JPCERT CC®

# ソースコード解析ツールを活用した CERT セキュアコーディングスタンダードの 有効性評価

2009年5月18日

JPCERTコーディネーションセンター
久保 正樹

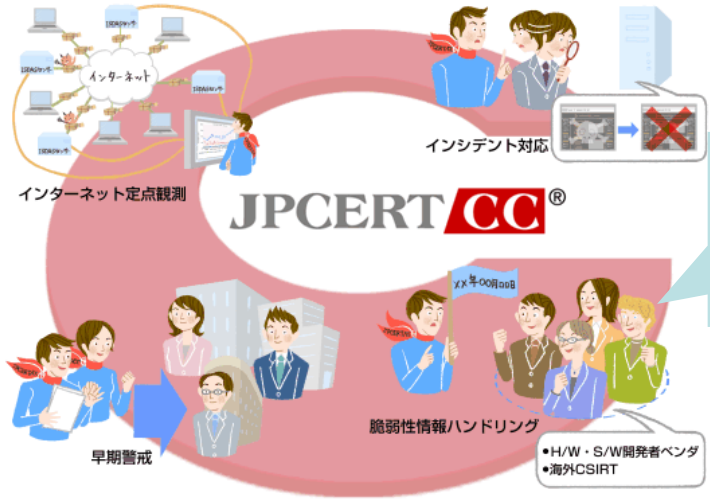
## JPCERT/CC?

JPCERT CC®

- JPCERT/CC
  - JP　:日本
  - Computer　:コンピュータ
  - Emergency　:緊急
  - Response　:対応
  - Team　:チーム
  - Coordination　:コーディネーション
  - Center　:センター
- 日本における情報セキュリティ対策活動の向上に取り組む

Copyright© 2009 JPCERT/CC All rights reserved.

# JPCERT/CCの主な活動




リアクティブな活動に加えて、プロアクティブな活動の実施

Copyright© 2009 JPCERT/CC All rights reserved.


# 目次

1. 脆弱性を取り巻く状況
2. CERT C/C++ セキュアコーディングスタンダードの必要性
3. 本プロジェクトについて
4. 脆弱性削減を目的としたJPCERT/CCの活動

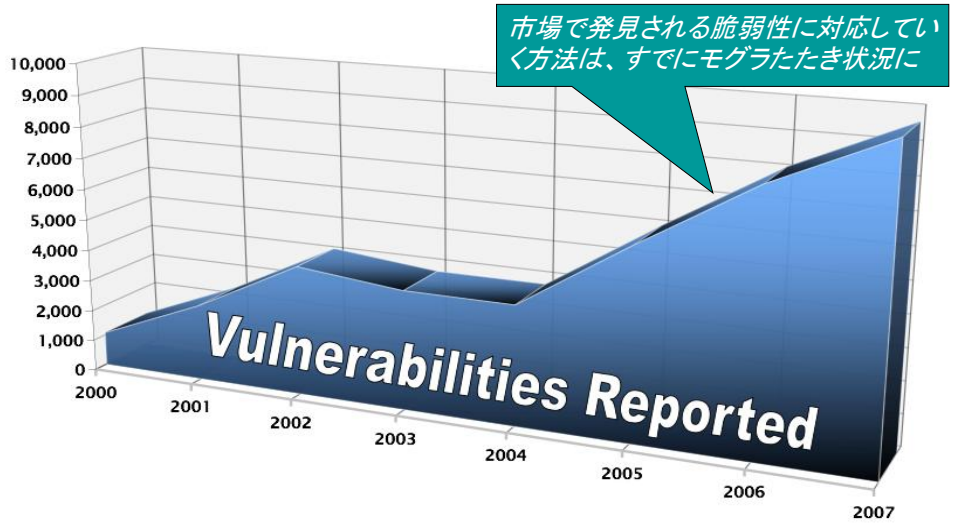
Copyright© 2009 JPCERT/CC All rights reserved.

JPCERT CC®

# 1. 脆弱性を取り巻く状況

## ワールドワイドで増加する脆弱性の数

JPCERT CC®




Copyright© 2009 JPCERT/CC All rights reserved.

## あらゆるソフトウェアが攻撃対象

JPCERT CC®

ウェブカレンダーの入力チェック不備が原因で、細工されたURL経由で任意のコードが実行される。※1

攻撃者が細工したJPEG画像をMMSメッセージで送りつけることで、携帯電話がのっとられてしまう。※2

世界各国の石油、ガス、食品・飲料などの工場でつかわれている制御システムに脆弱性。細工されたパケットを1つおくるだけで、システムがクラッシュ。※3

※1 CVE-2006-2261, ※2 CVE-2008-2548, ※3 CVE-2008-2005

Copyright© 2009 JPCERT/CC All rights reserved.


## 製品出荷後に発見される脆弱性の問題

JPCERT CC®

- 対応コストが大きい
  - 脆弱性を修正する際、場合によっては、ソフトウエアの設計の見直しを含み、以降の下位工程へ影響が発生する上、修正プログラムの開発・周知および配付のためのコストがかかる。
  - 利用者側においても、修正適用のためのリスクと、コストがかかる 。
  - 場合によっては、サービス一次停止など損失が大きくなるケースもある。
- インシデントの発生、被害に繋がるケース
  - 機密情報の漏洩
  - データの改ざん
  - データの破壊
  - システムが乗っ取られ、被害者自らが加害者となる

Copyright© 2009 JPCERT/CC All rights reserved.

## 攻撃者側の有利な事情

JPCERT CC®

1. 容易にアクセス可能なインターネット上の攻撃情報
   - 過去の成功事例はデータベース化
   - 攻撃ツール
2. ちょっと経験をつめば誰でも攻撃者に…
   - 組織的な犯罪
   - 脆弱性のブラックマーケットの存在
3. 攻めるのは容易、守るのは困難
   - 守る側（開発者）は、何千何万というソースコードで書かれたソフトウェアの「全ての」想定される脆弱性に対応しないとダメ
   - 一方、攻撃者は「1つ」脆弱性をみつけさえすればよい

Copyright© 2009 JPCERT/CC All rights reserved.


## 脆弱性ブラックマーケットの形成

JPCERT CC®

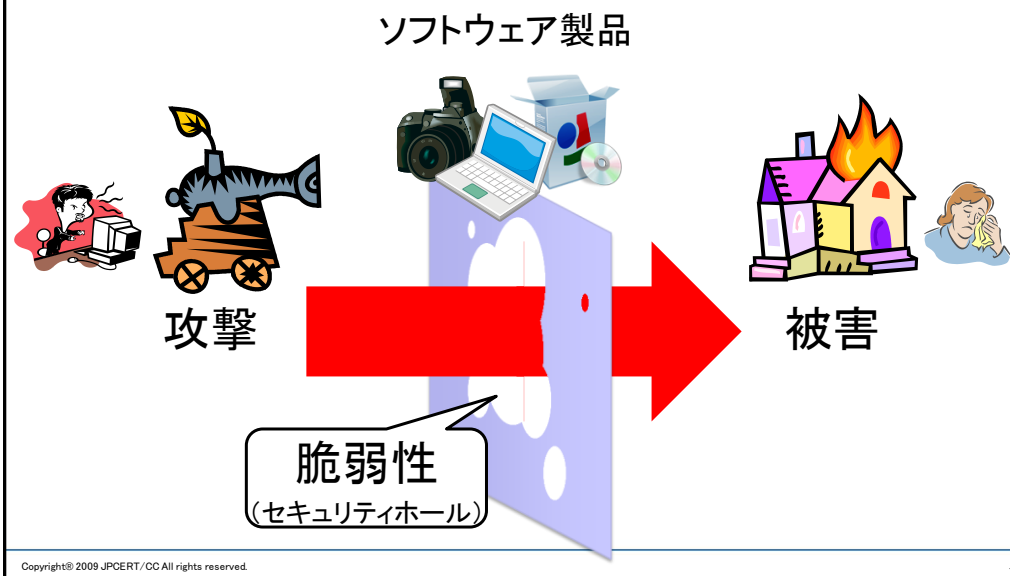
| 脆弱性/攻撃コード | 価格 | 情報ソース |
|---|---|---|
| 「ある」攻撃コード | 2200万円 – 2750万円 | 米政府当局者 |
| Internet Explorer | 660万円 – 1320万円 | H.D. Moore |
| Vista exploit | 550万円 | Raimund Genes, トレンドマイクロ |
| ZDI, iDefense purchase | 22万円 – 110万円 | David Maynor, SecureWorks |
| WMF exploit | 44万円 | David Maynor, SecureWorks |
| Microsoft Excel | ≧ 13万円 | Ebayオークション |
| Mozilla | 5万円 | Mozilla bug bounty |

出典：Charlie Miller *"The Legitimate Vulnerability Market"*
( http://weis07.infosecon.net/papers/29.pdf )

Copyright© 2009 JPCERT/CC All rights reserved.

## 攻撃は製品の脆弱性を突く

JPCERT CC®




Copyright© 2009 JPCERT/CC All rights reserved.


## 問題緩和に向けての取り組み

JPCERT CC®

- 出荷後のソフトウエア製品に発見される多くの脆弱性は、プログラミングエラーによって引き起こされている。

- 製品開発工程において、脆弱性につながるような欠陥を作りこまない取り組みが必要。
- 仮に作りこまれたとしても出荷前の検証等の段階で発見して対応が行われること等が有効な対策となり得る。

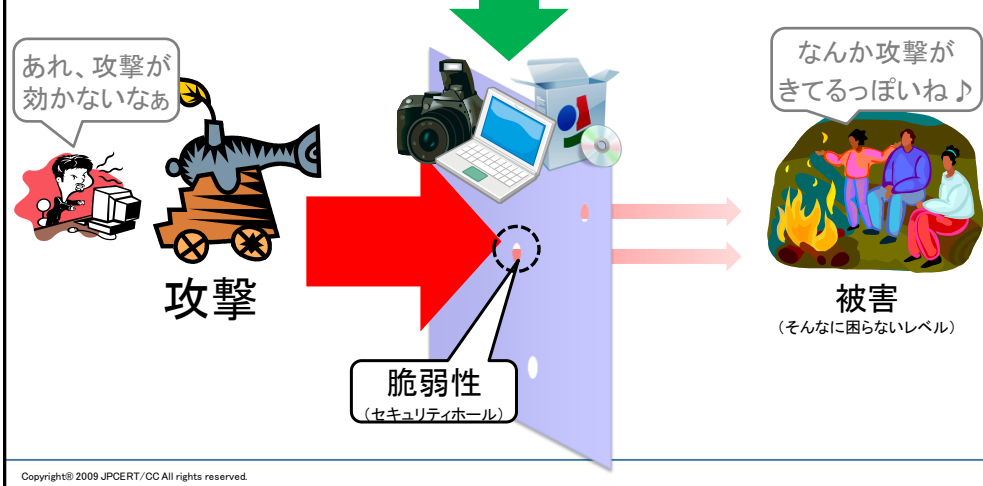
- 開発工程別、あるいは、開発プロセスを包含したよりセキュアな製品開発を実施するための幾つかの対策アプローチが存在する。
- ひとつの対策として「CERT C/C++ セキュアコーディングスタンダード」が整備、公開されている。

Copyright© 2009 JPCERT/CC All rights reserved.

## 対策は脆弱性を減らし、被害を小さくする

JPCERT CC®

セキュアな製品開発ための取り組み




Copyright© 2009 JPCERT/CC All rights reserved.



JPCERT CC®

## 2. C/C++ セキュアコーディング スタンダードの必要性

## C言語の精神："the Spirit of C"

JPCERT CC®

- プログラマを信頼する。
- プログラマが必要である事柄を行おうとすることを妨げない。

*JIS X 3010-1993 (ISO/IEC 9899:1990)
「プログラミング言語 C」解説より抜粋

C言語の仕様はJISC ( http://www.jisc.go.jp/app/JPS/JPSO0020.html )
で"X 3010" をキーワードに検索すると読める

Copyright© 2009 JPCERT/CC All rights reserved.


## クイズ：実行される puts() はどれ？

JPCERT CC®

```
signed char x, y;
x = -128;
y = -x;

if (x == y) puts("1");
if ((x - y) == 0) puts("2");
if ((x + y) == 2 * x) puts("3");
if (((char)(-x) + x) != 0) puts("4");
if (x != -y) puts("5");
```

多くの開発者が正解できない実態がある

Copyright© 2009 JPCERT/CC All rights reserved.

## コーディングエラーと脆弱性

JPCERT CC®

- プログラマが正しくC言語の挙動を理解せずにコードを書くと、コーディングエラーにつながります。
- コーディングエラーが原因で、様々な脆弱性が作り込まれてしまいます。

C言語の規格(JIS X 3010)をWebで閲覧
http://www.jisc.go.jp/app/pager?id=18190

Copyright© 2009 JPCERT/CC All rights reserved.


## 世界三大コーディングエラー

JPCERT CC®

**不適切な入力値検査**

・入力値の検査が行われていない、あるいは不十分であり、ソフトウェアが想定しない入力値を許してしまうエラー

**バッファオーバーフロー**

・確保されたバッファ領域外にデータの書き込みが行われることを許してしまうエラー

**不適切な整数利用**

・値がある整数型の最大値を超える、あるいは最小値を超えてしまうことを許してしまうエラー

Copyright© 2009 JPCERT/CC All rights reserved.

## 脆弱性につながるコーディングエラー

JPCERT CC®

- ◆ 無制限文字列コピー
- ◆ オフバイワンエラー
- ◆ NULL終端エラー
- ◆ 文字列の切り捨て
- ◆ スタック破壊
- ◆ ポインタ偽装
- ◆ メモリ割り当て関数の戻り値検査の誤り
- ◆ 解放済みメモリ領域の参照
- ◆ double free
- ◆ 整数オーバーフロー

- ◆無制限文字列コピー
- ◆符号エラー
- ◆整数の切り捨てエラー
- ◆フォーマットストリングの脆弱性
- ◆シンボリックリンクを利用した攻撃
- ◆文字列操作とパストラバーサル
- ◆temporary file への攻撃
- ◆unlink()競合攻撃

言語仕様を理解し、正しく駆使できるスキルが求められる

Copyright© 2009 JPCERT/CC All rights reserved.


## セキュアな C/C++ 開発を支援するガイドライン

JPCERT CC®

- C言語の精神と開発者のプロファイルにギャップ
- C言語仕様はコンパイラベンダ向けに記述されており、開発者が理解し辛い
- 開発者が、C/C++言語を安全に利用できるようサポートするガイドラインの必要性

***CERT C/C++ セキュアコーディングスタンダード***

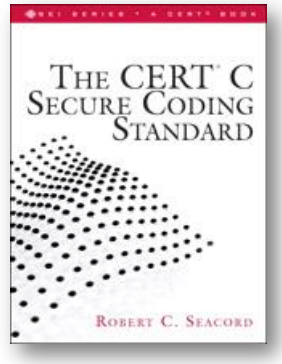
Copyright© 2009 JPCERT/CC All rights reserved.

## CERT C/C++ セキュア コーディングスタンダード

JPCERT CC®

https://www.securecoding.cert.org/

- 脆弱性を作り込まないためのコーディング規約や推奨される作法をサンプルと共に解説




Copyright® 2009 JPCERT/CC All rights reserved.



## CERT C/C++ セキュア コーディングスタンダード

JPCERT CC®

- C/C++言語を利用してセキュアな製品開発を行うため、米CERT/CCが中心となり整備されたコーディングスタンダード
- C言語用に約220、C++言語用に約250の”ルール”と”レコメンデーション”により構成
  - ◆ルールとは、CERT セキュアコーディングスタンダードへ適合する上で守るべきコーディングスタンダード
  - ◆レコメンデーションとは、ガイドラインあるいは好ましいコーディング作法として位置付け

Copyright® 2009 JPCERT/CC All rights reserved.

JPCERT CC®

# 3. ソースコード解析ツールを活用した CERT C/C++ セキュアコーディングスタンダードの有効性検証

## 有効性検証プロジェクトの概要・目的

JPCERT CC®

- 「CERT C/C++ セキュアコーディングスタンダード」の一部を実装した複数の「ソースコード解析ツール」を実際の製品開発プロジェクトに適用

- 「CERT C/C++ セキュアコーディングスタンダード」の有効性を評価
- 「CERT C/C++ セキュアコーディングスタンダード」への適合状況を機械的に効率よく検出することが可能であるか(実用性)を評価
- 利用された二つのソースコード解析ツールの比較評価は対象外

- 製品開発者が、より安全なソフトウエア製品を開発し提供するための対策の一つとして、本活動の結果を利用して頂く

Copyright© 2009 JPCERT/CC All rights reserved.

## ソースコード解析ツールの活用理由、期待効果

JPCERT CC®

- 受益者を開発者のみではなく、製品の品質向上を支援するソースコード解析ツール開発ベンダをも含めることで、相乗的な品質向上効果を狙う。

- ソースコード解析ツール開発ベンダは、本プロジェクトの成果を、製品の脆弱性に繋がるプログラミングエラー検出機能の向上に役立てることができ、より多くのコーディング不良を製品出荷前に検出できる製品を提供できる。

- 開発者は、CERTセキュアコーディングスタンダードを開発プロジェクトへ適用する、あるいは、同スタンダードへの違反／適合検査が可能なソースコード解析ツールの活用を通して品質向上に役立てることができる。

Copyright© 2009 JPCERT/CC All rights reserved.


## プロジェクトの体制、作業フェーズ、成果物

JPCERT CC®

| 作業フェーズ | CERT/CC | Fortify | LLNL | JPCERT/CC | SRA | 成果物 |
|---|---|---|---|---|---|---|
| 1. ツール選定 | ✓ | | | | | 対象ツール |
| 2. 拡張チェッカー開発 | ✓ | ✓ | ✓ | | | Fortify SCA用拡張チェッカー<br>ROSE用拡張チェッカー |
| 3. ソースコード評価データの収集 | | | | ✓ | ✓ | 評価用データ |
| 4. 有効性の評価 | ✓ | | | ✓ | | 評価レポート |

Copyright© 2009 JPCERT/CC All rights reserved.

## 拡張対象ソースコード解析ツールの選定（ツール選定）

JPCERT CC®

- 選定において有効性、拡張性、適合性を重視
  【有効性】既存のベストプラクティスを取り込んでいる
  【拡張性】拡張チェッカーの開発、追加が可能である
  【適合性】参加する開発者、開発プロジェクトのニーズに合う

- 『Fortify SCA』Fortify Software, Inc.
  http://www.fortifysoftware.co.jp/
- 『Compass/ROSE』Lawrence Livermore National Laboratory (LLNL)
  http://rosecompiler.org/

Copyright© 2009 JPCERT/CC All rights reserved.


## ソースコード解析ツールへのC/C++セキュアコーディングスタンダード実装（拡張チェッカー開発）

JPCERT CC®

- CERT C/C++セキュアコーディングスタンダードの中から、特に重要なもの、あるいは、ツールへの実装が容易であるものをピックアップして、二つのソースコード解析ツール用に実装
- 二つのツールにそれぞれ異なるセットのルール／レコメンデーションを実装

- Fortify SCA
  - C言語用　:38 のルール／レコメンデーション
  - C++言語用 :22 のルール／レコメンデーション
- Compass/ROSE
  - C言語用　:12 のルール／レコメンデーション
  - C++言語用 :15 のルール／レコメンデーション

Copyright© 2009 JPCERT/CC All rights reserved.

## 開発プロジェクトのC/C++セキュアコーディングスタンダードへの適合状況検査（評価データ収集）

JPCERT CC®

- 国内の製品開発者に検証のための複数の実製品開発プロジェクトを選定して頂き、これらプロジェクトに対しての、CERT C/C++セキュアコーディングスタンダードへの適合状況をソースコード解析ツールを利用して確認、同時にその有効性についての評価を実施

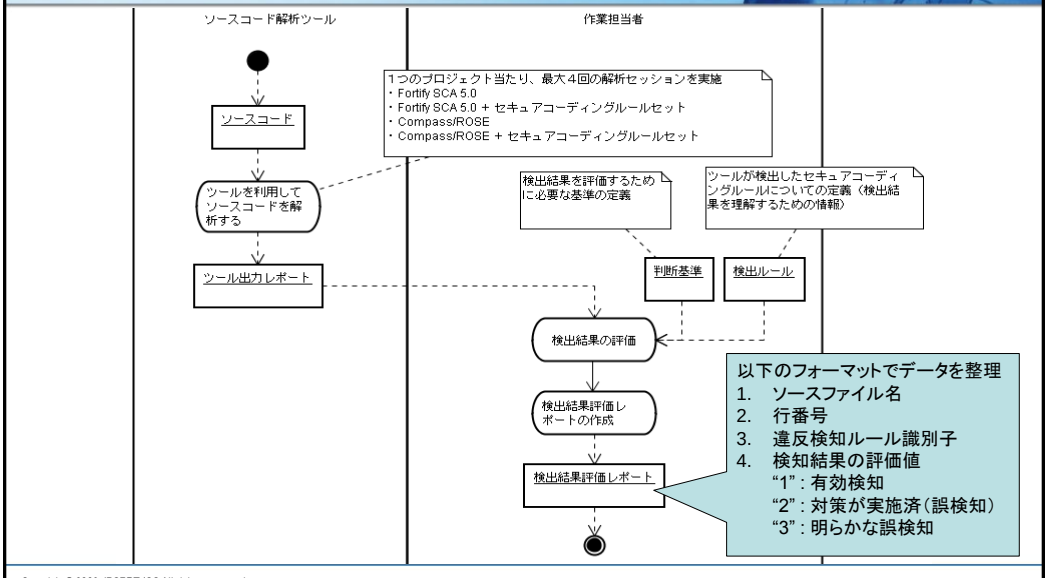
- 株式会社SRAにおいて、C言語、C++言語で書かれたプロジェクトそれぞれに対して、CERT C/C++セキュアコーディングが実装された二つのソースコード解析ツールを利用したソースコード分析作業を実施して頂いた

| 開発プロジェクト | 言語 | OS | コンパイラ | サイズ（KLOC） |
|---|---|---|---|---|
| 料金徴収システム関連のGUIアプリケーション | C++（Qt利用） | Miracle Linux | GCC 3.2.3 | 264 |
| 映像サービスの通信プロトコル | C | Cent OS（Linux） | GCC 3.4.6 | 30 |

Copyright® 2009 JPCERT/CC All rights reserved.


## 評価データ収集フロー

JPCERT CC®




Copyright® 2009 JPCERT/CC All rights reserved.

## 評価データ収集作業上の課題と対策

JPCERT CC®

- 違反検知されたコーディングスタンダードの内容解釈
  - 開発プロジェクト担当者間で内容解釈に差異が発生する可能性。
  - 英語で記述されているため、開発チーム内で共有するには最適ではない。

  ➡ *日本語による情報提供と、内容についての説明会の実施*

- ツールの誤検知、有効検知の判断
  - 誤検知を含むソース解析ツールの検知結果に対して、担当者がその妥当性の是非を判断できなければならない。
  - つまり、正しい判断を下すには、担当者は対象となる開発プロジェクトのソースコードと、違反が検知されたコーディングスタンダードの内容の両方を把握していることが重要。

  ➡ *プロジェクトに精通した担当者アサインと、スタンダードの説明*

Copyright© 2009 JPCERT/CC All rights reserved.


## ソースコード解析ツール検出結果の有効性評価（有効性の評価）

JPCERT CC®

- CERT C/C++セキュアコーディングスタンダードが実装された二つのソースコード解析ツールの検出結果の有効性を評価

| プロジェクト開発言語 | サイズ（KLOC） | ツール | 検出数 | 有効 | 無効 | 不明 | 有効検出率 | 補足 |
|---|---|---|---|---|---|---|---|---|
| C++言語 | 264 | Fortify | 186 | 125 | 61 | 0 | 67% | |
| | | ROSE | 200 | 19 | 51 | 130 | 27% | 全51の無効検出中43は障害が原因、70%まで改善可能 |
| C言語 | 30 | Fortify | 408 | 90 | 100 | 218 | 47% | |
| | | ROSE | 7 | 5 | 2 | 0 | 71% | |

- Compass/ROSE用に作成されたC++言語用のスタンダード検査部分の有効検出率が低い原因はソフトウエアの障害であり、修正することで同有効検出率は27%から70%へ大きく改善できる。

Copyright© 2009 JPCERT/CC All rights reserved.

## ソースコード解析ツール検出結果の有効性評価（有効性の評価）

JPCERT CC®

- CERT C/C++ セキュアコーディングスタンダードが実装されたソースコード解析ツールの活用を通して、脆弱性に繋がり得るプログラミングエラーを効果的に検出することができたと言える。
- 同スタンダードは、適合例／違反例のサンプルコードを含んだ形で提供されているため、ソースコード解析ツールを利用した分析作業を通して、セキュアコーディングプラクティスを学べる。

✓ CERT C/C++ セキュアコーディングスタンダードの有効性と、同スタンダードを既存のソースコード解析ツールと組み合わせで実用可能であることをサポート

✓ 既存のソースコード解析ツールが提供するプログラミングエラーの検出機能を更に向上できることをサポート

Copyright® 2009 JPCERT/CC All rights reserved.



## CERT C/C++ セキュアコーディングスタンダード対応ソースコード解析ツール

JPCERT CC®

- LDRA

  CERT C セキュアコーディングスタンダード version 1.0 をサポート

  約140のルール、レコメンデーションへの違反を検知できる

  http://www.fuji-setsu.co.jp/products/LDRA/TBsecure.html

  【各ルール、レコメンデーションへの対応状況】

  https://www.securecoding.cert.org/confluence/display/seccode/CERT+C+Rules+implemented+in+the+LDRA+tool+suite

**LDRA 社 TBsecure™ に CERT C コードの安全性ルールチェック機能を搭載して出荷開始**

Photocaption 1: Screenshot from the LDRA tool suite showing the selection of the CERT C secure coding standard from the C standards models

Photocaption 2: Screenshot from the Code Review Report within TBvision showing the CERT C model being applied

www.fuji-setsu.co.jp/files/TBsecure_CERT_C.pdf

Copyright® 2009 JPCERT/CC All rights reserved.

## CERT C/C++ セキュアコーディングスタンダード対応 ソースコード解析ツール

JPCERT CC®

- Compass/ROSE（オープンソース）
  ベータ版だが、C、C++ 合わせて約120のルール／レコメンデーションに部分的あるいは完全に対応可能であることが分かっている。Cへの対応が充実。
  【各ルール、レコメンデーションへの対応状況】
  https://www.securecoding.cert.org/confluence/display/seccode/ROSE+Checker+Code
  https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=9863304
- Fortify SCA 5.0
  CとC++ 合わせて少なくとも85のルール／レコメンデーションに部分的に対応
  【各ルール、レコメンデーションへの対応状況】
  https://www.securecoding.cert.org/confluence/display/seccode/C+Rules+Implemented+in+Fortify+SCA
  https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=7766071
- Coverity Prevent
  Cの約30のルール／レコメンデーションに部分的に対応
  【各ルール、レコメンデーションへの対応状況】
  https://www.securecoding.cert.org/confluence/display/seccode/Coverity+Prevent
  https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=7766071

Copyright© 2009 JPCERT/CC All rights reserved.


JPCERT CC®

# 4. 脆弱性の削減を目的としたJPCERT/CCの活動

～ JPCERT/CCのセキュアコーディングへの取組み ～

## C/C++ セキュアコーディングセミナー

JPCERT CC®

### ■ C/C++ セキュアコーディング ハーフデイキャンプ（午後半日）

一般向けオープンセミナとして開催

1. 文字列
2. 整数
3. 動的メモリ管理
4. ファイルI/O Part-1 ファイルシステム
5. ファイルI/O Part-2 パス名解決
6. ファイルI/O Part-3 競合状態
7. 書式指定文字列

詳しくは、以下URLをご覧ください。
http://www.jpcert.or.jp/event/index.html

2009年度は7月頃に開催予定

Copyright® 2009 JPCERT/CC All rights reserved.

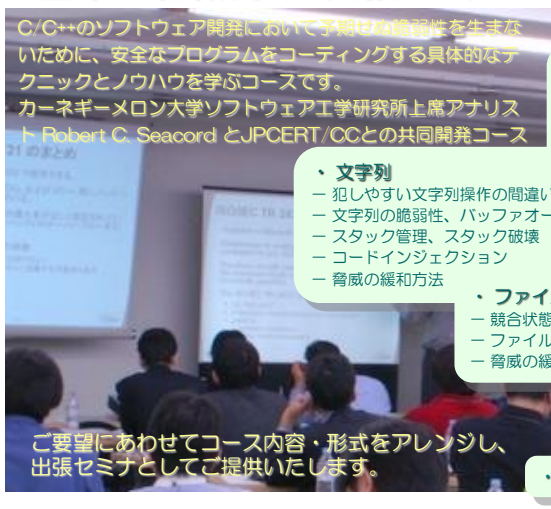

～ JPCERT/CCの脆弱性削減のための取組み ～

## C/C++ セキュアコーディング セミナー

JPCERT CC®

### ■ 企業／組織向け 出張セミナー

C/C++のソフトウェア開発において予期せぬ脆弱性を生まないために、安全なプログラムをコーディングする具体的なテクニックとノウハウを学ぶコースです。
カーネギーメロン大学ソフトウェア工学研究所上席アナリスト Robert C. Seacord とJPCERT/CCとの共同開発コース



ご要望にあわせてコース内容・形式をアレンジし、出張セミナとしてご提供いたします。

- **セキュアコーディング概論**
  - ソフトウェア脆弱性の事例
  - セキュリティの概念
  - セキュアコーディングの必要性
- **整数**
  - 整数のエラー条件
  - 整数オーバーフロー
  - 整数演算 代表的な脆弱性
  - 脅威の緩和方法
- **文字列**
  - 犯しやすい文字列操作の間違い
  - 文字列の脆弱性、バッファオーバーフロー
  - スタック管理、スタック破壊
  - コードインジェクション
  - 脅威の緩和方法
- **動的メモリ管理**
  - 動的メモリ管理における間違い
  - メモリマネージャにおける事例
  - 代表的な脆弱性
  - 脅威の緩和方法
- **ファイル入出力**
  - 競合状態、デッドロック
  - ファイルシステムへの攻撃
  - 脅威の緩和方法
- **書式指定文字列**
  - 書式指定出力関数、可変引数関数
  - 関数に対する攻撃
  - スタックのランダム化
  - 代表的な脆弱性、脅威の緩和方法
- **実機を使ったハンズオン演習（準備中）**

Copyright® 2009 JPCERT/CC All rights reserved.

## セミナー講義資料のサンプル

JPCERT CC®

バッファオーバーフローが発生する仕組みを、スタックの基礎から分かりやすく解説

### まとめ：呼び出された関数が実行中のスタックの状態

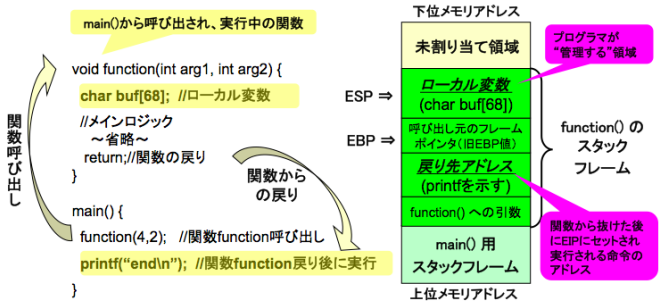
main()から呼び出され、実行中の関数

```
void function(int arg1, int arg2) {
  char buf[68];  //ローカル変数
  //メインロジック
    ～省略～
  return;//関数の戻り
}

main() {
  function(4,2);   //関数function呼び出し
  printf("end\n");  //関数function戻り後に実行
}
```

関数呼び出し

関数からの戻り




EIP ：インストラクションポインタ／プログラムカウンタ （実行される命令アドレスを示す）
ESP：スタックポインタ （スタックの一番上を示す）
EBP：ベースポインタ （データ格納領域の基点を示す）

Copyright © 2009 JPCERT/CC All rights reserved.
Copyright © 2007 JPCERT/CC All rights reserved.
48
JPCERT CC®

Copyright® 2009 JPCERT/CC All rights reserved.


## セミナー講義資料のサンプル

JPCERT CC®

整数取り扱いに関する単純なコーディングミスが重大な脆弱性につながった事例をコードで解説

### 整数オーバーフローの例

```
void getComment(unsigned int len, char *src) {
  unsigned int size;
  size = len - 2;
  char *comment = (char *)malloc(size + 1);
  memcpy(comment, src, size);
  return;
}
int main(void) {
  getComment(1, "Comment ");
  return 0;
}
```

0 バイトの malloc() が成功

大きな正の値 (0xffffffff) として解釈される

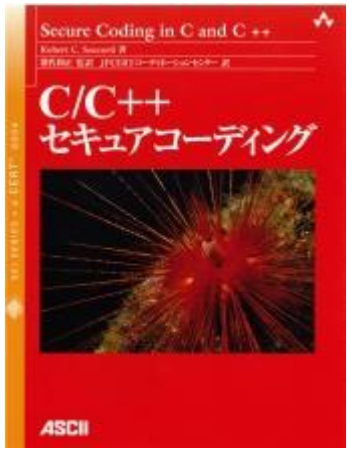
コメント長フィールドの値として 1 を持つ画像を作成することで、オーバーフローを引き起こす可能性がある。

Copyright © 2009 JPCERT/CC All rights reserved.
124
JPCERT CC®

Copyright® 2009 JPCERT/CC All rights reserved.

## 書籍『C/C++セキュアコーディング』

JPCERT CC®




- C/C++において脆弱性に繋がる一般的なプログラミングエラー、脆弱性を利用した攻撃、脅威緩和のための対策を解説
- C/C++の話だけでなく、それを取り巻く仕組みにもふれているのでコンピュータアーキテクチャの勉強にもなります

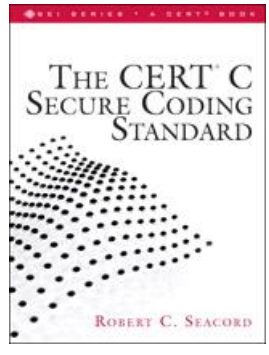
Copyright© 2009 JPCERT/CC All rights reserved.


## CERT C Secure Coding Standards 日本語版

JPCERT CC®

http://www.jpcert.or.jp/sc-rules/

- 脆弱性を作り込まないためのコーディング規約や推奨される作法をサンプルと共に解説




Copyright© 2009 JPCERT/CC All rights reserved.

# 日本語翻訳版のサンプルイメージ

JPCERT CC®

- 00. はじめに
- 01. プリプロセッサ (PRE)
- 02. 宣言と初期化 (DCL)
- 03. 式 (EXP)
- 04. 整数 (INT)
- 05. 浮動小数点 (FLP)
- 06. 配列 (ARR)
- 07. 文字と文字列 (STR)
- 08. メモリ管理 (MEM)
- 09. 入出力 (FIO)
- 10. 環境 (ENV)
- 11. シグナル (SIG)
- 12. エラー処理 (ERR)
- 49. Misc. (MSC)
- 50. POSIX (POS)
- AA. C References
- XX. お問い合わせ

**04. 整数 (INT)** 最終更新: 2009-4-10

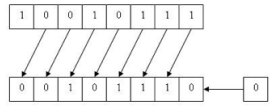
## INT34-C. 負のビット数、あるいはオペランドのビット数以上シフトしないこと

ビット単位のシフトには、シフト式 << 加減式 形式の左シフト演算と、シフト式 >> 加減式 形式の右シフト演算がある。オペランド(各オペランドは整数型)に対して整数格上げが行われる。シフトした結果の型は、格上げされた左オペランドの型である。右オペランドの値が負または格上げされた左オペランドのビット幅以上である場合、シフト動作は未定義となる。

ほとんどすべての場合、負のビット数またはオペランドのビット数以上のシフト演算はバグ(ロジックエラー)である。これは、単なる整数表現上の欠陥であるオーバーフローとは異なる。(INT32-C. Ensure that operations on signed integers do not result in overflow (符号付き整数演算がオーバーフローを引き起こさないようにすること)を参照)

### ルールに違反したコード例 (左シフト、符号付きの型)

E1 << E2 の結果は、E1 を E2 ビット左にシフトした値であり、空いたビットは 0 で埋められる。E1 が符号付きの型で非負の値であり、E1 * 2 $^{E2}$ が結果の型で表現できる場合、それが結果の値となる。それ以外の場合の動作は未定義である。




次のコードは、左オペランドおよび右オペランドの値が負ではないこと、および右オペランドが格上げされた左オペランドのビット幅以下であることを確認していないため、未定義の動作を引き起こす可能性がある。

```
int si1;
int si2;
int sresult;

/* si1 と si2 を初期化 */

sresult = si1 << si2;
```

シフト演算子やその他のビットワイズ演算子は、INT13-C. Use bitwise operators only on unsigned operands にしたがって、符号なし整数にたいしてのみ使用すべきである。

Copyright® 2009 JPCERT/CC All rights reserved.



JPCERT CC®

# ご清聴ありがとうございました。

◆本プロジェクトの成果論文(英語版、日本語版)
http://www.jpcert.or.jp/research/

◆セキュアコーディングセミナに関する情報
http://www.jpcert.or.jp/event/

◆セキュアコーディングセミナの講義資料のダウンロード
http://www.jpcert.or.jp/research/materials.html

◆JPCERT/CC Webページ
http://www.jpcert.or.jp/

◆問い合わせ先
email : secure-coding@jpcert.or.jp