GRACE HOPPER CELEBRATION

ANITA B.ORG

Leveraging Software Instrumentation for Android Security Assessment

Aisha Ali-Gombe, Ph.D.
Towson University
Louisiana State University

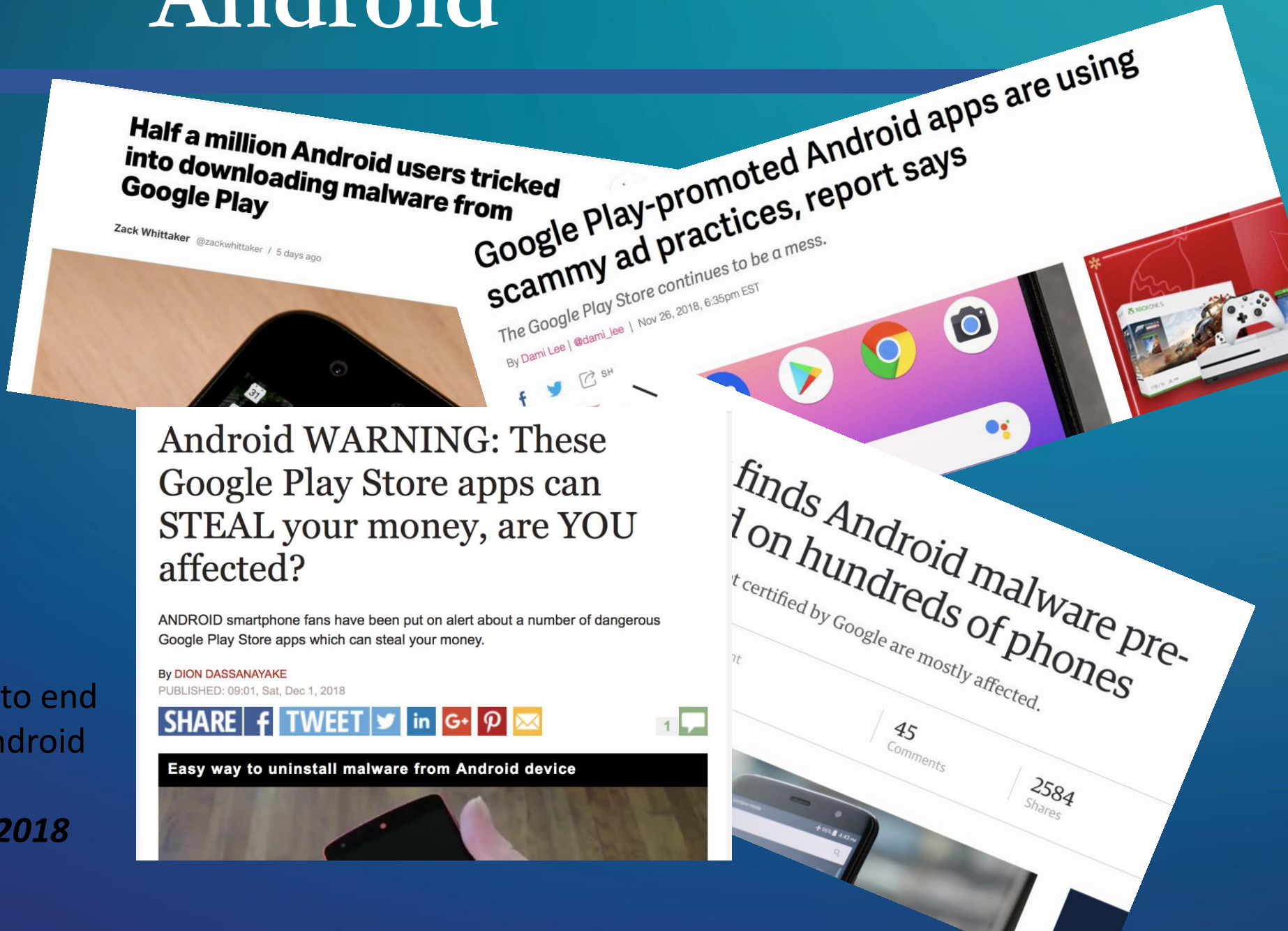
Outline

- Android Security
- Proposed App Vetting Technique
- Limitations and Current Research

Android

86.1% of all smartphones sold to end users were phones with the Android operating system.

- ***Statista 2018***



Android Security Challenges

- Challenges faced by Android - ***Applications***

Malware
Privacy-agnostic

- Spying on users and their communications
- Privilege escalation
- Resource abuse
- Data exfiltration
- Botnet
- ….

- 97% of all malware on mobile devices in 2014
- Over 2 million trojan applications have been detected in 2015 up by 50% from 2014

Malware Analysis Techniques

- Static Analysis/Fingerprinting
- Dynamic Analysis

Static Analysis

- Static analysis – examining program code base and its meta data
  - Permissions - Kirin (2009)
  - API calls, strings, and resources - Apposcopy (2014), DroidLegacy (2014), DroidAnalytics (2013)
  - Instruction sequences - DroidMoss(2012), Juxtapp (2012)

- Identifier transformation
- Encryption
- Dynamic class loading
- Java reflection
- Code reordering
- ……

Drawback – Obfuscation

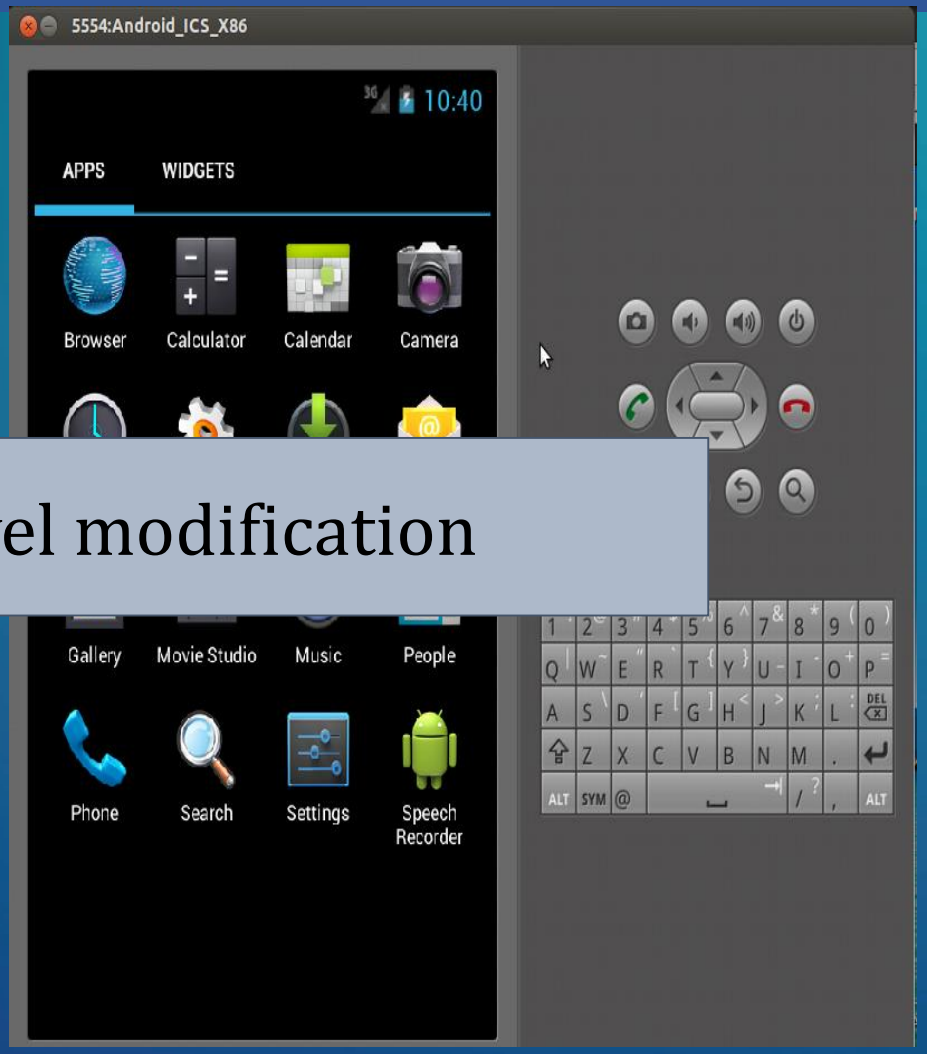
Rastogi et al. 2013 and Zhen et al. 2013

.....

Dynamic Analysis

- Dynamic Analysis
  - Modifying OS kernel/framework
  - Runtime dependent
  - OS dependent
  - Building a custom device is a painful process thus most runtime monitoring relies on emulation

Drawback - Low-level modification



Malware can detect emulation– *Vidas et al, 2014*

Research Question

How can we vet Android applications for unwanted (malicious) functionality **dynamically** on **real devices** without the need for **low-level modification** OS/Framework?

Software Instrumentation

- Analyzing programs by adding monitoring code:
  - execution environment
  - compiled code
  - source code

Normal program execution

```
public class test{
 public static void main(){
   int x=1;
   int y =2;
   int z = add(x,y);
   System.out.println(z);
 }
}
```

Program execution with instrumentation

```
public class test{
 public static void main(){
   int x=1;
   int y =2;
   int z = add(x,y);
```

hooking

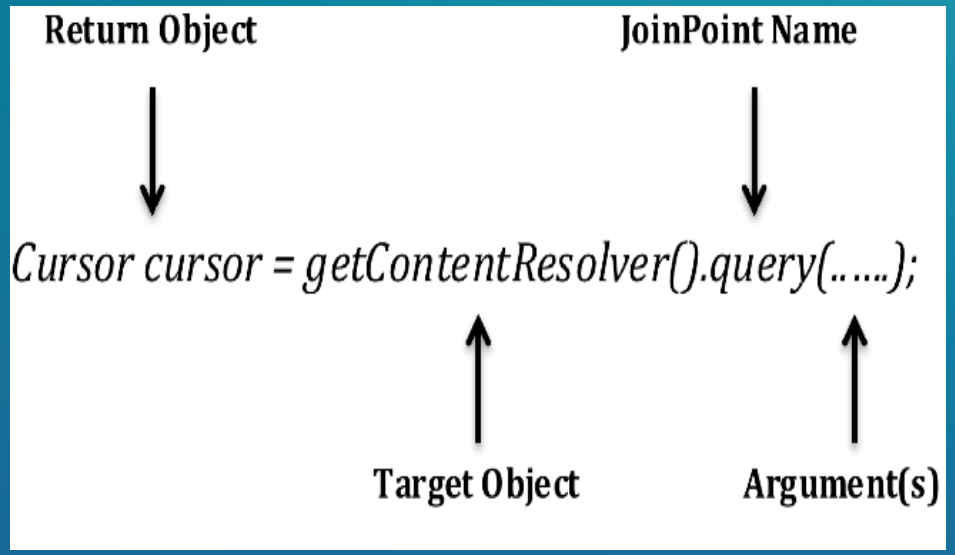
```
class inst{
  int routine(){
   log x;
   log y;
  }
}
```

```
   System.out.println(z);
 }
}
```

Monitoring code

Aspect-oriented Programming

- AspectJ - Bytecode Weaving
  - Aspect
  - Joinpoint
  - Advice (before, after and around)




```
pointcut getCurObj(Uri uri, String[] Projection,
      String Selection, String[] Selection_Args):
      call(*..*Cursor* *..*.query(..))
      || call(*..*Cursor* *..*.*Query(..)))
      && args(uri, Projection, Selection,
      Selection_Args,..) && NotNewLogger();
```

```
Object around(Object tar, Uri uri, String[] Projection,
      String Selection, String[] Selection_Args):
      target(tar) && getCurObj(uri, Projection,
      Selection, Selection_Args){
      //...
      //...
```

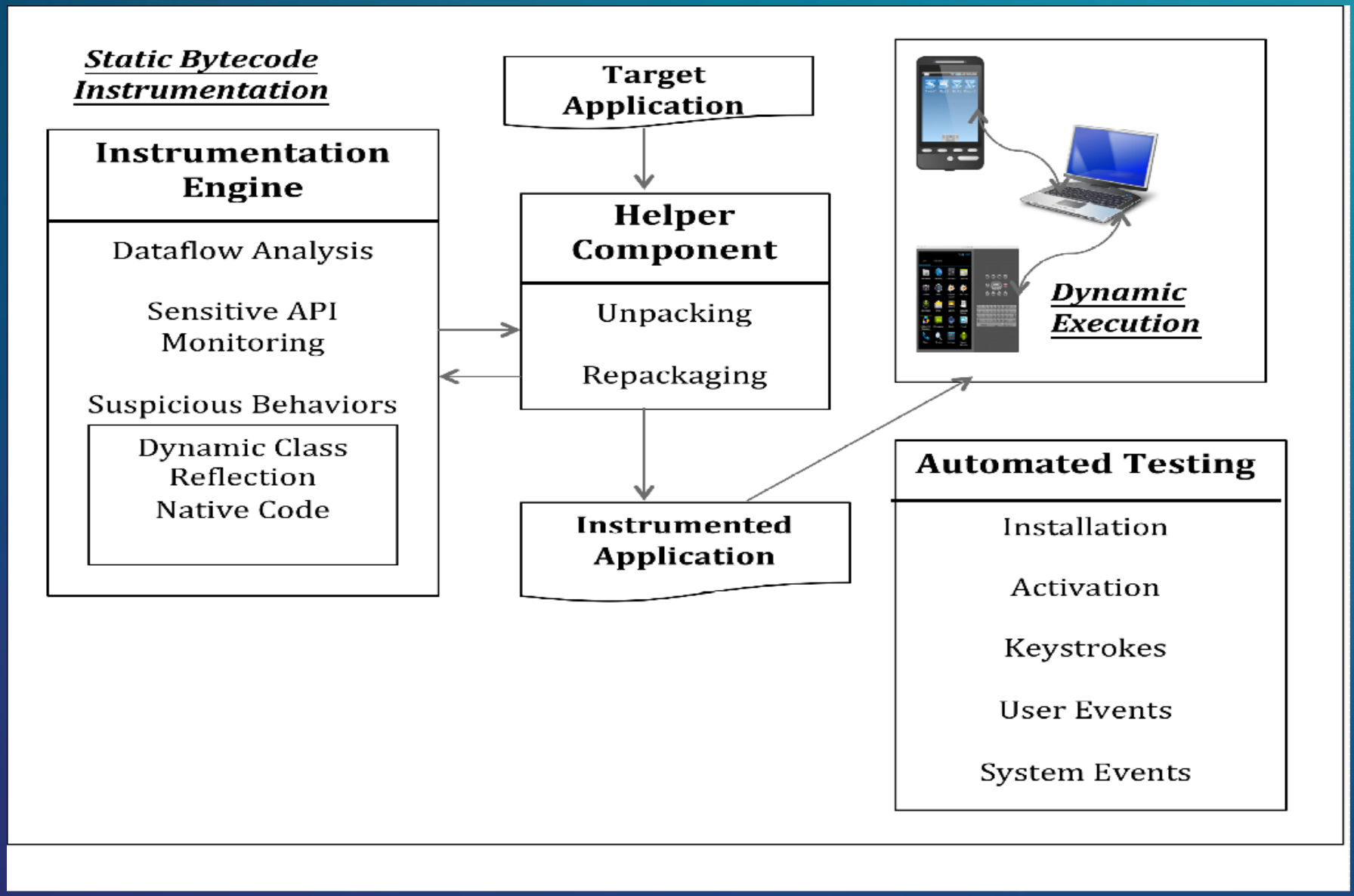
Outline

- Android Security
- Proposed App Vetting Technique
- Limitations and Current Research

AspectDroid - Hybrid Analysis System

- Methodology
  - develops aspects to detect explicit data exfiltration, sensitive API monitoring (resource abuse) and analytics of suspicious behaviors
  - software instrumentation via static bytecode weaving
  - dynamic monitoring
- Objectives
  - analyze unknown Android application without need for custom kernel
  - adaptable to all Android runtimes
  - low performance overhead

AspectDroid - System Design

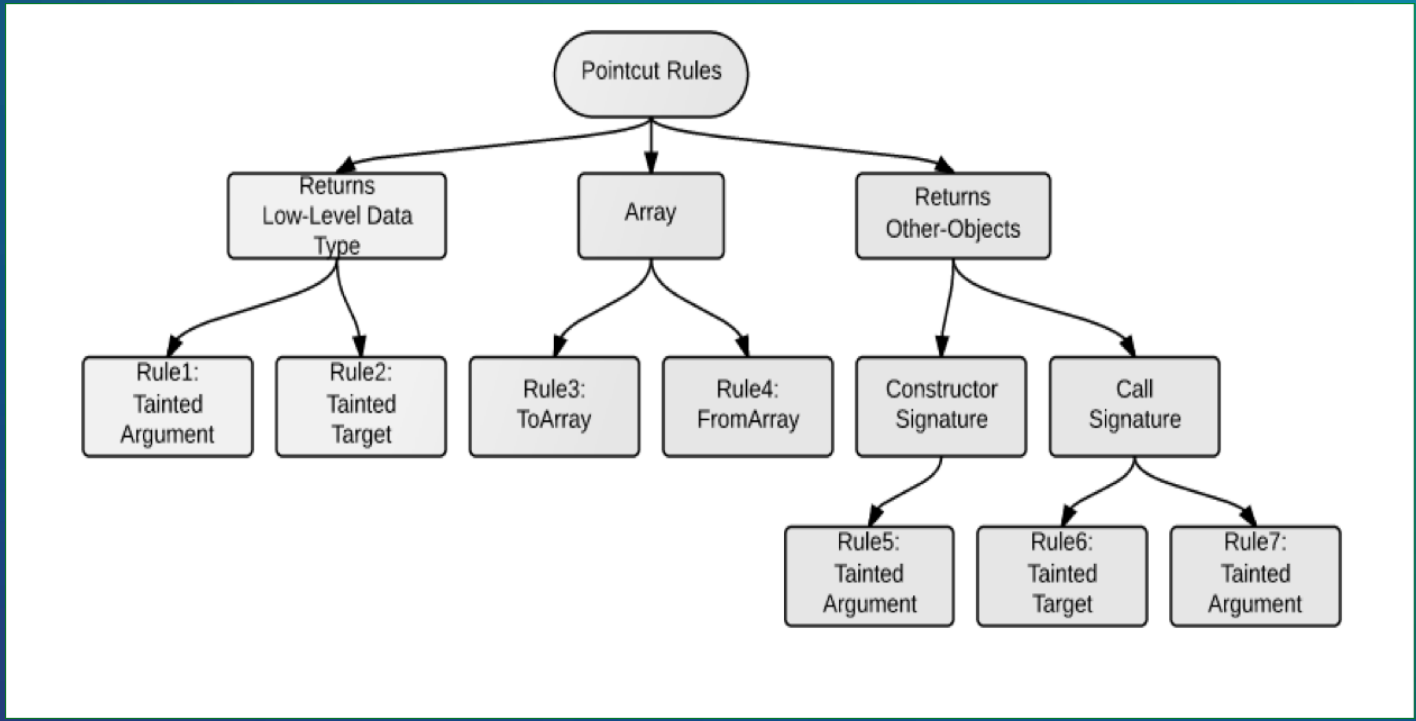


Dataflow Analysis

Taint Sources

(27 data sources e.g. IMEI, SMS)

Taint Propagation Rules



Taint Sink

(4 data sinks e.g. network)

Tracing

- Sensitive API Monitoring
  - Resource abuse tracing
    - SMS send, receive and read
    - Call send, receive and intercept
    - Access to content providers

    trace and trigger log

- Analytics of suspicious behaviors
  - Dynamic class loading

  Dynamic class call → halt execution → find class path → pull class to analysis machine → trigger instrumentation → push class to path → log → continue execution
  - Native code execution & Reflection

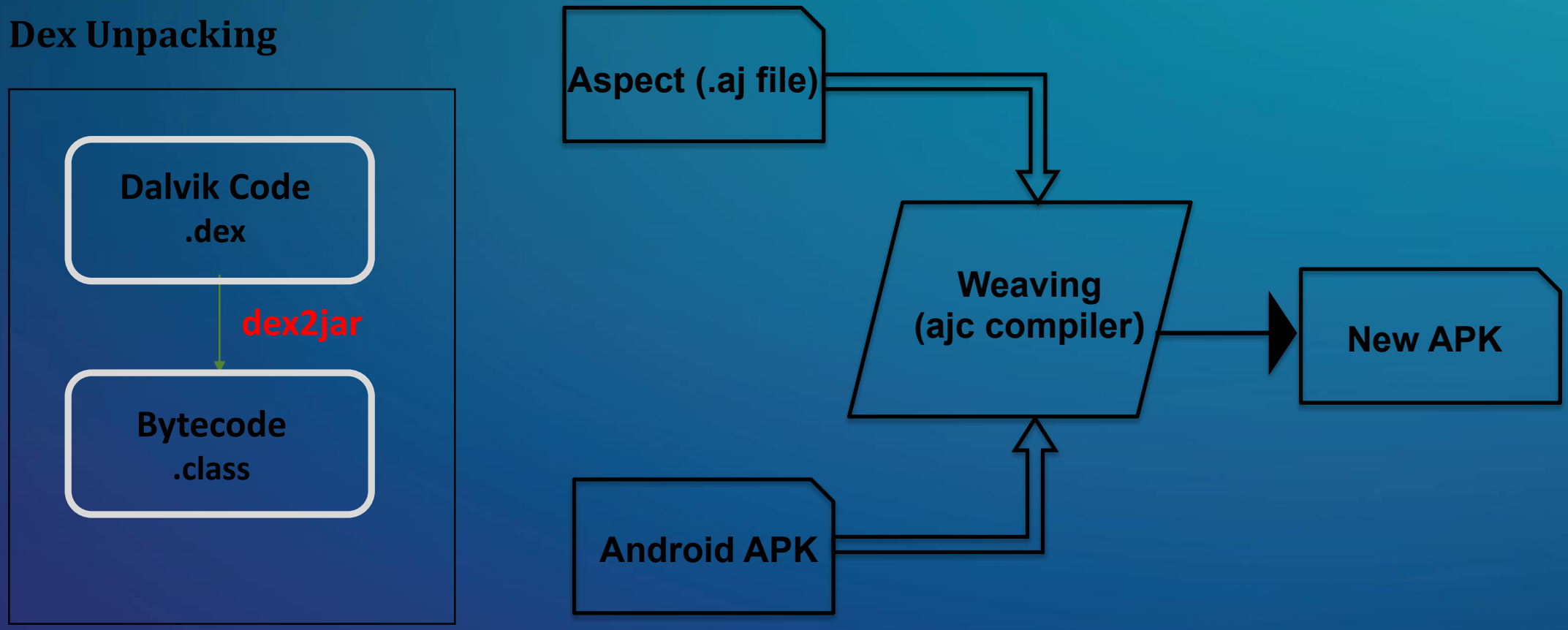
    trace and trigger log

```
pointcut getReflect(): !within(Logger) && call(*
    java.lang.reflect.Method.invoke(..));

Object around(): getReflect(){
   Object [] params = thisJoinPoint.getArgs();
   Object tar = thisJoinPoint.getTarget();
   logVals(tar, params); //Convert the objects to
       human readable form and log
}
```

Helper Component

- Repackaging an Android application



Evaluation

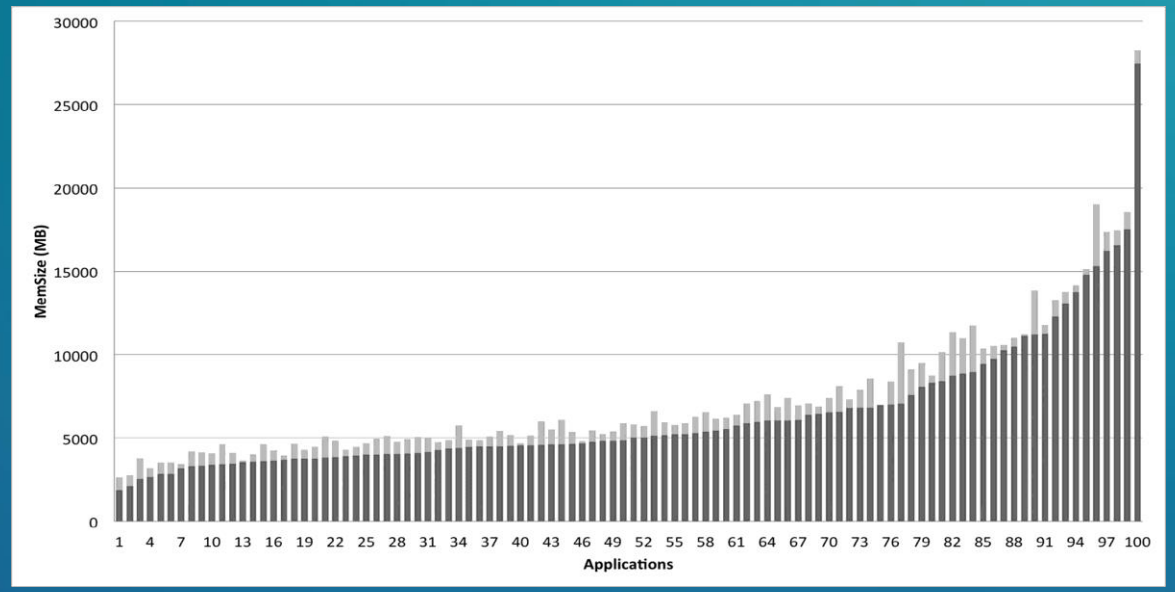
- Setup – HTC-One S9 (6.0), Samsung (4.4) & emulator
- 100 malware and 100 benign samples
- Runtime overhead
  - Memory size overhead ~ 1MB

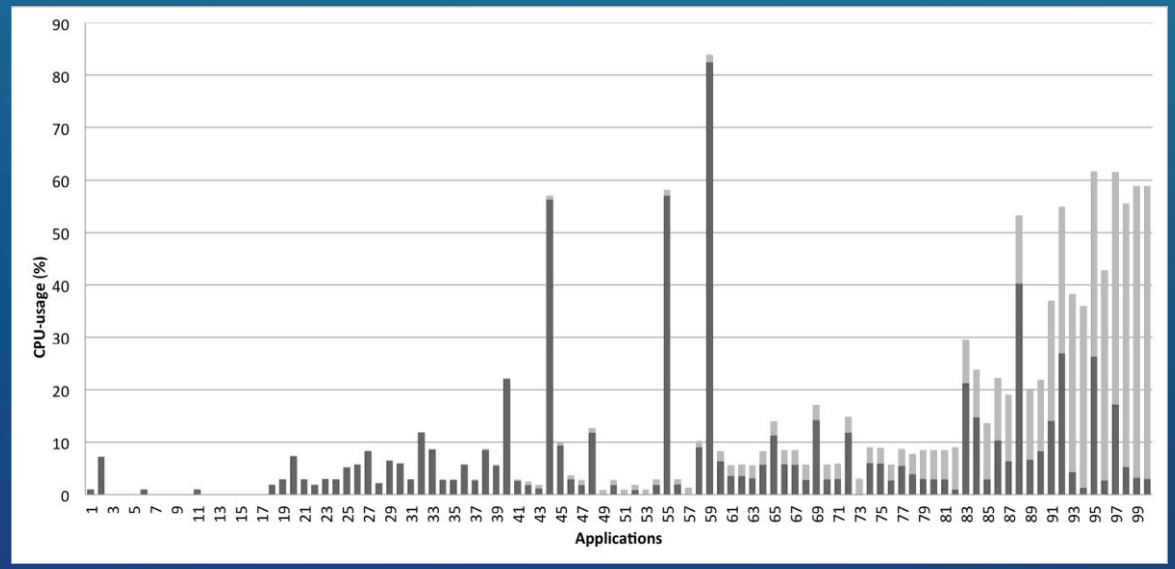
  *Memory size before and after instrumentation using **procrank** utility*
  - CPU usage ~ 5.91% increase

  $seconds = upTime - (startTime\ Hertz)$

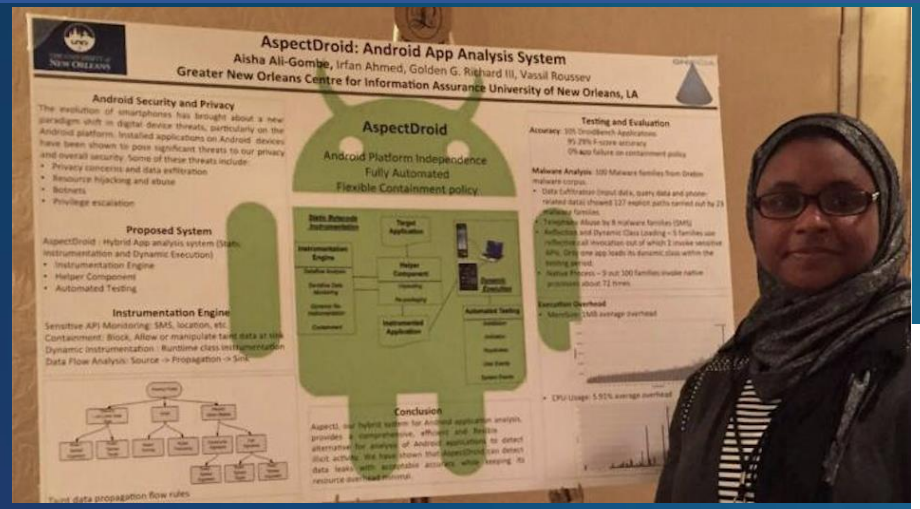
  $tTime = uTime + sTime + CuTime + csTime$

  $cpuUsage = ((tTime\ Hertz)\ seconds) * 100$
- Accuracy of Dataflow
  - DroidBench 2.0 – 105 simulated apps
  - Accuracy 94.68%, Precision - 96.4%, Recall - 93.02%





Preliminary Work & Award



Outstanding Poster Award @ ACM CODASPY 2016

AspectDroid: Android App Analysis System

Aisha Ali-Gombe
aaligomb@uno.edu

Irfan Ahmed
irfan@cs.uno.edu

Golden G. Richard III
golden@cs.uno.edu

Vassil Roussev
vassil@cs.uno.edu

Dept. of Computer Science
University of New Orleans
New Orleans LA 70148

ABSTRACT

The growing threat to user privacy related to Android applications (apps) has tremendously increased the need for more reliable and accessible app analysis systems. This paper presents AspectDroid, an application-level system designed to investigate Android applications for possible unwanted activities. AspectDroid is comprised of app instrumentation, automated testing and containment systems. By using static

their consent. Andrubis [3] performed an analysis on over a million malicious and benign apps, and found that 38.79% of the apps have various forms data leakage. The security and privacy concerns surrounding these revelations increases the need for reliable and accessible app analysis systems.

In this paper, we present `AspectDroid`, a dynamic analysis system for Android applications based on the AspectJ instrumentation framework. AspectDroid performs static

Research Paper – COSE Journal

COMPUTERS & SECURITY 73 (2018) 235–248

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Toward a more dependable hybrid analysis of android malware using aspect-oriented programming

Aisha I. Ali-Gombe [a,*], *Brendan Saltaformaggio* [b], *J. "Ram" Ramanujam* [c], *Dongyan Xu* [d], *Golden G. Richard III* [c]

[a] *Department of Computer and Information Science, Towson University, RM 447, 7800 York Road, Towson, MD 21252, USA*
[b] *School of Electrical and Computer Engineering, Georgia Institute of Technology, Klaus Advanced Computing Building, 266 Ferst Dr NW, Atlanta GA 30332, USA*
[c] *Center for Computation and Technology, Louisiana State University, 2027-C Digital Media Center, Baton Rouge, LA 70803, USA*
[d] *Department of Computer Science, Purdue University, 305 N. University Street, West Lafayette, IN 47907, USA*

ARTICLE INFO

Article history:
Received 24 March 2017
Received in revised form 2 November 2017

ABSTRACT

The growing threat to user privacy by Android applications (app) has tremendously increased the need for more reliable and accessible analysis techniques. This paper presents *AspectDroid*[1]—an offline app-level hybrid analysis system designed to investigate Android applications for possible unwanted activities. It leverages static bytecode instrumentation

Outline

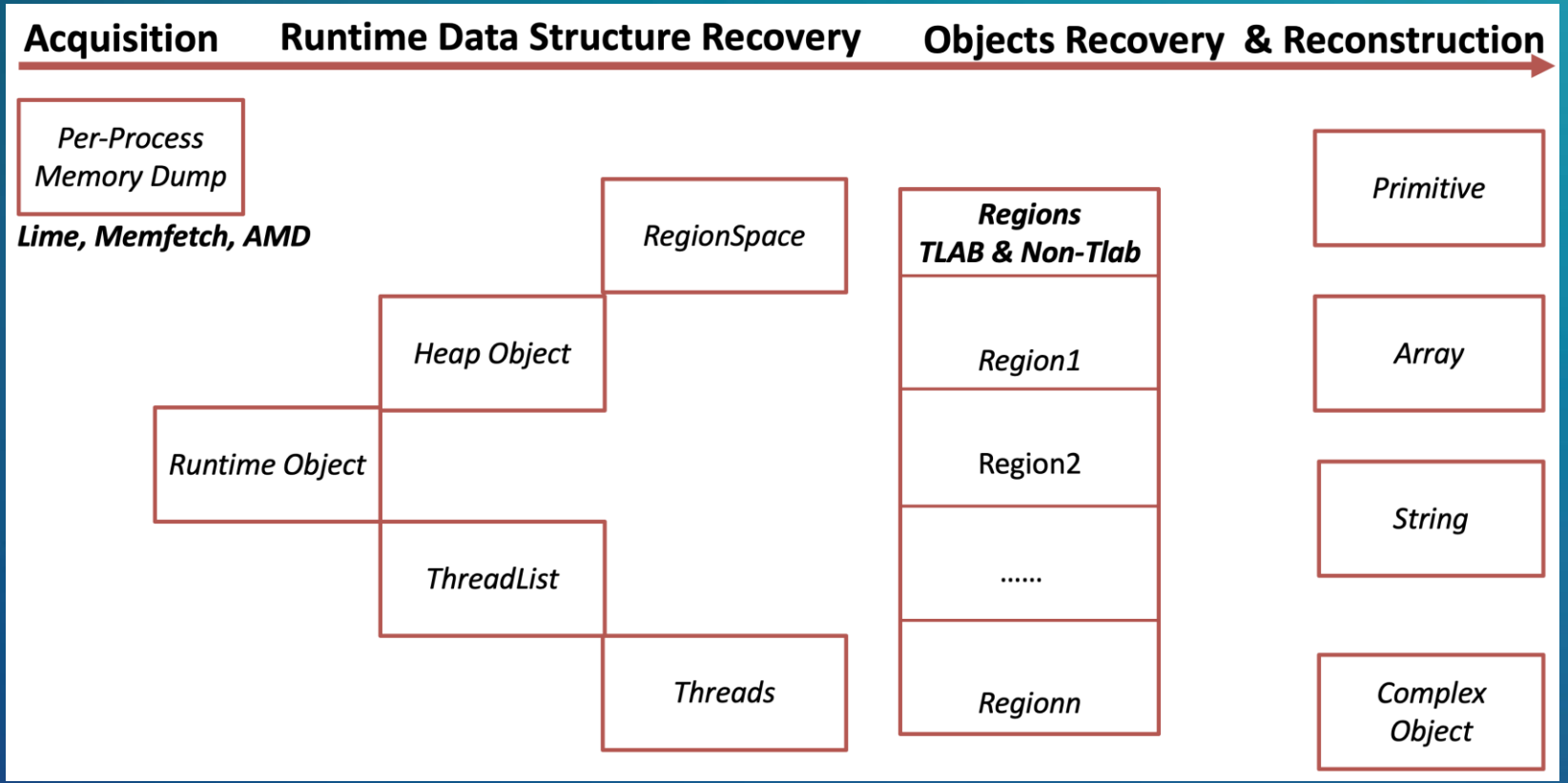
- Android Security
- Proposed App Vetting Technique
- Limitations and Current Research

Limitations

- Signature verification that detects changes in the developer's original signature at runtime
- Anti-repackaging techniques that detect and crash Dalvik to Java bytecode backward translation process at recompilation time
- Native code monitoring

Userland Memory Forensics

- DroidScraper



- Presentation and Publication – RAID 2019
  - https://github.com/apphackuno/DroidScraper.git
- NSF CRII Grant 2019-2021

Please remember to complete the session survey in the mobile app.

THANK YOU

YOU CAN *FOLLOW ME*

@aishagombe, #DFIR

aaligombe@towson.edu

GRACE HOPPER CELEBRATION

ANITA B.ORG

#GHC19