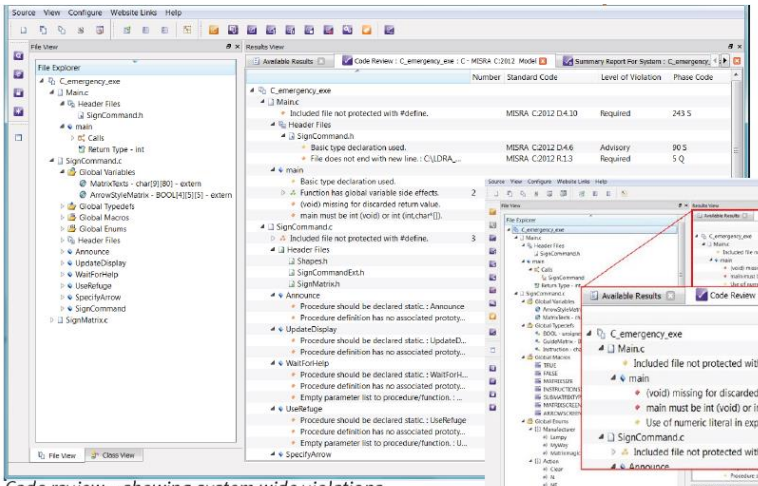
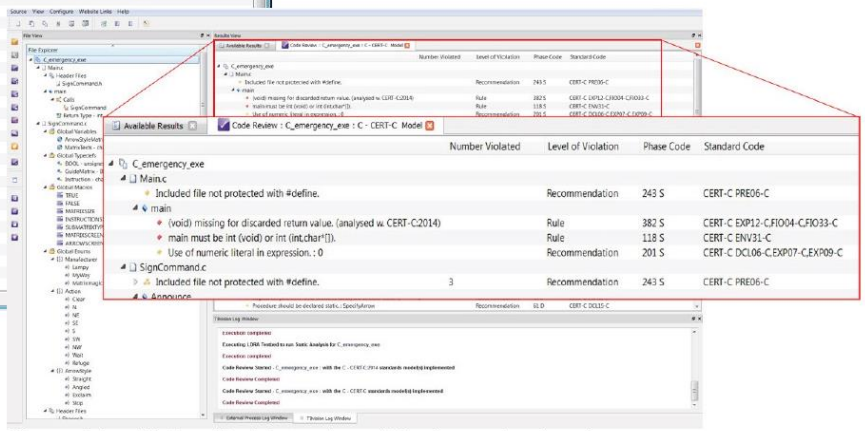
MISRA C Evolves to Address Security Threats in Today's Embedded Software

セキュアでないコーディングの例と その取り扱いルール



Code review – showing system wide violations



TBsecure's tangible benefits deliver an immediate return on investment.

www.ldra.com

MISRA C:2012 Amendment 1

セキュリティリスクに取り組む新たなガイドライン

MISRA C は本来、自動車業界のセーフティクリティカルな組込みアプリケーションで、Cプログラミング言語の使用を促進するために作成された、ソフトウェア開発言語のサブセットです。長年にわたる更新で追加された機能や改善により、プログラマがコンプライアンスへの取り組みよりもコーディングに集中できると同時に、セーフティクリティカルなアプリケーションのソフトウェアに関連するリスクの軽減を支援します。より安全で安心な、高品質のコード実装を支援することを目的としたMISRAガイドラインは、産業界で増大するセキュリティへの脅威を受けて、セキュリティリスクに取り組む新たなガイドラインを追加しました。

MISRA C:2012 Amendment 1 は、最新バージョンのMISRAガイドラインのセキュリティの対象を引き出す、安全なコード実装のための新しいルールを追加しています。これにより、ルールからの逸脱を認めていたコードが実はセキュリティの脆弱性に関わっていた、といった問題も避けることができるようになりました。この改正は拡張・強化であり、MISRA言語のガイドラインのすべてのエディションと完全に互換性があり、将来のすべてのエディションの標準的なアプローチとなります。これらのガイドラインに準拠することで、開発者はセキュリティの脆弱性を引き起こすコードを避けると同時に、これまで以上に可読性と保守性の高いコードを実装することができます。また適切なスタンダードチェックツールを活用することで、開発者は徹底的にコードを分析して、それらが安全かつセキュアコーディングな作法に従っていることを、規制当局やOEMに保証することができます。

Key Details

・ ISO C セキュアガイドラインによってハイライトされた、セキュリティ上の懸念をカバーする14の新しいCコーディングルール
・ 「信頼できない」データの使用という、よくあるセキュリティ上の脆弱性に付き物の具体的な問題へ取り組む
・ MISRA C:2012 Addendum 2 で、ISO/ IEC201217961：2013 に対する MISRA C:2012 のカバレッジをサポート
・ MISRAガイドラインのすべてエディションと完全に互換性がある拡張

セキュアでないコーディングの例とその取り扱いルール

Example 1: ドアを開けてパスワードを漏らすな

“外部から取得した値の妥当性をチェックしよう”

この例では、デバイスとデバイスあるいはデバイスとインターネットの接続が増えて危険度が高まった外部ソースへどんなデータが送信されるかをコントロールしている。もしプログラムが不適切であれば、内部データが外部ソースに晒されてしまう。このルールは、ハッカーが巧妙に作成したメッセージを使って、クリアテキストで保存されたパスワードを含むデータを、インターネット経由で抜き取ることができるHeartbleed型の脆弱性に対する保護となる。

以下のサンプルコードは場合によっては危険である。なぜならユーザーによる入力が読み込まれた後に検証されていないからだ。この例では読み込まれた文字列がヌル文字終端であるかがチェックされていないために、バッファーオーバーランを起こして不適切なメモリ領域をユーザーに晒してしまうかも知れない。実際のところ、この入力がプログラム内部で使われるのは読み込みが済んだ後である。攻撃者は悪意のあるデータを注入することで、この脆弱性を悪用できる。たとえば数値データを読み込ませてバッファーにアクセスするなど。悪意の攻撃者はこれを悪用してバッファー境界の外側へアクセスし、クリアテキストで保存されたパスワードのような重要なデータをネットワークサーバーから読み出すことができる。

```
void f1( void )
{
char input [ 128 ];
( void ) scanf ( “%128c”, input );
( void ) printf ( “%s”, input ); /* 不適合 */
}
```

Example 2: 処理に応じた正しい関数を使おう

“標準ライブラリ関数memcmpはヌル文字終端された文字列には使わない”

memcmp はメモリブロックの比較用に設計されており、パスワードのような文字列のためには作られていない。なぜならこの関数はtrueかfalseかだけでなく、それ以上の情報を返すので、データベース内のパスワードの推測に使えてしまうからだ。以下のサンプルコードはstrcmpが使われるべき所でmemcmpを使っている。

```
                extern char buffer1[ 12 ];
                extern char buffer2[ 12 ];
                void f1 ( void )
                {
                strcpy ( buffer1, “abc” );
                strcpy ( buffer2, “abc” );
                if ( memcmp ( buffer1, buffer2, sizeof ( buffer1 ) ) != 0 )
                {
                /* buffer1 と buffer2 に格納された文字列は異なると判定されるが
                * これは実際にはヌル文字の後の初期化されていない文字のせいである
                */
                }
}
```

Example 3: 話し方を操ることができれば何でも聞き出せる

“標準ライブラリ関数asctime、ctime、gmtime、localtime、localeconv、getenv、setlocale、strerrorが返すポインタは、もう一度同じ関数を呼び出した後では使わない。”

Locale messagesは文字列の書式を操作してソフトウェアが発する「言葉」を決める。もしハッカーが文字列の書式を操作できれば、あるプログラムがどんなプログラムとやり取りするかなど、好きなことをしゃべらせることができる。つまりそれらの文字列を使えば、任意のコマンドを実行してシステムを乗っ取ることもできる。

以下のサンプルコードは期待通りには動かないかも知れない。なぜならsetlocaleの二度目の呼び出しにより、res1が参照する文字列はres2が参照する文字列と同じになってしまうからだ。localeの操作は、文字列の書式を操作してリモートコンピュータ上でコマンドを実行させるために悪用されてきた。

```
void f1( void )
{
const char *res1;
const char *res2;
res1 = setlocale ( LC_ALL, 0 );
res2 = setlocale ( LC_MONETARY, “French” );
printf ( “%s¥n”, res1 ); /* “res1” may NOT be related to the ‘C’ locale. */
}
```

Related Updates

MISRA 委員会は、MISRA C:2012 Amendment 1 と合わせて、いくつかの関連文書をリリースしました。これら文書は、OEM や規制当局に、セキュアなコーディング実践の完全な証拠を提供する必要がある開発者のための情報を提供します。これらの文書は、次のとおりです。

・ MISRA C:2012 Addendum 2：セーフティを目的にする MISRA C は、セキュリティ関連の環境にも同様に適用可能であることを正当化する、ISO/IEC TS 17961:2013 の「C セキュア」に対する MISRA C:2012 のカバレッジ

・MISRA コンプライアンス 2016：MISRA コーディングガイドラインへの準拠と証明のためのガイダンスを提供。MISRA コンプライアンスの意味するところ、逸脱の使用に対する明確なガイダンス、および MISRA ガイドラインの分類と調整について

・ MISRA C:2004 Permits: 逸脱の事前承認の許可を確立するメカニズムを提供

・LDRA 社は MISRA C 委員会の 11 ポジションに委員を派遣するなど、セーフティでセキュリティクリティカルな業界標準の制定、及びそのサポートで、長年のリーダーシップを実証しています。また LDRA 社が提供するテストツールスイートは、MISRA スタンダードの最も包括的なサポートを提供しています。