Are Free Android App Security Analysis Tools Effective in Detecting Known Vulnerabilities?

Venkatesh-Prasad Ranganath **Joydeep Mitra**

Department of Computer Science
Kansas State University, USA

34th IEEE/ACM International Conference on Automated Software Engineering (ASE 2019)
San Diego, USA
November 11-15, 2019

Context

- Mobile app security is important
- Our Goal
  - Develop a methodology for designing secure apps
- Our First Step: Study vulnerabilities that plague Android apps
  - We started building Ghera
- Our Second Step: Understand the current landscape of security tools
  - How well do existing tools and techniques address Android app security?
  - Can we quantify the effectiveness of Android security analysis solutions?

Tool Effectiveness Evaluation Strategy

1. Select a set of Android App vulnerability benchmarks
2. Select a set of Android vulnerability detection tools
3. Measure the efficacy of each tool w.r.t the selected benchmarks

What is Ghera?

A repository of Android app vulnerability benchmarks

- Vulnerabilities were collected from various sources
  - CVE, documentation, source code
- Each benchmark is demonstrably authentic
  - A set of 3 apps: benign (vulnerable), malicious, and secure
- Benchmarks are spread across 7 different categories
  - Crypto, ICC, Networking, Permission, Storage, System, Web

Why use Ghera benchmarks?

Validity of benchmarks

- Captured vulnerabilities can be observed by executing the benchmarks

Exploitability of vulnerabilities

- Captured vulnerabilities can be used to inflict harm

Generality of vulnerabilities

- Captured vulnerabilities do not depend on uncommon constraints, e.g. rooted device

Representativeness of benchmarks?

Representativeness of vulnerability benchmark

The manifestation of a vulnerability in the benchmark should be similar to its manifestation in real-world apps.

Measuring representativeness == Compare benchmarks with real-world apps

- Consider producers and consumers of data
- Consider APIs involved in handling and processing data
- Consider data/control flow paths

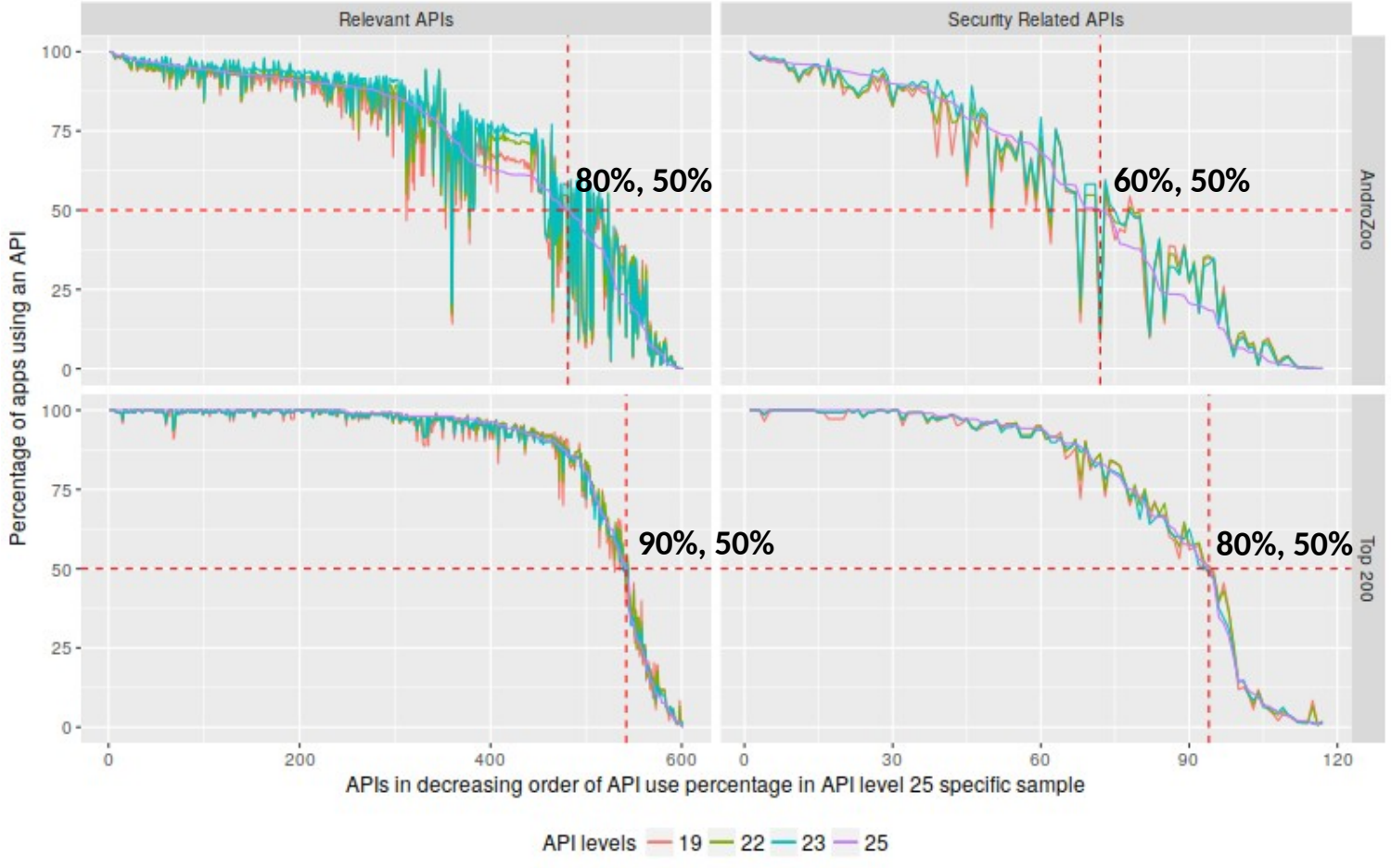
We used API usage as a metric for representativeness

- Extent to which APIs used in a benchmark are used in real-world apps

Representativeness Evaluation Methodology

1. Collected a sample of 111K real-world apps from AndroZoo
2. Collected top 200 apps from Google Play
3. Constructed API profiles for real-world apps and top 200 apps
   - XML elements and attributes used in the manifest of an app
   - Android APIs used and defined (callbacks) in an app
4. Collected APIs used in Ghera
5. Determined the % of real-world apps and top 200 apps that used an API used in Ghera benchmarks

Ghera Representativeness Results



Tool Effectiveness Evaluation Strategy

1. Select a set of Android App vulnerability benchmarks
2. Select a set of Android vulnerability detection tools
3. Measure the efficacy of each tool w.r.t the selected benchmarks

Tool selection

1. Started with 64 tools
2. Selected 23 of the 64 tools after *shallow pass*
   - Is the tool well documented?
   - Does the tool intend to detect vulnerabilities?
   - Can the tool be used off-the-shelf or with minimal configuration?
   - Is the tool freely available?
   - Is the tool applicable to Ghera benchmarks?
3. Selected 14 of the 23 tools after *deep pass*
   - Can the tool be built and setup with no or minimal effort?
   - Can the tool be executed on a sample of Ghera benchmarks?

Selected Tools

Tool	Crypto (4)	ICC (16)	Net (2)	Perm (1)	Store (6)	Sys (4)	Web (9)	S/D
Amandroid	4	16	1		5		4	S
AndroBugs*	4	16	2	1	6	4	9	S
AppCritique*	4	16	2	1	6	4	9	?
COVERT	4	16	1	1	6	4	1	S
DevKnox*	4	16	2	1	6	4	9	S
DIALDroid	4	16	1	1	6	4	1	S
FixDroid	4	3	2	1	4	4	9	S
FlowDroid		16	2		6			S
HornDroid	4	16	2		6		9	S
JAADS*	4	16	2	1	6	4	9	S
MalloDroid							4	S
Marvin-SA*	4	16	2	1	6	4	9	S
MobSF*	4	16	2	1	6	4	9	SD
QARK*	4	16	2	1	6	4	9	S

Ensuring Fair Evaluation

1. All minimal inputs required by a tool were provided
2. Every tool was evaluated against benchmarks/apps targeting the API levels supported by the tool
3. Every tool was evaluated against the category of vulnerabilities it was designed to detect
4. Every tool was evaluated against the vulnerabilities that were known before the tool was released
5. Tools that depend on certain versions of Android Studio were evaluated only against those versions

Year	# Vuln. found	# Tools released
2011	9	0
2012	13	0
2013	20	1
2015	39	3
2016	39	4
2017	40	9
2018	42	13

Tool Effectiveness Evaluation Strategy

1. Select a set of Android App vulnerability benchmarks
2. Select a set of Android vulnerability detection tools
3. Measure the efficacy of each tool w.r.t the selected benchmarks

Results

We made 20 observations

- 4 observations based only on the tool selection strategy
- 16 observations based of verdicts from tool execution

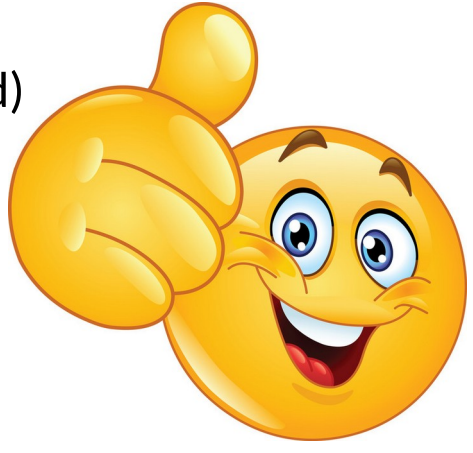
We identified 6 open questions for future exploration

- 2 questions based only on the tool selection strategy
- 4 questions based of verdicts from tool execution

Some Observations about the Current State

Tool developers are putting effort to release robust tools

- 16 of 23 tools were available as ready-to-use binaries
- 7 of 23 tools were built from source
- 21 tools were used with no additional configuration
- 1 tool was used with minimal configuration (DIALDroid)
- 1 tool was used with extensive configuration (CuckooDroid)



Some Observations about the Current State

All tools combined detect 30 of the 42 vulnerabilities captured in Ghera

No tool detects more than 15 of the 42 vulnerabilities in Ghera in isolation

Some Observations about the Current State

Shallow analysis tools report more true positives and less false positives than deep analysis tools

Non-academic tools detected more vulnerabilities than academic tools

Some Tool Developers Specific Observations

13 of the 14 tools were partially or not informed about the presence/absence of vulnerabilities (Informedness)

- Tools should consider 53 (18 security-related) APIs in the 12 undetected vulnerabilities. These APIs are used by more than 50% of real-world apps.

12 of the 14 tools provided verdicts that cannot be fully trusted (Markedness)

- Tools should do better in differentiating between vulnerable apps and secure apps as these apps often use the same set of APIs

Threats to Validity

1. Bias in tool selection
2. Mis-categorization of vulnerabilities
3. API usage is a weak measure of representativeness

Takeaways

Call for Action

Ghera benchmarks are representative of real-world apps (in terms of API usage)

- Evaluate the representativeness of benchmarks

All evaluated tools together detected only 30 of the 42 vulnerabilities captured in Ghera; no tool detected more than 15 vulnerabilities

- Continuously and rigorously evaluate Android security analysis tools
- Use Informedness and Markedness instead of Recall and Precision
- Maintain and extend benchmark suites