「Javaアプリケーション脆弱性事例調査資料」について

- この資料は、Javaプログラマである皆様に、脆弱性を身近な問題として感じてもらい、セキュアコーディングの重要性を認識していただくことを目指して作成しています。

- 「Javaセキュアコーディングスタンダード CERT/Oracle版」と合わせて、セキュアコーディングに関する理解を深めるためにご利用ください。

JPCERTコーディネーションセンター
セキュアコーディングプロジェクト
secure-coding@jpcert.or.jp

Apache CommonsのHttpClientにおけるSSLサーバ証明書検証不備

CVE-2012-5783

JVNDB-2012-005217

Apache Commons とは

- 主に Apache 関連製品で用いられる、再利用可能な Java コンポーネントをまとめているプロジェクト
- 現時点で 30 以上のサブプロジェクトが存在

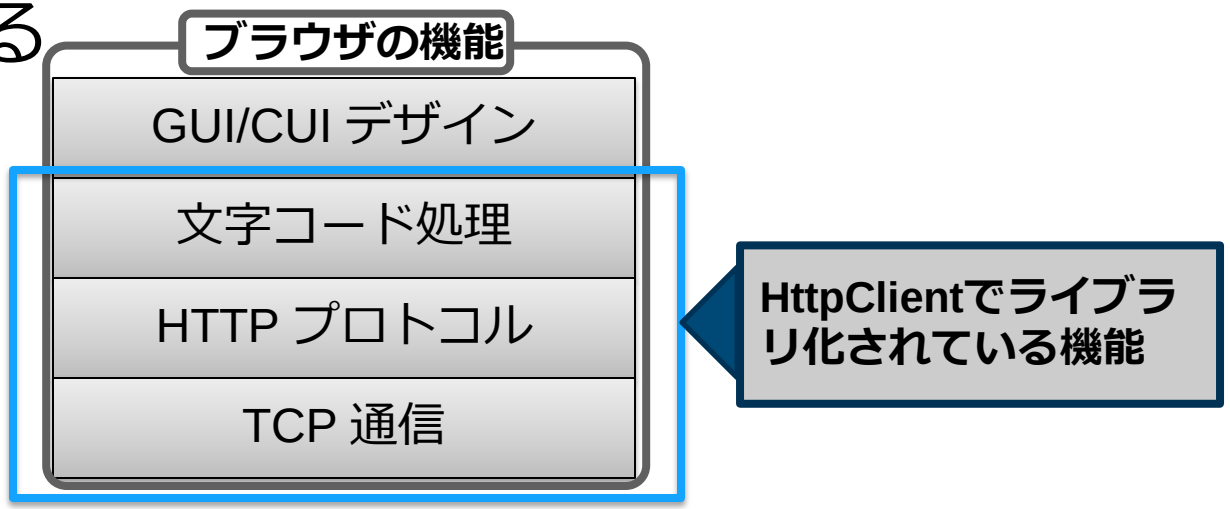
カテゴリ	主なサブプロジェクト名
Web関連	FileUpload, HttpClient※1, Net …
XML関連	Betwixt, Designer, Jelly, JXPath …
ユーティリティ	BeanUtils, Configuration, Logging …

…etc

※1 HttpClient は 2013.06 現在 HTTP Components プロジェクトに移行されている

HttpClient とは

- HTTPクライアントを開発するためのクラスライブラリ
- HTTP通信に関わる基本的な処理がライブラリ化されている
- HttpClientを利用することでHTTP通信を行うアプリケーション(ブラウザ等)を効率よく開発することができる



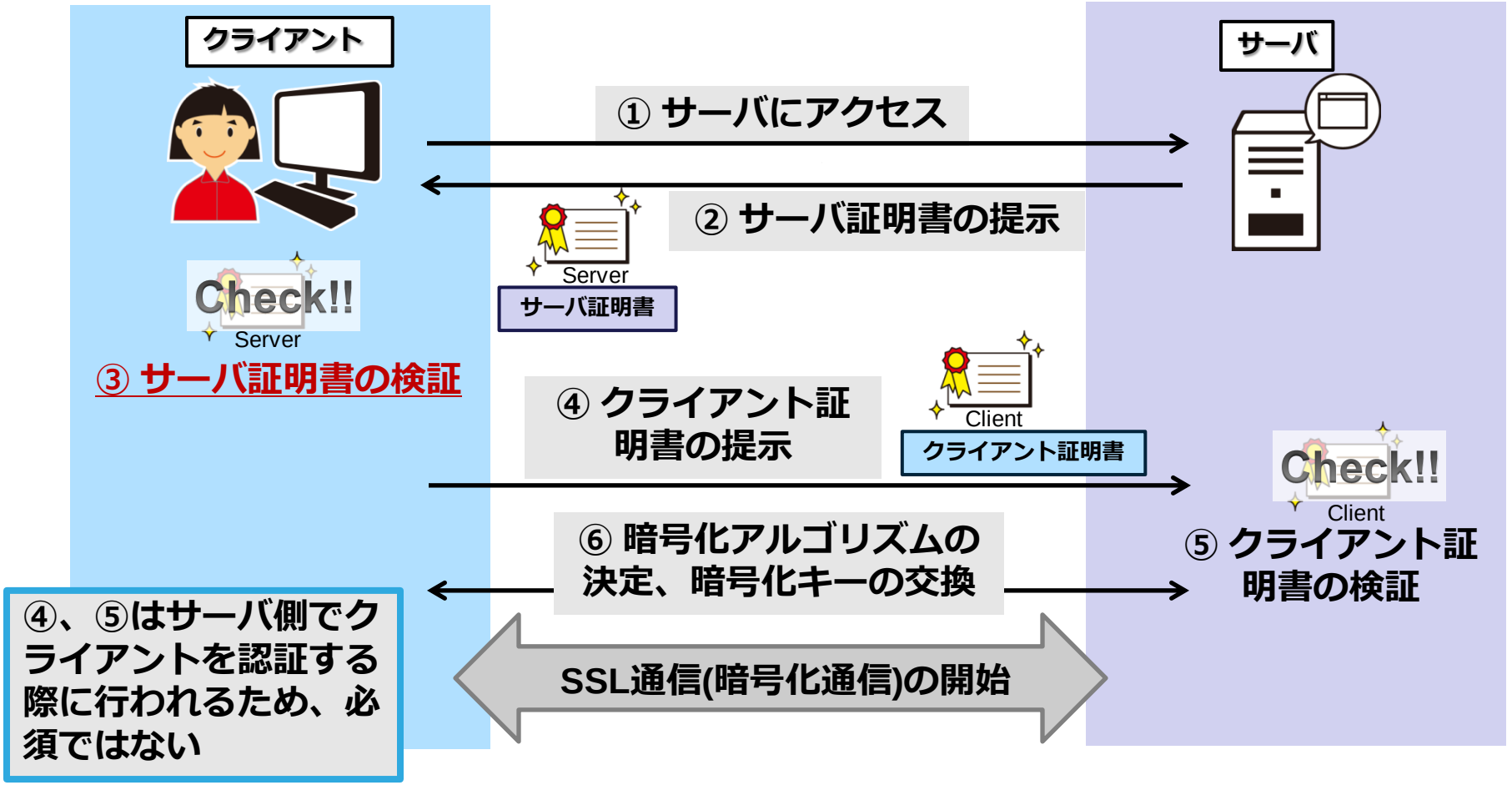
脆弱性の概要

- HttpClientがSSL通信を行う際に、接続先サーバのサーバ証明書の検証が行われていなかった
- そのため、HttpClientを使用したアプリケーションは、正規の証明書を持たないサーバへのSSL通信をエラーなしで許可してしまった

※本ドキュメントは、脆弱性を内包する Apache Commons 2.0.1 を対象に記載している。
(脆弱性は Apache Commons 2.0.1 以降に存在)

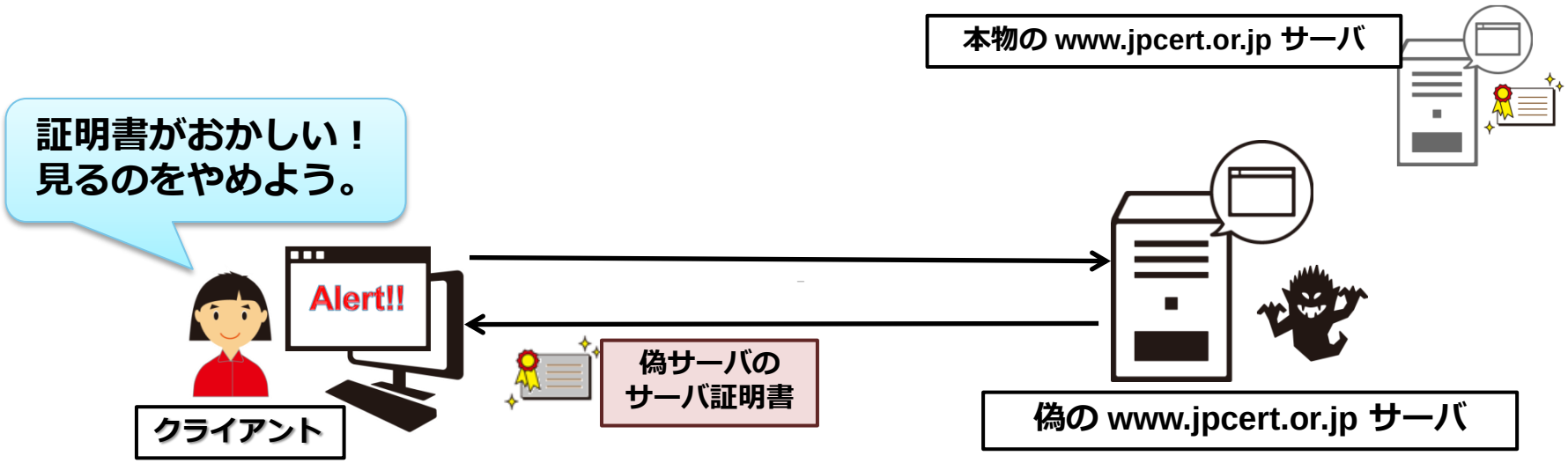
SSL通信における証明書認証

- SSL通信では以下のようなやりとりが行われる



サーバ証明書の検証

- クライアント-サーバ間でSSL通信(暗号化通信)をすることで、盗聴による情報漏えいを防ぐことができる
- ただし、接続先サーバが不正なホストに入れ替わってしまうとSSL通信(暗号化通信)の意味がない
- そこで、SSL通信開始時にサーバ証明書を検証することで、接続先サーバの身元を確認する



サーバ証明書検証に関する情報

■RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
(http://tools.ietf.org/html/rfc5280)

SSL証明書の検証内容として以下のような項目を挙げ、
検証に関するアルゴリズムを定義している。
(Section6. Certification Path Validation)

1.発行元の認証機関(CA)が**信頼されているCAであることを確認**する
2.現在の日付が**有効期間内にあるかどうかを確認**する
3.現時点において、**証明書が失効していないことを確認**する
4.発行元CAの公開キーにより、**送信者のデジタル署名を検証**する
5.サーバーの証明書の**ドメイン名がサーバー自体のドメイン名と一致するかどうかを確認**する

注意!!

この資料では、5. のドメイン名(ホスト名)を検証する処理を中心に見ていく。

サーバ証明書の記載内容

サーバ証明書には、以下のような情報が含まれる。

項目名	記載内容	
CN	サーバ名（ドメイン含む）	www.jpcert.or.jp
OU	運営団体の部署名	System Administration Group
O	運営団体名	Japan Computer Emergency Response Team Coordination Center
L	運営団体の所在（市区町村）	Chiyoda-ku
ST	運営団体の所在（都道府県）	Tokyo
C	運営団体の所在（国名）	Japan

- 証明書の有効期間
- 証明書を発行した証明機関(ＣＡ)のパス

※CNには１つのホストしか記述できないため、複数ホスト(バーチャルホスト)運用を想定した拡張フィールド subjectAltName も用意されており、CNフィールドよりも優先される。

サーバ証明書の検証内容

RFC 2818 「HTTP Over TLS」 では以下のように記載されている。

English

the client **MUST** check it against the server's identity as presented in the server's Certificate message, in order to prevent man-in-the-middle attacks.
(中略)
the (most specific) Common Name field in the Subject field of the certificate **MUST** be used.

日本語

中間者による攻撃を予防するために、クライアントは、これをサーバーの証明書メッセージ中に示されたサーバーの身元に照らしてチェックしなければなりません(**MUST**)。
(中略)
証明書の Subject フィールドにある(最も具体的)な Common Name フィールドを使用しなければなりません(**MUST**)。

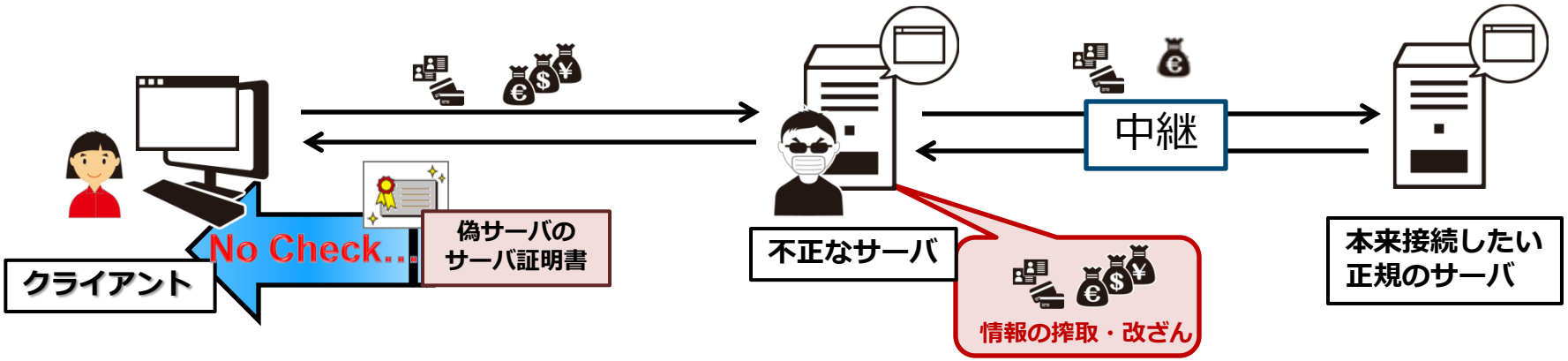
RFC では「証明書のCN(またはsubjectAltName)の値」と「ホスト名」の一致を検証することが必須、とされている。

脆弱性が悪用された場合のリスク

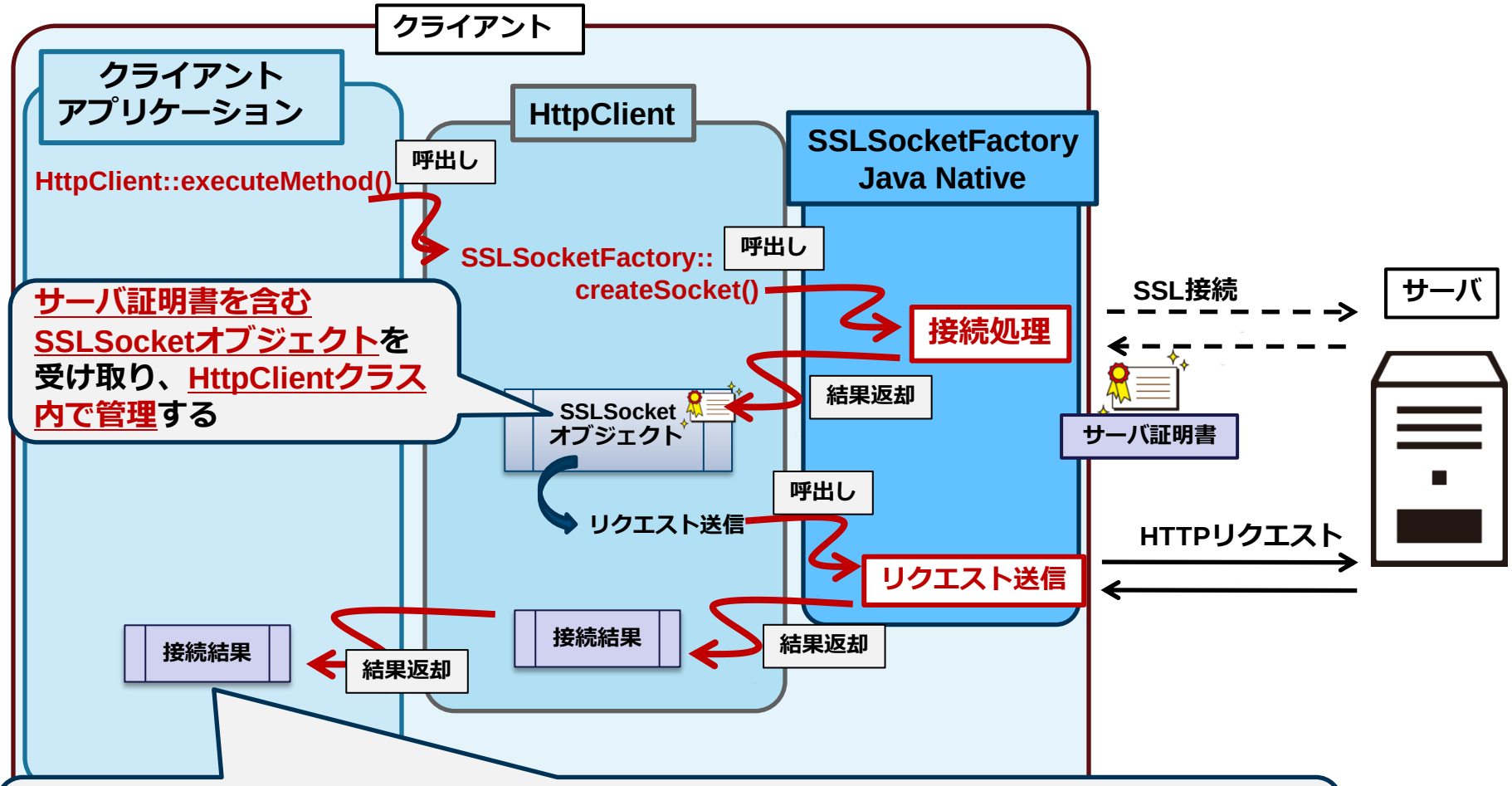
- SSLサーバの成りすましが可能
- それによりSSL通信の中間者攻撃(Man-In-The-Middle 攻撃)が可能
  - —SSL通信内容の傍受、改ざんが可能。

脆弱性の悪用例(Man-In-The-Middle攻撃)

- クライアントを、本来接続するサーバとは異なる不正なサーバに接続させる
- 不正なサーバでは、クライアントに代わって正規のサーバと通信し、クライアントに対しては正規のサーバとしてふるまう
- 不正なサーバ上で通信情報の傍受や改ざんが可能となる



HTTPSアクセス時の実行フロー



ソースコード解説

① クライアントアプリケーションがHttpClientライブラリのexecuteMethodメソッドを呼び出す。

② executeMethodメソッドからjavax.net.ssl.SSLSocketFactory クラスのcreateSocketメソッドが呼び出され 、サーバとのSSL 接続が確立する。

③ HttpClientライブラリにコネクションオブジェクト(SSLSocketオブジェクト)が返却される。

④ HttpClientがSSLSocketを使用してHTTPリクエストを送信する。

⑤ サーバからHTTPレスポンスを受け取ってクライアントアプリケーションに返却する。

①クライアントアプリケーションがHttpClientのexecuteMethodメソッドを呼び出す

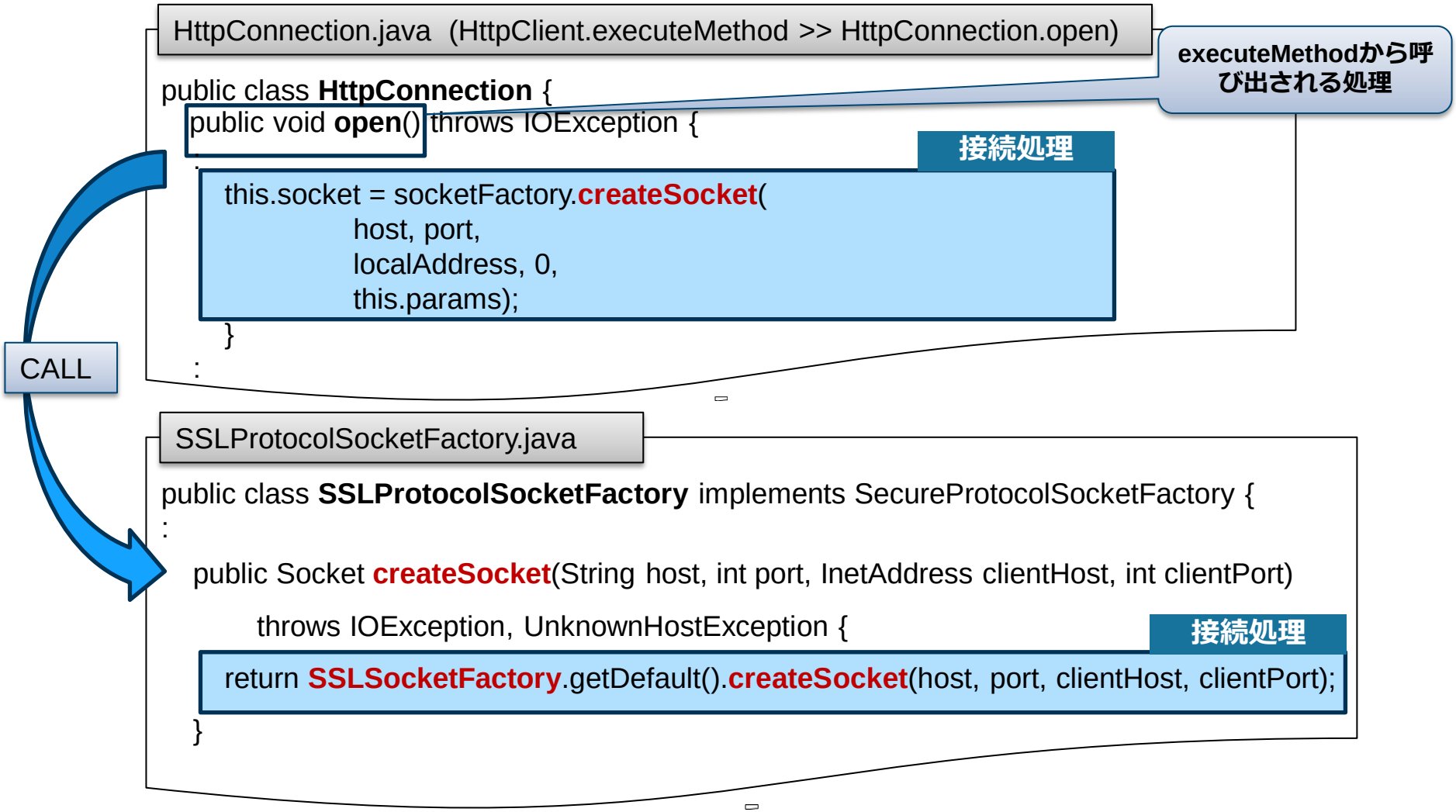
SimpleClient.java

```
public class SimpleClient {
    public static void main(String[] args) {
        try {
            HttpClient httpclient = new HttpClient();
            GetMethod httpget = new GetMethod("https://www.jpcert.or.jp/");
            try {
                int statusCode = httpclient.executeMethod(httpget);
                if (statusCode != HttpStatus.SC_OK) {
                    // Error handling here
                }
                byte[] response = httpget.getResponseBody();
            } finally {
                httpget.releaseConnection();
            }
:
```

接続処理

HttpClient インスタンスを作成、Get メソッドパラメータを構成し executeMethodを呼び出す。

② executeMethod内でSSLSocketFactoryクラスのcreateSocketメソッドが呼び出され、サーバとのSSL接続が確立する。



③HttpClientライブラリにコネクションオブジェクト(SSLSocketオブジェクト)が返却される。

SSLProtocolSocketFactory.java

```
public class SSLProtocolSocketFactory implements SecureProtocolSocketFactory {
:
    public Socket createSocket(String host, int port, InetAddress clientHost, int clientPort)
            throws IOException, UnknownHostException {
        return SSLSocketFactory.getDefault().createSocket(host, port, clientHost, clientPort);
    }
```

SSLSocketFactory::createSocketメソッドはSSLSocketオブジェクトを返却する

Return

HttpConnection.java (HttpClient.executeMethod >> HttpConnection.open)

```
public class HttpConnection {
public void open() throws IOException {
    .
    this.socket = socketFactory.createSocket(
                  host, port,
                  localAddress, 0,
                  params);
    }
    :
}
```

HttpClientライブラリのHttpConnectionクラスの内部に、SSLSocketオブジェクトを格納する。

④ HttpClientがSSLSocketを使用してHTTPリクエストを送信する。

HttpMethodDirector.java (HttpClient.executeMethod > HttpMethodDirector.executeWithRetry)

```
private void executeWithRetry(final HttpMethod method)
    throws IOException, HttpException {
    :
    this.conn.open();
    :
    applyConnectionParams(method);
    method.execute(state, this.conn);
```

リクエスト送信処理

その後、HttpClientライブラリ内に保持したSSLSocketオブジェクトを利用してリクエスト送信が行われる。

⑤ サーバからHTTPレスポンスを受け取りクライアントアプリケーションに返却する。

SimpleClient.java

```
public class SimpleClient {
    public static void main(String[] args) {
        try {
            HttpClient httpclient = new HttpClient();
            GetMethod httpget = new GetMethod("https://www.jpcert.or.jp/");
            try {
                int statusCode = httpclient.executeMethod(httpget);
                if (statusCode != HttpStatus.SC_OK) {
                    // Error handling here
                }
                byte[] response = httpget.getResponseBody();
            } finally {
                httpget.releaseConnection();
            }
:
```

レスポンス返却

executeMethodを呼び出した結果(送信したリクエストに対するレスポンス)をクライアントアプリケーションに返却する

問題点とその対策

- **今回のアプリケーションにおける具体的な問題点**

  前述の RFC 2818「HTTP Over TLS」に沿っていない！

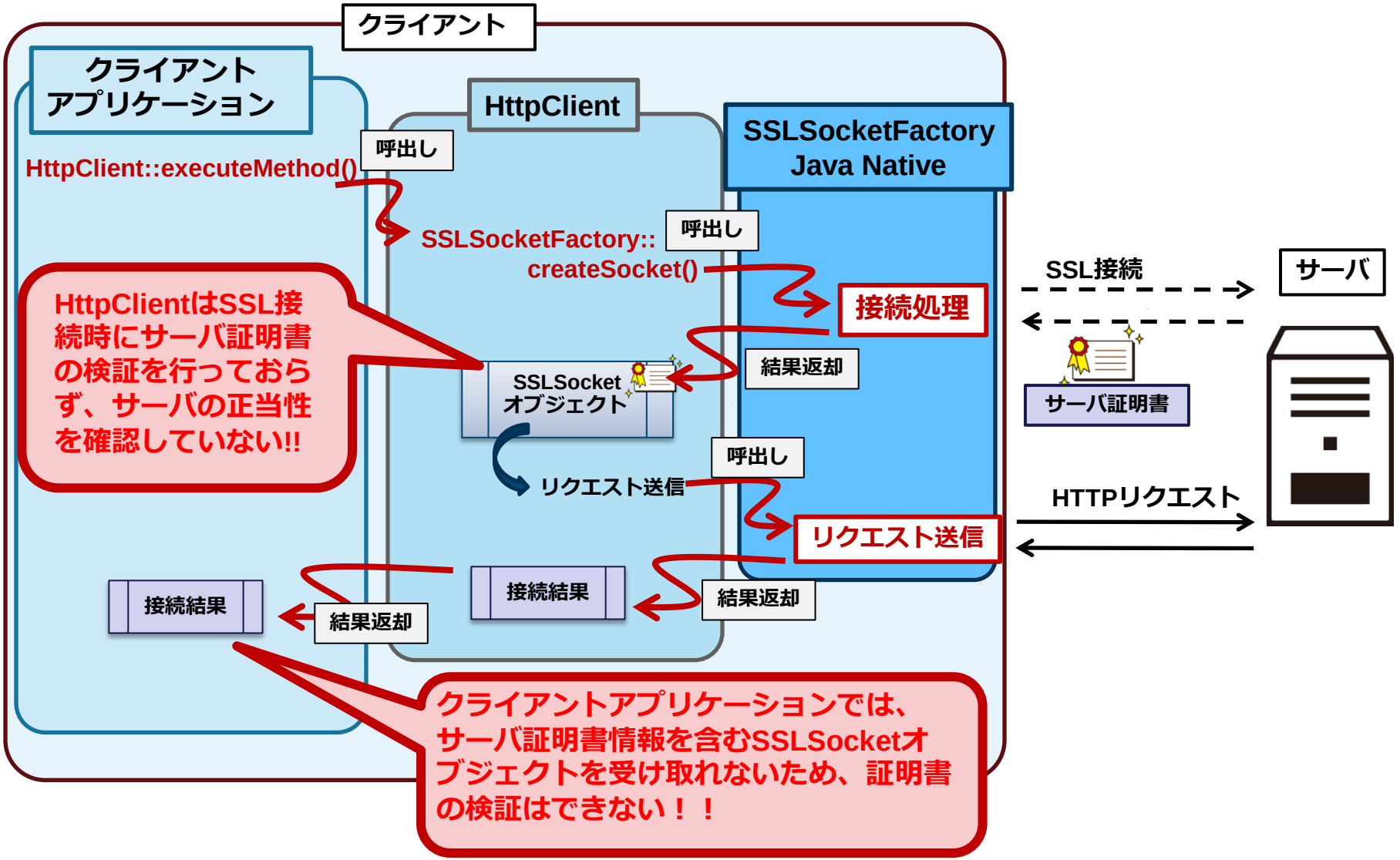
> (証明書の検証方法として、)証明書の Subject フィールドにある(最も具体的)な Common Name フィールドを使用しなければなりません(**MUST**) 。

  サーバへ接続後、SSLSocket がサーバ証明書を取得しているが、その検証を一切行っていなかったため、サーバ名(CNやsubjectAltName)が異なるサーバでも正常に繋がってしまった。

- **問題点に対してどうすべきだったか**

  サーバ証明書の検証を実施する。不正なサーバ証明書であることが判明したら、エラー状態として扱う。

HTTPSアクセス時の実行フローにおける問題点



HTTPSアクセス時の実行フローにおける問題点

「**③HttpClientライブラリにコネクションオブジェクト（SSLSocketオブジェクト）が返却される。**」のコードに問題がある。

SSLProtocolSocketFactory.java

```
public class SSLProtocolSocketFactory implements SecureProtocolSocketFactory {
:
    public Socket createSocket(String host, int port, InetAddress clientHost, int clientPort)
            throws IOException, UnknownHostException {
        return SSLSocketFactory.getDefault().createSocket(host, port, clientHost, clientPort);
    }
```

Point!

createSocket を呼び出した後、返却される Socket 型オブジェクトの内容を全く検証せず呼び出し元に返却している。

修正版コード

サーバ接続時のフローに証明書検証処理 ③’ が追加された。

① クライアントアプリケーションがサーバへの接続処理においてHttpClientライブラリのexecuteMethodメソッドを呼び出す。

② executeMethodメソッドからjavax.net.ssl.SSLSocketFactory クラスのcreateSocketメソッドが呼び出され、サーバとのSSL接続が確立する。

③ HttpClientライブラリにコネクションオブジェクト(SSLSocketオブジェクト)が返却される。

③’ SSLSocketオブジェクトに含まれるサーバ証明書を検証する

④ HttpClientがSSLSocketを使用してHTTPリクエストを送信する。

⑤ サーバからHTTPレスポンスを受け取ってクライアントアプリケーションに返却する。

修正版コード

ソケット情報から証明書情報を取得し、ホスト名検証を行う。

SSLProtocolSocketFactory.java

```
public Socket createSocket(String host, int port, InetAddress clientHost, int clientPort)
   throws IOException, UnknownHostException {
   Socket sslSocket = SSLSocketFactory.getDefault().createSocket(host, port, clientHost, clientPort);

   verifyHostName(host, (SSLSocket) sslSocket);
   return sslSocket;
}
```

接続処理

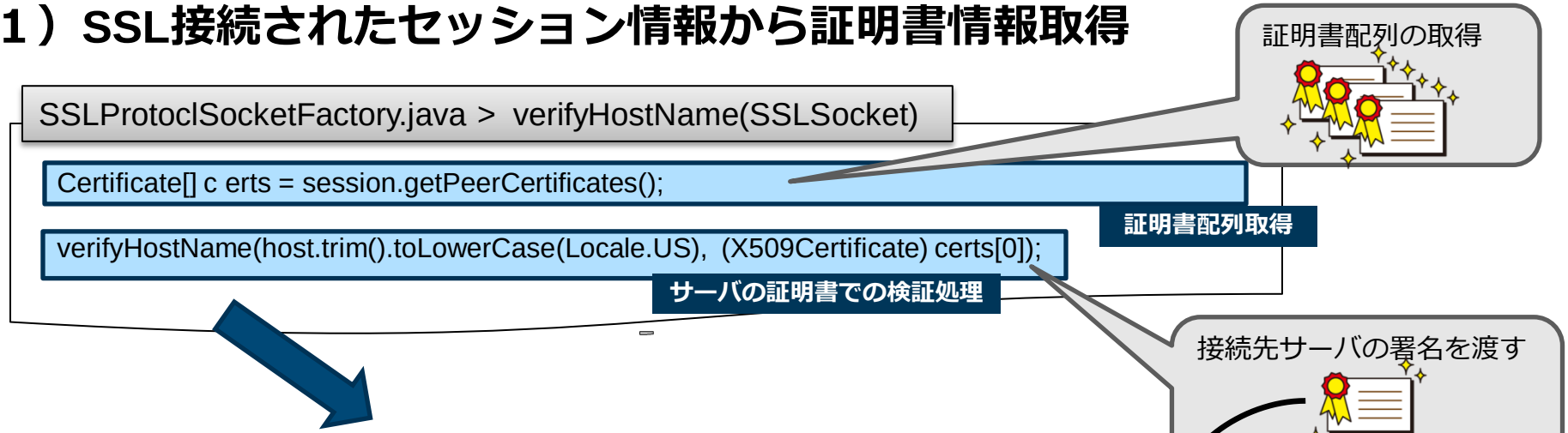
ホスト名検証処理

返却されたソケット情報(SSLSocket)を sslSocket に格納し、ホスト名を検証するメソッド verifyHostName() に渡すように修正。
ホスト名が一致しなかった場合は、SSLException がスローされ、接続処理が異常終了するようになった。

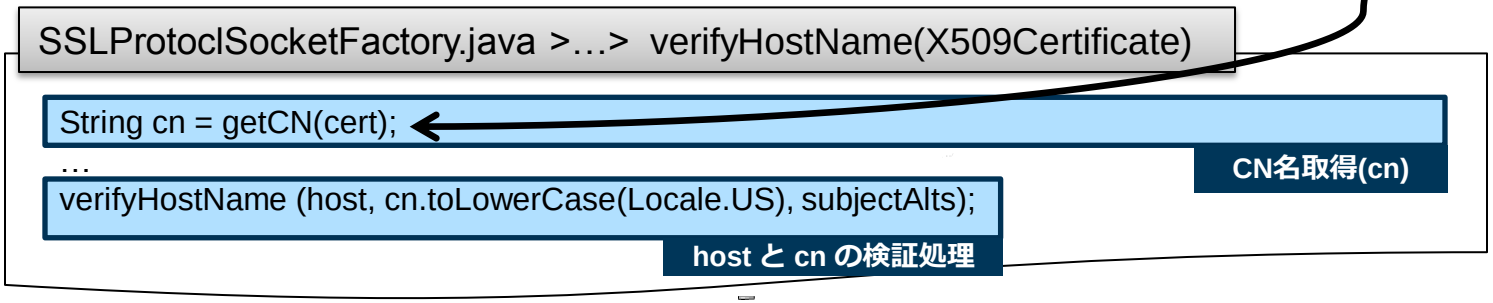
※ SSLException は IOException の派生クラス

補足：ホスト名検証処理

1）SSL接続されたセッション情報から証明書情報取得



2）証明書情報から CN(subjectAltName) エントリを取得し比較



※ 2) で呼び出される verifyHostName 内にて、適切にワイルドカードが使われているCNも併せて解決され、接続しようとしているホスト(host)と証明書のCNレコード(cn)が一致するかチェックされる。

サーバ証明書検証に関する参考情報(1)

(再掲)

■RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
(http://tools.ietf.org/html/rfc5280)

SSL証明書の検証内容として以下のような項目を挙げ、
検証に関するアルゴリズムを定義している。
(Section6. Certification Path Validation)

1.発行元の認証機関(CA)が**信頼されているCAであることを確認**する
2.現在の日付が**有効期間内にあるかどうかを確認**する
3.現時点において、**証明書が失効していないことを確認**する
4.発行元CAの公開キーにより、**送信者のデジタル署名を検証**する
5.サーバーの証明書の**ドメイン名がサーバー自体のドメイン名と一致するかどうかを確認**する

サーバ証明書検証に関する参考情報(2)

■Microsoft「SSL ハンドシェイクでのサーバー認証プロセスに関する説明」より
http://support.microsoft.com/kb/257587/ja

SSL を有効にしたクライアントでは、以下の手順を実行してサーバーの身元を認証します。
1. 現在の日付が**有効期間内にあるかどうかを確認**します。
2. 発行元の認証機関(CA)が**信頼されている CA であることを確認**します。
3. 発行元 CA の公開キーにより、**送信者のデジタル署名を検証**します。
4. サーバー証明書の**ドメイン名がサーバー自体のドメイン名と一致するかどうかを確認**します。

■Google「Browser Security Handbook」より
http://code.google.com/p/browsersec/wiki/Part2

ここでは以下を不正な証明書として扱っている。
- 信頼される機関によってサインされていない
- 期限切れ
- ドメイン名と合致しない

(引用、原文そのまま)
invalid certificates (not signed by a trusted entity, expired, not matching the current domain, or suffering from any other malady)

CWE: Common Weakness Enumeration

- CWE-295: Improper Certificate Validation
  ⇒全般的な話、証明書を正しく検証していない
- CWE-296: Improper Following of a Certificate's Chain of Trust
  ⇒証明書チェーンを正しく検証していない
- CWE-298: Improper Validation of Certificate Expiration
  ⇒期限が切れているかを正しく検証していない
- CWE-299: Improper Check for Certificate Revocation
  ⇒証明書が無効になっていないかを正しく検証していない

http://cwe.mitre.org/

著作権・引用や二次利用について

■本資料の著作権はJPCERT/CCに帰属します。

■本資料あるいはその一部を引用・転載・再配布する際は、引用元名、資料名および URL の明示をお願いします。

記載例

引用元：一般社団法人JPCERTコーディネーションセンター
Java アプリケーション脆弱性事例解説資料
Apache Commons の HttpClient における SSL サーバ証明書検証不備
https://www.jpcert.or.jp/securecoding/2012/No.07_Apache_Commons.pdf

■本資料を引用・転載・再配布をする際は、引用先文書、時期、内容等の情報を、JPCERT コーディネーションセンター広報(office@jpcert.or.jp)までメールにてお知らせください。なお、この連絡により取得した個人情報は、別途定めるJPCERT コーディネーションセンターの「プライバシーポリシー」に則って取り扱います。

本資料の利用方法等に関するお問い合わせ
JPCERTコーディネーションセンター
広報担当
E-mail：office@jpcert.or.jp

本資料の技術的な内容に関するお問い合わせ
JPCERTコーディネーションセンター
セキュアコーディング担当
E-mail：secure-coding@jpcert.or.jp