# Androidとセキュリティ：Android 2.3(Gingerbread) SDKに標準搭載されたProGuardを試す

*技術部　瀬戸 直喜(前・日本Androidの会四国支部長/情報セキュリティスペシャリスト)*

## はじめに

前回の記事ではAntを使う形で、難読化ツールであるProGuardの適用方法を紹介しました。*1

今回は本日(日本時間12/7)リリースされた Android 2.3(Gingerbread)のSDKに標準で搭載されたProGuardの適用方法について紹介します。

## ProGuardとは

詳細は前回の記事を参照頂きたいのですが、ProGuardが初めての方の為に簡単に説明します。ProGuardはソースコードをコンパイルする際に処理を最適化したり、プログラム中の変数やメソッドを意味のない文字列に置き換え、逆コンパイルされた際に処理の中身をわかりにくくする、いわゆる「難読化」を行うツールです。

これまでもAntを使ってProGuardの適用は行えましたが、最新のSDKでは標準搭載され、**Eclipse上から簡単に難読化が行える**ようになりました。

## GoogleがProGuardを標準搭載した理由

他者による解析を難しくさせるツールを標準機能として載せるということは、裏を返せば「Androidのアプリケーションは、リバースエンジニアリングが容易」ということでしょう。Google公式サイトの解説ページに「ProGuardは完全にオプションとして動作するが、適用を**"強くおすすめする"**」と書かれています。「Androidアプリ開発者はコードレベルで知的財産を守るべき」という、Googleからの静かなるメッセージなのかもしれません。

## ProGuardを有効にする方法

それではどのようにProGuardを適用すればよいか、順を追って解説します。

### プロジェクトの作成

Android 2.3のSDKを利用してEclipse上でプロジェクトを新規作成すると、ProGuardの設定が記述された「proguard.cfg」ファイルが自動的に生成されます。このファイルには予め、標準的に必要となる設定が記載されていますので、基本的に修正の必要はありません。

### default.propertiesの修正

作成したプロジェクトのホーム直下に配置されているdefault.propertiesの最終行に、下記を追記し

ます。

```
proguard.config = <絶対パス> または <プロジェクトのホームからの相対パス>
```

以下はプロジェクトのホームにproguard.cfgが配置されている状態での、default.propertiesの設定の例です。

```
# This file is automatically generated by Android Tools.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file must be checked in Version Control Systems.
#
# To customize properties used by the Ant build system use,
# "build.properties", and override values to adapt the script to your
# project structure.

# Project target.
target=android-9
proguard.config=proguard.cfg
```

基本的にはこの設定だけでProGuardが有効になります！*2

**ProGuardの設定変更**

基本的に前節の設定のみでProGuardが実行されますが、下記のような場合は実行に失敗することがあります。

- AndroidManifest.xmlファイルのみから参照されているクラス
- JNIから呼び出されるメソッド
- 動的参照される変数やメソッド

ProGuardの実行に失敗する場合は、proguard.cfgファイルに -keep オプションを付加した行を追記し、ProGuardの適用を除外する必要があります。

```
-keep public class <MyClass>
```

-keepオプションに関する詳しい情報は、前回の記事やProGuardのマニュアルを参照して下さい。

## ProGuardを適用する

**Antを使う**

従来の方法と同様、ant release コマンドでProGuardを適用することが可能です。詳細は前回の記事を参照して下さい。

**Eclipseのエクスポート機能を使う**

メニューから、ファイル -> エクスポート -> Android -> Export Android application と選択して、アプリケーションへの署名を行い、apkファイルを出力します。

apkファイルの出力と同時に、下記のようなログがEclipseのコンソールビューに出力されているは

ずです。

```
[2010-12-07 13:05:28 - HelloAndroid] Refreshing resource folders.
[2010-12-07 13:05:28 - HelloAndroid] Starting incremental Pre Compiler: Chec
[2010-12-07 13:05:28 - HelloAndroid] Nothing to pre compile!
[2010-12-07 13:05:28 - HelloAndroid] /android-sdk-mac_x86/platform-tools/aap
[2010-12-07 13:05:28 - HelloAndroid] Found 0 custom asset files in /Users/se
[2010-12-07 13:05:28 - HelloAndroid] Locale/Vendor pairs:
[2010-12-07 13:05:28 - HelloAndroid]    /
[2010-12-07 13:05:28 - HelloAndroid]    /
[2010-12-07 13:05:28 - HelloAndroid]    /
[2010-12-07 13:05:28 - HelloAndroid]    /
[2010-12-07 13:05:28 - HelloAndroid]
[2010-12-07 13:05:28 - HelloAndroid] ファイル:
[2010-12-07 13:05:28 - HelloAndroid]   drawable-hdpi/icon.png
[2010-12-07 13:05:28 - HelloAndroid]       Src: /Users/seto/Documents/worksp
[2010-12-07 13:05:28 - HelloAndroid]   drawable-ldpi/icon.png
[2010-12-07 13:05:28 - HelloAndroid]       Src: /Users/seto/Documents/worksp
[2010-12-07 13:05:28 - HelloAndroid]   drawable-mdpi/icon.png
[2010-12-07 13:05:28 - HelloAndroid]       Src: /Users/seto/Documents/worksp
[2010-12-07 13:05:28 - HelloAndroid]   layout/main.xml
[2010-12-07 13:05:28 - HelloAndroid]       Src: /Users/seto/Documents/worksp
[2010-12-07 13:05:28 - HelloAndroid]   values/strings.xml
[2010-12-07 13:05:28 - HelloAndroid]       Src: /Users/seto/Documents/worksp
[2010-12-07 13:05:28 - HelloAndroid]   AndroidManifest.xml
[2010-12-07 13:05:28 - HelloAndroid]       Src: /Users/seto/Documents/worksp
[2010-12-07 13:05:28 - HelloAndroid] Including resources from package: /andr
[2010-12-07 13:05:28 - HelloAndroid] applyFileOverlay for drawable
[2010-12-07 13:05:28 - HelloAndroid] applyFileOverlay for layout
[2010-12-07 13:05:28 - HelloAndroid] applyFileOverlay for anim
[2010-12-07 13:05:28 - HelloAndroid] applyFileOverlay for xml
[2010-12-07 13:05:28 - HelloAndroid] applyFileOverlay for raw
[2010-12-07 13:05:28 - HelloAndroid] applyFileOverlay for color
[2010-12-07 13:05:28 - HelloAndroid] applyFileOverlay for menu
[2010-12-07 13:05:28 - HelloAndroid]    (processed image /Users/seto/Docume
[2010-12-07 13:05:28 - HelloAndroid]    (processed image /Users/seto/Docume
[2010-12-07 13:05:28 - HelloAndroid]    (processed image /Users/seto/Docume
[2010-12-07 13:05:28 - HelloAndroid]    (new resource id icon from drawable
[2010-12-07 13:05:28 - HelloAndroid]    (new resource id icon from drawable
[2010-12-07 13:05:28 - HelloAndroid]    (new resource id icon from drawable
[2010-12-07 13:05:28 - HelloAndroid]    (new resource id main from /Users/s
[2010-12-07 13:05:28 - HelloAndroid] Opening '/var/folders/cb/cb2FqG62E5izgb
[2010-12-07 13:05:28 - HelloAndroid] Writing all files...
[2010-12-07 13:05:28 - HelloAndroid]      'res/layout/main.xml' (compressed
[2010-12-07 13:05:28 - HelloAndroid]      'AndroidManifest.xml' (compressed
[2010-12-07 13:05:28 - HelloAndroid]      'resources.arsc' (not compressed)
[2010-12-07 13:05:28 - HelloAndroid]      'res/drawable-hdpi/icon.png' (not
[2010-12-07 13:05:28 - HelloAndroid]      'res/drawable-ldpi/icon.png' (not
[2010-12-07 13:05:28 - HelloAndroid]      'res/drawable-mdpi/icon.png' (not
[2010-12-07 13:05:28 - HelloAndroid] Generated 6 files
[2010-12-07 13:05:28 - HelloAndroid] Included 0 files from jar/zip files.
[2010-12-07 13:05:28 - HelloAndroid] Checking for deleted files
[2010-12-07 13:05:28 - HelloAndroid] 終了!
[2010-12-07 13:05:28 - HelloAndroid] ProGuard, version 4.4
```

```
[2010-12-07 13:05:28 - HelloAndroid] Reading input...
[2010-12-07 13:05:28 - HelloAndroid] Reading program jar [/private/var/folde
[2010-12-07 13:05:28 - HelloAndroid] Reading library jar [/android-sdk-mac_x
[2010-12-07 13:05:30 - HelloAndroid] 初期化中...
[2010-12-07 13:05:30 - HelloAndroid] 注: the configuration refers to the unk
[2010-12-07 13:05:30 - HelloAndroid] 注: there were 1 references to unknown
[2010-12-07 13:05:30 - HelloAndroid]       You should check your configurati
[2010-12-07 13:05:30 - HelloAndroid] Ignoring unused library classes...
[2010-12-07 13:05:30 - HelloAndroid]   Original number of library classes: 2
[2010-12-07 13:05:30 - HelloAndroid]   Final number of library classes:    1
[2010-12-07 13:05:30 - HelloAndroid] Printing kept classes, fields, and meth
[2010-12-07 13:05:30 - HelloAndroid] Shrinking...
[2010-12-07 13:05:30 - HelloAndroid] Printing usage to [/Users/seto/Document
[2010-12-07 13:05:30 - HelloAndroid] Removing unused program classes and cla
[2010-12-07 13:05:30 - HelloAndroid]   Original number of program classes: 8
[2010-12-07 13:05:30 - HelloAndroid]   Final number of program classes:    2
[2010-12-07 13:05:30 - HelloAndroid] Optimizing...
[2010-12-07 13:05:30 - HelloAndroid]   Number of finalized classes:
[2010-12-07 13:05:30 - HelloAndroid]   Number of vertically merged classes:
[2010-12-07 13:05:30 - HelloAndroid]   Number of horizontally merged classes
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed write-only fields:
[2010-12-07 13:05:30 - HelloAndroid]   Number of privatized fields:
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined constant fields:
[2010-12-07 13:05:30 - HelloAndroid]   Number of privatized methods:
[2010-12-07 13:05:30 - HelloAndroid]   Number of staticized methods:
[2010-12-07 13:05:30 - HelloAndroid]   Number of finalized methods:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed method parameters:
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined constant parameters
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined constant return val
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined short method calls:
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined unique method calls
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined tail recursion call
[2010-12-07 13:05:30 - HelloAndroid]   Number of merged code blocks:
[2010-12-07 13:05:30 - HelloAndroid]   Number of variable peephole optimizat
[2010-12-07 13:05:30 - HelloAndroid]   Number of arithmetic peephole optimiz
[2010-12-07 13:05:30 - HelloAndroid]   Number of cast peephole optimizations
[2010-12-07 13:05:30 - HelloAndroid]   Number of field peephole optimization
[2010-12-07 13:05:30 - HelloAndroid]   Number of branch peephole optimizatio
[2010-12-07 13:05:30 - HelloAndroid]   Number of simplified instructions:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed instructions:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed local variables:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed exception blocks:
[2010-12-07 13:05:30 - HelloAndroid]   Number of optimized local variable fr
[2010-12-07 13:05:30 - HelloAndroid] Shrinking...
[2010-12-07 13:05:30 - HelloAndroid] Removing unused program classes and cla
[2010-12-07 13:05:30 - HelloAndroid]   Original number of program classes: 2
[2010-12-07 13:05:30 - HelloAndroid]   Final number of program classes:    2
[2010-12-07 13:05:30 - HelloAndroid] Optimizing...
[2010-12-07 13:05:30 - HelloAndroid]   Number of finalized classes:
[2010-12-07 13:05:30 - HelloAndroid]   Number of vertically merged classes:
[2010-12-07 13:05:30 - HelloAndroid]   Number of horizontally merged classes
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed write-only fields:
[2010-12-07 13:05:30 - HelloAndroid]   Number of privatized fields:
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined constant fields:
```

```
[2010-12-07 13:05:30 - HelloAndroid]   Number of privatized methods:
[2010-12-07 13:05:30 - HelloAndroid]   Number of staticized methods:
[2010-12-07 13:05:30 - HelloAndroid]   Number of finalized methods:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed method parameters:
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined constant parameters
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined constant return val
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined short method calls:
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined unique method calls
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined tail recursion call
[2010-12-07 13:05:30 - HelloAndroid]   Number of merged code blocks:
[2010-12-07 13:05:30 - HelloAndroid]   Number of variable peephole optimizat
[2010-12-07 13:05:30 - HelloAndroid]   Number of arithmetic peephole optimiz
[2010-12-07 13:05:30 - HelloAndroid]   Number of cast peephole optimizations
[2010-12-07 13:05:30 - HelloAndroid]   Number of field peephole optimization
[2010-12-07 13:05:30 - HelloAndroid]   Number of branch peephole optimizatio
[2010-12-07 13:05:30 - HelloAndroid]   Number of simplified instructions:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed instructions:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed local variables:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed exception blocks:
[2010-12-07 13:05:30 - HelloAndroid]   Number of optimized local variable fr
[2010-12-07 13:05:30 - HelloAndroid] Shrinking...
[2010-12-07 13:05:30 - HelloAndroid] Removing unused program classes and cla
[2010-12-07 13:05:30 - HelloAndroid]   Original number of program classes: 2
[2010-12-07 13:05:30 - HelloAndroid]   Final number of program classes:    2
[2010-12-07 13:05:30 - HelloAndroid] Optimizing...
[2010-12-07 13:05:30 - HelloAndroid]   Number of finalized classes:
[2010-12-07 13:05:30 - HelloAndroid]   Number of vertically merged classes:
[2010-12-07 13:05:30 - HelloAndroid]   Number of horizontally merged classes
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed write-only fields:
[2010-12-07 13:05:30 - HelloAndroid]   Number of privatized fields:
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined constant fields:
[2010-12-07 13:05:30 - HelloAndroid]   Number of privatized methods:
[2010-12-07 13:05:30 - HelloAndroid]   Number of staticized methods:
[2010-12-07 13:05:30 - HelloAndroid]   Number of finalized methods:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed method parameters:
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined constant parameters
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined constant return val
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined short method calls:
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined unique method calls
[2010-12-07 13:05:30 - HelloAndroid]   Number of inlined tail recursion call
[2010-12-07 13:05:30 - HelloAndroid]   Number of merged code blocks:
[2010-12-07 13:05:30 - HelloAndroid]   Number of variable peephole optimizat
[2010-12-07 13:05:30 - HelloAndroid]   Number of arithmetic peephole optimiz
[2010-12-07 13:05:30 - HelloAndroid]   Number of cast peephole optimizations
[2010-12-07 13:05:30 - HelloAndroid]   Number of field peephole optimization
[2010-12-07 13:05:30 - HelloAndroid]   Number of branch peephole optimizatio
[2010-12-07 13:05:30 - HelloAndroid]   Number of simplified instructions:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed instructions:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed local variables:
[2010-12-07 13:05:30 - HelloAndroid]   Number of removed exception blocks:
[2010-12-07 13:05:30 - HelloAndroid]   Number of optimized local variable fr
[2010-12-07 13:05:30 - HelloAndroid] Obfuscating...
[2010-12-07 13:05:30 - HelloAndroid] Printing mapping to [/Users/seto/Docume
[2010-12-07 13:05:30 - HelloAndroid] Writing output...
```

```
[2010-12-07 13:05:30 - HelloAndroid] Preparing output jar [/private/var/fold
[2010-12-07 13:05:30 - HelloAndroid]   Copying resources from program jar [/
[2010-12-07 13:05:30 - HelloAndroid] Printing classes to [/Users/seto/Docume
```

下記のように出力されていれば、難読化がされています。

```
[2010-12-07 13:05:30 - HelloAndroid] Obfuscating...
[2010-12-07 13:05:30 - HelloAndroid] Printing mapping to [/Users/seto/Docume
```

### ProGuard有効時に出力されるファイル/ディレクトリ

ProGuardが適用されると、<プロジェクトのホーム>/proguard/ にディレクトリが生成され、その中に下記のようなファイルが出力されます。*3

- dump.txt：.apkファイル中のクラスの内部構造が記載されています。
- mapping.txt：難読化前と難読化後のクラス、メソッド、変数の対応リスト
- seed.txt：難読化されていないクラスやメンバのリスト
- usage.txt：.apkファイルから外されたコードのリスト

特に、mapping.txtは、リリースビルドしたapkから送られる難読化されたバグリポート（スタックトレース）を可読化するために必要となります。

### ProGuard利用時の注意点

proguardディレクトリは、リリースビルドの際に上書きされます。これらを失うと、リリース後のバグレポートから原因を調査することが困難になりますので、proguardディレクトリの中身（特にmapping.txtファイル）は、リリースバージョン毎に保存しておく必要があります。

Android開発者サイトでは、保存の方法として下記のようなものが推奨されています。

- リリース時のバージョンまたはビルド番号を、フォルダやファイル名に追記する
- バージョン管理システムにファイルを登録する

### 難読化されたスタックトレースをデコードする方法

ProGuardは難読化において相応の効果を発揮するツールではありますが、反面、デバッグに必要な情報までをも難読化させてしまう欠点があります。そのため、最新のAndroid SDKには、難読化されたスタックトレースを可読化するための、シェルスクリプトが付属されています。
<sdk_root>/tools/proguard/ ディレクトリにretrace.sh（Windowsの場合、retrace.bat）が格納されていますので、このスクリプトを使用します。

コマンドの書式：

```
retrace.bat|retrace.sh [-verbose] mapping.txt [<stacktrace_file>]
```

使用例：

```
retrace.sh -verbose mapping.txt obfuscated_trace.txt
```

例のmapping.txtは、リリース時に保存しておいたファイル、obfuscated_trace.txtは、難読化されたバグリポート中のスタックトレースを記載したファイルです。

ちなみに、スタックトレースを記載したファイルを省略した場合、標準入力からスタックトレースを読み込んで、デコードします。

より簡単に使えるようになったProGuardで、あなたのアプリケーションを守りましょう！

*1：書いた矢先に標準搭載され、記事の価値が半分になり、ついカッとなって追加記事を書いたりはしていません。

*2：簡単すぎて前回の記事は一体何だったんだと言いたくなります。

*3：Antで実行した場合、出力先は<プロジェクトのホーム>/bin/proguard/ となります。

---