ホワイトペーパー

セキュリティ

Fortify Static Code Analyzer による Embedded C/C++ の静的コード解析

MICRO FOCUS

目次

ページ

ここでご紹介する機能は、
自動車業界に限らず、
本ホワイトペーパーの
「はじめに」に例示した
市場や領域でも適用できる
可能性があります。

はじめに

Embedded C/C++ をベースにしたマイクロコントローラーに搭載するアプリケーションは増え続けています。自動車、航空機、船舶、医療機器だけではなく、ホームオートメーションやスポーツなどの領域でも同様です。しかし、マイクロコントローラーは搭載されるリソース（特にメモリ）が限られているため、例えば、Green Hills Renesas v850 コンパイラのような、特定のマイクロコントローラーチップ向けのコンパイラがコード最適化のために、頻繁に使用されます。

Micro Focus® Fortify Static Code Analyzer（Fortify SCA）は Visual Studio、Eclipse、IntelliJ などの IDE との統合に優れていますが、Embedded C/C++ のプログラマーは、コマンドラインツールを多用する Linux や UNIX ベースの環境を好む傾向があります。こうした環境では、Embedded C/C++ コードのスキャンは困難です。ただし、このような環境であっても、わずかなカスタマイズで、Fortify Static Code Analyzer を使うことが可能になります。

たとえば、ドイツの開発者は、自動車産業向けの Embedded C/C++ アプリケーションのスキャンに大きな成果を収めています。コードが ANSI-C 準拠であり、Fortify Static Code Analyzer のパーサがコードを理解できることが前提条件です。ほとんどの自動車メーカーは、社外のコーディングベンダーに対して、この基準への準拠を要件としています。

ここでご紹介する機能は、自動車業界に限らず、本ホワイトペーパーの「はじめに」に例示した市場や領域でも適用できる可能性があります。

自動車産業では、主に AUTOSAR と GENIVI の 2 つの開発環境が使用されています。

AUTOSAR ソフトウェア開発プラットフォーム

AUTOSAR（AUTomotive Open System ARchitecture）は、自動車メーカー、サプライヤ、その他のエレクトロニクス、半導体、ソフトウェア業界各社の世界的な開発パートナーシップです。AUTOSAR は、世界のすべての自動車メーカーが貢献するソフトウェア開発プラットフォームでもあり、自動車に搭載される多数のマイクロコントローラーに実装されている Embedded C アプリケーションの開発に使用されています。

AUTOSAR により、自動車メーカーはソフトウェアコンポーネントの開発に注力し、その他の作業を AUTOSAR プラットフォームに任せることができます。マイクロコントローラーで実行される Embedded C アプリケーションには、(特にメモリなどのリソースが不足することの多い) マイクロコントローラーのプロセッサ用に高度に最適化されたコードが必要とされます。コードの最適化は、主に Green Hills または Wind River の特殊なコンパイラにより実行されます。

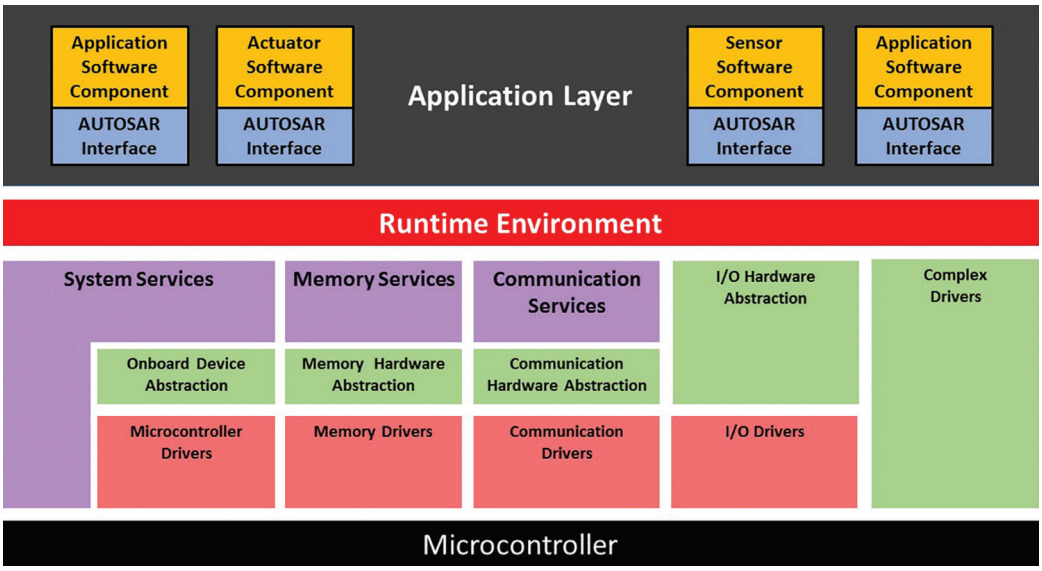

図 1.AUTOSAR プラットフォーム

これらのコンパイラは、Micro Focus Fortify SCA ではまだ公式にサポートされていません。しかし、私たちは、わずかなカスタマイズだけで、ある AUTOSAR アプリケーションソフトウェアコンポーネントと AUTOSAR テクノロジースタック全体をスキャンすることに成功しました。これには、Renesas V850 チップと Green Hills ccv850 コンパイラを使用し、マイクロコントローラーで実行されるアプリケーションソフトウェアコンポーネントが含まれます。このホワイトペーパーでは、これを実現するために必要なカスタマイズを紹介します。

また、同様の方法で、Wind River Diab コンパイラでコンパイルされるコードをスキャンすることにも成功しています。Fortify SCA で、さらに多くの AUTOSAR プラットフォーム上のコンパイラをサポートできるよう取り組んでおり、この分野での新しい事例が増えていくことが期待されています。

同様の方法で Wind River Diab コンパイラでコンパイルされるコードをスキャンすることにも成功しています。Fortify SCA で、さらに多くの AUTOSAR プラットフォーム上のコンパイラをサポートできるよう取り組んでおり、この分野での新しい事例が増えていくことが期待されています。

必要なカスタマイズ

Fortify SCA のプロパティ：fortify.sca.properties のコンパイラセクションに次のエントリを追加します。

```
# Support for Green Hills ccv850 compiler com.fortify.sca.compilers.ccv850 =
com.fortify.sca.util.compilers.GccCompiler com.fortify.sca.compilers.ccv850 =
com.fortify.sca.util.compilers.GppCompiler
```

Fortify SCA のエンジンにより、Embedded C ソースコードの様々な問題が発見されました。(これらの脆弱性にはお客様固有のコード情報が含まれるため、結果の詳細は公表できません。)

PATH 環境変数
sourceanalyser の絶対パスを追加します。さらに make と ccv850 について、後でビルドが実行される場所からのパスを PATH 変数に追加します。これらの実行ファイルを該当の場所から直接呼び出せることを確認します。例：sourceanalyser --help

Makefile の変更
Makefile の CC ステートメントを探し、ccv850 の呼び出し部分を、sourceanalyser でラップされた ccv850 の呼び出しに変更します。例：

```
CC = sourceanalyser -b ccv850
```

ビルドの実行とスキャンステップの実行
通常どおりにビルドを実行して、スキャンステップを実行します。

```
sourceanalyser –b <nameofautorsarapplication> -scan –f <nameof fprfile>
```

結果

Fortify SCA のエンジンにより、Embedded C ソースコードの様々な問題が発見されました。(これらの脆弱性にはお客様固有のコード情報が含まれるため、結果の詳細は公表できません。)

開発者は、この結果をレビューして、ソースコードの問題を修正することができます。

GENIVI ソフトウェア開発プラットフォーム

「GENIVI は非営利の自動車業界アライアンス」です。140 を超える企業が「オープンソースの In-Vehicle Infotainment（IVI）ソフトウェアの広範囲での採用を推進し、コネクテッドカー向けのオープンテクノロジー の提供に取り組んでいます。

完全に専有されている AUTOSAR とは対照的に、GENIVI はすべてオープンソース化されています。

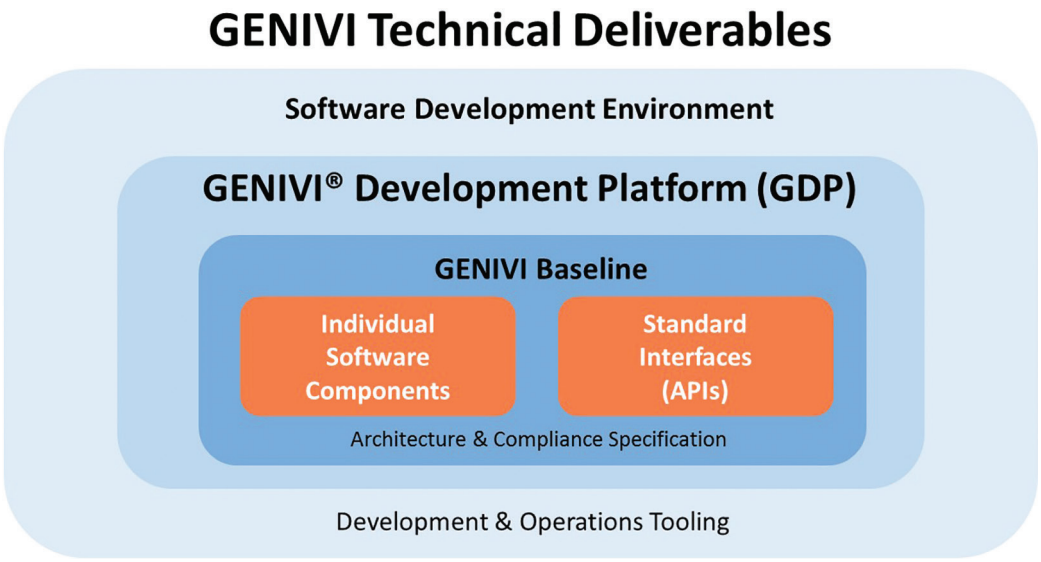

図 2.GENIVI 技術の成果物

GENIVI により、自動車向けインフォテイメントメーカーはソフトウェアコンポーネントの開発に注力し、その他の作業を GENIVI プラットフォームに任せることができます。

マイクロコントローラーで実行される Embedded C アプリケーションには、(特にメモリなどのリソースが不足することの多い) マイクロコントローラーのプロセッサ用に高度に最適化されたコードが必要とされます。

このホワイトペーパーでは、GENIVI Github のアプリケーションの 1 つである、VSOMEIP を対象にします。VSOMEIP アプリケーションコードは当初 BMW AG から提供されたもので、CMAKE をビルドツールとして使用します。

このホワイトペーパーでは、GENIVI Github のアプリケーションである、VSOMEIP を対象にします。VSOMEIP アプリケーションコードは当初 BMW AG から提供されたもので、CMAKE をビルドツールとして使用します。

必要なカスタマイズ

CMAKE は特に難しい課題の 1 つです。Fortify の変換プロセスをコンパイルステップに含める (sourceanalyser でコンパイラをラップする) ためのインターフェイスとなる設定ファイルがなかなか見つからないためです。

今回のケースでは、ビルド環境の最下層までたどり、コンパイラバイナリをシェルスクリプトに置き換えて、次のようにコンパイラの呼び出しを sourceanalyser でラップしました。

シェルスクリプト

```
#!/bin/bash
sourceanalyser -b g++-4.8 “$@”
```

高度な最適化を必要とするコードの場合、「低」に分類された事項も検討対象になるでしょう。デッドコードとすべての定義済み変数は、マイクロコントローラーの貴重なリソースであるメモリを余分に使用します。

fortify-sca.properties ファイルを見ると g++4* のコンパイラエントリがあることが分かります。これで準備は完了です。

```
com.fortify.sca.compilers.g++4* = com.fortify.sca.util.compilers.GppCompiler
```

Cmake と Make を実行すると、コードが正しくコンパイルされてリンクされます。その後、スタティック分析に必要な Fortify のトランスレーションステップを実行しました。

結果

アナライザー
次の 3 つのアナライザーで脆弱性が発見されました。

- Control Flow
- Semantic
- Structural

カテゴリ
高：メモリリーク、タイプの不一致 (符号なしの結果になるはずの結果が符号ありになる、またはその逆)、リリースされないリソース

低：デッドコード (実行されないコード)、定義されたが使用されていない変数、読み取られない変数、初期化されずに使用されている変数、null チェックの欠落、冗長な null チェック、プロセスコントロール、チェックされない戻り値

高度な最適化を必要とするコードの場合、「低」に分類された事項も検討対象になるでしょう。デッドコードとすべての定義済み変数は、マイクロコントローラーの貴重なリソースであるメモリを余分に使用します。

まとめ

最小限のカスタマイズにより、Fortify SCA で AUTOSAR アプリケーションをスキャンできました。AUTOSAR では Make を、GENIVI では Cmake をビルドツールとして使用しました。今回の結果を AUTOSAR と GENIVI プラットフォームの他のアプリケーションにお試しいただくことをお勧めいたします。多くの IoT デバイスにはマイクロコントローラーが搭載されているため、自動車業界以外にも応用可能です。

詳細情報はこちら：
www.microfocus.com/fortifysca

お問い合わせ先：
www.microfocus.com

マイクロフォーカスエンタープライズ株式会社
jp-info-enterprise@microfocus.com
www.microfocus-enterprise.co.jp

162-JA0266-001 | H | 03/20 | © 2020 Micro Focus or one of its affiliates.Micro Focus および Micro Focus ロゴは、英国、米国、およびその他の国における Micro Focus、その子会社、関連会社の商標または登録商標です。その他すべての商標は、該当する所有者に帰属します。