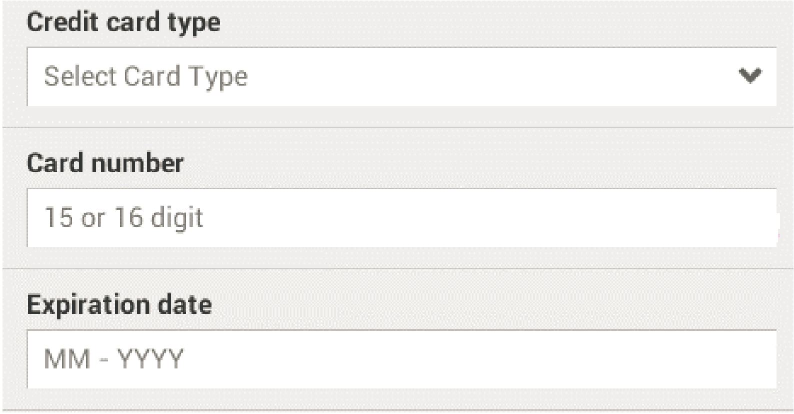
Expiration date

MM - YYYY



Comment:

Extracting Information



Credit card type

Select Card Type

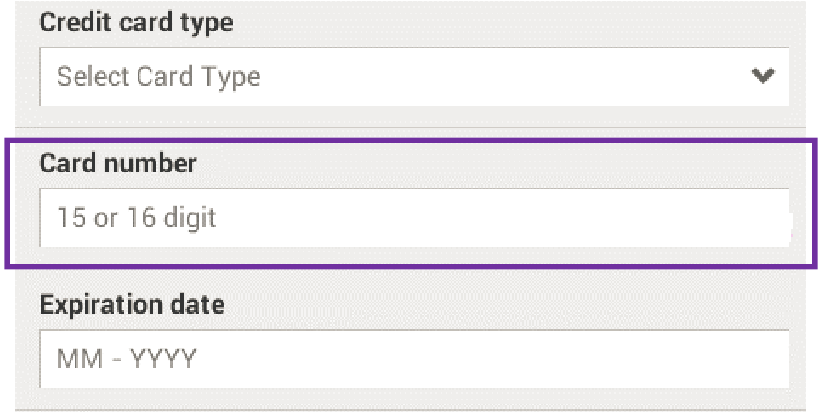
Card number

15 or 16 digit

Expiration date

MM - YYYY

Extracting Information



Credit card type

Select Card Type

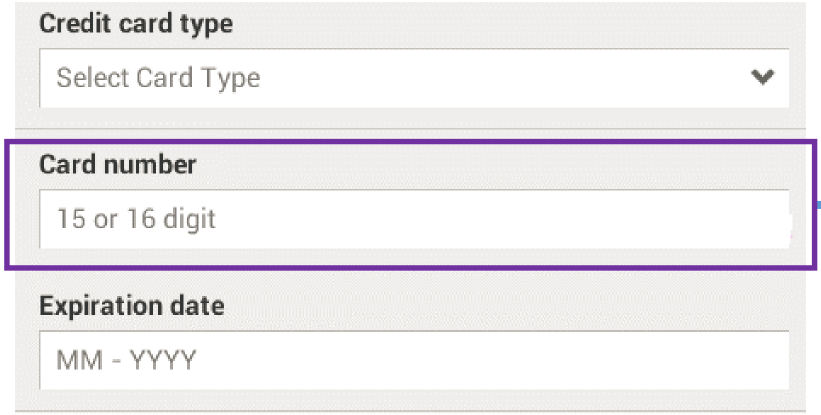
Card number

15 or 16 digit

Expiration date

MM - YYYY

Extracting Information



Credit card type

Select Card Type

Card number

15 or 16 digit

Expiration date

MM - YYYY

Collect information

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

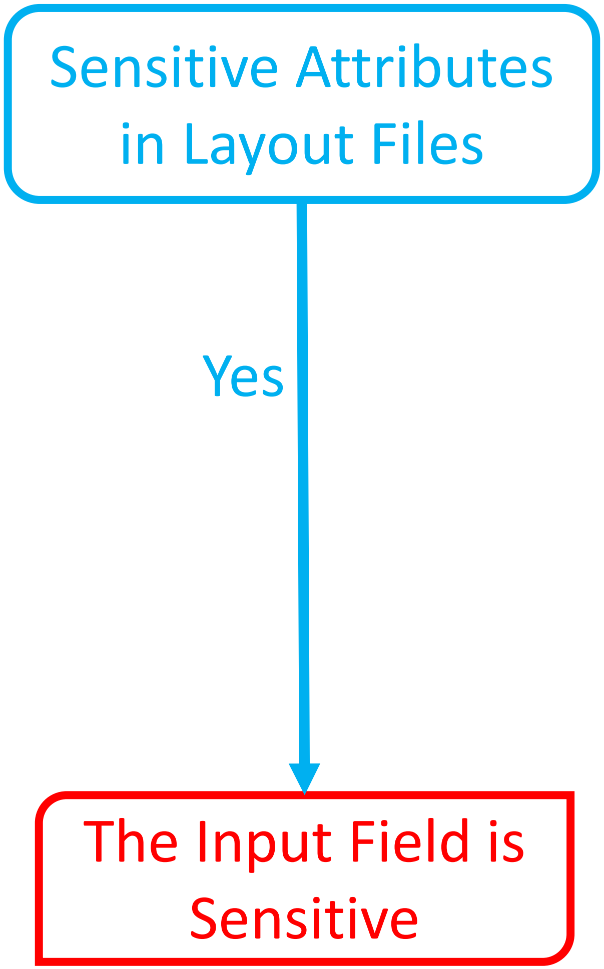
UI Sensitiveness Analysis

UI Sensitiveness Analysis

Sensitive Attributes
in Layout Files

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

UI Sensitiveness Analysis

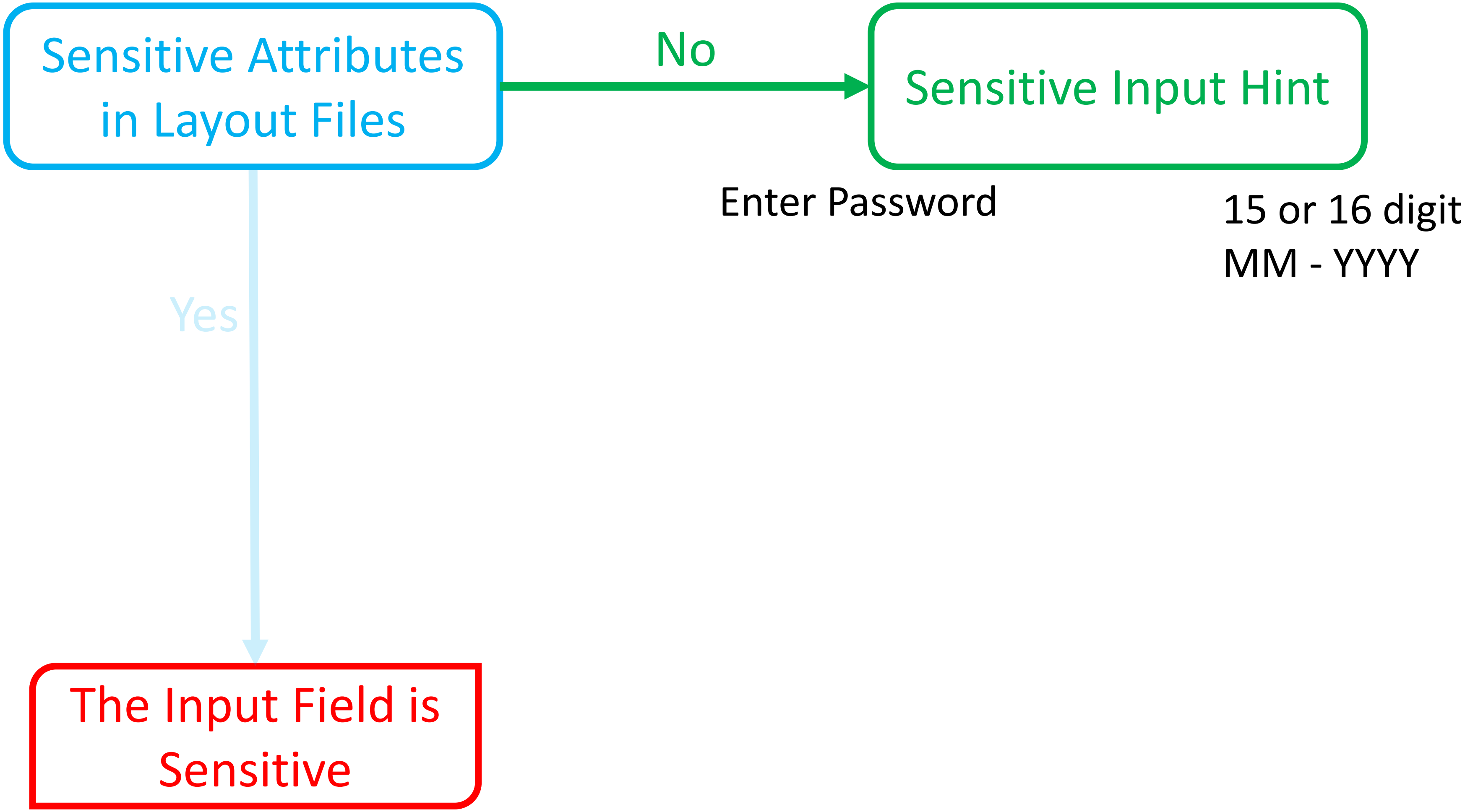


Sensitive Attributes in Layout Files

Yes

The Input Field is Sensitive

UI Sensitiveness Analysis



Sensitive Attributes in Layout Files

No

Sensitive Input Hint

Enter Password

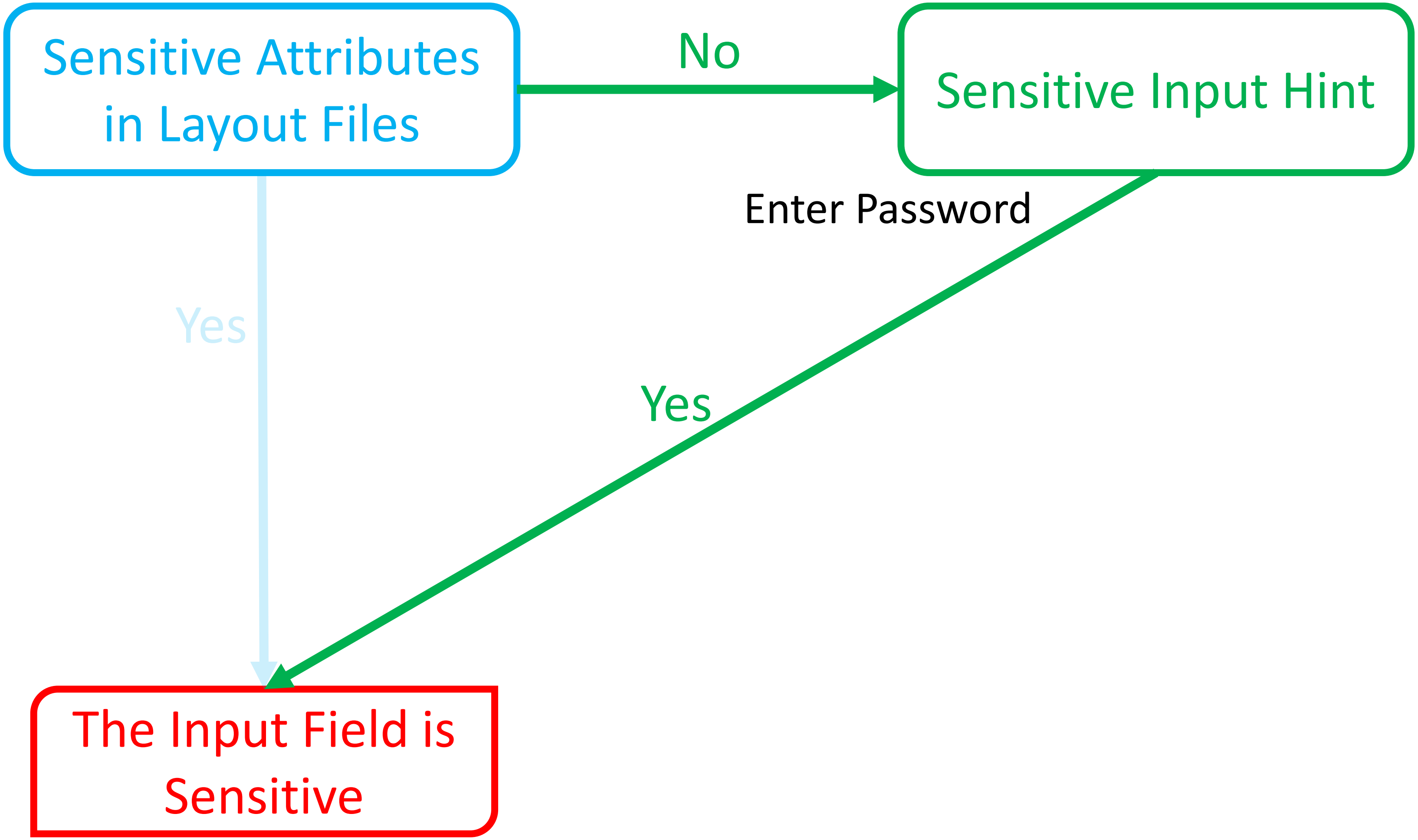
15 or 16 digit

MM - YYYY

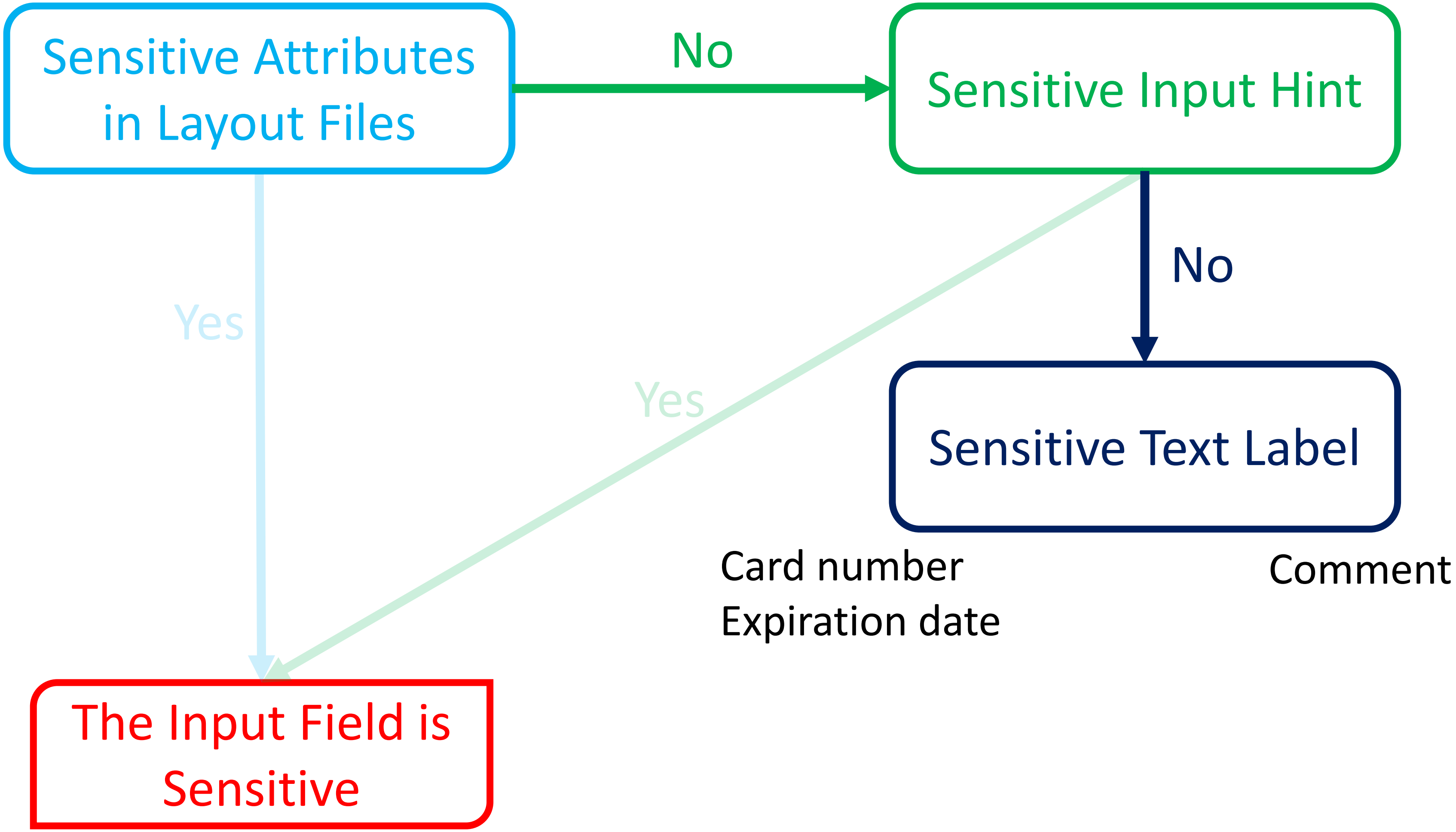
Yes

The Input Field is Sensitive

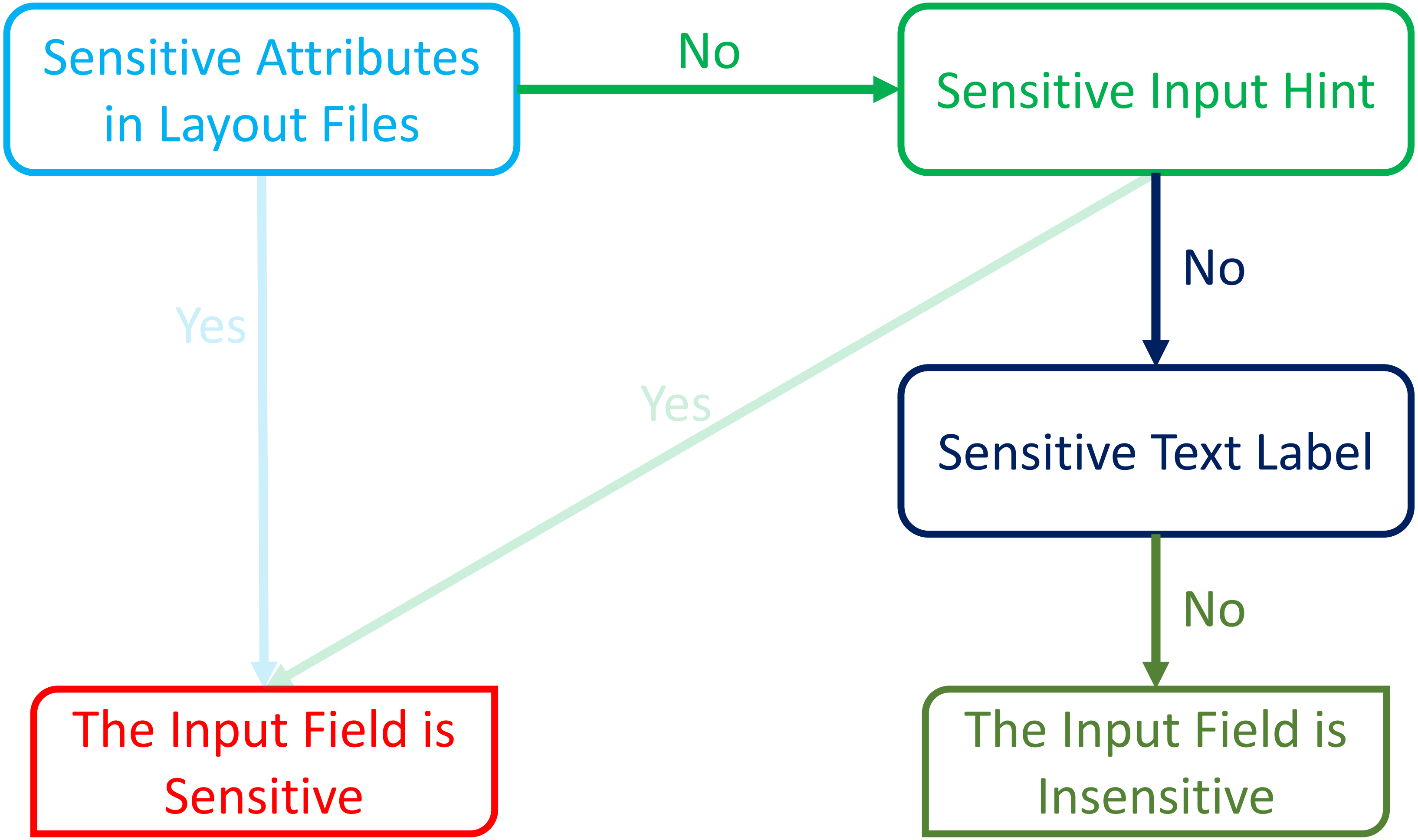
UI Sensitiveness Analysis


Sensitive Attributes in Layout Files
No
Sensitive Input Hint
Enter Password
Yes
Yes
The Input Field is Sensitive

UI Sensitiveness Analysis



UI Sensitiveness Analysis



Sensitive Attributes in Layout Files

No

Sensitive Input Hint

Yes

Yes

No

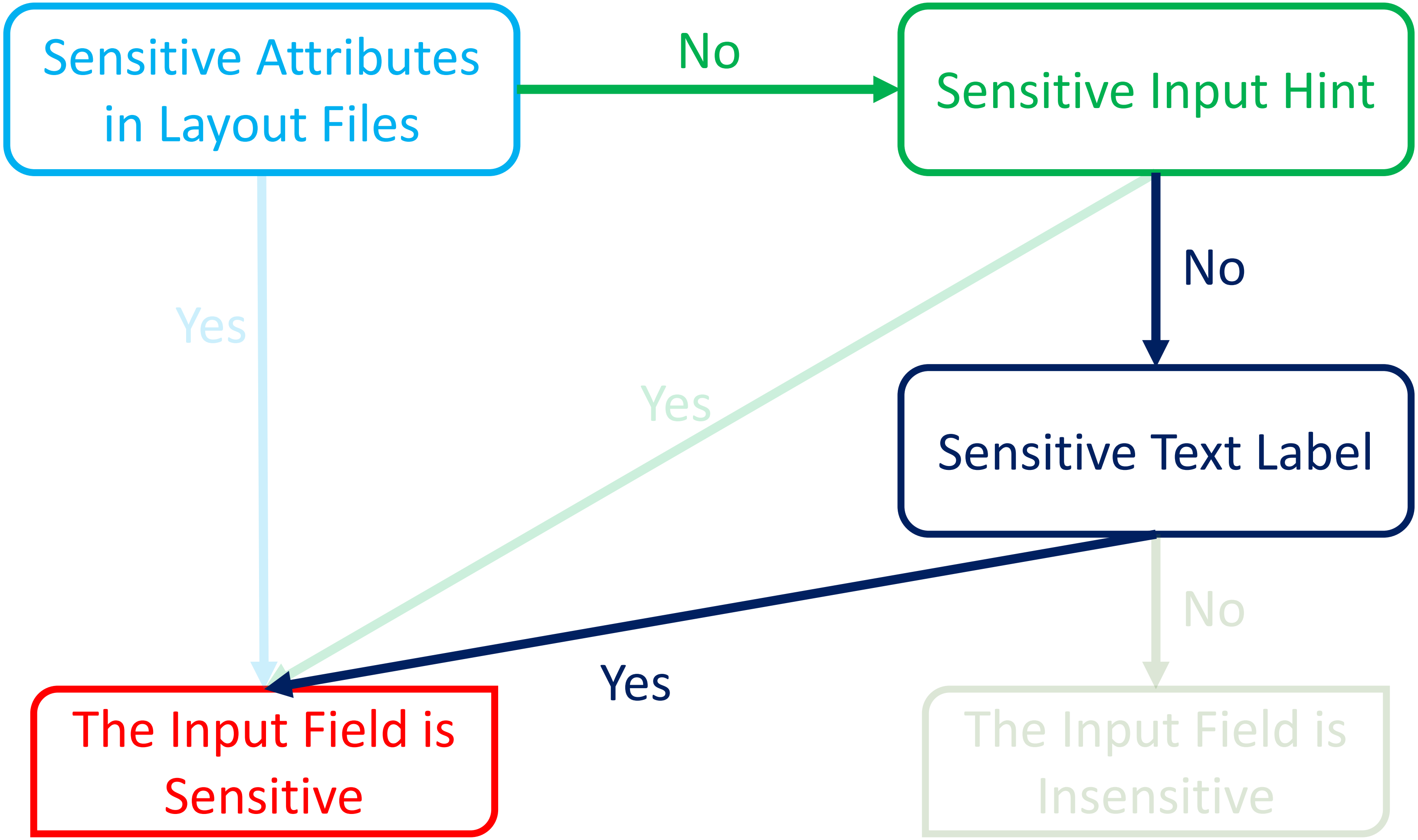
Sensitive Text Label

No

The Input Field is Sensitive

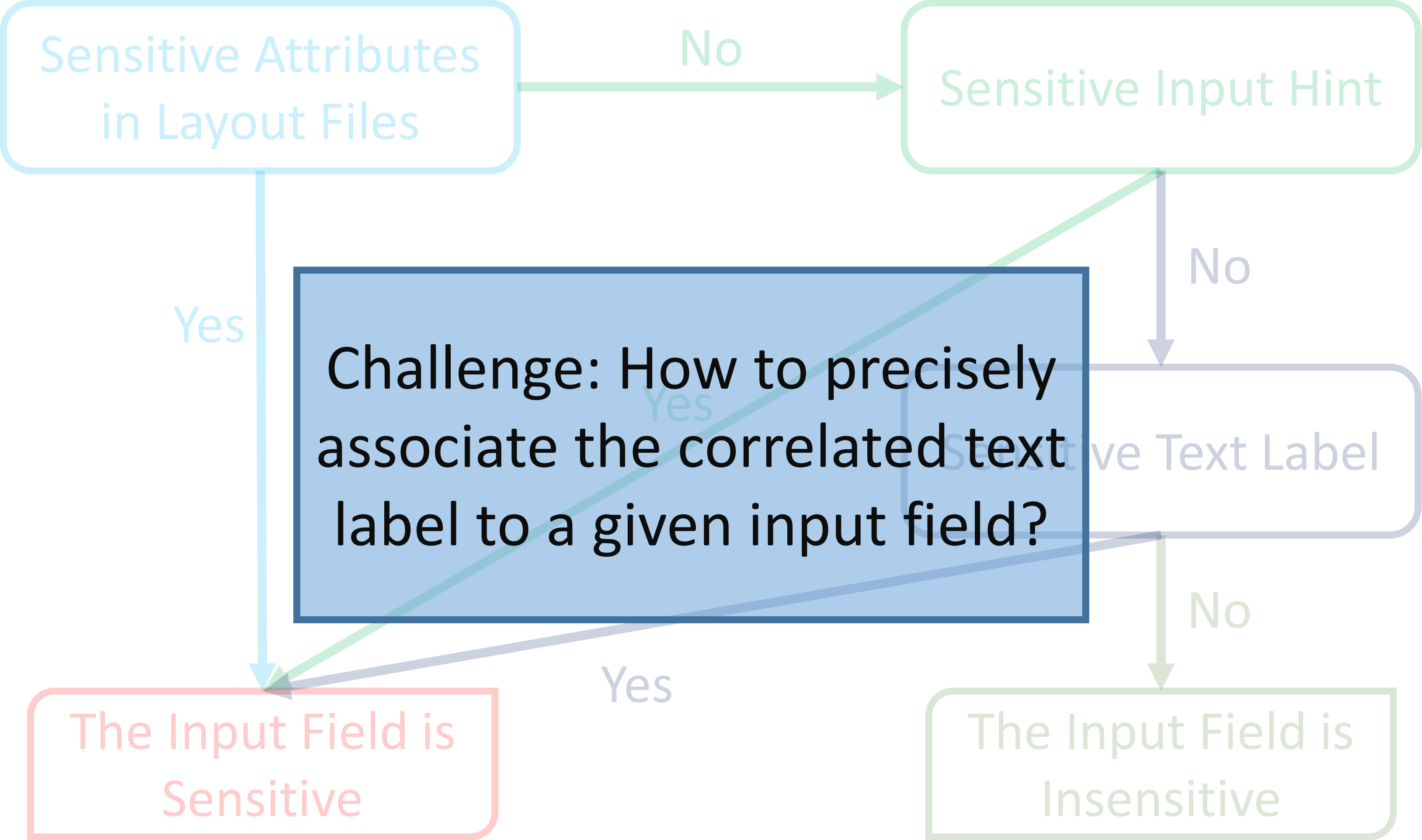
The Input Field is Insensitive

UI Sensitiveness Analysis



Sensitive Attributes in Layout Files
No
Sensitive Input Hint
No
Yes
Yes
Sensitive Text Label
No
Yes
The Input Field is Sensitive
The Input Field is Insensitive

UI Sensitiveness Analysis


Sensitive Attributes in Layout Files
No
Sensitive Input Hint
Yes
No
Yes
Sensitive Text Label
No
Yes
The Input Field is Sensitive
The Input Field is Insensitive

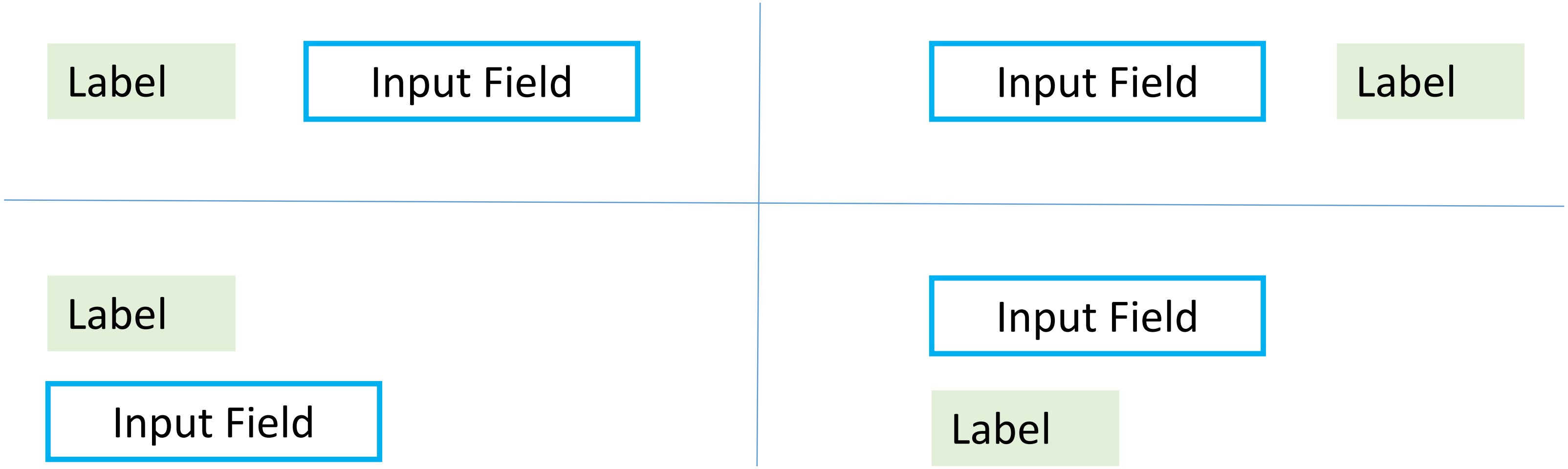
Challenge: How to precisely associate the correlated text label to a given input field?

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

4	2	8
0.8	Input Field	9
8	9	10

Associating Labels (2)

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

4	2	8
0.8	Input Field	9
8	9	10

Associating Labels (2)

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

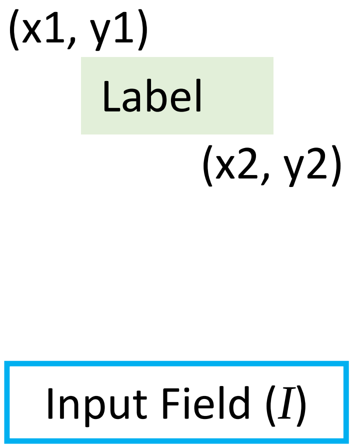
4	2	8
0.8	Input Field	9
8	9	10

Associating Labels (3)

- Geometry-based correlation score computation

Associating Labels (3)

- Geometry-based correlation score computation



(x1, y1)

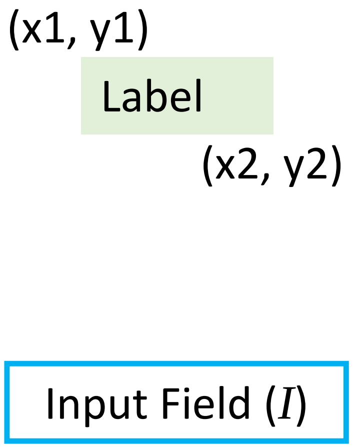
Label

(x2, y2)

Input Field (I)

Associating Labels (3)

- Geometry-based correlation score computation



(x1, y1)

Label

(x2, y2)

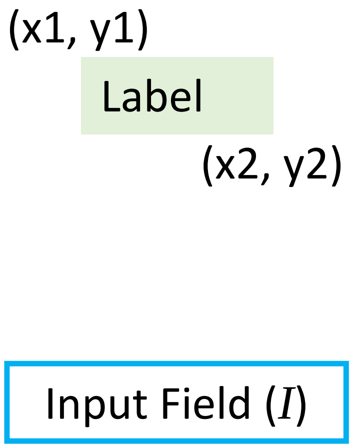
Input Field (*I*)

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

- *distance(I, x, y) * posWeight(I, x, y)*

Associating Labels (3)

- Geometry-based correlation score computation



➢ For each pixel (x,y) in a text label
  - $distance(I, x, y) * posWeight(I, x, y)$

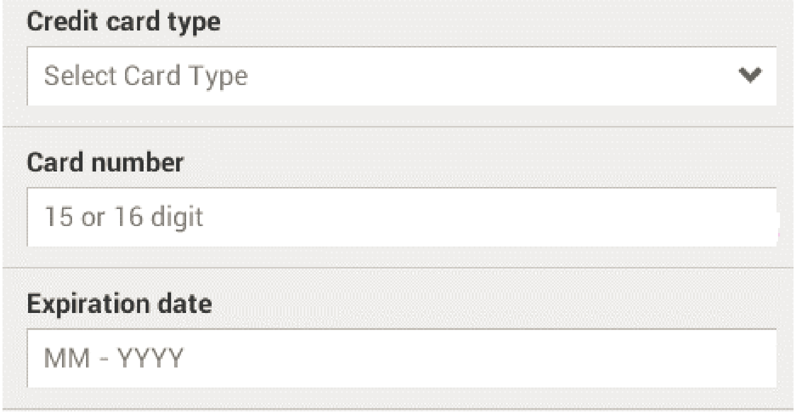
➢ Average the correlation score for the text label

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



Credit card type

Select Card Type

Card number

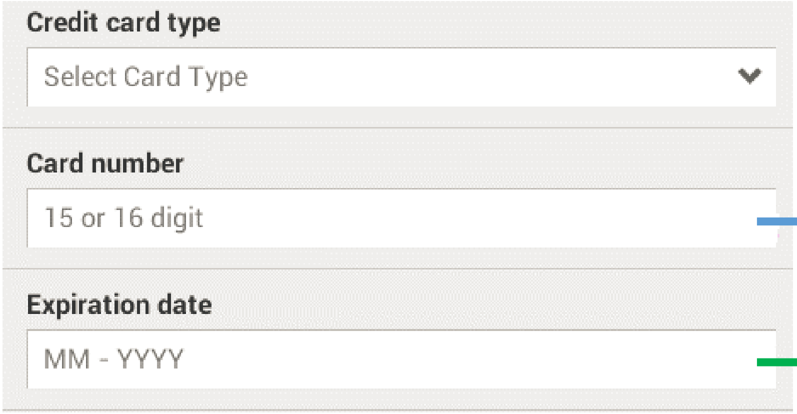
15 or 16 digit

Expiration date

MM - YYYY

Associating Labels (4)

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


Credit card type

Select Card Type

Card number

15 or 16 digit

Expiration date

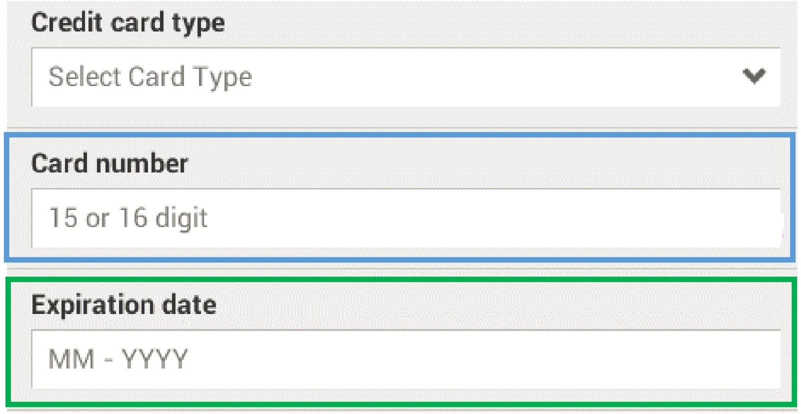
MM - YYYY

Correlation scores

Label	**Number Field**	**Date Field**
Credit card type	265.57	456.42
Card number	**76.47**	271.23
Expiration date	205.29	**75.40**

Associating Labels (4)

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



Credit card type

Select Card Type

Card number

15 or 16 digit

Expiration date

MM - YYYY

Correlation scores

Label	**Number Field**	**Date Field**
Credit card type	265.57	456.42
Card number	**76.47**	271.23
Expiration date	205.29	**75.40**

Determining Sensitiveness (1)

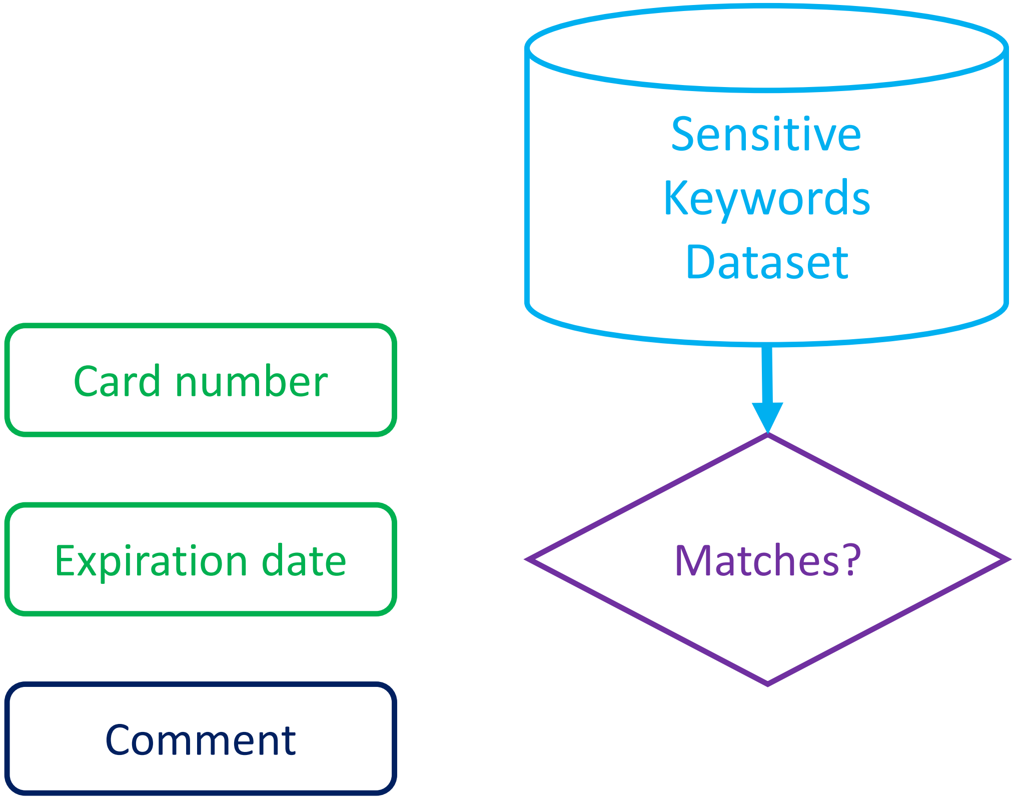
Card number

Expiration date

Comment

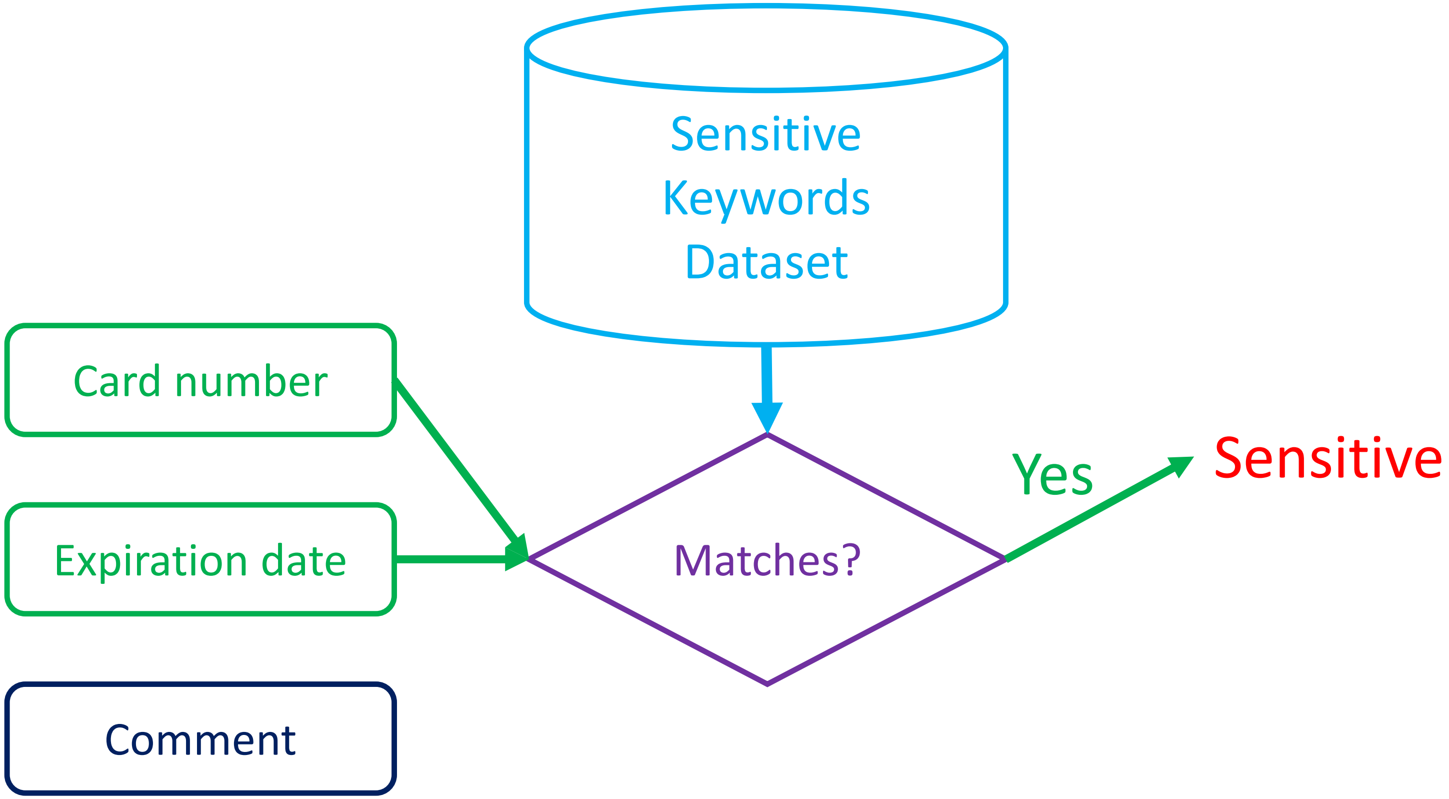
Determining Sensitiveness (1)

- Keyword matching approach



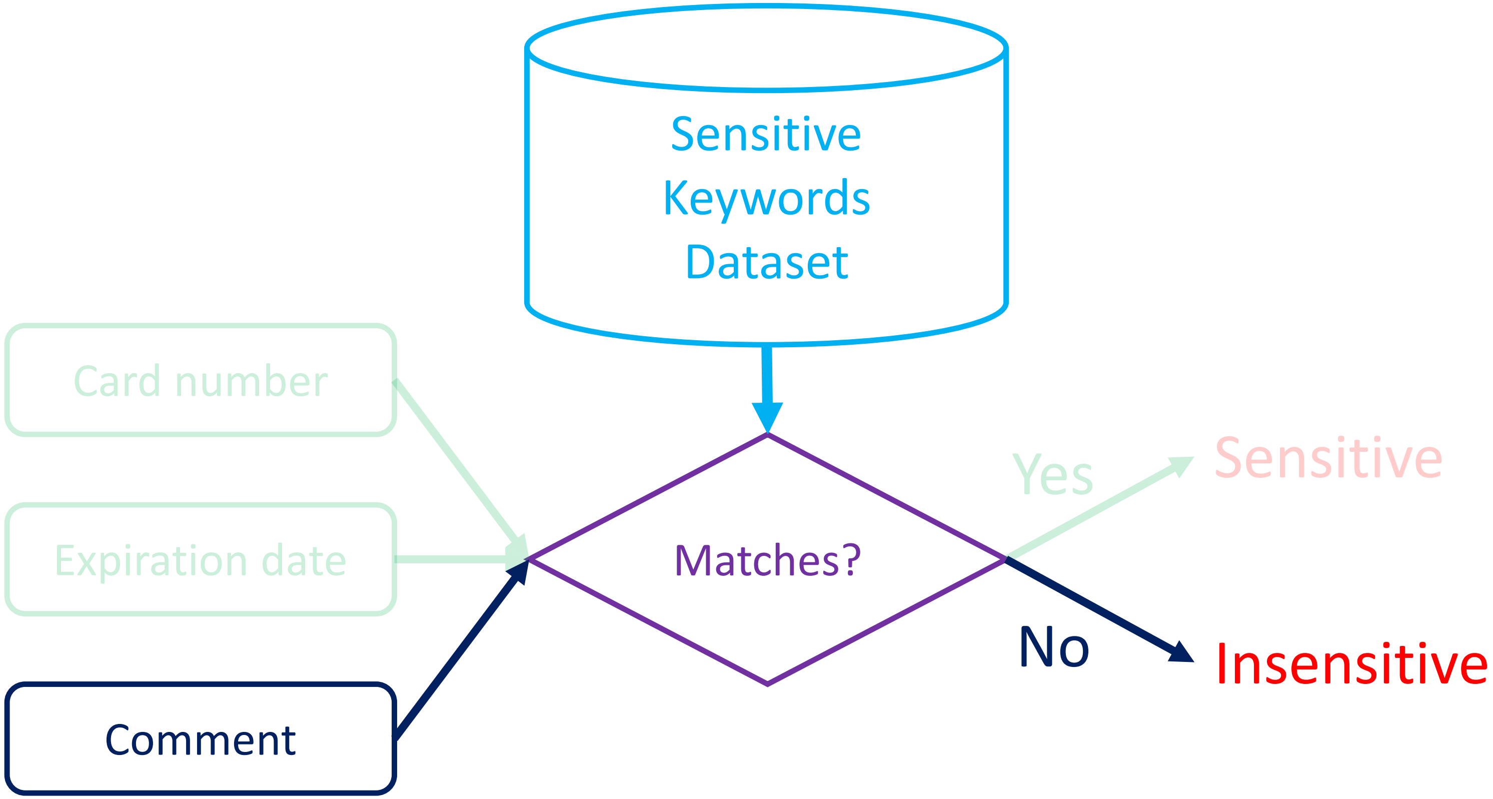
Determining Sensitiveness (1)

- Keyword matching approach



Determining Sensitiveness (1)

- Keyword matching approach

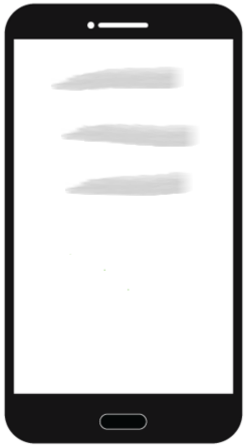


Determining Sensitiveness (2)

- Why is keyword matching approach effective?

Determining Sensitiveness (2)

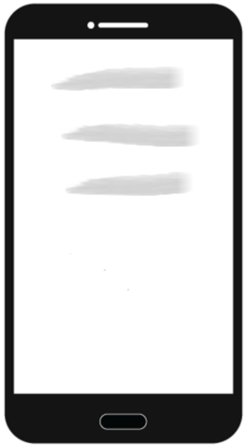
- Why is keyword matching approach effective?



- **Small** screen and **short** phrases or sentences

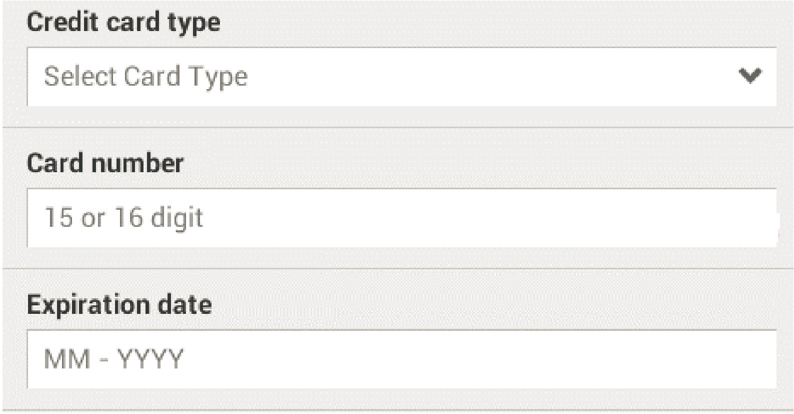
Determining Sensitiveness (2)

- Why is keyword matching approach effective?



- **Small** screen and **short** phrases or sentences
- We only analyze the **most relevant** text label

Binding Variables (1)



Credit card type

Select Card Type

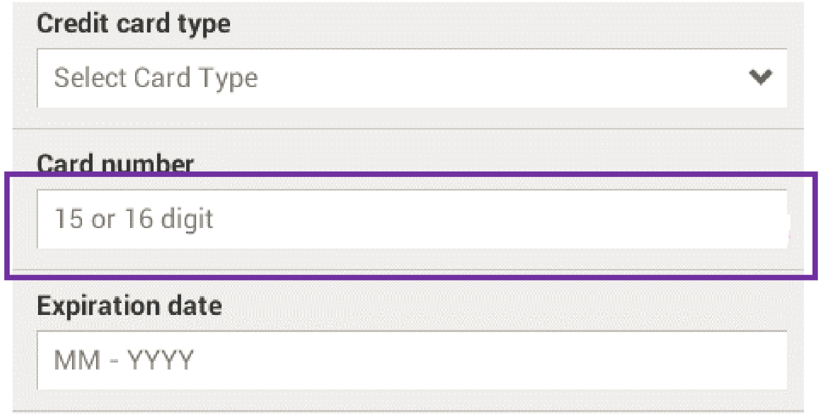
Card number

15 or 16 digit

Expiration date

MM - YYYY

Binding Variables (1)



Credit card type

Select Card Type

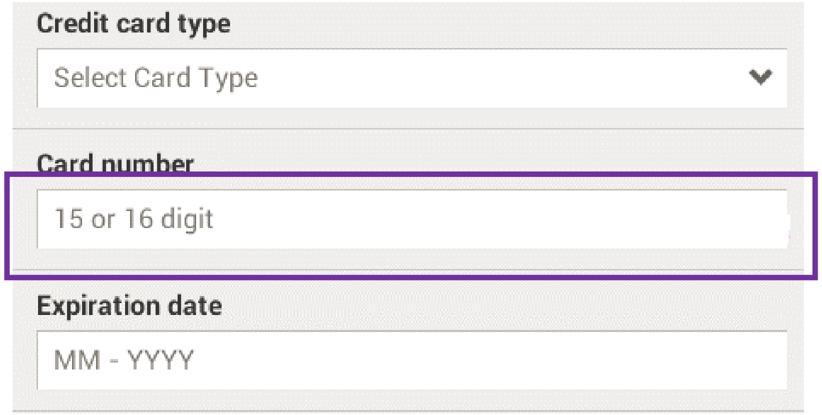
Card number

15 or 16 digit

Expiration date

MM - YYYY

Binding Variables (1)



Credit card type

Select Card Type

Card number

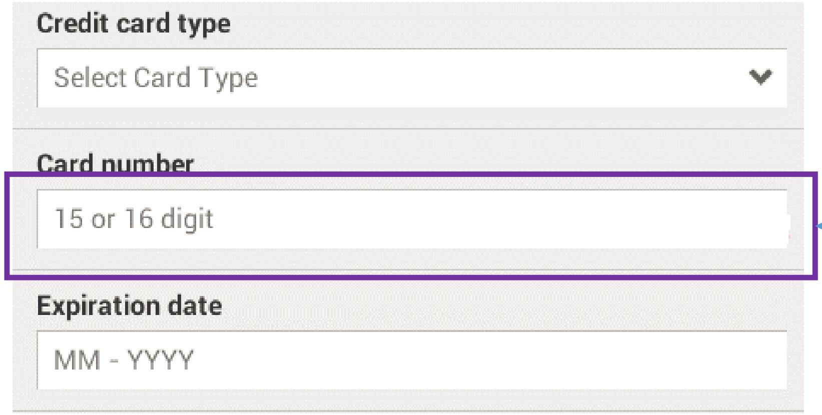
15 or 16 digit

Expiration date

MM - YYYY

```
Widget txtCN = findViewById(X);
Data cnum = txtCN.getText();
// use of “cnum”
```

Binding Variables (1)



Credit card type

Select Card Type

Card number

15 or 16 digit

Expiration date

MM - YYYY

Identifier: X

```
Widget txtCN = findViewById(X);
Data cnum = txtCN.getText();
// use of "cnum"
```

Binding Variables (2)

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

```
<TextView android:text= =“Card Number‟ />
<EditText android:id=“@+id/input1‟ … />
…
```

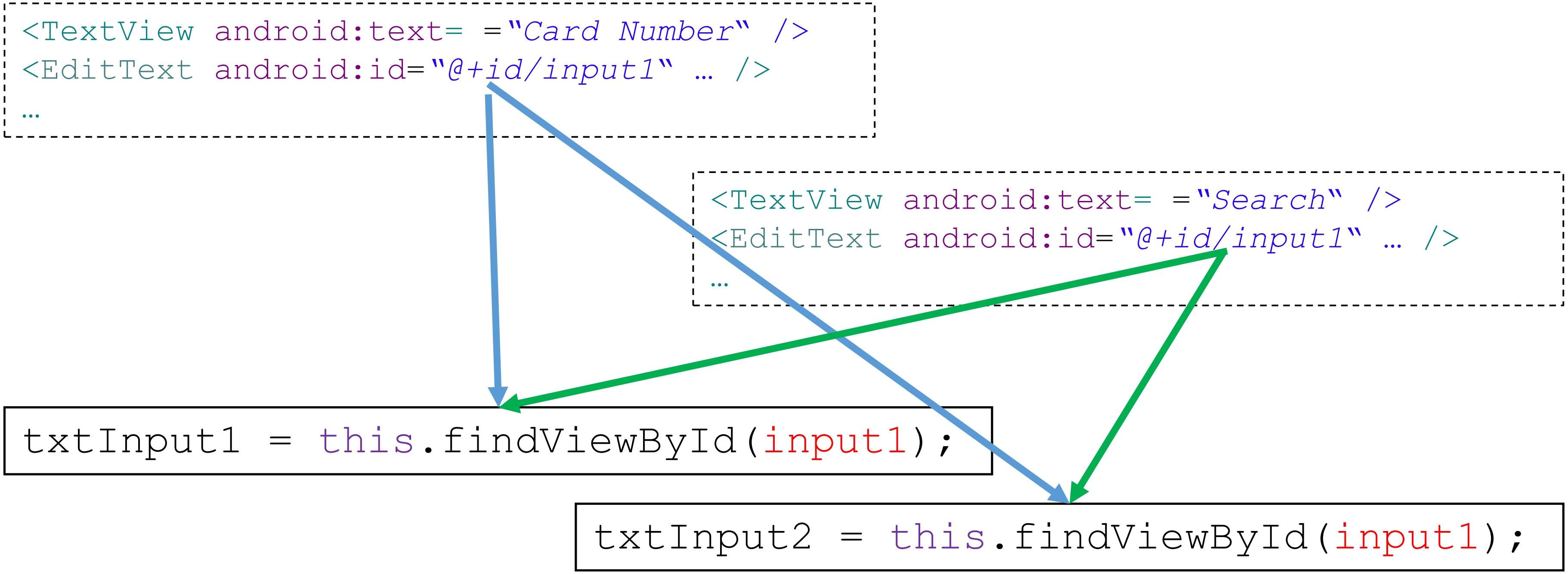
```
<TextView android:text= =“Search‟ />
<EditText android:id=“@+id/input1‟ … />
…
```

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

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

Binding Variables (2)

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



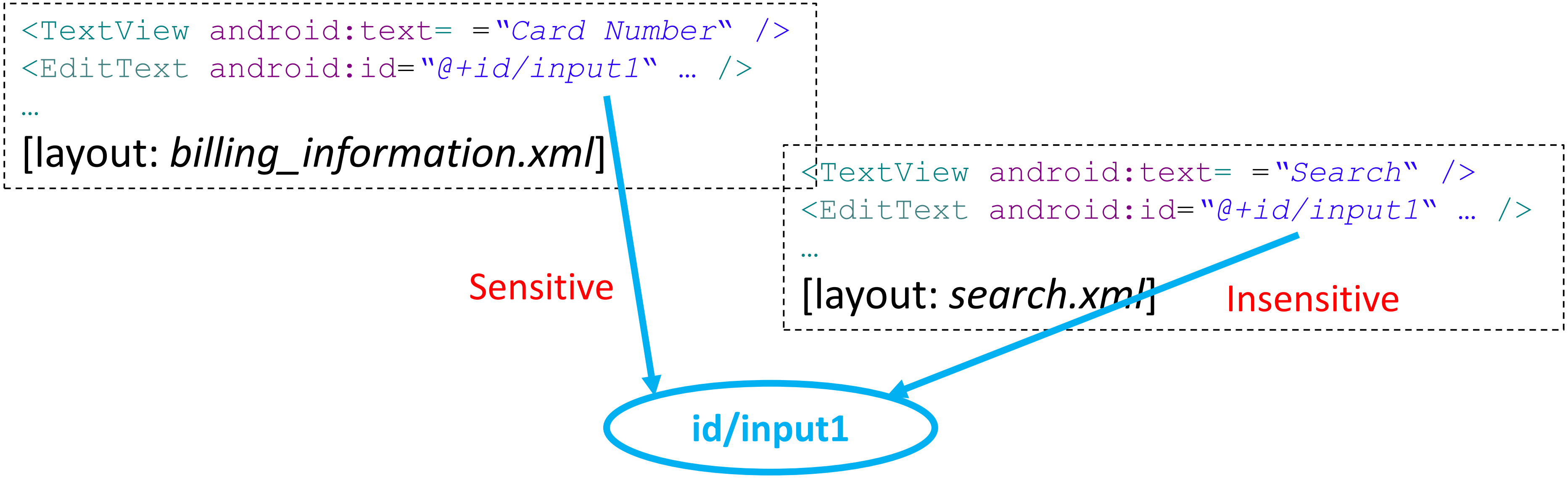
```
<TextView android:text= =“Card Number‟ />
<EditText android:id=“@+id/input1‟ … />
…
```

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

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

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

Binding Variables (3)



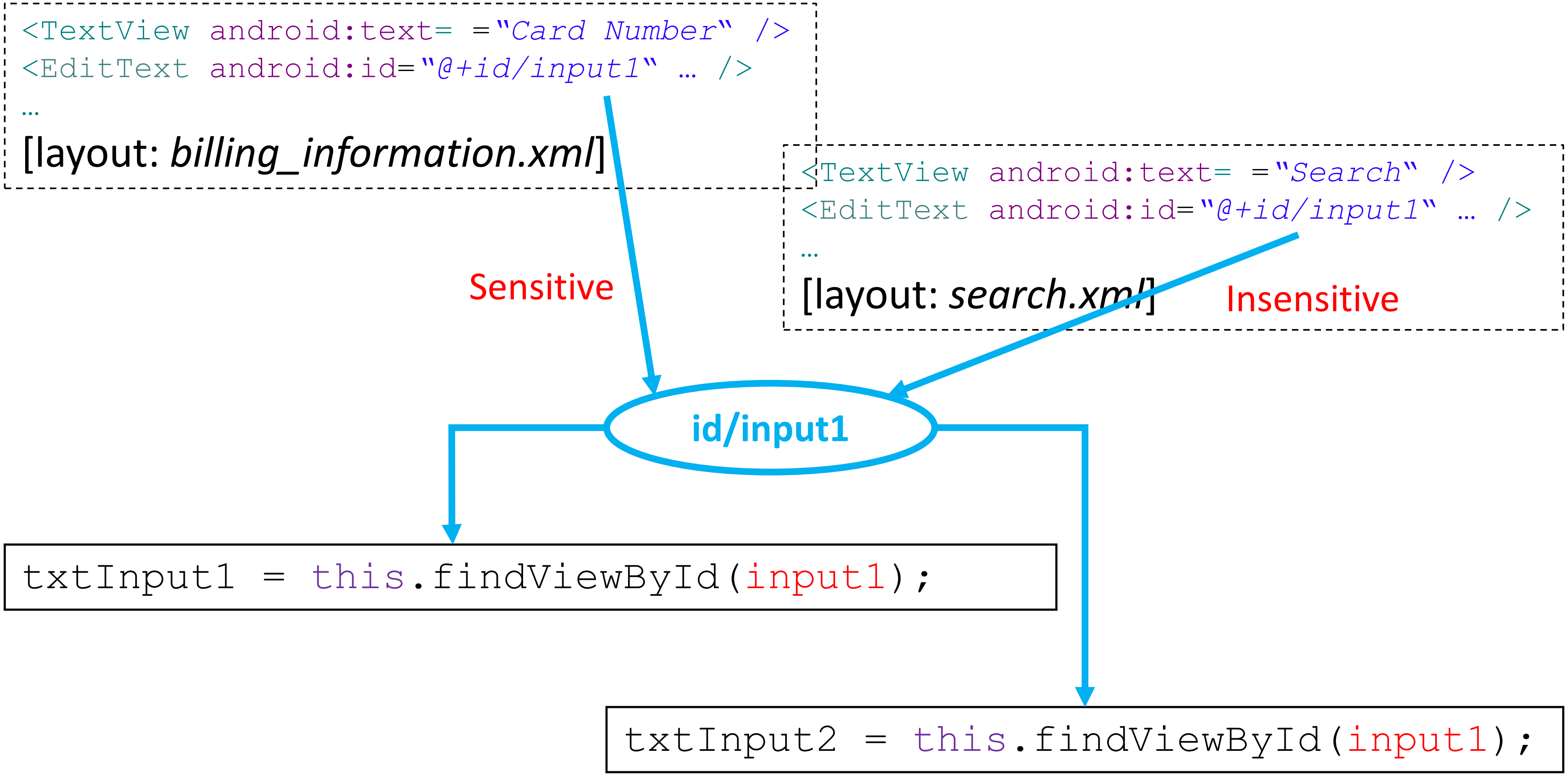
```
<TextView android:text= ="Card Number" />
<EditText android:id="@+id/input1" … />
…
```

[layout: *billing_information.xml*]

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

[layout: *search.xml*]

Binding Variables (3)



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

[layout: *billing_information.xml*]

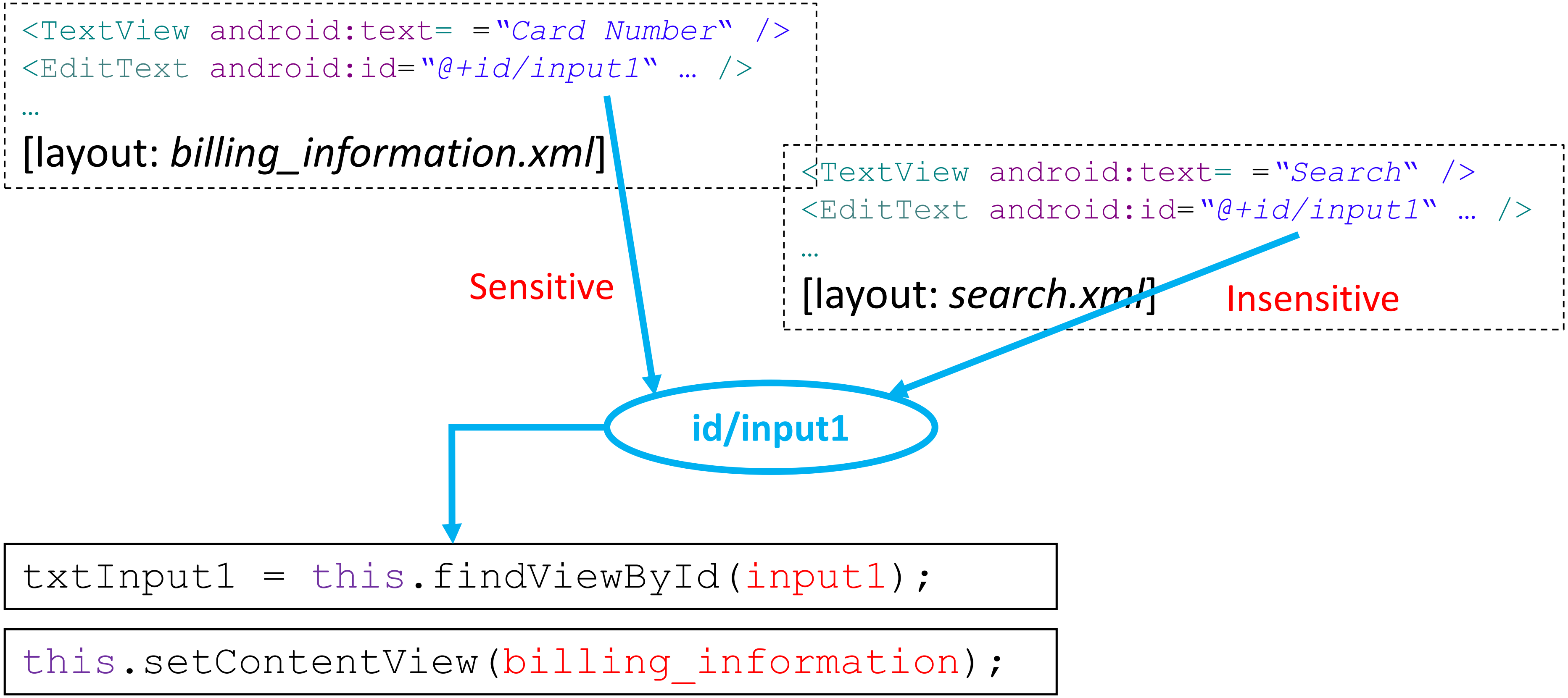
```
<TextView android:text= ="Search" />
<EditText android:id="@+id/input1" … />
…
```

[layout: *search.xml*]

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

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

Binding Variables (3)



```
<TextView android:text= =“Card Number‟ />
<EditText android:id=“@+id/input1‟ … />
…
```

[layout: *billing_information.xml*]

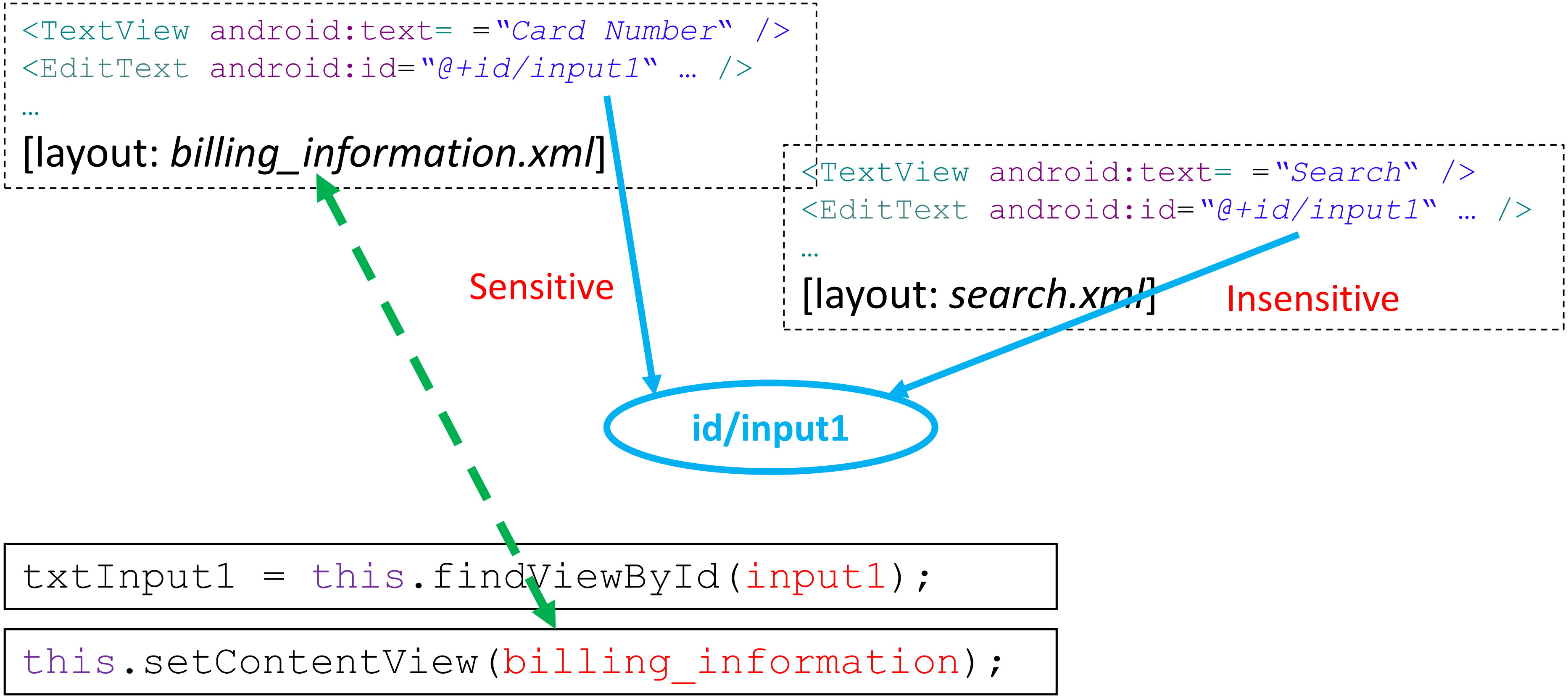
```
<TextView android:text= =“Search‟ />
<EditText android:id=“@+id/input1‟ … />
…
```

[layout: *search.xml*]

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

```
this.setContentView(billing_information);
```

Binding Variables (3)



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

[layout: *billing_information.xml*]

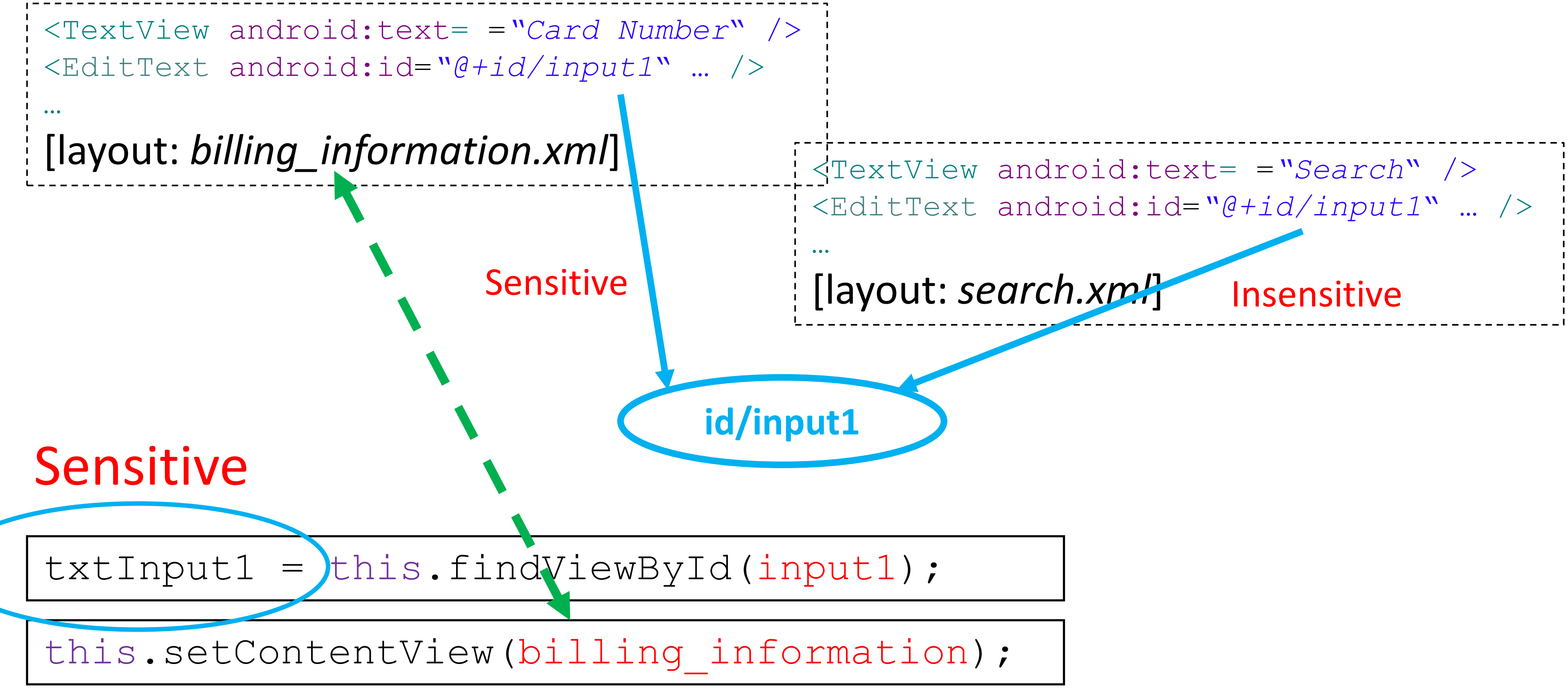
```
<TextView android:text= ="Search" />
<EditText android:id="@+id/input1" … />
…
```

[layout: *search.xml*]

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

```
this.setContentView(billing_information);
```

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

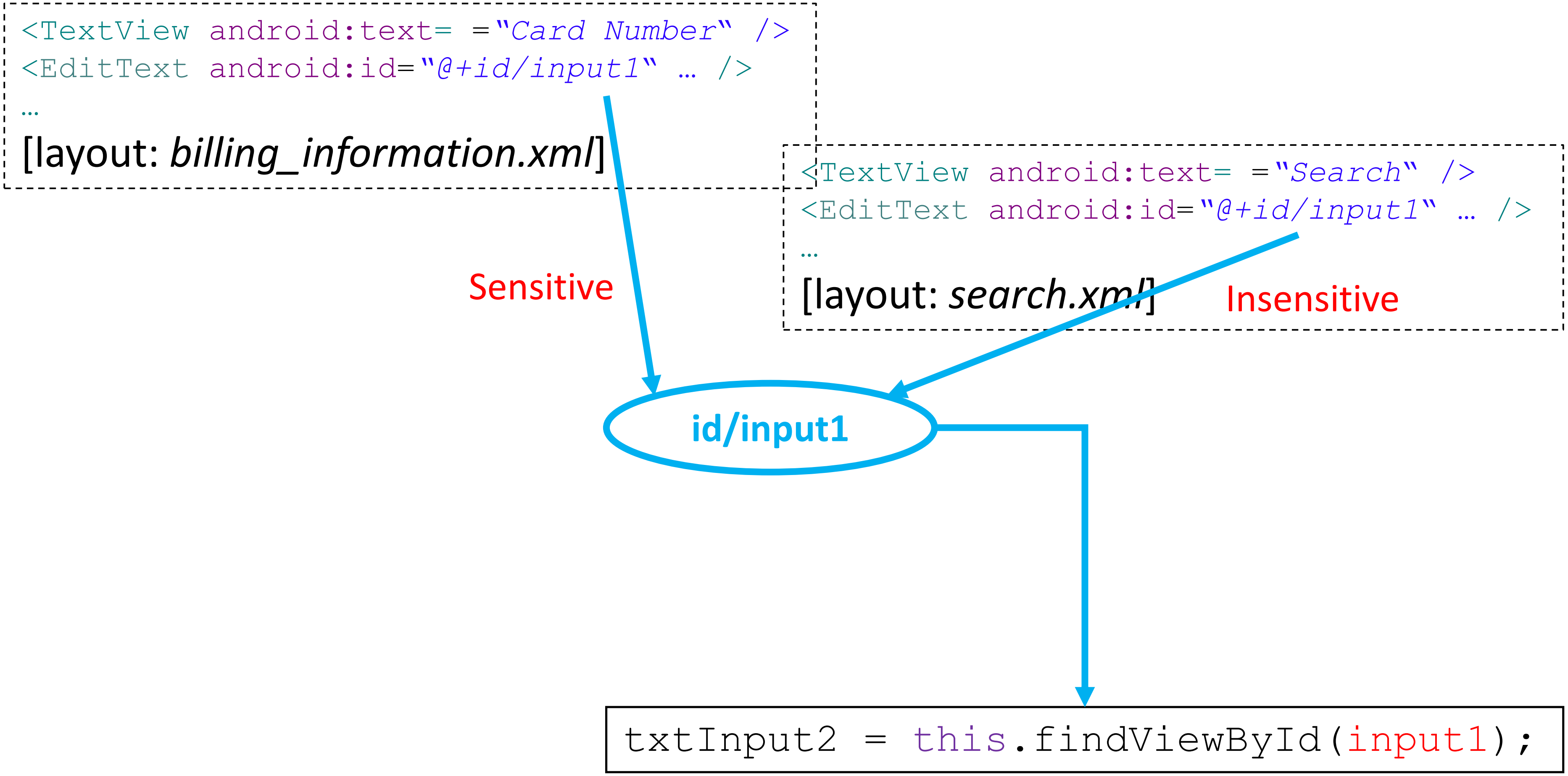
id/input1

Sensitive

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

```
this.setContentView(billing_information);
```

Binding Variables (3)



```
<TextView android:text= =“Card Number‟ />
<EditText android:id=“@+id/input1‟ … />
…
```

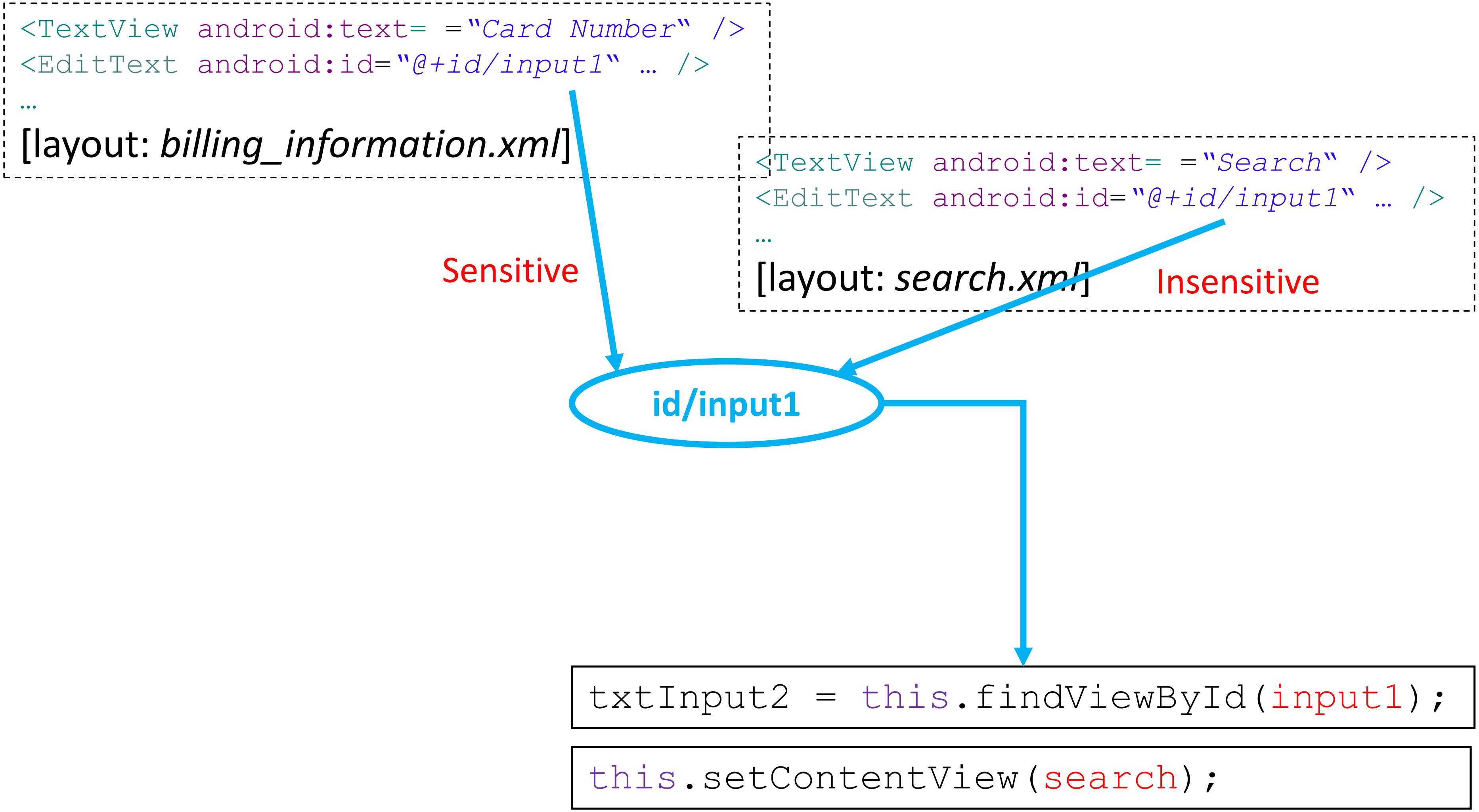
[layout: *billing_information.xml*]

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

[layout: *search.xml*]

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

Binding Variables (3)



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

[layout: *billing_information.xml*]

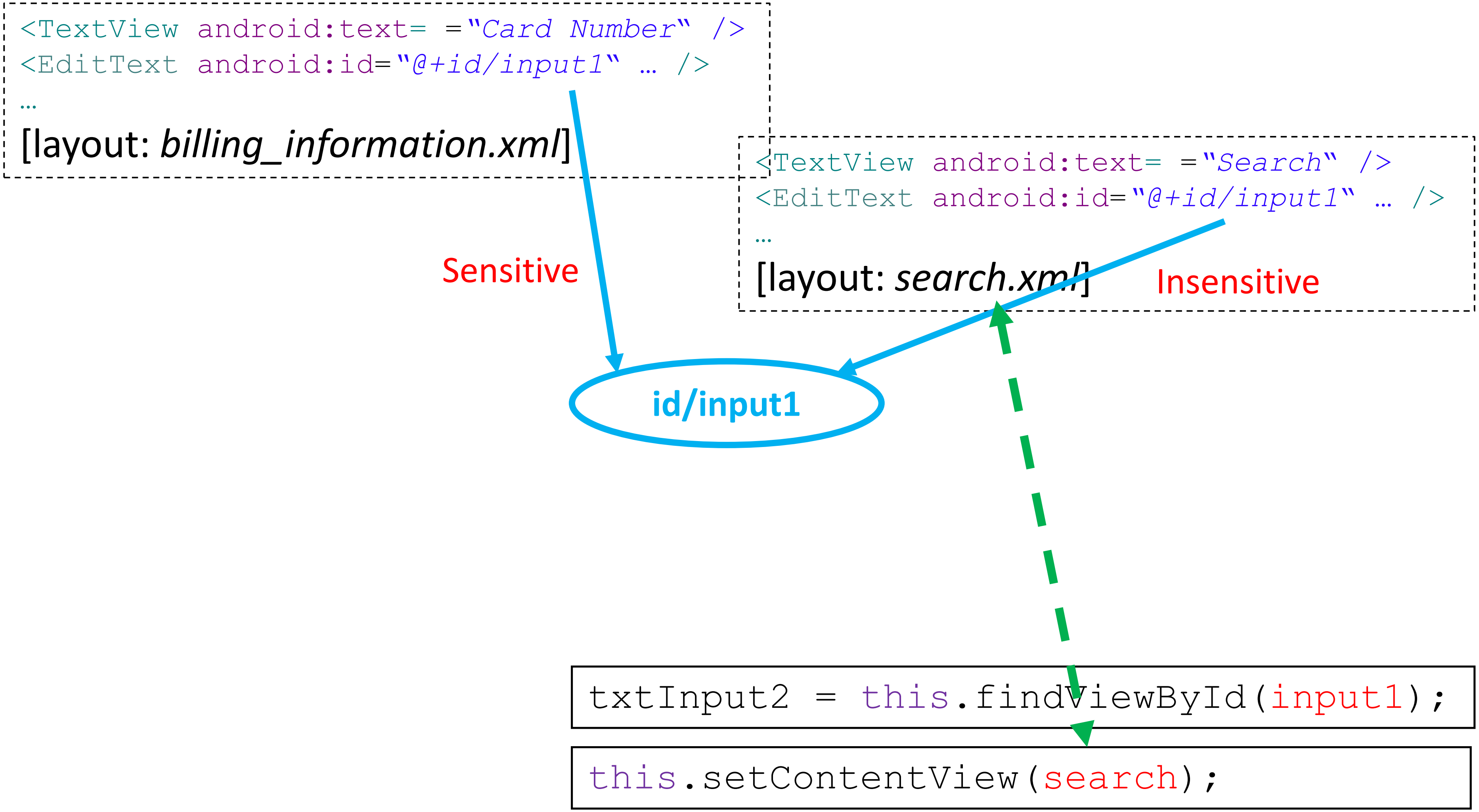
```
<TextView android:text= ="Search" />
<EditText android:id="@+id/input1" … />
…
```

[layout: *search.xml*]

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

```
this.setContentView(search);
```

Binding Variables (3)



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

[layout: *billing_information.xml*]

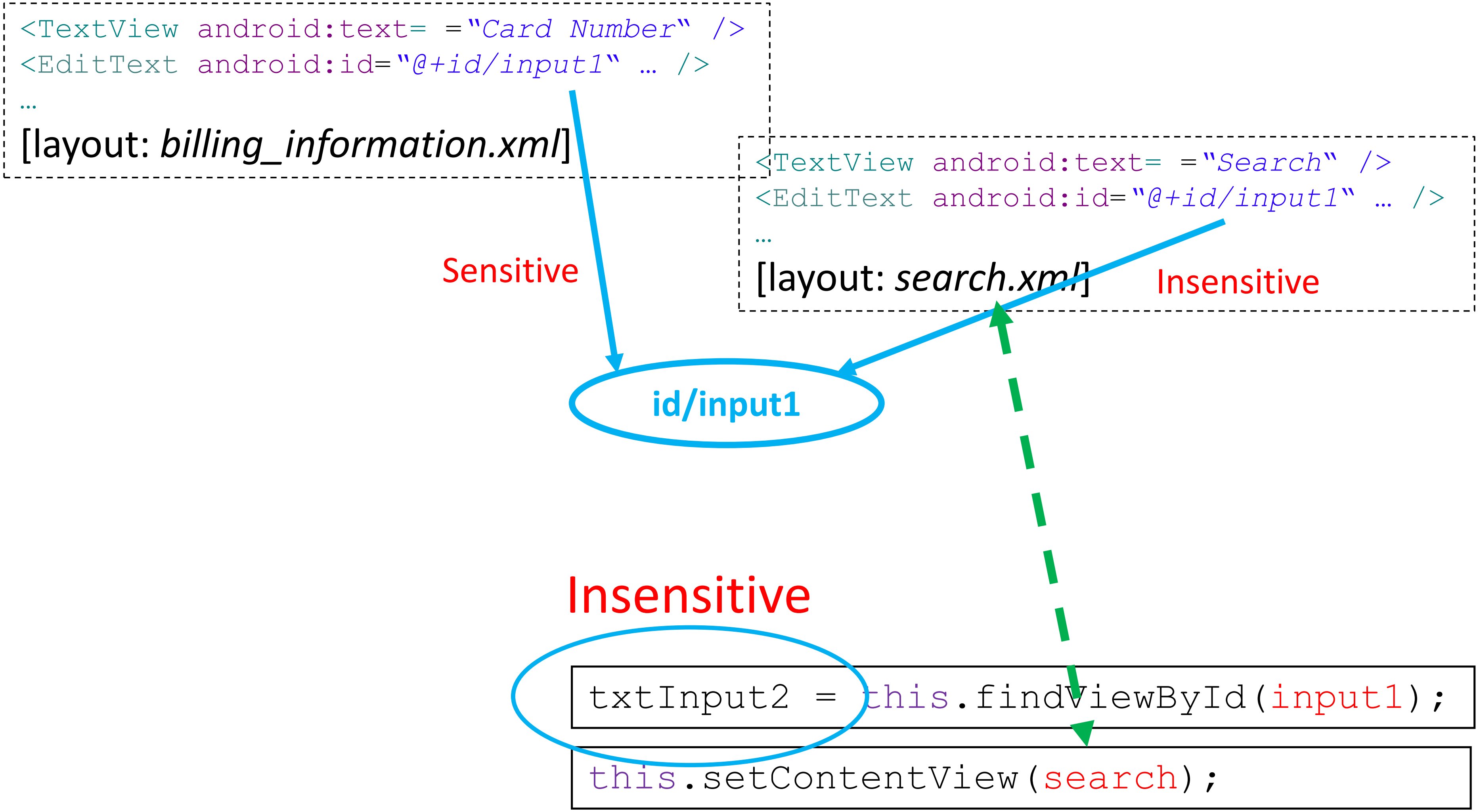
```
<TextView android:text= ="Search" />
<EditText android:id="@+id/input1" … />
…
```

[layout: *search.xml*]

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

```
this.setContentView(search);
```

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*]

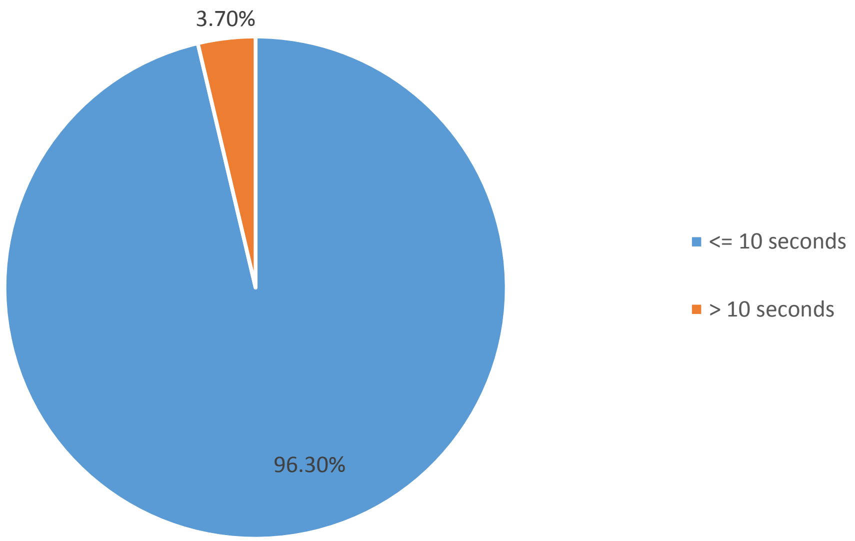
```
txtInput2 = this.findViewById(input1);
this.setContentView(search);
```

Implementation & Evaluation

- Implemented for Android apps and built on Dalysis[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)

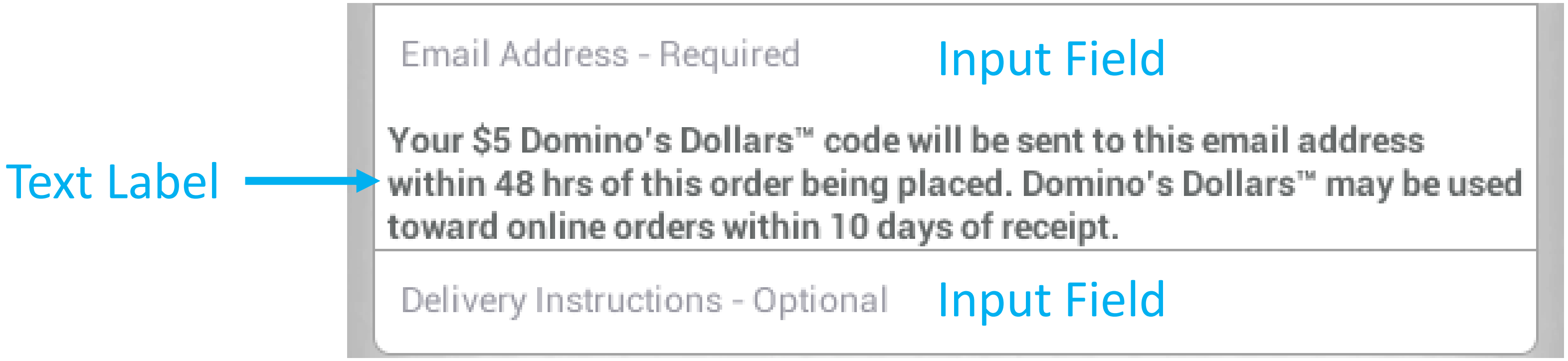
- 9,653 apps (60.33%) contains input fields
  - Accuracy
    - Manually examined 40 apps . 115 layouts are rendered and 485 input fields are analyzed.
    - **TP**: sensitive user inputs are identified as sensitive
    - **FP**: insensitive user inputs are identified as sensitive
    - **FN**: sensitive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

$$Recall = \frac{TP}{TP + FN} = 97.3\% \qquad Precision = \frac{TP}{TP + FP} = 97.3\%$$

- 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



Email Address - Required

Input Field

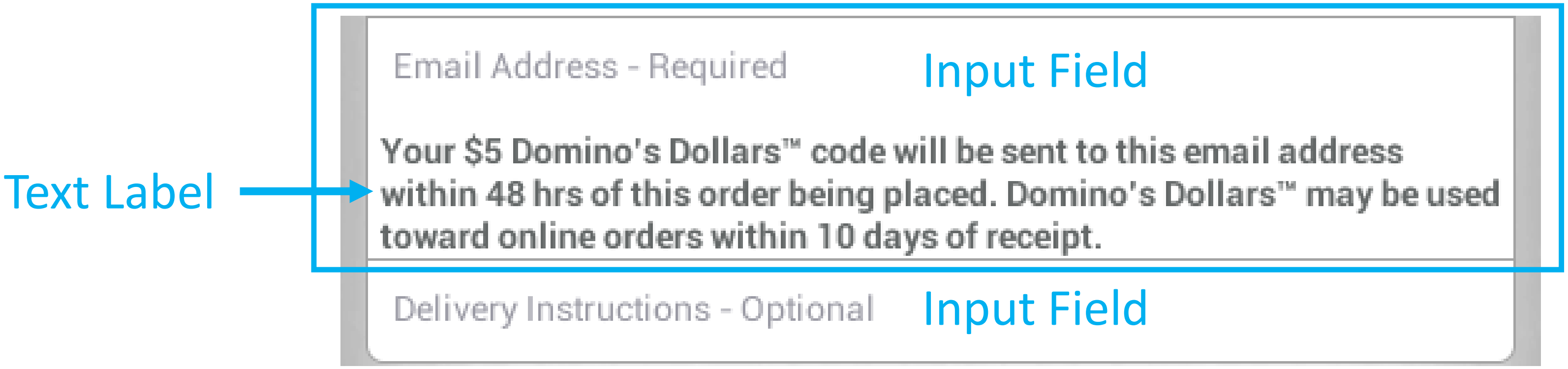
Text Label

Your $5 Domino's Dollars™ code will be sent to this email address within 48 hrs of this order being placed. Domino's Dollars™ may be used toward online orders within 10 days of receipt.

Delivery Instructions - Optional

Input 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

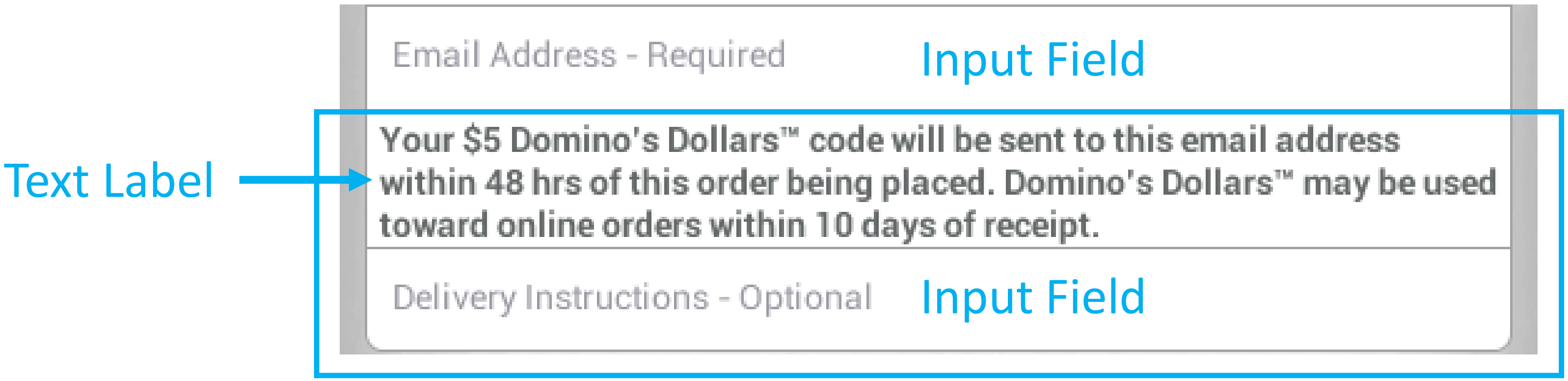


Email Address - Required

Your $5 Domino's Dollars™ code will be sent to this email address within 48 hrs of this order being placed. Domino's Dollars™ may be used toward online orders within 10 days of receipt.

Delivery Instructions - Optional

- 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

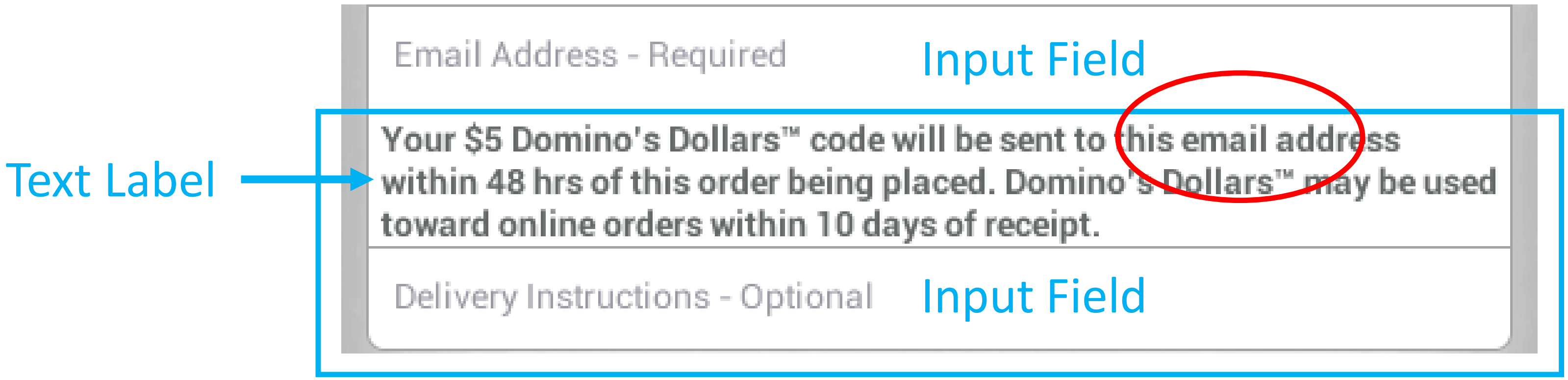


Email Address - Required

Your $5 Domino's Dollars™ code will be sent to this email address within 48 hrs of this order being placed. Domino's Dollars™ may be used toward online orders within 10 days of receipt.

Delivery Instructions - Optional

- 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

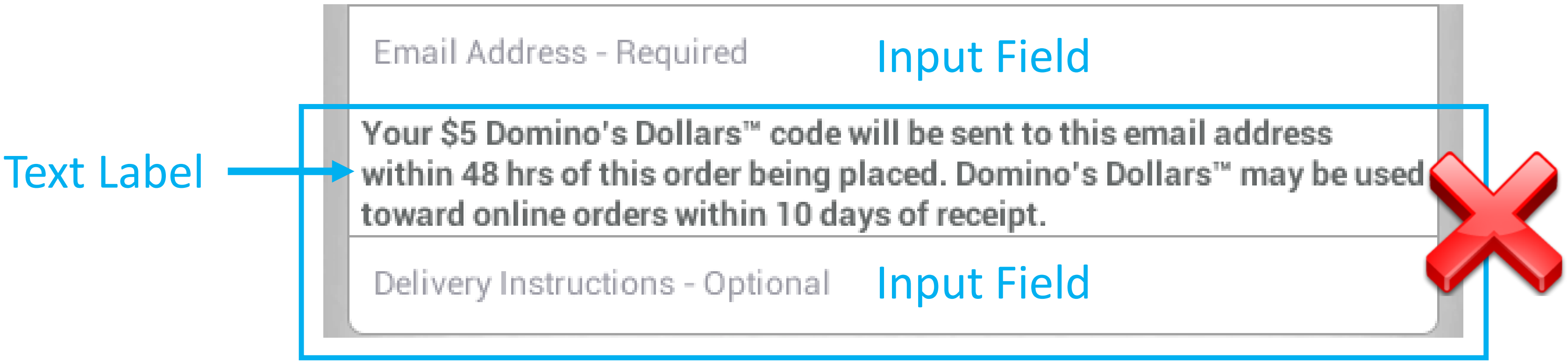


Email Address - Required

Your $5 Domino's Dollars™ code will be sent to this email address within 48 hrs of this order being placed. Domino's Dollars™ may be used toward online orders within 10 days of receipt.

Delivery Instructions - Optional

- 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



Email Address - Required

Your $5 Domino's Dollars™ code will be sent to this email address within 48 hrs of this order being placed. Domino's Dollars™ may be used toward online orders within 10 days of receipt.

Delivery Instructions - Optional

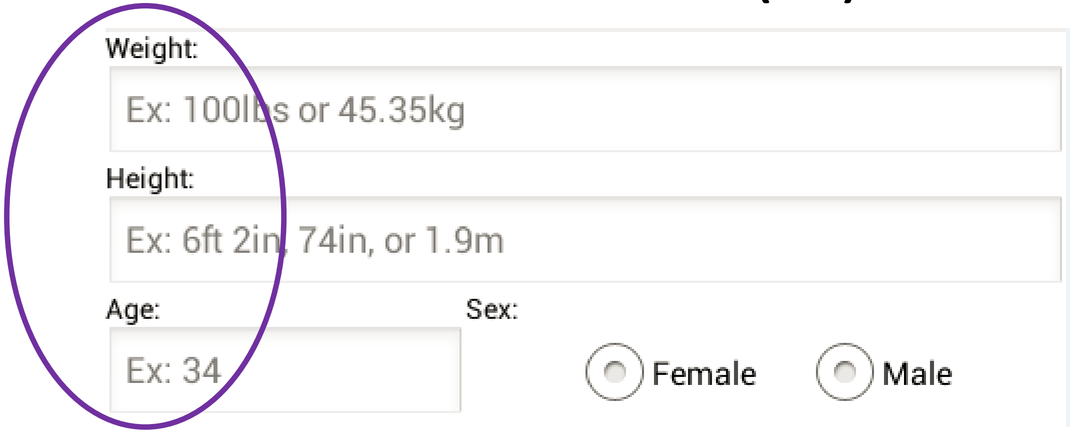
Evaluating Disclosure Analysis

- 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)



Weight:

Ex: 100lbs or 45.35kg

Height:

Ex: 6ft 2in, 74in, or 1.9m

Age:

Ex: 34

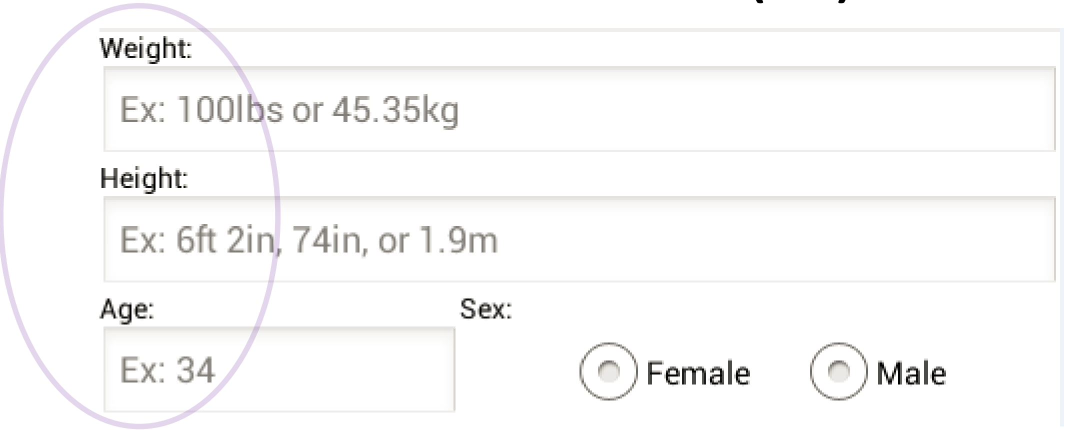
Sex:

○ Female ○ Male

com.canofsleep.wwdiary

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

Case Studies (1)



Weight:

Ex: 100lbs or 45.35kg

Height:

Ex: 6ft 2in, 74in, or 1.9m

Age:

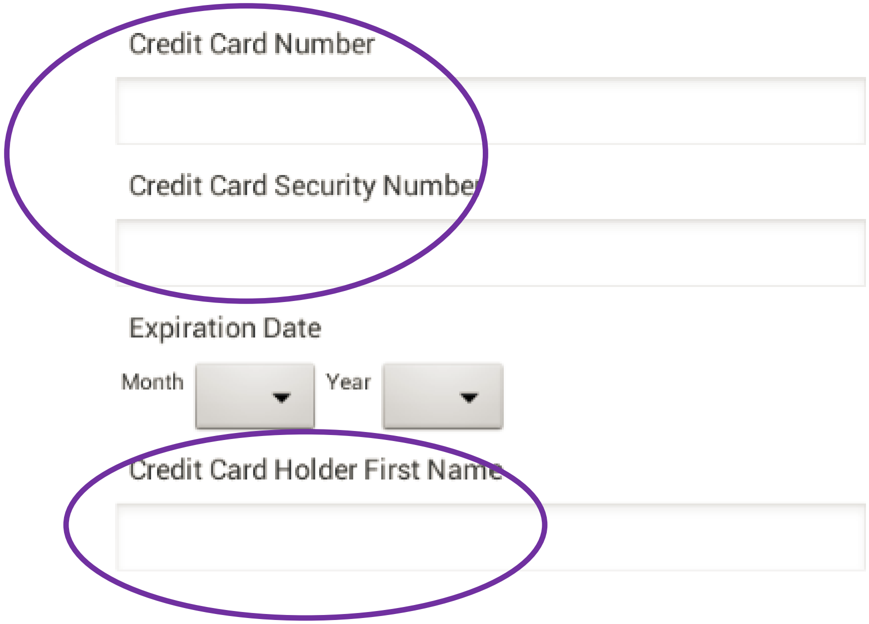
Ex: 34

Sex:

○ Female ○ Male

com.canofsleep.wwdiary

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



Credit Card Number

Credit Card Security Number

Expiration Date

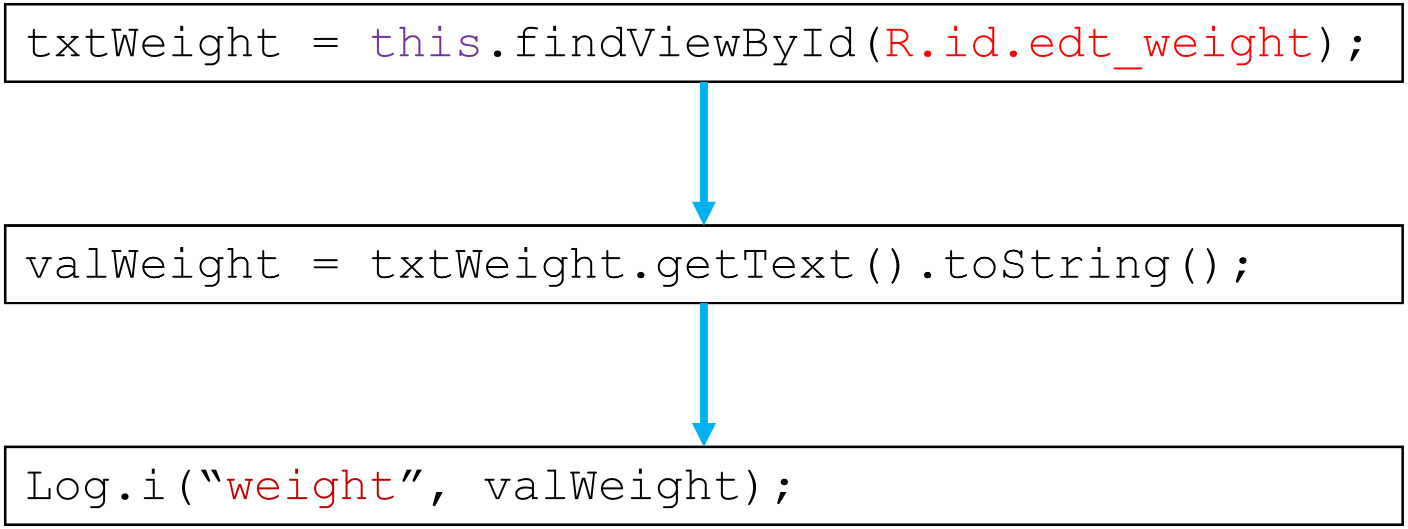
Month Year

Credit Card Holder First Name

com.nitrogen.android

The 3 marked inputs fields are identified sensitive and their data are disclosed.

Case Studies (2)



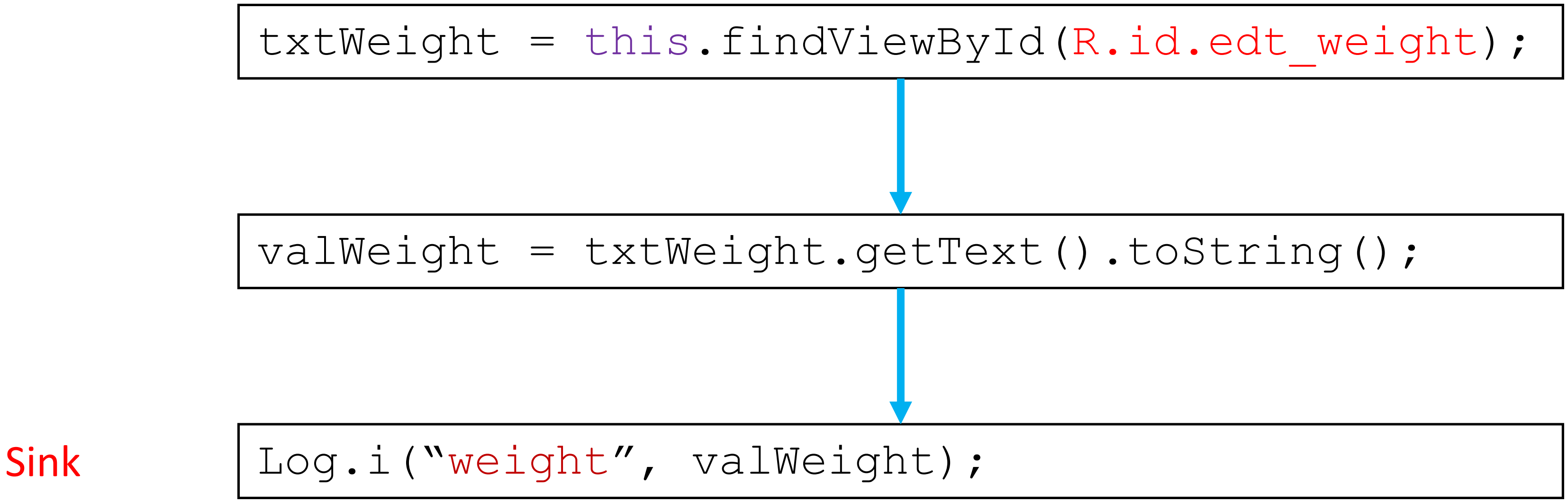
```
txtWeight = this.findViewById(R.id.edt_weight);
```

```
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.



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

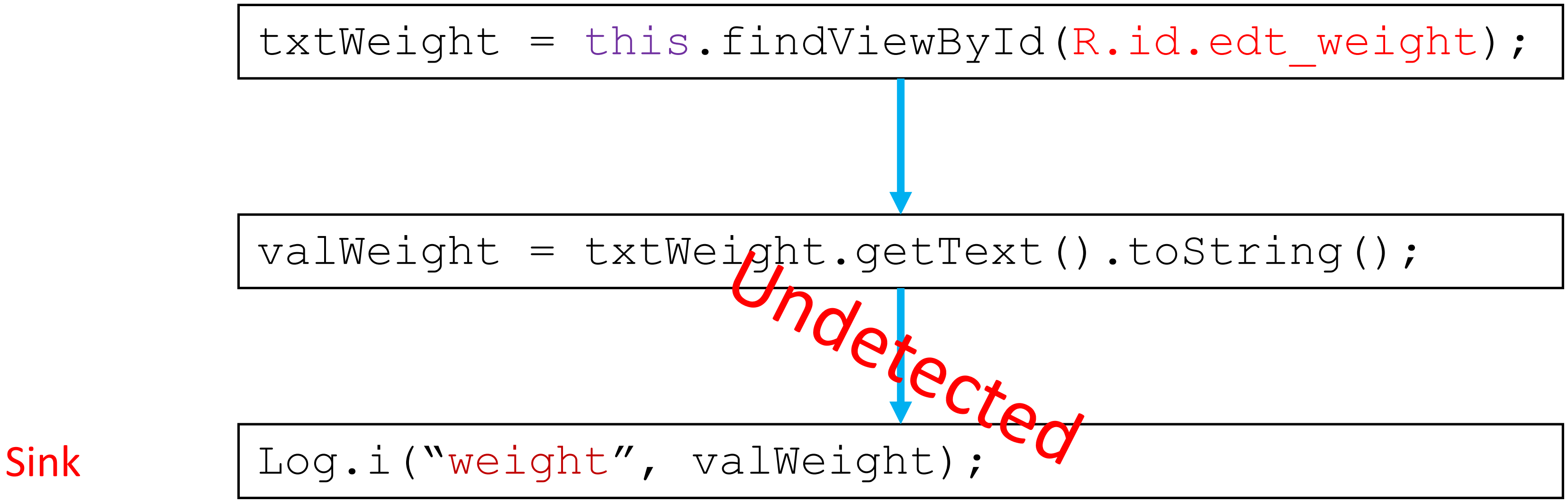
```
valWeight = txtWeight.getText().toString();
```

Sink

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

Case Studies (2)

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



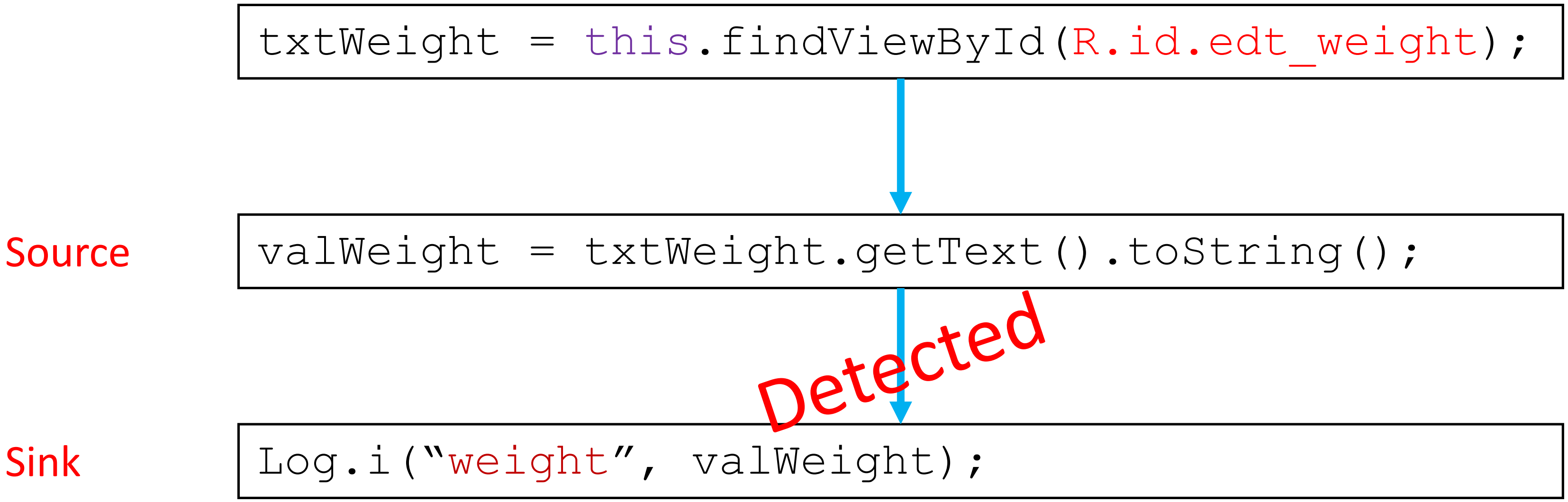
```
txtWeight = this.findViewById(R.id.edt_weight);
```

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

```
Log.i(“weight”, valWeight);
```

Case Studies (2)

- Disclosure analysis based on SUPOR



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

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

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.

Thank You!

Q & A

PURDUE UNIVERSITY

NEC Laboratories America
Relentless passion for innovation

Georgia Tech

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 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

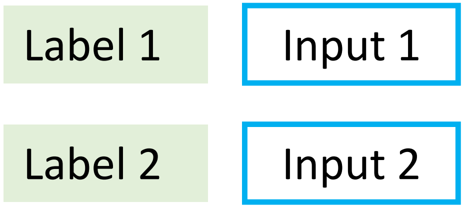
- Crawl texts from apps' resource files
- Adapt NLP techniques to extract nouns and noun phrases from the top 5,000 frequent text lines.
- Manually inspect top frequent nouns and noun phrases to identify sensitive keywords.

Why not use XML structure to compute correlation scores?

- Many developers defines 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?

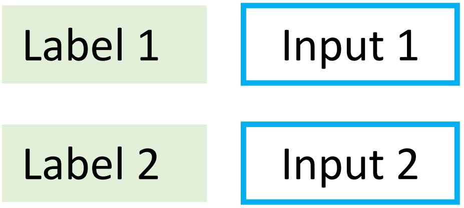
- Some cases in real Android apps.



Why not use XML structure to compute correlation scores?

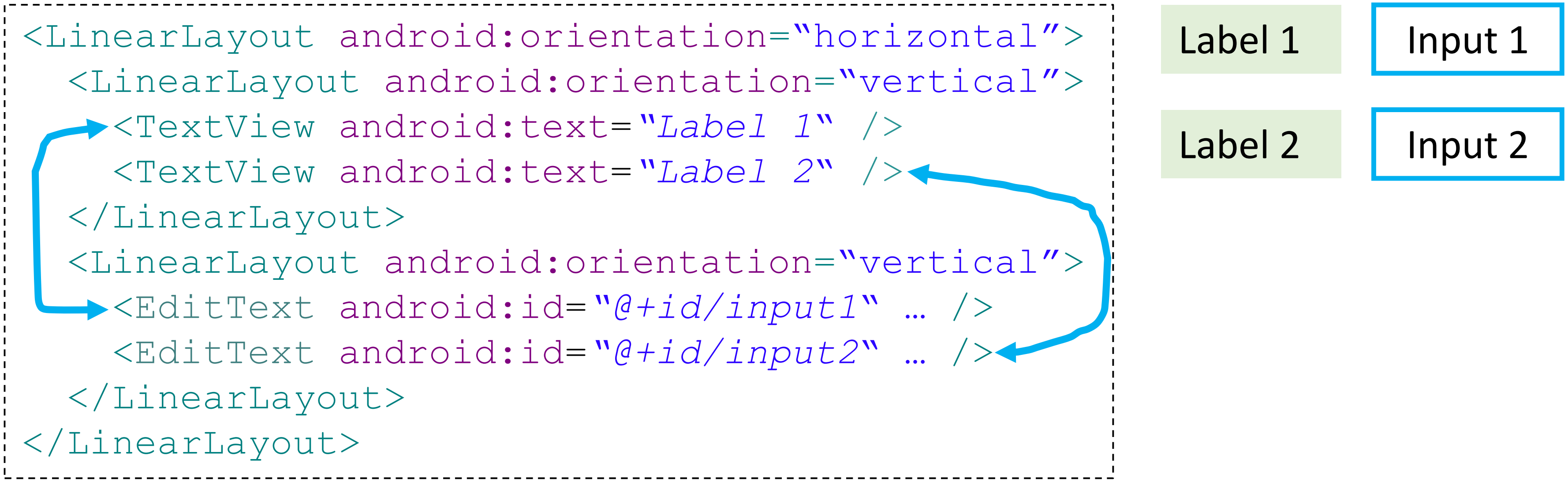
- Some cases in real Android apps.

```
<LinearLayout android:orientation=“horizontal”>
  <LinearLayout android:orientation=“vertical”>
    <TextView android:text=“Label 1‟ />
    <TextView android:text=“Label 2‟ />
  </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.



```
<LinearLayout android:orientation=“horizontal”>
  <LinearLayout android:orientation=“vertical”>
    <TextView android:text=“Label 1“ />
    <TextView android:text=“Label 2“ />
  </LinearLayout>
  <LinearLayout android:orientation=“vertical”>
    <EditText android:id=“@+id/input1“ … />
    <EditText android:id=“@+id/input2“ … />
  </LinearLayout>
</LinearLayout>
```