



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

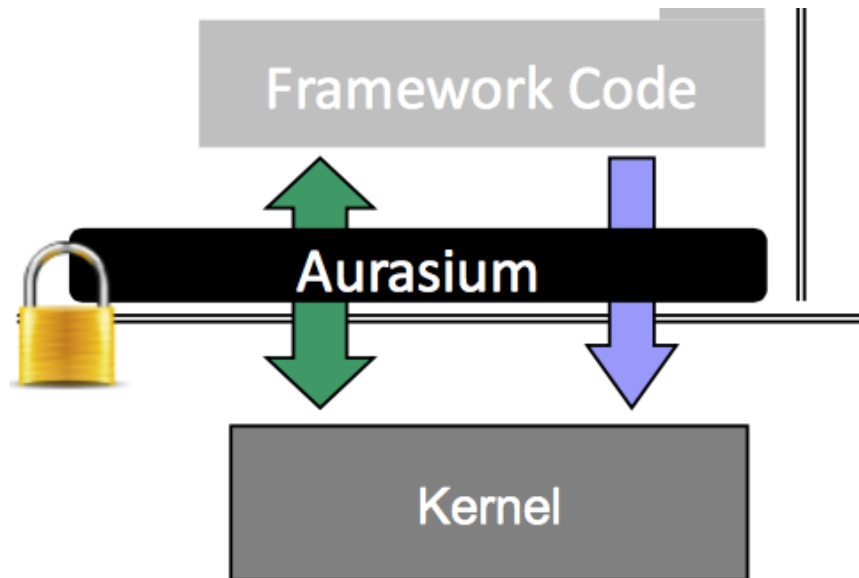


Motivation

Existing Works

Dynamic/Static taint analysis

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

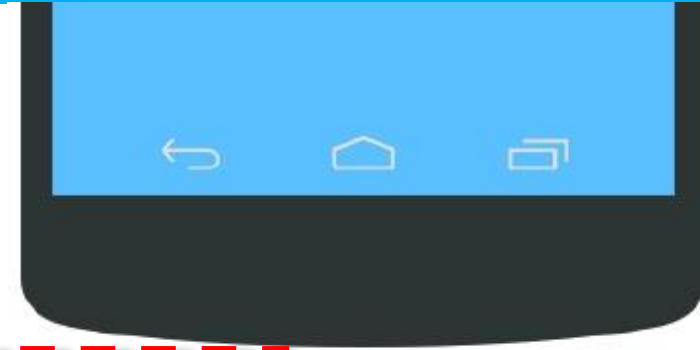


```
c = taint_source()
...
a = b + c
...
network_send(a)
```

Access control mechanisms

- SELinux on Android
- Auriasium [SECURITY'12]

But wait ...



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

System Centric Privacy Data

But wait ...

Account Credentials
& Profiles

Full name
First name, last name

Skype Name
6-32 characters

Password
6-20 characters

Repeat password
6-20 characters

Email
someone@example.com

Mobile number
Phone number (e.g. +44 1234567)

Yes, send me Skype news and promotions.

Location

Or enter a new shipping address
Be sure to click "Ship to this address" when done.

Full Name:

Address Line 1:
Street address, P.O. box, company name, c/o

Address line 2:
Apartment, suite, unit, building, floor, etc.

City:

State/Province/Region:

ZIP:

Financial

银行卡支付

中国银行(信用卡)
6227 6121 4583

03/20

369

bob

身份证
310109196509

138 1839

727114

同意《协议》并保存卡信息, 支付更便捷

支付金额¥95

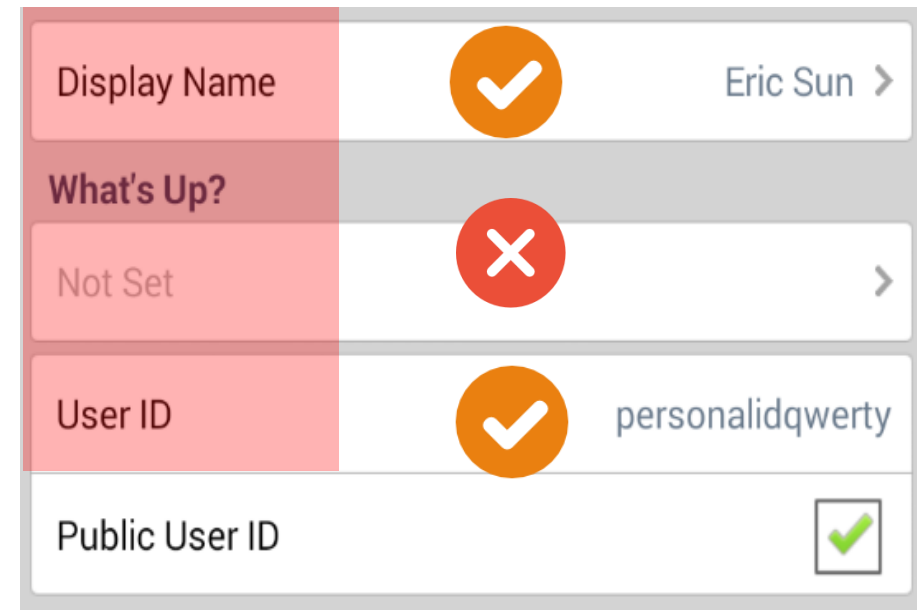
立即支付

User-Input Centric Privacy Data (UIP Data)

Challenges

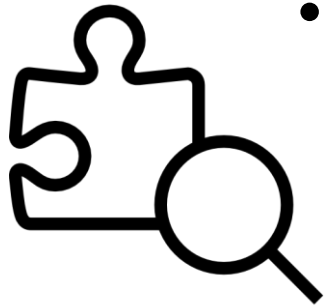
- UIP data cannot be found without parsing the context and semantic of UIs

- 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

Add a new credit card

Credit Card Number

Expiration Date: 01 2014

Card Type: MasterCard

Cardholder's Name

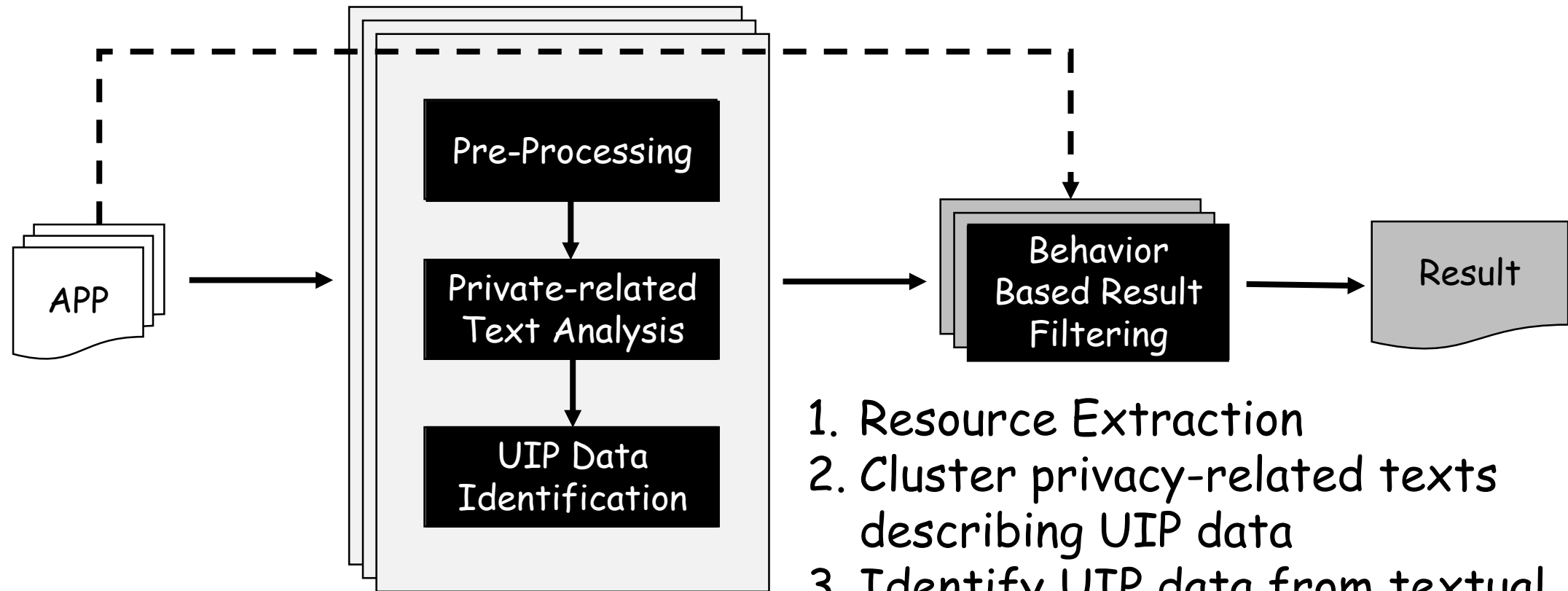
Add your card

```
<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"

1 UI Texts

Add a new credit card

Credit Card Number

Expiration Date: 01 2014

Card Type: MasterCard

Cardholder's Name

Add your card

2 Layout Descriptions

@id/opl_credit_card_number

@string/opl_new_credit_card_expiration_date_month

@string/opl_new_credit_card_save_button

Privacy-related Texts Analysis

Initial seeds

phonenumber
email
username
password

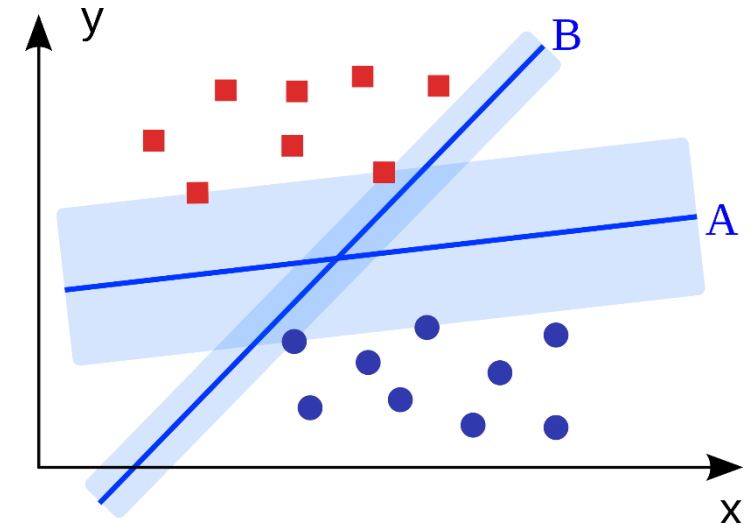
Inferred texts

firstnam
zipcod birthdat
location address
pay provinc nicknam
account password mm
zip yyyy
code payment
pwd citi pass
street yy debit
secur expir transact age mail
bank male countri locat
pin phone user birth
femal contact email
passwd usernam
dd
cellphon
lastnam birthday

- 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" />
```

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`

The screenshot shows a mobile app interface for signing up. A red dashed box highlights the top section, which includes a checked checkbox for "Upload my address book to connect me with my friends.", a "Sign Up" button, and a paragraph of text: "By tapping 'Sign Up' above, you are agreeing to the [Terms of Service](#) and [Privacy Policy](#), including [Cookies use](#)." Below this is another line of text: "Your email address and phone number may be used to connect you with others, but will not be". To the right of this text is a gear icon. A red circle with a white 'X' is overlaid on the right side of the highlighted area. Below the highlighted area, there are two input fields: "Expiration Date:" with a date picker showing "01" and "2014", and "Card Type:" with a dropdown menu showing "MasterCard". A red circle with a white checkmark is overlaid on the bottom right corner of the form.

Behavior Based Result Filtering

1. Locating UIP candidate & its layout

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

```
Void onCreate(Bundle bundle){
```

```
    InputBox IB = findViewById(2131231511);
```

```
    submitBtn = findViewById(2131623982);
```

```
    submitBtn.setOnClickListener(new addCardListener());
```

```
}
```

2. User-Triggered Event handling

```
addCardListener.onClick(View v) {
```

```
    .....
```

```
    creditCard = IB.getText();
```

```
    sendContent(creditCard)
```

```
    .....
```

```
}
```

Add a new credit card

Credit Card Number **UIP Data Elements**

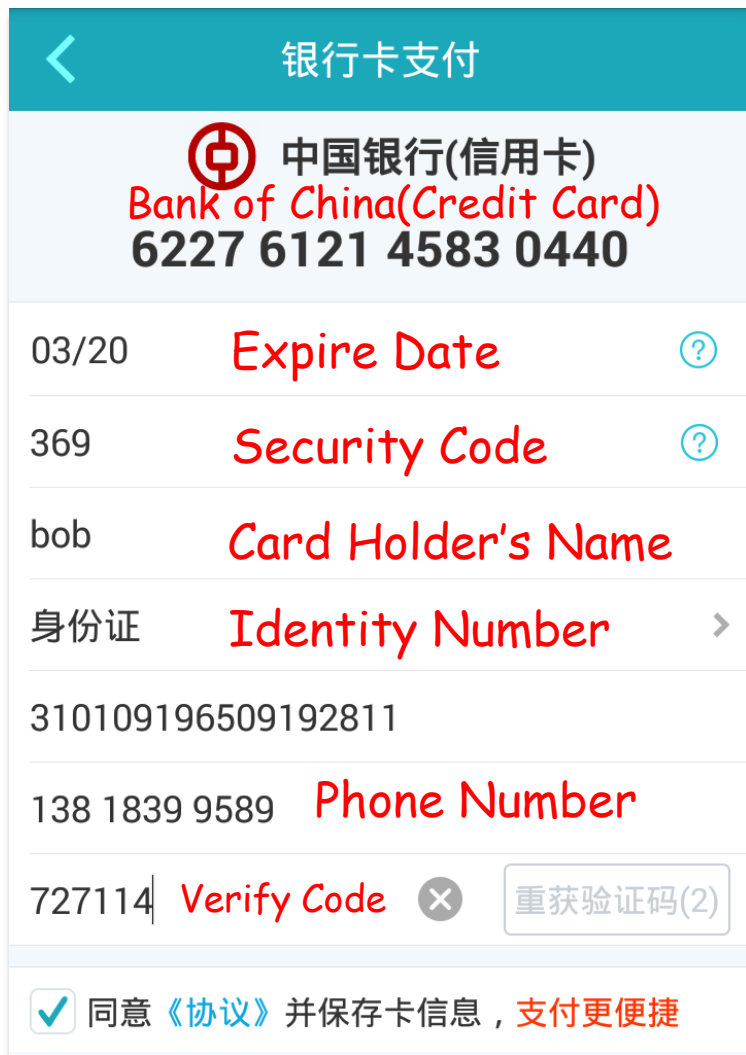
Expiration Date: 01 2014

Card Type: MasterCard

Cardholder's Name

Add your card

Runtime Security Enhancement

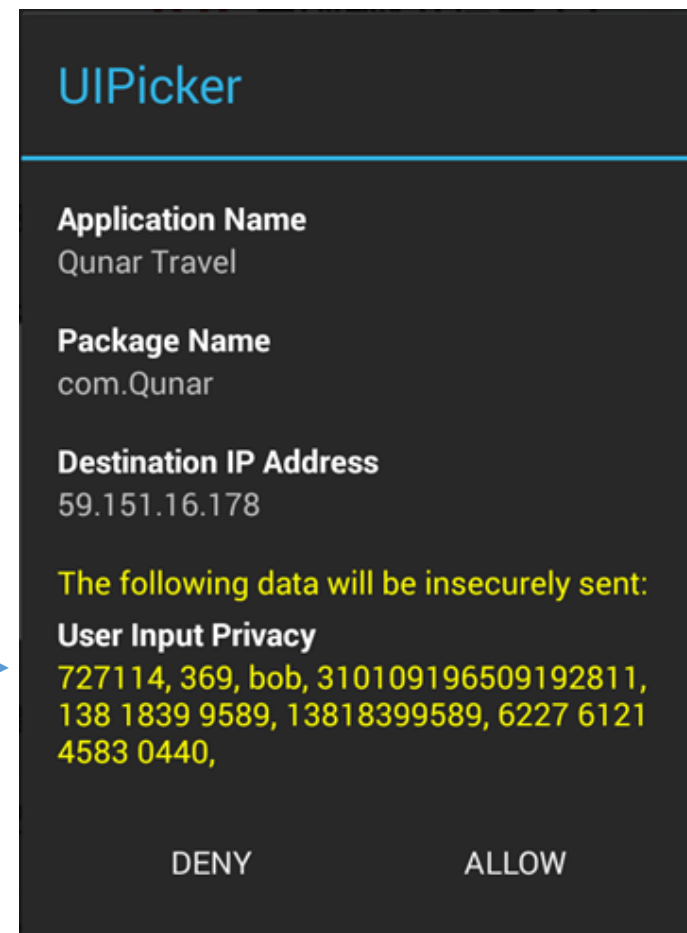


Taint Tracking based on UIPicker



TaintDroid

```
c = taint_source()  
...  
a = b + c  
...  
network_send(a)
```



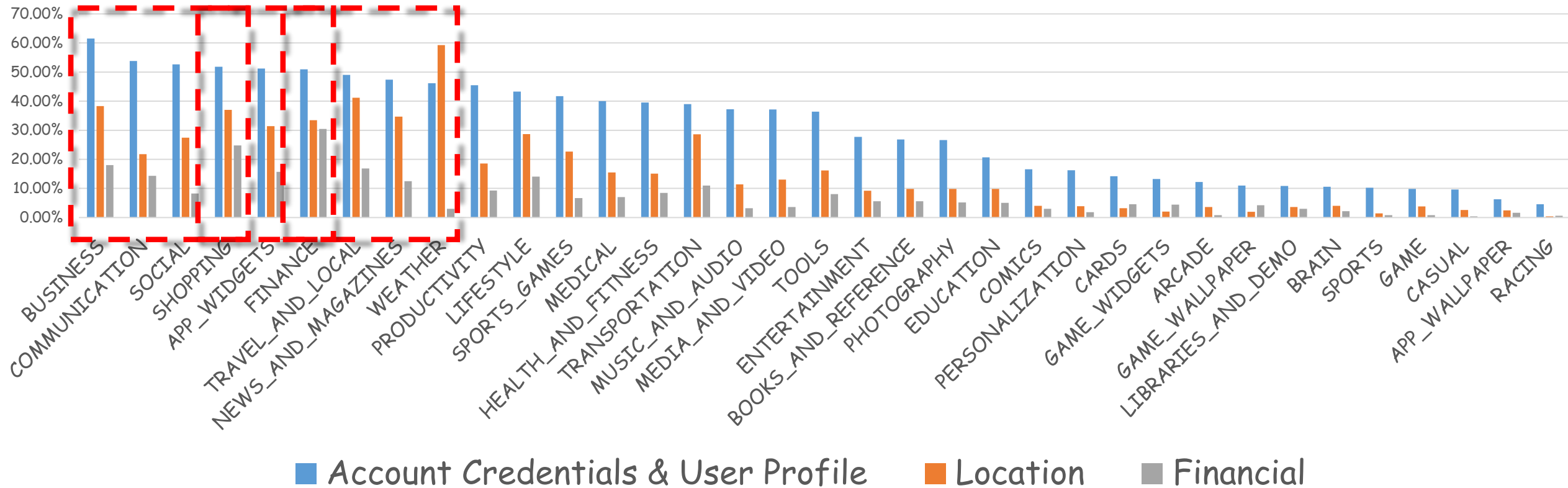
- Plain Text transmission
 - Runtime checking
- Insecure SSL implementation
 - Offline analysis with MalloDroid [CCS' 12]

UIPicker Evaluation

- Dataset
 - 17,425 apps crawled from Google-Play in Oct.2014
 - 35 categories, 500 apps for each

UIP Data Distribution

- 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)

TelephonyManager.getLine1Number()
AccountManager.getAccounts()

LocationManager.getLastKnownLocation()
Location.getLongitude()
Location.getLatitude()

Category	System Defined APIs	UIPicker	Overlap
Account Credentials & User Profile	4,900	5,330	1,340
Location	15,221	2,883	2,282
Financial	-	1,318	-
Total	15,632	6,179	-

Effectiveness

- Compare with Sensitive Attributes(# Elements)

textEmailAddress textPersonName
textPassword textVisiblePassword
password/email/phoneNumber

textPostalAddress

Category	Input Type	UIPicker	Incremental
Account Credentials & User Profile	24,021	46,227	26,087
Location	941	14,311	13,370
Financial	-	6,353	-
Total	24,962	71,224	46,262

Annotations: A red dashed box highlights the 'Input Type' column values (24,021, 941, -, 24,962). A red dashed box highlights the 'UIPicker' column values (46,227, 14,311, 6,353, 71,224). A red dashed box highlights the 'Incremental' column values (26,087, 13,370, -, 46,262). Red dashed arrows point from the 'textPostalAddress' box to the 'UIPicker' column and from the 'Input Type' box to the 'Input Type' column. Red arrows on the right indicate a 15x increase from Location to Total in the Incremental column, and a 2x increase from Financial to Total in the Incremental column.

Effectiveness

- Compare with Sensitive Attributes(# Elements)

Type	# Elements	% in UIP Data
TextView	10,582	14.86%
Customized	5,075	7.13%
Spinner	1,962	2.75%
Others	784	1.10%
Total	18,403	25.84%

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**

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

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?