情報処理学会論文誌 Vol.58 No.12 1843–1855 (Dec. 2017)

Androidアプリケーションのライブラリからみた脆弱性分析

古川 凌也[1,2,a)] 永井 達也[1] 熊谷 裕志[2] 神薗 雅紀[2]
白石 善明[1] 高野 泰洋[1] 毛利 公美[3] 星澤 裕二[2] 森井 昌克[1]

受付日 2017年3月13日, 採録日 2017年9月5日

概要：サードパーティ製のライブラリに含まれる脆弱性に起因して，多くのアプリに脆弱性が作りこまれるケースが報告されている．本論文では，日本の Google Play のランキングに掲載されているアプリ 15,064 個を対象に，一般的によく利用されているサードパーティ製ライブラリについて，当該アプリが利用しているバージョンを特定し，その分布を調査した．その結果，Google Play のランキングに掲載されているアプリの中にも，脆弱性の報告されている古いバージョンのライブラリを利用しているものが存在していることが分かった．さらに，そのような脆弱性がマーケットにおいて，どのように残留・分布しているのかを各アプリのランキング順位や累計ダウンロード数，ユーザレビューや最終更新日といったメタデータの項目をもとに調査したところ，それらの値によらず古いバージョンのサードパーティ製ライブラリを利用するアプリは存在し，ランキング順位の高いアプリや累計ダウンロード数の多いアプリ，ユーザレビューで高い評価を得ているアプリや最終更新日の新しいアプリの中にも脆弱性を含む可能性のあるアプリを複数確認することができた．

キーワード：Android，サードパーティ製ライブラリ，脆弱性

A Vulnerability Analysis of Android Applications from the View of Third-party-libraries

Ryoya Furukawa[1,2,a)] Tatsuya Nagai[1] Hiroshi Kumagai[2] Masaki Kamizono[2]
Yoshiaki Shiraishi[1] Yasuhiro Takano[1] Masami Mohri[3] Yuji Hoshizawa[2] Masakatu Morii[1]

Received: March 13, 2017, Accepted: September 5, 2017

Abstract: Security risks potentially involved in third-party-libraries on Android have been shown to expose many apps to vulnerability. In this paper, we investigate 15,064 of popular android apps to identify their library versions and to reveal statistical distribution of the versions. As a result, several apps even published on Japanese Google Play turn out to be using old version libraries warned their security risks. Furthermore, we survey how the vulnerability is left and spread on the market by mining metadata such as the ranking order, the number of downloads, the rating and the last modification date. The results presented in this paper reveal that, being independent of the metadata, there are apps using the libraries of out-of-date version, which verifies that some apps can be vulnerable although they are highly ranked, downloaded, rated, and/or updated recently.

Keywords: Android, third-party-libraries, vulnerability

1. はじめに

Android アプリのマーケットに対するさまざまな調査研究がなされている．石井らによる研究 [1], [2] では，アプリのリパッケージングに着目し，マーケットにおけるクローンアプリの流通状況やその分類別の内訳が明らかになっ

[1] 神戸大学
Kobe University, Kobe, Hyogo 657–8501, Japan
[2] PwC サイバーサービス合同会社
PwC Cyber Services LLC., Chiyoda, Tokyo 100–0004, Japan
[3] 岐阜大学
Gifu University, Gifu 501–1193, Japan
[a)] ryoya.furukawa@pwc.com

た．吉田らによる研究 [3] では，マーケットのアプリの署名情報を大規模に調査し，多くのアプリにおいて適切な電子署名がなされていないことと，それらのアプリがかかえる潜在的な脅威が示された．また孫らの研究 [4], [5] では，マーケットにおける不自然なレビューと悪性アプリの関連を調査することで，アプリストアにおけるプロモーショナル攻撃が現実的な脅威であることが示された．渡邉らによる研究 [6] では，マーケットにおけるアプリの説明文とアプリによるプライバシ情報へのアクセスの相関を分析し，多くのアプリが説明文での言及なしに暗黙的にプライバシ情報へのアクセスを試みることをその要因とともに明らかにした．石井らの研究 [7] では，公式の Google Play マーケット以外の Android サードパーティマーケットの安全性と健全性について大規模な調査を行い，問題点を明らかにしたうえで対応策を示している．これらのような市場の調査に関する研究は，脅威の発生源やマーケットへの影響を正確に把握し，対策を講じるうえで重要である．

2014 年 9 月 3 日に CERT/CC が行った報告 [8] では，複数のサードパーティ製ライブラリに SSL 証明書を適切に検証しない脆弱性が存在し，少なくとも 617 個のアプリが中間者攻撃の被害をうける可能性が示された．Android アプリの開発では，広告によるアプリの収益化，ソーシャルサービスとの統合，アプリの機能性の向上などを目的として，第三者が開発したサードパーティ製のフレームワークやライブラリ（以下ではライブラリと総称する）を利用することが一般的である．アプリの脆弱性とライブラリとの関連を調査した Watanabe らの研究 [9] によると，無料アプリにおける脆弱性の約 70%と有料アプリにおける脆弱性の 50%がライブラリのコードに起因するものであり，その大部分がサードパーティ製のライブラリによるものであった．

多くのアプリに利用されるライブラリに重大な脆弱性が発覚した場合，その影響範囲は非常に大きくなることが予想されるため，ライブラリの開発者は早急にセキュリティパッチを公開することが求められる．しかし，ライブラリの開発者が迅速な対応を行ったとしても，公開されたセキュリティパッチを適用し，アプリをアップデートするといった脆弱性への直接の対応はアプリの開発者に委ねられている．そのため，必ずしもすべてのアプリの脆弱性が取り除かれるとは限らず，長期間，重大な脆弱性が放置される可能性がある．すでに，Google Play で公開されているアプリについて，ある有名なソーシャルサービス，クラウドサービスの外部ライブラリに注目し，バージョンの変化を継続的に調べた報告 [10] がある．そこでは，ライブラリの新バージョン公開から当該アプリがその更新に追従するまでの時間が開いているという結果が示されている．

本論文では，多くの Google Play のアプリが利用しているサードパーティ製ライブラリについて，マーケットでのバージョンの分布に注目する．具体的には，日本の Google Play で公開されている各カテゴリの無料トップランキングに掲載されているアプリを対象に，サードパーティ製ライブラリに起因する脆弱性を持つアプリの拡散状況の調査，分析を行う．

まず，予備調査として，調査対象のアプリが利用するサードパーティ製ライブラリのバージョンを特定し，その分布を調べた．その結果，日本の Google Play でランキングに掲載されているアプリの中にも脆弱性が報告されている古いバージョンのサードパーティ製ライブラリを利用するアプリが複数存在することが分かった．

そこで，予備調査の結果をふまえ，マーケットに重大な脆弱性がどのように残留しているか本調査を行う．具体的には，サードパーティ製ライブラリの各バージョンを利用するアプリがマーケットにおいてどのように分布，拡散しているか，以下の仮説をもとに分析する．

仮説 1. Google Play に掲載されているランキングにおいて上位のアプリほど，アプリの更新率が高いと考えられる．その結果，ランキング上位のアプリほど新しいバージョンのライブラリが利用されている．

仮説 2. 同様の理由で累計ダウンロード数の多いアプリほど，新しいバージョンのライブラリが利用されている．

仮説 3. ユーザレビューの高いアプリは開発者の管理が行き届いていると考えられる．したがって高評価のアプリほど新しいバージョンのライブラリが利用されている．

仮説 4. アプリが更新されるとき，ライブラリも新しいバージョンのものに変更されると考えられる．したがって最終更新日が最近であるほど新しいバージョンのライブラリが利用されている．

本論文の構成は以下のとおりである．2 章では今回調査対象としたサードパーティ製ライブラリ，およびフレームワークとそれらに関連して報告されている脆弱性について説明する．次に 3 章では，予備調査の方法と結果を示し，4 章では，本調査の結果を示す．続く 5 章では，本調査でのマーケットに残留している脆弱性の特徴的な結果をふまえて実施した追加調査の結果を示す．6 章では，本研究調査にかかる制約事項について述べる．最後に 7 章で本論文をまとめる．

2. 調査対象のフレームワーク・ライブラリ

本論文では，過去に脆弱性が報告されている以下の Android 開発用フレームワーク・サードパーティ製ライブラリを調査の対象に定めた．

AdColony SSL の証明書を適切に検証しない脆弱性（CVE-2014-5524）が報告されている．

Adobe AIR v22.0.0.153 およびそれ以前のバージョンに，ランタイム分析を安全に転送できない脆弱性（CVE-

2016-6936）が報告されている．これに対して Adobe は，Adobe AIR SDK を v23.0.0.257 に更新することを推奨しているが，適用優先度は，開発者の任意のタイミングでの適用をしめす「3」としている．

Apache Cordova v4.1.1 未満に CVE-2015-5256 など複数の脆弱性が報告されており，Google からアプリ開発者に対して該当するバージョンを利用しないよう注意勧告がなされている．

Appsflyer v2.3.1.12 未満に SSL の証明書を適切に検証しない脆弱性（CVE-2014-5528）が報告されている．

Chartboost v2.0.2 未満に SSL の証明書を適切に検証しない脆弱性（CVE-2014-6025）が報告されている．

Dropbox-API v1.5.4 から v1.6.1 において，攻撃者が端末上のアプリを被害者の認証なしに攻撃者に制御された Dropbox アカウントへ接続することを許容してしまう脆弱性（CVE-2014-8889）が報告されている．

Facebook SDK v3.15.0 にアクセストークンを奪われアカウントを乗っ取られる危険のある脆弱性が報告されている [11]．

inmobi SSL の証明書を適切に検証しない脆弱性（CVE-2014-5526）が報告されている．

libpng v1.0.66，v.1.2.56，v.1.4.19，v1.5.26 未満であった場合に，CVE-2015-8540 の脆弱性の影響を受ける．この脆弱性について，該当するバージョンを利用しないよう，Google からアプリ開発者に対してすでに注意勧告がなされている．

OpenSSL v1.02f/1.01r 未満であった場合に CVE-2015-3194 の脆弱性の影響を受ける．Google はすでにこの脆弱性について，該当するバージョンを利用しないよう，アプリ開発者に対して注意勧告を行っている．

Tapjoy SSL の証明書を適切に検証しない脆弱性（CVE-2014-5527）が報告されている．

ここにあげたライブラリ以外にも，脆弱性が報告されているライブラリ，フレームワークは存在するが，今回の調査では，AppBrain が公開している統計情報 [12] をもとに，利用率の高いものを優先して調査を行った．

3. 予備調査：アプリが利用するサードパーティ製ライブラリのバージョンの分布

本章で述べる予備調査の目的は，調査対象のアプリが利用するサードパーティ製ライブラリのバージョンの分布を調査し，Google Play に公開されているアプリの中にライブラリに起因する脆弱性を含むアプリが存在するか否かを確認することである．

3.1 データセット

本論文では，2017 年 2 月 22 日の日本の Google Play における全カテゴリの無料トップランキングに掲載されていたアプリの Apk ファイルのうち，重複を除いた 15,064 個を調査対象とした．なお，調査対象の Apk ファイルの収集には，Google Play Unofficial Python API [13] を用い，ランキング順位や累計ダウンロード数，ユーザレビューの情報やアプリ開発者の情報といったメタデータも同時に取得した．

3.2 バージョン情報の特定

ライブラリおよびバージョン情報特定方法の関連研究に以下のようなものがある．Addetect [14] はアプリのパッケージ構造に対して階層型クラスタリングを行い，アプリケーションの主要なモジュールを検出することで外部ライブラリを特定する．Libradar [15] はアプリ内のサブパッケージごとに，コードに含まれる Android API コールの頻度から特徴ベクトルを生成し，クラスタリングを行うことでサードパーティ製のライブラリとそのバージョンを特定する．LibScout [10] はパッケージのツリー構造とコード内に定義されているメソッドの引数と返り値をもとに生成したハッシュ値をシグネチャとして外部ライブラリとそのバージョンの特定を行う．これらの方法は正確にライブラリやバージョン情報を取得することを目的としている．本研究ではライブラリの分布の傾向を把握することが目的であるため，以下のような簡便な方法とする．

アプリやライブラリのバージョン情報は，しばしば Apk ファイル内の class ファイルや so ファイル，txt ファイル内にハードコーディングされている．そこで，Apk ファイルに含まれる文字列情報を収集し，その中に含まれるバージョン情報を調査した．具体的な方法を以下のとおりである．

i. 各アプリについて，Apk ファイルを解凍して得られる全ファイルに含まれる正規表現*.*.*をバージョン情報の候補とし，前後 128 バイトの文字列情報，バージョン情報の候補が含まれたファイルのパスとともに抽出する．Apk ファイルの解凍には Apktool [16] と dex2jar [17] を使用し，パッケージ内の全ファイルが解凍されるまで再帰的に処理を行った．
ii. 得られた候補の中から目的のライブラリに対応するバージョンの情報を手作業で特定する．
iii. 特定したバージョン情報の前後 128 バイトの文字列情報とパスに含まれた単語をもとにシグネチャを作成する．シグネチャの例を**図 1** に示す．このとき，同一のライブラリであってもバージョンが違えばシグネチャも異なる場合があるため 1 つのライブラリにつきシグネチャが複数になることもある．
iv. 各 Apk から収集したバージョン情報の候補とシグネチャが一致するものを目的のライブラリのバージョン情報として特定する．

この調査では，上記の手順でバージョン情報の特定・収

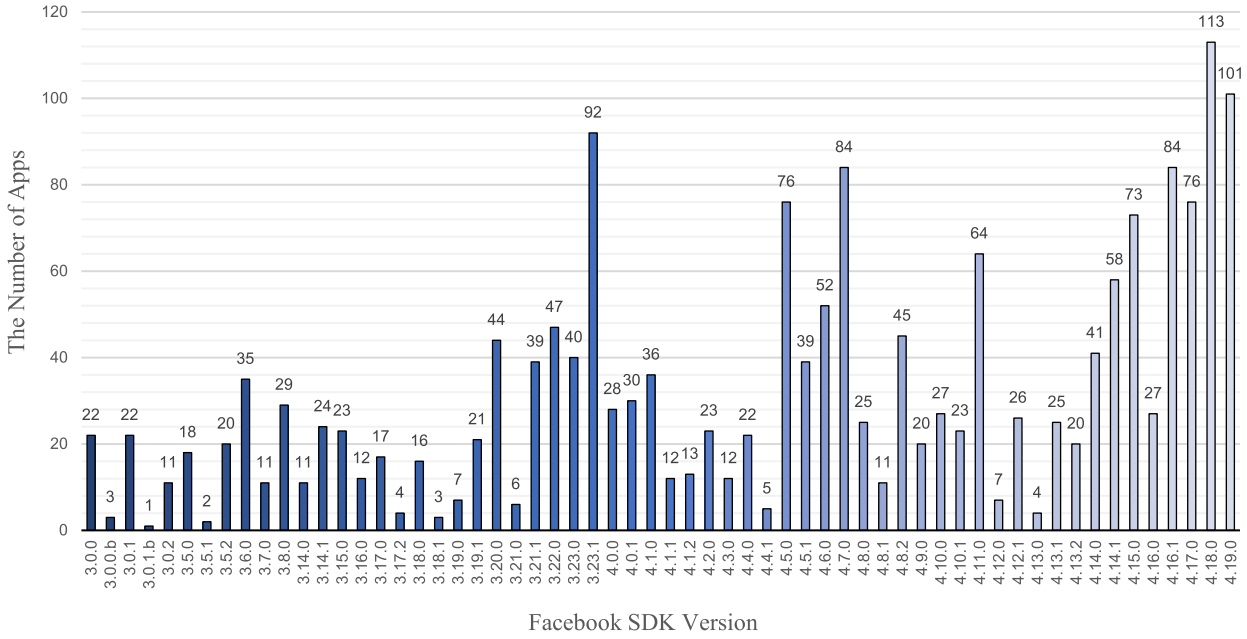

図 2 データセット内のアプリ利用されていた Facebook SDK のバージョンの分布

Fig. 2 Distribution of Facebook SDK versions for the Apps in our dataset.

バージョン情報が含まれているファイル

classes.dex/org/apache/cordova/CordovaWebView.class

バージョン情報の前後128バイトの文字列情報

2 D !org/apache/cordova/CordovaWebView
java/lang/Object CORDOVA_VERSION
Ljava/lang/String; *.*.* backHistory ()Z canGoBack
clearCache ()V (Z)V Ljava/lang/Deprecated; clearHistory
getConte

図 1 Apache Cordova のシグネチャの例

Fig. 1 Example of a version signature in Apache Cordova.

集を行ったが，すべてのライブラリのバージョンを特定できるわけではない．たとえばコードの難読化や圧縮 [18] によって本来シグネチャにマッチするコード部分が消失した場合やバージョン情報が文字列としてハードコーディングされておらず，実行時に動的に生成されるような場合には有効ではない．

3.3 予備調査の結果

収集したバージョン情報のうち，例として Facebook SDK のバージョンの分布を**図 2** に示す．横軸は検出された Facebook SDK のバージョン番号，縦軸はそのバージョンの Facebook SDK を利用していたアプリの個数である．

v3.x 系列，v4.x 系列のそれぞれにおいて，新しいバージョンの分布が高くなっている．しかし，古いバージョンを利用するアプリも多く存在し，脆弱性の報告されている v3.15 を利用するアプリも複数確認することができる．その他のライブラリのおいても，比較的古いバージョンや 2 章で述べた脆弱性の報告されているバージョンを利用するアプリが複数存在することが確認できた．シグネチャが確認されたアプリの個数とそのうち脆弱性が報告されているバージョンを利用していたアプリの個数を**表 1** にまとめる．Google Play に公開されているアプリの中にも，ライブラリに起因する脆弱性の影響を受けるアプリが存在することが分かった．

表 1 データセットに含まれるアプリ 15,064 個のうちシグネチャがマッチしたアプリの個数とそのうち脆弱性を持つバージョンのライブラリを利用していた個数

Table 1 The number of applications that contains signatures and those using a vulnerable library version among 15,064 Apps in our dataset.

ライブラリ	シグネチャがマッチしたアプリの個数	脆弱性を持つバージョンの利用数
AdColony	436	436
Adobe-AIR	164	111
Apache Cordova	811	112
Appsflyer	301	4
Chartboost	309	0
Dropbox-API	236	33
Facebook SDK	1848	23
inmobi	388	388
libpng	1300	444
OpenSSL	1120	798
Tapjoy	225	225

4. 本調査：ライブラリに起因する脆弱性を持つアプリの拡散状況の分析

3.3 節で，ライブラリのバージョンに起因する脆弱性を

表 2 ライブラリのバージョンとメタデータの各項目との間の Kendall 順位相関係数
Table 2 Kendall rank correlation coefficient between library versions and each item of metadata.

ライブラリ	ランキング順位	累計ダウンロード	レビュー点	最終更新日
AdColony	0.26	0.96	0.96	0.91
Adobe-AIR	0.17	0.94	0.94	0.82
Appsflyer	0.16	0.96	0.95	0.88
Chartboost	0.26	0.96	0.96	0.88
Dropbox-API	0.13	0.92	0.92	0.77
Facebook SDK	0.24	0.92	0.92	0.93
inmobi	0.18	0.96	0.97	0.90
libpng	0.27	0.94	0.94	0.93
OpenSSL	0.29	0.95	0.95	0.93
Apache Cordova	0.05	0.96	0.96	0.94
Tapjoy	0.34	0.95	0.95	0.85
平均	0.21	0.95	0.95	0.89

持つアプリが Google Play に公開されているアプリの中に複数存在していることが確認できたことを述べた．本章では，ランキング順位や累計ダウンロード数，ユーザレビューの評価値，最終更新日といった Google Play におけるアプリのメタデータに対して，各アプリが利用するライブラリのバージョンがどのように分布し，脆弱性を持つアプリが拡散しているのかを確認する．

確認方法として，まず各ライブラリのバージョン番号に対して，総合ランキングの順位と累計ダウンロード数，ユーザレビューの評価値，最終更新日についてそれぞれ Kendall の順位相関係数 [19] を求める．この数値が高いほど，バージョンの新しさとメタデータの間に強い関連性があるといえる．Kendall の順位相関係数を計算した結果を**表 2** に示す．また，アプリのメタデータの値に対してライブラリのバージョンが実際にはどのように分布しているかについてヒートマップを用いて確認する．本論文で用いるヒートマップにおいて，赤で塗られた箇所は脆弱性を持つバージョン，青に塗られた箇所は脆弱性の報告されていないバージョンを表しており，色が濃いほどその位置に分布するアプリの個数が多いことを表している．このヒートマップは，1 章に述べた仮説のもと，右上に行くほど青色が濃くなり，赤色は薄くなることが期待される．

4.1 ランキング順位に対するバージョンの分布

本節ではランキング順位に関する仮説 1 を検証する．ランキング順位とバージョンの新しさとの間には，平均して 0.21 の相関係数が得られた．したがって，ランキング順位とバージョンの新しさの間に有意な関連性はない．

総合ランキングの順位に対する libpng のバージョンの分布を**図 3** のヒートマップに示す．libpng は比較的多くのアプリに利用されており，検出されたバージョンの幅も大きいことから，ヒートマップの特徴が読み取りやすいため例とした．Google Play の総合ランキングは 1 位から 540 位までのみが公開されているため，ヒートマップにおけるランキング順位もその範囲に限られている．

libpng のバージョンごとの分布は v1.6.2, v1.6.10, v1.6.16 が特に濃くなっている．v1.6.10 と v1.6.16 については，それぞれ CVE-2014-0333 と CVE-2014-9495 に対応する修正が含まれていることから，積極的にアップデートが行われたものと考えられる．v1.6.2 については，人気のゲームアプリ開発用フレームワークである cocos2d-x の v3.0 から v3.5 が内部で利用しており，その影響を受けたものと考えられる．実際，libpng を利用しているアプリのカテゴリ別の数量をみると，全体の中ではゲームカテゴリのアプリの比率は約 15%であったのに対し，v1.6.2 を利用していたアプリに限定すると，約 42%がゲームカテゴリであった．

ランキングの順位に対するバージョンの分布をみると，ランキング順位にかかわらず，古いバージョンと新しいバージョンを利用しているアプリが混在している．また，1 位から 50 位の高順位の層においても v1.5.12 などの脆弱性のある古いバージョンを利用しているアプリが存在している．これらの結果から仮説 1 に反する以下の事実が示された．

【事実 1】 ランキング順位にかかわらずバージョンの古いものと新しいものが混在している．

【事実 2】 ランキング順位が高いアプリであっても新しいバージョンのライブラリを使っていないことがある．

【事実 1】と【事実 2】は，この他の調査対象のライブラリにも確認できている．

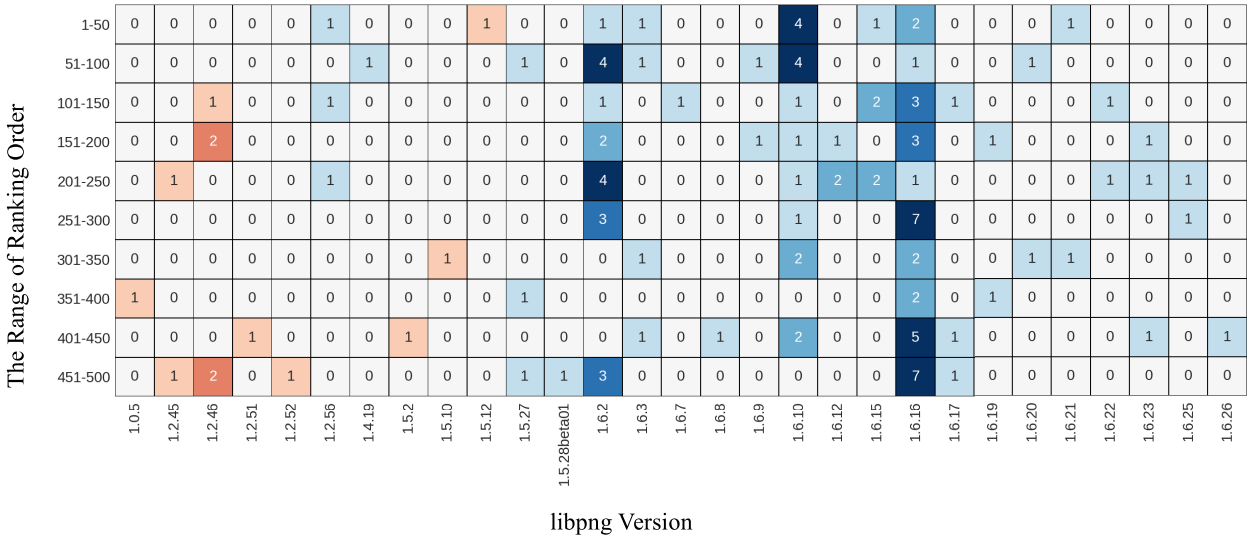

図 3 ランキングの順位に対する libpng のバージョンの分布
Fig. 3 Distribution of libpng versions vs. ranking orders.

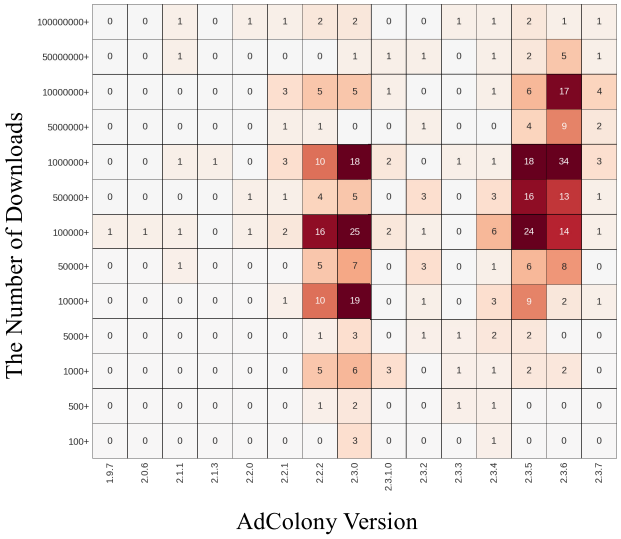

図 4 累計ダウンロード数に対する AdColony のバージョンの分布
Fig. 4 Distribution of AdColony versions vs. number of downloads.

4.2 累計ダウンロード数に対するバージョンの分布

本節では，累計ダウンロード数に関する仮説 2 について検証する．累計ダウンロード数とバージョンの新しさとの間の相関係数は，平均して 0.95 となった．これは仮説 2 の正当性を裏付ける．

データセット内での検出数が多い AdColony に注目して，累計ダウンロード数に対するバージョンの分布を**図 4** に示す．AdColony は v2.3.0，v2.3.6 の分布が特に大きくなっている．著者が調査した限り AdColony2.x 系列の更新履歴が残っておらず，原因は不明である．また，AdColony はアプリ収集時点で v3.0.7 までがリリースされていることを確認しているが，今回のデータセットからは v3.0.x 系列は検出されなかった．v3.0.x 以降の AdColony のコードからは，シグネチャにマッチする部分が消失している可能性がある．

相関係数とヒートマップから観測される事実は次のとおりである．

【事実 3】 累計ダウンロード数の多いアプリでも新しいバージョンのライブラリを使っていないことがある．

【事実 4】 全体としてはダウンロード数の多いアプリほど新しいバージョンが利用されている．

ヒートマップの特徴が比較的読み取りやすいことからここでは AdColony を例示したが，【事実 3】と【事実 4】は，AdColony 以外の調査対象のライブラリにおいても確認できている．

4.3 ユーザレビューに対するバージョンの分布

本節ではユーザレビューに関する仮説 3 についての検証を行う．ユーザレビューの値とバージョンの新しさとの間の相関係数は，平均して 0.95 となった．これは仮説 3 の正当性を裏付ける．

ユーザレビューに対する Dropbox-API のバージョンの分布を**図 5** に示す．Dropbox-API は，v1.6.3 の分布が特に高くなっており，v1.5.4 と v1.6.1 の分布も少し高い．v1.6.3 については，v1.5.4 から v1.6.1 までに影響を与える脆弱性 CVE-2014-8889 の報告にともない，多くの開発者が当時の最新版である v1.6.3 に SDK を更新したことが要因として考えられる．v1.5.4 については v1.5.x 系列の中で最新版であること，v1.6.1 については v1.6 から OAuth2.0 [20] による認証機能が追加された影響が考えられる．

相関係数とヒートマップから観測される事実は次のとおりである．

【事実 5】 ユーザレビューが高評価のアプリでも新しいバージョンのライブラリを使っていないことがある．

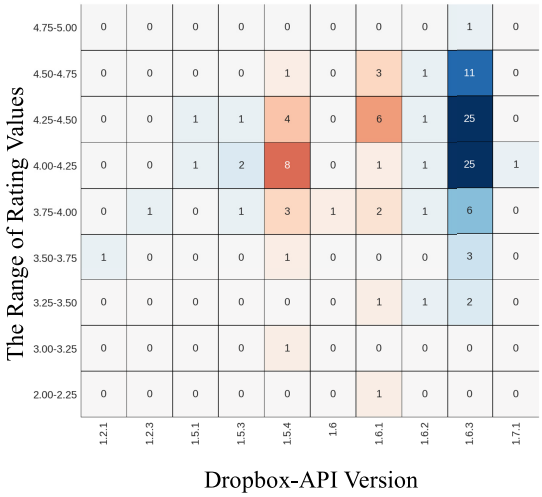

図 5 ユーザレビューの値に対する Dropbox-API のバージョンの分布

Fig. 5 Distribution of Dropbox-API versions vs. rating values.

【事実 6】 全体としてはユーザレビューが高評価のアプリほど新しいバージョンが利用されている．

ヒートマップの特徴が比較的読み取りやすいことからここでは Dropbox-API を例示したが，【事実 5】と【事実 6】は，Dropbox-API 以外の調査対象のライブラリにおいても確認できている．

4.4 最終更新日に対するバージョンの分布

本節では最終更新日に関する仮説 4 の検証を行う．最終更新日の新しさとバージョンの新しさとの間の相関係数は 0.89 となった．これは仮説 4 を支持する．

例として Apache Cordova を利用するアプリについて，各アプリの最終更新月に対する Apache Cordova のバージョンの分布を**図 6** のヒートマップに示す．Google はこれまでに Apache Cordova に関して報告されている脆弱性のうち，CVE-2015-5256，CVE-2015-1835，CVE-2014-3502 の 3 つの脆弱性に関し，アプリ開発者に向けて 2 回のセキュリティアラートを発信している．v3.5.1 と v4.1.1 はそれらのセキュリティアラートに該当する脆弱性の修正を含んだバージョンであることから，開発者が積極的にアップデートを行ったものと考えられる．また，Google は最新のセキュリティアラートにおいて，v4.1.1 未満の Apache Cordova を利用するアプリについては，上記の 3 つの脆弱性のいずれかの影響を受ける可能性があるとして，2016 年 7 月以降の新規公開およびアップデートを禁止しているが，Apache Cordova の公式ホームページでは，この脆弱性に対して，v5.1.x にアップデートすることを推奨している．このことから v5.1.1 の分布が高くなったと考えられる．v6.1.2 については，当該バージョンを利用するアプリの内訳を調査したところ，合計 73 個のアプリのうち，61 個が同一の開発者によって作成されたものであることが分かった．これらのアプリは開発者によっていっせいに v6.1.2 にアップデートされたか，v6.1.2 を利用する同一の開発環境で短期間のうちに新たに複数作成されたものと考えられる．

図 6 には，脆弱性の公開月，および影響を受けるバージョンを赤線で，Google から発信された 2 回のセキュリティアラートの発信月，および許容されたバージョンを緑線でそれぞれ表している．最終更新日が Google によるセキュリティアラートが発信されたあとの月であるものを境に脆弱性に対応したバージョンのアプリが増加していることが読み取れる．しかしながら，以下の事実があげられる．

【事実 7】 最終更新日が脆弱性の報告のあとにもかかわらずその脆弱性の影響を受けるバージョンを利用しつづけているアプリが存在する．

また，Apache Cordova に限っては，2016 年 7 月以降に更新されたアプリの中にも脆弱性の影響を受けるバージョンを利用しているアプリも存在している．Google Play にアップデートされたアプリは，必ず承認前に潜在的なセキュリティ上の問題も含めた安全性のチェックを受けるが [21]，これらのアプリは何らかの原因により，そのようなセキュリティチェックから漏れたと考えられる．

4.5 考察

予備調査および本調査の結果から考察する．表 1 に示したように，AdColony と inmobi，Tapjoy など，脆弱性の修正されていないライブラリがマーケットに広く分布しており，依然として多くのアプリに利用されていることが分かった．これらのライブラリについては，アプリの開発者がいかに利用するライブラリのバージョンを更新したとしても脆弱性がマーケットに残留することを示している．ライブラリの開発者には速やかに脆弱性を修正することが求められる．

表 2 に示したように，ランキング順位とバージョンの新しさの間の相関係数は低い．ランキングの順位はアプリごとに値の変化の激しい項目であるため，その影響を受けた可能性がある．また，累計ダウンロード数とバージョンの新しさとの間には高い相関係数が得られた．累計ダウンロード数が大きいことは，継続してユーザを獲得していることを意味する．結果的に，開発者はアプリを長期間保守することになり，定期的な更新が行われる傾向が表れたものと考えられる．同様に，ユーザレビューの値とバージョンの新しさとの間にも高い相関係数が得られた．ユーザレビューの値は，悪質なアプリ開発者や一部のユーザによって意図的に操作されることへの懸念から，評価値としては信頼性が低いと考えられることがあるが，レビューの母数が大きいアプリについては更新や管理がなされているかの指標となることが期待される．

調査対象のライブラリ全体において，各メタデータの値によらず，古いバージョンや脆弱性の報告されているバー

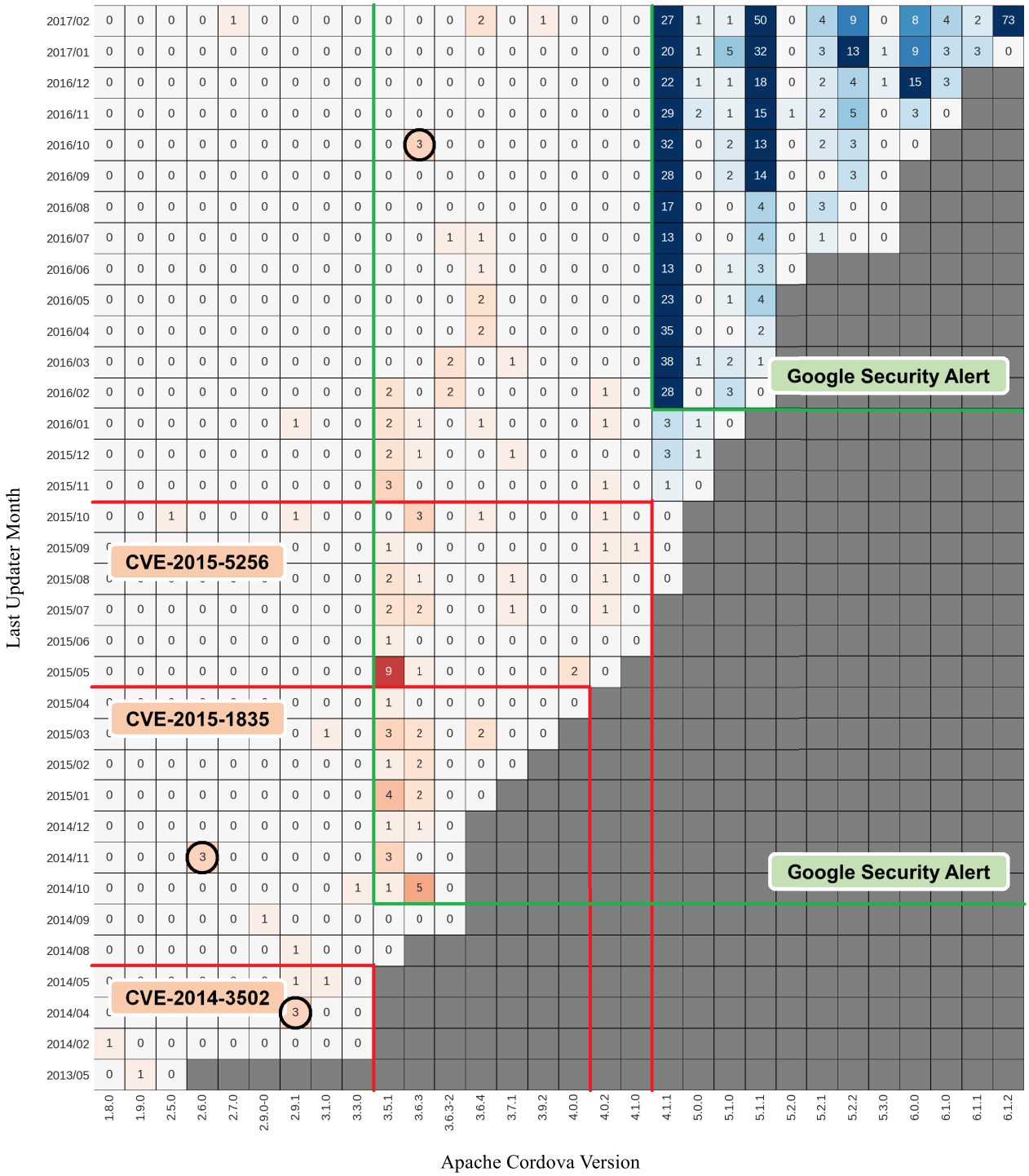

図 6 当該アプリの最終更新日に対する Apache Cordova のバージョンの分布と既知の脆弱性の報告月・影響範囲および Google によるセキュリティアラートの発信月・許容範囲

Fig. 6 Distribution of AdColony versions vs. last updated of each Apps and month of disclosing vulnerability with its impact range, month of sending security alert by Google with its tolerance range.

ジョンを利用するアプリが多く見られた．マーケット内のどのようなアプリをダウンロードしたとしても，そのアプリが脆弱性を持つライブラリを利用している可能性がある．

5. 追加調査：開発者との関連

4.4 節の図 6 のヒートマップにおいて，脆弱性が修正された直後のバージョンに縦縞型に分布が集中する傾向が見受けられた．しかし，それらの分布に追従せず，セキュリ

表 3 Apache Cordova を利用するアプリのうち，Google のセキュリティアラートに追従していない 3 グループ（2014/4-v2.9.1，2014/11-v2.6.0，2016/10-v3.6.3）の詳細情報

Table 3 Detailed information of 3 groups in Apps using Apache Cordova (2014/4 - V 2.9.1, 2014/11 - V 2.6.0, 2016/10 - V 3.6.3) that are not following the security alert by Google.

最終更新日	利用バージョン	開発者	カテゴリ	カテゴリ内順位	累計ダウンロード	ユーザーレビュー
2014/ 4/26	2.9.1	A（個人開発者）	COMICS	201-250	10000+	3.00-3.25
2014/ 4/26	2.9.1	A（個人開発者）	ENTERTAINMENT	351-400	10000+	3.50-3.75
2014/ 4/26	2.9.1	A（個人開発者）	COMICS	201-250	1000+	3.50-3.75
2014/11/18	2.6.0	B（某都市銀行）	FINANCE	251-300	5000+	3.75-4.00
2014/11/18	2.6.0	B（某都市銀行）	FINANCE	51-100	50000+	3.00-3.25
2014/11/18	2.6.0	B（某都市銀行）	FINANCE	101-150	10000+	3.00-3.25
2016/10/ 3	3.6.3	C（某事業連合）	SHOPPING	51-100	10000+	3.25-3.50
2016/10/ 3	3.6.3	C（某事業連合）	SHOPPING	101-150	10000+	2.25-2.50
2016/10/ 3	3.6.3	C（某事業連合）	LIFESTYLE	301-350	10000+	3.00-3.25

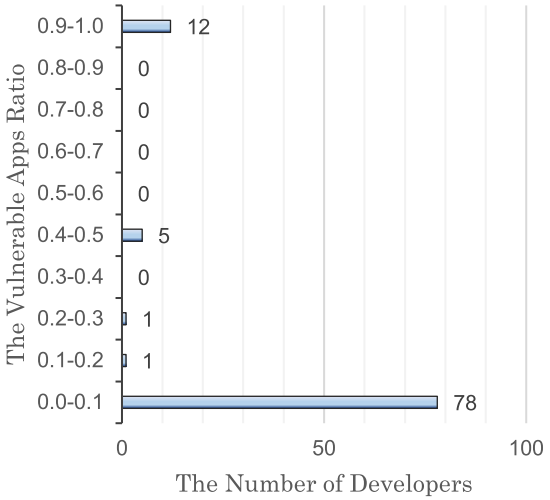

図 7 同一の開発者が Apache Cordova を利用して作成したアプリのうち脆弱性を持つバージョンを利用している割合の分布

Fig. 7 Distribution of the vulnerable Apps ratio among those created by a same developer with Apache Cordova.

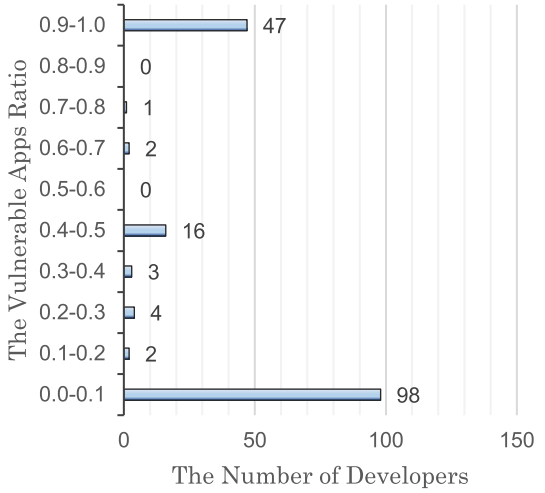

図 8 同一の開発者が libpng を利用して作成したアプリのうち脆弱性を持つバージョンを利用している割合の分布

Fig. 8 Distribution of the vulnerable Apps ratio among those created by a same developer with libpng.

ティアラートが発信された後の特定の月に更新され，同一の脆弱性を持つバージョンを利用しているアプリが数カ所に存在している．図 6 では，それらのアプリ群の位置を黒色の丸枠で示している．

この例にあてはまるものとして，2014 年 4 月に更新され，Apache Cordova の v2.9.1 を利用している 3 個のアプリ，2014 年 11 月に更新され，v2.6.0 を利用している 3 個のアプリ，2016 年 10 月に更新され，v3.6.3 を利用している 3 個のアプリについて，その内訳を調査した．**表 3** に示すとおり，これら 3 グループのアプリは，それぞれ同一の開発者によって，同一日に更新されていた．このことから，セキュリティアラートが報告された後でも，一部のアプリ開発者はライブラリ，フレームワークを更新せずに脆弱なコードを再利用している，ということが推測される．

そこで，Apache Cordova を利用して 2 個以上のアプリを作成している開発者について，それぞれが Apache Cordova を用いて作成しているアプリのうち，どれだけの割合で脆弱性の報告されているバージョンの Apache Cordova が利用されているかを調査した．その結果を**図 7** の分布図に示す．データセットに含まれたアプリの開発者計 9,359 名のうち，Apache Cordova を利用している開発者の合計は 480 名，脆弱性を持つバージョンを 1 つでも利用している開発者の数は 88 名であった．そのうち Apache Cordova を利用して 2 個以上のアプリを作成している開発者は合計で 97 名おり，脆弱性を持つバージョンを 1 つでも利用している開発者の数は 19 名であった．同様の調査を libpng に行った結果を**図 8** の分布図に示す．

Apache Cordova について，最も多いもので 61 個，平均

で 4.4 個のアプリを同一の開発者が作成していた．0.0-0.1 と 0.9-1.0 の分布が比較的大きくなっている．このことから脆弱性を持つバージョンの Apache Cordova を利用するアプリは，特定の開発者に偏って作成されていることが予想される．libpng についても Apache Cordova と同様の傾向が表れている．

これらから，同一の開発者によって作成されたアプリは，同一の外部ライブラリやフレームワークを利用して開発されており，次のことがいえる．

【事実 8】 あるアプリに脆弱性が見つかったとき，その開発者が作成した他のアプリも同様の脆弱性を有する場合がある．

6. 制約事項

6.1 調査対象アプリ

本研究では調査対象のマーケットとして公式の Google Play を選定し，日本から取得可能なアプリを収集することで，調査対象のデータセットを作成した．その他のサードパーティマーケットの調査は今後の課題とする．

また，マーケット内のすべてのアプリを調査することは現実的ではないため，本論文ではユーザによく利用されているアプリに限定して調査を行った．一般的にユーザは人気ランキングをもとにアプリのダウンロードを行うことから，各カテゴリの人気ランキングからダウンロードできるアプリをすべて収集した．しかしながら，人気ランキングは上位の一部のみが公開されているため，未収集部分が多く存在する．Watanabe らの調査 [9] によると，ランダムに取得したアプリは人気アプリに比べ，アップデート頻度が低い．したがって，本調査結果と傾向が異なる可能性はあるが，未収集部分のアプリは人気ランキングに掲載されているアプリ群よりも古いライブラリバージョンの利用率が高いことが予想される．

本調査におけるバージョン特定方法の精度は，調査対象アプリ群の APK ファイル内に，調査対象のライブラリのコードがどれだけ残っているかに依存する．具体的には，ProGuard [18] や DashO [22] のようなコード最適化ツールや難読化ツールによって，不要なコードが削除されたアプリや文字列データが暗号化されたアプリの場合，APK ファイルからライブラリの検出に必要なコード部分が消失している可能性があり，そのようなアプリについては，バージョン番号を特定することができない．

6.2 調査対象ライブラリ

本調査の目的は，ライブラリに起因する脆弱性の分析である．そのため過去に脆弱性が報告されているライブラリを調査対象とした．ただし，すべてのライブラリを調べることはできないため，まずはマーケットへの影響が大きいライブラリを優先して調査した．そこで 2 章で述べたように，AppBrain が公開している利用率ランキングをもとにライブラリの選定を行った．また本論文で用いたバージョン特定法は，3.2 節で述べたとおり，すべてのライブラリに対して有効なわけではない．したがって，シグネチャが作成できなかったライブラリについては除外した．

4.2 節の AdColony の調査結果のように，同一のライブラリであってもバージョンが異なればコードの改変によってシグネチャにマッチする箇所が消失する可能性がある．あるライブラリのすべてのバージョンを正確に検出するためには，それぞれのバージョンについて，作成したシグネチャが有効であるかを確認し，必要に応じてシグネチャを調整することとなる．

7. 結論

外部ライブラリに起因する脆弱性は複数のアプリに繰り返し作り込まれることから影響が大きく危険である．本調査の目的は，既知の脆弱性がマーケット内にいかに残留，分布しているかを調査し，確認することであった．実際に Google Play 内のアプリを調査したところランキング上位のアプリを含む多くのアプリに脆弱性が拡散している状況とそれに基づく以下の事実が確認できた．

【事実 1】 ランキング順位にかかわらずバージョンの古いものと新しいものが混在している．

【事実 2】 ランキングの順位が高いアプリであっても新しいバージョンのライブラリを使っていないことがある．

【事実 3】 累計ダウンロード数の多いアプリでも新しいバージョンのライブラリを使っていないことがある．

【事実 4】 全体としてはダウンロード数の多いアプリほど新しいバージョンが利用されている．

【事実 5】 ユーザレビューが高評価のアプリでも新しいバージョンのライブラリを使っていないことがある．

【事実 6】 全体としてはユーザレビューが高評価のアプリほど新しいバージョンが利用されている．

【事実 7】 最終更新日が脆弱性の報告のあとにもかかわらずその脆弱性の影響を受けるバージョンを利用しつづけているアプリが存在する．

【事実 8】 あるアプリに脆弱性が見つかったとき，その開発者が作成した他のアプリも同様の脆弱性を有する場合がある．

【事実 4】，【事実 6】は，それぞれ仮説 2，仮説 3 の正当性を裏付ける結果となったが，その他の事実からは以下のようなことが考えられる．

【事実 1】，【事実 2】，【事実 3】および【事実 5】より，ランキングの順位が高いアプリや累計ダウンロード数が多いアプリ，評価が高いアプリであってもセキュリティ上安全とは限らないことが分かった．現在の Google Play では，ユーザはアプリに関するセキュリティ情報を把握することができない．そのため，脆弱なアプリが利用され続けてし

まい，マーケットに残留するという悪循環を形成していると考えられる．対策として，マーケット管理者は，公開中のアプリが持つ脆弱性の指標を公表し，セキュリティリスクを認知させたうえでユーザにアプリを選択させる手段を提供するべきである．たとえば，アプリのパーミッション要求パターンを基に APK が持つリスクレベルを定量化する手法として DroidRisk [23] がある．同様にしてアプリが持つ脆弱性の個数や影響度から潜在的なリスクレベルを数値化することができれば，ユーザにセキュリティ上の評価基準を提供することが可能となる．

【事実 7】と【事実 8】はアプリ開発者に焦点を当てた対策が必要であることを示唆する．アプリ開発者が繰り返し脆弱性を作り込む要因とその対策を次の 2 つの場合に分けて考える．

（a）ライブラリの脆弱性が修正されない場合

ライブラリの開発者に脆弱性を修正するリソースが不足している可能性が考えられる．このとき，アプリ開発者はライブラリの脆弱性が修正されるまでの間アプリを非公開にするなど，ユーザが脆弱性の影響を受けないような対策をとるべきである．次に，ライブラリの開発者がセキュリティ上の問題を軽視しているために，脆弱性が放置されている可能性がある [24]．アプリの開発者は，同様な機能を持つ別のライブラリで代用するなどして，このようなライブラリの利用を避けるべきである．このとき，脆弱性が検知されていないライブラリを掲載するようなホワイトリストの作成と公開は有効であろう．アプリ開発者にとっては健全なライブラリを選択することができ，ホワイトリストに掲載されないライブラリの開発者に脆弱性の修正を促すこともできると考えられる．

（b）アプリがアップデートされない場合

(a) の場合と同様に，リソースの不足とセキュリティ問題の軽視が要因として考えられる．Watanabe ら [9] が述べているように，マーケット管理者は公開されているすべてのアプリに対して脆弱性調査を行うことが望まれる．そのうえで，脆弱性が検知されたアプリ開発者に対して警告を通知し，修正されないものについては非公開にするべきである．もう 1 つの要因として，開発者がアプリで利用しているライブラリの脆弱性情報を把握できていない可能性があげられる．マーケットを健全に保つためにも，マーケット管理者はライブラリの脆弱性情報を積極的に収集，検証すべきである．そして，脆弱性が発見された場合，マーケット管理者はアプリ開発者に対して迅速にアラートを通知すべきである．しかしながら，マーケット管理者がすべての脆弱性に対応を徹底することは困難であることが懸念される．そのような場合，アプリのコードや利用しているライブラリに含まれる脆弱性を評価・確認する第三者機関によるサービスの提供が有効であろう．そのようなサービスにおけるコード分析の相応のコストは，アプリの品質を保証し，結果的に開発者に還元されることと認知されるのが望ましい．詳細なコード分析が難しければ自動検査ツール（文献 [25], [26], [27] など）を統合して脆弱性の警告をするのでもセキュリティリスクの軽減が期待できる．

謝辞 本研究の一部は JSPS 科研費 16K00184 の助成を受けたものである．

参考文献

[1] 石井悠太，渡邉卓弥，秋山満昭，森 達哉：正規アプリに類似した Android アプリの実態解明，信学技報，Vol.114, No.489, ICSS2014-94, pp.187–192 (2015).
[2] 石井悠太，渡邉卓弥，秋山満昭，森 達哉：Android クローンアプリの大規模分析，コンピュータセキュリティシンポジウム 2015 論文集，Vol.2015, No.3, pp.207–214 (2015).
[3] 吉田奏絵，今井宏謙，芹沢奈々，森 達哉，金岡晃：Android アプリケーションにおける電子署名の大規模調査，コンピュータセキュリティシンポジウム 2016 論文集，Vol.2016, No.2, pp.480–487 (2016).
[4] 孫 博，渡邉卓弥，秋山満昭，森 達哉：Android アプリストアにおける不自然なレーティング・レビューの解析，コンピュータセキュリティシンポジウム 2015 論文集 (2015).
[5] 孫 博，秋山満昭，森 達哉：モバイルアプリストアにおけるプロモーショナル攻撃の自動検知システム，コンピュータセキュリティシンポジウム 2016 論文集, Vol.2016, No.2, pp.1040–1047 (2016).
[6] 渡邉卓弥，秋山満昭，酒井哲也，鷲崎弘宜，森達哉：Android アプリの説明文とプライバシー情報アクセスの相関分析，コンピュータセキュリティシンポジウム 2014 論文集 (2014).
[7] 石井悠太, 渡邉卓弥, 金井文宏, 高田雄太, 塩治榮太朗, 秋山満昭，八木 毅，森 達哉：Android サードパーティーマーケットの大規模調査，暗号と情報セキュリティシンポジウム (2017).
[8] Vulnerability Note VU#582497 – Multiple Android applications fail to properly validate SSL certificates, available from ⟨http://www.kb.cert.org/vuls/id/582497⟩ (accessed 2017-03-13).
[9] Watanabe, T., Akiyama, M., Kanei, F., Shioji, E., Takata, Y., Sun, B., Ishii, Y., Shibahara, T., Yagi, T. and Mori, T.: Understanding the Origins of Mobile App Vulnerabilities: A Large-scale Measurement Study of Free and Paid Apps, *Proc. IEEE/ACM 14th International Conference on Mining Software Repositories* (*MSR 2017*) (May 2017).
[10] Backes, M. et al.: Reliable third-party library detection in android and its security applications, *Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp.356–367, ACM (2016).
[11] Facebook SDK Vulnerability Allows Account Hijacking – Cyber Kendra - Latest Hacking News and Tech News, available from ⟨http://www.cyberkendra.com/2014/07/facebook-sdk-vulnerability-allows.html⟩ (accessed 2017-03-13).
[12] Android library statistics - AppBrain, available from ⟨https://www.appbrain.com/stats/libraries⟩ (accessed 2017-03-13).
[13] Google Play Unofficial Python API, available from ⟨https://github.com/egirault/googleplay-api⟩ (accessed 2017-03-13).
[14] Narayanan, A. et al.: Addetect: Automated detec-

tion of android ad libraries using semantic analysis, *2014 IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing* (*ISSNIP*), pp.1–6, IEEE (2014).
[15] Ma, Z. et al.: Libradar: Fast and accurate detection of third-party libraries in android apps, *Proc. 38th International Conference on Software Engineering Companion*, pp.653–656, ACM (2016).
[16] Apktool, available from ⟨https://ibotpeaches.github.io/Apktool⟩ (accessed 2017-03-13).
[17] dex2jar, available from ⟨https://github.com/pxb1988/dex2jar⟩ (accessed 2017-03-13).
[18] GuardSquare: Proguard java obfuscator, available from ⟨http://proguard.sourceforge.net⟩ (accessed 2017-03-13).
[19] Kendall, M.G.: A new measure of rank correlation. Biometrika, 30.1/2, pp.81–93 (1938).
[20] Hardt, D.: The OAuth 2.0 Authorization Framework, RFC 6749 (Proposed Standard) (Oct. 2012).
[21] App Security Improvement Program, available from ⟨https://developer.android.com/google/play/asi.html⟩ (accessed 2017-06-14).
[22] PreEmptive Solutions: Dasho java obfuscator, available from ⟨http://www.preemptive.com/products/dasho⟩ (accessed 2017-06-16).
[23] Wang, Y., Zheng, J., Sun, C., et al.: Quantitative security risk assessment of android permissions and applications, *DBSec* (2013).
[24] More Bad News for Mobile App Security – ADTmag, available from ⟨https://adtmag.com/articles/2015/03/23/more-bad-news-for-mobile-app-security.aspx⟩ (accessed 2017-06-16).
[25] AndroBugs, available from ⟨https://github.com/AndroBugs/⟩ (accessed 2017-06-16).
[26] Mallodroid, https://github.com/sfahl/mallodroid (accessed 2017-06-16).
[27] LinkedIn: QARK, available from ⟨https://github.com/linkedin/qark⟩ (accessed 2017-06-16).

古川 凌也

PwC サイバーサービス合同会社サイバーセキュリティ研究所研究員．2015 年神戸大学工学部電気電子工学科卒業．2017 年同大学大学院博士課程前期課程修了．2017 年より PwC．マルウェア解析や Android の脆弱性対策に興味を持つ．

永井 達也 （学生会員）

2016 年神戸大学工学部電気電子工学科卒業．現在，同大学大学院博士課程前期課程在学中．不正な Web サイトの識別・検知等ネットワークセキュリティと機械学習に興味を持つ．

熊谷 裕志

PwC サイバーサービス合同会社サイバーセキュリティ研究所上席研究員．2011 年より一般社団法人 JPCERT コーディネーションセンターにて脆弱性情報の解析やセキュアコーディングの普及啓発活動に従事．コードレベルでの脆弱性の原因調査や対策コードのアドバイス，セキュアなアプリを作成するためのセキュアコーディングガイドの作成に携わる．その後，PwC サイバーサービス合同会社にてサービスメニューの企画立案や株式会社イエラエセキュリティにて研究開発チームをリード．Web アプリケーションおよび Android における脆弱性対策について精通している．

神薗 雅紀

PwC サイバーサービス合同会社サイバーセキュリティ研究所所長．大学時代に国立研究開発法人情報通信研究機構（NICT）nicter システムの研究開発に従事する．NICT, JPCERT/CC, IPA，某省・民間のセキュリティに関する研究開発・コンサルティングに従事し，セキュリティに関する多数の国家プロジェクトの研究員およびプロジェクトマネージャーを経験する．同時に，マルウェアの解析作業やインシデントが発生した際のレスポンス対応を行う．また，マルウェア対策研究人材育成ワークショップでは不正な Web サイトの検知・解析および標的型攻撃に関わる論文にて 2 年連続優秀論文賞を受賞．AVAR 等の国内外のセキュリティカンファレンスにて研究発表も行っている．2015 年より PwC に入社し，PwC サイバーサービス合同会社の設立メンバとして，新たなサイバーセキュリティサービス/インテリジェンスサービスを開発．サイバーセキュリティ研究所を率いて新たなコア技術の研究開発や，サイバー攻撃の分析に従事．

白石 善明 （正会員）

1995 年愛媛大学工学部情報工学科卒業．1997 年同大学大学院博士前期課程修了．2000 年徳島大学大学院博士後期課程修了．博士（工学）．2002 年近畿大学理工学部情報学科講師．2006 年名古屋工業大学大学院情報工学専攻助教授．2013 年神戸大学大学院電気電子工学専攻准教授．情報セキュリティ，コンピュータネットワーク，教育支援，知識流通支援等の研究・教育に従事．2002 年電子情報通信学会オフィスシステム研究賞，2003 年暗号と情報セキュリティシンポジウム（SCIS）20 周年記念賞，2006 年 SCIS 論文賞．2007，2008，2011，2013 年 DICOMO 優秀論文賞．2012 年電子情報通信学会ライフインテリジェンスとオフィス情報システム研究会功労賞．2015 年本会高度交通システム研究会優秀論文賞．2017 年電子情報通信学会関西支部活動功労賞．2017 年より電子情報通信学会情報通信システムセキュリティ研究専門委員会委員長．電子情報通信学会シニア会員．

高野 泰洋

2016 年北陸先端科学技術大学院大学およびフィンランド・オウル大学博士後期課程修了．博士（情報科学，工学）．同年神戸大学大学院工学研究科助教．無線通信における信号処理等の研究・教育に従事．電子情報通信学会会員．

毛利 公美

1993 年愛媛大学工学部情報工学科卒業．1995 年同大学大学院工学研究科情報工学専攻博士前期課程修了．2002 年博士（工学）（徳島大学）．1995 年香川短期大学助手．1998 年徳島大学工学部知能情報工学科助手，2003 年同講師．2007 年岐阜大学総合情報メディアセンター准教授，2017 年同大学工学部電気電子・情報工学科准教授．ネットワークセキュリティ，符号・暗号理論，コンピュータネットワーク等の研究・教育に従事．電子情報通信学会シニア会員．

星澤 裕二

PwC サイバーサービス合同会社最高執行責任者．1998 年に株式会社シマンテックに入社以来，同社のインターネットセキュリティ研究所のマネージャとして，セキュリティの研究や新種マルウェアへの対応，脆弱性情報の収集・分析等を担当し，国内におけるマルウェア研究における地位を確立．2004 年 10 月，株式会社セキュアブレインの設立に参画．最高技術責任者として設立当初よりコア技術開発に従事．サイバーセキュリティに関して多くの IT 関連出版物に寄稿．また，Black Hat，Virus Bulletin，EICAR（European Expert Group for IT-Security），AVAR（Association of anti Virus Asia Researchers）等の国際会議でセキュリティ問題に関する研究発表も行っている．総務省，NISC（内閣サイバーセキュリティセンター），IPA（情報処理推進機構），情報処理学会等において多くの研究会委員，有識者会議メンバ，座長等を務める．2007 年度の情報化月間における情報化促進貢献個人表彰の中で，経済産業省商務情報政局長表彰「情報セキュリティ促進部門」を受賞．

森井 昌克

1989 年大阪大学大学院工学研究科通信工学専攻博士課程修了．工学博士．同年京都工芸繊維大学工芸学部電子情報工学科助手，1990 年愛媛大学工学部情報工学科講師，1992 年同助教授，1995 年徳島大学工学部知能情報工学科教授，2005 年神戸大学工学部電気電子工学科教授．現在，同大学大学院工学研究科教授．主に代数的符号理論，離散数学，ディジタル信号処理アルゴリズム，情報セキュリティおよびコンピュータネットワーク等の研究・教育に従事．電子情報通信学会情報セキュリティ研究専門委員会，同ライフインテリジェンスとオフィス情報システム研究専門委員会，同情報通信システムセキュリティ研究専門委員会，各委員長を歴任．2010 年度 FIT 船井ベストペーパー賞，2015 年度電子情報通信学会論文賞等．電子情報通信学会シニア会員．