メタ情報を活用したAndroidアプリケーションのリスク分析手法に関する検討

高橋健志† 班 涛† 三村隆夫‡ 中尾康二†

†国立研究開発法人 情報通信研究機構
184-8795 東京都小金井市貫井北町 4-2-1
takeshi_takahashi@nict.go.jp

‡株式会社セキュアブレイン
102-0083 東京都千代田区麹町 2-6-7 麹町 RK ビル 4F

あらまし 近年、Android 端末でのセキュリティインシデントは急増しており、その対策として Android Package (APK) 分析技術が多数報告されているが、それらの分析精度は更なる向上が求められている。これらの技術の多くはプログラム分析に基づく検知を実施しているが、我々はこれに加えて APK のオンラインマーケットから取得できるメタ情報を用いることで、分析精度を向上する技術を検討している。本稿では特に、アプリケーション (以下、アプリ) カテゴリおよび説明文に基づく APK のリスク・マルウェア分析手法を提案する。提案手法は、Web から取得した APK のアプリカテゴリ情報や説明文を最大限活用している点に特徴があり、リスクを数値化すると同時に、マルウェア判定を実現する。また、同様のパラメータを活用した機械学習ベースのマルウェア検知手法を実装し、両方式の性能を比較する。評価実験では、従来方式と比較して提案方式の area under curve 値が向上していることを示し、提案方式の有効性、そして APK リスク分析へのメタ情報の有用性を考察する。

Studies on Risk Level Evaluation Schemes using APK Metadata

Takeshi Takahashi† Tao Ban† Takao Mimura‡ Koji Nakao†

†National Institute of Information and Communications Technology.
4-2-1 Nukui-Kitamachi, Koganei, Tokyo 184-8795, JAPAN
takeshi_takahashi@nict.go.kjp

‡SecureBrain Corporation
Kojimachi RK Building 4F, 2-6-7 Kojimachi, Chiyoda-ku, Tokyo 102-0083, JAPAN

Abstract The number of security incidents faced by Android users is growing, along with the surge in malware targeting Android terminals. To cope with that, assorted Android Package (APK) analysis techniques have been reported, but most of them focus on program analyses. Different from these approaches, we use APKs' metadata obtained from the online APK markets to improve the accuracy of risk detection. In this paper, we propose APK analysis schemes based on application categories and descriptions. We propose both risk quantification approach and two-class classification one and evaluate them. This paper verifies the performance of the proposed schemes by comparing its area under curve values with that of a conventional scheme, and confirms the usability of APK metadata for the purpose of APKs' accurate risk analysis.

1 はじめに

近年、Android 端末でのセキュリティインシデントは急増しており、実際、Android 端末向けマルウェアは増加の一途をたどっている。それらのマルウェアは Android Package (APK) として端末へ到着するため、APK を分析することによりマルウェアを検知する技術の研究開発がこれまで報告されている。多数の方式が報告されているが、その一つである DroidRisk [18](以下、DR) では、各パーミッションがマルウェアに悪用される確率と悪用された際の影響度を算出することにより、リスクを定量化する。そして、そのリスク値を閾値評価することによりマルウェアを検知する。また、Support Vector Machine (SVM) を用いて APK の特性の分類もしくは異常値検出によりマルウェアを検知する方式 [12] も提案されている。現在、マルウェアの検知率向上に向け、これらの技術の更なる発展が求められている。

本稿では、これらの方式を検証すると同時に、オンライン APK マーケットから得られる APK のアプリカテゴリ情報などのメタ情報を活用することにより、リスク分析精度の向上を試みる。まず、通常の APK ファイル [1] の特徴はアプリのコンテクスト毎に異なるため、DR にコンテクストを鑑みた方式を構築し、その性能を評価する。また、SVM を用いる方式では、上述の DR の改善に用いたすべてのパラメータを用いて性能を評価する。評価実験では、area under curve (AUC) 値 [3] を用いて性能を比較し、改善に寄与したポイントを考察する [2]。

2 DroidRisk ベースのアプローチ

本節では APK のメタ情報を用い、DR を拡張する。また、その有効性を評価する。

2.1 DroidRisk

DR はパーミッション要求パターンに基づき、APK のリスクレベルを定量化する。具体的には、各パーミッションがマルウェアに悪用される確率とその際の影響度を算出し、それらを乗算することにより各パーミッションのリスク値を算出する。そして、一つの APK が利用するすべてのパーミッションに対し本リスク値を合算し、その値を当該 APK のリスク値とする。

DR はまず式 (1) に従い各パーミッション p のリスクレベル $R(p)$ を定量化し、次に式 (2) に従い各 APK に要求されるすべての p の $R(p)$ を合計して APK のリスクレベル R_A を算出する。

$$R(p) = L(p) \times I(p) \quad (1)$$

$$R_A = \sum_i R(p_i) \quad (2)$$

ここで、$L(p)$ は p がマルウェアに利用される確率を、$I(p)$ は p がマルウェアに悪用された際のインパクトの程度を示している。この R_A に対し適切な閾値を設定することにより、APK ファイルがマルウェアであるか否かを判定可能となる。

DR は定数である $L(p)$ と $I(p)$ の値をデータセット全体から学習して決定するが、本来、アプリのタイプによりこれらの値の最適値は異なるはずである。例えば「Travel&Maps」に属するアプリは GPS 利用に必要なパーミッションを要求するが、「Multimedia」に属するアプリがそれを要求することは稀である。そのため我々は DR を拡張し、APK のコンテクストに応じて $I(p)$ と $L(p)$ の値を決定する手法を提案する。

2.2 カテゴリ別 DroidRisk

我々は、アプリカテゴリ毎にセキュリティリスクを定量化する方式 $\mathrm{DR_{CT}}$ を提案する。DR と異なり、$\mathrm{DR_{CT}}$ はアプリのカテゴリ毎にデータセットを学習することにより、カテゴリ毎に $L(p)$ と $I(p)$ の値を決定する。

$\mathrm{DR_{CT}}$ は DR 同様、各パーミッション p のリスクレベルを定量化した後に、各 APK が要求するすべての p のリスク値を合算し、各 APK のリスク値 R_A を算出するが、カテゴリ毎に分

[1] マルウェア以外の APK を「通常の APK」と呼ぶ

[2] 本稿の第 2.2 節までの内容は、本研究の初期検討結果 [20] にて詳述されている。それ以降の内容については未発表のものである。

けて計算している点で異なっている。具体的には、式 (3) および式 (4) に従い各パーミッションのリスクレベルを定量化する。

$$R(c,p) = L(c,p) \times I(c,p), \tag{3}$$

$$R_A = \sum_i R(c,p_i). \tag{4}$$

ここで、$L(c,p)$、$I(c,p)$、および $R(c,p)$ は、カテゴリ c の中においてパーミッション p がマルウェアに利用される確率、p がマルウェアに悪用された際のインパクトの程度、そして p のリスクレベルである。この R_A に対し適切な閾値を設定することにより、ある APK ファイルがマルウェアか否かを判定可能となる。

2.3 クラスタ別 DroidRisk

DR_{CT} はカテゴリ毎に APK の特徴が顕著に異なることを前提としているが、実際にはオンラインマーケットのカテゴリは人間が設定したものであり、また各 APK にカテゴリを付与するのも人間の主観により行われているため、必ずしも APK 分析に最適なものにはなっていない。そこで我々は、人為が大きく関わるカテゴリではなく、APK の特徴により自動生成した分類を利用する方式 DR_{CL} を提案する。

DR_{CL} では、アプリ説明文を用いて自動生成したクラスタに基づき APK を分類する。具体的には、DR_{CT} はまず、文献 [6] 同様、アプリ説明文の前処理、トピックモデル構築、クラスタリングを実施する。そして、生成されたクラスタを元に、リスクレベルの定量化を実施する。クラスタはコンピュータが自動生成するため、人間の主観や判断ミスを避けることができる。

データの前処理　アプリ説明文から Latent Dirichlet Allocation (LDA) [2] に利用可能な単語を生成する。まず、言語検知ライブラリである [4] を活用し、英語以外の言語で書かれている説明文を除外する。テキストではない記述、すなわち数字や HTML タグ、ウェブリンクや e メールアドレスなども、除外する。次に、説明文から語幹を抽出し、ストップワードを削除する。語幹の抽出には stemmify [3] を利用しており、ストップワードのリストアップには MALLET [4] の stoplist/en.txt を活用している。最後に、生成された説明文の単語数を検査し、単語数が 10 以下の説明文を除外する。

トピックの抽出　データの前処理を経た説明文を、今度は LDA にて取りこみ、多数のトピックを学習させる。本処理には MALLET を利用し、topic 数を合計 300[5]、topic 比率の閾値を 0.05、エントリ毎に抽出する最大 topic 数を 4 と設定する。結果として、最大 4 つの情報ペア {topic number, proportion value} を抽出する。

クラスタリング　このペア情報に対し、k-means [9] を用いたクラスタリングを実施する。生成するクラスタ数を 12 とし[6]、Ruby gem の “kmeans” 関数 [8] を用いてクラスタリングを実施する。

リスクレベルの定量化　DR_{CT} 同様、式 (3)、(4) により APK ファイルのリスク値を算出するが、DR_{CT} とは異なり c にカテゴリではなくクラスタを入力して算出する。すなわち、各クラスタにおいて各パーミッションのリスク値を算出し、各 APK が利用しているすべてのパーミッションについて、そのリスク値を合算することにより、当該 APK のリスク値の定量化を実施する。

2.4 評価実験

本節では、[18] 同様、マルウェア検知における AUC の観点からリスクレベル定量化の性能を評価する。

[3] https://rubygems.org/gems/stemmify

[4] http://mallet.cs.umass.edu

[5] topic 数は任意の値を設定可能であるが、MALLET のマニュアルにある説明文 ‘The number of topics should depend to some degree on the size of the collection, but 200 to 400 will produce reasonably fine-grained results.” に従い、今回は 300 とした。

[6] Opera Mobile Store[7] におけるカテゴリ数と同数のクラスタを用意。

[8] https://rubygems.org/gems/kmeans/

2.4.1 データセット

我々は、データセットとして、{APK ファイル, パーミッション要求情報, メタ情報, マルウェア判定結果 } の 4 種のデータを準備した。我々はまず、Opera Mobile Store より無料のソフトウェアを対象に 2014 年 1 月から 9 月の期間に合計 87,182 個の APK ファイルを収集し、ruby_apk [9] を用いて、それらのファイルの AndroidManifest.xml ファイルからパーミッション要求情報を抽出した。同時にそれら APK ファイルのメタ情報も同マーケットより収集した。そのメタ情報には、アプリカテゴリ、アプリ説明文、ダウンロード数などが含まれているが、本論文では、このうちアプリカテゴリおよびアプリ説明文のみ利用する。次に VirusTotal [10] を用いて、それらの APK ファイルがマルウェアか否かを判定した。VirusTotal は、複数のベンダから提供されている評価エンジンを用いて APK がマルウェアであるか否かを分析するサービスである。本稿では、1 つ以上の評価エンジンがマルウェアであると判定した PK ファイルをマルウェアとして扱う。但し、検知名に”Adware” の文字列を含んだり、検知名が著名なアドウェアである場合など、アドウェアと考えられる検知についてはマルウェアとしては扱わない。

我々は、ここから今回の分析に活用できないデータを除外すべく、パーミッション要求情報を抽出できないファイル、VirusTotal が正常に機能しないファイル、そして第 2.3 節のデータの前処理を通過できないファイルを除外した。その結果、40,478 個の通常の APK ファイルと 21,252 個のマルウェアの合計 61,730 個の APK ファイルを獲得し、これを今回の評価実験のデータセットとした。

2.4.2 パラメータ値の決定

測定に先立ち、各方式のパラメータを算出する。$L(p)$ および $L(c,p)$ については、データセットの統計分析により算出した。$I(p)$ および $I(c,p)$ については、我々は [18] 同様、各 p に対し計算する代わりに np と dp に対し、値を算出した。ここで、dp は Android 4.4 r1 において、protectionLevel の値が “dangerous” となっているパーミッション [11] であり, np はそれ以外のパーミッションを指す。$I(np)$ と $I(dp)$ の値の決定にあたっては、それらのパーミッションが悪用された際の実際のインパクトの程度を決定する手段は存在しないため、[18] 同様、Receiver Operating Characteristic (ROC) curve を用いた経験的手法により決定した。具体的には、$I(np)$ の値を 1.0 に固定し、$I(dp)$ の値を 1.0 から 3.0 まで 0.1 ずつ増加させた際の AUC 値を測定し、その AUC の値が最大となった際の $I(dp)$ の値を最終的な値として選定した。但し、危険なパーミッションは通常のパーミッションよりもインパクトが大きいため、$I(dp)$ の値は $I(np)$ の値より大きい範囲にて探索を実施した。$I(c,p)$ の値についても同様に経験的手法により決定した。ここで、DR はアプリカテゴリを考慮しないため、L と I の値はカテゴリにかかわらず一定値であるが、$\mathrm{DR_{CL}}$ の用いるパラメータの $L_c(p)$ と $I_c(p)$ はカテゴリ毎に異なる値となる点に留意されたい。

尚、測定にあたっては、データセット中の通常のソフトウェアおよびマルウェアについて、20%を評価用とし、残りを学習用として利用した。また、本稿におけるすべての評価は、同様のデータセットの利用を実施している。

2.4.3 測定結果と考察

表 1 に、AUC 値の測定結果を示す。ここで、DR は APK を分類する概念がないため、$\mathrm{DR_{CT}}$ および $\mathrm{DR_{CL}}$ と AUC 値を単純比較することはできない。そこで我々は、DR と同一のパラメータ (L および I) を用いて DR をアプリカテゴリ別に走らせる方式を $\mathrm{DR_{BD}}$ と定義し、$\mathrm{DR_{BD}}$ の AUC 値をベンチマーク対象として利用する [12]。また、単一の L と I の最適値を用いてアプリク

[9] https://github.com/SecureBrain/ruby_apk
[10] http://www.virustotal.com/ja

[11] http://developer.android.com/reference/android/content/pm/PermissionInfo.html
[12] 但し、DR と $\mathrm{DR_{BD}}$ は別物である点に留意されたい。$\mathrm{DR_{BD}}$ は DR よりもマルウェア検知の精度が高いと推察されるが、これら 2 つの方式の比較は本稿の範囲外とし、本稿では両者は同一の性能を呈するものとして扱う。

ラスタ別に DR を走らせる方式を DR_{CL^-} と定義し、DR_{BD} と同様にベンチマーク対象として利用する。

DR_{CT} の有効性　表 1 より、DR_{CT} は DR_{BD} よりも高い AUC 値を実現していることがわかる。本表では省略されているが、すべてのカテゴリにおいて AUC 値が向上した。これは、DR_{BD} では固定であったパラメータを、DR_{CT} は各カテゴリで別々の値を設定して最適化するためである。しかしながら、DR_{BD} と比較した DR_{CT} の有効性は表 1 を見る限り大きくない。また、DR_{CT} の AUC 値はカテゴリごとにバラツキが大きく、いくつかのカテゴリでは他のカテゴリと比べてマルウェア判定精度が低いことから、カテゴリの最適化の余地が大きいと考えられる。

DR_{CL} の有効性　表 1 では、すべての方式の中で DR_{CL} が最も高い AUC の平均値と中央値を示している。DR_{CL} の AUC 値は平均値ベースで 0.036 ポイント (+4.7%) 増加しており、また、標準偏差も小さくパフォーマンスが安定していることがわかる。また、本表は、AUC 値の向上にはカテゴリを利用するかクラスタを利用するかにかかわらず、よい分類をすること自体が有効で、パラメータの最適化よりも効果が大きいことを示している。実際、DR_{CT} の DR_{BD} に対する利点、DR_{CL} の DR_{CL^-} に対する利点は、DR_{CL} の DR_{CT} に対する利点に比べると相対的に小さくなっている。

3　SVM ベースのリスク分析

第 2 節では、DR のパラメータの最適化によりリスク分析の最適化を実施している。本方式は、マルウェア分析にも利用可能であるものの、マルウェアの判定にはリスク値を一次元の閾値処理で判定する方式が最適とは限らない。実際、2 値のマルウェア判定には、Two-class SVM (TC-SVM)[13] が有効であることが知られているため、本節では、SVM の中でも TC-SVM を用いてメタ情報を活用するアプローチを定義し、そのパフォーマンスを測定することで、上述のリスク分析技術を考察する。

[13]本稿では特に断りがない限り、TC-SVM のことを SVM と記す。

3.1　Support Vector Machine

TC-SVM [16, 14] では、訓練サンプルの集合 $D = (x_i, y_i)|x_i \in R^d, y_i \in -1, +1, i = 1, ..., l$ より、下式の線形関数を学習する。

$$f(x) = \langle w, x \rangle + b \tag{5}$$

ここで、w は重みづけベクトル、b は閾値を示し、もし $f(x) > 0$ であれば正値の、そうでなければ負値のクラスが x に割り当てられる。w と b は、2 つのクラスの境界に最も近いサンプルとの距離が最大となる超平面が求まるよう選定される。訓練データセットは線形の超平面にて必ずしも分離できないが、その際には式 (5) にカーネル関数を組み込むことにより、分離を可能とする。各種のカーネル関数が存在するが、本稿ではラジアル基底関数を利用する。

3.2　SVM に基づく APK 分析

本節では、SVM を活用して APK がマルウェアか否かを検査する。我々は SVM に活用する特徴データに、従来のパーミッション要求情報のみでなく、第 2 節にて用いた APK のメタ情報を活用した。すなわち、APK のカテゴリ情報、アプリ説明文、クラスタ情報などを活用して、SVM を最大限活用する方式を検討した。本方式では、パーミッション要求、カテゴリ、およびクラスタ情報を SVM への入力として活用する。尚、本 APK 分析は、SVM に入力する情報の違いにより、下記の 4 通りの名前で区別するものとする。

1. SVM_B: パーミッション要求のみを SVM に活用
2. SVM_{CT}: パーミッション要求に加えてカテゴリ情報を SVM に活用
3. SVM_{CL}: パーミッション要求に加えてクラスタ情報を SVM に活用

表 1: DroidRisk ベースの各手法の AUC 値比較

	DR_{BD}	DR_{CT}	DR_{CL^-}	DR_{CL}
平均値	0.767	0.782	0.800	0.803
加重平均値	0.781	0.789	0.797	0.800
メジアン値	0.766	0.767	0.800	0.802
最小値	0.666	0.692	0.746	0.750
最大値	0.913	0.918	0.854	0.860
標準偏差	0.075	0.073	0.028	0.028

4. SVM_{CTCL}: パーミッション要求に加えてカテゴリ及びクラスタ情報を SVM に活用

3.3 評価と分析

第 2.4.1 節のデータセットを用いて各方式のパフォーマンスを測定した。表 2 にその結果を示す。尚、本表の数値は、マルウェア検知の精度が最大となる際の測定値である。

3.3.1 SVM に対するメタ情報の有用性

本表より、SVM の入力にパーミッション要求だけでなくカテゴリもしくはクラスタ情報を用いることで、より高いパフォーマンスを実現できることがわかる。また、その双方を活用した方式が最良のパフォーマンスを示していることがわかる。これは、カテゴリ及びクラスタがそれぞれ APK に関する独自の特徴情報を反映しているためである。

3.3.2 DroidRisk と SVM の比較

本節では表 1 と表 2 を比較し、マルウェア検知精度とリスク分析技術の有用性について考察する。

マルウェア検知精度比較　これらの表より、SVM ベースの方式は DR ベースの方式よりも高いマルウェア検知精度を実現していることがわかる。これは、SVM ベースの方式はパーミッション要求、カテゴリ、クラスタ情報などの入力情報に基づき、APK ファイルを 2 つのグループの分類を最適化しているためである。これに対し DR ベースの方式は、APK のカテゴリ作成と確立論に基づくリスク値の算出を最適化するが、本稿の実装においては、そのカテゴリ数の最適化を行っていない。また、最適化したとしても、APK ファイルを 2 つのグループに分類するという目的に対しては、リスクの数値化を目的としている DR ベースの方式よりも SVM に基づく方式のほうが高いパフォーマンスを示すのは当然であると考えている。すなわち、APK を 2 分するという目的においては、「パーミッションがマルウェアにより利用される確率と、利用された際のインパクトの乗算により、APK のリスクを数値化する」という固定的な特徴の射影を行っている DR に対し、射影方法を学習により柔軟に変更している SVM のほうが、より効率的な射影を実現できるためであると考えている。

リスク分析技術の有用性　APK をマルウェアとそれ以外に二分する際には、SVM ベースのアプローチが最適なことが多いと考えられ、今回の結果もそれに倣ったものとなっている。しかしながら、「注意すべきアプリ」とそれ以外に二分する際には、その限りではない。実際、DR ベースの方式は、リスクに対する考え方、知見、ポリシーなどを反映することが比較的用意であり、意図的に特定のパラメータに重みづけすることが可能である。例えば、permission.READ_CONTACTS にアクセスするアプリにより注意を払う際には、DR ベースの方式であれば、当該パーミッションに対応する I 値を増加させることで容易に対応可能となる。マルウェア判定の精緻さを求めるケースと、マルウェアでなくともリスクの高いアプリに対して

表 2: SVM を用いた手法の性能比較

	SVM_b	SVM_{CT}	SVM_{CL}	SVM_{CTCL}
精度	0.932	0.939	0.936	0.942
真陽性率	0.807	0.834	0.811	0.836
偽陽性率	0.027	0.026	0.024	0.023
AUC	0.926	0.954	0.950	0.962

警告したいケースの双方が存在するため、ユースケース毎に利用する方式を工夫してく必要があると考えられる。そして、DR ベースの方式は、マルウェアの警告漏れは最小化しつつ、リスクの高い APK に対して警告を発する技術として更なる発展をしていくことで、その有用性を高めることができる。本稿では、本分析は範囲外であり、今後検討していきたい。

4 関連研究

関連研究の一つにアプリ検査のためのガイドライン [17] がある。本ガイドラインは、アンドロイドアプリにも適用可能な内容であり、各組織内において実施すべきアプリ検査プロセスを説明している。

また、APK ファイルの分析技術も多数報告されている。第 Sec:DroidRisk 節で紹介した DR に加え、通常は使われないパーミッションを Category-based Rare Critical Permission (CRCP) と定義し、CRCP の利用を監視することにより APK のマルウェア判定を行う方式 [13] も報告されている。本手法では、アプリカテゴリ毎にパーミッション要求の頻度を分析し、利用頻度が極めて少ないもしくは使われないパーミッションを CRCP として選定する。CRCP はアプリカテゴリ毎に異なる点に留意されたい。

アプリ説明文を分析する研究もいくつか報告されている。WHYPER[11] はその先駆的な試みで、自然言語処理にて、アプリ説明文と実際のアプリの挙動のギャップを検知する。API 文書からセマンティックグラフを構築し、そのグラフを元にアプリ説明文を分析することで、そのギャップを検出する。

CHABADA[6] も、アプリ説明文と実際のアプリの挙動のギャップを検知する手法を、別のアプローチで提案している。まず、アプリ説明文からクラスタを生成し、その各クラスタ内に属する APK に対し One-Class SVM (OC-SVM) を実施する。そして、異常値を示すものを検出している。本手法はマルウェア検知にも利用できるが、その目的のためには例えば TC-SVM を利用するなどの改善が可能であると考えている。今後、検証していきたい。尚、本稿の第 2.3 節に記載のクラスタ生成手法は、利用しているライブラリとパラメータは異なるものの、CHABADA 記載のものと同一である。

SVM を用いたマルウェア分析に関する研究もいくつか報告されている。文献 [12] では、パーミッション要求および API 呼出を入力として SVM を用いたマルウェア検出方式が報告されている。尚、本稿の範囲では、第 2 節と第 3 節の比較のために API 呼出の分析結果は掲載していない。しかしながら、API 呼出を分析することで、パーミッション要求分析よりも精度の高い分析が実現できるため、今後、API 呼出レベルでの分析を継続していく所存である。

ほかにも、多数の動的および静的解析技術が報告されているが [5, 7, 8, 10, 15, 19]、紙面の都合上割愛する。

5 結論と今後の方向性

本稿では、オンラインマーケットから取得したアプリのメタ情報を活用することで、リスクレベルの定量化およびマルウェア検知技術の精緻化を実現した。今後、更なる分析精度向上をめざすための手法の検討を続けていく。例えば、パーミッションに代わり API 呼出を用いて特徴抽出を実施するほか、要求されているパーミッションと実際に必要なパーミッションの差異、すなわちパーミッションギャップ [1] を考慮する

などして、より精緻なリスク・マルウェア分析が実施できると考えている。

参考文献

[1] Bartel A, Klein J, Le Traon Y, et al. Automatically securing permission-based software by reducing the attack surface: An application to android. In *ASE*, 2012.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 2003.

[3] C. D. Brown and H. T. Davis. Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 2006.

[4] Cybozu Labs. Language Detection Library for Java. `https://code.google.com/p/language-detection/`, 2014.

[5] Enck W, Octeau D, McDaniel P, et al. A study of android application security. *USENIX Security Symposium*, 2011.

[6] Gorla A, Tavecchia I, Gross F, et al. Checking app behavior against app descriptions. In *ICSE*, 2014.

[7] Grace M, Zhou Y, Zhang Q, et al. Riskranker: scalable and accurate zero-day android malware detection. In *MobiSys*, 2012.

[8] C. Jarabek, D. Barrera, and J. Aycock. Thinav: Truly lightweight mobile cloud-based anti-malware. In *ACSAC*, 2012.

[9] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[10] Mulliner C, Oberheide J, Robertson W, et al. Patchdroid: Scalable third-party security patches for android devices. In *ACSAC*, 2013.

[11] Pandita R, Xiao X, Yang W, et al. Whyper: Towards automating risk assessment of mobile applications. In *USENIX Security*, 2013.

[12] N. Peiravian and X. Zhu. Machine learning for android malware detection using permission and api calls. In *ICTAI*, 2013.

[13] Sarma BP, Li N, Gates C, et al. Android permissions: A perspective combining risks and benefits. In *SACMAT*, 2012.

[14] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[15] Shabtai A, Kanonov U, Elovici Y, et al. “andromaly”: a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 2012.

[16] V. Vapnik. *Statistical learning theory*. Wiley, 1998.

[17] Voas J, Quirolgico S, Michael C, et al. *Technical Considerations for Vetting 3rd Party Mobile Applications (Draft)*. 2014. NIST special publication 800-163.

[18] Wang Y, Zheng J, Sun C, et al. Quantitative security risk assessment of android permissions and applications. In *DBSec*, 2013.

[19] Zhou Y, Wang Z, Zhou W, et al. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In *NDSS*, 2012.

[20] 高橋健志, 三村隆夫, 西田雅太, 中尾康二. カテゴリに基づく Android アプリのリスク値定量化技術の検討. In 信学技報. 2014.