MINING STRUCTURAL AND BEHAVIORAL PATTERNS IN SMART MALWARE

Author: Guillermo Suarez-Tangil
Supervisors: Juan E. Tapiador, Pedro Peris-Lopez

COmputer SEcurity (COSEC) Lab
Computer Department — Universidad Carlos III de Madrid

PhD Defense – October 2014
Madrid, Spain

COSEC UC3M COMPUTER SECURITY LAB

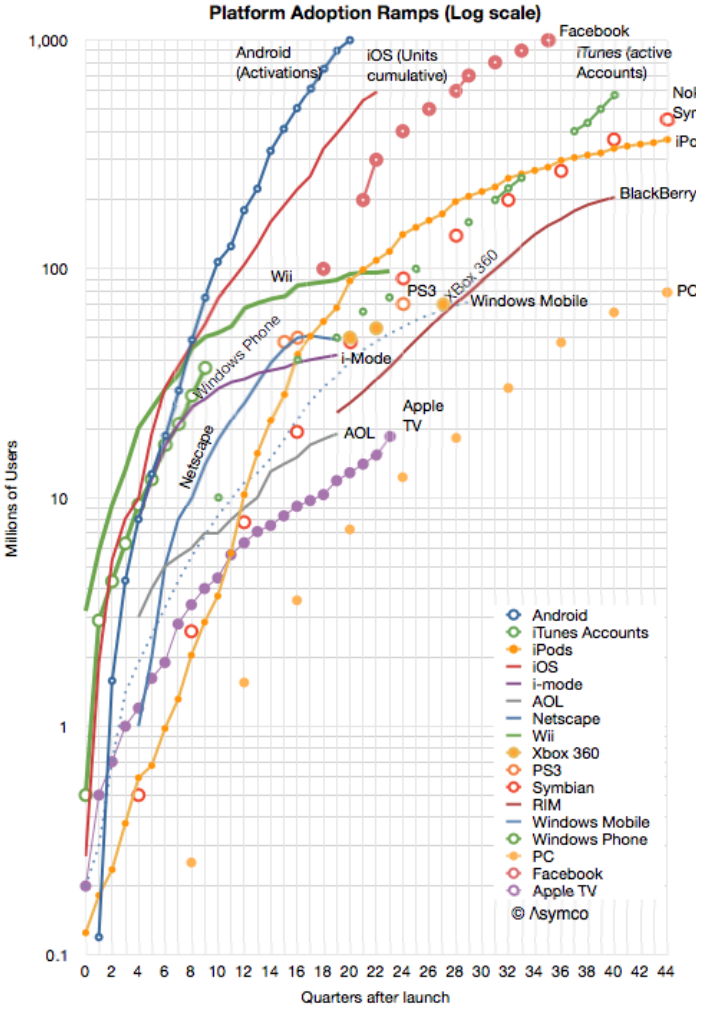
Outline

Outline

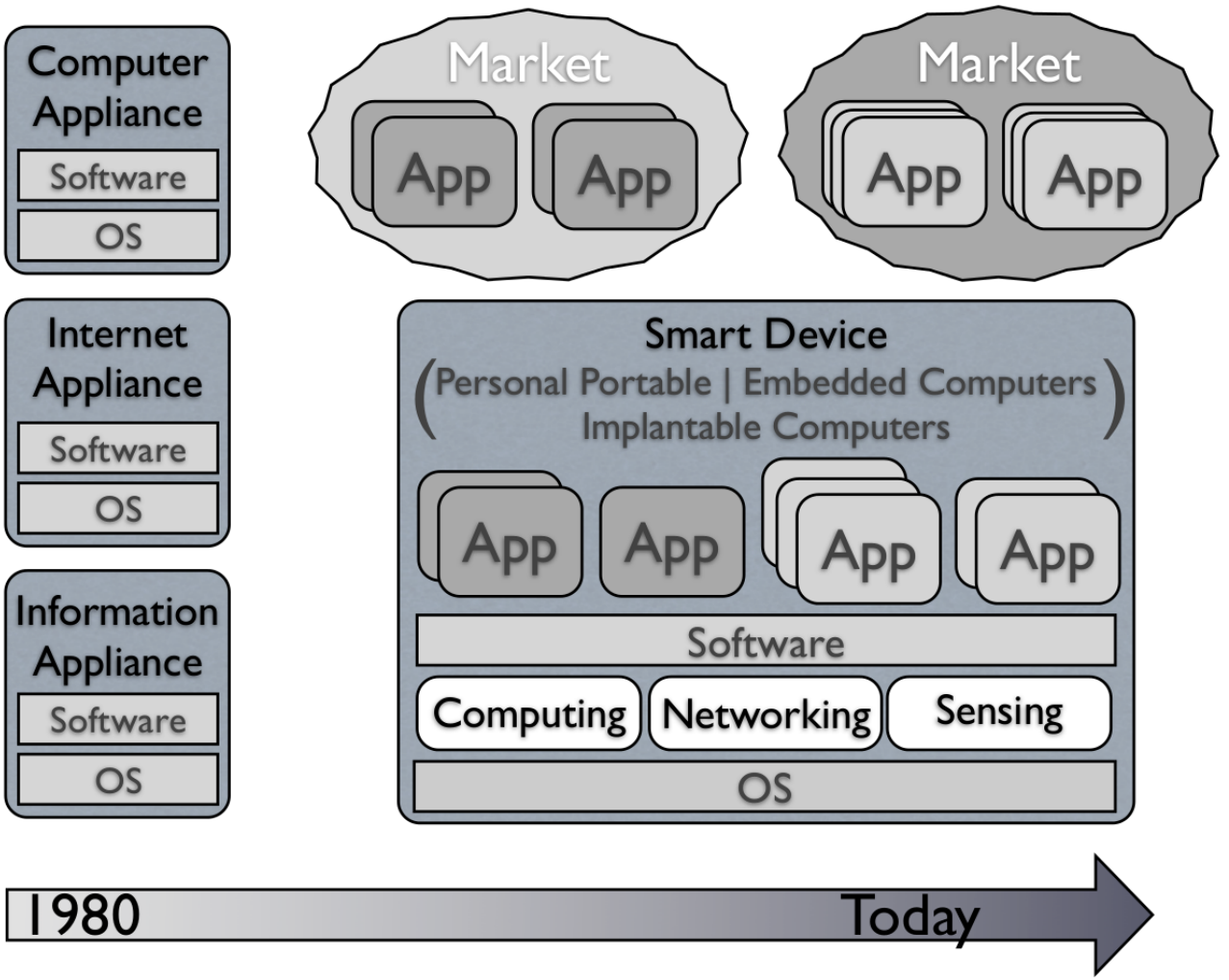
Contributions and Tools

- **Foundations & Tools**
  1. Evolution
  2. Maldroid Lab
- **Static-based Analysis**
  3. Dendroid
- **Dynamic-based Analysis**
  4. Alterdroid
- **Cloud-based Analysis**
  5. Meterdroid
  6. Targetdroid

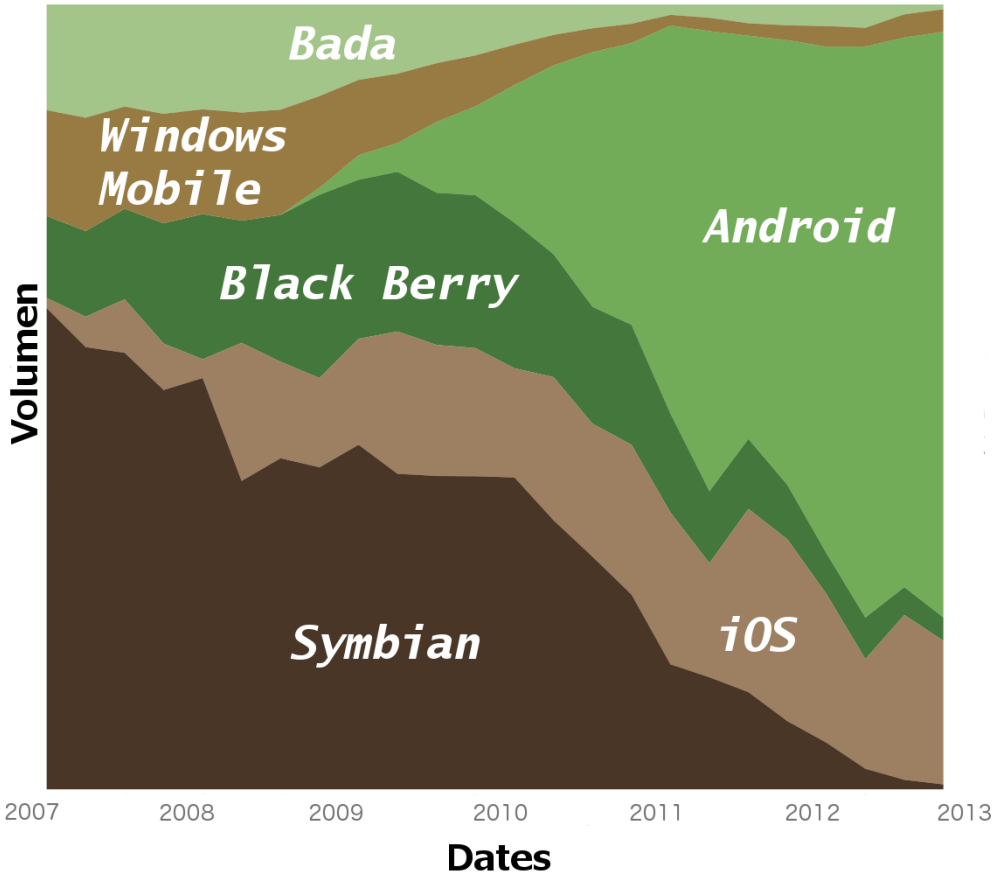
Smart devices are rapidly emerging as *popular appliances* with increasingly powerful capabilities



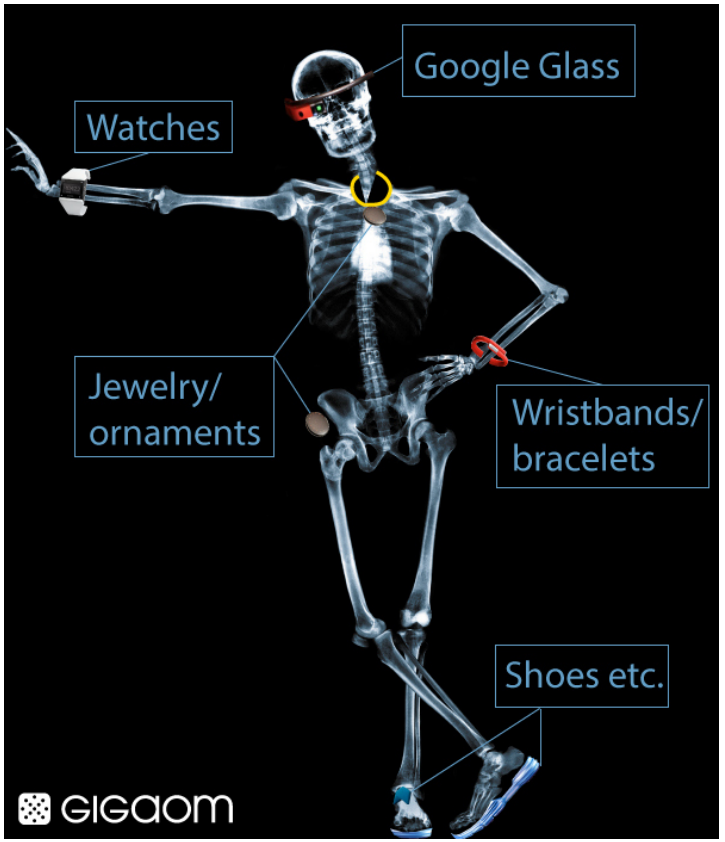
Smart devices offer the possibility to easily **incorporate third-party applications** through online markets



Smartphones, mainly *Android*, are the platform of choice, but **new smart devices** are appearing at a steady pace



(a) Market share until 2013...



(b) ...2014 and forthcoming

Smart devices present greater security and privacy issues to users due to their **situational awareness**

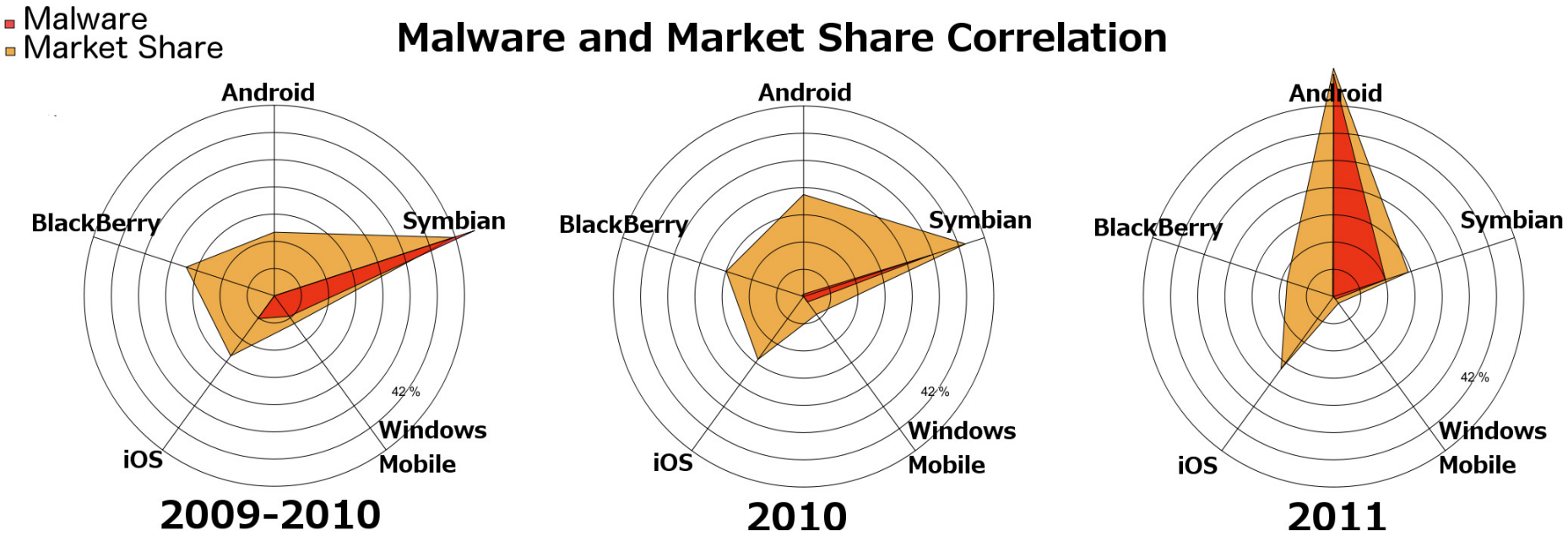
Popularity + Third-party Apps = Malware

One major source of security and privacy problems is precisely the ability to incorporate third-party applications

New security problems

- Leakage of users' personal information
- Traceability issues
- Cybercrime
- Etc

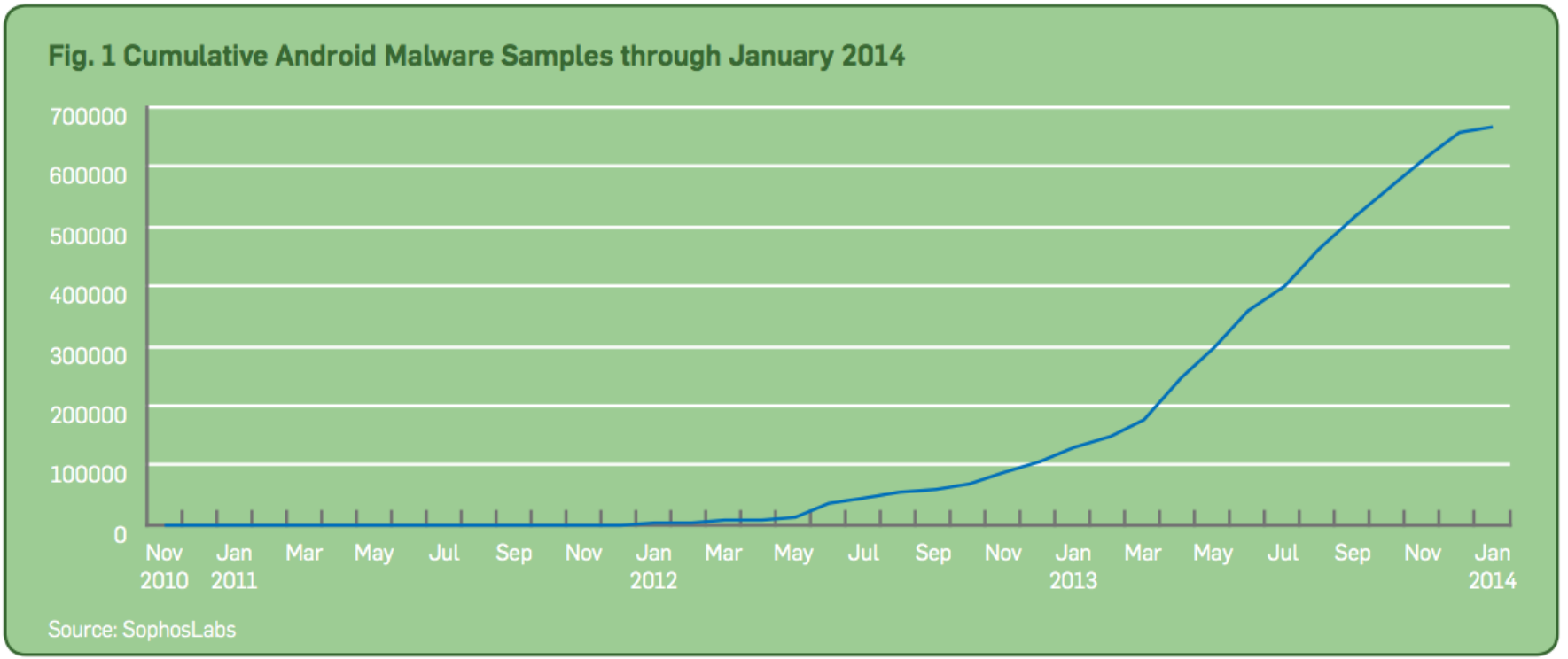
The rapid growth of smartphone raised a similar **increase in the number and sophistication of malware**



Malware-as-a-Service (MAAS)

$$Cost(Attack) < Potential\ Revenue$$

Correlations/causations are paramount to place efforts, and to understand future tendencies and threats



This Thesis...

... is strongly biased towards **smartphones** (particularly **Android**), since they currently are the most extended class of smart devices

Thwarting malware is a thriving research area with a substantial amount of still unsolved problems

Problem

Smart Devices

Impressive **growth** of goodware and malware

- Large number of potential targets
- Reuse-oriented malware
- Good | **Gray** | Malware

Smart Malware

Increase in the **sophistication**

- Advanced behavior
  - **Obfuscated**
  - **Targeted**

Goal

Smart Analysis

Research smart analysis & detection techniques

- **Automated** analysis
- **Efficient** analysis
- **Intelligent** analysis

This Thesis provides new findings for the analysis of **smart malware**

Design and develop a set of techniques to **assist security analysts and final users** upon the analysis of untrusted apps for smart devices and to **automate the identification of smart malware**

Partial objectives:

1. Study the evolution of malware and its analysis/detection
2. Develop methods for better analyzing malware in large markets:
   - Intelligent instruments
   - Automate the analysis
3. Facilitate the analysis of complex smart malware:
   - Targeting user-specific actions
   - Hindering detection with advanced obfuscation techniques
   - Exploiting platform limitations (e.g.: energy computation)
   - Etc.

Layout

Foundations & Tools

- **Contribution 1:** Evolution
- **Tools:** Maldroid Lab

Contribution 1: Evolution of Malware Analysis

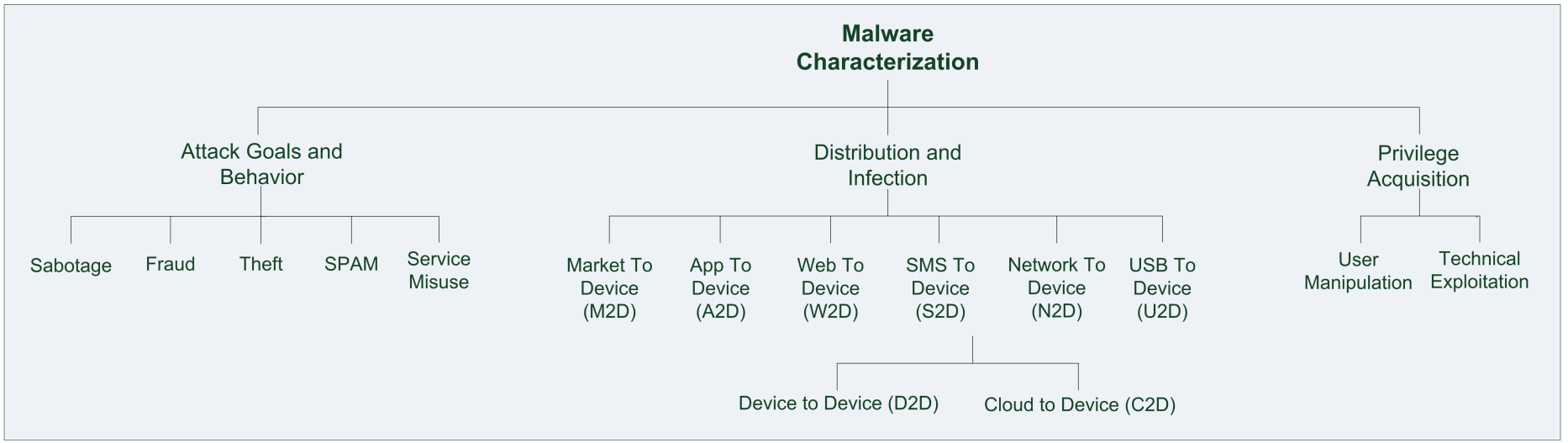
Foundations & Tools (I/II)

A comprehensive analysis of the evolution of untrusted code for smart devices and current detection strategies

Malware Characterization

Traditional classifications

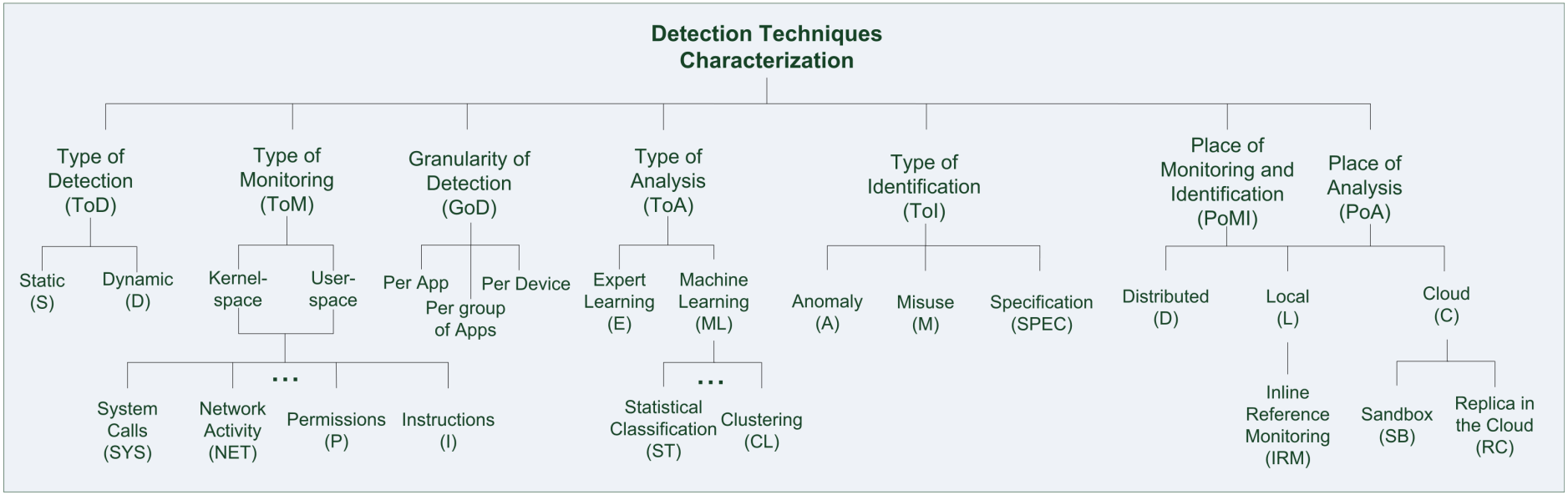
E.g., *virus*, *trojans* or *spyware* are rather imprecise due to the complexity and reuse-oriented nature



A Taxonomy of Detection Techniques

Malware detection

Malware **detection** is a complex process pulling together **monitoring**, **analysis** and **identification** tasks

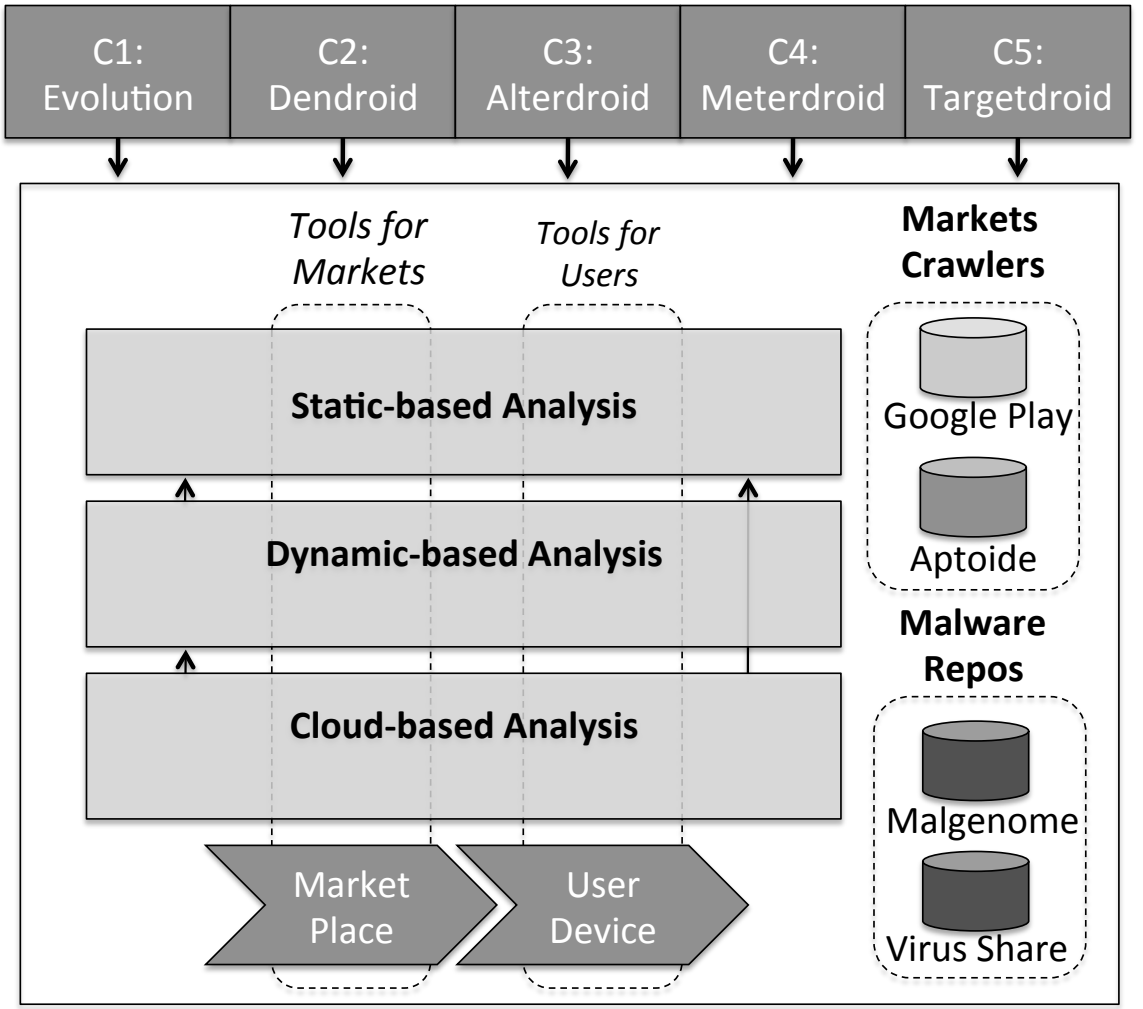


A research lab for smart malware analysis and detection

Maldroid Lab will be used throughout the contributions

Overview and Architecture

- Facilitate static analysis
- Automate repackaging
- Guarantee the isolation
- Automate VM allocation
- Automate installation
- Parallelize execution
- Automate event injection
- Support synchronized replicas in the cloud



Results: **Evolution**

P1: "Evolution, Detection and Analysis of Malware for Smart Devices".

- Authors: Guillermo Suarez-Tangil, Juan E. Tapiador, Pedro Peris-Lopez, and Arturo Ribagorda.
- *In: IEEE Communications Surveys & Tutorials*, 2013.
- I.F. (2012): **4.81**.
- Position in category: 2/132 (**Q1**) in Computer Science.

Software registration: Maldroid Lab.

- "A new generation laboratory of malware for testing smart malware and evaluating detection strategies"
-

Contribution 2: **Dendroid**

Static-based Analysis

A text mining approach for analyzing and classifying malware families

Human-driven analysis is unaffordable

Motivation and contribution

Challenges

- Dealing with bulky amounts of apps
- Classifying unseen samples
- Grouping malware into families

Context

- Malware Engineering
- Reuse Malware

Data Mining efficiently deals with massive amount of data

- To automatically analyze/classify —unknown— samples
- To obtain evolutionary analysis of malware families

We extract a high-level representation of the associated CFG and analyze various statistical features

Preliminaries

Control Flow Graph (CFG)

Possible execution paths that a program might traverse during its execution

Grammar

- **R**: Return
- **G**: Goto
- **I**: If
- **B**: BasicBlock
- **P**: Package
- **S**: String

Code Chunks

- **App**: $\texttt{Method}_a \mid \ldots \mid \texttt{Method}_y$
- **Method(s):**
  - `B[P0P1]B[I]B[P1R]`
  - `B[P1P1I]B[P0SP1P1P1]B[P1G]`
  - `B[P1P1I]B[I]B[P1R]`
  - `B[P0SP1P1P1]`
  - ...

The distribution of CCs suggests that each family can be characterized by just a few code structures

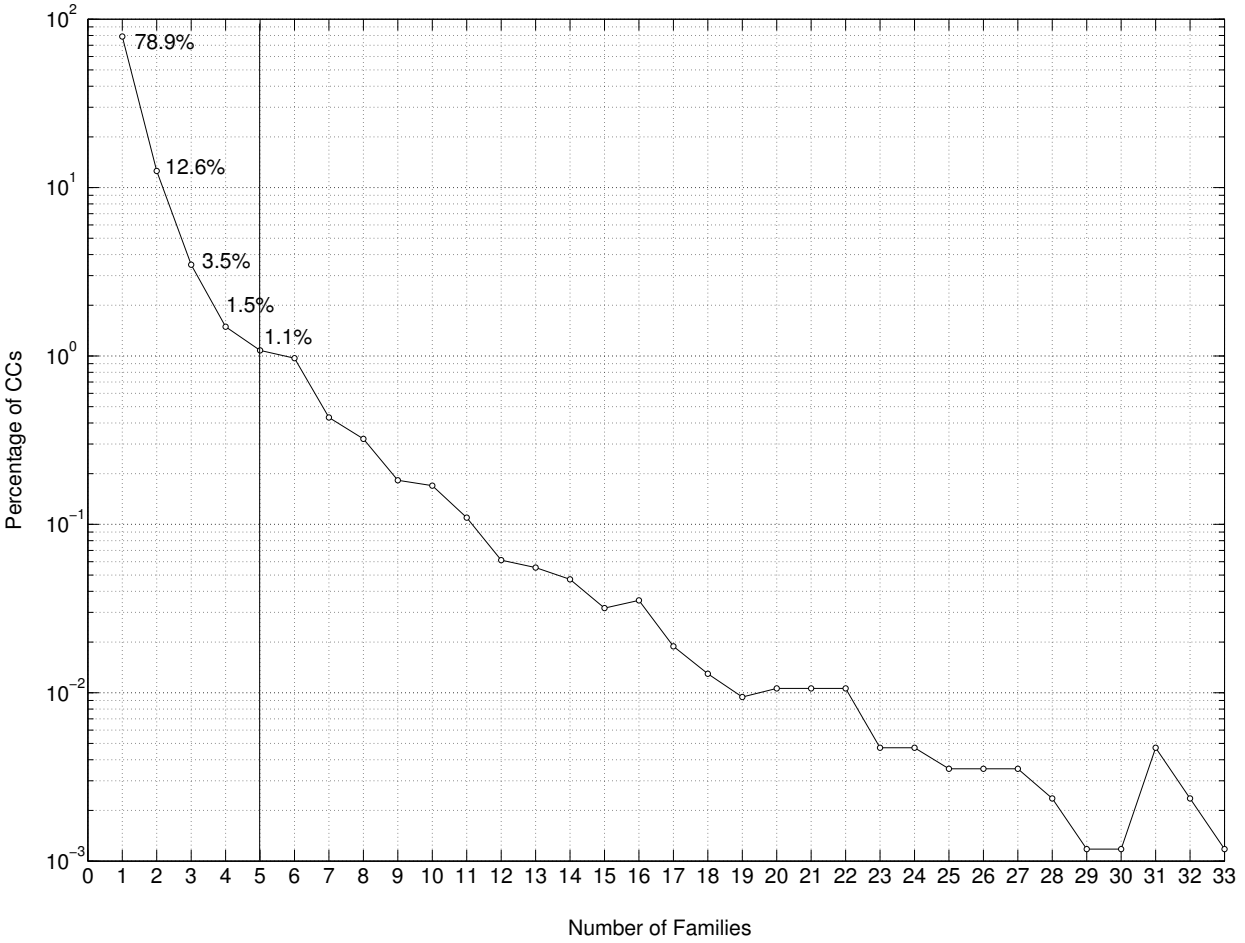
Experimental work with code chunks: B[P0P1]B[I]B[P1R]B[P1P1I]B[P0SP1P1P1]B[P1G]

Data set

- 1231 Specimens
- 33 Families
- 84854 CCs

Features

- # Unique CCs
- # Redundant CCs
- # Common CCs
- # Discriminant CCs



Mining Code Chunks in Malware Families

Text Mining

Vector Space Model (VSM)

We adapt to our problem various numerical indicators well researched in the field of information retrieval and text mining

Representation

$d_j = (w_{1j}, ..., w_{kj})$:

- documents
- words
- importance

Code Chunk Frequency

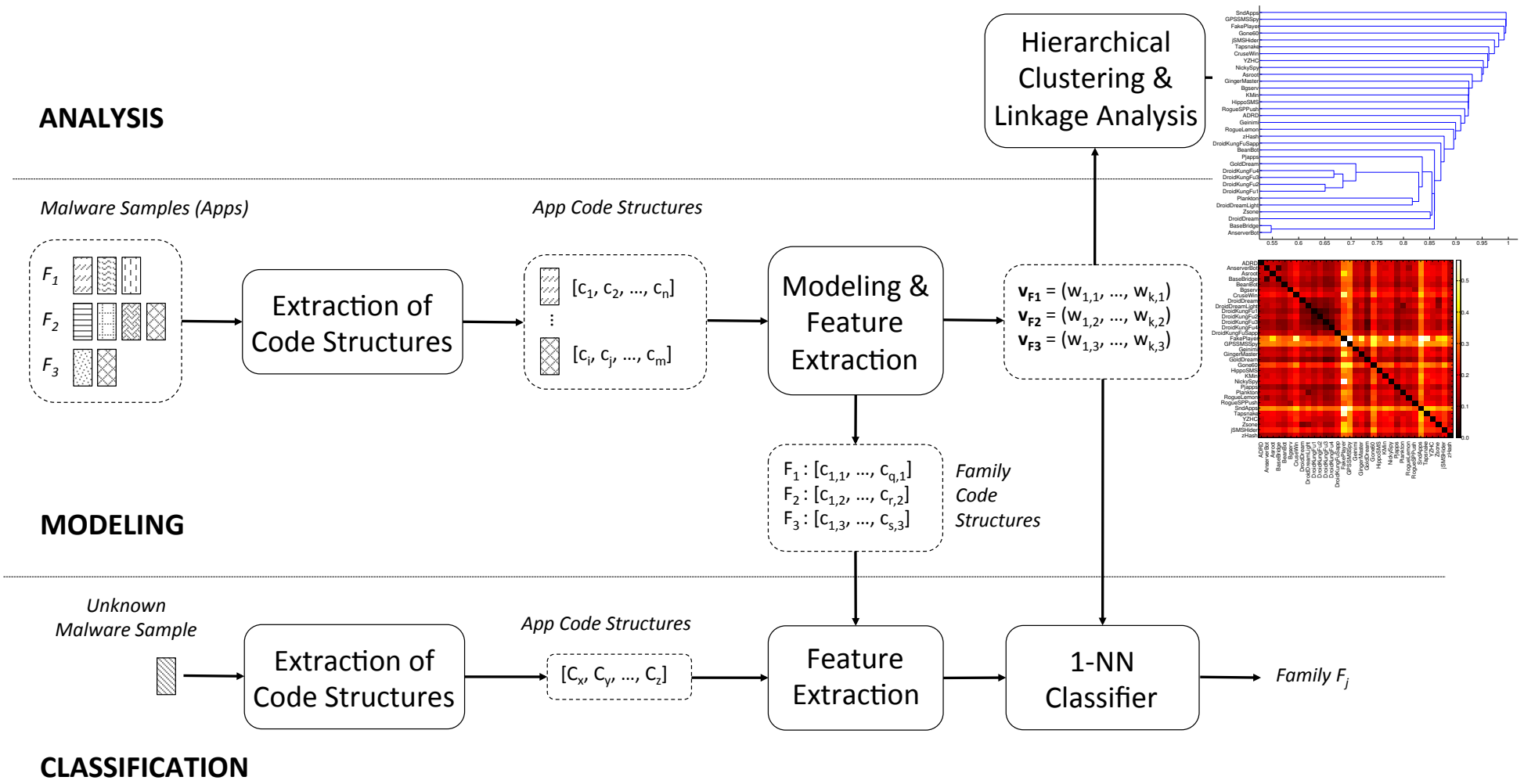
The frequency of a CC in a family $\mathcal{F}_j$

Inverse Family Frequency

The IFF of a CC c with respect to a set of malware families $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$

Dendroid: A Text Mining Approach

Our system



Malware classification error per family using 1-NN

Evaluation and results

Classification Error (%)			
ADRD	0.00%	GingerMaster	0.00%
AnserverBot	4.66%	GoldDream	0.00%
Asroot	0.00%	Gone60	0.00%
BaseBridge	7.92%	HippoSMS	0.00%
BeanBot	0.00%	KMin	0.00%
Bgserv	0.00%	NickySpy	0.00%
CruseWin	0.00%	Pjapps	0.00%
DroidDream	0.00%	Plankton	0.00%
DroidDreamLight	0.00%	RogueLemon	0.00%
DroidKungFu1	12.92%	RogueSPPush	0.00%
DroidKungFu2	19.46%	SndApps	0.00%
DroidKungFu3	8.12%	Tapsnake	0.00%
DroidKungFu4	18.21%	YZHC	0.00%
DroidKungFuSapp	0.00%	Zsone	0.00%
FakePlayer	0.00%	jSMSHider	0.00%
GPSSMSSpy	0.00%	zHash	0.00%
Geinimi	0.00%		

Table : Average classification error using 1-NN with 10-fold cross-validation.

Text classification is suitable for classifying malware

Conclusions

Text mining is suitable for

- Measuring similarity among malware samples
- Classifying unknown samples into known families

Hierarchical clustering

- Phylogenetic trees for malware families
- Extremely useful for analysts to identify the relationships among families

Automation of malware classification

- Results suggest that this technique is **fast**, **scalable** and **very accurate**

Results: **Dendroid**

P2: "Dendroid: A Text Mining Approach to Analyzing and Classifying Code Structures in Android Malware Families".

- Authors: Guillermo Suarez-Tangil, Juan E. Tapiador, Pedro Peris-Lopez, and Jorge Blasco.
- *In: Expert Systems with Applications (Elsevier)*, Vol. 41:4, pp. 1104-1117 (2014).
- I.F. (2012): **1.85**.
- *Position/Category*: 56/243 (**Q1**) in Engineering.

Contribution 3: **Alterdroid**

Dynamic-based Analysis

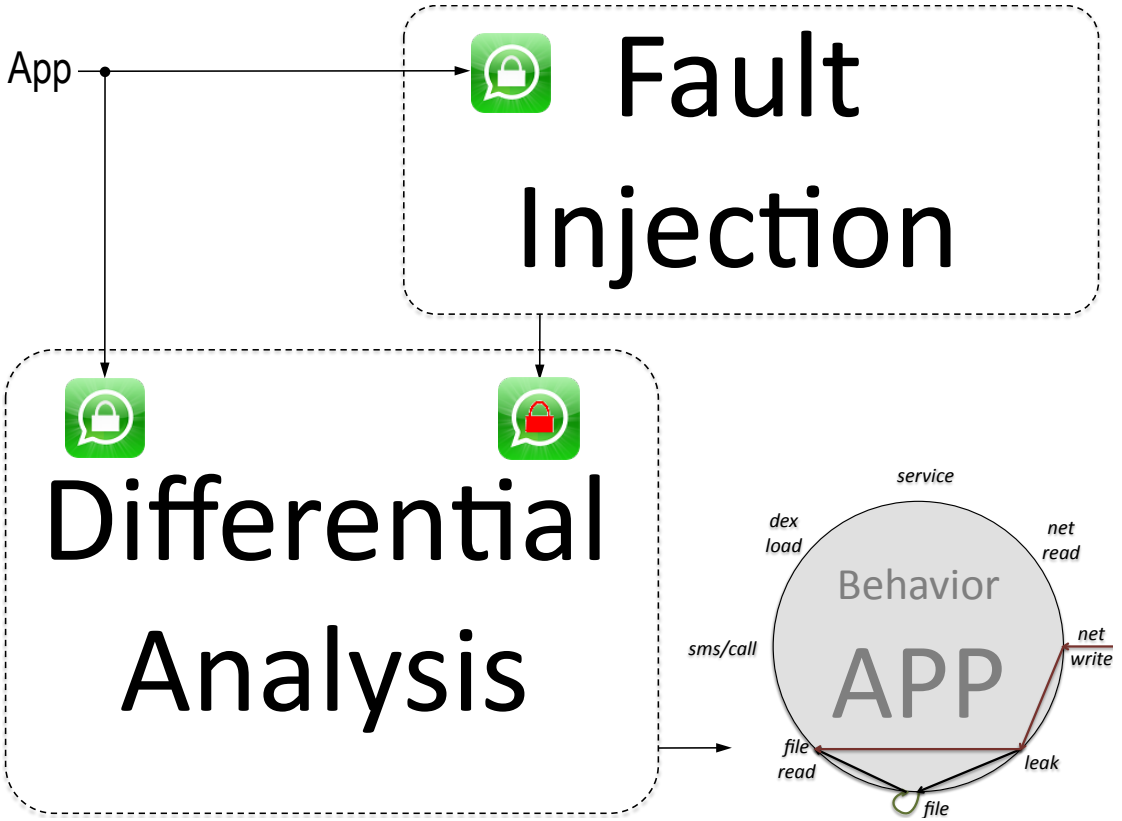
Differential fault analysis of obfuscated malware behavior

Recent malware hides and obfuscates its functionality

Motivation and contribution

Challenges

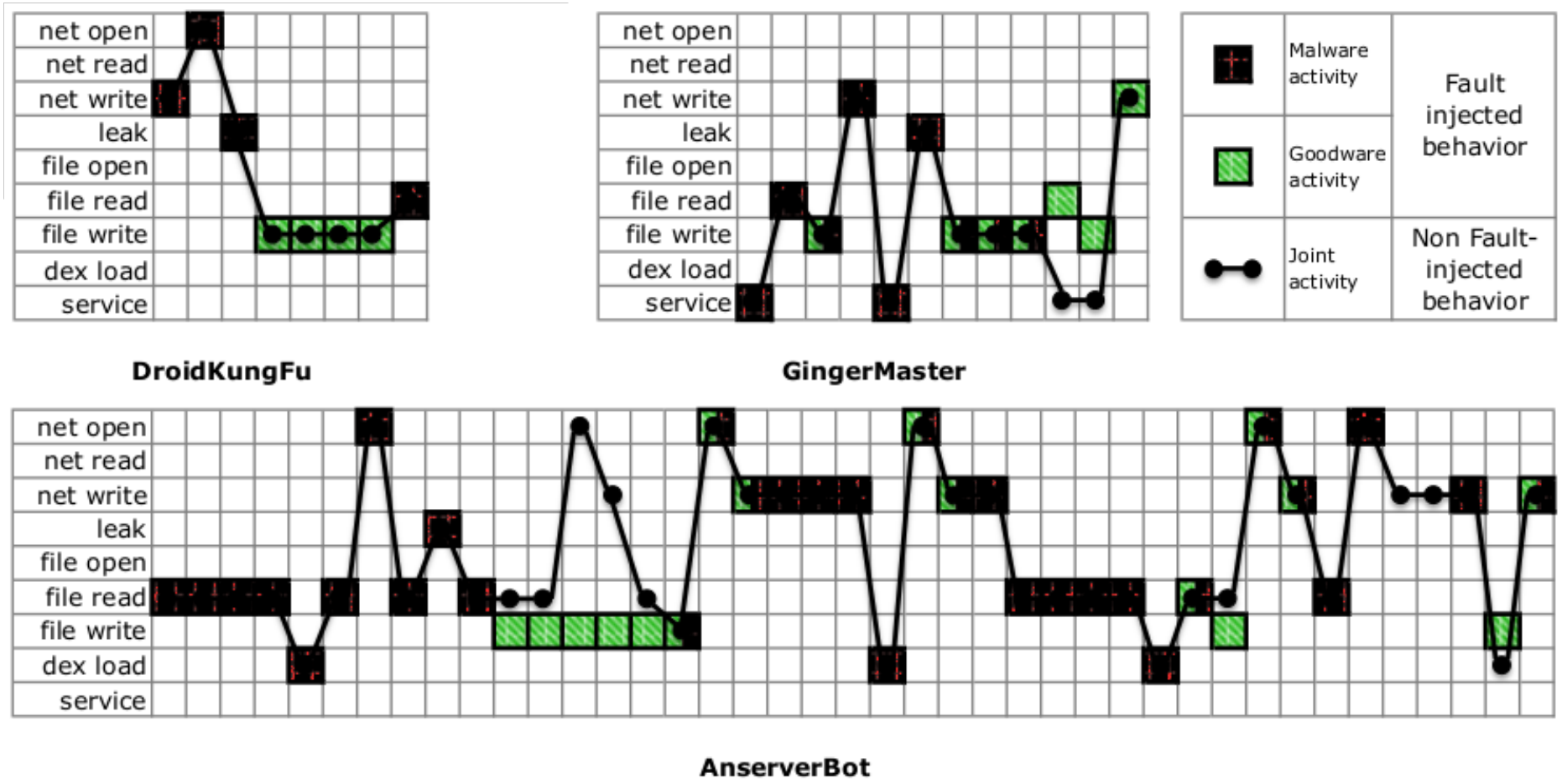
- Identification of grayware
- Identification of obfuscated mw
- Attribution of behaviors to parts of an app



Differential fault analysis can identify hidden malicious components

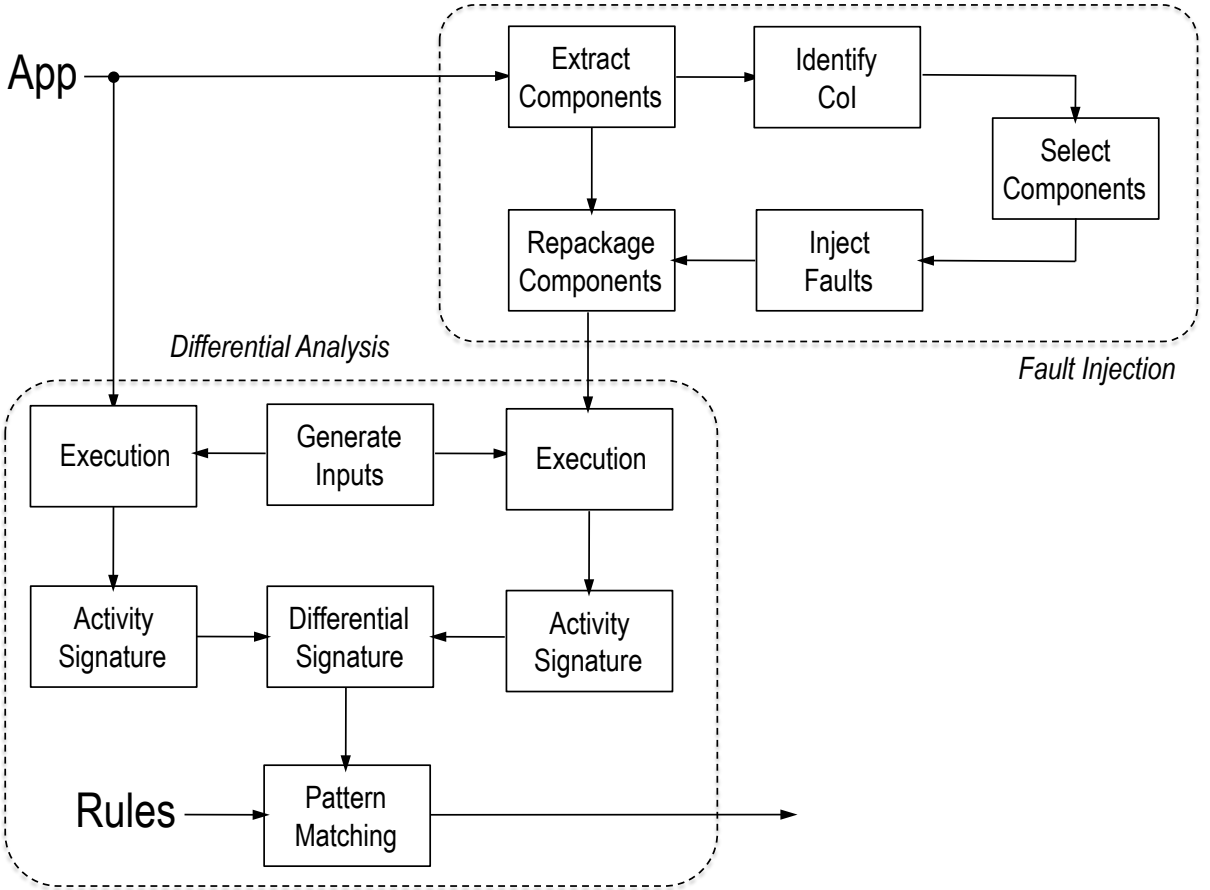
Differential fault analysis facilitate the analysis of complex obfuscated mobile as shown with these representative cases

Experimental work



Alterdroid can be used to automate the analysis of a dataset of samples

Our system



Alterdroid can be used to automate the analysis of a dataset of samples

Evaluation and results (I/III)

	#Apps	#CoI	#FIO	#Match	Overhead	TP	FN	Accuracy
DKF	34	6.11	6.11	11.71	283.93 s	33	1	97.06%
ASB	187	1.35	1.35	3.90	246.22 s	186	1	99.47%
GM	4	4.00	4.00	6.00	248.23 s	3	1	75%
GM+	4	4.00	4.00	3.00	1026.01 s	4	0	100%
Gray	16	2.88	2.88	4.19	248.24 s	16	0	100%
Good	81	0.57	0.00	0.00	0.00 s	81	0	100%

Table : Performance evaluation against existing malware, grayware, and goodware apps. The number of CoIs, FIOs, Matches, is given on average per app, and the overhead is given on average per FIO and app.

Alterdroid can be used to automate the analysis of a dataset of samples

Evaluation and results (II/III)

		VirusShare (VS)	Aptoide (AP)
Sum.	**No. Apps**	2 913	2 994
	Avg. No. CoIS	145.6	284.4
	Avg. No. FIOs	138.3	273.5
CoIS	**ImageFileMatch**	397 248	813 754
	EncOrCompressed	16 687	35 293
	ImgExtensionMismatch	5 771	5 246
	DEXFileMatch	2 827	2 995
	APKFileMatch	1 087	58
	APKExtensionMismatch	517	39
FIOs	**ImageFile**	397 248	813 754
	GenericMutationFile	5 714	5 237
Rules	**No. R_{FAC}**	2 802	2 962
	No. R_{NAC}	2 773	2 929
	No. R_{DLC}	1 971	669
	No. R_{SAC}	220	0
–	**Average Overhead**	584.51 s	666.67 s

Performance of the detection of an obfuscated component

Evaluation and results (III/III)

Time of a differential analysis

$t = n_{\text{fault}} \cdot t_{\text{genFault}} \cdot t_{\text{diff}}$

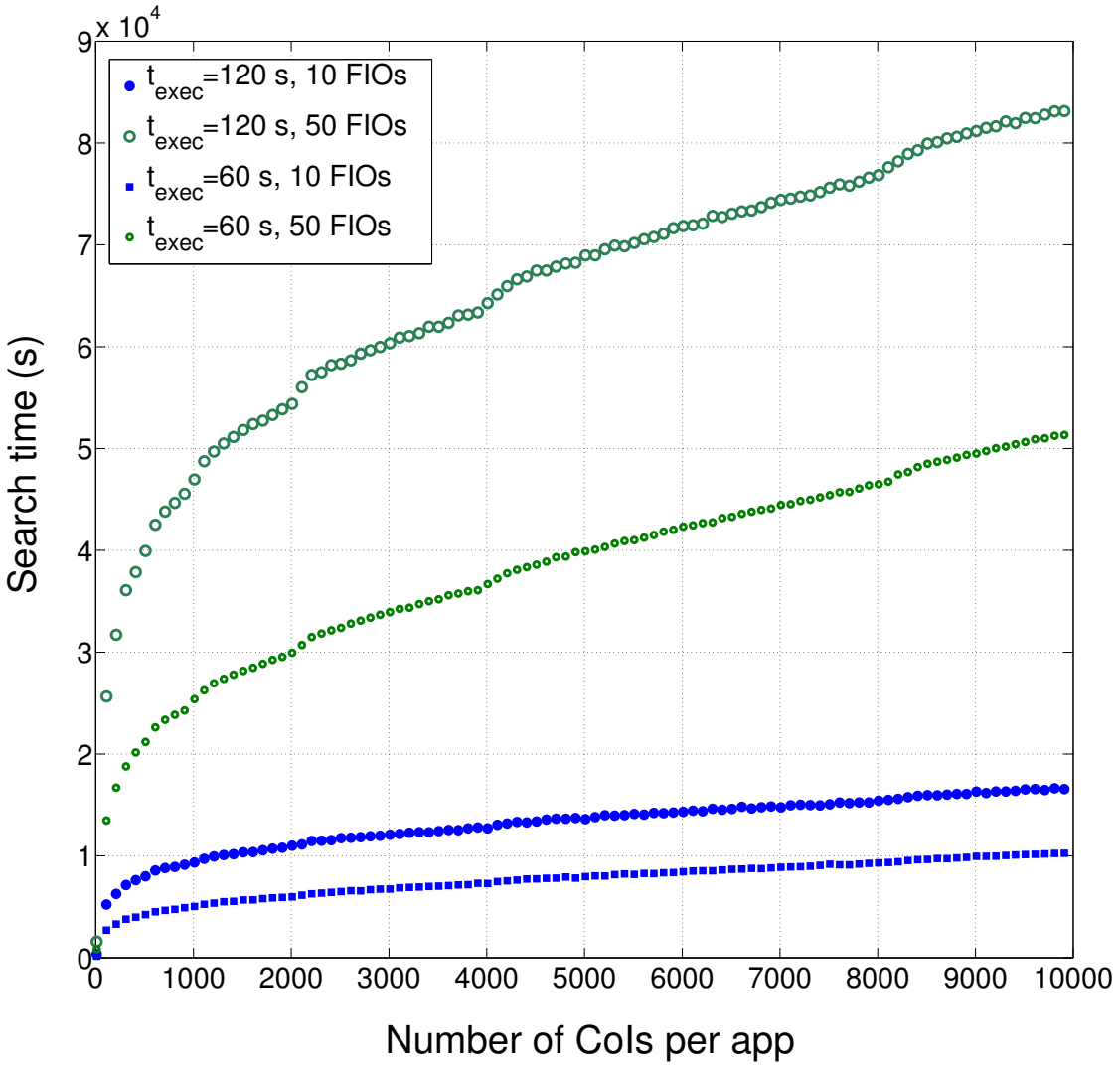
SearchComponent

$O(n + \log m)$

- $n = |CoIS|$
- $m = |FIOs|$

t = 5 minutes

- 100 CoIs
- 10 FIOs
- 120 s.



Alterdroid can automatically identify potentially malicious components hidden within apps

Conclusions

Current static analysis techniques

- Focused on inspecting code components
- Miss pieces of code hidden or obfuscated in data objects

Current dynamic analysis techniques

- Holistic vision of the behavior of an app
- Lack of attribution of behavior to components

Differential fault analysis

- Good complement to both static and dynamic analysis tools

Results: **Alterdroid**

P3: "Thwarting Obfuscated Malware via Differential Fault Analysis".

- Authors: Guillermo Suarez-Tangil, Flavio Lombardi, Juan E. Tapiador, and Roberto Di Pietro.
- *In: IEEE Computer*, vol. 47:6, pp. 24-31 (2014).
- I.F. (2012): **1.68** (**Q1**) in Computer Science.

P4: "Alterdroid: Differential Fault Analysis of Obfuscated Malware Behavior".

- Authors: Guillermo Suarez-Tangil, Juan E. Tapiador, Flavio Lombardi, and Roberto Di Pietro.
- To: *IEEE Trans. on Mobile Computing*.

Layout

Cloud-based Analysis

- Meterdroid
- Targetdroid

Contribution 4: **Meterdroid**

Cloud-based Analysis (I/II)

Detecting malware: *To cloud* or *not to cloud*?

Challenges

- Separating malicious from non-malicious activities
- On-platform analysis is very consuming
- Offloaded engines

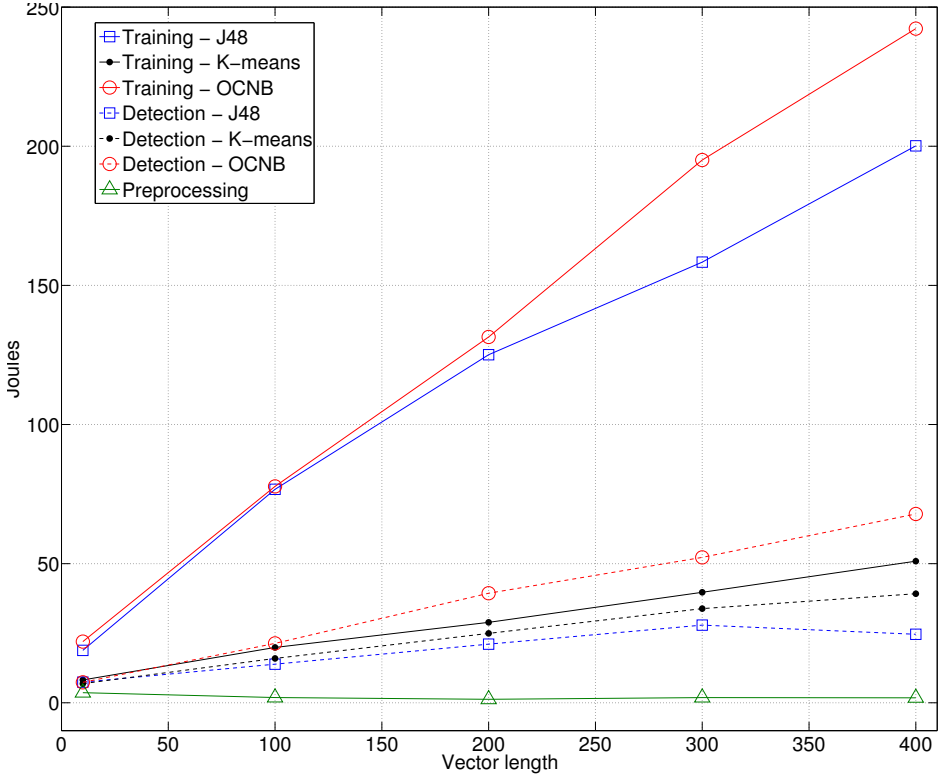
We evaluate energy-consumption trade-offs of offloading...

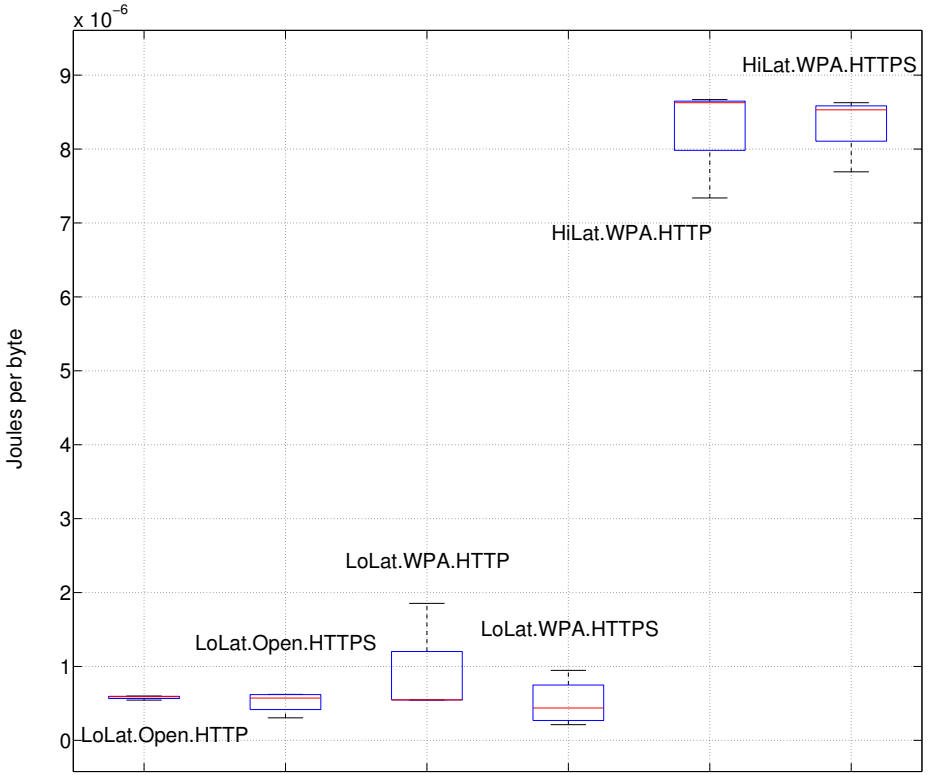
... anomaly detectors using ML systems

- Feature extraction
  - Monitoring + Processing
- Detection
  - Training + Testing

Energy consumption in Joules for all preprocessing, training, detection and communication tests

Experimental work





Different deployment strategies and their trade-offs

To cloud or not to cloud? Evaluation and results

Strategies

- Local Training
- Local Detection
- Remote Training
- Remote Detection

Parameters the detector

- Frequency of training ω_t
- Frequency of detection ω_d
- Size of the training set $|D|$
- Size of the model $|M|$
- Length of a vector $|v|$

Local Training, Local Detection

$$\omega_t |D| E_t(|v|) + \omega_d E_d(|v|)$$

Local Training, Remote Detection

$$\omega_t \left(|D| E_t(|v|) + E_c(|M|) \right) + \omega_d E_c(|v|)$$

Remote Training, Local Detection

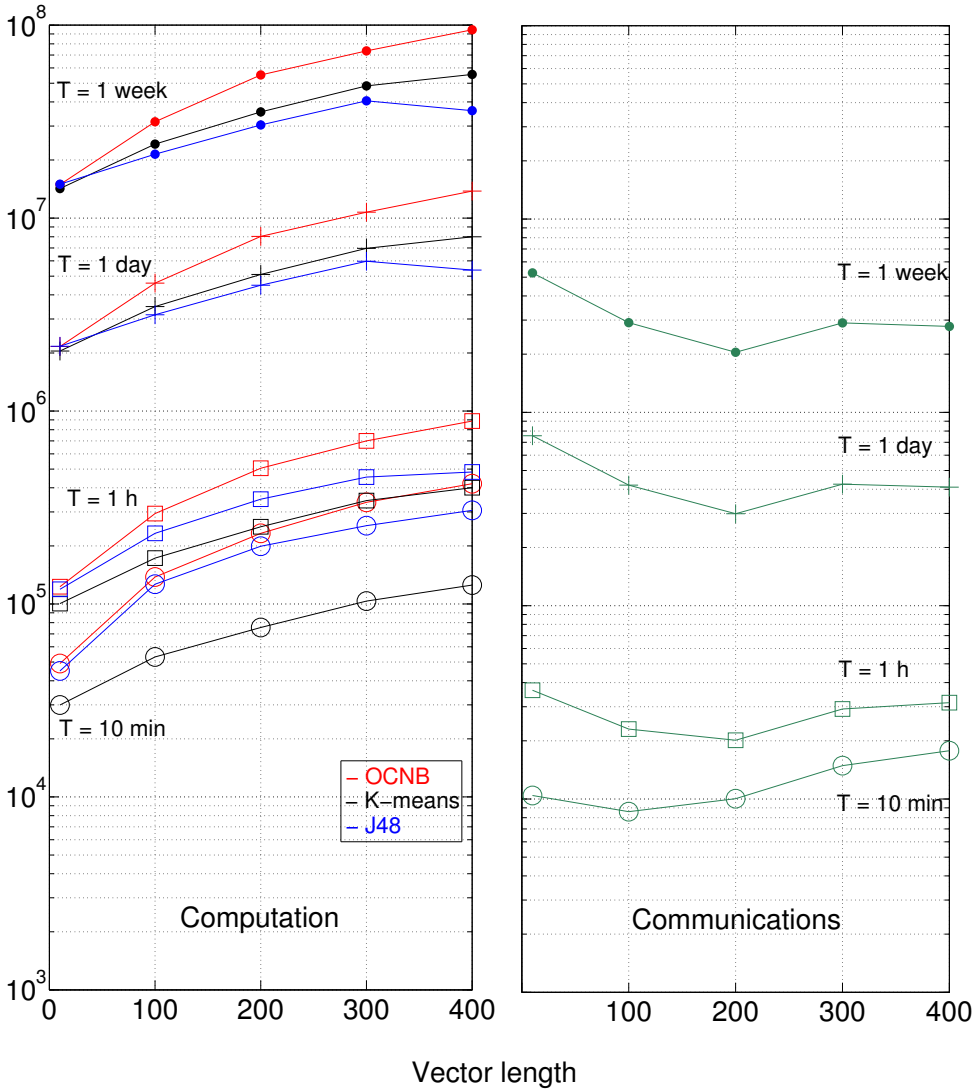
$$\omega_t \left(|D| E_c(|v|) + E_c(|M|) \right) + \omega_d E_d(|v|)$$

Remote Training, Remote Detection

$$\omega_t |D| E_c(|v|) + \omega_d E_c(|v|)$$

Case study: Repackaged malware

Local Training, Local Detection vs. Remote Training, Remote Detection



App	Total
YouTube	551.59
MX Moto	644.52
Facebook	637.27
Life battery	1.8×10^6

Table : Consumption (in Joules) of three popular apps during a time span of 10 minutes.

Our results confirm the intuition that externalized computation is the best option energy-wise

Conclusions

To offload

- Several orders of magnitude cheaper than on-platform computations
- Energy efficiency of the communications in current platforms
- Parameters related to the anomaly detection: dataset sizes and the operation frequency

Anomaly detectors are very consuming tasks

- Substantial differences among the ML algorithms tested
- They consume more than popular apps (games, OSN, ...)
- Need for more lightweight ML algorithms for on platform detection

Results: **Meterdroid**

P5: "Power-aware Anomaly Detection in Smartphones: An Analysis of On-Platform versus Externalized Operation".

- Authors: Guillermo Suarez-Tangil, Juan E. Tapiador, Pedro Peris-Lopez, and Sergio Pastrana.
- To: *Pervasive and Mobile Computing*, *submitted Feb. 2014*.
- I.F. (2012): **1.63** (**Q1**).

Contribution 5: **Targetdroid**

Cloud-based Analysis (II/II)

Detecting targeted malware in the cloud

Challenge

- Malware is becoming aware of the context
- Analysis dynamic tools fail on triggering the malware

Typical wake-up conditions

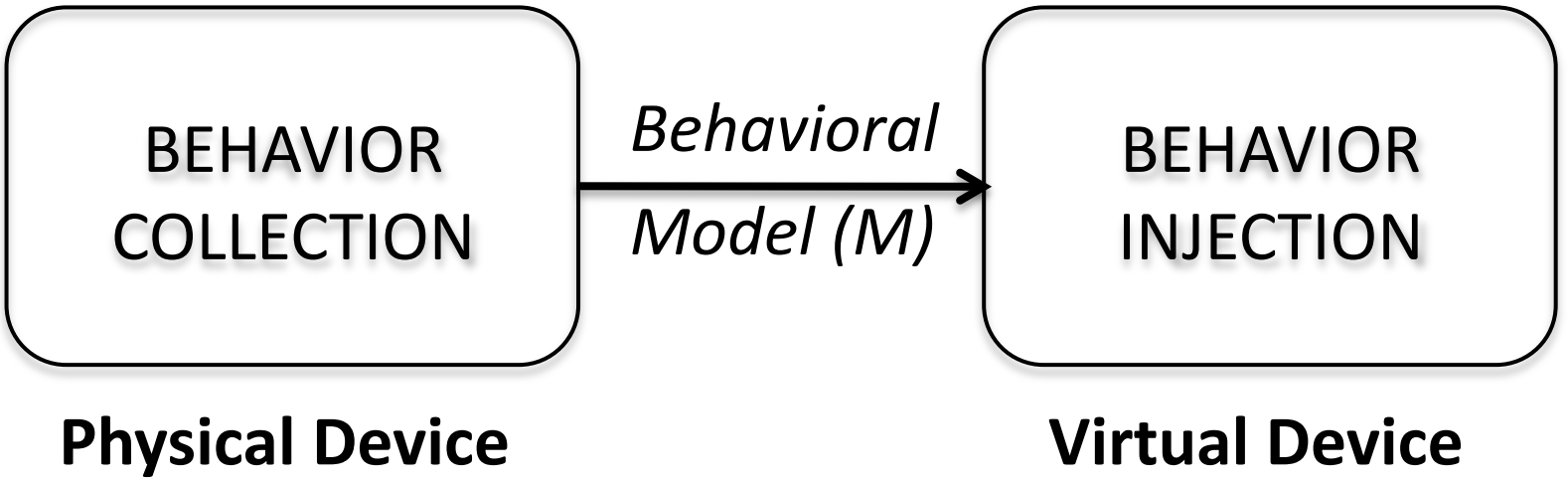
- **User present**: USB plug, screen-on, acceleration, answered incoming call...
- **Location**: Location change, near/leaving address, at a location during office hours...
- **Time**: A given time, after an event...
- **Hardware**: Power/LED status, LOCK event...
- **Config**: Apps installed, contact in agenda...

Inputs provided by the user to his device constitute a major source of stimuli for triggering certain app behaviors

Contribution: behavior-triggering model based on the user

User-centric models for triggering behaviors

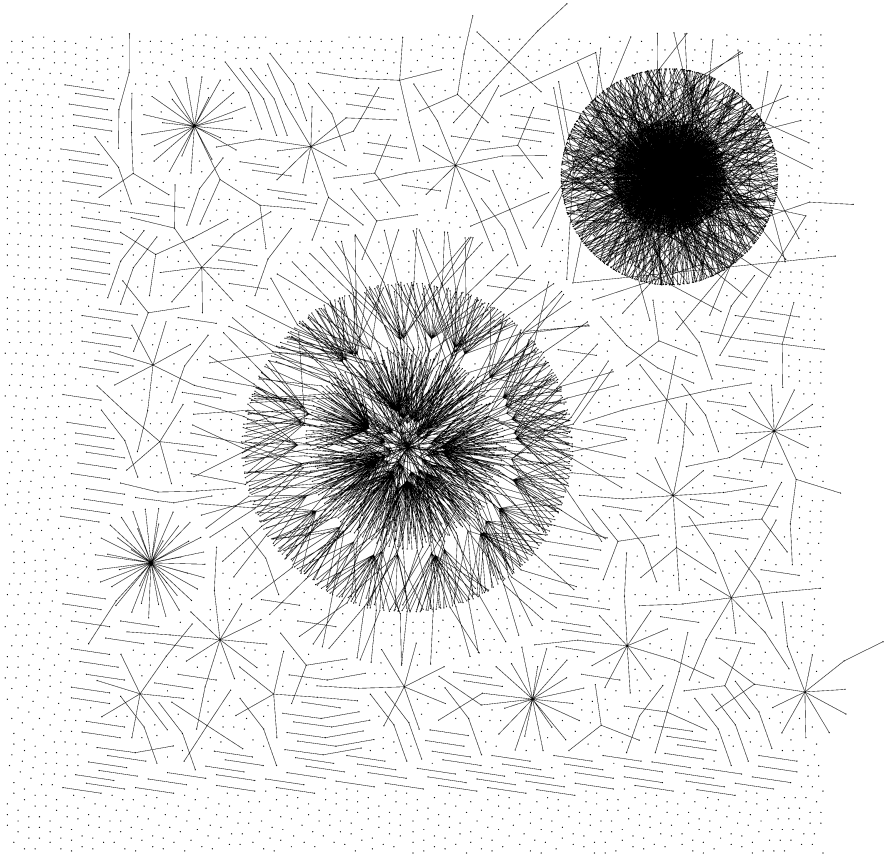
- We obtain an *actionable* model of user behavior from real users
- We test how apps behave when users execute them in some context

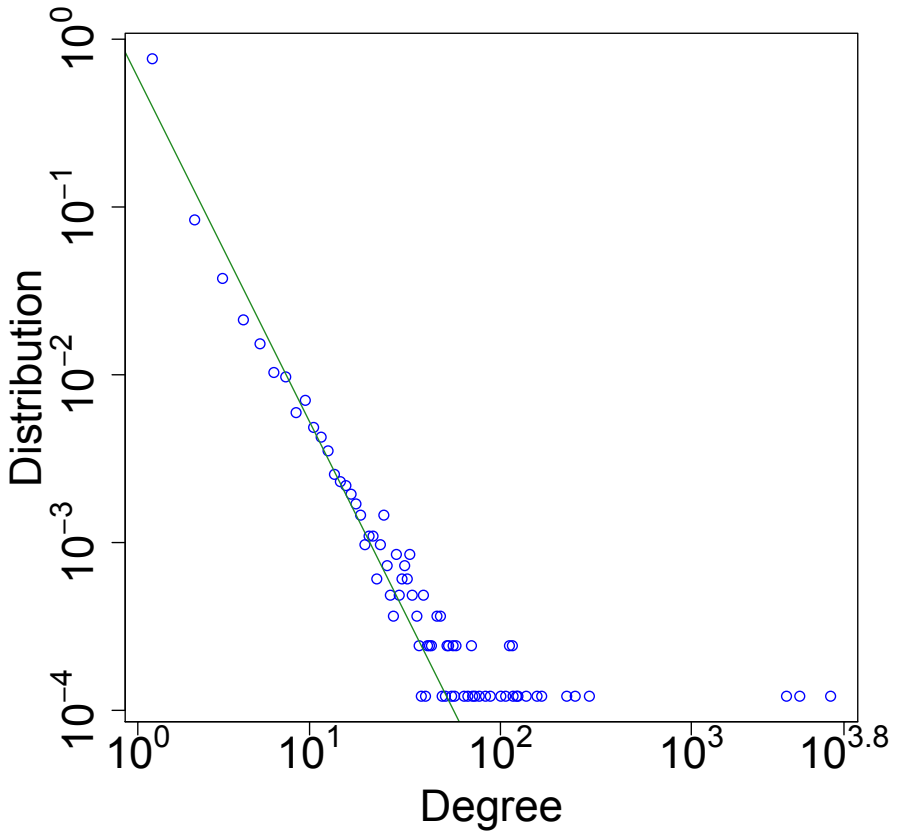


Most events have an extremely low number of *neighbors*

Experimental Work

Input events and their degree distribution for a user interacting with an Android platform





Behavior-triggering stochastic models

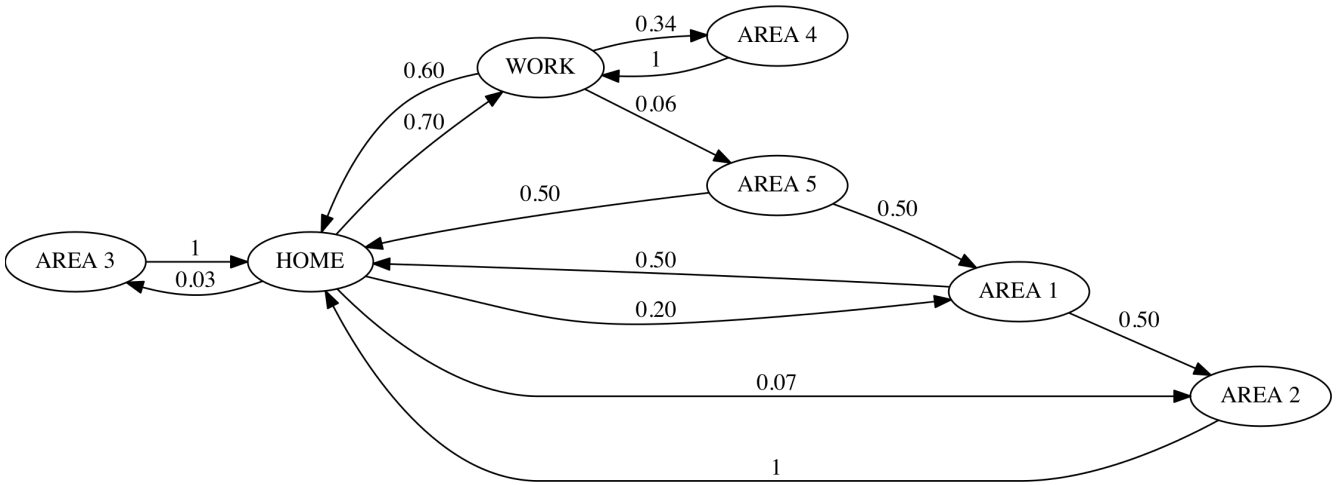
User-centric models

Behavioral Models

- Usage Patterns: $\mathbf{u} = \langle \epsilon_1, \epsilon_2, \ldots, \epsilon_k \rangle, \quad \epsilon_i \in \mathcal{E}$
- Context Patterns: $\mathbf{t} = \langle c_1, c_2, \ldots, c_l \rangle, \quad c_i \in \mathcal{C}$

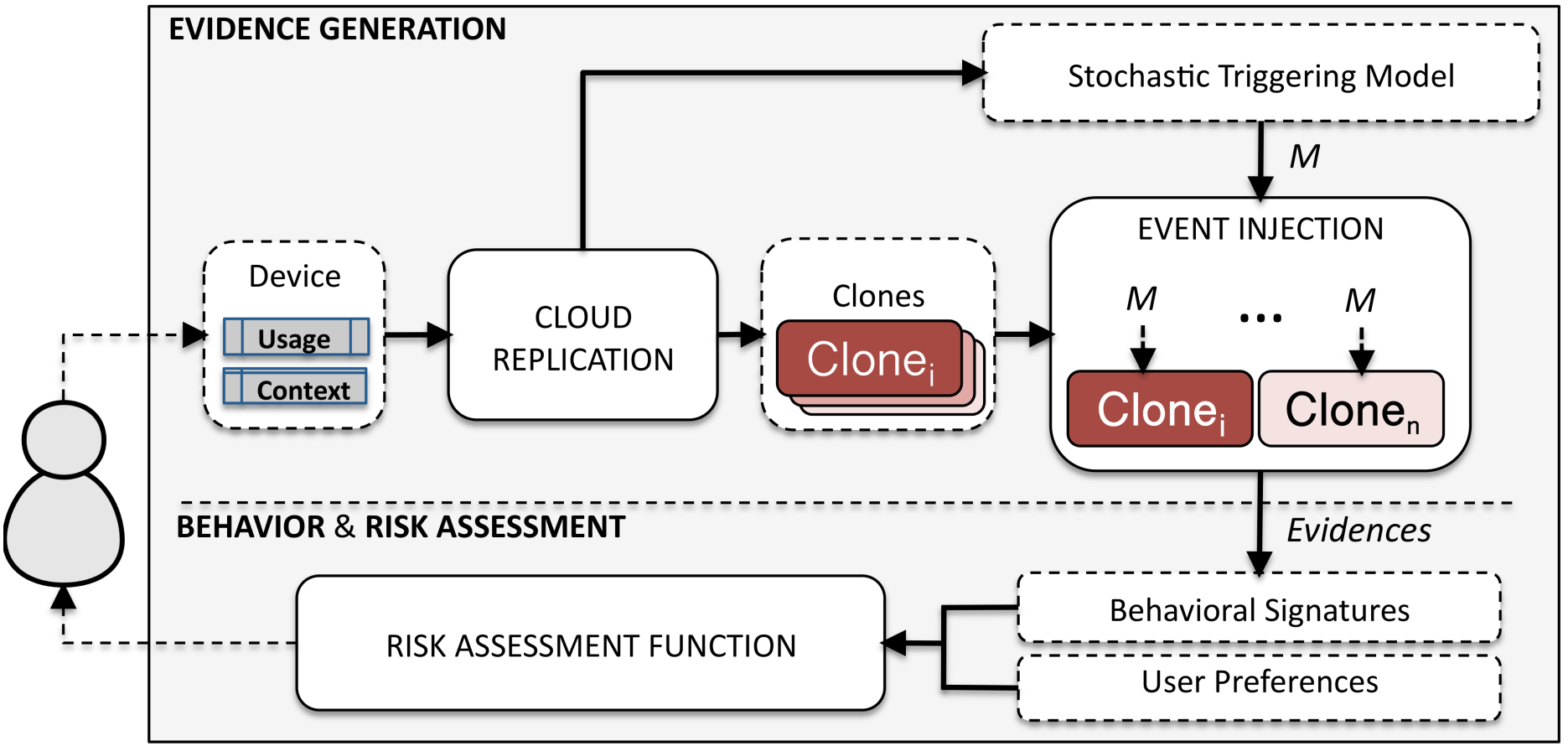
Stochastic Triggering Model

- Markov Chains: discrete-time first-order Markov process



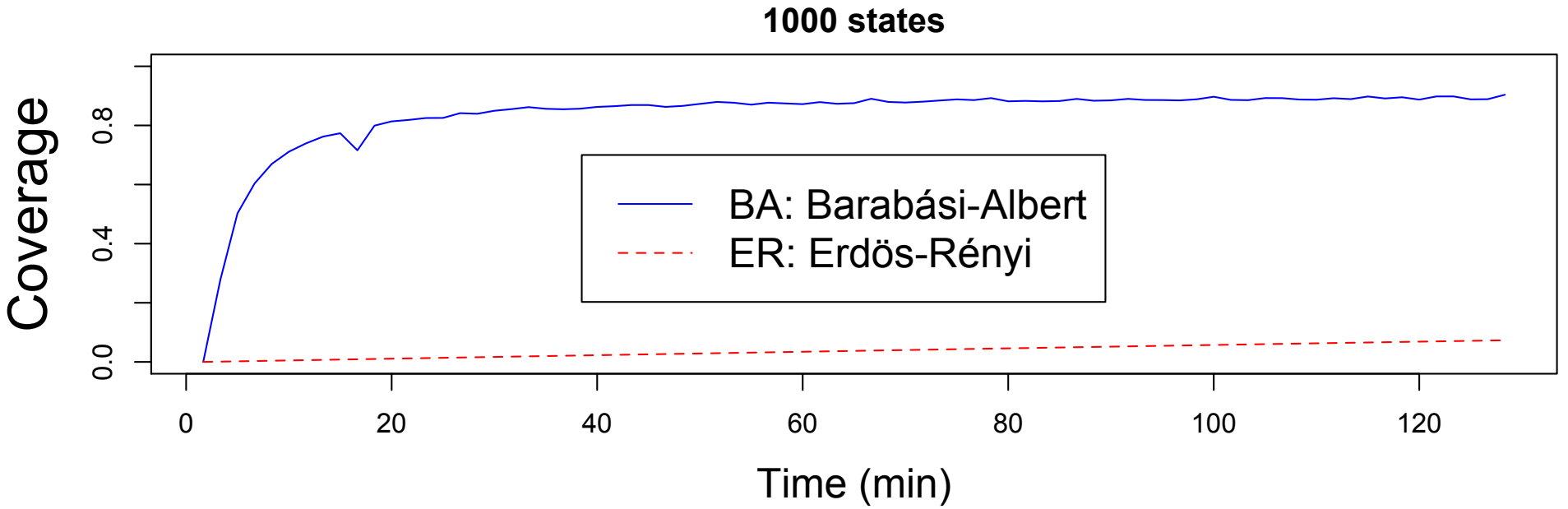
Targetdroid allows to detect targeted malware

Our system



It takes 3s to inject an SMS and 0.01s to inject a location

Evaluation



	Number of parallel clones for 4000 states									
	1	2	3	4	5	6	7	8	9	10
10 min.	42%	60%	68%	73%	76%	79%	81%	81%	82.5%	83.4%
60 min.	79%	86%	89%	90%	90%	91%	91%	91%	91%	95%
120 min.	84%	87%	88%	88%	93%	93%	93%	93%	93%	93%

Stochastic behavioral-triggering is a robust building block for thwarting targeted malware

Conclusions

Detecting targeted malware via behavioral analysis requires smart inputs

- Determining malware's triggering conditions is a complex problem
- Behavioral analysis requires user-centric inputs

Markov model chains

- Modeling inputs as Markov chains reduces the search complexity
- Offers an effective representation of the usage and context patterns
- Risk assessment is key to automatically detect targeted malware

Cloud clone replication systems

- Cloud infrastructure empowers devices with powerful detection
- Parallel testing is paramount to detect complex activation patterns

Results: **Targetdroid**

P6: "Detecting Targeted Smartphone Malware with Behavior-Triggering Stochastic Models".

- Authors: Guillermo Suarez-Tangil, Mauro Conti, Juan E. Tapiador, and Pedro Peris-Lopez.
- To: *European Symposium On Research In Computer Security* (ESORICS), *September 2014*.
- Rank (2013): CORE **A** in Computer Software.

Thwarting **smart malware** is a thriving research area

Conclusions

Smart malware analysis

- Impressive **growth** of goodware and malware
- Increase in the **sophistication**

This Thesis contributes to the analysis of **Smart Malware**

Discussion

Addresses several fundamental issues when **automating smart malware analysis in large-scale markets**

- **01**: Study the evolution of malware and its analysis/detection
- **02**: Develop methods for better analyzing malware in large markets
  - Intelligent instruments to automate the analysis
- **03**: Facilitate the analysis of complex smart malware

	O1	O2	O3
Evolution	⊗	⊗	⊗
Dendroid		⊗	
Alterdroid		⊗	⊗
Meterdroid			⊗
Targetdroid			⊗

All contributions resulting from this Thesis have been sent to **top ranked** journals and conferences of the area

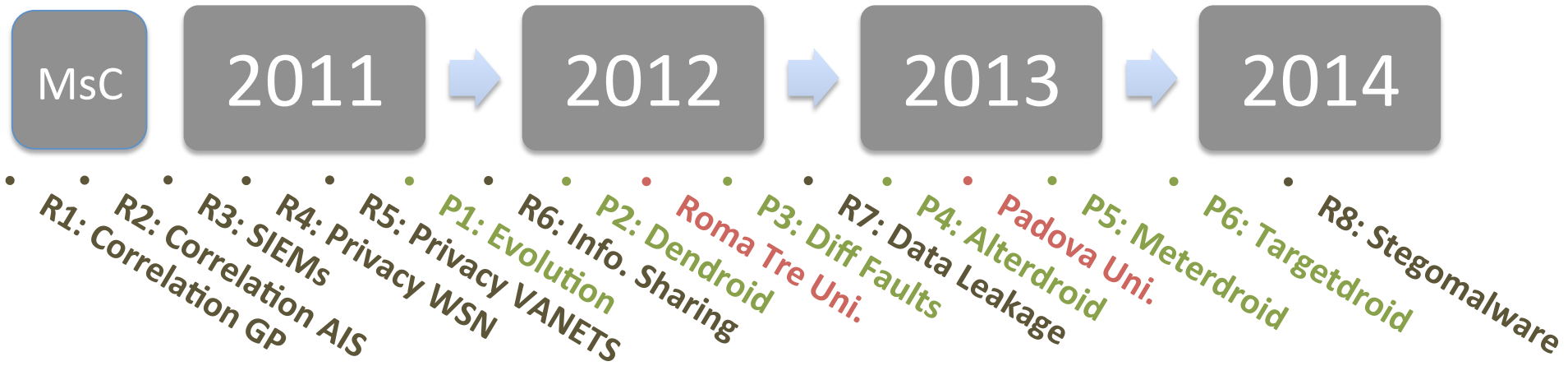
Results

	Publications		Indexes	Rank
Journals	*Published*:	3	JCR	Q1
	Submitted:	2	JCR	Q1
Conferences	*Published*:	1	CORE	A
Others	*Copyrighted*:	4	–	–
Total		10		

Table : Summary of the publications of this Thesis and the citation indexes of their corresponding publication venue.

Summary of the PhD training

Timeline: Publications + Visits + Related



- **R1**. Automatic Rule Generation Based on GP for Event Correlation (CISIS'10)
- **R2**. Artificial Immunity-based Correlation System (SECRYPT'11)
- **R3**. Providing SIEM systems with self-adaptation (INFFUS'13)
- **R4**. An Experimental Comparison of Privacy Methods in WSN (SENSIG'10)
- **R5**. A privacy-respectful telematic verification system (MOBIQUITOUS'11)
- **R6**. Information Sharing Models for Cooperative Cyber Defence (CYCON'13)
- **R7**. Hindering Data Theft with Encrypted Data Trees (*sub.* JSS'13)
- **R8**. Hindering Malware Detect. via Steganography in Smartdevices (*sub.* INSCRYPT'14)

Smart Malware still pose many challenges to be addressed with novel solutions

Future work

- **Stegomalware**: We have crawled several legitimate markets and mining their components finding evidences of apps using steganalysis
  - **Submitted**: *Hindering Malware Detection via Steganography*
- **Trusted Software**: Most users do not pay much attention to the permissions and/or the reputation of the app
- **Malware in Other Smart Devices**: Malware will also hit other smart devices as soon as they appear, with special focus on medical devices
- **Forensics Analysis**: Analyzing *post-mortem* malware can be useful to better improve future specimens using similar infection vector
- **Cooperative security**: Mutually monitoring schemes could be interesting, where each device monitors the behavior of others to detect compromise

MINING STRUCTURAL AND BEHAVIORAL PATTERNS IN SMART MALWARE

Author: Guillermo Suarez-Tangil
Supervisors: Juan E. Tapiador, Pedro Peris-Lopez

COmputer SEcurity (COSEC) Lab
Computer Department — Universidad Carlos III de Madrid

PhD Defense – October 2014
Madrid, Spain