ここが変だよ、
グローバルスタンダードの脆弱性対策
～入力値の考え方～

2012年3月27日

徳丸　浩

OWASP Japan 1st Local Chapter Meeting まことにおめでとうございます

OWASPと言えば･･･

OWASP TOP 10

OWASP TOP 10とは

セキュリティ動向チェック
Security&Trustウォッチ(47)
Webアプリケーションを作る前に知るべき10の脆弱性

上野宣
2007/6/13

また上野宣か

Webアプリケーションが攻撃者に付け込まれる脆弱性の多くは、設計者や開発者のレベルで排除することができます。実装に忙しい方も、最近よく狙われる脆弱性のトップ10を知ることで手っ取り早く概要を知り、開発の際にその存在を意識してセキュアなWebアプリケーションにしていただければ幸いです。

■Webの世界を脅かす脆弱性を順位付け

OWASP(Open Web Application Security Project)は、主にWebアプリケーションのセキュリティ向上を目的としたコミュニティで、そこでの調査や開発の成果物を誰でも利用できるように公開しています。

その中の「OWASP Top Ten Project」というプロジェクトでは、年に1回Webアプリケーションの脆弱性トップ10を掲載しています。2004年版は日本語を含む各国語版が提供されていますが、2007年版は現在のところ英語版のみが提供されています。

http://www.atmarkit.co.jp/fsecurity/column/ueno/47.html より引用

OWASP Top 10 2004

A1: Unvalidated Input

A2: Broken Access Control

A3: Broken Authentication and Session Management

A4: Cross Site Scripting

A5: Buffer Overflow

A6: Injection Flaws

A7: Improper Error Handling

A8: Insecure Storage

A9: Application Denial of Service

A10: Insecure Configuration Management

https://www.owasp.org/index.php/Top_10_2004 参照

PCI DSSでも

6.5 すべての Webアプリケーションは「Open Web Application Security Project」ガイドラインなどの安全なコーディング・ガイドラインに基づいて開発する。コーディングの脆弱性を特定するために、カスタム・アプリケーション・コードを見直す。次に示すような、ソフトウェア開発プロセスにおける共通のコーディング脆弱性の防止に努める。

6.5.1 入力データの未検証(Unvalidated input)

6.5.2 アクセス制御の不徹底（例えば、ユーザーIDの悪用）

6.5.3 認証・セッション管理の不徹底（アカウント信用証明とセッション・クッキーの使用）

6.5.4 XSS（cross-site scripting）攻撃

6.5.5 バッファ・オーバーフロー

6.5.6 入力不正（例えば、SQL＝structured query languageインジェクション）

6.5.7 不適切なエラー処理

6.5.8 安全でない保管

6.5.9 サービスの拒否。

6.5.10 信頼できない設定管理

PCI DSS v1.1 日本語訳 https://www.pcisecuritystandards.org/security_standards/documents.php より引用

「入力データの未検証」ってなんだ?

CWEに聞いてみよう

CWEとは

共通脆弱性タイプ一覧CWE（Common Weakness Enumeration）は、ソフトウェアにおけるセキュリティ上の弱点（脆弱性）の種類を識別するための共通の基準を目指しています。

1999年頃から米国政府の支援を受けた非営利団体のMITREが中心となり仕様策定が行われ、2006年3月に最初の原案が公開されました。その後、40を超えるベンダーや研究機関が協力して仕様改善や内容拡充が行われ、2008年9月9日にCWEバージョン1.0が公開されました。

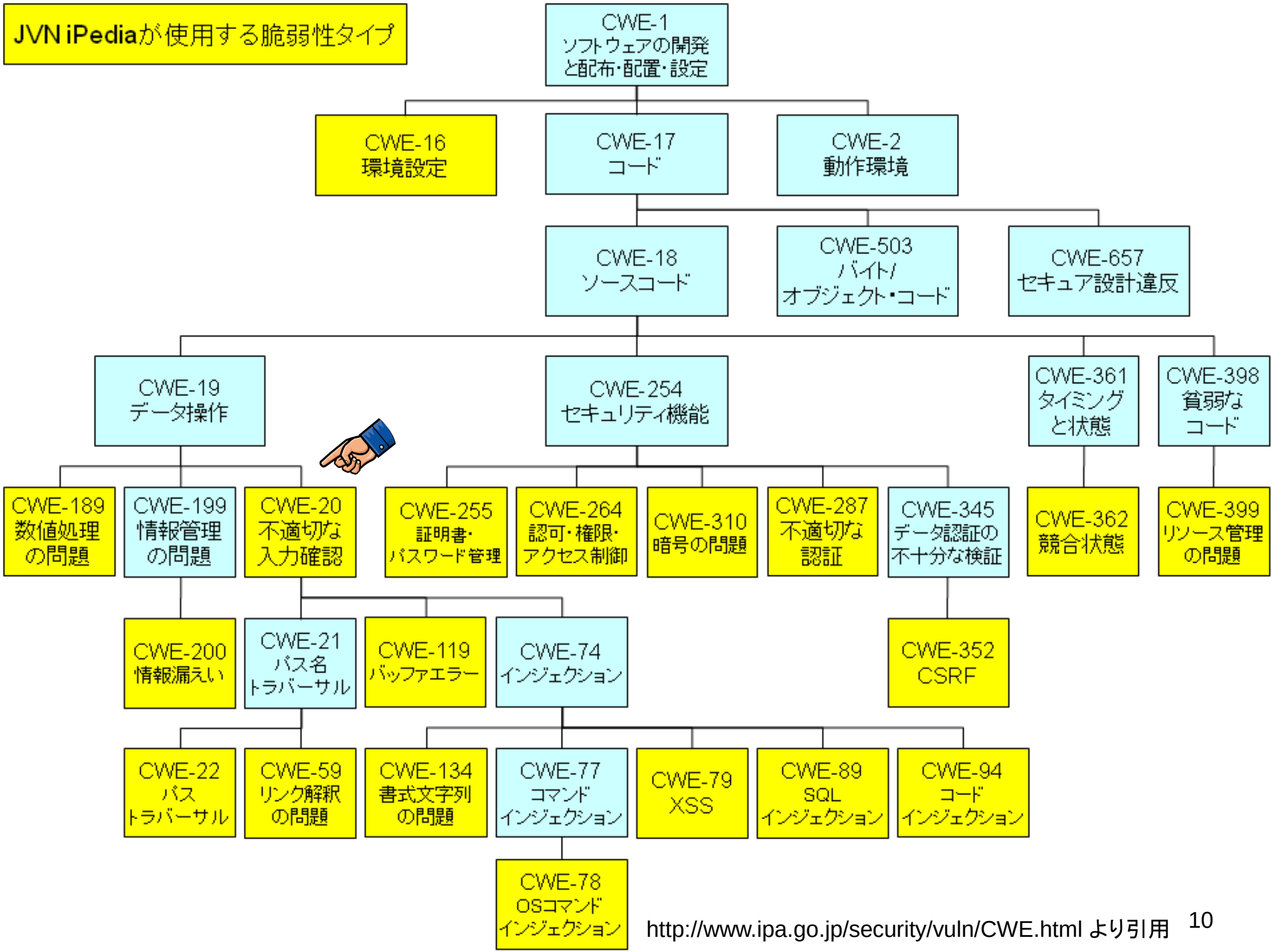
CWEでは、SQLインジェクション、クロスサイト・スクリプティング、バッファオーバーフローなど、多種多様にわたるソフトウェアの脆弱性を識別するための、脆弱性の種類（脆弱性タイプ）の一覧を体系化して提供しています。CWEを用いると、ソフトウェア開発者やセキュリティ専門家などに次のようなメリットがあります。

1. ソフトウェアのアーキテクチャ、デザイン、コードに内在する脆弱性に関して、共通の言葉で議論できるようになる。
2. 脆弱性検査ツールなど、ソフトウェアのセキュリティを向上させるための、ツールの標準の評価尺度として使用できる。
3. 脆弱性の原因を認識し、脆弱性の低減を行い、再発を防止するための共通の基準として活用できる。

現在、CWEは、NISTのNVD、OWASPのTop Ten Projectや、いくつかのセキュリティベンダーなどで実際に活用されています。

http://www.ipa.go.jp/security/vuln/CWE.html より引用

JVN iPediaが使用する脆弱性タイプ



http://www.ipa.go.jp/security/vuln/CWE.html より引用

CWE-20: Improper Input Validation

Improper Input Validation

Weakness ID: 20 *(Weakness Class)* **Status:** Usable

Description

Description Summary

The product does not validate or incorrectly validates input that can affect the control flow or data flow of a program.

Extended Description

When software does not validate input properly, an attacker is able to craft the input in a form that is not expected by the rest of the application. This will lead to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution.

http://cwe.mitre.org/data/definitions/20.html より引用

不適切な入力確認

解説

解説要約

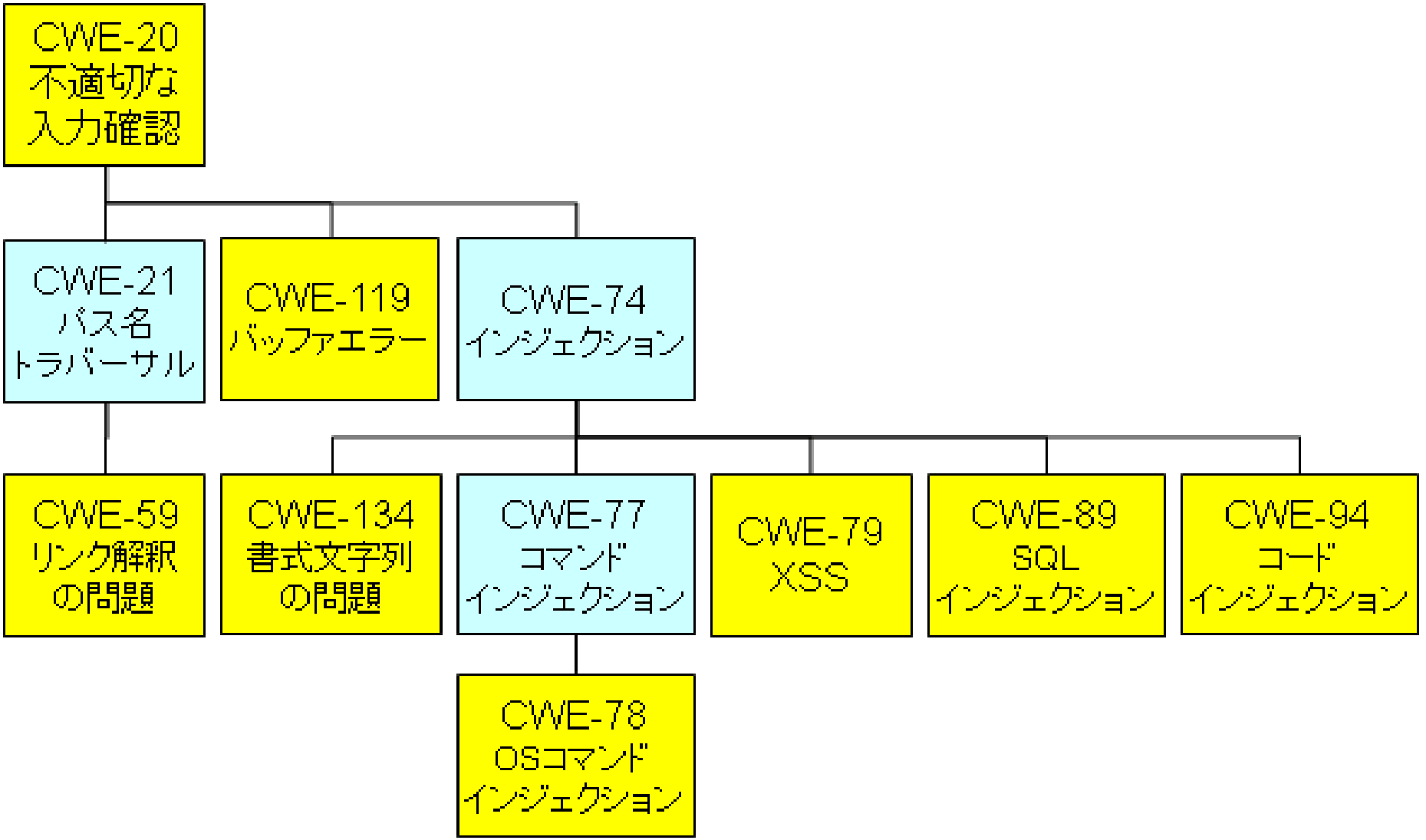
この脆弱性がある製品は、プログラムの制御フローおよびデータフローへ影響を及ぼす入力に対し、適切な妥当性チェックを行いません。

詳細な解説

ソフトウェアにおける入力の妥当性の確認が不十分な場合、攻撃者が他のアプリケーションのフォームに意図しない入力をする可能性があります。この入力はシステムの一部に受け渡され、制御フローの改ざん、任意のリソースの制御、任意のコードを実行される可能性があります。

http://jvndb.jvn.jp/ja/cwe/CWE-20.html より引用

CWE-20範囲広すぎwww



http://www.ipa.go.jp/security/vuln/CWE.html より引用

そもそも入力値検証でよいのか？

米国の書籍の解説例

バリデーション至上主義な説解説例1

ホワイトリストフィルタのルールは、甘過ぎても厳格過ぎても困ります。厳格すぎるパターンは、攻撃者としては付け入る隙を見つけにくくなるため、アプリケーションのセキュリティという観点からは容認されます。しかし利便性の観点からは、問題となります。正当なユーザーの実在する電子メールアドレスが拒否された場合、そのユーザーはサイトを利用できなくなり、大きな損害を被るおそれがあります。

試行錯誤の末、電子メールアドレスについては、次のようなルールに落ち着きます。

- アドレスの名前部分は英数字を基本とし、オプションとしてハイフンとピリオドも使用できる。ハイフンまたはピリオドの次には、英数字が続かなければならない
- 名前部分の次には、@記号が続かなければならない
- @記号の次には、アドレスのドメイン部分が続かなければならない。この部分は、最低でも1つ、最高で3つのテキストブロックで構成される必要がある。各テキストブロックは、英数字を基本都市、オプションでハイフンも使用でき、最後がピリオドで終わる。ハイフンの次には英数字が続かなければならない。
- アドレスの最後には.com、.net、.orgなどの有効なトップレベルドメインが、1つだけ置かれなければならない

やれやれ、電子メールアドレスのように単純に思えるものでも、ずいぶん複雑なルールとなりました。

【略】

Billy Hoffman、Bryan Sullivan著、GIJOE監訳、渡邉 了介訳「Ajaxセキュリティ」、毎日コミュニケーションズ、2008年、P112より引用

バリデーション至上主義な説解説例1

ホワイトリストフィルタのルールは、甘過ぎても厳格過ぎても困ります。厳格すぎるパターンは、攻撃者としては付け入る隙を見つけにくくなるため、アプリケーションのセキュリティという観点からは容認されます。しかし利便性の観点からは、問題となります。正当なユーザーの実在する電子メールアドレスが拒否された場合、そのユーザーはサイトを利用できなくなり、大きな損害を被るおそれがあります。

試行錯誤の末、電子メールアドレスについては、次のようなルールに落ち着きます。

- アドレスの
- 名前部分
- @記号の

徳丸　浩
@ockeghem
いわゆる徳丸本、「体系的に学ぶ安全なWebアプリケーションの作り方」http://amzn.to/iLquE6 の著者です。

メアドは'or'1'='1'--@tokumaru.org です

JIオドも使用できる。ハイフンまた

も1つ、最高で3

基本都市、オプションでハイフンも使用でき、最後がピリオドの

ハイフンの次には英数字が続かなければならない。

- アドレスの最後には.com、.net、.orgなどの有効なト

インが、1つだけ置かれなければならない

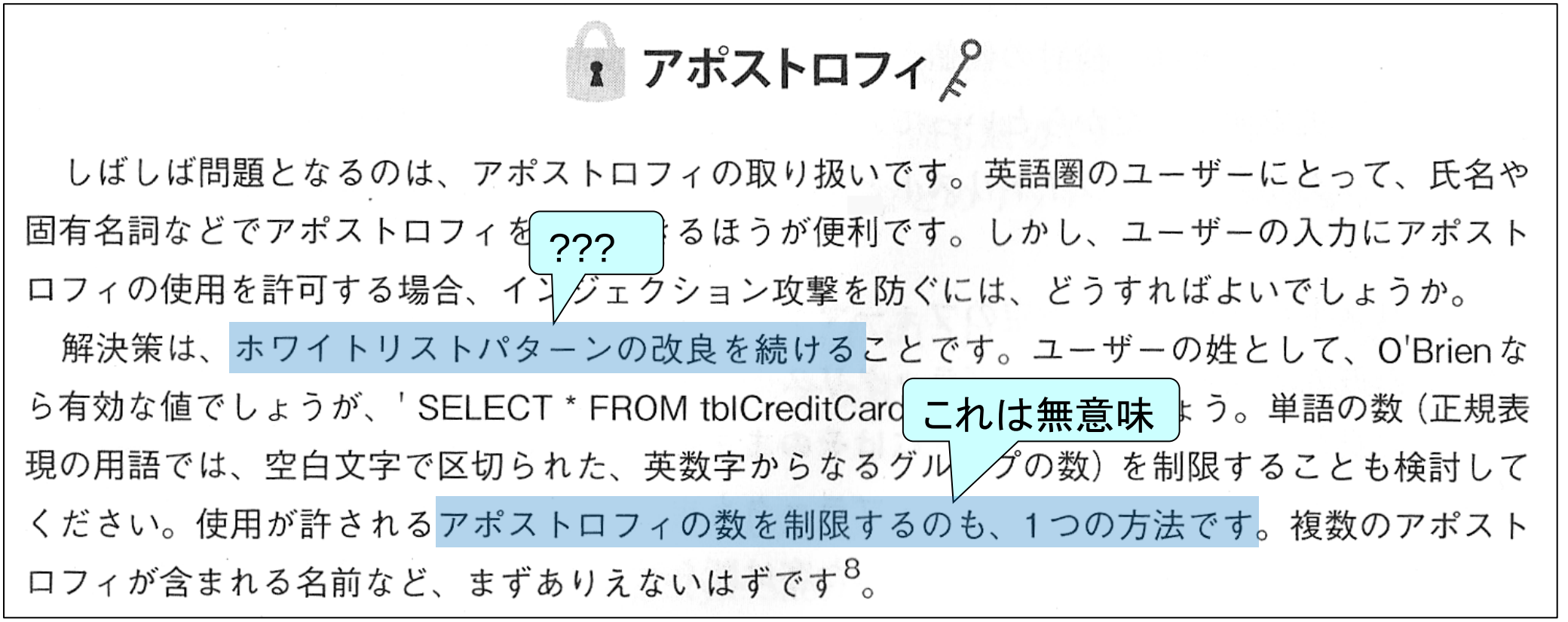
RFC 5322 適合

やれやれ、電子メールアドレスのように単純に思えるものでも、ずいぶん複雑なルールとなりました。

【略】

Billy Hoffman、Bryan Sullivan著、GIJOE監訳、渡邉 了介訳「Ajaxセキュリティ」、毎日コミュニケーションズ、2008年、P112より引用

バリデーション至上主義な説解説例2



アポストロフィ

しばしば問題となるのは、アポストロフィの取り扱いです。英語圏のユーザーにとって、氏名や固有名詞などでアポストロフィを[illegible]るほうが便利です。しかし、ユーザーの入力にアポストロフィの使用を許可する場合、インジェクション攻撃を防ぐには、どうすればよいでしょうか。

解決策は、ホワイトリストパターンの改良を続けることです。ユーザーの姓として、O'Brienなら有効な値でしょうが、' SELECT * FROM tblCreditCard[illegible]う。単語の数（正規表現の用語では、空白文字で区切られた、英数字からなるグループの数）を制限することも検討してください。使用が許されるアポストロフィの数を制限するのも、1つの方法です。複数のアポストロフィが含まれる名前など、まずありえないはずです[8]。

Billy Hoffman、Bryan Sullivan著、GIJOE監訳、渡邉 了介訳「Ajaxセキュリティ」、毎日コミュニケーションズ、2008年、P113より引用

- このような方法では効果が薄いだけではなく、ケースバイケースで正しい方法を考えなければならない点が問題
- 脆弱性対策は、もっと機械的に適用できるものでないと実用的でない

バリデーションで対応できる脆弱性は?

脆弱性パターン名	対応
クロスサイト・スクリプティング	△
SQLインジェクション（文字列）	△
SQLインジェクション（数値）	○
クロスサイト・リクエストフォージェリ	×
推測可能なセッションID	×
URL埋め込みのセッションID	×
セッションIDの固定化	×
オープンリダイレクタ	○
HTTPヘッダ・インジェクション（Cookie）	△
HTTPヘッダ・インジェクション（リダイレクト）	○
メールヘッダ・インジェクション	○
ディレクトリ・トラバーサル	○
OSコマンド・インジェクション	△
ファイルインクルード攻撃	○
evalインジェクション	△

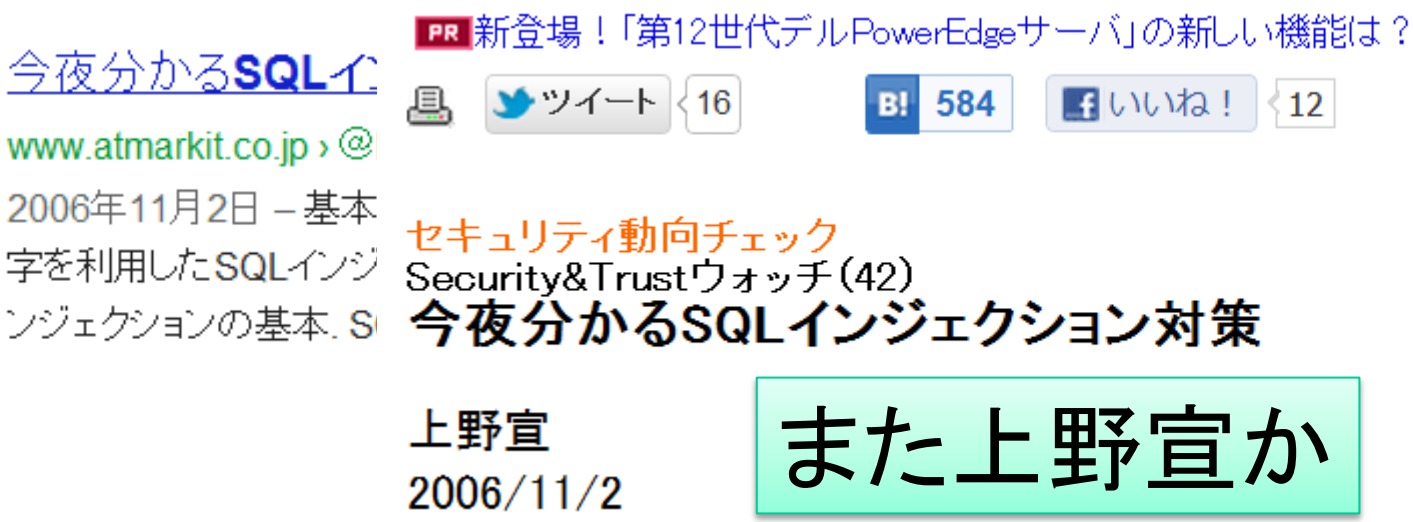
凡例
○：対応可
△：仕様次第
×：対応不可

いったんまとめ

- Validationは、米国（および、“グローバルスタンダード”）ではセキュリティ施策として極めて重要視されている
- Validationを「セキュリティ施策」と見る場合、メリットは、「多くの脆弱性に効き目がある」という「万能性」
- 同じく、デメリットは、「根本的解決でない」こと
  - 「入力値」に着目すると、「セカンドオーダーSQLインジェクション」のような本質的でないものにも注意が必要になる
  - 続きは 

いったんまとめ

- Validationは、米国（および、“グローバルスタンダード”）ではセキュリティ施策として極めて重要視されている
- Validationを「セキュリティ施策」と見る場合、メリットは、「多くの脆弱性に効き目がある」という「万能性」
- 同じく、デメリットは、「根本的解決でない」こと
  - 「入力値」に着目すると、「セカンドオーダーSQLインジェクション」のような本質的でないものにも注意が必要になる
  - 続きは セカンドオーダーSQLインジェクション



http://www.atmarkit.co.jp/fsecurity/column/ueno/42.html より引用

いや、待て
“アメリカ人”を一括りにするな

違う意見の人もいるはず…

OWASP Top 10 2007

A1: Cross Site Scripting (XSS)

A2: Injection Flaws

A3: Malicious File Execution

A4: Insecure Direct Object Reference

A5: Cross Site Request Forgery (CSRF)

A6: Information Leakage and Improper Error Handling

A7: Broken Authentication and Session Management

A8: Insecure Cryptographic Storage

A9: Insecure Communications

A10: Failure to Restrict URL Access

https://www.owasp.org/index.php/Top_10_2007 参照

OWASP Top 10 2010

A1: Injection

A2: Cross-Site Scripting (XSS)

A3: Broken Authentication and Session Management

A4: Insecure Direct Object References

A5: Cross-Site Request Forgery (CSRF)

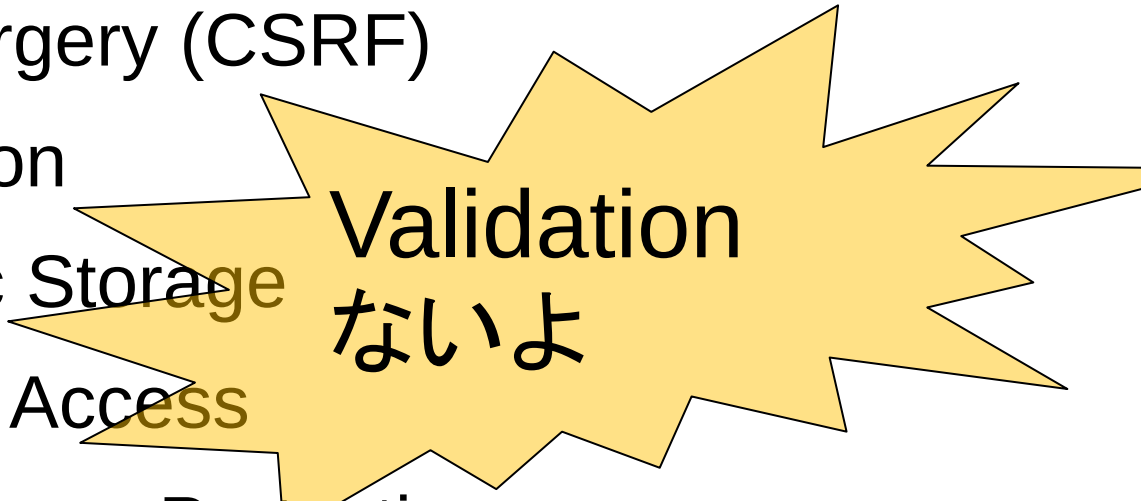
A6: Security Misconfiguration

A7: Insecure Cryptographic Storage

A8: Failure to Restrict URL Access

A9: Insufficient Transport Layer Protection

A10: Unvalidated Redirects and Forwards



https://www.owasp.org/index.php/Top_10_2010 参照

CWE-20の補足を読むと…

Terminology Notes

The "input validation" term is extremely common, but it is used in many different ways. In some cases its usage can obscure the real underlying weakness or otherwise hide chaining and composite relationships.

Some people use "input validation" as a general term that covers many different neutralization techniques for ensuring that input is appropriate, such as filtering, canonicalization, and escaping. Others use the term in a more narrow context to simply mean "checking if an input conforms to expectations without changing it."

http://cwe.mitre.org/data/definitions/20.html より引用

名称補足

「入力の妥当性チェック」という用語は極めて一般的ですが、用語の使い方は様々です。いくつかのケースでは、根本的な脆弱性を曖昧にするためや、関連した複雑な事象を隠すことを目的として使われます。

フィルタリング、正規化やエスケープのような、入力が適切であることを確認する様々な無効化手段をカバーする、総括的な用語としても使用されます。また、もっと狭い文脈において単純に「入力が変化せず、期待される値であることの確認」という意味でも使用されています。

http://jvndb.jvn.jp/ja/cwe/CWE-20.html より引用

お前とはうまい酒が飲めそうだ

まとめ

- OWASP Top 10 2004はかなり変だった
  - 2007, 2010 はかなり良くなったが、ツッコミどころはアリ
- 皆さん、バリデーションはちゃんとしましょうね
  - それが「セキュリティ対策」かどうかは、“どうでもいい”
- バリデーションの“万能性”に惑わされずに、脆弱性対処を淡々とやりましょう
  - それが漏れのない近道
- 米国でも似たような意見を持っている人は多いと予想
  - 私がまだ知らないだけで…まだ見ぬ同士よ!
- 相手が米国人だろうが有名人だろうが、言うべきことは言うぜ
  - OWASP Japanの存在感を!
  - 皆でOWASP Japanを盛り上げよう

Thank you!