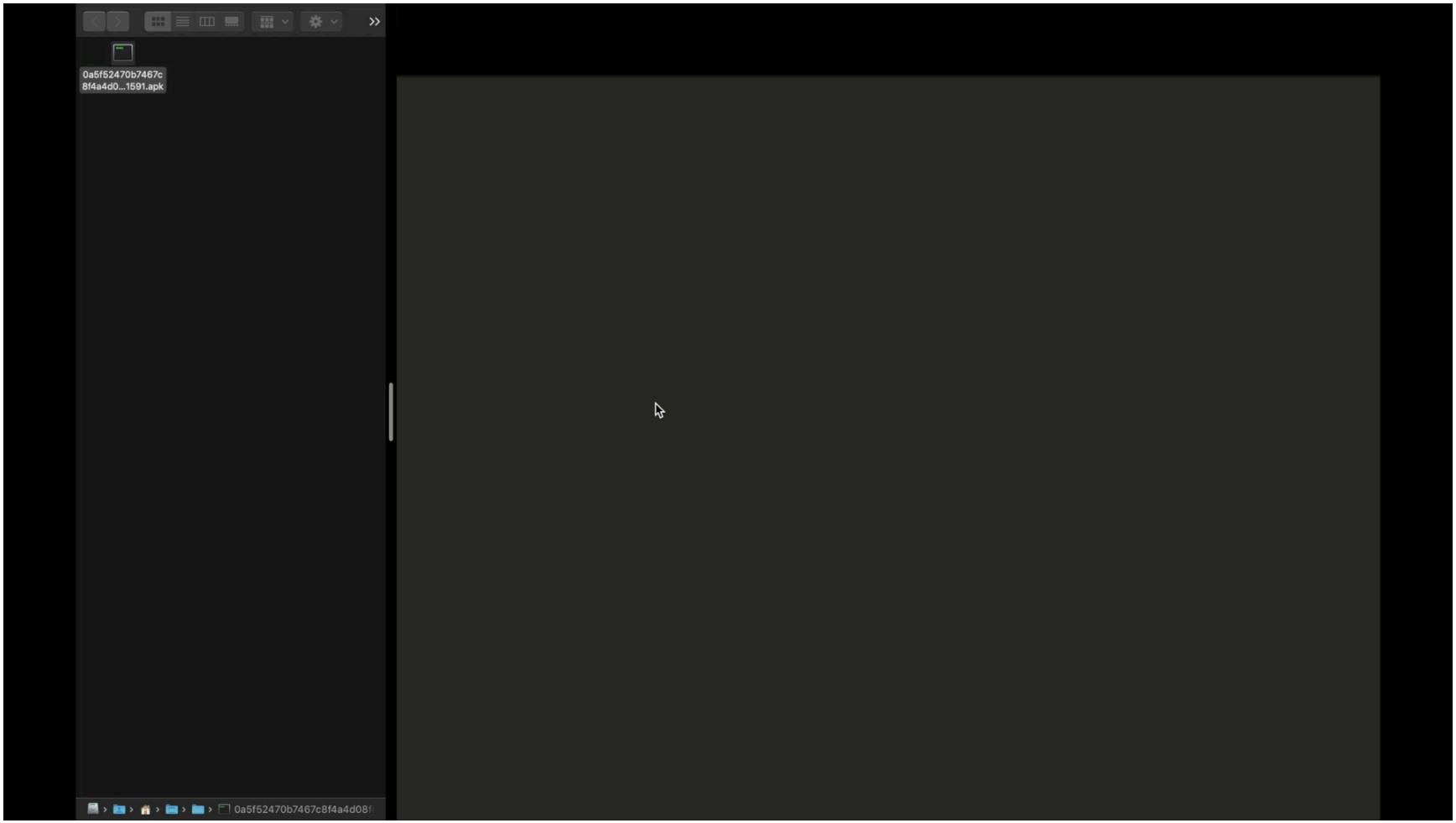
Identifying Malicious Android Applications in the Presence of Adversaries: A Cat-and-Mouse Game

Omid Mirzaei

Systems Security Lab (SecLab)
Khoury College of Computer Sciences
October 2019

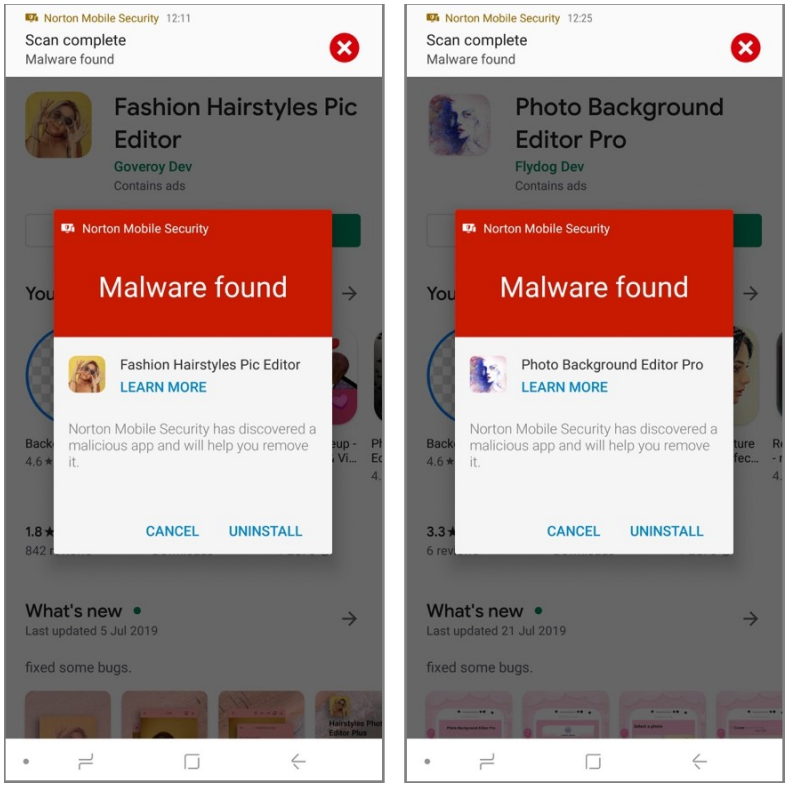
OWASP Boston Application Security Conference

Toy Example


0a5f52470b7467c
8f4a4d0...1591.apk
0a5f52470b7467c8f4a4d08f

Motivation (1/2)

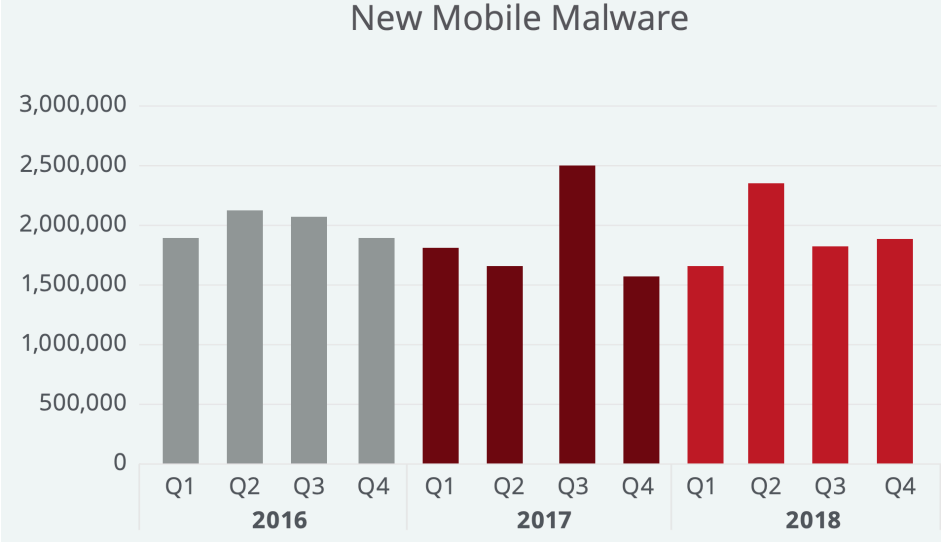
» Android malware finds its way to both official and third-party online app stores



Symantec (Sep. 2019)

Motivation (2/2)

» Wrong assumptions in previous studies:
  – Malware authors are passive
  – Attack vectors do not change regularly



McAfee Mobile Threat Report Q1, 2019

Outline

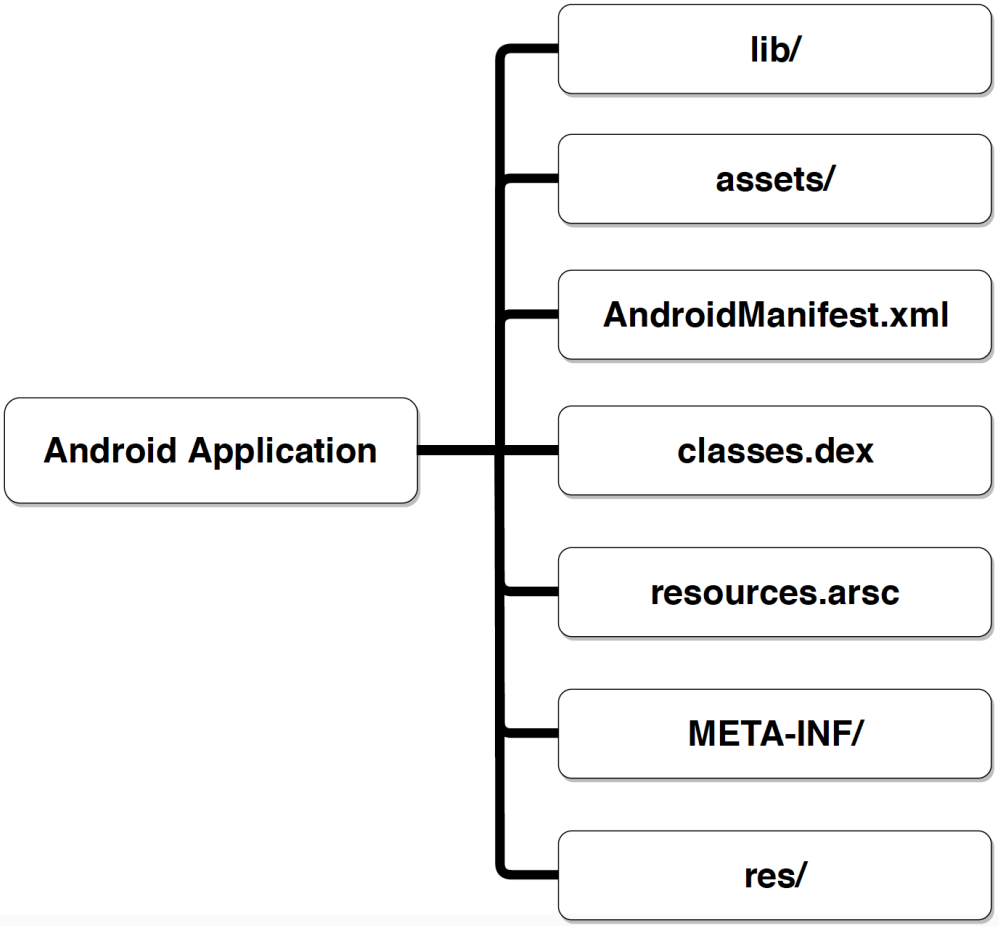
- Background Information
- Iconography
- Android Apps Triage
- Android Malware Detection
- Discussion
- Conclusion

Outline

- **Background Information**
- Iconography
- Android Apps Triage
- Android Malware Detection
- Discussion
- Conclusion

Background Information

Android app structure



Background Information

Anti-analysis techniques

- Data Manipulation (Perturbation)
- Obfuscation
- Dynamic Code Loading
- Packing
- Repackaging
- Piggybacking
- Emulation Detection

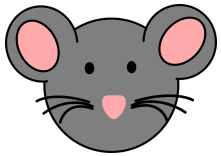
Outline

- Background Information
- **Iconography**
- Android Apps Triage
- Android Malware Detection
- Discussion
- Conclusion

Iconography

Android App Triage

Android Malware Detection



Evasion of Android Malware Detection

Outline

- Background Information
- Iconography
- **Android Apps Triage**
- Android Malware Detection
- Discussion
- Conclusion

Android Apps Triage

Definition

- Using fast tools or techniques to narrow down malware analysis
- Saving time
- Saving computational resources

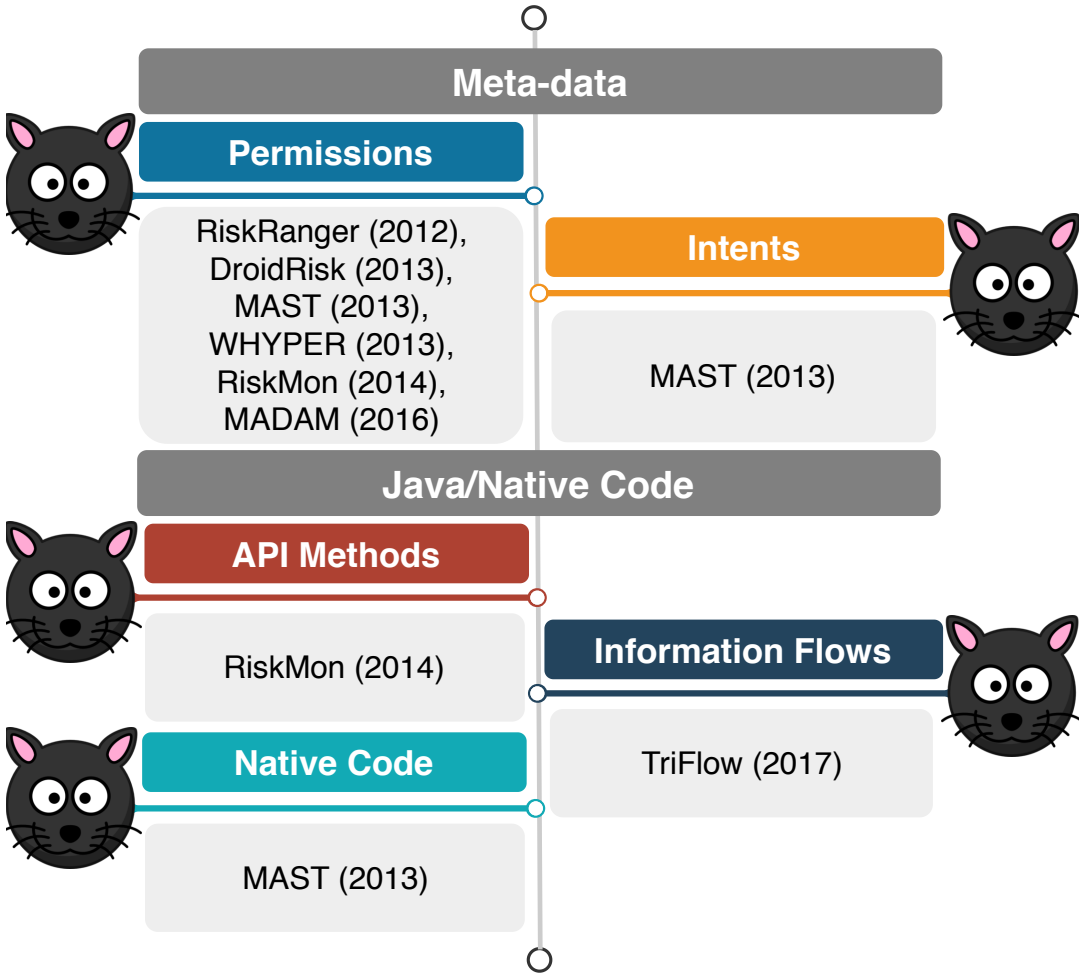
Android Apps Triage

Feature Categories

- Features extracted via **static analysis**:
  - Meta-data (Manifest file)
  - Market Data
  - Java/Native Code
- Features extracted via **dynamic analysis**:
  - App's Behavior (function calls)
  - Network Usage
  - Root Exploits

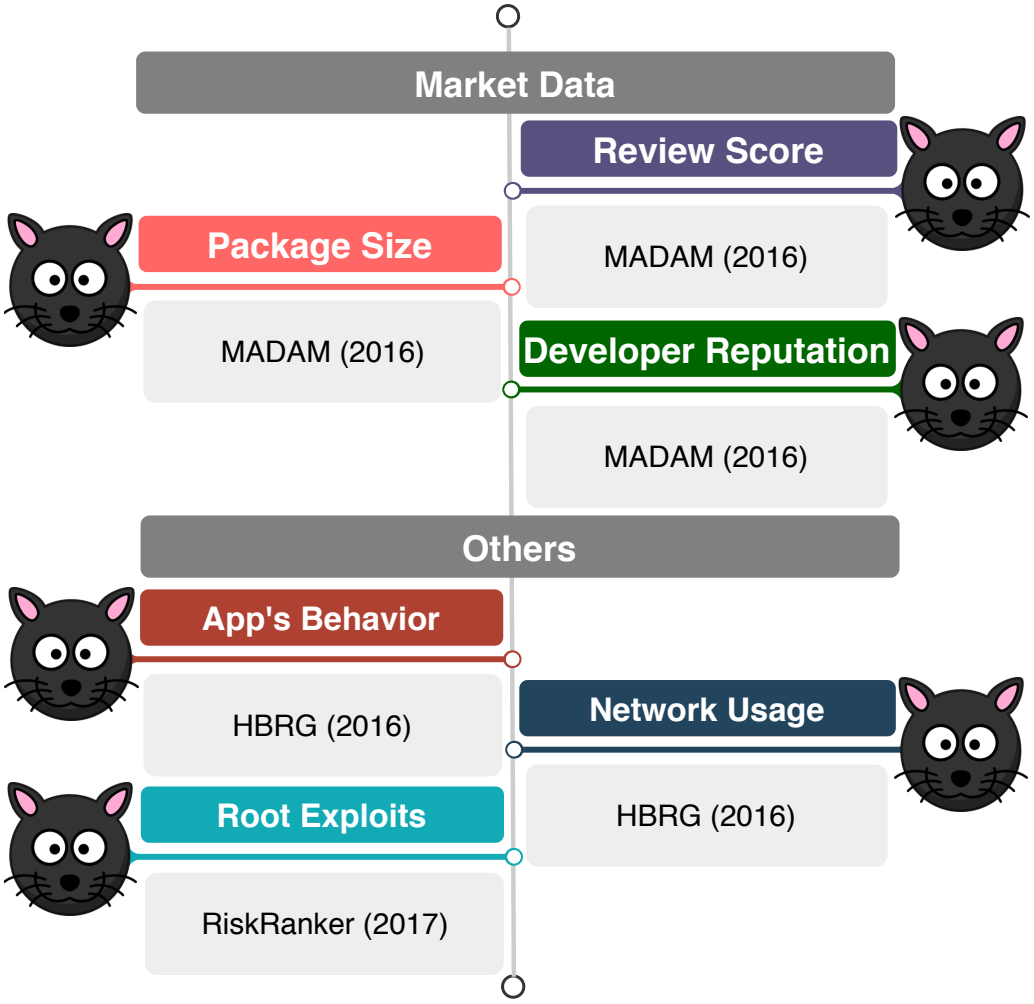
Android Apps Triage

Systems Overview (1/2)



Android Apps Triage

Systems Overview (2/2)



Outline

- Background Information
- Iconography
- Android Apps Triage
- **Android Malware Detection**
- Discussion
- Conclusion

Android Malware Detection

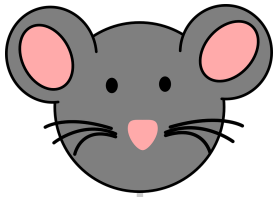
Overview

- **Signature-based Systems**
  - Outdated dictionaries containing signatures
  - Zero-day malware
  - Polymorphic/Metamorphic/Oligomorphic malware
- **ML-based Systems**
  - Computational resources
  - Need for re-training (in offline ML systems)
  - Vulnerability to adversarial attacks

Signature-based Systems

Vendor	Package Name	Version	# Downloads
AVG	com.antivirus	6.23.8	+100M
Symantec	com.symantec.mobilesecurity	4.7.0.4456	+10M
Lookout	com.lookout	10.28.1-f01e73e	+100M
ESET	com.eset.ems2.gp	5.1.25.0	+10M
Dr. Web	com.drweb	11.3.2	+100M
Kaspersky	com.kms.free	11.31.4.2437	+50M
Trend micro	com.trendmicro.tmmspersonal	11.0.1	+1M
ESTSoft	com.estsoft.alyac	2.1.11.7	+10M
Zoner	com.zoner.android.antivirus	1.15.3	+1M
Webroot	com.webroot.security	5.5.6.46428	+1M

Evasion of Signature-based Systems



Signature-based Malware Detection

Repacking
Disassembling and Reassembling
Obfuscation
Reflection

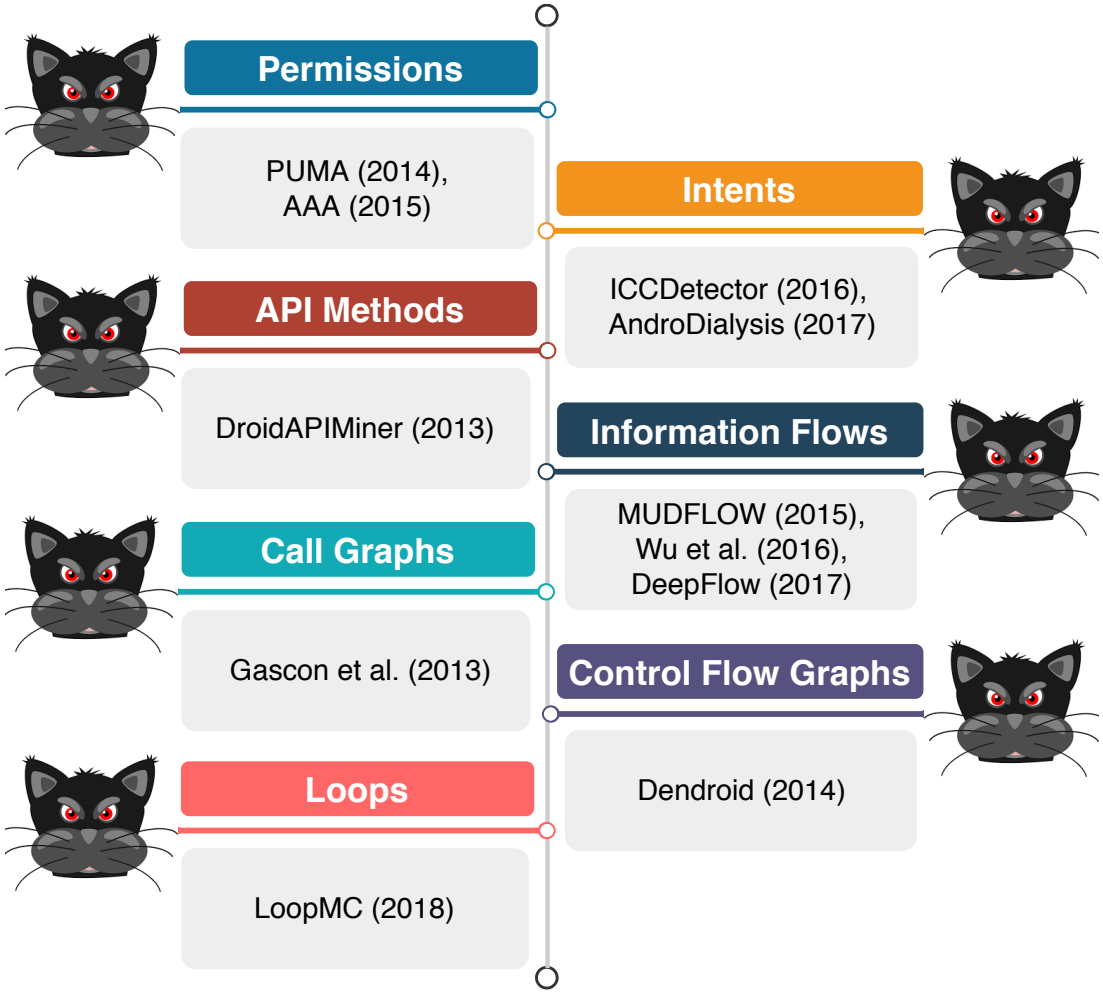
Android Malware Detection

ML-based Systems

- How features are extracted?
- Which ML algorithm(s) is(are) applied?
- Where the ML model is kept?

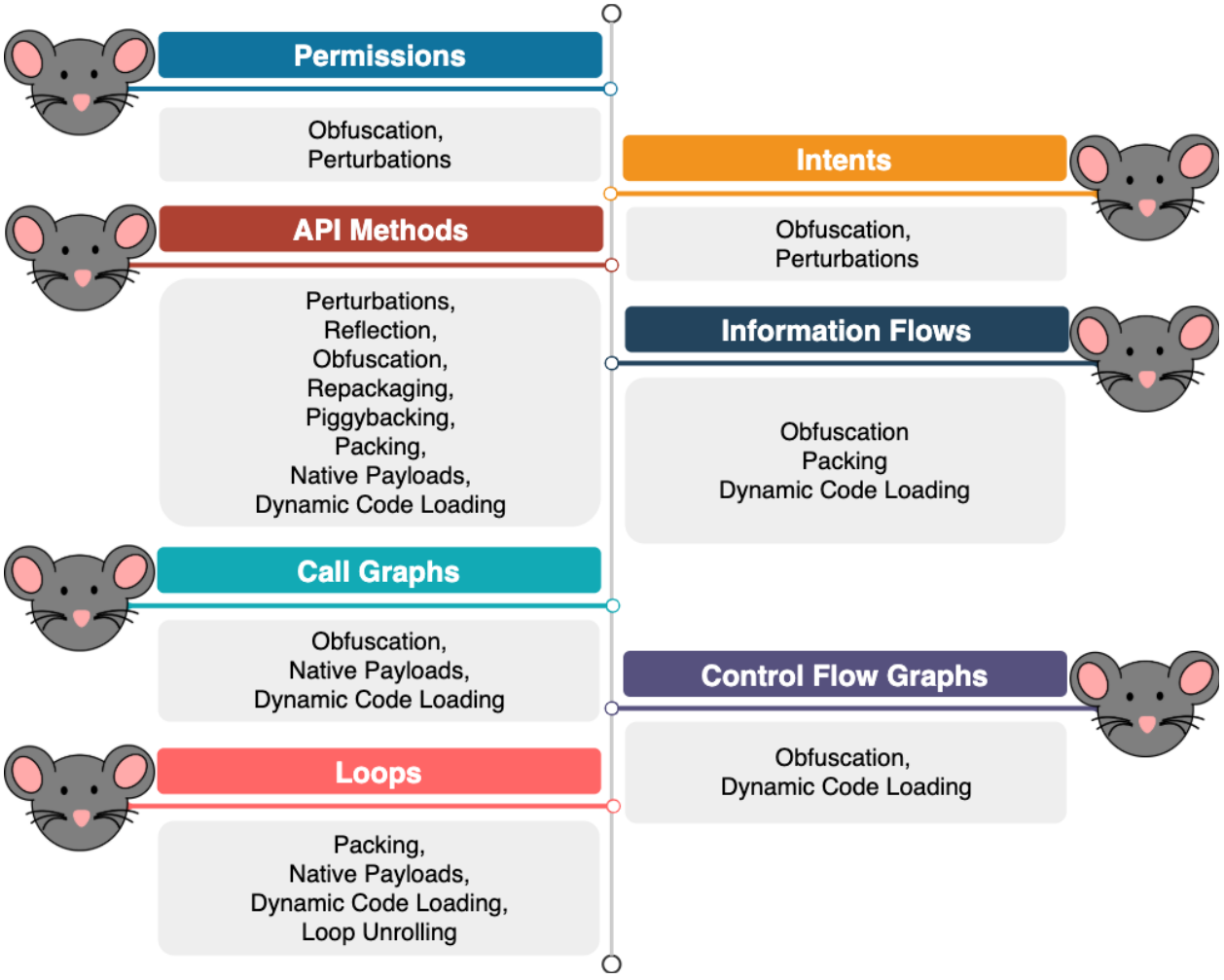
ML-based Systems

Features from Static Analysis



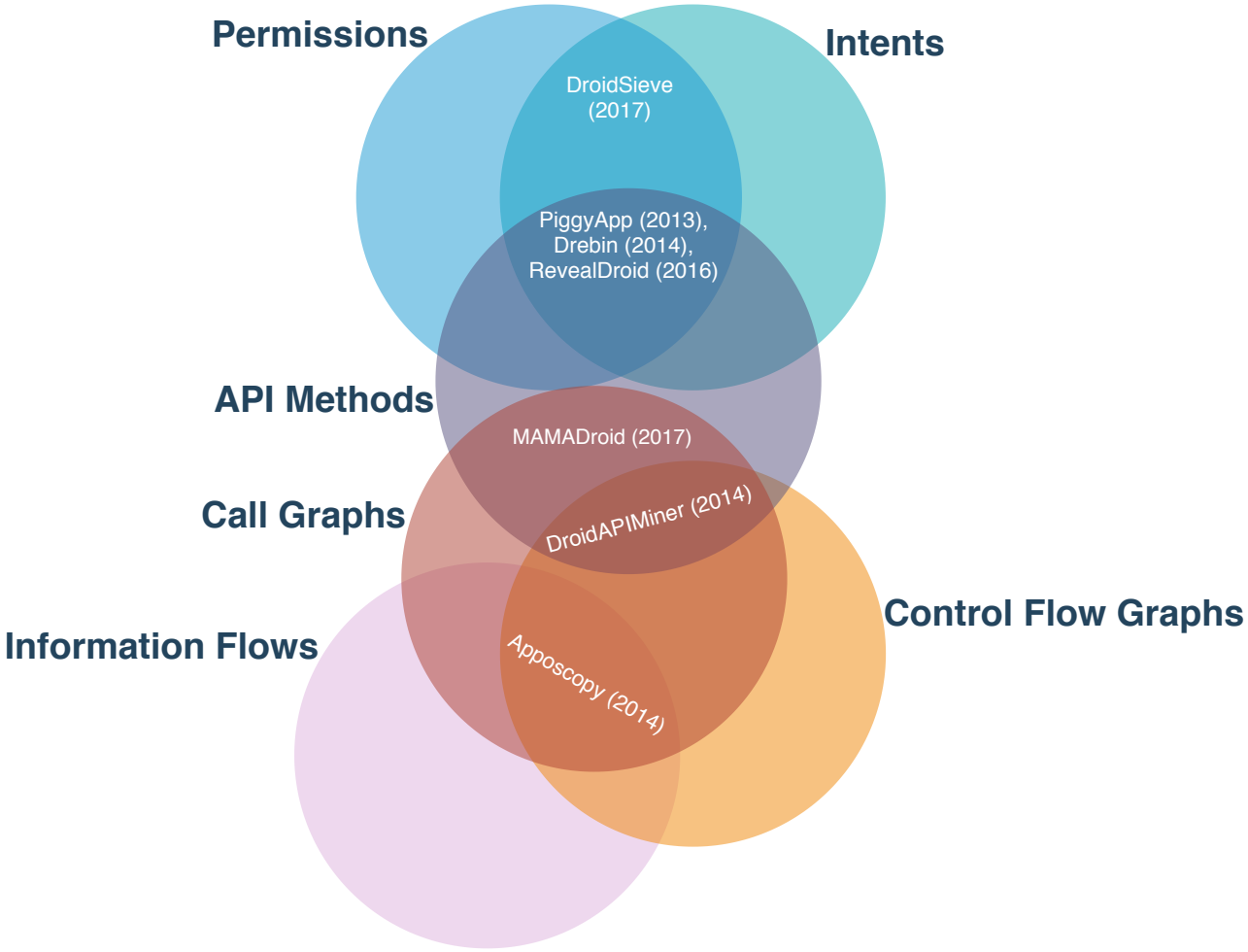
Evasion of ML-based Systems

Features from static analysis



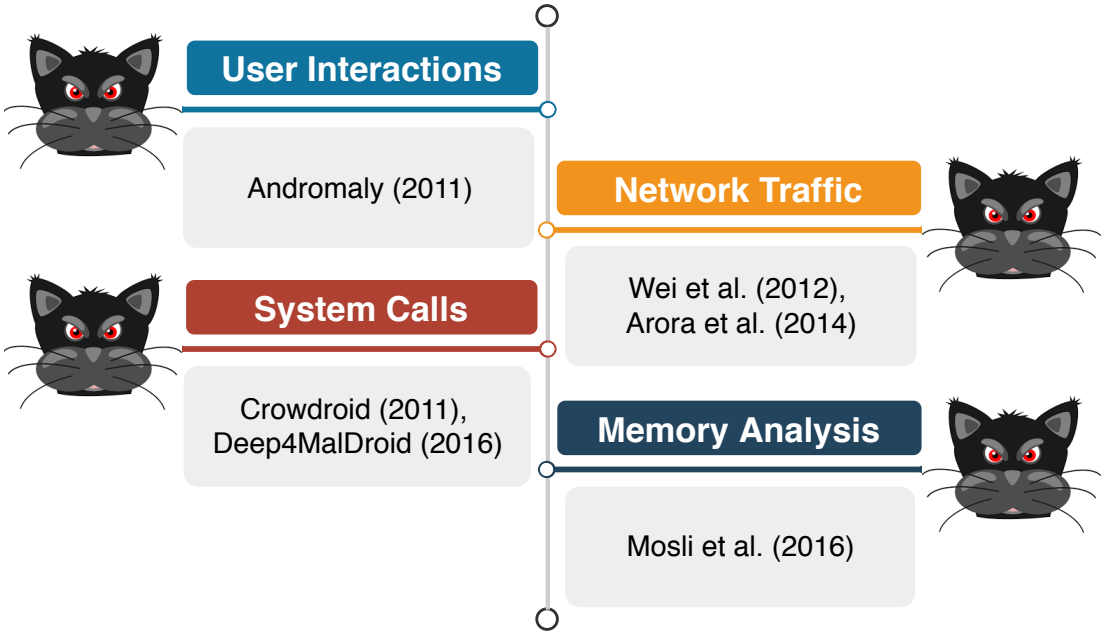
ML-based Systems

Combination of multiple features from static analysis



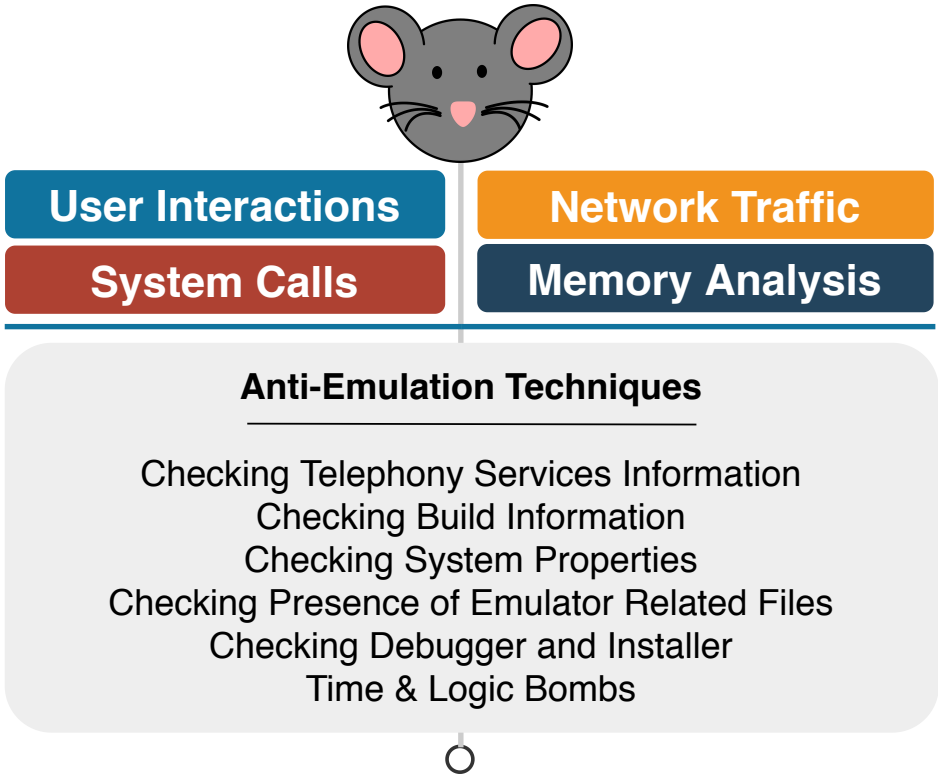
ML-based Systems

Features from dynamic analysis



Evasion of ML-based Systems

Features from dynamic analysis



ML-based Systems

Features from hybrid analysis

Features from Static Analysis	+	Features from Dynamic Analysis
DroidMat (2012)		
Permissions, API Methods, Control Flows, Information Flows		Logged Behavior Sequence, System Calls, Information Flows, Power Consumption
Droid-Sec (2014)		
Permissions, API Methods		Function Call Monitoring
Zhao et al. (2014)		
Components		Function Call Monitoring

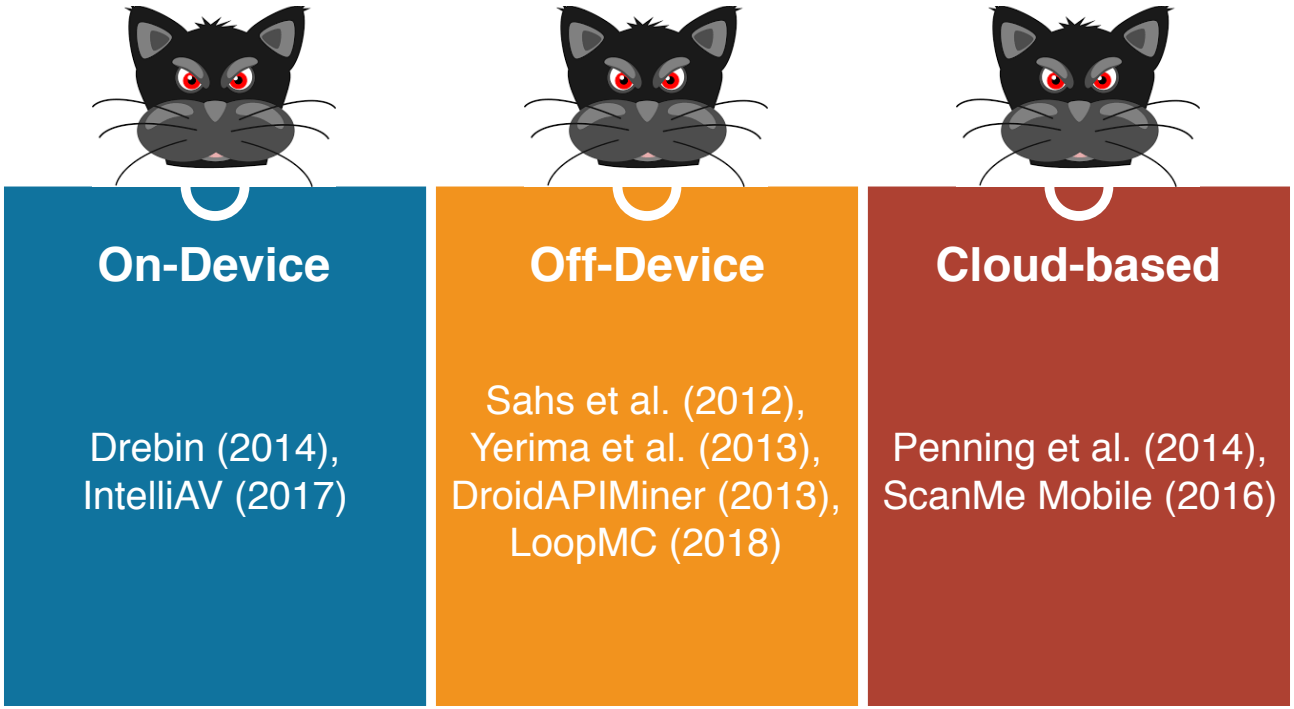
ML-based Systems

One vs. Multiple Learning Algorithms

Alam et al. (2013),
Yerima et al. (2015),
Sheen et al. (2015),
Bai et al. (2016),
Wang et al. (2017),
PIndroid (2017),
Mlifdect (2017),
DroidFusion (2017),
Feng et al. (2018)

ML-based Systems

On-Device vs. Off-Device



Outline

- Background Information
- Iconography
- Android Apps Triage
- Android Malware Detection
- **Discussion**
- Conclusion

Discussion

- Characteristics of realistic adversarial attacks:
  - Preserving the app's malicious behavior
  - Maintaining the app's integrity
  - Evading ML-based malware detectors
- Considerations for malware detection systems?
  - Robustness
  - Adversarial Training

Outline

- Background Information
- Iconography
- Android Apps Triage
- Android Malware Detection
- Discussion
- **Conclusion**

Conclusion

- Malware is **evolving continuously**.
- Both **academic and industrial systems** for Android malware detection **are prone to error** in the presence of **adversaries**.
- Even worse performance by the emergence of **intelligent malware** in near future.
- A real need for **resilient and robust** systems.

Questions?

Related Blog Post:
https://omirzaei.github.io/blog/2019/Identifying_Malicious_Android_Apps/

Other Relevant Information:
https://omirzaei.github.io/assets/pdf/PhD_Thesis.pdf
https://omirzaei.github.io/blog/2017/Android_Malware_Evolution/

Email: o.mirzaei@northeastern.edu