A Comprehensive Study of Learning-based Android Malware Detectors under Challenging Environments

Supplementary Material

Cuiying Gao
Huazhong University of Science and Technology, China
gaocy@hust.edu.cn

Gaozhun Huang
Huazhong University of Science and Technology, China
gaozhun@hust.edu.cn

Heng Li
Huazhong University of Science and Technology, China
liheng@hust.edu.cn

Bang Wu
Huazhong University of Science and Technology, China
bangw@hust.edu.cn

Yueming Wu
Nanyang Technological University, Singapore
wuyueming21@gmail.com

Wei Yuan*
Huazhong University of Science and Technology, China
yuanwei@mail.hust.edu.cn

In this document, we supply some discussions and experimental results that are not included in our paper due to space limitations.

1 EVALUATING DATA-MD

In general, the obfuscated samples generated by Obfuscapk[1] should not be contained in the training set when we evaluate various detectors in the scenario of code obfuscation. So we should check if our training set contains obfuscated samples that will cause bias to the experimental results. Unfortunately, to our knowledge, there are no obfuscation detectors that we can use for this purpose. To overcome this challenge, we downloaded a batch of clean samples without any obfuscation from the F-Droid app market, where the corresponding source code is all publicly available. We then manually checked whether the downloaded APKs were obfuscated or not. In this way, we collected 936 samples without any obfuscation, and built a dataset with them, called F-Droid. We then use contrast experiments to validate the Data-MD (i.e., the dataset used in our manuscript).

More specifically, we evaluate the effects of various obfuscation strategies on function call graphs over the F-Droid dataset, and then compare the experimental results with the results obtained on the dataset Data-MD. The comparison results are given in Table 1. It can be seen that the results shown in these two tables are similar and have almost identical statistical characteristics. This indicates that similar to F-Droid, Data-MD is appropriate to evaluate the detectors' resistance to code obfuscation and will not cause experimental bias. Otherwise, the statistical characteristics presented in the two tables should be different.

2 SUPPLEMENTARY MATERIALS FOR RQ2

We introduced a new obfuscation tool, *Allatori*, into our experiments. *Allatori* is a Java and Android obfuscator. It is a commercial software tool designed to protect applications from reverse engineering and unauthorized modifications. We provide the experimental results in Table 2. In this table, each row corresponds to a detector, and each column represents an obfuscation technique supported by *Allatori*, which is briefly explained below.

*Corresponding authors

[1]The obfuscation tool we use to generate obfuscated samples.

Table 1: Impact of code obfuscation on function call graphs of Data-MD and F-Droid.

Data-MD				
Obf.Tec	Node_diff	Node_p	Edge_diff	Edge_p
RBD	0.000%	0.317	0.000%	0.317
CR	0.155%	*<0.05*	0.083%	*<0.05*
MR	0.131%	*<0.05*	0.000%	0.317
CSE	0.140%	*<0.05*	0.676%	*<0.05*
JUNK	0.000%	0.317	0.000%	0.317
ROR	0.000%	0.317	0.000%	0.317
CID	27.728%	*<0.05*	19.473%	*<0.05*
REF	0.044%	*<0.05*	-0.174%	*<0.05*
F-Droid Dataset				
Obf.Tec	Node_diff	Node_p	Edge_diff	Edge_p
RBD	0.000%	0.317	0.000%	0.317
CR	0.090%	*<0.05*	0.000%	*<0.05*
MR	0.104%	*<0.05*	0.059%	*<0.05*
CSE	0.345%	*<0.05*	1.205%	*<0.05*
JUNK	0.000%	0.317	0.000%	0.317
ROR	0.000%	0.317	0.000%	0.317
CID	38.287%	*<0.05*	30.017%	*<0.05*
REF	0.127%	*<0.05*	-0.145%	*<0.05*

ClassRename (CR): rename the class name to a meaningless string. **MethodRename (MR)**: rename the method name to a meaningless string. **ConstStringEncryption (CSE)**: encrypt the constant strings in code. **Reorder (ROR)**: shuffle fields and methods. **Control Flow (CF)**: modify the standard code structures, such as loops, conditional statements, and branching instructions, and adjust the command sequence whenever possible to prevent identifying the equivalent code after decompilation.

As shown in Table 2, we show the decrease ratio of F1 under code obfuscation, i.e., $(F1_{ORI} - F1_{RBD})/F1_{ORI} \times 100\%$. $F1_{ORI}$ and $F1_{RBD}$ denote the F1 score measured on the original and the obfuscation test set, respectively.

(a) The Image-based detectors suffer significant performance degradation over all obfuscation test sets, with an average F1 decrease ratio of 38.78% and 19.47% on ImgDroid and MDMC, respectively.

(b) The Graph-based detectors have different sensitivities to different obfuscation techniques. The *ROR* obfuscation technique

Table 2: The decrease ratio of F1 under obfuscation conducted by *Allatori*.

Tools	CR	MR	CSE	ROR	CF
Drebin	1.40%	0.74%	0.46%	0.26%	0.69%
RevealDroid	0.92%	0.56%	1.39%	0.57%	0.87%
MudFlow	1.09%	0.25%	0.38%	0.23%	0.66%
ALDroid	1.08%	0.23%	1.19%	0.04%	0.01%
Bai's	1.49%	0.73%	1.99%	1.69%	0.28%
ImgDroid	41.31%	35.01%	37.51%	38.05%	42.00%
MDMC	28.71%	23.03%	13.36%	17.56%	14.67%
MaMa-fml	37.42%	0.33%	15.63%	0.92%	0.01%
MaMa-pkg	19.26%	8.11%	15.00%	0.03%	2.19%
Malscan	10.93%	4.10%	2.00%	0.08%	0.66%
APIGraph_p	22.80%	11.70%	9.35%	0.07%	5.16%
APIGraph_f	28.87%	0.66%	15.71%	0.09%	5.45%
EFCG	13.54%	0.52%	0.06%	0.39%	0.23%

almost has no influence on all the detectors. On the obfuscation test sets of *ROR*, the F1 of detectors is essentially unchanged, while it suffers different degrees of degradation on the obfuscation test sets of *CR*, *CSE*, *MR*, and *CF*. Moreover, different detectors show varying degrees of sensitivity to the same obfuscation technique. For instance, on the *CR* test set, the F1 score is decreased by 37.42%, 28.87%, 19.26%, and 10.93% for MaMa_fml, APIGraph_f, MaMa_pkg, and Malscan, respectively.

(c) The String-based detectors perform more stably on the obfuscation test sets than the others. Even in the worst case, the F1 of Drebin, RevealDroid, MudFlow, ALDrid and Bai's is reduced by only 1.40%, 1.39%, 1.09%, 1.19% and 1.99%, respectively.

By comparing the experimental results of *Allatori* and *Obfuscapk*, we can conclude that the impact of various obfuscation tools on different detector types demonstrates similar patterns. Therefore, our previous conclusion still holds true, i.e., "*The String-based detectors are relatively stable under code obfuscation. The Image-based detectors deteriorate more severely. The Graph-based detectors perform well under most obfuscation techniques, but their performance significantly decreases when facing some obfuscation techniques.*"

3 SUPPLEMENTARY MATERIALS FOR RQ3

Here we provide more details about the experiments for RQ3. We first use Tables 3 and 4 to show the top 100 commonly-used sensitive API calls and the top 40 commonly-used permissions and actions of 2016 year APKs.

We then focus on the left subfigure in Figure 5 of our manuscript. It can be seen that the use frequency of most APIs exhibits a decreasing trend as the years increase. It should be pointed out this decreasing trend is not resulted by our APK sampling bias, i.e., selecting small-size APKs in some years (e.g., 2020) for API counting. To verify it, we use Figure 1 to show the size distributions of APKs and classes.dex files selected in different years. Clearly, the size distributions across different years are similar, thus validating the inexistence of APK sampling bias.

Table 3: The list of top 100 sensitive API calls.

Ljava/lang/reflect/Method;->invoke	Landroid/support/v4/widget/SlidingPaneLayout$SlidingPanelLayoutImplJB;->invalidateChildRegion
Ljava/lang/Class;->forName	Landroid/support/v4/content/FileProvider;->parsePathStrategy
Ljava/io/File;->delete	Landroid/support/v4/content/WakefulBroadcastReceiver;->startWakefulService
Ljava/lang/Class;->getDeclaredMethod	Landroid/support/v4/text/ICUCompatIcs;-><clinit>
Ljava/io/FileOutputStream;->write	Landroid/support/v4/text/ICUCompatIcs;->addLikelySubtags
Landroid/widget/TextView;->setText	Landroid/support/v4/text/ICUCompatIcs;->getScript
Landroid/view/ViewGroup;->addView	Landroid/support/v4/media/TransportMediatorJellybeanMR2;->windowAttached
Landroid/content/Context;->startActivity	Landroid/support/v4/os/EnvironmentCompat;->getStorageState
Landroid/content/Context;->registerReceiver	Landroid/support/v4/content/ContextCompat;->getObbDirs
Landroid/os/Environment;->getExternalStorageDirectory	Landroid/support/v4/app/NotificationCompatJellybean;->getExtras
Ljava/io/File;->listFiles	Landroid/support/v4/content/ContextCompat;->getExternalCacheDirs
Landroid/content/Context;->startService	Landroid/support/v4/content/ContextCompat;->getExternalFilesDirs
Ljava/lang/Class;->getDeclaredField	Landroid/support/v4/app/NotificationManagerCompat$SideChannelManager;->ensureServiceBound
Landroid/util/Base64;->decode	Landroid/support/v4/app/NotificationCompatJellybean;->ensureActionReflectionReadyLocked
Landroid/content/Context;->bindService	Landroid/support/v4/view/ViewCompat$BaseViewCompatImpl;->dispatchFinishTemporaryDetach
Ljava/io/File;->mkdir	Landroid/support/v4/view/ViewCompat$BaseViewCompatImpl;->dispatchStartTemporaryDetach
Ljava/io/File;->createNewFile	Landroid/location/LocationManager;->getLastKnownLocation
Landroid/database/sqlite/SQLiteDatabase;->execSQL	Landroid/support/v4/view/ViewCompat$BaseViewCompatImpl;->bindTempDetach
Ljava/io/File;->renameTo	Landroid/app/DownloadManager;->enqueue
Landroid/support/v4/view/ViewPager;->addView	Landroid/support/v4/view/ViewCompat$ICSViewCompatImpl;->hasAccessibilityDelegate
Landroid/database/sqlite/SQLiteDatabase;->delete	Landroid/support/v4/provider/RawDocumentFile;->createDirectory
Ljava/net/HttpURLConnection;->connect	Landroid/support/v4/provider/RawDocumentFile;->createFile
Landroid/database/sqlite/SQLiteDatabase;->rawQuery	Landroid/support/v4/provider/RawDocumentFile;->delete
Landroid/database/sqlite/SQLiteDatabase;->query	Landroid/support/v4/provider/RawDocumentFile;->listFiles
Landroid/os/PowerManager;->newWakeLock	Landroid/support/v4/provider/RawDocumentFile;->renameTo
Landroid/database/sqlite/SQLiteDatabase;->insert	Landroid/support/v4/provider/RawDocumentFile;->deleteContents
Landroid/support/v4/view/ViewPager;->setChildrenDrawingOrderEnabledCompat	Landroid/app/ActivityManager;->getRunningTasks
Landroid/content/Context;->sendBroadcast	Landroid/support/v4/view/ViewCompatEclairMr1;->setChildrenDrawingOrderEnabled
Landroid/support/v4/app/FragmentManagerImpl;->moveToState	Landroid/support/v4/view/LayoutInflaterCompatHC;->forceSetFactory2
Landroid/support/v4/app/ListFragment;->setEmptyText	Landroid/support/v7/widget/Toolbar;->setSubtitle
Landroid/support/v4/app/TaskStackBuilder;->startActivities	Landroid/support/v7/widget/Toolbar;->setTitle
Landroid/support/v4/app/ListFragment;->ensureList	Ljava/io/File;->list
Landroid/support/v4/content/FileProvider;->delete	Landroid/location/LocationManager;->isProviderEnabled
Landroid/support/v4/widget/SimpleCursorAdapter;->setViewText	Landroid/support/v4/view/ViewCompatBase;->getMinimumHeight
Landroid/support/v4/view/PagerTitleStrip;->updateText	Landroid/support/v4/view/ViewCompatBase;->getMinimumWidth
Landroid/app/ActivityManager;->getRunningAppProcesses	Lcom/google/android/gms/security/ProviderInstaller;->installIfNeeded
Landroid/support/v4/app/ActionBarDrawerToggleHoneycomb$SetIndicatorInfo;-><init>	Lcom/google/android/gms/analytics/CampaignTrackingReceiver;->onReceive
Landroid/support/v4/app/ActionBarDrawerToggleHoneycomb;->setActionBarDescription	Landroid/support/v4/app/FragmentHostCallback;->onStartActivityFromFragment
Landroid/support/v4/app/ActionBarDrawerToggleHoneycomb;->setActionBarUpIndicator	Landroid/support/v4/text/ICUCompatApi23;-><clinit>
Landroid/support/v4/util/AtomicFile;->delete	Landroid/support/v4/widget/PopupWindowCompatApi21;-><clinit>
Landroid/support/v4/util/AtomicFile;->failWrite	Landroid/support/v4/graphics/drawable/DrawableCompatJellybeanMr1;->setLayoutDirection
Landroid/support/v4/util/AtomicFile;->finishWrite	Landroid/support/v4/graphics/drawable/DrawableCompatJellybeanMr1;->getLayoutDirection
Landroid/support/v4/util/AtomicFile;->openRead	Landroid/support/v4/text/ICUCompatApi23;->maximizeAndGetScript
Landroid/support/v4/util/AtomicFile;->startWrite	Landroid/telephony/TelephonyManager;->getDeviceId
Landroid/support/v4/app/ActivityCompatJB;->startActivity	Landroid/support/v7/widget/SearchView;->launchQuerySearch
Landroid/support/v4/widget/DrawerLayout;->addView	Landroid/support/v7/widget/SearchView;->launchIntent
Ljava/lang/Runtime;->getRuntime	Landroid/support/v7/widget/SearchView;->onVoiceClicked
Landroid/database/sqlite/SQLiteDatabase;->update	Landroid/support/v4/widget/CompoundButtonCompatDonut;->getButtonDrawable
Landroid/support/v4/widget/SlidingPaneLayout$SlidingPanelLayoutImplJB;-><init>	Landroid/support/v4/widget/PopupWindowCompatGingerbread;->setWindowLayoutType
Landroid/app/Application;->onCreate	Landroid/support/v7/widget/SearchView$AutoCompleteTextViewReflector;-><init>

Table 4: The list of top40 permissions and actions

android.intent.action.MAIN	ACCESS_COARSE_LOCATION	BLUETOOTH
INTERNET	RECEIVE_BOOT_COMPLETED	INSTALL_SHORTCUT
ACCESS_NETWORK_STATE	com.google.android.c2dm.intent.RECEIVE	RECORD_AUDIO
WRITE_EXTERNAL_STORAGE	CAMERA	android.intent.action.PACKAGE_REMOVED
ACCESS_WIFI_STATE	GET_TASKS	READ_LOGS
WAKE_LOCK	RECEIVE	CALL_PHONE
READ_PHONE_STATE	SYSTEM_ALERT_WINDOW	android.intent.action.PACKAGE_ADDED
android.intent.action.VIEW	android.intent.action.USER_PRESENT	SET_WALLPAPER
READ_EXTERNAL_STORAGE	CHANGE_WIFI_STATE	MODIFY_AUDIO_SETTINGS
VIBRATE	C2D_MESSAGE	com.google.firebase.INSTANCE_ID_EVENT
READ_SETTINGS	android.net.conn.CONNECTIVITY_CHANGE	READ_CONTACTS
android.intent.action.BOOT_COMPLETED	MOUNT_UNMOUNT_FILESYSTEMS	BILLING
WRITE_SETTINGS	GET_ACCOUNTS	
ACCESS_FINE_LOCATION	CHANGE_NETWORK_STATE	

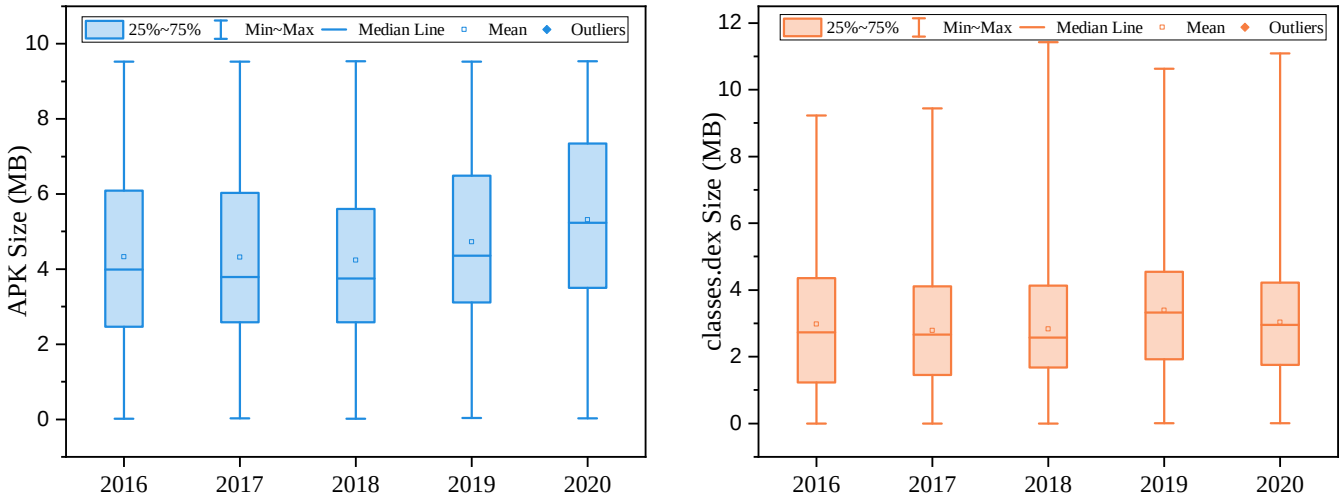

Figure 1: The distribution of APK and classes.dex sizes across different years in Data-MD.