

Android アプリケーションにおける IP アドレスハードコーディングの大規模調査

金岡 晃^{1,a)} 小林 裕¹ 岡田 雅之²

概要： IP を巡る環境は大きく変わりつつあり、IPv6 への移行は本格化しつつある。エンドユーザが利用する環境においても、IPv4 を利用せず IPv6 だけを利用するシングルスタック環境の利用も視野にいれなければならない。エンドユーザ環境で利用される OS が IPv6 対応がされていても、アプリケーションが IPv6 に対応していない、あるいは IPv6 シングルスタック環境に対応していない場合、アプリケーションが誤作動を起こす可能性がある。さらには、その状況を利用して悪意のある第 3 者が中間者攻撃をする可能性がある。IPv6 シングルスタック未対応アプリケーションの原因の 1 つに、アプリケーションにおける IPv4 アドレスのハードコーディングがあると考えられる。IPv4 アドレスがアプリケーションのソースコードに直接記載されていてそれが変更できない場合、IPv6 シングルスタック環境では不具合が起こる可能性がある。そういった問題がある一方で、アプリケーションにおいて IPv4 アドレスがどれほどハードコーディングされているかの大規模かつ詳細な調査は行われていない。そこで本研究では Android アプリケーションを対象に初めて大規模に IPv4 アドレスのハードコーディング状況を調査する。調査結果によりハードコーディングの発生率が明らかになり、利用されている IP アドレスの特徴やその用途などが分析され明らかになった。

キーワード： IPv6、IP アドレス、Android アプリケーション

AKIRA KANAOKA^{1,a)} YU KOBAYASHI¹ MASAYUKI OKADA²

Abstract: The environment surrounding IP is changing drastically, and the transition to IPv6 is in full swing. In the end-user environment, it is necessary to consider the use of a single stack environment that uses only IPv6 instead of IPv4. Even if the OS used in the end-user environment supports IPv6, if the application does not support IPv6 or does not support IPv6 single-stack environment, the application will misbehave. Furthermore, there is a possibility that a malicious third party may take advantage of the situation to conduct MITM attacks. One of the causes of IPv6 single stack unsupported applications is the hard coding of IPv4 addresses in applications. There has been no large-scale and detailed study on how much IPv4 addresses are hard-coded in applications. This study is the first large-scale investigation of IPv4 address hardcoding in Android applications. The results of the study will reveal the incidence of hardcoding, and analyze and discuss the characteristics of IP addresses in use and their applications.

Keywords: IPv6, IP Address, Android Application

¹ 東邦大学
Toho University, Funabashi, Chiba 274-8510, Japan (小林の所属は研究当時)

² 長崎県立大学
University of Nagasaki, 1-1-1 Manabino, Nagayo-cho, Nishi-Sonogi-gun, Nagasaki 851-2195, Japan

a) akira.kanaoka@is.sci.toho-u.ac.jp

1. はじめに

インターネット通信の中心的なプロトコルである IP は、現在でも IP Version 4 (以後 IPv4) が広く利用されているが、インターネットの利用者が増え続けたことにより IPv4

アドレスが枯渇していることが問題になっている。IPv4 アドレスは 32 ビットのビット長を持ち、約 43 億個のアドレスを表現することができる。この数はインターネット黎明期の頃は十分な大きさと考えられていたが、インターネットに接続される機器が増え続けたことにより、割当の限界を迎えつつある。そのためアドレス空間を IPv4 から大幅に拡張するなど対策が取られた IP Version6 (以後 IPv6) の導入がされ、IPv4 から IPv6 への移行が進んでいる。

IPv4 アドレスと IPv6 アドレスの間には互換性がなく、相互通信を行うことが出来ない。そのため端末などのプラットフォームが IPv4 と IPv6 の双方に対応できるようにし、同じくネットワークも IPv4 と IPv6 を同時に提供できるようにし、IPv6 未対応な端末などがなくなった後の IPv6 環境に完全移行するアプローチがとられている。こういった混在環境を経て、いずれ IPv6 だけの環境 (IPv6 シングルスタック環境) へと移行していく。モバイル環境においては、Apple の iPhone や iPad に提供されるアプリケーションでは IPv6 に対応することが必須とされている [1]。一方、Android OS 上ではそういった要件は示されていない。すでに IPv6 シングルスタック環境からのアクセスにおいて、正しく動作しない Android アプリが複数発見されている。過去 2012 年には IETF にて IPv6 Only Network についての情報共有がなされ [2]、2018 年には北口らによって OS 各種の IPv6 対応状況調査が行われ [3]、その後、加茂によって Android に焦点を当て v6 のみの環境で OS が稼働するか、マーケットが対応しているか、Android アプリケーションが動くかという複数の視点で行われた [6]。その結果、古い OS での IPv6 未対応状況、マーケットの IPv6 未対応状況、アプリの大部分が IPv6 に対応していないことが分かった。しかし、Android アプリケーションが未対応である原因についてはまだ調査がされていなかった。Android アプリケーションが IPv6 に対応していない原因を考えた場合、IPv4 と IPv6 の混在環境や IPv6 シングルスタック環境環境で動作するよう OS やネットワークは対応されているため、アプリケーション側で特定の IP バージョンが指定されていることから IPv6 シングルスタック環境で動かないことが 1 つの原因として考えられる。IP アドレスが IPv4 であることを前提として指定されていた場合、IPv6 シングルスタック環境ではそのアプリケーションが動作しないだけでなく、利用者が悪意のある攻撃者によって Man in the middle(中間者攻撃) を受けるリスクも存在する。

そこで本研究では Android アプリケーションを対象に初めて大規模に IPv4 アドレスのハードコーディング状況を調査する。調査結果によりハードコーディングの発生率が明らかにし、利用されている IP アドレスの特徴やその用途などが分析され明らかにする。

2. 関連研究

Android の OS、ライブラリ、アプリといった Android の環境全体にフォーカスを当てた IPv6 対応調査は著者らの知る限り行われてはいないが、その他の視点での調査がいくつか存在する。ここではその調査を紹介する。

北口らは、IPv6 対応状況の研究として OS の視点から調査を行った [3]。そこでは各種 OS における IPv6 実装状況の検証がされ、さらにネットワーク運用管理に与える影響について考察を行っていた。Android OS については、Version 4, 5, 6, 7 を対象に調査が行われており、すべてのバージョンで DHCPv6 に未対応であることと、Version 4 では IPv6 だけの環境では動作しないことが示されている。また運用管理の課題としては、セキュリティインシデント時のトレーサビリティ困難性が挙げられている。

我々の研究目的の 1 つは、IPv6 シングルスタック環境において OS 等の環境が未対応であった場合の脅威考察がある。加茂らは IPv6 シングルスタック環境において代表的な Android アプリケーションが動作するかを実証実験した [6]。そこでは人気アプリ 100 個に対してシングルスタック環境において動作を試みたものの、97 のアプリに対して問題なく動いたアプリは 10 個に過ぎず、それ以外のアプリについては起動直後にエラーになるケースや一定作業中にエラーが発生するケースなど、不具合が多く起こることが明らかになった。

さらに別の視点で IPv6 のセキュリティ脅威について議論した研究がいくつか存在する。Durdagi らは IPv6 と IPv4 のセキュリティと脅威の比較調査を行った [4]。しかし、技術仕様の視点からの指摘はされているが、運用の視点や混在環境での脅威といった視点では議論は行われていない。Rosli らは IPv6 環境の脅威に焦点をあて、セキュリティ対策の提案をした [5]。しかし、IPv4 と IPv6 の混在環境についての言及はあるが、脅威に関しては技術仕様の視点のみであり、運用の視点や可用性の視点の提案はなされていない。

3. IPv6 シングルスタック環境において考える脅威

IPv6 シングルスタック環境では、IPv4 は利用されない。現在のインターネット接続サービスを提供するプロバイダは IPv6 に対応しているケースが多く、IPv6 だけでインターネットへ接続することは可能となっている。また Windows 等の汎用の OS も OS 自体が IPv6 に対応し、IPv6 だけで動作することが可能な状態になっている。ネットワークと OS が IPv6 対応を進めている一方で、アプリケーションは対応しているとはいいがたい。先述したように加茂らの調査により動作に不具合を生じるアプリケーションが多いこ

とが示されている。エンドユーザの視点に立つと、ユーザが利用するアプリケーションが IPv6 シングルスタック環境において不具合を生じた場合、たとえ OS やネットワークが IPv6 シングルスタックに対応していようとユーザは対応しているとは感じなくなってしまう。

アプリケーションが IPv6 シングルスタック環境に対応せずに動作しないだけであれば、まだ問題は少ないかもしれない。より大きな問題となるのは、一部分だけが動作せず、一見すると正確に動作していると思えるような状態である。この場合、なんらかの情報がサーバ等のサービス事業者に送られないことが考えられるために、ユーザや事業者が望んだコミュニケーションが取れなくなり、齟齬が起きる可能性がある。これが例えば決済等に用いられる場合には金銭的被害につながったり、なんらかの申請等に用いられる場合には申請が受理されなかったりという問題が起る可能性がある。

第 3 者による悪意のある中間者攻撃の可能性も考えられる。IPv6 シングルスタック環境に端末がある場合、エンドユーザがその環境下にいることを認識する可能性は低い。そして OS 等のプラットフォームは、その環境下にあつたに IPv4 の通信が提供された場合、IPv6 と同時に利用することが十分に考えられる。OS 等によっては IPv4 が優先される可能性もある。そういった環境下において、悪意のある第 3 者が IPv4 の無線 LAN ルータの設置や IPv4 アドレスの DHCP サーバをネットワーク内に設置した場合、エンドユーザが気づかないまま IPv4 アドレスがユーザ端末に付与され、悪意のある第 3 者が設置した機器を介してインターネット接続が開始される可能性がある。この場合、通信の内容によっては通信内容の改ざんやアクセス先の操作による偽サイトへの誘導が可能になる恐れがある。

このように、IPv4 から IPv6 への移行過渡期である現在においては、IPv6 シングルスタック環境においてエンドユーザに脅威が存在することがわかる。それらの脅威の回避には、アプリケーション側の適切な IP 通信設定や、ネットワーク管理の厳格化が求められることとなる。

4. IPv6 シングルスタック未対応アプリの原因考察

前章で考察した IPv6 シングルスタック環境における脅威は、アプリケーションが IPv6 シングルスタックに対応することで回避できる部分がある。しかし加茂らの調査の結果は、アプリケーションがまだ完全な IPv6 シングルスタック環境対応には及んでいないことが示されている。この章ではアプリケーションが IPv6 シングルスタック環境に対応していな部分の可能性を考察する。

考察にあたって、アプリケーションが利用される環境においてインターネット接続されるネットワークとアプリケーションが動作する OS 等のプラットフォームは IPv6

シングルスタック環境に問題なく対応しているものと仮定する。

IPv6 シングルスタック環境に未対応である原因は、サーバサイドとクライアントサイドに大別される。双方ともに IPv6 シングルスタックに対応していることで初めてそのサービスが IPv6 のみで成立することとなる。サーバサイドのソフトウェア実装とクライアントサイドのアプリケーション実装の双方が同時に IPv6 シングルスタック対応することが求められる。その場合、サービス独自のプログラミングだけでなくサービスに利用されるソフトウェアライブラリや Web を介した API 等の IPv6 シングルスタック対応も必要となる。これらすべてがそろふことでようやくサービスの IPv6 シングルスタック対応が達成される。

サービス独自のプログラミングにおいて IPv6 シングルスタック未対応となるケースの 1 つに、IPv4 アドレスのハードコーディングがある。つまりサーバサイドのソフトウェアやクライアントサイドのアプリケーションのソースコードに直接 IPv4 アドレスが指定され、それが固定されているために IPv6 環境において適切な IP アドレス設定がされず通信が行えなくなってしまうものである。IP アドレスを直接ソフトウェアに指定することは適切ではなく、たとえば FQDN での指定をし通信先の IP アドレスの取得を OS 等のプラットフォームを通じて DNS サーバから取得することで IP のバージョンにかかわらずに通信を実現することが可能になる。しかし開発においてそういったネットワーク環境の利用を想定していない場合は、ハードコードにより直接サーバホストの IP アドレスを記載するなどが行われる。本研究ではこの IP アドレスのハードコーディングに注目した。

5. IP アドレスハードコーディング状況の調査方法

5.1 Android アプリケーション

本研究では IP アドレスのハードコーディング状況を調査する。ハードコーディングの状況を調査する方法は複数考えられる。たとえば Github 等のオープンソースで公開されているソースコードや、StackOverflow のコミュニティに展開されるコード断片 (スニペット) における記載状況を分析することが可能である。実際に開発に従事しているエンジニアに募集をかけ、そういった実装の経験やその原因、問題点の認識などをヒアリングすることも可能であろう。そういったユーザ実験の場合は、実際にソースコードを書いてもらいその内容について議論をする調査も可能である。

本研究では Android アプリケーションに注目した。Android アプリケーションは Kotlin 言語や Java 言語で開発され、ビルドされ APK ファイルとしてパッケージ化される。APK ファイルの実態は Zip ファイルであり、圧縮さ

```
(([1-9]?[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([1-9]?[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])
```

図 1 IPv4 アドレスの正規表現

れた中に実際に実行される dex ファイルが他のリソースファイル等とともに同梱されている。この dex ファイルは C/C++等の実行ファイルとは異なり、静的解析が容易なことが知られている。dex ファイルはソースコードに完全に復元するデコンパイルは困難であるものの、開発時にどういった情報が記載されていたかの情報を得ることが比較的容易になっている。

Android OS は世界中の多くのユーザで利用されており、数多くのアプリケーションが存在しているために分析の対象も広く採ることができる。また Androzoo のようにさまざまなアプリケーション配信マーケット上のアプリケーションを収集して公開しているデータセットも充実している。これらのことから、コーディングの実態を広く網羅的に調査できることができると考えられるため、本研究では Android アプリケーションに注目し、そこで IP アドレスのハードコーディングの状況を調査することとした。

5.2 Android アプリケーションの収集

多くのアプリケーションを分析するために、Androzoo のデータセットを利用する [7]。Androzoo は Android 公式の Google Play を中心にさまざまなマーケットより Android アプリケーションを収集し、その APK ファイルを公開している。2021 年 8 月現在では 1600 万を超える APK ファイルが利用可能となっている。本研究では、Androzoo データセットのうちランダムに選択した 492,979 のアプリケーションに対して調査を行う。Androzoo データセットよりそれぞれの APK ファイルをダウンロードし、実際の分析を行う。

5.3 APK ファイルにおける IP アドレスハードコーディングの抽出

APK ファイルは apktool を用いて展開することで、dex ファイルの機械語命令がソースコードの各クラスが smali ファイルとして出力される。その smali ファイルに直接 IP アドレスが記載されているかを判定することでハードコーディングの実施有無を判断する。

smali ファイルはテキスト形式であるため、IP アドレス記載の判定は正規表現による文字列一致検索を利用する。IP アドレスを示す正規表現は図 1、2 のものを利用した。

IPv4 アドレスに関しては、図 1 の正規表現だけでは誤検知が発生する。たとえば、OID のように数値が連続してドットで区切られて表現される場合は、OID の一部分が IPv4 アドレスとして判定されてしまう。またアプリケーション自体のバージョン番号指定に類似した記法がされて

```
(([0-9a-fA-F]{1,4}){7,7}|[0-9a-fA-F]{1,4})|([0-9a-fA-F]{1,4}){1,7}:(|([0-9a-fA-F]{1,4}){1,6}:|[0-9a-fA-F]{1,4})|([0-9a-fA-F]{1,4}){1,5}:(|[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}){1,3}:(|[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}){1,2}:(|[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}):(|([0-9a-fA-F]{1,4}){1,6}):(|([0-9a-fA-F]{1,4}){1,7}):)|fe80:(|[0-9a-fA-F]{0,4}){0,4}%[0-9a-zA-Z]{1,1}:|(ffff:0{1,4}){0,1}:(|0{1,4}|(25[0-5]|(2[0-4]|1[0-9])|0{1,1}){0-9})\.){3,3}(25[0-5]|(2[0-4]|1[0-9])|0{1,1}){0-9})|([0-9a-fA-F]{1,4}){1,4}:(|(25[0-5]|(2[0-4]|1[0-9])|0{1,1}){0-9})\.){3,3}(25[0-5]|(2[0-4]|1[0-9])|0{1,1}){0-9})
```

図 2 IPv6 アドレスの正規表現

いる場合にはその番号も IPv4 アドレスとして判定されてしまう。そこで本調査では、IPv4 アドレス判定をされた文字列の前後 1 文字ずつを確認し、それらが数値あるいはピリオドだった場合は IPv4 アドレスではないという追加の判定を入れることで OID を誤判定することを避けた。また IPv4 アドレスと判定された文字列と同一の行に”version”を含む場合はバージョン番号であるものとして判定から除外した。

OID の中には IP アドレスと全く同等の表現で表されるものがある。つまり、ドットが 3 つありドットを挟んだ数値が 4 つあり、それらの数値が 0 - 255 に収まるものである。たとえば X509 証明書の証明書拡張である authorityKeyIdentifier の OID は”2.5.29.1”となっている。Android アプリケーションでは電子署名や証明書を扱うものもあり、そういったデータを扱うために OID 指定をすることから OID としてこれらの数値をハードコードしているケースも散見されたため、これらの数値は IPv4 アドレスの判定から除外した。IPv4 アドレス判定において除外に用いた OID 情報については付録 A.1 に詳述する。

6. 調査結果

3 章で用いた手法により、Android アプリケーションにおける IP アドレスのハードコーディング状況を調査した。本章ではその結果を示す。先述の通り、分析したアプリケーションは 492,979 個であり、Androzoo のデータセットのリストよりランダムに抽出した。抽出時に用いた Androzoo のリストも付録 A.1 に記載する。

6.1 IPv4 アドレスをハードコードしたアプリケーション

APK ファイルを展開した smali ファイル群を正規表現による文字列一致と特定情報の除外により IPv4 アドレスが記載されたと判定されたアプリケーションは 184,354 であった。これは 1 つのアプリケーションに少なくとも 1 つ以上の IPv4 アドレスがあると判定されたアプリケーション数である。調査対象全体の 37.40%が IPv4 アドレスのハードコードをされていることがわかった。また延べ 24,980 種類の IP アドレスの出現が観測された。

表 1 に出現数上位の IP アドレスを記載する。最も多く出現した IP アドレスがローカルループバックアドレスの 127.0.0.1 であり、次に多いのが IPv4 アドレス空間全体、

表 1 IPv4 アドレスと判定された文字列のうち出現数の多いもの

アドレス	出現数	IP アドレスタイプ
127.0.0.1	217,494	ローカルループバック
0.0.0.0	127,043	インターネット全体
1.4.8.32	38,534	グローバル
10.0.2.2	31,233	プライベート
1,2.10.27	26,903	グローバル
1.4.7.32	19,377	グローバル
117.97.87.6	17,553	グローバル
10.0.1.1	15,761	プライベート
2.7.0.33	15,516	グローバル
1.4.1.19	14,541	グローバル

つまりインターネット全体を示す 0.0.0.0 であることがわかる。その他のアドレスを見ると、グローバル IP アドレスもあるが複数のプライベート IP アドレスがあることもわかる。こういったアドレスが記載される明確な理由は出現数からは判然とはしないものの、IPv4 アドレスはいくつかのアドレス帯が特殊な用途で用いられているため、次にそれら用途別での出現数を測ることとした。

6.2 IPv4 アドレスのタイプ別分析

IPv4 アドレスは、その用途により複数のアドレス帯に分割できる。まずは LAN 内などで自由に設定ができるプライベート IP アドレスである。これはクラスにより 3 つのアドレス帯があり、クラス A では 10.0.0.0~10.255.255.255、クラス B では 172.16.0.0~172.31.255.255、クラス C では 192.168.0.0~192.168.255.255 が設定されている。

クラス D のアドレス帯である 224.0.0.0~239.255.255.255 はホスト単体に用いられるアドレスではなくマルチキャスト通信に用いられるアドレスとなっている。クラス E のアドレス帯である 240.0.0.0~255.255.255.255 は実験用として予約されているアドレスであり、こちらもホストに用いられるアドレスではない。

127.0.0.0~127.0.0.254 のアドレス帯の IP アドレスはローカルループバックアドレスとして用いられるものであり、ホスト自身を指すアドレスとして利用される。169.254.0.0~169.254.255.255 はリンクローカルアドレスとして利用されるアドレス帯であり、DHCP サーバにより IP が自動的に割り当てられない場合にホスト自身が割り当て IP アドレスとして利用されている。

そしてその他のアドレスはいわゆるグローバル IP アドレスと呼ばれるものとなる。

ハードコードされた IP アドレスがこれらのどれに当たるかをさらに分析した。分析にあたり、まずそれぞれのタイプの IP アドレスが出現したアプリケーションがいくつかあるかを数えた。その結果を表 2 に示す。グローバル IP アドレスの出現が最も多く、次いでローカルループバックアドレスが多いことがわかる。プライベート IP アドレス

表 2 アドレスタイプごとのハードコードされたアプリケーション数

アドレスタイプ	アプリ数	
インターネット全体 (0.0.0.0)	45,412	(24.63%)
グローバル	120,394	(65.31%)
プライベート	42,567	(23.09%)
ローカルループバック	87,006	(47.20%)
リンクローカル	120	(0.07%)
クラス D	1,311	(0.71%)
クラス E	653	(0.35%)

表 3 アドレスタイプごとのハードコードされたアプリケーション数(他のアドレスタイプを含まないもの)

アドレスタイプ	アプリ数	
インターネット全体 (0.0.0.0)	3,514	(1.91%)
グローバル	63,068	(34.21%)
プライベート	14,081	(7.64%)
ローカルループバック	26,303	(14.27%)
リンクローカル	7	(0.00%)
クラス D	72	(0.04%)
クラス E	15	(0.01%)

が一定のアプリケーションにより使われているが、その用途は筆者の知る限りでは明確ではない。また少数ではあるがリンクローカルアドレスやクラス D、クラス E のアドレスが記載されているアプリケーションもあることがわかる。

次に、これらの IP アドレスのタイプにおいて、同一アプリケーション内で他の IP アドレス帯が出てこないようなアプリケーションがいくつかあるかを数えた。その結果を表 3 に示す。傾向は表 2 と同じであるが、それぞれの出現率は下がっている。表 3 に計上されていないが表 2 に計上されているアプリケーションは複数のタイプの IP アドレスがアプリケーション内にハードコードされていることを示している。特にインターネット全体を示す 0.0.0.0 の出現率が下がっている。このことから、0.0.0.0 がアプリケーション内で使用されるときは他の IP アドレスも同時に記載されているケースが多いことが伺える。

IPv6 シングルスタック環境での動作不具合の視点で見ると、ローカルループバックやリンクローカル、クラス D、クラス E のアドレスは影響が少ないと考えられるが、インターネット全体やグローバル IP、プライベート IP アドレスがハードコードされているアプリケーションはシングルスタック環境において通信とそれをもとにしたサービスに不具合が発生する可能性が存在する。たとえばグローバル IP アドレスだけがハードコードされたアプリケーションは全体の 34.21%と少なくなく、影響が広範にわたる可能性が示唆されている。

6.3 IPv6 アドレスをハードコードしたアプリケーション

IPv4 アドレスと同様に IPv6 アドレスもハードコードされている可能性はある。IPv4 アドレスと同様の調査を IPv6 アドレスに対しても行った。調査対象は同じデータ

表 4 IPv6 アドレスと判定された文字列のうち出現数の多いもの

アドレス	出現数
::	726,834
e::	16,532
ed::	4,170
2::	3,247
d::	2,839
ce::	2,674
E::	1,053
a::	1,001
de::	761
aded::	565

セットとした。

APK ファイルを展開した smali ファイル群を正規表現による文字列一致により IPv6 アドレスが記載されたと判定されたアプリケーションは 91,123 であった。これは 1 つのアプリケーションに少なくとも 1 つ以上の IPv6 アドレスがあると判定されたアプリケーション数である。調査対象全体の 18.48% が IPv4 アドレスのハードコードをされていることがわかった。また延べ 511 種類の IP アドレスの出現が観測された。

表 4 に出現数上位の IP アドレスを記載する。これらの評価は注意を要する。IPv6 アドレスの表記方法は自由度が高く、正規表現による文字列一致検索では本来は IPv6 アドレスではない記載も IPv6 アドレスとして検出される可能性は IPv4 アドレスより高い。実際に、最も多く出現したと判定された IPv6 アドレスは”::”であり、この表現は他のコードの中に出現する可能性は大きく、この出現数が実際の IPv6 アドレスの出現数であることは考えにくい。出現数上位 10 のアドレスはすべてにこの”::”を含んだアドレスとなっているなど、誤まって計上されている可能性は高い。実際に、出現回数は 726,834 回と調査対象のアプリケーション数の 1.47 倍となっており、2 番目に出現数が多い IPv6 アドレスとも大きな差がある。

そこで本研究では、これらの出現回数は参考として扱うことにして、その値の大小により状況を分析や考察することは行わないこととした。一方で、この正規表現によるフィルタリングは、IPv6 アドレス以外を計上することはあるが、IPv6 アドレスを計上しないことはないために、一致が発見されなかったアプリケーションは IPv6 アドレスが記載されていない、ということと言える。次節ではこの点を踏まえた分析を行う。

6.4 IPv6 アドレスのハードコードと IPv4 との併記分析

前節で IPv4 アドレスのハードコードの影響が IPv6 シングルスタック環境において広範に及ぶ可能性に言及したが、アプリケーション内に IPv4 アドレスと同時に IPv6 アドレスも同時に記載されていれば、IPv6 シングルスタック

表 5 IPv4 アドレスと判定された文字列のうち出現数の多いもの

	アプリ数 (内訳)
両方の記載あり	63,298
(IPv4 は 0.0.0.0 のみ記載)	483
(IPv4 はグローバルのみ記載)	16,590
(IPv4 はプライベートのみ記載)	3,417
IPv4 のみ記載あり	121,056
IPv6 のみ記載あり	27,825

環境でも問題なく動作する可能性がある。そこで IPv4 アドレスと IPv6 アドレスが同じアプリに同時に記載されているかの調査も行った。

その結果を表 5 に示す。IPv4 と IPv6 アドレスの両方の記載があったアプリケーションは 63,298 個であった。そのうち、記載されている IPv4 アドレスが 0.0.0.0 のみであったアプリケーションは 483 個、グローバル IP アドレスのみが記載されていたアプリケーションは 16,590 個、プライベートのみが記載されているアプリケーションは 3,417 個であることがわかった。

これらを踏まえると、こので示された数値を表 3 の数値から引いて考えると、IPv6 アドレスの記載がなく IPv4 のみの記載がされたアプリケーションの状況がわかる。前節までに考察したように「インターネット全体やグローバル IP、プライベート IP アドレスがハードコードされているアプリケーションはシングルスタック環境において通信とそれをもとにしたサービスに不具合が発生する可能性が存在する」ようなアプリケーションは総数で 60,173 個と全体の 12.21% にわたることが示された。

7. 制限と今後の課題

今回の調査では、IPv4 アドレスと IPv6 アドレスの正規表現を用いて文字列一致をもってハードコーディングがされていると判定をした。一方で、言及したようにこれらの一致は誤った計上がされている可能性は高い。より正確な調査を行うためには、一致と判定された情報がどういったタイプの情報でその情報がどこのクラス・メソッドで利用されるか、など静的解析をより高度に行う必要がある。

また、IPv4 や IPv6 のアドレスがハードコードされている場合においても、注意深い実装ではアプリケーション立ち上げ時にサーバ側との通信が DNS のトラブル等で成功しなかった場合に代替的にアクセスする手段として用意している可能性もある。そういった場合にはハードコードされているといっても IPv6 シングルスタック環境で問題なく動作することが考えられる。これらの判断が可能な高度な解析が必要となるだろう。

IPv4 のアドレス記載方法は 1 種類だけではないことも注意したい。用途としては一般的ではないが、IPv4 アドレスは複数の記載方法がある。W3C が示した URL Standard では IPv4 アドレスはドット区切りで数値が 4 つある今回

の調査対象表現だけでなく、ドット区切りで数値を3組とする方法やドット区切りを2組とする方法、ドット無の方法がゆるさされている。また10進数表記ではなく0xを数値の前に記載することで16進数として記載することや、0を数値の前に記載することで8進数として記載されることも許可されているなど、幅は広い。用途としては一般的とは言えないため、ハードコーディングがこれらの形式で行われている可能性は低いと考えられるが、厳格な分析のためにはこういった部分の検査も行うことが望ましいだろう。

<<<< ToDo >>>> 頻出 IP アドレスの他の部分の考察

8. おわりに

謝辞 本研究の一部は JSPS 科研費 JP19K11972、JP19H04111、JP19H04101 の助成を受けたものです。

参考文献

- [1] Support - Apple Developer, “Supporting IPv6-only Networks”, 2016, <https://developer.apple.com/support/ipv6/>
- [2] Jari Arkko, Ari Keranen, “Experiences from an IPv6-Only Network”, 2012, <https://tools.ietf.org/html/rfc6586>
- [3] 北口 善明、近堂 徹、鈴木 伊知郎、小林 貴之、前野 譲二、“クライアント OS の IPv6 実装検証から見たネットワーク運用における課題の考察”, デジタルプラクティス, 2018
- [4] Emre Durdagi, Ali Buldu, “IPV4/IPV6 security and threat comparisons” Procedia Social and Behavioral Sciences 2 (2010) 5285–5291, 2010
- [5] Athirah Rosli, Wan Nor Ashiqin Wan Ali, Abidah Haji Mat Taib Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Perlis, Perlis, Malaysia “IPv6 Deployment: Security Risk Assessment Using i-SeRP System in Enterprise Network” 2012 IEEE Student Conference on Research and Development (SCORED), 2012
- [6] 加茂恵梨香, 岡田雅之, 金岡晃 (東邦大学理学部情報科学科) “Android 環境の IPv6 対応の調査と分析”, 2019 年暗号と情報セキュリティシンポジウム (SCIS 2019), 2019
- [7] K. Allix, T. F. Bissyandé, J. Klein and Y. L. Traon, “AndroZoo: Collecting Millions of Android Apps for the Research Community,” 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), 2016, pp. 468-471.

付 録

A.1 詳細データ

詳細なデータを Github で公開している。
<https://github.com/kanaoka-laboratory/IPAddrHardCodingSurvey>