FlowCog: Context-aware Semantics Extraction and Analysis of Information Flow Leaks in Android Apps

Xiang Pan, Yinzhi Cao, Xuechao Du, Boyuan He, Gan Fang, Yan Chen.

Northwestern University, Johns Hopkins University
Zhejiang University, Google
Roadmap

1. **Motivating Example**
2. FlowCog Overview
3. Design
   a. View Dependency Explorer
   b. Flow and Semantics Correlation Inference
4. Implementation
5. Evaluation & Case Study
6. Conclusion
Let our app provide accurate weather information based on your current location, this ...

"Share location to automatically update city"
Cannot tell which flow is legitimate!

58% flows are legitimate!

Big burden on users!
Roadmap

1. Motivating Example
2. FlowCog Overview
3. Design
   a. View Dependency Explorer
   b. Flow and Semantics Correlation Inference
4. Implementation
5. Evaluation & Case Study
6. Conclusion
FlowCog Overview

- First system to extract Android app's flow-specific semantics.
- Associate the semantics with flow's behavior.

High level steps:
- Associate each flow with its related views via static analysis and an optional dynamic analysis.
- Extract view semantics. (e.g., “Update Weather”)
- Determine if semantics provides information about flow behavior.
FlowCog Architecture: Semantics Extraction (1/2)

1. Data flow analysis with FlowDroid.
2. Activation event and guarding conditions.
3. View dependency explorer.

"Share location to automatically update city"

UpdateWeatherAsyncTask: doInBackground
if (allowShareLoc) sendData
else ...

Guarding Condition
FlowCog: Flow and Semantics Correlation Inference (2/2)

Semantic Extractor

Ex: “Share location to automatically update city” =>
<update, city>, <share, location>

Classifier

[<Verb, Noun>, ...

Filter

Learning classifier.
Learning-free classifier

App Description

App description.

Flow-specific texts.

NLP
Roadmap

1. Motivating Example
2. FlowCog Overview
3. **Design**
   a. View Dependency Explorer
   b. Flow and Semantics Correlation Inference
4. Implementation
5. Evaluation & Case Study
6. Conclusion
Design: View Dependency Explorer

- For analysis purposes:
  - Formally into taint analysis problem.
    - Source: all the views.
    - findViewById(...) call statements.
    - All new statements that create classes (and their builders) extending View and Dialog classes.
  - Sink:
    - Statements in given data flow.
    - Guarding condition statements.
    - All the activation events’ registration statements.
  - Use IFDS framework provided by FlowDroid.
Design: Flow and Semantics Correlation Inference

- Vectorize input using TF-IDF.
- Classify using SVM and Gradient Boosting.
- Use Word2Vec to convert two inputs into two vector lists, and then compute their similarity score.

App description.

Flow-specific texts.

Documentation of source and sink methods.
Roadmap

1. Motivating Example
2. FlowCog Overview
3. Design
   a. View Dependency Explorer
   b. Flow and Semantics Correlation Inference
4. Implementation
5. Evaluation & Case Study
6. Conclusion
## Implementation

<table>
<thead>
<tr>
<th>Component</th>
<th>Language</th>
<th>Loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow-related Semantics Extraction</td>
<td>Java</td>
<td>~12,000</td>
</tr>
<tr>
<td>Classifier</td>
<td>Python</td>
<td>~3,000</td>
</tr>
<tr>
<td>Dynamic Analysis</td>
<td>Python, Java</td>
<td>~1,000</td>
</tr>
<tr>
<td>Total</td>
<td>Python, Java</td>
<td>~16,000</td>
</tr>
</tbody>
</table>
Roadmap

1. Motivating Example
2. FlowCog Overview
3. Design
   a. View Dependency Explorer
   b. Flow and Semantics Correlation Inference
4. Implementation
5. Evaluation & Case Study
6. Conclusion
## Evaluation: Ground Truth

<table>
<thead>
<tr>
<th>Type</th>
<th>Apps</th>
<th>Apps with Flows</th>
<th>Legitimate Flows</th>
<th>Malicious Flows</th>
<th>Total Flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>1,299/4,500</td>
<td>361</td>
<td>688</td>
<td>355</td>
<td>1,043</td>
</tr>
<tr>
<td>Malicious [Drebin dataset]</td>
<td>586/1,500</td>
<td>255</td>
<td>675</td>
<td>624</td>
<td>1,299</td>
</tr>
<tr>
<td>Overall</td>
<td>1,885/6,000</td>
<td>616</td>
<td>1,363</td>
<td>979</td>
<td>2,342</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Flows</th>
<th>Precision</th>
<th>Recall</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>1,043</td>
<td>90.3%</td>
<td>95.1%</td>
<td>90.7%</td>
</tr>
<tr>
<td>Malicious</td>
<td>1,299</td>
<td>89.9%</td>
<td>91.0%</td>
<td>89.6%</td>
</tr>
<tr>
<td>Overall</td>
<td>2,342</td>
<td>90.1%</td>
<td>93.1%</td>
<td>90.2%</td>
</tr>
</tbody>
</table>
Evaluation: Size of Data Set Matters.

<table>
<thead>
<tr>
<th>Type</th>
<th># of Flows</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credential</td>
<td>443</td>
<td>96.2%</td>
</tr>
<tr>
<td>Device/Card Id</td>
<td>373</td>
<td>91.4%</td>
</tr>
<tr>
<td>Phone number</td>
<td>103</td>
<td>94.2%</td>
</tr>
<tr>
<td>Internet</td>
<td>1,009</td>
<td>91.7%</td>
</tr>
<tr>
<td>SMS</td>
<td>233</td>
<td>83.7%</td>
</tr>
<tr>
<td>Location</td>
<td>173</td>
<td>79.2%</td>
</tr>
<tr>
<td>Contact</td>
<td>132</td>
<td>79.5%</td>
</tr>
<tr>
<td>Calendar</td>
<td>12</td>
<td>66.7%</td>
</tr>
</tbody>
</table>

Less accurate flow types can be improved with more training data.
Case Study: Home of Ocarina

-Leaks out users’ geo-location.
- Labeled as legitimate because of extracted semantics.

“Map”, “The location of home of Ocarina”
Case Study: SMS Irritate

-Leaks out user-specific information via SMS.
- Labeled as legitimate.

“Send SMS”, “Number of SMS to flood”, “Message”
Case Study: Merry Christmas

-Leaks out users’ information to Internet.
- Labeled as malicious.

“Move the box to the target empty position...”
Conclusion

- FlowCog is the first system to extract flow-specific semantics.
- FlowCog adopts NLP techniques to associate flow-specific semantics with flow behaviors.
- Our evaluation results show that FlowCog can achieve a precision of 90.1% and a recall of 93.1%.
Thanks!

FlowCog open-source at: https://github.com/SocietyMaster/FlowCog