Computer Security Symposium 2021
26 - 29 October 2021

Androidマルウェア分類器に対するパッキングを用いた効果的な回避攻撃

古川 和祈[1,a)]　畑田 充弘[2]　吉浦 裕[3]　市野 将嗣[1]

概要：スマートフォンをターゲットとしたマルウェアが増加していることから，未知のマルウェアを検知できるように，機械学習を用いた検知手法が提案されるようになった．他方で近年，機械学習に対する攻撃手法である回避攻撃が提案されている．回避攻撃は，あるクラスに分類される入力に対して摂動を加えることで，別のクラスに誤分類するように仕向ける攻撃である．マルウェア検知に対する回避攻撃が実現すると，未知マルウェアの検知が困難となる脅威が生まれる．本稿では，既存の特徴量改変型回避攻撃手法に対して汎用的に組み込み可能なパッキングを用いた効果的な回避攻撃手法を提案する．パッキングを用いることで特徴量隠蔽と実行可能性検証可能な細工済みマルウェアの構築を実現した．Pierazzi らの勾配型回避攻撃手法に対して提案手法を組み込み，最先端の Android マルウェア分類器である DREBIN と回避攻撃に対して堅牢である Sec-SVM 及び Random Forest に対する回避攻撃の実験を行い，全ての分類器に対して 90%以上の回避攻撃成功率を実現したことが確認できた．

キーワード：回避攻撃，パッカー，機械学習

Effective evasion attack using packing against Android malware classifiers

Kazuki Furukawa[1,a)]　Mitsuhiro Hatada[2]　Hiroshi Yoshiura[3]　Masatsugu Ichino[1]

Abstract: As the number of malware targeting smartphones is increasing, detection methods based on machine learning have been proposed to detect unknown malware. On the other hand, in recent years, evasive attacks have been proposed as an attack method against machine learning. An evasion attack is an attack that perturbs inputs that are classified into one class so that they are misclassified into another class. The realization of evasive attacks against malware detection creates a threat that makes it difficult to detect unknown malware. In this paper, we propose an effective evasion attack using packing, which can be universally incorporated into existing feature-modification evasion attacks. We incorporate the proposed method into Pierazzi et al.'s gradient-based evasion attack method, and conduct experiments on evasion attacks against DREBIN, Sec-SVM, and Random Forest. We confirmed that the proposed method achieves more than 90% success rate of evasion attacks against all classifiers.

Keywords: Evasion Attack, Packer, Machine Learning

1. はじめに

スマートフォンの普及に伴って，スマートフォンを対象としたマルウェアが増加している．増加するマルウェアへの対策として，マルウェア検知の必要性が高まっている．従来のシグネチャ型検知の欠点である未知マルウェア検知

[1] 電気通信大学
The University of Electro-Communications
[2] 日本電信電話株式会社
Nippon Telegraph and Telephone Corporation
[3] 京都橘大学
Kyoto Tachibana University
[a)] k-furukawa@uec.ac.jp

において，機械学習を用いたマルウェア検知が広く使われるようになった．

他方で近年，機械学習に対する攻撃手法が多く提案されている．機械学習に対する攻撃手法の 1 つに回避攻撃というものが存在する．回避攻撃は，あるクラスに分類される入力に対して摂動を加えることで，別のクラスに誤分類されるようにする攻撃である．機械学習を用いたマルウェア検知に対して回避攻撃を適用する研究も提案されるようになった．

マルウェア検知に対して回避攻撃を適用すると，悪性ファイルとして検知されたマルウェアを良性クラスに誤分類するように仕向けることが可能となる．よって，実際にマルウェア検知に対する回避攻撃が実現すると，機械学習を用いることで検知することが出来ていた未知マルウェアの検知が困難となる脅威が存在する．

近年の Android マルウェア対策研究の分野における回避攻撃において分類器のパラメータや評価関数の勾配に基づいて摂動をマルウェアに加える手法（勾配型回避攻撃）や複数の分類器の間に存在する Transferability という性質を用いた回避攻撃手法（ブラックボックス型回避攻撃）といった特徴量改変型回避攻撃が検討されてきた．特徴量改変型回避攻撃は勾配型回避攻撃やブラックボックス型回避攻撃といった特徴量に細工を加えることで回避を実現する回避攻撃手法を指す．またこれらの手法においてマルウェアの実行可能性は特徴量を追加する細工のみを用いることで保証されていた．

しかし，これまでの研究ではパッキングを特徴量改変型回避攻撃に組み込んだ攻撃手法は検討されてこなかった．パッキングを Android マルウェアに適用することで当該マルウェアの実行コードに関する特徴量を隠蔽することで擬似的に特徴量を削除することが可能となる．また，細工を加えたパック済みマルウェアの実行可能性を検証する場合，スタブコードの実行可能性検証のみでマルウェアの実行可能性を保証することが可能となる．そこで，本稿では既存の特徴量改変型回避攻撃手法に対して汎用的に組み込み可能な，パッキングを用いた効果的な回避攻撃手法を提案する．

本研究の貢献をまとめると，以下の通りである．

- パッキングを用いた特徴量隠蔽による効果的な回避攻撃の実現
- パッキングを用いることによる回避攻撃適用後の実行可能性の保証

本稿の構成を以下に示す．第 2 章は関連研究を紹介する．第 3 章は，提案手法について説明する．第 4 章は評価実験の手法および設定について説明する．第 5 章は，評価実験の結果を提示する．第 6 章では評価実験の結果をもとに提案手法に関する考察を行う．最後に第 7 章は本稿のまとめを述べる．

2. 関連研究

2.1 回避攻撃

回避攻撃とは，機械学習を用いた分類器に対する攻撃の一種である．回避攻撃はあるクラスに分類される入力に対して摂動を与えることで，別のクラスに分類されるようにすることができる．これをマルウェアの分類器に対して行うことで，あるマルウェアを良性ソフトウェアとして誤認識させることが可能となる．

回避攻撃の分野における最初の重要な研究は 2004 年に遡り，Dalvi ら [1] と Lowd・Meek[2][3] がスパムフィルタリングに対する回避攻撃の研究を行った．スパムメールの内容にいくつかの細工を加えることで，スパムメッセージの可読性に大きな影響を与えることなく，線形分類器を用いたスパムフィルタリングを回避することができることが示された．その後，Szegedy ら [4] は画像分類を行う Deep Neural Network（以下 DNN）に対して人間が認識できないような小さな摂動を適切に選び，入力に与えると回避攻撃が実現できることを示した．これに端を発して DNN を用いた分類器に対する回避攻撃の研究が広く取り組まれるようになった．

近年のマルウェア対策研究の分野における回避攻撃の研究では Android, PE, PDF を対象とした研究が主に取り組まれている [5][6][7].

2.2 Android マルウェア分類器に対する回避攻撃

Arp ら [8] は DREBIN と呼ばれる二値の静的特徴量を用いた Android マルウェア分類器を提案した．この論文内では分類器, 特徴量及びデータセットが同時に提案及び公開されており, それぞれが独立して様々な論文で用いられているため, 以降で特に区別する際は DREBIN 分類器, DREBIN 特徴量，DREBIN データセットと記述する．DREBIN 分類器では分類アルゴリズムとして線形 SVM が利用されている．Pendlebury ら [9] の研究によると，DREBIN 分類器は頻繁に再学習を行うことで，Android マルウェアの検知において最先端の性能を持つことがわかっている．このため，Android マルウェア対策の分野における回避攻撃の研究では DREBIN に対する攻撃が多くの研究で検討されてきた [12][13][14].

Grosse ら [5] は，DREBIN 特徴量を用いて学習させた DNN を用いた Android マルウェア分類器に対する回避攻撃手法を提案した．この研究では特徴量に対する変更を 0 から 1, つまりマルウェアの機能が削減されない変更に制限することで，マルウェアの実行可能性を保証している．しかし，摂動を加えた後の特徴量と対応するマルウェアに対する細工が存在することは保証されておらず，どのようにその細工をマルウェアに適用するのかは検討されていない．Shahpasand ら [10] は，DREBIN データセットで学習した

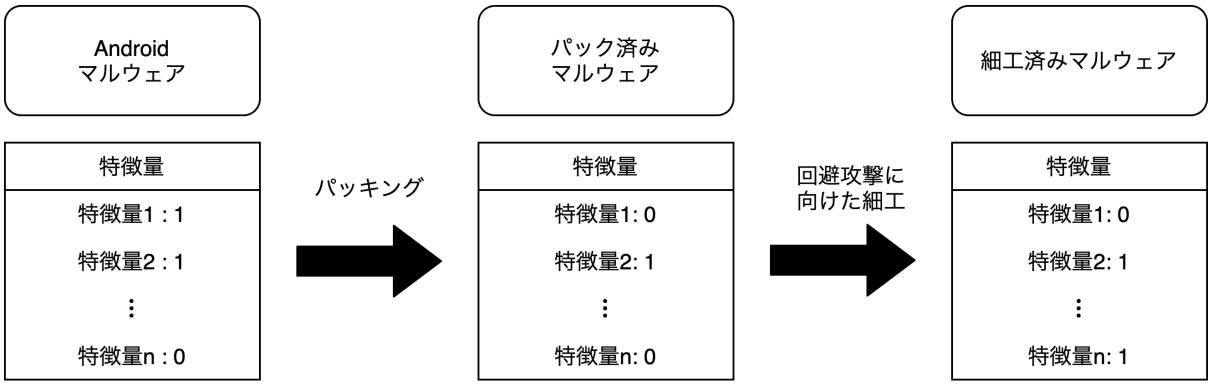

図 1 提案手法の概要
Fig. 1 Overview of proposed method.

Neural Networks，Random Forest，Logistic Regression の分類器及び DREBIN 分類器に対する GAN を用いた回避攻撃を実現した．この研究では追加する特徴量数に上限を設けることで実行可能性を保証していると主張しているが，Grosse らと同様に摂動を加えた後の特徴量と対応するマルウェアに対する細工が存在することは保証されていない．

Demontis ら [11] は，DREBIN 分類器で用いられている線形 SVM の学習アルゴリズムを改良し回避攻撃に対してより堅牢な分類器である Sec-SVM を提案した．Sec-SVM は，特徴量の重みが均等に分散するように線形分類器を学習させる．特徴量の重みが均一であるほど，回避攻撃において特徴量の変更数がより多く必要となる．Melis ら [12] は，DREBIN データセット及び TESSERACT データセット [9] で学習した DREBIN 分類器，Sec-SVM，SVM（RBF カーネル），Logistic Regression 及び Ridge Regression に対して Projected Gradient Descent を用いた回避攻撃を行い，5 種類の分類器の中で Sec-SVM が最も回避攻撃に対して堅牢だということを示した．Pierazzi ら [13] は Opaque Predicates を用いたコード挿入による回避攻撃手法を提案し，DREBIN 分類器及び Sec-SVM に対して回避攻撃を行った．Opaque Predicates を用いることでマルウェアの実行可能性への影響を減らした細工の埋め込みを可能としている．

Sec-SVM の他に Random Forest を用いた Android マルウェア分類器も回避攻撃に対して堅牢であることが先行研究で示されている．Abaid ら [14] は，DREBIN データセットで学習した線形分類器（線形 SVM，Logistic Regression）と非線形分類器（Random Forest，Neural Networks）に対して回避攻撃を行った．分類器の完全な知識を攻撃者が持っているとした場合に，線形分類器の検知率は 100% から 0%に低下し，分類器の知識を攻撃者が一切持っていないとした場合でも 12%に低下した．一方，非線形分類器はこれらの攻撃に対してより耐性があることが示された．Shahpasand ら [15] は，DREBIN データセットで学習した Random Forest，線形 SVM，SVM（RBF カーネル），Multi-Layer Perceptron，Logistic Regression の分類器に対して Jacobian-based Saliency Map Attack，Fast Gradient Sign Method，GAN を用いた回避攻撃を行った．この実験において Random Forest は全ての回避攻撃に対して他の分類器よりも堅牢であることが示された．

2.3 本研究の位置づけ

本研究は既存の特徴量改変型回避攻撃手法に対して汎用的に組み込み可能なパッキングを用いた回避攻撃手法を提案する．

関連研究では実行可能性を保証するために特徴量に対する変更が 0 から 1 というマルウェアの機能が削減されない変更に制限されていた．そこで本研究ではパッキングを用いて特徴量を 1 から 0 への変更を行えるようにすることで，効果的な回避攻撃を実現する．

また多くの関連研究では特徴量空間での操作を制限することで実行可能性を保証している．しかし，従来の手法では細工されたマルウェアの実行可能性を検証することは一般的なソフトウェアに対するテスト以上に困難である．特にマルウェアのソースコードを持たないマルウェア利用者が細工を加えた場合，テストのカバレッジを確保することは困難であり，テストを行う際のオラクルはクラッシュ判定だけでは不十分である．そこで本研究では，パッキングを用いることで細工後のマルウェアの実行可能性検証を可能とする．

3. 提案手法

提案手法の概要を**図 1** に示す．提案手法ではパッキングを用いることで特徴量隠蔽を行い，効果的な回避攻撃を実現する．提案手法は静的特徴量を用いた分類器に対して有効である．

3.1 パッキング

Android アプリケーションは APK フォーマットというファイル形式でスマートフォンにインストールされる．

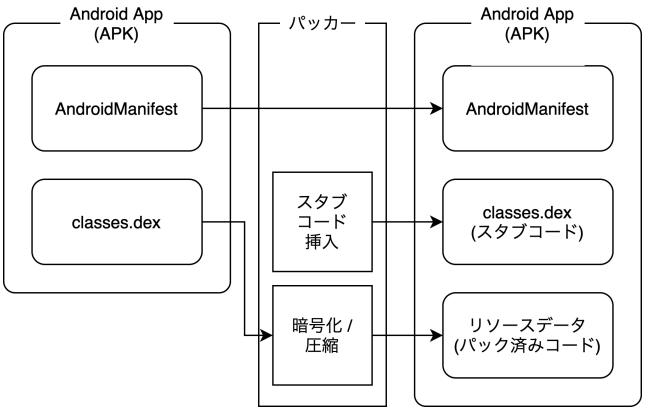

図 2 パッキングの概念図
Fig. 2 Conceptual diagram of packing.

APK の中でもプログラムの処理に大きく関わるものは AndroidManifest と classes.dex である．AndroidManifest には主にパーミッション，インテントフィルター，アプリのコンポーネントやアプリが必要とするハードウェア機能・ソフトウェア機能に関する情報が含まれている．classes.dex は Java クラスファイルを圧縮したもので，基本的に実行される Java バイトコード全てがこのファイルに含まれている．

提案手法で用いるパッキングの概念図を**図 2** に示す．AndroidManifest の情報を実行時に書き換えることはできないため，パッキング後も殆ど同一の内容のものが用いられる．一方，classes.dex は動的にメモリにロードして実行することが可能である．そこでオリジナルの classes.dex は暗号化や圧縮といった可逆変換を加えられた後にリソースデータとして保存される．そして新しい classes.dex としてスタブコードがパック後の APK に挿入される．スタブコードとは，暗号化または圧縮処理がされたオリジナルのコードを動的に展開して処理をオリジナルのコードの適切な場所に遷移させる機能を持ったコードを指す．パック済みの APK が実行されるとスタブコードがリソースデータに保存されているコードを復元して実行することでオリジナルのコードの機能を担保する．

3.2 回避攻撃手法

提案手法は回避攻撃の細工をマルウェアに適用する前処理としてパッキングを組み込むため，静的特徴量を用いた回避攻撃手法に対して新たな制約を加えること無く適用することが可能である．勾配型回避攻撃やブラックボックス型回避攻撃といった既存の回避攻撃手法だけでなく，今後提案される回避攻撃手法においても汎用的に提案手法が利用できる．

3.3 特徴量隠蔽

パッキングをマルウェアに施すことで，静的特徴量のうち実行コードに関する特徴量が隠蔽され，スタブコードの特徴量に置き換わる．パッキングによって変化する特徴量は静的特徴量のみであり，オリジナルの実行コードに対して処理を加えていないため API コールトレースなどの動的特徴量に対して細工が加わることは無い．

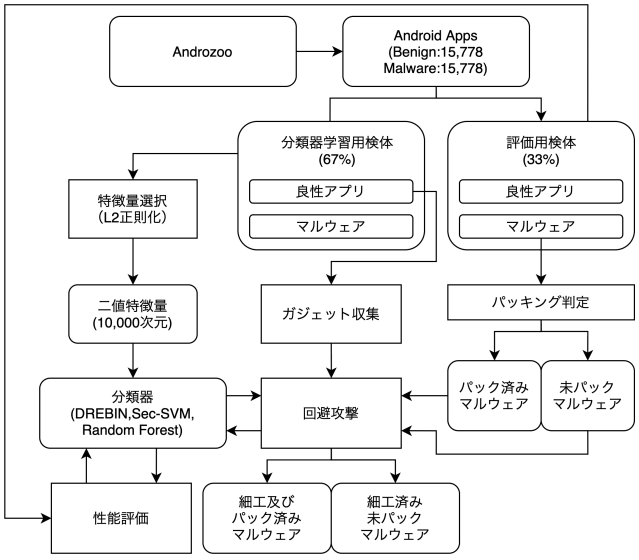

図 3 評価実験の概要
Fig. 3 Overview of experiment.

3.4 実行可能性の検証

提案手法においてマルウェアの動作に影響する細工はスタブコードにのみ加えられるため，スタブコードの機能性を検証することで細工及びパック済みマルウェアの実行可能性を保証することが可能である．細工及びパックがされたマルウェアを実行し，スタブコードによる展開が終わり正常にオリジナルのコードの処理に遷移することを確認することで，マルウェアの実行可能性を検証したとみなせる．このとき，マルウェアの実行可能性はパッカーの保証に依存している．例えば，パッカーがサポートしていない Android バージョンや環境ではマルウェアの実行可能性は保証されない．

4. 評価実験

提案手法の回避攻撃性能を評価するため，評価実験を行った．評価実験の概要を**図 3** に示す．本実験では Pierazzi ら [13] の手法に対し提案手法を組み込むことで提案手法の性能を評価する．Pierazzi らの手法は DREBIN 及び Sec-SVM に対して最先端の回避攻撃性能を持つことが示されていること，特徴量空間での攻撃だけでなく Opaque Predicates を用いることで特徴量の変更に対応する細工を実際のマルウェアに対して適用することが可能であることから本実験に適当であるといえる．

4.1 回避攻撃アルゴリズム

Pierazzi らの回避攻撃アルゴリズムの概要を**図 4** に示す．分類器の内部パラメーターである各特徴量の重みに基づいて良性クラスに近づける特徴量の上位 n_f 個のそれぞれについて n_d 個のガジェット候補を良性アプリから抽出する．抽出したガジェットには前後の関連するコードが含まれているため，良性クラスに近づける特徴量の他に，マルウェアクラスに近づける特徴量が含まれている場合がある．そこで分類器の特徴量の重みに基づいてガジェット全体としてどの程度良性クラスに寄りやすいか計算する．最後に収集したガジェットの中で最も良性クラスに寄りやすいガジェットから順番にマルウェアに対して追加していく．分類器がマルウェアを良性アプリとして分類する，または分類スコアが目標とした値に到達した時点でガジェットの追加を止める．

ガジェット候補を収集する際の特徴量数 n_f は増やすことで回避攻撃の精度を高めることが可能であるが，実験においてガジェットの収集は特に多くの時間を必要とするプロセスでもある．そこで本実験では各標的モデルの回避攻撃に対する堅牢性に応じてガジェット候補を収集する際の特徴量数 n_f を設定し，DREBIN 分類器に対する回避攻撃実験では $n_f = 100$，Sec-SVM に対する回避攻撃実験では $n_f = 100$ または $n_f = 500$，Random Forest に対する回避攻撃実験では $n_f = 1000$ とした．各特徴量について収集するガジェット候補数 n_d については大きく回避攻撃性能に影響しないため，先行研究と同じく $n_d = 5$ とした．

Pierazzi らの回避攻撃アルゴリズムにおいて終了条件を確信度に応じて変更しており，先行研究では低確信度と高確信度の 2 つの条件で実験が行われた．先行研究では負のスコアが良性クラスとして扱われている．低確信度の条件ではスコアが 0 を下回る，つまり分類器が良性として分類した時点でガジェットの追加は終了となる．一方で，高確信度では良性アプリの 25%の負スコア以下つまり四分位範囲内になった時点で処理が終了する．本実験では高確信度の終了条件で実験を行った．また，この回避攻撃アルゴリズムは DREBIN 分類器及び Sec-SVM に対する回避攻撃を想定したものであるため，本実験で Random Forest に対する回避攻撃を行う際に実装の一部変更を行った．オリジナルの実装では特徴量の重みとして偏回帰係数を用いているが，ジニ不純度に基づいた特徴量の重要度を代わりに用いた．

4.2 標的モデル

4.2.1 特徴量

Pierazzi らの実験と同じく DREBIN 特徴量を用いた．DREBIN 特徴量は二値の静的特徴量であり，DREBIN 特徴量は AndroidManifest から取得する 4 種類の特徴量と classes.dex が取得する 4 種類の特徴量で構成されている．

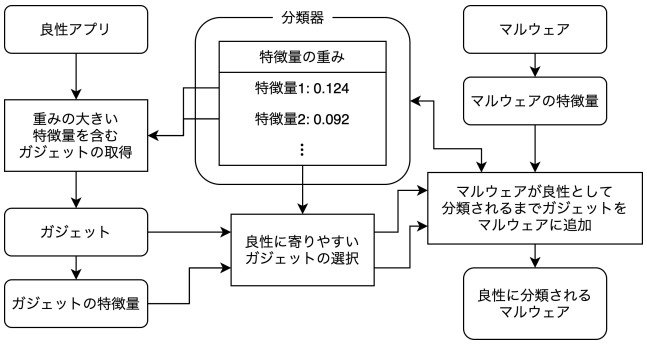

図 4 Pierazzi らの回避攻撃手法の概要
Fig. 4 Overview of Pierazzi et al.'s evasion attack method.

表 1 DREBIN 特徴量
Table 1 DREBIN Features.

AndroidManifest	classes.dex
S_1: Hardware components	S_5: Restricted API calls
S_2: Requested permissions	S_6: Used permissions
S_3: App components	S_7: Suspicious API calls
S_4: Filtered intents	S_8: Network addresses

表 1 に DREBIN 特徴量の特徴量セットを示す．

4.2.2 特徴量選択

Pierazzi らの実験では PyTorch はスパースベクトルをサポートしていないため，計算効率を上げるために特徴選択が行われた．本実験でも同様に，L2 正則化（Ridge）を用いて特徴量を上位 10,000 個に絞り込んだ．

4.2.3 分類アルゴリズム

標的モデルの分類アルゴリズムとして DREBIN 分類器，Sec-SVM 及び Random Forest を用いることとした．DREBIN 分類器で用いられている線形 SVM のコストパラメータは $C = 1$，Sec-SVM の特徴量重みの最大値は $k = 0.25$ とした．これらは Pierazzi らの実験で用いられたパラメータの値と同一である．Random Forest の決定木の個数を 100，決定木の最大深度を 50 とした．

DREBIN 分類器及び Random Forest は Scikit-Learn の実装を用いた．Sec-SVM は Pierazzi らが研究者に向けて公開したコードに含まれる実装を用いた．

4.3 データセット

評価実験で用いるデータセットには Pierazzi らの実験で使われたものと同一の検体を用いた．これらの検体は Androzoo[16] から収集した．Androzoo は Google Play Store 及びサードパーティストアからクロールされたタイムスタンプ付きの Android アプリと VirusTotal のサマリーレポートを含む大規模データセットである．Pierazzi らの実験では Androzoo から良性アプリを 135,859 検体，マルウェアを 15,778 検体収集して分類器の学習及び回避攻撃の評価に用いられていた．しかし，良性アプリとマルウェアの数に大きな偏りが存在すると分類器の適切な学習が行われない

可能性があることから，本実験では良性アプリとマルウェアをそれぞれ 15,778 検体として実験を行った．135,859 検体の良性アプリから 15,778 検体をサンプリングする際には無作為抽出を用いた．

4.3.1 データセットの分割

Pierazzi らの実験と同様に分類器学習用検体数が 67%，評価用検体数が 33%となるようにランダムに分割した．

4.3.2 評価用検体

回避攻撃の評価にはマルウェアでかつ標的モデルがマルウェアとして分類する検体が必要である．標的モデルの分類アルゴリズムが異なり，データセットがランダム変更されるため評価実験を行うたびに回避攻撃の性能評価に用いることが可能な検体数は変化するが，平均で約 4966 検体，最低でも 4804 検体が利用可能だった．これは評価用検体に含まれるマルウェアを平均で約 95%，最低でも約 92%を評価に用いることが出来ているといえる．

4.3.3 パック済みマルウェア

既存の Android アプリケーション向けパッカーは主にオンラインサービスとして提供されるものが殆どである．そのため，未パックマルウェアをパックするためには第三者のオンラインサービスへアップロードすることが必要となるが，これは研究倫理上の問題が存在する．そこで本研究では評価用検体に元々含まれていたパック済みマルウェアを用いて提案手法の評価を行った．パック済みマルウェアの抽出には APKiD を用いた．APKiD は APK の表層解析が可能なオープンソースソフトウェアであり，APKiD はシグネチャーマッチによってある Android アプリケーションが特定のパッカーファミリーに属するかどうか検知することが可能である．データセット全体のマルウェアは 15,778 検体であり，この内パック済みの検体が 6,217 検体，未パックの検体が 9,561 検体であった．パック済みマルウェアに対して Pierazzi らの回避攻撃を適用することで，細工されたパック済みマルウェアが出力される．

4.3.4 未パックマルウェア

未パックマルウェアは評価用検体のマルウェアに対して APKiD を用いて，既存のパッカーが一件も用いられていないと判定された検体を未パックマルウェアとした．未パックマルウェアに対する回避攻撃はパック済みマルウェアと同様の回避攻撃処理が行われ，最終的に細工済みの未パックマルウェアが出力される．

5. 実験結果

第 4.3 節で示したデータセットを用いて学習した DREBIN 分類器，Sec-SVM 及び Random Forest の性能評価実験を行った．更にそれぞれの分類器に対して回避攻撃を行い，提案手法の評価を行った．

表 2 標的モデルの精度
Table 2 Performance of target model.

	Precision	Recall	F1	ROC AUC
DREBIN	0.951	0.965	0.958	0.981
Sec-SVM	0.940	0.928	0.934	0.973
Random Forest	0.963	0.964	0.964	0.991

表 3 回避攻撃の成功率
Table 3 Attack success rate of the evasion attack.

	パック済み	未パック
DREBIN ($n_f = 100$)	1.00	1.00
Sec-SVM ($n_f = 100$)	0.901	0.685
Sec-SVM ($n_f = 500$)	1.00	1.00
Random Forest ($n_f = 1000$)	0.901	0.665

表 4 特徴量変更 1 つあたりのスコア変動
Table 4 Score changes per distortion.

	パック済み	未パック
DREBIN ($n_f = 100$)	0.199	0.244
Sec-SVM ($n_f = 100$)	0.0300	0.0292
Sec-SVM ($n_f = 500$)	0.0388	0.0266
Random Forest ($n_f = 1000$)	0.00315	0.00891

5.1 標的モデルの精度

DREBIN 分類器，Sec-SVM 及び Random Forest の性能を**表 2** に示す．全ての分類器が ROC-AUC において 0.97 以上のスコアを示しており，Android マルウェア分類器として充分な性能を持っていると言える．

5.2 回避攻撃成功率

回避攻撃成功率は細工対象マルウェア数のうち，どの程度の検体について良性ファイルとして分類されるような細工を見つけられたのかを示す．回避攻撃成功率 = 良性に分類された検体数/細工対象検体数 で求めることができる．

DREBIN 分類器，Sec-SVM 及び Random Forest に対する回避攻撃の成功率を**表 3** に示す．表 3 において，DREBIN ($n_f = 100$), Sec-SVM ($n_f = 100$) 及び Sec-SVM ($n_f = 500$) は 5 回の実験，Random Forest は 2 回の実験を行った際の回避攻撃の成功率の平均を示している．

特徴量変更 1 つあたりのスコア変動を**表 4** に示す．表 3 と同様に DREBIN 及び Sec-SVM は 5 回の実験，Random Forest は 2 回の実験を行った際の平均を示している．ただし，Random Forest に関しては他の分類器とは異なり，Random Forest 内の木の予測クラス確率の平均値をスコアとして用いているため，他の分類器の結果と直接比較することはできない．

同一のデータセット分割の条件の元で，n_f を 100 から 500 まで 100 ずつ増やした際の Sec-SVM に対する攻撃成功率を**表 5** に示す．

表 5　回避攻撃の成功率の推移

Table 5　Transition of attack success rate of the evasion attack.

	パック済み	未パック
Sec-SVM ($n_f = 100$)	0.956	0.644
Sec-SVM ($n_f = 200$)	1.00	0.909
Sec-SVM ($n_f = 300$)	1.00	0.993
Sec-SVM ($n_f = 400$)	1.00	0.997
Sec-SVM ($n_f = 500$)	1.00	1.00

6. 考察

6.1 提案手法の回避攻撃成功率

表 3 より，パック済みマルウェアに対して細工を加えた場合，全ての分類器に対して未パックマルウェアと同等もしくはそれよりも高い成功率で回避攻撃が可能であることが示された．

パック済みマルウェアに対して細工を加えた場合に回避攻撃の成功率が高い理由として，パッキングによってマルウェアの特徴量が一部擬似的に削除された際に，擬似的に削除された特徴量の中に悪意のあるコードが持っている傾向にある特徴量が含まれていたことが予想できる．このとき，特徴量の追加のみが行われる未パックマルウェアに比べてマルウェアらしい特徴量が削除されたパック済みマルウェアはより容易に良性アプリの特徴量に近づけられるといえる．

一方，表 4 によると特徴量変更 1 つあたりのスコアに対する影響はパック済みマルウェアが必ず大きいわけではない．パック済みマルウェアの方が回避攻撃成功率が高い Sec-SVM($n_f = 100$) と Random Forest($n_f = 1000$) に着目すると，特徴量変更 1 つあたりのスコアに対する影響は，Sec-SVM($n_f = 100$) ではパック済みのほうが大きいが，Random Forest($n_f = 1000$) では未パックのほうが大きい．

この原因としてまずパック済みマルウェアよりも未パックマルウェアの方が特徴量に 1 が多く含まれていることが考えられる．ガジェットには複数の特徴量が含まれている．あるガジェットをパック済み及び未パック済みマルウェアのそれぞれに埋め込んだ場合，パック済みマルウェアの特徴量は削減されているためガジェットの特徴量との差分が大きくなりやすいが，未パックマルウェアには 1 が多く含まれているためガジェットの特徴量との差分が小さくなりやすいと考えられる．

また，細工による特徴量変更以外に回避攻撃成功率に影響を与えている要素があることも考えられる．そこで，パック済みマルウェアの回避攻撃成功率と追加特徴量数のグラフを **図 5**，未パックマルウェアの回避攻撃成功率と追加特徴量数のグラフを**図 6** に示した．図 5 において，全ての分類器に対して 90%程度の回避攻撃成功率を達成しているが，図 6 では Random Forest と Sec-SVM($n_f = 100$) において 60%の付近で成功率の増加が止まっている．この原因としてまず分類器のスコアに影響を与えるガジェットが収集できていないことが想定され，パッキングによる特徴量削減によってパック済みマルウェアでは有用なガジェットであっても未パックマルウェアには影響を与えづらいと考えられる．この他に，マルウェアで参照しているクラスとガジェットで参照しているクラスに共通していることが考えられる．Pierazzi らの実験において細工対象のマルウェアと埋め込むガジェットに共通するクラスが存在すると適切に埋め込みが出来ないことから一部のガジェットの埋め込みが省略される．パック済みマルウェアにおいてはスタブコードから参照するクラスも減るため，より多くのガジェットを埋め込めると考えられる．

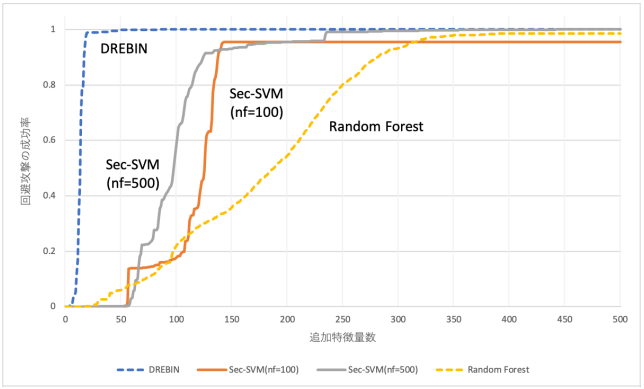

図 5　パック済みマルウェアの回避攻撃成功率と追加特徴量数

Fig. 5　Attack Success Rate and Number of Added Features of packed malwares.

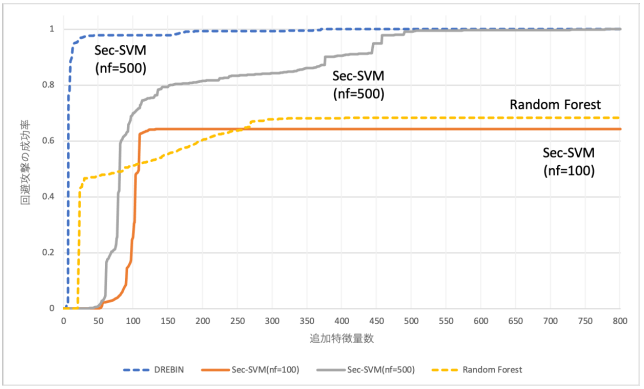

図 6　未パックマルウェアの回避攻撃成功率と追加特徴量数

Fig. 6　Attack Success Rate and Number of Added Features of not packed malwares.

6.2 ガジェット収集に用いる特徴量数 n_f と性能の関係

表 5 より，ガジェット収集に用いる特徴量数 n_f が増えるほど，パック済みマルウェアと未パックマルウェアのどちらも回避攻撃成功率が上昇することが確認できた．これはガジェット収集に用いる特徴量数が増えることで回避攻撃に利用可能なガジェットが増加し，より良性クラスに近

づける必要がある検体を回避させることが可能となったと考えられる．

6.3 パッキングによる特徴量隠蔽

提案手法及び実験において，パッキングによって特徴量が削減されていることを前提としている．そこで実際にパッキングによって特徴量が隠蔽されるのかパッカーを用いて検証した．しかしマルウェアやデータセットに含まれるアプリケーションを第三者のサービスへアップロードすることは出来ないため，Google Play Store から良性アプリケーションであるノートアプリを取得しパッキング前後の特徴量の変化を調査した．パッキングには実験に用いたデータセットでも最も多く使われていた Qihoo360 が開発・提供している Jiagu Packer を用いた．パッキング前のアプリケーションに含まれていた特徴量は 155 個であり，パッキング後は 81 個まで減少していた．つまり 74 個の特徴量が削減されたことになる．このことから，検証はできないが今回のデータセットに含まれているパック済みマルウェアも同様に特徴量の削減が行われていたと考えられる．

6.4 パッキングによる実行可能性の検証

パッキングによって細工済みマルウェアの実行可能性を検証することが可能となる．細工後のマルウェアをエミュレータ上で動作させ，オリジナルのコードへ遷移することを確認することで実行可能性を検証することが可能である．従来手法であれば細工後のマルウェア全体を何らかのテスト手法を用いて検証する必要があるが，提案手法を用いることで検証する必要があるコードを削減することが可能である．

6.5 制限

パッキング処理において，パッキング後の APK に対してスタブコード及びそれに付随するデータが挿入される．パッキングによって特徴量が減ることは第 6.3 節において示したが，パッキングによる意図しない特徴量挿入が発生する可能性が存在する．

提案手法において，マルウェアの実行可能性はパッカーの保証に依存している．パッカーのサポート外の環境においてマルウェアの実行可能性は保証されない．

7. おわりに

本研究では，パッキングを用いることで特徴量隠蔽を行い，効果的に回避攻撃を実現する手法を提案した．評価実験では最先端の回避攻撃手法に提案手法を組み込むことで，DREBIN 分類器，Sec-SVM 及び Random Forest の全てに対して 90%以上の Android マルウェアの回避に成功した．今後の課題として，パック及び細工を行ったマルウェアの実行可能性を実験を通じて検証を行うことが挙げられる．

参考文献

[1] DALVI, Nilesh, et al. Adversarial classification. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. 2004. p. 99-108.

[2] LOWD, Daniel; MEEK, Christopher. Adversarial learning. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. 2005. p. 641-647.

[3] LOWD, Daniel; MEEK, Christopher. Good Word Attacks on Statistical Spam Filters. In: CEAS. 2005.

[4] SZEGEDY, Christian, et al. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[5] GROSSE, Kathrin, et al. Adversarial examples for malware detection. In: European symposium on research in computer security. Springer, Cham, 2017. p. 62-79.

[6] SUCIU, Octavian; COULL, Scott E.; JOHNS, Jeffrey. Exploring adversarial examples in malware detection. In: 2019 IEEE Security and Privacy Workshops (SPW). IEEE, 2019. p. 8-14.

[7] BIGGIO, Battista, et al. Evasion attacks against machine learning at test time. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, Berlin, Heidelberg, 2013. p. 387-402.

[8] ARP, Daniel, et al. Drebin: Effective and explainable detection of android malware in your pocket. In: Ndss. 2014. p. 23-26.

[9] PENDLEBURY, Feargus, et al. TESSERACT: Eliminating experimental bias in malware classification across space and time. In: 28th USENIX Security Symposium (USENIX Security 19). 2019. p. 729-746.

[10] SHAHPASAND, Maryam, et al. Adversarial attacks on mobile malware detection. In: 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile). IEEE, 2019. p. 17-20.

[11] DEMONTIS, Ambra, et al. Yes, machine learning can be more secure! a case study on android malware detection. IEEE Transactions on Dependable and Secure Computing, 2017, 16.4: 711-724.

[12] MELIS, Marco, et al. Do Gradient-based Explanations Tell Anything About Adversarial Robustness to Android Malware?. arXiv preprint arXiv:2005.01452, 2020.

[13] PIERAZZI, Fabio, et al. Intriguing properties of adversarial ml attacks in the problem space. In: 2020 IEEE Symposium on Security and Privacy (SP). IEEE, 2020. p. 1332-1349.

[14] ABAID, Zainab; KAAFAR, Mohamed Ali; JHA, Sanjay. Quantifying the impact of adversarial evasion attacks on machine learning based android malware classifiers. In: 2017 IEEE 16th international symposium on network computing and applications (NCA). IEEE, 2017. p. 1-10.

[15] SHAHPASAND, Maryam, et al. Feature-Based Adversarial Attacks Against Machine Learnt Mobile Malware Detectors. In: 2020 30th International Telecommunication Networks and Applications Conference (ITNAC). IEEE, 2020. p. 1-8.

[16] ALLIX, Kevin, et al. Androzoo: Collecting millions of android apps for the research community. In: 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR). IEEE, 2016. p. 468-471.