Applying Behavioral Detection on Android-based Devices

Asaf Shabtai, Yuval Elovici

Department of Information Systems Engineering,
Deutsche Telekom Laboratories @ Ben-Gurion University,
Ben-Gurion University, Israel

Mobilware 2010

Outline

- Motivation
- Method and goals
- Evaluation
- Concussions

Motivation

Smart mobile devices security

- Smartphones have evolved into sophisticated, compact minicomputers
- A platform for running various applications
- Smartphones usage is on the raise (43%/year)*
- Stores sensitive/private information and services
- Susceptible to various PC-like types of attacks
- Open-platform; Open-source

*Forecast: Mobile Devices, Worldwide, 2003-2014, Gartner Inc.

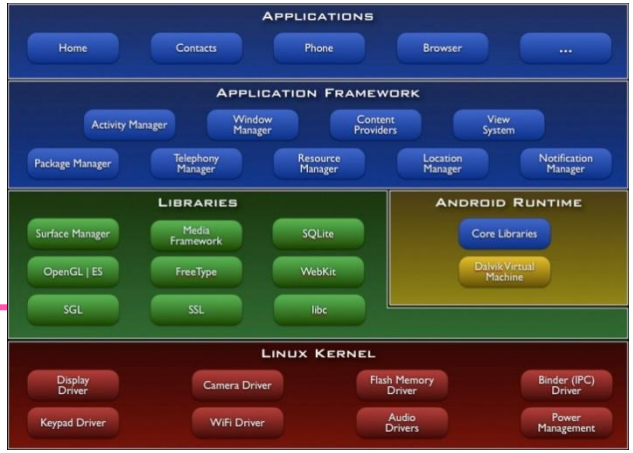
Motivation

Smart mobile devices security

- The importance of security mechanisms is not yet understood
- Security mechanisms are not sufficient
  - Antivirus – mostly signature-based
  - Protecting functionalities (e.g., in Android)
- Jail-breaking devices
- Variety of platforms

Motivation

Smart mobile devices security

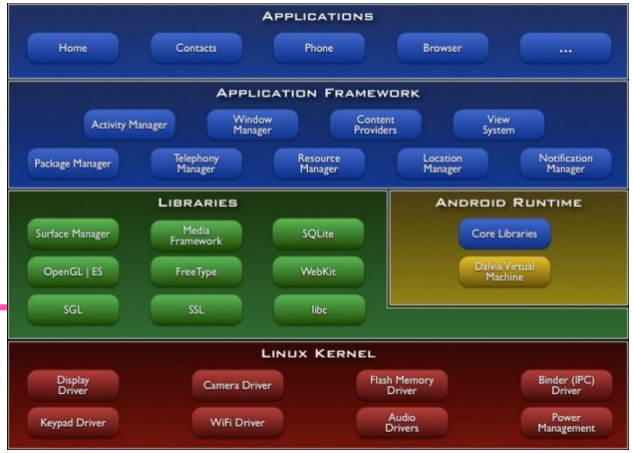


- Android security review*
  - Vulnerability in one of the kernel modules or core libraries
  - Application-level permissions mechanism
    - Call, access contacts, geographical location, access the internet...
    - adb install, “all or none”, applications can share permissions

*Shabtai, A. et al. **Google Android: A Comprehensive Security Assessment.** *IEEE Security and Privacy*, 8(2):35-44, March/April 2010

Motivation

Smart mobile devices security



- According to SMobile Systems
  "Threat Analysis of the Android Market", June 2010
  - 48,694 applications in the Android Market, (68% of all applications in the Market)
  - 20% request permissions to access private or sensitive information
  - 5% has the ability to place a call or send SMS to any number
  - Brick, read or use the authentication credentials from another service or application
  - 40% are suspicious because they request two or more of the permissions

Proposed method

- Part of an on-going research that we conduct for the past 2 years, focusing on protecting smart mobile devices – specifically Android
  - Intrusion detection, firewalls, malware detection, static code analysis, access control (e.g., SELinux*), data leakage prevention
- Behavioral analysis on Android devices
  - Extract various parameters from the system
  - Apply machine learning algorithms to derive the device health

*Shabtai, A. Fledel, Y. Elovici, Y. **Securing Android-Powered Mobile Devices Using SELinux.** *IEEE Security and Privacy*, 8(3):36-44, May/June 2010

Research goals

- Evaluate various detection algorithms
- Understanding the feasibility of running these methods on Android devices
  - Detection capabilities
  - Resource consumption
- What features are good for detection
- Evaluated the ability to differentiate between different types of applications (e.g., games and tools), which is expected to provide a positive indication about the ability of such methods to learn and model the behavior applications and potentially detect malware

Related works

- Anomaly detection focusing on detection of fraudulent use of operator services; based on phone call features [Moreau, 1997], [Samfat, 1997]
- Battery/energy feature [Kim, 2008], [Buennemeyer, 2008], [Nash, 2005], [Jacoby, 2006], [Miettinen, 2006]
- [Schmidt, 2009] proposed monitoring smartphones for anomaly detection and sending the features to remote server for analysis
- Signature-based detection [Yap, 2005], [Bose, 2008], [Kim, 2008], [Jacoby, 2006], [Samfat, 1997]
- Keystroke dynamics [Hwang, 2009]
- Prevention by enhancing Android security at the application level permissions: Kirin system [Enck, 2009] and SAINT [Ongtang, 2009]
- Static analysis/code verification ScanDroid [Adam, 2009]

The “Andromaly”

- A lightweight Host-based Intrusion Detection System for Android-based devices
- Providing real-time, monitoring, collection, preprocessing and analysis of various system metrics
- Open framework – possible to apply different types of detection techniques

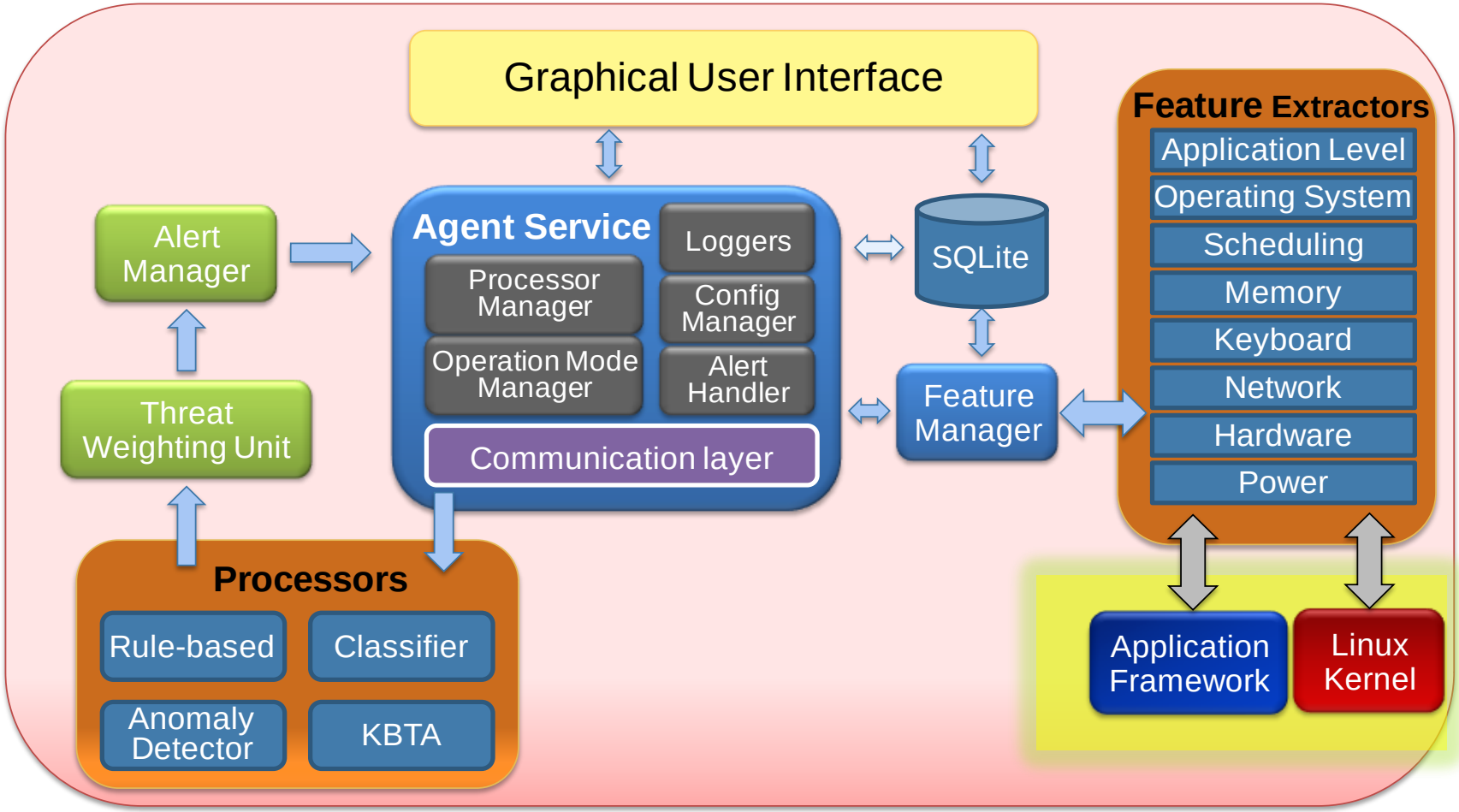
  Shabtai, A. Kanonov, U. Elovici, Y. **Intrusion Detection on Mobile Devices Using the Knowledge Based Temporal-Abstraction Method**. *Journal of Systems and Software*, 83(8):1524-1537, 2010
- Threat assessments (TAs) are weighted and smoothed to avoid instantaneous false alarms
- An alert is matched against a set of automatic/manual countermeasures
  - Log, notify, uninstall, kill process, disconnect radios, encrypt data, change firewall policies

The “Andromaly”

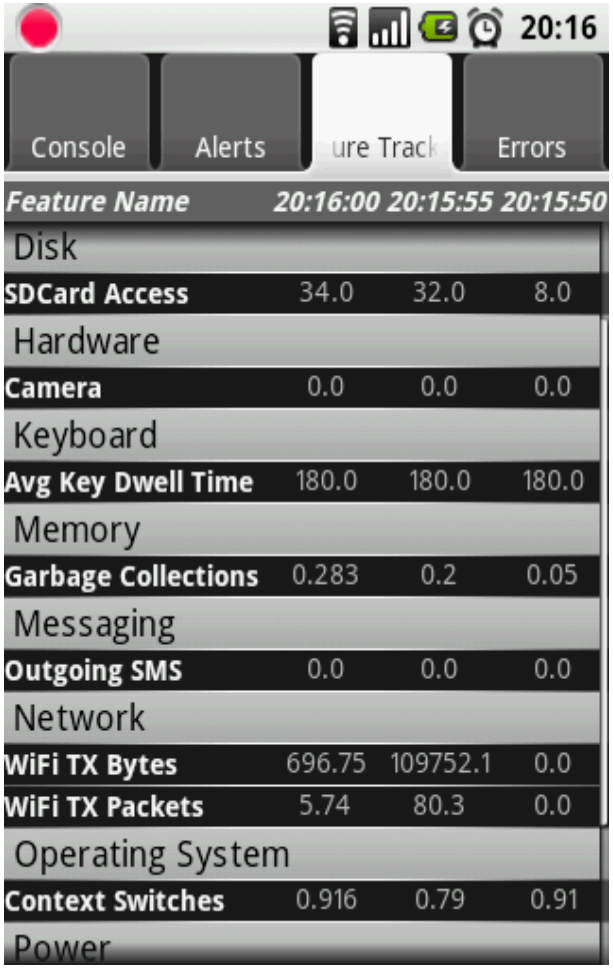
- Detection (or only reporting) at a centralized location
  - Collaborative detection
  - Detection of malware propagation patterns across a community of mobile devices
  - Reporting suspicious behavior of applications to the Android Market

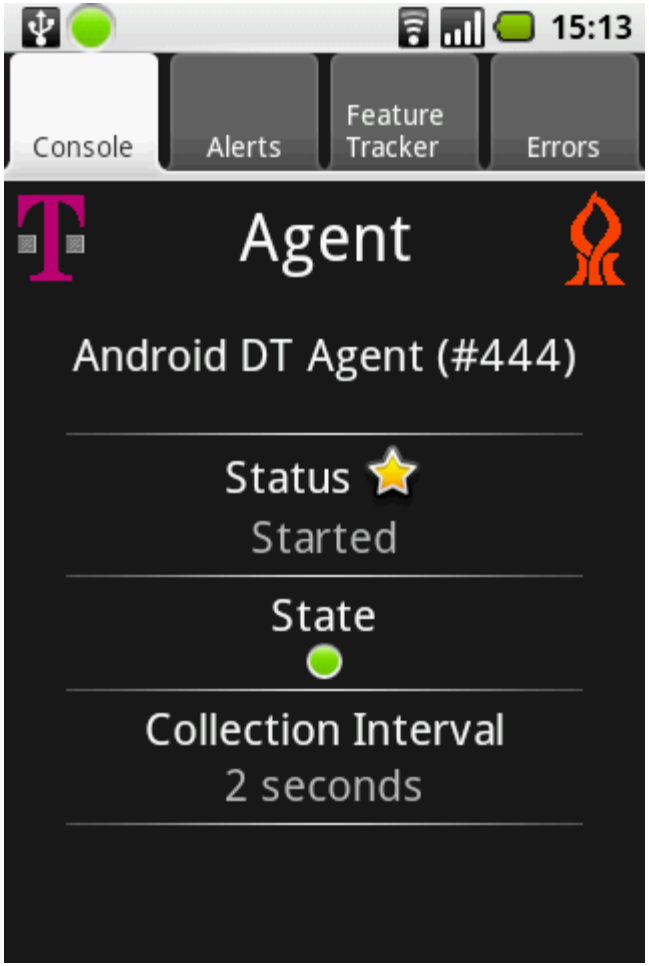
    “The Android Market was chosen to be the place for reporting security issues by users” (Google’s Android Security Leader, USENIX 2009)

The “Andromaly” architecture



Few screenshots...

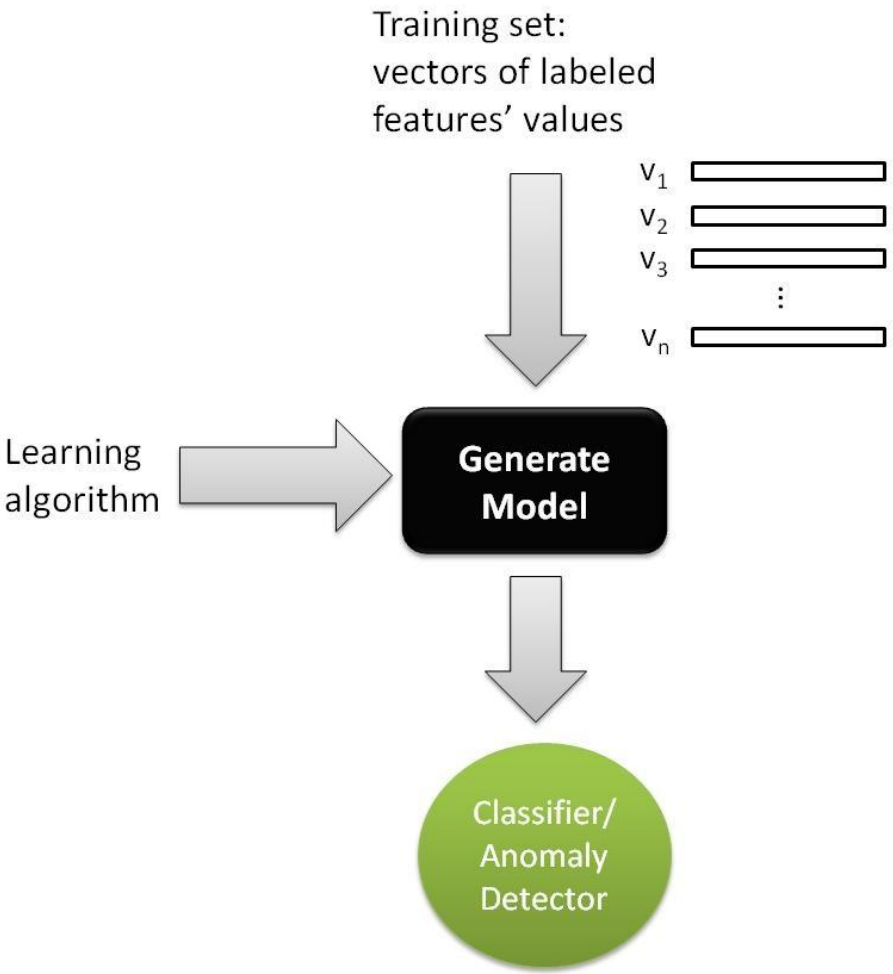




Few screenshots...

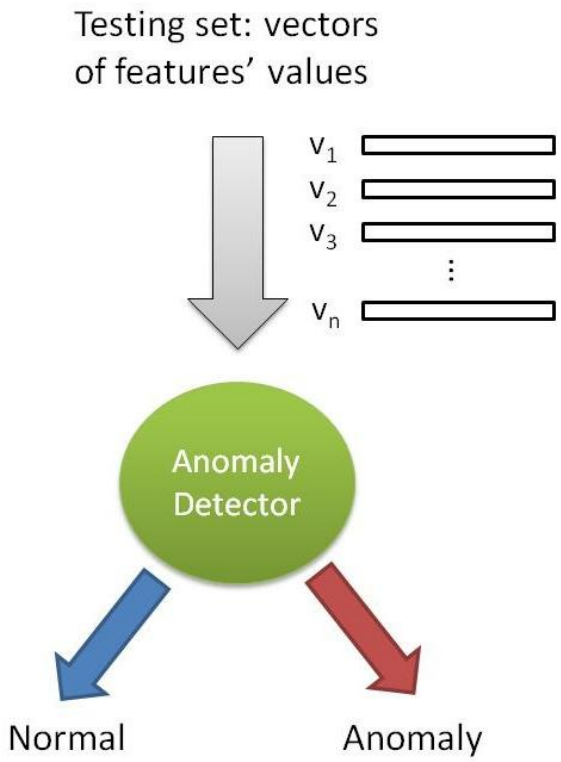
Detection method

Learning phase



Detection method

Detection phase



Evaluation

Preparation of the data-sets

- The applications were installed on 5 Android G1devices (each device has one user only)
- Each user activate each application for 10 minutes
- In the background the Android agent was running and logging data (feature vectors) on the SD-card (88 features each 2 seconds)
- The feature vectors were added to our data-set and labeled with the device id, application name and class (game/tool)

Evaluation

Collected features

Collected Features (88)		
Touch screen:	**Memory:**	**Network:**
Avg_Touch_Pressure	Garbage_Collections	Local_TX_Packets
Avg_Touch_Area	Free_Pages	Local_TX_Bytes
Keyboard:	Inactive_Pages	Local_RX_Packets
Avg_Key_Flight_Time	Active_Pages	Local_RX_Bytes
Del_Key_Use_Rate	Anonymous_Pages	WiFi_TX_Packets
Avg_Trans_To_U	Mapped_Pages	WiFi_TX_Bytes
Avg_Trans_L_To_R	File_Pages	WiFi_RX_Packets
Avg_Trans_R_To_L	Dirty_Pages	WiFi_RX_Bytes
Avg_Key_Dwell_Time	Writeback_Pages	**Hardware:**
Keyboard_Opening	DMA_Allocations	Camera
Keyboard_Closing	Page_Frees	USB_State
Scheduler:	Page_Activations	**Binder:**
Yield_Calls	Page_Deactivations	BC_Transaction
Schedule_Calls	Minor_Page_Faults	BC_Reply
Schedule_Idle	**Application:**	BC_Acquire
Running_Jiffies	Package_Changing	BC_Release
Waiting_Jiffies	Package_Restarting	Binder_Active_Nodes
CPU Load:	Package_Addition	Binder_Total_Nodes
CPU_Usage	Package_Removal	Binder_Ref_Active
Load_Avg_1_min	Package_Restart	Binder_Ref_Total
Load_Avg_5_mins	UID_Removal	Binder_Death_Active
Load_Avg_15_mins	**Calls:**	Binder_Death_Total
Runnable_Entities	Incoming_Calls	Binder_Transaction_Active
Total_Entities	Outgoing_Calls	Binder_Transaction_Total
Messaging:	Missed_Calls	Binder_Trns_Complete_Active
Outgoing_SMS	Outgoing_Non_CL_Calls	Binder_Trns_Complete_Total
Incoming_SMS	**Operating System:**	**Leds:**
Outgoing_Non_CL_SMS	Running_Processes	Button_Backlight
Power:	Context_Switches	Keyboard_Backlight
Charging_Enabled	Processes_Created	LCD_Backlight
Battery_Voltage	Orientation_Changing	Blue_Led
Battery_Current		Green_Led
Battery_Temp		Red_Led
Battery_Level_Change		
Battery_Level		

Evaluation

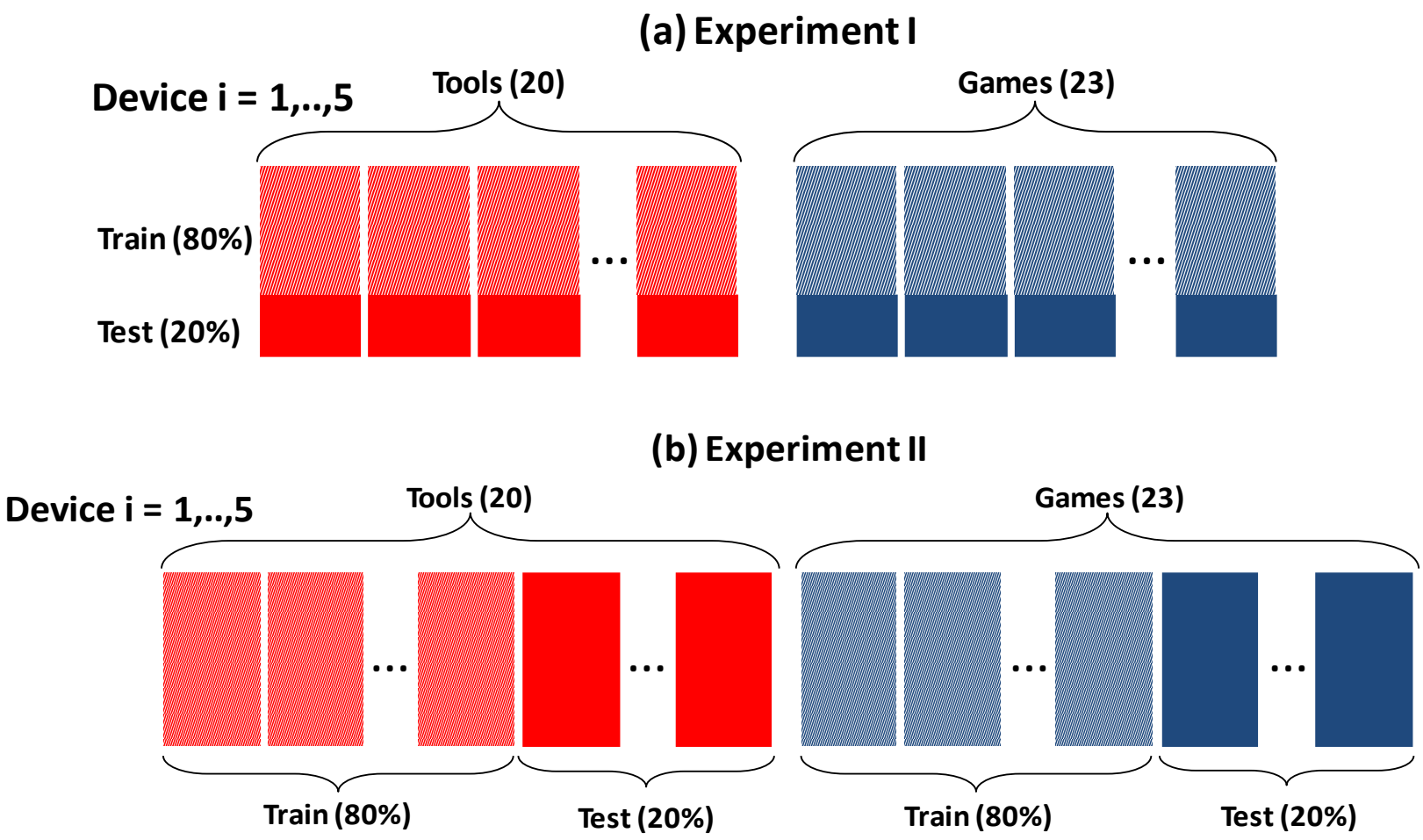
Used applications

- 23 games and 20 tools

	Application	Device1	Device2	Device3	Device4	Device5
Games	abduction	322	197	205	257	341
	armageddonoid	343	222	198	264	326
	battleformars	544	208	269	294	582
	bonsai	355	287	267	253	304
	breadfactory	300	208	241	246	291
	connect4	308	204	225	249	289
	flyinghigh	279	250	206	234	365
	froggo	308	185	241	263	301
	hangemhigh	342	348	274	255	339
	labyrinthlite	317	297	244	252	324
	lexic	354	266	222	273	347
	minesweeper	342	267	249	285	541
	mushroom	251	172	193	199	209
	pairup	410	275	298	259	582
	picacrossexpress	300	357	255	302	332
	smarttactoe	298	233	254	262	300
	snake	294	254	250	233	314
	solitaire	334	378	281	288	439
	switcher	303	231	269	236	395
	tankace	336	287	250	262	330
	throttlecopter	321	230	228	210	280
	trap	454	245	189	251	398
	wordpops	301	314	302	320	267
Tools	browser	307	274	218	336	300
	calculator	296	251	266	276	365
	calendar	319	233	250	270	341
	camera	302	251	249	260	294
	contacts	303	230	218	256	680
	email	219	250	445	275	339
	im	371	245	238	251	385
	iofilemanager	295	235	241	289	284
	maps	342	216	245	278	429
	messaging	322	247	255	278	296
	music	304	251	272	256	343
	mytracks	356	233	269	271	543
	noteeverything	329	212	287	275	408
	oxforddictionary	323	275	272	283	374
	pdfviewer	280	249	248	263	319
	phonalyzer	304	240	268	290	318
	phone	125	224	140	110	244
	tasks	300	226	250	263	302
	voicememo	312	230	270	253	269
	weather	372	242	272	269	297

Evaluation

Experiments



Evaluation

Feature selection

- Large number of features present several problems
  - Misleading the learning algorithm
  - Causing over-fitting to the data and therefore reducing generalization capabilities
  - Increasing complexity and run-time

- Three feature selection methods were applied to the data-set: Information Gain (IG), Chi Square (CS), Fisher Score (FS)

- We chose for each feature selection the top: 10, 20 and 50 best features

Evaluation

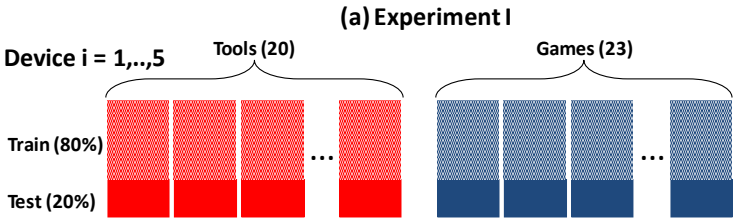
Research questions

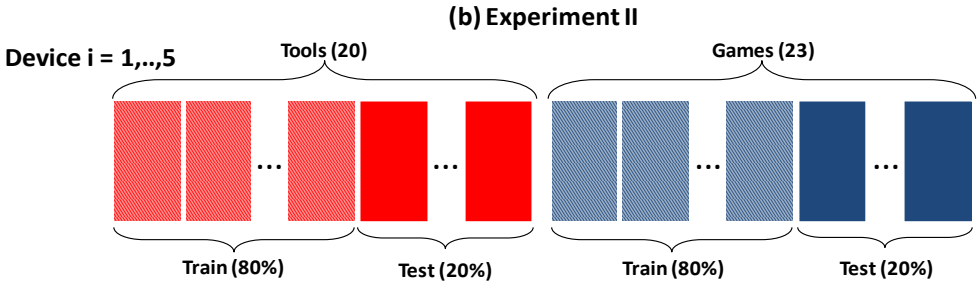
- Is it possible to detect unknown instances of known application types on Android devices?
- Which classifier is most accurate in detecting malware on Android devices: Decision Tree (DT), Naïve Bayes (NB), Bayesian Networks (BN), k-Means, Histogram or Logistic Regression (LR)?
- Which number of extracted features and feature selection method yield the most accurate detection results: 10, 20 or 50 top- features selected using Chi-Square, Fisher Score or InfoGain?
- What are the specific features that yield the maximum detection accuracy?

Evaluation

Experiments

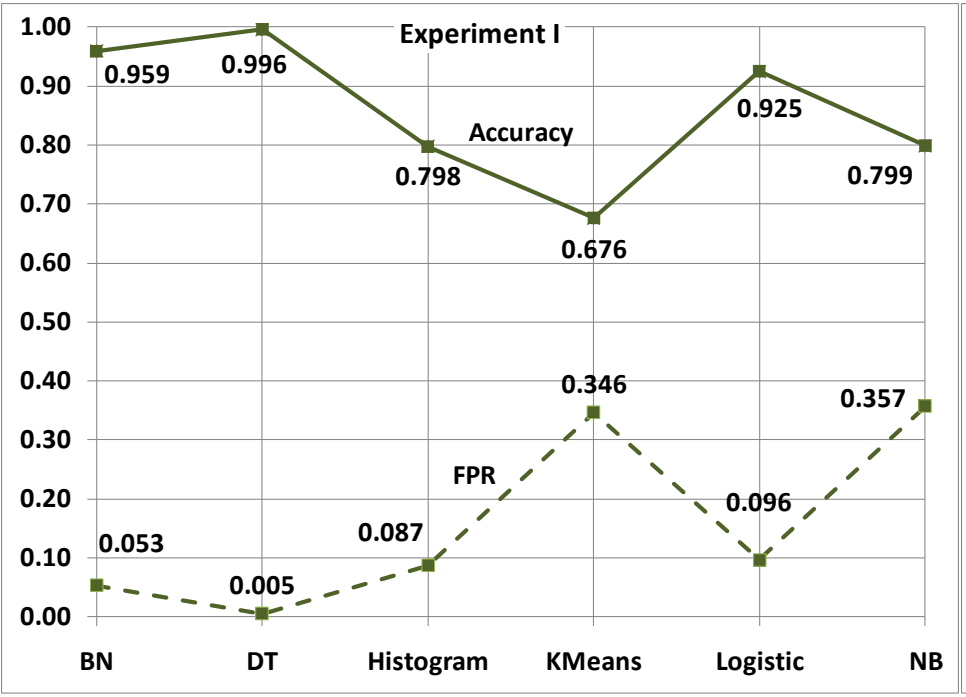
Exp.	# of detection algorithms	# of feature selection methods	# of top feature groups	# of devices	# of iterations	Total number or runs	Testing on applications not in training set
I	6	3	3	5	20	5,400	-
II	6	3	3	5	20	5,400	+

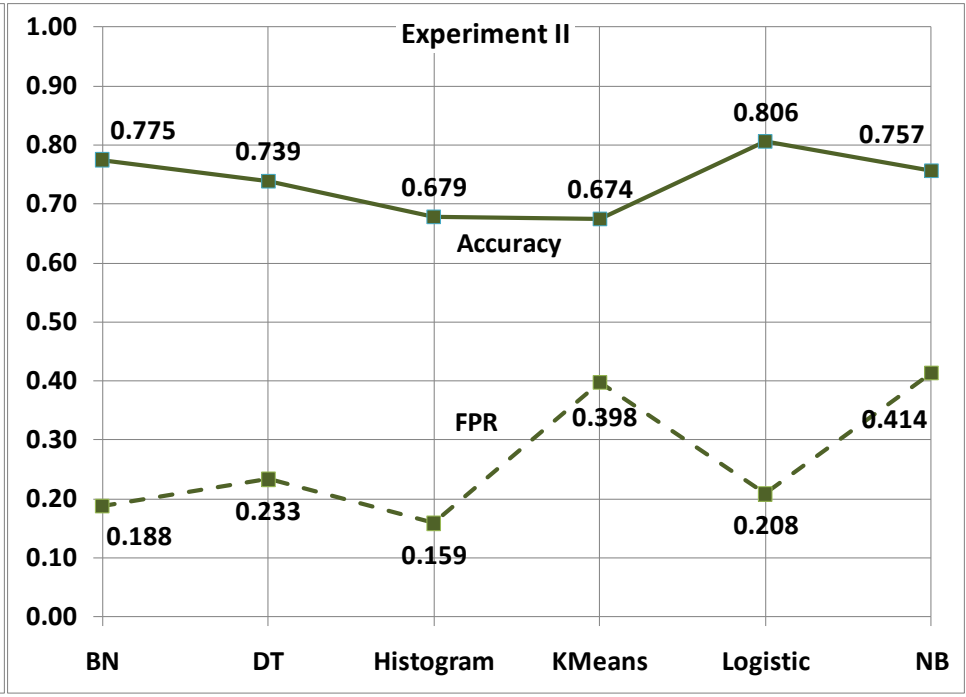




Evaluation

Results - classifiers





- Best classifiers: Decision Tree, Logistic Regression and Bayesian Networks
- The five devices exhibited similar results

Evaluation

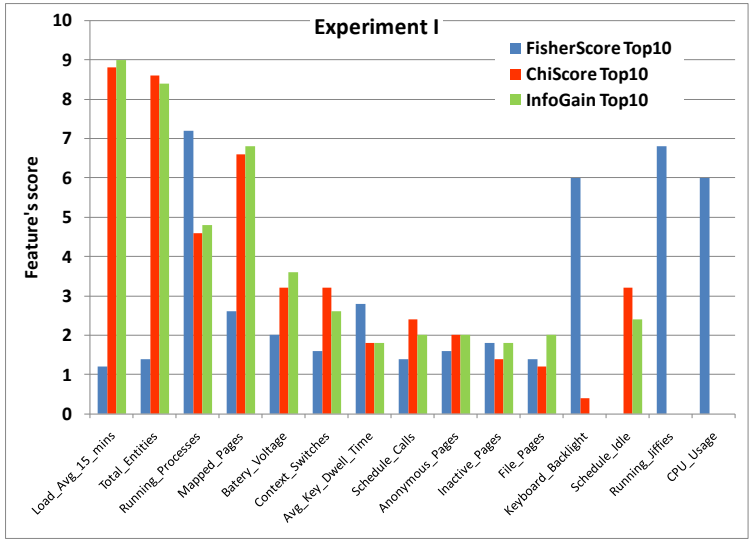
Results – feature selection

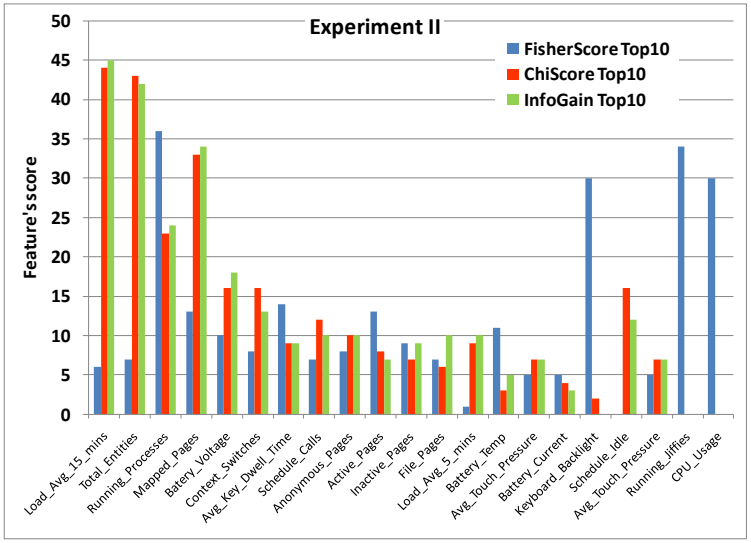
Exp	Feature Selection Method	FPR			Accuracy		
		10	20	50	10	20	50
I	ChiSquare	0.160	0.134	0.172	0.852	0.876	0.850
	FisherScore	0.152	0.174	0.167	0.857	0.860	0.857
	InfoGain	0.155	**0.129**	0.174	0.850	**0.877**	0.850
II	ChiSquare	0.270	0.258	0.280	0.732	0.751	0.725
	FisherScore	**0.250**	0.263	0.268	**0.750**	0.750	0.735
	InfoGain	0.265	0.263	0.282	0.729	0.747	0.724

Evaluation

Results – selected features

- Features were scored according to
  - their rank assigned by feature selection methods
  - number of devices that the feature was effective
- Highest scored features: Load_Avg_15_mins, Total_Entities, Running_Processes, Mapped_Pages, Battery_Voltage, Context_Switches, Schedule_Calls, Anonymous_Pages





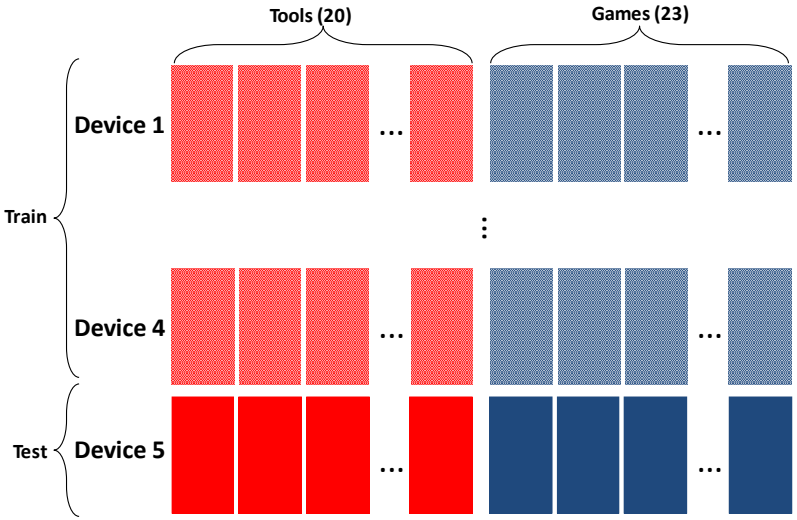
Summary and conclusions

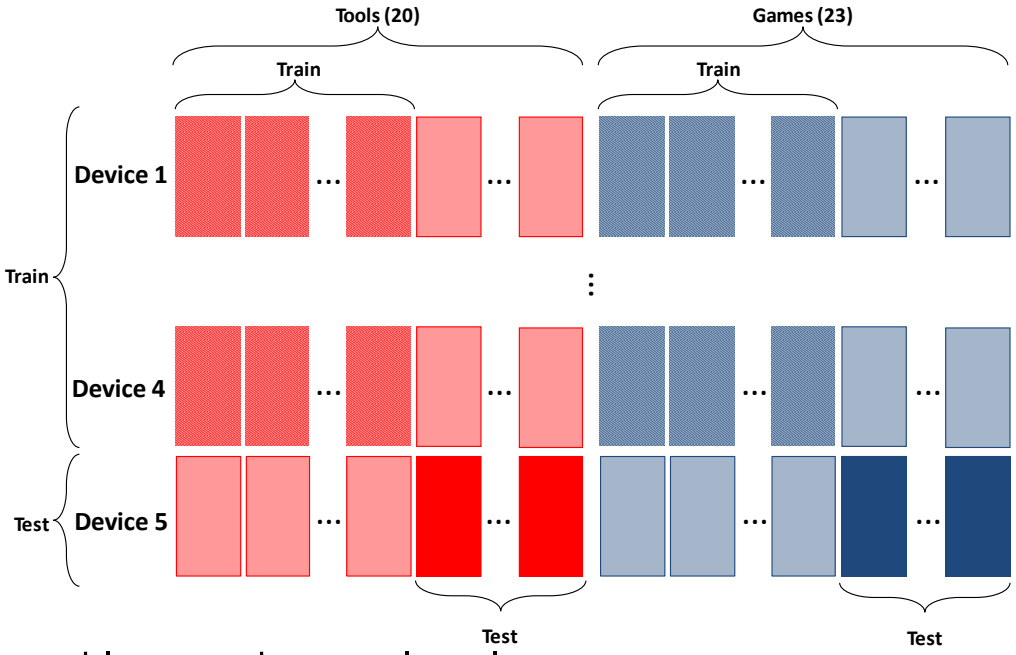
Experiment	Best Configuration	TRP	FPR	AUC	Accuracy
I	DT\J48 InfoGain 20	0.997	0.004	0.998	0.997
II	LR FisherScore 20	0.828	0.199	0.888	0.818

- The detection can be effective even when using a small number of features (20 features) and simple detection algorithms
- The five devices exhibit similar performance indicating that we can learn on a set of devices and detect effectively even on other devices
- High similarity in the features that were selected on each of the experiments
- These observations strengthen the viability of the proposed method for malware detection on mobile devices

Future work

- Train on a set of devices and can detect on other devices





- Evaluation using emulated malware, and later using real malware

Future work

- Remove features with high variance between different devices
- Additional feature (e.g., system calls)
- Add temporal perspective
  - augmenting the collected features with a time stamp (change of the battery level in the last 10min rather than the level of the battery at a certain point in time
  - logging sequences of events (a Boolean feature that is true if there was an access to an SD-card after an installation of application)
- Ensemble with additional detectors (rule-based, knowledge-based)
- Alert about a detected anomaly when it persists
- Focus on monitoring and detection of processes rather than monitoring the whole system

Resources consumption

- Measured 30 features and applied 8 different classifiers that were launched sequentially (i.e., one-by-one)
- The result shows an approx 10% performance degradation

- Feature Extractors are in the critical execution loop and must be optimized aggressively
  - implementing resource exhaustive feature extractors as native code
  - sending only the required features to each processor to reduce the Binder communication overhead

Thank you