Android Malware Detection Using Kullback-Leibler Divergence

Vanessa N. Cooper, Hisham M. Haddad, Hossain Shahriar
Department of Computer Science, Kennesaw State University, Kennesaw, Georgia, USA

KEYWORDS

Android malware detection
Kullback-Leibler Divergence
Back-off smoothing

ABSTRACT

Many recent reports suggest that malware applications cause high billing to victims by sending and receiving hidden SMS messages. Given that, there is a need to develop necessary technique to identify malicious SMS operations as well as differentiate between good and bad SMS operations within applications. In this paper, we apply Kullback-Leibler Divergence (KLD) as a distance metric to identify the difference between good and bad SMS operations. We develop a set of elements that represent sending or receiving of SMS messages, both legitimately and maliciously. Then, we compare the divergence of the trained set of elements. Our evaluation shows that the divergence between good and bad applications remains significantly high, whereas between two applications performing the same SMS operations remain low. We evaluate the proposed KLD-based concept for identifying a set of malware applications. The initial results show that our approach can identify all known malware and has less false positive warning.

1 Introduction

Android has become the leading smartphone OS in the world with staggering sales figure of 60 million phones in the third quarter of 2011, 50% market share [AARON, D. B. 2011]. A recent study shows that more than 50% of Android mobile have unpatched vulnerabilities, opening them up to malicious applications (malware) and attacks. A compromised smartphone can inflict severe damage to both users and the cellular service provider. Malware applications can make the phone partially or fully unusable, cause unwanted billing, steal private information, or infect every name in a user's phonebook [REZA, H. *et al*. 2012]. Recently, a malware affected more than 100,000 Android devices in China (known as *MMarketPay*). This malware is a hidden application that appeared to be legitimate and is designed to purchase applications and contents without the consent of the device users (victims). As a result, victims saw a staggering amount of bills [BALDWIN, C. 2012]. The incident prompted Google, the developer of the Android OS, to introduce stricter rules for applications on Android such as naming of applications and banning applications that disclose personal information without user permission. An Android Short Message Service (SMS) malware firm was fined £50,000 by the UK premium phone services regulator *PhonepayPlus* [PHONEPAYPLUS, 2013]. An SMS is a text messaging service available on most mobile devices and is a very popular choice of communication. Among most known malicious activities performed by malware, SMS message sending and receiving operations are the starting point to perform further malicious activities. Given that, it is important to ensure that functionalities related to sending and receiving SMS messages are checked for their potential malicious behaviors even before applications are installed to reduce much of the unwanted consequences.

This work is intended to identify malicious SMS operations (both sending and receiving) in Android applications. More specifically, we address the following research question: *Given that we have an access to both legitimate and malicious applications performing a specific* *functionality, how do we distinguish the good behavior from the bad behavior?* To answer this question, in this paper, we propose Kullback-Leibler Divergence (KLD) as a choice of measure to differentiate SMS operations between legitimate and malicious applications.

Our work is motivated by a number of works that apply the concept of KLD as a measure to solve a number of problems from various domains. Our assumption is that KLD between a good and malware application for a specific functionality should be significantly higher than that of another good application performing the same functionality. However, to compute the KLD between two applications, the following two challenges arise: (i) identifying the specific set of elements with the occurrence probabilities, where the elements are relevant to performing SMS operations in good and malicious ways, and (ii) addressing the computational issue inherent for KLD, where zero probability for any element in a set would result in an infinite KLD distance. This paper addresses the above issues, and performs an initial evaluation on the effectiveness of KLD as a measure to differentiate between malware and good applications.

This paper is organized as follows: Section 2 discusses the motivation of our approach. Section 3 presents the theory and discussion of related work; it includes an example application of KLD-based measure to detect malware. Section 4 discusses the results; finally, Section 5 concludes and discusses the future work.

2 Related Work

Bigi [BIGI, B. 2003] applied KLD to identify authorship of documents. The approach first builds a model of each document author by aggregating documents generated by that author. First, it develops a set of candidate models. Then, for a given document of unknown author, the approach finds the smallest KLD between a known model and the document. The model that is closest to the document is selected as the author. Similar to this work, we apply constant back-off smoothing technique to address the missing elements (or tokens derived from Java code of the malware). Specifically, we compare the KLD between the code level features captured by population elements of an application and the expected population obtained from benign applications. The deviation, if exceeds a given threshold value, provides an indication of the presence of malware operation in an application.

Tapiador [TAPIADOR, J. *et al*. 2010] detected masquerade attacks based on an anomaly-based technique that compares a given request with known normal request using KLD measure. In a masquerade attack, an attacker steals credentials of legitimate users and performs further malicious actions using the credentials. The KLD enables the detection of padding in command sequences independent of the length and position in a block of request. In contrast, we apply KLD to detect malware activities based on code level features.

Li [LI. H. *et al*. 2012] applied differential KLD to detect anomalous data value in wireless sensor networks. The network is divided into clusters. In each cluster, the sensors remain physically close to each other and sense similar values. The outlier values are detected using KLD. Sarkar [SARKAR, S. *et al*. 2007] applied information theoretic measure including KLD to measure the quality of modularization in non-object oriented software systems. Fukui [FUKUI, K. *et al*. 2010] measured the similarity of events based on KLD and applied it in the domain of fuel-cell study.

3 KLD-Based Approach

Instead of using heuristic-based approaches, such as Euclidean Distance or other measures, to compare an application with known sample applications, this work uses a formal method based on probabilistic models. It is assumed that a hidden probabilistic model is generated for each benign application (*M_benign*) and malicious application (*M_malicious*). The hypothesis is that the divergence between the models *M_benign* and *M_malicious* should be detectable. Then, KLD is used to evaluate the divergence between the *M_benign* and *M_malicious* models.

Since the hidden probabilistic models are unknown, observable features generated from either model are used to approximate the model. For this purpose, features (f_1 to f_{10}) are extracted. It is further assumed that each application is generated by randomly sampling (f_1 to f_{10}) from the hidden model. Since the observed population is very limited, a smoothing technique is needed to avoid zero probability of feature observation. The KLD computes the divergence between two given probability distributions. Let us assume that *P* and *Q* represent two probability distributions, where

$$P = \{p_i, \ldots, p_n\} \quad (1)$$

$$Q = \{q_i, \ldots, q_n\} \quad (2)$$

Then, the KLD is defined as follows [COVER, T. *et al*. 2006]:

$$KLD(P,Q) = \sum_{i}^{n} p_i * \log_2\left(\frac{p_i}{q_i}\right) \quad (3)$$

We start with a hypothesis that the KLD between benign and malicious application for performing a specific operation should be relatively high [COOPER, V. N. 2014]. On the other hand, the KLD among benign applications performing the same operation should be relatively low. This approach uses different features to detect malicious applications. We define feature elements from the source code that relates to the primary purpose of the application's functionality. Using this information, we are able to determine suspicious malware applications. Our prototype implementation analyzes the source code of a suspected malware application in a secure environment without running the malware application on a mobile device.

We consider SMS message sending as a case study for this work. For a given SMS functionality, we identify the source code responsible for invoking it along with source of inputs. The malicious applications typically do not accept inputs from users and mostly supplies static values during the invocation of method calls. On the other hand, the legitimate applications, while performing the same functionality, rely on user-supplied inputs. This makes a difference between the behavior of a malicious and a legitimate application. KLD can be a suitable measure to understand it as an automated process; hence, it can be used to detect malicious applications. To compute the KLD between two population sets (or probability distributions), we need to define a set of elements relevant to the specific SMS operations and obtain a collection of legitimate application samples to build *P* set. Now, given that we have a new application (*Q*), we can then find how divergent is the new application compared to the *P* set with respect to SMS operation to label the new application as malware or good application.

Tables 1 and 2 show the list of 10 elements (f_1-f_{10}) that we consider in building the population of elements and compute their occurrence probabilities from Android applications. Among them, the first five elements are commonly found to be legitimate ways of sending (f_1-f_4) or receiving (f_5) SMS messages (based on extensive survey and reports from related work).

Type	Name	Description
Benign	f_1	SMS message is sent with visual input, through even handler method
	f_2	SmsManager object is created, sendTxtMsg is invoked, variable argument is present
	f_3	Create Intent object, write SMS message, variable argument message, start Activity
	f_4	Start activity with "smsto:" string in Uri.parse method and variable parameter for SMS message
	f_5	Message delivery or receiving status is notified

Table 1: SMS Operational Elements for Building Population Set of Malicious Actions

Type	Name	Description
Malicious	f_6	SMS message is sent without input from visual interfaces, and in presence or absence of event handler method
	f_7	SmsManager object is created, sendTxtMsg is invoked, constant argument present
	f_8	Using intent object, putting SMS body, and constant argument message
	f_9	Start activity with "smsto:" string in Uri.parse method and constant parameter representing SMS message
	f_{10}	Message delivery or receiving status is not notified

Table 2: SMS Operational Elements for Building Population Set of Malicious Actions

For example, f_1 represents sending SMS message by creating a visual Action window where a user can provide message and destination number for sending a message. At the Java source code level, we then look for the following sequence of method call invocation: *setContentView()* that allows for displaying of an Action window on the screen, one or more call of *getText()* to access the current values of input from GUIs passed as SMS sending operation argument, and the presence of the event handler that invokes the text retrieval operation and SMS sending operation. Good applications send SMS messages using variables as part of their arguments of the respective method (*sendTextMessage()* and variable argument) as shown in f_2. An application may rely on creating an Intent object and store SMS messages as part of the method call argument (*putExtra*) followed by launching the Activity (f_3). The *Uri.parse()* method can be invoked as well for sending messages (f_4).

For a given set of legitimate Android applications, we compute the *P* set containing the occurrence of f_1 - f_{10} and the probability distribution. Then, given a new Android application we identify the *Q* set containing the occurrence probability of f_1 - f_{10} and see how divergent the two sets are to understand the closeness. The less divergence we find, the closer the two sets, hence *Q* is identified to be good application with respect to the specific SMS operation. On the other hand, if the distance is very high, then we label *Q* as malware. As one or more elements from *P* and *Q* may not have any occurrence (zero probability), they need to be smoothed.

However, the challenge here is computing the term $p_i * log_2 (p_i/q_i)$. It can be rewritten as subtraction of two terms: $p_i * log_2(p_i) - p_i * log_2(q_i)$. While we compute KLD (*P*, *Q*), if either p_i or q_i is zero (no occurrence of probability is observed from applications), then the term becomes infinite, which results in KLD (*P*, *Q*) to be zero. To address this issue, we propose to apply a well-known smoothing technique known as constant back-off [BIGI, B. 2003]. Here, all zero probability values in both *P* and *Q* are substituted with a very negligible constant value and all the non-zero values are equally subtracted with the same constant amount proportionally so that Equations (1) and (2) are still satisfied. This simple step results in two smoothed probability distributions denoted as *P*' (derived from *P*) and *Q*' (derived from *Q*). So, we essentially compute KLD (*P*', *Q*') to avoid infinity problem instead of KLD (*P*, *Q*).

4 Results

We evaluated our approach as follows: first we gather a set of legitimate Android applications downloaded randomly from the web, where each of the applications contains Java code for performing SMS functionalities. To ensure diversity in the test applications, selected applications rely on different known techniques of sending or receiving SMS messages (*SmsManager*, *putExtra* for *Intent*, *Uri.parse*). We have 17 applications in our data set to construct the *P* set. For the *Q* set, we use one application that we are comparing with the *P* set. Table 3 shows the KLD between *P* and each of the malware (*Q*). We show the results in terms of *P*' and *Q*' (after smoothing the sets). The value ranges between 12.47 and 17.25, which provides a basis of threshold values for consideration to detect new malware samples for their benign nature or maliciousness.

Malware Application (*Q*′)	**KLD (*P*′, *Q*′)**
AndroidDogwar	16.93
DroidDeluxe	17.25
DroidDreamlight2	17.25
DroidKungFu2A	12.47
DroidSlasher_1_1.0.1	12.47
HippoSMS	12.47
Lovetrap	12.47
Spitmo	16.38
Zitmo	17.25
zj_NinjaChicken_other	12.47

Table 3: KLD Between Good (*P*') and Malware (*Q*') Applications

Good Application (*Q*′)	**KLD (*P*′, *Q*′)**
Barcode Scanner	10.81
FxCamera	9.97
Huffington Post	11.82
My Currency – Converter	8.77
Skype	7.23
To-Do Calendar Planner	5.12
Viber	9.42
Virtual Table Tennis 3D	17.25
WhatsApp	12.32
YouTube	8.65

Table 4: KLD Between Good (*P*') and Good (*Q*') Applications

To further complement our evaluation, we randomly computed the KLD between the trained samples (*P*) and another new set of good applications performing SMS operations. Table 4 shows a snapshot of the obtained KLD values showing the divergence between good and good applications ranges between 5.12 and 17.25. Our experiment led to one false-positive warning. Considering the threshold values obtained from malware analysis in Table 4 (12.47-17.25), we find that Virtual Table Tennis 3D application is labeled as malware. The other nine applications are considered as benign. Thus, KLD can be a suitable measure to identify malware and benign applications for SMS operations if the threshold of divergence is considered carefully.

Here, we will demonstrate how another metric-based approach will give less accurate results when compared to applying our KLD-based approach. The metric-based approach is defined as follows:

A malicious application is defined as follows:

$$Sum(f_6 - f_{10}) \geq Sum(f_1 - f_5) \qquad (4)$$

A benign application is defined as follows:

$$Sum(f_1 - f_5) \geq Sum(f_6 - f_{10}) \qquad (5)$$

Application	***Sum* (f_1-f_5)**	***Sum* (f_6-f_{10})**
SMS_Android-Build-In-SMS-Application-Example	0	1
SMS_Android-Send-SMS-Example	3	0
SMS_AndroidSMSExample_1	3	0
SMS_AndroidSMSExample_2	1	0
SMS_apriorit_SecureMessages	0	0
SMS_Cloud SMS	1	0
SMS_Free SMS India	8	2
SMS_GO SMS Pro	11	2
SMS_Handcent SMS	0	0
SMS_javacodegeeks_AndroidSMSExample_1	3	0
SMS_MightyText.src	8	1
SMS_mkyong-Android-Send-SMS-Example	3	0
SMS_mkyoung-Android-Build-In-SMS-Application-Example	0	1
SMS_msatpathy_SMSTest	6	1
SMS_Ninja SMS	0	1
SMS_SecureMessages	0	0
SMS_SMSTest	6	1
Total	**53**	**10**

Table 5: Sum of Elements in the *P* Set

Table 5 compares the sum of the benign, *Sum(f_1-f_5)*, elements with the sum of the malicious, *Sum(f_6-f_{10})*, elements. We see that this metric-based approach does show that the total sum for all benign elements is greater than all of the malicious elements. When we compare the sums for each of the applications in the *P* set, we also see that most of the applications have a higher *Sum(f_1-f_5)*value that indicates the application is harmless. However, we also see in Table 5 that *Sum(f_1-f_5)*is not always greater than *Sum(f_6-f_{10})*. Three of the applications had a *Sum(f_1-f_5)*value that was less than the *Sum(f_6-f_{10})*. Our KLD-Based approach shows that all of the applications in the *P* set were within the benign threshold of values.

P set	Correct	14/17
	Incorrect	3/17

Table 6: Accuracy of Metric-Based Approach for the *P* Set

Next, we tested the metric-based approach on the suspected malicious applications in the *Q* set. Table 7 compares the sum of the benign, *Sum(f_1-f_5),* elements with the sum of the malicious, *Sum(f_6-f_{10}),* elements. As shown in Table 3, our KLD-Based approach shows that all of the applications in the malicious *Q* set fall within the threshold of values. In Table 8, we see that the accuracy of the metric-based approach continues to decrease even though it still holds true to our hypothesis.

Application	*Sum (f_1-f_5)*	*Sum (f_6-f_{10})*
AndroidDogwar	0	2
DroidDeluxe	0	1
DroidDreamlight2	0	1
DroidKungFu2A	0	0
DroidSlasher_1_1.0.1	1	1
HippoSMS	0	1
Lovetrap	1	1
Spitmo	0	2
Zitmo	0	1
zj_NinjaChicken_other	1	1

Table 7: Sum of Elements in the Malicious *Q* Set

Malicious *Q* set	Correct	6/10
	Incorrect	4/10

Table 8: Accuracy of Metric-Based Approach for the *Q* Set

Lastly, we tested the metric-based approach on the suspected benign applications in the other *Q* set. Table 9 compares the sum of the benign, *Sum(f_1-f_5),* elements with the sum of the malicious, *Sum(f_6-f_{10}),* elements.

Application	*Sum (f_1-f_5)*	*Sum (f_6-f_{10})*
Barcode Scanner	0	1
FxCamera	0	0
Huffington Post	0	0
My Currency – Converter	0	0
Skype	0	0
To-Do Calendar Planner	1	0
Viber	1	0
Virtual Table Tennis 3D	0	1
WhatsApp	0	0
YouTube	0	1

Table 9: Sum of Elements in the Benign *Q* Set

In Table 10, we see that the accuracy of the metric-based approach is poor in comparison to our KLD-Based approach. We received only one false-positive warning for the Virtual Table Tennis 3D application.

Benign *Q* set	Correct	7/10
	Incorrect	3/10

Table 10: Accuracy of Metric-Based Approach for the *Q* Set

Currently, our KLD-based approach is being executed as a desktop application. The average time to build our *P* set was a total of 0.146 seconds. The average time to build our malicious *Q* set was a total of 0.153 seconds. The average time to build our benign *Q* set was a total of 0.113 seconds. These average times are considered to be fairly efficient since they do not require an excessive amount of time to analyze the chosen applications and generate the CSV file that tracks the occurrence of the population elements. This performance would change once transitioning from an offline desktop application to a running service on a mobile device.

The offline analysis of scanning Android applications does not require an Internet connection. However, as malicious activities continue to evolve, the *P* set would require updating. Our initial intention for the deployment phase was to distribute the approach as a running service on the Android device. After careful consideration, we realized that the large variety of device hardware would affect the consistency of implementation and efficiency. The added constraint of declining battery power and device lifespan would deter users from running our service on their devices. In our future research, we plan to deploy our approach as a service in the cloud environment in order to maximize performance.

5 Conclusion

In this work, we propose to choose the Kullback-Liebler Divergence (KLD) as a measurement to differentiate between legitimate and malicious application behavior at source code level. The methodology builds probability distributions from the available source code of an application performing a specific functionality. We show some highlights of choosing possible elements of interest that can be useful to differentiate between a benign and malicious application behavior. Then, we apply the KLD measure to show that the difference between a legitimate and malicious application is infinite, whereas the difference between two legitimate applications is close to zero.

We believe that the application of KLD is very practical and simply deduces the elements of population for each functionality type into a threshold of values (which can identify a simple pass/fail). False positives were also investigated to ensure that the range of values is correct for both benign and malignant applications. We conclude that our application implementation of the KLD method accounts for more mitigation techniques. By examining the Android Manifest file (permission analysis), we can determine the intended functionality of each application and automatically generate its elements of population from a predetermined list. Using that information, our static analysis of the source code will yield very accurate results by checking for obfuscated code. Also, this is being done in an isolated environment (sandboxing) and the application is not being dynamically executed which greatly reduces risk of infection.

Our future research includes theoretical and implementation goals. On the theoretical side, our goals are: (i) choosing an appropriate smoothing technique to practically compute KLD, when one of the elements occurrence probability is found to be zero, (ii) finding more elements of population to cover more cases, (iii) documenting all possible known code patterns for performing specific functionality of interests that are common in malware applications, and (iv) validating our hypothesis using a larger collection of sample Android applications consisting of both legitimate and malicious behaviors.

On the implementation side, the conditions that we used to check the occurrence of population elements may not be exhaustive and accurate for all types of malware activities. However, we plan to create an interface where the end user can specify the population elements based on the activity. Our future goal includes automating the process for decompiling the APK file and analyzing the source code. We also plan to research the possibilities of deploying the application as a service in the cloud environment.

6 Acknowledgment

This work has been partially funded by a Graduate Research Assistantship from the School of Graduate Studies, Kennesaw State University, Georgia, USA.

7 References

[AARON, D. B. 2011]	AARON, D. B. (2011, November 17). Google android passes 50% of Smartphone Sales. *Bloomberg Businessweek*. Retrieved August 21, 2013, from http://www.businessweek.com/news/2011-11-17/google2android-passes-50-of-smartphone-sales-gartner-says.html.
[BALDWIN, C. 2012]	Baldwin, C. (2012, September 17). Android devices vulnerable to security breaches. *ComputerWeekly.com*. Retrieved August 21, 2013, from http://www.computerweekly.com/news/2240163351/Android-devices-vulnerable-to-security-breaches.
[BIGI, B. 2003]	Bigi, B. (2003). Using Kullback-Leibler Distance for Text Categorization. *Lecture Notes in Computer Science (LNCS*). Volume 2633, 2003, pp. 305-319.
[COOPER, V. N. 2014]	Cooper, V. N. (2014). Android Malware Detection Based on Kullback-Leibler Divergence", Invited Student Research Abstract to the SAC 2014 Student Research Competition (SRC) program. *Proceedings of the ACM-SIGAPP Conference on Applied Computing (SAC 2014)*, Gyeongju, Korea, March 2014, pp. 1695-1696.
[COVER, T. *et al*.2006]	Cover, T.& Thomas, J. *Elements of Information Theory*, John Wiley and Sons, 2006.
[FUKUI, K. *et al*. 2010]	Fukui, K., Sato, K., Mizusaki, J., & Numao, M. (2010). Kullback-Leibler Divergence Based Kernel SOM for Visualization of Damage Process on Fuel Cells. *IEEE International Conference on Tools with Artificial Intelligence*, October 2010, pp. 233-240.
[LI. H. *et al*. 2012]	Li, G. & Wang, Y. (2012). Differential Kullback-Leibler Divergence Based Anomaly Detection Scheme in Sensor Networks. In *Proceedings of 12th IEEE International Conference on Computer and Information Technology (CIT)*, October 2012, pp. 966-970.
[REZA. H. *et al*. 2012]	Reza, H. & Mazumder, N. (2012). A Secure Software Architecture for Mobile Computing. In *Proceedings of the 9th International Conference on Information Technology- New Generations* (ITNG 2012), Las Vegas, NV, pp. 566-571.
[PHONEPAYPLUS, 2013]	PhonePay Plus. *(2013). Phonepayplus.org.uk*. Retrieved August 21, 2013, from *http://www.phonepayplus.org.uk*.
[SARKAR, S. *et al*. 2007]	Sarkar, S., Rama, G. & Kak, A. (2007). API-Based and Information-Theoretic Metrics for Measuring the Quality of Software Modularization. *IEEE Transactions on Software Engineering*, January 2007, Vol. 33, No. 1, pp. 14-32.
[TAPIADOR, J. *et al*. 2010]	Tapiador, J. & Clark, J. (2010). Information-Theoretic Detection of Masquerade Mimicry Attacks. In *Proceedings of 4th International Conference on Network and System Security (NSS*), September 2010, pp. 183-190.

.