[S3]今日こそわかる、安全なWebアプリの作り方2010

HASHコンサルティング株式会社
徳丸 浩
Twitter id: @ockeghem

アジェンダ

- 本日の構成
  - 脆弱性の分類
  - Webアプリの構造と脆弱性の原因箇所
  - 「入力」では何をすればよいのか
  - SQLインジェクション対策の考え方と実際
    - 原理の話（グローバル）
    - 文字コードの話（グローバル&ローカル）
  - ケータイWebアプリのセキュリティ（ローカル）
- 議論の焦点
  - Webアプリケーションのセキュリティ施策の考え方
  - グローバル v.s. ローカル
  - 対策の歴史とあるべき姿

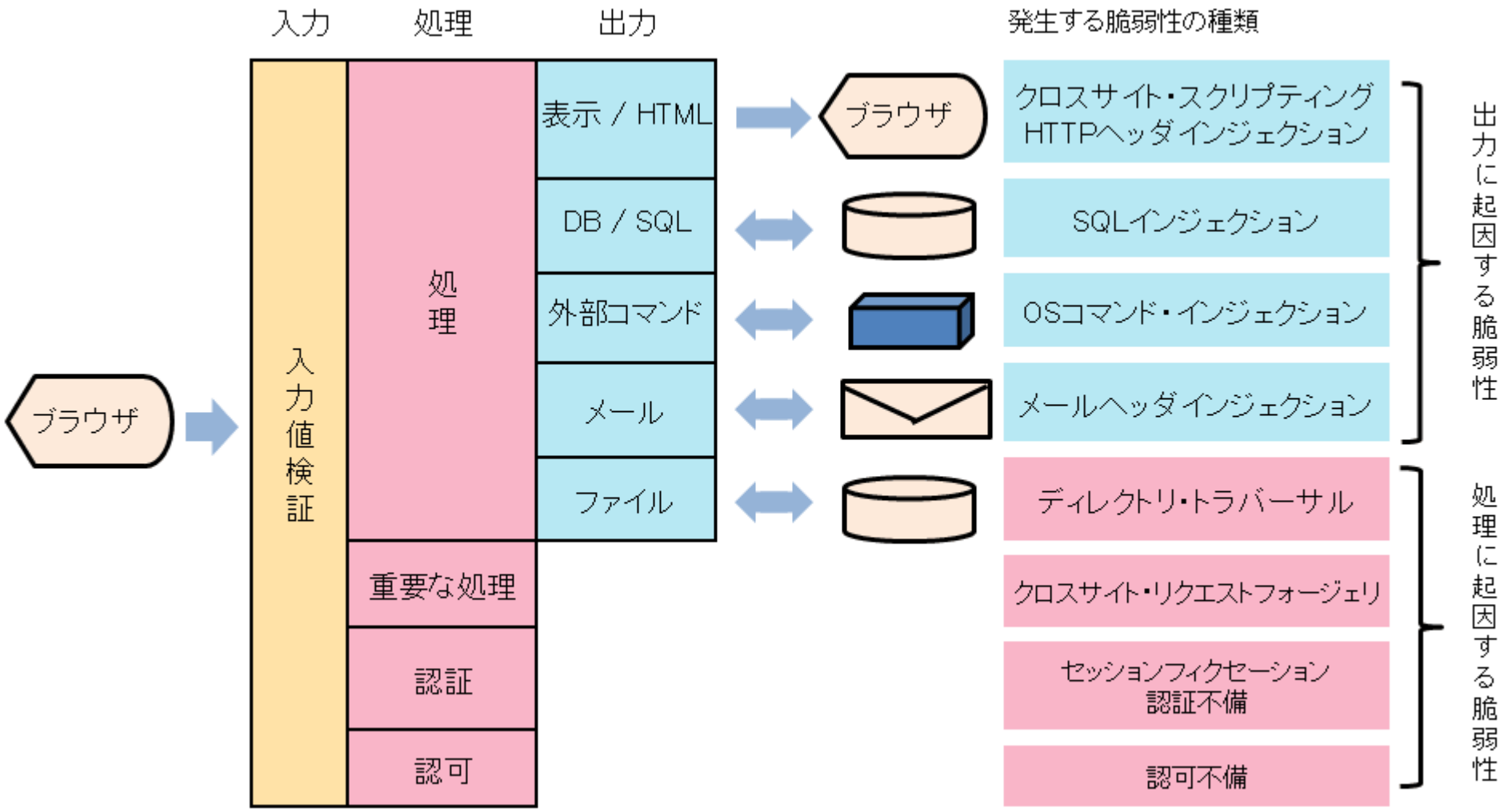
徳丸浩の自己紹介

- 経歴
  - 1985年 京セラ株式会社入社
  - 1995年 京セラコミュニケーションシステム株式会社(KCCS)に出向・転籍
  - 2008年 KCCS退職、HASHコンサルティング株式会社設立
- 経験したこと
  - 京セラ入社当時はCAD、計算幾何学、数値シミュレーションなどを担当
  - その後、企業向けパッケージソフトの企画・開発・事業化を担当
  - 1999年から、携帯電話向けインフラ、プラットフォームの企画・開発を担当
    Webアプリケーションのセキュリティ問題に直面、研究、社内展開、寄稿などを開始
  - 2004年にKCCS社内ベンチャーとしてWebアプリケーションセキュリティ事業を立ち上げ
- その他
  - 1990年にPascalコンパイラをCabezonを開発、オープンソースで公開
    「大学時代のPascal演習がCabezonでした」という方にお目にかかることも
- 現在
  - HASHコンサルティング株式会社　代表　http://www.hash-c.co.jp/
  - 京セラコミュニケーションシステム株式会社　技術顧問　http://www.kccs.co.jp/security/
  - 独立行政法人情報処理推進機構　非常勤研究員　http://www.ipa.go.jp/security/

脆弱性の分類

- 元々セキュリティと無関係なバグが脆弱性となるもの
  - SQLインジェクション
  - クロスサイト・スクリプティング
  - OSコマンドインジェクション
  - HTTPヘッダインジェクション
- セキュリティの考慮不足
  - ディレクトリ・トラバーサル（アクセス制御不足）
  - クロスサイト・リクエストフォージェリ（画面遷移のチェック洩れ）
  - 認証の不備
  - 認可の不備
- 狭義の脆弱性ではないもの
  - HTTPSを使っていない
  - ファイアウォールがない
  - パスワードをデータベース上で暗号化していない

Webアプリの構造と脆弱性の原因箇所



対策の考え方の変遷

- 第1段階：攻撃手法に着目し、攻撃の手段を封じるアプローチ
  - キーワード：サニタイジング
  - メタ文字「<」や「’」、「;」に着目
  - 「危険な文字」として、入力から削除してしまう
- 第2段階：脆弱性が生まれる根本原因に注目し、原因をつぶす
  - キーワード：サニタイズ言うなキャンペーン
  - メタ文字はエスケープするアプローチ
  - メタ文字に頼らない記述ができればベスト
    - Prepared Statement、DOM、シェルを通さない外部コマンド起動
  - 「安全なウェブサイトの作り方」にまとめられる
- 第3段階：ミドルウェアの組み合わせを考慮した現実的な書き方
  - キーワード：安全なSQLの呼び出し方
  - バインド機構を使っていてもSQLインジェクション脆弱性が混入する…
  - →なぜ脆弱性が混入するのか、どうすれば安全なのか

入力値検証ではなにをすべきか?

- 入力値検証はセキュリティのためにするのではない
- 入力値検証は「要件」に従うかどうかのチェック
  - 電話番号 例:099-9999-9999
  - メールアドレス 例: foo@eaxmple.jp
  - 氏名 例:鈴木一朗
- 文字種と文字数のチェックが一般的
- 入力値検証ではSQLインジェクションなどの対策はできない
  - 「‘ union select a,b,c 」などを常にエラーにできるわけではない
  - 現実問題「 O‘reilly」、「O’brien」など、アポストロフィつきの名前は珍しくない
- 入力値検証は「保険的対策」としては有効
  - 本質的な対策が洩れた場合のセーフティネットとして(運が良ければ)
  - 制御文字(メタ文字ではない)による攻撃への防御
  - 文字コードの問題への対策(但し入力値検証だけでは不十分)

入力値検証の正規表現の具体例

チェック内容	正規表現
数字1文字以上，8文字以下	/¥A[0-9]{1,8}¥z/
英字0文字以上，10文字以下	/¥A[a-z]{0,10}¥z/i
「F」または「M」	/¥A(F|M)¥z/
制御文字以外100文字以下	/¥A¥P{Cc}{0,100}¥z/
制御文字以外100文字以下 ただし，改行とタブは認める	/¥A[¥t¥r¥n¥P{Cc}]{0,100}¥z/

- 「制御文字以外」という正規表現は¥P{Cc}と書ける
  - PHPのmb_eregの場合は [[:^cntrl:]]
- 文字列の先頭末尾は ¥A と ¥z で指定すること。^と$は「行」の先頭・末尾
  - $は行末の改行にマッチするので、特に注意
- M文字以上、N文字以下は、{M,N}で指定（量指定子）

入力値検査をしなかった場合の問題

- アプリケーションの処理が進んだ後でエラーになると、データベースの不整合などを起こしやすい
  - 例：数値であるべき項目に、英字や記号が入っているケース
- 制御文字による攻撃
  - ヌルバイト攻撃（ヌル文字）
  - メールヘッダインジェクション（改行）
  - HTTPヘッダインジェクション（改行）
  - 入力値で対策するのはおかしいが、保険的にやっておくとよい
- 壊れた文字コードによる攻撃
  - 半端な先行バイトによる攻撃
  - UTF-8の非最短形式
  - 文字コードの不正は、処理を継続するべきでないので、入力検証で対策する合理性がある

制御文字による攻撃例(ヌルバイト攻撃)

```
<html><body>
<?php
  $p = $_GET['p'];
  if (isset($p) && ereg("^[0-9]+$", $p)) {
    echo $p;          // pが数字のみの場合は表示
  } else {
    echo 'Error:数値を指定してください'; // それ以外はエラー
  }
?>
</body></html>
```

XSS脆弱なサンプル

攻撃例:`p=1%00<script>alert(document.cookie)</script>`

インジェクション系脆弱性対策は出力時のエスケープ

- SQLインジェクション
  - Prepared Statement（メタ文字を不要にする）
  - Quoteメソッドにエスケープ
- クロスサイト・スクリプティング(XSS)
  - HTMLメタ文字のエスケープ
  - JavaScriptの動的生成を避ける
- OSコマンド・インジェクション
  - 外部コマンドの呼び出しを避ける
  - シェル経由で外部コマンドを呼び出さない
- HTTPヘッダインジェクション
  - HTTPヘッダ生成に高レベルのAPIを使う
  - 改行文字のチェックを行う
- メールヘッダインジェクション
  - メール生成にsendmailコマンドを呼ばず高水準のAPIを使う
  - 改行文字のチェックを行う

正しくないSQLインジェクション対策の今昔

正しくない例1(2001年)

一つの方法として，文字列中に「'」が登場した場合，「''」と重複してやることで，文字列を囲むシングル・クォーテーションではなく，文字としてのシングル・クォーテーションとして評価されるようにするというやり方がある。

【中略】

特にセキュリティを重視した場合においては，Webアプリケーションの仕様上許されるのであれば，「'」以外にも，半角の特殊文字列やSQL 文の予約文字列（SELECTやINSERTなど）を全角にしてからデータベースに格納するというようなチェック機構を作っておくことが望ましい。

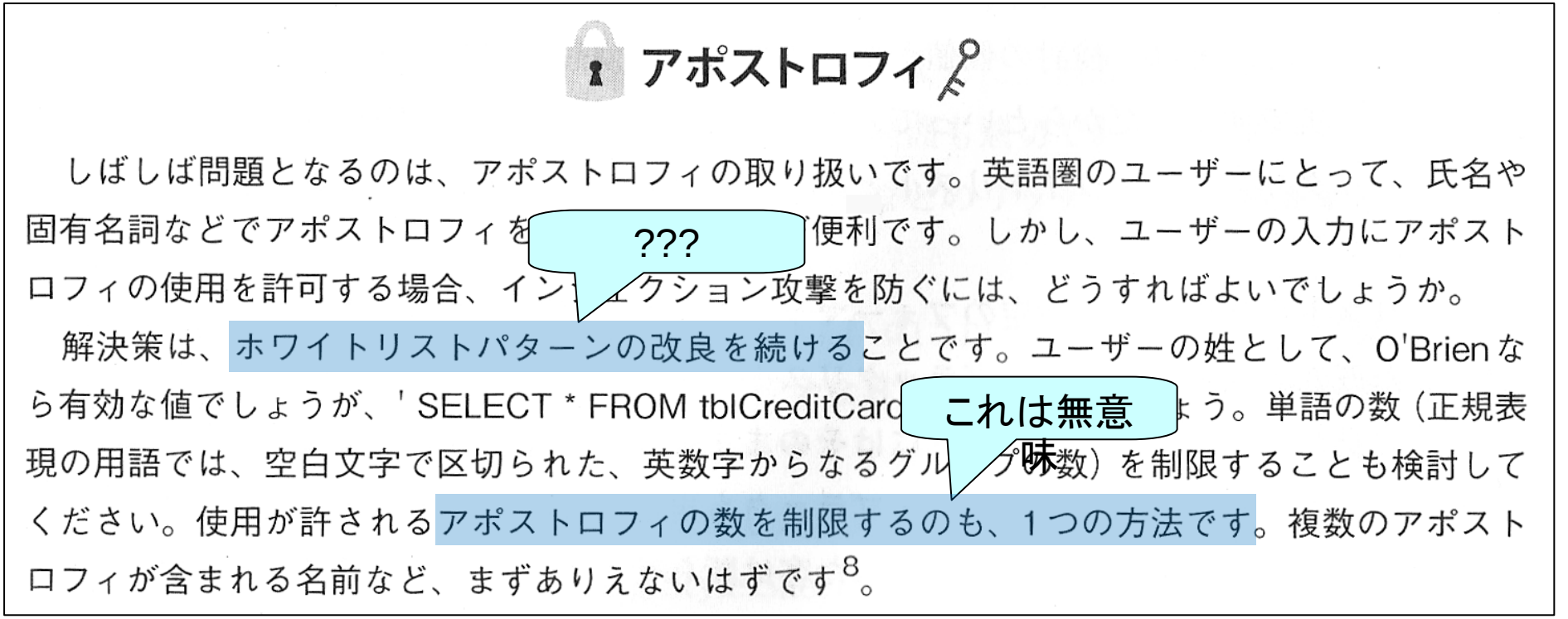
正しい

これは余計

でもこれは
2001年の記事だから…

実践! セキュアなWebプログラミング　日経オープンシステム2001年5月号 から引用

正しくない例2(2008年)



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[???]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[これは無意味]ょう。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

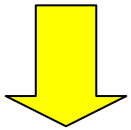
- このような方法では効果が薄いだけではなく、ケースバイケースで正しい方法を考えなければならない点が問題
- 脆弱性対策は、もっと機械的に適用できるものでないと実用的でない

正しくない例3(2008年)

【前提1】
・Webアプリケーションは文字列を入力として受理できる
・リレーショナル・データベース管理システムと連携

パーセントエンコードをデコードしてからでないと無意味

【入力の例】
DECLARE%20@S%20NVARCHAR(4000)SET%20@S=hogehoge EXEC(@S)



・文字列として入力
・特殊文字を文字として扱うために「¥」を挿入
・「;」は削除

むやみに「¥」を挿入しても…

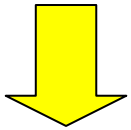
セミコロンを勝手に削除しないで!

【入力値チェックの結果】
DECLARE¥%20@S¥%20NVARCHAR¥(4000¥)SET¥%20@S¥=hogehoge EXEC¥(@S¥)

【前提2】
・SQLクエリーはアプリケーションで生成
・SQL構文に用いるような文字列はユーザーの入力としてはありえない

意味不明

【入力の例(入力値チェックの結果)】
DECLARE¥%20@S¥%20NVARCHAR¥(4000¥)SET¥%20@S¥=hogehoge EXEC¥(@S¥)



・SQL構文に用いられる代表的な文字列をフィルタリングして削除
・アットマーク(@)はデータベース上で変数の識別子やスクリプトの実行に用いられることがあるため削除

サニタイズ!!

【サニタイジングの例】
¥%20S¥%20¥(4000¥) ¥%20S¥=hogehoge ¥(S¥)

被害が続くSQLインジェクション攻撃，もう一度対策を見直そう より引用
http://itpro.nikkeibp.co.jp/article/COLUMN/20080514/301660/

なぜ誤った解説がなくならないのか

- 攻撃方法からの発想
  - 攻撃に使用する文字・文字列を削除・改変するアプローチ
  - いわゆる「サニタイズ」
  - 脆弱性が混入する根本原因からのアプローチではない
- 実はアプリケーションなんか書いた人が説明している?
  - セキュリティのプロが全員アプリケーションを書けるとは限らない
- そのサンプルコード、動かしてみた?
  - でもテスト環境構築するだけでも大変だしぃ
- コピペの悪弊
  - 昔の間違った解説が延命されられる

みんな攻撃が大好きだww

処理系のバグ（不適切な仕様）を避ける目的も…（過去の話題）

問題は、このプログラムではユーザ入力データをそのままSQL文の中に入れていることだ。ここで例えば下記のようなデータをフォームに入力したとしよう。

|shell("cmd /c echo aaa > c:¥test.txt")| (注: "|" は縦棒文字)

このプログラムは以下のSQL文をJetデータベースエンジンに投げる。

select * from Customers　where City='|shell("cmd /c echo aaa > c:¥test.txt")|‘

Jetは縦棒文字(|)で囲まれた部分をVBAスクリプトだと解釈するので、VBA Shell()関数を呼び出す。すなわち、サーバ上で cmd /c echo aaa > c:¥test.txt が実行されることになる。このサンプルプログラムでは単純に City=‘word’ という検索条件だが、City like ‘%word%’ というような検索条件であったとしても、入力文字列を巧妙に工夫することによりJetにShell関数を呼び出させることは可能である。

セキュリティ勧告 - NTサーバ上におけるJetセキュリティ問題
http://www.trusnet.com/advisories/jetshell/jetshell.txt より引用

- 昔のJetデータベースエンジンでは、文字列リテラル中のパイプ記号「|」にVBAスクリプトを呼び出す機能があった
- しかも、パイプ記号をエスケープする手段が提供されていなかった
- 不適切な仕様だが、現在では改修されている

【参考】なぜセミコロン「;」を削除したがるのか?

- 実はセミコロンの削除には実効的な意味はあまりない
- セミコロン削除の意図は、複文実行の防止と思われる
  - SELECT * FROM XXX WERE ID=''**;**UPDATE XXX SET …
- SQLインジェクションの文脈で複文が実行できるのは、MS SQLとPostgreSQL
- 現実にMS SQLは、複文を使った改ざん事件が多発
- しかし、MS SQLは、**セミコロンなしでも複文が書ける**
  - SELECT * FROM XXX WERE ID=''UPDATE XXX SET …　　でもよい
- すなわち、セミコロンの削除で保険的にせよ意味があるのは、PostgreSQLの場合だけ

続きはWebで http://www.tokumaru.org/d/20080502.html
http://www.tokumaru.org/d/20080627.html

SQLインジェクション対策の考え方

そもそもなぜSQLインジェクションが発生するのか?

- 原因は、リテラルとして指定したパラメータが、**リテラルの枠をはみ出し**、SQLの一部として解釈されること
- 文字列リテラルの場合
  - シングルクォートで囲まれた(クォートされた)範囲をはみ出す

    SELECT * FROM XXX WHERE A=''OR'A'=A'
- 数値リテラルの場合

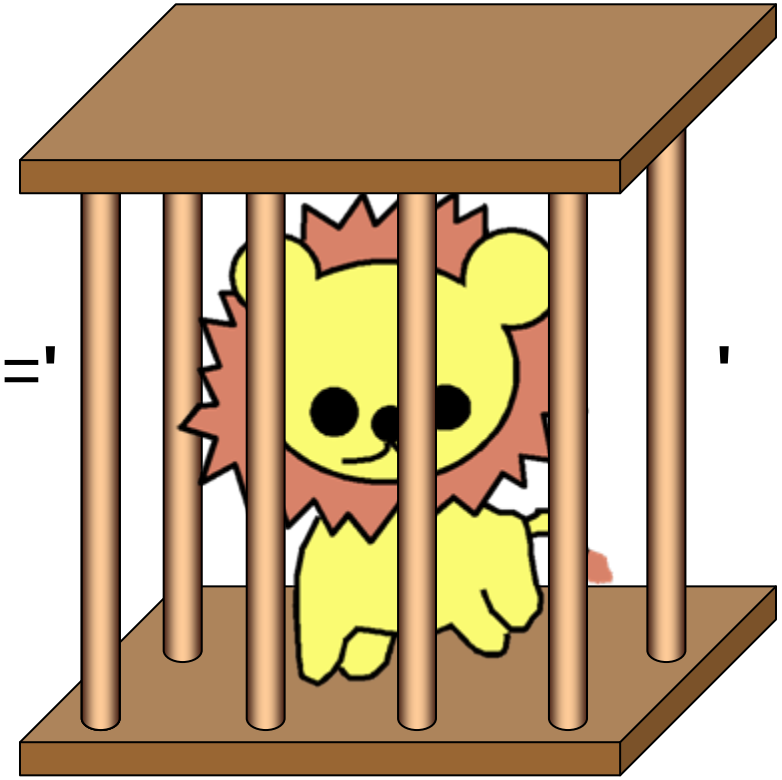
  はみ出した部分
  - 数値でない文字(空白、英字、記号など)を使う

    SELECT * FROM XXX WHERE A=1OR TRUE

エスケープは檻にしっかり入れるイメージ

- 檻に入っている分には、中身の「危険性」を気にする必要なはい
- 危険性がなくても、檻から出てしまうのはバグ

select * from animals whre kind='



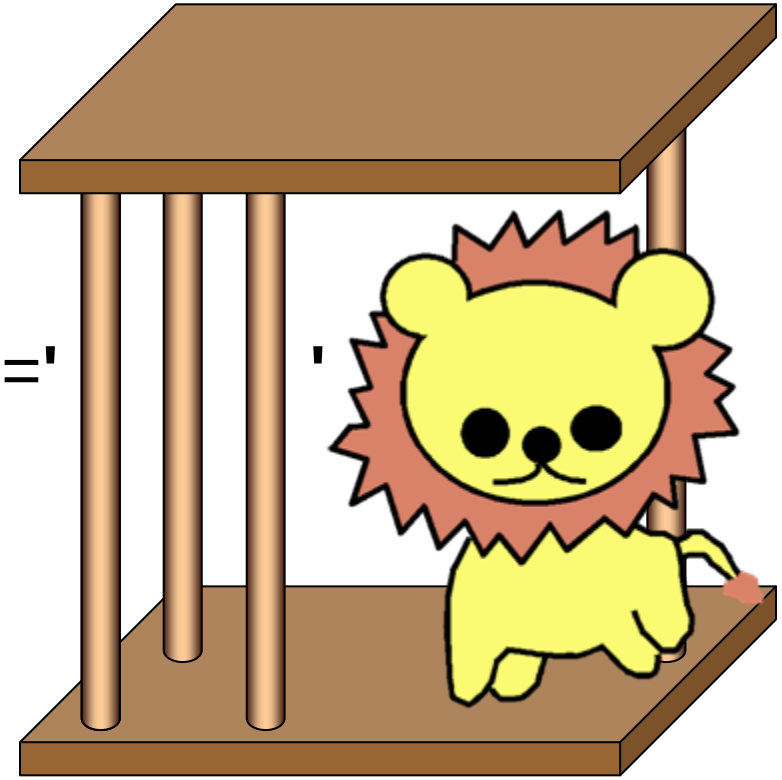
'

「危険な文字・文字列」; | ' @ declare union xp_cmdshell @ ...

SQLインジェクションは檻から逃げるイメージ

- パラメタがリテラルからはみ出し、SQL文の命令として解釈される状態

select * from animals whre kind='



'

「危険な文字・文字列」; | ' @ declare union xp_cmdshell @ ...

SQLインジェクション対策の実際

SQLの呼び出し方

SQLに動的な変数を埋め込む方法には2種類ある
(1)文字列連結によるSQL文組み立て

```
$name = ...;
$sql = "SELECT * FROM employee WHERE name='" . $name . "'";
```

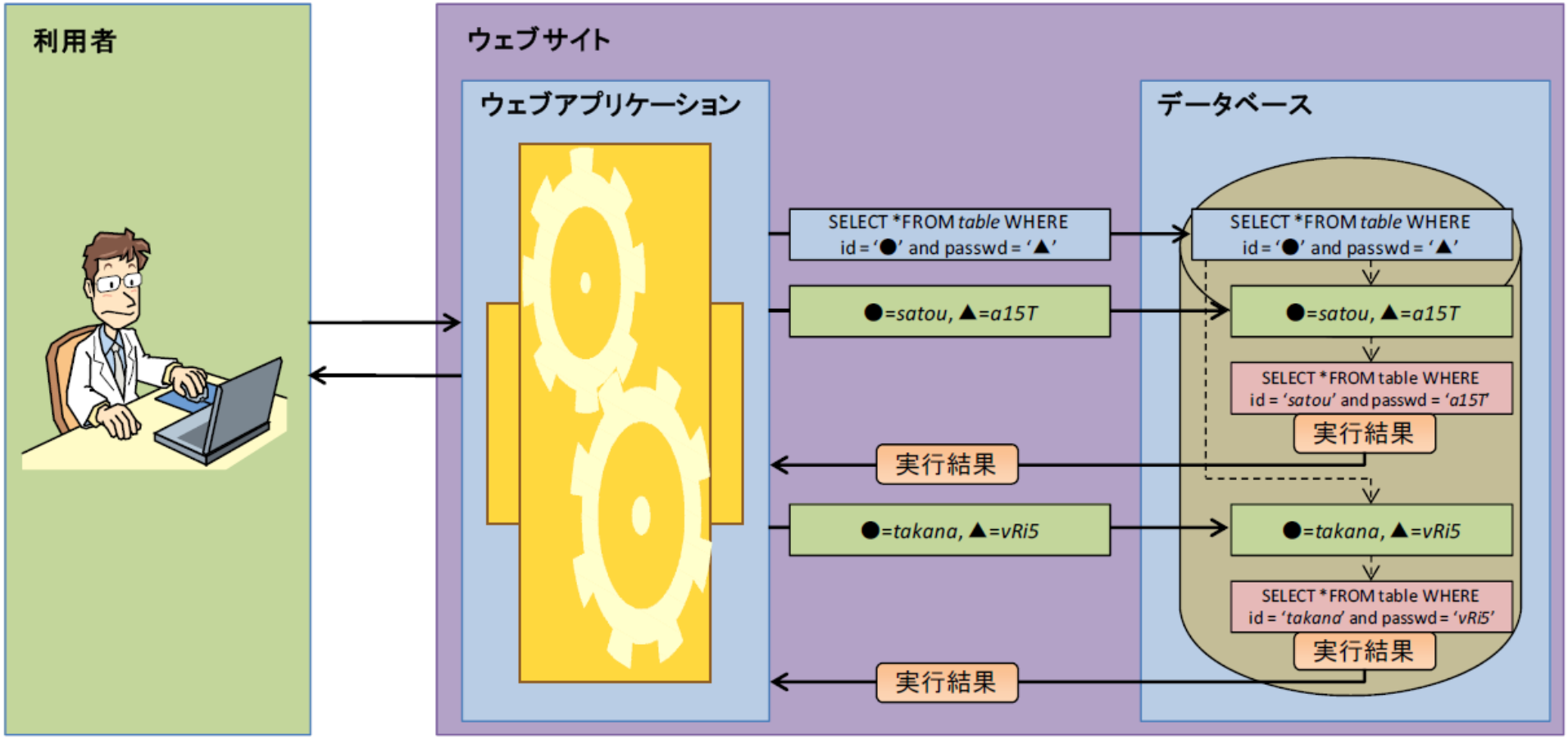
※ $nameをエスケープしていないのでSQLインジェクション脆弱性あり

(2)プレースホルダによるSQL文組み立て

```
PreparedStatement prep = onn.prepareStatement(
    "SELECT * FROM employee WHERE name=?");
prep.setString(1, “山田”);
```

※ プレースホルダには2種類ある（静的・動的）

静的プレースホルダ

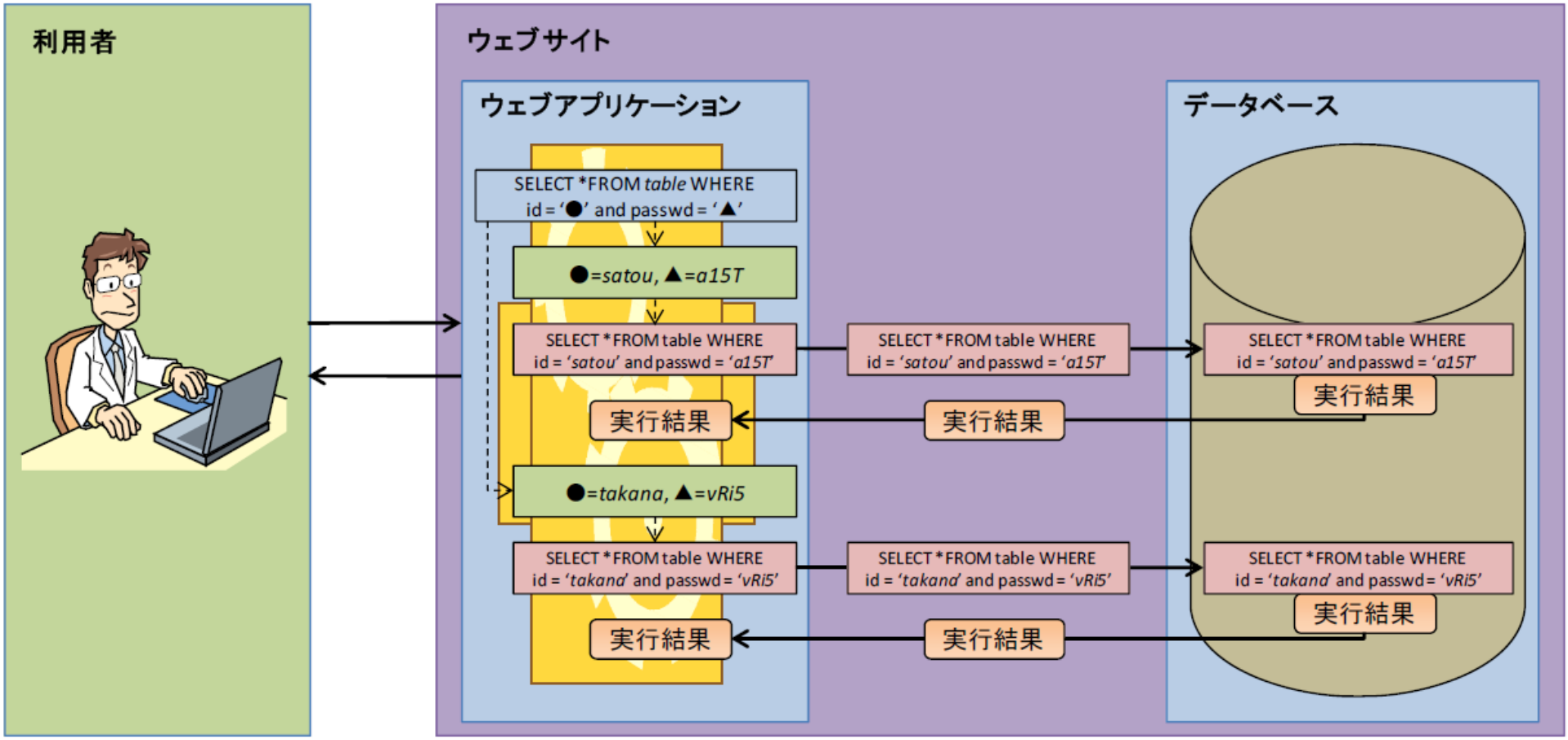


安全なSQLの呼び出し方より引用 http://www.ipa.go.jp/security/vuln/websecurity.html

- SQLとパラメータは別々にサーバーに送られる
- パラメータ抜きでSQLは構文解析される
- パラメータは後から割り当てられる

⇒ SQLインジェクション脆弱性の可能性が原理的になくなる

動的プレースホルダ

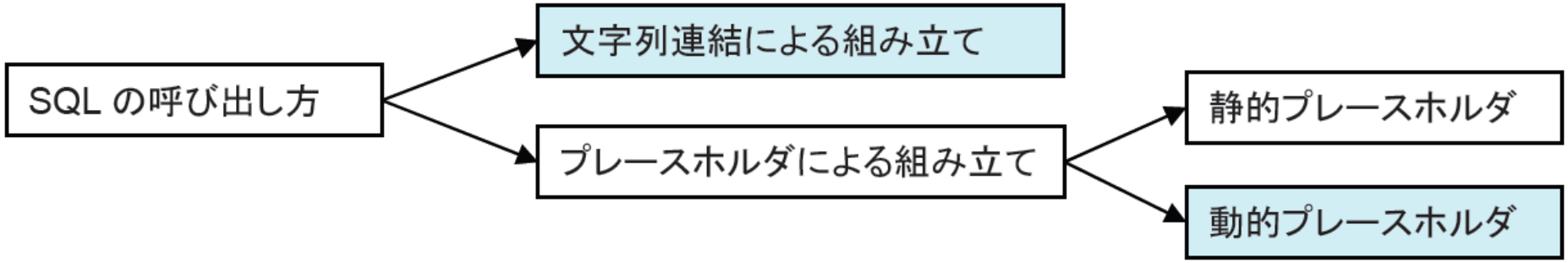


安全なSQLの呼び出し方より引用 http://www.ipa.go.jp/security/vuln/websecurity.html

- SQLとパラメータは呼び出し側で、エスケープ、連結される
- データベースサーバー側は、組み立てられた文字列をSQLとして実行するだけ

⇒ ライブラリにバグがなければ、SQLインジェクション脆弱性は発生しない

SQLの組み立て方とSQLインジェクションの可能性の関係



- 文字列連結による組み立ては、アプリケーション開発者の無知や不注意によりSQLインジェクション脆弱性の可能性がある
- 動的プレースホルダは、ライブラリのバグによりSQLインジェクション脆弱性の可能性がある（詳細はデモで）
- 静的プレースホルダは原理的にSQLインジェクション脆弱性の可能性がない

文字列連結によるSQL組み立てを安全に行うには

- 文字列連結によるSQL組み立て時のパラメータの要件
  - 文字列リテラルに対しては、エスケープすべき文字をエスケープすること
  - 数値リテラルに対しては、数値以外の文字を混入させないこと
- 意外に面倒
  - データベースによってエスケープすべきメタ文字が異なる
  - オプションによってもエスケープすべきメタ文字が異なる
- Perl、PHP等ではquoteメソッドが便利
  - Perl DBI
  - PHP　Pear::MDB2、PDO
- quoteメソッドはデータベースの種類や設定に応じたエスケープをしてくれる…はず
- 例外（バグ?、仕様?）もある

【参考】商用RDBMSの文字列リテラルの定義

- Oracle:cは、データベース・キャラクタ・セットの任意の要素です。リテラル内の一重引用符(‘)の前には、エスケープ文字を付ける必要があります。***リテラル内で一重引用符を表すには、一重引用符を2 つ使用します***。

  http://otndnld.oracle.co.jp/document/products/oracle10g/102/doc_cd/server.102/B19201-02/sql_elements.html#41297

- DB2:ストリング区切り文字で始まりストリング区切り文字で終わる文字のシーケンス。この場合のストリング区切り文字はアポストロフィ (') です。【中略】***文字ストリング内で 1 つのストリング区切り文字を表したいときは、ストリング区切り文字を 2 つ連続して使用します***。

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.sql.ref.doc/doc/r0000731.html

- MS SQL:***単一引用符で囲まれた文字列に単一引用符を埋め込む場合は、単一引用符を 2 つ続けて並べることで 1 つの単一引用符を表します***。
  http://technet.microsoft.com/ja-jp/library/ms179899.aspx

【参考】MySQLの文字列リテラルの定義

8.1.1. 文字列
一部のシーケンスは、個々の文字列内で特別な意味を持ちます。これらのシーケンスは、いずれも、エスケープ文字として知られるバックスラッシュ（‘¥’）で始まります。MySQLでは、次のエスケープシーケンスが認識されます。

¥0	ASCII 0 (NUL) 文字。
¥'	単一引用符 (“'”) 文字。
¥"	二重引用符(“"”) 文字。
¥b	バックスペース文字。
¥n	改行文字(LF)。
¥r	復帰改行文字。
¥t	タブ文字。
¥Z	ASCII 26 (Control-Z)。表の下部にある注釈を参照してください。
¥¥	バックスラッシュ(“¥”)文字。
¥%	“%”文字。表の下部にある注釈を参照してください。
¥_	“_”文字。表の下部にある注釈を参照してください。

¥' 単一引用符 (“'”) 文字。

文字列に引用符を含める方法は、いくつかあります。
- *‘’’で囲んだ文字列内で、‘’’を使用する場合、‘’‘’と記述することができます*。【後略】

「MySQL :: MySQL 5.1 リファレンスマニュアル :: 8.1.1 文字列」から引用
http://dev.mysql.com/doc/refman/5.1/ja/string-syntax.html

【参考】PostgreSQLの文字列リテラルの定義

4.1.2.1. 文字列定数

SQLにおける文字列定数は、単一引用符（‘）で括られた任意の文字の並びです。例えば、’This is a string‘です。***文字列定数内の単一引用符の記述方法は、2つ続けて単一引用符を記述することです***。

【中略】

エスケープ文字列の中では、バックスラッシュ文字（¥）によりC言語のようなバックスラッシュシーケンスが始まります。バックスラッシュと続く文字の組み合わせが特別なバイト値を表現します。¥bは後退（バックスペース）を、¥fは改頁を、¥nは改行を、¥rは復帰（キャリッジリターン）を、¥tはタブを表します。また、¥digitsという形式もサポートし、digitsは8進数バイト値を表します。¥xhexdigitsという形式では、hexdigitsは16進数バイト値を表します。（作成するバイトの並びがサーバの文字セット符号化方式として有効かどうかはコード作成者の責任です。）ここに示した以外のバックスラッシュの後の文字はそのまま解釈されます。***したがって、バックスラッシュ文字を含めるには、2つのバックスラッシュ（¥¥）を記述してください。また、通常の''という方法以外に、¥'と記述することで単一引用符をエスケープ文字列に含めることができます***。

http://www.postgresql.jp/document/current/html/sql-syntax-lexical.html#SQL-SYNTAX-CONSTANTS

文字列リテラルのエスケープ

- どの文字をエスケープするのか?
  - SQL製品の文字列リテラルのルールに従う
  - ISO標準では、「'」→「''」
  - MySQLとPostgreSQLは「'」→「''」「¥」→「¥¥」
    - NO_BACKSLASH_ESCAPES=onの場合は、ISO標準と同じ方法になる
  - PostgreSQLの場合は、standard_conforming_stringsおよびbackslash_quoteの影響を受ける
    - standard_conforming_strings=onの場合は、ISO標準と同じ方法になる
    - backslash_quoteの場合は、「'」→「¥'」というエスケープがエラーになる

	元の文字	エスケープ後
Oracle MS SQL IBM DB2	’	’’
MySQL PostgreSQL	’	’’ または ¥’ (’’を推奨)
	¥	¥¥

MySQLとPostgreSQLで「¥」のエスケープが必要な理由

SELECT * FROM XXX WHERE ID='$id'

$id として ¥'or 1=1# が入力されると

¥'or 1=1#
　↓ エスケープ(「¥」のエスケープをしない場合)
¥''or 1=1#

元のSQLに適用すると、

SELECT * FROM XXX WHERE ID='¥'' or 1=1#'

すなわち、SQLの構文が改変された(「¥’」で「’」一文字と見なされる)

数値型パラメータの対処

- 一部で、数値パラメータについても、エスケープしてクォートする（引用符で囲む）ことを推奨しているが…

  例: select * from employee wehre age > '27'

- SQLは「型付けの強い言語」であり、数値をクォートすると副作用が大きい
  - 文字列から数値への「暗黙の型変換」が発生
  - 文字列から数値の変換は処理系依存であり、予期しない結果を生む
- 暗黙の型変換の奇妙な結果の例(MySQL)

```
create table dtest (d0 decimal(20, 0));
insert into dtest values(12345678901234567890);
insert into dtest values(12345678901234570000);
select * from dtest where d0 = '12345678901234567890';
 +----------------------+
 | d0                   |
 +----------------------+
 | 12345678901234570000 |
 +----------------------+
```

数値型パラメータの対処（続き）

- なぜ、奇妙な結果になるか?

  select * from dtest where d0 = '123456789012345678901';

- 数値例d0と文字列リテラル'12345678901234567890'の比較に際して、文字列→浮動小数点型への変換が発生する

> 次のルールは、比較の演算に対してどのように変換が行われるかを示しています：
> * 【中略】
> * 他のすべてのケースでは、引数は浮動少数点（実）数として比較されます。
>
> *「MySQL 5.1 リファレンスマニュアル :: 11 関数と演算子 :: 11.1 演算子 :: 11.1.2 式評価でのタイプ変換」より引用*
> *http://dev.mysql.com/doc/refman/5.1/ja/type-conversion.html*

```
mysql> select '123456789012345678901'+0;
 +---------------------------+
 | '123456789012345678901'+0 |
 +---------------------------+
 |    1.2345678901234e+020 |
 +---------------------------+
```

⇐ 浮動小数点数に変換されている

- 数値は数値のまま扱うこと

quoteメソッドの詳細

PHPのPear::MDB2におけるquoteの呼び出し

```
require_once ‘MDB2.php’;  //ライブラリのロード
// DBへの接続（PostgreSQLの場合）
$db = MDB2::connect('pgsql://dbuser:password@hostname/dbname?charset=utf8');

// 文字列型を指定して、文字列リテラルのクォート済み文字列を得る
（略）$db->quote($s, 'text')（略）

// 数値型を指定して、数値リテラルの文字列を得る
（略）$db->quote($n, 'decimal')（略）
```

データ	型指定	戻り値
abc	’text’	’abc’（PHPの文字列型の値、クォートを含む）
O’Reilly	’text’	O’’Reilly’（PHPの文字列型の値、クォートを含む）
−123	’decimal’	−123（PHPの文字列型の値）
123abc	’decimal’	123（PHPの文字列型の値）
−123	’integer’	−123（PHPの整数型の値）
123abc	’integer’	123（PHPの整数型の値）

quoteメソッドの数値データの処理結果

Perl DBI/DBD、PHPのPDO, Pear::MDB2でquoteの処理内容を調査

1a¥’ をINTEGER型を指定してquoteすると、どうなるか?

サンプルスクリプト:

```
DBI:  $dbh->quote("1a¥¥'", SQL_INTEGER)
PDO:  $dbh->quote("1a¥¥'", PDO::PARAM_INT)
MDB2: $dbh->quote("1a¥¥'", 'integer')
```

モジュール名	結果	
DBI(DBD::mysql)	1a¥’	⇦ なにもしていない！（脆弱性）
DBI(DBD::PgPP)	’1a¥¥¥’’	
PDO	’1a¥¥¥’’	
MDB2	1 (int型)	⇦ 正しい結果

- quoteメソッドに期待したが、現状SQLの仕様通り動作するのはMDB2のみ
- プレースホルダの利用を推奨

【文字コードの問題1】5C問題によるSQLインジェクション

- 5C問題とは
  - Shift_JIS文字の2バイト目に0x5Cが来る文字に起因する問題
    ソ、表、能、欺、申、暴、十 … など出現頻度の高い文字が多い
  - 0x5CがASCIIではバックスラッシュであり、ISO-8859-1など1バイト文字と解釈された場合、日本語の1バイトがバックスラッシュとして取り扱われる
  - 一貫して1バイト文字として取り扱われれば脆弱性にならないが、1バイト文字として取り扱われる場合と、Shift_JISとして取り扱われる場合が混在すると脆弱性が発生する

ソースコード（要点のみ）

```
<?php
  header('Content-Type: text/html; charset=Shift_JIS');
  $key = @$_GET['name'];
  if (! mb_check_encoding($key, 'Shift_JIS')) {
    die('文字エンコーディングが不正です');
  }
  // MySQLに接続(PDO)
  $dbh = new PDO('mysql:host=localhost;dbname=books', 'phpcon', 'pass1');
  // Shift_JISを指定
  $dbh->query("SET NAMES sjis");
  // プレースホルダによるSQLインジェクション対策
  $sth = $dbh->prepare("SELECT * FROM books WHERE author=?");
  $sth->setFetchMode(PDO::FETCH_NUM);
  // バインドとクエリ実行
  $sth->execute(array($key));
?>
```

5C問題によるSQLインジェクションの説明

対策

- 文字エンコーディング指定のできるデータベース接続ライブラリを選定し、文字エンコーディングを正しく指定する

```
$dbh = new PDO('mysql:host=xxxx;dbname=xxxx;charset=cp932',
                    'user', 'pass', array(
  PDO::MYSQL_ATTR_READ_DEFAULT_FILE => '/etc/mysql/my.cnf',
    PDO::MYSQL_ATTR_READ_DEFAULT_GROUP => 'client', ));
  # http://gist.github.com/459499 より引用(by id:nihen)
```

- 静的プレースホルダを使うよう指定する、あるいはプログラミングする

```
$dbh->setAttribute(PDO::ATTR_EMULATE_PREPARES,
                                  false);
```

- 詳しくは「安全なSQLの呼び出し方」を参照
  http://www.ipa.go.jp/security/vuln/websecurity.html

【文字コードの問題2】U+00A5によるSQLインジェクション

```
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection(
  "jdbc:mysql://localhost/books?user=phpcon&passw
  ord=pass1");
String sql = "SELECT * FROM books where
  author=?";
// プレースホルダ利用によるSQLインジェクション対策
PreparedStatement stmt =
  con.prepareStatement(sql);
// ? の場所に値を埋め込む（バインド）
stmt.setString(1, key);
ResultSet rs = stmt.executeQuery();  // クエリの実行
```

U+00A5によるSQLインジェクションの原理

【入力文字列（Unicode）】

コードポイント	00A5	0027	006F	0072	0020	0031	003D	0031	0023
文字	¥	’	o	r	SP	1	=	1	#

【エスケープ処理後の文字列（Unicode）】

コードポイント	00A5	005C	0027	006F	0072	0020	0031	003D	0031	0023
文字	¥	\	’	o	r	SP	1	=	1	#

【Shift_JIS に変換した文字列】

文字コード	5C	5C	27	6F	72	20	31	3D	31	23
文字	¥	¥	’	o	r	SP	1	=	1	#

【動的プレースホルダにバインドした SQL 文】

```
SELECT * FROM test WHERE name='¥¥' or 1=1#'
```

SQL

IPA:安全なSQLの呼び出し方（http://www.ipa.go.jp/security/vuln/websecurity.html）より引用

U+00A5によるSQLインジェクションの条件と対策

- 脆弱性が発生する条件
  - JDBCとしてMySQL Connector/J 5.1.7以前を使用
  - MySQLとの接続にShift_JISあるいはEUC-JPを使用
  - 静的プレースホルダを使わず、エスケープあるいは動的プレースホルダ（クライアントサイドのバインド機構）を利用している
- 対策（どれか一つで対策になるがすべて実施を推奨）
  - MySQL Connector/Jの最新版を利用する
  - MySQLとの接続に使用する文字エンコーディングとしてUnicode(UTF-8)を指定する
    （接続文字列にcharacterEncoding=utf8を指定する）
  - 静的プレースホルダを使用する
    （接続文字列にuseServerPrepStmts=trueを指定する）

簡単にできるテスト

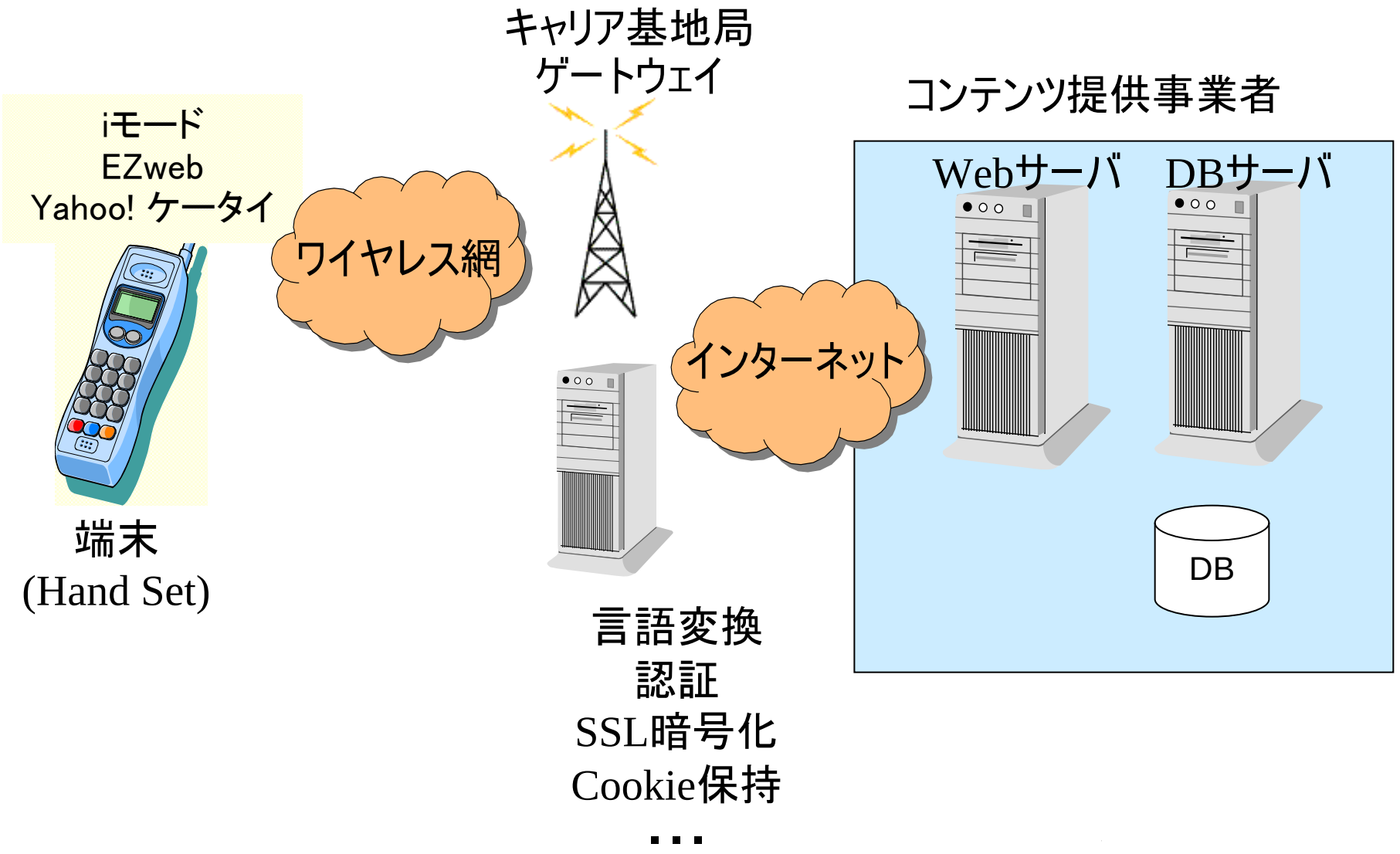
- 以下の文字を入力・登録して、どのように表示されるかを調べる
  - ￥ (U+00A5)　　バックスラッシュに変換されないか
  - 髃 (U+9AB6)　　JIS X 0208にない文字
  - 𠮷 (U+20BB7)　　BMP外の文字 UTF-8では4バイトになる
- 尾骶骨テストや「つちよし」テストで、Unicodeがきちんと通るか確認しよう

SQLインジェクション対策

- 入力値
  - 文字エンコーディングの検証 and/or 文字エンコーディングの変換
  - 要件に従った入力値の検証（制御文字のチェックは必須）
- SQL呼び出し
  - ともかくプレースホルダを使うこと
  - 静的プレースホルダの利用が「原理的に」安全
  - 「安全なSQLの呼び出し方」をよく読む
- 文字コードの選定
  - アプリケーションを通してUnicodeを使う
    - HTTPメッセージはUTF-8
    - アプリケーションの内部はUTF-8かUTF-16
  - ケータイ向けサイトはHTTPメッセージの文字エンコーディングをShift_JISにするが、内部はUTF-8とする
    - EUC-JPという選択もあり得るが、使える言語が少ない
    - 円記号U-00A5 → バックスラッシュ(5C) の変換に注意
  - 尾骶骨テストのすすめ

ケータイサイトのセキュリティ

ケータイWebアプリの構成



ケータイWebアプリの特徴

- 基本的には、PCサイトと同じようなWebアプリケーションである
  - HTMLまたは類する言語で記述され、HTTPを通じてインターネット・アクセスされる.
  - 昔：CompactHTML、HDML、MML　→　今：HTML/XHTML
- PCブラウザとの違い
  - 大半の端末でJavaScriptが使えない
  - Cookieが使えない・・・ことがある
  - キャリアごとに、端末固有ID(uid、EZ番号)がある
  - HTMLソースが読めない
- キャリアのゲートウェイ(Proxy)経由でアクセスされる
  - キャリアのゲートウェイ経由でアクセスされる
  - IPアドレスはゲートウェイが持つ
  - ゲートウェイには、コンテンツ変換機能、認証・課金機能がある
- SSLが利用できる(最初期以外)
  - しかし、端末の「世代」により動作が少し異なる

ケータイWebアプリの脆弱性とは

Cookieが使えない・・・ことがある

- iモードでは伝統的にCookieが使えなかった
- EZweb、Yahoo! ケータイではCookieが利用できる(ものもある)が
  - RFCに忠実に実装されているわけではない
  - Expiresの解釈・実装がRFC(PCブラウザ)とは異なる場合がある
  - Yahoo! ケータイのガイドラインには、「一部の3GC型端末では、期限が指定されていないCookieを一時型として扱わないので注意すること」とある!?
- 結局、ケータイコンテンツを作成する際には、Cookieを使わないで実装することが多い
  - URLにセッションIDを埋め込む
  - 端末固有IDをセッション識別に利用する
  - 着メロなどでは、セッション管理機能を利用しないで全部引き回す場合も・・・
- Cookieを使わないで、ケータイ固有のセッション管理手法を用いることから、脆弱性が生まれる

【参考】EzwebのCookieの挙動

EZweb対応端末においてCookieは、EZサーバに保管されます。

※ただし、WAP2.0ブラウザ搭載端末ではEnd to EndのSSL通信時は端末に保管されます。
なお、EZサーバに保管されたCookieはKDDI設備のメンテナンスなどによりリセットされる場合があります。

Cookieの保存領域とサイズ制限

利用可能なCookieのサイズについては、保存領域に応じて以下のように制限があります。

	EZサーバにて保存・管理される場合	端末にて保存・管理される場合 (WAP2.0ブラウザ搭載端末のみ)
1Cookieあたりのサイズ	次の「長さ」までの動作を保証します。 ● Nameの長さ=2000バイト ● valueの長さ=4096バイト ● Pathの長さ=256バイト ● Domainの長さ=256バイト	「Set-Cookie」フィールドに埋め込まれる文字列サイズが「1024バイト」以内の動作を保証します。
1ユーザあたりのサイズ	1ユーザ(1サブスクライバID)あたりの保証値は「30件」です。	1ユーザ(1端末)あたりの最低保存領域は「4096バイト」です。

•SSL時には端末、非SSL時にはゲートウェイにCookie格納と明記されている
•SSLと非SSL共存のサイトは特に注意

http://www.au.kddi.com/ezfactory/tec/spec/cookie.html から引用

【参考】ソフトバンクのSSL挙動

SSL(Secure Sockets Layer)、TLS(Transport Layer Security)は、サーバに対してのなりすましや、盗聴、データの改竄の防止に利用されている暗号化プロトコルです。SSL/TLSに対応しているソフトバンク携帯電話とWebサーバ間をより安全にデータ通信することができます。

ソフトバンク3G携帯電話では、「端末⇔弊社GW(以降、GW)」および「GW⇔Webサーバ」の通信区間でそれぞれSSL/TLSのセッションを確立し、GWは「端末⇔Webサーバ」間のSSL/TLSセッションの中継を行います。GWでは、Webサーバのレスポンスに含まれる、XHTMLドキュメント等に記載されたWebサーバへのhttpsリンクを、GWへのリンクに変換します。端末から見た場合、リクエストはGWへのSSLセッションとなります。

Ex: https://www.foo.com/bar.html というURIはGWにて
https://secure.softbank.ne.jp/www.foo.com/bar.html と変換されます。

メール本文等に記載されたhttps://を含むURIへアクセスした場合、GWはSSL/TLSセッションの中継を行わず、End to Endでセッションを確立します。

•ソフトバンクのSSLは、原則としてゲートウェイ経由になる
SSLの場合でも、端末固有ID付与や絵文字変換を行っている
•しかし、リンクを経由せずダイレクトに接続した場合は、End to EndのSSLになる
•このため、SSLの場合と、非SSLの場合にドメインが異なることになり、セッションIDが共有できないという問題
•secure.softbank.ne.jp経由のSSLは廃止の方向とのこと
産総研の高木浩光氏とソフトバンク宮川潤一CTOがtwitter上で会話され、廃止の方向性が決定される

http://creation.mb.softbank.jp/web/web_ssl.htmlから引用

宮川様、ケータイWeb開発に携わる皆が困っております。要点としては、secure.softbank.ne.jpを通さないhttps:リンクの手段をまずは用意して頂きたいということに尽きます。QT @co3k .@miyakawa11 御社携帯端末でHTTPS通信の挙動が…

HiromitsuTakagi
2010-06-15 23:56:06

@HiromitsuTakagi 高木先生、皆様に御迷惑をお掛け致しました。至急にて確認対応致します。一旦、預からせて下さい。明日中には回答致します。

miyakawa11
2010-06-16 00:06:43

@miyakawa11 感謝いたします。こうした仕組みはWebの根幹を成すものであり、Web開発者ら皆が当事者なのですから、本来はキャリアが一方的に決めるのでなく、皆がオープンに議論可能な場で決めていくべき仕様であると思います。そうでなければ必ずいつか首を絞めることになります。

HiromitsuTakagi
2010-06-16 00:16:07

@miyakawa11 私ももっと早くこれに気づくべきでしたが、一昨年この仕組みの存在に気づいたときは何もしませんでした。そのときにもっと調べるべきだったと悔やんでいます。正直当時は「携帯のことは業界で勝手にやってれば？どうせNDAだし議論にも出て来ないし」と思っていたのでした。

HiromitsuTakagi
2010-06-16 00:22:47

@miyakawa11 しかし今や日本のケータイWebは、Web技術者らの感覚を変え、通常のインターネットのやり方さえ変えてしまいかねない勢いです。ケータイWebの技術方式が健全にならないと、この数年で大変まずいことになるのではないかと危惧しております。

HiromitsuTakagi
2010-06-16 00:28:20

@HiromitsuTakagi 危惧に対する是正、キャリアとして取り組まねばと痛感致しました。コメント大事に拝見しております。いつも有難うございます。

miyakawa11
2010-06-16 00:37:02

@miyakawa11 ご理解いただけているようで嬉しいです。今回の件もそうですが、御社の今後の行動がケータイWEBの正常化につながっていくことを期待します

co3k
2010-06-16 01:21:56

@HiromitsuTakagi 社内調整中ですが、secure..jp経由とせず、ダイレクトにSSLを行う事に変更し、暫定にて公式サイト使用を許可したCP様を対象に、ホワイトリスト化したサイトのみsecure..jpを経由する方向で検討していますが、アドバイス戴けますか？

miyakawa11
2010-06-16 01:36:55

SoftBank

携帯サイト閲覧時の重要なお知らせ

掲載日 2010年10月15日

2011年2月に予定しているネットワークの仕様変更に伴い一部のコンテンツご利用においてお客様のご対応が必要な場合があります。

平素はソフトバンクをご利用いただき、誠にありがとうございます。
この度弊社では、サイト開発の利便性向上などを目的とした、暗号化通信※に関する仕様変更を下記内容にて予定しております。

※暗号化通信とは？
インターネット上の情報を暗号化して送受信する通信方式です。
通信先のアドレスが「https://～」と表示されているサイトは暗号化通信を使用しています。
また弊社携帯電話では暗号化通信を開始する際、右記のような画面が表示されます。

スケジュール

2011年2月1日から仕様を変更いたします。

対象機種

Yahoo!ケータイ対応機種。

仕様変更の概要

暗号化通信の際に弊社ネットワーク機器にて一旦通信を中継しておりましたが、中継処理をしない方式に変更いたします。本ネットワークの仕様変更に伴いまして、インターネットをご利用いただく際に、影響が出る場合がありますので、以下のご対応をお願いいたします。

http://mb.softbank.jp/mb/information/details/101015.html　より引用

基本的には、PCサイトと同じようなWebアプリケーションである

- 一般的なWebアプリケーション脆弱性パターンは、ケータイWebアプリケーションでも同じように成立する
  - SQL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
  - HTTPヘッダ・インジェクション
  - メールヘッダ・インジェクション
  - ・・・
  - 他者権限の利用
  - CSRF
  - セッションフィクセーション

大半の端末でJavaScriptが使えない

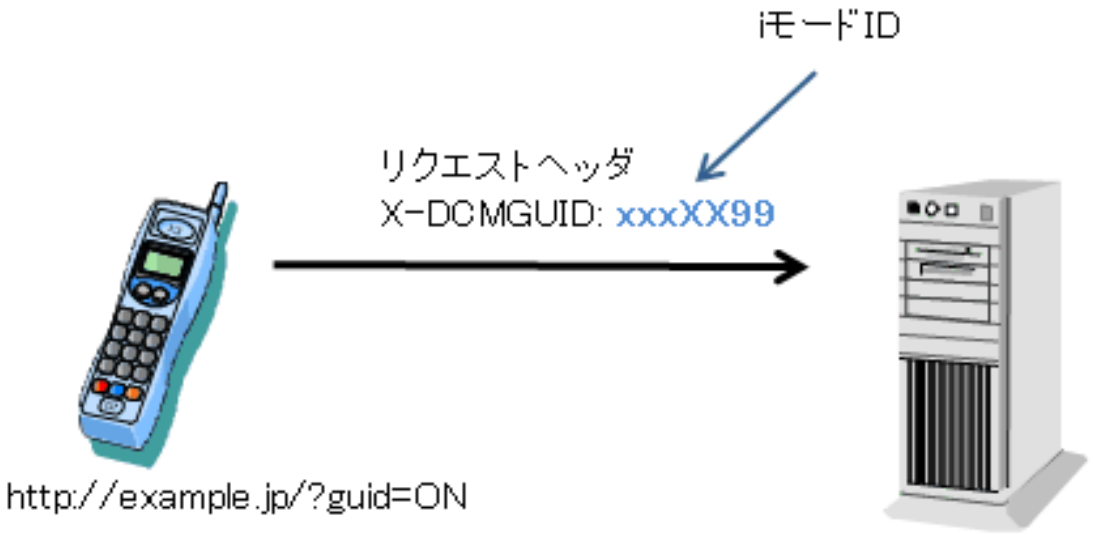
- ケータイブラウザではJavaScriptがサポートされていない
  - ドコモの2009年夏モデルからはJavaScriptサポート
  - ソフトバンクの2010年夏モデルから一部でJavaScript正式サポート
  - 狭義のCross-Site Scripting(XSS)攻撃は成立しなかったことになるが・・・
- XSSを広義(タグのエスケープ漏れ)にとらえると、ケータイブラウザ上でも「悪いこと」できる可能性がある
  - 画面の改ざん
  - フィッシングへの悪用
  - その他の特殊なタグ・・・
- マイナーなタグの挙動は、キャリア、世代、機種によって微妙に異なる
  - ある端末で「悪いこと」が起こらなくても、安心はできない
- 今後ドコモ、ソフトバンクの新機種はJavaScript対応になるので、JavaScriptを前提にしたセキュリティ施策が必要

キャリアのゲートウェイ(Proxy)経由でアクセスされる

- 一般に、ケータイWebアプリは、「ゲートウェイがあるから安全」と言われることが多いが・・・
- 現実には、ゲートウェイとWebサーバー間はインターネットで接続されるので、必ずしも安全ではない
- 安全な例(PCからアクセスできない)
  - キャリアとWebサーバー間を専用線で接続する(最近はあまり聞かない)
  - ファイヤーウォールなどで、ゲートウェイ以外からのアクセスを拒絶する
- 安全でない例(PCからアクセスできる)
  - アクセス制限をしていない
  - User-Agentによりアクセス制限をしている・・・PCで簡単に偽装できる
- いずれにせよ、ケータイ実機を使った不正アクセスに対しては、ゲートウェイは無力
- Wi-Fi経由の利用が多いスマートフォンが普及すると、今後は、IPアドレス制限が掛けにくくなる可能性

ケータイ固有の脆弱性問題

「かんたんログイン」とは

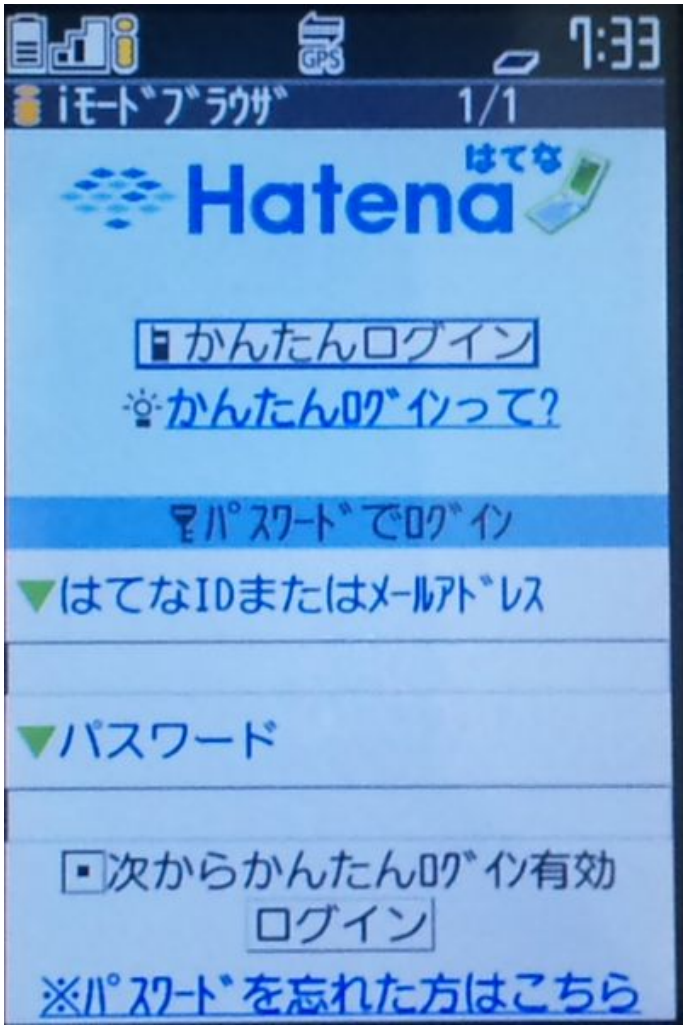


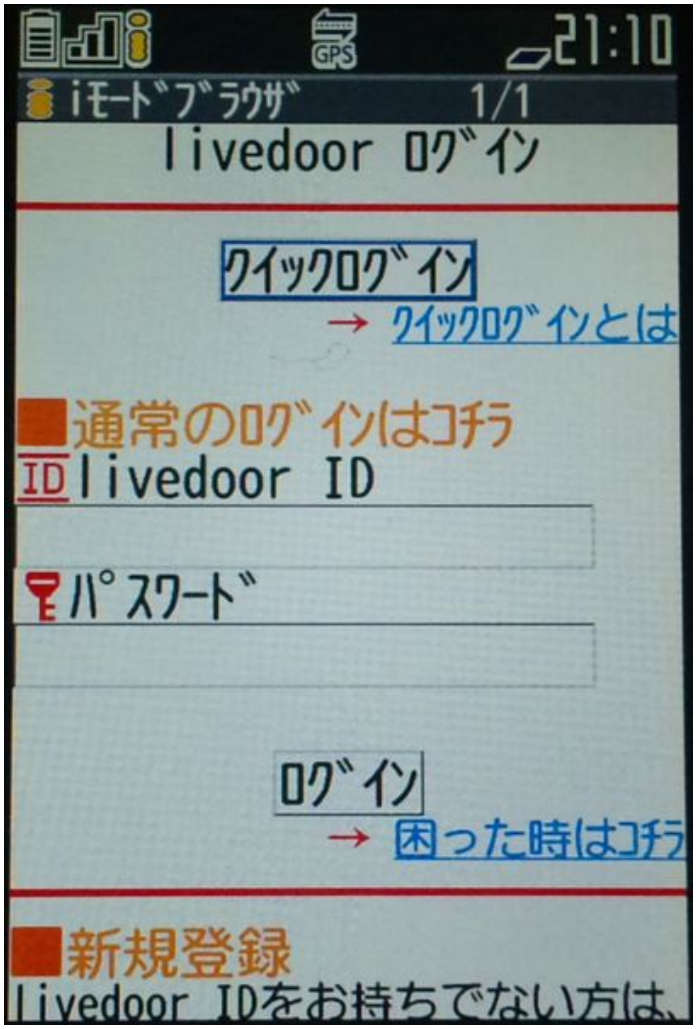
端末固有IDのみを
キーとして認証する

端末固有IDあれこれ

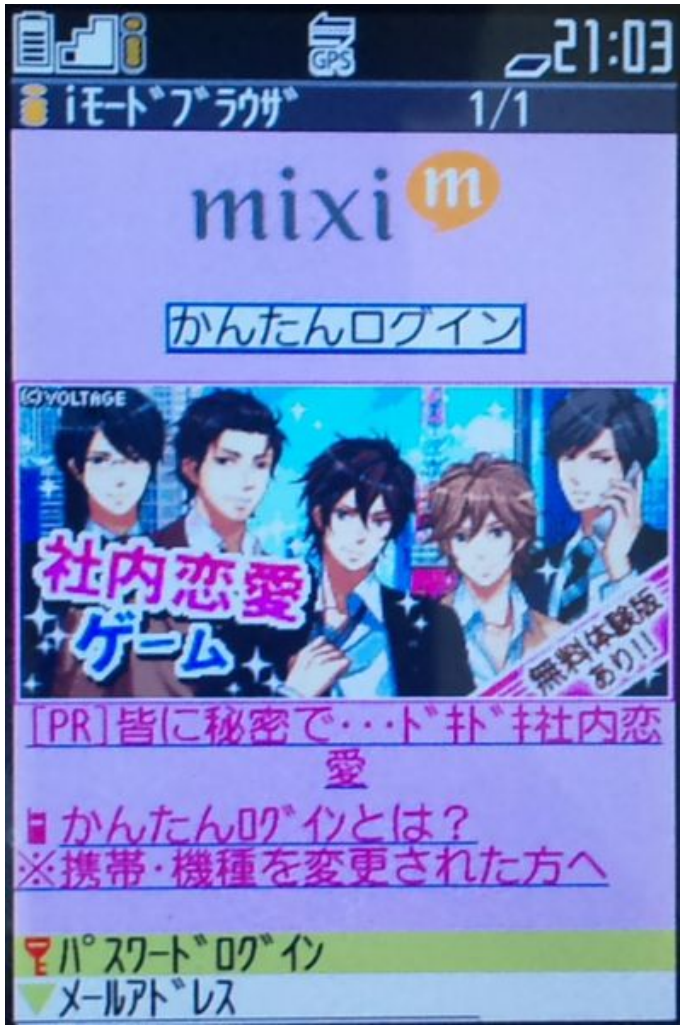
キャリア	名称	HTTPヘッダ
NTTドコモ	FOMA端末識別番号	User-Agent
	iモードID	X-DCMGUID
Au	EZ番号	X-UP-SUBNO
ソフトバンク	端末シリアル番号	User-Agent
	ユーザーID	X-JPHONE-UID

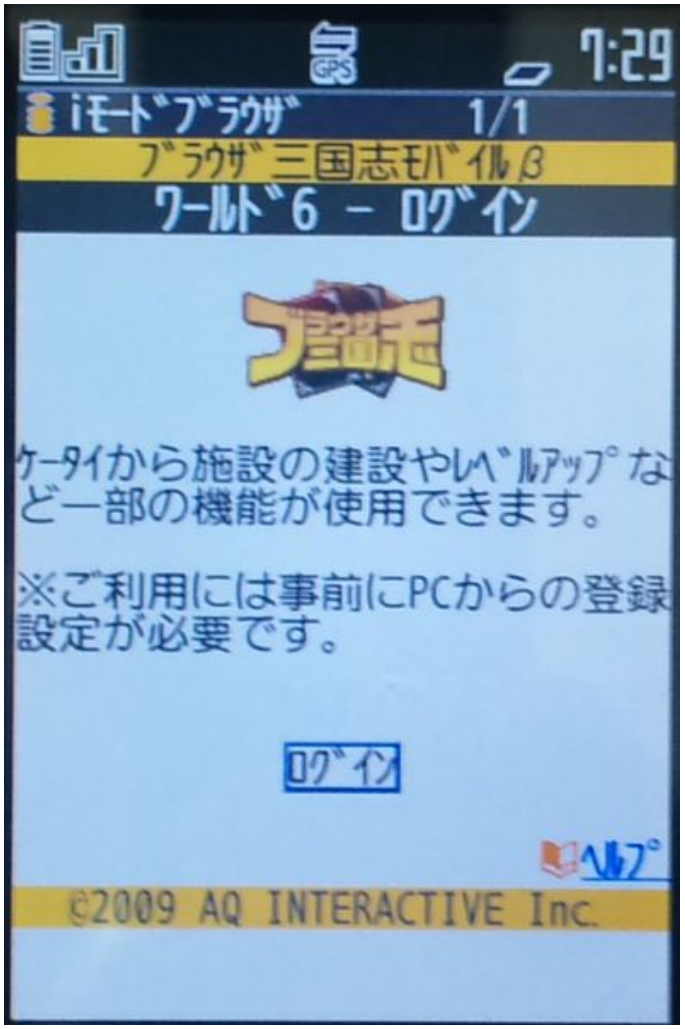
かんたんログイン画面の例(1)



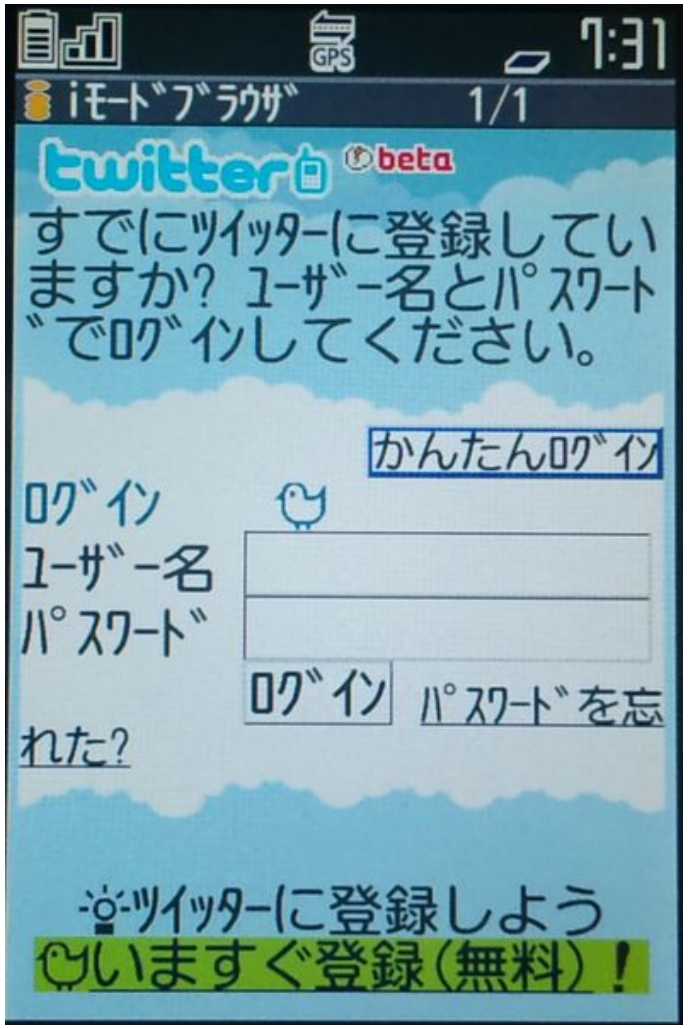


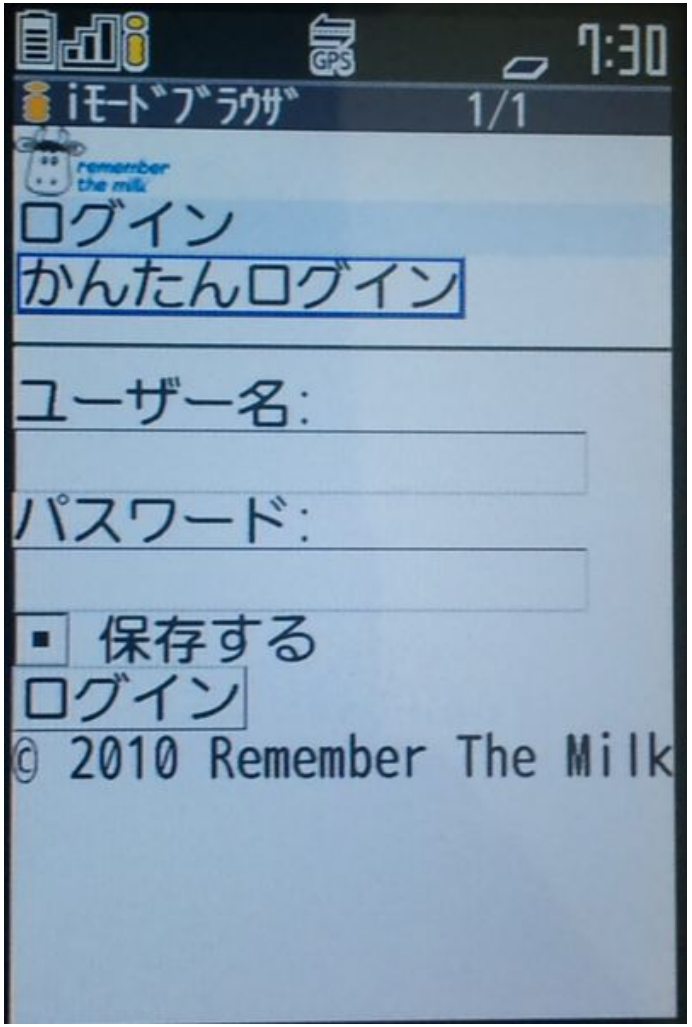
かんたんログイン画面の例(2)





かんたんログイン画面の例(3)





「かんたんログイン」がなりたつための条件

- 端末固有IDは、同一端末であれば、すべてのサイトに同じ値が送出される。すなわち、端末固有IDは秘密情報ではない
- 秘密情報でない、固定のIDで認証するためには、端末固有IDは書き換えができないという前提が必要
- 端末固有IDはHTTPヘッダに乗ってくる値なので、通常は任意に書き換えができる。携帯電話を使っている時は変更できないと考えられているが…

- まとめると、かんたんログインはケータイの以下の条件によって支えられている
  - ケータイWebが閉じたネットワークで利用される
  - ケータイ端末の機能が低く、HTTPヘッダを書き換える機能がない

かんたんログイン脆弱性報告の例

高木浩光@自宅の日記

2010年02月21日 63 users 38 tweets

■ はてなのかんたんログインがオッピロゲだった件 B! 486 users

- かんたんログイン手法の脆弱性に対する責任は誰にあるのか, 徳丸浩の日記, 2010年2月12日

といった問題が指摘されているところだが、それ以前の話があるので書いておかねばならない。

昨年夏の話。

2009年8月初め、はてなブックマーク界隈では、ケータイサイトの「iモードID」などの契約者固有IDを用いた、いわゆる「かんたんログイン」機能の実装が危ういという話題で持ち切りだった。かんたんログイン実装のために必須の、IPアドレス制限*1を突破できてしまうのではないかという話だ。

まず最初に調べたのは「ポケットはてな」だった。最も身近なサイトであるのと同時に、「はてならやりかねない」と、そう思ったからだ。

半信半疑ながら、検査手順を整理して改めてテストしてみると、やっぱりログイン状態になる。つまり、なんと、ポケットはてなは、はじめっからIPアドレス制限なんぞ、まったくしていなかったのだ。

これにはぶったまげた。いくらなんでもそれはないわ。あれだけ度々、はてなブックマークのセキュリティタグ界隈で、契約者固有IDによる「かんたんログイン」の危うさが話題にのぼっていたのに、はてなの人らは、見てないのか？

かんたんログイン脆弱性および事故の例

ヤマト運輸トップ ＞ 携帯版「クロネコメンバーズのWebサービス」クイックログイン機能の脆弱性への対応について

平成22年10月25日

お客様各位

携帯版「クロネコメンバーズのWebサービス」
クイックログイン機能の脆弱性への対応について

いつも「クロネコメンバーズのWebサービス」をご利用いただきありがとうございます。
このたび、携帯版「クロネコメンバーズのWebサービス」に脆弱性が見つかり、下記の通り対応いたしましたのでお知らせいたします。メンバーズの皆様にはご心配をお掛けしましたことを深くお詫び申し上げます。

記

1．脆弱性の内容
携帯版「クロネコメンバーズのWebサービス」のクイックログイン機能をご利用の場合、一部のスマートフォンから特定のアプリケーションを利用し、特定の操作を行うことで、ご本人とは別の方がログインできてしまうことが判明いたしました。

2．影響範囲
過去のデータを確認することで上記アプリケーションの影響調査は完了しており、1名のお客様が、2名のお客様からログインされたことが確認できました。該当のお客様については個別に対応いたしております。

3．対応方法
（1） 10月19日（火）よりクイックログイン機能を停止いたしました。
（2） 10月25日（月）よりクロネコIDの入力のみを省略し、パスワードは入力していただくよう修正して、クイックログイン機能を再開いたしました。
（3） 今後とも継続してシステム的な対応を行ってまいります。

以上

http://www.kuronekoyamato.co.jp/info/info_101025.html

iモードブラウザ2.0の登場

2009年5月19日づけNTTドコモ社の報道発表より

2. iモードブラウザの高機能化、大容量化
   iモードブラウザがさらに大容量化、高機能化し、これまで以上に幅広い、多彩なコンテンツをお楽しみいただけるようになります。
   - ページサイズの拡大
     ページサイズが100KBから500KBに拡大。より表現力豊かでリッチなコンテンツを楽しめるようになります。
   - ページ内動画再生（FLV）対応
     インラインフラッシュ動画に対応し、より手軽に迫力のある映像を楽しめます。
   - JavaScript対応
     JavaScriptに対応し、地図などの表示・操作がより簡単かつ表現力豊かになります。
   - マルチウインドウ対応
     タブブラウザの標準搭載で、複数のサイト間をスムーズに切り替えながら閲覧可能となります。
   - 動画フォーマットの種類増加
     WMV®の動画をiモード上で閲覧することが可能となり、より多くの動画を楽しめます。
   - Cookie、Referer対応
     Cookie、Refererにiモード上で新たに対応することで、各種サイトへのログインの簡略化、サイト遷移の情報収集が図れるため、より便利にサイトをご利用いただけます。

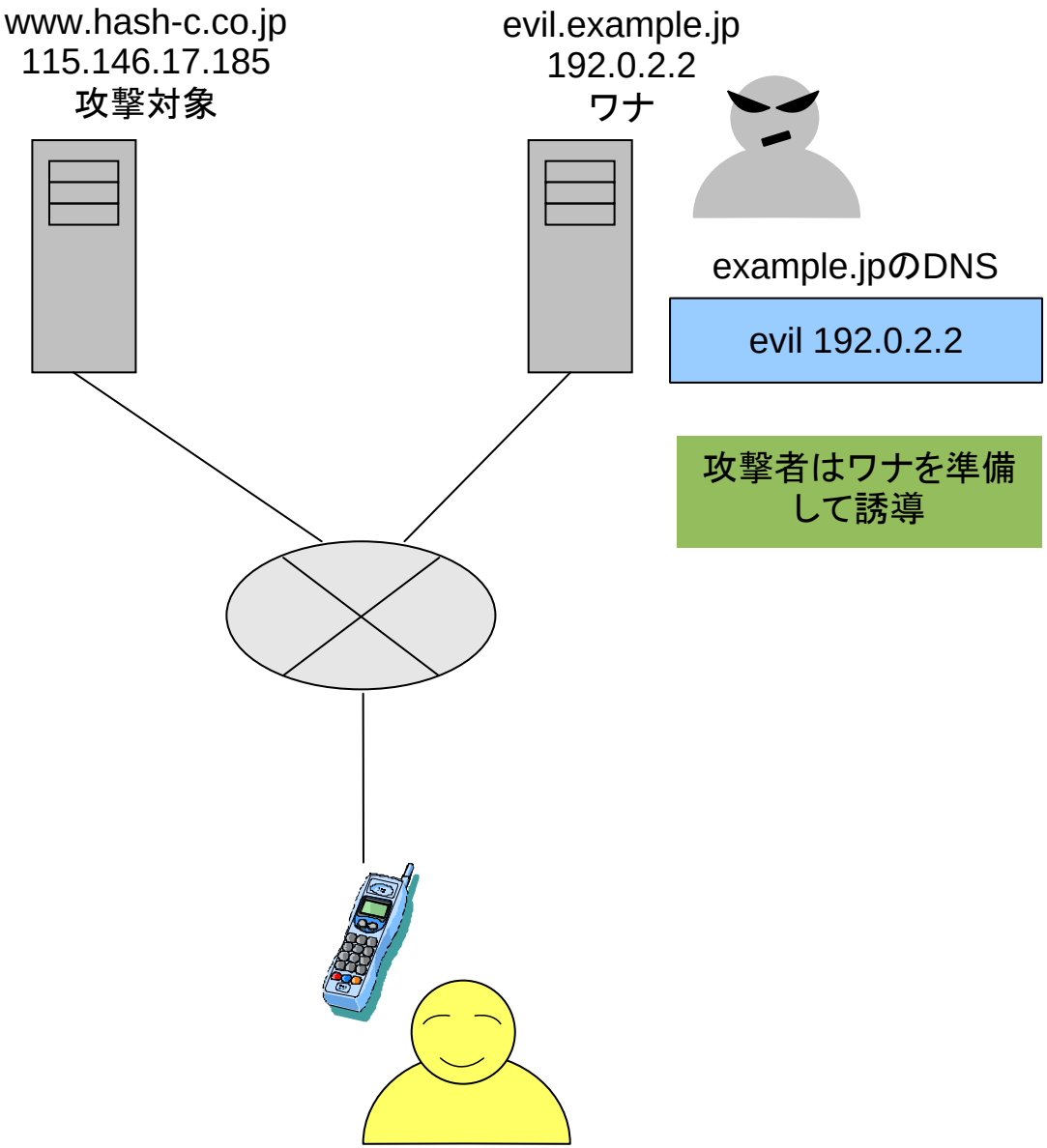
http://www.nttdocomo.co.jp/info/news_release/page/090519_00.html より引用

ケータイJavaScriptで端末固有IDを書き換える条件

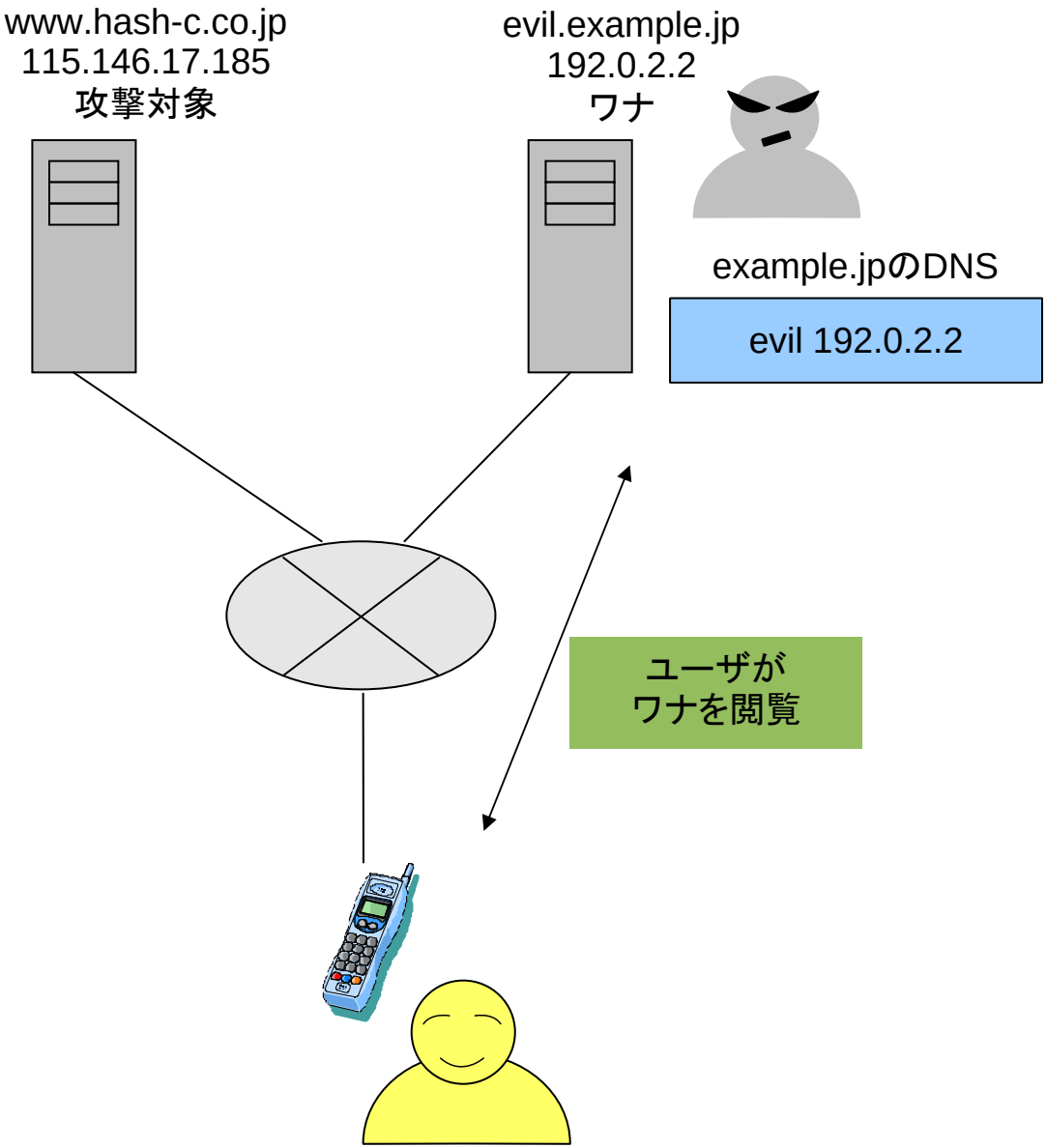
- 以下の三条件がそろえば、端末固有IDの書き換え、すなわち「かんたんログイン」なりすましができるが・・
  a. 携帯電話のJavaScriptでXMLHttpRequestオブジェクトが利用できる
  b. XMLHttpRequestにてsetRequestHeaderメソッドが利用できる
  c. setRequestHeaderメソッドにてUserAgentなどのリクエストヘッダが書き換えできる
- 10月末のJavaScript再有効化の際に、setRequestHeaderメソッドが無効化された模様。すなわち、**上記b、cが成立しなくなった**。
- 元々のNTTドコモのサイトではJavaスクリプトの仕様書にsetRequestHeaderもちゃんと載っていたのだが…現時点では削除されている
- XMLHttpRequest自体にも制限が加わり、JavaScriptが置かれたコンテンツのディレクトリかサブディレクトリのみがアクセス可能

参照: http://www.tokumaru.org/d/20090805.html#p01

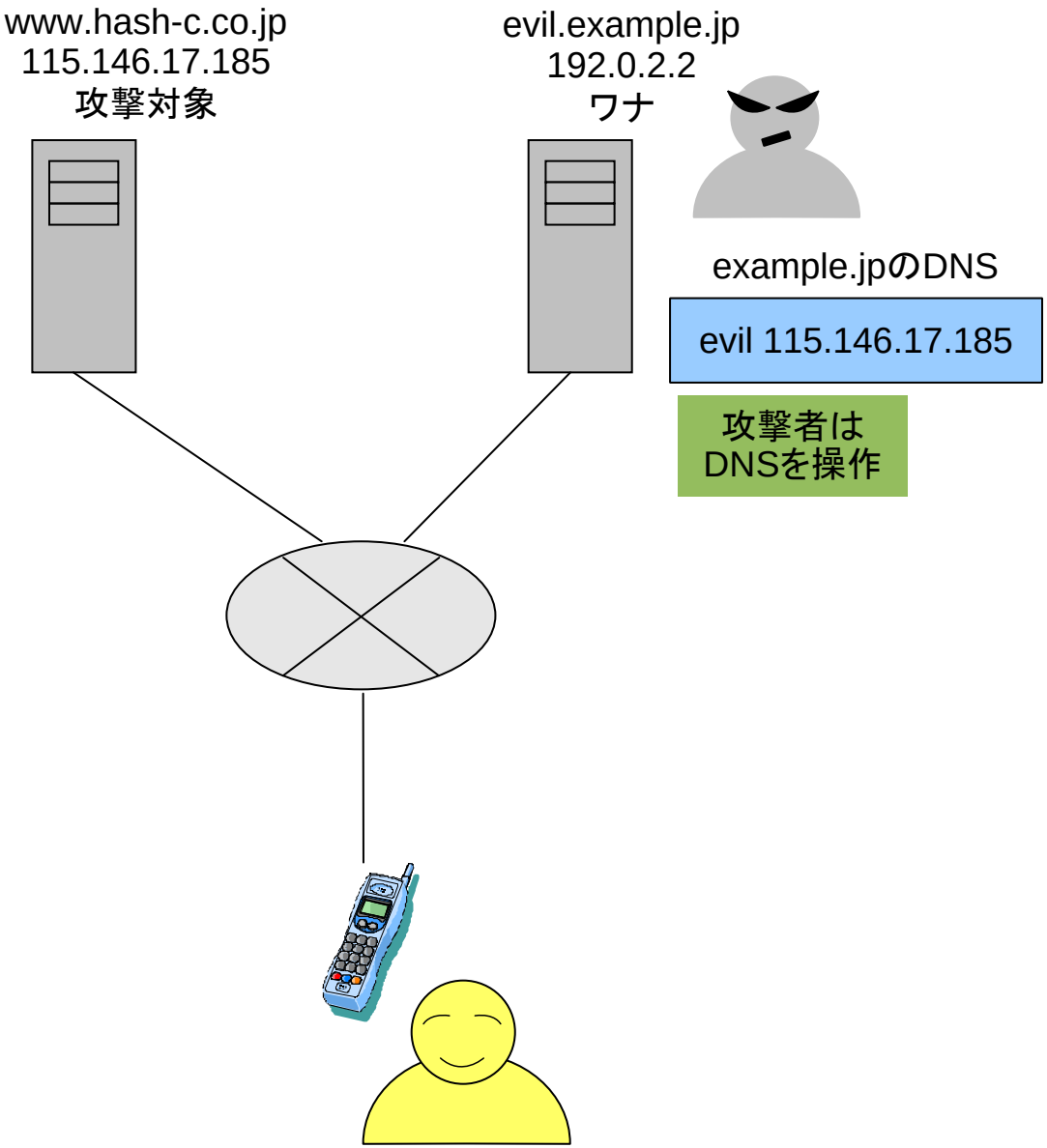
DNS Rebindingによるなりすまし攻撃



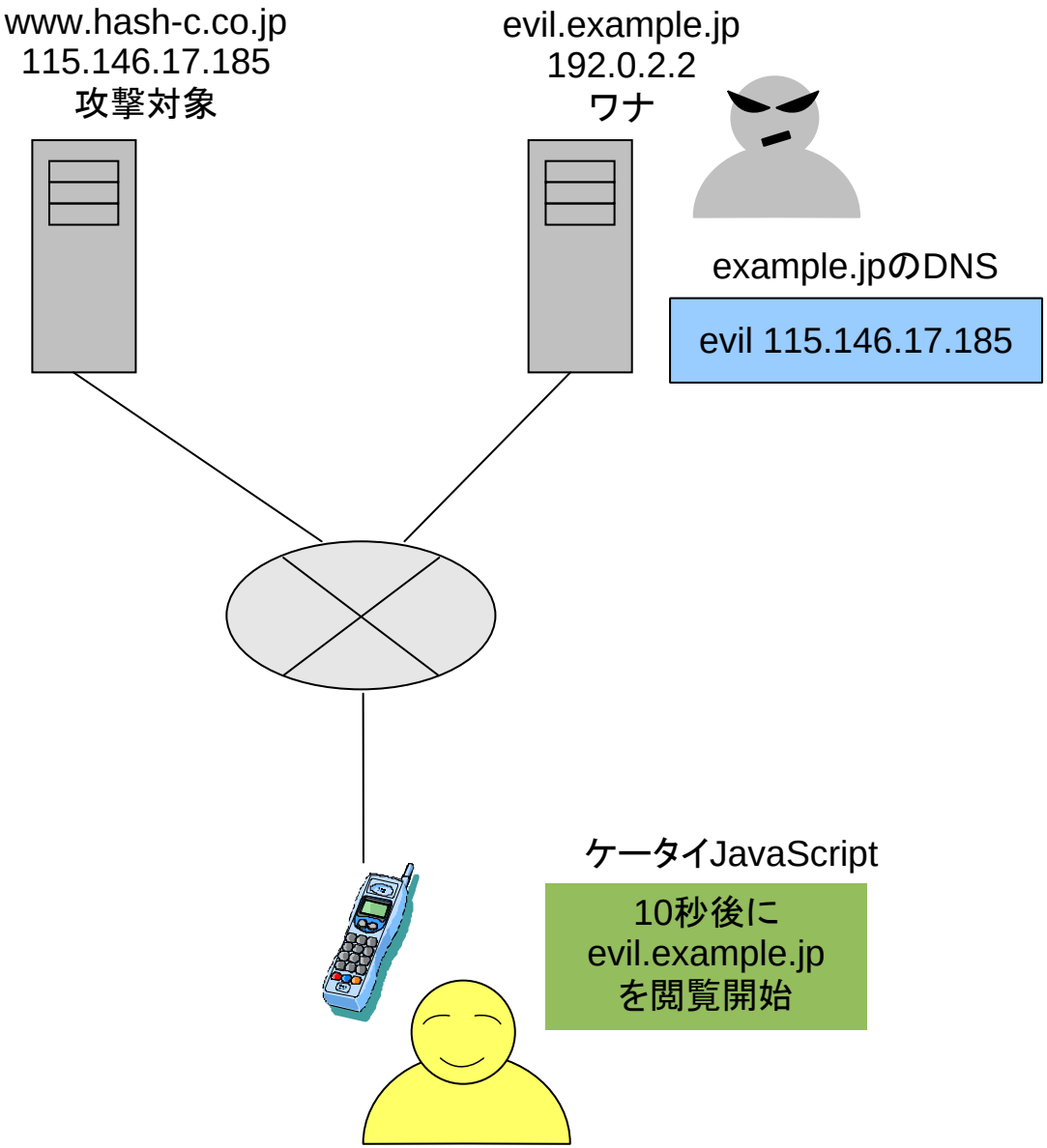
DNS Rebindingによるなりすまし攻撃



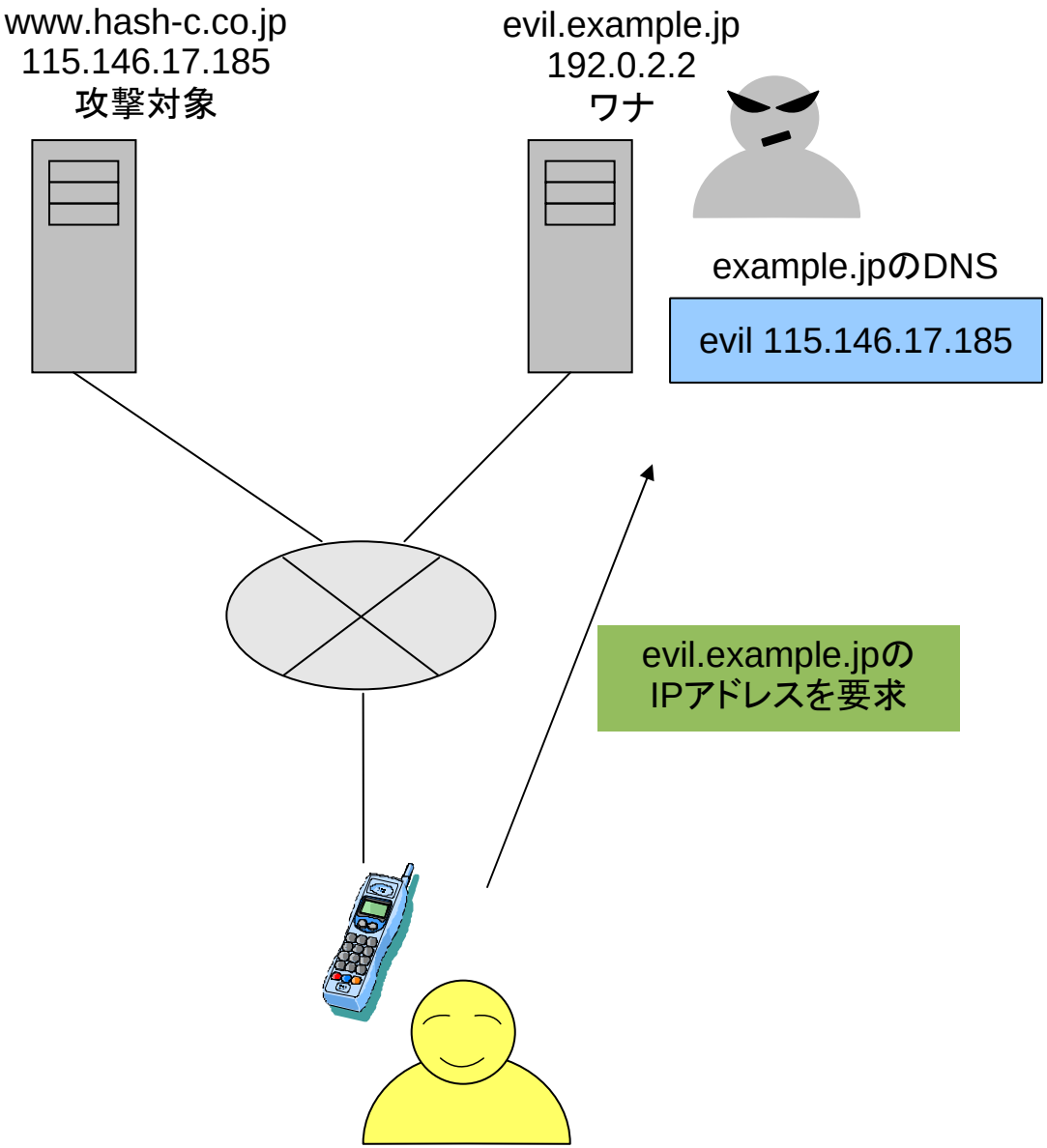
DNS Rebindingによるなりすまし攻撃



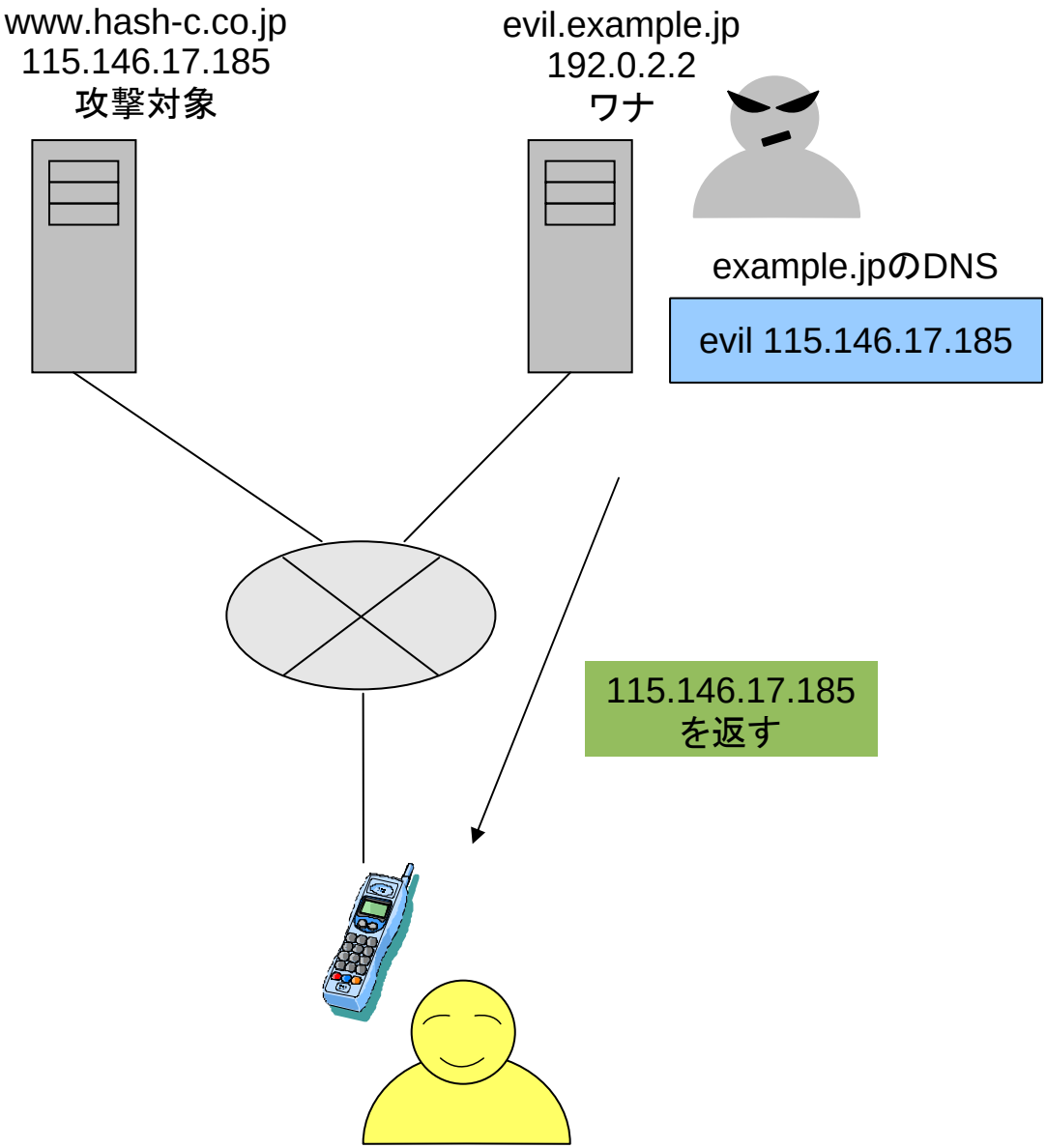
DNS Rebindingによるなりすまし攻撃



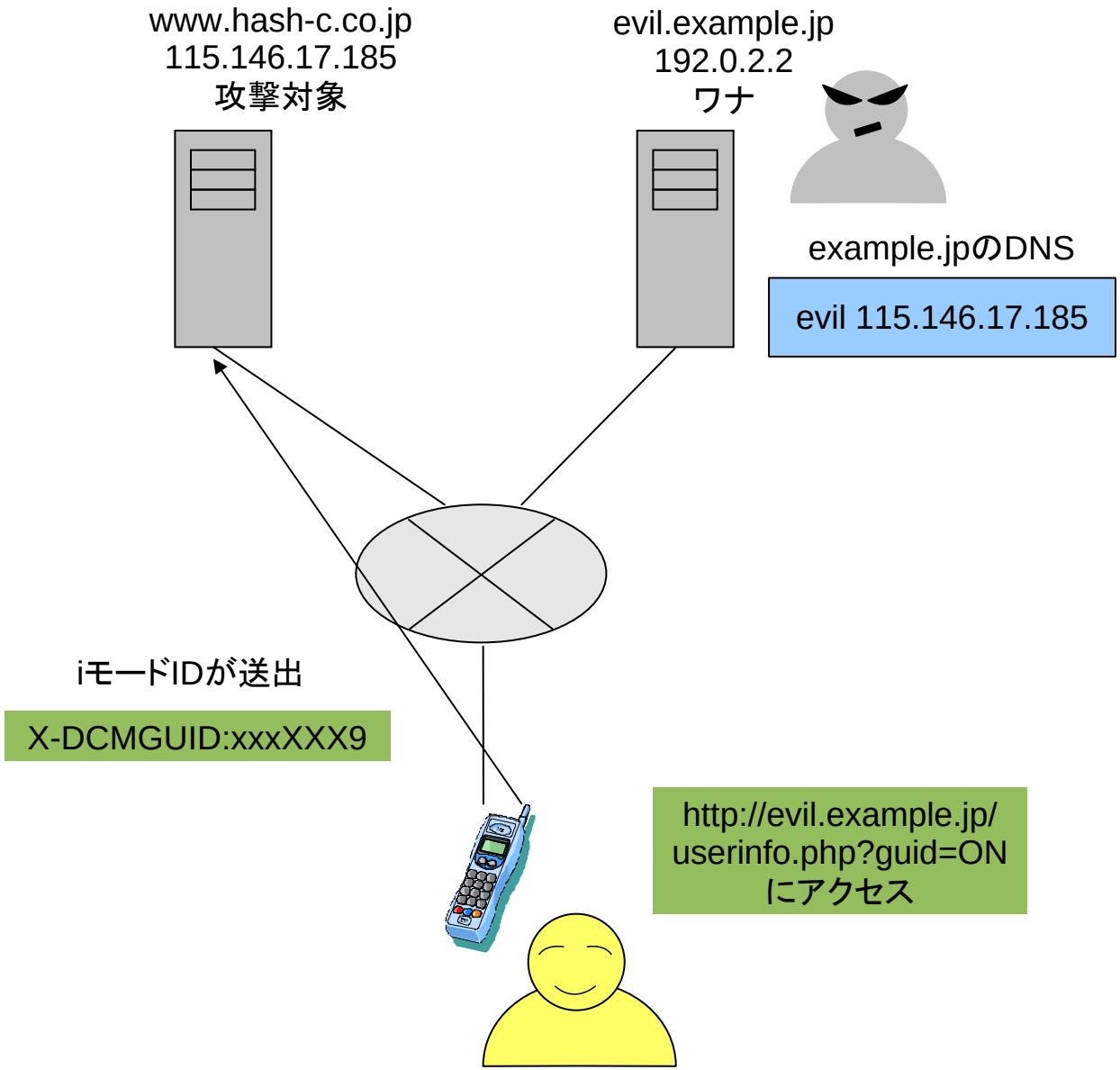
DNS Rebindingによるなりすまし攻撃



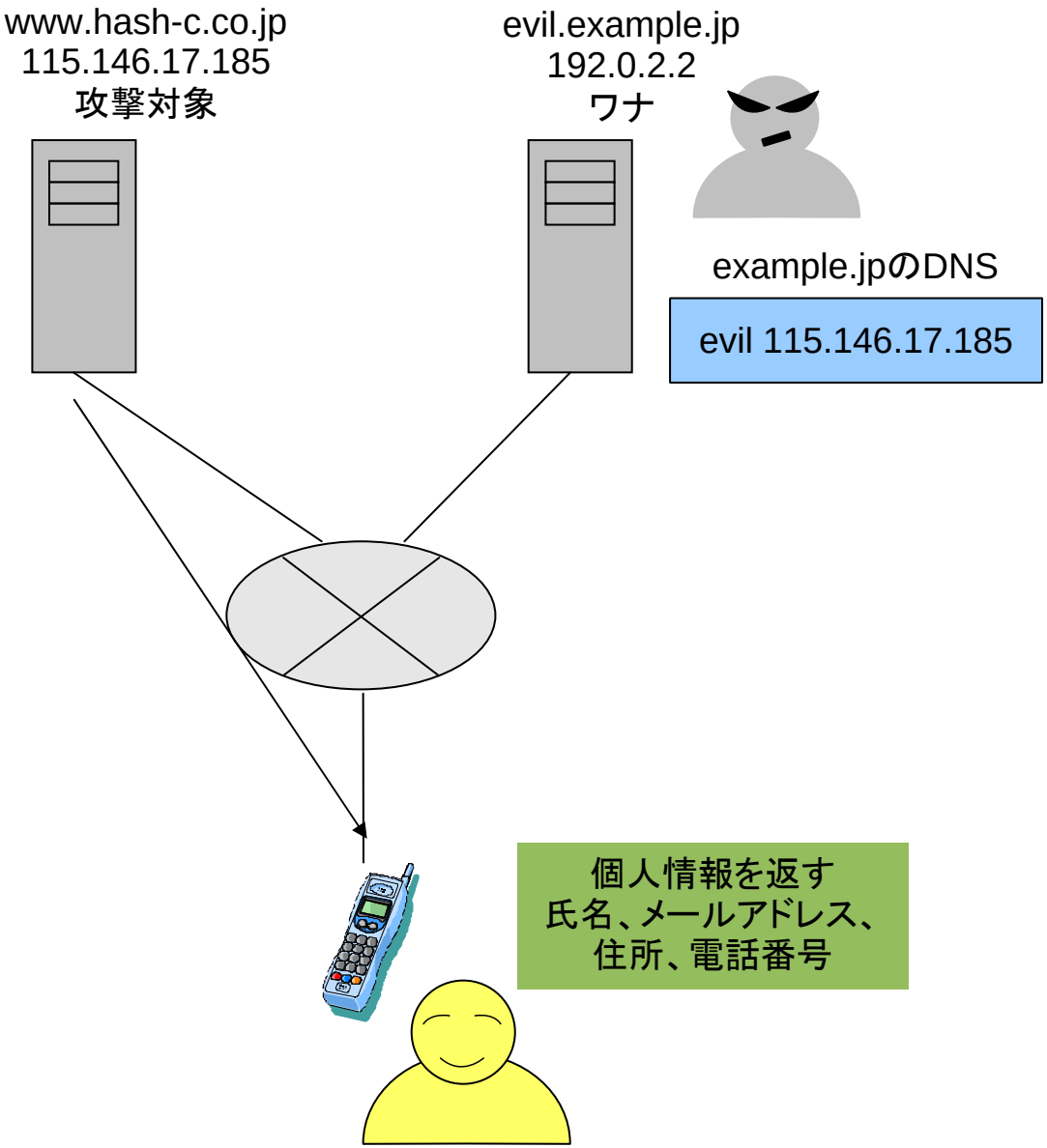
DNS Rebindingによるなりすまし攻撃



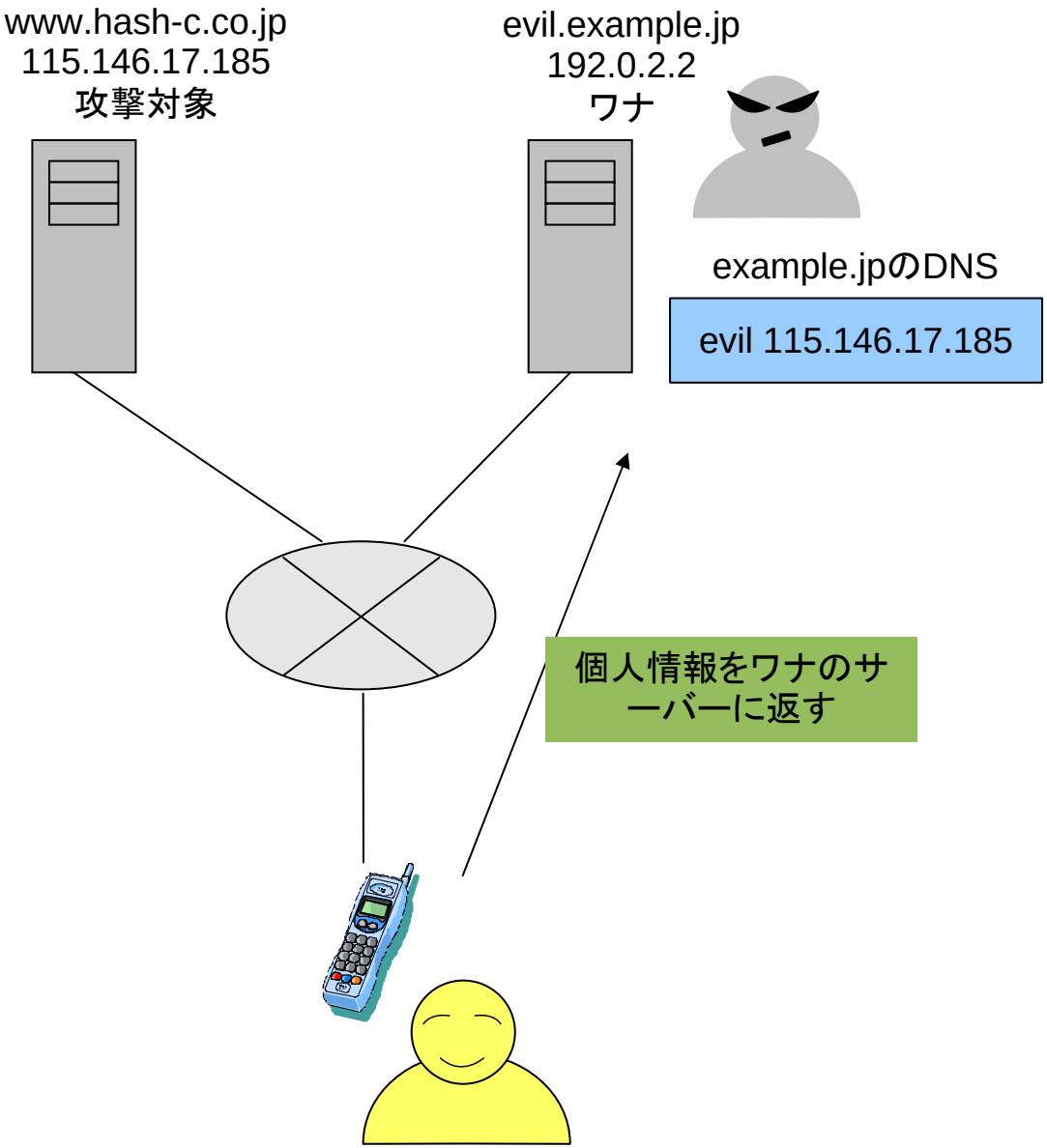
DNS Rebindingによるなりすまし攻撃



DNS Rebindingによるなりすまし攻撃



DNS Rebindingによるなりすまし攻撃



セキュリティ情報

iモードIDを用いた「かんたんログイン」のDNS Rebinding脆弱性

HASHコンサルティング株式会社
公開日:2009年11月24日

概要

iモードブラウザ2.0のJavaScriptとDNS Rebinding(DNSリバインディング)問題の組み合わせにより、iモードIDを利用した認証機能(以下かんたんログイン)に対する不正アクセスが可能となる場合があることを確認したので報告する。危険度の高い攻撃手法であるので、サイト運営者には至急の対策を推奨する。

背景

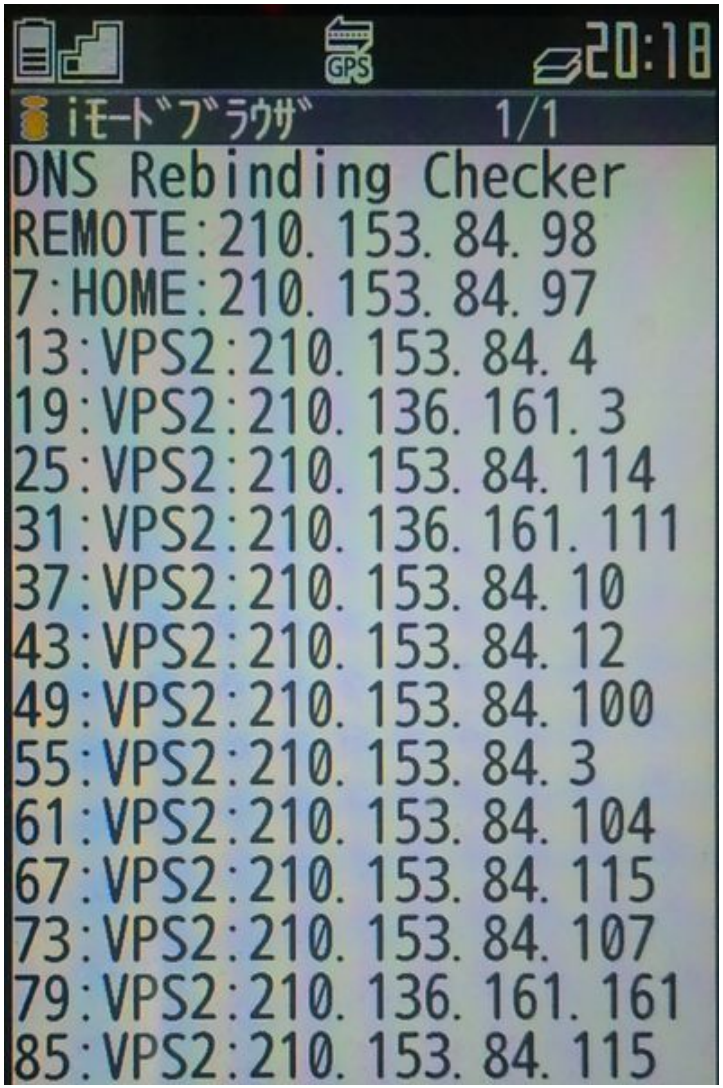
携帯電話のかんたんログインとは、ケータイブラウザ(たとえばiモードブラウザ)に用意された契約者固有IDを利用した簡易的な認証であり、ユーザがIDやパスワードを入力しなくても認証が可能となる。iモードIDは、NTTドコモの提供する契約者固有IDの一種で、URLにguid=ONというクエリストリングを含めることにより、端末固有の7桁のIDがWebサーバに送出される。現在、iモードIDは、ケータイWebアプリケーションに広く利用されている。
2009年5月より、NTTドコモの携帯電話ブラウザ(iモードブラウザ2.0)ではJavaScriptが利用できる。ケータ

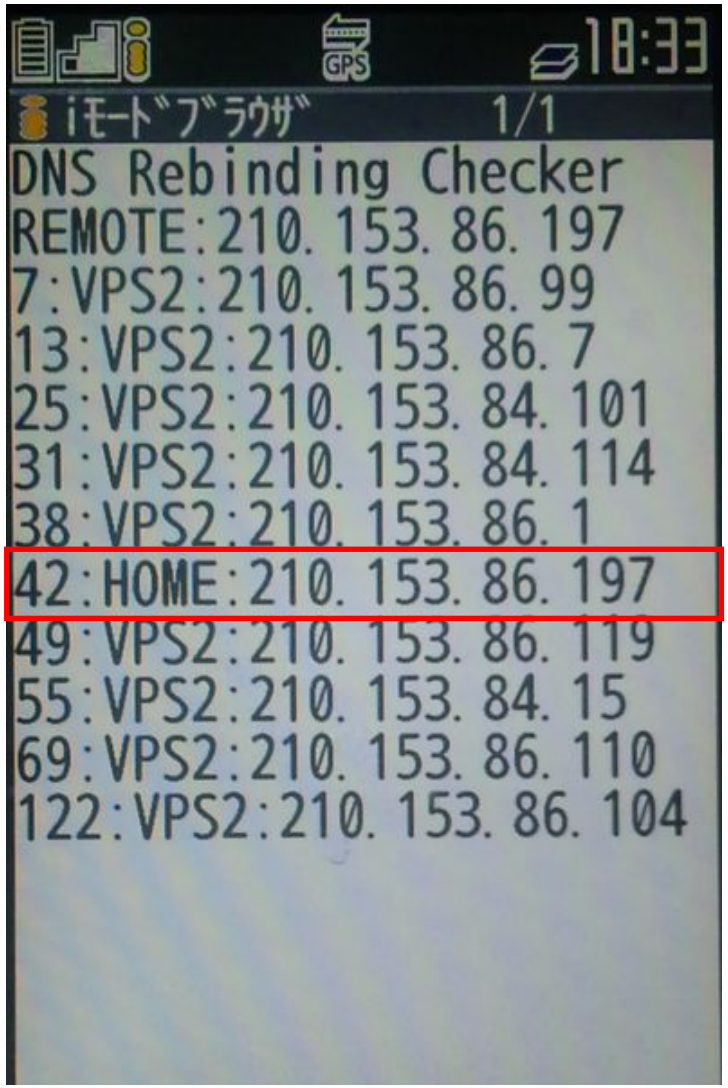
DNSリバインディング対策はケータイ事業者側ではできない

- 本来はケータイブラウザあるいはゲートウェイで対策するべきものではあるが…
- ケータイブラウザはゲートウェイ（PROXY）経由のアクセスなので、ブラウザでは名前解決をしていない。すなわち、ブラウザでは対策（DNS Pinning）できない
- ゲートウェイはマルチユーザが対象なので、文脈を意識したPinningは不可能で、いつかはIPアドレスを切り替えなければならない

- キャリアのDNSが最低1時間DNSキャッシュする想定では、以下のような攻撃が可能となる
- 攻撃者はワナサイトを用意する
- 攻撃者は自ら携帯電話でワナサイトをアクセスする
- ワナサイトのIPアドレスをターゲットサイトのIPに切り替えておく
- 58分後に、ワナサイトのURLをまいて誘導する
- 被害者がワナサイトを閲覧した直後に、DNSサーバーのキャッシュ保持が無効となる
- 被害者のブラウザ上でDNSリバインディングを悪用したリクエストが発行されるが、既に標的サイトのIPアドレスが有効となり、攻撃が成立する

ゲートウェイIPの変遷





Webサイト側でのDNSリバインディング対策

- iモードブラウザ2.0のXMLHttpRequestでは、setRequestHeaderメソッドが無効化されているので、リクエストヘッダのHostフィールドはワナサイトのドメインになっている
- したがって、かんたんログインの際に、Hostフィールドをチェックすればよい
- 簡単に実装するには、名前ベースのバーチャルホストにすればよい

ケータイtwitter(twtr.jp)においてDNS Rebinding攻撃に対する脆弱性を発見・通報し、即座に修正された

twitterのケータイ版twtr.jpにおいて、DNS Rebindingによるなりすましを許す脆弱性が発見され、1/15に通報したところ、その日のうちに修正された。以下、その経緯について報告する。

経緯

今年の1月12日に読売新聞の記事が出たのを受けて、現実のサイトはどうなのだろうかと改めて気になった。

> *NTTドコモの携帯電話のうち、インターネット閲覧ソフト「iモードブラウザ2・0」を搭載した最新29機種を通じて、利用者の個人情報を不正取得される恐れのあることが、専門家の指摘で明らかになった。*
>
> *同社は携帯サイトの運営者にパスワード認証などの安全対策を呼びかけている。携帯電話の機能が高機能化するにつれ、こうした危険は増しており、利用者も注意が必要になってきた。*

[ドコモ携帯、情報流出の恐れ…最新29機種より引用]

私自身も携帯電話でいくつかのサイトを巡回しており、その一つにtwitter.comの日本の携帯電話向けフロントエンドであるtwtr.jpも含まれる。ご存じのように、twitterは政府要人や有名人も数多く使っているし、twitter.comの画面で確認する限り、鳩山由紀夫首相や原口一博総務相などの閣僚の方々は携帯から書き込みをされることも多いようなので、仮に脆弱性があった場合、影響も大きくなると思った。原口総務相に関しては、記者団からの「Twitterの質問を受けて、つぶやきを確認する原口氏」という報道写真も公開されているので、(スタッフ任せではなく)自ら携帯電話を使いこなしてtwtr.jpにアクセスしておられることは間違いないだろう。

手始めにtwtr.jpのIPアドレスを別のドメインにセットして、携帯電話からアクセスすると、正常に画面が表示される(右の写真)。これはまずい。このため、DNS Rebindingを使ってなりすましができてしまわないか調べることにした。当然ながら、不正アクセス禁止法に抵触しないよう、自分のアカウントを使用して確認を行った。

http://www.tokumaru.org/d/20100222.html#p01

2010年02月28日

■ 著名優良サイトでもiモード2.0の脆弱性に対応していなかった。なぜか。

今月中旬のこと。私は2テラのハードディスクを買溜めするため秋葉原の街に出た。しかし、どの店が最安か調べずに出たため、やむなく携帯電話で調べることにし、価格.comのサイトを探した。すると、携帯電話用のサイト m.kakaku.com があり、私は初めてそこを使った。

サイトはとても使いやすく、すぐに意中の製品を見つけることができた。が、ここで、画面に「履歴」というリンクがあることに気づいた。「履歴」の画面に入ると、なんと、閲覧した製品が既に記録されていた。ログインしていないのに。いや、アカウントさえないのに。

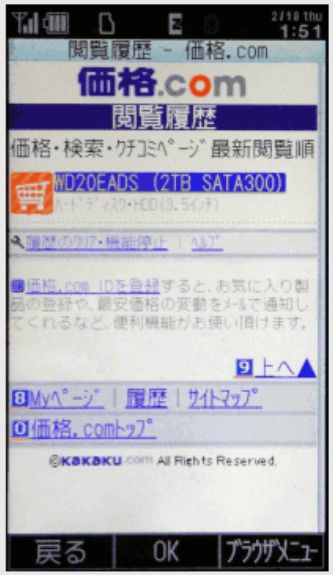


図1: m.kakaku.com の「履歴」機能

これはたしかに便利な機能ではあるが、契約者固有IDを用いて実現されていることにギョッとした。同様の機能は普通のPCのインターネットでもcookieを使って実現できるわけだけども、契約者固有IDの取得と保管は、個人情報の取得と同程度に慎重さが求められるべきと考えている私にとっては、予告と同意なく記録されたことにギョッとした*1。そして、契約者固有IDがもはやこんな用途にまで使われ始めてしまっていることに落胆した。（加えて言えば、契約者固有IDはちゃんとハッシュ化（鍵付きで）してから使っているのだろうかといった疑問もわくが、これは今回のテーマではない。）

http://takagi-hiromitsu.jp/diary/20100228.html#p01 より引用

ソフトバンクでもJavaScriptに対応

ケータイ Watch

記事検索

検索

最新ニュースIndex

- 「F906i」「N906iL」に不具合、ソフト更新開始
- BIGLOBEが「andronavi」刷新、有料アプリ提供も
- ローソンやドコモなど出資のデジタルサイネージ「東京メディア」
- ダイジェストニュース（2010年5月19日）
- パピレス、iPad対応の電子書籍配信サービス「電子貸本Renta!」

Yahoo!ケータイのブラウザが仕様改定、Ajaxなどに対応

ソフトバンクモバイルは、Yahoo!ケータイのブラウザに新機能を搭載する。2010年夏モデルにおける対象機種は944SH、945SH（945SH G）で、Ajaxなどに対応する。

今回のブラウザ（ACCESS製）の仕様改定では、1回当たりのファイル読み込みサイズが300KBから、最大500KBへ拡張される。また、Flash Lite 3.0をサポートし、ページ内での戦略Flash動画再生が可能になる。JavaScriptにも対応し、いわゆるAjaxで構築されたページが利用できるようになる。また、タブブラウザ機能も用意され、テキストで構成されたリンクからタブで新しいページを開いてアクセスすることもできる。

http://k-tai.impress.co.jp/docs/news/20100518_367787.html より引用

本当に2010年夏モデルからなのか?

- 実はかなり以前からソフトバンク端末の一部のモデルでJavaScriptに対応していた
- ノキア 702NK(2004年12月発売)では簡単なJavaScriptに対応
  - XMLHttpRequestやIFRAME、DOMには対応していない
- 804SS(2006年3月)、910T(2006年10月)、910SH(2006年11月)では、NetFront 3.3によるJavaScript対応
  - XMLHttpRequestには対応していない
  - IFRAME、DOMに対応…攻撃に利用できる可能性*1
- 922SH(2008年3月)では、NetFront 3.4にてAjaxに対応
  - XMLHttpRequestのサポート
  - setRequetHeaderのサポート (!)
- 独自調査の結果、以下5機種は、デフォルトでAjaxが有効
  - 820N, 821N, 831N, 830CA, 940SC

Ajax有効な機種のサマリ

発売時期	SHARP		NEC		CASIO		Samsung	
2007冬以前	なし	多数	なし	複数	該当なし		なし	多数
2008春	△	922SH	該当なし		該当なし		なし	820SC
2008夏	△	923SH	あり	820N/821N	該当なし		該当なし	
2008冬	△	930SH	該当なし		あり	830CA	なし	930SC
2009春	△	932SH	なし	830N	なし	930CA	なし	731SC
2009夏	△	933SH	なし	931N	該当なし		なし	931SC
2009冬 － 2010春	△	943SH	なし	940N	該当なし		あり	940SC

- △はデフォルトでAjax無効、オプションにより有効化可能（SHARP端末は多いので抜粋）
- 上記以外のPanasonic、Toshiba等はAjax無効

ブラウザのスクリプト設定について

2010年5月27日

平素はソフトバンクをご利用いただき、誠にありがとうございます。
弊社携帯電話の下記の機種にて、Yahoo!ケータイまたはPCサイトブラウザご使用時の設定の一つであるスクリプト設定(JavaScript)を有効にした状態で悪意あるサイトを閲覧した場合に、情報の詐取が生じる可能性があることが確認されました。
お客さまにおかれましては、安全の為、携帯電話のスクリプト設定をOFFに変更していただけますようお願い申し上げます。
なお、今後発売されるスクリプト対応の機種においては、よりセキュリティ機能を強化しておりますが、インターネットにはさまざまな悪意あるサイトの存在があり、新しい攻撃手法が発生しておりますため、インターネットへのアクセスの際にはサイトの信頼性をご確認の上、ご利用いただけますようお願い申し上げます。
本件に関しまして、お客さまにはお手数をおかけしますことを深くお詫び申し上げます。弊社では今後もより良い商品・サービスのご提供に努力する所存でございますので、引き続きご愛顧をお願い申し上げます。

スクリプト設定の変更は下記をご参照ください。

■Yahoo!ケータイの場合
「Yahoo!ケータイ設定」→「セキュリティ設定」→「スクリプト設定」
■PCサイトブラウザの場合
「PCサイトブラウザ」→「PCサイトブラウザ設定」→「セキュリティ設定」→「スクリプト設定」

「かんたんログイン」は、ギリギリの瀬戸際に立たされている

- かんたんログインに対して、ケータイ　JavaScriptによる攻略手法がわかってきた
- かんたんログインに際しての必須対策
  - キャリアゲートウェイのIPアドレスとのみ通信を制限する
  - キャリアの判定にもIPアドレスを用いる
  - Hostヘッダのチェック（あるいは名前ベースのバーチャルホスト）
  - ソフトバンク端末については、Ajax規制あるいはスクリプトの禁止
  - SSLでは、かんたんログインは行わない
  - …
- これで完璧かどうかは誰にも分からない
  - とくに、ソフトバンク端末のように機種依存性が多いと、全ての端末について検証しないと確かなことは言えない
- それでも、まだ、かんたんログイン続けますか？

能動的攻撃の可能性

ケータイJavaScriptによる能動的攻撃の可能性

- iモードブラウザ2.0はsetRequestHeader()が無効化されているので、User-AgentやX-DCM-GUIDの変更は不可能と思われる
- ソフトバンク端末では、setRequestHeader()関数自体は無効化されていないが、User-AgentやX-JPHONE-UIDの変更は禁止されている
- ・・・
- 抜け道はないのか？

2種類のトリックによる端末固有IDの改変が可能

- トリック1：End-EndのSSLを使う
  - キャリアゲートウェイのチェックをすり抜ける
  - ソフトバンクにはゲートウェイ型のSSLもあるが、こちらはゲートウェイのチェックが有効（2011年2月廃止予定）
- トリック2：ハイフン「-」の代わりにアンダースコア「_」を用いる
  - リクエストヘッダ中のハイフンは、アプリケーションが利用する際にアンダースコアに変更される仕様を悪用
  - アンダースコアはゲートウェイだとチェックされるが、端末のチェックはすり抜ける

SSLを利用したリクエストヘッダ改変スクリプトの例

```
var requester = new XMLHttpRequest();
requester.open('GET', 'https://example.com/login.php', true);
requester.onreadystatechange = function() {
  if (requester.readyState == 4) {
    onloaded(requester);
  }
};
requester.setRequestHeader("Host", "twtr.com");
requester.setRequestHeader("User_Agent",
  "SoftBank/1.0/943SH/SHJ001/SN359XXXXXXXXXX0 "  +
  " Browser/NetFront/3.5 Profile/MIDP-2.0 Configuration/CLDC-1.1");
requester.setRequestHeader(
    "X_JPHONE_UID", "a3XXXXXXXXXXXXP");
requester.send(null);
```

アプリケーションが受け取ったリクエストの例（一部）

HTTP_USER_AGENT=DoCoMo/2.1 P07A3(c500;TB;W24H15)Fake
SERVER_NAME=twitter.com
SERVER_PORT=443
HTTP_COOKIE=aaa=bbbx
REMOTE_ADDR=123.108.237.4
SERVER_PROTOCOL=HTTP/1.1
HTTP_X_JPHONE_UID=fakejphoneuid
HTTP_X_DCMGUID=fakedcmguid
HTTP_X_UP_SUBNO=fakesubno
HTTP_HOST=twitter.com

※URLと証明書のドメインが異なるので警告が出るが、処理は続行可能

SSLを悪用した能動型なりすまし攻撃への対策

- SSLではかんたんログインを受け付けない
  - 必須はソフトバンクのみ禁止だが、全キャリア禁止することが無難
- キャリア判定は、User-Agentではなく、IPアドレスで行う

セッション管理の問題

ケータイWebアプリのセッション管理手法

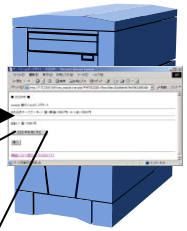
- ケータイWebアプリでセッションIDを保持する場所はいくつか候補がある
  - URL埋め込み
    - 現在の主流
    - しかし、色々と注意点がある
  - Cookie
    - iモードブラウザがCookie非対応だったのであまり使われなかった
    - 今後iモードもCookie対応になるので、中・長期的にはCookieへの移行が望ましい
  - 端末固有IDそのものをセッションIDとして使用する方法
    - 最近増えつつあるようだが
    - 前述の理由でやめた方がよい
    - SSLに対応できない
    - 一般的なセキュリティ対策が使えない場合がある

セッションIDをURLに埋め込む場合の問題点

- URLに埋め込まれたセッションIDがReferrer経由で漏洩する
- セッションフィクセイション攻撃の可能性
- ユーザがURLをメールなどで教えてしまう(!)

URLに埋め込まれたセッションIDがReferrer経由で漏洩する

http://hogehoge.jp



WebサイトAにアクセス

http://hogehoge.jp/item.jsp;JSESSIONID=12345ABCDEF

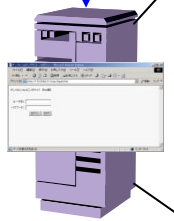
WebサイトBへのリンク
honya

リンクをクリック

http://honya.jp/l

WebサイトBのログに、
別サイトのセッションIDが記録される

なりすましの危険性



WebサイトB
http://honya.jp/

【アクセスログ】
…
…
http://hogehoge.jp/item.jsp;JSESSIONID=12345ABCDEF
…
…

RefererによるセッションID漏洩

- そもそもケータイブラウザではRefererは送出されるのか?
  - iモードでは送出されないことになっていたが・・・
    - iモードブラウザ2.0では送出されるようになった
  - EZwebでは送出することになっているが・・・
  - Yahoo! ケータイでは送出することになっているが・・・
- キャリアごとの端末世代依存、機種依存もある
  - 端末のバグという話もあるが、このレベルのバグは改修されないことが多い
- ケータイWebアプリでは、URLにセッションIDを埋め込む実装が多く用いられるので、RefererからのセッションID漏洩には注意が必要
- キャリアごとに違いにより、テスト漏れが発生しやすい
- 結局、細かいことを考えずに粛々とReferer対策をしておくのが一番安全

セッションフィクセイション攻撃の可能性

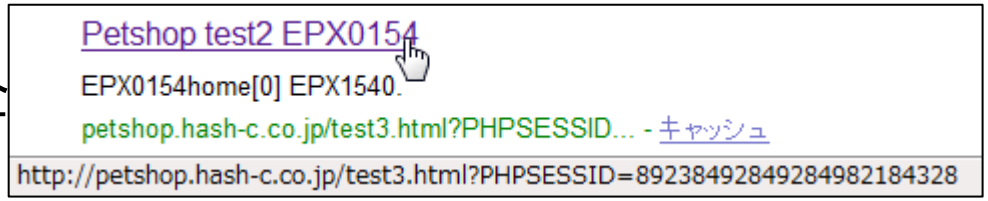
- 攻撃者がセッションIDを取得する
- セッションIDつきのURLをメールなどで、正規ユーザ(被害者)に送りつけ、巧妙なメッセージでURLを実行させる
- 正規ユーザが同URLをアクセスすると、攻撃者は同じセッションを共有できることになる。

ユーザがURLをメールなどで教えてしまう(!)

- セッションフィクセイションと似ているが・・・
- セッションIDつきのURLを正規ユーザが知人などにメールで知らせてしまうorz
- 自爆型セッションフィクセイション

セッションフィクセイション対策

- 一般的なセッションフィクセイション対策は必須
  - ログイン後にセッションIDを振り直す
  - PHPの場合は session_regenerate_id() が使える
- ログイン前には極力セッションを使用しない
  - hiddenの方が安全
  - ログイン前のセッションは、全
- 様々な「保険的対策」
  - 検索エンジンにはセッションIDつきのURLが保存されないように注意
  - SNSなどでは、ユーザが自分のページのURLをセッションIDつきで添付しないようにチェックする
  - Refererチェックにより外部からリンクされた場合はセッションをリセットする
  - セッションタイムアウトを短めに設定する / 明示的なログアウト
- Cookieへの移行を真剣に検討する



スマートフォンのセキュリティ（グローバルな話題）

スマートフォンのWebアプリケーションセキュリティ

- PC向けのWebアプリケーションセキュリティと同じ
- 普通にセキュリティ対策する
  - SQLインジェクション
  - クロスサイト・スクリプティング
  - クロスサイト・リクエストフォージェリ
  - …
- いわゆる「かんたんログイン」は実装してはいけない
  - もっとも、iPhoneやAndroidのブラウザは端末固有IDを送出しないので、やりたくてもできないが…
  - アプリはどうか?

クライアントアプリのセキュリティ脅威

- 端末の紛失・盗難を気にする人が多いが
  - 端末が他人の手に渡ったら、いくらでも時間を掛けて解析できるので、端末上のデータは守れないと考えた方がよい
  - 重要なデータはサーバー（クラウド）に
- サーバーとの通信にはHTTP/HTTPSが広く利用される
  - 基本的にはWebアプリケーションと同じ脅威
  - SQLインジェクションなど…
  - XSS、CSRFなどブラウザ経由で起こる問題は発生しない
    - ただし、アプリからブラウザを起動している場合などは例外
- 認証に注意
  - 端末認証によりパスワードなしで使えるアプリが多い
  - 認証方式に注意
  - 端末の紛失・盗難時に、速やかにアカウントロックができるようにサーバー側で考慮を

クライアントアプリのセキュリティ

- 重要な情報はサーバー上に保持する
- Webアプリケーションと共通のセキュリティ対策
- 認証がカギ
- 「見えないこと」に頼ったセキュリティは脆弱
  - スマートフォンの場合、Wi-Fiパケットは簡単にキャプチャできる
  - アプリのリバースエンジニアリングの可能性
- クライアントアプリなら端末固有IDが取得できるが
  - iPhone: ICCID、UDIDなど
  - Android: ANDROID_ID、IMEI、SIMシリアル番号など
  - いずれも認証に使ってはいけない

AT&Tのウェブサイトから「iPad」ユーザーのデータが大量流出

- ブログネットワークのGawker Mediaは米国時間6月9日、AT&Tのウェブサイトから「iPad Wi-Fi ＋ 3G」ユーザー約11万4000人分の電子メールアドレス情報が流出したと報じた。漏えいしたデータには、米国政府の高官や映画監督、企業の最高経営責任者（CEO）のものも含まれているようだ。
- 同報道によると、Goatse Securityと名乗るハッカーグループが、AT&Tのウェブサイトから、iPadのSIMカードのシリアル番号を含んだHTTPリクエストを送信し、電子メールアドレスを入手したという。このICCIDと呼ばれるシリアル番号は順番に付けられているため、容易に推測できるとしている。
- AT&Tの広報担当者は米CNETに対しデータ流出があったことを認め、同社は7日、Goatse Securityとは関係しない人物からiPadのICCIDのデータ漏えいの可能性について報告を受け、その翌日に電子メールアドレスを提供する機能を停止したと述べた。また、ICCIDから引き出せる情報は、電子メールアドレスだけであり、同社は調査を継続するとともに、情報の流出について顧客に報告していくと発表した。Goatse SecurityとAppleにもコメントを求めたが、返答は得られていない。

http://itpro.nikkeibp.co.jp/article/MAG/20100611/349088/ より引用

「電波チェッカー」の事例

電波チェッカー

屋外の電波状況を確認、圏外で使えなかったお店などの場所を登録することによって
ソフトバンクへ電波改善要望を簡単に伝えることができるiPhone用のアプリケーション

登録した電波状況はサーバに保存されるので、登録したiPhone上で確認ができます。
なお、計測したデータが圏外等で送信できなかった場合、データはiPhone端末に一時保存されアプリケーション起動中に圏内に入った時、もしくは次回アプリケーション起動時に自動で送信します。

ソフトバンクではエリア改善・品質向上に努めてまいりますので、ぜひとも圏外エリア情報の収集にご協力ください。

お知らせ

不正プログラムを使用することで、別のiPhoneで登録した計測結果（電波状況とその位置情報）が地図に表示される可能性がありますため、情報の保護を第一として安全性が確認するまで間、登録いただいた情報の表示を停止させていただきます。

- 引き続き、「屋外計測」および「圏外登録」を行っていただけます。
  ご不便をお掛けしますが、何卒、ご理解賜りますようお願い申し上げます。

■このアプリケーションは端末識別のためUDID情報を使用します。

対策

ケータイWebアプリの対策

- 認証とセッション管理が最大の課題
  - いずれもcookieを使えば解決
- Webアプリケーションの一般原則として、セキュアな作りにする
  - XSS、SQLインジェクション対策などをふつーに実装する
  - 「これはケータイではできないはず」などと手を抜かない
  - ケータイ固有の作法を極力使わない。cookieが使えればcookieを使う
- ケータイ固有の特性をよく勉強する
  - キャリアとの契約により情報を入手する
  - 地道に技術解説書を読む
  - ネット等のセキュリティ勧告を地道に追う

ご清聴ありがとうございました