International Journal of Computer (IJC)

ISSN 2307-4523 (Print & Online)

http://ijcjournal.org/

Detection of Android Malware based on Sequence Alignment of Permissions

Franklin Tchakounté[a]*, Albert Djakene Wandala[b], Yélémou Tiguiane[c]

[a,b]*Faculty of Science, University of Ngaoundéré, Ngaoundéré, Cameroon*

[c]*Higher School of Computer Science, Université Nazi BONI, Bobo-Dioulasso, Burkina Faso*

[a]*Email: tchafros@gmail.com*

[b]*Email: djakenealbert@gmail.com*

[c]*Email: tyelemou@gmail.com*

Abstract

Permissions control accesses to critical resources on Android. Any weaknesses from their exploitation can be of great interest to attackers. Investigation about associations of permissions can reveal some patterns against attacks. In this regards, this paper proposes an approach based on sequence alignment between requested permissions to identify similarities between applications. Permission patterns for malicious and normal samples are determined and exploited to evaluate a similarity score. The nature of an application is obtained based on a threshold, judiciously computed. Experiments have been realized with a dataset of 534 malicious samples (300 training and 234 testing) and 534 normal samples (300 training and 234 testing). Our approach has been able to recognize testing samples (either malware or normal) with an accuracy of 79%, an average precision of 76% and an average recall of 75%. This research reveals that sequence alignment can improve malware detection research.

Keywords: Sequence alignment; permissions; Android; malicious; normal.

1. Introduction

Android is the mobile operating system most popular and is forecasted to remain popular until 2022 [1]. Its openness and popularity attracts malicious people which exploit sophisticated techniques to destroy their targets [2]. Android relies on the use of normal and dangerous permissions to control access to resources [3].

---

* Corresponding author.

Permissions are dangerous if they filter access to critical user resources. They are normal if once granted to an application, the risk is low. However, attackers exploit these permissions with bad intentions. Indeed, they try to combine multiple permissions to maliciously gain privilege levels to unauthorized resources [4]. Authors made several proposals based on permissions for detecting malware [5]. Authors exploit specific permissions to make decisions [6], combine permissions with other features and machine learning algorithms [7–9] and raise user awareness about risks to grant some permissions [6, 10, 11]. In this work, we use permissions as footprints to derive similarities among applications. Sequence alignment is so powerful in bioinformatics because it determines similarity between gene sequences [12], so that authors exploit them in data science [13] and for malware detection [14]. This work adapts this approach to determine similarity based on permissions for malicious and normal families. The adaptation of local alignment here supposes, as in reality, no order between apparitions of permissions. The proposed approach determines a DeoxyriboNucleic Acid (DNA) based permissions for the malicious family and do likewise for the normal family. Then, it computes a classification threshold based on the similarity score between the DNA of the tested application and the family DNAs. Experiments have been realized with a dataset of 600 samples (300 training and 300 testing) and 752 normal (326 training and 326 testing) applications. Our approach has been able to recognize testing samples (either malware or normal) with an accuracy of 79.58. This work reveals that Android applications are somehow similar based on their permissions. All the artefacts supporting this work are found in [15]. The rest of the document is structured in xxx sections. The first section presents concepts about sequence alignment and permissions. The second section presents the proposed approach. The third section presents results and discussions. The document ends with a conclusion and some perspectives.

2. Background

This section relates main concepts about sequence alignment and permissions.

2.1. Permissions

An application requests a permission to make operations on resources [3]. Such resources can be critical or sensitive to the user security. In this case, Google classifies protection permissions as dangerous. Permissions are normal when they protect less risky data. Android automatically prompts the user to either grant or deny dangerous permissions. Signature permissions include permissions granted at install time to applications signed by the same certificate as the application that defines those permissions. Permissions are declared in the manifest file by the developer based on application requirements. For example, if the application needs to send data to a distant server, the developer will add INTERNET permission. If the application needs to read contacts, the manifest file will include READ_CONTACTS. This work considers any type of permissions.

2.2. Sequence alignment

Sequence alignment is a technique used in bioinformatics to represent two or more sequences one under the other, to highlight common regions between DNAs [12]. The purpose of alignment is to arrange the components to identify areas of agreement. These alignments are carried out by software whose objective is to maximize the

number of coincidences of the elements in the different sequences. There are two types of alignments. The local alignment makes possible to search for similarity between parts of sequences whereas global alignment the whole sequences to perform. Local alignment is the most used since it is not as strict as the global alignment. The example illustrates a local alignment. It presents two sequences (TGK-G and AGKVG) with three similarities (second position, third position and fifth position) and two differences (first position and fourth position).

T G K – G

A G K V G

It gives a score of alignment equals to $\text{score} = \frac{(\text{id(T,A)}+\text{id(G,G)}+\text{id(K,K)}+\text{id}(-,\text{V})+\text{id(G,G)})}{n} = \frac{0+1+1+0+1}{5} = 60\%$

, where

$$x = \begin{cases} 1 & if\ X = Y \\ 0 & otherwise \end{cases}$$

This work takes a DNA element as a permission. Since developers just include their appearance in the manifest file, the sequence order is not relevant for applications. Let consider the two manifests in Table 1.

Table 1: Manifest examples

Manifest 1	Manifest 2
INTERNET	WRITE_SMS
READ_CONTACTS	READ_PHONE_STATE
WRITE_CONTACTS	INTERNET
WRITE_SMS	READ_CONTACTS

The position in which a permission appears is not important. A possible DNA of the first manifest is (INTERNET - READ_CONTACTS - WRITE_CONTACTS - WRITE_SMS) and the second (INTERNET-READ_CONTACTS - - - WRITE_SMS). The sequence alignment in this case is

INTERNET READ_CONTACTS WRITE_CONTACTS WRITE_SMS

INTERNET READ_CONTACTS ---- WRITE_SMS

$$\text{score} = \frac{1+1+0+1}{5} = 60\%$$

2.3. Evaluation measures

Some metrics are used within the scope of this work to test the reliability of the similarity approach[a].

- *True Positive* (TP): It the number of malware correctly detected as malware.
- *True Negative* (TN): It is the number of benign samples correctly detected as normal
- *False Positive* (FP): It the number of malware incorrectly detected as malware.
- *False Negative* (FN): It is the number of benign samples incorrectly detected as normal
- *Accuracy:* It is the rate of correctly detected samples within the whole dataset

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- *Precision for positive:* It is the proportion of malware identifications which was actually correct.

$$Precision_{positive} = \frac{TP}{TP + FP}$$

- *Precision for negative:* It is the proportion of benign identifications which was actually correct.

$$Precision_{negative} = \frac{TN}{TN + FN}$$

- *Recall:* It is the proportion of actual malware that was identified correctly.

$$Recall = \frac{TP}{TP + FN}$$

3. Methodology

Our approach to detecting android malware uses techniques used in bioinformatics to detect similarity regions between sequences DNA, RNA and others. For our case, it is a question of detecting regions of similarity between malicious Android applications using their sequences of permissions, which, unlike DNA sequences, do not change according to the order of appearance of the permissions. The goal here is to find a score based on sequence alignment between the DNA of any tested application and the DNA of the whole malware samples. Then to compute a similarity threshold from which a decision making can be made whether an application is normal or malicious. To achieve this objective, this study follows different steps:

- Structuration of applications: This step aims to extract relevant information such as permission and to organize them.
- Threshold determination: This step selects the threshold value providing the best classification performance, according to the dataset of malware.
- Malware detection: This step experiments the detection of malware and normal based on the similarity threshold.

[a] https://developers.google.com/machine-learning/crash-course/classification/video-lecture

3.1. Structuring the dataset

Let be E_{Mal}, the set of malicious samples and E_{Nor}, the set of normal samples. We extract the manifest file *AndroidManifest.xml* of each application *A* belonging to E_{Mal}. These permissions are included in *AllPerm*. The set of permissions of *A* is $\text{AllPerm}_{A,E_{mal}} = \{\text{Perm} \in \text{A}, \text{A} \in E_{Mal}\}$.

Now we remove duplicated permissions to obtain the DNA of A called $\text{StrictAllPerm}_{A,E_{mal}}$.

$$\text{StrictAllPerm}_{A,E_{mal}} = \left\{\text{Perm} \in \text{AllPerm}_{A,E_{mal}} \wedge \exists \text{Q} \in \text{AllPerm}_{A,E_{mal}}, P \neq Q\right\}$$

The sets of distinct permissions for each application are merged to obtain the DNA of the whole malware dataset, namely DNAModel_{Mal}

$$\text{DNAModel}_{Mal} = \{\forall \text{A} \in E_{Mal}, \cup \text{StrictAllPerm}_{A,E_{mal}}\}$$

We create $\text{CountModel}_{\text{Mal}}$ which is the set of occurrences for each permission belonging to DNAModel_{Mal} and $\text{CountModel}_{\text{Nor}}$ which is the set of occurrences of each permission belonging to DNAModel_{Mal} in the set of normal applications E_{Nor}.

$$\text{DNADiffModel}_{Mal} = \{\forall \text{A} \in \text{DNAModel}_{Mal}, |occurrence_{E_{Mal}}(\text{A}) - occurrence_{E_{Nor}}(\text{A})|\}$$

$\text{DNADiffModel}_{Mal}$ keeps the gap between occurrences of the frequent permissions used by malware and occurrences of these permissions in benign applications. It measures therefore the degree of representativeness of permissions in malicious and benign applications.

3.2. Threshold determination

This phase aims to define from which value an application is similar to a malware or goodware. The following pseudo-code is performed to achieve this task. The objective is to find the best threshold which discriminates efficiently between malicious and normal.

Algorithm 1: Determination of threshold

Begin

1. Inputs: E_{Mal} and E_{Nor}; Output: ∂
2. We sum up all possible occurrences of permissions of malware

$$\text{S} = \text{S} + \text{DNADiffModel}_{Mal}(\text{i}), \qquad 1 \leq \text{i} \leq \text{sizeof}(\text{DNADiffModel}_{Mal}).$$

3. **for** any $\partial = 0$ *to* 1

For any application *u* in E_{Mal}

a. Determine the DNA of *s*, $\text{StrictAllPerm}_{u,E_{mal}}$

b. Compute the sequence alignment score $score_{alignment}$ between its DNA and DNAModel_{Mal}

$Score_{alignment} = \sum CountModel_{Mal}(\text{position})$, *Position* is the index for each permissions included in the DNA of u

c. **if** $\frac{Score_{alignment}}{S} \geq \partial$ **then** TP=TP+1

else FN=FN+1

endif

endfor

for any application *v* in E_{Nor}

d. Determine the DNA of *v*, $\text{StrictAllPerm}_{v,E_{Nor}}$

e. Compute the sequence alignment score $score_{alignment}$ between its DNA and DNAModel_{Mal}

$Score_{alignment} = \sum CountModel_{Mal}(\text{position})$, *Position* is the indexes of permissions included in the DNA of *v*

f. **if** $\frac{Score_{alignment}}{S} \geq \partial$ **then** FP=FP+1

else TN=TN+1

endif

endfor

g. compute five performance metrics : accuracy, $\text{precision}_{positive}$, $\text{precision}_{negative}$ and recall.

$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, $Precision_{positive} = \frac{TP}{TP+FP}$, $Precision_{negative} = \frac{TN}{TN+FN}$; $Recall_{positive} = \frac{TP}{TP+FN}$, $Recall_{negative} = \frac{TN}{TN+FP}$

h. compute the recall gap (Diff_Recall) such that $Diff\_Recall = |Recall_{positive} - Recall_{negative}|$

endfor

4. select ∂ such that $(Diff_{Recall} is\ the\ lowest)$
5. if there is only one ∂ that meets the previous condition return ∂
6. else if there are several ∂s, return $average\ of\ \partial s$

End

There are two datasets: malicious samples and normal samples (line 1). The sequence alignment between its DNA and the DNA of the whole malware is applied for every malicious and normal sample A (lines 3.b and 3.e). Then, we compute a score based on found permissions in A by summing their occurrence gaps within the malicious dataset (lines 3.b and 3.e). This score is exploited to check whether the sample is correctly classified or not. Several metrics are saved for each threshold (lines 3.c and 3.f). They are True Positive (TP), False Negative (FN), True Negative (TN), False Negative (FN), Accuracy, Precision for malware, Precision for normal applications and Recall (line 3.g). We finally select the threshold with acceptable accuracy, precisions and recall simultaneously (line 4). This condition is not exclusive because accuracy alone is not enough for class-imbalanced dataset. That is why we couple the other metrics such as precision and recall.

3.3. Malware detection

This phase aims to decide whether an application is malicious or benign based on the selected threshold ∂.

We test the condition $\frac{Score_{alignment}}{S} \geq \partial$. If it is satisfied then the application is malware. Otherwise, it is a benign application.

- $0.9 <$ accuracy <1, the approach is excellent
- $0.8 <$ accuracy <0.9, the approach is good
- $0.7 <$ accuracy <0.8, the approach is acceptable
- $0.6 <$ accuracy <0.7, the approach is not good

4. Experiments and results

This section describes experiments and discusses results.

4.1. Datasets

We have gathered 1252 samples including 626 malicious and 626 benign applications. The dataset of malicious samples is split into 300 training samples to determine the threshold and 326 testing samples to evaluate the model. Likewise, the dataset of benign samples is split into 300 training samples to determine the threshold and 326 testing samples to evaluate the model. We have ensured that malicious and benign samples are disjoints as well as both training and testing for malicious and normal. We proceeded by checking the contents and hashed of each application. We collected the malicious samples from the Android malware dataset CICAndMal2017[b]. We collected the benign samples from Google Play.

Tools exploited

[b] https://www.unb.ca/cic/datasets/andmal2017.html

We have used the apktool 2.3.4 to reverse engineer applications from binary into readable data. Moreover, we used Python 2.7 with pandas library, numpi, csv, lxml and other libraries to dissect permissions of applications.

4.2. Threshold determination

We have experimented this phase with 300 malicious and 300 benign samples. Details of extraction are available in the Github page dedicated to this study [15]. A script written in Python has been used to automate the threshold process. The ADN of the whole malware with occurrences of each permission is also available in [15]. Table 2 shows the results obtained. We see that more the threshold grows more the precision for positive and recall for negative tend to 1. It means in this case that the model is more precise to detect malware. In the contrary, the model is more precise for detection of benign applications when the threshold tends to zero (precision for negative). However, decision making is not made based on metrics taken separately and should be optimal for malware and benign applications simultaneously.

Table 2: Experiments for the threshold determination

Threshold	TP	FN	TN	FP	Precision for positive	Precision for negative	Recall for negative	Recall for positive	Diff_Recall	Accuracy
0.05	293	7	55	245	0,54	0,88	0,18	0,97	0,79	0,58
0.1	293	7	68	232	0,55	0,90	0,22	0,97	0,75	0,60
0.15	282	18	83	217	0,56	0,82	0,27	0,94	0,66	0,60
0.2	273	27	99	201	0,57	0,78	0,33	0,91	0,58	0,62
0.25	266	34	126	174	0,60	0,78	0,42	0,88	0,46	0,65
0.3	264	36	150	150	0,63	0,80	0,5	0,88	0,38	0,69
0.35	260	40	168	132	0,66	0,80	0,56	0,86	0,30	0,71
0.4	256	44	195	105	0,70	0,81	0,65	0,85	0,20	0,75
0.45	248	52	217	83	0,74	0,80	0,72	0,82	0,10	0,77
0.5	247	53	232	68	0,78	0,81	0,77	0,82	0,05	0,79
0.55	229	71	244	56	0,80	0,77	0,81	0,76	0,05	0,78
0.6	222	78	260	40	0,84	0,76	0,86	0,74	0,12	0,80
0.65	212	88	280	20	0,91	0,76	0,93	0,70	0,22	0,82
0.7	206	94	283	17	0,92	0,75	0,94	0,68	0,25	0,81
0.75	184	116	285	15	0,92	0,71	0,95	0,61	0,33	0,78
0.8	163	137	288	12	0,93	0,677	0,96	0,54	0,41	0,75
0.85	149	151	289	11	0,93	0,65	0,96	0,49	0,46	0,73
0.9	135	165	292	8	0,94	0,63	0,97	0,45	0,52	0,71
0.95	131	169	293	7	0,94	0,63	0,97	0,43	0,54	0,70

We observe that that the lowest recall gap is 0.05 and two thresholds correspond to it. The first is 0.5 and the

second 0.55. According to Algorithm 1, we take the mean of both. That means, the threshold is 0.525. This threshold will be used for classification.

4.3. Evaluation of classification

Two new distinct datasets are exploited for testing the similarity approach. Table 3 is obtained with 234 malicious samples and 234 normal samples under a threshold of 0.525.

Table 3: Performance results

Threshold	TP	FN	TN	FP	Precision for positive	Precision for negative	Recall for negative	Recall for positive	Accuracy
0.525	194	40	177	57	0,71	0,81	0,69	0,82	0,79

This model is accurate with 79% of correct detections with an average precision of 76%. The model is more precise to detect normal (81%) than malware (71%). However, the model is able to correctly observe 82% of malware (recall of 82%). Although these results demonstrate that our approach is reliable, we note that forty malware and 57 benign applications that are mistakenly classified. An association with features such API and code monitoring as well as runtime analysis can reveal other traces to improve this classification.

5. Related work

Several authors have made two orientations to identify malware based on permissions. The first orientation includes works which rely only permissions to raise risk levels to assist users being aware. Sarma and his colleagues [6] propose an approach using the permissions requested by an application, its category and what permissions are requested by other applications in the same category to better inform users about the risks of installing this application really fits with its objectives. PUREDroid [10] evaluates the security risk related to granted dangerous permissions as well as their negative impact. Al Jutail and his colleagues [11] propose to build an application to monitor for dangers associated to permissions of the scanned applications. Then, this tool presents results in an understandable manner to the normal user who can therefore determine whether an application can affect privacy or not. The second orientation includes works which combine permissions to other features using machine learning. Its main objective is to improve detection performance. DroidAPIMiner [9] associates dangerous APIs and critical permissions, and other features with machine learning algorithms. Liu and Liu […] rely on requested permissions and required permissions as features of machine learning techniques to classify an application as benign or malicious. Drebin [16] associates static analysis of permissions and Application Programming Interfaces (APIs) with machine learning to identify malware. SaMaDroid [8] proposes a three-level detection architecture which extracts requested permissions, APIs and other features from the manifest. These features, once structured, are transferred to the learning process to guess the class of the application. Likewise, SigPID [4] retrieves significant permissions from applications, structures this information to effectively detect malware using supervised learning algorithms. This work does not aim to raise awareness about permissions or to look for features providing better performance. But, it relies on the sequence alignment

principle to find the optimal score similarity between the query application and the permission DNAs of malware. It is more lightweight than the second orientation's works.

6. Conclusion

This work has proposed a malware detection based on the alignment of permissions. This work relies on sequence alignment principle to determine similarity based on permissions for malicious and normal families. A classification threshold based on the similarity score between the DNA of the tested application and the family DNAs, is determined. Experiments has been realized on dataset of 600 samples (300 training and 300 testing) and 752 normal (326 training and 326 testing) applications. The proposed approach is able to recognize testing samples (either malware or normal) with an accuracy of 79.58. This work reveals that Android applications are somehow similar based on their permissions. However, we still have to improve by combining with other approaches and other features to improve detection performance.

References

[1] Statista, "Smartphone unit shipments worldwide by operating system from 2016 to 2022 (in million units)," 2019. [Online]. Available: https://www.statista.com/statistics/309448/global-smartphone-shipments-forecast-operating-system/. [Accessed: 17-Jul-2019].

[2] GDATA, "Some 343 new Android malware samples every hour in 2017," 2018. [Online]. Available: https://www.gdatasoftware.com/blog/2018/02/30491-some-343-new-android-malware-samples-every-hour-in-2017. [Accessed: 28-Jul-2019].

[3] Android developers, "Permissions overview," 2019. [Online]. Available: https://developer.android.com/guide/topics/permissions/overview. [Accessed: 17-Jul-2019].

[4] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, Jul. 2018.

[5] A. Qamar, A. Karim, and V. Chang, "Mobile Malware Attacks: Review, Taxonomy & Future Directions," *Future Generation Computer Systems*, vol. 97, pp. 887–909, Aug. 2019.

[6] B. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Android Permissions: A Perspective Combining Risks and Benefits," in *Symposium on Access control Models and Technologies*, 2012, pp. 13–22.

[7] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, Jul. 2018.

[8] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," *IEEE Access*, vol. 6, pp. 4321–4339, 2018.

[9] Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android," Springer, Cham, 2013, pp. 86–103.

[10] A. Alshehri, P. Marcinek, A. Alzahrani, H. Alshahrani, and H. Fu, "PUREDroid: Permission Usage and Risk Estimation for Android Applications," in *Proceedings of the 2019 3rd International Conference on Information System and Data Mining - ICISDM 2019*, 2019, pp. 179–184.

[11] M. Al Jutail, M. Al-Akhras, and A. Albesher, "Associated Risks in Mobile Applications Permissions," *Journal of Information Security*, vol. 10, pp. 69–90, 2019.

[12] A. Zielezinski, S. Vinga, J. Almeida, and W. M. Karlowski, "Alignment-free sequence comparison: benefits, applications, and tools.," *Genome biology*, vol. 18, no. 1, p. 186, 2017.

[13] M. Vijini, "Pairwise Sequence Alignment using Biopython – Towards Data Science," 2017. [Online]. Available: https://towardsdatascience.com/pairwise-sequence-alignment-using-biopython-d1a9d0ba861f. [Accessed: 02-Mar-2019].

[14] J. M. Vidal, M. A. S. Monge, and L. J. G. Villalba, "A novel pattern recognition system for detecting Android malware by analyzing suspicious boot sequences," *Knowledge-Based Systems*, vol. 150, pp. 198–217, Jun. 2018.

[15] Djakene, "Malwares-Detection-based-on-sequences-alignment-of-permissions," 2019. [Online]. Available: https://github.com/djakene/Malwares-Detection-based-on-sequences-alignment-of-permissions. [Accessed: 01-Aug-2019].

[16] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in *Proceedings 2014 Network and Distributed System Security Symposium*, 2014.