DashO 簡単操作ガイド

Android アプリ難読化

AG-TECH CORP.

目次

I. はじめに

DashO は汎用的な Java 難読化ツールとして登場しましたが、Android の難読化ツールとしても利用可能な拡張がされています。

このドキュメントでは、Gradle 環境に特化して Android アプリケーションの難読化の基本的な操作方法をご説明いたします。

(1) 公式マニュアルについて

DashO の公式マニュアルでは、DashO を使用する上での詳細が記述されています。

DashO を使用する上で、より詳しい内容は、

http://www.agtech.co.jp/download/manual/preemptive/

II. 機能概要

DashO での機能を簡単に紹介します。

DashO における主な機能は、

・名前の変更

・制御フロー

・文字列の暗号化

・ランタイム チェック

の４つから成り立っています。

（1）名前の変更

名前の変更による難読化では、クラス、メソッド、フィールド、アノテーション、およびパッケージの階層の名前を変更できます。名前から意味や機能が推測できるようなメソッド名を、 “a” のような意味のない名前に変換します。ソースコードの可読性はかなり落ちることになります。また、Overload-Induction（オーバーロード誘導）という米国で特許を取得したテクノロジを実装することで、名前変更の大部分が、単に古い名前 1 つにつき新

しい 1 つの名前を割り当てるのではなく、可能な限り重複させるように変更されます。

（２）制御フロー

コードの働きを変えないで、理解しにくくなるようにコードを変更し、逆コンパイラがソースコードを再構築しようとするときに探すパターンを削除します。ソースコードの可読性の低下はもちろんですが、度々難読化されたアプリの逆コンパイルの失敗を誘います。

（３）文字列の暗号化

アプリケーション内の文字列を暗号化させることによって、ハッカーがバイナリ内部の文字列参照を検索して、重要なコードを検索しても文字列が見つからなくなります。

（４）ランタイムチェック

実行環境で検出された下記の条件を検知し、対応できるようにします

・デバッグチェック

・ルートチェック

・Shelf Life チェック

・改ざんチェック

※本資料では、ランタイムチェックの機能は紹介しません。

III. DashO を使用する方法

DashO を Gradle ビルド環境にて操作する方法を簡単にご紹介します。

全体の流れとしては、

・難読化の設定をする前に、プロジェクトをビルド

・GUI にて難読化の設定

・プロジェクトをビルド(難読化アプリの出力)

になります。

※Java ソースプログラムは書き換えませんが、既存の プロジェクトで DashO が動作するようにいくつかの build.gradle の書き換えが発生しますので、DashO の Wizard を実行する前に、プロジェクトのファイルはバックアップを取っておく事をお勧めします。

（1）GUI で開く前に、一度プロジェクトをビルド

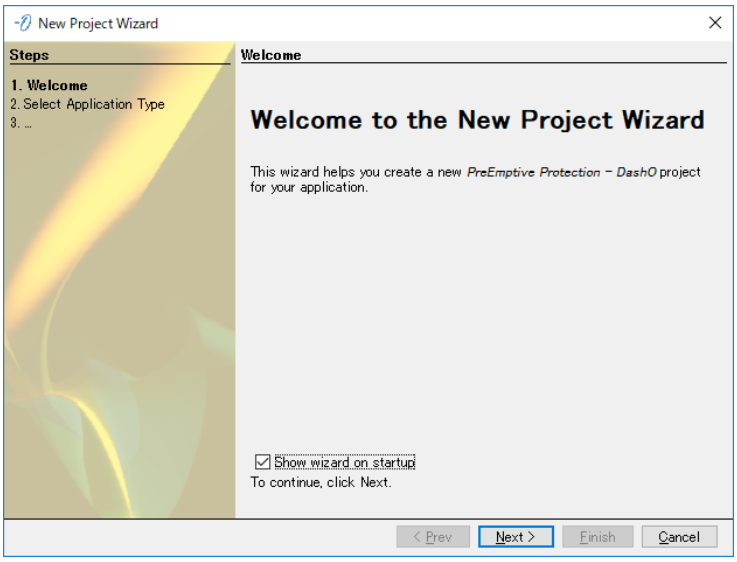
初めて DashO の GUI で開く場合、一度プロジェクトをビルドする必要があります。以下のコマンドでビルドを行ってください。

gradlew build　　（プロジェクトフォルダで実行します）

なお、一度ビルドを行った後は、再度ビルドすることなく、DashO の GUI で開く事ができます。**ただし、メソッドやフィールド等を追加・変更・削除した場合は再度ビルドする必要があります。**

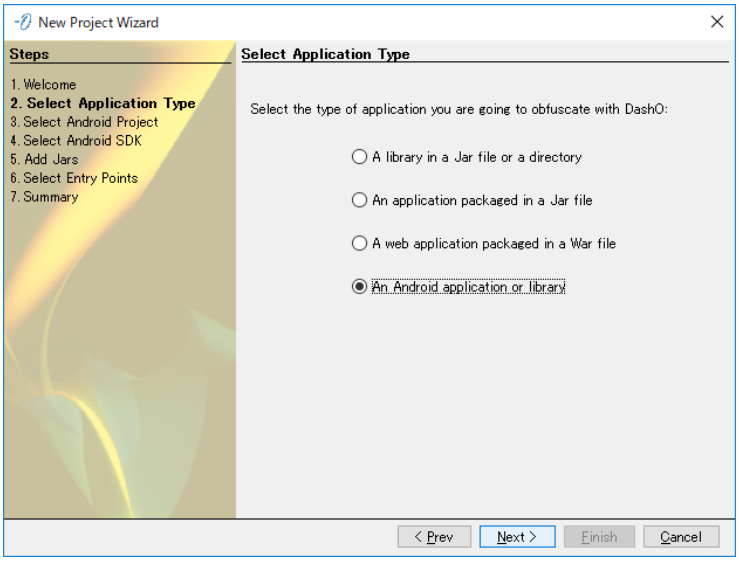
（2）GUI にてプロジェクトの構成

DashO を起動すると、下記のウィザードが立ち上がります。



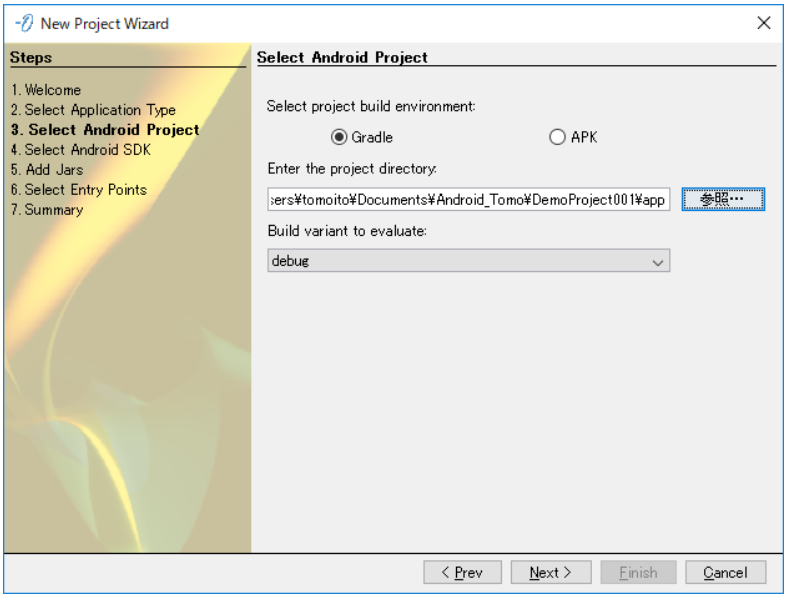
難読化対象のアプリの種類を選択します。

今回は "An Android application or library" を選択します。

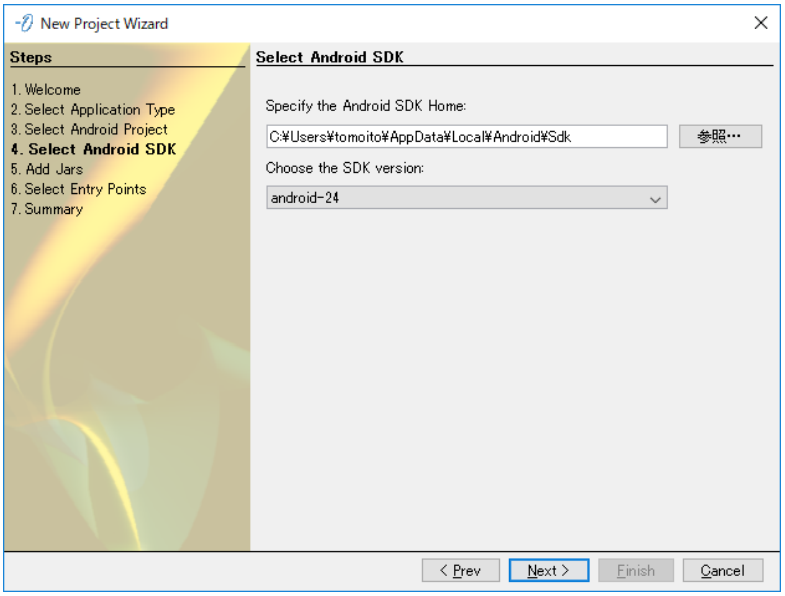


次に、"Gradle" ビルド環境を選択して、Android プロジェクトのディレクトリを入力します。評価するビルドバリアントのデフォルトは「debug」ですが、異なるバリアントも指定可能です。

※Gradle によって使用される build.gradle ファイルの階層を Android プロジェクトのディレクトリと見なします。

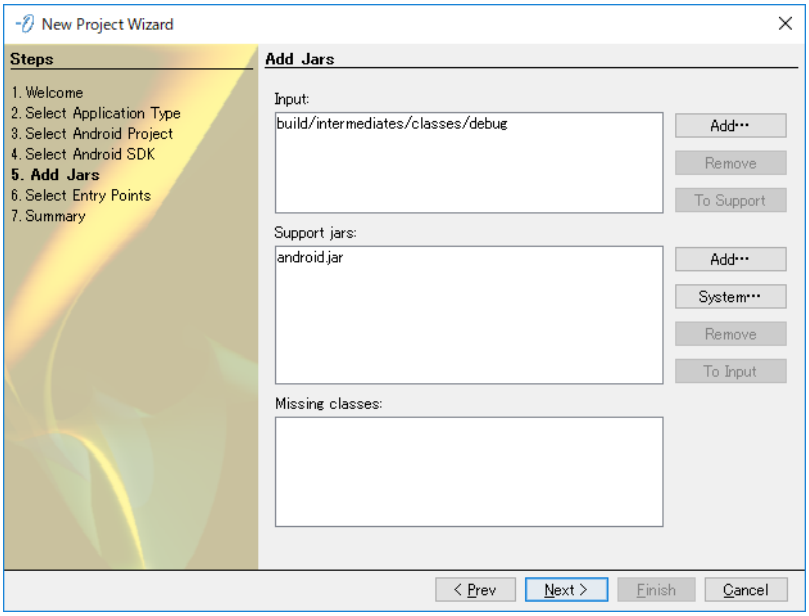


プロジェクトに構成されている SDK のバージョンを選択します。

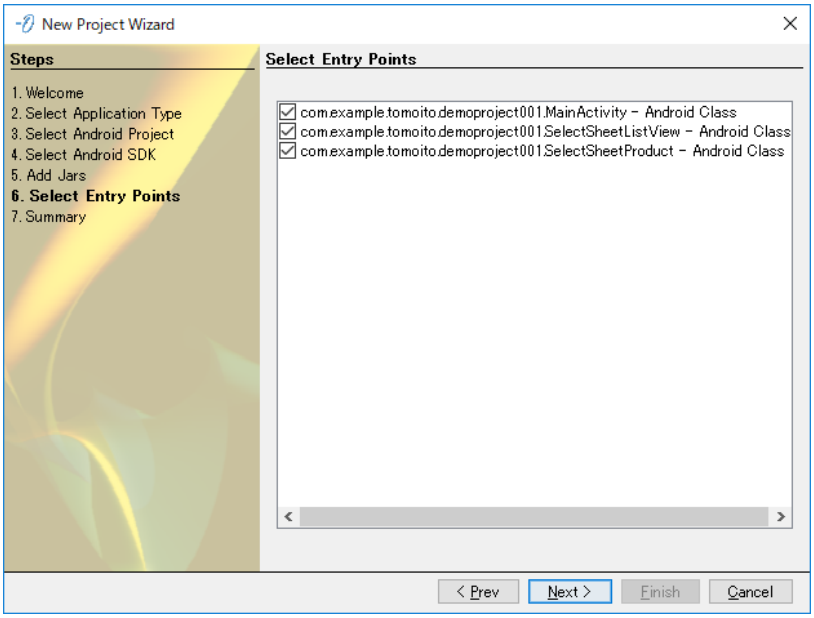


ウィザードはプロジェクトを調べて、難読化に必要となる依存関係を決定します。

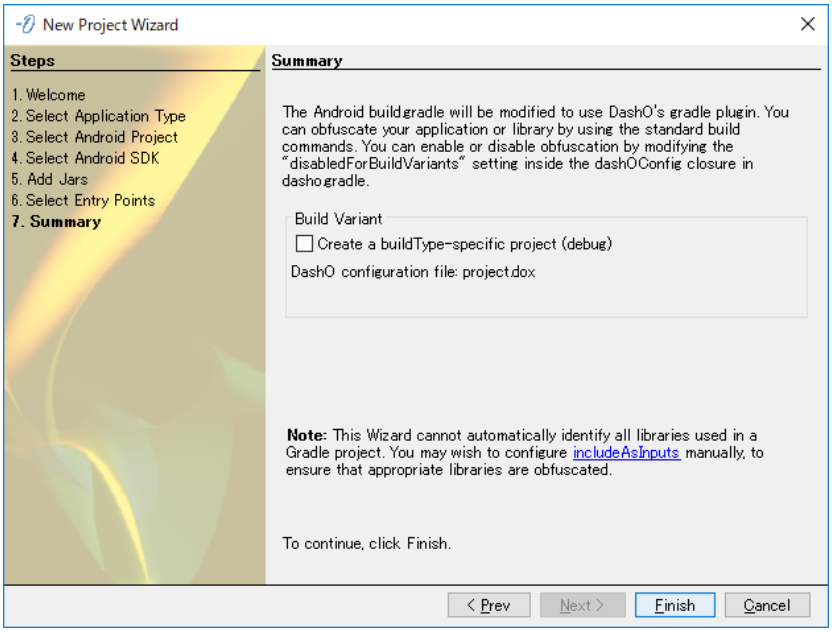
※［Missing classes］一覧は、プロジェクトによって参照されているが、見つからなかったクラスが表示されます。



コンパイルされたクラスを分析して、アプリケーションまたはライブラリのエントリポイントを決定します



プロダクトフレーバーまたはビルドタイプに固有のプロジェクトを生成することができます。

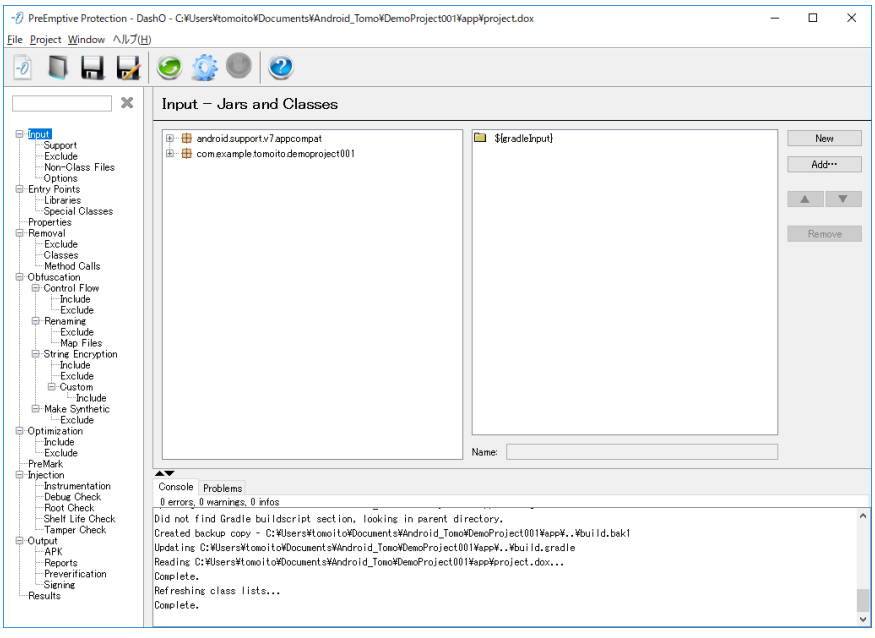


上記で基本的な設定は完了です。

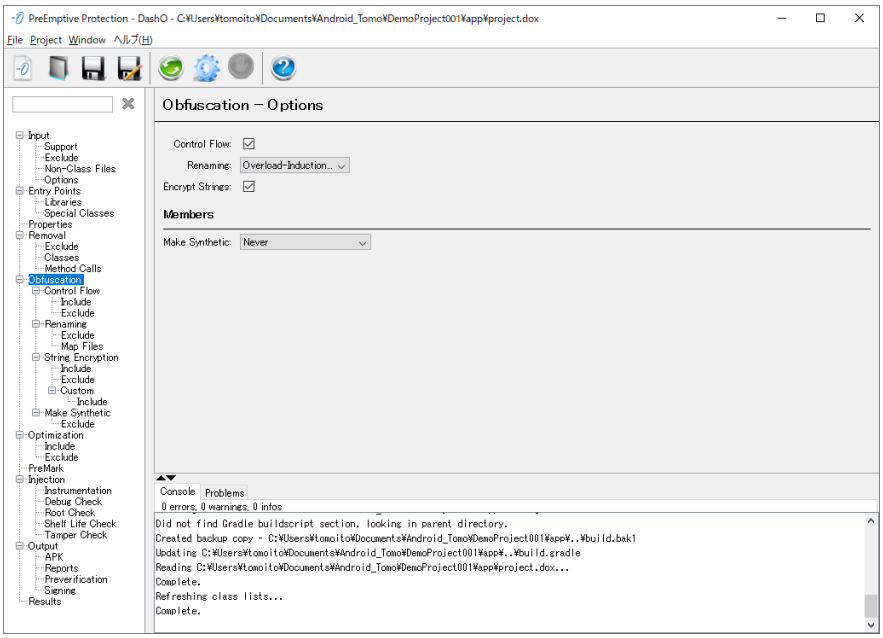
（ウィザードは、上記作業のほとんどを自動的に設定します。）

これで、DashO のプラグインをビルドプロセスで使用するようになりました。

続いて、難読化の個別設定を行う画面を簡単に説明します。

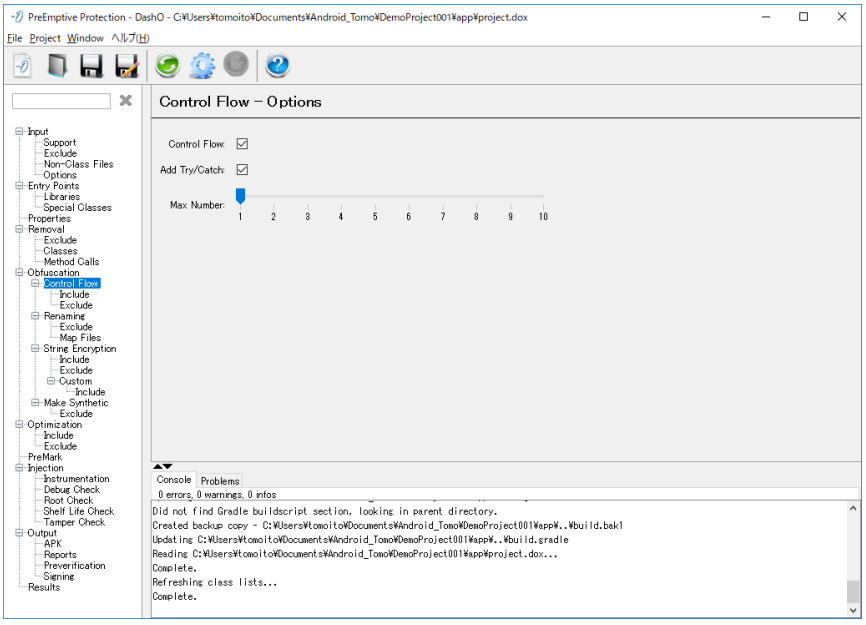


Obfuscation ― Options にて、基本的な難読化設定を制御します。

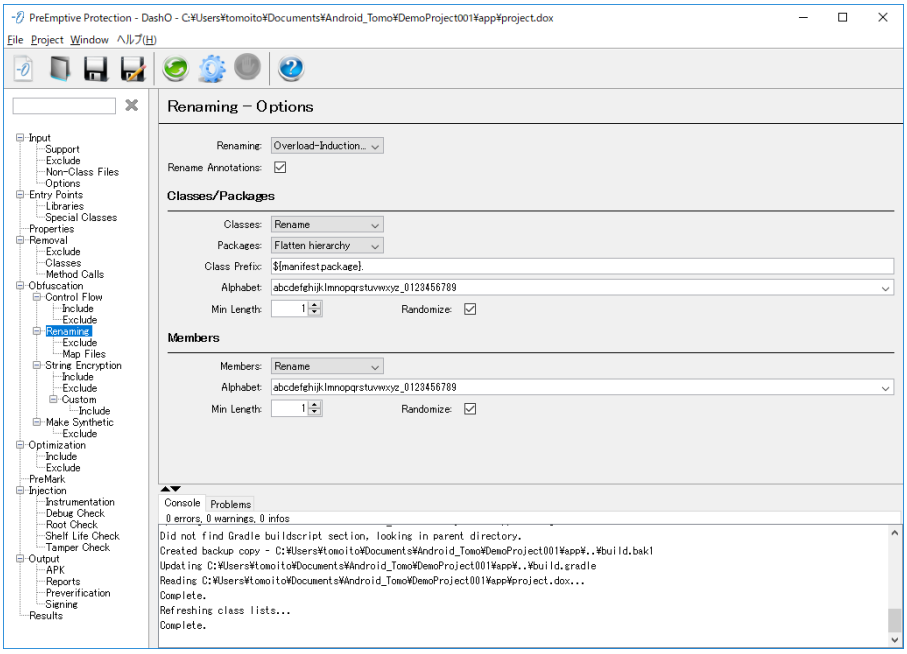


Control Flow – Options パネルでは、Try/Catch ハンドラーを追加する必要があるかどうかを決定します。また、追加するハンドラーの最大数を選択します。

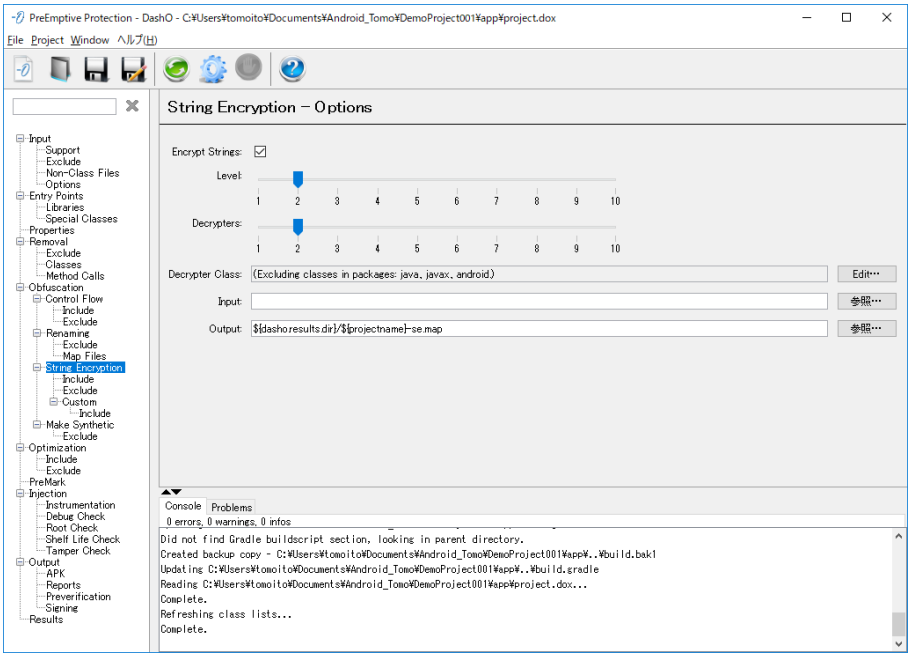
※追加するハンドラーによって、パフォーマンスに影響する可能性がありますので、十分に検証を行ってください。



Renaming – Options パネルで、アプリケーションに含まれる項目の名前の変更をより詳細に制御できます。



String Encryption – Options パネルでは、使用する文字列暗号化のレベルを選択することができます。



（３）プロジェクトのビルド

Wizard 処理後の Gradle スクリプトは、難読化タスクを含んでいます。 assembleRelease や build 等の Gradle タスクにより、難読化のタスクも実行されるようになっています。

通常通り、以下のコマンドでビルドを行ってください。

gradlew build　　（プロジェクトフォルダで実行します）

難読化の出力フォルダは使用する Gradle Plugin の仕様に依存します。

DashO Gradle Plugin 3.0 の場合、app¥build フォルダに出力されます。 このフォルダ内には、各種サブフォルダが生成されますが、難読化 APK は app¥build¥outputs¥apk に出力されます。

難読化の結果は、app¥build¥dasho-results 以下に出力されています。

※次回難読化の差分用として使用可能な MAP ファイルも格納されています。

IV. トラブルシューティング

難読化によりプログラムは、元のプログラムとロジック的に同等の動作をするように加工されますが、プログラムの実行速度に影響があったり、プログラムの実行時に例外が引き起こされる可能性も十分にありますので、難読化後のテストは十分に行ってください。エラー発生時に確認する事項をまとめました、

（1）ビルド時のエラー

・正しい　DashO Gradle プラグインを選択する。

Android Gradle プラグインに基づいて、適切なバージョンの DashO Gradle プラグインを選択する必要があります。

バージョン間の互換性は、下記のページをご参照ください。

https://www.preemptive.com/dasho/pro/userguide/en/gradle/versionCompatability.html

（2）実行時のエラー

・サードパーティ製のアセンブリはすべて難読化の対象から除外する。

一般的に、サードパーティ製アセンブリ自体は難読化しないことをお勧めします。サードパーティ製のアセンブリを除外する方法については、「DashO マニュアル」の「Input ー Global Processing Exclusions」セクションを参照してください。

https://www.preemptive.com/dasho/pro/userguide/ja/ui_advanced.html

・問題になっている難読化の変換を特定する

個別の難読化の機能を一つずつオフにして、エラーが解消されるかを確認していただくことを推奨します。**実行時エラーはほとんどの場合、名前の変更に関係しています。**

・DashO に渡されたすべてのパラメーターを確認する

DashO に渡されたすべてのパラメータを表示する場合は、gradlew を実行するときにコマンドラインに-DSHOW_DASHO_CMD を追加します。 DashO に渡された.properties ファイルの内容が表示され、DashO の実行に使用された完全なコマンドラインが表示されます。

V. まとめ

DashO の簡単な利用方法をご説明しましたが、開発時は難読化を意識したコーディングをする必要がありません。また、ウィザードは、難読化に必要な作業のほとんどを自動的に実行しますので、手間と時間を大幅に節約することになります。無償で利用できる難読化ツールも存在しており、Android においては ProGuard が提供する難読化機能で事足りる場合もあるかもしれませんが、企業の看板となるアプリの場合には、よりレベルの高いリバースエンジニアの対策が求められます。そのような重要なアプリに対しては、弊社が販売している DashO または Dotfuscator をご検討ください。