5ZA-07

未展開APKファイルを利用したAndroidマルウェアの検知

山崎 一希 † 彌冨 仁 †
† 法政大学 理工学部 応用情報工学科

概 要

Androidマルウェアへの対策は急務となっている．近年の深層学習技術を用いて未展開APKファイルを解析することで, 高速かつ高精度の検疫システムがこれまで提案されているが，システムが本質的にマルウェア特徴をとらえているか疑問が残されていた．本報ではAndroidマルウェアの特徴が現れやすく，静的解析で頻繁に利用されるDEXファイルとマニフェストファイルを未展開APKファイルから抽出し，1次元畳み込みニューラルネットワーク (1-D CNN) で解析することで従来システムの有効性や解釈可能性を議論した．

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]. Androidマルウェアへの対策は急務であり，近年では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 [2, 3, 4, 5].

様々な機械学習手法の中でも畳み込みニューラルネットワーク (CNN) を利用した手法が多く提案されており，Mclaughlin et al.[4] 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CNNで識別する手法を提案し，高い精度を報告している．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.[5] 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．しかしながらこの研究では高い精度を達成しているが，過学習が疑われ，また解析位置が固定されているためマルウェアの特徴の本質をとらえているか疑問が残る．

本報では入力を静的解析で頻繁に利用されているDEXファイルとマニフェストファイルを入力とすることで説明可能性や精度の向上を図るとともに，学習データより新しい本質的に異なるデータセットによる評価を行い従来の研究との比較，議論を行った．

表 1: 入力が1,024バイトのときのモデルの構成

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

2 手法

2.1 未展開 APK ファイルからの特定領域の抽出

APKファイルの圧縮アルゴリズムはZIPと同様であり，各ファイルがそれぞれの名前と特定のフッターで囲まれている．この特徴を利用して未展開APKファイルからDEXファイル，マニフェストファイルに対応するビット列を抽出した．

2.2 前処理

未展開APKファイルから抽出したDEXファイルとマニフェストファイルのデータの先頭1,024，2,048，4,096，8,192バイトを切り出して入力とした．またこれらのサイズより小さいデータに対してはゼロパディングを行った．

2.3 モデル

実験で利用するモデルはHasegawa et al.[5] 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．入力を1,024バイトとしたときのモデルの構成を表1に示す．

Android malware detection without decompressing APK files
Kazuki YAMAZAKI, Hitoshi IYATOMI
Hosei University,184-8584, Koganei, Tokyo, Japan
kazuki.yamazaki.3m@stu., iyatomi@hosei.ac.jp

3 実験

3.1 データセット

学習に用いるデータセットはマルウェア，グッドウェアともに Hasegawa et al.[5] 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．具体的にはマルウェアとして AMD[6] データセットを利用した．このデータセットには合計 24,553 体の様々な種類のマルウェアが存在しており，そこからランダムに抽出した 5,000 体を学習に用いるマルウェアのデータセットとした．学習に用いるグッドウェアは長谷川らの研究で収集されたデータを利用した．その詳細は Appsapk[1] と Apkpure[2] 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[3]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 5,000 体である．なお用意したマルウェア 5,000 体とグッドウェア 5,000 体のうちそれぞれ 4,500 体ずつを教師データに，500 体ずつを検証データに使用した．

評価に用いる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 [7] 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．

3.2 評価

提案手法の検証にあたり，DEX ファイル，マニフェストファイルそれぞれを入力としたものと，ベースラインである Hasegawa et al. で最も高い精度が報告されている未展開 APK ファイルから末尾 1,024 バイトを切り出したものの比較を行った．評価指標として検証データと評価データそれぞれに対して f1-score を用いた．

4 結果と考察

各条件下でのモデルの識別結果を表 2 に示す．先行研究を含め，同じデータセット内での評価となる検証データに対する識別結果に比べ，未知の評価データに対する識別結果は大きく下がっていることから，いずれのモデルも学習データセットへの過学習しており，本質的な精度が得られていないことが確認できた．

また，未展開 APK ファイルからデータを切り出した場合でも先行研究同様，現時点では精度が低く，また入力サイズを変えても精度があまり変化しないことからマルウエアの本質的な特徴を捉えられていないといえる．一方で，マニフェストファイルの検証データに対する識別能が他より大きく低くなっていることから，何かしらの理由で学習そのものがしにくいことが示唆される．この差を踏まえ，本質的な特徴の所在の探索および，過学習抑制の観点から，今後この違いをより検証し精度向上を検討していく．

表 2: 各入力に対する識別結果

ファイル	バイト	val F1[%]	test F1[%]
ベースライン	1024	95.9	**62.2**
DEX	1024	95.2	59.2
	2048	95.1	56.1
	4096	94.8	54.0
	8192	**96.1**	56.8
マニフェスト	1024	84.8	56.5
	2048	85.6	56.7
	4096	83.0	51.3
	8192	86.9	50.7

5 おわりに

本報では未展開 APK ファイルからセキュリティに関連が深いと考えられる部の解析を 1 D-CNN で行ったが，現時点ではマルウェアの本質的な特徴を捉えられていなかった．今後さらなる検討を行い本質的に効果的なシステムの開発を目指す．

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] 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.

[3] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, ““andromaly”: a behavioral malware detection framework for android devices,” *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 161–190, 2012.

[4] 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*. ACM, 2017, pp. 301–308.

[5] 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)*. IEEE, 2018, pp. 99–102.

[6] 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*. Springer, 2017, pp. 252–276.

[7] 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)*. IEEE, 2018, pp. 1–7.

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

[2] 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)

[3] “Virustotal,” https://www.virustotal.com/gui/home/upload, (Accessed on 09/05/2019)