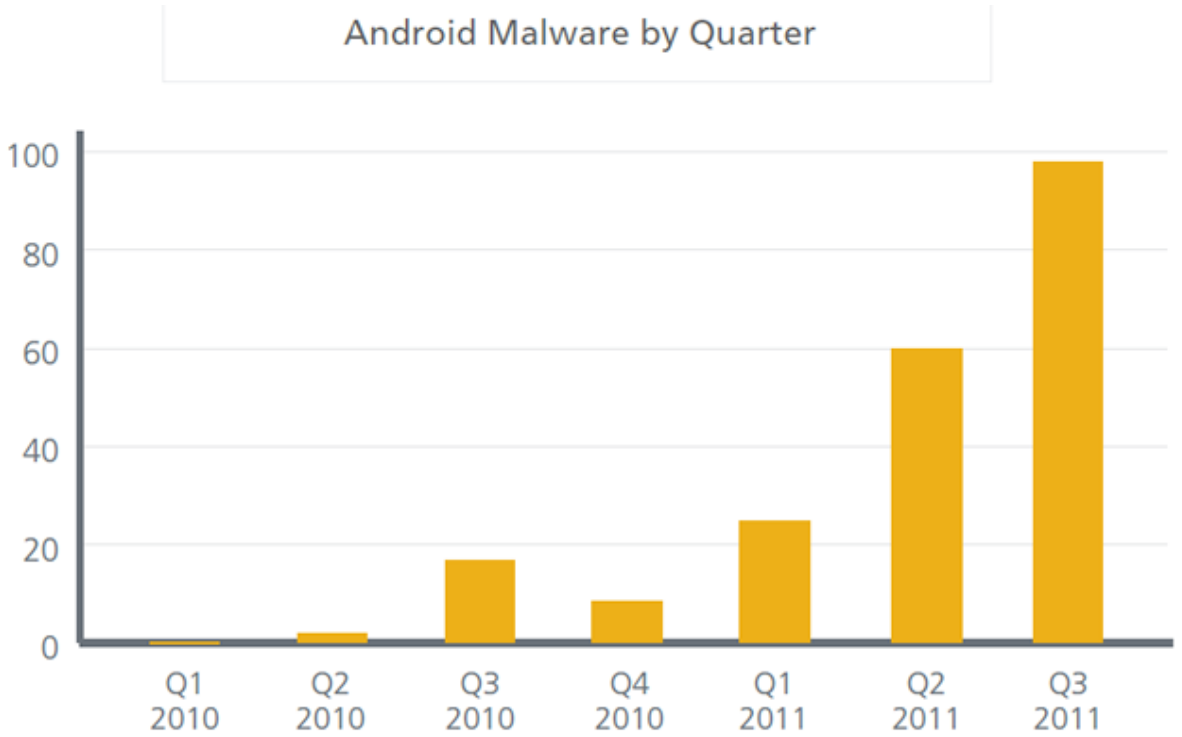
User-Centric Dependence Analysis For Identifying Malicious Mobile Apps

Karim Elish, Danfeng (Daphne) Yao, Barbara G. Ryder

Department of Computer Science
Virginia Tech

Mobile Security Technologies (MoST) Workshop, May 2012
In conjunction with IEEE Security and Privacy Symposium

Legitimate or Malicious: an app-classification problem



Problem: How to classify unknown apps as benign or malicious?

Source: http://news.cnet.com/8301-1009_3-57328575-83/androids-a-malware-magnet-saysmcafee/?tag=mncol;topStories

An Anomaly Detection Approach

Can one enforce properties of legitimate programs, as opposed to chasing malware patterns?

Challenge: what is norm?

Attack model: stealthy malware accesses system resources without user's awareness or consent

Our key observations:

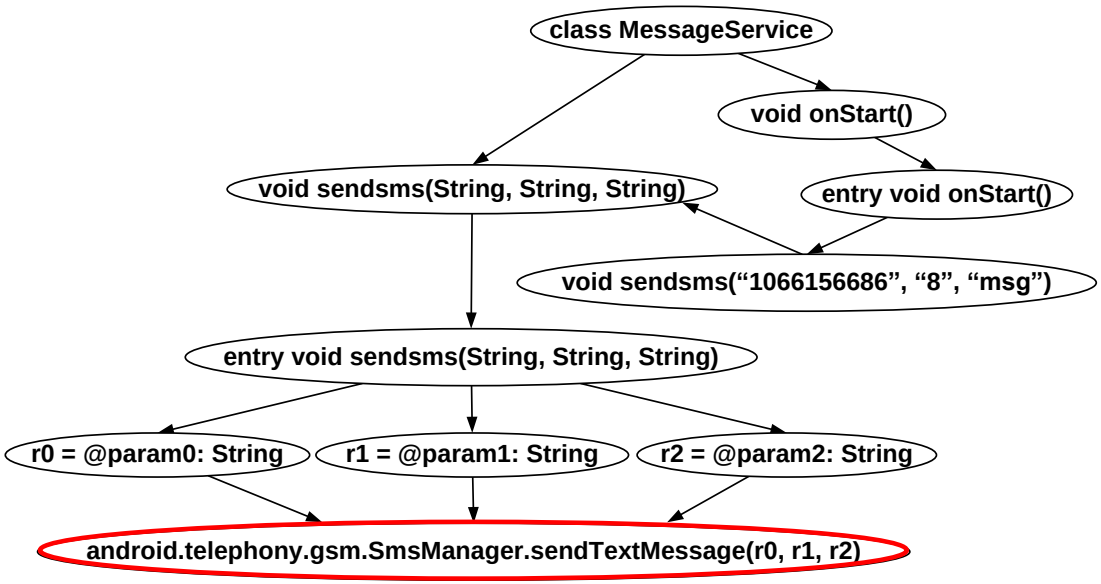
- legitimate critical system events are typically initiated by user inputs/actions
- Android mobile apps require a lot of user interactions

Our Goal:

Use static program analysis to track the dependence between the definition and use of user-generated data in programs

Example of Android Malware: HippoSMS

This malware sends SMS messages to a hard-coded premium-rated number without the user's awareness



A Data Dependence Graph

```
public class MessageService{
    .....
    public void onStart(){
        sendsms("1066156686", "8", "");
    }
    public void sendsms(param1, param2,
    param3){
        .....
        localSmsManager.sendTextMessage(
        param1, param2, param3);
    }
}
```

Malicious code

What is the norm? How to enforce it?

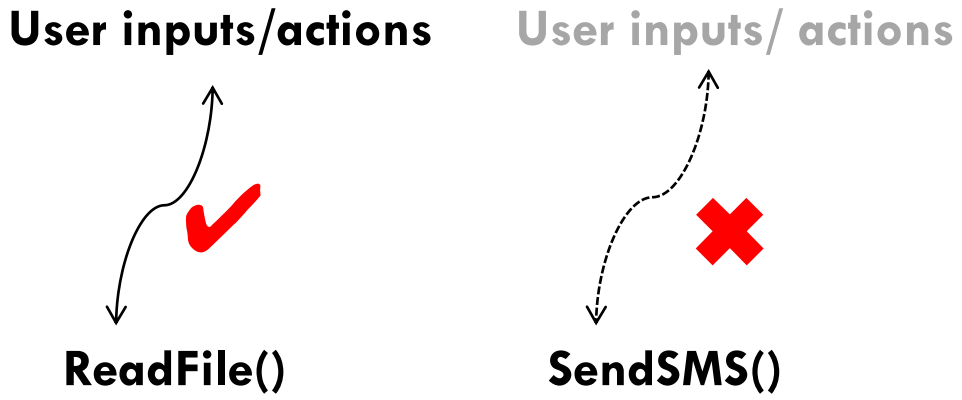
Requests to access system resources should be based on user inputs / actions

Our approach:

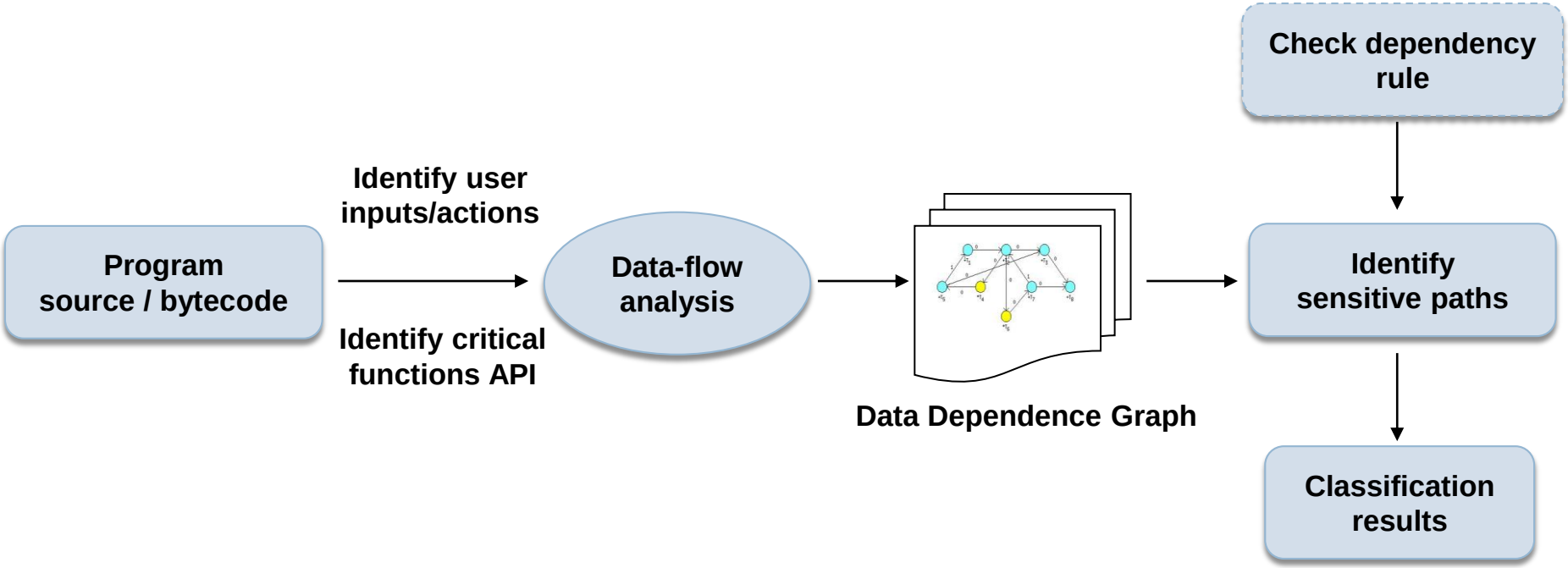
Identify the dependency relation between critical system events and user-initiated events in programs

Resources to protect from malicious programs:

- File system access
- Network access
- Sensitive/personal data



Our User-Centric Dependence Based Anomaly Detection Approach



Our Static Analysis Tool:

- We utilize def-use structures provided by Soot (a static analysis toolkit for Java)
- Inter-procedural and context-sensitive analysis

Soot Bytecode Optimization Framework

Evaluation Results on Legitimate and Malicious Android Apps

Most malware apps do not satisfy our data dependence requirement (FP = 0, FN = 1)

	App/Malware Name	# of User Inputs/ Actions (Source)	% of Sensitive Func. Calls without User Inputs	Library of Sensitive Function Calls
Legitimate	SendSMS	3	0%	android.telephony.gsm
	BMI Calculator	2	0%	android.app.Activity
	BluetoothChat	2	0%	java.io.OutputStream
	SendMail	4	0%	android.app.Activity
	Tip Calculator	4	0%	android.widget
Malicious	GGTracker.A	0	100%	org.apache.http.impl.client
	HippoSMS	0	100%	android.telephony.gsm android.content.ContentResolver
	Fakeneflic	3	0%	org.apache.http.impl.client
	GoldDream	0	100%	android.content.Context java.io.FileOutputStream
	Walk & Text	0	100%	android.content.ContentResolver org.apache.http.impl.client
	RogueSPPush	0	100%	android.telephony.gsm android.content.ContentResolver
	Dog Wars	0	100%	android.telephony.gsm android.content.ContentResolver

Security Analysis

Attacks	Countermeasures
phishing apps / social engineering apps	site authentication and user education
using superfluous user inputs and actions	easy to detect by using our approach to track the dependency
Code obfuscation or Java reflection	dynamic taint analysis

Related Work

- Static program analysis for malware detection
  - [Wagner '01, Bhatkar '06, …]
- Malware apps detection based on power consumption
  - [Liu '09, Dixon '11]
- Identify leakage of sensitive information
  - Panorama [Yin '07], TaintDroid [Enck '10]
    - Dynamic taint analysis
    - Paths between source and sink are expressed as violations

Conclusion and Future Work

- We proposed and implemented a promising approach for malware identification based on *user-centric data dependence analysis*
  - Our solution allows the detection of suspicious Android apps and Java programs
- Future Work
  - To enhance our analysis to handle inter-application Intents
  - To utilize dynamic taint analysis in addition to our current static program analysis
  - To extend our evaluation with more mobile apps including Android-based apps and Java ME-based apps

Acknowledgement

- This work has been supported in part by ***Security and Software Engineering Research Center (S²ERC)***, an NSF Industry/University Cooperative Research Center (I/UCRC)

Questions?

kelish@vt.edu