Master's thesis
Predictive Identification of Android Malware through Hybrid Analysis

Author: Johannes Thon

Technische Universität Berlin, Fakultät IV Elektrotechnik und Informatik

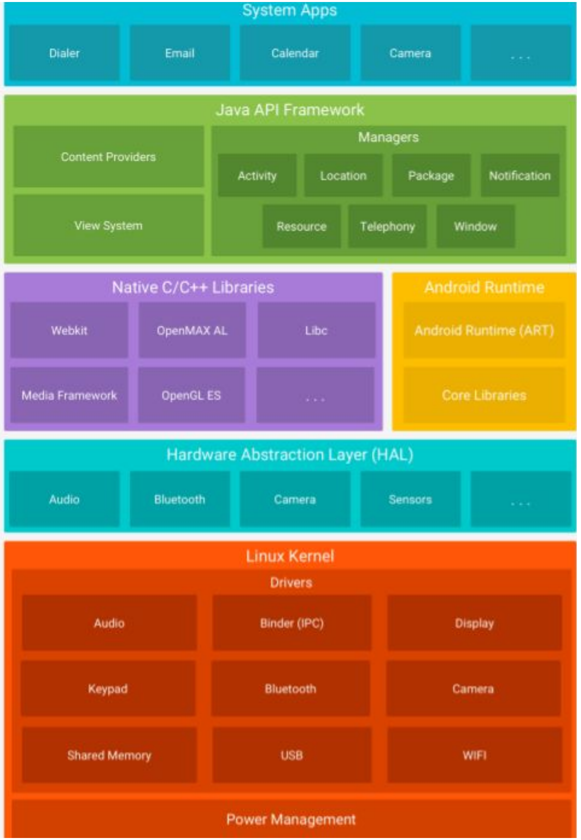
Presented by: Dilver Huertas Guerrero
Master's student in systems and computer engineering
Universidad Nacional de Colombia

UNSECURELAB

Thesis structure

- Introduction
- Background
- Problem Description
- Concept
- Data Retrieval
- Hybrid Analysis
- Machine Learning
- Evaluation
- Conclusion and Outlook

The Android Stack



System Apps

Dialer | Email | Calendar | Camera | . . .

Java API Framework

Content Providers

View System

Managers

Activity | Location | Package | Notification

Resource | Telephony | Window

Native C/C++ Libraries

Webkit | OpenMAX AL | Libc

Media Framework | OpenGL ES | . . .

Android Runtime

Android Runtime (ART)

Core Libraries

Hardware Abstraction Layer (HAL)

Audio | Bluetooth | Camera | Sensors | . . .

Linux Kernel

Drivers

Audio | Binder (IPC) | Display

Keypad | Bluetooth | Camera

Shared Memory | USB | WIFI

Power Management

Tools

elasticsearch

scikit learn

SciPy

Problem description

Central problem

Many scientific publications, as well as more recent ones, utilize outdated malware in supervised machine learning approaches, in order to predict the detection of untrained Android malware.

Goals

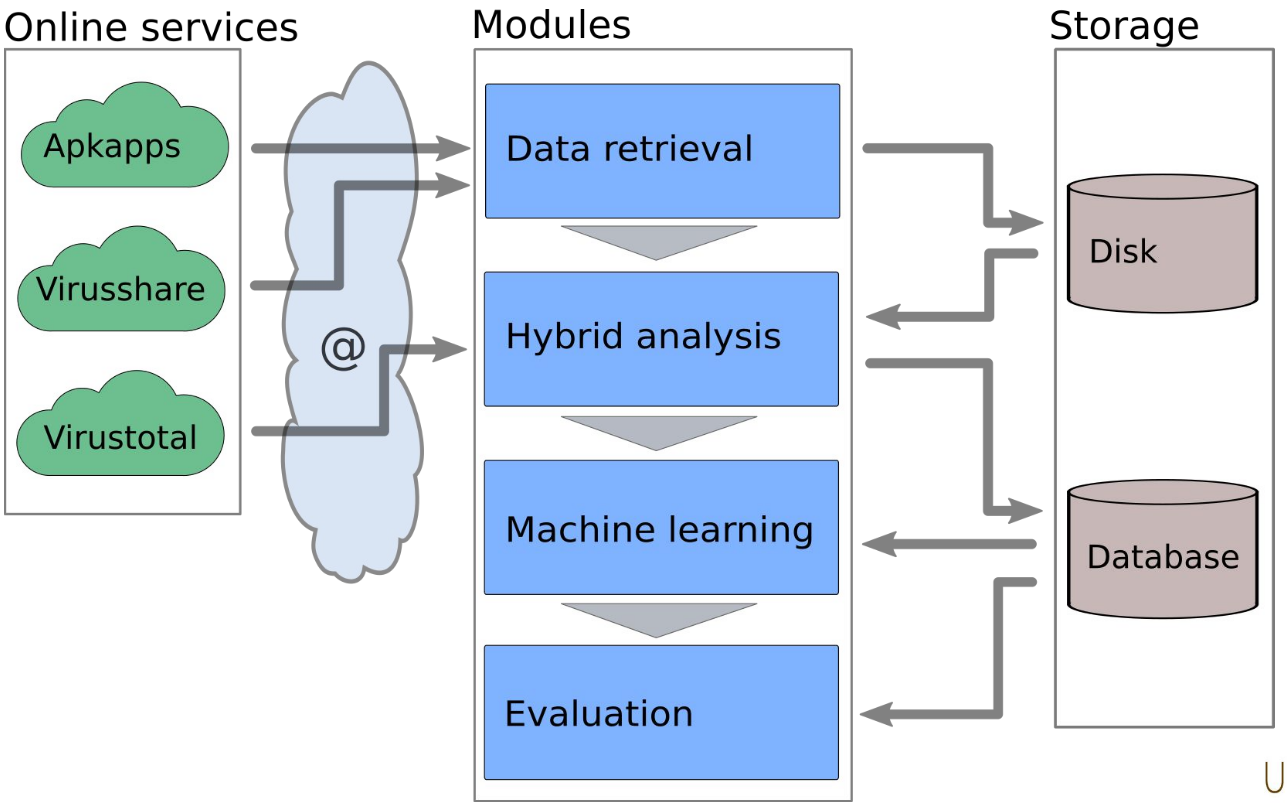
- Build up a collection of recent malicious and benign apps of 2017
- Implement a hybrid analysis prototype
- Realize two different dynamic analysis approaches

Central research question

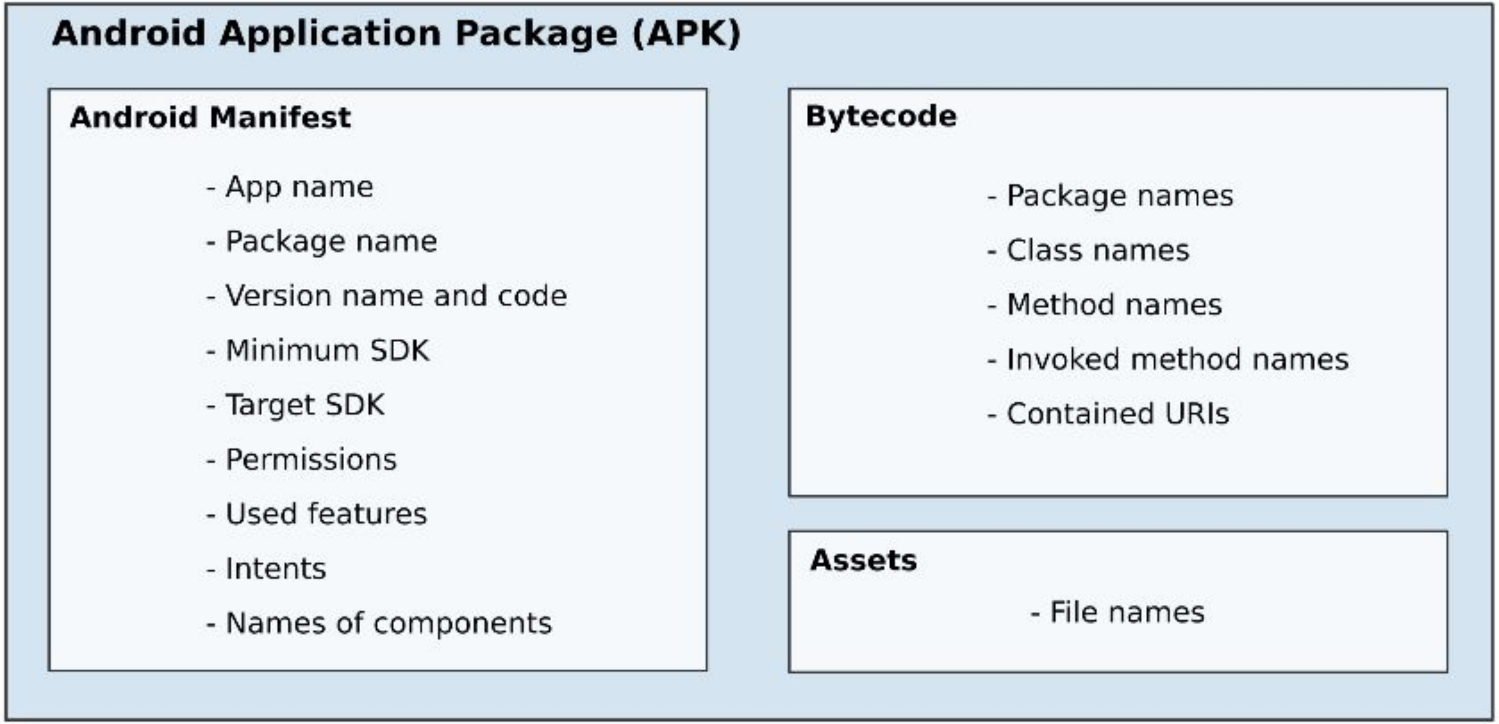
How precise is the predictive identification of recent Android malware by comparing two different hybrid analysis approaches?

Concept

Concept overview



Static analysis



Android Application Package (APK)

Android Manifest

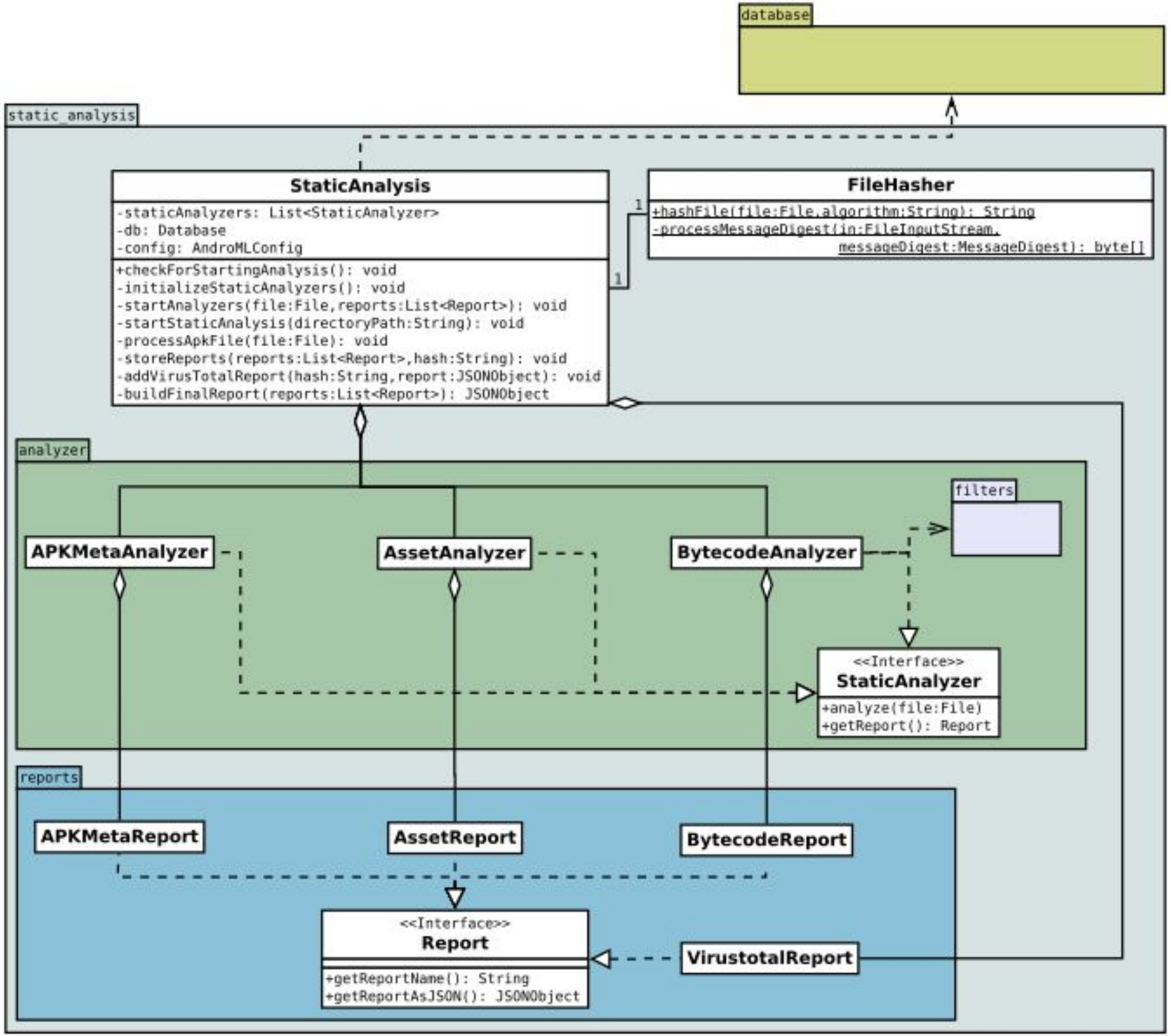
- App name
- Package name
- Version name and code
- Minimum SDK
- Target SDK
- Permissions
- Used features
- Intents
- Names of components

Bytecode

- Package names
- Class names
- Method names
- Invoked method names
- Contained URIs

Assets

- File names



database

static_analysis

StaticAnalysis

-staticAnalyzers: List<StaticAnalyzer>
-db: Database
-config: AndroMLConfig

+checkForStartingAnalysis(): void
-initializeStaticAnalyzers(): void
-startAnalyzers(file:File,reports:List<Report>): void
-startStaticAnalysis(directoryPath:String): void
-processApkFile(file:File): void
-storeReports(reports:List<Report>,hash:String): void
-addVirusTotalReport(hash:String,report:JSONObject): void
-buildFinalReport(reports:List<Report>): JSONObject

1

1

FileHasher

+hashFile(file:File,algorithm:String): String
-processMessageDigest(in:FileInputStream,
messageDigest:MessageDigest): byte[]

analyzer

APKMetaAnalyzer

AssetAnalyzer

BytecodeAnalyzer

filters

<<Interface>>
StaticAnalyzer

+analyze(file:File)
+getReport(): Report

reports

APKMetaReport

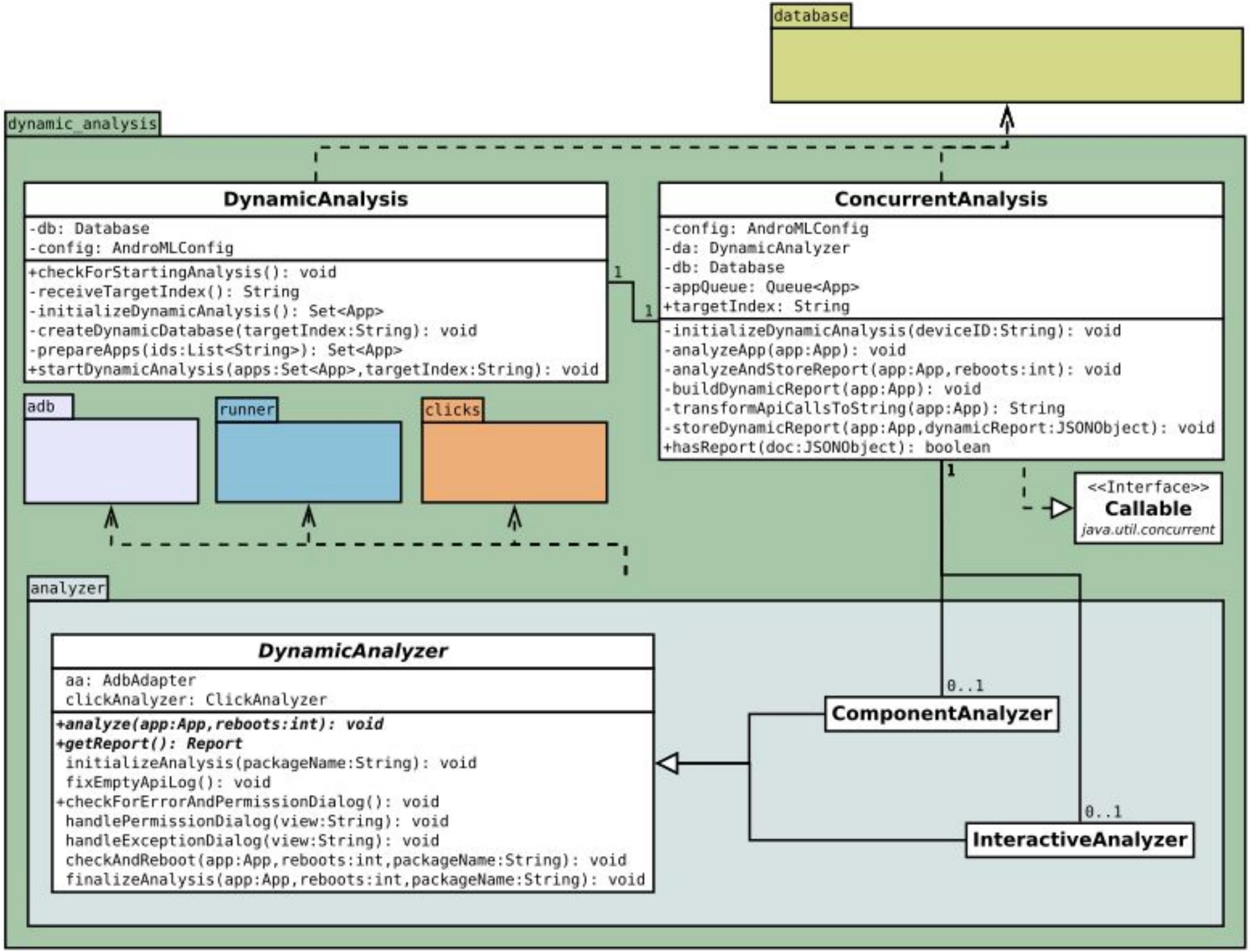
AssetReport

BytecodeReport

<<Interface>>
Report

+getReportName(): String
+getReportAsJSON(): JSONObject

VirustotalReport



DynamicAnalysis

-db: Database
-config: AndroMLConfig

+checkForStartingAnalysis(): void
-receiveTargetIndex(): String
-initializeDynamicAnalysis(): Set<App>
-createDynamicDatabase(targetIndex:String): void
-prepareApps(ids:List<String>): Set<App>
+startDynamicAnalysis(apps:Set<App>,targetIndex:String): void

ConcurrentAnalysis

-config: AndroMLConfig
-da: DynamicAnalyzer
-db: Database
-appQueue: Queue<App>
+targetIndex: String

-initializeDynamicAnalysis(deviceID:String): void
-analyzeApp(app:App): void
-analyzeAndStoreReport(app:App,reboots:int): void
-buildDynamicReport(app:App): void
-transformApiCallsToString(app:App): String
-storeDynamicReport(app:App,dynamicReport:JSONObject): void
+hasReport(doc:JSONObject): boolean

<<Interface>>
Callable
java.util.concurrent

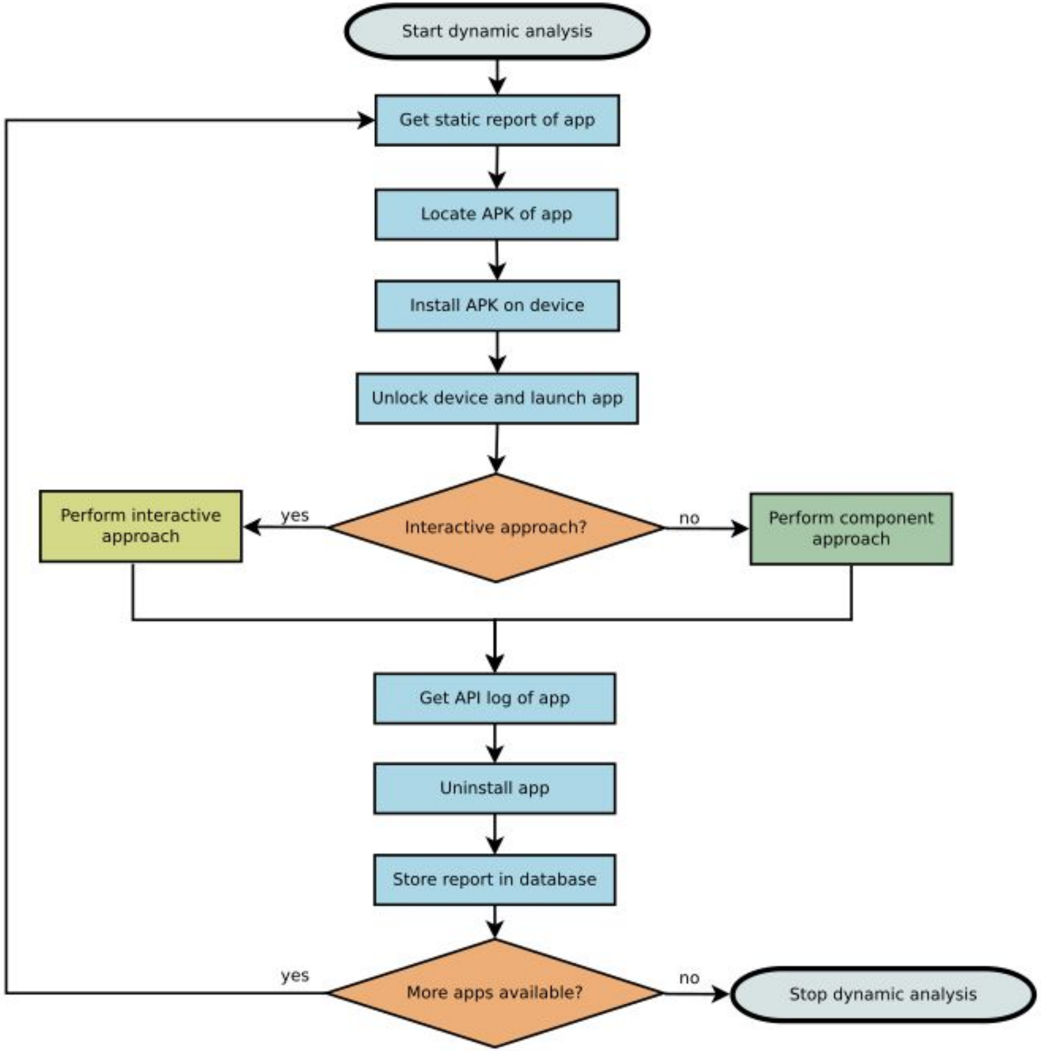
DynamicAnalyzer

aa: AdbAdapter
clickAnalyzer: ClickAnalyzer

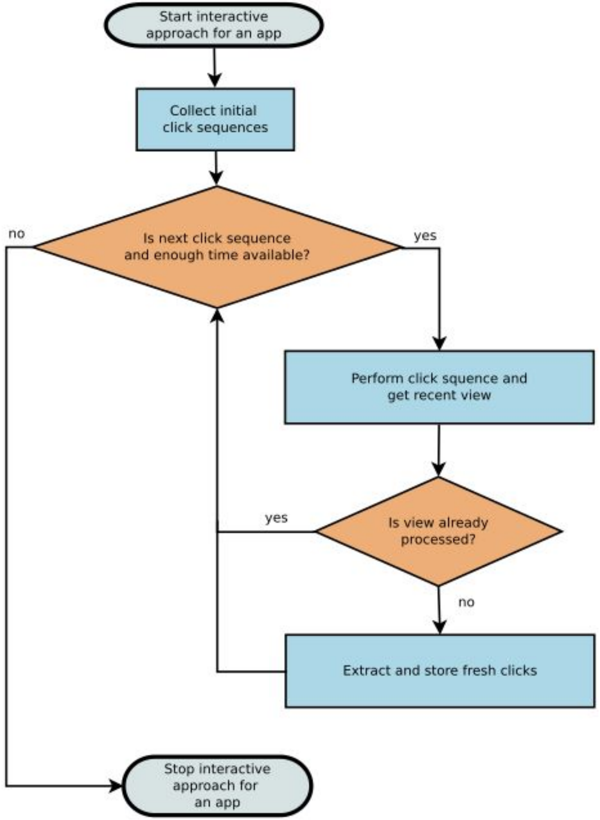
+analyze(app:App,reboots:int): void
+getReport(): Report
initializeAnalysis(packageName:String): void
fixEmptyApiLog(): void
+checkForErrorAndPermissionDialog(): void
handlePermissionDialog(view:String): void
handleExceptionDialog(view:String): void
checkAndReboot(app:App,reboots:int,packageName:String): void
finalizeAnalysis(app:App,reboots:int,packageName:String): void

ComponentAnalyzer

InteractiveAnalyzer

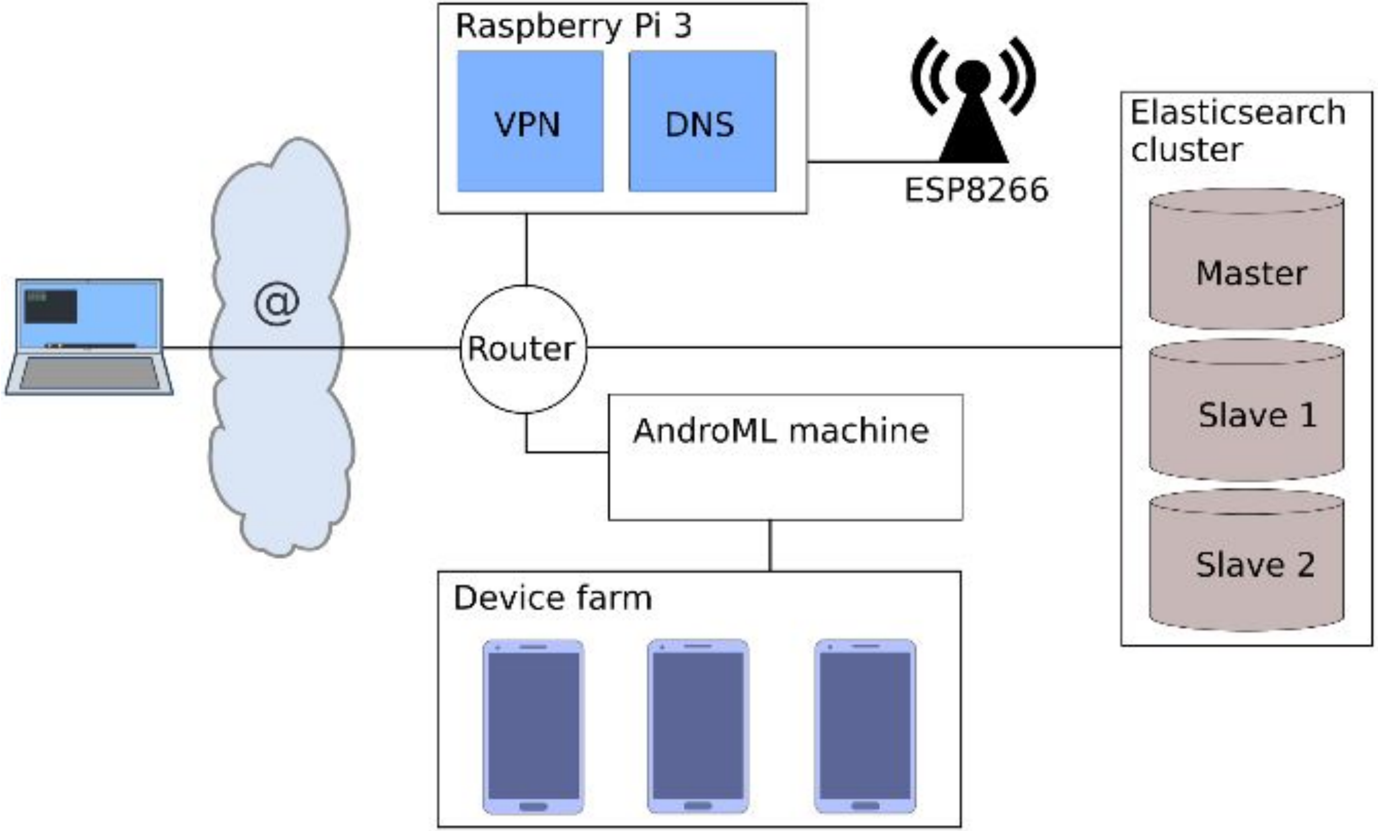

Start dynamic analysis
Get static report of app
Locate APK of app
Install APK on device
Unlock device and launch app
Interactive approach?
yes
Perform interactive approach
no
Perform component approach
Get API log of app
Uninstall app
Store report in database
More apps available?
yes
no
Stop dynamic analysis

Interactive analysis program flow


Start interactive approach for an app
Collect initial click sequences
Is next click sequence and enough time available?
no
yes
Perform click squence and get recent view
Is view already processed?
yes
no
Extract and store fresh clicks
Stop interactive approach for an app

UNSECURELAB

Environment overview



UNSECURELAB

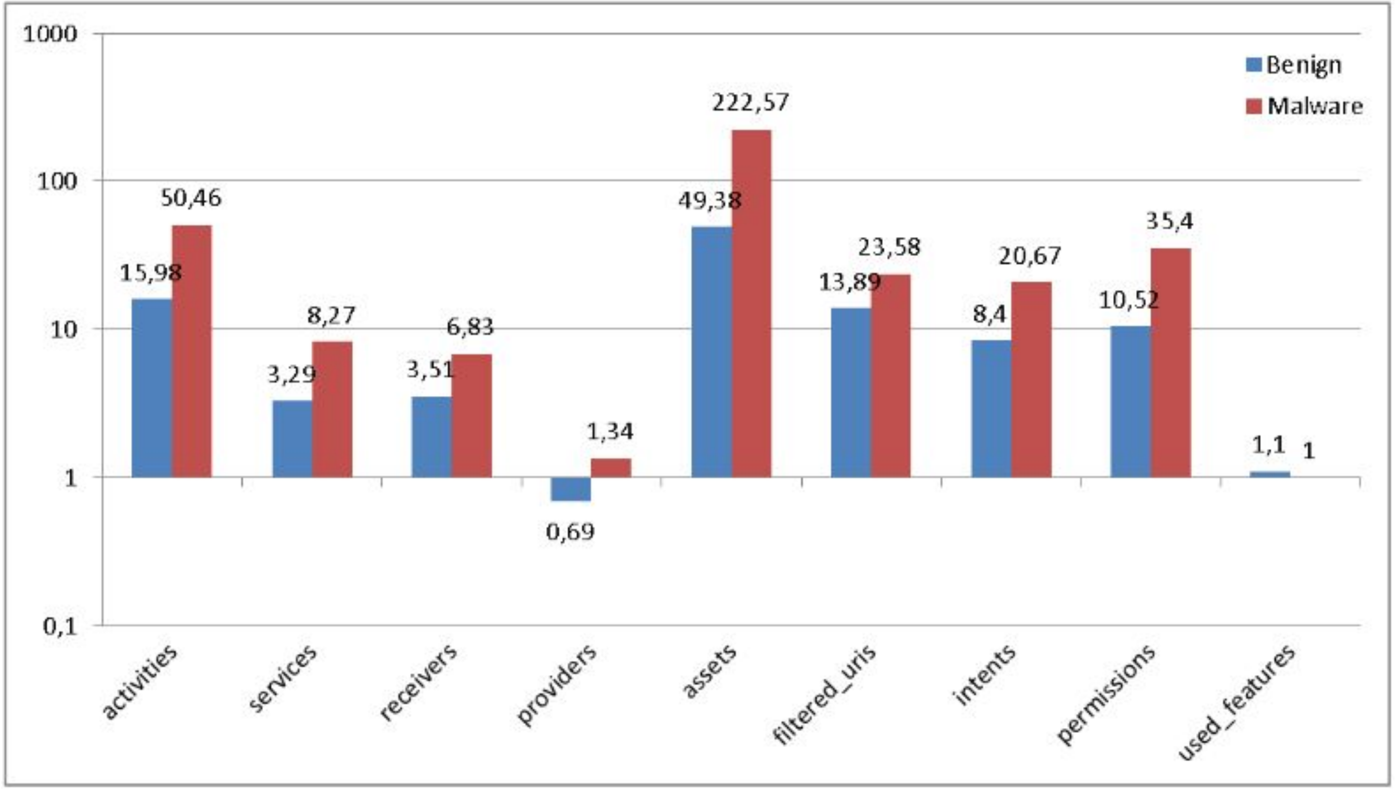
Results

Previous to the analysis

- A set of 4054 malicious and 5151 benign apps were analyzed
- The static analysis for 9205 apps took 40 hours in total
- The component analysis took around two weeks
- The interactive analysis for 9205 apps took in total around 40 days

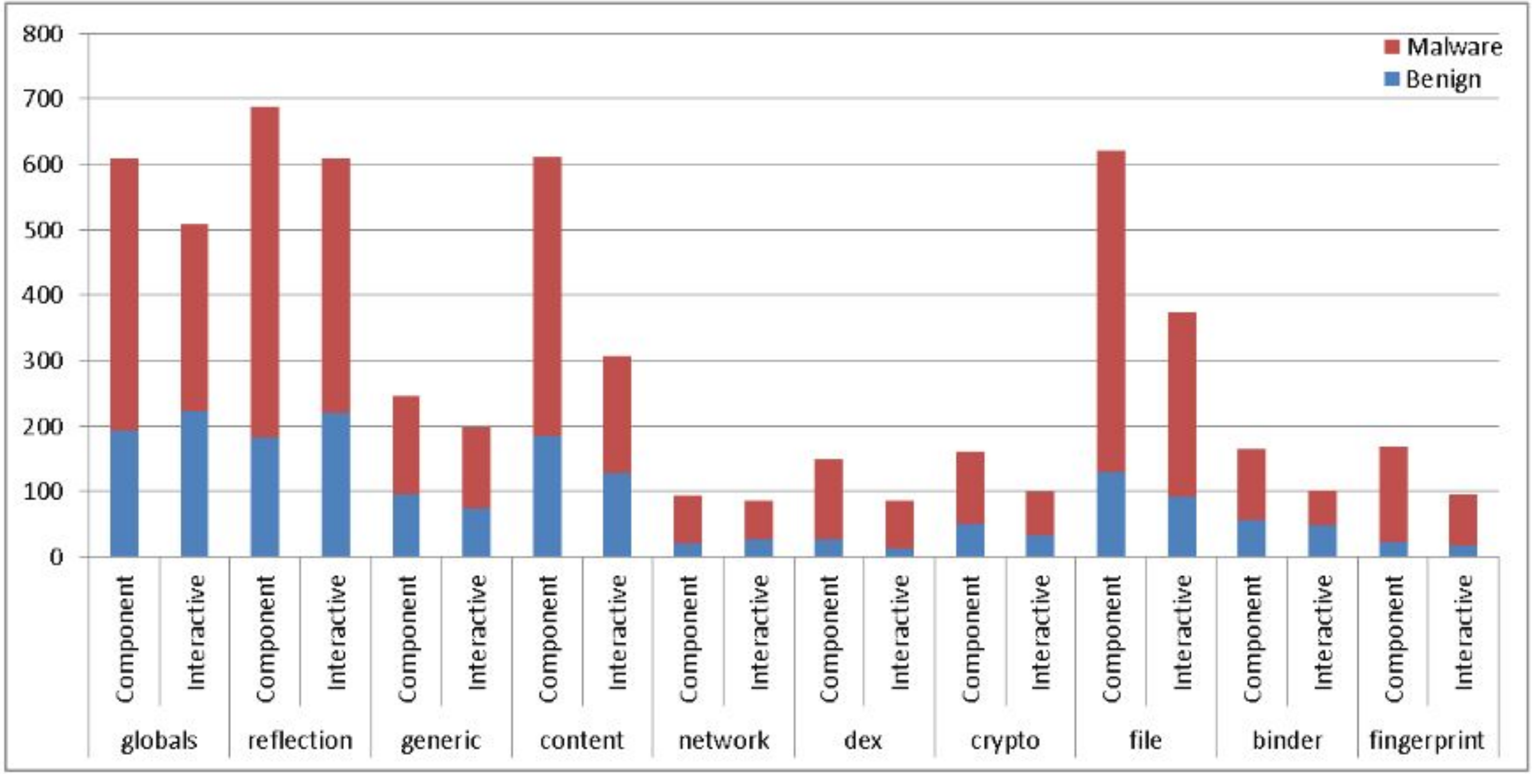
UNSECURELAB

Average amounts of collected static data



UNSECURELAB

Average API calls per category



Main clusters

Amount of apps	Keywords
163	skymobi, smspay
1754	trojan, pup, generic
106	smsreg, emagsoftware, dinehu
108	kuguo, dowgin, addisplay
397	smsreg, riskware, risktool
147	secapk, pup, pua
390	dropper, ztorg, blouns

Answer to the central research question

The predictive identification for detecting recent Android malware lies at around 90% detection accuracy. Both analysis approaches do not differ noteworthy for the inspected scenario.

Thank you!

Questions?

Dilver Huertas Guerrero
@dilverhuertas
djhuertasg@unal.edu.co