Detecting Potentially Malicious Behavior in Mobile Apps

with Static Code Analysis on Android OS

Pascal Gadient

AS2016 – Seminar Software Composition
University of Bern
1

Introduction
1.1 Introduction - Problem Statement

- Software gets more complex
  - Large Android OS fragmentation
  - Many security issues in media / web frameworks
  - Many security issues in HW drivers
  - Evolving «Freemium» apps

- «Malware» omnipresent on today's devices
  - User tracking
  - Collection of personal data
  - Advertisement SDKs
  - Exploitation of premium SMS / voice numbers
  - Phishing
1.1 Introduction - Problem Statement

⇒ We need to explore new approaches:

Static Code Analysis Feature Model Evaluation

(SCAFME)
1.2 Introduction - Concept

- Static Code Analysis
  - Applied on Android apps with FlowDroid\textsuperscript{[1]}

- Feature Model
  - Methods / Classes
  - Permissions
  - ...

- Evaluation
  - SVM training and application in R
2

Workflows
2.1 Workflows - Overview

❖ Conceptual view

APK → Data Extraction → Data Filtering → Data Analysis → Data Consolidation → Score

❖ Technical view

APK → Ubelix Cluster → Lua Scripting Engine → R Scripting Engine → Lua Scripting Engine → Score
2.2 Workflows - Virus Archives

- Encrypted, renamed and multi-platform
- We need pre-processing of:
  - ZIP headers (80 75 03 04)
  - ZIP integrity (TOC validity)
  - Folder structure (manifests)

- MALWARE IS DANGEROUS!

  ubuntuNBK-VIRUSSHARE-EDU:~$ lua apkMalwareFilter.lua
  Bus error (core dumped)
  ubuntuNBK-VIRUSSHARE-EDU:~$
2.3 Workflows - Data Extraction

- System overview

- Quirks
  - Job management
  - Linux environment (Quotas / Software)
2.4 Workflows - Data Filtering

- Log parsing
  - Feature extraction
  - Feature selection
  - Data conversion into ORCA\textsuperscript{[2]} format
  - ORCA outlier removal

- CSV file creation
2.5 Workflows - Data Analysis

- Analysis runs in R
  - Creation / execution of R scripts

- SVM algorithm from library "e1071"
  - Two configurations used[3]
2.6 Workflows - Data Consolidation

- Parsing of R results
  - Weighting of results
  - Calculation of final scores
3

Measures
3.1 Measures -
Android Package Files

- AndroidManifest.xml
  - Activity classes
  - Receiver classes
  - Services classes
  - App permissions

- apktool.yml
  - Required SDK version
  - Targeted SDK version
    *(needs backwards compatibility in code)*
3.2 Measures - Flows

❖ Call flow sequence diagram

Source Class : Source Method
PhoneManager : getDeviceId()

... ...

Sink Class : Sink Method
URLSocket : sendData()
3.3 Measures - SuSi Categories

- **SuSi**\(^4\)  
  - Software developed by Steven Arzt et al.  
  - Categorization of flows  
  - Supervised machine learning approach  
  - Currently 31 categories available

- **Examples**  
  - `UNIQUE_IDENTIFIER`, `NETWORK_INFORMATION`, `SMS_MMS`, `EMAIL`, `BLUETOOTH_INFORMATION`, `NFC`, ...
4

Evaluation
4.1 Evaluation - Computational Complexity

- Static analysis suffers combinatorial explosion
- Testbed configuration:
  - 8 CPU cores (x64)
  - 80 GB RAM
  - 3 hours runtime

- Still insufficient memory and CPU

  # There is insufficient memory for the Java Runtime Environment to continue.
  # Native memory allocation (malloc) failed to allocate 4088 bytes for AllocateHeap
  # Possible reasons:
  #   The system is out of physical RAM or swap space
  #   In 32 bit mode, the process size limit was hit
4.2 Evaluation - Data Set

- **Ubelix**
  - Data generated ourselves
  - Includes 182 benign apps
  - Includes 1,131 malign apps

- **MudFlow**
  - Data used from existing data set
  - Includes 2,800 benign apps
  - Includes 15,097 malign apps
4.3 Evaluation - Data Set Quality

- **Ubelix**
  - High-quality settings
  - No speed hacks
  - Slow (> 3 hours per file)

- **MudFlow**
  - Low-quality settings
  - Application of speed hacks
  - Fast (~ 15 minutes per file)
4.4 Evaluation - SVM Parameters

- **Method "eps-regression"**
  - Traditional *regression* based model
  - Determined best epsilon by 10-fold crossvalidation
  - Benign apps as training set (supervised)
  - Predicts each SuSi category count
  - Comparison of measured and predicted value

- **Method "one-classifier"**
  - Traditional *classification* model for novelty detection
  - Settings from MudFlow
  - Benign apps as training set (supervised)
  - Classifies measured values into "similar" or "not similar"
5

Results
5.1 Results - Potential malicious behavior #1

![Graph showing similarity: SuSi Categories]
5.2 Results - Potential malicious behavior #2

Similarity: SuSi Categories (Subset)

- benign training set vs. benign test set
- benign training set vs. malign test set

Diagram showing prediction discrepancies for apps tested.
5.3 Results - Potential malicious behavior #3

Similarity: Flows Within SuSi Categories

- benign training set vs. benign test set
- benign training set vs. malign test set
6

Conclusions
6.1 Conclusions - Real-World Applications

- App verification in app stores
- Behavioral anti-malware rankings
- Malware evolution analysis
- Malware trend prediction
6.2 Conclusions - Future Work

- Work on current analysis
  - Optimization of feature selection / SVM parameters
  - More comprehensive source / sink lists
  - Speed improvements
  - Much more training data

- Work on conceptual level
  - Evaluation of text description features
  - Evaluation of in-app-string features
  - Adaption to other platforms (Apple iOS, ...)
  - Integration of dynamic (hybrid) analysis models
References
7.1 References

[1] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden
http://dx.doi.org/10.1145/2594291.2594299

"Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule", Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003
http://stephenbay.net/papers/outliers.kdd03.pdf

"Mining Apps for Abnormal Usage of Sensitive Data", 2015

[4] Steven Arzt, Siegfried Rasthofer, and Eric Bodden
Questions?