オープンソースの「今」を伝える
オープンソースカンファレンス
2011 Nagoya

オープンソースカンファレンス 2011 Nagoya

セキュアコーディングノススメ（Java編）

2011年08月20日

戸田 洋三

JPCERT コーディネーションセンター

secure-coding@jpcert.or.jp

JPCERT CC®

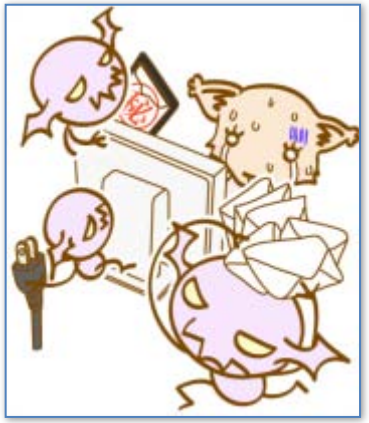
アジェンダ

0. JPCERT/CC とは

1. OSSにもセキュアコーディングを!

～OSSとセキュリティをめぐる状況～

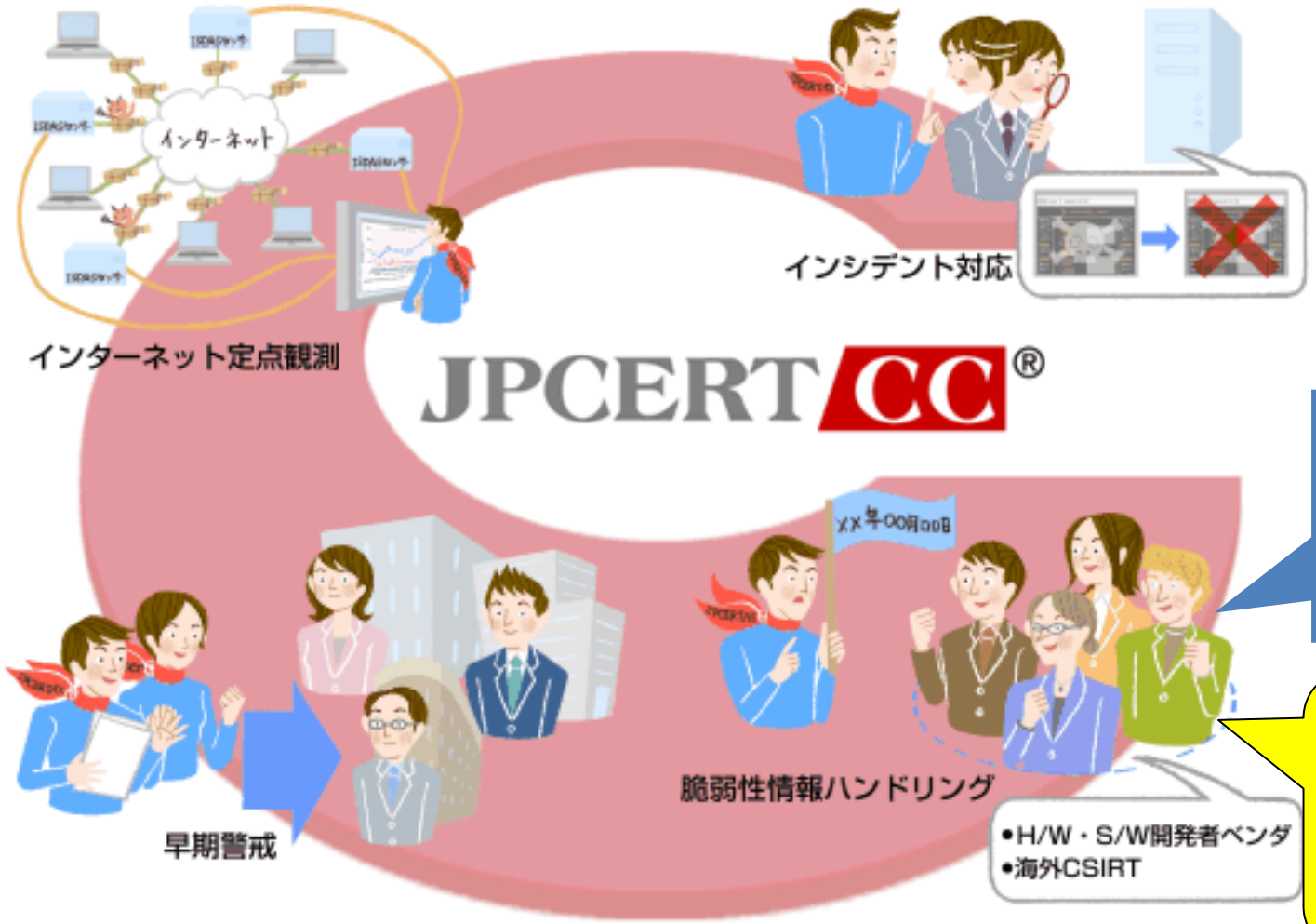
2. セキュアコーディングスタンダード for Java



はじめに: JPCERT/CCとは

- JPCERT/CC
  - **JP**　　: 日本
  - **C**omputer　: コンピュータ
  - **E**mergency : 緊急
  - **R**esponse　: 対応
  - **T**eam　　: チーム
  - **C**oordination : コーディネーション
  - **C**enter　　: 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

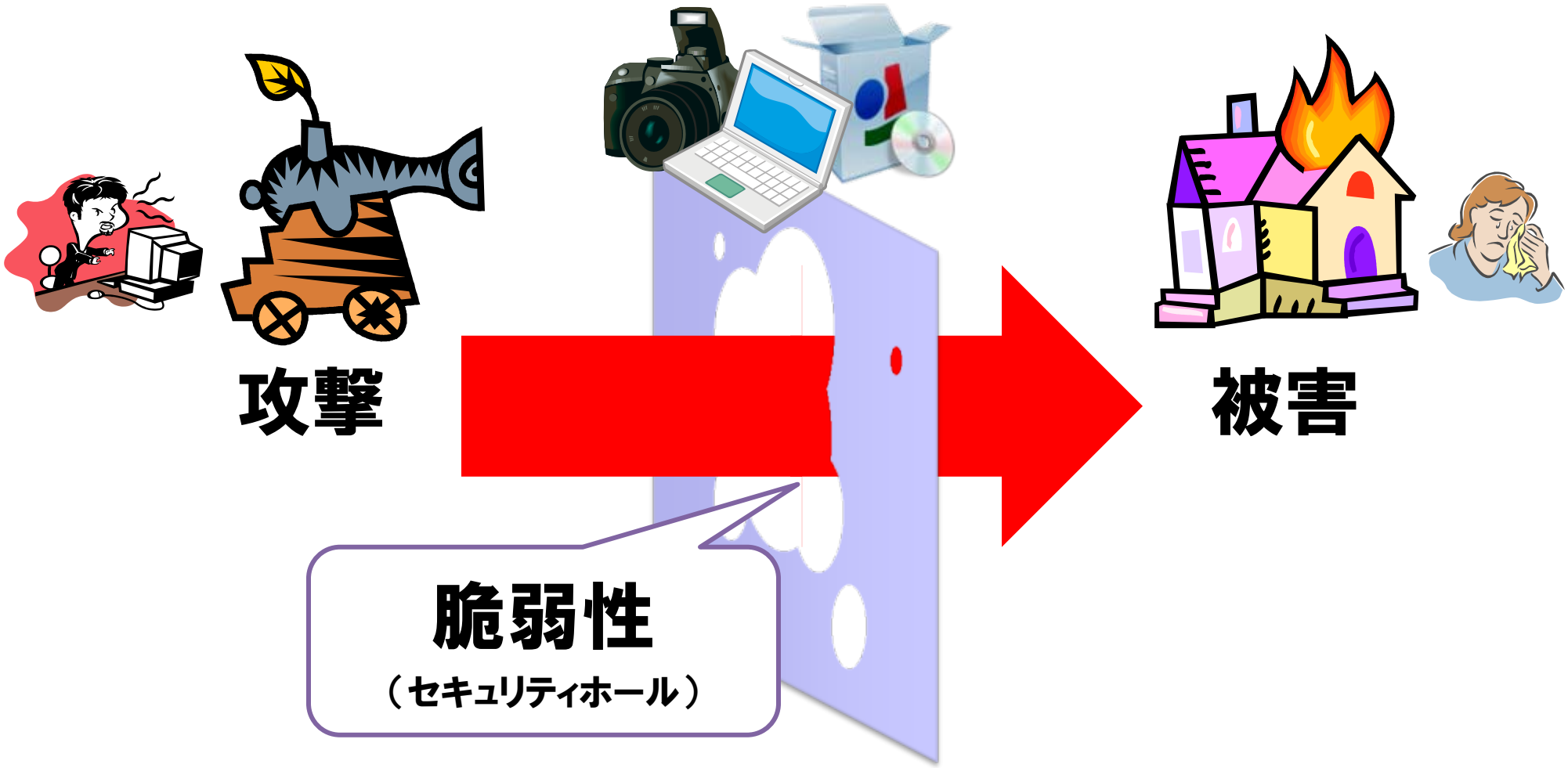
はじめに: JPCERT/CCの主な活動(1)



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

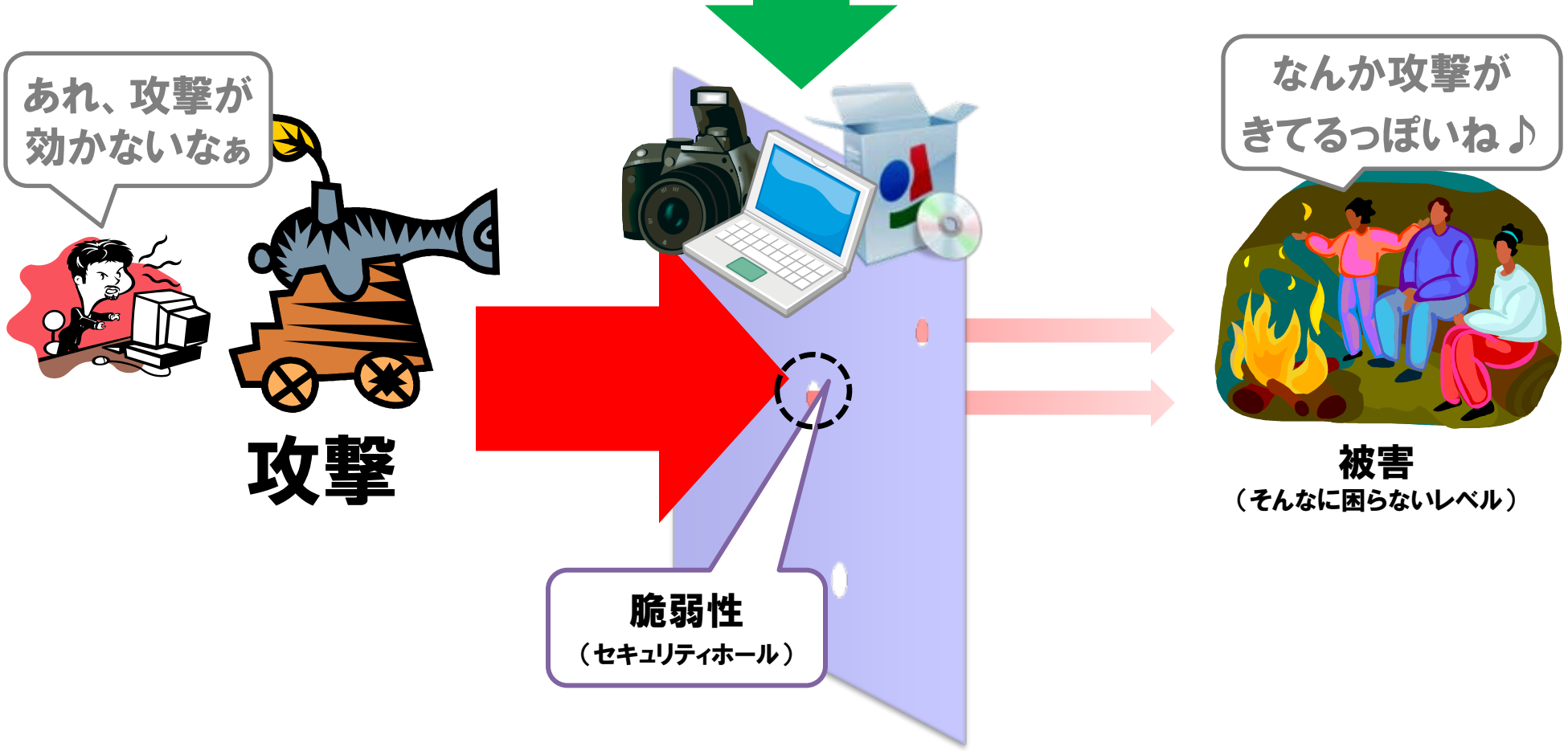
セキュアコーディングに関する啓発活動

製品の脆弱性を突いて攻撃が行われる



脆弱性を減らし、被害を小さくする

セキュアな製品開発のための取り組み



はじめに: JPCERT/CCの主な活動(2)

脆弱性情報ハンドリング, JVN 運営

JVN Japan Vulnerability Notes

セキュアコーディングの普及啓発活動

2006年度:『C/C++セキュアコーディング』出版

2007年度: セミナ事業開始

2008年度：32回講演、受講者2250名

2009年度: 1000名以上の方が受講

『CERT Cセキュアコーディングスタンダード』出版

海外(ASEAN諸国)でもセミナを実施

2010年度: 福岡、大阪、名古屋、札幌、東京でセミナ実施

ASEAN諸国でセミナを実施

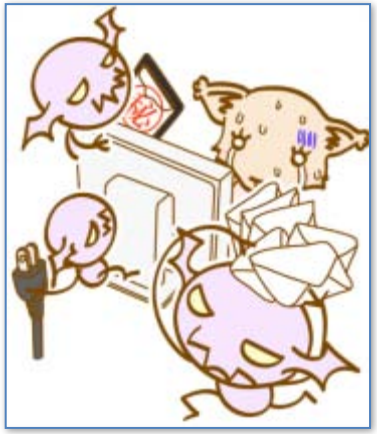
アジェンダ

0. JPCERT/CC とは

1. OSSにもセキュアコーディングを!

～OSSとセキュリティをめぐる状況～

2. セキュアコーディングスタンダード for Java

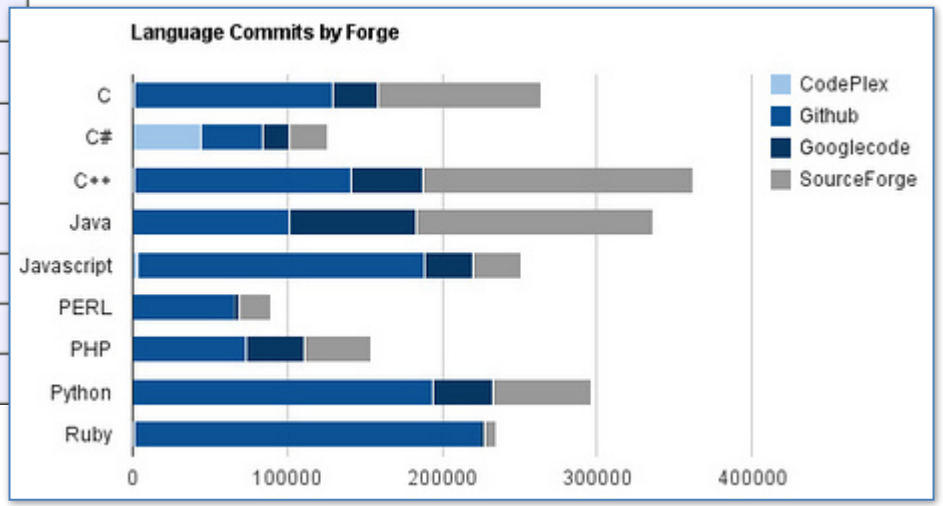


人気、使用率ともに世界的に高い

- TIOBE Programming Index: 2011年8月の指標 Java,C,C++で44%のシェア
- sourceforge.net のプロジェクト数: Java 47267, C 28544, C++ 37756

Position Aug 2011	Position Aug 2010	Delta in Position	Programming Language	Ratings Aug 2011	Delta Aug 2010	Status
1	1	=	Java	19.409%	+1.42%	A
2	2	=	C	17.390%	-0.48%	A
3	3	=	C++	8.433%	-1.23%	A
4	4	=	PHP	6.134%	-3.05%	A
5	6	↑	C#	6.042%	+1.06%	A
6	9	↑↑↑	Objective-C	5.494%	+2.34%	A
7	5	↓↓	(Visual) Basic	5.013%	-0.40%	A
8	7	↓	Python	3.415%	-0.81%	A
9	8	↓	Perl	2.315%	-1.11%	A
10	11	↑	JavaScript	1.557%	-0.84%	A

TIOBE Programming Community Index for August 2011



http://redmonk.com/sogrady/2011/06/02/blackduck-webinar/

みんな大好き！CとC++とJava(2)

Language Breakdown in Open Source

All open source project releases

Rank	Language	%
1	C	44.47
2	C++	13.09
3	Java	9.25
4	Javascript	6.15
5	Shell	6.08

Releases within the last 12 months

Rank	Language	%
1	C	41.04
2	C++	12.43
3	Javascript	9.01
4	Java	8.63
5	PHP	6.31

BlackDuck Software のデータから
(54万のOSSプロジェクトが調査対象)

OSS開発言語としてもC/C++/Java のシェアは大きい

出典: http://www.blackducksoftware.com/oss/projects#language
(August, 2011)

開発企業のコスト削減を支えるOSS

Black Duck の分析によると (2009年11月10日公表)

- ソフトウェア製品/アプリの22%はOSSで構成されている
- これは$2600万ドル(約26億円) / 製品の価値がある

より効率的、低コストで開発するためにOSSは欠かせない存在

Commercial Use of Open Source Software Drives Significant Cost Savings for Enterprises and Development Organizations, says Black Duck Analysis

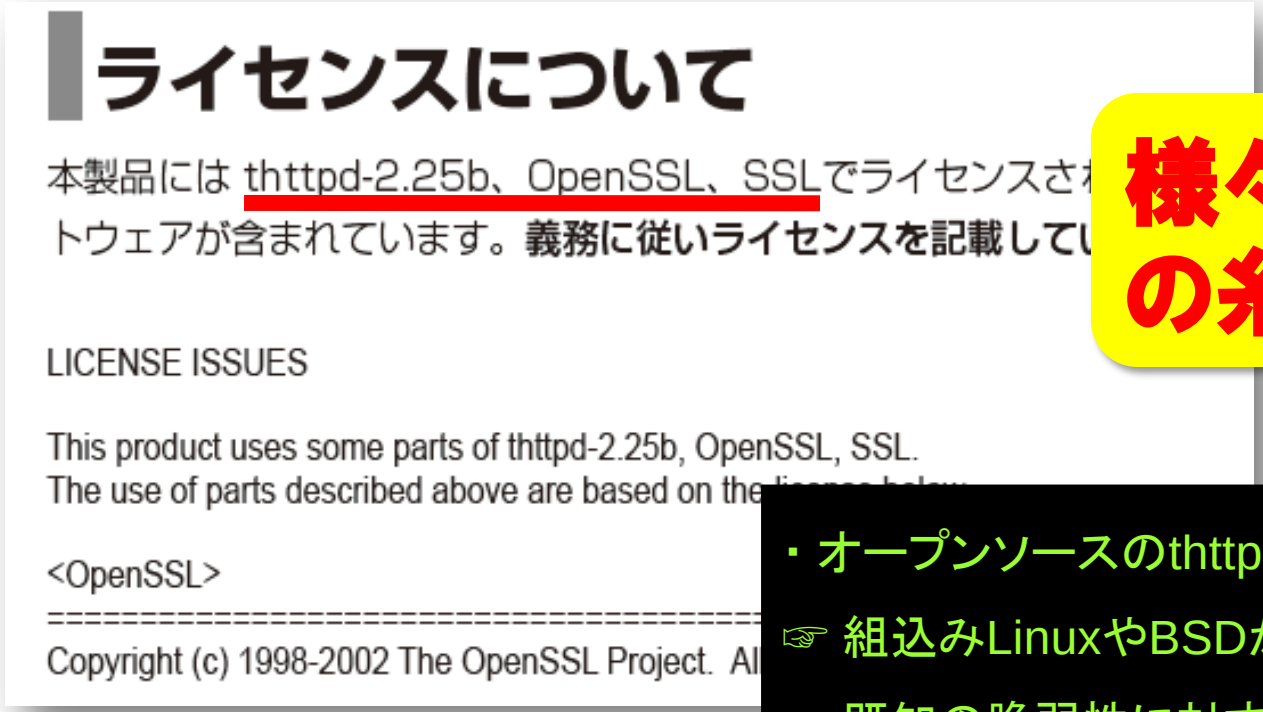
The average software product or application contains 22 percent open source software with an estimated value of $26 million per product or application

WALTHAM, MA - November 10, 2009 - Businesses and organizations implementing multi-source development with open source software benefit from significant cost savings and improved application development efficiencies, according to a recent survey of enterprise development organizations conducted by Black Duck Software. Black Duck is the leading global provider of products and services for accelerating software development through the managed use of open source software.

http://www.blackducksoftware.com/news/releases/2009-11-10

もしOSSに脆弱性が見つかったら…

CE機器にも利用されるOSS



ライセンスについて

本製品には thttpd-2.25b、OpenSSL、SSLでライセンスさ
トウェアが含まれています。義務に従いライセンスを記載して

LICENSE ISSUES

This product uses some parts of thttpd-2.25b, OpenSSL, SSL.
The use of parts described above are based on the

<OpenSSL>

Copyright (c) 1998-2002 The OpenSSL Project. Al

様々な情報が攻撃の糸口を与える

・オープンソースのthttpdやOpenSSLが使われている

☞ 組込みLinuxやBSDが使われていそう

☞ 既知の脆弱性に対する対策はされているだろうか？

**thttpdには過去16件の脆弱性が見つかっている。
そのうち4件は2.25bに影響あり。**

OpenSSLには過去79件の脆弱性が見つかっている。

OSSプログラマに対して高まる潜在的期待

- 様々なOSSのOS, ライブラリ、アプリがメジャーな製品に利用される世の中
- オープンであるがゆえ脆弱性の研究や攻撃の対象となりやすい
- OSSへの社会的依存とそのインパクトは小さくない

☞セキュアなOSSが求められる世の中

- OSSの脆弱性が原因で、X社が数千万円のコストをかけて修正パッチをエンドユーザに配ったり (cf. 自動車のリコール)
- 様々なソフトウェア製品のユーザをセキュリティ上のリスクにさらす可能性

☞セキュアコーディングできるOSSプログラマが求められている

攻撃の多くはアプリケーションレイヤーを狙っている

- 攻撃者はネットワークやシステムのセキュリティを破ろうとするのではなく、アプリケーションソフトをターゲットにしてセキュリティを破ろうとする
  - 攻撃の75%はアプリケーションレベルで発生
  - 攻撃可能な脆弱性の大半はセキュアコーディングしなかったのが原因

「ソフトウェアに脆弱性が存在する ≠ 必ず攻撃される」とは言うものの、アプリレイヤーは狙われている

参考：OWASP 2009 Washington DCにおけるJoe Jarzombeck氏の講演

Java言語とセキュリティ

- C/C++と同様、言語仕様の脆弱性が存在
  - JVM仕様の実装依存部分など、プログラマが知った上でコーディングできる必要がある
- 誤用しやすいAPIの存在
  - C/C++の標準ライブラリ関数と同様、プログラマが間違って使うと脆弱性につながる可能性がある
  - Java 外部との適切なやりとりが必要(ファイルシステムの扱いなど)
- 言語のセキュリティ機能が複雑で誤用しやすい

C/C++と同様、プログラマが正しいセキュアコーディングの知識を持った上でJava言語を使わないと、脆弱性につながる

アンドロイドアプリの脆弱性とその影響

- 入力値検査の不備
- 権限管理の不備
- リソース浪費
- ……

- サービス運用妨害
- 情報流出
- マルウェア感染
- ……

アンドロイドアプリの脆弱性

thomascannon.net

Android Data Stealing Vulnerability

TUE NOV 23RD

While doing an application security assessment one evening I found a general vulnerability in Android which allows a malicious website to get the contents of any file stored on the SD card. It would also be possible to retrieve a limited range of other data and files stored on the phone using this vulnerability.

The vulnerability is present because of a combination of factors. I've been asked nicely to remove some details from the following section, and as my intention is to inform people about the risk, not about how to exploit users, I've agreed:

- The Android browser doesn't prompt the user when downloading a file, for example `"payload.html"`, it automatically downloads to `/sdcard/download/payload.html`
- It is possible, using JavaScript, to get this payload to automatically open, causing the browser to render the local file.
- When opening an HTML file within this local context, the Android browser will run JavaScript without prompting the user.
- While in this local context, the JavaScript is able to read the contents of files (and other data).

Then, once the JavaScript has the contents of a file it can post it back to the malicious website. This is a simple exploit involving JavaScript and redirects, meaning it should also work on multiple handsets and multiple Android versions without any effort.

One limiting factor of this exploit is that you have to know the name and path of the file you want to steal. However, a number of applications store data with consistent names on the SD card, and pictures taken on the camera are stored with a consistent naming convention too. It is also not a root exploit, meaning it runs within the Android sandbox and cannot grab all files on the system, only those on the SD card and a limited number of others.

SDカード上のファイルの取り扱いに関連して情報漏洩の可能性

アンドロイドアプリの脆弱性

Android Class Loading Hijacking

Updated: 29 Jun 2011 | Translations available: 日本語

Mario Ballano SYMANTEC EMPLOYEE

+2
2 Votes

Symantec. | Official Blog

We have been taking a close look at Android threats since they first appeared, looking for ways to analyze and classify them, as well as looking at possible attack vectors they may use in the near future. Some of our research has uncovered how Android applications could potentially exploit other installed applications to steal their private information or execute malicious code. In particular, we came across something that resembles Windows DLL Hijacking. Bear in mind that we are not talking about Android vulnerabilities per se, but application-specific issues. We found a few applications in the Google Android Marketplace that were susceptible to this attack and have notified the application developers accordingly.

Android provides APIs that allow an application to dynamically load code to be executed. For example, an application may support plug-ins that are downloaded and then loaded at a later time. Unfortunately, if these plug-ins are stored in an insecure location, this process can be hijacked, allowing access to private data and unexpected arbitrary code execution by malicious applications. Two classes allow the loading of additional code:

Public Constructors

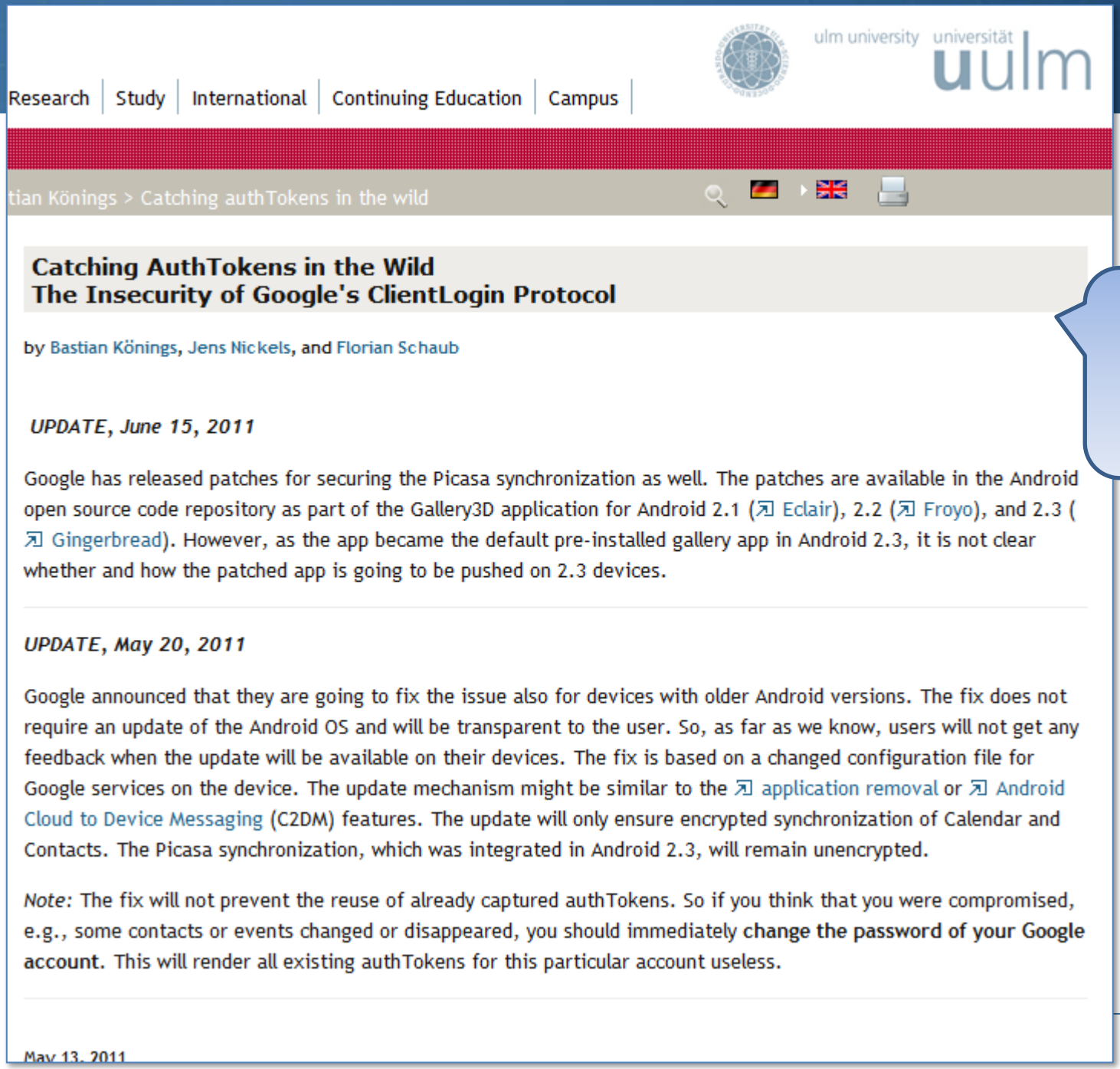
DexClassLoader (String dexPath, String dexOutputDir, String libPath, ClassLoader parent)

Creates a DexClassLoader that finds interpreted and native code.

PathClassLoader (String path, String libPath, ClassLoader parent)

Creates a PathClassLoader that operates on two given lists of files and directories.

AndroidアプリにもDLLローディングの問題が!



Research | Study | International | Continuing Education | Campus

ulm university universität uulm

tian Könings > Catching authTokens in the wild

Catching AuthTokens in the Wild
The Insecurity of Google's ClientLogin Protocol

by Bastian Könings, Jens Nickels, and Florian Schaub

UPDATE, June 15, 2011

Google has released patches for securing the Picasa synchronization as well. The patches are available in the Android open source code repository as part of the Gallery3D application for Android 2.1 (Eclair), 2.2 (Froyo), and 2.3 (Gingerbread). However, as the app became the default pre-installed gallery app in Android 2.3, it is not clear whether and how the patched app is going to be pushed on 2.3 devices.

UPDATE, May 20, 2011

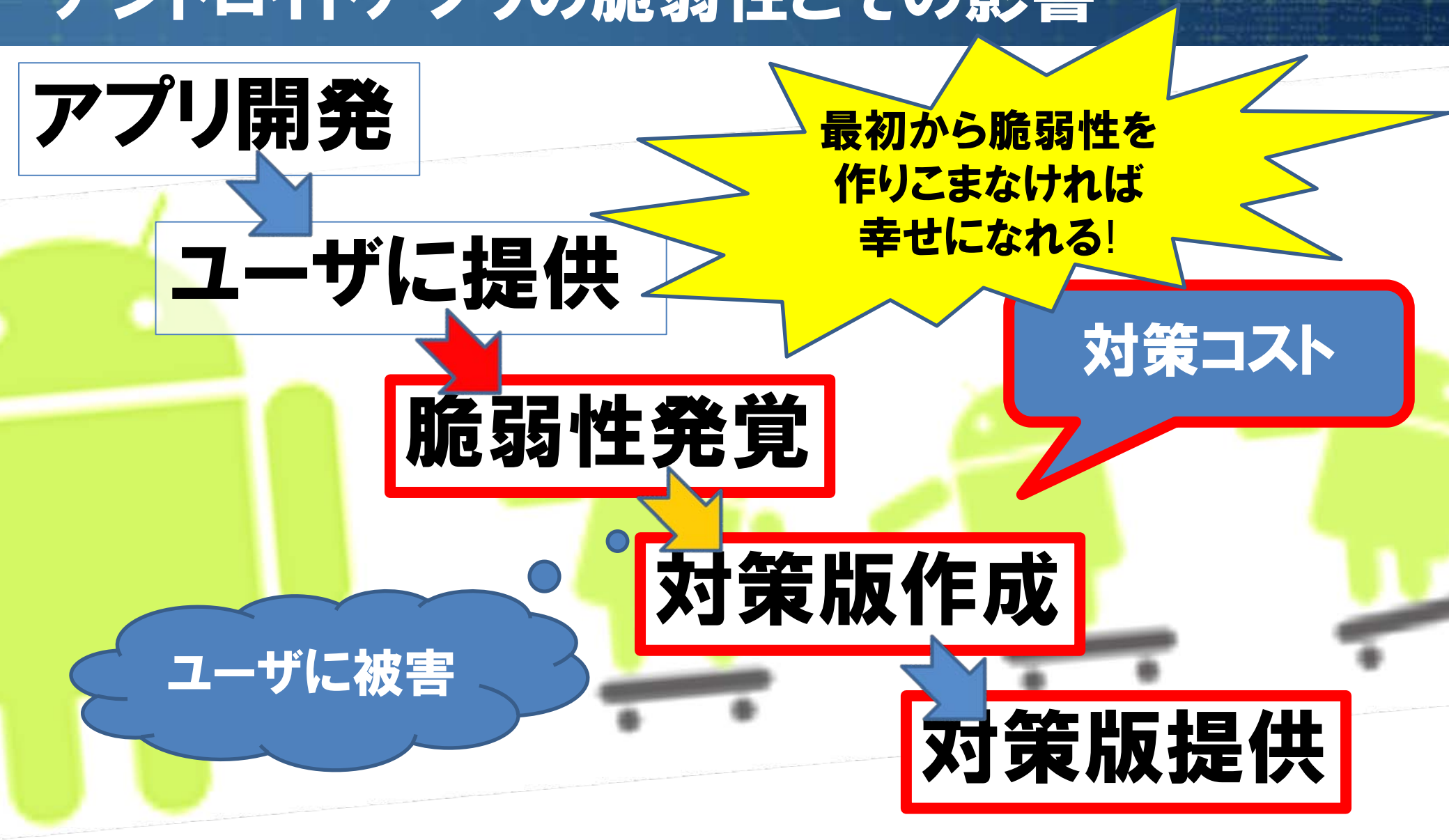
Google announced that they are going to fix the issue also for devices with older Android versions. The fix does not require an update of the Android OS and will be transparent to the user. So, as far as we know, users will not get any feedback when the update will be available on their devices. The fix is based on a changed configuration file for Google services on the device. The update mechanism might be similar to the application removal or Android Cloud to Device Messaging (C2DM) features. The update will only ensure encrypted synchronization of Calendar and Contacts. The Picasa synchronization, which was integrated in Android 2.3, will remain unencrypted.

Note: The fix will not prevent the reuse of already captured authTokens. So if you think that you were compromised, e.g., some contacts or events changed or disappeared, you should immediately **change the password of your Google account.** This will render all existing authTokens for this particular account useless.

May 13, 2011

セッション情報の取り扱いに問題が…

アンドロイドアプリの脆弱性とその影響



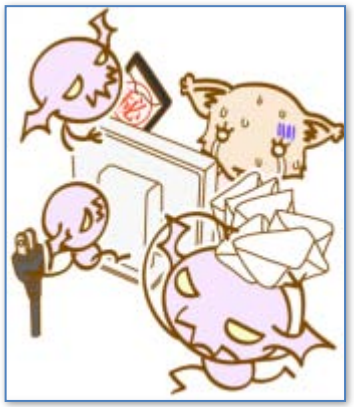
アジェンダ

0. JPCERT/CC とは

1. OSSにもセキュアコーディングを!

   ～OSSとセキュリティをめぐる状況～

2. セキュアコーディングスタンダード for Java



Java のセキュアコーディングガイドラインといえば…

Oracle Secure Coding Guidelines for the Java Programming Language, V3.0

The CERT Oracle Secure Coding Standard for Java

Oracle Secure Coding Guidelines for Java

Oracle Secure Coding Guidelines for the Java Programming Language, V3.0

http://www.oracle.com/technetwork/java/seccodeguide-139067.html

0. Fundamentals
1. Accessibility and Extensibility
2. Input and Output Parameters
3. Classes
4. Object Construction
5. Serialization and Deserialization
6. Access Control

2009年公開
言語機能に関する基本的な事項

Secure Coding Guidelines for the Java Programming Language, Version 3.0

The CERT Oracle Secure Coding Standard for Java

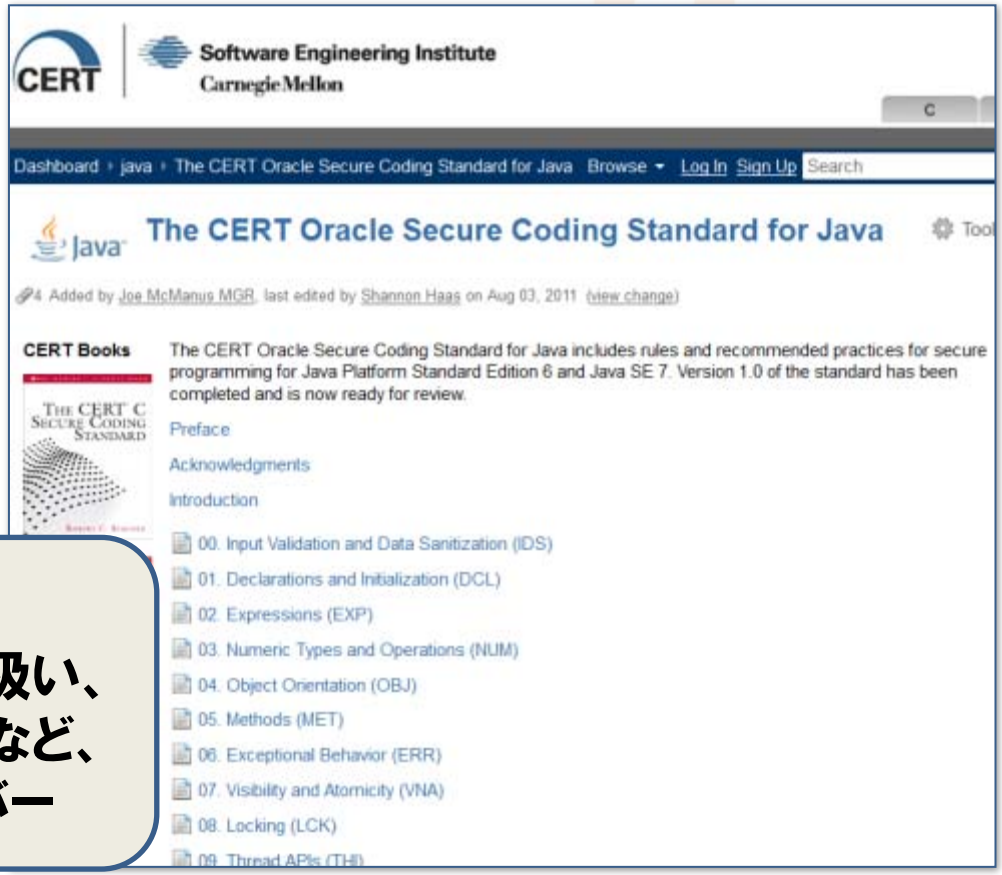
The CERT Oracle Secure Coding Standard for Java

https://www.securecoding.cert.org/confluence/display/java/The+CERT+Oracle+Secure+Coding+Standard+for+Java

https://www.securecoding.cert.org/confluence/x/Ux

00. **入力値検査とデータ無害化** (IDS)
01. **宣言と初期化** (DCL)
02. **式** (EXP)
03. **数値型と操作** (NUM)
04. **オブジェクト** (OBJ)
05. **メソッド** (MET)
06. **例外時の動作** (ERR)
07. **可視性と原子性** (VNA)
08. **ロック** (LCK)
09. **スレッドAPI** (THI)
10. **スレッドプール** (TPS)
11. **スレッド安全性** (TSM)
12. **入出力** (FIO)
13. **シリアライゼーション** (SI
14. **プラットフォームセキュリ**
15. **実行環境** (ENV)
49. **雑則** (MSC)

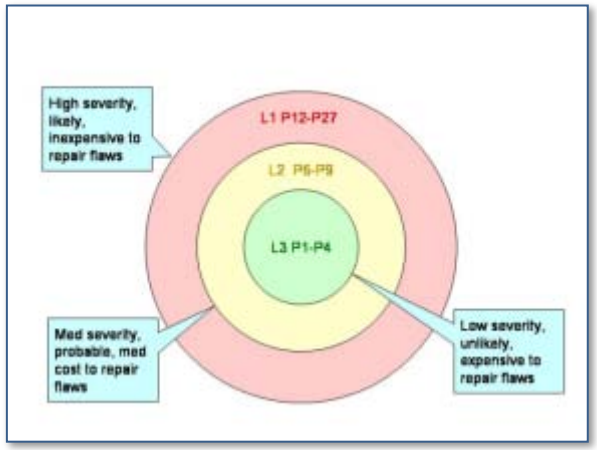
現在も発展中
言語機能、スレッドの扱い、
並行性、入出力処理など、
より広い範囲をカバー



Secure Coding Standard for Java (1)

「CERT Oracle セキュアコーディングスタンダード for Java」とは?

- Javaプログラマ向けの(プラットフォームに依存しない)ガイドライン
- Java SE 6 & 7 API に基づく
- 間違いやすいコードパターン、注意すべきAPIの使い方など
- 解説とともに、違反コード例と適合コード例を提示

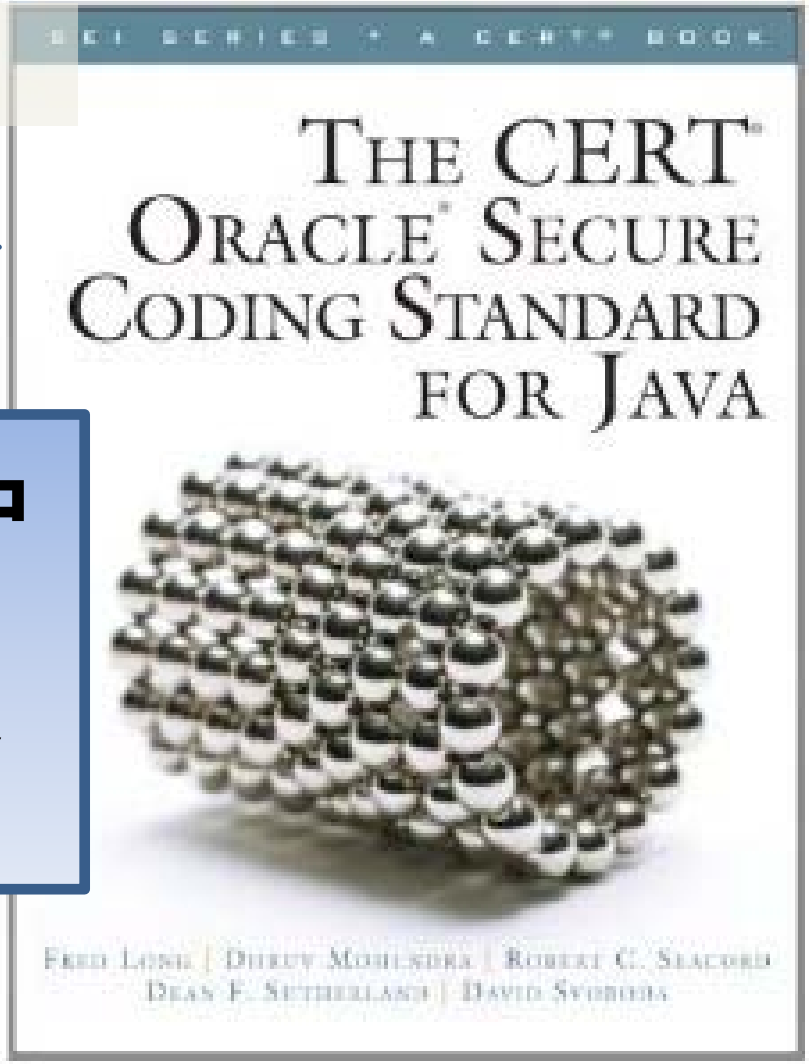


Javaを使うAndroidアプリ
開発者にも役に立つ!

Secure Coding Standard for Java (2)

CERT/CC のWikiサイト上で開発中

JavaOne2011に合わせ、v1.0 を書籍として出版



JPCERT/CC にて日本語版作成中

- Concurrency Guideline日本語版
- 全てのガイドラインの日本語版

Java言語とセキュリティ(再掲)

- C/C++と同様、言語仕様の脆弱性が存在
  - JVM仕様の実装依存部分など、プログラマが知った上でコーディングできる必要がある

IDS, EXP, METカテゴリのガイドラインなど

- 誤用しやすいAPIの存在

IDS, EXP, FIOカテゴリのガイドラインなど

  - C/C++の標準ライブラリ関数と同様、プログラマが間違って使うと脆弱性につながる可能性がある
  - Java 外部との適切なやりとりが必要(ファイルシステムの扱いなど)
- 言語のセキュリティ機能が複雑で誤用しやすい

ERR,SECカテゴリのガイドラインなど

セキュアコーディングスタンダードの活用

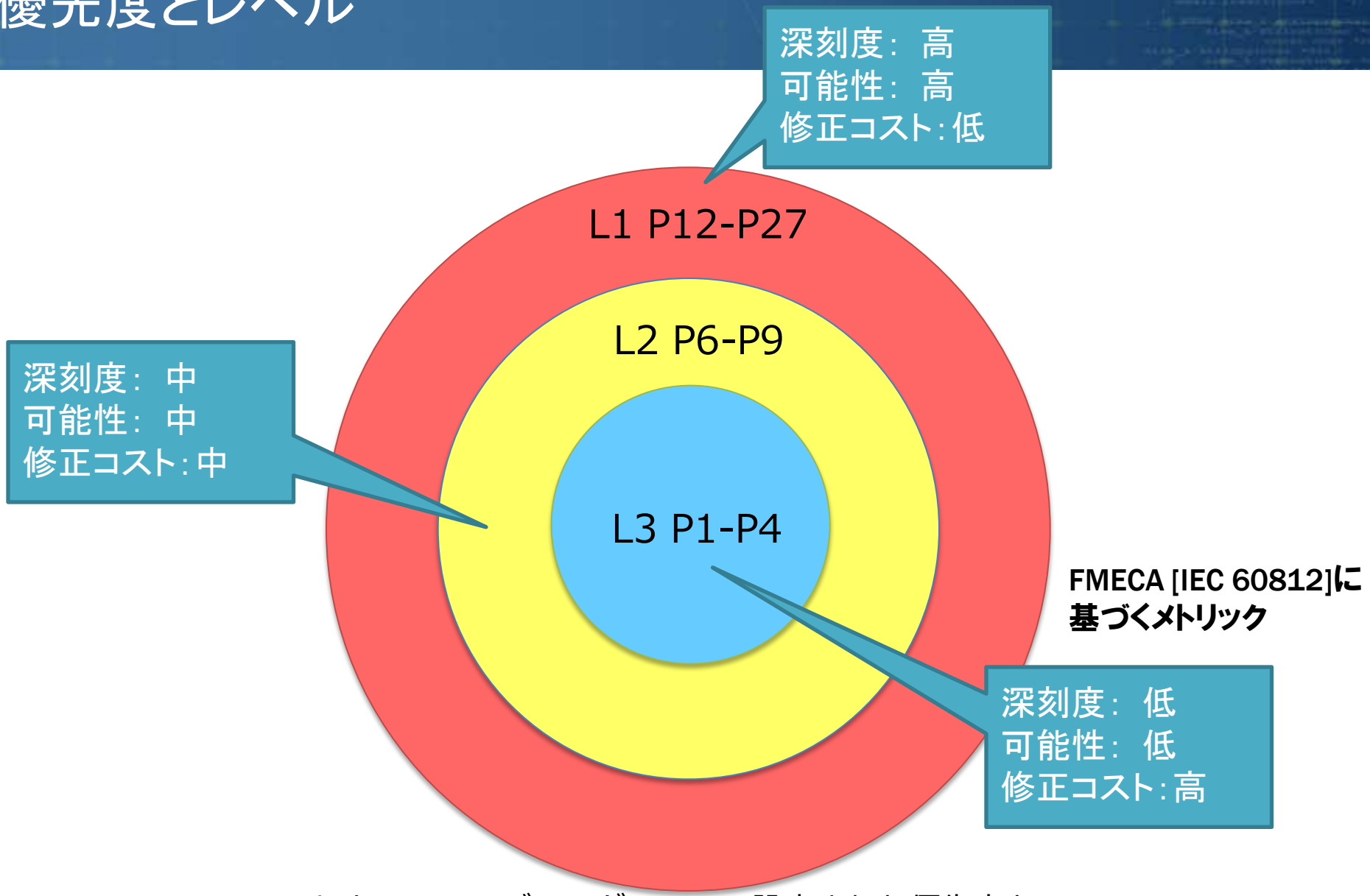
- セキュアコーディングの学習に



- 開発チーム内の認識共有

- 製品のセキュリティ関連仕様の明確化


優先度とレベル



CERT セキュアコーディングルールに設定された優先度とレベル

Secure Coding Standard for Java

00. **入力値検査とデータ無害化** (IDS)
01. **宣言と初期化** (DCL)
02. **式** (EXP)
03. **数値型と操作** (NUM)
04. **オブジェクト** (OBJ)
05. **メソッド** (MET)
06. **例外時の動作** (ERR)
07. **可視性と原子性** (VNA)
08. **ロック** (LCK)

内容に応じて分類

09. **スレッド**API (THI)
10. **スレッドプール** (TPS)
11. **スレッド安全性** (TSM)
12. **入出力** (FIO)
13. **シリアライゼーション** (SER)
14. **プラットフォームセキュリティ** (SEC)
15. **実行環境** (ENV)
49. **雑則** (MSC)

ルールの例

EXP00-J. メソッドの返り値を無視しない
FIO03-J. 処理を終了する前に一時ファイルを削除する
IDS07-J. 信頼できないソースからの入力を無害化せずに
Runtime.exec()メソッドに渡さない
MSC00-J. セキュアなデータ交換にはSocketsではなく
SSLSocketsを使う
MSC03-J. センシティブな情報をハードコードしない

……

例: FIO03-J. 処理を終了する前に一時ファイルを削除する(1)

●一時ファイルを共有ディレクトリに置いて操作すると、ファイル作成の妨害、symlink攻撃などの危険がある。共有ディレクトリの危険性についてはFIO00-Jも参照。
●一時ファイルの置き場所は「セキュアなディレクトリ」にするのがおすすめ。
●一時ファイルの削除方法に注意すること。

以下のコード例では、ファイルの作成場所はセキュアなディレクトリ、パーミションも適切に設定しているという前提↓

例: FIO03-J. 処理を終了する前に一時ファイルを削除する (2)

違反コード例

```
class TempFile {
  public static void main(String[] args) throws IOException{
    File f = new File("tempnam.tmp");
    if (f.exists()) {
      System.out.println("This file already exists");
      return;
    }

    FileOutputStream fop = null;
    try {
      fop = new FileOutputStream(f);
      String str = "Data";
      fop.write(str.getBytes());
    } finally {
      if (fop != null) {
        try {
          fop.close();
        } catch (IOException x) {
          // handle error
        }
      }
    }
  }
}
```

一時ファイル名をハードコードしている

異常終了時には、
Finally節が必ず実行される保証はない.

例: FIO03-J. 処理を終了する前に一時ファイルを削除する (3)

違反コード例 (createTempFile(), deleteOnExit())

```
class TempFile {
  public static void main(String[] args) throws IOException{
    File f = File.createTempFile("tempnam",".tmp");
    FileOutputStream fop = null;
    try {
      fop = new FileOutputStream(f);
      String str = "Data";
      fop.write(str.getBytes());
      fop.flush();
    } finally {
      // Stream/file はまだオープンされたまま
      f.deleteOnExit(); // Delete the file when the JVM terminates

      if (fop != null) {
        try {
          fop.close();
        } catch (IOException x) {
          // エラー処理
        }
      }
    }
  }
}
```

一時ファイルを生成させているのでベター

deleteOnExit()も、JVM異常終了時に必ず実行される保証はない.

例: FIO03-J. 処理を終了する前に一時ファイルを削除する (4)

適合コード (Java SE 7: DELETE_ON_CLOSE)

JavaSE7で導入された NIO2 パッケージの Files クラスおよび try-with-resourcesを使用したコード

```
class TempFile {
  public static void main(String[] args) {
    Path tempFile = null;
    try {
      tempFile = Files.createTempFile("tempnam", ".tmp");
      try(BufferedWriter writer =
          Files.newBufferedWriter(tempFile, Charset.forName("UTF8"),
                                  StandardOpenOption.DELETE_ON_CLOSE)) {
        // ファイルに書き込み
      }
      System.out.println("Temporary file write done, file erased");
    } catch (FileAlreadyExistsException x) {
      System.err.println("File exists: " + tempFile);
    } catch (IOException x) {
      // パーミション違反など、その他のエラー
      System.err.println("Error creating temporary file: " + x);
    }
  }
}
```

例: FIO03-J. 処理を終了する前に一時ファイルを削除する (5)

リスク評価

プログラムの終了前に一時ファイルを削除しないと、情報漏えいやリソースの枯渇につながる恐れがある。

ルール	深刻度	可能性	修正コスト	優先度	レベル
FIO03-J	中	中	中	P8	L2

関連ガイドライン

- FIO43-C. Do not create temporary files in shared directories
- FIO43-CPP. Do not create temporary files in shared directories
- CWE-377. “Insecure Temporary File”
- CWE-459. “Incomplete Cleanup”

参考情報

- [API 2006] Class File, methods createTempFile, delete, deleteOnExit
- [Darwin 2004] 11.5 Creating a Transient File
- [J2SE 2011]
- [SDN 2008] Bug IDs: 4171239, 4405521, 4635827, 4631820
- [Secunia 2008] Secunia Advisory 20132

リスク評価とともに、関連情報などへのリンクも紹介.

まとめ

- OSSでもセキュリティを高める取り組みが必要

Java でもセキュリティを高める取り組みが必要

- CERT セキュアコーディングスタンダードやJPCERT/CCの資料を活用し、セキュアコーディングの実践に役立ててください

Java 関連これから充実させていきますよ!

- みなさんと一緒に、ソフトウェアのセキュリティを高めていきましょう！

- セキュアコーディングのページ
  - Java セキュアコーディング 並行処理編
    https://www.jpcert.or.jp/securecoding_materials.html
  - C/C++セキュアコーディングセミナ講義資料
    https://www.jpcert.or.jp/research/materials.html
  - CERT C Secure Coding Standard日本語版
    https://www.jpcert.or.jp/sc-rules/

- 問い合わせ先

  email : secure-coding@jpcert.or.jp