機械学習を用いたAndroidマルウェア検知の再考

YAMAZAKI, Kazuki / 山崎, 一希

(出版者 / Publisher)
法政大学大学院理工学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 理工学研究科編
(巻 / Volume)
63
(開始ページ / Start Page)
1
(終了ページ / End Page)
6
(発行年 / Year)
2022-03-24
(URL)
https://doi.org/10.15002/00025388

機械学習を用いた Android マルウェア検知の再考

RETHINKING OF ANDROID MALWARE DETENCITION USING MACHINE LEARNING

山崎 一希
Kazuki YAMAZAKI
指導教員 彌冨仁

法政大学大学院理工学研究科応用情報工学専攻修士課程

The Android OS has been gaining market share in recent years due to the increase in IoT devices and the popularity of smartphones. In general, attackers target systems with a large number of users, and the number of Android malware victims has been increasing. In 2017, it was reported that 750,000 new Android malware were discovered in the first quarter alone. In 2017, 750,000 new Android malware were reported to have been discovered in the first quarter of the year alone, making Android malware an urgent issue to be addressed. In the past, three methods of malware detection were used: surface analysis, dynamic analysis, and static analysis. However, these methods have disadvantages in terms of human and time costs. Recently, in addition to these conventional rule-based methods, detection methods using machine learning have been reported. In particular, methods based on deep learning have been attracting attention. Compared to classical machine learning methods such as logistic regression, methods based on deep learning are said to have higher generalization performance and more accurate detection capability. Many reports on the application of deep learning techniques have shown accuracies of over 90%. However, in the case of Android malware, which is difficult to collect data, it is often evaluated within the same dataset, and overtraining on the dataset is suspected. In this study, we evaluate the generalization performance of the model on both known and unknown datasets. In addition, by comparing classical machine learning and deep learning, we reconsidered the pros and cons of using deep learning. As a result, we found that the discrimination accuracy of the unknown dataset is about 10 15% lower than that of the known dataset. In addition, there was no significant difference in accuracy between classical machine learning and deep learning.

Keywords *: Android malware, machine learning*

1. はじめに

スマートフォンや IoT の普及により，Android OS は Windows OS を抑えトップのシェアを誇っている [1]．多くの場合，攻撃者はシェアの高いものに対し攻撃を仕掛ける．2017 年には第 1 四半期だけで 750,000 体の新たな Android マルウェアが発見されたという報告も上がっている [2]．このことから Android マルウェアへの対策は急務であると言える．

一般的に，あるアプリケーションがマルウェアであるかどうかを判定するために，3 つの解析手法が用いられる [3]．1 つ目は表層解析である．表層解析は，ファイル中に含まれる文字列などの調査する．含まれている文字列からは，新たな攻撃の起点となるサーバのドメインや URL 等を得ることができる可能性がある．2 つ目の動的解析では，サンドボックス環境でアプリケーションを動かしながら解析していく．実際に動かし，監視やトレースすることで悪意のある通信などを見つけやすくなる．そして，上記の二つの解析手法を踏まえて，さらに細かく調査する場合，3 つ目の手法である静的解析が用いられる．静的解析は，逆アセンブルや逆コンパイルして得られるソースコード情報を元に解析する手法であり，特に Android アプリの形式である APK(Android Package Kit) ファイルの場合，逆コンパイルすることで，Java のソースコードなどを復元することができる．例えば，静的解析を用いることにより，マルウェア作成者が解析を妨害するために利用する難読化された変数名や関数名といった特徴を抽出することができる．しかしながら，日々マーケットにアップロードされ増えていく新しいアプリケーションに対して，これらの手法を用いて解析する場合，多くの人的コストと時間がかかる．それを受けこれらの手法を自動化する方法が多数提案されている [4–13]．Enck et al. [10] はアプリケーションの動作を監視し，異常を検知する動的解析ベースのマルウェア検知システム Taint-Droid を提案している．表層解析をベースとしたシステムとして Canfora et al. [11] が Android Dalvik オペコードを監視し，そのオペコードの頻度に基づいて悪意あるアプリケーションか否かを識別するシステムを提案している．静的解析をベースとしたシステムとしては Mercaldo et al. がソースコードを CSS(calculus for communicating systems) 文に変換し，ソフトウェアの動作をチェックするメソッドを

利用して，ランサムウェア攻撃を防止する方法を提案している．また Damshenas et al. [12] はサーバー上でアプリケーションを実行してそのシステムコールを解析し，シグネチャを生成することで，シグネチャベースにユーザーへ渓谷を行う静的解析と動的解析を組み合わせた M0Droid と呼ばれる分散型システムを提案している．しかし，これらの手法でも人的なコストの問題は解決されておらず，また，新種や亜種のマルウェアが現れた場合，静的解析や動的解析などから得られる Android マルウェアの特徴を手動で更新するのは困難であるという問題がある．

そこで近年ではこれら従来のルールベースのシステムに加え，機械学習技術を用いることで Android マルウェアの検知を試みた報告がなされている [14–18].

Afonso et al. [17] は Andorid アプリケーションの API 呼び出しやシステムコールを利用し，サポートベクターマシン，ベイジアンネットワーク，決定木などいくつかの古典的な機械学習でマルウェア検出を行い，0.968 の accuracy を報告している．また，Milosevic et al. [19] は古典的な機械学習を応用した検出方法を提案している．Milosevic et al. は，APK ファイルを逆コンパイルして得られる，パーミッション情報やソースコードを利用して，ランダムフォレストやロジスティクス回帰，サポートベクターマシンなどの代表的な機械学習アルゴリズム，また，それらのアンサンブル学習を用いて，0.956 の f1-score を達成している．

さらに近年の研究では，機械学習技術の発展に伴い様々な機械学習手法の中でも深層学習を利用した手法が多く提案されている [3, 20–31]．Mclaughlin et al. [32] は展開した APK ファイルからソースコードである DEX ファイルを取り出し，逆アセンブルしたのちに畳み込みニューラルネットワーク (CNN) で識別する手法を提案し，高い精度を報告している．また，Hota et al. [29] は Mclaughlin et al. と同様に Dex ファイルを取り出し，その byte 列を 1 つのドキュメントとみなして Doc2Vec [33] による，Dex ファイルの固定長ベクトル変換を行い，CNN を用いて識別を行うモデルを提案した．しかし APK ファイルを展開する場合，動的解析を行う方が確実であったり，検知に時間がかかるなどの問題がある．

この問題に対し，Hasegawa et al. [34] は，APK ファイルの展開を行わず先頭あるいは末尾 512 バイトから 4,096 バイトを入力とし，1-D CNN を用いて識別を行い，10 分割交差検証において 95.4% から 97.0% の識別精度を達成している．

深層学習を応用した手法は古典的な機械学習に比べて，汎化性能が高く，より高精度な検知能力を誇ると言われている．しかし Android マルウェアを初めとするマルウェアのデータは収集が難しく，大量のデータを用意することが困難である．それに伴い同一のデータセット内で評価を行っていることもある．Android マルウェアの検知を対象とした多くの研究で 90% を超える精度が報告されているが，少量のデータを利用していることに伴うデータセットへの過学習が疑われる．このことから多くの報告で深層学習モデルを利用した手法が期待する汎化性能の向上が実現できているか疑問が残る．

Table 1　モデルにとって既知となるデータについて

データセット	AMD	Drebin	グッドウェア
データ数	24,629	5,556	5,247
学習に用いたデータ数		27,164	4,697
評価に用いたデータ数		3,021	550
合計		30,185	5,247

本研究では静的解析で頻繁に利用される Android アプリケーションにおけるソースコードである dex ファイルを未展開の APK ファイルから取り出し入力として古典的な機械学習手法とディープラーニングモデルで検知を行いその検知性能を比較する．またディープラーニングモデルに期待される未知データへの検出能力を調べるために，学習では利用しないデータセットに対する検知精度を評価する．

2. 実験

本研究では古典的な機械学習手法であるロジスティック回帰，ランダムフォレスト，サポートベクターマシンと深層学習手法である CNN を使ったモデルを比較する．

(1) データセット

識別モデルにとって既知となるデータセット (以後，既知データセットと呼ぶ) として 3 つのデータセットを使用した．その一部を評価データとして使用して，後述するモデルにとって未知となるデータセット (以後，未知データセットと呼ぶ) に対する識別能力と比較した．既知データセットはグッドウェアには Hasegawa et al. [34] で利用されているデータセットと同様のものを利用した．その詳細は Appsapk [35] と Apkpure [36] から Android アプリケーションを収集し，合計 62 個のツールを用いて検査する VirusTotal [37] による検査を行い，悪性と判断したツールが 2 個以下の合計 5,247 体である．マルウェアには AMD [38] データセットと Android Drebin Project [39], [40] データセットを利用した．AMD データセットは合計 24,629 体のマルウェアからなるデータセットである．Android Drebin Project データセットは合計 5,556 体のマルウェアからなるデータセットである．これらの合計 30,185 体のマルウェアと 5,247 体のグッドウェアを学習: 評価=9:1 にわけて使用した．Table2 に具体的な数字をまとめる．

未知データセットはモデルにとって未知のデータでテストを行うために CICAndMal2017 データセット [41] を利用した．このデータセットは 2015 年から 2017 年に収集されたグッドウェア 1,700 体とマルウェア 426 体，合計 2,126 体のデータセットである．またマルウェアの分類ごとに分けられておりその内訳はアドウェア 104 体，ランサムウェア 101 体，スケアウェア 112 体，SMS マルウェア 109 体となっている．

Table 2　モデルにとって未知となるデータについて

データセット名	CICAndMal2017	
分類	マルウェア	グッドウェア
データ数	426	1,700
学習に用いたデータ数	0	0
評価に用いたデータ数	426	1,700

(2)　**古典的機械学習手法**

a)　**入力**

入力するデータは未展開の APK ファイルから抽出した DEX ファイルを使用した．DEX ファイルは Android アプリケーションにおけるソースコードにあたるファイルであり，機械学習を用いた識別で多く利用されているファイルの一つである．

b)　**前処理**

前処理には Bag-of-words を適用した．Bag-of-words は主に自然言語処理で利用される方法で，文章にその単語が含まれるかどうかを判定する方法である．その文章中に指定した単語がいくつ含まれるかをカウントする．本実験の入力は byte 列であるためそのバイトがどれだけ出現したかをカウントした．

c)　**モデル**

モデルにはロジスティック回帰，ランダムフォレスト，サポートベクターマシンを使用した．

(3)　**深層学習手法**

a)　**入力**

入力するデータは古典的機械学習手法と同様に未展開の APK ファイルから抽出した DEX ファイルを使用した．DEX ファイルは Android アプリケーションにおけるソースコードにあたるファイルであり，機械学習を用いた識別で多く利用されているファイルの一つである．

b)　**前処理**

前処理は 2 つの方法を使用した．1 つ目は Hasegawa et al. [34] が提案した手法の中で最も高い精度が報告されている，先頭 1,024Bytes を切り出す方法である．本実験では APK ファイルの先頭ではなく抽出した DEX ファイルの先頭を使用した．2 つ目は Hota et al. [29] が Andrid マルウェアの識別における特徴抽出方法として応用したドキュメント同士の関係をベクトルで表現する Doc2Vec を使用して，各 DEX ファイルを 1024 次元のベクトルとして出力した．

c)　**モデル**

実験で利用するモデルは Hasegawa et al. [34] の研究で利用されているモデルと同様の 4 層 1-D CNN を利用した．ただし，出力の活性関数を softmax から sigmoid に変更した．全ての畳み込み層は 5×1 のフィルタを 32 枚持ち，すべての Maxpooling 層のプーリングサイズは 5×1 である．畳み込み層においてストライドは 1 とした．また，すべての畳み込み層に Batch Normalization と活性化関数 ReLU を適用した．入力を 1,024 バイトとしたときのモデルの構成を table 3 に示す．

Table 3　1-D CNN モデルの構成

入力 (1024×1)
畳み込み層 (5×1)-32
Batch normalization
ReLU
畳み込み層 (5×1)-32
Batch normalization
ReLU
Maxpooling 層 (5×1)-32
畳み込み層 (5×1)-32
Batch normalization
ReLU
畳み込み層 (5×1)-32
Batch normalization
ReLU
Maxpooling 層 (5×1)-32
全結合層
Sigmoid

Table 4　Accuracy

モデル	Accuracy(既知)	Accuracy(未知)
ロジスティック回帰	0.945	0.676
ランダムフォレスト	0.938	0.609
SVM(rbf)	0.951	0.630
SVM(linear)	0.870	0.730
Doc2Vec+CNN	0.833	0.200
CNN(Hasegawa method)	0.967	0.687

Table 5　AUC

モデル	AUC(既知)	AUC(未知)
ロジスティック回帰	0.929	0.764
ランダムフォレスト	0.958	0.841
SVM(rbf)	0.929	0.802
SVM(linear)	0.908	0.758
Doc2Vec+CNN	0.467	0.453
CNN(Hasegawa method)	0.986	0.876

3.　評価

未知のデータに対する識別精度の評価を行うために既知データセットに対する accuracy と Receiver Operator Characteristic(ROC) 曲線の下面積である Area Under the Curve(AUC) を算出し，各モデルごとの精度を比較した．

4.　結果

各モデルの既知データセット，未知データセットに対する accuracy を table.4 に示す．

各モデルの既知データセット，未知データセットに対する AUC を table.5 に示す．

Doc2vec+CNN の混同行列を Fig.1 に示す．

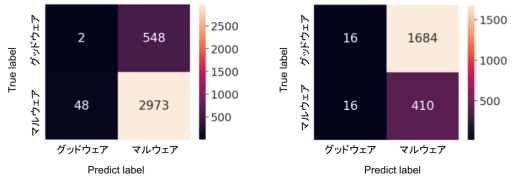

Figure 1　既知データ (左) と未知データ (右) に対する Doc2Vec+CNN モデルの混同行列

5.　考察

(1)　古典的機械学習手法と深層学習手法の比較

Table4 より，既知データに対する accuracy で古典的な機械学習手法の中で最も高いものは SVM(rbf) の 0.951 であった．深層学習手法では CNN(Hasegawa method) の 0.967 であった．未知データに対する accuracy では SVM(linear) が 0.730 が最も高く，次点で CNN(Hasegawa method) の 0.687 であった．この結果から古典的な手法と深層学習手法では accuracy においては大きな精度の差は見られないことがわかる．Table5 より，既知データに対して最も高いスコアが得られたのは CNN(Hasgawa method) の 0.986 で，未知データに対しても最も高いスコア 0.876 が得られていることが分かる．時点ではランダムフォレストであり既知データに対するスコアが 0.958，未知データに対するスコアが 0.841 である．こちらでも 2 つのモデル間のスコアに大きな差が出ることはなかった．

(2)　既知データと未知データに対する検知能力の比較

Table4, 5 より，どのモデルにおいても未知データに対するスコアは既知データに対するスコアに比べて小さくなっていることがわかる．これは Andorid マルウェアの検知において，データセットのドメインに対する過学習が起きていることが原因と考えられる．マルウェアの収集年や収集環境によって似たようなデータが集まることがこれを引き起こしていると考えられる．しかし，Doc2Vec を適用したモデルを除き，どのモデルでも未知データに対して一定の識別能力は発揮できている．そのため機械学習技術を用いたマルウェア検知自体は発展の要素があると言える．

(3)　Doc2Vec を用いたモデルについて

Table4, 5 より，Doc2Vec + CNN モデルは他のモデルに比べて著しくスコアが落ちていることがわかる．Fig.1 を見ると多くのデータに対してマルウェアであるという識別を行ってしまっていることが分かる．これは Dex ファイル内におけるマルウェアらしさを示す部分がファイル全体に比べるとごく一部であることが，Doc2Vec で固定長ベクトルに変換する際にマルウェアらしさを特徴としてうまく出力できなかったことが原因ではないかと考えられる．これにより CNN での学習時にデータ数の多いマルウェアと識別してしまうモデルになってしまったと考えられる．

6.　まとめと今後の展望

本研究ではモデルにとって既知であるデータと未知であるデータそれぞれに評価を行うことで機械学習を用いた Andoirid マルウェア検知が，本当に汎化性能を獲得しているのかを確認するとともに，古典的機械学習と深層学習の間にどれだけ差があるのかを確認した．その結果，多くのデータを収集することが難しい Android マルウェア検知では古典的機械学習，深層学習ともに未知データに対しての識別は精度が下がってしまうことが分かった．また，古典的機械学習と深層学習手法の間でも大きな識別能力の差が見られなかったことから，マシンコストの点で古典的機械学習手法を使うことに十分なメリットがあるということがわかった．しかし深層学習手法の方がやや高い精度が出ることも分かったので今後はモデルの見直しなどで未知データに対する識別能力の向上を目指していく．

参考文献

[1] “Operating system market share worldwide — statcounter global stats.” https://gs.statcounter.com/os-market-share. (Accessed on 09/05/2019).

[2] G. DATA, “8400 new android malware samples every day.,” 2017.

[3] T. K. Barsiya, M. Gyanchandani, and R. Wadhwani, “Android malware analysis: A survey paper,” *International Journal of Control, Automation, Communication and Systems (IJCACS)*, vol. 1, no. 1, pp. 35–42, 2016.

[4] M. Christodorescu and S. Jha, “Static analysis of executables to detect malicious patterns,” in *12th USENIX Security Symposium (USENIX Security 03)*, 2003.

[5] M. Christodorescu, S. Jha, and C. Kruegel, “Mining specifications of malicious behavior,” in *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pp. 5–14, 2007.

[6] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, “Learning and classification of malware behavior,” in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 108–125, Springer, 2008.

[7] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, “Google android: A comprehensive security assessment,” *IEEE Security & Privacy*, vol. 8, no. 2, pp. 35–44, 2010.

[8] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. Clausen, O. Kiraz, K. A. Yuksel, S. A.

Camtepe, and S. Albayrak, "Static analysis of executables for collaborative malware detection on android," in *2009 IEEE International Conference on Communications*, pp. 1–5, IEEE, 2009.

[9] T. Bläsing, L. Batyuk, A.-D. Schmidt, S. A. Camtepe, and S. Albayrak, "An android application sandbox system for suspicious software detection," in *2010 5th International Conference on Malicious and Unwanted Software*, pp. 55–62, IEEE, 2010.

[10] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, pp. 1–29, 2014.

[11] G. Canfora, F. Mercaldo, and C. A. Visaggio, "Mobile malware detection using op-code frequency histograms," in *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, vol. 4, pp. 27–38, IEEE, 2015.

[12] M. Damshenas, A. Dehghantanha, K.-K. R. Choo, and R. Mahmud, "M0droid: An android behavioral-based malware detection model," *Journal of Information Privacy and Security*, vol. 11, no. 3, pp. 141–157, 2015.

[13] F. Mercaldo, V. Nardone, A. Santone, and C. A. Visaggio, "Download malware? no, thanks: how formal methods can block update attacks," in *Proceedings of the 4th FME Workshop on Formal Methods in Software Engineering*, pp. 22–28, 2016.

[14] D. Gavriluţ, M. Cimpoeşu, D. Anton, and L. Ciortuz, "Malware detection using machine learning," in *2009 International Multiconference on Computer Science and Information Technology*, pp. 735–741, IEEE, 2009.

[15] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[16] H. V. Nath and B. M. Mehtre, "Static malware analysis using machine learning methods," in *International Conference on Security in Computer Networks and Distributed Systems*, pp. 440–450, Springer, 2014.

[17] V. M. Afonso, M. F. de Amorim, A. R. A. Grégio, G. B. Junquera, and P. L. de Geus, "Identifying android malware using dynamically obtained features," *Journal of Computer Virology and Hacking Techniques*, vol. 11, no. 1, pp. 9–17, 2015.

[18] S. Y. Yerima, S. Sezer, and I. Muttik, "Android malware detection: An eigenspace analysis approach," in *2015 Science and Information Conference (SAI)*, pp. 1236–1242, IEEE, 2015.

[19] N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided android malware classification," *Computers & Electrical Engineering*, vol. 61, pp. 266–274, 2017.

[20] J. Sahs and L. Khan, "A machine learning approach to android malware detection," in *2012 European Intelligence and Security Informatics Conference*, pp. 141–147, IEEE, 2012.

[21] D. Li, Z. Wang, and Y. Xue, "Fine-grained android malware detection based on deep learning," in *2018 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–2, IEEE, 2018.

[22] J. Booz, J. McGiff, W. G. Hatcher, W. Yu, J. Nguyen, and C. Lu, "Tuning deep learning performance for android malware detection," in *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 140–145, IEEE, 2018.

[23] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *European symposium on research in computer security*, pp. 62–79, Springer, 2017.

[24] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "{TESSERACT}: Eliminating experimental bias in malware classification across space and time," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 729–746, 2019.

[25] A. Naway and Y. Li, "Using deep neural network for android malware detection," *arXiv preprint arXiv:1904.00736*, 2019.

[26] Y. Ye, S. Hou, L. Chen, J. Lei, W. Wan, J. Wang, Q. Xiong, and F. Shao, "Aidroid: When heterogeneous information network marries deep neural network for real-time android malware detection," *arXiv preprint arXiv:1811.01027*, 2018.

[27] Z. Xu, K. Ren, S. Qin, and F. Craciun, "Cdgdroid: Android malware detection based on deep learning using cfg and dfg," in *International Conference on Formal Engineering Methods*, pp. 177–193, Springer, 2018.

[28] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "Maldozer: Automatic framework for android malware detection using deep learning," *Digital Investigation*, vol. 24, pp. S48–S59,

2018.

[29] A. Hota and P. Irolla, "Deep neural networks for android malware detection.," in *ICISSP*, pp. 657–663, 2019.

[30] M. Odusami, O. Abayomi-Alli, S. Misra, O. Shobayo, R. Damasevicius, and R. Maskeliunas, "Android malware detection: A survey," in *International conference on applied informatics*, pp. 255–266, Springer, 2018.

[31] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, "A survey of android malware detection with deep neural models," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–36, 2020.

[32] N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé, *et al.*, "Deep android malware detection," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pp. 301–308, ACM, 2017.

[33] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, PMLR, 2014.

[34] C. Hasegawa and H. Iyatomi, "One-dimensional convolutional neural networks for android malware detection," in *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*, pp. 99–102, IEEE, 2018.

[35] "Download apk android apps and games — appsapk." https://www.appsapk.com/. (Accessed on 09/05/2019).

[36] "Apkpure apk を通じてオンラインで apk をダウンロードする — apkpure 公式サイト." https://apkpure.com/jp/. (Accessed on 09/05/2019).

[37] "Virustotal." https://www.virustotal.com/. (Accessed on 09/05/2019).

[38] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep ground truth analysis of current android malware," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 252–276, Springer, 2017.

[39] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket.," in *Ndss*, vol. 14, pp. 23–26, 2014.

[40] S. Michael, E. Florian, S. Thomas, C. F. Felix, and J. Hoffmann, "Mobilesandbox: Looking deeper into android applications," in *Proceedings of the 28th International ACM Symposium on Applied Computing (SAC)*, 2013.

[41] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," in *2018 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–7, IEEE, 2018.