DBank: Predictive Behavioral Analysis of Recent Android Banking Trojans

Chongyang Bai, Qian Han, Ghita Mezzour, Fabio Pierazzi, and V.S. Subrahmanian

Online Supplementary Material

Appendix A
Baseline Android Feature Sets

We hereby provide details of the lightweight static and dynamic features that we compare against and that have been used in past research for scalable malware detection [9], [4], [2], [8]. In particular, we consider three types of *static* features (extracted by analyzing an application metadata and code, without executing the application) and one type of *dynamic* features (obtained by running the application).

A.1 Manifest

The *Manifest* feature set consists of matadata information associated with Android applications. These features are extracted from the Android `Manifest`, that is a file declaring the name of application, the requested permissions, and some auxiliary information about code declarations (e.g., name and type of Java classes implemented). In particular the Manifest feature set consists of: filesize, signature-hash of the application (to estimate the authorship), and number of Android components (Activities, Receivers, Intents, Content Providers, Services). In addition, the set contains information about requested Android permissions as binary vectors (where a permission is 1 if requested by the application, and 0 if not). The manifest features reliably replicate the features described in [4], which are a direct extension of other popular works (e.g., [2], [10]).

A.2 API Class and API Package

In the Android system, API libraries to interact with the operating system and the hardware sensors (e.g., read and write operations) are organized into programming packages, such as `android.os` and `android.telephony`. Each package contains one or more classes, such as `android.telephony.CellLocation`. Past literature has often considered API classes and packages as feature sets (e.g., [8], [2], [10]).

- *C. Bai, Q.Han and V.S. Subrahmanian are with the Department of Computer Science and the Institute for Security, Technology, and Society, Dartmouth College, Hanover, NH 03755, USA. G. Mezzour is with the Dept. of Computer Science and Logistic and the TICLab, Universite Internationale de Rabat, Sala El Jadida, Morocco. F. Pierazzi is with King's College London and Royal Holloway, University of London, UK.*
- **Corresponding author:** *Professor V.S. Subrahmanian.*

In particular, here for each application we consider the total number of times (i.e., frequency) in which a method or class of a certain package as been called (as *API Package* feature set) and the number of times in which a method from a particular class has been called (as *API Class* feature set). As in prior work [9], [4], [8], we do not consider method call frequencies because it is too low-level, it increases too much the size of the feature space, and we aim to avoid overfitting of the training data characteristics.

An important observation is that we do not consider advanced API call graphs such as control-flow graphs in FlowDroid [3] or dependency graphs like in DroidSIFT [12], because they are very computationally intensive to extract and hence not scalable. For example, FlowDroid can take up to an hour to extract features from an individual application. Instead, we focus on API class frequency that have been successfully adopted as a basis for building features, and are easy to obtain in a scalable way.

A.3 Dynamic

The *Dynamic* features are extracted by executing the applications on Koodous [1], an online service for Android malware sandboxing and analysis. In particular, Koodous executes the Android application for 60 seconds, and record the major activities by monitoring the app behavior with `cuckoo` and `Droidbox` sandboxes. As in prior work [4], [9], we consider 1-gram, 2-gram and 3-gram of simplified system call activities associated with the following events: Read/Write operations, System operations (e.g., class loading, start of background service), Network operations, and SMS activity. For each app, the n-gram frequency counts are considered as part of the feature vector. We replicate the dynamic features described in more detail in [4], which are inspired by [9].

An important observation is that we do not consider dynamic coverage or advanced application stimulation because it is an open challenge [11] outside the scope of this paper. Instead, we focus on lightweight dynamic analysis that can easily be performed at scale.

Appendix B
Listing and Description of Most Important Features

We report two tables that summarize the most important features in separating ABTs from Goodware (Table 1) and

in separating ABTs from other malware (Table 2). We then report the description of the 10 most important features characterizing `FakeToken` in Table 3, in terms of features distinguishing FakeToken from goodware and from other malware. NOTE: **Descriptions with gray background are reported *verbatim* from the official Android documentation [5], [6], [7]. We report them here just as a quick reference for the reader.**

APPENDIX C
RESULTS WITH ISOMORPHIC SAMPLES

Table 4 and Table 5 show the prediction AUC (%) and FPR (%) with isomorphic samples. We report only individual features because it already shows that the performance are highly inflated (unlike the ones we reported in Section 4, which did not include isomorphic samples).

APPENDIX D
FEATURE HISTOGRAMS

We report the full histograms reporting the key features of ABTs vs. other categories. In particular, these histograms refer to the features with highest *importance scores* accorting to DT-based classifiers, as they are the best performing ones in our dataset. Figure 1 reports the top-25 feature histograms of ABT vs goodware. Figure 2 reports the top-25 feature histograms of ABT vs other malware. Figure 3 reports top-10 features of `FakeToken` vs goodware and the top-10 features of `FakeToken` vs other-malware. Similarly, Figure 4 for `Svpeng`, Figure 5 for `Asacub`, Figure 6 for `BankBot`, and Figure 7 for `Marcher`.

APPENDIX E
ADVERSARY FEATURE DISTANCES

We report additional plots for the feature distances in the robustness experiment in Section 5 of the main text. In particular, Figure 8 reports the results with respect to the Manhattan distance, Figure 9 reports the results with respect to the Cosine distance, and Figure 10 reports the results with respect to the Chebyshev distance.

REFERENCES

[1] Koodous. https://koodous.com/. Accessed: November 2018.

[2] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In *NDSS*, 2014.

[3] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel. FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm SIGPLAN Notices*, 2014.

[4] T. Chakraborty, F. Pierazzi, and V. S. Subrahmanian. Ec2: Ensemble clustering and classification for predicting android malware families. *IEEE Transactions on Dependable and Secure Computing*, 2017.

[5] A. P. Documentation. Api package list. https://developer.android.com/reference/android/Manifest.permission, Visited in Mar. 2019.

[6] A. P. Documentation. Api package list. https://developer.android.com/reference, Visited in Mar. 2019.

[7] A. P. Documentation. Api package list. https://developer.android.com/reference/packages, Visited in Mar. 2019.

[8] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini. MaMaDroid: Detecting android malware by building markov chains of behavioral models. *NDSS*, 2017.

[9] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov. Learning and classification of malware behavior. In *DIMVA*. Springer, 2008.

[10] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro. Droidsieve: Fast and accurate classification of obfuscated android malware. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pages 309–320. ACM, 2017.

[11] K. Tam, S. J. Khan, A. Fattori, and L. Cavallaro. Copperdroid: Automatic reconstruction of android malware behaviors. In *NDSS*, 2015.

[12] M. Zhang, Y. Duan, H. Yin, and Z. Zhao. Semantics-aware android malware classification using weighted contextual api dependency graphs. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1105–1116. ACM, 2014.

TABLE 1: Description of Top-25 features for ABT vs. Goodware. NOTE: **Descriptions with gray background have been reported *verbatim* from the official Android documentation [5], [6], [7].**

Feature	Description
permission: RECEIVE_SMS	Allows an application to receive SMS messages.
android.view	Provides classes that expose basic user interface classes that handle screen layout and interaction with the user.
filesize	The filesize of the application.
android.widget	The widget package contains (mostly visual) UI elements to use on Application screen.
num_std_permissions	The number of standard permissions (manifest permissions in Android developers documentation) used in application.
permission: READ_PHONE_STATE	Allows read only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any PhoneAccounts registered on the device.
android.net	Classes that help with network access, beyond the normal java.net.* APIs.
android.text	Provides classes used to render or track text and text spans on the screen.
javax.net	Provides classes for networking applications. These classes include factories for creating sockets.
java.text	Provides classes and interfaces for handling text, dates, numbers, and messages in a manner independent of natural languages.
num_non_std_permissions	The number of non-standard permissions (not available in Android developers documentation) used in application.
permission: SYSTEM_ALERT_WINDOW	Allows an app to create windows using the type WindowManager.LayoutParams. TYPE_APPLICATION_OVERLAY, shown on top of all other apps.
android.app.admin	Provides device administration features at the system level, allowing you to create security-aware applications that are useful in enterprise settings, in which IT professionals require rich control over employee devices.
permission: (UN)MOUNT_FILESYSTEMS	Allows mounting and unmounting file systems for removable storage.
num_receivers	The number of receivers (Broadcast receivers enable applications to receive intents that are broadcast by the system or by other applications, even when other components of the application are not running.
android.content	Contains classes for accessing and publishing data on a device.
android.view.inputmethod	Framework classes for interaction between views and input methods (such as soft keyboards).
android.media.session	Allows interaction with media controllers, volume keys, media buttons, and transport controls.
android.appwidget	Contains the components necessary to create "app widgets", which users can embed in other applications (such as the home screen) to quickly access application data and services without launching a new activity.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects. Reflection allows programmatic access to information about the fields, methods and constructors of loaded classes, and the use of reflected fields, methods, and constructors to operate on their underlying counterparts, within security restrictions.
android.telephony	Provides APIs for monitoring the basic phone information, such as the network type and connection state, plus utilities for manipulating phone number strings.
write b'appsflyer-data.xml'	Dynamic feature: writing binary file b'appsflyer-data.xml during operation
android.content.res	Contains classes for accessing application resources, such as raw asset files, colors, drawables, media, or other files in the package, plus important device configuration details (orientation, input types, etc.) that affect how the application may behave.
servicestart b'PG_Service'	Dynamic feature: start service PG_Service during operation
read b'com.q3600.app.outsourcing.binding-1.apk'	Dynamic feature: read file com.q3600.app.outsourcing.binding-1.apk during operation

TABLE 2: Description of Top-25 features for ABT vs. Other-Malware. NOTE: **Descriptions with gray background have been reported *verbatim* from the official Android documentation [5], [6], [7].**

Feature	Description
permission: READ_SMS	Allows an application to read SMS messages.
android.app.admin	Provides device administration features at the system level, allowing you to create security-aware applications that are useful in enterprise settings, in which IT professionals require rich control over employee devices.
filesize	The filesize of the application
dalvik.system	Provides utility and system information classes specific to the Dalvik VM
permission: GET_TASKS	Allows an application to get information about the currently or recently running tasks.
java.io	Provides for system input and output through data streams, serialization and the file system.
android.view	Provides classes that expose basic user interface classes that handle screen layout and interaction with the user.
num_non_std_permissions	The number of non-standard permissions (not available in Android developers documentation [6]) used in application.
android.media	Provides classes that manage various media interfaces in audio and video.
android.app	Contains high-level classes encapsulating the overall Android application model.
num_intents	The number of intents (An Intent is a simple message object that is used to communicate between android components such as activities, content providers, broadcast receivers and services) in application.
android.database	Contains classes to explore data returned through a content provider.
android.view.animation	Provides classes that handle tweened animations.
android.net	Classes that help with network access, beyond the normal java.net.* APIs.
android.graphics	Provides low level graphics tools such as canvases, color filters, points, and rectangles that let you handle drawing to the screen directly.
android.content	Contains classes for accessing and publishing data on a device.
num_services	The number of services (A Service is an application component that can perform long-running operations in the background, and it doesn't provide a user interface) in application
android.os	Provides basic operating system services, message passing, and inter-process communication on the device.
android.hardware	Provides support for hardware features, such as the camera and other sensors.
java.util.zip	Provides classes for reading and writing the standard ZIP and GZIP file formats.
num_receivers	The number of receivers (Broadcast receivers enable applications to receive intents that are broadcast by the system or by other applications, even when other components of the application are not running.) in application.
permission: CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call.
android.preference	Provides classes that manage application preferences and implement the preferences UI.
android.telephony	Provides APIs for monitoring the basic phone information, such as the network type and connection state, plus utilities for manipulating phone number strings.
java.util	Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).

TABLE 3: Description of top-10 features for `FakeToken` vs. Goodware (left table) and `FakeToken` vs. Other-malware (right table). NOTE: **Descriptions with gray background have been reported *verbatim* from the official Android documentation [5], [6], [7].**

Feature	Description
android.content	Contains classes for accessing and publishing data on a device.
permission: WRITE_SMS	Allows the app to write to SMS messages stored on your phone or SIM card. Malicious apps may delete your messages.
filesize	The filesize of the application.
android.text	Provides classes used to render or track text and text spans on the screen.
num_std_permissions	The number of standard permissions (manifest permissions in Android developers documentation) used in application.
read b'/data/'	Dynamic feature: read file com.q3600.app.outsourcing.b inding-1.apk during operation
permission: GET_TASKS	Allows an application to get information about the currently or recently running tasks.
dexclass b'com.example.testcallchange-1.apk'	Dynamic feature: read dex file (Compiled Android application code file) com.example.testcallchange-1.apk during operation.
permission: READ_PHONE_STATE	Allows read only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any PhoneAccounts registered on the device.
read b'com.jeujlu.gnvldarosz-1.apk'	Dynamic feature: read file com.jeujlu.gnvldarosz-1.apk during operation.

Feature	Description
num_intents	The number of intents (An Intent is a simple message object that is used to communicate between android components such as activities, content providers, broadcast receivers and services) in application.
java.io	Provides for system input and output through data streams, serialization and the file system.
num_activities	The number of activities (The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model.) in application.
java.util	Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).
java.lang	Provides classes that are fundamental to the design of the Java programming language.
num_std_permissions	The number of standard permissions (manifest permissions in Android developers documentation) used in application.
android.app.admin	Provides device administration features at the system level, allowing you to create security-aware applications that are useful in enterprise settings, in which IT professionals require rich control over employee devices.
android.content	Contains classes for accessing and publishing data on a device.
android.view	Provides classes that expose basic user interface classes that handle screen layout and interaction with the user.
android.os	Provides basic operating system services, message passing, and inter-process communication on the device.

TABLE 4: AUC (%) and FPR (%) on ABT vs. Goodware with isomorphic samples. Cells with gray background highlight the best AUC in the group. We report results for the following features: TSG, Manifest (M), Dynamic (D), API package (AP), and API class (AC).

		KNN		LR		DT		NB		RF		GBDT		MLP		SVM	
	Features	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR
INDIVIDUAL	M (Manifest)	96.4	9.0	99.4	1.7	97.8	3.2	96.8	4.4	99.8	0.5	98.7	2.7	50.0	40.0	44.0	60.5
	AP (API Package)	98.6	6.1	98.6	6.3	98.1	2.9	95.8	10.5	**99.9**	0.9	99.0	2.7	94.2	7.1	88.3	15.2
	AC (API Class)	98.5	6.1	99.1	2.7	98.5	2.1	94.5	9.6	**99.9**	0.9	99.1	1.9	94.7	11.8	96.5	6.5
	D (Dynamic)	83.7	36.9	96.0	0.9	94.6	8.3	94.9	4.7	97.1	4.6	96.8	10.4	96.5	14.3	96.1	14.7
	TSG (Ours)	98.7	6.4	98.9	3.5	98.1	2.8	93.5	10.0	**99.9**	0.7	99.1	2.6	64.8	78.0	76.7	29.3

TABLE 5: AUC (%) and FPR (%) on ABT vs. Other-malware with isomorphic samples. Cells with gray background highlight the best AUC in the group. We report results for the following features: TSG, Manifest (M), Dynamic (D), API package (AP), and API class (AC).

		KNN		LR		DT		NB		RF		GBDT		MLP		SVM	
	Features	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR	AUC	FPR
INDIVIDUAL	M (Manifest)	87.6	26.1	96.0	13.7	95.4	6.0	88.6	14.2	99.0	4.6	98.5	6.8	50.0	50.0	49.0	57.7
	AP (API Package)	97.9	7.8	93.8	11.9	97.0	7.1	85.9	20.2	**99.3**	5.9	98.6	7.7	92.9	24.5	65.1	18.0
	AC (API Class)	97.7	8.7	97.0	12.8	96.9	10.4	51.6	81.6	**99.3**	5.3	98.6	6.7	89.7	40.9	98.1	6.4
	D (Dynamic)	88.4	5.8	95.1	24.8	92.0	19.8	93.7	5.3	96.6	22.0	95.5	22.6	95.4	20.5	95.3	24.1
	TSG (Ours)	97.6	8.6	97.0	8.7	96.6	7.6	80.0	25.4	**99.3**	6.1	98.5	7.5	73.7	65.9	74.9	20.4

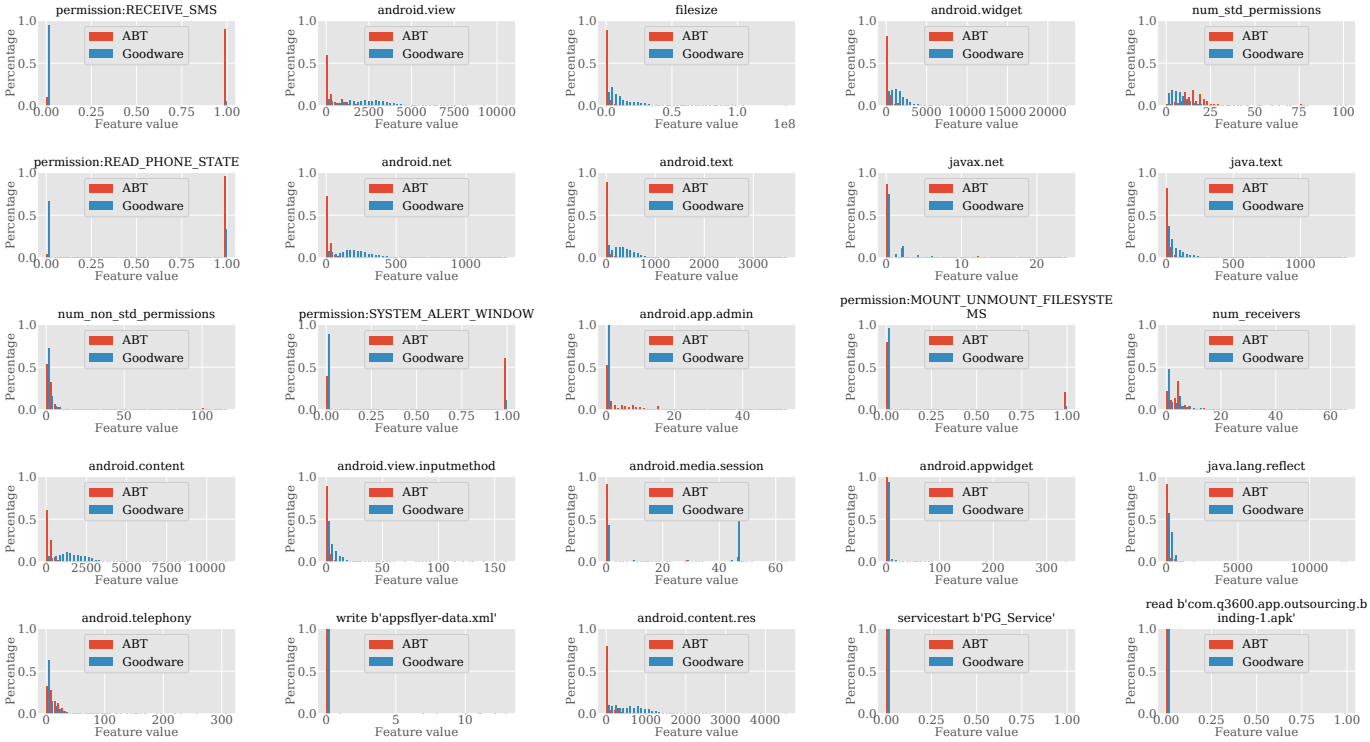


Fig. 1: Top-25 features of ABT vs. Goodware.

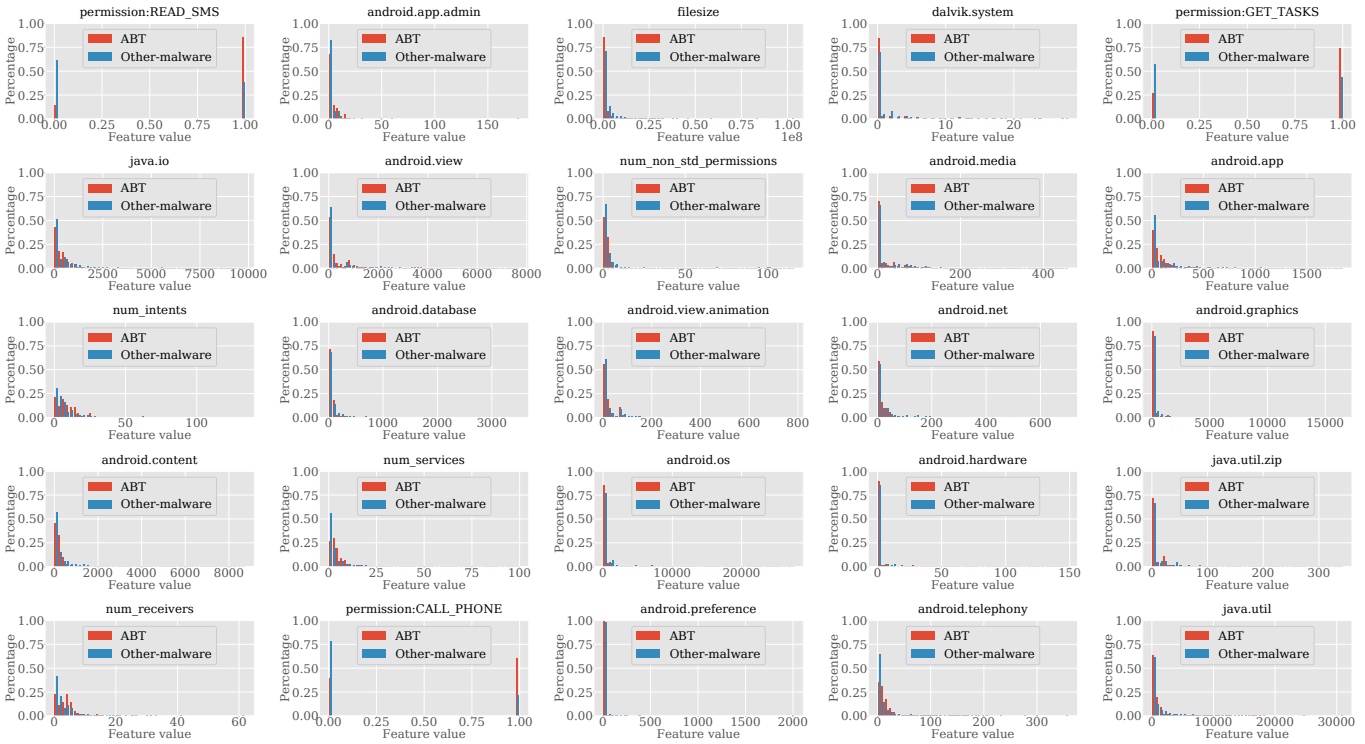
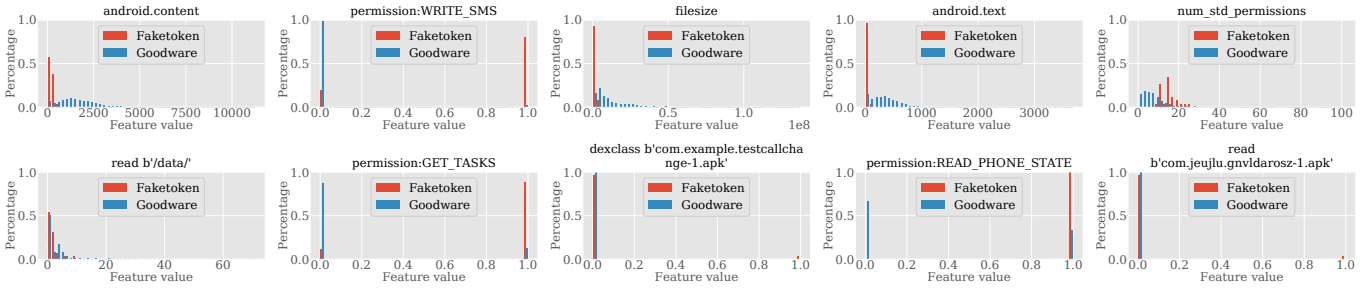
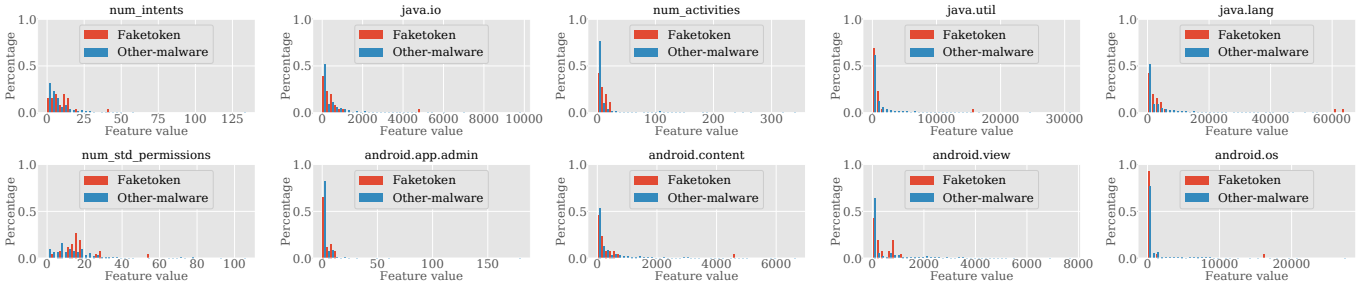


Fig. 2: Top-25 features of ABT vs. Other-Malware.

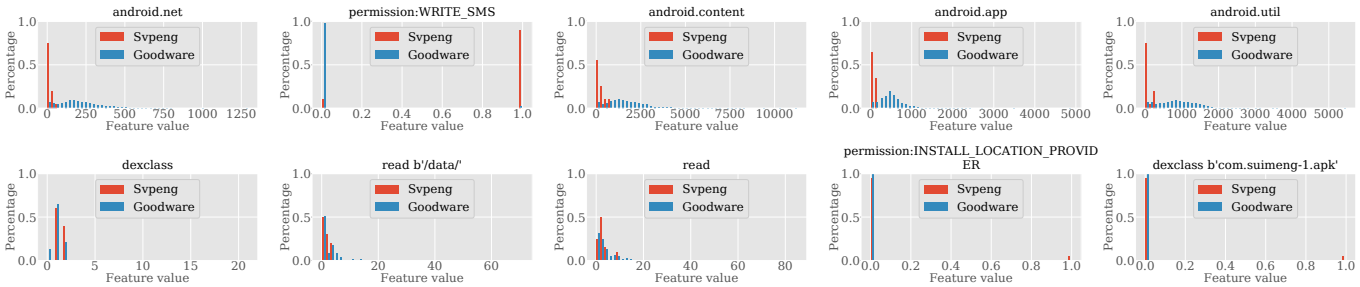


(a) `FakeToken` vs. Goodware

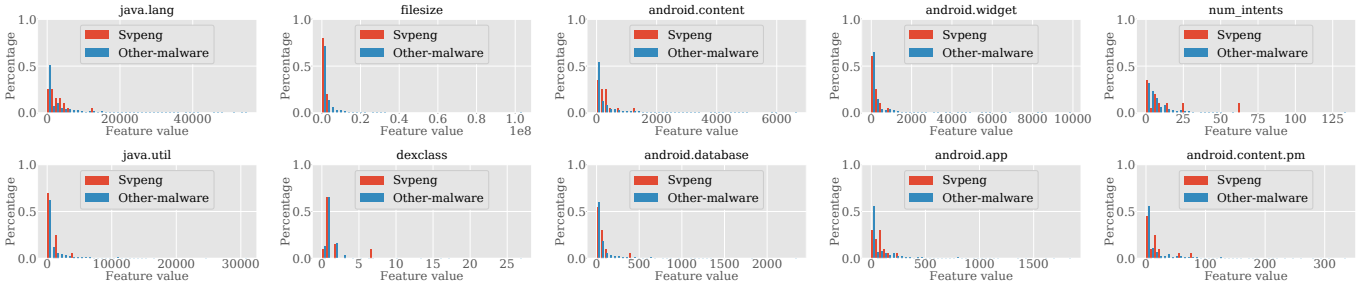


(b) `FakeToken` vs. Other-malware

Fig. 3: Top-10 features of `FakeToken`.

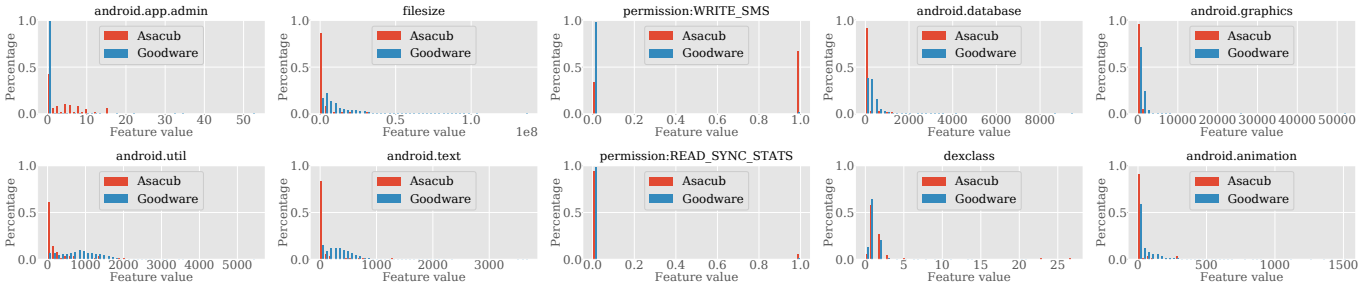


(a) `Svpeng` vs. Goodware

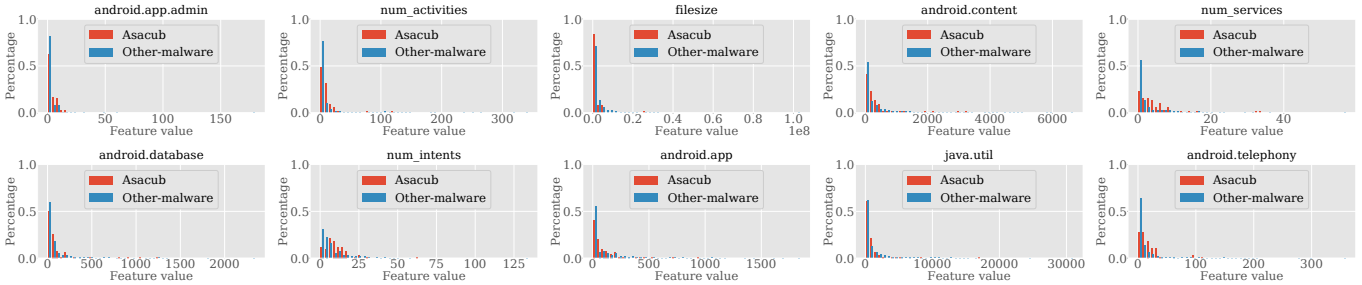


(b) `Svpeng` vs. Other-malware

Fig. 4: Top-10 features of `Svpeng`.

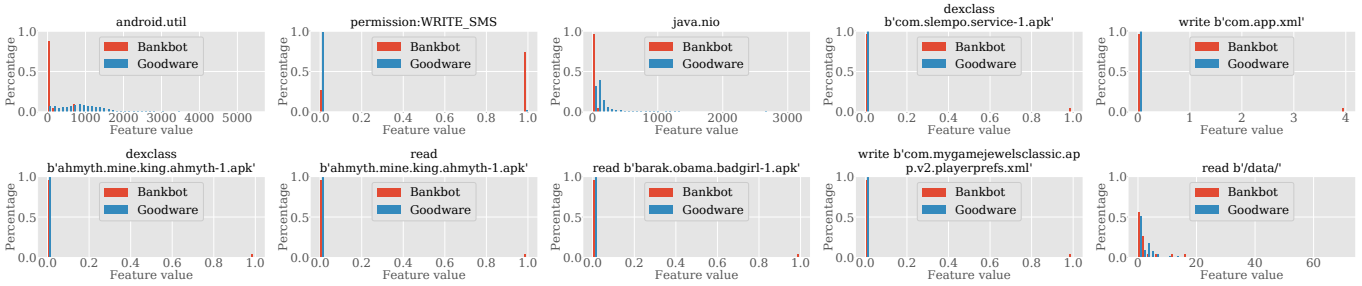


(a) `Asacub` vs. Goodware

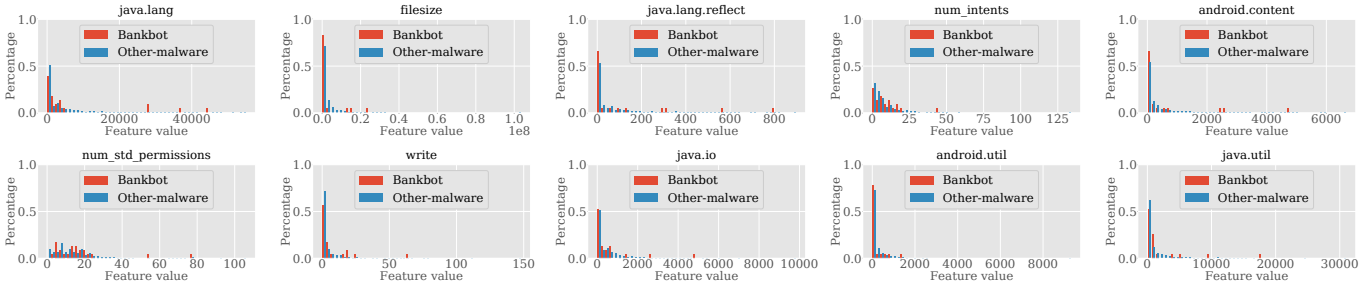


(b) `Asacub` vs. Other-malware

Fig. 5: Top-10 features of `Asacub`.

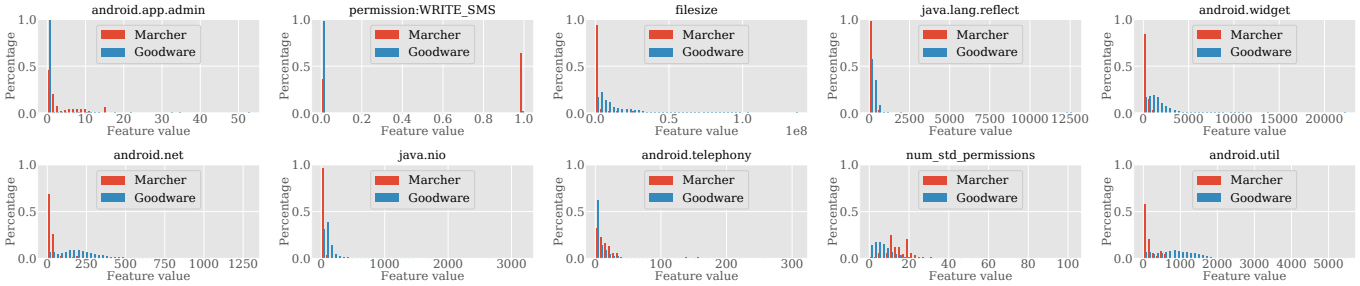


(a) `BankBot` vs. Goodware

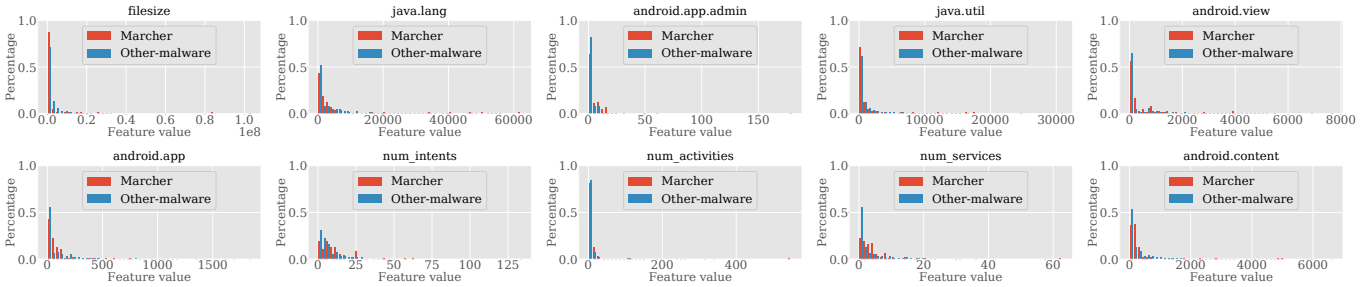


(b) `BankBot` vs. Other-malware

Fig. 6: Top-10 features of `BankBot`.

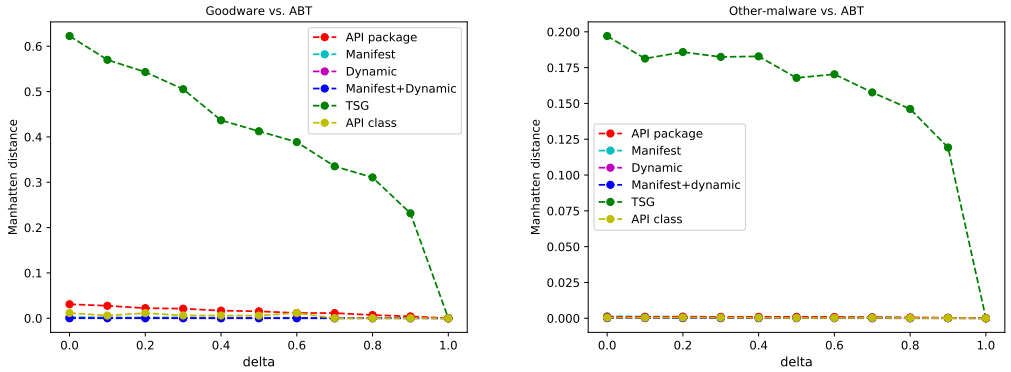


(a) `Marcher` vs. Goodware



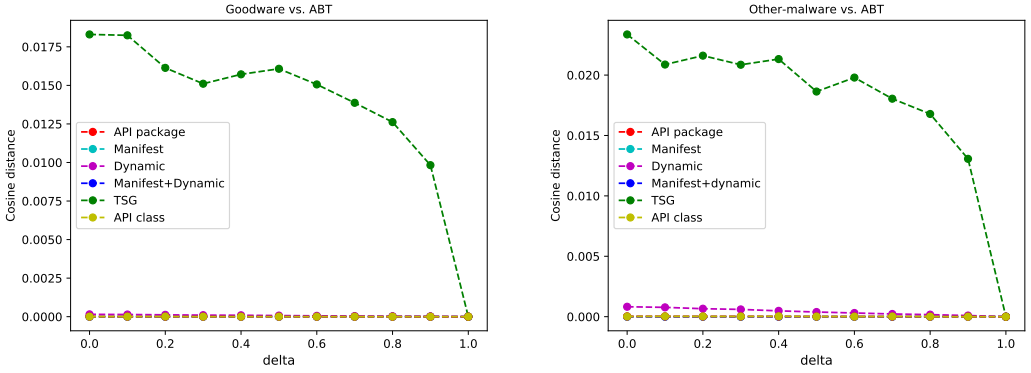
(b) `Marcher` vs. Other-malware

Fig. 7: Top-10 features of `Marcher`.



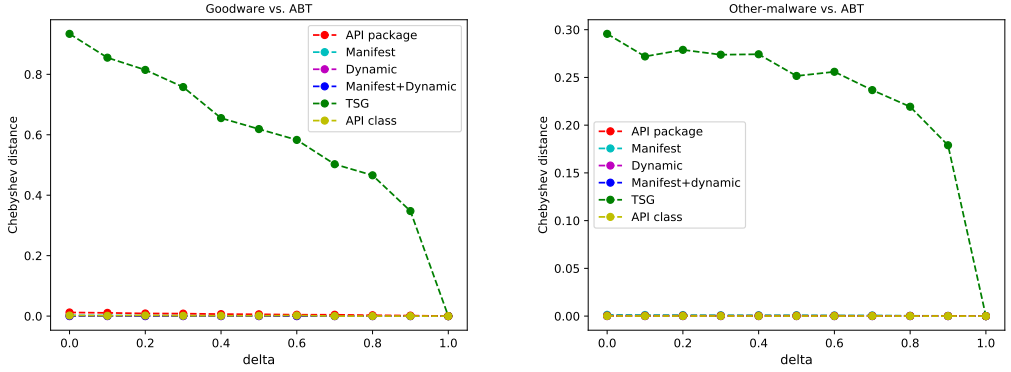
(a) Goodware vs. ABT (b) Other-malware vs. ABT

Fig. 8: Manhattan distance among attacker's and defender's feature sets centroids. High values imply higher robustness.



(a) Goodware vs. ABT (b) Other-malware vs. ABT

Fig. 9: Cosine distance among attacker's and defender's feature sets centroids. High values imply higher robustness.



(a) Goodware vs. ABT (b) Other-malware vs. ABT

Fig. 10: Chebyshev distance among attacker's and defender's feature sets centroids. High values imply higher robustness.