UIPicker: User-Input Privacy Identification in Mobile Applications

Yuhong Nan, Min Yang, Zhemin Yang, Shunfan Zhou
Fudan University

Guofei Gu
Texas A&M University

Xiaofeng Wang
Indiana University at Bloomington
Motivation

Privacy leakage

- App’s insecure implementation
- Vulnerabilities in system
Motivation

Existing Works

Dynamic/Static taint analysis

- TaintDroid [OSDI’10]
- FlowDroid [PLDI’14]

Access control mechanisms

- SELinux on Android
- Auriasium [SECURITY’12]

\[
\begin{align*}
c & = \text{taint_source}() \\
\ldots\end{align*}
\[
\begin{align*}
a & = b + c \\
\ldots \\
\text{network_send}(a)
\end{align*}
\]
But wait …

- Location
  - LocationManager.getLastKnownLocation()

- Contact
  - ContentResolver.query(CONTACT_URI)

- SMS
  - SmsMessage.getMessageBody()

- Phone Number
  - TelephonyManager.getLine1Number()

- ...

System Centric Privacy Data
But wait …

User-Input Centric Privacy Data (UIP Data)

Account Credentials & Profiles

Location

Financial
Challenges

• UIP data cannot be found without parsing the context and semantic of UIs

• Label all input as sensitive
  • Large number of false positives

• Manually specify such contents need to be protected
  • Intensive human intervention
Our Work

- **UIPicker**
  - Novel framework for automatic, large-scale User-Input Privacy (UIP data) identification within Android apps.

- **Runtime Security-Enhancement Mechanism**
  - Protect insecure UIP data transmission in app’s runtime
Key Observation

- Privacy-related semantics exist in
  - UI Screens
  - Layout resource files

```xml
<TextView android:text="@string/opl_new_payment_credit_card_number" />
<EditText android:id="@id/opl_credit_card_number" android:inputType="number" />
```
Technical challenges

• How to get complete privacy-related texts
  • Highly unstructured semantics
    • E.g. password, pass, pwd, ...
  • Sparsely distributed in different UIs
    • Impractical for manual collecting

• To what extent, a layout element could be UIP data
  • Keyword based search? imprecise results
    • E.g. Split “Username” into “user” & “name”
UIPicker Overview

1. Resource Extraction
2. Cluster privacy-related texts describing UIP data
3. Identify UIP data from textual semantics
4. Confirm UIP data with application code behaviors
Pre-Processing

- Resource Extraction
- Word splitting
  - Delimiter/Letter-separated words
- Redundant content removal
  - Non-English strings/Stop words
- Stemming
  - Reduce the number of texts
  - E.g. Change “secured”, “security” into “secure”

UI Texts

1. Add a new credit card
   - Credit Card Number
   - Expiration Date: 01/2014
   - Card Type: MasterCard
   - Cardholder’s Name
   - Add your card

Layout Descriptions

2. @id/opl_credit_card_number
   @string/opl_new_credit_card_expiration_date_month
   @string/opl_new_credit_card_save_button
Privacy-related Texts Analysis

- Privacy-related layout Selection
  - E.g. Login, Registration, settings page.
  - By initial seeds with heuristic rules

- Privacy-related texts clustering
  - Words appeared most in privacy-related layouts rather than normal layouts.
  - By applying Chi-Square test.
**UIP Data Identification**

- **Classifier**: Support Vector Machine
- **Features**
  - Privacy-related texts
  - Sibling elements
    - E.g. Short phrase for describing the input field.
- **Training Data**
  - Subset of UIP data element
    - Text fields with sensitive `inputType`.
    - `<EditText android:id="@id/password_edittext" android:inputType="textPassword" />`
    - Manual labelled element for Financial inputs
Behavior Based Result Filtering

- Matching UIP data behaviors reflected in application’s program code
- Filter out static labels which only contain privacy-related semantics
- Recover Input fields other than “EditText”
  - Implicit user input
    - E.g. Drop down list
  - Customized input
    - E.g. com.alipay.inputBox
1. Locating UIP candidate & its layout

```java
Activity.setContentView(R.layout.add_credit_card.xml)
```

```java
Void OnCreate(Bundle bundle){
    InputBox IB = findViewById(2131231511);
    submitBtn = findViewById(2131623982);
    submitBtn.setOnClickListener(new addCardListener());
}
```

2. User-Triggered Event handling

```java
addCardListener.onClick(View v) {
    creditCard = IB.getText();
    sendContent(creditCard);
}
```
## Runtime Security Enhancement

### Taint Tracking based on UIPicker

**UIPicker**

- **Application Name**: Qunar Travel
- **Package Name**: com.Qunar
- **Destination IP Address**: 59.151.16.178
- **The following data will be insecurely sent**: 

**Verify Code**: 727114

<table>
<thead>
<tr>
<th>银行卡支付</th>
<th>银行卡支付</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank of China (Credit Card)</td>
<td>银行卡支付</td>
</tr>
<tr>
<td>6227 6121 4583 0440</td>
<td>银行卡支付</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>日期</th>
<th>03/20</th>
<th>银行卡支付</th>
</tr>
</thead>
<tbody>
<tr>
<td>信贷卡</td>
<td>银行卡支付</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>信用额度</th>
<th>369</th>
<th>银行卡支付</th>
</tr>
</thead>
<tbody>
<tr>
<td>信用额度</td>
<td>银行卡支付</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>持卡人姓名</th>
<th>bob</th>
<th>银行卡支付</th>
</tr>
</thead>
<tbody>
<tr>
<td>持卡人姓名</td>
<td>银行卡支付</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>身份证号</th>
<th>310109196509192811</th>
<th>银行卡支付</th>
</tr>
</thead>
<tbody>
<tr>
<td>身份证号</td>
<td>银行卡支付</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>电话号码</th>
<th>138 1839 9589</th>
<th>银行卡支付</th>
</tr>
</thead>
<tbody>
<tr>
<td>电话号码</td>
<td>银行卡支付</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>验证码</th>
<th>727114</th>
<th>银行卡支付</th>
</tr>
</thead>
<tbody>
<tr>
<td>验证码</td>
<td>银行卡支付</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>银行卡支付</th>
<th>银行卡支付</th>
</tr>
</thead>
<tbody>
<tr>
<td>银行卡支付</td>
<td>银行卡支付</td>
</tr>
</tbody>
</table>

### Plain Text transmission
- Runtime checking
- Insecure SSL implementation
- Offline analysis with MalloDroid [CCS' 12]
UIPicker Evaluation

• Dataset
  • 17,425 apps crawled from Google-Play in Oct. 2014
  • 35 categories, 500 apps for each
• 6,179 (35.46%) contain UIP data in 17,425 Apps
• Exist in more than half of apps in 9 out of 35 categories.
Effectiveness

- Compare with System Defined APIs (# Apps)

<table>
<thead>
<tr>
<th>Category</th>
<th>System Defined APIs</th>
<th>UIPickerView</th>
<th>Overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Credentials &amp; User Profile</td>
<td>4,900</td>
<td>5,330</td>
<td>1,340</td>
</tr>
<tr>
<td>Location</td>
<td>15,221</td>
<td>2,883</td>
<td>2,282</td>
</tr>
<tr>
<td>Financial</td>
<td>-</td>
<td>1,318</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>15,632</td>
<td>6,179</td>
<td>-</td>
</tr>
</tbody>
</table>

- TelephonyManager.getLine1Number()
- AccountManager.getAccounts()
- LocationManager.getLastKnownLocation()
- Location.getLongitude()
- Location.getLatitude()
## Effectiveness

- **Compare with Sensitive Attributes (# Elements)**

<table>
<thead>
<tr>
<th>Category</th>
<th>InputType</th>
<th>UIPickerView</th>
<th>Incremental</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Credentials &amp; User Profile</td>
<td>24,021</td>
<td>46,227</td>
<td>26,087</td>
</tr>
<tr>
<td>Location</td>
<td>941</td>
<td>14,311</td>
<td>13,370</td>
</tr>
<tr>
<td>Financial</td>
<td>-</td>
<td>6,353</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>24,962</td>
<td>71,224</td>
<td>46,262</td>
</tr>
</tbody>
</table>

Sensitive Attributes:
- `textEmailAddress`  
- `textPersonName`  
- `textPassword`  
- `textVisiblePassword`  
- `password/email/phoneNumber`  
- `textPostalAddress`  

*15x compared to Incremental*  
*2x compared to Incremental*
## Effectiveness

- **Compare with Sensitive Attributes(# Elements)**

<table>
<thead>
<tr>
<th>Type</th>
<th># Elements</th>
<th>% in UIP Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextView</td>
<td>10,582</td>
<td>14.86%</td>
</tr>
<tr>
<td>Customized</td>
<td>5,075</td>
<td>7.13%</td>
</tr>
<tr>
<td>Spinner</td>
<td>1,962</td>
<td>2.75%</td>
</tr>
<tr>
<td>Others</td>
<td>784</td>
<td>1.10%</td>
</tr>
<tr>
<td>Total</td>
<td>18,403</td>
<td>25.84%</td>
</tr>
</tbody>
</table>
Precision

• Manual Validation
  • 200 random selected apps in 10 categories (20 in each)
  • False Positives: 67/1042 elements
  • False Negatives: 107 elements
  • Overall: \(93.6\%\) Precision and \(90.1\%\) Recall

\[
\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}
\]
Conclusion

• UIPicker: Identify UIP data based on novel combination of NLP, machine-learning approach, and static analysis techniques

• Runtime security enhancement based on UIPicker

• Easily be deployed for other existing mechanisms for privacy analysis/protection.
Thanks

Questions?