KOF2014 version

KOF 2014 KANSAI OPEN FORUM 2014
関西オープンソース 2014 関西コミュニティ大決戦 2014

JPCERT CC®

～誰かの失敗を他山の石に～
脆弱性事例に学ぶセキュアコーディング
「SSL/TLS証明書検証」編

JPCERT/CC 情報流通対策グループ
戸田 洋三 (yozo.toda@jpcert.or.jp)

自己紹介

JPCERT/CC
情報流通対策グループ 解析チーム
リードアナリスト　戸田 洋三

http://www.tomo.gr.jp/root/e9706.html

脆弱性情報分析, セキュアコーディング普及啓発活動……に努めてます

JPCERT Coordination Center

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んでいる組織

JPCERT CC®
Japan Computer Emergency Response Team Coordination Center
JPCERTコーディネーションセンター

JPCERT/CCの主な活動

本日の話題

- ✓ SSL/TLS証明書検証って?
- ✓ 証明書検証に関連する脆弱性事例紹介
- ✓ 脆弱性事例のポイントおさらい
- ✓ 参考情報など

SSL/TLS証明書検証って？

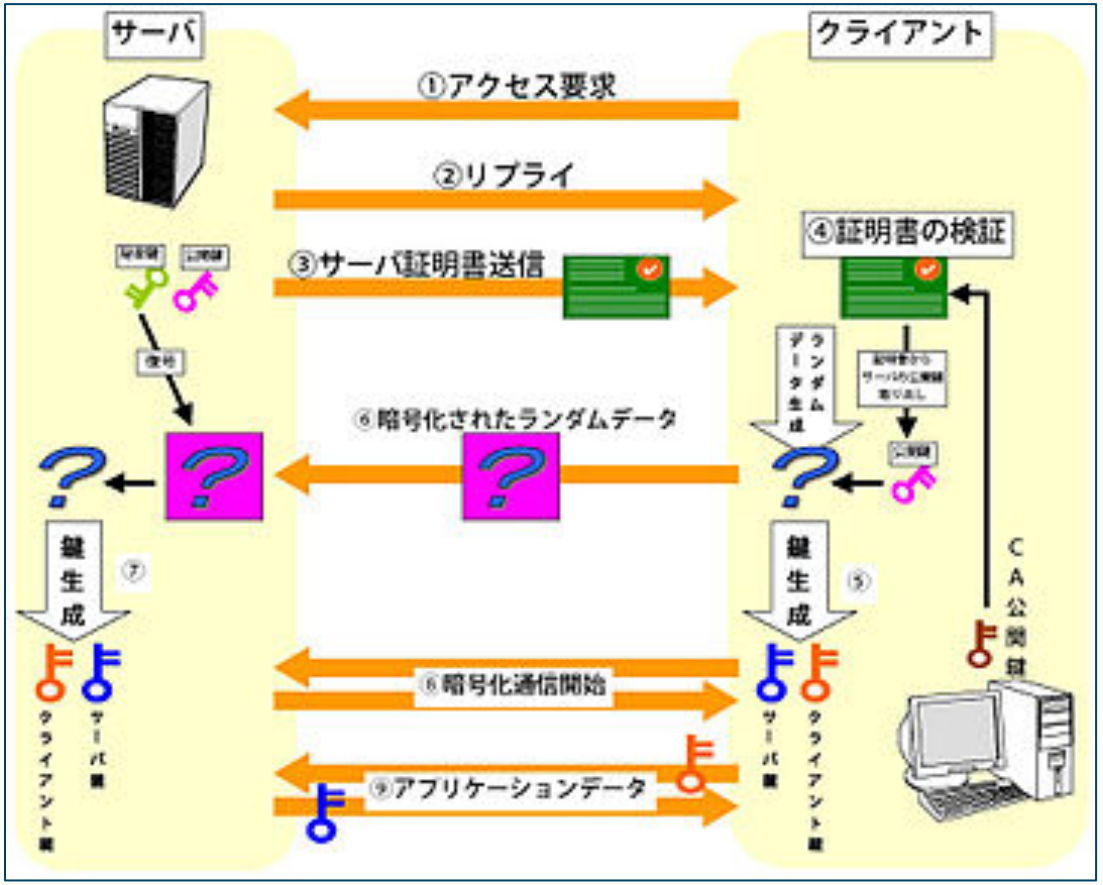
脆弱性事例

脆弱性事例ポイントおさらい

参考情報など

SSL/TLS通信手順

SSL/TLS
Transport層の上で暗号化通信路を実現する仕組み



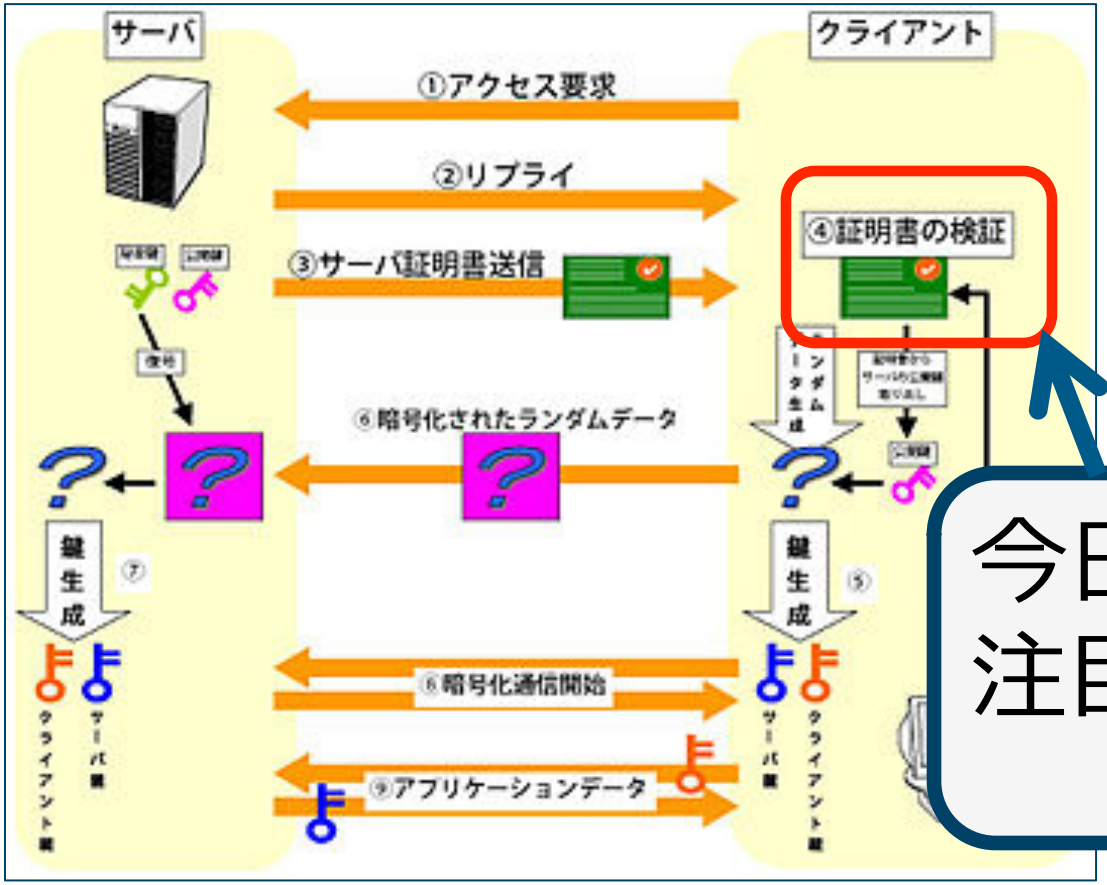
SSL 3.0 - RFC6101
TLS 1.0 - RFC2246
TLS 1.1 - RFC4346
TLS 1.2 - RFC5246
……

攻撃手法への対策や新たな暗号化スイート追加など，現在も進化中…

参考: http://ja.wikipedia.org/wiki/Transport_Layer_Security

SSL/TLS通信手順

SSL/TLS
Transport層の上で暗号化通信路を実現する仕組み



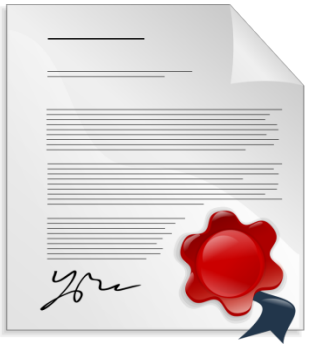
SSL 3.0 - RFC6101
TLS 1.0 - RFC2246
TLS 1.1 - RFC4346
TLS 1.2 - RFC5246
……

今日はここだけに注目します.

参考: http://ja.wikipedia.org/wiki/Transport_Layer_Security

サーバ証明書って?

- **サーバの公開鍵**と**サーバのドメイン名**がはいっている (https で使う場合)
- とある CA (認証局)が, ドメイン名と公開鍵の関係を確認してくれたもの

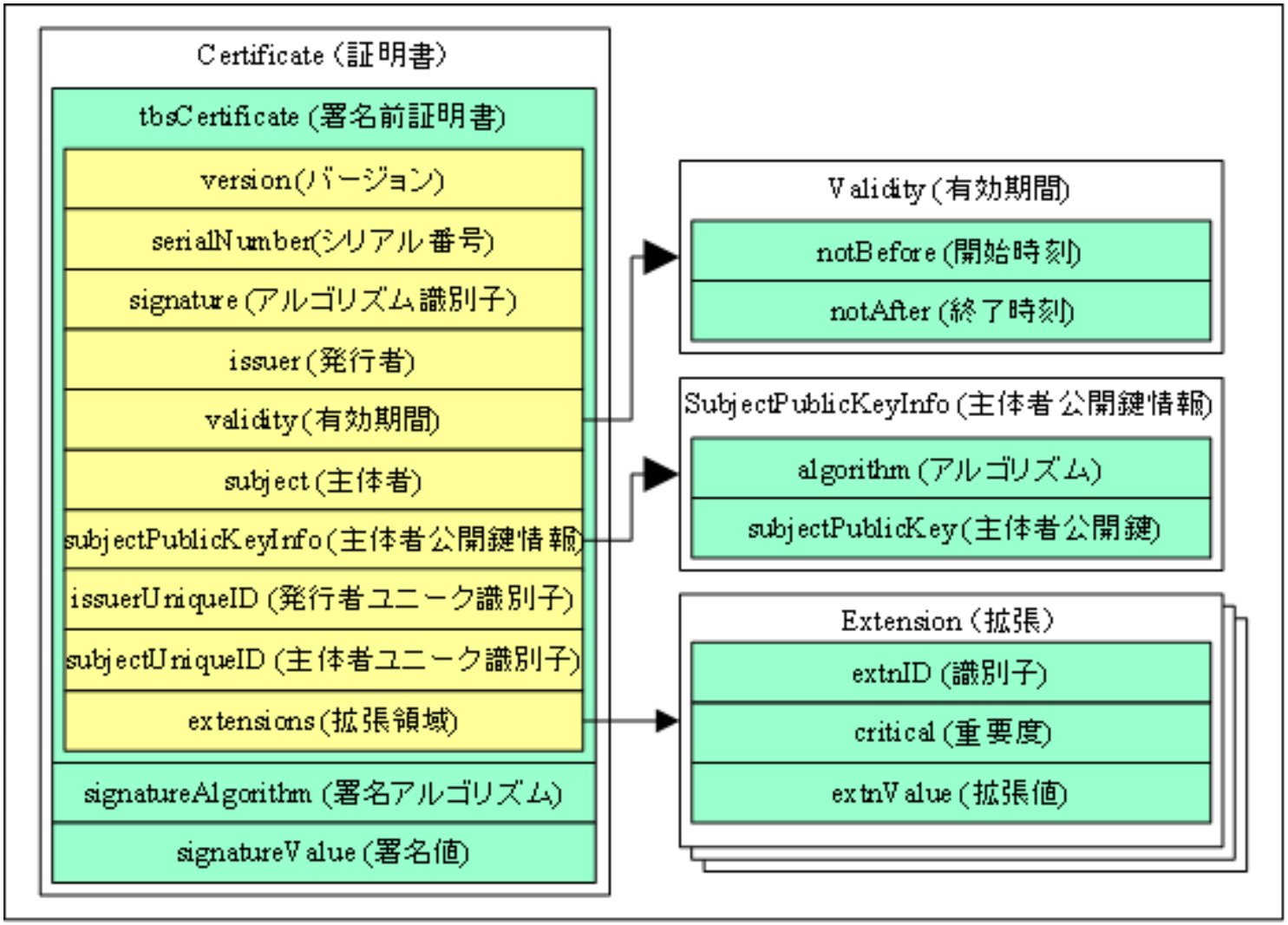


- ITU-T規格 X.509 で決められているものを流用
  - RFC5280, RFC6818
- Web ブラウザなどでは, 信頼する CA のオレオレ証明書をあらかじめ持っておく

証明書の検証には2つの処理が含まれている

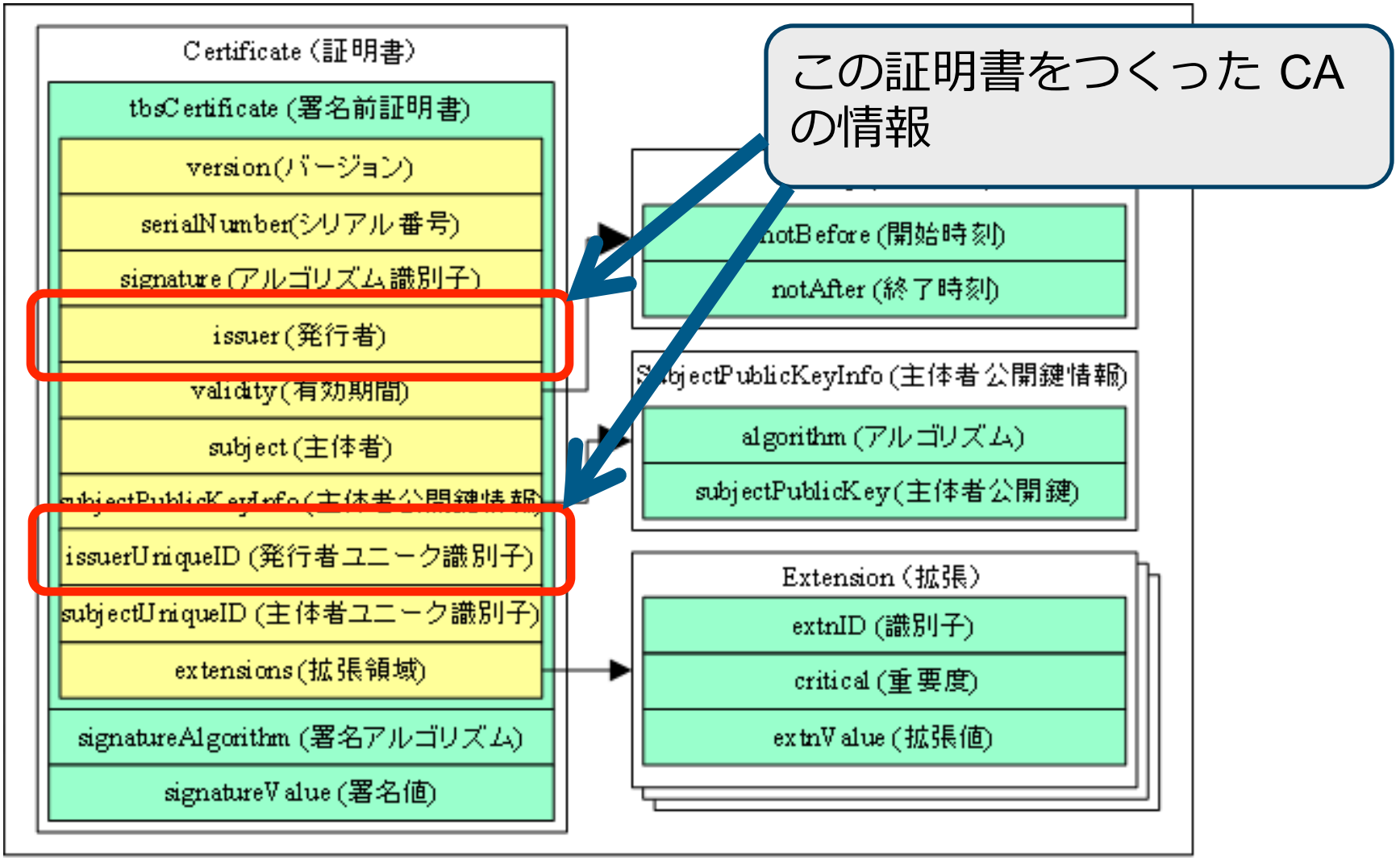
- 受け取ったサーバ証明書が, 信頼する CA を起点として正しく作られたものであることを確認する
  - ⇒**証明書検証**
- 信頼できるサーバ証明書が持っているサーバ名がアクセスしようとしているサーバ名と一致することを確認する
  - ⇒**ホスト名検証**

X.509 v3 証明書の構造



https://www.ipa.go.jp/security/pki/033.html

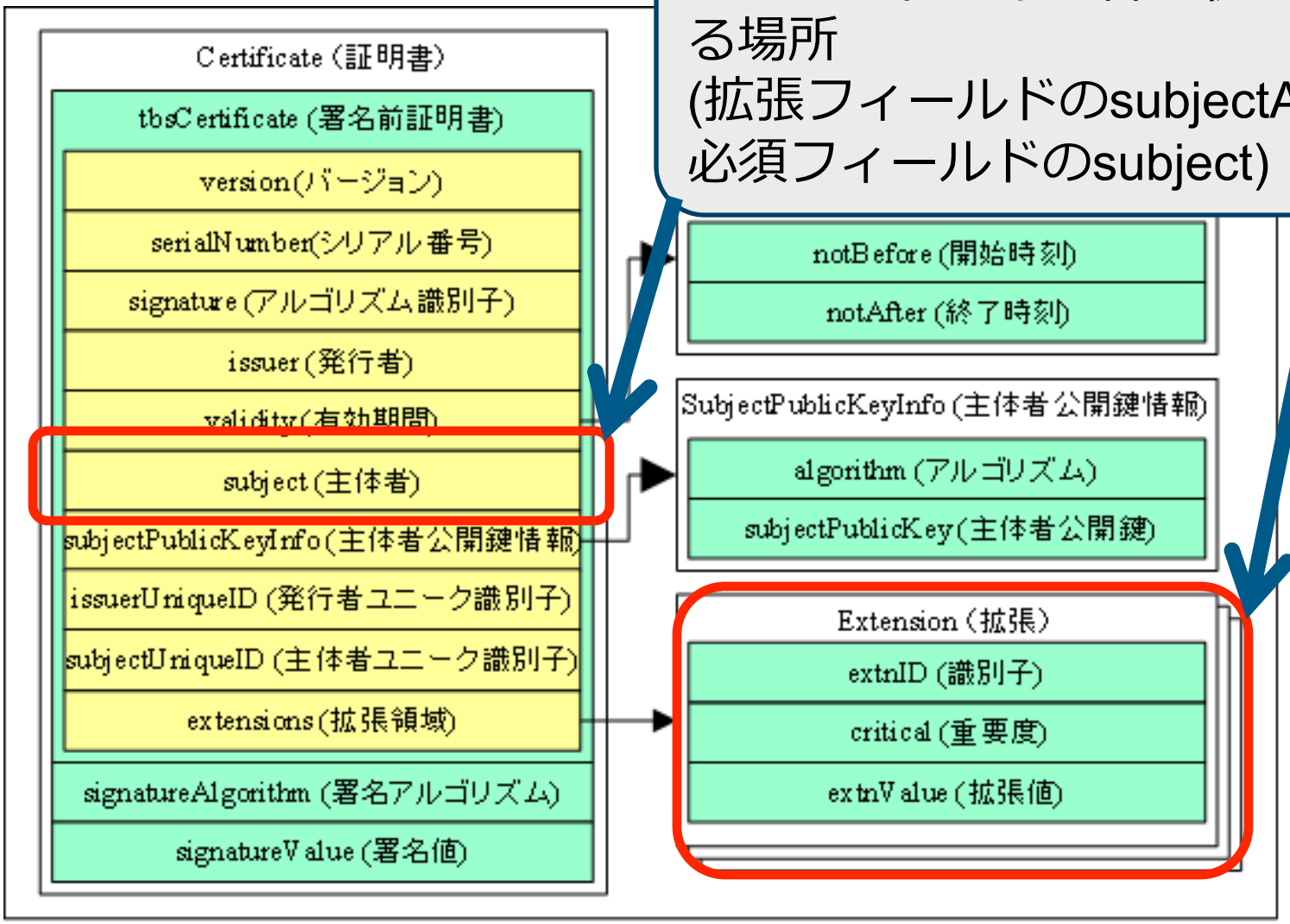
X.509 v3 証明書の構造



https://www.ipa.go.jp/security/pki/033.html

X.509 v3 証明書の構造

サーバのドメイン名が収められている場所
(拡張フィールドのsubjectAltNameと必須フィールドのsubject)



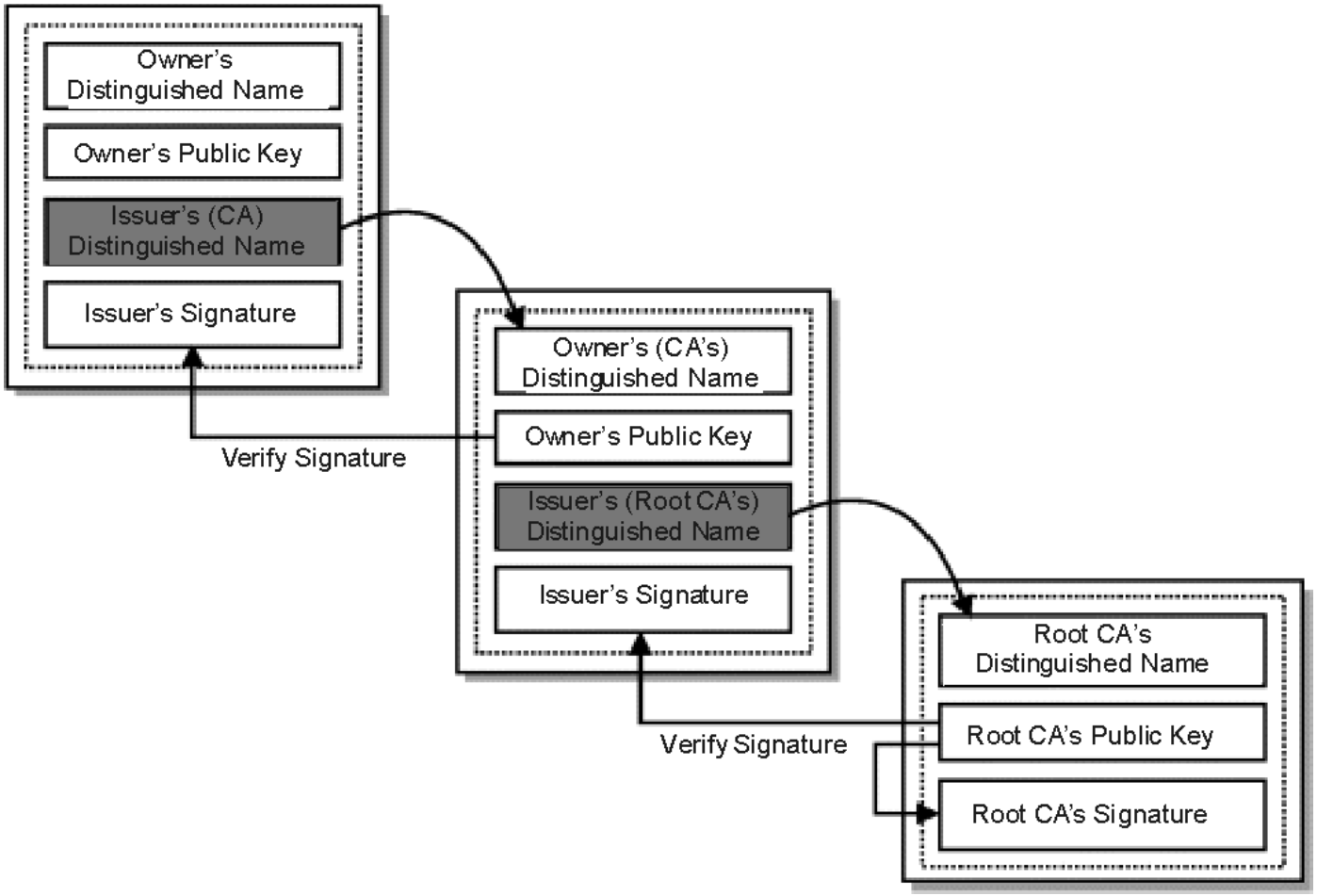
https://www.ipa.go.jp/security/pki/033.html

X.509 証明書検証

- ASN.1 データ構造として正しく読めるか?
- CAによる署名は正しくなされているか?
- 有効期限は切れていないか?
- 失効されていないか?
- **証明書チェーンをたどって信頼している(CAの)証明書に辿りつけるか?**

RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile 6. Certification Path Validation http://tools.ietf.org/html/rfc5280#section-6
Certification path validation algorithm http://en.wikipedia.org/wiki/Certification_path_validation_algorithm

証明書チェーンの検証処理



https://security.stackexchange.com/questions/56389/ssl-certificate-framework-101-how-does-the-browser-actually-verify-the-validity

ホスト名検証

●アクセスしようとしているサーバ名(ドメイン名)とサーバ証明書に示されているサーバ名が一致するか?
●サーバ証明書に subjectAltName が含まれている場合はそれを使うこと
●一致するかどうかの検証アルゴリズムは証明書検証で使うアルゴリズムと同じ

RFC2818: HTTP Over TLS 3.1. Server Identity http://tools.ietf.org/html/rfc2818#section-3.1
RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile 7. Processing Rules for Internationalized Names http://tools.ietf.org/html/rfc5280#section-7

SSL/TLS証明書検証って？

脆弱性事例

脆弱性事例ポイントおさらい

参考情報など

証明書の検証不備

jvn.jp の掲載事例で「証明書の検証不備」を拾い出してみると…(2013年以降)

JVN#27388160: Android 版 「スマ保」における SSL/TLS サーバ証明書の検証不備の脆弱性
JVN#48270605: Yahoo!ボックス(Android版) における SSL サーバ証明書の検証不備の脆弱性
JVN#04560253: Android 版アプリ「ゆこゆこ」における SSL サーバ証明書の検証不備の脆弱性
JVN#17637243: Android 版アプリ Kindle における SSL サーバ証明書の検証不備の脆弱性
JVN#27702217: Android 版 Ameba における SSL サーバ証明書の検証不備の脆弱性
JVN#72950786: Android 版 Outlook.com における SSL サーバ証明書の検証不備の脆弱性
JVN#10603428: Android 版アプリ「JR東日本アプリ」における SSL サーバ証明書の検証不備の脆弱性
JVN#16263849: Android 版アプリ「出前館」における SSL サーバ証明書の検証不備の脆弱性
JVN#48810179: Android 版アプリ「デニーズ」における SSL サーバ証明書の検証不備の脆弱性
JVN#97810280: KDrive個人版 PCクライアントソフトにおける SSL サーバ証明書の検証不備の脆弱性
JVN#75084836: Android 版 Yahoo!ショッピングにおける SSL サーバ証明書の検証不備の脆弱性
JVN#68156832: ヤフオク! における SSL サーバ証明書の検証不備の脆弱性
JVN#39218538: Android 版 ピザハット公式アプリ　宅配ピザのPizzaHut における SSL サーバ証明書の検証不備の脆弱性
JVN#85812843: FileMaker Pro における SSL サーバ証明書の検証不備の脆弱性
JVNVU#389795: Windows Phone 7 に SSL サーバ証明書の検証不備の脆弱性
JVN#39707339: Opera における SSL サーバ証明書の検証不備の脆弱性
JVN#82029095: spモードメールアプリにおける SSL サーバ証明書の検証不備の脆弱性

Android アプリでよく報告されてます.
(もちろん他の実行環境の例もあり)

参考
SSL/TLS ではなく IPsec での例もあり: CVE-2013-1028
Apple OS X v10.8.5 and Security Update 2013-004

某宅配ピザアプリにおけるSSL証明書検証不備の脆弱性

公開日：2013/06/07　最終更新日：2013/06/07

JVN#39218538
Android 版 ピザハット公式アプリ　宅配ピザのPizzaHut における SSL サーバ証明書の検証不備の脆弱性

概要
Android 版 ピザハット公式アプリ　宅配ピザのPizzaHut には、SSL サーバ証明書の検証不備の脆弱性が存在します。

修正済み

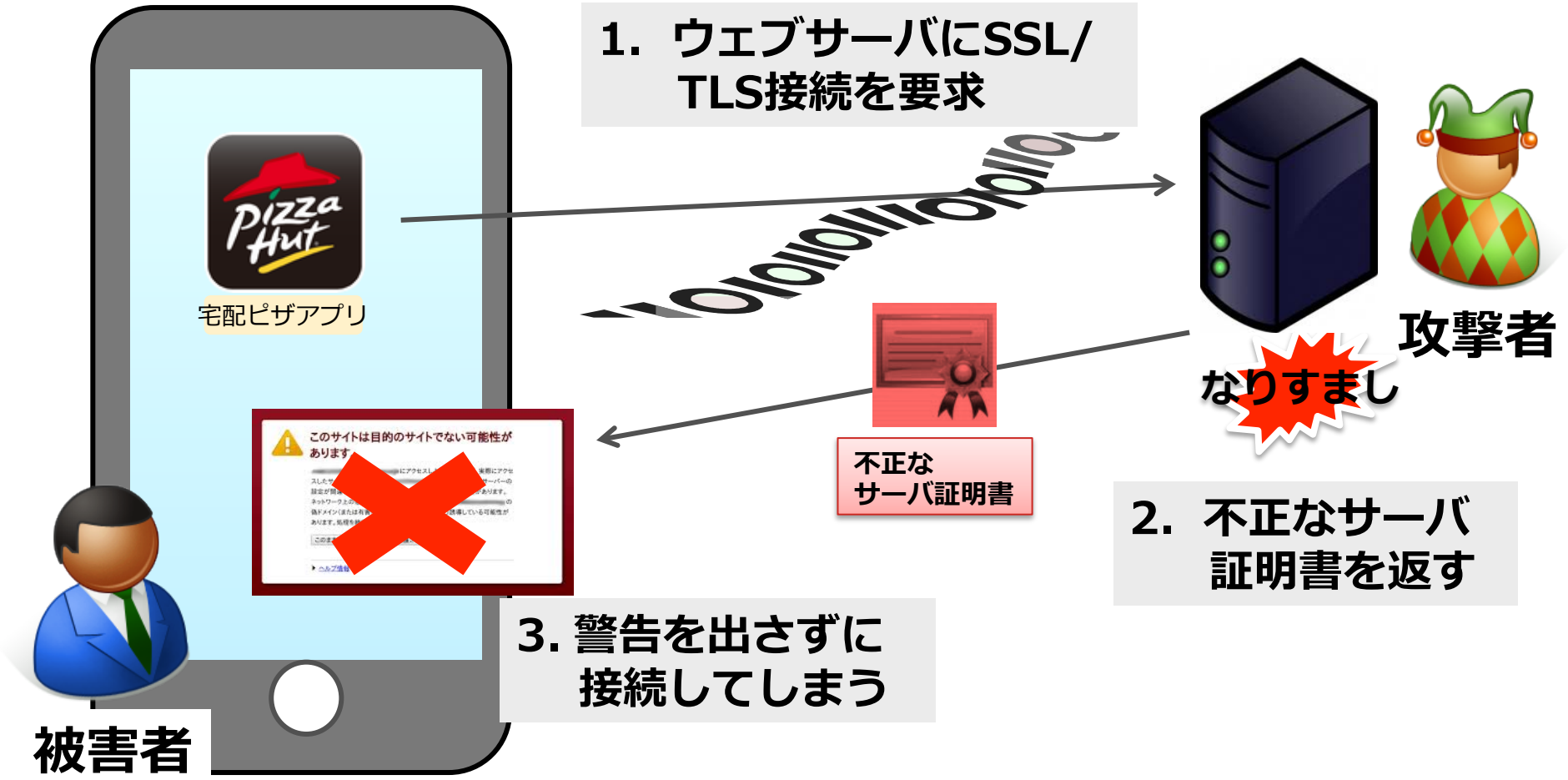
盗聴や改ざ
プリユーザ
ない

ピザのデリバリーを予約したり注文したりできるアプリ。

通信にはユーザの氏名、住所、メールアドレスなどの情報が含まれる

https://jvn.jp/jp/[redacted]538/index.html
https://play.google.[redacted]/store/apps/details?id=jp.pizzahut.aorder

攻撃のシナリオ



脆弱性の原因

jp/pizzahut/aorder/data/DataUtil.java

```
public static HttpClient getNewHttpClient() {
   DefaultHttpClient v6;
   try {
      KeyStore v5 = KeyStore.getInstance(KeyStore.getDefaultType());
      v5.load(null, null);
      MySSLSocketFactory mySSLScoket = new MySSLSocketFactory(v5);
      if(PizzaHutDefineRelease.sAllowAllSSL) {
         ((SSLSocketFactory)mySSLScoket).setHostnameVerifier
                                        (SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);
      }

      BasicHttpParams v2 = new BasicHttpParams();
      HttpConnectionParams.setConnectionTimeout(((HttpParams)v2), 30000);
      ...
   }
   catch(Exception v1) {
      v6 = new DefaultHttpClient();
   }
   return ((HttpClient)v6);
}
```

脆弱性の原因

```
public static HttpClient getNewHttpClie
   DefaultHttpClient v6;
   try {
      KeyStore v5 = KeyStore.getInstanc
      v5.load(null, null);
      MySSLSocketFactory mySSLScoket =
      if(PizzaHutDefineRelease.sAllowAl
         ((SSLSocketFactory)mySSLScoket).setHostnameVerifier
                                        (SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);
      }

      BasicHttpParams v2 = new BasicHttpParams();
      HttpConnectionParams.setConnectionTimeout(((H        ms)v2), 30000);
      ...
   }
   catch(Exception v1) {
      v6 = new DefaultHttpClient();
   }
   return ((HttpClient)v6);
}
```

public class

Summary: Ctors | Methods | Inherited Methods | [Expand All]

Added in API level 1

AllowAllHostnameVerifier

extends AbstractVerifier

java.lang.Object
↳org.apache.http.conn.ssl.AbstractVerifier
↳org.apache.http.conn.ssl.AllowAllHostnameVerifier

Class Overview

The ALLOW_ALL HostnameVerifier essentially turns hostname verification off. This implementation is a no-op, and never throws the SSLException.

ホスト名検証を無効にしている!

他の脆弱なパターン

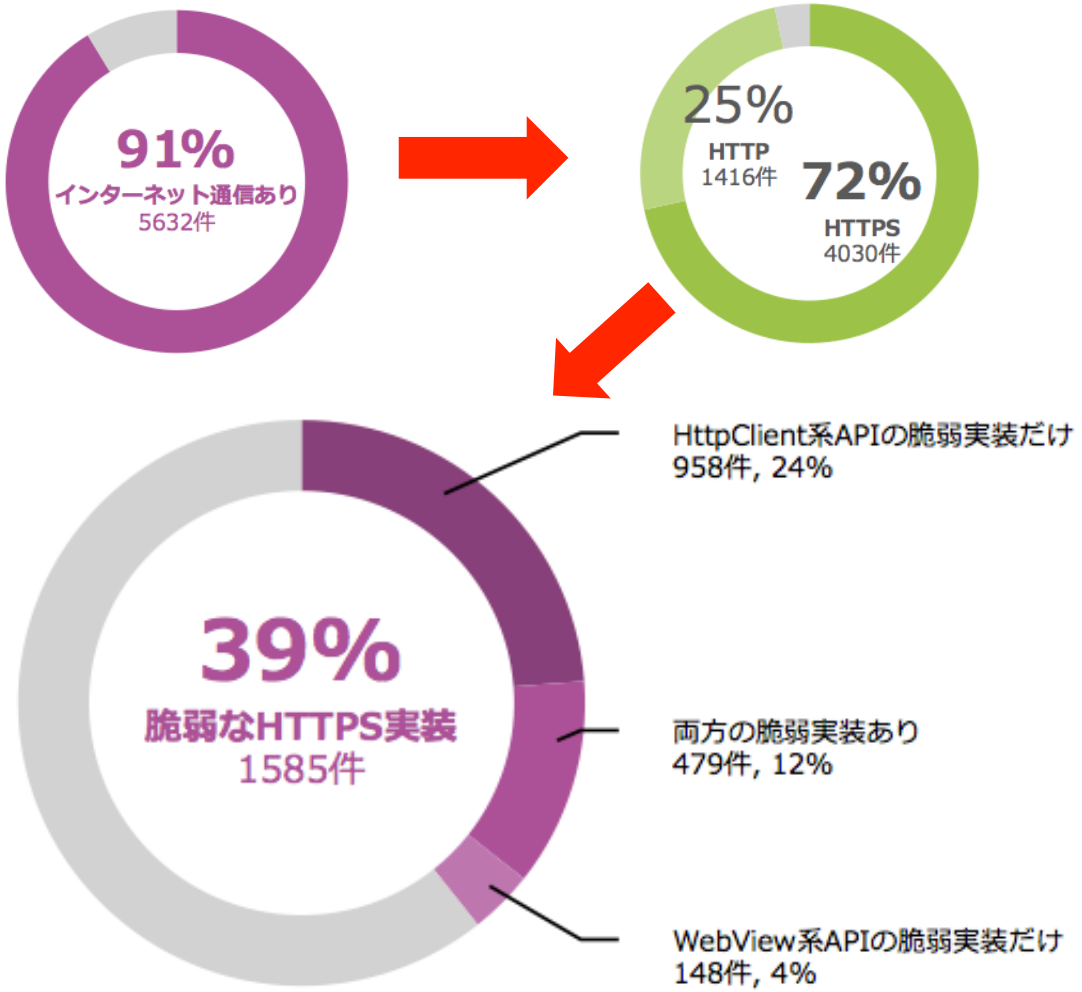
空っぽの HostnameVerifier

```
HostnameVerifier hv = new HostnameVerifier() {
   @Override
   public boolean verify(String hostname, SSLSession session) {
      // 常に true を返す → どんなホスト名でも受付ける
   return true;
   }
};
```

空っぽの TrustManager

```
TrustManager tm = new X509TrustManager() {
   @Override
   public void checkClientTrusted(X509Certificate[] chain,
            String authType) throws CertificateException {
      // 何もしない → どんな証明書でも受付ける
   }
   @Override
   public void checkServerTrusted(X509Certificate[] chain,
            String authType) throws CertificateException {
      // 何もしない → どんな証明書でも受付ける
   }
   @Override
   public X509Certificate[] getAcceptedIssuers() {
      return null;
   }
};
```

Androidアプリの25%は脆弱なHTTPSを実装



アプリの4つに1つは脆弱なHTTPS実装を抱えている

Androidアプリ脆弱性調査レポート 2013年10月版
http://www.sonydna.com/sdna/solution/android_vulnerability_report_201310.pdf

脆弱なHTTPS実装の原因

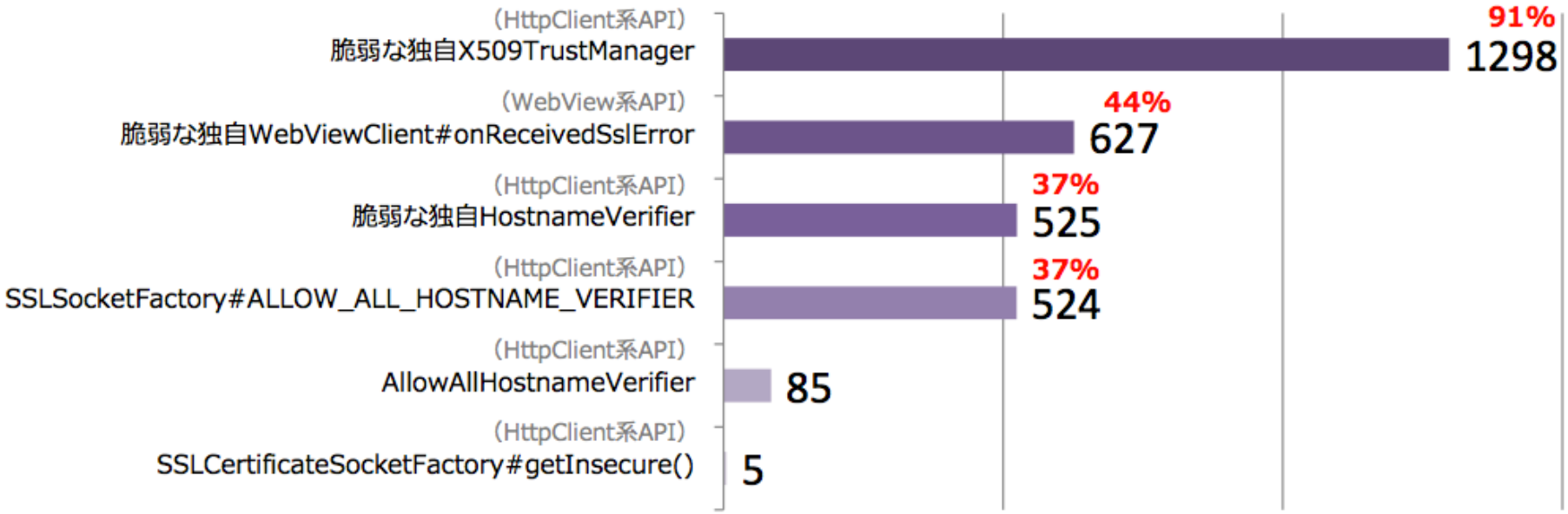


図 8 脆弱な HTTPS 実装の原因

Androidアプリ脆弱性調査レポート 2013年10月版
http://www.sonydna.com/sdna/solution/android_vulnerability_report_201310.pdf

ホスト名検証不備: CVE-2013-4073 Ruby

OpenSSL クライアントにおけるホスト名検証バイパス脆弱性 (CVE-2013-4073)

https://www.ruby-lang.org/ja/news/2013/06/27/hostname-check-bypassing-vulnerability-in-openssl-client-cve-2013-4073/

Ruby
A PROGRAMMER'S BEST FRIEND

“Rubyの SSL クライアントはホストネームが一意であることをチェックする機構を持っていますが、 その機構は **null バイト文字が含まれている**証明書のホストネームを適切に処理することができません。”

ホスト名検証不備: CVE-2014-3577 Apache HttpComponents

CVE-2014-3577 Apche HttpComponents client: Hostname verification susceptible to MITM attack
http://seclists.org/fulldisclosure/2014/Aug/48

類似事例: CVE-2012-6153, CVE-2012-5783

“Apache HttpComponents … may be susceptible to a 'Man in the Middle Attack' due to **a flaw in the default hostname verification** during SSL/TLS when a **specially crafted** server side certificate is used.”

“a (crafted) DN with a O field such as
O="foo,CN=www.apache.orgâ
and ..... ordered such that the O appears prior to the CN field would incorrectly match on the <www.apache.org> ..."

証明書チェーン検証不備: Fake ID

Android Fake ID Vulnerability Lets Malware Impersonate Trusted Applications, Puts All Android Users Since January 2010 At Risk

https://bluebox.com/technical/android-fake-id-vulnerability/

Presented at BlackHat 2014 USA

ANDROID FAKEID VULNERABILITY WALKTHROUGH

https://www.blackhat.com/us-14/archives.html#android-fakeid-vulnerability-walkthrough

Android OS のアプリ署名に関する脆弱性

証明書チェーン検証不備: Fake ID

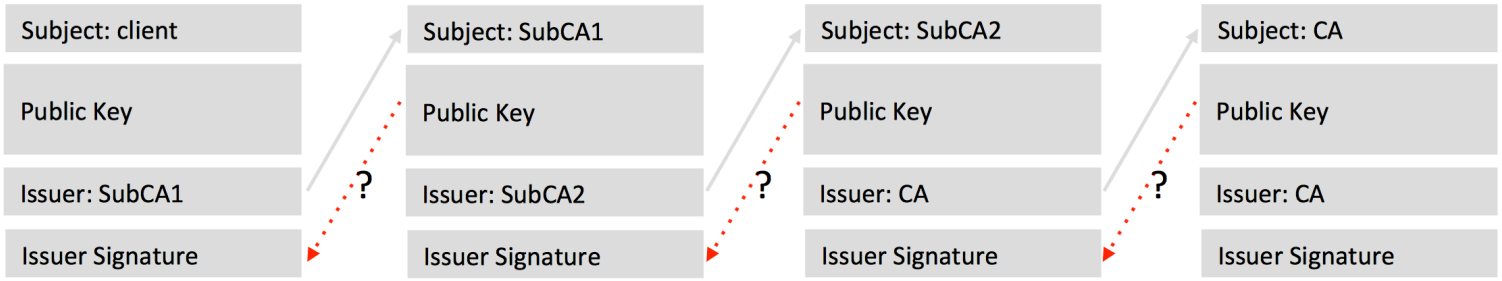
- Android アプリには電子署名がつけられている
- Android OS はアプリインストール時に電子署名を確認
  - SSL/TLSにおけるサーバ証明書検証と同等の処理

- 電子署名を確認するコードは Apache Harmony 由来のもの

- 証明書チェーンの確認に問題あり

Subject/Issuer の文字列としての一致のみを確認し, 署名の確認を行っていなかった!!

“there is a conspicuous **absence of cryptographic verification** of any issuer cert claims, instead defaulting to **simple subjectDN to issuerDN string matching**.”



A certificate can **claim** to be issued by any other certificate …

… and that claim is not verified

PKI Chaining - Android

BlackHat2014での発表資料から

証明書チェーン検証不備: Apple iOS

TWSL2011-007: iOS SSL Implementation Does Not Validate Certificate Chain
http://blog.spiderlabs.com/2011/07/twsl2011-007-ios-ssl-implementation-does-not-validate-certificate-chain.html
https://www3.trustwave.com/spiderlabs/advisories/TWSL2011-007.txt

“iOS's SSL certificate parsing contains a flaw where it fails to check the **basicConstraints** parameter of certificates in the chain.”

証明書チェーンのなかに「CA ではない」という属性の証明書があっても, エラーにしていなかった

ACM CCS 2012

- Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security
  http://www2.dcsec.uni-hannover.de/files/android/p50-fahl.pdf
- The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software
  https://crypto.stanford.edu/~dabo/pubs/abstracts/ssl-client-bugs.html

多くのアプリが SSL/TLS ライブラリを誤用している!!

- 証明書検証を無効にしてしまう
- ホスト名検証を無効にしてしまう
-……

ACM CCS 2013

- Rethinking SSL Development in an Appified World
  http://android-ssl.org/files/p49.pdf

SSL/TLS関連の脆弱性の発生原因を探る
- 開発者の理解不足
- 開発中の一時的な設定のままリリース
- ユーザからの要望

SSL/TLS証明書検証って？

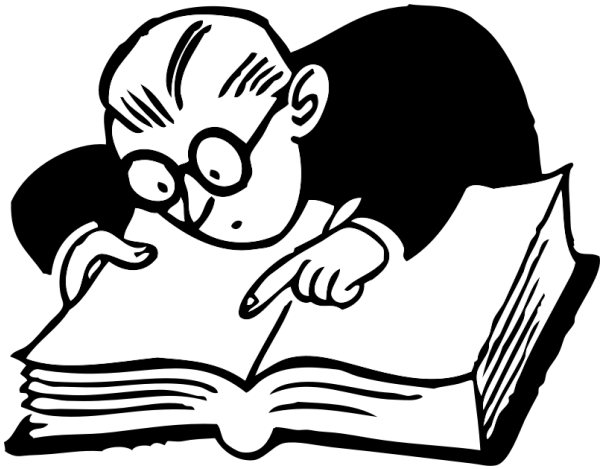
脆弱性事例

脆弱性事例ポイントおさらい

参考情報など

ポイント1: 証明書を検証する

- SSL/TLS 通信を行うならちゃんと証明書を検証しましょう
- 動作テストで検証を無効にしたのをそのまま公開、なんてダメよ
- Java/Androidの場合
  - —SSLException は無視しない
  - —TrustManager を独自実装しない
  - —HostnameVerifier を独自実装しない



ポイント2: 証明書チェーンとホスト名検証に注意

- ライブラリなどで提供されている場合には下手にカスタマイズせずにそのまま使おう
- 自前で実装しなければならない場合は…
  - 仕様をしっかり読み込む
  - 動作テストもしっかりとね
  - 既存の脆弱性事例を参考にして細工された証明書でも正しく動作するように注意する

参考: Proxy で動作検証

- Proxy 経由で通信させることで, 証明書の検証処理を確認できる
- 有名どころとしては Burp proxy, dsniff, Fiddler など
- これらのプロキシでは https 通信に対して自己署名証明書を提示したり, あらかじめ設定しておいた証明書を提示したりといった動作が可能
    - 信頼できない証明書(オレオレ証明書など)
    - ホスト名検証
    - 有効期限切れ
    - 失効済み証明書

SSL/TLS証明書検証って？

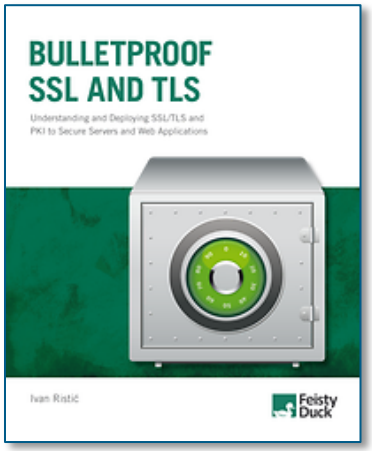
脆弱性事例

脆弱性事例ポイントおさらい

参考情報など

参考書籍

- マスタリングTCP/IP SSL/TLS編
  - http://shop.ohmsha.co.jp/shop/shopdetail.html?brandcode=000000001666&search=4-274-06542-1



- Bulletproof SSL and TLS
  - https://www.feistyduck.com/books/bulletproof-ssl-and-tls/

参考情報

- IPA: PKI 関連技術情報
  - https://www.ipa.go.jp/security/pki/index.html
- Introduction to Public-Key Cryptography
  - https://developer.mozilla.org/en-US/docs/Introduction_to_Public-Key_Cryptography
- Exciting Updates to Certificate Verification in Gecko
  - https://blog.mozilla.org/security/2014/04/24/exciting-updates-to-certificate-verification-in-gecko/
- JNSA PKI Day
  - PKI Day 2014 (http://www.jnsa.org/seminar/pki-day/2014/)
- 一般社団法人日本スマートフォンセキュリティ協会『Androidアプリのセキュア設計・セキュアコーディングガイド』
  - https://www.jssec.org/report/securecoding.html
- SSL/TLSライブラリの正しい使い方(もしくは、コモンネームの検証について)
  - http://blog.kazuhooku.com/2014/01/ssltls.html
- Android Pinning by Moxie Marlinspike
  - https://github.com/moxie0/AndroidPinning

References

- SSL Vulnerabilities: Who listens when Android applications talk?
  - http://www.fireeye.com/blog/technical/2014/08/ssl-vulnerabilities-who-listens-when-android-applications-talk.html
- Defeating SSL Certificate Validation for Android Applications
  - https://secure.mcafee.com/us/resources/white-papers/wp-defeating-ssl-cert-validation.pdf
- OnionKit by Android Library Project for Multi-Layer Network Connections (Better TLS/SSL and Tor)
  - https://github.com/guardianproject/OnionKit
- CERT/CC Vulnerability Note VU#582497: Multiple Android applications fail to properly validate SSL certificates
  - https://www.kb.cert.org/vuls/id/582497

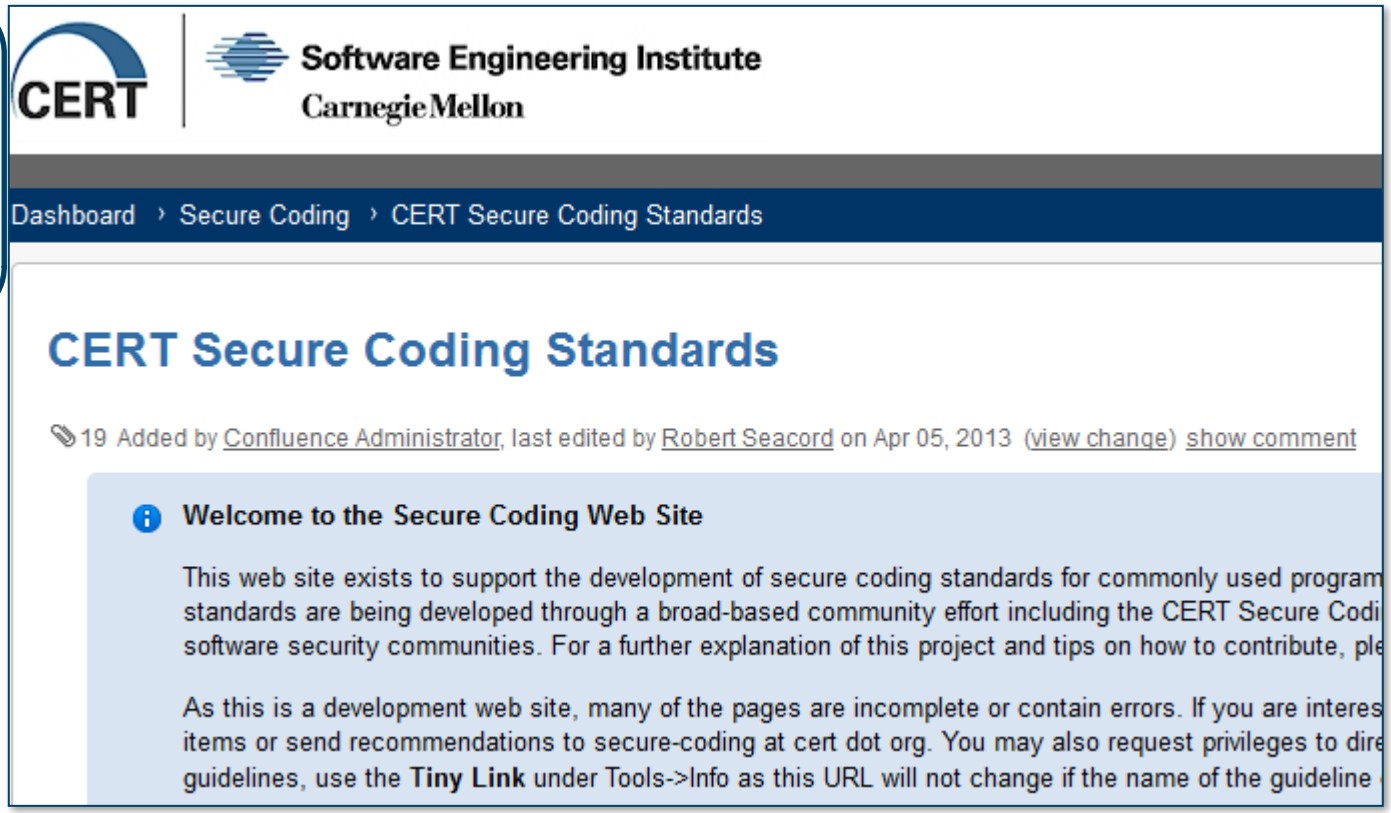
付録
セキュアコーディングスタンダード

セキュアコーディングスタンダード

米国 CMU/SEI の the CERT Secure Coding Initiative によるコーディングスタンダードシリーズ

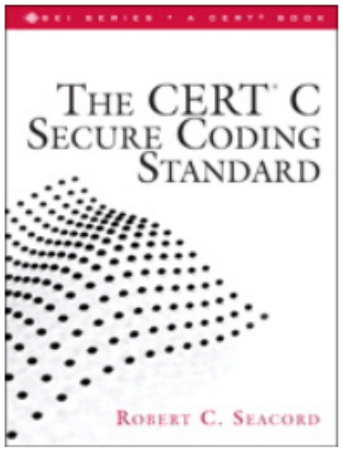
https://www.securecoding.cert.org/

いまのところ，4種類公開されています．



C セキュアコーディングスタンダード

The CERT C Secure Coding Standard



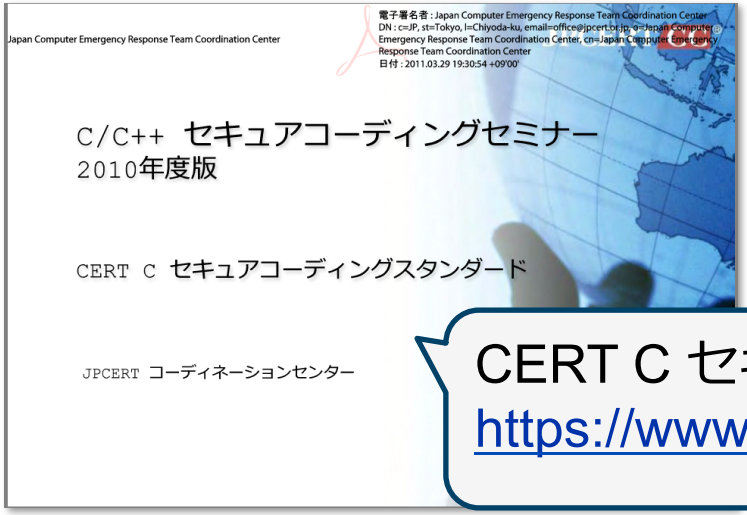
Version 1.0 of The CERT C Secure Coding Standard is now available as a book from Addison-Wesley. This official release can be used as a fixed point of reference for the development of compliant applications and source code analysis tools.

Development of the next version of the CERT C Secure Coding Standard is being performed here on the secure coding wiki. This version is a work in progress and reflects the current thinking of the secure coding community.

interes

There is a
JPCERT/C

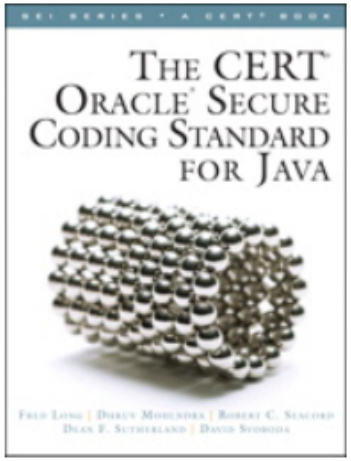
JPCERT/CC で日本語公開中!!
https://www.jpcert.or.jp/sc-rules/



CERT C セキュアコーディングスタンダード紹介
https://www.jpcert.or.jp/research/materials.html#secure

Java セキュアコーディングスタンダード

The CERT Oracle Secure Coding Standard for Java



Version 1.0 of *The CERT Oracle Secure Coding Standard for Java* is now available as a book from Addison-Wesley.

Development of the next version of the The CERT Oracle Secure Coding Standard for Java is being performed here on the secure coding wiki. This version is a work in progress and reflects the current thinking of the secure coding community. Subsequent official releases of this standard will be issued as dictated by th

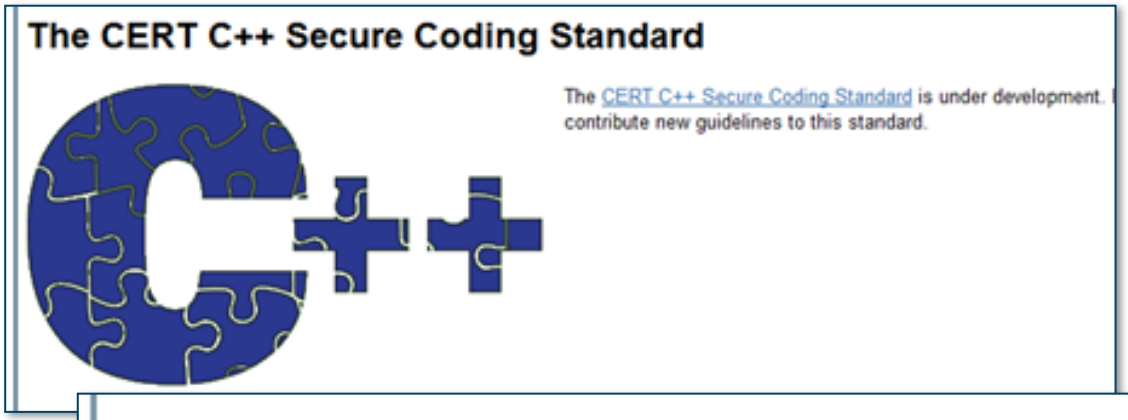
Java is a t
countries.

There is als
partner JPC

JPCERT/CC で日本語公開中!!
https://www.jpcert.or.jp/java-rules/

OSC2011@Nagoya でも紹介してます
http://www.ospn.jp/osc2011-nagoya/pdf/osc2011nagoya-JPCERT_CC.pdf

セキュアコーディングスタンダード(C++, Perl)



“under development”
(開発中)

一般社団法人JPCERTコーディネーションセンター
(https://www.jpcert.or.jp/)

セキュアコーディング
(https://www.jpcert.or.jp/securecoding/)

お問い合わせはこちらにどうぞ…
(secure-coding@jpcert.or.jp)